0. python imports

In [1]:
```
from matplotlib import pyplot as plt
import numpy as np
import pandas as pd
from scipy.stats import f_oneway
from scipy.stats import linregress
import statsmodels.api as sm
from statsmodels.formula.api import ols # ordinary least squares
```

1. data loading

In [20]:
```
rate = pd.read_csv('./data/rate_by_city.csv')
rate.head()
```

Out[20]:

	Rate	City
0	13.75	1
1	13.75	1
2	13.50	1
3	13.50	1
4	13.00	1

In [3]:
```
rate.shape
```

Out[3]: (54, 2)

In [4]:
```
auto = pd.read_csv('./data/auto-mpg.csv')
auto.head() # mpg stands for miles per gallon, or fuel consumption
```

Out[4]:

	mpg	cylinders	displacement	horse_power	weight	acceleration	model_year	car_name
0	18.0	8	307.0	130.0	3504	12.0	70	\t"chevrolet chevelle malibu"
1	15.0	8	350.0	165.0	3693	11.5	70	\t"buick skylark 320"
2	18.0	8	318.0	150.0	3436	11.0	70	\t"plymouth satellite"
3	16.0	8	304.0	150.0	3433	12.0	70	\t"amc rebel sst"
4	17.0	8	302.0	140.0	3449	10.5	70	\t"ford torino"

2. anova using scipy

In [14]:
```
rate['city_count'] = rate.groupby('City').cumcount()
rate_pivot = rate.pivot(columns='City', values='Rate', index='city_count')
rate_pivot
```

Out[14]:

City	1	2	3	4	5	6
city_count						
0	13.75	14.25	14.00	15.00	14.50	13.50
1	13.75	13.00	14.00	14.00	14.00	12.25
2	13.50	12.75	13.51	13.75	14.00	12.25
3	13.50	12.50	13.50	13.59	13.90	12.00
4	13.00	12.50	13.50	13.25	13.75	12.00
5	13.00	12.40	13.25	12.97	13.25	12.00
6	13.00	12.30	13.00	12.50	13.00	12.00
7	12.75	11.90	12.50	12.25	12.50	11.90
8	12.50	11.90	12.50	11.89	12.45	11.90

https://nikgrozev.com/2015/07/01/reshaping-in-pandas-pivot-pivot-table-stack-and-unstack-explained-with-pictures/ (https://nikgrozev.com/2015/07/01/reshaping-in-pandas-pivot-pivot-table-stack-and-unstack-explained-with-pictures/)

In [13]:
```
rate_pivot_b = pd.DataFrame({f'city_{i}': rate.loc[rate['City'] == i, 'Rate
'].values
                             for i in rate['City'].unique()})
rate_pivot_b
```

Out[13]:

	city_1	city_2	city_3	city_4	city_5	city_6
0	13.75	14.25	14.00	15.00	14.50	13.50
1	13.75	13.00	14.00	14.00	14.00	12.25
2	13.50	12.75	13.51	13.75	14.00	12.25
3	13.50	12.50	13.50	13.59	13.90	12.00
4	13.00	12.50	13.50	13.25	13.75	12.00
5	13.00	12.40	13.25	12.97	13.25	12.00
6	13.00	12.30	13.00	12.50	13.00	12.00
7	12.75	11.90	12.50	12.25	12.50	11.90
8	12.50	11.90	12.50	11.89	12.45	11.90

In [36]:
```
f_oneway(*[rate_pivot[i] for i in range(1,7)])
```

Out[36]:
```
F_onewayResult(statistic=4.8293848737024, pvalue=0.001174551414504048)
```

3. anova using statsmodels

- https://stackoverflow.com/questions/30650257/ols-using-statsmodel-formula-api-versus-statsmodel-ap (https://stackoverflow.com/questions/30650257/ols-using-statsmodel-formula-api-versus-statsmodel-ap)
- http://www.statsmodels.org/dev/example_formulas.html (http://www.statsmodels.org/dev/example_formulas.html)
- C(variable) is for categorical variables

In [34]:
```
formula = 'Rate ~ C(City)'
model = ols(formula=formula, data=rate).fit()
```

In [39]:
```
anova_table = sm.stats.anova_lm(model, typ='II')
anova_table
```

Out[39]:

	sum_sq	df	F	PR(>F)
C(City)	10.945667	5.0	4.829385	0.001175
Residual	21.758133	48.0	NaN	NaN

4. linear regression using scipy

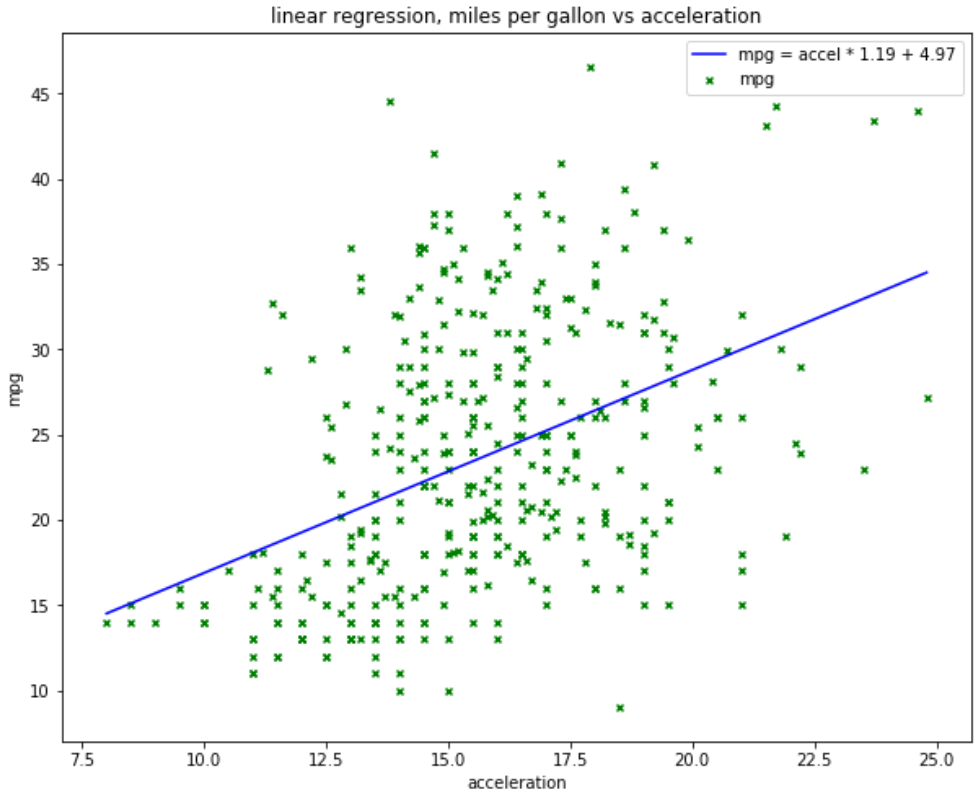
In [55]:
```
reg_result = linregress(auto['acceleration'], auto['mpg'])
reg_names = ['slope', 'intercept', 'r_value', 'p_value', 'std_err']
print('\n'.join([f'{reg_names[i]}: {reg_result[i]}'
                 for i in range(len(reg_result))]))
```

```
slope: 1.1912045293502274
intercept: 4.9697930042539085
r_value: 0.4202889121016507
p_value: 1.8230915350787203e-18
std_err: 0.12923643283101396
```

In [59]:
```
x = np.linspace(auto['acceleration'].min(), auto['acceleration'].max(), 50)
y = reg_result[1] + reg_result[0] * x
```

Plot results:

In [92]:
```
plt.figure(figsize=(10, 8))
plt.plot(x, y, c='b',
         label=f'mpg = accel * {reg_result[0]:.2f} + {reg_result[1]:.2f}')
plt.scatter(auto['acceleration'],
            auto['mpg'],
            marker='x',
            c='g',
            s=16,
            label='mpg')
plt.title('linear regression, miles per gallon vs acceleration')
plt.xlabel('acceleration')
plt.ylabel('mpg')
plt.legend()
plt.show()
```



5. linear regression using statsmodels

for only one variable (acceleration)

In [97]:
```
Y = auto['mpg']
X = sm.add_constant(auto['acceleration'])

model = sm.OLS(Y, X).fit() # ordinary least squares
predictions = model.predict(X)
```

In [100]:
```
model.summary()
```

Out[100]:

OLS Regression Results

Dep. Variable:	mpg	**R-squared:**	0.177
Model:	OLS	**Adj. R-squared:**	0.175
Method:	Least Squares	**F-statistic:**	84.96
Date:	Sun, 14 Jul 2019	**Prob (F-statistic):**	1.82e-18
Time:	13:22:59	**Log-Likelihood:**	-1343.9
No. Observations:	398	**AIC:**	2692.
Df Residuals:	396	**BIC:**	2700.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	4.9698	2.043	2.432	0.015	0.953	8.987
acceleration	1.1912	0.129	9.217	0.000	0.937	1.445

Omnibus:	17.459	**Durbin-Watson:**	0.677
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	18.214
Skew:	0.497	**Prob(JB):**	0.000111
Kurtosis:	2.670	**Cond. No.**	91.1

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

for several variables

In [104]:
```
auto.columns
```

Out[104]:
```
Index(['mpg', 'cylinders', 'displacement', 'horse_power', 'weight',
       'acceleration', 'model_year', 'car_name'],
      dtype='object')
```

In [40]:
```
X = sm.add_constant(auto[[
#     'acceleration',
#     'cylinders',
    'weight',
#     'horse_power',
    'model_year',
#     'displacement'
                          ]].apply(lambda x: x.fillna(x.mean())))
Y = auto['mpg']
model = sm.OLS(Y, X).fit()
predictions = model.predict(X)
```

```
/home/david/miniconda3/envs/ironhack/lib/python3.7/site-packages/numpy/core/fromnumeric.py:2389: FutureWarning: Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

In [41]:
```
model.summary()
```

Out[41]:

OLS Regression Results

Dep. Variable:	mpg	**R-squared:**	0.808
Model:	OLS	**Adj. R-squared:**	0.807
Method:	Least Squares	**F-statistic:**	830.4
Date:	Tue, 16 Jul 2019	**Prob (F-statistic):**	3.26e-142
Time:	20:44:46	**Log-Likelihood:**	-1054.3
No. Observations:	398	**AIC:**	2115.
Df Residuals:	395	**BIC:**	2127.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-14.1980	3.968	-3.578	0.000	-21.998	-6.398
weight	-0.0067	0.000	-31.161	0.000	-0.007	-0.006
model_year	0.7566	0.049	15.447	0.000	0.660	0.853

Omnibus:	41.827	**Durbin-Watson:**	1.216
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	68.734
Skew:	0.665	**Prob(JB):**	1.19e-15
Kurtosis:	4.541	**Cond. No.**	7.12e+04

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.12e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

- check table for significant variables (model_year, weight)

¿car brand? not suitable for linear regression out of the box, one hot encoding needed!

In [147]:
```
auto['brand'] = auto['car_name'] \
                .str.lstrip('\t') \
                .str.strip('"') \
                .str.split(' ') \
                .str[0].astype('category')

pd.get_dummies(auto['brand'].head())
```

Out[147]:

	amc	audi	bmw	buick	cadillac	capri	chevroelt	chevrolet	chevy	chrysler	...	renault	saab	subar
0	0	0	0	0	0	0	0	1	0	0	...	0	0	
1	0	0	0	1	0	0	0	0	0	0	...	0	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	0	
3	1	0	0	0	0	0	0	0	0	0	...	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	

5 rows × 37 columns

Alternative using scikit-learn:

In [39]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

X = auto[[
#     'acceleration',
#     'cylinders',
    'weight',
#     'horse_power',
    'model_year',
#     'displacement'
                        ]]

y = auto['mpg']

model = LinearRegression(fit_intercept=True)
model.fit(X, y)
predictions = model.predict(X)

print(r2_score(y_true=y, y_pred=predictions))
print(model.coef_)
print(model.intercept_)
```

```
0.8078621345742751
[-0.00666388  0.75657249]
-14.197981575721002
```