Simple linear regression with Note

August 27, 2022

metadata https://rdrr.io/cran/AER/man/CASchools.html

[11]:
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

import statsmodels.formula.api as smf
import statsmodels.stats.api as sms
import statsmodels.graphics.api as smg

from scipy import stats
```

[17]:
```
from statsmodels.compat import lzip
```

[5]:
```
df = pd.read_csv("caschool.csv", engine='python')

print ("Row and columns = ", df.shape)
df.head()
```

```
Row and columns =  (420, 18)
```

[5]:
```
   Observation Number  dist_cod   county                        district  \
0                   1     75119  Alameda              Sunol Glen Unified
1                   2     61499    Butte            Manzanita Elementary
2                   3     61549    Butte     Thermalito Union Elementary
3                   4     61457    Butte  Golden Feather Union Elementary
4                   5     61523    Butte        Palermo Union Elementary

  gr_span  enrl_tot   teachers   calw_pct   meal_pct  computer     testscr  \
0   KK-08       195  10.900000   0.510200   2.040800        67  690.799988
1   KK-08       240  11.150000  15.416700  47.916698       101  661.200012
2   KK-08      1550  82.900002  55.032299  76.322601       169  643.599976
3   KK-08       243  14.000000  36.475399  77.049202        85  647.700012
4   KK-08      1335  71.500000  33.108601  78.427002       171  640.849976

   comp_stu     expn_stu        str     avginc     el_pct    read_scr  \
0  0.343590  6384.911133  17.889910  22.690001   0.000000  691.599976
```

```
1  0.420833  5099.380859  21.524664   9.824000   4.583333  660.500000
2  0.109032  5501.954590  18.697226   8.978000  30.000002  636.299988
3  0.349794  7101.831055  17.357143   8.978000   0.000000  651.900024
4  0.128090  5235.987793  18.671330   9.080333  13.857677  641.799988

     math_scr
0  690.000000
1  661.900024
2  650.900024
3  643.500000
4  639.900024
```

[6]:
```
df.columns
```

[6]:
```
Index(['Observation Number', 'dist_cod', 'county', 'district', 'gr_span',
       'enrl_tot', 'teachers', 'calw_pct', 'meal_pct', 'computer', 'testscr',
       'comp_stu', 'expn_stu', 'str', 'avginc', 'el_pct', 'read_scr',
       'math_scr'],
      dtype='object')
```

0.0.1 Pearson correlation

[18]:
```
pearsonc = stats.pearsonr(df['testscr'], df['avginc'])
name =  ["Coefficient", "p value"]
lzip(name, pearsonc)
```

[18]:
```
[('Coefficient', 0.7124308316760397), ('p value', 2.751833507384696e-66)]
```

[8]:
```
help(stats.pearsonr)
```

```
Help on function pearsonr in module scipy.stats.stats:

pearsonr(x, y)
    Pearson correlation coefficient and p-value for testing non-correlation.

    The Pearson correlation coefficient [1]_ measures the linear relationship
    between two datasets.  The calculation of the p-value relies on the
    assumption that each dataset is normally distributed.  (See Kowalski [3]_
    for a discussion of the effects of non-normality of the input on the
    distribution of the correlation coefficient.)  Like other correlation
    coefficients, this one varies between -1 and +1 with 0 implying no
    correlation. Correlations of -1 or +1 imply an exact linear relationship.
    Positive correlations imply that as x increases, so does y. Negative
    correlations imply that as x increases, y decreases.

    The p-value roughly indicates the probability of an uncorrelated system
    producing datasets that have a Pearson correlation at least as extreme
    as the one computed from these datasets.
```

```
Parameters
----------
x : (N,) array_like
    Input array.
y : (N,) array_like
    Input array.

Returns
-------
r : float
    Pearson's correlation coefficient.
p-value : float
    Two-tailed p-value.

Warns
-----
PearsonRConstantInputWarning
    Raised if an input is a constant array.  The correlation coefficient
    is not defined in this case, so ``np.nan`` is returned.

PearsonRNearConstantInputWarning
    Raised if an input is "nearly" constant.  The array ``x`` is considered
    nearly constant if ``norm(x - mean(x)) < 1e-13 * abs(mean(x))``.
    Numerical errors in the calculation ``x - mean(x)`` in this case might
    result in an inaccurate calculation of r.

See Also
--------
spearmanr : Spearman rank-order correlation coefficient.
kendalltau : Kendall's tau, a correlation measure for ordinal data.

Notes
-----
The correlation coefficient is calculated as follows:

.. math::

    r = \frac{\sum (x - m_x) (y - m_y)}
             {\sqrt{\sum (x - m_x)^2 \sum (y - m_y)^2}}

where :math:`m_x` is the mean of the vector :math:`x` and :math:`m_y` is
the mean of the vector :math:`y`.

Under the assumption that x and y are drawn from independent normal
distributions (so the population correlation coefficient is 0), the
probability density function of the sample correlation coefficient r
is ([1]_, [2]_)::
```

```
        (1 - r**2)**(n/2 - 2)
f(r) = ---------------------
           B(1/2, n/2 - 1)
```

where n is the number of samples, and B is the beta function. This is sometimes referred to as the exact distribution of r. This is the distribution that is used in `pearsonr` to compute the p-value. The distribution is a beta distribution on the interval [-1, 1], with equal shape parameters a = b = n/2 - 1. In terms of SciPy's implementation of the beta distribution, the distribution of r is::

```
dist = scipy.stats.beta(n/2 - 1, n/2 - 1, loc=-1, scale=2)
```

The p-value returned by `pearsonr` is a two-sided p-value. For a given sample with correlation coefficient r, the p-value is the probability that abs(r') of a random sample x' and y' drawn from the population with zero correlation would be greater than or equal to abs(r). In terms of the object ``dist`` shown above, the p-value for a given r and length n can be computed as::

```
p = 2*dist.cdf(-abs(r))
```

When n is 2, the above continuous distribution is not well-defined. One can interpret the limit of the beta distribution as the shape parameters a and b approach a = b = 0 as a discrete distribution with equal probability masses at r = 1 and r = -1. More directly, one can observe that, given the data x = [x1, x2] and y = [y1, y2], and assuming x1 != x2 and y1 != y2, the only possible values for r are 1 and -1. Because abs(r') for any sample x' and y' with length 2 will be 1, the two-sided p-value for a sample of length 2 is always 1.

References

.. [1] "Pearson correlation coefficient", Wikipedia,
       https://en.wikipedia.org/wiki/Pearson_correlation_coefficient
.. [2] Student, "Probable error of a correlation coefficient",
       Biometrika, Volume 6, Issue 2-3, 1 September 1908, pp. 302-310.
.. [3] C. J. Kowalski, "On the Effects of Non-Normality on the Distribution
       of the Sample Product-Moment Correlation Coefficient"
       Journal of the Royal Statistical Society. Series C (Applied
       Statistics), Vol. 21, No. 1 (1972), pp. 1-12.

Examples

```
>>> from scipy import stats
>>> a = np.array([0, 0, 0, 1, 1, 1, 1])
>>> b = np.arange(7)
```

```
>>> stats.pearsonr(a, b)
(0.8660254037844386, 0.011724811003954649)

>>> stats.pearsonr([1, 2, 3, 4, 5], [10, 9, 2.5, 6, 4])
(-0.7426106572325057, 0.1505558088534455)
```

0.0.2 Spearman correlation

[19]:
```
spearmanc = stats.spearmanr(df['testscr'], df['avginc'])
name =  ["Coefficient", "p value"]
lzip(name, spearmanc)
```

[19]:
```
[('Coefficient', 0.6897405358562295), ('p value', 1.3683335426820145e-60)]
```

0.0.3 Creating first model

[25]:
```
model = smf.ols('testscr ~avginc', data=df)
model_fit = model.fit()

#model_fit = smf.ols('testscr ~avginc', data=df).fit()
print(model_fit.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                testscr   R-squared:                       0.508
Model:                            OLS   Adj. R-squared:                  0.506
Method:                 Least Squares   F-statistic:                     430.8
Date:                Sun, 12 Sep 2021   Prob (F-statistic):           2.75e-66
Time:                        15:36:03   Log-Likelihood:                -1684.5
No. Observations:                 420   AIC:                             3373.
Df Residuals:                     418   BIC:                             3381.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    625.3836      1.532    408.106      0.000     622.371     628.396
avginc         1.8785      0.091     20.756      0.000       1.701       2.056
==============================================================================
Omnibus:                        2.473   Durbin-Watson:                   0.650
Prob(Omnibus):                  0.290   Jarque-Bera (JB):                2.480
Skew:                          -0.148   Prob(JB):                        0.289
Kurtosis:                       2.766   Cond. No.                         39.8
==============================================================================

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

specified.

[26]:
```
dir(model_fit)
```

[26]:
```
['HC0_se',
 'HC1_se',
 'HC2_se',
 'HC3_se',
 '_HCCM',
 '__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__le__',
 '__lt__',
 '__module__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 '__weakref__',
 '_cache',
 '_data_attr',
 '_get_robustcov_results',
 '_is_nested',
 '_use_t',
 '_wexog_singular_values',
 'aic',
 'bic',
 'bse',
 'centered_tss',
 'compare_f_test',
 'compare_lm_test',
```

```
'compare_lr_test',
'condition_number',
'conf_int',
'conf_int_el',
'cov_HC0',
'cov_HC1',
'cov_HC2',
'cov_HC3',
'cov_kwds',
'cov_params',
'cov_type',
'df_model',
'df_resid',
'diagn',
'eigenvals',
'el_test',
'ess',
'f_pvalue',
'f_test',
'fittedvalues',
'fvalue',
'get_influence',
'get_prediction',
'get_robustcov_results',
'initialize',
'k_constant',
'llf',
'load',
'model',
'mse_model',
'mse_resid',
'mse_total',
'nobs',
'normalized_cov_params',
'outlier_test',
'params',
'predict',
'pvalues',
'remove_data',
'resid',
'resid_pearson',
'rsquared',
'rsquared_adj',
'save',
'scale',
'ssr',
'summary',
```

```
 'summary2',
 't_test',
 't_test_pairwise',
 'tvalues',
 'uncentered_tss',
 'use_t',
 'wald_test',
 'wald_test_terms',
 'wresid']
```

[31]:
```python
model_fit.resid
```

[31]:
```
0       22.792119
1       17.361563
2        1.350780
3        5.450817
4       -1.591458
          …
415     24.970164
416      2.966831
417    -24.967195
418     28.121110
419      6.880795
Length: 420, dtype: float64
```

0.1 Diagnostics for the first model

https://www.statsmodels.org/dev/examples/notebooks/generated/regression_diagnostics.html

0.1.1 Linearlity

[38]:
```python
help(sms.linear_harvey_collier)
```

```
Help on function linear_harvey_collier in module statsmodels.stats.diagnostic:

linear_harvey_collier(res, order_by=None)
    Harvey Collier test for linearity

    The Null hypothesis is that the regression is correctly modeled as linear.

    Parameters
    ----------
    res : RegressionResults
        A results instance from a linear regression.
    order_by : array_like, default None
        Integer array specifying the order of the residuals. If not provided,
        the order of the residuals is not changed. If provided, must have
        the same number of observations as the endogenous variable.
```

```
    Returns
    -------
    tvalue : float
        The test statistic, based on ttest_1sample.
    pvalue : float
        The pvalue of the test.

    Notes
    -----
    This test is a t-test that the mean of the recursive ols residuals is zero.
    Calculating the recursive residuals might take some time for large samples.
```

[72]:
```python
name = ["t value", "p value"]
test = sms.linear_harvey_collier(model_fit)
lzip(name, test)
```

[72]:
```
[('t value', 15.234032545944318), ('p value', 5.841712400696894e-42)]
```

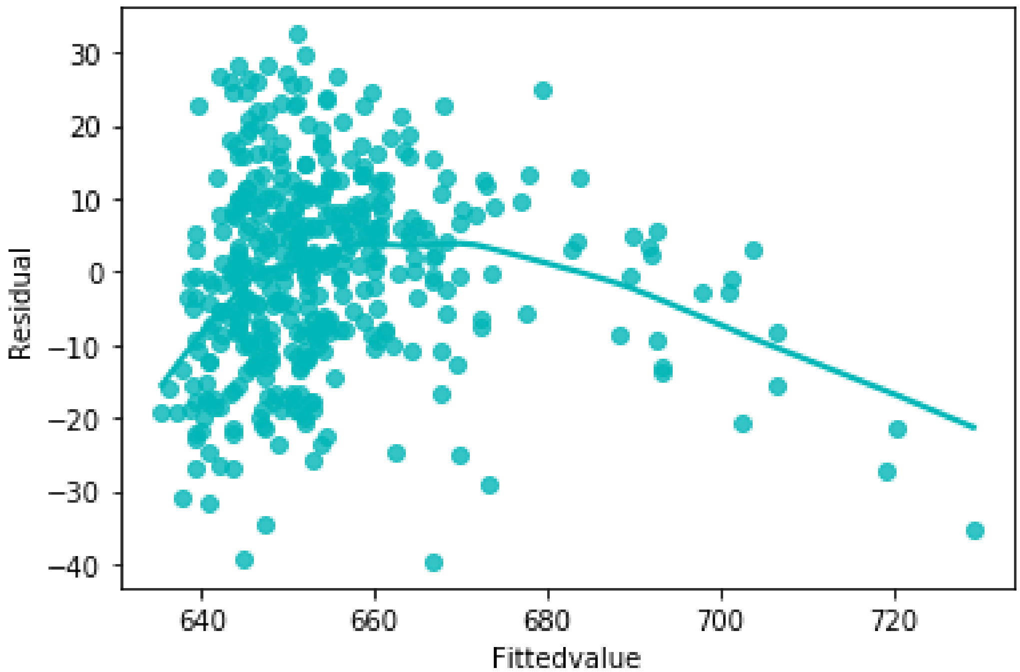
Let significance level = 0.05

p-value = 0.0000

As p-value <=0.05 , we can conclude that linearity assumption is violated.

[37]:
```python
sns.regplot(model_fit.fittedvalues,
                model_fit.resid,
                data=df,
                lowess=True,
                color='#01B6B7')

plt.xlabel('Fittedvalue')
plt.ylabel('Residual')
```

[37]:
```
Text(0, 0.5, 'Residual')
```



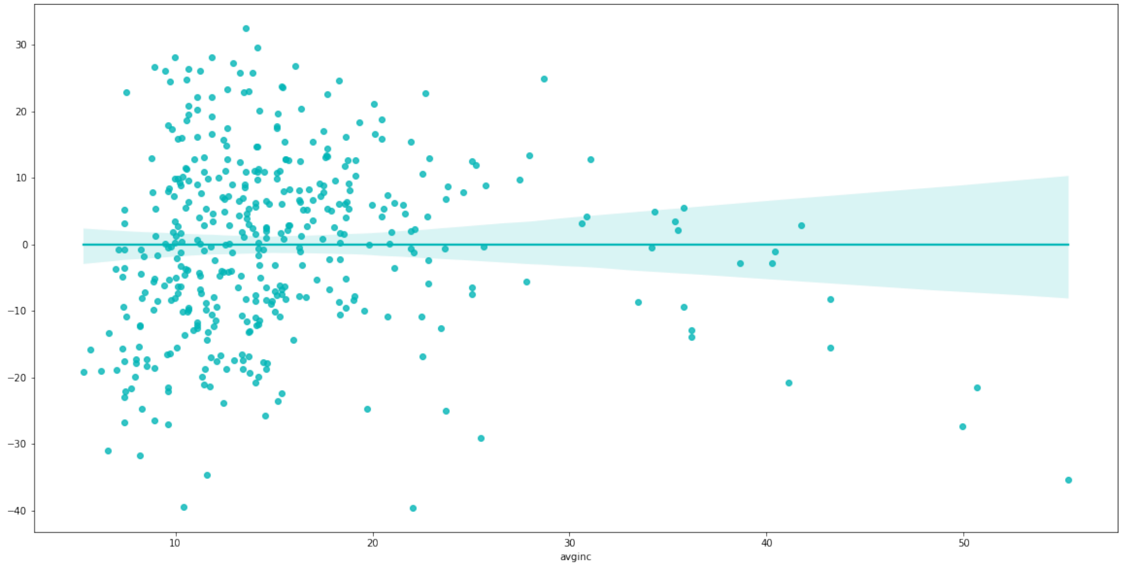
0.1.2 Independence

https://www.statsmodels.org/stable/generated/statsmodels.stats.stattools.durbin_watson.html

Durbin-Watson: 0.650

[40]:
```
plt.rc('figure', figsize=(20,10))

sns.regplot(df['avginc'], model_fit.resid ,
                    data=df,  color='#01B6B7')
```

[40]: <matplotlib.axes._subplots.AxesSubplot at 0x1fc4ac82708>



0.1.3 Normality

[45]:
```
help(sms.jarque_bera)
```

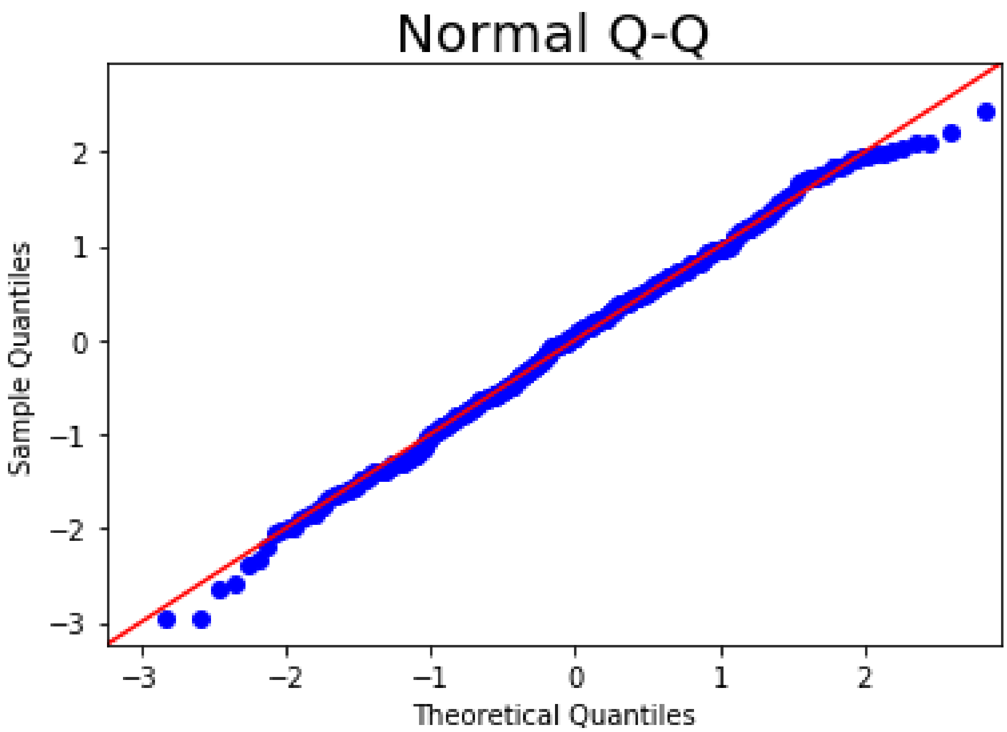
```
Help on function jarque_bera in module statsmodels.stats.stattools:

jarque_bera(resids, axis=0)
    The Jarque-Bera test of normality.

    Parameters
    ----------
    resids : array_like
        Data to test for normality. Usually regression model residuals that
        are mean 0.
    axis : int, optional
        Axis to use if data has more than 1 dimension. Default is 0.

    Returns
    -------
    JB : {float, ndarray}
        The Jarque-Bera test statistic.
    JBpv : {float, ndarray}
        The pvalue of the test statistic.
    skew : {float, ndarray}
        Estimated skewness of the data.
    kurtosis : {float, ndarray}
        Estimated kurtosis of the data.
```

```
Notes
-----
Each output returned has 1 dimension fewer than data

The Jarque-Bera test statistic tests the null that the data is normally
distributed against an alternative that the data follow some other
distribution. The test statistic is based on two moments of the data,
the skewness, and the kurtosis, and has an asymptotic :math:`\chi^2_2`
distribution.

The test statistic is defined

.. math:: JB = n(S^2/6+(K-3)^2/24)

where n is the number of data points, S is the sample skewness, and K is
the sample kurtosis of the data.
```

[46]:
```
jarque = sms.jarque_bera(model_fit.resid)

names = ['Test stattics', 'p-value' , 'skew', 'kurtosis']
lzip(names, jarque)
```

[46]:
```
[('Test stattics', 2.480011745487751),
 ('p-value', 0.28938251846464735),
 ('skew', -0.147592895772964 86),
 ('kurtosis', 2.766375717541 14)]
```

[43]:
```
plt.rc('figure', figsize=(6,4))
probplot  = smg.qqplot(model_fit.resid, fit=True, line="45")

plt.title('Normal Q-Q', fontsize=20)
plt.show()
```



0.1.4 Heteroskedascity

[49]:
```
help(het_white)
```

```
Help on function het_white in module statsmodels.stats.diagnostic:

het_white(resid, exog)
    White's Lagrange Multiplier Test for Heteroscedasticity.

    Parameters
    ----------
    resid : array_like
        The residuals. The squared residuals are used as the endogenous
        variable.
    exog : array_like
        The explanatory variables for the variance. Squares and interaction
        terms are automatically included in the auxiliary regression.

    Returns
    -------
    lm : float
        The lagrange multiplier statistic.
    lm_pvalue :float
```

```
        The p-value of lagrange multiplier test.
    fvalue : float
        The f-statistic of the hypothesis that the error variance does not
        depend on x. This is an alternative test variant not the original
        LM test.
    f_pvalue : float
        The p-value for the f-statistic.

    Notes
    -----
    Assumes x contains constant (for counting dof).

    question: does f-statistic make sense? constant ?

    References
    ----------
    Greene section 11.4.1 5th edition p. 222. Test statistic reproduces
    Greene 5th, example 11.3.
```

[50]:

```python
from statsmodels.stats.api import het_white
name = ["Lagrange multiplier statistic", "p-value", "f-value", "f p-value"]

test = het_white(model_fit.resid, model_fit.model.exog)

lzip(name, test)
```

[50]:

```
[('Lagrange multiplier statistic', 25.02860405604711),
 ('p-value', 3.6737338031689657e-06),
 ('f-value', 13.212257898357608),
 ('f p-value', 2.731869286255225e-06)]
```

[75]:

```python
print("Breush-Pagan test:")
name = ["Lagrange multiplier statistic", "p-value", "f-value", "f p-value"]
test = sms.het_breuschpagan(model_fit.resid, model_fit.model.exog)
lzip(name, test)
```

```
Breush-Pagan test:
```

[75]:

```
[('Lagrange multiplier statistic', 0.07868804387878292),
 ('p-value', 0.7790832684859628),
 ('f-value', 0.07832801385599489),
 ('f p-value', 0.7797145934511552)]
```

[76]:

```python
print("Goldfeld-Quandt test:")
name = ["F statistic", "p-value"]
test = sms.het_goldfeldquandt(model_fit.resid, model_fit.model.exog)
lzip(name, test)
```

```
Goldfeld-Quandt test:
```

[76]:
```
[('F statistic', 0.6354101327810716), ('p-value', 0.9994277828994063)]
```

0.1.5 Creating alternative model

0.1.6 1. log regression model

[51]:
```python
df['log_testscr'] = np.log(df['testscr'])
```

[53]:
```python
df.head()
```

[53]:
```
   Observation Number  dist_cod    county                        district  \
0                   1     75119   Alameda              Sunol Glen Unified
1                   2     61499     Butte            Manzanita Elementary
2                   3     61549     Butte     Thermalito Union Elementary
3                   4     61457     Butte  Golden Feather Union Elementary
4                   5     61523     Butte        Palermo Union Elementary

  gr_span  enrl_tot   teachers   calw_pct   meal_pct  computer     testscr  \
0   KK-08       195  10.900000   0.510200   2.040800        67  690.799988
1   KK-08       240  11.150000  15.416700  47.916698       101  661.200012
2   KK-08      1550  82.900002  55.032299  76.322601       169  643.599976
3   KK-08       243  14.000000  36.475399  77.049202        85  647.700012
4   KK-08      1335  71.500000  33.108601  78.427002       171  640.849976

   comp_stu     expn_stu        str     avginc     el_pct    read_scr  \
0  0.343590  6384.911133  17.889910  22.690001   0.000000  691.599976
1  0.420833  5099.380859  21.524664   9.824000   4.583333  660.500000
2  0.109032  5501.954590  18.697226   8.978000  30.000002  636.299988
3  0.349794  7101.831055  17.357143   8.978000   0.000000  651.900024
4  0.128090  5235.987793  18.671330   9.080333  13.857677  641.799988

     math_scr  log_testscr
0  690.000000     6.537850
1  661.900024     6.494056
2  650.900024     6.467077
3  643.500000     6.473428
4  639.900024     6.462795
```

[54]:
```python
model2 = smf.ols('log_testscr ~avginc ', data=df)

model2_fit = model2.fit()
print(model2_fit.summary())
```

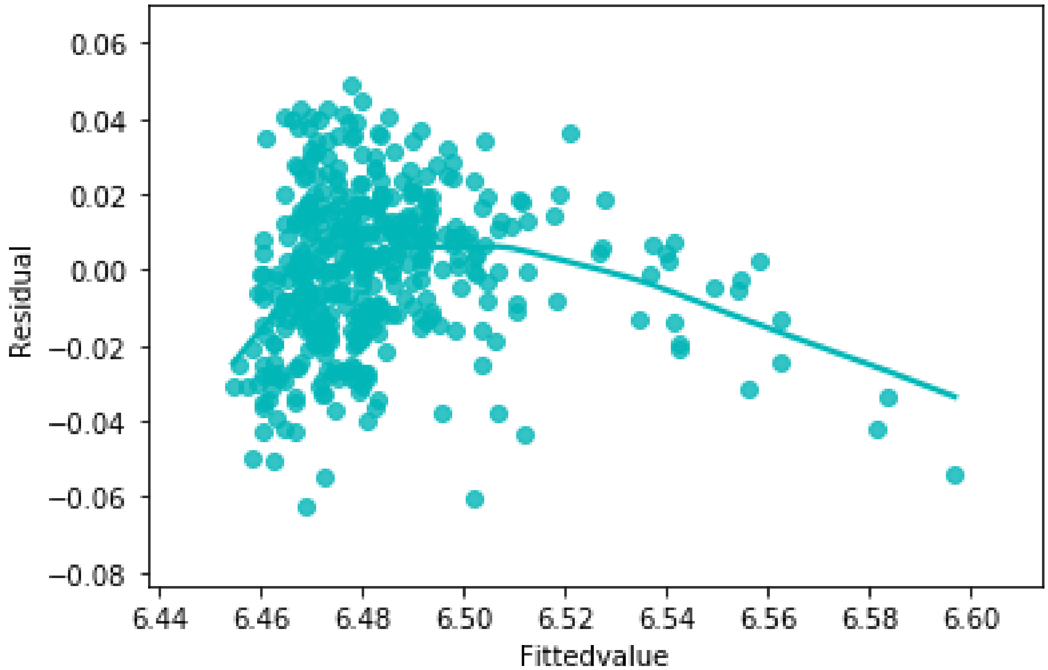
```
                            OLS Regression Results
==============================================================================
Dep. Variable:            log_testscr   R-squared:                       0.498
Model:                            OLS   Adj. R-squared:                  0.497
```

```
Method:                 Least Squares   F-statistic:                     415.0
Date:                Sun, 12 Sep 2021   Prob (F-statistic):           1.41e-64
Time:                        16:16:52   Log-Likelihood:                 1034.7
No. Observations:                 420   AIC:                            -2065.
Df Residuals:                     418   BIC:                            -2057.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      6.4394      0.002   2724.161      0.000       6.435       6.444
avginc         0.0028      0.000     20.372      0.000       0.003       0.003
==============================================================================
Omnibus:                        3.487   Durbin-Watson:                   0.627
Prob(Omnibus):                  0.175   Jarque-Bera (JB):                3.538
Skew:                          -0.200   Prob(JB):                        0.171
Kurtosis:                       2.795   Cond. No.                         39.8
==============================================================================

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

0.1.7 Linearity

[56]:
```
sns.regplot(model2_fit.fittedvalues,
                model2_fit.resid,
                data=df,
                lowess=True,
                color='#01B6B7')

plt.xlabel('Fittedvalue')
plt.ylabel('Residual')
```

[56]: Text(0, 0.5, 'Residual')



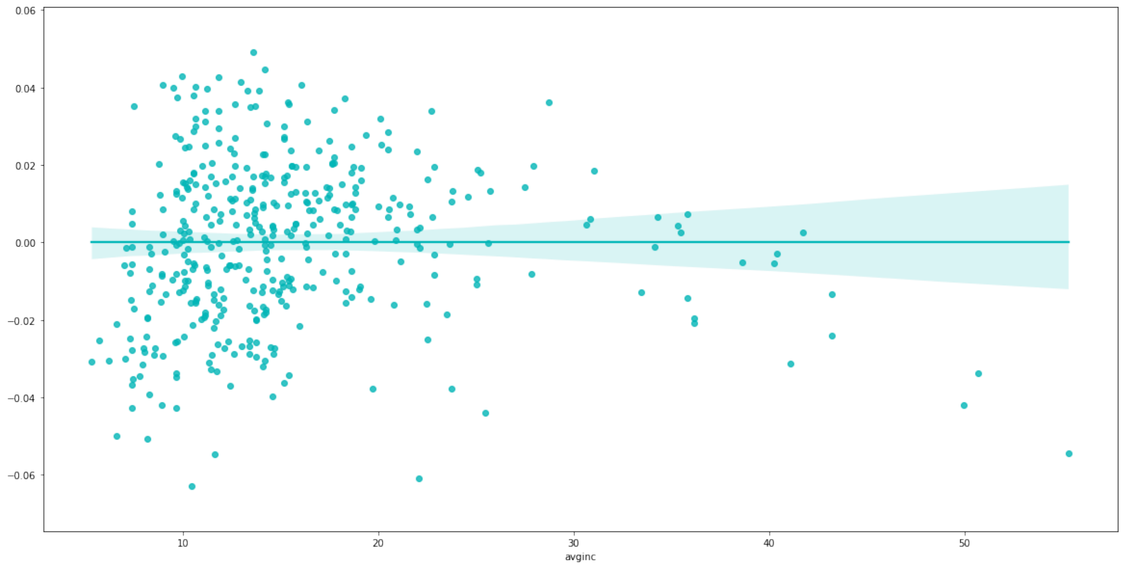
[58]:
```
name = ["t value", "p value"]
test = sms.linear_harvey_collier(model2_fit)
lzip(name, test)
```

[58]: [('t value', 15.150618564756465), ('p value', 1.3244115424471496e-41)]

0.1.8 Independence

[59]:
```
plt.rc('figure', figsize=(20,10))

sns.regplot(df['avginc'], model2_fit.resid ,
                    data=df,  color='#01B6B7')
```

[59]: <matplotlib.axes._subplots.AxesSubplot at 0x1fc4b209748>

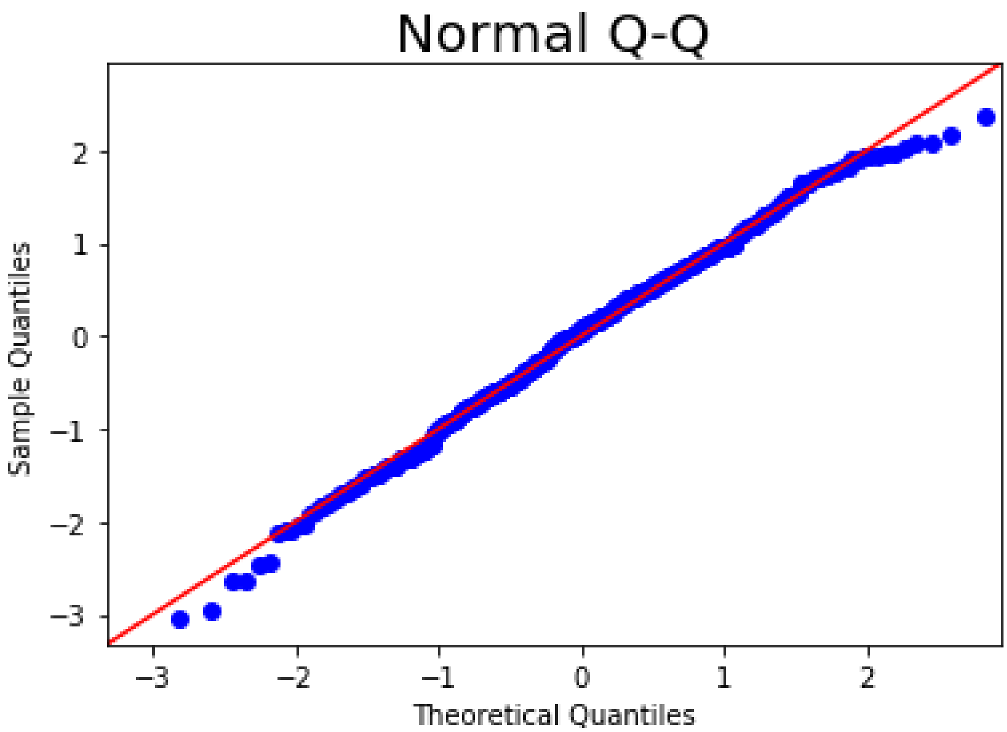


0.1.9 Normality

[60]:
```
jarque = sms.jarque_bera(model2_fit.resid)

names = ['Test stattics', 'p-value' , 'skew', 'kurtosis']
lzip(names, jarque)
```

[60]:
```
[('Test stattics', 3.53750575606874),
 ('p-value', 0.1705455473528997),
 ('skew', -0.20016309552074732),
 ('kurtosis', 2.7953487678334685)]
```

[61]:
```
plt.rc('figure', figsize=(6,4))
probplot  = smg.qqplot(model2_fit.resid, fit=True, line="45")

plt.title('Normal Q-Q', fontsize=20)
plt.show()
```



0.1.10 Heteroskedascity

[62]:
```
name = ["Lagrange multiplier statistic", "p-value", "f-value", "f p-value"]

test = het_white(model2_fit.resid, model2_fit.model.exog)

lzip(name, test)
```

[62]:
```
[('Lagrange multiplier statistic', 28.363263031205392),
 ('p-value', 6.934189218217106e-07),
 ('f-value', 15.100065401876554),
 ('f p-value', 4.663551148637077e-07)]
```

0.1.11 2. log-log regression model

[65]:
```
df.head()
```

[65]:
```
   Observation Number  dist_cod   county                      district  \
0                   1     75119  Alameda            Sunol Glen Unified
1                   2     61499    Butte          Manzanita Elementary
2                   3     61549    Butte      Thermalito Union Elementary
3                   4     61457    Butte  Golden Feather Union Elementary
4                   5     61523    Butte         Palermo Union Elementary
```

```
  gr_span  enrl_tot   teachers   calw_pct   meal_pct  computer     testscr  \
0   KK-08       195  10.900000   0.510200   2.040800        67  690.799988
1   KK-08       240  11.150000  15.416700  47.916698       101  661.200012
2   KK-08      1550  82.900002  55.032299  76.322601       169  643.599976
3   KK-08       243  14.000000  36.475399  77.049202        85  647.700012
4   KK-08      1335  71.500000  33.108601  78.427002       171  640.849976

   comp_stu     expn_stu        str     avginc     el_pct    read_scr  \
0  0.343590  6384.911133  17.889910  22.690001   0.000000  691.599976
1  0.420833  5099.380859  21.524664   9.824000   4.583333  660.500000
2  0.109032  5501.954590  18.697226   8.978000  30.000002  636.299988
3  0.349794  7101.831055  17.357143   8.978000   0.000000  651.900024
4  0.128090  5235.987793  18.671330   9.080333  13.857677  641.799988

     math_scr  log_testscr  log_avginc
0  690.000000     6.537850    3.121924
1  661.900024     6.494056    2.284828
2  650.900024     6.467077    2.194777
3  643.500000     6.473428    2.194777
4  639.900024     6.462795    2.206111
```

[64]:
```python
df['log_avginc'] = np.log(df['avginc'])
```

[66]:
```python
model3 = smf.ols('log_testscr ~log_avginc ', data=df)
model3_fit = model3.fit()
print(model3_fit.summary())
```

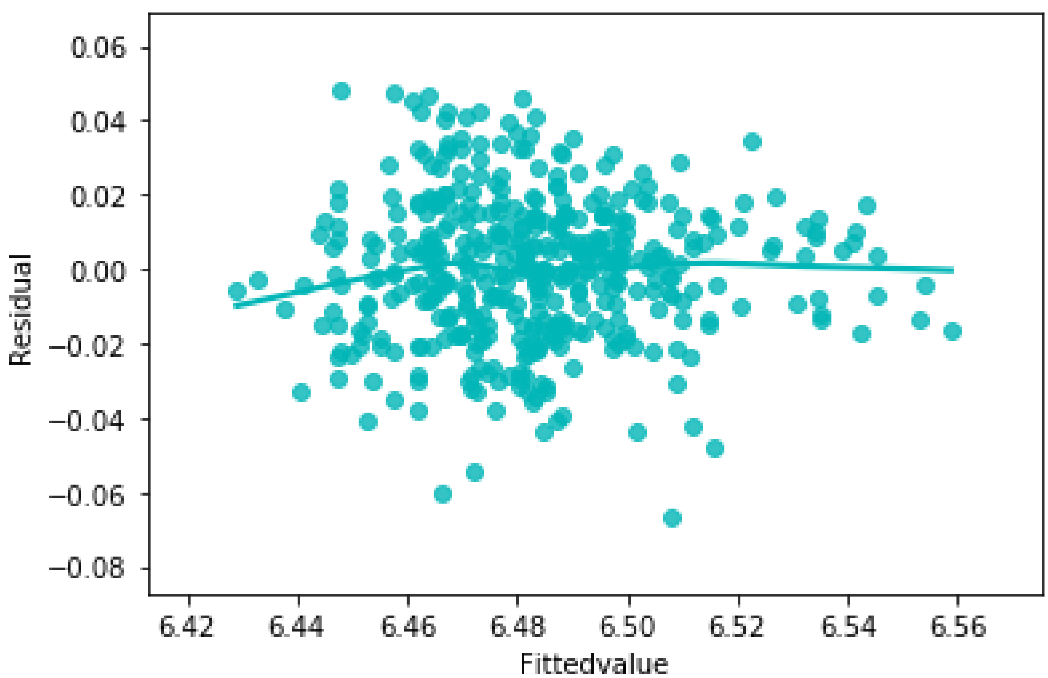
```
                            OLS Regression Results
==============================================================================
Dep. Variable:            log_testscr   R-squared:                       0.558
Model:                            OLS   Adj. R-squared:                  0.557
Method:                 Least Squares   F-statistic:                     527.2
Date:                Sun, 12 Sep 2021   Prob (F-statistic):           4.52e-76
Time:                        16:36:00   Log-Likelihood:                 1061.2
No. Observations:                 420   AIC:                            -2118.
Df Residuals:                     418   BIC:                            -2110.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      6.3363      0.006    981.902      0.000       6.324       6.349
log_avginc     0.0554      0.002     22.962      0.000       0.051       0.060
==============================================================================
Omnibus:                        1.085   Durbin-Watson:                   0.972
Prob(Omnibus):                  0.581   Jarque-Bera (JB):                0.886
Skew:                          -0.096   Prob(JB):                        0.642
```

```
Kurtosis:                       3.118   Cond. No.                       20.7
==============================================================================

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

0.1.12 Linearity

[67]:
```python
sns.regplot(model3_fit.fittedvalues,
                model3_fit.resid,
                data=df,
                lowess=True,
                color='#01B6B7')

plt.xlabel('Fittedvalue')
plt.ylabel('Residual')
```

[67]: Text(0, 0.5, 'Residual')



[68]:
```python
name = ["t value", "p value"]
test = sms.linear_harvey_collier(model3_fit)
lzip(name, test)
```

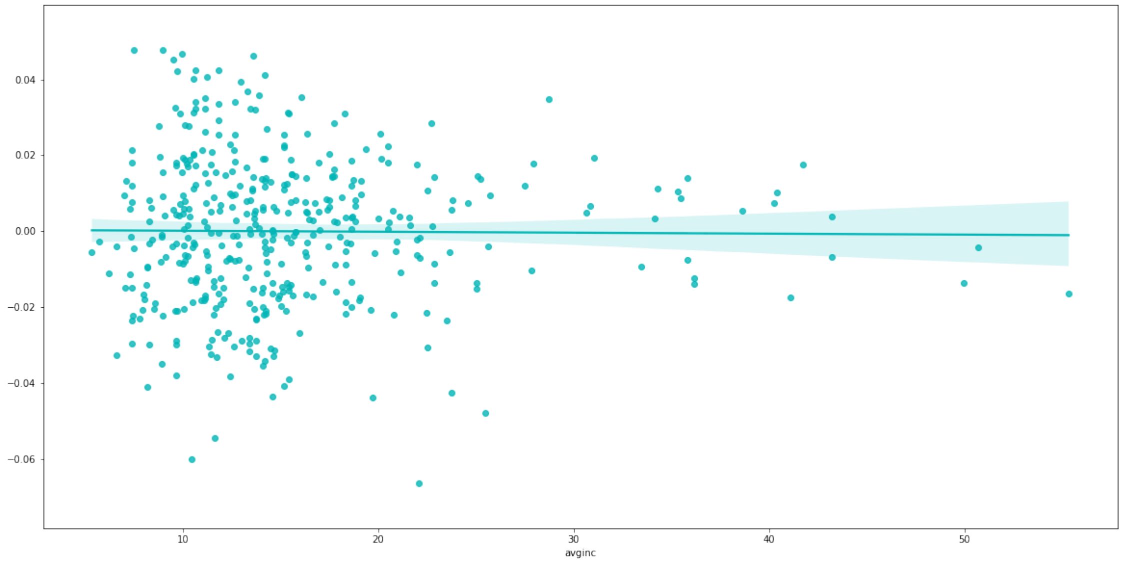
[68]: [('t value', 19.104283757713628), ('p value', 7.230829376569102e-59)]

0.1.13 Independence

Durbin-Watson: 0.972

[69]:
```
plt.rc('figure', figsize=(20,10))

sns.regplot(df['avginc'], model3_fit.resid ,
                  data=df,  color='#01B6B7')
```

[69]: <matplotlib.axes._subplots.AxesSubplot at 0x1fc4bad3808>

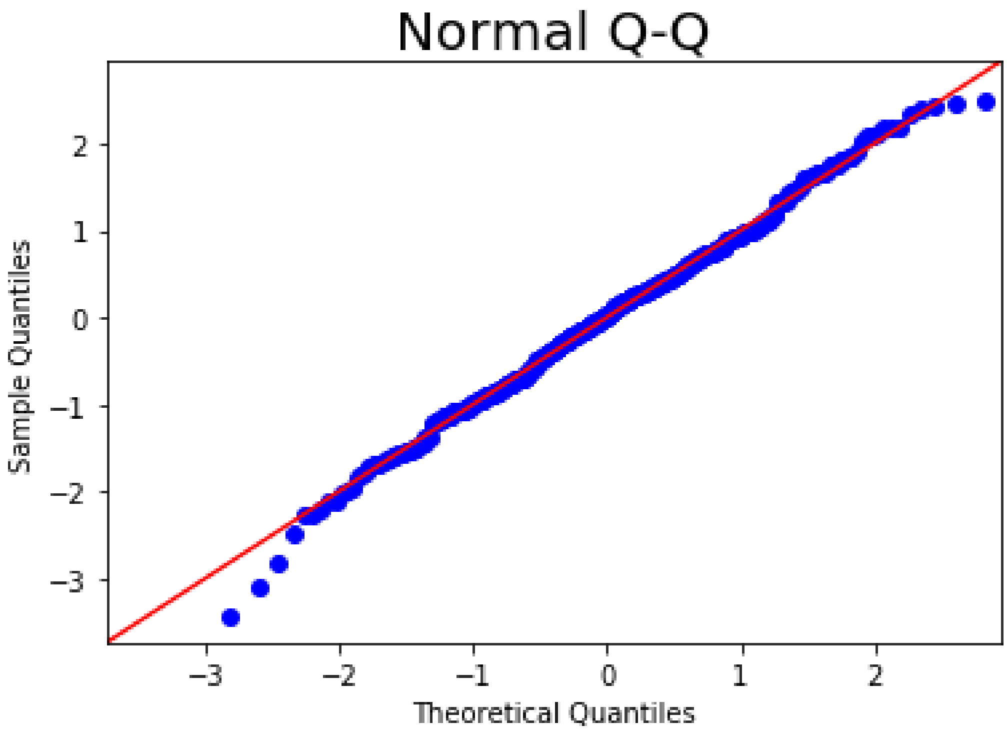


0.1.14 Normality

[71]:
```
jarque = sms.jarque_bera(model3_fit.resid)

names = ['Test stattics', 'p-value' , 'skew', 'kurtosis']
lzip(names, jarque)
```

[71]:
```
[('Test stattics', 0.8856399073699899),
 ('p-value', 0.6422228265323846),
 ('skew', -0.09571830973640676),
 ('kurtosis', 3.1181525092060163)]
```

[73]:
```
plt.rc('figure', figsize=(6,4))
probplot  = smg.qqplot(model3_fit.resid, fit=True, line="45")

plt.title('Normal Q-Q', fontsize=20)
plt.show()
```



0.1.15 Heteroskedascity

[76]:
```
name = ["Lagrange multiplier statistic", "p-value", "f-value", "f p-value"]

test = het_white(model3_fit.resid, model3_fit.model.exog)

lzip(name, test)
```

[76]:
```
[('Lagrange multiplier statistic', 6.465177397798771),
 ('p-value', 0.039455228765083505),
 ('f-value', 3.2596758816070572),
 ('f p-value', 0.03938155975048735)]
```

[ ]:
```

```