test-assumptions-2022

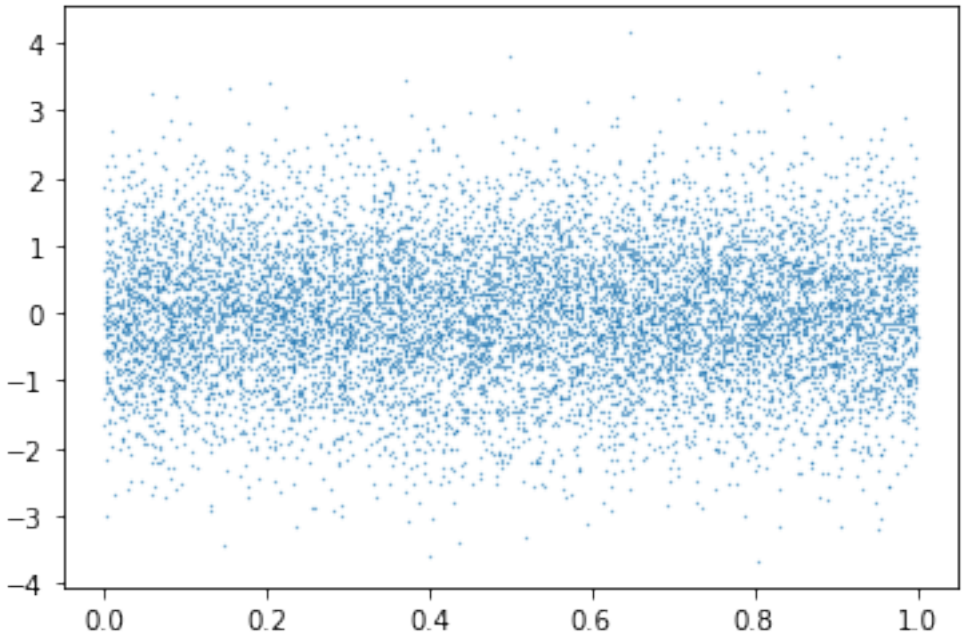
March 23, 2022

[4]:
```
# dataframes
import pandas as pd
from numpy.random import rand, randn, beta, poisson
from numpy import sin
import numpy as np
import scipy
import math

# plotting
import matplotlib.pyplot as plt
import seaborn as sns

# linear regression two ways
import statsmodels.api as sm
from sklearn.linear_model import LinearRegression

# for choosing covariates to include in model
from patsy import dmatrices
```

1 Method 1

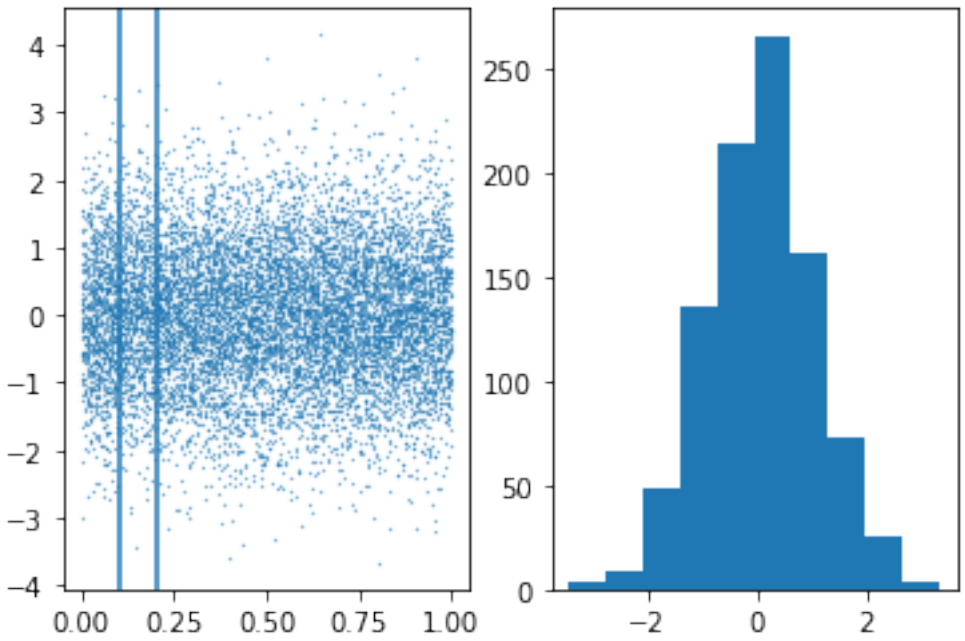
[3]:
```
n=500
eps = exp(randn(n)) # not normal x = 10*rand(n)
y = x + eps
model = sm.OLS(y,x).fit()
sm.qqplot(model.resid , line='r')# Basic verison of method 1
n = 10000
x = rand(n) # uniform
eps = randn(n) # normal, mean 0 variance 1, independent
y = 1 + 2*x + eps
model = sm.OLS(y,sm.add_constant(x)).fit()
plt.plot(x,model.resid,'.',markersize=0.5)
```

[3]:
```
[<matplotlib.lines.Line2D at 0x7f9bd12fc7d0>]
```



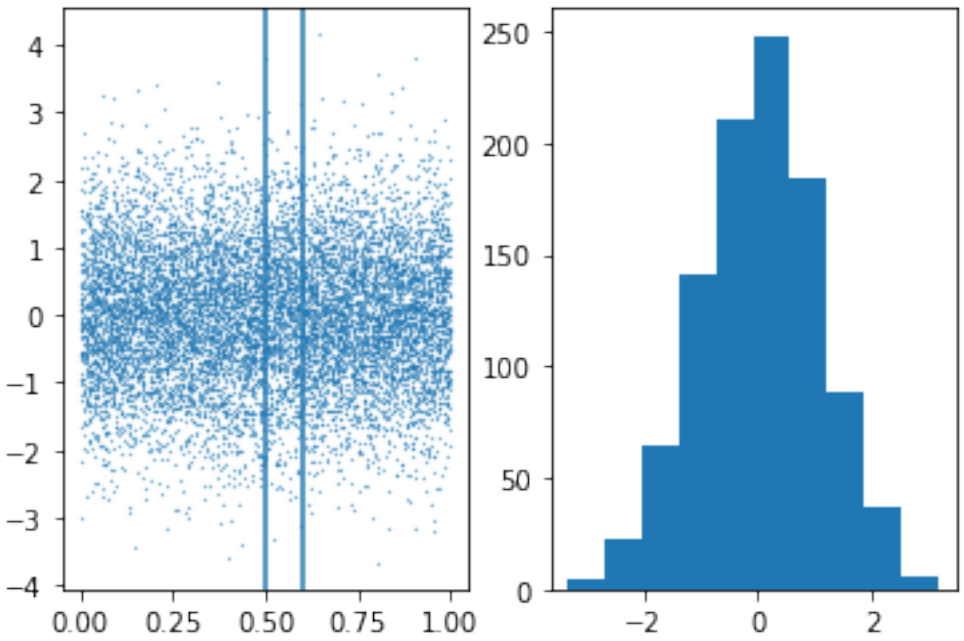
[4]:
```
# Here's what you're looking for with method 1
lb = 0.1 # lower bound
ub = 0.2 # upper bound
plt.subplot(1,2,1)
plt.plot(x,model.resid,'.',markersize=0.5)
plt.axvline(lb)
plt.axvline(ub)
plt.subplot(1,2,2)
plt.hist(eps[(x >= lb) & (x <= ub)])
```

[4]:
```
(array([  3.,   9.,  49., 136., 214., 266., 161.,  73.,  26.,   3.]),
 array([-3.43145709, -2.75509365, -2.07873021, -1.40236677, -0.72600333,
        -0.04963989,  0.62672354,  1.30308698,  1.97945042,  2.65581386,
         3.3321773 ]),
 <a list of 10 Patch objects>)
```



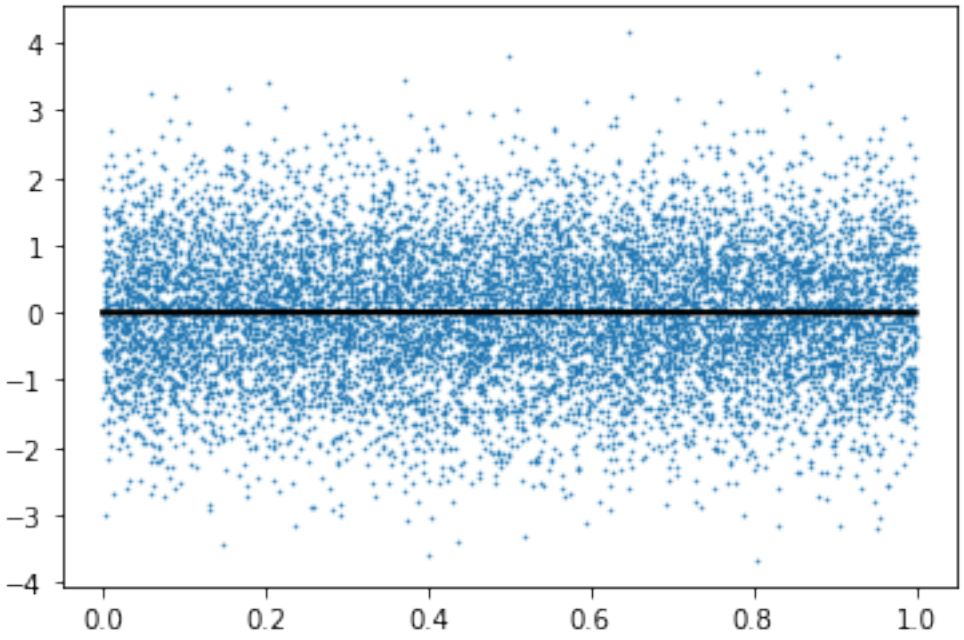
[5]:
```
lb = 0.5 # lower bound
ub = 0.6 # upper bound
plt.subplot(1,2,1)
plt.plot(x,model.resid,'.',markersize=0.5)
plt.axvline(lb)
plt.axvline(ub)
plt.subplot(1,2,2)
plt.hist(eps[(x >= lb) & (x <= ub)])
```

[5]:
```
(array([  4.,  22.,  64., 141., 211., 248., 184.,  88.,  37.,   6.]),
 array([-3.31575219, -2.66948245, -2.02321271, -1.37694297, -0.73067323,
        -0.08440348,  0.56186626,  1.208136  ,  1.85440574,  2.50067549,
         3.14694523]),
 <a list of 10 Patch objects>)
```



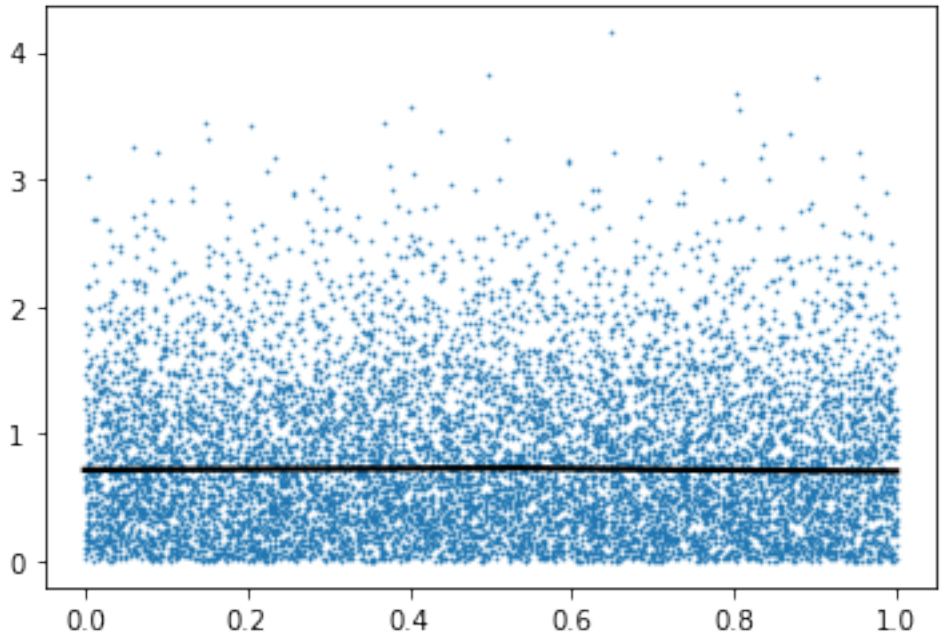
[6]:
```
# Look for the black line to stay constant
sns.regplot(x, model.resid, lowess=True, line_kws = {'color': 'black'},␣
↪scatter_kws = {'s' : 0.5})
```

[6]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9c409d3d50>



[7]:
```
# This is a nice way to check for non-constant variance
sns.regplot(x, abs(model.resid), lowess=True, line_kws = {'color': 'black'},␣
↪scatter_kws = {'s' : 0.5})
```

[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9be8904f90>
```

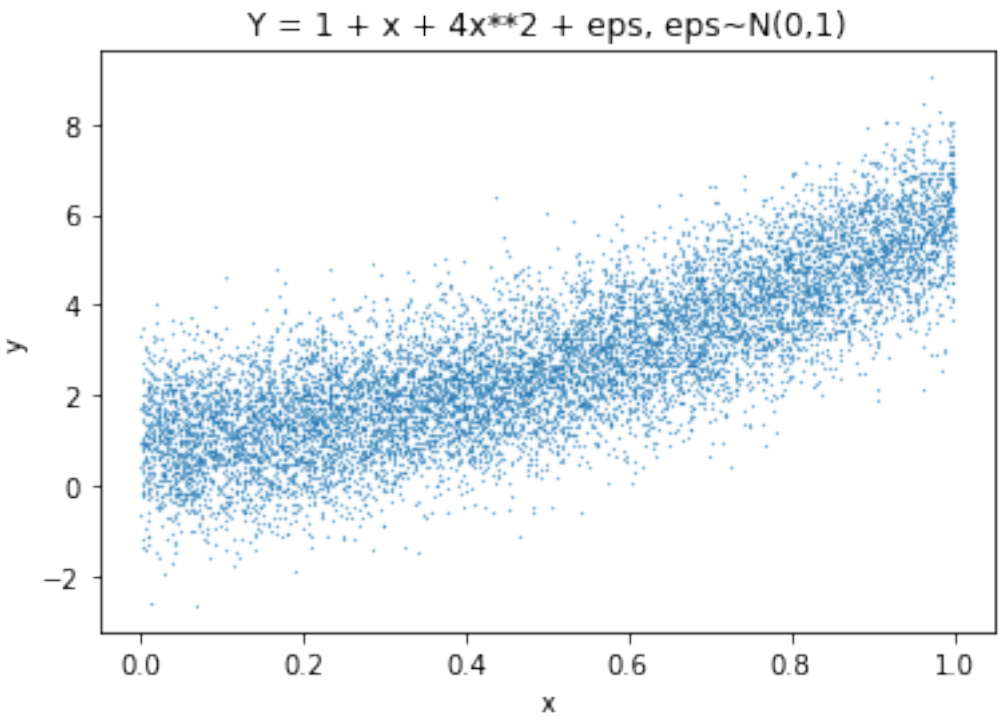


1.0.1 Example where data is quadratic in x

[8]:
```
# What happens when the model is poorly specified: data is quadratic in x, but␣
↪the fit assumed it was linear
n = 10000
x = rand(n) # uniform
eps = randn(n) # normal, mean 0 variance 1, independent
y = 1 + x + 4*x**2 + eps
```

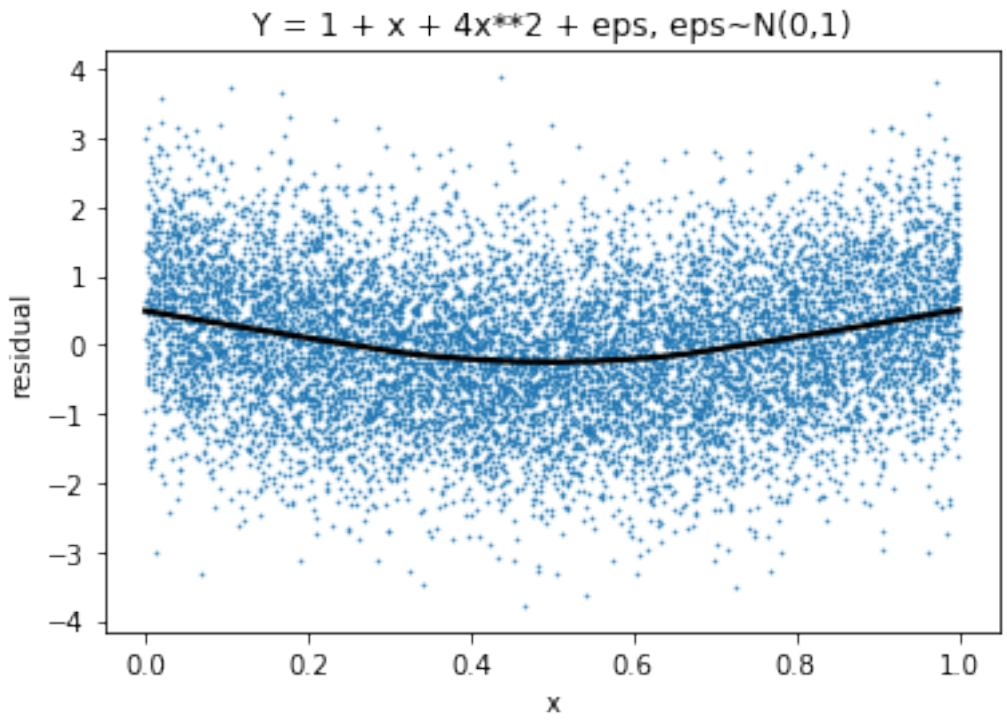
[9]:
```
plt.plot(x,y,'.',markersize=0.5)
plt.xlabel('x')
plt.ylabel('y')
plt.title('Y = 1 + x + 4x**2 + eps, eps~N(0,1)')
```

[9]:
```
Text(0.5, 1.0, 'Y = 1 + x + 4x**2 + eps, eps~N(0,1)')
```



[10]:
```
model = sm.OLS(y,sm.add_constant(x)).fit()
sns.regplot(x, model.resid, lowess=True, line_kws = {'color': 'black'},␣
 ↪scatter_kws = {'s' : 0.5})
plt.xlabel('x')
plt.ylabel('residual')
plt.title('Y = 1 + x + 4x**2 + eps, eps~N(0,1)')
#plt.plot(x,model.resid,'.',markersize=0.5)
```

[10]:
```
Text(0.5, 1.0, 'Y = 1 + x + 4x**2 + eps, eps~N(0,1)')
```

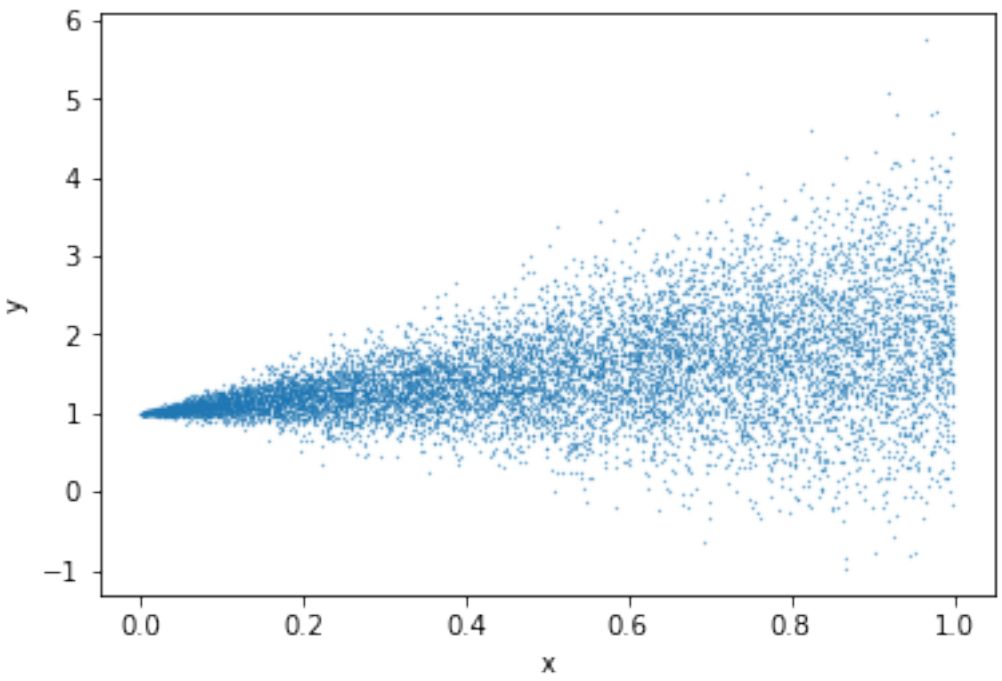


1.0.2 Example where variance of the noise depends on x

[11]:
```
n = 10000
x = rand(n) # uniform
eps = x*randn(n) # normal, mean 0 variance x^2, independent
y = 1 + x + eps
```

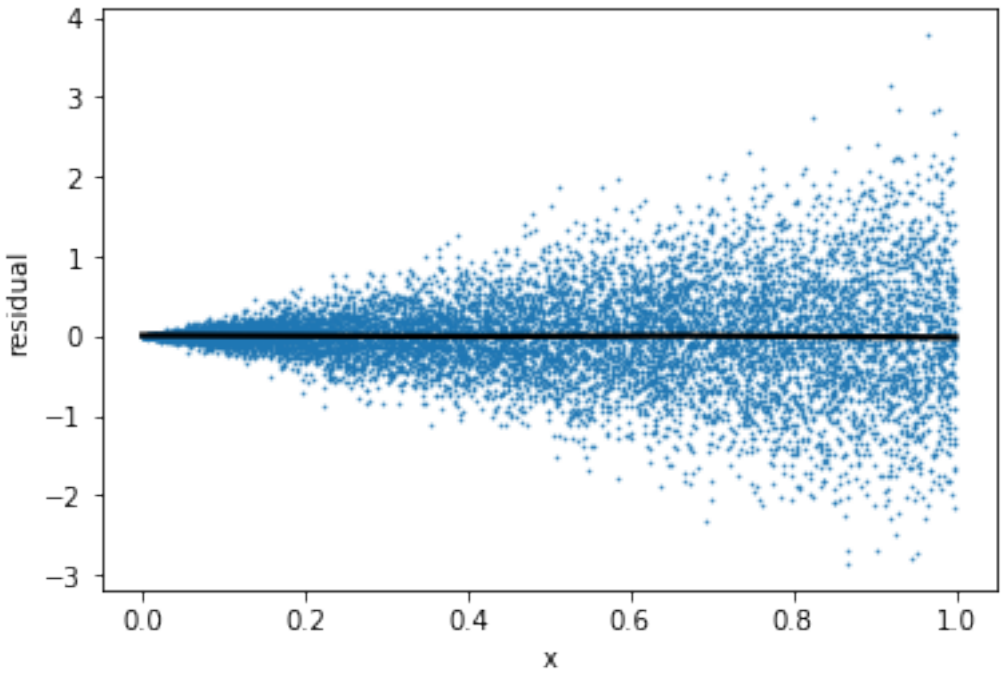
[12]:
```
plt.plot(x,y,'.',markersize=0.5)
plt.xlabel('x')
plt.ylabel('y')
```

[12]:
```
Text(0, 0.5, 'y')
```



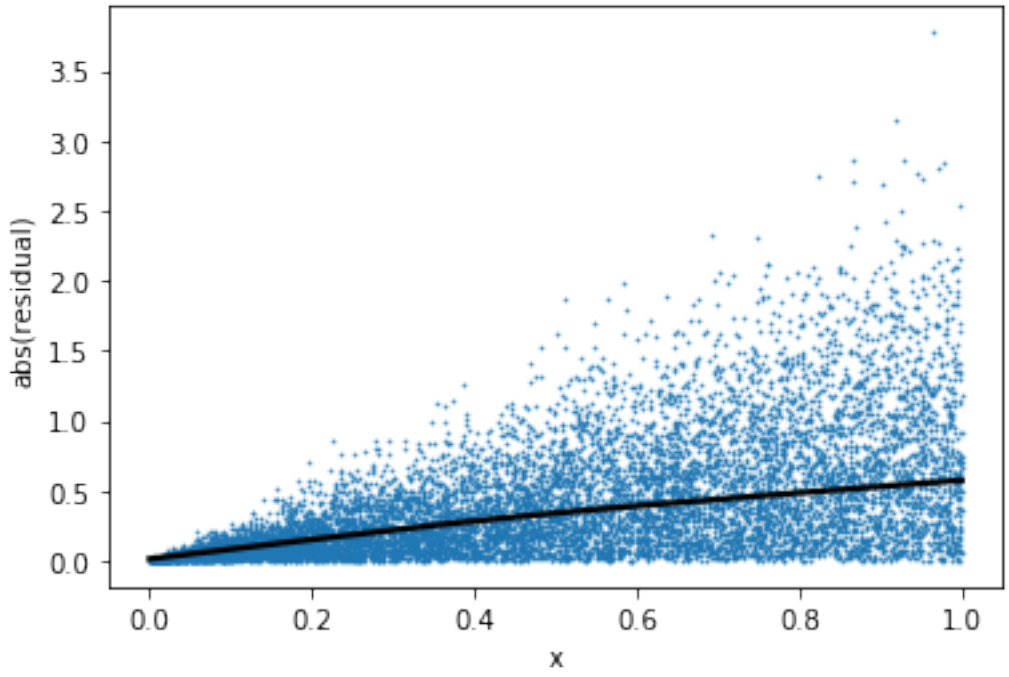
[13]:
```
model = sm.OLS(y,sm.add_constant(x)).fit()
sns.regplot(x, model.resid, lowess=True, line_kws = {'color': 'black'},␣
 ↪scatter_kws = {'s' : 0.5})
plt.xlabel('x')
plt.ylabel('residual')
```

[13]:
```
Text(0, 0.5, 'residual')
```



[14]:
```
sns.regplot(x, abs(model.resid), lowess=True, line_kws = {'color': 'black'},␣
↪scatter_kws = {'s' : 0.5})
plt.xlabel('x')
plt.ylabel('abs(residual)')
```

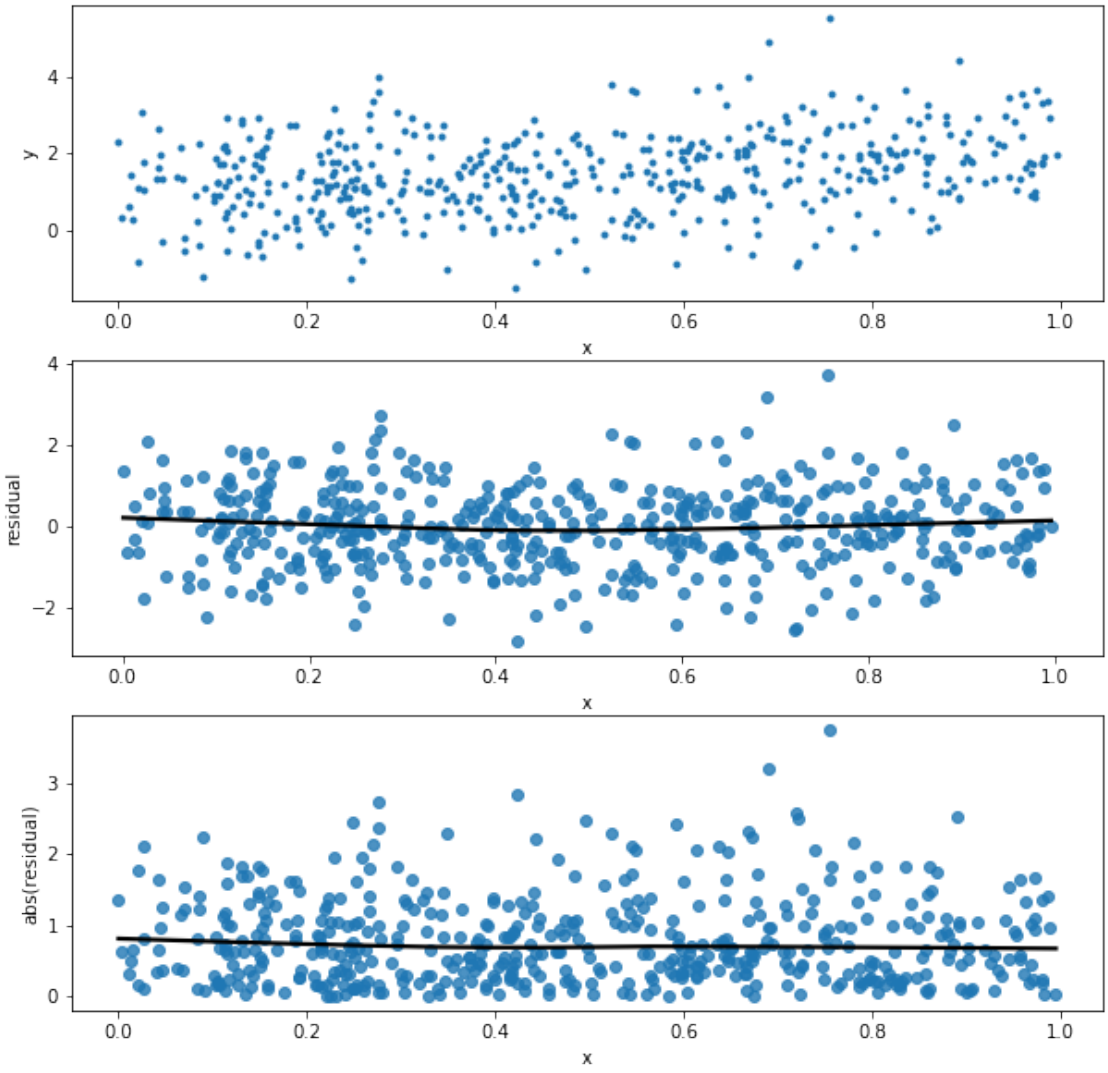
[14]:
```
Text(0, 0.5, 'abs(residual)')
```



1.0.3 Correctly specified model, fewer datapoints

[25]:
```
n = 500
x = rand(n) # uniform
eps = randn(n) # normal, mean 0 variance x^2, independent
y = 1 + x + eps
model = sm.OLS(y,sm.add_constant(x)).fit()

f, axs = plt.subplots(3, figsize=(10,10))

axs[0].plot(x,y,'.')
axs[0].set_ylabel('y')
axs[0].set_xlabel('x')

sns.regplot(x, model.resid, lowess=True, line_kws = {'color': 'black'},␣
 ↪ax=axs[1])
axs[1].set_ylabel('residual')
axs[1].set_xlabel('x')

sns.regplot(x, abs(model.resid), lowess=True, line_kws = {'color': 'black'}, \
            scatter_kws = {'color' : '#1f77b4' }, ax=axs[2])
axs[2].set_ylabel('abs(residual)')
axs[2].set_xlabel('x')
```

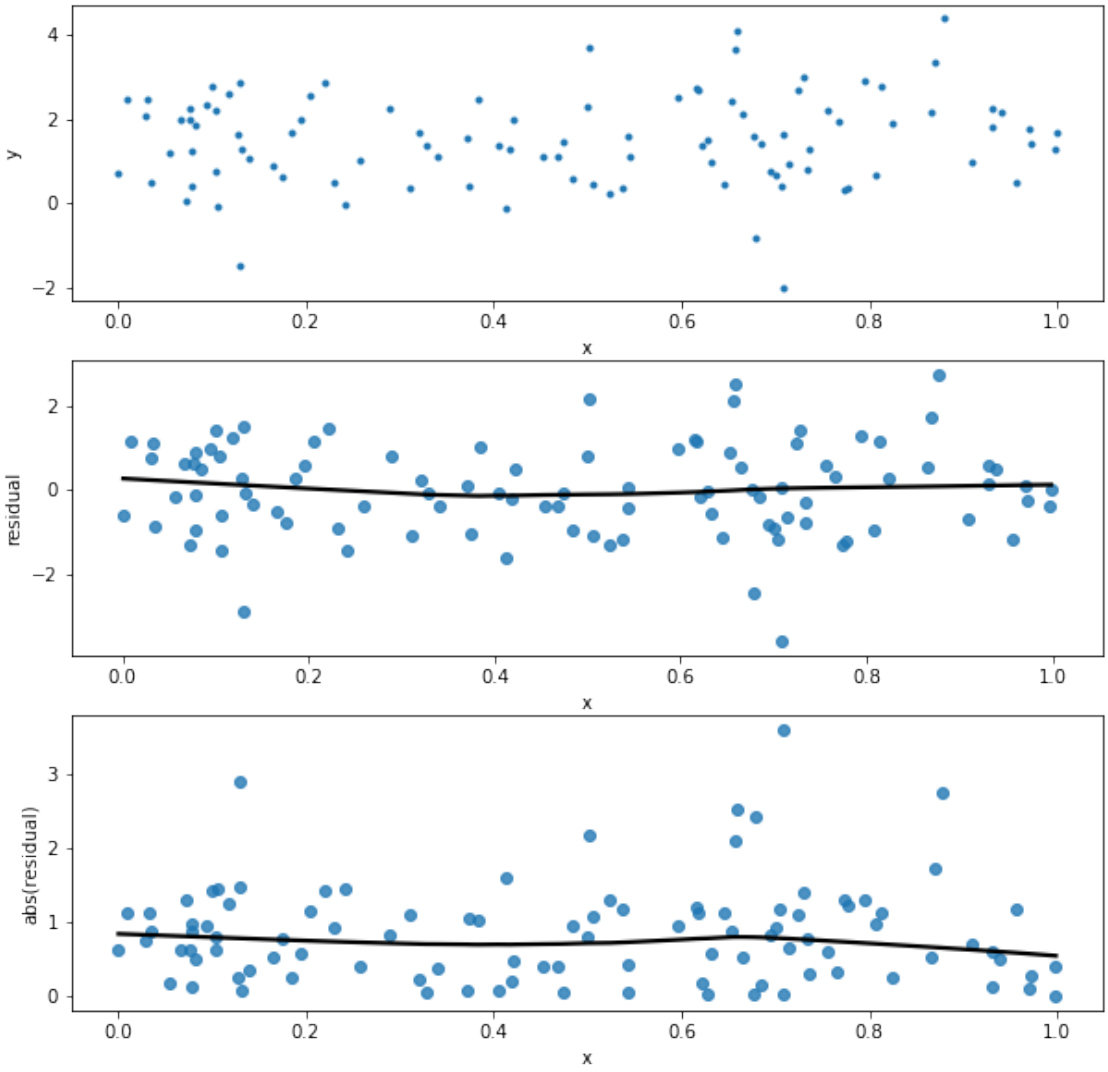
[25]: Text(0.5, 0, 'x')



[26]:
```
n = 100
x = rand(n) # uniform
eps = randn(n) # normal, mean 0 variance x^2, independent
y = 1 + x + eps
model = sm.OLS(y,sm.add_constant(x)).fit()

f, axs = plt.subplots(3, figsize=(10,10))

axs[0].plot(x,y,'.')
axs[0].set_ylabel('y')
axs[0].set_xlabel('x')
```

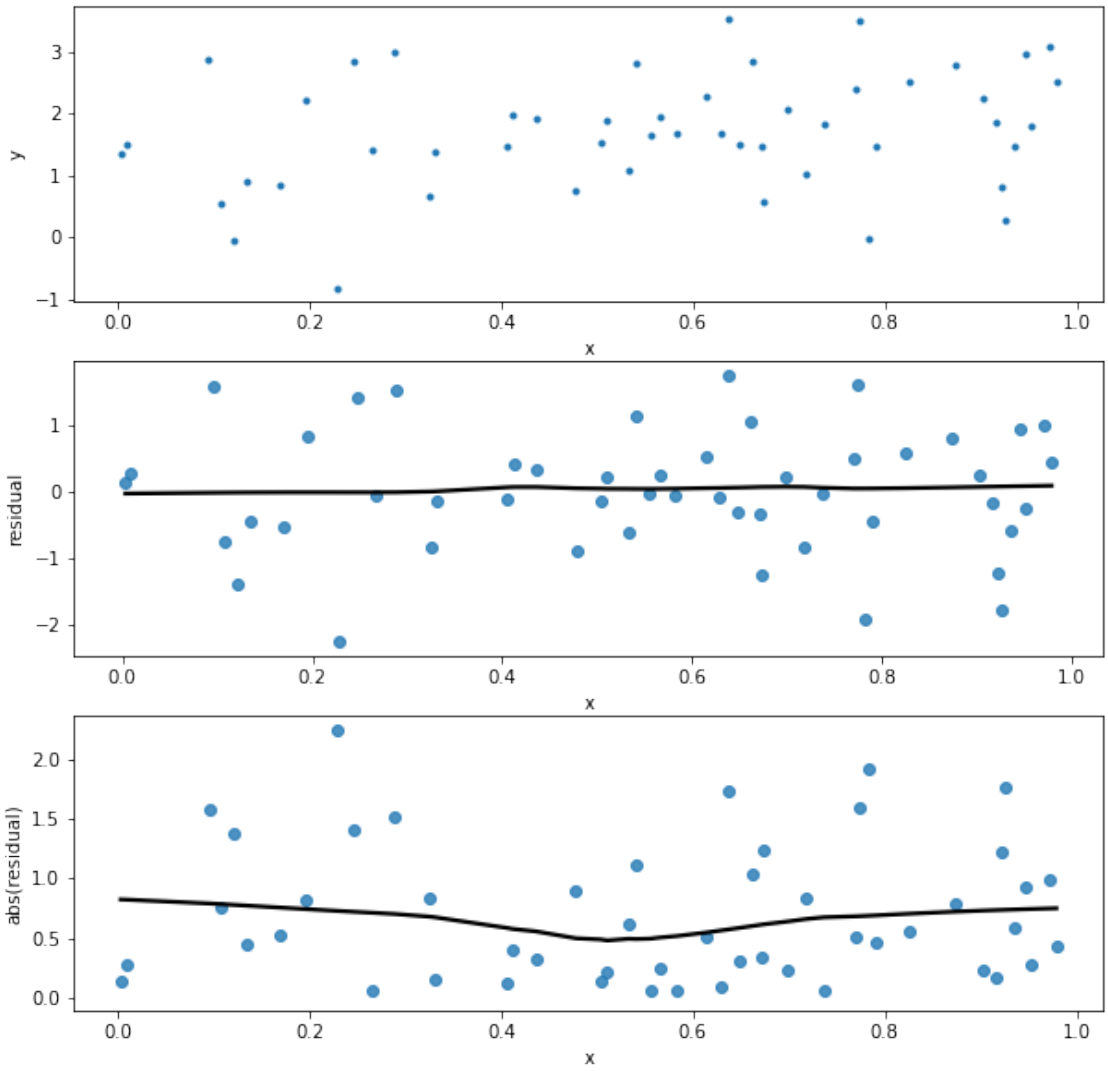
```
sns.regplot(x, model.resid, lowess=True, line_kws = {'color': 'black'},␣
 ↪ax=axs[1])
axs[1].set_ylabel('residual')
axs[1].set_xlabel('x')

sns.regplot(x, abs(model.resid), lowess=True, line_kws = {'color': 'black'}, \
            scatter_kws = {'color' : '#1f77b4' }, ax=axs[2])
axs[2].set_ylabel('abs(residual)')
axs[2].set_xlabel('x')
```

[26]: Text(0.5, 0, 'x')



[27]:
```
n = 50
x = rand(n) # uniform
eps = randn(n) # normal, mean 0 variance x^2, independent
y = 1 + x + eps
model = sm.OLS(y,sm.add_constant(x)).fit()

f, axs = plt.subplots(3, figsize=(10,10))

axs[0].plot(x,y,'.')
axs[0].set_ylabel('y')
axs[0].set_xlabel('x')

sns.regplot(x, model.resid, lowess=True, line_kws = {'color': 'black'},␣
 ↪ax=axs[1])
axs[1].set_ylabel('residual')
axs[1].set_xlabel('x')

sns.regplot(x, abs(model.resid), lowess=True, line_kws = {'color': 'black'}, \
            scatter_kws = {'color' : '#1f77b4' }, ax=axs[2])
axs[2].set_ylabel('abs(residual)')
axs[2].set_xlabel('x')
```

[27]:
```
Text(0.5, 0, 'x')
```

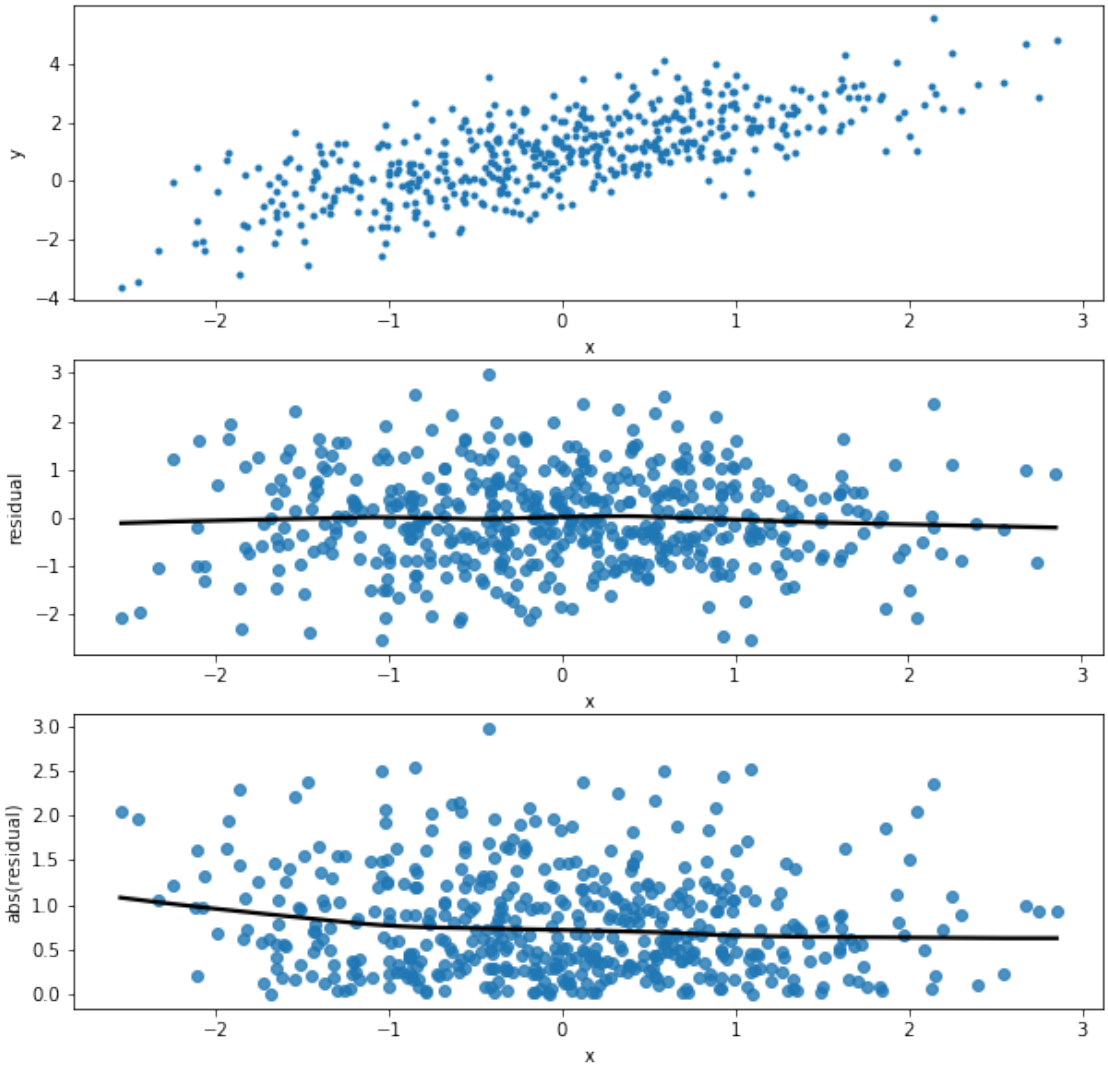


1.0.4 Correctly specified model, different distribution over x

[28]:
```
n = 500
x = randn(n) # normal
eps = randn(n) # normal, mean 0 variance x^2, independent
y = 1 + x + eps
model = sm.OLS(y,sm.add_constant(x)).fit()

f, axs = plt.subplots(3, figsize=(10,10))

axs[0].plot(x,y,'.')
axs[0].set_ylabel('y')
axs[0].set_xlabel('x')
```

```
sns.regplot(x, model.resid, lowess=True, line_kws = {'color': 'black'},␣
 ↪ax=axs[1])
axs[1].set_ylabel('residual')
axs[1].set_xlabel('x')

sns.regplot(x, abs(model.resid), lowess=True, line_kws = {'color': 'black'}, \
            scatter_kws = {'color' : '#1f77b4' }, ax=axs[2])
axs[2].set_ylabel('abs(residual)')
axs[2].set_xlabel('x')
```

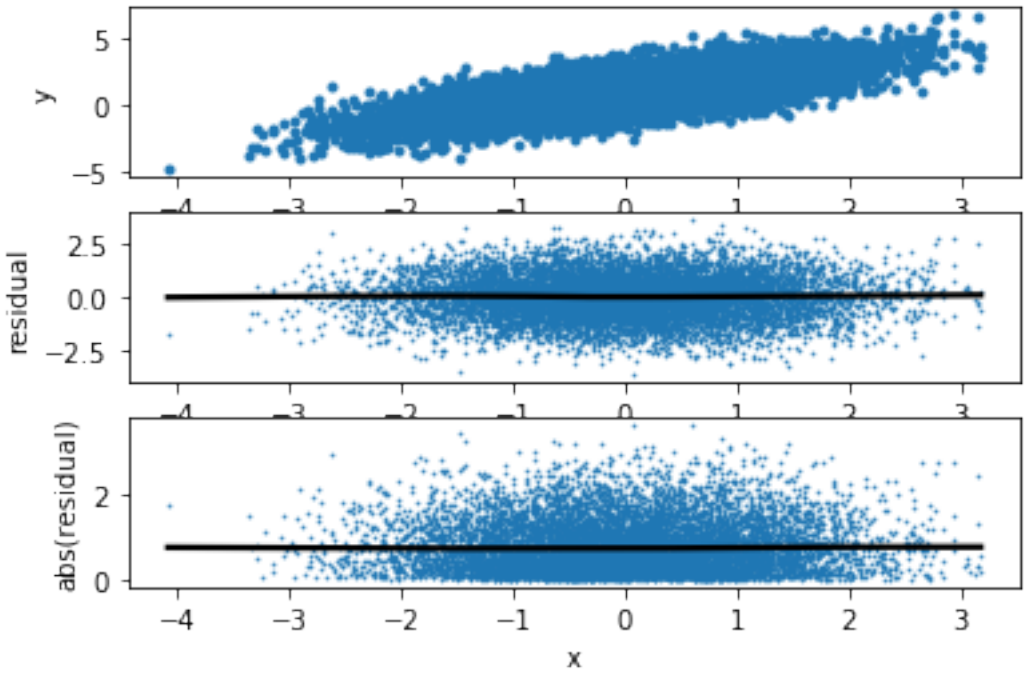
[28]: Text(0.5, 0, 'x')



1.0.5 Lecture questions

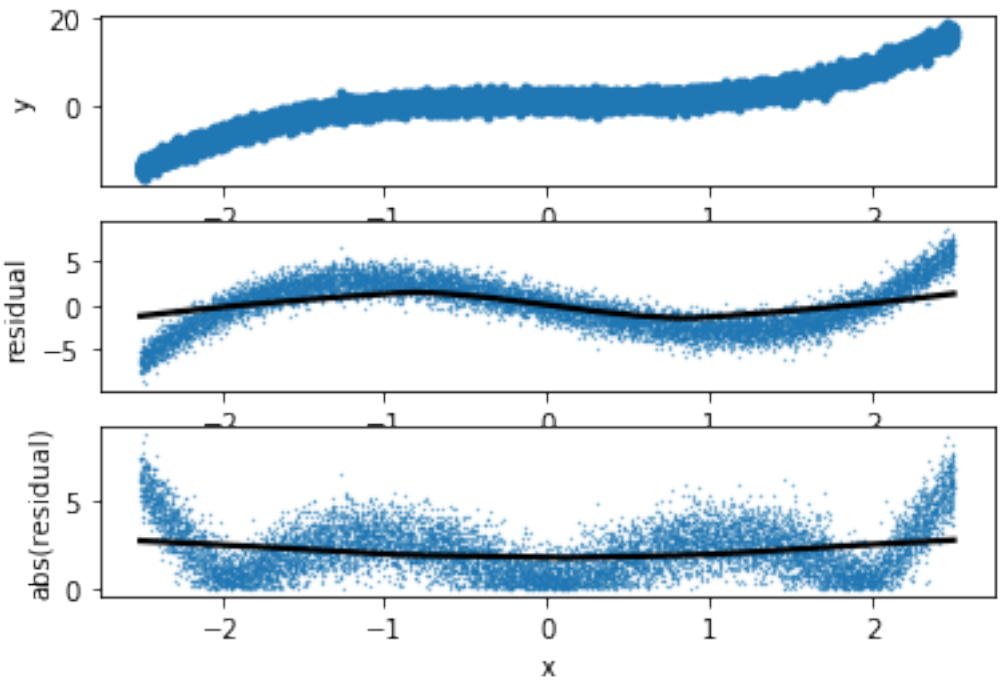
[76]:
```
n = 10000
x = randn(n)
eps = randn(n) # normal, mean 0 variance x^2, independent
y = 1 + x + eps
model = sm.OLS(y,sm.add_constant(x)).fit()

plt.subplot(3,1,1)
plt.plot(x,y,'.')
plt.ylabel('y')

plt.subplot(3,1,2)
sns.regplot(x, model.resid, lowess=True, line_kws = {'color': 'black'},␣
 ↪scatter_kws = {'s' : 0.5})
plt.ylabel('residual')

plt.subplot(3,1,3)
sns.regplot(x, abs(model.resid), lowess=True, line_kws = {'color': 'black'},␣
 ↪scatter_kws = {'s' : 0.5})
plt.xlabel('x')
plt.ylabel('abs(residual)')
```

[76]:
```
Text(0, 0.5, 'abs(residual)')
```



[81]:
```
n = 10000
x = 5*(rand(n)-.5) # uniform
eps = randn(n) # normal, mean 0 variance x^2, independent
y = 1 + x**3 + eps
model = sm.OLS(y,sm.add_constant(x)).fit()

plt.subplot(3,1,1)
plt.plot(x,y,'.')
plt.ylabel('y')

plt.subplot(3,1,2)
sns.regplot(x, model.resid, lowess=True, line_kws = {'color': 'black'},␣
↪scatter_kws = {'s' : 0.1})
plt.ylabel('residual')

plt.subplot(3,1,3)
sns.regplot(x, abs(model.resid), lowess=True, line_kws = {'color': 'black'},␣
↪scatter_kws = {'s' : 0.1})
plt.xlabel('x')
plt.ylabel('abs(residual)')
```

[81]:
```
Text(0, 0.5, 'abs(residual)')
```



2 Method 2: Autocorrelation Plots

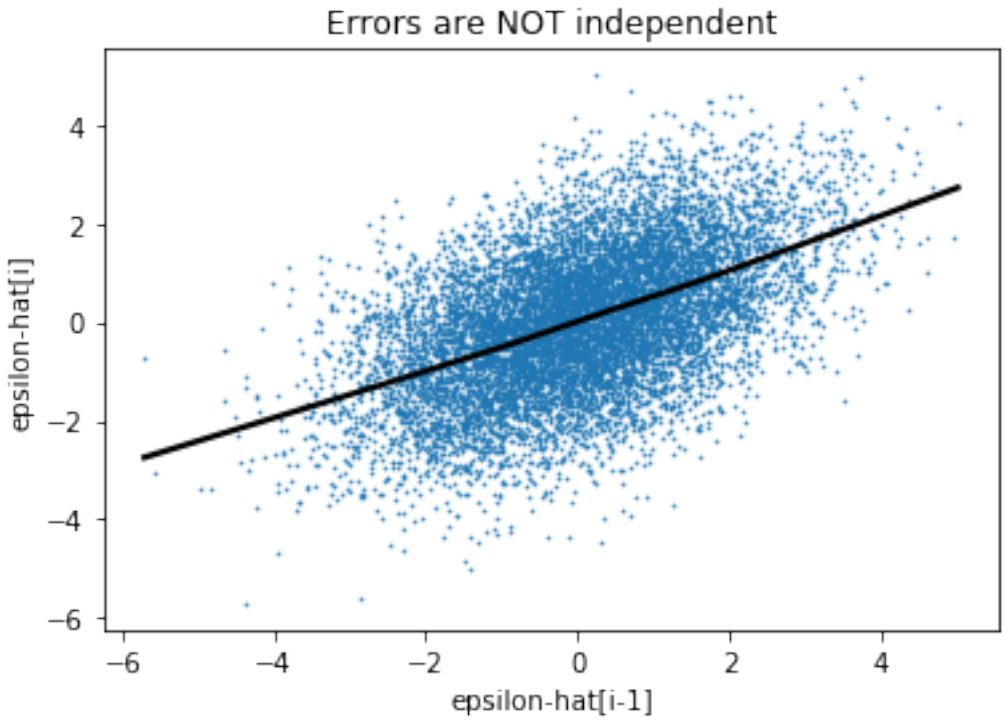
[38]:
```
# generate data where the errors are independent
n = 10000 # number of observations
eps = randn(n) # independent normal(0,1)
x = 10*rand(n) # uniform(0,10)
y = x + eps

# form and fit model
model = sm.OLS(y, x).fit()
resid = model.resid
sns.regplot(resid[:-1], resid[1:], lowess=True, \
            line_kws = {'color': 'black'}, scatter_kws = {'s' : 0.5})
plt.xlabel('epsilon-hat[i-1]')
plt.ylabel('epsilon-hat[i]')
plt.title('Errors are independent')
```

[38]:
```
Text(0.5, 1.0, 'Errors are independent')
```

[39]:
```
# generate data where errors are independent
n   = 10000 # number of observations
a   = 1     # use this to control the correlation
w   = randn(n+1) # independent normal(0,1)
eps = w[:-1] + a*w[1:] # normal, not independent
```

```
  x   = 10*rand(n) # uniform(0,10)
  y   = x + eps

# form and fit model
model = sm.OLS(y, x).fit()
resid = model.resid
sns.regplot(resid[:-1], resid[1:], lowess=True, \
            line_kws = {'color': 'black'}, scatter_kws = {'s' : 0.5})
plt.xlabel('epsilon-hat[i-1]')
plt.ylabel('epsilon-hat[i]')
plt.title('Errors are NOT independent')
```

[39]: `Text(0.5, 1.0, 'Errors are NOT independent')`

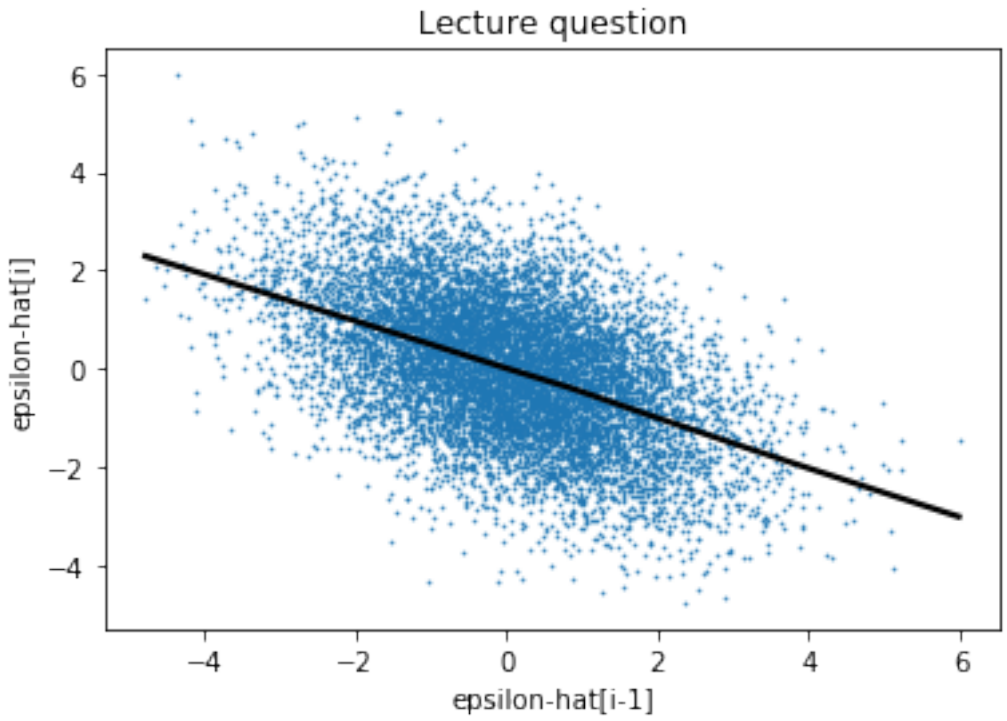


2.0.1 Lecture question

[41]:
```
  # generate data where errors are independent
  n   = 10000 # number of observations
  a   = -1     # use this to control the correlation
  w   = randn(n+1) # independent normal(0,1)
  eps = w[:-1] + a*w[1:] # normal, not independent
  x   = 10*rand(n) # uniform(0,10)
  y   = x + eps
```

```
# form and fit model
model = sm.OLS(y, x).fit()
resid = model.resid
sns.regplot(resid[:-1], resid[1:], lowess=True, \
            line_kws = {'color': 'black'}, scatter_kws = {'s' : 0.5})
plt.xlabel('epsilon-hat[i-1]')
plt.ylabel('epsilon-hat[i]')
plt.title('Lecture question')
```

[41]: Text(0.5, 1.0, 'Lecture question')

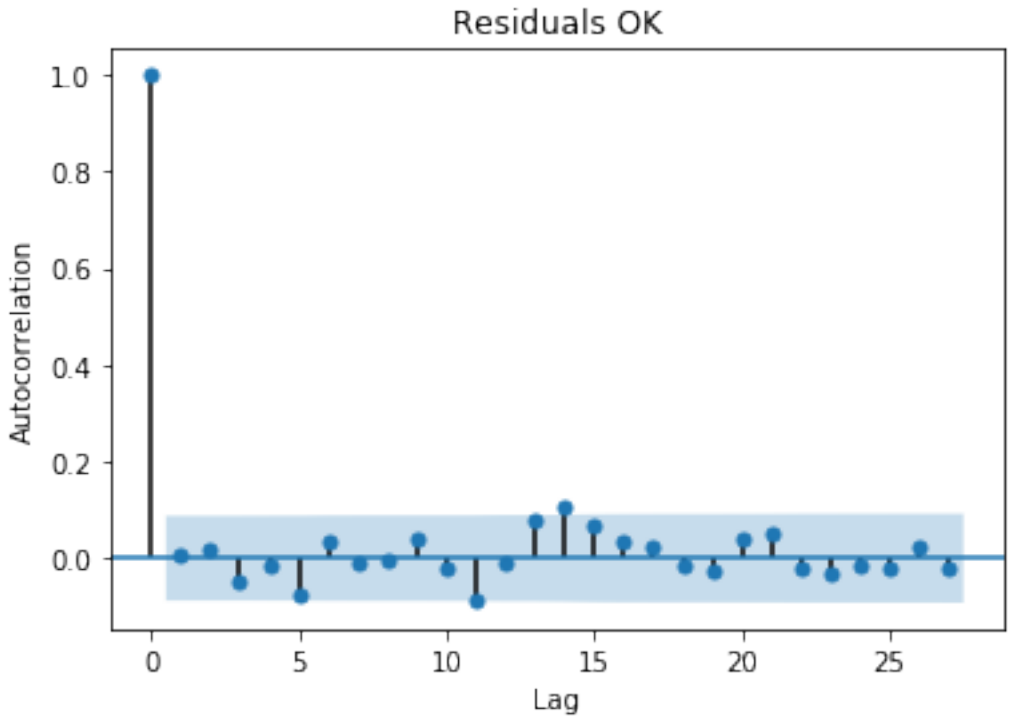


3 test mutual independence by plotting autocorrelations

[5]:
```
def indep_resid(n=500):
    eps = randn(n) # independent normal(0,1)
    x   = 10*rand(n) # uniform(0,10)
    y   = x + eps

    # form and fit model
    model = sm.OLS(y, x).fit()
    return x,y,model.resid
```

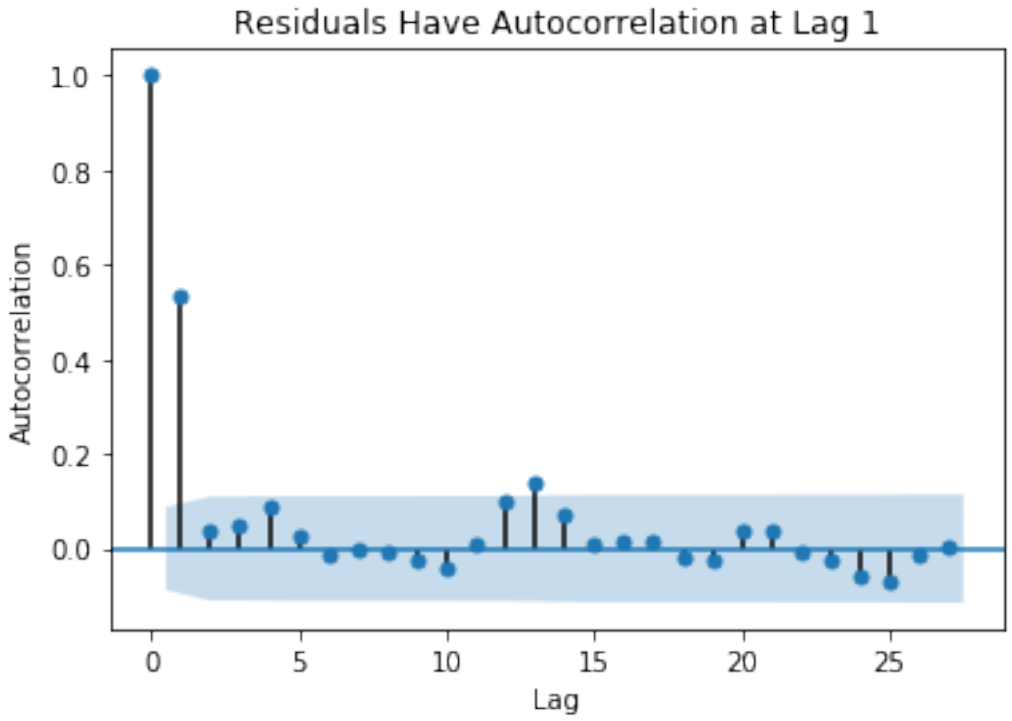
```
def dep_resid(n=500, a=1):
    w   = randn(n+1) # independent normal(0,1)
    eps = w[:-1] + a*w[1:] # normal, not independent
    x   = 10*rand(n) # uniform(0,10)
    y   = x + eps

    # form and fit model
    model = sm.OLS(y, x).fit()
    return x,y,model.resid
```

[19]:
```
import statsmodels.graphics.tsaplots as tsa
x,y,resid = indep_resid()
tsa.plot_acf(resid)
plt.xlabel('Lag')
plt.ylabel('Autocorrelation')
plt.title('Residuals OK')
plt.show()
```



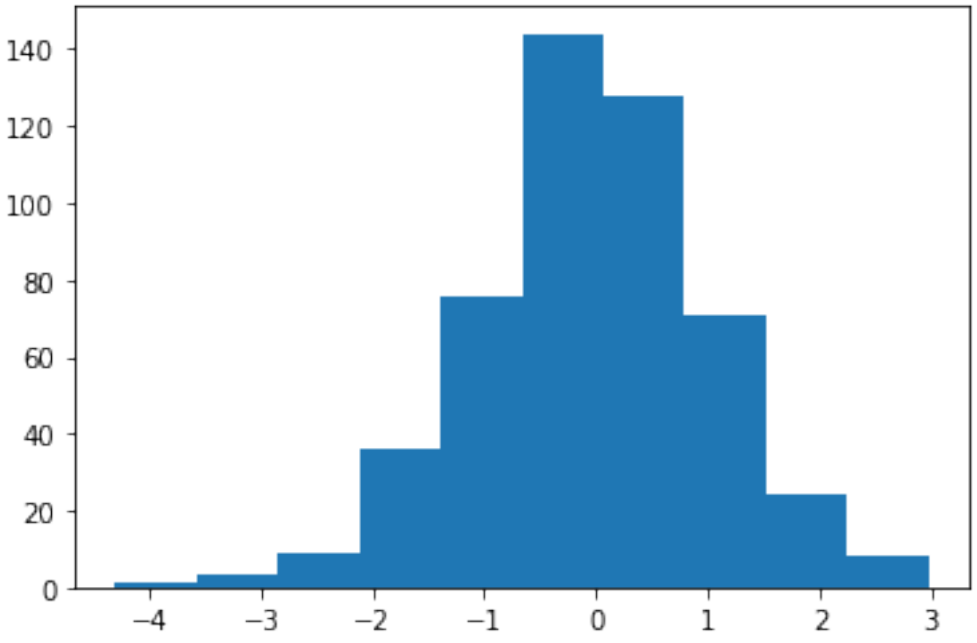
[20]:
```
x,y,resid = dep_resid()
tsa.plot_acf(resid)
plt.xlabel('Lag')
plt.ylabel('Autocorrelation')
```

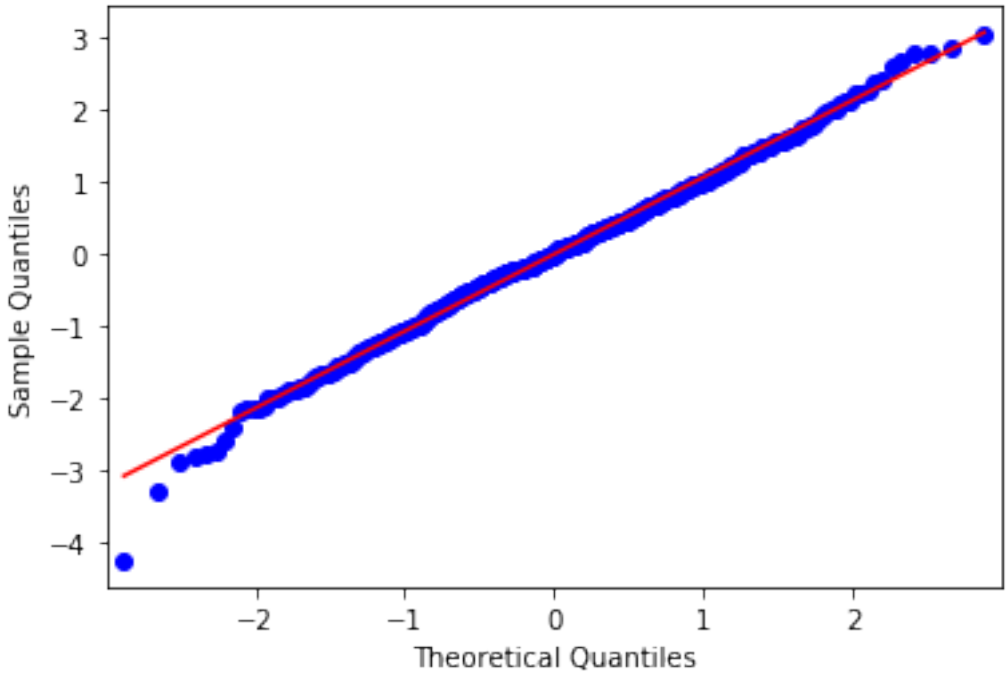
```
plt.title('Residuals Have Autocorrelation at Lag 1')
plt.show()
```



4 Method 3: QQ plots for assessing normality

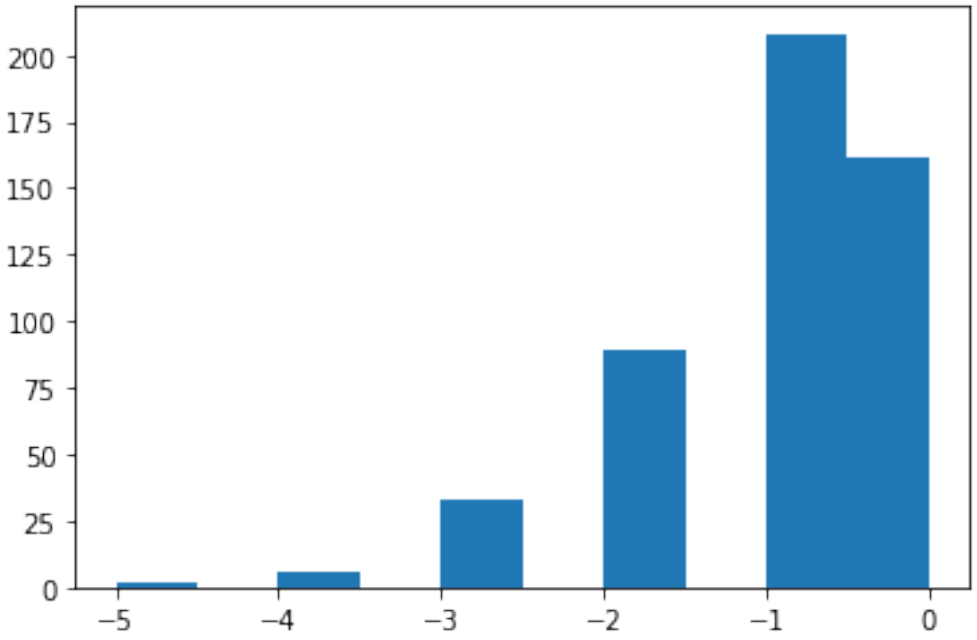
[21]:
```
n=500
eps = randn(n) # normal residuals
x = 10*rand(n)
y = x + eps
model = sm.OLS(y,x).fit()

# scipy.stats.probplot(data, dist="norm", plot=plt);
plt.hist(eps)
sm.qqplot(model.resid, line='r');
```

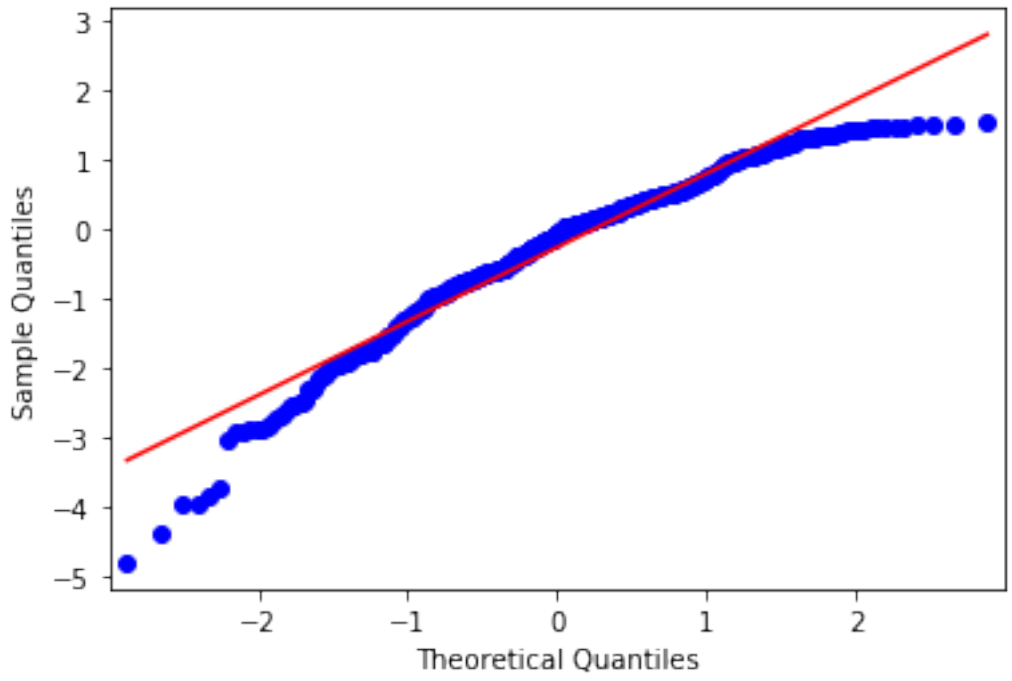




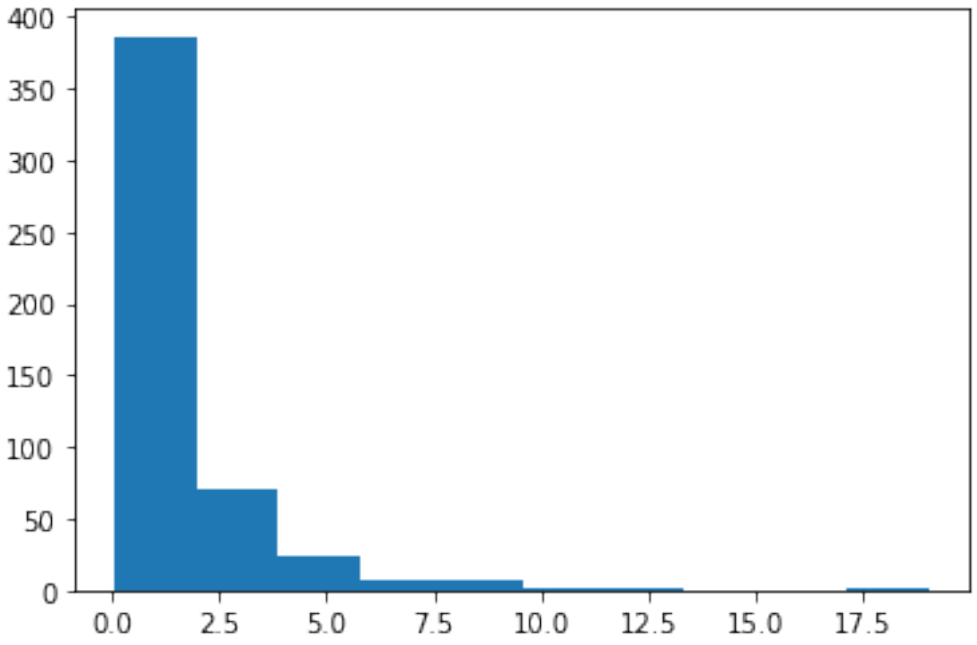
[29]:
```
n=500
eps = -poisson(1,n) # left-skewed residuals
x = 10*rand(n)
```

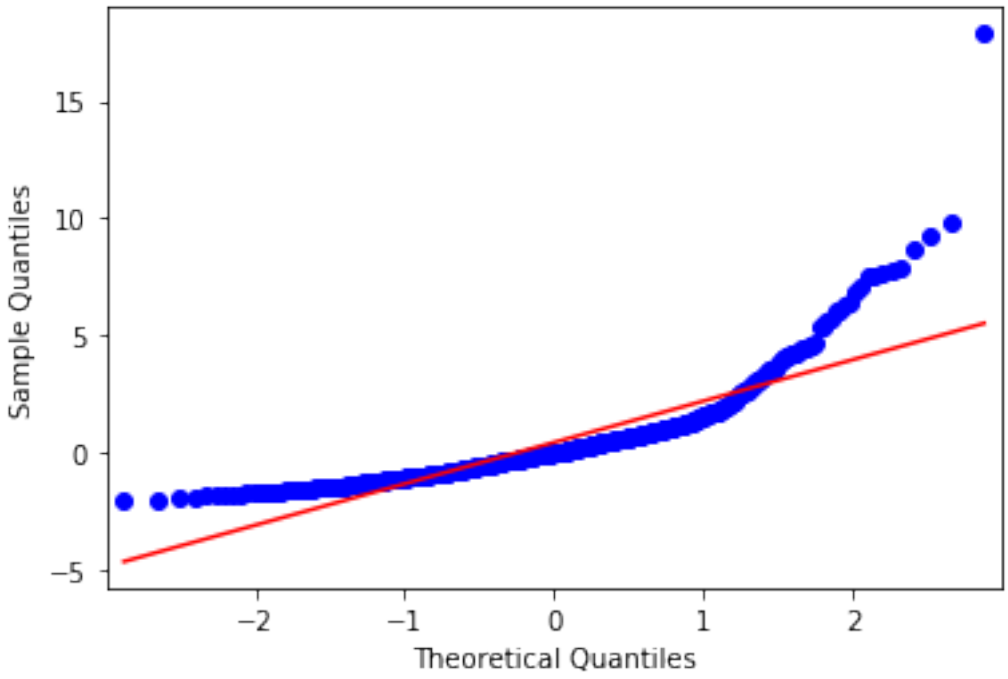
```
y = x + eps
model = sm.OLS(y,x).fit()

# scipy.stats.probplot(data, dist="norm", plot=plt);
plt.hist(eps)
sm.qqplot(model.resid, line='r')
plt.show()
```





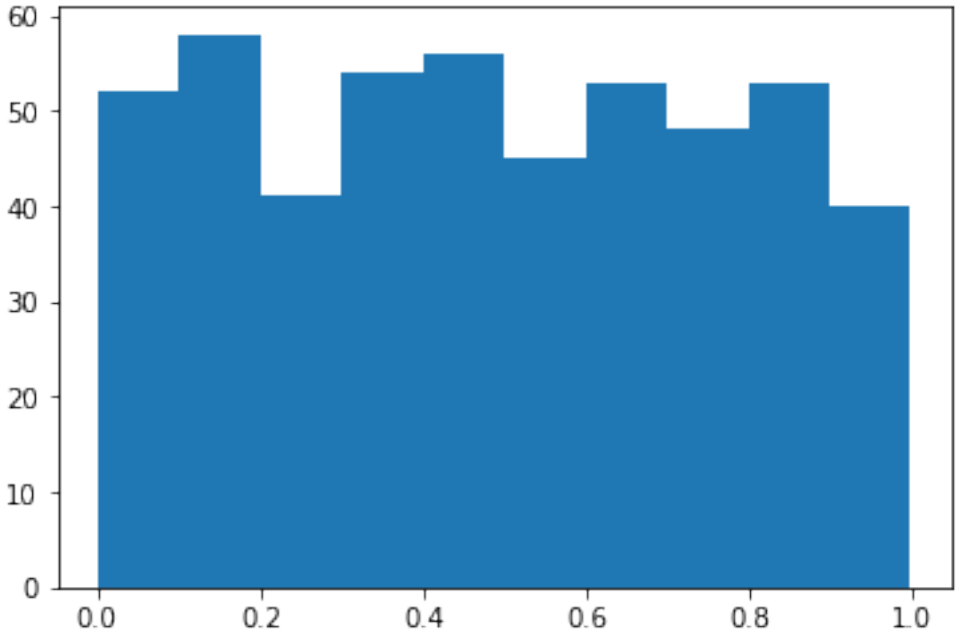
[30]:
```
n=500
eps = np.exp(randn(n)) # right-skewed residuals
x = 10*rand(n)
y = x + eps
model = sm.OLS(y,x).fit()

# scipy.stats.probplot(data, dist="norm", plot=plt);
plt.hist(eps)
sm.qqplot(model.resid, line='r');
```

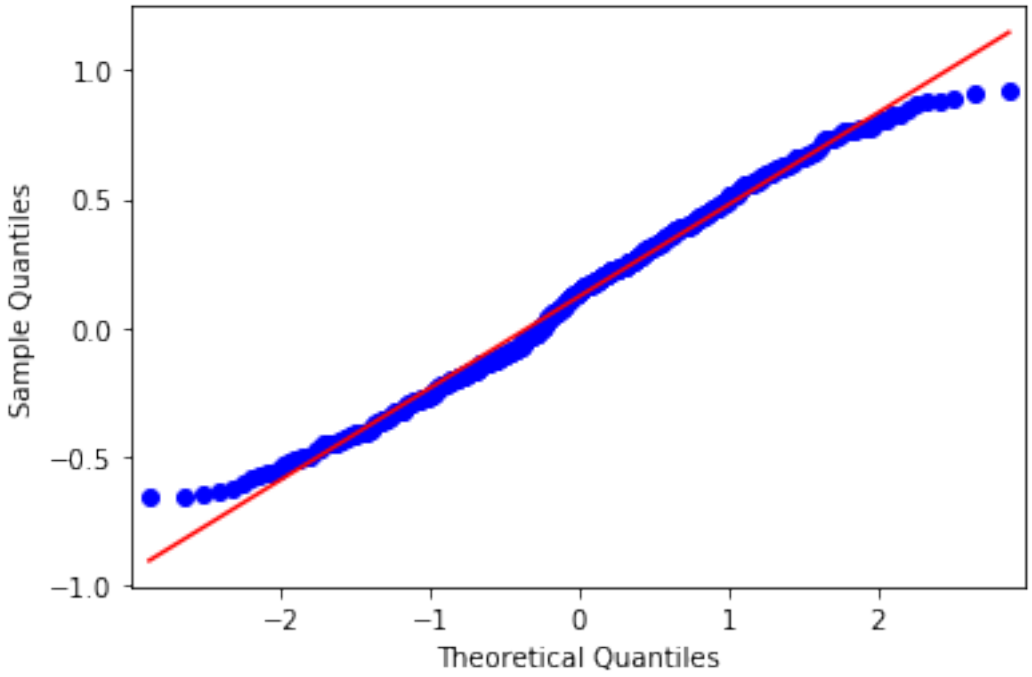




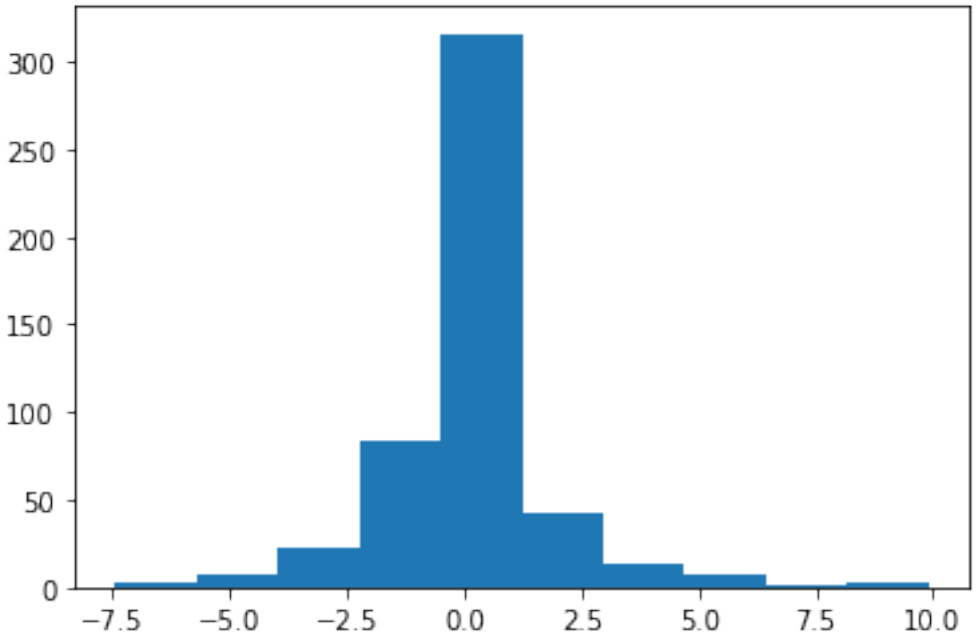
[31]:
```
n=500
eps = rand(n) # np.power(np.abs(randn(n)),.8) # light-tailed residuals
x = 10*rand(n)
```

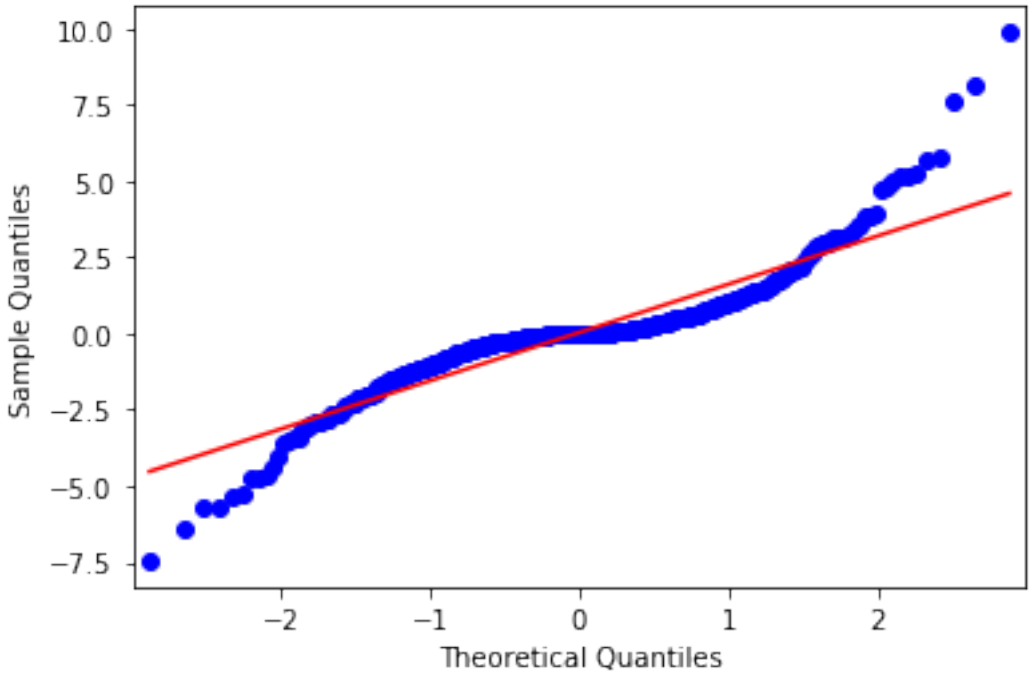
```
y = x + eps
model = sm.OLS(y,x).fit()

# scipy.stats.probplot(data, dist="norm", plot=plt);
plt.hist(eps)
sm.qqplot(model.resid, line='r');
```





[32]:
```
n=500
eps = randn(n)**2 * np.sign(randn(n)) # heavy-tailed residuals
x = 10*rand(n)
y = x + eps
model = sm.OLS(y,x).fit()

# scipy.stats.probplot(data, dist="norm", plot=plt);
plt.hist(eps)
sm.qqplot(model.resid, line='r');
```

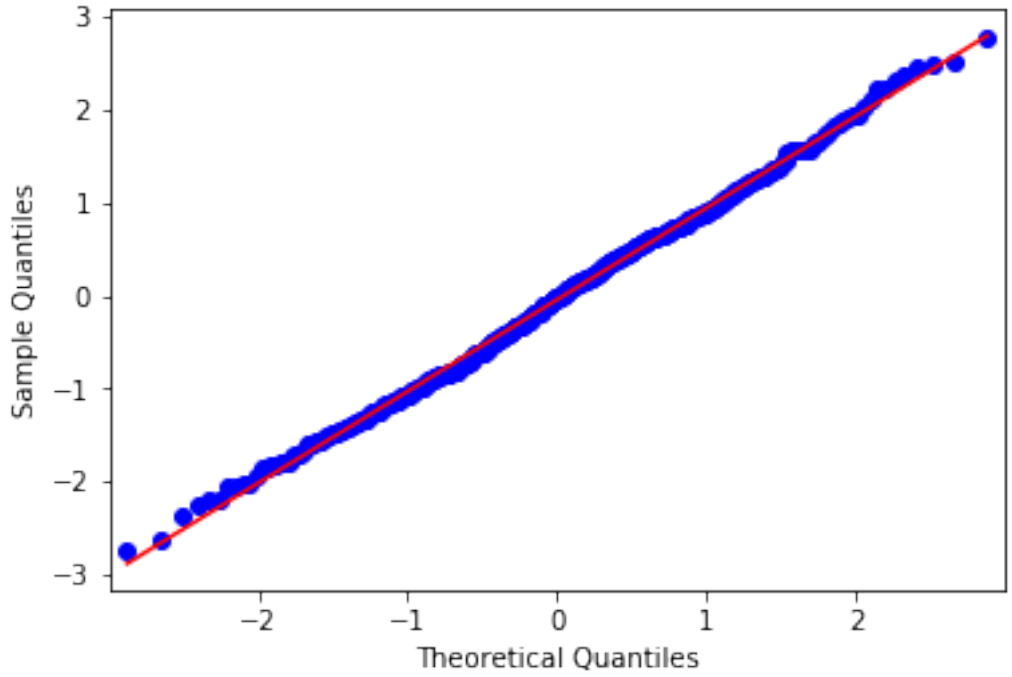


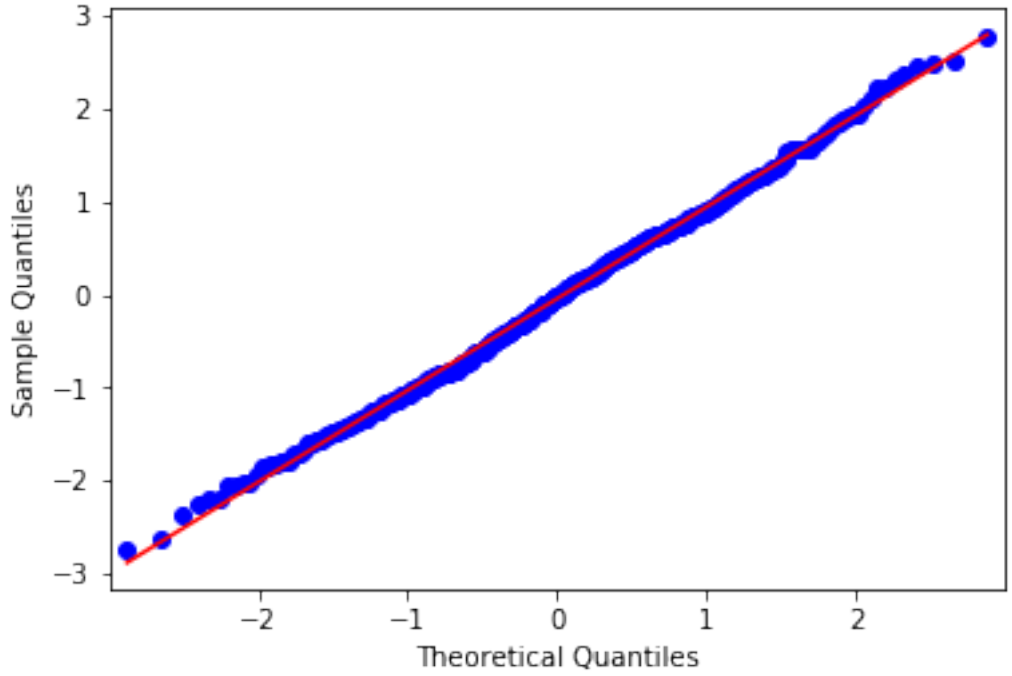


[36]:
```
n=500
eps = randn(n) # normal residuals x = 10*rand(n)
y = x + eps
```

```
model = sm.OLS(y,x).fit()
sm.qqplot(model.resid , line='r')
```

[36]:





[ ]: