ORIE 3120: Practical Tools for OR, DS, and ML

Linear Regression Theory and Residual Analysis (Lecture 12)

Professor Frazier

2022

Outline

Understanding confidence intervals and p-values

- Formula for our slope estimator, $\hat{\beta}_1$
- Distribution of our slope estimator, $\hat{\beta}_1$
- Confidence Intervals
- P-values

Visual inspection of residuals

- Method 1: Plot residuals vs. covariate
- Method 2: Autocorrelation plots
- Method 3: Normal probability plots

Goals for this segment

Goal: In the context of single-variable linear regression, focusing on $\hat{\beta}_1$, understand:

- how setting the derivative of the sum of squared errors to 0 gives a formula for $\hat{\beta}_1$ in terms of the data
- that $\hat{\beta}_1$ is random because the data has random noise in it
- what a confidence interval is and how it is computed
- what a p-value is and how it is computed

These same ideas apply to multi-variable linear regression and to all $\hat{\beta}_j$

Deriving least-squares estimates

Model:

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$$

Least squares: (all sums are i from 1 to n)

$$\min_{\hat{\beta}_0, \hat{\beta}_1} \sum (Y_i - \hat{\beta}_0 - \hat{\beta}_1 X_i)^2$$

To minimize, set derivative to 0:

$$0 = \frac{\partial}{\partial \hat{\beta}_0} \sum (Y_i - \hat{\beta}_0 - \hat{\beta}_1 X_i)^2$$

$$0 = \frac{\partial}{\partial \hat{\beta}_1} \sum (Y_i - \hat{\beta}_0 - \hat{\beta}_1 X_i)^2$$

Evaluating the derivatives:

$$0 = \sum (Y_i - \hat{\beta}_0 - \hat{\beta}_1 X_i)(-2)$$

$$0 = \sum (Y_i - \hat{\beta}_0 - \hat{\beta}_1 X_i)(-2X_i)$$

Deriving least-squares estimates, continued

Putting these equations together:

$$\begin{pmatrix} n & \sum X_i \\ \sum X_i & \sum X_i^2 \end{pmatrix} \begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{pmatrix} = \begin{pmatrix} \sum Y_i \\ \sum Y_i X_i \end{pmatrix}$$

$$\begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{pmatrix} = \frac{1}{n\sum X_i^2 - (\sum X_i)^2} \begin{pmatrix} \sum X_i^2 & -\sum X_i \\ -\sum X_i & n \end{pmatrix} \begin{pmatrix} \sum Y_i \\ \sum Y_i X_i \end{pmatrix}$$

Expressions for $\hat{\beta}_0$ and $\hat{\beta}_1$

After some algebra:

$$\hat{\beta}_1 = \frac{\sum Y_i(X_i - \overline{X})}{\sum (X_i - \overline{X})^2}$$

$$\hat{\beta}_0 = \overline{Y} - \hat{\beta}_1 \overline{X}$$

What's coming next

- We have a formula for $\hat{\beta}_1$ in a problem with 1 covariate
- $\hat{\beta}_1$ depends on Y_i, which is random
- We'll use the probability distribution of Y_i and the formula to figure out the probability distribution of $\hat{\beta}_1$
- We'll use this to explain how to create a confidence interval for β_1 and what this means
- We'll also use this to explain how the p-value for β_1 works

What is the distribution of our slope estimator, $\hat{\beta}_1$?

From last page:

$$\hat{\beta}_1 = \frac{\sum Y_i(X_i - \overline{X})}{\sum (X_i - \overline{X})^2}$$

Recall that:

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$$

Therefore,

$$\hat{\beta}_1 = \frac{\sum (\beta_0 + \beta_1 X_i + \epsilon_i)(X_i - \overline{X})}{\sum (X_i - \overline{X})^2} \quad (1)$$

What is the distribution of our slope estimator, $\hat{\beta}_1$?

Continuing from last page:

$$\begin{aligned}
\hat{\beta}_1 &= \frac{\sum(\beta_0 + \beta_1 X_i + \epsilon_i)(X_i - \overline{X})}{\sum(X_i - \overline{X})^2} && (2)\\
&= \frac{\beta_0\{\sum(X_i - \overline{X})\} + \beta_1\{\sum X_i(X_i - \overline{X})\} + \sum(X_i - \overline{X})\epsilon_i}{\sum(X_i - \overline{X})^2} && (3)\\
&= \beta_1 + \frac{\sum(X_i - \overline{X})\epsilon_i}{\sum(X_i - \overline{X})^2} && (4)
\end{aligned}$$

since

$$\sum(X_i - \overline{X}) = 0$$

and

$$\sum X_i(X_i - \overline{X}) = \sum(X_i - \overline{X})^2$$

The distribution of our slope estimator, $\hat{\beta}_1$

$$\hat{\beta}_1 = \beta_1 + \frac{\sum(X_i - \overline{X})\epsilon_i}{\sum(X_i - \overline{X})^2} = \beta_1 + \sum w_i \epsilon_i$$

where

$$w_i = \frac{X_i - \overline{X}}{\sum(X_i - \overline{X})^2}$$

Recall that

$$\epsilon_i \sim \text{Normal}(0, \sigma_\epsilon^2) \text{ and } \epsilon_1, \ldots, \epsilon_n \text{ independent}$$

Therefore, $\hat{\beta}_1$ is normal with this mean and variance:

$$E(\hat{\beta}_1) = \beta_1 \text{ (unbiased)}$$

$$\text{Var}(\hat{\beta}_1) = \sigma_\epsilon^2 \sum w_i^2 = \frac{\sigma_\epsilon^2}{\sum(X_i - \overline{X})^2}$$

Question 1

```
betahat1 = []
for i in range(10000):
    n = 20
    x = np.random.randn(n) # generates a series of n Normal(0,1) random variables
    eps = np.random.randn(n) # generates another series of n Normal(0,1) random variables
    y = 1 + 2 * x + eps
    x = sm.add_constant(x)
    model = sm.OLS(y, x).fit()
    betahat1.append(model.params[1]) # extracts betahat1 from the model and adds it to a list
```

Where will a histogram of `betahat1` be centered?

1. Centered at 0
2. Centered at 1
3. Centered at 2
4. Centered at 3

Question 2

```
betahat1 = []
for i in range(10000):
    n = 20
    x = np.random.randn(n) # generates a series of n Normal(0,1) random variables
    eps = np.random.randn(n) # generates another series of n Normal(0,1) random variables
    y = 1 + 2 * x + eps
    x = sm.add_constant(x)
    model = sm.OLS(y, x).fit()
    betahat1.append(model.params[1]) # extracts betahat1 from the model and adds it to a list
```

What shape will a histogram of `betahat1` be?

1. Symmetric
2. More mass to the right of the mode
3. More mass to the left of the mode

Question 3

```
betahat1 = []
for i in range(10000):
    n = 20
    x = np.random.randn(n) # generates a series of n Normal(0,1) random variables
    eps = np.random.randn(n) # generates another series of n Normal(0,1) random variables
    y = 1 + 2 * x + eps
    x = sm.add_constant(x)
    model = sm.OLS(y, x).fit()
    betahat1.append(model.params[1]) # extracts betahat1 from the model and adds it to a list
```

What will happen if we increase the variance of eps?

1. The histogram will get wider
2. The histogram will get narrow
3. The histogram will get narrow
4. The histogram will shift right
5. The histogram will shift left
6. The histogram won't change

Question 4

```
betahat1 = []
for i in range(10000):
    n = 20
    x = np.random.randn(n) # generates a series of n Normal(0,1) random variables
    eps = np.random.randn(n) # generates another series of n Normal(0,1) random variables
    y = 1 + 2 * x + eps
    x = sm.add_constant(x)
    model = sm.OLS(y, x).fit()
    betahat1.append(model.params[1]) # extracts betahat1 from the model and adds it to a list
```

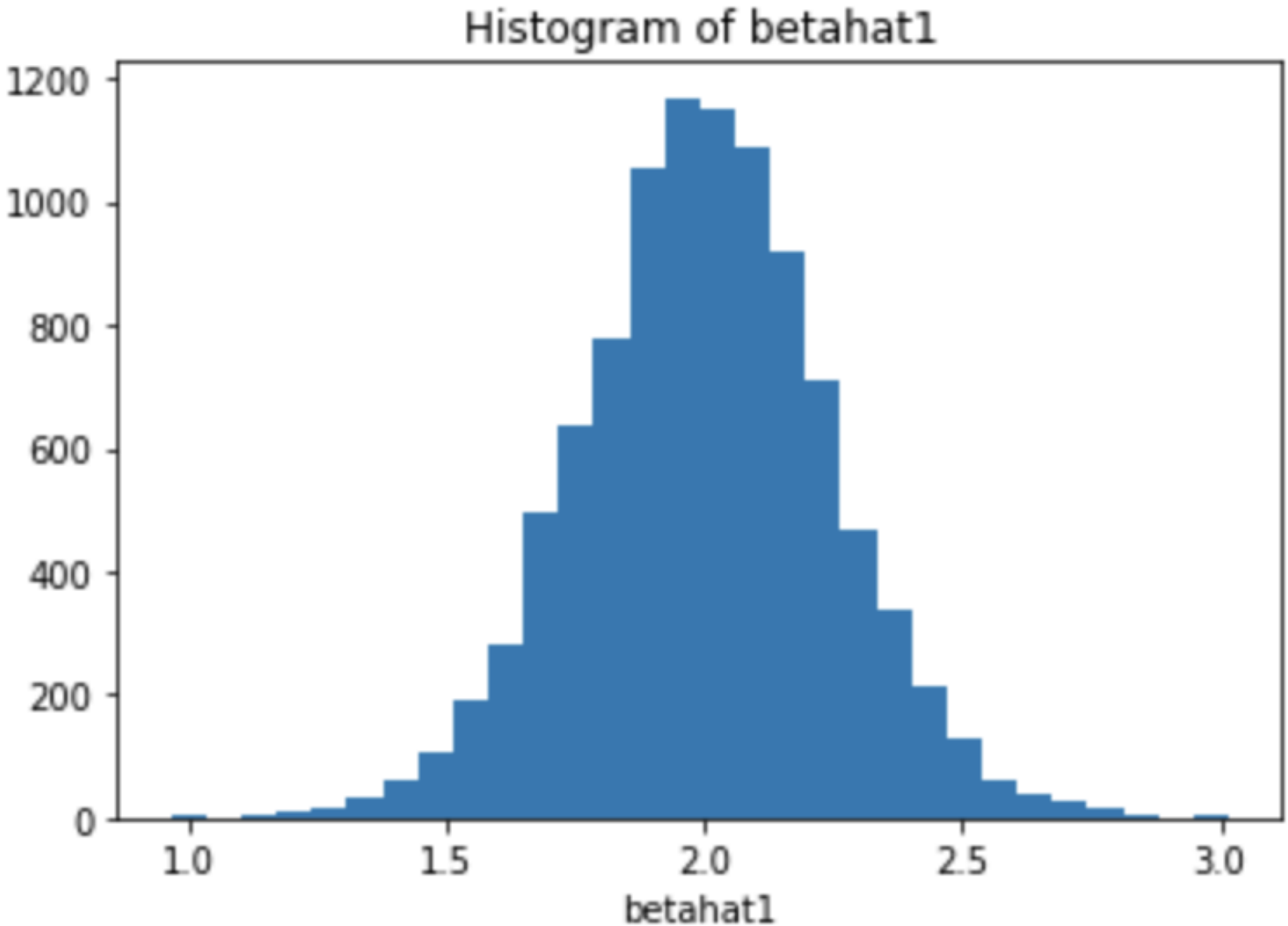
What will happen if we increase n?

1. The histogram will get wider
2. The histogram will get narrow
3. The histogram will get narrow
4. The histogram will shift right
5. The histogram will shift left
6. The histogram won't change

For discussion

What will happen if we increase the variance of x?

Distribution of $\hat{\beta}_1$



Confidence Interval

A $100(1-\alpha)\%$ confidence interval for β_1 is an interval that:

- depends on the data, so is random
- contains β_1 with probability $100(1-\alpha)\%$

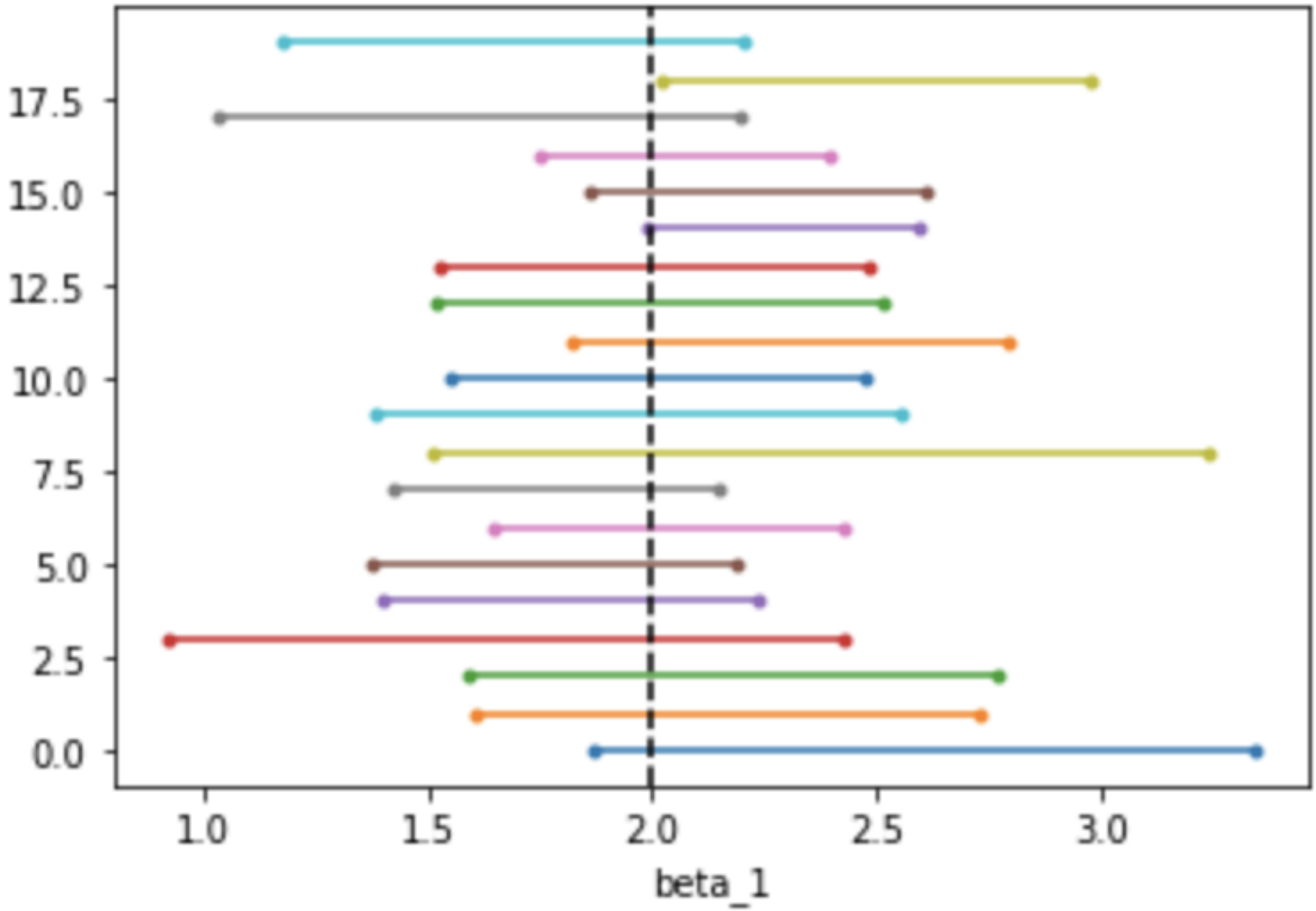
Usually $\alpha = 0.05$. In this case the interval contains β_1 with probability 95%.

Confidence Interval

To create a confidence interval for β_1, python takes the interval that is:

- centered at $\hat{\beta}_1$
- has a width proportional to an estimate of $\sqrt{\mathrm{Var}(\hat{\beta}_1)}$.

Confidence Interval



Confidence Interval

To create a confidence interval for β_1:

- Get an estimate $\hat{\sigma}_\epsilon^2$ of σ_ϵ^2
- Then $S^2 = \frac{\hat{\sigma}_\epsilon^2}{\sum(X_i - \overline{X})^2}$ is an estimate of $\mathrm{Var}(\hat{\beta}_1)$
- Let $W = (\beta_1 - \hat{\beta}_1)/S$. This has a known distribution (a student-t distribution with $n-2$ degrees of freedom).
- Our $100(1-\alpha)\%$ confidence interval is $\hat{\beta}_1 \pm cS$, for a constant c that depends on α and n
- Then $\beta_1 \in [\hat{\beta}_1 - cS, \hat{\beta}_1 + cS]$ if and only if $W \in [-c, c]$. c is chosen so that this happens with probability $100(1-\alpha)\%$.
- To make $P(W \in [-c, c]) = 1 - \alpha$, python sets c to the inverse cdf at $\alpha/2$ of the student t distribution with $n-2$ degrees of freedom

P-values

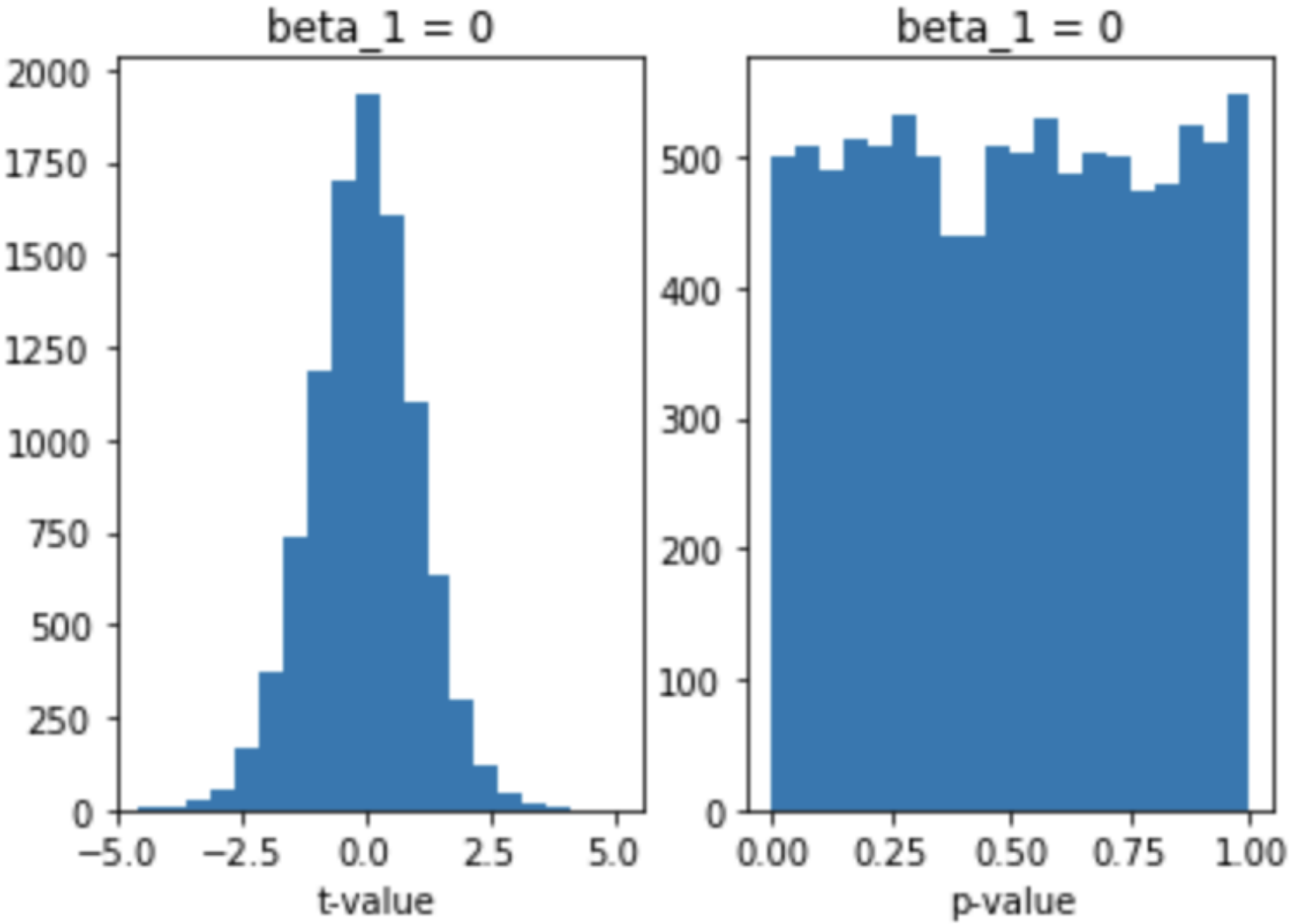
If $\beta_1 = 0$:

- then $\hat{\beta}_1$ is normal with mean 0 and variance $\mathrm{Var}(\hat{\beta}_1)$.
- The t-value $t = \hat{\beta}_1 / S$ is distributed according to a student-t distribution with $n - 2$ degrees of freedom.

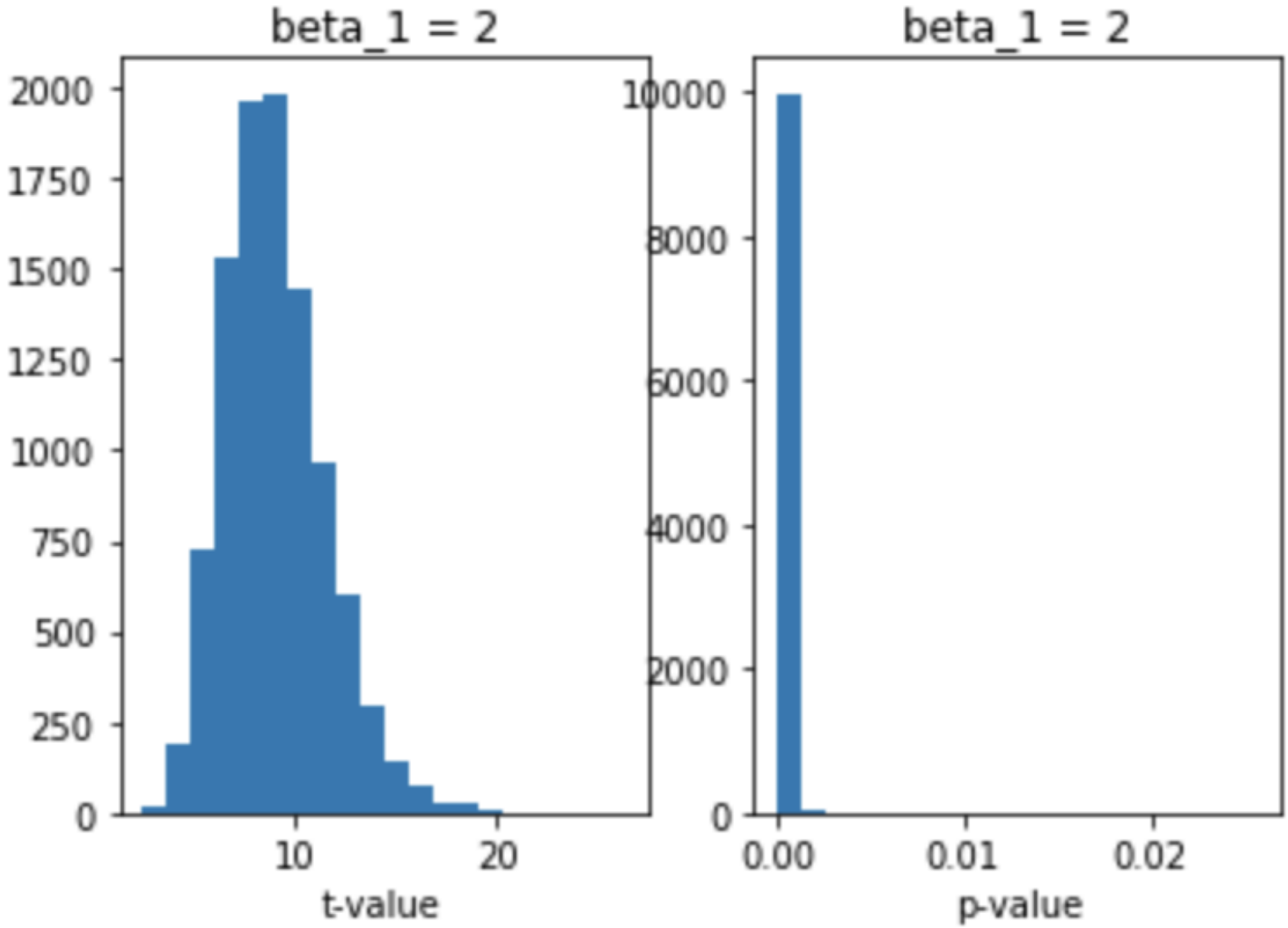
The p-value is the probability under $\beta_1 = 0$ of seeing a t-value as big or bigger (in an absolute value sense) than the one that we actually saw.

If we see a t-value that is really big compared to what we would be likely to see if $\beta_1 = 0$, then this is a sign that β_1 might not be 0. In this situation, we'll see a small p-value.

t-values and p-values generated from $\beta_1 = 0$



t-values and p-values generated from $\beta_1 = 2$



p-values have a uniform distribution under the null ($\beta_1 = 0$)

- Let F_t be the cdf of the absolute value of a student-t distributed random variance with $n - 2$ degrees of freedom.
- When $\beta_1 = 0$, the absolute value of the t-value, $|t|$, has cdf F_t.
- $1 - F_t(x)$ is the probability that the absolute value of a student-t is above x.
- The p-value is $1 - F_t(|t|)$
- In general, if you have a random variable Z with a cdf F_Z, then $F_Z(Z)$ and $1 - F_Z(Z)$ are both uniform $(0, 1)$.

$F_Z(Z)$ is uniform

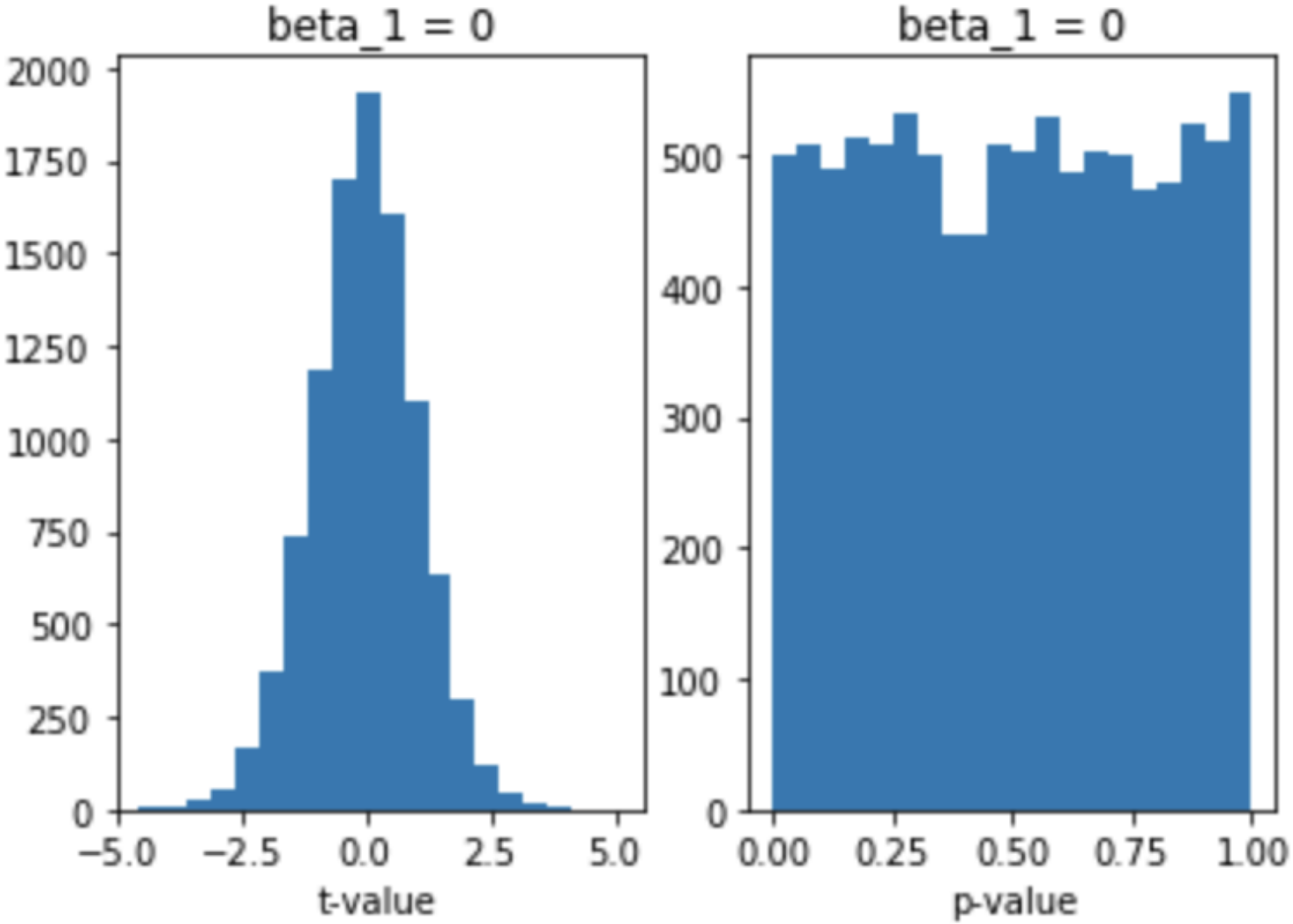
For numbers $u \in (0, 1)$ and a strictly increasing cdf F_Z,

$$P(F_Z(Z) <= u) = P(Z <= F_Z^{-1}(u)) = F_Z(F_Z^{-1}(u)) = u$$

This is the cdf of a uniform(0,1) distribution.

Since the density of the uniform(0,1) is symmetric around 0.5, $1 - F_Z(Z)$ has the same distribution as $F_Z(Z)$.

p-values have a uniform(0,1) distribution when $\beta_j = 0$



How p-values are used in practice

- Hypothesis test: a rule for deciding when $\beta_1 \neq 0$
- Type I error: the probability of saying that $\beta_1 \neq 0$ when in reality $\beta_1 = 0$ (false positive)
- Type II error: the probability of failing to say that $\beta_1 \neq 0$ when in reality $\beta_1 \neq 0$ (false negative)
- Most common hypothesis test: Say that $\beta_1 \neq 0$ when the p-value is < 0.05. This has Type I error of 5%
- If the p-value is > 0.05, this *isn't proof that* $\beta_1 = 0$. It could just be that we don't have much data. Not having much data makes the Type II error big.

Outline

Understanding confidence intervals and p-values

- Formula for our slope estimator, $\hat{\beta}_1$
- Distribution of our slope estimator, $\hat{\beta}_1$
- Confidence Intervals
- P-values

Visual inspection of residuals

- Method 1: Plot residuals vs. covariate
- Method 2: Autocorrelation plots
- Method 3: Normal probability plots

Goals for this segment

Confidence intervals and p-values are only valid if the assumption we made in deriving them are true.

The assumptions are:

- $Y_i = \beta_0 + \beta_1 X_{i,1} + \cdots + \beta_p X_{i,p} + \epsilon_i$
- ϵ_i are independent normal random variables with mean 0 and constant variance

Examples are in `test-assumptions-2022.ipynb`

Real-world example when assumptions are violated

- i indexes people using a mobile app over a 24 hour period (e.g., Instagram)
- $X_{i,1}$ is whether the treatment was on for person i (e.g., a new algorithm for pre-fetching videos when network bandwidth is low)
- Y_i is the user's engagement with the app
- Our linear model is $Y_i = \beta_0 + \beta_1 X_{i,1} + \epsilon_i$
- β_1 is how much the treatment improves enagement
- Linear regression assumes the ϵ_i are independent
- In reality, ϵ_i may be positively correlated, e.g., when network bandwidth is lower than normal for one person in Bangalore at 7pm, it is also probably also lower than normal for another person nearby
- If we don't adjust for this, our $\hat{\beta}_j$ will be less accurate and we may be overconfident: p-values too small, confidence intervals too narrow

We can check assumptions by plotting residuals

- A really good technique for checking these assumptions is to plot the residuals $\hat{\epsilon}_i = Y_i - \hat{Y}_i$
- The residuals should be independent normal random variables with mean 0 and constant variance

We check assumptions by examining these properties of the residuals

Goal: check that the residuals are independent normal random variables with mean 0 and constant variance

- Method 1: Plot the residuals vs. each covariate. Visually inspect the mean and variance. Make sure they aren't changing.
- Method 2: Autocorrelation plots
- Method 3: Q-Q plots

Method 1

Method 1: Plot the residuals vs. each covariate. Visually inspect the mean and variance. Make sure they aren't changing.

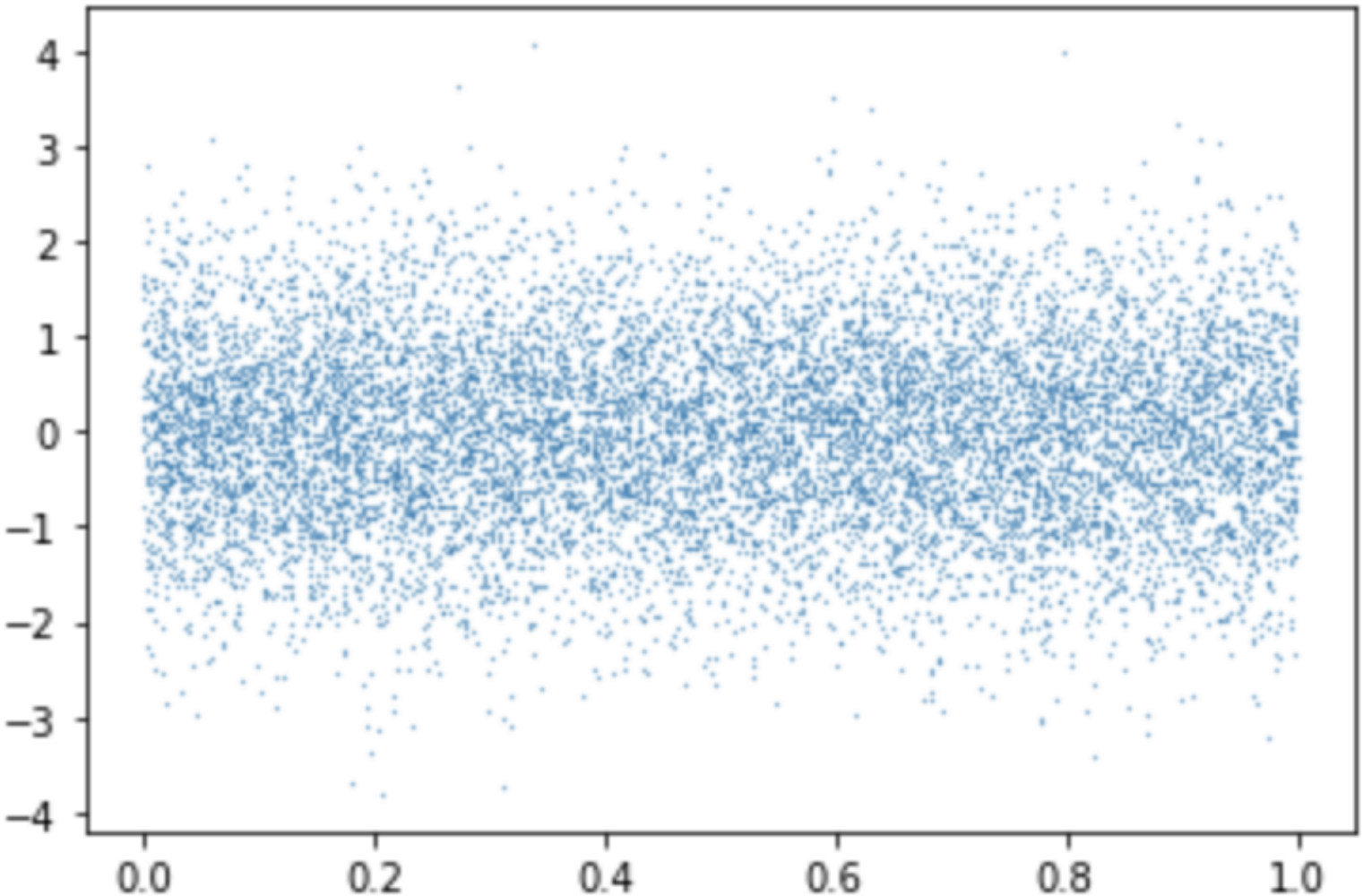
Catches these issues:

- $E[Y_i|X_{i,1}, \ldots, X_{i,p}] \neq \beta_0 + \beta_1 X_{i,1} + \cdots + \beta_p X_{i,p}$, i.e., the model is not linear in the predictors
- The variance of ϵ_i depends on $X_{i,j}$ for some j, i.e., the errors do not have constant variance.

Example of method 1

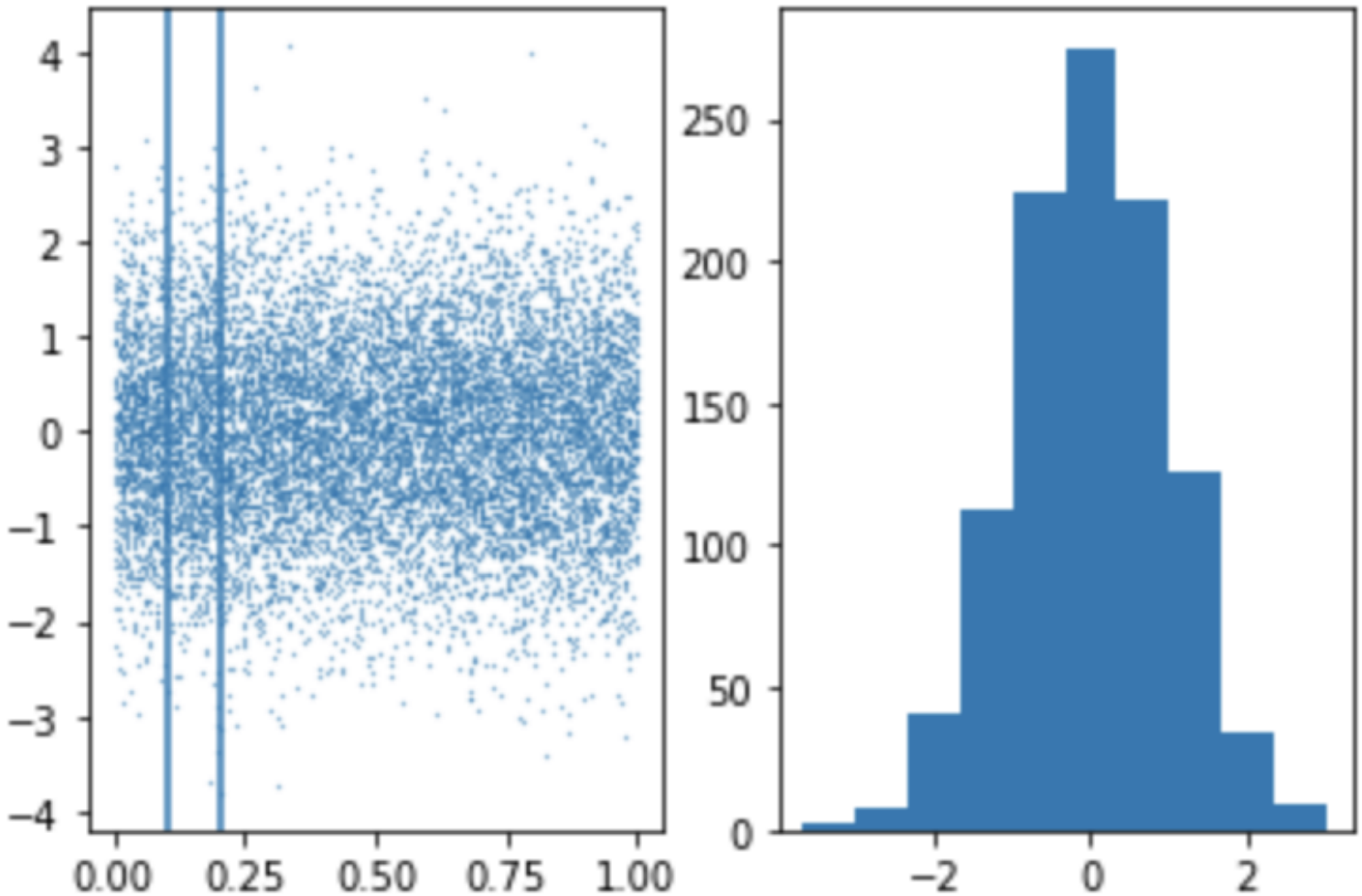
```
n = 10000
x = rand(n) # uniform
eps = randn(n) # eps are independent normal(0,1)
y = 1 + 2*x + eps
model = sm.OLS(y,sm.add_constant(x)).fit()
plt.plot(x,model.resid,'.',markersize=0.5)
```

Example of method 1



What to look for in method 1

In each window on the x-axis, we want the mean and variance of the y-values to be the same

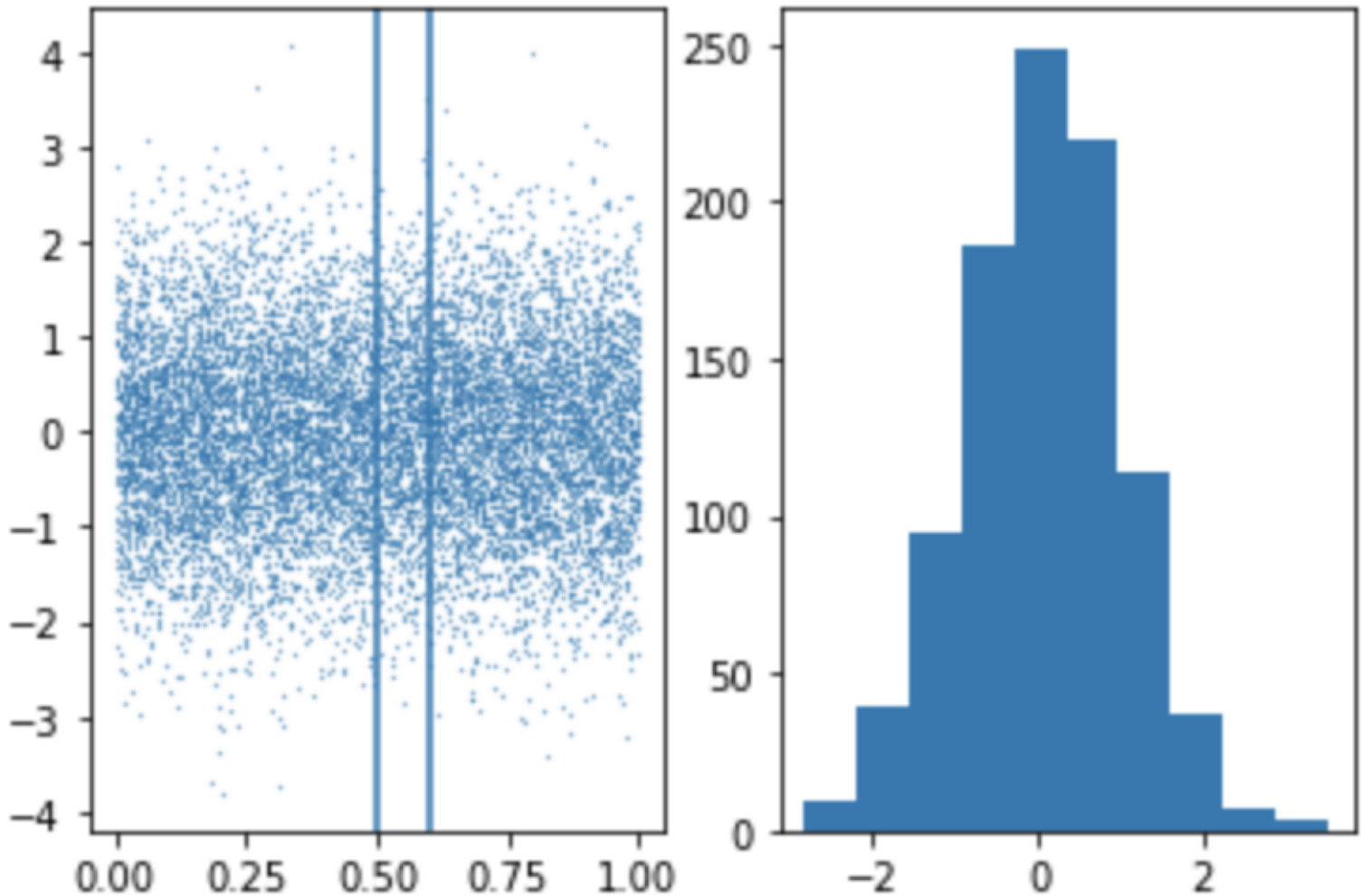


Code for the previous slide

```
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)])
```

What to look for in method 1

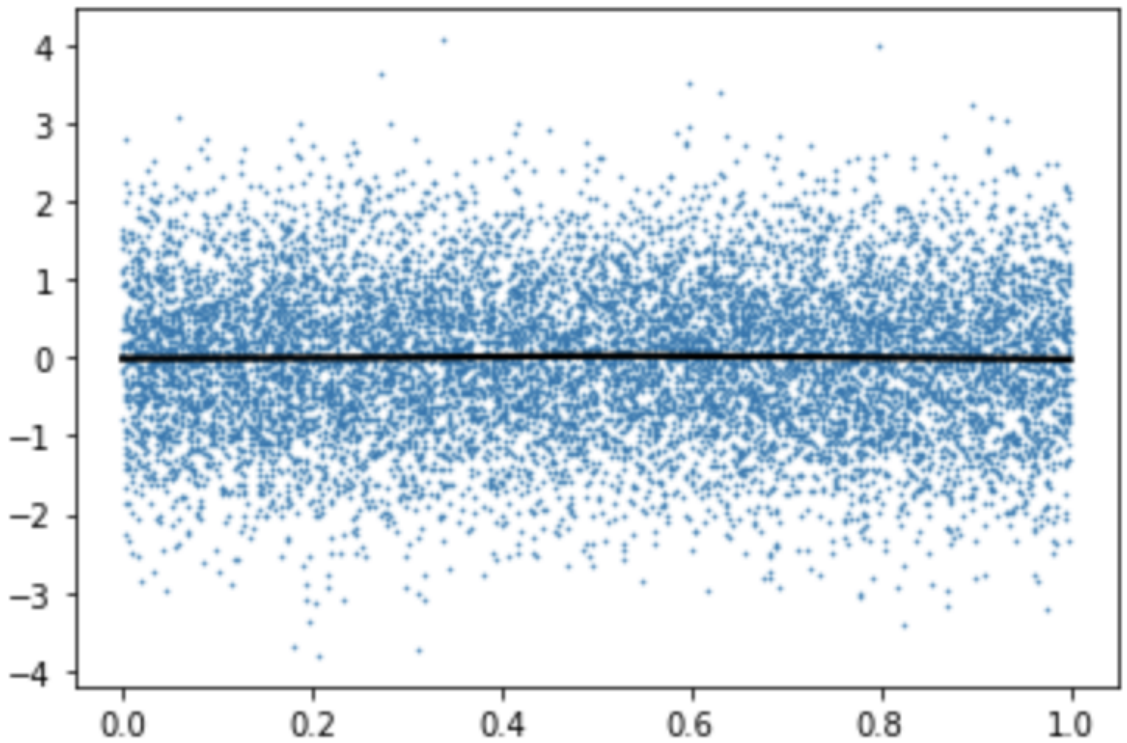
Here's the same plot, but moving the window. We want the mean and variance to be the same regardless of where the window is.



LOWESS can help us look for changing mean

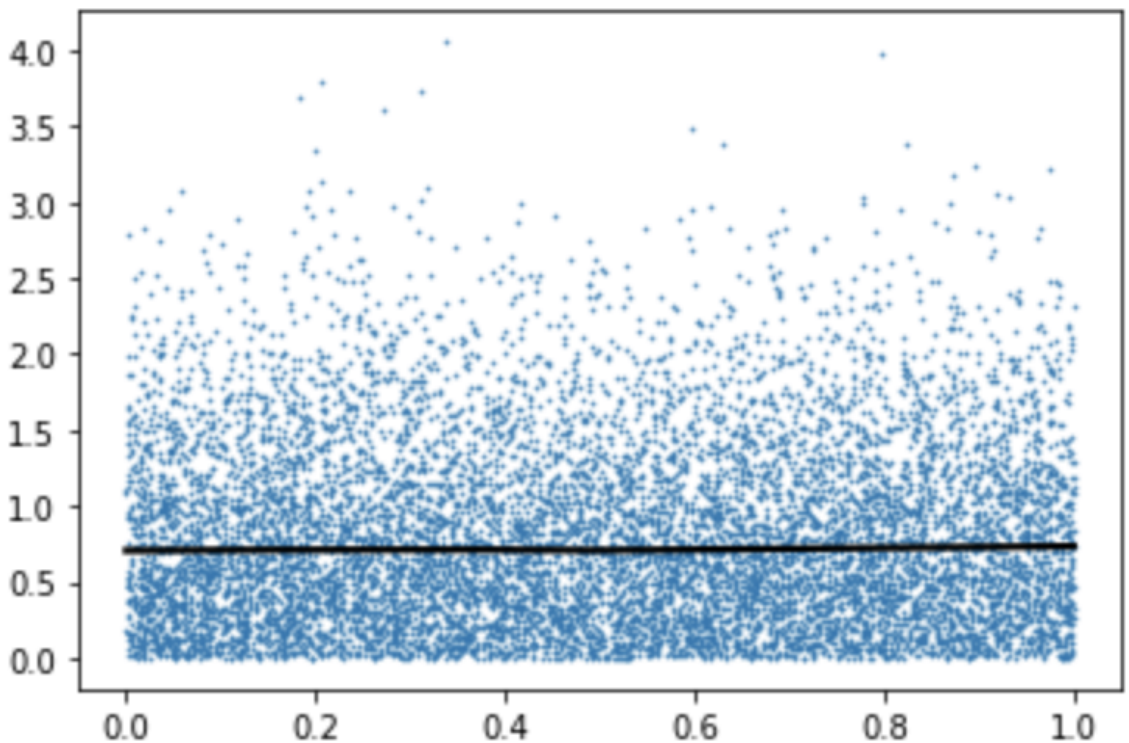
We can use a LOWESS (locally weighted scatterplot smoothing) plot to help us see if the mean is changing.

```
import seaborn as sns
sns.regplot(x, model.resid, lowess=True, \
    line_kws = {'color': 'black'}, \
    scatter_kws = {'s' : 0.5})
```



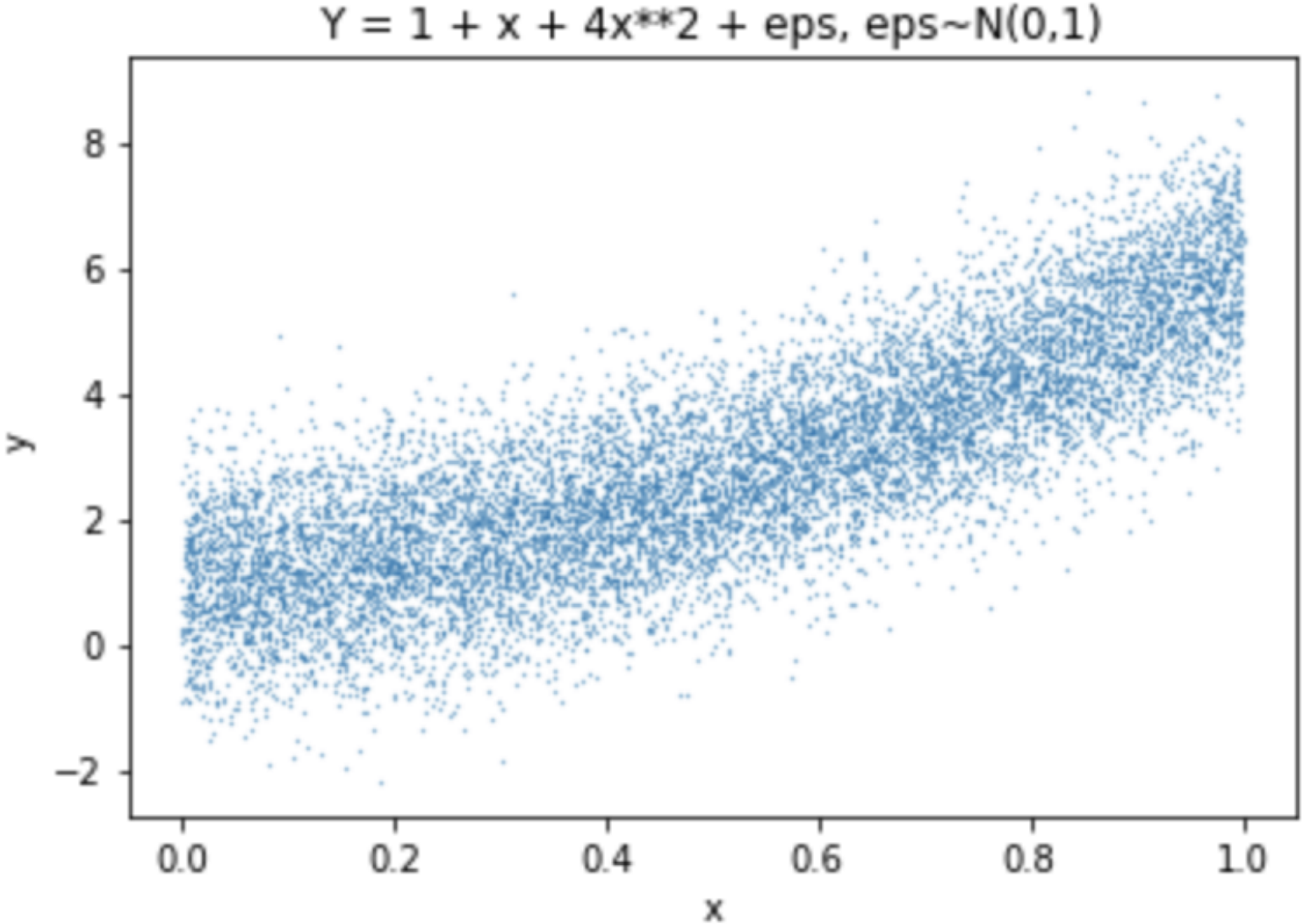
Plotting the absolute value of the residuals can make it easier to see changes in variance

```
sns.regplot(x, abs(model.resid), lowess=True, \
    line_kws = {'color': 'black'}, \
    scatter_kws = {'s' : 0.5})
```



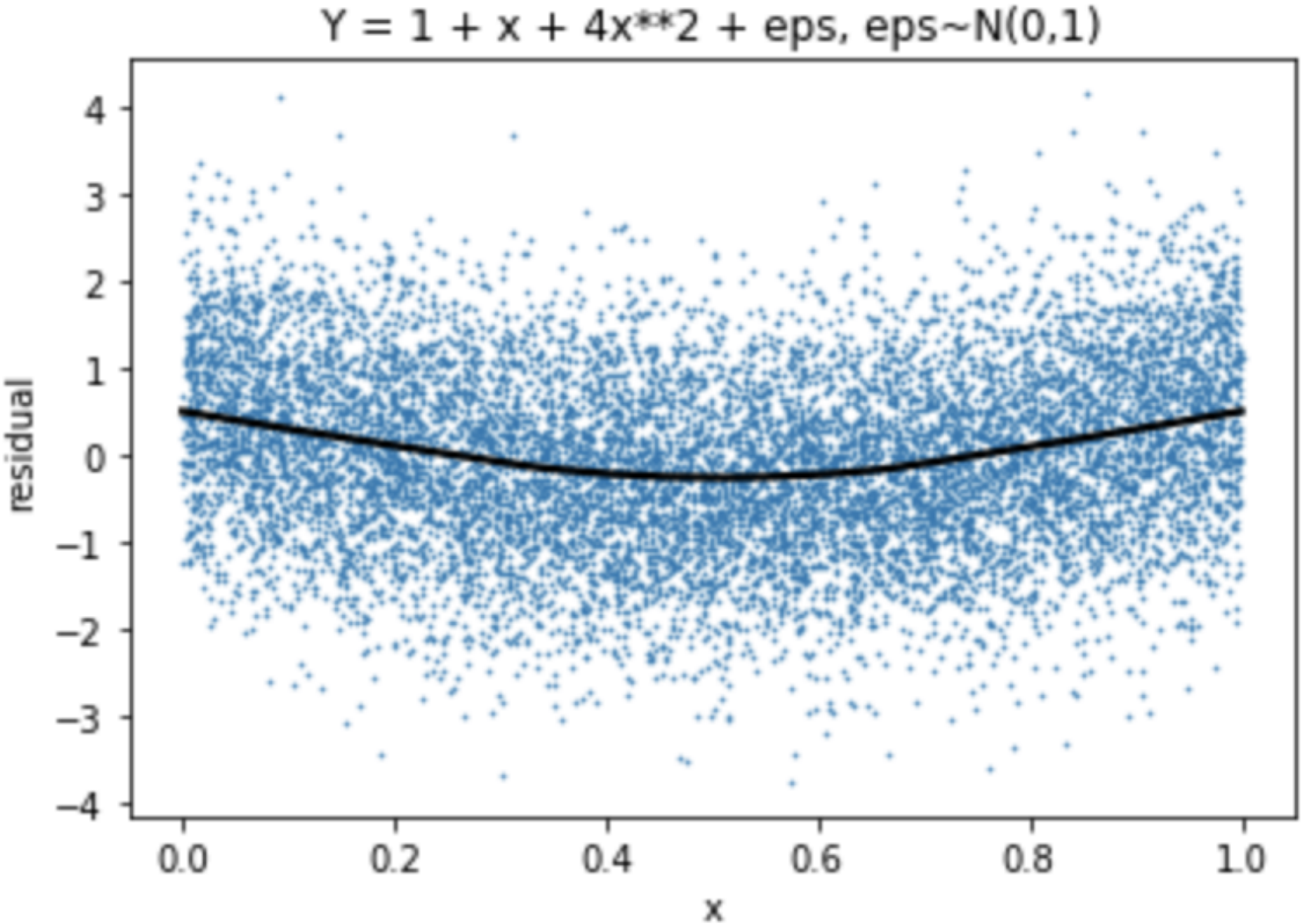
Example where method 1 can catch a problem

Here we fit $Y = \beta_0 + \beta_1 X$ even though Y is actually quadratic in X.



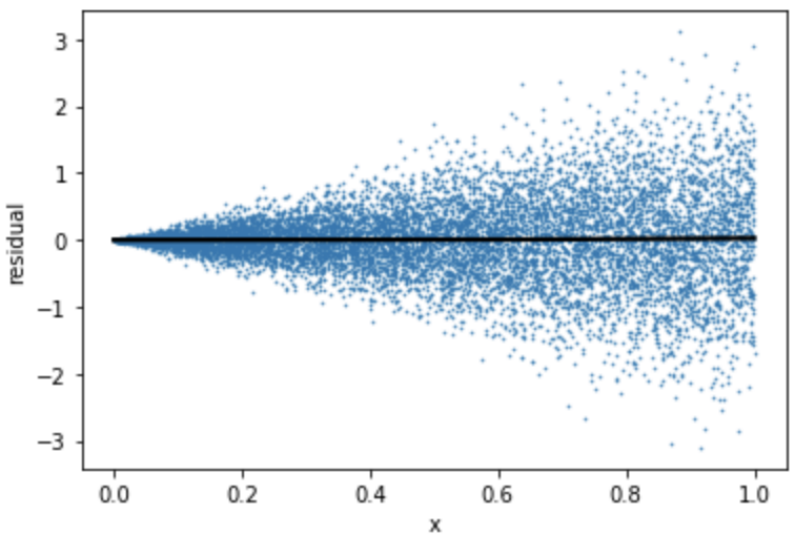
Example where method 1 can catch a problem

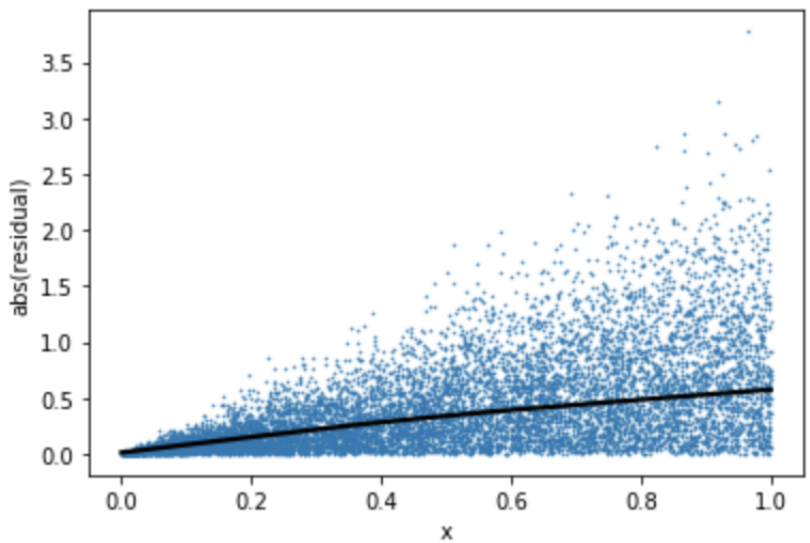
Here we fit $Y = \beta_0 + \beta_1 X$ even though Y is actually quadratic in X.



Example with non-constant variance

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





Examples with fewer datapoints

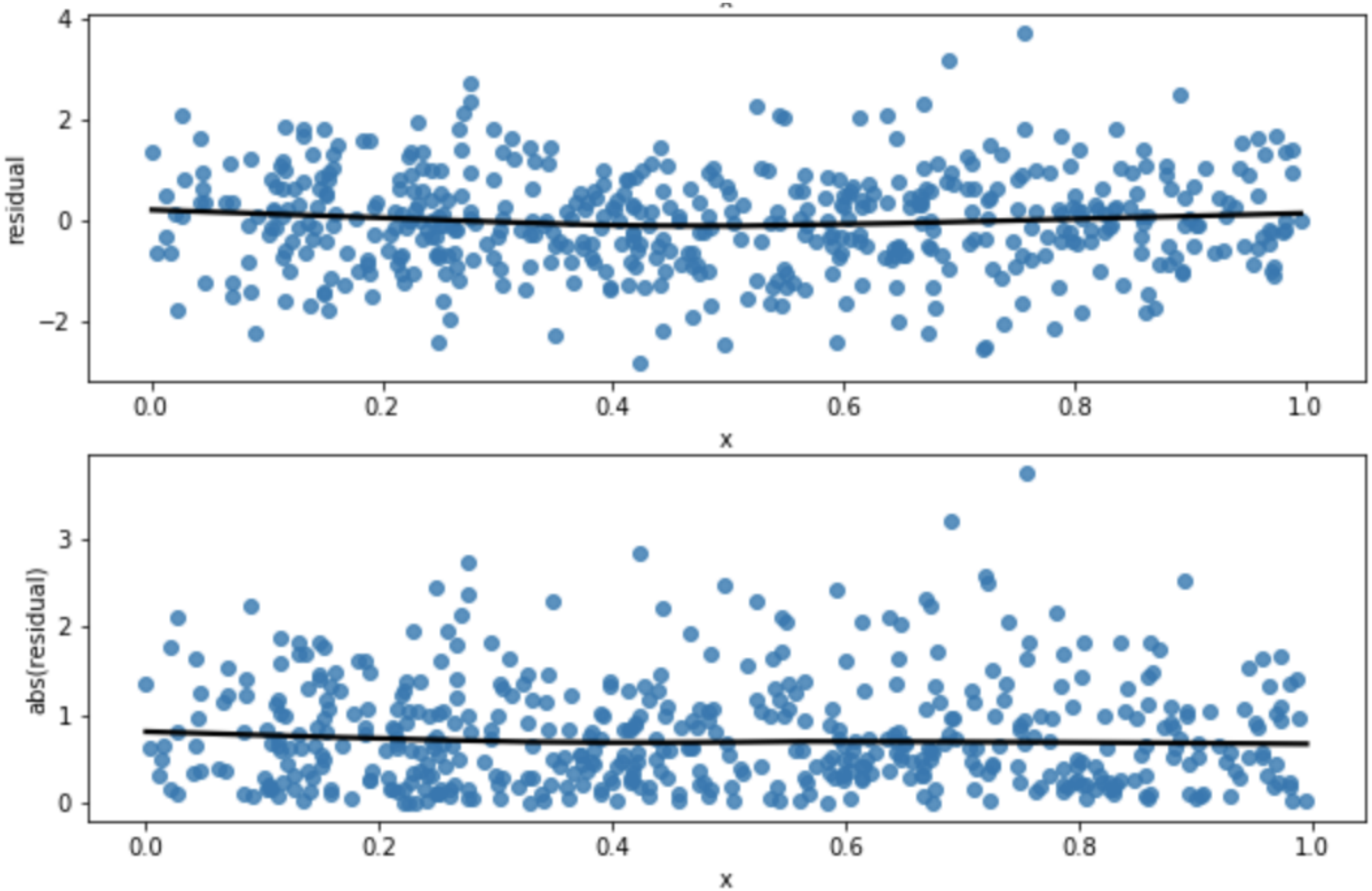
In each of the following examples, the residuals satisfy our assumptions.

They were generated with this code, with different values of `n`.

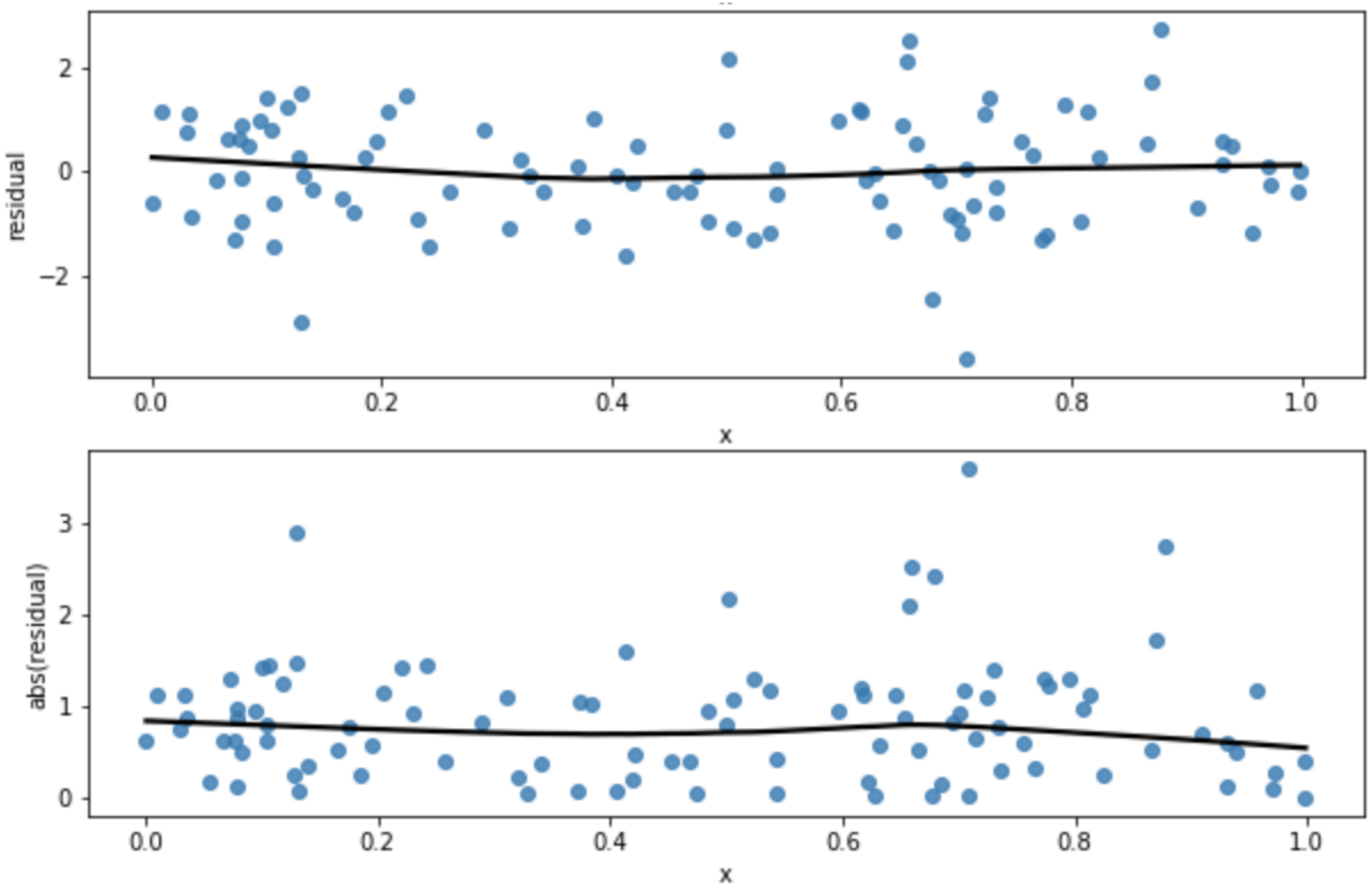
```
x = rand(n) # uniform
eps = randn(n) # normal, mean 0 variance x^2, independent
y = 1 + x + eps
model = sm.OLS(y,sm.add_constant(x)).fit()
```

When there are fewer datapoints, residual plots are a little noiser and we should tolerate more variation in the LOWESS lines.

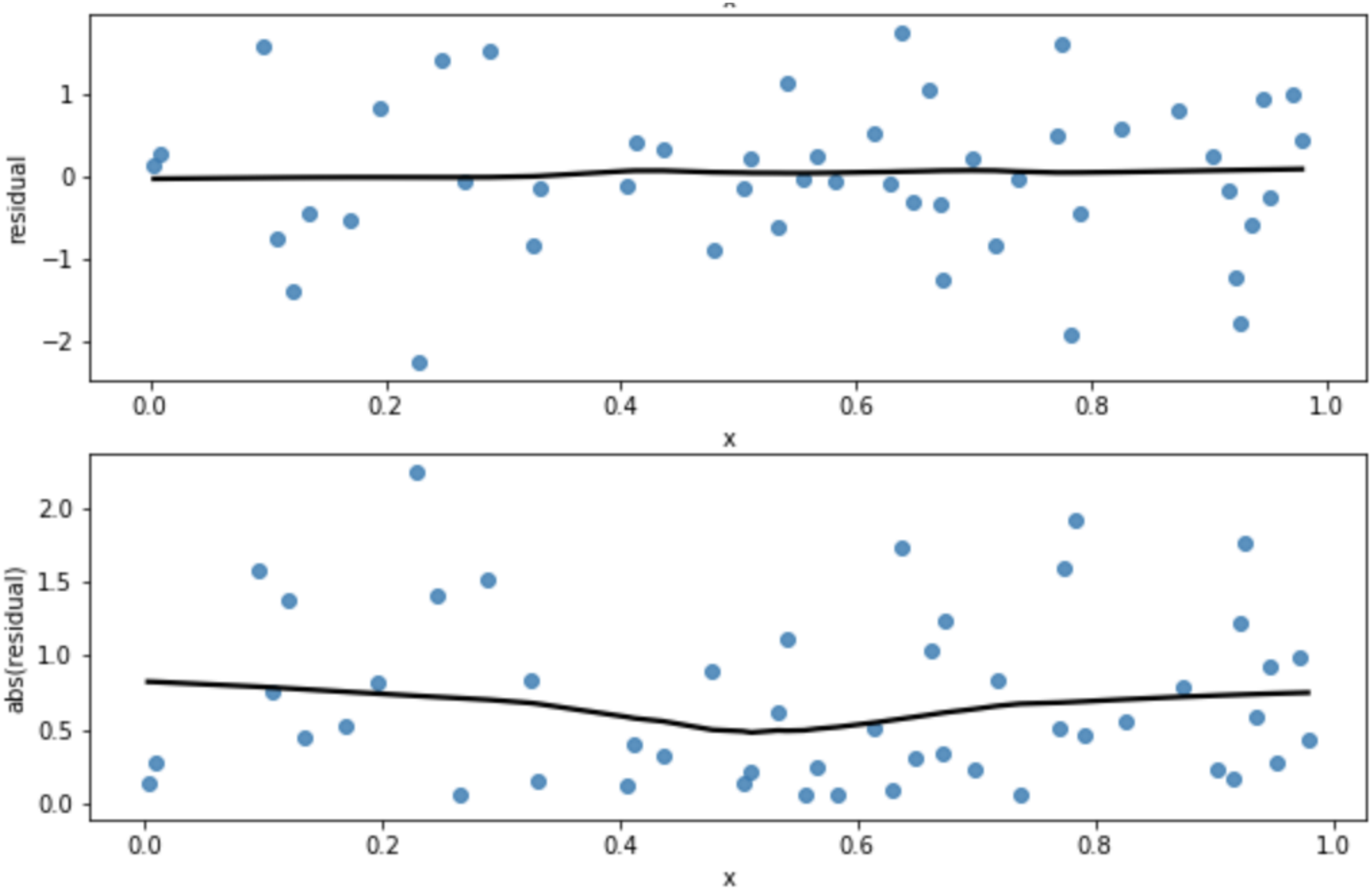
Example with 500 datapoints



Example with 100 datapoints



Example with 50 datapoints



Example where x is not uniform

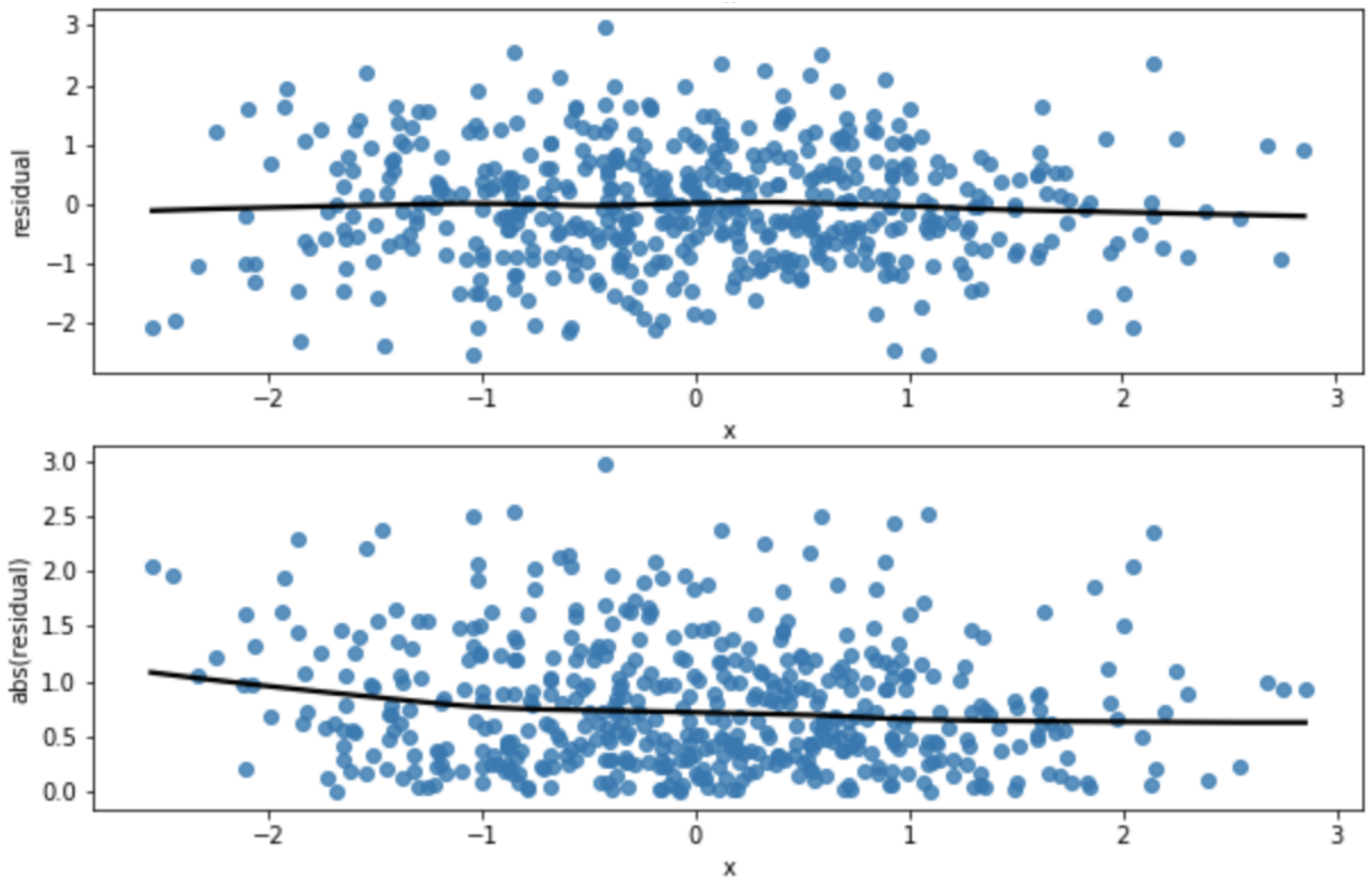
When x is not uniform, residual plots look a little different.

In regions where the x are sparse, residual plots are a little noiser and we should tolerate more variation in the LOWESS lines.

The plots on the next slide were generated with this code.

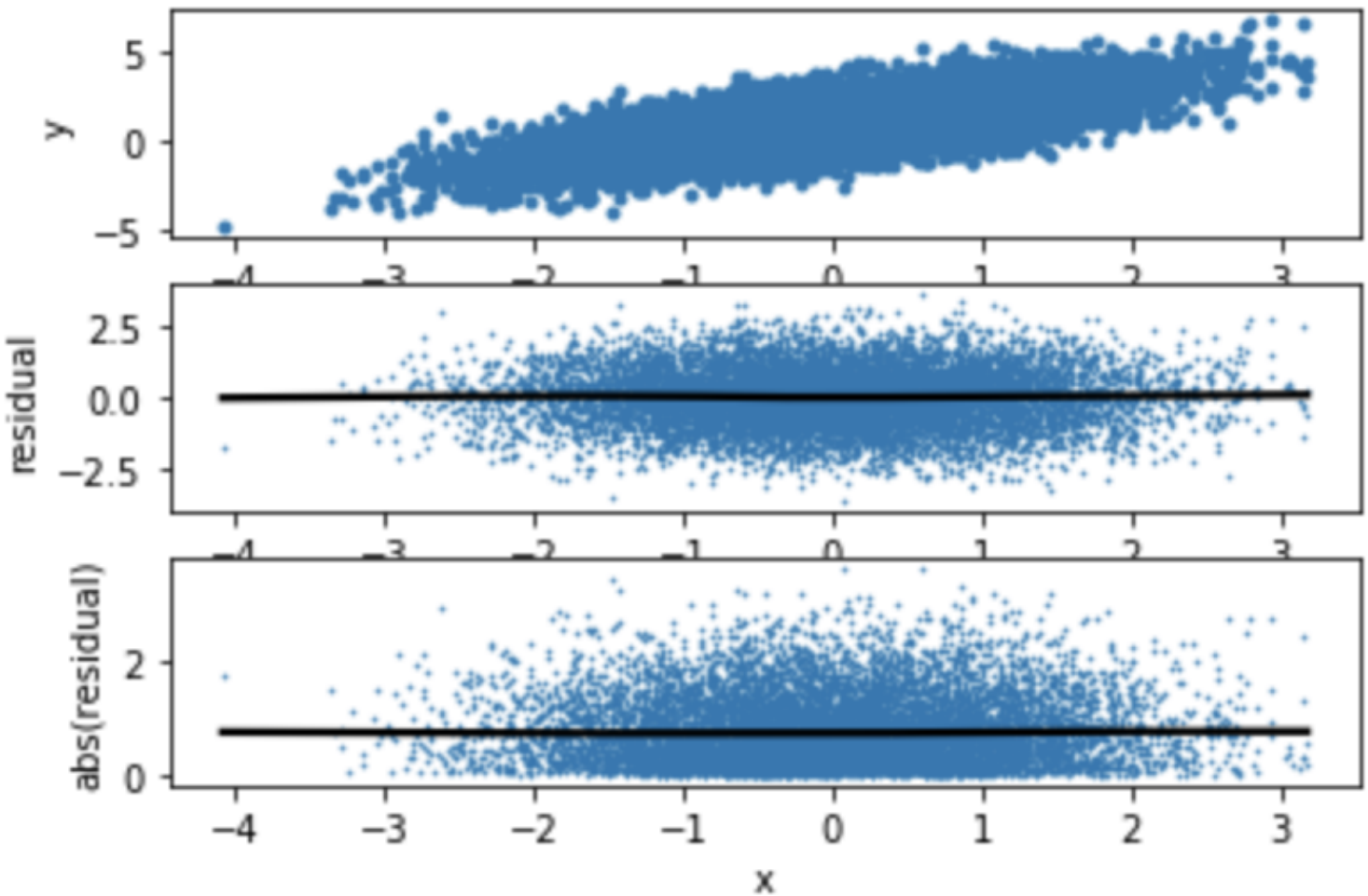
```
n = 500
x = randn(n) # normal
eps = randn(n) # normal, mean 0 variance x^2, independent
y = 1 + x + eps
model = sm.OLS(y,sm.add_constant(x)).fit()
```

Example where x is not uniform



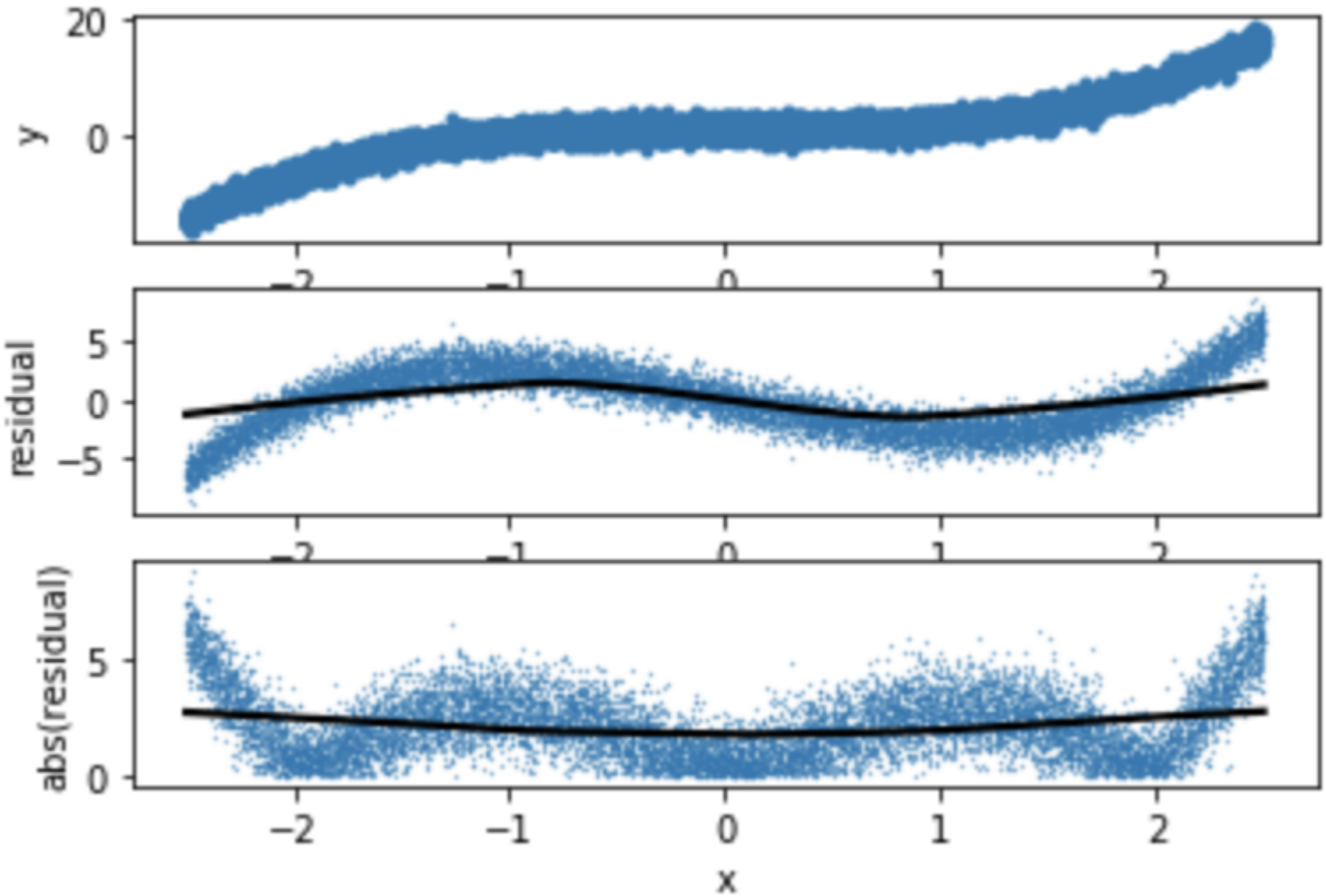
Question 5: are there assumptions violated?

- A. No assumption violated
- B. Assumption(s) violated



Question 6: are there assumptions violated?

- A. No assumption violated
- B. Assumption(s) violated



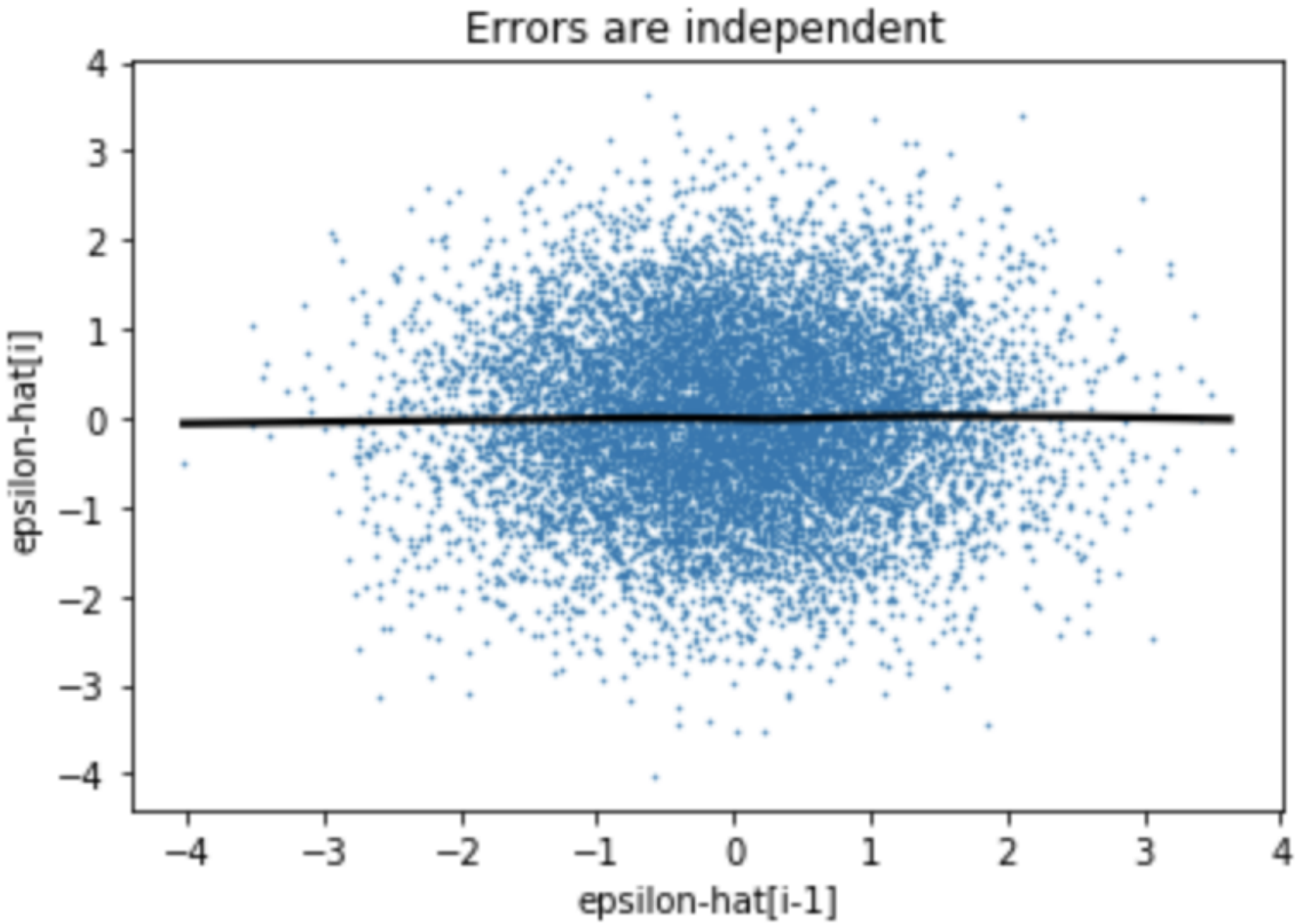
Method 2: Autocorrelation plot

- Our goal now is to check the assumption that $\epsilon_1, \ldots, \epsilon_n$ are mutually independent
- One way to check this is via plotting $\widehat{\epsilon}_i$ versus $\widehat{\epsilon}_{i-1}$ (and also versus $\widehat{\epsilon}_{i-2}$, $\widehat{\epsilon}_{i-3}$, etc.)
  - should see that $\widehat{\epsilon}_i$ wouldn't help us predict $\widehat{\epsilon}_{i-1}$ (or $\widehat{\epsilon}_{i-2}$, $\widehat{\epsilon}_{i-3}$, etc.) and vice versa.
- A faster way to get the same information is via an autocorrelation plot

Example: data with mutual independence, plotting $\widehat{\epsilon}_i$ vs. $\widehat{\epsilon}_{i-1}$

```
# generate data where 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})
```

Example: data with mutual independence, plotting $\widehat{\epsilon}_i$ vs. $\widehat{\epsilon}_{i-1}$

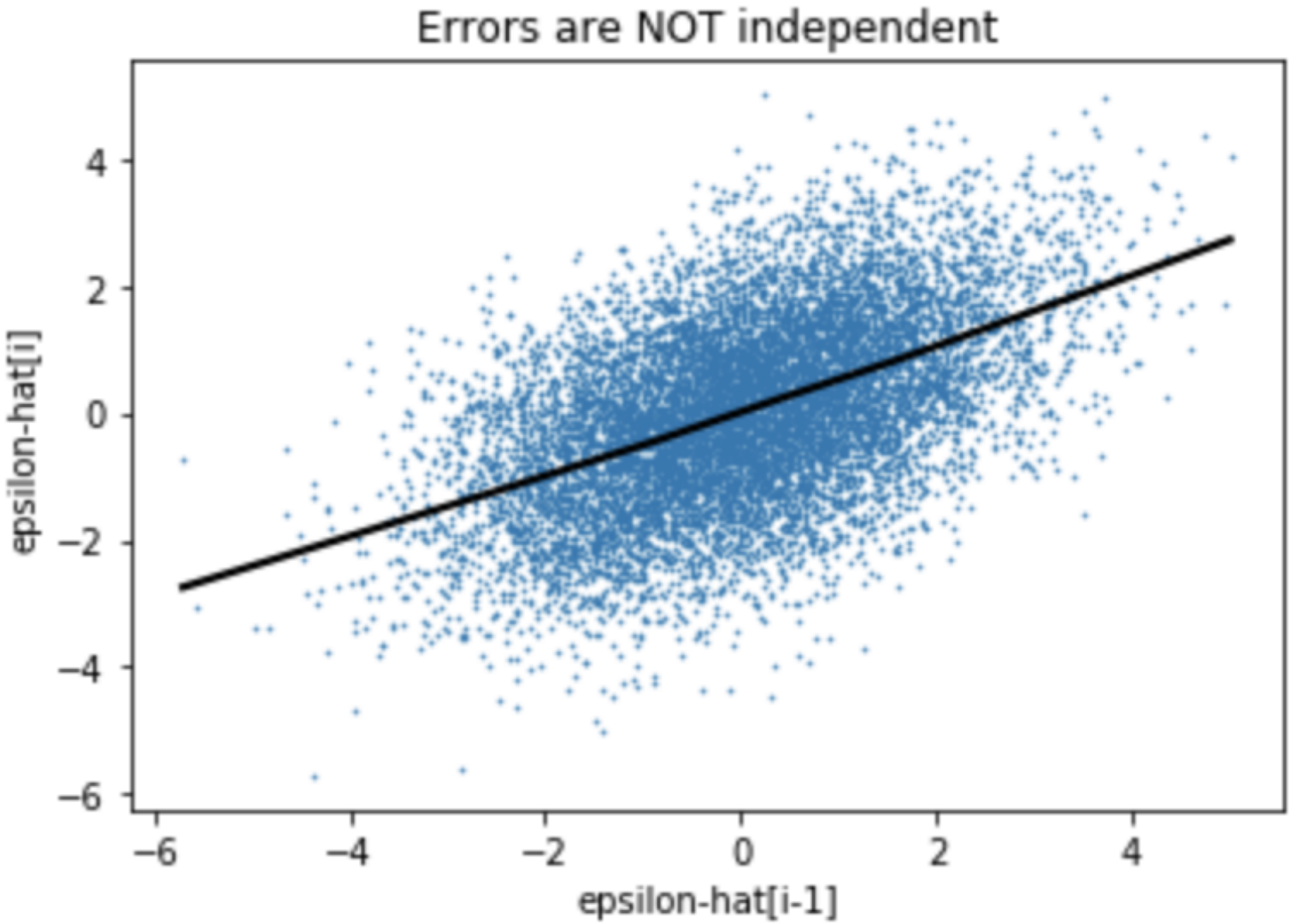


The LOWESS line is straight. This means that the mean of $\widehat{\epsilon}_{i-1}$ doesn't depend on ϵ_i.

Example: data without mutual independence, $\widehat{\epsilon}_i$ vs. $\widehat{\epsilon}_{i-1}$

```
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})
```

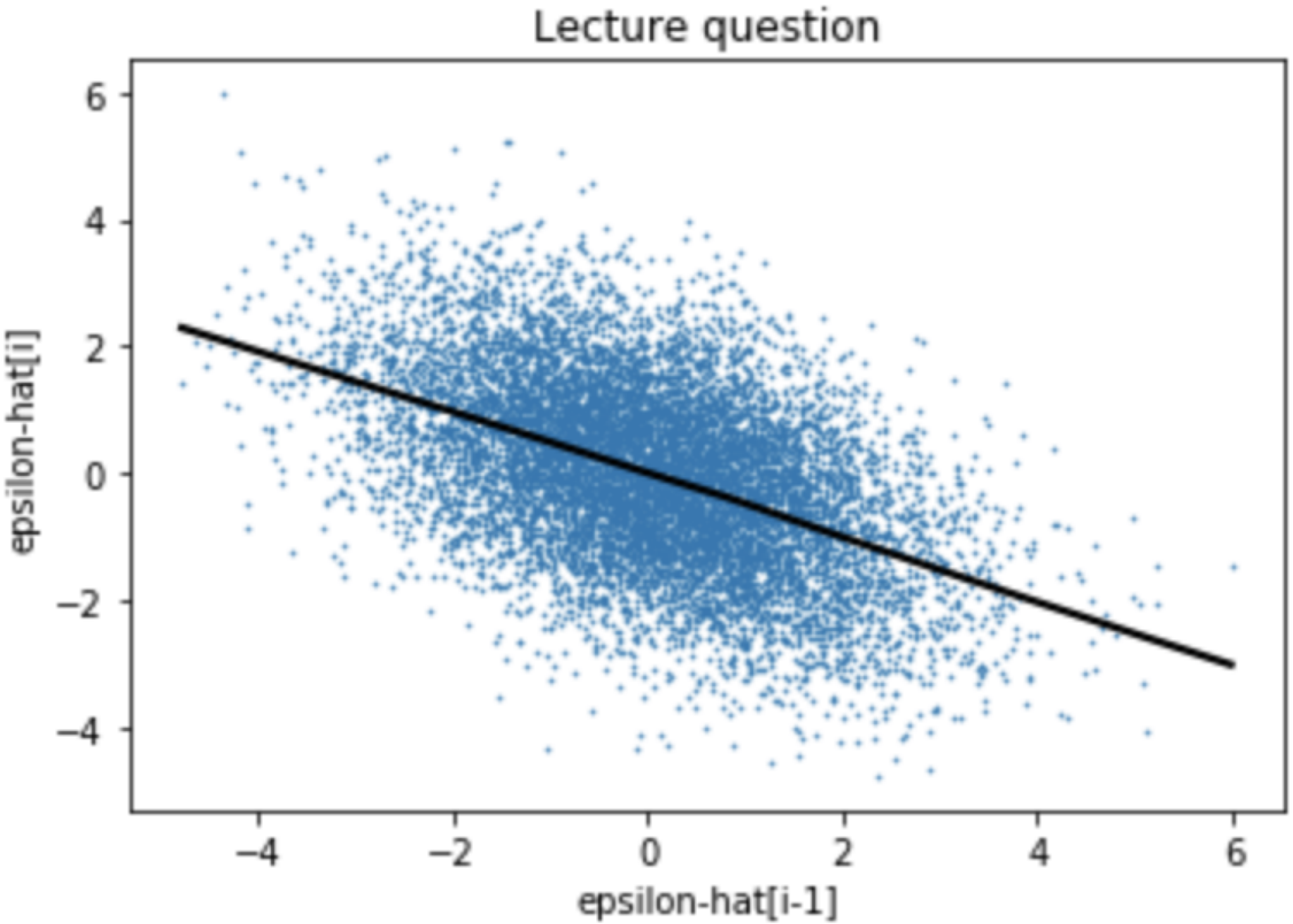
Example: data without mutual independence, $\widehat{\epsilon}_i$ vs. $\widehat{\epsilon}_{i-1}$



The LOWESS line is not straight. This means that knowing ϵ_i would help us predict $\widehat{\epsilon}_{i-1}$ and the errors aren't independent.

Lecture Question 7: are the errors mutually independent?

A = yes, B= no

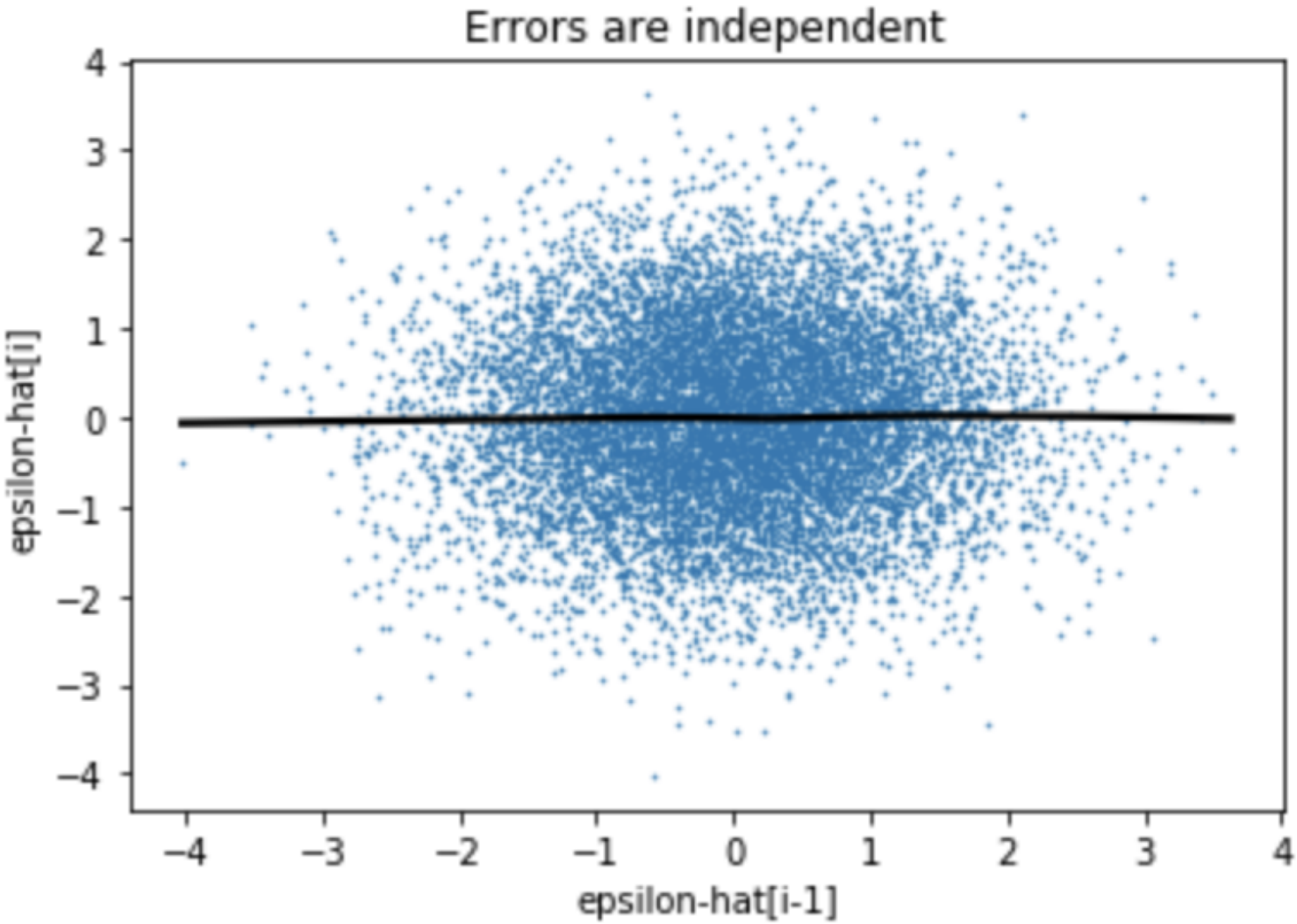


Correlation can tell us whether $\widehat{\epsilon}_i$ and $\widehat{\epsilon}_{i-1}$ are independent

- The correlation of two vectors x and y is $\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}))/\sigma_x\sigma_y$ where $\bar{x}$ is the average of x, σ_x is the standard devation of x, and similarly for $\bar{y}$ and σy
- The correlation is proportional to the slope of the line that one would get by regressing y on x.
  - You can see this by noting that the expression for $\hat{\beta}_1$ on slide 5 implies $\hat{\beta}_1 = [\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x})]/\sum_{i=1}^{n}(x_i - \bar{x})^2$, which is the correlation times σ_y/σ_x.
- Independence implies that correlation is 0. Except in very special circumstances, lack of independence implies that the correlation is non-zero.

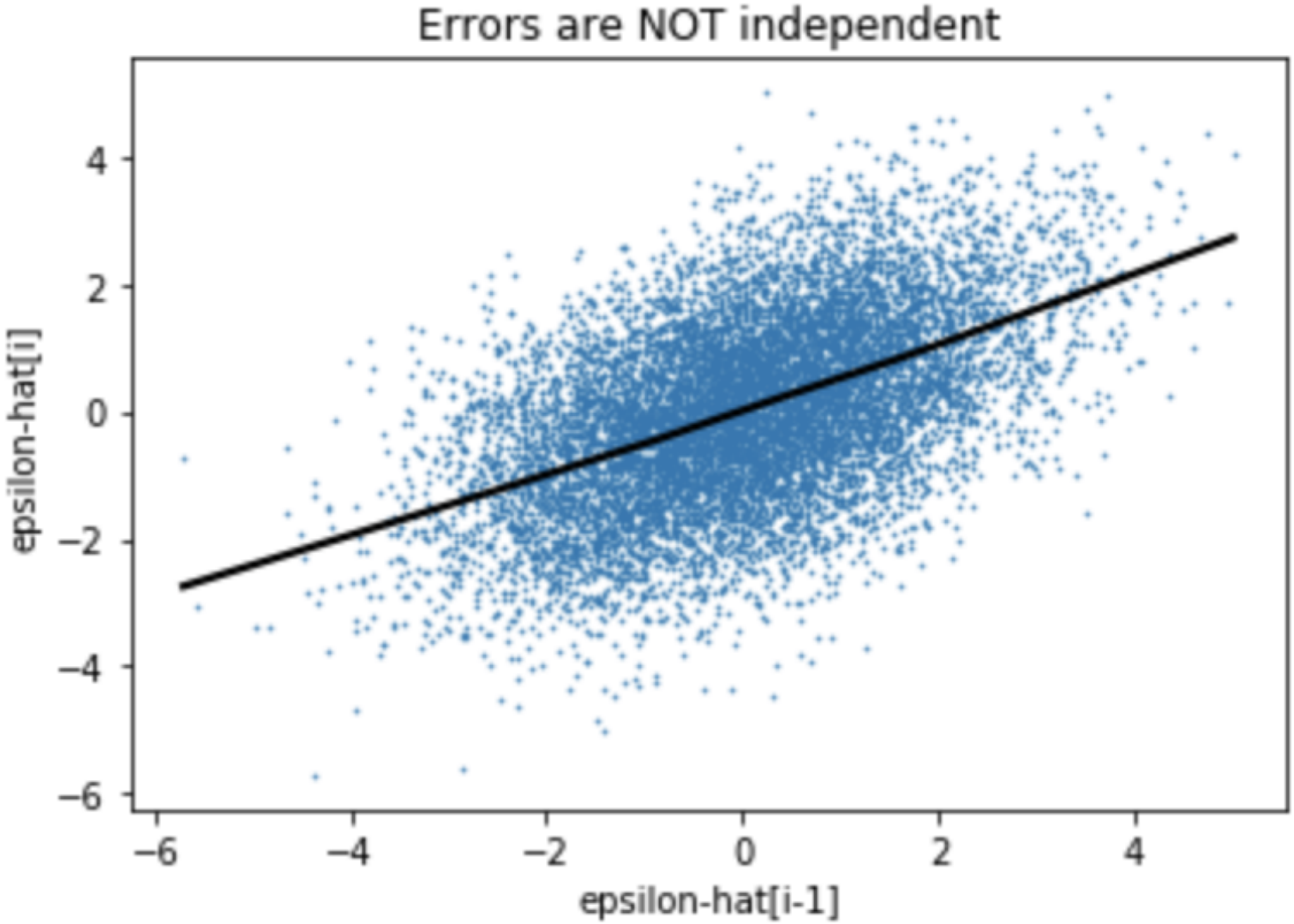
Lecture Question 8: What is the correlation of $\widehat{\epsilon}_i$ with $\widehat{\epsilon}_{i-1}$?

A = positive, B = negative, C = zero



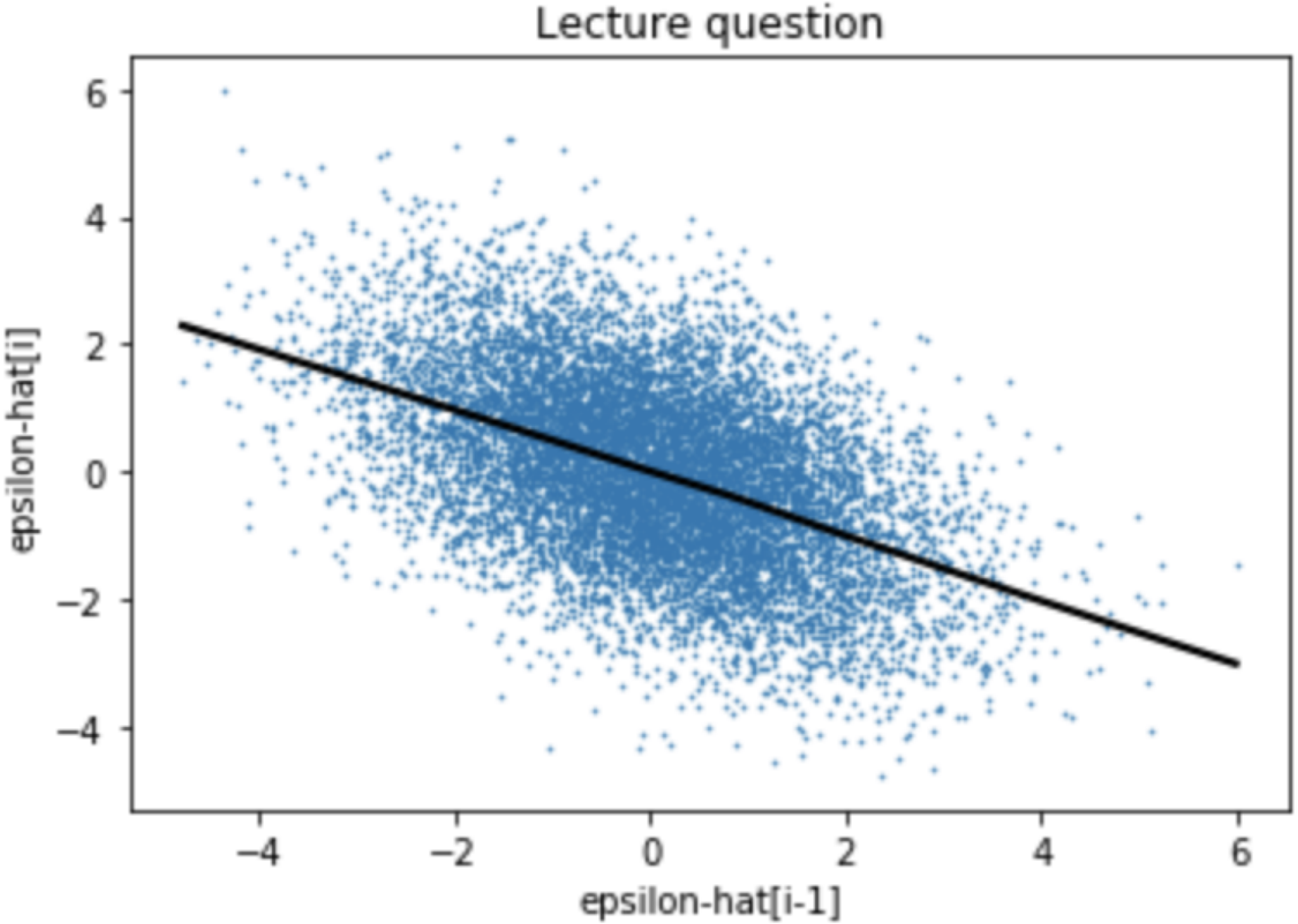
Lecture Question 9: What is the correlation of $\widehat{\epsilon}_i$ with $\widehat{\epsilon}_{i-1}$?

A = positive, B = negative, C = zero



Lecture Question 10: What is the correlation of $\widehat{\epsilon}_i$ with $\widehat{\epsilon}_{i-1}$?

A = positive, B = negative, C = zero



We should also check the correlation at other lags

- When we plot $\widehat{\epsilon}_i$ vs. $\widehat{\epsilon}_{i-1}$, this is called plotting $\widehat{\epsilon}_i$ vs. itself at "lag 1"
- To check for mutual independence, it's a good idea to also check for independence of $\widehat{\epsilon}_i$ with itself at other lags: lag 2, lag 3, etc.
- E.g., we might see independence at lag 1 but lack of independence at lag 24 if the data is collected once per hour and the same hour of day tends to be similar
- This would require us to look at a lot of plots
- Fortunately, there is a better way: an autocorrelation plot

We can quickly check mutual independence with autocorrelation plots

- An *autocorrelation plot* plots the autocorrelation function:

$$r(t) = \text{Correlation}(\widehat{\epsilon}_i, \widehat{\epsilon}_{i-t})$$

- t is called the lag
- $r(t)$ should be near 0 for all $t > 0$ (except for random variation)
- it won't be exactly 0 because of noise
- python provides test bounds — we want no (or only a few) $r(t)$ for $t > 0$ outside the test bounds
- Those $r(t)$ with $t > 0$ that exceed the test bounds should only exceed it by a little bit

Autocorrelation plot, errors are independent

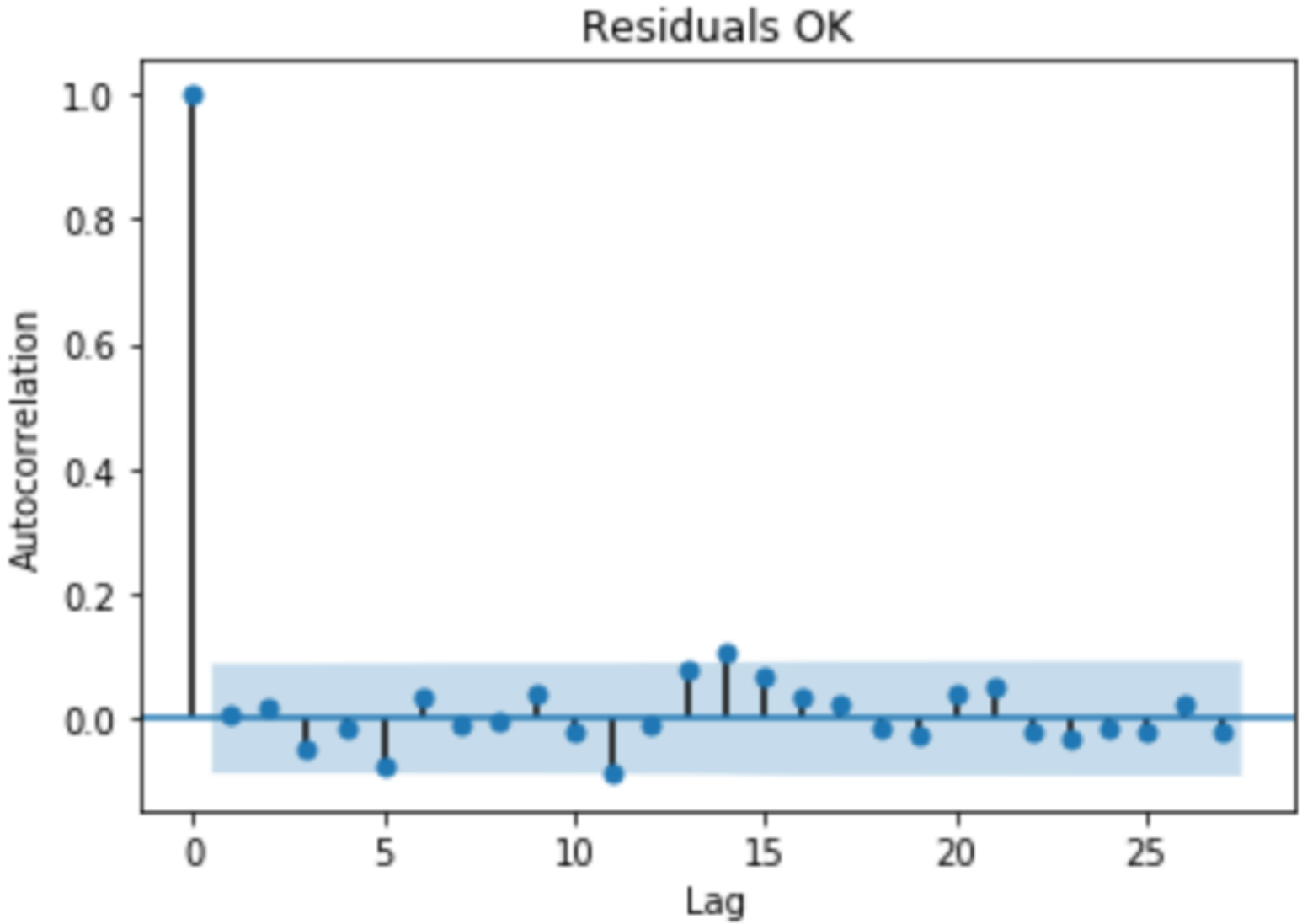
This code makes data with mutual independence.

```
n   = 500   # 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

plt.acorr(resid)
```

This code makes an autocorrelation plot:

```
import statsmodels.graphics.tsaplots as tsa
tsa.plot_acf(resid)
plt.show() # Prevents plot showing up twice
```

Autocorrelation plot, errors are independent

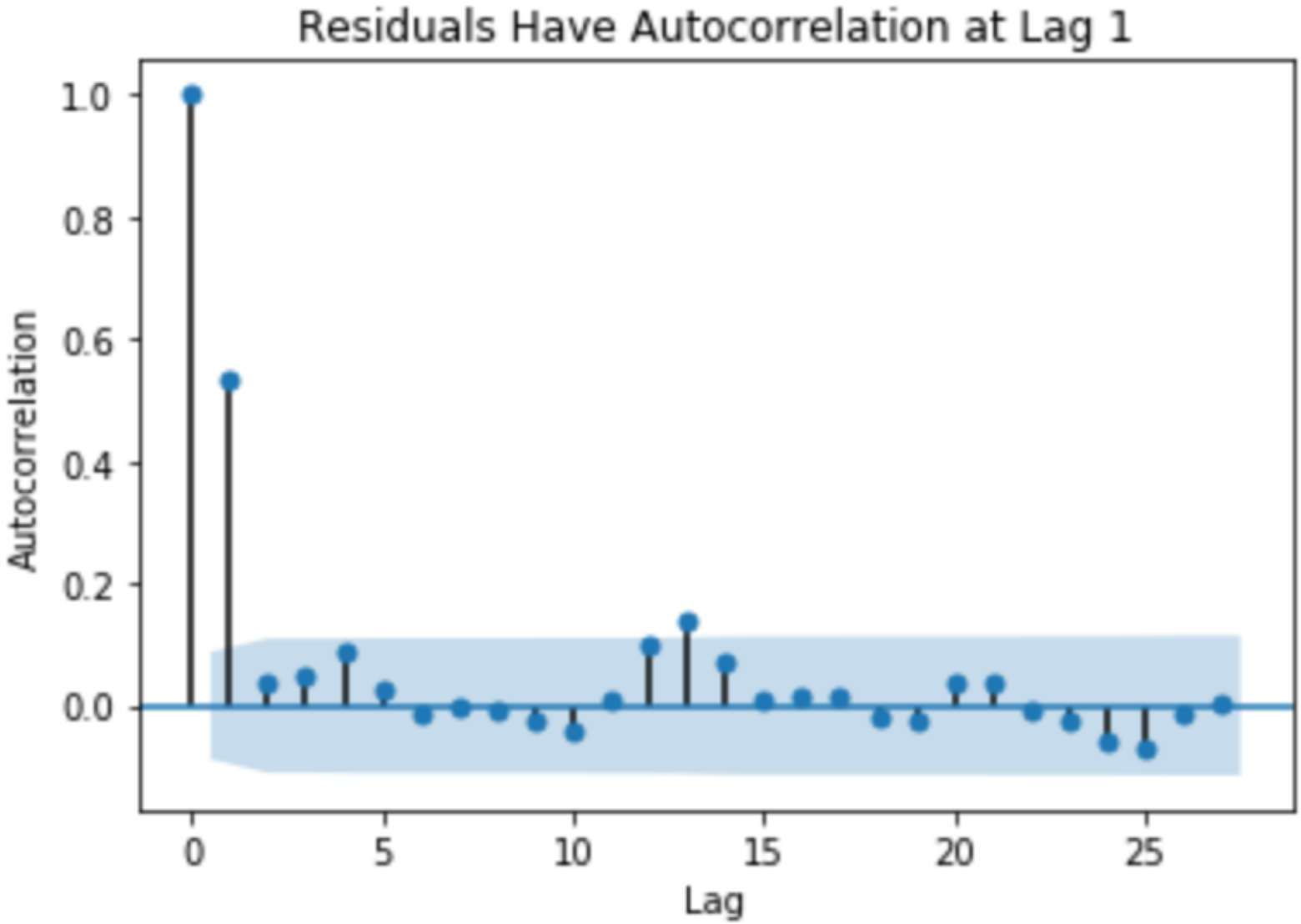


Autocorrelation plot, errors are NOT independent

This makes data **without** mutual independence.

```
n   = 500   # 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
```

Autocorrelation plot, errors are NOT independent



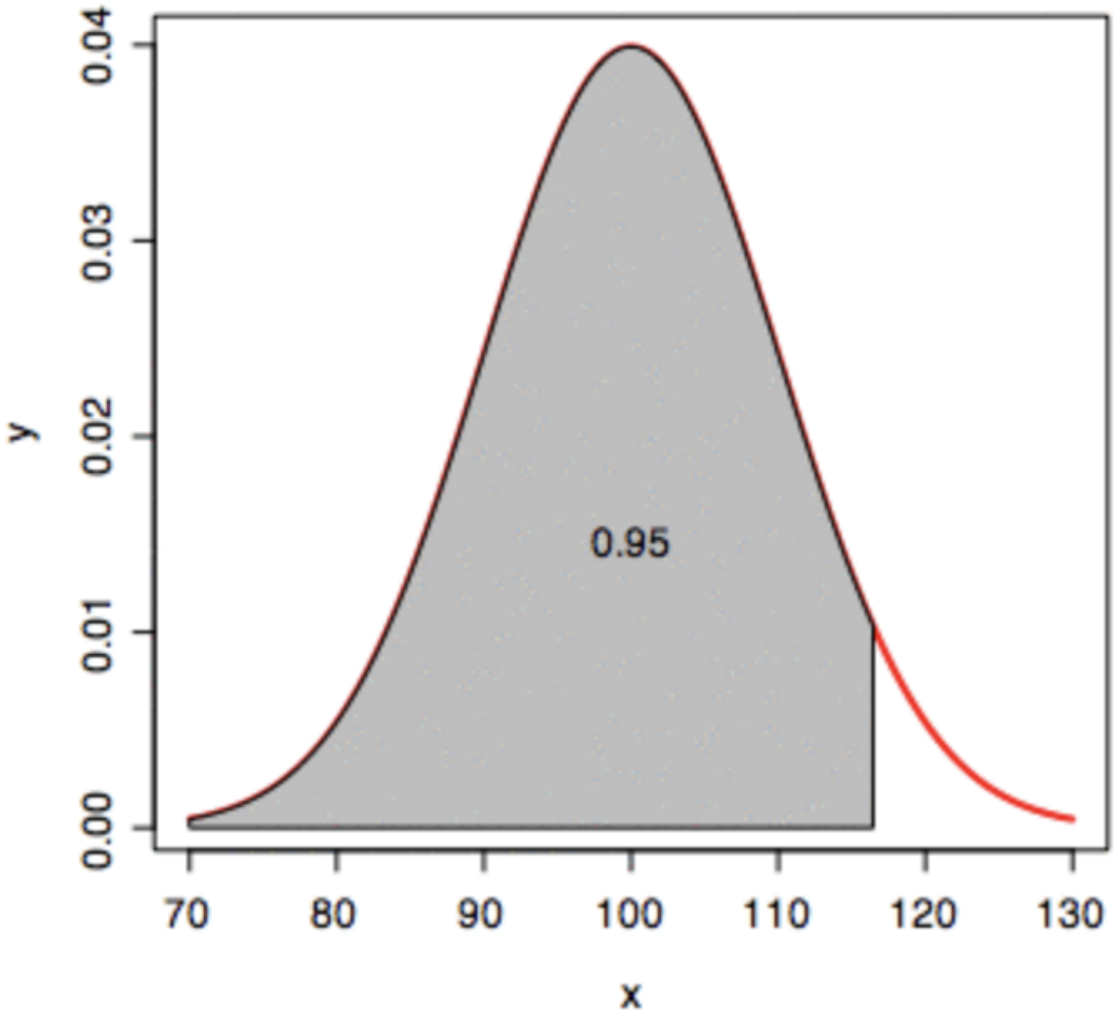
Method 3

Method 3. checks whether $\epsilon_1, \ldots, \epsilon_n$ are normally distributed using a normal probability plot

- plots sample quantiles vs. theoretical quantiles
- should see a straight line
- a curved line means the residuals aren't normal

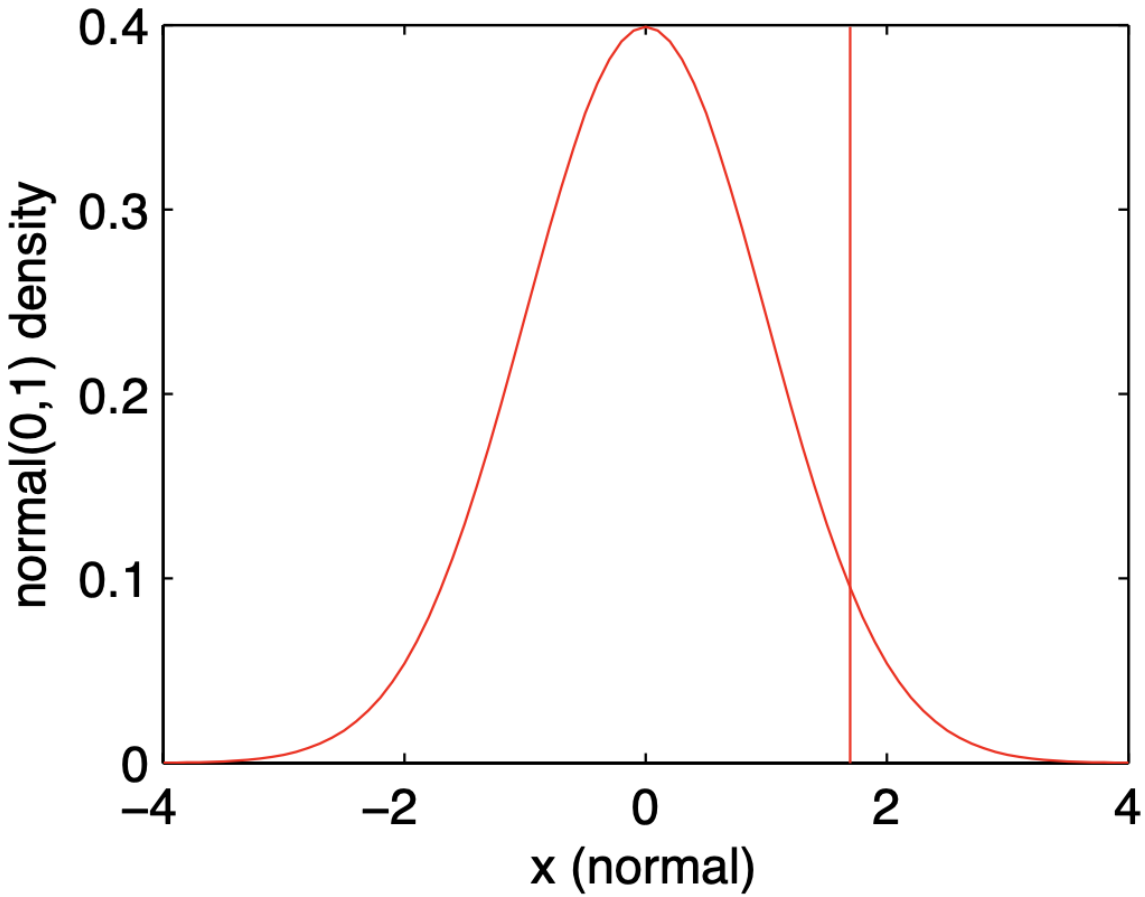
What's a quantile?

The α-quantile of the distribution of a random variable X is the number q that satisfies $P(X \leq q) = \alpha$



The standard normal distribution has quantiles

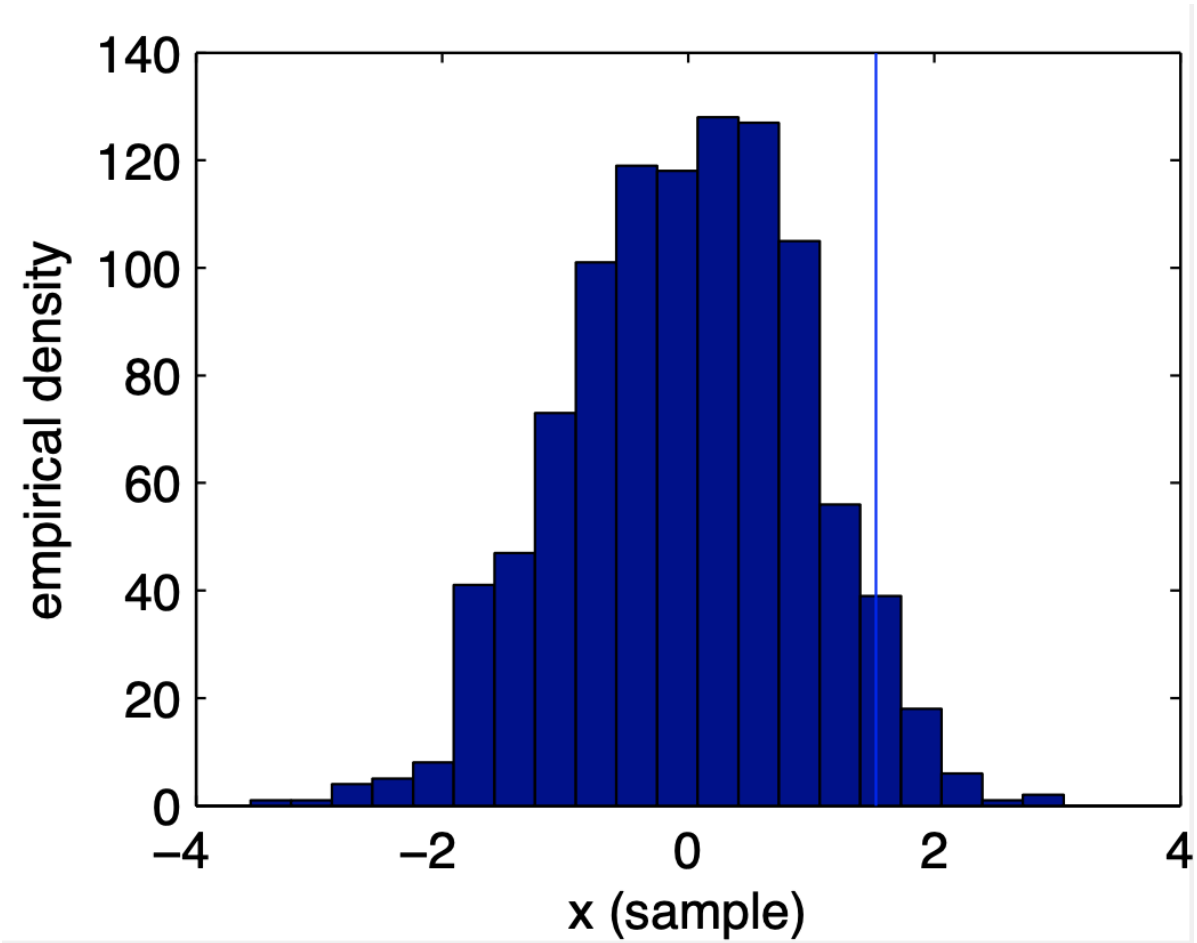
The vertical red line is at the 95% quantile of the standard normal distribution



We call this the *theoretical quantile*.

The histogram of the data has quantiles

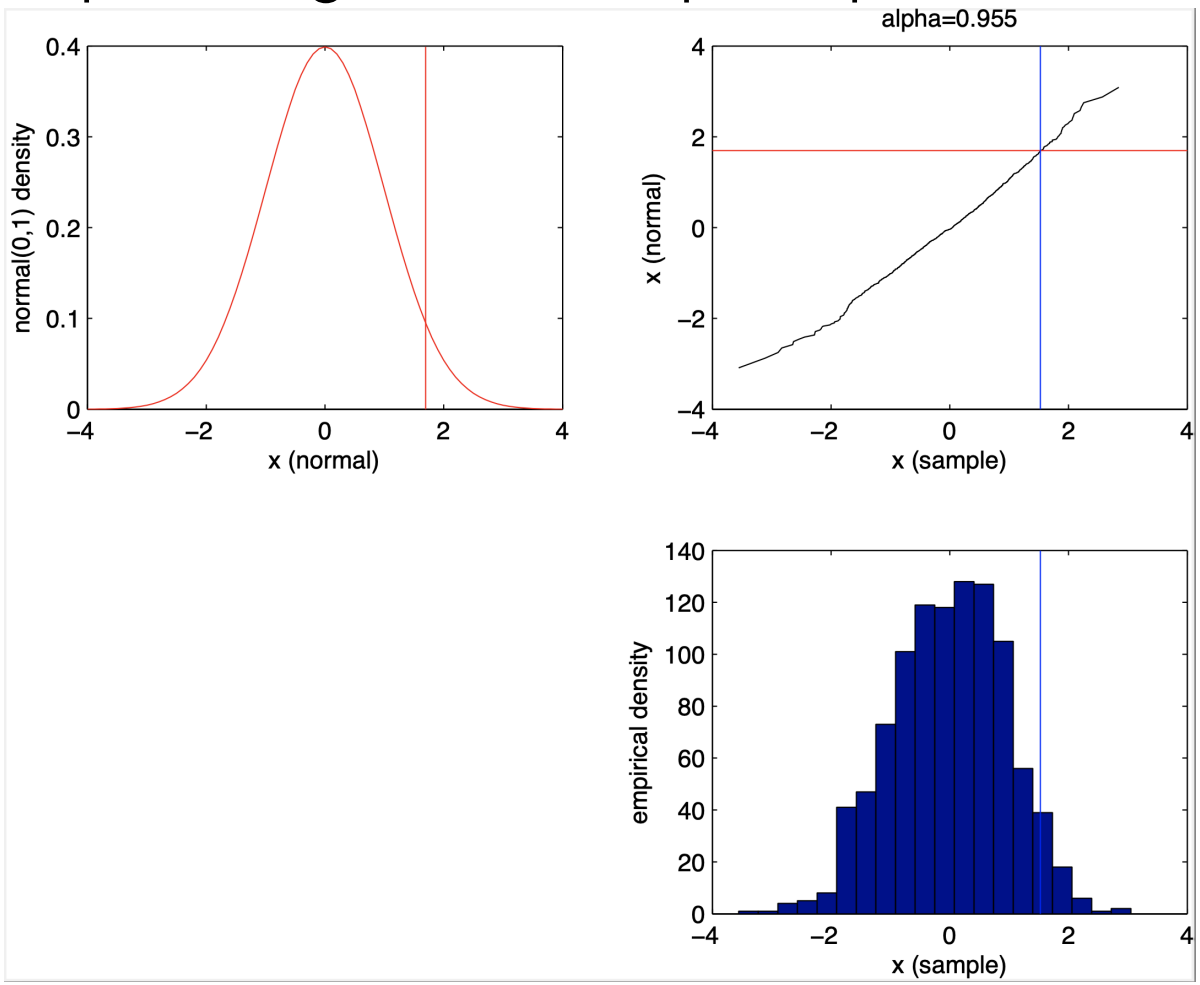
The vertical red line is at the 95% quantile of the data's histogram.



We call this the *sample quantile* or the *empirical quantile*.

Normal probability plot

A normal probability plot varies α from 0 to 1 and plots the theoretical α-quantile against the sample α-quantile.



Normal probability plot

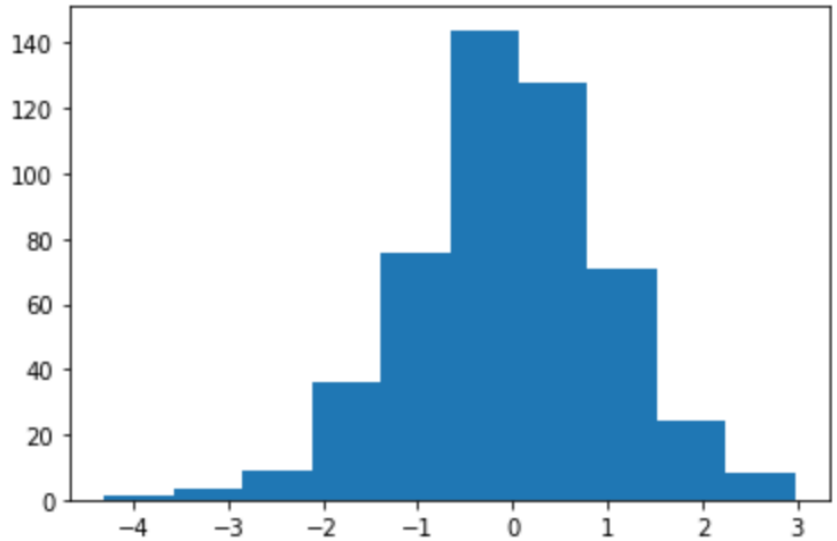
See animations (std_normal.pdf, etc.)

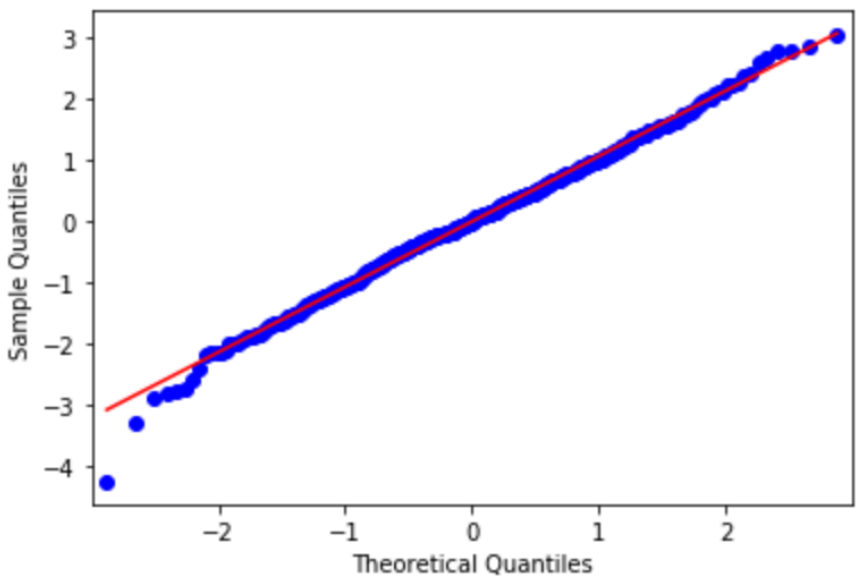
Make them using

```
import statsmodels.api as sm
sm.qqplot(model.resid , line='r')
```

Interpreting normal plots

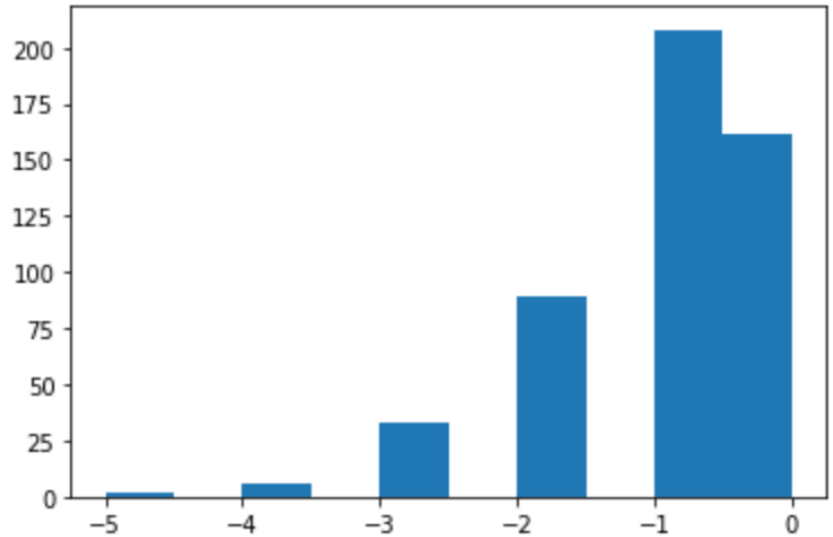
linear = normal

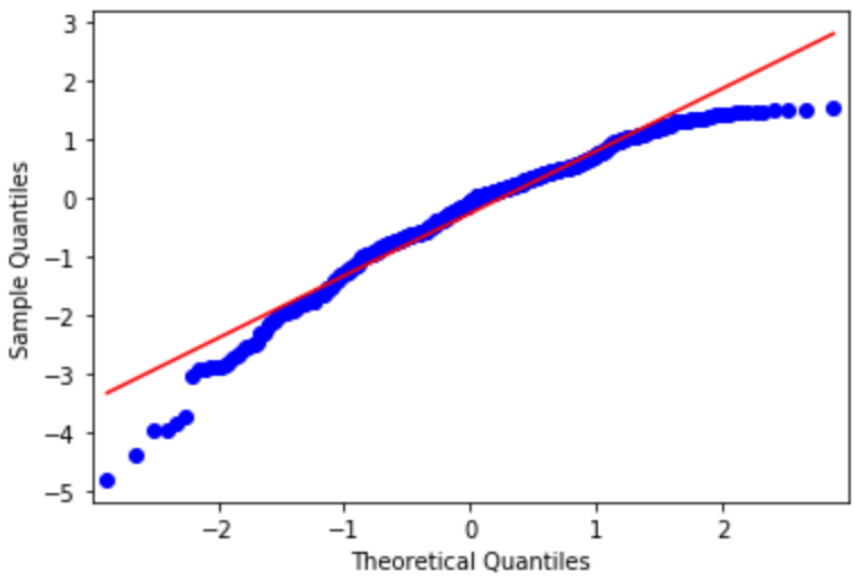




Interpreting normal plots

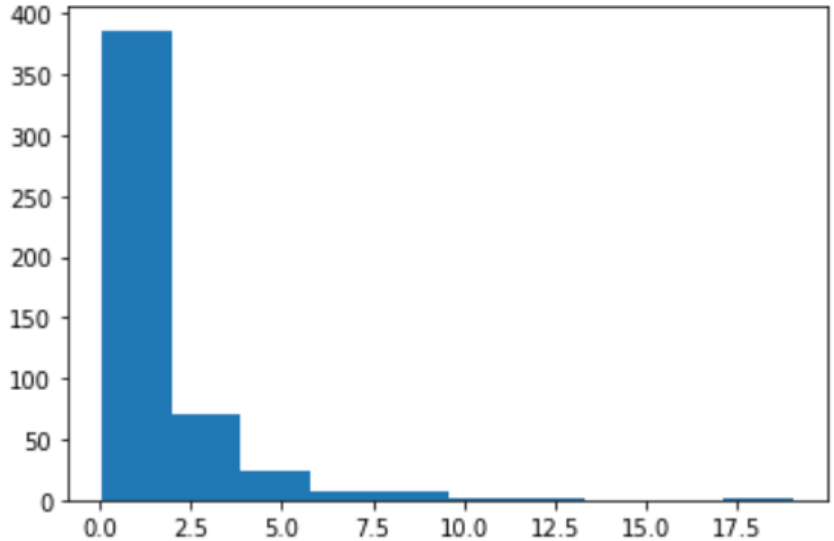
concave = left-skewed

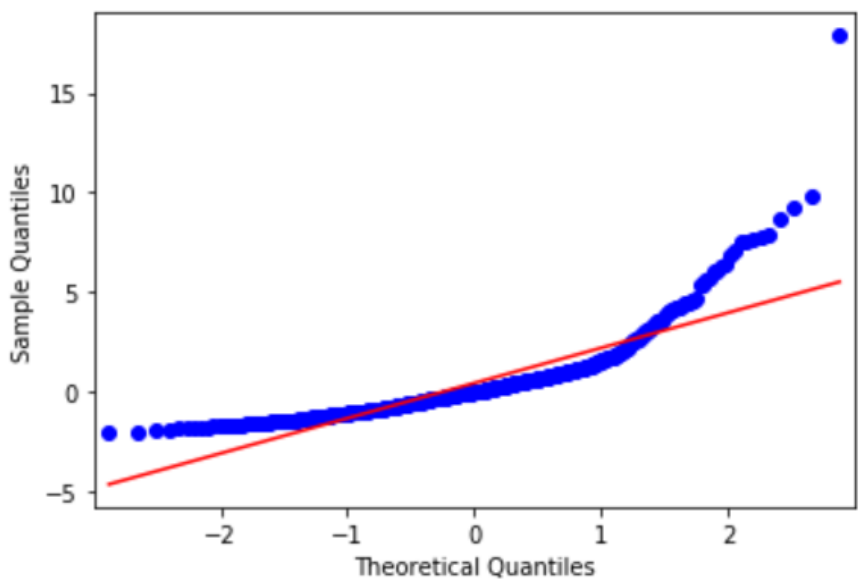




Interpreting normal plots

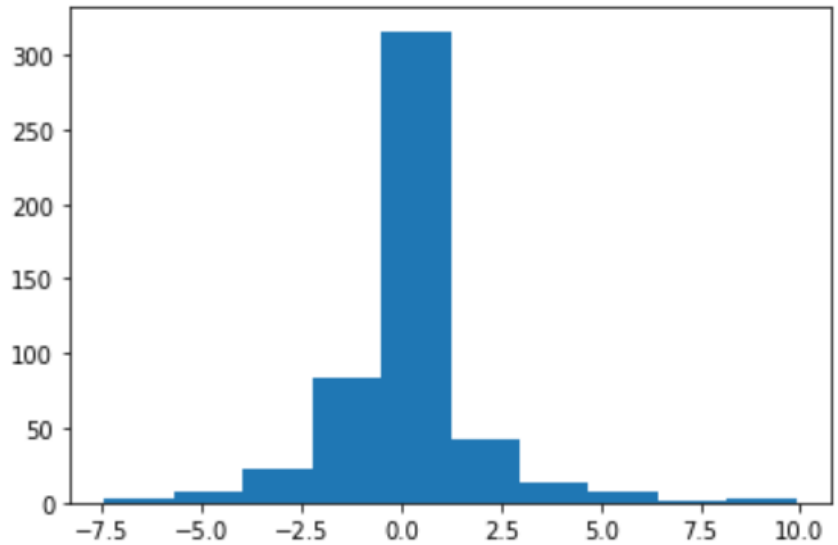
convex = right-skewed

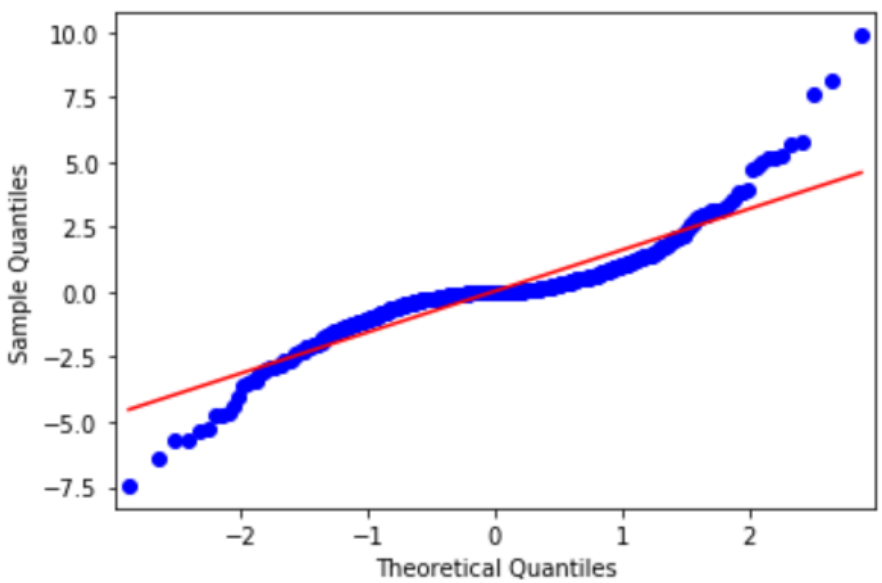




Interpreting normal plots

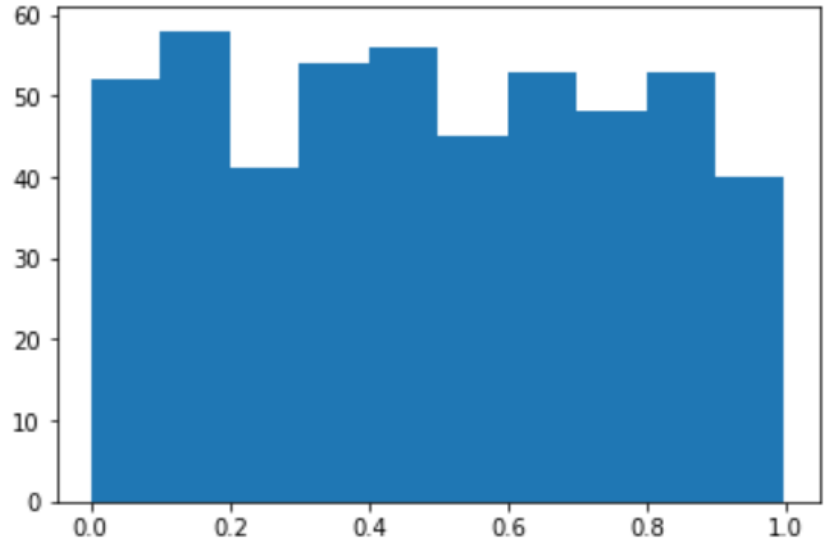
concave-convex = heavy-tailed

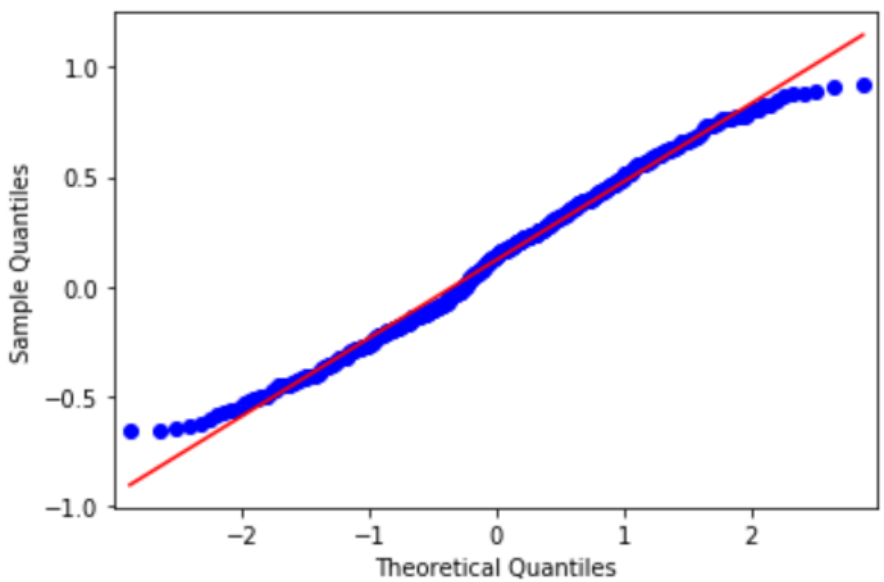




Interpreting normal plots

convex-concave = light-tailed





Checking for normal errors

This code generates q-q plots for normal residuals:

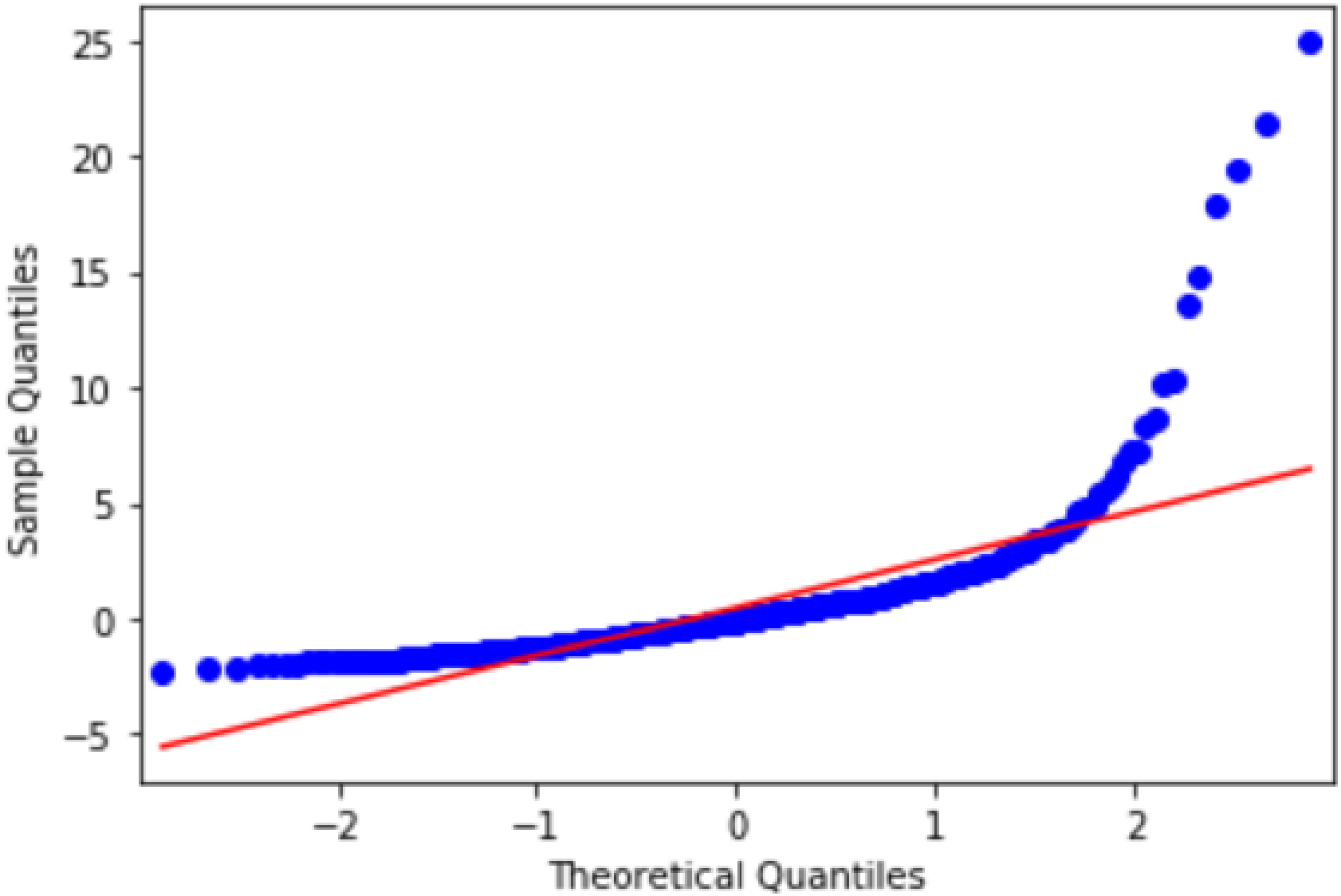
```
n=500
eps = randn(n) # normal residuals
x = 10*rand(n)
y = x + eps
model = sm.OLS(y,x).fit()

sm.qqplot(model.resid, line='r')
```

Checking for normal errors

This code generates q-q plots for residuals that are **not** normal:

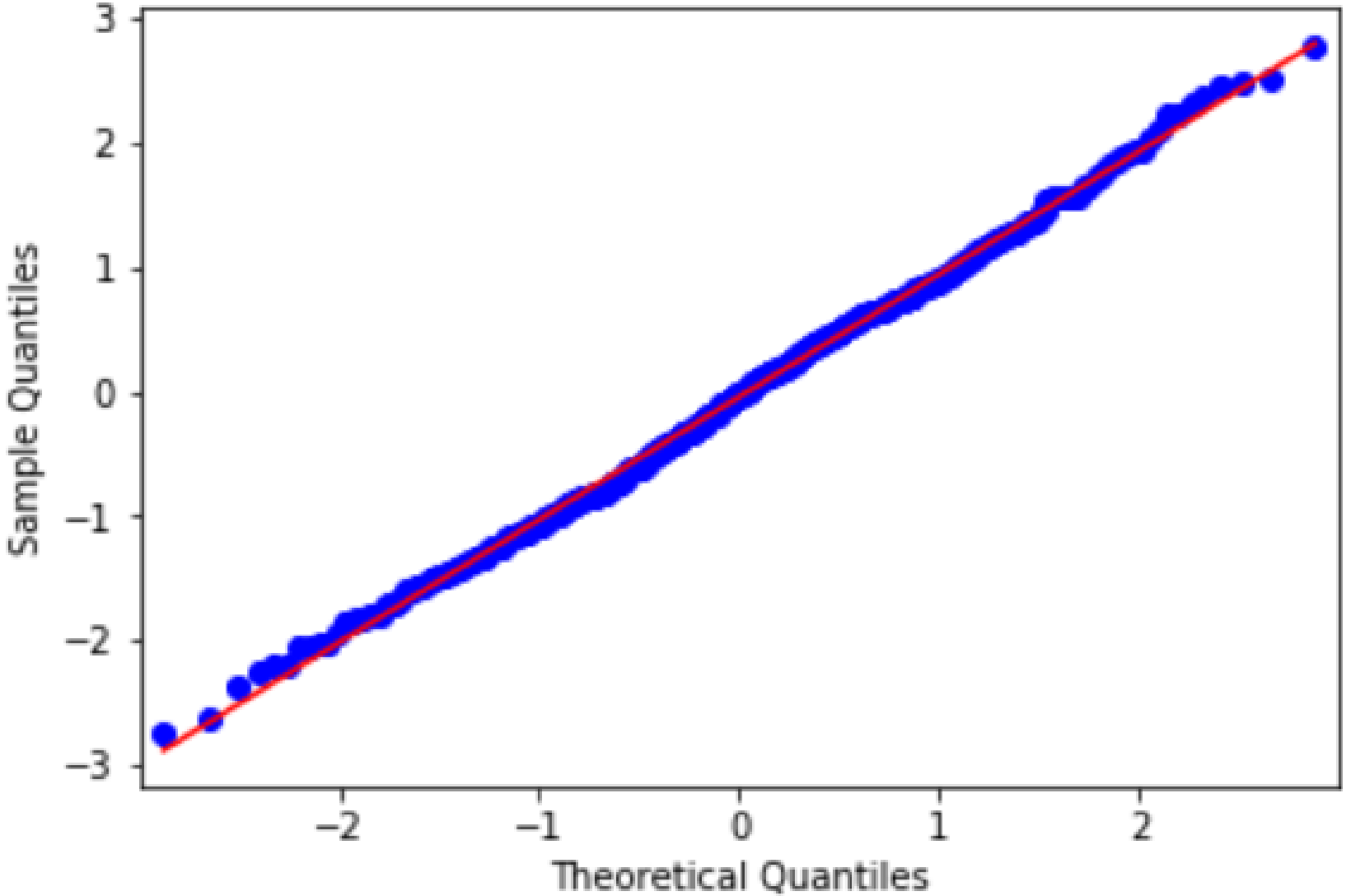
```
n=500
eps = np.exp(randn(n)) # not normal
x = 10*rand(n)
y = x + eps
model = sm.OLS(y,x).fit()

sm.qqplot(model.resid, line='r')
```

Q11: Checking for normal errors



poll: (yes) residuals are normal (no) residuals are not normal

Q12: Checking for normal errors



poll: (yes) residuals are normal (no) residuals are not normal

Strategy for regression data analysis:

1. Decide: what problem(s) are you trying to solve?
   - keep the problem in mind while doing the remaining steps
2. Find (or collect) useful data
3. Find a useful model
   - all models are wrong (George Box)
   - some models are useful
4. Check model
   - how well does the model fit the data?
5. Modify model, if necessary
6. Use model to solve problem(s)