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

Assumptions of Linear Regression

Professor Udell

Operations Research and Information Engineering
Cornell

April 14, 2020

Announcements

- submit recitation by 4:30pm ET Friday
- homework due 2:30pm ET Wednesday
- project milestone 1 due Sunday 4/19/2020 at noon

Rubric for projects

- is the report well-written and complete?
- is the report driven by a few well-chosen questions?
- does the report manage to answer those questions?
- are the visualizations easy to understand?
  - axes are labeled
  - colors are easy to read
  - chart type makes sense
- do the visualizations provide insight into the data?
  - well chosen: not too many and not too few
  - visualizations tell a story
  - not just plots-of-everything

Outline

How to check assumptions that undergird statistics?

Statistics computed are valid if $\epsilon_1, \ldots, \epsilon_n$

1. **independence I:** are mutually independent
2. **independence II:** are independent of covariates
3. **normality:** are normally distributed
4. **homoskedasticity:** have a constant variance

To check whether these assumptions are true, we must look at the residuals

How to check assumptions that undergird statistics?

Statistics computed are valid if $\epsilon_1, \ldots, \epsilon_n$

1. **independence I:** are mutually independent
2. **independence II:** are independent of covariates
3. **normality:** are normally distributed
4. **homoskedasticity:** have a constant variance

To check whether these assumptions are true, we must look at the residuals

Demo:
`https://github.com/madeleineudell/orie3120-sp2020/blob/master/demos/test-assumptions.ipynb`

Residuals analysis: mutual independence

Let's look at each assumption and see how it can be checked.

Assumption 1. $\epsilon_1, \ldots, \epsilon_n$ are mutually independent

- this assumption might be violated if the observations are in time or spatial order
- check by: plotting $\widehat{\epsilon}_i$ versus $\widehat{\epsilon}_{i-1}$
  - should see no pattern

Residuals analysis – checking mutual independence with a scatterplot

This code will show scatterplots from data with mutual independence.

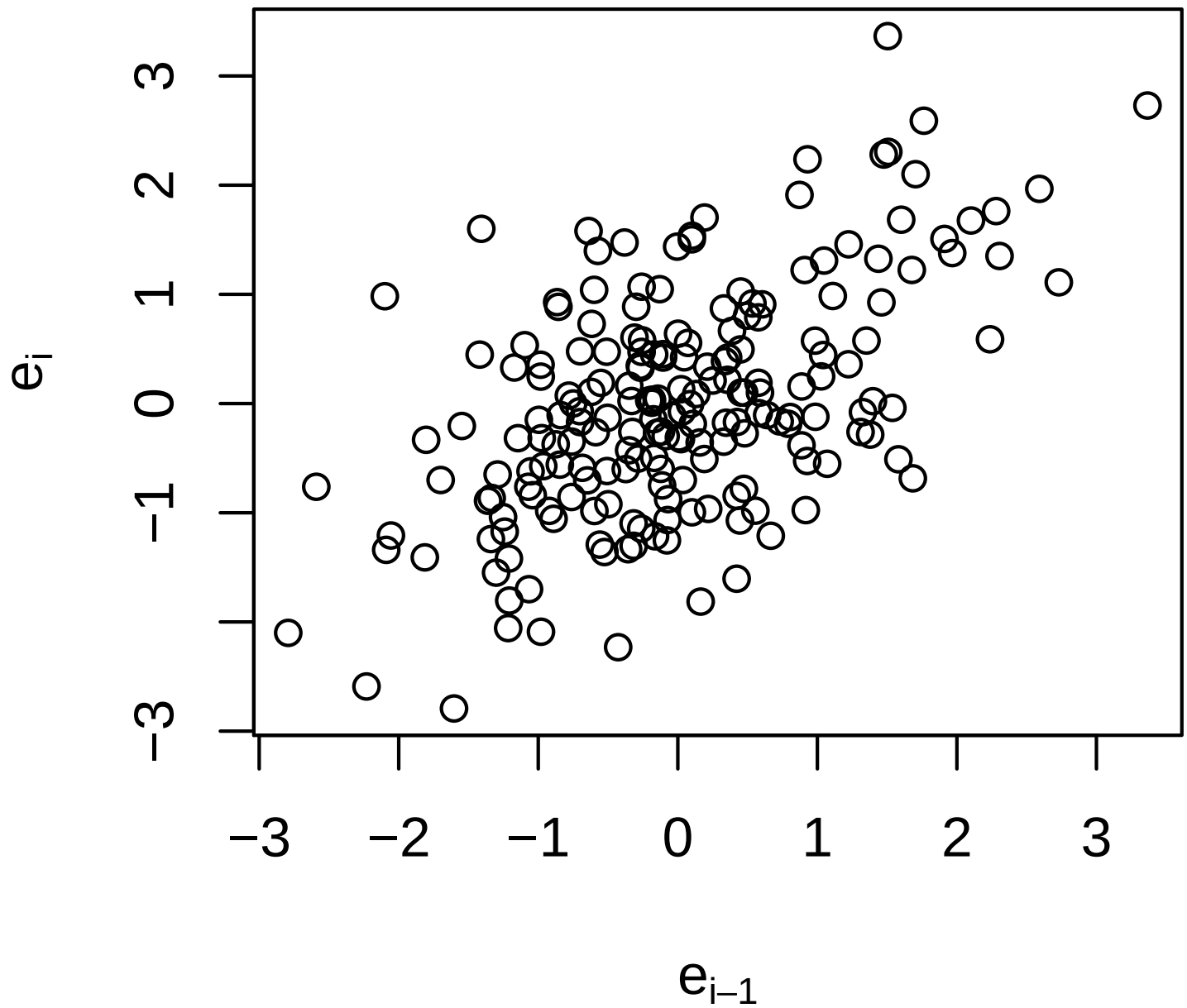
```
# generate data
n   = 500 # number of observations
eps = randn(n) # independent normal(0,1)
x   = 10*rand(n) # uniform(0,10)
y   = x + eps

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

plt.scatter(resid[:-1], resid[1:])
```

Residuals analysis – checking mutual independence with a scatterplot

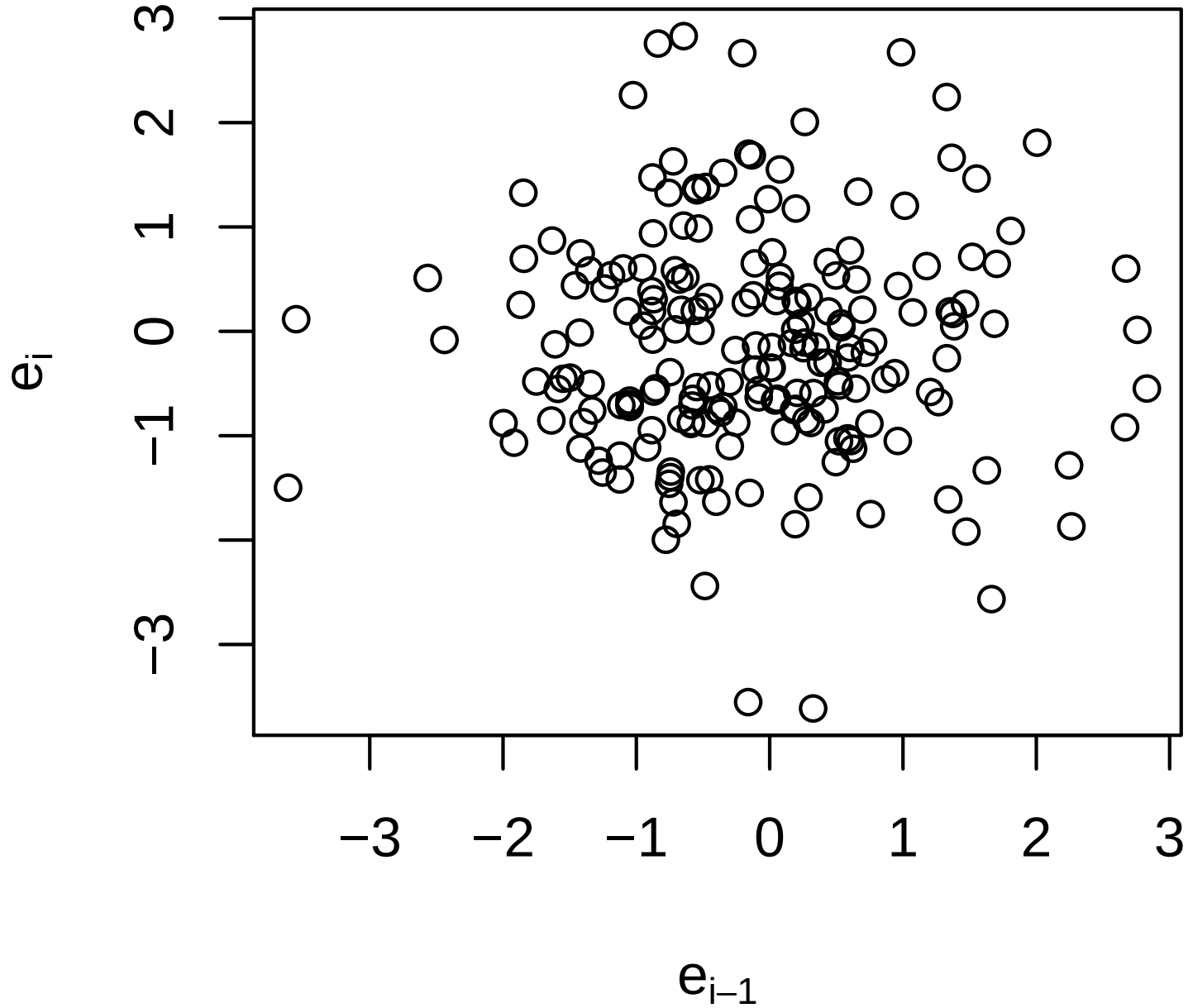
This code will show scatterplots from data **without** mutual independence.

```
# generate data
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

plt.scatter(resid[:-1], resid[1:])
```

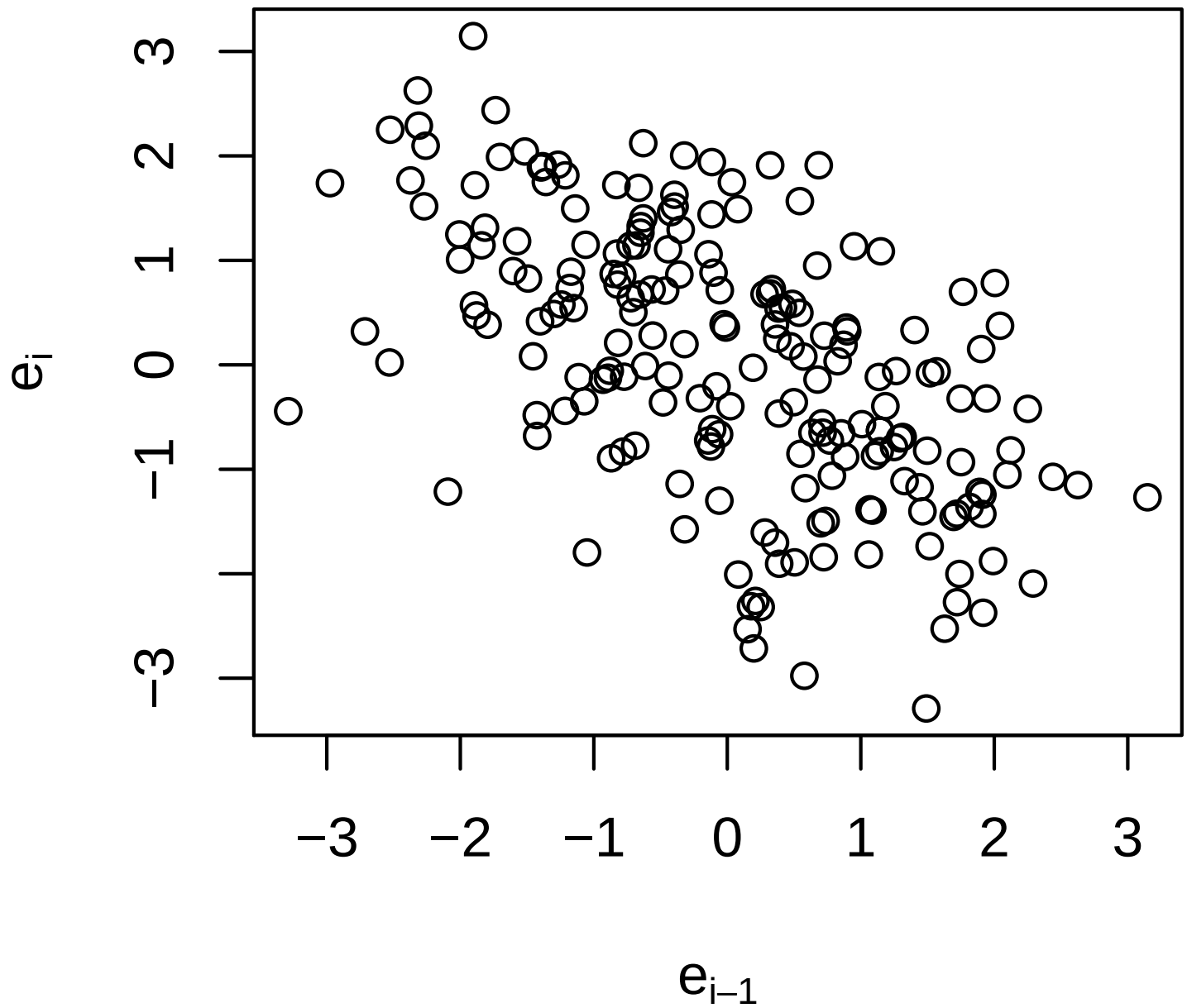
Mutually independent residuals? yes/no



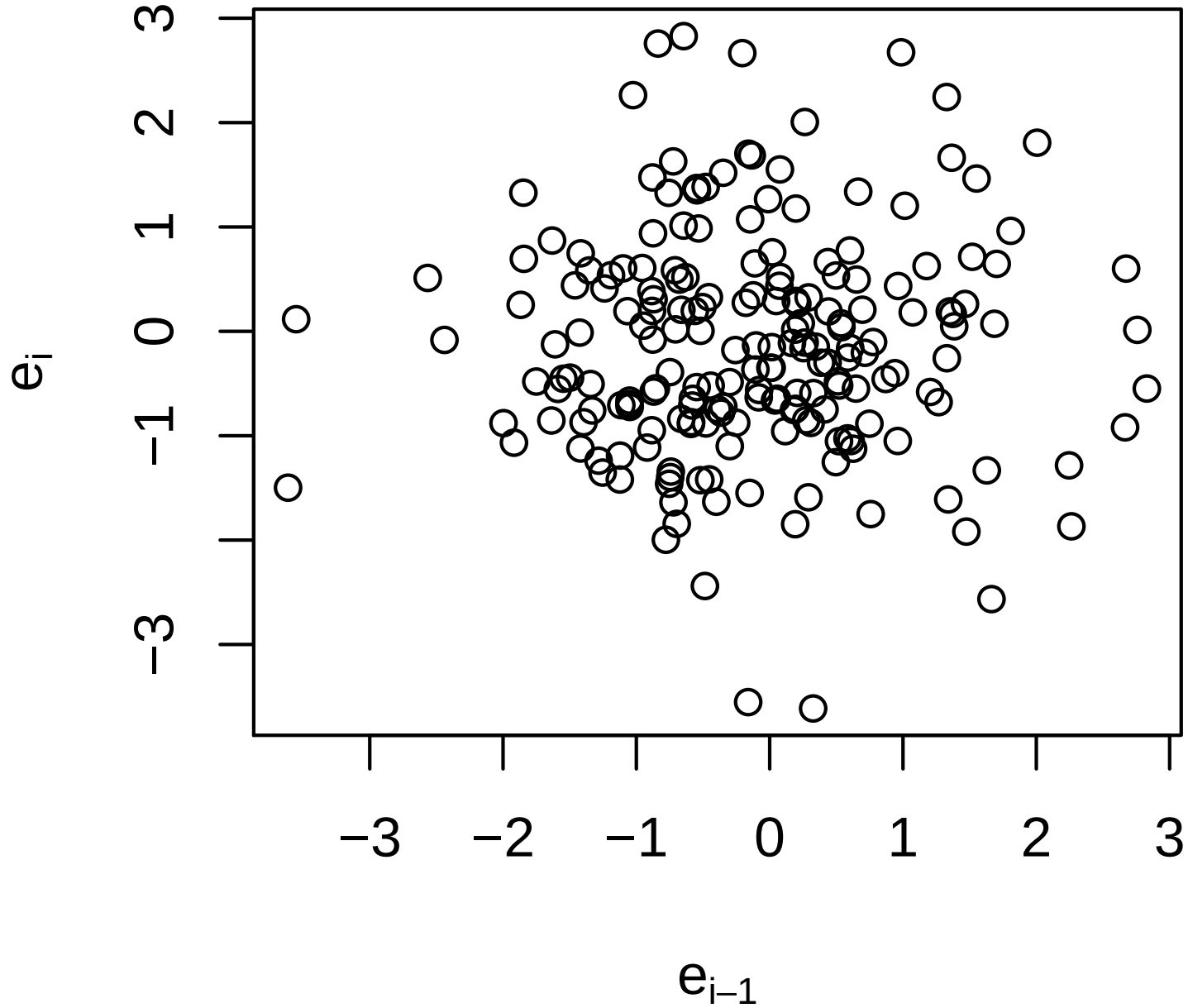
Mutually independent residuals? yes/no



Mutually independent residuals? yes/no



Mutually independent residuals? yes/no



Check mutual independence with autocorrelations

- plot the autocorrelation function:

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

  - $r(t)$ should be 0 for all $t > 0$ (except for random variation)
  - no (or only a few) autocorrelations should be outside the test bounds
  - t is called the lag
- the scatterplots only looked at lag $= 1$
  - of course, we could have looked at other lags
  - but autocorrelations let us look at all lags simultaneously

Check mutual independence with autocorrelations

This code plots the autocorrelation for data with mutual independence.

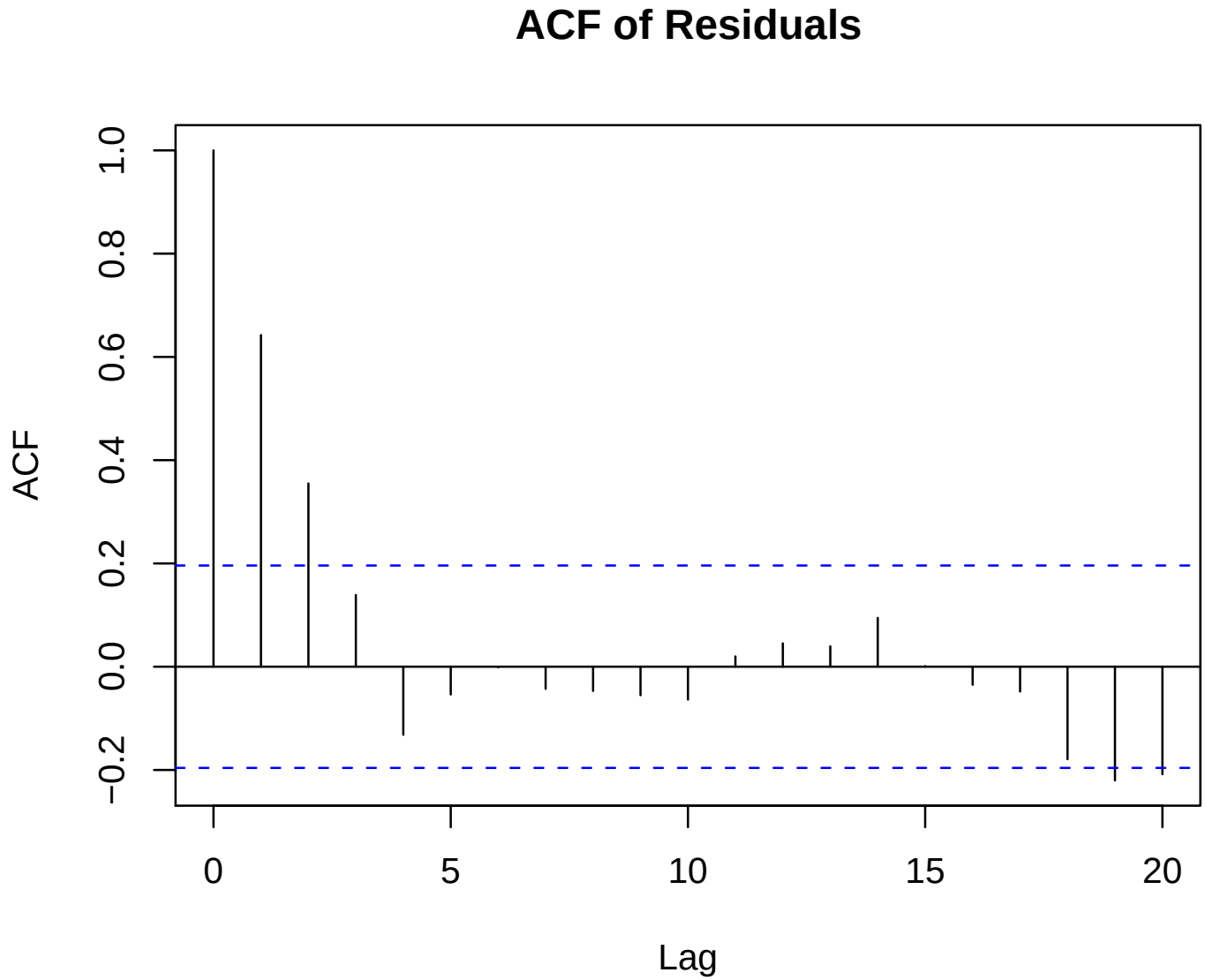
```
n    = 500    # number of observations
eps  = randn(n) # independent normal(0,1)
x    = 10*rand(n) # uniform(0,10)
y    = x + eps

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

plt.acorr(resid)
```

Check mutual independence with autocorrelations

This code plots the autocorrelation for data **without** mutual independence.

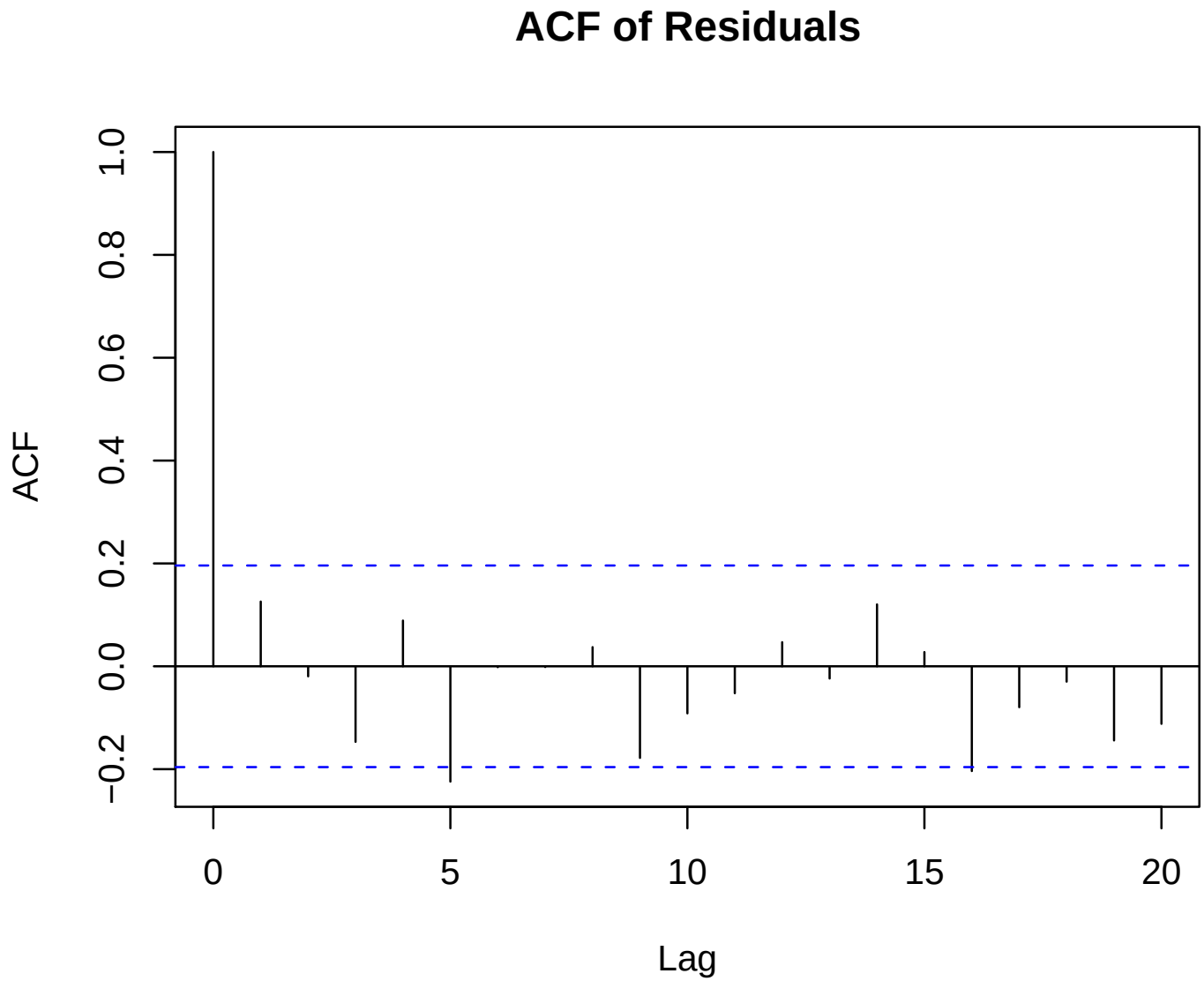
```
n   = 500   # number of observations
a   = 1     # use this to control the correlation
w   = randn(n+1) # independent normal(0,1)
eps = w[:-1] + a*w[1:] # normal, not independent
x   = 10*rand(n) # uniform(0,10)
y   = x + eps

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

plt.acorr(resid)
```

Check mutual independence with autocorrelations



quick poll: (yes) independent (no) not independent

Check mutual independence with autocorrelations



quick poll: (yes) independent (no) not independent

Residuals analysis – linear in the predictors

Assumption 2. model is linear in the predictors (the $X_{i,j}$)

- equivalently, $\epsilon_1, \ldots, \epsilon_n$ are independent of all $X_{i,j}$
- Check by: plotting $\widehat{\epsilon}_i$ versus $X_{i,j}$ for $j = 1, \ldots, p$
- we should see that the average value of the $\widehat{\epsilon}_i$ does not depend on $X_{i,j}$.
- if it does, then there is a problem

Residuals analysis – linear in the predictors

Plot residuals vs covariates to test linearity

```
plt.subplot(2,1,1)
p = plt.scatter(x,y,marker='o',label="observed")
plt.scatter(x,yhat,marker="+",color="red",label="
plt.legend()

plt.subplot(2,1,2)
plt.scatter(x,resid)
plt.xlabel("x")
plt.ylabel("residual")
```

Residuals analysis – linear in the predictors

This code forms a model for which outcome is **not** linear in the predictor.

```
n    = 500 # number of observations
eps  = randn(n) # independent normal(0,1)
x    = 10*rand(n) # uniform(0,10)
y    = x + x**2 + eps

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

Residuals analysis – linear in the predictors

Here's how we fix the fit on the previous slide:
use the square as a feature

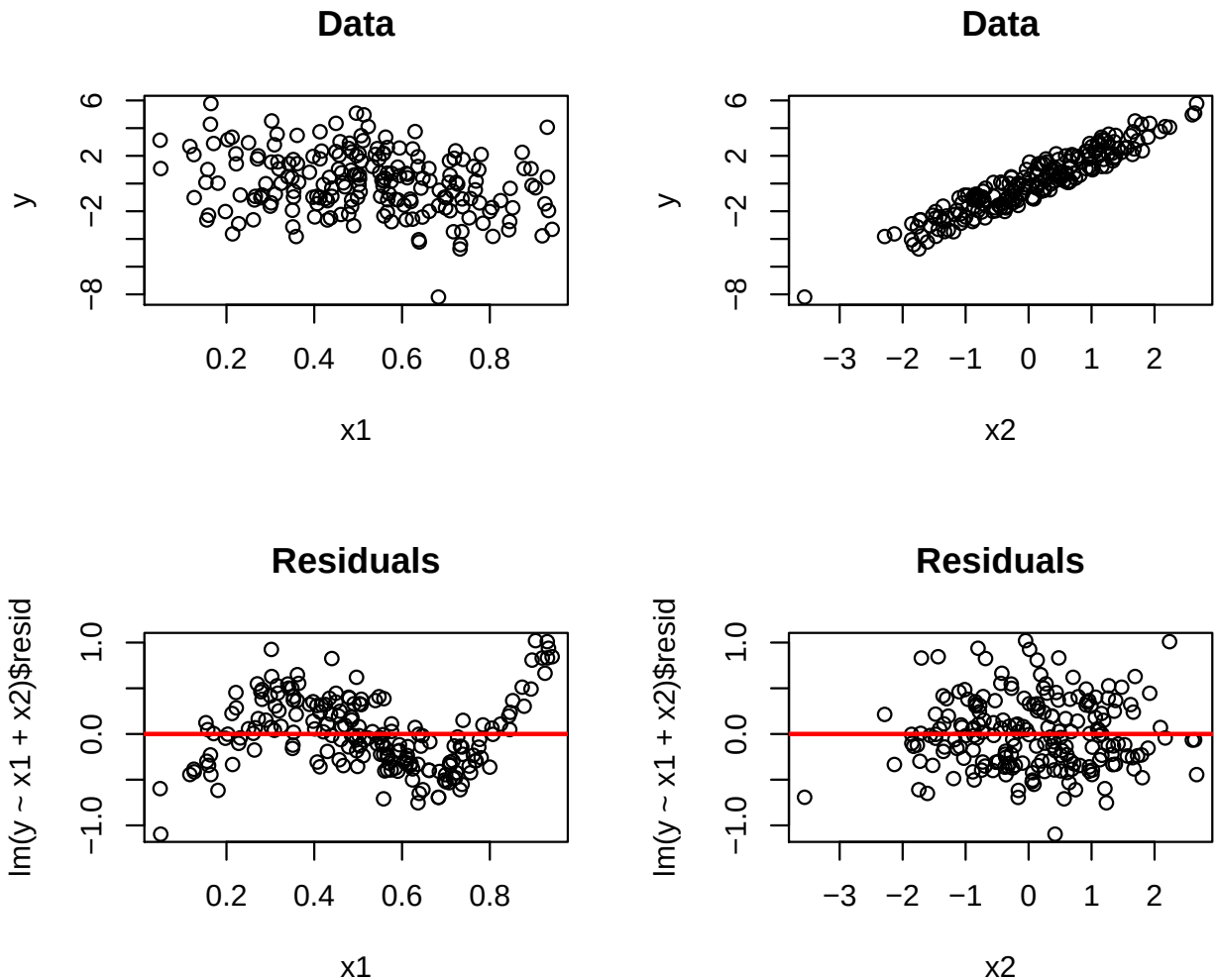
```
df = pd.DataFrame()
df['x'] = x
df['xsq'] = x**2
model = sm.OLS(y, df).fit()
```

Residuals detect nonlinearity better than raw data

In the next slide, data are simulated from:

```
n = 200
x1 = beta(2,2,n)
x2 = randn(n)
y = sin(2*math.pi*x1) + 2*x2 + .23*randn(n)
```

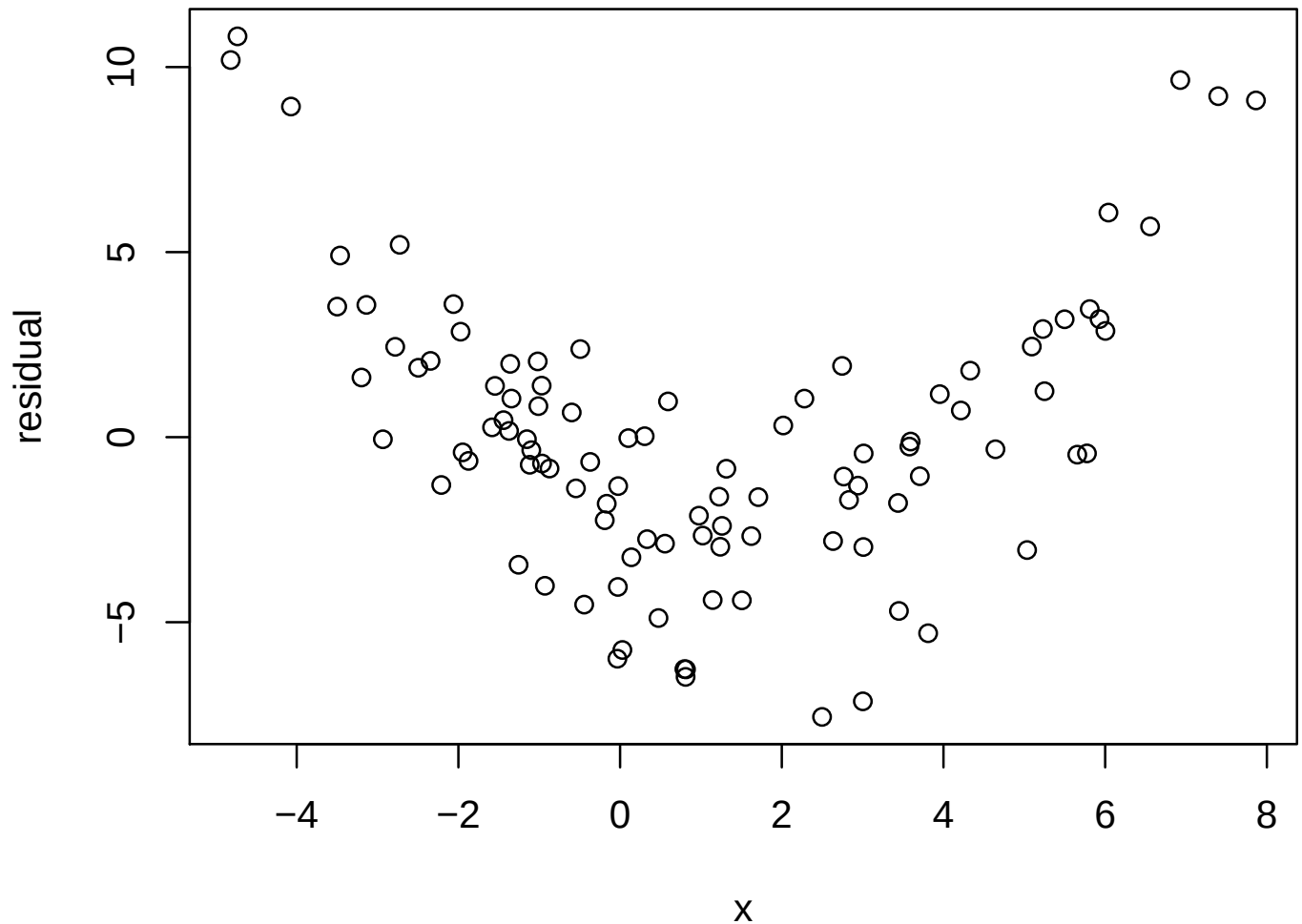
Residuals detect nonlinearity better than raw data



Raw data: Scatter due to X_2 obscures nonlinearity in X_1

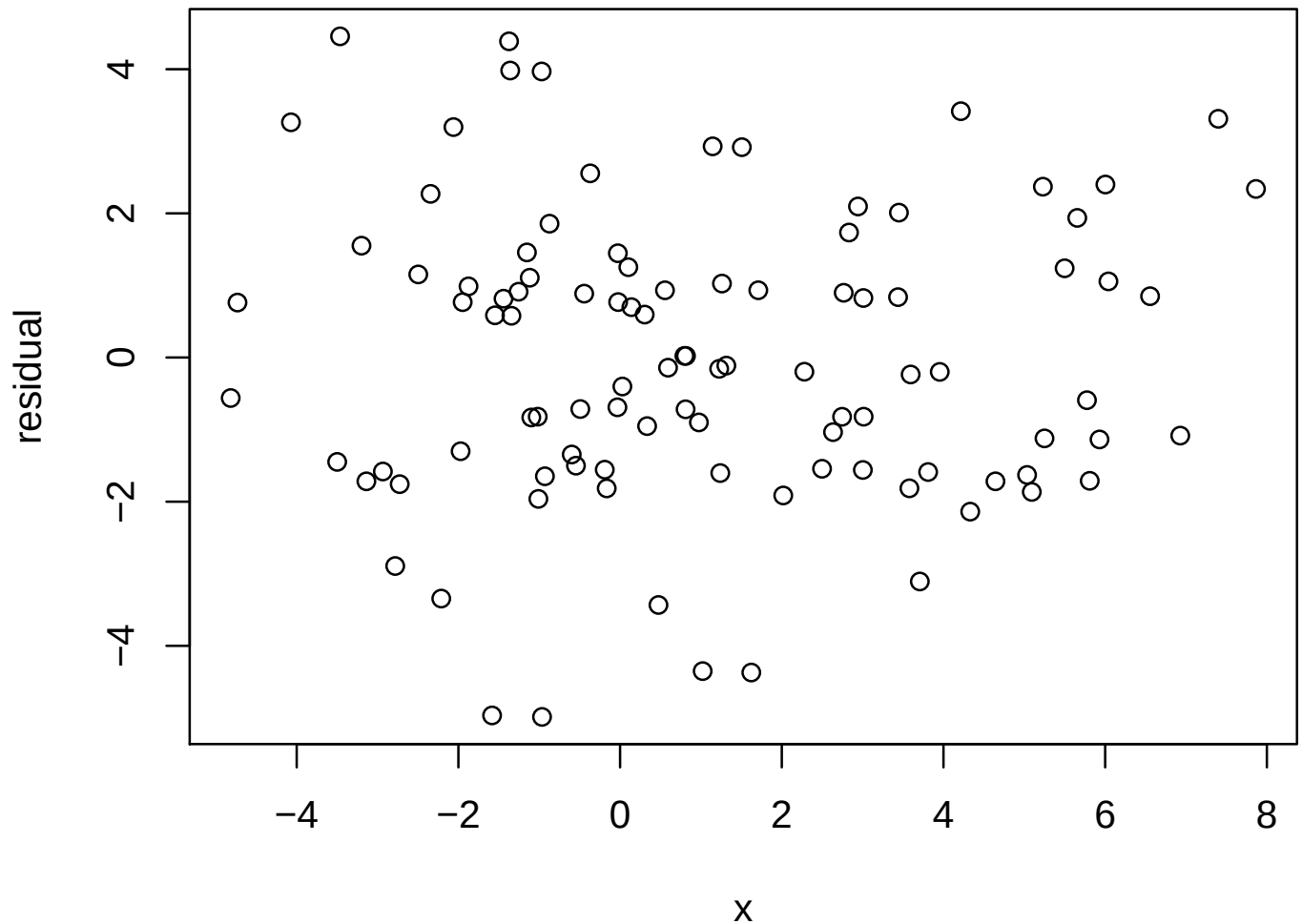
Residuals: Scatter due to X_2 is removed and nonlinearity in X_1 is revealed

Checking for nonlinearity



quick poll: (yes) linear in x (no) not linear in x

Checking for nonlinearity



quick poll: (yes) linear in x (no) not linear in x

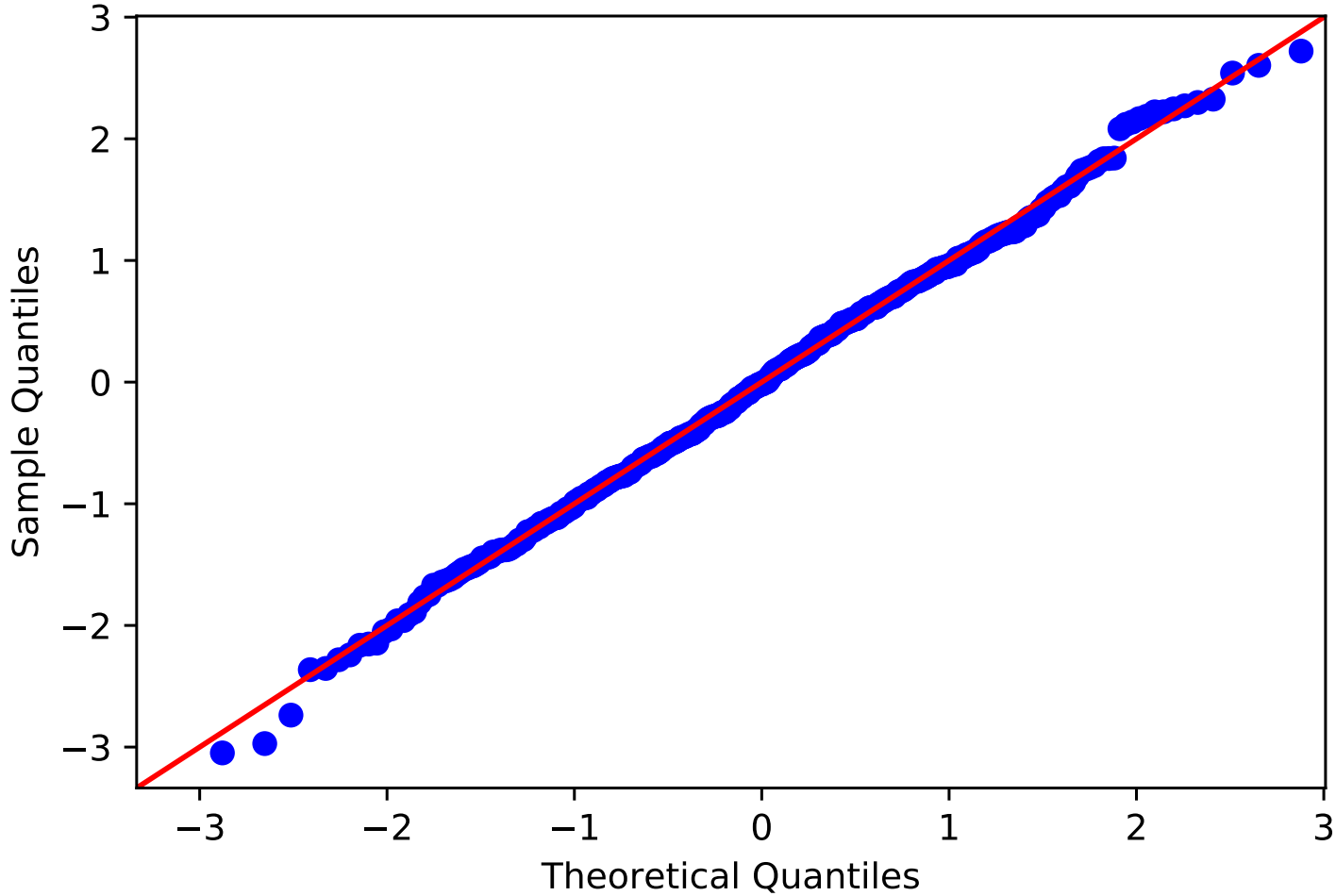
Residuals analysis – normal distribution

Assumption 3. $\epsilon_1, \ldots, \epsilon_n$ are normally distributed

- normal probability plot
- should see a straight line
- a pattern means skewness or heavy-tails

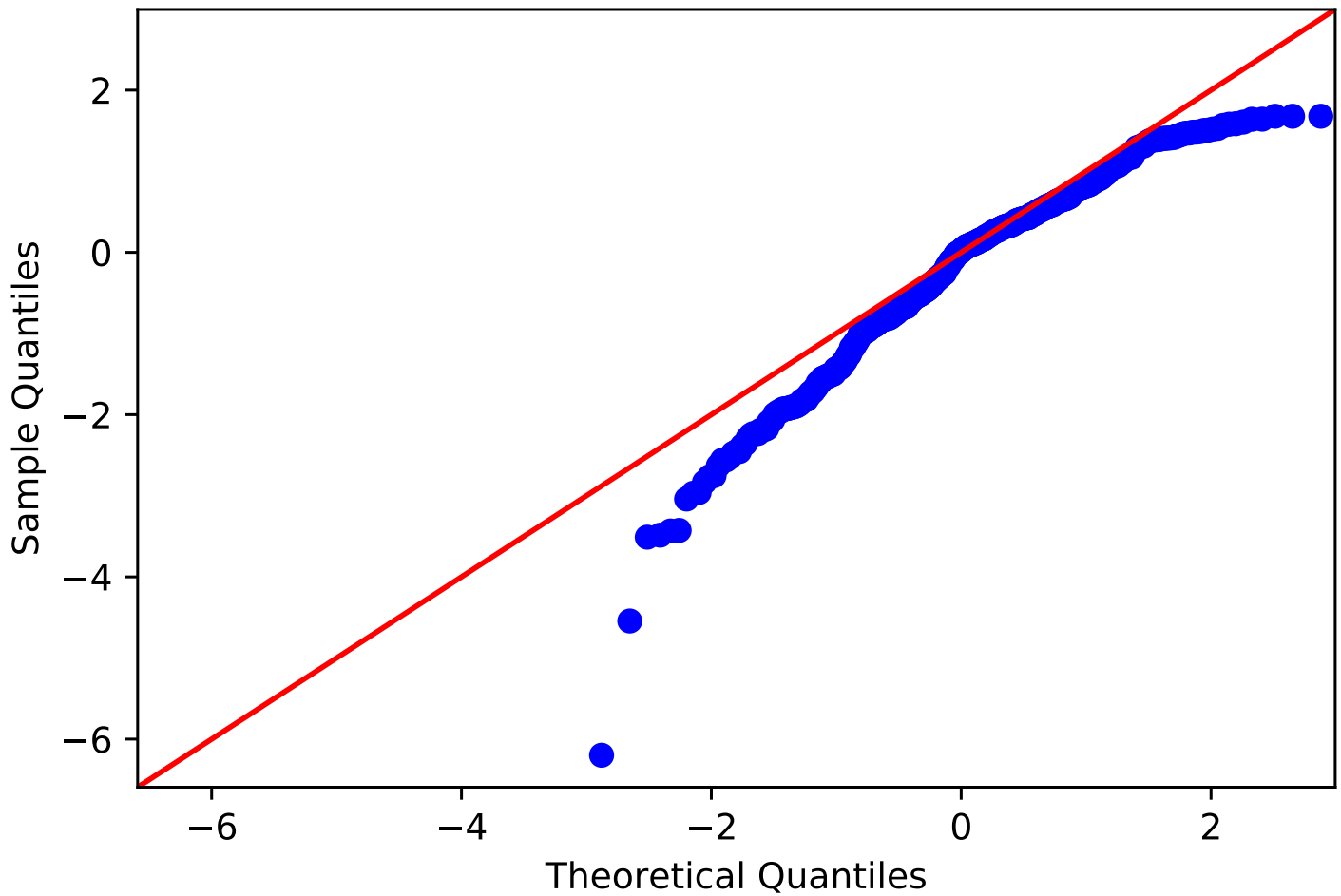
Interpreting normal plots

linear = normal



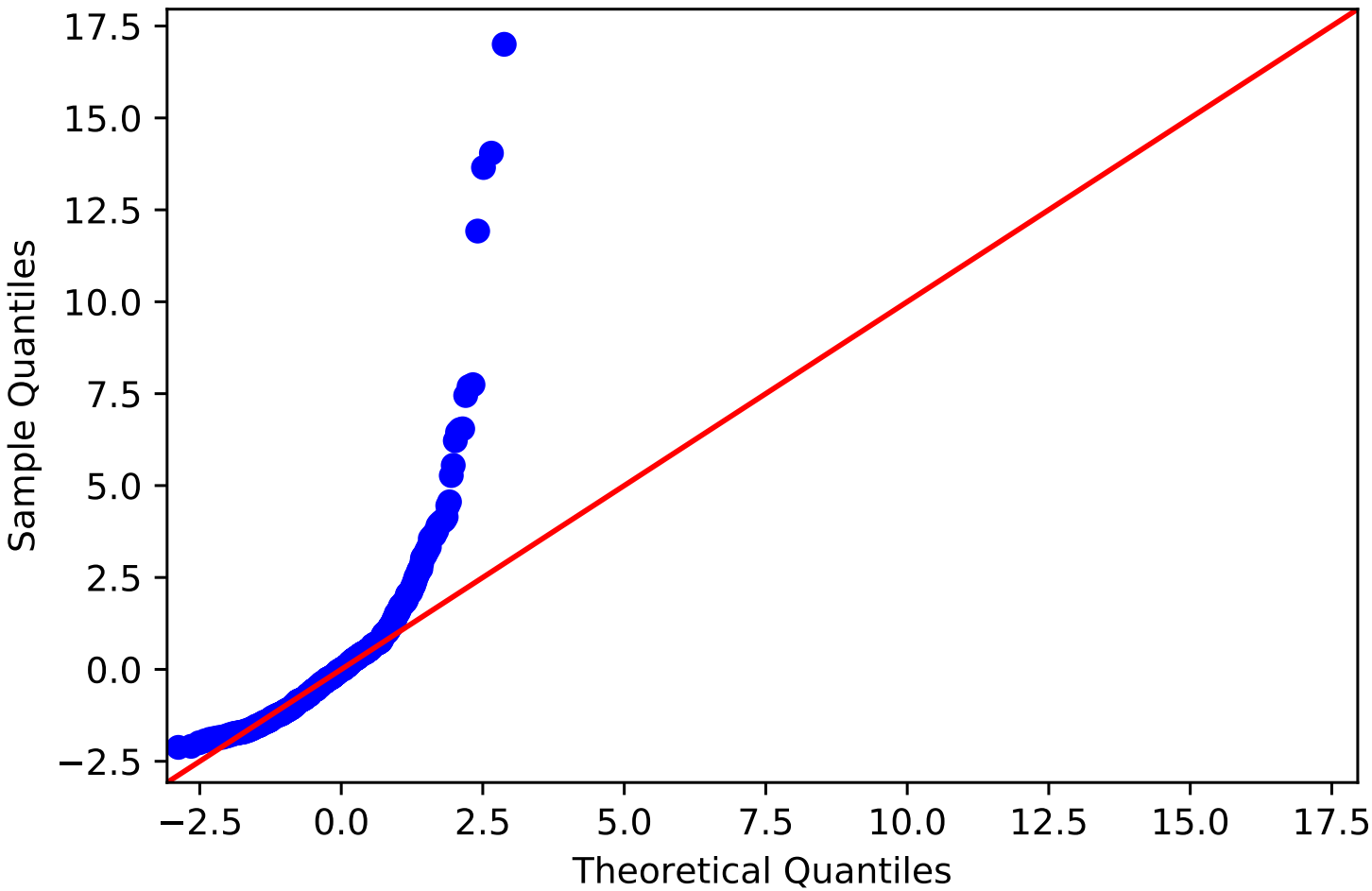
Interpreting normal plots

concave = left-skewed



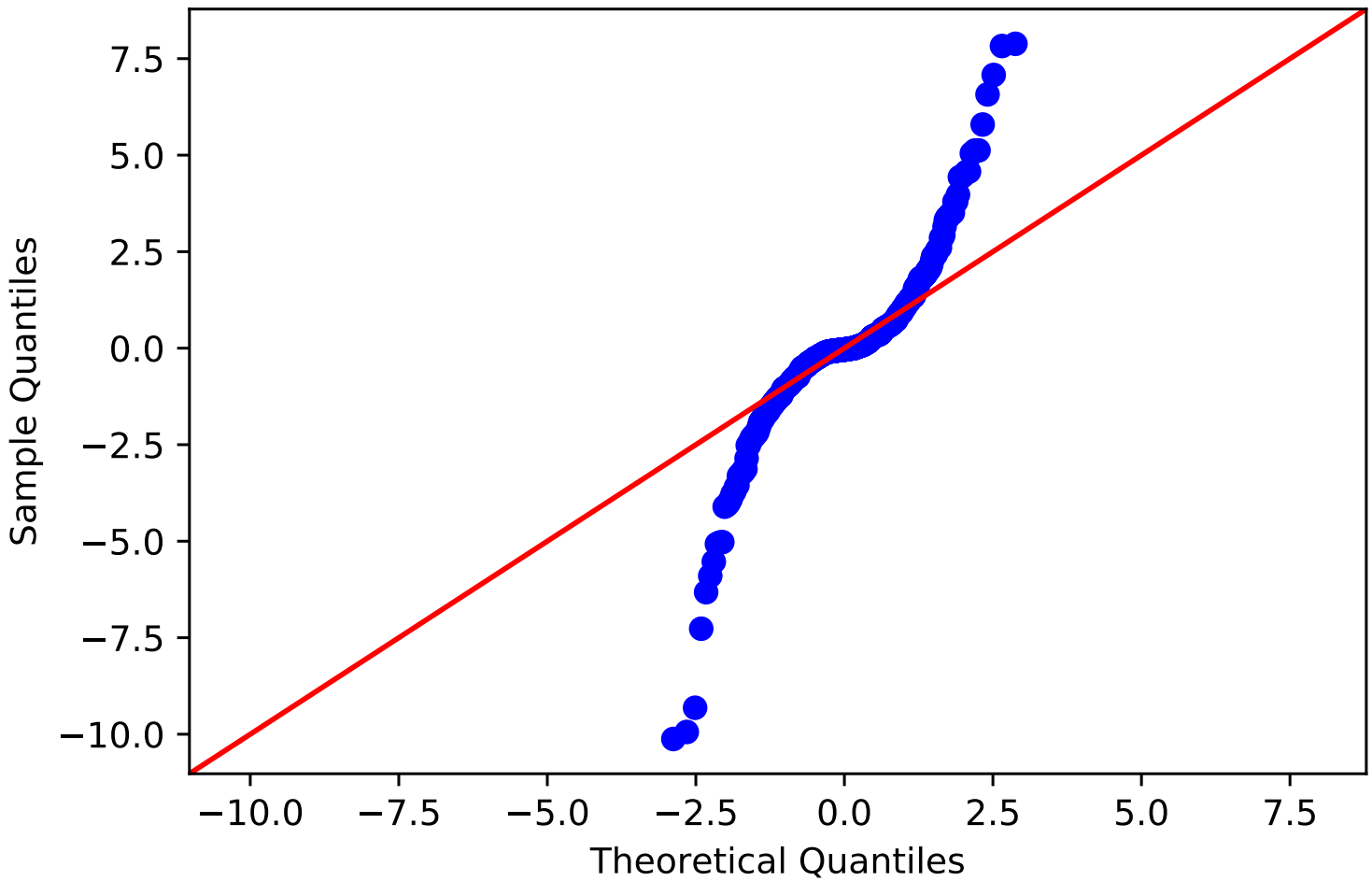
Interpreting normal plots

convex = right-skewed



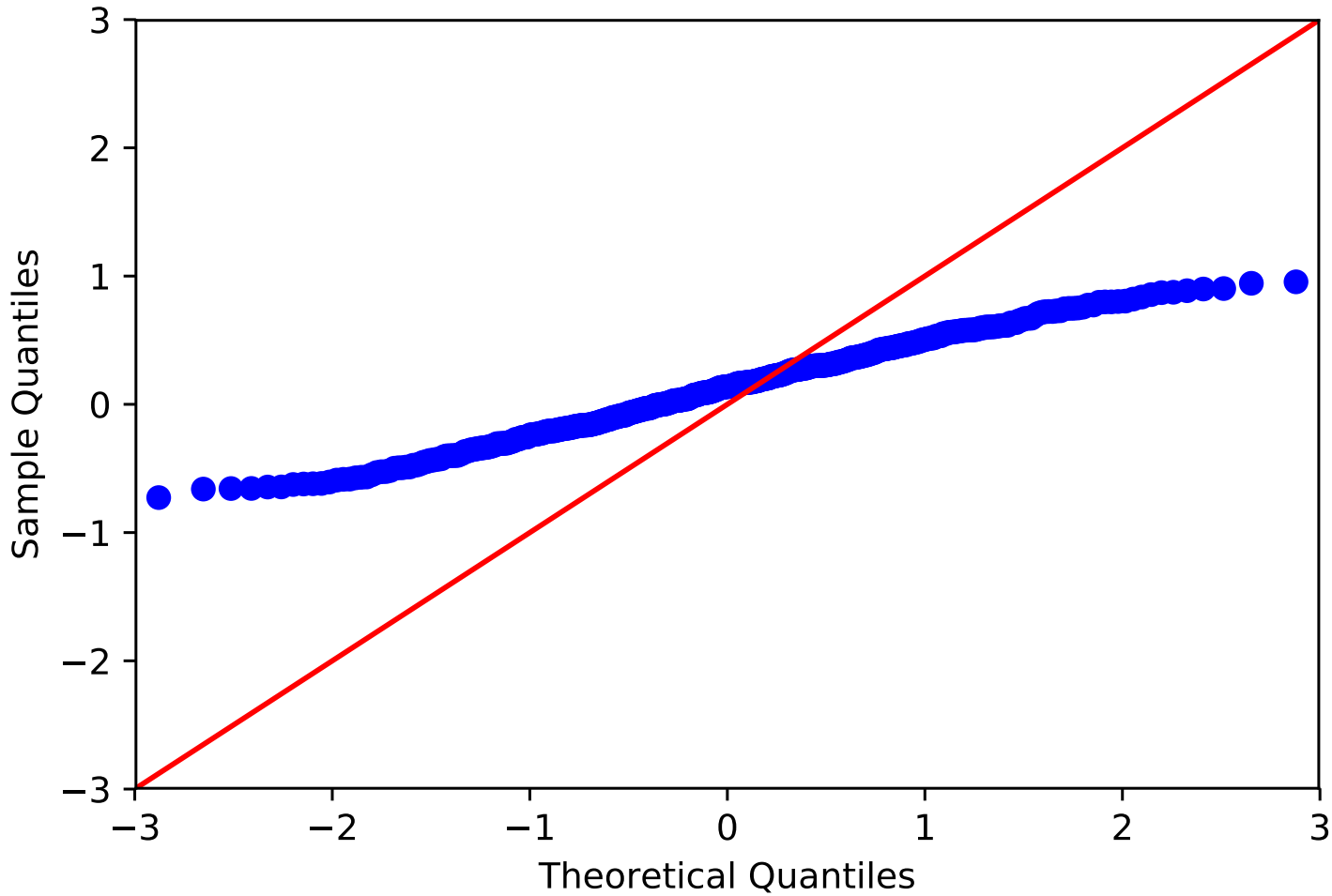
Interpreting normal plots

concave-convex = heavy-tailed



Interpreting normal plots

convex-concave = light-tailed



Checking for normal errors

This code generates q-q plots for normal residuals:

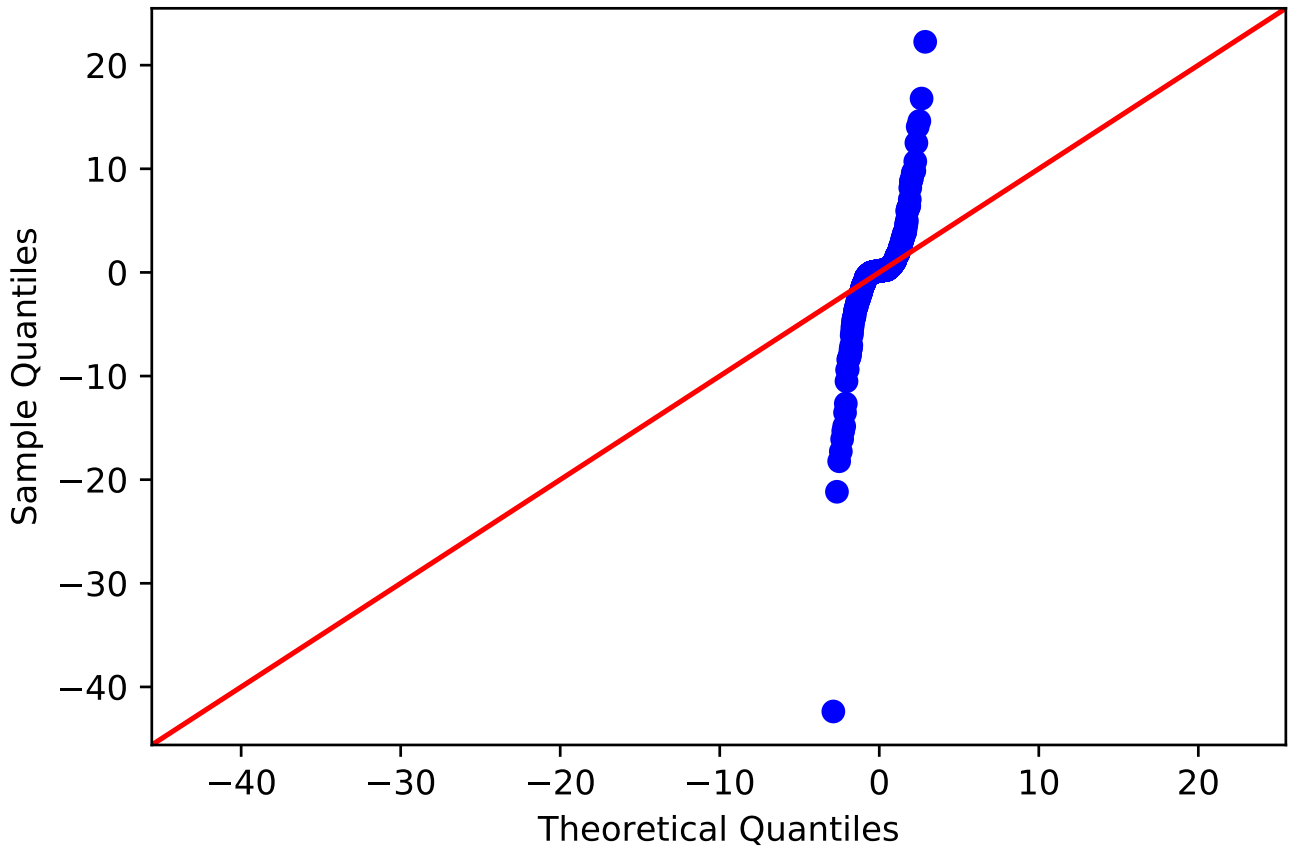
```
n=500
eps = randn(n) # normal residuals
x = 10*rand(n)
y = x + eps
model = sm.OLS(y,x).fit()

sm.qqplot(model.resid, line='45');
```

Checking for normal errors

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

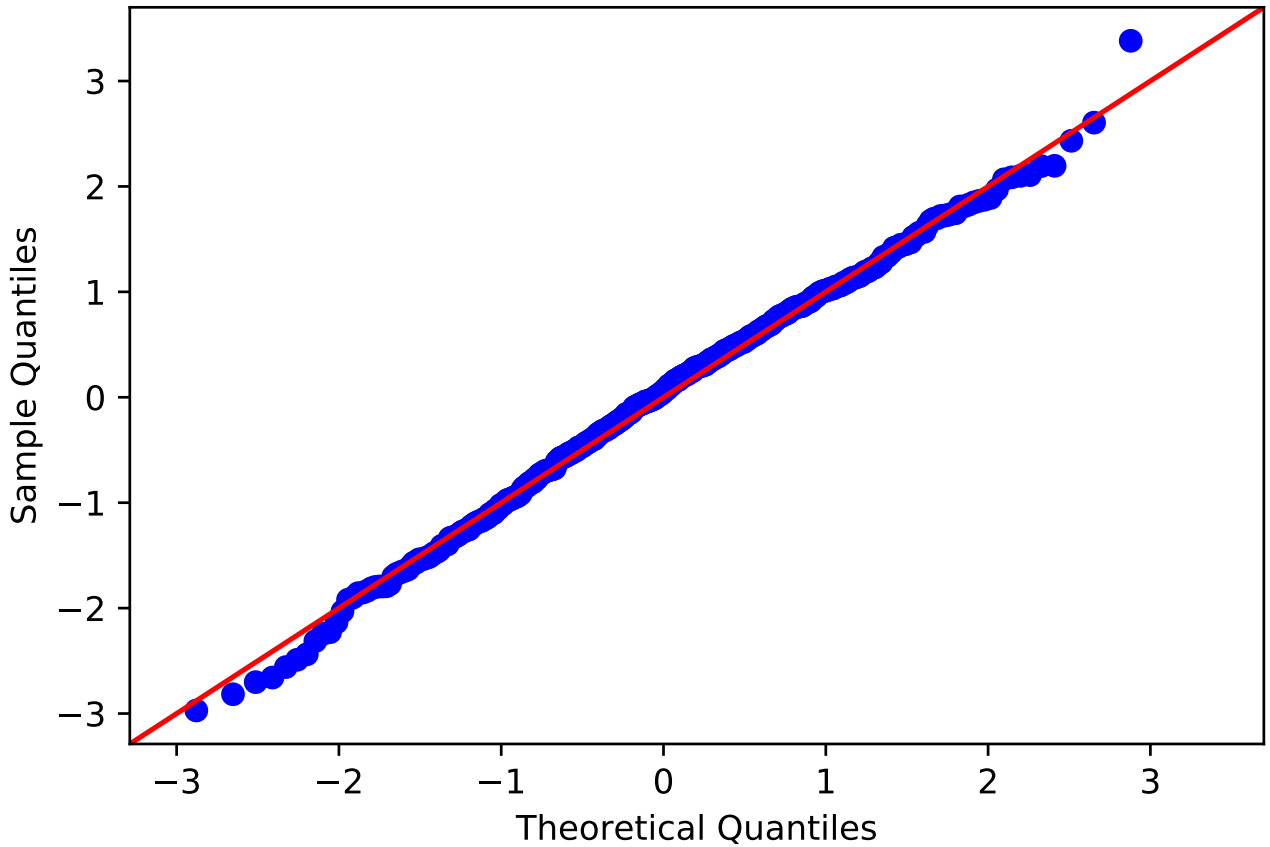
```
n=500
eps = exp(randn(n)) # not normal
x = 10*rand(n)
y = x + eps
model = sm.OLS(y,x).fit()

sm.qqplot(model.resid, line='45');
```

Checking for normal errors



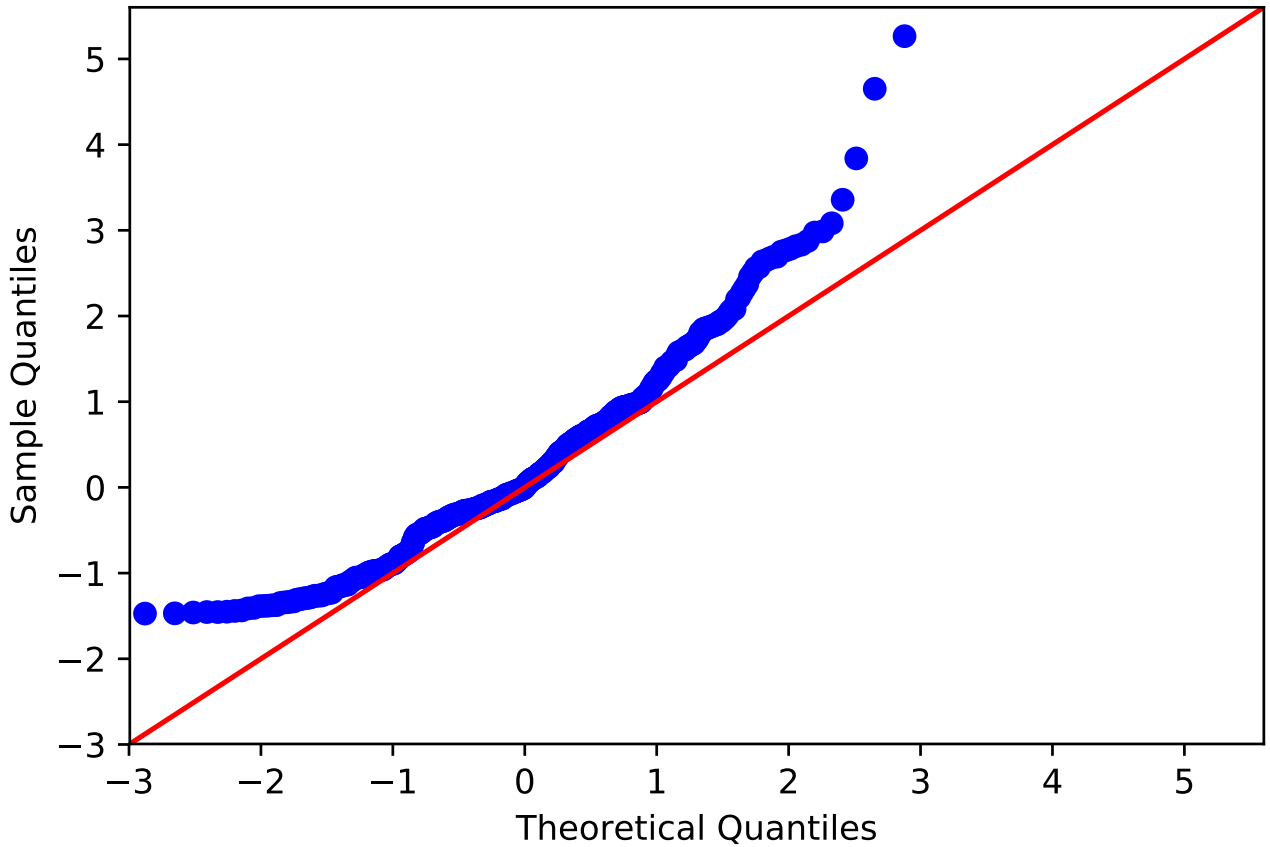
poll: (yes) residuals are normal (no) residuals are not normal

Checking for normal errors



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

Checking for normal errors



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

Residuals analysis – constant variance

Assumption 4. $\epsilon_1, \ldots, \epsilon_n$ have a constant variance

- plot absolute residuals against fitted values
- plot absolute residuals against $X_{i,j}$ for each j
  - should see that the distribution does not depend on $X_{i,j}$
  - if it does, then the variance is not constant
- we call non-constant variance "heteroscedasticity"

Checking for normal errors

This code generates absolute residual plots for constant variance:

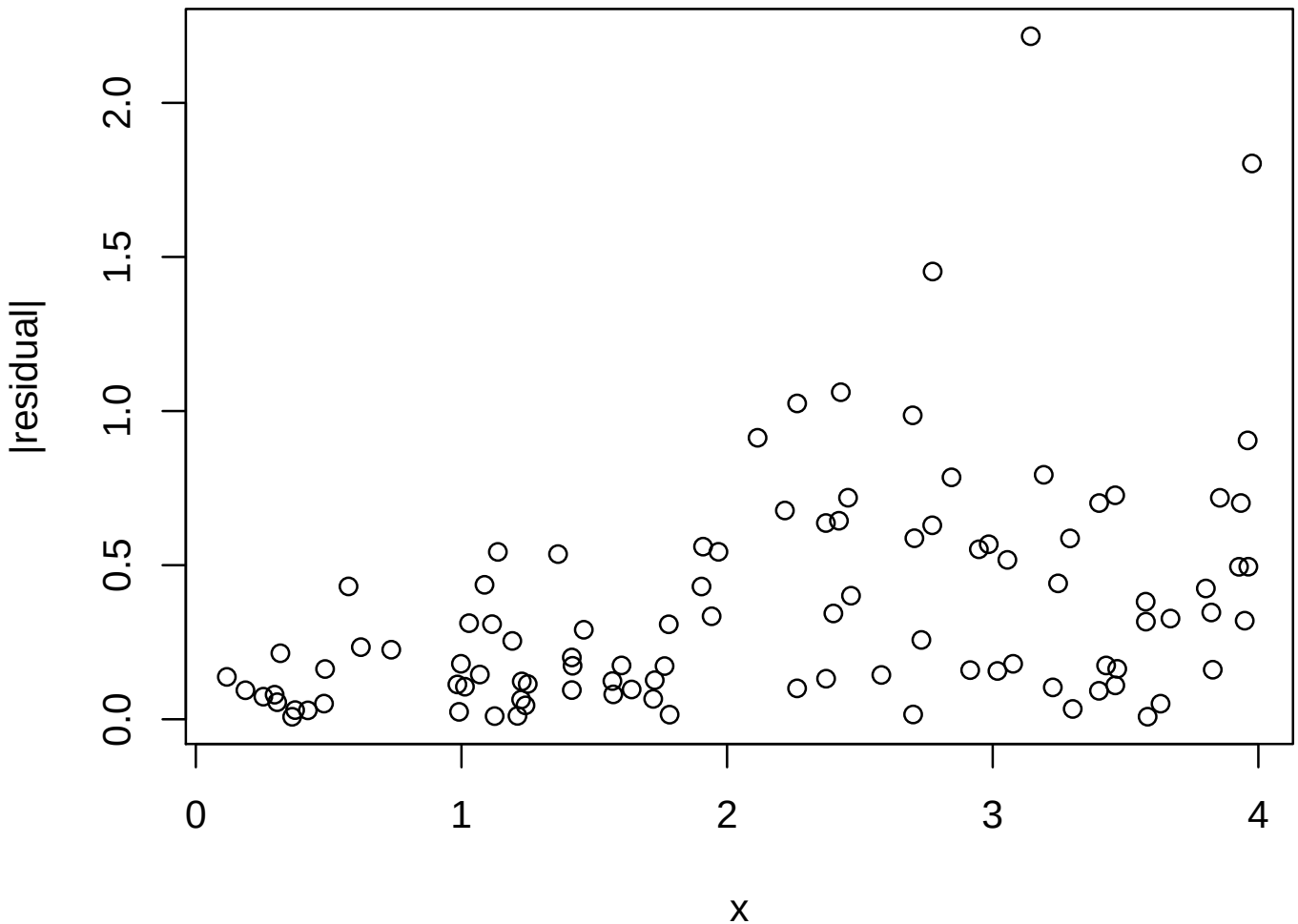
```
n=500
eps = randn(n)
x = 10*rand(n)
y = x + eps
model = sm.OLS(y,x).fit()
plt.scatter(x,np.abs(model.resid))
```

Checking for normal errors

This code generates absolute residual plots for **non-constant** variance:

```
n=500
x = 10*rand(n)
eps = x*randn(n) # variance of noise depends on x
y = x + eps
model = sm.OLS(y,x).fit()
plt.scatter(x,np.abs(model.resid))
```

Checking for non-constant variance



poll: (yes) variance is constant (no) variance is not constant

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)

Model selection

which features should appear in your model? two regimes

small data: (this class)

- use domain knowledge to decide features
- drop features with very small p values

big data: (ORIE 4741)

- use cross-validation to select best model
- use held-out test set to assess model performance

Model selection and p values

- if you fit **very few** models, and assumptions hold, then p values are reliable
- p values are **not** reliable if you fit many models or select from many features