13) Residual Analysis: Heteroscedastic

Vitor Kamada

July 2018

Reference

Tables, Graphics, and Figures from

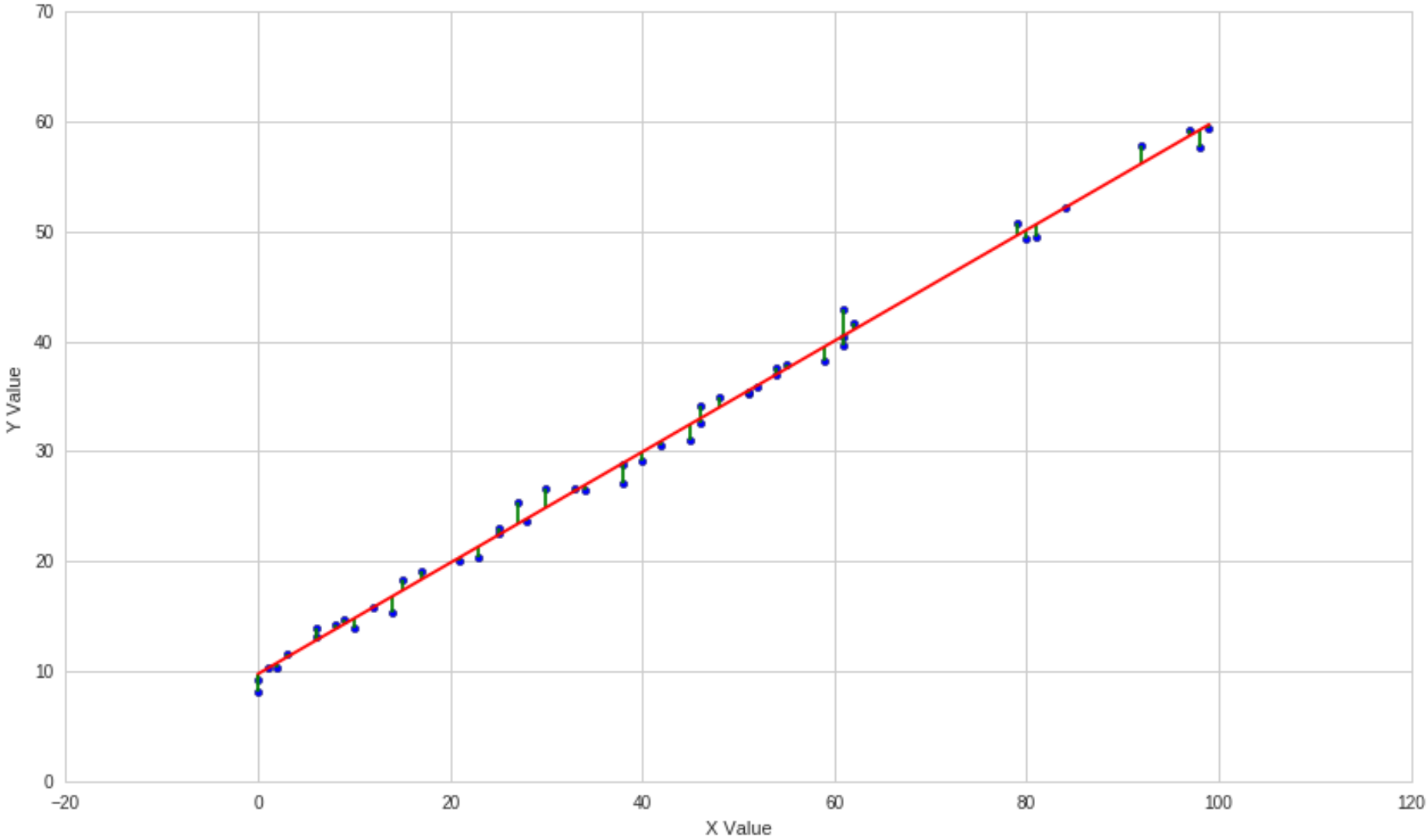
https://www.quantopian.com/lectures

Lecture 18 Residual Analysis

Import Libraries

```
import numpy as np
import pandas as pd
from statsmodels import regression
import statsmodels.api as sm
import statsmodels.stats.diagnostic as smd
import scipy.stats as stats
import matplotlib.pyplot as plt
import math
```

Toy Example



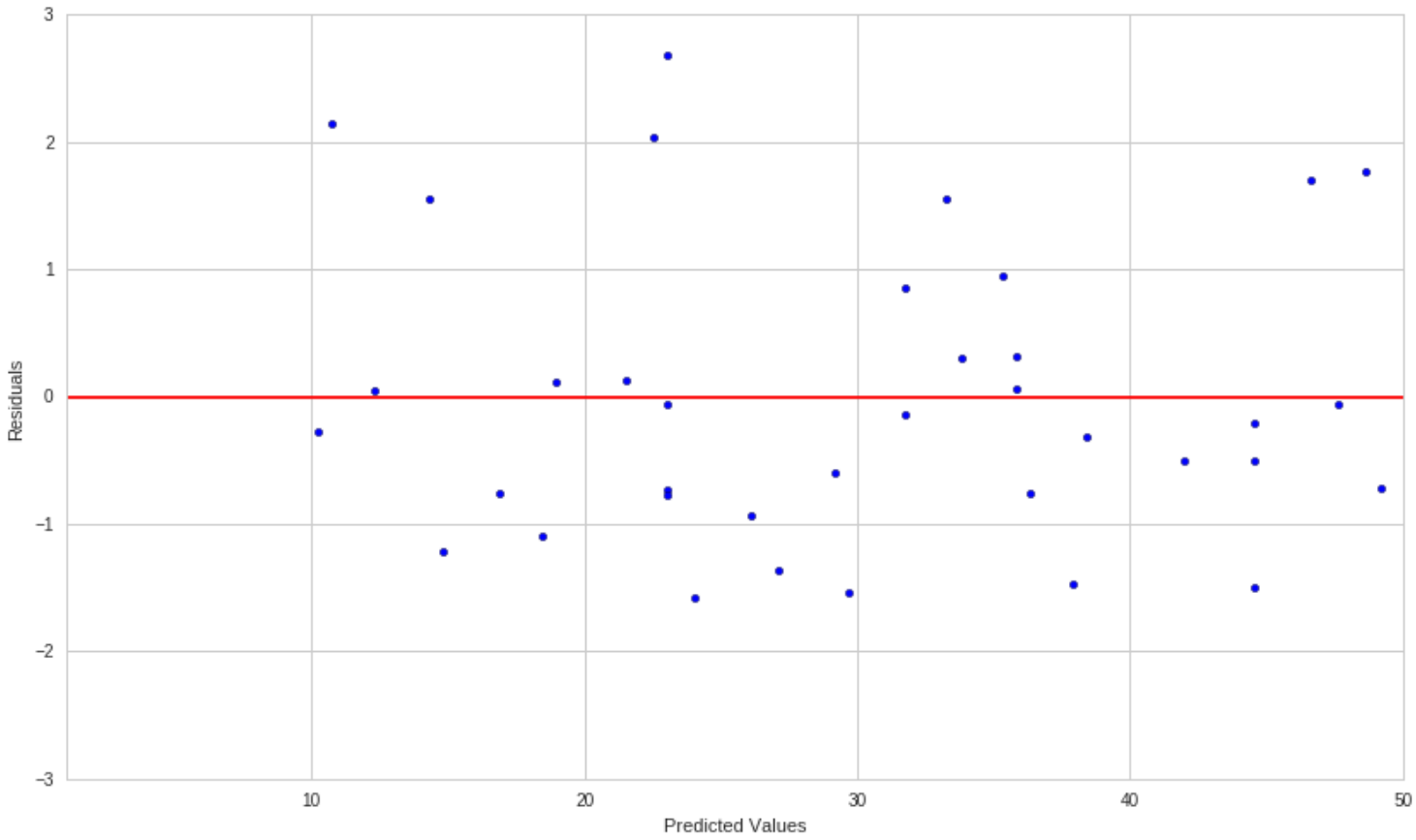
$r_i = Y_i - \hat{Y}_i$

```
residuals = model.resid
print residuals
```

```
[ 0.12178854 -1.11575232  0.77519042  0.57768863  2.03835741  0.06656651
  2.15102068  0.94456828 -1.47151503  1.55866833 -0.72661068 -0.5091077
  0.13172654 -0.07051635 -0.9349856   0.63238898 -1.58140307 -0.77609354
 -1.0985825   0.29932538  0.05236783  1.70668648 -0.97968704 -0.27426657
  0.5665373  -0.05289547 -1.21705277 -1.50327316 -1.36567936  1.55966862
 -2.10670533  0.70301288 -0.317796   -0.13759102  0.65568126 -0.20368754
 -1.53647546 -0.59926817 -0.50020968  1.76796167  0.97258904 -0.75143242
 -0.05352346  0.84946056 -0.76227328 -0.71173348  0.31988648  0.73084658
 -0.50147865  2.67760725]
```

Diagnosing Residuals

```
plt.scatter(model.predict(), residuals);
plt.axhline(0, color='red')
plt.xlabel('Predicted Values');
plt.ylabel('Residuals');
plt.xlim([1,50]);
```



$Y = \beta_0 + \beta_1 X^2 + \epsilon$

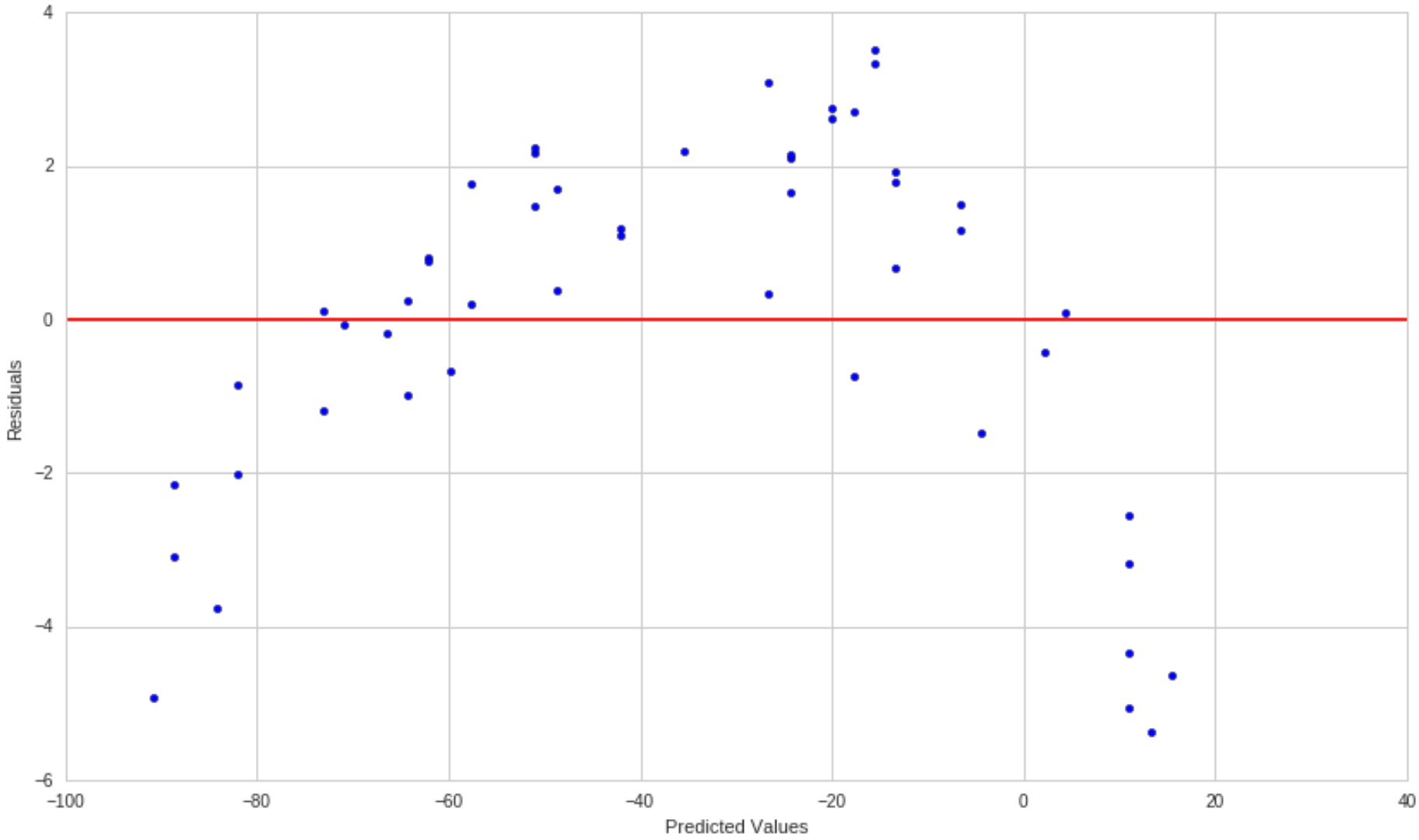
```
n = 50
X = np.random.randint(0, 50, n)
epsilon = np.random.normal(0, 1, n)
Y_nonlinear = 10 - X**1.2 + epsilon

model = sm.OLS(Y_nonlinear, sm.add_constant(X)).fit()
B0, B1 = model.params
residuals = model.resid

print 'beta_0: ', B0
print 'beta_1: ', B1
plt.scatter(model.predict(), residuals);
plt.axhline(0, color='red')
plt.xlabel('Predicted Values');
plt.ylabel('Residuals');
```

```
beta_0:  15.5066921239
beta_1:  -2.21573111573
```

Inverted-U Shape



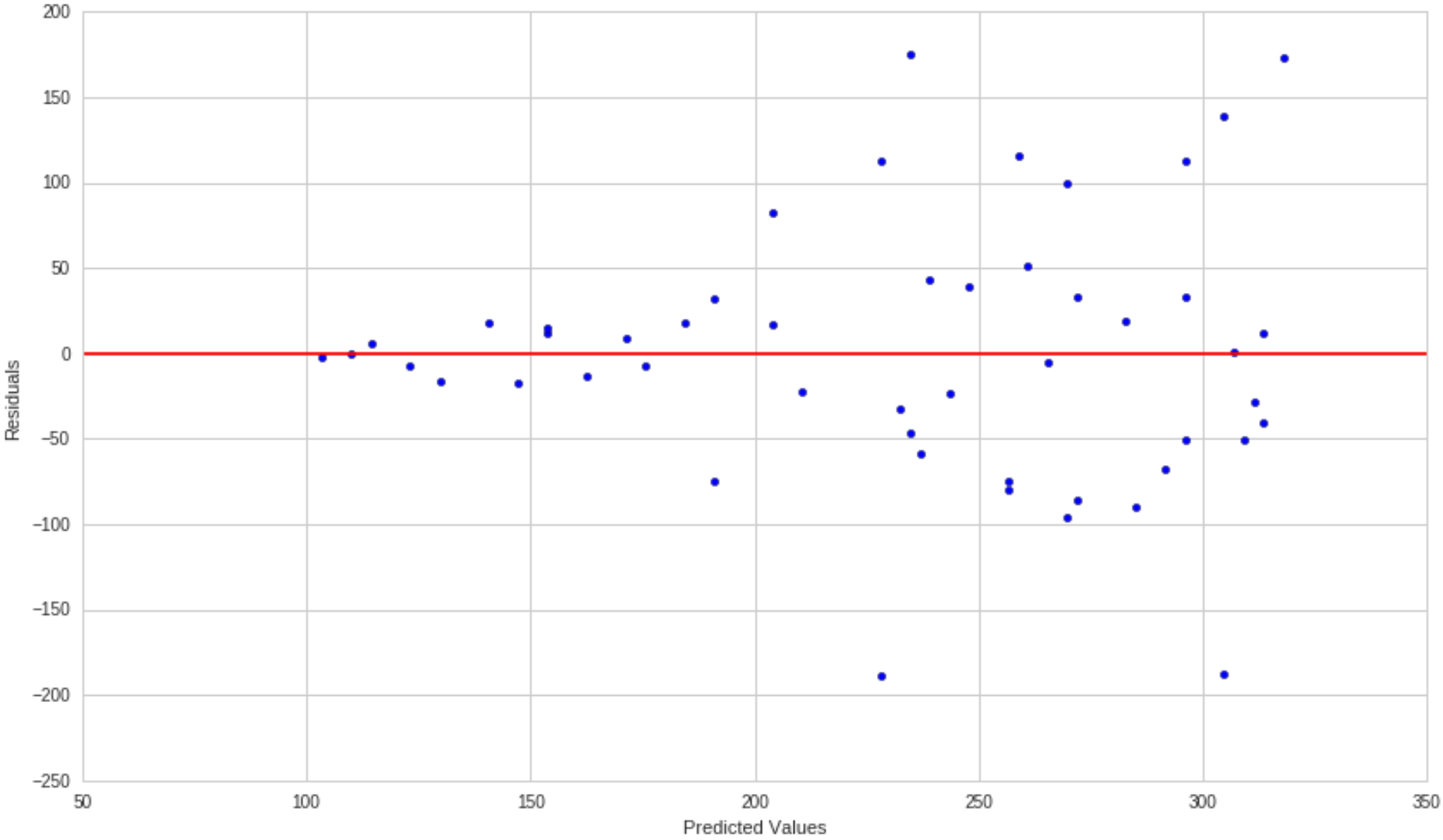
Heteroscedasticity

```
n = 50
X = np.random.randint(0, 100, n)
epsilon = np.random.normal(0, 1, n)
Y_heteroscedastic = 100 + 2*X + epsilon*X

model = sm.OLS(Y_heteroscedastic,
               sm.add_constant(X)).fit()
B0, B1 = model.params
residuals = model.resid

plt.scatter(model.predict(), residuals);
plt.axhline(0, color='red')
plt.xlabel('Predicted Values');
plt.ylabel('Residuals');
```

Tapered Cloud in One Direction



Testing for Heteroskedasticity

$$y = \beta_0 + \beta_1 x_1 + ... + \beta_k x_k + u$$

$$H_0 : Var(u|x_1, ..., x_k) = E(u^2) = \sigma^2$$

$$u^2 = \delta_0 + \delta_1 x_1 + ... + \delta_k x_k + v$$

$$F = \frac{R^2_{\hat{u}^2}/k}{(1-R^2_{\hat{u}^2})/(n-k-1)}$$

$$LM = nR^2_{\hat{u}^2} \sim \chi^2_k$$

Breusch-Pagan Test

```
breusch_pagan_p = smd.het_breushpagan(model.resid,
                                      model.model.exog)[1]
print breusch_pagan_p
if breusch_pagan_p > 0.05:
    print "The relationship is not heteroscedastic."
if breusch_pagan_p < 0.05:
    print "The relationship is heteroscedastic."
```

```
0.0152908639858
The relationship is heteroscedastic.
```

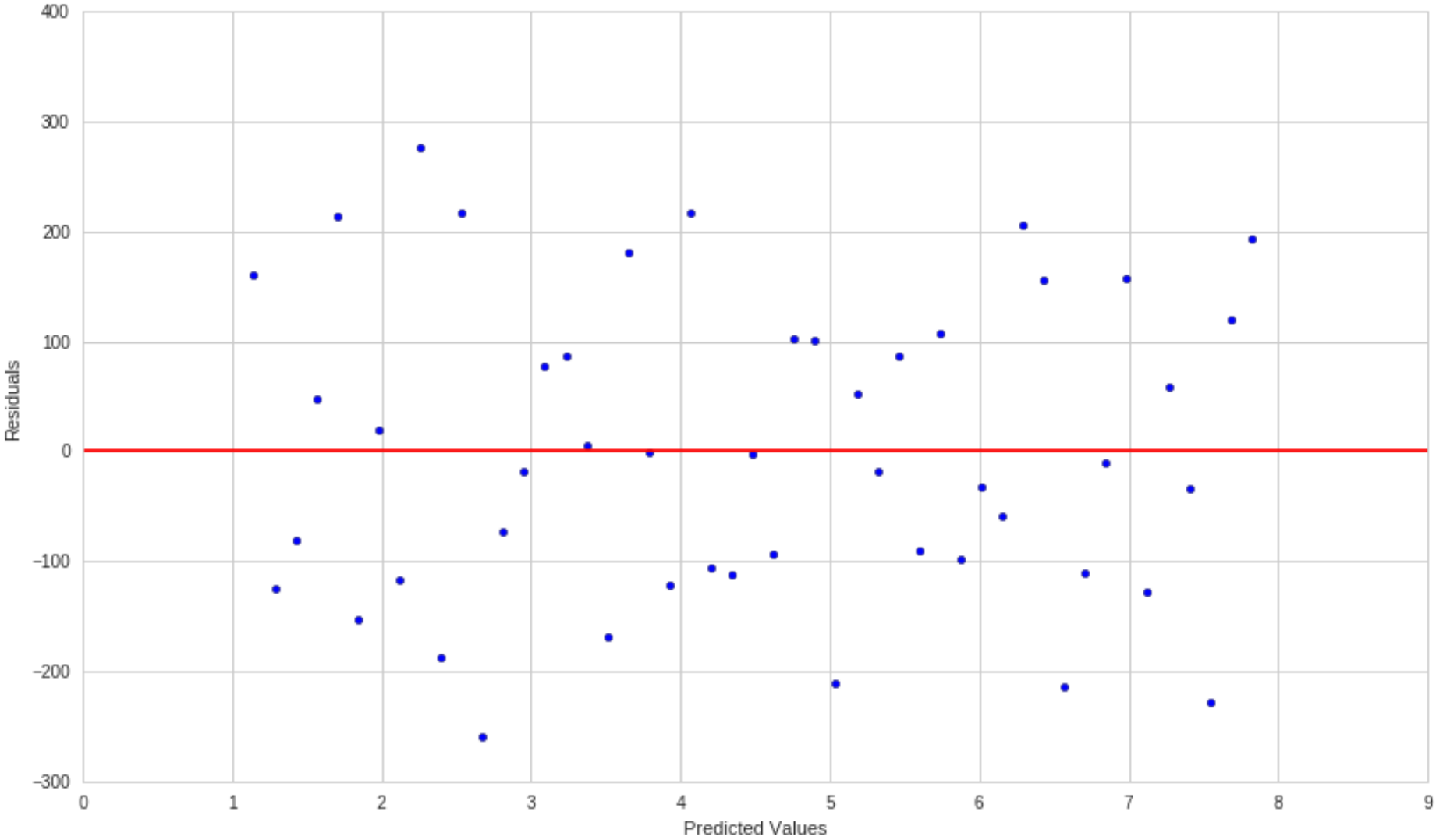
Differences Analysis

```
Y_heteroscedastic_diff = np.diff(Y_heteroscedastic)
```

```
model = sm.OLS(Y_heteroscedastic_diff,
               sm.add_constant(X[1:])).fit()
B0, B1 = model.params
residuals = model.resid

plt.scatter(model.predict(), residuals);
plt.axhline(0, color='red')
plt.xlabel('Predicted Values');
plt.ylabel('Residuals');
```

No Pattern


400
300
200
100
0
−100
−200
−300
Residuals
0
1
2
3
4
5
6
7
8
9
Predicted Values

Heteroscedastic Test

```
breusch_pagan_p = smd.het_breushpagan(residuals,
                                      model.model.exog)[1]
print breusch_pagan_p
if breusch_pagan_p > 0.05:
    print "The relationship is not heteroscedastic."
if breusch_pagan_p < 0.05:
    print "The relationship is heteroscedastic."
```

```
0.503010135414
The relationship is not heteroscedastic.
```

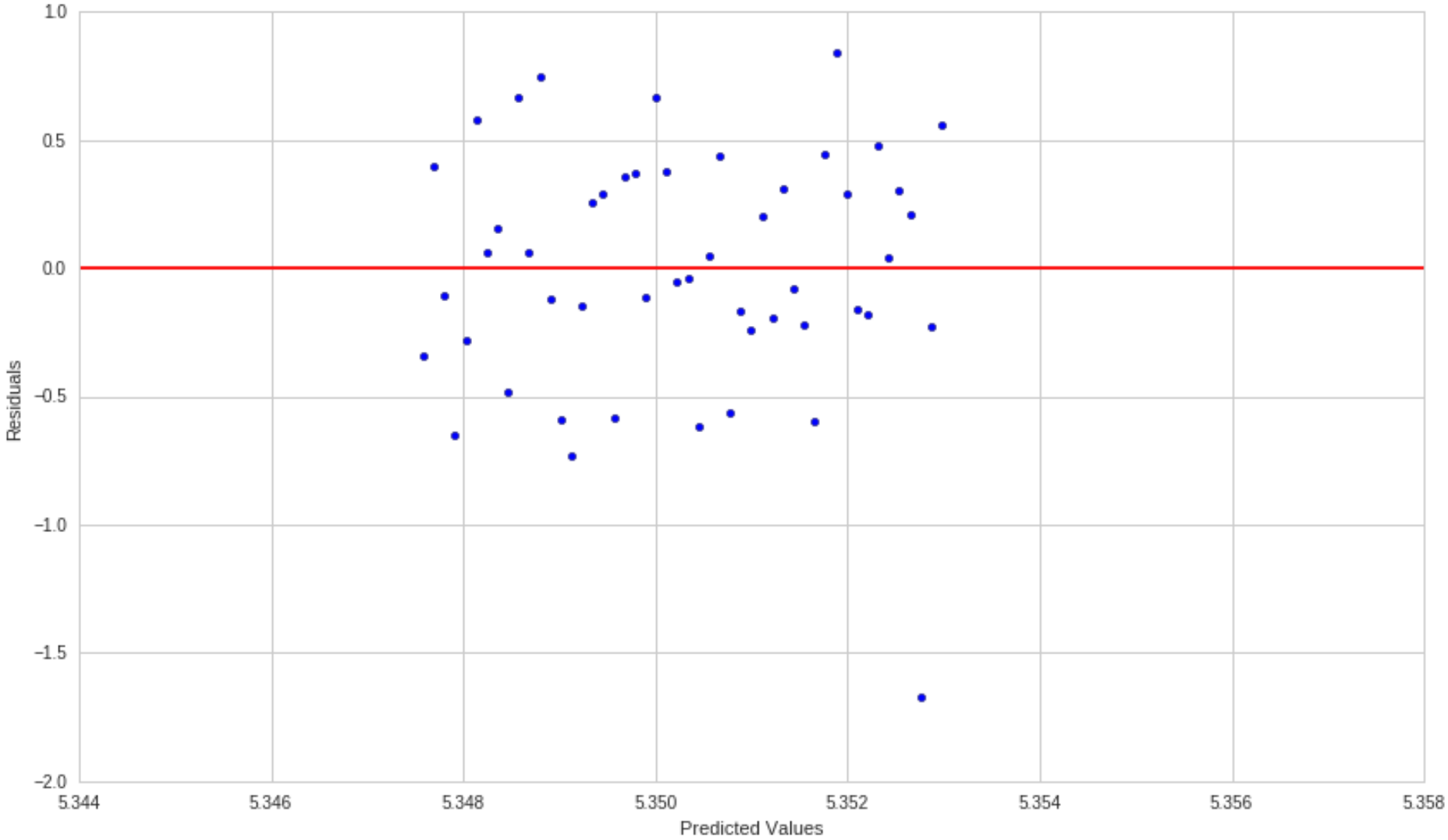
Logarithmic Transformation

```
Y_heteroscedastic_log = np.log(Y_heteroscedastic)
```

```
model = sm.OLS(Y_heteroscedastic_log,
               sm.add_constant(X)).fit()
B0, B1 = model.params
residuals = model.resid

plt.scatter(model.predict(), residuals);
plt.axhline(0, color='red')
plt.xlabel('Predicted Values');
plt.ylabel('Residuals');
```

No Pattern


1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
5.344
5.346
5.348
5.350
5.352
5.354
5.356
5.358
Predicted Values

Heteroscedastic Test

```
breusch_pagan_p = smd.het_breushpagan(residuals,
                                      model.model.exog)[1]
print breusch_pagan_p
if breusch_pagan_p > 0.05:
    print "The relationship is not heteroscedastic."
if breusch_pagan_p < 0.05:
    print "The relationship is heteroscedastic."
```

```
0.277597815077
The relationship is not heteroscedastic.
```

Box-Cox Transformation

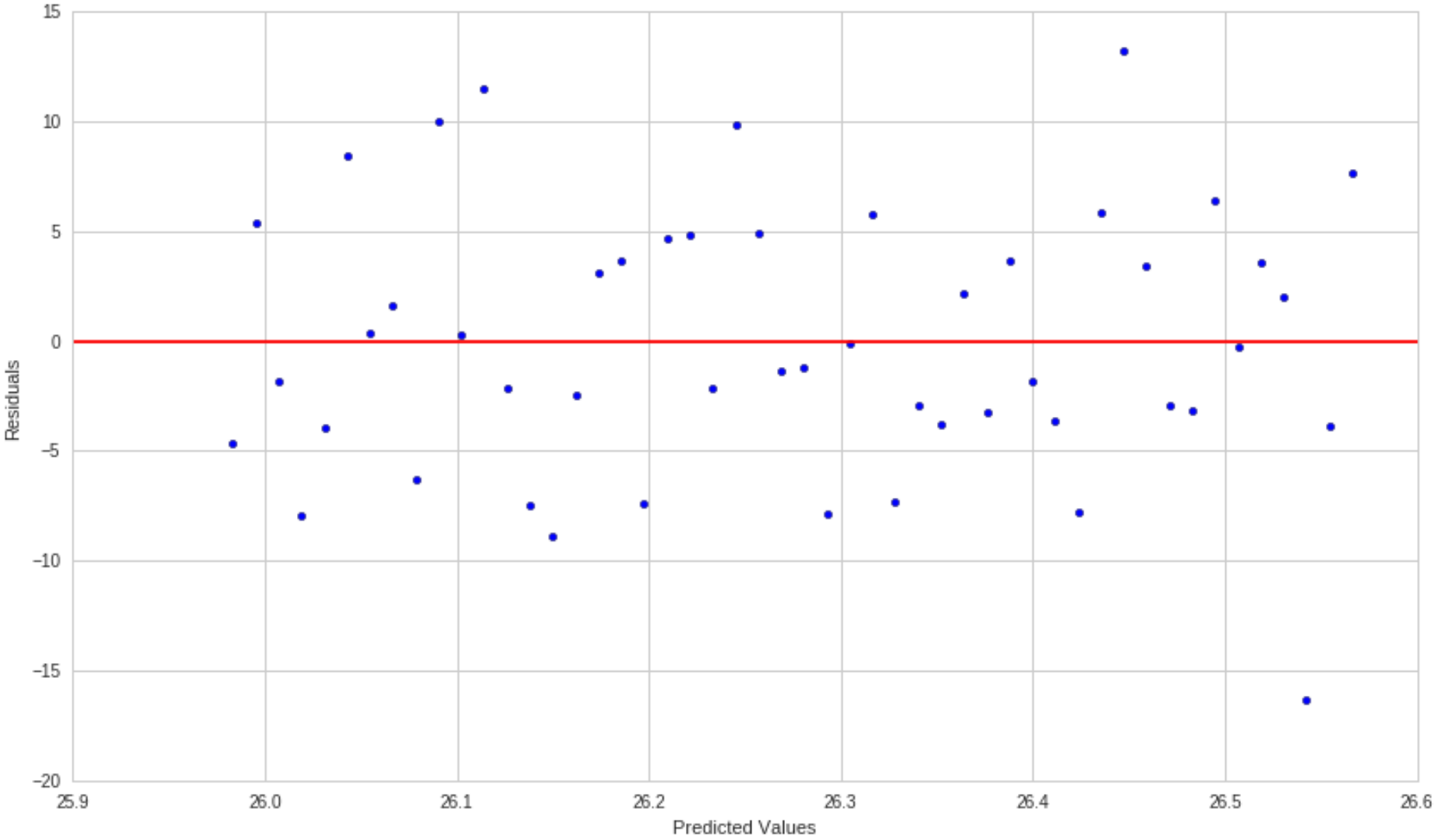
$$Y^{(\lambda)} = \begin{cases} \frac{Y^{\lambda}-1}{\lambda} & : \lambda \neq 0 \\ logY & : \lambda = 0 \end{cases}$$

```
Y_heteroscedastic_box_cox = stats.boxcox(Y_heteroscedastic)[0]
```

```
model = sm.OLS(Y_heteroscedastic_box_cox, sm.add_constant(X)).fit()
B0, B1 = model.params
residuals = model.resid

plt.scatter(model.predict(), residuals);
plt.axhline(0, color='red')
plt.xlabel('Predicted Values');
plt.ylabel('Residuals');
```

No Pattern


15
10
5
0
−5
−10
−15
−20
Residuals
25.9
26.0
26.1
26.2
26.3
26.4
26.5
26.6
Predicted Values

Heteroscedastic Test

```
breusch_pagan_p = smd.het_breushpagan(residuals,
                                      model.model.exog)[1]
print breusch_pagan_p
if breusch_pagan_p > 0.05:
    print "The relationship is not heteroscedastic."
if breusch_pagan_p < 0.05:
    print "The relationship is heteroscedastic."
```

```
0.534411131103
The relationship is not heteroscedastic.
```