7.9

March 12, 2018

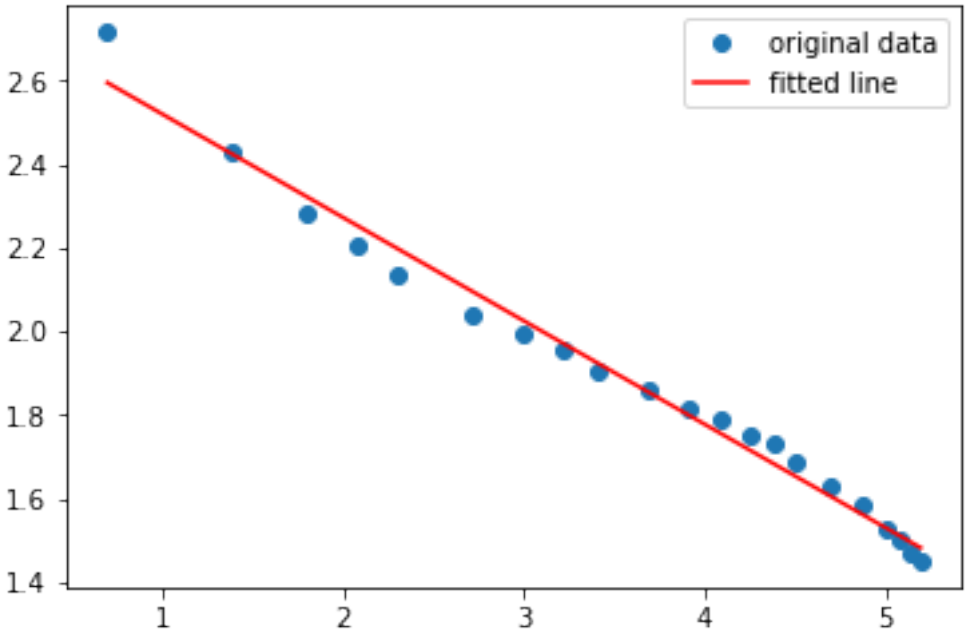
1 7.9.

At http://www.statsci.org/data/general/brunhild.html, you will find a dataset that measures the concentration of a sulfate in the blood of a baboon named Brunhilda as a function of time. Build a linear regression of the log of the concentration against the log of time.

- Prepare a plot showing (a) the data points and (b) the regression line in log-log coordinates.
- Prepare a plot showing (a) the data points and (b) the regression curve in the original coor-dinates.
- Plot the residual against the fitted values in log-log and in original coor-dinates.
- Use your plots to explain whether your regression is good or bad and why.

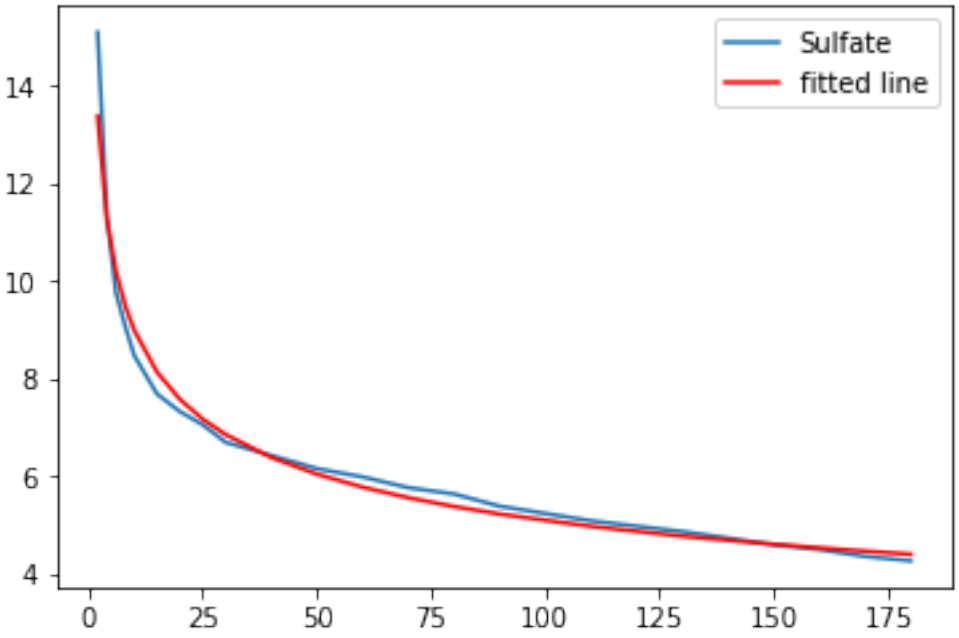
In [1]:
```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pylab as plt
# import statsmodels.api as sm
from scipy.stats import linregress

%matplotlib inline

df = pd.read_table('brunhild.txt')
x_log = np.log(df.Hours)
y_log = np.log(df.Sulfate)
```

In [2]:
```
# ref: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.linregress.htm
slope, intercept, r_value, p_value, std_err = linregress(x_log, y_log)
plt.plot(x_log, y_log, 'o', label='original data')
plt.plot(x_log, intercept + slope*x_log, 'r', label='fitted line')
plt.legend()
plt.show()
```

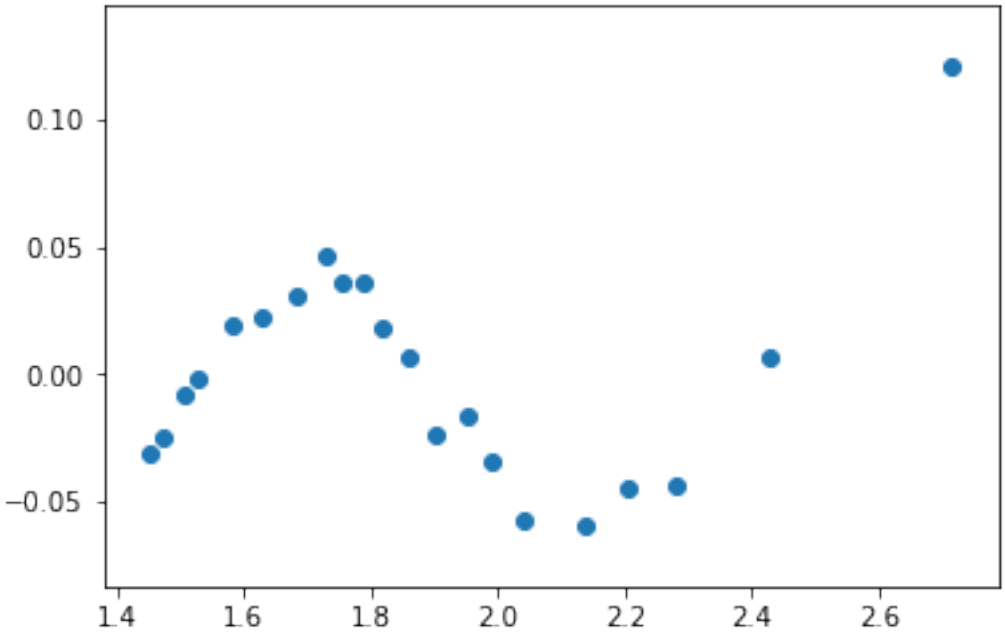


In [3]:
```
y_log_predicted = slope * x_log + intercept
y_log_residual = y_log - y_log_predicted
```

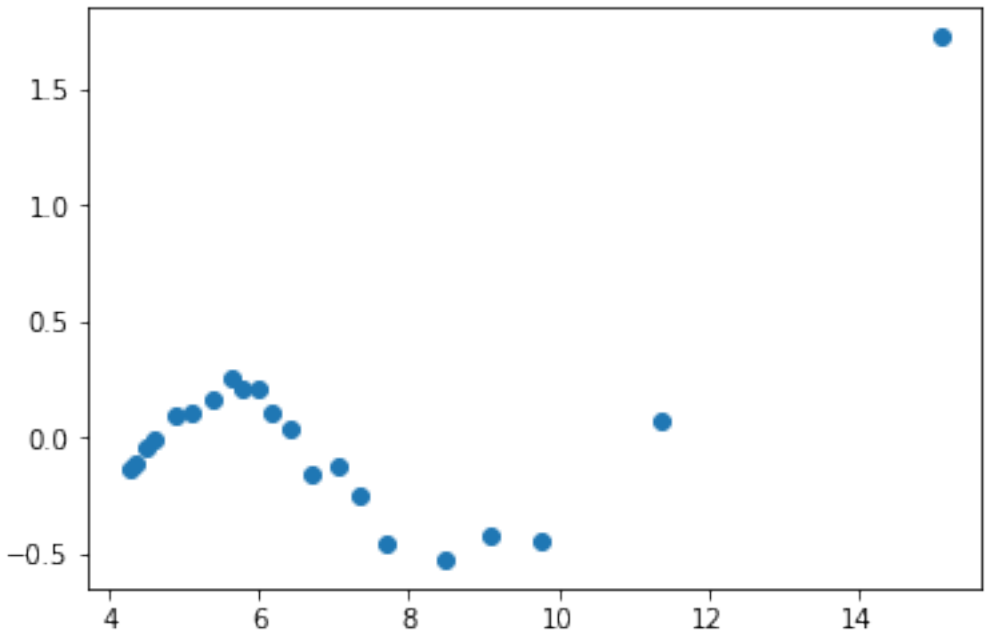
In [4]:
```
x = df.Hours
y = df.Sulfate

y_predicted = np.exp(y_log_predicted)

plt.plot(x, y)

plt.plot(x, y_predicted, 'r', label='fitted line')
plt.legend()
plt.show()
```



In [5]:
```
y_residual = y - y_predicted

plt.scatter(y_log, y_log_residual)
plt.show()
```



```
In [6]: plt.scatter(y, y_residual)
        plt.show()
```



A log-log regression seems to be a good fit for this data. The residual plot averages that 0 and except one outlier data point, it follows the data closely.