Jacob Miller - Homework 5

March 31, 2020

0.1 Setup

[1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import patsy
import statsmodels.api as sm
from itertools import combinations, compress
```

[2]:
```python
def title_print(text):
    '''
    Used throughout to print section titles
    '''
    print()
    print('#' * (len(text) + 4))
    print('|', text, '|')
    print('#' * (len(text) + 4))
```

0.2 Problem 5.1

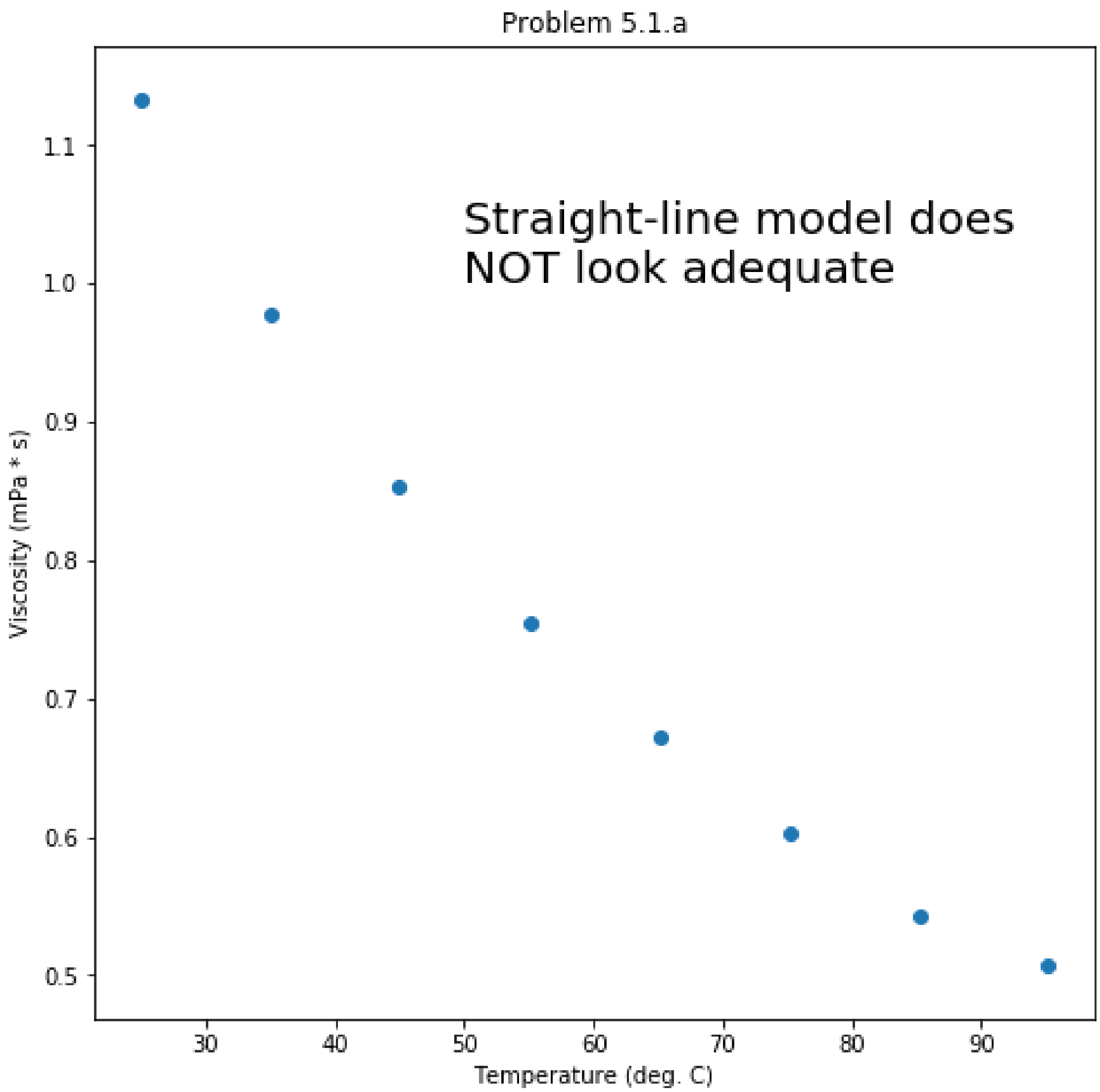
[3]:
```python
df = pd.DataFrame(data = {'Temperature' : [24.9, 35.0, 44.9, 55.1,
                                           65.2, 75.2, 85.2, 95.2],
                          'Viscosity' : [1.133, 0.9772, 0.8532, 0.7550,
                                         0.6723, 0.6021, 0.5420, 0.5074]})
```

0.2.1 Problem 5.1.a

[6]:
```python
title_print('Problem 5.1.a')
fig = plt.figure(figsize = (8, 8))
plt.scatter(df['Temperature'], df['Viscosity'])
plt.xlabel('Temperature (deg. C)')
plt.ylabel('Viscosity (mPa * s)')
plt.title('Problem 5.1.a')
plt.text(50, 1, 'Straight-line model does\nNOT look adequate',
         fontdict = {'fontsize': 20})
```

```
plt.show()
```

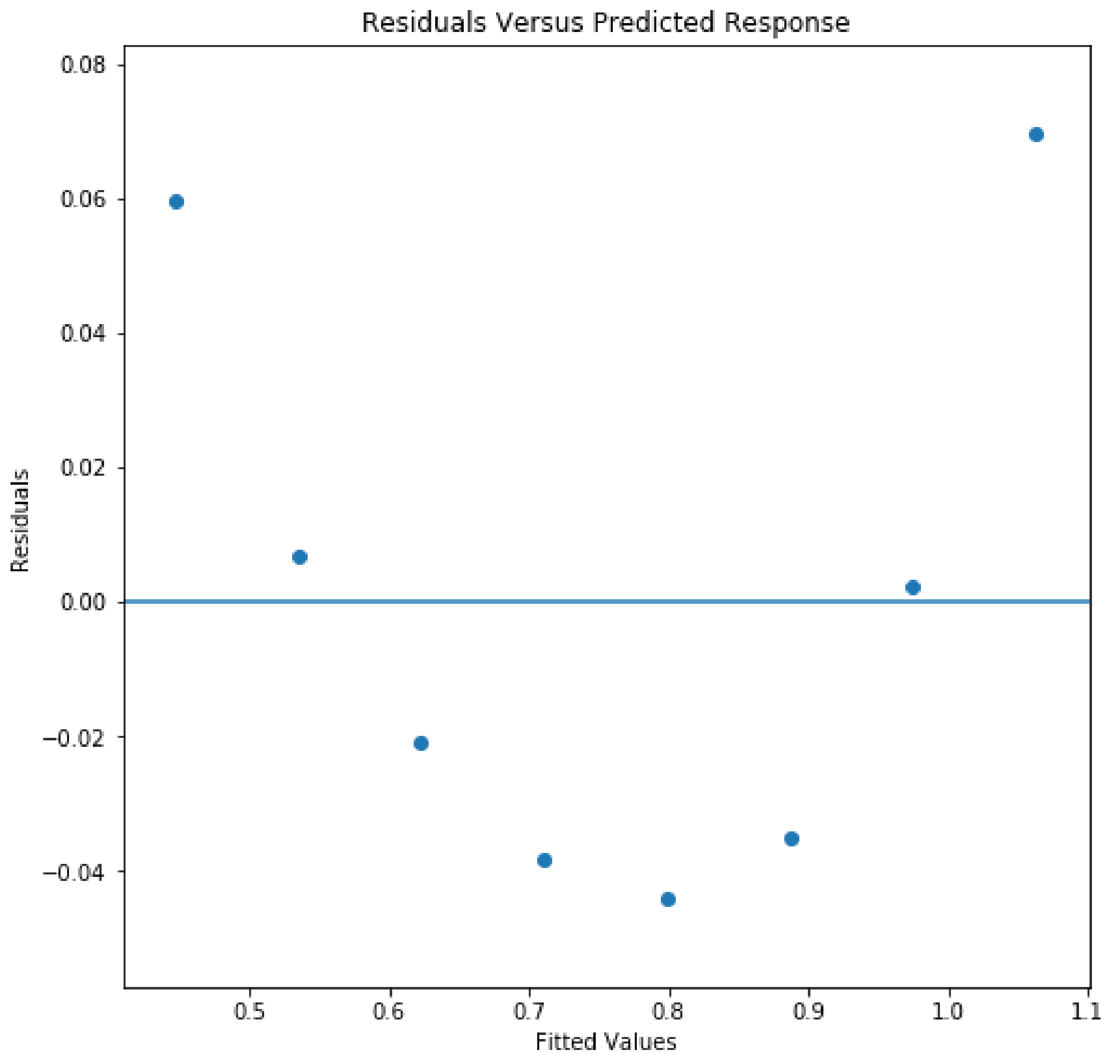
```
#################
| Problem 5.1.a |
#################
```



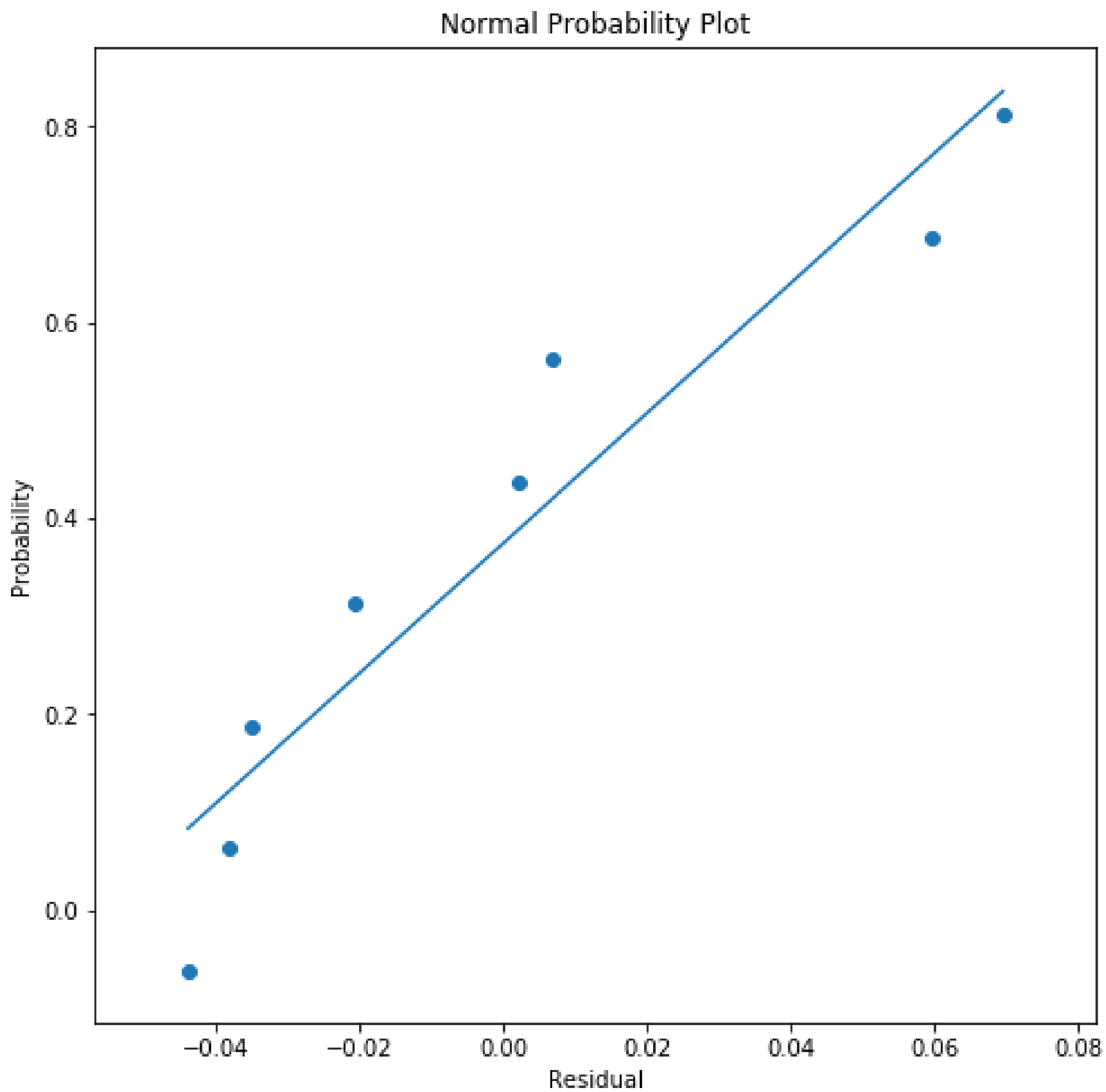

```
#################
| Problem 5.1.b |
#################
                            OLS Regression Results
==============================================================================
```

```
Dep. Variable:              Viscosity   R-squared:                       0.960
Model:                            OLS   Adj. R-squared:                  0.954
Method:                 Least Squares   F-statistic:                     144.6
Date:                Tue, 31 Mar 2020   Prob (F-statistic):           2.01e-05
Time:                        15:02:37   Log-Likelihood:                 14.187
No. Observations:                   8   AIC:                            -24.37
Df Residuals:                       6   BIC:                            -24.21
Df Model:                           1
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
Intercept       1.2815      0.047     27.343      0.000       1.167       1.396
Temperature    -0.0088      0.001    -12.024      0.000      -0.011      -0.007
==============================================================================
Omnibus:                        1.431   Durbin-Watson:                   0.734
Prob(Omnibus):                  0.489   Jarque-Bera (JB):                0.942
Skew:                           0.642   Prob(JB):                        0.624
Kurtosis:                       1.915   Cond. No.                         180.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

```
/Users/Jake/anaconda3/envs/Data/lib/python3.7/site-
packages/scipy/stats/stats.py:1535: UserWarning: kurtosistest only valid for
n>=20 … continuing anyway, n=8
  "anyway, n=%i" % int(n))
```



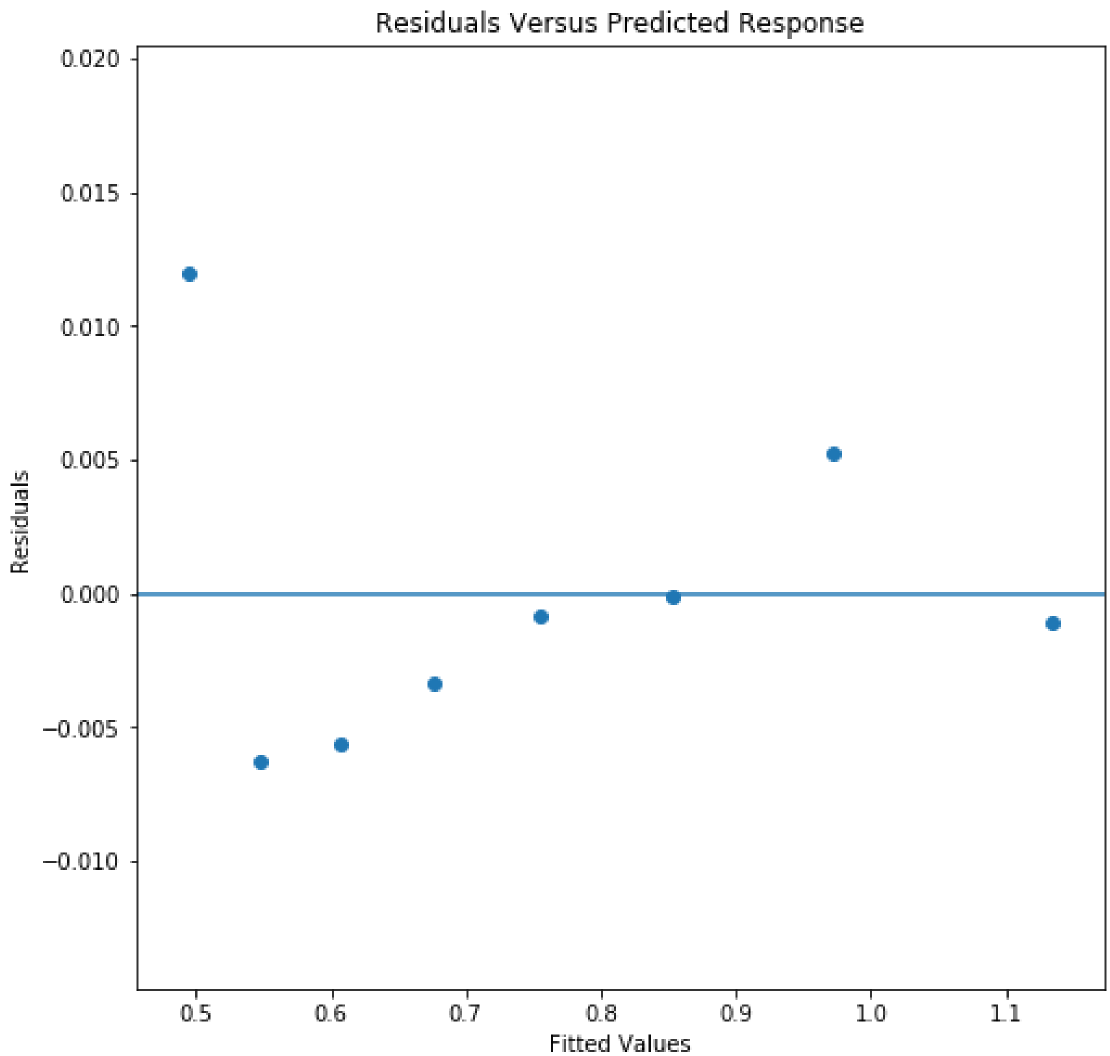
Residuals Versus Predicted Response
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
Residuals
0.5
0.6
0.7
0.8
0.9
1.0
1.1
Fitted Values

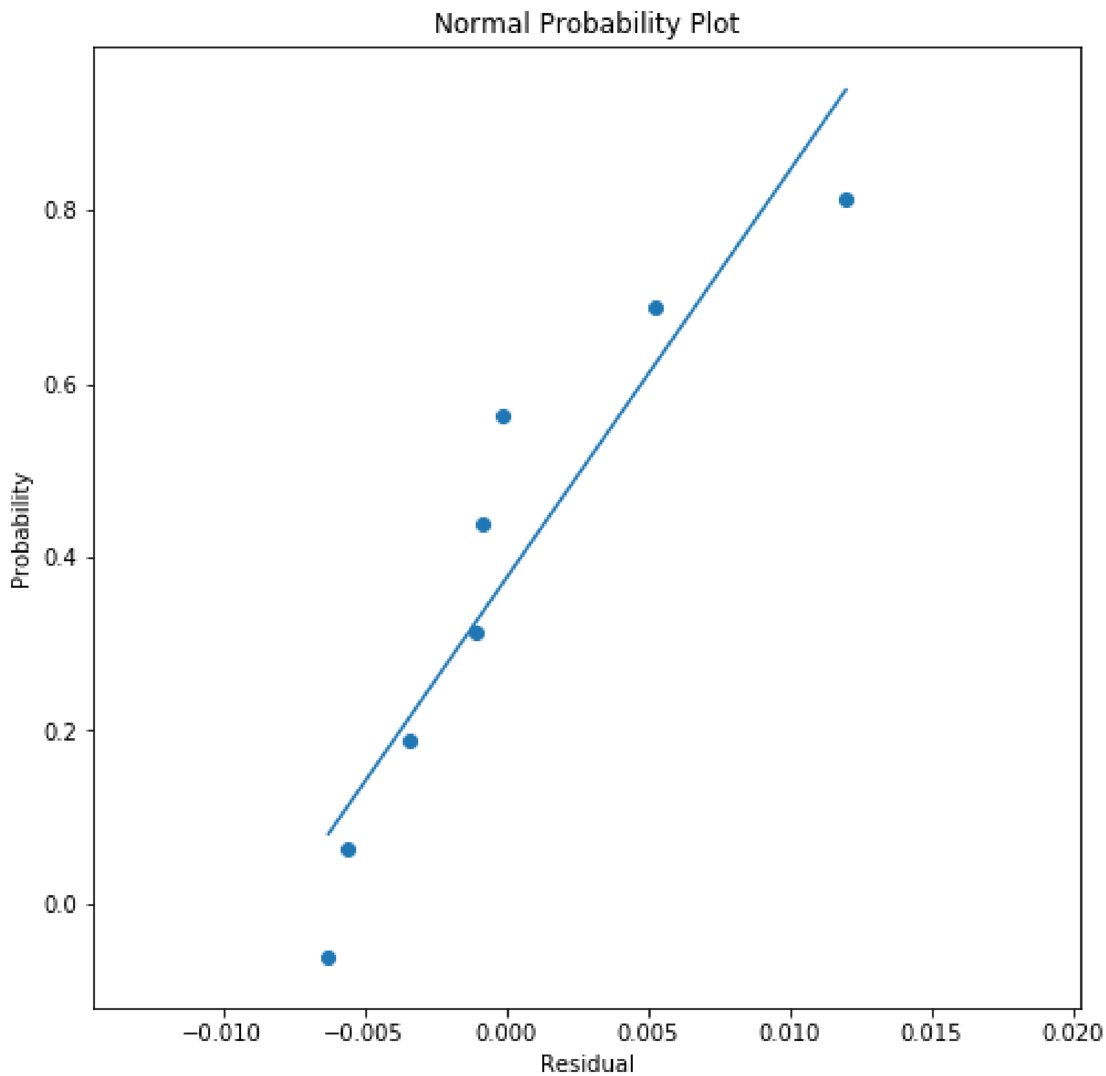



```
--> Clear non-linearity in residual plot <--
--> Normality appears to have problems <--

#################
| Problem 5.1.c |
#################
                            OLS Regression Results
==============================================================================
Dep. Variable:              Viscosity   R-squared:                       0.999
Model:                            OLS   Adj. R-squared:                  0.999
Method:                 Least Squares   F-statistic:                     7952.
Date:                Tue, 31 Mar 2020   Prob (F-statistic):           1.34e-10
```

```
Time:                        15:02:37   Log-Likelihood:                 30.056
No. Observations:                   8   AIC:                            -56.11
Df Residuals:                       6   BIC:                            -55.95
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
=======
                          coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-------
Intercept               2.6651      0.022    123.721      0.000       2.612
2.718
np.log(Temperature)    -0.4762      0.005    -89.172      0.000      -0.489
-0.463
==============================================================================
Omnibus:                        3.567   Durbin-Watson:                   1.623
Prob(Omnibus):                  0.168   Jarque-Bera (JB):                1.267
Skew:                           0.975   Prob(JB):                        0.531
Kurtosis:                       2.954   Cond. No.                         40.0
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

```
/Users/Jake/anaconda3/envs/Data/lib/python3.7/site-
packages/scipy/stats/stats.py:1535: UserWarning: kurtosistest only valid for
n>=20 … continuing anyway, n=8
  "anyway, n=%i" % int(n))
```


Residuals Versus Predicted Response
0.020
0.015
0.010
0.005
0.000
−0.005
−0.010
Residuals
0.5
0.6
0.7
0.8
0.9
1.0
1.1
Fitted Values




```
--> Residual vs. Response plot mildly improved <--
--> R-squared value slightly increased <--
--> Overall improvement seems minimal <--
```

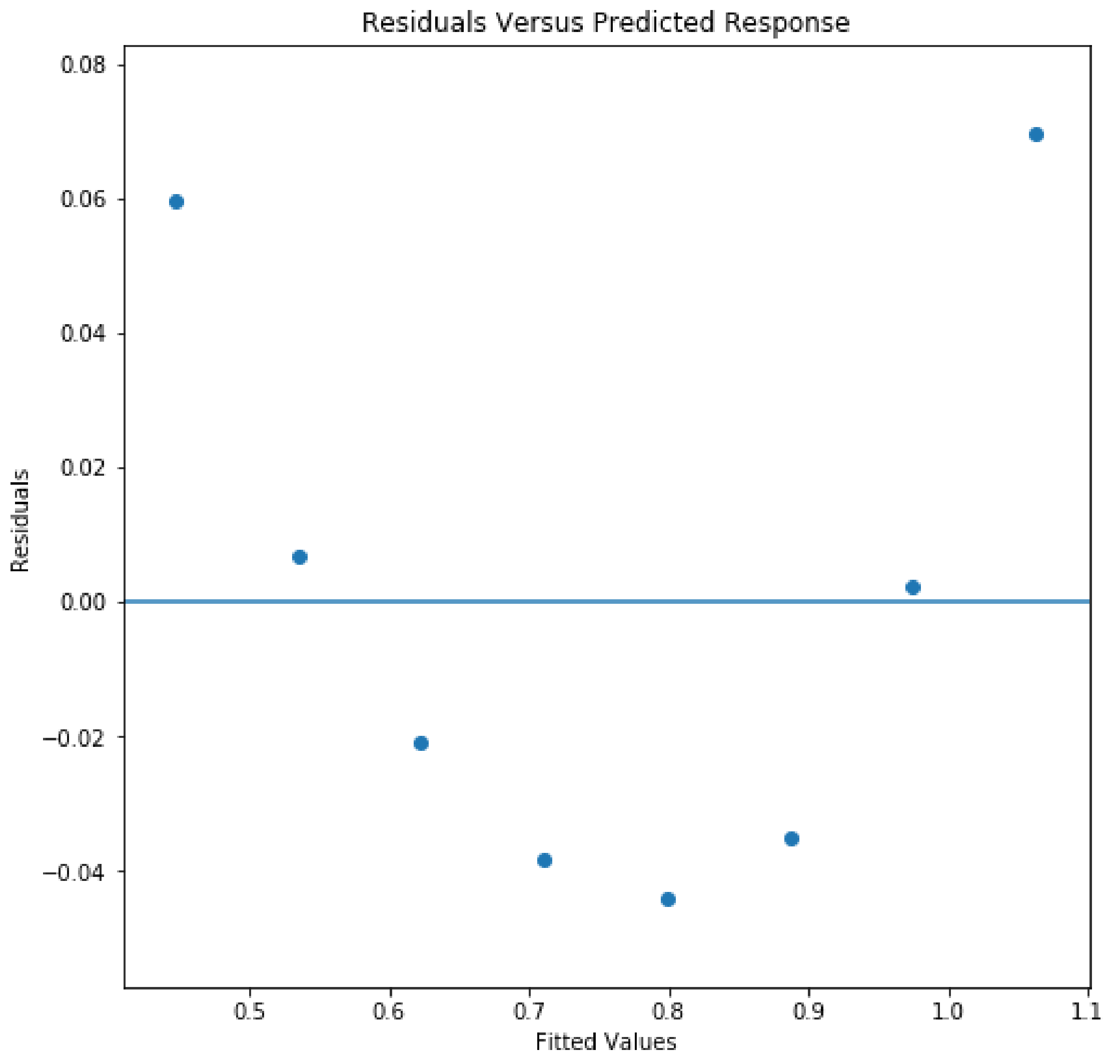
0.2.2 Problem 5.1.b

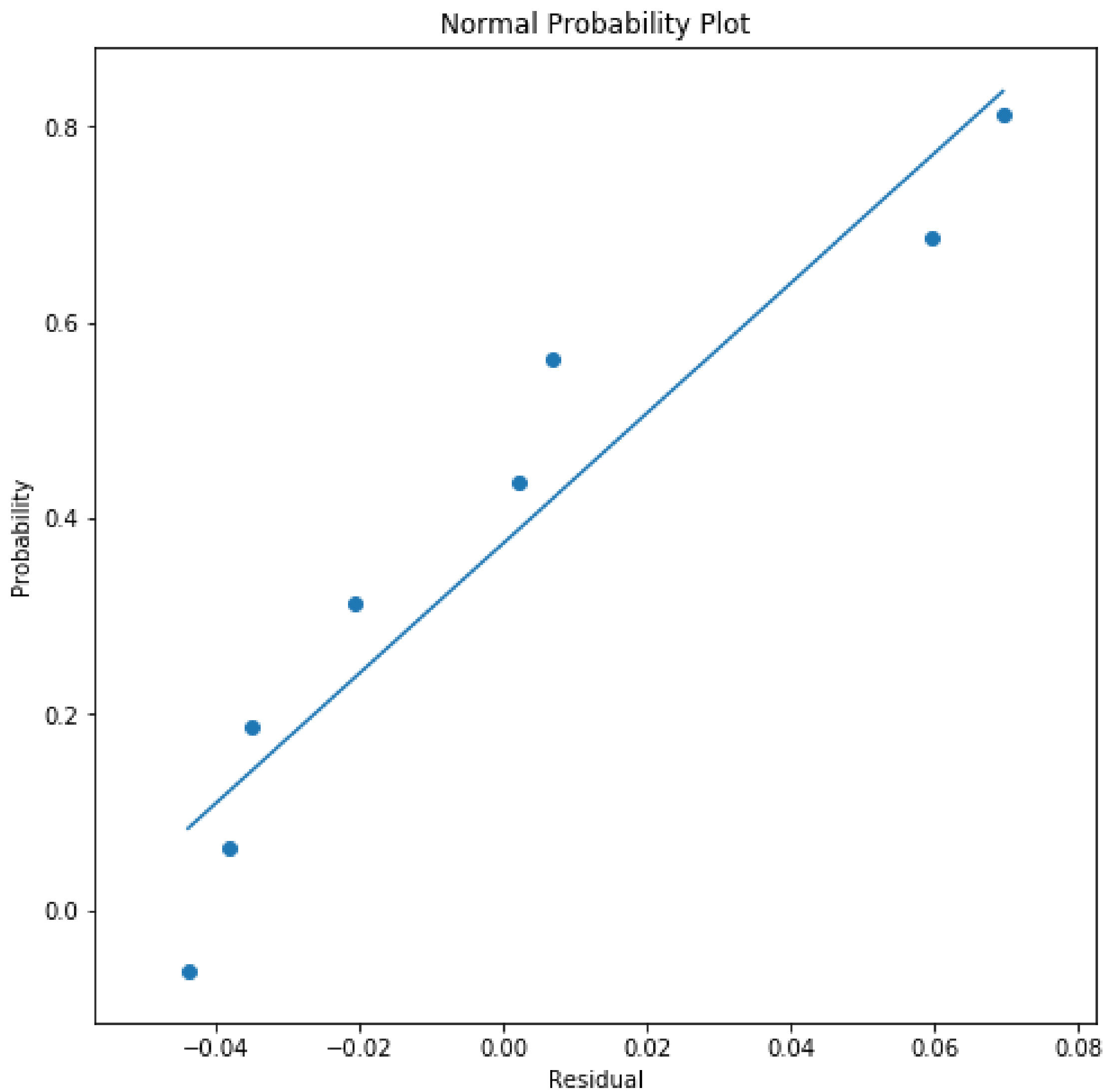
[7]:
```python
title_print('Problem 5.1.b')
y, X = patsy.dmatrices('Viscosity ~ Temperature', df)
model = sm.OLS(y, X)
results = model.fit()
results.model.data.design_info = X.design_info
print(results.summary())
```

```python
# Get residuals and probability for plot
residuals = results.resid
Prob = [(i - 1/2) / len(y) for i in range(len(y))]

# Plot residuals vs. fitted values
fig, ax = plt.subplots(figsize = (8, 8))
ax.scatter(results.fittedvalues, residuals)
ax.axhline(0)
ax.set_xlabel('Fitted Values')
ax.set_ylabel('Residuals')
plt.title('Residuals Versus Predicted Response')
plt.show()

# Calculate OLS using residuals to plot straight line. Get y values from model
resid_results = sm.OLS(Prob, sm.add_constant(sorted(residuals))).fit()
X_range = np.linspace(min(residuals), max(residuals), len(residuals))

# Normality plot
fig = plt.figure(figsize = (8, 8))
plt.scatter(sorted(residuals), Prob)
plt.plot(X_range, resid_results.params[0] + resid_results.params[1] * X_range)
plt.xlabel('Residual')
plt.ylabel('Probability')
plt.title('Normal Probability Plot')
plt.show()

print('\n--> Clear non-linearity in residual plot <--')
print('--> Normality appears to have problems <--')
```

```
#################
| Problem 5.1.b |
#################
                            OLS Regression Results
==============================================================================
Dep. Variable:              Viscosity   R-squared:                       0.960
Model:                            OLS   Adj. R-squared:                  0.954
Method:                 Least Squares   F-statistic:                     144.6
Date:                Tue, 31 Mar 2020   Prob (F-statistic):           2.01e-05
Time:                        15:03:43   Log-Likelihood:                 14.187
No. Observations:                   8   AIC:                            -24.37
Df Residuals:                       6   BIC:                            -24.21
Df Model:                           1
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
```

```
-------------------------------------------------------------------------------
Intercept       1.2815      0.047     27.343      0.000       1.167       1.396
Temperature    -0.0088      0.001    -12.024      0.000      -0.011      -0.007
==============================================================================
Omnibus:                        1.431   Durbin-Watson:                   0.734
Prob(Omnibus):                  0.489   Jarque-Bera (JB):                0.942
Skew:                           0.642   Prob(JB):                        0.624
Kurtosis:                       1.915   Cond. No.                         180.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.

/Users/Jake/anaconda3/envs/Data/lib/python3.7/site-
packages/scipy/stats/stats.py:1535: UserWarning: kurtosistest only valid for
n>=20 … continuing anyway, n=8
  "anyway, n=%i" % int(n))
```


Residuals Versus Predicted Response
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
Residuals
0.5
0.6
0.7
0.8
0.9
1.0
1.1
Fitted Values




```
--> Clear non-linearity in residual plot <--
--> Normality appears to have problems <--
```

0.2.3 Problem 5.1.c

[ ]:
```
title_print('Problem 5.1.c')
y, X = patsy.dmatrices('Viscosity ~ np.log(Temperature)', df)
model = sm.OLS(y, X)
results = model.fit()
results.model.data.design_info = X.design_info
print(results.summary())
```

```python
# Get residuals and probability for plot
residuals = results.resid
Prob = [(i - 1/2) / len(y) for i in range(len(y))]

# Plot residuals vs. fitted values
fig, ax = plt.subplots(figsize = (8, 8))
ax.scatter(results.fittedvalues, residuals)
ax.axhline(0)
ax.set_xlabel('Fitted Values')
ax.set_ylabel('Residuals')
plt.title('Residuals Versus Predicted Response')
plt.show()

# Calculate OLS using residuals to plot straight line. Get y values from model
resid_results = sm.OLS(Prob, sm.add_constant(sorted(residuals))).fit()
X_range = np.linspace(min(residuals), max(residuals), len(residuals))

# Normality plot
fig = plt.figure(figsize = (8, 8))
plt.scatter(sorted(residuals), Prob)
plt.plot(X_range, resid_results.params[0] + resid_results.params[1] * X_range)
plt.xlabel('Residual')
plt.ylabel('Probability')
plt.title('Normal Probability Plot')
plt.show()

print('\n--> Residual vs. Response plot mildly improved <--')
print('--> R-squared value slightly increased <--')
print('--> Overall improvement seems minimal <--')
```

0.3 Problem 6.1

Note that Python uses 0-based indexing, so add 1 to the points in the following dataframe

[14]:
```python
def run_analysis(drop_point = None):
    '''
    Parameters
    ----------
    drop_point : int or list-like, optional
        Point(s) to drop for analysis. The default is None.

    Returns
    -------
    Points to potentially drop, if drop_point == None.

```

```
    '''
    df = pd.read_excel('Data/data-table-b2.xlsx')
    df.columns = ['y', 'x1', 'x2', 'x3', 'x4', 'x5']

    # Drop influential points, if necessary
    if drop_point:
        df = df.drop(np.array(drop_point) - 1)
    y, X = patsy.dmatrices('y ~ x1 + x2 + x3 + x4 + x5', df)

    # Fit model, get influence statistics
    model = sm.OLS(y, X)
    results = model.fit()
    results.model.data.design_info = X.design_info

    # If dropping points, only run to here and then exit function
    if drop_point:
        print('\nPoints dropped: {}'.format(drop_point))
        print('Coefficients: {}'.format(np.round(results.params, 3)))
        print('R-squared: {}'.format(round(results.rsquared, 3)))
        return
    else:
        print('Coefficients: {}'.format(np.round(results.params, 3)))
        print('R-squared: {}\n'.format(round(results.rsquared, 3)))

    infl = results.get_influence()
    infl_df = infl.summary_frame()
    print(infl_df)

    # Hat diagonals
    n, p = df.shape
    lev_pt = 2 * p / n
    dhat_pts = list(infl_df[infl_df['hat_diag'] > lev_pt].index + 1)
    print('\n***| Hat Diagonal |***')
    print('Leverage calculation (2 * p \ n) = {}'.format(round(lev_pt, 3)))
    print('Points where hat diagonal exceeds leverage calculation: {}'.
          format(dhat_pts))

    # Cook's D
    print('\n***| Cook\'s D |***')
    print('Points where Cook\'s D is > 1: {}'.
      format(list(infl_df[infl_df['cooks_d'] > 1].index + 1)))

    # DFFITS
    print('\n***| DFFITS |***')
    DFFITS_cutoff = 2 * np.sqrt(p / n)
    print('Points which exceed DFFITS cutoff: {}'.
          format(list(infl_df[infl_df['dffits'] > DFFITS_cutoff].index + 1)))
```

```
    # DFBETAS
    print('\n***| DFBETAS |***')
    DFBETAS_cutoff = 2 / np.sqrt(n)
    for col in infl_df.columns:
        if 'dfb' in col:
            print('Points which exceed DFBETAS cutoff for {}: {}'.
                  format(col,
                         list(infl_df[infl_df[col] > DFBETAS_cutoff].
                              index + 1)))

    # COVRATIO
    print('\n***| COVRATIO |***')
    COVRATIO_cutoff_pos = 1 + 3 * p / n
    COVRATIO_cutoff_neg = 1 - 3 * p / n
    gt_cutoff = np.array(list(compress(range(len(infl.cov_ratio)),
                              infl.cov_ratio > COVRATIO_cutoff_pos))) + 1
    lt_cutoff = np.array(list(compress(range(len(infl.cov_ratio)),
                              infl.cov_ratio < COVRATIO_cutoff_neg))) + 1
    print('Points which are greater than COVRATIO upper bound cutoff: {}'.
          format(gt_cutoff))
    print('Points which are less than COVRATIO lower bound cutoff: {}'.
          format(lt_cutoff))

    return dhat_pts
```

[15]:

```
# All points
leverage_points = run_analysis(drop_point = None)

title_print('Try dropping influential points')

# Drop influential points
for i in range(1, len(leverage_points) + 1):
    comb = combinations(leverage_points, i)
    for pts in comb:
        run_analysis(pts)

# Points 1, 4 appear influential
print('\n--> Points 1 and 4 appear influential <--')
```

```
Coefficients: [ 3.25436e+02  6.80000e-02  2.55200e+00  3.80000e+00 -2.29490e+01
  2.41700e+00]
R-squared: 0.899

    dfb_Intercept    dfb_x1    dfb_x2    dfb_x3    dfb_x4    dfb_x5   cooks_d  \
0       -0.650262 -0.435297  0.193029  1.411300  0.378643 -1.099424  0.375089
1       -0.011559  0.005656  0.024343  0.003432 -0.010825  0.011227  0.000309
```

2	−0.014808	−0.024453	−0.016779	0.023476	0.022013	0.026848	0.005696
3	0.002208	0.987087	−1.022694	−0.878006	1.873989	−0.615708	1.101218
4	0.040237	−0.059983	−0.032735	0.011278	−0.045723	0.012378	0.001509
5	−0.008331	0.077484	−0.012171	−0.036789	0.029652	0.012072	0.002200
6	0.034348	−0.251545	0.004691	0.107034	−0.025644	−0.107824	0.015202
7	0.314302	−0.758889	−0.255013	0.104827	−0.034604	−0.286666	0.129157
8	0.181243	0.176124	−0.192785	−0.209058	−0.326112	0.513198	0.077888
9	0.067652	0.078741	−0.083447	−0.093096	−0.127054	0.239105	0.021118
10	−0.002020	−0.001250	0.002395	0.004031	0.001818	−0.006794	0.000025
11	−0.043673	0.020632	0.010777	−0.048159	0.092226	0.053902	0.014138
12	−0.016456	−0.011500	0.027630	−0.012353	0.029844	−0.006728	0.001216
13	0.022355	−0.004016	−0.034586	0.003643	−0.018740	0.005802	0.000404
14	0.023058	0.001983	0.003856	−0.065121	0.010484	0.015792	0.002367
15	0.438398	−0.138762	−0.625543	−0.143128	−0.196630	0.205495	0.085585
16	−0.021730	0.010100	0.015312	0.004751	0.020566	0.001969	0.000234
17	−0.058175	0.082987	0.292622	0.182041	−0.312095	−0.210324	0.153429
18	0.346051	−0.017083	−0.196286	−0.273684	−0.382732	0.311561	0.034934
19	−0.086891	0.010615	0.014062	0.066022	0.144261	−0.106660	0.005143
20	−0.011577	0.058752	0.076128	−0.052995	−0.076632	0.119400	0.006715
21	−0.942814	0.434040	1.116032	0.144539	0.496061	0.128682	0.270561
22	−0.010436	0.026076	−0.007384	−0.061900	0.047793	0.081672	0.006697
23	0.049658	0.265351	0.044671	0.220436	−0.480786	−0.121384	0.245601
24	−0.023504	0.034986	0.011158	0.006817	0.004580	0.017579	0.000459
25	0.126933	−0.156130	−0.038363	−0.034092	−0.090438	−0.045799	0.009698
26	0.083988	−0.127925	0.025901	−0.015913	−0.110619	−0.018061	0.008388
27	−0.019469	0.026777	−0.008167	0.000589	0.036939	−0.004132	0.000549
28	−0.206022	−0.122063	0.342483	0.205125	−0.010703	−0.098899	0.036763

	standard_resid	hat_diag	dffits_internal	student_resid	dffits
0	0.969344	0.705461	1.500179	0.968017	1.498124
1	0.129202	0.099829	0.043027	0.126408	0.042096
2	0.561851	0.097692	0.184873	0.553311	0.182063
3	2.894219	0.440963	2.570468	3.549908	3.152811
4	−0.205319	0.176817	−0.095157	−0.200990	−0.093151
5	0.266154	0.157045	0.114880	0.260706	0.112528
6	−0.567502	0.220710	−0.302015	−0.558955	−0.297466
7	−1.636853	0.224346	−0.880308	−1.703144	−0.915960
8	−1.515296	0.169111	−0.683615	−1.562007	−0.704689
9	−0.953275	0.122370	−0.355958	−0.951303	−0.355222
10	0.043494	0.074643	0.012353	0.042540	0.012082
11	−1.114583	0.063917	−0.291249	−1.120772	−0.292866
12	−0.272683	0.089362	−0.085420	−0.267121	−0.083678
13	0.089136	0.233600	0.049211	0.087192	0.048138
14	−0.350873	0.103433	−0.119176	−0.344083	−0.116870
15	0.857663	0.411107	0.716598	0.852554	0.712330
16	0.085527	0.161091	0.037478	0.083660	0.036661
17	−1.905844	0.202199	−0.959466	−2.031229	−1.022590
18	0.799950	0.246732	0.457826	0.793483	0.454125

```
19       -0.379522  0.176443        -0.175668      -0.372347 -0.172347
20        0.518279  0.130428         0.200722       0.509873  0.197467
21        2.091182  0.270723         1.274115       2.272648  1.384678
22        0.480335  0.148327         0.200455       0.472151  0.197040
23       -1.799757  0.312686        -1.213921      -1.898987 -1.280851
24       -0.071693  0.348858        -0.052476      -0.070125 -0.051329
25        0.521723  0.176124         0.241223       0.513302  0.237329
26        0.529176  0.152345         0.224339       0.520724  0.220756
27       -0.151075  0.126199        -0.057414      -0.147828 -0.056179
28        1.086494  0.157438         0.469658       1.090978  0.471596

***| Hat Diagonal |***
Leverage calculation (2 * p \ n) = 0.414
Points where hat diagonal exceeds leverage calculation: [1, 4]

***| Cook's D |***
Points where Cook's D is > 1: [4]

***| DFFITS |***
Points which exceed DFFITS cutoff: [1, 4, 22]

***| DFBETAS |***
Points which exceed DFBETAS cutoff for dfb_Intercept: [16]
Points which exceed DFBETAS cutoff for dfb_x1: [4, 22]
Points which exceed DFBETAS cutoff for dfb_x2: [22]
Points which exceed DFBETAS cutoff for dfb_x3: [1]
Points which exceed DFBETAS cutoff for dfb_x4: [1, 4, 22]
Points which exceed DFBETAS cutoff for dfb_x5: [9]

***| COVRATIO |***
Points which are greater than COVRATIO upper bound cutoff: [ 1 14 16 25]
Points which are less than COVRATIO lower bound cutoff: [4]

###################################
| Try dropping influential points |
###################################

Points dropped: (1,)
Coefficients: [ 3.8803e+02  8.0000e-02  2.3110e+00  1.7350e+00 -2.3975e+01
4.4080e+00]
R-squared: 0.899

Points dropped: (4,)
Coefficients: [ 3.25263e+02  4.40000e-02  3.59300e+00  4.84600e+00 -2.70800e+01
  3.32500e+00]
R-squared: 0.934

Points dropped: (1, 4)
```

```
Coefficients: [ 3.58769e+02  5.20000e-02  3.43400e+00  3.71100e+00 -2.75100e+01
  4.36500e+00]
R-squared: 0.933

--> Points 1 and 4 appear influential <--
```