Rodriguez_Felipe_DSC530_10.2Exercise

February 17, 2023

[1]:
```
import pandas
import numpy as np
import statsmodels.formula.api as smf

import thinkplot
import thinkstats2
import timeseries
import regression
```

Exercise 12-1: The linear model I used in this chapter has the obvious drawback that it is linear, and there is no reason to expect prices to change linearly over time. We can add flexibility to the model by adding a quadratic term, as we did in Section 11.3.

Use a quadratic model to fit the time series of daily prices, and use the model to generate predictions. You will have to write a version of `RunLinearModel` that runs that quadratic model, but after that you should be able to reuse code from the chapter to generate predictions.

[2]:
```
# Variation of RunLinearModel
def RunQuadraticModel(daily):
    # Adds quadratic term in years
    daily['years2'] = daily.years**2
    # Runs model
    model = smf.ols('ppg ~ years + years2', data=daily)
    results = model.fit()
    return model, results
```

[20]:
```
# Plots Model
def PlotQuadraticModel(daily, name):
    # Runs Quadratic model
    model, results = RunQuadraticModel(daily)
    # Performs Regression
    regression.SummarizeResults(results)
    # Plots Fitted Values
    timeseries.PlotFittedValues(model, results, label=name)
    thinkplot.Show(root='timeseries1',
                   title='Fitted Values',
                   xlabel='Years',
                   xlim=[-0.1, 4],
```

```
                    ylabel='price per gram ($)')
    # Plots Predictions
    years = np.linspace(0, 5, 101)
    thinkplot.Scatter(daily.years, daily.ppg, alpha=0.1, label=name)
    timeseries.PlotPredictions(daily, years, func=RunQuadraticModel)
    thinkplot.Show(root='timeseries2',
                    title='Predictions',
                    xlabel='Years',
                    xlim=[years[0]-0.1, years[-1]+0.1],
                    ylabel='Price per Gram ($)')
```

Exercise 12-2: Write a definition for a class named `SerialCorrelationTest` that extends `HypothesisTest` from Section 9.2. It should take a series and a lag as data, compute the serial correlation of the series with the given lag, and then compute the p-value of the observed correlation.

Use this class to test whether the serial correlation in raw price data is statistically significant. Also test the residuals of the linear model and (if you did the previous exercise), the quadratic model.

[8]:
```
# Serial Correlation Test Class
class SerialCorrelationTest(thinkstats2.HypothesisTest):
    def TestStatistic(self, data):
        series, lag = data
        # Computes Serial Correlation
        test_stat = abs(thinkstats2.SerialCorr(series, lag))
        return test_stat

    def RunModel(self):
        series, lag = self.data
        # Creates Permutation
        permutation = series.reindex(np.random.permutation(series.index))
        return permutation, lag
```

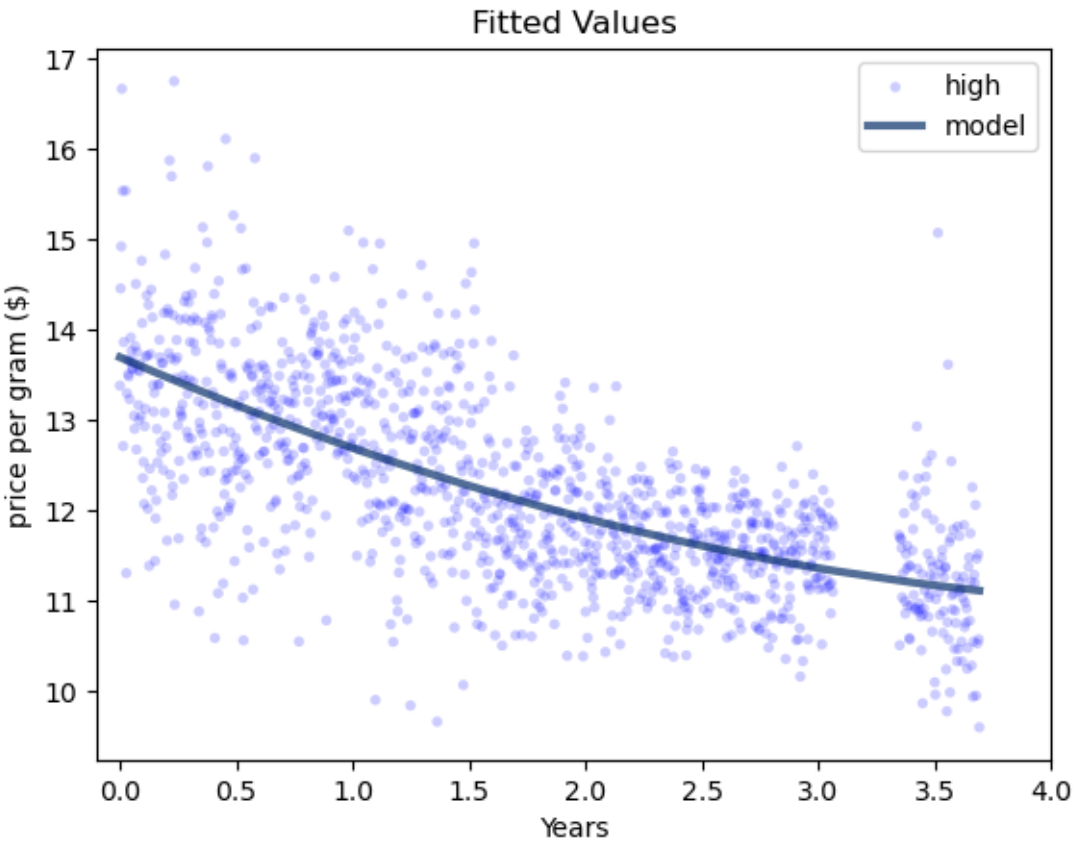
[17]:
```
def TestSerialCorr(daily):
    # Tests the correlation between consecutive prices
    series = daily.ppg
    test = SerialCorrelationTest((series, 1))
    pvalue = test.PValue()
    print('Correlation between prices\n', 'Test Value:', test.actual, 'P-Value:
 ↪', pvalue)

    # Tests serial correlation in residuals of the linear model
    _, results = timeseries.RunLinearModel(daily)
    series = results.resid
    test = SerialCorrelationTest((series, 1))
    pvalue = test.PValue()
    print('Correlation of residuals in LM\n', 'Test Value:', test.actual,␣
 ↪'P-Value:', pvalue)
```

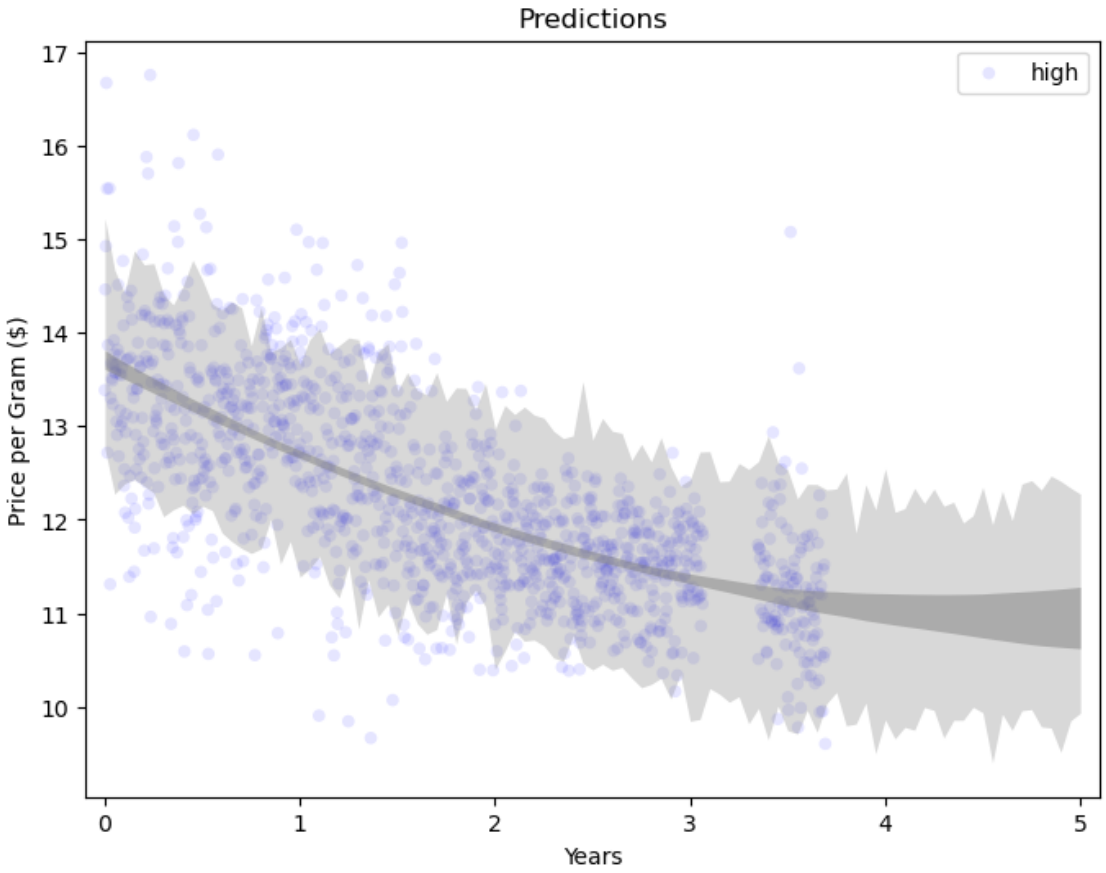
```
    # Tests for serial correlation in residuals of the quadratic model
    _, results = RunQuadraticModel(daily)
    series = results.resid
    test = SerialCorrelationTest((series, 1))
    pvalue = test.PValue()
    print('Correlation of residuals in QM\n', 'Test Value:', test.actual,␣
 ↪'P-Value:', pvalue)
```

Output for both assignments displayed below

[21]:
```
def main(name):
    transactions = timeseries.ReadData()

    dailies = timeseries.GroupByQualityAndDay(transactions)
    name = 'high'
    daily = dailies[name]

    PlotQuadraticModel(daily, name)
    TestSerialCorr(daily)

if __name__ == '__main__':
    import sys
    main(sys.argv)
```

```
Intercept   13.7   (0)
years   -1.12   (5.86e-38)
years2   0.113   (4.82e-07)
R^2 0.4553
Std(ys) 1.096
Std(res) 0.809
```


Fitted Values
high
model
price per gram ($)
Years
17
16
15
14
13
12
11
10
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0




```
Correlation between prices
 Test Value: 0.4852293761947381 P-Value: 0.0
Correlation of residuals in LM
 Test Value: 0.07570473767506267 P-Value: 0.01
Correlation of residuals in QM
 Test Value: 0.05607308161289924 P-Value: 0.049
```

```
<Figure size 800x600 with 0 Axes>
```