**Week 10 Assignment**

Michael K. Zoucha

College of Science & Information Technology, Bellevue University

DSC 510: Introduction to Programming

Professor Eller

04 November 2021

**Week 10 Assignment**

Exercise 12-1:

Function RunQuadraticModel takes the linear model and adds a year^2 variable before running the regression model

def RunQuadraticModel(daily):  
  
 daily['years2'] = daily.years\*\*2  
 model = smf.ols('ppg ~ years + years2', data=daily)  
 results = model.fit()  
 return model, results  
  
name = 'high'  
daily = dailies[name]  
  
model, results = RunQuadraticModel(daily)  
print(results.summary())

From the results, we can see that years2 is still statistically significant with a p-value of less than 0.000

Text

Description automatically generated

Using the PlotFittedValues function given in the exercise, we can plot the actual values using a scatter plot and the fitted (model) values using a line

PlotFittedValues(model, results, label=name)  
thinkplot.Config(title='Fitted values',  
 xlabel='Years',  
 xlim=[-0.1, 3.8],  
 ylabel='price per gram ($)')

Chart, scatter chart

Description automatically generated

Using the PlotPredictions function, we can use the OLS model to predict future values

ears = np.linspace(0, 5, 101)  
thinkplot.Scatter(daily.years, daily.ppg, alpha=0.1, label=name)  
PlotPredictions(daily, years, func=RunQuadraticModel)  
thinkplot.Config(title='predictions',  
 xlabel='Years',  
 xlim=[years[0]-0.1, years[-1]+0.1],  
 ylabel='Price per gram ($)')

Chart, scatter chart

Description automatically generated

Results:

From both the OLS summary and fitted values chart, we can see the model is an excellent fit to the actual data provided. Using this model, we can predict the price per gram is going to continue dropping but will level out around year 5.

Exercise 12-2:

SerialCorrelationTest extends Hypothesis test and contains two functions within the class: TestStatistic for computing the test stat (p-value) and RunModel to run the model of the null hypopthesis and return simulated data.

class SerialCorrelationTest(thinkstats2.HypothesisTest):  
  
 def TestStatistic(self, data):  
  
 series, lag = data  
 test\_stat = abs(SerialCorr(series, lag))  
 return test\_stat  
  
 def RunModel(self):  
  
 series, lag = self.data  
 permutation = series.reindex(np.random.permutation(series.index))  
 return permutation, lag

Test correlation and statistical significance of that correlation

name = 'high'  
daily = dailies[name]  
  
series = daily.ppg  
test = SerialCorrelationTest((series, 1))  
pvalue = test.PValue()  
print(test.actual, pvalue)



Test correlation and statistical significance of that correlation using linear model residuals

\_, results = RunLinearModel(daily)  
series = results.resid  
test = SerialCorrelationTest((series, 1))  
pvalue = test.PValue()  
print(test.actual, pvalue)



Test correlation and statistical significance of that correlation using quadratic model residuals

\_, results = RunQuadraticModel(daily)  
series = results.resid  
test = SerialCorrelationTest((series, 1))  
pvalue = test.PValue()  
print(test.actual, pvalue)



Results:  
As shown from the low p-values in all 3 results, we know that there is a highly statistically significant serial correlation between the past and future price per gram.