D214: Data Analytics Graduate Capstone

Performance Assessment

Task 2

Logan Rosemeyer

Western Governor's University

D214: Data Analytics Graduate Capstone

Dr. Daniel Smith

May 8, 2023

Contents

A: Research Question

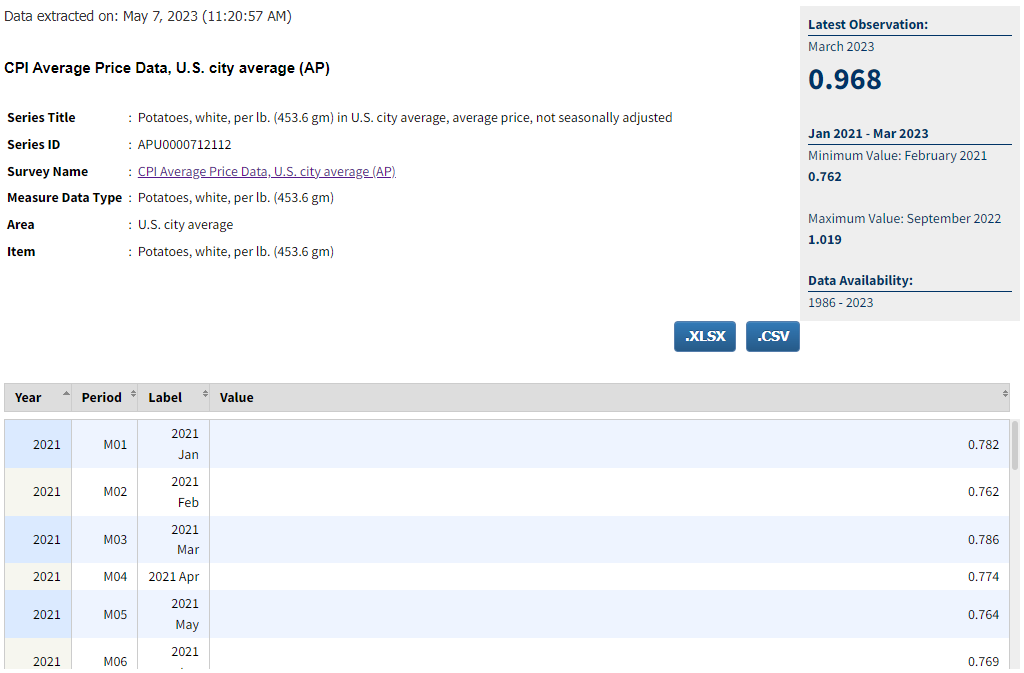
The original research question from task 1 was: Is the price of gas useful in predicting the price of various common grocery items? I feel like this is a good research question because it involves time series data with a lot of very common grocery items where data should be abundant. This data would be useful for food distributors and grocers. My null hypothesis is that the relationship between gas and grocery prices is not statistically significant. My alternate hypothesis is that the relationship between gas and grocery prices is statistically significant. This is significant for grocers and distributors because it can help them plan what there expenses will be, as well as what to set their prices at.

B: Data Collection

I collected monthly data on grocery and diesel prices from 1980 to 2023 from the US Department of Labor. One advantage was that the files could easily be downloaded as csv files and were relatively small. They really only contained the price and the date, which made the data really easy to work with. One disadvantage was that there were months that did not have prices. Usually there were not too many missing values in a row so I was able to take the average of the prices from the previous month and next month. This might not be exactly accurate, but it was the best option. The diesel data set was monthly average price per gallon data from 1998 to present. The eggs data set was monthly average price grade A, large, per dozen from 1980 to present. The bread data set was monthly average price per pound from 1980 to present. The flour data set was monthly average price per pound from 1980 to present. The ground beef data set was monthly average price per pound from 1980 to present. The chicken breast data set was monthly average price per pound from 1980 to present. The potatoes data set was monthly average price per pound from 1980 to present. The lettuce data set was monthly average price per pound from 1980 to 2020. The tomatoes data set was monthly average price per pound from 1980 to present. The bananas data set was monthly average price per pound from 1980 to present. The strawberries data set was monthly average price per pound from 1980 to present.

C: Data Extraction and Preparation

The extraction from the website was pretty simple. I just chose the years of data that I wanted to view and then the website gave me the option to download as an .xlsx or .csv file.



Data extracted on: May 7, 2023 (11:20:57 AM)

CPI Average Price Data, U.S. city average (AP)

Series Title	:	Potatoes, white, per lb. (453.6 gm) in U.S. city average, average price, not seasonally adjusted
Series ID	:	APU0000712112
Survey Name	:	CPI Average Price Data, U.S. city average (AP)
Measure Data Type	:	Potatoes, white, per lb. (453.6 gm)
Area	:	U.S. city average
Item	:	Potatoes, white, per lb. (453.6 gm)

Latest Observation:
March 2023
0.968

Jan 2021 - Mar 2023
Minimum Value: February 2021
0.762

Maximum Value: September 2022
1.019

Data Availability:
1986 - 2023

.XLSX .CSV

Year	Period	Label	Value
2021	M01	2021 Jan	0.782
2021	M02	2021 Feb	0.762
2021	M03	2021 Mar	0.786
2021	M04	2021 Apr	0.774
2021	M05	2021 May	0.764
2021	M06	2021	0.769

After I downloaded the .csv files, I uploaded them into my Jupyter Notebook using pandas. Then, I imputed missing values by finding the mean of the previous and next month. The data for diesel prices starts in 1998, so I filtered out all years before that for all grocery items.

```
import pandas as pd


diesel = pd.read_csv("Diesel1998.csv")
eggs = pd.read_csv("Eggs1980.csv")
flour = pd.read_csv("Flour1980.csv")
lettuce = pd.read_csv("Lettuce19802020.csv")
potatoes = pd.read_csv("Potato1986.csv")
strawberries = pd.read_csv("Strawberry1980.csv")
tomatoes = pd.read_csv("Tomato1980.csv")
bananas = pd.read_csv("Banana1980.csv")
beef = pd.read_csv("Beef1984.csv")
bread = pd.read_csv("Bread1980.csv")
chicken = pd.read_csv("Chicken1980.csv")


diesel.head()
```

Show hidden output

```
diesel = diesel[diesel.Year < 2023]
eggs = eggs[(eggs.Year >=1998) & (eggs.Year < 2023)]
flour = flour[(flour.Year >= 1998) & (flour.Year < 2023)]
lettuce = lettuce[(lettuce.Year >= 1998) & (lettuce.Year < 2023)]
potatoes = potatoes[(potatoes.Year >= 1998) & (potatoes.Year < 2023)]
strawberries = strawberries[(strawberries.Year >= 1998) & (strawberries.Year <2023)]
tomatoes = tomatoes[(tomatoes.Year >= 1998) & (tomatoes.Year < 2023)]
bananas = bananas[(bananas.Year >= 1998) & (bananas.Year < 2023)]
beef = beef[(beef.Year >= 1998) & (beef.Year < 2023)]
bread = bread[(bread.Year >=1998) & (bread.Year < 2023)]
chicken = chicken[(chicken.Year >= 1998) & (chicken.Year < 2023)]


diesel.Year.value_counts()
```

Show hidden output

```
eggs["Year"].value_counts()
```

Show hidden output

```
beef["Year"].value_counts()
```

Show hidden output

```
print(beef[beef.Year == 2012])
```

Show hidden output

```
beefimpute = {"Series ID":"APU0000703112", "Year":2012, "Period":"M10", "Label":"2012 Oct", "Value":3.100}
beef=beef.append(beefimpute, ignore_index=True)
beef.drop_duplicates(subset=['Label', 'Value'], keep='first', inplace=True)
beef = beef.sort_values(by=['Year','Period'], ignore_index=True)
print(beef[beef["Year"]==2012])
```

Show hidden output

```
chicken["Year"].value_counts()
```

Show hidden output

```
print(chicken[chicken["Year"]==2020])
```

Show hidden output

```
chickenimpute = {"Series ID":"APU0000706111", "Year":2020, "Period":"M05", "Label":"2020 May", "Value":1.659}
chicken=chicken.append(chickenimpute, ignore_index=True)
chicken.drop_duplicates(subset=['Label', 'Value'], keep='first', inplace=True)
chicken = chicken.sort_values(by=['Year','Period'], ignore_index=True)
print(chicken[chicken["Year"]==2020])
```

```
print(flour[flour["Year"]==2020])
```

Show hidden output

```
flourimpute = {"Series ID":"APU0000701111", "Year":2020, "Period":"M04", "Label":"2020 Apr", "Value":0.452}
flour=flour.append(flourimpute, ignore_index=True)
flour.drop_duplicates(subset=['Label', 'Value'], keep='first', inplace=True)
flour = flour.sort_values(by=['Year', 'Period'], ignore_index=True)
print(flour[flour["Year"]==2020])
```

Show hidden output

```
lettuce["Year"].value_counts()
```

Show hidden output

```
print(lettuce[lettuce["Year"]==2013])
```

Show hidden output

```
lettuceimpute = {"Series ID":"APU0000712211", "Year":2013, "Period":"M10", "Label":"2013 Oct", "Value":1.023}
lettuce=lettuce.append(lettuceimpute, ignore_index=True)
lettuce.drop_duplicates(subset=['Label', 'Value'], keep='first', inplace=True)
lettuce = lettuce.sort_values(by=['Year', 'Period'], ignore_index=True)
print(lettuce[lettuce["Year"]==2013])
```

Show hidden output

```
potatoes["Year"].value_counts()
```

Show hidden output

```
print(potatoes[potatoes["Year"]==2020])
```

Show hidden output

```
potatoesimpute = {"Series ID":"APU0000712112", "Year":2020, "Period":"M03", "Label":"2020 Mar", "Value":0.809}
potatoes=potatoes.append(potatoesimpute, ignore_index=True)
potatoesimpute = {"Series ID":"APU0000712112", "Year":2020, "Period":"M04", "Label":"2020 Apr", "Value":0.832}
potatoes=potatoes.append(potatoesimpute, ignore_index=True)
potatoes.drop_duplicates(subset=['Label'], keep='first', inplace=True)
potatoes = potatoes.sort_values(by=['Year', 'Period'], ignore_index=True)
print(potatoes[potatoes["Year"]==2020])
print(potatoes["Year"].value_counts())
```

Show hidden output

```
print(strawberries["Year"].value_counts())
print(strawberries["Period"].value_counts())
```

Show hidden output

```
print(strawberries[strawberries["Year"]==2013])
```

Show hidden output

```
strawberriesimpute = {"Series ID":"APU0000711415", "Year":2013, "Period":"M10", "Label":"2013 Oct", "Value":2.316}
strawberries=strawberries.append(strawberriesimpute, ignore_index=True)
strawberries.drop_duplicates(subset=['Label'], keep='first', inplace=True)
strawberries = strawberries.sort_values(by=['Year', 'Period'], ignore_index=True)
print(strawberries[strawberries["Year"]==2013])
print(strawberries[strawberries["Year"]==2015])
```

Show hidden output

```
strawberriesimpute = {"Series ID":"APU0000711415", "Year":2015, "Period":"M11", "Label":"2015 Nov", "Value":3.013}
strawberries=strawberries.append(strawberriesimpute, ignore_index=True)
strawberries.drop_duplicates(subset=['Label'], keep='first', inplace=True)
strawberries = strawberries.sort_values(by=['Year', 'Period'], ignore_index=True)
print(strawberries[strawberries["Year"]==2015])
```

```
strawberriesimpute = {"Series ID":"APU0000711415", "Year":2002, "Period":"M12", "Label":"2002 Dec", "Value":2.200}
strawberries=strawberries.append(strawberriesimpute, ignore_index=True)
strawberriesimpute = {"Series ID":"APU0000711415", "Year":2003, "Period":"M01", "Label":"2003 Jan", "Value":2.176}
strawberries=strawberries.append(strawberriesimpute, ignore_index=True)
strawberries.drop_duplicates(subset=['Label'], keep='first', inplace=True)
strawberries = strawberries.sort_values(by=['Year', 'Period'], ignore_index=True)
print(strawberries[strawberries["Year"]==2003])
print(strawberries[strawberries["Year"]==2004])
```

Show hidden output

```
strawberriesimpute = {"Series ID":"APU0000711415", "Year":2003, "Period":"M12", "Label":"2003 Dec", "Value":2.445}
strawberries=strawberries.append(strawberriesimpute, ignore_index=True)
strawberries.drop_duplicates(subset=['Label'], keep='first', inplace=True)
strawberries = strawberries.sort_values(by=['Year', 'Period'], ignore_index=True)
print(strawberries["Year"].value_counts())
```

Show hidden output

```
print(strawberries[strawberries["Year"]==1998])
print(strawberries[strawberries["Year"]==1999])
#print(strawberries[strawberries["Year"]==2000])
#print(strawberries[strawberries["Year"]==2001])
#print(strawberries[strawberries["Year"]==2002])
```

Show hidden output

```
strawberriesimpute = {"Series ID":"APU0000711415", "Year":1998, "Period":"M11", "Label":"1998 Nov", "Value":1.860}
strawberries=strawberries.append(strawberriesimpute, ignore_index=True)
strawberriesimpute = {"Series ID":"APU0000711415", "Year":1998, "Period":"M12", "Label":"1998 Dec", "Value":1.941}
strawberries=strawberries.append(strawberriesimpute, ignore_index=True)
strawberriesimpute = {"Series ID":"APU0000711415", "Year":1999, "Period":"M01", "Label":"1999 Jan", "Value":2.022}
strawberries=strawberries.append(strawberriesimpute, ignore_index=True)
strawberries.drop_duplicates(subset=['Label'], keep='first', inplace=True)
strawberries = strawberries.sort_values(by=['Year', 'Period'], ignore_index=True)
print(strawberries["Year"].value_counts())
print(strawberries[strawberries["Year"]==1999])
print(strawberries[strawberries["Year"]==2000])
print(strawberries[strawberries["Year"]==2001])
#print(strawberries[strawberries["Year"]==2002])
```

Show hidden output

```
strawberriesimpute = {"Series ID":"APU0000711415", "Year":1999, "Period":"M12", "Label":"1999 Dec", "Value":2.058}
strawberries=strawberries.append(strawberriesimpute, ignore_index=True)
strawberriesimpute = {"Series ID":"APU0000711415", "Year":2000, "Period":"M11", "Label":"2000 Nov", "Value":1.749}
strawberries=strawberries.append(strawberriesimpute, ignore_index=True)
strawberriesimpute = {"Series ID":"APU0000711415", "Year":2000, "Period":"M12", "Label":"2000 Dec", "Value":1.879}
strawberries=strawberries.append(strawberriesimpute, ignore_index=True)
strawberriesimpute = {"Series ID":"APU0000711415", "Year":2001, "Period":"M01", "Label":"2001 Jan", "Value":2.009}
strawberries=strawberries.append(strawberriesimpute, ignore_index=True)
strawberries.drop_duplicates(subset=['Label'], keep='first', inplace=True)
strawberries = strawberries.sort_values(by=['Year', 'Period'], ignore_index=True)
print(strawberries["Year"].value_counts())
print(strawberries[strawberries["Year"]==1999])
print(strawberries[strawberries["Year"]==2000])
print(strawberries[strawberries["Year"]==2001])
#print(strawberries[strawberries["Year"]==2002])
```

```
strawberriesimpute = {"Series ID":"APU0000711415", "Year":2020, "Period":"M04", "Label":"2020 Apr", "Value":2.271}
strawberries=strawberries.append(strawberriesimpute, ignore_index=True)
strawberriesimpute = {"Series ID":"APU0000711415", "Year":2021, "Period":"M12", "Label":"2021 Dec", "Value":3.093}
```

ps://colab.research.google.com/drive/1m-1JtL5zjgifN0l5vvb1orYkNVnHAc9J#scrollTo=tLwvDO7bxxIP&printMode=true


```
strawberries=strawberries.append(strawberriesimpute, ignore_index=True)
strawberries.drop_duplicates(subset=['Label'], keep='first', inplace=True)
strawberries = strawberries.sort_values(by=['Year', 'Period'], ignore_index=True)
print(strawberries["Year"].value_counts())
print(strawberries["Period"].value_counts())
```

Show hidden output

```
print(tomatoes["Year"].value_counts())
print(tomatoes["Period"].value_counts())
```

Show hidden output

```
print(tomatoes[tomatoes["Year"]==2013])
print(tomatoes[tomatoes["Year"]==2020])
```

Show hidden output

```
tomatoesimpute = {"Series ID":"APU0000712311", "Year":2013, "Period":"M10", "Label":"2013 Oct", "Value":1.633}
tomatoes=tomatoes.append(tomatoesimpute, ignore_index=True)
tomatoesimpute = {"Series ID":"APU0000712311", "Year":2020, "Period":"M03", "Label":"2020 Mar", "Value":2.082}
tomatoes=tomatoes.append(tomatoesimpute, ignore_index=True)
tomatoesimpute = {"Series ID":"APU0000712311", "Year":2020, "Period":"M04", "Label":"2020 Apr", "Value":1.967}
tomatoes=tomatoes.append(tomatoesimpute, ignore_index=True)
tomatoes.drop_duplicates(subset=['Label'], keep='first', inplace=True)
tomatoes = tomatoes.sort_values(by=['Year', 'Period'], ignore_index=True)
print(tomatoes["Year"].value_counts())
print(tomatoes["Period"].value_counts())
```

After this, I created one data frame with month/year and all grocery values for that year. I had to make two data frames to match the time periods because lettuce only had data until 2020.

```
bananas = bananas.rename(columns={"Value":"bananas_value"})
bananas_value = bananas[['bananas_value']]
bananas_value.reset_index(inplace=True, drop=True)
beef = beef.rename(columns={"Value":"beef_value"})
beef_value = beef[['beef_value']]
beef_value.reset_index(inplace=True, drop=True)
bread = bread.rename(columns={"Value":"bread_value"})
bread_value = bread[['bread_value']]
bread_value.reset_index(inplace=True, drop=True)
chicken = chicken.rename(columns={"Value":"chicken_value"})
chicken_value = chicken[['chicken_value']]
chicken_value.reset_index(inplace=True, drop=True)
diesel = diesel.rename(columns={"Value":"diesel_value"})
eggs = eggs.rename(columns={"Value":"eggs_value"})
eggs_value = eggs[['eggs_value']]
eggs_value.reset_index(inplace=True, drop=True)
flour = flour.rename(columns={"Value":"flour_value"})
flour_value = flour[['flour_value']]
flour_value.reset_index(inplace=True, drop=True)
lettuce = lettuce.rename(columns={"Value":"lettuce_value"})
lettuce_value = lettuce[['lettuce_value']]
lettuce_value.reset_index(inplace=True, drop=True)
potatoes = potatoes.rename(columns={"Value":"potatoes_value"})
potatoes_value = potatoes[['potatoes_value']]
potatoes_value.reset_index(inplace=True, drop=True)
strawberries = strawberries.rename(columns={"Value":"strawberries_value"})
strawberries_value = strawberries[['strawberries_value']]
strawberries_value.reset_index(inplace=True, drop=True)
tomatoes = tomatoes.rename(columns={"Value":"tomatoes_value"})
tomatoes_value = tomatoes[['tomatoes_value']]
tomatoes_value.reset_index(inplace=True, drop=True)
```

```
prices = diesel[['Label','diesel_value']]
prices = pd.concat([prices,bananas_value, beef_value, bread_value, chicken_value, eggs_value, flour_value, potatoes_value, strawberries_value
print(prices)
```

Show hidden output

```
import matplotlib.pyplot as plt
prices.set_index('Label').plot();
```

```
diesel2019 = diesel[diesel['Year']<=2019]
lettuce2019 = lettuce[lettuce['Year']<=2019]
print(diesel2019['Period'].value_counts())
print(lettuce2019['Period'].value_counts())
```

Show hidden output

```
lettuce2019 = lettuce2019.rename(columns={"Value":"lettuce_value"})
lettuce_value2019 = lettuce2019[['lettuce_value']]
lettuce_value2019.reset_index(inplace=True, drop=True)
prices2019 = diesel2019[['Label','diesel_value']]
prices2019 = pd.concat([prices2019, lettuce_value2019], axis=1)
print(prices2019)
prices2019.set_index('Label').plot();
```

Show hidden output

```
from datetime import datetime as dt
prices['Label'] = pd.to_datetime(prices['Label'], format='%Y %b')
prices2019['Label'] = pd.to_datetime(prices2019['Label'], format='%Y %b')
prices['Label'] = prices['Label'].dt.strftime('%Y %b')
prices2019['Label'] = prices2019['Label'].dt.strftime('%Y %b')
```

```
prices.dtypes
```

Show hidden output

```
#prices.set_index('Label', inplace=True)
#prices2019.set_index('Label', inplace=True)
print(prices.dtypes)
print(prices2019.dtypes)
print(prices)
print(prices2019)
```

Show hidden output

```
print(prices.isnull().sum())
print(prices2019.isnull().sum())
```

Show hidden output

One advantage of the data preparation process was that the code was very repeatable. There was a lot of copy and pasting and changing the grocery item name. This made preparation very quick. One disadvantage was there were a few items that were missing multiple months in a row. I imputed values that would all be even differences between the previous known month and the next known month. The more months that were missing, the more inaccurate the data probably ended up being. The justification for imputing missing values instead of leaving them blank is there are usually not big swings between months. Because of this, using an average value for imputation would not lead to numbers that would be wildly inaccurate.

D: Analysis

I used the Granger Causality Test to determine whether one time series is a factor in forecasting another time series (Li, 2020). This was perfect for my analysis, because deciding whether one time series variable can be used to forecast another time series variable is exactly what my research question is. That is the main advantage. I used the gct function from the statsmodels library. The gct function tests whether the second variable can predict the first variable. I performed the gct test on diesel and each grocery item twice, once with diesel predicting the grocery item, and the other the grocery item

predicting diesel. For all results, the relationship with diesel predicting the grocery price is statistically significant, while the grocery price predicting diesel is not.

```
from statsmodels.tsa.stattools import grangercausalitytests as gct
gct(prices[['bananas_value', 'diesel_value']], maxlag=1)
gct(prices[['diesel_value','bananas_value']], maxlag=1)
```

```
Granger Causality
number of lags (no zero) 1
ssr based F test:         F=8.4802  , p=0.0039  , df_denom=296, df_num=1
ssr based chi2 test:   chi2=8.5662  , p=0.0034  , df=1
likelihood ratio test: chi2=8.4457  , p=0.0037  , df=1
parameter F test:         F=8.4802  , p=0.0039  , df_denom=296, df_num=1

Granger Causality
number of lags (no zero) 1
ssr based F test:         F=0.0044  , p=0.9475  , df_denom=296, df_num=1
ssr based chi2 test:   chi2=0.0044  , p=0.9471  , df=1
likelihood ratio test: chi2=0.0044  , p=0.9471  , df=1
parameter F test:         F=0.0044  , p=0.9475  , df_denom=296, df_num=1
{1: ({'ssr_ftest': (0.004350242286473185, 0.9474570889182683, 296.0, 1),
   'ssr_chi2test': (0.004394332579917171, 0.9471471338339039, 1),
   'lrtest': (0.004394300289277453, 0.9471473277379349, 1),
   'params_ftest': (0.0043502422865803305, 0.9474570889174756, 296.0, 1.0)},
  [<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f5787a832e0>,
   <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f5787a80580>,
   array([[0., 1., 0.]])])}
```

```
[ ] gct(prices[['beef_value', 'diesel_value']], maxlag=1)
    gct(prices[['diesel_value','beef_value']], maxlag=1)
```

```
Granger Causality
number of lags (no zero) 1
ssr based F test:         F=5.0088  , p=0.0260  , df_denom=296, df_num=1
ssr based chi2 test:   chi2=5.0596  , p=0.0245  , df=1
likelihood ratio test: chi2=5.0172  , p=0.0251  , df=1
parameter F test:         F=5.0088  , p=0.0260  , df_denom=296, df_num=1

Granger Causality
number of lags (no zero) 1
ssr based F test:         F=0.3164  , p=0.5742  , df_denom=296, df_num=1
ssr based chi2 test:   chi2=0.3196  , p=0.5718  , df=1
likelihood ratio test: chi2=0.3195  , p=0.5719  , df=1
parameter F test:         F=0.3164  , p=0.5742  , df_denom=296, df_num=1
{1: ({'ssr_ftest': (0.316415121436443, 0.5741957872744528, 296.0, 1),
   'ssr_chi2test': (0.31962203145100154, 0.5718348787704173, 1),
   'lrtest': (0.3194513199150606, 0.5719375682792552, 1),
   'params_ftest': (0.31641512143634043, 0.5741957872744727, 296.0, 1.0)},
  [<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f57877ec370>,
   <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f57877ed6f0>,
   array([[0., 1., 0.]])])}
```

```
gct(prices[['bread_value', 'diesel_value']], maxlag=1)
gct(prices[['diesel_value','bread_value']], maxlag=1)
```

```
Granger Causality
number of lags (no zero) 1
ssr based F test:         F=10.4108 , p=0.0014  , df_denom=296, df_num=1
ssr based chi2 test:   chi2=10.5163 , p=0.0012  , df=1
likelihood ratio test: chi2=10.3356 , p=0.0013  , df=1
parameter F test:         F=10.4108 , p=0.0014  , df_denom=296, df_num=1

Granger Causality
number of lags (no zero) 1
ssr based F test:         F=0.6915  , p=0.4063  , df_denom=296, df_num=1
ssr based chi2 test:   chi2=0.6985  , p=0.4033  , df=1
likelihood ratio test: chi2=0.6977  , p=0.4036  , df=1
parameter F test:         F=0.6915  , p=0.4063  , df_denom=296, df_num=1
{1: ({'ssr_ftest': (0.6914716362892799, 0.4063345746979591, 296.0, 1),
   'ssr_chi2test': (0.6984797947651847, 0.403294977159975, 1),
   'lrtest': (0.6976652202624791, 0.4035693278909719, 1),
   'params_ftest': (0.6914716362891931, 0.4063345746979926, 296.0, 1.0)},
  [<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f5787934a90>,
   <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f5787934220>,
   array([[0., 1., 0.]])])}
```

```
[ ] gct(prices[['chicken_value', 'diesel_value']], maxlag=1)
    gct(prices[['diesel_value','chicken_value']], maxlag=1)
```

```
Granger Causality
number of lags (no zero) 1
ssr based F test:         F=6.9704  , p=0.0087  , df_denom=296, df_num=1
ssr based chi2 test:   chi2=7.0411  , p=0.0080  , df=1
likelihood ratio test: chi2=6.9594  , p=0.0083  , df=1
parameter F test:         F=6.9704  , p=0.0087  , df_denom=296, df_num=1

Granger Causality
number of lags (no zero) 1
ssr based F test:         F=0.0877  , p=0.7673  , df_denom=296, df_num=1
ssr based chi2 test:   chi2=0.0886  , p=0.7659  , df=1
likelihood ratio test: chi2=0.0886  , p=0.7659  , df=1
parameter F test:         F=0.0877  , p=0.7673  , df_denom=296, df_num=1
{1: ({'ssr_ftest': (0.08774836458243596, 0.767267123189002, 296.0, 1),
   'ssr_chi2test': (0.08863770611536605, 0.7659162266304116, 1),
   'lrtest': (0.08862457051259298, 0.7659330658089711, 1),
   'params_ftest': (0.08774836458252029, 0.767267123188875, 296.0, 1.0)},
  [<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f57877ee0e0>,
   <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f57877ee5c0>,
   array([[0., 1., 0.]])])}
```

```
[ ] gct(prices[['eggs_value', 'diesel_value']], maxlag=1)
    gct(prices[['diesel_value','eggs_value']], maxlag=1)
```

```
Granger Causality
number of lags (no zero) 1
ssr based F test:         F=12.2139 , p=0.0005  , df_denom=296, df_num=1
ssr based chi2 test:   chi2=12.3377 , p=0.0004  , df=1
likelihood ratio test: chi2=12.0900 , p=0.0005  , df=1
parameter F test:         F=12.2139 , p=0.0005  , df_denom=296, df_num=1

Granger Causality
number of lags (no zero) 1
ssr based F test:         F=1.2515  , p=0.2642  , df_denom=296, df_num=1
ssr based chi2 test:   chi2=1.2641  , p=0.2609  , df=1
likelihood ratio test: chi2=1.2615  , p=0.2614  , df=1
parameter F test:         F=1.2515  , p=0.2642  , df_denom=296, df_num=1
{1: ({'ssr_ftest': (1.2514549763784133, 0.2641817921541829, 296.0, 1),
   'ssr_chi2test': (1.2641386416795457, 0.26086915358884283, 1),
   'lrtest': (1.2614738314821352, 0.26137229868204576, 1),
   'params_ftest': (1.2514549763786058, 0.26418179215414417, 296.0, 1.0)},
  [<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f5787935480>,
   <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f57879349d0>,
   array([[0., 1., 0.]])])}
```

```
[ ] gct(prices[['flour_value', 'diesel_value']], maxlag=1)
    gct(prices[['diesel_value','flour_value']], maxlag=1)
```

```
Granger Causality
number of lags (no zero) 1
ssr based F test:         F=13.6090 , p=0.0003  , df_denom=296, df_num=1
ssr based chi2 test:   chi2=13.7469 , p=0.0002  , df=1
likelihood ratio test: chi2=13.4403 , p=0.0002  , df=1
parameter F test:         F=13.6090 , p=0.0003  , df_denom=296, df_num=1

Granger Causality
number of lags (no zero) 1
ssr based F test:         F=2.0594  , p=0.1523  , df_denom=296, df_num=1
ssr based chi2 test:   chi2=2.0802  , p=0.1492  , df=1
likelihood ratio test: chi2=2.0730  , p=0.1499  , df=1
parameter F test:         F=2.0594  , p=0.1523  , df_denom=296, df_num=1
{1: ({'ssr_ftest': (2.059375120046051, 0.15232795292306975, 296.0, 1),
   'ssr_chi2test': (2.080247165181653, 0.14921601282313543, 1),
   'lrtest': (2.073044053467356, 0.14992202433366592, 1),
   'params_ftest': (2.0593751200462433, 0.15232795292305804, 296.0, 1.0)},
  [<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f5787935a80>,
   <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f5787937e20>,
   array([[0., 1., 0.]])])}
```

```
gct(prices[['potatoes_value', 'diesel_value']], maxlag=1)
gct(prices[['diesel_value','potatoes_value']], maxlag=1)
```

```
Granger Causality
number of lags (no zero) 1
ssr based F test:         F=14.0819 , p=0.0002  , df_denom=296, df_num=1
ssr based chi2 test:   chi2=14.2247 , p=0.0002  , df=1
likelihood ratio test: chi2=13.8967 , p=0.0002  , df=1
parameter F test:         F=14.0819 , p=0.0002  , df_denom=296, df_num=1

Granger Causality
number of lags (no zero) 1
ssr based F test:         F=0.3748  , p=0.5409  , df_denom=296, df_num=1
ssr based chi2 test:   chi2=0.3786  , p=0.5383  , df=1
likelihood ratio test: chi2=0.3784  , p=0.5385  , df=1
parameter F test:         F=0.3748  , p=0.5409  , df_denom=296, df_num=1
{1: ({'ssr_ftest': (0.37482557449099313, 0.5408556015501305, 296.0, 1),
   'ssr_chi2test': (0.37862448234056406, 0.5383403109269729, 1),
   'lrtest': (0.37838495794017035, 0.5384688493386187, 1),
   'params_ftest': (0.3748255744909425, 0.540855601550166, 296.0, 1.0)},
  [<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f578798cd90>,
   <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f578798faf0>,
   array([[0., 1., 0.]])])}
```

```
[ ] gct(prices[['strawberries_value', 'diesel_value']], maxlag=1)
    gct(prices[['diesel_value','strawberries_value']], maxlag=1)
```

```
Granger Causality
number of lags (no zero) 1
ssr based F test:         F=7.3135  , p=0.0072  , df_denom=296, df_num=1
ssr based chi2 test:   chi2=7.3876  , p=0.0066  , df=1
likelihood ratio test: chi2=7.2978  , p=0.0069  , df=1
parameter F test:         F=7.3135  , p=0.0072  , df_denom=296, df_num=1

Granger Causality
number of lags (no zero) 1
ssr based F test:         F=0.0306  , p=0.8613  , df_denom=296, df_num=1
ssr based chi2 test:   chi2=0.0309  , p=0.8605  , df=1
likelihood ratio test: chi2=0.0309  , p=0.8605  , df=1
parameter F test:         F=0.0306  , p=0.8613  , df_denom=296, df_num=1
{1: ({'ssr_ftest': (0.030582009031774424, 0.8612957414464664, 296.0, 1),
   'ssr_chi2test': (0.03089196182601538, 0.8604816369870885, 1),
   'lrtest': (0.03089036609435425, 0.8604852035087048, 1),
   'params_ftest': (0.0305820090318818, 0.8612957414461713, 296.0, 1.0)},
  [<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f5787935840>,
   <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f5787936710>,
   array([[0., 1., 0.]])])}
```

```
[ ] gct(prices[['tomatoes_value', 'diesel_value']], maxlag=1)
    gct(prices[['diesel_value','tomatoes_value']], maxlag=1)
```

```
Granger Causality
number of lags (no zero) 1
ssr based F test:         F=6.9448  , p=0.0088  , df_denom=296, df_num=1
ssr based chi2 test:   chi2=7.0152  , p=0.0081  , df=1
likelihood ratio test: chi2=6.9342  , p=0.0085  , df=1
parameter F test:         F=6.9448  , p=0.0088  , df_denom=296, df_num=1

Granger Causality
number of lags (no zero) 1
ssr based F test:         F=0.7269  , p=0.3946  , df_denom=296, df_num=1
ssr based chi2 test:   chi2=0.7343  , p=0.3915  , df=1
likelihood ratio test: chi2=0.7334  , p=0.3918  , df=1
parameter F test:         F=0.7269  , p=0.3946  , df_denom=296, df_num=1
{1: ({'ssr_ftest': (0.7268993396130958, 0.39457940634851807, 296.0, 1),
   'ssr_chi2test': (0.734266562649715, 0.39150371561896524, 1),
   'lrtest': (0.7333664517188936, 0.3917941621322323, 1),
   'params_ftest': (0.7268993396130834, 0.3945794063485396, 296.0, 1.0)},
  [<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f578798fbe0>,
   <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f578798c6a0>,
   array([[0., 1., 0.]])])}
```

```
[ ] gct(prices2019[['lettuce_value', 'diesel_value']], maxlag=1)
    gct(prices2019[['diesel_value','lettuce_value']], maxlag=1)
```

```
Granger Causality
number of lags (no zero) 1
ssr based F test:         F=9.6699  , p=0.0021  , df_denom=260, df_num=1
ssr based chi2 test:   chi2=9.7815  , p=0.0018  , df=1
likelihood ratio test: chi2=9.6040  , p=0.0019  , df=1
parameter F test:         F=9.6699  , p=0.0021  , df_denom=260, df_num=1

Granger Causality
number of lags (no zero) 1
ssr based F test:         F=1.9584  , p=0.1629  , df_denom=260, df_num=1
ssr based chi2 test:   chi2=1.9810  , p=0.1593  , df=1
likelihood ratio test: chi2=1.9735  , p=0.1601  , df=1
parameter F test:         F=1.9584  , p=0.1629  , df_denom=260, df_num=1
{1: ({'ssr_ftest': (1.958358981692023, 0.16288085445178893, 260.0, 1),
   'ssr_chi2test': (1.9809554314807771, 0.15928979023550632, 1),
   'lrtest': (1.973532256498629, 0.16007343149713094, 1),
   'params_ftest': (1.958358981692001, 0.16288085445178893, 260.0, 1.0)},
  [<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f578798f190>,
   <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f578798cdf0>,
   array([[0., 1., 0.]])])}
```

One disadvantage would be that I didn't account for general inflation, which could have a slight impact on the analysis.

E: Data Summary and Implications

Overall, every single one of my grocery items could be predicted with gas prices. All Granger Causality Tests came back with a p-value under 0.05, which is statistically significant. When flipped the other way where we tested if the grocery items predicted gas prices, none of the test came back statistically significant. One limitation was that I was not able to adjust the priced for general inflation. Based on these results, I believe that grocers can use this information to better plan their buying and selling of products based on gas prices. One direction that could be taken with this dataset is to adjust it for inflation and see how that affects the results. Another approach that could be taken is to look at products that are made in factories. I used products that primarily come from farming, which likely uses more gas because of the use of tractors, as opposed to products that come from factories.

F: Sources

Li, Susan. *A Quick Introduction On Granger Causality Testing For Time Series Analysis.* Towards Data Science. Dec 23, 2020. Retrieved May 5, 2023.
https://towardsdatascience.com/a-quick-introduction-on-granger-causality-testing-for-time-series-analysis-7113dc9420d2