Avocado Price Analysis

February 15, 2021

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from scipy import stats
from scipy.stats import linregress
```

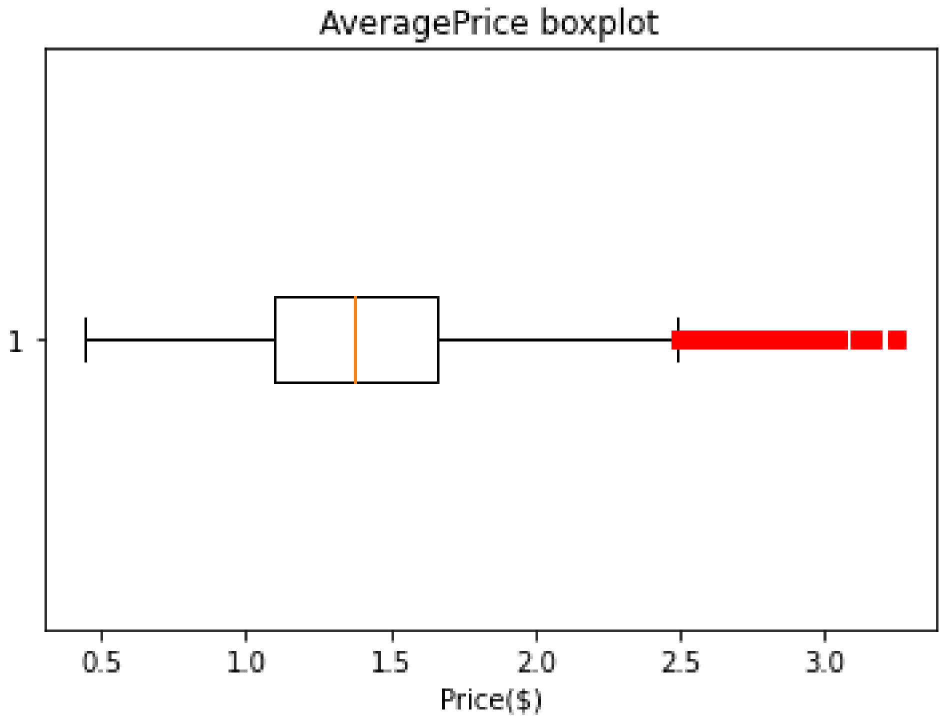
[2]:
```
data = pd.read_csv('avocado.csv',delimiter = ',')
data.drop(columns='Unnamed: 0',inplace=True)
```

[3]:
```
data.info()
```

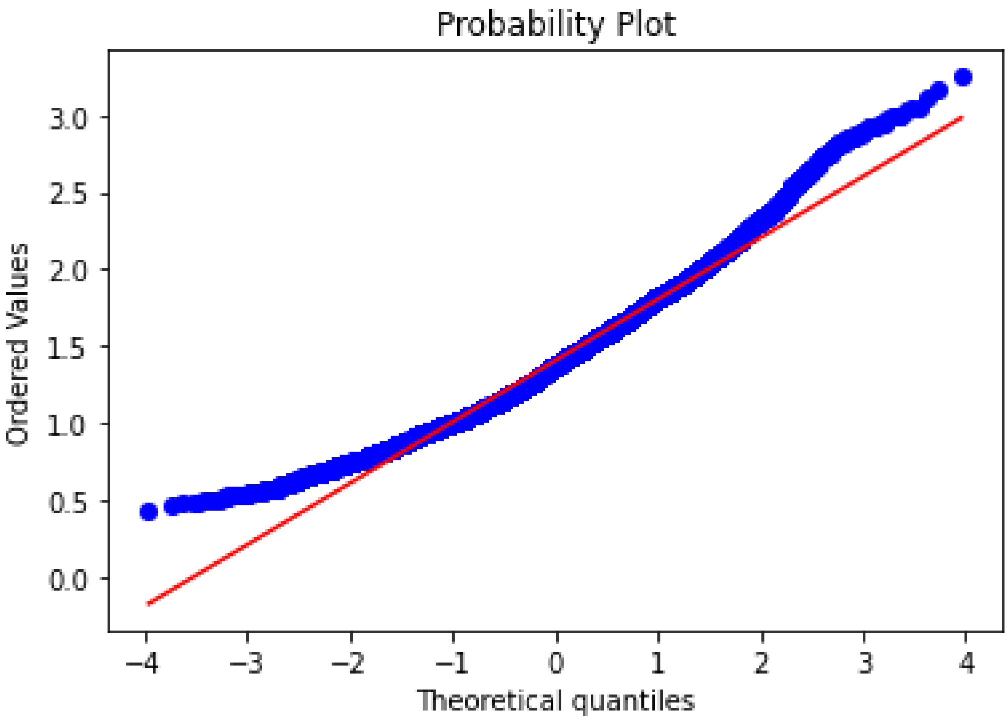
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18249 entries, 0 to 18248
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Date          18249 non-null  object
 1   AveragePrice  18249 non-null  float64
 2   Total Volume  18249 non-null  float64
 3   4046          18249 non-null  float64
 4   4225          18249 non-null  float64
 5   4770          18249 non-null  float64
 6   Total Bags    18249 non-null  float64
 7   Small Bags    18249 non-null  float64
 8   Large Bags    18249 non-null  float64
 9   XLarge Bags   18249 non-null  float64
 10  type          18249 non-null  object
 11  year          18249 non-null  int64
 12  region        18249 non-null  object
dtypes: float64(9), int64(1), object(3)
memory usage: 1.8+ MB
```

[4]:
```
plt.boxplot(data['AveragePrice'], 0, 'rs', 0)
plt.title('AveragePrice boxplot')
plt.xlabel('Price($)')
plt.plot()
```

[4]: []



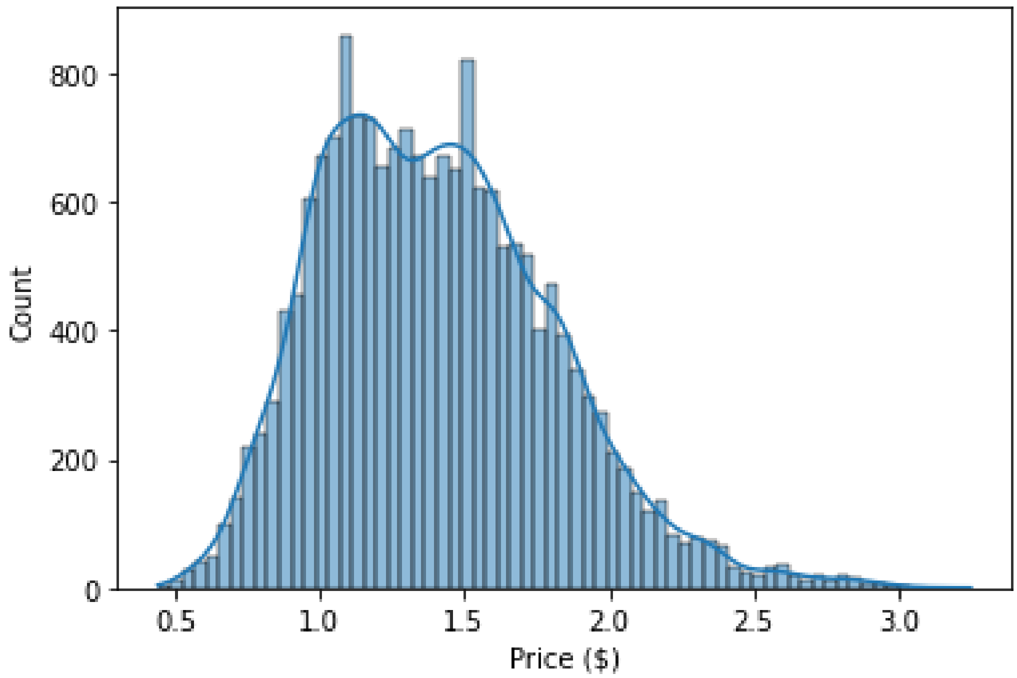
[5]:
```
ax4 = plt.subplot()
res = stats.probplot(data['AveragePrice'],dist ='norm',plot=plt)
```



Not a great fit for a normal distriibution.

[6]:
```
sns.histplot(data['AveragePrice'],kde=True)
plt.xlabel('Price ($)')
plt.plot()
```

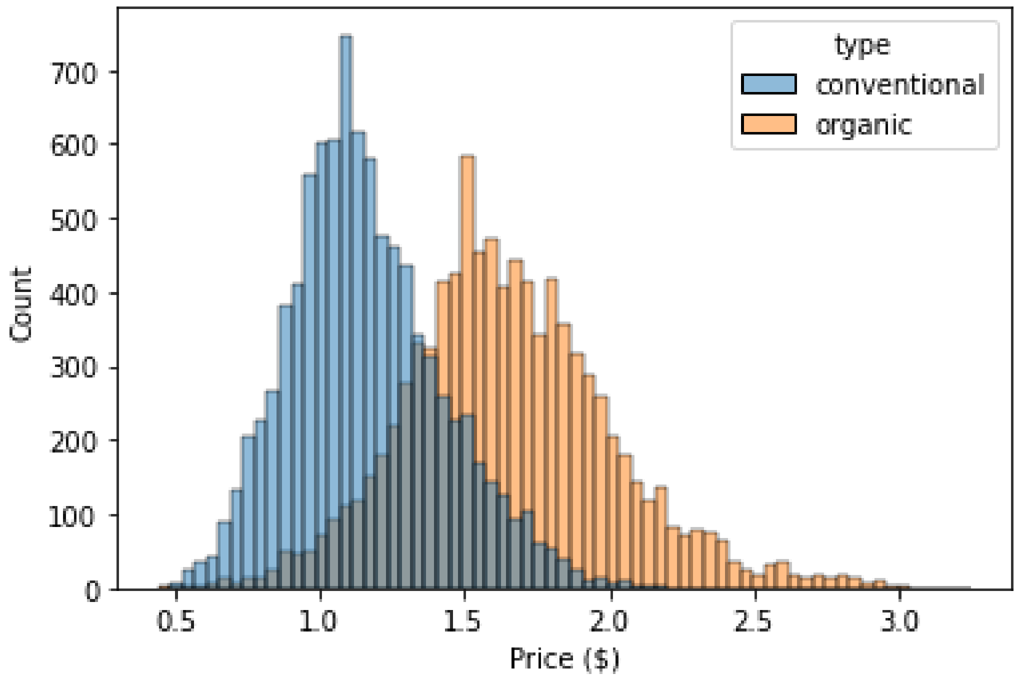
[6]: []



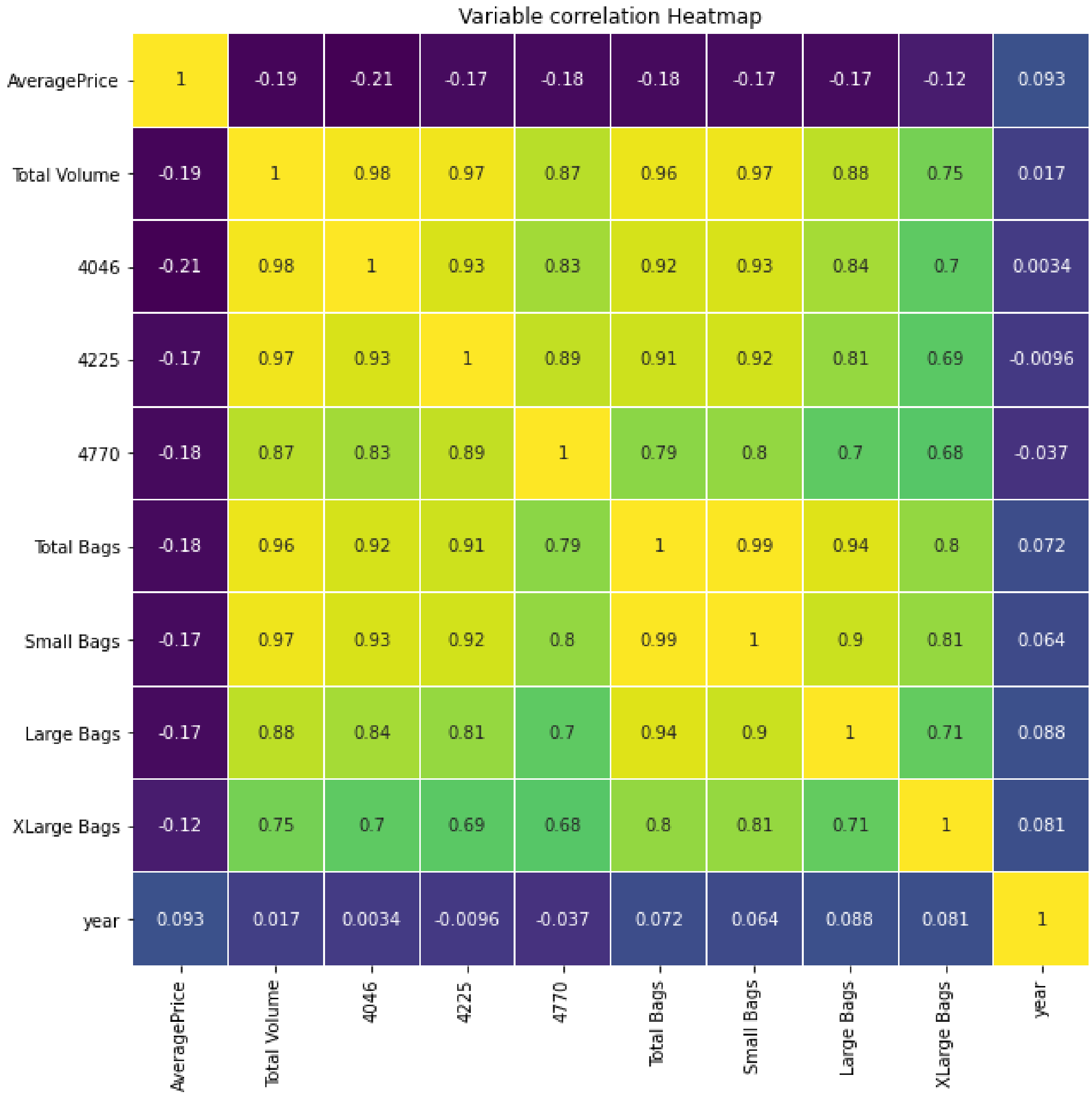
It appears to be a bimodal distribution, which is strange for a price of the same item you would expect it to be unimodal.

later in the analysis, I come to compare the variable 'type' which categorizes the avocados between organic and conventional. this explains the bimodal distribution, as the skews for organic and conv are different

[7]:
```
sns.histplot(data,x='AveragePrice',hue='type')
plt.xlabel('Price ($)')
plt.show()
```



[8]:
```
d_corr = data.corr()
fig,ax = plt.subplots(figsize=(10,10))
ax = sns.heatmap(d_corr,annot=True,linewidth=0.01,cbar=False,cmap='viridis')
ax.set_title('Variable correlation Heatmap')
plt.show()
```

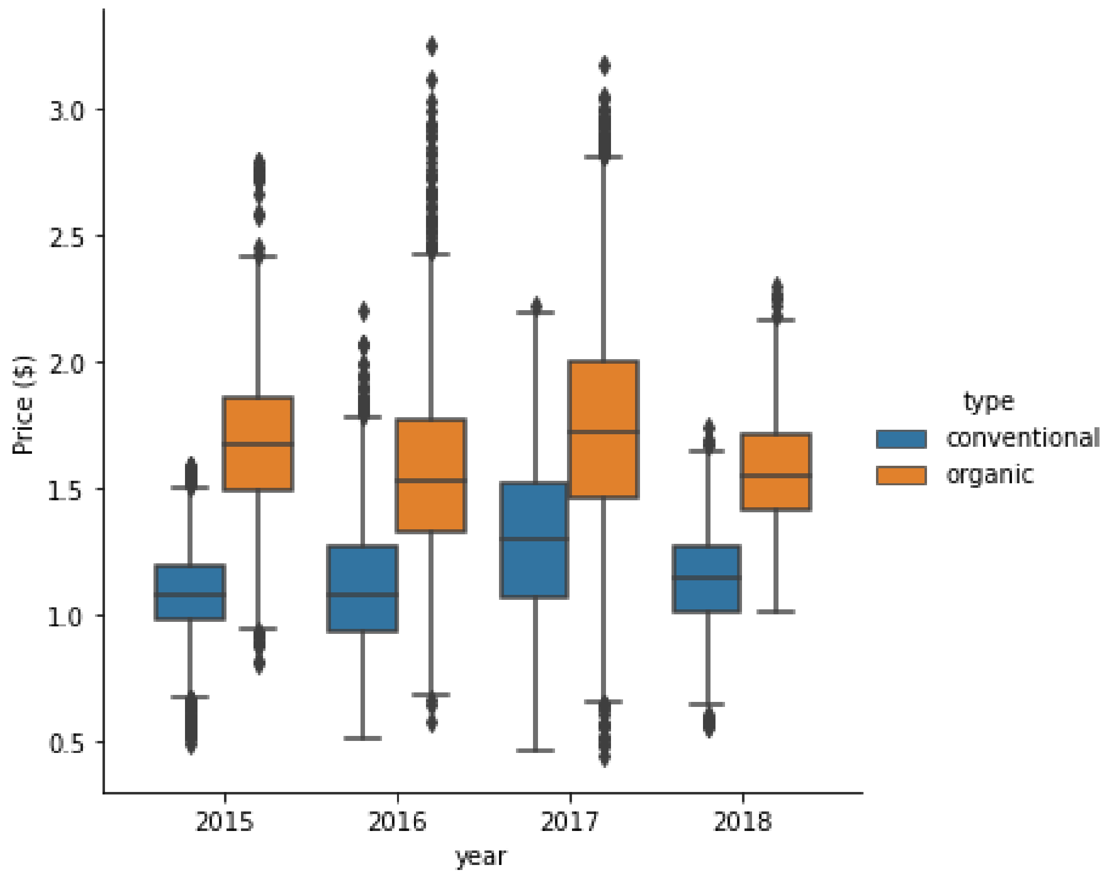


0.1 Categorical

0.1.1 Type organic / conventional

[9]:
```
sns.catplot(x ='year',y='AveragePrice', data=data,hue='type',kind='box')
plt.ylabel('Price ($)')
plt.plot()
```

[9]: []

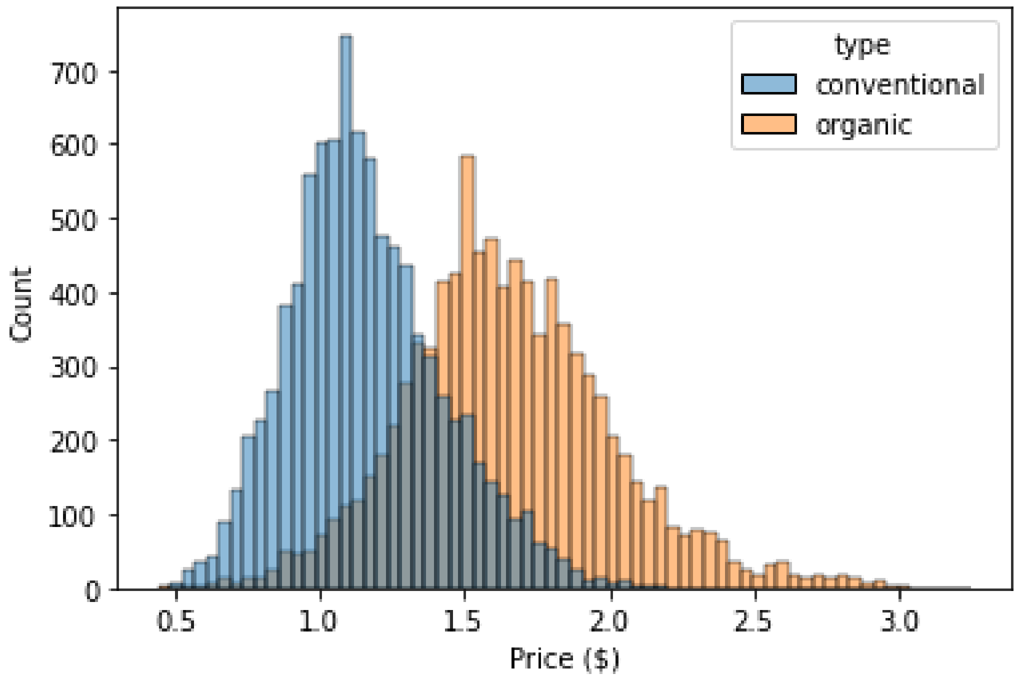


- Organic avocados tend to be more expensive that conventional avocados.

bernulli 1-organic, 0-conventional

[10]:
```
sns.histplot(data,x='AveragePrice',hue='type')
plt.xlabel('Price ($)')
plt.plot()
```

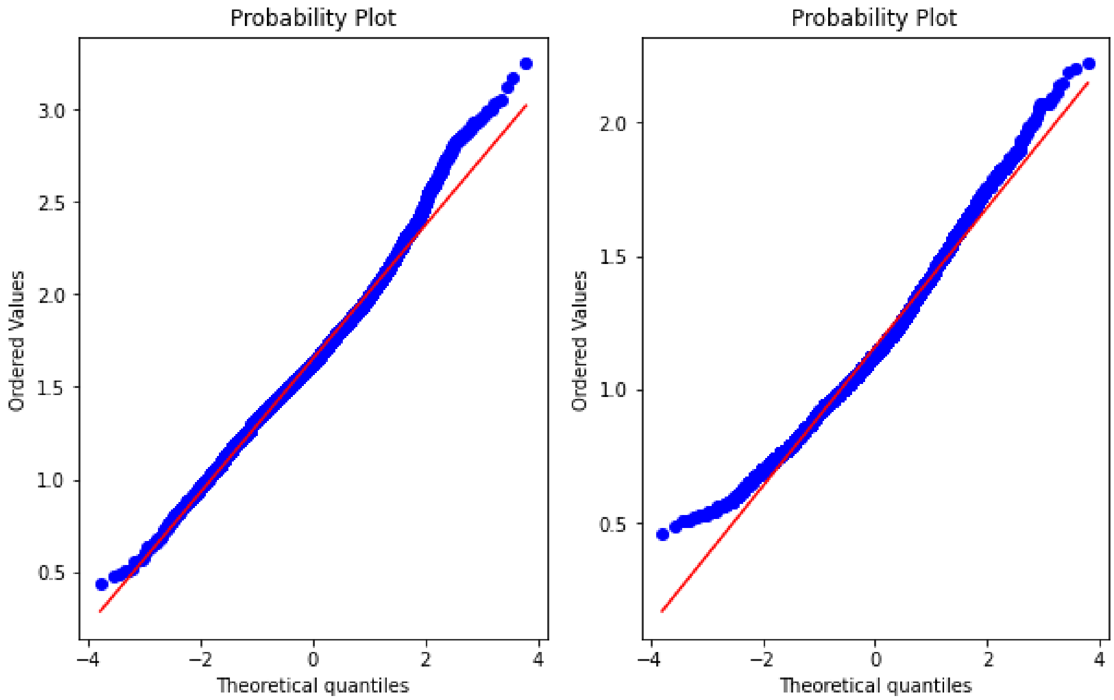
[10]: []



distribution fit

[11]:
```
f, ax = plt.subplots(1,2,figsize=(10,6))

stats.probplot(data.loc[data['type']=='organic','AveragePrice'],dist␣
↪='norm',plot=ax[0])
stats.probplot(data.loc[data['type']=='conventional','AveragePrice'],dist␣
↪='norm',plot=ax[1])
plt.plot()
```

[11]: []



the spread of values for organic is much larger. 4Q >3.0 compared to 4Q<3.0

[12]:
```
avo_type  = data.groupby('type').agg(['mean','std','count'])
avo_type = round(avo_type,2)
#mean diff
mean_diff = avo_type['AveragePrice']['mean']['organic']-avo_type['AveragePrice']['mean']['conventional'
#standard deviation
organic_s1 = avo_type['AveragePrice']['std']['organic']
conventional_s2 = avo_type['AveragePrice']['std']['conventional']
print('mean difference',mean_diff,'p')
```

```
mean difference 0.49 p
```

X_1 = Organic Avocados, AveragePrice dist.

X_2 = Conventional Avocados, AveragePrice dist.

$\bar{X}_1 - \bar{X}_2 = 0.49$

$\alpha = 0.01$

$$\sigma^2_{\bar{X}_1+\bar{X}_2} \approx \frac{S_1}{n_1} + \frac{S_2}{n_2}$$

```
[13]: sampling_stderr = (organic_s1/
       ↪avo_type['AveragePrice']['count']['organic'])+(conventional_s2/
       ↪avo_type['AveragePrice']['count']['conventional'])
      sampling_stderr = np.sqrt(sampling_stderr)
      print('sampling std error approx',sampling_stderr)

      crit_limit =2.33
      limit = crit_limit*sampling_stderr
      print(round(mean_diff - limit,2),'to',round(mean_diff +limit,2))
```

```
sampling std error approx 0.008243223411136655
0.47 to 0.51
```

Confident (That the true Mean difference in price between Organic avocados and conventional avocados is between 0.47p and 0.51p) $\approx 99\%$

0.2 Region

Groupby region, with lambda agg to reduce oulier prices.

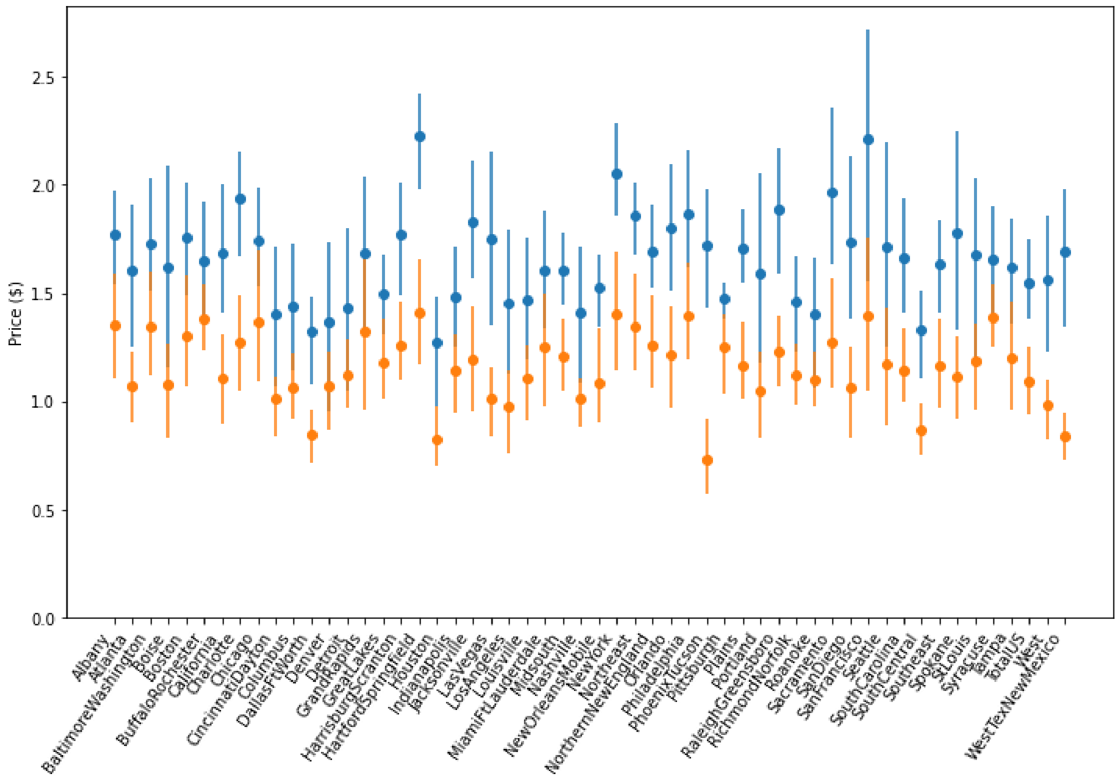
```
[14]: region_data = data.groupby(['region','type'])['AveragePrice'].agg(['mean',
                                                                    lambda x : np.
       ↪quantile(x,.15),
                                                                    lambda x : np.
       ↪quantile(x,.85)])
      region_data = region_data.unstack()
      region_data.rename(columns={'<lambda_0>':'q15','<lambda_1>':'q85'},inplace=True)
      err_pdata = region_data.copy()
      region_data.head()
```

```
[14]:                          mean                    q15                  q85  \
      type              conventional   organic conventional organic conventional
      region
      Albany                1.348757  1.773314        1.110    1.54        1.588
      Atlanta               1.068817  1.607101        0.902    1.25        1.230
      BaltimoreWashington   1.344201  1.724260        1.120    1.51        1.600
      Boise                 1.076036  1.620237        0.836    1.16        1.268
      Boston                1.304379  1.757396        1.070    1.49        1.580


      type                organic
      region
      Albany                1.970
      Atlanta               1.906
      BaltimoreWashington   2.028
      Boise                 2.088
      Boston                2.010
```

[15]:
```python
#reformating the quartile columns for the error bar plot
err_pdata['q15'] = err_pdata['mean'] - err_pdata['q15']
err_pdata['q85']= err_pdata['q85'] - err_pdata['mean']
err_pdata.head()
```

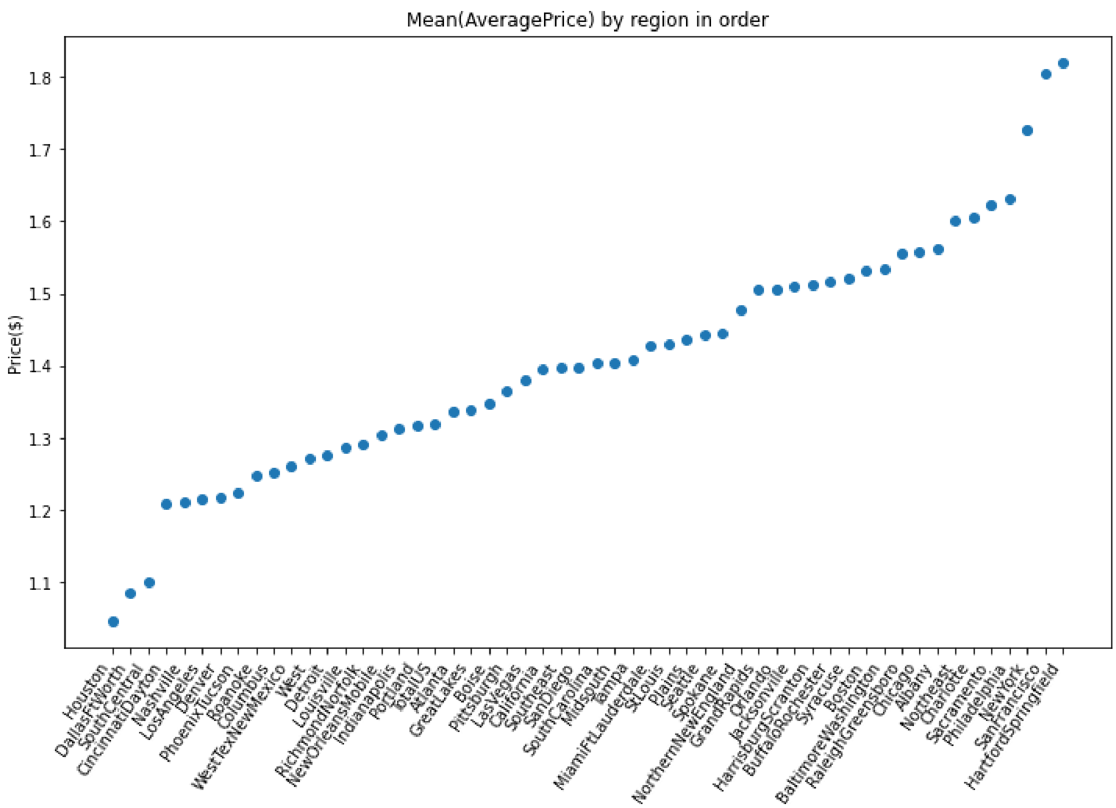
[15]:
```
                            mean                     q15               \
type                conventional   organic conventional   organic
region
Albany                  1.348757  1.773314     0.238757  0.233314
Atlanta                 1.068817  1.607101     0.166817  0.357101
BaltimoreWashington     1.344201  1.724260     0.224201  0.214260
Boise                   1.076036  1.620237     0.240036  0.460237
Boston                  1.304379  1.757396     0.234379  0.267396

                             q85
type                conventional   organic
region
Albany                  0.239243  0.196686
Atlanta                 0.161183  0.298899
BaltimoreWashington     0.255799  0.303740
Boise                   0.191964  0.467763
Boston                  0.275621  0.252604
```

[16]:
```python
fig,ax = plt.subplots(figsize=(12,8))
fig.autofmt_xdate(rotation=55 )
ax.errorbar(x = err_pdata.
 ↪index,y=err_pdata['mean']['organic'],yerr=[err_pdata['q15']['organic'],err_pdata['q85']['or
 ↪='o')
ax.errorbar(x = err_pdata.
 ↪index,y=err_pdata['mean']['conventional'],yerr=[err_pdata['q15']['conventional'],err_pdata[
 ↪='o')
ax.set_yticks([0,0.5,1,1.5,2,2.5])
ax.set_ylabel('Price ($)')
plt.show()
```



[17]:
```python
price_by_region = data.groupby('region')['AveragePrice'].agg(['mean'])
price_by_region = price_by_region.sort_values('mean')
price_by_region = price_by_region.reset_index()
fig,ax = plt.subplots(figsize = (12,8))
fig.autofmt_xdate(rotation=55 )
ax.scatter(x=price_by_region['region'],y=price_by_region['mean'])
ax.set_title('Mean(AveragePrice) by region in order')
ax.set_ylabel('Price($)')
plt.plot()
```

[17]: []



- Houston on average has the cheapest avocados
- Houston also has on average the cheapest organic avocados
- pitsburg on average has the cheapest Conventional avocados

Even though i think i will be a terrible fit, i want to fit a linear reggresion line to Price over year.

[18]:
```
houston = data[data['region']=='Houston']
h_organic = houston[houston['type']=='organic']
h_conv = houston[houston['type']=='conventional']
```

[19]:
```
sns.catplot(data = houston,x='year',y='AveragePrice',hue='type')
```

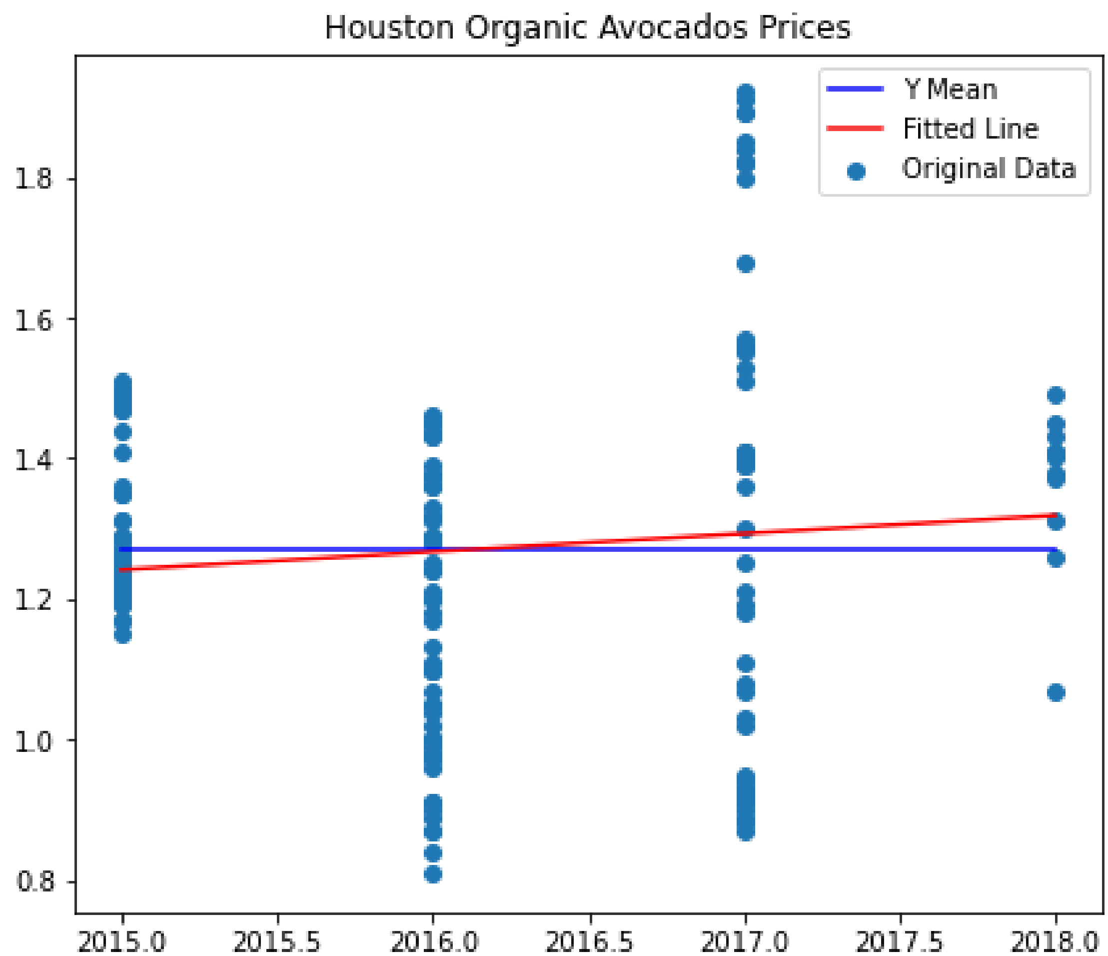
[19]: <seaborn.axisgrid.FacetGrid at 0x16ef9db6f08>

0.2.1 Organic

[20]:
```
slope_org, intercept_org, r_org, p_org, se_org =␣
↪linregress(h_organic['year'],h_organic['AveragePrice'])
```

[21]:
```
fig = plt.figure(figsize=(7,6))
plt.scatter(h_organic['year'],h_organic['AveragePrice'],label='Original Data')
plt.plot(h_organic['year'],h_organic['year'].apply(lambda val :
↪h_organic['AveragePrice'].mean()),label = 'Y Mean',c='b')
plt.plot(h_organic['year'],h_organic['year'].apply(lambda val : val*slope_org +␣
↪intercept_org),label='Fitted Line',c='r')
plt.title('Houston Organic Avocados Prices')
plt.legend()
plt.plot()
```

[21]: []



coefficient of determination

[22]:
```
h_organic = h_organic.copy()
h_organic.loc[:,'y'] = h_organic.loc[:,'AveragePrice']
h_organic.loc[:,'mx+b'] = h_organic.loc[:,'year'].apply(lambda val :␣
 ↪val*slope_org + intercept_org)
h_organic.loc[:,'SE_line']  = (h_organic.loc[:,'y'] - h_organic.loc[:
 ↪,'mx+b'])**2
```

[23]:
```
one_minusr2 = round((1- h_organic.loc[:,'SE_line'].sum()/((h_organic.loc[:
 ↪,'y']-h_organic.loc[:,'y'].mean())**2).sum()),2)
print(one_minusr2*100,'% of the total variation is described by the regression␣
 ↪line y=',round(slope_org,2),'x',round(intercept_org,2))
```

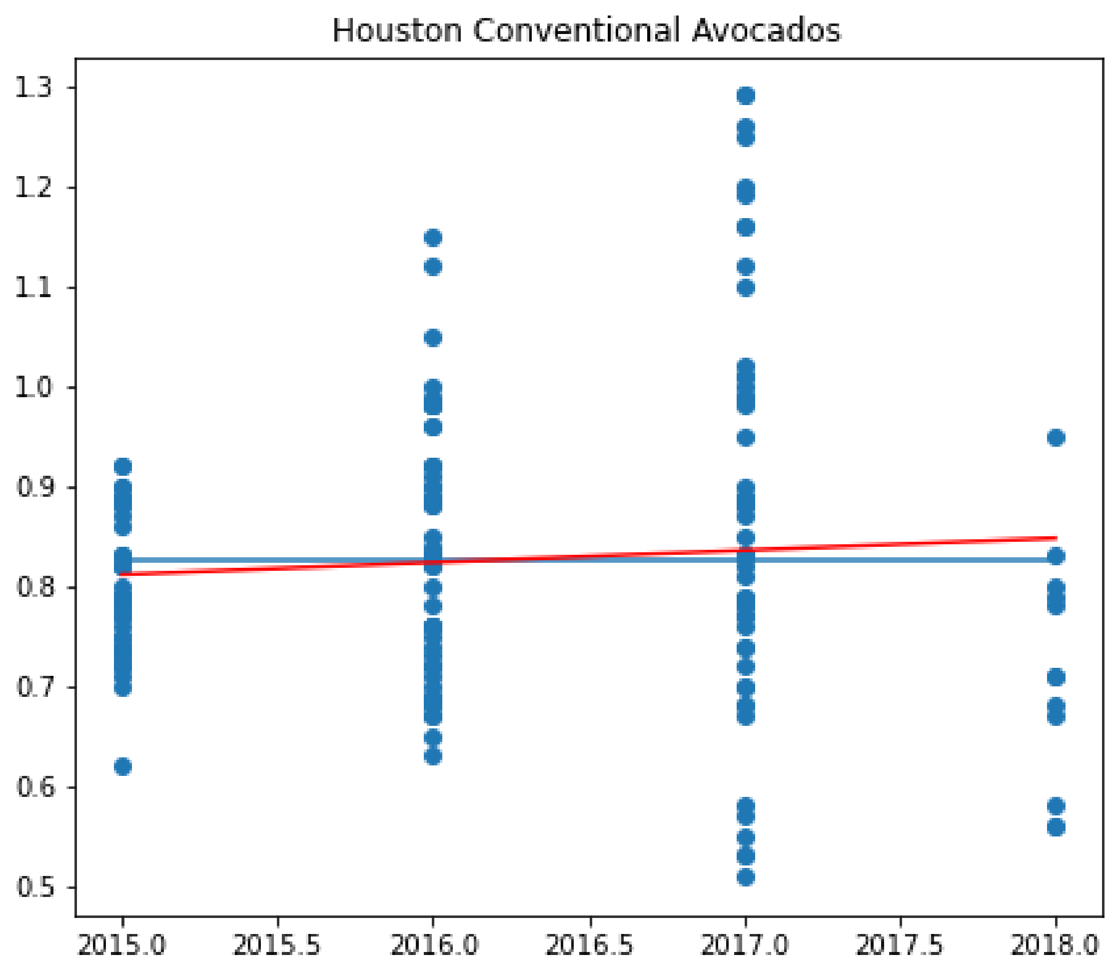
```
1.0 % of the total variation is described by the regression line y= 0.03 x
-50.59
```

0.2.2 Conventional

[24]:
```
slope_con, intercept_con, r, p, se =␣
↪linregress(h_conv['year'],h_conv['AveragePrice'])
```

[25]:
```
fig = plt.figure(figsize=(7,6))
plt.scatter(h_conv['year'],h_conv['AveragePrice'])
plt.plot(h_conv['year'],h_conv['year'].apply(lambda val :h_conv['AveragePrice'].
↪mean()))
plt.plot(h_conv['year'],h_conv['year'].apply(lambda val : val*slope_con +␣
↪intercept_con),c='r')
plt.title('Houston Conventional Avocados')
plt.plot()
```

[25]: []



coefficient of determination

[26]:
```
h_conv = h_conv.copy()
h_conv.loc[:,'y'] = h_conv.loc[:,'AveragePrice']
h_conv.loc[:,'mx+b'] = h_conv.loc[:,'year'].apply(lambda val : val*slope_con +␣
↪intercept_con)
h_conv.loc[:,'SE_line']  = (h_conv.loc[:,'y'] - h_conv.loc[:,'mx+b'])**2
```

[27]:
```
one_minusr2_con = round((1- h_conv.loc[:,'SE_line'].sum()/((h_conv.loc[:
↪,'y']-h_conv.loc[:,'y'].mean())**2).sum()),2)
print(one_minusr2_con*100,'% of the total variation is described by the␣
↪regression line y=',round(slope_con,2),'x',round(intercept_con,2))
```

```
1.0 % of the total variation is described by the regression line y= 0.01 x
-23.65
```

- Not surprisingly simple linear regression, least squared line has not produced a reliable predictor, non the the less i wanted to put it into practive and visualize it.

[28]:
```
#noramlizing
from sklearn import preprocessing
from sklearn.decomposition import PCA
from sklearn.linear_model import LinearRegression
```

[29]:
```
def standardize(column,int_= False):
    if int_ ==True:
        values = normalized[column].sort_values().unique()
    else:
        values = normalized[column].unique()

    for i,v in enumerate(values):
        normalized.loc[normalized[column]==v, column]=i
```

[30]:
```
normalized = data.copy()
```

[31]:
```
#type- binary
normalized.loc[normalized['type'] == 'conventional','type']=0
normalized.loc[normalized['type'] == 'organic','type']=1

#region -nominal
standardize('region')

#year - ordinal
standardize('year',True)

#Date - ordinal
standardize('Date',True)
```

[32]:
```
normalized.head()
```

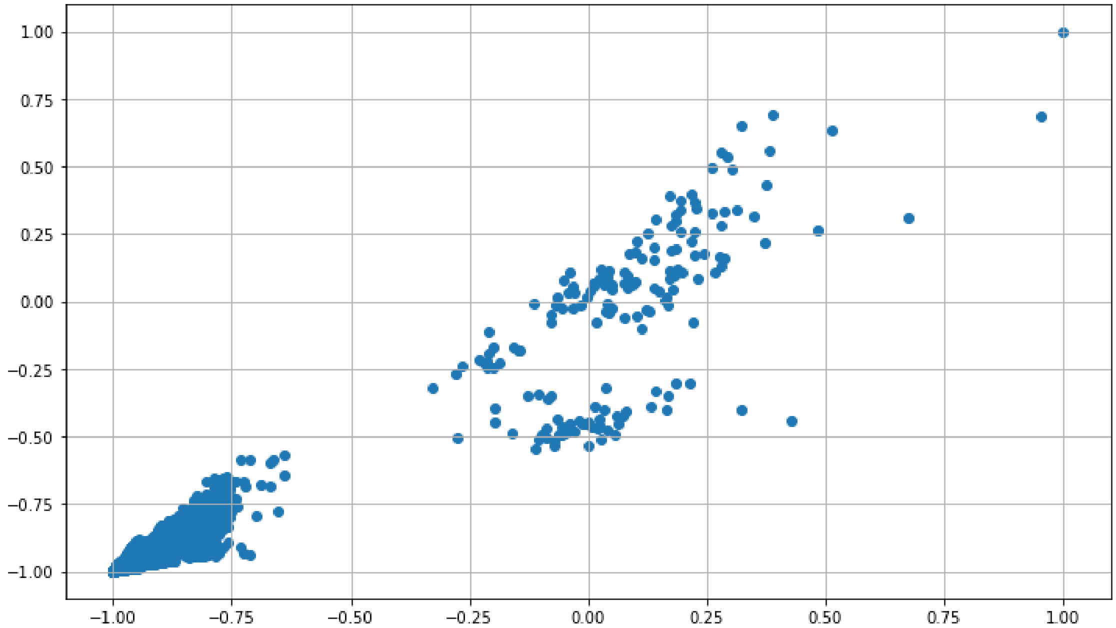
[32]:
```
   Date  AveragePrice  Total Volume     4046       4225    4770  Total Bags  \
0    51          1.33      64236.62  1036.74   54454.85   48.16     8696.87
1    50          1.35      54876.98   674.28   44638.81   58.33     9505.56
2    49          0.93     118220.22   794.70  109149.67  130.50     8145.35
3    48          1.08      78992.15  1132.00   71976.41   72.58     5811.16
4    47          1.28      51039.60   941.48   43838.39   75.78     6183.95

   Small Bags  Large Bags  XLarge Bags type  year region
0     8603.62       93.25          0.0    0     0      0
1     9408.07       97.49          0.0    0     0      0
2     8042.21      103.14          0.0    0     0      0
3     5677.40      133.76          0.0    0     0      0
4     5986.26      197.69          0.0    0     0      0
```

[33]:
```python
#dir(preprocessing)
```

[34]:
```python
scaler = preprocessing.MinMaxScaler(feature_range = (-1,1))
names = normalized.columns

d = scaler.fit_transform(normalized)

stand_data=pd.DataFrame(data=d, columns=names)
stand_data.head()
```

[34]:
```
       Date  AveragePrice  Total Volume      4046      4225      4770  \
0 -0.392857     -0.366548     -0.997947 -0.999909 -0.994680 -0.999962
1 -0.404762     -0.352313     -0.998247 -0.999941 -0.995639 -0.999954
2 -0.416667     -0.651246     -0.996220 -0.999930 -0.989336 -0.999898
3 -0.428571     -0.544484     -0.997475 -0.999900 -0.992968 -0.999943
4 -0.440476     -0.402135     -0.998370 -0.999917 -0.995717 -0.999940

   Total Bags  Small Bags  Large Bags  XLarge Bags  type  year  region
0   -0.999102   -0.998714   -0.999967         -1.0  -1.0  -1.0    -1.0
1   -0.999019   -0.998594   -0.999966         -1.0  -1.0  -1.0    -1.0
2   -0.999159   -0.998798   -0.999964         -1.0  -1.0  -1.0    -1.0
3   -0.999400   -0.999152   -0.999953         -1.0  -1.0  -1.0    -1.0
4   -0.999362   -0.999105   -0.999931         -1.0  -1.0  -1.0    -1.0
```

[35]:
```python
plt.figure(figsize=(12,7))
plt.scatter(stand_data['Total Volume'],stand_data['Total Bags'])
plt.grid()
```



[36]:

```
correlations = stand_data.corr()
correlations[(correlations>0.5) | (correlations <-0.5)]
```

[36]:

	Date	AveragePrice	Total Volume	4046	4225	\
Date	1.000000	NaN	NaN	NaN	NaN	
AveragePrice	NaN	1.000000	NaN	NaN	NaN	
Total Volume	NaN	NaN	1.000000	0.977863	0.974181	
4046	NaN	NaN	0.977863	1.000000	0.926110	
4225	NaN	NaN	0.974181	0.926110	1.000000	
4770	NaN	NaN	0.872202	0.833389	0.887855	
Total Bags	NaN	NaN	0.963047	0.920057	0.905787	
Small Bags	NaN	NaN	0.967238	0.925280	0.916031	
Large Bags	NaN	NaN	0.880640	0.838645	0.810015	
XLarge Bags	NaN	NaN	0.747157	0.699377	0.688809	
type	NaN	0.615845	NaN	NaN	NaN	
year	0.950274	NaN	NaN	NaN	NaN	
region	NaN	NaN	NaN	NaN	NaN	

	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	\
Date	NaN	NaN	NaN	NaN	NaN	
AveragePrice	NaN	NaN	NaN	NaN	NaN	
Total Volume	0.872202	0.963047	0.967238	0.880640	0.747157	
4046	0.833389	0.920057	0.925280	0.838645	0.699377	
4225	0.887855	0.905787	0.916031	0.810015	0.688809	
4770	1.000000	0.792314	0.802733	0.698471	0.679861	
Total Bags	0.792314	1.000000	0.994335	0.943009	0.804233	

```
Small Bags     0.802733    0.994335    1.000000    0.902589     0.806845
Large Bags     0.698471    0.943009    0.902589    1.000000     0.710858
XLarge Bags    0.679861    0.804233    0.806845    0.710858     1.000000
type                NaN         NaN         NaN         NaN          NaN
year                NaN         NaN         NaN         NaN          NaN
region              NaN         NaN         NaN         NaN          NaN

                   type      year  region
Date                NaN  0.950274     NaN
AveragePrice   0.615845       NaN     NaN
Total Volume        NaN       NaN     NaN
4046                NaN       NaN     NaN
4225                NaN       NaN     NaN
4770                NaN       NaN     NaN
Total Bags          NaN       NaN     NaN
Small Bags          NaN       NaN     NaN
Large Bags          NaN       NaN     NaN
XLarge Bags         NaN       NaN     NaN
type           1.000000       NaN     NaN
year                NaN  1.000000     NaN
region              NaN       NaN     1.0
```

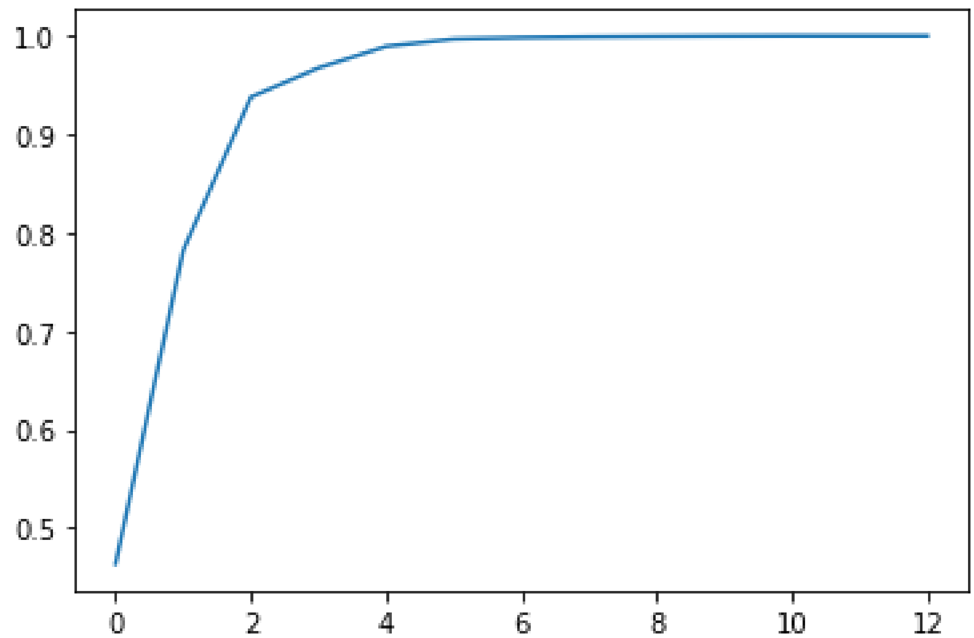
0.3 PCA

which attributes account for the least varation on our data, those will be the least nessicary

[62]:
```
pca = PCA(n_components=stand_data.shape[1])

pca_fitted = pca.fit(stand_data)
components = pca.transform(stand_data)
```

[63]:
```
#this is our ratio of variance explained by attributes.
var = pca_fitted.explained_variance_ratio_

#now we can visually see the redundant attributes
plt.plot(np.cumsum(var))
plt.plot()
```

[63]: []



[64]:
```
pca_c = ['pca_'+ str(i) for i in range(0,components.shape[1])]
pca_data = pd.DataFrame(components,columns=pca_c)
pca_data
```

[64]:

	pca_0	pca_1	pca_2	pca_3	pca_4	pca_5	pca_6
0	0.995166	-0.809407	-1.010630	-0.061394	0.191101	0.196340	-0.016465
1	0.992663	-0.816917	-1.010737	-0.063891	0.203148	0.184953	-0.017301
2	1.047656	-0.836696	-1.008749	-0.009316	-0.080734	0.234164	-0.012389
3	1.028121	-0.840589	-1.009538	-0.029344	0.018608	0.204874	-0.014889
4	1.002134	-0.843083	-1.010455	-0.054671	0.151833	0.168688	-0.017434
…	…	…	…	…	…	…	…
18244	-1.033556	1.511647	0.997426	-0.084335	-0.132351	-0.146628	0.027066
18245	-1.043858	1.505797	0.997079	-0.094067	-0.079813	-0.166299	0.026818
18246	-1.064532	1.502185	0.996460	-0.112975	0.026778	-0.196996	0.026047
18247	-1.072192	1.495792	0.996255	-0.119661	0.065945	-0.213902	0.025417
18248	-1.031736	1.479087	0.997513	-0.081675	-0.144471	-0.179802	0.026485

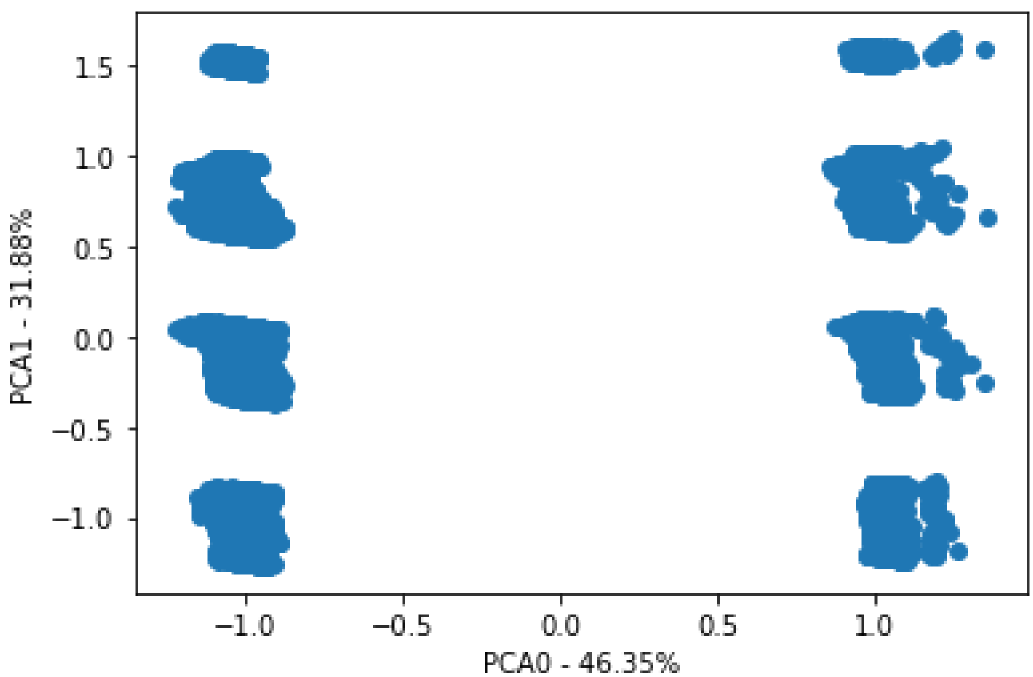
\

	pca_7	pca_8	pca_9	pca_10	pca_11	pca_12
0	-0.010422	0.000401	-0.003641	0.001072	7.966001e-08	1.284986e-10
1	-0.010468	0.000321	-0.004473	0.000530	8.228737e-08	1.237921e-10
2	-0.007996	0.000881	0.005166	0.004296	5.861508e-08	1.297201e-10
3	-0.008751	0.000582	0.000689	0.002713	7.046306e-08	1.237488e-10
4	-0.010013	0.000348	-0.003803	0.001082	8.515053e-08	1.129118e-10
…	…	…	…	…	…	…
18244	0.010713	-0.001993	0.003924	-0.000413	-2.549018e-07	6.141789e-11
18245	0.010031	-0.001929	0.002736	-0.000041	-2.453427e-07	5.232955e-11

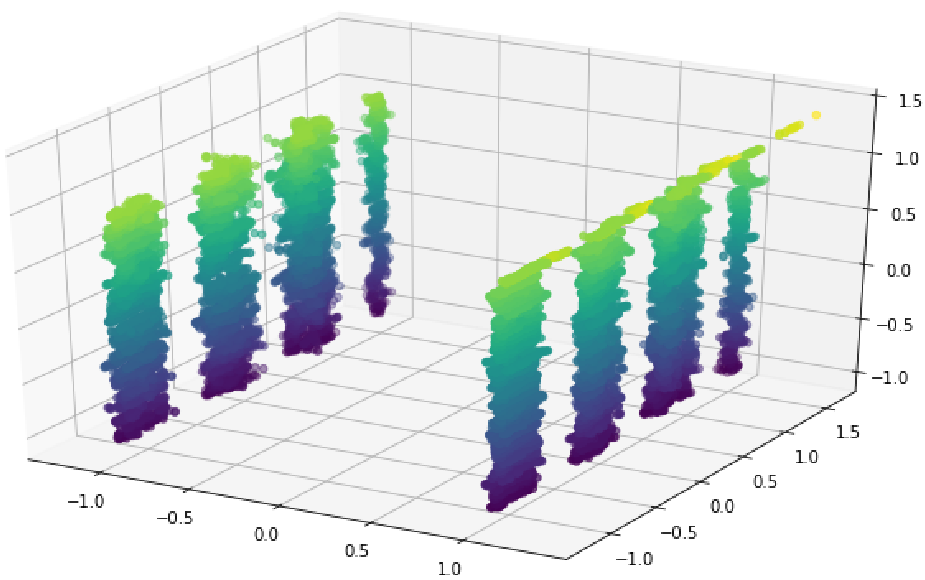
```
18246  0.008989 -0.001803  0.000255 -0.000658 -2.323529e-07  3.902631e-11
18247  0.008436 -0.001870 -0.000510 -0.000927 -2.265541e-07  3.037124e-11
18248  0.010710 -0.002090  0.003911 -0.000270 -2.466722e-07  4.163929e-11

[18249 rows x 13 columns]
```

[65]:
```
plt.scatter(pca_data.pca_0,pca_data.pca_1)
plt.xlabel('PCA0 - {0}%'.format(round(var[0]*100,2)))
plt.ylabel('PCA1 - {0}%'.format(round(var[1]*100,2)))
plt.plot()
```

[65]: []



[66]:
```
plt.figure(figsize=(12,7))

ax = plt.axes(projection='3d')
xdata = pca_data.pca_0
ydata = pca_data.pca_1
zdata = pca_data.pca_2
ax.scatter3D(xdata,ydata,zdata,c=zdata)
plt.show()
```


−1.0
−0.5
0.0
0.5
1.0
−1.0
−0.5
0.0
0.5
1.0
1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

```
[ ]:
```