customer_segments

February 3, 2016

1 Creating Customer Segments

In this project you, will analyze a dataset containing annual spending amounts for internal structure, to understand the variation in the different types of customers that a wholesale distributor interacts with.

Instructions:

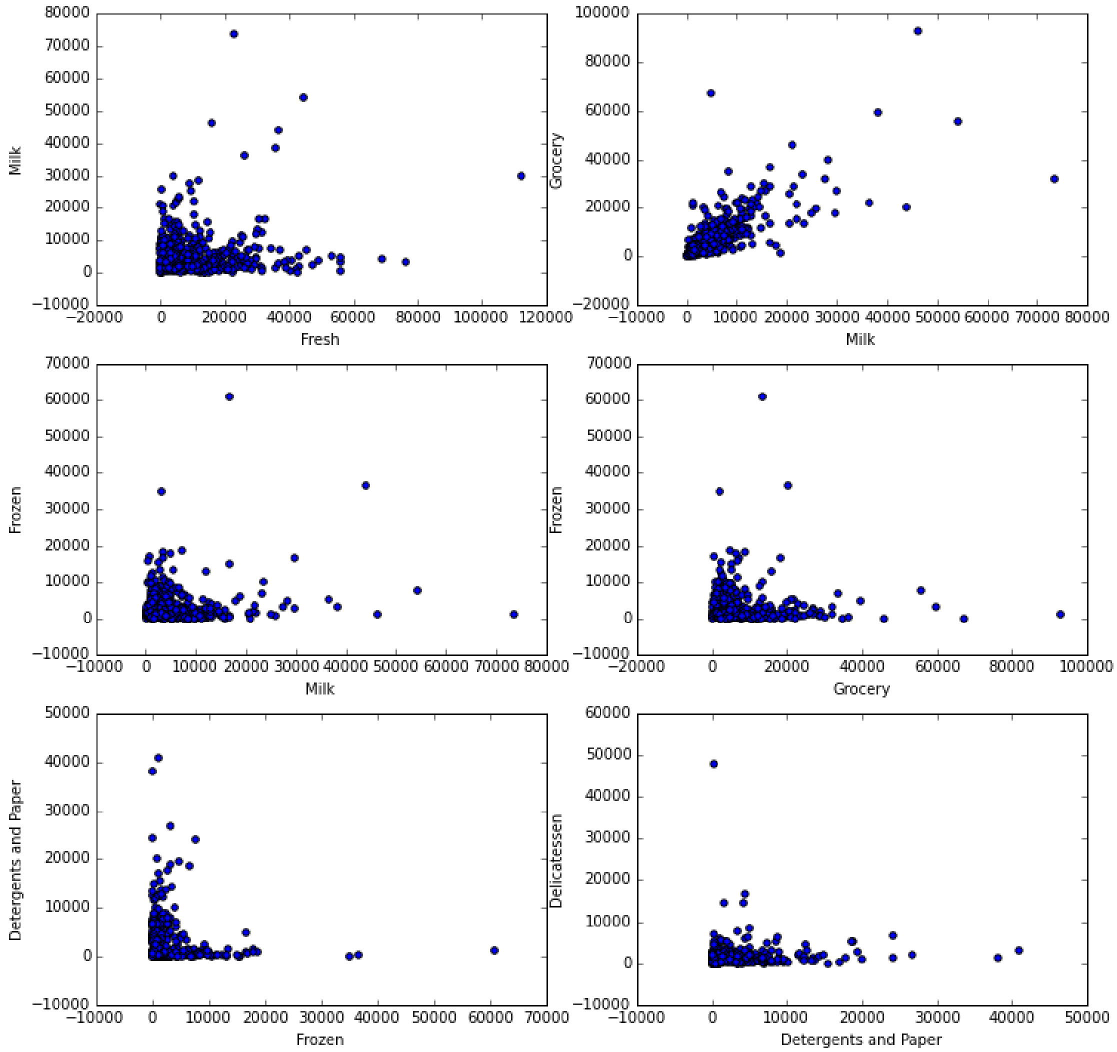
- Run each code block below by pressing **Shift+Enter**, making sure to implement any steps marked with a TODO.
- Answer each question in the space provided by editing the blocks labeled "Answer:".
- When you are done, submit the completed notebook (.ipynb) with all code blocks executed, as well as a .pdf version (File > Download as).

In [1]:
```
# Import libraries: NumPy, pandas, matplotlib
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Tell iPython to include plots inline in the notebook
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')

pd.options.display.float_format = '{:.5f}'.format
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)

# Read dataset
data = pd.read_csv("wholesale-customers.csv")
num_features=data.shape[1]
num_data_points=data.shape[0]
print "Dataset has {} rows, {} columns".format(num_data_points,num_features)
print data.head()  # print the first 5 rows
print data.describe()
```

```
Dataset has 440 rows, 6 columns
   Fresh  Milk  Grocery  Frozen  Detergents_Paper  Delicatessen
0  12669  9656     7561     214              2674          1338
1   7057  9810     9568    1762              3293          1776
2   6353  8808     7684    2405              3516          7844
3  13265  1196     4221    6404               507          1788
4  22615  5410     7198    3915              1777          5185
             Fresh        Milk     Grocery      Frozen  Detergents_Paper  Delicatessen
count    440.00000   440.00000   440.00000   440.00000         440.00000     440.00000
```

```
mean    12000.29773  5796.26591  7951.27727  3071.93182        2881.49318    1524.87045
std     12647.32887  7380.37717  9503.16283  4854.67333        4767.85445    2820.10594
min         3.00000    55.00000     3.00000    25.00000           3.00000       3.00000
25%      3127.75000  1533.00000  2153.00000   742.25000         256.75000     408.25000
50%      8504.00000  3627.00000  4755.50000  1526.00000         816.50000     965.50000
75%     16933.75000  7190.25000 10655.75000  3554.25000        3922.00000    1820.25000
max    112151.00000 73498.00000 92780.00000 60869.00000       40827.00000   47943.00000
```

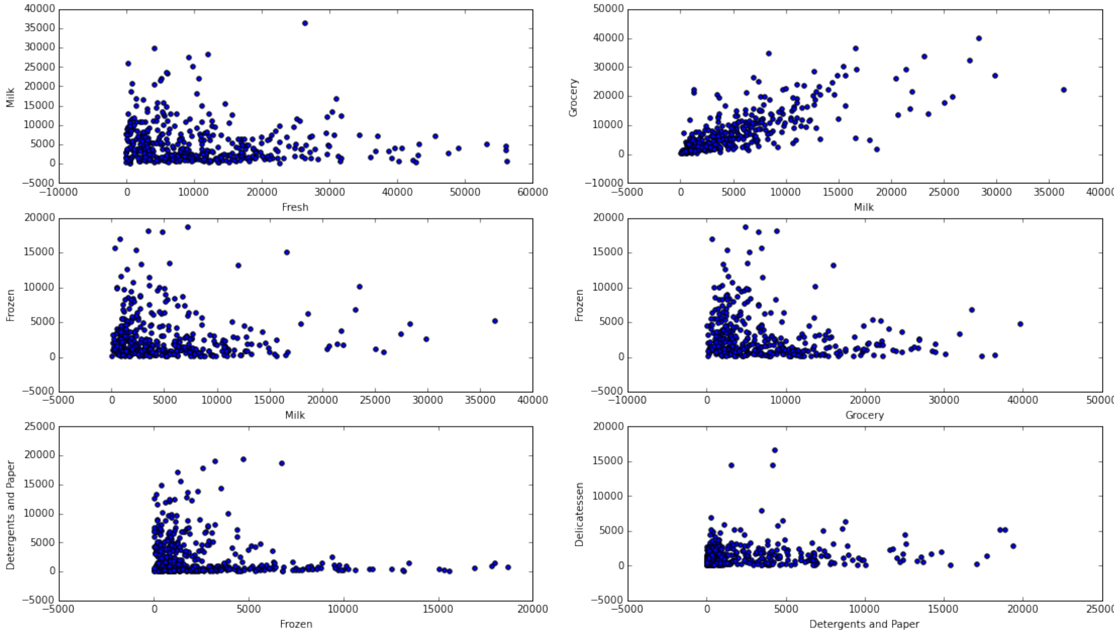
```
In [2]: ## Cleanup data, remove outliers.

        f, ((ax1, ax2), (ax3, ax4), (ax5, ax6)) = plt.subplots(3, 2)
        ax1.scatter(data.iloc[:,0],data.iloc[:,1])
        ax1.set_xlabel('Fresh')
        ax1.set_ylabel('Milk')

        ax2.scatter(data.iloc[:,1],data.iloc[:,2])
        ax2.set_xlabel('Milk')
        ax2.set_ylabel('Grocery')


        ax3.scatter(data.iloc[:,1],data.iloc[:,3])
        ax3.set_xlabel('Milk')
        ax3.set_ylabel('Frozen')

        ax4.scatter(data.iloc[:,2],data.iloc[:,3])
        ax4.set_xlabel('Grocery')
        ax4.set_ylabel('Frozen')


        ax5.scatter(data.iloc[:,3],data.iloc[:,4])
        ax5.set_xlabel('Frozen')
        ax5.set_ylabel('Detergents and Paper')


        ax6.scatter(data.iloc[:,4],data.iloc[:,5])
        ax6.set_xlabel('Detergents and Paper')
        ax6.set_ylabel('Delicatessen')

        fig = plt.gcf()
        fig.set_size_inches(12, 12)
        #fig.set_size_inches(18.5, 10.5, forward=True)
        plt.show()
```



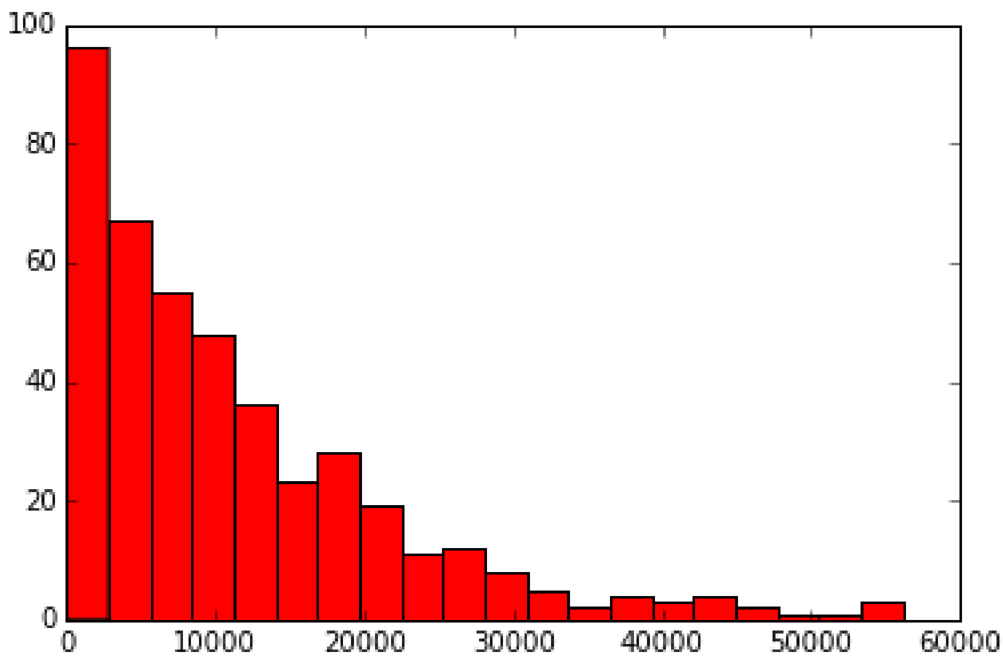
In [3]:
```
cleaned_data=data.copy(deep=True)
print '-'*100
print " After cleanup"
print '-'*100
cleaned_data=cleaned_data[cleaned_data['Fresh']<60000]
cleaned_data=cleaned_data[cleaned_data['Milk']<50000]
cleaned_data=cleaned_data[cleaned_data['Grocery']<50000]
cleaned_data=cleaned_data[cleaned_data['Frozen']<30000]
cleaned_data=cleaned_data[cleaned_data['Detergents_Paper']<20000]
cleaned_data=cleaned_data[cleaned_data['Delicatessen']<20000]

#from sklearn import preprocessing
#cleaned_data[['Fresh','Milk','Grocery','Frozen','Detergents_Paper','Delicatessen']] = cleaned_

f, ((ax1, ax2), (ax3, ax4), (ax5, ax6)) = plt.subplots(3, 2)
ax1.scatter(cleaned_data.iloc[:,0],cleaned_data.iloc[:,1])
ax1.set_xlabel('Fresh')
```

```
ax1.set_ylabel('Milk')

ax2.scatter(cleaned_data.iloc[:,1],cleaned_data.iloc[:,2])
ax2.set_xlabel('Milk')
ax2.set_ylabel('Grocery')

ax3.scatter(cleaned_data.iloc[:,1],cleaned_data.iloc[:,3])
ax3.set_xlabel('Milk')
ax3.set_ylabel('Frozen')

ax4.scatter(cleaned_data.iloc[:,2],cleaned_data.iloc[:,3])
ax4.set_xlabel('Grocery')
ax4.set_ylabel('Frozen')


ax5.scatter(cleaned_data.iloc[:,3],cleaned_data.iloc[:,4])
ax5.set_xlabel('Frozen')
ax5.set_ylabel('Detergents and Paper')


ax6.scatter(cleaned_data.iloc[:,4],cleaned_data.iloc[:,5])
ax6.set_xlabel('Detergents and Paper')
ax6.set_ylabel('Delicatessen')

fig = plt.gcf()
fig.set_size_inches(12, 12)
fig.set_size_inches(18.5, 10.5, forward=True)
plt.show()

f, ((axis1, axis2), (axis3, axis4), (axis5, axis6)) = plt.subplots(3, 2)
### Visualize data spread.
colormap = np.array(['r', 'g', 'b','c','m','y'])
for i in range(cleaned_data.shape[1]):
    print cleaned_data.columns[i]
    plt.hist(cleaned_data.iloc[:,i],bins=20,color=colormap[i])
    plt.show()
```

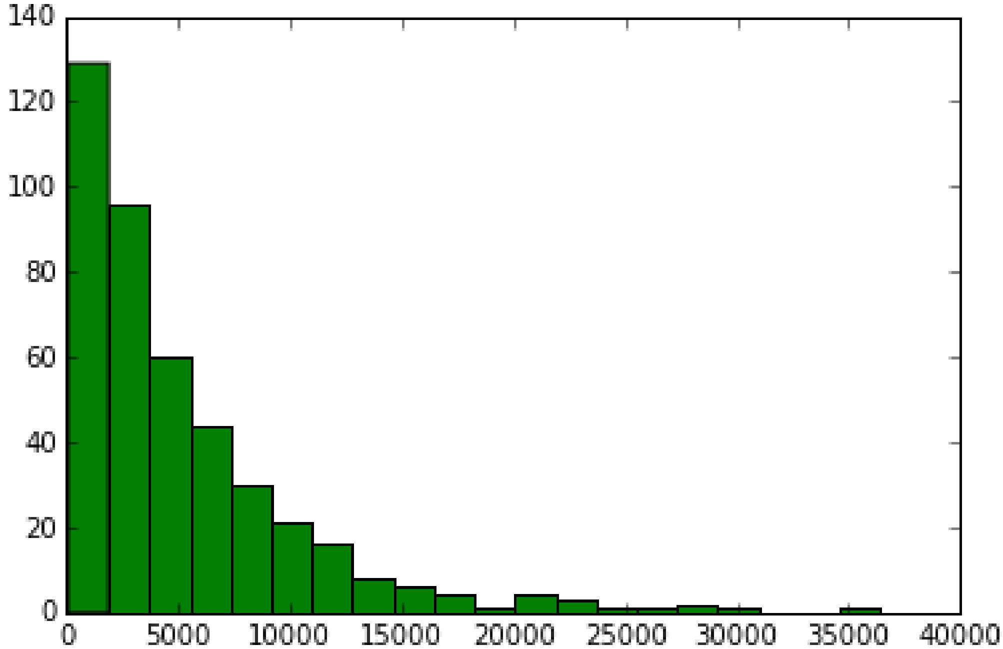
```
-------------------------------------------------------------------------------------------------
After cleanup
-------------------------------------------------------------------------------------------------
```


Milk
Fresh
Grocery
Milk
Frozen
Milk
Frozen
Grocery
Detergents and Paper
Frozen
Delicatessen
Detergents and Paper

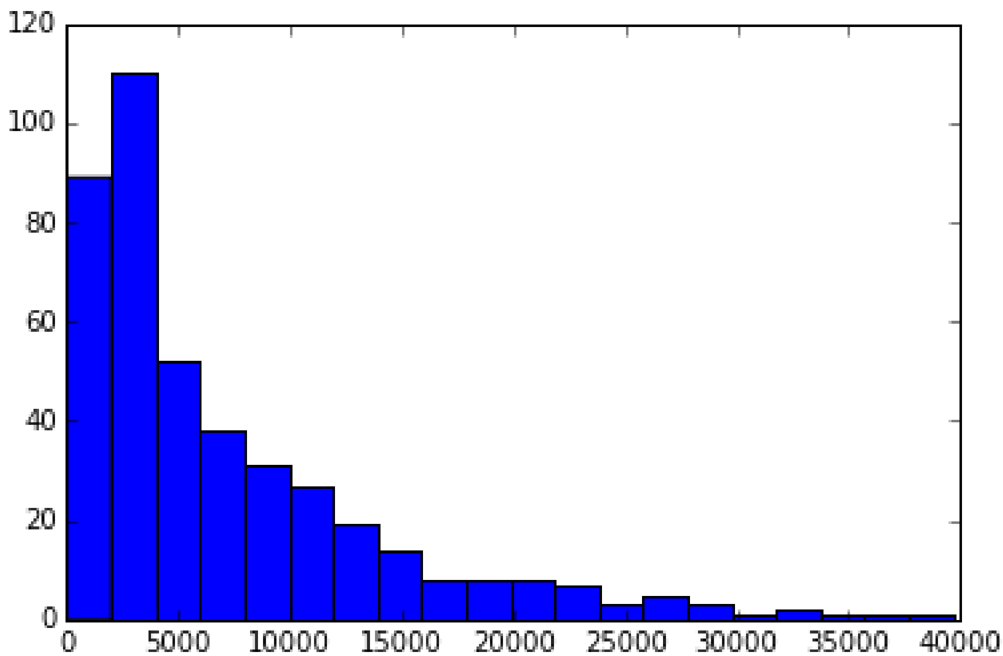
Fresh



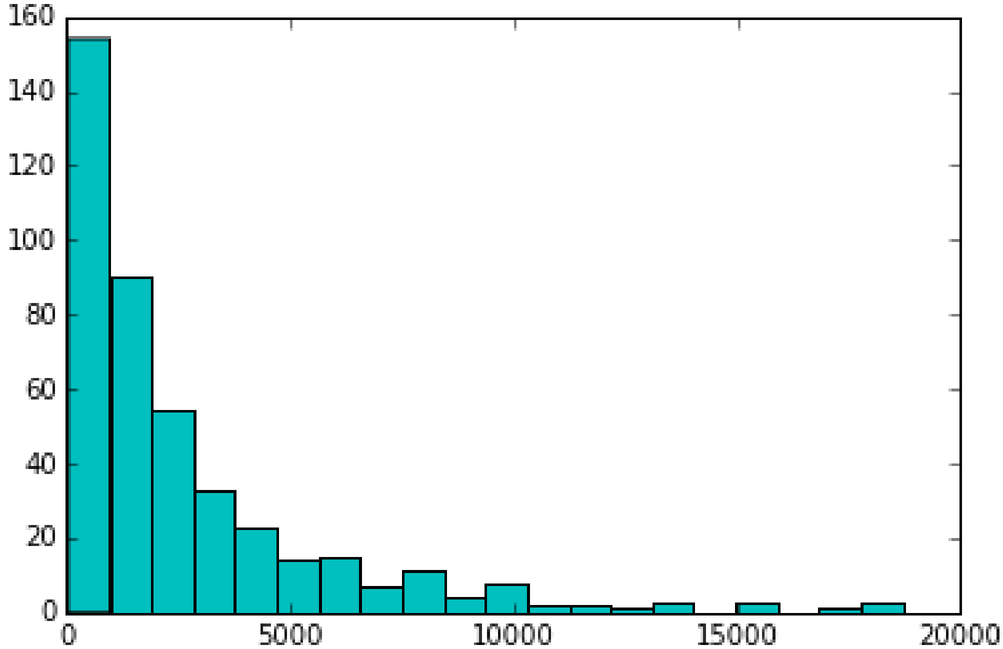
Milk



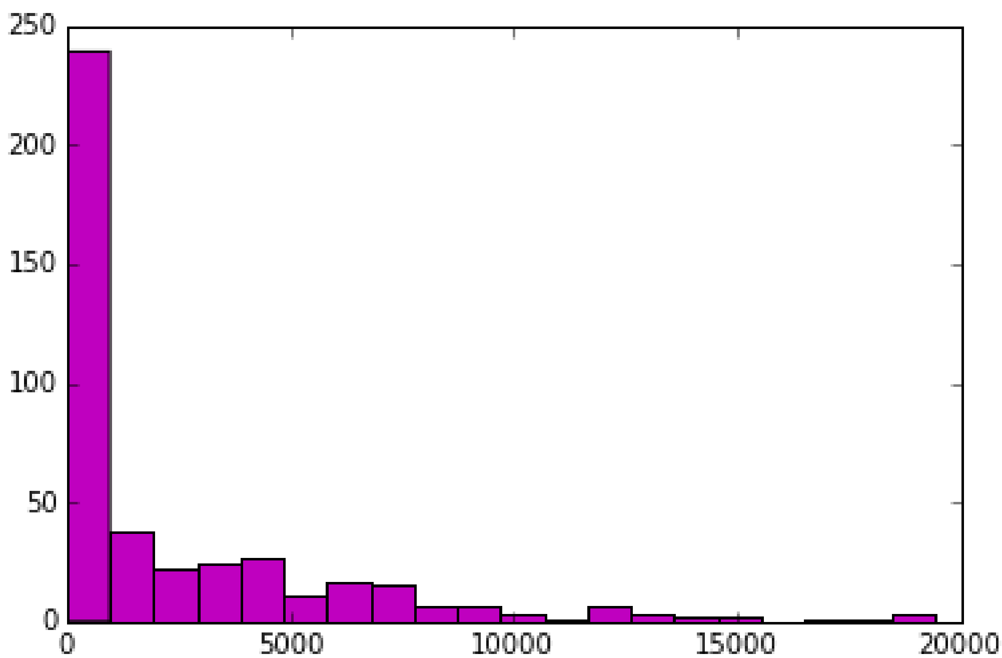
Grocery



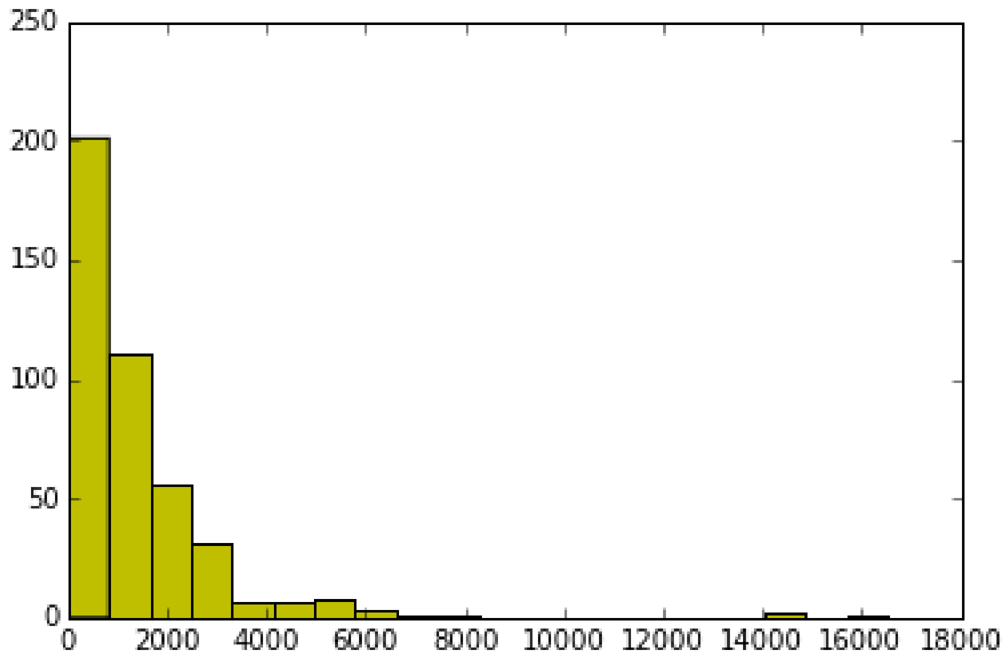
Frozen



Detergents_Paper



Delicatessen



1.1 Feature Transformation

1) In this section you will be using PCA and ICA to start to understand the structure of the data. Before doing any computations, what do you think will show up in your computations? List one or two ideas for what might show up as the first PCA dimensions, or what type of vectors will show up as ICA dimensions.

Answer:

Idea 1. Based on data spread, first PCA would be either fresh, or it could be combination of milk and groceries. Second PCA could include Frozen and Detergent_Paper, and Third PCA could be delicatessen.

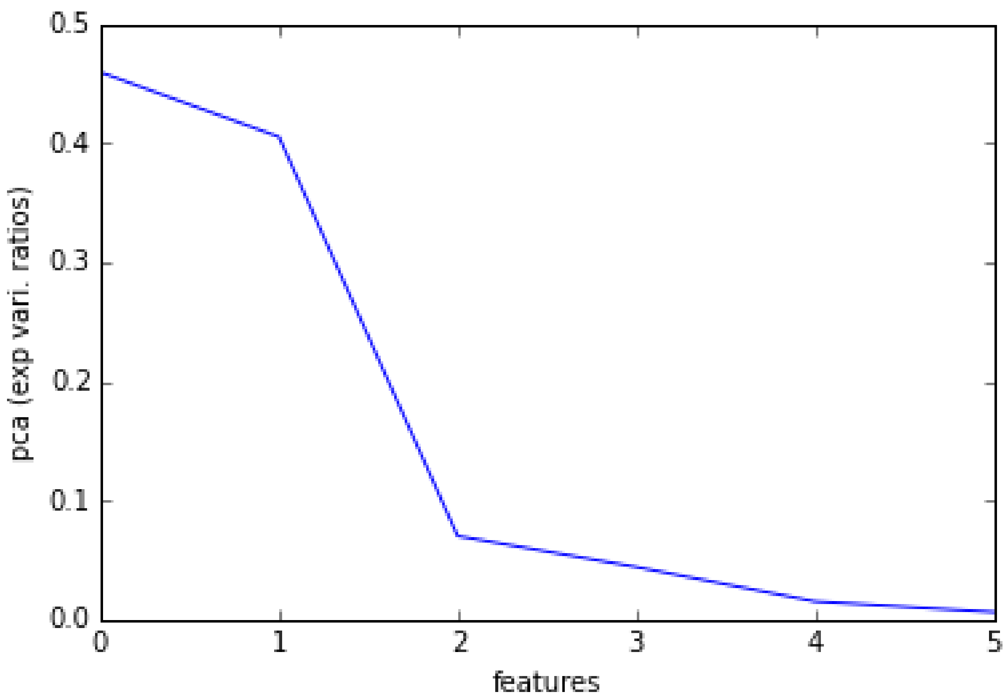
Idea 2. ICA could identify perishability as the differentiator in consumables / non-Delicatessen.

1.1.1 PCA

```
In [4]: # TODO: Apply PCA with the same number of dimensions as variables in the dataset
        # Using original data
        from sklearn.decomposition import PCA
        pca = PCA(n_components=num_features,whiten=True)
        pca.fit(data)

        # Print the components and the amount of variance in the data contained in each dimension
        print data.columns.values
        print pca.components_
        print pca.explained_variance_ratio_
        print pca.noise_variance_

        print "Visualizing all features"
        pc_df=pd.DataFrame({"pca":pca.explained_variance_ratio_})
        plt.plot(pc_df)
        plt.ylabel('pca (exp vari. ratios)')
```

```
        plt.xlabel('features')
        plt.show()
```

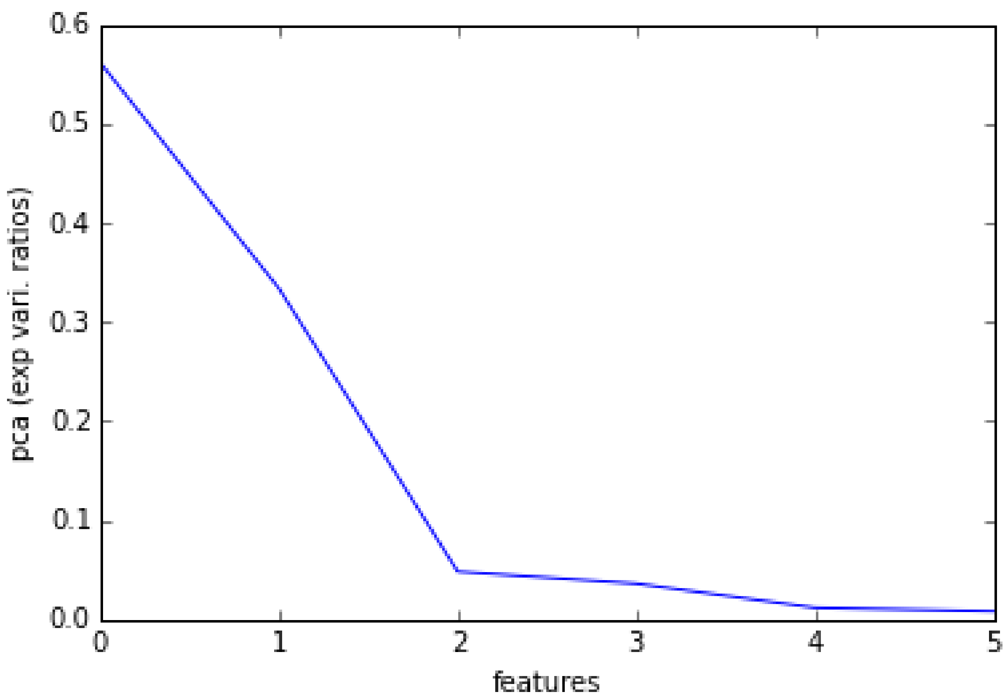
```
['Fresh' 'Milk' 'Grocery' 'Frozen' 'Detergents_Paper' 'Delicatessen']
[[-0.97653685 -0.12118407 -0.06154039 -0.15236462  0.00705417 -0.06810471]
 [-0.11061386  0.51580216  0.76460638 -0.01872345  0.36535076  0.05707921]
 [-0.17855726  0.50988675 -0.27578088  0.71420037 -0.20440987  0.28321747]
 [-0.04187648 -0.64564047  0.37546049  0.64629232  0.14938013 -0.02039579]
 [ 0.015986    0.20323566 -0.1602915   0.22018612  0.20793016 -0.91707659]
 [-0.01576316  0.03349187  0.41093894 -0.01328898 -0.87128428 -0.26541687]]
[ 0.45961362  0.40517227  0.07003008  0.04402344  0.01502212  0.00613848]
0.0
Visualizing all features
```




```
In [5]: # TODO: Apply PCA with the same number of dimensions as variables in the dataset
        # Using cleaned up data

        from sklearn.decomposition import PCA
        pca = PCA(n_components=num_features,whiten=True)
        pca.fit(cleaned_data)

        # Print the components and the amount of variance in the data contained in each dimension
        print cleaned_data.columns.values
        print pca.components_
        print pca.explained_variance_ratio_
        print pca.noise_variance_

        print "Visualizing all features"
```

```
pc_df=pd.DataFrame({"pca":pca.explained_variance_ratio_})
plt.plot(pc_df)
plt.ylabel('pca (exp vari. ratios)')
plt.xlabel('features')
plt.show()
```

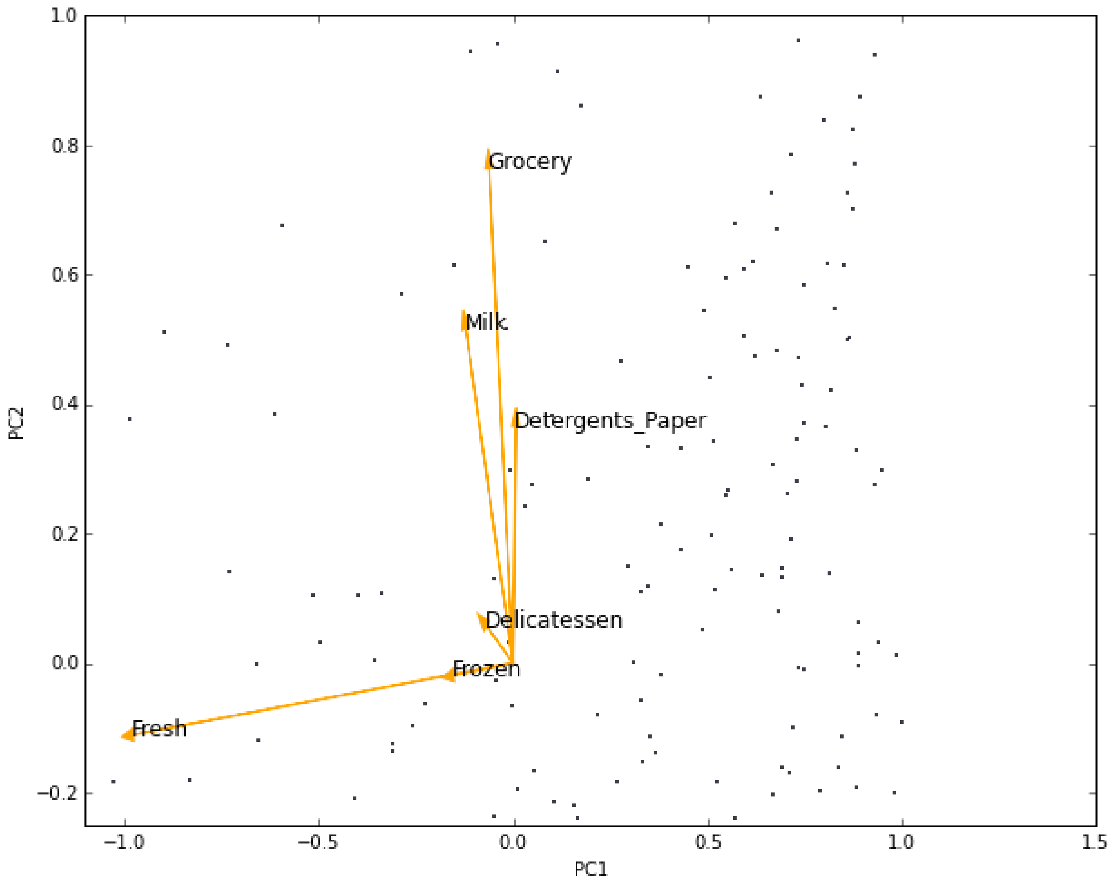
```
['Fresh' 'Milk' 'Grocery' 'Frozen' 'Detergents_Paper' 'Delicatessen']
[[-0.93484845  0.15350373  0.26484215 -0.09783119  0.14979567 -0.01854519]
 [ 0.33727454  0.49357837  0.72998125  0.00529907  0.32169172  0.0789937 ]
 [-0.09132697  0.61832533 -0.30822162  0.6719085  -0.2013167   0.14947626]
 [ 0.048662    0.57071797 -0.34542213 -0.72864576 -0.14113319  0.041873  ]
 [ 0.03907126  0.12678591 -0.31604449  0.08770818  0.64625958 -0.67614383]
 [-0.00831154 -0.09325834 -0.28772361 -0.01796047  0.62926724  0.71564593]]
[ 0.56148278  0.33367526  0.04836925  0.03623325  0.01188166  0.0083578 ]
0.0
Visualizing all features
```

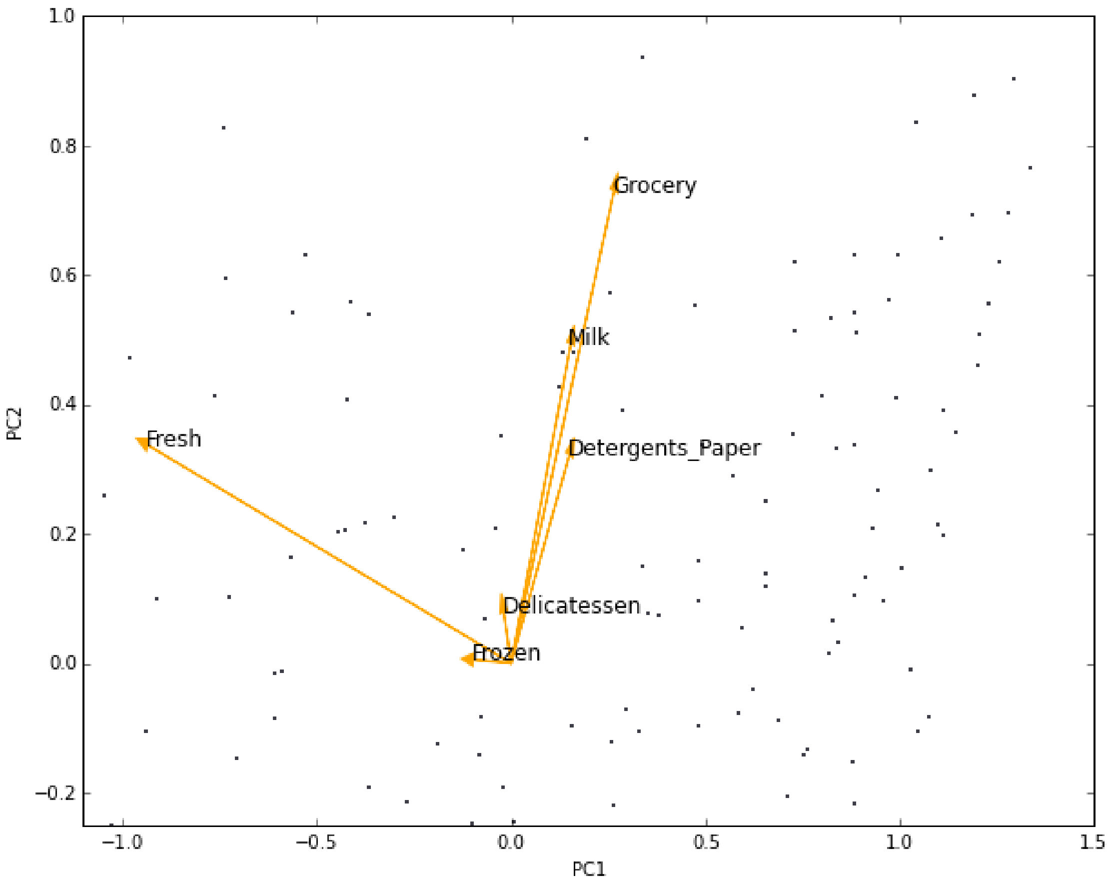


In [6]:
```
''' Following function has been taken from Udacity Forum:
https://discussions.udacity.com/t/
having-trouble-with-pca-and-ica-specifically-with-explaining-what-the-dimensions-mean/41890/12
'''

def biplot(df):
    # Fit on 2 components
    pca = PCA(n_components=2, whiten=True).fit(df)

    # Plot transformed/projected data
    ax = pd.DataFrame(
```

```
        pca.transform(df),
        columns=['PC1', 'PC2']
    ).plot(kind='scatter', x='PC1', y='PC2', figsize=(10, 8), s=0.8)

    # Plot arrows and labels
    for i, (pc1, pc2) in enumerate(zip(pca.components_[0], pca.components_[1])):
        ax.arrow(0, 0, pc1, pc2, width=0.001, fc='orange', ec='orange')
        ax.annotate(df.columns[i], (pc1, pc2), size=12)

    return ax

ax = biplot(data)
# Play around with the ranges for scaling the plot
ax.set_xlim([-1.1, 1.5])
ax.set_ylim([-.25, 1])


ax = biplot(cleaned_data)
# Play around with the ranges for scaling the plot
ax.set_xlim([-1.1, 1.5])
ax.set_ylim([-.25, 1])
```

Out[6]: (-0.25, 1)





2) How quickly does the variance drop off by dimension? If you were to use PCA on this dataset, how many dimensions would you choose for your analysis? Why?

Answer:

3) What do the dimensions seem to represent? How can you use this information?

Answer:

1.1.2 ICA

In [7]:
```
# TODO: Fit an ICA model to the data
# Note: Adjust the data to have center at the origin first!
from sklearn.decomposition import FastICA
ica = ?

# Print the independent components
print ica.components_
```

```
  File "<ipython-input-7-956165f9398b>", line 4
ica = ?
      ^
SyntaxError: invalid syntax
```

4) For each vector in the ICA decomposition, write a sentence or two explaining what sort of object or property it corresponds to. What could these components be used for?
Answer:

1.2 Clustering

In this section you will choose either K Means clustering or Gaussian Mixed Models clustering, which implements expectation-maximization. Then you will sample elements from the clusters to understand their significance.

1.2.1 Choose a Cluster Type

5) What are the advantages of using K Means clustering or Gaussian Mixture Models?
Answer:
6) Below is some starter code to help you visualize some cluster data. The visualization is based on this demo from the sklearn documentation.

In [ ]:
```
# Import clustering modules
from sklearn.cluster import KMeans
from sklearn.mixture import GMM
```

In [ ]:
```
# TODO: First we reduce the data to two dimensions using PCA to capture variation
reduced_data = ?
print reduced_data[:10]  # print upto 10 elements
```

In [ ]:
```
# TODO: Implement your clustering algorithm here, and fit it to the reduced data for visualizat
# The visualizer below assumes your clustering object is named 'clusters'

clusters = ?
print clusters
```

In [ ]:
```
# Plot the decision boundary by building a mesh grid to populate a graph.
x_min, x_max = reduced_data[:, 0].min() - 1, reduced_data[:, 0].max() + 1
y_min, y_max = reduced_data[:, 1].min() - 1, reduced_data[:, 1].max() + 1
hx = (x_max-x_min)/1000.
hy = (y_max-y_min)/1000.
xx, yy = np.meshgrid(np.arange(x_min, x_max, hx), np.arange(y_min, y_max, hy))

# Obtain labels for each point in mesh. Use last trained model.
Z = clusters.predict(np.c_[xx.ravel(), yy.ravel()])
```

In [ ]:
```
# TODO: Find the centroids for KMeans or the cluster means for GMM

centroids = ?
print centroids
```

In [ ]:
```
# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(1)
plt.clf()
plt.imshow(Z, interpolation='nearest',
           extent=(xx.min(), xx.max(), yy.min(), yy.max()),
           cmap=plt.cm.Paired,
           aspect='auto', origin='lower')

plt.plot(reduced_data[:, 0], reduced_data[:, 1], 'k.', markersize=2)
```

```
plt.scatter(centroids[:, 0], centroids[:, 1],
            marker='x', s=169, linewidths=3,
            color='w', zorder=10)
plt.title('Clustering on the wholesale grocery dataset (PCA-reduced data)\n'
          'Centroids are marked with white cross')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()
```

7) What are the central objects in each cluster? Describe them as customers.
Answer:

1.2.2 Conclusions

** 8)** Which of these techniques did you feel gave you the most insight into the data?
Answer:
9) How would you use that technique to help the company design new experiments?
Answer:
10) How would you use that data to help you predict future customer needs?
Answer: