Creating Customer Segments

In this project you, will analyze a dataset containing annual spending amounts for internal structure, that a wholesale distributor interacts with.

Instructions:

- Run each code block below by pressing **Shift+Enter**, making sure to implement any steps mar
- 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

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

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

# Read dataset
data = pd.read_csv("wholesale-customers.csv")
print "Dataset has {} rows, {} columns".format(*data.shape)
print data.head()  # print the first 5 rows
```

```
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
```

Feature Transformation

1) In this section you will be using PCA and ICA to start to understand the structure of the data. Bef your computations? List one or two ideas for what might show up as the first PCA dimensions, or w

Answer:

Right now, I have a dataset of 440 rows and 6 columns (Fresh, Milk, Grocery, etc.) When I perform F
matrix, where each customer's wholesale delivery is no longer represented explicitly in terms of the
Components (i.e. components that explain the variances in the data the best). On the other hand, w
sort of notion of where the orders might have come from (i.e. How many milk orders came from sma
task, I expect two vectors representing the orders of small businesses and big businesses over time

PCA

In [2]:

```
# TODO: Apply PCA with the same number of dimensions as variables in the c
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(data)

# Print the components and the amount of variance in the data contained ir
print pca.components_
print pca.explained_variance_ratio_
```

```
[[-0.97653685 -0.12118407 -0.06154039 -0.15236462  0.00705417 -0.06810471]
 [-0.11061386  0.51580216  0.76460638 -0.01872345  0.36535076  0.05707921]
[ 0.45961362  0.40517227]
```

2) How quickly does the variance drop off by dimension? If you were to use PCA on this dataset, hc

Answer:

Variance drops off relatively slowly from the 1st principal component to the 2nd (~5%). However, va
the 3rd. I would just use the first two principal components since they account for 0.45961362 + 0.4
the variation. Plus, it would be easier to visualize anyway, since it's easy to keep track of two things

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

Answer:

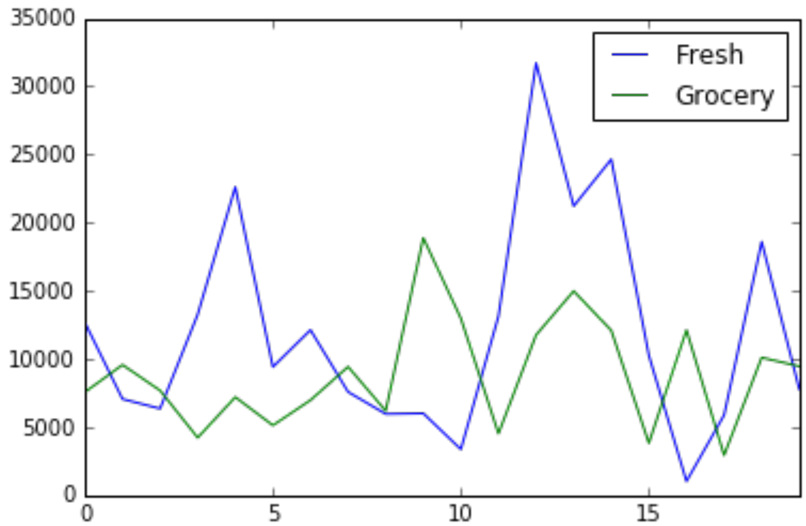
The first two dimensions seem to correspond strongly with annual customer spending on "fresh" an
graphs of PCA for the first and last 20 customers, it seems that the first principal component is nega
(see below). We observe that peaks in "fresh" annual spending per customer correspond to troughs
troughs in "fresh" annual spending per correspond to peaks in the latter. On the other hand, the sec
annual spending on "grocery" items.

This information makes sense. When we look at README, we find out that annual spending on fresl
annual spending per customer on grocery items has the second biggest standard deviation (9,503.1
data on them are maximized. Standard deviation is just the square root of the variance, so it follows
intimately related to the first and second principal components.

What this tells me is that if I really want the business to pay attention to only a couple of types of pr
belong to the "fresh" and "grocery" labels. Together, they seem to capture much of the information a

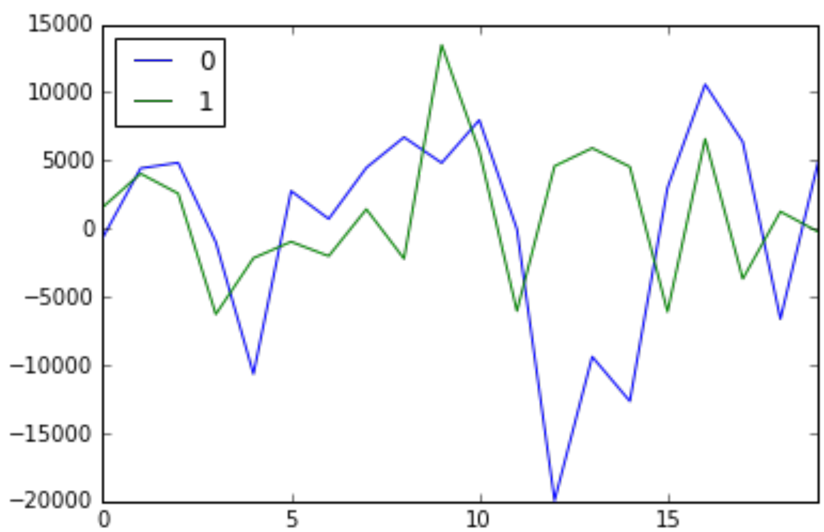
In [3]:
```
data[['Fresh', 'Grocery']].head(20).plot()
```

Out[3]: <matplotlib.axes._subplots.AxesSubplot at 0x115f14890>



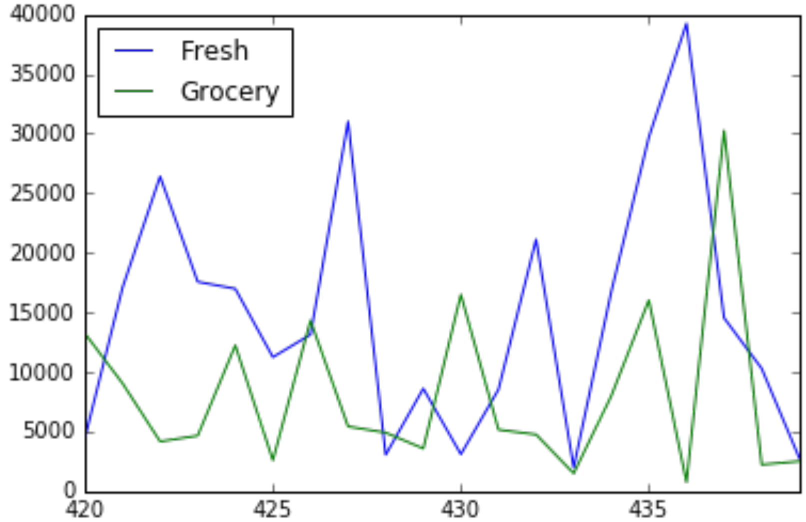
In [4]:
```
pd.DataFrame(pca.transform(data)).head(20).plot()
```

Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x1160e55d0>



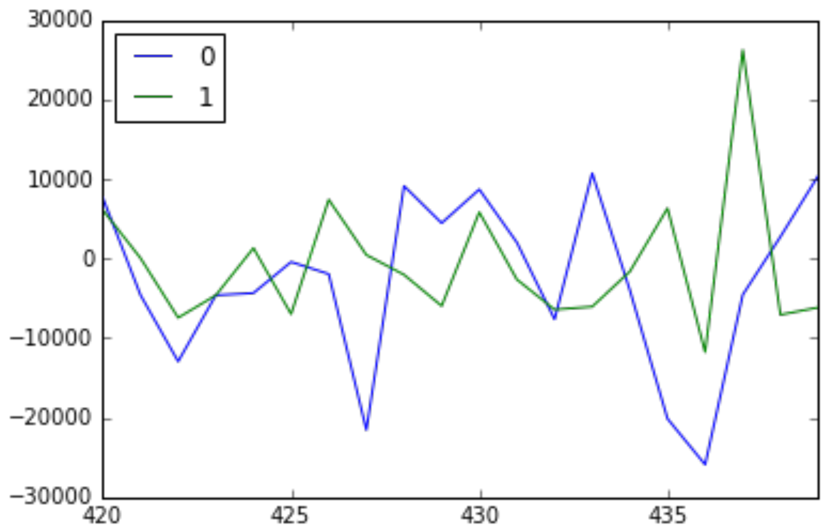
In [5]:
```
data[['Fresh', 'Grocery']].tail(20).plot()
```

Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x1156b1cd0>



In [6]:
```
pd.DataFrame(pca.transform(data)).tail(20).plot()
```

Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x1157ad410>



ICA

In [7]:
```
means = pd.DataFrame({'Fresh': [12000.30],
                      'Milk': [5796.27],
                      'Grocery': [7951.28],
                      'Frozen': [3071.93],
                      'Detergents_Paper': [2881.49],
                      'Delicatessen': [1524.87]})

means
```

Out[7]:

	Delicatessen	Detergents_Paper	Fresh	Frozen	Grocery	Milk
0	1524.87	2881.49	12000.3	3071.93	7951.28	5796.27

In [8]:
```
columns = ['Fresh','Milk','Grocery','Frozen','Detergents_Paper','Delicates
columns
```

Out[8]: `['Fresh', 'Milk', 'Grocery', 'Frozen', 'Detergents_Paper', 'Delicatessen']`

In [9]:
```
centered = pd.DataFrame(data[columns].values - means[columns].values,colum
centered
```

Out[9]:

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	668.7	3859.73	-390.28	-2857.93	-207.49	-186.87
1	-4943.3	4013.73	1616.72	-1309.93	411.51	251.13
2	-5647.3	3011.73	-267.28	-666.93	634.51	6319.13
3	1264.7	-4600.27	-3730.28	3332.07	-2374.49	263.13
4	10614.7	-386.27	-753.28	843.07	-1104.49	3660.13
5	-2587.3	2462.73	-2825.28	-2405.93	-1086.49	-73.87
6	125.7	-2597.27	-976.28	-2591.93	258.51	-979.87
7	-4421.3	-840.27	1474.72	-1402.93	439.51	1041.13
8	-6037.3	-2148.27	-1759.28	-2646.93	-1165.49	-774.87
9	-5994.3	5296.73	10929.72	-1912.93	4543.51	573.13

In [10]:
```
data.head(6)
```

Out[10]:

	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
5	9413	8259	5126	666	1795	1451

In [11]:
```
# 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 = FastICA()
ica_transformed = ica.fit_transform(centered)

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

```
[[ -2.11397544e-07   1.87388820e-06  -6.40732002e-06  -4.12939995e-07
    7.77076162e-07   1.45808053e-06]
 [  8.65005741e-07   1.40130758e-07  -7.73153584e-07  -1.11462356e-05
    5.54170292e-07   5.95549998e-06]
 [ -3.86763394e-07  -2.20400753e-07  -5.98208862e-07  -5.20149266e-07
    5.05571627e-07   1.80914105e-05]
 [ -3.97601692e-06   8.58892598e-07   6.30204434e-07   6.77372928e-07
   -2.07198807e-06   1.04084320e-06]
 [ -2.98707413e-07   2.27729498e-06   1.20814825e-05  -1.46126224e-06
   -2.82159750e-05  -5.71856392e-06]
 [ -1.52152755e-07  -9.85457174e-06   5.78481583e-06   3.68832268e-07
   -3.23641801e-06   6.06924079e-06]]
```

In [12]:
```
pd.DataFrame(ica_transformed)
```

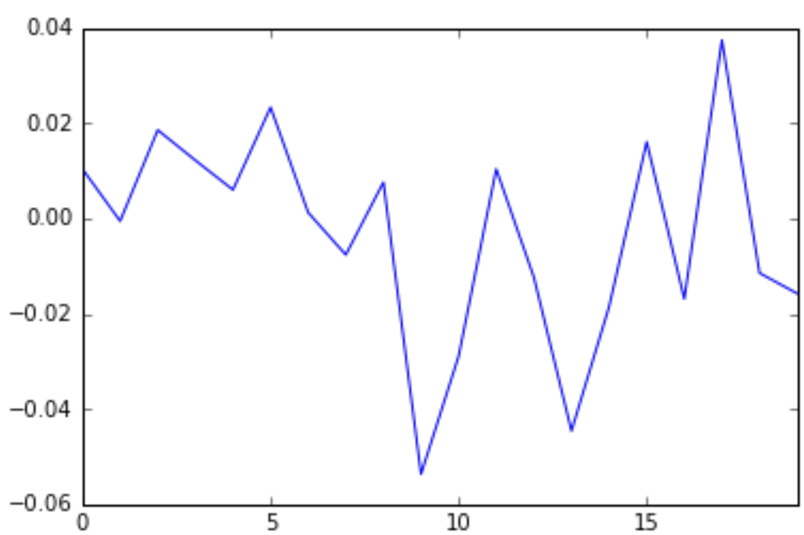
Out[12]:

	0	1	2	3	4	5
0	0.010338	0.032048	-0.002875	-0.001290	0.014974	-0.041912
1	-0.000566	0.011361	0.005493	0.022642	0.019016	-0.029740
2	0.018532	0.041163	0.116670	0.029683	-0.047749	0.005687
3	0.012176	-0.033555	0.004583	-0.003880	0.004704	0.034073
4	0.005989	0.021499	0.061650	-0.036342	-0.004149	0.023934
5	0.023306	0.026066	0.001514	0.011166	0.006842	-0.038039
6	0.001204	0.023698	-0.015141	-0.006657	-0.015650	0.012189
7	-0.007650	0.016999	0.020800	0.017010	0.000919	0.021863
8	0.007580	0.020080	-0.009370	0.020866	0.016841	0.010005
9	-0.053681	0.014360	0.008274	0.025157	0.017218	0.000010

Independent Component 0 corresponds to "Fresh" Product

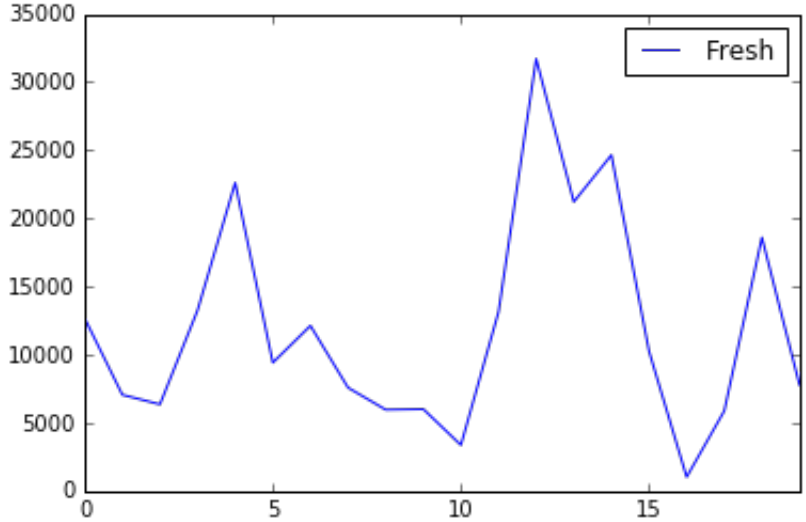
In [13]:
```
pd.DataFrame(ica_transformed)[0].head(20).plot()
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x115ea4d10>



In [14]:
```
pd.DataFrame(data[['Fresh']].head(20)).plot()
```

Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x116fed390>

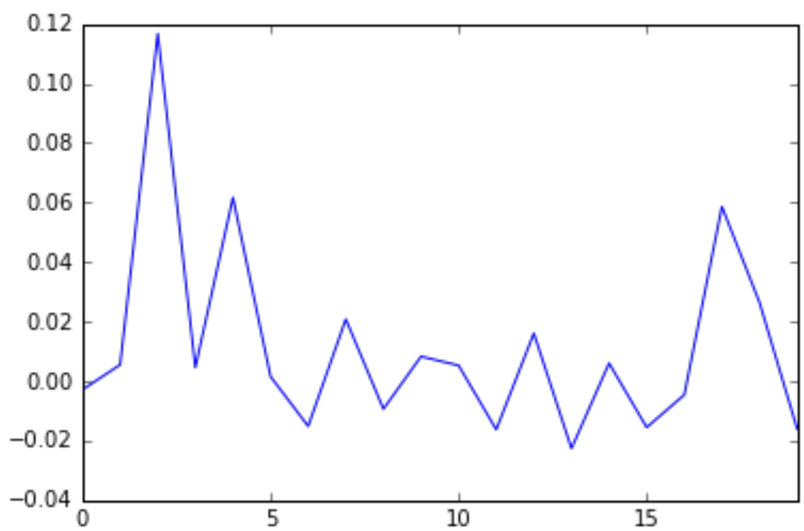


Independent Component 2 follows the general trend of "Gro "Detergents_Paper" Products.

It makes sense that the amount of money spent on grocery products is proportional to the amount (spent on the latter is a fraction of what people usually spend on groceries.

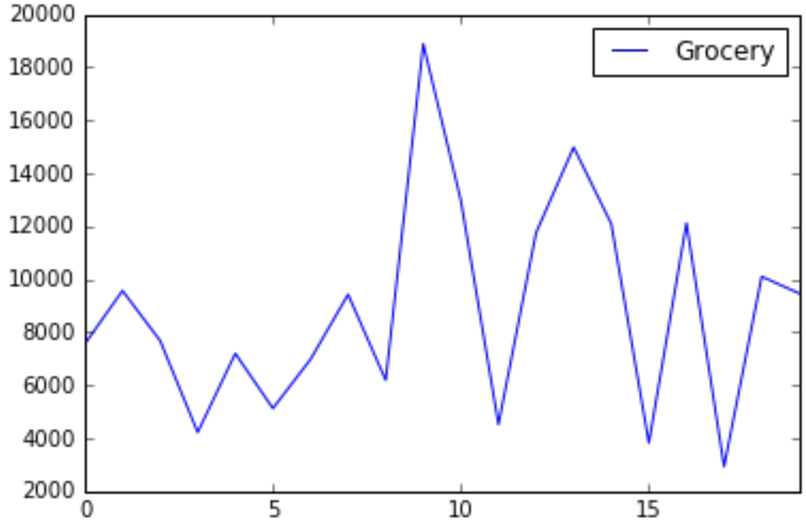
In [15]:
```
pd.DataFrame(ica_transformed)[2].head(20).plot()
```

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x1170b66d0>



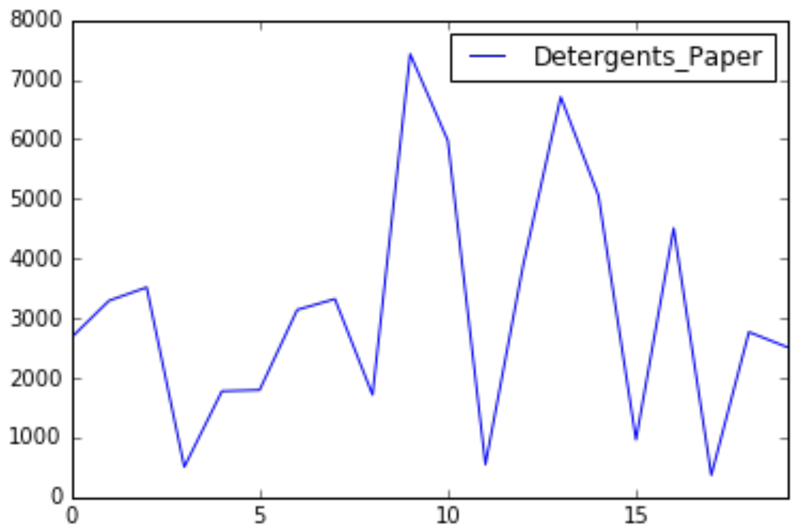
In [16]:
```
pd.DataFrame(data[['Grocery']].head(20)).plot()
```

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x11719be90>



In [17]:
```
pd.DataFrame(data[['Detergents_Paper']].head(20)).plot()
```

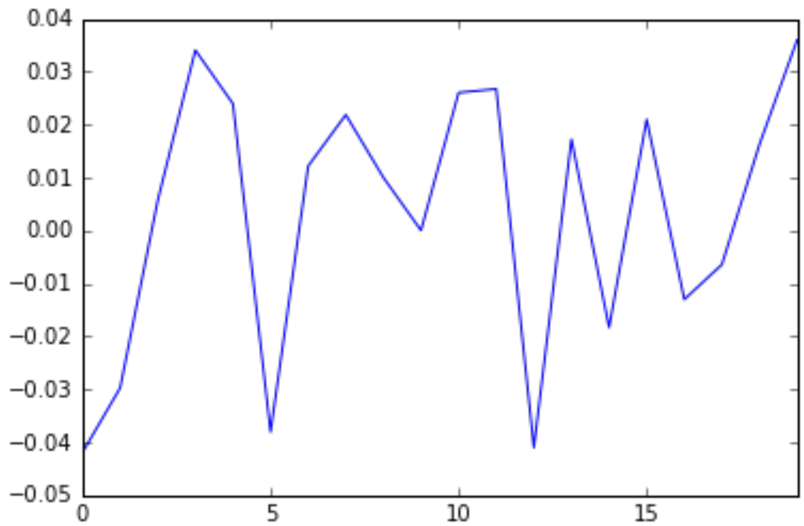
Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x11739ec10>



Independent Component 5 follows the general trend of "De

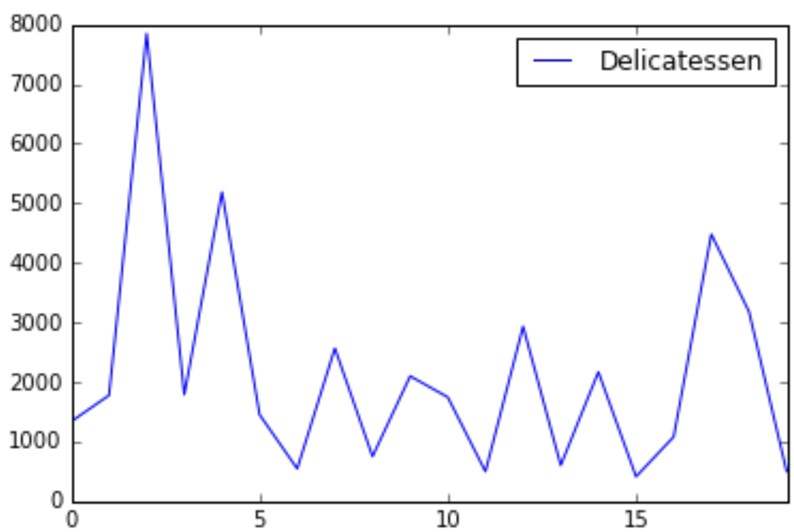
In [18]:
```
pd.DataFrame(ica_transformed)[5].head(20).plot()
```

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x117542890>



In [19]:
```
pd.DataFrame(data[['Delicatessen']].head(20)).plot()
```

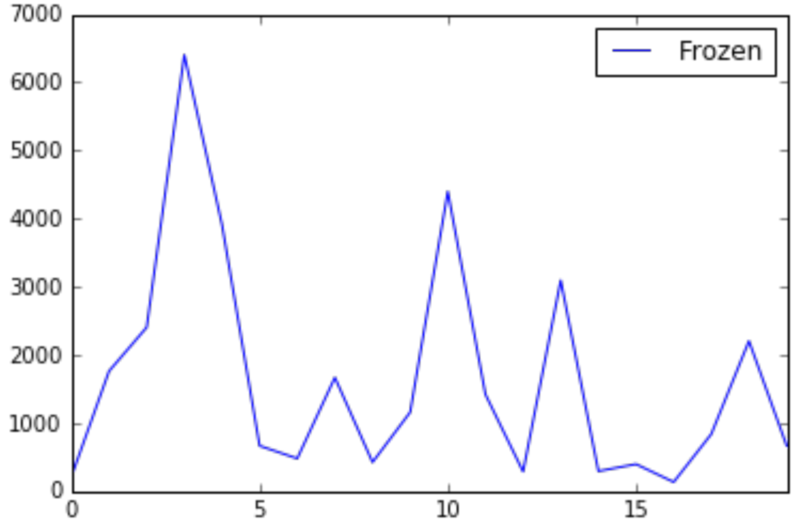
Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x117725650>



Not sure about the rest...

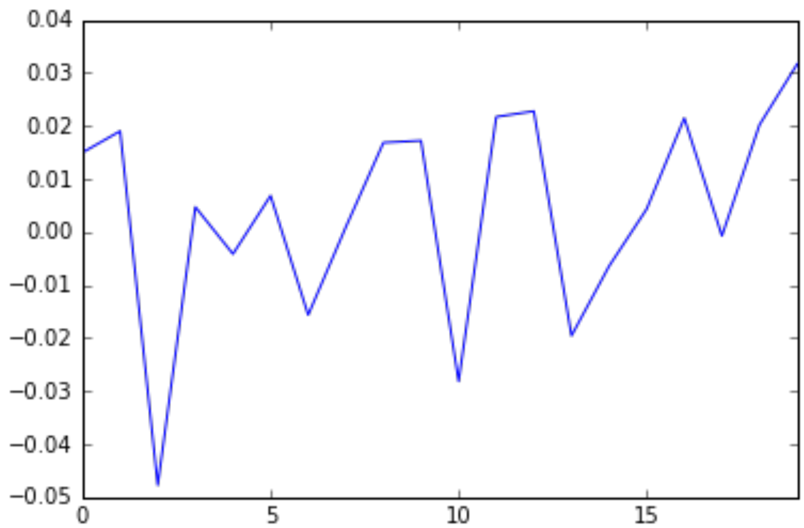
In [20]:
```
pd.DataFrame(data[['Frozen']].head(20)).plot()
```

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x1178863d0>



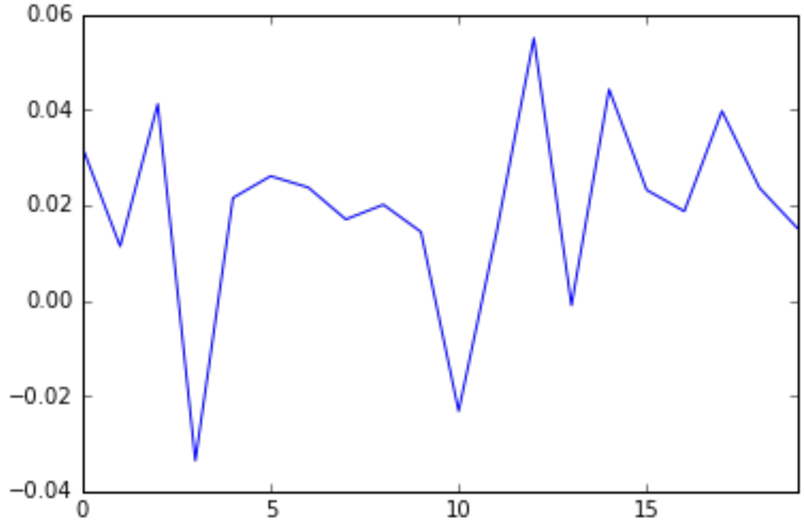
In [21]:
```
pd.DataFrame(ica_transformed)[4].head(20).plot()
```

Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x11777c1d0>



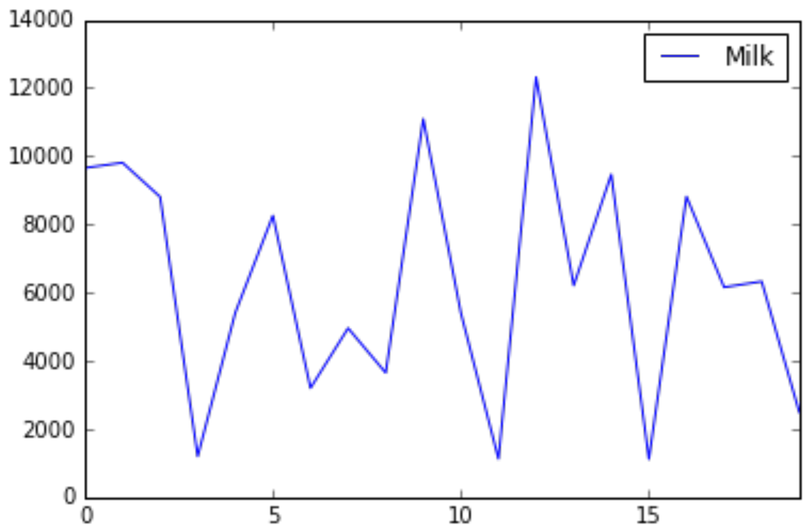
In [22]:
```
pd.DataFrame(ica_transformed)[1].head(20).plot()
```

Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x117d85a50>



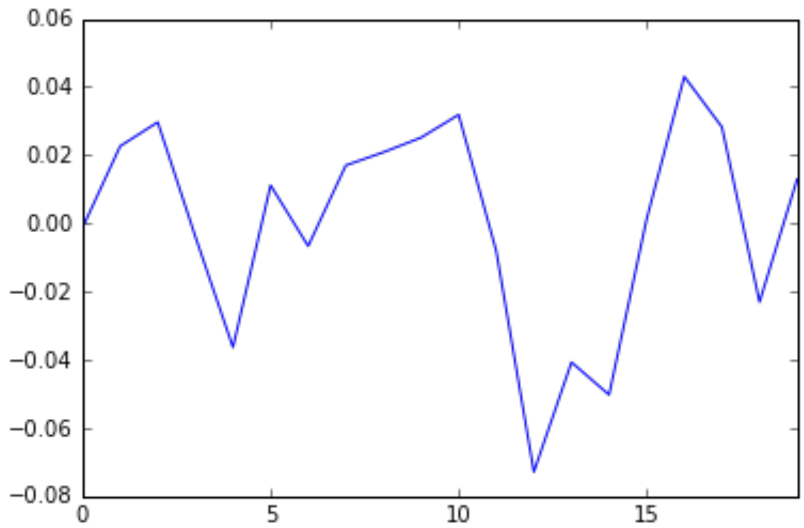
In [23]:
```
pd.DataFrame(data[['Milk']].head(20)).plot()
```

Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x117e6d6d0>



In [24]:
```
pd.DataFrame(ica_transformed)[3].head(20).plot()
```

Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x117efce50>



4) For each vector in the ICA decomposition, write a sentence or two explaining what sort of object used for?

Answer:

Each vector in the ICA decomposition is an independent component. ICA maps the data for each cu statistically independent from one another. ICA is usually used for separating superimposed signals learn.org/stable/modules/decomposition.html#ica (http://scikit-learn.org/stable/modules/decompos

The 0th independent component seems to correspond to the "fresh" products. The 2nd independen and "detergents_paper" products. The 5th independent component seems to be related to "delicate understand and they seem to not be as straightforward.

I suppose we could use these components as a way of representing data to be less noisy, since ICA statistically independent of each other.

Clustering

In this section you will choose either K Means clustering or Gaussian Mixed Models clustering, whic elements from the clusters to understand their significance.

Choose a Cluster Type

5) What are the advantages of using K Means clustering or Gaussian Mixture Models?

Answer:

The advantage of K Means Clustering is that it is relatively simple to understand. Once we select a r
feature space and are then iteratively "pulled by rubber bands" toward clumps of points. After sever
minima (sometimes they don't), and points in the feature space are assigned to the centroid that is c

The advantage of Gaussian Mixture Models is that their goals are similar to K Means, but they also t
http://scikit-learn.org/stable/modules/mixture.html (http://scikit-learn.org/stable/modules/mixture.ht
do soft assignment (i.e. assign a probability to the class of the point of interest) instead classifying t
because it reflects the uncertainty in classifying points that are right in between the decision bounda

6) Below is some starter code to help you visualize some cluster data. The visualization is based on
learn.org/stable/auto_examples/cluster/plot_kmeans_digits.html) from the sklearn documentation.

In [25]:

```
# Import clustering modules
from sklearn.cluster import KMeans
from sklearn.mixture import GMM
```

In [26]:

```
# TODO: First we reduce the data to two dimensions using PCA to capture va
reduced_data = pca.transform(data)
print reduced_data[:10]  # print upto 10 elements
```

```
[[  -650.02212207   1585.51909007]
 [  4426.80497937   4042.45150884]
 [  4841.9987068    2578.762176  ]
 [  -990.34643689  -6279.80599663]
 [-10657.99873116  -2159.72581518]
 [  2765.96159271   -959.87072713]
 [   715.55089221  -2013.00226567]
 [  4474.58366697   1429.49697204]
 [  6712.09539718  -2205.90915598]
 [  4823.63435407  13480.55920489]]
```

In [27]:

```
# TODO: Implement your clustering algorithm here, and fit it to the reduce
# The visualizer below assumes your clustering object is named 'clusters'

def cluster(clusterer):
    clusterer.fit(reduced_data)
    clusters = clusterer
    print clusters
    return clusters
```

In [28]:
```
# Plot the decision boundary by building a mesh grid to populate a graph.

def plot_boundary(clusters):
    x_min, x_max = reduced_data[:, 0].min() - 1, reduced_data[:, 0].max()
    y_min, y_max = reduced_data[:, 1].min() - 1, reduced_data[:, 1].max()
    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_m

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

In [29]:
```
# TODO: Find the centroids for KMeans or the cluster means for GMM

def cluster_means(clusters, func_name):
    centroids = getattr(clusters, func_name)
    print centroids
    return centroids
```

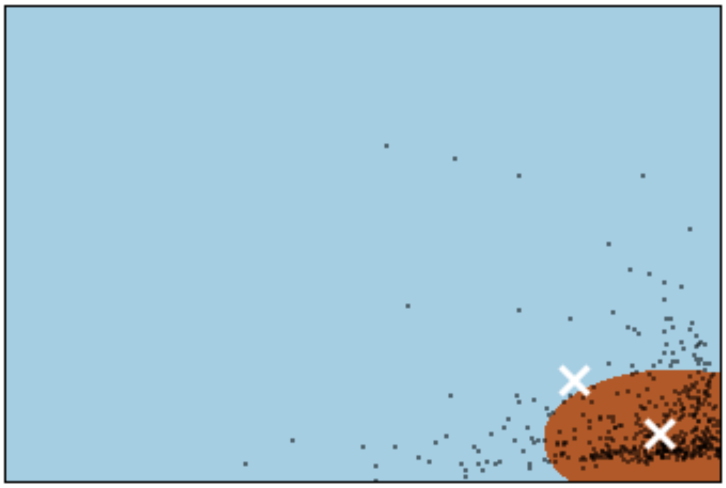
In [30]:
```
# Put the result into a color plot
def color_plot(clusters, Z,xx,yy,x_min,x_max,y_min,y_max,centroids):
    Z = Z.reshape(xx.shape)
    plt.figure(1)
    plt.clf()
    plt.imshow(Z, interpolation='nearest',
               extent=(xx.min(), xx.max(), yy.min(), yy.max()),
               cmap=plt.cm.Paired,
               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 da
              'Centroids are marked with white cross')
    plt.xlim(x_min, x_max)
    plt.ylim(y_min, y_max)
    plt.xticks(())
    plt.yticks(())
    plt.show()
```

In [31]:
```
def cluster_and_plot(clusterer,func_name):
    clusters = cluster(clusterer)
    Z,xx,yy,x_min,x_max,y_min,y_max = plot_boundary(clusters)
    centroids = cluster_means(clusters,func_name)
    color_plot(clusters,Z,xx,yy,x_min,x_max,y_min,y_max,centroids)
```

In [32]:
```
clusterer = GMM(n_components=2)
cluster_and_plot(clusterer, 'means_')
```

```
GMM(covariance_type='diag', init_params='wmc', min_covar=0.001,
  n_components=2, n_init=1, n_iter=100, params='wmc', random_state=None,
  thresh=None, tol=0.001)
[[-10810.23008886   9858.15532401]
 [  3308.39301792  -3017.01739698]]
```

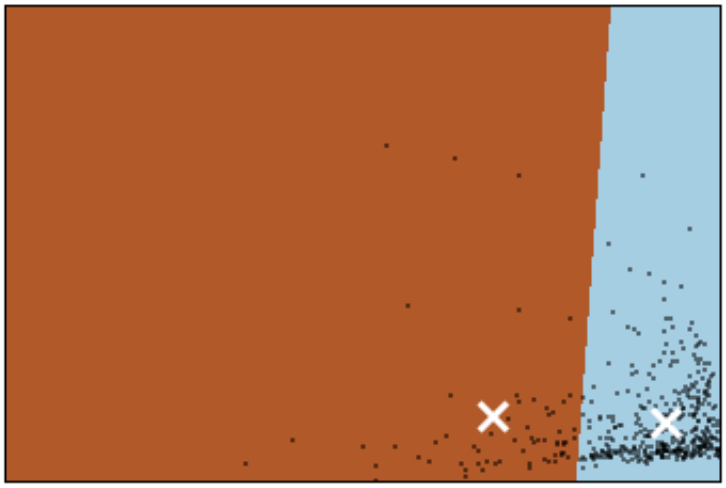
Clustering on the wholesale grocery dataset (PCA-reduced data)
Centroids are marked with white cross



In [33]:
```
clusterer = KMeans(n_clusters=2)
cluster_and_plot(clusterer, 'cluster_centers_')
```

```
KMeans(copy_x=True, init='k-means++', max_iter=300, n_clusters=2, n_init=1
    n_jobs=1, precompute_distances='auto', random_state=None, tol=0.0001,
    verbose=0)
[[  4175.31101293   -211.15109304]
 [-24088.33276689   1218.17938291]]
```

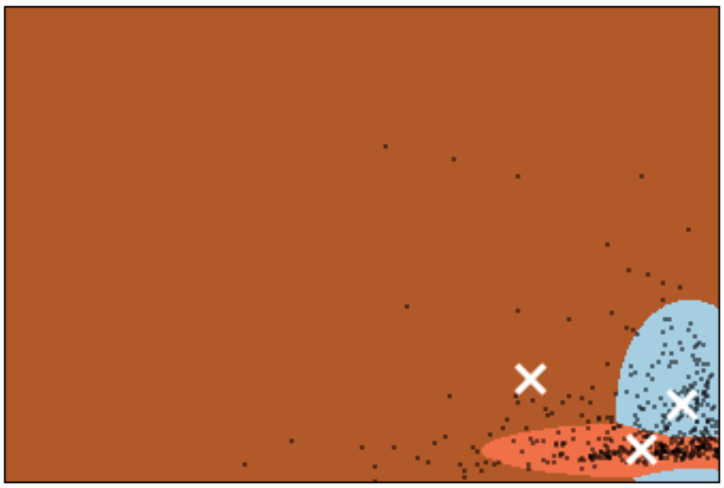
Clustering on the wholesale grocery dataset (PCA-reduced data)
Centroids are marked with white cross



In [34]:
```
clusterer = GMM(n_components=3)
cluster_and_plot(clusterer, 'means_')
```

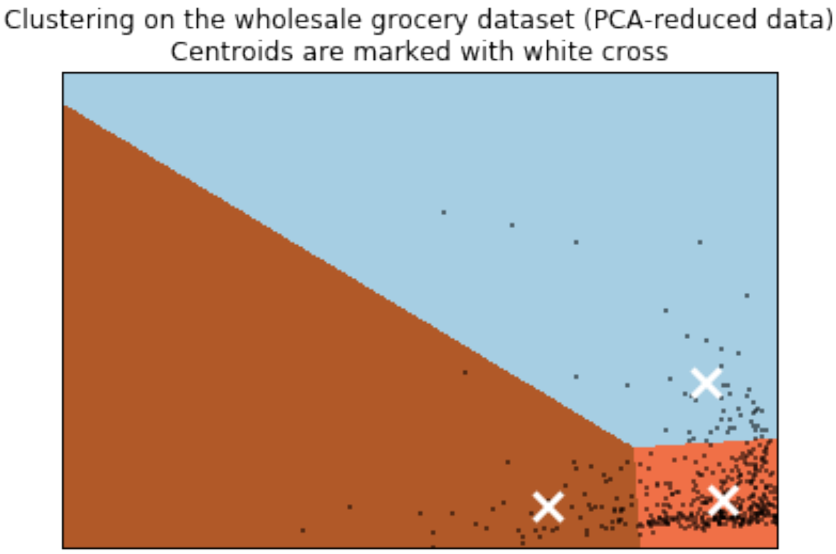
```
GMM(covariance_type='diag', init_params='wmc', min_covar=0.001,
  n_components=3, n_init=1, n_iter=100, params='wmc', random_state=None,
  thresh=None, tol=0.001)
[[  6986.10611001   4252.253209  ]
 [   276.11243352  -6508.22600668]
 [-17878.44951149  10108.28824429]]
```

Clustering on the wholesale grocery dataset (PCA-reduced data)
Centroids are marked with white cross



In [35]:
```
clusterer = KMeans(n_clusters=3)
cluster_and_plot(clusterer, 'cluster_centers_')
```

```
KMeans(copy_x=True, init='k-means++', max_iter=300, n_clusters=3, n_init=1
    n_jobs=1, precompute_distances='auto', random_state=None, tol=0.0001,
    verbose=0)
[[  1497.13461172  24998.27760147]
 [  4106.90273941  -3168.41202086]
 [-24220.71188261  -4364.45560022]]
```



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

Answer:

Central objects in each cluster represent the centroids. In terms of customers, since they are "smac
"average" customer in that grouping.

Conclusions

8) Which of these techniques did you feel gave you the most insight into the data?

Answer:

I felt that the best techniques to use were to use PCA to summarize the data into two principal com
products respectively. After doing PCA, using Gaussian Mixture Models with 2 or 3 components ma
component captures some sort of "oval" clump, while the other captures everything that's not in the
"small" businesses. With 3 GMM components, the rough oval shape became divided into two small
the two ovals.

In the 3-component clusters using GMM, we can identify 3 groups: one of customers that order lots products AND grocery items, and lastly, one of customers that don't belong in those two groups. M cluster of customers that predominantly order only fresh products. Since the delivery schedule has those customers are probably most affected. The products in the "fresh" category are probably not hours of operation (i.e. they have more hours open during the day than during night), this means tha wait till the next day to get bought. Also, "bulk" deliveries probably means we are not delivering to t products are likely to become more stale because of the new changes. I would test this idea out, ag customers are in the clusters.

9) How would you use that technique to help the company design new experiments?

Answer:

Now that we have some clusters, and given the information that some companies have complaints we know have issues with the new change onto the first two principal components as we've done a interesting experiment would be to figure out if these unhappy customers are overwhelmingly conce clusters, a null hypothesis could be that customers who have problems should be evenly distributed complaints, we could map each of those customers onto the new projection and test our hypothesi customers overwhelmingly are concentrated in one cluster), we could then reach out to the other cu needs and maybe give them special offers so that they keep doing business with us (such as restor

10) How would you use that data to help you predict future customer needs?

Answer:

After the A/B test, if we got to the point of successfully identifying the cluster that most of the unhap that cluster to prefer the old method of delivery. We predict that those customers in that cluster prol could assign them labels based on which cluster they belong to. After that, we could treat this as a

In [ ]:

```

```