Declaration

I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

____________________________________  __________________
Name: James Marron                      Date: April 6th 2011
Permission to Lend

I agree that the Library and other agents of the College may lend or copy this thesis upon request.

__________________________________________ April 6\textsuperscript{th} 2011

James Marron
Acknowledgment

I wish to thank my supervisor Prof. Khurshid Ahmad for the help, guidance and direction he has provided me over the course of this project.

I owe a debt of gratitude to my family for the support they have shown me throughout my entire time at university.

Lastly, to thank my editor-in-chief Sinead for reading and re-reading this document. The last four weeks were infinitely more enjoyable with you on board.
Abstract
This paper aims to examine the link between daily average temperature and the price of a barrel of oil. The approach taken in this project is to compare the number of Heating Degree Days for a representative population of the United States of America for the period 2001-2010 and correlate these figures with the daily price of a barrel of oil. The results suggest that although factors such as political instability in a region or economic conditions may have a more profound impact on oil prices, there is in fact a strong correlation between daily average temperatures and oil prices.
Contents
Declaration.................................................................................................................................1
Permission to Lend.........................................................................................................................2
Acknowledgment ..........................................................................................................................3
Abstract........................................................................................................................................4
1. Key Definitions .........................................................................................................................6
1.1 Derivative ..................................................................................................................................6
1.2 Weather Derivative ..................................................................................................................6
1.3 Heating Degree Day ................................................................................................................6
1.4 Cooling Degree Day ................................................................................................................6
2. Motivation ..................................................................................................................................8
3. Literature Review ......................................................................................................................9
4. Design and Implementation ......................................................................................................10
5. Results ......................................................................................................................................13
6. Future Work and Conclusion .................................................................................................18
Appendix .......................................................................................................................................20

Data included in the Appendix can be found on the attached CD.
1. Key Definitions

1.1 Derivative
In finance a derivative is a security with an expiration date whose price depends on an underlying asset. The derivative itself is merely a contract between two or more parties. The value of a derivative is determined by predicted changes in the value of the underlying asset. The derivative itself has no value, but can be traded before its expiration date as though it were an asset. Derivatives are often used as a method of minimising exposure to risk.

1.2 Weather Derivative
A weather derivative is a derivative whose payoff depends on a specified event e.g. the average temperature in a city for a particular month. It differs from a traditional derivative because the underlying has no direct value on which to price the derivative. There is no standard model to determine the value of weather derivatives because average temperatures, rainfall levels etc. cannot be traded.

1.3 Heating Degree Day
A Heating Degree Day is a relative measure of how cold a day was. The commonly accepted baseline for this relative measure is 65°F or 18°C. To calculate the number of HDD’s on a specific day, subtract the average temperature from the baseline. For example, on a 35°F day, the HDD is 65 – 35 = 30. For a 30 day month of similar days, the HDD total is 900. If a day is warmer than the baseline, the HDD for that day is 0. The number of HDD’s \( z_i \) on a particular day can be expressed as follows:

\[ z_i = \max(To - Ti, 0) \]

where \( T_o \) is the baseline temperature and \( T_i \) is the average temperature of the day in question. An HDD index \( x \) over an \( N_d \) day period is defined as the sum of the HDD’s over all days during that period:

\[ x = \sum_{i=1}^{N_d} z_i \]

1.4 Cooling Degree Day
A Cooling Degree Day is a relative measure of how hot a day was. The baseline for this relative measure is 65°F or 18°C. To calculate the number of CDD’s on a particular day, subtract the baseline from the average temperature. For example, on a 23°C day, the CDD is 23-18 = 5. For a 30 day month of similar days, the CDD total is 150. If a day is colder than the baseline, the CDD for that day is 0. The number of CDD’s \( z_i \) on a particular day can be expressed as follows:

\[ z_i = \max(Ti - To, 0) \]
As with HDD’s, a CDD index \( x \) over an \( N_d \) day period is defined as the sum of the CDD’s over all days during that period:

\[
x = \sum_{i=1}^{N_d} z_i
\]

A record of monthly and especially daily HDD totals can be very useful for trading weather derivatives. If we can see patterns of temperatures for particular regions, we can make more accurate decisions as to the value of a weather derivative contract. This is especially true in the oil market. Since the majority of houses are heated using oil, it can be useful to have a picture of demand based on how hot or cold we predict an area to be at a particular time of year. Population of a region is an important factor in our analysis. We assign a greater weight of importance to information regarding a large city such as New York than we would to a small town in Kentucky for example.

Furthermore, since weather derivatives do not have a standard model of valuation such as the Black-Scholes method, HDD patterns and calculations provide a useful reference tool in the process of pricing weather derivatives.
2. Motivation

Since the drilling of the world’s first commercial oil well in Poland in 1853 our society has steadily become more dependent on crude oil. In the US, Petroleum comprises 40% of total energy consumption, yet is responsible for only 2% of electricity generation. According to the U.S. Energy Information Administration (EIA), an independent source of energy related statistics and analysis, it is estimated that there are between 1,119 and 1,317 billion barrels of oil reserves still in the ground. At current daily production levels of 84,000,000 barrels of oil per day, we have approximately 43 years worth of oil left. As these resources diminish, it becomes more important to make future price projections with increasing accuracy. The aim of this project is to explore the relationship, if any, between weather - specifically temperature and oil prices. Daily Heating Degree Day calculations are calculated for a number of U.S. cities across a 10 year period to build up a temperature profile. This profile can then be used to examine various statistical correlations with the price of oil.

This profile can also be used as part of the information gathering process when trying to set a price for certain weather derivatives. Since there is no standard model for the pricing of weather derivatives, it is important to accumulate as much relevant information as possible. If we can see patterns of temperatures for particular regions, we can make more accurate decisions as to the value of a weather derivative contract.

The National Oceanic and Atmospheric Association (NOAA) compile and record population weighted HDD information for individual states and regions across the U.S. My project seeks to go a step further and keep a track of daily HDD’s from a representative sample of cities rather than state-wide totals. Due to the constant fluctuation in oil prices, even on an hourly basis, there is an argument to be made that analysing monthly data may be prove sufficient and that to look at daily patterns or trends may be of benefit.
3. Literature Review

3.1 Asset Price Dynamics, Volatility and Prediction  
Stephen J. Taylor

This book was a very helpful introduction to some of the more complex economic concepts related to my project particularly volatility and its various forms. Volatility of prices (the rate of change of prices) is a topic that is especially relevant to oil prices. Furthering my understanding of volatility and its different forms was a worthwhile investment of my time. Taylor also outlines a definition of stylised facts – general statistical properties of time series. Since I am concerned with a daily time series concerning temperature, Heating Degree Days and the closing price of a barrel of crude oil, this seemed a topic worthy of further exploration. Of particular importance were the following three stylised facts:

1. The distribution of returns is not normal.
2. The correlation between today’s return and any subsequent return is almost zero.
3. There are transformations of returns that reveal positive correlation between observations made at nearby times.

Taylor uses clear concise examples linked to what he calls the ‘twenty series’. This is a short list of some common economic benchmarks such as the S&P 500-share, Coca-Cola, General Electric, General Motors, FT 100-share, Gold and Live Cattle among others.

3.2 Weather Derivative Valuation  
Stephen Jewson and Anders Brix

This book by Jewson and Brix was my main introduction to the world of weather derivatives and some of the associated terminology. They write that the purpose of weather derivatives is to allow businesses and other organisations insure themselves against fluctuations in the weather. This insurance against risk or ‘hedging’ is desirable because it reduces the year-to-year volatility of profits. This reduction is risk is beneficial for a number of reasons:

- low volatility in profits can reduce the interest rate at which companies borrow money
- in a publicly traded company low volatility in profits usually translates into low volatility in the share price
- low volatility in profits reduces the risk of bankruptcy

The authors laid out the concept of a derivative contract and more importantly a weather derivative contract which has the following attributes:

- the contract period: a start date and an end date
- a measurement station
- a weather variable, measurement station over the contract period
- a pay-off function which converts the aggregation of the weather variable into cash flow that settles the derivative shortly after the contract period

Jewson and Brix also put forward the concept of a Heating and Cooling Degree Day and the relative importance of each when pricing weather derivatives given the absence of a method similar to Black-Scholes. The Black-Scholes method develops partial differential equations to price financial markets and derivatives. Weather Derivative Valuation also provides tables of data detailing HDD totals for the previous 30 years for a number of worldwide cities, some of which I have included in my U.S. profile. These served as useful reference points for some of the information when it came to the analysis stage of my project.

4. Design and Implementation

Web scraping is the process of extracting useful information from an Internet website. This extraction can be performed manually or a script can be written to do so. Given the volume of information I would be extracting from the Weather Underground website, and to ensure its accuracy, I decided to write a scraper that would accurately and efficiently gather the required information.

Diagram of System Architecture
The aim of the scraper is to access the desired web-page, depending on the city and date, as shown from the diagram above. Once found, the html of the page is scanned to find the values for the maximum and minimum temperatures for the specific day in question. This data is then written out to a file which can later be used in Microsoft Excel as the foundation for the statistical calculations and all its associated conclusions.

I chose Microsoft Excel as the program for my statistics related work due to its prevalence in the working world. The chance to further my knowledge of such a widespread program proved of interest. Excel is a very thorough, if slightly crude tool for statistical analysis but for my objectives seemed appropriate.

Early generations of my scraper were coded in Java for a number of reasons. Firstly, it is the language with which I am more experienced, having used it for many projects during my time as a student. Secondly, there is a wealth of Java related knowledge available not only on the Internet but also here in the school of Computer Science and Statistics. This would mean that any problems that could arise could be discussed and expanded upon face to face; therefore allowing for a more specific discussion than might otherwise be available on an Internet forum.

After a considerable time invested into developing a Java web scraper, I changed languages to the Python programming language. I identified this Final Year Project as a unique opportunity to expand my skill base, free from the narrow time constraints of a typical in-class assignment. Despite my reluctance to discard my early work with Java, Python’s suitability as a scripting language coupled with the challenge of learning a new skill made the switch an easy one to make.

When writing the script which would act as my web scraper, I was conscious to include only that which was necessary. This script wasn’t designed with any further use beyond the scope of this project in mind. Its goal is to extract the daily average temperature from the Weather Underground website. This minimalist approach made it very easy to lay out my code from a requirements point of view.

Once I had a successful scraper working I now had to choose my data set. The wide variation in climates across the U.S. e.g. from snow capped mountains in Colorado to the desert in Arizona and everything in between meant that it would be important to have a representative sample of this variation in my data set. Similarly, the densities of population across regions would have to be taken into account; from major cities such as New York and Los Angeles to sparser regions such as Alaska. With these factors in mind I chose eight cities to represent my profile: Anchorage, Alaska; Atlanta, Georgia; Chicago, Illinois; Houston, Texas; Los Angeles, California, New York, New York; Philadelphia, Pennsylvania and Seattle, Washington. I chose Atlanta, Chicago, Houston, New York and Philadelphia because Jewson used them in his analysis in *Weather Derivative Valuation* and having those figures to compare mine to (albeit with the inclusion of population weights in my figures) would be a helpful guide as I analysed my data. I chose Los Angeles because it is the second most populous city in the United States and so the effects on its population with regard to oil consumption are important. This also meant that my list included the three most populous
cities - New York, Los Angeles and Chicago. Seattle was added to the list to include the region of the Pacific North West. I thought Alaska to be an interesting inclusion because it is not only the largest state by area but also the least densely populated. Its climate would also provide an interesting contrast and counter balance to some of the other cities listed.
5. Results

When analysing any of the following data, it is important to note that the HDD function is a threshold function and as such statistical calculations and conclusions can be somewhat difficult as the result suddenly tends to 0. Furthermore, the results seen from data gathered from Los Angeles can be somewhat misleading as the majority of HDD’s occur somewhat counter-intuitively during non-winter months. This may be explained by the relatively small gap between daily maximum and daily minimum temperatures coupled with higher than average minimum temperatures during certain months.

The system I have designed takes two inputs: (1) daily temperature by city as sourced from Weather Underground - wunderground.com and (2) the NOAA monthly record of state wide HDD totals. The respective processing of each input can be seen in the following diagram.

\[
\text{max}(T_0 - T_i, 0)
\]

where \(T_0\) would represent the baseline temperature of 18°C and \(T_i\) would represent the average temperature for the day in question. The HDD values for
each city would then be compared with the state wide totals as gathered by the National Oceanic and Atmospheric Administration to find the relative proportions of HDD’s occurring in each city. My calculated HDD values would then be weighted according to the population of each city and then by correlating with oil price information received from Trade Station (an online source of trading related information) a picture of “oil demand” could be seen.

My data set is limited to beginning in May 2001 due to the data available from Trade Station. I chose to end the data set in December to coincide with the end of the calendar year. The full table of results can be seen on the attached Excel workbook labelled ‘Daily Temps’. The tab labelled ‘Sheet 2’ gives details of daily average temperature, HDD calculation and population weighted HDD calculation for each day between 14/05/2001 and 31/12/2010. The following table details some of the important basic information which would be used for my results such as total HDD calculation for the time period, city population, population ratio, correlation between daily temperature and the closing price of a barrel of oil, correlation between daily HDD and the closing price of a barrel of oil and the weighted calculations of both correlations.

<table>
<thead>
<tr>
<th></th>
<th>Anch</th>
<th>Atl</th>
<th>Chi</th>
<th>Hou</th>
<th>LA</th>
<th>NY</th>
<th>Phi</th>
<th>Sea</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDD Totals</td>
<td>51292</td>
<td>14932</td>
<td>30096</td>
<td>6172</td>
<td>5088</td>
<td>23440</td>
<td>23586</td>
<td>22885</td>
<td>177491</td>
</tr>
<tr>
<td>Pop.</td>
<td>286174</td>
<td>542822</td>
<td>2695598</td>
<td>2695598</td>
<td>3833995</td>
<td>8391881</td>
<td>1556396</td>
<td>608660</td>
<td>20014977</td>
</tr>
<tr>
<td>Ratio</td>
<td>0.014</td>
<td>0.027</td>
<td>0.135</td>
<td>0.105</td>
<td>0.192</td>
<td>0.419</td>
<td>0.078</td>
<td>0.03</td>
<td>1</td>
</tr>
<tr>
<td>Temp Cor</td>
<td>-0.240</td>
<td>-0.193</td>
<td>-0.199</td>
<td>-0.159</td>
<td>0.172</td>
<td>-0.201</td>
<td>-0.186</td>
<td>-0.17</td>
<td>-0.15</td>
</tr>
<tr>
<td>HDD Cor</td>
<td>0.241</td>
<td>0.168</td>
<td>0.194</td>
<td>0.095</td>
<td>-0.137</td>
<td>0.197</td>
<td>0.182</td>
<td>0.18</td>
<td>0.14</td>
</tr>
<tr>
<td>W Temp Cor</td>
<td>-0.240</td>
<td>-0.193</td>
<td>-0.199</td>
<td>-0.159</td>
<td>0.172</td>
<td>-0.201</td>
<td>-0.186</td>
<td>-0.169</td>
<td>-0.15</td>
</tr>
<tr>
<td>W HDD Cor</td>
<td>0.241</td>
<td>0.168</td>
<td>0.194</td>
<td>0.095</td>
<td>-0.137</td>
<td>0.197</td>
<td>0.182</td>
<td>0.175</td>
<td>0.14</td>
</tr>
</tbody>
</table>

Table of Population, Population Ratio and Price Correlations

A quick analysis of the HDD correlation and indeed the weighted HDD correlation with the price of oil does not look promising for our hypothesised link between HDD values and oil prices. With this in mind, the following table narrows the HDD calculations to the traditional winter months of November, December and January.

~ 14 ~
Winter HDD Calculations and Correlations

From the ‘Diff’ column (Total Correlation – Winter Correlation) we can see that the correlation gets stronger across the board when the HDD calculation are limited to these winter months, lending credence to our hypothesis. The unusual nature of the HDD’s for Los Angeles can be seen here. This may be explained by the fact that even though the daily maximum temperatures are not that high (mid-teens), the minimum temperature hovers around the low teens giving a high value for the daily average and so relatively few HDD’s.

Having calculated the daily HDD values for each city the table below shows the correlation between the HDD values across the entire time span to illustrate the diversity of the climates involved.

<table>
<thead>
<tr>
<th></th>
<th>Anch</th>
<th>Atl</th>
<th>Chi</th>
<th>Hou</th>
<th>LA</th>
<th>NY</th>
<th>Phi</th>
<th>Sea</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anch</td>
<td>1</td>
<td>0.72</td>
<td>0.78</td>
<td>0.54</td>
<td>-0.47</td>
<td>0.77</td>
<td>0.77</td>
<td>0.79</td>
</tr>
<tr>
<td>Atl</td>
<td>0.72</td>
<td>1</td>
<td>0.85</td>
<td>0.77</td>
<td>-0.42</td>
<td>0.86</td>
<td>0.87</td>
<td>0.63</td>
</tr>
<tr>
<td>Chi</td>
<td>0.78</td>
<td>0.85</td>
<td>1</td>
<td>0.72</td>
<td>-0.42</td>
<td>0.89</td>
<td>0.88</td>
<td>0.76</td>
</tr>
<tr>
<td>Hou</td>
<td>0.54</td>
<td>0.77</td>
<td>0.72</td>
<td>1</td>
<td>-0.31</td>
<td>0.64</td>
<td>0.64</td>
<td>0.51</td>
</tr>
<tr>
<td>LA</td>
<td>-0.47</td>
<td>-0.42</td>
<td>-0.42</td>
<td>-0.31</td>
<td>1</td>
<td>-0.40</td>
<td>-0.42</td>
<td>-0.37</td>
</tr>
<tr>
<td>NY</td>
<td>0.77</td>
<td>0.86</td>
<td>0.89</td>
<td>0.64</td>
<td>-0.40</td>
<td>1</td>
<td>0.99</td>
<td>0.72</td>
</tr>
<tr>
<td>Phi</td>
<td>0.77</td>
<td>0.87</td>
<td>0.88</td>
<td>0.64</td>
<td>-0.42</td>
<td>0.99</td>
<td>1</td>
<td>0.72</td>
</tr>
<tr>
<td>Sea</td>
<td>0.79</td>
<td>0.63</td>
<td>0.76</td>
<td>0.51</td>
<td>-0.37</td>
<td>0.72</td>
<td>0.72</td>
<td>1</td>
</tr>
</tbody>
</table>

Total HDD Correlations between cities

The column ‘Wgt Series’ in the main table (Sheet 2 of ‘Daily Temps’) is the sum of the respective weighted HDD values from the eight cities for the day in question and gives a national perspective to the HDD values. This value is particularly important as we seek to achieve the representative sample of the national climate as discussed in the Design and Implementation section. ‘Price Dif’ tracks the difference in the price of a barrel of oil between consecutive days. ‘Price Return’ for day $n$ is $\log (\text{price}_n / \text{price}_{n-1})$ and is a common measure of the listing the market returns of a stock. This is because the logarithmic function, when plotted, tends to flatten out the growth curve and in doing so aids visualisation.
One of the most important factors when trying to analyse a data set as large as mine is to uncover the interesting data; to filter out the meaningful data from results that can act as background noise and overwhelm the reader from arriving at any worthwhile conclusions. One of the things I did to try and combat this problem was to look at prices when the weather was at its coldest i.e. when the HDD totals were highest. For each of the ten years, I looked at the coldest day of the year and mapped the oil price on that day against the maximum oil price for that year. The table and graph below show the resulting output.

<table>
<thead>
<tr>
<th>Date</th>
<th>Close $</th>
<th>Yearly Max</th>
<th>Close / Max</th>
<th>Infl Value</th>
<th>Diff</th>
<th>Wgt HDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>2001</td>
<td>73.56</td>
<td>79</td>
<td>0.931</td>
<td>5.44</td>
<td>16.63</td>
<td></td>
</tr>
<tr>
<td>2002</td>
<td>89.48</td>
<td>91.11</td>
<td>0.982</td>
<td>86.77</td>
<td>1.63</td>
<td>16.11</td>
</tr>
<tr>
<td>2003</td>
<td>91.43</td>
<td>120.44</td>
<td>0.759</td>
<td>109.49</td>
<td>29.01</td>
<td>20.95</td>
</tr>
<tr>
<td>2004</td>
<td>108.26</td>
<td>126.61</td>
<td>0.855</td>
<td>110.10</td>
<td>18.35</td>
<td>20.81</td>
</tr>
<tr>
<td>2005</td>
<td>103.7</td>
<td>139.78</td>
<td>0.742</td>
<td>116.48</td>
<td>36.08</td>
<td>20.25</td>
</tr>
<tr>
<td>2006</td>
<td>128.13</td>
<td>144.27</td>
<td>0.888</td>
<td>115.42</td>
<td>16.14</td>
<td>18.59</td>
</tr>
<tr>
<td>2007</td>
<td>97.1</td>
<td>101.09</td>
<td>0.961</td>
<td>77.76</td>
<td>3.99</td>
<td>21.37</td>
</tr>
<tr>
<td>2008</td>
<td>155.76</td>
<td>186.55</td>
<td>0.835</td>
<td>138.19</td>
<td>30.79</td>
<td>20.16</td>
</tr>
<tr>
<td>2009</td>
<td>137.36</td>
<td>150.36</td>
<td>0.914</td>
<td>107.40</td>
<td>13.00</td>
<td>22.44</td>
</tr>
<tr>
<td>2010</td>
<td>96.46</td>
<td>101.09</td>
<td>0.954</td>
<td>69.72</td>
<td>4.63</td>
<td>19.11</td>
</tr>
<tr>
<td>Average</td>
<td>108.124</td>
<td>124.03</td>
<td>0.872</td>
<td>103.48</td>
<td>15.91</td>
<td>19.64</td>
</tr>
<tr>
<td>Std Dev</td>
<td>24.9939958</td>
<td>32.39204017</td>
<td>0.772</td>
<td>21.48</td>
<td>12.46</td>
<td>2.04</td>
</tr>
<tr>
<td>Avg Diff / Avg Max</td>
<td>0.124031008</td>
<td>Correl</td>
<td>0.48</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Closing price on coldest day of each year along with maximum price for that year

Weighted HDD Calculation, closing price on coldest day and maximum price for each year

~ 16 ~
The purple line denotes the HDD calculation on the coldest day of the year in question and is mapped on the secondary axis. The blue line denotes the price of a barrel of oil on that day and the red line shows the maximum price of oil for the given year. From the table above, a correlation of 48% is calculated. The major discrepancies in the graph may be attributed to the Iraq war in the period 2003/2004 and the effects of the recent economic downturn may also be seen in 2008. In an attempt to mitigate against these ‘shocks’ I constructed a new table. These ‘shock’ values were replaced an average of the two preceding year max prices and the year immediately following to give the ‘adjusted closing price’. When these changes were taken into account the correlation figure rose to 72%.

<table>
<thead>
<tr>
<th>Date</th>
<th>Adj Close $</th>
<th>Wgt HDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>2001.00</td>
<td>73.56</td>
<td>16.63</td>
</tr>
<tr>
<td>2002.00</td>
<td>89.48</td>
<td>16.11</td>
</tr>
<tr>
<td>2003.00</td>
<td>91.43</td>
<td>20.95</td>
</tr>
<tr>
<td>2004.00</td>
<td>108.26</td>
<td>20.81</td>
</tr>
<tr>
<td>2005.00</td>
<td>103.70</td>
<td>20.25</td>
</tr>
<tr>
<td>2006.00</td>
<td>103.02</td>
<td>18.59</td>
</tr>
<tr>
<td>2007.00</td>
<td>97.10</td>
<td>21.37</td>
</tr>
<tr>
<td>2009.00</td>
<td>137.36</td>
<td>22.44</td>
</tr>
<tr>
<td>2010.00</td>
<td>96.46</td>
<td>19.11</td>
</tr>
<tr>
<td>Average</td>
<td>100.04</td>
<td>19.58</td>
</tr>
<tr>
<td>Std Dev</td>
<td>17.29</td>
<td>2.16</td>
</tr>
<tr>
<td>Correl</td>
<td>0.72</td>
<td></td>
</tr>
</tbody>
</table>

Closing Price adjusted to remove ‘shocks’

Adjusted price along with weighted HDD total

This ‘Adjusted Closing Price’ for a barrel of oil imitates the mapping of the HDD total much more closely.
6. Future Work and Conclusion

To attempt to predict oil prices there are three scenarios:

(1) Correlation of oil prices with ‘USA-wide’ HDD’s
(2) Correlate oil prices with a representative sample as selected above
(3) Correlate oil prices with state-wide HDD’s

Both scenarios 1 and 3 can be calculated using the NOAA calculated monthly data. The work of my project is towards scenario 2 where daily information is taken into account. The correlations proposed in scenarios 1 and 3 would have proved an interesting exercise but unfortunately due to time constraints on this project could not be completed in full.

<table>
<thead>
<tr>
<th>City</th>
<th>City Pop</th>
<th>State Pop</th>
<th>Pop Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anch</td>
<td>286174</td>
<td>710231</td>
<td>40.29%</td>
</tr>
<tr>
<td>Atl</td>
<td>542822</td>
<td>9687653</td>
<td>5.60%</td>
</tr>
<tr>
<td>Chi</td>
<td>2695598</td>
<td>12830632</td>
<td>21.01%</td>
</tr>
<tr>
<td>Hou</td>
<td>2099451</td>
<td>25145561</td>
<td>8.35%</td>
</tr>
<tr>
<td>LA</td>
<td>3833995</td>
<td>37253956</td>
<td>10.29%</td>
</tr>
<tr>
<td>NY</td>
<td>8391881</td>
<td>19378102</td>
<td>43.31%</td>
</tr>
<tr>
<td>Phi</td>
<td>1556396</td>
<td>12702379</td>
<td>12.25%</td>
</tr>
<tr>
<td>Sea</td>
<td>608660</td>
<td>6724540</td>
<td>9.05%</td>
</tr>
<tr>
<td>Average</td>
<td>2501872</td>
<td>15554131.8</td>
<td>18.77%</td>
</tr>
<tr>
<td>St Dev</td>
<td>2669714</td>
<td>11500119.5</td>
<td>0.1</td>
</tr>
</tbody>
</table>

The table above shows the populations of each of the cities, the populations of their respective states and the ratio of city population to state population. Since the cities shown represent 18% of their state population on average, I feel that more data must be presented.

When trying to explore the impact of exogenous variables (those outside the system) on pricing it is important to acknowledge the presence of multiple exogenous variables. When calculating the price of a barrel of oil some examples of exogenous variables are: sentiment, impact of announcements and weather. Both sentiment and impact of announcements are qualitative variables – they are calculated subjectively. Weather is a quantitative variable – it can be measured independently and so must be considered in a category separate to sentiment and impact of announcements.
We know that the price of an asset can be expressed as the weighted sum of \( n \) previous day’s prices plus an error term:

\[
P(t) = \alpha_1 P(t-1) + \alpha_2 P(t-2) + \ldots + \alpha_n P(t-n) + \epsilon_t
\]

Where \( \epsilon_t \) represents the incorporation of associated exogenous variables. However, since we make the point that weather is a separate exogenous variable due to its quantitative nature we re-write the above equation as follows:

\[
P'(t) = \alpha_1 P(t-1) + \alpha_2 P(t-2) + \ldots + \alpha_n P(t-n) + \beta(HDD_t) + \epsilon'_t
\]

Now \( \epsilon'_t \) is the impact of the remaining exogenous variables. As scientists, we want to attempt to eliminate variables such as sentiment from our system due to the vagueness of their nature. The separation of impact of weather is an important step in this regard.

Future work on this project could be done in association with the Enterprise Ireland sponsored project ‘Fair Achean’ which studies the effect of sentiment on market prices. The nature of the work presented and the way in which the scrapers have been written mean it would be a straightforward task to expand the scope of the project to incorporate more cities into the analysis.
Appendix

B. The following data represents the Historical Climatological Series as measured and calculated by the National Oceanic and Atmospheric Administration (NOAA) and is sourced from the following website: http://www.ncdc.noaa.gov/oa/documentlibrary/hcs/hcs.html

This information is also available on the accompanying CD.

C. The following is the source code for the web scrapers for each of the eight cities in my project:

Anchorage Scraper

import urllib2
import re

from BeautifulSoup import BeautifulSoup

# setup proxy, or at least try :/
#proxy_info = {
#    'user' : 'username',
#    'pass' : 'password'',
#    'host' : 'proxyb.tcd.ie',
#    'port' : 8080 }
# build a new opener that uses a proxy requiring authorization
#proxy_support = urllib2.ProxyHandler({"http" : 
#    "http://%(user)s:%(pass)s@%(host)s:%(port)d" % proxy_info})
#opener = urllib2.build_opener(proxy_support, urllib2.HTTPHandler)
# install it
#urllib2.install_opener(opener)

def is_leap_year(y):
    print "Checking if %s is leap year" % y
    if y%400 == 0:
        return True
    elif y%100 == 0:
        return False
    elif y%4 == 0:
        return True
    else:
        return False
return False

f = open('anch-data.txt', 'w')
f.close()
f = open('anch-data.txt', 'a')
# Iterate through year, month, and day
for y in range(2001, 2011): #should be 2010
    for m in range(1, 13): #should be 13
        for d in range(1, 32): #should be 32
            # Check if leap year
            print (d, m, y)
            leap = is_leap_year(y)

            # Check if already gone through month
            print "Checking if month already done"
            if (m == 2 and leap and d > 29):
                continue
            elif (m == 2 and d > 28):
                continue
            elif (m in [4, 6, 9, 10] and d > 30):
                continue

            # Open wunderground.com url
            print "Opening URL"
            url = "http://www.wunderground.com/history/airport/PAMR/" + str(y) + "/" + str(m) + "/" + str(d) + "/DailyHistory.html?req_city=Anchorage&req_state=AK&req_statename=Alaska"

            print "URL open, assigning page "
            page = urllib2.urlopen(url)

            soup = BeautifulSoup(page)

            # Get temperature from page
            print "Getting temperature"
            meanTemp = soup.findAll(attrs={"class" : "b"})
            dayTemp = str(meanTemp[1])
            print "Mean temperature:" , dayTemp

            # Format month for timestamp
            print "Formatting month for time stamp"
            if len(str(m)) < 2:
                mStamp = '0' + str(m)
            else:
                mStamp = str(m)
mStamp = str(m)

# Format day for timestamp
print "Formatting day for timestamp"
if len(str(d)) < 2:
    dStamp = '0' + str(d)
else:
    dStamp = str(d)

# Build timestamp
print "Building timestamp"
timestamp = str(y) + mStamp + dStamp

# Write timestamp and temperature to file
print "Writing timestamp and temp to file"
#f.write(timestamp + ',' + dayTemp + '\n')
f.write(dayTemp + '\n')
print

# Done getting data! Close file.
f.close()

Atlanta Scraper

import urllib2
import re

#setup proxy, or at least try :/
proxy_info = {
    'user': 'username',
    'pass': 'password',
    'host': 'proxyb.tcd.ie',
    'port': 8080
}
# build a new opener that uses a proxy requiring authorization
proxy_support = urllib2.ProxyHandler({'http':
    "http://%s:%s@%s:%d" % (proxy_info['user'],
                            proxy_info['pass'],
                            proxy_info['host'],
                            proxy_info['port'])})
 opener = urllib2.build_opener(proxy_support, urllib2.HTTPHandler)
# install it
urllib2.install_opener(opener)

from BeautifulSoup import BeautifulSoup

def is_leap_year(y):
    print "Checking if %s is leap year" % y
    if y%400 == 0:

~ 22 ~
    if y%100 == 0:
        return False
    elif y%4 == 0:
        return True
    else:
        return False

f = open('atlanta-data.txt', 'w')
f.close()
f = open('atlanta-data.txt', 'a')
# Iterate through year, month, and day
for y in range(2001, 2011): #should be 2011
    for m in range(1, 13): #should be 13
        for d in range(1, 32): #should be 32

            # Check if leap year

            print (d, m, y)
            leap = is_leap_year(y)

            # Check if already gone through month
            print "Checking if month already done"
            if (m == 2 and leap and d > 29):
                continue
            elif (m == 2 and d > 28):
                continue
            elif (m in [4, 6, 9, 10] and d > 30):  
                continue

            # Open wunderground.com url
            print "Opening URL"
            url = "http://www.wunderground.com/history/airport/KPDK/"+str(y)+ "/"+str(m)+"/"+str(d)+"/
/DailyHistory.html?req_city=Atlanta&req_state=GA&req_statename=Georgia"

            print"URL open, assigning page "
            page = urllib2.urlopen(url)

            soup = BeautifulSoup(page)

            # Get temperature from page
            print "Getting temperature"
            meanTemp = soup.findAll(attrs={"class" : "b"})
            dayTemp = str(meanTemp[1])
            print "Mean temperature:" , dayTemp
# Format month for timestamp
print "Formatting month for time stamp"
if len(str(m)) < 2:
    mStamp = '0' + str(m)
else:
    mStamp = str(m)

# Format day for timestamp
print "Formatting day for timestamp"
if len(str(d)) < 2:
    dStamp = '0' + str(d)
else:
    dStamp = str(d)

# Build timestamp
print "Building timestamp"
timestamp = str(y) + mStamp + dStamp

# Write timestamp and temperature to file
print "Writing timestamp and temp to file"
#f.write(timestamp + ',' + dayTemp + '\n')
f.write(dayTemp + '\n')
print

# Done getting data! Close file.
f.close()

Chicago Scraper

import urllib2
import re

from BeautifulSoup import BeautifulSoup

#setup proxy, or at least try :/
#proxy_info = {
    #'user': 'username',
    #'pass': 'password''
    #'host': 'proxyb.tcd.ie',
    #'port': 8080 }
# build a new opener that uses a proxy requiring authorization
#proxy_support = urllib2.ProxyHandler({"http" : \
    #"http://%s:%s@%s:%s%d % proxy_info})
#opener = urllib2.build_opener(proxy_support, urllib2.HTTPHandler)
# install it
#urllib2.install_opener(opener)

def is_leap_year(y):
    print "Checking if %s is leap year" % y
    if y%400 == 0:
        return True
    elif y%100 == 0:
        return False
    elif y%4 == 0:
        return True
    else:
        return False

f = open('chicago-data.txt', 'w')
f.close()
f = open('chicago-data.txt', 'a')
# Iterate through year, month, and day
for y in range(2001, 2011):
    for m in range(1, 13):
        for d in range(1, 32):
            # Check if leap year
            print (d, m, y)
            leap = is_leap_year(y)

            # Check if already gone through month
            print "Checking if month already done"
            if (m == 2 and leap and d > 29):
                continue
            elif (m == 2 and d > 28):
                continue
            elif (m in [4, 6, 9, 10] and d > 30):
                continue

            # Open wunderground.com url
            print "Opening URL"
            url = "http://www.wunderground.com/history/airport/KMDW/"+str(y)+"/"+str(m)+"/"+str(d) 
            +"/DailyHistory.html?req_city=Chicago&req_state=IL&req_statename=Illinois"

            print "URL open, assigning page "
            page = urllib2.urlopen(url)
soup = BeautifulSoup(page)

# Get temperature from page
print "Getting temperature"
meanTemp = soup.findAll(attrs={"class" : "b"})
dayTemp = str(meanTemp[1])
print "Mean temperature:", dayTemp

# Format month for timestamp
print "Formatting month for time stamp"
if len(str(m)) < 2:
    mStamp = '0' + str(m)
else:
    mStamp = str(m)

# Format day for timestamp
print "Formatting day for timestamp"
if len(str(d)) < 2:
    dStamp = '0' + str(d)
else:
    dStamp = str(d)

# Build timestamp
print "Building timestamp"
timestamp = str(y) + mStamp + dStamp

# Write timestamp and temperature to file
print "Writing timestamp and temp to file"
f.write(timestamp + ',' + dayTemp + '
')
f.write(dayTemp + '
')

# Done getting data! Close file.
f.close()

**Houston Scraper**

import urllib2
import re
from BeautifulSoup import BeautifulSoup

#setup proxy, or at least try :/
#proxy_info = {
#'user' : 'username',

~ 26 ~
# 'pass' : 'password''
# 'host' : 'proxyb.tcd.ie'
# 'port' : 8080 

# build a new opener that uses a proxy requiring authorization
#proxy_support = urllib2.ProxyHandler({'http' : 
#"http://%(user)s:%(pass)s@%(host)s:%(port)d" % proxy_info})
#opener = urllib2.build_opener(proxy_support, urllib2.HTTPHandler)

# install it
#urllib2.install_opener(opener)

def is_leap_year(y):
    print "Checking if %s is leap year" % y
    if y%400 == 0:
        return True
    elif y%100 == 0:
        return False
    elif y%4 == 0:
        return True
    else:
        return False

f = open('houston-data.txt', 'w')
f.close()

f = open('houston-data.txt', 'a')

# Iterate through year, month, and day
for y in range(2001, 2011): #should be 2010
    for m in range(1, 13): #should be 13
        for d in range(1, 32): #should be 32
            # Check if leap year
            print (d, m, y)
            leap = is_leap_year(y)

            # Check if already gone through month
            print "Checking if month already done"
            if (m == 2 and leap and d > 29):
                continue
            elif (m == 2 and d > 28):
                continue
            elif (m in [4, 6, 9, 10] and d > 30):
                continue

            # Open wunderground.com url
            print "Opening URL"
url = "http://www.wunderground.com/history/airport/KHOU/\+str(y)+\+"/\+str(m)+\+"/\+str(d)\+"/DailyHistory.html?req_city=Houston&req_state=TX&req_statename=Texas"

print"URL open, assigning page 
page = urllib2.urlopen(url)

soup = BeautifulSoup(page)

 # Get temperature from page
print "Getting temperature"
meanTemp = soup.findAll(attrs="class" : "b")
dayTemp = str(meanTemp[1])
print "Mean temperature: ", dayTemp

 # Format month for timestamp
print "Formatting month for time stamp"
if len(str(m)) < 2:
    mStamp = '0' + str(m)
else:
    mStamp = str(m)

 # Format day for timestamp
print "Formatting day for timestamp"
if len(str(d)) < 2:
    dStamp = '0' + str(d)
else:
    dStamp = str(d)

 # Build timestamp
print "Building timestamp"
timestamp = str(y) + mStamp + dStamp

print "Writing timestamp and temp to file"
f.write(timestamp + ',' + dayTemp + '
')
print(dayTemp + '
')

# Done getting data! Close file.
f.close()

Los Angeles Scraper

import urllib2
import re

# setup proxy, or at least try :/
proxy_info = {
'user': 'username',
'pass': 'password',
'host': 'proxyb.tcd.ie',
'port': 8080}
# build a new opener that uses a proxy requiring authorization
proxy_support = urllib2.ProxyHandler({'http': 
'http://%(user)s:%(pass)s@%(host)s:%(port)d" % proxy_info})
# opener = urllib2.build_opener(proxy_support, urllib2.HTTPHandler)
# install it
# urllib2.install_opener(opener)

from BeautifulSoup import BeautifulSoup

def is_leap_year(y):
    print "Checking if %s is leap year" % y
    if y%400 == 0:
        return True
    elif y%100 == 0:
        return False
    elif y%4 == 0:
        return True
    else:
        return False

f = open('la-data.txt', 'w')
f.close()
f = open('la-data.txt', 'a')
# Iterate through year, month, and day
for y in range(2001, 2011): #should be 2011
    for m in range(1, 13): #should be 13
        for d in range(1, 32): #should be 32
            # Check if leap year
            print (d, m, y)
            leap = is_leap_year(y)

            # Check if already gone through month
            print "Checking if month already done"
            if (m == 2 and leap and d > 29):
                continue
elif (m == 2 and d > 28):  
    continue
elif (m in [4, 6, 9, 10] and d > 30):
    continue

# Open wunderground.com url
print "Opening URL"
url = "http://www.wunderground.com/history/airport/KCQT/\"+str(y)+
"/\" + str(m) + "/\" + str(d) + 
"/DailyHistory.html?req_city=Los+Angeles&req_state=CA&req_statename=California"

print"URL open, assigning page ", page = urllib2.urlopen(url)
soup = BeautifulSoup(page)

    # Get temperature from page
    print "Getting temperature"
    meanTemp = soup.findAll(attrs={"class" : "b"})
    dayTemp = str(meanTemp[1])
    print "Mean temperature:", dayTemp

    # Format month for timestamp
    print "Formatting month for time stamp"
    if len(str(m)) < 2:
        mStamp = '0' + str(m)
    else:
        mStamp = str(m)

    # Format day for timestamp
    print "Formatting day for timestamp"
    if len(str(d)) < 2:
        dStamp = '0' + str(d)
    else:
        dStamp = str(d)

    # Build timestamp
    print "Building timestamp"
timestamp = str(y) + mStamp + dStamp

    # Write timestamp and temperature to file
    print "Writing timestamp and temp to file"
    #f.write(timestamp + ',' + dayTemp + 'n')
f.write(dayTemp + 'n')
print

~ 30 ~
New York Scraper

import urllib2
import re

from BeautifulSoup import BeautifulSoup

# setup proxy, or at least try :/
#proxy_info = {
#    'user' : 'username',
#    'pass' : 'password',
#    'host' : 'proxyb.tcd.ie',
#    'port' : 8080 }
# build a new opener that uses a proxy requiring authorization
#proxy_support = urllib2.ProxyHandler("http:\n#    "http://%(user)s:%(pass)s@%(host)s:%(port)d" % proxy_info))
#opener = urllib2.build_opener(proxy_support, urllib2.HTTPHandler)
#
# install it
#urllib2.install_opener(opener)

def is_leap_year(y):
    print "Checking if %s is leap year" % y
    if y%400 == 0:
        return True
    elif y%100 == 0:
        return False
    elif y%4 == 0:
        return True
    else:
        return False

f = open('ny-data.txt', 'w')
f.close()
f = open('ny-data.txt', 'a')
# Iterate through year, month, and day
for y in range(2001, 2011): #should be 2010
    for m in range(1, 13): #should be 13
        for d in range(1, 32): #should be 32
            # Check if leap year
            print (d, m, y)
leap = is_leap_year(y)

# Check if already gone through month
print "Checking if month already done"
if (m == 2 and leap and d > 29):
    continue
elif (m == 2 and d > 28):
    continue
elif (m in [4, 6, 9, 10] and d > 30):
    continue

# Open wunderground.com url
print "Opening URL"
url = "http://www.wunderground.com/history/airport/KNYC/$y" + str(m) + "/" + str(d) + "/DailyHistory.html?req_city=New+York&req_state=NY&req_statename=New+York"

print"URL open, assigning page "
page = urllib2.urlopen(url)

soup = BeautifulSoup(page)

# Get temperature from page
print "Getting temperature"
meanTemp = soup.findAll(attrs={"class" : "b"})
dayTemp = str(meanTemp[1])
print "Mean temperature:" , dayTemp

# Format month for timestamp
print "Formatting month for time stamp"
if len(str(m)) < 2:
    mStamp = '0' + str(m)
else:
    mStamp = str(m)

# Format day for timestamp
print "Formatting day for timestamp"
if len(str(d)) < 2:
    dStamp = '0' + str(d)
else:
    dStamp = str(d)

# Build timestamp
print "Building timestamp"
timestamp = str(y) + mStamp + dStamp
# Write timestamp and temperature to file
print "Writing timestamp and temp to file"
f.write(timestamp + ',' + dayTemp + '\n')
f.write(dayTemp + '\n')
print

# Done getting data! Close file.
f.close()

Philadelphia Scraper

import urllib2
import re

from BeautifulSoup import BeautifulSoup

#setup proxy, or at least try :/
#proxy_info = {
#  'user' : 'username',
#  'pass' : 'password'',
#  'host' : 'proxyb.tcd.ie',
#  'port' : 8080 }
# build a new opener that uses a proxy requiring authorization
#proxy_support = urllib2.ProxyHandler({'http' : '
#"http://%(user)s:%(pass)s@%(host)s:%(port)d" % proxy_info})
#opener = urllib2.build_opener(proxy_support, urllib2.HTTPHandler)

# install it
#urllib2.install_opener(opener)

def is_leap_year(y):
    print "Checking if %s is leap year" % y
    if y%400 == 0:
        return True
    elif y%100 == 0:
        return False
    elif y%4 == 0:
        return True
    else:
        return False

f = open('philly-data.txt', 'w')
f.close()
f = open('philly-data.txt', 'a')
# Iterate through year, month, and day
for y in range(2001, 2011): #should be 2010

~ 33 ~
for m in range(1, 13): #should be 13
    for d in range(1, 32): #should be 32

        # Check if leap year
        print (d, m, y)
        leap = is_leap_year(y)

        # Check if already gone through month
        print "Checking if month already done"
        if (m == 2 and leap and d > 29):
            continue
        elif (m == 2 and d > 28):
            continue
        elif (m in [4, 6, 9, 10] and d > 30):
            continue

        # Open wunderground.com url
        print "Opening URL"
        url = "http://www.wunderground.com/history/airport/KPNE/"+str(y)+
        "/" + str(m) + "/" + str(d) +
        "/DailyHistory.html?req_city=Philadelphia&req_state=PA&req_statename=Pennsylvania"

        print"URL open, assigning page "
        page = urllib2.urlopen(url)

        soup = BeautifulSoup(page)

        # Get temperature from page
        print "Getting temperature"
        meanTemp = soup.findAll(attrs={"class" : "b"})
        dayTemp = str(meanTemp[1])
        print "Mean temperature:" , dayTemp

        # Format month for timestamp
        print "Formatting month for time stamp"
        if len(str(m)) < 2:
            mStamp = '0' + str(m)
        else:
            mStamp = str(m)

        # Format day for timestamp
        print "Formatting day for timestamp"
        if len(str(d)) < 2:
            dStamp = '0' + str(d)
        else:
dStamp = str(d)

# Build timestamp
print "Building timestamp"
timestamp = str(y) + mStamp + dStamp

# Write timestamp and temperature to file
print "Writing timestamp and temp to file"
#f.write(timestamp + ',' + dayTemp + '\n')
f.write(dayTemp + '\n')
print

# Done getting data! Close file.
f.close()

**Seattle Scraper**

import urllib2
import re

from BeautifulSoup import BeautifulSoup

#setup proxy, or at least try :/
#proxy_info = {
#  'user' : 'username',
#  'pass' : 'password''
#  'host' : 'proxyb.tcd.ie',
#  'port' : 8080 }
# build a new opener that uses a proxy requiring authorization
#proxy_support = urllib2.ProxyHandler({'http' : '
#  http://%(user)s:%(pass)s@%(host)s:%(port)d % proxy_info})
#opener = urllib2.build_opener(proxy_support, urllib2.HTTPHandler)

# install it
#urllib2.install_opener(opener)

def is_leap_year(y):
    print "Checking if %s is leap year" % y
    if y%400 == 0:
        return True
    elif y%100 == 0:
        return False
    elif y%4 == 0:
        return True
    else:
        return False
f = open('seattle-data.txt', 'w')
f.close()
f = open('seattle-data.txt', 'a')

# Iterate through year, month, and day
for y in range(2001, 2011):  # should be 2010
    for m in range(1, 13):  # should be 13
        for d in range(1, 32):  # should be 32

            # Check if leap year
            print (d, m, y)
            leap = is_leap_year(y)

            # Check if already gone through month
            print "Checking if month already done"
            if (m == 2 and leap and d > 29):
                continue
            elif (m == 2 and d > 28):
                continue
            elif (m in [4, 6, 9, 10] and d > 30):
                continue

            # Open wunderground.com url
            print "Opening URL"
            url = "http://www.wunderground.com/history/airport/KBFI/"+str(y)+
"/"+str(m)+"/"+str(d)+"/
"DailyHistory.html?req_city=Seattle&req_state=WA&req_statename=Washington"

            print"URL open, assigning page "
            page = urllib2.urlopen(url)

            soup = BeautifulSoup(page)

            # Get temperature from page
            print "Getting temperature"
            meanTemp = soup.findAll(attrs={"class" : "b"})
            dayTemp = str(meanTemp[1])
            print "Mean temperature:" , dayTemp

            # Format month for timestamp
            print "Formatting month for time stamp"
            if len(str(m)) < 2:
                mStamp = '0' + str(m)
            else:
                mStamp = str(m)
# Format day for timestamp
print "Formatting day for timestamp"
if len(str(d)) < 2:
    dStamp = '0' + str(d)
else:
    dStamp = str(d)

# Build timestamp
print "Building timestamp"
timestamp = str(y) + mStamp + dStamp

# Write timestamp and temperature to file
print "Writing timestamp and temp to file"
#f.write(timestamp + ',' + dayTemp + '\n')
f.write(dayTemp + '\n')
print

# Done getting data! Close file.
f.close()