For this project, I'm aiming to scrape three websites with information useful for understanding import and export patterns for select food commodities:

1. MIT Observatory of Economic Complexity
   - the data used from this site would provide import and export values for the commodities studied in the US Department of Agriculture datasets (R Shiny Project)
2. UNData - an online service created by the United Nations Statistics Division
   - This site contains the original datasets offered by the MIT Observatory of Economic Complexity. While the API's offered by MIT are more streamlined, the UNData has the complete set of data for different product classifications.
3. YCharts and Index Mundi - two services that provide a variety of commodity market data.
   - A key part of the webscraping aspect of the study is to go from comparing metric tons for consumption, production, import, and export, to USD for import, export, and spot prices.

In [11]:
```
from IPython.display import HTML
from bs4 import BeautifulSoup
import requests
import re
import pandas as pd
import json
import sys
import requests
import urllib2
from IPython.display import HTML
```

In [36]:
```
url_import = 'http://atlas.media.mit.edu/sitc/import/1962.2012/usa/all/sh
text = requests.get(url_import).text
```

In [119]:
```
len(text)
```

Out[119]: 20195317

In [44]:
```
#Form of prettify seems to be the same as the original import, in json.
response = urllib2.urlopen('http://atlas.media.mit.edu/sitc/import/1962.2
Pythonsoup = json.loads(response)
```

In [60]:
```
Pythonsoup.values()
```

```
   u'year': 1962.0},
  {u'export_rca': 3.06753,
   u'export_val': 404989000.0,
   u'import_val': 8000.0,
   u'origin_id': u'nausa',
   u'sitc_id': u'107230',
   u'sitc_id_len': 6.0,
   u'year': 1962.0},
  {u'export_rca': 2.19867,
   u'export_val': 2813000.0,
   u'import_val': 0.0,
   u'origin_id': u'nausa',
   u'sitc_id': u'107240',
   u'sitc_id_len': 6.0,
   u'year': 1962.0},
  {u'export_rca': 0.738413,
   u'export_val': 32260000.0,
   u'import_val': 46731000.0,
   u'origin_id': u'nausa',
   u'sitc_id': u'107243',
```

In [90]:
```
#Right now, Pythonsoup is a nested list of dictionaries. Need to convert
#Data frame that can be easily exported as a csv.
Pythonsoup
```

Out[90]:
```
[[{u'export_rca': 0.514179,
   u'export_val': 2529000.0,
   u'import_val': 804000.0,
   u'origin_id': u'nausa',
   u'sitc_id': u'105722',
   u'sitc_id_len': 6.0,
   u'year': 1962.0},
  {u'export_rca': 0.917209,
   u'export_val': 74069000.0,
   u'import_val': 22799000.0,
   u'origin_id': u'nausa',
   u'sitc_id': u'106250',
   u'sitc_id_len': 6.0,
   u'year': 1962.0},
  {u'export_rca': 1.36011,
   u'export_val': 2596000.0,
   u'import_val': 0.0,
   u'origin_id': u'nausa',
   u'sitc_id': u'106280',
```

In [75]:
```
import csv
```

In [89]:
```
Pythonsoup2 = pd.DataFrame(Pythonsoup[0])

Pythonsoup2.head()
```

Out[89]:

	export_rca	export_val	export_val_growth_pct	export_val_growth_pct_5	export_va
0	0.514179	2529000	NaN	NaN	NaN
1	0.917209	74069000	NaN	NaN	NaN
2	1.360110	2596000	NaN	NaN	NaN
3	0.750449	5965000	NaN	NaN	NaN
4	1.712770	33426000	NaN	NaN	NaN

In [ ]:
```

```

In [25]:
```
from collections import defaultdict
import csv
from datetime import date
import itertools
import texttable
import pandas as pd
```

In [101]:
```
response2 = urllib2.urlopen('http://atlas.media.mit.edu/sitc/export/1962.
Pythonsoup3 = json.loads(response2)
print Pythonsoup3
```

```
{u'data': [{u'export_rca': 0.514179, u'origin_id': u'nausa', u'expor
t_val': 2529000.0, u'import_val': 804000.0, u'year': 1962.0, u'sitc_
id_len': 6.0, u'sitc_id': u'105722'}, {u'export_rca': 0.917209, u'or
igin_id': u'nausa', u'export_val': 74069000.0, u'import_val': 227990
00.0, u'year': 1962.0, u'sitc_id_len': 6.0, u'sitc_id': u'106250'},
{u'export_rca': 1.36011, u'origin_id': u'nausa', u'export_val': 2596
000.0, u'import_val': 0.0, u'year': 1962.0, u'sitc_id_len': 6.0, u's
itc_id': u'106280'}, {u'export_rca': 0.750449, u'origin_id': u'naus
a', u'export_val': 5965000.0, u'import_val': 1240000.0, u'year': 196
2.0, u'sitc_id_len': 6.0, u'sitc_id': u'106282'}, {u'export_rca': 1.
71277, u'origin_id': u'nausa', u'export_val': 33426000.0, u'import_v
al': 31837000.0, u'year': 1962.0, u'sitc_id_len': 6.0, u'sitc_id':
u'106289'}, {u'export_rca': 2.38713, u'origin_id': u'nausa', u'expor
t_val': 6551000.0, u'import_val': 1017000.0, u'year': 1962.0, u'sitc
_id_len': 6.0, u'sitc_id': u'106352'}, {u'export_rca': 1.04441, u'or
igin_id': u'nausa', u'export_val': 12617000.0, u'import_val': 146600
0.0, u'year': 1962.0, u'sitc_id_len': 6.0, u'sitc_id': u'106577'},
{u'export_rca': 1.16918, u'origin_id': u'nausa', u'export_val': 8441
000.0, u'import_val': 2519000.0, u'year': 1962.0, u'sitc_id_len': 6.
```

In [102]:
```
Pythonsoup3 = Pythonsoup3.values()
```

In [103]:
```
type(Pythonsoup3)
```

Out[103]: list

In [104]:
```
print(Pythonsoup3)
```

```
[[{u'export_rca': 0.514179, u'origin_id': u'nausa', u'export_val': 2
529000.0, u'import_val': 804000.0, u'year': 1962.0, u'sitc_id_len':
6.0, u'sitc_id': u'105722'}, {u'export_rca': 0.917209, u'origin_id':
u'nausa', u'export_val': 74069000.0, u'import_val': 22799000.0, u'ye
ar': 1962.0, u'sitc_id_len': 6.0, u'sitc_id': u'106250'}, {u'export_
rca': 1.36011, u'origin_id': u'nausa', u'export_val': 2596000.0, u'i
mport_val': 0.0, u'year': 1962.0, u'sitc_id_len': 6.0, u'sitc_id':
u'106280'}, {u'export_rca': 0.750449, u'origin_id': u'nausa', u'expo
rt_val': 5965000.0, u'import_val': 1240000.0, u'year': 1962.0, u'sit
c_id_len': 6.0, u'sitc_id': u'106282'}, {u'export_rca': 1.71277, u'o
rigin_id': u'nausa', u'export_val': 33426000.0, u'import_val': 31837
000.0, u'year': 1962.0, u'sitc_id_len': 6.0, u'sitc_id': u'106289'},
{u'export_rca': 2.38713, u'origin_id': u'nausa', u'export_val': 6551
000.0, u'import_val': 1017000.0, u'year': 1962.0, u'sitc_id_len': 6.
0, u'sitc_id': u'106352'}, {u'export_rca': 1.04441, u'origin_id':
u'nausa', u'export_val': 12617000.0, u'import_val': 1466000.0, u'yea
r': 1962.0, u'sitc_id_len': 6.0, u'sitc_id': u'106577'}, {u'export_r
ca': 1.16918, u'origin_id': u'nausa', u'export_val': 8441000.0, u'im
port_val': 2519000.0, u'year': 1962.0, u'sitc_id_len': 6.0, u'sitc_i
d': u'106631'}, {u'export_rca': 1.40800, u'origin_id': u'nausa', u'e
```

In [105]:
```
Pythonsoup4 = pd.DataFrame(Pythonsoup3[0])

Pythonsoup4.head()
```

Out[105]:

	export_rca	export_val	export_val_growth_pct	export_val_growth_pct_5	export_va
0	0.514179	2529000	NaN	NaN	NaN
1	0.917209	74069000	NaN	NaN	NaN
2	1.360110	2596000	NaN	NaN	NaN
3	0.750449	5965000	NaN	NaN	NaN
4	1.712770	33426000	NaN	NaN	NaN

In [107]:
```
Pythonsoup4.to_csv("Export_SITC", sep='\t')
Pythonsoup2.to_csv("Import_SITC", sep = '\t')
```

Despite trying various different API calls for different trade keys, it seems that the data which was returned for both the import and export calls -- even for various products -- only returns a dataset of export values, for all years featured in the SITC.

In [2]:
```
#Inspecting tables from Index Mundi
Swine = requests.get('http://www.indexmundi.com/commodities/?commodity=po
stat = BeautifulSoup(Swine)
```

In [3]:
```
stat
```

Out[3]:
```
<!DOCTYPE html>
<html>
<head><meta content="en" http-equiv="content-language"/><meta charse
t="utf-8"/><meta content="IE=edge" http-equiv="X-UA-Compatible"/><me
ta content="width=device-width, initial-scale=1.0" name="viewport"/>
<title>
        Swine (pork) - Monthly Price - Commodity Prices - Price Char
ts, Data, and News - IndexMundi
</title><meta content="696085087" property="fb:admins"/><meta conten
t="http://www.indexmundi.com/img/compare-200x200.jpg" property="og:i
mage"/><link href="/s/bootstrap.min.css" rel="stylesheet"/><link hre
f="/s/site.css" rel="stylesheet"/><link href="/s/commodities.2.0.1.c
ss" rel="stylesheet" type="text/css"/>
<style type="text/css">
:focus {
  outline: none;
}
.row {
  margin-right: 0;
```

The above table typifies the data received from index mundi. The table rows seemed to be marked with the tag with descendents. Luckily, the tables from index mundi all show on one page, so scraping should be fairly straightforward.

In [5]:
```
#Inspecting tables from YCharts
YCharts = requests.get('https://ycharts.com/indicators/us_consumer_price_
stat1 = BeautifulSoup(YCharts)
```

In [6]:
```
stat1
```

Out[6]:
```
<!DOCTYPE html>
<html id="ng-app" lang="en" ng-app="indicatorOverviewApp">
<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type"/>
<meta content="IE=Edge" http-equiv="X-UA-Compatible"/>
<meta content="i9iq5Aq8HOr2_ebN6-FKRGJPsoifozi80SykYRy_cNI" name="go
ogle-site-verification"/>
<meta content="jt31FVcT0qMeqwVhuxyhDlXCGNxDrmqTdBk9wE6SqDw=" name="v
erify-v1"/>
<meta content="03E016CD8CFF5D076F411476CB779219" name="msvalidate.0
1"/>
<meta content="51fbe4636e8abe10" name="y_key"/>
<meta content="US Consumer Price Index: Pork historical data, chart
s, stats and more. US Consumer Price Index: Pork is at a current lev
el of 212.18, down from 213.73 last month and down from 229.41 one y
ear ago.  This is a change of -0.73% from last month and -7.51% from
one year ago.." name="description"/>
<title>US Consumer Price Index: Pork (Monthly, SA, Index 1982-84=10
0)</title>
```

Once again, the table in this site is marked with a tag. However, the issue here is that the table goes for multple pages that require clicking on links at the bottom to view. Hence, it's necessary to inspect the url to see what happens when different pages and different "date ranges" are selected.

Base URL: https://ycharts.com/indicators/us_consumer_price_index_pork (https://ycharts.com/indicators/us_consumer_price_index_pork)

URL after page change: https://ycharts.com/indicators/us_consumer_price_index_pork (https://ycharts.com/indicators/us_consumer_price_index_pork)

Uh oh! the URL remains the same despite changing pages. I was hoping, initially, that there could be someting in the URL string that I could write a function for. Unfortunately, this is not the case.

The next step is to inspect the html on the browser itself to see if anything is different in the body -- besides the table row values -- that indicates a page change. Using the Google Chrome inspection tool, I pointed to the "Next" button whcih changes the page and found the element: Prev

Specifically, for the 'First' 'Prev' 'Next' and 'Last' buttons, there is a Last Next

While looking at the Network panel in the inspector toolbar, I clicked the 'next' button to see if any specific URL requests were being made. Here's what I found:

https://ycharts.com/indicators/us_consumer_price_index_pork.json?endDate=01/31/2016&pageNum=7&startDate=01/31/1947 (https://ycharts.com/indicators/us_consumer_price_index_pork.json?

endDate=01/31/2016&pageNum=7&startDate=01/31/1947)

This is quite different from the original URL. Here, there is clearly a page descriptor at 2016&pageNum=7&startDate=01/31/1947. This "pageNum" part of the form is what I need. The response to the click gives the following:

"
<td class="col2
"> 99.90

Data for this Date Range	
Nov. 30, 1986	114.50
Oct. 31, 1986	112.90
Sept. 30, 1986	111.70
Aug. 31, 1986	110.70
July 31, 1986	106.90
June 30, 1986	101.30
May 31, 1986	102.40
April 30, 1986	102.50
March 31, 1986	102.20
Feb. 28, 1986	102.40
Jan. 31, 1986	101.90
Dec. 31, 1985	100.00
Nov. 30, 1985	99.40
Oct. 31, 1985	96.60
Sept. 30, 1985	96.80
Aug. 31, 1985	97.50
July 31, 1985	98.30
June 30, 1985	98.30
May 31, 1985	99.30
April 30, 1985	
March 31, 1985	101.30
Feb. 28, 1985	100.70
Jan. 31, 1985	100.10
Dec. 31, 1984	99.70

Nov. 30, 1984	98.10

Oct. 31, 1984	98.40
Sept. 30, 1984	98.70
Aug. 31, 1984	100.10
July 31, 1984	99.40
June 30, 1984	99.40
May 31, 1984	99.50
April 30, 1984	99.50
March 31, 1984	98.10
Feb. 29, 1984	97.70
Jan. 31, 1984	97.50
Dec. 31, 1983	94.20
Nov. 30, 1983	94.10
Oct. 31, 1983	95.10
Sept. 30, 1983	95.90
Aug. 31, 1983	96.40
July 31, 1983	97.90
June 30, 1983	100.80
May 31, 1983	103.10
April 30, 1983	105.10
March 31, 1983	106.90
Feb. 28, 1983	106.90
Jan. 31, 1983	106.00
Dec. 31, 1982	105.50
Nov. 30, 1982	107.00
Oct. 31, 1982	106.90

"

A table formatted in JSON. What is interesting about the repsonse is that it describes the last page number. By changing each pageNum argument, the response points me directly to that part of the table. With this information in mind, the resulting functions for Y Charts and Index Mundi are described below.

In [123]:
```
#Trying to import the jason
url_import = 'https://ycharts.com/indicators/us_consumer_price_index_pork
text1 = requests.get(url_import).text
text1
```

Out[123]:
```
u"You don't have access to this feature."
```

...And that is what can happen. The Y Charts website requires the user to register to the site. Despite the fact that I am an active member of the site and signed into the site on this browser, the Y Charts website will not allow me to see any page of the table beyond the first. For the moment, I have two choices: A - manually pull each link and each page (13 tables with nearly 30 pages for each table) B - try to feed my password cookie into a function and see if I can still pull the data.

Choice B seems well worth it. We will give it a try, but not right now. Let's move onto Index Mundi.

In [10]:
```
#For Index Mundi, beef values:
from bs4 import BeautifulSoup
import requests
response = requests.get('http://www.indexmundi.com/commodities/?commodity

soup = BeautifulSoup(response)
Mundi =  open("results.txt","w")
table = soup.find_all('table', class_="tblData")
souptable = table[0]
```

In [11]:
```
type(souptable)
```

Out[11]:
```
bs4.element.Tag
```

In [12]:
```
soup_string = str(souptable)
#converting bs4 element into string
```

In [14]:
```
import requests
from bs4 import BeautifulSoup
Date=[]
Price=[]
Change=[]

for i in souptable.findAll('tr'):
    td = i.findAll('td')
    for s, p in enumerate(td):
        if s==0:
            Date.append(p.text)
        if s==1:
            Price.append(p.text)
        if s==2:
            Change.append(p.text)
print type (Price)
```

```
<type 'list'>
```

In [15]:
```
#Right now, the above list is a list of unicode strings.
Date = [item.encode('utf-8') for item in Date]
Price = [item.encode('utf-8') for item in Price]
Change = [item.encode('utf-8') for item in Change]
```

In [16]:
```
Price2 = map(lambda item: float(item),Price)
#Now, the price values -- which are the most important -- are integers.
```

In [18]:
```
import pandas as pd
CPI_beef = pd.DataFrame({'Date' : Date,
 'Price' : Price,
 'Change':Change
  })
```

In [88]:
```
print CPI_beef
```

```
343   13.76 %  Aug 2014  256.80
344    5.19 %  Sep 2014  272.30
345   -1.97 %  Oct 2014  266.93
346   -2.03 %  Nov 2014  261.50
347   -8.38 %  Dec 2014  239.59
348   -3.16 %  Jan 2015  232.02
349   -9.54 %  Feb 2015  209.88
350   -0.99 %  Mar 2015  207.80
351    3.19 %  Apr 2015  214.43
352   -7.10 %  May 2015  199.21
353   -2.09 %  Jun 2015  195.05
354    4.80 %  Jul 2015  204.41
355    3.71 %  Aug 2015  212.00
356   -4.52 %  Sep 2015  202.41
357   -8.82 %  Oct 2015  184.55
358   -4.05 %  Nov 2015  177.07
359   -5.62 %  Dec 2015  167.11
360   -2.67 %  Jan 2016  162.64

[361 rows x 3 columns]
```

In [89]:
```
#let's see how well this imports into a csv...
import csv
CPI_beef.to_csv("CPI_beef", sep='\t')
```

The above process can be applied for the remaining products necessary for analysis from Index Mundi: Coconut Oil, Palm Oil, and Pork(swine).

In [33]:
```
#For Pork(swine)
from bs4 import BeautifulSoup
import requests
response = requests.get('http://www.indexmundi.com/commodities/?commodity
soup = BeautifulSoup(response)
Mundi =  open("results.txt","w")
table = soup.find_all('table', class_="tblData")
souptable = table[0]
```

In [34]:
```
soup_string = str(souptable)
```

In [35]:
```
import requests
from bs4 import BeautifulSoup
Date_Pork =[]
Price_Pork =[]
Change_Pork =[]

for i in souptable.findAll('tr'):
    td = i.findAll('td')
    for s, p in enumerate(td):
        if s==0:
            Date_Pork.append(p.text)
        if s==1:
            Price_Pork.append(p.text)
        if s==2:
            Change_Pork.append(p.text)
```

In [36]:
```
#Right now, the above list is a list of unicode strings.
Date_Pork = [item.encode('utf-8') for item in Date_Pork]
Price_Pork = [item.encode('utf-8') for item in Price_Pork]
Change_Pork = [item.encode('utf-8') for item in Change_Pork]
```

In [37]:
```
Price_Pork2 = map(lambda item: float(item),Price_Pork)
```

In [38]:
```
CPI_Pork = pd.DataFrame({'Date_Pork' : Date_Pork,
 'Price_Pork2' : Price_Pork,
 'Change_Pork':Change_Pork
  })
```

In [120]:
```
import csv
CPI_Pork.to_csv("CPI_Pork", sep='\t')
```

In [3]:
```
#For Coconut Oil
from bs4 import BeautifulSoup
import requests
response = requests.get('http://www.indexmundi.com/commodities/?commodity

soup = BeautifulSoup(response)
Mundi =  open("results.txt","w")
table = soup.find_all('table', class_="tblData")
souptable = table[0]
soup_string = str(souptable)

Date_Coco =[]
Price_Coco =[]
Change_Coco =[]

for i in souptable.findAll('tr'):
    td = i.findAll('td')
    for s, p in enumerate(td):
        if s==0:
            Date_Coco.append(p.text)
        if s==1:
            Price_Coco.append(p.text)
        if s==2:
            Change_Coco.append(p.text)
```

In [4]:
```
Date_Coco = [item.encode('utf-8') for item in Date_Coco]
Price_Coco = [item.encode('utf-8') for item in Price_Coco]
Change_Coco = [item.encode('utf-8') for item in Change_Coco]
```

In [6]:
```
import pandas as pd
CPI_Coco = pd.DataFrame({'Date_Coco' : Date_Coco,
 'Price_Coco2' : Price_Coco,
 'Change_Coco':Change_Coco
  })
```

In [7]:
```
CPI_Coco = pd.DataFrame({'Date_Coco' : Date_Coco,
 'Price_Coco2' : Price_Coco,
 'Change_Coco':Change_Coco
  })
```

In [135]:
```
CPI_Coco.to_csv("CPI_Coco", sep='\t')
```

In [15]:
```
#For Palm Oil
response = requests.get('http://www.indexmundi.com/commodities/?commodity

soup = BeautifulSoup(response)
Mundi =  open("results.txt","w")
table = soup.find_all('table', class_="tblData")
souptable = table[0]
soup_string = str(souptable)

Date_Palm =[]
Price_Palm =[]
Change_Palm =[]

for i in souptable.findAll('tr'):
    td = i.findAll('td')
    for s, p in enumerate(td):
        if s==0:
            Date_Palm.append(p.text)
        if s==1:
            Price_Palm.append(p.text)
        if s==2:
            Change_Palm.append(p.text)

Date_Palm = [item.encode('utf-8') for item in Date_Palm]
Price_Palm = [item.encode('utf-8') for item in Price_Palm]
Change_Palm = [item.encode('utf-8') for item in Change_Palm]

CPI_Palm = pd.DataFrame({'Date_Palm' : Date_Palm,
 'Price_Palm' : Price_Palm,
 'Change_Palm':Change_Palm
  })
```

In [138]:
```
CPI_Palm.to_csv("CPI_Palm", sep='\t')
```

In [42]:
```
#In inspecting the original csv files, there's some cleaning to do
#before they are easily readable in R. Dataframes are:
#CPI_Palm, CPI_Coco, CPI_beef, CPI_pork

print type(CPI_Pork)
print type(CPI_beef)
print type(CPI_Coco)
print type(CPI_Palm)
```

```
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.frame.DataFrame'>
```

In [43]:
```
CPI_Pork.head(10)
```

Out[43]:

	Change_Pork	Date_Pork	Price_Pork2
0	-	Jan 1986	115.06
1	-15.86 %	Feb 1986	96.81
2	-1.69 %	Mar 1986	95.17
3	-4.02 %	Apr 1986	91.34
4	27.26 %	May 1986	116.24
5	17.61 %	Jun 1986	136.71
6	24.53 %	Jul 1986	170.24
7	-1.31 %	Aug 1986	168.01
8	-17.23 %	Sep 1986	139.07
9	-17.97 %	Oct 1986	114.08

In [44]:
```
#removing % sign
CPI_Pork['Change_Pork'] = CPI_Pork['Change_Pork'].map(lambda x:x.rstrip
        (' %'))
```

In [45]:
```
CPI_Pork.head(10)
```

Out[45]:

	Change_Pork	Date_Pork	Price_Pork2
0	-	Jan 1986	115.06
1	-15.86	Feb 1986	96.81
2	-1.69	Mar 1986	95.17
3	-4.02	Apr 1986	91.34
4	27.26	May 1986	116.24
5	17.61	Jun 1986	136.71
6	24.53	Jul 1986	170.24
7	-1.31	Aug 1986	168.01
8	-17.23	Sep 1986	139.07
9	-17.97	Oct 1986	114.08

In [47]:
```
CPI_Pork['Change_Pork'][0] = 0
```

In [48]:
```
CPI_Pork.head(10)
```

Out[48]:

	Change_Pork	Date_Pork	Price_Pork2
0	0	Jan 1986	115.06
1	-15.86	Feb 1986	96.81
2	-1.69	Mar 1986	95.17
3	-4.02	Apr 1986	91.34
4	27.26	May 1986	116.24
5	17.61	Jun 1986	136.71
6	24.53	Jul 1986	170.24
7	-1.31	Aug 1986	168.01
8	-17.23	Sep 1986	139.07
9	-17.97	Oct 1986	114.08

In [49]:
```
#extracting the month
CPI_Pork['month'] = CPI_Pork['Date_Pork'].str.extract('([A-Z]\w{0,})')
CPI_Pork.head(10)
```

Out[49]:

	Change_Pork	Date_Pork	Price_Pork2	month
0	0	Jan 1986	115.06	Jan
1	-15.86	Feb 1986	96.81	Feb
2	-1.69	Mar 1986	95.17	Mar
3	-4.02	Apr 1986	91.34	Apr
4	27.26	May 1986	116.24	May
5	17.61	Jun 1986	136.71	Jun
6	24.53	Jul 1986	170.24	Jul
7	-1.31	Aug 1986	168.01	Aug
8	-17.23	Sep 1986	139.07	Sep
9	-17.97	Oct 1986	114.08	Oct

In [51]:
```
CPI_Pork['Year'] = CPI_Pork['Date_Pork'].str.extract('(\d\d\d\d)')
CPI_Pork.head(10)
#Now month and year are in different columns
```

Out[51]:

	Change_Pork	Date_Pork	Price_Pork2	month	Year
0	0	Jan 1986	115.06	Jan	1986
1	-15.86	Feb 1986	96.81	Feb	1986
2	-1.69	Mar 1986	95.17	Mar	1986
3	-4.02	Apr 1986	91.34	Apr	1986
4	27.26	May 1986	116.24	May	1986
5	17.61	Jun 1986	136.71	Jun	1986
6	24.53	Jul 1986	170.24	Jul	1986
7	-1.31	Aug 1986	168.01	Aug	1986
8	-17.23	Sep 1986	139.07	Sep	1986
9	-17.97	Oct 1986	114.08	Oct	1986

In [53]:
```
CPI_Pork = CPI_Pork.drop('Date_Pork', axis=1 )
CPI_Pork.head(10)
```

Out[53]:

	Change_Pork	Price_Pork2	month	Year
0	0	115.06	Jan	1986
1	-15.86	96.81	Feb	1986
2	-1.69	95.17	Mar	1986
3	-4.02	91.34	Apr	1986
4	27.26	116.24	May	1986
5	17.61	136.71	Jun	1986
6	24.53	170.24	Jul	1986
7	-1.31	168.01	Aug	1986
8	-17.23	139.07	Sep	1986
9	-17.97	114.08	Oct	1986

In [54]:
```
CPI_Pork.to_csv("CPI_Pork2", sep=',')
```

In [56]:
```
#CPI_Pork reads properly in R. Will follow the same process for other thr
#For CPI_beef
CPI_beef['Change'] = CPI_beef['Change'].map(lambda x:x.rstrip
        (' %'))
CPI_beef['Change'][0] = 0
CPI_beef['month'] = CPI_beef['Date'].str.extract('([A-Z]\w{0,})')
CPI_beef['Year'] = CPI_beef['Date'].str.extract('(\d\d\d\d)')
CPI_beef = CPI_beef.drop('Date', axis=1 )
CPI_beef.head(10)
```

Out[56]:

	Change	Price	month	Year
0	0	99.00	Jan	1986
1	-1.01	98.00	Feb	1986
2	1.43	99.40	Mar	1986
3	-5.81	93.62	Apr	1986
4	0.06	93.68	May	1986
5	-2.28	91.54	Jun	1986
6	-4.48	87.44	Jul	1986
7	3.66	90.64	Aug	1986
8	4.68	94.88	Sep	1986
9	0.76	95.60	Oct	1986

In [57]:
```
CPI_beef.to_csv("CPI_beef2", sep=',')
```

In [61]:
```
print CPI_Palm.head(10)
print CPI_Coco.head(10)
```

```
  Change_Palm Date_Palm Price_Palm
0           -  Jan 1986     282.62
1    -17.25 %  Feb 1986     233.86
2    -14.13 %  Mar 1986     200.81
3     -0.41 %  Apr 1986     199.98
4     -1.66 %  May 1986     196.67
5      2.11 %  Jun 1986     200.81
6     -9.05 %  Jul 1986     182.63
7    -10.86 %  Aug 1986     162.79
8      5.08 %  Sep 1986     171.06
9     31.40 %  Oct 1986     224.77
  Change_Coco Date_Coco Price_Coco2
0           -  Jan 1986      380.00
1    -16.32 %  Feb 1986      318.00
2     -7.86 %  Mar 1986      293.00
3     -9.22 %  Apr 1986      266.00
4    -12.41 %  May 1986      233.00
5      9.01 %  Jun 1986      254.00
6     -9.84 %  Jul 1986      229.00
7     -7.86 %  Aug 1986      211.00
8     13.74 %  Sep 1986      240.00
9     42.08 %  Oct 1986      341.00
```

In [16]:
```
#For CPI_Palm
CPI_Palm['Change'] = CPI_Palm['Change_Palm'].map(lambda x:x.rstrip
        (' %'))
CPI_Palm['Change'][0] = 0
CPI_Palm['month'] = CPI_Palm['Date_Palm'].str.extract('([A-Z]\w{0,})')
CPI_Palm['Year'] = CPI_Palm['Date_Palm'].str.extract('(\d\d\d\d)')
CPI_Palm = CPI_Palm.drop('Date_Palm', axis=1 )
CPI_Palm = CPI_Palm.drop('Change_Palm', axis=1)
CPI_Palm.head(10)
```

Out[16]:

	Price_Palm	Change	month	Year
0	282.62	0	Jan	1986
1	233.86	-17.25	Feb	1986
2	200.81	-14.13	Mar	1986
3	199.98	-0.41	Apr	1986
4	196.67	-1.66	May	1986
5	200.81	2.11	Jun	1986
6	182.63	-9.05	Jul	1986
7	162.79	-10.86	Aug	1986
8	171.06	5.08	Sep	1986
9	224.77	31.40	Oct	1986

In [8]:
```
#For CPI_Coco
CPI_Coco['Change'] = CPI_Coco['Change_Coco'].map(lambda x:x.rstrip
        (' %'))
CPI_Coco['Change'][0] = 0
CPI_Coco['month'] = CPI_Coco['Date_Coco'].str.extract('([A-Z]\w{0,})')
CPI_Coco['Year'] = CPI_Coco['Date_Coco'].str.extract('(\d\d\d\d)')
CPI_Coco = CPI_Coco.drop('Date_Coco', axis=1 )
CPI_Coco.head(10)
```

Out[8]:

	Change_Coco	Price_Coco2	Change	month	Year
0	-	380.00	0	Jan	1986
1	-16.32 %	318.00	-16.32	Feb	1986
2	-7.86 %	293.00	-7.86	Mar	1986
3	-9.22 %	266.00	-9.22	Apr	1986
4	-12.41 %	233.00	-12.41	May	1986
5	9.01 %	254.00	9.01	Jun	1986
6	-9.84 %	229.00	-9.84	Jul	1986
7	-7.86 %	211.00	-7.86	Aug	1986
8	13.74 %	240.00	13.74	Sep	1986
9	42.08 %	341.00	42.08	Oct	1986

In [12]:
```
CPI_Coco = CPI_Coco.drop('Change_Coco', axis=1 )
CPI_Coco.head(10)
```

Out[12]:

	Price_Coco2	Change	month	Year
0	380.00	0	Jan	1986
1	318.00	-16.32	Feb	1986
2	293.00	-7.86	Mar	1986
3	266.00	-9.22	Apr	1986
4	233.00	-12.41	May	1986
5	254.00	9.01	Jun	1986
6	229.00	-9.84	Jul	1986
7	211.00	-7.86	Aug	1986
8	240.00	13.74	Sep	1986
9	341.00	42.08	Oct	1986

In [13]:
```
CPI_Coco.to_csv("CPI_Coco2", sep=',')
```

In [17]:
```
CPI_Palm.to_csv("CPI_Palm2", sep=',')
```

In [ ]:

In [ ]: