A study of drug spending and prices in the United States

Note: Please install mpld3 using pip install mpld3 for the graphs to be interactive, otherwise they can be a little convoluted

This project will study the increases in drug prices in the United States in the years 2011 - 2015. With one of the word's most expensive costs for healthcare, the public often relies on Medicare programs in order to finance their health spending. While these programs help lower acquisition costs and co-pays, they can also present a heavy burden on the budget of the country, as funds need to be redirected towards purchasing drugs. Fortunately, the government has the power, influence, and size to significantly reduce the purchasing costs of many drugs, such that the price per unit is lower than if, say, bought at a pharmacy.

The cost structure of Medicare is of interest to me, and I wanted to take a further look into how prices rise, or remain flat for an entitiy like the U.S. government. Furthermore, I wonder if the agreements between drug companies and the government have anything to do with the number of beneficiaries of the drug. Perhaps, the most common drugs are the most stable as the sheer size and volume of purchases is all that is needed to maintain profitability. But will that mean that the government is subject to price hikes on drugs with only a few thousand, or hundred beneficiaries?

To attempt to answer this question, I will be using the Medicare Drug Spending Data, collected by Centers for Medicare & Medicaid Services, which can be found in the CMS page (https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Information-on-Prescription-Drugs/2015MedicareData.html)

Data Report:

As mentioned above, the data will be pulled from the CMS page. The variables in the table are the following: HCPCS Code, HCPS Description,Claim Count, Total Spending, Beneficiary Count, Total Annual Spending, Unit Count, Average Cost per Unit, and Average Beneficiary Cost Share for the years between 2011 and 2015.

Variable Description

- `HCPCS Code:` Healthcare Common Procedure Coding System (HCPCS) code for the drug.
- `HCPCS Description:` HCPCS description of the drug.
- `Claim Count:` Number of Medicare Part B claims.
- `Total Spending:` Aggregate drug spending for the Part B program.
- `Beneficiary Count:` Number of Medicare Part B fee-for-service beneficiaries utilizing the drug.
- `Total Annual Spending per User:` Total Spending divided by the number of unique beneficiaries utilizing the drug (Beneficiary Count) during the benefit year.
- `Unit Count:` Total dosage units of medication billed during the calendar year (e.g. number of tablets, grams, milliliters or other units).
- `Average Cost per Unit:` Total Spending divided by the number of dosage units.
- `Average beneficiary Cost Share:` Average amount that beneficiaries using the drug paid out of pocket during the year.
- `Annual Change in Average cost per unit (2015 only):` Annual change in average cost per unit reflects the percent change in average cost per unit between 2014 and 2015. The average cost per unit is calculated for each year at the HCPCS level by dividing the total payment by total units and then a percentage change in unit costs between the two years is calculated. Available for drugs utilized by more than 5,000 beneficiaries in 2014 and 2015.

Note: For the variable Annual Change in Average Cost per unit, it is only available for the 2015 year, however I will instead calculate the difference myself based on the Average Cost Per unit across the years

My Packages

I will be using the following packages

- `Pandas` package for importing, manipulating, and analazing data
- `Matplotlib` for plotting data
- `Mpld3` please install this using "pip install mpld3". This is important because I will be plotting 60 lines, with this I can make the graphs interactive
- `Requests`, `Zipfile` and `IO` are packages necessary for me to retrieve the data, open the zip file, and import it into a dataframe
- `Numpy` will be used to create arrays and do calculation within the array, since Pandas was giving me an error regarding creating a new column based on the values of other columns

The Project

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt, mpld3 #Will be used to plot
import requests, zipfile, io #This helps retrieve the zip file from the
 page and extract it in oder to read it
import numpy as np #Will be needed for some of the array stuff

plt.rcParams['font.size'] = 12.5 #This sets a global fontsize for all la
bels
```

Introductory Data: Putting the U.S. Federal Budget Into Perspective

Before we get started with the project, I want to highlight the importance of the questions I am asking. To do this, I have created a graph to put it all into perspective

Retrieving the Data: Putting the U.S. Federal Budget Into Perspective

In [2]:
```
#Here I am reading a CSV, getting only the rows that matter and finally puttin
g it into a DataFrame
url = 'https://www.usgovernmentspending.com/rev/usgs_downbudget_curr.php?span=
usgs302&year=2012_2017&fy=fy19&view=2&expand=&expandC=&units=b&csv=&actual='
budget = pd.read_csv(url, skiprows = 3, nrows = 10)
budget = budget[['Year','FY 2015 Outlays']]

#Here is a look at the resulting Dataframe:
print(budget)
print(budget.dtypes)
```

```
                Year FY 2015 Outlays
0            Pensions           953.6
1         Health Care         1,028.4
2           Education           133.8
3             Defense           798.0
4             Welfare           361.9
5          Protection            34.0
6      Transportation            89.5
7  General Government            43.7
8      Other Spending            22.4
9            Interest           223.2
Year               object
FY 2015 Outlays    object
dtype: object
```

The CSV files have everything as strings, which makes it hard to perform calculations, so first we have to clean up the data a little bit, before converting to a float from a string we must remove the commas from numbers like 1,028.40

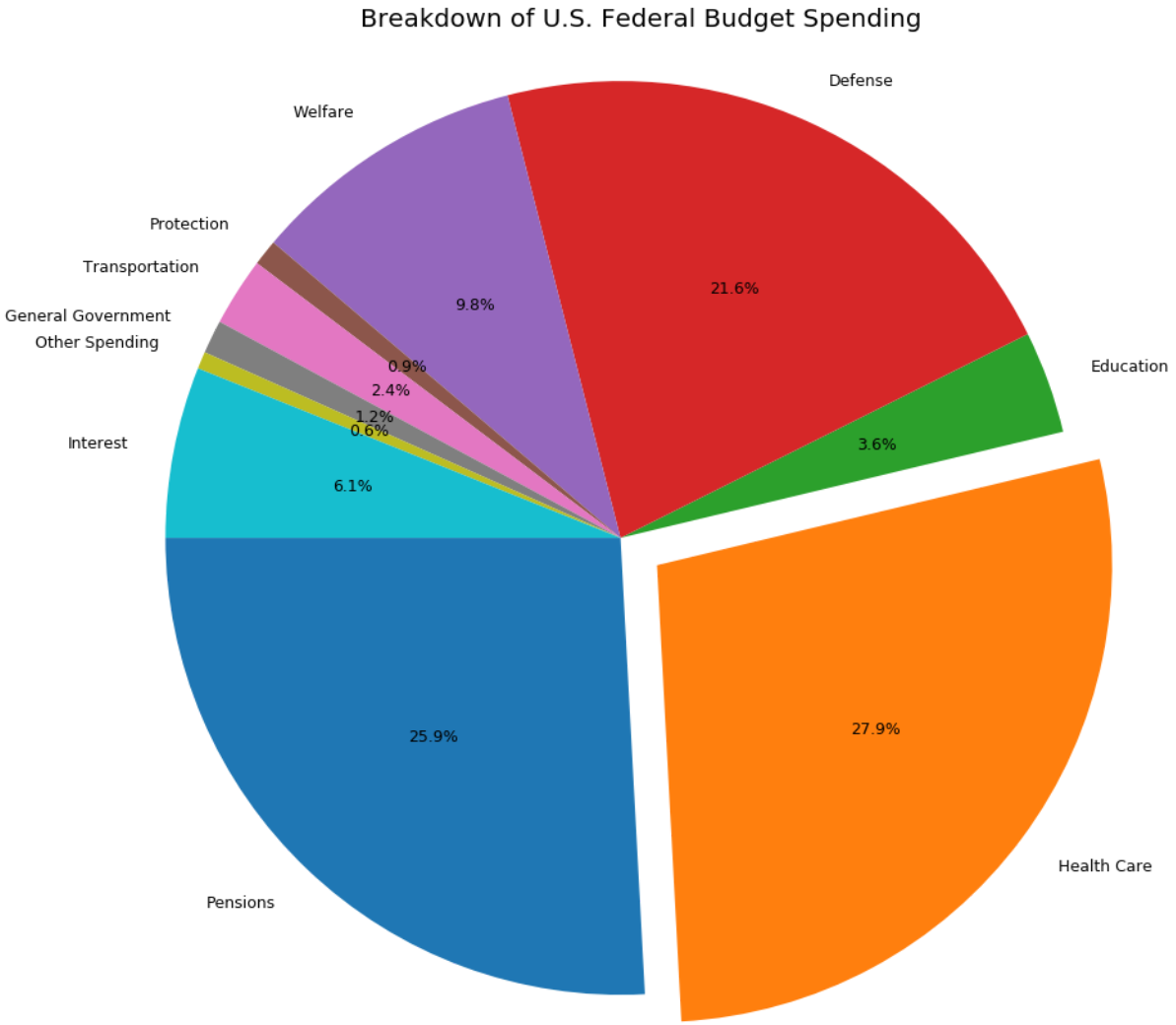
Manipulating the data: Putting the U.S. Federal Budget Into Perspective

In [3]:
```
#The CSV files have everything as strings, which makes it hard to perform calc
ulations,
#so first we have to clean up the data a little bit, before converting to a fl
oat from a string
#we must remove the commas from numbers like 1,028.40
for i in range (0,10):
    budget.iloc[i,1] = budget.iloc[i,1].replace(',', '')

budget['FY 2015 Outlays'] = budget['FY 2015 Outlays'].astype(float)
budget.rename(index=str, columns={"Year": "Sector", "FY 2015 Outlays": "2015"
}, inplace = True)

#Now the data is looking nicely, and has the correct formatting, here is a loo
k:
print(budget)
print(budget.dtypes)
```

```
              Sector    2015
0            Pensions   953.6
1         Health Care  1028.4
2           Education   133.8
3             Defense   798.0
4             Welfare   361.9
5          Protection    34.0
6      Transportation    89.5
7  General Government    43.7
8      Other Spending    22.4
9            Interest   223.2
Sector     object
2015      float64
dtype: object
```

Plotting the Data: Putting the U.S. Federal Budget Into Perspective

In [4]:
```
labels = budget['Sector']
explode = (0, 0.1, 0, 0, 0, 0, 0, 0, 0, 0)
fig1, ax1 = plt.subplots(figsize = (12,12))
ax1.pie(budget['2015'], explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=False, startangle=180)

ax1.axis('equal')
plt.title('Breakdown of U.S. Federal Budget Spending',fontsize = 20)
plt.tight_layout()
plt.show()
```



As we can see, Health Care spending accounts for 27.9% of all Federal spending (not counting military spending, which is not included here). With such a big portion of the federal budget being spent on this, the government must surely be affected by price changes in medications that it purchases through its Medicaid programs

Analysis of Drug Prices

With an understanding of the importance of Medicare spending to the Federal Budget of the United States, we can get started on analyzing one of the areas of spending within this sector, drug purchases for Medicare beneficiaries. If you're not familiar with the Medicare program, I just want to point out that sometimes these medication purchases require a co-pay, that is, the government pays for a portion of the drug, and the beneficiary pays for the remaining share of the cost. Since I am only interested in the U.S. Budget Data, however, I will only be looking at the U.S. consumption.

Importing the Data

In [5]:
```
r = requests.get('https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Information-on-Prescription-Drugs/Downloads/Part_B_All_Drugs_2015.zip')
z = zipfile.ZipFile(io.BytesIO(r.content))
z.extractall()

#The filename once downloaded will become this:
filename = "Medicare_Drug_Spending_PartB_All_Drugs_YTD_2015_12_06_2016.xlsx"
```

Now that the data is in the directory where we are currently working, we can now put it into a dataframe. The file has three sheets, and we're only interested in the third, named ***Data***. Furthermore, this sheet also has some rows that we need to skip in order to get to the actual table, as well as some rows with empty values.

Below, I make sure to account for that when reading the data

In [6]:
```
#The parameters sheetname and skiprows ensure we get eactly what we want, and
 the method dropna() ensures we only get populated data
data = pd.read_excel(filename, sheetname= 'Data', skiprows = 3)
# I choose to only work with drugs that have all the data available, so I drop
 all rows with NaN values
data.dropna(inplace= True)
data
```

Out[6]:

	HCPCS Description	HCPCS Code	Claim Count, 2011	Total Spending, 2011	Beneficiary Count, 2011	Total Annual Spending Per User, 2011	U
0	5% dextrose/normal saline (500 ml = 1 unit)	J7042	53117.0	2.835140e+04	19222.0	1.474945	76
1	5% dextrose/water (500 ml = 1 unit)	J7060	81401.0	1.383810e+05	18668.0	7.412737	12
3	Albuterol, inhalation solution, fda-approved f...	J7611	40863.0	3.957224e+05	19915.0	19.870568	43
4	Albuterol, inhalation solution, fda-approved f...	J7613	1882064.0	2.824052e+07	537176.0	52.572185	45
5	Albuterol, up to 2.5 mg and ipratropium bromid...	J7620	1770476.0	4.511230e+07	403021.0	111.935361	18
9	Aminolevulinic acid hcl for topical administra...	J7308	90967.0	1.513499e+07	60167.0	251.549621	10
14	Arformoterol, inhalation solution, fda approve...	J7605	298228.0	9.400010e+07	56096.0	1675.700614	17
17	Bcg (intravesical) per instillation	J9031	153086.0	1.827720e+07	28873.0	633.020567	15
18	Budesonide, inhalation solution, fda-approved ...	J7626	756310.0	1.981164e+08	142240.0	1392.831498	44
21	Capecitabine, 500 mg	WW093	110836.0	1.977389e+08	27063.0	7306.613039	83
27	Cyclophosphamide, 100 mg	J9070	104111.0	1.123902e+07	24880.0	451.728927	11
31	Cyclosporine, oral, 100 mg	J7502	57797.0	1.187167e+07	7693.0	1543.178716	36
32	Cyclosporine, oral, 25 mg	J7515	99910.0	1.223813e+07	12443.0	983.535250	14

	HCPCS Description	HCPCS Code	Claim Count, 2011	Total Spending, 2011	Beneficiary Count, 2011	Total Annual Spending Per User, 2011	U
55	Formoterol fumarate, inhalation solution, fda ...	J7606	169474.0	4.762261e+07	31558.0	1509.050230	10
68	Hyaluronan or derivative, euflexxa, for intra-...	J7323	192138.0	3.644276e+07	61637.0	591.248041	25
70	Hyaluronan or derivative, hyalgan or supartz, ...	J7321	558654.0	6.590216e+07	128758.0	511.829646	73
72	Hyaluronan or derivative, orthovisc, for intra...	J7324	221264.0	4.825383e+07	68538.0	704.044896	28
73	Hyaluronan or derivative, synvisc or synvisc-o...	J7325	315198.0	1.384324e+08	174192.0	794.711849	11
76	Influenza virus vaccine, split virus, when adm...	Q2035	256455.0	3.175953e+06	255452.0	12.432679	25
77	Influenza virus vaccine, split virus, when adm...	Q2036	2545961.0	2.361530e+07	2533051.0	9.322868	25
78	Influenza virus vaccine, split virus, when adm...	Q2037	4471424.0	5.952555e+07	4443728.0	13.395408	44
79	Influenza virus vaccine, split virus, when adm...	Q2038	4138334.0	5.317694e+07	4102493.0	12.962103	41
81	Infusion, albumin (human), 25%, 50 ml	P9047	34562.0	1.090595e+07	11899.0	916.543791	15
82	Infusion, albumin (human), 5%, 250 ml	P9045	14907.0	1.261827e+07	5528.0	2282.610682	18

	HCPCS Description	HCPCS Code	Claim Count, 2011	Total Spending, 2011	Beneficiary Count, 2011	Total Annual Spending Per User, 2011	U
87	Infusion, normal saline solution , 1000 cc	J7030	289847.0	2.540921e+05	101628.0	2.500217	35
88	Infusion, normal saline solution , 250 cc	J7050	1188428.0	5.679271e+05	300028.0	1.892914	19
89	Infusion, normal saline solution, sterile (500...	J7040	444497.0	3.246159e+05	114444.0	2.836460	56
94	Injection infliximab, 10 mg	J1745	334376.0	9.307639e+08	58402.0	15937.192919	15
95	Injection, abatacept, 10 mg (code may be used ...	J0129	161144.0	2.519842e+08	19775.0	12742.564365	12
119	Injection, alteplase recombinant, 1 mg	J2997	77025.0	2.066632e+07	49739.0	415.495357	51
...	...	...	...	...	...	...	...
518	Injection, triamcinolone hexacetonide, per 5mg	J3303	65505.0	4.767753e+05	45617.0	10.451701	33
520	Injection, triptorelin pamoate, 3.75 mg	J3315	40291.0	2.518152e+07	19071.0	1320.409243	13
523	Injection, vancomycin hcl, 500 mg	J3370	96814.0	1.068576e+06	15908.0	67.172252	38
529	Injection, vitamin b-12 cyanocobalamin, up to...	J3420	2746049.0	1.084689e+06	641776.0	1.690137	27
543	Insulin for administration through dme (i.e., ...	J1817	61165.0	1.744131e+07	14205.0	1227.829031	63

	HCPCS Description	HCPCS Code	Claim Count, 2011	Total Spending, 2011	Beneficiary Count, 2011	Total Annual Spending Per User, 2011	U
548	Ipratropium bromide, inhalation solution, fda-...	J7644	529426.0	7.170555e+06	144562.0	49.601939	27
549	Leuprolide acetate (for depot suspension), 7.5...	J9217	372193.0	2.736109e+08	160231.0	1707.602808	12
553	Levalbuterol, inhalation solution, fda-approve...	J7614	90287.0	3.365871e+06	36490.0	92.240918	13
554	Low osmolar contrast material, 100-199 mg/ml i...	Q9965	77434.0	3.541845e+06	53699.0	65.957379	34
555	Low osmolar contrast material, 200-299 mg/ml i...	Q9966	227356.0	1.165651e+06	123031.0	9.474448	38
556	Low osmolar contrast material, 300-399 mg/ml i...	Q9967	1132730.0	1.534395e+07	843379.0	18.193417	91
563	Methotrexate sodium, 5 mg	J9250	89422.0	6.111302e+04	6832.0	8.945114	32
570	Mycophenolate mofetil, oral, 250 mg	J7517	350480.0	7.980222e+07	44375.0	1798.359820	60
571	Mycophenolic acid, oral, 180 mg	J7518	155908.0	8.996050e+07	20244.0	4443.810426	27
579	Pneumococcal vaccine for injection into muscle	90670	6331.0	6.307843e+05	5958.0	105.871823	63
581	Prednisone, oral, per 5mg	J7506	195976.0	2.746911e+05	30801.0	8.918254	92
591	Ringers lactate infusion, up to 1000 cc	J7120	35521.0	3.904597e+04	16557.0	2.358276	37
593	Sirolimus, oral, 1 mg	J7520	65106.0	4.884078e+07	8636.0	5655.486046	46

	HCPCS Description	HCPCS Code	Claim Count, 2011	Total Spending, 2011	Beneficiary Count, 2011	Total Annual Spending Per User, 2011	U
595	Tacrolimus, immediate release, oral, 1 mg	J7507	537773.0	2.257975e+08	61938.0	3645.540648	80
603	Tetanus toxoid injection into muscle	90703	32251.0	8.907703e+05	32157.0	27.700665	32
611	Vaccine for Hepatitis B (3 dose schedule) for ...	90740	97964.0	1.197206e+07	46119.0	259.590651	99
613	Vaccine for Hepatitis B (4 dose schedule) for ...	90747	142635.0	1.767117e+07	62862.0	281.110535	14
614	Vaccine for Hepatitis B adult dosage (3 dose s...	90746	62511.0	3.740608e+06	40762.0	91.767041	64
621	Vaccine for influenza for injection into muscle	90662	1664649.0	4.915491e+07	1659199.0	29.625687	16
622	Vaccine for influenza for injection into muscl...	90656	1937494.0	2.423359e+07	1923163.0	12.600905	19
628	Vaccine for influenza injection into skin	90654	14480.0	2.472541e+05	14461.0	17.097997	14
631	Vaccine for pneumococcal polysaccharide for in...	90732	1842184.0	9.314460e+07	1820534.0	51.163341	18
633	Vaccine for tetanus and diphtheria toxoids inj...	90714	28275.0	5.404593e+05	28226.0	19.147571	28
634	Vaccine for tetanus, diphtheria toxoids and ac...	90715	8755.0	3.894092e+05	8694.0	44.790566	87

	HCPCS Description	HCPCS Code	Claim Count, 2011	Total Spending, 2011	Beneficiary Count, 2011	Total Annual Spending Per User, 2011	U
637	Vincristine sulfate, 1 mg	J9370	41758.0	3.112500e+05	10971.0	28.370244	78(

161 rows × 38 columns

Manipulating the data

There are a lot of rows, which can make the data plotting too convoluted. Since I am particularly interested in the price changes depending on the number of users, I will take the top 30 most used drugs and the least 30 most used drugs, that way, I can compare how prices are increased depending on number of users.

Below, I get two new Dataframes (top 30, bottom 30), using Beneficiary Count, 2011 as an indicator of number of users and then I sort the data using that column. After I get the data I want to focus on, I merge the two new dataframes to create a new one, which should have a total of 60 rows comprising just the data I want.

In [7]:
```
data = data.sort_values('Beneficiary Count, 2011', ascending=False) #Sorting t
he data by Beneficiary Count
data_top = data.head(n=30) #Getting the most used drugs into a DataFrame
data_bottom = data.tail(n=30) #Getting the least used drugs into a DataFrame

#Merging the dataframes to create a single dataframe with the right data
frames = [data_top, data_bottom]
drug_data = pd.concat(frames)

#Now for this specific plot, I will only be needing some columns so I create a
nother dataframe, keeping the old one in case I
#decide to use the data
price_data = drug_data[['HCPCS Description','Average Cost Per Unit, 2011','Ave
rage Cost Per Unit, 2012',
                        'Average Cost Per Unit, 2013','Average Cost Per Unit,
 2014',
                        'Average Cost Per Unit, 2015']]
price_data.columns = ['Name', '2011','2012','2013','2014','2015'] #Renaming th
e columns for ease of use

#Resetting the index, not actually needed but it makes the DataFrame look nice
r,
#also, if someone wants to access a specific value using .loc[row, column], re
setting the indexes allows for easy access
#this is preferrable to using the Name as the index, the numbers will make def
initely help with retrival of specific entries
price_data.reset_index(drop=True, inplace = True)
```

Finally, we have a DataFrame! It looks like this:

In [8]:
```
price_data
```

Out[8]:

	Name	2011	2012	2013	2014	2015
0	Influenza virus vaccine, split virus, when adm...	13.311594	13.634089	13.859630	14.197731	15.079521
1	Influenza virus vaccine, split virus, when adm...	12.847727	12.228661	11.797035	11.915256	12.394812
2	Influenza virus vaccine, split virus, when adm...	9.238225	10.241759	9.446370	9.374694	7.430369
3	Vaccine for influenza for injection into muscl...	12.431823	12.434328	12.286621	13.508952	12.696957
4	Vaccine for pneumococcal polysaccharide for in...	50.543199	58.191674	63.694286	67.996518	71.578122
5	Vaccine for influenza for injection into muscle	29.527399	30.144199	30.347737	31.472941	34.516282
6	Injection, triamcinolone acetonide, not other...	1.620285	1.683443	1.744921	1.750780	1.748570
7	Injection, methylprednisolone acetate, 40 mg	2.759352	3.476352	2.828619	2.948240	3.517409
8	Injection, regadenoson, 0.1 mg	51.844593	52.447633	52.788253	52.009950	51.964209
9	Injection, methylprednisolone acetate, 80 mg	6.916163	6.722732	5.481541	5.637354	6.527013
10	Injection, dexamethasone sodium phosphate, 1mg	0.090043	0.116342	0.110009	0.132792	0.141847
11	Low osmolar contrast material, 300-399 mg/ml i...	0.166896	0.136751	0.162199	0.182210	0.150818

	Name	2011	2012	2013	2014	2015
12	Injection, betamethasone acetate 3mg and betam...	5.685841	5.507060	5.555766	5.593230	5.778574
13	Injection, vitamin b-12 cyanocobalamin, up to...	0.391317	0.546809	1.159332	2.064908	2.876492
14	Injection, ceftriaxone sodium, per 250 mg	0.932666	0.823591	0.877528	0.710678	0.713137
15	Albuterol, inhalation solution, fda-approved f...	0.061902	0.059478	0.050351	0.051076	0.064177
16	Albuterol, up to 2.5 mg and ipratropium bromid...	0.239385	0.252469	0.195211	0.180748	0.166854
17	Injection, ketorolac tromethamine, per 15 mg	0.294007	0.255936	0.420809	0.343357	0.623355
18	Injection, gadolinium-based magnetic resonance...	2.177379	1.967424	1.821699	1.934095	1.915845
19	Infusion, normal saline solution , 250 cc	0.294587	0.302621	0.301261	0.336679	0.395837
20	Influenza virus vaccine, split virus, when adm...	12.382118	11.812216	11.582643	13.104589	14.812218
21	Hyaluronan or derivative, synvisc or synvisc-o...	12.019348	12.247995	12.233612	12.275477	12.514762
22	Injection, bevacizumab, 10 mg	60.155896	61.076517	63.147275	64.781418	67.503242
23	Injection, methylprednisolone acetate, 20 mg	1.379760	3.060622	3.016997	3.244253	2.943742

	Name	2011	2012	2013	2014	2015
24	Injection, palonosetron hcl, 25 mcg	18.653788	18.595569	18.948947	19.226327	20.028187
25	Injection, diphenhydramine hcl, up to 50 mg	0.753932	0.773767	0.736658	0.610415	0.590520
26	Injection, dipyridamole, per 10 mg	0.948390	0.813203	0.849095	0.802430	0.787890
27	Leuprolide acetate (for depot suspension), 7.5...	211.092048	214.992902	205.190197	215.699859	227.980932
28	Ipratropium bromide, inhalation solution, fda-...	0.262628	0.261688	0.235069	0.231347	0.219651
29	Budesonide, inhalation solution, fda-approved ...	4.490388	4.856332	5.199145	5.162385	5.277954
30	Injection, etoposide, 10 mg	0.724221	0.786128	0.731941	0.685900	0.634344
31	Vincristine sulfate, 1 mg	3.988063	4.208468	4.520071	5.506797	4.784853
32	Injection, morphine sulfate, up to 10 mg	1.087204	1.068428	2.552596	1.289760	0.709994
33	Injection, cetuximab, 10 mg	50.032358	50.312373	51.482779	52.010573	52.433723
34	Injection, baclofen, 10 mg	199.459019	188.280996	178.366773	171.789004	174.182478
35	Injection, penicillin g benzathine, 100,000 un...	3.916559	4.274664	4.859783	5.501004	6.677723
36	Injection, octreotide, depot form for intramus...	116.362952	123.161115	131.059417	129.954926	147.696383
37	Vaccine for tetanus, diphtheria toxoids and ac...	44.473410	38.005193	34.005234	33.496984	31.741343
38	Sirolimus, oral, 1 mg	10.497785	11.174396	12.419446	14.794649	13.708494

	Name	2011	2012	2013	2014	2015
39	Injection, azacitidine, 1 mg	5.197859	5.346777	5.613893	4.639791	3.303712
40	Injection, tobramycin sulfate, up to 80 mg	2.345248	2.368884	3.021805	2.613959	2.537253
41	Injection, omalizumab, 5 mg	20.874362	22.404553	24.094072	26.102412	28.232003
42	Injection, immune globulin, (gamunex-c/gammake...	37.324404	37.607625	38.552337	39.371625	39.500690
43	Injection, degarelix, 1 mg	2.713355	2.926426	3.067460	3.328677	3.450654
44	Cyclosporine, oral, 100 mg	3.227323	3.314631	3.351463	3.153225	3.021765
45	Injection, paclitaxel protein-bound particles,...	9.396080	9.522704	9.400029	9.458445	9.569157
46	Injection, natalizumab, 1 mg	10.260730	11.150725	12.446929	14.148191	15.702195
47	Methotrexate sodium, 5 mg	0.186849	0.185300	0.222223	0.228757	0.231738
48	Injection, immune globulin (privigen), intrave...	34.679049	34.395428	35.744179	36.303776	37.008697
49	Injection, orphenadrine citrate, up to 60 mg	5.710032	7.058320	6.192024	5.834078	4.736240
50	Pneumococcal vaccine for injection into muscle	99.602766	131.679052	135.787245	143.651573	158.792777
51	Injection, ertapenem sodium, 500 mg	27.692368	29.952625	31.654087	33.786927	37.200940
52	Injection, levoleucovorin calcium, 0.5 mg	1.645979	1.773138	1.782794	1.707135	1.772483

	Name	2011	2012	2013	2014	2015
53	Infusion, albumin (human), 5%, 250 ml	66.587889	61.681871	59.699198	67.067426	53.666912
54	Injection, tocilizumab, 1 mg	3.469278	3.497406	3.619055	3.628976	3.755251
55	Injection, gadobenate dimeglumine (multihance ...	2.282894	2.155508	2.080733	1.546349	2.092472
56	Injection, dexamethasone, intravitreal implant...	193.461138	191.847055	188.535293	195.983851	197.533300
57	Injection, thyrotropin alpha, 0.9 mg, provided...	1001.295581	1018.420761	1059.124228	1271.914296	1378.927072
58	Injection, collagenase, clostridium histolytic...	37.389870	37.377708	37.583695	37.669496	37.532456
59	Injection, immune globulin, (octagam), intrave...	51.396460	37.247978	31.141144	32.131350	39.190093

Plotting the Data

With the data in a DataFrame, I can now begin plotting. I was having some trouble adding new calculated columns. I was specifically getting this error:

A value is trying to be set on a copy of a slice from a DataFrame. Try using .loc[row_indexer,col_indexer] = value instead See the the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy)

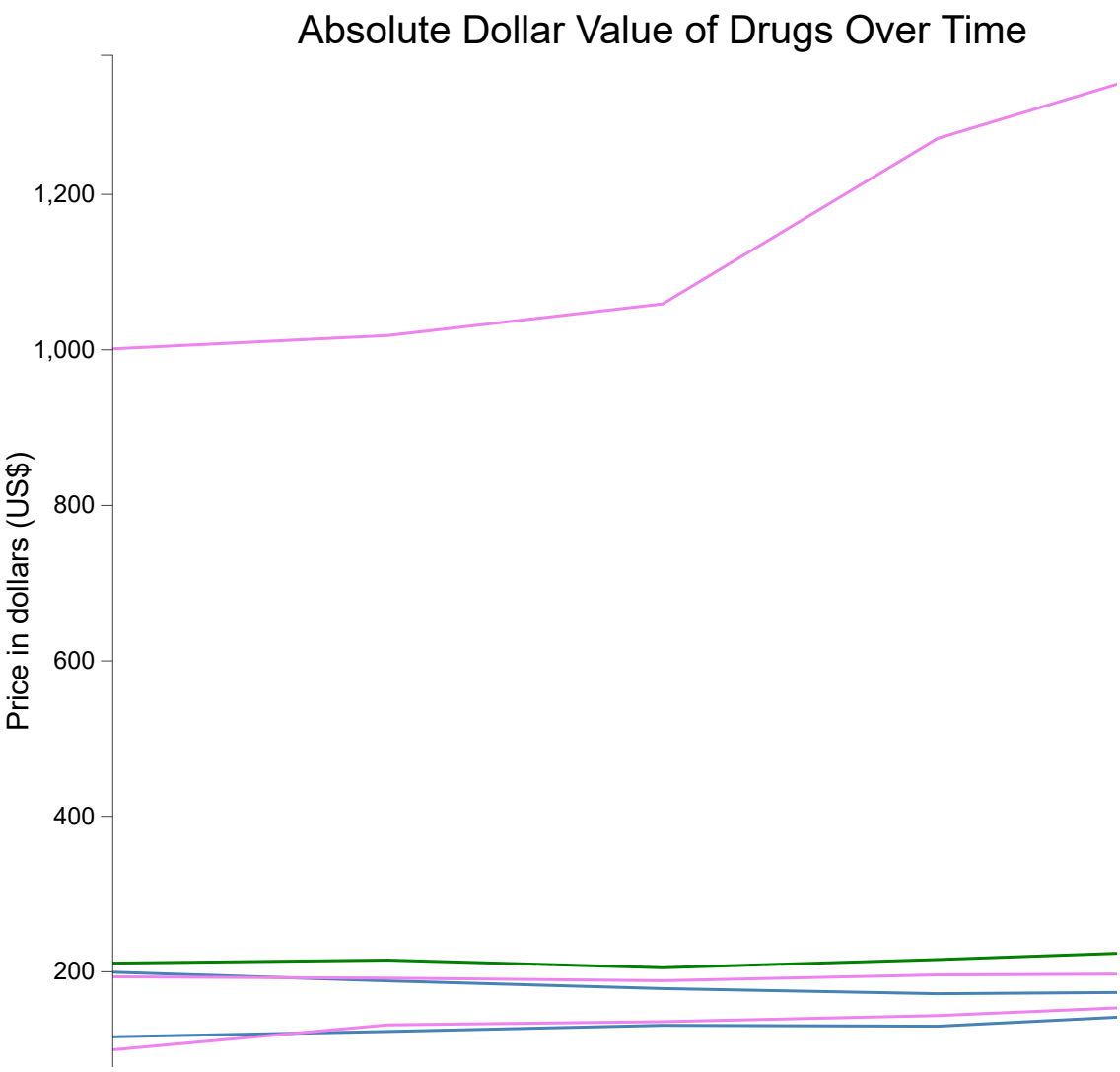
Even after trying using .loc[row_indexer, col_indexer] I was having problems, so I decided to transer the data into an array.

Because some drugs are exhorbitantly more expensive than others, I decided to use the mpld3 package. This allows for two things:

- If you hover over a line, it will show you the name of the line to which it belongs
- On the bottom left of the screen there are three icons, which allow you to zoom into an area of the graph, and generally move and explore the graph

In [9]:
```
price_array = price_data.as_matrix()   #converted the dataframe to an array fo
r ease of use
new_price_array=np.delete(price_array,0,-1)  #Removing the first column, which
 is not needed

#Setting the parameters for the axis and the plot
year = [2011, 2012, 2013, 2014, 2015]
my_xticks = ['2011', '2012', '2013', '2014', '2015']
f, ax = plt.subplots(figsize=(10, 10))
plt.xticks(year, my_xticks)
plt.title('Absolute Dollar Value of Drugs Over Time',fontsize = 20)
ax.set_ylabel('Price in dollars (US$)',fontsize = 15)  #setting y label
ax.set_xlabel('Year',fontsize = 15)   #setting x label
plt.margins(x=0) #Making sure there is no gap between the plotted lines and th
e x axis
plt.margins(y=0) #Making sure there is no gap between the plotted lines and th
e y axis

# Plotting the drugs, I decided to go in increments of 10, instead of 30 as or
iginally outlined in the Data Report
# because this makes it much easier to notice trends across the groups. The co
de below iterates through
# the array and plots the lines by groups of 10, the first 10 (orange) are the
 most used,
#the bottom 10 are the least used (violet)

for i in range (0,10): #plotting first 10 as orange
    line = ax.plot(year, new_price_array[i],color='darkorange')
    #The code below connects the plotted line to its corresponding name in the
 array
    mpld3.plugins.connect(f, mpld3.plugins.LineLabelTooltip(line[0], label=pri
ce_array[i][0]))
for i in range (10,20): #plotting the next 10 as red
    line = ax.plot(year, new_price_array[i],color='tomato')
    mpld3.plugins.connect(f, mpld3.plugins.LineLabelTooltip(line[0], label=pri
ce_array[i][0]))
for i in range (20,30): #plotting the next 10 as green
    line = ax.plot(year, new_price_array[i],color='green')
    mpld3.plugins.connect(f, mpld3.plugins.LineLabelTooltip(line[0], label=pri
ce_array[i][0]))
for i in range (30,40): #plotting the next 10 as blue
    line = ax.plot(year, new_price_array[i],color='steelblue')
    mpld3.plugins.connect(f, mpld3.plugins.LineLabelTooltip(line[0], label=pri
ce_array[i][0]))
for i in range (40,50): #plotting the next 10 as violet
    line = ax.plot(year, new_price_array[i],color='saddlebrown')
    mpld3.plugins.connect(f, mpld3.plugins.LineLabelTooltip(line[0], label=pri
ce_array[i][0]))
for i in range (50,60):#plotting the next 10 as violet
    line = ax.plot(year, new_price_array[i],color='violet')
    mpld3.plugins.connect(f, mpld3.plugins.LineLabelTooltip(line[0], label=pri
ce_array[i][0]))

mpld3.display()
```

Out[9]:



Notes on using the graph: This section explains how to use the interactive graph. There are three buttons [home, directional arows, zoom] (left to right) on the bottom left corner.

- **Home:** This will reset the graph to the original position and zoom
- **Arrows:** Once zoomed in, use this tool to move around the graph. Use it by selecting the tool, then dragging or 'pulling' across the graph to move
- **Zoom:** This will allow you to zoom into areas of the graph. To use it, select the tool, click and drag to create an area to zoom into. The smaller/more specific the area, the larger the zoom. You can zoom multiple times. You can also use your trackpad's multitouch feature to zoom in, currently unsure if you can use the mouse wheel, but I think so. What a great package!
- Sometimes the graph gets a little buggy with displaying names, usually you can fix this by exiting the graph with your cursor and entering touching just the line you want

Observations:

After interacting a little and exploring the graph, there are some interesting observations.

- The least used drugs are generally the more expensive (per unit). This is noticeable because the violet lines tend to be towards the top of the graph. On average, the U.S. government pays more money per unit for these drugs.
- Similarly, the most used drugs tend to be the least costly (per unit). This is noticeable because out of the bottom 8 drugs, 6 of them are painted orange!

Plotting relative increases/decreases

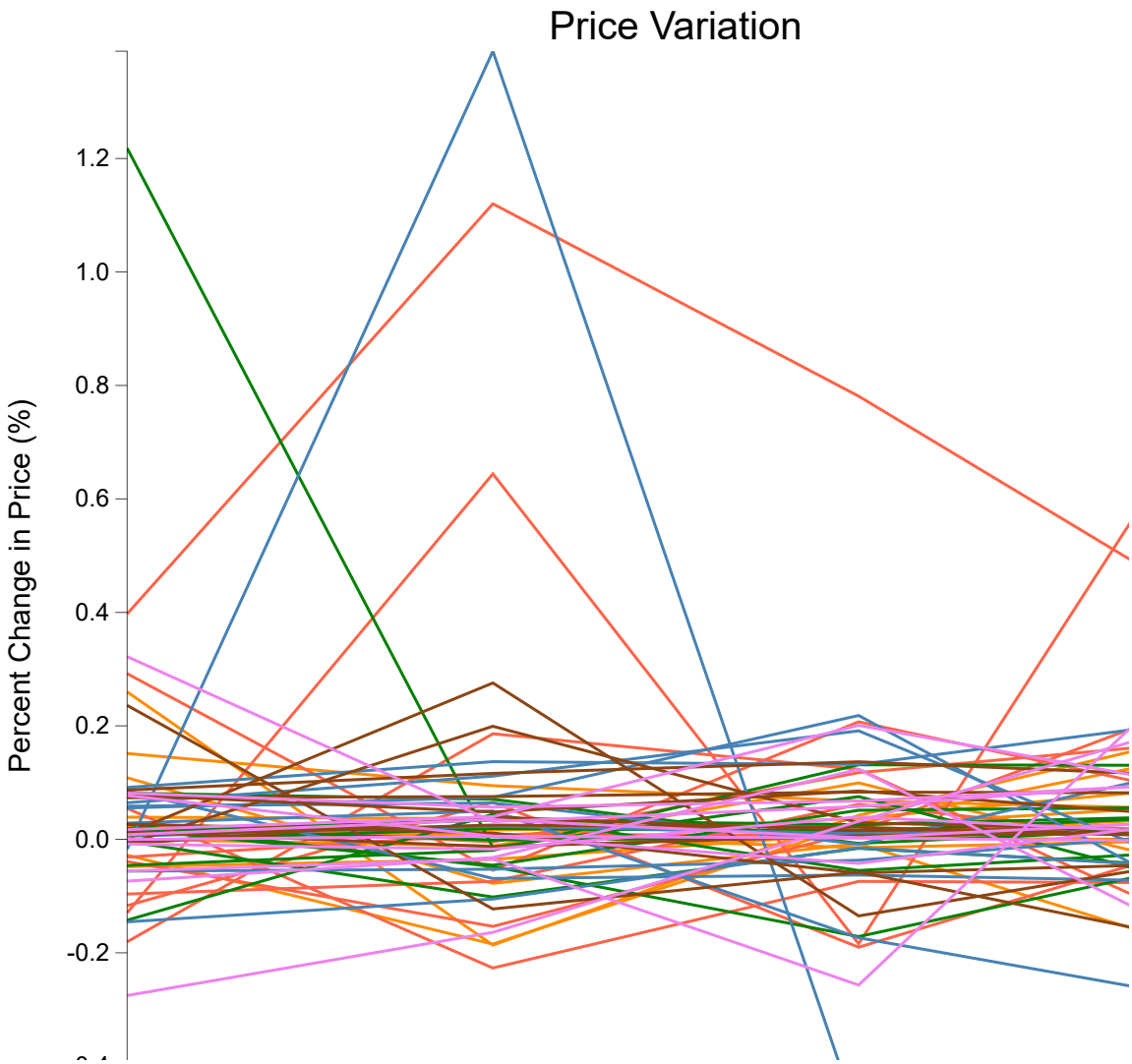
The absolute values are interesting, and they give us important information on a specific drug's performance over time. But my research question attempts to look at the effect of price increases in drugs on the U.S. budget, therefore it is most important to use a percentage increase/decrease in a drugs price.

This will help by normalizing the data between the most expensive and cheapest drugs, and paint a clearer picture of which drugs are subject to the most volatility.

In [10]:
```
#To plot this, I will create a new array with the data. Because it uses
 the percentage differences, it will be smaller than the
#previous by one column
percentchange = np.zeros(shape=(60,4))


for i in range(0,len(new_price_array)):
    percentchange[i][0]=(new_price_array[i][1]-new_price_array[i][0])/ne
w_price_array[i][0]
    percentchange[i][1]=(new_price_array[i][2]-new_price_array[i][1])/ne
w_price_array[i][1]
    percentchange[i][2]=(new_price_array[i][3]-new_price_array[i][2])/ne
w_price_array[i][2]
    percentchange[i][3]=(new_price_array[i][4]-new_price_array[i][3])/ne
w_price_array[i][3]

#Setting the parameters for the axis and the plot
pctchange_year = [2011,2012, 2013, 2014]
pctchange_xticks = ['2011', '2012', '2013', '2014']

pct_f, pct_ax = plt.subplots(figsize=(10, 10))
plt.xticks(pctchange_year, pctchange_xticks)
plt.title('Price Variation',fontsize = 20)
pct_ax.set_ylabel('Percent Change in Price (%)', fontsize = 15)  #settin
g y label
pct_ax.set_xlabel('Year', fontsize = 15)   #setting x label
plt.margins(x=0) #Making sure there is no gap between the plotted lines
 and the x axis
plt.margins(y=0) #Making sure there is no gap between the plotted lines
 and the y axis

#Plotting the drugs, again by increments, the color scheme remains the s
ame
for i in range (0,10):
    pct_line = pct_ax.plot(pctchange_year, percentchange[i],color='darko
range')
    mpld3.plugins.connect(pct_f, mpld3.plugins.LineLabelTooltip(pct_line
[0], label=price_array[i][0]))
for i in range (10,20):
    pct_line = pct_ax.plot(pctchange_year, percentchange[i],color='tomat
o')
    mpld3.plugins.connect(pct_f, mpld3.plugins.LineLabelTooltip(pct_line
[0], label=price_array[i][0]))
for i in range (20,30):
    pct_line = pct_ax.plot(pctchange_year, percentchange[i],color='gree
n')
    mpld3.plugins.connect(pct_f, mpld3.plugins.LineLabelTooltip(pct_line
[0], label=price_array[i][0]))
for i in range (30,40):
    pct_line = pct_ax.plot(pctchange_year, percentchange[i],color='steel
blue')
    mpld3.plugins.connect(pct_f, mpld3.plugins.LineLabelTooltip(pct_line
[0], label=price_array[i][0]))
for i in range (40,50):
    pct_line = pct_ax.plot(pctchange_year, percentchange[i],color='saddl
ebrown')
```

```
    mpld3.plugins.connect(pct_f, mpld3.plugins.LineLabelTooltip(pct_line
[0], label=price_array[i][0]))
for i in range (50,60):
    pct_line = pct_ax.plot(pctchange_year, percentchange[i],color='viole
t')
    mpld3.plugins.connect(pct_f, mpld3.plugins.LineLabelTooltip(pct_line
[0], label=price_array[i][0]))

mpld3.display()
```

Out[10]:



Note on interpreting the graph: This graph shows the amount that the price changed from year to the next. For example, the green line for the drug: "Injection, methylprednisolone acetate, 20 mg" has a 1.2 value for the year 2011. This means that for the year 2011-2012 the drug increased by 1.2% in average price per unit

Observations:

Now we can get to interpreting observations:

- Immediately we notice that some of the bigger fluctuations come from two orange (most used) drugs. This means that the U.S. needs to be weary of price increases for drugs that represent a huge volume. It can easily affect the medicare spending budget and throw it off balance
- The drugs with the most fluctuation are all injections. Perhaps they saw themselves affected by a shortage? Maybe even the anti-vaccine movement
- Most drugs, however trend around the following range: -0.2 < percent change < 0.2. There doesn't seem to be a particular correlation between how many beneficiaries these drugs have and their changes in price over time. Which is good news for the Federal budget!

Further analysis

Some of the lines with very large fluctuations have peaked my interest. Why were the fluctuations so much? Did it really come from increases in drug prices, or, since the data is Average spending per unit, from the number of beneficiaries? Fortunately, I can easily look into this data using Pandas! With a quick hover over the green, blue, and orange lines with huge fluctuations, I noted down that the drugs I am interested in are the following:

- Injection, methylprednisolone acetate, 20 mg
- Injection, morphine sulfate, up to 10 mg
- Injection, vitamin b-12 cyanocobalamin, up to 1000 mcg
- Injection, ketorolac tromethamine, per 15 mg

Good thing we saved the drug_data DataFrame all the way in the beginning, those extra columns will come in handy!

In [11]:
```
#First, we want to set the index to the Name of the drug to make it easi
er to search for it.
drug_data.reset_index(drop=True, inplace = True)

beneficiary_count = drug_data[['HCPCS Description','Beneficiary Count, 2
011','Beneficiary Count, 2012',
                        'Beneficiary Count, 2013','Beneficiary Count, 20
14',
                        'Beneficiary Count, 2015']]
beneficiary_count.columns = ['Name', '2011','2012','2013','2014','2015']
 #Renaming the columns for ease of use

#Here I create an array with the names of the drugs I am interested in,
 as well as some randomly selected drugs for reference
names_of_drugs = ['Injection, methylprednisolone acetate, 20 mg','Inject
ion, morphine sulfate, up to 10 mg'
                 ,'Injection, vitamin b-12 cyanocobalamin, up', 'Injecti
on, ketorolac tromethamine, per 15 mg'
                  ,'Injection, dexamethasone, intravitreal implant, 0.1
 mg', 'Injection, levoleucovorin calcium, 0.5 mg'
                  ,'Methotrexate sodium, 5 mg']
```

Now I create an array with the indexes of the drugs to do the final analysis. To do this I iterate through the names_of_drugs array, each time getting the appropriate index within the DataFrame and appending it to a new dataframe. I am sure there must be a method that returns just the index value, but I could not find it. Right now the index_of_drugs, alas, I could not find it, so I did some string slicing to each element as it came into the array

In [12]:
```
index_of_drugs = []
for val in names_of_drugs:
    index_of_drugs.append(str(beneficiary_count[beneficiary_count['Name'].str.
contains(val)].index)[12:14])
index_of_drugs = list(map(int, index_of_drugs)) #converting back into an int a
rray

#Now I create a new Dataframe, with the same columns as the previous
selected_drugs = pd.DataFrame(columns= beneficiary_count.columns.tolist())
#And now I populate the DataFrame
i = 0
for val in index_of_drugs:
    selected_drugs.loc[i] = beneficiary_count.loc[val].tolist()
    i = i+1

#We end up with a DataFrame that looks like this:
selected_drugs
```

Out[12]:

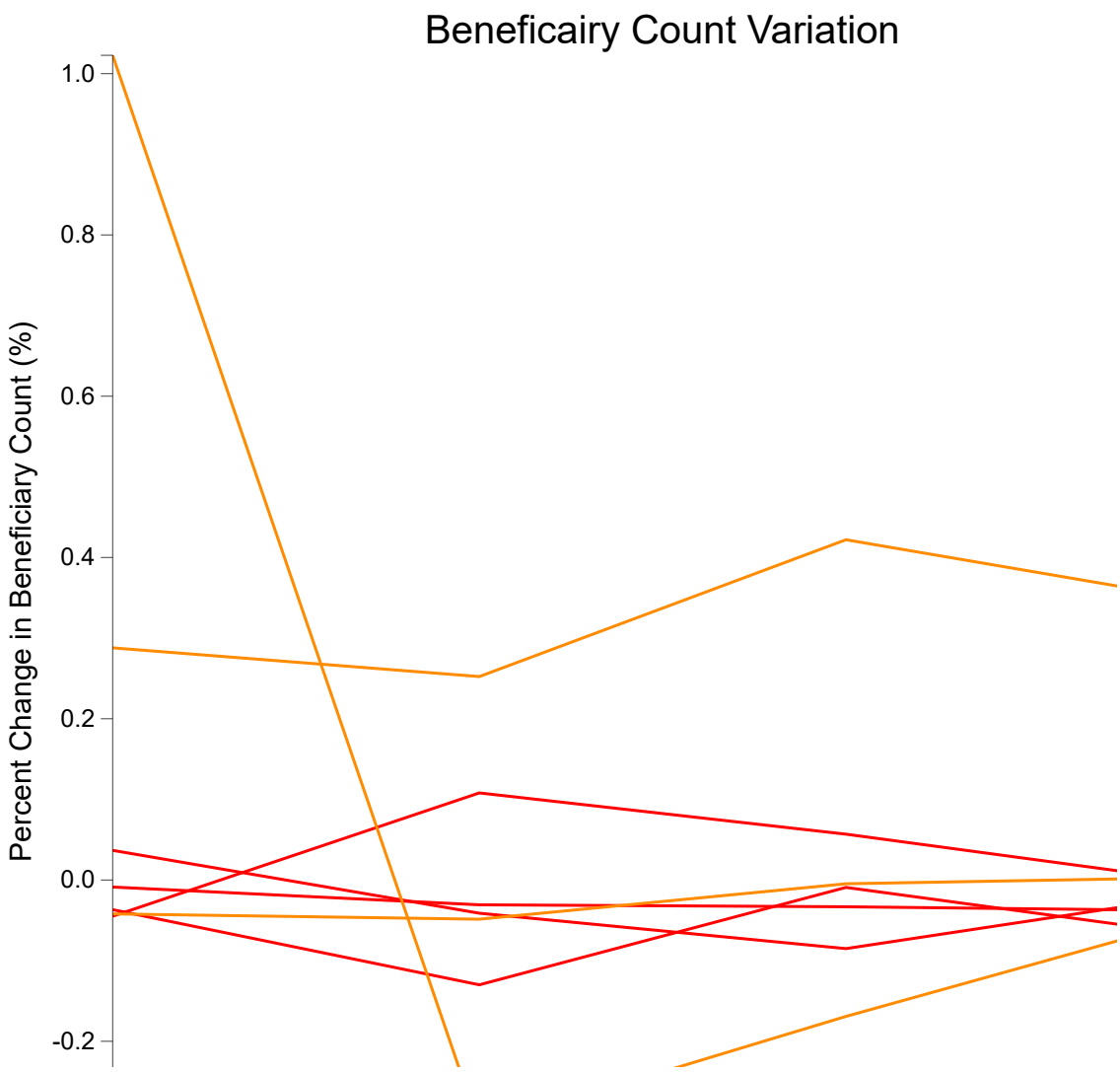
	Name	2011	2012	2013	2014	2015
0	Injection, methylprednisolone acetate, 20 mg	171707.0	170210.0	164977.0	159503.0	153424.0
1	Injection, morphine sulfate, up to 10 mg	10529.0	10142.0	8824.0	8743.0	8124.0
2	Injection, vitamin b-12 cyanocobalamin, up to...	641776.0	665385.0	637994.0	583712.0	574137.0
3	Injection, ketorolac tromethamine, per 15 mg	389944.0	372422.0	412644.0	436105.0	434174.0
4	Injection, dexamethasone, intravitreal implant...	3970.0	5113.0	6403.0	9105.0	12241.0
5	Injection, levoleucovorin calcium, 0.5 mg	5622.0	11372.0	7886.0	6552.0	6273.0
6	Methotrexate sodium, 5 mg	6832.0	6545.0	6227.0	6198.0	6219.0

Plotting the Data: Beneficiary Count changes

In [13]:
```
selected_array = selected_drugs.as_matrix()   #converted the dataframe to an a
rray for ease of use
new_selected=np.delete(selected_array,0,-1)  #Removing the first column, which
 is not needed

selected_percentchange = np.zeros(shape=(7,4))

#Calculating the percentage change for this array
for i in range(0,len(new_selected)):
    selected_percentchange[i][0]=(new_selected[i][1]-new_selected[i][0])/new_s
elected[i][0]
    selected_percentchange[i][1]=(new_selected[i][2]-new_selected[i][1])/new_s
elected[i][1]
    selected_percentchange[i][2]=(new_selected[i][3]-new_selected[i][2])/new_s
elected[i][2]
    selected_percentchange[i][3]=(new_selected[i][4]-new_selected[i][3])/new_s
elected[i][3]
```

In [14]:
```
#Setting the parameters for the axis and the plot
selected_year = [2011,2012, 2013, 2014]
selected_xticks = ['2011', '2012', '2013', '2014']

slct_f, slct_ax = plt.subplots(figsize=(10, 10))
plt.xticks(selected_year, selected_xticks)
plt.title('Beneficairy Count Variation',fontsize = 20)
slct_ax.set_ylabel('Percent Change in Beneficiary Count (%)', fontsize = 15)
#setting y label
slct_ax.set_xlabel('Year', fontsize = 15)   #setting x label
plt.margins(x=0) #Making sure there is no gap between the plotted lines and th
e x axis
plt.margins(y=0) #Making sure there is no gap between the plotted lines and th
e y axis

#This time I took the lines of the four drugs I originally outlined as having
 a lot of variation in price and plotted them as red
for i in range (0,4):
    slct_line = slct_ax.plot(selected_year, selected_percentchange[i],color='r
ed')
    mpld3.plugins.connect(slct_f, mpld3.plugins.LineLabelTooltip(slct_line[0],
 label=selected_array[i][0]))
#The randomly selected drugs, used for reference, are plotted as orange
for i in range (4,7):
    slct_line = slct_ax.plot(selected_year, selected_percentchange[i],color='d
arkorange')
    mpld3.plugins.connect(slct_f, mpld3.plugins.LineLabelTooltip(slct_line[0],
 label=selected_array[i][0]))


mpld3.display()
```

Out[14]:



Observations:

It seems like the large variations in price were not due to large increases in beneficiary count, in fact some of the drugs with less variation in price, experiences heavier fluctuations in population count ("Injections, levoleucovorin calcium, 0.5 mg" for example, doubled in number of beneficiaries from 2011 to 2012)

Further Analysis: Final Data

Fortunately, it is now pretty easy to plot several charts using the same code as before! All that I need is to change the array specifying what columns to keep, and the rest is just changing variable names for ease of use. With that in mind, I decided that maybe beneficiary count wasn't exactly what I was looking for, since Average Unit Price uses Unit Count per year instead of Beneficiary count to be calculated. So, below, I plot a graph for the fluctuations in Unit Count.

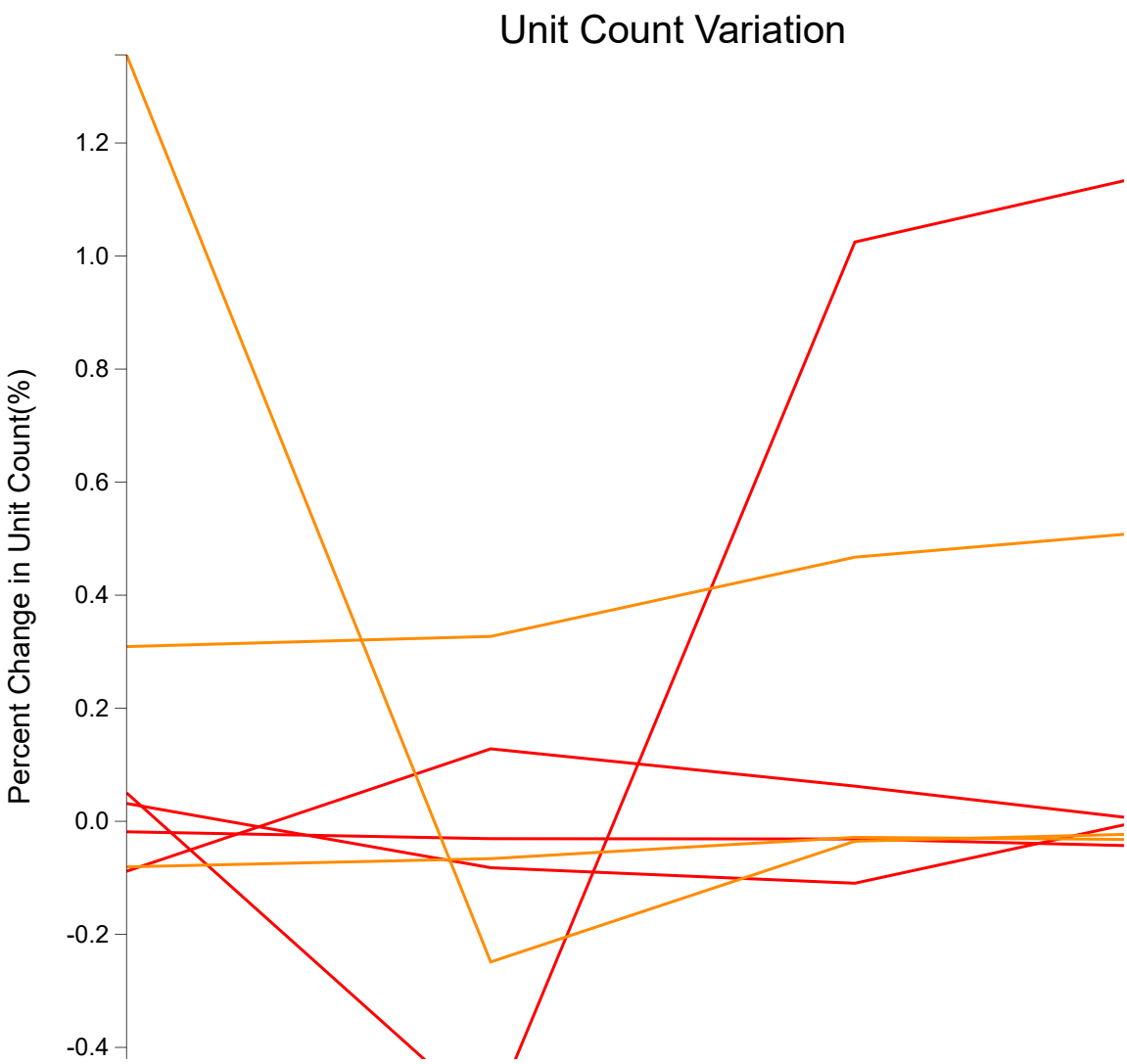
In [15]:
```
#First, we want to set the index to the Name of the drug to make it easier to
 search for it.
drug_data.reset_index(drop=True, inplace = True)

unit_count = drug_data[['HCPCS Description','Unit Count, 2011','Unit Count, 20
12',
                        'Unit Count, 2013','Unit Count, 2014',
                        'Unit Count, 2015']]
unit_count.columns = ['Name', '2011','2012','2013','2014','2015'] #Renaming th
e columns for ease of use

index_of_drugs = []
for val in names_of_drugs:
    index_of_drugs.append(str(unit_count[unit_count['Name'].str.contains(val)]
.index)[12:14])
index_of_drugs = list(map(int, index_of_drugs)) #converting back into an int a
rray

#Now I create a new Dataframe, with the same columns as the previous
selected_drugs = pd.DataFrame(columns= unit_count.columns.tolist())
#And now I populate the DataFrame
i = 0
for val in index_of_drugs:
    selected_drugs.loc[i] = unit_count.loc[val].tolist()
    i = i+1

#We end up with a DataFrame that looks like this:
selected_drugs
```

Out[15]:

	Name	2011	2012	2013	2014	2015
0	Injection, methylprednisolone acetate, 20 mg	424241.4	416365.0	403560.3	390929.2	372570.1
1	Injection, morphine sulfate, up to 10 mg	107631.0	113048.7	53552.9	108432.1	235448.0
2	Injection, vitamin b-12 cyanocobalamin, up to...	2771896.3	2859408.5	2624646.6	2337047.6	2407486.6
3	Injection, ketorolac tromethamine, per 15 mg	1949867.9	1777479.0	2005344.7	2130084.7	2104282.2
4	Injection, dexamethasone, intravitreal implant...	42489.0	55621.0	73814.0	108305.0	164843.0
5	Injection, levoleucovorin calcium, 0.5 mg	19199270.7	45230493.2	33975431.2	32779286.0	32159375.0
6	Methotrexate sodium, 5 mg	327072.1	300695.4	280817.6	272844.5	263759.4

In [16]:
```
selected_array = selected_drugs.as_matrix()   #converted the dataframe t
o an array for ease of use
new_selected=np.delete(selected_array,0,-1)  #Removing the first column,
 which is not needed

selected_percentchange = np.zeros(shape=(7,4))

#Calculating the percentage change for this array
for i in range(0,len(new_selected)):
    selected_percentchange[i][0]=(new_selected[i][1]-new_selected[i][0])
/new_selected[i][0]
    selected_percentchange[i][1]=(new_selected[i][2]-new_selected[i][1])
/new_selected[i][1]
    selected_percentchange[i][2]=(new_selected[i][3]-new_selected[i][2])
/new_selected[i][2]
    selected_percentchange[i][3]=(new_selected[i][4]-new_selected[i][3])
/new_selected[i][3]

#Setting the parameters for the axis and the plot
selected_year = [2011,2012, 2013, 2014]
selected_xticks = ['2011', '2012', '2013', '2014']

slct_f, slct_ax = plt.subplots(figsize=(10, 10))
plt.xticks(selected_year, selected_xticks)
plt.title('Unit Count Variation',fontsize = 20)
slct_ax.set_ylabel('Percent Change in Unit Count(%)', fontsize = 15)  #s
etting y label
slct_ax.set_xlabel('Year', fontsize = 15)   #setting x label
plt.margins(x=0) #Making sure there is no gap between the plotted lines
 and the x axis
plt.margins(y=0) #Making sure there is no gap between the plotted lines
 and the y axis

#This time I took the lines of the four drugs I originally outlined as h
aving a lot of variation in price and plotted them as red
for i in range (0,4):
    slct_line = slct_ax.plot(selected_year, selected_percentchange[i],co
lor='red')
    mpld3.plugins.connect(slct_f, mpld3.plugins.LineLabelTooltip(slct_li
ne[0], label=selected_array[i][0]))
#The randomly selected drugs, used for reference, are plotted as orange
for i in range (4,7):
    slct_line = slct_ax.plot(selected_year, selected_percentchange[i],co
lor='darkorange')
    mpld3.plugins.connect(slct_f, mpld3.plugins.LineLabelTooltip(slct_li
ne[0], label=selected_array[i][0]))


mpld3.display()
```

Out[16]:



Observations:

Now it is still generally looking about the same. However, one drug's fluctuations can be attributed, in part, to large change in unit count. "Injections, morphine sulfate, up to 10 mg" changed quite a bit in the amount of units purchased.

Interestingly, "Injection, levoleucovorin calcium, 0.5 mg" also fluctuated a lot but, as it is colored orange, the drug price actually didn't change all that much.

For the remaining of the drugs, with little fluctuation in Unit Count, it must be the price that is changing.

Another interesting observation is that some drugs like "Injection, levoleucovorin calcium, 0.5 mg" have almost 1:1 mappings of the line from Beneficiary Count to Unit count, while some drugs like "Injections, morphine sulfate, up to 10 mg" fluctuated heavily without the corresponding increases in Beneficiary Count

Summary of findings

This proved to be a very interesting exercise, and it got me thinking about how the U.S. plans its federal budget. With such a large percentage of spending going towards Medicare and other health related services, the government is heavily afflicted by changes in the prices or number of beneficiaries of certain drugs.

Among the more interesting ideas is that the fewer beneficiaries of a drug, the more expensive it is in absolute terms. Yet this doesn't mean these will be the most volatile drugs, in fact, while there was no significant correlation, the most volatile drugs were some of the most used drugs

Furthermore, a deeper analysis into why the average price per unit was so volatile, revealed that beneficiary increases/decreases were not the root cause, sometimes, the government bought more or less units regardless of beneficiary increases (and sometimes almost perfectly correlated with beneficiary increases) it must be the actual price of these medications that is putting a strain on the federal budget.