40156103_3062_report

November 25, 2018

1 Analysis of Cost of Antiepileptics (AEDs) to the NHS in NI

This report is targeted at the NHS in Northern ireland, specifically highlighting the cost of Prescriptions of AEDs - which are the category of Central Nervous system drugs that cost the most to the NHS each year.

The NHS, as we know, is in a budget crisis - not having enough money or resources to satisfy the vast demand it needs to deliver to the British people, because of this the NHS values immensely any cost analysis that can give real and attainable solutions to cutting costs. Like cutting costs in supermarket shopping, I found an easy way to cut costs is to switch from a name brand prescription drug to a generic brand of presription drug. The potential cost savings from this were surpprising and in some cases staggering. Also in my report I will pinpoint which particular Practices in NI are the largest transgressors of supplying the more expensive name brand prescription. By doing this I can identify real world solutions to how this problem can be addressed. For my analysis I am analysing the combined records of GP Prescriptions in NI from June 2015 - June 2018

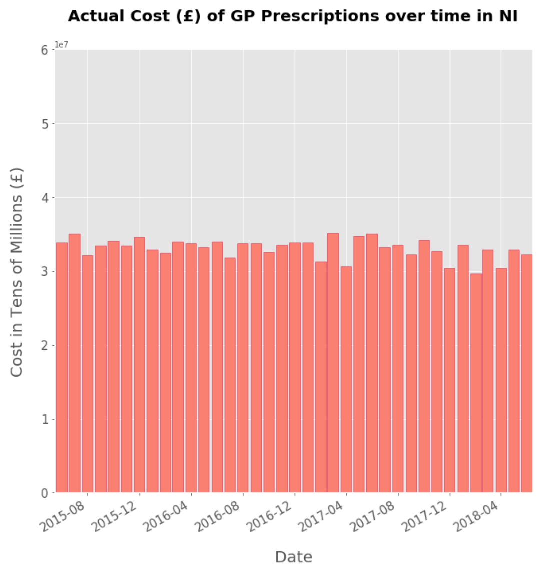
1.1 Ethical & legal concerns

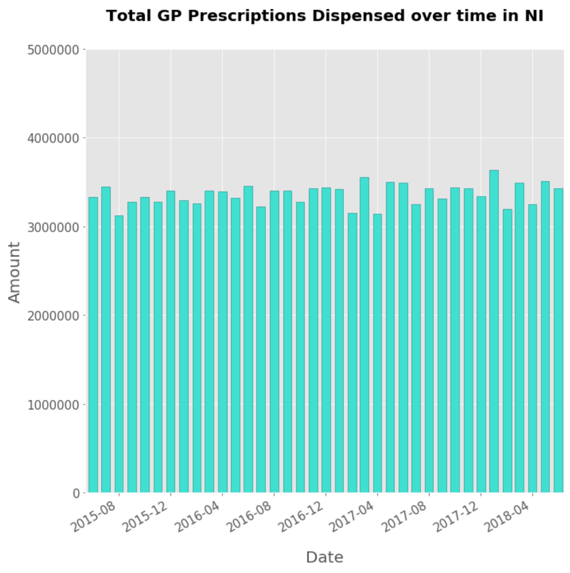
My report has no reason to be described as having ethical concerns as it only shows an overview of how to save money on prescription ourchases. It does not recommend that these drugs are not purchased, because while some of them are extortionatly priced, I cannot pretend to know what prescription medicines are best for patients. Especially for patients who have extreme epilepsy who would depend on some of these specific medicines more than most. I made sure to research diligently what the effects were of switching from branded to generic - and if there were any risks associated with my recommendations. Inded in the case of Clonazepam, I made sure to note that some patients may require the need of the liquid form of the medicine and this could explain why GPs may choose to dispense the more expensive branded version. Clearly no discrimination of sex, race or religion was present in my report either Legally, whilst I idnetify individual practices and recommend action against them in some cases; I believe this is fair practice - as these medical practices are on public record and they spend public money to fulfill these prescriptions and so are accountble to the public. I made sure to not unfairly label them as outliers in thei branded spend by including their spend on generic products in the equation

```
In [14]: import glob, os
         import pandas as pd

         #Function that converts all csv files in a directory into one dataframe. Allows you t
         #specify if you want any columns enamed before concatenation so that data is more com
         #Also has the option to drop unwanted columns. All parameters apart from the path are
         # just in case the user doesn't require the particular functinality those arguements
         def concatenate_files(path, colchangefrom = None, colchangeto = None, dropcols = None
             all_files = glob.glob(os.path.join(path, "*.csv"))
             appended_df = []
             for file in all_files:
                 df = pd.read_csv(file, encoding = 'ISO-8859-1', low_memory = False)
                 if colchangefrom in df.columns:
                     df.rename(columns ={colchangefrom: colchangeto}, inplace = True)
                 appended_df.append(df)

             concatenated_df   = pd.concat(appended_df, ignore_index=True, sort = True)

             if dropcols:
                 concatenated_df.drop(dropcols, axis=1,inplace = True)
                 concatenated_df.reset_index(drop = True, inplace = True)
                 concatenated_df = concatenated_df.dropna(how = 'all')
                 concatenated_df.reset_index(drop = True, inplace = True)
         #I can do this blunt drop of null rows safely without fear of data loss as I demonstr
         #only if I drop uneccessary columns though. Otherwise it's best to avoid broad drops
             return concatenated_df

In [15]: import numpy as np
         import datetime
         GPcost = concatenate_files(r'C:\Users\fionn\OneDrive\Documents\3062dataanalysis\Hospi
         GPcost = GPcost[['Year', 'Month', 'Actual Cost (č)', 'Total Items']]
         GPcost = GPcost.dropna()
         GPcost.reset_index(drop = True, inplace = True)
         cols = ["Year", "Month"]
         GPcost[cols] = GPcost[cols].applymap(np.int64)

         GPcost["Date"] =  GPcost["Month"].astype(str) + GPcost["Year"].astype(str)
         GPcost['Date'] = pd.to_datetime(GPcost['Date'], format ='%m%Y')

In [16]: def mySum(results):
             total = 0
             for r in results:
                 total += int(r*100)

             return total/100

         def groupbylist(df, accum, group):
             allgroup = list(df[group].unique())
             allgroup = sorted(allgroup)
             returnlist = []
             for group1 in allgroup:
                 df1 = df[df[group] == group1]
                 listitem = list(df1[accum])
```

```
                returnlist.append(mySum(listitem))
            return returnlist
```

```
In [27]: import matplotlib.pyplot as plt

         allDates = list(GPcost["Date"].unique())
         allDates = sorted(allDates)
         costPerMonth = groupbylist(GPcost, "Actual Cost (č)", "Date")
         itemsPerMonth = groupbylist(GPcost, "Total Items", "Date")

         fig, (ax, ax2) = plt.subplots(1, 2)
         fig.set_size_inches(20,10)
         #plt.subplot(1, 2, 1)
         fig.autofmt_xdate()
         ax.set_xlabel('Date', fontsize = 20, labelpad = 20.0)
         ax.set_ylabel('Cost in Tens of Millions (č)', fontsize = 20, labelpad = 20.0)
         ax.set_title('Actual Cost (č) of GP Prescriptions over time in NI', fontsize = 20, fo
         ax.grid(True)
         ax.bar(allDates, costPerMonth, width = 25, color = 'salmon', edgecolor = 'crimson')
         ax.xaxis.set_tick_params(labelsize = 15)
         ax.yaxis.set_tick_params(labelsize = 15)
         ax.set_ylim([0,60000000])
         ax.set_xlim([datetime.date(2015, 5, 15),datetime.date(2018, 6, 15)])


         ax2.set_xlabel('Date', fontsize = 20, labelpad = 20.0)
         ax2.set_ylabel('Amount', fontsize = 20, labelpad = 5.0)
         ax2.set_title('Total GP Prescriptions Dispensed over time in NI', fontsize = 20, font
         ax2.grid(True)
         ax2.bar(allDates, itemsPerMonth, width = 20, color = 'turquoise', edgecolor = 'teal')
         ax2.set_ylim([0,5000000])
         ax2.set_xlim([datetime.date(2015, 5, 15),datetime.date(2018, 6, 15)])
         ax2.xaxis.set_tick_params(labelsize = 15)
         ax2.yaxis.set_tick_params(labelsize = 15)
         plt.style.use('seaborn-muted')
         plt.tight_layout()
         plt.show()
```





At an overall broad glance on NHS spending and prescription allocation in Northern ireland in the past three years, we can we that the trends have remained largely the same - despite many promises from politicians to streamline costs and save money. Obviously we can't prescribe less medicine, as patients undoubtedly require their prescriptions, but rather we can find cheaper ways to prescribe this medicine. Through my analysis later I have found that much can be done to reduce costs in the way GP Prescriptions are dispensed and that these solutions are rather trivial and would be simple to implement. But first we must determine where exactly these costs largely come from. I thought it would be useful as an intro in my customer report to note that largely little has been done to curb the costs of GP Prescriptions in NI as the overall cost each month has largely stayed the same. Having the total amount of prescriptions alongside in my opinion was also useful as it shows that any momentary dips in cost have been directly down to less prescriptions in that month than any cost efficient strategy by the NHS

1.2 Breaking down GP Prescription costs by type of medicine

In this section we will take a look at the categories of drugs which are the biggest drain on NHS resources

```
In [22]: BNFdf = concatenate_files(r'C:\Users\fionn\OneDrive\Documents\3062dataanalysis\Hospita
                                    'Practice', 'PRACTICE')
         BNFdf = BNFdf[BNFdf["BNF Chapter"] != '-']
         BNFdf["BNF Chapter"] = pd.to_numeric(BNFdf["BNF Chapter"])
         BNFdf = BNFdf[['BNF Chapter', 'Actual Cost (č)', 'Total Items']]
         BNFdf = BNFdf.dropna()
         BNFdf.reset_index(drop = True, inplace = True)
```

```
In [23]: #Using this function I will change the list values in BNFChapters list to their corre
         # Will make the data far more understandable and easy to interpret
         def replaceChaptersWithDescription(chapterlist):
             for index, item in enumerate(chapterlist, start = 0):
```

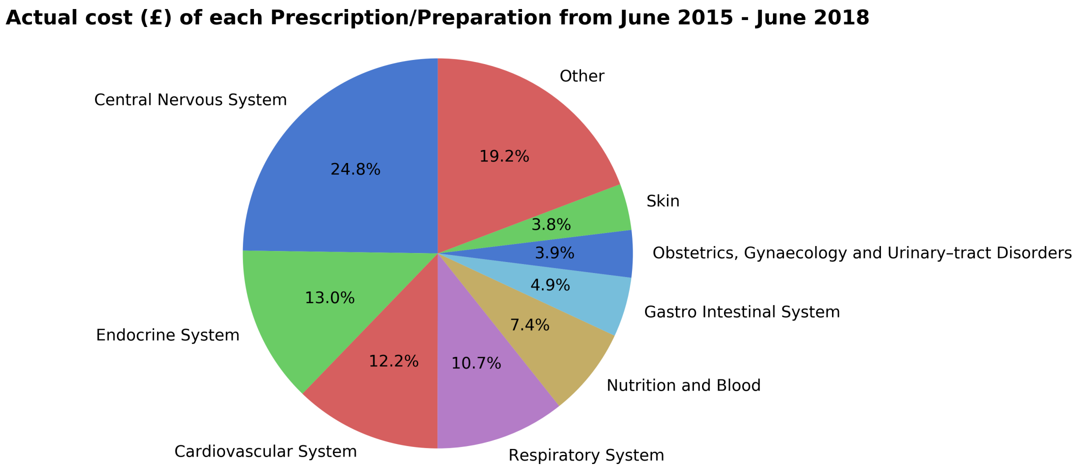
```
item = int(item)
if item == 1:
    chapterlist[index] = "Gastro Intestinal System"

elif item == 2:
    chapterlist[index] = "Cardiovascular System"

elif item == 3:
    chapterlist[index] = "Respiratory System"

elif item == 4:
    chapterlist[index] = "Central Nervous System"

elif item == 5:
    chapterlist[index] = "Infections"

elif item == 6:
    chapterlist[index] = "Endocrine System"

elif item == 7:
    chapterlist[index] = "Obstetrics, Gynaecology and Urinarytract Disorders"

elif item == 8:
    chapterlist[index] = "Malignant disease and immunosuppression"

elif item == 9:
    chapterlist[index] = "Nutrition and Blood"

elif item == 10:
    chapterlist[index] = "Musculoskeletal and Joint Diseases"

elif item == 11:
     chapterlist[index] = "Eyes"

elif item == 13:
     chapterlist[index] = "Skin"

elif item == 14:
    chapterlist[index] = "Immunological products and vaccines"

elif item == 15:
    chapterlist[index] = "Anaesthesia"

elif item == 18:
     chapterlist[index] = "Preparations used in diagnosis"

elif item == 19:
     chapterlist[index] = "Other drugs and preparations"
```

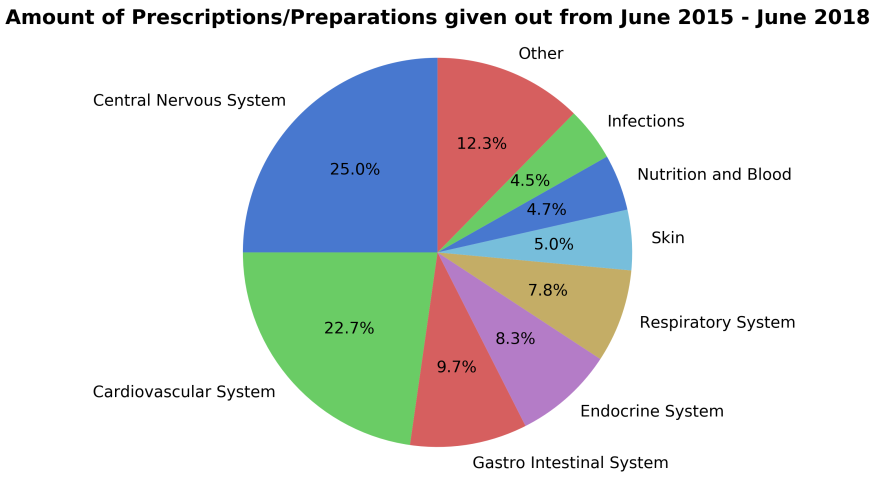
```
            elif item == 20:
                 chapterlist[index] = "Dressings"

            elif item == 21:
                chapterlist[index] = "Appliances"

            elif item == 22:
                chapterlist[index] = "Incontinence appliances"

            elif item == 23:
                 chapterlist[index] = "Stoma appliances"

            elif item == 99:
                 chapterlist[index] = "Unclassified"

            else:
                 chapterlist[index]  = "Other"

        return chapterlist

    #Handy function returning two lists from a previously made dict; one for labels on a
    #Creates a top number of values to plot and puts the rest in "Other" by default, this
    def createValuesandLabels(dictionary, numvals, otherlabel = "Other"):
        labels = []
        values = []

        for key in sorted(dictionary, key=dictionary.get, reverse=True):
            labels.append(key)
            values.append(dictionary.get(key))

        labelslist = labels[0:numvals]
        valueslist = values[0:numvals]


    #This code will make sure that no database entries are left behind, the remaining
    #BNF chapters outside the top 11 will be included under the umbrella label "Other"
        other = 0
        for i in range(numvals, len(values)):
            other += values[i]
        labelslist.append(otherlabel)
        valueslist.append(other)
        return labelslist, valueslist
```

```
In [24]: allBNF = list(BNFdf["BNF Chapter"].unique())
         allBNF = list(map(int, allBNF))
         allBNF = sorted(allBNF)
```

```
costs = groupbylist(BNFdf, "Actual Cost (č)" , "BNF Chapter")
costPerBNFChapter = dict(zip((allBNF), (costs)))
items = groupbylist(BNFdf, "Total Items", "BNF Chapter" )
itemsPerBNFChapter = dict(zip((allBNF), (items)))

BNFChapterslist1, valsperBNFChapters1 = createValuesandLabels(costPerBNFChapter, 8, o
BNFChapterslist2, valsperBNFChapters2 = createValuesandLabels(itemsPerBNFChapter, 8, 
newBNFlabels = replaceChaptersWithDescription(BNFChapterslist1)
newBNFlabels1 = replaceChaptersWithDescription(BNFChapterslist2)
```

```
In [28]: import matplotlib.pyplot as plt
         fig1, (ax1, ax2) = plt.subplots(2)

         ax1.pie(valsperBNFChapters1, labels = newBNFlabels,  autopct = '%1.1f%%',\
                 shadow = False, startangle = 90,  textprops={'fontsize': 55} )
         ax1.set_title('Actual cost (č) of each Prescription/Preparation from June 2015 - June
                   fontsize = 70, fontweight = 1000, pad = 50.5)
         ax1.axis('equal')

         ax2.pie(valsperBNFChapters2, labels = newBNFlabels1,  autopct = '%1.1f%%',\
                 shadow = False, startangle = 90,  textprops={'fontsize': 55} )
         ax2.set_title('Amount of Prescriptions/Preparations given out from June 2015 - June 2
                   fontsize = 70, fontweight = 1000, pad = 50.5)
         ax2.axis('equal')
         fig1.set_size_inches(60,60)
         plt.style.use('seaborn-muted')
         plt.show()
```





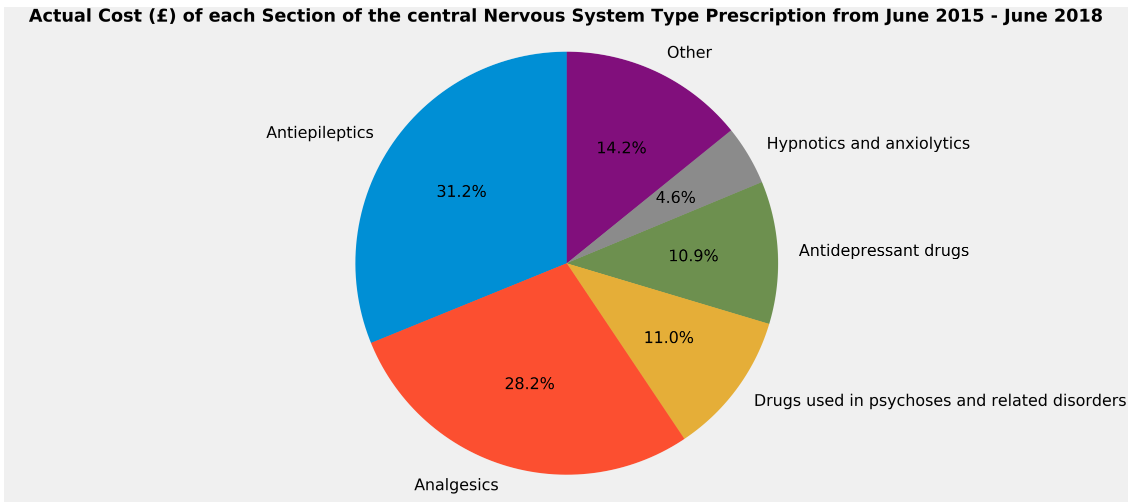
1.3 Central Nervous System Prescriptions

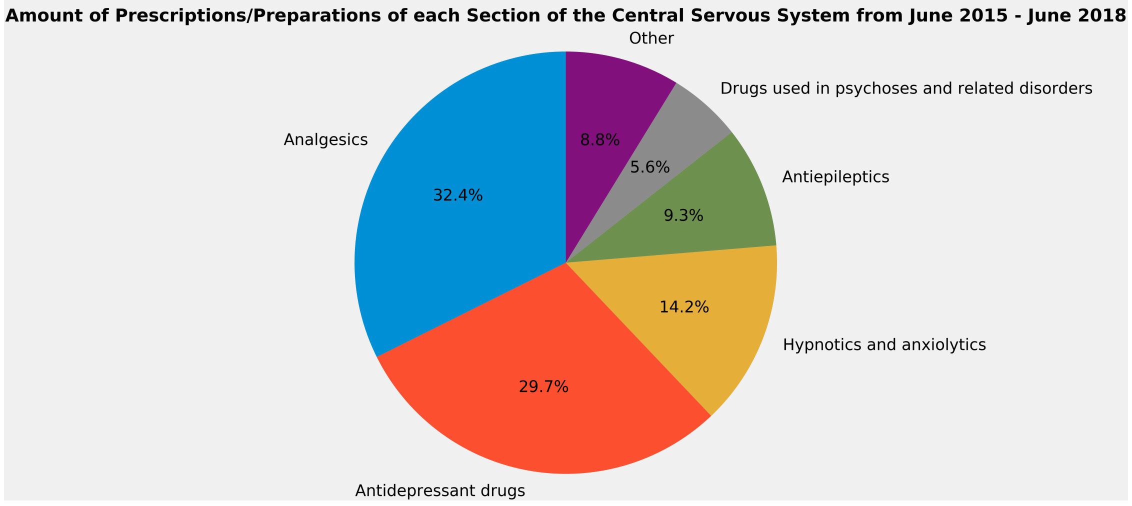
With the above visualisation we can determine that drugs used to treat the central nervous system are of the biggest cost and the most prescribed in the NHS consistently and clearly. By investigating this category of drug more closely we can harness more of a cost saving if inefficiencies are diagnosed and rectified

```
In [29]: CNSdf = concatenate_files(r'C:\Users\fionn\OneDrive\Documents\3062dataanalysis\Hospita
                                   'Practice', 'PRACTICE')
         CNSdf = CNSdf[CNSdf["BNF Chapter"] != '-']
         CNSdf["BNF Chapter"] = pd.to_numeric(CNSdf["BNF Chapter"])
         CNSdf["BNF Section"] = pd.to_numeric(CNSdf["BNF Section"])
         CNSdf = CNSdf[['BNF Chapter', 'Actual Cost (č)', 'Total Items', 'Month', 'Year', 'BNF
         CNSdf = CNSdf.dropna()
         CNSdf = CNSdf[CNSdf["BNF Chapter"] == 4]
         allBNFCodes = list(CNSdf["BNF Code"].unique())
         print("From June 2015 - June 2018 there were", int(mySum(CNSdf["Total Items"])),\
                  "prescriptions dispensed relating to the Central Nervous System, with", l
             "specific types of prescriptions (generic and branded)")
         CNSdf.reset_index(drop = True, inplace = True)
         CNSdf.head()

From June 2015 - June 2018 there were 31119907 prescriptions dispensed relating to the Central


Out[29]:    BNF Chapter  Actual Cost (č)  Total Items  Month    Year  BNF Section  \
         0          4.0             3.81          2.0    4.0  2016.0          2.0
         1          4.0             6.75          2.0    4.0  2016.0          2.0
         2          4.0             5.02          1.0    4.0  2016.0          2.0
         3          4.0             8.44          4.0    4.0  2016.0          3.0
         4          4.0            10.77          8.0    4.0  2016.0          3.0

                   BNF Code         VTM_NM
         0  0402010A0AAAGAG    Amisulpride
         1  0402010A0AAAAAA    Amisulpride
         2  0402010A0AAABAB    Amisulpride
         3  0403010B0AAAGAG  Amitriptyline
         4  0403010B0AAAHAH  Amitriptyline

In [30]: allSec = list(CNSdf["BNF Section"].unique())
         allSec = list(map(int, allSec))
         allSec = sorted(allSec)
         costPerSection = groupbylist(CNSdf, "Actual Cost (č)", "BNF Section")

         itemsPerSection = groupbylist(CNSdf, "Total Items", "BNF Section")


         costPerSection = dict(zip((allSec), (costPerSection)))
         itemsPerSection = dict(zip((allSec), (itemsPerSection)))

In [31]: def replaceSectionsWithDescription(sectionlist):
             for index, item in enumerate(sectionlist, start = 0):
                 item = int(item)
                 if item == 1:
                     sectionlist[index] = "Hypnotics and anxiolytics"

                 elif item == 2:
                     sectionlist[index] = "Drugs used in psychoses and related disorders"

                 elif item == 3:
                     sectionlist[index] = "Antidepressant drugs"

                 elif item == 4:
                     sectionlist[index] = "CNS stimulants and drugs used for attention deficie

                 elif item == 5:
```

```
                sectionlist[index] = "Drugs used in the treatment of obesity"

            elif item == 6:
                sectionlist[index] = "Drugs used in nausea and vertigo"

            elif item == 7:
                sectionlist[index] = "Analgesics"

            elif item == 8:
                sectionlist[index] = "Antiepileptics"

            elif item == 9:
                sectionlist[index] = "Drugs used in Parkinsonism and related disorders"

            elif item == 10:
                sectionlist[index] = "Drugs used in substance dependence"

            elif item == 11:
                 sectionlist[index] = "Drugs for dementia"

            else:
                 sectionlist[index]  = "Other"

        return sectionlist
```

```
In [34]: newBNFSection1, costPerSection1 = createValuesandLabels(costPerSection, 5, otherlabel
         newBNFSection2, itemsPerSection1 = createValuesandLabels(itemsPerSection, 5, otherlabe
         newBNFSection1 = replaceSectionsWithDescription(newBNFSection1)
         newBNFSection2 = replaceSectionsWithDescription(newBNFSection2)

         fig1, (ax1, ax2) = plt.subplots(2)
         ax1.pie(costPerSection1, labels = newBNFSection1, autopct = '%1.1f%%',\
                 shadow = False, startangle = 90,  textprops={'fontsize': 90} )
         ax1.set_title('Actual Cost (č) of each Section of the central Nervous System Type Pres
                   fontsize = 100, fontweight = 1000, pad = 50.5)
         ax1.axis('equal')


         ax2.pie(itemsPerSection1, labels = newBNFSection2, autopct = '%1.1f%%',\
                 shadow = False, startangle = 90, textprops={'fontsize': 90} )

         ax2.set_title('Amount of Prescriptions/Preparations of each Section of the Central Se
                   fontsize = 100, fontweight = 1000, pad = 50.5)
         ax2.axis('equal')
         fig1.set_size_inches(100,100)
         plt.style.use('fivethirtyeight')
         plt.show()
```





I show the customer all four pie charts above to explain why exactly I decided to pinpoint antieplieptics for my investigation

1.4 Antiepileptics

A curious finding here, for me personally, at this juncture was that antiepileptics were the largest cost to the NHS. What's striking about this is that it outstrips the cost of analgesics (painkillers) and antidepressants even though it only is prescribed a third as much and also it's relative obscurity in pop culture in comparison to these prescriptions. The Lehman's solution to this dilemma would probably be to legalise CBD oil as it has been portrayed as a champion cure for seizures especially - however what is not known to the lehman is that the cost per year's dose of CBD oil is much greater than most AED preparations. As this report's prejortive is determining realistic and attainable cost saving in the present day, we will not look into CBD oil Another broad Lehman's solution to saving money is the switching from name brand products to generic brand products, this kind of thinking can be transferred over to how we look at prescription drugs. There is little concern with switching from brand name AEDs to generic AEDs, in fact according to NICE

(National Institute for Health and Care Excellence) who are the authority in the UK on GP medications:
"Where non-proprietary ('generic') titles are given, they should be used in prescribing. This will enable any suitable product to be dispensed, thereby saving delay to the patient and sometimes expense to the health service. The only exception is where there is a demonstrable difference in clinical effect between each manufacturer's version of the formulation, making it important that the patient should always receive the same brand; in such cases, the brand name or the manufacturer should be stated."

Source: https://bnf.nice.org.uk/guidance/guidance-on-prescribing.html Also via studies of the American Epilepsy society, the switching between Branded and Generic does not damage the effectiveness of treatment: Source: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3280470/

```
In [35]: antdf = CNSdf[CNSdf["BNF Section"] == 8]
         antdf = antdf[antdf["BNF Code"] != '-']
         antdf['Generic'] = antdf['BNF Code'].str[10:12]
         antdf['Generic'].replace('AA', 'Generic',inplace=True)
         antdf.loc[antdf['Generic'] != 'Generic', 'Generic'] = 'Not Generic'
         antdf.reset_index(inplace = True, drop = True)
```

```
In [36]: print(("From June 2015 - June 2018 antiepileptics have cost NI č%.2f:") % \
               (mySum(antdf['Actual Cost (č)'])))

         print("And there were", int(mySum(antdf['Total Items'])), "epilepsy prescriptions give

         allBNFCodes = list(antdf["BNF Code"].unique())
         allEpileptics = list(antdf['VTM_NM'].unique())
         print("\nThere are", len(allBNFCodes),\
                     "specific types of preparations used to treat epilepsy")
         print("But there are only", len(allEpileptics),\
                     "specific drug types (by chemical) that are used to treat epilepsy")
```

```
From June 2015 - June 2018 antiepileptics have cost NI č94130724.87:
And there were 2885658 epilepsy prescriptions given out from June 2015 - June 2018

There are 283 specific types of preparations used to treat epilepsy
But there are only 30 specific drug types (by chemical) that are used to treat epilepsy
```

In three years antiepileptics have cost the NHS č94,130,742.87. There is a lot of saving potential here.

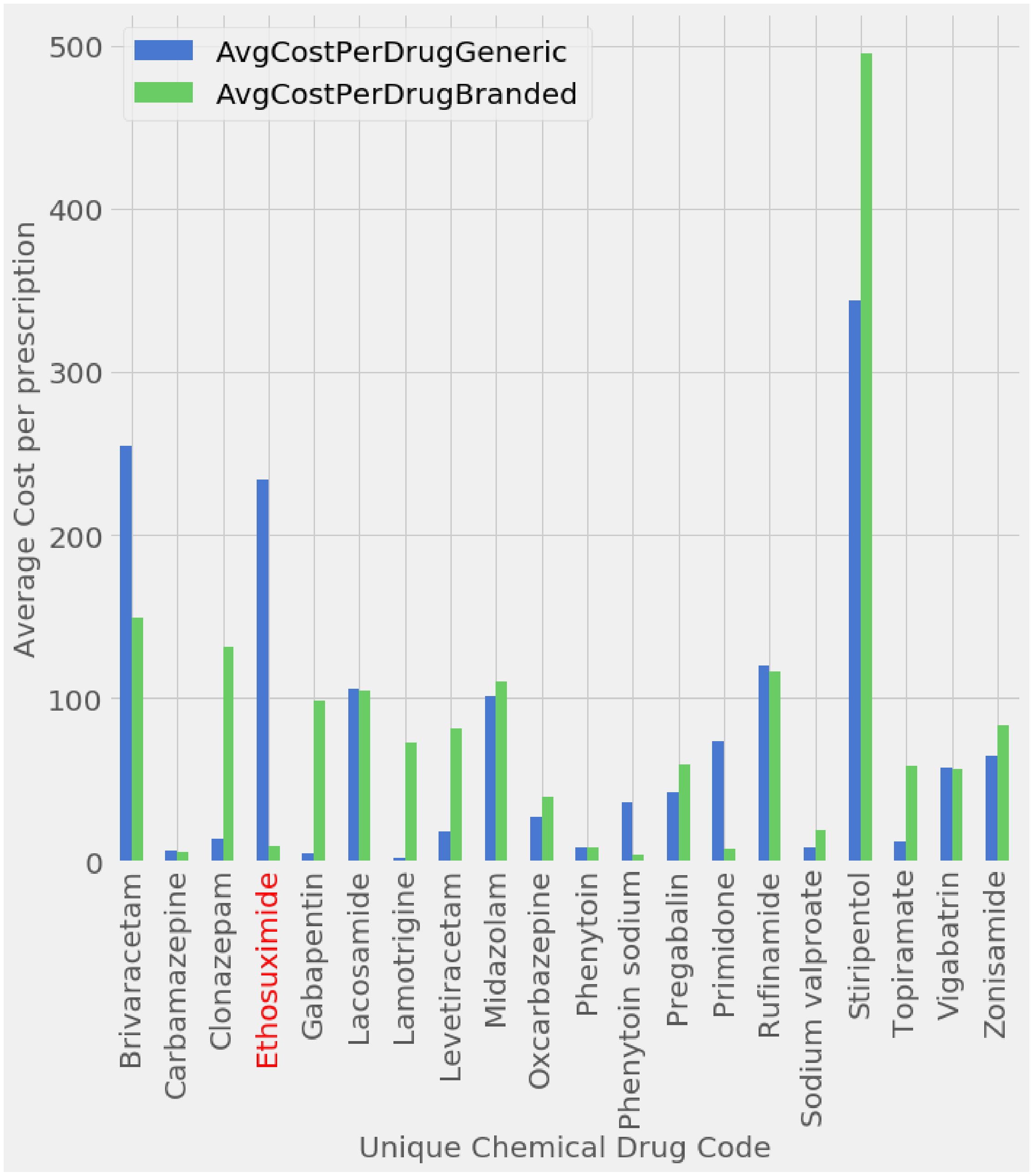
```
In [38]: import numpy as np

         drugsgeneric = antdf[antdf['Generic'] == 'Generic']
         drugsbranded = antdf[antdf['Generic'] != 'Generic']
         drugsbranded = drugsbranded.dropna()

         allDrugs = list(drugsgeneric["VTM_NM"].unique())
         allDrugs = sorted(allDrugs) # Sorting to make sure we divide the same drug cost/items
```

```
        costPerDrug = np.array(groupbylist(drugsgeneric, "Actual Cost (č)", "VTM_NM"))
        itemsPerDrug = np.array(groupbylist(drugsgeneric, "Total Items", "VTM_NM"))
        costPerItemDrug = costPerDrug/itemsPerDrug
        costPerItemDrug = np.round(costPerItemDrug, 2)


        allDrugs1 = list(drugsbranded["VTM_NM"].unique())
        allDrugs1 = sorted(allDrugs1) # Sorting to make sure we divide the same drug cost/ite
        costPerDrug1 = np.array(groupbylist(drugsbranded, "Actual Cost (č)", "VTM_NM"))
        itemsPerDrug1 = np.array(groupbylist(drugsbranded, "Total Items", "VTM_NM"))
        costPerItemDrug1 = costPerDrug1/itemsPerDrug1
        costPerItemDrug1 = np.round(costPerItemDrug1, 2)
```

```
In [39]: generic = pd.DataFrame(
             {'Drug_Name': allDrugs,
              'AvgCostPerDrugGeneric': costPerItemDrug
             })

         nonGeneric = pd.DataFrame(
             {'Drug_Name': allDrugs1,
              'AvgCostPerDrugBranded': costPerItemDrug1
             })
         alldf = pd.merge(generic, nonGeneric, on="Drug_Name")
         alldf.set_index("Drug_Name", inplace = True)
         print(alldf)
```

```
                  AvgCostPerDrugGeneric  AvgCostPerDrugBranded
Drug_Name
Brivaracetam                     254.57                 149.69
Carbamazepine                      6.75                   6.23
Clonazepam                        14.05                 131.64
Ethosuximide                     233.72                   9.39
Gabapentin                         4.67                  98.36
Lacosamide                       105.83                 104.99
Lamotrigine                        2.13                  73.15
Levetiracetam                     18.56                  81.26
Midazolam                        101.36                 110.36
Oxcarbazepine                     26.99                  40.07
Phenytoin                          8.49                   8.20
Phenytoin sodium                  36.55                   3.99
Pregabalin                        42.44                  59.53
Primidone                         73.31                   7.40
Rufinamide                       120.17                 115.98
Sodium valproate                   8.48                  19.50
Stiripentol                      343.91                 495.22
Topiramate                        12.01                  58.09
Vigabatrin                        57.11                  57.10
Zonisamide                        64.60                  83.43
```

```
In [41]: import numpy as np

         np.random.seed(44)
         ax = alldf.plot(kind='bar', figsize=(10,10))
         ax.legend(fontsize=20)
         ax.set_ylabel("Average Cost per prescription",fontsize = 20)
         ax.set_xlabel("Unique Chemical Drug Code",fontsize = 20)
         ax.xaxis.set_tick_params(labelsize = 20)
         ax.yaxis.set_tick_params(labelsize = 20)
         ax.xaxis.get_ticklabels()[3].set_color('red')
         plt.style.use('seaborn-muted')
```



1.5 Ethosuximide

In the majority of cases the generic version of the drug costs less than it's branded counterpart. Although there is a notable exception to the rule - the drug Ethosuximide. This drug's bar label has been highlighted in red.

Statistical analysis below will attempt to investigate why this is so:

N.B Ethosuximide is a drug used to treat abscence seizures in adults and children

```
In [62]: import datetime
         import numpy as np
         dropcols = ["Unnamed: 17", 'Unnamed: 18', 'Gross Cost (č)', \
                     'BNF Chapter', 'BNF Paragraph','BNF Section' , 'BNF Sub-Paragraph']
         ethodrug = concatenate_files(r'C:\Users\fionn\OneDrive\Documents\3062dataanalysis\Hosp
                                   'Practice', 'PRACTICE', dropcols)
         cols = ["Year", "Month"]
         ethodrug[cols] = ethodrug[cols].applymap(np.int64)

         ethodrug["Date"] =  ethodrug["Month"].astype(str) + ethodrug["Year"].astype(str)
         ethodrug['Date'] = pd.to_datetime(ethodrug['Date'], format ='%m%Y')
         ethodrug = ethodrug[ethodrug["BNF Code"] != '-']



         ethodrug['Generic'] = ethodrug['BNF Code'].str[10:12]
         ethodrug['Generic'].replace('AA', 'Generic',inplace=True)
         ethodrug.loc[ethodrug['Generic'] != 'Generic', 'Generic'] = 'Not Generic'

         ethodrug = ethodrug[ethodrug['VTM_NM'] == 'Ethosuximide']
         ethodrug.reset_index(inplace = True, drop = True)

In [63]: drugsbranded = ethodrug[ethodrug['Generic'] != 'Generic']
         drugsbranded.reset_index(inplace = True, drop = True)

         drugsgeneric = ethodrug[ethodrug['Generic'] == 'Generic']
         drugsgeneric.reset_index(inplace = True, drop = True)

         print("\nThere were", int(mySum(drugsgeneric["Total Items"])),\
                     "generic Ethosuximide prescriptions from June 2015 - June 2018, which cos

         print("And", int(mySum(drugsbranded["Total Items"])), \
                      "branded Ethosuximide prescriptions from June 2015 - June 2018")

         print(("\nCost of Ethosuximide from June 15 - June 18  to the NHS in Northern Ireland
               mySum(ethodrug['Actual Cost (č)']))
```

```
There were 2685 generic Ethosuximide prescriptions from June 2015 - June 2018, which cost
And 137 branded Ethosuximide prescriptions from June 2015 - June 2018

Cost of Ethosuximide from June 15 - June 18  to the NHS in Northern Ireland: č628829.06
```

The most honest way to compare these drugs is across similar strengths of the drug and when possible the same presentation - as the measurement in each case is the same at 250mg/5ml. For this reason I will compare the price of the generic version of Ethosuximide with it's branded version over the 250mg/5ml dose

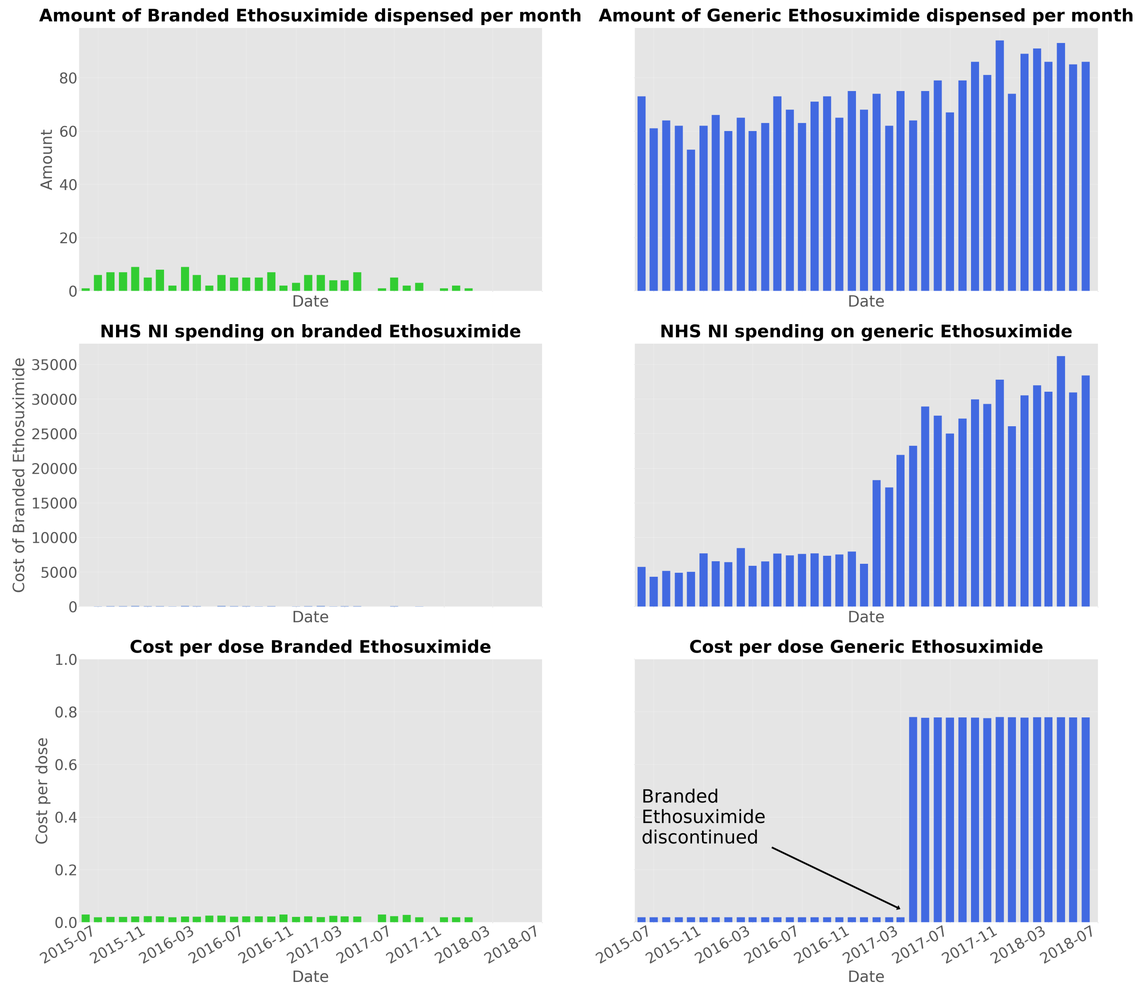
```
In [65]: drugsgeneric1 = drugsgeneric[(drugsgeneric['Presentation'] == 'Oral solution') & \
                                     (drugsgeneric['Strength'] == '250mg/5ml')]
         drugsgeneric1.reset_index(inplace = True, drop = True)

         print("There were", int(mySum(drugsbranded["Total Items"])), \
               "branded Ethosuximide prescriptions - as an oral solution -  from June 15 - June

         costpermlgeneric = mySum(drugsgeneric1['Actual Cost (č)'])/mySum(drugsgeneric1['Total

         print(("\nAverage cost per prescription for generic Ethosuximide was č%.3f per syrup d
                (costpermlgeneric))

         costpermlbranded = mySum(drugsbranded['Actual Cost (č)'])/mySum(drugsbranded['Total Qu
         print(("Average cost per prescription for branded Ethosuximide was č%.3f per syrup dos
               (costpermlbranded))

         times = costpermlgeneric/costpermlbranded
         gencost = mySum(drugsgeneric["Actual Cost (č)"])
         brancost = mySum(drugsbranded["Actual Cost (č)"])
         savings = brancost + gencost/times
         savings = (gencost + brancost) - savings

         print(("\nGeneric Ethosuximide costs the NHS %.2fx more than the branded version on av
         print(("\nIf the NHS in NI used the Branded Ethosuximide instead of the generic versio
```

```
There were 137 branded Ethosuximide prescriptions - as an oral solution -  from June 15 - June

Average cost per prescription for generic Ethosuximide was č0.363 per syrup dose
Average cost per prescription for branded Ethosuximide was č0.022 per syrup dose

Generic Ethosuximide costs the NHS 16.78x more than the branded version on average when they a

If the NHS in NI used the Branded Ethosuximide instead of the generic version from June 2015 -
```

```
In [22]: costPerMonth = groupbylist(ethodrug, "Actual Cost (č)", "Date")
         itemsPerMonth = groupbylist(ethodrug, "Total Items", "Date")
```

```
        ethoBcost = groupbylist(drugsbranded, "Actual Cost (č)", "Date")
        ethoGcost = groupbylist(drugsgeneric, "Actual Cost (č)", "Date")

        ethoBitems = groupbylist(drugsbranded, "Total Items", "Date")
        ethoGitems = groupbylist(drugsgeneric, "Total Items", "Date")

        gendates = sorted(drugsgeneric["Date"].unique())
        brandates = sorted(drugsbranded["Date"].unique())

        Gquanpermonth = np.array(groupbylist(drugsgeneric1, "Total Quantity", "Date" ))
        Gcostpermonth = np.array(groupbylist(drugsgeneric1, "Actual Cost (č)", "Date" ))
        Gcostpermonth = Gcostpermonth/Gquanpermonth

        Bquanpermonth = np.array(groupbylist(drugsbranded, "Total Quantity", "Date" ))
        Bcostpermonth = np.array(groupbylist(drugsbranded, "Actual Cost (č)", "Date" ))
        Bcostpermonth = Bcostpermonth/Bquanpermonth

In [37]: import datetime
        import matplotlib.pyplot as plt
        import matplotlib.dates as mdates
        fig, ((ax5, ax6), (ax3, ax4), (ax1, ax2)) = plt.subplots(3, 2, sharey = 'row', sharex
        fig.set_size_inches(100,100)
        fig.autofmt_xdate()

        ax3.set_xlabel('Date', fontsize = 85, labelpad = 20.0)
        ax3.set_ylabel('Cost of Branded Ethosuximide', fontsize = 85, labelpad = 20.0)
        ax3.set_title('NHS NI spending on branded Ethosuximide', fontsize = 95, fontweight = 
        ax3.grid(True)
        ax3.bar(brandates, ethoBcost, width = 20, )
        ax3.set_xlim([datetime.date(2015, 5, 15),datetime.date(2018, 7, 1)])
        ax3.xaxis.set_tick_params(labelsize = 80)
        ax3.yaxis.set_tick_params(labelsize = 80)

        ax4.set_xlabel('Date', fontsize = 85, labelpad = 20.0)
        ax4.grid(True)
        ax4.bar(gendates, ethoGcost, width = 20, color = "royalblue")# edgecolor = "firebrick
        ax4.set_title('NHS NI spending on generic Ethosuximide', fontsize = 95, fontweight = 
        ax4.set_xlim([datetime.date(2015, 5, 15),datetime.date(2018, 7, 1)])
        ax4.xaxis.set_tick_params(labelsize = 80)


        ax5.set_xlabel('Date', fontsize = 85, labelpad = 20.0)
        ax5.set_ylabel('Amount', fontsize = 85, labelpad = 20.0)
        ax5.set_title('Amount of Branded Ethosuximide dispensed per month', fontsize = 95, fo
        ax5.grid(True)
        ax5.bar(brandates, ethoBitems, width = 20, color = "limegreen")
        ax5.set_xlim([datetime.date(2015, 5, 15),datetime.date(2018, 7, 1)])
```

```
ax5.yaxis.set_tick_params(labelsize = 80)

ax6.set_xlabel('Date', fontsize = 85, labelpad = 20.0)
ax6.set_title('Amount of Generic Ethosuximide dispensed per month', fontsize = 95, for
ax6.grid(True)
ax6.bar(gendates, ethoGitems, width = 20, color = "royalblue")
ax6.set_xlim([datetime.date(2015, 5, 15),datetime.date(2018, 7, 1)])

ax1.set_xlabel('Date', fontsize = 85, labelpad = 20.0)
ax1.set_ylabel('Cost per dose', fontsize = 85, labelpad = 20.0)
ax1.set_title('Cost per dose Branded Ethosuximide', fontsize = 95, fontweight = 800, p
ax1.grid(True)
ax1.bar(brandates, Bcostpermonth, width = 20, color = "limegreen")
ax1.set_xlim([datetime.date(2015, 5, 15),datetime.date(2018, 7, 1)])
ax1.xaxis.set_tick_params(labelsize = 80)
ax1.yaxis.set_tick_params(labelsize =80)

ax2.set_xlabel('Date', fontsize = 85, labelpad = 20.0)
ax2.grid(True)
ax2.bar(gendates, Gcostpermonth, width = 20, color = "royalblue")
ax2.set_title('Cost per dose Generic Ethosuximide', fontsize = 95, fontweight = 800, p
ax2.set_xlim([datetime.date(2015, 5, 15),datetime.date(2018, 7, 1)])
ax1.set_ylim(0, 1)
ax2.xaxis.set_tick_params(labelsize = 80)
ax2.annotate('Branded \nEthosuximide \ndiscontinued', fontsize = 100, xy =(mdates.date
xytext = (mdates.date2num(gendates[0]), 0.3), arrowprops = dict(facecolor = "black",\
                                            headlength = 20, width = 10, headwidth = 3
plt.style.use('ggplot')
```



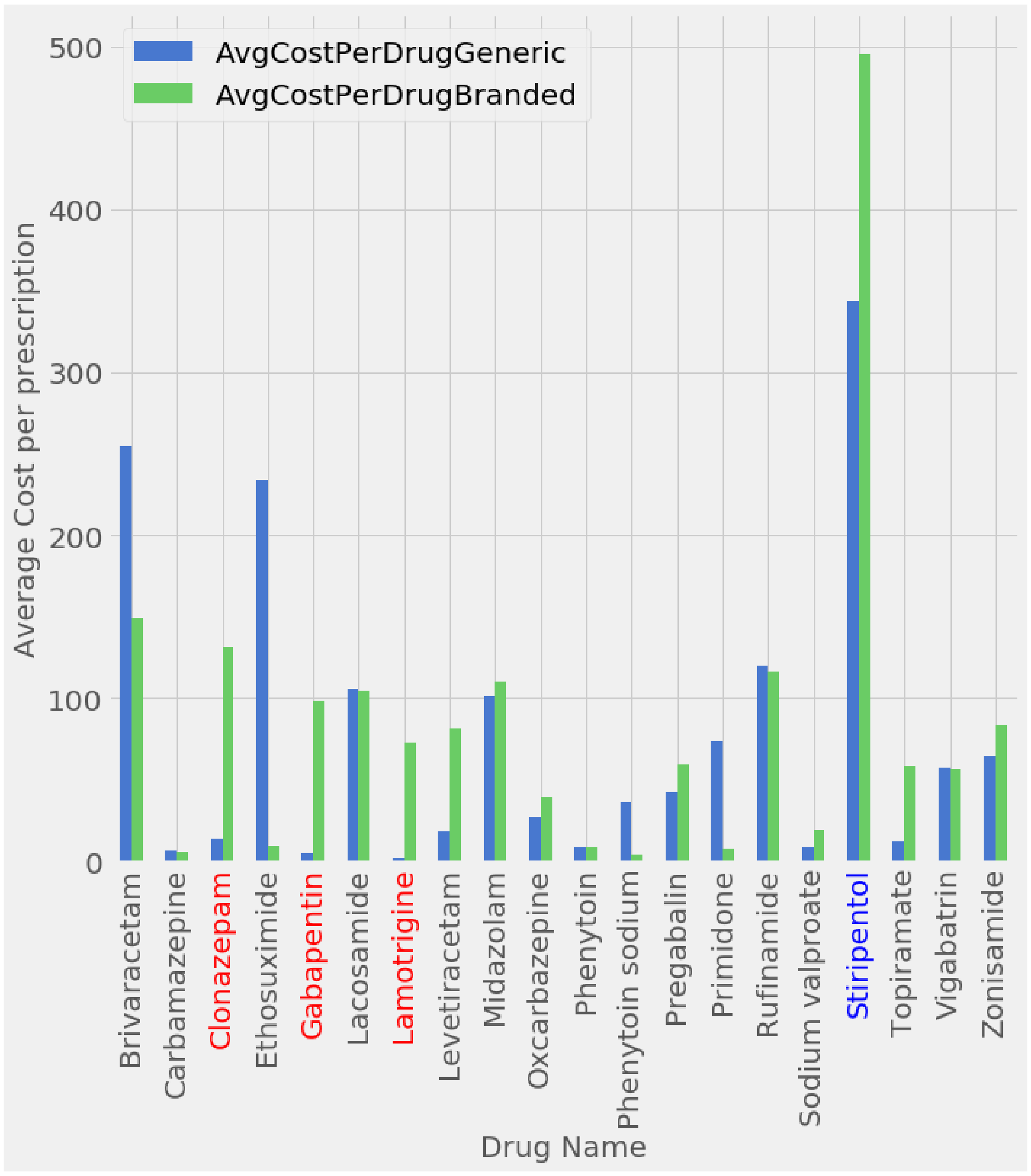
Zarontin and Emeside - (the branded versions of Ethosuximide that were prescribed by the NHS and that are far cheaper) - were discontinued by the pharmacuticals that produced them in mid 2017. As we can clearly see by the above graph this has been reflected in the sudden change in price of the generic version of Ethosuximide. When the generic version became the only available option, it rose it's prices steeply. This demonstrates there is still a possibility of a cheaper alternative to the prescribing of Ethosuximide than the way that the NHS seems to be doing at the moment, as these branded prescriptions showed that there is a cheaper way to dispense Ethosuximide than relying on the more expensive generic versions that the NHS is currently prescribing.

I would suggest that NICE (National Institute for Health and Care Excellence) the de-facto regulatory body of the NHS, should maybe look into finding a cheaper alternative Ethosuximide, either another branded or generic version of the drug and approve it's use. The potential for savings is massive, also they could perhaps lobby pfizer or another pharma giant to make the branded Ethosuximide again for UK use. I would not be surprised if this has gone unnoticed, as in the grand scheme of things, the amount of people prescribed this drug is very small (some 70 - 90 prescriptions a month in NI). However, this drug costs the NHS a lot of money for the small amount of patients it serves., savings of around č200,000 per calendar year are possible with just a different supplier of Ethosuximide

2 Potential price savings on AEDs if switching from generic to branded

Let's remind ourselves of the price comparison between generic and branded AEDs by re-plotting the bar graph from before

```
In [55]: ax = alldf.plot(kind='bar', figsize=(10,10))
         ax.legend(fontsize=20)
         ax.set_ylabel("Average Cost per prescription",fontsize = 20)
         ax.set_xlabel("Drug Name",fontsize = 20)
         ax.xaxis.set_tick_params(labelsize = 20)
         ax.yaxis.set_tick_params(labelsize = 20)
         ax.xaxis.get_ticklabels()[4].set_color('red')
         ax.xaxis.get_ticklabels()[2].set_color('red')
         ax.xaxis.get_ticklabels()[6].set_color('red')
         ax.xaxis.get_ticklabels()[16].set_color('blue')
         plt.show()
```



Highlighted above are the Unique drugs that have a large price discrepancy between it's branded drug and it's generic drug, we can do likewise analysis of what was done above and identify what this is costing the NHS. The drugs as highlighted above, are as follows: *** - Clonazepam - Gabapentin - Lamotrigine *** Stiripentol, (in blue), while appearing to be the largest differential between branded and generic pricings of the drug is actually the same. The average difference in cost is more of a reflection of the way that branded and generic versions of the drug are prescribed - the branded version is usually of a larger strength/dose and this explains the contrast in cost

I decided not to include an in depth analysis of stiripentol for my customer report, as in my other notebook I found it did not give a noteworthy result, I thought about dropping it as a measured drug in my graph but I decided it would not be an accurate representation or at least not a completely honest representationof the antiepileptic drugs as a whole. A quick explanation,I thought, would suffice.

```
In [57]: dropcols = ['Unnamed: 17','Unnamed: 18', 'Gross Cost (č)', \
                     'BNF Chapter', 'BNF Paragraph','BNF Section' , 'BNF Sub-Paragraph' ]
         aeddrug = concatenate_files(r'C:\Users\fionn\OneDrive\Documents\3062dataanalysis\Hospi
                                     'Practice', 'PRACTICE', dropcols)

         aeddrug['Unique_Chemical_Drug'] = aeddrug['BNF Code'].str[:9]
         aeddrug = aeddrug.loc[(aeddrug['VTM_NM'] == 'Clonazepam') | \
                     (aeddrug['VTM_NM'] == 'Gabapentin') | \
                     (aeddrug['VTM_NM'] == 'Lamotrigine')]



         cols = ["Year", "Month"]
         aeddrug[cols] = aeddrug[cols].applymap(np.int64)

         aeddrug["Date"] =  aeddrug["Month"].astype(str) + aeddrug["Year"].astype(str)
         aeddrug['Date'] = pd.to_datetime(aeddrug['Date'], format ='%m%Y')
         aeddrug = aeddrug[aeddrug["BNF Code"] != '-']
         aeddrug['Generic'] = aeddrug['BNF Code'].str[10:12]

         aeddrug['Generic'].replace('AA', 'Generic',inplace=True)
         aeddrug.loc[aeddrug['Generic'] != 'Generic', 'Generic'] = 'Not Generic'
         aeddrug.reset_index(inplace=True,drop = True)

         drugsbranded = aeddrug[aeddrug['Generic'] != 'Generic']
         drugsbranded.reset_index(inplace = True, drop = True)

         drugsgeneric = aeddrug[aeddrug['Generic'] == 'Generic']
         drugsgeneric.reset_index(inplace = True, drop = True)

In [58]: ClonazepamG = drugsgeneric['VTM_NM'] ==  'Clonazepam'
         GabapentinG = drugsgeneric['VTM_NM'] ==  'Gabapentin'
         LamotrigineG = drugsgeneric['VTM_NM'] ==  'Lamotrigine'

         ClonazepamB = drugsbranded['VTM_NM'] ==  'Clonazepam'
         GabapentinB = drugsbranded['VTM_NM'] ==  'Gabapentin'
         LamotrigineB = drugsbranded['VTM_NM'] ==  'Lamotrigine'
```

The code above is setting up the overarching organised data I'll need for the visualisatins of the cost of each drug as well as the prescrptions, the code below is setting up for when I single out practices as outliers

```
In [60]: import numpy as np
         dropcols = ["VMP_NM", "Total Quantity", "Strength", "Presentation", "AMP_NM", \
                     "Gross Cost (č)","BNF Chapter","BNF Paragraph", "BNF Section", "BNF Sub-Pa
```

```
aed2018 = concatenate_files(r'C:\Users\fionn\OneDrive\Documents\3062dataanalysis\Hospi
cols = ["Year", "Month"]
aed2018[cols] = aed2018[cols].applymap(np.int64)

aed2018 = aed2018.loc[(aed2018['VTM_NM'] == 'Clonazepam') | \
           (aed2018['VTM_NM'] == 'Gabapentin') | \
           (aed2018['VTM_NM'] == 'Lamotrigine')]

aed2018['Generic'] = aed2018['BNF Code'].str[10:12]
aed2018['Generic'].replace('AA', 'Generic', inplace = True)
aed2018.loc[aed2018['Generic'] != 'Generic', 'Generic'] = 'Not Generic'
aed2018.reset_index(drop = True, inplace = True)

aed2018["Date"] =  aed2018["Month"].astype(str) + aed2018["Year"].astype(str)
aed2018['Date'] = pd.to_datetime(aed2018['Date'], format ='%m%Y')

practicejul18 = pd.read_csv(r'C:\Users\fionn\OneDrive\Documents\3062dataanalysis\GP Pr
                            encoding = 'ISO-8859-1', low_memory = False)
practicejul18.drop(["Unnamed: 8", "Unnamed: 9"], axis=1, inplace = True)
practicejul18.reset_index(drop = True, inplace = True)
practicejul18.rename(columns ={"Practice No": "Practice"}, inplace = True)

AED2018 = pd.merge(aed2018, practicejul18, on="Practice", how = "left")

AED2018['Practice Name'] = AED2018['Practice Name'].str.strip()
AED2018['New Practice Name'] = AED2018['Practice Name'].astype(str) + " " + AED2018['

drugsgeneric18 = AED2018[AED2018['Generic'] == 'Generic']
drugsgeneric18.reset_index(inplace = True, drop = True)

drugsbranded18 = AED2018[AED2018['Generic'] != 'Generic']
drugsbranded18.reset_index(inplace = True, drop = True)

ClonazepamB18 = drugsbranded18['VTM_NM'] ==  'Clonazepam'
GabapentinB18 = drugsbranded18['VTM_NM'] ==  'Gabapentin'
LamotrigineB18 = drugsbranded18['VTM_NM'] ==  'Lamotrigine'

ClonazepamG18 = drugsgeneric18['VTM_NM'] ==  'Clonazepam'
GabapentinG18 = drugsgeneric18['VTM_NM'] ==  'Gabapentin'
LamotrigineG18 = drugsgeneric18['VTM_NM'] ==  'Lamotrigine'
```

3 Clonazepam

Clonazepam drug was prescribed 61,729 times in NI from June 2015 - June 2018. To ensure an accurate as possible cost analysis, I will only compare prices of the generic and branded versions of the drugs at the same strength

```
In [41]: print("\nThere were", int(mySum(drugsgeneric[ClonazepamG]["Total Items"])), \
               "generic prescriptions of Clonazepam from June 2015 - June 2018")

         print("and there were", int(mySum(drugsbranded[ClonazepamB]["Total Items"])), \
               "branded prescriptions of Clonazepam from June 2015 - June 2018")
```

```
There were 59284 generic prescriptions of Clonazepam from June 2015 - June 2018
and there were 2463 branded prescriptions of Clonazepam from June 2015 - June 2018
```

```
In [42]: import numpy as np

         compar = drugsbranded[ClonazepamB]
         compar = compar.loc[compar['Strength'] == '500microgram/5ml']
         compar.reset_index(drop = True, inplace =True)

         GBquanpermonth = np.array(groupbylist(compar, "Total Quantity", "Date" ))
         GBcostpermonth = np.array(groupbylist(compar, "Actual Cost (č)", "Date" ))
         GBcostpermonth = GBcostpermonth/GBquanpermonth

         compar1 = drugsgeneric[ClonazepamG]
         compar1 = compar1.loc[compar1['Strength'] == '500microgram']
         compar1.reset_index(drop = True, inplace =True)

         GCquanpermonth = np.array(groupbylist(compar1, "Total Quantity", "Date" ))
         GCcostpermonth = np.array(groupbylist(compar1, "Actual Cost (č)", "Date" ))
         GCcostpermonth = GCcostpermonth/GCquanpermonth

         costperbrandedC = mySum(compar['Actual Cost (č)'])/mySum(compar['Total Quantity'])
         costpergenericC = mySum(compar1['Actual Cost (č)'])/mySum(compar1['Total Quantity'])


         print(("Average cost per prescription for generic Clonazepam was č%.3f/500 micrograms'
                (costpergenericC))
         print(("Average cost per prescription for branded Clonazepam was č%.3f/500 micrograms'
                (costperbrandedC))


         times = costperbrandedC/costpergenericC
         Cgencost = mySum(drugsgeneric[ClonazepamG]["Actual Cost (č)"])
         Cbrancost = mySum(drugsbranded[ClonazepamB]["Actual Cost (č)"])
         savings = Cgencost + Cbrancost/times
         savings = (Cgencost + Cbrancost) - savings

         print(("\nBranded Clonazepam costs the NHS %.2fx more than the generic version on ave
         print(("\nIf the NHS used the generic Clonazepam instead of the branded version from
```

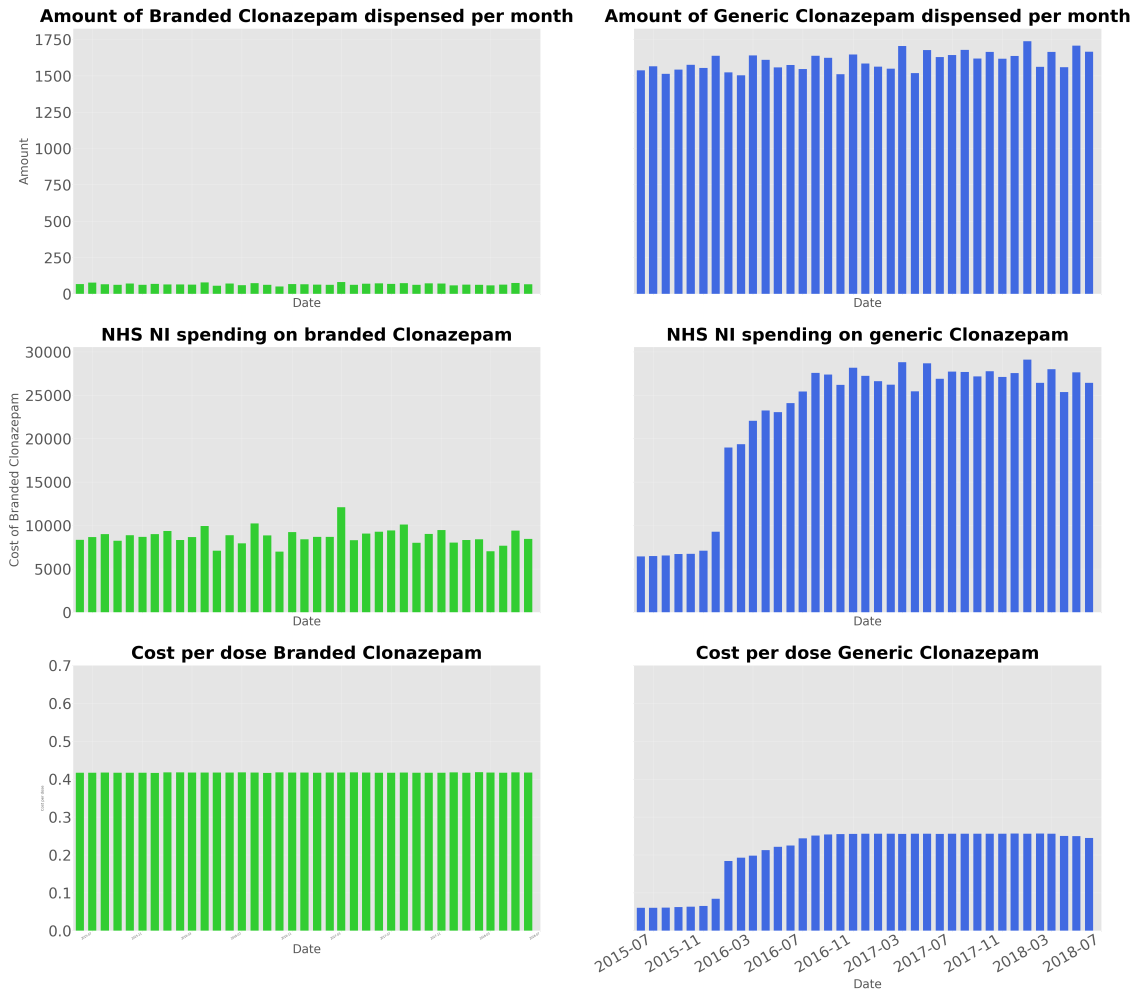
```
Average cost per prescription for generic Clonazepam was č0.211/500 micrograms
Average cost per prescription for branded Clonazepam was č0.417/500 micrograms

Branded Clonazepam costs the NHS 1.97x more than the generic version on average when they are 

If the NHS used the generic Clonazepam instead of the branded version from June 2015 - June 20
```

Based on this informaton, one could say it's an easy suggestion to persuade GP practices to switch from branded to generic Clonazepam as the cost saving on the NHS is evident. However in my research into AEDs online I found that sometimes GPs may prescribe patients with liquid form medication for extreme cases as some epileptic patients struggle to consume their medication in tabular form sometimes - especially small children.

Whenever this is not the case however, the switch from branded Clonazepam to generic should be done to save costs.

```
In [43]: compar = drugsbranded[ClonazepamB]
         compar.reset_index(drop = True, inplace =True)

         compar1 = drugsgeneric[ClonazepamG]
         compar1.reset_index(drop = True, inplace =True)

         brandates = sorted(compar["Date"].unique())
         gendates = sorted(compar1["Date"].unique())


         clonBitems = groupbylist(compar, "Total Items", "Date")
         clonGitems = groupbylist(compar1, "Total Items", "Date")

         clonBcost = groupbylist(compar, "Actual Cost (č)", "Date")
         clonGcost = groupbylist(compar1, "Actual Cost (č)", "Date")

         compar2 = drugsbranded[ClonazepamB]
         compar2 = compar2.loc[compar2['Strength'] == '500microgram/5ml']
         compar2.reset_index(drop = True, inplace =True)

         GBquanpermonth = np.array(groupbylist(compar2, "Total Quantity", "Date" ))
         GBcostpermonth = np.array(groupbylist(compar2, "Actual Cost (č)", "Date" ))
         GBcostpermonth = GBcostpermonth/GBquanpermonth

         compar3 = drugsgeneric[ClonazepamG]
         compar3 = compar3.loc[compar3['Strength'] == '500microgram']
         compar3.reset_index(drop = True, inplace =True)

         GCquanpermonth = np.array(groupbylist(compar3, "Total Quantity", "Date" ))
         GCcostpermonth = np.array(groupbylist(compar3, "Actual Cost (č)", "Date" ))
         GCcostpermonth = GCcostpermonth/GCquanpermonth

In [45]: fig, ((ax5, ax6), (ax3, ax4), (ax1, ax2)) = plt.subplots(3, 2, sharey = 'row', sharex 
         fig.set_size_inches(100,100)
         fig.autofmt_xdate()


         ax3.set_xlabel('Date', fontsize = 65, labelpad = 20.0)
         ax3.set_ylabel('Cost of Branded Clonazepam', fontsize = 65, labelpad = 20.0)
         ax3.set_title('NHS NI spending on branded Clonazepam', fontsize = 95, fontweight = 800
         ax3.grid(True)
         ax3.bar(brandates, clonBcost, width = 20, color = "limegreen")
         ax3.set_xlim([datetime.date(2015, 5, 15),datetime.date(2018, 7, 1)])
         ax3.xaxis.set_tick_params(labelsize = 80)
         ax3.yaxis.set_tick_params(labelsize = 80)

         ax4.set_xlabel('Date', fontsize = 65, labelpad = 20.0)
         ax4.grid(True)
         ax4.bar(gendates, clonGcost, width = 20, color = "royalblue")
         ax4.set_title('NHS NI spending on generic Clonazepam', fontsize = 95, fontweight = 800
         ax4.set_xlim([datetime.date(2015, 5, 15),datetime.date(2018, 7, 1)])
         ax4.xaxis.set_tick_params(labelsize = 80)


         ax5.set_xlabel('Date', fontsize = 65, labelpad = 20.0)
         ax5.set_ylabel('Amount', fontsize = 65, labelpad = 20.0)
         ax5.set_title('Amount of Branded Clonazepam dispensed per month', fontsize = 95, fontw
         ax5.grid(True)
         ax5.bar(brandates, clonBitems, width = 20, color = "limegreen")
         ax5.set_xlim([datetime.date(2015, 5, 15),datetime.date(2018, 7, 1)])
         ax5.yaxis.set_tick_params(labelsize = 80)

         ax6.set_xlabel('Date', fontsize = 65, labelpad = 20.0)
         ax6.set_title('Amount of Generic Clonazepam dispensed per month', fontsize = 95, fontw
         ax6.grid(True)
         ax6.bar(gendates, clonGitems, width = 20,color = "royalblue")
         ax6.set_xlim([datetime.date(2015, 5, 15),datetime.date(2018, 7, 1)])

         ax1.set_xlabel('Date', fontsize = 65, labelpad = 20.0)
         ax1.set_ylabel('Cost per dose', fontsize = 20, labelpad = 20.0)
         ax1.set_title('Cost per dose Branded Clonazepam', fontsize = 95, fontweight = 800, pad
         ax1.grid(True)
         ax1.bar(brandates, GBcostpermonth, width = 20, color = "limegreen")
         ax1.set_xlim([datetime.date(2015, 5, 15),datetime.date(2018, 7, 1)])
         ax1.xaxis.set_tick_params(labelsize = 15)
         ax1.yaxis.set_tick_params(labelsize =80)

         ax2.set_xlabel('Date', fontsize = 65, labelpad = 20.0)
         ax2.grid(True)
         ax2.bar(gendates, GCcostpermonth, width = 20, color = "royalblue")
         ax2.set_title('Cost per dose Generic Clonazepam', fontsize = 95, fontweight = 800, pad
         ax2.set_xlim([datetime.date(2015, 5, 15),datetime.date(2018, 7, 1)])
```

```
ax2.set_ylim(0, 0.7)
ax2.xaxis.set_tick_params(labelsize = 80)
```



As it is so clealy demonstrated above; the cost of branded Clonazepam is nearly double that of it's generic counterpart. Although it is rarely prescribed in comparison to the generic version. Still though, money could be saved if the needless prescribing of the branded Clonazepam was eradicated - (around č53,000 per calendar year). We can pinpoint which practices in Northern Ireland prescribe the branded version the most by pinpointing them as outliers in how much they spend on branded Clonazepam compared to the generic Clonazepam.

```
In [7]: allpracticeCB = list(drugsbranded18[ClonazepamB18]["New Practice Name"].unique())
        allpracticeCB = sorted(allpracticeCB)

        ClonazepamCostB = groupbylist(drugsbranded18[ClonazepamB18], "Actual Cost (č)", "New P

        allpracticeCG = list(drugsgeneric18[ClonazepamG18]["New Practice Name"].unique())
        allpracticeCG = sorted(allpracticeCG)
        ClonazepamCostG = groupbylist(drugsgeneric18[ClonazepamG18], "Actual Cost (č)", "New P

        brandedclon = pd.DataFrame(
            {'Practice_Name': allpracticeCB,
             'Branded_Spend': ClonazepamCostB,
             #'Branded_Prescriptions': ClonazepamItemsB
            })
        genericclon =  pd.DataFrame(
            {'Practice_Name': allpracticeCG,
             'Generic_Spend': ClonazepamCostG,
             #'Generic_Prescriptions': ClonazepamItemsG
            })
        clon = pd.merge(brandedclon, genericclon, on="Practice_Name")
        clon["Spend_diff"] = clon["Branded_Spend"] - clon["Generic_Spend"]
        q25, q75 = np.percentile(clon['Spend_diff'], [25, 75])
        upper = q75 + 1.5*(q75-q25)
        outlierC = clon[clon['Spend_diff'] > upper]
        outlierC.reset_index(drop = True, inplace = True)
        outlierC.head(10)
```

```
Out[7]:                     Practice_Name  Branded_Spend  Generic_Spend  Spend_diff
        0  Dr. COULTER & PARTNERS BT4 3HZ        2839.99         539.27     2300.72
        1    Dr. DEANE & PARTNER BT48 8NL        1862.94         197.15     1665.79
        2  Dr. GLANCY & PARTNERS BT45 5DD        3120.22         477.48     2642.74
```

These specific practices above may need some notification informing them to stop spending on branded Clonazepam

4 Lamotrigine

Lamotrigine was prescribed 319,274 times in NI from June 2015 - June 2018. To ensure an accurate as possible cost analysis, I will only compare prices of the generic and branded versions of Lamotrigine at the same strength

```
In [46]: print("\nThere were", int(mySum(drugsgeneric[LamotrigineG]["Total Items"])), \
               "generic prescriptions of Lamotrigine from June 2015 - June 2018")

         print("There were", int(mySum(drugsbranded[LamotrigineB]["Total Items"])), \
               "branded prescriptions of Lamotrigine from June 2015 - June 2018")
```

```

There were 141212 generic prescriptions of Lamotrigine from June 2015 - June 2018
There were 178062 branded prescriptions of Lamotrigine from June 2015 - June 2018
```

```
In [47]: compar = drugsbranded[LamotrigineB]
         compar = compar.loc[compar['Strength'] == '100mg']
         compar.reset_index(drop = True, inplace =True)
         #print(compar.head())

         compar1 = drugsgeneric[LamotrigineG]
         compar1 = compar1.loc[compar1['Strength'] == '100mg']
         compar1.reset_index(drop = True, inplace =True)

         costperbrandedL = mySum(compar['Actual Cost (č)'])/mySum(compar['Total Quantity'])
         costpergenericL = mySum(compar1['Actual Cost (č)'])/mySum(compar1['Total Quantity'])


         print(("Average cost per prescription for generic Lamotrigine was č%.3f/500 microgram
                (costpergenericL))
         print(("Average cost per prescription for branded Lamotrigine was č%.3f/500 microgram
                (costperbrandedL))

         times = costperbrandedL/costpergenericL

         Lgencost = mySum(drugsgeneric[LamotrigineG]["Actual Cost (č)"])
         Lbrancost = mySum(drugsbranded[LamotrigineB]["Actual Cost (č)"])
         savings = Lgencost + Lbrancost/times
         savings = (Lgencost + Lbrancost) - savings

         print(("\nBranded Lamotrigine costs the NHS %.2fx more than the generic version on av
         print(("\nIf the NHS used the generic Lamotrigine instead of the branded version from
```

```
Average cost per prescription for generic Lamotrigine was č0.032/500 micrograms
Average cost per prescription for branded Lamotrigine was č1.233/500 micrograms

Branded Lamotrigine costs the NHS 38.86x more than the generic version on average when they ar

If the NHS used the generic Lamotrigine instead of the branded version from June 2015 - June 2
```

This information strongly suggests that the NHS should refrain from prescribing the branded version of Lamotrigine as it has been proven to be far too overpriced, and at approximately an extra cost of č4,229,967 a calender year in NI alone it should be addressed immediately.

There seems to be no advantage or reason for prescribing the branded version of this drug over the generic version. The branded version does not offer any alternative presentation or strength nor does it promise any greater effectiveness.

5 Lamotrigine Price analysis

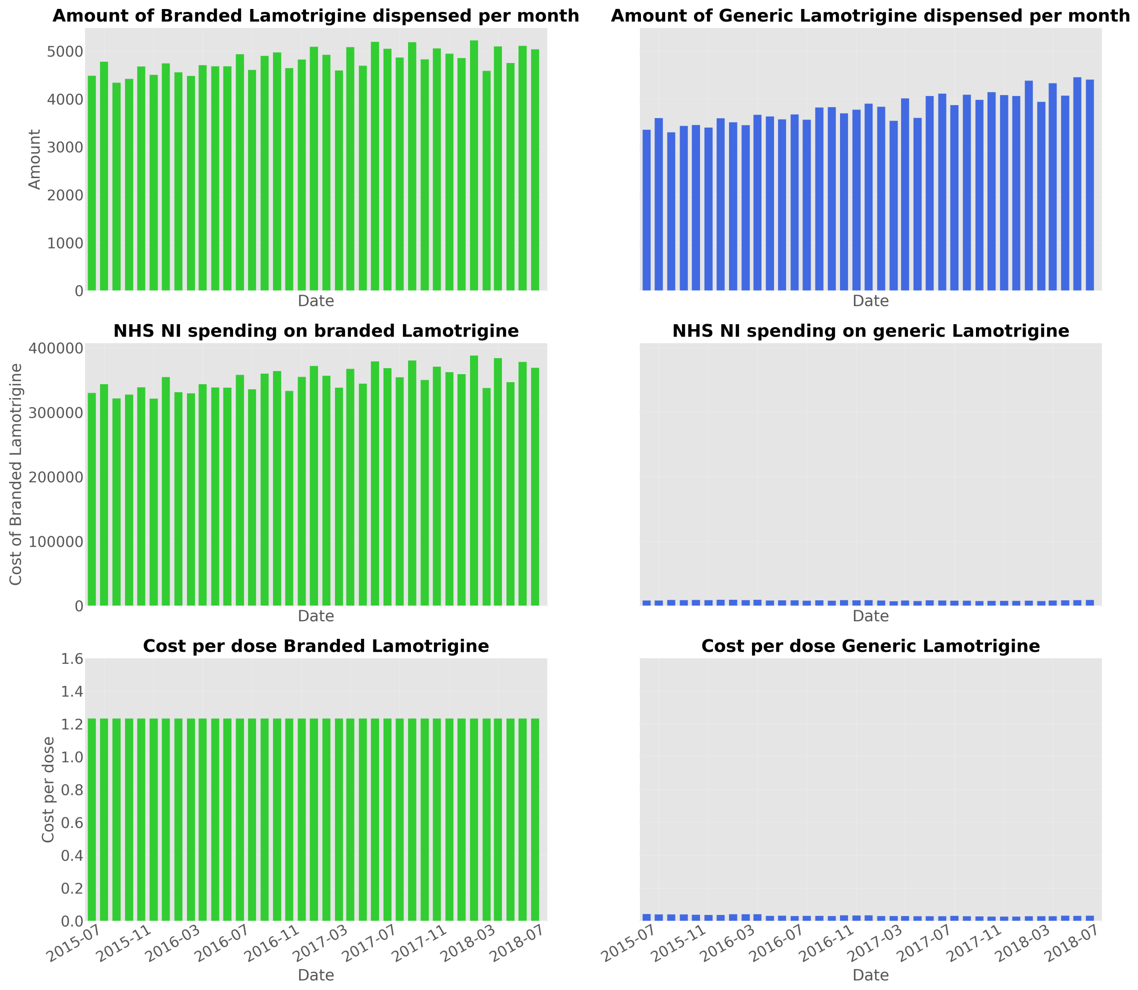
Below is a more direct comparison between the generic and branded versions of Lamotrigine

```
In [73]: compar = drugsbranded[LamotrigineB]
         compar.reset_index(drop = True, inplace =True)
         compar1 = drugsgeneric[LamotrigineG]
         compar1.reset_index(drop = True, inplace =True)
```

```
gendates = sorted(compar1["Date"].unique())
brandates = sorted(compar["Date"].unique())
lamoBitems = groupbylist(compar, "Total Items", "Date")
lamoGitems = groupbylist(compar1, "Total Items", "Date")
lamoBcost = groupbylist(compar, "Actual Cost (č)", "Date")
lamoGcost = groupbylist(compar1, "Actual Cost (č)", "Date")


#For the quantity per cost Branded
qcompar = compar.loc[compar['Strength'] == '100mg']
BLquanpermonth = np.array(groupbylist(qcompar, "Total Quantity", "Date" ))
BLcostpermonth = np.array(groupbylist(qcompar, "Actual Cost (č)", "Date" ))
BLcostpermonth = BLcostpermonth/BLquanpermonth

#For quantity of cost Generic
qcompar1 = compar1.loc[compar1['Strength'] == '100mg']
GLquanpermonth = np.array(groupbylist(qcompar1, "Total Quantity", "Date" ))
GLcostpermonth = np.array(groupbylist(qcompar1, "Actual Cost (č)", "Date" ))
GLcostpermonth = GLcostpermonth/GLquanpermonth
```

```
In [69]: fig, ((ax5, ax6), (ax3, ax4), (ax1, ax2)) = plt.subplots(3, 2, sharey = 'row', sharex
         fig.set_size_inches(100,100)
         fig.autofmt_xdate()

         ax3.set_xlabel('Date', fontsize = 85, labelpad = 20.0)
         ax3.set_ylabel('Cost of Branded Lamotrigine', fontsize = 85, labelpad = 20.0)
         ax3.set_title('NHS NI spending on branded Lamotrigine', fontsize = 95, fontweight = 8
         ax3.grid(True)
         ax3.bar(brandates, lamoBcost, width = 20, color = "limegreen")
         ax3.set_xlim([datetime.date(2015, 5, 15),datetime.date(2018, 7, 1)])
         ax3.xaxis.set_tick_params(labelsize = 80)
         ax3.yaxis.set_tick_params(labelsize = 80)

         ax4.set_xlabel('Date', fontsize = 85, labelpad = 20.0)
         ax4.grid(True)
         ax4.bar(gendates, lamoGcost, width = 20, color = "royalblue")
         ax4.set_title('NHS NI spending on generic Lamotrigine', fontsize = 95, fontweight = 8
         ax4.set_xlim([datetime.date(2015, 5, 15),datetime.date(2018, 7, 1)])
         ax4.xaxis.set_tick_params(labelsize = 80)


         ax5.set_xlabel('Date', fontsize = 85, labelpad = 20.0)
         ax5.set_ylabel('Amount', fontsize = 85, labelpad = 20.0)
         ax5.set_title('Amount of Branded Lamotrigine dispensed per month', fontsize = 95, fon
         ax5.grid(True)
         ax5.bar(brandates, lamoBitems, width = 20, color = "limegreen")
         ax5.set_xlim([datetime.date(2015, 5, 15),datetime.date(2018, 7, 1)])
```

```
ax5.yaxis.set_tick_params(labelsize = 80)

ax6.set_xlabel('Date', fontsize = 85, labelpad = 20.0)
ax6.set_title('Amount of Generic Lamotrigine dispensed per month', fontsize = 95, fon
ax6.grid(True)
ax6.bar(gendates, lamoGitems, width = 20, color = "royalblue")
ax6.set_xlim([datetime.date(2015, 5, 15),datetime.date(2018, 7, 1)])

ax1.set_xlabel('Date', fontsize = 85, labelpad = 20.0)
ax1.set_ylabel('Cost per dose', fontsize = 85, labelpad = 20.0)
ax1.set_title('Cost per dose Branded Lamotrigine', fontsize = 95, fontweight = 800, pa
ax1.grid(True)
ax1.bar(brandates, BLcostpermonth, width = 20, color = "limegreen")
ax1.set_xlim([datetime.date(2015, 5, 15),datetime.date(2018, 7, 1)])
ax1.xaxis.set_tick_params(labelsize = 80)
ax1.yaxis.set_tick_params(labelsize =80)

ax2.set_xlabel('Date', fontsize = 85, labelpad = 20.0)
ax2.grid(True)
ax2.bar(gendates, GLcostpermonth, width = 20, color = "royalblue")
ax2.set_title('Cost per dose Generic Lamotrigine', fontsize = 95, fontweight = 800, pa
ax2.set_xlim([datetime.date(2015, 5, 15),datetime.date(2018, 7, 1)])
ax2.set_ylim(0, 1.6)
ax2.xaxis.set_tick_params(labelsize = 80)
```



These visualisations clearly demonstrate that swift action needs to be taken in regards to replacing the branded Lamotrigine with it's generic counterpart. The saving potential is huge, especially when one considers how easy something like this would be to rectify or at least partially minimise. These graphs show that in NI whilst roughly a similar amount of generic and branded Lamotrigine are dispensed, the spending on the branded Lamotrigine absolutely dwarfs that of the generic version.

```
In [12]: allpracticeLB = list(drugsbranded18[LamotrigineB18]["New Practice Name"].unique())
         allpracticeLB = sorted(allpracticeLB)

         LamotrigineCostB = groupbylist(drugsbranded18[LamotrigineB18], "Actual Cost (č)", "New
         #LamotrigineItemsB = groupbylist(drugsbranded18[LamotrigineB], "Total Items", "New Pr

         allpracticeLG = list(drugsgeneric18[LamotrigineG18]["New Practice Name"].unique())
         allpracticeLG = sorted(allpracticeLG)

         LamotrigineCostG = groupbylist(drugsgeneric18[LamotrigineG18], "Actual Cost (č)", "New
         #LamotrigineItemsG = groupbylist(drugsgeneric18[LamotrigineG], "Total Items", "New Pr
         brandedlam = pd.DataFrame(
             {'Practice_Name': allpracticeLB,
              'Branded_Spend': LamotrigineCostB
              #'Branded_Prescriptions': LamotrigineItemsB
             })
         genericlam =  pd.DataFrame(
             {'Practice_Name': allpracticeLG,
              'Generic_Spend': LamotrigineCostG
              #'Generic_Prescriptions': LamotrigineItemsG
             })
         lam = pd.merge(brandedlam, genericlam, on="Practice_Name")
         lam["Spend_diff"] = lam["Branded_Spend"] - lam["Generic_Spend"]

         q25, q75 = np.percentile(lam['Spend_diff'], [25, 75])
         upper = q75 + 1.5*(q75-q25)
         outlierL = lam[lam['Spend_diff'] > upper]
         outlierL.reset_index(drop = True, inplace = True)
         outlierL.head(100)
```

Out[12]:

```
                         Practice_Name  Branded_Spend  Generic_Spend  \
0       Dr. BAILIE & PARTNERS BT81 7AZ       31489.04         256.73
1       Dr. BOLTON & PARTNERS BT38 7HT       28631.09         172.39
2      Dr. DOHERTY & PARTNERS BT48 7DH       19339.89         285.07
3    Dr. GALLAGHER & PARTNERS BT79 0NR       18070.29         314.67
4     Dr. JOHNSTON & PARTNERS BT53 6HB       20530.91           1.46
5     Dr. LALSINGH & PARTNERS BT48 7GW       33256.52         158.28
6       Dr. LAWSON & PARTNERS BT18 9AD       22111.75         313.58
7        Dr. LEAVY & PARTNERS BT12 7DP       17857.97          30.86
8        Dr. LOGAN & PARTNERS BT39 9HL       23315.25          73.88
9      Dr. MAGUIRE & PARTNERS BT13 1AD       17735.21          80.09
10   Dr. MCCLOSKEY & PARTNERS BT48 7NY       25531.54         280.73
11      Dr. MEENAN & PARTNERS BT37 9UH       20052.38          62.30
12   Dr. O'DONNELL & PARTNERS BT48 7DH       22368.20         164.11
13  Dr. O'FLAHERTY & PARTNERS BT47 6AH       29931.83          92.78

    Spend_diff
0     31232.31
1     28458.70
2     19054.82
3     17755.62
4     20529.45
5     33098.24
6     21798.17
7     17827.11
8     23241.37
9     17655.12
10    25250.81
```

```
11     19990.08
12     22204.09
13     29839.05
```

The practices above are outliers in respect to how much they spend on branded Lamotrigine vs the generic Lamotrigine. In most of these cases the spending on the branded version of the drug absolutely overshadows the spending on the generic drug - which suggests that these practices almost only prescribe the branded version These practices need to be advised better and notified to stop using the branded Lamotrigine as much. The potential for savings is substantial.

6 Gabapentin

Again a relatively common drug, there were 546,419 prescriptions of Gabapentin dispensed in NI from June 2015 - June 2018

N.B This drug is prescribed for children and adults who suffer from focal seizures. It can also be used as a painkiller

```
In [50]: print("\nThere were", int(mySum(drugsgeneric[GabapentinG]["Total Items"])), \
               "generic prescriptions of Gabapentin from June 2015 - June 2018")

         print("There were", int(mySum(drugsbranded[GabapentinB]["Total Items"])), \
               "branded prescriptions of Gabapentin from June 2015 - June 2018")
```

```
There were 536890 generic prescriptions of Gabapentin from June 2015 - June 2018
There were 9529 branded prescriptions of Gabapentin from June 2015 - June 2018
```

```
In [51]: compar = drugsbranded[GabapentinB]
         compar = compar.loc[compar['Strength'] == '300mg']
         compar.reset_index(drop = True, inplace =True)


         compar1 = drugsgeneric[GabapentinG]
         compar1 = compar1.loc[compar1['Strength'] == '300mg']
         compar1.reset_index(drop = True, inplace =True)

         costperbrandedG = mySum(compar['Actual Cost (č)'])/mySum(compar['Total Quantity'])
         costpergenericG = mySum(compar1['Actual Cost (č)'])/mySum(compar1['Total Quantity'])


         print(("Average cost per prescription for generic Gabapentin was č%.3f/300 micrograms'
                (costpergenericG))
         print(("Average cost per prescription for branded Gabapentin was č%.3f/300 micrograms'
                (costperbrandedG))


         times = costperbrandedG/costpergenericG
```

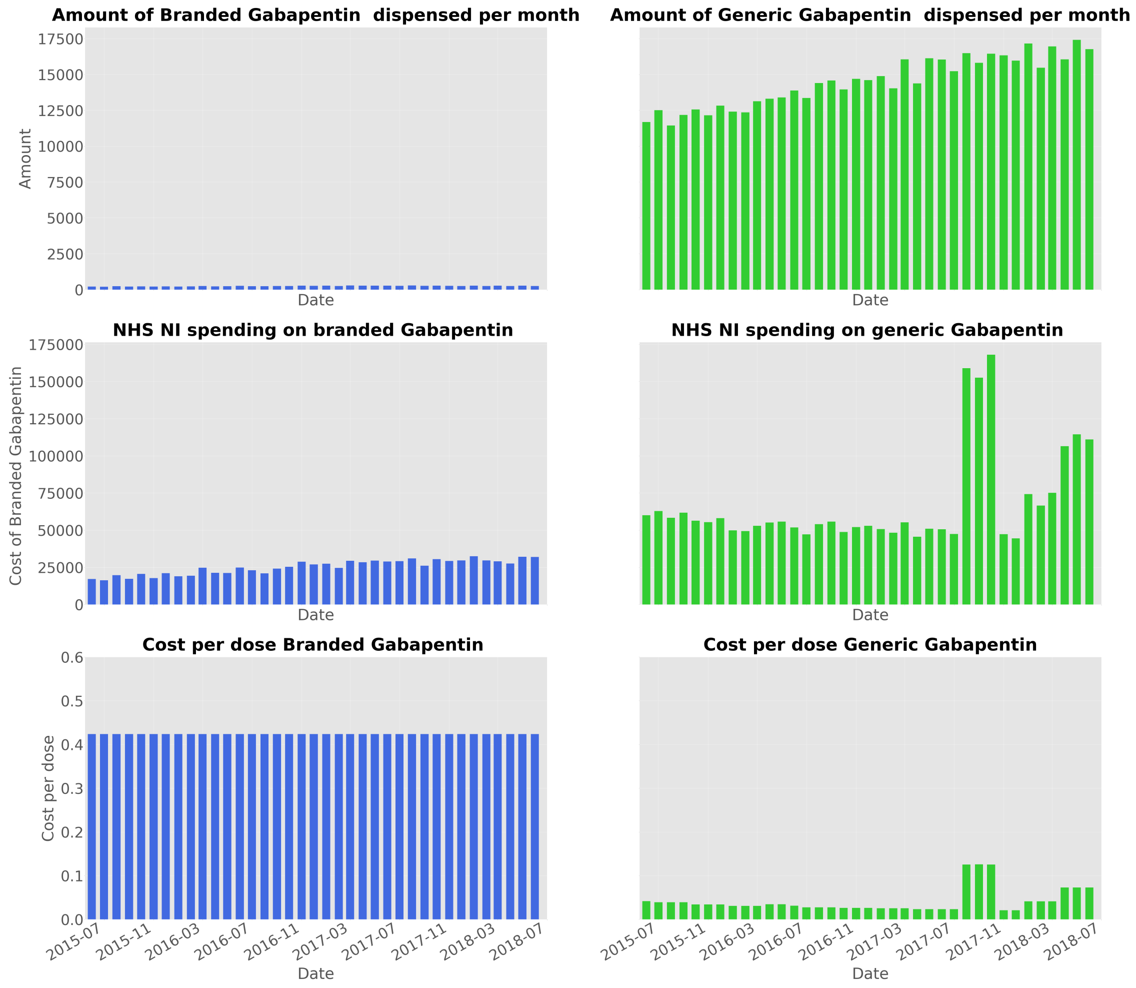
```
        Ggencost = mySum(drugsgeneric[GabapentinG]["Actual Cost (č)"])
        Gbrancost = mySum(drugsbranded[GabapentinB]["Actual Cost (č)"])
        savings = Ggencost + Gbrancost/times
        savings = (Ggencost + Gbrancost) - savings

        print(("\nBranded Gabapentin costs the NHS %.2fx more than the generic version on ave
        print(("\nIf the NHS used the generic Gabapentin instead of the branded version from
```

```
Average cost per prescription for generic Gabapentin was č0.043/300 micrograms
Average cost per prescription for branded Gabapentin was č0.424/300 micrograms

Branded Gabapentin costs the NHS 9.88x more than the generic version on average when they are

If the NHS used the generic Gabapentin instead of the branded version from June 2015 - June 20
```

7 Gabapentin price analysis

```
In [76]: brandates = sorted(compar["Date"].unique())
         gendates = sorted(compar1["Date"].unique())

         compar = drugsbranded[GabapentinB]
         compar.reset_index(drop = True, inplace =True)

         compar1 = drugsgeneric[GabapentinG]
         compar1.reset_index(drop = True, inplace =True)


         gabaBitems = groupbylist(compar, "Total Items", "Date")
         gabaGitems = groupbylist(compar1, "Total Items", "Date")

         gabaBcost = groupbylist(compar, "Actual Cost (č)", "Date")
         gabaGcost = groupbylist(compar1, "Actual Cost (č)", "Date")

         compar2 = drugsbranded[GabapentinB]
         compar2 = compar2.loc[compar2['Strength'] == '300mg']
         compar2.reset_index(drop = True, inplace =True)

         BGquanpermonth = np.array(groupbylist(compar2, "Total Quantity", "Date" ))
         BGcostpermonth = np.array(groupbylist(compar2, "Actual Cost (č)", "Date" ))
         BGcostpermonth = BGcostpermonth/BGquanpermonth

         compar3 = drugsgeneric[GabapentinG]
         compar3 = compar3.loc[compar3['Strength'] == '300mg']
         compar3.reset_index(drop = True, inplace =True)
         GGquanpermonth = np.array(groupbylist(compar3, "Total Quantity", "Date" ))
         GGcostpermonth = np.array(groupbylist(compar3, "Actual Cost (č)", "Date" ))
         GGcostpermonth = GGcostpermonth/GGquanpermonth

In [80]: fig, ((ax5, ax6), (ax3, ax4), (ax1, ax2)) = plt.subplots(3, 2, sharey = 'row', sharex
         fig.set_size_inches(100,100)
         fig.autofmt_xdate()

         ax3.set_xlabel('Date', fontsize = 85, labelpad = 20.0)
         ax3.set_ylabel('Cost of Branded Gabapentin ', fontsize = 85, labelpad = 20.0)
         ax3.set_title('NHS NI spending on branded Gabapentin ', fontsize = 95, fontweight = 80
         ax3.grid(True)
         ax3.bar(brandates, gabaBcost, width = 20, color = "limegreen")
         ax3.set_xlim([datetime.date(2015, 5, 15),datetime.date(2018, 7, 1)])
         ax3.xaxis.set_tick_params(labelsize = 80)
         ax3.yaxis.set_tick_params(labelsize = 80)

         ax4.set_xlabel('Date', fontsize = 85, labelpad = 20.0)
         ax4.grid(True)
         ax4.bar(gendates, gabaGcost, width = 20, color = "royalblue")
         ax4.set_title('NHS NI spending on generic Gabapentin ', fontsize = 95, fontweight = 80
         ax4.set_xlim([datetime.date(2015, 5, 15),datetime.date(2018, 7, 1)])
         ax4.xaxis.set_tick_params(labelsize = 80)

         ax5.set_xlabel('Date', fontsize = 85, labelpad = 20.0)
         ax5.set_ylabel('Amount', fontsize = 85, labelpad = 20.0)
         ax5.set_title('Amount of Branded Gabapentin  dispensed per month', fontsize = 95, font
         ax5.grid(True)
         ax5.bar(brandates, gabaBitems, width = 20, color = "limegreen")
         ax5.set_xlim([datetime.date(2015, 5, 15),datetime.date(2018, 7, 1)])
         ax5.yaxis.set_tick_params(labelsize = 80)

         ax6.set_xlabel('Date', fontsize = 85, labelpad = 20.0)
         ax6.set_title('Amount of Generic Gabapentin  dispensed per month', fontsize = 95, font
         ax6.grid(True)
         ax6.bar(gendates, gabaGitems, width = 20, color = "royalblue")
         ax6.set_xlim([datetime.date(2015, 5, 15),datetime.date(2018, 7, 1)])

         ax1.set_xlabel('Date', fontsize = 85, labelpad = 20.0)
         ax1.set_ylabel('Cost per dose', fontsize = 85, labelpad = 20.0)
         ax1.set_title('Cost per dose Branded Gabapentin ', fontsize = 95, fontweight = 800, pa
         ax1.grid(True)
         ax1.bar(brandates, BGcostpermonth, width = 20, color = "limegreen")
         ax1.set_xlim([datetime.date(2015, 5, 15),datetime.date(2018, 7, 1)])
         ax1.xaxis.set_tick_params(labelsize = 80)
         ax1.yaxis.set_tick_params(labelsize =80)

         ax2.set_xlabel('Date', fontsize = 85, labelpad = 20.0)
```

```
ax2.grid(True)
ax2.bar(gendates, GGcostpermonth, width = 20, color = "royalblue")
ax2.set_title('Cost per dose Generic Gabapentin', fontsize = 95, fontweight = 800, pac
ax2.set_xlim([datetime.date(2015, 5, 15),datetime.date(2018, 7, 1)])
ax2.set_ylim(0, 0.6)
ax2.xaxis.set_tick_params(labelsize = 80)
```



Whilst the overall NHS supply of branded gabapentin pales in comparion to the generic supply, the spending on the branded version is near the spending on the generic version. This is shown by the much larger cost per dose of the branded Gabapentin, which is nearly 10x the price of it's generic counterpart. If the branded spend of Gabapentin was substitued for the generic, the potential in savings would be around č280,000 per calendar year.

```
In [13]: allpracticeGB = list(drugsbranded18[GabapentinB18]["New Practice Name"].unique())
         allpracticeGB = sorted(allpracticeGB)

         GabapentinCostB = groupbylist(drugsbranded18[GabapentinB18], "Actual Cost (č)", "New I
         #GabapentinItemsB = groupbylist(drugsbranded18[GabapentinB], "Total Items", "New Prac
         allpracticeGG = list(drugsgeneric18[GabapentinG18]["New Practice Name"].unique())
         allpracticeGG = sorted(allpracticeGG)

         GabapentinCostG = groupbylist(drugsgeneric18[GabapentinG18], "Actual Cost (č)", "New 
         #GabapentinItemsG = groupbylist(drugsgeneric18[GabapentinG], "Total Items", "New Prac

         brandedgab = pd.DataFrame(
             {'Practice_Name': allpracticeGB,
              'Branded_Spend': GabapentinCostB,
              #'Branded_Prescriptions': ClonazepamItemsB
             })
         genericgab =  pd.DataFrame(
             {'Practice_Name': allpracticeGG,
              'Generic_Spend': GabapentinCostG,
              #'Generic_Prescriptions': ClonazepamItemsG
             })
         gab = pd.merge(brandedgab, genericgab, on="Practice_Name")
         gab["Spend_diff"] = gab["Branded_Spend"] - gab["Generic_Spend"]

         q25, q75 = np.percentile(gab['Spend_diff'], [25, 75])
         upper = q75 + 1.5*(q75-q25)
         outlierG = gab[gab['Spend_diff'] > upper]
         outlierG.reset_index(drop = True, inplace = True)
         outlierG.head(10)
```

```
Out[13]:                        Practice_Name  Branded_Spend  Generic_Spend  Spend_diff
         0     Dr. CHEYNE & PARTNERS BT7 3GG        3312.00         629.53     2682.47
         1   Dr. JOHNSTON & PARTNER BT41 3TF        3364.58         518.01     2846.57
         2  Dr. THOMPSON & PARTNERS BT71 6BX        7078.05        1326.10     5751.95
```

Again, these practices need to be notified to try and minimise their supplying of branded Gabapentin. It costs 10x as much as it's generic counterpart whilst not promising any upgrade in effectiveness that could justify it's use.