In [1]:

```
#changing the default directory
import os
os.chdir("/Users/Avinash/Desktop/CapOne/namesbystate")
```

In [2]:

```
#combining all txt files into a single master file
```

In [3]:

```
import glob
input_files = glob.glob("*.TXT")
with open("combined_file.txt", "w") as output_file:
    for file in input_files:
        with open(file, "r") as input_file:
            output_file.write(input_file.read())
```

In [4]:

```
#importing pandas
import pandas as pd
```

In [5]:

```
#reading the combined text file
baby_names = pd.read_csv("combined_file.txt", sep = ",",header= None)
```

In [6]:

```
#taking a look at the data
baby_names.head()
```

Out[6]:

	0	1	2	3	4
0	AK	F	1910	Mary	14
1	AK	F	1910	Annie	12
2	AK	F	1910	Anna	10
3	AK	F	1910	Margaret	8
4	AK	F	1910	Helen	7

In [7]:

```
#assigning column names
baby_names.columns = ["State","Sex","Year","Name","Frequency"]
```

In [8]:

```
#checking the column names
baby_names.head()
```

Out[8]:

	State	Sex	Year	Name	Frequency
0	AK	F	1910	Mary	14
1	AK	F	1910	Annie	12
2	AK	F	1910	Anna	10
3	AK	F	1910	Margaret	8
4	AK	F	1910	Helen	7

In [9]:

```
#describing the data
baby_names.describe()
```

Out[9]:

	Year	Frequency
count	5647426.000000	5647426.000000
mean	1972.391787	52.923814
std	29.573899	180.810001
min	1910.000000	5.000000
25%	1949.000000	7.000000
50%	1977.000000	13.000000
75%	1999.000000	34.000000
max	2014.000000	10023.000000

In [10]:

```
#Question 1: Please describe the format of the data files. Can you identify any limitations or distor
tions of the data?

#the data is stored in comma separated text files for each state. Based on the readme file available
along with the
#data we can see that unique names with less than 5 frequency are ignored for privacy reason. Also na
mes longer than
#15 letters are also ignored. So we may have missed some unique long names. Finding the actual number
of unique names
#is not possible. Also the data set includes only first name. A person may have more than one word in
their first name.
```

In [11]:

```
#question 2: Most popular name of all time

#defining a function for most popular name
def most_pop_name(arr):

#grouping by the name and then summing the frequencies for all years and genders
    most_popular_names = arr.groupby(by=["Name"])["Frequency"].sum()
    most_popular_names.sort_values(inplace = True, ascending = False)
    print("The most popular name of all time is",most_popular_names.index[0])

most_pop_name(baby_names)
```

```
The most popular name of all time is James
```

In [12]:

```
#Question 3: What is the most gender ambiguous name in 2013? 1945?

#defining a function
def most_gender_ambiguous(arr,year):
    #subsetting for the year requested
    gender_ambi_names = pd.DataFrame(arr[arr["Year"]==year])

    #groupby and sum
    ambi_diff = pd.DataFrame(gender_ambi_names.groupby(["Name","Sex"])["Frequency"].sum()).reset_inde
x()

    #dropping duplicates and sorting
    ambi_diff = ambi_diff.drop_duplicates(subset = "Name", keep = "first").reset_index()
    ambi_total = pd.DataFrame(gender_ambi_names.groupby(["Name"])["Frequency"].sum()).reset_index()
    ambi_total.sort_values(["Name"])
    ambi_diff.sort_values(["Name"])

    #if Male and Female are equally divided then the ratio would be 0. if it is purely male or female
the ratio would be
    #1 and for other cases in between 0 and 1
    ambi_total["Factor"] = abs((2*ambi_diff["Frequency"])/ambi_total["Frequency"]-1)
    ambi_total.sort_values(["Factor","Frequency"],ascending = [True,False],inplace=True)

    #returning the list of all ambiguous names for the year
    return(ambi_total[ambi_total["Factor"]==0])

most_gender_ambiguous(baby_names,2013)
```

Out[12]:

	Name	Frequency	Factor
7136	Nikita	94	0
2260	Cree	22	0
2645	Devine	20	0
1045	Arlin	10	0
8416	Sonam	10	0

In [13]:

```
most_gender_ambiguous(baby_names,1945)
```

Out[13]:

	Name	Frequency	Factor
2187	Maxie	38	0

In [14]:

```
#question 4: Of the names represented in the data, find the name that has had the largest percentage
increase in
#popularity since 1980. Largest decrease?
#for this problem i am only considering names which were there in 1980 and in 2013.

#defining a function
def popularity(arr,year_1,year_2,least_popular = True):
    #subset year_1  and year_2 data
    names_1 = arr[arr["Year"]==year_1]
    names_2 = arr[arr["Year"]==year_2]
    names_1 = pd.DataFrame(names_1.groupby(["Name"])["Frequency"].sum()).reset_index()

    #popularity of each name by dividing the name frequency by total population
    names_1["Popularity"] = names_1["Frequency"]/sum(names_1["Frequency"])
    names_2 = pd.DataFrame(names_2.groupby(["Name"])["Frequency"].sum()).reset_index()
    names_2["Popularity"] = names_2["Frequency"]/sum(names_2["Frequency"])

    #inner merging both data frames to obtain new data frame with common names
    combined_names = pd.merge(names_1, names_2, on='Name', how='inner')
    combined_names.columns = ["Name","Freq_Names_1","Popularity_1","Freq_Names_2","Popularity_2"]

    #calculating change in popularity
    combined_names["Percnt_Change"] = combined_names["Popularity_2"]-combined_names["Popularity_1"]
    combined_names.sort_values("Percnt_Change",ascending=least_popular,inplace = True)
    return (combined_names["Name"].iloc[0])
popularity(baby_names,1980,2013,False)
```

Out[14]:

```
'Sophia'
```

In [15]:

```
popularity(baby_names,1980,2013,True)
```

Out[15]:

```
'Jennifer'
```

In [16]:

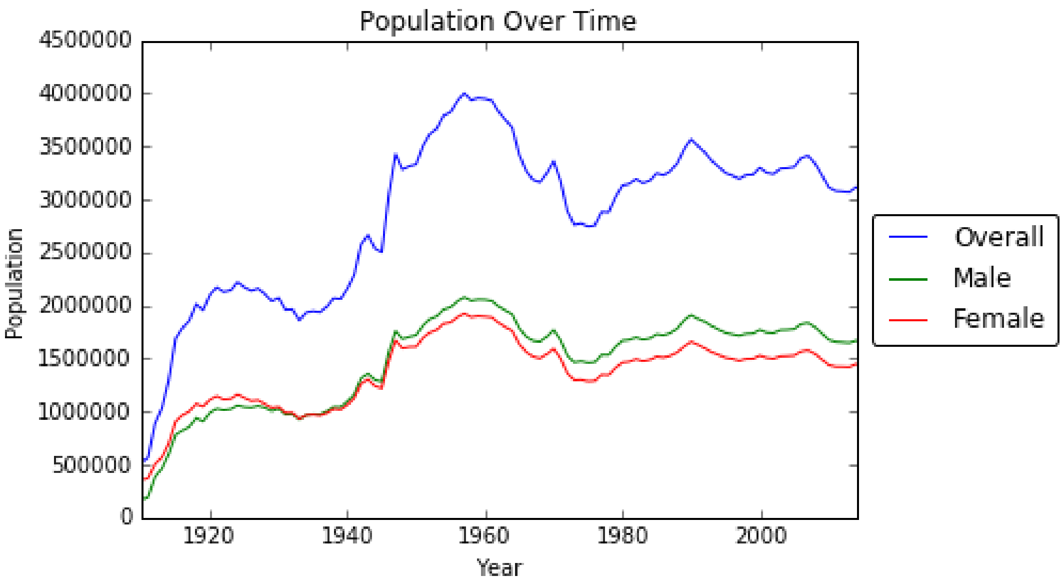
```
#question 5: Can you identify names that may have had an even larger increase or decrease in populari
ty?

#part 4 was tackled with the assumption that the names has to be there in 1980 and 2013. But there ma
y be situations
#where the name could have started after 1980 and became more popular by 2013. Or there could have be
en names which
#were there in 1980 but became unpopular with frequency less than 5 thus not appearing in the 2013 li
st. Similarily if
#names started in 1980 with frequency less than 5 and became more popular by 2013 we cannot identify
them due to basic
#rules of the data set
```

In [17]:

```
#Part - 2
#importing matplot lib for visualization
import matplotlib.pyplot as plt
%matplotlib inline
```

In [18]:

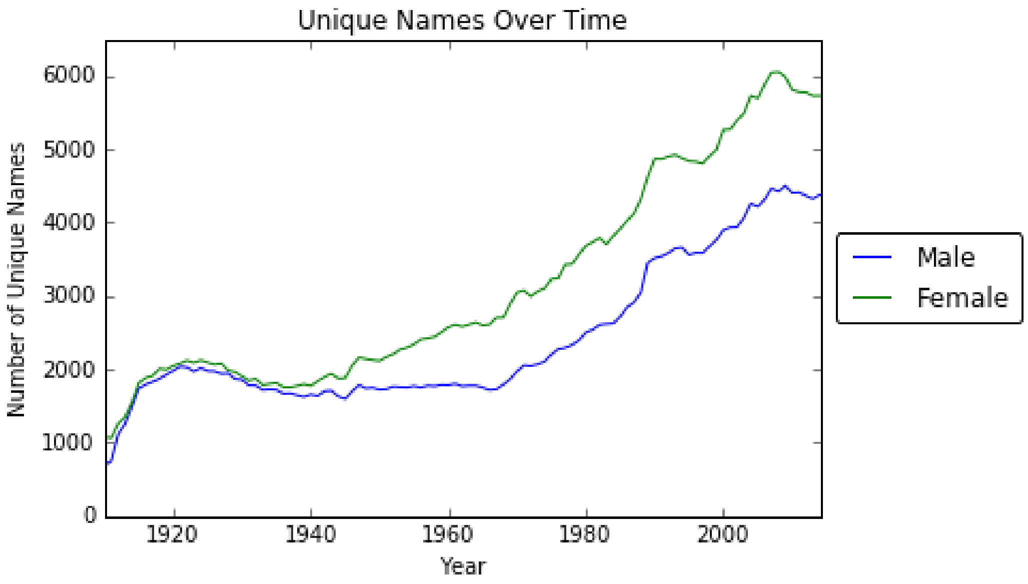
```
#number of births or baby names per year
total_births = pd.DataFrame(baby_names.groupby(["Year"])["Frequency"].sum()).reset_index()

total_births_male = pd.DataFrame(baby_names[baby_names["Sex"]=="M"].groupby(["Year"])["Frequency"].sum()).reset_index()

total_births_female = pd.DataFrame(baby_names[baby_names["Sex"]=="F"].groupby(["Year"])["Frequency"].sum()).reset_index()

#graph customization and plotting
def plotter_3(X,Y1,Y2,Y3,label_Y1,label_Y2,label_Y3,Xlabel,Ylabel,Title,x_min,x_max,y_min,y_max):
    plt.plot(X,Y1,label = label_Y1)
    plt.plot(X,Y2,label = label_Y2)
    plt.plot(X,Y3,label = label_Y3)
    plt.ylabel(Ylabel)
    plt.xlabel(Xlabel)
    plt.title(Title)
    plt.legend(loc="right", bbox_to_anchor=[1.3, 0.5],
               ncol=1, shadow=False, fancybox=True)
    plt.axis([x_min,x_max,y_min,y_max])
    return(plt.show())

#calling the function
plotter_3(X = total_births["Year"],Y1 = total_births["Frequency"], Y2 = total_births_male["Frequency"],
        Y3 =total_births_female["Frequency"],label_Y1 = "Overall",label_Y2 = "Male",label_Y3 = "Female",
        Xlabel = "Year",Ylabel = "Population",Title = "Population Over Time",x_min = 1910,x_max = 2014,
        y_min = 0,y_max = 4500000 )
```



In [19]:

```
#the population trend over time is shown above. Both Male and Female followed the same trend over time but the number
#of female names has been higher initially and male has been higher after around 1940
```

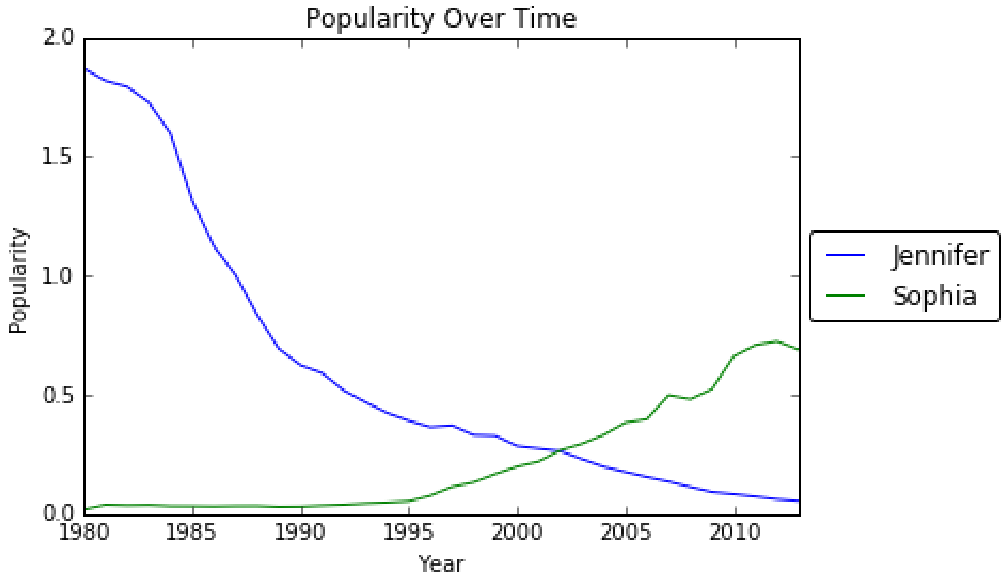
```
In [20]: unique_names_male = pd.DataFrame(baby_names[baby_names["Sex"]=="M"].groupby(["Year"])["Name"].nunique
         ()).reset_index()
         unique_names_female = pd.DataFrame(baby_names[baby_names["Sex"]=="F"].groupby(["Year"])["Name"].nuniq
         ue()).reset_index()

         #graph customization and plotting
         def plotter_2(X,Y1,Y2,label_Y1,label_Y2,Xlabel,Ylabel,Title,x_min,x_max,y_min,y_max):
             plt.plot(X,Y1,label = label_Y1)
             plt.plot(X,Y2,label = label_Y2)
             plt.ylabel(Ylabel)
             plt.xlabel(Xlabel)
             plt.title(Title)
             plt.legend(loc="right", bbox_to_anchor=[1.3, 0.5],
                        ncol=1, shadow=False, fancybox=True)
             plt.axis([x_min,x_max,y_min,y_max])
             return(plt.show())

         #calling the function
         plotter_2(X = unique_names_male["Year"],Y1 = unique_names_male["Name"], Y2 = unique_names_female["Nam
         e"],
                 label_Y1 = "Male",label_Y2 = "Female",
                 Xlabel = "Year",Ylabel = "Number of Unique Names",Title = "Unique Names Over Time",x_min = 19
         10,x_max = 2014,
                 y_min = 0,y_max = 6500 )
```



```
In [21]: #based on the above analysis we can se that eventhough the number of men is higher after 1940, number
         of unique names
         #more for woman compared to men. Initially the diversity in names was almost the same and very low. T
         he diversity kept
         #increasing with time for both men and women
```

In [22]:

```
#next analyse how the popularity of Jennifer (most decrease in popularity) and Sophia (most increase
in popularity)
#varied from 1980 - 2013
DataForTimePeriod = baby_names[baby_names["Year"]>=1980]
Overall = pd.DataFrame(DataForTimePeriod.groupby("Year")["Frequency"].sum().reset_index())
Jennifer = DataForTimePeriod[DataForTimePeriod["Name"]=="Jennifer"]
Sophia = DataForTimePeriod[DataForTimePeriod["Name"]=="Sophia"]
Jennifer_freq = pd.DataFrame(Jennifer.groupby("Year")["Frequency"].sum().reset_index())
Sophia_freq = pd.DataFrame(Sophia.groupby("Year")["Frequency"].sum().reset_index())

#graph customization and plotting
plotter_2(X = Jennifer_freq["Year"],Y1 = Jennifer_freq["Frequency"]/Overall["Frequency"]*100,
          Y2 = Sophia_freq["Frequency"]/Overall["Frequency"]*100,
        label_Y1 = "Jennifer",label_Y2 = "Sophia",
        Xlabel = "Year",Ylabel = "Popularity",Title = "Popularity Over Time",x_min = 1980,x_max = 201
3,
        y_min = 0,y_max = 2 )
```



In [23]:

```
#from the graph we can see that Jennifer popularity gradually declined where as Sophia's popularity i
ncreased
#dramatically after 1995
```

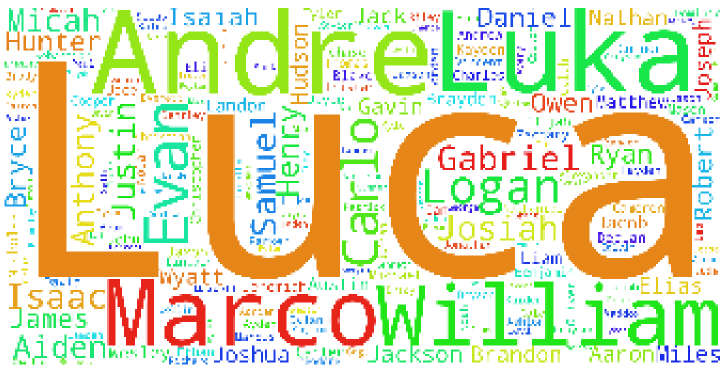
In [24]:

```
from wordcloud import WordCloud
```

In [25]:

```
def word_cloud(arr,year,gender):
    DataForTimePeriod = arr[arr["Year"]==year]
    top_ = DataForTimePeriod[DataForTimePeriod["Sex"]==gender].reset_index()

    #creating a text with all the names
    top_words = ""
    for ix in range(len(top_)):
        top_words += top_["Name"][ix] + " "

    #generating the wordcloud
    wordcloud = WordCloud(background_color='white',width=3000,height=1500).generate(top_words)
    plt.figure()
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.show()

word_cloud(baby_names,2013,"M")
```

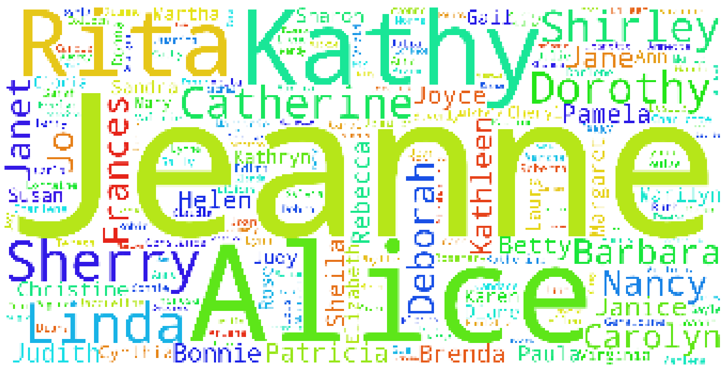
```
/Users/Avinash/anaconda/lib/python3.5/site-packages/PIL/ImageDraw.py:104: UserWarning: setfont() is
deprecated. Please set the attribute directly instead.
  "Please set the attribute directly instead.")
```



In [26]:

```
word_cloud(baby_names,1950,"F")
```

```
/Users/Avinash/anaconda/lib/python3.5/site-packages/PIL/ImageDraw.py:104: UserWarning: setfont() is
deprecated. Please set the attribute directly instead.
  "Please set the attribute directly instead.")
```

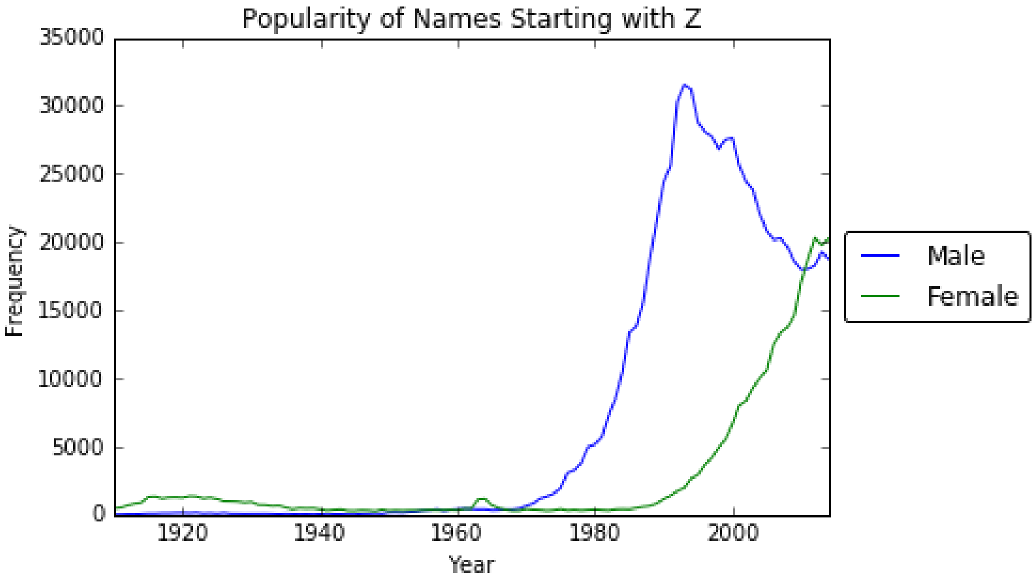


In [27]:

```
#both the above word cloud give us an idea about the most popular boy and girl names in a particular
year. For example
#in 1950, Sherry, Bonnie and Carol seem to be the most popular girl names
```

In [28]:

```
#lets analyse the time graph of words starting with a particular letter
def start_with(arr,letter):
    starts_with_M = pd.DataFrame(arr[arr["Sex"]=="M"])

    #filtering words starting with a letter
    starts_with_M = starts_with_M[starts_with_M['Name'].str.startswith(letter)]
    starts_with_M = pd.DataFrame(starts_with_M.groupby("Year")["Frequency"].sum().reset_index())

    starts_with_F = pd.DataFrame(arr[arr["Sex"]=="F"])

    #filtering words starting with a letter
    starts_with_F = starts_with_F[starts_with_F['Name'].str.startswith(letter)]
    starts_with_F = pd.DataFrame(starts_with_F.groupby("Year")["Frequency"].sum().reset_index())

    #calling the function
    plotter_2(X = starts_with_M["Year"],Y1 = starts_with_M["Frequency"],
              Y2 = starts_with_F["Frequency"],
            label_Y1 = "Male",label_Y2 = "Female",
            Xlabel = "Year",Ylabel = "Frequency",Title = "Popularity of Names Starting with " + lette
r ,
            x_min = 1910,x_max = 2014,
            y_min = 0,y_max = 35000 )

start_with(baby_names,"Z")
```



In [29]:

```
#the above chart gives an interesting insight about the popularity of beginning letter over time. We
can see that
#names starting with Z were very low in the initial years but aftr 1970, the number of people startin
g with Z increased
#exponentially. The number is higher in males compared to females.
```

In [ ]:

In [ ]: