Saul and Kalyan

October 7, 2016

```
In [74]: %reset
```

```
Once deleted, variables cannot be recovered. Proceed (y/[n])? y
```

```
In [6]: %matplotlib inline
        import re
        import nltk
        import pandas as pd
```

```
In [7]: en_corpus = open("../data/en.txt").read()
        es_corpus = open("../data/es.txt").read()
```

```
In [8]: def pre_process(corpus, clean = ’no’, char = ’no’):
            decode_corpus = corpus.decode(’utf-8’)
            token_corpus = nltk.word_tokenize(decode_corpus)

            if char == ’yes’:
                token_corpus = []
                for words in decode_corpus:
                    for j in words:
                        token_corpus.append(j)


            if clean == ’yes’:
                no_number_corpus = map(lambda x : x.translate(’0123456789’) , token_corpus)
                lower_corpus = map(lambda x : x.lower(), no_number_corpus)
                token_corpus = map(lambda x : re.sub(r’[^\w\s]’,’’,x), lower_corpus)

            freq_corpus = nltk.FreqDist(token_corpus)

            sorted_corpus = sorted(freq_corpus.items(), key=lambda x: x[1], reverse=True)
            no_space_corpus =  filter(lambda x : len(x[0]) > 0, sorted_corpus)

            headers = [’word’, ’frequency’]
            df = pd.DataFrame(no_space_corpus, columns=headers)
            df[’rank’] = df.index + 1

            return df
```

Questions Covered Below

- Question1: Read Data
- Question2: Read Corpus and tokenize
- Question3: Check Zipf Law, plot

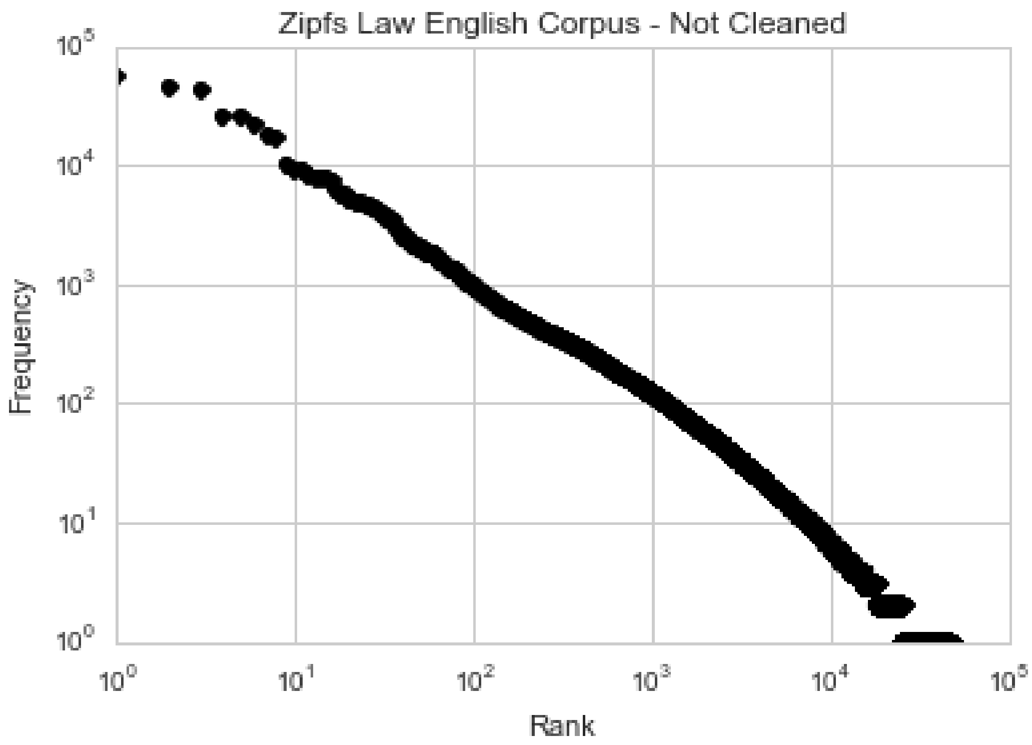
- Question4: Compute proportionality constant, deviation, average.
- Question5: Add more preprocssing steps like lowercase, punctuation and numbers
- Question6: Char Level Zipfs Law

```
In [9]: en_df = pre_process(en_corpus)
        es_df = pre_process(es_corpus)
        en_clean_df = pre_process(en_corpus, clean = 'yes')
        es_clean_df = pre_process(es_corpus, clean = 'yes')
        en_char_df = pre_process(en_corpus, char = 'yes')
        es_char_df = pre_process(es_corpus, char = 'yes')
```

Zipf Law f(x) = c * (rank + b) ˆ a

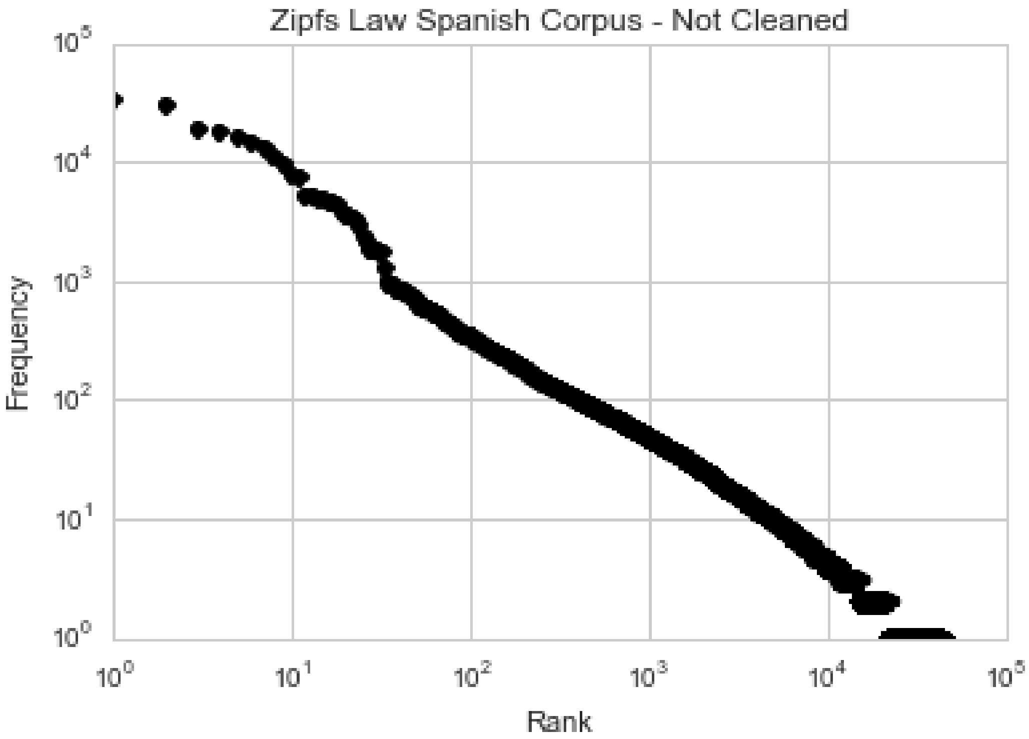
```
In [24]: plt.title('Zipfs Law English Corpus - Not Cleaned')
         plt.xlabel('Rank')
         plt.ylabel('Frequency')
         plt.yscale('log')
         plt.xscale('log')
         plt.plot(en_df['rank'], en_df['frequency'], 'ko')
```

```
Out[24]: [<matplotlib.lines.Line2D at 0x11cfb0410>]
```

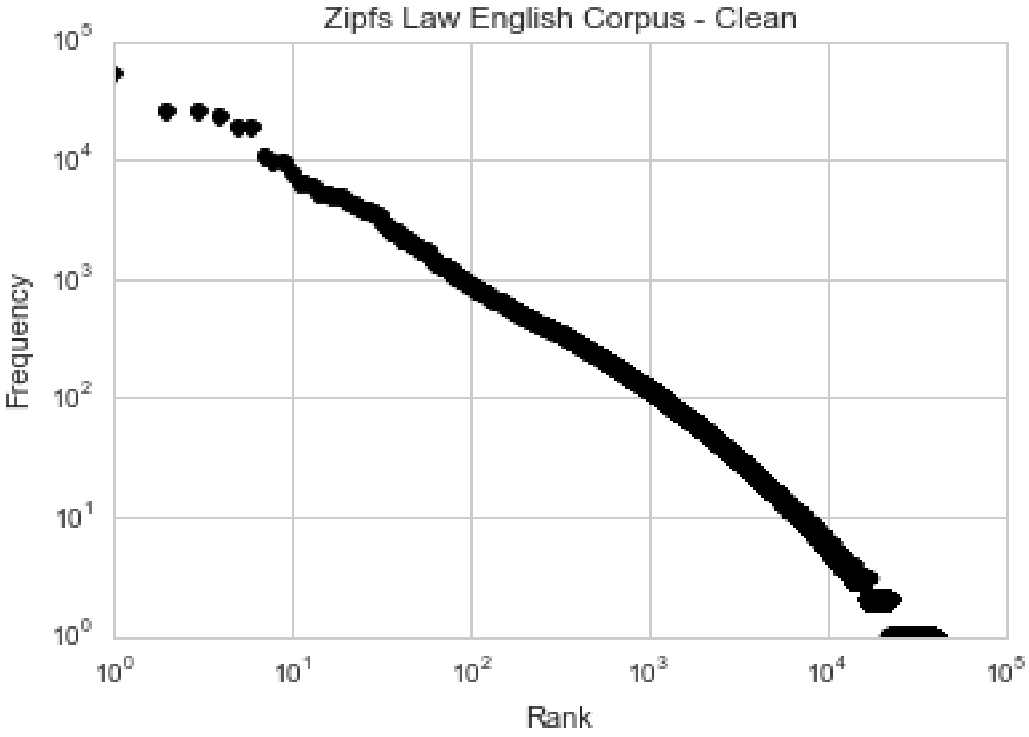


```
In [11]: plt.title('Zipfs Law Spanish Corpus - Not Cleaned')
         plt.xlabel('Rank')
         plt.ylabel('Frequency')
         plt.yscale('log')
         plt.xscale('log')
         plt.plot(es_df['rank'], es_df['frequency'], 'ko')
```

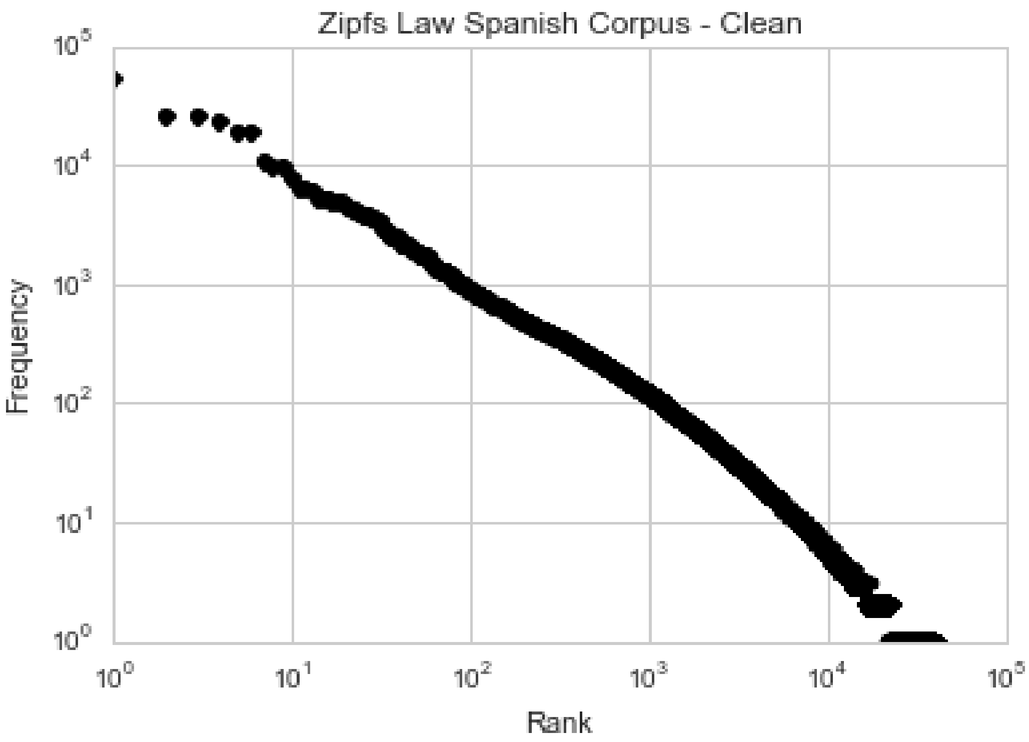
```
Out[11]: [<matplotlib.lines.Line2D at 0x11aaebe50>]
```



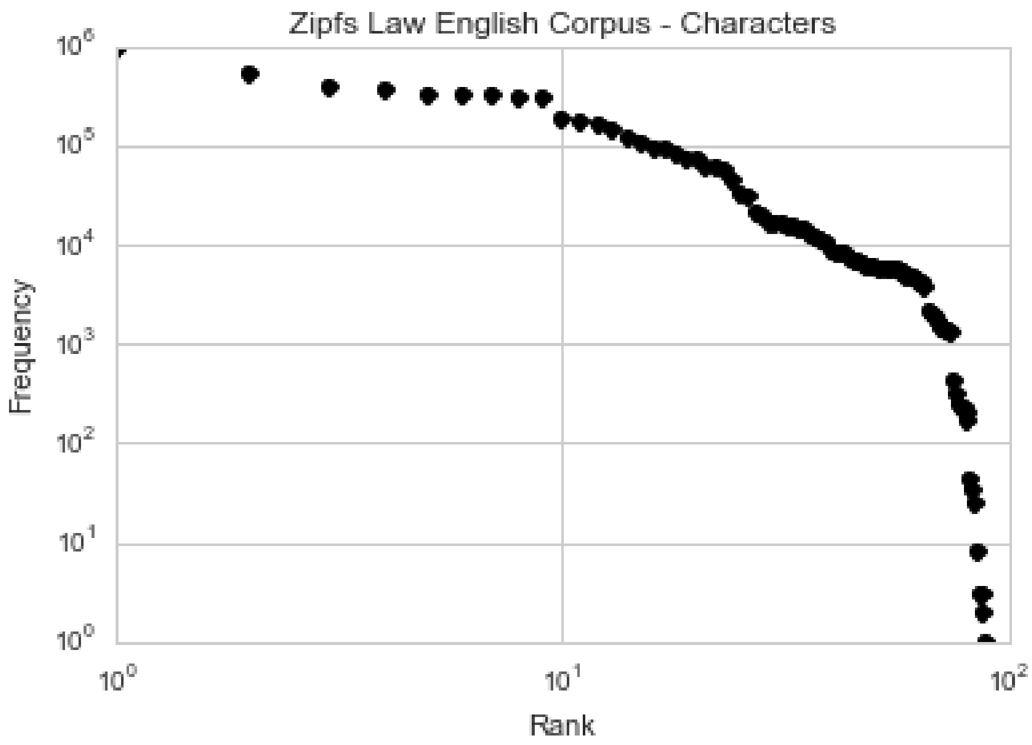
```
In [12]: plt.title('Zipfs Law English Corpus - Clean')
         plt.xlabel('Rank')
         plt.ylabel('Frequency')
         plt.yscale('log')
         plt.xscale('log')
         plt.plot(en_clean_df['rank'], en_clean_df['frequency'], 'ko')

Out[12]: [<matplotlib.lines.Line2D at 0x11afaf710>]
```



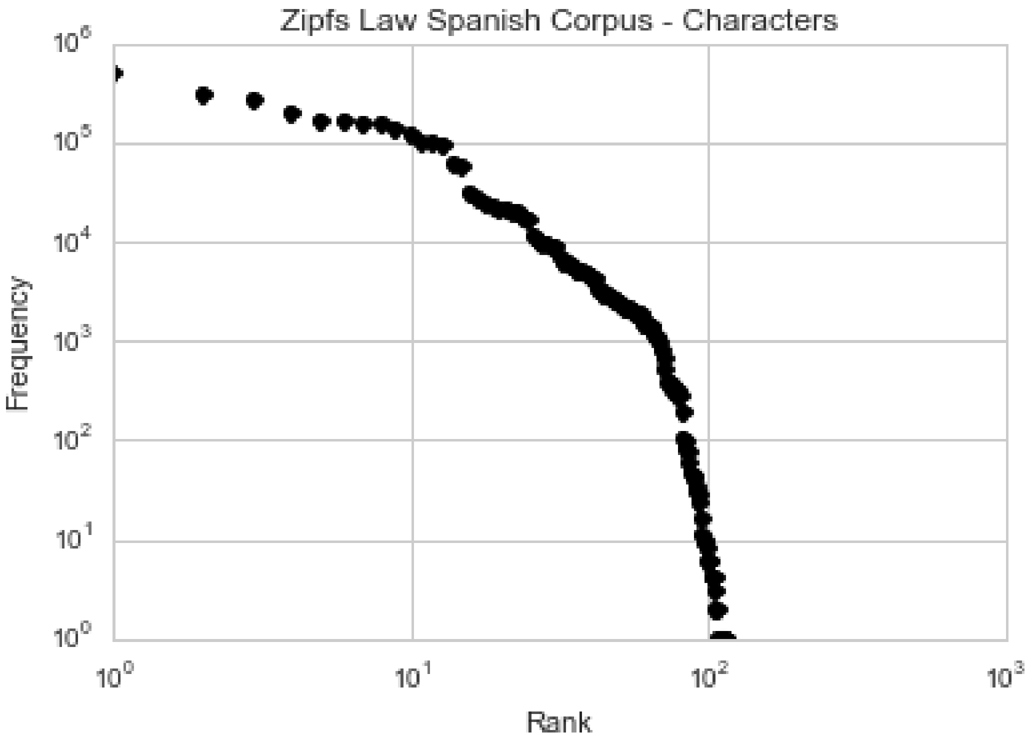
```
In [13]: plt.title('Zipfs Law Spanish Corpus - Clean')
         plt.xlabel('Rank')
         plt.ylabel('Frequency')
         plt.yscale('log')
         plt.xscale('log')
         plt.plot(en_clean_df['rank'], en_clean_df['frequency'], 'ko')

Out[13]: [<matplotlib.lines.Line2D at 0x11b226c50>]
```



```
In [14]: plt.title('Zipfs Law English Corpus - Characters')
         plt.xlabel('Rank')
         plt.ylabel('Frequency')
         plt.yscale('log')
         plt.xscale('log')
         plt.plot(en_char_df['rank'], en_char_df['frequency'], 'ko')

Out[14]: [<matplotlib.lines.Line2D at 0x11b98f3d0>]
```



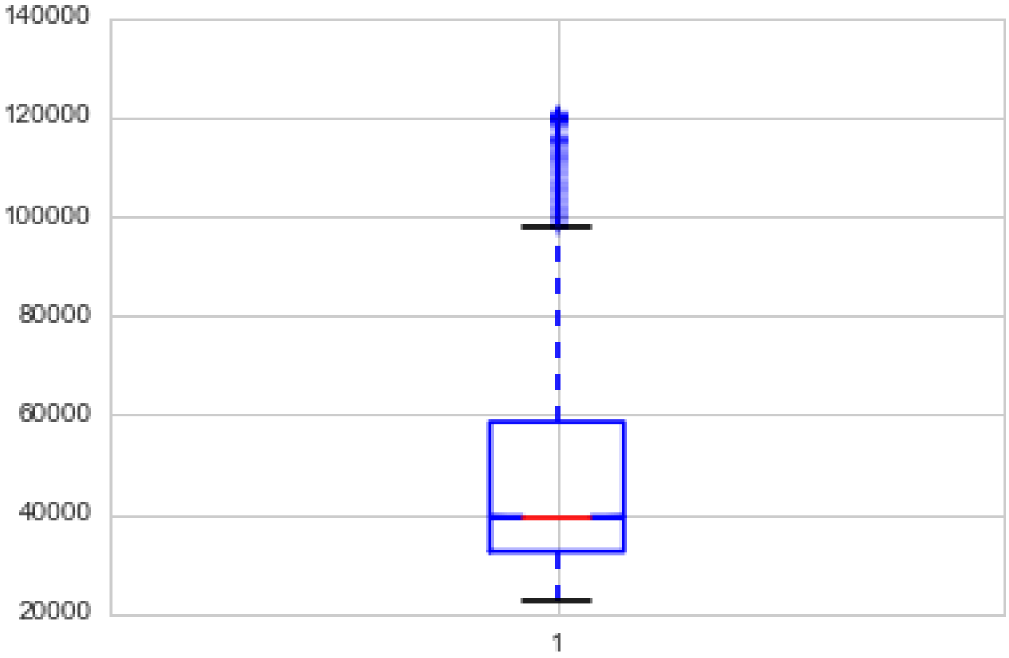
```
In [15]: plt.title('Zipfs Law Spanish Corpus - Characters')
         plt.xlabel('Rank')
         plt.ylabel('Frequency')
         plt.yscale('log')
         plt.xscale('log')
         plt.plot(es_char_df['rank'], es_char_df['frequency'], 'ko')

Out[15]: [<matplotlib.lines.Line2D at 0x11ced3210>]
```



```
In [27]: en_clean_df['K'] = en_clean_df['frequency']*en_clean_df['rank']
         es_clean_df['K'] = es_clean_df['frequency']*es_clean_df['rank']
```

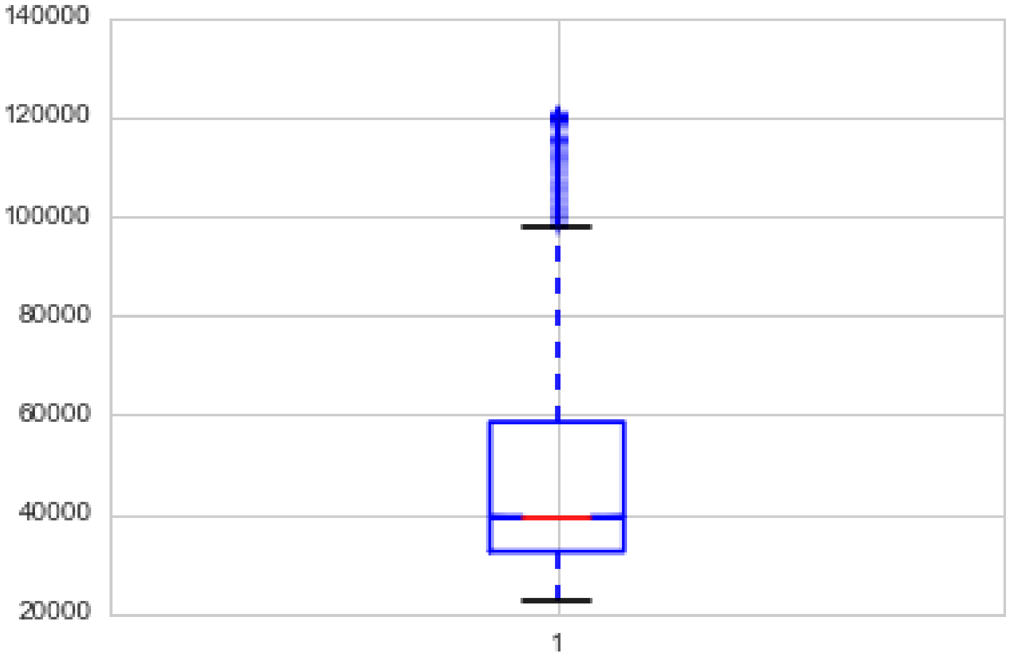
```
In [38]: plt.boxplot(en_clean_df['K'],'-')
         print "Boxplot English Corpus K"
```

```
Boxplot English Corpus K
```



```
In [37]: plt.boxplot(en_clean_df['K'],'-')
         print "Boxplot Spanish Corpus K"
```

```
Boxplot Spanish Corpus K
```



```
In [39]: en_clean_df[’K’].describe()
```

```
Out[39]: count     39930.000000
         mean      49018.568695
         std       24328.985981
         min       22497.000000
         25%       32479.250000
         50%       39406.000000
         75%       58563.750000
         max      121500.000000
         Name: K, dtype: float64
```

```
In [40]: es_clean_df[’K’].describe()
```

```
Out[40]: count    38376.000000
         mean     35337.260658
         std       8185.550050
         min      20206.000000
         25%      29282.000000
         50%      35091.500000
         75%      40854.000000
         max      70532.000000
         Name: K, dtype: float64
```

```
In [50]: en_char_df[’K’] = en_char_df[’frequency’]*en_char_df[’rank’]
         es_char_df[’K’] = es_char_df[’frequency’]*es_char_df[’rank’]
         print en_char_df[’K’].describe(), "\n#############\n", es_char_df[’K’].describe()
```

```
count         89.000000
mean      621047.224719
std       656906.877798
min           89.000000
25%       136077.000000
50%       311490.000000
75%      1038168.000000
max      2637441.000000
Name: K, dtype: float64
#############
count        115.000000
mean      222734.373913
std       318760.200905
min          109.000000
25%         5662.000000
50%       107502.000000
75%       268026.000000
max      1185456.000000
Name: K, dtype: float64
```