HW1 - Reproduce Power Laws

In class, we talked about how various papers claim that Zipf's law is a general property of many discrete distributions. Wikipedia Mathworld

The original paper by Belevitch examined this question theoretically. However, if this is true, it should be easy to reproduce experimentally, namely by picking various discrete distributions, computing word frequency by rank, and plotting the result.

For working with the worksheet, remember that you can insert additional cells, both to add text and explanations, and to add additional code cells. You can change existing cells if you like; they are just there to help you get started.

```
In [1]: from pylab import *
        from collections import Counter
```

Start by generating a random sample. Here is an example of a function that generates a uniform sample. Obviously, this particular choice of distribution will not reproduce Zipf's law, so you need to modify this to try to come up with distributions that will reproduce Zipf's law.

```
In [2]: def generate_sample(nsamples,vocabulary_size):
            return array(rand(nsamples)*vocabulary_size,'i')
```

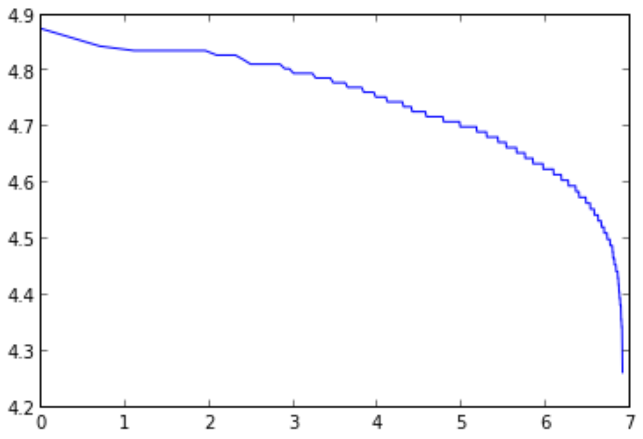
```
In [3]: data = generate_sample(100000,1000)
```

Compute a histogram.

```
In [4]: histogram = Counter(data)
```

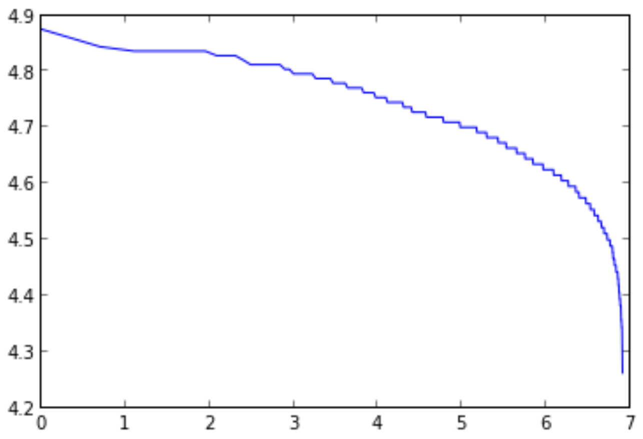
Compute frequency by rank.

```
In [5]: frequencies = sort(histogram.values())[::-1]
In [6]: frequencies = sort(histogram.values())[::-1]
        plot(log(1+arange(len(frequencies))),log(frequencies))

Out[6]: [<matplotlib.lines.Line2D at 0x23c1b50>]
```



Plot the result.

```
In [7]: plot(log(1+arange(len(frequencies))),log(frequencies))

Out[7]: [<matplotlib.lines.Line2D at 0x26bb850>]
```

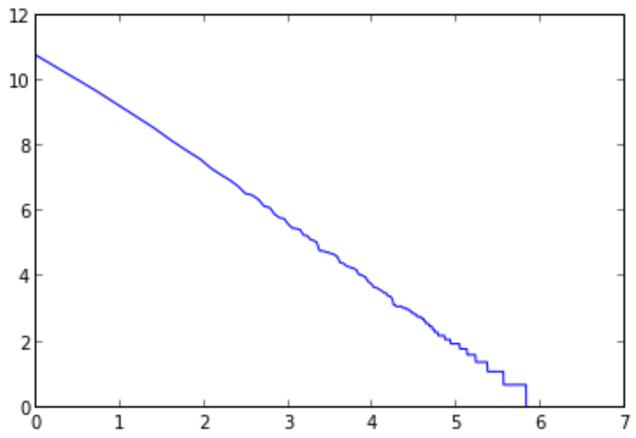


Now wrap this up as a function.

```
In [8]: def frequency_by_rank_plot(samples):
            histogram = Counter(samples)
            frequencies = sort(histogram.values())[::-1]
            plot(log(1+arange(len(frequencies))),log(frequencies))
```

Now define multiple distributions and plot their frequency ranks. Which ones give rise to power laws?

In [10]:
```
def distribution(n,q):
    return array(rand(n)**-1.1,'i')
data = distribution(100000,1000)
samples = Counter(data)
frequency_by_rank_plot(data)
```



In [ ]:

In [ ]:

In [ ]: