In real world, no corpus will be balanced. We need to count the occurrences of relevant words in the dataset to get some statistical information. We need to know the frequency distribution of different words. Word frequencies refer to the number of word tokens that are instances of a word type. We can perform word counts over corpora with the R tau package.

Zipf's law:

Zipf's law is an interesting phenomenon that can be applied universally in many contexts, such as social sciences, cognitive sciences, and linguistics. When we consider a variety of datasets, there will be an uneven distribution of words. Zipf's law says that the frequency of a word, *f (w)*, appears as a nonlinearly decreasing function of the rank of the word, *r (w)*, in a corpus. This law is a power law: the frequency is a function of the negative power of rank. *C* is a constant that is determined by the particulars of the corpus; it's the frequency of the most frequent word:

$$f(w) = \frac{C}{r(w)^a}$$

$$f(w) = \frac{C}{r(w)^a}$$

Given a collection of words, we can estimate the frequency of each unique word, which is nothing but the number of times the word occurs in the collection.

If we sort the words in descending order of their frequency of occurrence in the collection, and compute their rank, the product of their frequency and associated rank reveals a very interesting pattern.

• **N**: Sample size or corpus size
• **V**: Vocabulary size, count of distinct type in the corpus
• **Vm**: Count of hapax terms, types that occur just once in a corpus

Let us consider a small sample **S: a a a a b b b c c d d:**

1. Here, N= 11, V = 4, Vm = 0.

2. Load Brown and Dickens frequency data:

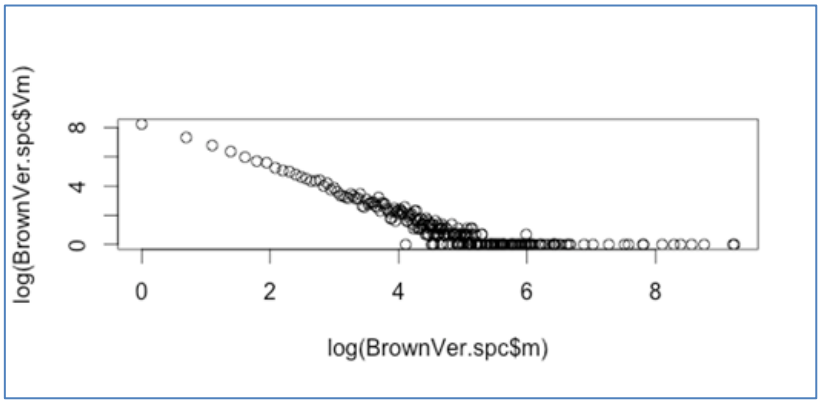
```
library(zipfR)
data(Dickens.spc)
data(BrownVer.spc)
```

3. Check sample size and vocabulary and hapax counts:

```
N(BrownVer.spc) # 166262
V(BrownVer.spc) # 10007
Vm(BrownVer.spc,1) # 3787
N(Dickens.spc) # 2817208
V(Dickens.spc) # 41116
Vm(Dickens.spc,1) # 14220
```

4. Zipf rank-frequency plot:

plot(log(BrownVer.spc$m),log(BrownVer.spc$Vm))

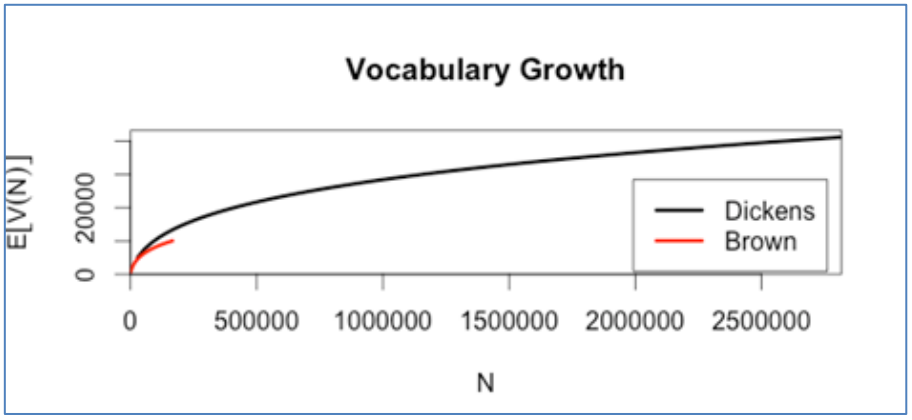


5. Compute binomially interpolated growth curves:

*di.vgc <- vgc.interp(Dickens.spc,(1:100)*28170)*
*br.vgc <- vgc.interp(BrownVer.spc,(1:100)*1662)*

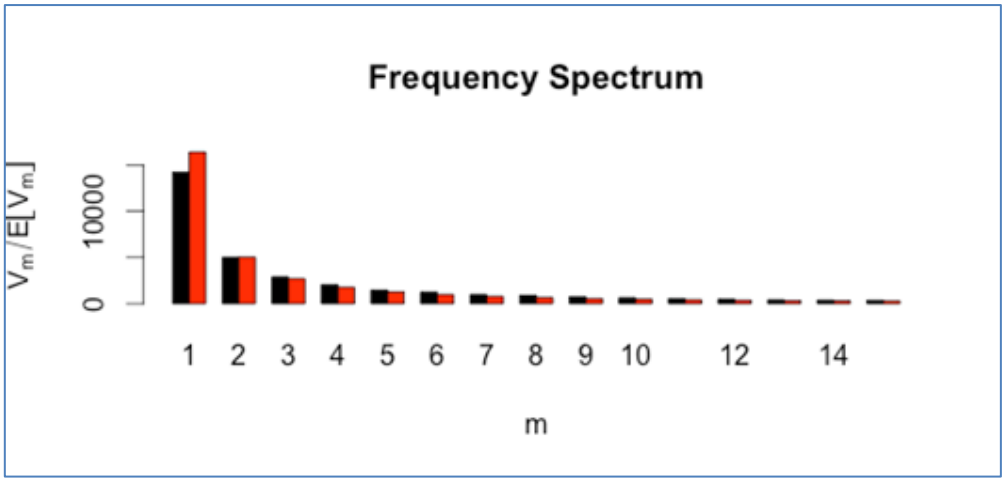
6. Plot vocabulary growth:

plot(di.vgc,br.vgc,legend=c("Dickens","Brown"))



7. Compute Zipf-Mandelbrot model from Dickens data:

zm <- lnre("zm",Dickens.spc)
plot observed and expected spectrum
zm.spc <- lnre.spc(zm,N(Dickens.spc))



Let there be a word w which has the rank r' in a document and the probability of this word to be at rank r' be defined as P(r'). The probability P(r') can be expressed as the function of frequency of occurrence of the words as follows:

P(r') = Freq(r')/N,
Where N is the sample size and Freq(r') is the frequency of occurrence of r' in the corpus.
As per Zipf's law, r' * P(r') = K, where K is a constant. The value of K is assumed to be close to 0.1.

Heaps' law

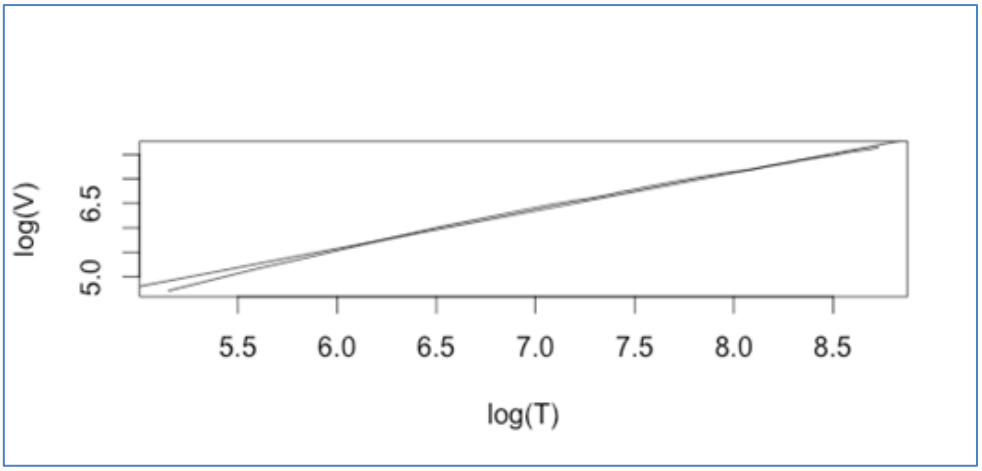
Heaps' law is also known as Herdan's Law. This law was discovered by Gustav Herdan, but the law is sometimes attributed to Harold Heaps. It is an empirical law which describes the relationship between type and tokens in linguistics. In simpler terms, Heaps' law defines the relation between the count of distinct words in document and the length of the specified document.

The relation can be expressed as:
Vr(n) = C* nb

Here, Vr is the count of distinct words in document and n is the size of the document. C and b are parameters defined empirically.

The similarity between Heaps' Law and Zipf's law is attributed to the fact that typetoken relation is derivable from type distribution:

```
library(tm)
data("acq")
termdoc <- DocumentTermMatrix(acq)
Heaps_plot(termdoc)
```


log(V)
5.0
6.5
5.5
6.0
6.5
7.0
7.5
8.0
8.5
log(T)