Zipf's Law

By -
Aaruni Kaushik
Cyril George
Joseph Puthumana
Nandita Raghunath

Introduction

- Zipf's Law is named after George Kingley Zipf
- It states that given some corpus of natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table.
- The same relationship occurs in many other rankings unrelated to language, such as the population ranks of cities in various countries, corporation sizes, income rankings, ranks of number of people watching the same TV channel, and so on.

List of Books Considered

- 1984 by George Orwell
- Animal Farm by George Orwell
- The Picture of Dorian Gray by Oscar Wilde
- Robinson Crusoe by Daniel Defoe
- Teens Wander Around a House generated by username "DariusK"

Methodology Followed

- Books were converted to plain text format for ease of "reading".
- Data was mined from the books using C++.
- Data was tabulated for each book.
- Goodness of fit was approximated by finding the value of R^2 using method of least squares.

```cpp
#include<iostream>  //basic IO functions
#include<fstream>   //file stream functions
#include<string.h>  //string functions

using namespace std;  //to avoid writing std::cin

int main()
{
  int wcount=0,flag;
  char inword[256],words[30380][256];
  unsigned long long int ar[30380];
  for(int i=0;i<30380;i++)
  {
    ar[i]=0;    //frequency of words
  }

  ifstream in("/path/to/book.txt");

  if(!in)
  {
    cout<<"Book not found. Please verify path\n";
    return 1; //errored exit
  }

  while(in>>inword)
  {
    for(int i=0;i<int(strlen(inword));i++)    //normalising words, stripping capitalisation and punctuations
    {
      inword[i]=tolower(inword[i]);
      if(inword[i]<'a' || inword[i]>'z')
      {
        if (i>0)
        {
          inword[i]='\0';
```

```
35          }
36          else
37          {
38            for(int j=0; ; j++)
39            {
40              if(inword[j]=='\0')
41                break;
42              inword[j]=inword[j+1];
43            }
44          }
45        }
46        inword[i]=tolower(inword[i]);
47      }                                   //all words now normalised
48
49      if(!strlen(inword))   //if string of size 0 then ignore
50        continue;
51      for(int i=0;i<wcount;i++)
52      {
53        flag=0;           //flag checks for unique words
54        if(!strcmp(inword,words[i]))
55        {
56          ar[i]++;
57          flag=1;
58          break;
59        }
60      }
61      if(!flag)
62      {
63        strcpy(words[wcount+1],inword);
64        ar[wcount+1]++;
65        wcount++;
66      }
67    }
68
69    for(int i=1;i<wcount;i++)
70    {
71      cout<<words[i]<<'\t'<<ar[i]<<endl;
72    }
73
74    return 0;
75  }
```

Fun Facts

- The word "the" is the most frequent occurring word in all the books.
- There are totally 30019 words in Animal Farm, and only 3897 unique words.
- 1984 has 94965 words in total. The word "the" accounts for 6501 (6.8%) of those words.
- The book "Teens Wander Around a House" has been "written" by a computer as a part of NaNoGenMo.

Data Snapshots

Word	Rank	Frequency
the	1	2207
and	2	963
of	3	902
to	4	809
was	5	633
a	6	620
in	7	541
had	8	528
that	9	447
it	10	395
they	11	342
he	12	322
were	13	290
his	14	272
for	15	259
animals	16	248
on	17	247
with	18	231
at	19	211
their	20	197

Animal Farm

Word	Rank	Frequency
the	1	6501
of	2	3490
a	3	2549
and	4	2423
to	5	2335
was	6	2310
he	7	1957
it	8	1914
in	9	1854
that	10	1490
had	11	1339
his	12	1086
you	13	1005
not	14	845
with	15	791
as	16	727
at	17	662
for	18	659
be	19	657
they	20	655

1984

Word	Rank	Frequency
the	1	5907
i	2	5091
and	3	4754
to	4	4263
of	5	3512
a	6	2272
my	7	2121
was	8	1969
in	9	1943
that	10	1850
it	11	1810
had	12	1553
as	13	1524
for	14	1292
me	15	1224
but	16	1082
with	17	1078
not	18	970
which	19	883
he	20	879

Robinson Crusoe

Data Snapshots

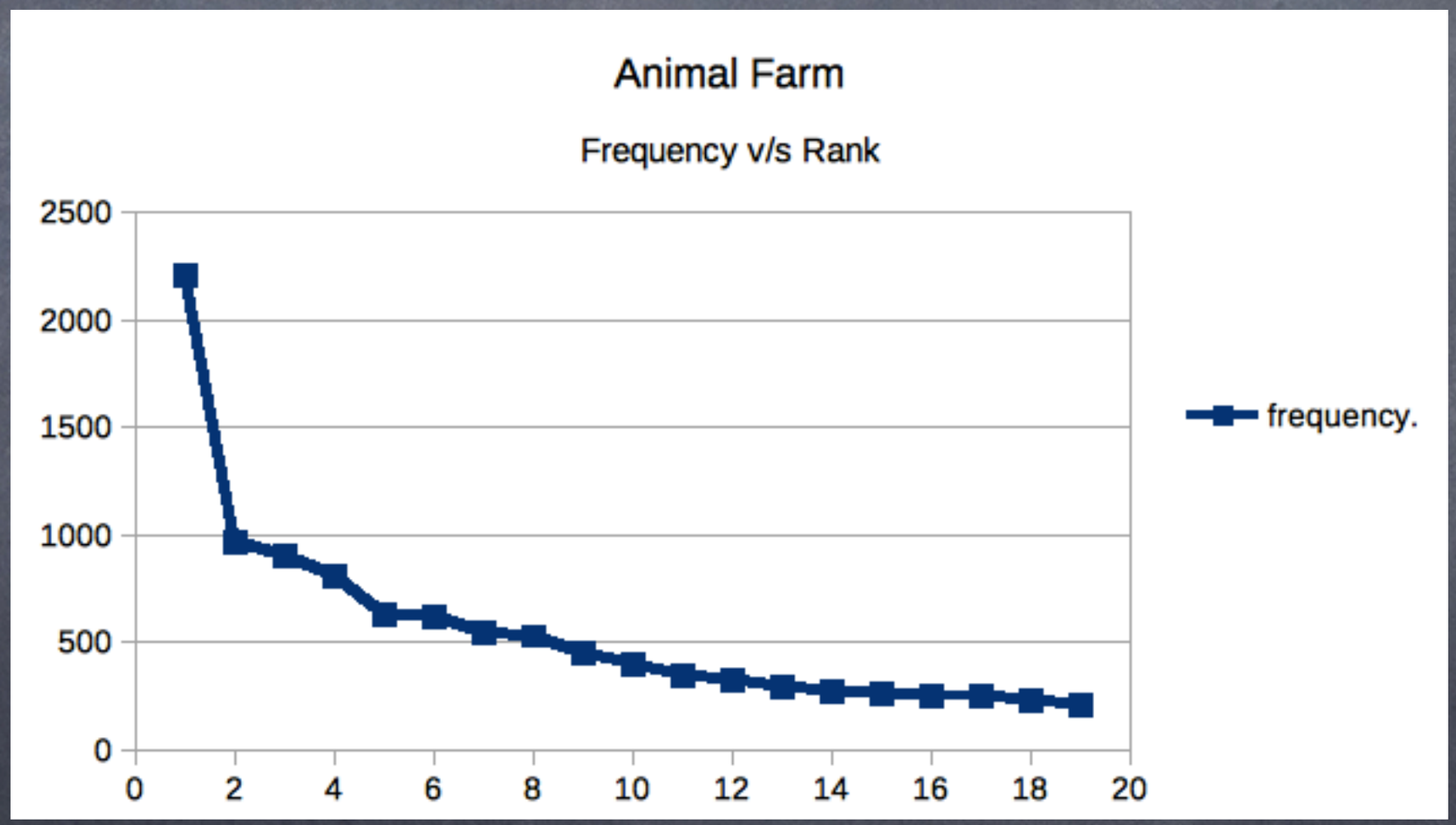
Word	Rank	Frequency
the	1	3763
and	2	2190
of	3	2174
to	4	2096
i	5	1682
a	6	1667
he	7	1533
you	8	1441
it	9	1341
that	10	1340
in	11	1206
was	12	1078
his	13	992
is	14	905
had	15	828
him	16	661
with	17	659
for	18	584
as	19	572
have	20	560

The Picture of Dorian Gray

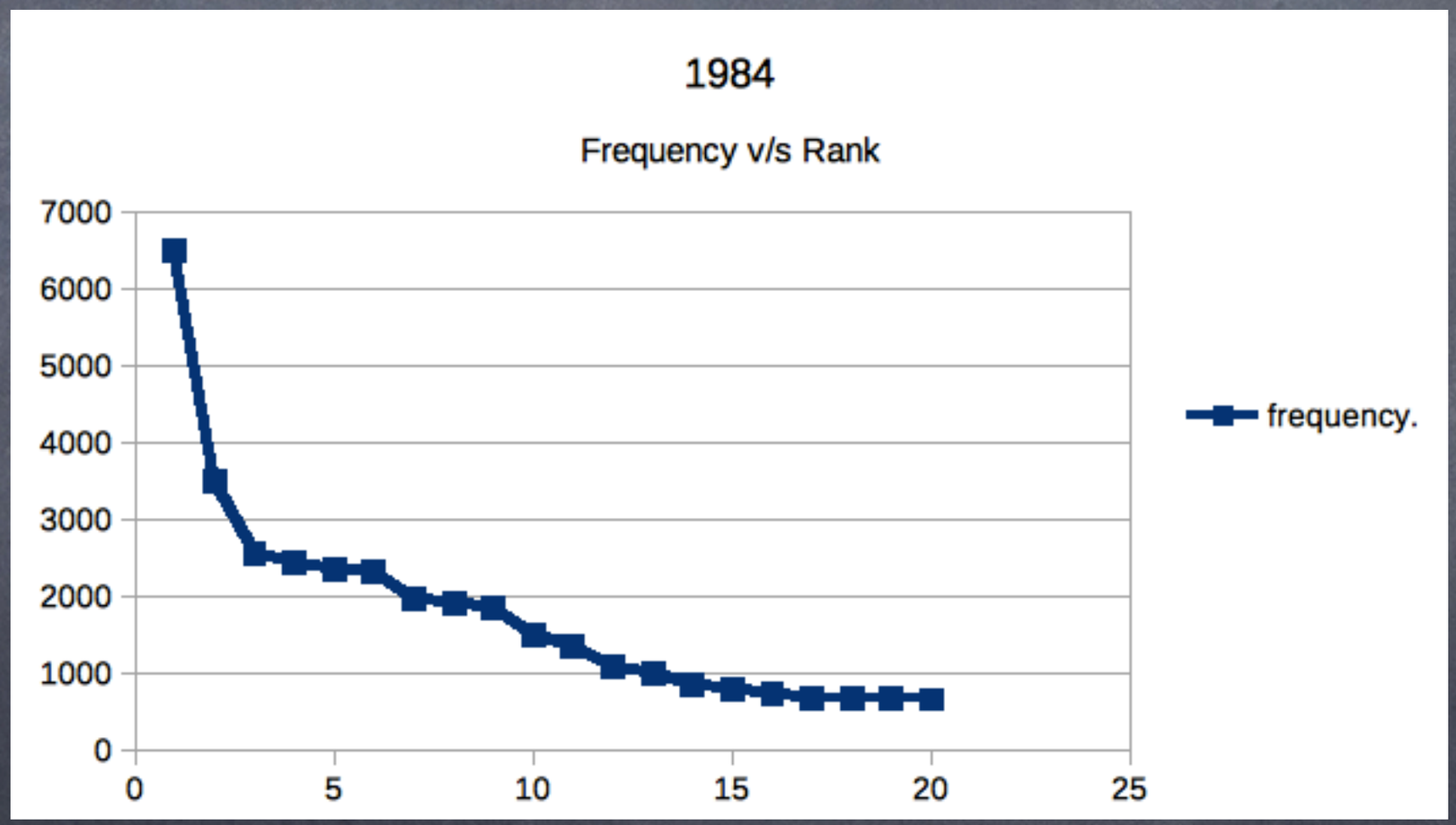
Word	Rank	Frequency
the	1	6458
she	2	4450
to	3	2730
her	4	2613
in	5	2523
was	6	2218
and	7	1937
it	8	1698
thought	9	1689
of	10	1653
they	11	1602
a	12	1496
vivianna	13	1187
philomena	14	1175
dita	15	1149
gale	16	1129
darby	17	1122
kiah	18	1098
i	19	1020
about	20	938

Teens Wander Around A House

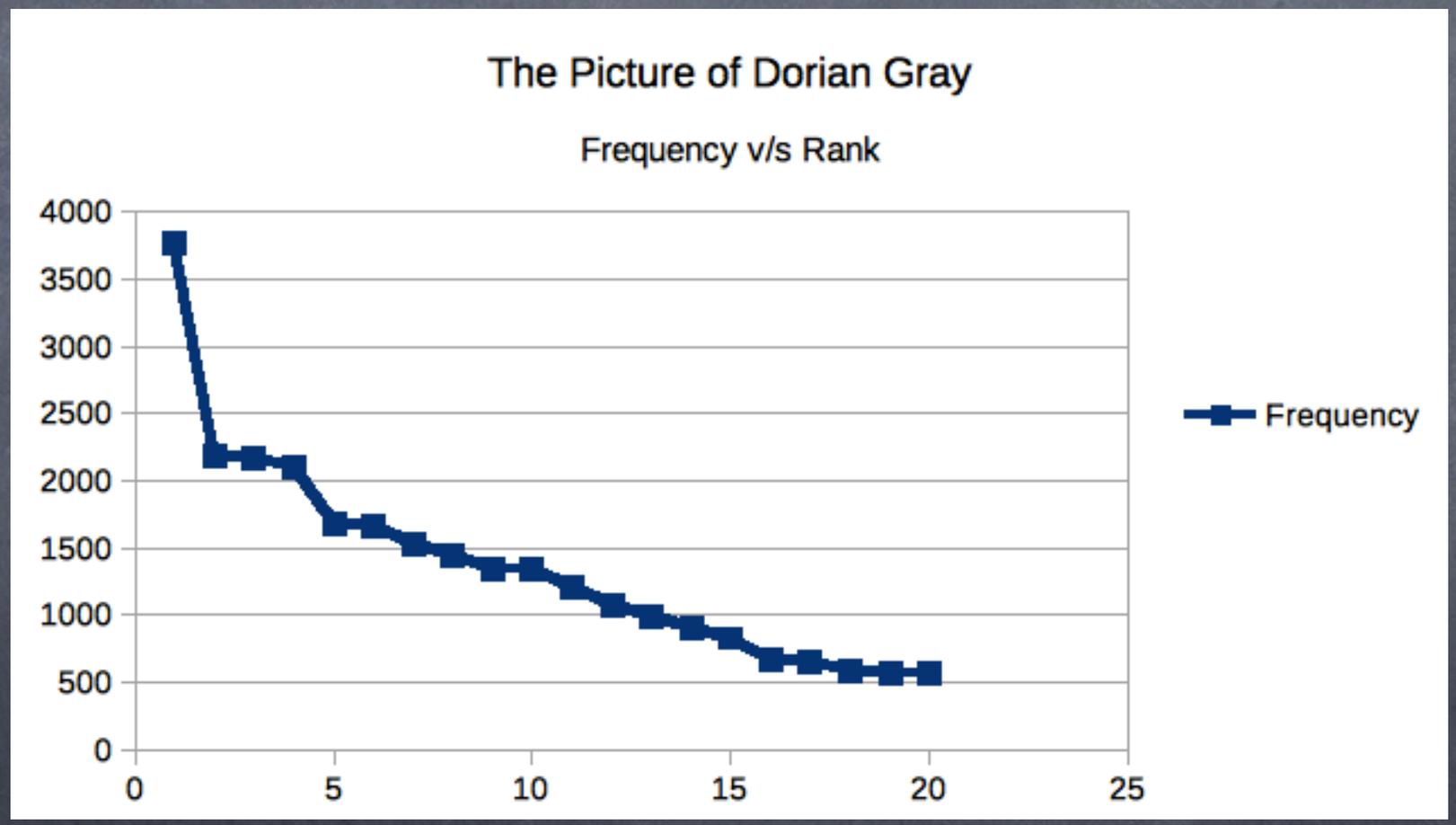
Data Representation



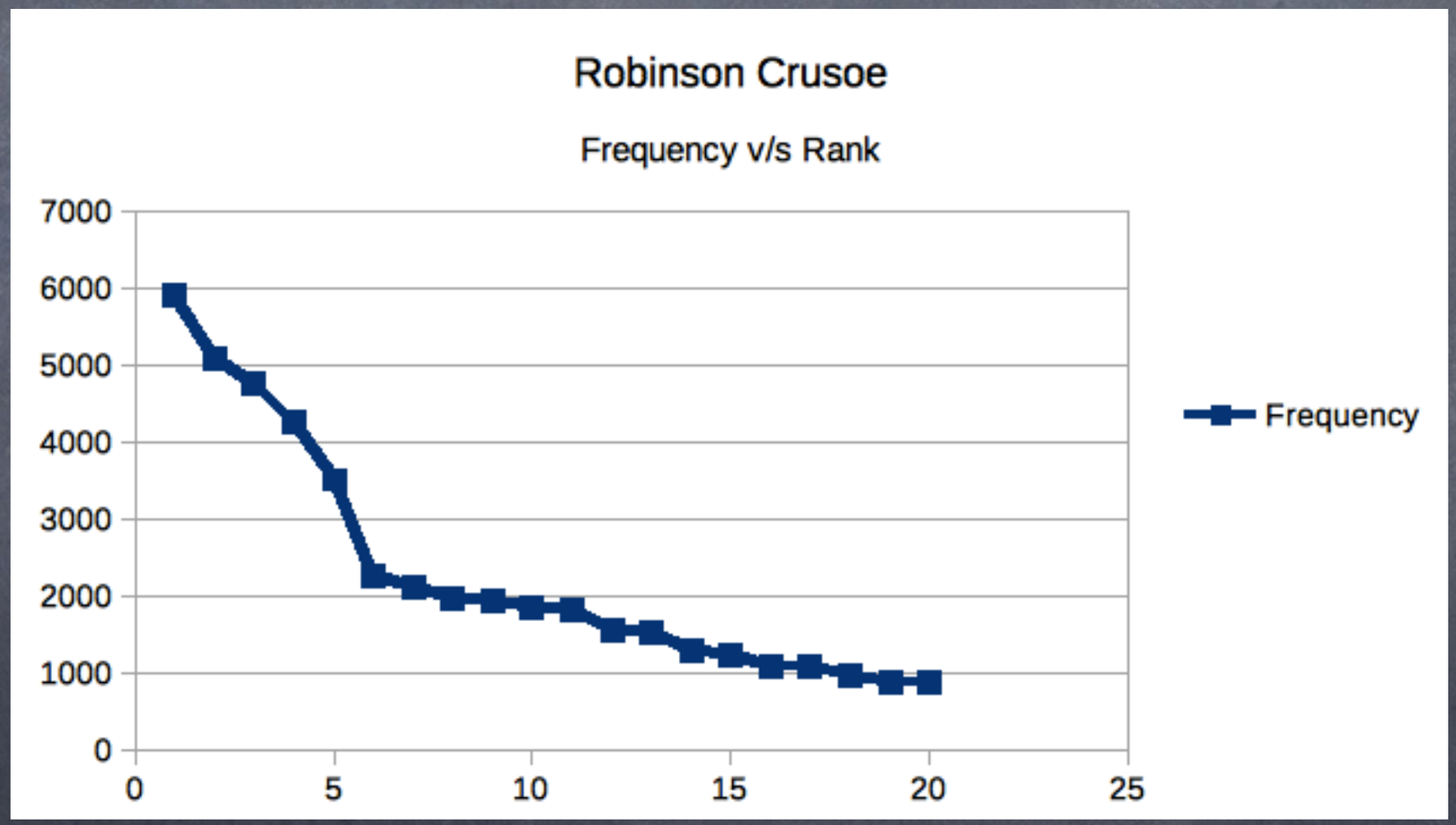
Data Representation



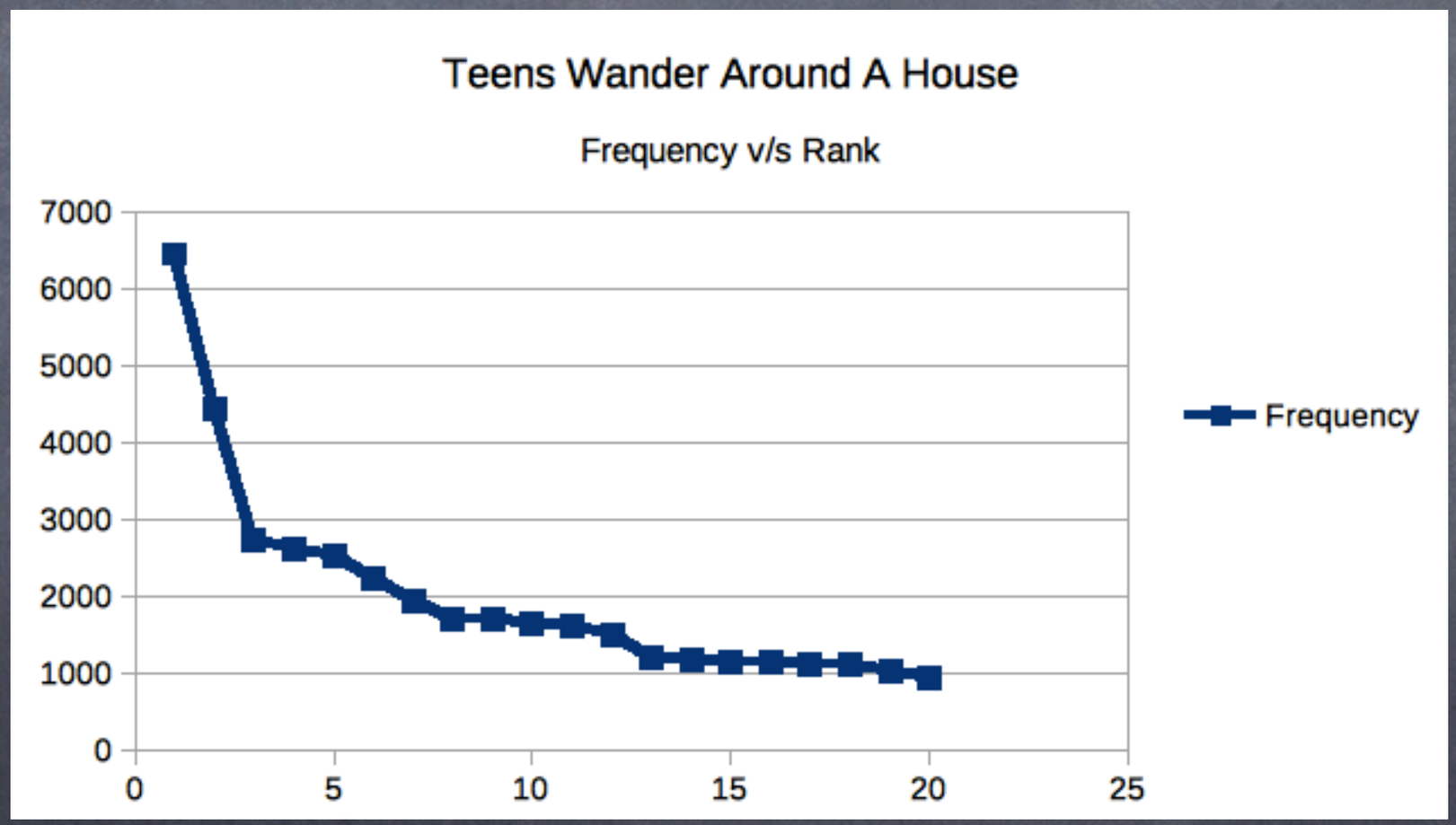
Data Representation



Data Representation



Data Representation



Analysis of Data

- Our aim is to estimate the value of R^2. We do so using the method of least squares.
- To that end, we require the expected values of frequency fractions, as given by the PMF of Zipfian Distribution :

$$f(x) = \frac{1}{x^s} \times \frac{1}{\sum_{1}^{n} i^{-s}}$$

Analysis of Data

- The parameter **s** is estimated empirically for each book. The value of the parameter for each book is listed below :

- Animal Farm : S = 0.89718
- 1984 : S = 0.88555
- The Picture Of Dorian Gray : S = 0.83075
- Robinson Crusoe : S = 0.83077
- Teens Wander Around A House : S = 0.82625

Analysis of Data

- The value of PMF for each rank of word for each book was evaluated en-masse using the help of the following computer program (C++).

```cpp
#include<iostream>  //for basic IO functions
#include<math.h>    //for exponential operator

using namespace std;    //to avoid writing std::cin

double pdf(double r,double n,double s);

int main()
{
  double n=2217,s;    //dummy value for n
  cin>>s;           //s is determined by a series of trial-and-error iterations
  for(int i=1;i<=n;i++)
  {
    cout<<pmf(i,n,s)<<endl;
  }
  return 0;
}

double pdf(double r, double n, double s)
{
  double sum=0,freq;
  for(int i=1;i<=n;i++)
  {
    sum+=1/pow(i,s);
  }
  freq=1/pow(r,s)/sum;    //the formal PDF of zipfian distribution
  return freq;
}
```

Analysis of Data

- Once the expected and actual values for frequency fractions are know, the value of R^2 is calculated as follows :

$$SS_{\text{res}} = \sum_i (y_i - f_i)^2$$

$$SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2,$$

$$R^2 \equiv 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$

Analysis Of Data

	A	B	C	D	E	F	G	H	I	J
1	Word	Rank	Frequency	Frequency Fraction	Zipfian Estimate	PMF	Residal Squares	Total Squares		
2	the	1	2207	0.073520103934175	S=0.897178	0.0735202	9.22864272821482E-15	0.005367539895108	0.011396448252561	Total SS
3	and	2	963	0.032079682867517		0.0394757	5.47010694239785E-05	0.001012708119482	0.000345051477346	Residual SS
4	of	3	902	0.030047636496885		0.0274375	6.8128125323727E-06	0.000887505402735	0.969722893510422	R2
5	to	4	809	0.026949598587561		0.0211959	3.31050474365039E-05	0.000712515767862		
6	was	5	633	0.021086645124754		0.0173503	1.39602748912754E-05	0.000433890463283		
7	a	6	620	0.020653586062161		0.0147322	3.50628128971502E-05	0.000416036730386		
8	in	7	541	0.018021919451014		0.0128293	2.69632967630527E-05	0.000315606305164		
9	had	8	528	0.017588860388421		0.0113808	3.85400137862778E-05	0.000300406986722		
10	that	9	447	0.014890569306106		0.0102396	2.16315154863414E-05	0.000214152835216		
11	it	10	395	0.013158333055731		0.00931599	1.47636001579271E-05	0.000166454519735		
12	they	11	342	0.011392784569773		0.00855249	8.06727324307902E-06	0.000124014437502		
13	he	12	322	0.01072653985809		0.00791023	7.93160121677422E-06	0.000109619481489		
14	were	13	290	0.009660548319398		0.0073621	5.28286467694218E-06	8.84341010539464E-05		
15	his	14	272	0.009060928078883		0.00688852	4.71935686119775E-06	7.75160590948851E-05		
16	for	15	259	0.00862786901629		0.00647506	4.63458666061815E-06	7.00780176998603E-05		
17	animals	16	248	0.008261434424864		0.00611078	4.62531445518819E-06	6.40772524960663E-05		
18	on	17	247	0.00822812218928		0.00578729	5.95766177622599E-06	6.35450448471058E-05		
19	with	18	231	0.007695126419934		0.00549799	4.82740844780058E-06	5.53315622345129E-05		
20	at	19	211	0.007028881708251		0.00523766	3.20847520811121E-06	4.58636965972928E-05		
21	their	20	197	0.006562510410074		0.00500209	2.43491185617432E-06	3.97644102500944E-05		
22	as	21	188	0.006262700289816		0.00478785	2.17518337737167E-06	3.60731493986666E-05		
23	not	22	183	0.006096139111896		0.00459214	2.26201332858335E-06	3.4100128278424E-05		
24	all	23	174	0.005796328991639		0.0044126	1.91470592230126E-06	3.06885130969786E-05		
25	but	24	171	0.005696392284886		0.00424729	2.09989743206252E-06	2.95912574179229E-05		
26	be	25	171	0.005696392284886		0.00409455	2.56589870564957E-06	2.95912574179229E-05		
27	napoleon	26	168	0.005596455578134		0.00395297	2.70104484553395E-06	2.85139764295801E-05		
28	farm	27	166	0.005529831106966		0.00382137	2.91883935401408E-06	2.78068861544145E-05		
29	been	28	162	0.005396582164629		0.0036987	2.88280384496549E-06	2.64193385250338E-05		
30	them	29	158	0.005263333222293		0.00358406	2.8199585551088E-06	2.50673014569207E-05		
31	there	30	141	0.004697025217362		0.00347669	1.48921804273478E-06	1.97173086165818E-05		
32	would	31	138	0.00459708851061		0.0033759	1.49130137844574E-06	1.88397745353694E-05		
33	this	32	138	0.00459708851061		0.0032811	1.73182576005739E-06	1.88397745353694E-05		

Results

- R^2 value for each of the books is given below :

Name of Book	R^2 Value
Animal Farm	0.97
1984	0.98
The Picture of Dorian Gray	0.86
Robinson Crusoe	0.82
Teens Wander Around A House	0.92

Inference

- Since the R^2 value is highly positive, we can conclude that the Zipfian Distribution is a good model to fit the data of rank-frequency of words in any given book, or generally speaking, in any given body of natural text.
- That is, Zipf's Law has been validated.

Limitations of Study

- The cripplingly small amount of data chosen as sample helps only to verify Zipf's Law for a very small subset of available bodies of natural language.
- The parameter for the PDF of the distribution was empirically estimated, and not calculated, leading to imprecision.

References

- Project Gutenberg : https://gutenberg.org
- Project Gutenberg AU : https://gutenberg.net.au
- CPlusPlus : https://cplusplus.com
- Calibre : https://calibre-ebook.com
- WolframALpha : https://wolframalpha.com
- DariusK's NaNoGenMo Repo : https://github.com/dariusk/dialogue/blob/master/index.js