Charalambos Charalambous

Step 1: Download the file from the given link

Step 2: I implemented in Java a code to first read the data of interest from the file and see how the code is structured in the file to then figure out how to extract the data I need

Initial code:

```
package ry; import
java.io.*;

public class ry{

public static void main(String[] args) throws IOException {


FileReader fileReader=new
FileReader("C:\\\\Users\\Admin\\OneDrive\Documents\Panepistimio\Analisi
dedomenon\dblp-2020-04-01.xml");;
BufferedReader bufferedReader=new BufferedReader(fileReader);;
FileWriter fileWriter=new FileWriter("output.txt");
BufferedWriter bufferedWriter=new BufferedWriter(fileWriter);
PrintWriter printWriter=new PrintWriter(bufferedWriter);
             String line;{
for(int i=0;i<2000;i++){
       line=bufferedReader. readLine();
      printWriter.println((line));;
        }


               bufferedReader.close(); bufferedWriter.flush();
bufferedWriter.close();



}
}
}
```

Next, to extract the number of publications per year I have implemented a method in a class that returns an integer, which takes as input a string, given by main, which is each new row in the file. This

method checks if the string is empty or has >6 characters. It truncates the first 6 and checks if they are equal to "<year>"; if they are equal it takes the next 4 characters which is the chronology and returns it to the program. Main first opens the file and reads it line by line to the end in one iteration. Inside, it calls the method with the new string adds to a table representing the dates from 1918 to 2020 one unit, in the position returned by the method. At the end it writes to the output file the whole table right adding as 1st column the 1918 of the table to get the dates correct and the publications are automatically sorted since I used treemap.

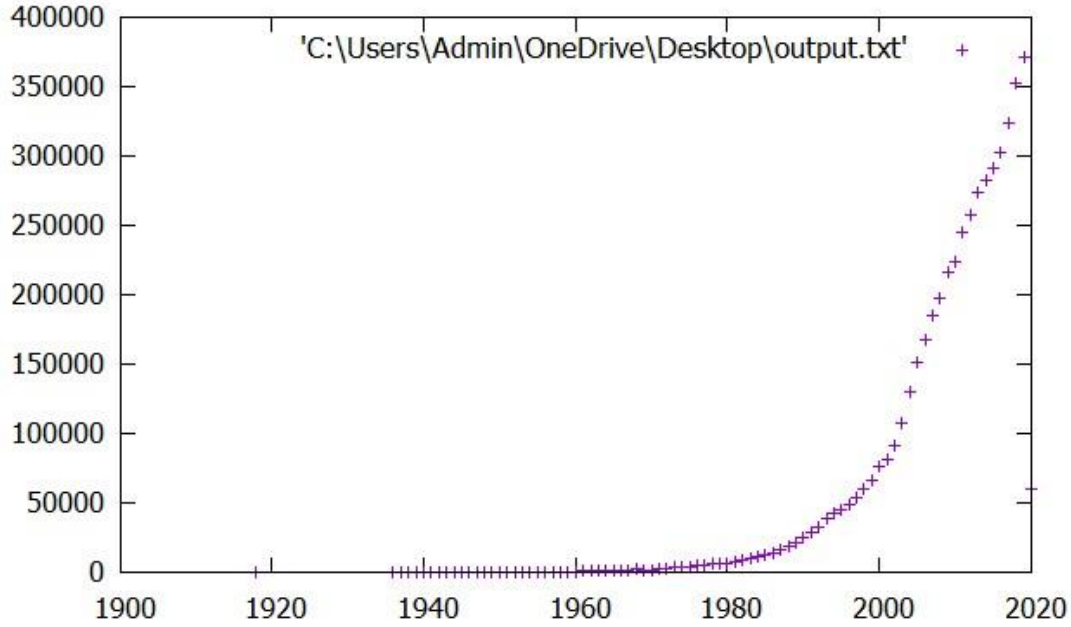
Final code:

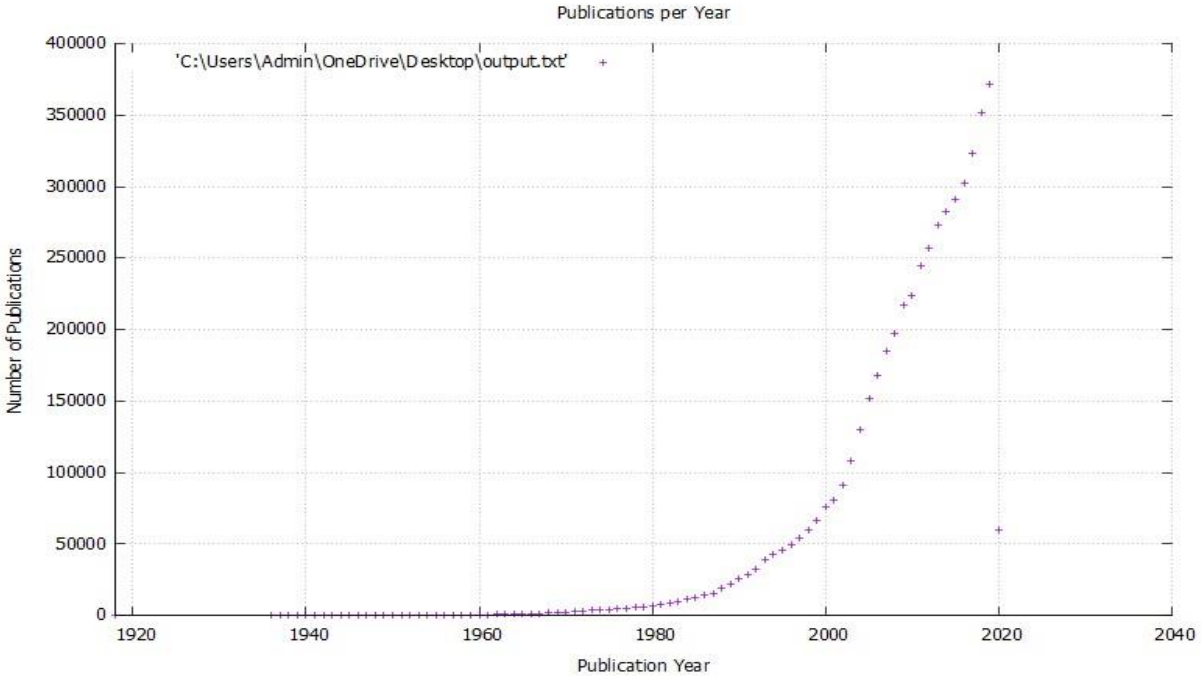
```
import java.io.*; import java.util.Map; import java.util.Map.Entry; import

java.util.Map.Entry; import java.util.TreeMap; public class Main{            private static

Map<String,Integer> yearMap = new TreeMap<String,Integer>();            private static

void parseData(String line) throws Exception{

    String year;

 if(line != null &&( !(line.isEmpty()))&&(line.length()>6)){

          year=line.substring(0,6);          if(year.equals("<year>")){

  year=line.substring(6,10);

  if (yearMap.containsKey(year)) {        yearMap.put(year, yearMap.get(year) + 1);

       } else {

   yearMap.put(year, 1);

           }

           }

           }

           }

public static void main(String[] args) throws IOException {

          try{ FileReader fileReader=new
FileReader("C:\\\\Users\\\\Admin\\\\OneDrive\\Documents\\Panepistimio\\Analisi
dedomenon\\dblp-2020-04-01.xml");;

     BufferedReader bufferedReader=new BufferedReader(fileReader);;

     FileWriter fileWriter=new FileWriter("output.txt");

     BufferedWriter bufferedWriter=new BufferedWriter(fileWriter); PrintWriter

printWriter=new PrintWriter(bufferedWriter);

     try{
```

```
        String line;
line=bufferedReader.readLine();
while(line!=null){ parseData(line);
line=bufferedReader.readLine();
          }
          for (Entry<String, Integer> entry : yearMap.entrySet()) {
printWriter.println(entry.getKey() + "\t" + entry.getValue());
          }
      } finally{
bufferedReader.close();
bufferedWriter.flush();
bufferedWriter.close();
      }
      }catch(FileNotFoundException e){
          e.printStackTrace();
}catch(IOException e){
          e.printStackTrace();
      }catch(Exception e){
          e.printStackTrace();
      }
   }
}
```

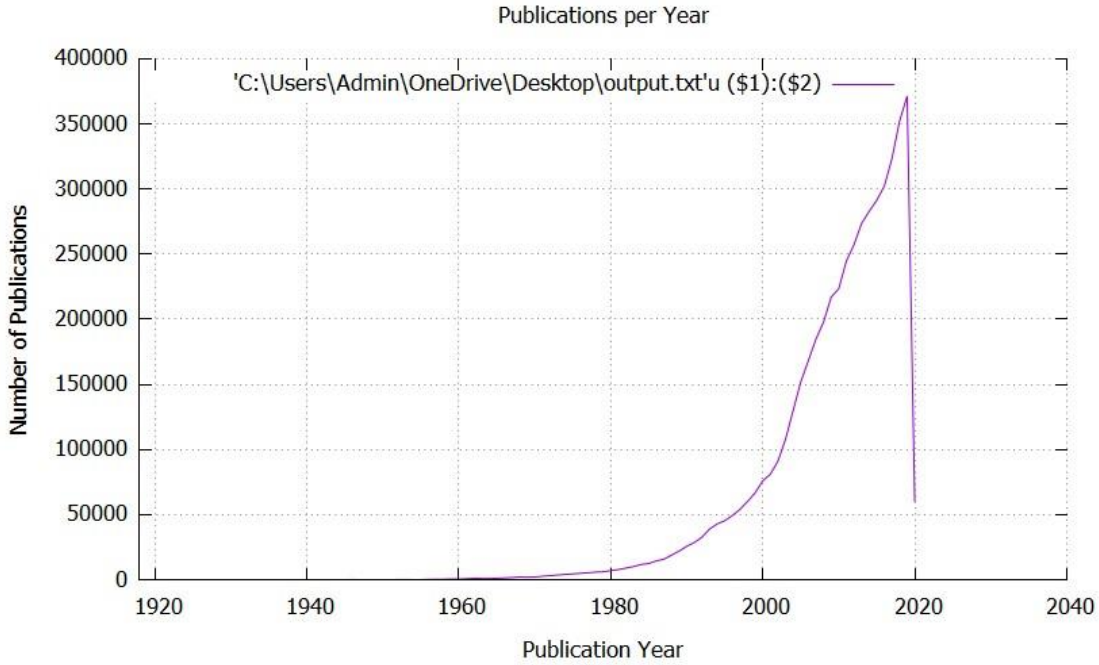
And then I run various commands in gnuplot to represent the data of the table I have taken

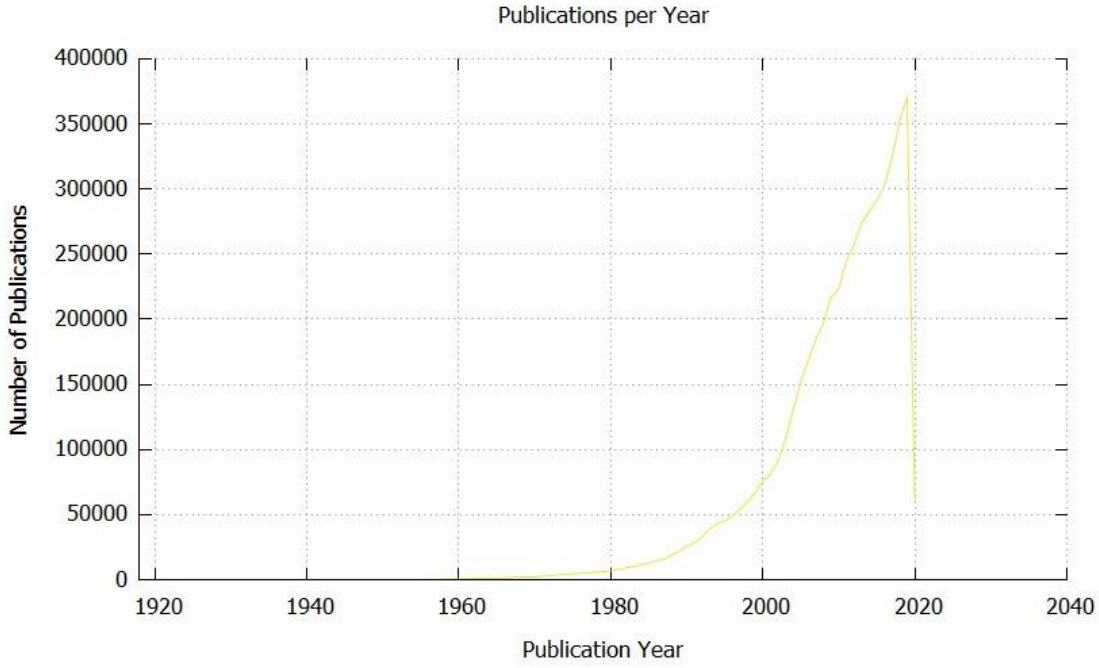
At first just to show what the simple data graph looks like

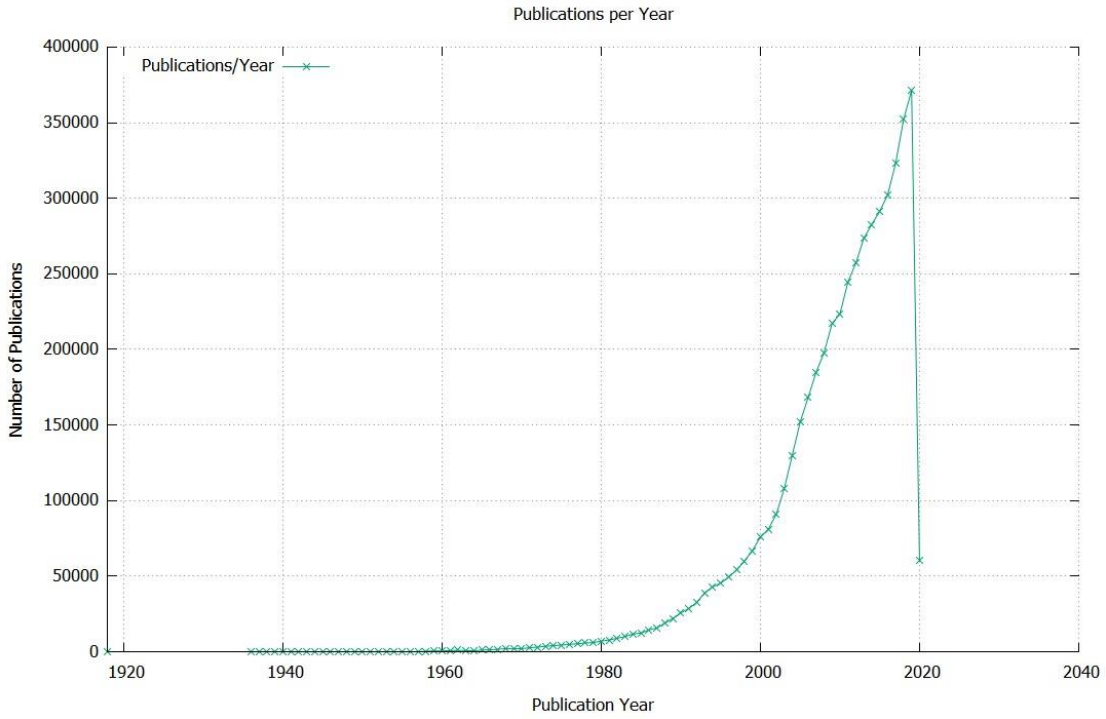




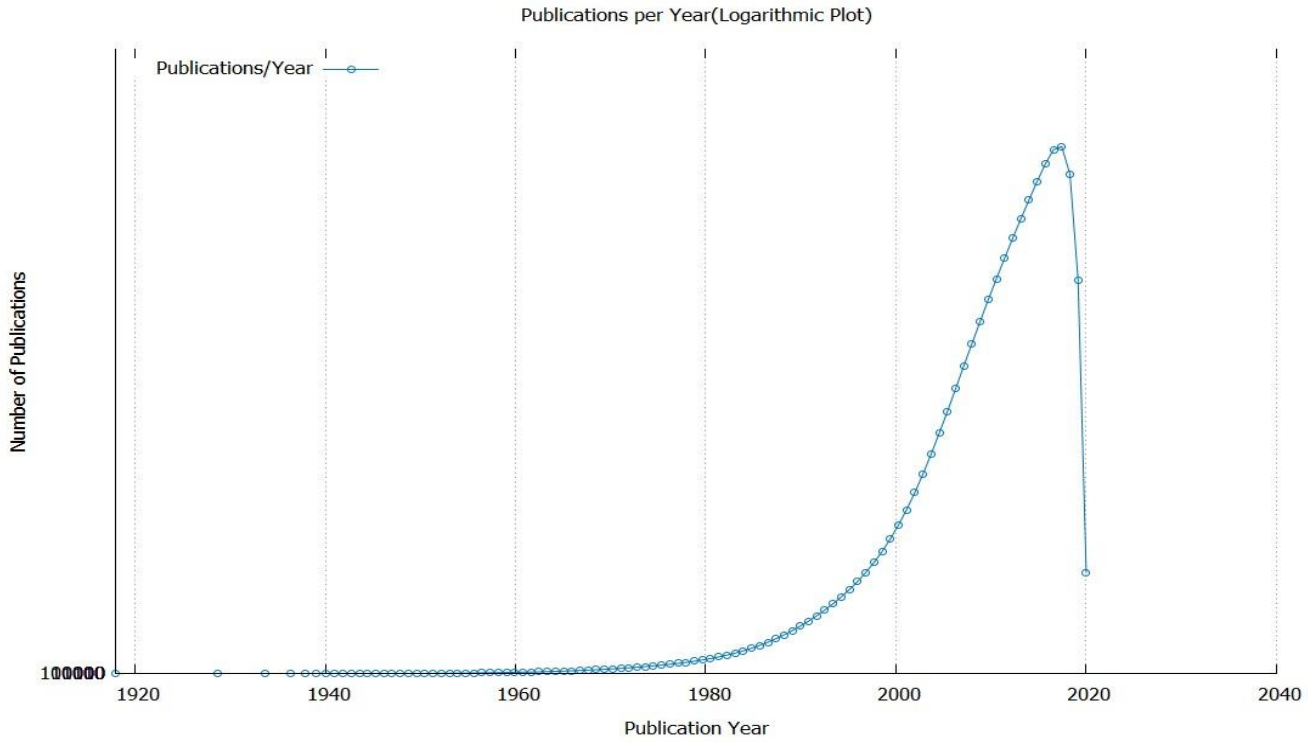
And then with various representation techniques to see the data better.


Publications per Year
'C:\Users\Admin\OneDrive\Desktop\output.txt'u ($1):($2)
Number of Publications
Publication Year
0
50000
100000
150000
200000
250000
300000
350000
400000
1920
1940
1960
1980
2000
2020
2040

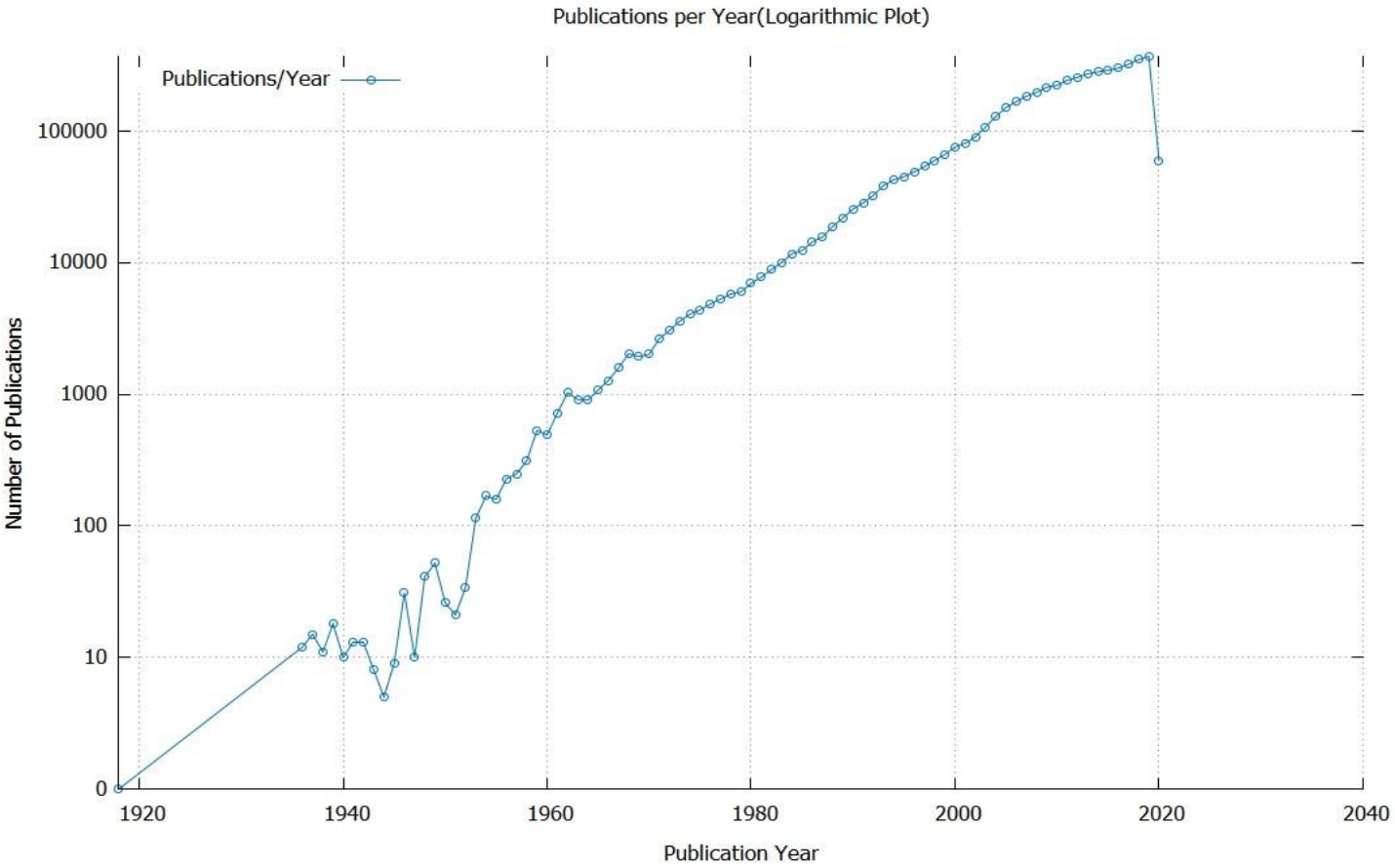

Publications per Year
Number of Publications
Publication Year
0
50000
100000
150000
200000
250000
300000
350000
400000
1920
1940
1960
1980
2000
2020
2040

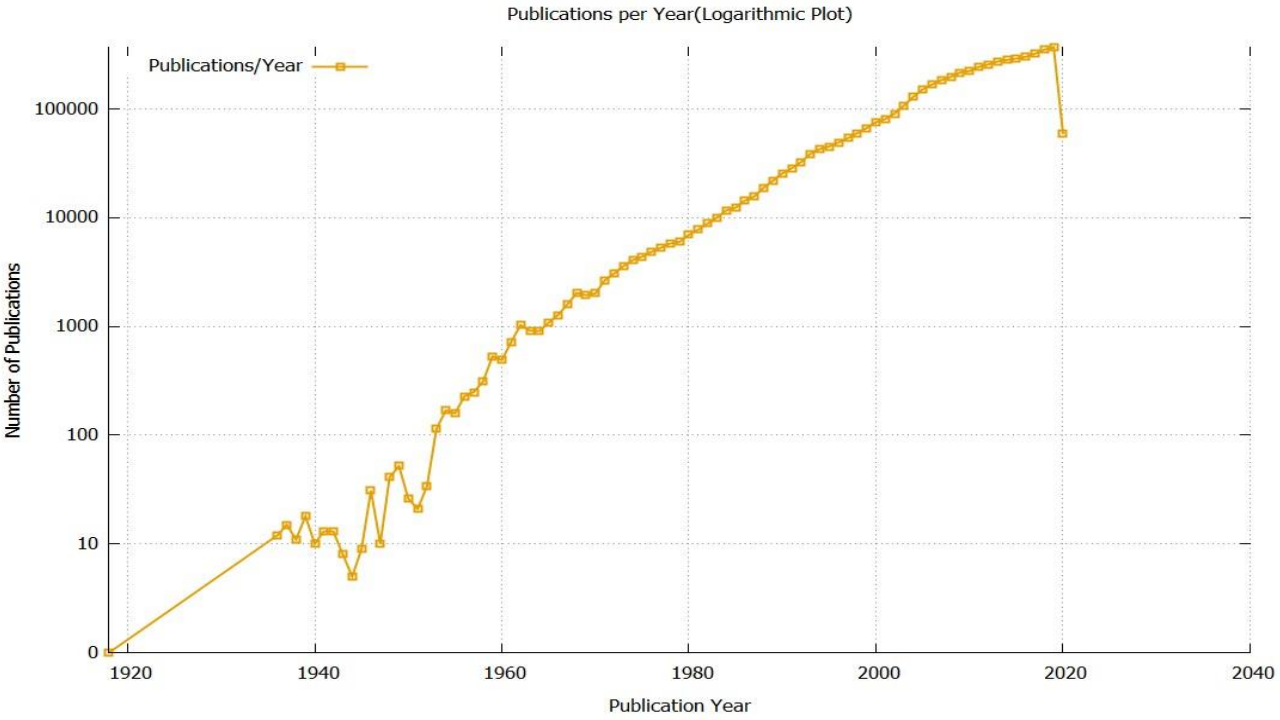


Then I do a simple smooth first visualization of the data

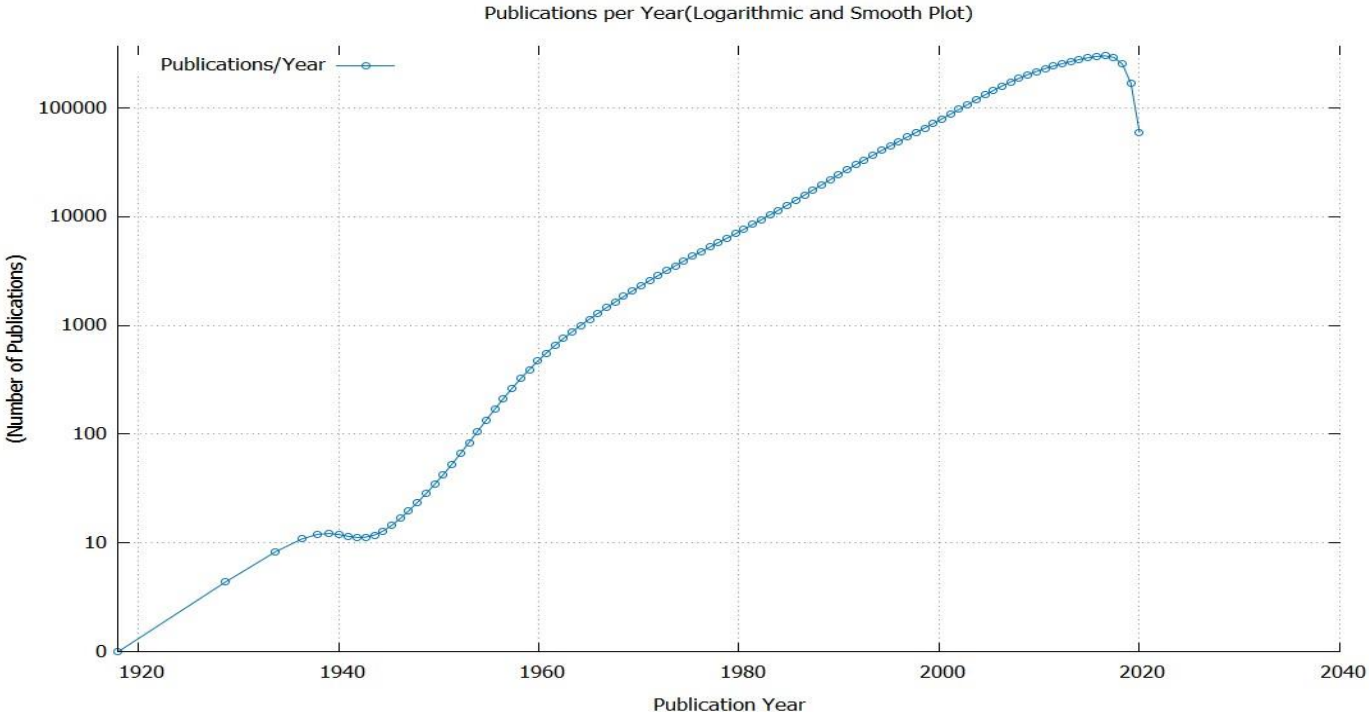


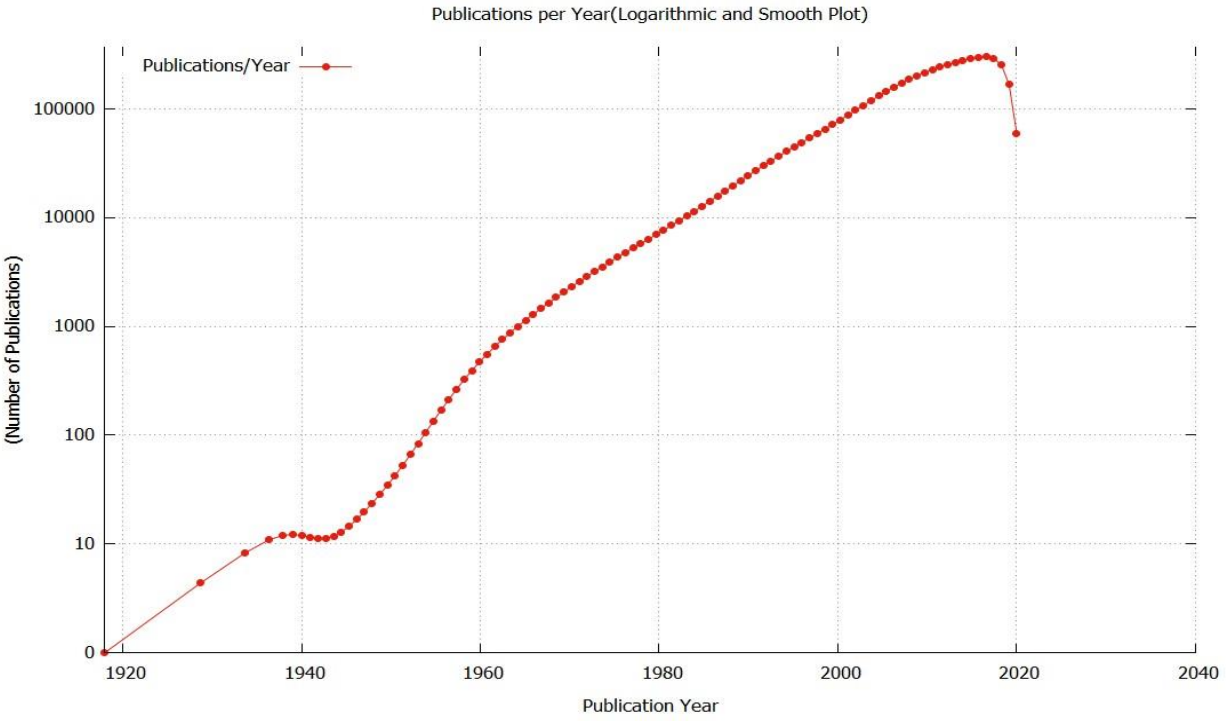
We then logarithmically approximate a better representation of the data to achieve a better representation.





And then we smooth the logarithmic for visualization purposes to get rid of the noise



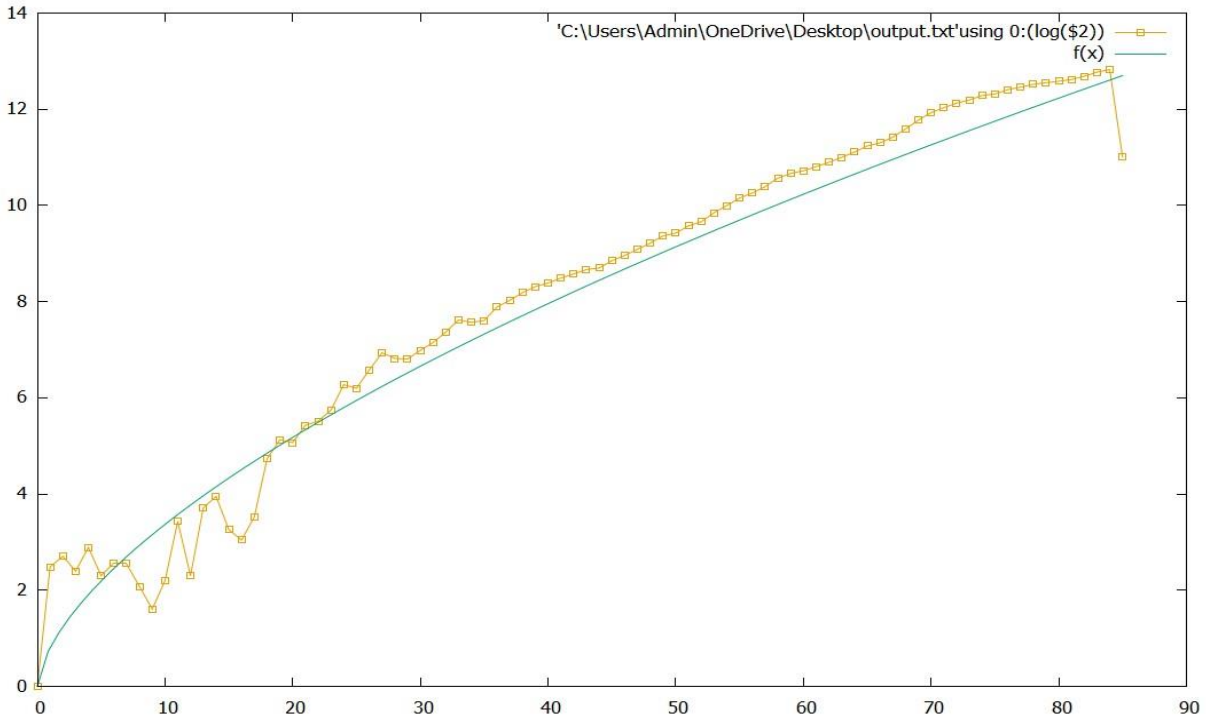


Βήμα 3ο:

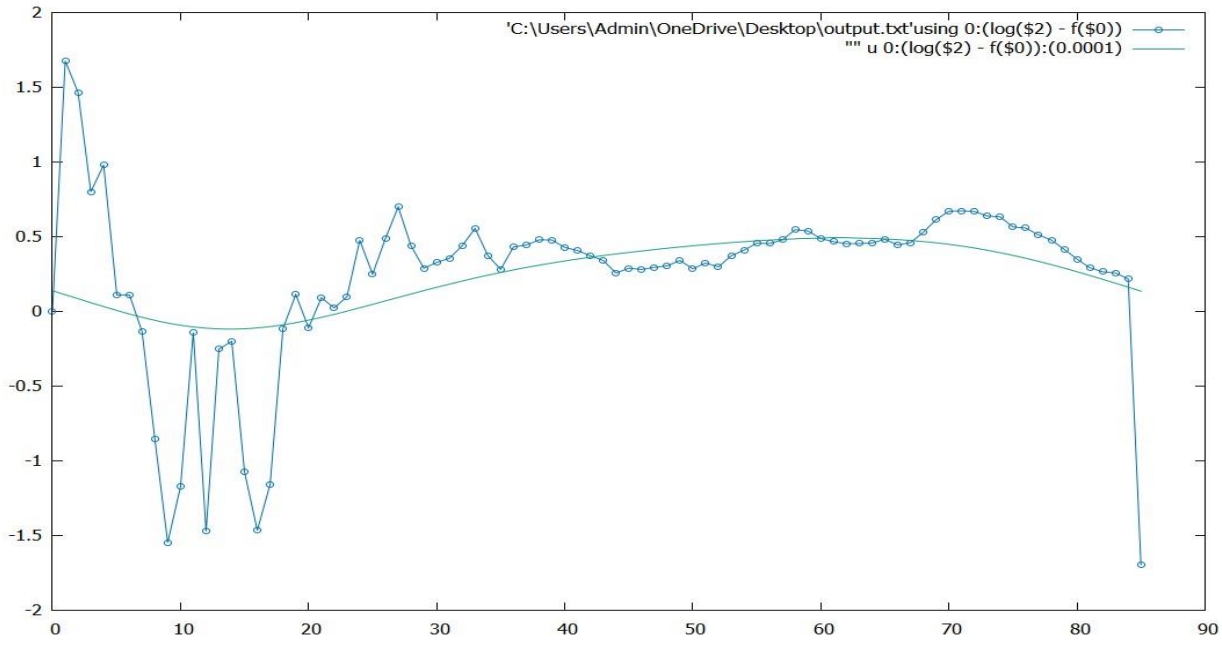
Το πιο κυρίαρχο χαρακτηριστικό είναι η τάση. η τάση είναι μη γραμμική: εάν αγνοήσουμε τη βραχυπρόθεσμη διακύμανση, η καμπύλη είναι κυρτή προς τα πάνω. Σε αυτό το στάδιο θα εφαρμοστούν τεχνικές ανάλυσης χρονοσειρών στα εξαγόμενα δεδομένα, με αποτέλεσμα να εμφανιστούν περαιτέρω στοιχεία του δείγματος μας. Σε αυτό το στάδιο καλούμαστε να δημιουργήσουμε μια συνάρτηση f(x) η οποία θα αποτελεί μια προσέγγιση της αυξητικής τάσης του γραφήματος. Γνωρίζουμε όμως για αρχή ότι η συνάρτηση πρέπει να έχει κοινή αρχή και να συμπίπτει με το γράφημα μας σε ένα τουλάχιστον σημείο. Οπότε αρκεί λοιπόν να εντοπίσουμε ένα διακριτό σημείο στο γράφημα μας. Σύμφωνα με τη βιβλιογραφία γνωρίζουμε ότι η σχέση που περιγράφει την παραπάνω συνάρτηση είναι η $f(x) = b * (x/a)k$ .We distinguish the point (x=85,y=12.7). Therefore our function will have the formula :

$f(x) = 12.7 * (x/85)**$

We then give different values for k until we find a value for which our function best describes the trend of our data. And after several experiments the value k = 0.62 gives a pretty good approximation for the trend of the data



We can then create a voltage-free representation by simply subtracting the values of f(x) from the values of its y-axis and for ease of visualization approximate it using a Smooth curve. This results in the following illustration


'C:\Users\Admin\OneDrive\Desktop\output.txt'using 0:(log($2) - f($0))
"" u 0:(log($2) - f($0)):(0.0001)
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
10
20
30
40
50
60
70
80
90