Assignment 5 - MAT 3375
Iona Buchanan
6671041
Dec 1, 2016

# Transformation of non-linear to linear model

One way to estimate the curve of best fit for a non-linear model is to transform the data into linear data by taking functions of the dependent and independent variables. Once we have linear data, we can estimate $\beta_0$ and $\beta_1$ for a linear regression function $\hat{y} = \beta_0 + \beta_1 x$. These coefficients are then the corresponding coefficients to transform the non-linear parent function. We can then preform regression analysis on the results.

To transform the non-linear data, we must estimate the parent function of the curve. This will tell us which transformations to apply to the data. For example, the following graph (Figure 1) looks to show quadratic data ($\hat{y} = (\beta_0 + \beta_1 x)^2$). Therefore, we should plot the square root of the dependent variable versus the independent variable which to give $\sqrt{y} = \beta_0 + \beta_1 x$, which is linear with respect to $\sqrt{y}$.

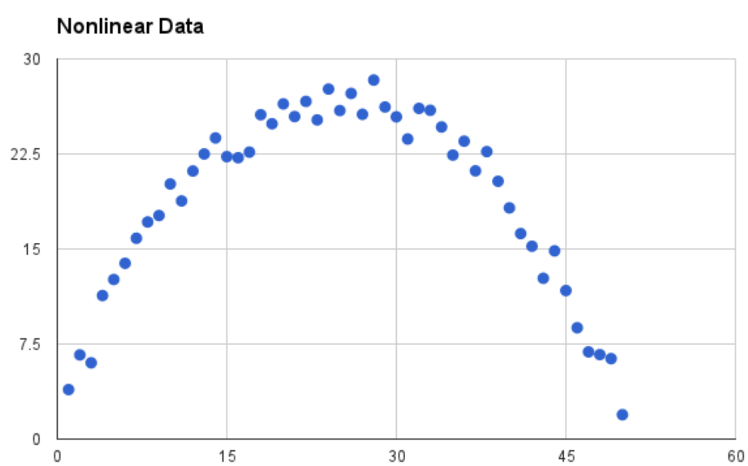



Figure 1: Quadratic data[1]

Table 1 gives the transformations and corresponding regression equations for each type of model.

[1] http://3.bp.blogspot.com/-wg7eovEn0GA/VEXW-DPhGuI/AAAAAAAAD4Y/hDWD4TZKRWU/s1600/quadratic_data.png

Table 1: Transformations for Common Parent Functions.[2]

| Method | Transformation(s) | Regression equation | Predicted value ($\hat{y}$) |
|---|---|---|---|
| Standard linear regression | None | $y = b_0 + b_1x$ | $\hat{y} = b_0 + b_1x$ |
| Exponential model | Dependent variable = log(y) | $\log(y) = b_0 + b_1x$ | $\hat{y} = 10^{b_0 + b_1x}$ |
| Quadratic model | Dependent variable = sqrt(y) | $\text{sqrt}(y) = b_0 + b_1x$ | $\hat{y} = ( b_0 + b_1x )^2$ |
| Reciprocal model | Dependent variable = 1/y | $1/y = b_0 + b_1x$ | $\hat{y} = 1 / ( b_0 + b_1x )$ |
| Logarithmic model | Independent variable = log(x) | $y= b_0 + b_1\log(x)$ | $\hat{y} = b_0 + b_1\log(x)$ |
| Power model | Dependent variable = log(y)<br>Independent variable = log(x) | $\log(y)= b_0 + b_1\log(x)$ | $\hat{y} = 10^{b_0 + b_1\log(x)}$ |

[2] StatTrek.com. (2016). *Transformations to Achieve Linearity*. http://stattrek.com/regression/linear-transformation.aspx?Tutorial=AP

## Example: Frequency of Word Lengths in a document (*War and Peace* by Leo Tolstoy).

In most English text, longer words tend to appear less often that shorted ones. For example, there are usually a lot more four-letter words than 20-letter words. I decided to construct a data set based on the amount of words of certain length in the text *War and Peace* by Leo Tolstoy[3]. As this is a very long text, I hoped to get lots of data.

I wrote a program in Java that calculates the length of each word in a text and the number of n-letter words (Appendix B). The program then exports the data to a .csv file (Appendix A).

I then imported the data into R (Appendix B) and began by plotting the data. As you can see the data is decidedly non-linear (Figure 2).

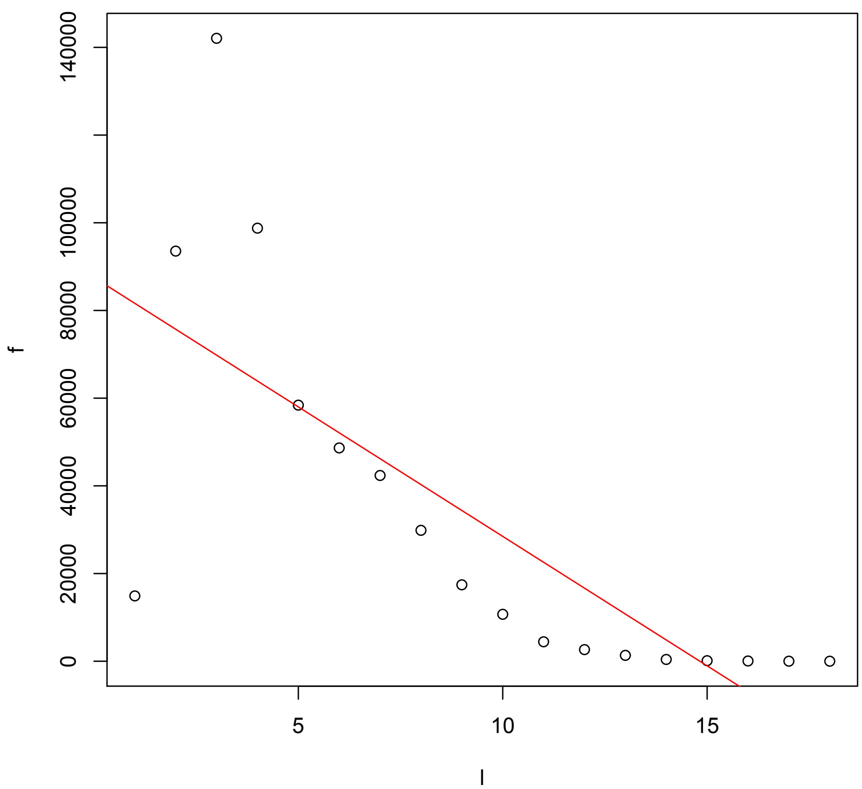



Figure 2: Plot of word length versus frequency

[3] Tolstoy, L. (posted 2009). *War and Peace.* https://www.gutenberg.org/files/2600/2600-0.txt

After preforming residual analysis (Figure 3), we can see that this fit is not accurate. From the residual plot we can confirm that the model is not linear. As predicted, most of the error comes from trying to fit non-linear data to a linear model.

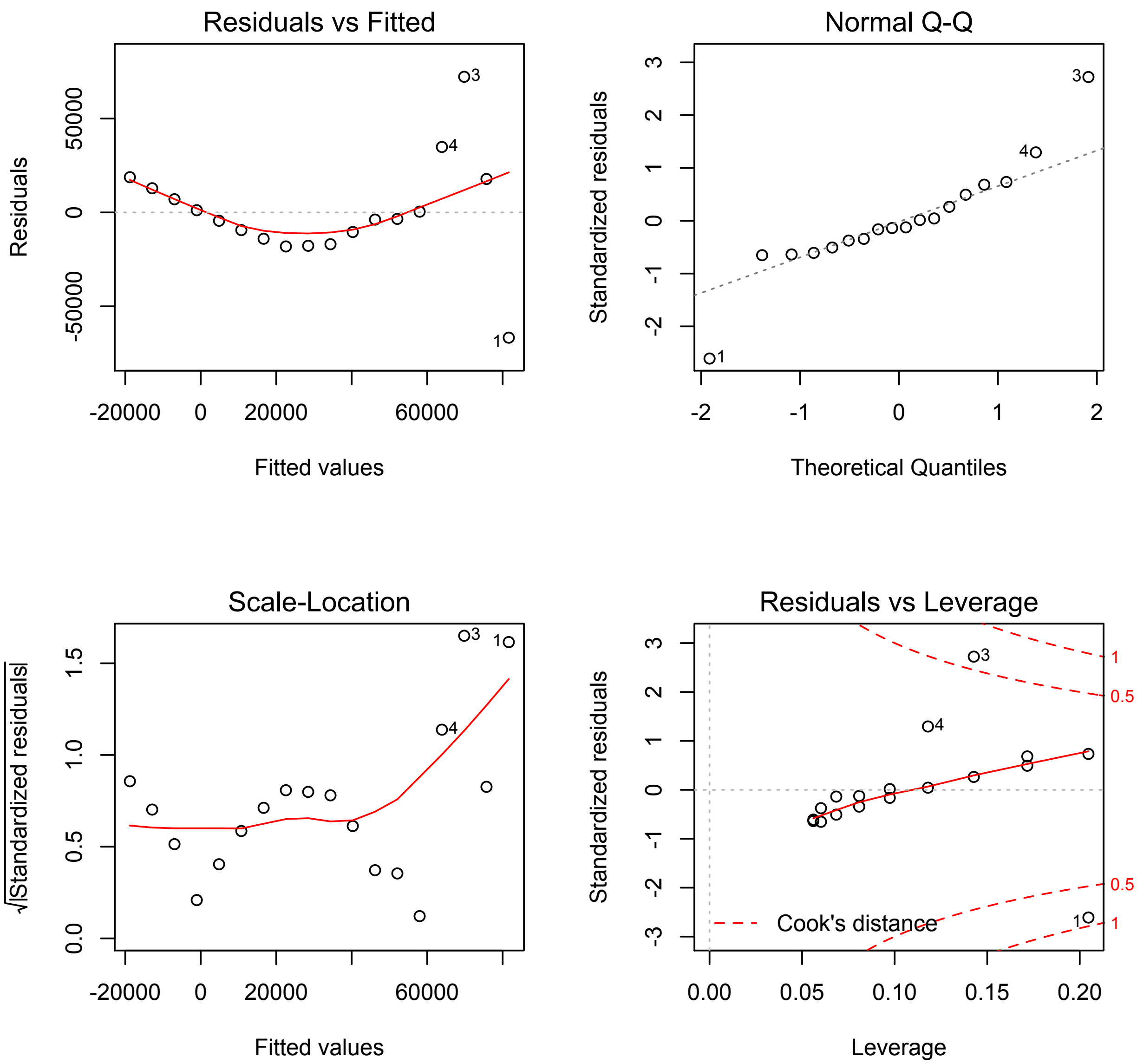



Figure 3: Residual analysis of original model

I then tried various transformations to find the model that best fit the data (Figure 4).

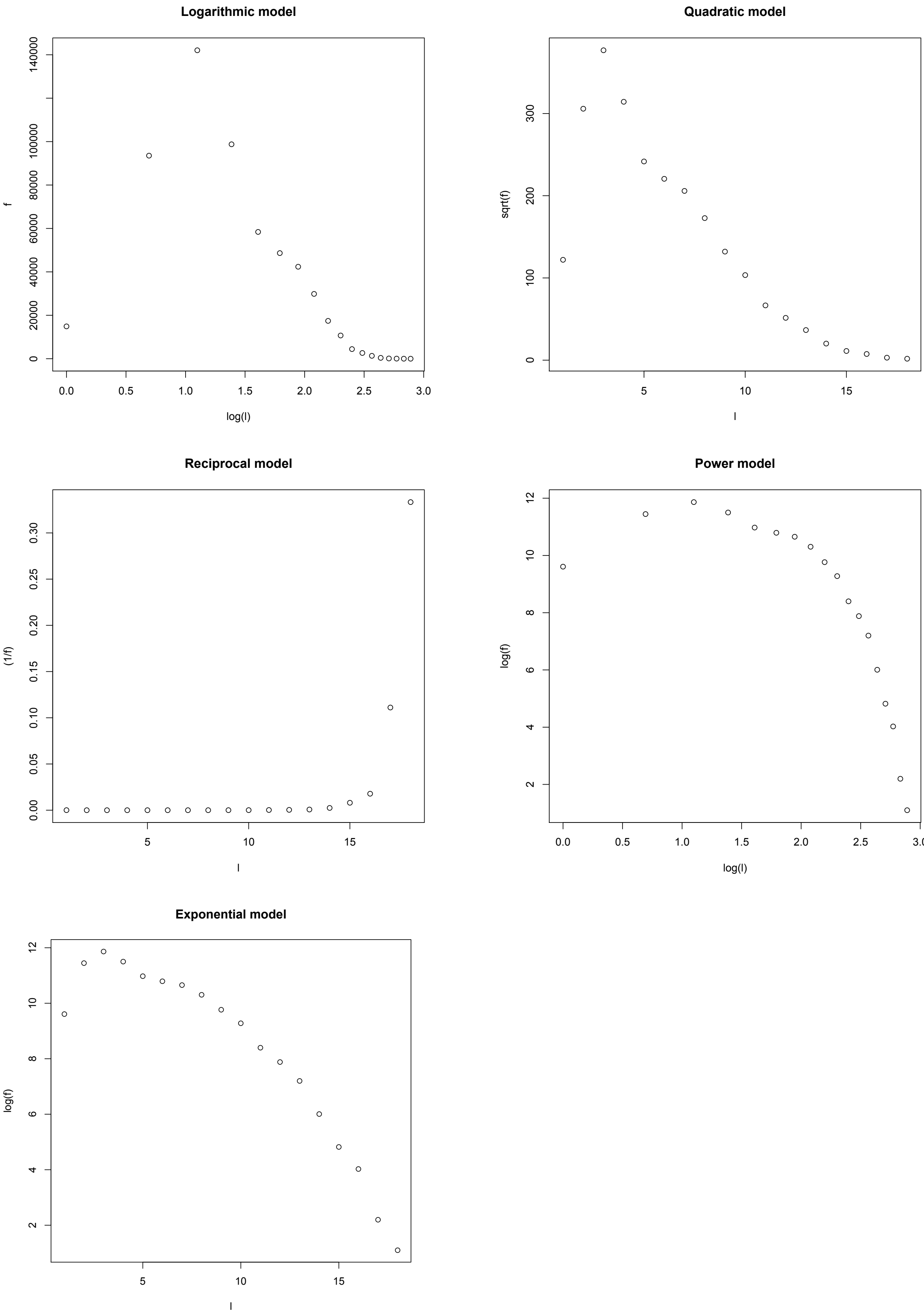
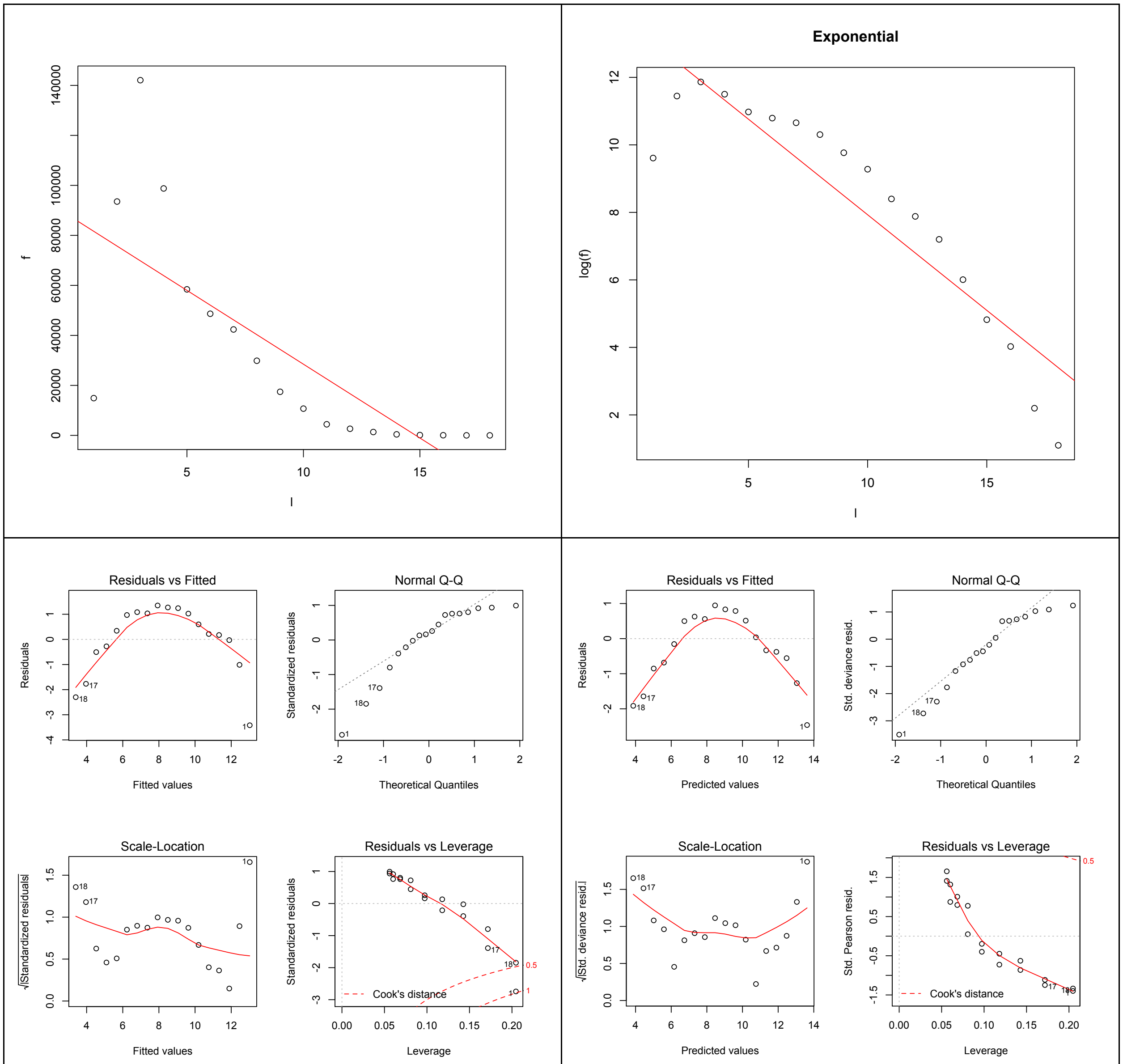



Figure 4: Various transformations

The exponential model is not perfect but it is the best out of the five transformations.

I then compared the original model to the new exponential model:




| Multiple R-squared:  0.5624 | Multiple R-squared:  0.8327 |
|---|---|

The multiple R-squared for the linearized exponential model is better than the original model. Although we still do not have a linear model (biased residual plot), the Q-Q plot is straight and we can barely see Cook's lines.

Using the exponential model, the line of best fit is $\hat{y} = e^{13.5918 - 0.5663x}$ (Figure 5).

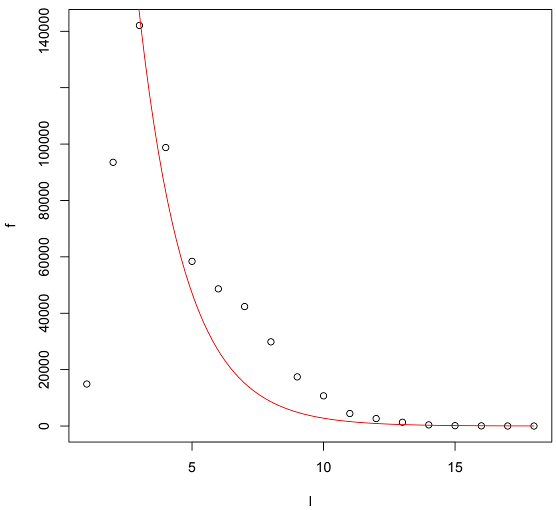



Figure 5: Exponential fit

However, perhaps there is an even better model. The original data is heavily skewed to the right, like negative binomial, Poisson, beta and gamma distributions. It now makes sense that the exponential transformation was the most successful since these distributions have exponential probability distribution functions. Since our data is discrete, it will follow either a negative binomial or Poisson distribution.

If our data was normal but with slight right skew, it would be possible to centralize the data using a logarithmic approach. Notice that the logarithmic model has reduced the skew of the data but it is still not centred. Therefore, we are still unable to linearize due to the heavy skew.

Let's now try to linearize, assuming the data has Poisson distribution.
The probability mass function of the Poisson distribution is $y = \frac{\lambda^x e^{-\lambda}}{x!}$.
Therefore, we have

$$y = \frac{\lambda^x e^{-\lambda}}{x!}$$
$$x!\,y = \lambda^x e^{-\lambda}$$
$$\log(x!\,y) = \log \lambda^x e^{-\lambda}$$
$$\log(x!\,y) = x \log \lambda + \log e^{-\lambda}$$
$$\log(x!\,y) = x \log \lambda - \lambda$$

Therefore, if $x!\,y$ is exponentially proportional to $x$ so if we plot $\log(x!\,y)$ versus $x$, we should get a linear model.

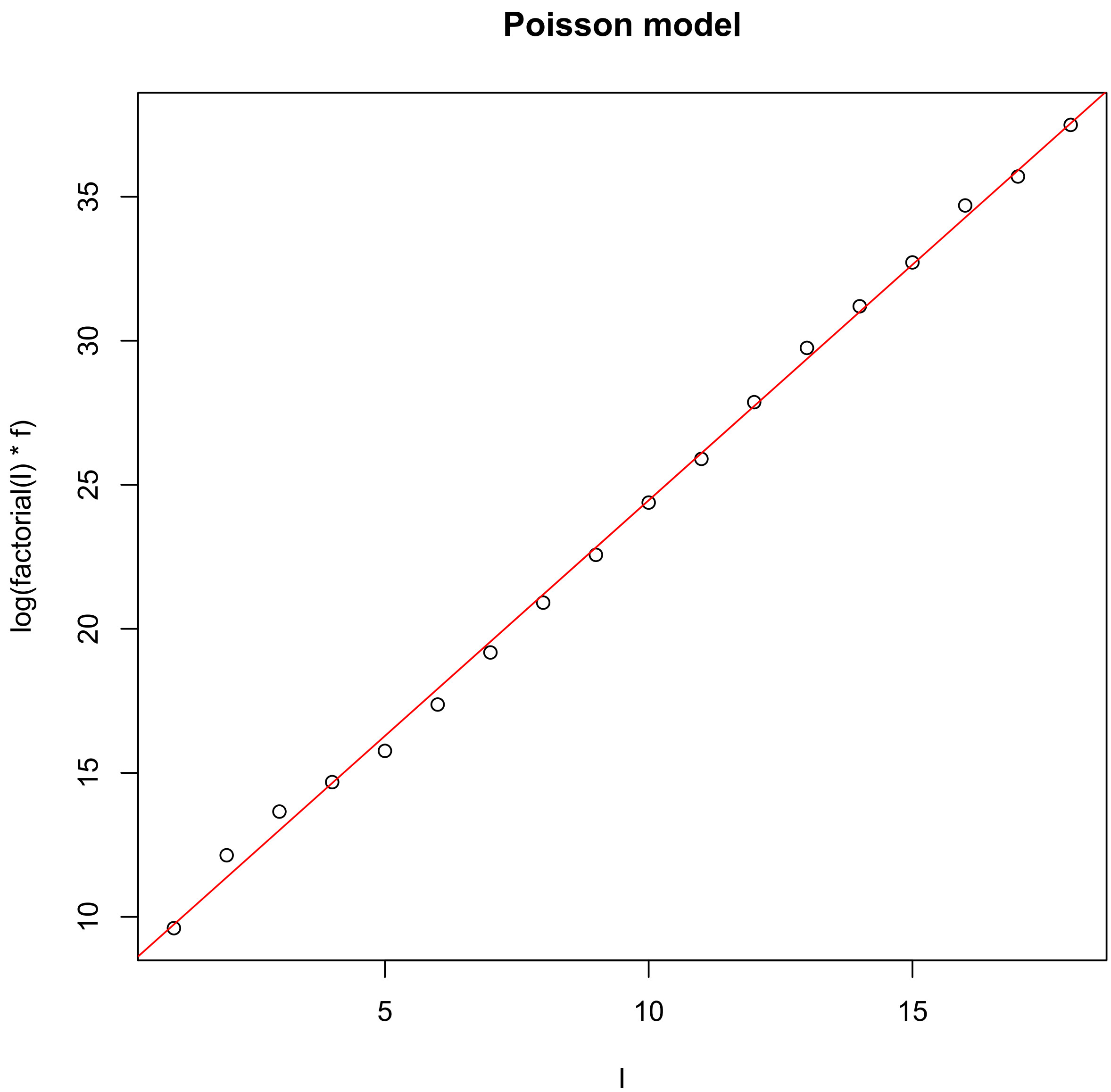



Figure 6: Poisson fit

As predicted, we have a fairly accurate linear model.

Now, we perform residual analysis on our linear model.

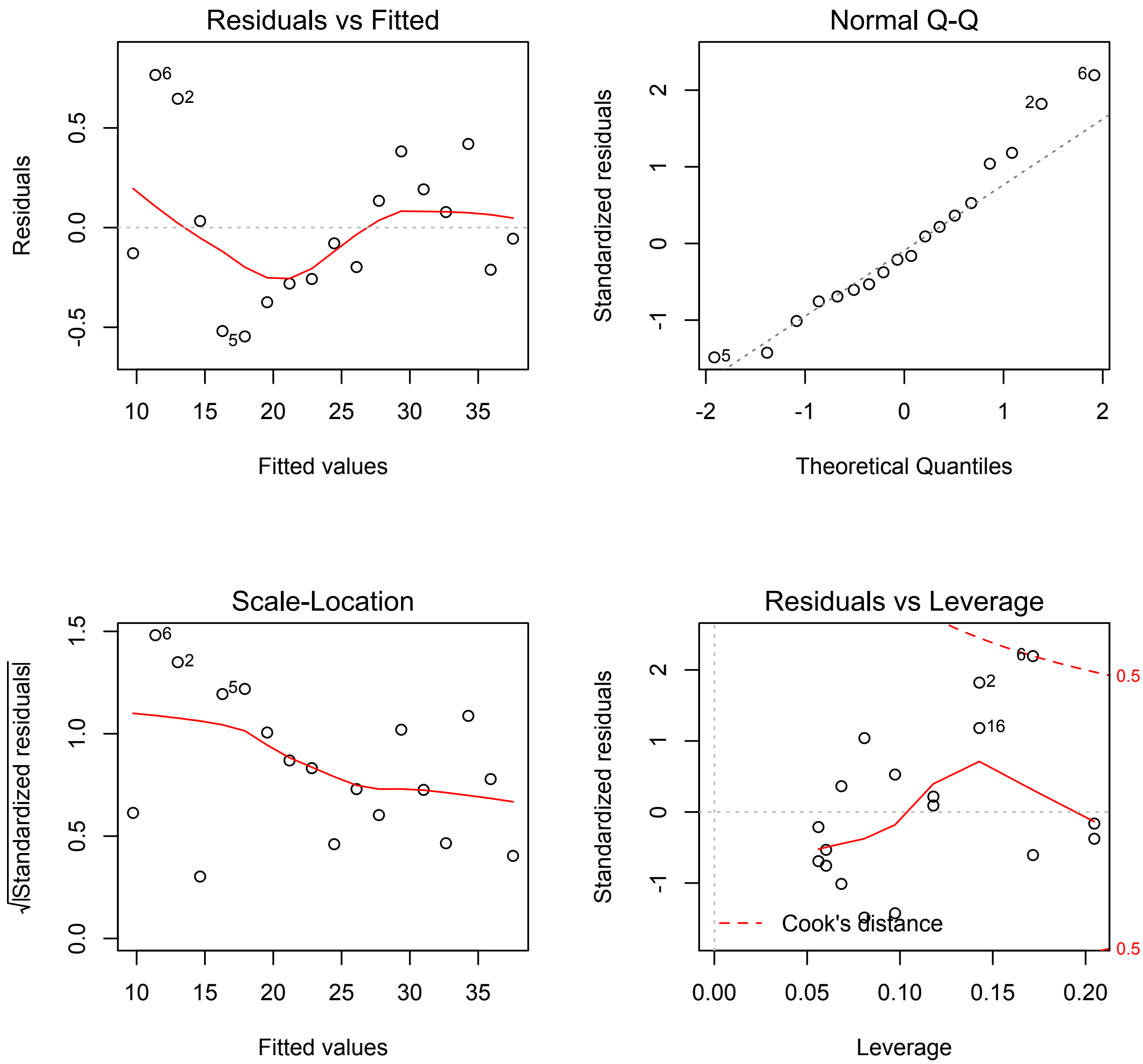



Figure 7: Residual analysis of Poisson fit

From the residual analysis, we can see that we have a good linear fit. The residual versus fitted plot is unbiased and shows no heteroscedasticity. The QQ-plot is fairly straight and we see little of the Cook's lines.

Moreover, the R-squared value is 0.998 which is very good.

$$\log(x!\,y) = 8.10162 + 1.636x$$
$$x!\,y = e^{8.10162+1.636x}$$

Therefore we can plot the exponential curve of best fit on $x!\,y$ versus $x$.

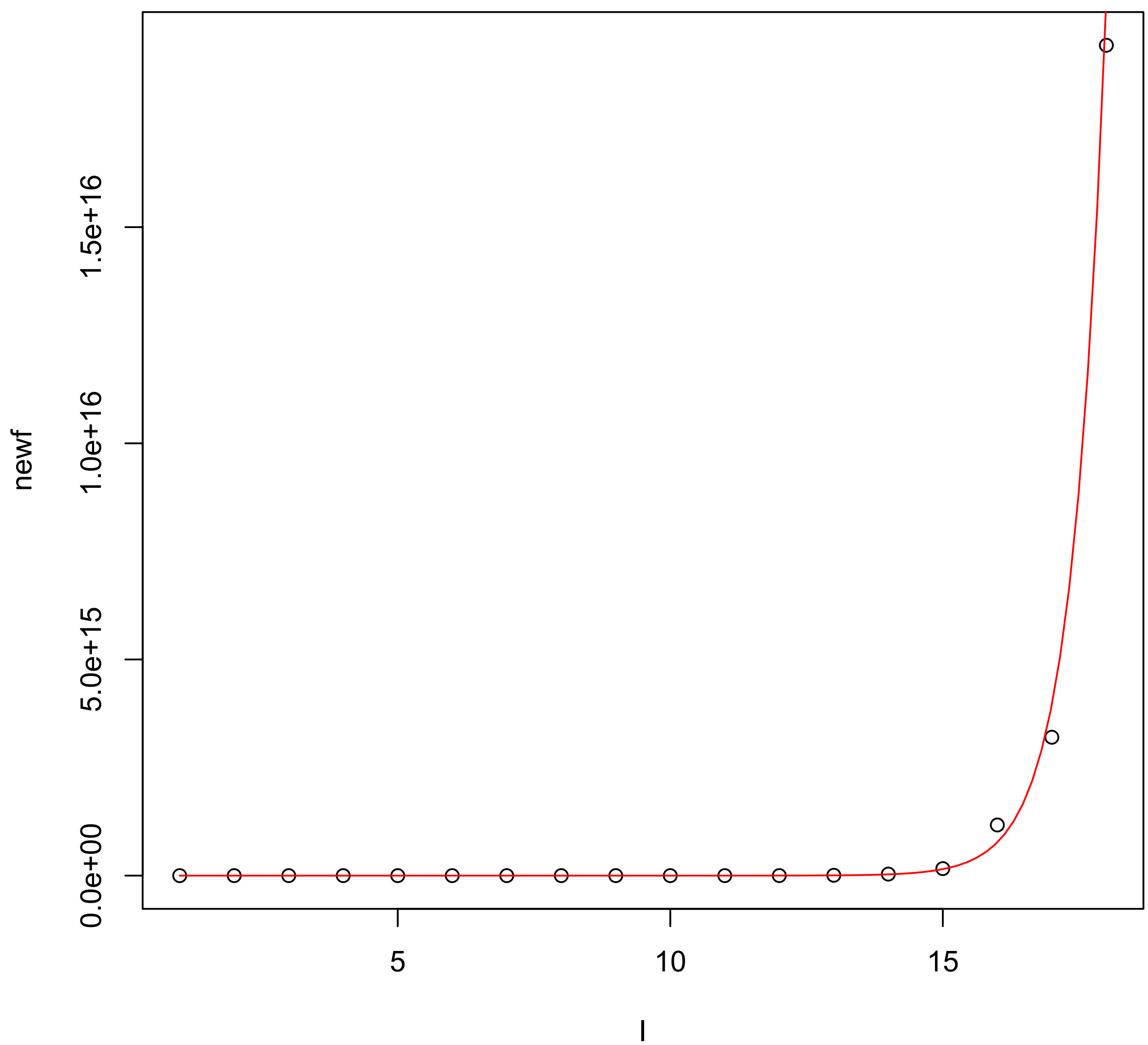



Figure 8: Length!*frequency versus Frequency

To plot the curve of best fit on the original function we have,

$$x!\,y = e^{8.10162+1.636x}$$

$$y = \frac{e^{8.10162+1.636x}}{x!} = \frac{e^{8.10162}5.13459^{x}}{x!}$$

which is almost Poisson.

When we plot this in R we get

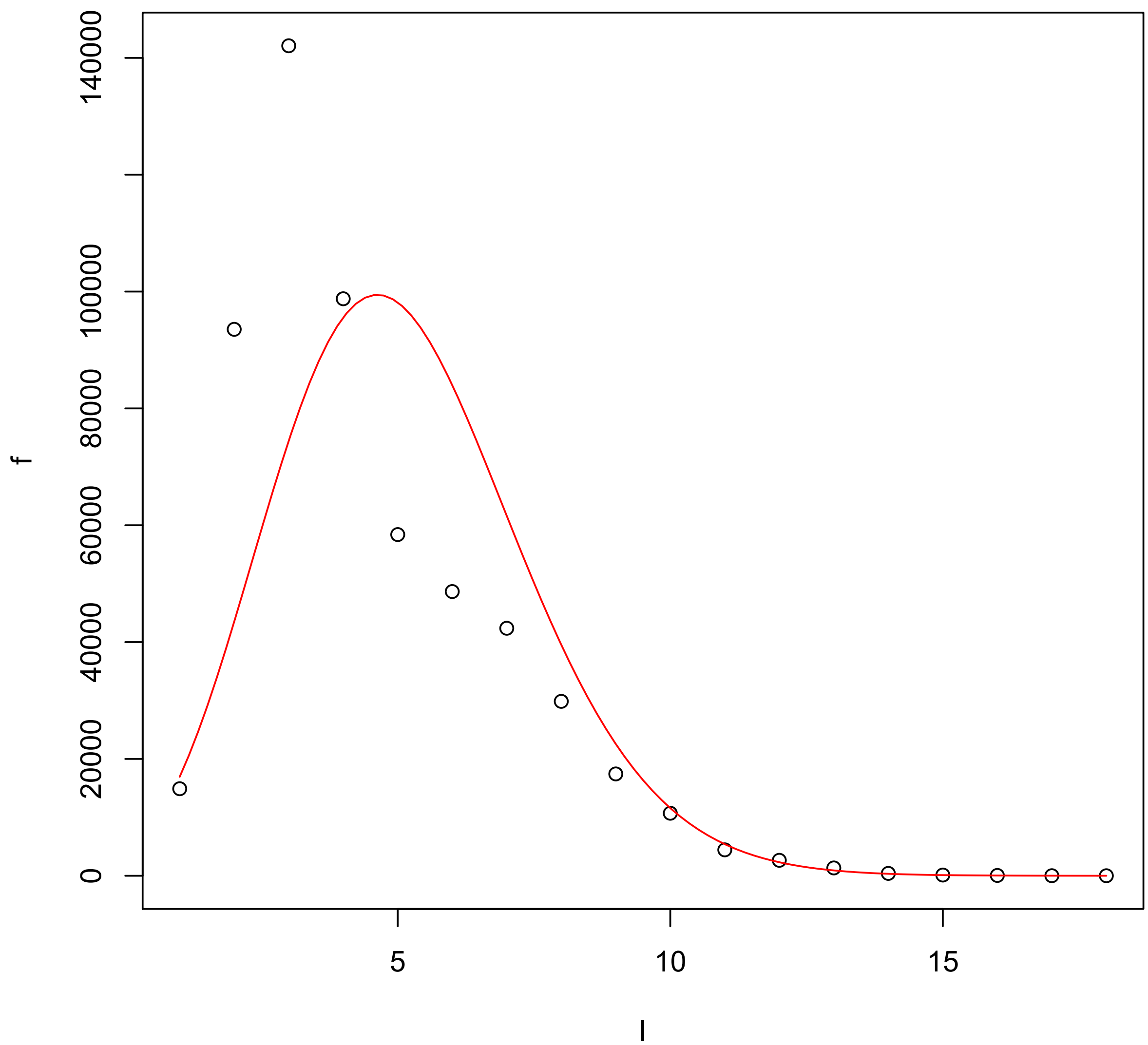



Figure 9: Curve of best fit

## GLM function in R

In R, the function glm() is used for data that follows non-normal exponential distributions. Since our data is close to Poisson, let's try fitting the data using a quasiPoisson generalized linear model with log link.

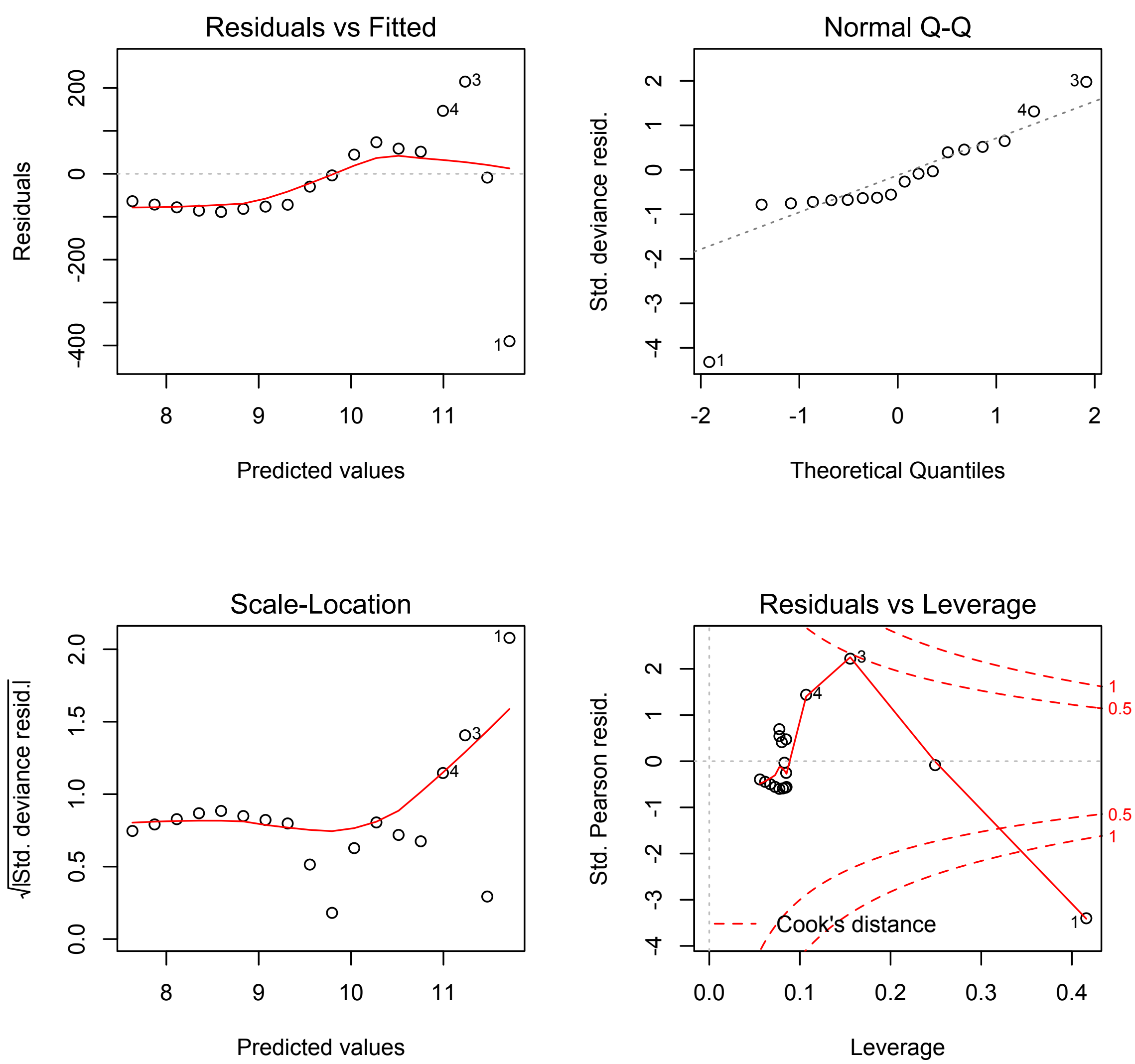



Figure 10: Residual Analysis of Poisson fit using glm()

The results (Figure 10) are a bit better than the exponential fit but not as good as our previous Poisson model. Although we see more Cook's distant lines, the residual vs. fitted plot is less curved. We have some heteroscedasticity but there is more evidence of a linear model. Moreover, Hoslem and Lemeshow's goodness of fit test returns a p-value of 1, suggesting a good model.

# Appendix A: Data Sets

## Length vs Frequency in *War and Peace*.

| lengths | freqs |
|---|---|
| 4 | 98768 |
| 3 | 142072 |
| 7 | 42367 |
| 1 | 14898 |
| 6 | 48650 |
| 2 | 93537 |
| 5 | 58403 |
| 11 | 4436 |
| 8 | 29854 |
| 10 | 10703 |
| 9 | 17432 |
| 12 | 2646 |
| 13 | 1339 |
| 14 | 406 |
| 15 | 124 |
| 16 | 56 |
| 17 | 9 |
| 18 | 3 |

# Appendix B: Code

## Length vs Frequency code

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

public class LengthVsFrequency {

	public static void main(String[] args) throws Exception{
		@SuppressWarnings("resource")
		Scanner sc = new Scanner(new File("Document.txt"));		//input text
		ArrayList<Integer> lengths = new ArrayList<Integer>();
		ArrayList<Integer> freqs = new ArrayList<Integer>();

		while(sc.hasNext()){
			String s = sc.next();
			String[] arr;

			// Makes dashed words into separate words
			s = s.replaceAll("-", " ");
			// removes any non-word character
			s = s.replaceAll("[^a-zA-ZÀ-ÿ ]", "");

			//splits strings if they contain spaces (aka: if they had a dash)
			arr = s.split(" ");

			for (int i = 0; i < arr.length; i++){
				//gets individual strings (normally only one of two.)
				s = arr[i];
				if (!s.equals("")){
					int len = s.length();

					if (!lengths.contains(len) && len!=0){
						lengths.add(len);
						freqs.add(1);
					}
					else freqs.set(lengths.indexOf(len),
freqs.get(lengths.indexOf(len))+1);
				}
			}
		}
		toFile(lengths,freqs);
	}


	//write to csv file
	public static void toFile(ArrayList<Integer> lengths,ArrayList<Integer> freqs) throws
IOException {
		FileWriter fw = new FileWriter("LengthsVsFreqs.csv");
		fw.write("lengths,freqs\n");

		for (int i = 0; i < lengths.size(); i++)
			fw.write(lengths.get(i).toString() + "," + freqs.get(i).toString() +"\n");

		fw.close();
	}


	//print functions
	public static <E> void print(E str){
		System.out.print(str.toString());
	}
	public static <E> void println(E str){
		System.out.println(str.toString());
	}
	public static <E> void println(){
		System.out.println();
	}
}
```

## R code (raw)

```
> words=read.csv("~/Documents/CSI2110/Words/LengthsVsFreqs.csv")
> words=words[order(words$lengths),]
> words
   lengths  freqs
4        1  14898
6        2  93537
2        3 142072
1        4  98768
7        5  58403
5        6  48650
3        7  42367
9        8  29854
11       9  17432
10      10  10703
8       11   4436
12      12   2646
13      13   1339
14      14    406
15      15    124
16      16     56
17      17      9
18      18      3

> l=words[["lengths"]]
> f=words[["freqs"]]
> plot(f~l, main = "Lengths vs. Frequency")
> fit=lm(f~l)
> summary(fit)

Call:
lm(formula = f ~ l)

Residuals:
   Min     1Q Median     3Q    Max 
-66724 -13122  -3635  11406  72260 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)    87528      14097   6.209 1.25e-05 ***
l              -5905       1302  -4.534 0.000339 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 28670 on 16 degrees of freedom
Multiple R-squared:  0.5624,	Adjusted R-squared:  0.535 
F-statistic: 20.56 on 1 and 16 DF,  p-value: 0.0003386


> par(mfrow = c(2,2))
> plot(fit)
> anova(fit)
Analysis of Variance Table

Response: f
          Df     Sum Sq    Mean Sq F value    Pr(>F)    
l          1 1.6895e+10 1.6895e+10  20.561 0.0003386 ***
Residuals 16 1.3148e+10 8.2172e+08                      
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1


> plot(f~log(l), main = "Logarithmic model")
> plot(sqrt(f)~l, main = "Quadratic model")
> plot((1/f)~l, main = "Reciprocal model")
> plot(log(f)~log(l), main = "Power model")
> plot(log(f)~l, main = "Exponential model")
> expfit=lm(log(f)~l)
> summary(expfit)

Call:
lm(formula = log(f) ~ l)

Residuals:
    Min      1Q  Median      3Q     Max 
```

```
-3.4165 -0.4486  0.2788  1.0328  1.3494

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 13.59176    0.68692  19.786 1.13e-12 ***
l           -0.56629    0.06346  -8.924 1.31e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.397 on 16 degrees of freedom
Multiple R-squared:  0.8327,    Adjusted R-squared:  0.8222
F-statistic: 79.63 on 1 and 16 DF,  p-value: 1.307e-07


> plot(log(factorial(l)*f)~l, main="Poisson model")
> poissonlm=lm(log(factorial(l)*f)~l)
> summary(poissonlm)

Call:
lm(formula = log(factorial(l) * f) ~ l)

Residuals:
    Min      1Q  Median      3Q     Max
-0.5460 -0.2462 -0.0673  0.1775  0.7656

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  8.10162    0.18849   42.98   <2e-16 ***
l            1.63600    0.01741   93.95   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3833 on 16 degrees of freedom
Multiple R-squared:  0.9982,    Adjusted R-squared:  0.9981
F-statistic:  8827 on 1 and 16 DF,  p-value: < 2.2e-16

> anova(poissonlm)
Analysis of Variance Table

Response: log(factorial(l) * f)
          Df  Sum Sq Mean Sq F value    Pr(>F)
l          1 1296.77 1296.77  8826.8 < 2.2e-16 ***
Residuals 16    2.35    0.15
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> abline(poissonlm, col="red")


> newf=f*factorial(l)
> newf
 [1] 2.370432e+06 8.524320e+05 2.135297e+08 1.489800e+04 3.502800e+07
 [6] 1.870740e+05 7.008360e+06 1.770709e+11 1.203713e+09 3.883905e+10
[11] 6.325724e+09 1.267438e+12 8.337981e+12 3.539439e+13 1.621516e+14
[16] 1.171676e+15 3.201187e+15 1.920712e+16
> plot(newf~l)
> newfit=lm(log(newf)~l)
> summary(newfit)

Call:
lm(formula = log(newf) ~ l)

Residuals:
    Min      1Q  Median      3Q     Max
-0.5460 -0.2462 -0.0673  0.1775  0.7656

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  8.10162    0.18849   42.98   <2e-16 ***
l            1.63600    0.01741   93.95   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3833 on 16 degrees of freedom
Multiple R-squared:  0.9982,    Adjusted R-squared:  0.9981
F-statistic:  8827 on 1 and 16 DF,  p-value: < 2.2e-16
> plot((factorial(l)*f)~l)
> curve(exp(newfit$coef[1]+newfit$coef[2]*x), add=T, col="red")
```

```
> plot(f~l)
> curve(exp(newfit$coef[1]+newfit$coef[2]*x)/factorial(x), add=T, col="red")


> poissonfit=glm(f~l, family=quasipoisson(link=log))
> par(mfrow=c(2,2))
> plot(poissonfit)
> summary(poissonfit)

Call:
glm(formula = f ~ l, family = quasipoisson(link = log))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-390.16   -77.71   -46.89    49.74   214.76

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 11.95505    0.25027  47.769  < 2e-16 ***
l           -0.24012    0.04385  -5.476 5.07e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for quasipoisson family taken to be 13968.93)

    Null deviance: 885453  on 17  degrees of freedom
Residual deviance: 282572  on 16  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 5

> anova(poissonfit)
Analysis of Deviance Table

Model: quasipoisson, link: log

Response: f

Terms added sequentially (first to last)


     Df Deviance Resid. Df Resid. Dev
NULL                    17     885453
l     1   602881        16     282572

> install.packages("ResourceSelection")
Installing package into ‘/Users/ionabuchanan/Library/R/3.2/library’
(as ‘lib’ is unspecified)
also installing the dependency ‘pbapply’

trying URL 'https://muug.ca/mirror/cran/bin/macosx/mavericks/contrib/3.2/pbapply_1.3-1.tgz'
Content type 'application/x-gzip' length 39118 bytes (38 KB)
==================================================
downloaded 38 KB

trying URL 'https://muug.ca/mirror/cran/bin/macosx/mavericks/contrib/3.2/ResourceSelection_0.3-0.tgz'
Content type 'application/x-gzip' length 440881 bytes (430 KB)
==================================================
downloaded 430 KB


The downloaded binary packages are in
      /var/folders/tx/kfk1q6zn06z7ms9j94qq0t7c0000gn/T//RtmppuxQti/downloaded_packages
> library(ResourceSelection)
ResourceSelection 0.3-0   2016-11-04
Warning message:
package ‘ResourceSelection’ was built under R version 3.2.5
> hoslem.test(f, fitted(poissonfit))

      Hosmer and Lemeshow goodness of fit (GOF) test

data:  f, fitted(poissonfit)
X-squared = -8.3208, df = 8, p-value = 1
```