Assignment 1 Planets

Ben Lu - 558464180

2023-07-27

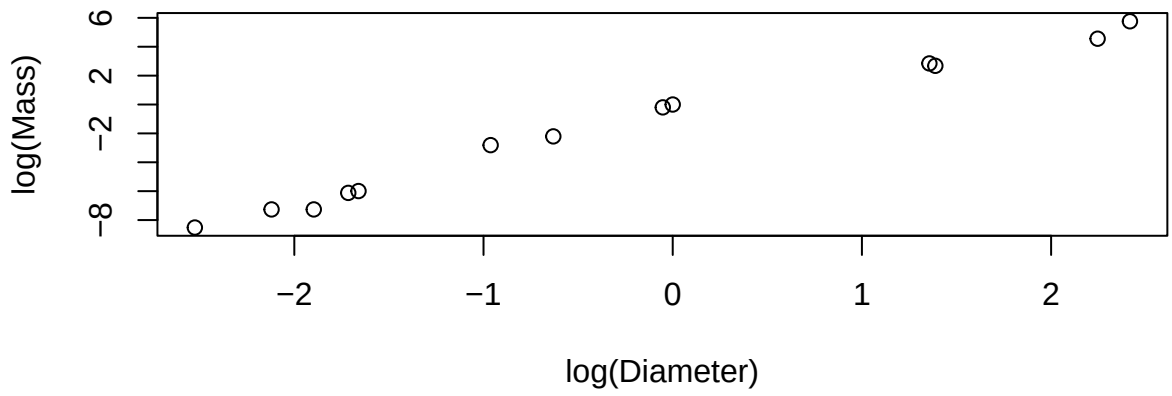
Question 1

a

We are interested in investigating how the mass of a planet is related to the cube of its diameter.

(i) Plot mass and diameter on a log-log scale

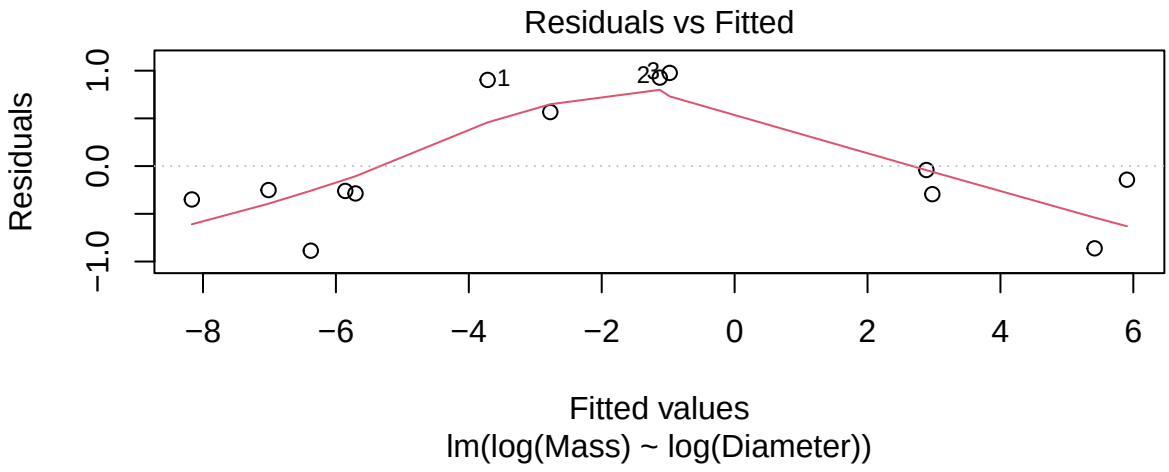
```
Moons.df <- read.csv("Moons.csv")
plot(log(Mass) ~ log(Diameter), data = Moons.df)
```



The data in our plot appears to be roughly linear, there is a positive relationship between log mass and log diameter.

(ii) Fit linear regression model

```
Moons.lm = lm(log(Mass) ~ log(Diameter), data = Moons.df)
plot(Moons.lm, which =1)
```




```
summary(Moons.lm)
```

```
##
## Call:
## lm(formula = log(Mass) ~ log(Diameter), data = Moons.df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.8864 -0.2943 -0.2511  0.5663  0.9767
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -0.9767     0.1890  -5.167  0.00031 ***
## log(Diameter)   2.8471     0.1134  25.117 4.59e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6689 on 11 degrees of freedom
## Multiple R-squared:  0.9829, Adjusted R-squared:  0.9813
## F-statistic: 630.8 on 1 and 11 DF,  p-value: 4.585e-11
```

```
confint(Moons.lm)
```

```
##                   2.5 %     97.5 %
## (Intercept)   -1.392724 -0.5606083
## log(Diameter)  2.597632  3.0966241
```

Our linear model describes a power law relationship between mass and diameter, and it explains 98% of the total variability in our data which is very accurate. Our residual plot shows a non-constant scatter of our residuals, however this is no concern as we have a very small sample size.

The model of our power relationship is:

$$log(Mass_i) = \beta_0 + \beta_1 \times log(Diameter_i) + \epsilon_i,$$

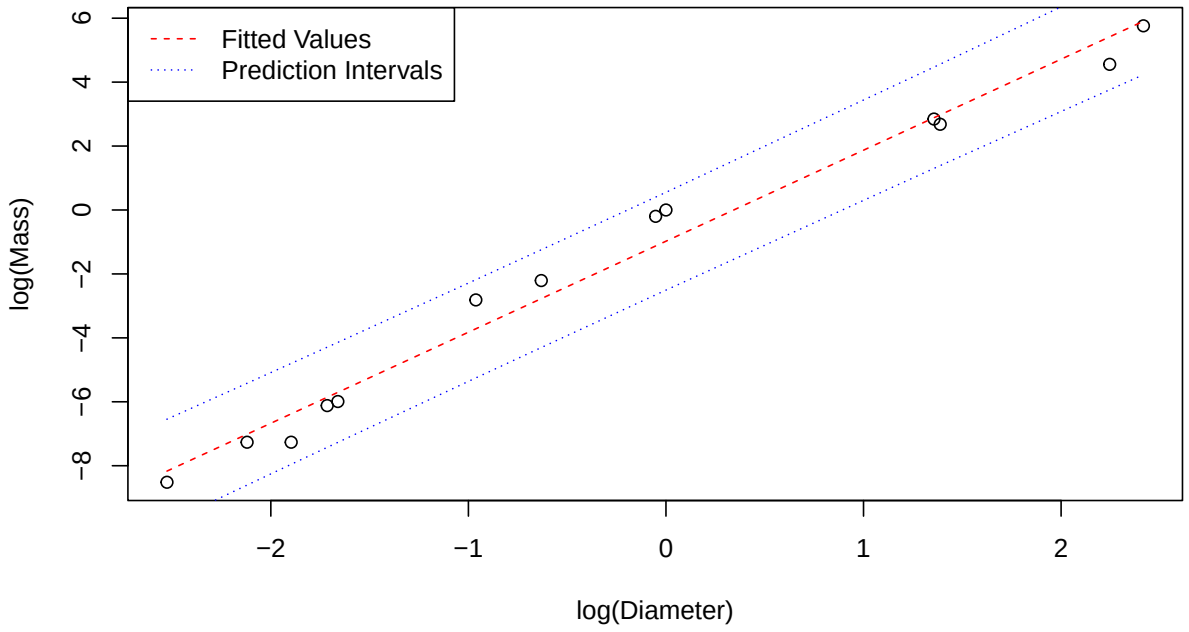
where $\epsilon_i \sim iid\ N(0, \sigma^2)$.

(iii) Superimpose the predicted values and their upper and lower 95% bounds on the plot above

```
x=sort(Moons.df$Diameter) # Sort diameter in ascending order
preds=predict(Moons.lm, newdata=data.frame(Diameter=x),interval = "prediction") # Generate prediction v
plot(log(Mass) ~ log(Diameter), data = Moons.df) # Plot the data prior to adding prediction intervals

# Superimpose the fitted values and prediction intervals on the plot
lines(log(x), preds[, "fit"], col = "red", lty = 2)  # Fitted values
lines(log(x), preds[, "lwr"], col = "blue", lty = 3) # Lower bound
lines(log(x), preds[, "upr"], col = "blue", lty = 3) # Upper bound

# Add a legend to the plot
legend("topleft", legend = c("Fitted Values", "Prediction Intervals"), col = c("red", "blue"), lty = c(
```



(iv) Test the hypthosis that planet mass is proportional to the cube of its diameter

```
b1.null = 3 # Hypothesis that beta 1 is 3
b1.estimate = coef(summary(Moons.lm))[2,1]
SE = coef(summary(Moons.lm))[2,2]
tstats = (b1.estimate - b1.null)/SE
# Calculate p value from two-tailed test
pvalue = 2 * (1 - pt(abs(tstats), 11))
pvalue
```

```
## [1] 0.2045774
```

We tested against the hypothesis that the power of the diameter is 3 using our estimate value of 2.85 and obtained a p-value of 0.2 which indicates no evidence against our initial assumption. Therefore, we accept

the hypothesis that the mass of the planet is proportional to the cube of its diameter. Our model estimate of 2.85 for the power of the diameter is reasonably adequate as it's close to the theorized value of 3.

b

We would like to verify whether the Physicist's claim that rocky planets have the highest density, followed by ice giants and then gas giants are true.

(i)

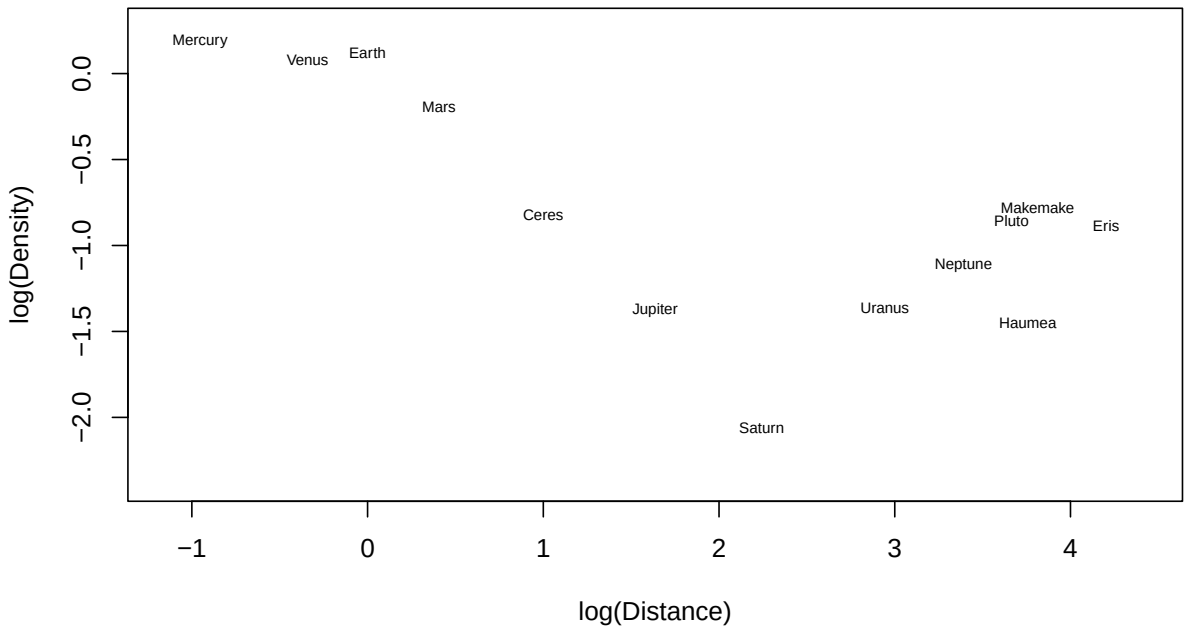
```
# Create density column
Moons.df$Density <- Moons.df$Mass / Moons.df$Diameter^3
```

(ii)

```
plot(log(Density) ~ log(Distance), data = Moons.df, col = colors, xlim = range(log(Moons.df$Distance)) +
```

```
## Warning in plot.xy(xy, type, ...): supplied color is neither numeric nor
## character
```

```
# Superimpose the names of the planets on the plot
text(log(Moons.df$Distance), log(Moons.df$Density), labels = Moons.df$Name, pos = 3, cex=0.6)
```



```
print(Moons.df)
```

```
##         Name Distance Diameter     Mass Moons   Density
## 1    Mercury      0.39    0.382   0.0600     0 1.0763690
## 2      Venus      0.72    0.949   0.8200     0 0.9594342
## 3      Earth      1.00    1.000   1.0000     1 1.0000000
## 4       Mars      1.52    0.532   0.1100     2 0.7305632
## 5      Ceres      2.75    0.080   0.0002     0 0.3906250
## 6    Jupiter      5.20   11.209 317.8000    64 0.2256593
## 7     Saturn      9.54    9.449  95.2000    62 0.1128442
## 8     Uranus     19.22    4.007  14.6000    27 0.2269315
## 9    Neptune     30.06    3.883  17.2000    13 0.2937827
## 10     Pluto     39.50    0.180   0.0022     4 0.3772291
## 11    Haumea     43.35    0.150   0.0007     2 0.2074074
## 12 Makemake     45.80    0.120   0.0007     0 0.4050926
## 13      Eris     67.70    0.190   0.0025     1 0.3644846
```

(iii)

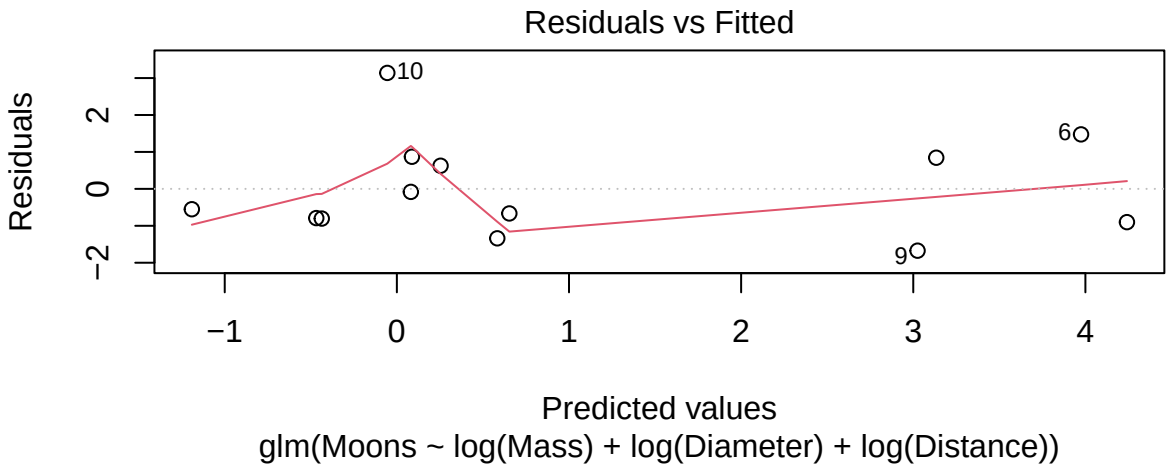
It appears that planets which are classified as rocky such as Mercury, Venus, Earth and Mars have the highest density in our data. In our next comparison between gas giants (Jupiter, Saturn) and ice giants (Uranus, Neptune), we observe on our plot that the density of Jupiter and Neptune appears to be the same, however on a closer inspection of our data we could confirm that the density of Uranus (0.227) is higher than Jupiter (0.226). We could also observe on our plot that the density of Neptune is higher than Saturn. We draw the same conclusion with the Physicist's claim that rocky giants have the highest density, followed by ice giants which has a slightly higher density than gas giants.

c

We are interested in predicting the mean number of moons for planets based on its' mass, diameter and distance.

(i)

```
Moons.glm1 = glm(Moons ~ log(Mass) + log(Diameter) + log(Distance), family = poisson, data = Moons.df)
plot(Moons.glm1, which = 1)
```




```
summary(Moons.glm1)
```

```
##
## Call:
## glm(formula = Moons ~ log(Mass) + log(Diameter) + log(Distance),
##     family = poisson, data = Moons.df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8933  -1.1191  -0.7332   0.7780   2.3294
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)     0.6536     0.3494   1.871   0.0614 .
## log(Mass)      -0.5162     0.2179  -2.369   0.0179 *
## log(Diameter)   2.5206     0.5728   4.400 1.08e-05 ***
## log(Distance)   0.1236     0.1144   1.080   0.2800
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 388.253  on 12  degrees of freedom
## Residual deviance:  20.435  on  9  degrees of freedom
## AIC: 62.451
##
## Number of Fisher Scoring iterations: 5
```

```
pvalue = 1 - pchisq(20.435, 9) #over dispersion check
pvalue
```

```
## [1] 0.01540965
```

We initially fitted a Poisson GLM with three explanatory variables of logged Mass, Diameter and Distance to predict the log mean number of moons. It appears that for every 1 unit increase in:

```
log(Mass) decreases the log mean number of moons by 0.516

log(Diameter) increases the log mean number of moons by 2.521

log(Distance) increases the log mean number of moons by 0.124
```

Our initial equation is:

$$log(\mu_i) = \beta_0 + \beta_1 \times log(Mass_i) + \beta_2 \times log(Diameter_i) + \beta_3 \times log(Distance_i)$$

(ii)

The residual plot shows a trend but there is no concern as we have a small sample size, however when checking for over-dispersion, we obtained a very small p-value of 0.015 which indicates evidence of over-dispersion. This means our initial GLM fit using Poisson distribution was insufficient and we must change to QuasiPoisson for our intermediate model.

```
Moons.glm2 = glm(Moons ~ log(Mass) + log(Diameter) + log(Distance), family = quasipoisson, data = Moons
summary(Moons.glm2)
```

```
##
## Call:
## glm(formula = Moons ~ log(Mass) + log(Diameter) + log(Distance),
##     family = quasipoisson, data = Moons.df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8933  -1.1191  -0.7332   0.7780   2.3294
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     0.6536     0.5378   1.215   0.2552
## log(Mass)      -0.5162     0.3355  -1.539   0.1583
## log(Diameter)   2.5206     0.8818   2.859   0.0188 *
## log(Distance)   0.1236     0.1761   0.702   0.5005
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 2.369642)
##
##     Null deviance: 388.253  on 12  degrees of freedom
## Residual deviance:  20.435  on  9  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 5
```

For our intermediate model, it appears log(Mass) has a p-value of 0.16 and log(Distance) has a p-value of 0.5, which indicates their effects were statistically insignificant. Since log(Distance) was the most statistically insignificant, we removed this variable for our next analysis.

```
Moons.glm3 = glm(Moons ~ log(Mass) + log(Diameter), family = quasipoisson, data = Moons.df)
summary(Moons.glm3)
```

```
##
## Call:
## glm(formula = Moons ~ log(Mass) + log(Diameter), family = quasipoisson,
##     data = Moons.df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1557  -1.1371  -0.9279   0.7344   2.3666
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     0.8647     0.3883   2.227  0.05011 .
## log(Mass)      -0.6259     0.2833  -2.209  0.05163 .
## log(Diameter)   2.7867     0.7568   3.682  0.00423 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 2.198265)
##
##     Null deviance: 388.253  on 12  degrees of freedom
## Residual deviance:  21.621  on 10  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 5
```

We obtained a p-value of 0.0516 for log(Mass) which is rounded to 0.05, hence the effects of log(Mass) were significant and was retained for our final model.

```
Moons.glm4 = glm(Moons ~ log(Diameter) + log(Mass), family = quasipoisson, data = Moons.df)
summary(Moons.glm4)
```

```
##
## Call:
## glm(formula = Moons ~ log(Diameter) + log(Mass), family = quasipoisson,
##     data = Moons.df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1557  -1.1371  -0.9279   0.7344   2.3666
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     0.8647     0.3883   2.227  0.05011 .
## log(Diameter)   2.7867     0.7568   3.682  0.00423 **
## log(Mass)      -0.6259     0.2833  -2.209  0.05163 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 2.198265)
##
##     Null deviance: 388.253  on 12  degrees of freedom
## Residual deviance:  21.621  on 10  degrees of freedom
## AIC: NA
```

```
##
## Number of Fisher Scoring iterations: 5
```

(iii)

Our final model is:

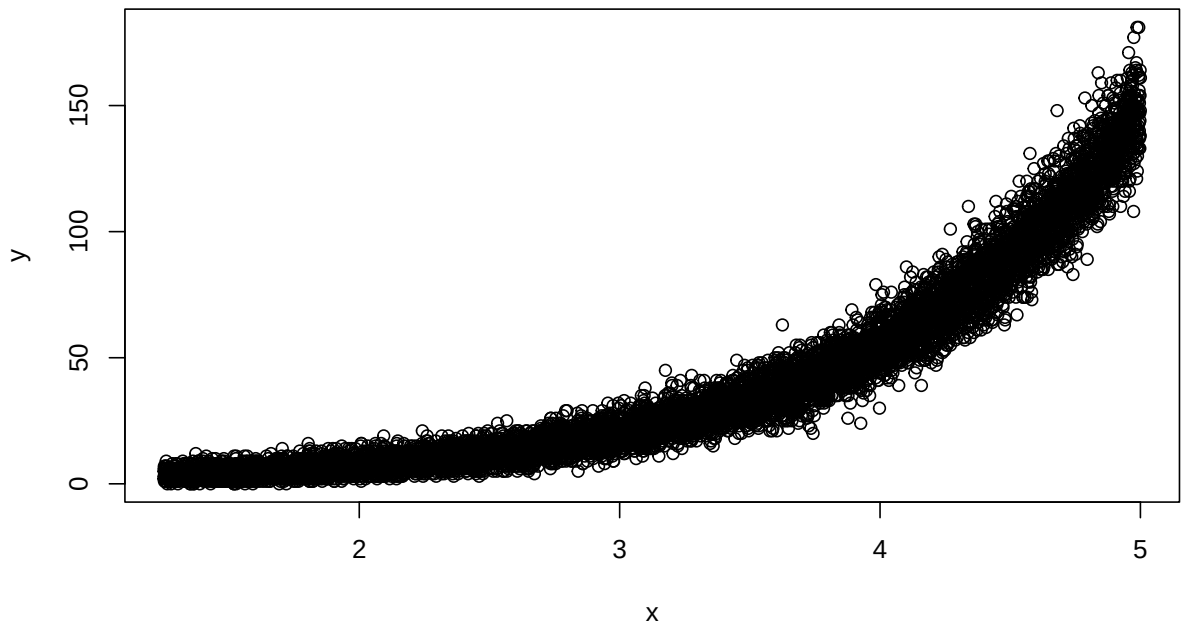
$$log(\mu_i) = \beta_0 + \beta_1 \times log(Mass_i) + \beta_2 \times log(Diameter_i)$$

Where μ_i is the count of the number moons, $Mass_i$ is the mass of the planet relative to Earth's and $Diameter_i$ is the diameter of the planet relative to Earth's.

Question 2

a

```
set.seed(123)
nx=1e4
x=seq(1.25,5, length=nx)
y=rpois(nx, lambda=exp(0+1*x))
plot(y~x)
```



set.seed(123) - sets the random number generator's seed to 123 which ensures we get the same random numbers each time we run the code.

nx=1e4 - Assigns value of 10,000 to the variable nx.

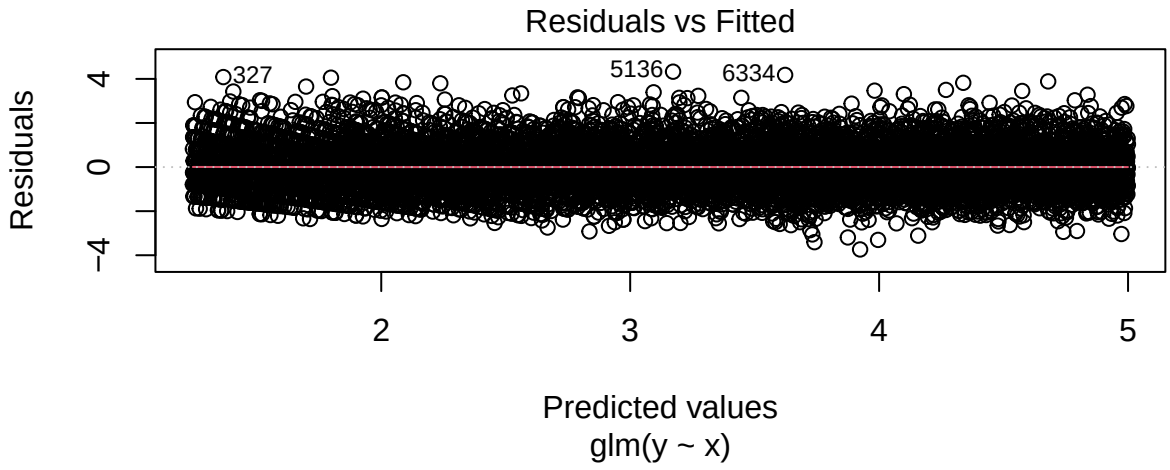
x=seq(1.25,5, length=nx) - seq() function generates a sequence of numbers. This generates a sequence of numbers from 1.25 to 5 with 10,000 equally spaced values, which is stored in variable x.

y=rpois(nx, lambda=exp(0+1*x)) - This function generates 10,000 random numbers from the Poisson distribution. The parameter taken by lambda generates y values that increases exponentially as x increases, this creates an exponential curve when y is plotted against x.

plot(y~x) - This creates a scatter plot of y against x.

(b)

```
simulation.glm = glm(y~x, family= "poisson")
plot(simulation.glm, which = 1)
```




```
confint(simulation.glm)
```

```
## Waiting for profiling to be done...
```

```
##                   2.5 %      97.5 %
## (Intercept) -0.02524899 0.007345989
## x            0.99779861 1.005614175
```

The plot of our residuals appears to be equally distributed, with the trend of our residuals being centered at 0 which indicates no concerns about our GLM.

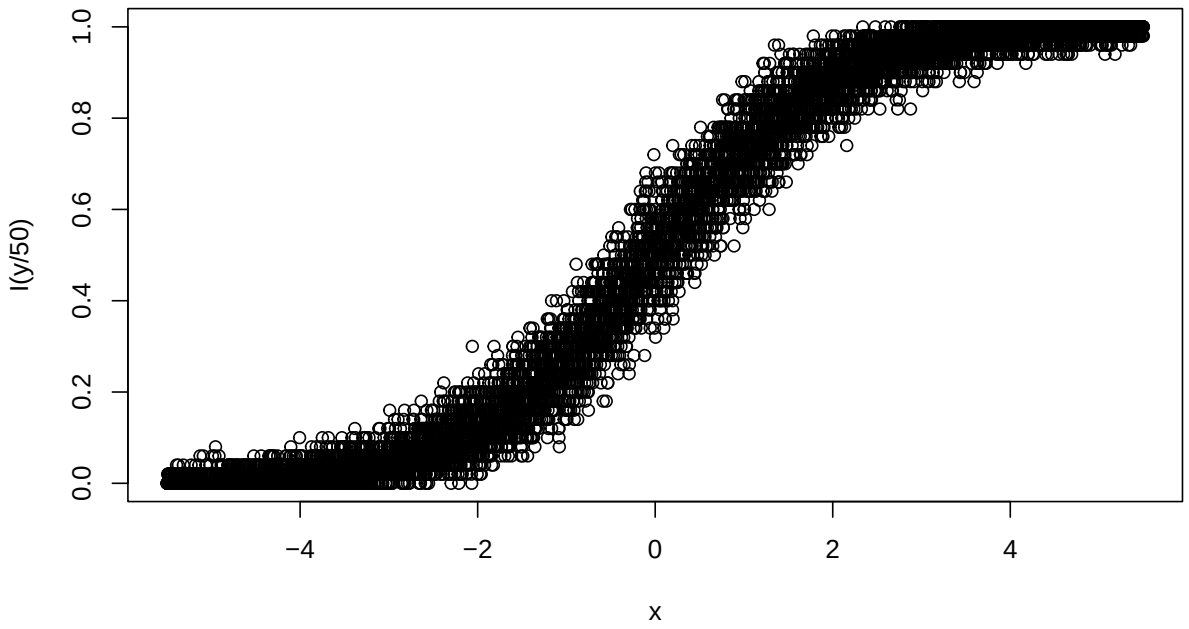
We estimate with 95% confidence on average that our y-intercept is between -0.025 and 0.007 which seems reasonable because we expect the y-intercept from our simulation data to be 0 after taking the logarithm of lambda.

We estimate with 95% confidence on average that the slope of every increase in x results in between 0.998 and 1.006 increase in y which also makes sense as we expect the slope from our simulation data to be 1 after taking the logarithm of lambda. It is evident that using Poisson family from our GLM has converted the multiplicative effects from our simulation into additive effects which gives us a linear slope.

Question 3

a

```
set.seed(345)
x=seq(-5.5, 5.5, length=nx)
y=rbinom(n=nx,size=50, prob=exp(0+1*x)/(1+exp(0+1*x)))
plot(I(y/50)~x)
```




```
ymod = y/50
```

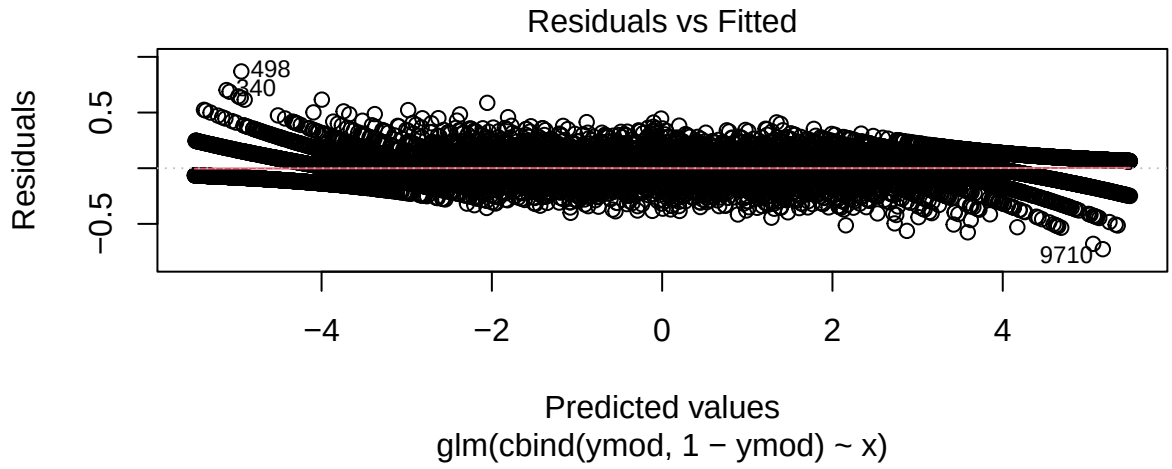
We observe an S shape of our plotted values which is a characteristic of binomial distribution, it appears that the probability success increases as x increases.

```
simulation.glm2 = glm(cbind(ymod, 1-ymod)~x, family="binomial")
summary(simulation.glm2)
```

```
##
## Call:
## glm(formula = cbind(ymod, 1 - ymod) ~ x, family = "binomial")
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.48971  -0.10891  -0.00019   0.10817   0.50191
##
```

```
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 0.002769   0.033278   0.083    0.934
## x           0.998401   0.019312  51.698   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8187.02  on 9999  degrees of freedom
## Residual deviance:  202.42  on 9998  degrees of freedom
## AIC: 3054.2
##
## Number of Fisher Scoring iterations: 6
```

```
plot(simulation.glm2, which = 1)
```




```
confint(simulation.glm2)
```

```
## Waiting for profiling to be done...

##                   2.5 %     97.5 %
## (Intercept) -0.06246458 0.06800544
## x            0.96115049 1.03686686
```

The trend of our residuals plot is centered at 0 which indicates no concerns about our GLM using binomial family.

We estimate with 95% confidence on average that our y-intercept is between -0.063 and 0.068 which seems reasonable as our expected y-intercept of 0 falls within this range.

We estimate with 95% confidence on average that the slope of every increase in x also increases the log-odds of success by between 0.96 and 1.1, this also seems reasonable as our expected slope of 1 falls within this range. Our GLM using binomial family has logged the data from our simulation, converted the S-shape curve into a linear relationship between log-odds of success and x.