# Chapter 13: Modelling count data using the Poisson distribution

STATS 201/8

University of Auckland

# Learning Outcomes

In this chapter you will learn about:

- The nature of count data
- Example - modelling the number of R packages
- The Poisson Distribution
- The generalized linear model (GLM)
- GLM re-analysis of the example
- The quasi-Poisson correction
- Interpretting the fitted GLM
- Relevant R-code.

# Section 13.1
# The nature of count data

# The nature of count data

In many studies the **response** variable will be a count.

A count variable is one where the measurement is the count of the number of times some event occurs. Obviously, it can only take values that belong to the non-negative integers $\{0,1,2,3...\}$.

In statistical parlance, a count variable is an example of a *discrete* variable, since the values it can take are discretely separated.

(In contrast, a variable with a Normal Distribution is an example of a *continuous variable*, since if can take any value on a continuum.)

# The nature of count data...

In this course we shall encounter three types of count data:

1. Counts of the number of "events" occurring, where ideally, the events occur independently of one another and with no specific upper limit on the maximum number.
2. Counts of the number of "successes"[1] from a fixed number of trials. E.g., the number of Heads from tossing a coin 10 times.
3. Counts of the number of items in a category. E.g., The count of A, B, and C grades in the course.

In the Chapter we focus on the first of the above types. Some examples are given below.

[1] Here, "successes" is used generically. E.g., It could be the number of females in a class of 146.

# The nature of count data...

## Some examples of counting the number of events occurring

- The success of a marketing campaign can be assessed by analysing the change in number of purchases.
- In epidemiology, the spread of an infectious disease can be modelled by observing the number of those infected each day.
- In operations research, a business can run more efficiently by being able to predict the number of customer interactions in a day, and hence the number of staff required.
- In ecology, the success of habitat restoration can be assessed by gathering counts of the number of animals before and after the restoration.
- The efficacy of fluoridation can be assessed by counting the number of tooth cavities in the mouths of subjects.

**Question:** For which of the above scenarios would it be reasonable to assume that the events being counted are occurring independently?

# Analysis of count data

## Why do we care?

In some situations linear regression can be successfully used to analyse count data. In fact, you may have already seen this done in this course. However, these are special cases.[2]

**Linear regression is not generally applicable to count data.** This is demonstrated in the example below.

[2] We will soon see an example where an inappropriate application of a linear regression model to count data was responsible for the most expensive human mistake in history (up to that point in time).

# Section 13.2
# Example: Number of R packages submitted to the Comprehensive R Archive Network (CRAN)

# Submissions to CRAN

In the following example we will model a count variable (number of R packages submitted to the CRAN over the years) using what we know to date: the *linear model* via the function `lm`.

We will contrast this to a more general and more appropriate way of modelling counts: the *generalized linear model* (GLM) via the function `glm`.

Along the way you will be introduced to a new distribution, the Poisson distribution.

# Submissions to CRAN...

As R is an open-source programming environment, people are encouraged to submit software packages used in their research for others to access via the CRAN[3], which was created in 2005. We would like to understand the nature of its growth. With this in mind, the number of new packages submitted in each year, from 2005 to 2016, has been recorded:

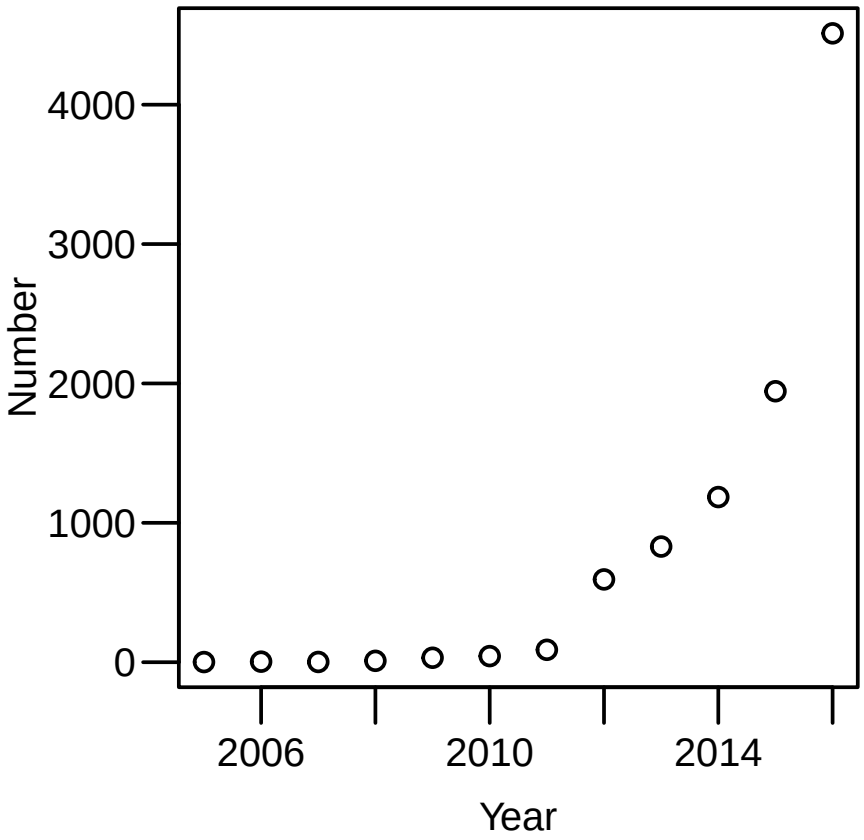
```
> CRAN.df = read.table("Data/CRAN.txt", header=T)
> CRAN.df
   Year Number
1  2005      1
2  2006      4
3  2007      2
4  2008     10
5  2009     32
6  2010     44
7  2011     89
8  2012    594
9  2013    830
10 2014   1185
11 2015   1944
12 2016   4512
```

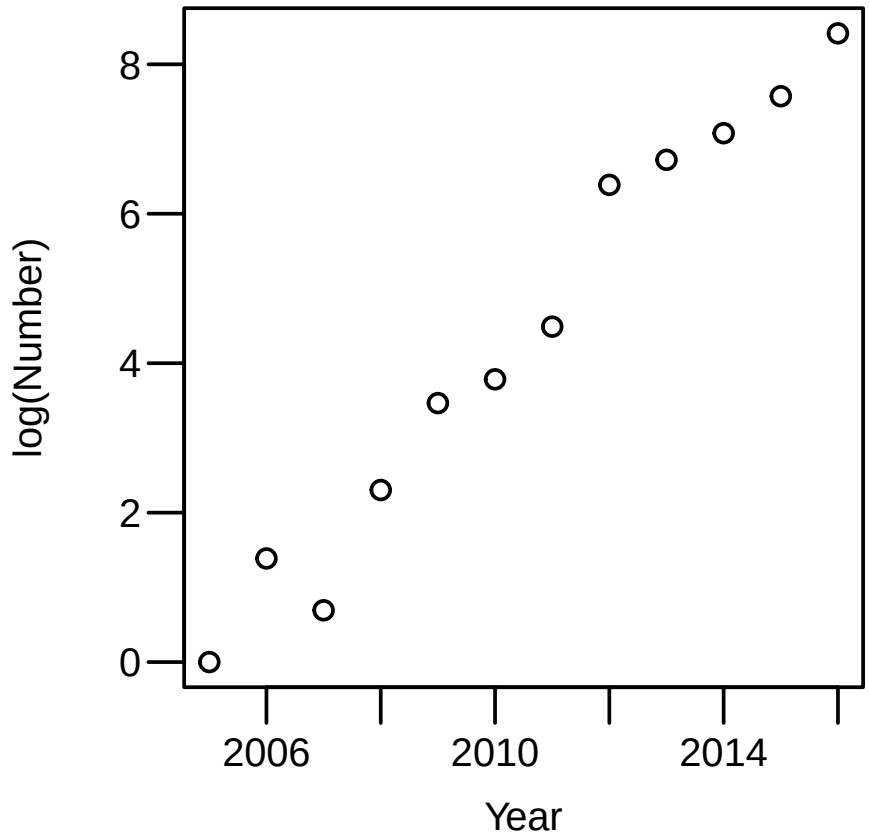
[3]You probably downloaded your R installation from the CRAN.

# Submissions to CRAN...

## Plotting the data

```
> ## One-by-two figure layout
> par(mfrow=c(1,2))
> ## Scatter plot using raw y
> plot(Number ~ Year, data = CRAN.df)
> ## Scatter plot using log y
> plot(log(Number) ~ Year, data = CRAN.df)
```



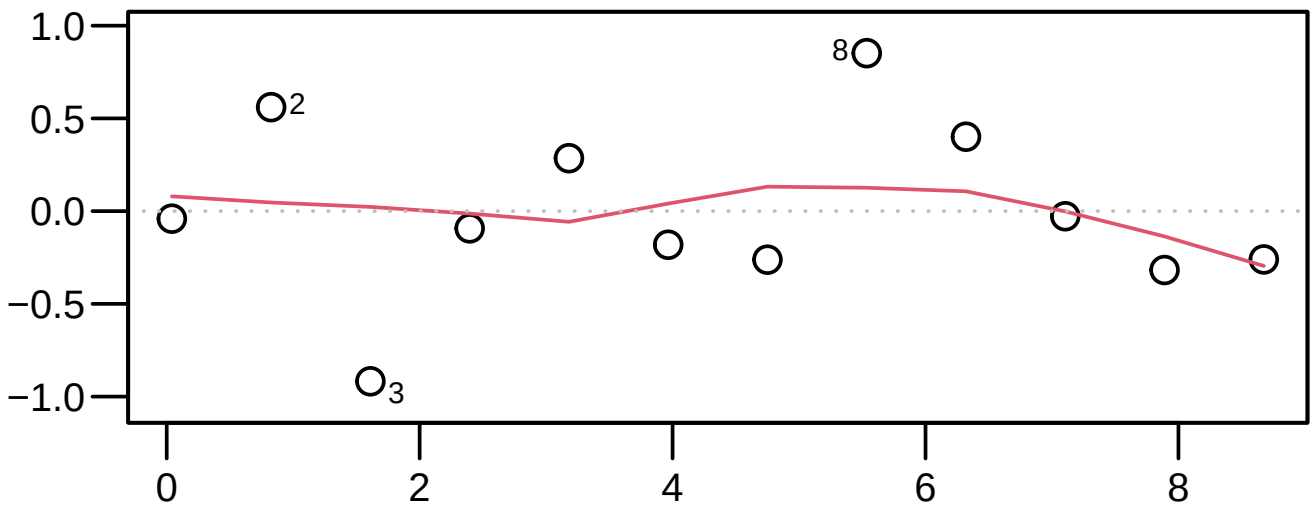



# Submissions to CRAN...

## Analysis via `lm`

The relationship between year and number of submissions looks reasonably linear on the log scale, so we'll fit a linear model to $\log(Y)$.

```
> CRAN.fit = lm(log(Number) ~ Year, data = CRAN.df)
> plot(CRAN.fit,which=1)
```

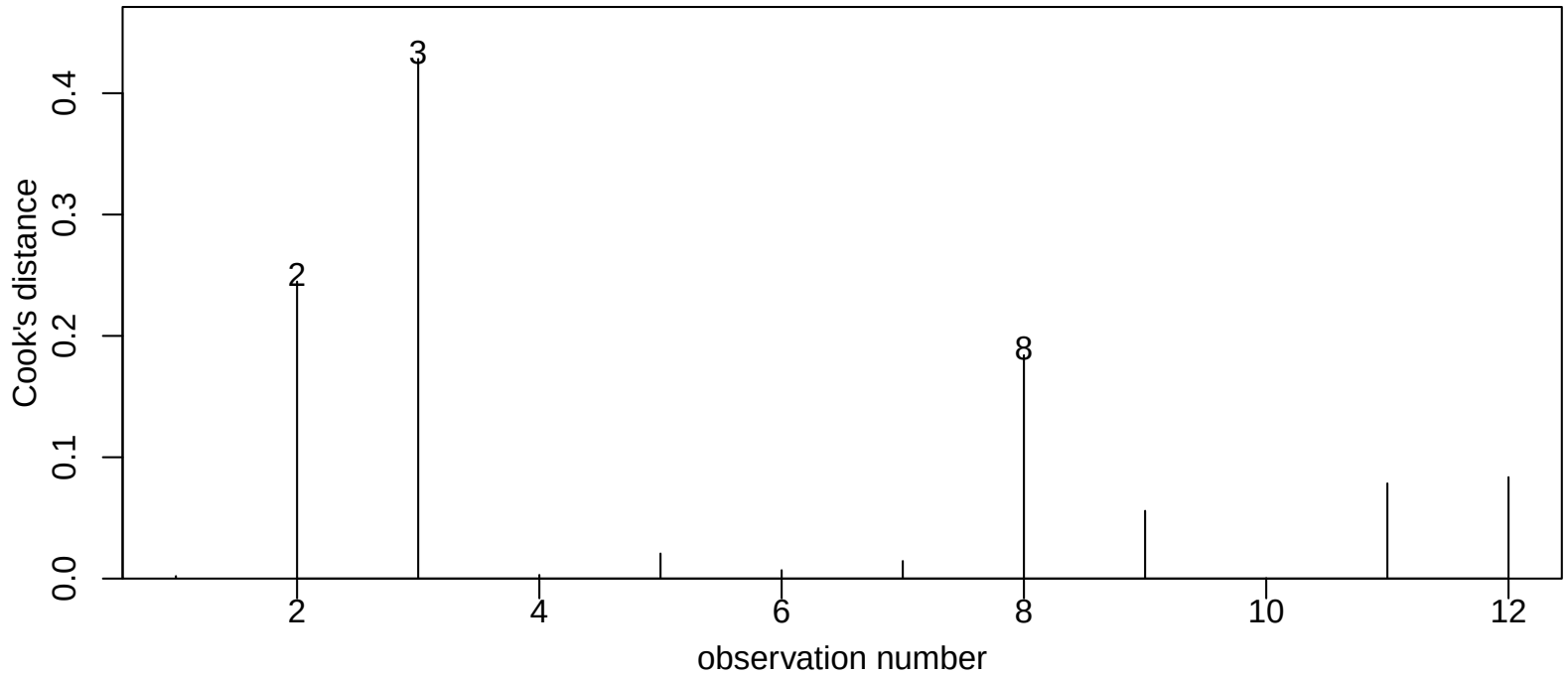



No concerns.

# Submissions to CRAN...

## Analysis via `lm`

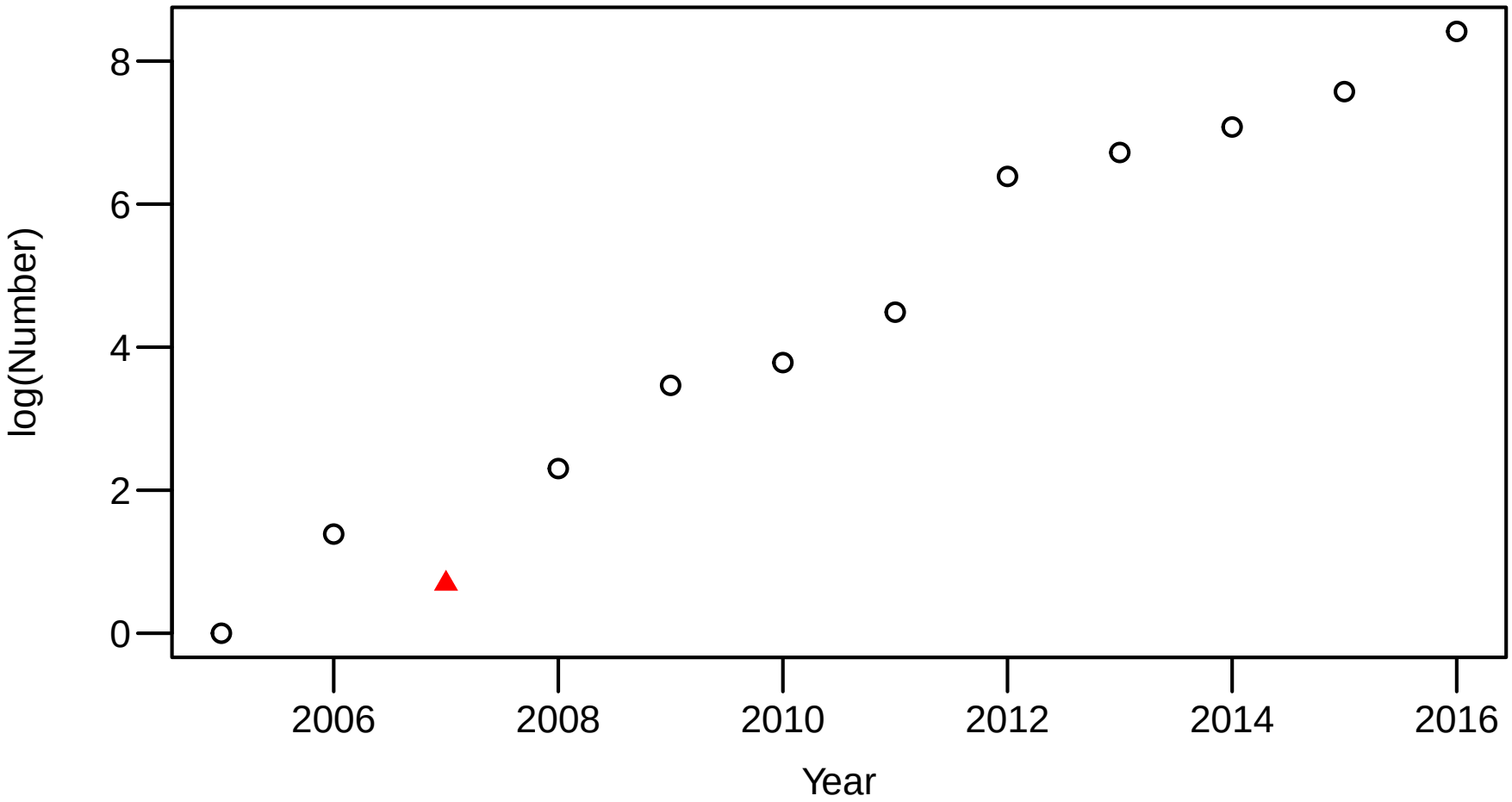
```
> cooks20x(CRAN.fit)
```




The Cook's distance for observation 3 exceeds our threshold of 0.4. However, with only 12 observations this is perhaps not of great concern.

# Submissions to CRAN...

## Analysis via `lm`...




Indeed, the third observation in red does not seem to be influential in the exploratory plots. So, we will keep it in the model.

# Submissions to CRAN...

## Analysis via `lm`...

```
> summary(CRAN.fit)
```

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.574e+03  8.245e+01  -19.09 3.39e-09 ***
Year         7.849e-01  4.101e-02   19.14 3.30e-09 ***
---
Residual standard error: 0.4904 on 10 degrees of freedom
Multiple R-squared:  0.9734,Adjusted R-squared:  0.9708
F-statistic: 366.4 on 1 and 10 DF,  p-value: 3.295e-09
```

# Submissions to CRAN...

## Analysis via `lm`...

Back-transform to get the multiplicative effect of year.

```
> ## Estimated annual multiplier
> exp(CRAN.fit$coef["Year"])
    Year
2.192237
> ## Confidence interval
> exp(confint(CRAN.fit))
                2.5 %  97.5 %
(Intercept) 0.00000 0.00000
Year        2.00081 2.40198
```

So, the Executive Summary would have said that the median annual number of submissions to CRAN multiplies by between 2.00 to 2.40 times each year. In other words, it increases by between 100% and 140% per annum.

# Submissions to CRAN...

## Analysis via `lm`...

We will use this model to estimate the median of the distribution of the number of submissions in 2017.

```
> predCRAN.df = data.frame(Year = 2017)
> pred2017 = predict(CRAN.fit, predCRAN.df, interval = "confidence")
>
> ## Prediction on the log scale
> pred2017
      fit     lwr      upr
1 9.45978 8.78731 10.13225
> ## Back-transform for the median of the number of submissions in 2017
> exp(pred2017)
       fit      lwr      upr
1 12833.05 6550.586 25140.85
```

So, the Executive Summary would have said that the median of the number of submissions to CRAN in 2017 is between 6550 and 25100.

# Submissions to CRAN...

## Comments on the analysis via `lm`...

There is a well known saying:

"If you only have a hammer, you tend to see every problem as a nail." – Abraham Maslow [Replace "hammer" with "linear model", and "a nail" with "normally distributed"]

We were able to do an analysis with `lm` (our hammer) by working with the logged data (our normally distributed nails). But, was this really a sensible way to analyse the CRAN data?

Before we answer this question, we'll look at a different methodology that is tailored for the analysis of count data. The first thing we need is a more appropriate type of distribution to replace the normal distribution. One that is tailored for describing the distribution of count data.

# Section 13.3
# The Poisson Distribution

# The Poisson distribution

Statisticians often use the Poisson distribution to describe the behaviour of counts of events that occur randomly in space or time, such as:

- The number of alcohol related arrivals at a hospital.
- The number of dolphins in a pod.
- The number of pods of dolphins sighted during an aerial survey.
- The annual number of road fatalities.
- The annual number of fatal road accidents.

The Poisson is a distribution that takes values on the non-negative integers $\{0,1,2,3,...\}$ and it has no upper limit.

The Poisson distribution is specified by a single parameter, its mean (i.e., expected value). We will write, Pois($\mu$) to denote a Poisson distribution with mean of $\mu$.

# The Poisson distribution...

The probability that the non-negative integer value $y$ will be observed if generated by a Pois($\mu$) distribution is given by the following formula:

$$\Pr(y) = \frac{\exp(-\mu)\mu^y}{y!} \ .$$

where $y! = \texttt{factorial(y)} = 1 \times 2 \times ... \times (y-1) \times y$ (and $0! = 1$).

For $y = 12$ and $\mu = 9.61$, this could be calculated in R using the code

```
> y=12; mu=9.61
> (exp(-mu)*mu^y)/factorial(y)
[1] 0.08685078
```

# The Poisson distribution...

In R, the in-built function dpois(y, $\mu$) calculates these Poisson probabilities. E.g., the probability that $y = 12$ will be observed from a Pois(9.61) distribution is

```
> dpois(12,9.61)
[1] 0.08685078
```

You can also generate random Poisson values. E.g., here are 20 random values from a Pois(10) distribution,

```
> rpois(20,10)
 [1]  2  6 12 12 11 11 11  7 11  9  8 10 11 11 11 11 14 13  5 16
```

# The Poisson distribution...

## An example

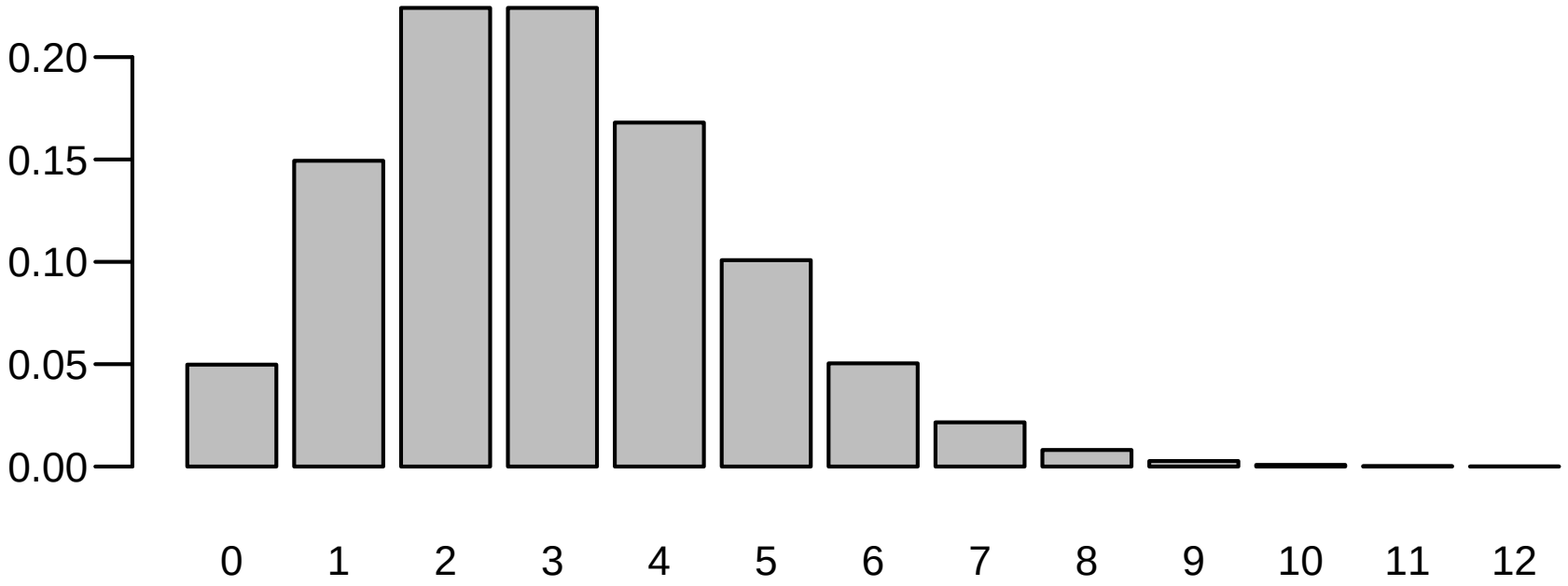
Suppose that a hospital *expects* to handle 3 victims of alcohol related mis-adventure on a Friday night.

Assuming that the *actual* number on any given Friday night can be described by a Poisson distribution with a mean of 3 (Pois(3)), what does the distribution of the number of alchohol victims look like?

# The Poisson distribution...

## An example...

```
> barplot(dpois(0:12,3),names=0:12)
```



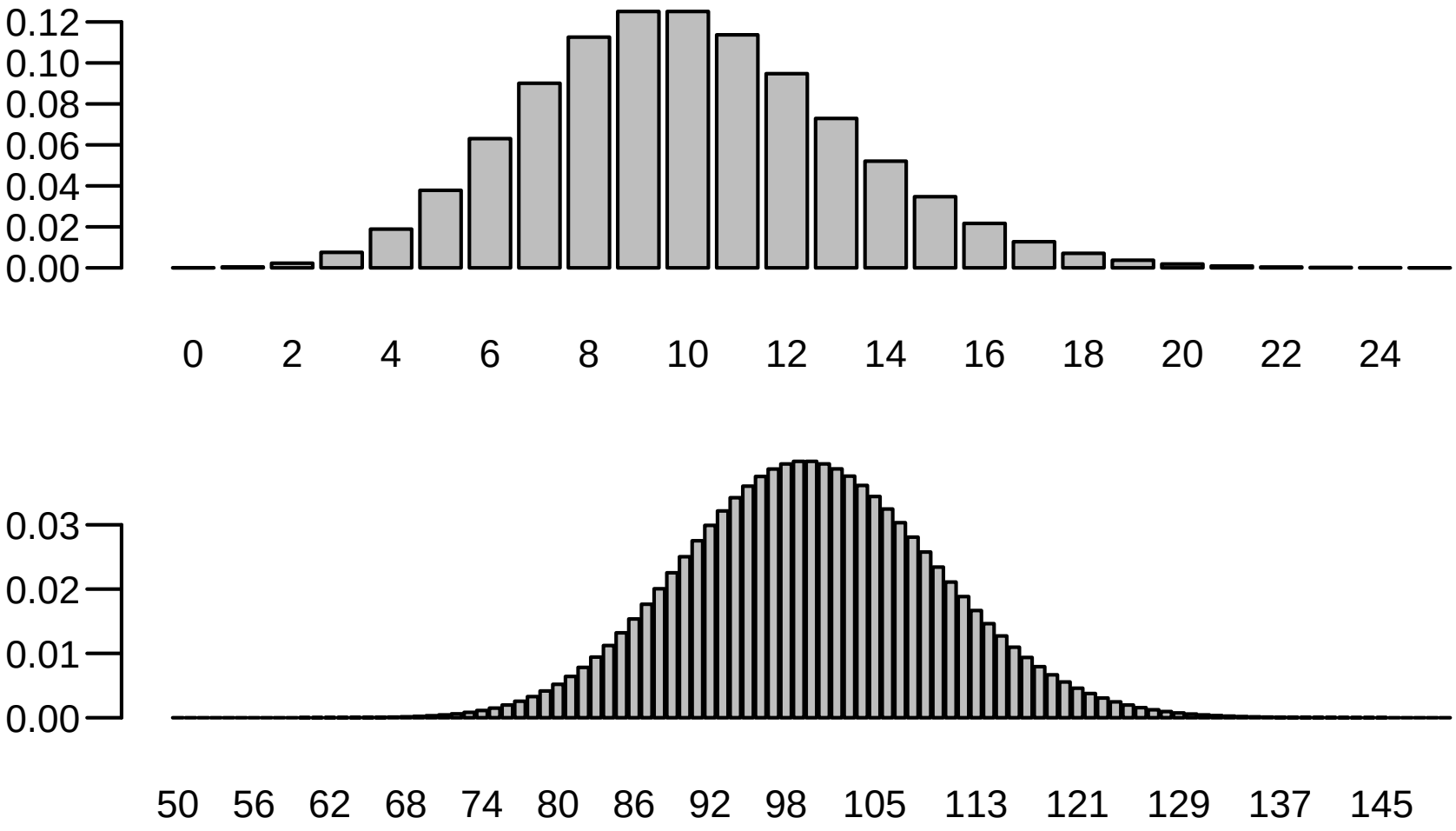

The probabilities for the number of victims from 0 to 20 are:

```
> round(dpois(0:20,3),6)
 [1] 0.049787 0.149361 0.224042 0.224042 0.168031 0.100819 0.050409 0.021604
 [9] 0.008102 0.002701 0.000810 0.000221 0.000055 0.000013 0.000003 0.000001
[17] 0.000000 0.000000 0.000000 0.000000 0.000000
```

# The Poisson distribution...

## More Poisson distributions

```
> par(mfrow=c(2,1)) ## Two-by-One figure layout
> barplot(dpois(0:25,10),names=0:25) ## Pois(10)
> barplot(dpois(50:150,100),names=50:150,las=1) ## Pois(100)
```



# The Poisson distribution...

## Properties of the Poisson distribution

- Variability increases with the mean $\mu$. In fact, the variance of a Pois($\mu$) is also $\mu$. That is, if $Y$ is Pois($\mu$) distributed then

$$\text{Var}(Y) = \text{E}(Y) = \mu$$

- The Poisson distribution is right-skewed for small values of $\mu$, but looks very much like a (discretised) normal distribution for large $\mu$.

# The Poisson distribution...

## Properties of the Poisson distribution...

The Poisson distribution is a good distribution to describe count data provided that the events being counted are independent.

- The number of alcohol related arrivals at a hospital could be described by a Poisson distribution provided that the arrivals occur independently...does this seem reasonable to you???
- What about the number of dolphins in a pod?
- What about the number of pods of dolphins sighted during an aerial survey?
- What about annual number of road fatalities?
- What about annual number of fatal road accidents?

- Would it be sensible for an insurance company to assume that the number of earthquake damage claims was Poisson distributed?

# Section 13.4
# The generalized linear model (GLM)

# The generalized linear model

Count data are not normally distributed, so we need to use a model that replaces the normality assumption with something more reasonable, such as assuming the data are Poisson distributed.

We will use a class of models called generalized linear models. They generalize the standard linear model (for normal data) by making it applicable to other types of data.

The linear model of Chaps 1-12 is a special case of a GLM.

# The Poisson regression GLM for the CRAN data

We will redo this analysis using a GLM. That is, the model will now assume that the number of packages in a given year is a Poisson random variable:

$$Y \sim \text{Poisson}(\mu)$$

where the Poisson parameter ($\mu = E[Y]$) changes with respect to an $x$ variable as follows:

$$\mu = \exp(\beta_0 + \beta_1 x).$$

Here, $x$ is year and $Y$ is the number of CRAN submissions in that year.

Since $\beta_0$ and $\beta_1$ can be negative or positive then so can $\beta_0 + \beta_1 x$, but $\mu = \exp(\beta_0 + \beta_1 x) > 0$, as required.

The relationship $\mu = \exp(\beta_0 + \beta_1 x)$ can equivalently be expressed as

$$\log(\mu) = \log E[Y|x] = \beta_0 + \beta_1 x$$

and so sometimes people call this log-linear modelling.

# Fitting a generalized linear model in R

## glm function

Fitting a generalized linear model is the easy part. We simply use the `glm` function instead of `lm`, and instruct `glm` that the response variable is Poisson distributed by giving it the `family = poisson` argument.

**NOTE:** Although only a simple change to the `R` code is required, the methodology "under the hood" is very different. It is no longer based on sums of squares, and instead uses a technique called maximum likelihood.

Maximum likelihood is the most widely used tool in statistical estimation. Linear regression (of normally distributed data) is a special case of maximum likelihood.

For more on maximum likelihood, see STATS 310, 330 and 730.

# Submissions to CRAN using Poisson regression

## Analysis via glm

```
> CRAN.gfit = glm(Number ~ Year, family = poisson, data = CRAN.df)
> summary(CRAN.gfit)
```

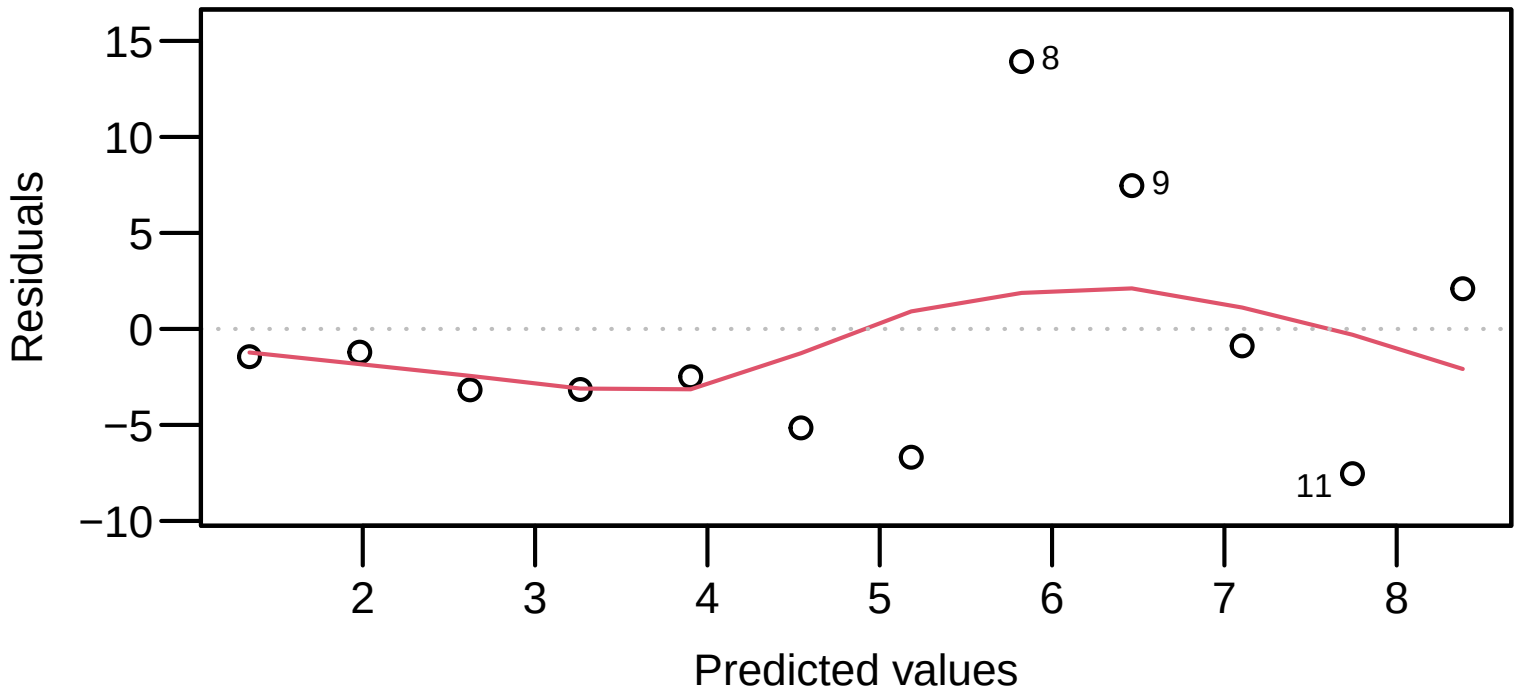
```
Call:
glm(formula = Number ~ Year, family = poisson, data = CRAN.df)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.282e+03  1.384e+01  -92.64   <2e-16 ***
Year         6.401e-01  6.868e-03   93.20   <2e-16 ***
---
(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 19374.51  on 11  degrees of freedom
Residual deviance:   402.61  on 10  degrees of freedom
```

# Submissions to CRAN using Poisson regression...

## Assumption checks

First, check the residuals to see if they look random.

```
> plot(CRAN.gfit, which = 1)
```




Hmmm, a strange pattern, though no clear systematic trend in the residuals.

# Submissions to CRAN using Poisson regression...

## Assumption checks

It is not as easy to check the assumptions of a GLM compared to a linear model.

In the plot of residuals vs predicted values:

- The predicted values on the x-axis are values of the so-called "linear predictor". That is, they are the fitted values $\hat{\beta}_0 + \hat{\beta}_1 x$.
- The residuals on the y-axis are not standard residuals. They are "standardized residuals" and should be approximately normally distributed with mean of 0 and variance of 1 if the Poisson assumption is satisfied **and** $\mu > 5$ – see STATS 330 for more information. Clearly, the Poisson assumption is not valid here. We shall see a cure for this below.

STATS 330 covers checking of GLMs in greater detail. In particular, it demonstrates the use of simulations to show us when a model is violating assumptions.

# Submissions to CRAN using Poisson regression...

## Assumption checks...

There is another **very important check** that is essential:

- Checking the Poisson assumption that the variances of the counts are equal to their means.

If the model is correct, then the residual deviance (printed near the bottom of the `summary` output) is approximately distributed as chi-squared $\chi^2_q$ where $q$ is the degrees of freedom (number of observations minus the number of parameters estimated).

If the residual deviance is unreasonably large then we have evidence that the model is not valid.

In this example, the residual deviance is 402.61, with 10 df. The $P$-value is

```
> 1 - pchisq(402.61,10)
[1] 0
```

If this $P$-value is small then we conclude that our model is not adequate. That is clearly the case with the CRAN data.

Fortunately there is a very simple fix.

# Submissions to CRAN using Poisson regression...

## quasi-Poisson correction

If the model is found to be inadequate, there are two possible reasons:

- We do not have the right explanatory terms in the model.
- The Poisson assumption is wrong.

If the residual plot looks OK, then we can rule out the first possibility. In this example, it looks like we have a problem with the Poisson assumption.

The fix is very simple – we just replace `family = poisson` with `family = quasipoisson`.[4]

[4] Those of you who go on to do more advanced STATS courses will encounter advanced methods to handle count data that do not satisfy the Poisson assumption.

# Submissions to CRAN using Poisson regression...

## quasi-Poisson correction...

The `family = quasipoisson` specification is saying that the data have different variance than if they were Poisson distributed.

Recall, if $Y$ is Poisson then $\text{Var}(Y) = \text{E}(Y)$. If $Y$ is quasi-Poisson then we assume

$$\text{Var}(Y) \propto \text{E}(Y), \text{or}$$
$$\text{Var}(Y) = k\,\text{E}(Y)$$

When the data have more variance than we would expect under a Poisson assumption, the consequence of using `family = quasipoisson` is that the standard errors of the estimated coefficients are increased, compared to using `family = poisson`.

# Submissions to CRAN using Poisson regression...

## Without quasi-Poisson correction

Recall:

```
> CRAN.gfit= glm(Number~Year,family=poisson,data=CRAN.df)
> summary(CRAN.gfit)
```

```
Call:
glm(formula = Number ~ Year, family = poisson, data = CRAN.df)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.282e+03  1.384e+01  -92.64   <2e-16 ***
Year         6.401e-01  6.868e-03   93.20   <2e-16 ***
---
(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 19374.51  on 11  degrees of freedom
Residual deviance:   402.61  on 10  degrees of freedom
```

# Submissions to CRAN using Poisson regression...

## With quasi-Poisson correction

Compare with:

```
> CRAN.quasigfit = glm(Number ~ Year, family = quasipoisson, data = CRAN.df)
> summary(CRAN.quasigfit)
```

```
Call:
glm(formula = Number ~ Year, family = quasipoisson, data = CRAN.df)

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.282e+03  8.889e+01  -14.42 5.09e-08 ***
Year         6.401e-01  4.411e-02   14.51 4.81e-08 ***
---
(Dispersion parameter for quasipoisson family taken to be 41.25925)

    Null deviance: 19374.51  on 11  degrees of freedom
Residual deviance:   402.61  on 10  degrees of freedom
```

The estimated coefficients are not changed. But, the z-values[5] in the `summary` output will decrease in magnitude, and the $P$-value will increase.

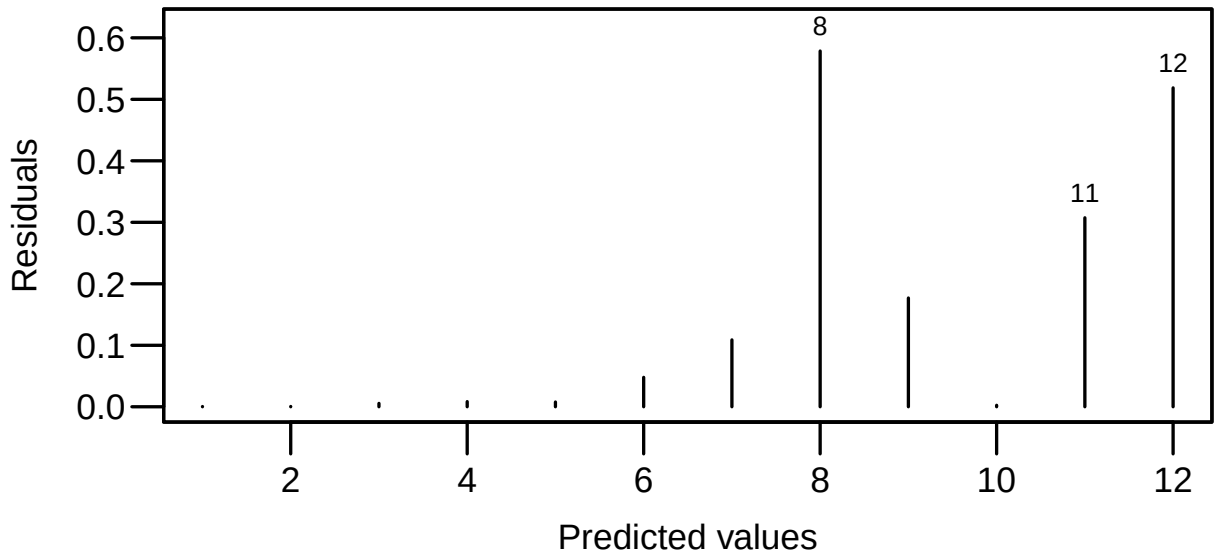
[5] In the quasi-Poisson output the z-values are renamed t-values due to it using an estimated variance.

# Submissions to CRAN using Poisson regression...

## Influence

Let's check the influence of the observations.

```
> plot(CRAN.quasigfit, which = 4)
```




We see that Observations 8 and 12 exceed our 0.4 threshold. However, this has to be interpretted with caution. Unlike the normal linear regression, in a Poisson regression the observations with higher values of $\mu$ are expected to have higher influence than those with lower values of $\mu$.

# Section 13.5
# Interpretting the GLM output

# Submissions to CRAN using Poisson regression...

## Inference

Let us look at the estimated rate of annual increase in submissions of R packages to the CRAN.

Recall that the model is for expected number of submissions to the CRAN is $\mu = \exp(\beta_0 + \beta_1 \times \textit{Year}$, so we need to exponentiate our estimate and the confidence interval.

```
> ## The estimated annual rate of increase
> exp(CRAN.quasigfit$coef["Year"])
    Year
1.896614
```

# Submissions to CRAN using Poisson regression...

## Confidence intervals for effects

We can still use the function `confint()` to get confidence intervals for the parameters.

```
> exp(confint(CRAN.quasigfit))
Waiting for profiling to be done...
                2.5 %   97.5 %
(Intercept) 0.000000 0.000000
Year        1.745819 2.075781
```

Sometimes you might see `confint.default()` used instead.

```
> exp(confint.default(CRAN.quasigfit))
                2.5 %   97.5 %
(Intercept) 0.000000 0.000000
Year        1.739517 2.067898
```

They are doing slightly different things, based on two different approximations. In most cases they give very similar results.

It is generally best to use `confint()` rather than `confint.default()`. See STATS 730 for the reason why.

# Submissions to CRAN using Poisson regression...

## Confidence intervals for effects...

So, our Executive Summary would say that the expected annual number of submissions to CRAN multiplies by between 1.75 and 2.08 times each year. That is, it increases by between 75% and 108% per year.

[This compares to a median multiplier of between 2.00 and 2.40 times each year from the naive `lm` fit to $\log(y)$.]

Note that the GLM model is making statements about the **mean** rather than the **median**. This is because the GLM does not transform $y$.

# Submissions to CRAN using Poisson regression...

## Confidence interval for expected number of submissions

And finally, we will use the quasi-Poisson model to estimate the expected number of submissions in 2017. We first calculate the confidence interval on the linear predictor scale, and then exponentiate.

```
> pred2017.quasi=predict(CRAN.quasigfit, predCRAN.df, se.fit=TRUE)
> # CI for log mean
> lower = pred2017.quasi$fit-1.96*pred2017.quasi$se.fit
> upper = pred2017.quasi$fit+1.96*pred2017.quasi$se.fit
> #CI for mean value
> pred2017.ci.mean=exp(c(lower, upper))
> pred2017.ci.mean
       1        1
 6626.596 10383.014
```

So our Executive Summary would say that the expected number of submissions to CRAN in 2017 is between 6600 and 10400.

This compares to the estimated median number being between 6600 and 25100 from the naive `lm` fit to $\log(y)$.

# Submissions to CRAN using Poisson regression...

## Confidence interval for expected number of submissions...

The above calculations are done for us using the `predictGLM` function.

The confidence interval on the linear predictor scale is

```
> predictGLM(CRAN.quasigfit, predCRAN.df)
***Estimates and CIs are on the link scale***
       fit      lwr      upr
1 9.023387 8.798851 9.247922
```

and on the scale of the response variable is

```
> predictGLM(CRAN.quasigfit, predCRAN.df,type="response")
***Estimates and CIs are on the response scale***
      fit      lwr      upr
1 8294.82 6626.623 10382.97
```

**NOTE:** For GLMs, it is not straightforward to obtain prediction intervals for new values of the response. This is left to STATS 330.

# Section 13.6
# Closing Remarks

# Linear vs generalized linear model

The linear model that we fitted to the logged CRAN data on slide 12 has some undesirable properties.

- Logged count data do not have equal variance.[6]
- Logged count data are highly variable for small expected counts.
- Logged count data can not handle a zero count, since $\log(0)$ is negative infinity.

The influence plot from the LM fitted to the logged CRAN data showed that the 2nd and 3rd observations were the most influential. This is very dangerous, as these are the most unreliable data points.

[6] In fact, if $Y$ is Poisson, then $\text{Var}(\log(Y)) = \infty$!!!

# Linear vs generalized linear model

A GLM is a more natural way to model count data.

It accounts for the fact that the variance of count data $Y$ increases with their mean without having to transform the response.

- It does not transform $y$.
- Inference is about **means** (rather than medians).
- It accounts for the fact that the variance of $Y$ increases with their mean.
- It implicitly allows the higher counts to have more influence.[7]

The GLM better models the CRAN data:

- The plot of predicted values versus residuals clearly shows the sudden increase at year 8.
- The influence plot better shows the true influence of each observation.
- It results in narrower confidence intervals for the expected annual increase and expected future counts.

[7] This is because the signal-to-noise ratio (i.e., standard deviation over mean) of the higher counts is greater than for the smaller counts.

# Linear vs generalized linear model

The two key points of difference are:

- $Y|x$ is assumed to be a Poisson distributed count variable (rather than a normally distributed continuous variable).
- Instead of our model being of additive form

$$E[Y|x] = \beta_0 + \beta_1 x + \dots$$

it is of multiplicative form

$$E[Y|x] = \exp(\beta_0 + \beta_1 x + \dots)$$

or equivalently

$$\log(E[Y|x]) = \beta_0 + \beta_1 x + \dots$$

**NOTE:** The GLM transforms the linear model rather than transforming the data.

# Section 13.7
# Relevant R-code

# Most of the R-code you need for this chapter

We started by identifying what our response is – in this case a count. Therefore our response data is definitely not from a normal distribution. As $Y$ is a count variable we propose to model it using the Poisson distribution.

Instead of using `lm` we use `glm`, and add `family=poisson`.

```
> CRAN.gfit= glm(Number~Year,family=poisson,data=CRAN.df)
> summary(CRAN.gfit)
```

Look at predicted vs residual plot and make sure there's no trend you've missed and/or atypical observations.

```
> plot(CRAN.gfit, which = 1)
```

Do **NOT** check for normality!
Influence can still be assessed using `cooks20x`, but be aware that larger counts will naturally tend to be more influential than smaller counts.

# Most of the R-code you need for this chapter...

To check if the data are consistent with the Poisson model you use

```
> 1 - pchisq(CRAN.gfit$deviance,CRAN.gfit$df.residual)
```

and if the *P*-value is very small then you can to deal with this using the quasi-Poisson model:

```
> CRAN.quasigfit = glm(Number ~ Year, family = quasipoisson, data = CRAN.df)
```

Once you've chosen your final model you can interpret the effect of the variables as a multiplicative change in the **mean** or expected value:

```
> exp(confint(CRAN.quasigfit))
```

If you wish to estimate a 95% CI for an expected value use `predictGLM`, or do it the hard way as follows:

```
> pred2017.quasi=predict(CRAN.quasigfit, predCRAN.df,se.fit=TRUE)
> # CI for log mean
> lower = pred2017.quasi$fit-1.96*pred2017.quasi$se.fit
> upper = pred2017.quasi$fit+1.96*pred2017.quasi$se.fit
> #CI for mean value
> pred2017.ci.mean=exp(c(lower, upper))
> pred2017.ci.mean
```