Chapter 6: Multiplicative linear models

STATS 201/8

University of Auckland

Learning Outcomes

In this chapter you will learn about:

- Mean versus median – which to use?
- Transforming the response variable using the log function
- Multiplicative models
- Why inference is about the median (not the mean)
- A multiplicative simple linear regression example
- A multiplicative two-sample t-test example
- Relevant R-code.

Section 6.1
Mean versus median – which to use?

Auckland house prices

In early 2021 one of the STATS 20x lecturers was looking to buy a house in a working-class suburb of Auckland, about 15 minutes by train to downtown.[1]

The lecturer downloaded 94 recent sales prices for the suburb, and put the prices ($1000's) in a text file called `AkldHousePrices.txt`. The lecturer just wants to know the typical house price in the suburb – there are no explanatory variables.

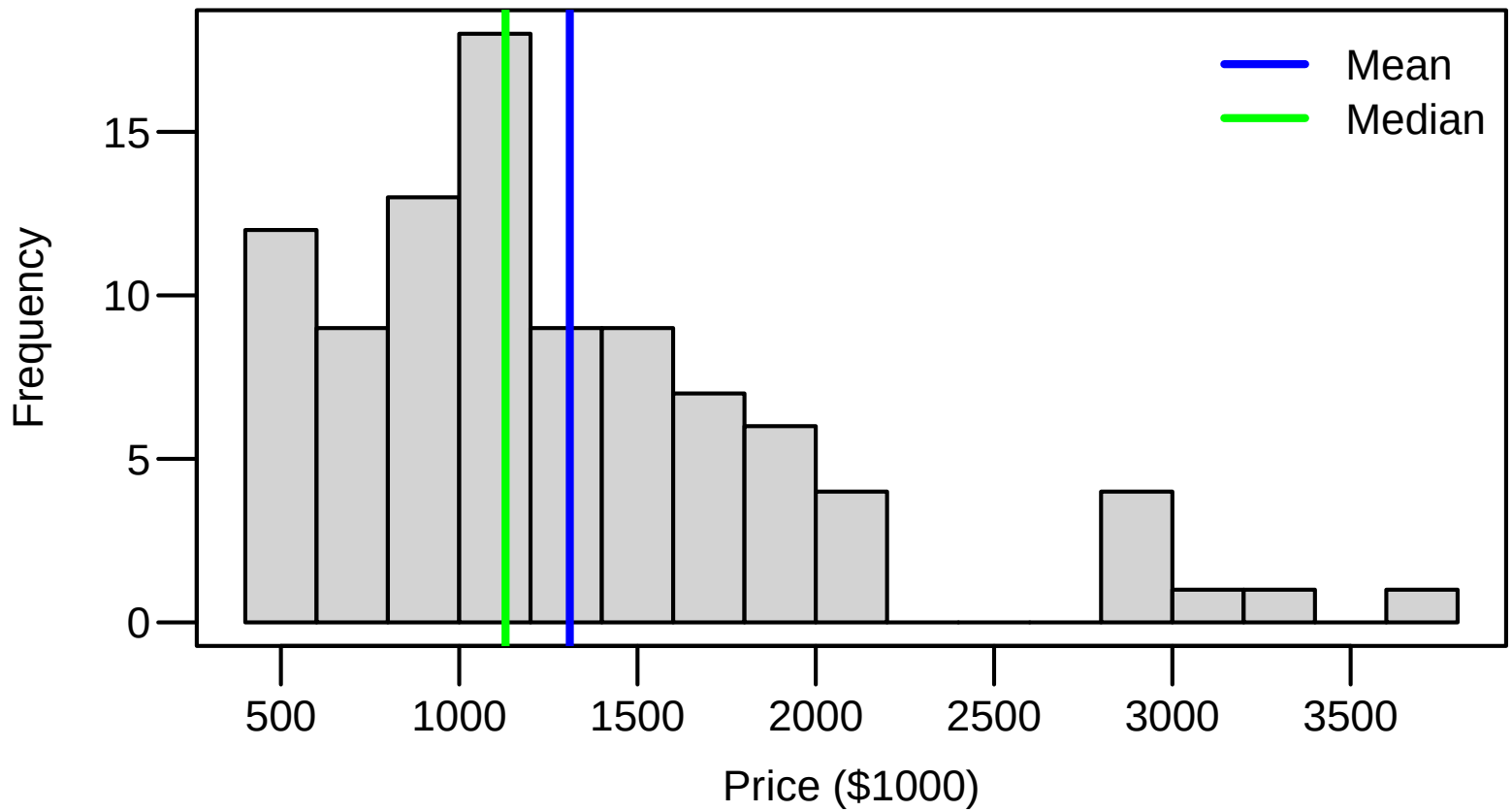
It sounds like an easy analysis. We just need to fit a null model. Let's take a look...

[1] Auckland house prices are very high by international standards.

Auckland house prices...

Inspect the data

```
> Houses.df=read.table("Data/AkldHousePrices.txt",header=T)
> hist(Houses.df$price, breaks=20,main="",xlab="Price ($1000)")
> abline(v = c(mean(Houses.df$price), median(Houses.df$price)),
+        col = c("blue", "green"), lwd = 2)
```



Auckland house prices...

Inspect the data...

```
> summary(Houses.df$price)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  450.0   832.5  1130.0  1310.1  1597.5  3710.0
```

Clearly, we are not dealing with data that come from a normal distribution. See how the (sample) median is markedly lower than the (sample) mean.

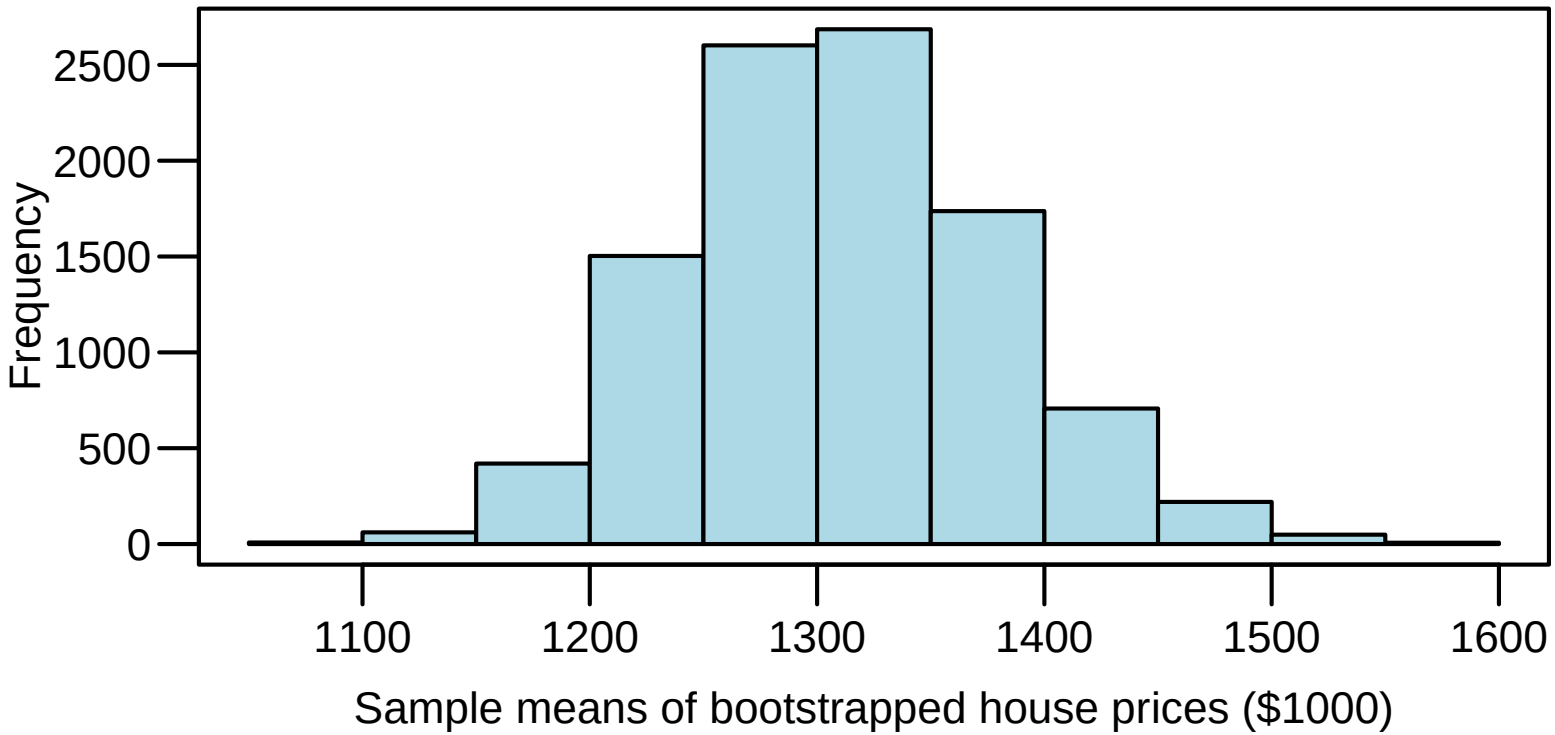
This type of right-skew distribution is very common when it comes to things involving money ($$$), resources, growth, salary, age, advantage and energy, to name but a few.

We can still make inference using a linear model for the expected house price because we can apply the **CLT** since we have a largish number of observations ($n = 94$). Alternatively, we can use the bootstrap since it does not require the assumption of normality.

Auckland house prices...

Inference about the mean

Here is histogram of 10,000 bootstrap sample means from resampling the house price data.



The approximate normality ot the bootstrapped sample means shows that the sample mean has an approximate normal distribution.

Auckland house prices...

Inference about the mean...

Here is the bootstrap 95% CI for the expected price, along with output from the null model.

```
> quantile(bootstrappedMeanPrices, c(.025, .975))
    2.5%    97.5%
1181.168 1452.875
```

```
> HousesNull.fit=lm(price~1, data=Houses.df)
> summary(HousesNull.fit)
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1310.1       70.1   18.69   <2e-16 ***
---
Residual standard error: 679.7 on 93 degrees of freedom
```

```
> confint(HousesNull.fit)
                2.5 %   97.5 %
(Intercept) 1170.899 1449.313
```

Here we see that, to within less than NZ$10 000, the bootstrap CI and the model CI are the same. This is confirmation that the **CLT** works.

Auckland house prices...

Inference about the mean...

So we can safely say that the mean (i.e., expected) sale price for the entire suburb is somewhere between, say, NZ$1.17 million to NZ$1.45 million. However, the gross non-normality of the data prevents us from making prediction intervals for house prices.

Moreover, the estimated mean price of NZ$1.31 million is somewhat misleading.[2] Our house-hunting lecturer just wants a typical house, about midway in affordability at most. It would make more sense to estimate the **median** house price, since our lecturer will then know that half of the houses going on the property market will sell for less than that price.

In general, for highly skewed data the median is usually a better measure of 'typical' value than the mean.

[2] It is also not statistically robust since it is very sensitive to the number of expensive ($\geq$ \$3 million, say) houses sold

Auckland house prices...

Inference about the median

To estimate the median sale price of the entire suburb the natural estimate is the median of our sample:

```
> median(Houses.df$price)
[1] 1130
```

and we can use a bootstrap to get a 95% CI for the suburb median

```
> quantile(bootstrappedMedianPrices, c(.025, .975))
 2.5% 97.5%
 1040  1320
```

so we can say that the median sale price for the entire suburb is somewhere between, say, NZ$1.04 million to NZ$1.32 million.

The 95% CI for the median comes as something of a relief to our house-hunting lecturer. It is much more reasonable than the $1.17 to $1.45 million CI for the mean which was markedly higher due to the data being so right-skewed.

Inference about the median

In the above house price example we are working with iid data, so it is natural to use the sample median to estimate the population median.

We'll see in the next section that the linear model framework can also be used to make inference about the median provided that the logged response variable is approximately normally distributed. This approach has the advantage that it can also be applied to more general situations where we have explanatory variables that may be associated with the response variable.

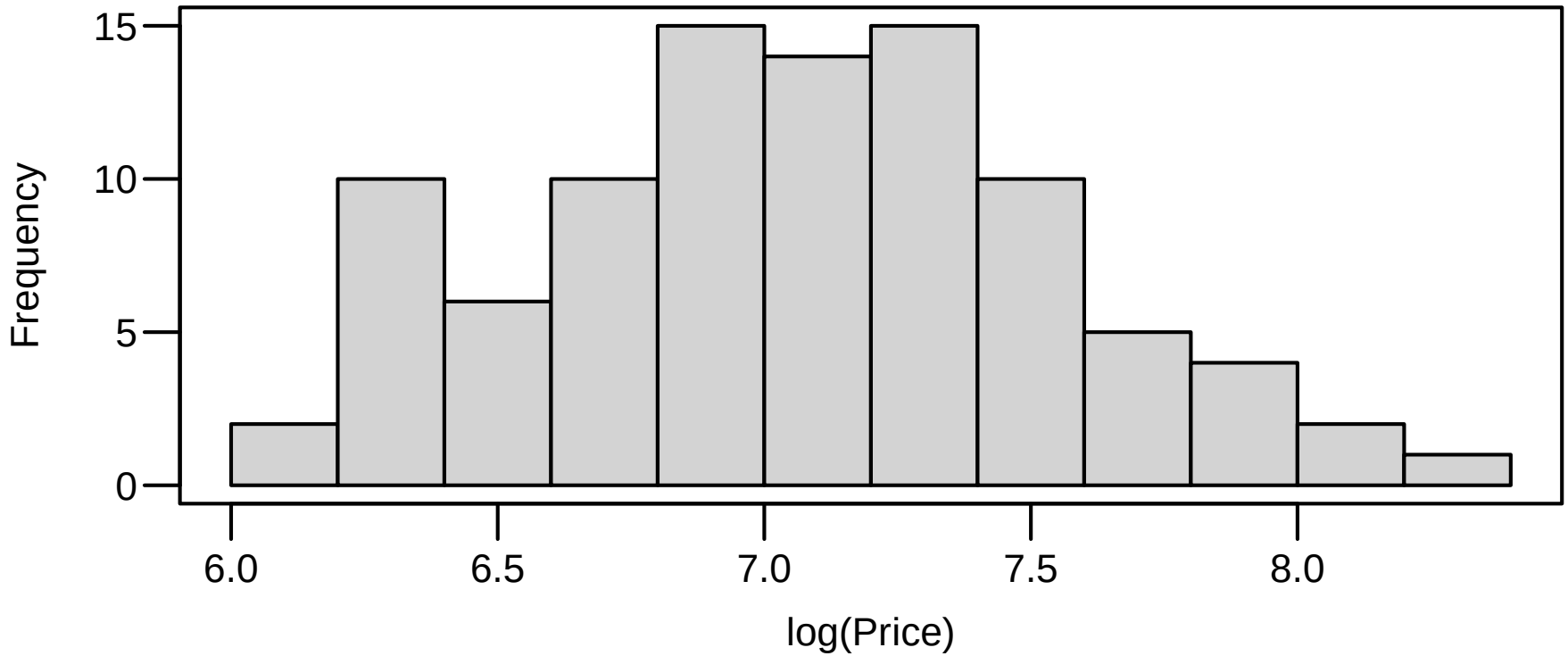
We'll also see that fitting linear models to logged response data results in the effects of explanatory variables acting multiplicatively on medians.

Section 6.2
Transforming the response variable using the log function

Auckland house prices...

Transforming the data

Let's consider making a transformation of the prices. In particular, the log transformation. Here is the histogram of `log(price)`.



This looks reasonably close to normal, so if we fit a linear model to these data then all inferences will be valid.

Auckland house prices...

Null model fitted to logged house price data

```
> LoggedPriceNull.fit=lm(log(price)~1, data=Houses.df)
> coef(summary(LoggedPriceNull.fit))
             Estimate Std. Error  t value      Pr(>|t|)
(Intercept) 7.060405 0.04974049 141.9448 1.628721e-110
```

```
> confint(LoggedPriceNull.fit)
               2.5 %  97.5 %
(Intercept) 6.96163 7.15918
```

Well that is interesting, but logged house prices don't mean much to anyone who is hoping to buy a house. The inference needs to be back-transformed to NZ$.

Since we've used the log transformation, the back-transformation is the exponential function `exp()`.

```
> exp(confint(LoggedPriceNull.fit))
                2.5 %   97.5 %
(Intercept) 1055.353 1285.856
```

Auckland house prices...

The effect of transforming

The above confidence interval is quite different from the one we calculated for the mean house price of the suburb. The reason for this is because the above CI is for the **median** house price.

To see why this is, let's take a look at what happens when we transform summary statistics using the `log()` and `exp()` functions:

Summaries of price:

```
> summary(Houses.df$price);
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  450.0   832.5  1130.0  1310.1  1597.5  3710.0
```

Summaries of log(price):

```
> summary(log(Houses.df$price))
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  6.109   6.724   7.030   7.060   7.376   8.219
```

Back-transformed summaries of log(price):

```
> exp(summary(log(Houses.df$price)))
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  450.0   832.5  1130.0  1164.9  1597.0  3710.0
```

Auckland house prices...

The effect of transforming...

Note that the sample mean changed after transforming then back-transforming, but the other summary values did not. This is because transforming and back-transforming does not change the order of the data – the smallest value will still be the smallest value, the middle value will still be the middle value.

Auckland house prices...

Why inference is about the population median

If the logged response variable is normally distributed, then on the log scale its population mean and median are exactly the same number.[3] In that case, a 95% confidence interval for the population mean of the logged response variable is also a 95% CI for the population median of the logged response variable.

Since the median is unaffected by transforming and back-transforming, it follows that back-transforming the above 95% confidence interval will give a 95% confidence interval for the population median on the original scale. However, it will not be a 95% confidence interval for the population mean.

We now have a recipe for calculating a confidence interval for the **median** – fit a linear model to the log-transformed data, calculate a 95% confidence interval for the **mean** (and hence **median**), and back transform.

[3] The sample mean and median may differ a little, as we saw with the logged house prices

Auckland house prices...

Inference about the median

Our back-transformed estimate ($\exp(\hat{\beta}_0)$) and 95% CI for the median suburb sale price are

```
> exp(coef(LoggedPriceNull.fit))
(Intercept)
   1164.917
> exp(confint(LoggedPriceNull.fit))
                2.5 %   97.5 %
(Intercept) 1055.353 1285.856
```

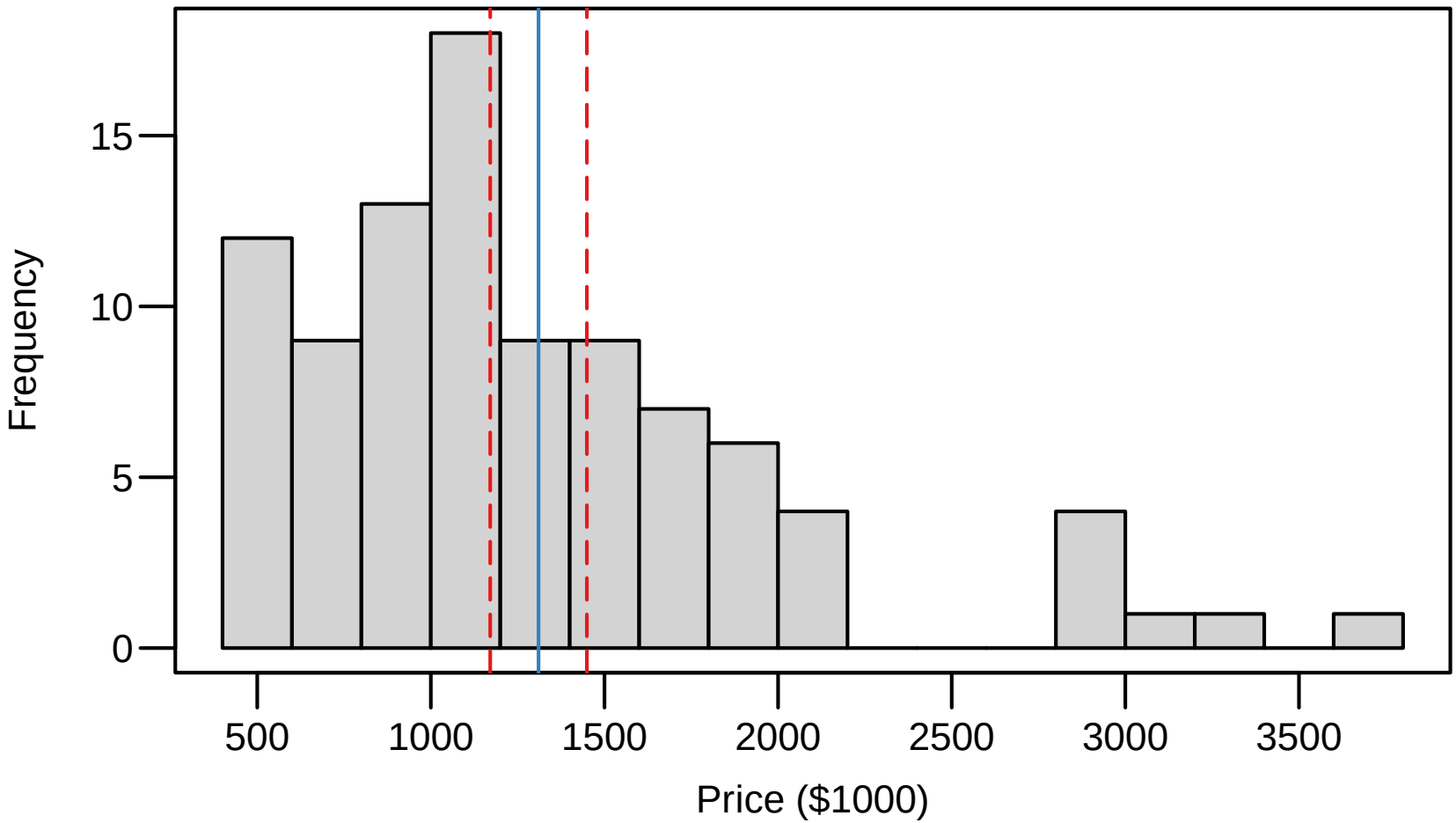
So we can say we are reasonably sure (95% confident) that the median house price is somewhere between NZ$1.06 and NZ$1.29 million.

These values differ a little from the sample median (NZ$1130) and the 95% bootstrap CI of NZ$1040 to NZ$1320 we saw in the previous section. This is because the machinery being used is different.

Auckland house prices...

Inference for the house price data

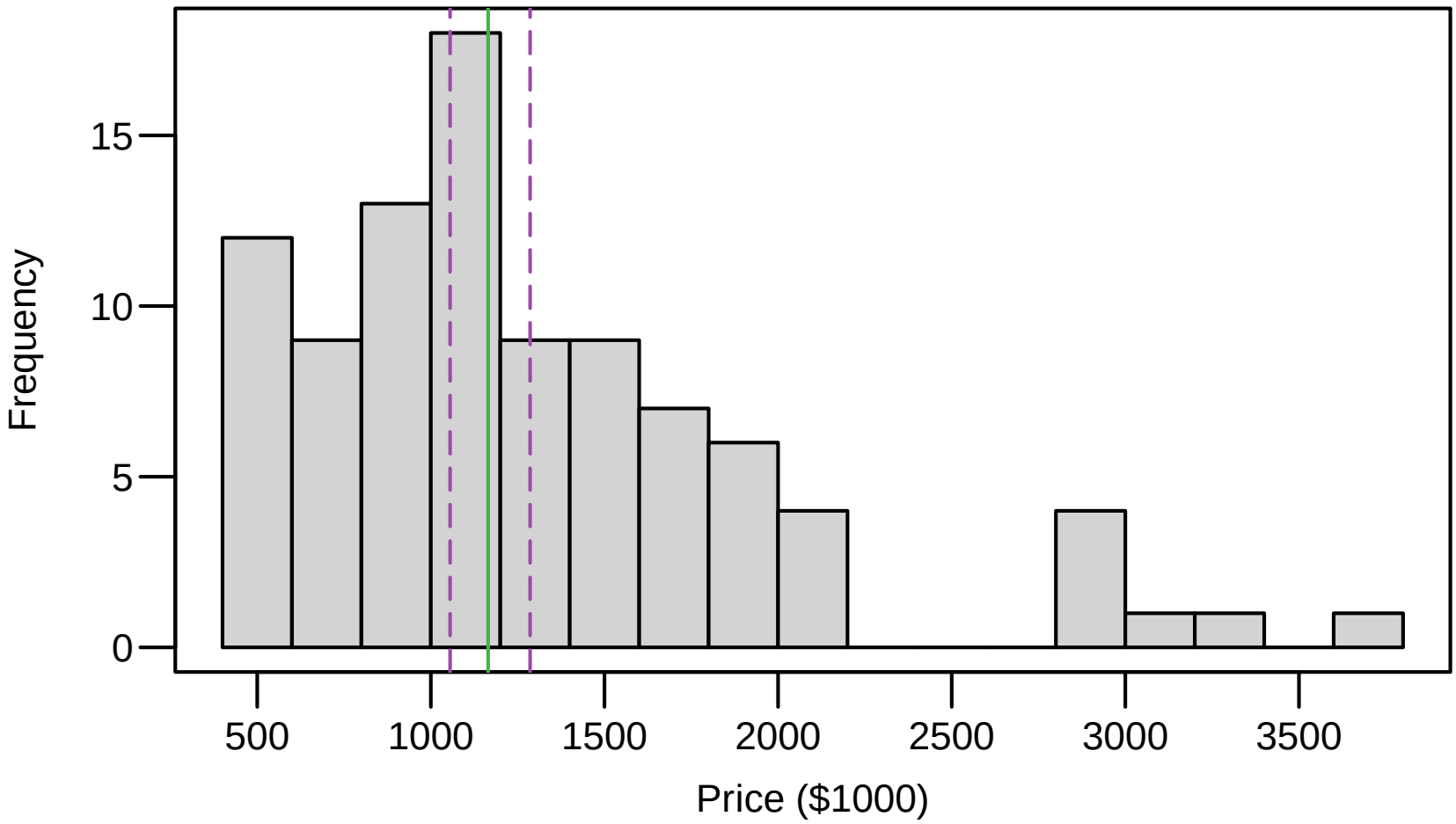
Here is the confidence interval for the mean suburb price with the sample mean shown in blue.



Auckland house prices...

Inference for the house price data...

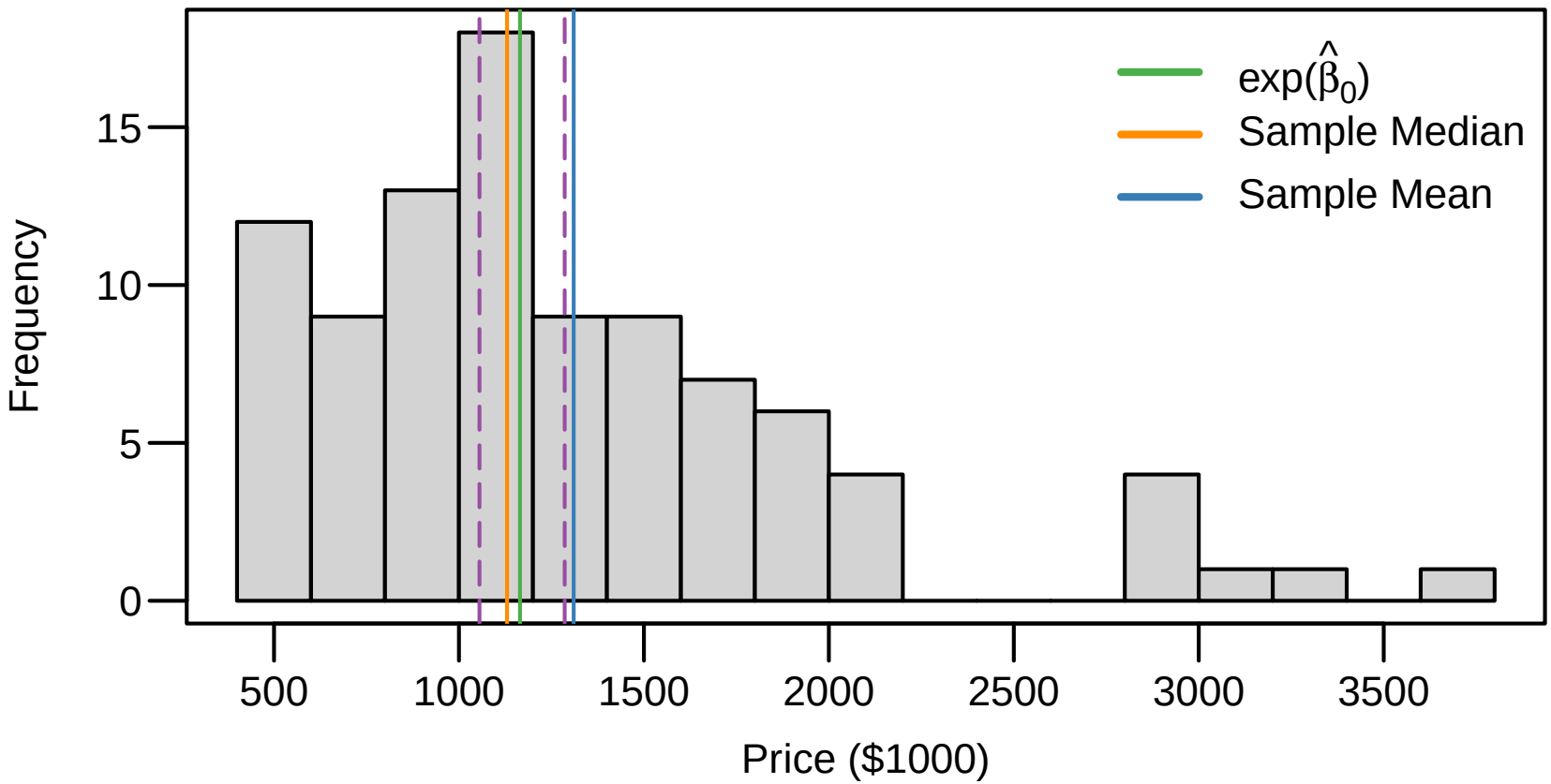
Here is the confidence interval for the median suburb price with the estimate obtained from the linear model approach ($\exp(\hat{\beta}_0)$) shown in green.



Auckland house prices...

Inference for the house price data...

Here is the above plot with the sample mean and sample median shown as well.



Section 6.3
The log function turns multiplicative effects in to additive effects

Why the log transformation?

In the above house price example it would have been possible to use many other choices for our transformation of the prices. How did we know that the log-transformation would transform the data to be approximately normally distributed? We didn't! But, the log transformation has a very special property that makes it a very sensible choice. **The log transformation turns multiplicative effects in to additive effects.**

In many situations, the response variable is subject to effects that act multiplicatively. E.g., would proximity to a train station act additively or multiplicatively on the price of a house?[4] What about having a view or being beside the estuary?

In such cases, taking the log of the response variable results in these effects acting additively on the log scale...and by virtue of the CLT, the consequence of a lot of effects adding together is approximate normality on the log scale.

[4] In other words, does being close to the train station make a house worth, say, \$150,000 more, or 10% more?

Logarithm refresher

Turning multiplication in to addition

You have used logarithms before. When you multiply 100 by $100,000$ you add the zeroes to create $10,000,000$. That is, two 0s + five 0s = seven 0s $= 10,000,000$. More formally:

$$100 \times 100,000 = 10^2 \times 10^5 = 10^{2+5} = 10^7 = 10,000,000$$

Using base 10 (the number of fingers/toes we humans have) we can write $log_{10}(100) = log_{10}(10^2) = 2$. So ,

$$\begin{aligned} log_{10}(100 \times 100,000) &= log_{10}(10^2 \times 10^5) \\ &= log_{10}(10^{2+5}) = 2 + 5 = 7 \\ &= log_{10}(100) + log_{10}(100,000) \ . \end{aligned}$$

So we see that the log of a product (i.e., multiplication) is the sum (i.e, addition) of the logs.

In Springfield, they *should* work in base 8. This means that "100" in Springfield would correspond to our 64.

Why use base e?

In the scientific world we use base e logarithms[5] ($\log_e$) which are referred to as natural logs[6].

In `R` the `log` function is the natural logarithm. If you need to calculate the logarithm base 10, then you can use the `log10` function, or type `log(x, base = 10)`, where `x` is the value for which you are trying to calculate the logarithm.

In many situations, it does not matter which base you use as the laws of logarithms hold regardless of the base you use. That is,

$$\log_b(x \times y) = \log_b(x) + \log_b(y)$$

for any base $b > 0$, and values $x, y > 0$.

[5] $e = \lim_{n\to\infty}(1 + \frac{1}{n})^n = 2.718282\ldots$

[6] Some programming languages and books use ln for the $\log_e$ function.

The inverse log function, e^x

Turning addition in to multipication...

Because we use natural logs ($\log_e$), it follows that the inverse transformation (i.e., back transformation) is the exponential function. This is the `exp` function in `R`.

Exponentiating a sum is equivalent to multiplying the separate exponentials. That is,

$$e^{x+y} = e^x \times e^y$$

For example:

```
> exp(2)
[1] 7.389056
> exp(3)
[1] 20.08554
> exp(2+3)
[1] 148.4132
> exp(2)*exp(3)
[1] 148.4132
```

Section 6.4
Example 1: Multiplicative simple linear regression model

Multiplicative model with numerical explanatory variable

Mazda price data

When we also have explanatory variables (x), using the log transformation (applied to the response variable y) changes the shape of the relationship between x and y from additive to multiplicative. This is because we use the exponential to back-transform our additive linear model fitted to $\log(y)$.

We demonstrate with a new example:

The year of manufacturer and asking price of 123 Mazda cars were collected from the Melbourne Age newspaper in 1991. The variables measured were:

`price`	price of vehicle in Australian \$
`year`	year of manufacture ($1990 = 90$)

We will assume that these data are a random sample from the population.[7]

[7] What would the population be here?

Multiplicative model with numerical explanatory variable

Depreciation of Mazda cars

Here we wish to understand how the cars lose value as they age. We have the year the car was manufactured so we can ascertain its age since these data were collected in 1991 (or '91). That is, `age=91-year`.

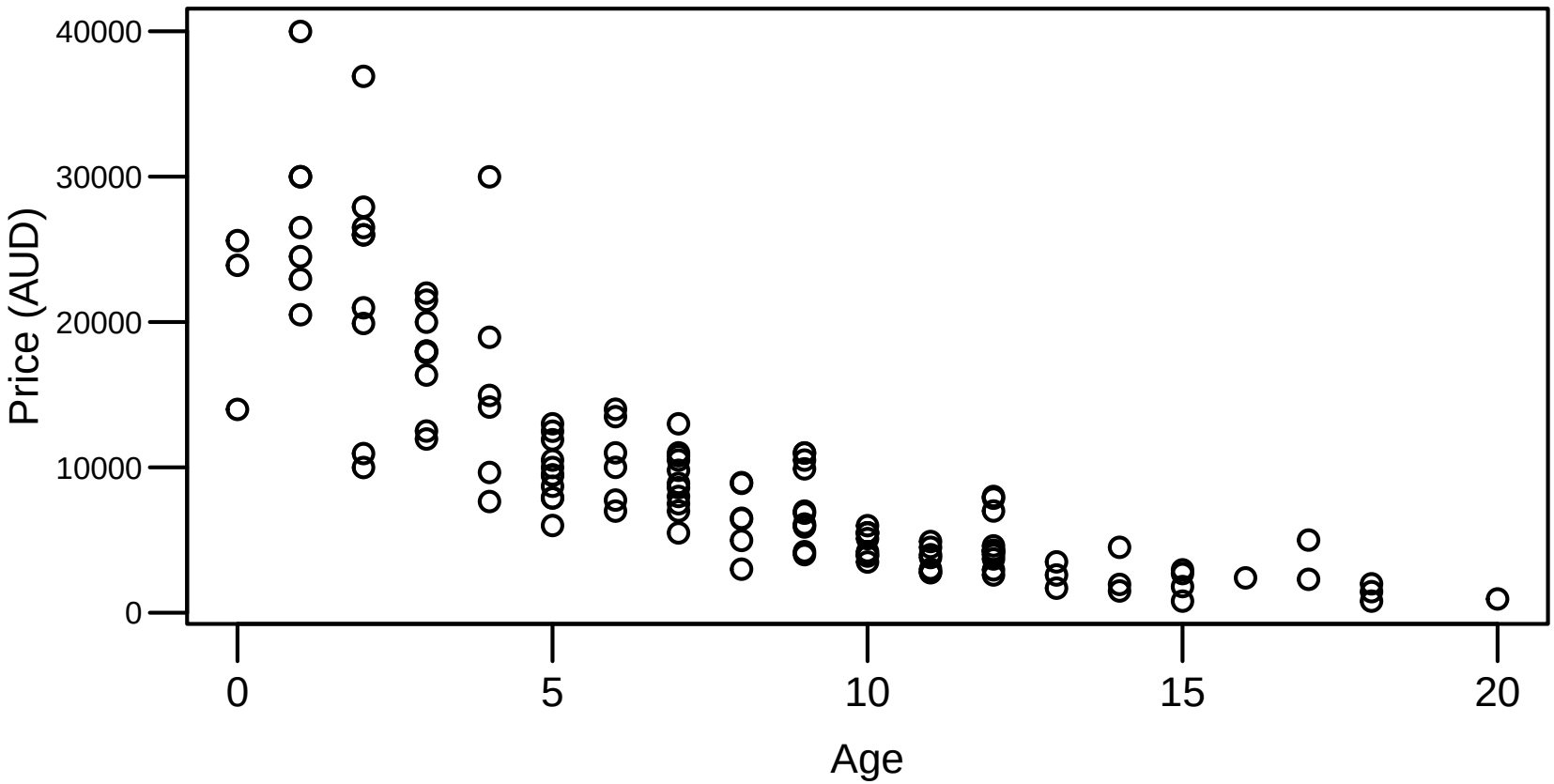
Intuition: What does common sense tell us about the shape of the relationship between car price and age? That is, what do we *expect* to see?

- An additive decrease in price with age?
- A multiplicative decrease in price with age?

Multiplicative model with numerical explanatory variable

Depreciation of Mazda cars...

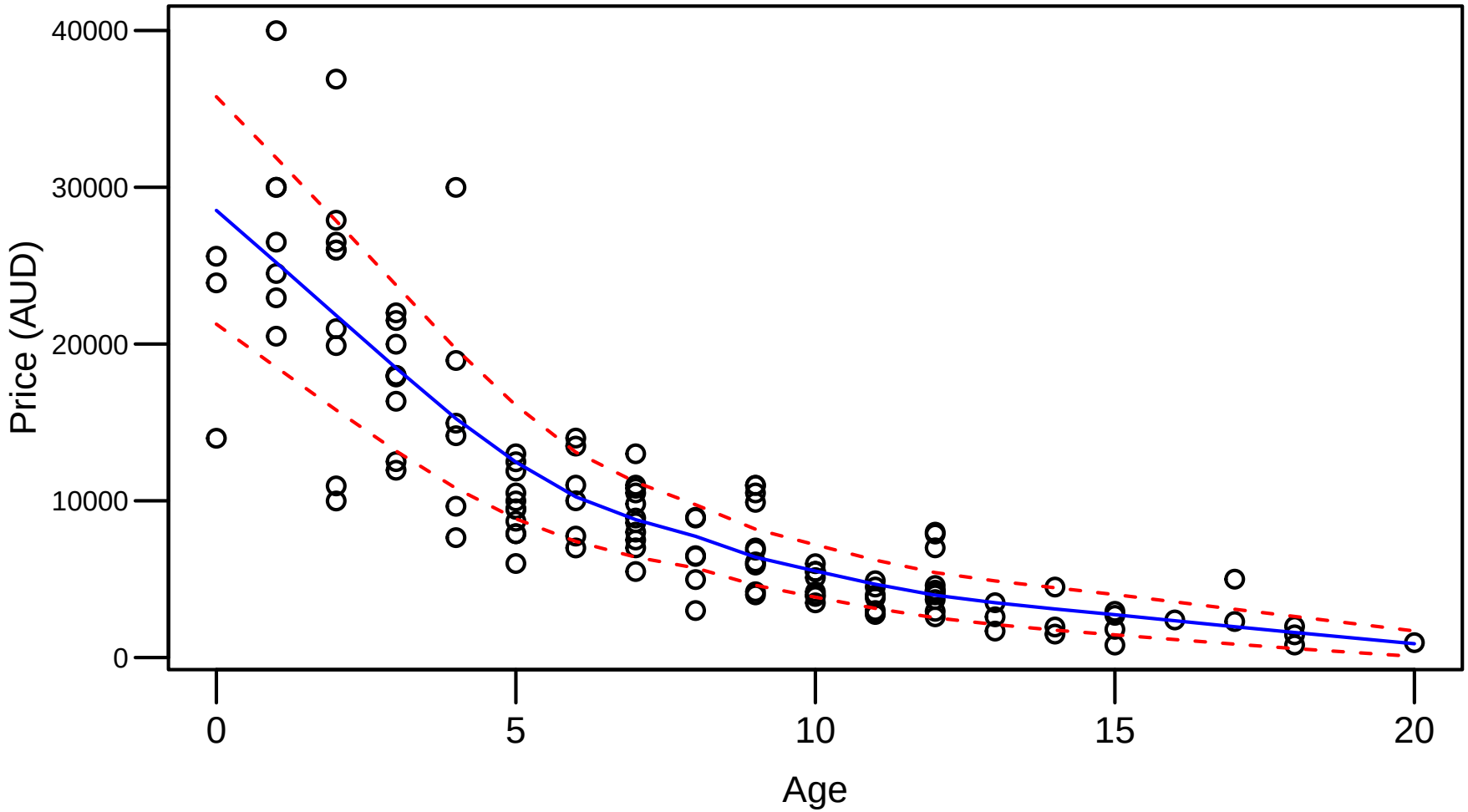
```
> Mazda.df = read.table("Data/mazda.txt",header=T)
> Mazda.df$age = 91 - Mazda.df$year ## Create the age variable
> plot(price~age, data = Mazda.df, xlab = "Age", ylab = "Price (AUD)")
```



Multiplicative model with numerical explanatory variable

Depreciation of Mazda cars...

Let us look at the two components we are interested in, trend and scatter.



Multiplicative model with numerical explanatory variable

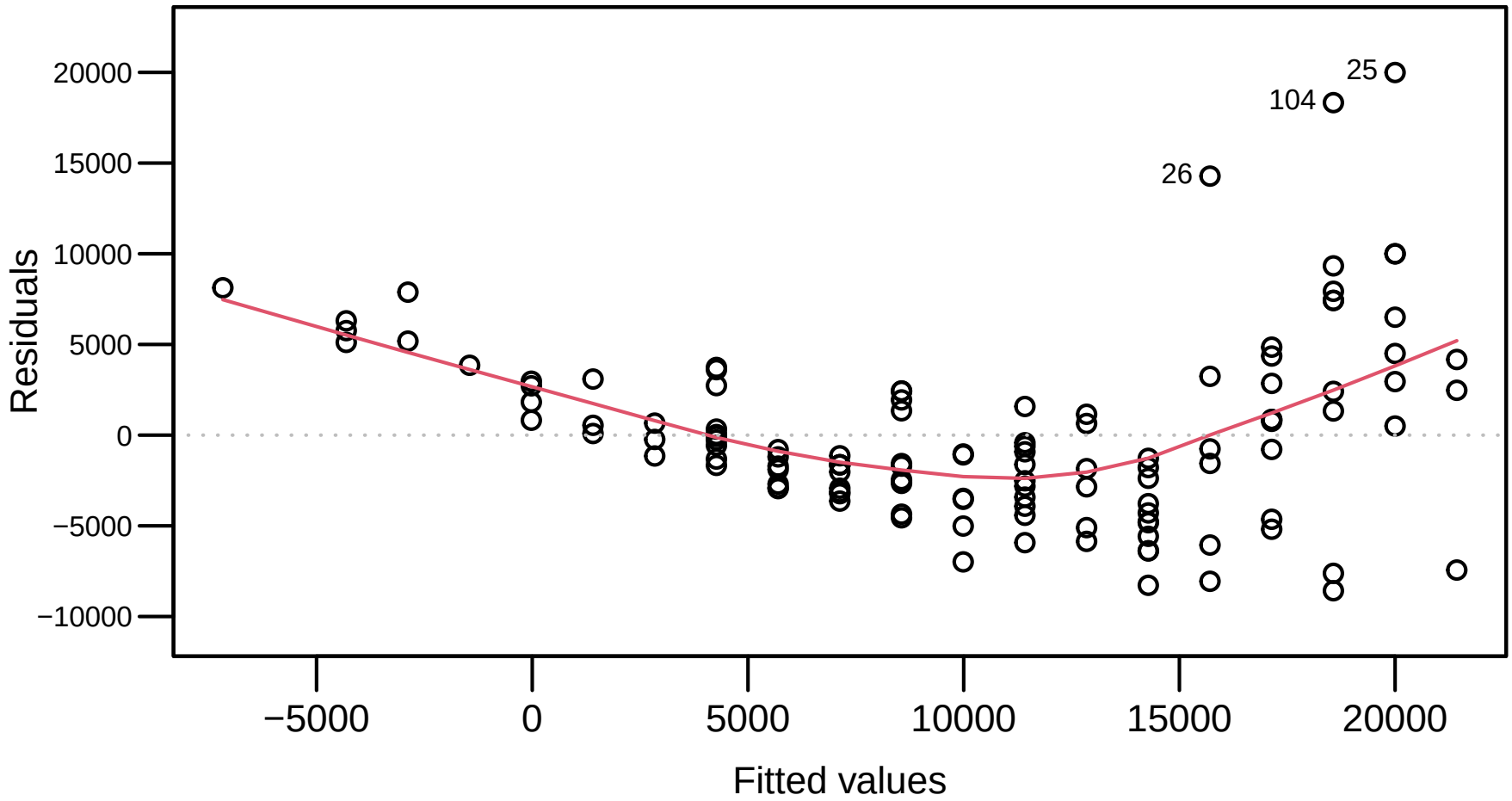
Depreciation of Mazda cars: A naïve price vs age models

The trend is decreasing (exponentially), along with decreasing scatter – these are classic symptoms of an underlying multiplicative model. Assuming **EOV** would be naïve in this case. Let us be naïve and see where it takes us. Let us fit a a linear model and see what the residual plot tells us.

Multiplicative model with numerical explanatory variable

Naïve price vs age models. . .

```
> PriceAge.fit=lm(price~age, data=Mazda.df)
> plot(PriceAge.fit,which=1)
```



Multiplicative model with numerical explanatory variable

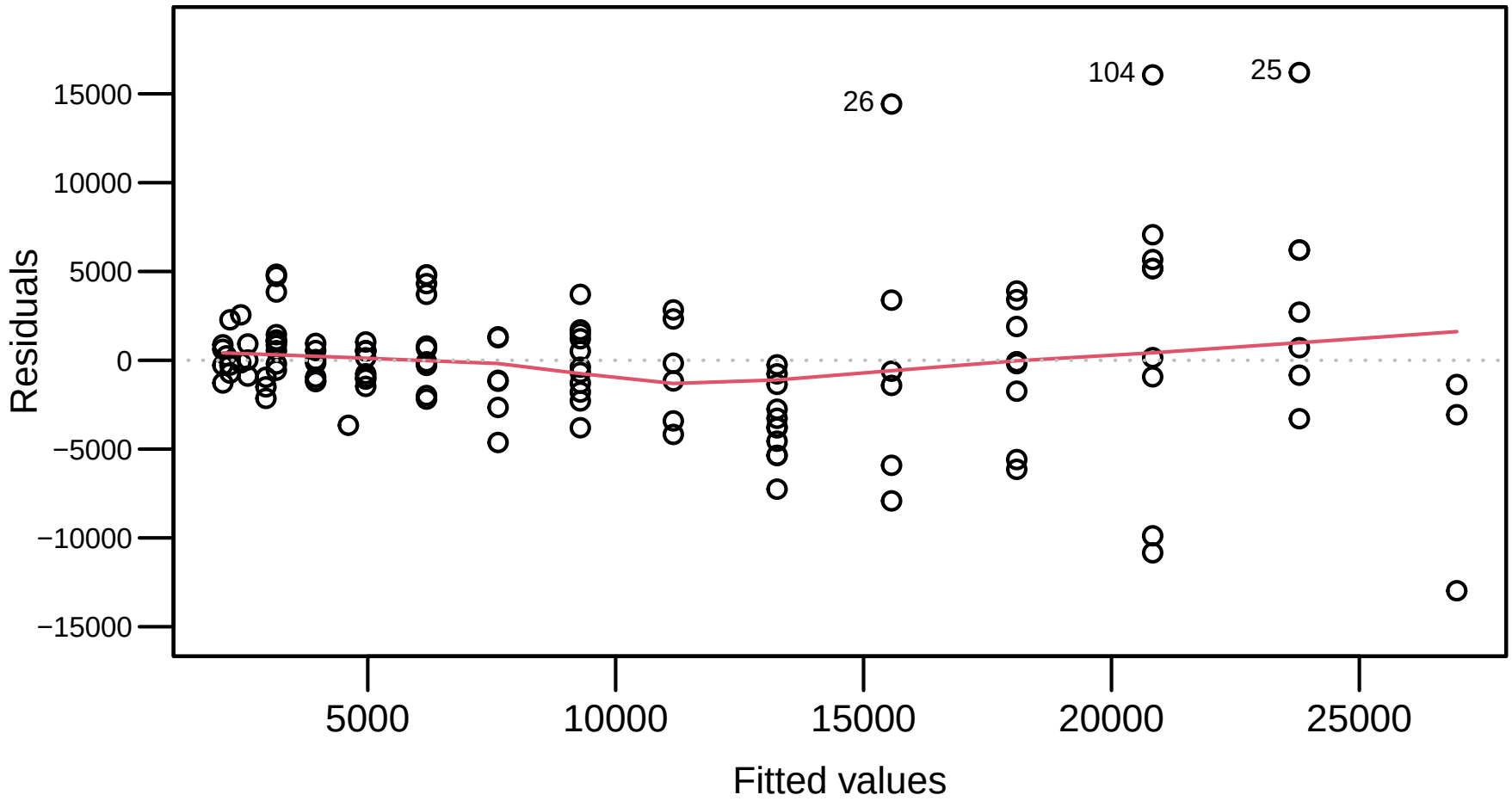
Naïve price vs age models. . .

The decreasing non-linear trend and non-constant scatter has become even more apparent. Note: the higher fitted values on the horizontal axis are associated with newer cars. Let us be slightly less naïve and deal with the trend by fitting a quadratic term.

Multiplicative model with numerical explanatory variable

Naïve price vs age models. . .

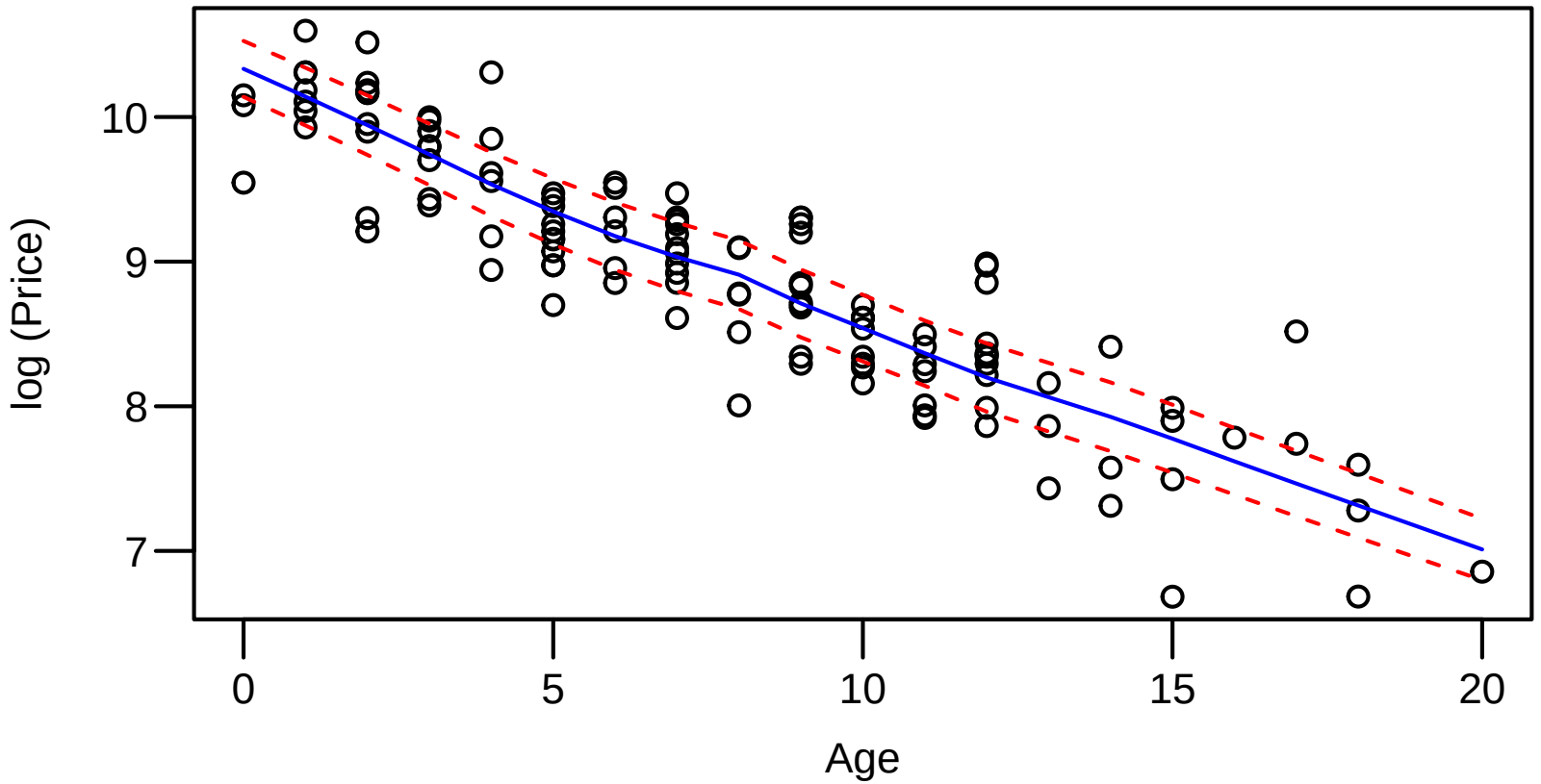
```
> PriceAge.fit2=lm(price~age+I(age^2), data=Mazda.df)
> plot(PriceAge.fit2,which=1)
```



Multiplicative model with numerical explanatory variable

Multiplicative price vs age model

We have eliminated trend from these residuals but the **EOV** assumption is still violated. Let us 'tear up' this approach and take logs of price.



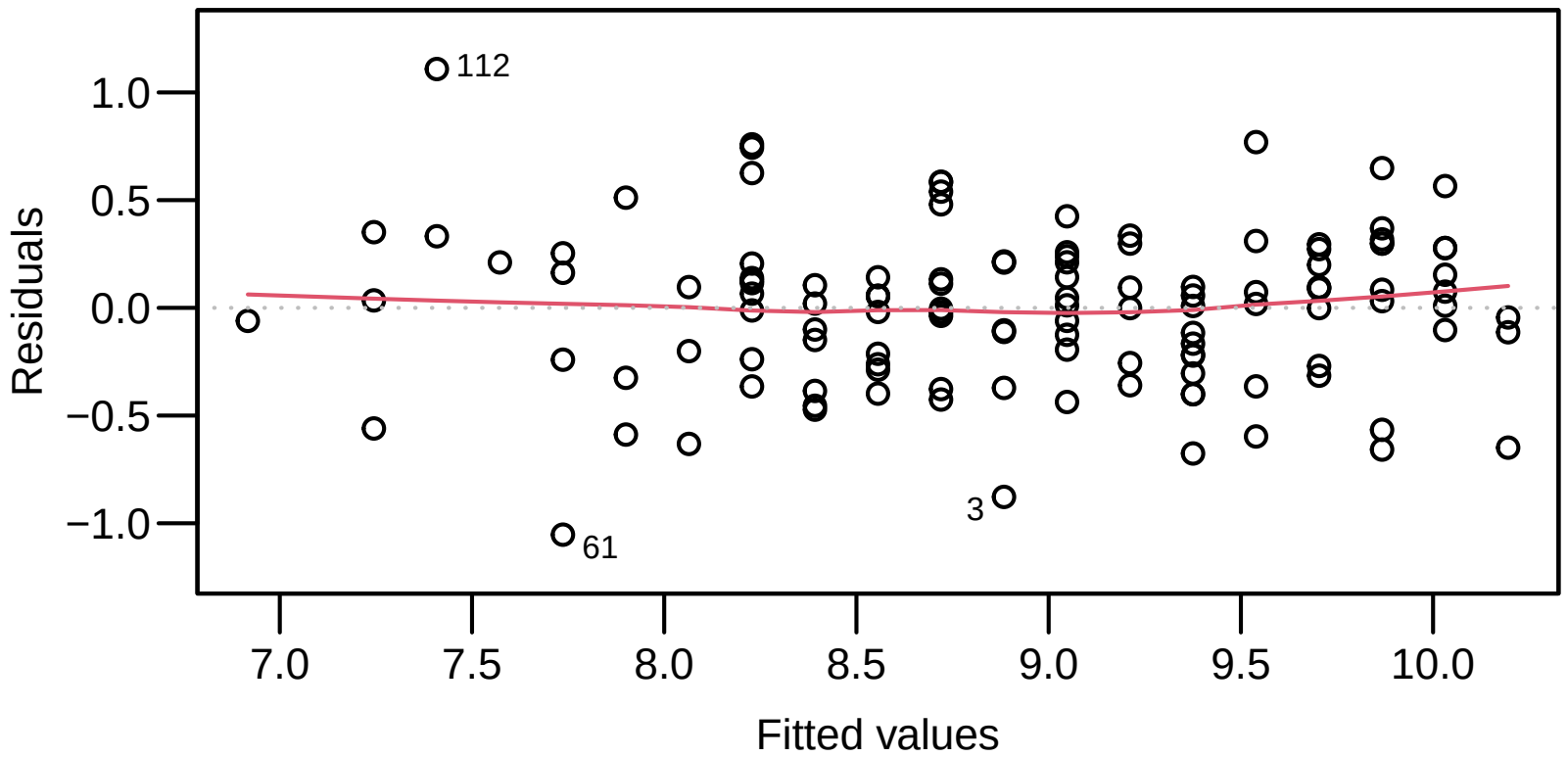
This is something we have seen before, is it not?

Multiplicative model with numerical explanatory variable

Multiplicative price vs age model...

Let us fit the more appropriate multiplicative model.

```
> LogPriceAge.fit=lm(log(price)~age, data=Mazda.df)
> plot(LogPriceAge.fit,which=1)
```



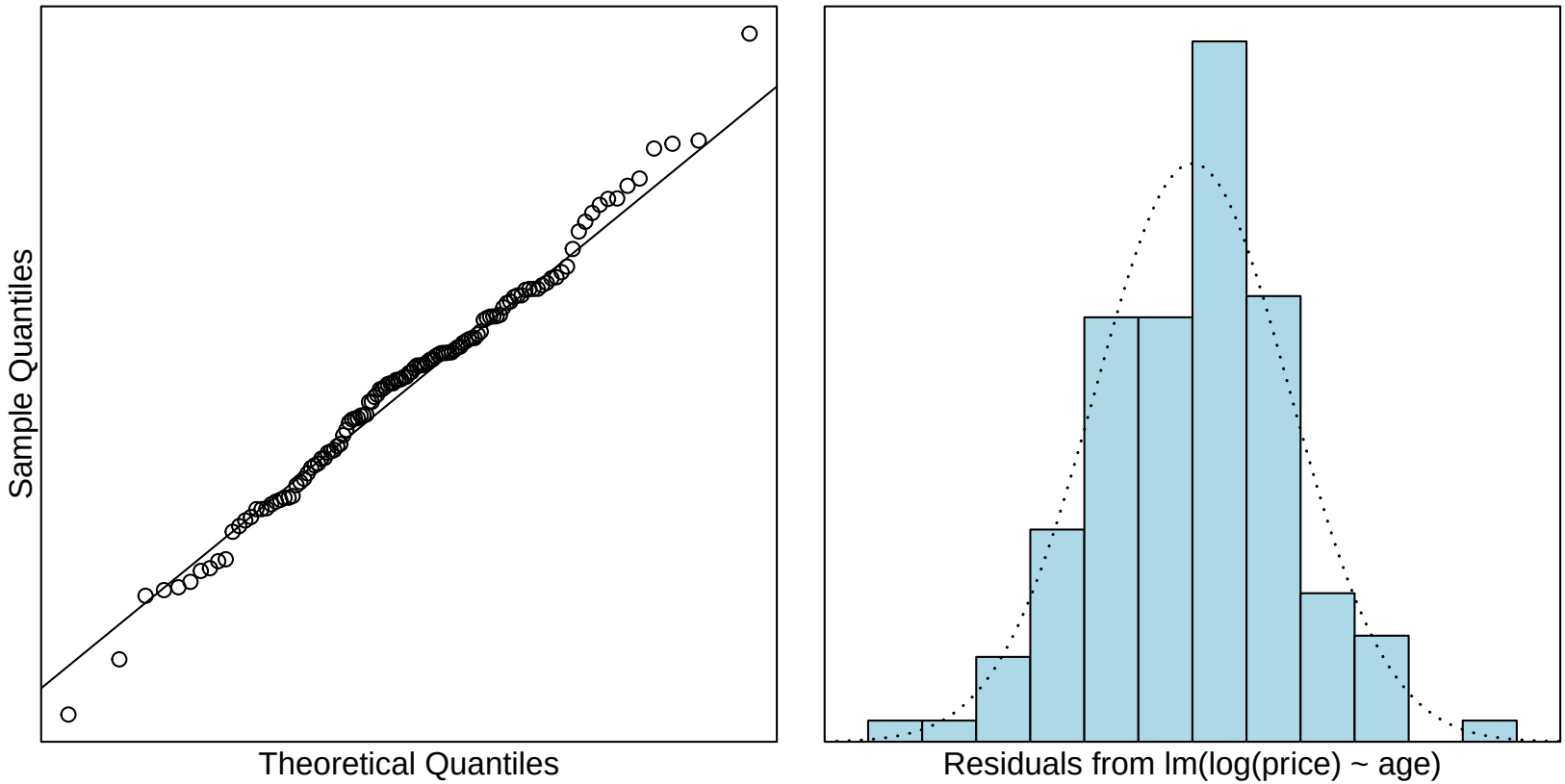
EOV assumption is now satisfied.

Multiplicative model with numerical explanatory variable

Multiplicative price vsage model...

Check for normality of the residuals.

```
> normcheck(LogPriceAge.fit)
```



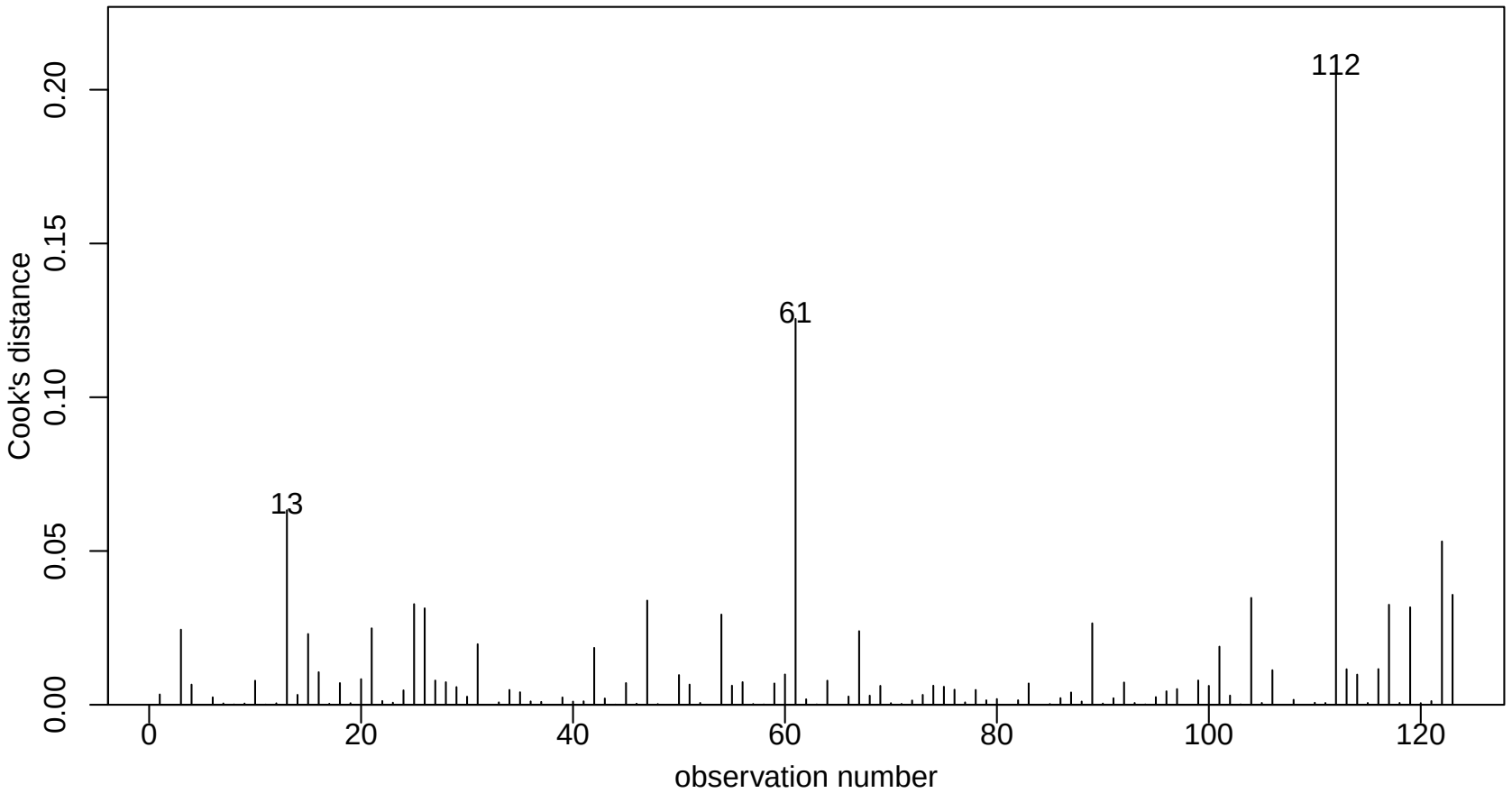
Normality assumption seems justified.

Multiplicative model with numerical explanatory variable

Multiplicative price vs age model...

Check for unduly influential data points.

```
> cooks20x(LogPriceAge.fit)
```



It looks fine.

Multiplicative model with numerical explanatory variable

Multiplicative price vs age model...

Our assumptions are sound, so we can trust the resulting output.

```
> summary(LogPriceAge.fit)
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 10.195210   0.063602   160.3   <2e-16 ***
age         -0.163915   0.007034   -23.3   <2e-16 ***
---
Residual standard error: 0.3615 on 121 degrees of freedom
Multiple R-squared:  0.8178,Adjusted R-squared:  0.8163
F-statistic: 543.1 on 1 and 121 DF,  p-value: < 2.2e-16
```

```
> confint(LogPriceAge.fit)
                 2.5 %     97.5 %
(Intercept) 10.0692935 10.3211263
age         -0.1778406 -0.1499902
```

There is a massively significant effect of age.

But how do we turn these estimated values into something meaningful?

Multiplicative model with numerical explanatory variable

Inference from multiplicative price vs age model

The above fitted model gives median log-price for a car of a given age. So, exponentiating this model will give median price on the raw scale of \$A. Using the equation for the fitted model, the estimated median prices is:

$$\begin{aligned}\widehat{\texttt{price}} &= e^{\hat{\beta}_0+\hat{\beta}_1\times\texttt{age}} \\ &= e^{\hat{\beta}_0}e^{\hat{\beta}_1\times\texttt{age}} \\ &= e^{10.195210} \times e^{-0.163915\times\texttt{age}} \\ &\approx \$26,775 \times (0.85)^{\texttt{age}}\end{aligned}$$

This means that for every year the car ages it is worth .85 (85% of) the previous year. That is, there is a 15% decline in value. In accounting this is known as depreciation of an asset.
E.g., we estimate that the median price of a new Mazda is about \$26,775, a 1 year old about $\$26,775 \times 0.85^1 \approx \$22,727$, and a 2 year old about $\$26,775 \times 0.85^2 \approx \$19,291\ldots$

Multiplicative model with numerical explanatory variable

Inference from multiplicative price vs age model

Note: We exponentiate the model fitted to the logged Mazda prices to obtain a model for Mazda price.

Assumption checks, CIs, predictions (etc) use the model you fitted using the `lm` 'machinery'. That is, on the log scale.

The CIs and predictions are for use in the 'real world', and so must be back-transformed onto the scale of the raw data.

Multiplicative model with numerical explanatory variable

CIs from multiplicative price vs age model

We can obtain the confidence interval for median price of a new car by back-transforming the CI for the intercept value, just like we did with the null model discussed earlier.

```
> exp(confint(LogPriceAge.fit))
                  2.5 %       97.5 %
(Intercept) 2.360688e+04 3.036744e+04
age         8.370758e-01 8.607164e-01
```

The output: `(Intercept) 2.3607e+04 3.0367e+04` [8] says that the we are confident that the median price of a new Mazda car in 1991 is somewhere between AUD23,600 and AUD30,400 (to the nearest \$100).

[8] `2.3607e+04`$= 2.3607 \times 10^4 =$ \$23607

Multiplicative model with numerical explanatory variable

CIs from multiplicative price vs age model...

Now let us look at the coefficient for age i.e. $\hat{\beta}_1$.

If we exponentiate we get $e^{\hat{\beta}_1} = e^{-0.1639154} \approx 0.85$ or using R:

```
> exp(coef(LogPriceAge.fit)[2])
      age
0.8488138
```

From the previous page we see that the 95% CI for this coefficient is 8.37076e-01 to 8.60716e-01. That is, between 0.837 and 0.861, say.

Alternatively, we can calculate the percentage rate of depreciation:

```
> 100*(exp(confint(LogPriceAge.fit)[2,])-1)
    2.5 %    97.5 %
-16.29242 -13.92836
```

This says that our 95% CI for the annual depreciation in median price of Mazda cars is between $100\% \times (1 - 0.861) = 13.9\%$ and $100\% \times (1 - 0.837) = 16.3\%$

Multiplicative model with numerical explanatory variable

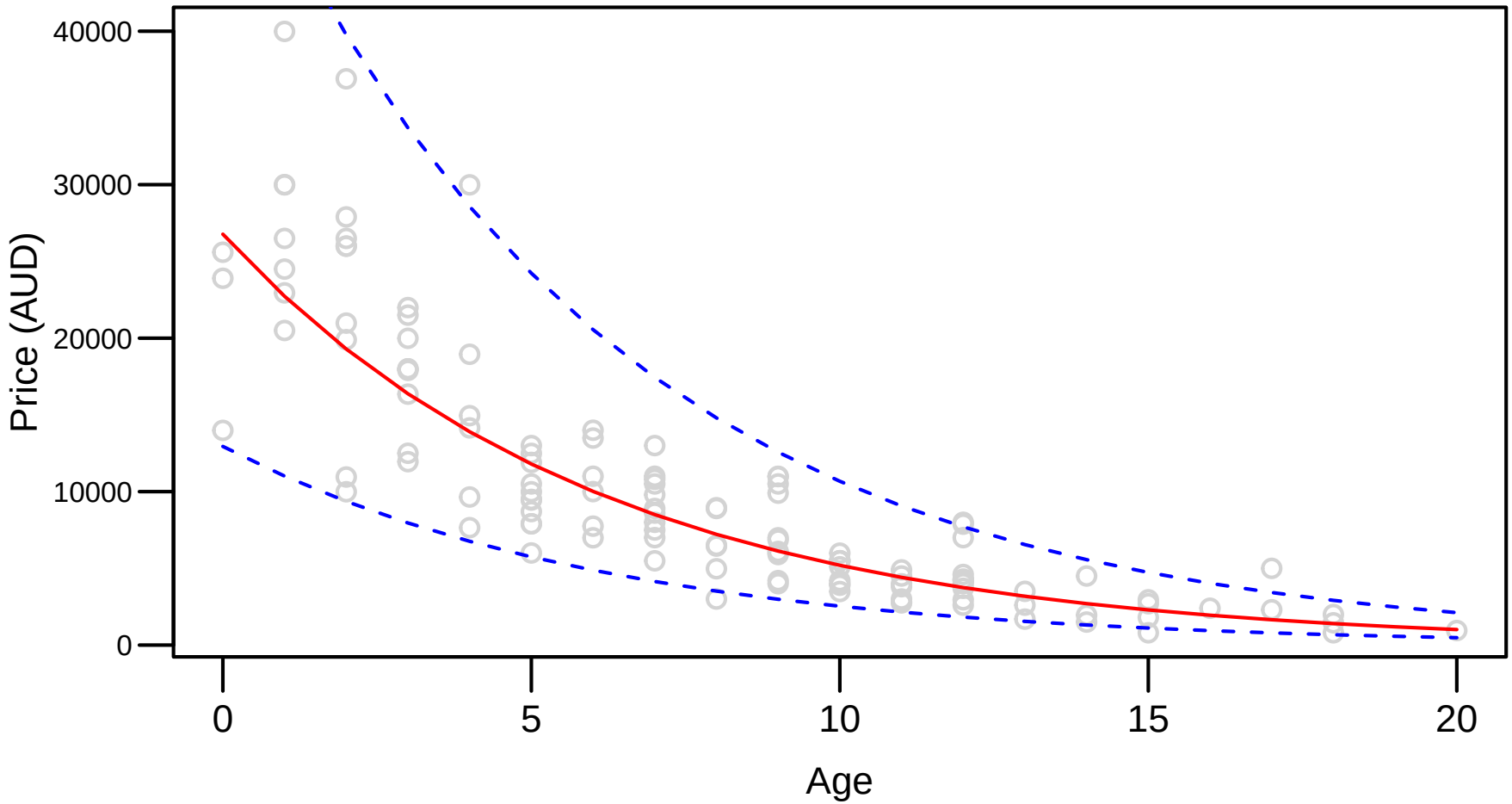
Predictions from multiplicative price vs age model

Let us now construct prediction intervals for cars that are 0 – 20 years old:

```
> plot(price~age, col="light grey",data=Mazda.df)
>
> ## Create a data.frame which has car ages from 0 to 20
> pred.df=data.frame(age=0:20)
>
> ## Exponentiate the fitted values - in the same order as above
> preds = exp(predict(LogPriceAge.fit,pred.df, interval="prediction"))
>
> lines(0:20, preds[,"fit"],col="red") ## Predicted value
> lines(0:20, preds[,"lwr"],col="blue", lty=2) ## Lower bound
> lines(0:20, preds[,"upr"],col="blue", lty=2) ## Upper bound
```

Multiplicative model with numerical explanatory variable

Predictions from multiplicative price vs age model...



Notice how the fitted line is at the half way value (median) and is robust against extreme values.

Multiplicative model with numerical explanatory variable

Executive summary: Mazda price vs age

We were interested in how the price of Mazda cars changed as they aged.

We estimate that the median value of a new car is about \$26,800 (we are confident the median is somewhere between \$23,600 and \$30,400) and that for every year the car ages the median price depreciates at about 15.1% per year, and we are confident it is somewhere between 16.3 and 13.9%

Take home message:
Do not buy new cars!!!...unless you can truly afford the depreciation hit.

Mazda price vs. age

Multi-year depreciation

Most of us keep our cars for more than a year, so we might be interested to know how much Mazdas depreciate over a 5 year period, say.

The CI for 5-year depreciation is obtained by raising the CI (for 1-year depreciation) to the power of 5. This is an intuitive thing to do, as depreciation acts multiplicatively.

```
> exp(confint(LogPriceAge.fit)[2,])^5
    2.5 %    97.5 %
0.4109832 0.4723896
```

As a percentage change this is

```
> 100*(exp(confint(LogPriceAge.fit)[2,])^5-1)
    2.5 %    97.5 %
-58.90168 -52.76104
```

Ouch, the median price of Mazdas drops between 52.8% and 58.9% over 5 years.

Mazda price vs. age...

Multi-year depreciation...

Equivalently, the above CI can be obtained by back-transforming using five times the estimated effect.

```
> exp(5*confint(LogPriceAge.fit)[2,])
    2.5 %    97.5 %
0.4109832 0.4723896
```

As a percentage change this is

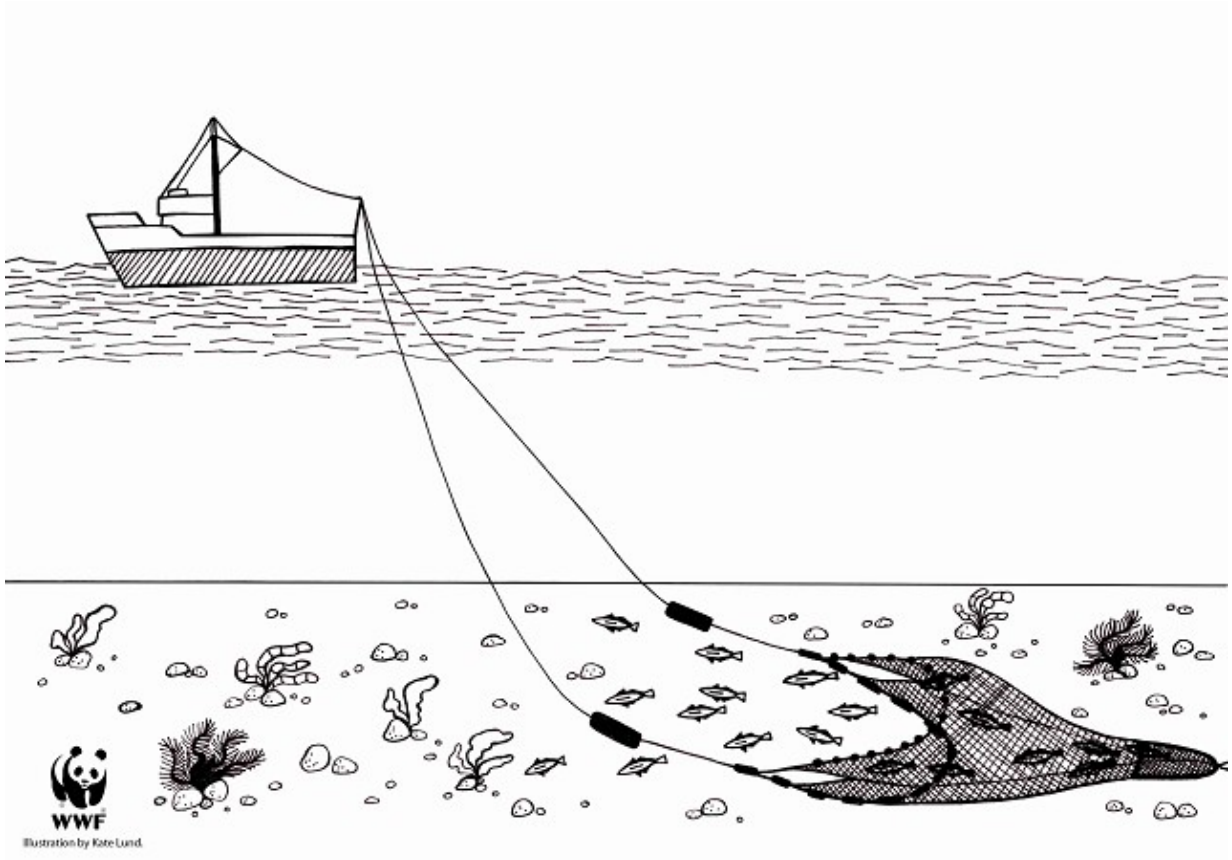
```
> 100*(exp(5*confint(LogPriceAge.fit)[2,])-1)
    2.5 %    97.5 %
-58.90168 -52.76104
```

Section 6.5
Example 2: Multiplicative model with categorical explanatory variable

Multiplicative model with categorical explanatory variable

Trawl bycatch

It was of interest to compare the amount of bycatch caught by two types of fishing trawl.

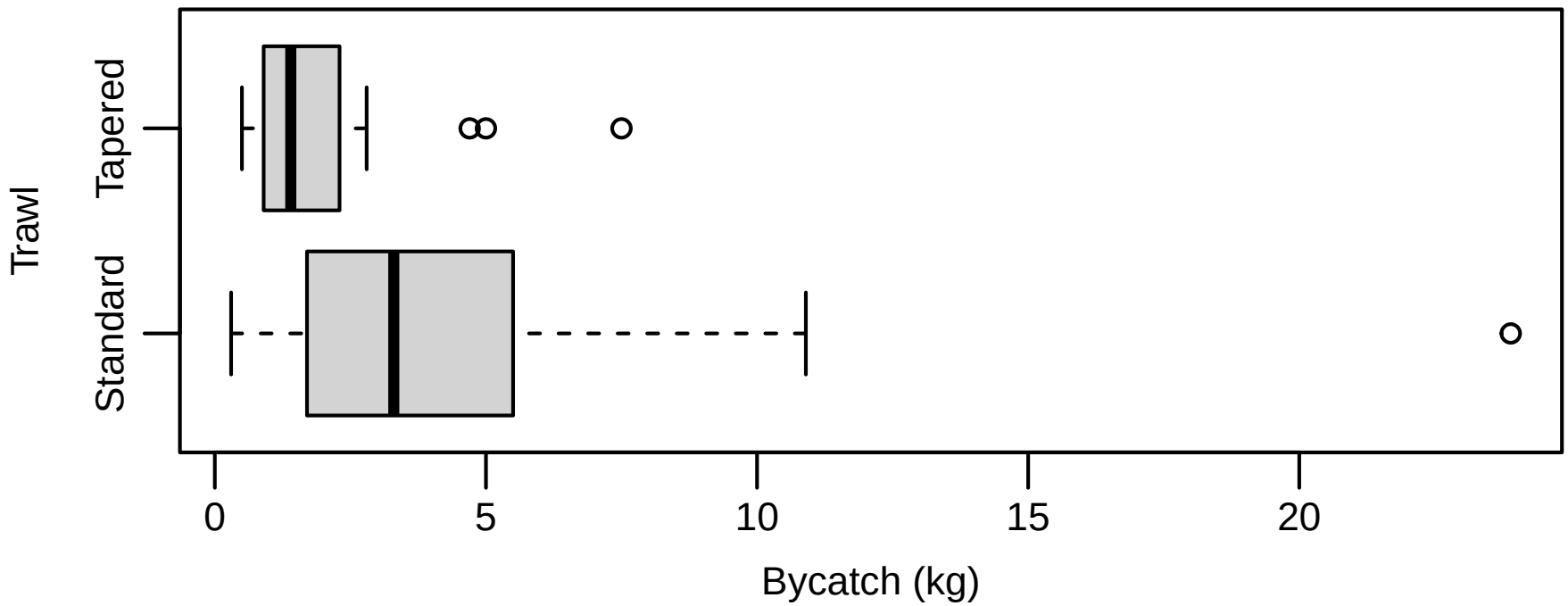


Intuition: Would you expect the effect of trawl type to be additive or multiplicative?

Multiplicative model with categorical explanatory variable

Trawl by-catch...

```
> Bycatch.df=read.table("Data/Bycatch.txt",header=T)
> boxplot(Bycatch~Trawl,data=Bycatch.df, horizontal=T,xlab="Bycatch (kg)")
```



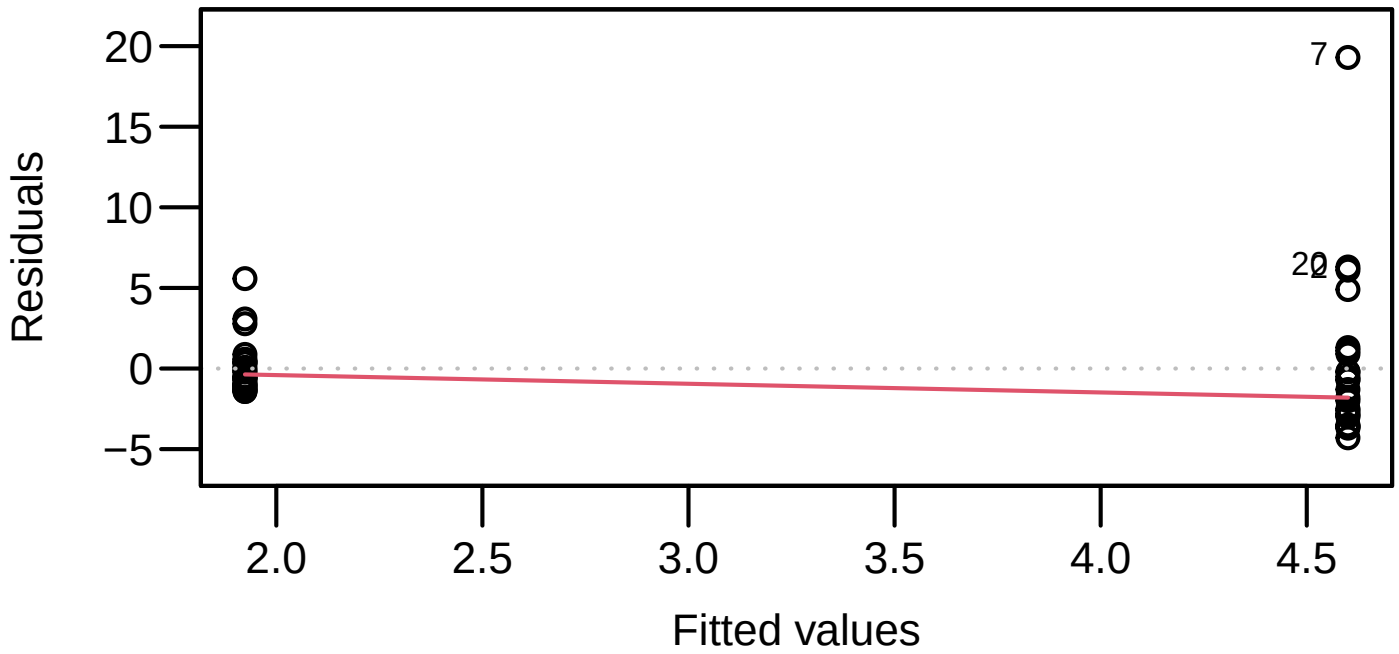

```
> summaryStats(Bycatch~Trawl,data=Bycatch.df)
         Sample Size  Mean Median  Std Dev Midspread
Standard          25 4.600    3.3 4.983138       3.8
Tapered           25 1.924    1.4 1.643999       1.4
```

Multiplicative model with categorical explanatory variable

Trawl by-catch...

This seems to confirm our intuition that these data should be modelled on the log scale. We will play stupid for now, and fit a linear model to the raw data....

```
> Trawl.lm=lm(Bycatch~Trawl,data=Bycatch.df)
> plot(Trawl.lm,which=1)
```

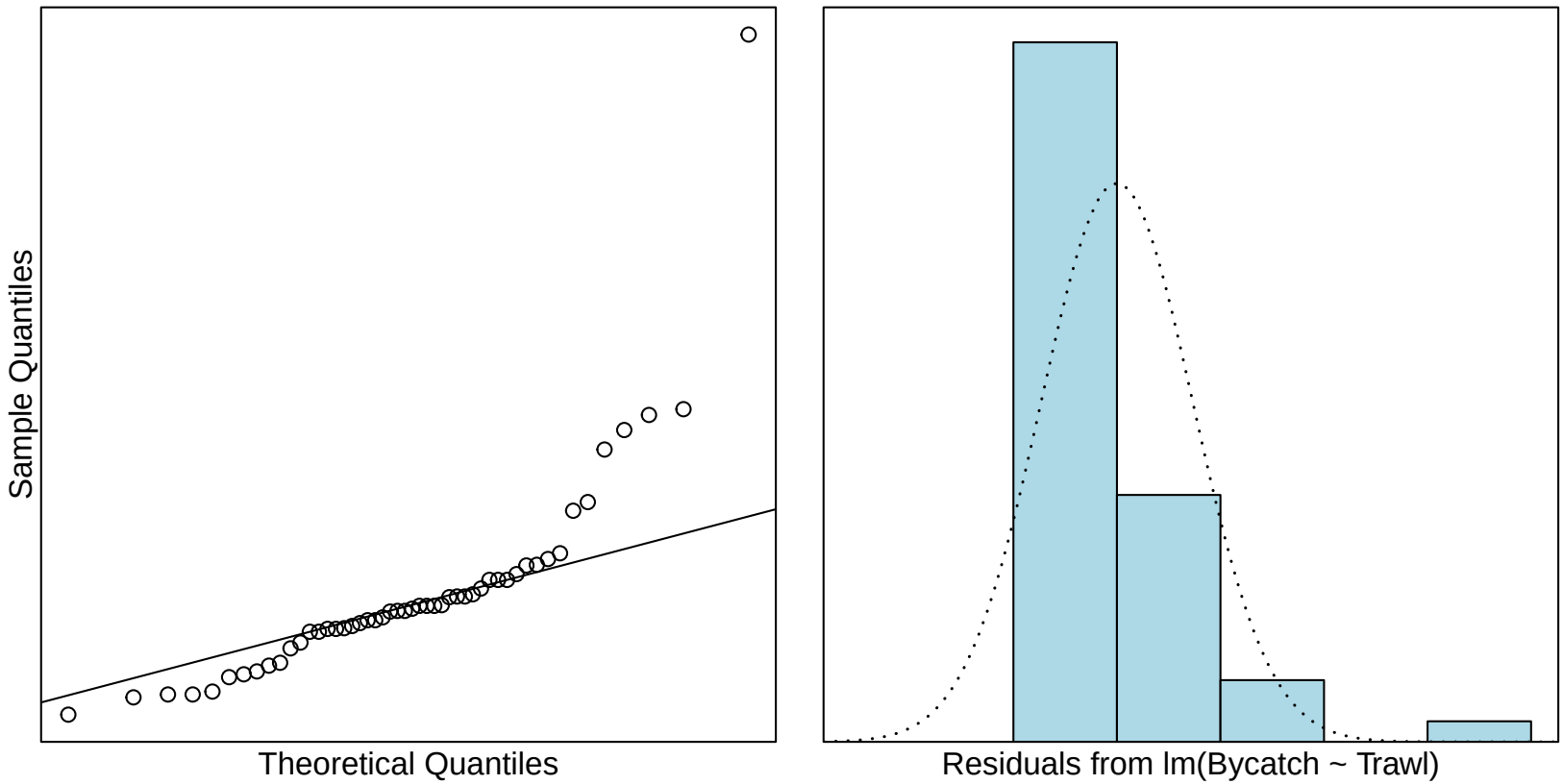


The **EOV** assumption is very doubtful. Normality?

Multiplicative model with categorical explanatory variable

Trawl by-catch...

```
> normcheck(Trawl.lm)
```

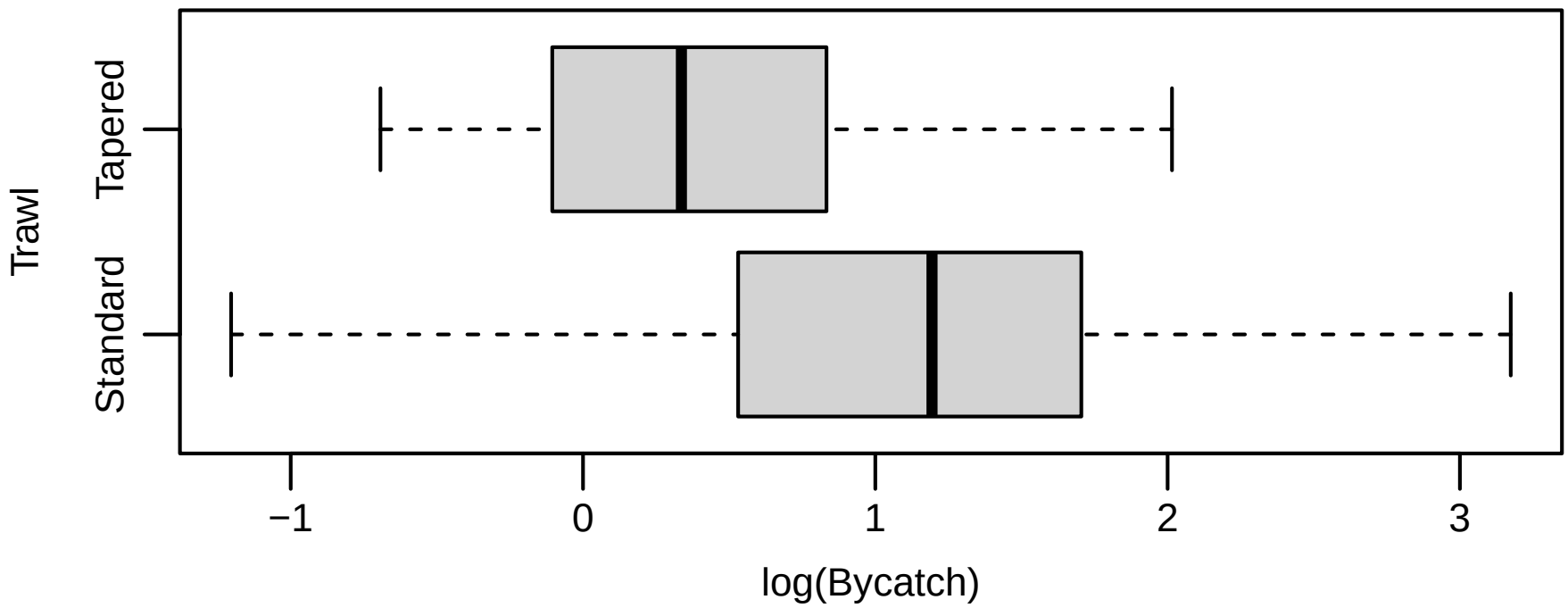


These plots of the residuals confirm that **EOV** and normality do not hold.

Multiplicative model with categorical explanatory variable

Using logged trawl by-catch

```
> boxplot(log(Bycatch)~Trawl,data=Bycatch.df,horizontal=T,xlab="log(Bycatch)")
```

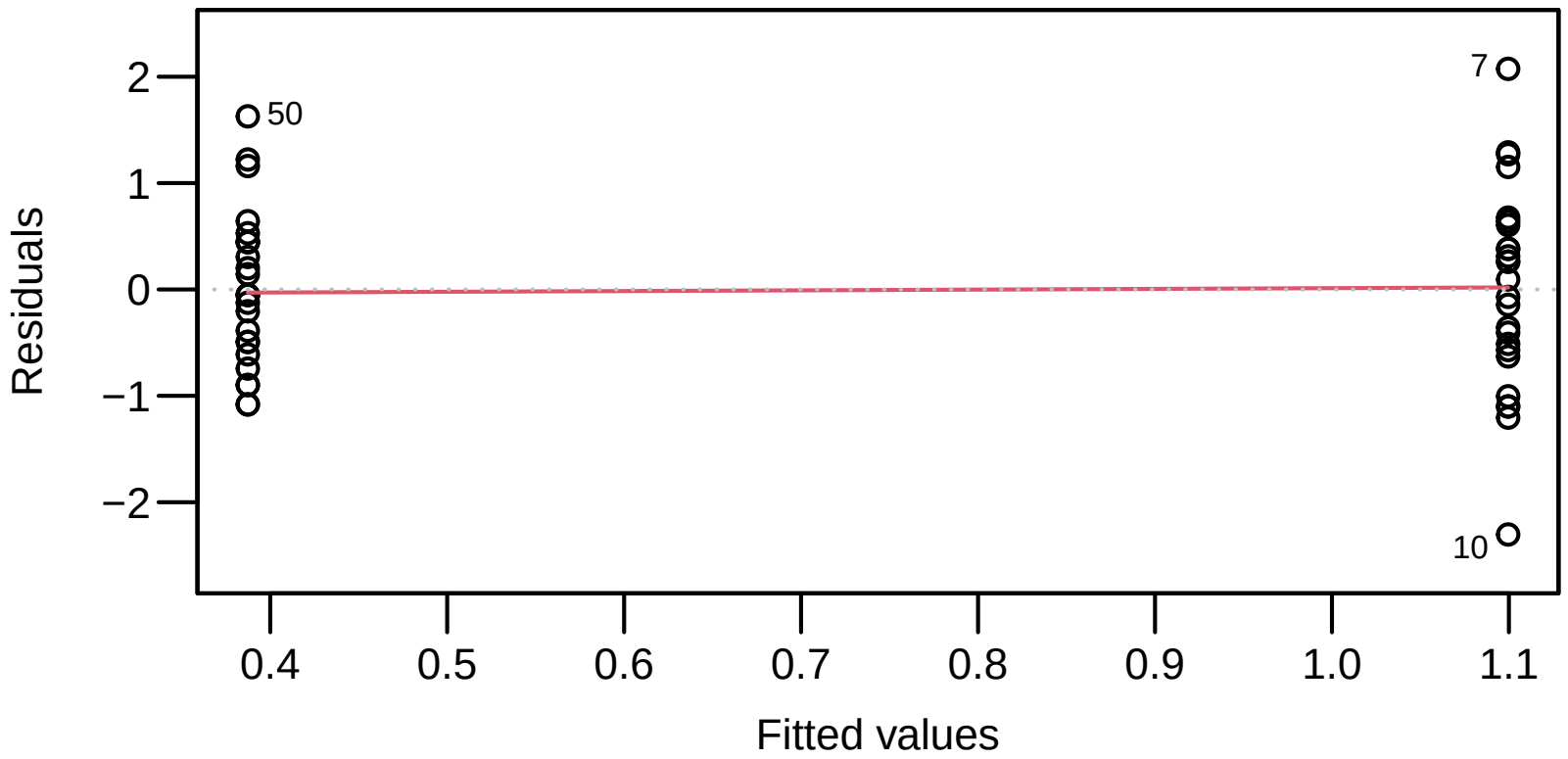


Looking much better.

Multiplicative model with categorical explanatory variable

Using logged trawl by-catch...

```
> Trawl.lmlog=lm(log(Bycatch)~Trawl,data=Bycatch.df)
> plot(Trawl.lmlog,which=1)
```

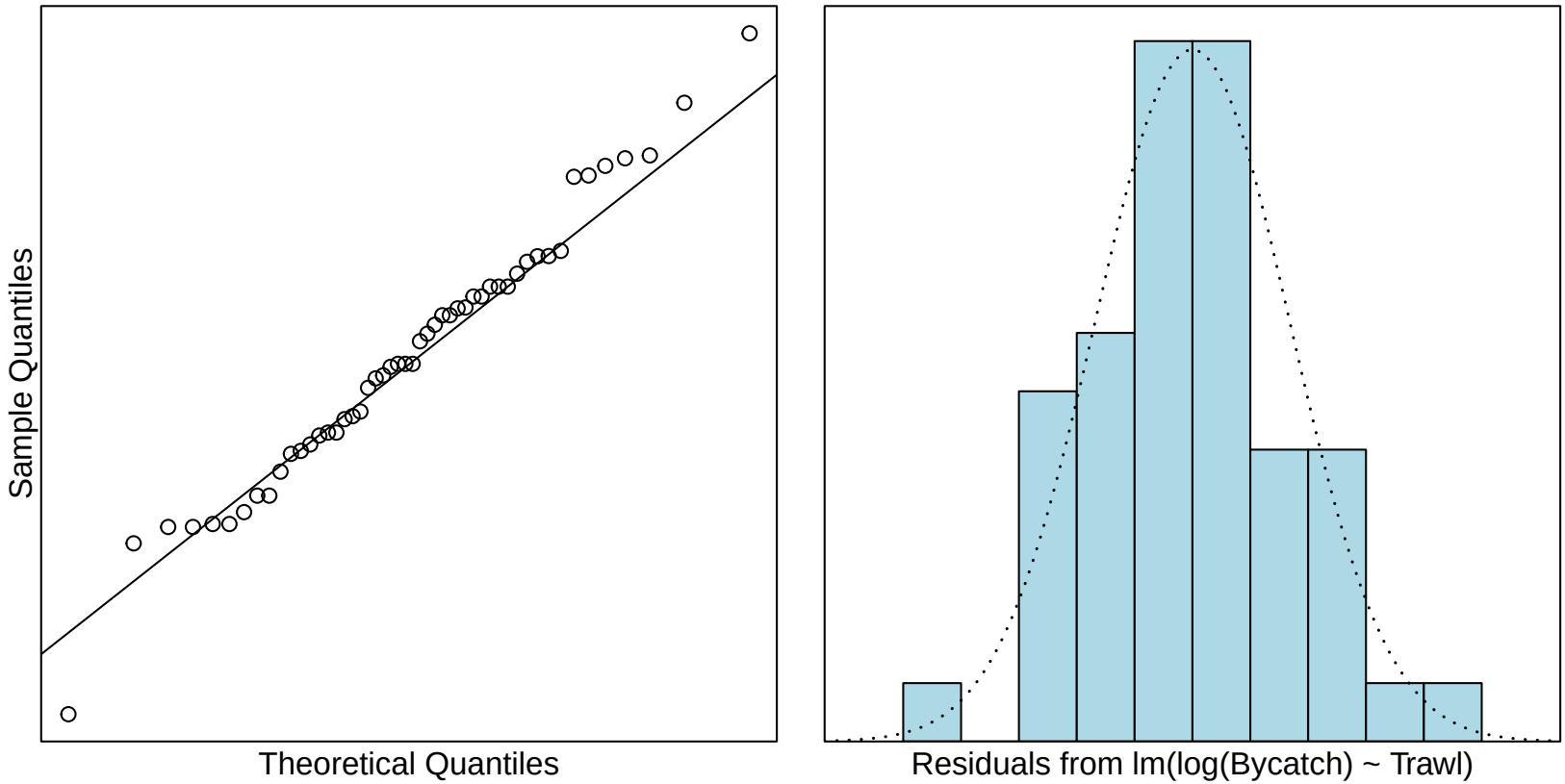


Sweet as.

Multiplicative model with categorical explanatory variable

Using logged trawl by-catch...

```
> normcheck(Trawl.lmlog)
```

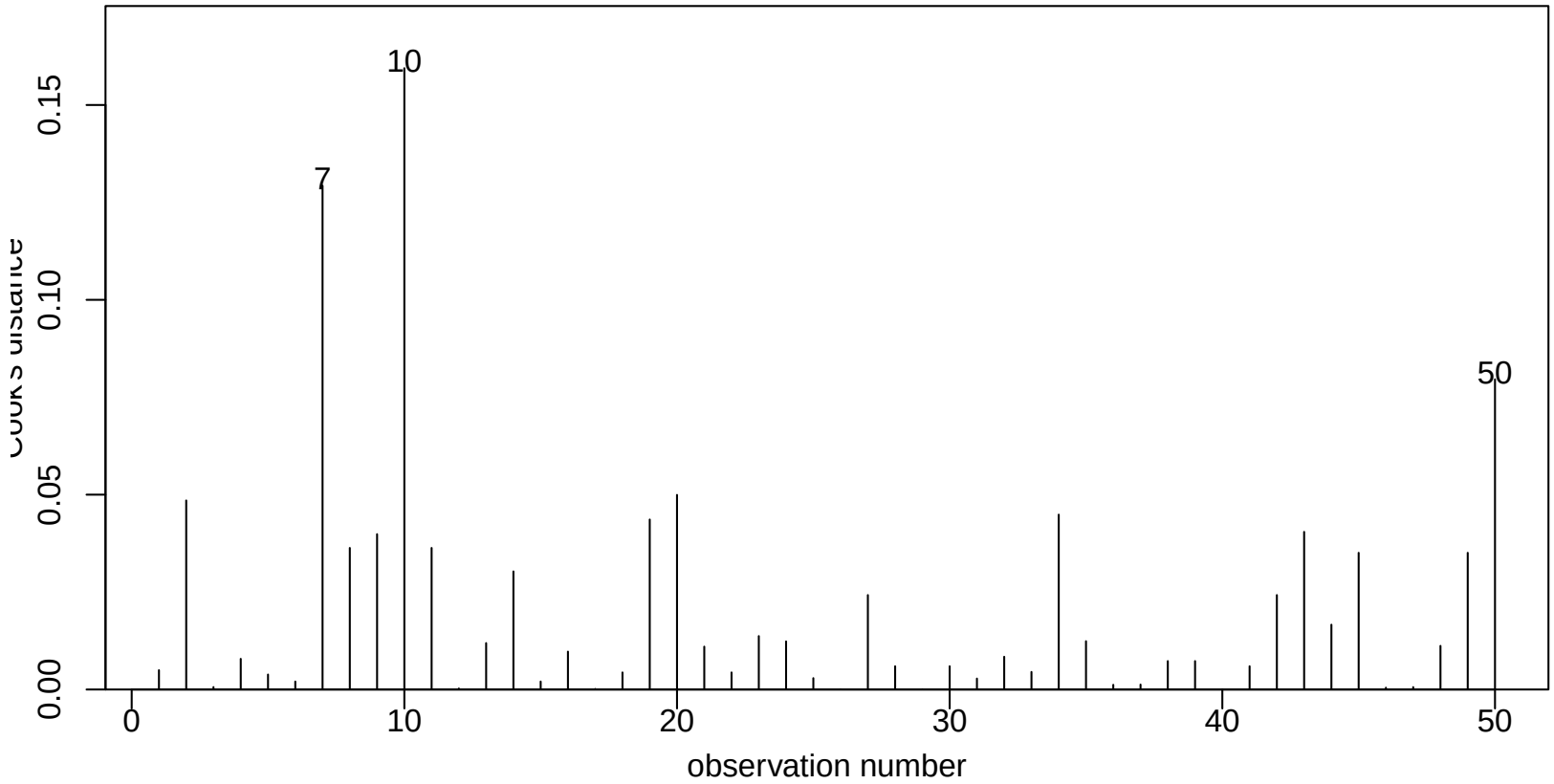


Looking good.

Multiplicative model with categorical explanatory variable

Using logged trawl by-catch...

```
> cooks20x(Trawl.lmlog)
```



No problems.

Multiplicative model with categorical explanatory variable

Results: Using logged trawl by-catch

Assumptions are satisfied. We can trust the fitted model.

```
> summary(Trawl.lmlog)
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    1.0996     0.1700   6.469 4.79e-08 ***
TrawlTapered  -0.7122     0.2404  -2.963  0.00473 **
---
Residual standard error: 0.8498 on 48 degrees of freedom
Multiple R-squared:  0.1546,Adjusted R-squared:  0.137
F-statistic:  8.78 on 1 and 48 DF,  p-value: 0.004728
```

There is a statistically significance effect of trawl type (P-value ≈ 0.005). However, our model only explained 15% of the variability in the logged data and will not be very good for prediction.

Multiplicative model with categorical explanatory variable

Interpretation: Using logged trawl by-catch

We need to back-transform to the raw data scale to make sense of the above results:

```
> exp(confint(Trawl.lmlog))
                 2.5 %    97.5 %
(Intercept)  2.1336329 4.2261691
TrawlTapered 0.3025531 0.7953873
```

The median by-catch using the tapered trawl was estimated to be between 30% and 80% that of the standard trawl.

Alternatively,

```
> 100*(exp(confint(Trawl.lmlog)[2,])-1)
    2.5 %    97.5 %
-69.74469 -20.46127
```

the median by-catch using the tapered trawl was estimated to be between 20% and 70% smaller than when using the standard trawl.

Section 6.6
Closing remarks and relevant R-code

Closing remarks

NOTE: Any of the types of linear model that you encounter *could* be applied to logged data if that is more appropriate, either due to rationale and/or due to right-skewness in the data.

In all cases you will follow the same procedure:

1. Fit the linear model to the logged data.

2. Back-transform (i.e., exponentiate) the relevant estimated coefficients, and confidence or prediction intervals.

3. **Remember** that the effect is multiplicative, and that back-transformed coefficients and CIs are for the median.

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

If any of the following hold:

- A multiplicative effect of the explanatory variables seems appropriate
- Right skewness of the variability (inspect the residuals to check)
- A funnel effect in the plot of residuals vs fitted values

then a log transformation of the response variable y may be appropriate. The usual steps (fitting a linear model and assumption checking, CIs, etc) are then applied to the logged response.

```
> Trawl.lmlog=lm(log(Bycatch)~Trawl,data=Bycatch.df)
> # then check if it's okay
> plot(Trawl.lmlog,which=1)
```

Confidence intervals can be back-transformed and interpreted as a multiplicative increases/decreases - in this case relative to baseline:

```
> exp(confint(Trawl.lmlog))
                 2.5 %    97.5 %
(Intercept)  2.1336329 4.2261691
TrawlTapered 0.3025531 0.7953873
```

and we interpret this with respect to change in the median value of y.