Chapter 2: The basics of simple linear regression

STATS 201/8

University of Auckland

Learning outcomes

In this chapter you will learn about:

- Fitting the linear model by minimizing the sum of the squared residuals
- The standard assumptions of the linear model
- Model checks of these assumptions – independence, EOV and normality
- Checking for points of undue influence using `cooks20x`
- Making inference from your model (provided the assumptions check out)
- Confidence intervals for population-level inference
- Prediction intervals for individual-level inference
- A recipe for subsequent analyses and relevant `R`-code.

Section 2.1
Fitting the linear model by minimizing the sum of the squared residuals

The fitted model

Fitted values and residuals

Recall, our simple linear model is $y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$, where β_0, β_1 and ε_i are all unknown. After we fit this model to data, we can write

$$\begin{aligned} y_i &= \hat{y}_i + r_i \\ &= \left(\hat{\beta}_0 + \hat{\beta}_1 x_i\right) + r_i \end{aligned}$$

The two components in the above formulation are:

- The fitted (or predicted) value, $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$. This is the estimated value of $E[Y_i|x_i] = \beta_0 + \beta_1 x_i$.
- And what is left over, the **residual**, r_i, the estimated value of ε_i.

A natural question is: How are $\hat{\beta}_0$ and $\hat{\beta}_1$ estimated from the data???

We will answer this question in the next few slides in the context of our exam vs test mark example.

Exam vs. Test marks...

Fitted values and residuals...

First, let us examine one student in particular – student (observation) number $i = 21$.

```
> ## Student # 21
> Stats20x.df = read.table("Data/STATS20x.txt", header=TRUE)
> Stats20x.df[21,]

   Grade Pass Exam Degree Gender Attend Assign Test  B  C MC Colour Stage1
21     A  Yes   77    BSc Female     No   16.4 12.7 15 18 22  Green      A
   Years.Since Repeat
21         0.5     No

> examtest.fit = lm(Exam ~ Test, data = Stats20x.df)
```

She got 12.7 out of 20 for the `Test` and 77 for an exam mark. Again if we say $y =$ `Exam` and $x =$ `Test` then for her: $y_{21} = 77$ and $x_{21} = 12.7$.

Exam vs. Test marks...

Fitted values and residuals...

According to the fitted model her fitted/predicted value is

$$\hat{y}_{21} = \hat{\beta}_0 + \hat{\beta}_1 x_{21} = 9.08 + 3.79 \times 12.7 = 57.1657$$

so she seems to have done better than predicted! The residual value for her is:

$$r_{21} = y_{21} - \hat{y}_{21} = 77 - 57.1657 = 19.8343.$$

These calculations are done for all $i = 1, 2, \ldots, n = 146$ students.

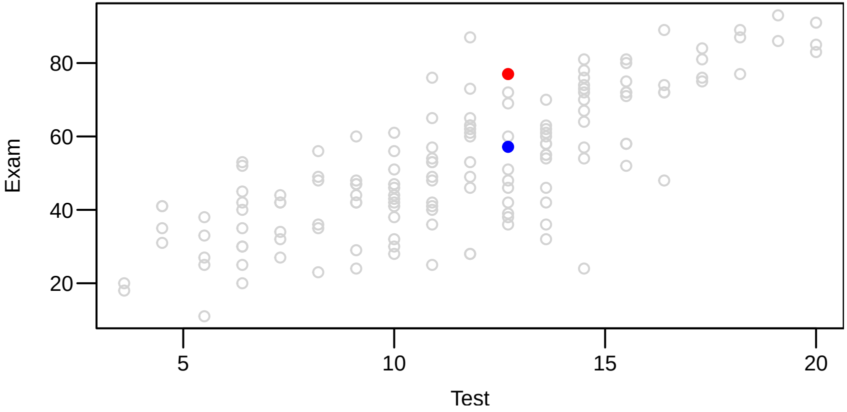
`R` did these calculations when we created the `lm` object `examtest.fit`. Here is how to extract this information for student $i = 21$:

```
> resid(examtest.fit)[21]

     21
19.8343

> fitted(examtest.fit)[21]

     21
57.1657
```

Exam vs. Test marks...

Fitted values and residuals...

Here is her observed value y_{21} - (red point) and her associated fitted[1] value $\hat{y}_{21}$ marked (blue point).

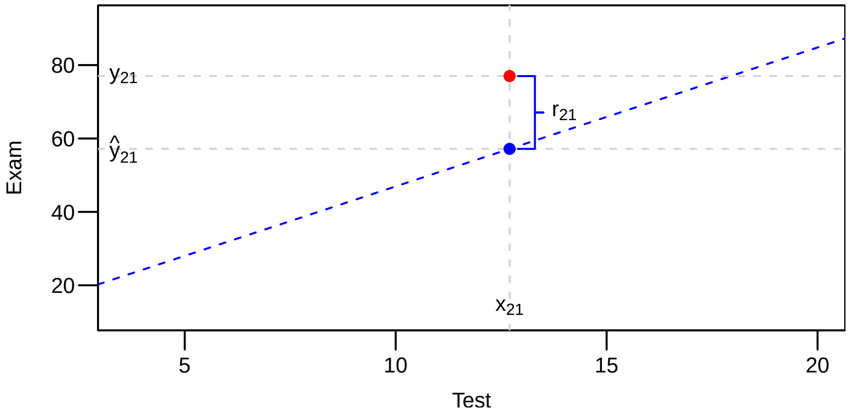


[1] We frequently use these expressions "fitted" values and "predicted" values interchangeably – with some exceptions – more later!

Exam vs. Test marks. . .

Fitted values and residuals. . .

In greater detail:



The residual is the vertical distance between the observed exam mark and the predicted exam mark.

Calculating $\hat{\beta}_0$ and $\hat{\beta}_1$

Least squares

The estimates, $\hat{\beta}_0$ and $\hat{\beta}_1$, are obtained as the values of β_0 and β_1 that minimise the least squares equation:

$$\sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2.$$

This minimized value is

$$\sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2 = \sum_{i=1}^{n} r_i^2.$$

In effect this minimises the sum (over all observations) of the squared residuals. Hence the name **"least squares"**.

This can be solved using a bit of calculus and/or linear algebra taught in, say, MATHS 208. A bit more detail is also given in STATS 330.

Exam vs. Test marks...

Residual sum of squares and R^2.

Recall that we've already seen the sum of the squared residuals from the fitted model, $\sum_{i=1}^{n} r_i^2$. It is the residual sum of squares (RSS). It can be calculated using the R code

```
> sum(resid(examtest.fit)^2)

[1] 20901.08
```

Similarly, the RSS of the null model is the total sum of squares,

```
> sum(resid(lm(Exam~1,data=Stats20x.df))^2)

[1] 50585.78
```

Note that, compared to the null model, the RSS is reduced by 59% when test score is in the model. That is, the R^2 of `examtest.fit` is 0.59.

Section 2.2
The standard assumptions of the linear model

General assumptions of all linear models

In this chapter we will discuss the set of assumptions we make when using linear models for inference.

We will show how these assumptions relate specifically to the simple linear model we have seen so far.

However, whatever we learn here is also applicable to the other types of linear models that we will encounter.

In later chapters we will encounter situations where these assumptions cannot hold. However, we will find appropriate ways to still use a linear model.

General assumptions of all linear models…

The underlying assumptions

Let us state and examine our model a little more carefully.

Our simple linear regression model is

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \quad \varepsilon_i \overset{iid}{\sim} N(0, \sigma^2)$$

where each observation is indexed[2] with a subscript i and iid stands for **i**ndependently and **i**dentically **d**istributed.

The above model formula is saying that

$$\varepsilon_i = y_i - \beta_0 + \beta_1 x_i$$

is distributed according to

$$\varepsilon_i \overset{iid}{\sim} N(0, \sigma^2)$$

[2] The indexing variable i is simply a labelling device. So, in out test vs exam example where $y =$ `Exam` and $x =$ `Test` we are saying that each student (observation) has a label associated with them from $i = 1, 2, 3 \ldots, n = 146$.

General assumptions of all linear models...

The underlying assumptions...

The model assumptions are with regard to the random variability component

$$\varepsilon_i \overset{iid}{\sim} N(0, \sigma^2)$$

1. **Independent:** We require every observation to be independent of every other observation.[3] This requires that the data are collected in an appropriate manner, such as a simple random sample. In practice, achieving truely independent data can be challenging.
2. **Identically** distributed: The ε_i must all come from the same distribution, and this distribution must have mean of 0. All ε_i must necessarily have the same variance, which we refer to as the equality of variance, (**EOV**), assumption.
3. **Distributed** normally: The ε_i must be from a normal distribution, at least approximately.

[3]If the y_i are independent then so too are the ε_i.

Model assumptions

Exam vs. Test marks

Let's consider the underlying assumptions about $\varepsilon_i \overset{iid}{\sim} N(0, \sigma^2)$ in the context of our ongoing example.

In decreasing order of importance:

- **Independent:** It was an invigilated exam so the variabilities in the exam scores of students should all be independent.
- **Identically** distributed: If the **linear** relationship is reasonable then
  - The residuals[4] will be randomly scattered around 0.
  - The residuals should have roughly constant scatter, i.e., satisfy the equality Of Variance (EOV) requirement.
- **Distributed** normally: The residuals should be (at least roughly) normally distributed – only check this if the first two assumptions are validated.

[4]Recall, the residuals r_i are our estimates of ε_i

Very important aside!

If the linear model assumptions do not hold, it **does not** mean we cannot do anything.

When assumptions are not satisfied there are often techniques that we can use to transform our data and/or model so that the assumptions **will** be satisfied. We will see some of these techniques later in this course.

There are also other techniques that can “free up” some of the assumptions. For example, lack of independence in time series data is covered later in this course, and other techniques are seen in more advanced courses.

Section 2.3
Checks of model assumptions – independence, EOV and normality

EOV check: Plotting residuals vs fitted values

Let us check the **EOV** assumption. We are assuming that what cannot be explained has random constant scatter about zero regardless of the fitted value.

That is, for each fitted (or predicted) value we have a similar distribution for what cannot be explained. We can check this by looking at each fitted (predicted) value versus its associated residual value.

Our fitted model is:

$$y_i = \hat{y}_i + r_i = \left(\hat{\beta}_0 + \hat{\beta}_1 x_i\right) + r_i.$$

It is useful to draw a scatter plot with $\hat{y}_i$ on the horizontal axis and r_i on the vertical axis.

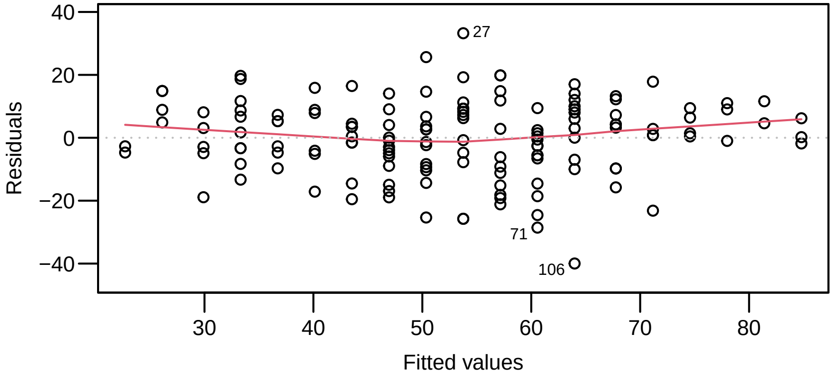
This is more generally known as a *residuals vs fitted values* plot.

Exam vs. Test marks...

EOV check: Plotting residuals vs fitted values...

Producing a scatter plot of residuals versus fitted values is easy[5].

```
> plot(examtest.fit, which=1)
```



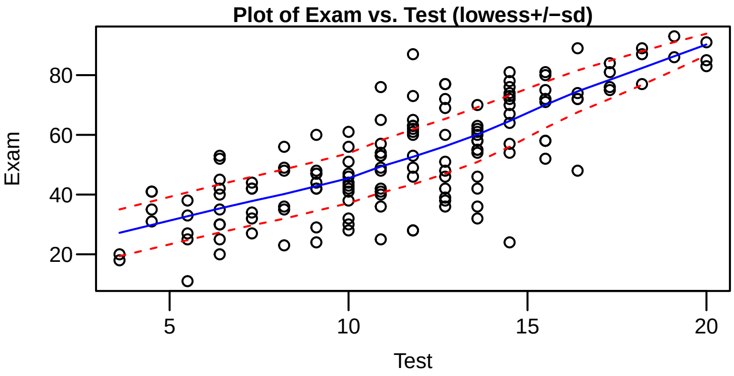
[5]You can also use the s20x function eovcheck, i.e., eovcheck(exam.fit)

Exam vs. Test marks. . .

EOV check: Plotting residuals vs fitted values. . .

The above residual plot exhibits a patternless band of random points - this is to be expected since the original scatter plot of the data had a straight line trend and reasonably constant scatter:

```
> trendscatter(Exam~Test, data =Stats20x.df)
```



Exam vs. Test marks. . .

EOV check: Plotting residuals vs fitted values. . .

The plot of residuals versus fitted values confirms reasonably constant scatter (variance).

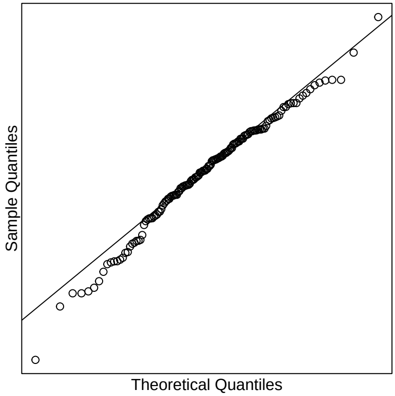
Note that at the high end of exam/test scores there are not that many students (unfortunately!) so the relative lack of spread is not surprising.

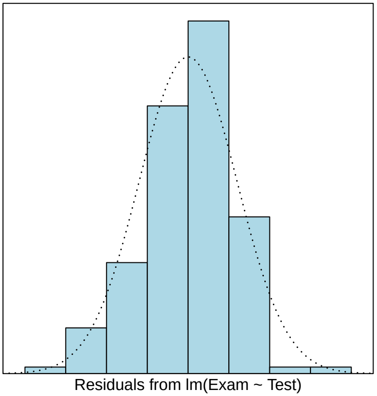
Exam vs. Test marks. . .

Normality of residuals

Having validated the **EOV** assumption, let us perform the normality check by examining the residuals. We have written a function in R called normcheck to do just that:

```
> normcheck(examtest.fit)
```





Exam vs. Test marks. . .

Normality of residuals

The histogram of the residuals (right plot) tells us we are okay since it closely matches a normal bell-shaped curve (shown by the dashed line).

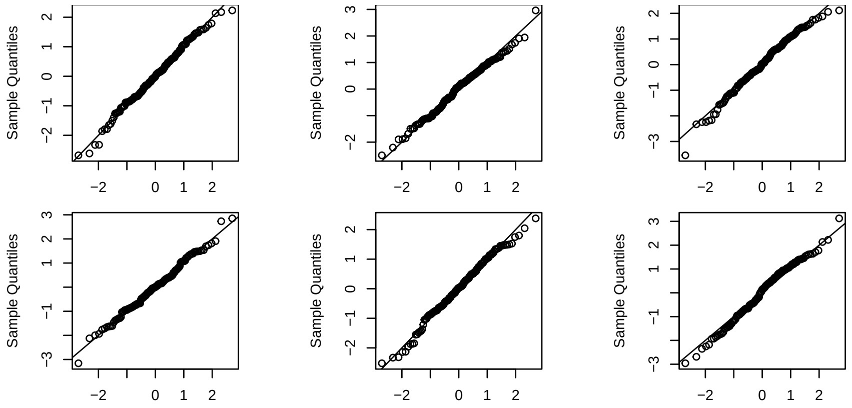
The normal quantile-quantile-plot (`qqplot`) in the left hand plot gives further insight. In this example, the 146 theoretical quantile values on the x-axis can be regarded as the values that would give a near perfect bell-shaped histogram (i.e., would best match the dashed bell-shaped curve in the right hand plot). Ideally, our residuals should match these theoretical quantiles and lie approximately on the straight line shown on the `qqplot`.

It can be difficult to judge if a qq-plot is sufficiently close to a straight line or not. Some intuition can be gained by seeing how the qq-plot looks when the assumptions are satisfied, as shown on the next slide. In comparison, the above qq-plot does look a little atypical, but not enough to cause major concerns.

Exam vs. Test marks. . .

qq-plot intuition

```
> for(i in 1:6) {
+   y=rnorm(146)
+   qqnorm(resid(lm(y~1)),main=""); abline(0,1) }
```



Exam vs. Test marks. . .

A further check: Influence

Before we rush in and conclude that all is well, there is one other concern that we should investigate, namely *influence*.[6]

Essentially, we need to check whether the fit is highly influenced by any single (or small group of) observation(s).

[6] This is not a model assumption per se, but is a prudent model check.

Section 2.4
Checking for points of undue influence using `cooks20x`

Points of undue influence

The fitted model is based on minimizing the sum of the squared residuals. Because of this, a single observation with a very large residual can have a big influence on the overall fit. We can be misled if we assume all is fine when one (or several) data points are unduly influential, and perhaps of questionable validity.

Moreover, a data point can also be influential on the model fit by having an extreme x value, since it can apply a lot of "torque" to the fit.

Exam vs. Test marks. . .

Points of undue influence. . .

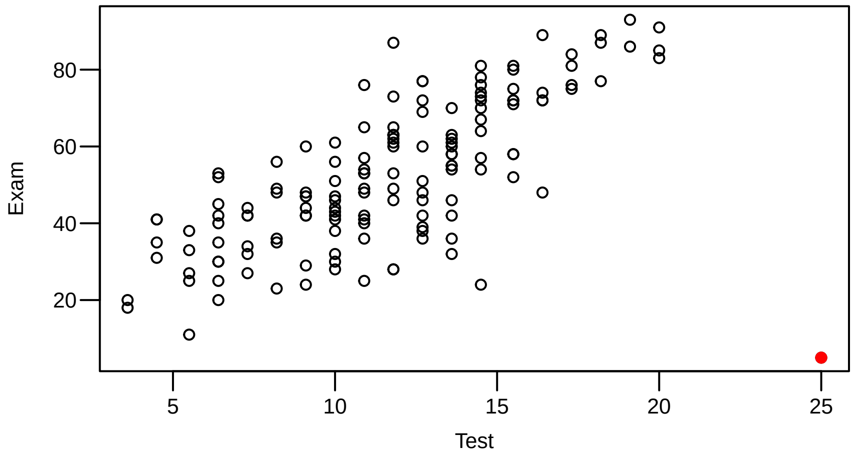
To illustrate: Imagine a student had mistakenly been awarded 25 out of 20 in their `Test`, yet obtained just 5 in the final exam.

Here is some tricky `R` code to add this incorrect data point to our dataframe:

```
> n=nrow(Stats20x.df)
> ## Add extra point by repeating last observation, n(=146)
> Stats20xnew.df=Stats20x.df[c(1:n,n),]
> ## Replace with new test /Exam #'s
> Stats20xnew.df[n+1,c("Test", "Exam")]=c(25,5)
> ## Plot our new dataset
> plot(Exam~ Test, data=Stats20xnew.df)
> ## Mark our new observation
> points(25,5,pch=19,col="red")
```

Exam vs. Test marks. . .

Points of undue influence. . .



The bogus student is shown on the plot as the red point!

Exam vs. Test marks. . .

Points of undue influence. . .

Here is the new model fitted to the altered data.

```
> examtest.fit2=lm(Exam~Test, data=Stats20xnew.df)
> coef(summary(examtest.fit2))
```

```
             Estimate Std. Error   t value     Pr(>|t|)
(Intercept) 15.237389  3.7171579  4.099204 6.875383e-05
Test         3.200551  0.3022693 10.588407 9.162607e-20
```

Compare this to the original fit:

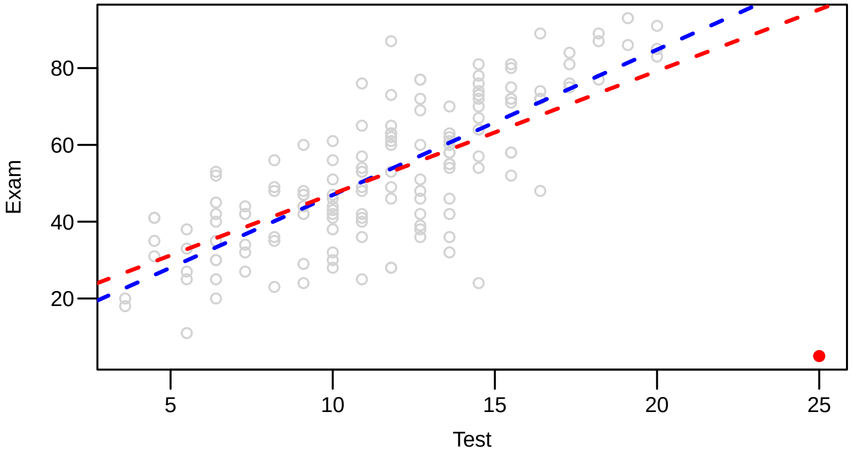
```
> coef(summary(examtest.fit))

            Estimate Std. Error   t value     Pr(>|t|)
(Intercept) 9.084463  3.2204410  2.820876 5.465681e-03
Test        3.785924  0.2647333 14.300897 1.985079e-29
```

Note how $\hat{\beta}_1$ has changed from 3.79 to 3.20. This is a change of more than two standard errors (here, we are referring to the standard error from the original fit, 0.265).

Exam vs. Test marks. . .

Points of undue influence. . .

Here is the old model fitted line (blue) compared to this new model (red).



Exam vs. Test marks...

Points of undue influence...

How does this change predictions?

In this model we would make a prediction using the following estimates (to 2 d.p.). $\texttt{Exam} = 15.24 + 3.20 \times \texttt{Test}$ compared to the old model estimates: $\texttt{Exam} = 9.08 + 3.79 \times \texttt{Test}$

Our new vs. previous estimates are now (previous estimate in brackets):

- `Test =  0` then $\texttt{Exam} = 15.24 + 3.20 \times \texttt{0} = 15.24$ (9.08),
- `Test = 10` then $\texttt{Exam} = 15.24 + 3.20 \times \texttt{10} = 47.24$ (46.94),
- `Test = 20` then $\texttt{Exam} = 15.24 + 3.20 \times \texttt{20} = 79.25$ (84.80).

Note that this has had a huge change for the extreme `Test` scores – for those who got 0 in the `Test` we predict a higher mark than before and those who got a 20 (out of 20) get a lower predicted mark than before.

Identifing points of undue influence

The addition of student 147 has drastically changed our fitted model and its predictions. This studend is so atypical of the other 146 students that we should exclude them from the analysis. In this case it is clearly a data entry mistake, but in practice it is not always so obvious.

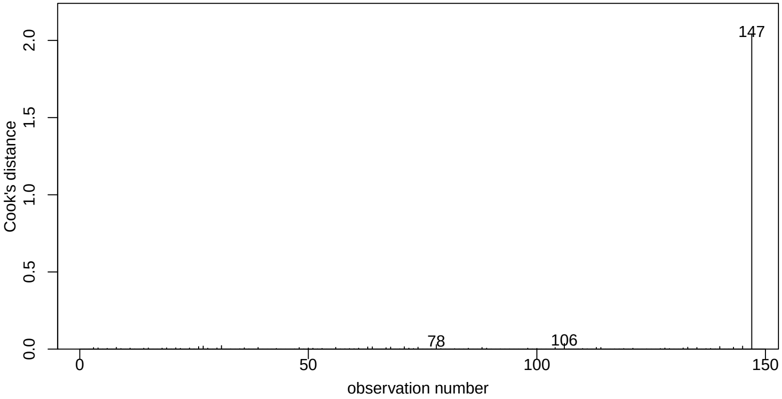
We've seem that to determine whether an observation is influential we need to remove it from the dataset and refit the model. This can be a lot of work when there a lot of observations. The Cook's distance uses linear algebra to avoid this work.

Identifing points of undue influence. . .

Cook's distance

The following Cook's distance plot gives the influence of all 147 observations in the altered dataframe. Student 147 dominates this plot and is clearly identified as being highly influential.

```
> cooks20x(examtest.fit2)
```



Identifing points of undue influence. . .

As a rough "rule of thumb"[7], an observation is deemed to be influential if:

- Removal of the observation changes any estimated parameter value by more than one standard error, or
- Its Cook's distance is greater than 0.4.

In STATS 20x we'll use the above Cook's distance threshold of 0.4. (Be aware that other courses or texts may use other thresholds, and the threshold may depend on the number of observations.)

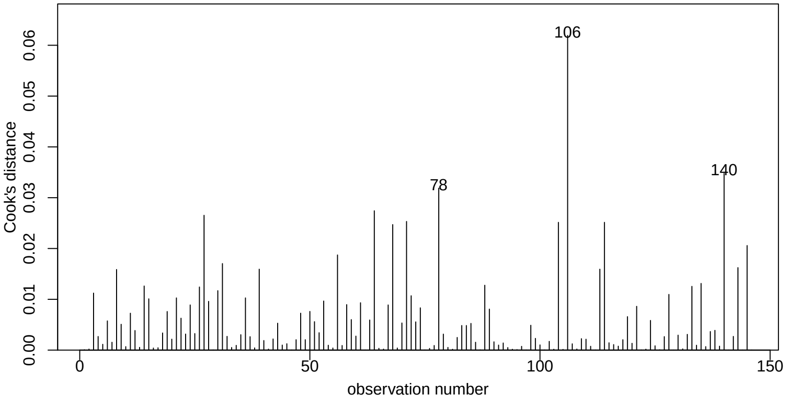
Let us get back to the original fitted model and see what its Cook's distance plot looks like.

[7] **A rule of thumb** is a principle with broad application that is not intended to be strictly accurate or reliable for every situation. It is an easily learned and easily applied procedure for approximately calculating or recalling some value, or for making some determination. Thanks Wikipedia.

Exam vs. Test marks...

Identifing points of undue influence...

```
> cooks20x(examtest.fit)
```



Exam vs. Test marks. . .

Validity of assumptions

We may now conclude that we can (mostly) trust the output of our analysis as our underlying assumptions seem reasonable and we do not have any unduly influential data points.

Section 2.5
Making inference from your model (provided the assumptions check out)

Statistical properties of the fitted linear model

Provided that **all** of the assumptions of the linear model hold, it can be shown (see STATS 310) that $\hat{\beta}_0$ and $\hat{\beta}_1$ are normally distributed[8] with expected values β_0 and β_1, respectively.

This means that the standard statistical techniques for making inference from normally distributed data can still be used.

In particular, the t-statistic is used for testing hypotheses, and the t-multiplier for constructing confidence and prediction intervals.

[8] That is, under repetition of the experiment that generated the data.

Interpreting the output

Testing for significant effects

```
> examtest.fit = lm(Exam ~ Test, data = Stats20x.df)
> summary(examtest.fit)
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   9.0845     3.2204   2.821  0.00547 **
Test          3.7859     0.2647  14.301  < 2e-16 ***
---
Residual standard error: 12.05 on 144 degrees of freedom
Multiple R-squared:  0.5868,Adjusted R-squared:  0.5839
F-statistic: 204.5 on 1 and 144 DF,  p-value: < 2.2e-16
```

So here we can see that the *P*-value associated with the `Test` variable is highly statistically significant `< 2e-16 ***`.[9]

Our underlying belief was that we could describe the relationship between exam mark and test mark using an increasing straight line. This belief has been confirmed.

[9] `2e-16` $= 2 \times 10^{-16}$, which is so small that we may just as well call it zero.

Interpreting the output...

Testing for significant effects...

The null hypothesis is always that there is no effect. In this case, that there is no relationship between test score and exam score, i.e.,

H_0: there is no relationship between `Test` and `Exam`.

Or equivalently,

$H_0 : \beta_1 = 0$, which is equivalent to saying that the data follow the `null` model $y = \beta_0 + \varepsilon$.

We will pretend that the null hypothesis is true (i.e., our working model of an increasing linear relationship between test and exam is incorrect), at least until we get evidence to the contrary.

This approach is known as the philosophy of "falsification". It is not the natural way we like to do things as humans, but is a great way to ensure we do not come to erroneous conclusions based on wishful thinking.

Interpreting the output...

Calculating the t value

Under the null hypothesis we assume $\beta_1 = 0$ and compare this to the estimated value here of $\hat{\beta}_1 = 3.7859$. As we are statisticians, we need to measure how precise this estimate is – this is measured by the standard error: $se(\hat{\beta}_1) = 0.2647$.

So under the assumption that $\beta_1 = 0$ we can say that our data tell us that we are

$$\frac{3.7859 - 0}{0.2647} = 14.301$$

standard deviations[10] away from this value. You might recognize this as the t-statistic for testing $H_0 : \beta_1 = 0$. It is the `t value` in our output.

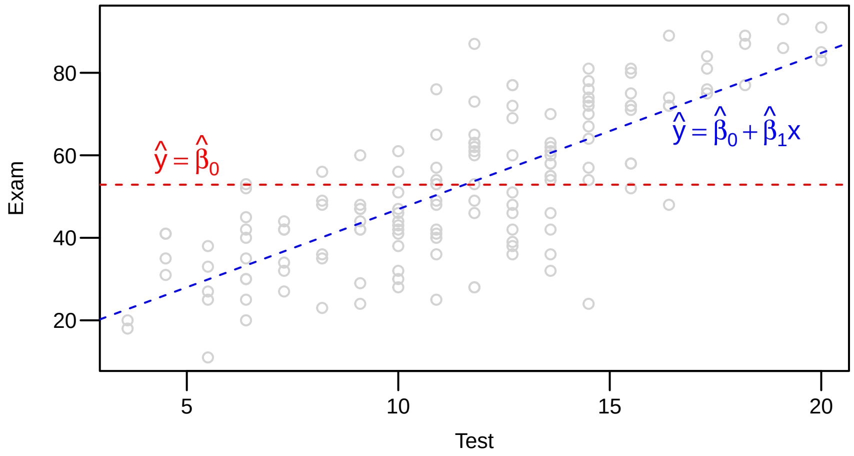
Here, we conclude that our working model was reasonable. That is, it is unreasonable to assume that `Test` had no effect on `Exam` as the chances of fluking such a result is $< 2 \times 10^{-16}$.

[10] The term "standard error" is used to refer to the standard deviation of an estimated value – in this case the standard deviation of the estimated linear effect of `Test`.

Interpreting the output...

Null vs fitted models

If y = `Exam` and x = `Test`, then here are our fitted null (the red line) and working (the blue line) models.



Statistical inference from our model

Confidence intervals for effects

We can also get confidence intervals for β_1 (and β_0 if need be) by:

```
> confint(examtest.fit)

                2.5 %    97.5 %
(Intercept) 2.719020 15.449907
Test        3.262659  4.309189
```

Here we can say that, on average, every increase in a student's test mark of 1 unit (out of 20) results in a 3.26 to 4.31 increase in the student's exam result.

Note: the default setting is 95% in `confint` – but this can be changed with the optional `level` argument. E.g., a 99% CI is given by

```
> confint(examtest.fit,level=0.99)

                0.5 %    99.5 %
(Intercept) 0.6778171 17.491110
Test        3.0948635  4.476984
```

Using confidence intervals to do hypothesis tests

We can be confident that the 95% confidence interval for β_1 contains the true value of β_1 because, under repetition of the experiment, 95% of the calculated confidence intervals will contain the true value of β_1. That is, we'd have to be pretty unlucky for a confidence interval not to contain what we are trying to estimate.

This means that if a hypothesized value of β_1 is not in our confidence interval, then we can be confident that it is not the true value of β_1. For example, if the value 0 is not in our 95% confidence for β_1 then we can reject the null hypothesis $H_0 : \beta_1 = 0$ (at the 5% level of significance). In fact, in the current example we would reject all values that are less than 3.26 or greater than 4.31

Choosing the best model

In our exam vs test mark example the effect of `Test` was highly significant. That is, the null hypothesis $H_0 : \beta_1 = 0$ was rejected, and so we used the fitted simple linear regression model `examtest.fit` for inference.

However, in situations where the null hypothesis $H_0 : \beta_1 = 0$ is **not** rejected, then we would conclude that there is little evidence of an association between x and y. In that case the null model would be our preferred model, and would be used for inference.

Also, note that we did not do a hypothesis test of the intercept, i.e.,$H_0 : \beta_0 = 0$. This is because the intercept is not an "explanatory variable" and so this hypothesis is rarely of interest. We generally leave the intercept term in the model regardless of whether it is significant or not.[11]

[11] There may be circumstances where it is meaningful to test the statistical significance of the intercept term, but these are very rare.

Section 2.6
Confidence intervals for coefficients and fitted values, and prediction intervals for individual predictions

Estimation of fitted values

Getting fitted values using predict

Earlier, we calculated fitted/predicted values for students who got $0, 10$, or 20 in their Test. Here is a way to avoid doing this by hand using the predict function.

Before using predict we have to set up a new dataframe that contains the Test values for which we want to predict Exam.

```
> ## Create data.frame of values of interest: Test = 0, 10, 20:
> ## Names of vars must be exactly the same as in the data data.frame
> preds.df=data.frame(Test=c(0,10,20))
> predict(examtest.fit, preds.df)

        1         2         3
 9.084463 46.943703 84.802942
```

These values are our estimates of the expected Exam scores for students with Test scores of 0, 10 or 20, respectively.

Estimation of fitted values...

Confidence intervals for expected values using predict

The above fitted values are **point** estimates of the expected Exam score when Test is $0, 10$, or 20. We would also like to have confidence intervals for these expected scores. This is easy:

```
> predict(examtest.fit, preds.df, interval="confidence")

        fit      lwr      upr
1  9.084463  2.71902 15.44991
2 46.943703 44.80912 49.07828
3 84.802942 79.97021 89.63568
```

Note how the CI is relatively narrow for Test = 10 compared to the extreme Test values. Why is this?

Prediction of new observations

Prediction intervals for new y values using predict

It may also be of interest to ask for an interval that predicts the Exam score for a single student (rather than an interval for the expected score, as done above). This is also easy – just ask for a prediction interval instead:

```
> predict(examtest.fit, preds.df, interval="prediction")

        fit       lwr       upr
1  9.084463 -15.56475  33.73368
2 46.943703  23.03510  70.85231
3 84.802942  60.50438 109.10151
```

These intervals are much wider as they have to include the variability in individual students, and we have seen that this is large. In fact, there are some out-of-range results in these predictions – `Exam = -15.565` and `Exam = 109.102`!!

Exam vs. Test marks. . .

Conclusions about predictive ability

This is telling us that we have reached the limits of this linear model approximation. It does a fairly good job of explaining the trend but falls down as we can only account for 59% of the overall variation from `Test` and /or the straight line relationship may be a little too naive.

At the top end of exam/test there are fewer students, and there may be a different dynamic at this end of the data due to the constraint that exam mark must be between 0 and 100.

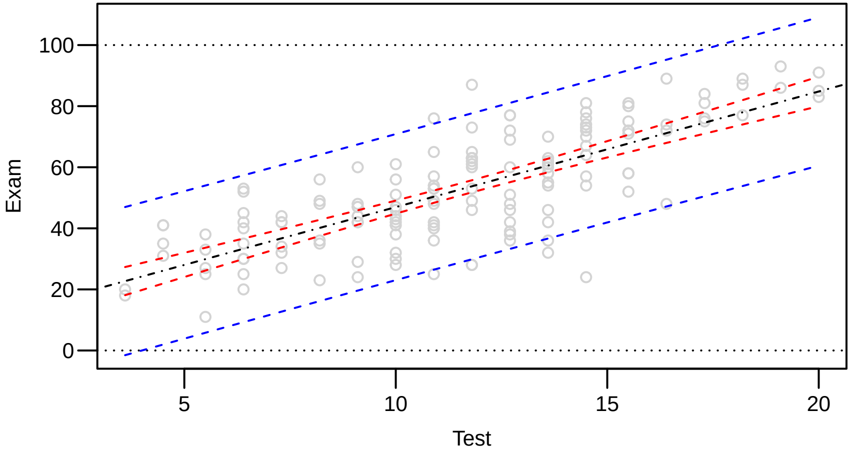
We could not use this to give a student an aegrotat mark based only on their test mark alone.

We would, ideally, like to explain more of the variability in `Exam` by using additional variables of interest – this is known as multiple (as opposed to simple) linear regression – which has the same underlying assumptions – so stay tuned.

Exam vs. Test marks. . .

Conclusions about predictive ability. . .

Here is what the confidence (red) and prediction intervals (blue) look like for these data.



Section 2.7
A recipe for subsequent analyses

A recipe for subsequent analyses

Here is a recipe (algorithm[12]) to use for problems like this where we are interested in a numerical **response** variable y (e.g.: `Exam`) and its relationship with a possible **explanatory** variable x (e.g.:`Test`):

[12] In mathematics and computer science, an algorithm is a self-contained step-by-step set of operations to be performed. Named after the the mathematician, Mohammed ibn-Musa al-Khwarizmi, who was part of the royal court in Baghdad and who lived from about 780 to 850 AD.

A recipe for subsequent analyses...

- Plot the data and see what sort of relation (if any) it suggests (there may also be a statement of research intent to guide you). Propose an appropriate working model. In the above example we decided that $y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \;\; \varepsilon_i \overset{iid}{\sim} N(0, \sigma^2)$ (where $\beta_1 > 0$.)
- Fit the working model using `lm`.
- Check the assumptions you are using and validate them. Independence OK? (how were the data collected?), EOV Okay? – `plot(examtest.fit, which = 1)`, Normality Okay? – `normcheck`. If these are okay then,
- Remove any non-significant explanatory variables where appropriate (more about this later). If so, check new working model.
- Make sure that individual points are not having undue influence and, perhaps, eliminate/correct them – `cooks20x`. If these are okay then,
- Make conclusions/predictions, discuss limitations, and answer relevant research questions.

Note in the above do not go to the next step until you are satisfied with the current step.

Section 2.8
Relevant R-code

Most of the R-code you need for this chapter

Fitting a linear (straight line) model to these data.

```
> examtest.fit=lm(Exam~Test, data=Stats20x.df)
```

Note: Independence is evaluated by investigating how the data was collected. Observations are assumed to be acting independently of each other. Much thought must go into making sure this assumption holds.

Checking for EOV

```
> plot(examtest.fit.which=1)
```

Checking approximate normality

```
> normcheck(examtest.fit)
```

Checking for points of undue influence:

```
> cooks20x(examtest.fit)
```

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

Estimated values from the fitted model:

```
> summary(examtest.fit)
```

Confidence intervals for the model parameters (intercept and slope)

```
> confint(examtest.fit)
```

Creating a data frame of new values (for, say, Test =0, 10 or 20):

```
> preds.df=data.frame(Test=c(0,10,20))
```

Confidence intervals for expected values and prediction intervals for new observations:

```
> # confidence interval for expected value:
> predict(examtest.fit, preds.df, interval="confidence")
> # prediction interval for new observation:
> predict(examtest.fit, preds.df, interval="prediction")
```