Chapter 1: Getting started with linear regression

STATS 201/8

University of Auckland

Learning outcomes

In this chapter you will learn about:

- Getting started using R with an example that analyses data from a previous class of STATS 20x students
- Getting data into R – creating a dataframe
- Working with dataframes in R
- How to fit straight lines to your data – the simple linear model
- How to predict using the fitted model
- How good is the fitted model for prediction? – R^2
- Some technical asides and relevant R-code for this section.

Section 1.1
A motivating example - previous 20x students

Example – Former STATS 20x Students

Over the next few weeks we will repeatedly make use of a dataset we collected from previous students who sat STATS 20x.

In this chapter we will examine a simple straight line model to see if a student's test mark can explain and predict their final exam mark.

Data collected from former students in 20x

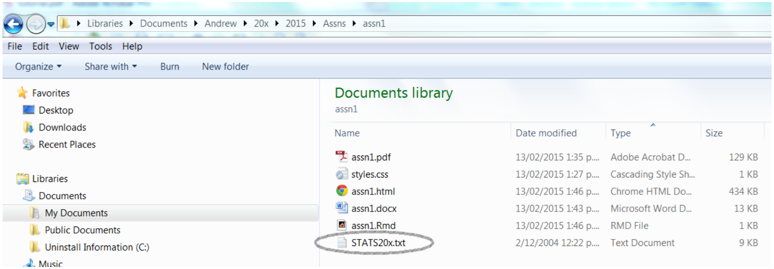
`Exam`	the student's final exam mark (out of 100)
`Degree`	the degree the student is enrolled for: Arts = BA, Commerce = BCom Science = BSc, Other = Other
`Gender`	the student's gender: Female or Male
`Attend`	whether the student attended lectures regularly: No or Yes
`Assign`	the student's Assignment mark (out of 20)
`Test`	the student's midterm mark (out of 20)
`Stage1`	the student's grade for Stage 1 Statistics: A, B or C
`Years.Since`	the number of years since the student passed Stage 1 Statistics
`Repeat`	whether the student is repeating the course: No or Yes.

Note: In this Chapter we'll be using only `Test` to predict `Exam`. What other variables in the dataset might be useful for predicting `Exam`?

Section 1.2
Getting data into R – creating a dataframe

How do we import this data into R?

The original data are stored in a text file called `STATS20x.txt` in a suitable folder.


▸ Libraries ▸ Documents ▸ Andrew ▸ 20x ▸ 2015 ▸ Assns ▸ assn1
File Edit View Tools Help
Organize ▾ Share with ▾ Burn New folder
Favorites
Desktop
Downloads
Recent Places
Libraries
Documents
My Documents
Public Documents
Uninstall Information (C:)
Music

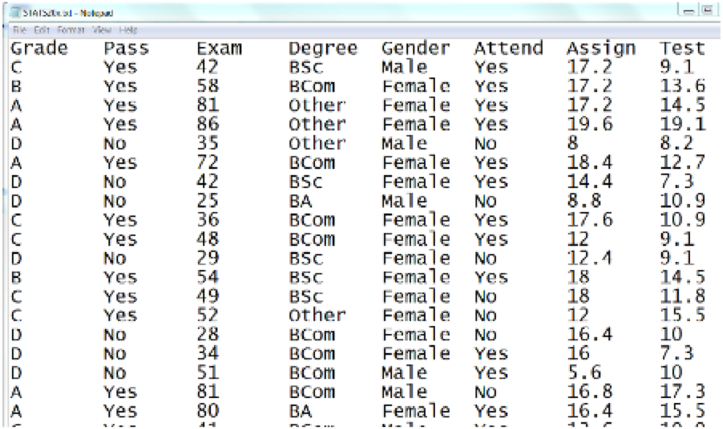
Documents library
assn1

Name	Date modified	Type	Size
assn1.pdf	13/02/2015 1:35 p....	Adobe Acrobat D...	129 KB
styles.css	13/02/2015 1:27 p....	Cascading Style Sh...	1 KB
assn1.html	13/02/2015 1:46 p....	Chrome HTML Do...	434 KB
assn1.docx	13/02/2015 1:43 p....	Microsoft Word D...	13 KB
assn1.Rmd	13/02/2015 1:46 p....	RMD File	1 KB
STATS20x.txt	2/12/2004 12:22 p....	Text Document	9 KB

Text files often have `.txt` appended to their name. The letters after the full stop in the file name are sometimes called a file-name *extension*, and are often used by the operating system (Windows, Mac OS X, Linux) to decide which application/programme should be used to open the file.

What do the data look like?

As data analysts we love our data to be in ‘rectangular’ (i.e., matrix or array) form. Our raw data looks like this in the Notepad text editor.



Grade	Pass	Exam	Degree	Gender	Attend	Assign	Test
C	Yes	42	BSc	Male	Yes	17.2	9.1
B	Yes	58	BCom	Female	Yes	17.2	13.6
A	Yes	81	Other	Female	Yes	17.2	14.5
A	Yes	86	Other	Female	Yes	19.6	19.1
D	No	35	Other	Male	No	8	8.2
A	Yes	72	BCom	Female	Yes	18.4	12.7
D	No	42	BSc	Female	Yes	14.4	7.3
D	No	25	BA	Male	No	8.8	10.9
C	Yes	36	BCom	Female	Yes	17.6	10.9
C	Yes	48	BCom	Female	Yes	12	9.1
D	No	29	BSc	Female	No	12.4	9.1
B	Yes	54	BSc	Female	Yes	18	14.5
C	Yes	49	BSc	Female	No	18	11.8
C	Yes	52	Other	Female	No	12	15.5
D	No	28	BCom	Female	No	16.4	10
D	No	34	BCom	Female	Yes	16	7.3
D	No	51	BCom	Male	Yes	5.6	10
A	Yes	81	BCom	Male	No	16.8	17.3
A	Yes	80	BA	Female	Yes	16.4	15.5

Note that the first row of this `.txt` file contains the variable (i.e. column) names. Each row represents a different student. The columns within each row are separated by a `TAB` character – i.e. the file is *`TAB` delimited*.

How do we import this data-file?

```
> ## Importing data into R
> Stats20x.df = read.table("Data/STATS20x.txt", header=TRUE)
```

Here we are importing the `STATS20x.txt` data from a folder called `Data` that resides within our "working directory". We are also telling `R` that the 1st line of data contains the variable names by using the argument `header = TRUE`.[1]

If the data file resides in the working directory then it is enough to use

```
> Stats20x.df = read.table("STATS20x.txt", header=TRUE)
```

You can set the working directory using the `Session` menu of RStudio.

Note: In `R` code, we preface comment lines with the `#` symbol.

[1] `T` can be used as an abbreviation for `TRUE`.

Section 1.3
Working with dataframes in R

Where are the data?

It is there all right![2] The data exist in the R session as a dataframe object with the name `Stats20x.df`.

You can use whatever name you want (within reason) for objects you create in your R session, so always try to use meaningful names that will help you remember what the object is. That is why we used the `.df` suffix for our dataframe.

To see the dataframe you just need to ask. For example:

```
> ## 1st 5 rows and 7 columns of data set Stats20x.df
> Stats20x.df[1:5,1:7]

  Grade Pass Exam Degree Gender Attend Assign
1     C  Yes   42    BSc   Male    Yes   17.2
2     B  Yes   58   BCom Female    Yes   17.2
3     A  Yes   81  Other Female    Yes   17.2
4     A  Yes   86  Other Female    Yes   19.6
5     D   No   35  Other   Male     No    8.0
```

[2] Provided that you have **Run** the code chunk – knitting does not save the objects created.

How do we obtain the dimensions of a dataframe?

What are the dimensions of the Stats20x.df dataframe?

```
> dim(Stats20x.df)

[1] 146  15
```

It consists of 146 rows (students) and 15 columns. Each column represents a recorded measurement, characteristic, or feature of the students – that is, a variable.

Note: R is case sensitive. For example, the dataframe stats20x.df does not exist, and so asking for its dimensions will give an error:

```
> dim(stats20x.df)

Error in eval(expr, envir, enclos):  object 'stats20x.df' not found
```

Convention: Many programming languages refer to a rectangular table of data as a *dataframe*. In this course we will use the R function data.frame to create new dataframes.

Summarising STATS 20x exam marks

You can ask for one variable at a time using the $ sign after the dataframe name. For example, we type Stats20x.df$Exam to obtain the Exam variable.

Here we just look at the first 10 observations for brevity.

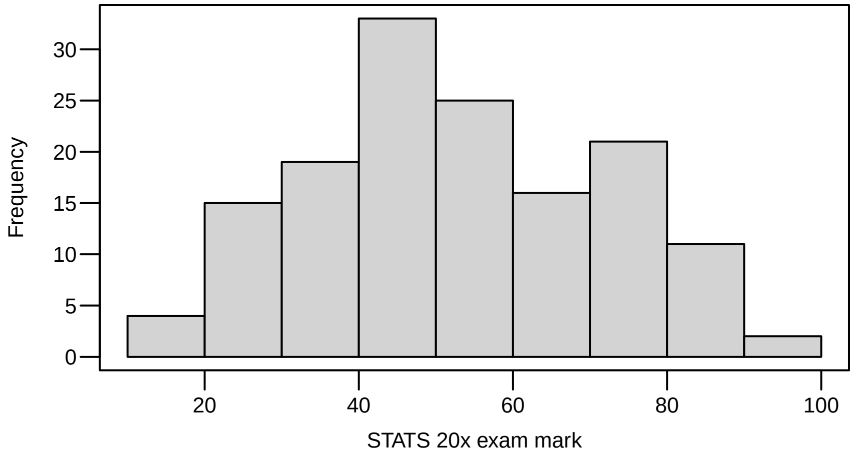
```
> Stats20x.df$Exam[1:10]

 [1] 42 58 81 86 35 72 42 25 36 48
```

If we want to look at the distribution of Exam, the visualisation tool of choice will be a histogram.

Summarising STATS 20x exam marks...

A histogram

```
> #Using xlab="STATS 20x exam score" to label the x-axis.
> hist(Stats20x.df$Exam, xlab="STATS 20x exam score")
```



Summarising STATS 20x exam marks. . .

Numerical summaries

Let us obtain some numerical summaries.

```
> summary(Stats20x.df$Exam)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  11.00   40.00   51.50   52.88   68.50   93.00 
```

- So the lowest mark was 11 (`Min.` $\equiv$ minimum),
- the highest 93 (`Max.` $\equiv$ maximum),
- $\frac{1}{4}$ got 40 or less (`1st Qu.` $\equiv$ lower quartile),
- $\frac{1}{2}$ got less/more than 51.5 (`Med.` $\equiv$ median),
- $\frac{1}{4}$ got more than 68.5 (`3rd Qu.` $\equiv$ upper quartile),
- and the average (`Mean`) mark was about 53 (52.88) marks.

Section 1.4
How to fit straight lines to your data - a.k.a. the simple linear model

Example – Former STATS 20x students' exam marks

From the above summary we have some understanding about STATS 20x students' final exam marks. This is mildly interesting, but perhaps we could ask more interesting questions than this?

Here is one: Does my mid-term test mark relate to my final exam mark?

To be honest, I am pretty sure we know the answer to this question but let us answer this question, based on our data (not based on a strong opinion) and see if this suspected relationship is real or not.

The specific research question addressed in this chapter is **can we use mid-term test score to explain final exam score?**

Exam vs. Test marks

The particular variables of interest

`Exam`	the student's exam mark (out of 100)
`Test`	the student's Test mark (out of 20)

Note: that our question really is about the 'typical' or average relationship. Some students may do well in the term test but not in the exam and vice-versa. We are really interested in the 'expected' effect.

As our variables `Test` and `Exam` are both numerical we draw a scatter-plot to see what relationship is suggested (if any). A scatter-plot (sometimes called a *xy*-plot) has two axes, a horizontal *x*-axis and a vertical *y*-axis.

As scientists we are firstly interested in whether x is related to y.

If so, is x "associated with" y? Or, is x correlated with y.

Exam vs. Test marks. . .

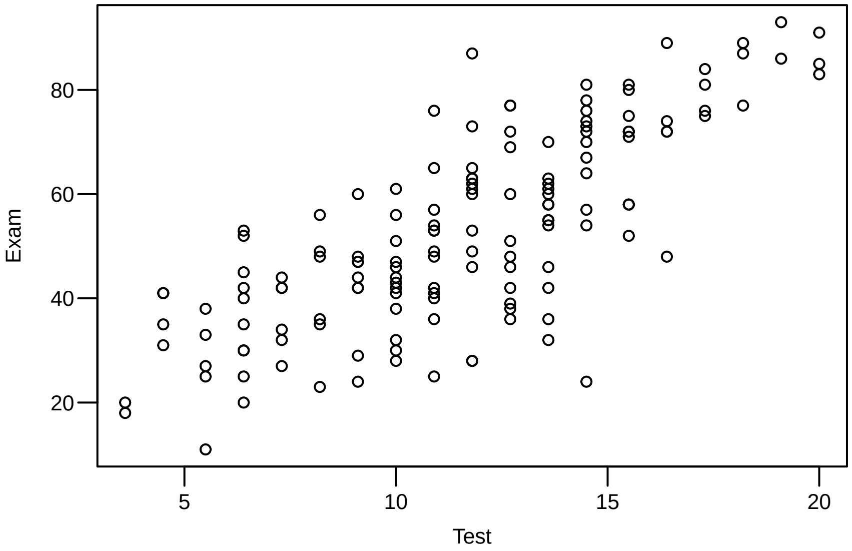
Let us look at these data

Test is the explanatory (or independent) variable and goes on the *x*-axis.
Exam is the response (or dependent) variable and goes on the *y*-axis.

```
> ## Invoke the s20x library
> library(s20x)
> ## Importing data into R
> Stats20x.df = read.table("Data/STATS20x.txt", header=T)
> ## Plot the data
> plot(Exam ~ Test, data = Stats20x.df)
```

Exam vs. Test marks. . .

Let us look at these data. . .



Exam vs. Test marks...

Let us look at these data...

Note that the `plot(Exam ~ Test, data = Stats20x.df)` statement asks `R` to produce a plot showing how `Test` (x-axis variable) is related to `Exam` (y-axis variable).

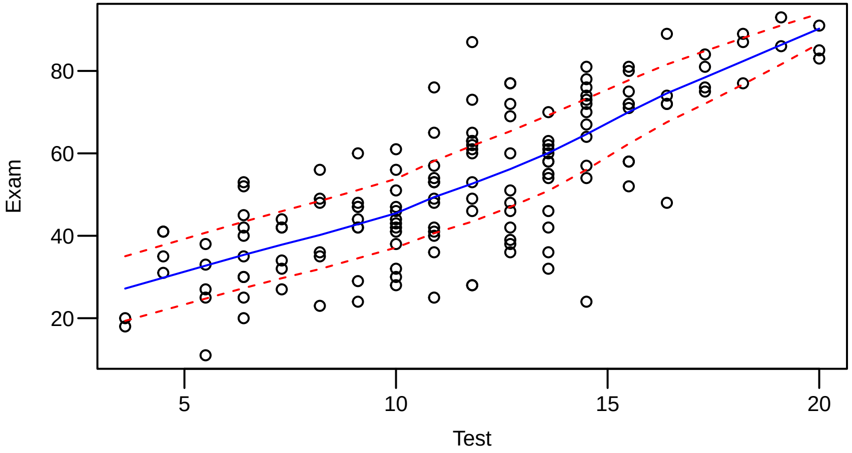
Looking at this plot, it is clear that there is some relationship, but there is also a lot of variability in exam score amongst students with the same test score, especially in the middle of the data.

As we are interested in the 'typical' student we would like to visually see what the underlying trend is, and how much the exam scores vary (scatter) about that trend. It is helpful to run a smooth trend line through the scatter-plot.

Exam vs. Test marks. . .

Let us look at these data. . .

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



Exam vs. Test marks. . .

Let us look at these data. . .

This is our first application of a 'bespoke'[3] function, `trendscatter`, made just for you. It is one of the many functions to be found when you load the `s20x` package. The `trendscatter` function computes a lowess[4] smoother to the data (the blue line) along with estimates of the variation about that line (the red lines).

The underlying trend here looks like a straight line. The variability looks roughly constant – except, perhaps, for those students who got high test marks.

Note: When we talk about "variability" it is in the vertical direction. That is, we are talking about the variability in exam score for a given fixed value of test score.

[3](Of a computer program) written or adapted for a specific user or purpose: *"completely bespoke software systems" – Oxford Dictionary.*

[4]*lowess* stands for locally weighted scatterplot smoothing.

Section 1.5
How to predict using the fitted model?

Exam vs. Test marks. . .

Fitting a simple linear model

Pro Tip: If it looks like a straight line then fit a straight line.

A straight line is a simple (but not the simplest) form of a linear model.

```
> 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
```

Note that `examtest.fit` now resides in our `R` session as an `lm` *object*. An object, in the `R` language, is a structure than can contain data, variables, functions or other objects.

Exam vs. Test marks. . .

The simple linear model

Let us see what this means.

The summary output has a number of different components. Let us start with the `Estimate` column.

You will recall from high school (or should!) that the equation for a straight line,

$$y = a + bx,$$

is described using two numbers (parameters), where:

- a is called the *intercept* since it is the value of y when the line intercepts the y-axis. Note that the y-axis corresponds to $x = 0$.
- b is called the *slope*. It tells us the rate of increase (or decrease) in y, for every additional unit increase in the value x.

Exam vs. Test marks. . .

The simple linear model

In the linear modelling context, it is the expected value of variable y, $\mathrm{E}[Y|x]$, that is assumed to follow a linear relationship with x. We read $\mathrm{E}[Y|x]$ as "the expected value of y given x." This is called a *conditional expectation*. It means that the expected value (or mean) of y will, or may, change depending on the value of x. That is, we assume

$$\mathrm{E}[Y|x] = a + bx$$

The first row of the above `summary(examtest.fit)` table is labelled `Intercept`, and in the `Estimate` column is the estimated value of a (i.e., the estimated value of $y =$ `Exam` when $x =$ `Test = 0`). The second row gives the estimated value of b, which is the increase in expected value of $y =$ `Exam` when $x =$ `Test` increases by one mark.

Exam vs. Test marks...

The simple linear model, general notation

In the previous slides we used a and b to denote the intercept and slope parameters. This is just a naming choice, and we could have used

$$\mathrm{E}[Y|x] = \theta + \gamma x$$

or

$$\mathrm{E}[Y|x] = c + dx,$$

or

$$\mathrm{E}[Y|x] = \beta_0 + \beta_1 x \ .$$

In this class we shall use the β_0, β_1, choice of parameter names since it is easiest to generalize to more complex models having many parameters.

Exam vs. Test marks...

The simple linear model

In the linear modelling context, it is the expected or typical value of variable y that follows a linear model. So, for the simple linear model:

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

We can say that each y can be broken into what we expect to see plus a component that is random and, therefore, cannot be explained.[5]

We can write this as

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

where ε (epsilon) is random with expected value 0. This expresses how an observation varies around its expected or typical value. We will discuss these ideas in greater detail in the next chapter.

[5]This also applies to the more complex linear models we see later.

Exam vs. Test marks. . .

Let us look at this model

Let us extract the coefficients component from the fitted model object:

```
> coef(examtest.fit)

(Intercept)        Test
   9.084463    3.785924
```

You can see this corresponds to the `Estimate` column of the regression summary table.

We are saying that the 'best' fit of the data tells us that the predicted exam score for a student with test score `Test` is given by the following equation (to two decimal places):

$$\widehat{\texttt{Exam}} = 9.08 + 3.79 \times \texttt{Test}^{6}.$$

[6] We use the notation $\widehat{\texttt{Exam}}$ as this is an estimated value.

Exam vs. Test marks...

Let us look at this model...

So, if you have received 0 in the `Test`, then we estimate that you should be able (on average) to get 9.08 marks for just turning up to the exam and signing your name.

Thereafter, for every extra mark you get in the the test (out of 20) your final exam score will increase (on average) by about 3.79 marks.

Let us make some predictions for typical students who got test mark of 0, 10 or 20:

- `Test =  0` then $\widehat{\texttt{Exam}} = 9.08 + 3.79 \times 0 = 9.08\%$ in the exam,
- `Test = 10` then $\widehat{\texttt{Exam}} = 9.08 + 3.79 \times 10 = 46.94\%$ in the exam,
- `Test = 20` then $\widehat{\texttt{Exam}} = 9.08 + 3.79 \times 20 = 84.80\%$ in the exam.

Exam vs. Test marks. . .

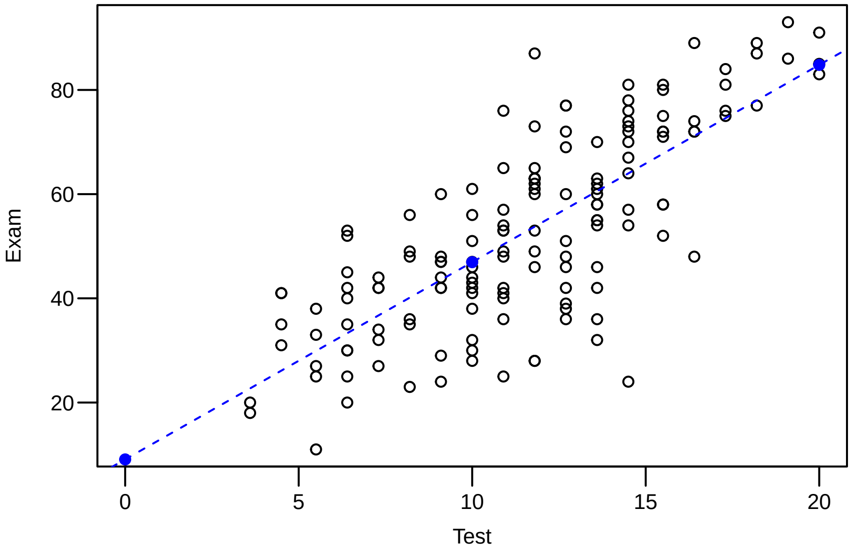
Let us look at this model. . .

Let us see how this model works out overall. Here we are plotting the 'best' estimated straight line that we obtained from fitting our model.

```
> plot(Exam ~ Test, data = Stats20x.df, xlim = c(0, 20))
> ## Add the lm estimated line to this plot where a=intercept, b=slope
> abline(examtest.fit, lty = 2, col = "blue")
> ## Include estimated points for students who get test=0,10,20
> points(c(0,10,20), predict(examtest.fit, newdata = data.frame(Test=c(0,10,20))),
+     col = "blue", pch = 19)
```

Exam vs. Test marks. . .

Let us look at this model. . .



Section 1.6
How good is the fitted model for prediction? – R^2

Exam vs. Test marks. . .

The null model

This model is not too bad. It appears to do a reasonable job of describing the data and appears to make sense – but compared to what? How useful is the variable `Test` in explaining and predicting `Exam`?

We can fit a model that is even simpler than the straight line model. This can be called the null[7] model and simply estimates the typical exam mark without an explanatory variable x. That is , $\mathrm{E}[Y|x] = \mathrm{E}[Y]$. Here is our null model for $y = Exam$:

$$\begin{aligned} y &= \mathrm{E}[Y|x] + \varepsilon \\ &= \mathrm{E}[Y] + \varepsilon \\ &= \beta_0 + \varepsilon \\ &\equiv \mu + \varepsilon \end{aligned}$$

That is, $x = Test$, say, is not related to exam mark.

[7]The null model is the model assumed by a one sample t-test, as seen in Chapter 3.

Exam vs. Test marks...

The null model...

Let us fit this *null* model:

```
> ## Null model
> examnull.fit=lm(Exam ~ 1, data = Stats20x.df)
> summary(examnull.fit)
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   52.877      1.546   34.21   <2e-16 ***
---
Residual standard error: 18.68 on 145 degrees of freedom
```

In comparison, the simple linear model `examtest.fit` gives us different predictions for different test marks. Naturally we would suspect this to be a better model for predicting *Exam* as we are pretty sure that *Test* explains *Exam* quite well.

Exam vs. Test marks...

The null model...

Our null model was $Exam = \beta_0 + \varepsilon$ where $\varepsilon \overset{iid}{\sim} N(0, \sigma^2_{Null})$[8] with the best estimate of $\mu = \beta_0$ being the sample mean, $\bar{y} = 52.88$.

Note that the estimated standard deviation (σ) of the random ε component was $\hat{\sigma}_{Null} = s_{Null} = 18.68$.[9]

[8] If we were were doing a one sample t-test we would use parameter name μ instead of the more general β_0.

[9] Here we use the hat notation to say $\hat{\sigma}_{Null}$ estimates the unknown parameter σ_{Null}.

Exam vs. Test marks. . .

Null vs. simple linear model

Compare the null model to the simple linear model that used `Test` as an explanatory variable.

```
> ## Exam vs. Test model
> 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
```

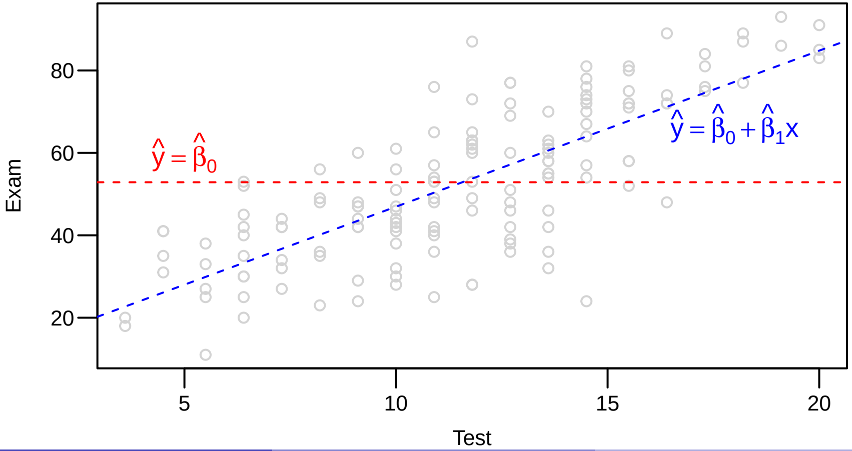
Note that the estimated standard deviation (s_{Test}) of the random standard deviation of the random (ε) component (σ_{Test}) has been reduced to $s_{Test} = 12.05$.

This is also called the residual standard error.

Exam vs. Test marks. . .

How good is this simple linear model?

If $y =$ `Exam` and $x =$ `Test`, then here are our fitted null (the red line) and linear (the blue line) models. We clearly see that the blue line does a far better job of describing this relationship than the red line.



Exam vs. Test marks. . .

Multiple R-squared

The overall variation in the null model is given by the sum of the squared residuals, denoted by SS_{Null}. In Chapter 2 we will see that $SS_{Null} = 50586$. SS_{Null} is more commonly known as the **total sum of squares**, SS_{Tot}.

The sum of squares of the residuals from the model using the variable `Test`, denoted here by SS_{Test}, is 20901.

The reduction in overall variation from the `Null` model to the `Test` model can be expressed as a proportion of SS_{Null}:

$$\frac{SS_{Null} - SS_{Test}}{SS_{Null}} = 1 - \frac{SS_{Test}}{SS_{Null}} = 1 - \frac{20901}{50586} \approx 0.59.$$

This statistic is usually called multiple **R-squared** (R^2) or the proportion of variation explained.

In the above output it is: `Multiple R-squared:  0.5868`

R-squared interpretation

We can say that $(100 \times R^2) = 59\%$ of the (total) variation in the exam mark is explained by using a straight line relationship (i.e., simple linear model) with test score.

So by using just one explanatory variable, $x = Test$, we can explain 59% of the variation that we observe in $y = Exam$. That is a pretty good return on one 'piece of information', the `Test` score.

You may have seen (from STATS 210 perhaps) that the estimates of σ_{Null} and σ_{Test} are obtained from the residual sums of squares:
$s_{Null} = \sqrt{SS_{Null}/(n-1)}$ and $s_{Test} = \sqrt{SS_{Test}/(n-2)}$.

In the above calculation, n is the number of observations ($n = 146$ in this example), and $n - 1$ is the degrees of freedom of the null model and $n - 2$ is the degrees of freedom of the straight line model.

Word of caution

In practice we need to first check the assumptions of our model before we can safely use it for inference such as calculating confidence intervals, making predictions, etc.

This is the focus of the next Chapter.

Section 1.7
Some technical asides

Aside: Getting help on functions in R

In this first chapter we have already used several `R` functions. It is easy to forget the details (e.g., the arguments that the function needs), in which case you can get help about the function by simply preceding its name with a question mark.

For example, try

```
> ?summary
```

or

```
> ?lm
```

In RStudio this opens a sub-window for the help file. It can be useful to look at the examples (at the bottom of the help file) of a function's usage to get an idea of how it is used.

Unfortunately, the `R` help can be rather difficult to understand at first because of all the computer jargon it uses. There is plenty of other online help on most `R` functions that can be found through your favourite search engine.

Aside: Data types in R

There are several data types in R, but the three that we need to know about are *numerical*, *character* and *logical*.

We'll see in Chapter 5 that we can also use character data as an explanatory variable. In that chapter we use the character variable class attendance ("Yes" or "No") to explain exam mark. We regard attendance as a two-level factor variable – stay tuned...

```
> summary(Stats20x.df$Attend)

   Length     Class      Mode
      146 character character
```

```
> summary(as.factor(Stats20x.df$Attend))

 No Yes
 46 100
```

Section 1.8
Relevant R-code

Most of the R-code you need for this chapter

Creating a data frame by importing the text file Data/STATS20x.txt from the subfolder Data:

```
> Stats20x.df = read.table("Data/STATS20x.txt", header=TRUE)
```

Plotting a scatter plot of y (Exam) vs. x (Test):

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

and/or a trend-scatter-plot:

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

Fitting a linear (straight line) model and getting the estimated values:

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

Adding your fitted/predicted blue-dashed line to the scatter-plot created above:

```
> abline(examtest.fit, lty = 2, col = "blue")
```