Chapter 3: Equivalence of the null linear model and the one-sample t-test

STATS 201/8

University of Auckland

Learning outcomes

In this chapter you will learn about:

- The equivalence of fitting the null model and doing a one-sample t-test.
- The paired t-test
- Relevant R-code.

Section 3.1
Revisiting the null model

Null vs simple linear regression fits

We have already encountered an example of the null model in Chapter 1.

We saw that we could explain approximately 59% of the variation of *Exam* by fitting a straight line model using *Test*. We calculated this by comparing the sums-of-squares of the residuals for the simple linear model to the sums-of-squares of the residuals of the null model and noticed it had decreased by 59% (ie., $R^2 = 0.59$).

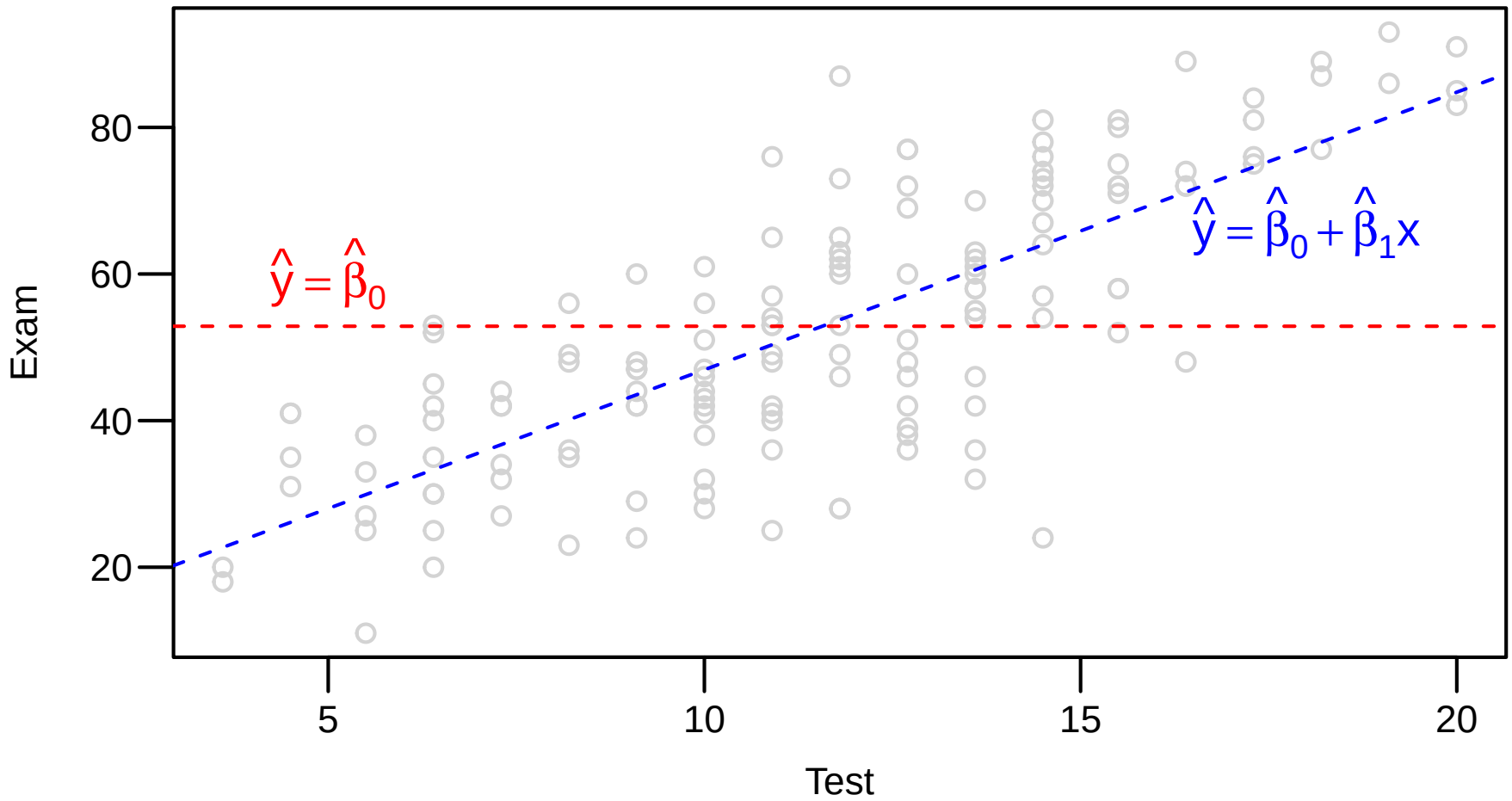
In this chapter we will examine the null model in greater detail and see that it is equivalent to applying the t-distribution for obtaining confidence intervals, and to the one-sample t-test of the null hypothesis that the population mean is zero.

It is also the model we need to use when we have paired comparisons – two repeated measures on the same subject.

Null vs simple linear regression fits...

Exam vs. Test marks

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



A note on the model formula

In linear models, the intercept parameter (β_0) is fitted by default. That is why `lm(y~x)` fits not just the effect of x, but also an intercept. In fact, `lm(y~x)` is a shortened version of `lm(y~1+x)`. The latter form makes it explicit that the model being fitted is

$$y = 1 \times \beta_0 + x \times \beta_1 + \varepsilon,$$

where the ε are *iid* $N(0, \sigma^2)$. This is why the null or intercept-only model can be can be fitted using `lm(y~1)` since this specifies

$$y = 1 \times \beta_0 + \varepsilon.$$

Since the null model is simply just specifying the mean (i.e., expected) value of y, it is common practive to relabel β_0 as μ, in which case we have $y = \mu + \varepsilon$.

This can be abbreviated with the model formula $y \overset{iid}{\sim} N(\mu, \sigma^2)$.

Inference about the expected exam mark

What do we mean by "expected exam mark"??? (Hint: It is also called the population mean.)

Remember, the data are assumed to be a random sample from a bigger population.

Every STATS 20x class differs a bit in the difficulty of the test and exam, for the simple reason that there are different questions every semester. The teaching staff also differ each semester.....so, it would be naive to regard these data as a random sample from all STATS 20x students that we have or will ever teach.

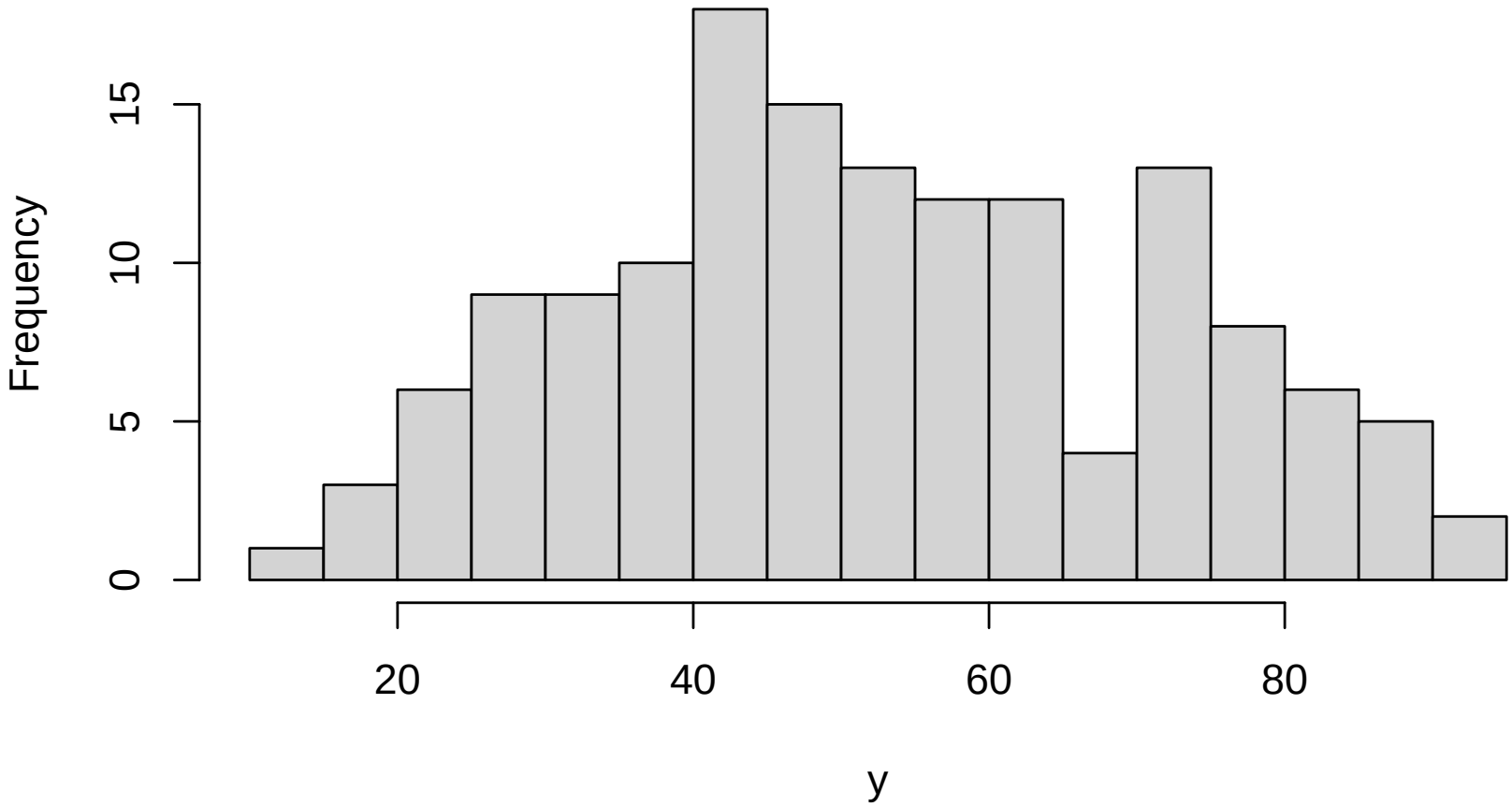
However, it would be reasonable to assume they are from the hypothetical population of all students who could have taken STATS 20x in that particular semester.

Here we wish to see what we can say about the average, or typical, value a student in this hypothetical population will get in the exam in the absence of any other information about them.

Inference about the expected exam mark...

To save some typing we'll let `y` be the vector `Stats20x.df$Exam` of exam scores.

```
> y=Stats20x.df$Exam
> hist(y,breaks=20,main="") #Use main to suppress plot title
```



Inference about the expected exam mark...

Using the null model

The histogram could be better (i.e., more normal in shape), but we'll go ahead with using the null model. We're going to be lazy[1] and not include the assumption checks!

```
> null.fit=lm(y~1)
> coef(summary(null.fit)) #Only give coefficients from summary

            Estimate Std. Error  t value     Pr(>|t|)
(Intercept) 52.87671   1.545802 34.20666 2.632011e-71

> confint(null.fit)

               2.5 %   97.5 %
(Intercept) 49.8215 55.93193
```

[1] Actually, it's more like we are taking a shortcut – the assumption checks won't tell us anything more than what we already see in the histogram of the exam marks.

Inference about the expected exam mark...

Using the null model...

Conclusions

- The near zero `Pr(>|t|)` p-value totally rejects the null hypothesis that $H_0 : \mu \equiv \beta_0 = 0$.
- The 95% confidence interval for μ is 49.82 to 55.93.

The confidence interval is useful... but the p-value for $H_0 : \mu = 0$ is absolutely useless since we would never be interested in asking whether $\mu = 0$.

Inference about the expected exam mark...

A more meaningful null hypothesis

It might be more interesting to test a hypothesis like $H_0 : \mu = 60$, say, where we suppose that 60 corresponds to the target expected score when lecturers prepare STATS 20x exams.[2]

Note that the above null hypothesis is $H_0 : \mathrm{E}[Y] = 60$ and is equivalent to $H_0 : \mathrm{E}[Y - 60] = 0$.

So, if we use the response variable $y - 60$ instead of y then we can get a P-value for this H_0 using the `lm` function, as shown on the next slide.

[2] FYI, in recent semesters the average test and exam scores have been around 70.

Inference about the expected exam mark...

A more meaningful null hypothesis...

```
> null.fit60=lm(I(y-60)~1)
> coef(summary(null.fit60))

             Estimate Std. Error   t value     Pr(>|t|)
(Intercept) -7.123288   1.545802 -4.608151 8.828149e-06
```

Here we have used the inhibit function `I()` to prevent `lm` from mis-interpreting the model formula.

We see that the sample average exam score is about 4.61 standard errors below the target population exam score. The small p-value shows that this is implausible under the null hypothesis.

Question: In plain English, how would you state our conclusion?

Section 3.2
Revisiting the t-test

Inference about the population mean...

Using the t-test

Recall from STATS 10x that we can use the t-distribution to make inference about μ when $y \overset{iid}{\sim} N(\mu, \sigma^2)$.

First, we will do it the hard way, by hand.

Then, we'll let R do it for us.

Inference about the population mean...

Using the t-test...

It can be shown (in STATS 310) that

$$T = \frac{\overline{y} - \mu}{\frac{s}{\sqrt{n}}} \sim t_{n-1},$$

where $\overline{y}$ and s are the sample mean and sample standard deviation we calculate from our sample.[3]

We can use this result to do hypothesis tests and get confidence intervals about the quantity of interest, μ (the population mean exam mark).[4]

[3] Recall that this is interpretted as being the distribution of T when the experiment is repeated. That is, if other random samples are taken from the population.

[4] Recall that T is t_{n-1}-distributed rather than normal as we have additional variability from using s^2 to estimate σ^2.

Inference about the population mean...

Calculating the t-value

In 10x you were taught how to calculate a 95% CI for μ from your sample of n observations having sample mean $\bar{y}$ and sample standard deviation s:

$$\bar{y} \pm t_{n-1}^{(0.975)} \frac{s}{\sqrt{n}}$$

where $t_{n-1}^{(0.975)}$ is the t-multiplier. For a 95% CI this multiplier is pretty close to 2 provided that $n > 30$. [5]

[5] In this example, $t_{n-1}^{(0.975)} = t_{145}^{(0.975)} = 1.97646 \approx 2$.

Inference about the population mean...

Calculating the t-value...

Let us calculate the t-statistic for the null hypothesis that $\mu = 60$. This is

$$T = \frac{\bar{y} - 60}{\frac{s}{\sqrt{n}}}.$$

```
> n=length(y) #146 students
> tstat=(mean(y)-60)/(sd(y)/sqrt(n))
> tstat

[1] -4.608151
```

How does this t-value compare to the one from `coef(summary(null.fit60))` that we saw a few pages earlier?

Inference about the population mean...

Calculating the confidence interval

Let us now manually compute a 95% CI based on the t-distribution:

```
> ## t-multiplier
> tmult = qt(1-.05/2, df=n-1)
> ## We want the upper 97.5% (or 1-.05/2) bound of the CI
> ## NOTE: mean = sample mean; sd = standard deviation; sqrt = square root
> mean(y) - tmult*sd(y)/sqrt(n)

[1] 49.8215


> ## Upper bound of CI
> mean(y) + tmult*sd(y)/sqrt(n)

[1] 55.93193


> ## Or if we want both the lower and upper bounds of the CI in one statement
> mean(y) + c(-1,1)*tmult*sd(y)/sqrt(n)

[1] 49.82150 55.93193
```

How does this CI compare to the one from `confint(null.fit)`?

Inference about the population mean...

The `t.test` function in R

Of course, R has a convenient function to do the t-test calculations for us.

To test $H_0 : \mu = 60$, we include `mu=60` in the call of `t.test`

```
> t.test(y,mu=60)


One Sample t-test

data:  y
t = -4.6082, df = 145, p-value = 8.828e-06
alternative hypothesis: true mean is not equal to 60
95 percent confidence interval:
 49.82150 55.93193
sample estimates:
mean of x
 52.87671
```

Inference about the population mean...

The null model versus the *t*-test

We've seen that we get the same results from using the null model as we do from using the *t*-test. This is because they are both based on the same statistical theory.

The null model is a special case (in fact, it is the simplest case) of a linear model.[6]

For completeness we will also look at using the bootstrap, which you saw in STATS 10x – recall that the bootstrap samples **with replacement** from the data.

[6] If you have done some maths courses, you might recall that a linear model is one that has constant derivative with respect to its coefficients

Section 3.3
Inference about μ using the bootstrap (Non examinable)

Inference about the population mean...

Bootstrapping

```
> ## Resampling the exam marks, N times with replacement:
> N=10000 # The number of bootstrap resamples we want
> # The new sample means are stored in ybar
> ybar=rep(NA,N) ## A vector of length N to store our resampled means
>
> ## A loop - allows us to do something N (10,000) times
> for (i in 1:N){
+   ## Take the average of this sample (below) from a sample of size n = 146 from y
+   ybar[i]=mean( sample(y,n, replace=T) )
+ }
> mean(ybar)

[1] 52.84468
```

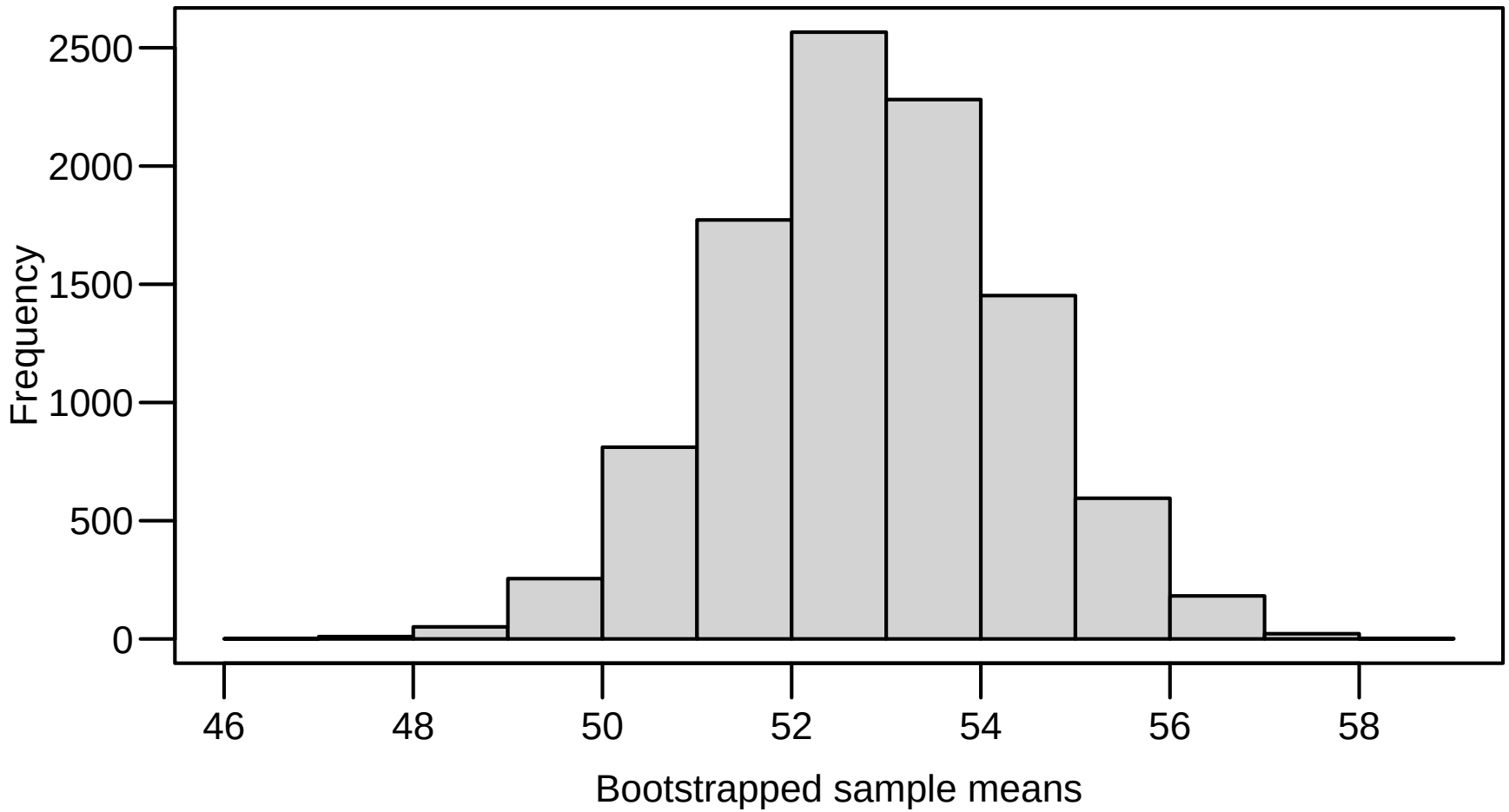
Here is a simpler way of doing the bootstrap, but it requires the `bootstrap` package to be installed:

```
> library(bootstrap)
> ybar = bootstrap(y, 10000, mean)$thetastar
```

Inference about the population mean...

Bootstrapping ...

```
> ## Histogram of these 10,000 bootstrap means
> hist(ybar,xlab="Bootstrapped sample means")
```



Inference about the population mean...

Bootstrapping ...

This looks very 'Normal'.

Note: the mean value of `ybar`, 52.84, is about the same as that of the original sample (52.88), but the values have less scatter. They have lower quartile of 51.8 (in the original sample it was 40), and upper quartile 53.9 (versus 68.5).

A 95% bootstrap confidence interval for the expected exam mark is given by the following code:

```
> ## 2.5% in the lower tail and 2.5% in the upper tail
> quantile(ybar, c(.025, .975))

    2.5%    97.5%
49.86301 55.85616
```

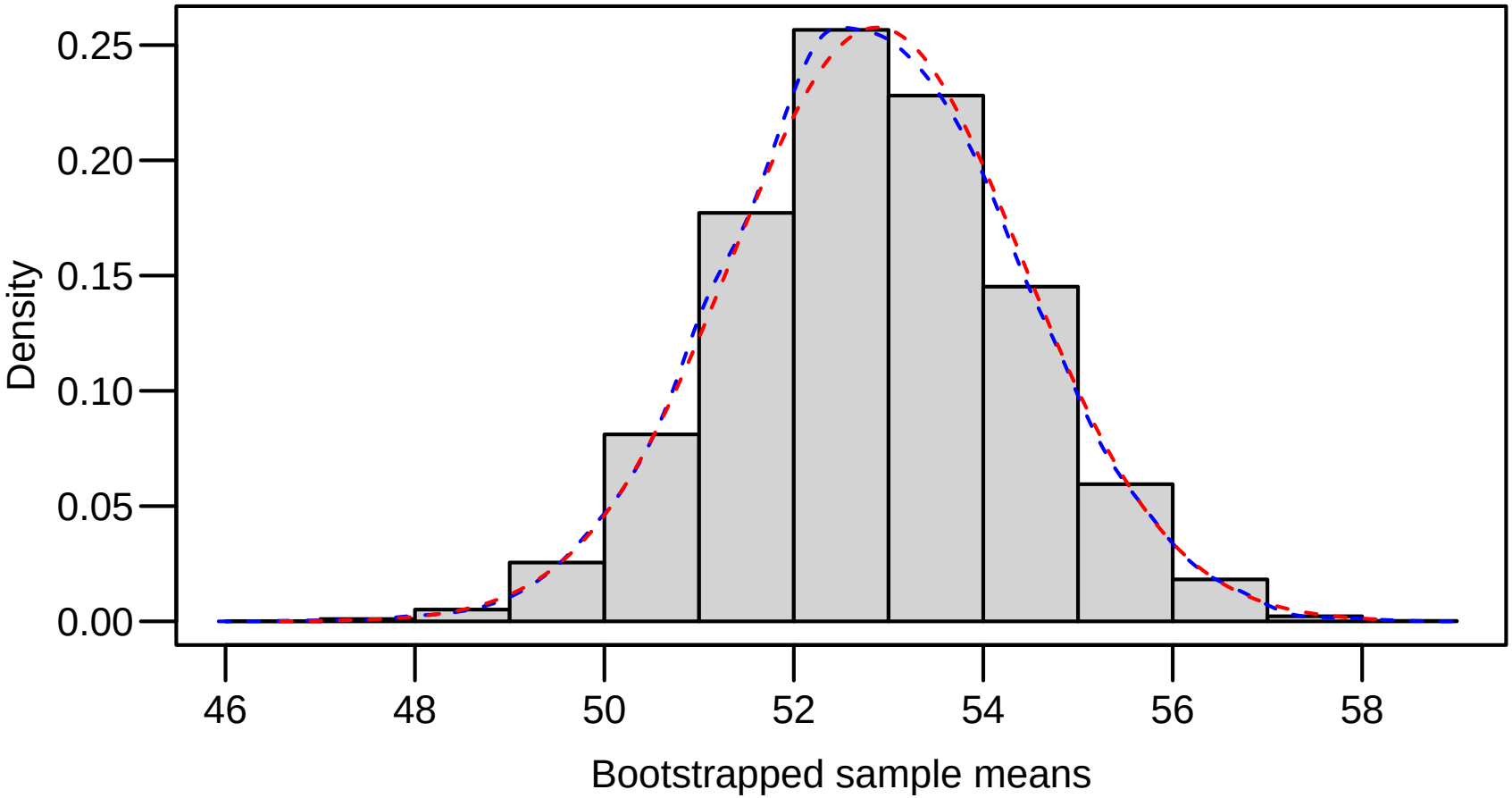
We say: We are 95% confident that the expected exam mark is somewhere between 49.9 to 55.9 marks.[7]

[7] The magic of the bootstrap is that these quantiles of the bootstrapped means give a CI for the population mean – see STATS 730 for proof of this.

Inference about the population mean...

Bootstrapping...

Here is the distribution of sample means we bootstrapped, with a density plot (in blue – which can be though of as a 'fine-grain' histogram), along with the underlying theoretical Normal based-distribution (in red):



Inference about the population mean...

Bootstrapping and the t-distribution

So what is going on here? Provided we have a large enough sample, we know that the distribution of all possible sample means we could have obtained from repeating the experiment is distributed approximately normally[8] and hence, these sample means can be described well with a normal distribution.

This is known as the **Central limit effect** or **theorem**, referred henceforth as the **CLT**. Both the bootstrap and t-distribution follow the CLT.

[8] There are some other necessary conditions but this holds for most populations.

Inference about the population mean...

Bootstrapping and the t-distribution...

Let us compare the bootstrap CI to the t-distribution CI:

```
> ## Bootstrap:
> quantile(ybar, c(.025, .975))

    2.5%    97.5%
49.86301 55.85616

> ## t-distribution:
> mean(y) + c(-1,1)*tmult*sd(y)/sqrt(n)

[1] 49.82150 55.93193
```

Very similar.

Note: From now on we will not do any more bootstrapping as it does not generalize easily to more complex models.

Section 3.4
The paired t-test

Paired comparisons $\equiv$ one-sample t-test

Recall (from STATS 10x) that the paired t-test is just an application of the one-sample t-test applied to differences.

Here, we demonstrate this by comparing *Test* and *Exam* marks.

Comparing *Test* and *Exam* marks

Suppose we want to know if the midterm test marks and exam marks have the same expected value. Note that the test and exam scores are not independent (why?).

The data are paired since we have a test score and exam score from each student[9].

For a meaningful comparison, We will need to make them have the same scale, so we multiply the test mark by 5 so that it is also out of 100. This can be done very easily with the following R code:

```
> Stats20x.df$Test2 = 5 * Stats20x.df$Test
> ## Check that it worked
> Stats20x.df[1:3, c("Exam","Test","Test2")]

  Exam Test Test2
1   42  9.1  45.5
2   58 13.6  68.0
3   81 14.5  72.5
```

[9] Two measurements on the same student constitutes a paired measure and this is an example of a repeated (twice) measures study.

Comparing *Test* and *Exam* marks...

We wish to see how the exam score and scaled test score (out of 100) differ. We might suspect that they have the same expected value μ.

A student who scores high on the exam would be expected to score high on the test and vice-versa. So these two measurements are not independent of each other. However, when we look at their differences (`Diff = Test2 - Exam`) these constitute a single measurement from each student, and moreover these differences could reasonably be assumed to be independent of each other.

In effect, we have eliminated the student effect on test and exam scores by working with the difference between test and exam for each student.

Comparing *Test* and *Exam* marks...

Calculating the difference

Let us name the new variable `Diff` (and check that we have done it correctly):

```
> Stats20x.df$Diff = Stats20x.df$Test2 - Stats20x.df$Exam
> ## Check the first 5 measurements
> Stats20x.df[1:5, c("Test2","Exam","Diff")]

  Test2 Exam Diff
1  45.5   42  3.5
2  68.0   58 10.0
3  72.5   81 -8.5
4  95.5   86  9.5
5  41.0   35  6.0

> ## Looks good!
```

Comparing *Test* and *Exam* marks...

Null hypothesis for the expected difference

If test and exam scores have the same expected value, then their difference must have an expected value of 0. We will denote this $\mu_{diff} = 0$. This is our null hypothesis.

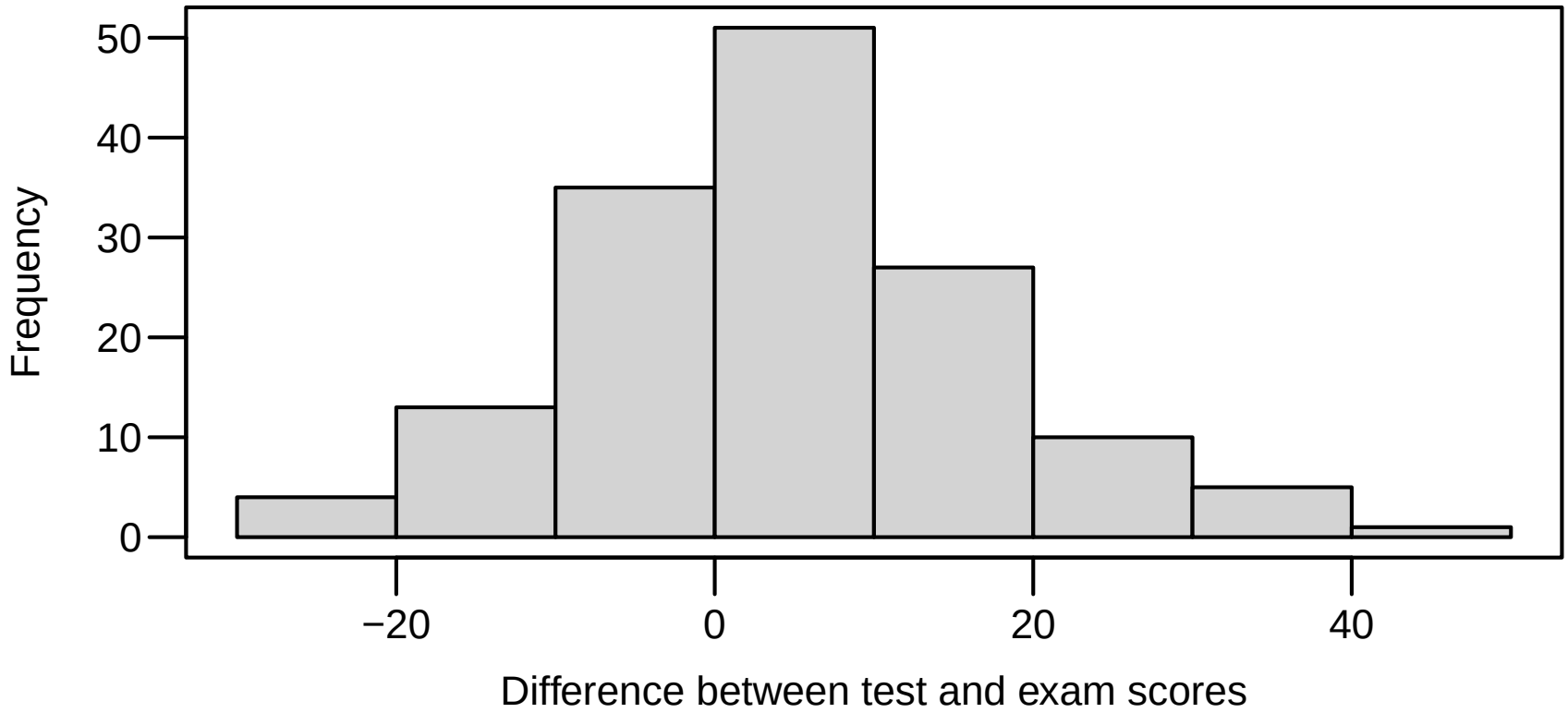
Before we do the test of $H_0 : \mu_{diff} = 0$ we should do some assumption checks.

We have independence sorted, and we are necessarily assuming identical distribution. Let us produce a histogram to check normality.

Comparing *Test* and *Exam* marks...

Inspecting the differences

```
> hist(Stats20x.df$Diff,xlab="Difference between test and exam scores")
```



Looks very normalish. Let's fit a null linear model (i.e., one-sample t-test) to the differences.

Testing for a significant difference

```
> diff.fit = lm(Diff~1, data=Stats20x.df)
> coef(summary(diff.fit))

            Estimate Std. Error  t value     Pr(>|t|)
(Intercept) 4.958904   1.063718 4.661861 7.042125e-06
```

```
> confint(diff.fit)

               2.5 %   97.5 %
(Intercept) 2.856509 7.061299

> t.test(Stats20x.df$Diff)


One Sample t-test

data:  Stats20x.df$Diff
t = 4.6619, df = 145, p-value = 7.042e-06
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 2.856509 7.061299
sample estimates:
mean of x
 4.958904
```

Comparing *Test* and *Exam* marks...

Conclusions

So it appears that, on average, students do worse in the exam than the term test (P-value $\approx 7 \times 10^{-6}$).

We estimate this difference to be about 2.9 to 7.1 marks (out of 100).

The exam was considerably harder than the test – this is something that lecturers normally try to avoid. It's not a good idea for the test to be easier than the exam, since it may lull students into a false sense of security.

Comparing *Test* and *Exam* marks...

Closing remarks

Some history; The `Stats20x.df` data were collected some years ago. Back then, lecturers were able to scale marks as they saw fit, and were also to choose the final grade ranges for awarding the letter grades (A+, A,...etc).

So, lecturers would often deliberately set very challenging tests and exams since they would be able to adjust grades upwards and lower the grade requirement for a letter grade. This would have been the case for the test and exam marks in `Stats20x.df`.

Lecturers no longer have as much flexibility, and so these days test and exam (and assignment) marks must now be more representative of the final grade.

Section 3.5
Relevant R-code

Most of the R-code you need for this chapter

Fitting the model with no explanatory variables (i.e. no x):

```
> exam.fit=lm(Exam~1, data=Stats20x.df)
> confint(exam.fit)
> exam.fit60=lm(I(Exam-60)~1, data=Stats20x.df)
> coef(summary(exam.fit60))
```

Equivalent output can be obtained from the t-test:

```
> t.test(Stats20x.df$Exam,mu=60)
```

For a paired comparison we need to create the difference variable:

```
> Stats20x.df$Diff = Stats20x.df$Test2 - Stats20x.df$Exam # create differences
```

Then either of these will suffice:

```
> # confidence interval for fitted value:
> diff.fit=lm(Diff~1, data = Stats20x.df)
> coef(summary(diff.fit))
> confint(diff.fit)
```

or equivalently

```
> t.test(Stats20x.df$Diff)
```