STATS 330

Handout 10
Introduction to simulation

Department of Statistics, University of Auckland

Simulation studies

When we conduct an analysis, we start with data, then we fit models that makes assumptions about these data in order obtain estimates and confidence intervals for unknown population parameters. Our final model might represent our best guess at what the 'true' model is that generated the data, and its estimates might represent our best guess at the 'true' parameter values. However, we never know for sure what the 'true' model that generated the data is, nor what the 'true' values of these population parameters are.

A simulation study does things the other way around. When we conduct a simulation study, we pretend we know the 'true' model and the 'true' popoulation parameters. We repeatedly simulate data sets that are generated by this process, and fit models to these simulated data.

Simulation studies

Why?

Simulation studies can help us gain an understanding of the models we fit, and draw conclusions from the data we've collected. We'll see a few examples of how simulation studies can be useful over the next few handouts.

In this handout, we'll show how we can use a simulation study to investigate the following important question:

- Can we trust the estimates we get from a linear model if its assumptions are met?

We already know that the answer is "Yes". The standard way of showing this result relies on statistical theory. In this handout, we will instead establish the trustworthiness of our methods empirically, by using simulation.

Simulation studies

Why?

The reason we are able to answer this question with a simulation study is because, in a sense, we are 'playing God'. Since we are generating the data ourselves, we know what the 'truth' is. When we fit models to the data we have generated, we are able to see how well they are able to estimate the true parameter values.

We can almost never do this with any real data set, because we do not know the true process that generated our data, nor the true parameter values of that process.

Simulation in R - some details

Every distribution that R handles has four functions. There is a root name, for example, the root name for the normal distribution is norm. This root is prefixed by one of the letters:

- `p` — for "probability", the cumulative distribution function (c.d.f.);
- `q` — for "quantile", the inverse c.d.f.;
- `d` — for "density", the density function (p.f. or p.d.f.);
- `r` — for "random", a random variable having the specified distribution. Use `set.seed()` for reproducibility.

For the normal distribution, these functions are `pnorm`, `qnorm`, `dnorm`, and `rnorm`. For the binomial distribution, these functions are `pbinom`, `qbinom`, `dbinom`, and `rbinom`—and so forth.

For a discrete distribution (like the binomial), the `d` function calculates the density (p.f.), which in this case is a probability $f(x) = P(X = x)$ and hence is useful in calculating probabilities.

Simulation in R

We can use R to generate, or 'simulate' pretend data that come from a particular distribution with particular parameters.

For example, to simulate three numbers from a normal distribution with a mean of 5.9 and a standard deviation of 2.2 (a variance of $2.2^2 = 4.84$) we use the following code:

```
rnorm(3, 5.9, 2.2)
```

```
## [1] 3.244455 6.510344 8.285771
```

Every time we run this same line of code we get different values:

```
rnorm(3, 5.9, 2.2)
```

```
## [1] 0.7394651 6.8440743 7.0133230
```

```
rnorm(3, 5.9, 2.2)
```

```
## [1] 4.635572 4.697410 4.658206
```

Simulation in R

Here are 10 values from a Poisson distribution with a mean of 7.1:

```
rpois(10, 7.1)
```

```
##  [1]  5  5  6  6  4  3  5  9  7 11
```

And 9 values from a binomial distribution, all with 7 trials and each trial having a 0.5 probability of success:

```
rbinom(9, 7, 0.5)
```

```
## [1] 5 1 3 3 3 4 2 4 2
```

Each number above could be considered a simulated number of heads obtained from seven flips of a fair coin.

Simulation in R

We can also simulate values from the same distribution, but with different parameter values, in a single line of code. For example, the code below simulates three numbers from a normal distribution:

1. One with a mean of 5.1 and a standard deviation of 4.
2. One with a mean of 102.0 and a standard deviation of 10.
3. One with a mean of -23.1 and a standard deviation of 0.1.

```
means <- c(5.1, 102.0, -23.1)
sds <-  c(4, 10, 0.1)
rnorm(3, means, sds)

## [1]   2.511924 110.681809 -23.062436
```

Simulating from a model

Consider the following simple linear regression model:

$$\mu_i = \beta_0 + \beta_1 x_i$$
$$Y_i \sim \text{Normal}(\mu_i, \sigma^2)$$

where we have ten observations for which

- $x_1 = 2, x_2 = 4, \cdots, x_{10} = 20$,
- $\beta_0 = 20$,
- $\beta_1 = 3$, and
- $\sigma = 10$.

Simulating from a model

This means the expected value for each observation is

$$\begin{aligned}\mu_1 &= 20 + 3 \times 2 = 26,\\ \mu_2 &= 20 + 3 \times 4 = 32,\\ &\dots\\ \mu_{10} &= 20 + 3 \times 20 = 80\end{aligned}$$

So the first observation comes from a normal distribution with a mean of 26 and a standard deviation of 10, the second observation comes from a normal distribution with a mean of 32 and a standard deviation of 10, and so on.

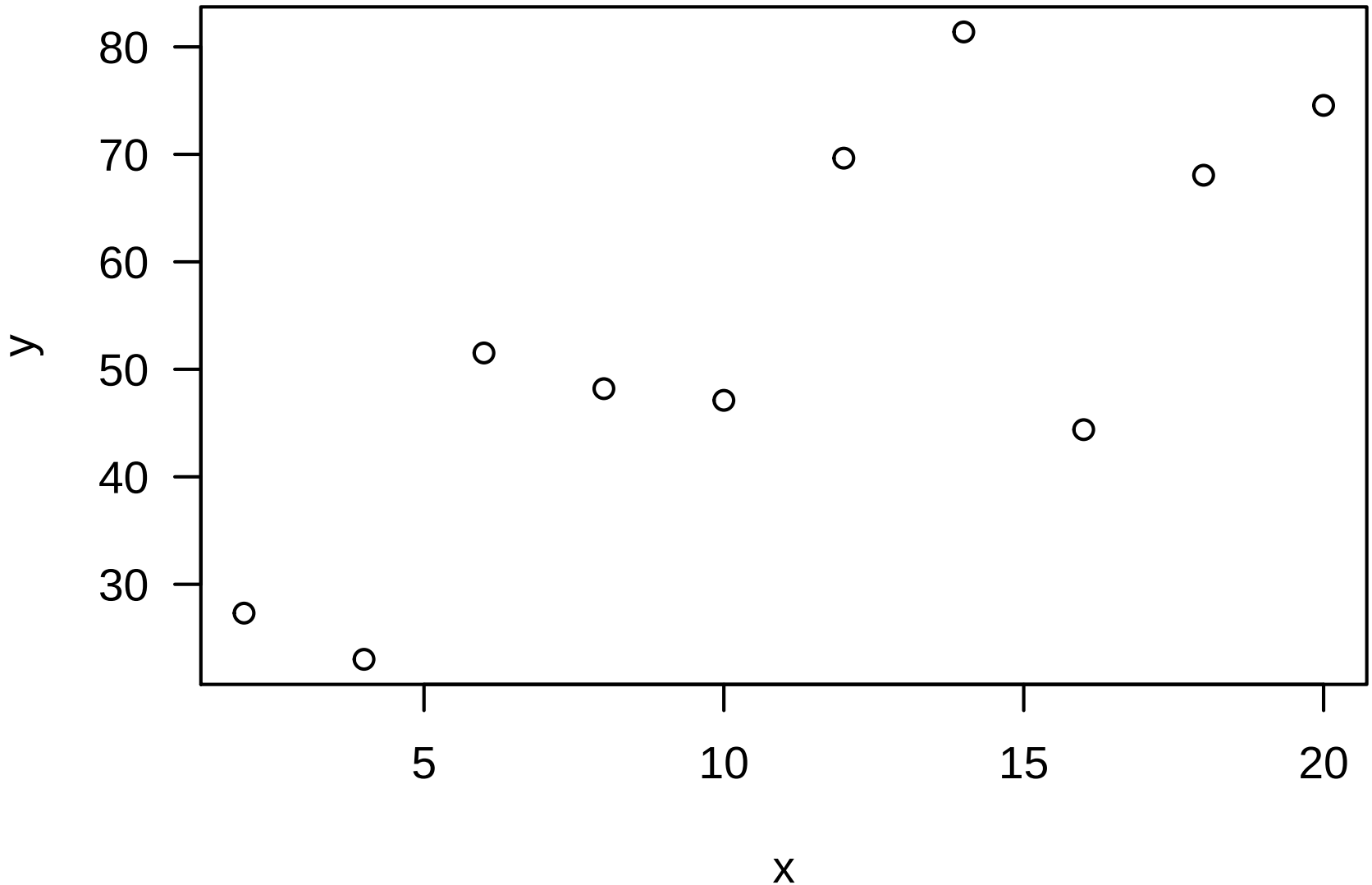
Simulating from a model

We can simulate the response data under this model by constructing a vector with these means, and then using `rnorm()`:

```
b0 <- 20; b1 <- 3; sigma <- 10
x <- seq(2, 20, length.out = 10)
x

##  [1]  2  4  6  8 10 12 14 16 18 20

means <- b0 + b1*x
means

##  [1] 26 32 38 44 50 56 62 68 74 80

y <- rnorm(10, means, sigma)
round(y, 1)

##  [1] 27.3 23.0 51.5 48.2 47.1 69.6 81.4 44.4 68.1 74.6
```

Simulating from a model

The simulated data



Simulating from a model

Fitting the model

We can now fit a model to our simulated data:

```
fit <- lm(y ~ x)
summary(fit)

## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  25.9588     9.0117   2.881  0.02049 *
## x             2.5056     0.7262   3.450  0.00869 **
## ---
## Residual standard error: 13.19 on 8 degrees of freedom
## Multiple R-squared:  0.5981,Adjusted R-squared:  0.5479
## F-statistic: 11.91 on 1 and 8 DF,  p-value: 0.008689
```

We know this model is appropriate because we simulated the data in a way that was consistent with the model's assumptions.

Simulating from a model

Fitting the model: How did we do?

We can compare the estimated coefficients to the true values from which the data were simulated:

	Parameter	Estimate	95% CI
β_0	20	25.96	(5.18, 46.74)
β_1	3	2.51	(0.83, 4.18)
σ	10	13.19	—

We can assess how well our model did:

- Did our estimates get close to the true parameter values?
- Did the 95% CIs capture the true parameter values?

Simulating from a model

Fitting the model: How did we do?

We can compare the estimated coefficients to the true values from which the data were simulated:

	Parameter	Estimate	95% CI
β_0	20	25.96	(5.18, 46.74)
β_1	3	2.51	(0.83, 4.18)
σ	10	13.19	—

We can assess how well our model did:

- Did our estimates get close to the true parameter values?
  - Yes! More or less.
- Did the 95% CIs capture the true parameter values?

Simulating from a model

Fitting the model: How did we do?

We can compare the estimated coefficients to the true values from which the data were simulated:

	Parameter	Estimate	95% CI
β_0	20	25.96	(5.18, 46.74)
β_1	3	2.51	(0.83, 4.18)
σ	10	13.19	—

We can assess how well our model did:

- Did our estimates get close to the true parameter values?
  - Yes! More or less.
- Did the 95% CIs capture the true parameter values?
  - Yes!

Simulating from a model

Fitting the model: How did we do?

We can compare the estimated coefficients to the true values from which the data were simulated:

	Parameter	Estimate	95% CI
β_0	20	25.96	(5.18, 46.74)
β_1	3	2.51	(0.83, 4.18)
σ	10	13.19	—

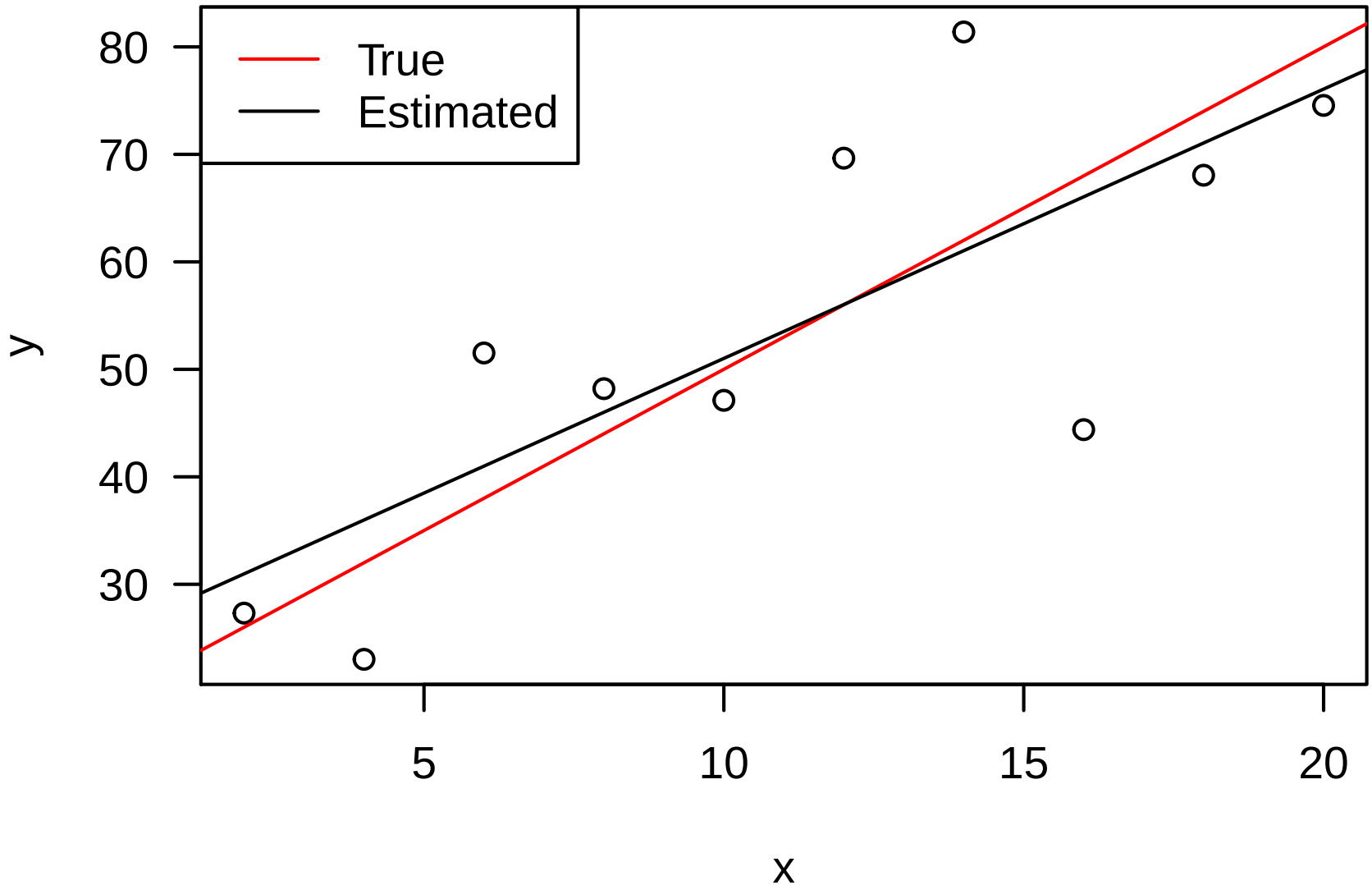
We can assess how well our model did:

- Did our estimates get close to the true parameter values?
  - Yes! More or less.
- Did the 95% CIs capture the true parameter values?
  - Yes!

So it looks like our model did a reasonable job of estimating the underlying parameters.

Simulating from a model

Fitting the model: How did we do?



Simulating from a model

Fitting the model: How did we do?

Fitting our model to these data resulted in estimates that were pretty close to the true values, and 95% confidence intervals that captured them.

Would we always expect to do this well, or did we just get lucky?

We can answer this question by simulating many more data sets, fitting our model to each data set, and saving the results from each. This is what we usually mean by 'conducting a simulation study'.

A simulation study

To simulate the data set and fit the model we used the following:

```
## Setting the model's parameters.
b0 <- 20; b1 <- 3; sigma <- 10
## Setting the fixed x-values.
x <- seq(2, 20, length.out = 10)
## Calculating expected response values.
means <- b0 + b1*x
## Simulating the responses.
y <- rnorm(10, means, sigma)
## Fitting the model.
fit <- lm(y ~ x)
## Extracting the model parameters.
est.b0 <- coef(fit)[1]; est.b1 <- coef(fit)[2]
```

If we wanted to simulate 10 000 data sets and fit 10 000 models we could copy-paste the code above 10 000 times. . . . But that doesn't seem very practical.

A simulation study

A much easier way is to use a 'for loop', which results in the same lines of code being executed many times over and over again:

```
## Setting the model's parameters.
b0 <- 20; b1 <- 3; sigma <- 10
## Setting the fixed x-values.
x <- seq(2, 20, length.out = 10)
## Calculating expected response values.
means <- b0 + b1*x
## Setting the number of data sets and models to simulate.
n.sims <- 10000
## Creating vectors in which to store estimates.
est.b0 <- numeric(n.sims)
est.b1 <- numeric(n.sims)
## The for loop.
for (i in 1:n.sims){
    ## Simulating the responses.
    y <- rnorm(10, means, sigma)
    ## Fitting the model.
    fit <- lm(y ~ x)
    ## Extracting the model parameters.
    est.b0[i] <- coef(fit)[1]
    est.b1[i] <- coef(fit)[2]
}
```

A simulation study

Additionally, if we wish to save confidence intervals for β_0 and β_1 estimated from each data set, we can use the following:

```
...
## Creating vectors in which to store estimates.
est.b0 <- numeric(n.sims)
est.b1 <- numeric(n.sims)
## Creating matrices in which to store CIs.
ci.b0 <- matrix(0, nrow = n.sims, ncol = 2)
ci.b1 <- matrix(0, nrow = n.sims, ncol = 2)
## The for loop.
for (i in 1:n.sims){
    ## Simulating the responses.
    y <- rnorm(10, means, sigma)
    ## Fitting the model.
    fit <- lm(y ~ x)
    ## Extracting the model parameters.
    est.b0[i] <- coef(fit)[1]
    est.b1[i] <- coef(fit)[2]
    ## Extracting the CIs.
    ci.b0[i, ] <- confint(fit)[1, ]
    ci.b1[i, ] <- confint(fit)[2, ]
}
```

A simulation study

Some results

We have simulated 10 000 data sets, have fitted a model to each of these data sets, obtaining 10 000 different estimates and confidence intervals for β_0 and β_1.

Here are the first six estimates and confidence intervals for β_0, which has a true value of 20:

```
head(est.b0)

## [1] 15.97736 25.40637 24.37645 17.48091 16.75212 30.66799

head(ci.b0)

##              [,1]     [,2]
## [1,]  5.08316477 26.87155
## [2,] 13.76995213 37.04279
## [3,] 10.45218106 38.30071
## [4,]  7.92595789 27.03587
## [5,] -0.03301697 33.53726
## [6,] 16.79376364 44.54222
```

A simulation study

Some results

We have simulated 10 000 data sets, have fitted a model to each of these data sets, obtaining 10 000 different estimates and confidence intervals for β_0 and β_1.

Here are the first six estimates and confidence intervals for β_1, which has a true value of 3:

```
head(est.b1)

## [1] 2.899190 2.222505 2.434682 3.364795 3.605515 2.084689

head(ci.b1)

##            [,1]     [,2]
## [1,] 2.0213112 3.777069
## [2,] 1.2848160 3.160195
## [3,] 1.3126329 3.556731
## [4,] 2.5948345 4.134755
## [5,] 2.2529297 4.958100
## [6,] 0.9666721 3.202706
```

A simulation study

Measuring estimator performance

How well does our linear model do at estimating β_0 and β_1? To assess this, we need to understand a desirable properties of estimators and confidence intervals.

Unbiasedness: Sometimes an estimate will be higher than the true value. Sometimes it will be lower. An estimator[1] is unbiased if, on average, it is equal to the true value.

Variance: Sometimes an estimate will be very close to the true value. Sometimes it will be far away. An estimator is 'precise' if its variance is low, which means estimates are almost always very close to one another. If the estimator variance is high then it is 'imprecise', and some estimates might be quite far from the true value.

[1] What is the difference between an estimate and and estimator?

A simulation study

Measuring estimator performance

Confidence Interval coverage: We hope the true value of the parameter will lie within the confidence interval, but this won't always happen. The 'coverage' of a method used to calculate confidence intervals is equal to the proportion of confidence intervals calculated using this method that include the true value. If a method is used to calculate 95% confidence intervals, then 95% of intervals calculated using this method should contain the true parameter value.

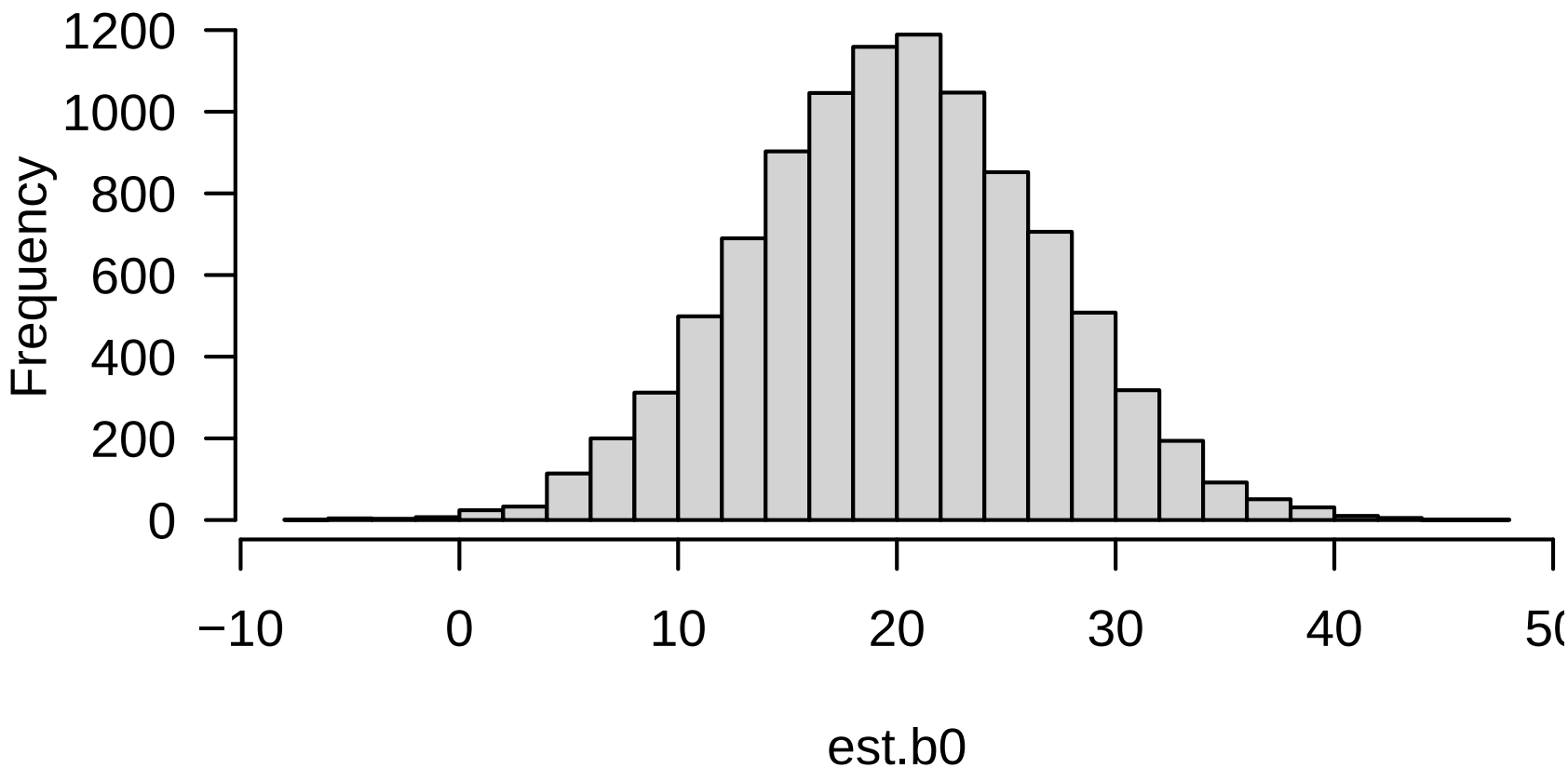
Take a look at the previous two slides. From a very informal assessment of the first six estimates and confidence intervals, how do things look? Does it look like the estimates might possibly be unbiased? How precise are they? Do the confidence intervals contain the true values?

A simulation study

Measuring estimator performance: Unbiasedness

Here is a histogram of the estimates of $\beta_0 = 20$. Does our estimator appear to be more or less unbiased?

```
hist(est.b0)
```

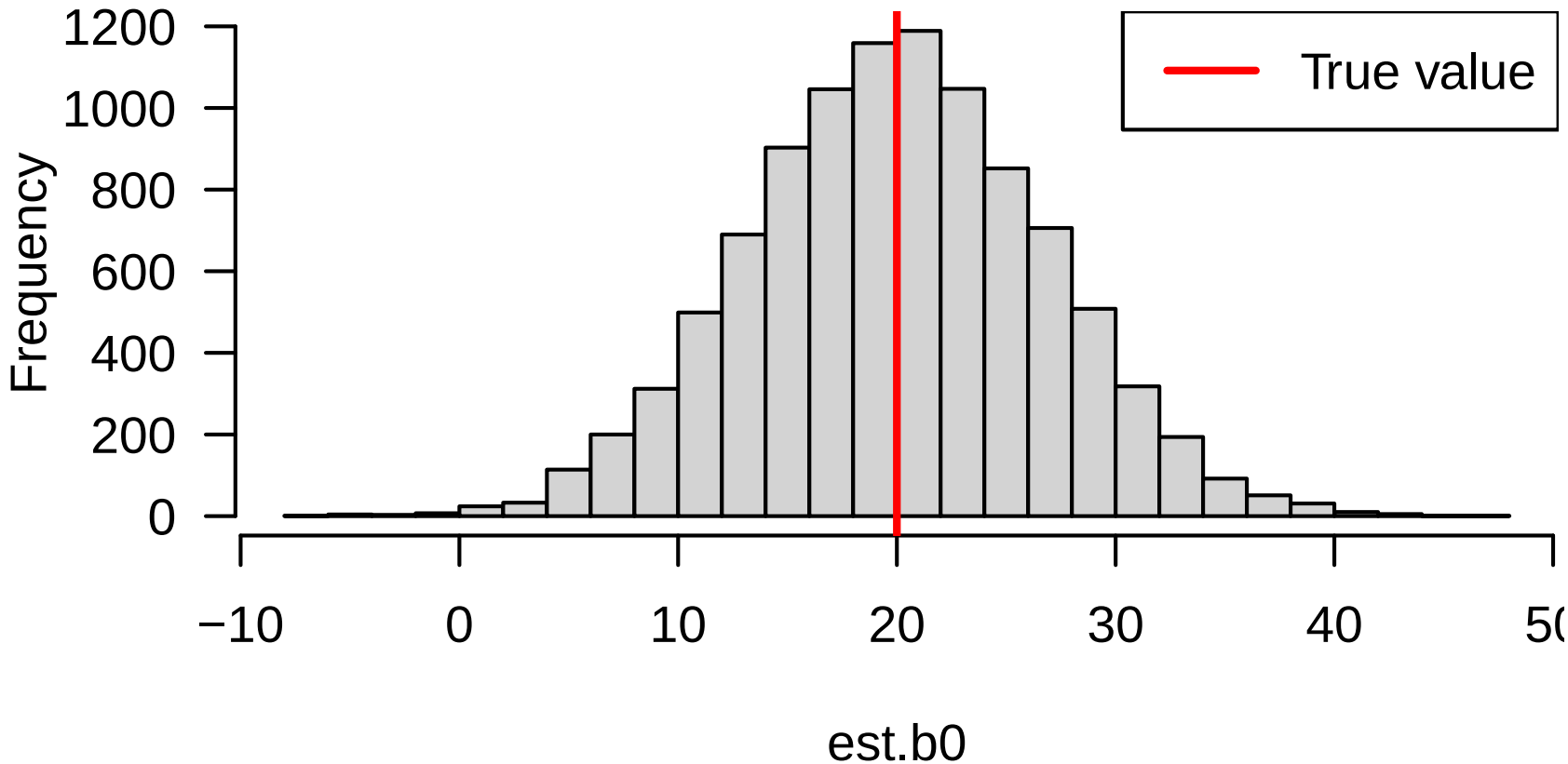


A simulation study

Measuring estimator performance: Unbiasedness

Here is a histogram of the estimates of $\beta_0 = 20$. Does our estimator appear to be more or less unbiased? Yes!

```
hist(est.b0); abline(v = b0, col = "blue")
```

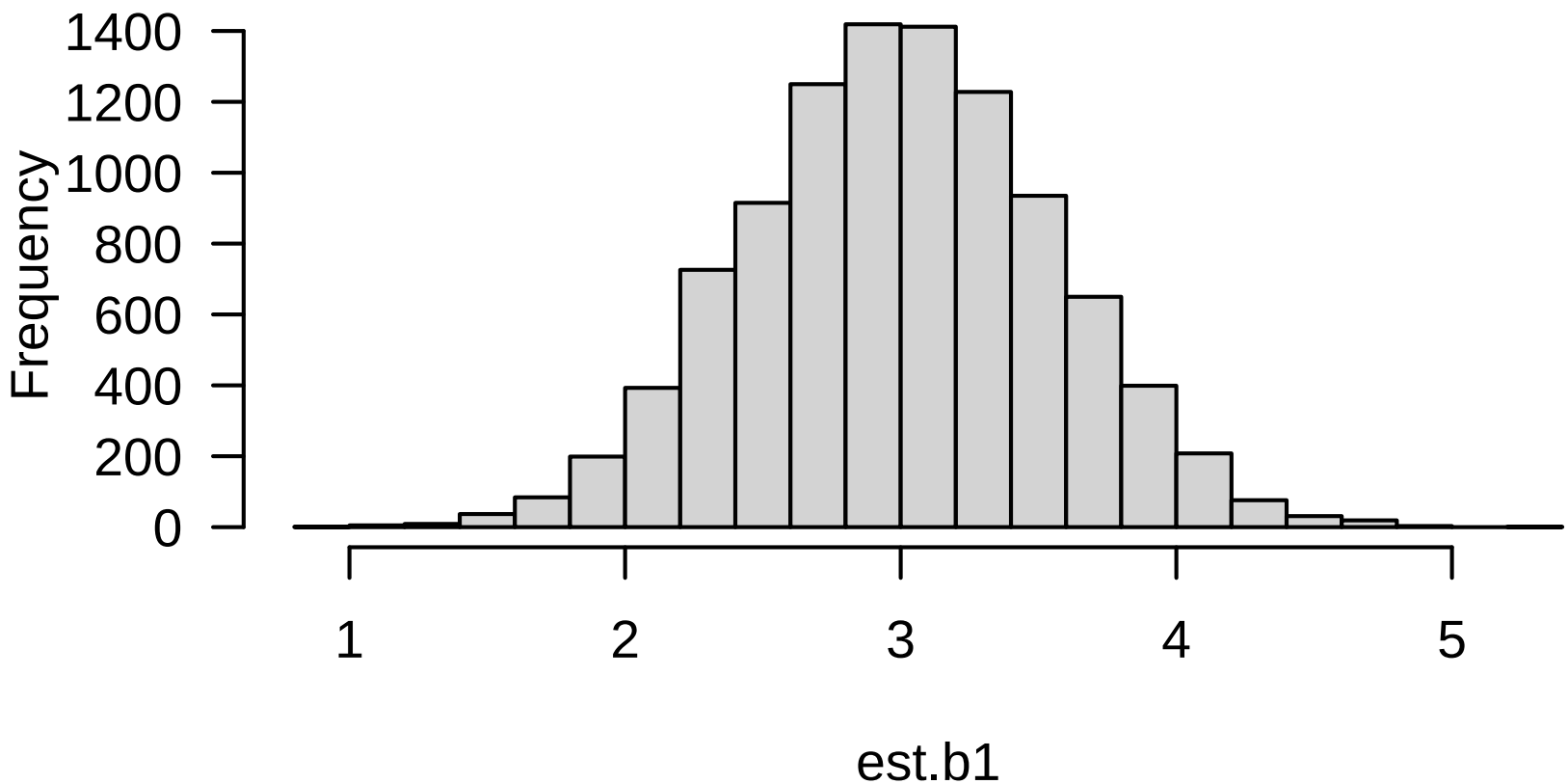


A simulation study

Measuring estimator performance: Unbiasedness

Here is a histogram of the estimates of $\beta_1 = 3$. Does our estimator appear to be more or less unbiased?

```
hist(est.b1)
```

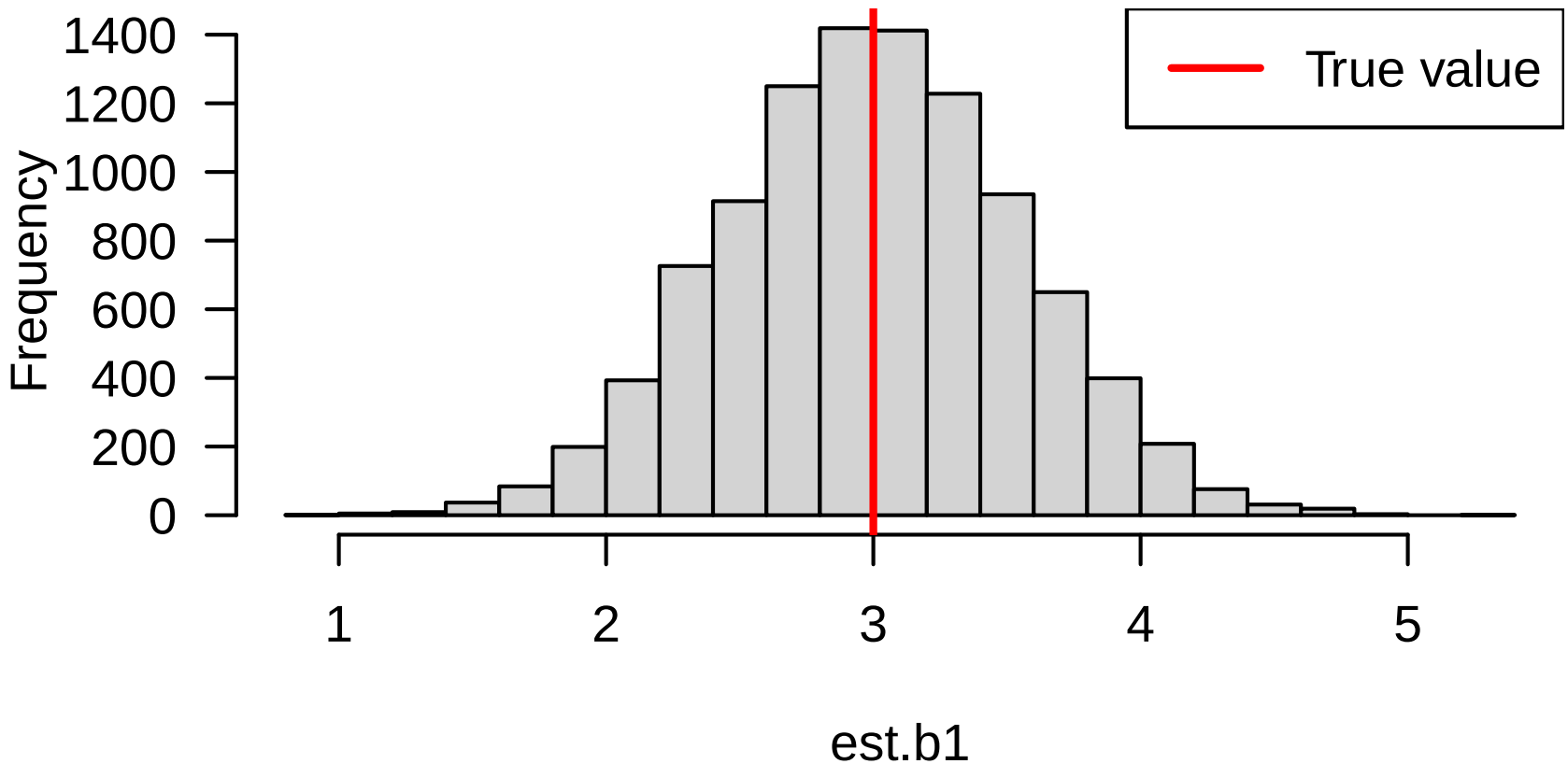


A simulation study

Measuring estimator performance: Unbiasedness

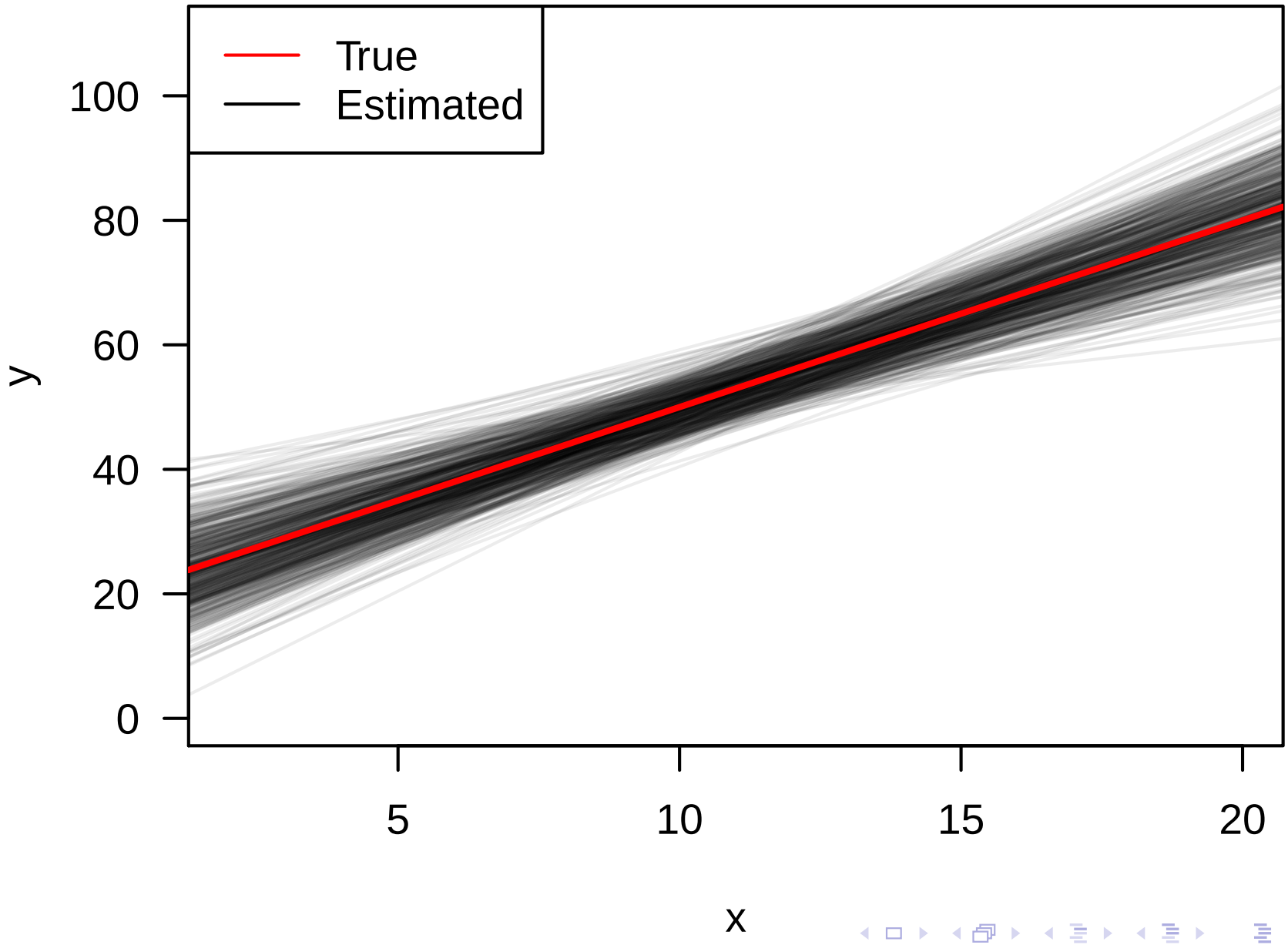
Here is a histogram of the estimates of $\beta_1 = 3$. Does our estimator appear to be more or less unbiased? Yes!

```
hist(est.b1); abline(v = b1, col = "blue")
```



A simulation study

Measuring estimator performance: Unbiasedness



A simulation study

Measuring estimator performance: Unbiasedness and variance

We can also directly compute the means, variances, and standard errors of our estimates:

For $\beta_0 = 20$:

```
mean(est.b0)

## [1] 20.00945

var(est.b0)

## [1] 46.2044

sd(est.b0)

## [1] 6.797382
```

A simulation study

Measuring estimator performance: Unbiasedness and variance

We can also directly compute the means, variances, and standard errors of our estimates:

For $\beta_1 = 3$:

```
mean(est.b1)

## [1] 2.996121

var(est.b1)

## [1] 0.3031838

sd(est.b1)

## [1] 0.5506212
```

A simulation study

Measuring estimator performance: Variance

Let's have a quick glance at the summary of the model fitted to our very first simulated data set:

```
summary(fit)

## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  17.8265     6.1527   2.897 0.019972 *
## x             3.2481     0.4958   6.551 0.000178 ***
## ---
## Residual standard error: 9.007 on 8 degrees of freedom
## Multiple R-squared:  0.8429,Adjusted R-squared:  0.8232
## F-statistic: 42.92 on 1 and 8 DF,  p-value: 0.0001782
```

A simulation study

Measuring estimator performance: Variance

See how the standard errors are close to the standard deviations of our estimated parameters from the simulated data sets?

A standard error estimates the standard deviation of the parameter's ' sampling distribution': the distribution of estimates we would get if we were to repeatedly resample from the population of interest and fit the same model.

In other words, by fitting a model to just one single sample, we are able to estimate how variable our estimates would be if we were to resample from the population a large number of times.

A simulation study

Measuring estimator performance: Confidence interval coverage

For each confidence interval calculated from each simulated data set, we can determine whether or not it captured the true value.

```
head(ci.b0)

##               [,1]     [,2]
## [1,]  5.08316477 26.87155
## [2,] 13.76995213 37.04279
## [3,] 10.45218106 38.30071
## [4,]  7.92595789 27.03587
## [5,] -0.03301697 33.53726
## [6,] 16.79376364 44.54222
```

We can compute if we captured the true value:

```
head((b0 >= ci.b0[, 1]) & (b0 <= ci.b0[, 2]))

## [1] TRUE TRUE TRUE TRUE TRUE TRUE
```

The first six confidence intervals all captured $\beta_0 = 20$.

A simulation study

Measuring estimator performance: Confidence interval coverage

```
b0.ci.captured <- (b0 >= ci.b0[, 1]) & (b0 <= ci.b0[, 2])
table(b0.ci.captured)

## b0.ci.captured
## FALSE  TRUE
##   491  9509

mean(b0.ci.captured)

## [1] 0.9509
```

Out of the 10 000 confidence intervals computed for β_0, 9509 of them contained the true value $\beta_0 = 20$.

So our method for calculating 95% confidence intervals works for β_0, because approximately 95% of intervals calculated using this method contain the true value.

A simulation study

Measuring estimator performance: Confidence interval coverage

We can do the same for $\beta_1 = 3$:

```
b1.ci.captured <- (b1 >= ci.b1[, 1]) & (b1 <= ci.b1[, 2])
table(b1.ci.captured)

## b1.ci.captured
## FALSE  TRUE
##   484  9516

mean(b1.ci.captured)

## [1] 0.9516
```

Out of the 10 000 confidence intervals computed for β_0, 9516 of them contained the true value $\beta_1 = 3$.

So our method for calculating 95% confidence intervals works for β_1, because approximately 95% of intervals calculated using this method contain the true value.

Summary

In this handout we have used simulation to verify that a simple linear model's estimates and confidence intervals do what they are supposed to if the model's assumptions are met.

We could extend this idea to test various 'what if?' scenarios:

- What if the response is not normally distributed?
- What if the response does not have constant variance?
- What if the observations are not independent?

We can simulate data that are non-normal and/or have nonconstant variance and/or are not independent, and see how well our methods still work.

Over the next few handouts, we will see other handy uses of simulation methods, including calculation of confidence intervals and testing for goodness-of-fit in scenarios where our standard methods may not work.