Introduction to Estimation

```r
library(fastR2)
```

Introduction

This notebook will introduce the second important procedure in statistical inference, estimation. Our goal is to introduce the notion of **confidence intervals**. For us, hypothesis testing and estimation, especially confidence intervals will be the most important statistical concepts.

Eventually we will learn about the **bootstrap** method for estimating the confidence interval of a parameter or test statistic. The bootstrap is a computational method that is absolutely essential in data science and machine learning. However, before we begin to discuss the bootstrap in detail, it is very helpful to get a feel for what is known as point estimation. We will do this by way of a simple example.

Point Estimation Example

Consider the following problem. Someone gives us a coin. If we toss the coin, we will land heads with some probability ρ. However, we do not necessarily know *a priori* what the numerical value of ρ is. In this case, we call ρ a **population** parameter because it is **fixed but unknown**. What we will attempt to do is to use data to estimate a value for ρ in the hopes that our estimate will be reasonably accurate. Of course it will be necessary to assess the accuracy of our estimate.

How can we estimate ρ from data? If we toss the coin say n times then, as we have seen, a binomial distribution with probability of success equal to ρ provides a good model for the probability that we obtain n_{heads}. Perhaps our understanding of this distribution can help us to estimate ρ. This is indeed the case. It should be fairly intuitive that an reasonable estimate, denoted by $\hat{\rho}$, for ρ is

$\hat{\rho} = \frac{n_{\text{heads}}}{n}$

This result can be derived mathematically using something called maximum likelihood but this is beyond the scope of this course. The important point for us is that the **estimator** $\hat{\rho}$ is a **random variable** because each time we perform the experiment we will possibly obtain a different number for n_{heads} and hence a different value for

$\hat{\rho} = \frac{n_{\text{heads}}}{n}$.

Just to emphasize this, a random variable that provides a means (like a formula) to estimate a population parameter is called an estimator and a specific value for an estimator obtained from sample data is called an **estimate**, or a point estimate because it will be a single number.

Let's exemplify all of this by way of simulation.

Estimation Simulation

Let's fix a value for the population parameter ρ. Note that in reality we will not typically know the value of a population parameter.

```r
# population parameter value fixed for simulation purposes
rho <- 0.65
```

Now let's simulate tossing a coin with fixed probability of success 0.65 the value of which we will pretend that we do not know. Suppose we toss the coin 12 times. We can simulate this by

```
n <- 12
(coin_toss_sim <- rbinom(n,1,rho))
```

```
##  [1] 1 1 0 1 1 1 1 1 0 1 1 1
```

Remeber that 1 corresponds to heads while 0 corresponds to tails. Each time we run this code we will get a different number of heads. Then our estimate for ρ is

```
sum(coin_toss_sim)/n
```

```
## [1] 0.8333333
```

But again, this value will change with each run of the simulation.

Let's put of of this code together in a function and run the function many times.

```
est_sim <- function(rho=0.65,n=12){
  return(sum(rbinom(n,1,rho))/n)
}
```

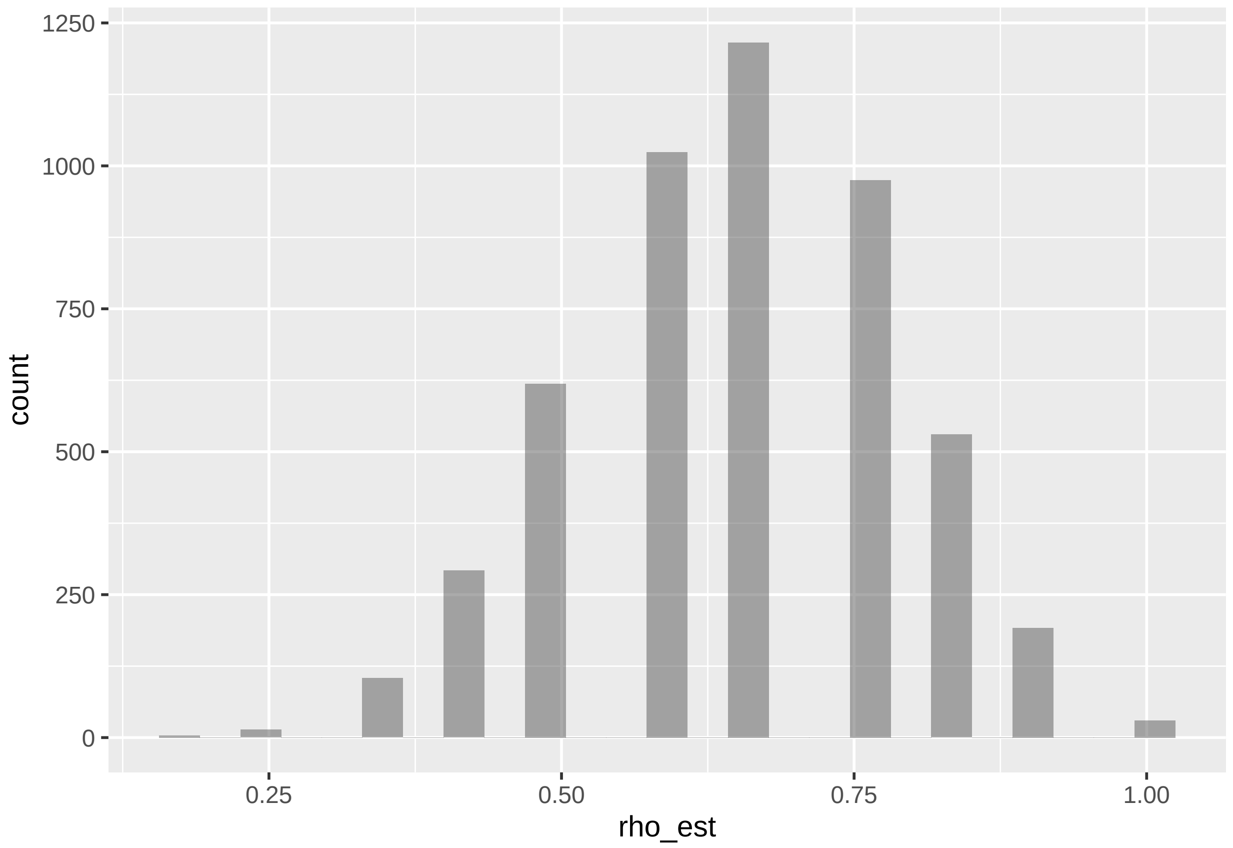
Let's try out our function:

```
est_sim()
```

```
## [1] 0.8333333
```

Now let's run our function some large number of times, collect the results in a data frame and plot a histogram of the values we obtain.

```
N <- 5000
est_results <- do(N)*c(rho_est = est_sim())
est_results %>% gf_histogram(~rho_est)
```



What values occurs most often? Definitely values that are pretty close to 0.65.

This shoud provide some assurance that, at least with reasonably high probability, $\hat{\rho} = \frac{n_{\text{heads}}}{n}$ gives a good estimate for ρ. The issue is that, while with simulation it is easy to run the same experiment over and over again some large number of times, in real life it is often difficult to impossible to repeat an experiment. In this case, we get a single estimate for a population parameter. How do we assess the accuracy of our estimate? One common approach is with a confidence interval. Let's explore this idea.

Intro to Confidence Intervals

Let's begin by simulating some more sample data and compute the corresponding estimate.

```r
coin_toss_sim <- rbinom(n,1,rho)
n_heads <- sum(coin_toss_sim)
(rho_est <- n_heads/n)
```

```
## [1] 0.4166667
```

In R, we can use a built-in function to estimate the 95% confidence interval for our estimate as follows:

```r
(int_est <- prop.test(n_heads,n)$conf.int)
```

```
## [1] 0.1649925 0.7140072
## attr(,"conf.level")
## [1] 0.95
```

The details for how this is obtained is not important right now. The point is to get an intuitive sense for what a confidence interval is.

What we will do is repeat the process of

1) obtaining sample data
2) computing an estimate
3) obtaining the corresponding 95% confidence interval:

The following code does this:

```r
est_ci_sim <- function(rho=0.65,n=12){
  samp_dat <- sum(rbinom(n,1,rho))
  prop_est <- samp_dat/n
  int_est <- prop.test(samp_dat,n)$conf.int
  return(c(est_val=prop_est,lower=int_est[1],upper=int_est[2]))
}

N <- 100
est_ci_res <- do(N)*c(est_ci_sim())
```

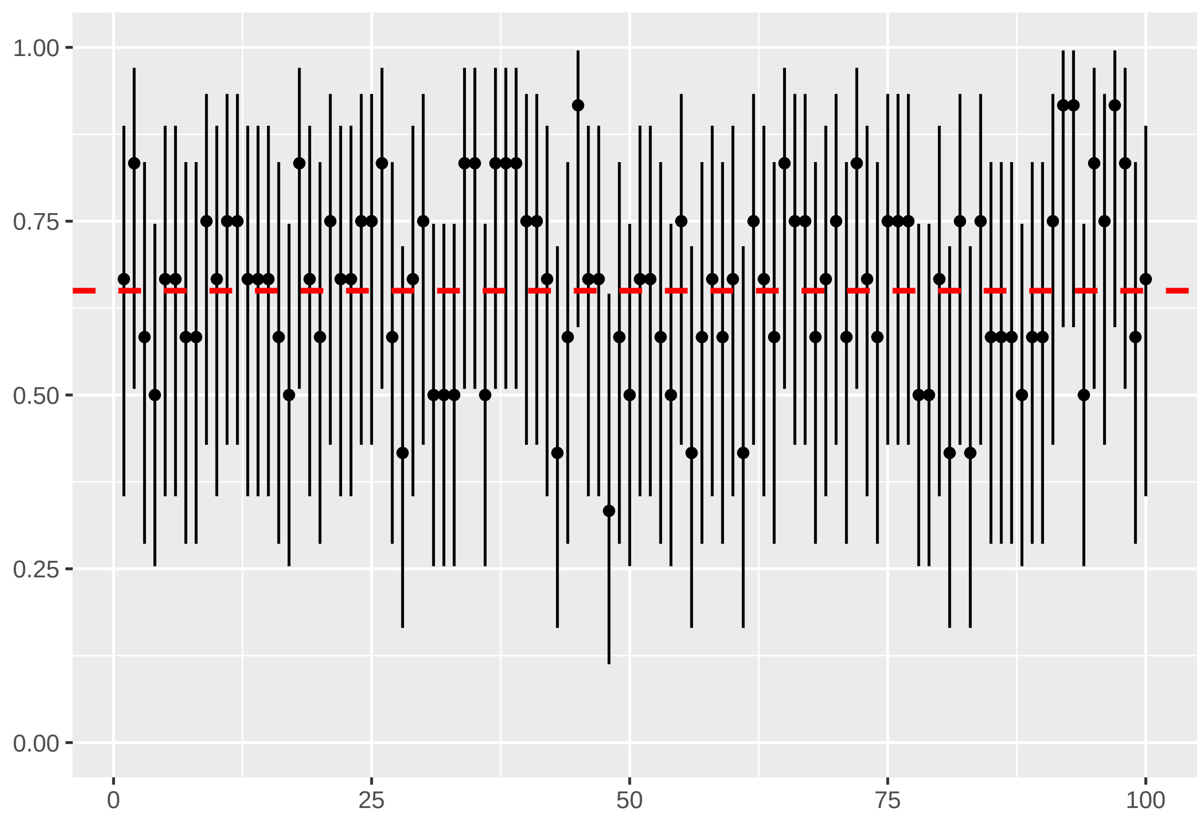
Let's look at what we have obtained:

```r
head(est_ci_res)
```

```
##     est_val     lower     upper
## 1 0.6666667 0.3543690 0.8872714
## 2 0.8333333 0.5088142 0.9705909
## 3 0.5833333 0.2859928 0.8350075
## 4 0.5000000 0.2537816 0.7462184
## 5 0.6666667 0.3543690 0.8872714
## 6 0.6666667 0.3543690 0.8872714
```

A plot is very helpful:

```r
est_ci_res %>% ggplot(aes(x=seq(1,nrow(est_ci_res)),y=est_val)) +
  geom_point() +
  geom_linerange(aes(x=seq(1,nrow(est_ci_res)),ymin=lower,ymax=upper)) +
  ylim(c(0,1)) +
  geom_hline(yintercept = rho,color="red",linetype="dashed",size=1) +
  labs(x="",y="")
```



This shows

1) the true population value (dashed red horizontal line)
2) 100 point estimate values (black dots)
3) the corresponding 95% confidence interval for each point estimate (vertical black lines)

Notice that the true population parameter falls within all but just a few of the confidence intervals (vertical black lines). In fact, we can compute the percentage of confidence intervals that contain the population parameter:

```
k <- 0
for (i in 1:nrow(est_ci_res)){
  if (est_ci_res$lower[i] <= rho & rho <= est_ci_res$upper[i]){
    k <- k + 1
  }
}
k/nrow(est_ci_res)
```

```
## [1] 0.99
```

This should be close to 95%.

To simultaneously develop greater understanding for and to learn a method to compute (approximate) confidence intervals, we will now introduce the bootstrap method.

Bootstrap for Confidence Intervals

We begin by writing a function in R that will input some sample data, sample **with replacement** from the data, and compute the resulting point estimate for our parameter of interst:

```
boot_prop <- function(coin_data){
  return(sum(sample(coin_data,replace = T))/length(coin_data))
}
```

Here's a an example call of this function:

```
boot_prop(coin_toss_sim)
```

```
## [1] 0.5
```

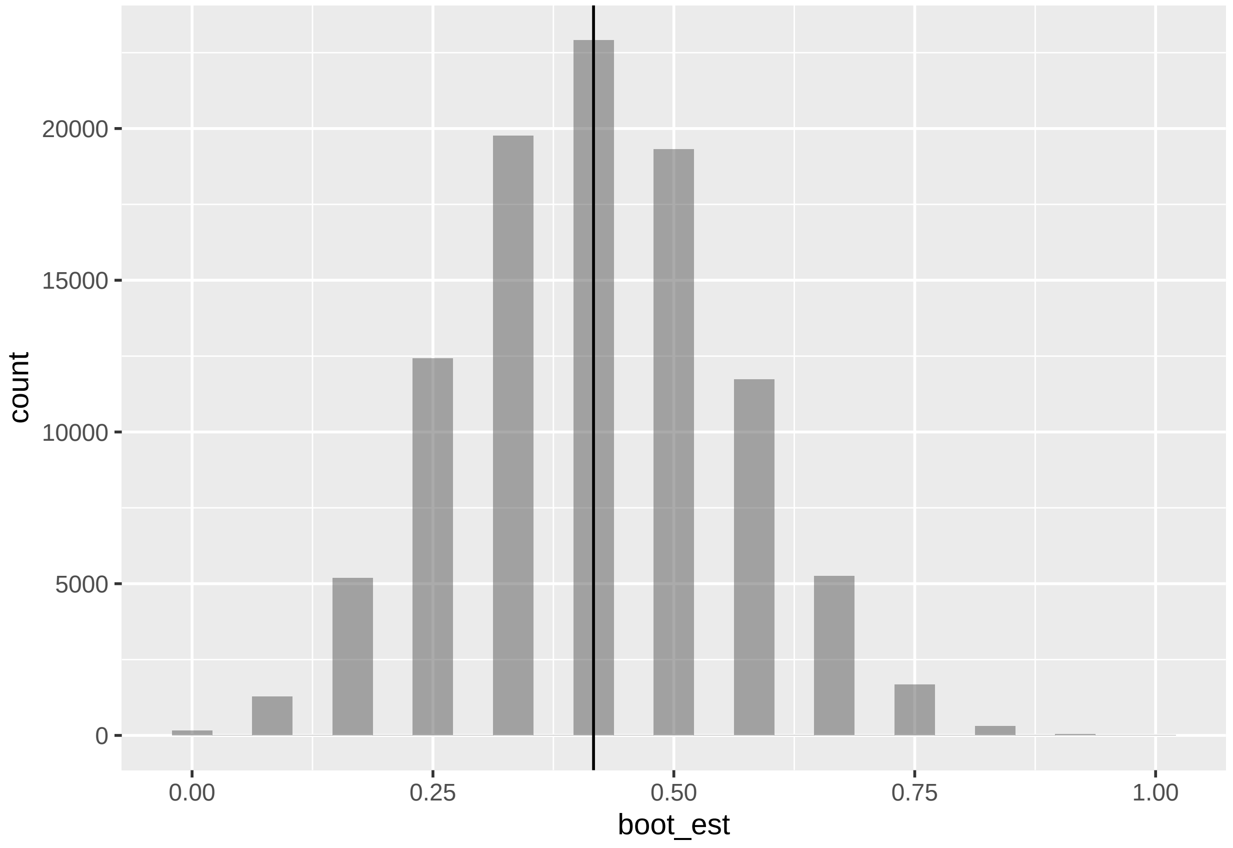
Now what we will do is call this function repeatedly some large number of times:

```
N <- 10^5
boot_res <- do(N)*c(boot_est = boot_prop(coin_toss_sim))
```

The reason why we do this is because resampling with replacement from a sample allows us to obtain what is called the **bootstrap** distribution for our statistic. The bootstrap distribution is effectively an estimate for the sampling distribution that is obtained by way of simulation. We can plot the bootstrap distribution for our running example:

```
boot_res %>% gf_histogram(~boot_est) %>%
  gf_vline(xintercept = rho_est)
```

```
## Warning: geom_vline(): Ignoring `mapping` because `xintercept` was provided.
```



The vertical black line is the value of our estimate from the original data.

As we mentioned before, the standard deviation of the sampling distribution for a statistic is called the standard error. Since the bootstrap distribution estimates the smapling distribution, the standard deviation of the bootstrap distribution (the so-called **bootstrap standard error**) provides us with an estimate for the standard error of our estimate:

```r
sd(boot_res$boot_est)
```

```
## [1] 0.1418566
```

There are two ways common ways to use the bootstrap to estimate confidence intervals. One way is to use the bootstrap standard error because confidence intervals are usually obtained by a formula such as

$CI = (\text{estimate} - \text{standard error} \times \text{some factor}, \text{estimate} + \text{standard error} \times \text{some factor})$

where the factor is chosen depending on the **confidence level** (*e.g.*, 95%). If a parameter comes from a normal distribution, then the scale factor corresponding to a 95% confidence level is about 2, therefore our confidence interval estimate using the bootstrap is:

```r
rho_est + 2*c(-1,1)*sd(boot_res$boot_est)
```

```
## [1] 0.1329534 0.7003800
```

Another way to estimate confidence intervals using the bootstrap is to obtain a **bootstrap percentile interval** from the bootstrap distribution. For example, a 95% bootstrap percentile in our example would be obtained with:

```r
quantile(boot_res$boot_est,c(0.025,0.975))
```

```
##      2.5%     97.5% 
## 0.1666667 0.6666667
```

Notice that this is reasonably close to what we obtained before.

Using classical theory, a 95% confidence interval for our example parameter is obtained by:

```r
prop.test(n_heads,n)$conf.int
```

```
## [1] 0.1649925 0.7140072
## attr(,"conf.level")
## [1] 0.95
```

Again, this is pretty close to what we obtained using the bootstrap technique.