Monte Carlo Simulations with SLR

1. Introduction

Monte Carlo simulations are used to estimate the probabilities associated with various outcomes of a random event. Individual random events are unpredictable: we cannot know for sure the outcome of an individual random event (one coin toss). However, if we were able to repeat the random event many times (i.e. an infinite number of times), the probabilities associated with each possible outcome is predictable (with one million coin tosses, we would expect close to half of tosses to be heads).

Simulations are carried out on computers to quickly repeat these random events many times, so we can:

- List all possible outcomes, and
- Estimate the probabilities associated with each possible outcome.

2. History

Researchers working on the Manhattan project in Los Alamos, NM, wanted to investigate all the possible outcomes associated with deploying the atomic bomb due to the random nature of nuclear fission.

Monte Carlo simulations were developed and used by Stanislaw Ulam and John von Neumann. The work needed to be secret and so required a code name. A colleague on the project, Nicholas Metropolis, suggested the name Monte Carlo, after the Monte Carlo casino in Monaco where Ulam's uncle was known to borrow money from relatives to gamble. It was known that a number of researchers on the Manhattan project liked playing poker with each other during evenings.

One of the conclusions from applying Monte Carlo simulations on the atomic bomb was that it was close to impossible to blow up the world.

3. Simulations in Linear Regression

Recall in a linear regression setting, we have

$$y = \beta_0 + \beta_1 x + \epsilon, \tag{1}$$

where ϵ are assumed to be i.i.d. normal with mean 0 and constant variance σ^2.

The parameters β_0, β_1 are estimated using the method of least squares, by minimizing the sum of squared residuals, SS_{res}.

We know from statistical theory that, if the assumption regarding the errors ϵ is correct, then we have the following:

1. $\mathrm{E}(\hat{\beta}_1) = \beta_1$, $\mathrm{E}(\hat{\beta}_0) = \beta_0$

Note: An estimator is **unbiased** if its expected value is exactly equal to the parameter it is estimating.

2. The variance of $\hat{\beta}_1$ is

$$\mathrm{Var}(\hat{\beta}_1) = \frac{\sigma^2}{\sum (x_i - \bar{x})^2} \tag{2}$$

3. The variance of $\hat{\beta}_0$ is

$$\mathrm{Var}(\hat{\beta}_0) = \sigma^2 \left[\frac{1}{n} + \frac{\bar{x}^2}{\sum (x_i - \bar{x})^2} \right] \tag{3}$$

4. $\hat{\beta}_1$ and $\hat{\beta}_0$ both follow a normal distribution.

Next, we will use simulations to verify that $\hat{\beta}_1$ is an unbiased estimator of β_1 and that $\mathrm{Var}(\hat{\beta}_1) = \frac{\sigma^2}{\sum (x_i - \bar{x})^2}$, in a one predictor setting.

a. Create function that simulates response given predictor

We need to simulate the values of the response variable y per the relationship and assumptions expressed in (1). We will write a function to simulate these values. The function will require us to provide the values of the predictors, the regression parameters, and the standard deviation of the error terms.

```r
####################################################
##Function to simulate y values from linear model##
####################################################

gety <- function(x,intercept,slopes,eps.sigma)
{
  n<-nrow(x)
  y <- intercept + x%*%slopes + rnorm(n,0,eps.sigma)
  return(y)
}
##x needs to be n by k matrix; its the design matrix minus the column of 1s
```

b. Initialize the values

In order to simulate the y variable from the `gety()` function, we need to:

- specify the sample size,
- generate some values for the predictor (it is not important how we generate these values; the distribution of the predictor do not play a role and no assumptions are made about it),
- specify values for the regression parameters β_0, β_1,
- specify the value for the standard error of the error term, σ,
- specify how many times we want to run (or replicate) the simulation (ideally, this should be a large number; the larger it is, the closer our estimated probabilities will be to their true values),

```
##sample size
n<-1000

##if you want to reproduce results
set.seed(41)
##generate the values of x1
x1<-runif(n, -2, 2)
X<-cbind(x1)

##initialize values for simulation
beta0 <- 1 ##intercept
betas <- 2 ##slope
sig <- sqrt(2) ##sd of error term

##run simulation 10000 times
reps <- 10000
```

c. Store required values

We are planning to show that $E(\hat{\beta}_1) = \beta_1$; i.e. the long run average of the estimated slopes is equal to the value of the actual slope. So we need to store the values of the estimated slopes from each replication.

```
##create an array to store the estimated slope and its SE from each rep
store.slope<-array(0,reps)
```

d. Run simulation

We next write a loop that runs this simulation 10000 times. For each rep, we will

- simulate the response variable
- regress the response variable against the predictor
- record the estimated coefficient, $\hat{\beta}_1$.

I also added a line of code before the loop, and two lines of code after the loop, to compute the time it takes to run this simulation for 10000 reps.

```
##if you are curious about how long your loop runs
start_time <- Sys.time() ##start time of loop

for (i in 1:reps)

{
  y<-gety(X, intercept=beta0, slope=betas, eps.sigma=sig)

  ##use least squares to obtain regression equation on simulated data
  result<-lm(y~X)

  ##store the estimated slope from this rep
  store.slope[i]<-result$coeff[2]

  ##optional line if you want to see how fast your loop is going
  ##print(paste("Iteration", i))
}

end_time <- Sys.time() ## end time of loop
end_time - start_time ##time taken by loop
```

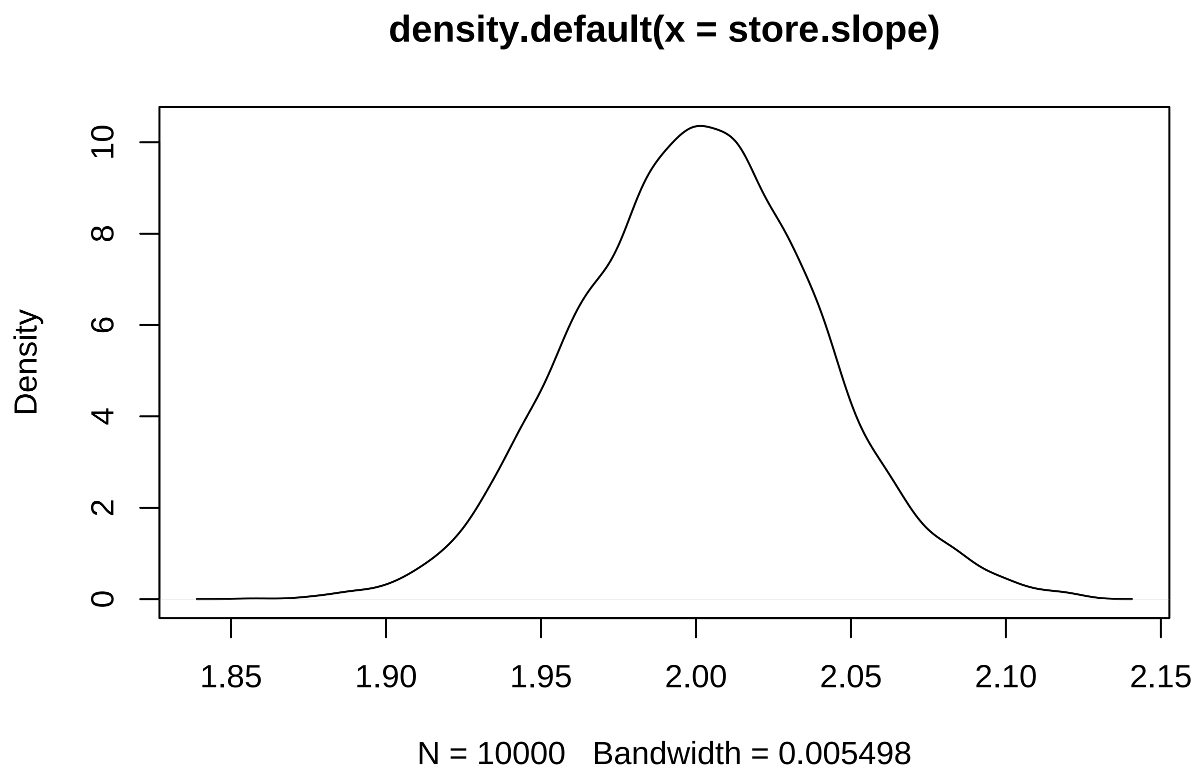
```
## Time difference of 5.860529 secs
```

e. Use results from simulation

Sampling distribution of estimated parameters

We now use the results from our simulation. Statistical theory tells us that the distribution of $\hat{\beta}_1$ is bell-shaped. We create a density plot of the 10000 estimated slopes. The shape is consistent with theory.

```
##distribution of est slope
plot(density(store.slope))
```



Estimated parameters are unbiased

Statistical theory tells us the long run average of $\hat{\beta}_1$ is equal to β_1, if the regression assumptions are met. So the average of the 10000 estimated slopes should be approximately equal to 2 (if we had run the loop an infinite number of times, the average of the estimated slopes will be equal to 2).

```
##bias of est slope
mean(store.slope)-betas
```

```
## [1] 0.000407397
```

Note: the expectation of an estimator minus its true parameter is the bias of the estimator.

Variance of estimated parameters

With one predictor, $Var(\hat{\beta}_1) = \frac{\sigma^2}{\sum(x_i = \bar{x})^2}$. We can compare the variance of the 10000 estimated slopes to $\frac{\sigma^2}{\sum(x_i = \bar{x})^2}$. With 10000 reps, these values should be close.

```
##empirical variance of est slope
var(store.slope)
```

```
## [1] 0.001485931
```

```
##theoretical formula for variance of estimated slope
sig^2/sum((X-mean(X))^2)
```

```
## [1] 0.001511193
```

Our simulations are consistent with statistical theory. Simulations are often used in statistical research to confirm statistical theory, after proving the theory mathematically.