Problem Set 5 Part I Simulation

Olivia Bogiages

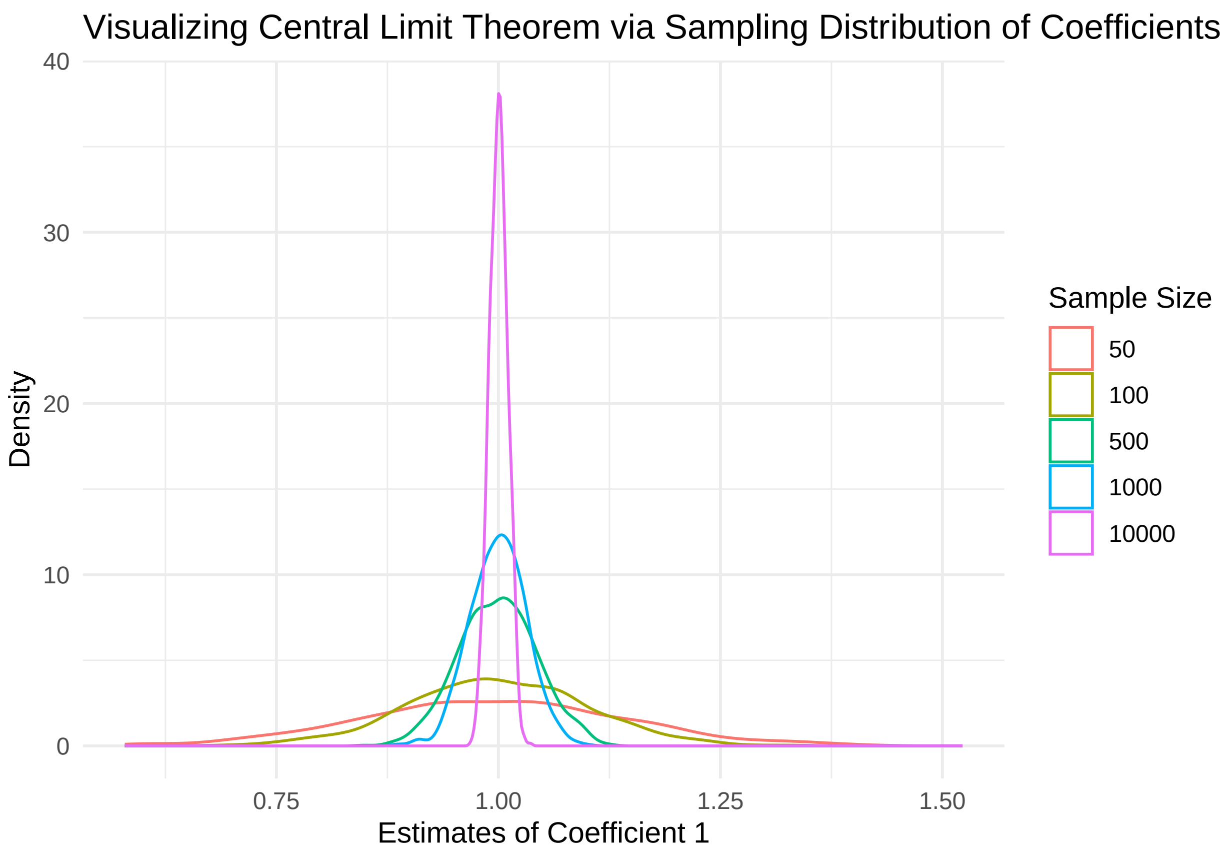
2025-12-11

Create a simulated data set with a dependent variable that is a linear function of a treatment variable and a confounding variable. Fit a linear model for the true data generating process and print the summary table.

```r
set.seed(123)
library(ggplot2)
#Create Variables
z<-rnorm(100000,0,1) #confounder
x<- rnorm(100000,0,1) + z #independent variable
y<- rnorm(100000,0,1) +z+x #dependent variable

#True Process
population<-data.frame(x,y,z)
model_population<-lm(y~x+z, data=population)
summary(model_population)
```

```
##
## Call:
## lm(formula = y ~ x + z, data = population)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.1296 -0.6739  0.0026  0.6723  4.5211
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.0005948  0.0031617  -0.188    0.851
## x            1.0000543  0.0031494 317.537   <2e-16 ***
## z            0.9978776  0.0044573 223.876   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9998 on 99997 degrees of freedom
## Multiple R-squared:  0.833,  Adjusted R-squared:  0.833
## F-statistic: 2.494e+05 on 2 and 99997 DF,  p-value: < 2.2e-16
```

```r
#different sample sizes
sample_size<-c(50,100,500,1000,10000)

#how many regressions run
n_sim<-1000
res <- data.frame(
  sample_size   = numeric(5000),
  sim           = integer(5000),
  intercept     = numeric(5000),
  coefficient1  = numeric(5000),
  coefficient2  = numeric(5000)
)

N <- nrow(population)  # observations in population
row <- 1

#Create for loop for each sample size
 for (n in sample_size) {
   #create for loop to repeat 10000 times
   for (i in 1:n_sim) {
     #randomly draw from population
    samp_index<-sample(1:N, size=n, replace=TRUE)
#create the sample
 samp<-population[samp_index,]

#fit the linear model for each sample
model<-lm(y~x+z, data=samp)
#coefficients for each model
coefs<-coef(model)


#save results
res$sample_size[row]<-n #denotes sample size
res$sim[row]<-i #denote regression coefficient
res$intercept[row]<-coefs["(Intercept)"]#saving coefficients
res$coefficient1[row]<-coefs["x"]
res$coefficient2[row]<-coefs["z"]

row<-row+1 #denoting the move to the next row

   }
 }


#make sample size into categorical variable
res$sample_size <- factor(res$sample_size)

ggplot(res, aes(x = coefficient1, color = sample_size)) +
  geom_density() +
  labs(
    x = "Estimates of Coefficient 1",
    y = "Density",
    color = "Sample Size",
    title = "Visualizing Central Limit Theorem via Sampling Distribution of Coefficients"
  ) +
  theme_minimal()
```



```
#new results data fram
res1 <- data.frame(
  sample_size    = numeric(n_sim),
  sim            = integer(n_sim),
  intercept      = numeric(n_sim),
  coefficient1   = numeric(n_sim),
  coefficient2   = numeric(n_sim)
)

N <- nrow(population)
n <- 500
row <- 1

for (i in 1:n_sim) {

# bootstrap sample
samp_index1 <- sample(1:N, size = n, replace = TRUE)
samp1 <- population[samp_index1, ]

# fit model
model <- lm(y ~ x + z, data = samp1)
coefs1 <- coef(model)

# save results
res1$sample_size[row]  <- n
res1$sim[row]          <- i
res1$intercept[row]    <- coefs1["(Intercept)"]
```

```r
res1$coefficient1[row] <- coefs1["x"]
res1$coefficient2[row] <- coefs1["z"]

row <- row + 1
}

# Bootstrapped SE for the treatment variable (x)
boot_se <- sd(res1$coefficient1)
boot_se
```

```
## [1] 0.04416682
```

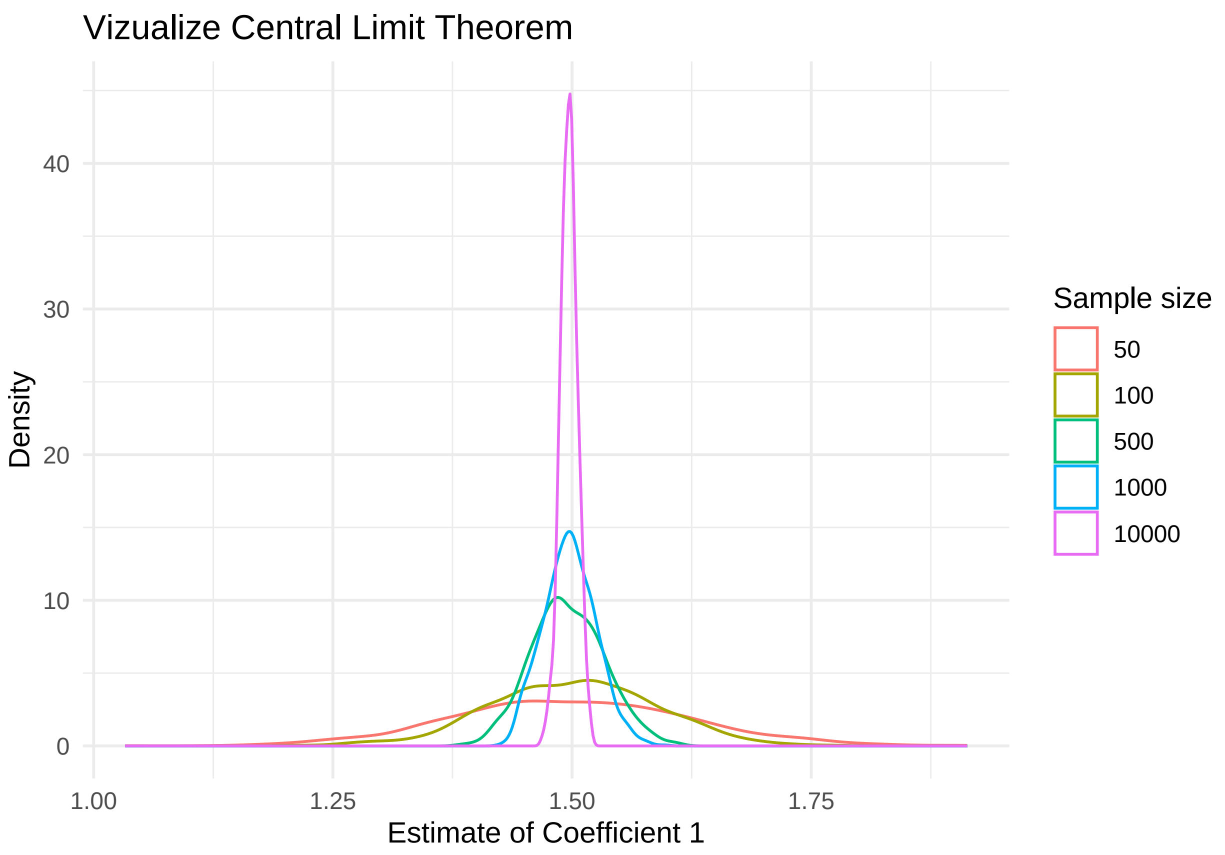
Compute the bootstrapped standard error for the coefficient of the treatment variable.

```r
#collect the data and define the statistic
# resample data with replacement
#calculate the statistic for each sample
#calculate the sd of the calculated statistics
print(sd(res1$coefficient1))
```

```
## [1] 0.04416682
```

```r
#create a data results frame #number of samplesizes (5) multiplied by number of regressions (1000)
res2 <- data.frame(
  sample_size   = numeric(5000),
  sim           = integer(5000),
  intercept     = numeric(5000),
  coefficient1  = numeric(5000),
  coefficient2  = numeric(5000)
)

N<-nrow(population)
row<-1 #start at row 1
 #create a for loop to run the code for each sample size
for (n in sample_size){
  for (i in 1:n_sim){
    #randomly choose which observations to draw from the population
samp_index<-sample(1:N, size=n, replace=TRUE)
samp2<-population[samp_index,]

#fit the model
model2<-lm(y~x, data=samp2)
coefs2<-coef(model2) #coefficient for each model

#save results
res2$sample_size[row]<-n #denoting the sample size
res2$sim[row]<-i #denote the regression
res2$intercept[row]<-coefs2["(Intercept)"]#saving coefficeints
res2$coefficient1[row]<-coefs2["x"]
res2$coefficient2[row]<-coefs2["z"]

row<-row+1 #move to the next row

  }
}
```

```r
res2$sample_size<-as.factor(res2$sample_size)

#vizualize
ggplot(res2, aes(x=coefficient1, color=sample_size)) +
  geom_density()+
  labs(
    x= "Estimate of Coefficient 1",
    y="Density",
    color="Sample size",
    title= "Vizualize Central Limit Theorem "
  ) +
  theme_minimal()
```



How do your results differ? The results, as the sample size gets bigger, becomes more concentrated. The spead decreases and the mean approaches the true mean. This demonstrate the central limit theorem. What does this imply about statistical tests based on a coefficient's sampling distribution? Given that the second simulation is biased, this implies that the t-distribution does not tell us the true mean even with a larger sample size because the model is biased.

"