Problem Set 5

Fernanda Valdez

2025-12-06

PART 1: SIMULATION 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
rm(list=ls())
set.seed(1239)
library(ggplot2)
```

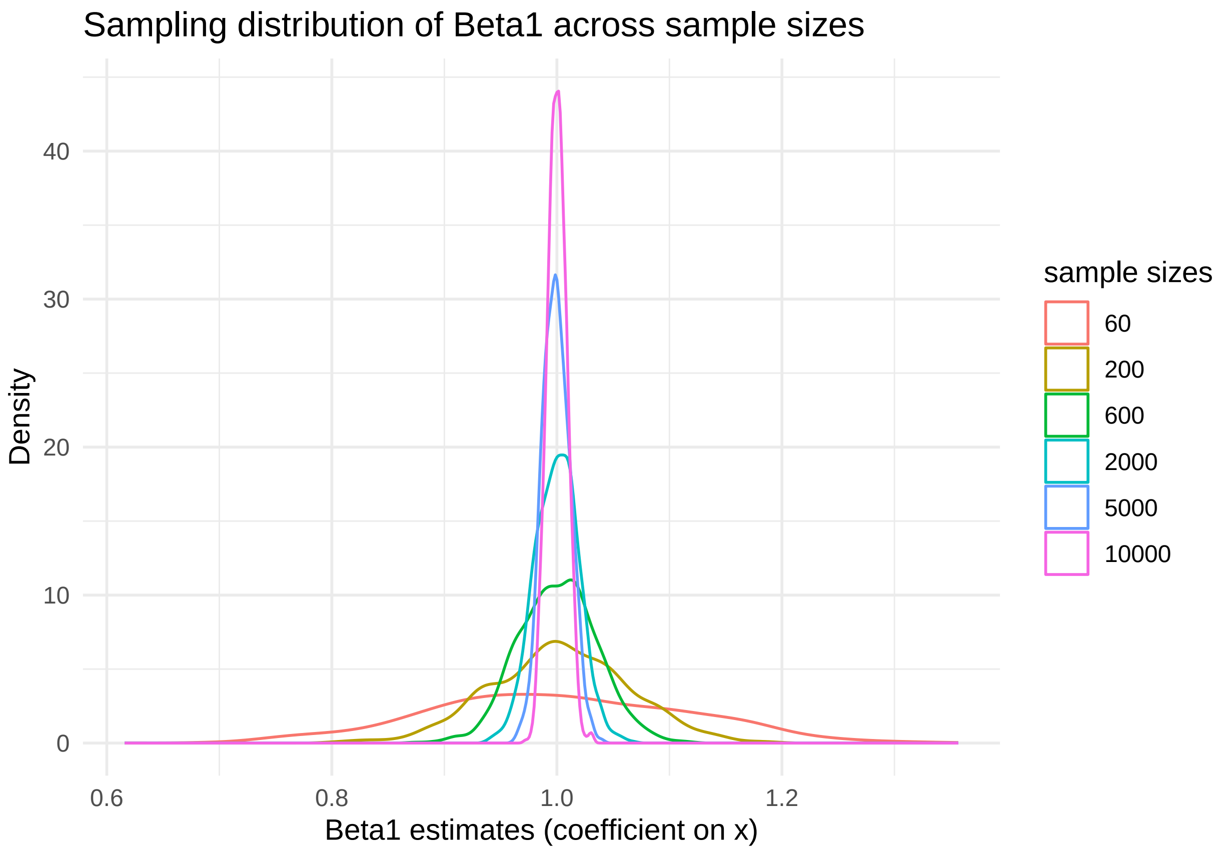
```r
# Data set
# confounder = z
z <- rnorm(1000000,3,5)
# independent = x
x <- rnorm(1000000,6,8) + z
# dependent = y
y <- rnorm(1000000,2,7) + x + z


# Data frame for the true DGP
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
## -31.6015  -4.7066  -0.0065   4.7209  31.3425
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.9799316  0.0097017   204.1   <2e-16 ***
## x           1.0000286  0.0008737  1144.6   <2e-16 ***
## z           1.0029425  0.0016498   607.9   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.996 on 999997 degrees of freedom
## Multiple R-squared:  0.7704, Adjusted R-squared:  0.7704
## F-statistic: 1.678e+06 on 2 and 999997 DF,  p-value: < 2.2e-16
```

Then, do the following:

a) Using the true model, demonstrate that the coefficient for your treatment variable follows the central limit theorem. That is, demonstrate that the coefficient's sampling distribution is approximately normal.

```r
# Sample sizes:
n_sample <- c(60, 200, 600, 2000, 5000, 10000)
# Number of simulations per sample size
n_simulation <- 1000

# Total regressions 6000
results <- data.frame(
  s_size = numeric(6000),
  sim         = integer(6000),
  Beta0       = numeric(6000),
  Beta1       = numeric(6000),
  Beta2       = numeric(6000)
)

# Population observations
N <- nrow(population)

# Start filling from row 1
row <- 1

# Loop over each sample size
for (n in n_sample) {
# Repeat the regression 1000 times
  for (i in 1:n_simulation) {

# Randomly pick observations from the population
    sample_in <- sample(1:N, size = n, replace = FALSE)
    samp <- population[sample_in, ]

# Fit linear model on the sample
    s_model <- lm(y ~ x + z, data = samp)
    coefficients <- coef(s_model)

    # Results saved
    results$s_size[row] <- n
    results$sim[row]         <- i
    results$Beta0[row]       <- coefficients["(Intercept)"]
    results$Beta1[row]       <- coefficients["x"]
    results$Beta2[row]       <- coefficients["z"]

    # Move to next row
    row <- row + 1
  }
}

# Convert sample size to categorical variable for plotting
results$s_size <- as.factor(results$s_size)

# Visualize the sampling distribution of beta1 (coefficient on x)
ggplot(results, aes(x = Beta1, color = s_size)) +
  geom_density() +
  labs(
    x     = "Beta1 estimates (coefficient on x)",
    y     = "Density",
    color = "sample sizes",
    title = "Sampling distribution of Beta1 across sample sizes"
  ) +
  theme_minimal()
```



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

```
# Creating new results dataframe
results <- data.frame(
  s_size = numeric(6000),
  sim = integer(6000),
  Beta0=numeric(6000),
  Beta1=numeric(6000),
  Beta2=numeric(6000)
)

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

#create for loop for each sample size
for (n in n_sample) {
  #creating for loop to repeat regression 1000 times
```

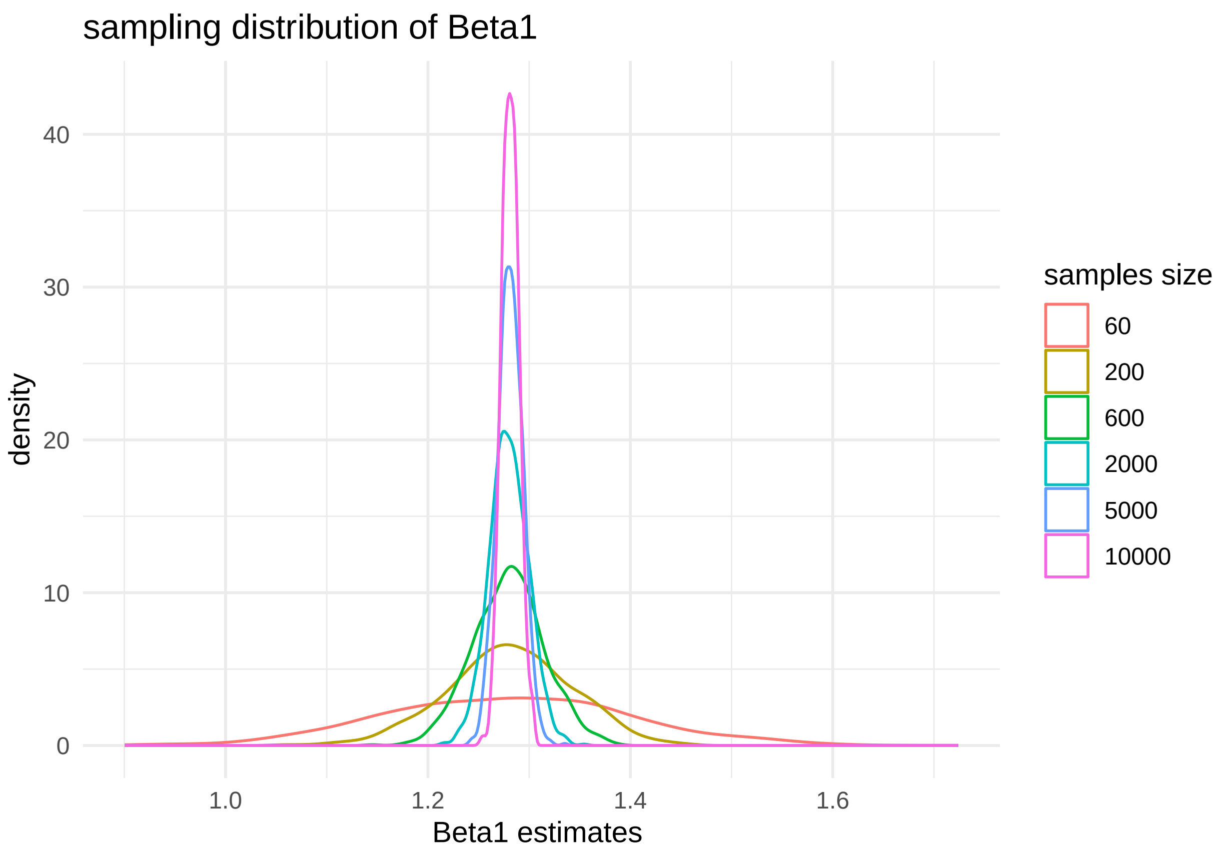
```r
  for (i in 1:n_simulation){
    #randomly pick observations to pick from population
    sample_in1 <- sample(1:N, size = n, replace = TRUE)
    sample_1 <- population[sample_in1,]

    #fit linear model for each sample
    model <- lm(y ~ x + z, data=sample_1)
    coefficients <- coef(model)

    #saving results
    results$s_size[row] <- n #sample size
    results$sim[row] <- i #regression iteration
    results$Beta0[row] <- coefficients["(Intercept)"]
    results$Beta1[row] <- coefficients["x"]
    results$Beta2[row] <- coefficients["z"]

    row <- row + 1
  }
}

print(sd(results$Beta1))
```

```
## [1] 0.05677106
```

c) Fit a model that omits the confounding variable. Repeat part (a) for this new model and plot the sampling distribution of the treatment variable's coefficient. How do your results differ? What does this imply about statistical tests based on a coefficient's sampling distribution?

If I omit a confounding variable, then the distribution of beta could move away from the true value so the sampling distribution will be biased. So, I won't be able to trust the the statistical tests.

```r
#Create different results data frame; 6000

results<- data.frame(
  s_size=numeric(6000),
  sim=integer(6000),
  Beta0=numeric(6000),
  Beta1=numeric(6000),
  Beta2=numeric(6000)
)

N <- nrow(population)
row <- 1 #start at row 1

#create for loop for sample size
for (n in n_sample) {
  #create for loop to repeat regression 1000x
  for(i in 1:n_simulation){
    #randomly pick which observations drawn from  population
    sample_1 <- sample(1:N, size = n, replace=FALSE)
    sample_2 <- population[sample_1,]

    #fit linear model for each sample
    model <- lm(y~x, data = sample_2)
    coefficients <- coef(model)
    #saving results
    results$s_size[row] <- n #sample size
    results$sim[row] <- i #regression iterations
    results$Beta0[row] <-coefficients["(Intercept)"] #saving coefficients
    results$Beta1[row] <- coefficients["x"]
    results$Beta2[row] <- coefficients["z"]

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

results$s_size <- as.factor(results$s_size) #converting sample size into a categorical variable

#creating a plot to visualize the sampling distribution for each sample size

ggplot(results, aes(x=Beta1, color=s_size)) +
  geom_density() +
  labs(
    x="Beta1 estimates ",
    y="density",
    color="samples size",
    title="sampling distribution of Beta1"
  ) +
  theme_minimal()
```



PART 2: DATA ANALYSIS

Use any data set. a) Conduct a hypothesis test for a difference in means. You decide what the hypotheses are, whether you use a t-test or a z-test, and what the level of significance is. Explain your decisions, and interpret your results both substantively and statistically.

I hypothesize that if class size is small, then math scores will increase. I did a Welch two sample t-test because I don't need to assume a certain distribution. The results show that the difference in is 2.74 standard deviations away from 0. This shows that small classes have an effect on math scores.

```r
star<-read.csv("STAR.csv")
```

```r
#Create treatment
star$treat <- ifelse(star$classtype == "small", 1, 0)

#Dataset for treatment and control
treatment <- star[star$treat == 1, ]
control <- star[star$treat == 0, ]

#difference in means: WELCH
two_tailed <- t.test(treatment$math, control$math, mu = 0, alternative = "two.sided", conf.level = 0.95
print(two_tailed)
```

```
##
##  Welch Two Sample t-test
##
## data:  treatment$math and control$math
## t = 2.7404, df = 1219.3, p-value = 0.006227
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   1.701545 10.278265
## sample estimates:
## mean of x mean of y
##  634.8274  628.8374
```

b) Using the same data, fit a linear model. Interpret the coefficient, standard error, t-value, and p-value.

The treatment coefficient tells us that students in smaller classes scored 5.99 points higher than students than regular class students. The standard error is 2.178 and this is our uncertainty for our treatment group. The t-value is 2.75 which is the distance of the standard errors from 0. The p-value is 0.00604 which is less than .05 so it shows that small classes can increase math scores.

```r
# Fit linear model. Predicts math scores from class type
lm_model <- lm(math ~ treat, data = star)

summary(lm_model)
```

```
##
## Call:
## lm(formula = math ~ treat, data = star)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -119.827  -27.585   -0.827   26.163  145.163
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  628.837      1.476  426.09  < 2e-16 ***
## treat          5.990      2.178    2.75  0.00604 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 38.74 on 1272 degrees of freedom
## Multiple R-squared:  0.005911,   Adjusted R-squared:  0.00513
## F-statistic: 7.564 on 1 and 1272 DF,  p-value: 0.006039
```