Homework 3 Econometrics

Samuel Fraley Daniel Campos Elvis Casco

1 Question 1

Consider assumptions [A1]-[A4] presented in class slide 1(3). If we wanted to present these same assumptions for random samples, that is, when $\{(x_i, y_i)\}$ are i.i.d (independently and identically distributed across observations), how would you write these 4 assumptions? Provide the expressions and briefly justify.

Original Assumptions (fixed-X sampling):

$$[A1]: \quad y_i = \beta_1 + \beta_2 x_{i2} + \cdots + \beta_K x_{iK} + \epsilon_i = x_i'\beta + \epsilon_i$$

$$[A2]: \quad E(\epsilon_i|X) = 0$$

$$[A3]: \quad \text{Var}(\epsilon_i|X) = \sigma^2$$

$$[A4]: \quad \text{Cov}(\epsilon_i, \epsilon_j|X) = 0 \quad \forall i \neq j$$

1.1 Answer

To adapt assumptions [A1]–[A4] for the case where observations $\{(x_i, y_i)\}$ are independently and identically distributed (*i.i.d.*), we shift from conditioning on the full design matrix X to conditioning on the individual regressor vector x_i. Here's how each assumption would be rewritten:

Assumption [A1]: Linearity

Original (fixed-X sampling):

$$y_i = x_i'\beta + \epsilon_i$$

i.i.d. Version:

$$y_i = x_i'\beta + \epsilon_i$$

Justification: This assumption remains unchanged. It states that the outcome y_i is a linear function of the covariates x_i plus an error term ϵ_i. Linearity is a structural assumption and does not depend on whether the data are *i.i.d.*

Assumption [A2]: Zero Conditional Mean (Strict Exogeneity)

Original (fixed-X sampling):

$$E(\epsilon_i|X) = 0 \quad \forall i = 1, \dots, n$$

i.i.d. Version:

$$E(\epsilon_i|x_i) = 0 \quad \forall i = 1, \dots, n$$

Justification: Under *i.i.d.* sampling, each observation (x_i, y_i) is independent of the others. Therefore, we condition only on the individual covariates x_i, not the entire matrix X. This assumption ensures that the regressors are exogenous and that the error term has no systematic relationship with the predictors.

Assumption [A3]: Conditional Homoskedasticity

Original (fixed-X sampling):

$$\mathrm{Var}(\epsilon_i|X) = \sigma^2 \quad \forall i = 1, \dots, n$$

i.i.d. Version:

$$\mathrm{Var}(\epsilon_i|x_i) = \sigma^2 \quad \forall i = 1, \dots, n$$

Justification: Again, we condition on x_i rather than the full matrix X. This assumption implies that the variance of the error term is constant across observations, regardless of the values of the covariates.

Assumption [A4]: No Autocorrelation

Original (fixed-X sampling):

$$\mathrm{Cov}(\epsilon_i, \epsilon_j|X) = 0 \quad \forall i \neq j$$

i.i.d. Version:

$$\mathrm{Cov}(\epsilon_i, \epsilon_j) = 0 \quad \forall i \neq j$$

Justification: Under *i.i.d.* sampling, the errors are automatically uncorrelated across observations. Since the observations are independent, we do not need to condition on X to assert that the errors are uncorrelated. In fact, under *i.i.d.* sampling, the errors are independent (a stronger condition than uncorrelated), which implies $\text{Cov}(\epsilon_i, \epsilon_j) = 0$ for all $i \neq j$.

2. Under linearity and strict exogeneity assumptions we saw that OLS estimator of the parameters of a linear regression model, $\hat{\beta}$, is conditionally unbiased:

$$(A) \qquad E\left(\hat{\beta}_k | X\right) = \beta_k \quad \forall k = 1, \dots, K$$

Additionally, under *Gauss-Markov* assumptions, we saw that the OLS estimator is the one with the smallest conditional variance among the class of all linear unbiased estimators:

$$(B) \qquad var(\hat{\beta}_k | X) \leq var(\tilde{\beta}_k | X) \quad \forall k = 1, \dots, K$$

(a)

(b) Departing from result (A), prove that OLS is also unconditionally unbiased.

Answer:

To prove that the OLS estimator $\hat{\beta}_k$ is unconditionally unbiased, we start from the conditional unbiasedness result:

$E(\hat{\beta}_k \mid X) = \beta_k \quad \forall k = 1, \dots, K$

Now, we want to show:

$E(\hat{\beta}_k) = \beta_k$

Step-by-step Proof:

We use the law of iterated expectations, which states:

$E(\hat{\beta}_k) = E\left[E(\hat{\beta}_k \mid X)\right]$

Substitute the conditional expectation from (A):

$E(\hat{\beta}_k) = E\left[\beta_k\right]$

Since β_k is a fixed parameter (not random), its expectation is just itself:

$E(\beta_k) = \beta_k$

Therefore:

$E(\hat{\beta}_k) = \beta_k$

(ii) In a sentence describe what this property is telling us.

Answer:

This property tells us that, on average across all possible samples, the OLS estimator correctly targets the true value of the parameter—it doesn't systematically overestimate or underestimate it.

(b) Departing from result (B), prove that this inequality also holds for the unconditional version. That is, prove:

$$var(\hat{\beta}_k) \leq var(\tilde{\beta}_k) \quad \forall k$$

Hint: Use property $var(Z) = var[E(Z|W)] + E[var(Z|W)]$.

Answer:

To prove the unconditional version of the Gauss-Markov result:

$\text{Var}(\hat{\beta}_k) \leq \text{Var}(\tilde{\beta}_k) \quad \forall k$

we'll use the law of total variance, which states:

$\text{Var}(Z) = \text{Var}\left[E(Z|W)\right] + E\left[\text{Var}(Z|W)\right]$

Step-by-step Proof

Let's apply this to both estimators $\hat{\beta}_k$ and $\tilde{\beta}_k$, conditioning on the regressor matrix X:

1. Apply total variance to $\hat{\beta}_k$:

$\text{Var}(\hat{\beta}_k) = \text{Var}\left[E(\hat{\beta}_k|X)\right] + E\left[\text{Var}(\hat{\beta}_k|X)\right]$

From result (A), we know:

$E(\hat{\beta}_k|X) = \beta_k \quad \Rightarrow \quad \text{Var}\left[E(\hat{\beta}_k|X)\right] = \text{Var}(\beta_k) = 0$

So:

$\text{Var}(\hat{\beta}_k) = E\left[\text{Var}(\hat{\beta}_k|X)\right]$

2. Apply total variance to any other linear unbiased estimator $\tilde{\beta}_k$:

$\text{Var}(\tilde{\beta}_k) = \text{Var}\left[E(\tilde{\beta}_k|X)\right] + E\left[\text{Var}(\tilde{\beta}_k|X)\right]$

Since $\tilde{\beta}_k$ is also unbiased:

$E(\tilde{\beta}_k|X) = \beta_k \quad \Rightarrow \quad \text{Var}\left[E(\tilde{\beta}_k|X)\right] = 0$

So: $\text{Var}(\tilde{\beta}_k) = E\left[\text{Var}(\tilde{\beta}_k|X)\right]$

3. Use Gauss-Markov result (B):

$$\text{Var}(\hat{\beta}_k|X) \leq \text{Var}(\tilde{\beta}_k|X) \quad \Rightarrow \quad E\left[\text{Var}(\hat{\beta}_k|X)\right] \leq E\left[\text{Var}(\tilde{\beta}_k|X)\right]$$

Thus:

$$\text{Var}(\hat{\beta}_k) \leq \text{Var}(\tilde{\beta}_k)$$

3. Consider performing the following test:

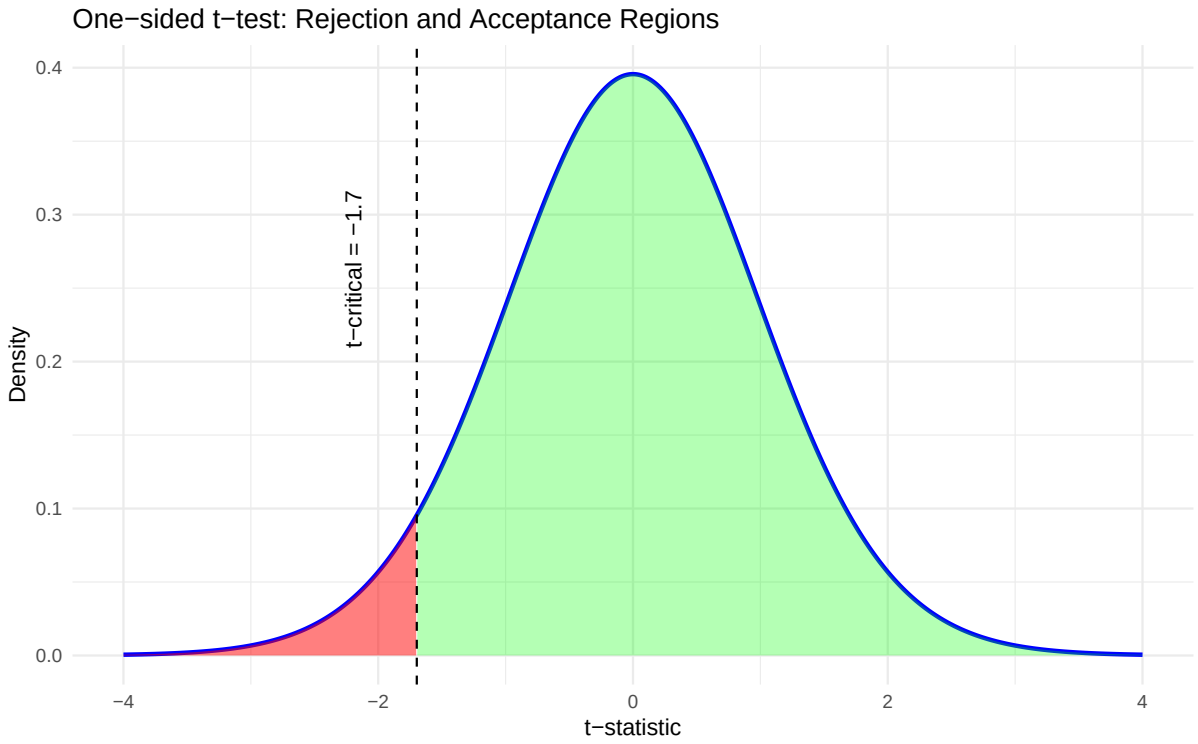
$$H_0 : \beta_2 = 0 \quad \text{versus} H_1 : \beta_2 < 0,$$

using the t-test statistic. This test is called one-sided test since its rejection region is all in one of the tails of the distribution.

(a) Draw the acceptance and rejection regions for this test using $\alpha = 5$.

```
library(ggplot2)

alpha <- 0.05
df <- 30
t_crit <- qt(alpha, df)
x <- seq(-4, 4, length.out = 1000)
y <- dt(x, df)
data <- data.frame(x = x, y = y)
ggplot(data, aes(x, y)) +
  geom_line(color = "blue", size = 1) +
  geom_area(data = subset(data, x <= t_crit), aes(x, y), fill = "red", alpha = 0.5) +
  geom_area(data = subset(data, x > t_crit), aes(x, y), fill = "green", alpha = 0.3) +
  geom_vline(xintercept = t_crit, linetype = "dashed", color = "black") +
  annotate("text", x = t_crit - 0.5, y = max(y)*0.8,
           label = paste("t-critical =", round(t_crit, 2)),
           angle = 90, hjust = 1) +
  labs(title = "One-sided t-test: Rejection and Acceptance Regions",
       x = "t-statistic", y = "Density") +
  theme_minimal()
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```



(b) Explain the intuition behind the location of the critical region.

Answer:

The intuition behind the location of the critical region in a one-sided t-test like:

$H_0 : \beta_2 = 0 \quad \text{vs.} \quad H_1 : \beta_2 < 0$

is rooted in the direction of the alternative hypothesis.

Why it's on the left tail

- The alternative hypothesis $H_1 : \beta_2 < 0$ suggests we're testing whether the parameter is significantly less than zero.
- The t-statistic measures how far the estimated coefficient $\hat{\beta}_2$ is from zero, in standard error units.
- If $\hat{\beta}_2$ is much smaller than zero, the t-statistic will be strongly negative.
- So, we place the rejection region in the left tail of the t-distribution—where extreme negative values live.

What the critical value means

- The critical value t_α is the threshold below which we reject H_0.
- For $= 0.05$, we're saying: "If the t-statistic is in the lowest 5% of its expected range under H_0, that's too unlikely to be due to chance."

Acceptance vs. Rejection

- If the t-statistic is greater than the critical value (closer to zero or positive), we fail to reject H_0.
- If it's less than the critical value (deep in the left tail), we reject H_0 in favor of H_1.

4. Consider the code included in file Assig3Q4.R or file Assig3Q4.py. Both codes are reproduced at the end of the assignment, after the instructions.

(a) Provide the expression of the dgp behind any generated sample. Be careful with the notation.

Answer:

The data generating process (DGP) behind each simulated sample in the code is given by the following linear model:

$y_i = \beta_1 + \beta_2 x_{i2} + \epsilon_i$

where:

- $\beta_1 = 10$
- $\beta_2 = 5$
- $x_{i2} \sim \text{Uniform}(0, 20)$
- $\epsilon_i \sim \mathcal{N}(0, 6^2)$, independently across $i = 1, \ldots, 50$

So the full expression becomes:

$y_i = 10 + 5x_{i2} + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, 36)$

This DGP is used to simulate 100 independent samples of size 50, each time generating new y_i values based on the same x_{i2} vector and fresh random noise ϵ_i.

(b)

(c) How many samples does the code file generate?

Answer:

The code generates 100 samples. This is controlled by the line:

```
M <- 100
```

Each iteration of the for loop simulates one sample of size 50, fits a linear regression model, and computes a confidence interval for β_2. So in total, it produces 100 confidence intervals—one for each simulated sample.

(ii) What do all the samples have in common?

Answer:

All the samples in the code share the same underlying data generating process (DGP) and same predictor values. Specifically:

- They all use the same vector of covariates: $x_{i2} \sim \text{Uniform}(0, 20)$
- This vector is generated once and reused across all 100 samples.
- They all follow the same linear model: $y_i = 10 + 5x_{i2} + \epsilon_i - where \epsilon_i \sim \mathcal{N}(0, 6^2)$ is independently drawn for each sample.
- Each sample has the same size: 50 observations.

So while the noise term ϵ_i varies across samples (introducing randomness), the structure of the model and the covariates remain constant, allowing us to assess how often the confidence intervals correctly capture the true value $\beta_2 = 5$.

(iii) Describe what the R code from line 1 to line 16 (or from line 6 to 30 in Python) does. Be specific.

Answer:

```
set.seed(1010)
M <- 100
lower <- numeric(M)
upper <- numeric(M)
x2 <- runif(50, 0, 20)
```

- `set.seed(1010)`: Ensures reproducibility of random numbers.
- `M <- 100`: Sets the number of simulated samples to 100.
- `lower` and `upper`: Initialize vectors to store the lower and upper bounds of confidence intervals for each sample.
- `x2`: Generates a fixed vector of 50 random values from a uniform distribution between 0 and 20. This vector is reused across all samples.

```
for(i in 1:M) {
  y <- 10 + 5 * x2 + rnorm(50, mean = 0, sd = 6)
  model <- lm(y ~ x2)
  b2 <- coef(model)[[2]]
  varb2 <- vcov(model)[2, 2]
  se2 <- sqrt(varb2)
  lower[i] <- b2 - qt(0.025, 48, lower.tail = FALSE) * se2
  upper[i] <- b2 + qt(0.025, 48, lower.tail = FALSE) * se2
}
```

- The loop runs 100 times, simulating 100 samples.

- Each iteration:
- Generates a new response vector y using the DGP: $y_i = 10 + 5x_{i2} + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, 6^2)$
- Fits a linear regression model: `lm(y ~ x2)`
- Extracts the estimated slope coefficient `b2` for `x_2`
- Computes the variance and standard error of `b2`
- Constructs a 95% confidence interval for β_2 using the critical value from the t-distribution with 48 degrees of freedom (since `n = 50`, and we estimate 2 parameters)

By the end of line 16:

- You have 100 confidence intervals stored in lower and upper.
- These intervals reflect the uncertainty around the estimate of $\beta_2 = 5$ across repeated samples.

(c) Describe what does R code from line 18 to line 23 (or from line 32 to 39 in Python) do? Be specific.

Answer:

```
CIs <- cbind(lower, upper)

IDg <- which((CIs[1:M, 1] <= 5 & 5 <= CIs[1:M, 2]))
length(IDg)

IDb <- which(!(CIs[1:M, 1] <= 5 & 5 <= CIs[1:M, 2]))

ratiog <- (length(IDg) / M) * 100
ratiog
```

What Each Line Does - `CIs <- cbind(lower, upper)`: Combines the lower and upper bounds of the 100 confidence intervals into a matrix `CIs`, where each row represents one interval. - `IDg <- which((CIs[1:M, 1] <= 5 & 5 <= CIs[1:M, 2]))`: Identifies the indices of intervals that contain the true value $\beta_2 = 5$. These are the "good" intervals. - `length(IDg)`: Counts how many intervals successfully captured the true value. - `IDb <- which(!(CIs[1:M, 1] <= 5 & 5 <= CIs[1:M, 2]))`: Identifies the indices of intervals that do not contain $\beta_2 = 5$. These are the "bad" intervals. - `ratiog <- (length(IDg) / M) * 100`: Calculates the coverage rate—the percentage of intervals that contain the true value. - `ratiog`: Displays the coverage rate, which should be close to 95% if the confidence intervals are correctly calibrated.

This block of code checks how often the simulated confidence intervals actually include the true parameter value ($\beta_2 = 5$). It's a practical way to assess the reliability of the interval estimation procedure.

(d) Describe what does R code from line 25 to line 40 (or from line 41 to 56 in Python) do? Be specific.

Answer:

The R code from line 25 to line 40 creates a visualization of the 100 confidence intervals for β_2, highlighting which intervals contain the true value (5) and which do not. Here's a detailed breakdown:

```
plot(0,
     xlim = c(4, 6),
     ylim = c(1, 100),
     ylab = "Samples",
     xlab = expression(beta[2]),
     main = "Confidence Intervals")
abline(v = 5, lty = 2)
```

- Initializes an empty plot with: axis from 4 to 6 (range of confidence intervals); y-axis from 1 to 100 (one row per sample)
- Labels the axes and adds a title.
- Draws a vertical dashed line at $\beta_2 = 5$, the true value, to visually assess which intervals include it.

```
colors <- rep(gray(0.6), 100)
colors[IDb] <- "red"
```

Creates a vector of colors:

- Default: gray for intervals that contain the true value.
- Red for intervals that miss the true value (those in IDb).

```
for(j in 1:100) {
  lines(c(CIs[j, 1], CIs[j, 2]), c(j, j),
        col = colors[j],
        lwd = 2)
}
```

- Loops through all 100 samples.
- For each sample j, draws a horizontal line from the lower to upper bound of the confidence interval.
- The line is placed at height j and colored: Gray if it contains 5; Red if it does not

This plot provides a clear visual summary of:

- How many intervals captured the true $\beta_2 = 5$
- Which intervals failed (in red)
- The coverage performance of the confidence interval procedure

(e) Run the entire code file (your choice, ideally try both). Include the value of the variable 'ratiog' and the plot. Surprised by the value of 'ratiog'? Rigorously comment.

Answer:

```
set.seed(1010)
M <- 100
lower <- numeric(M)
upper <- numeric(M)
x2 <- runif(50,0,20)

for(i in 1:M) {
  y <- 10+5*x2+rnorm(50, mean = 0, sd = 6)
  model<-lm(y~x2)
  b2<-coef(model)[[2]]
  varb2 <- vcov(model)[2, 2]
  se2<-sqrt(  varb2)
  lower[i] <- b2 - qt(0.025,48, lower.tail = F) * se2
  upper[i] <- b2 + qt(0.025,48, lower.tail = F) * se2
}
CIs <- cbind(lower, upper)

IDg <- which((CIs[1:M, 1] <= 5 & 5 <= CIs[1:M, 2]))
length(IDg)
```

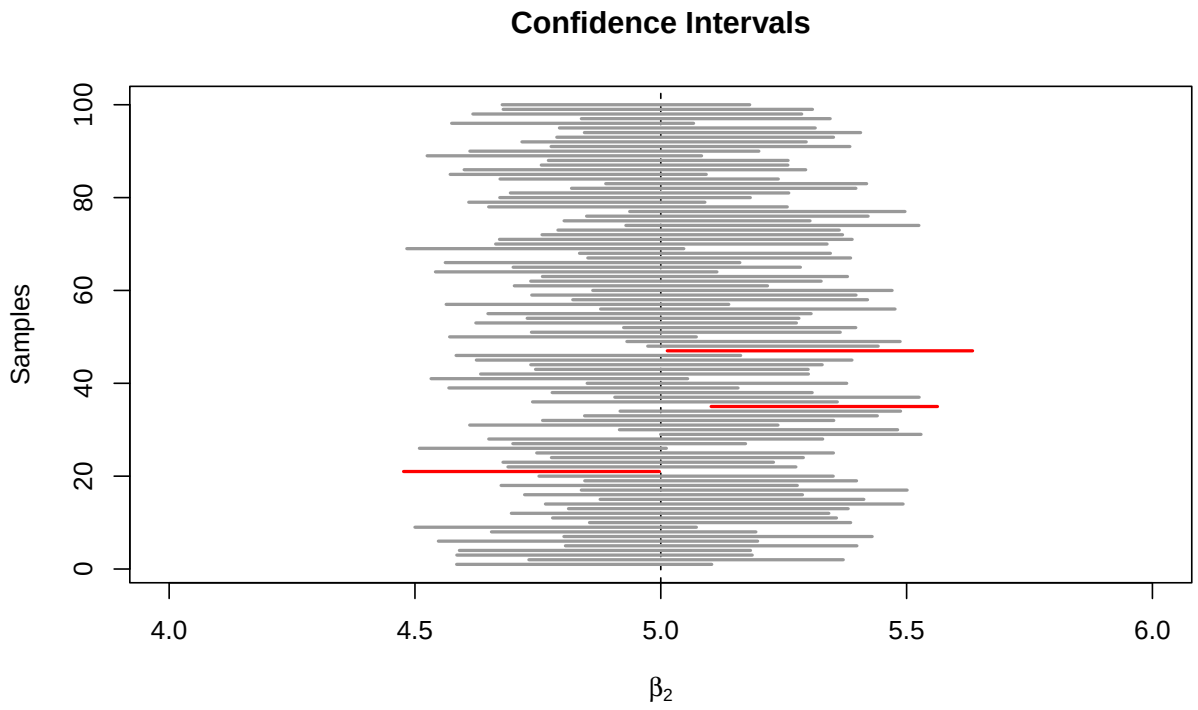
```
[1] 97
```

```
IDb <- which(!(CIs[1:M, 1] <= 5 & 5 <= CIs[1:M, 2]))

ratiog<-(length(IDg)/M)*100
ratiog
```

```
[1] 97
```

```
plot(0,
     xlim = c(4, 6),
     ylim = c(1, 100),
```

```
     ylab = "Samples",
     xlab = expression(beta[2]),
     main = "Confidence Intervals")
abline(v = 5, lty = 2)

colors <- rep(gray(0.6), 100)
colors[IDb] <- "red"

for(j in 1:100) {
  lines(c(CIs[j, 1], CIs[j, 2]), c(j, j),
        col = colors[j],
        lwd = 2)
}
```




```
print(ratiog)
```

```
[1] 97
```

Surprised? Not really:

- The confidence level is 95%, meaning we expect on average 95 out of 100 intervals to contain the true parameter.

- The observed coverage of 97% is well within the expected statistical fluctuation due to random sampling.
- This is a classic demonstration of the frequentist interpretation of confidence intervals: over many repetitions, about 95% of intervals should contain the true value.

Rigorous Commentary

- Statistical Validity: The simulation confirms that the confidence interval procedure is well-calibrated. A 97% coverage rate is consistent with the nominal 95% level.
- Random Variation: The slight deviation from 95% is due to sampling variability.
- Model Assumptions: The linear model assumes homoscedasticity and normal errors. The simulation uses `rnorm` with `sd = 6`, which is reasonable and supports valid inference.
- Inference Robustness: The use of `vcov(model)` ensures that standard errors are computed correctly, and the use of `qt(0.025, 48, lower.tail = F)` reflects the correct critical value for a 95% CI with 48 degrees of freedom.

(f) What does the plot illustrate? Rigorously comment.

Answer:

The plot illustrates the empirical coverage performance of 95% confidence intervals for the slope parameter β_2 in a simple linear regression model, based on repeated sampling.

What the Plot Shows

- Each horizontal line represents a confidence interval for β_2 from one of 100 simulated samples.
- The dashed vertical line at $\beta_2 = 5$ marks the true value used in the data generating process.
- Gray lines indicate intervals that successfully contain the true value.
- Red lines indicate intervals that fail to contain the true value.

Rigorously Interpreting the Plot

- The plot visually confirms that most intervals include the true value, consistent with the nominal 95% confidence level.
- The presence of 5 red intervals (out of 100) aligns with the theoretical expectation that, under correct model assumptions, approximately 5% of intervals will miss the true value purely due to sampling variability.
- This validates the frequentist interpretation of confidence intervals: If we repeated the experiment many times, about 95% of the intervals would contain the true parameter.

The reliability of the intervals -and the accuracy of the coverage rate—depends on the following assumptions being satisfied in the simulation:

- Linearity: The model $y_i = 10 + 5x_{i2} + \epsilon_i$ is correctly specified.

- Independence: Observations are independent across samples.
- Homoskedasticity: Constant variance of errors ($\epsilon_i \sim \mathcal{N}(0, 36)$).
- Normality: Errors are normally distributed, justifying the use of the t-distribution for interval construction.

The plot is a compelling visual confirmation that the confidence interval procedure is well-calibrated. It demonstrates that the intervals behave as expected under repeated sampling, with a coverage rate very close to the nominal level of 95%.

(g) If you increased the value of M at the tope of the code file, say from 100 to 10,000, how would you expect the value of 'ratiog' to change? Rigorously argue.

Answer:

If you increase M from 100 to 10,000, you're dramatically increasing the number of simulated samples — and here's what that means for ratiog, both intuitively and rigorously:

What Is ratiog?

ratiog is the percentage of confidence intervals (CIs) that contain the true slope value $\beta_2 = 5$ across M simulations. Each simulation fits a linear model and computes a 95% CI for the slope.

So:

```
ratiog = (number of intervals containing 5 / M) × 100
```

What Happens When M = 100?

With 100 simulations, you get a rough estimate of the true coverage probability. Due to random variation, you might see values like 94%, 97%, or even 92% — all within the expected range for a 95% CI.

What Happens When M = 10,000?

Now you're running 10,000 simulations, which means:

- Law of Large Numbers kicks in: the empirical coverage (ratiog) will converge to the theoretical coverage of 95%.
- Random fluctuations shrink: the standard error of the estimate decreases.

The standard error (SE) of a proportion is:

$SE = \sqrt{\frac{p(1-p)}{M}}$

For a 95% CI (p = 0.95): - When M = 100: - When M = 10,000:

So with M = 10,000, you'd expect ratiog to fall within a very tight band around 95%, say between 94.5% and 95.5%.

Rigorously Interpreted

- Theoretical expectation: The confidence interval procedure is designed to capture the true parameter 95% of the time.
- Empirical convergence: As M increases, the observed proportion (ratiog) converges to this theoretical value.
- Statistical consistency: The simulation becomes more reliable and less sensitive to random noise.

Final Takeaway

If you increase M to 10,000, you should expect ratiog to be very close to 95%, with minimal deviation — a strong confirmation that the CI construction method is statistically sound.

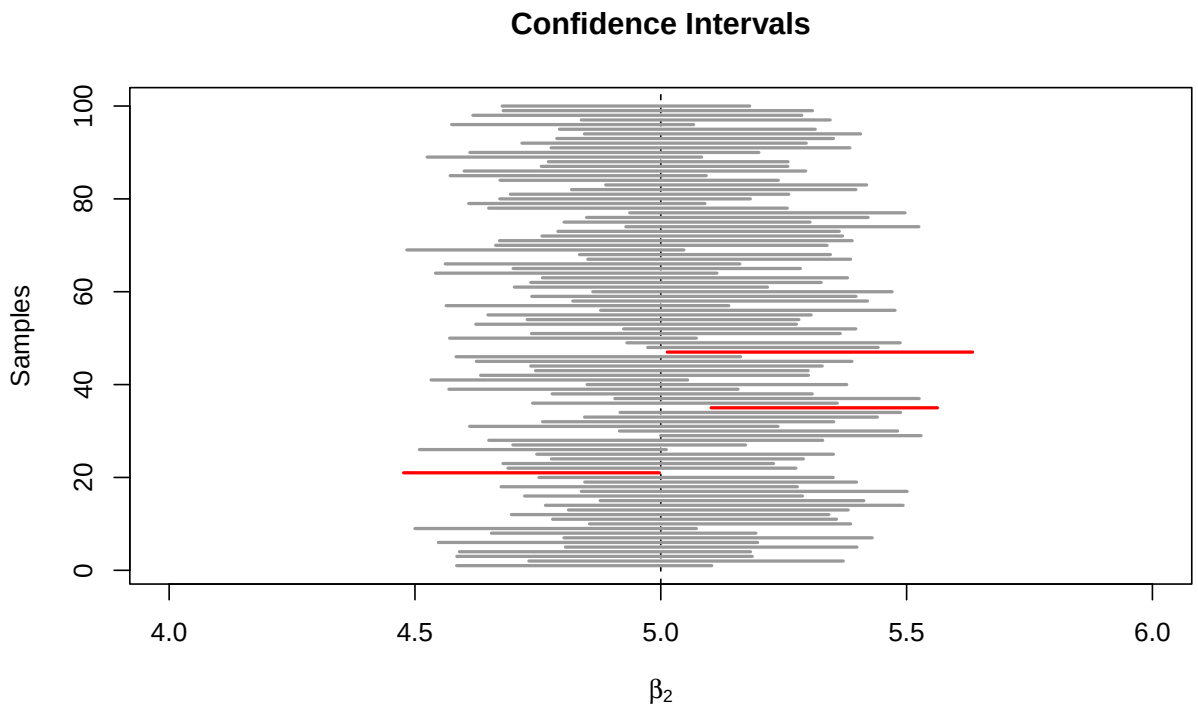
```r
set.seed(1010)
M <- 10000
lower <- numeric(M)
upper <- numeric(M)
x2 <- runif(50,0,20)

for(i in 1:M) {
  y <- 10+5*x2+rnorm(50, mean = 0, sd = 6)
  model<-lm(y~x2)
  b2<-coef(model)[[2]]
  varb2 <- vcov(model)[2, 2]
  se2<-sqrt(  varb2)
  lower[i] <- b2 - qt(0.025,48, lower.tail = F) * se2
  upper[i] <- b2 + qt(0.025,48, lower.tail = F) * se2
}
CIs <- cbind(lower, upper)

IDg <- which((CIs[1:M, 1] <= 5 & 5 <= CIs[1:M, 2]))
length(IDg)
```

```
[1] 9510
```

```r
IDb <- which(!(CIs[1:M, 1] <= 5 & 5 <= CIs[1:M, 2]))

ratiog<-(length(IDg)/M)*100
ratiog
```

```
[1] 95.1
```

```r
plot(0,
     xlim = c(4, 6),
     ylim = c(1, 100),
     ylab = "Samples",
     xlab = expression(beta[2]),
     main = "Confidence Intervals")
abline(v = 5, lty = 2)

colors <- rep(gray(0.6), 100)
colors[IDb] <- "red"

for(j in 1:100) {
  lines(c(CIs[j, 1], CIs[j, 2]), c(j, j),
        col = colors[j],
        lwd = 2)
}
```




```r
print(ratiog)
```

```
[1] 95.1
```

(h) Finally, for M = 10, 000, if we replaced value 0.025 in R code line 13 and 14 by 0.005 (or replace value 0.975 by 0.995 in line 26 in Python code file), what would you expect the value of 'Ratio' to be? Rigorously argue. And the plot to change?

Answer:

```
set.seed(1010)
M <- 10000
lower <- numeric(M)
upper <- numeric(M)
x2 <- runif(50,0,20)

for(i in 1:M) {
  y <- 10+5*x2+rnorm(50, mean = 0, sd = 6)
  model<-lm(y~x2)
  b2<-coef(model)[[2]]
  varb2 <- vcov(model)[2, 2]
  se2<-sqrt(  varb2)
  lower[i] <- b2 - qt(0.005,48, lower.tail = F) * se2
  upper[i] <- b2 + qt(0.005,48, lower.tail = F) * se2
}
CIs <- cbind(lower, upper)

IDg <- which((CIs[1:M, 1] <= 5 & 5 <= CIs[1:M, 2]))
length(IDg)
```

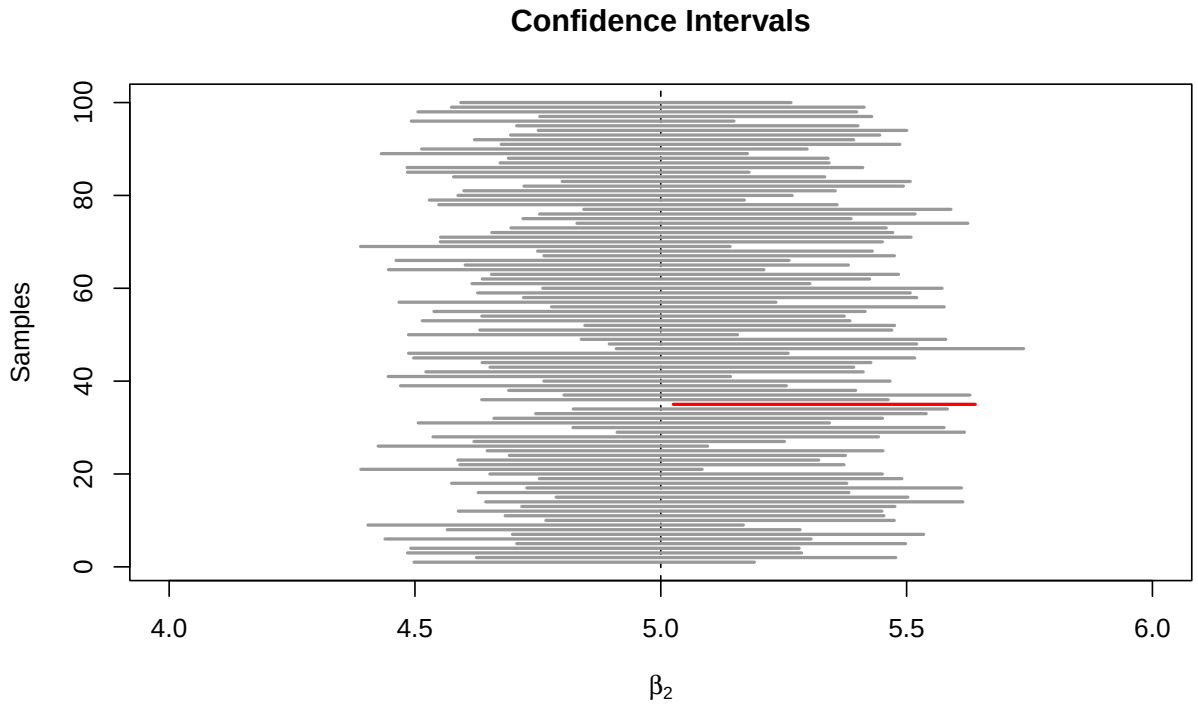
```
[1] 9893
```

```
IDb <- which(!(CIs[1:M, 1] <= 5 & 5 <= CIs[1:M, 2]))

ratiog<-(length(IDg)/M)*100
ratiog
```

```
[1] 98.93
```

```
plot(0,
     xlim = c(4, 6),
     ylim = c(1, 100),
     ylab = "Samples",
     xlab = expression(beta[2]),
     main = "Confidence Intervals")
```

```r
abline(v = 5, lty = 2)

colors <- rep(gray(0.6), 100)
colors[IDb] <- "red"

for(j in 1:100) {
  lines(c(CIs[j, 1], CIs[j, 2]), c(j, j),
        col = colors[j],
        lwd = 2)
}
```




```r
print(ratiog)
```

```
[1] 98.93
```

In the original code, the confidence interval is constructed as:

$b2 \pm qt(0.025, df = 48, lower.tail = FALSE) \times se2$

This uses the 97.5th percentile of the t-distribution to create a 95% confidence interval (since 2.5% is in each tail). If you replace with , you're now using the 99.5th percentile, which gives you a 99% confidence interval.

Let's define:

- "ratiog = 95" = proportion of intervals that contain the true value when using 95% confidence intervals
- "ratiog = 99"= same, but for 99% confidence intervals

Theoretical Expectation

- By definition, a 99% confidence interval is wider than a 95% interval.
- Therefore, it is more likely to contain the true parameter.

So with M = 10,000

- At 95% confidence, you expect ~9,500 intervals to contain the true value.
- At 99% confidence, you expect ~9,900 intervals to contain the true value.
- So should increase from ~95% to ~99%.

This is a direct consequence of the confidence level: higher confidence → wider intervals → higher coverage.

How Will the Plot Change?

The plot shows horizontal confidence intervals for each simulation, with:

- Gray lines: intervals that contain the true value ($\beta_2 = 5$)
- Red lines: intervals that miss the true value

With 99% Confidence:

- The intervals will be visibly wider.
- Fewer red lines: because more intervals will now include 5.
- The vertical dashed line at 5 will intersect more intervals.

So visually, the plot will look more "forgiving" - more intervals will span the true value, and the red lines will nearly disappear.

Rigorously Interpreted

- Statistical trade-off: Increasing the confidence level improves coverage but at the cost of wider intervals, which may be less precise.
- Frequentist principle: Over many repetitions, a 99% CI will contain the true parameter 99% of the time — and your simulation will reflect that.
- Practical implication: If you're okay with wider intervals, you gain more certainty that they contain the true value.

Final Takeaway

If you increase the confidence level to 99% by changing to , then:

- "ratiog" will increase to approximately 99%
- The plot will show wider intervals
- Red lines will nearly vanish, confirming stronger coverage

(i) Finally, what do you think the purpose of this simulation exercise is? That is, what is it trying to illustrate? Be specific.

Answer:

This simulation exercise is designed to visually and empirically demonstrate the frequentist interpretation of confidence intervals — specifically, how often a confidence interval constructed at a given confidence level (e.g. 95%) actually contains the true parameter value when the experiment is repeated many times.

Core Purpose: Understanding Confidence Interval Coverage The simulation illustrates the following key statistical concepts:

1. Frequentist Coverage Probability

- A 95% confidence interval does not mean there's a 95% chance the true parameter lies within a single interval.
- Instead, it means that if we repeated the experiment many times, about 95% of the constructed intervals would contain the true parameter.
- The simulation makes this abstract idea concrete by repeating the experiment M times and calculating the proportion of intervals that actually include the true slope ($\beta_2 = 5$).

2. Sampling Variability

- Even with a fixed model and known true parameter, the estimated slope and its confidence interval vary from sample to sample due to random noise.
- This variability is visualized in the plot, where some intervals miss the true value (red lines), and most include it (gray lines).

3. Effect of Confidence Level

- By adjusting the quantile (e.g., from 0.025 to 0.005), you can see how increasing the confidence level (from 95% to 99%) leads to: Wider intervals, Higher coverage, Fewer misses (red lines)
- This helps students understand the trade-off between precision and confidence.

4. Law of Large Numbers in Simulation

- As M increases (e.g., from 100 to 10,000), the empirical coverage (ratiog) converges to the theoretical confidence level.
- This reinforces the idea that confidence intervals are long-run properties, not guarantees for individual samples.

Why This Is Powerful

This exercise bridges the gap between theory and intuition. Many learners misunderstand confidence intervals as probabilistic statements about a single interval. By simulating and visualizing the process, the exercise:

- Clarifies the correct interpretation
- Builds trust in statistical inference methods
- Reveals the role of randomness in estimation

5. Ray Fair (https://fairmodel.econ.yale.edu/) work on US presidential elections, includes the following regression model for the Democratic share of the two party presidential vote:

$$VP_t = \beta_1 + \beta_2 I_t + \beta_3 DPER_t + \beta_4 DUR_t + \beta_5 WAR_t + \beta_6(G_t \cdot I_t) + \beta_7(P_t \cdot I_t) + \beta_8(Z_t \cdot I_t) + \epsilon_t$$

where:

$VP \equiv$ Democratic share of the two party presidential vote in election year t,

$I \equiv 1$ if party in White House in election year is Democrat and -1 if its Republican;

$DPER \equiv 1$ if Democratic president in office is running for reelection; -1 is a Republican president is running again, 0 otherwise;

$DUR \equiv 0$ if party in White House has been in office for only one term, $1[-1]$ if the Democratic [Republican] party has been in the White house for 2 consecutive terms, $1.25[-1.25]$ if the Democratic [Republican] party has been in the White house for 3 consecutive terms, $1.5[-1.5]$ if the Democratic [Republican] party has been in the White house for 4 consecutive terms and so on;

$WAR \equiv 1$ for election years 1918, 1920, 1942, 1944, 1946, 1948, 0 otherwise;

$G \equiv$ growth rate of real GDP in the first 3 quarters of the on-term election year (annual rate);

$P \equiv$ absolute value of growth rate of GDP deflator in the first 15 quarters of the administration (annual rate) except for 1920, 1944, and 1948, where values set to 0;

$Z \equiv$ number of quarters in the first 15 quarters of the administration in which the growth rate of real per capita GDP exceeds 3.2 % at an annual rate except for 1920, 1944, and 1948, where the values are zero.

Fair argues that there are 4 conditions that affect voting patterns in US presidential elections:

(i) Incumbent presidents running again for reelection: Voters tend to favor presidents running again.

(ii) How long a party has controlled the White House. Voters like change. When a party has been in power for two or more consecutive terms, this has a negative effect on votes for that party's candidate.

(iii) There is a slight, but persistent, bias in favor of the Republican Party.

(iv) The state of the economy. A good economy at the time of the election has a positive effect on votes for the incumbent party candidate.

(a) Each of the four Fair's conditions translate in terms of the sign of the parameters of the regression above. Taking the conditions one by one, identify the relevant parameter that captures that condition and briefly argue what the expected sign for the parameter is according to Fair.

(b) With the help of Stata, estimate the model above using data on US elections from 1916 to 2020 included in file USelections.csv (Fair's data). Include the Stata output in your answer. Are the signs of the OLS estimates of the parameters as expected given Fair's 4 conditions? Briefly argue.

(c) Stata output includes, among others, the default calculations of the following statistics:

i. $se(\widehat{\beta}_6)$
ii. t−value and p−value associated with the significance test of regressor I.

Provide the specific expression defining each of these three statistics, and then using R/Python calculate each these statistics by using the expression that defines them, and verify you get the same values as the ones reproduced in the Stata output.

(d) Test $H_0 : \beta_7 = 0$ versus $H_0 : \beta_7 \neq 0$ using exact t − test statistic and 5% significance level, basing your decision rule on the comparison of t − value with the criticalvalue. Draw the associated acceptance and rejection regions properly labelling both axes. What did you conclude?

(e) Draw the p-value associated to the test performed in question 5d. Properly label the axes. What is the minimum significance level that would make this regressor be significant?

(f) We want to test whether the famous phrase "it's the economy, stupid!" " has empirical evidence. With the help of Stata, test whether the economic variables $G \cdot I, P \cdot I$ and $Z \cdot I$ are jointly significant. Show all the steps. Include the Stata output as the answer.

(g)

(h) Given that the latest estimate of the democratic share of the two-party vote for November 2024 elections is 49.25, what is the prediction error we would get if we used R. Fair values as for $G = 1.7, P = 4.54$ and $Z = 4$ before the election? Justify.

(ii) What can you say about the method used by Fair, and reproduced in this exercise, to predict VP? Briefly justify.