Econometrics

TA Session 4

Lucia Sauer

2025-10-15

Overview

- Global Hypothesis Testing
- Multiple Hypothesis Testing
- Monte Carlo Simulations

Let's start by running the following regression:

$$\texttt{colGPA}_i = \beta_1 + \beta_2 \texttt{hsGPA}_i + \beta_3\ \texttt{job19}_i + \beta_4\ \texttt{job20}_i + \beta_5\ \texttt{skipped}_i + \beta_6\ \texttt{bgfriend}_i + \beta_7\ \texttt{alcohol}_i + \varepsilon_i$$

where:

- `colGPA`: college GPA
- `hsGPA`: high school GPA
- `job19`: worked in 2019 (1=yes, 0=no)
- `job20`: worked in 2020 (1=yes, 0=no)
- `skipped`: skipped classes (1=yes, 0=no)
- `bgfriend`: has a boyfriend/girlfriend (1=yes, 0=no)
- `alcohol`: alcohol consumption (1=yes, 0=no)

```
bcuse gpa1, clear
regress colgpa hsGPA job19 job20 skipped bgfriend alcohol
```

Global Hypothesis Testing

What is testing the F value present in the regression output?

Global Hypothesis Testing

We want to test whether our regression model adds explanatory power beyond the mean.

Exercise

1. Indicate **null and alternative hypotheses**.
2. Write the expression of the **F-test statistic** used for this test, and its assumed distribution.
3. Run the restricted model, compute the RSSE (restricted model), SSE (unrestricted model) and F-statistic.
4. Find the critical value of the F-distribution and compute the p-value.

---

1. Indicate null and alternative hypotheses.

---

2. F-test statistic

---

3. RSSE and SSE

---

4. Critical value and p-value

Multiple Hypothesis Testing

Consider testing whether job19 and job20 are jointly significant at $\alpha = 0.05$:

Exercise

1. Indicate **null and alternative hypotheses**.
2. Write the expression of the **F-test statistic** used for this test, and its assumed distribution.
3. Run the restricted model, compute the RSSE (restricted model), SSE (unrestricted model) and F-statistic.
4. Find the critical value of the F-distribution and compute the p-value.
5. Draw the p-value and the critical value.
6. Compare the results with the ones obtained in Stata.

Monte Carlo Simulations

Monte Carlo Casino

Monte Carlo: a lab for estimators

Workflow

1. **Specify a known DGP** (the "true" model).
2. **Generate** many random samples from it.
3. **Estimate** the coefficients repeatedly.
4. **Observe** the estimator's behavior across replications:
    - mean (*bias*)
    - spread (*variance*)
    - shape (*sampling distribution*)

Data-Generating Process (DGP)

We will generate $m = 10000$ samples of size $n = 100$ from the following DGP:

DGP

$$y_i = 4 + 2x_{i2} + 2x_{i3} + \varepsilon_i$$

$$\varepsilon_i \,|\, X_i \sim \text{i.i.d. } N(0, 32)$$

$$x_{i2} \sim U[0, 40], \quad x_{i3} = x_{i2} + v_i,\ v_i \sim N(0, 16)$$

Function in Python

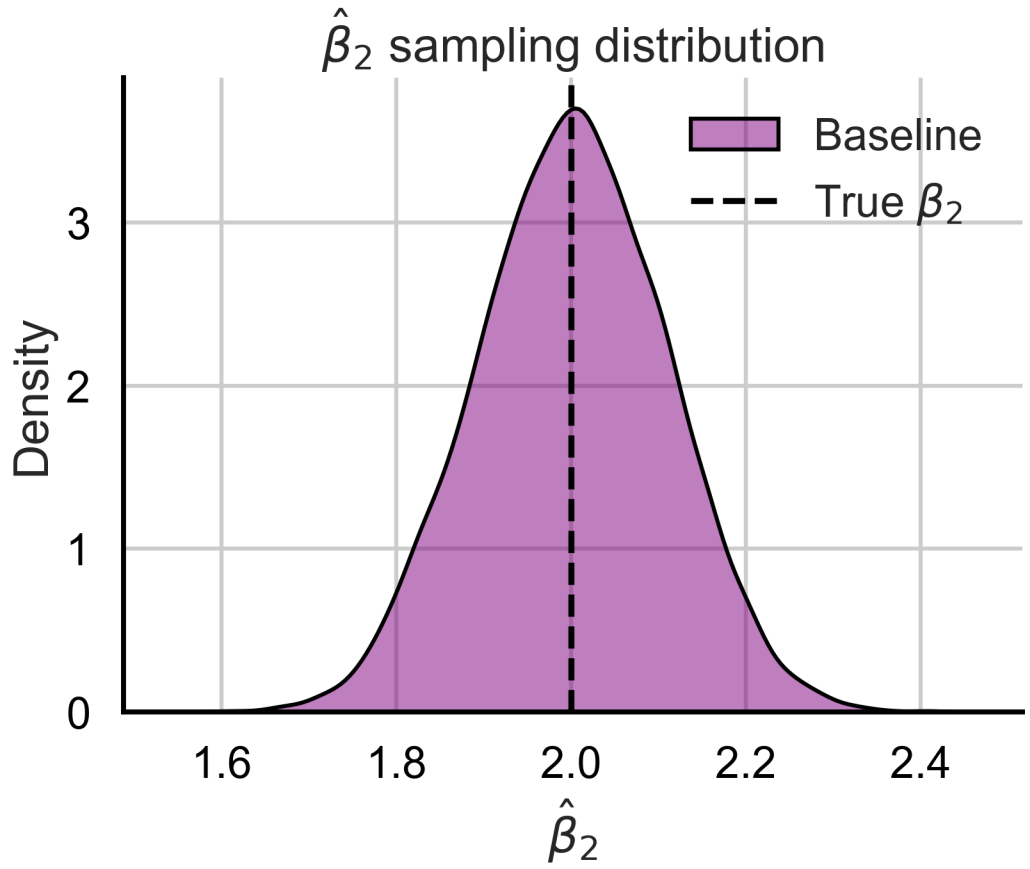
Let's create a function to analyze the behavior of $\hat{\beta}_2$:

```
def simulate_betas(n=100, sigma_eps=32, sigma_v=16, reps=10000, conditional=False):
    """
    Monte Carlo simulation of   from y = 4 + 2x  + 2x  +  .
    """
    betas = []

    # For conditional distribution: fix X once
    if conditional:
        x2_fixed = np.random.uniform(0, 40, n)
        v_fixed  = np.random.normal(0, sigma_v, n)
        x3_fixed = x2_fixed + v_fixed

    for _ in range(reps):
        if conditional:
            x2, x3 = x2_fixed, x3_fixed
        else:
            x2 = np.random.uniform(0, 40, n)
            v  = np.random.normal(0, sigma_v, n)
            x3 = x2 + v

        eps = np.random.normal(0, sigma_eps, n)
        y = 4 + 2*x2 + 2*x3 + eps

        X = sm.add_constant(np.column_stack([x2, x3]))
        model = sm.OLS(y, X).fit()
        betas.append(model.params[1])
```

```
    return np.array(betas)
```

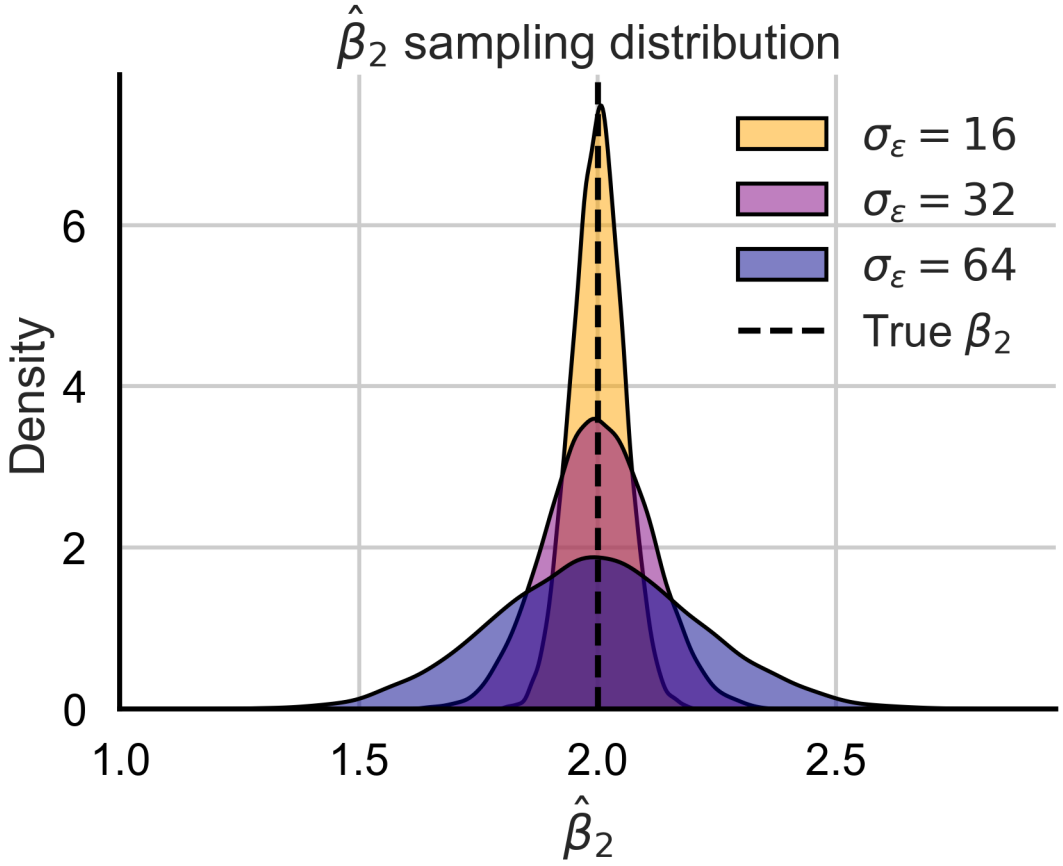
Let's run the simulation for the conditional distribution of $\hat{\beta}_2$:

```
betas_base = simulate_betas(n=1000, sigma_eps=32, sigma_v=16, reps=10000)

plt.figure(figsize=(6,5))
sns.kdeplot(betas_base, fill=True, alpha=0.5, color="purple", label="Baseline", edgecolor="bl
plt.axvline(2, color="black", ls="--", label=r"True $\beta_2$")
plt.title(r"$\hat{\beta}_2$ sampling distribution")
plt.xlabel(r"$\hat{\beta}_2$")
plt.show()
```



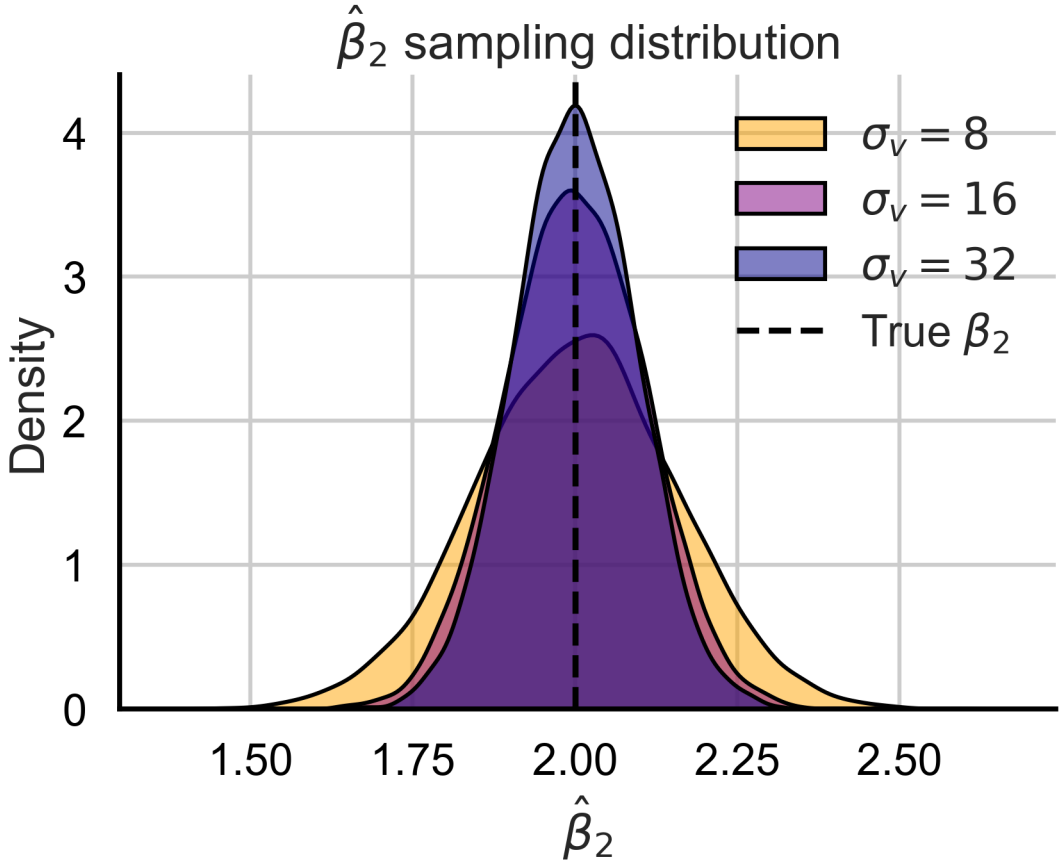
Now, let's increase σ_ε^2:

```
betas_high_sigma = simulate_betas(n=1000, sigma_eps=64, sigma_v=16, reps=10000)
betas_low_sigma = simulate_betas(n=1000, sigma_eps=16, sigma_v=16, reps=10000)

plt.figure(figsize=(6,5))
for sns, color, label in zip([betas_low_sigma, betas_base, betas_high_sigma], [r"$\sigma_{\va
    sns.kdeplot(sns, fill=True, alpha=0.5, edgecolor="black", color=color, label=label)
plt.axvline(2, color="black", ls="--", label=r"True $\beta_2$")
plt.title(r"$\hat{\beta}_2$ sampling distribution")
plt.xlabel(r"$\hat{\beta}_2$")
plt.show()
```



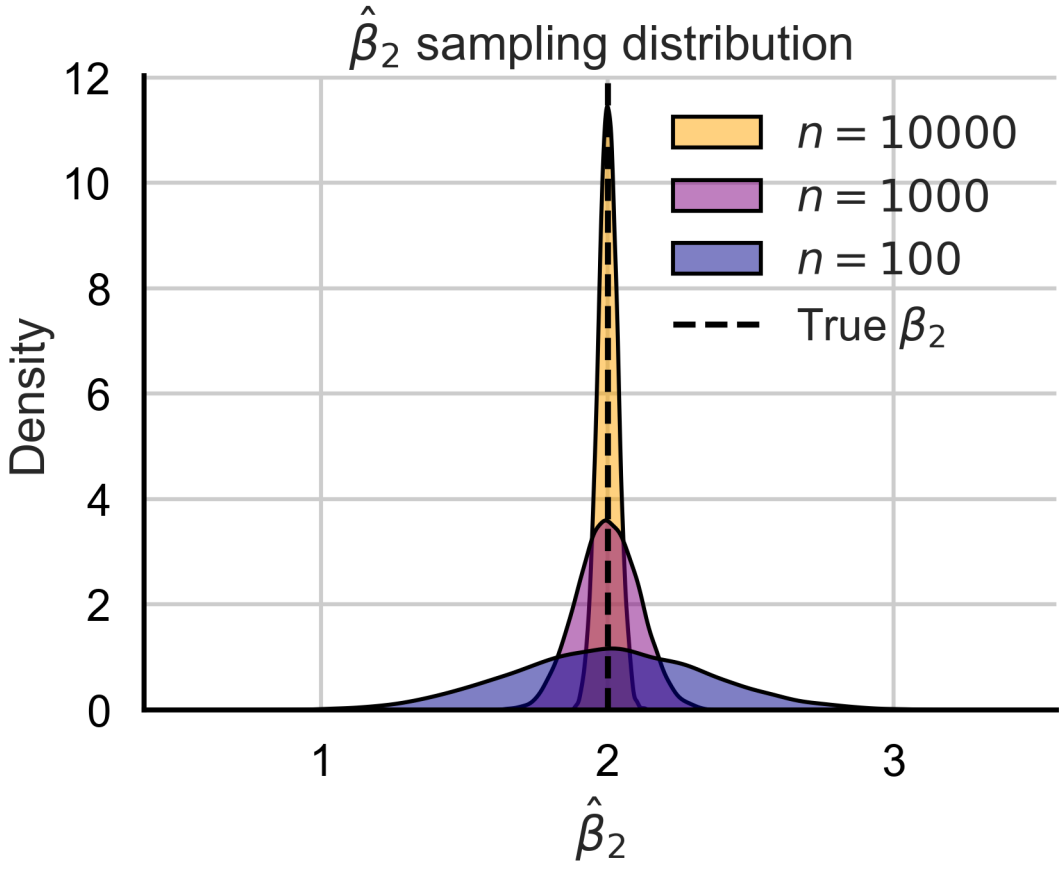
Now, let's reduce σ_v^2, the collinearity between x_2 and x_3:

```
betas_high_collinear = simulate_betas(n=1000, sigma_eps=32, sigma_v=32, reps=10000)
betas_low_collinear = simulate_betas(n=1000, sigma_eps=32, sigma_v=8, reps=10000)

plt.figure(figsize=(6,5))
for sns, color, label in zip([betas_low_collinear, betas_base, betas_high_collinear], [r"$\s
    sns.kdeplot(sns, fill=True, alpha=0.5, edgecolor="black", color=color, label=label)
plt.axvline(2, color="black", ls="--", label=r"True $\beta_2$")
plt.title(r"$\hat{\beta}_2$ sampling distribution")
plt.xlabel(r"$\hat{\beta}_2$")
plt.show()
```



Increasing the sample size n:

```
betas_large_sample = simulate_betas(n=10000)
betas_small_sample = simulate_betas(n=100)


plt.figure(figsize=(6,5))
for sns, color, label in zip([betas_small_sample, betas_base, betas_large_sample], [r"$n=100$
    sns.kdeplot(sns, fill=True, alpha=0.5, edgecolor="black", color=color, label=label)
plt.axvline(2, color="black", ls="--", label=r"True $\beta_2$")
plt.title(r"$\hat{\beta}_2$ sampling distribution")
plt.xlabel(r"$\hat{\beta}_2$")
plt.show()
```



Now, running the simulation for the unconditional distribution of $\hat{\beta}_2$:

```
betas_base_cond = simulate_betas()
betas_base_uncond = simulate_betas(conditional=True)


plt.figure(figsize=(6,5))
for sns, color, label in zip([betas_base_uncond, betas_base_cond], [r"$Conditioned on X$", r'
    sns.kdeplot(sns, fill=True, alpha=0.5, edgecolor="black", color=color, label=label)
plt.axvline(2, color="black", ls="--", label=r"True $\beta_2$")
plt.title(r"$\hat{\beta}_2$ sampling distribution")
plt.xlabel(r"$\hat{\beta}_2$")
plt.show()
```