ECON 124: Problem Set #1

Due on Jun 15, 2025

Dr. Deniz Baglan

Alejandro Ouslan

Problem 1

Consider two least-square Regressions

$$y = X_1\beta_1 + \epsilon$$

and

$$y = X_1\beta_1 + X_2\beta_2\epsilon$$

Let R_1^2 and R_2^2 be the R+squared from the two regressions, respectively. Show that $R_2^2 \geq R_1^2$

Proof.

$$\begin{aligned} R_1^2 \leq R_2^2 &\Rightarrow 1 - \frac{SSR_1}{TSS} \leq 1 - \frac{SSR_2}{TSS} \\ &\Rightarrow SSR_2 \leq SSR_1 \\ &\Rightarrow ||(I - P_{X_2})y||^2 \leq ||(I - P_{X_1})y||^2 \end{aligned}$$

Given that $col(X_1) \subset col(X_2)$ and P_{X_1} is a projection of y onto the subspace $col(X_1)$ then:

$$\begin{aligned} &\Rightarrow ||(I - P_{X_2})y||^2 \leq ||(I - P_{X_1})y||^2 \\ &\Rightarrow SSR_2 \leq SSR_1 \\ &\Rightarrow 1 - \frac{SSR_1}{TSS} \leq 1 - \frac{SSR_2}{TSS} \\ &\Rightarrow R_1^2 \leq R_2^2 \end{aligned}$$

Thus $R_1^2 \leq R_2^2$ □

Problem 2

Use the cps09mar data for this question. The data set and the description file is attached under the assignment of Canvas. Estimate a log wage regression for the subsample of white male Hispanics. In addition to education, experience, and its square, include a set of binary variables for Northeast, South and West so that Midwest is the exuded group. For marital status, create variables for married, widow or divorced, and separated, so that single (never married) is the excluded group.

Variable	**Coefficient**
Intercept	1.0019
Education	0.0885
North Dummy	0.0574
South Dummy	-0.0729
West Dummy	0.0231
Married Dummy	0.1682
Widowed Dummy	0.1853
Separated Dummy	0.0523
Experience (exp)	0.0337
Experience^2 (exp_2)	-0.0004

Table 1: Estimated coefficients from regression model

Problem 2 continued on next page...

Python Code

```python
import numpy as np
import polars as pl
import statsmodels.api as sm

df = pl.read_excel("data/Koop_Tobias_subsample.xlsx").to_pandas()

Y = df["lwage"].values.reshape(-1, 1)
X_1 = df[["Education", "Experience", "Ability"]].values.reshape(-1, 3)
X_1 = sm.add_constant(X_1)

n = len(Y)
p = 3

beta_1 = np.linalg.pinv(X_1.T @ X_1) @ X_1.T @ Y

y_pred = X_1 @ beta_1

ss_res = np.sum((Y - y_pred) ** 2)
ss_tot = np.sum((Y - np.mean(Y)) ** 2)
r_squared = 1 - ss_res / ss_tot

r_squared_adj = 1 - ((1 - r_squared) * (n - 1)) / (n - p - 1)

print("Model 1")
print("R:", r_squared)
print(beta_1)
print("Adjusted R:", r_squared_adj)

Y = df["lwage"].values.reshape(-1, 1)
X_1 = df[["Mothers_Educ", "Fathers_Educ", "Siblings"]].values.reshape(-1, 3)

n = len(Y)
p = 3

beta_1 = np.linalg.pinv(X_1.T @ X_1) @ X_1.T @ Y

y_pred = X_1 @ beta_1

ss_res = np.sum((Y - y_pred) ** 2)
ss_tot = np.sum((Y - np.mean(Y)) ** 2)
r_squared = 1 - ss_res / ss_tot

r_squared_adj = 1 - ((1 - r_squared) * (n - 1)) / (n - p - 1)

print("model 2")
print("R:", r_squared)
print(beta_1)
print("Adjusted R:", r_squared_adj)
```

Problem 2 continued on next page...

```
data = df.to_pandas()
Y = data["lwage"].values.reshape(-1, 1)
X_1 = data[
    ["Education", "Experience", "Ability", "Mothers_Educ", "Fathers_Educ", "Siblings"]
].values.reshape(-1, 6)

n = len(Y)
p = 6

beta_1 = np.linalg.pinv(X_1.T @ X_1) @ X_1.T @ Y

y_pred = X_1 @ beta_1

ss_res = np.sum((Y - y_pred) ** 2)
ss_tot = np.sum((Y - np.mean(Y)) ** 2)
r_squared = 1 - ss_res / ss_tot

r_squared_adj = 1 - ((1 - r_squared) * (n - 1)) / (n - p - 1)

print("full model")
print("R:", r_squared)
print(beta_1)
print("Adjusted R:", r_squared_adj)


residuals = []  # c
X1_vars = [
    "Education",
    "Experience",
    "Ability",
]
X2_vars = [
    "Mothers_Educ",
    "Fathers_Educ",
    "Siblings",
]


X1 = sm.add_constant(df[X1_vars].values.reshape(-1, 3))

for var in X2_vars:
    Y = df[var].values.reshape(-1, 1)
    bata_hat = np.linalg.pinv(X1.T @ X1) @ X1.T @ Y
    res = Y - X1 @ bata_hat
    residuals.append(res)

X2_star2 = np.column_stack(residuals)
print(X2_star2)
```

Problem 2 continued on next page...

Problem 3

The data koop tobias subsample is extracted from koop and tobias (2004) study of the relationship between wages and education, ability, and family characterises. Et X_1 equal a constant, education, experience, and ability. Let X_2 contain the mothers education, the fathers education, and the number of siblings. Let y e the log of the hourly wage. Show you regression output

1. Compute the least square regression coefficients in the regression of y on X_1. Report the coefficients.

Variable	**Coefficient**
Intercept	1.6636
Education	0.0145
Experience	0.0710
Ability	0.0266

Table 2: Estimated coefficients for the regression model

2. Compute the least square regression coefficients in the regression of y on X_1 and X_2. Report the coefficients.

Variable	**Coefficient**
Mother's Education	0.1028
Father's Education	0.0490
Siblings	0.0847

Table 3: Regression coefficients without intercept

3. Regress each of the three variables in X_2 on all the variables in X_1 and copete the residuals form each regression. Arrange these new variables in the (15X3) matrix $X_2^\star$. What are the sample means of these three variables?

$$X_2^\star = \begin{bmatrix} -0.8891 & -1.2215 & -0.6868 \\ -0.1152 & -1.7148 & 0.0547 \\ -0.4563 & 0.0457 & -2.1323 \\ -0.6006 & -2.5489 & 1.5568 \\ -0.0837 & -0.6653 & -1.2062 \\ -0.1541 & 3.2126 & -0.0721 \\ -0.1044 & -1.1987 & -0.5662 \\ 0.2152 & 2.2027 & 0.0112 \\ -0.5327 & -1.1005 & 0.2349 \\ 0.4595 & -0.5267 & -0.2678 \\ 2.6149 & 3.0896 & 1.0509 \\ 2.1254 & 0.7824 & 1.2776 \\ 0.4188 & 0.2255 & 0.8159 \\ -0.8591 & -0.0135 & -0.7581 \\ -2.0388 & -0.5686 & 0.6875 \end{bmatrix}$$

4. Compute the R^2 for the regression of y on X_1 and X_2. Repeat the computation for the case in which the constant term is omitted from X_1. What happens to R^2?
   When the intercept is included, the R^2 values are positive and reflect the proportion of variance explained by the model. However, when the intercept is omitted, some R^2 values become negative, indicating that the regression model fits the data worse than a simple horizontal line at the mean of y.
5. Compute the adjusted R^2 for the full regression including the constant term.

Problem 3 continued on next page...

Regression Setup	R^2
With intercept included (first model)	0.1834
With intercept included (alternative)	0.1916
Without intercept included (firs model)	-0.3537
Without intercept included (alternative)	-0.0219

Table 4: Values of R^2 under different regression setups

$$\text{Adjusted } R^2 = 0.1530$$

Python Code

```python
import numpy as np
import polars as pl
import statsmodels.api as sm

df = pl.read_excel("data/Koop_Tobias_subsample.xlsx").to_pandas()

Y = df["lwage"].values.reshape(-1, 1)
X_1 = df[["Education", "Experience", "Ability"]].values.reshape(-1, 3)
X_1 = sm.add_constant(X_1)

n = len(Y)
p = 3

beta_1 = np.linalg.pinv(X_1.T @ X_1) @ X_1.T @ Y

y_pred = X_1 @ beta_1

ss_res = np.sum((Y - y_pred) ** 2)
ss_tot = np.sum((Y - np.mean(Y)) ** 2)
r_squared = 1 - ss_res / ss_tot

r_squared_adj = 1 - ((1 - r_squared) * (n - 1)) / (n - p - 1)

print("Model 1")
print("R:", r_squared)
print(beta_1)
print("Adjusted R:", r_squared_adj)

Y = df["lwage"].values.reshape(-1, 1)
X_1 = df[["Mothers_Educ", "Fathers_Educ", "Siblings"]].values.reshape(-1, 3)

n = len(Y)
p = 3

beta_1 = np.linalg.pinv(X_1.T @ X_1) @ X_1.T @ Y

y_pred = X_1 @ beta_1
```

```
ss_res = np.sum((Y - y_pred) ** 2)
ss_tot = np.sum((Y - np.mean(Y)) ** 2)
r_squared = 1 - ss_res / ss_tot

r_squared_adj = 1 - ((1 - r_squared) * (n - 1)) / (n - p - 1)

print("model 2")
print("R:", r_squared)
print(beta_1)
print("Adjusted R:", r_squared_adj)


data = df.to_pandas()
Y = data["lwage"].values.reshape(-1, 1)
X_1 = data[
    ["Education", "Experience", "Ability", "Mothers_Educ", "Fathers_Educ", "Siblings"]
].values.reshape(-1, 6)

n = len(Y)
p = 6

beta_1 = np.linalg.pinv(X_1.T @ X_1) @ X_1.T @ Y

y_pred = X_1 @ beta_1

ss_res = np.sum((Y - y_pred) ** 2)
ss_tot = np.sum((Y - np.mean(Y)) ** 2)
r_squared = 1 - ss_res / ss_tot

r_squared_adj = 1 - ((1 - r_squared) * (n - 1)) / (n - p - 1)

print("full model")
print("R:", r_squared)
print(beta_1)
print("Adjusted R:", r_squared_adj)


residuals = []  # c
X1_vars = [
    "Education",
    "Experience",
    "Ability",
]
X2_vars = [
    "Mothers_Educ",
    "Fathers_Educ",
    "Siblings",
]
```

Problem 3 continued on next page...

```
X1 = sm.add_constant(df[X1_vars].values.reshape(-1, 3))

for var in X2_vars:
    Y = df[var].values.reshape(-1, 1)
    bata_hat = np.linalg.pinv(X1.T @ X1) @ X1.T @ Y
    res = Y - X1 @ bata_hat
    residuals.append(res)

X2_star2 = np.column_stack(residuals)
print(X2_star2)
```

Suppose that you have two independent unbiased estimators of the same parameter θ, say $\hat{\theta}_1$ and $\hat{\theta}_2$, with different variances v_1 and v_2, What linear combination $\hat{\theta} = c_1\hat{\theta}_1 + c_2\hat{\theta}_2$ is the minimum variance unbiased estimator of θ?

$$\hat{\theta} = c_1\hat{\theta}_1 + c_2\hat{\theta}_2$$

$$E[\hat{\theta}] = c_1E[\hat{\theta}_1] + c_2E[\hat{\theta}_2] = c_1\theta + c_2\theta = (c_1 + c_2)\theta.$$

$$c_1 + c_2 = 1.$$

$$\text{Var}(\hat{\theta}) = c_1^2v_1 + c_2^2v_2.$$

$$\text{Var}(\hat{\theta}) = c_1^2v_1 + (1 - c_1)^2v_2 = c_1^2v_1 + (1 - 2c_1 + c_1^2)v_2 = c_1^2(v_1 + v_2) - 2c_1v_2 + v_2.$$

$$\frac{d}{dc_1}\text{Var}(\hat{\theta}) = 2c_1(v_1 + v_2) - 2v_2 = 0 \implies c_1 = \frac{v_2}{v_1 + v_2}.$$

$$\boxed{\hat{\theta} = \frac{v_2}{v_1 + v_2}\hat{\theta}_1 + \frac{v_1}{v_1 + v_2}\hat{\theta}_2.}$$

The variance of this estimator is

$$\text{Var}(\hat{\theta}) = \frac{v_1v_2}{v_1 + v_2}.$$

Analyze the properties of the Least Squares (LS) estimator of the slope coefficient in a simple linear regression model using the Monte Carl method. Specifically, the true model is given by

$$y_i = 1 + 2x_i + \epsilon_i$$

Where the intercept is 1, and the slope coefficient is 2. The errors ϵ_i are normally distributed with mean zero and variance 3, $\epsilon \sim N(0, 6)$. The independent variable x is uniformly distributed between 0 and 6, $X \sim (0, 6)$.

Conduct the following experiment: Draw 1000 samples of sizes 25,50, and 100 from the distribution of the error term and the predictor variable. For each sample, estimate the slope coefficient using the LS estimator. Then, analyze the

1. Unbiasedness, variance, and the shape of the distribution of the LS estimator of the slope coefficient by plotting the distribution for each sample size.

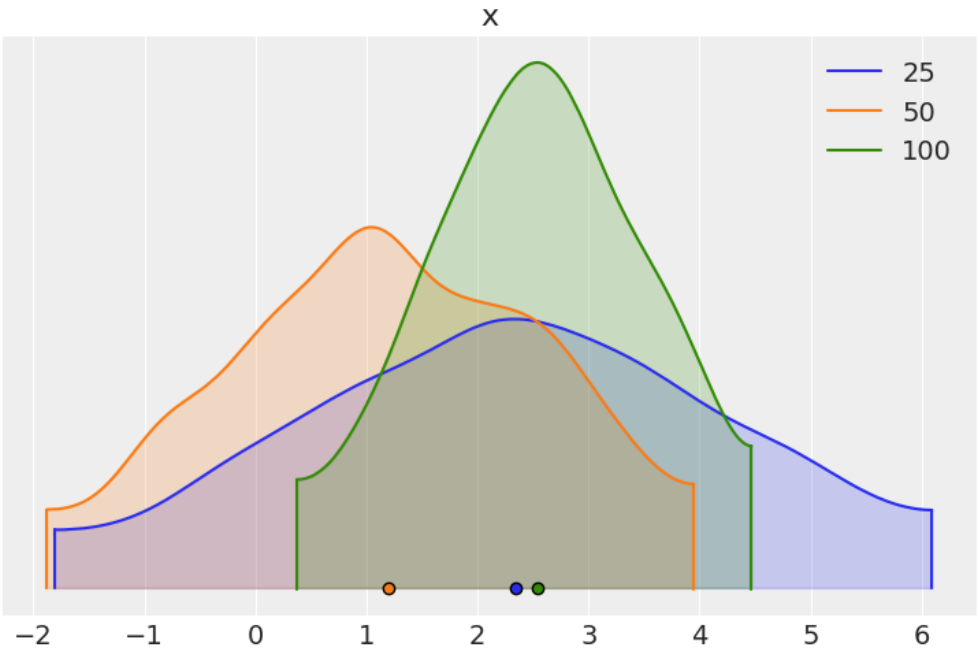

Figure 1: Convergencia de los métodos implementados

2. Compare the variance of the estimated slope coefficient obtained form the Monte Carlo simulation with the variance obtained using the asymptotic formula and the variance obtained using a bootstrap method

Sample Size	Posterior Var	Asymptotic Var	Bootstrap Var
25	4.587303	4.070104	3.484380
50	2.539172	2.327265	2.142014
100	1.193076	1.073991	1.129780

Table 5: Comparison of variances for different sample sizes

Python Code

```
import arviz as az
import numpy as np
import pandas as pd
import bambi as bmb
from sklearn.linear_model import LinearRegression

az.style.use("arviz-darkgrid")


def mc_func(size, intercept, slope, SEED=787):
    rng = np.random.default_rng(SEED)
    x = np.linspace(0, 1, size)
    true_regression_line = intercept + slope * x
    y = true_regression_line + rng.normal(scale=3, size=size)
    data = pd.DataFrame({"x": x, "y": y})
```

```
    model = bmb.Model("y ~ x", data)
    idata = model.fit(draws=1000, chains=1)
    return idata


def asymptotic_variance(size, intercept, slope, seed=787):
    rng = np.random.default_rng(seed)
    x = np.linspace(0, 1, size)
    true_regression_line = intercept + slope * x
    y = true_regression_line + rng.normal(scale=3, size=size)
    data = pd.DataFrame({"x": x, "y": y})

    model = LinearRegression().fit(data[["x"]], data["y"])
    residuals = data["y"] - model.predict(data[["x"]])
    sigma_sq = np.var(residuals, ddof=1)
    x_centered = data["x"] - data["x"].mean()
    var_beta1 = sigma_sq / np.sum(x_centered**2)

    return var_beta1


def bootstrap_slope_variance(size, intercept, slope, n_bootstrap=1000, seed=787):
    rng = np.random.default_rng(seed)
    x = np.linspace(0, 1, size)
    true_regression_line = intercept + slope * x
    y = true_regression_line + rng.normal(scale=3, size=size)
    data = pd.DataFrame({"x": x, "y": y})

    coefs = []
    for _ in range(n_bootstrap):
        sample = data.sample(n=size, replace=True, random_state=rng.integers(1e6))
        model = LinearRegression().fit(sample[["x"]], sample["y"])
        coefs.append(model.coef_[0])

    return np.var(coefs)


true_intercept = 1
true_slope = 2

asymp_var_25 = asymptotic_variance(25, true_intercept, true_slope)
asymp_var_50 = asymptotic_variance(50, true_intercept, true_slope)
asymp_var_100 = asymptotic_variance(100, true_intercept, true_slope)

boot_var_25 = bootstrap_slope_variance(25, true_intercept, true_slope)
boot_var_50 = bootstrap_slope_variance(50, true_intercept, true_slope)
boot_var_100 = bootstrap_slope_variance(100, true_intercept, true_slope)

idata1 = mc_func(size=25, slope=2, intercept=1)
```

```
idata2 = mc_func(size=50, slope=2, intercept=1)
idata3 = mc_func(size=100, slope=2, intercept=1)

posterior_slope_25 = idata1.posterior["x"].values.flatten()
posterior_slope_50 = idata2.posterior["x"].values.flatten()
posterior_slope_100 = idata3.posterior["x"].values.flatten()

posterior_var_25 = posterior_slope_25.var()
posterior_var_50 = posterior_slope_50.var()
posterior_var_100 = posterior_slope_100.var()

az.plot_density(
    [idata1, idata2, idata3],
    var_names=["x"],
    data_labels=["25", "50", "100"],
    shade=0.2,
)

df = pd.DataFrame(
    {
        "Sample Size": [25, 50, 100],
        "Posterior Var": [posterior_var_25, posterior_var_50, posterior_var_100],
        "Asymptotic Var": [asymp_var_25, asymp_var_50, asymp_var_100],
        "Bootstrap Var": [boot_var_25, boot_var_50, boot_var_100],
    }
)

print(df)
```