One-way ANOVA as a regression

STAT 6160 • Experimental Design • Fall 2026

Siddharth Vishwanath

University of Virginia

§1. The data

Four treatments

Group i	1	2	3	4
Treatment	A	B	C	D
n_i	4	5	4	7
μ_i	20	20	20	24

$$y_{ij} = \mu_i + \epsilon_{ij}, \qquad \epsilon_{ij} \overset{iid}{\sim} \mathcal{N}(0, \sigma^2), \qquad \sigma = 1.5, \qquad N = \sum_i n_i = 20$$

- A is the control. Only D differs from it.
- The group sizes are unequal.

Simulating the data

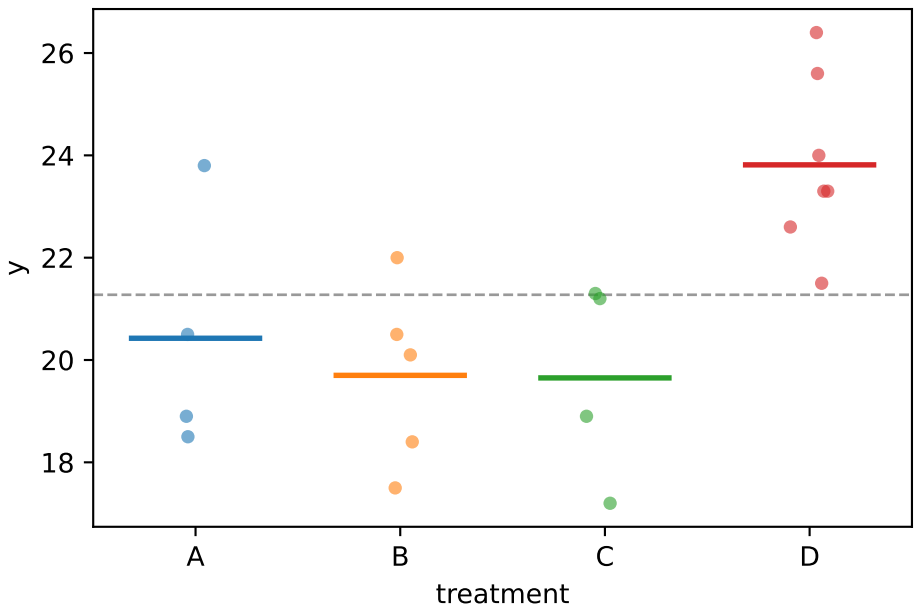
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.formula.api as smf
from statsmodels.stats.anova import anova_lm

groups = ["A", "B", "C", "D"]
n = np.array([4, 5, 4, 7])
mu = np.array([20, 20, 20, 24])
N = n.sum()

rng = np.random.default_rng(6181)
df = pd.DataFrame({"treatment": np.repeat(groups, n)})
df["y"] = np.repeat(mu, n) + rng.normal(0, 1.5, N)
df["y"] = df["y"].round(1)
```

Point plot

```
fig, ax = plt.subplots()
sns.stripplot(data=df, x="treatment",
              y="y", hue="treatment",
              jitter=0.1, alpha=0.6,
              legend=False, ax=ax)
sns.pointplot(data=df, x="treatment",
              y="y", hue="treatment",
              errorbar=None, markers="_",
              markersize=50,
              legend=False, ax=ax)
ax.axhline(df["y"].mean(), color="0.6",
           ls="--", lw=1)
plt.show()
```



§2. Population and sample quantities

Means and differences

Population	Sample
group mean μ_i	$\bar{y}_{i\bullet} = \frac{1}{n_i}\sum_j y_{ij}$
difference from control $\mu_i - \mu_1$	$\bar{y}_{i\bullet} - \bar{y}_{1\bullet}$
weighted mean $\mu_w = \sum_i p_i \mu_i, p_i = n_i/N$	$\bar{y}_{\bullet\bullet} = \frac{1}{N}\sum_{i,j} y_{ij} = \sum_i p_i \bar{y}_{i\bullet}$

- $\bar{y}_{i\bullet}$ is unbiased for μ_i, with $\mathrm{Var}\,(\bar{y}_{i\bullet}) = \sigma^2/n_i$.
- $\bar{y}_{\bullet\bullet}$ estimates μ_w, which depends on the group sizes.

Parameters and estimates

Population

i	n_i	μ_i	$\mu_i - \mu_1$
A	4	20	0
B	5	20	0
C	4	20	0
D	7	24	4

$$\mu_w = 21.4$$

Sample

i	n_i	$\bar{y}_{i\bullet}$	$\bar{y}_{i\bullet} - \bar{y}_{1\bullet}$
A	4	20.425	0
B	5	19.700	-0.725
C	4	19.650	-0.775
D	7	23.814	3.389

$$\bar{y}_{\bullet\bullet} = 21.275$$

Sample quantities in Python

```
ybar = df.groupby("treatment")["y"].mean()

pd.DataFrame({
    "n": n,
    "ybar": ybar,
    "diff_A": ybar - ybar["A"],
}).round(3)
```

treatment	n	ybar	diff_A
A	4	20.425	0.000
B	5	19.700	−0.725
C	4	19.650	−0.775
D	7	23.814	3.389

```
print(f"grand mean = {df['y'].mean():.3f}")
print(f"sum p_i ybar_i = {(n / N) @ ybar:.3f}")
```

```
grand mean = 21.275
sum p_i ybar_i = 21.275
```

§3. Two models

Two ways to write the model

Dummy coding (reference A)

$$z_k = \mathbb{1}\{\text{unit in group } \ k\}, \qquad k = 2, 3, 4$$

$$y = \beta_0 + \beta_2 z_2 + \beta_3 z_3 + \beta_4 z_4 + \epsilon$$

Cell means (no intercept)

$$x_i = \mathbb{1}\{\text{unit in group } \ i\}, \qquad i = 1, 2, 3, 4$$

$$y = \mu_1 x_1 + \mu_2 x_2 + \mu_3 x_3 + \mu_4 x_4 + \epsilon$$

- With an intercept, only three of the four indicators can enter: $x_1 + x_2 + x_3 + x_4 = 1$.

The coded values

Each row gives the values of the columns for a unit in that group.

Dummy

	1	z_2	z_3	z_4
A	1	0	0	0
B	1	1	0	0
C	1	0	1	0
D	1	0	0	1

Cell means

	x_1	x_2	x_3	x_4
A	1	0	0	0
B	0	1	0	0
C	0	0	1	0
D	0	0	0	1

- Call the left table C_D. The right table is I_4.

The design matrices

Rows are grouped by treatment: $\mathbf{1}_{n_i}$ is a column of n_i ones.

Dummy

$$X_D = \begin{pmatrix} \mathbf{1}_{n_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{1}_{n_2} & \mathbf{1}_{n_2} & \mathbf{0} & \mathbf{0} \\ \mathbf{1}_{n_3} & \mathbf{0} & \mathbf{1}_{n_3} & \mathbf{0} \\ \mathbf{1}_{n_4} & \mathbf{0} & \mathbf{0} & \mathbf{1}_{n_4} \end{pmatrix}$$

```
fit_d = smf.ols("y ~ C(treatment)",
data=df).fit()
X_D = np.array(fit_d.model.exog).astype(int)
```

Cell means

$$X_C = \begin{pmatrix} \mathbf{1}_{n_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1}_{n_2} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1}_{n_3} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1}_{n_4} \end{pmatrix}$$

```
fit_c = smf.ols("y ~ 0 + C(treatment)",
data=df).fit()
X_C = np.array(fit_c.model.exog).astype(int)
```

Both are $N \times 4$ with rank 4.

	0	1	2	3
0	1	0	0	0
1	1	0	0	0
2	1	0	0	0
3	1	0	0	0
4	1	1	0	0
5	1	1	0	0
6	1	1	0	0
7	1	1	0	0
8	1	1	0	0
9	1	0	1	0
10	1	0	1	0
11	1	0	1	0
12	1	0	1	0
13	1	0	0	1
14	1	0	0	1
15	1	0	0	1
16	1	0	0	1
17	1	0	0	1
18	1	0	0	1
19	1	0	0	1

	0	1	2	3
0	1	0	0	0
1	1	0	0	0
2	1	0	0	0
3	1	0	0	0
4	0	1	0	0
5	0	1	0	0
6	0	1	0	0
7	0	1	0	0
8	0	1	0	0
9	0	0	1	0
10	0	0	1	0
11	0	0	1	0
12	0	0	1	0
13	0	0	0	1
14	0	0	0	1
15	0	0	0	1
16	0	0	0	1
17	0	0	0	1
18	0	0	0	1
19	0	0	0	1

What the parameters are

Dummy

$$\mathbb{E}(y_{1j}) = \beta_0, \qquad \mathbb{E}(y_{ij}) = \beta_0 + \beta_i$$

$$\Rightarrow \quad \beta_0 = \mu_1, \qquad \beta_i = \mu_i - \mu_1$$

Here:

- $\boldsymbol{\beta} = (20, 0, 0, 4)$
- $H_0: \quad \beta_2 = \beta_3 = \beta_4 = 0$

Cell means

$$\mathbb{E}(y_{ij}) = \mu_i$$

The coefficients are the group means.

Here:

- $\boldsymbol{\mu} = (20, 20, 20, 24)$
- $H_0: \quad \mu_1 = \mu_2 = \mu_3 = \mu_4$

```
print(fit_d.summary().tables[1])
```

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      20.4250      0.960     21.272      0.000      18.389      22.461
C(treatment)[T.B]    -0.7250      1.288     -0.563      0.581      -3.456       2.006
C(treatment)[T.C]    -0.7750      1.358     -0.571      0.576      -3.654       2.104
C(treatment)[T.D]     3.3893      1.204      2.816      0.012       0.838       5.941
==============================================================================
```

```
print(fit_c.summary().tables[1])
```

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
C(treatment)[A]    20.4250      0.960     21.272      0.000      18.389      22.461
C(treatment)[B]    19.7000      0.859     22.938      0.000      17.879      21.521
C(treatment)[C]    19.6500      0.960     20.465      0.000      17.614      21.686
C(treatment)[D]    23.8143      0.726     32.809      0.000      22.276      25.353
==============================================================================
```

From one model to the other

$$X_D = X_C C_D$$

$$C_D = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}, \qquad C_D^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix}$$

$$\mu = C_D \beta, \qquad \text{and, conversely,} \qquad \beta = C_D^{-1} \mu$$

§4. Least squares by hand

The least squares estimate

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y}$$

- $\boldsymbol{X}^\top \boldsymbol{X}$ depends only on the design.
- $\boldsymbol{X}^\top \boldsymbol{y}$ depends on the data through the group totals $n_i \bar{y}_{i\bullet}$.

$X^\top X$

Dummy

$$X_D^\top X_D = \begin{pmatrix} N & n_2 & n_3 & n_4 \\ n_2 & n_2 & 0 & 0 \\ n_3 & 0 & n_3 & 0 \\ n_4 & 0 & 0 & n_4 \end{pmatrix}$$

Cell means

$$X_C^\top X_C = \begin{pmatrix} n_1 & 0 & 0 & 0 \\ 0 & n_2 & 0 & 0 \\ 0 & 0 & n_3 & 0 \\ 0 & 0 & 0 & n_4 \end{pmatrix}$$

- The cell-means columns do not overlap, so $X_C^\top X_C$ is diagonal.

$$(X_C^\top X_C)^{-1} = \begin{pmatrix} \frac{1}{n_1} & 0 & 0 & 0 \\ 0 & \frac{1}{n_2} & 0 & 0 \\ 0 & 0 & \frac{1}{n_3} & 0 \\ 0 & 0 & 0 & \frac{1}{n_4} \end{pmatrix}$$

```
XD_T_XD = X_D.T @ X_D
```

20	5	4	7
5	5	0	0
4	0	4	0
7	0	0	7

```
XC_T_XC = X_C.T @ X_C
```

4	0	0	0
0	5	0	0
0	0	4	0
0	0	0	7

```
INV_XD_T_XD = np.linalg.inv(XD_T_XD)
```

0.25	−0.25	−0.25	−0.25
−0.25	0.45	0.25	0.25
−0.25	0.25	0.5	0.25
−0.25	0.25	0.25	0.392857

```
INV_XC_T_XC = np.linalg.inv(XC_T_XC)
```

0.25	0	0	0
0	0.2	0	0
0	0	0.25	0
0	0	0	0.142857

$X^\top y$ and the estimates

Cell means

$$X_C^\top y = \begin{pmatrix} n_1 & \overline{y}_{1\bullet} \\ n_2 & \overline{y}_{2\bullet} \\ n_3 & \overline{y}_{3\bullet} \\ n_4 & \overline{y}_{4\bullet} \end{pmatrix}$$

which implies that

$$\boxed{\hat{\mu}_i = \frac{n_i \overline{y}_{i\bullet}}{n_i} = \overline{y}_{i\bullet}}$$

$\hat{\mu}_i$

```
print(fit_c.params)
```

```
C(treatment)[A]    20.425000
C(treatment)[B]    19.700000
C(treatment)[C]    19.650000
C(treatment)[D]    23.814286
dtype: float64
```

```
print(
    (INV_XC_T_XC @ (X_C.T @ df.y)).round(2)
)
```

```
[20.42 19.7  19.65 23.81]
```

$\bar{y}_{i\bullet}$

```
df.groupby("treatment")["y"].mean()
```

```
treatment
A    20.425000
B    19.700000
C    19.650000
D    23.814286
Name: y, dtype: float64
```

Similarly, for the **dummy coding**:

$$\boldsymbol{X}_D^\top \boldsymbol{y} = \begin{pmatrix} N\ \overline{y}_{\bullet\bullet} \\ n_2\ \overline{y}_{2\bullet} \\ n_3\ \overline{y}_{3\bullet} \\ n_4\ \overline{y}_{4\bullet} \end{pmatrix}$$

and, it can be shown that $\boxed{\hat{\beta}_0 = \overline{y}_{1\bullet}, \qquad \hat{\beta}_i = \overline{y}_{i\bullet} - \overline{y}_{1\bullet}}$

$\hat{\beta}_0, \hat{\beta}_i$

```
print(fit_d.params.round(3))
```

```
Intercept            20.425
C(treatment)[T.B]    -0.725
C(treatment)[T.C]    -0.775
C(treatment)[T.D]     3.389
dtype: float64
```

```
print(
    (INV_XD_T_XD @ (X_D.T @ df.y)).round(3)
)
```

```
[20.425 -0.725 -0.775  3.389]
```

$\bar{y}_{i\bullet} - \bar{y}_{1\bullet}$

```
df.groupby("treatment")["y"].mean() -
df.groupby("treatment")["y"].mean()["A"]
```

```
treatment
A    0.000000
B   -0.725000
C   -0.775000
D    3.389286
Name: y, dtype: float64
```

From one model to the other

$$X_D = X_C C_D$$

$$C_D = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}, \qquad C_D^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix}$$

$$\mu = C_D \beta, \qquad \text{and, conversely,} \qquad \beta = C_D^{-1} \mu$$

```
C_D = np.array([[1, 0, 0, 0],
                [1, 1, 0, 0],
                [1, 0, 1, 0],
                [1, 0, 0, 1]])
INV_C_D = np.linalg.inv(C_D)
```

```
(INV_C_D @ fit_c.params).round(3)
```

```
array([20.425, -0.725, -0.775,  3.389])
```

```
fit_c.params.round(3)
```

```
C(treatment)[A]    20.425
C(treatment)[B]    19.700
C(treatment)[C]    19.650
C(treatment)[D]    23.814
dtype: float64
```

```
fit_d.params.round(3)
```

```
Intercept            20.425
C(treatment)[T.B]    -0.725
C(treatment)[T.C]    -0.775
C(treatment)[T.D]     3.389
dtype: float64
```

Standard errors

$$\text{Var}\,(\hat{\boldsymbol{\beta}}) = \sigma^2(\boldsymbol{X}^\top \boldsymbol{X})^{-1}, \qquad \hat{\sigma}^2 = \frac{\text{SS}_{\text{Res}}}{N-4}$$

Dummy

$$\text{Var}\,(\hat{\beta}_0) = \frac{\sigma^2}{n_1},$$

$$\text{Var}\,(\hat{\beta}_i) = \sigma^2\left(\frac{1}{n_1} + \frac{1}{n_i}\right)$$

Cell means

$$\text{Var}\,(\hat{\mu}_i) = \frac{\sigma^2}{n_i},$$

$$\text{Cov}(\hat{\mu}_i, \hat{\mu}_k) = 0$$

and

$$\begin{aligned}\text{Var}\,(\hat{\mu}_i - \hat{\mu}_k) &= \text{Var}\,(\hat{\mu}_i) + \text{Var}\,(\hat{\mu}_k) \\ &= \sigma^2\left(\frac{1}{n_i} + \frac{1}{n_k}\right)\end{aligned}$$

Standard errors in Python

```
s = np.sqrt(fit_c.scale)
pd.DataFrame({
    "dummy": fit_d.bse.values,
    "by hand": s * np.sqrt(
        [1 / n[0], *(1 / n[0] + 1 / n[1:])]),
    "cell means": fit_c.bse.values,
    "s/sqrt(n)": s / np.sqrt(n),
}).round(3)
```

	dummy	by hand	cell means	s/ sqrt(n)
0	0.960	0.960	0.960	0.960
1	1.288	1.288	0.859	0.859
2	1.358	1.358	0.960	0.960
3	1.204	1.204	0.726	0.726

§5. Sums of squares from the hat matrix

One hat matrix for both models

For $J_k = \mathbf{1}_k \mathbf{1}_k^\top$ (a $k \times k$ matrix of ones):

$$\boldsymbol{H} = \boldsymbol{X}_C(\boldsymbol{X}_C^\top \boldsymbol{X}_C)^{-1}\boldsymbol{X}_C^\top = \begin{pmatrix} \frac{1}{n_1}\boldsymbol{J}_{n_1} & & & \\ & \frac{1}{n_2}\boldsymbol{J}_{n_2} & & \\ & & \frac{1}{n_3}\boldsymbol{J}_{n_3} & \\ & & & \frac{1}{n_4}\boldsymbol{J}_{n_4} \end{pmatrix}, \qquad \hat{y}_{ij} = \overline{y}_{i\bullet}$$

With $\boldsymbol{X}_D = \boldsymbol{X}_C \boldsymbol{C}_D$:

$$\boldsymbol{X}_D(\boldsymbol{X}_D^\top \boldsymbol{X}_D)^{-1}\boldsymbol{X}_D^\top = \underbrace{\ldots}_{\text{check this!}} = \boldsymbol{X}_C(\boldsymbol{X}_C^\top \boldsymbol{X}_C)^{-1}\boldsymbol{X}_C^\top$$

Three projections

$$H_1 = \frac{1}{N}\mathbf{1}\mathbf{1}^\top = \begin{pmatrix} 1 & 1 \ldots 1 & \\ 1 & 1 \ldots 1 & \\ \vdots & \vdots & \ldots \ \vdots \\ 1 & 1 \ldots 1 & \end{pmatrix}$$

which implies that

$$H_1 y = \bar{y}_{\bullet\bullet}\mathbf{1}$$

- $\mathbf{1} = X_C \mathbf{1}_4$, so $H\mathbf{1} = \mathbf{1}$ and $HH_1 = H_1$.

Sum of Squares

$$\mathrm{SS}_{\mathrm{Tot}} = \sum_{i,j} (y_{ij} - \bar{y}_{\bullet\bullet})^2 = \boldsymbol{y}^\top (\boldsymbol{I} - \boldsymbol{H}_1)\boldsymbol{y}$$

From which it follows that

$$\begin{aligned}
\mathrm{SS}_{\mathrm{Tot}} &= \boldsymbol{y}^\top (\boldsymbol{I} - \boldsymbol{H}_1)\boldsymbol{y} \\
&= \boldsymbol{y}^\top (\boldsymbol{I} - \boldsymbol{H} + \boldsymbol{H} - \boldsymbol{H}_1)\boldsymbol{y} \\
&= \boldsymbol{y}^\top (\boldsymbol{I} - \boldsymbol{H})\boldsymbol{y} + \boldsymbol{y}^\top (\boldsymbol{H} - \boldsymbol{H}_1)\boldsymbol{y} \\
&= \mathrm{SS}_{\mathrm{Res}} \qquad\qquad + \mathrm{SS}_{\mathrm{Reg}}
\end{aligned}$$

where:

$$\mathrm{SS}_{\mathrm{Reg}} = \boldsymbol{y}^\top (\boldsymbol{H} - \boldsymbol{H}_1)\boldsymbol{y}, \qquad \text{and } \mathrm{SS}_{\mathrm{Res}} = \boldsymbol{y}^\top (\boldsymbol{I} - \boldsymbol{H})\boldsymbol{y}$$

- $\boldsymbol{H} - \boldsymbol{H}_1$ and $\boldsymbol{I} - \boldsymbol{H}$ are projections with $(\boldsymbol{I} - \boldsymbol{H})(\boldsymbol{H} - \boldsymbol{H}_1) = \boldsymbol{0}$.

$\mathrm{SS}_{\mathrm{Reg}} = \mathrm{SS}_{\mathrm{Trt}}$

$$\boldsymbol{y}^\top \boldsymbol{H} \boldsymbol{y} = \sum_{i=1}^{4} n_i \bar{y}_{i\bullet}^2, \qquad \boldsymbol{y}^\top \boldsymbol{H}_1 \boldsymbol{y} = N \bar{y}_{\bullet\bullet}^2$$

$$\mathrm{SS}_{\mathrm{Reg}} = \sum_{i=1}^{4} n_i \bar{y}_{i\bullet}^2 - N \bar{y}_{\bullet\bullet}^2 = \sum_{i=1}^{4} n_i (\bar{y}_{i\bullet} - \bar{y}_{\bullet\bullet})^2 = \mathrm{SS}_{\mathrm{Trt}}$$

$$\mathrm{SS}_{\mathrm{Res}} = \sum_{i,j} y_{ij}^2 - \sum_{i=1}^{4} n_i \bar{y}_{i\bullet}^2 = \sum_{i,j} (y_{ij} - \bar{y}_{i\bullet})^2$$

Degrees of freedom:

$$\mathrm{tr}(\boldsymbol{H} - \boldsymbol{H}_1) = g - 1 = 4 - 1, \qquad \mathrm{tr}(\boldsymbol{I} - \boldsymbol{H}) = N - g = N - 4$$

The hat matrix in practice

```
y = df["y"].values

def hat(X):
    return X @ np.linalg.solve(X.T @ X, X.T)

H, H1 = hat(Xc), np.full((N, N), 1 / N)
print(np.allclose(H, hat(Xd)))

ss_reg = y @ (H - H1) @ y
ss_res = y @ (np.eye(N) - H) @ y
print(f"SS_Reg = {ss_reg:.3f}")
print(f"SS_Res = {ss_res:.3f}")
print(f"traces: {np.trace(H - H1):.0f}, "
      f"{np.trace(np.eye(N) - H):.0f}")
```

```
True
SS_Reg = 70.991
SS_Res = 59.006
traces: 3, 16
```

ANOVA tables are identical*

```
tab_d = anova_lm(fit_d)
print(tab_d.round(3))
```

```
                df  sum_sq  mean_sq      F  PR(>F)
C(treatment)   3.0  70.991   23.664  6.417   0.005
Residual      16.0  59.006    3.688    NaN     NaN
```

```
fit_0 = smf.ols("y ~ 1", df).fit() # intercept-only model
tab_c = anova_lm(fit_0, fit_c)
cols = ["ssr", "ss_diff", "F", "Pr(>F)"]
print(tab_c[cols].round(3))
```

```
       ssr  ss_diff      F  Pr(>F)
0  129.998      NaN    NaN     NaN
1   59.006   70.991  6.417   0.005
```

- Without an intercept, `anova_lm` uses $\boldsymbol{y}^\top \boldsymbol{H}\boldsymbol{y}$ on 4 df.
- That tests $\mu_1 = \ldots = \mu_4 = 0$, not equal means.

```
tab_c = anova_lm(fit_c)
print(tab_c.round(3))
```

```
                df    sum_sq   mean_sq        F  PR(>F)
C(treatment)   4.0  9123.504  2280.876  618.479     0.0
Residual      16.0    59.006     3.688      NaN     NaN
```

Summary

	Dummy	Cell means
Coefficients	$\mu_1, \mu_i - \mu_1$	μ_i
Estimates	$\bar{y}_{1\bullet}, \bar{y}_{i\bullet} - \bar{y}_{1\bullet}$	$\bar{y}_{i\bullet}$
$X^\top X$	not diagonal	$\mathrm{diag}(n_i)$
H, $\mathrm{SS}_{\mathrm{Res}}$, fitted values	same	same
`anova_lm(fit)`	tests equal means	tests zero means

- The cell-means model gives every quantity in closed form.
- The dummy model reports differences from the control and their standard errors directly.