Optimal designs

STAT 6160 • Experimental Design • Fall 2026

Siddharth Vishwanath

University of Virginia

Treatment assignment

- **Completely randomized designs** assign treatments at random across experimental units.
- **Blocking** groups similar units and randomizes within each group.
- **Full factorials** run every combination of levels to estimate main effects and interactions.
- **Fractional factorials** save runs by aliasing some of these effects.
- **Nested designs** use replication at the level that receives the treatment, as when classrooms receive curricula.
- **Split plots** randomize factors at different levels, as when recipes go to batches and temperatures go to cakes.

Design goals

Optimal design. Which cook times give a precise estimate of the time effect?

Response surface methodology. Which power and time maximize mean yield?

§1. Optimal designs

Microwave popcorn

How does cook time affect the percentage of usable popcorn?

One fresh batch per run. Use the same microwave, power, and brand.

- Cook time t can range from 120 to 180 seconds.
- The response y is the **percentage of kernels that pop without burning**.
- We have 12 batches.

Which times would you run?

Three ways to choose the times

Design	Cook times (seconds)	Batches
Near the usual time	$135, 141, 147, 153, 159, 165$	Two at each time
At the endpoints	$120, 180$	Six at each time
Uniform random	Draw uniformly from 120 to 180	One per draw

The usual cook time is 150 seconds. Randomize the order after choosing the times.

Which design would give the most precise estimate of the time effect?

A line for popcorn yield

Code cook time as $x_i = (t_i - 150)/30$, so that $-1 \leq x_i \leq 1$.

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i, \qquad \boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \qquad \boldsymbol{X} = \begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}$$

- β_0 is the mean yield at 150 seconds.
- β_1 is the change in mean yield, in percentage points, per additional 30 seconds.

Covariance of $\hat{\beta}$

For a fixed, full-rank X, assume $\mathbb{E}(\epsilon \mid X) = \mathbf{0}$ and $\mathrm{Var}\,(\epsilon \mid X) = \sigma^2 I$.

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

$$\mathrm{Var}\,(\hat{\beta} \mid X) =$$

What determines precision?

$$M = \frac{X^\top X}{n},$$

$$\boxed{\Sigma = \mathrm{Var}\,(\hat{\beta} \mid X) = \sigma^2 (X^\top X)^{-1} = \frac{\sigma^2}{n} M^{-1}}$$

- The true coefficients β set the center of the sampling distribution.
- For a fixed budget n and batch-to-batch noise σ^2, the cook times determine the covariance through M.

The covariance ellipse

For a fixed radius c, the ellipse contains the coefficient pairs $\boldsymbol{b} = (b_0, b_1)^\top$ with

$$(\boldsymbol{b} - \boldsymbol{\beta})^\top \Sigma^{-1} (\boldsymbol{b} - \boldsymbol{\beta}) \leq c^2$$

The eigenvectors of Σ give the axes. Each semiaxis has length $c\sqrt{\lambda_j(\Sigma)}$.

A joint confidence ellipse centered at $\hat{\boldsymbol{\beta}}$ uses the same shape. The range of $b_0 + b_1 x$ over that ellipse gives the **Scheffé band** for mean yield.

Comparing ellipsoids

Fix the model, factor coding, and run budget. Let s_j be the singular values of $\boldsymbol{X}/\sqrt{n}$, with p columns in $\boldsymbol{X}$.

Design	What to shrink	Equivalent criterion for $\boldsymbol{X}$
D	Volume, proportional to $\sqrt{\det(\boldsymbol{M}^{-1})}$	Maximize $\det(\boldsymbol{M}) = \prod_j s_j^2$.
E	Longest semiaxis, proportional to $\sqrt{\lambda_{\max}(\boldsymbol{M}^{-1})}$	Maximize $\lambda_{\min}(\boldsymbol{M}) = \min_j s_j^2$.
A	Mean of squared semiaxis lengths, proportional to $\operatorname{tr}(\boldsymbol{M}^{-1})/p$	Minimize $\frac{1}{p}\sum_j 1/s_j^2$.

Precision at a cook time

For $\boldsymbol{f}(x) = (1, x)^\top$, the fitted mean yield is $\hat{\mu}(x) = \boldsymbol{f}(x)^\top \hat{\boldsymbol{\beta}}$.

$$\operatorname{Var}\{\hat{\mu}(x) \mid \boldsymbol{X}\} = \frac{\sigma^2}{n} \boldsymbol{f}(x)^\top \boldsymbol{M}^{-1} \boldsymbol{f}(x)$$

- G-optimality minimizes the **largest** of these variances over the allowed cook times.
- I-optimality minimizes the **average** of these variances over the cook times of interest.

Cook-time spread

$$M = \begin{pmatrix} 1 & \overline{x} \\ \overline{x} & \overline{x^2} \end{pmatrix}, \qquad \det(M) = \overline{x^2} - \overline{x}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2$$

Where would you put the twelve runs to make this determinant large?

The two design matrices

Each row is one batch, with an intercept and a coded cook time.

$$X_{\text{near}} = \begin{pmatrix} 1 & -0.5 \\ 1 & -0.5 \\ 1 & -0.3 \\ 1 & -0.3 \\ 1 & -0.1 \\ 1 & -0.1 \\ 1 & 0.1 \\ 1 & 0.1 \\ 1 & 0.3 \\ 1 & 0.3 \\ 1 & 0.5 \\ 1 & 0.5 \end{pmatrix} \qquad X_{\text{ends}} = \begin{pmatrix} 1 & -1 \\ 1 & -1 \\ 1 & -1 \\ 1 & -1 \\ 1 & -1 \\ 1 & -1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix}$$

Information and determinants

Both designs have $\bar{x} = 0$. Compute $\boldsymbol{M} = \boldsymbol{X}^\top \boldsymbol{X}/12$.

$$\boldsymbol{M}_{\text{near}} = \begin{pmatrix} 1 & 0 \\ 0 & 7/60 \end{pmatrix}, \qquad \boldsymbol{M}_{\text{ends}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\det(\boldsymbol{M}_{\text{near}}) = 7/60 \approx 0.117, \qquad \det(\boldsymbol{M}_{\text{ends}}) = 1$$

The endpoint design is D-optimal for a line on $[-1, 1]$.

$$\frac{\text{area at the endpoints}}{\text{area near the usual time}} = \sqrt{\frac{7}{60}} \approx 0.34.$$

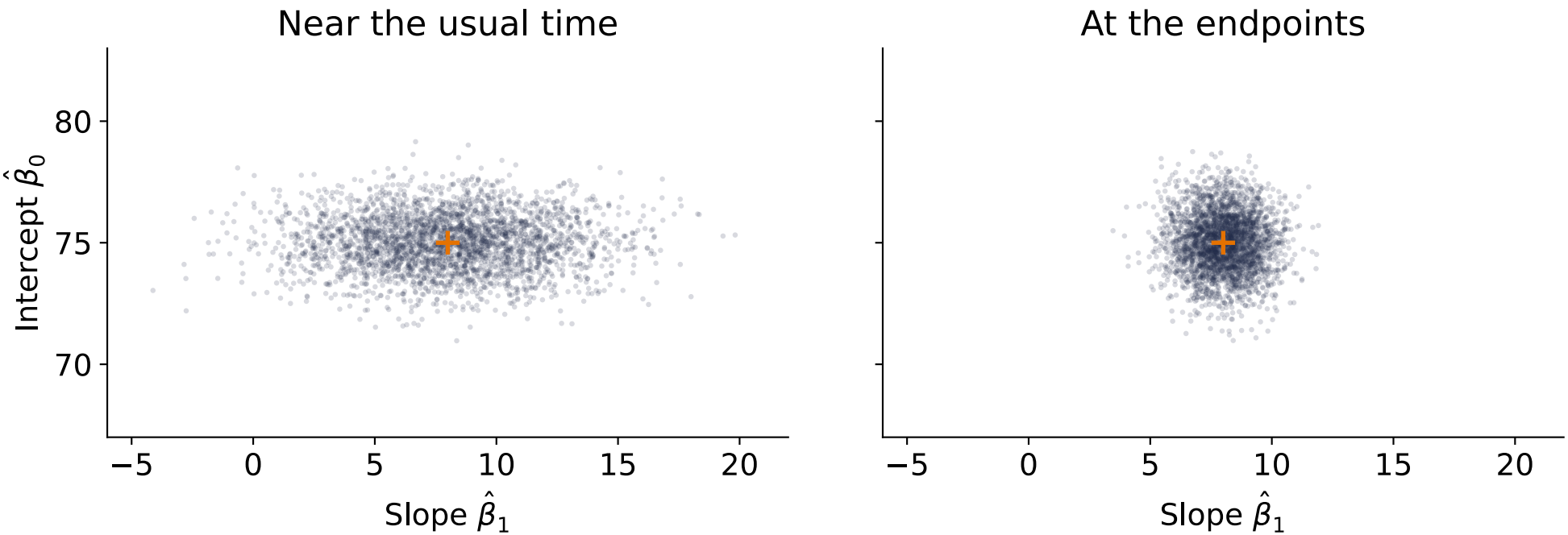
The simulation model

$$y_i = 75 + 8x_i + \epsilon_i, \qquad \epsilon_i \overset{iid}{\sim} \mathcal{N}(0, 4^2)$$

- Simulate 12 batch yields at the chosen times and fit a line.
- Repeat $3,000$ times for each design with the same true coefficients and error variance.

Repeated estimates

One fit per dot. The cross marks the truth. Matrix order (β_0, β_1).

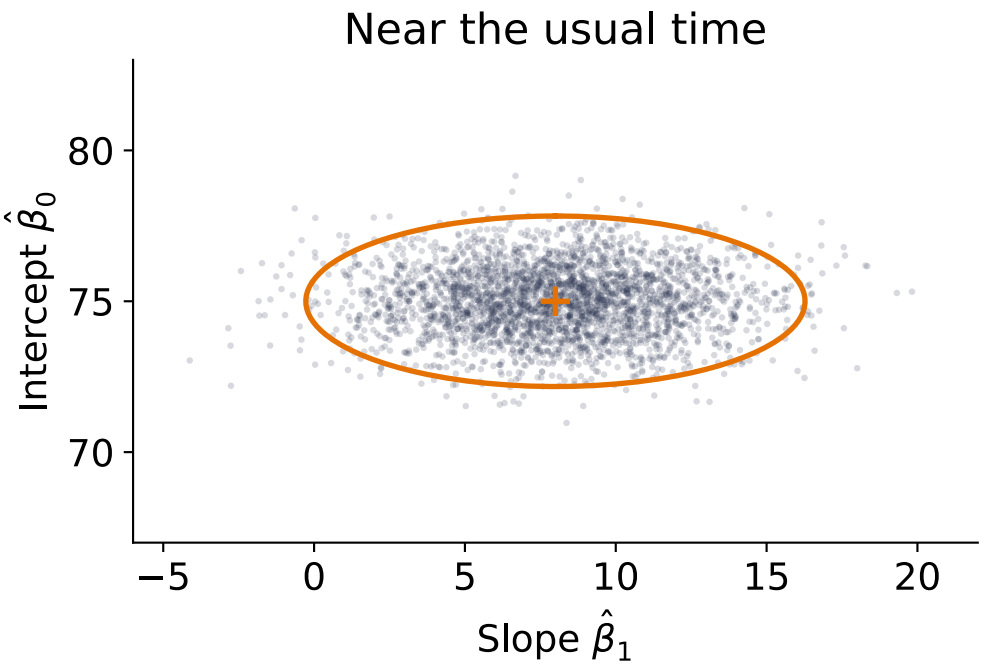


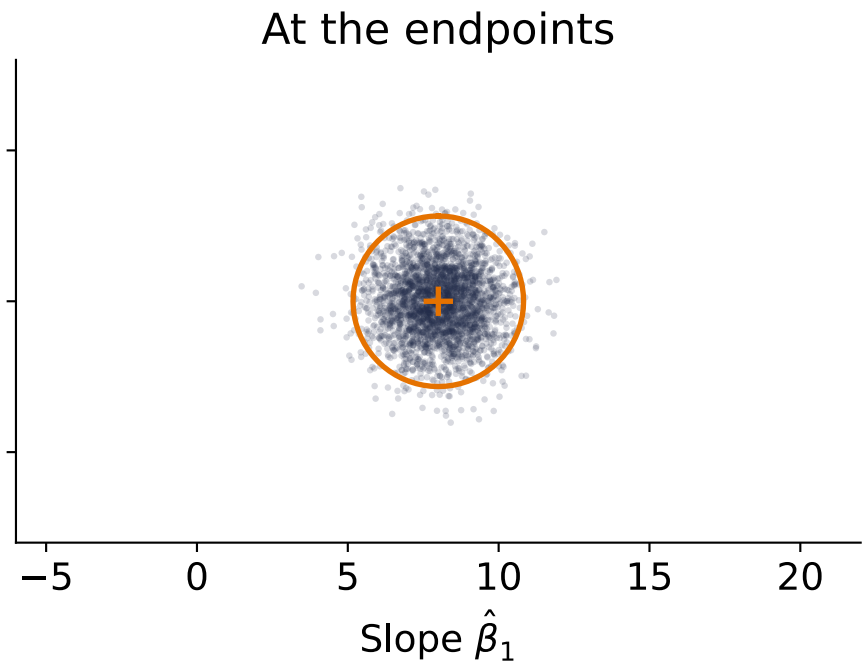
$$M_{\text{near}}^{-1} = \begin{pmatrix} 1 & 0 \\ 0 & 60/7 \end{pmatrix}$$

$$M_{\text{ends}}^{-1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

The covariance ellipses

95% sampling ellipses from the model covariance. Matrix order (β_0, β_1).





$$M_{\text{near}}^{-1} = \begin{pmatrix} 1 & 0 \\ 0 & 60/7 \end{pmatrix}$$

$$M_{\text{ends}}^{-1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

What if the mean bends?

The endpoint design gives a precise line through the two mean yields.

What would a run at 150 seconds let us check?

The G-optimal cook times

For the straight-line model, let $s_x^2 = \det(\boldsymbol{M})$. Divide the fitted-mean variance by σ^2/n to compare designs.

$$v(x) = \boldsymbol{f}(x)^\top \boldsymbol{M}^{-1} \boldsymbol{f}(x) = 1 + \frac{(x - \bar{x})^2}{s_x^2}.$$

$$G = \max_{x \in [-1,1]} v(x) = 1 + \frac{(1 + |\,\bar{x}\,|)^2}{s_x^2} \geq 2, \qquad s_x^2 \leq 1.$$

Near the usual time	At the endpoints
$G = 67/7 \approx 9.57$	$G = 2$

Six batches at each endpoint attain the minimum $G = 2$.

The I-optimal cook times

Average $v(x)$ uniformly over the cook times from 120 to 180 seconds.

$$I = \frac{1}{2}\int_{-1}^{1} v(x)\,dx = 1 + \frac{1/3 + \bar{x}^2}{s_x^2} \geq \frac{4}{3}.$$

Near the usual time	At the endpoints
$I = 27/7 \approx 3.86$	$I = 4/3 \approx 1.33$

Six batches at each endpoint also attain the minimum $I = 4/3$.

Three factors for popcorn yield

For each illustrative range, subtract the midpoint and divide by half the range. The low, middle, and high settings map to -1, 0, and $+1$.

- Cook time t (120–180 s) gives $x_1 = (t - 150)/30$.
- Power P (400–800 W) gives $x_2 = (P - 600)/200$.
- Batch mass m (20–100 g) gives $x_3 = (m - 60)/40$.

Each plan runs all eight settings $(\pm h_1, \pm h_2, \pm h_3)$. Each row of $\mathbf{X}$ is $(1, x_1, x_2, x_3)$.

Plan	Time h_1	Power h_2	Batch size h_3
1	1	1	0.5
2	0.7	0.7	0.7
3	1	0.75	0.625

Which plan would you choose if the least precise coefficient matters most?

Three plans in Python

Compute M, $D = \det(M)$, $E = \lambda_{\max}(M^{-1})$, and $A = \operatorname{tr}(M^{-1})/4$.

```python
import numpy as np
from itertools import product

signs = np.array(list(product((-1, 1), repeat=3)))
widths = [(1, 1, .5), (.7, .7, .7), (1, .75, .625)]
for j, h in enumerate(widths, start=1):
    X = np.column_stack((np.ones(8), signs * h))
    M = X.T @ X / len(X)
    eig = np.linalg.eigvalsh(np.linalg.inv(M))
    D = np.linalg.det(M)
    E = eig.max()
    A = eig.mean()
    print(f"M{j} =\n{M.round(3)}")
    print(f"D {D:.3f}    E {E:.3f}    A {A:.3f}")
```

```
M1 =
[[1.   0.   0.   0.  ]
 [0.   1.   0.   0.  ]
 [0.   0.   1.   0.  ]
 [0.   0.   0.   0.25]]
D 0.250    E 4.000    A 1.750
M2 =
[[1.   0.   0.   0.  ]
 [0.   0.49 0.   0.  ]
 [0.   0.   0.49 0.  ]
 [0.   0.   0.   0.49]]
D 0.118    E 2.041    A 1.781
M3 =
[[1.    0.    0.    0.   ]
 [0.    1.    0.    0.   ]
 [0.    0.    0.562 0.   ]
 [0.    0.    0.    0.391]]
D 0.220    E 2.560    A 1.584
```

Among these plans, D chooses **1**, E chooses **2**, and A chooses **3**.

Reading

- Owen, *A First Course in Experimental Design*, §12.6, pp. 132–134.
- Montgomery, *Design and Analysis of Experiments*, 8th ed., §9.6.1, p. 433, and §11.4, pp. 513–514.
- Wu and Hamada, *Experiments: Planning, Analysis, and Optimization*, 3rd ed., §13.2, p. 590.

Appendix

Simulating the batch yields

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import chi2

rng = np.random.default_rng(6160)
beta = np.array([75, 8])
sigma, repeats = 4, 3000
near = np.repeat(np.arange(135, 166, 6), 2)
ends = np.repeat([120, 180], 6)

simulations = []
for name, seconds in [("Near the usual time", near), ("At the endpoints", ends)]:
    x = (seconds - 150) / 30
    X = np.column_stack([np.ones_like(x), x])
    errors = rng.normal(0, sigma, (len(x), repeats))
    y = (X @ beta)[:, None] + errors
    beta_hat = np.linalg.lstsq(X, y, rcond=None)[0]
    covariance = sigma**2 * np.linalg.inv(X.T @ X)
    simulations.append((name, beta_hat, covariance))

plt.rcParams.update({"font.size": 14, "axes.titlesize": 16})
blue, orange = "#232D4B", "#E57200"
```

Plotting the estimates

```
theta = np.linspace(0, 2 * np.pi, 361)
circle = np.array([np.cos(theta), np.sin(theta)])
radius = np.sqrt(chi2.ppf(0.95, df=2))

def plot_estimates(overlay=False):
    fig, axes = plt.subplots(1, 2, figsize=(10.8, 3.4),
                             sharex=True, sharey=True, layout="constrained")
    for ax, (name, beta_hat, covariance) in zip(axes, simulations):
        ax.scatter(beta_hat[1], beta_hat[0], s=6, color=blue,
                   alpha=0.18, linewidths=0)
        if overlay:
            L = np.linalg.cholesky(covariance)
            ellipse = beta[:, None] + radius * (L @ circle)
            ax.plot(ellipse[1], ellipse[0], color=orange, lw=2)
        ax.plot(beta[1], beta[0], "+", color=orange, ms=11, mew=2)
        ax.set(title=name, xlabel=r"Slope $\hat{\beta}_1$",
               xlim=(-6, 22), ylim=(67, 83), aspect="equal")
        ax.spines[["top", "right"]].set_visible(False)
    axes[0].set_ylabel(r"Intercept $\hat{\beta}_0$")
    plt.show()

plot_estimates()
plot_estimates(overlay=True)
```