# PS1

September 14, 2025

Note: Write your code in the code cells, and your responses in markdown. Run the entire script and display the outputs of your code.

Due: **11:59PM Central Time on Friday, 09/12**. Upload both your code (.ipynb) and responses (html or pdf) to Canvas by then.

## Prep

Please make sure you have installed Python (3.10+) and jupyter notebook before you start the assignment. - e.g., pip install jupyter notebook - Type "jupyter notebook" or "python -m jupyter notebook" in command prompt (Terminal on Mac, Powershell on Windows): this should generate a link for you to open in the browser and create a Jupyter notebook. - Alternatively, you can also use Jupyter in IDE like Visual Studio, Cursor (with AI chats), or Google Colab.

Jupyter/Colab allow you to run the entire script and display the outputs (including figures or tables). Make sure to upload both the original notebook (.ipynb) and the output in HTML/PDF format: - HTML: HTML first. Use "Export" in Jupyter/Colab or in bash:(py -m) *jupyter nbconvert PS1.ipynb –to html* to save a HTML. You may print the HTML as PDF. - PDF: Export the notebook as PDF directly via *jupyter nbconvert PS1.ipynb –to pdf*. This requires you have installed LaTex, Pandoc, and maybe nb_pdf_template to make the PDF look like the original notebook.

I recommend writing the code on your own, or at least a pseudocode before you ask for help from AI assistants. Please complete the Disclaimer on any use of AI at the end of the problem set.

[ ]:
```
# Packages you might need: (pip install ... if you don't have them)
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt

print(pd.__version__)
print(np.__version__)
print(matplotlib.__version__)
```

## 1 Refresher

Consider a random variable $X$ with (population) mean $\mu$ and variance $\sigma^2$. $(x_1, x_2, ..., x_n)$ is a random sample of $X$ from the population, in which the data points are independent and identically distributed (iid).

1. Prove the following: (a) $E[(x_i - \mu)x_i] = \sigma^2$, and (b) $E[x_i^2] - E[x_i]^2 = \sigma^2$.

### 1.0.1 Solution 1(a)

We want to show:

$$\mathbb{E}[(x_i - \mu)x_i] = \sigma^2$$

By linearity of expectation:

$$\mathbb{E}[(x_i - \mu)x_i] = \mathbb{E}[x_i^2] - \mu\,\mathbb{E}[x_i]$$

Since $\mathbb{E}[x_i] = \mu$, this becomes:

$$\mathbb{E}[x_i^2] - \mu^2$$

But by the definition of variance:

$$\sigma^2 = \mathbb{E}[x_i^2] - (\mathbb{E}[x_i])^2$$

Therefore:

$$\mathbb{E}[(x_i - \mu)x_i] = \sigma^2$$

### 1.0.2 Solution 1(b)

We want to show:

$$\mathbb{E}[x_i^2] - \big(\mathbb{E}[x_i]\big)^2 = \sigma^2$$

Since $x_i \overset{d}{=} X$ with $\mathbb{E}[x_i] = \mu$ and $\mathrm{Var}(x_i) = \sigma^2$:

$$\mathrm{Var}(x_i) = \mathbb{E}\big[(x_i - \mu)^2\big] = \mathbb{E}[x_i^2] - 2\mu\,\mathbb{E}[x_i] + \mu^2$$

But since $\mathbb{E}[x_i] = \mu$:

$$\mathrm{Var}(x_i) = \mathbb{E}[x_i^2] - \mu^2$$

Hence:

$$\mathbb{E}[x_i^2] - \big(\mathbb{E}[x_i]\big)^2 = \sigma^2$$

b

2. Define the sample variance as $s_n = \frac{1}{n-1}\sum_i (x_i - \bar{x})^2$, where $\bar{x} = \frac{1}{n}\sum_i x_i$. Show $s_n$ is an unbiased estimator for population variance, $E[s_n] = \sigma^2$.

### 1.0.3 Solution 2:

We want to show that the sample variance

$$s_n = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2, \qquad \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

is an unbiased estimator of the population variance, i.e.,

$$\mathbb{E}[s_n] = \sigma^2.$$

$$s_n = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2.$$

We write

$$x_i - \bar{x} = (x_i - \mu) - (\bar{x} - \mu).$$

So:

$$s_n = \frac{1}{n-1}\sum_{i=1}^{n}\Big((x_i - \mu) - (\bar{x} - \mu)\Big)^2.$$

$$\begin{aligned} s_n &= \frac{1}{n-1}\sum_{i=1}^{n}\Big[(x_i - \mu)^2 - 2(x_i - \mu)(\bar{x} - \mu) + (\bar{x} - \mu)^2\Big] \\ &= \frac{1}{n-1}\left[\sum_{i=1}^{n}(x_i - \mu)^2 - 2(\bar{x} - \mu)\sum_{i=1}^{n}(x_i - \mu) + \sum_{i=1}^{n}(\bar{x} - \mu)^2\right]. \end{aligned}$$

- Note that
$$\sum_{i=1}^{n}(x_i - \mu) = n(\bar{x} - \mu).$$

- Therefore:
$$-2(\bar{x} - \mu)\sum_{i=1}^{n}(x_i - \mu) = -2(\bar{x} - \mu)\, n(\bar{x} - \mu) = -2n(\bar{x} - \mu)^2.$$

- Also,
$$\sum_{i=1}^{n}(\bar{x} - \mu)^2 = n(\bar{x} - \mu)^2.$$

So the whole expression becomes:

$$s_n = \frac{1}{n-1}\left[\sum_{i=1}^{n}(x_i - \mu)^2 - n(\bar{x} - \mu)^2\right].$$

Now take the expectation of both sides:

$$\mathbb{E}[s_n] = \frac{1}{n-1}\left[\sum_{i=1}^{n}\mathbb{E}[(x_i - \mu)^2] - n\,\mathbb{E}[(\bar{x} - \mu)^2]\right].$$

Since $x_i \overset{iid}{\sim} X$ with variance $\mathrm{Var}(X) = \sigma^2$,

$$\mathbb{E}[(x_i - \mu)^2] = \sigma^2.$$

Therefore:

$$\sum_{i=1}^{n} \mathbb{E}[(x_i - \mu)^2] = n\sigma^2.$$

Because $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ and the $x_i$ are i.i.d.,

$$\mathrm{Var}(\bar{x}) = \frac{1}{n^2}\sum_{i=1}^{n} \mathrm{Var}(x_i) = \frac{1}{n^2}\, n\sigma^2 = \frac{\sigma^2}{n}.$$

Hence:

$$\mathbb{E}[(\bar{x} - \mu)^2] = \mathrm{Var}(\bar{x}) = \frac{\sigma^2}{n}.$$

$$\mathbb{E}[s_n] = \frac{1}{n-1}\left(n\sigma^2 - n \cdot \frac{\sigma^2}{n}\right).$$

Simplify:

$$\mathbb{E}[s_n] = \frac{1}{n-1}\left(n\sigma^2 - \sigma^2\right) = \frac{(n-1)\sigma^2}{n-1} = \sigma^2.$$

**Conclusion:** The sample variance $s_n$ is an unbiased estimator of the population variance:

$$\mathbb{E}[s_n] = \sigma^2.$$

3. Suppose we run a regression of $x_i$ on a constant 1, $x_i = \beta * 1 + \epsilon_i$. What is $\beta$ in the population regression, and why? What is the OLS coefficient $\hat{\beta}$ in the sample regression?

### 1.0.4 Solution 3:

We observe i.i.d. draws $(x_1, \ldots, x_n)$ from a random variable $X$ with mean $\mu = \mathbb{E}[X]$ and variance $\sigma^2 = \mathrm{Var}(X)$.
Consider the regression:

$$x_i = \beta \cdot 1 + \varepsilon_i.$$

The population coefficient $\beta$ is the constant $b$ that minimizes expected squared error:

$$\beta = \arg\min_{b \in \mathbb{R}} \ \mathbb{E}\big[(X - b)^2\big].$$

Differentiating the objective and setting the derivative equal to zero:

$$\frac{d}{db}\,\mathbb{E}\big[(X-b)^2\big] = \mathbb{E}[-2(X-b)] = -2(\mathbb{E}[X] - b) = 0 \implies b = \mathbb{E}[X].$$

Hence,

$$\beta = \mathbb{E}[X] = \mu.$$

That is, the best horizontal line in the population is the population mean.

*(Alternative view: with a constant regressor, the population normal equation is* $\mathbb{E}[x_i - \beta] = 0 \Rightarrow \beta = \mathbb{E}[x_i]$*.)*

The OLS estimator minimizes the sample average squared error:

$$\hat{\beta} = \arg\min_{b \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} (x_i - b)^2.$$

First-order condition:

$$\frac{\partial}{\partial b}\Big(\frac{1}{n} \sum_{i=1}^{n} (x_i - b)^2\Big) = -2\Big(\frac{1}{n} \sum_{i=1}^{n} x_i - b\Big) = 0.$$

Therefore,

$$\hat{\beta} = \bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i.$$

Thus: - In the population, $\beta = \mu = \mathbb{E}[X]$.
- In the sample, $\hat{\beta} = \bar{x}$.

## 2 Simulation - Bernoulli

Set up an Python program to conduct a simulation of drawing a sample size of $n$ from a Bernoulli distribution with mean $p$. In each "replication" $r$ you will draw a sample, construct the estimated mean from that replication (which we will denote as $\overline{Y}_r$) , and calculate the 95% confidence interval $(\overline{Y}_r - 1.96 * s_r/\sqrt{n}, \overline{Y}_r - 1.96 * s_r/\sqrt{n})$ where $s_r$ is the estimated standard deviation in that replication, $s_r = \sqrt{\overline{Y}_r * (1 - \overline{Y}_r)}$. In each replication, record the length of the confidence interval, and whether or not the true mean is inside the interval.

For given $(n, p)$, conduct $1,000$ replications are report the following statistics:

1. the mean estimate of $p$;
   - plot the distribution of $\overline{Y}_r$ (sample estimates of $p$) across replications.
2. the mean estimate of the true standard deviation.
3. the fraction of time that the confidence interval contains the true $p$. This is called the *coverage rate.*

Conduct the analysis for the cases $n = 30$ using $p = 0.05$ and $p = 0.25$, and again for $n = 60$ using $p = 0.05$ and $p = 0.25$ (a total of 4 cases). It is often claimed that $n$ of 30 or larger is enough to insure that asymptotic confidence intervals work well. Do you agree or not?

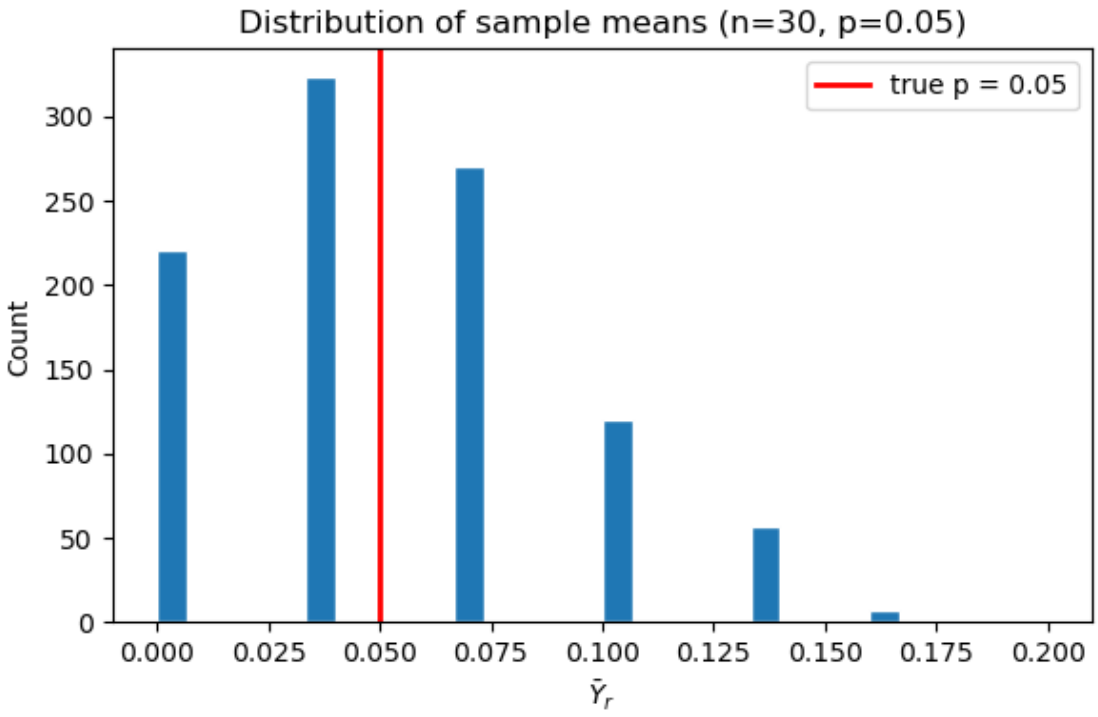
[5]:
```
# Simulation & reporting for the Bernoulli problem
```

```python
# Importing packages again just as a sanity check
import numpy as np
import matplotlib.pyplot as plt

def bernoulli_simulation_full(n, p, R=1000, seed=42, clip_ci=False):
    """
    Run R replications of Bernoulli(n, p) sampling and compute:
    - sample mean
    - sample SD (unbiased, ddof=1)
    - 95% normal CI for the mean
    - indicator if CI covers the true p
    - CI length

    clip_ci: if True, clip CI bounds to [0, 1].
    """
    rng = np.random.default_rng(seed)

    sample_means     = np.zeros(R)
    sample_sds       = np.zeros(R)
    ci_lower         = np.zeros(R)
    ci_upper         = np.zeros(R)
    contains_true_p  = np.zeros(R, dtype=bool)
    ci_length        = np.zeros(R)

    for r in range(R):
        # 1) Draw Bernoulli sample
        sample = rng.binomial(1, p, size=n)

        # 2) Sample mean and sd
        xbar = np.mean(sample)
        s    = np.std(sample, ddof=1)

        sample_means[r] = xbar
        sample_sds[r]   = s

        # 3) 95% normal CI for the mean
        se = s / np.sqrt(n)
        lo = xbar - 1.96 * se
        hi = xbar + 1.96 * se

        if clip_ci:
            lo = max(0.0, lo)
            hi = min(1.0, hi)

        ci_lower[r] = lo
        ci_upper[r] = hi
        ci_length[r] = hi - lo
```

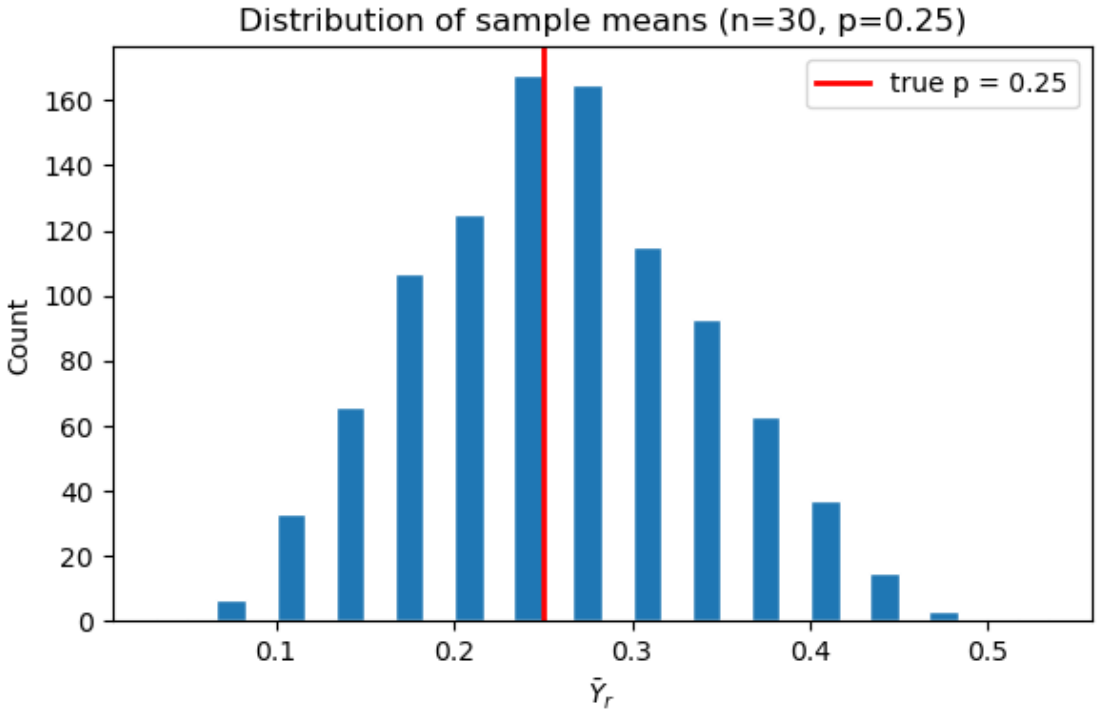
```
        # 4) Coverage
        contains_true_p[r] = (lo <= p <= hi)

    return {
        "sample_means":    sample_means,
        "sample_sds":      sample_sds,
        "ci_lower":        ci_lower,
        "ci_upper":        ci_upper,
        "ci_length":       ci_length,
        "coverage":        contains_true_p
    }

def summarize_and_plot(sim, p, n, show_plot=True, bins=30):
    """
    Print summary stats and (optionally) plot the distribution of sample means.
    Returns a dict with scalars you can collect across scenarios.
    """
    sample_means = sim["sample_means"]
    sample_sds   = sim["sample_sds"]
    coverage     = sim["coverage"]
    ci_length    = sim["ci_length"]

    mean_of_means   = np.mean(sample_means)      # estimator of p
    mean_of_sds     = np.mean(sample_sds)        # estimator of true SD
    coverage_rate   = np.mean(coverage)          # fraction of CIs containing p
    avg_ci_length   = np.mean(ci_length)

    print(f"(n={n}, p={p:.2f})")
    print(f"  Mean of sample means (est. p):     {mean_of_means:.4f}")
    print(f"  Mean of sample SDs (est. sd):      {mean_of_sds:.4f}   (true sd =␣
 ↪sqrt(p*(1-p)) = {np.sqrt(p*(1-p)):.4f})")
    print(f"  Coverage rate (95% normal CI):      {coverage_rate:.4f}")
    print(f"  Average CI length:                  {avg_ci_length:.4f}")
    print("-"*60)

    if show_plot:
        plt.figure(figsize=(6,4))
        plt.hist(sample_means, bins=bins, edgecolor='white')
        plt.axvline(p, color='red', linewidth=2, label=f"true p = {p:.2f}")
        plt.title(f"Distribution of sample means (n={n}, p={p:.2f})")
        plt.xlabel(r"$\bar{Y}_r$")
        plt.ylabel("Count")
        plt.legend()
        plt.tight_layout()
        plt.show()
```

```
    return {
        "n": n,
        "p": p,
        "mean_of_means": mean_of_means,
        "mean_of_sds": mean_of_sds,
        "true_sd": np.sqrt(p*(1-p)),
        "coverage_rate": coverage_rate,
        "avg_ci_length": avg_ci_length
    }

# Run simulations for the four scenarios
scenarios = [(30, 0.05), (30, 0.25), (60, 0.05), (60, 0.25)]

summaries = []
for (n, p) in scenarios:
    sim = bernoulli_simulation_full(n=n, p=p, R=1000, seed=42, clip_ci=False)
    summary = summarize_and_plot(sim, p=p, n=n, show_plot=True, bins=30)
    summaries.append(summary)
```

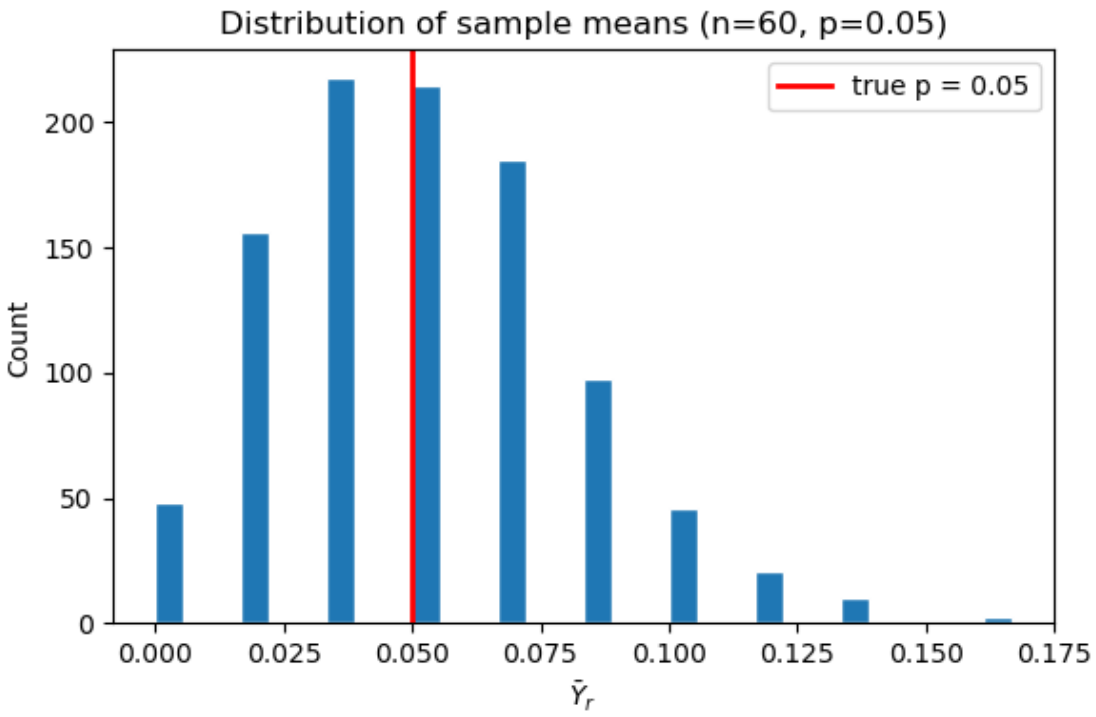
```
(n=30, p=0.05)
  Mean of sample means (est. p):     0.0498
  Mean of sample SDs (est. sd):      0.1870   (true sd = sqrt(p*(1-p)) = 0.2179)
  Coverage rate (95% normal CI):      0.7780
  Average CI length:                  0.1339
----------------------------------------------------------------
```



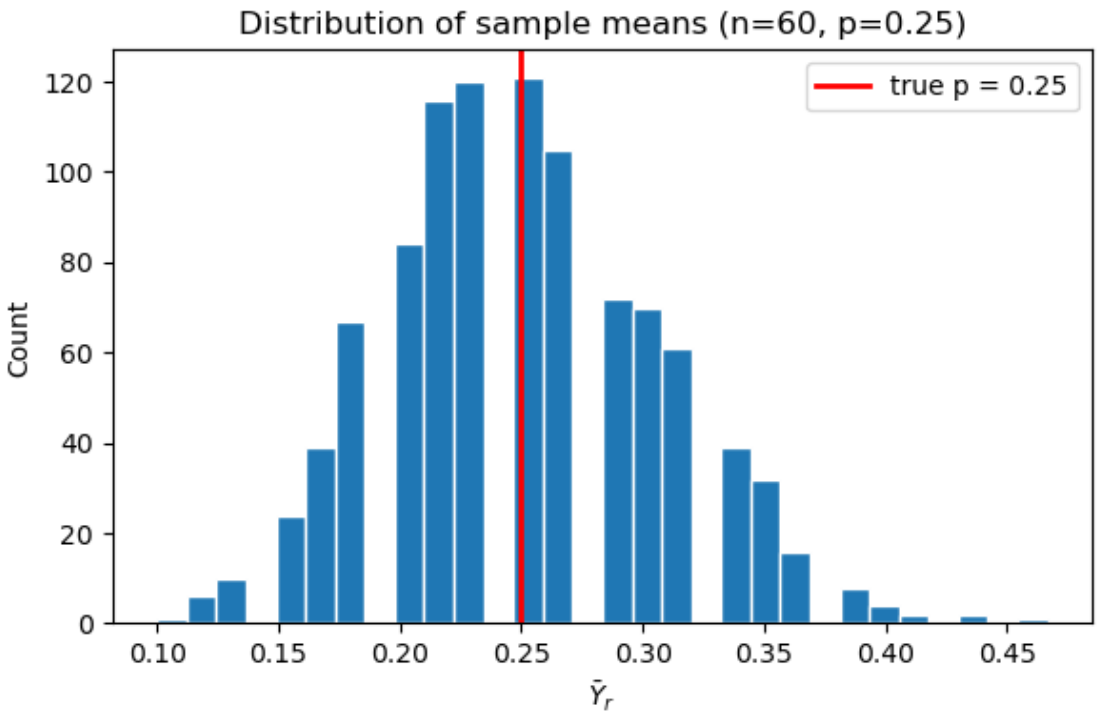
```
(n=30, p=0.25)
  Mean of sample means (est. p):     0.2510
  Mean of sample SDs (est. sd):      0.4303   (true sd = sqrt(p*(1-p)) = 0.4330)
  Coverage rate (95% normal CI):      0.9390
  Average CI length:                  0.3080
------------------------------------------------------------
```




```
(n=60, p=0.05)
  Mean of sample means (est. p):     0.0500
  Mean of sample SDs (est. sd):      0.2060   (true sd = sqrt(p*(1-p)) = 0.2179)
  Coverage rate (95% normal CI):      0.7930
  Average CI length:                  0.1042
------------------------------------------------------------
```




```
(n=60, p=0.25)
  Mean of sample means (est. p):     0.2503
  Mean of sample SDs (est. sd):      0.4318   (true sd = sqrt(p*(1-p)) = 0.4330)
  Coverage rate (95% normal CI):      0.9420
  Average CI length:                  0.2185
------------------------------------------------------------
```




### 2.0.1 Discussion: Bernoulli Simulation and Confidence Interval Coverage

We simulated drawing $n$ observations from a Bernoulli($p$) distribution, repeated $R = 1000$ times. In each replication $r$:

1. Draw $X_1, \ldots, X_n \overset{iid}{\sim} \text{Bernoulli}(p)$.
2. Compute the sample mean

$$\bar{Y}_r = \frac{1}{n} \sum_{i=1}^{n} X_i$$

   and the sample standard deviation $s_r$ (with $ddof = 1$).
3. Construct the **Wald 95% confidence interval**:

$$\bar{Y}_r \pm 1.96 \cdot \frac{s_r}{\sqrt{n}}.$$

4. Record the CI length

$$L_r = 2 \cdot 1.96 \cdot \frac{s_r}{\sqrt{n}},$$

   and whether the interval contains the true $p$ (coverage indicator).

From the $R$ replications, we report:

- Mean of the $\bar{Y}_r$ (estimate of $p$),
- Mean of the $s_r$ (estimate of $\sqrt{p(1-p)}$),

- Coverage rate = fraction of CIs containing $p$,
- Average CI length $\bar{L} = \frac{1}{R}\sum_{r=1}^{R} L_r$.

#### 2.0.2 Results

- $(n = 30, p = 0.05)$
  Mean $\bar{Y}$: 0.0498
  Mean $s$: 0.1870 (true $\sqrt{p(1-p)} = 0.2179$)
  Coverage: 0.7780
  Avg CI length: 0.1339
- $(n = 30, p = 0.25)$
  Mean $\bar{Y}$: 0.2510
  Mean $s$: 0.4303 (true = 0.4330)
  Coverage: 0.9390
  Avg CI length: 0.3080
- $(n = 60, p = 0.05)$
  Mean $\bar{Y}$: 0.0500
  Mean $s$: 0.2060 (true = 0.2179)
  Coverage: 0.7930
  Avg CI length: 0.1042
- $(n = 60, p = 0.25)$
  Mean $\bar{Y}$: 0.2503
  Mean $s$: 0.4318 (true = 0.4330)
  Coverage: 0.9420
  Avg CI length: 0.2185

#### 2.0.3 Interpretation

- The sample mean $\bar{Y}$ is an **unbiased estimator** of $p$. Across all cases, the average of $\bar{Y}_r$ is relatively close to the true $p$.
- The estimated standard deviations $s_r$ are close to the theoretical $\sqrt{p(1-p)}$, especially when $p = 0.25$. For rare events ($p = 0.05$), $s_r$ tends to be underestimated.
- The **coverage rate** of the Wald 95% CI:
  - For $p = 0.25$, coverage is close to 95%: about 0.94 at both $n = 30$ and $n = 60$.
  - For $p = 0.05$, coverage drops badly to around 0.78–0.79, even at $n = 60$.
    This shows the Wald CI is too narrow and under-covers when $p$ is small.

**So TL;DR:** The rule-of-thumb that "$n \geq 30$ is enough for asymptotic confidence intervals" is **not reliable in all cases**:

- For moderate $p$ (e.g. 0.25), the Wald CI performs well even at $n = 30$.
- For small $p$ (e.g. 0.05), the Wald CI fails — coverage is far below 95%, even when $n = 60$.

The asymptotic confidence interval works well when $p$ is moderate, but not when $p$ is near 0 (rare

events). In such cases, alternatives like Wilson, Agresti–Coull, or exact (Clopper–Pearson) intervals are recommended.

## 3 Data analysis: Age Profile and Gender Gap in the Use of Cell Phone

Download the dataset "october_cps", which is an extract of data from the October 2012 CPS.

There are several variables describing each person in the survey, including age, sex, "educ" (which is education, coded in a certain way) as well as variables about how someone uses their cell phone (if they have one): cell_use_phone, cell_use_msg, cell_use_video, cell_use_browse_web, cell_use_email, cell_use_games, cell_use_social_media, cell_use_download_apps, and cell_use_music. Each of these is coded "1" if the person uses the cell phone for that use, "2" if not, and (as a phone; for text messages, for video, to browse the web, for email for games, to access social media, to download apps, or to listen to music).

1. Develop a graph to show how people of different ages use their cell phone. Be creative.
2. In your sample, test whether males (sex=1) and females (sex=2) use their cell phone at the same rate for each of the 9 different uses.

[14]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Load the dataset (CSV is in the same folder as this notebook)
CSV_PATH = "october_cps.csv"
df = pd.read_csv(CSV_PATH)

# We'll focus on these columns that describe different ways people use their␣
 ↪cell phones.
# In the data, a 1 means "yes" (uses for that purpose), and a 2 means "no".
use_cols = [
    "cell_use_phone_calls",
    "cell_use_text_msg",
    "cell_use_photo_video",
    "cell_use_browse_web",
    "cell_use_email",
    "cell_use_games",
    "cell_use_social_media",
    "cell_use_download_apps",
    "cell_use_music",
]

# Quick check that all the expected columns are actually in the dataset
missing = [c for c in use_cols + ["age"] if c not in df.columns]
if missing:
    raise ValueError(f"Missing columns in the CSV: {missing}")
```

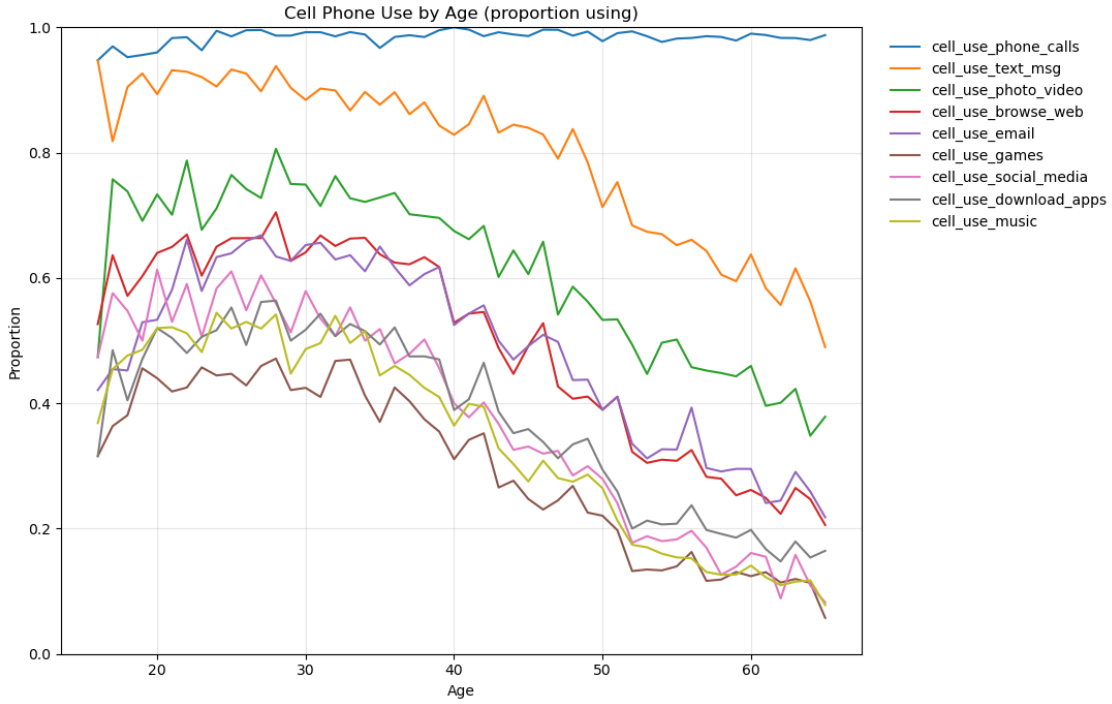
```
# Make sure "age" is numeric, drop rows with missing ages,
# and restrict to plausible ages between 10 and 90
df["age"] = pd.to_numeric(df["age"], errors="coerce")
df = df.dropna(subset=["age"])
df = df[(df["age"] >= 10) & (df["age"] <= 90)]
df["age"] = df["age"].astype(int)

# Recode the phone use variables: 1 becomes 1 (yes), 2 becomes 0 (no).
# Anything else is treated as missing and filled with 0.
def recode_1_yes_2_no(x):
    if x == 1:
        return 1
    if x == 2:
        return 0
    return np.nan

df[use_cols] = df[use_cols].applymap(recode_1_yes_2_no).fillna(0).astype(int)

# ---------------------------------------------------------------------
# Part 1: Look at how cell phone use varies by age.
# For each age, we compute the proportion of people who use their phone
# for each purpose, then plot the results so we can see the patterns.
# ---------------------------------------------------------------------

# For each age, compute the proportion of people who use their phone for each␣
 ↪purpose
age_usage = df.groupby("age", as_index=True)[use_cols].mean().sort_index()

print("Proportion using by age (first few rows):")
display(age_usage.head())

# Plot the usage patterns by age
plt.figure(figsize=(11, 7))
for col in use_cols:
    plt.plot(age_usage.index, age_usage[col], label=col)

plt.title("Cell Phone Use by Age (proportion using)")
plt.xlabel("Age")
plt.ylabel("Proportion")
plt.ylim(0, 1)
plt.grid(alpha=0.3)
plt.legend(bbox_to_anchor=(1.02, 1), loc="upper left", frameon=False)
plt.tight_layout()
plt.show()

# ---------------------------------------------------------------------
# Part 2: Compare males (sex=1) and females (sex=2).
```

```python
# We'll test whether they use their phones at the same rate
# for each type of use with two-proportion z-tests.
# ---------------------------------------------------------------------
from math import sqrt, erf

def two_prop_ztest(x1, n1, x2, n2):
    """Return (z, p_two_sided) for H0: p1 == p2 using pooled-variance z test."""
    if n1 == 0 or n2 == 0:
        return float('nan'), float('nan')
    p1 = x1 / n1
    p2 = x2 / n2
    p_pool = (x1 + x2) / (n1 + n2)
    se = sqrt(p_pool * (1 - p_pool) * (1/n1 + 1/n2))
    if se == 0:
        return 0.0, 1.0
    z = (p1 - p2) / se
    def norm_cdf(t): return 0.5 * (1.0 + erf(t / sqrt(2.0)))  # Normal CDF via
 ↪erf
    p_two = 2.0 * (1.0 - norm_cdf(abs(z)))
    return z, p_two

# Keep only rows with valid sex codes (1 = male, 2 = female)
df_sex = df[df['sex'].isin([1, 2])].copy()

# Run the z-test for each type of phone use
rows = []
for col in use_cols:
    male = df_sex.loc[df_sex['sex'] == 1, col]
    fem  = df_sex.loc[df_sex['sex'] == 2, col]

    n_m = male.notna().sum()
    n_f = fem.notna().sum()
    x_m = male.fillna(0).sum()
    x_f = fem.fillna(0).sum()

    z, pval = two_prop_ztest(x_m, n_m, x_f, n_f)

    rate_m = x_m / n_m if n_m else float('nan')
    rate_f = x_f / n_f if n_f else float('nan')

    rows.append({
        "use": col,
        "n_male": n_m,   "rate_male": rate_m,
        "n_fem":  n_f,   "rate_fem":  rate_f,
        "diff (m - f)": rate_m - rate_f,
        "z_stat": z,
        "p_value": pval,
```

```
        "reject_5%": (pval < 0.05)
    })

test_table = pd.DataFrame(rows).sort_values("use").reset_index(drop=True)
display(test_table)

# Bar plot to compare male vs female usage rates
plt.figure(figsize=(10, 5))
x = np.arange(len(use_cols))
bar_w = 0.38
plt.bar(x - bar_w/2, test_table["rate_male"], width=bar_w, label="Male",␣
 ↪alpha=0.9)
plt.bar(x + bar_w/2, test_table["rate_fem"],  width=bar_w, label="Female",␣
 ↪alpha=0.9)
plt.xticks(x, test_table["use"], rotation=45, ha="right")
plt.ylim(0, 1)
plt.ylabel("Proportion using")
plt.title("Phone-use rates by sex (with two-proportion z-tests run per␣
 ↪feature)")
plt.legend()
plt.tight_layout()
plt.show()

# Print a quick summary of which features show statistically significant␣
 ↪differences
sig_uses = test_table.loc[test_table["reject_5%"], "use"].tolist()
print(f"Features where we reject equal use at 5%: {sig_uses if sig_uses else␣
 ↪'None (no differences at 5%)'}")
```

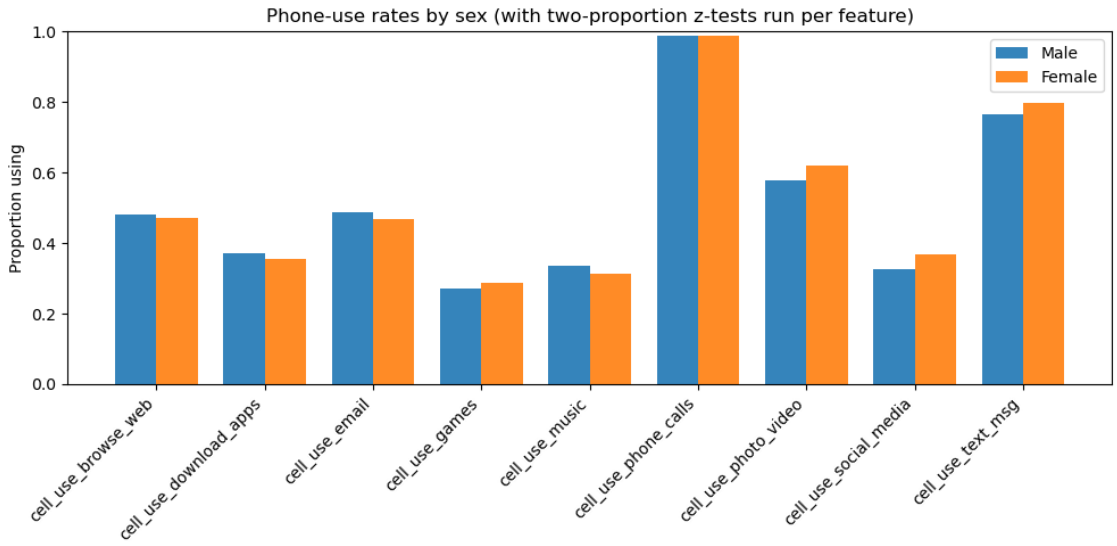
```
Proportion using by age (first few rows):

/var/folders/px/1d1phm911s94712v2c0rv4gr0000gn/T/ipykernel_43488/3063815825.py:4
4: FutureWarning: DataFrame.applymap has been deprecated. Use DataFrame.map
instead.
  df[use_cols] = df[use_cols].applymap(recode_1_yes_2_no).fillna(0).astype(int)

     cell_use_phone_calls  cell_use_text_msg  cell_use_photo_video  \
age
16               0.947368           0.947368              0.473684
17               0.969697           0.818182              0.757576
18               0.952381           0.904762              0.738095
19               0.955882           0.926471              0.691176
20               0.960000           0.893333              0.733333

     cell_use_browse_web  cell_use_email  cell_use_games  \
age
16              0.526316        0.421053        0.315789
17              0.636364        0.454545        0.363636
```

```
18              0.571429        0.452381        0.380952
19              0.602941        0.529412        0.455882
20              0.640000        0.533333        0.440000

     cell_use_social_media  cell_use_download_apps  cell_use_music
age
16                0.473684                0.315789        0.368421
17                0.575758                0.484848        0.454545
18                0.547619                0.404762        0.476190
19                0.500000                0.470588        0.485294
20                0.613333                0.520000        0.520000
```




```
                    use  n_male  rate_male  n_fem  rate_fem  diff (m - f)  \
0     cell_use_browse_web    4993   0.480473   6601  0.470383      0.010089
1  cell_use_download_apps    4993   0.370719   6601  0.353886      0.016833
2          cell_use_email    4993   0.488083   6601  0.466899      0.021184
3          cell_use_games    4993   0.271580   6601  0.288138     -0.016558
4          cell_use_music    4993   0.334869   6601  0.311922      0.022946
5    cell_use_phone_calls    4993   0.986781   6601  0.986669      0.000113
6    cell_use_photo_video    4993   0.577208   6601  0.619452     -0.042244
7   cell_use_social_media    4993   0.326657   6601  0.368278     -0.041620
8       cell_use_text_msg    4993   0.765872   6601  0.797455     -0.031583

     z_stat   p_value  reject_5%
0  1.077257  0.281366      False
```

```
1  1.868517  0.061690      False
2  2.261592  0.023723       True
3 -1.964059  0.049523       True
4  2.618845  0.008823       True
5  0.052537  0.958101      False
6 -4.599946  0.000004       True
7 -4.651376  0.000003       True
8 -4.090977  0.000043       True
```




```
Features where we reject equal use at 5%: ['cell_use_email', 'cell_use_games',
'cell_use_music', 'cell_use_photo_video', 'cell_use_social_media',
'cell_use_text_msg']
```

### 3.0.1 Findings: Age Profile & Gender Gap in Cell Phone Use

**Age trends:**
Cell phone use varies substantially with age. Younger individuals show higher usage across nearly all categories (texting, browsing, social media, etc.), while usage declines steadily with age. Phone calls remain consistently high across ages.

**Gender differences:**
Using two-proportion z-tests, we find statistically significant differences (at the 5% level) between males and females for several activities: - **Different rates:** email, games, music, photo/video, social media, and text messaging.
- **Similar rates:** browsing the web, downloading apps, and phone calls.

**Takeaway:**
Overall, younger people use their phones more diversely, while older groups rely more on calls. Between genders, certain activities show clear differences in use (albeit marginal), while others (notably calls and web browsing) are essentially equal.