§9.3 Hypothesis testing

Hypothesis testing asks whether a proposed probability model $H_0$ could plausibly have generated the dataset.

❖ The $p$-value is the probability that an outcome as extreme as what we actually saw might have come about by chance, if $H_0$ were true.

❖ A low $p$-value suggests we should reject $H_0$.

❖ “Extreme” is measured by a test statistic $t$, which is up to us to choose.

QUESTION.
Why do you think we define the $p$-value this way, rather than defining it to be “the probability of the $t$ that we actually saw”?
Hypothesis testing is good for questions that we can cast as “Does the evidence suggest rejecting $H_0$?”

- Is my probability model a good enough fit for the dataset?
- Is my new algorithm better than the standard one?
- Does this new UI allow users to do their task faster than before?
- Is this drug effective, compared to placebo?

There’s a common way to set out hypothesis tests for comparing groups (as well as for many similar tasks), called the Neyman-Pearson approach.
Neyman-Pearson hypothesis testing

Let \( x \) be the dataset.

Propose a general parametric model \( H_1 \), and express \( H_0 \) as a restriction on one or more parameters

1. Choose a test statistic based on mle estimates of the parameters of \( H_0 \) and \( H_1 \)

2. Define a random synthetic dataset \( X^* \), what we might see if \( H_0 \) were true.

3. Let \( p \) be the probability (assuming \( H_0 \) to be true) of seeing \( t(X^*) \) as or more extreme than the observed \( t(x) \).

A low \( p \)-value is a sign that \( H_0 \) should be rejected.

\[
\begin{align*}
&\text{General model} \\
&H_1: \quad X_i \sim N(a, \sigma^2) \\
&\quad Y_i \sim N(b, \sigma^2) \\
&\quad Z_i \sim N(c, \sigma^2) \\
&H_0: \quad a = b = c. \quad \text{All samples} \sim N(\mu, \sigma^2) \\
\end{align*}
\]

\[
\hat{\mu} = \text{concat}(x, y, z)
\]

\[
\begin{align*}
\hat{a} &= \bar{x} \\
\hat{b} &= \bar{y} \\
\hat{c} &= \bar{z}
\end{align*}
\]

Can I invent a test statistic using these four parameters, which is liable to be bigger if \( H_0 \) is false?

More generally:

\[
t = \frac{\max_{\text{params of } H_1} \Pr(\text{data} \mid H_1)}{\max_{\text{params of } H_0} \Pr(\text{data} \mid H_0)}
\]
Exercise 9.3.2 (Equality of group means).

We are given three groups of observations from three different systems

\[ x = [7.2, 7.3, 7.8, 8.2, 8.8, 9.5] \]
\[ y = [8.3, 8.5, 9.2] \]
\[ z = [7.4, 8.5, 9.0] \]

Do all three groups have the same mean?

---

```python
# 1. Define test statistic
def t(x, y, z):
    μ = np.mean(np.concatenate([x, y, z]))
    a, b, c = [np.mean(v) for v in [x, y, z]]
    return (a - μ)**2 + (b - μ)**2 + (c - μ)**2

# 2. To generate a synthetic dataset, assuming \( H_0 \) ...
xyz = np.concatenate([x, y, z])
μ̂ = np.mean(xyz)
σ̂ = np.sqrt(np.mean((xyz - μ̂)**2))
def rxyz_star():
    return (np.random.normal(size=len(x), loc=μ̂, scale=σ̂),
            np.random.normal(size=len(y), loc=μ̂, scale=σ̂),
            np.random.normal(size=len(z), loc=μ̂, scale=σ̂))

# 3. Sample the test statistic, find the p-value
_t_ = np.array([t(*rxyz_star()) for _ in range(10000)])
p = np.mean(_t_ >= t(x, y, z))
```
EXERCISE.
Consider the data for IA student marks.

a. What’s a sensible $H_0$ to test?
b. What’s a natural test statistic?
c. How might we generate a random synthetic dataset?

<table>
<thead>
<tr>
<th>gender</th>
<th>mark</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>17</td>
</tr>
<tr>
<td>F</td>
<td>14</td>
</tr>
<tr>
<td>M</td>
<td>18</td>
</tr>
<tr>
<td>O</td>
<td>11</td>
</tr>
<tr>
<td>M</td>
<td>17</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

I think gender affects marks.

I think everyone gets pretty much the same marks, regardless of gender.

(a) Introduce a richer model $H_1: \text{Mark} \sim \hat{\mu}_{\text{gender}} + N(0, \sigma^2)$

Let $H_0$ be: $M_F = M_M = \bar{M}$

(b) $t = (\hat{\mu}_F - \hat{\mu})^2 + (\hat{\mu}_M - \hat{\mu})^2 + (\hat{\mu}_O - \hat{\mu})^2$

where $\hat{\mu}_F, \hat{\mu}_M, \hat{\mu}_O$ are MLE under $H_1$

and $\hat{\mu}$ is MLE under $H_0$

(c) Parametric resampling. Under $H_0$, Marks $\sim \hat{\mu} + N(0, \hat{\sigma}^2)$

Conclusion: for the real marks from last year, $p = 0.71\%$ So we reject $H_0$. 
NON-PARAMETRIC RESAMPLING

(a) $H_0$: marks for all three genders are drawn from the same distribution.

(c) If $H_0$ is true, then the best fit is the empirical distribution of all marks (concatenated together). Let’s simply resample from this.

Conclusion: $p = 0.80\%$

PERMUTATION TESTING

(a) $H_0$: you’d get the same mark regardless of your gender.

(c) Imagine a parallel universe where every student gets assigned a random gender (25 Women, 110 Men, 5 Other). Simulate this parallel universe by randomly permuting the gender column.

Conclusion: $p = 0.82\%$
Why is our mirror image flipped left-right, and not up-down?
IB Data Science syllabus

- Using a probability model to describe data
- Fitting a model’s unknown parameters using MLE
- Reasoning about uncertainty
- Models that depend on linear combinations of features
- Fitting via least squares (when appropriate)
- Parameter interpretation and identifiability
"Induction is the glory of Science and the scandal of Philosophy."

C.D. Broad, 1926

- Maximum likelihood estimation gives us a model that fits the training dataset

But how well will our model work on new data? ("The challenge of induction.")

- Bayesianism and frequentism address this by making careful claims about the Laws of Nature that generated the dataset.

- Alternatively, we could simply say "The performance on in-the-wild data is approximately the performance on holdout data."
Most ML papers don’t state an inductive claim.

Perhaps the authors haven’t thought hard enough to be able to state one?

Perhaps they prefer to leave you, the reader, to make the inference?

“All science is either physics or stamp-collecting.”

Ernest Rutherford (1871–1937)
<table>
<thead>
<tr>
<th>Model Selection</th>
<th>Confidence Intervals for Model Parameters</th>
<th>Confidence Intervals for Predictions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bayesianist</strong></td>
<td>- Given two models, each with a prior weight, use the data to reweight the models</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Empiricist</strong></td>
<td>- Given two models, prefer the one that works better on holdout data</td>
<td>Who cares?</td>
</tr>
<tr>
<td><strong>Frequentist</strong></td>
<td>- Given a model, is it a good enough explanation of the data?</td>
<td>✓</td>
</tr>
</tbody>
</table>

My personal approach:
1. If there’s anything for which I have a justified prior belief, put it into my model as a random variable
2. Choose between competing models empirically
3. Check my final model using frequentist tests
4. Read off confidence intervals, using Bayesianism or frequentism as appropriate.
Challenge: find a 95% confidence interval for the rate of temperate increase in Cambridge from 1985 to the present, in °C/century.
Is CoPilot Bayesianist or frequentist?

CoPilot likes to give me a frequentist confidence interval for the mean of a Gaussian distribution, but I can’t persuade it to give any other answer.

```python
import numpy as np

x = np.array([7.2, 7.3, 7.8, 8.2, 8.8, 9.5])

# Assuming x is sampled from an Exponential distribution with rate lambda,
# find a 95% confidence interval for lambda.

def confint(x):
    n = len(x)
    xbar = np.mean(x)
    s = np.std(x, ddof=1)
    t = 2.776
    return xbar - t*s/np.sqrt(n), xbar + t*s/np.sqrt(n)
```

CoPilot knows a few library calls for hypothesis testing, but it doesn’t know any substance.

ChatGPT4 gives textbook-like dumps with many different choices, but gives spurious answers (including hallucinated library calls) when I ask for specifics.

```python
import numpy as np

x = np.array([7.2, 7.3, 7.8, 8.2, 8.8, 9.5])
y = np.array([3, 2.5, 7.3])

# Test if x and y come from the same distribution

def test_same_distribution(x, y):
    # Test if x and y come from the same distribution
    from scipy.stats import ks_2samp
    ks = ks_2samp(x, y)
    print(ks)
    if ks[1] > 0.05:
        print('Same distribution')
    else:
        print('Different distribution')
```