Inferential Statistics (Lightweight)

What Problem This Solves

Inferential statistics helps you make decisions from incomplete information.

You can't measure everything:

- Can't test every user (sample a subset)
- Can't run A/B test forever (make a decision)
- Can't know if difference is real or luck

Inferential statistics answers: "Is this pattern real or random?"

Intuition & Mental Model

Think: Judging an Ocean by a Cup

You don't need to drain the ocean to know it's salty—one cup tells you enough.

Sample → Infer about Population

```
Population: All users (millions)
Sample: 1000 users you tested
Question: Does sample represent population?
```

Core Concepts

1. Samples vs Populations

Population: Everyone/everything you care about **Sample**: Subset you actually measure

```
// Population (unknown)
const allUsers = /* millions of users */;

// Sample (what you measure)
const sampleUsers = randomSample(allUsers, 1000);

// Goal: Learn about population from sample
```

Key question: Is sample representative?

---

2. Sampling Bias

Bias = sample doesn't represent population

Examples of biased samples:

```
// Bad: Only measure during work hours
const workHourUsers = users.filter(u =>
  u.lastSeen.getHours() >= 9 && u.lastSeen.getHours() <= 17
```

```
);
// Misses night users!

// Bad: Only survey users who respond
const surveyResponders = /* users who filled out form */;
// People who respond are different!

// Good: Random sample
const randomUsers = shuffle(users).slice(0, 1000);
```

Always ask: "Who's missing from this sample?"

---

3. Confidence Intervals

Range where true value likely falls

```
Sample mean: 52ms
95% Confidence Interval: [48ms, 56ms]

Interpretation: "We're 95% confident the true average
                 is between 48ms and 56ms"
```

NOT: "95% chance true value is in this range" **YES**: "95% of intervals calculated this way contain true value"

```
function confidenceInterval(data, confidence = 0.95) {
  const avg = mean(data);
  const std = standardDeviation(data);
  const n = data.length;

  // Standard error
  const se = std / Math.sqrt(n);

  // Z-score for confidence level (simplified)
  const z = confidence === 0.95 ? 1.96 :
            confidence === 0.99 ? 2.58 : 1.96;

  const margin = z * se;

  return {
    lower: avg - margin,
    upper: avg + margin,
    mean: avg
  };
}

const responseTimes = [/* sample of 100 measurements */];
const ci = confidenceInterval(responseTimes, 0.95);
// "95% confident true mean is between ci.lower and ci.upper"
```

Wider interval = less certain Narrower interval = more certain

Larger sample → narrower interval

---

4. Statistical Significance

Is difference real or just random?

```
// Variant A: 5% conversion (1000 users)
// Variant B: 6% conversion (1000 users)

// Question: Is B actually better, or could this
//           happen by chance?
```

P-value: Probability of seeing this difference if there's really no difference

```
p < 0.05 → Statistically significant
           (Less than 5% chance this is random)

p > 0.05 → Not significant
           (Could easily be random)
```

Simplified A/B test:

```
function simpleABTest(controlConversions, controlTotal,
                      variantConversions, variantTotal) {
  const pControl = controlConversions / controlTotal;
  const pVariant = variantConversions / variantTotal;

  // Pooled proportion
  const pPooled = (controlConversions + variantConversions) /
                  (controlTotal + variantTotal);

  // Standard error
  const se = Math.sqrt(
    pPooled * (1 - pPooled) *
    (1/controlTotal + 1/variantTotal)
  );

  // Z-score
  const z = (pVariant - pControl) / se;

  // Approximate p-value (two-tailed)
  const pValue = 2 * (1 - normalCDF(Math.abs(z)));

  return {
    controlRate: pControl,
    variantRate: pVariant,
    difference: pVariant - pControl,
    zScore: z,
    pValue: pValue,
    significant: pValue < 0.05
  };
```

```
}

// Variant B has 6% vs Control A has 5%
// With 1000 users each
// Is this significant?
```

Important: Statistical significance ≠ practical significance

```
// Statistically significant but meaningless
{
  controlRate: 0.05000,  // 5.00%
  variantRate: 0.05001,  // 5.01%
  pValue: 0.04,          // Significant!
  // But 0.01% improvement is not worth deploying
}
```

---

5. Correlation vs Causation

Correlation: Two things move together **Causation**: One thing causes the other

```
Ice cream sales ↔ Drownings
Correlated, but ice cream doesn't cause drownings
Both caused by summer weather (confounding variable)
```

In software:

```
// Observation: Users with dark mode enabled
//              have higher engagement

// Correlation: Yes
// Causation: Maybe not!
//   - Power users more likely to enable dark mode
//   - Power users already more engaged
//   - Dark mode didn't cause engagement
```

Test causation: Randomized experiments (A/B tests)

```
// Good: Random assignment
users.forEach(user => {
  user.variant = Math.random() < 0.5 ? 'A' : 'B';
});

// Now can claim causation if B performs better
```

---

6. Sample Size Matters

Larger sample = more confidence

```
// Small sample (n=10)
const small = [52, 48, 55, 50, 53, 49, 51, 54, 50, 52];
confidenceInterval(small);
// Wide interval: [48.5, 54.3]

// Large sample (n=1000)
const large = /* 1000 measurements with similar variance */;
confidenceInterval(large);
// Narrow interval: [51.2, 52.8]
```

Rule of thumb for A/B tests:

```
function minimumSampleSize(baselineRate, minDetectableDiff,
                           alpha = 0.05, power = 0.8) {
  // Simplified formula
  const p1 = baselineRate;
  const p2 = baselineRate + minDetectableDiff;
  const pAvg = (p1 + p2) / 2;

  // Z-scores for alpha and power
  const zAlpha = 1.96;  // 95% confidence
  const zBeta = 0.84;   // 80% power

  const n = (zAlpha + zBeta) ** 2 * pAvg * (1 - pAvg) /
            (p1 - p2) ** 2;

  return Math.ceil(n);
}

// Baseline: 5%, want to detect 1% improvement
minimumSampleSize(0.05, 0.01);
// Need ~7800 users per variant

// Want to detect 0.1% improvement
minimumSampleSize(0.05, 0.001);
// Need ~780,000 users per variant!
```

Can't detect tiny differences without huge samples.

Software Engineering Connections

1. A/B Testing

```
class ABTest {
  constructor(name) {
    this.name = name;
    this.variants = {
      A: { shown: 0, converted: 0 },
      B: { shown: 0, converted: 0 }
```

```
    };
  }

  assignVariant(userId) {
    // Consistent hash for same user
    return simpleHash(userId) % 2 === 0 ? 'A' : 'B';
  }

  recordImpression(userId) {
    const variant = this.assignVariant(userId);
    this.variants[variant].shown++;
  }

  recordConversion(userId) {
    const variant = this.assignVariant(userId);
    this.variants[variant].converted++;
  }

  getResults() {
    const { A, B } = this.variants;

    if (A.shown < 100 || B.shown < 100) {
      return { status: 'insufficient_data' };
    }

    const test = simpleABTest(
      A.converted, A.shown,
      B.converted, B.shown
    );

    return {
      status: 'complete',
      winner: test.significant && test.variantRate > test.controlRate ? 'B' : 'A',
      ...test
    };
  }
}
```

---

2. Performance Monitoring

```
class PerformanceMonitor {
  async detectRegression(currentMetrics, historicalMetrics) {
    // Compare current performance to historical baseline
    const currentMean = mean(currentMetrics);
    const historicalCI = confidenceInterval(historicalMetrics, 0.95);

    if (currentMean > historicalCI.upper) {
      return {
        regression: true,
        message: `Performance degraded: ${currentMean}ms vs baseline
```

```
${historicalCI.mean}ms`,
        confidence: 0.95
      };
    }

    return { regression: false };
  }
}
```

---

3. Feature Rollout

```
// Gradual rollout with monitoring
async function rolloutFeature(feature, targetPercent = 100) {
  const rolloutSteps = [1, 5, 10, 25, 50, 100];

  for (const percent of rolloutSteps) {
    if (percent > targetPercent) break;

    // Enable for percent of users
    await enableForPercent(feature, percent);

    // Wait and monitor
    await sleep(3600000); // 1 hour

    // Check metrics
    const metrics = await getMetrics(feature);
    const baseline = await getBaselineMetrics();

    // Statistical test: is error rate higher?
    if (metrics.errorRate > baseline.errorRate) {
      const test = testSignificance(metrics.errors, baseline.errors);

      if (test.significant) {
        await rollback(feature);
        throw new Error('Feature caused significant regression');
      }
    }
  }
}
```

---

4. Alerting with Confidence

```
class AlertingSystem {
  constructor(metric, threshold) {
    this.history = [];
    this.threshold = threshold;
  }

  record(value) {
```

```
    this.history.push(value);

    // Keep last 24 hours
    if (this.history.length > 1440) {
      this.history.shift();
    }
  }

  shouldAlert(currentValue) {
    if (this.history.length < 100) {
      // Not enough data
      return false;
    }

    // Calculate baseline
    const ci = confidenceInterval(this.history, 0.99);

    // Alert if current value outside 99% CI
    return currentValue > ci.upper || currentValue < ci.lower;
  }
}

// Usage
const latencyMonitor = new AlertingSystem('latency', 100);

setInterval(() => {
  const current = getCurrentLatency();
  latencyMonitor.record(current);

  if (latencyMonitor.shouldAlert(current)) {
    alert(`Latency anomaly: ${current}ms`);
  }
}, 60000);
```

Common Misconceptions

❌ "P < 0.05 means 95% chance hypothesis is true"

No! P-value is probability of seeing data IF null hypothesis is true.

Not the same as probability hypothesis is true.

❌ "Statistically significant = important"

Significance = unlikely to be chance **Importance** = meaningful in practice

```
// Significant but unimportant
{
  improvement: 0.001%,  // Tiny
  pValue: 0.001,        // Very significant
```

```
  sampleSize: 10000000  // Huge sample detects tiny effects
}
```

❌ "Need equal sample sizes"

Helpful but not required. Statistical tests work with unequal sizes.

```
// Valid test
simpleABTest(50, 1000, 150, 3000); // Different sizes OK
```

❌ "Correlation implies causation"

Never! Need randomized experiment to claim causation.

❌ "Can keep testing until significant"

No! This is "p-hacking" and invalidates results.

```
// Bad: Keep checking until p < 0.05
while (test.pValue > 0.05) {
  collectMoreData();
  test = runTest();
}

// Good: Decide sample size upfront, test once
```

Practical Mini-Exercises

Exercise 1: Evaluate A/B Test

```
const results = {
  A: { users: 5000, conversions: 250 },  // 5%
  B: { users: 5000, conversions: 275 }   // 5.5%
};
```

Is B significantly better?

▶ Solution

Exercise 2: Sample Size Planning

You want to improve conversion from 10% to 11% (1% absolute, 10% relative).

How many users needed per variant (95% confidence, 80% power)?

▶ Solution

Exercise 3: Identify Sampling Bias

Which samples are biased?

```
// A) Survey: Email all users, analyze responders
// B) Performance: Measure during peak traffic hours
// C) Survey: Randomly select 1000 users, call until you reach them
```

▶ Solution

Summary Cheat Sheet

Key Concepts

```
// Confidence interval
const ci = confidenceInterval(sample, 0.95);
// "95% confident true value is between ci.lower and ci.upper"

// Statistical significance
if (pValue < 0.05) {
  // Result is statistically significant
  // Less than 5% chance this is random
}

// Sample size
// Larger sample → narrower confidence intervals
// Larger sample → can detect smaller effects
```

A/B Testing Checklist

1. **Plan sample size** before starting
2. **Randomize assignment** (no selection bias)
3. **Define success metric** upfront
4. **Run until planned size** reached
5. **Check for significance** ($p < 0.05$)
6. **Check practical significance** (is lift meaningful?)
7. **Verify across segments** (does it work for everyone?)

Red Flags

- ✗ Stopping test early because "it's significant"
- ✗ Testing multiple metrics, reporting only significant ones
- ✗ Claiming causation from observational data
- ✗ Ignoring practical significance
- ✗ Not accounting for sample bias

Next Steps

Inferential statistics helps you make confident decisions from samples. You now understand how to test hypotheses, run A/B tests properly, and avoid common statistical pitfalls.

Next, we'll explore **data distributions**—understanding the shapes and patterns that appear in real-world data.

Continue to: 09-data-distributions.md