

# Evaluation

CS 197 | Stanford University | Michael Bernstein

# Administrivia

Evaluation plan assignment going live today, due in Week 8

(Details on the assignment page.)

Reminder: project reports through week 8, evaluation plan due week 8, draft paper due in Week 9

**“But how would we  
even evaluate that?”**

People often rush to this question early on in ideation.

Today's goal is to provide scaffolding for how to answer it.

# Today's big idea: evaluation

How do we get precise about what we need to evaluate for our project?

How do we design an appropriate evaluation?

How do we analyze our evaluation results?

**Why perform  
evaluation in research?**

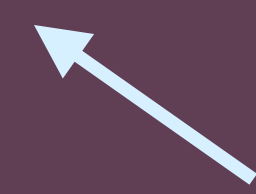
# Idea Shark Tank

Recall from Week 1 that research introduces a new idea into the world.

So...how do we know if that idea is worth adopting or paying attention to?

Option 1 (“The Simon Cowell Solution”): Academia’s Got Talent, Shark Tank,

Option 2: Construct an evaluation to test the idea fairly

 **Let’s do this one: the goal isn’t advocacy; it’s an understanding of the idea’s strengths and limits**

# Standards of evidence

Every field has an accepted standard of evidence — a set of methods that are agreed upon for proving a point

Medicine: Double-blind randomized controlled trial

Philosophy: Rhetoric

Math: Formal proof

Applied Physics: Measurement

# Standards of evidence

In computing, because areas use different methods, the standard of evidence differs based on the area.

Your goal: convince an expert in your area.

So, use the methods appropriate to your area.



# Designing an evaluation

# Problematic point of view

“But how would we evaluate this?”

Why is this point of view problematic?

Implication: “I believe the idea is right, but I don’t believe that we can prove it.”

Implication: “The thread of designing the evaluation is different than the process Evaluation is distinct from the validity of the idea.”

Neither implication is correct. **If you can precisely articulate your idea and your bit flip, then you can design an appropriate evaluation. If you can’t precisely articulate your idea and your bit flip, then you can’t design an appropriate evaluation.**

# Step 1: articulate your thesis

A much more productive approach is to derive an evaluation design directly from your idea.

What is the main thesis of your work?

(Lucky for you, you came up with this when writing the Introduction of your paper. It's the topic sentence of your bit flip paragraph.)

## Recall:

### Bit

Network behaviors are defined in hardware, statically.

Code compilers should utilize smart algorithms to optimize into machine code.

A minimum graph cut algorithms should always return correct answers.

### Flip

If we define the behaviors in software, networks can become dynamic and more easily debuggable.

Code compilers will find more efficient outcomes if they just do monte carlo (random!) explorations of optimizations.

A randomized, probabilistic algorithm will be much faster, and we can still prove a limited probability of an error.

Discuss your thesis with your team [4min]

# Step 2: map your thesis onto a claim

There are only a small number of claim structures implicit in most theses:

**$x > y$** : approach  $x$  is better than approach  $y$  at solving the problem

**$\exists x$** : it is possible to construct an  $x$  that satisfies some criteria, whereas it was not possible before

**bounding  $x$** : approach  $x$  only works given certain assumptions

## Bit

Network behaviors are defined in hardware, statically.

Code compilers should utilize smart algorithms to optimize into machine code.

A minimum graph cut algorithms should always return correct answers.

## Flip

If we define the behaviors in software, networks can become dynamic.

Code compilers will find more efficient outcomes if they just do monte carlo (random!) explorations of optimizations.

A randomized, probabilistic algorithm will be much faster, and we can still prove a limited probability of an error.

## Claim

$\exists x$ : software-defined behaviors can be changed on the fly, whereas hardware cannot

$x > y$ : monte carlo exploration will produce more optimized code than hand-tuned compilers

$x > y$ : a randomized graph cut algorithm is faster and has bounded error

Discuss your claim with your team [4min]



# Step 3: claims imply an evaluation design

Each claim structure implies an evaluation design

**$x > y$** : given a representative task or set of tasks, test whether  $x$  in fact outperforms  $y$  at the problem

**$\exists x$** : demonstrate that your approach achieves  $x$

**bounding  $x$** : demonstrate bounds inside or outside of which approach  $x$  fails

## Flip

If we define the behaviors in software, networks can become dynamic.

Code compilers will find more efficient outcomes if they just do monte carlo (random!) explorations of optimizations.

A randomized, probabilistic algorithm will be much faster, and we can still prove a limited probability of an error.

## Claim

$\exists x$ : software-defined behaviors can be changed on the fly, whereas hardware cannot

$x > y$ : monte carlo exploration will produce more optimized code than hand-tuned compilers

$x > y$ : a randomized graph cut algorithm is faster and has bounded error

## Implied evaluation

Demonstrate that behaviors propagate, and which kind of behaviors can be authored

Compare runtime of generated machine code against known best approaches

Prove runtime for randomized algorithm (vs. prior algorithm) and probability of error

Discuss the high-level design with your team [4min]

# Architecture of an Evaluation

# Four constructs that matter

To develop your evaluation plan, you need to get precise about four components of your evaluation:

Dependent variable

Independent variable

Task

Threats

# DV: dependent variable

In other words, what's the outcome you're measuring?

Efficiency? Accuracy? Performance? Satisfaction? Trust? Psychological safety? Learning transfer? Adherence to behavior change?

The choice of this quantity should be clearly implied by your thesis.

It's often tempting to measure many DVs, and I'm not against doing so. However, one should be your central outcome, and the others auxiliary.

Discuss with your team [2min]

# IV: independent variable

In other words, what determines what  $x$  and  $y$  are? What are you manipulating in order to cause the change in the dependent variable?

The IV is the construct that leads to conditions in your evaluation. Examples might include:

- Algorithm

- Dataset size or quality

- Interface

Discuss with your team [2min]

# Task

What, specifically, is the routine being followed in order to manipulate the independent variable and measure the dependent variable?

We will perform 1-shot prediction of classes at the 25th percentile of popularity in ImageNet according to Google search volume.

Participants will have thirty seconds to identify each article as disinformation or not, within-subjects, randomizing across interfaces

We will run a performance benchmark drawn from Author et al. against each system

Discuss with your team [2min]



# Threats

What are your threats to validity? In other words, what might bias your results or mean that you're telling an incomplete story?

Might your selection of which classes to predict influence the outcome?

Are you running on particular cloud architectures that are amenable to, or not amenable to, your task?

Are your participants biased toward healthy young technophiles?

Do your participants always see the best interface first?

# Threats

There are typically three ways to handle these kinds of issues:

- 1) Argue as irrelevant: yes, that bias might exist, but it's not conceptually important to the phenomenon you're studying and is unlikely to strongly effect the outcome or make the results less generalizable
- 2) Stratify: re-run your evaluation in each setting to see whether the outcomes change
- 3) Randomize: explicitly randomize (e.g., people) across values of the control variable. For example, randomize the order in which people see the interface.

Discuss with your team [2min]

# Find your Patronus

There's no need to start from scratch on this.

Your nearest neighbor paper, and the rest of your literature search, has likely already introduced evaluation methods into this literature that can be adapted to your purpose.

Start here: figure out what the norms are, and tweak them. Talk to your TA if helpful.

# Statistical Hypothesis Testing

a dramatically incomplete primer

# Are you just lucky?

So your idea came out ahead. Great!

...but is that really true in general? Or did you just get lucky in the people you sampled, or in the inputs you sampled, and it could have easily come out a wash?

You live in one world in which the results came out the way they did. If we tried it in one hundred parallel worlds, in how many would it have come out the same way?

1? 80? 100?

# Enter statistics

Statistical hypothesis testing is a way of formalizing our intuition on this question. It quantifies: in what % of parallel worlds would the results have come out this way?

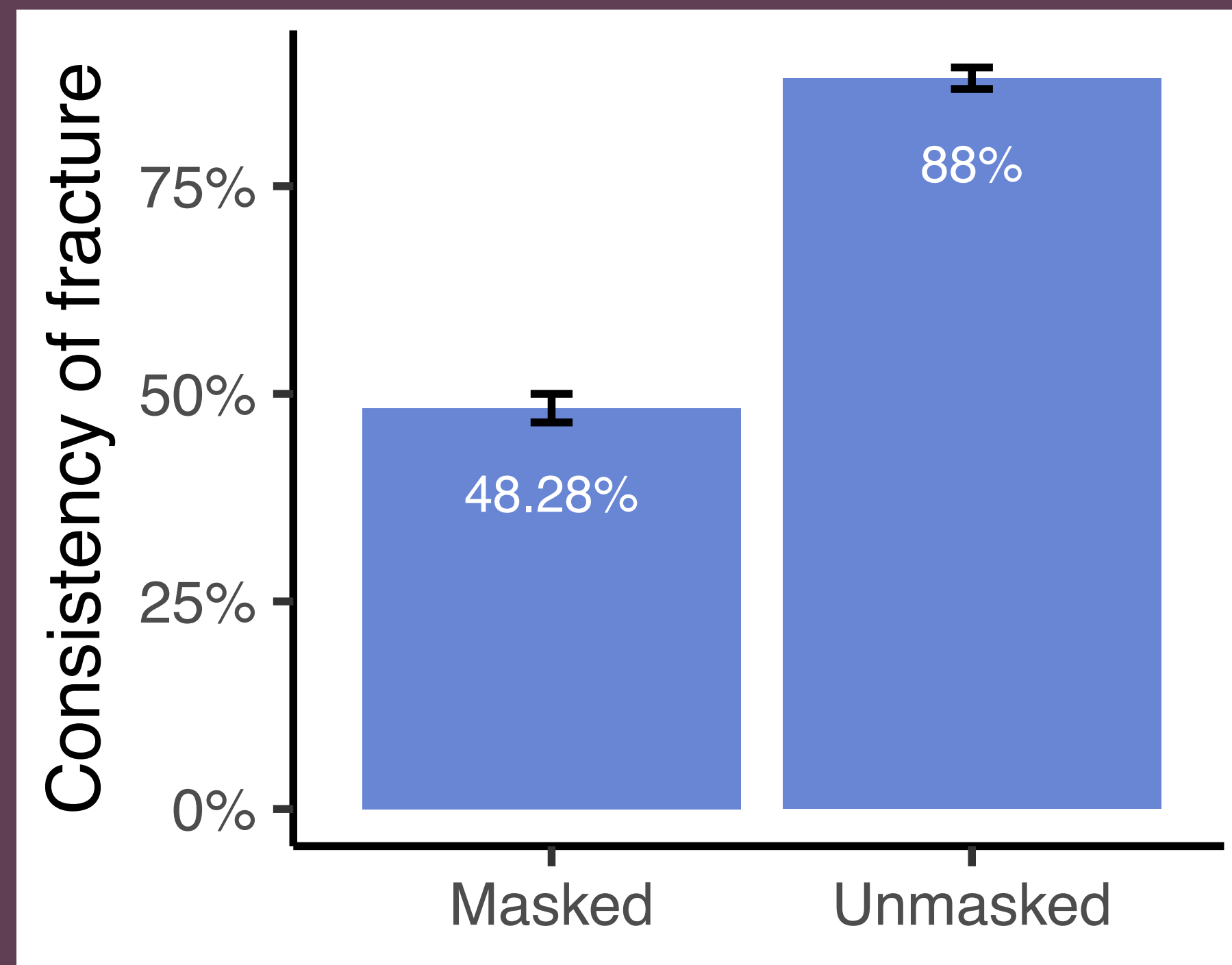
This is what we call a **p-value**.

$p < .05$  intuitively means “a result like this is likely to have come up in at least 95% of parallel worlds”

Scientific communities have different standards for what level of  $p$  to use for statistical significance, especially in an era of big data. Many still use .05. It's a topic for another class.

# Step 1: don't run the stats

Instead, visualize your results. Create graphs, report descriptive statistics



Make sure to include error bars: they give you an intuitive sense of how much variation there is around that mean, which can hint you to outliers

Rushing first to statistics often fails to identify outliers and other weird artifacts that can mess with your stats

# Step 2: learn the stats

Know what you are testing and the assumptions that your test makes. This is outside the scope of CS 197, so I recommend working with your TA. For example, you might consider:

Categorical data? Chi-square

Continuous data with two conditions? t-test

Continuous data with  $>$  two conditions? ANOVA with posthoc tests



# Mid-quarter feedback

[hci.st/197feedback](https://hci.st/197feedback)

# Assignment 7 (what!?)

Assignment 7 is your evaluation plan.

Thesis, Claim, Evaluation Design, and Writeup

We are launching Assignment 7 early! It's not formally due until Week 8.

But, some projects, which are more study- or measurement-oriented, need more lead time to complete their evaluation. If you are in this set, turn this assignment in early so that you can proceed with data collection.

# Evaluation

Slide content shareable under a Creative Commons Attribution-NonCommercial 4.0 International License.