Will AI Replace Software Engineers?

Our 100k-Engineer Study Says… Not So Fast

Yegor Denisov-Blanch
Software Engineering Productivity Research Group
Stanford University

Tech CEOs made bold predictions…

Jan 2025

Meta Engineering headcount increased 11% year-over-year[1]

Mar 2025

How close are we to replacing software engineers with AI?

1) Source: LinkedIn

Time-series cross-sectional study on software engineering productivity

Time-Series

Git History

… | 2023 | 2024 | 2025

Historical data

Cross-Sectional

600+ Companies
100k+ Software Engineers

Enterprise | Mid-Size | Startup

What we'll cover

1. **Limitations of Existing Productivity Metrics**
2. Our Methodology: Panel of Experts
3. Case Study: Higher PR Counts, Lower Quality
4. Results: Impact of AI on Developer Productivity

Existing metrics used to measure the impact of AI on engineering productivity have limitations

1 Commits / PRs

- Task size varies
- More tasks != more productivity
- Tasks to fix bugs introduced by AI

2 DORA Metrics

- Measure of CI/CD adoption or DevOps performance
- Not a measure of productivity

3 Greenfield Tasks

- AI is great at greenfield, boilerplate code
- Most of software engineering isn't greenfield

4 Surveys

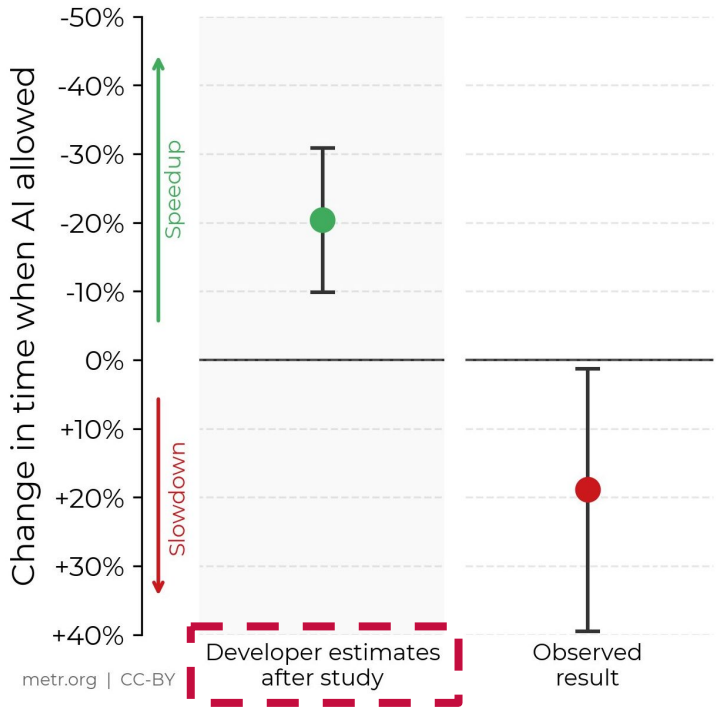
- Ineffective measure of developer productivity
- Perception != reality

Developers thought AI was speeding them up by 20%... but it actually slowed them down by 20%

Potential reasons

Participants…

- **Were deeply familiar with repos** ▶ AI helps less in familiar repos
- **Had little experience with AI coding assistants** ▶ More AI experience = better results



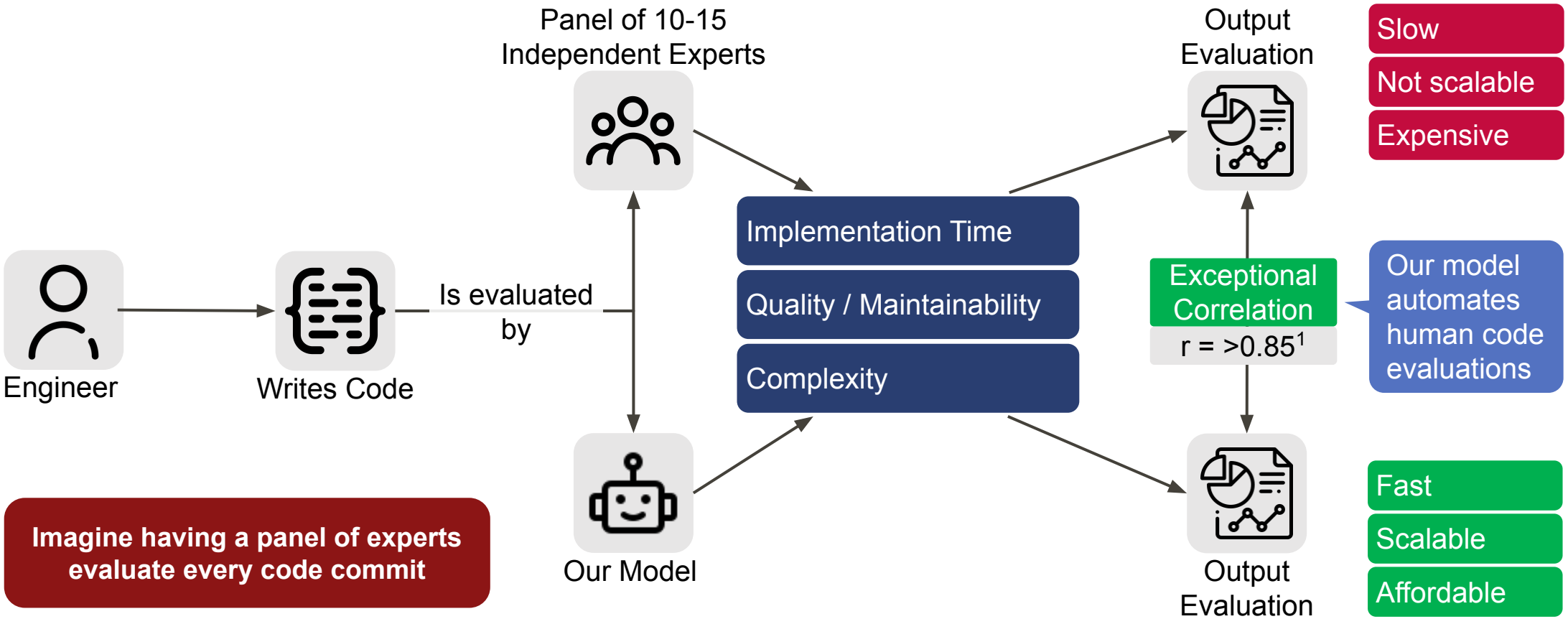
Surveys can't accurately estimate AI productivity impact

Source: Becker, J., Rush, N., Barnes, E., Rein, D. (2025). Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity. arXiv preprint arXiv:2507.09089.

What we'll cover

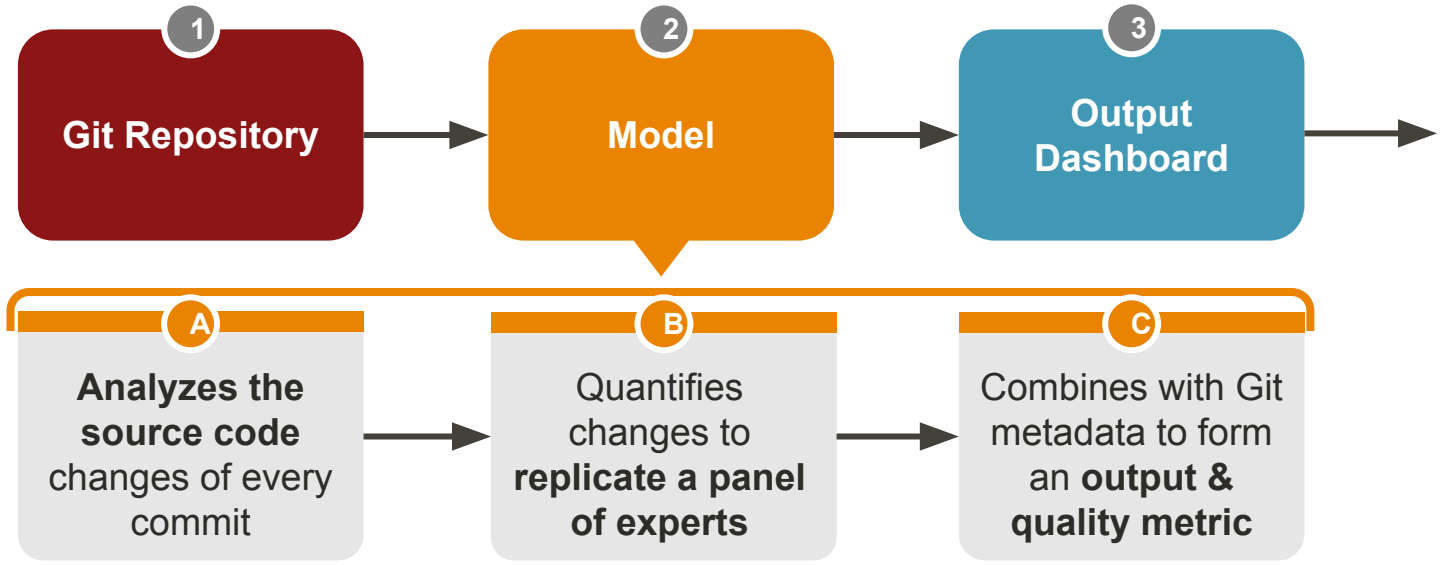
1. Limitations of Existing Productivity Metrics
2. **Our Methodology: Panel of Experts**
3. Case Study: Higher PR Counts, Lower Quality
4. Results: Impact of AI on Developer Productivity

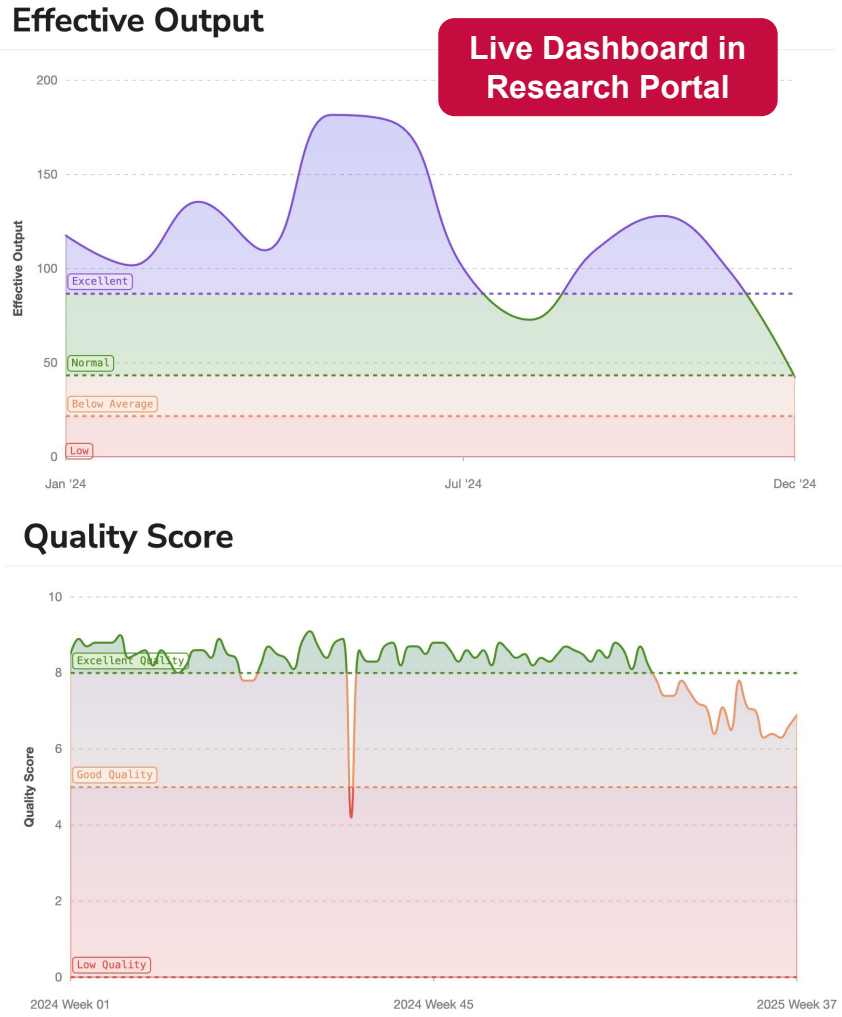
Our model automates human code evaluations at scale



1) Significant improvements in correlation & ICC2K achieved since publishing "Predicting Expert Evaluations in Software Code Reviews"

Our model quantitatively evaluates software engineering output by analyzing the source code changes of every commit

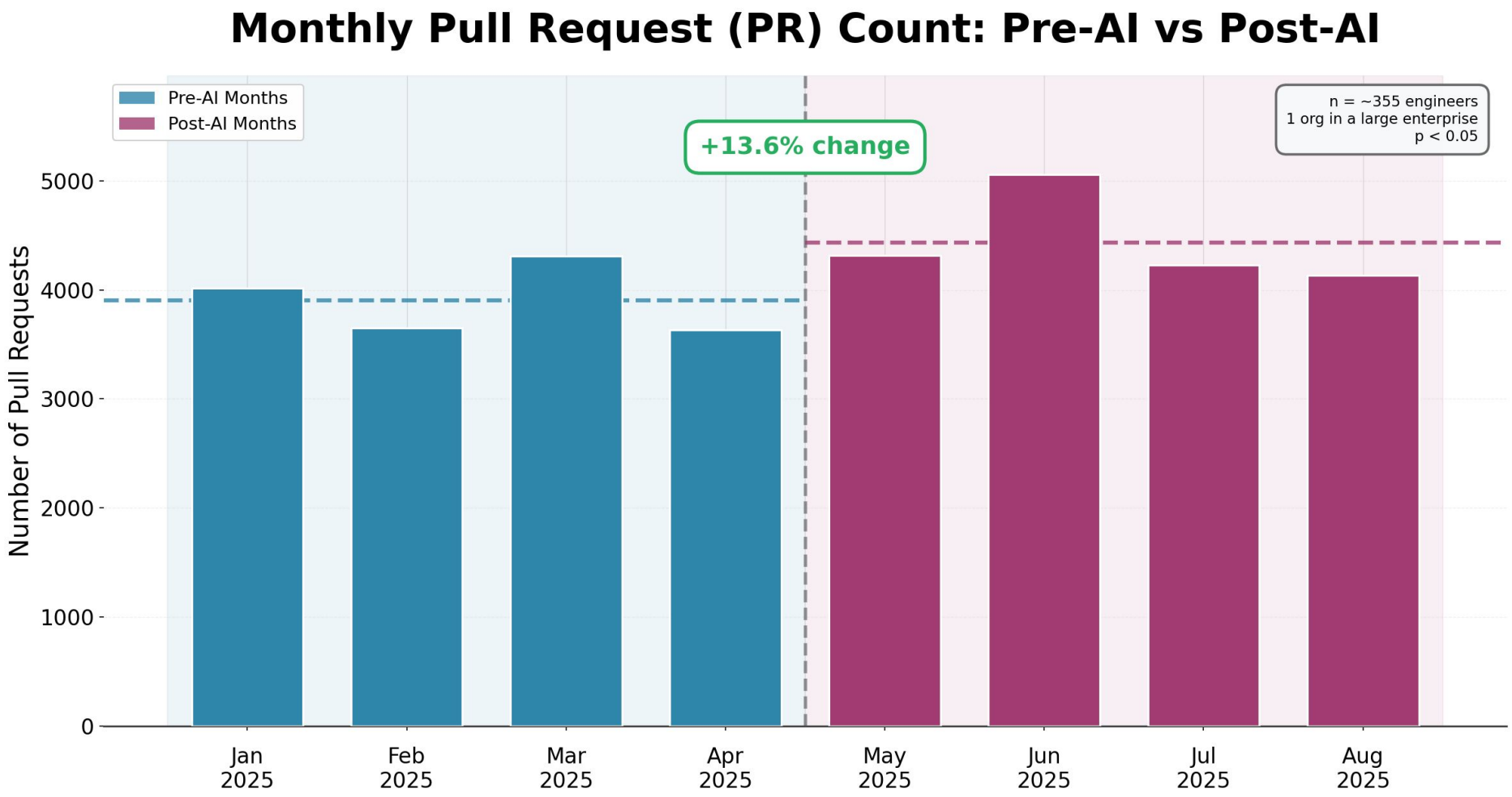




What we'll cover

1. Limitations of Existing Productivity Metrics
2. Our Methodology: Panel of Experts
3. **Case Study: Higher PR Counts, Lower Quality**
4. Results: Impact of AI on Developer Productivity

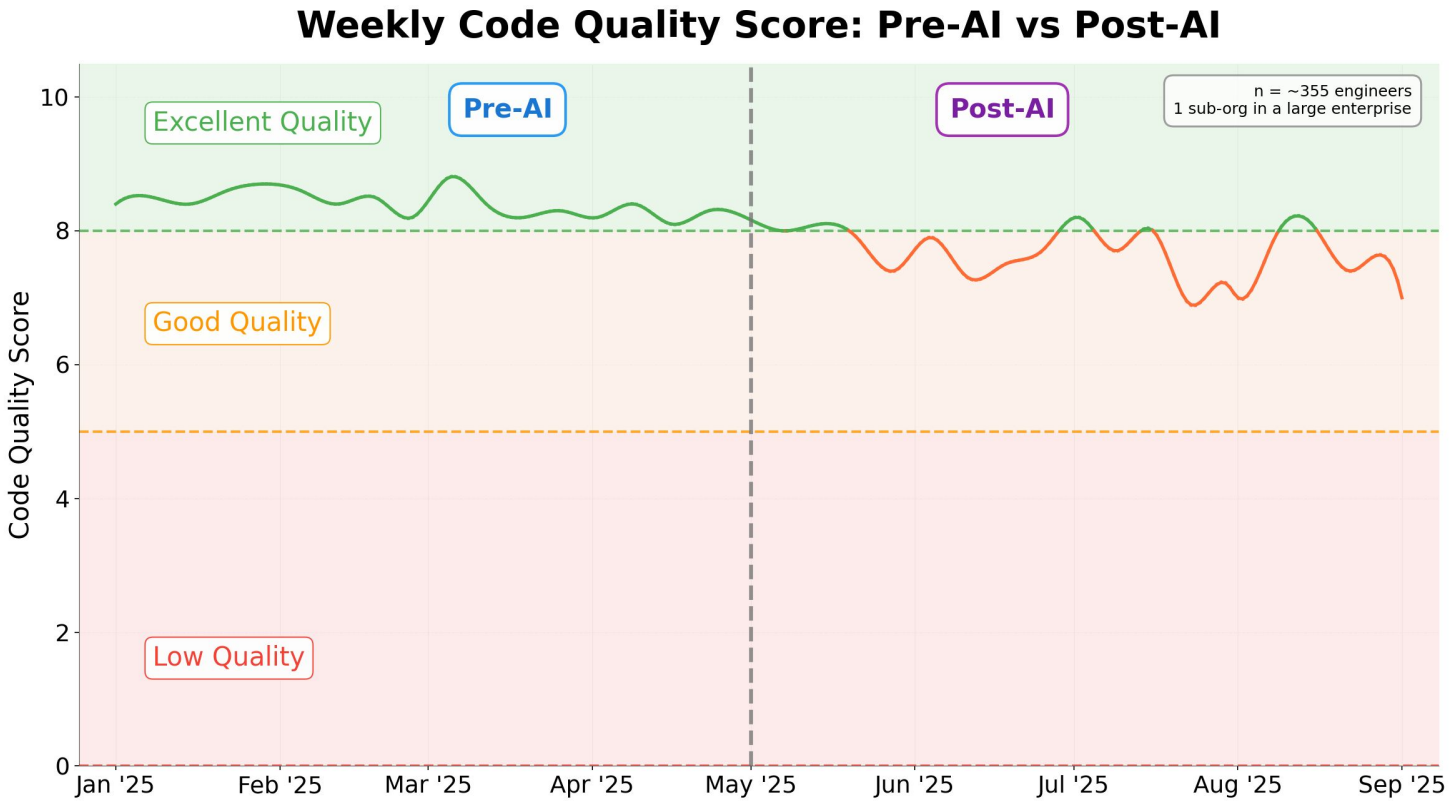
Adopting AI increased PRs by 14%... but more PRs doesn't mean better

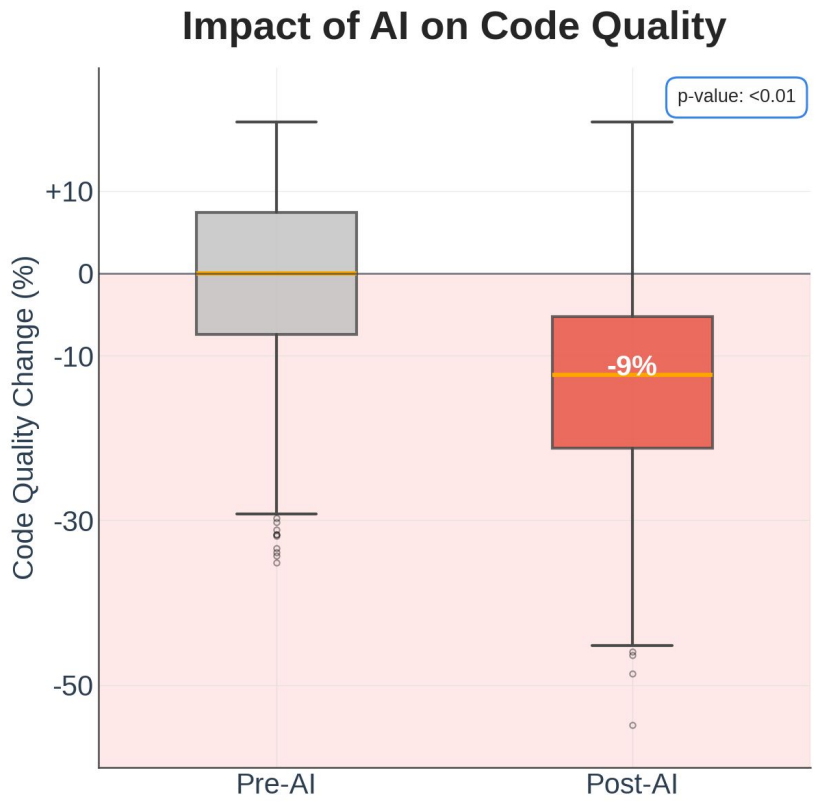


PR Count Limitations

- **Reviewer Burden**
- **Smaller / Simpler PRs**
- **Ignores Quality**

Adopting AI decreased Code Quality by 9% and increased variance by 3.6x





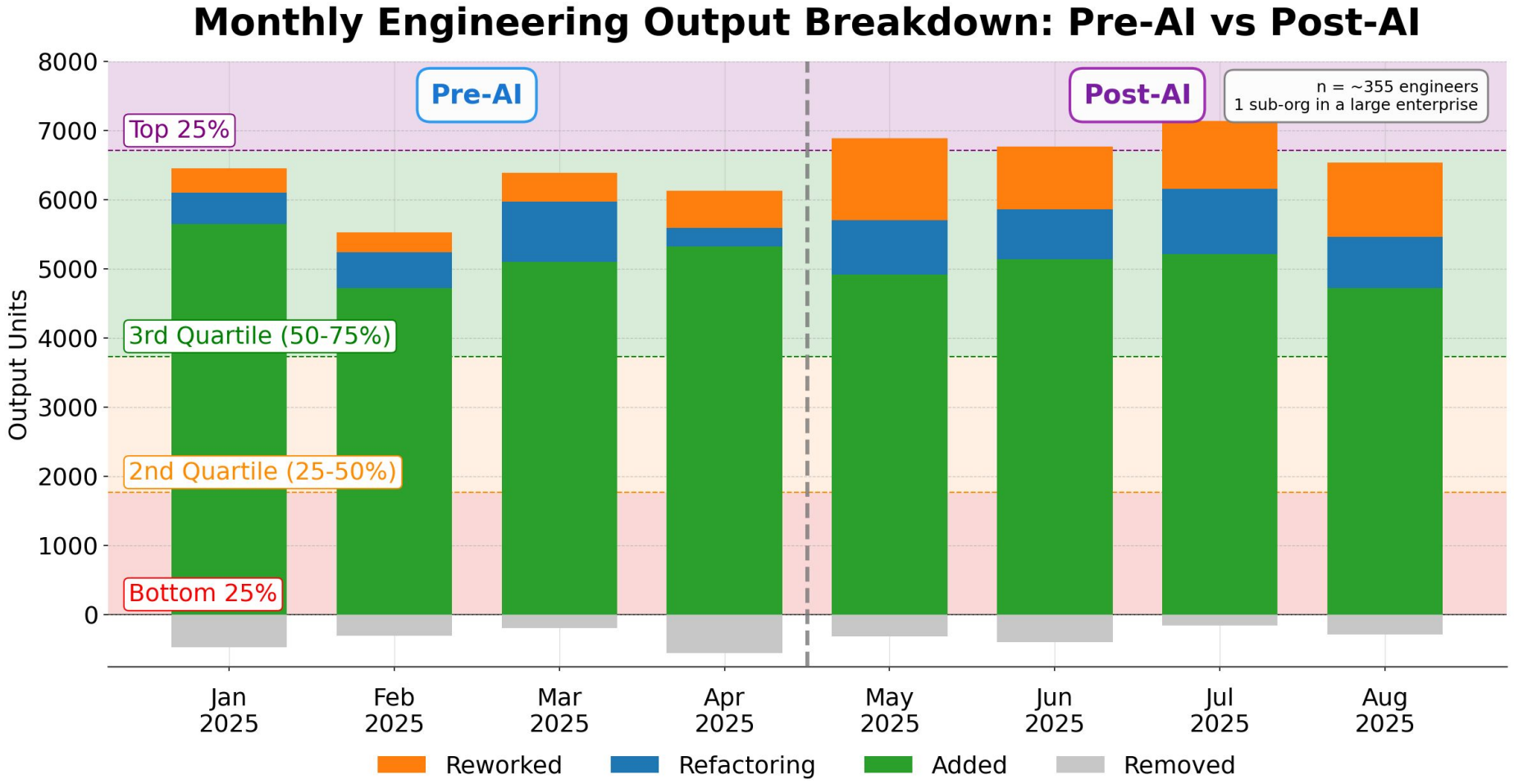
Worse Quality

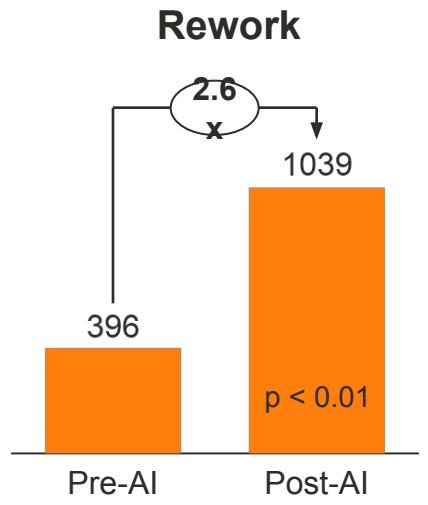
-9% Decrease in Code Quality

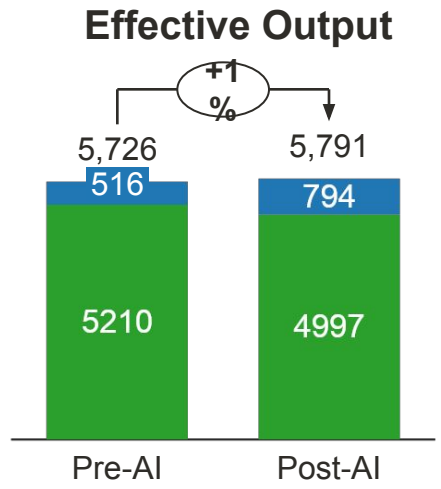
More Erratic Quality

3.6x Increase in Variance

Adopting AI didn't increase "effective output" and increased rework by 2.6x







Not lines of code, but our metric (replicates a panel of human experts)

Adopting AI didn't yield positive results for this company…

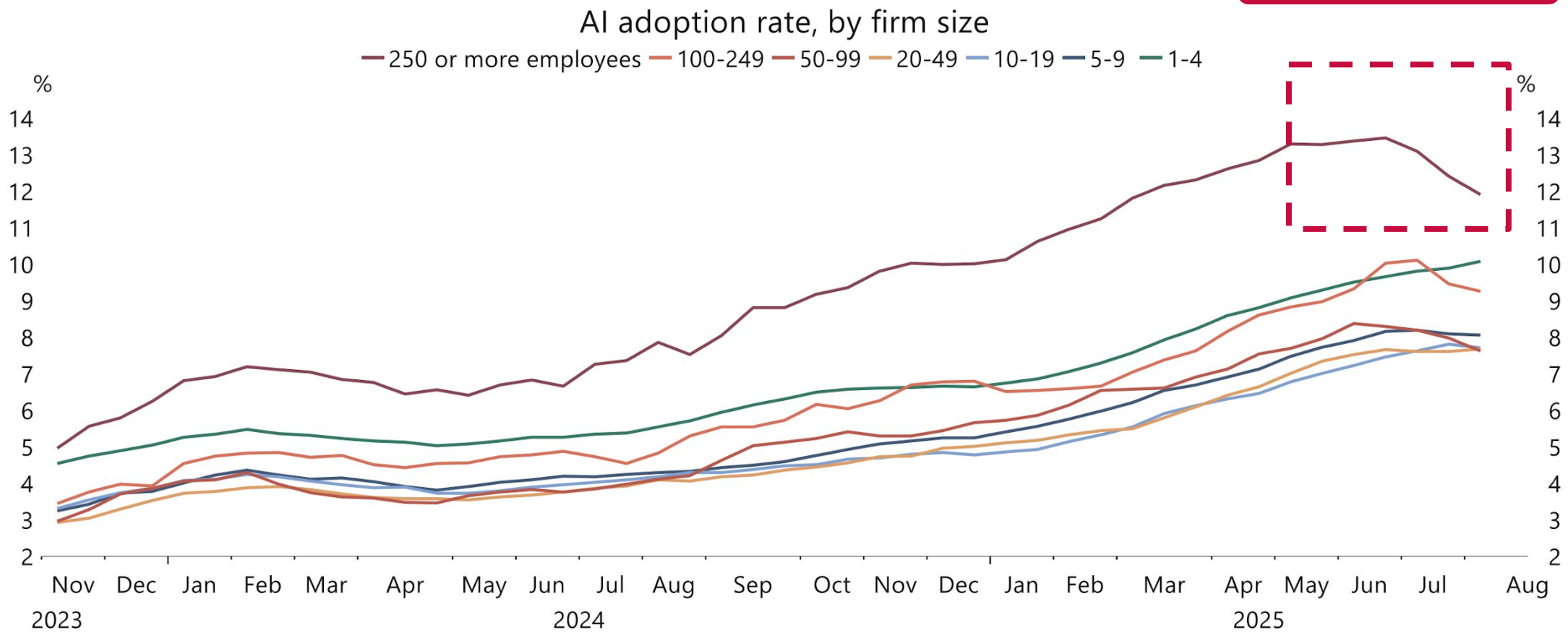
AI Impact on Productivity - Metrics Summary

Pull Requests (PRs)	Code Quality	Effective Output (our metric)	Rework
▶ +13.6% Increase	▼ -9% Decrease	▶ +1% Increase	▼ 2.6x Increase
Inconclusive More PRs =/= better	**Problematic**	No meaningful change	**Problematic**

Is this a failed AI coding assistant pilot?

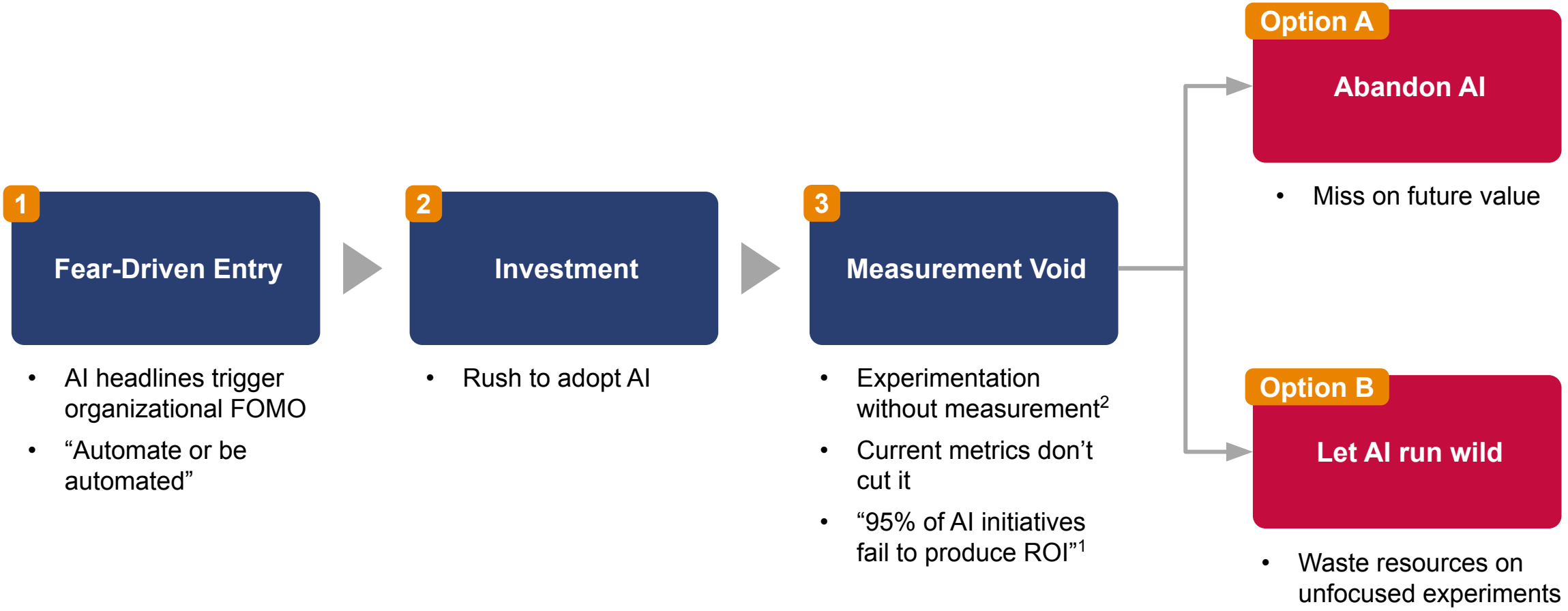
Enterprise AI adoption rates are declining

Likely understated adoption, but focus on downward trend



Note: Data is six-survey moving average. The survey is conducted biweekly. Sources: US Census Bureau, Macrobond, Apollo Global Management Chief Economist

Fear-driven AI investment without the right metrics leads to wasteful experimentation and failure

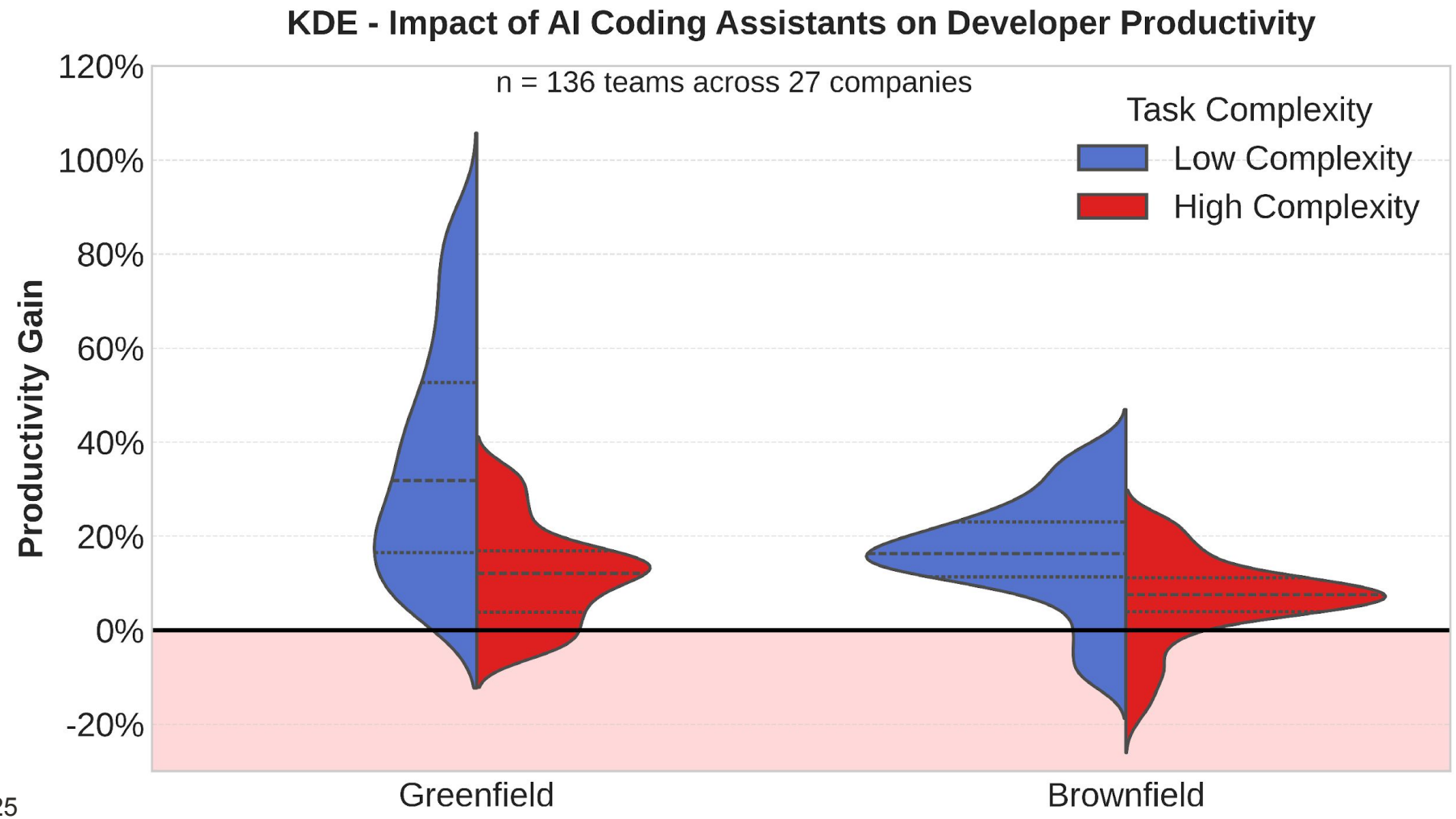


1) MIT NANDA, The GenAI Divide, State of AI in Business (2025); 2) Thinking influenced by "Beware the AI Experimentation Trap" (2025, Nathan Furr, Andrew Shipilov)

What we'll cover

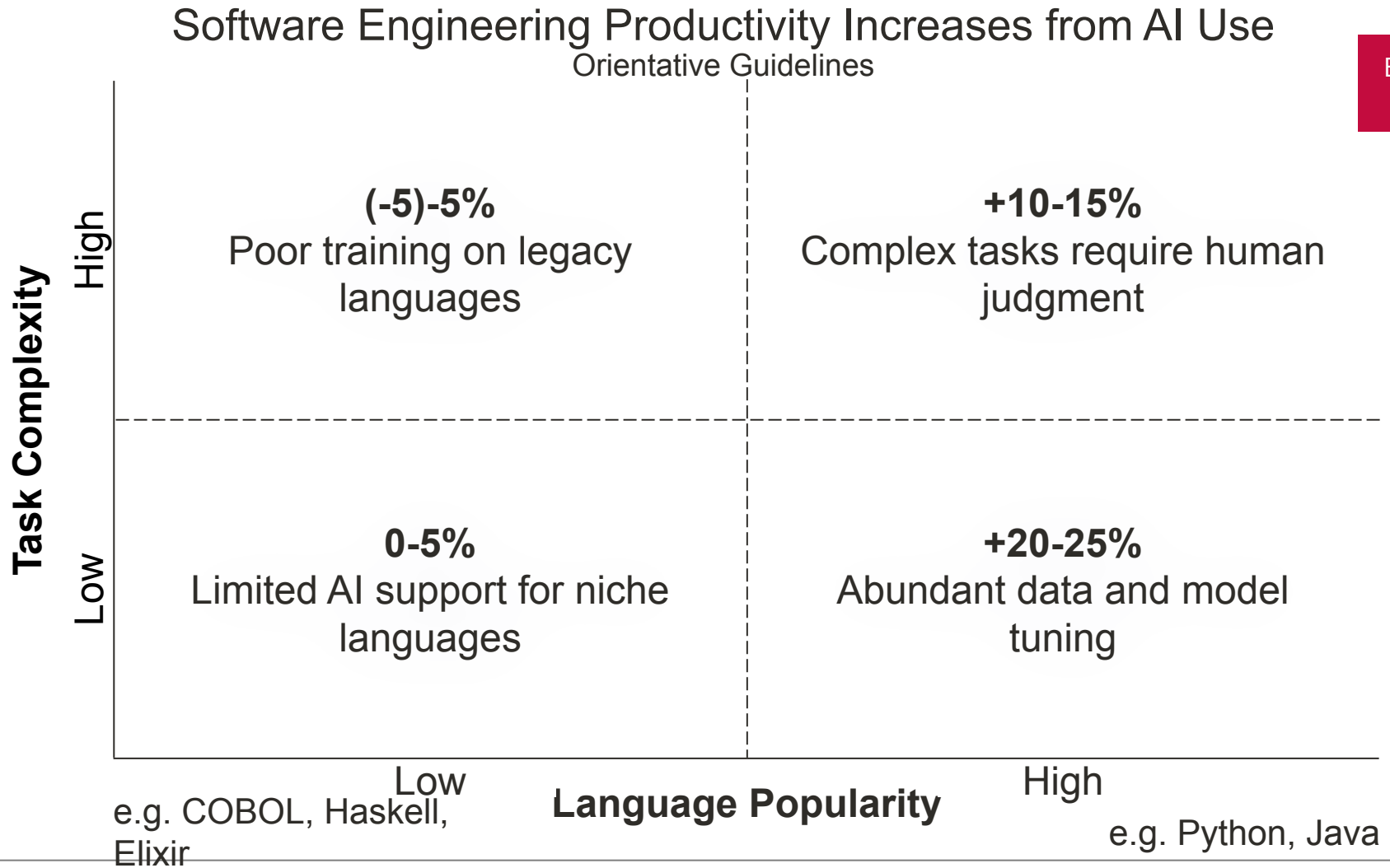
1. Limitations of Existing Productivity Metrics
2. Our Methodology: Panel of Experts
3. Case Study: Higher PR Counts, Lower Quality
4. **Results: Impact of AI on Developer Productivity**

AI delivers larger engineering productivity gains on easy tasks and greenfield projects

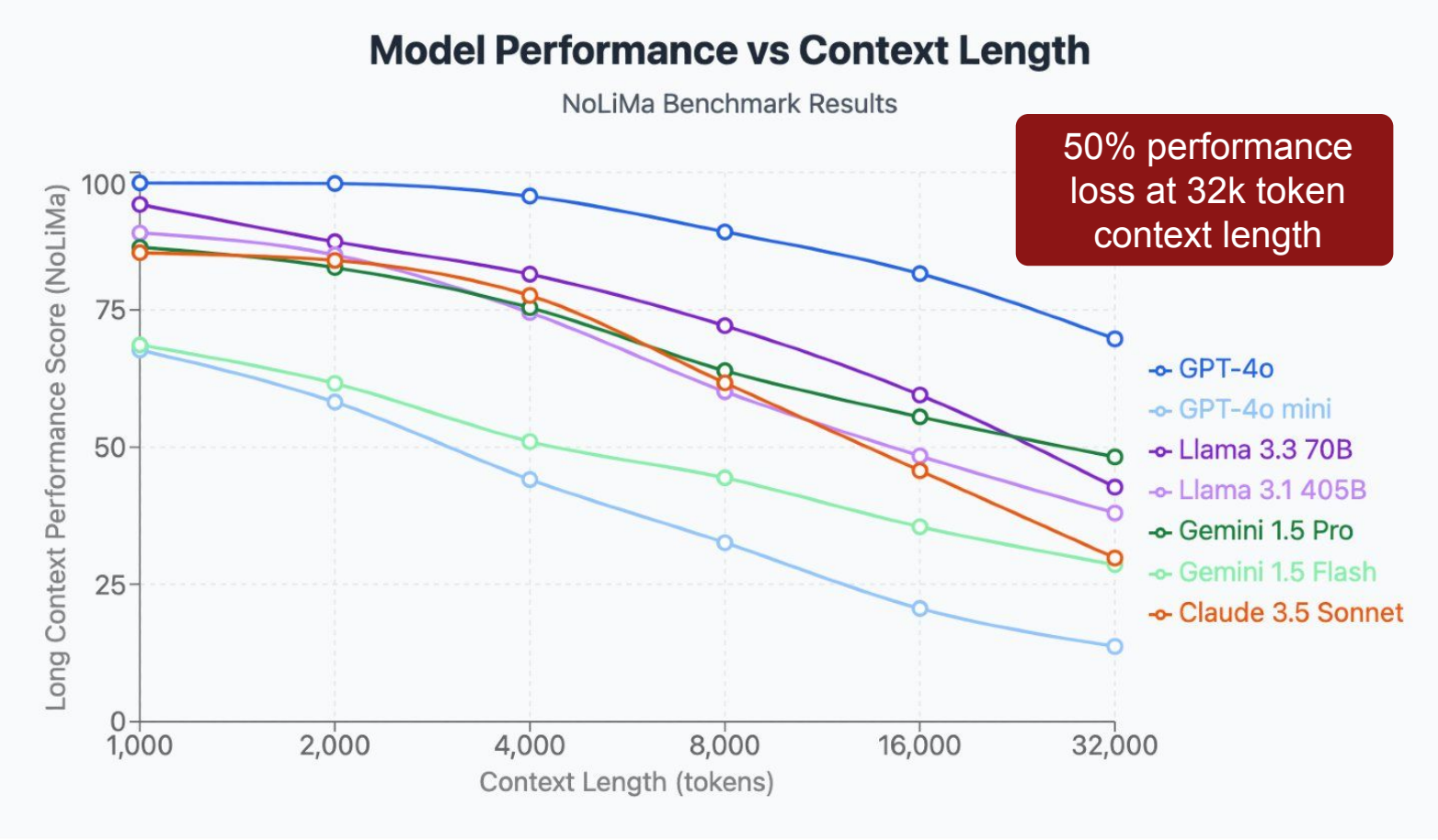


Data as of Q1 2025

AI boosts productivity by 10–25% in popular languages, but can *decrease* productivity in niche languages



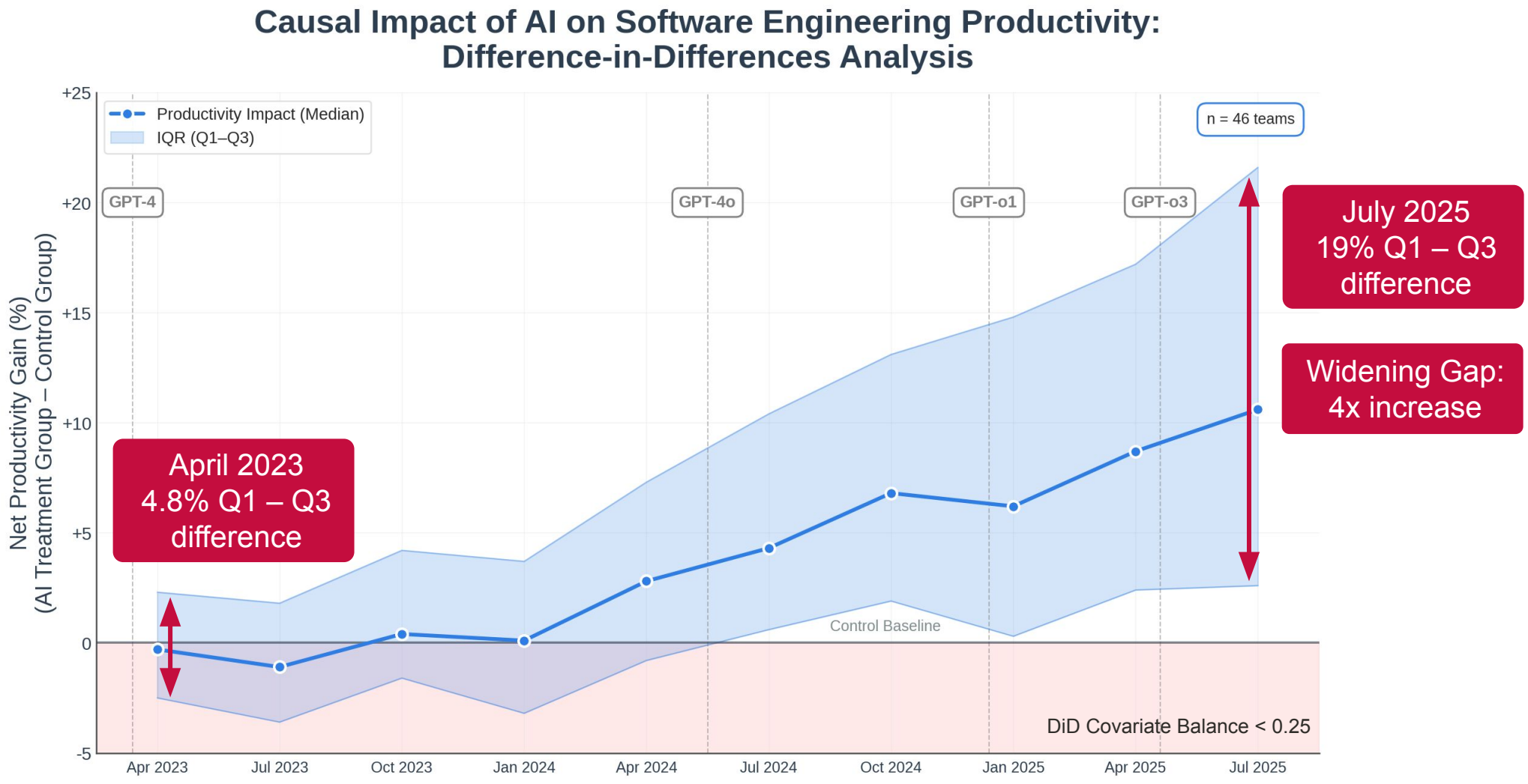
AI model coding performance drops with context length



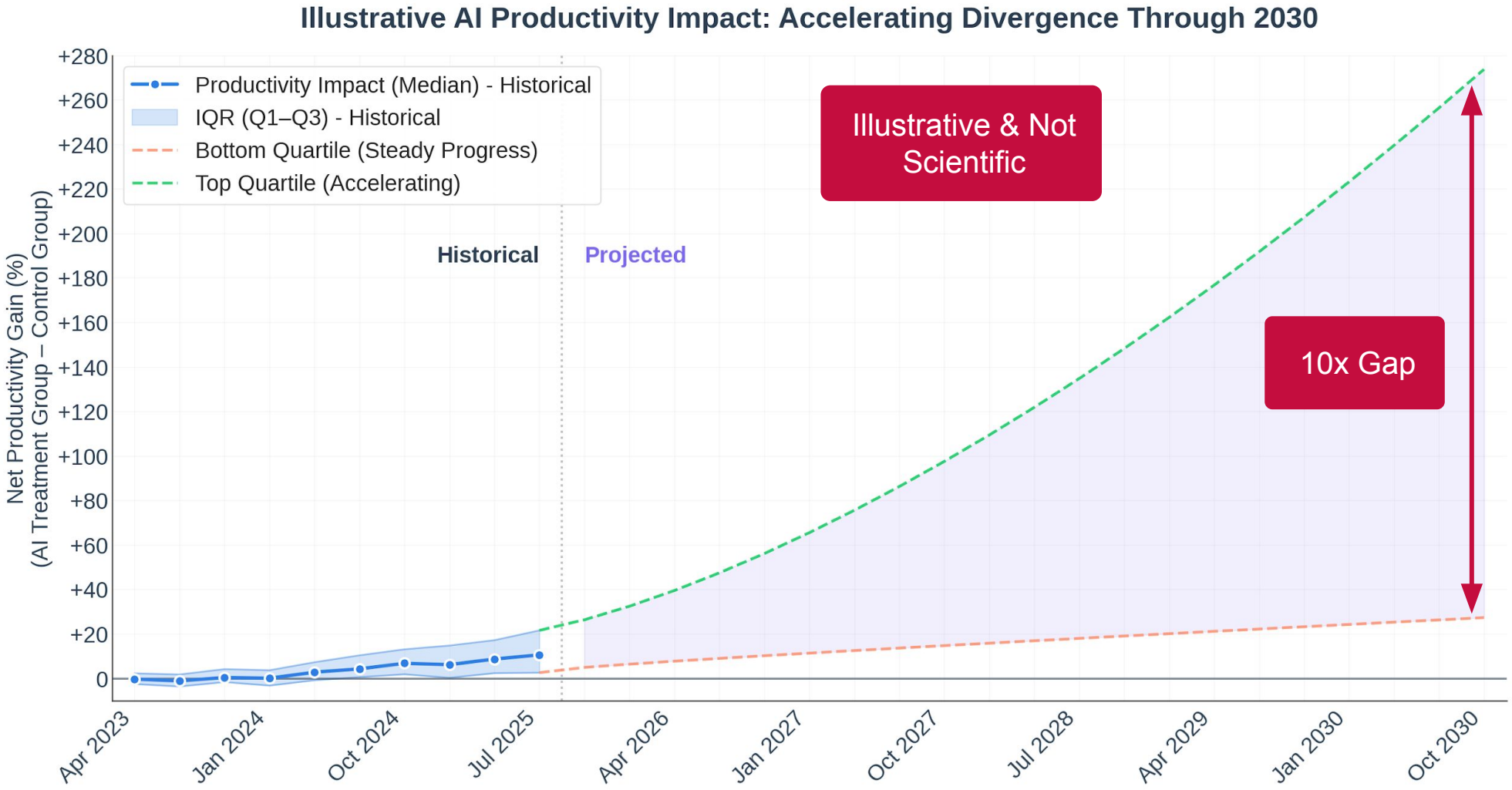
Model	Context Window (K)
GPT-4o	128
GPT-4o mini	128
Llama 3.3 70B	128
Llama 3.1 405B	128
Gemini 1.5 Pro	2,000
Gemini 1.5 Flash	1,000
Claude 3.5 Sonnet	200

Source: Modarressi A., Deilamsalehy H., Dernoncourt F., Bui T., Rossi R., Yoon S., & Schütze H. (2025). NOLIMA: Long-Context Evaluation Beyond Literal Matching. arXiv preprint arXiv:2502.05167 v2, 26 March 2025.

Teams that master AI are accelerating their productivity gains, widening the gap with laggards



“The rich get richer” – successful early AI adopters might compound their gains while strugglers could fall further behind



Companies are faced with 3 choices…

Replace Everyone

1

Reduce Engineering Headcount

- Demand for software will explode
- Won't be able to keep up

Spray AI & Pray

2

Freeze Headcount

Deploy AI w/o Control & Measurement

- Recipe for failure
- People resist new tools – learning curve

Most companies currently

Good Business Acumen

3

Deploy AI

Disciplined ROI, Measurement, Learning

- Learning how to deploy & use AI takes time
- Must champion a learning culture
- Hard to leapfrog from bottom quartile to top

What are top-quartile teams doing differently?

1 AI in Every Stage

- Not just IDE[1], but across SDLC[2]
- e.g. CI/CD[3], Testing, QA[4]

2 Multiple Parallel AI Agents

- Don't just prompt your AI agent and sit there watching it think
- Problem: context switching is hard

Most people currently

3 Rigorous Measurement & A/B Testing

- Measure AI deployments just like any other part of the business
- Turn AI deployments into scientific experiments

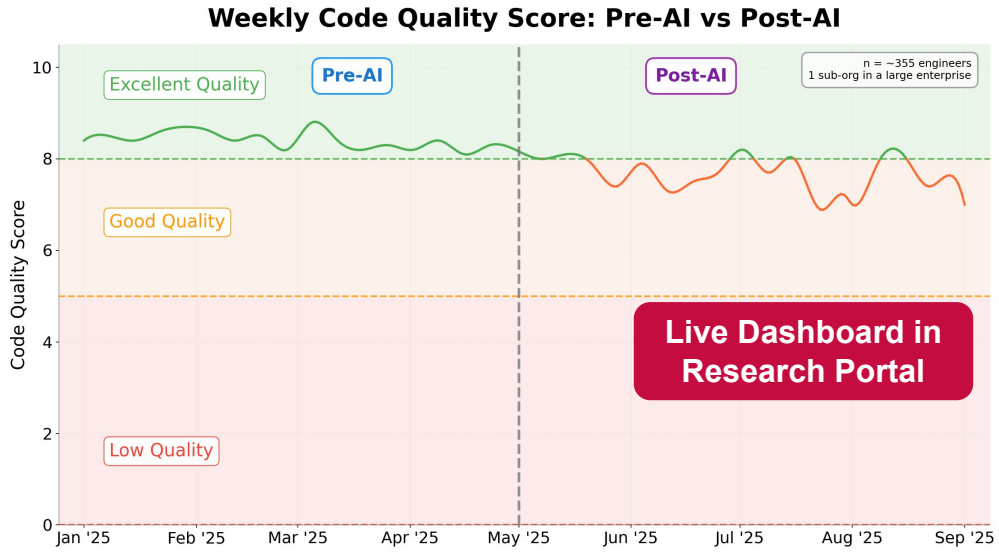
1) Integrated Development Environment; 2) Software Development Life Cycle; 3) Continuous Integration / Continuous Delivery; 4) Quality Assurance

Q: How close are we to replacing software engineers with AI?

A: Not very close

Would you like similar insights for your company?

Participate in our Research



Sign up at

softwareengineeringproductivity.stanford.edu

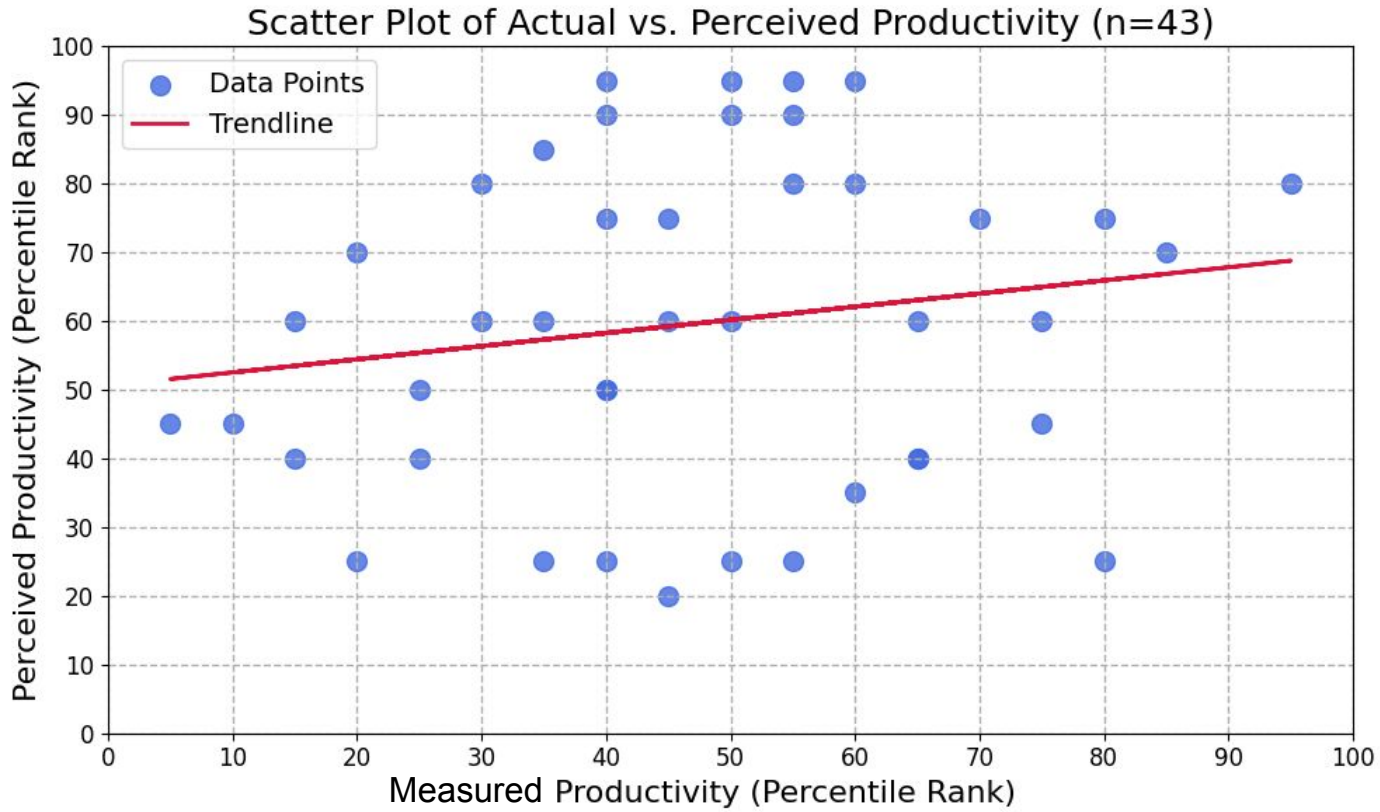
Yegor Denisov-Blanch

ydebl@stanford.edu

Acknowledgements: Simon Obstbaum, Michal Kosinski, Igor Ciobanu, Ion Manoil, Horatiu Mocian

Appendix

Self-assessment surveys are an inaccurate way to measure developer productivity



0.17
Correlation (r)

0.03
R^2

Self-assessment surveys (perceived productivity) are an ineffective predictor of productivity

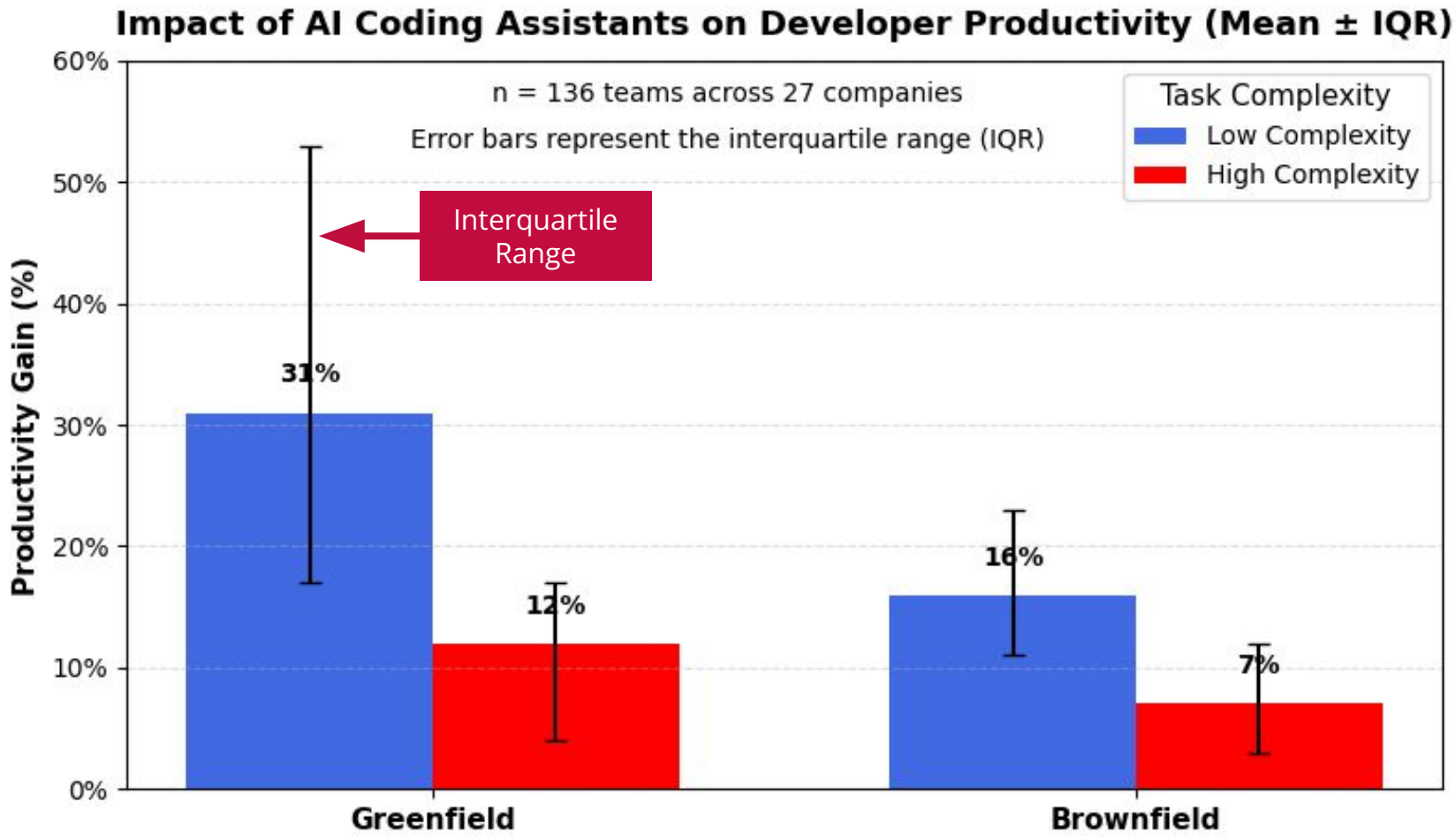
People misjudge their productivity by ~30 percentile points

Only **1 in 3** people estimated their productivity **within one quartile**

Surveys are valuable for understanding employee satisfaction and morale

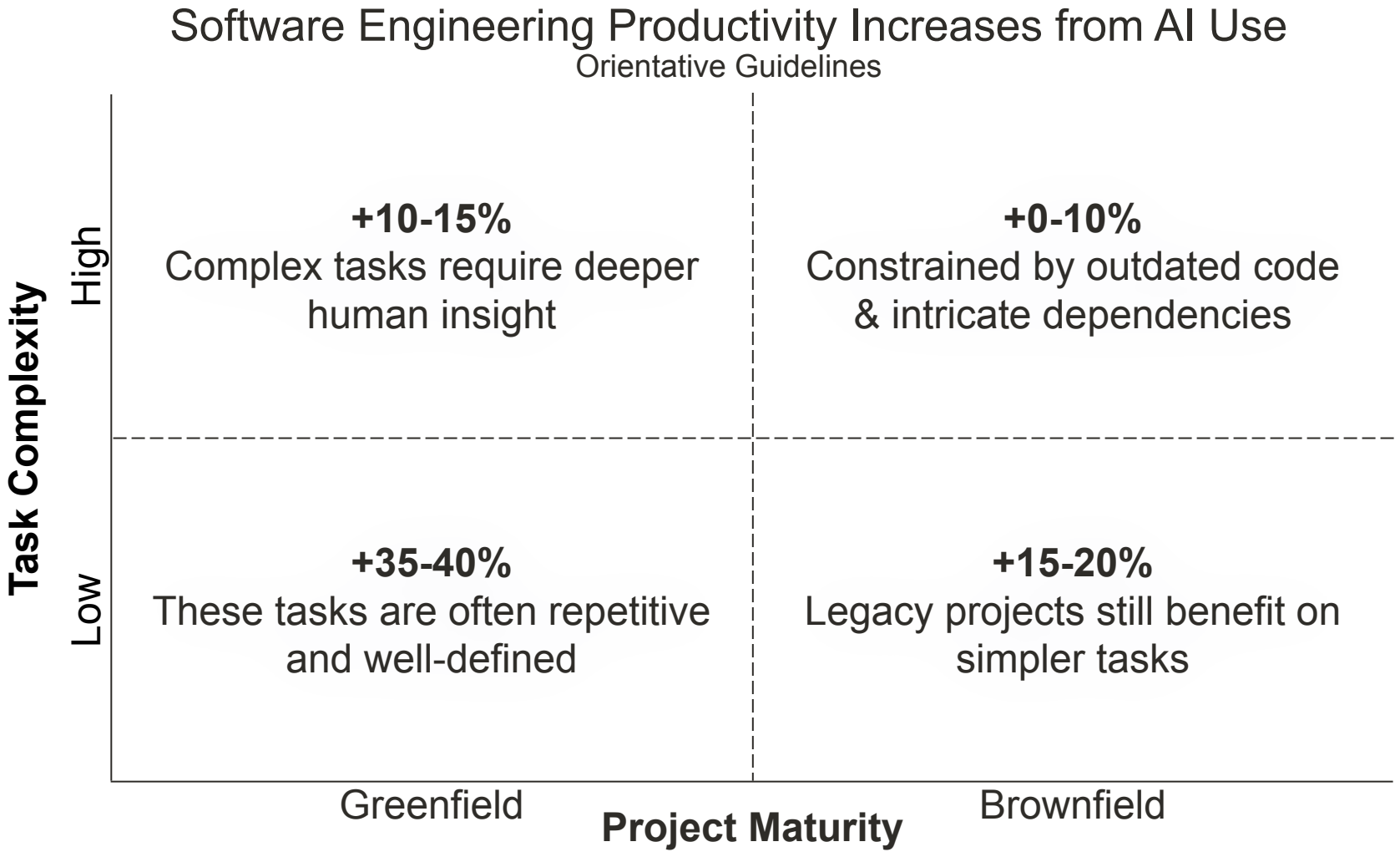
We surveyed 43 software engineers from a statistically representative sample, asking them to rate their productivity on a scale from 0 to 100 in 5-percentile increments, relative to the global average over the past year. We then compared these self-assessments with their actual performance, recorded over the same period, and rounded to the nearest 5 percentile.

Greenfield projects gain 30–35% on simple tasks and 10–15% on complex ones, versus 15–20% and 5–10% in brownfield projects

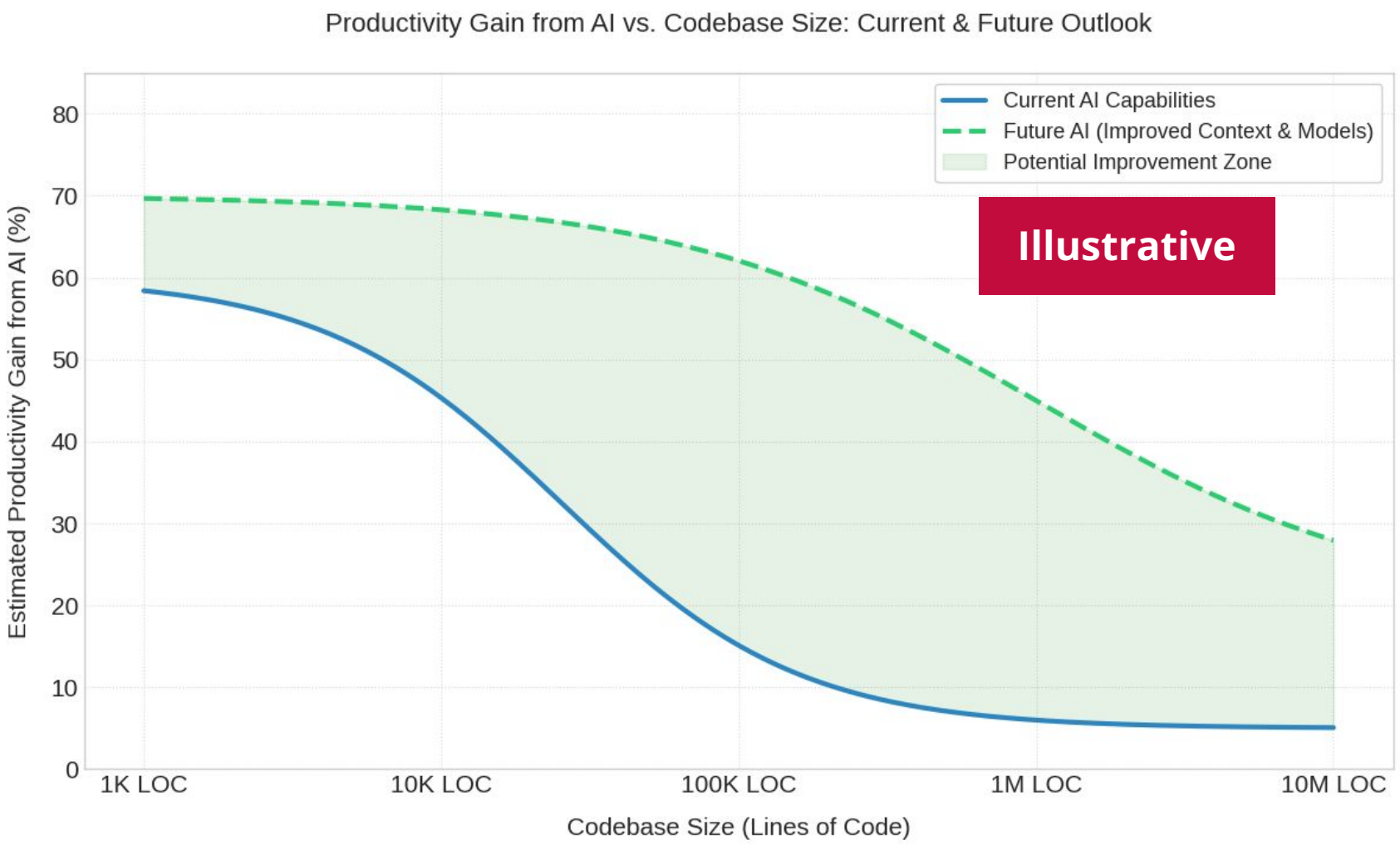


Research at Stanford University, Yegor Denisov-Blanch (ydebl@stanford.edu), softwareengineeringproductivity.stanford.edu

Greenfield projects gain 30–35% on simple tasks and 10–15% on complex ones, versus 15–20% and 5–10% in brownfield projects



As codebase size increases, productivity gain from AI decreases



Context Window Limitations

- Performance gains decrease with larger context windows

Signal:Noise Ratio

- Large codebases have more noise that can mislead the model

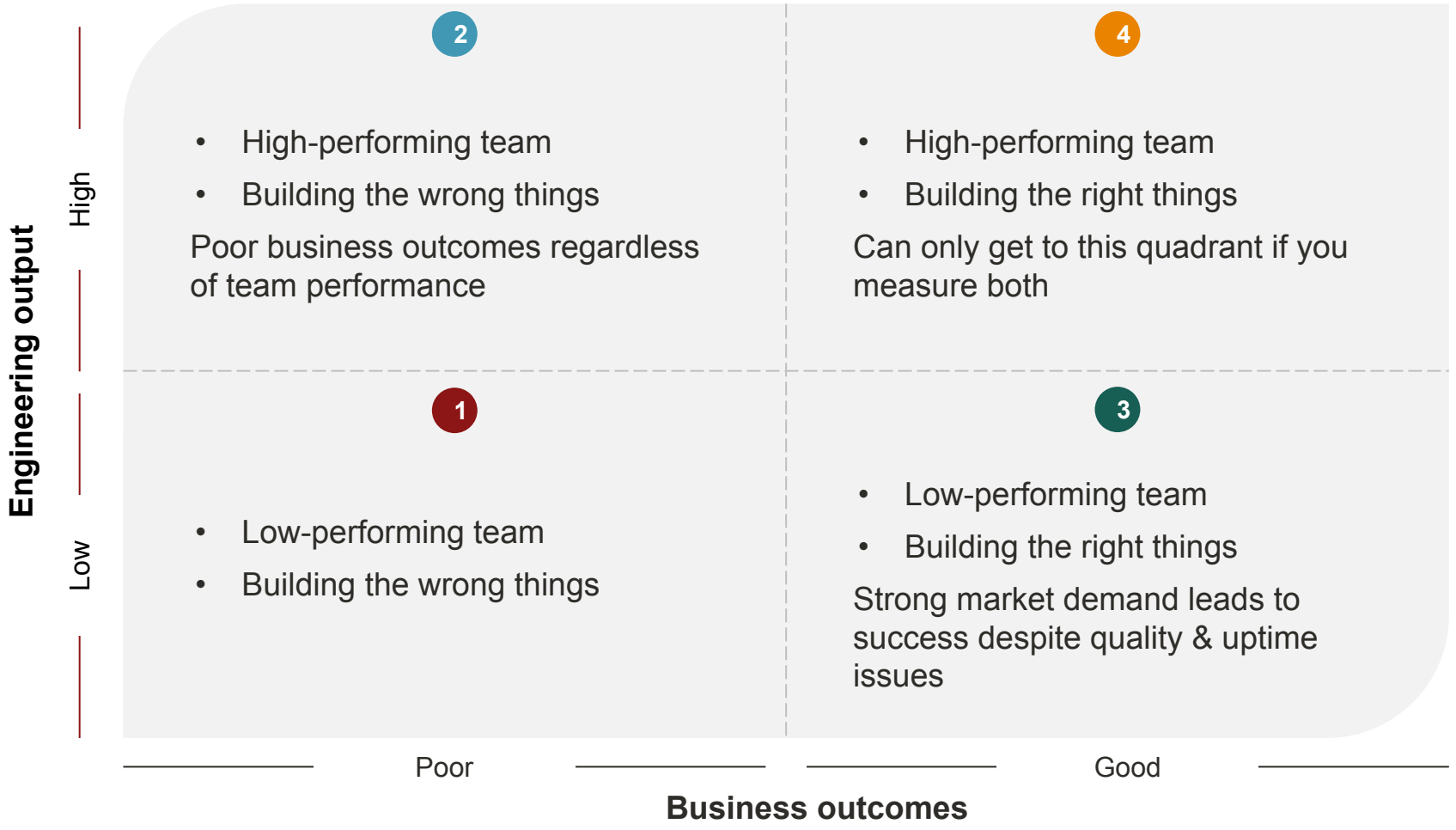
Complexity

- Dependencies
- Domain-specific logic

The difference between Output and Outcomes in Software Engineering

Output	Outcomes
Tangible work produced by engineers	Business results that stem from building the right things
Velocity, building things right	Feature prioritization

Measuring both output and outcomes is necessary to achieve a high-performing software org



1. Problematic
2. High output alone doesn't guarantee success; it needs to be aligned with building the right things
3. Room for improvement in execution
4. Best teams optimize both output and outcomes

Need to measure both output & outcomes

Our research focuses on output:

1

Easier to gather objective & comparable data across orgs

2

Product prioritization frameworks exist to drive “building the right things”

3

All else equal, high output is better than low output

1

Industry Benchmark: Performance

Industry Benchmark: Software Engineering Output

Industry SaaS & Web

Company Size Medium

Software Engineering Performance Benchmark

	Team	Overall	Senior	Mid	Junior
Product A	Frontend				
	Backend				
Product B	Frontend				
	Backend				
	Data				

Top 25% | 3rd Quartile | 2nd Quartile | Bottom 25%

1

Analyze software engineering performance

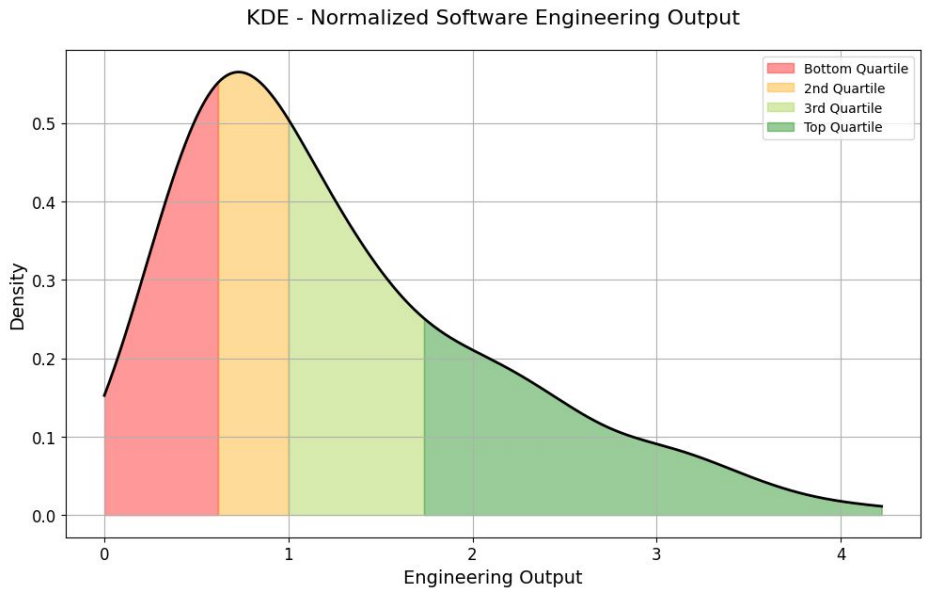
For a given time period:

$$\frac{\sum \text{Impact of Code Contributions}}{\text{Avg. \# Team Members}}$$

Using our model based on research conducted at Stanford

2

Benchmark against similar teams



2

Industry Benchmark: Cost Efficiency

Analyze results & make data-driven decisions

Industry SaaS & Web

Company Size Medium

Software Engineering Cost Efficiency Benchmark

	Team	Overall	Senior	Mid	Junior
Product A	Frontend				
	Backend				
Product B	Frontend				
	Backend				
	Data				

Top 25% | 3rd Quartile | 2nd Quartile | Bottom 25%

1 **Calculate cost efficiency**

For a given time period:

$$\frac{\sum \text{Impact of Code Contributions}}{\text{Avg. Cost of Team}}$$

Using our model based on research conducted at Stanford

2 **Benchmark against similar teams**

Industry | **Company Size** | **Tech Stack**

Location Agnostic