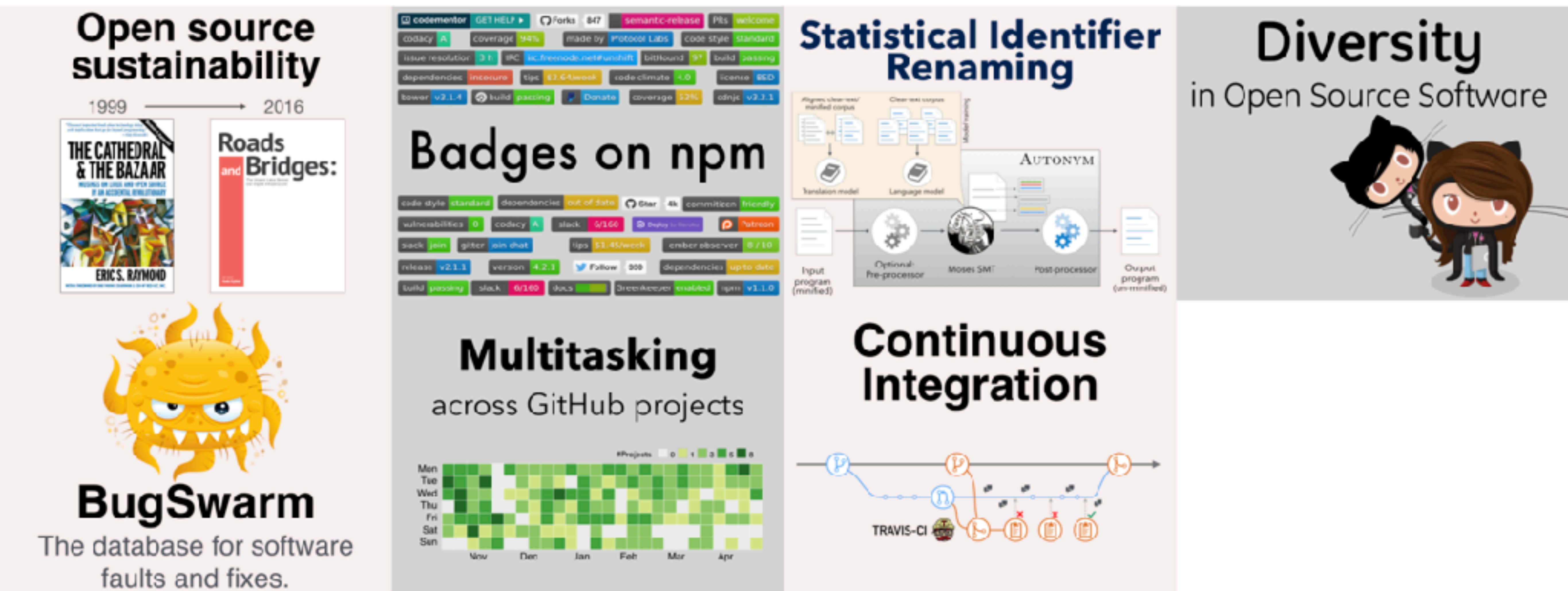
Data Science for Software Engineering

Bogdan Vasilescu

Intro

- 2009 - 2014: MSc & PhD, TU Eindhoven
- 2014 - 2016: Postdoc, UC Davis
- 2016 - : Assistant Professor, CMU ISR
    - Software analytics research lab
      https://cmustrudel.github.io/

STRUDEL
SOCIO-TECHNICAL RESEARCH
USING DATA EXCAVATION LAB



Today

- First session:
  - Intro: the **Science** of Software Engineering
  - Hands-on: segmented regression analysis of interrupted time series data

- Second session:
  - Intro: the Naturalness of Software theory
  - Hands-on: language modeling

Today

- First session:
  - Intro: the **Science** of Software Engineering
  - Hands-on: segmented regression analysis of interrupted time series data
- Second session:
  - Intro: the Naturalness of Software theory
  - Hands-on: language modeling

Many slides thanks to:

- Thomas Zimmermann, Microsoft Research:
  https://speakerdeck.com/tomzimmermann

- Greg Wilson, Mozilla
  https://www.slideshare.net/gvwilson/presentations

- Laurie Williams, NC State
  https://www.slideshare.net/laurieannwilliams/writing-good-software-engineering-research-papers-revisited

- Prem Devanbu, UC Davis
  https://www.slideshare.net/pdevanbu/beliefevidenceicse

- Steve Easterbrook, U Toronto
  http://www.cs.uoregon.edu/events/fse14/docsym_docs/FSE06DocSymp-keynote-v5.pdf

Once Upon a Time...

Seven Years' War (1754-63)

Britain loses 1,512 sailors to enemy action...

...and almost 100,000 to scurvy

Oh, the Irony

James Lind (1716-94)

1747: (possibly) the first-ever controlled medical experiment

- × cider
- × sulfuric acid
- × vinegar
- × sea water
- √**oranges**
- ×barley water

No-one paid attention until a proper Englishman repeated the experiment in 1794...

Like Water on Stone

1992: Sackett coins the term “evidence-based medicine”

Randomized double-blind trials are accepted as the gold standard for medical research

The Cochrane Collaboration (http://www.cochrane.org/) now archives results from hundreds of medical studies

What about Software Engineering?

What metrics are the **best predictors of failures**?

If I increase **test coverage**, will that actually increase software quality?

What is the **data quality** level used in empirical studies and how much does it actually matter?

Are there any **metrics that are indicators of failures** in both Open Source and Commercial domains?

I just submitted a **bug report**. Will it be fixed?

Should I be writing **unit tests** in my software project?

How can I tell if a piece of software will have **vulnerabilities**?

Do **cross-cutting concerns** cause defects?

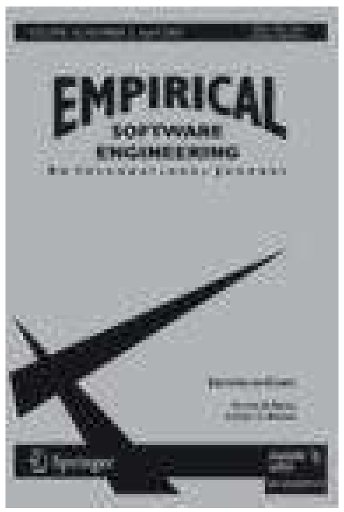
Is strong **code ownership** good or bad for software quality?

Does **Distributed/Global software development** affect quality?

Does **Test Driven Development** (TDD) produce better code in shorter time?

Software Engineering is becoming more like modern medicine, i.e., evidence-based

The Times They Are A-Changin'



Growing emphasis on empirical studies in software engineering research since the mid-1990s

Papers describing new tools or practices routinely include results from some kind of field study

ICSE 2009
VANCOUVER

Yes, many are flawed or incomplete, but standards are constantly improving

Contributions (RQ2)

Types of research contribution in ICSE 2016 submissions and acceptances						
Type of contribution	Submitted (2002)	Submitted (2016)	Accepted (2002)	Accepted (2016)	Ratio (2002)	Ratio (2016)
Procedure or technique	152 (44%)	195 (37%)	28 (51%)	35 (35%)	18%	18%
Qualitative or descriptive model	50 (14%)	22 (4%)	4 (7%)	4 (4%)	8%	18%
Empirical model	4 (1%)	29 (5%)	1 (2%)	5 (5%)	25%	17%
Analytic model	48 (14%)	54 (10%)	7 (13%)	8 (8%)	15%	15%
Tool or notation	49 (14%)	83 (16%)	10 (18%)	16 (16%)	20%	19%
Specific solution	34 (10%)	14 (3%)	5 (9%)	2 (2%)	15%	14%
Empirical Report	11 (3%)	103 (19%)	0 (0%)	31 (31%)	0%	**30%**

Validation (RQ3)

Types of Validation in ICSE 2016 submissions and acceptances						
Type of result	Submitted (2002)	Submitted (2016)	Accepted (2002)	Accepted (2016)	Ratio (2002)	Ratio (2016)
Analysis	48 (16%)	72 (14%)	11 (26%)	19 (19%)	23%	26%
Evaluation	21 (7%)	188 (35%)	1 (2%)	65 (64%)	5%	35%
Experience	34 (11%)	19 (4%)	8 (19%)	4 (4%)	24%	21%
Example	82 (27%)	61 (12%)	16 (37%)	1 (1%)	20%	2%
Underspecified	6 (2%)	94 (18%)	1 (2%)	11 (11%)	17%	12%
Persuasion	25 (8%)	37 (7%)	0 (0%)	1 (1%)	0%	3%
No validation	84 (28%)	31 (6%)	6 (14%)	0 (0%)	7%	0%

Analysis/Evaluation/Experience becoming ICSE requirement compared to 2002

Q: What enabled this?

A: Data science played a big role

Aside:
Do we <u>really</u> need evidence?

"We hold these Truths to be **self-evident**, …"

Engineering software is inherently a human venture

My Favorite Little Result

Aranda & Easterbrook (2005): “Anchoring and Adjustment in Software Estimation”

“How long do you think it will take to make a change to this program?”

Control Group: “*I’d like to give an estimate for this project myself, but I admit I have no experience estimating. We’ll wait for your calculations for an estimate.”*

Group A: *“I admit I have no experience with software projects, but I guess this will take about 2 months to finish.*”

Group B: *“...I guess this will take about 20 months...*”

Results

Group A (lowball)	5.1 months
Control Group	7.8 months
Group B (highball)	15.4 months

The anchor mattered more than experience, how formal the estimation method was, or anything else.

40 percent of major decisions are based not on facts, but on the **manager's gut**.

Accenture survey among 254 US managers in industry.
http://newsroom.accenture.com/article_display.cfm?article_id=4777

Microsoft Research

Opinion Source

Devanbu, P., Zimmermann, T., & Bird, C. (2016, May). Belief & evidence in empirical software engineering. In *Proceedings of the 38th international conference on software engineering* (pp. 108-119). ACM.

Opinion Source

Code quality (defect occurrence) depends on which programming language is used.

1. *Strongly Agree*
2. *Agree*
3. *Neutral*
4. *Disagree*
5. *Strongly Disagree*

Devanbu, P., Zimmermann, T., & Bird, C. (2016, May). Belief & evidence in empirical software engineering. In *Proceedings of the 38th international conference on software engineering* (pp. 108-119). ACM.

Opinion Source

Code quality (defect occurrence) depends on which programming language is used.

1. *Strongly Agree*
2. *Agree*
3. *Neutral*
4. *Disagree*
5. *Strongly Disagree*

Devanbu, P., Zimmermann, T., & Bird, C. (2016, May). Belief & evidence in empirical software engineering. In *Proceedings of the 38th international conference on software engineering* (pp. 108-119). ACM.

Opinion Source

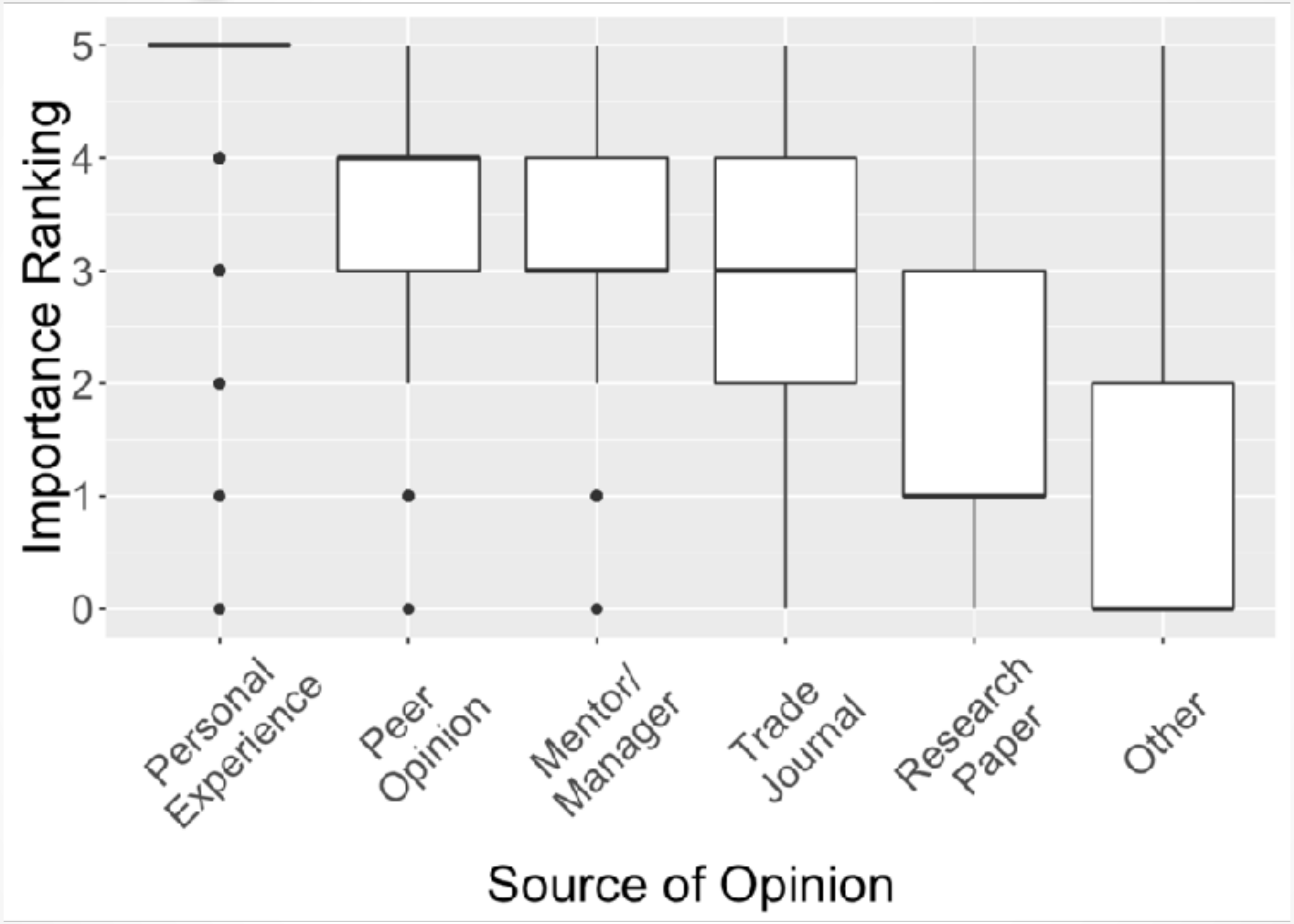
Code quality (defect occurrence) depends on which programming language is used.

1. *Strongly Agree*
2. *Agree*
3. *Neutral*
4. *Disagree*
5. *Strongly Disagree*

Where do they originate?

Devanbu, P., Zimmermann, T., & Bird, C. (2016, May). Belief & evidence in empirical software engineering. In *Proceedings of the 38th international conference on software engineering* (pp. 108-119). ACM.

Opinion Formation



Devanbu, P., Zimmermann, T., & Bird, C. (2016, May). Belief & evidence in empirical software engineering. In *Proceedings of the 38th international conference on software engineering* (pp. 108-119). ACM.

Another example:

Perl - low entry barrier

The Biggest Challenge

http://tinyurl.com/nwit-randomo

Stefik et al: “An Empirical Comparison of the Accuracy Rates of Novices using the Quorum, Perl, and Randomo Programming Languages.” *PLATEAU'11*

We present here an empirical study comparing the accuracy rates of novices writing software in three programming languages: Quorum, Perl, and Randomo. The first language, Quorum, we call an evidence-based programming language, where the syntax, semantics, and API designs change in correspondence to the latest academic research and literature on programming language usability. Second, while Perl is well known, we call Randomo a Placebo-language, where some of the syntax was chosen with a random number generator and the ASCII table. We compared novices that were programming for the first time using each of these languages, testing how accurately they could write simple programs using common program constructs (e.g., loops, conditionals, functions, variables, parameters). Results showed that while Quorum users were afforded significantly greater accuracy compared to those using Perl and Randomo, Perl users were unable to write programs more accurately than those using a language designed by chance.

Empirical studies are hard to get right

Sobel, A. E. K., & Clarkson, M. R. (2002). Formal methods application: An empirical tale of software development. *IEEE Transactions on Software Engineering*, *28*(3), 308-320.

- Two classes of students at Miami University of Ohio that studied object-oriented (OO) design in a one semester course:
  - Control group (random sample): OO design class
  - Treatment group (volunteers): OO design class + formal methods
    - No statistical difference between the abilities of the two groups on standardized ACT pre-tests
- As project, both classes were assigned the development of an elevator system
  - Hand in functioning executable + source code (+ formal specification written using first-order logic)

Sobel, A. E. K., & Clarkson, M. R. (2002). Formal methods application: An empirical tale of software development. *IEEE Transactions on Software Engineering*, *28*(3), 308-320.

- Standard set of test cases:
  - 45.5% of control teams passed all tests
  - 100% of treatment teams
- Conclusions:
  - “formal methods students had increased complex-problem solving skills”
  - “the use of formal methods during software development produces ‘better’ programs”

Berry, D. M., & Tichy, W. F. (2003). Comments on" Formal methods application: an empirical tale of software development". *IEEE Transactions on Software Engineering*, *29*(6), 567-571.

- "Unfortunately, the paper contains several subtle problems. The reader unfamiliar with the basic principles of experimental psychology may easily miss them and interpret the results incorrectly. Not only do we wish to point out these problems, but we also aim to illustrate what to look for when drawing conclusions from controlled experiments."

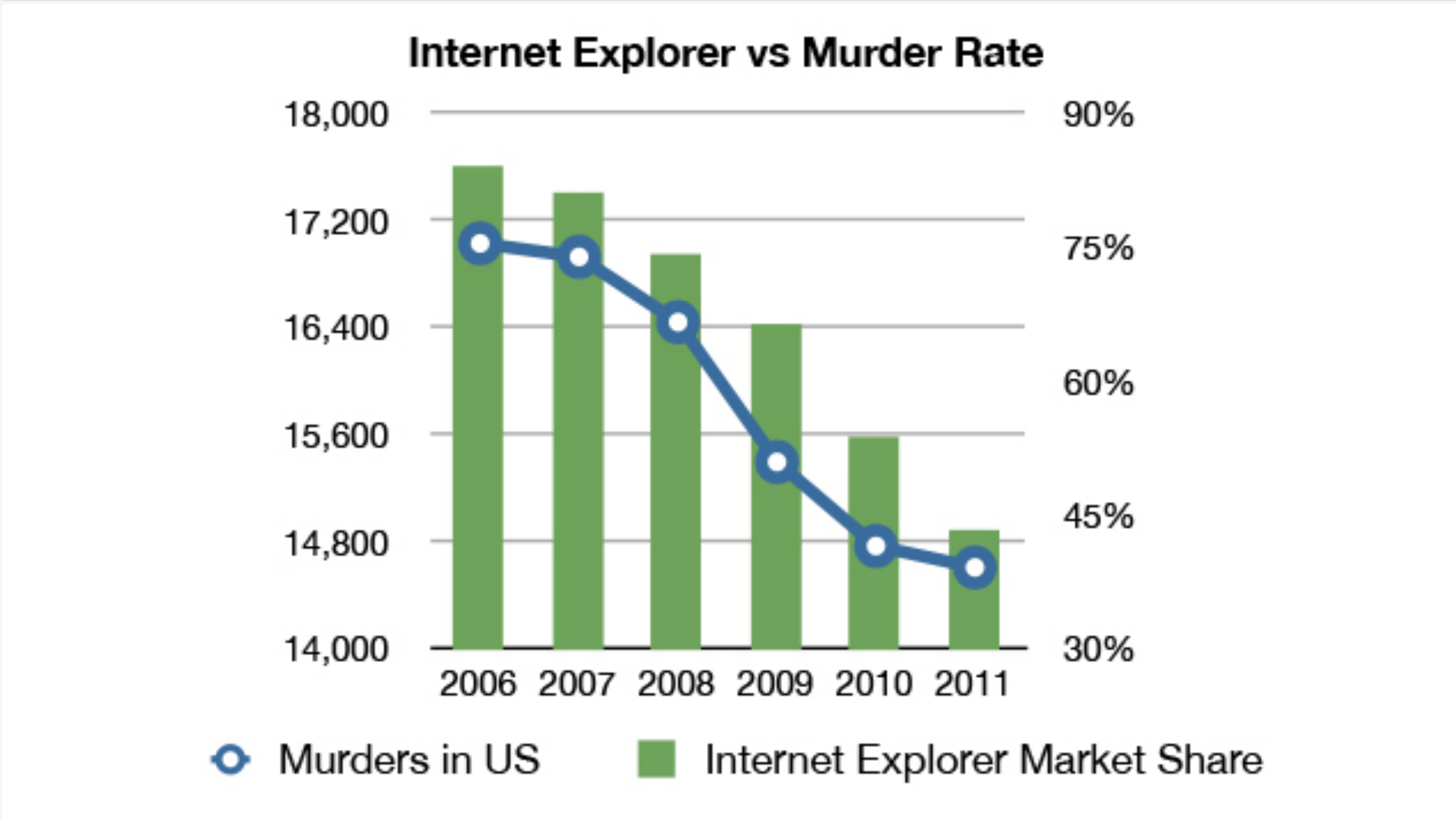
Berry, D. M., & Tichy, W. F. (2003). Comments on" Formal methods application: an empirical tale of software development". *IEEE Transactions on Software Engineering*, *29*(6), 567-571.

- Confounding variables:
  - differences in motivation (treatment group volunteers more motivated)
  - differences in exposure (treatment group more instruction)
  - differences in learning style (treatment group better learners)
  - differences in skills (outside of ACT)
- Novelty effects
- …

Why big data needs thick data

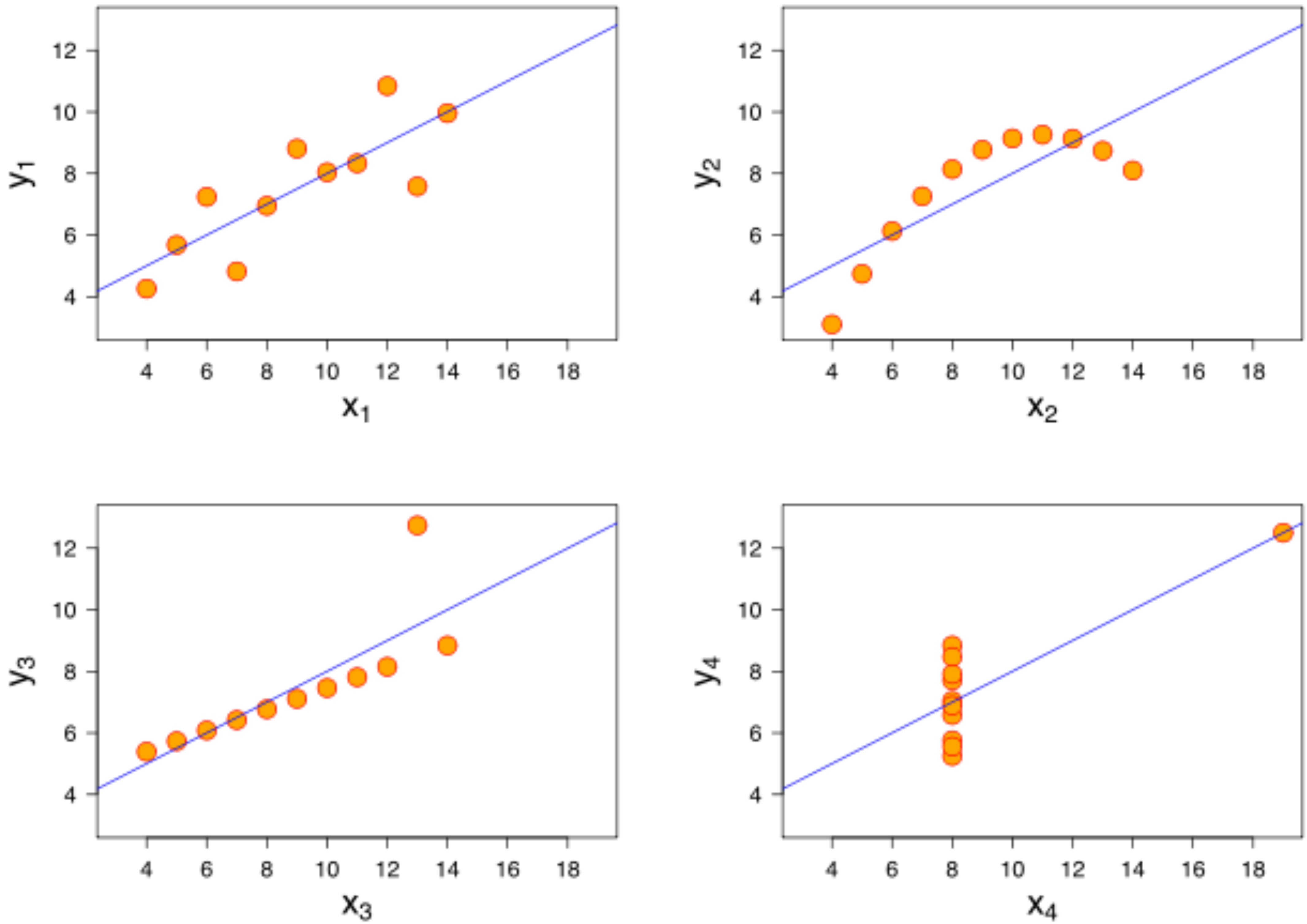
Credits: M.-A.Storey, “Lies, damned lies, and analytics: Why big data needs thick data”

“Data is like people – interrogate it hard enough and it will tell you whatever you want to hear”



Internet Explorer vs Murder Rate
18,000
17,200
16,400
15,600
14,800
14,000
90%
75%
60%
45%
30%
2006
2007
2008
2009
2010
2011
Murders in US
Internet Explorer Market Share

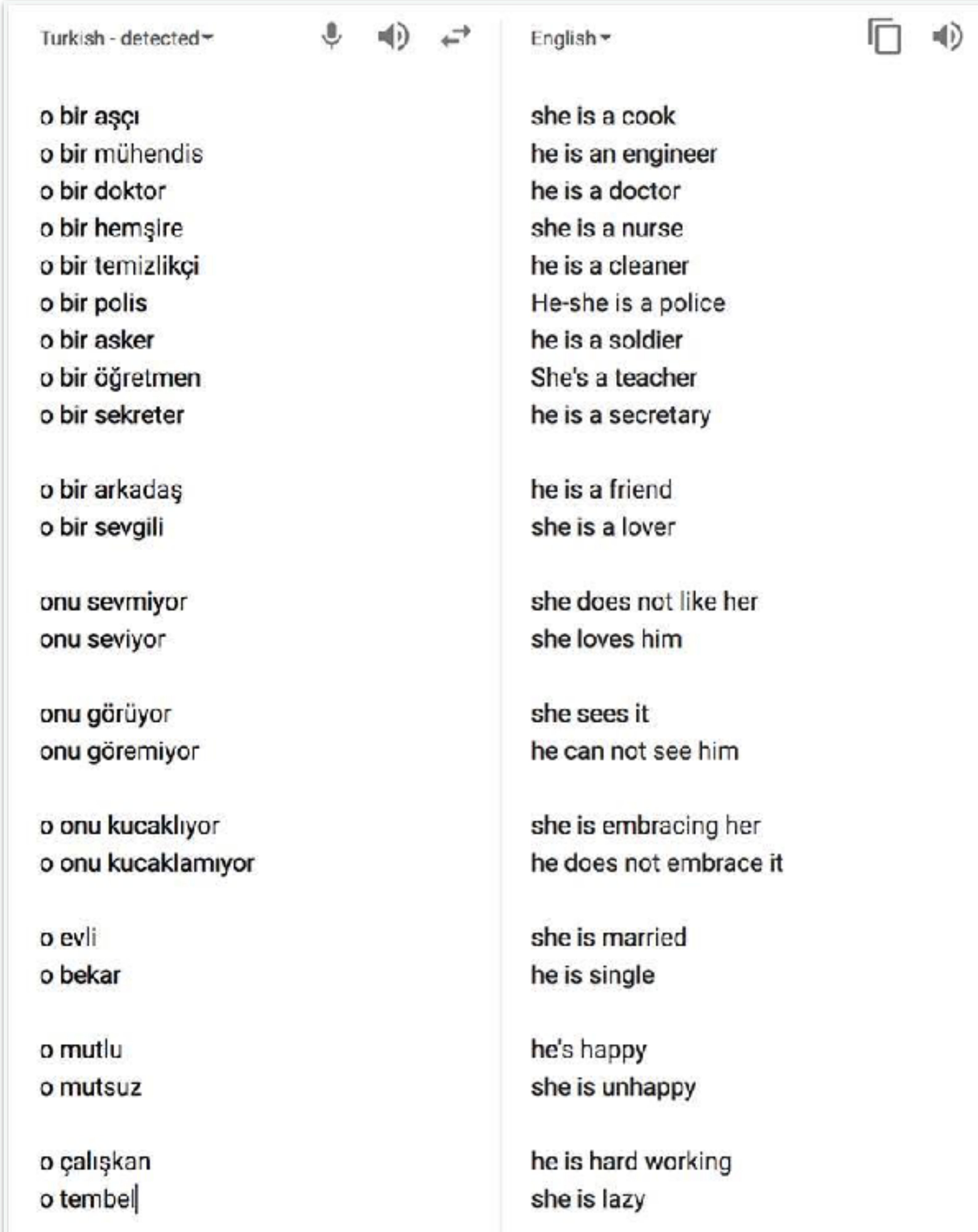
Anscombe's quartet



Percentage of women in top 100 Google image search results for CEO: 11%
Percentage of U.S. CEOs who are women: 27%

Percentage of women in the top 100 Google image search results for telemarketers: 64%
Percentage of U.S. telemarketers who are women: 50%

Kay, M., Matuszek, C., & Munson, S. A. (2015, April). Unequal representation and gender stereotypes in image search results for occupations. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (pp. 3819-3828). ACM.


Turkish - detected
English
o bir aşçı
o bir mühendis
o bir doktor
o bir hemşire
o bir temizlikçi
o bir polis
o bir asker
o bir öğretmen
o bir sekreter
o bir arkadaş
o bir sevgili
onu sevmiyor
onu seviyor
onu görüyor
onu göremiyor
o onu kucaklıyor
o onu kucaklamıyor
o evli
o bekar
o mutlu
o mutsuz
o çalışkan
o tembel
she is a cook
he is an engineer
he is a doctor
she is a nurse
he is a cleaner
He-she is a police
he is a soldier
She's a teacher
he is a secretary
he is a friend
she is a lover
she does not like her
she loves him
she sees it
he can not see him
she is embracing her
he does not embrace it
she is married
he is single
he's happy
she is unhappy
he is hard working
she is lazy

Data Science for SE:

- We need appropriate research methods, applied rigorously
- But also:

You Gotta Have A Theory

Steve Easterbrook

sme@cs.toronto.edu

www.cs.toronto.edu/~sme

Science and Theory

- A (scientific) theory is:
  - more than just a description - it explains and predicts
  - Logically complete, internally consistent, falsifiable
  - Simple and elegant.
- Components of a theory:
  - concepts, relationships, causal inferences
    - E.g. Conway's Law- structure of software reflects the structure of the team that builds it. A theory should explain why.
- Theories lie at the heart of what it means to do science.
  - Production of generalizable knowledge
  - Scientific method ⇔ Research Methodology ⇔ Proper Contributions for a Discipline
- Theory provides orientation for data collection
  - Cannot observe the world without a theoretical perspective

The Role of Theory Building

- Theories allow us to compare similar work
  - Theories include precise definition for the key terms
  - Theories provide a rationale for which phenomena to measure
- Theories support analytical generalization
  - Provide a deeper understanding of our empirical results
  - ...and hence how they apply more generally
  - Much more powerful than statistical generalization
- ...but in SE we are very bad at stating our theories
  - Our vague principles, guidelines, best practices, etc. could be strengthened into theories
  - Every tool we build represents a theory

Theories are good for generalization...

Statistical Generalization

- First level generalization:
  - From sample to population
- Well understood and widely used in empirical studies
- Can only be used for quantifiable variables
- Based on random sampling:
  - Standard statistical tests tell you if results on a sample apply to the whole population
- Not useful when:
  - You can't characterize the population
  - You can't do random sampling
  - You can't get enough data points

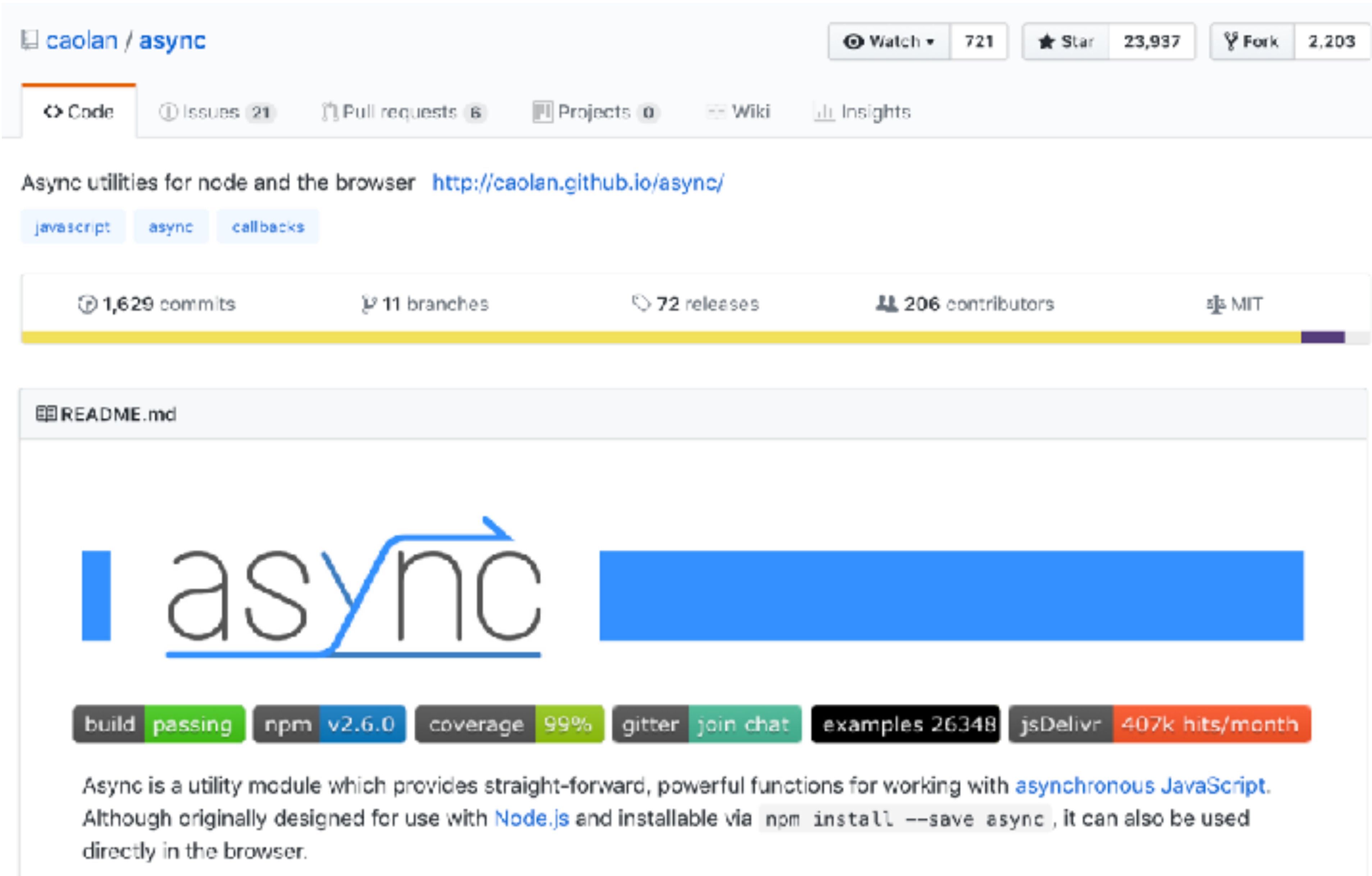
Analytical Generalization

- Second level generalization:
  - From findings to theory
- Applicable to quantitative and qualitative studies
- Compares findings with theory
  - Do the data support or refute the theory?
  - Or: do they support this theory better than rival theories?
- Supports empirical induction:
  - Evidence builds if subsequent studies also support the theory (& fail to support rival theories)
- More powerful than stats
  - Doesn't rely on correlations
  - Examines underlying mechanisms

Today

- First session:
  - Intro: the **Science** of Software Engineering
  - Hands-on: segmented regression analysis of interrupted time series data
- Second session:
  - Intro: the Naturalness of Software theory
  - Hands-on: language modeling

GitHub Repository Badges



Enlarged to show detail.

Trockman, A., Zhou, S., Kästner, C., and Vasilescu, B.
Adding Sparkle to Social Coding: An Empirical Study of Repository Badges in the npm Ecosystem.
International Conference on Software Engineering, ICSE, ACM (2018), 511–522.

Theory fragments

- Projects that adopt dependency management badges have “fresher” dependencies
  - because developers act on the warnings generated by their dependency management tool
  - because out-of-date dependencies would reflect negatively on their project

dependencies | up to date   dependencies | out of date

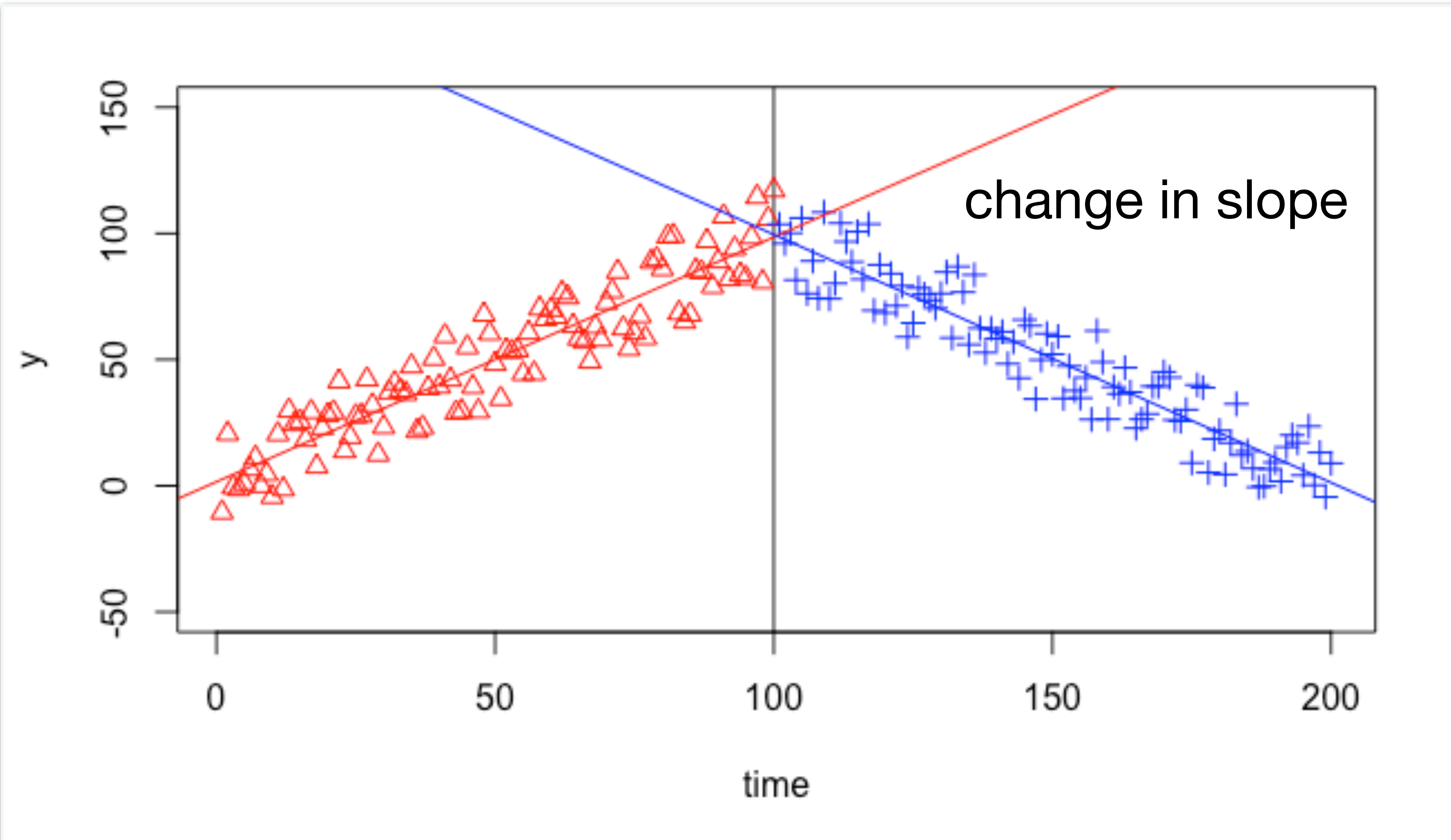
- Badges with underlying analyses are stronger predictors than badges that merely state intentions or provide links

npm | v1.1.0

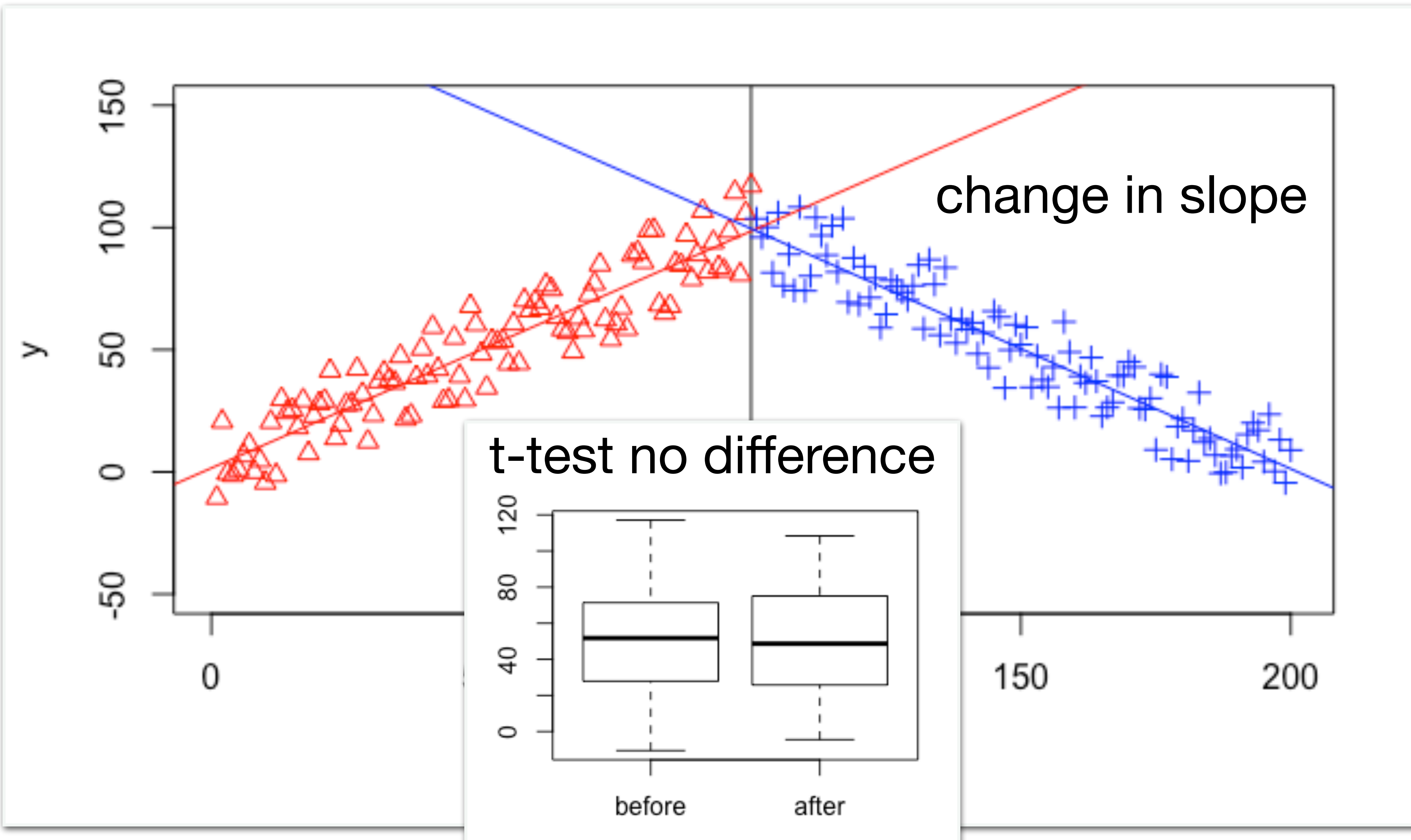
How to test?

- Idea: consider the badge adoption as an “intervention”
- Analyze the time series of dependency freshness
- Compare before vs after intervention

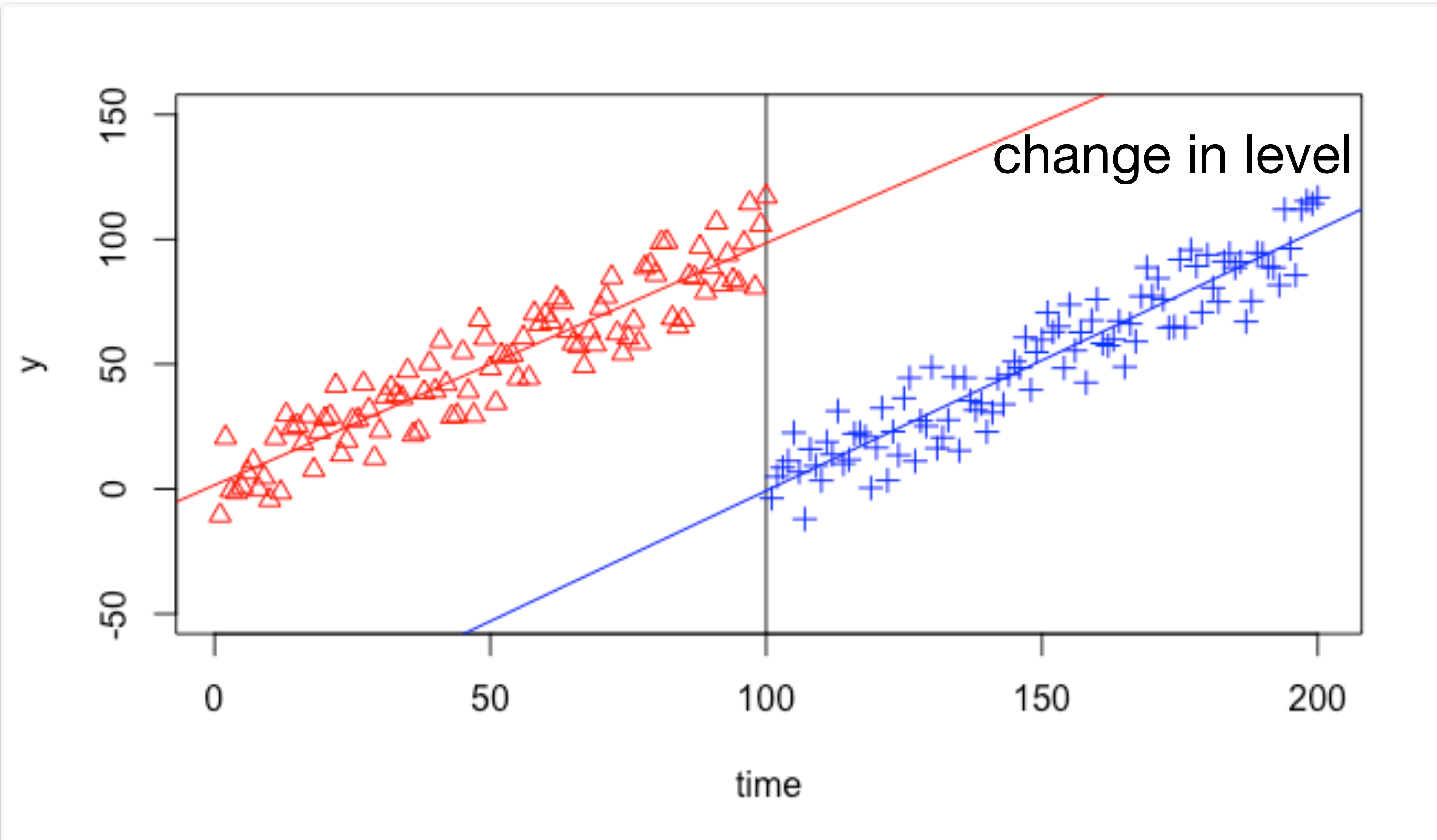
Evaluating the effects of an intervention



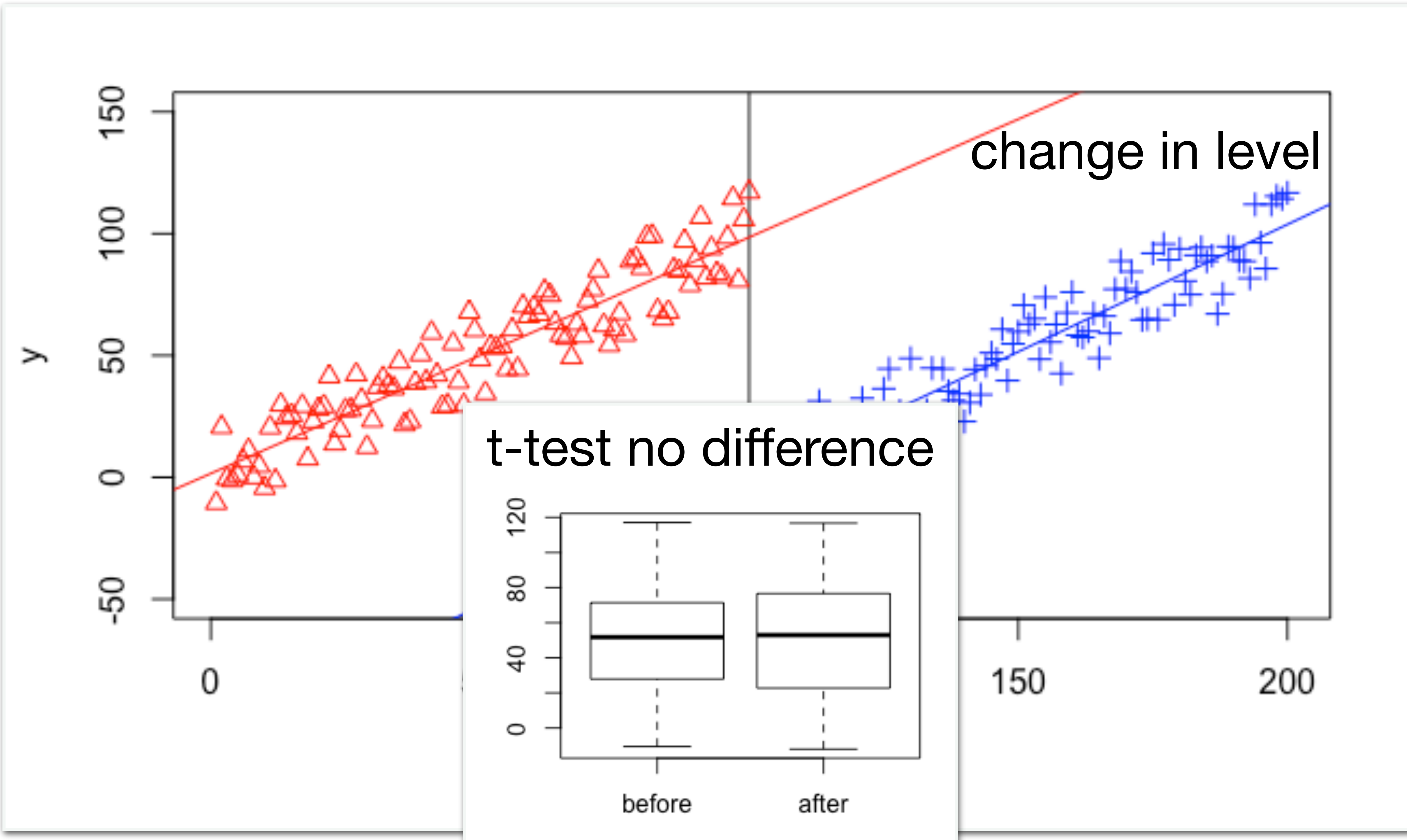
Evaluating the effects of an intervention



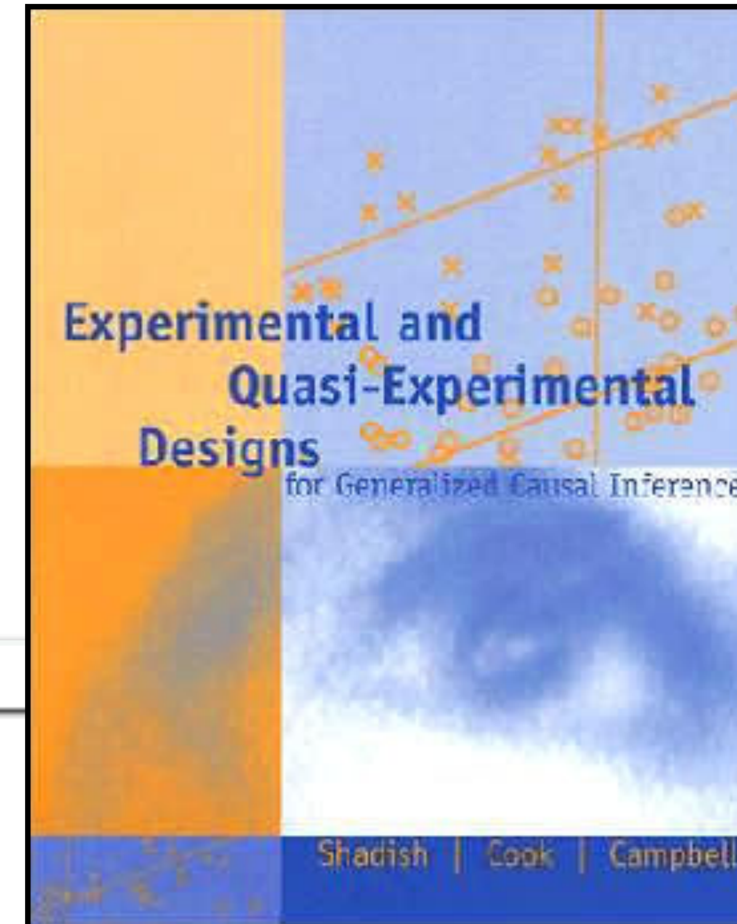
Evaluating the effects of an intervention

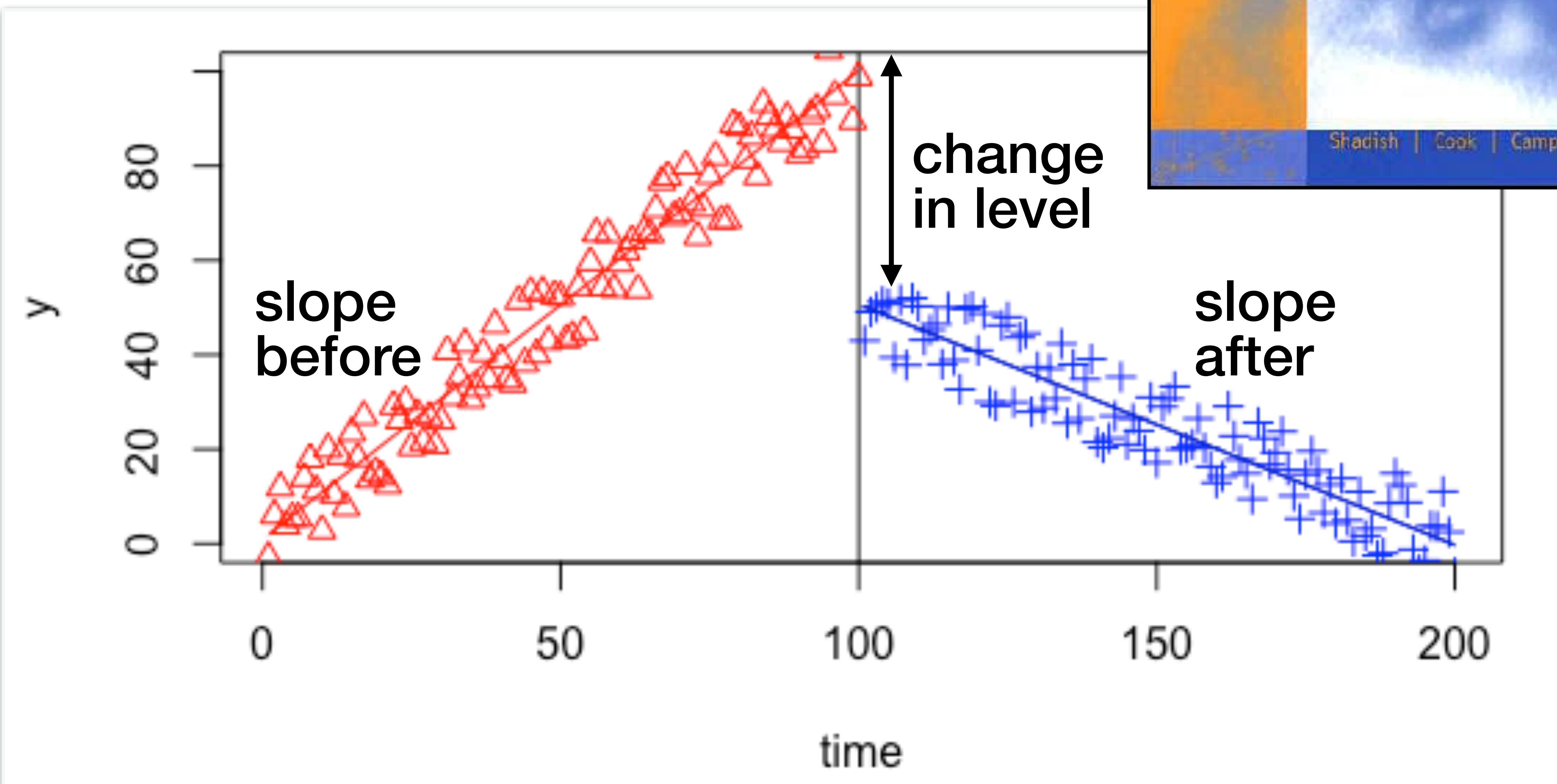


Evaluating the effects of an intervention



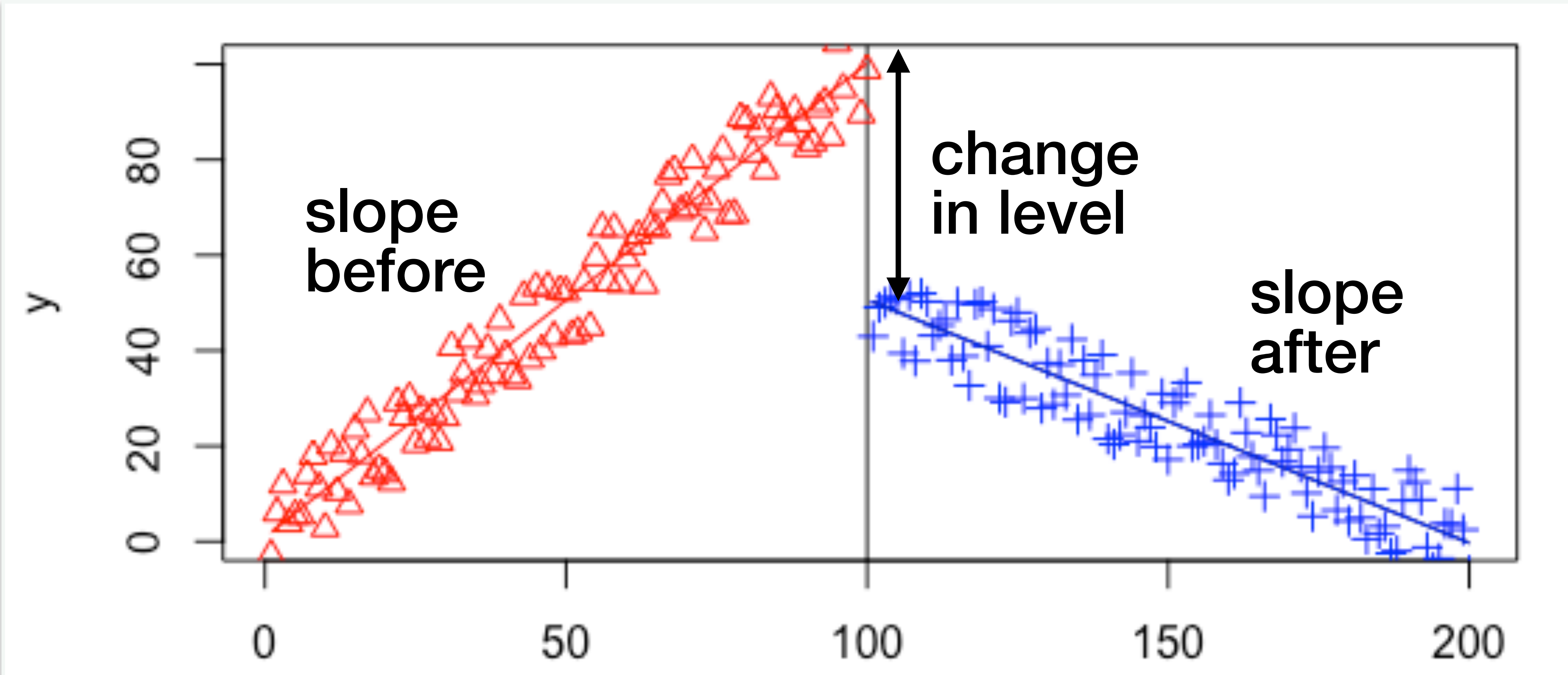
Interrupted time series



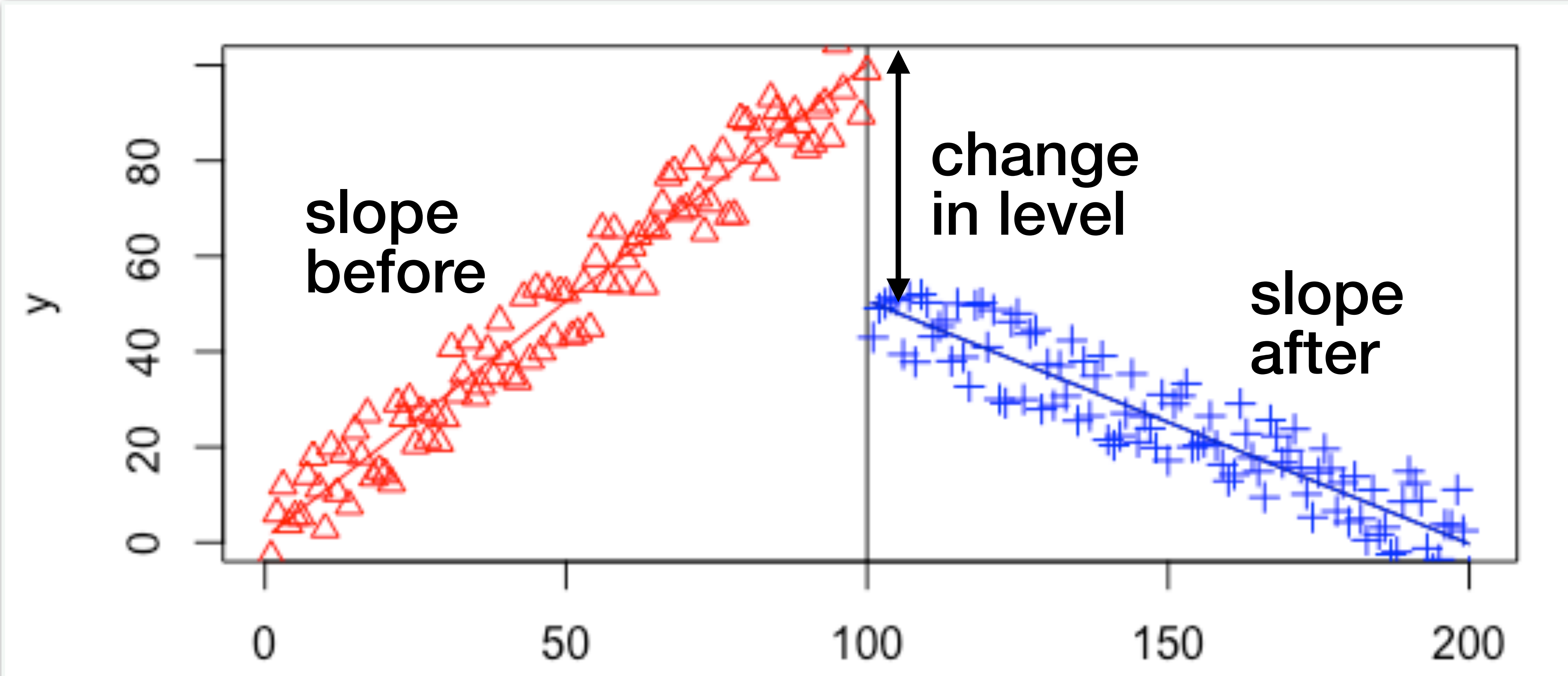


Interrupted Time Series Design

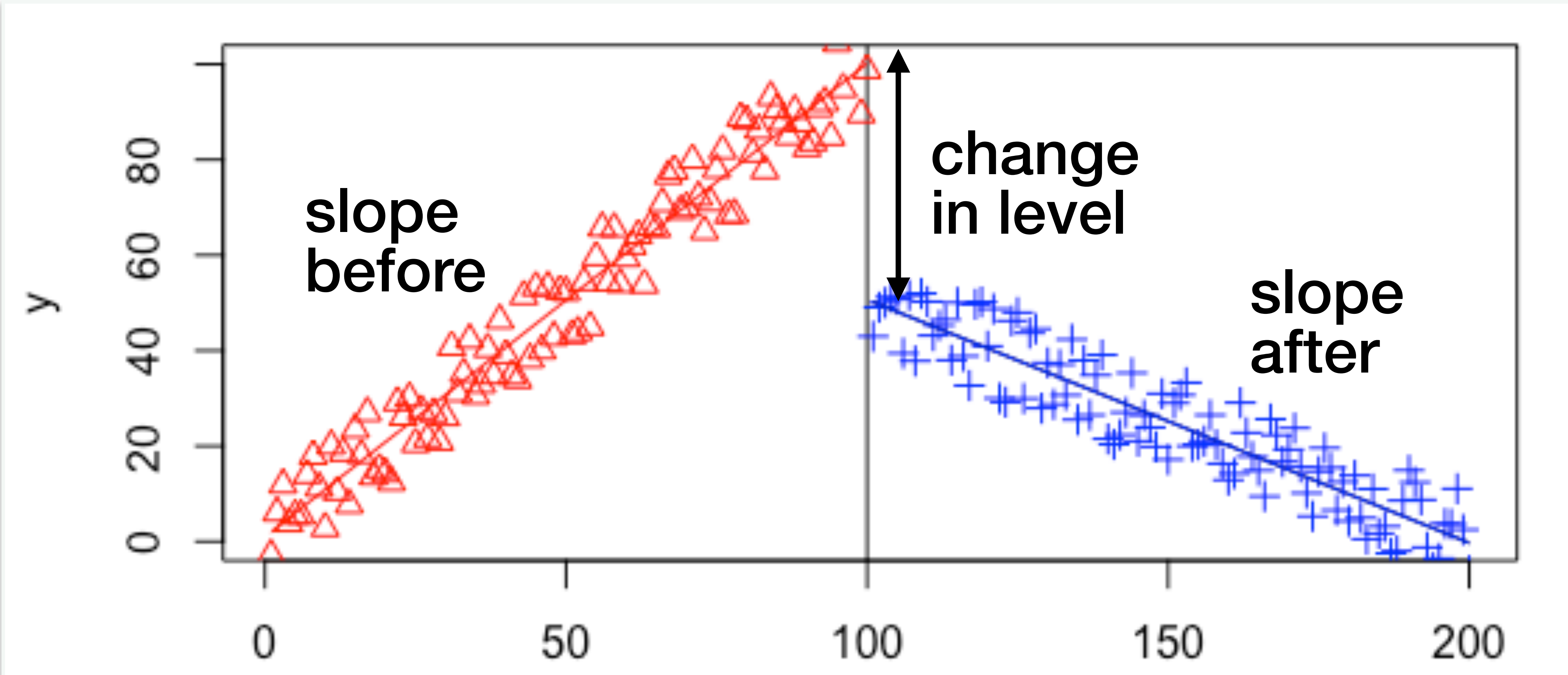
- The strongest quasi-experimental design to evaluate longitudinal effects of time-delimited interventions.
- How much did an intervention change an outcome of interest?
  - immediately and over time;
  - instantly or with delay;
  - transiently or long-term;
- Could factors other than the intervention explain the change?



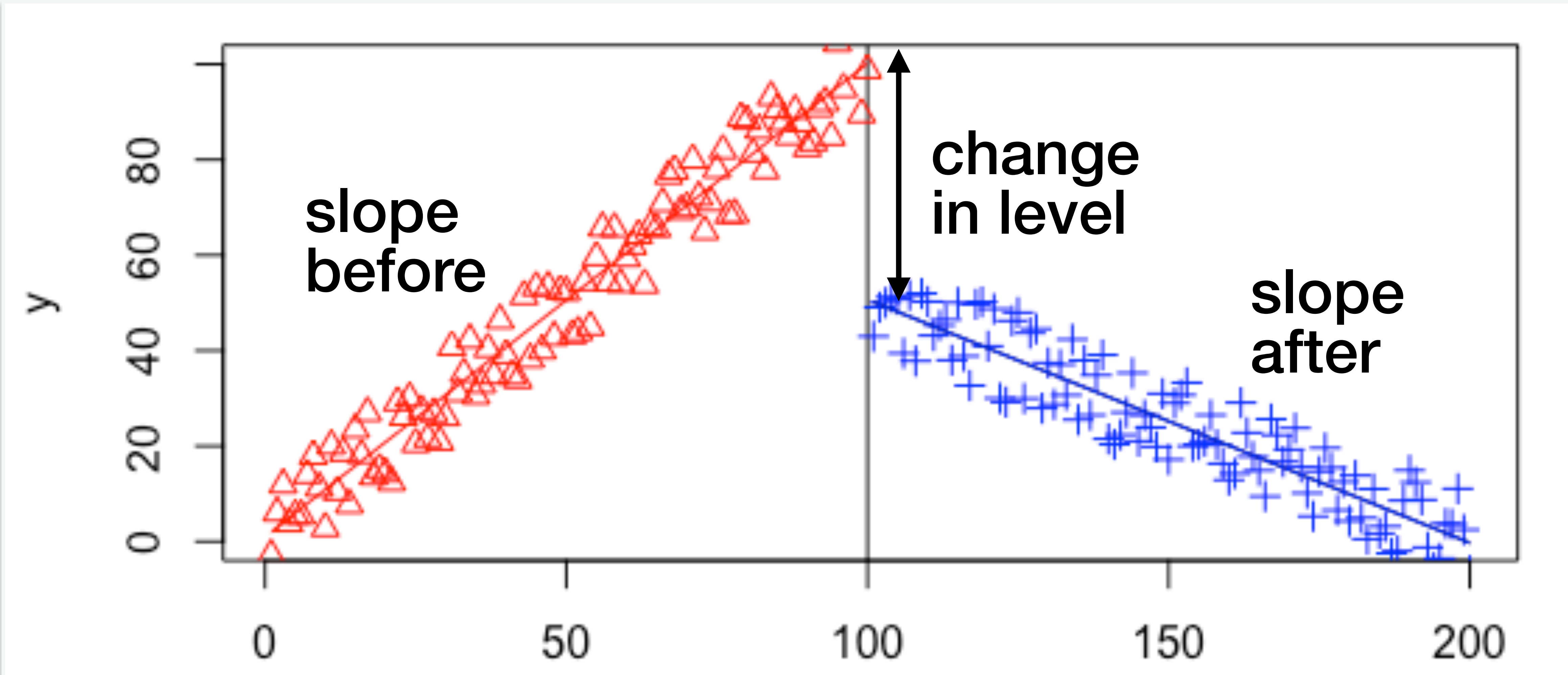
slope before
change in level
slope after
y
0
20
40
60
80
0
50
100
150
200



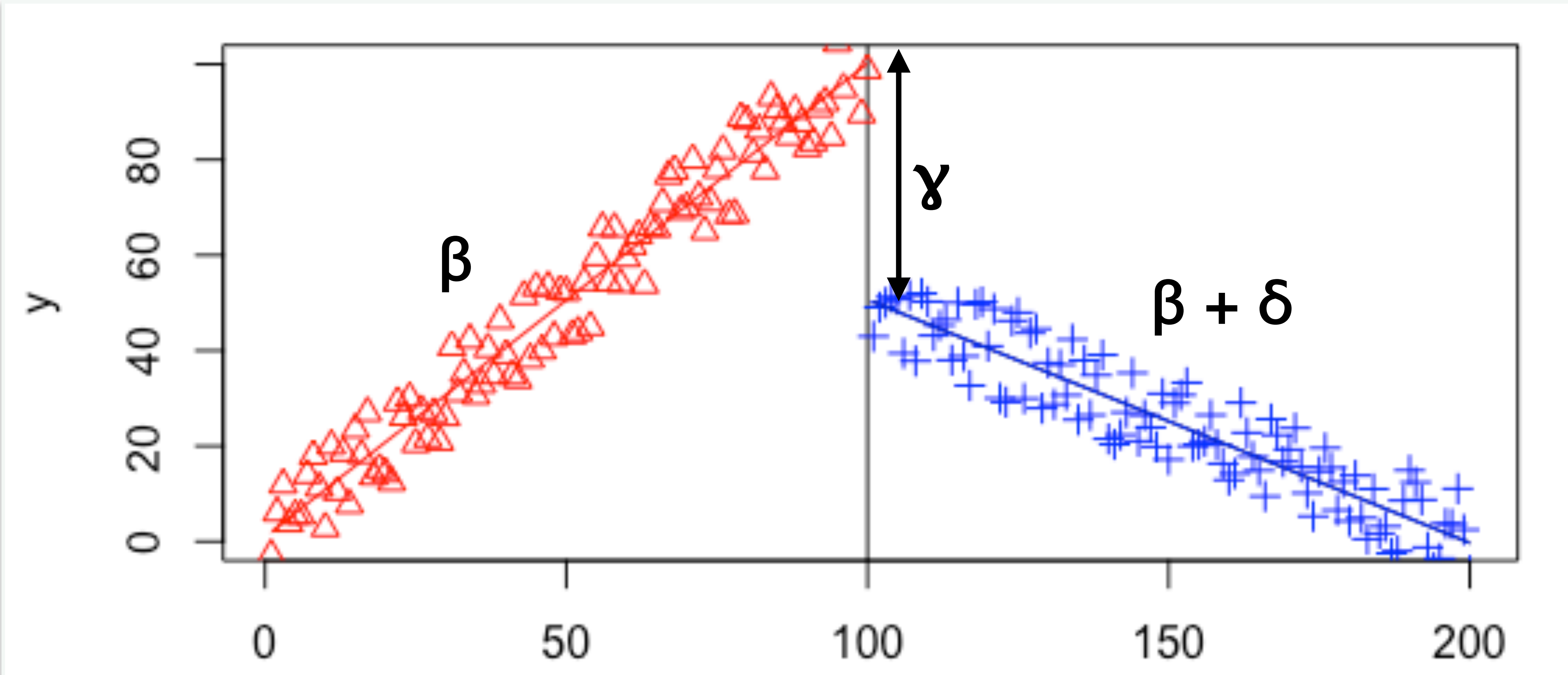
time: 1 2 3 … … … 100 101 102 … … … 200



time:	1	2	3	… … …	100	101	102 … … …	200
time after intervention:	0	0	0	… … …	1	2	3 … … …	100



time:	1	2	3	………	100	101	102	………	200
time after intervention:	0	0	0	………	1	2	3	………	100
intervention:	F	F	F	………	T	T	T	………	T



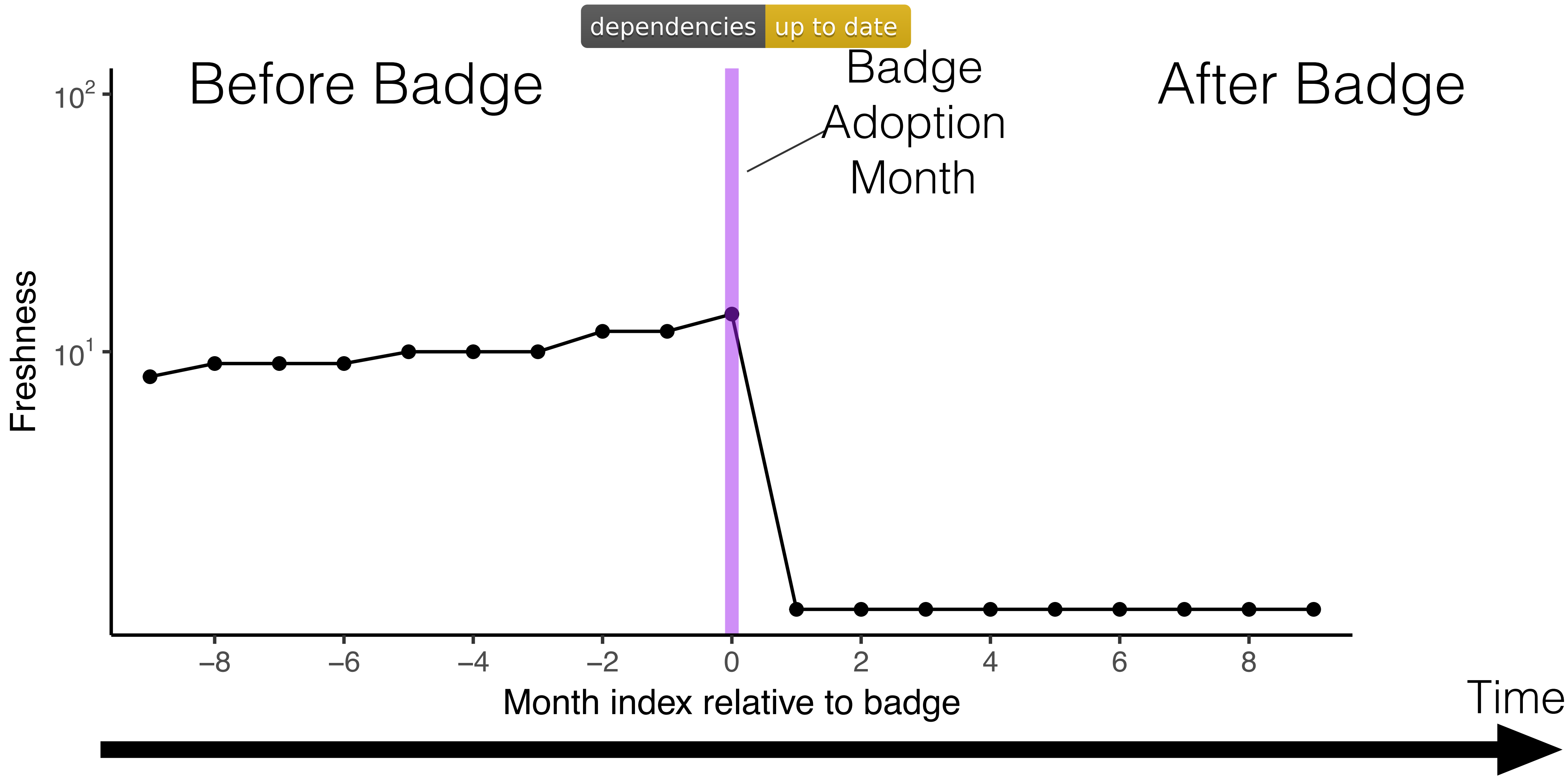
time:	1	2	3	… … …	100	101	102	… … …	200
time after intervention:	0	0	0	… … …	1	2	3	… … …	100
intervention:	F	F	F	… … …	T	T	T	… … …	T

$$
\begin{aligned}
y_i = \alpha + \beta \cdot \mathrm{time}_i \; + & \\
& \gamma \cdot \mathrm{intervention}_i \; + \\
& \delta \cdot \mathrm{time\_after\_intervention}_i + \varepsilon_i
\end{aligned}
$$

R time

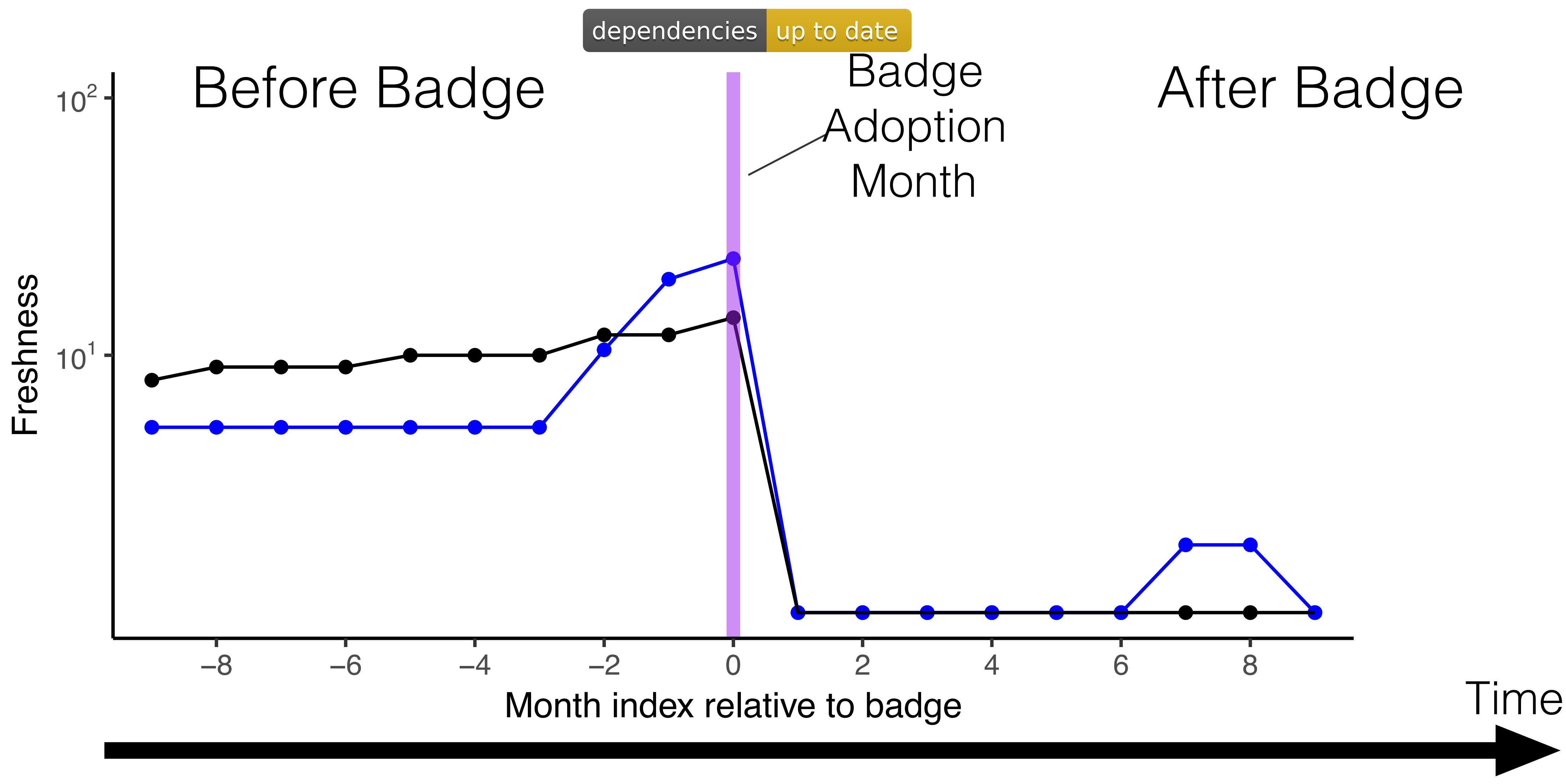
- Data: http://bit.ly/vasilescu-midwest

R time



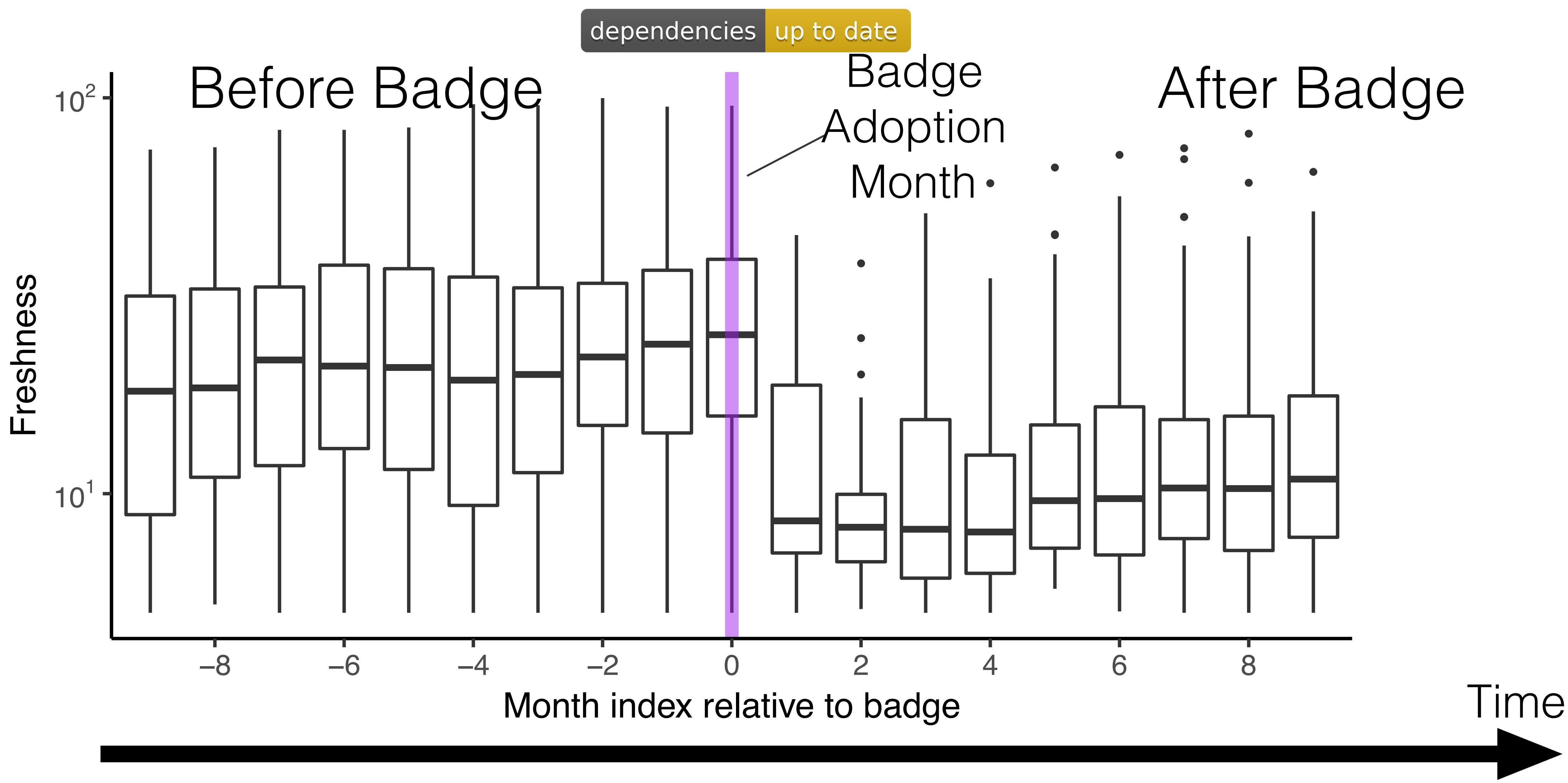
dependencies
up to date
Before Badge
Badge
Adoption
Month
After Badge
10^2
10^1
Freshness
−8
−6
−4
−2
0
2
4
6
8
Month index relative to badge
Time

R time

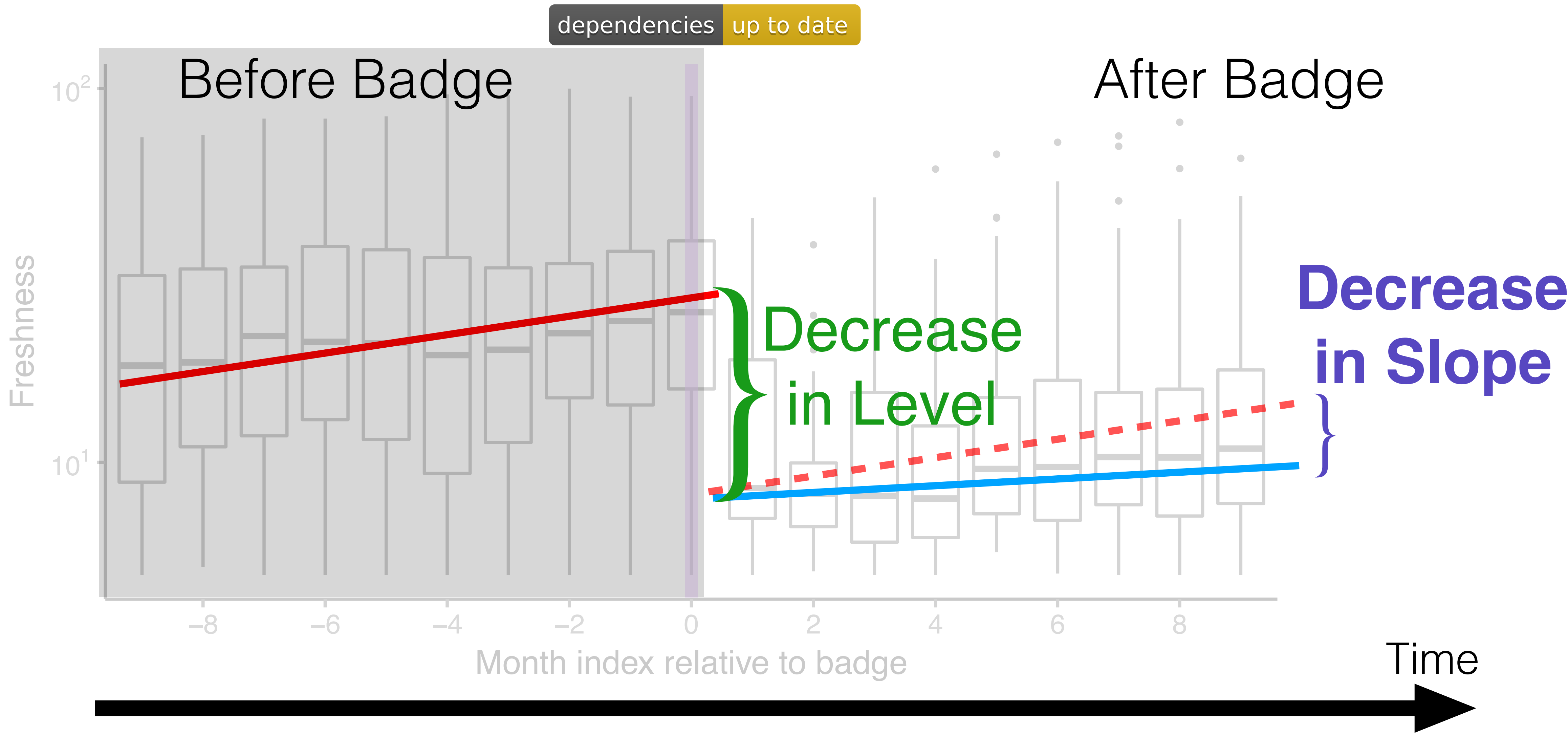


dependencies
up to date
Before Badge
Badge Adoption Month
After Badge
10^2
10^1
Freshness
–8
–6
–4
–2
0
2
4
6
8
Month index relative to badge
Time

R time


dependencies up to date
Before Badge
Badge Adoption Month
After Badge
Freshness
10^2
10^1
−8
−6
−4
−2
0
2
4
6
8
Month index relative to badge
Time

R time


dependencies up to date
Before Badge
After Badge
10^2
10^1
Freshness
Decrease in Level
Decrease in Slope
−8
−6
−4
−2
0
2
4
6
8
Month index relative to badge
Time

Data Science for Software Engineering
Part 2

Bogdan Vasilescu

Today

- First session:
  - Intro: the **Science** of Software Engineering
  - Hands-on: segmented regression analysis of interrupted time series data

- Second session:
  - Intro: the Naturalness of Software theory
  - Hands-on: language modeling

Slides thanks to:

- Prem Devanbu, UC Davis

Natural languages are complex

Natural languages are complex

..but Natural Utterances are simple & repetitive

English, தமிழ், German

English, தமிழ், German

Can be Rich, Powerful, Expressive



English, தமிழ், German

Can be Rich, Powerful, Expressive

..but “in nature” is mostly Simple, Repetitive, Boring

English, தமிழ், German

Can be Rich, Powerful, Expressive

..but "in nature" is mostly Simple, Repetitive, Boring

Statistical Models

The “naturalness of software” thesis

Programming Languages are complex...

...but Natural Programs are simple & repetitive.

and this, too, CAN BE EXPLOITED!!

[Hindle et al, 2011]

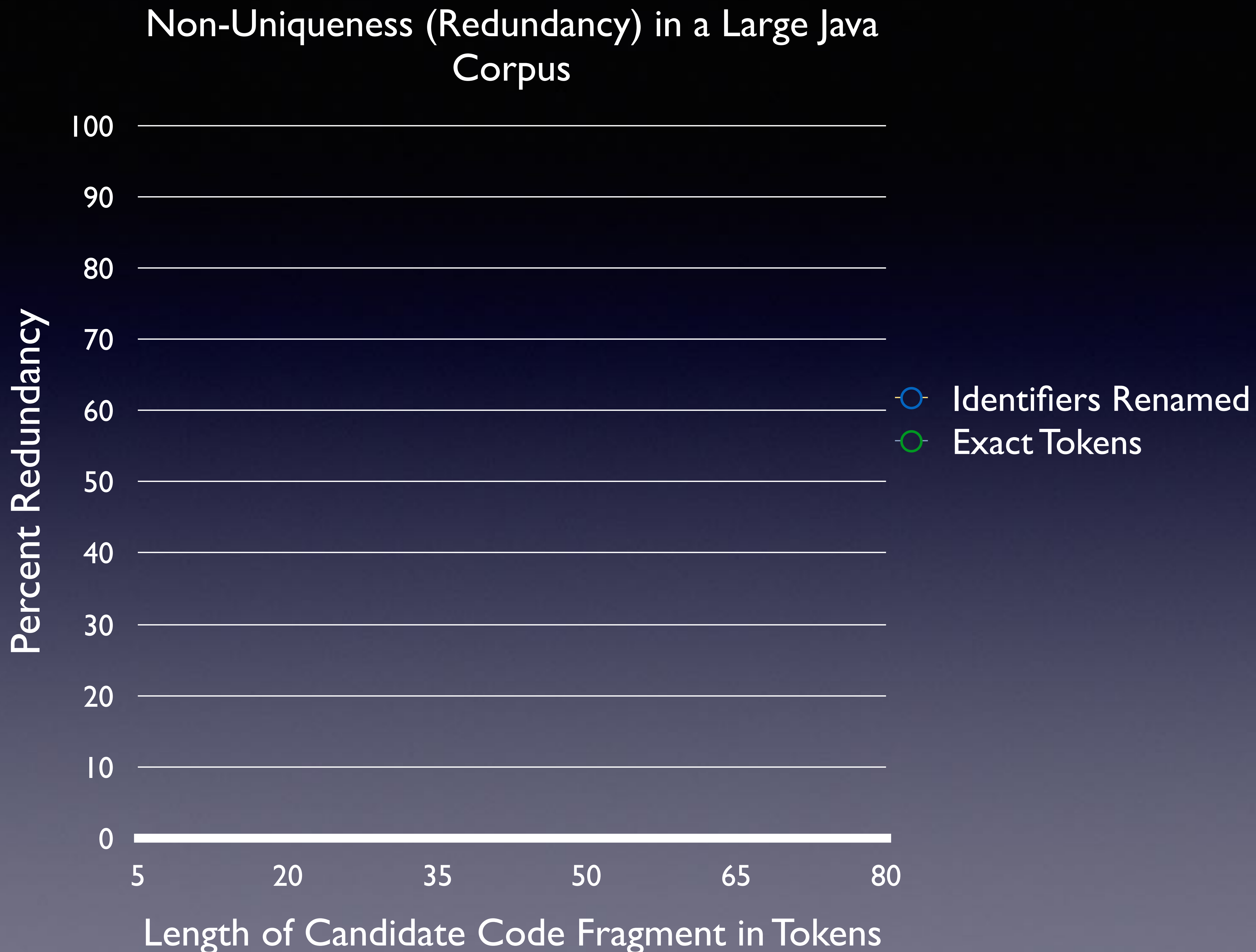
Repetition

↓↓↓

Statistical Models

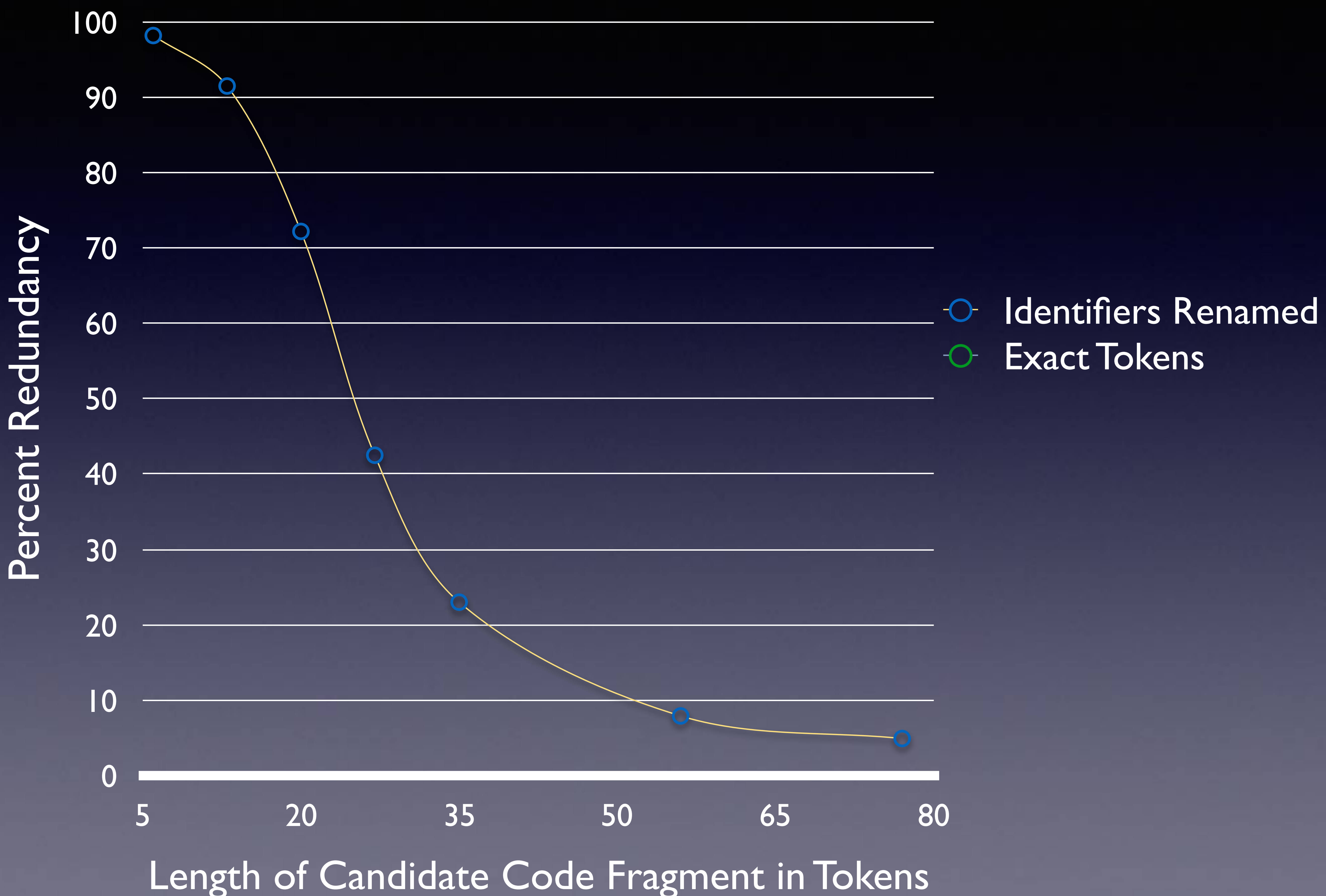
↓↓↓

Make “Search”
Algorithms faster.

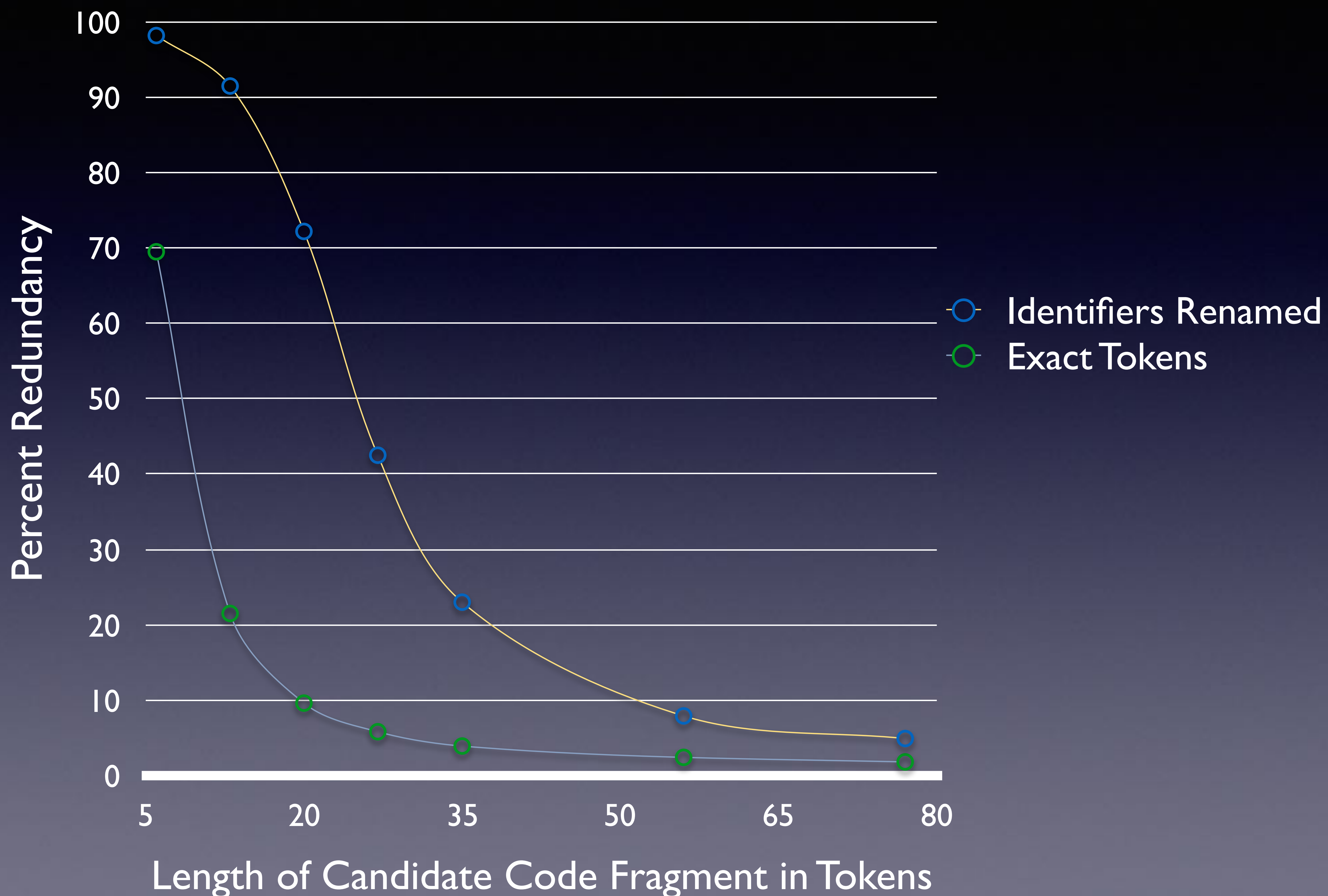


Non-Uniqueness (Redundancy) in a Large Java Corpus
Percent Redundancy
100
90
80
70
60
50
40
30
20
10
0
5
20
35
50
65
80
Length of Candidate Code Fragment in Tokens
Identifiers Renamed
Exact Tokens

Non-Uniqueness (Redundancy) in a Large Java Corpus



Non-Uniqueness (Redundancy) in a Large Java Corpus



Software ***is*** really repetitive.

how can we use this?

How has “naturalness” (repetitive structure) of Natural Language been exploited?

Large Corpora

↓↓↓

Language Models

↓↓↓

Speech Recognition,
Translation, etc.

Language Models

Language Models

For any utterance U, $0 \leq p(U) \leq 1$

If Ua is more often uttered than Ub $p(U_a) > p(U_b)$

Language Models

For any utterance U, $0 \leq p(U) \leq 1$

If Ua is more often uttered than Ub $p(U_a) > p(U_b)$

$$p(\text{“}EuropeanCentralFish\text{”}) < p(\text{“}EuropeanCentralBank\text{”})$$

Language Models

For any utterance U, $0 \leq p(U) \leq 1$

If Ua is more often uttered than Ub $p(U_a) > p(U_b)$

$$p(\text{``}EuropeanCentralFish\text{''}) < p(\text{``}EuropeanCentralBank\text{''})$$

$$p(\texttt{for}(\texttt{i} = 0; \texttt{i} < 10; \texttt{fish} + +)) < p(\texttt{for}(\texttt{i} = 0; \texttt{i} < 10; \texttt{i} + +))$$

Exploiting Code Language Models

Exploiting Code Language Models

Suggest next tokens for developers

Exploiting Code Language Models

Suggest next tokens for developers

Complete next tokens for developers

Exploiting Code Language Models

Suggest next tokens for developers

Complete next tokens for developers

Assistive (speech, gesture) coding for convenience and disability.

Exploiting Code Language Models

Suggest next tokens for developers

Complete next tokens for developers

Assistive (speech, gesture) coding for convenience and disability.

Statistical translation approach to summarization & retrieval

Exploiting Code Language Models

Suggest next tokens for developers

Complete next tokens for developers

Assistive (speech, gesture) coding for convenience and disability.

Statistical translation approach to summarization & retrieval

fast, “good guess” static analysis.

Exploiting Code Language Models

Suggest next tokens for developers

Complete next tokens for developers

Assistive (speech, gesture) coding for convenience and disability.

Statistical translation approach to summarization & retrieval

fast, “good guess” static analysis.

Search-based Software Engineering.

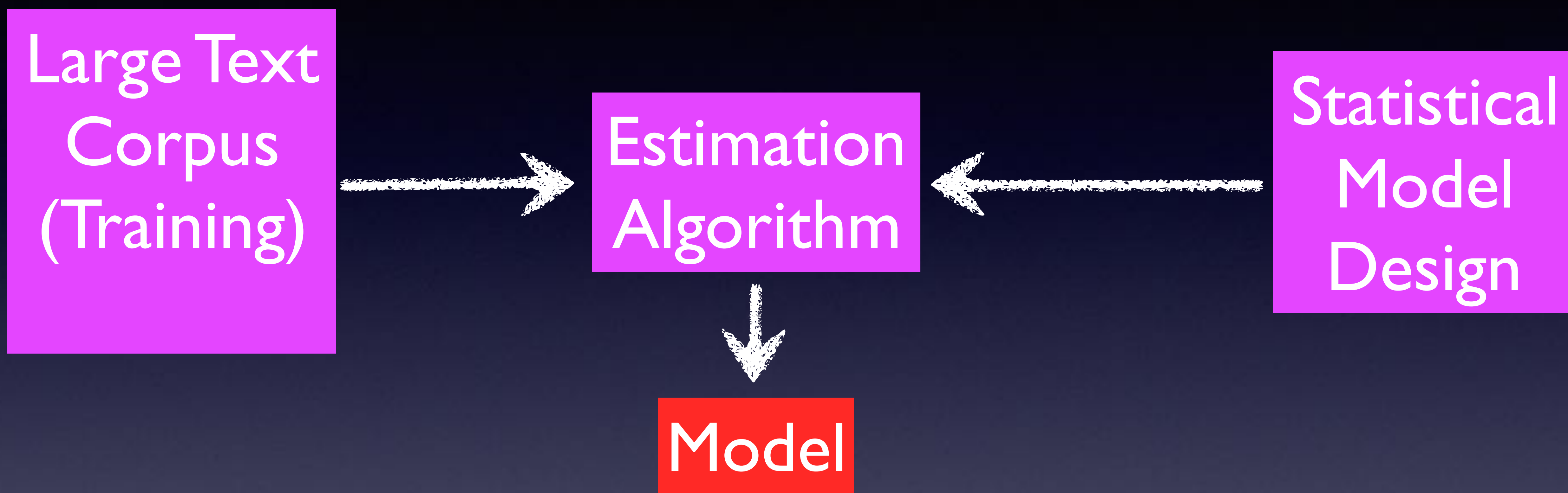
How to build an LM.

How to build an LM.

Large Text Corpus (Training)

Statistical Model Design

How to build an LM.



How to build an LM.

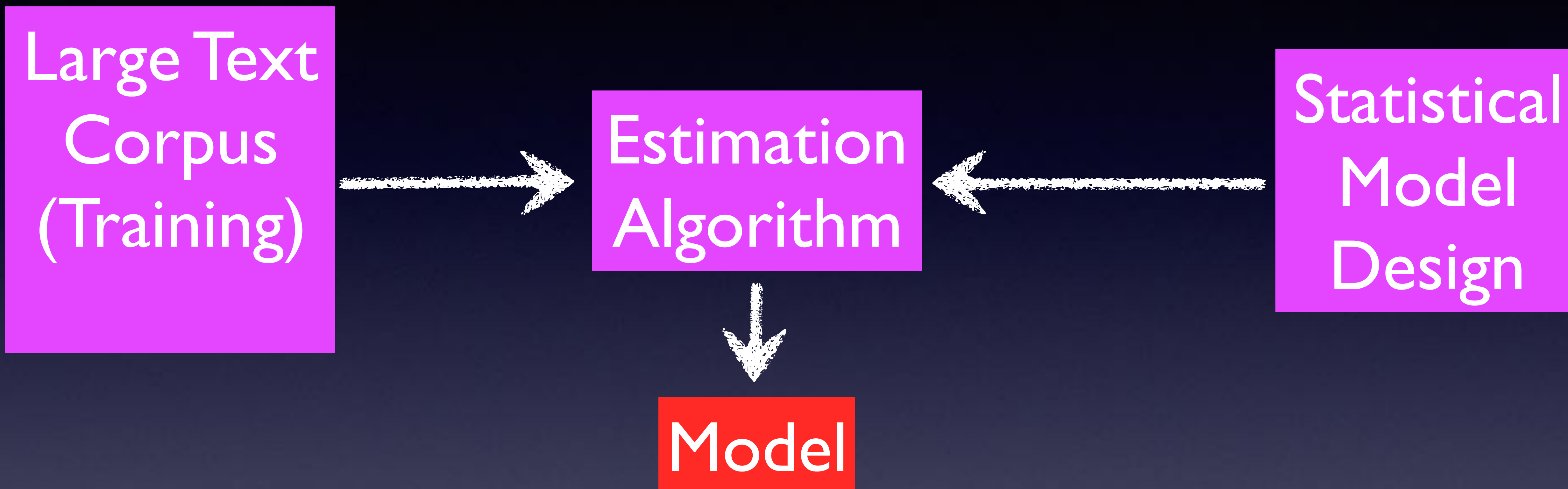
Large Text Corpus (Training)

Estimation Algorithm

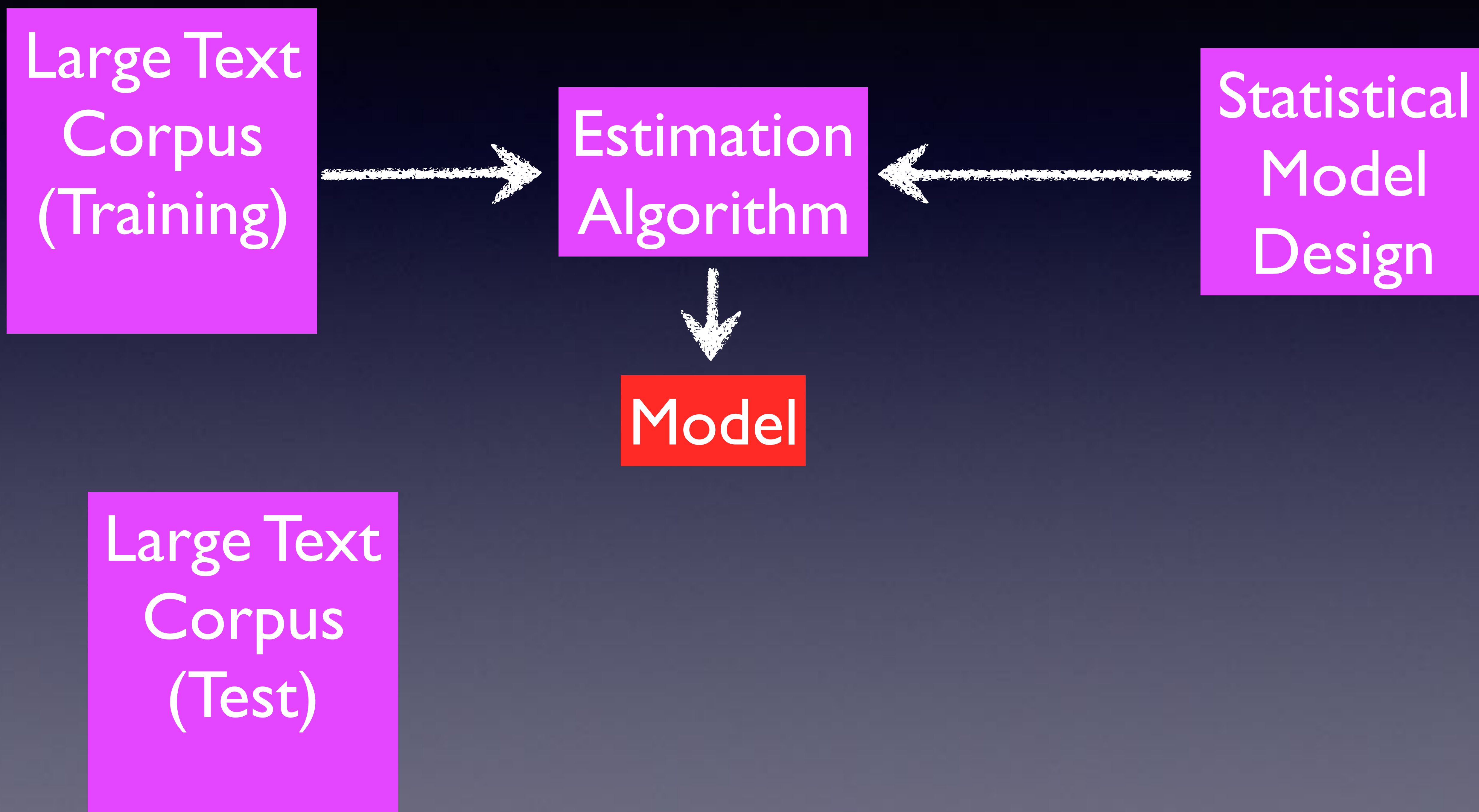
Statistical Model

Estimated using frequency of occurrence!

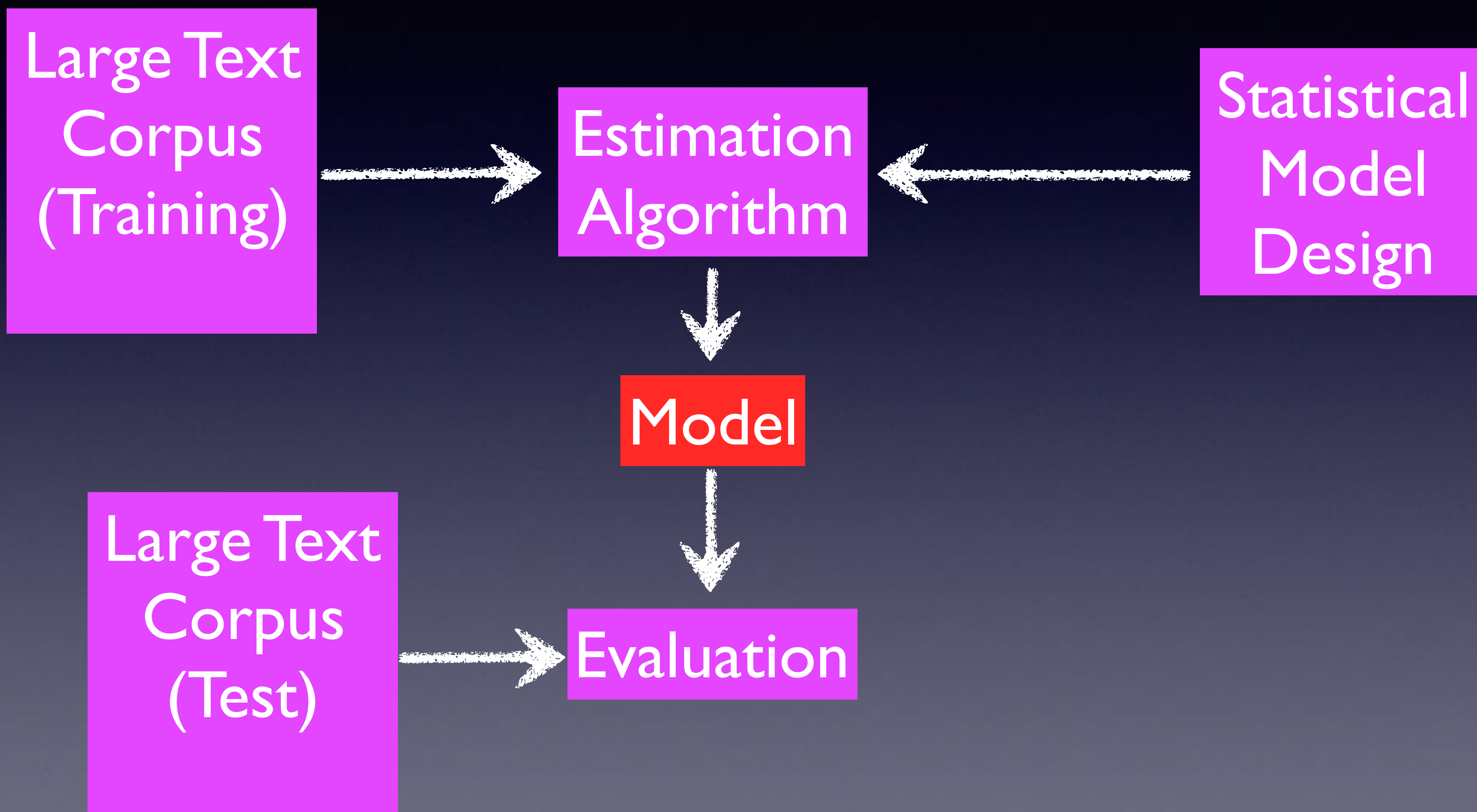
How to build an LM.



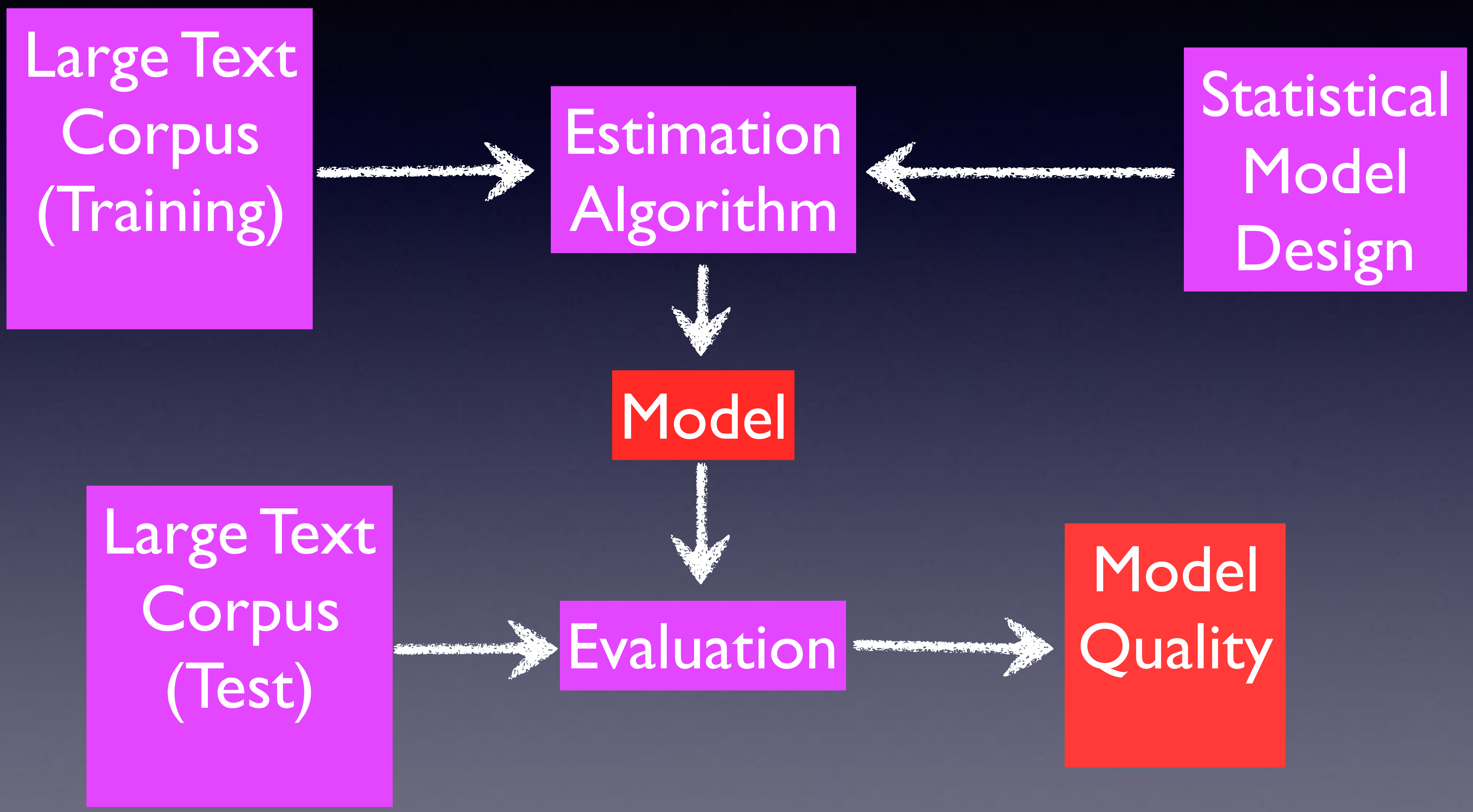
How to build an LM.



How to build an LM.



How to build an LM.



What a Language Model does

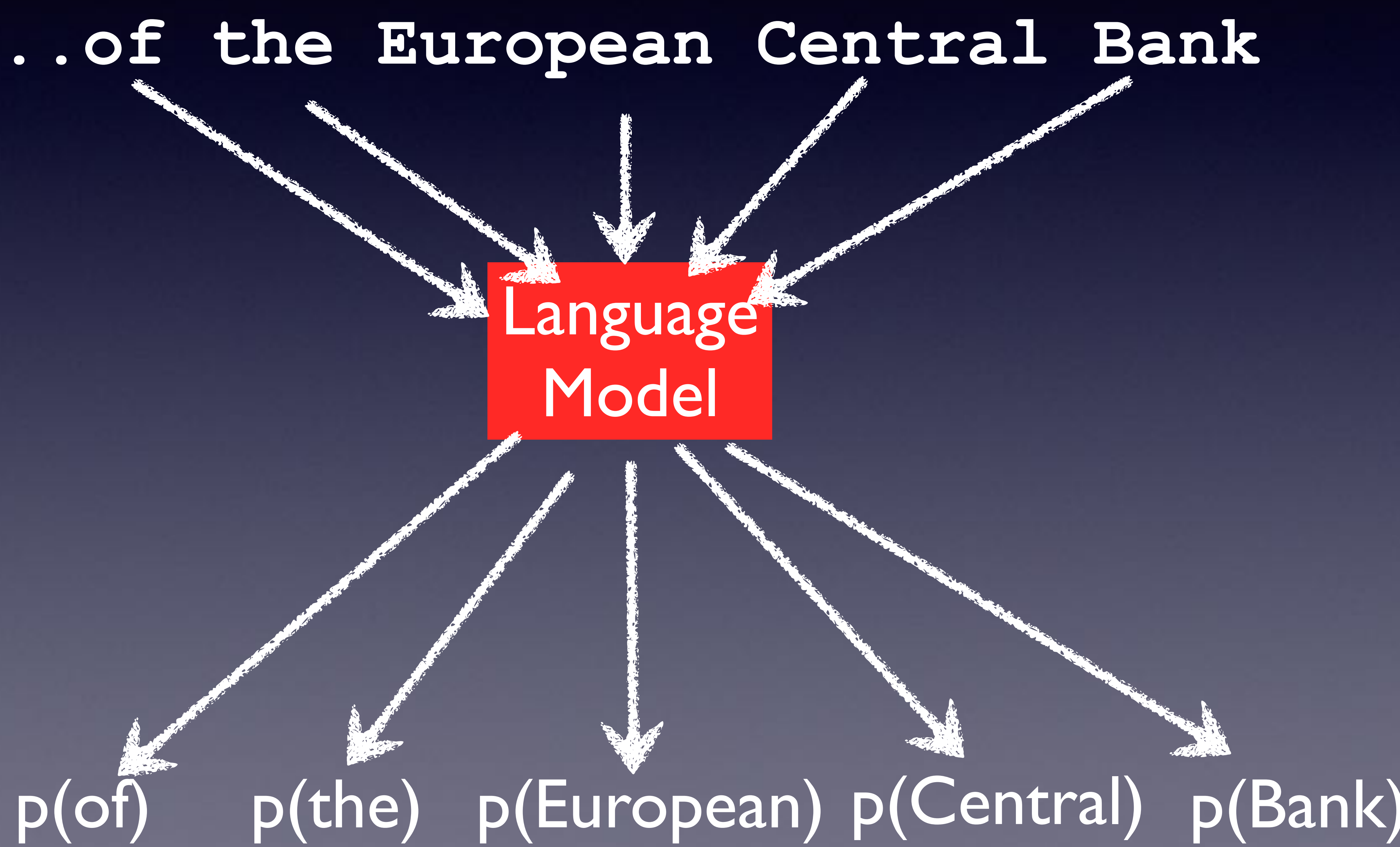
Language Model

What a Language Model does

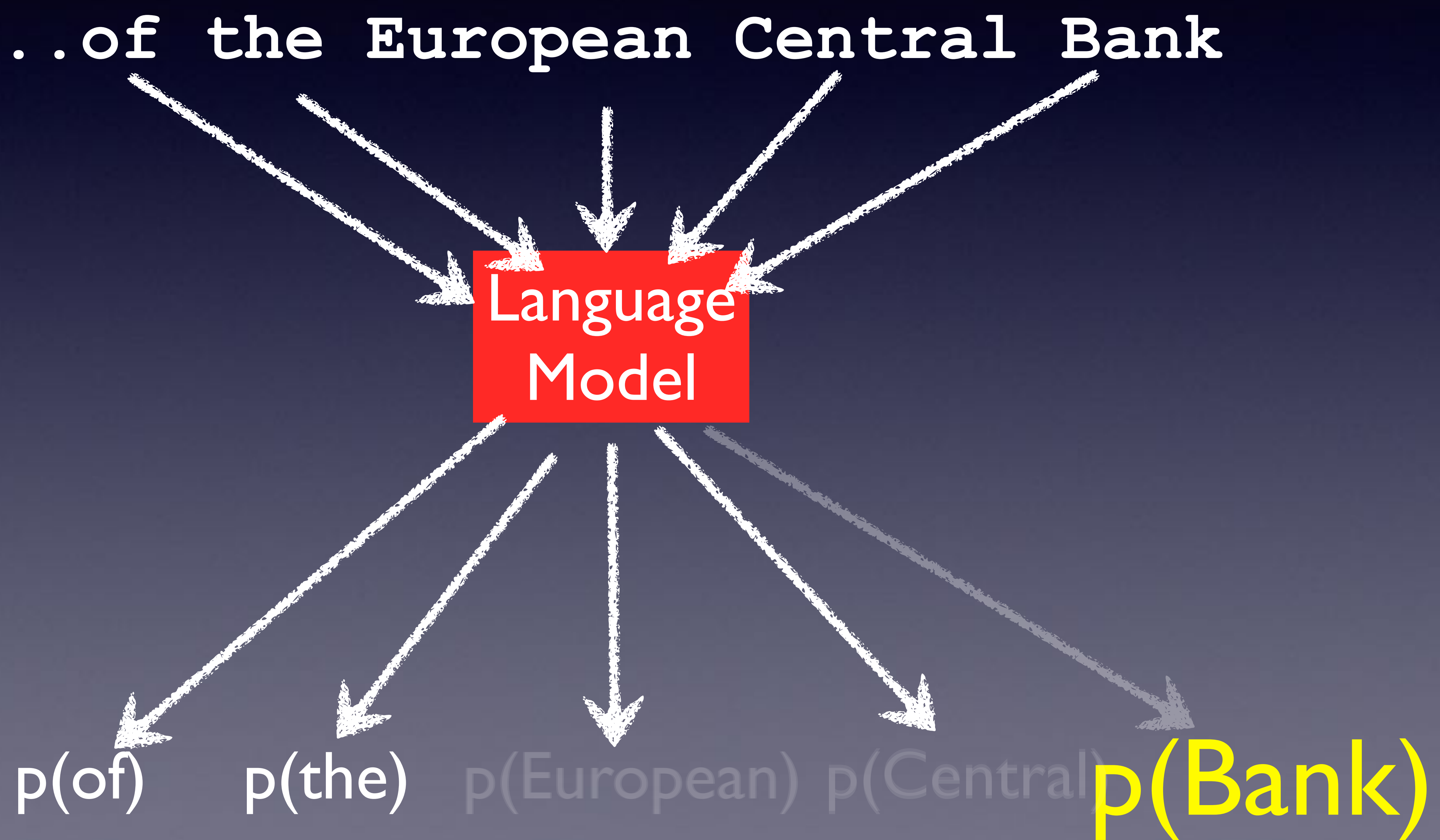
`..of the European Central Bank`

Language Model

What a Language Model does



What a Language Model does



Language Models

Vastly more complex

Language Models

Language Models

Vastly more complex

Almost always face data-sparsity

Language Models

Vastly more complex

Almost always face data-sparsity

Novel, NLP-specific estimation methods

Evaluating a LM's quality

Evaluating a LM's quality

The words it encounters are not "too surprising" to it.

Evaluating a LM's quality

The words it encounters are not "too surprising" to it.

- ☑ Frequently encountered language events are assigned higher probability

Evaluating a LM's quality

The words it encounters are not "too surprising" to it.

- ☑ Frequently encountered language events are assigned higher probability
- ☑ Infrequent language events are assigned lower probability.

Evaluating a LM's quality

The words it encounters are not "too surprising" to it.

- ☑ Frequently encountered language events are assigned higher probability
- ☑ Infrequent language events are assigned lower probability.
- ☑ *....measured using "Cross-Entropy"*

Background
Cross Entropy

Language Model

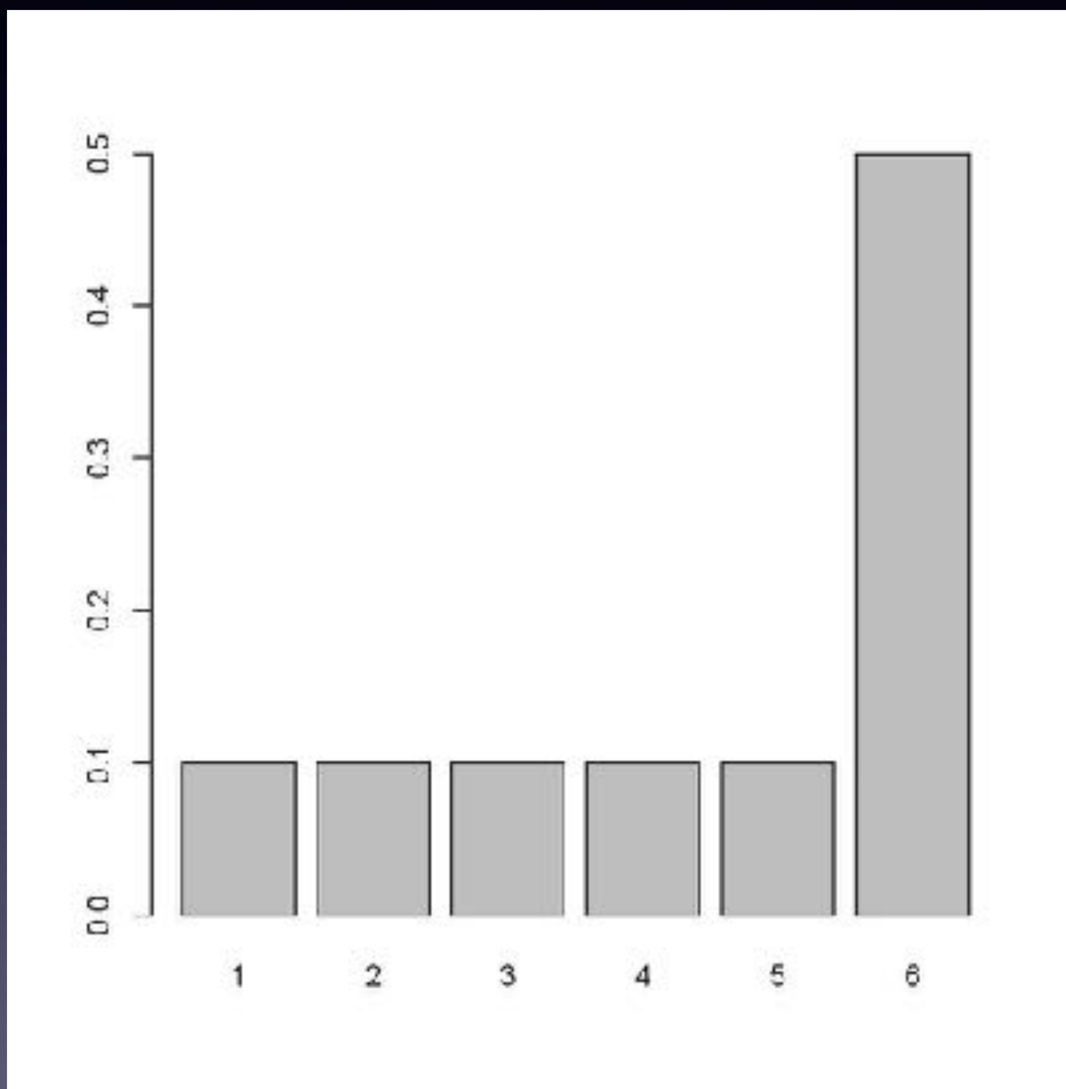
Good Description?

```
public class FunctionCall {
    public static void funct1 () {
        System.out.println ("Inside funct1");
    }
    public static void main (String[] args) {
        int val;
        System.out.println ("Inside main");
        funct1();
        System.out.println ("About to call funct2");
        val = funct2(8);
        System.out.println ("funct2 returned a value of " + val);
        System.out.println ("About to call funct2 again");
        val = funct2(-3);
        System.out.println ("funct2 returned a value of " + val);
    }
    public static int funct2 (int param) {
        System.out.println ("Inside funct2 with param " + param);
        return param * 2;
    }
}
```

Background-Entropy

$$\sum_i -p(e_i) log\ p(e_i)$$

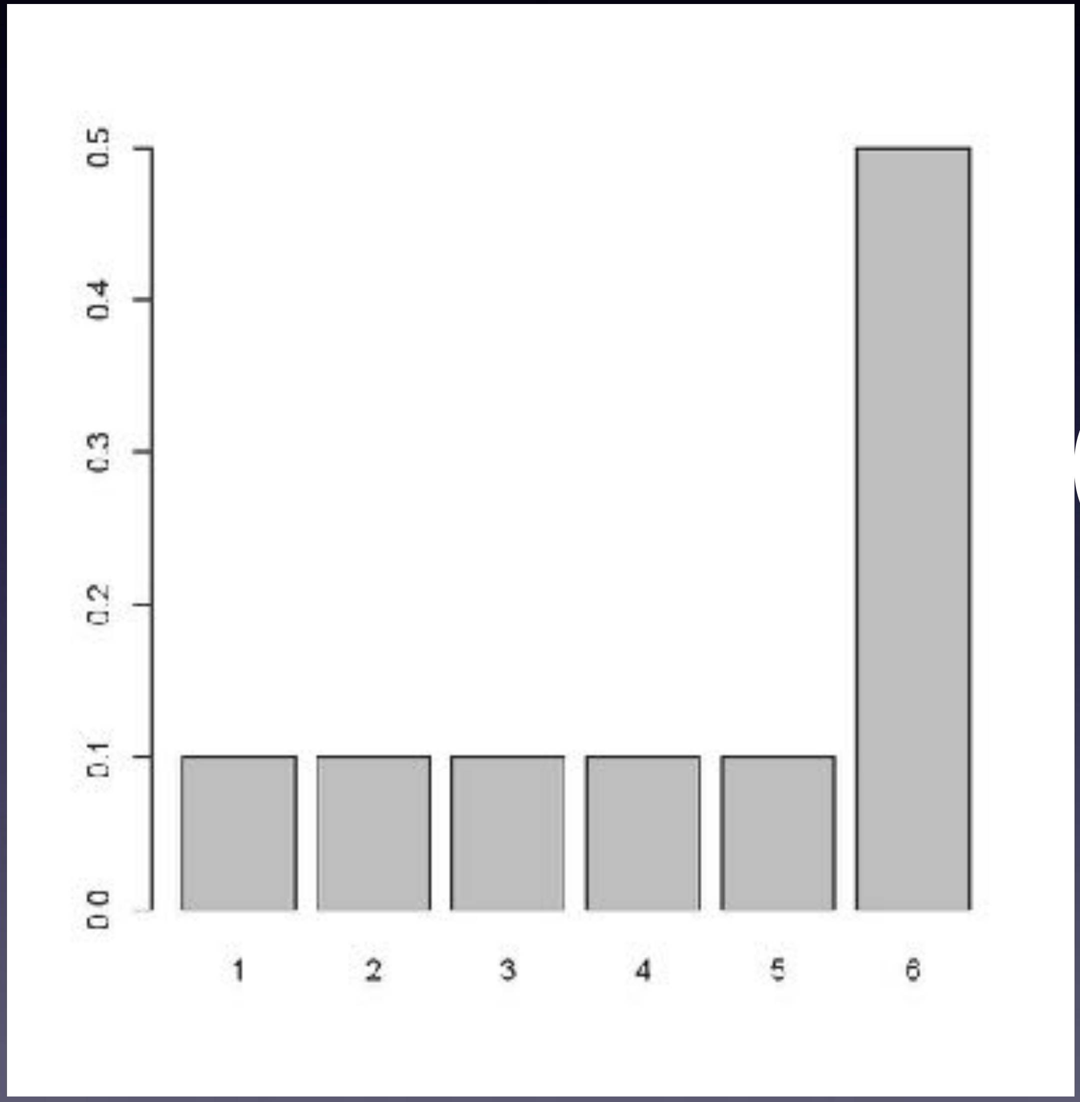
Background-Entropy



$$(e_i)log\ p(e_i)$$

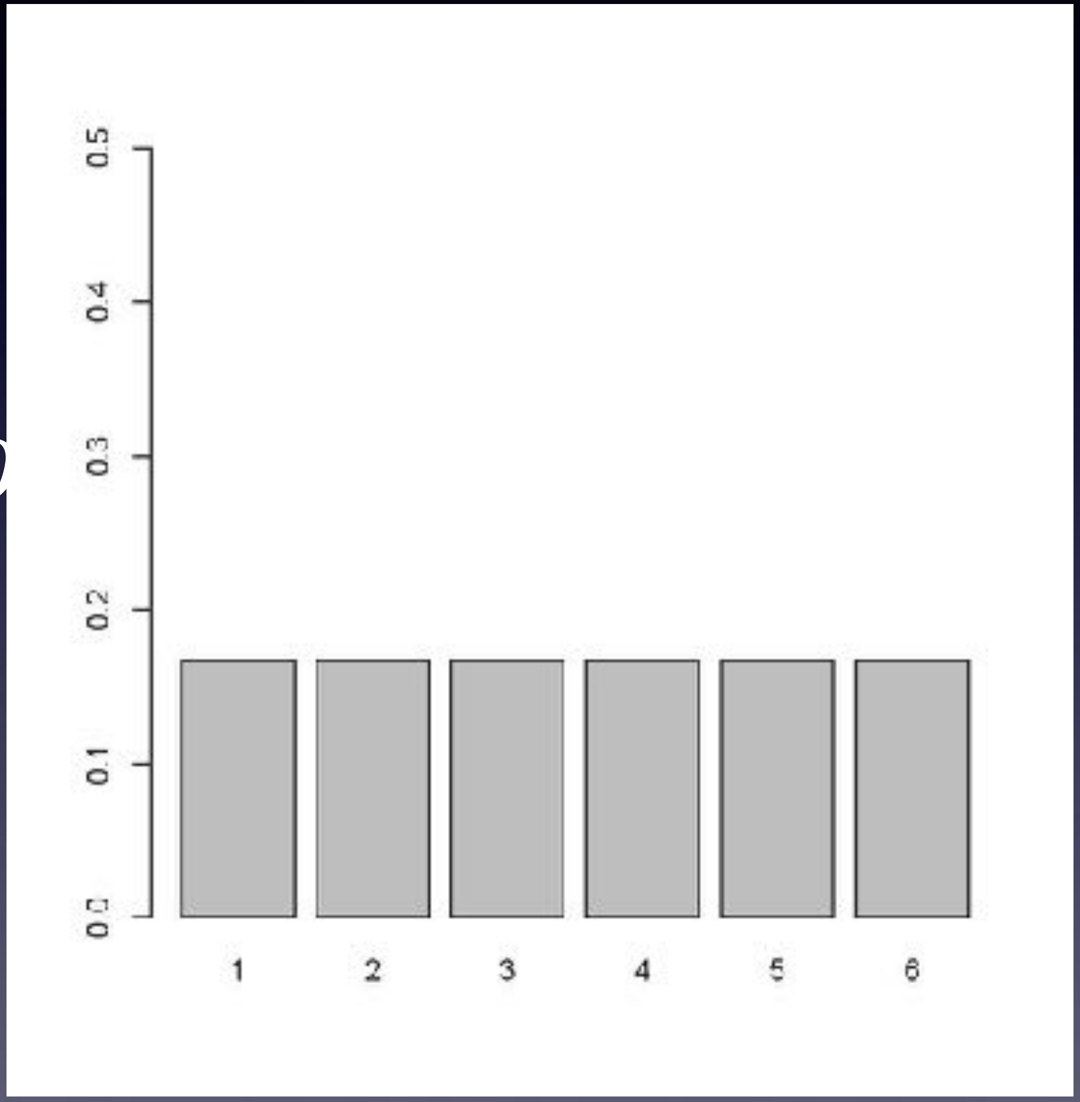
Low Entropy

Background-Entropy



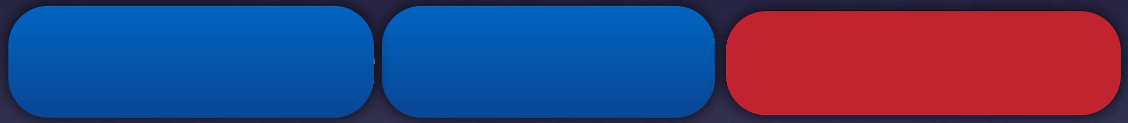
Low Entropy

$(e_i)lo$



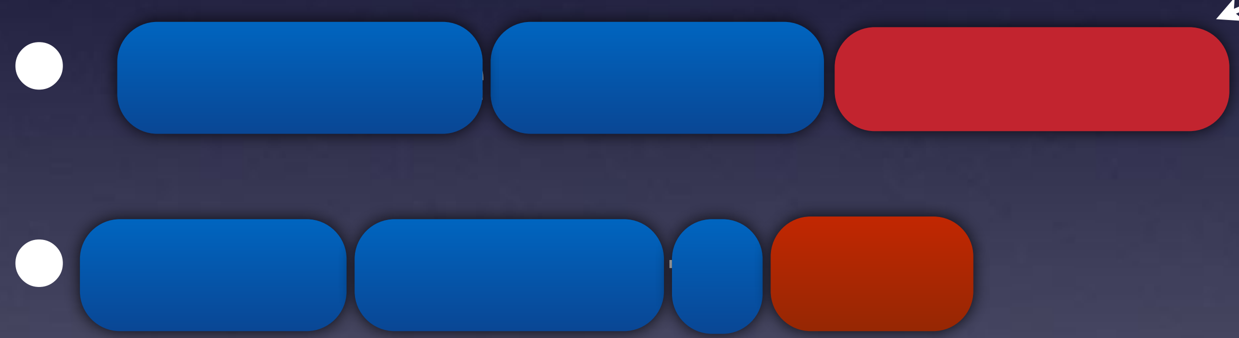
High Entropy

n-gram models

- Intuition: Local Context Helps.
- Examples (NL, then code)
    - 
    - 
-

n-gram models

- Intuition: Local Context Helps.
- Examples (NL, then code)



What is This?

n-gram models

- Intuition: Local Context Helps.
- Examples (NL, then code)
  - choice
  -
-

What is This?

n-gram models

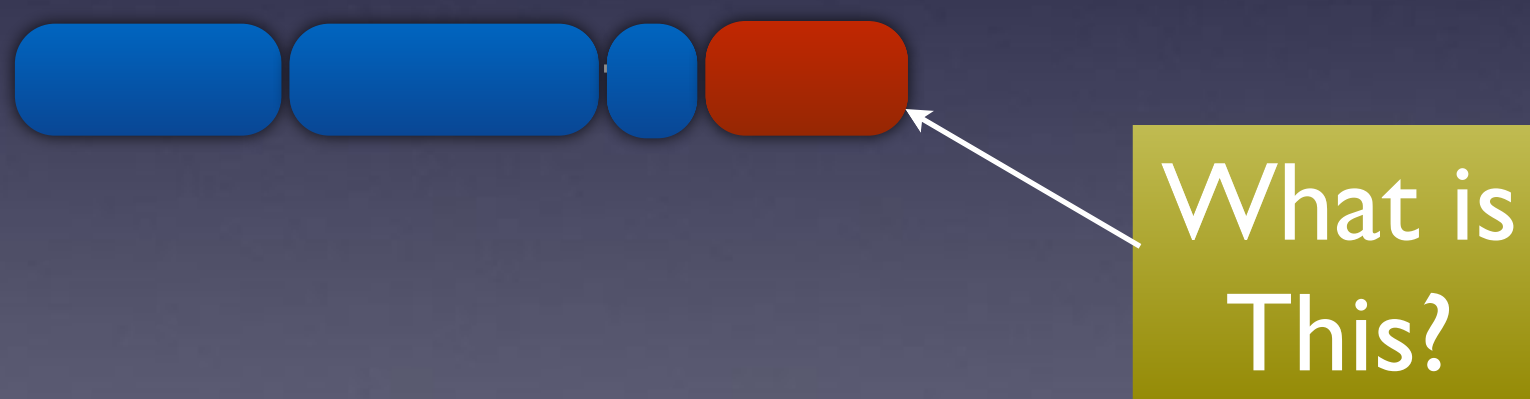
- Intuition: Local Context Helps.
- Examples (NL, then code)
  - multiple choice
  -
-

What is This?

n-gram models

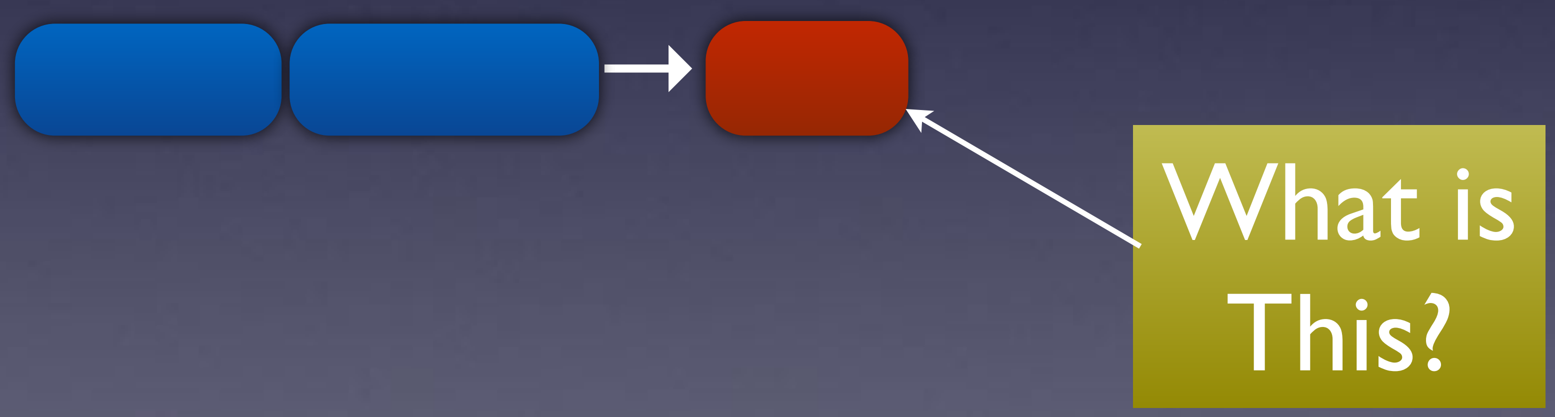
- Intuition: Local Context Helps.
- Examples (NL, then code)
  - multiple choice question
  - 
-

n-gram models

- Intuition: Local Context Helps.
- Examples (NL, then code)
  - multiple choice question
  - 

-

n-gram models

- Intuition: Local Context Helps.
- Examples (NL, then code)
    - multiple choice question
    - 

-

n-gram models

- Intuition: Local Context Helps.
- Examples (NL, then code)
  - multiple choice question
  - = item →

What is This?

-

n-gram models

- Intuition: Local Context Helps.
- Examples (NL, then code)
    - multiple choice question
    - item = item→

What is This?

-

n-gram models

- Intuition: Local Context Helps.
- Examples (NL, then code)
  - multiple choice question
  - item = item→next
-

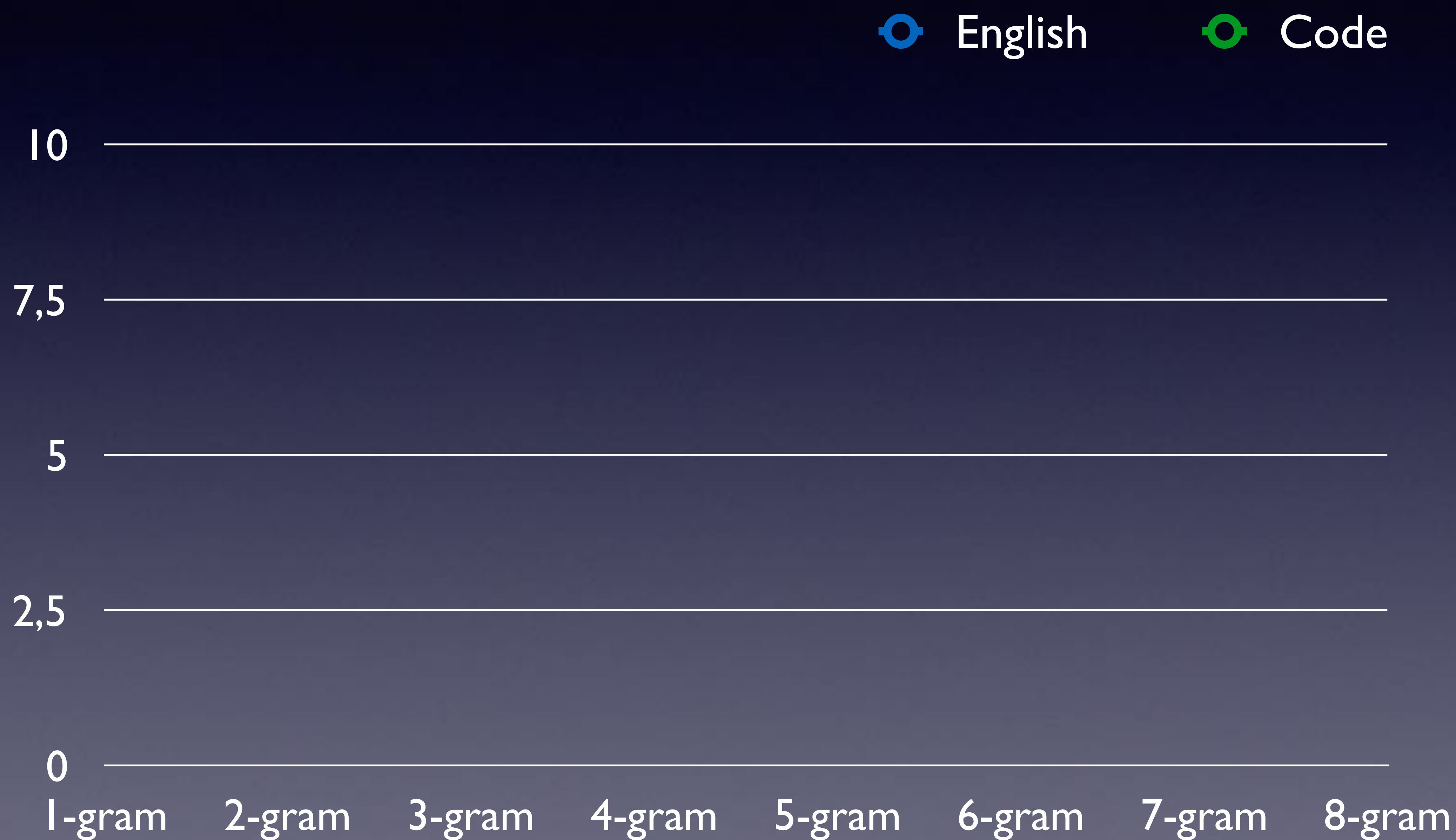
n-gram models

- Intuition: Local Context Helps.
- Examples (NL, then code)
  - multiple choice question
  - item = item→next

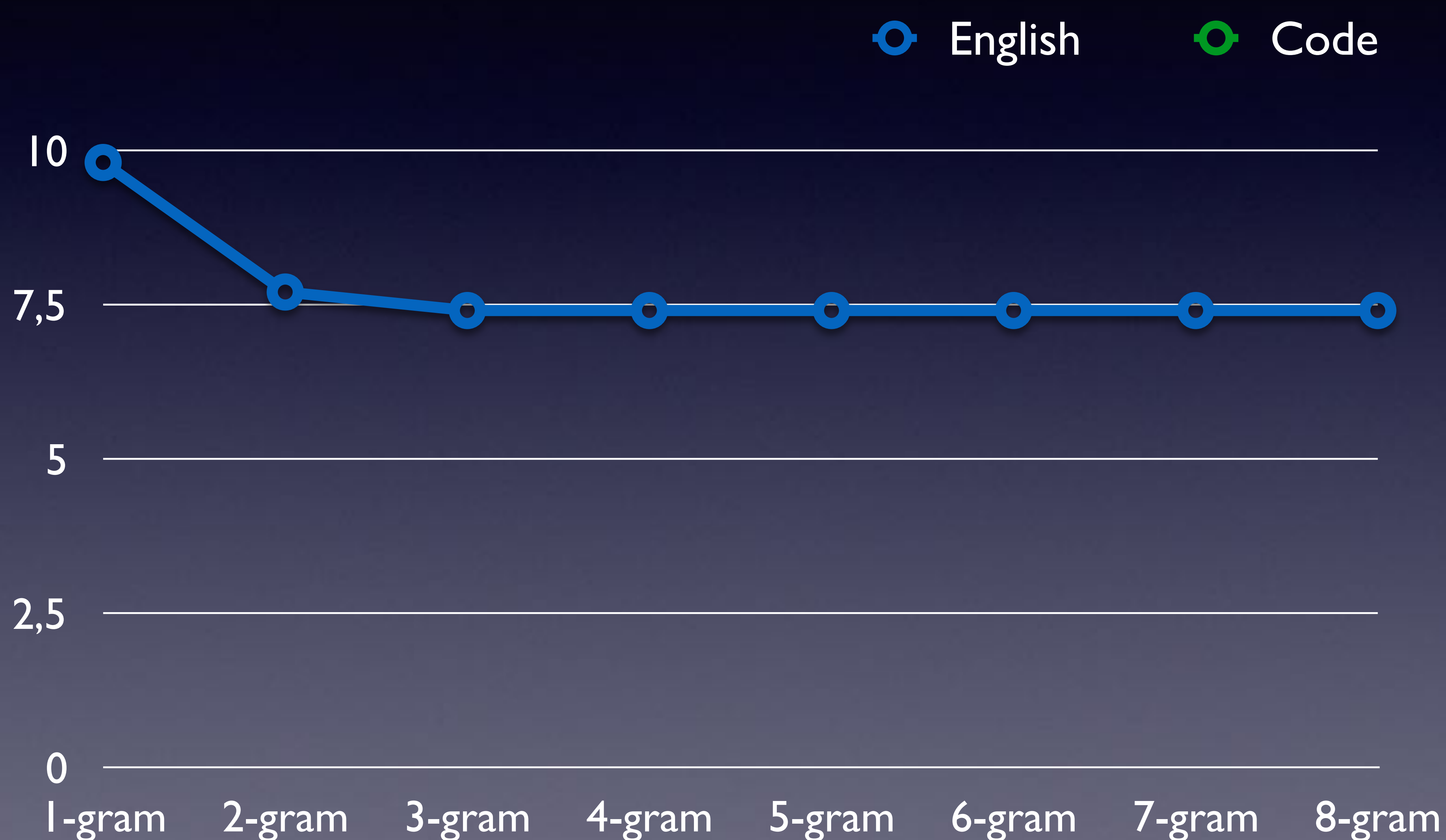
- More context helps more!!

N-gram Cross Entropy

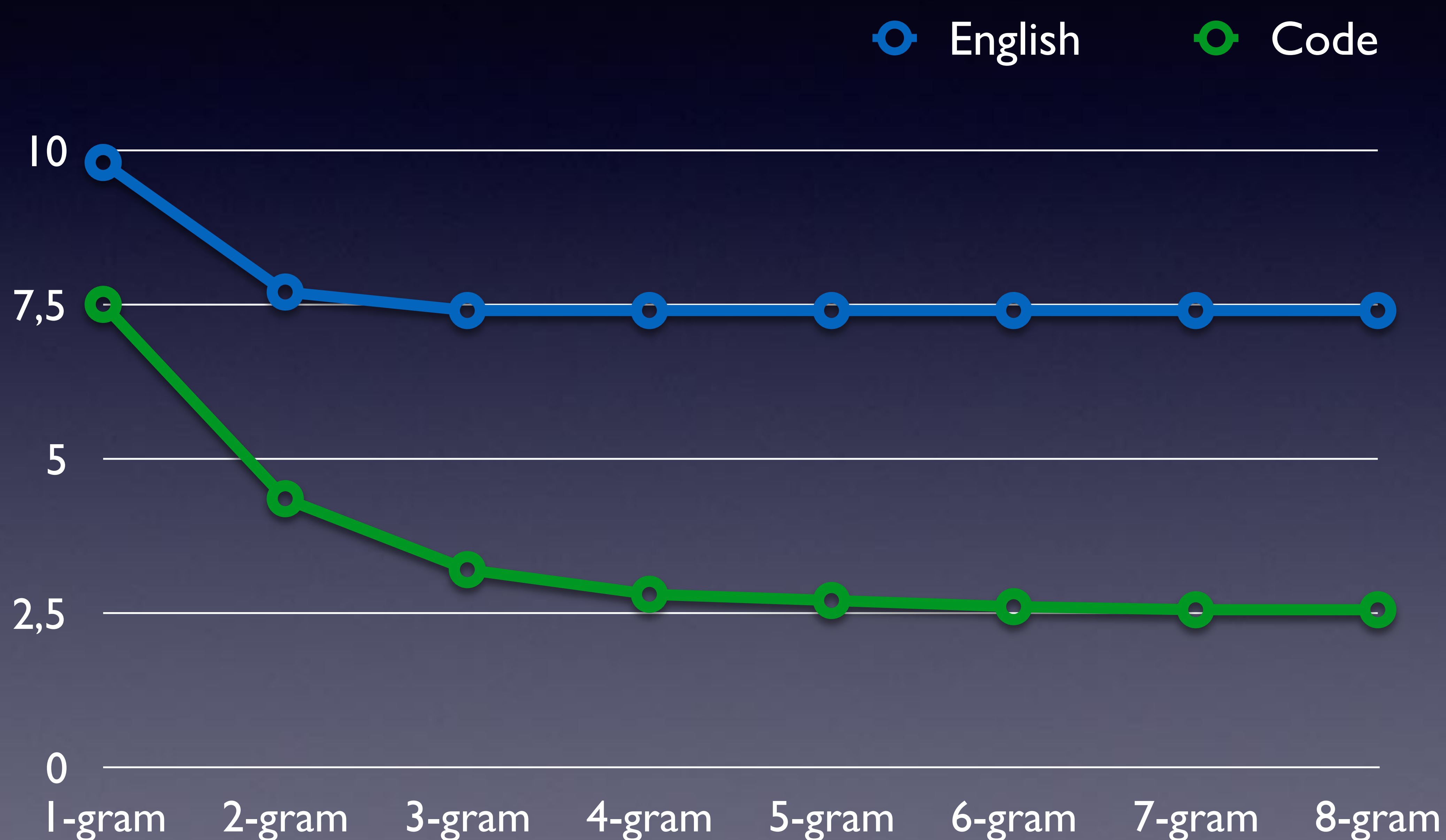
N-gram Cross Entropy



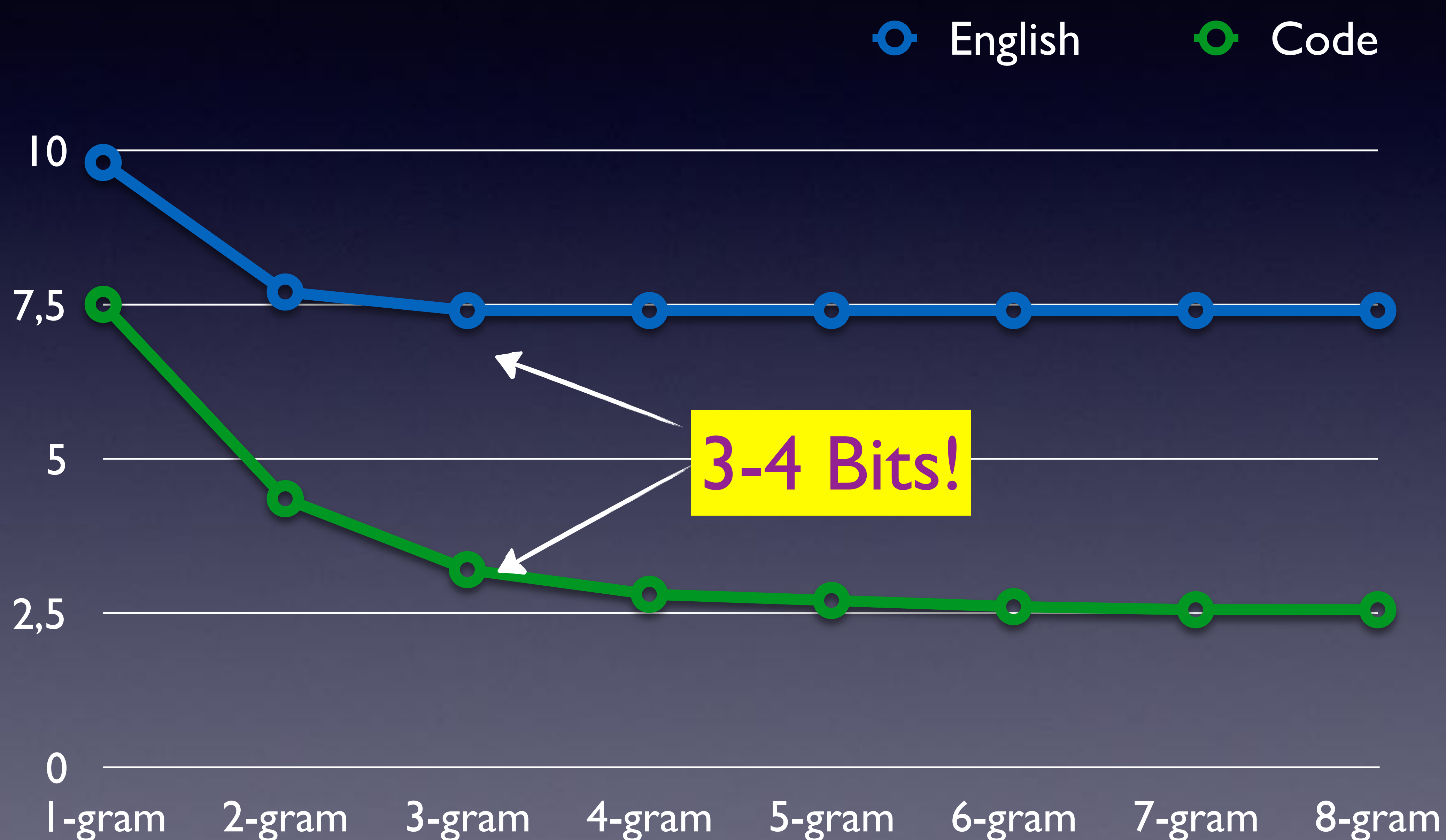
N-gram Cross Entropy



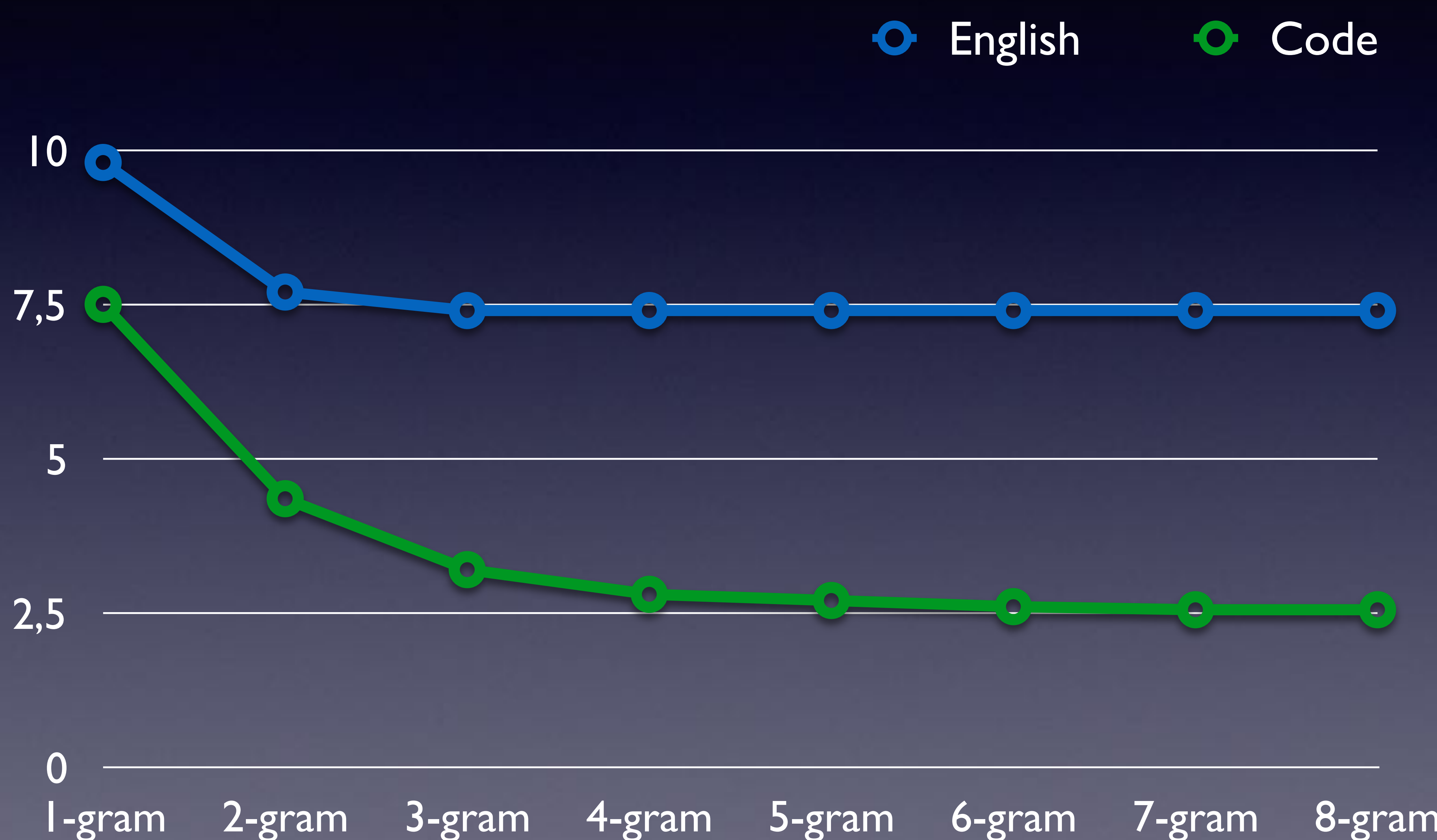
N-gram Cross Entropy



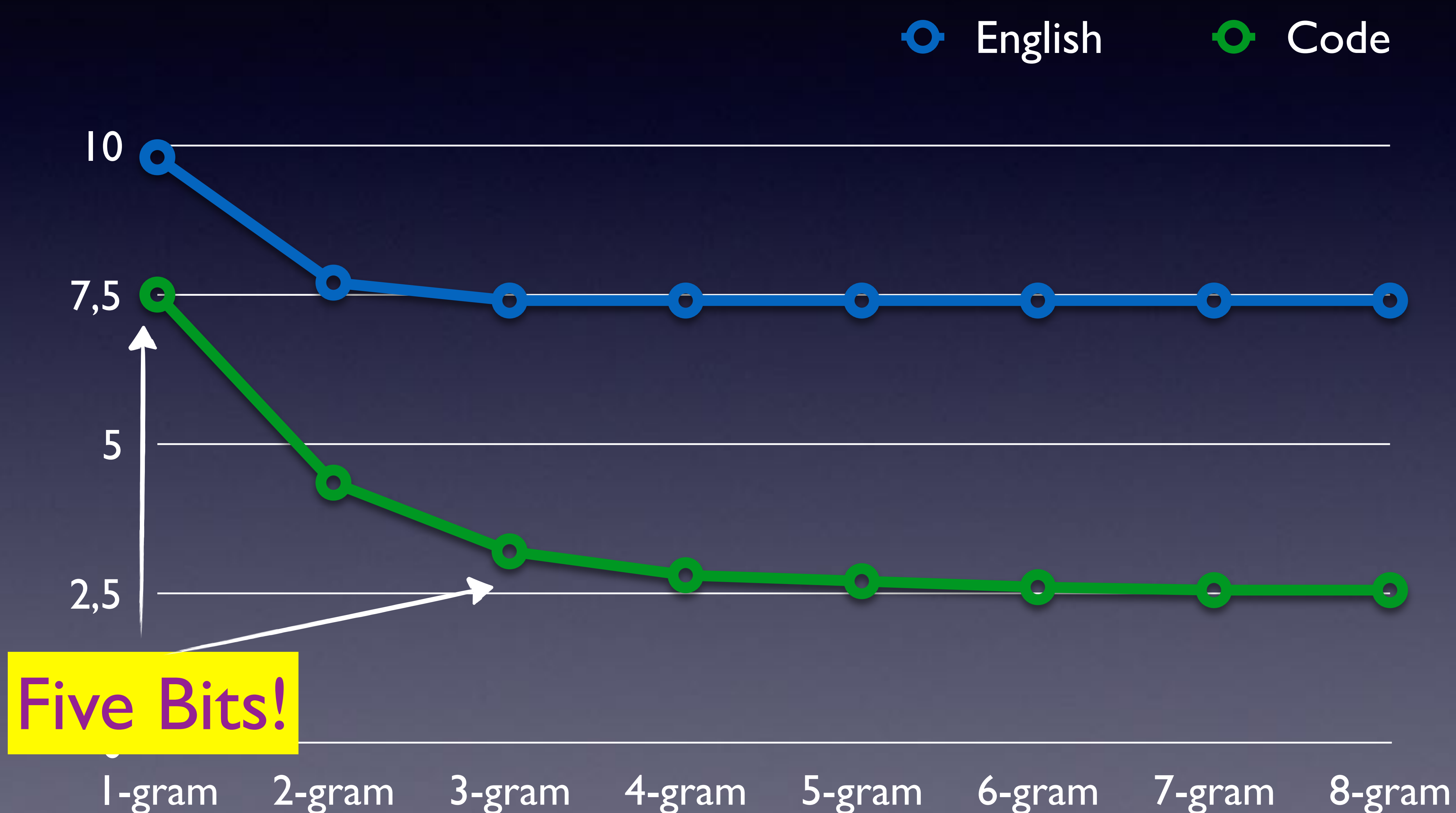
N-gram Cross Entropy



N-gram Cross Entropy



N-gram Cross Entropy



The Skeptic asks..

The Skeptic asks..

Is it just that C, Java, Python... are simpler than English?

The Skeptic asks..

Is it just that C, Java, Python... are simpler than English?

➡ Do cross-project testing!

The Skeptic asks..

Is it just that C, Java, Python... are simpler than English?

➡ Do cross-project testing!

➡ Train on one project, Test on the others.

The Skeptic asks..

Is it just that C, Java, Python... are simpler than English?

➡ Do cross-project testing!

➡ Train on one project, Test on the others.

➡ If it's all "in the language", entropy should be similar.

The "Naturalness" Vision

The "Naturalness" Vision

Suggest & Complete next tokens for developers

Assistive (speech, gesture) coding for convenience and disability.

Code Summarization & Retrieval

Porting

"Typo" Error Correction

Search-based Software Engineering.

The "Naturalness" Vision

Suggest & Complete next tokens for developers

Assistive (speech, gesture) coding for convenience and disability.

Code Summarization & Retrieval

Porting

"Typo" Error Correction

Search-based Software Engineering.

Hands-on time

- Instructions: http://bit.ly/vasilescu-midwest
- Need: Python, NLTK, Pygments