

# (No) Influence of Continuous Integration on the Commit Activity in GitHub Projects

**Sebastian Baltes**

 @s\_baltes

# Thanks to my co-authors!

## (No) Influence of Continuous Integration on the Commit Activity in GitHub Projects

Sebastian Baltes  
University of Trier  
Trier, Germany  
research@sbaltes.com

Jascha Knack  
University of Trier  
Trier, Germany  
jk@jascha-knack.de

Daniel Anastasiou  
University of Trier  
Trier, Germany  
anastasiou.daniel@gmail.com

Ralf Tymann  
University of Trier  
Trier, Germany  
s4ratyma@uni-trier.de

Stephan Diehl  
University of Trier  
Trier, Germany  
diehl@uni-trier.de

### ABSTRACT

A core goal of Continuous Integration (CI) is to make small incremental changes to software projects, which are integrated frequently into a mainline repository or branch. This paper presents an empirical study that investigates if developers adjust their commit activity towards the above-mentioned goal after projects start using CI. We analyzed the commit and merge activity in 93 GitHub

### ACM Reference Format:

Sebastian Baltes, Jascha Knack, Daniel Anastasiou, Ralf Tymann, and Stephan Diehl. 2018. (No) Influence of Continuous Integration on the Commit Activity in GitHub Projects. In *Proceedings of the 4th ACM SIGSOFT International Workshop on Software Analytics (SWAN '18), November 5, 2018, Lake Buena Vista, FL, USA*. ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/3278142.3278143>

# Martin Fowler's CI Practices (2006)

*„Continuous Integration is a software development practice where members of a team **integrate their work frequently** [...].“*



*„**Frequent commits** encourage developers to break down their work into **small chunks** of a few hours each.“*

*„[...] have a **mainline**: a single branch of the project currently under development.“*

# Question

**Small and frequent**  
commits into a **main**  
repository/branch

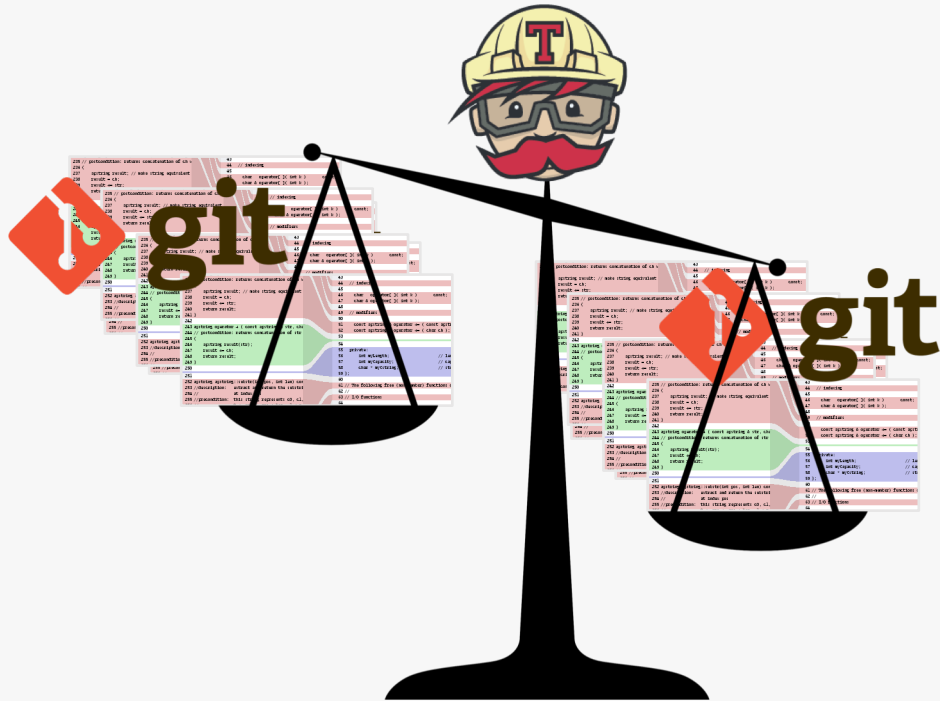


Do developers **adjust**  
**their commit activity**  
towards this goal  
after introducing CI?





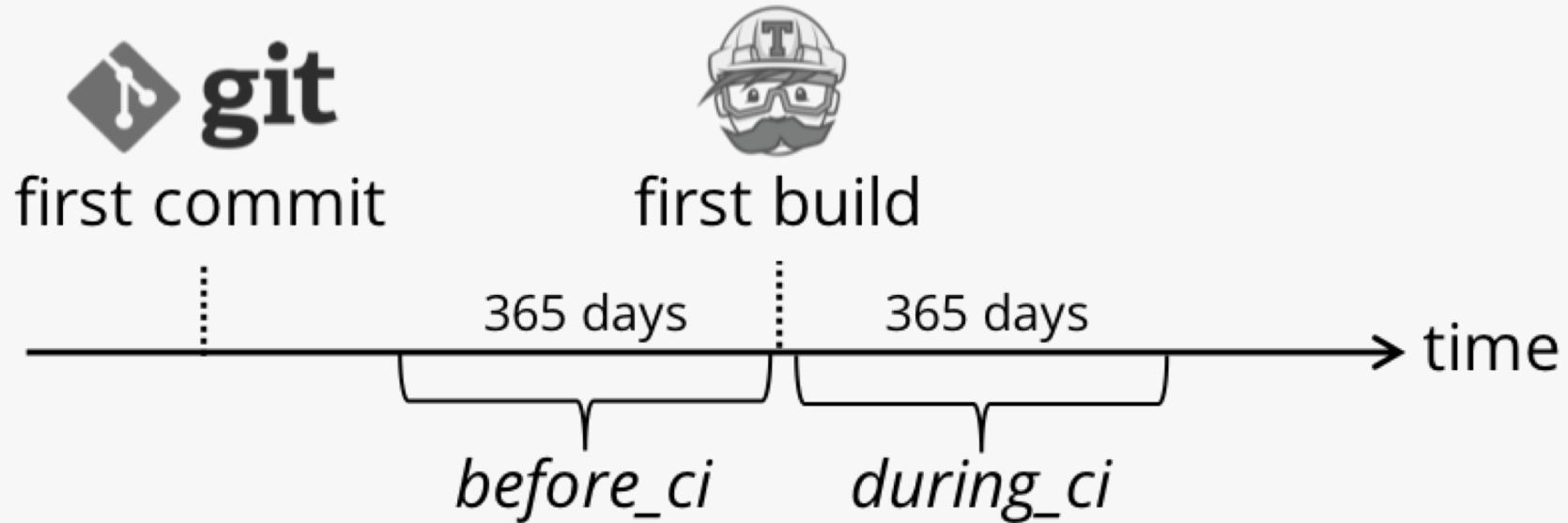
# Our Approach



Compare commit activity  
**before vs. after**  
introduction of CI

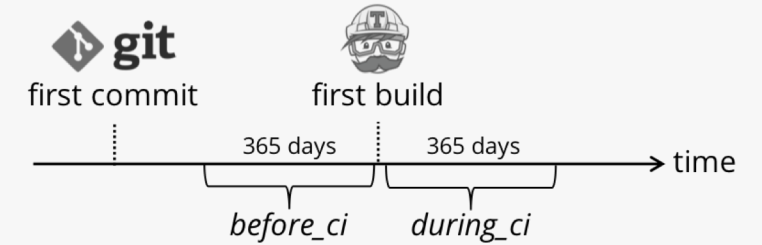
<http://open-tube.com/best-free-file-comparison-tools/>

# Study Design



- *Units of observation:* Commits in *GitHub* projects
- *Units of analysis:* Commits and projects
- *Compared measures:* Commit rate, commit size, merge ratio

# Study Design



## Filter criteria for *GitHub* projects:

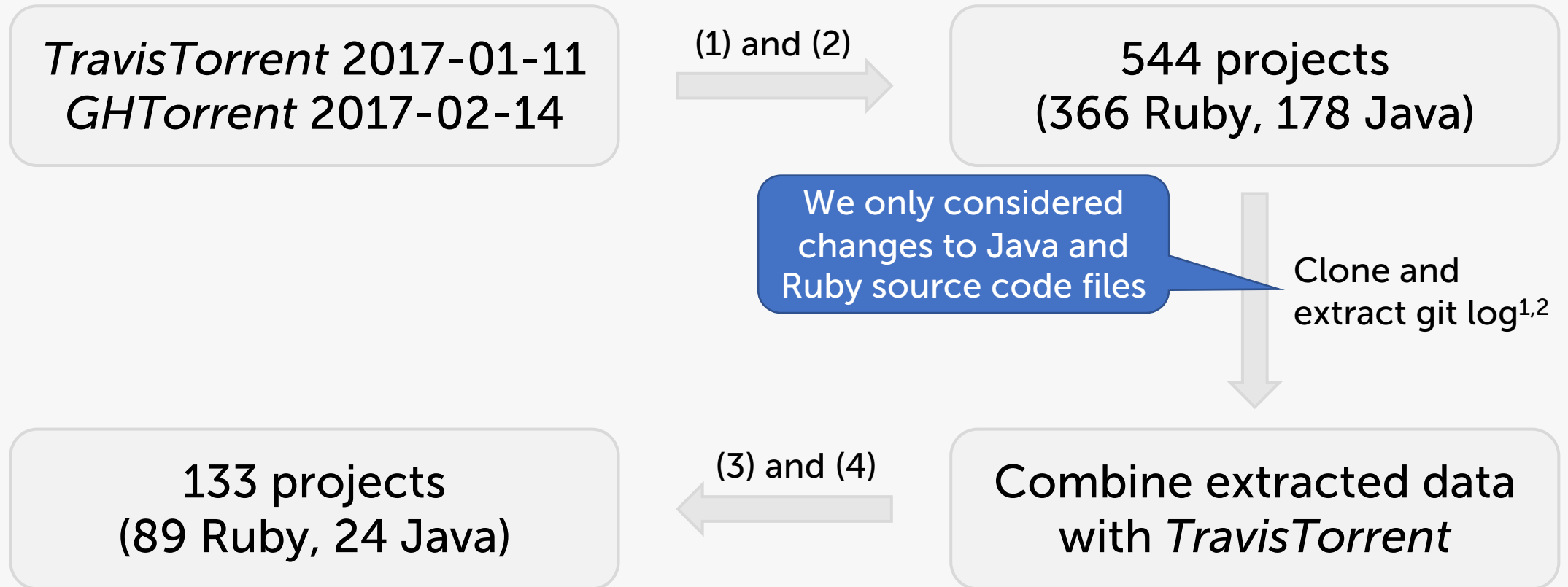
- (1) Active for  $\geq 1$  year before first build  
(*before\_ci*)
- (2) Used *TravisCI* for  $\geq 1$  year  
(*during\_ci*)
- (3) Commit activity on default branch  
in both phases
- (4) Used the default branch to trigger  
builds

Commit activity  
for comparison

"Mainline"

- (1) Active for  $\geq 1$  year before first build
- (2) Used *TravisCI* for  $\geq 1$  year
- (3) Commit activity on default branch
- (4) Used the default branch to trigger builds

# Filtering



<sup>1</sup> <https://github.com/git-log-extractor>

<sup>2</sup> <https://github.com/git-log-parser>

# Definitions

$\mathcal{C}$  set of all commits in all projects in our sample

$\mathcal{P}_{\mathcal{C}}$  power set of  $\mathcal{C}$

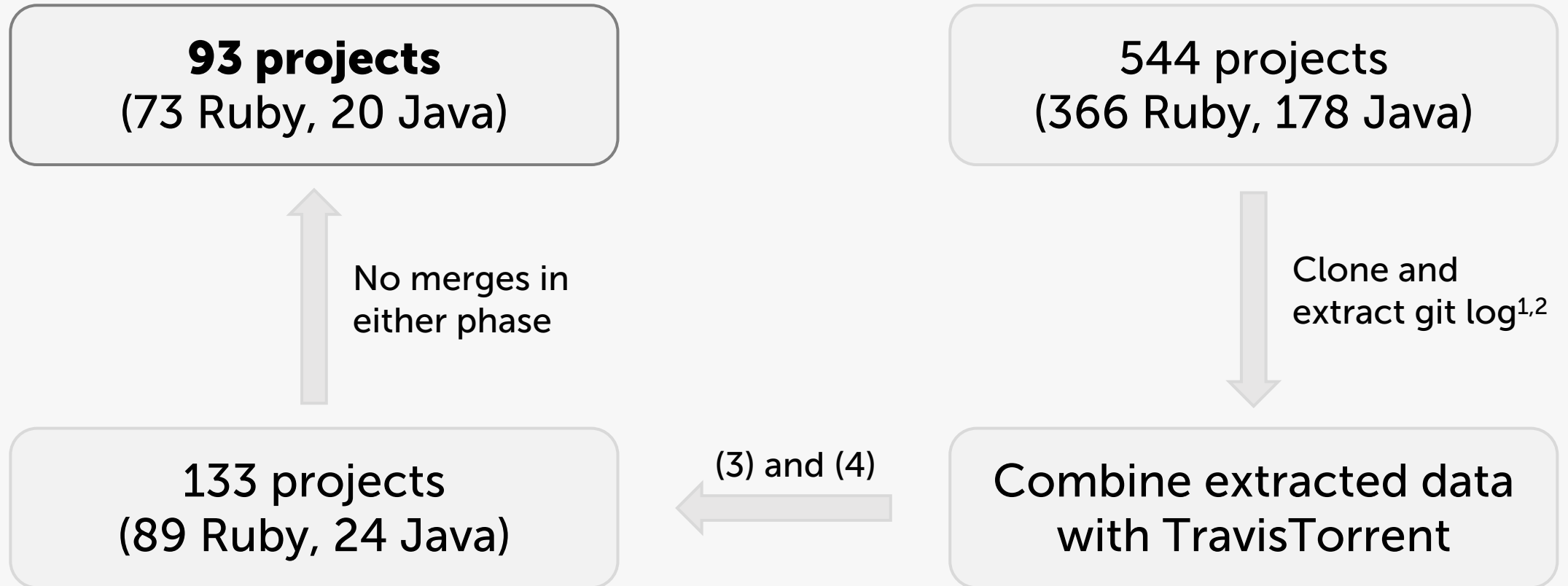
$C^w = C^{w_1} \dots C^{w_n}$  partition that divides a set of commits  $C$  into one subset for each week



No non-merging commit activity in 52% of the weeks.  
No merge activity in 71% of the weeks.

- (1) Active for  $\geq 1$  year before first build
- (2) Used *TravisCI* for  $\geq 1$  year
- (3) Commit activity on default branch
- (4) Used the default branch to trigger builds

# Filtering



<sup>1</sup> <https://github.com/git-log-extractor>

<sup>2</sup> <https://github.com/git-log-parser>



The background of the image is a collection of various measuring tools. There are several rulers of different lengths and colors, including yellow and white. A yellow folding ruler is prominent, running diagonally across the frame. Several white plastic rulers are also visible, some with red markings. In the top right corner, there is a yellow protractor with degree markings. The tools are arranged in a way that they overlap each other, creating a sense of depth and variety in measurement instruments.

# Compared Measures

# Commit Rate

**Definition 2.1** (Commit Rate). Let  $C \in \mathcal{P}_{\mathcal{C}}$  be a set of commits and  $C^w = C^{w_1} \dots C^{w_n}$  be a weekly partition of  $C$ . We define the *median commit rate*  $\overline{c_w}: \mathcal{P}_{\mathcal{C}} \rightarrow \mathbb{R}_0^+$  as:

$$\overline{c_w}(C) = \text{median}(|C^{w_i}|), \quad i \in \{1, \dots, n\}, \quad w_i \text{ is active}$$

$$\text{median}(1, 1, 1, 1, 1, 1, 1, 1, 1, 1) = 1$$

$$\text{median}(10, 0, 0, 0, 0, 0, 0, 0, 0, 0) = 0$$

$$\text{median}(10) = 10$$

← We only consider active weeks!



# Change Size



$n_f(c)$  number of source code files **changed** by a commit  $c \in C$

$l_c^+$  sum of all lines **added** to the modified files of a  
commit  $c \in C$

$l_c^-$  sum of all lines **deleted** from the modified files of a  
commit  $c \in C$

} Line-based  
git diff

$n_l(c) = l_c^+ + l_c^-$  **code churn** of commit  $c \in C$

# Change Size



**Definition 2.2** (Change Size). For a commit  $c \in \mathcal{C}$ , we define the change breadth  $b: \mathcal{C} \rightarrow \mathbb{R}_0^+$  and the change depth  $d: \mathcal{C} \rightarrow \mathbb{R}_0^+$  as:

$$b(c) = n_f(c) \quad \text{and} \quad d(c) = \frac{n_l(c)}{n_f(c)}$$

For a set  $C \in \mathcal{P}_{\mathcal{C}}$  of commits, we define the *median change breadth*  $\bar{b}(C)$  as:

$$\bar{b}(C) = \text{median}(b(c)), \quad c \in C$$

For a partition  $C^\phi = \{C^1 \dots C^n\}$  of  $C$ , we define the *median change breadth*  $\bar{b}(C^\phi)$  as:

$$\bar{b}(C^\phi) = \text{median}(\bar{b}(C^i)), \quad i \in \{1, \dots, n\}$$

The *median change depths*  $\bar{d}(C)$  and  $\bar{d}(C^\phi)$  are defined analogously.

# Merge Ratio

$n_m(C)$       number of merging commits in  $C \in \mathcal{P}_C$

**Definition 2.3** (Merge Ratio). For a set  $C \in \mathcal{P}_C$  of commits, we define the *merge ratio*  $m: \mathcal{P}_C \rightarrow [0, 1]$  as:

$$m(C) = \frac{n_m(C)}{|C|}$$

# Comparison Perspectives

## Origin:

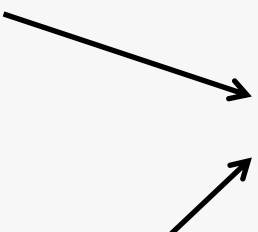
$C^p$  Partitions  $C$  into one set of commits for each project  $p \in \text{projects}(C)$ .

$C^{p,b}$  Partitions  $C$  into one set of commits for each branch in each project ( $\forall p \in \text{projects}(C) \forall b \in \text{branches}(p)$ ).

## Time:

$C_{before}$  Contains all commits before the first CI build for each project, but not more than 365 days before the first build.

$C_{during}$  Contains all commits after the first but before the last CI build for each project, but not more than 365 days after the first build.



$C^{p,b}_{before}$

# Comparison Perspectives

In the following:

$$C^p \hat{=} C^{p,b} : b = \text{default\_branch}(p)$$

$$C_{\text{before}}^p \quad \text{vs.} \quad C_{\text{during}}^p$$



Default branch usually triggers the builds.  
Our sample: 81.8% of the builds



The image features a dense, 3D perspective view of a grid of small cubes. The cubes are colored in four distinct ways: light blue, orange, red, and white. They are arranged in a regular grid pattern, but some cubes are slightly offset or raised, creating a sense of depth and movement. A semi-transparent white rectangular box is centered in the upper-middle portion of the image, containing the word "Results" in a black, sans-serif font. The background is a dark gray, and the lighting creates soft shadows on the surface beneath the cubes.

Results



# Results: Commit Rate

**Hypothesis:** After the introduction of CI, the commit rate increases.

$$\{\overline{c}_w(C_{before}^p)\} \text{ vs. } \{\overline{c}_w(C_{during}^p)\}, p \in projects(C)$$



We failed to reject the null hypothesis that the commit rate decreases or does not change.\*

\*Significance test: two-sided Wilcoxon signed rank test,  $\alpha = 0.01$ , effect size: Cliff's delta

# Results: Change Size

## Hypothesis:

After the introduction of CI, the commit changes become smaller, they have a lower change breadth and depth.

$$\{\bar{b}(C_{before}^p)\} \text{ vs. } \{\bar{b}(C_{during}^p)\}, p \in projects(C)$$

$$\{\bar{d}(C_{before}^p)\} \text{ vs. } \{\bar{d}(C_{during}^p)\}, p \in projects(C)$$



We failed to reject the null hypothesis that the change size decreases or does not change.\*

\*Significance test: two-sided Wilcoxon signed rank test,  $\alpha = 0.01$ , effect size: Cliff's delta



# Results: Merge Ratio ↶

**Hypothesis:** After the introduction of CI, the merge ratio increases.

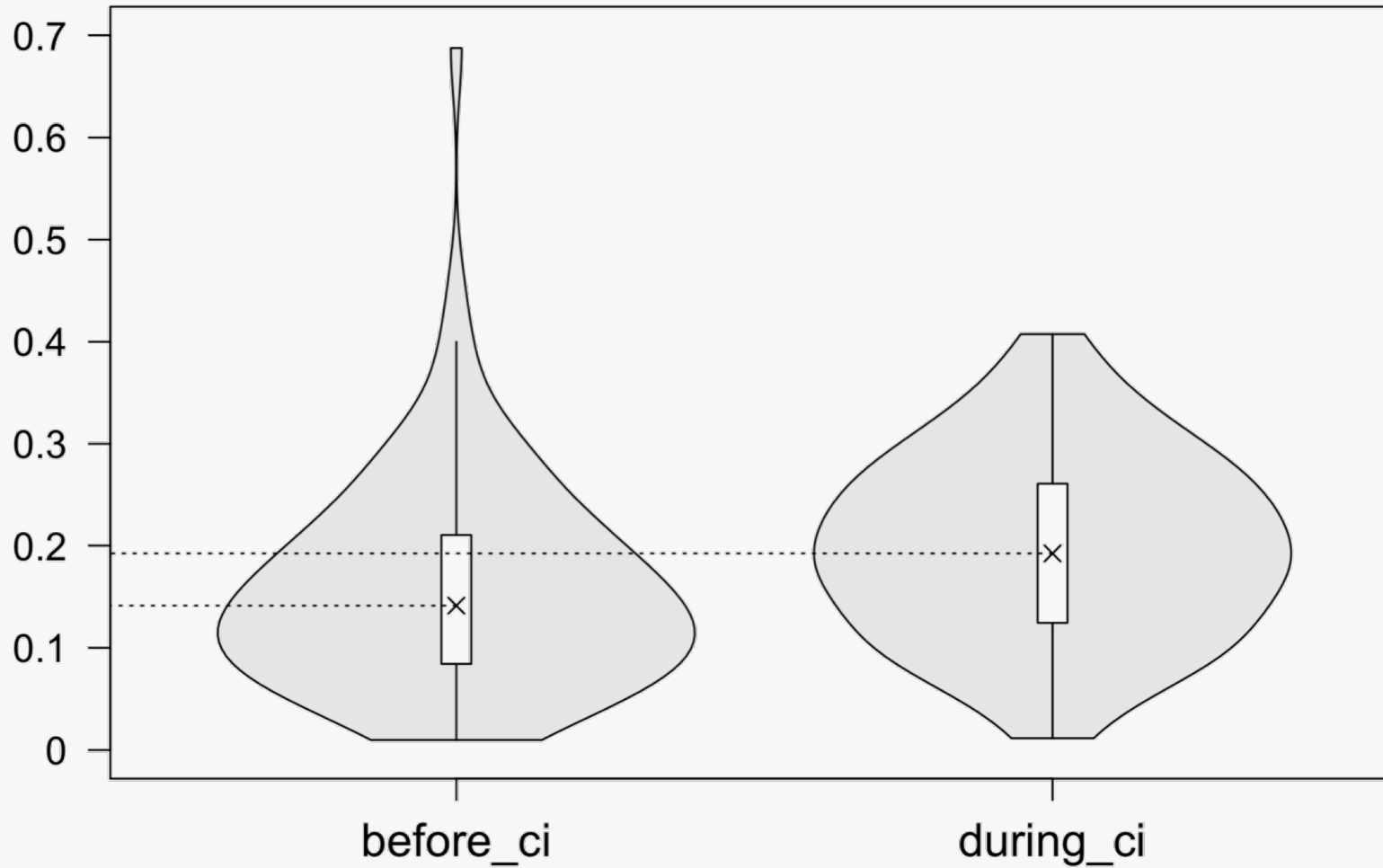
$$\{m(C_{before}^p)\} \text{ vs. } \{m(C_{during}^p)\}, p \in projects(C)$$



We rejected the null hypothesis that the merge ratio decreases or does not change.\*

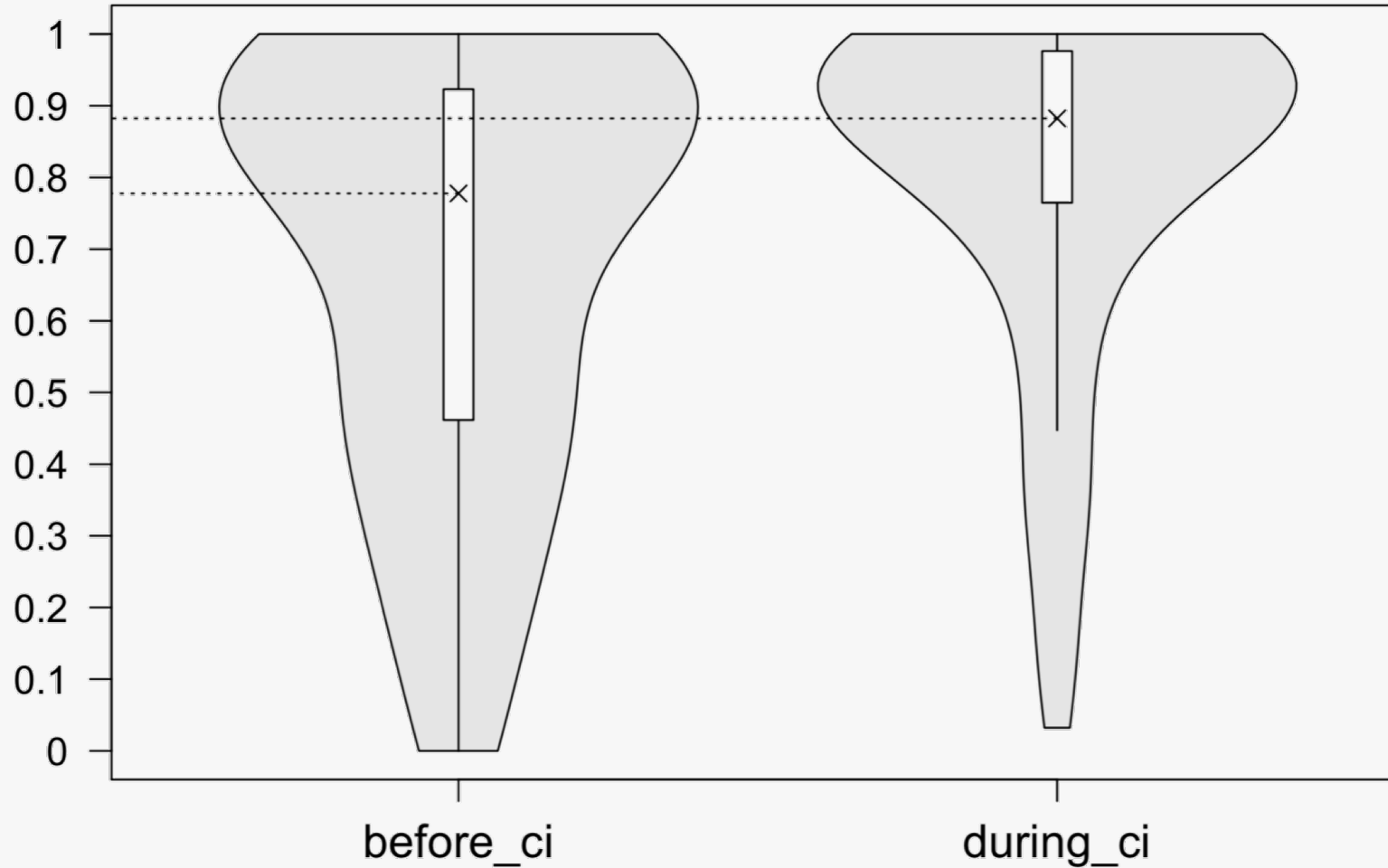
\*Significance test: two-sided Wilcoxon signed rank test,  $\alpha = 0.01$ , effect size: Cliff's delta

# Results: Merge Ratio



$$\delta = 0.28 \text{ (small)}$$
$$CI_{\delta} = [0.12, 0.43]$$

# Results: Pull Request Ratio



$$\delta = 0.24 \text{ (small)}$$
$$CI_{\delta} = [0.08, 0.40]$$

# What caused the change?



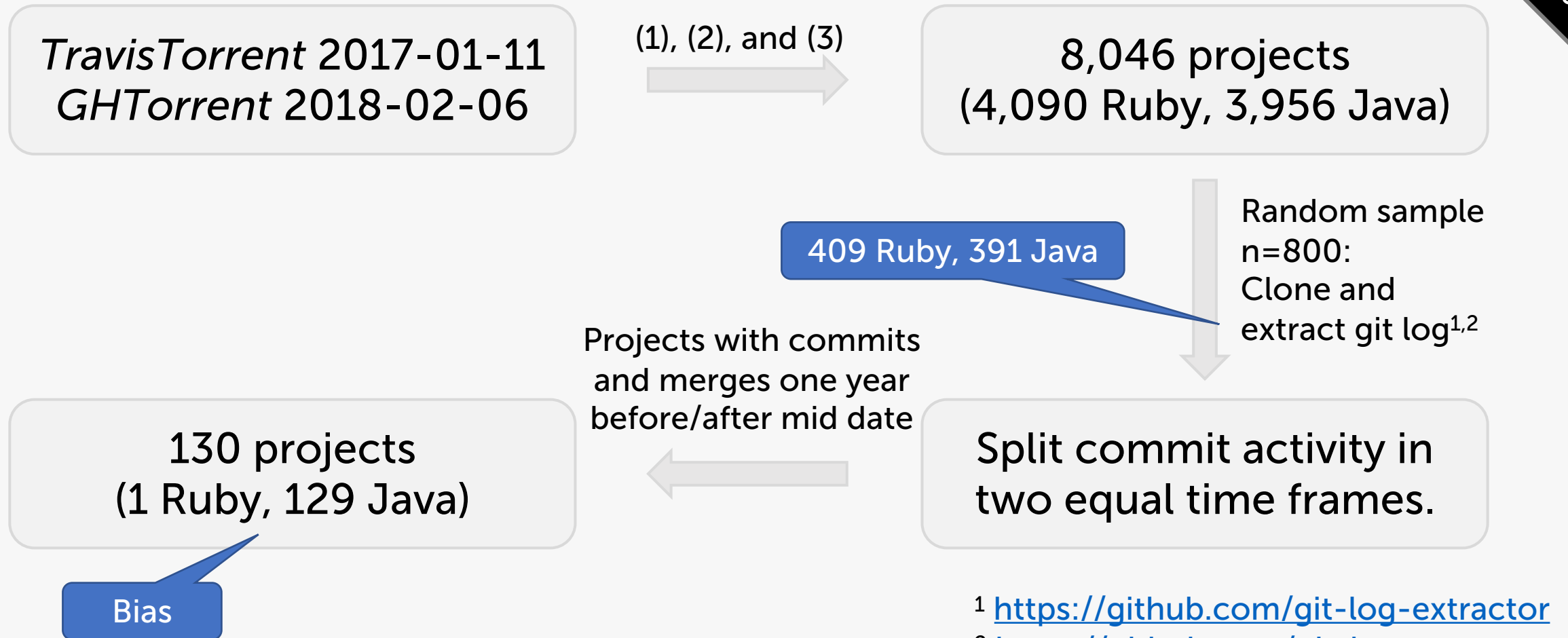
# Comparison Sample

- Can the increased merge ratio be **attributed** to the introduction of CI?
- Analyze random **comparison sample** of projects not using CI
- **Filter criteria** for *GitHub* projects:
  - (1) Java or Ruby as project language
  - (2) Not in the *TravisTorrent* dataset
  - (3) Commit activity  $\geq 2$  years ( $\geq 730$  days)
  - (4) Are “engineered” software projects ( $\geq 10$  watchers)



- (1) Java or Ruby as project language
- (2) Not in the *TravisTorrent* dataset
- (3) Commit activity for  $\geq 2$  year
- (4) “Engineered” software projects

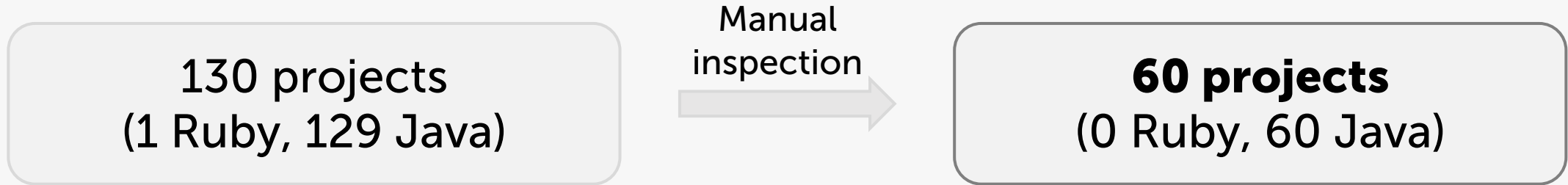
# Filtering



<sup>1</sup> <https://github.com/git-log-extractor>

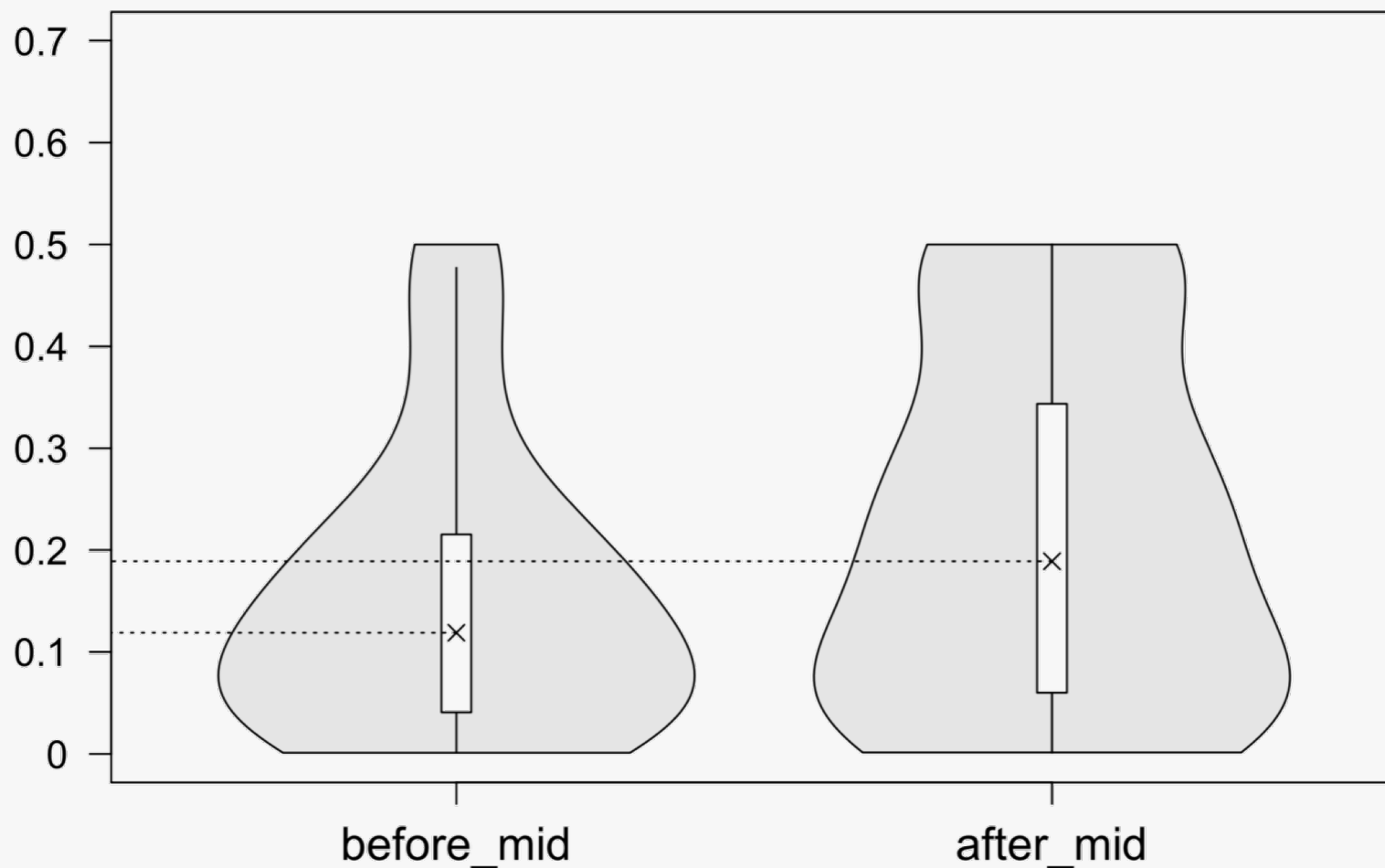
<sup>2</sup> <https://github.com/git-log-parser>

# Filtering



- No indication that those 60 projects use CI
- Next step: Compare merge ratio one year before/after mid date

# Results: Comparison Sample



$$\delta = 0.20 \text{ (small)}$$

$$CI_{\delta} = [-0.008, 0.40]$$

**Similar effect as in the CI sample.**



# Limitations

- Focus on **Ruby and Java** projects
- Projects may have used a **different CI tool** than *TravisCI* (*before\_ci* actually *during\_ci*)
- Comparison sample:
  - **Different split** date could yield different results
  - No Ruby projects
- In commercial projects, the number of **inactive weeks** is likely to be lower (transferability limited)
- Simple **two-group comparison** of median values (no advanced statistical modeling)



<https://schoolofauthors.wordpress.com/2017/03/02/importance-of-expressing-study-limitations/>

# Conclusion



- Only found one effect, an **increased merge ratio**
- Observed same effect in sample of projects not using CI
- We **do not attribute this effect** to the introduction of **CI**
- More likely:
  - Projects use merges more frequently when they **grow** and mature
  - General trend of adopting the **pull-based software development**
- Results show **importance of having a baseline** when observing trends

Sebastian Baltes

 @s\_baltes

**[sbaltes.com/ci](https://sbaltes.com/ci)**

*Data and scripts available on Zenodo/GitHub*