Empirical analysis of the relationship between CC and SLOC in a large corpus of Java methods

Davy Landman - SWAT

Joint work with: Alexander Serebrenik and Jurgen Vinju

Software Metrics Revisited

Davy Landman - SWAT

Joint work with: Alexander Serebrenik and Jurgen Vinju

Software Transformations

SWAT

Analysis

Software

Transformations

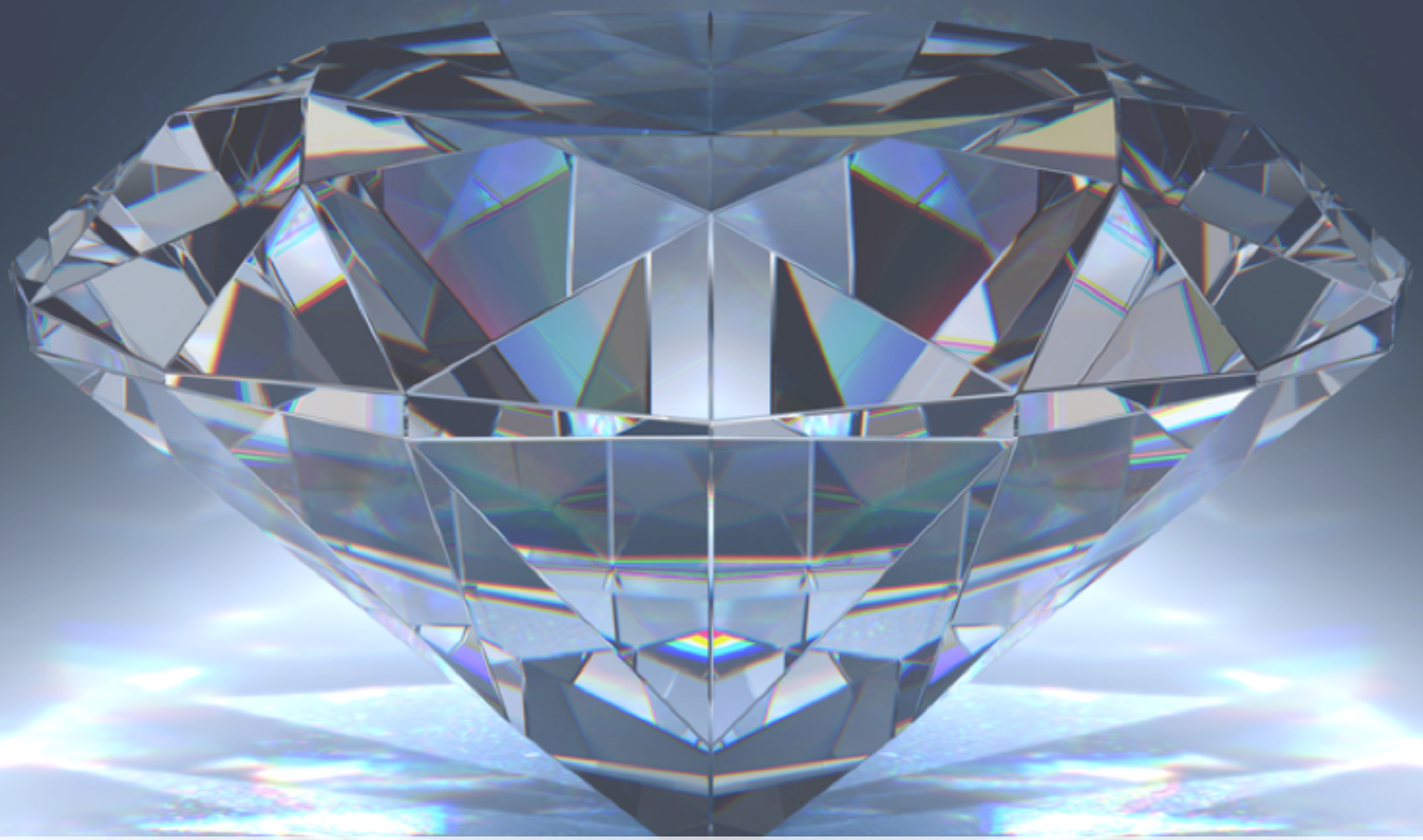
SWAT

Analysis

- Semantics
- Knowledge
- Quality

Software Quality

Software Quality



- Are there defects in my software?
- Can I extend my software easily?

Common solutions

- Ask developers
- Analyse bug reports
- Calculate software metrics

Common software metrics

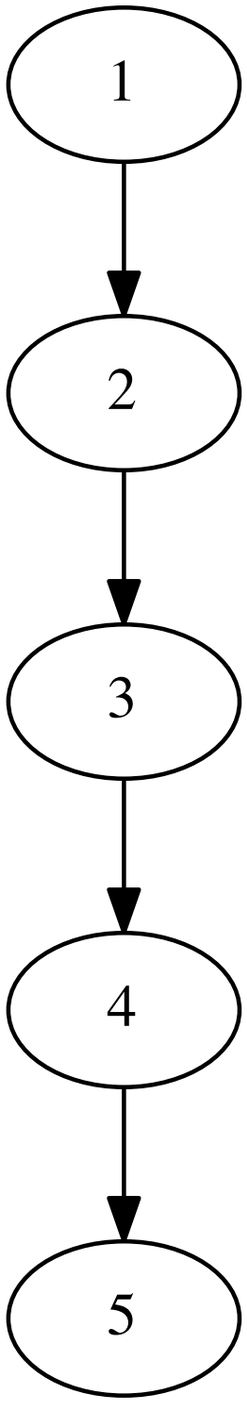
- Lines of Code (LOC/SLOC)
- Cyclomatic Complexity (CC)
  - Count linear independent paths in the control flow graph
- Popular in practice and research

Common software metrics

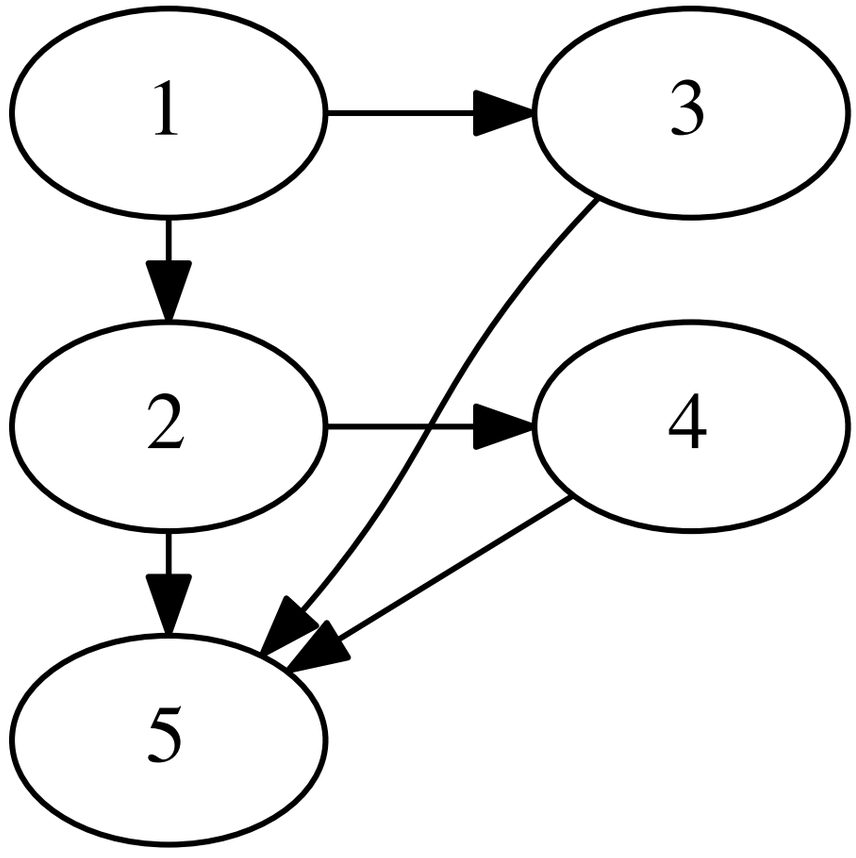
- Lines of Code (LOC/SLOC) = 7
- Cyclomatic Complexity (CC) = 2
  - Count linear independent paths in the control flow graph

```
1. public double sqrt(int n){
2.   // Newton-Raphson method
3.   double r = n / 2.0;
4.   while (abs(r - (n / r)) > 0.00001) {
5.     r = 0.5 * (r + (n / r));
6.   }
7.   return r;
8. }
```

Cyclomatic Complexity



CC = 1



CC = 3

A critique of cyclomatic complexity as a software metric

by Martin Shepperd

McCabe's cyclomatic complexity metric is widely cited as a useful predictor of various software attributes such as reliability and development effort. This critique demonstrates that it is based upon poor theoretical foundations and an inadequate model of software development. The argument that the metric provides the developer with a useful engineering approximation is not borne out by the empirical evidence. **Furthermore, it would appear that for a large class of software it is no more than a proxy for, and in many cases is outperformed by, lines of code.**

1 Introduction

The need for some objective measurement of software complexity has been long acknowledged. Two early contributions to this field are Halstead's 'software science' (Ref. 1) and the cyclomatic complexity approach of McCabe (Ref. 2). Both metrics are based upon the premise that software complexity is strongly related to various measurable properties of program code.

Although initially well received by the software engineering community, software science based metrics have been increasingly subject to criticism. Attacks have been made upon the underlying pyschological model (Refs. 3 and 4). The soundness of many empirical 'validations' has been questioned (Ref. 5) and difficulties noted with counting rules (Ref. 6). The ability of software science metrics to capture program complexity in general would thus appear to be in

M. Shepperd. "A critique of cyclomatic complexity as a software metric." Software Engineering Journal 3.2 (1988)

CC redundant?

- We studied papers analyzing CC & SLOC
- 33 papers from 1979 till 2014
- Fortran, Pascal, COBOL, C, C++, Java, Python,..
- Consensus: SLOC & CC correlate linearly

$$R^2 = 0.60\text{-}0.95$$

Our research

- Detect differences between 33 papers

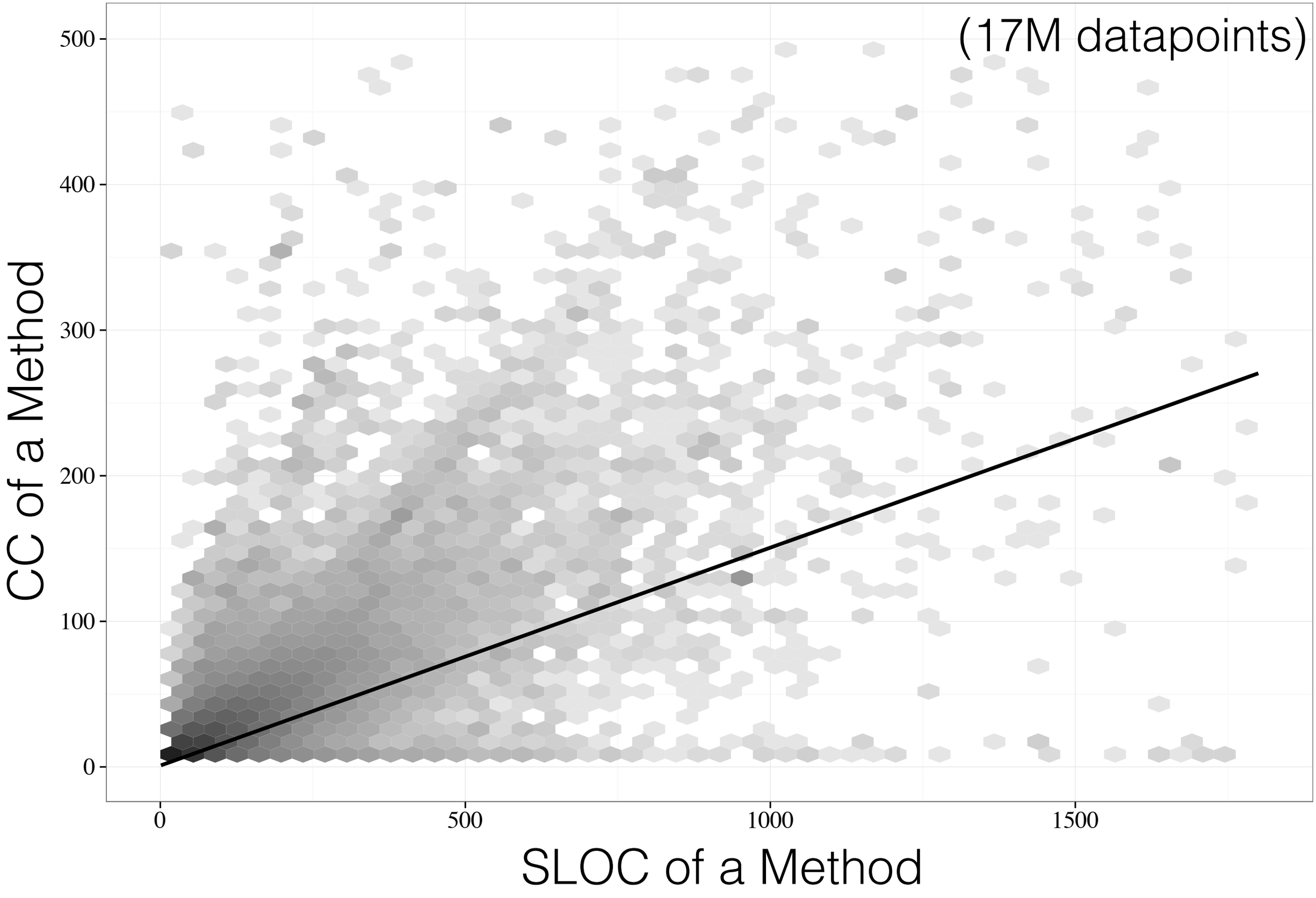
- Create large dataset[1,2]

  (23K projects, 24M methods, 450MSLOC)

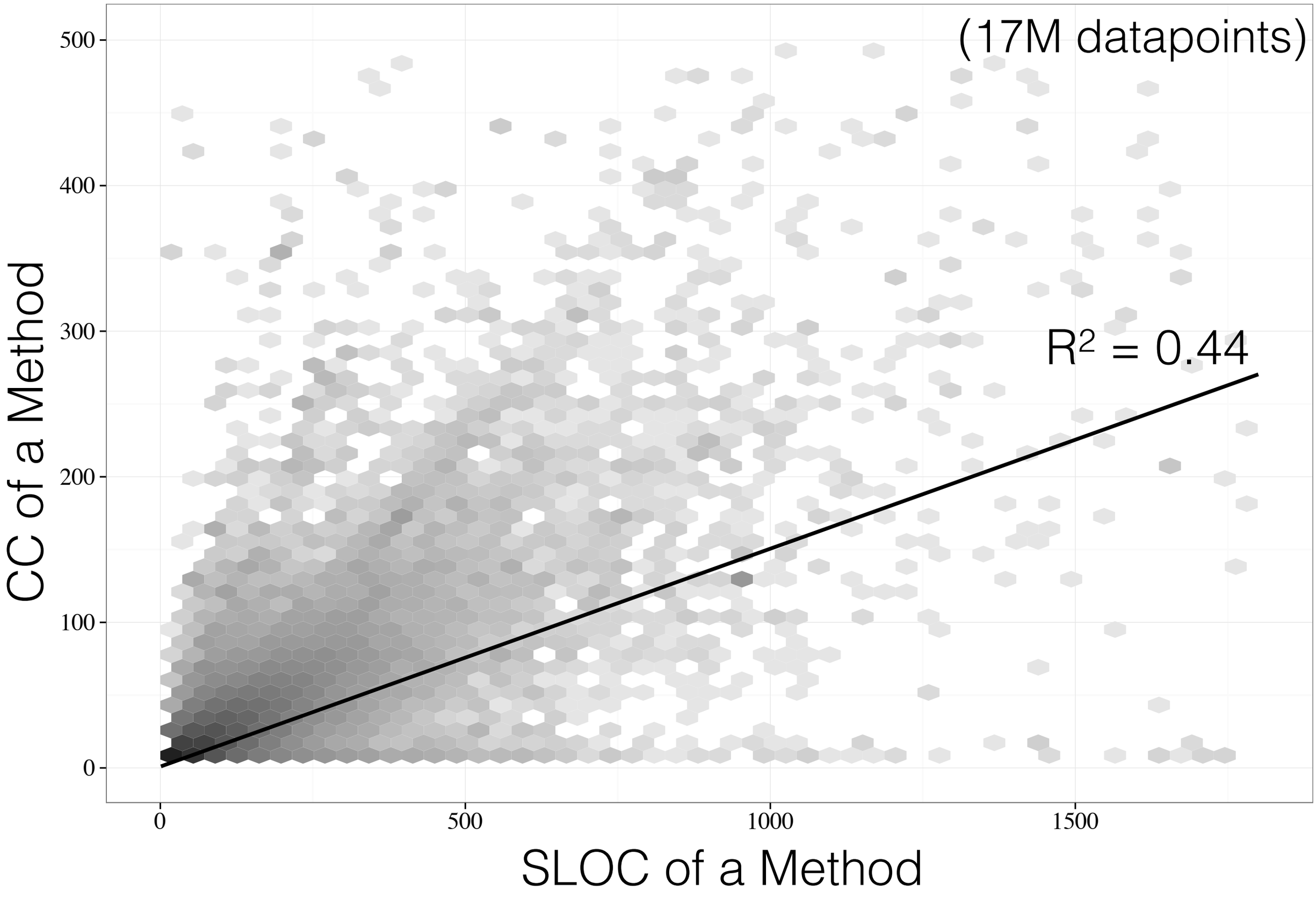
- Reproduce!

[1] https://repository.cwi.nl/noauth/search/fullrecord.php?publnr=23154
[2] https://repository.cwi.nl/noauth/search/fullrecord.php?publnr=23357

Weak linear correlation



Weak linear correlation



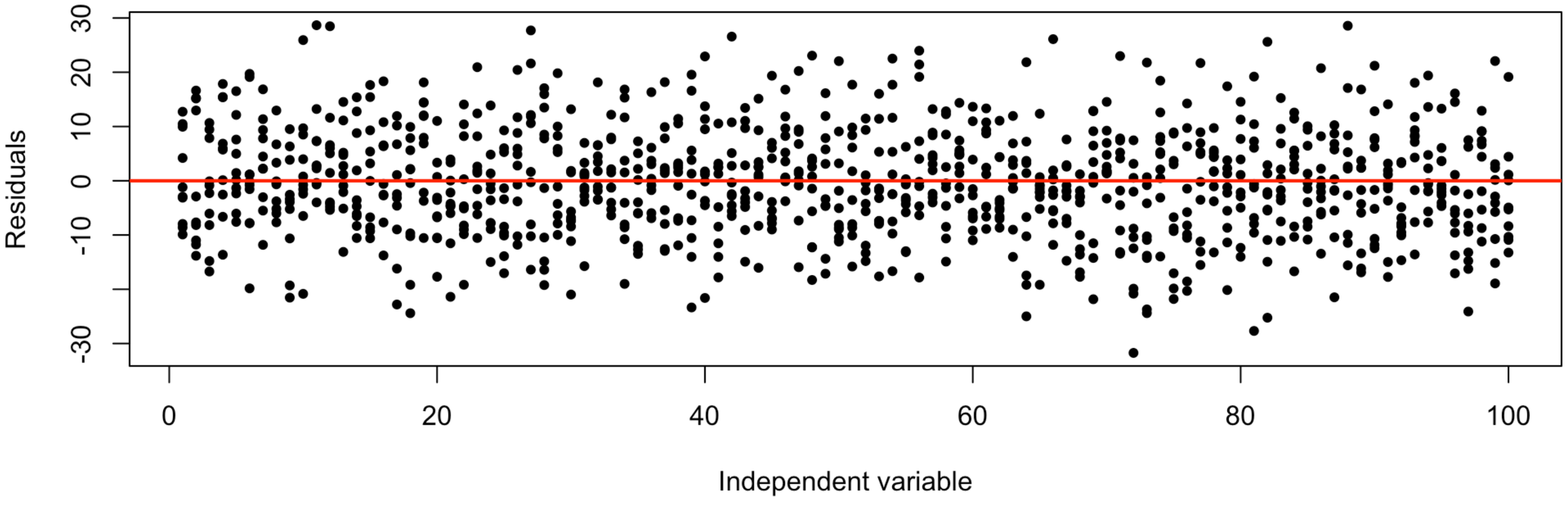
Result

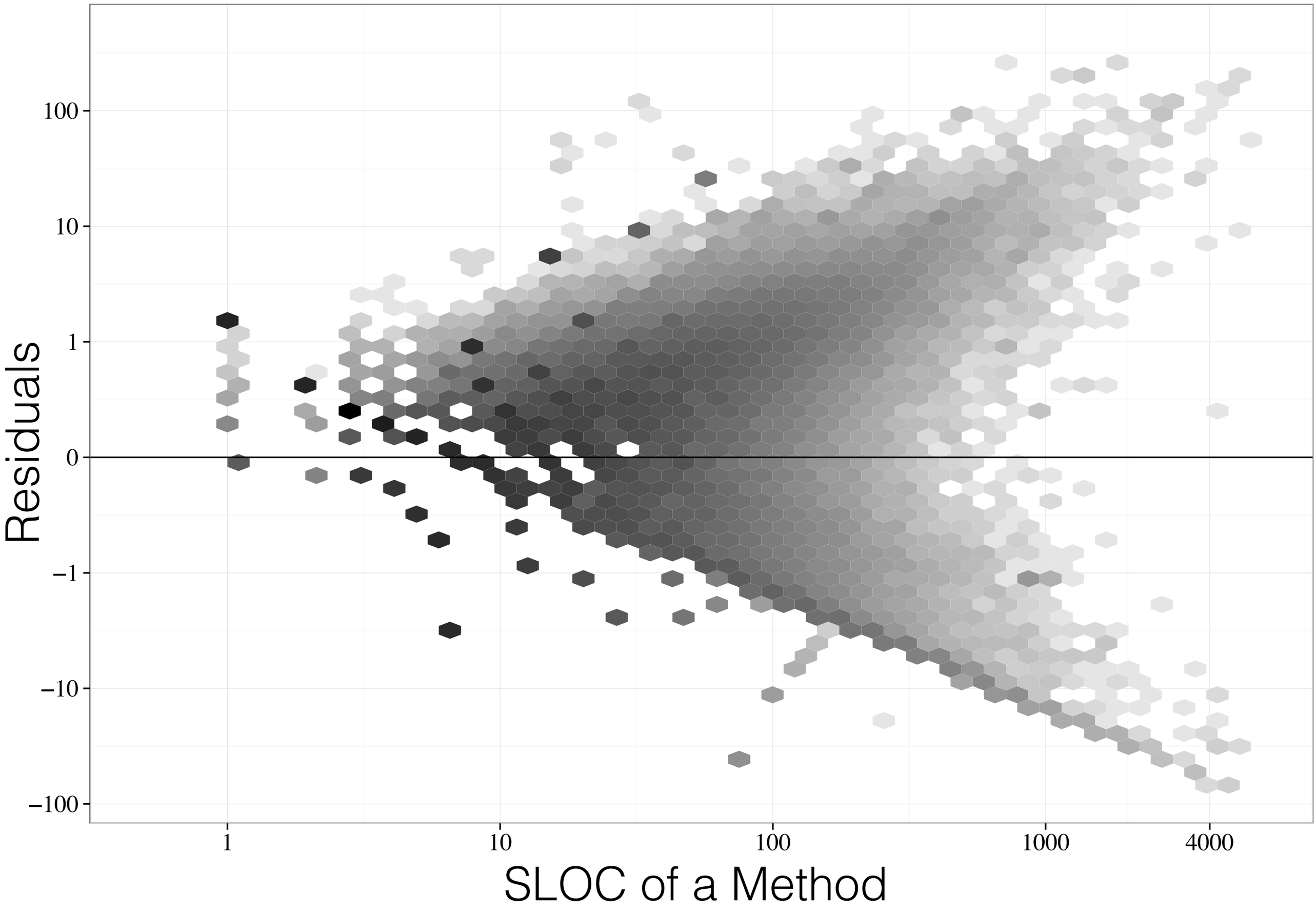
we do **not** conclude that
CC is redundant with SLOC

- Our result: $R^2 = 0.44$
- Other influential factors researched:
  - Transformation of the data (log/aggregate)
  - Impact of older/smaller data sets

Variance?

- R^2 = **0.44** means **56%** variance not explained
- Variance = actual CC – predicted CC




Residuals
100
10
1
0
−1
−10
−100
1
10
100
1000
4000
SLOC of a Method

Summary

- Common theory is that CC linearly correlates with SLOC
- All existing work supports this theory
- Using two large corpora of Open Source Software
- We carefully invalidated this theory