Measurement of x10 Productivity Differences Replicates!

“One of the most replicated results in software engineering research is the 10-fold difference in productivity and quality between different programmers with the same levels of experience.” – Steve McConnell [McConnell, 2002]

<table>
<thead>
<tr>
<th>Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Sackman, 1968]</td>
</tr>
<tr>
<td>[Card 1987]</td>
</tr>
<tr>
<td>[Curtis 1981]</td>
</tr>
<tr>
<td>[Boehm and Papaccio 1988]</td>
</tr>
<tr>
<td>[Mills 1983]</td>
</tr>
<tr>
<td>[Valett and McGarry 1989],</td>
</tr>
<tr>
<td>[DeMarco and Lister 1985]</td>
</tr>
<tr>
<td>[Boehm 1975, 2000]</td>
</tr>
<tr>
<td>[Curtis et al. 1986]</td>
</tr>
<tr>
<td>[Schwartz, 1968]</td>
</tr>
</tbody>
</table>

Reported Productivity ranges of 5-1 to 28-1
Measurement of x10 Productivity Differences Replicates!

“One of the most replicated results in software engineering research is the 10-fold difference in productivity and quality between different programmers with the same levels of experience.” – Steve McConnell [McConnell, 2002]

But they were wrong! I will show some data to challenge how we think about this and maybe lead to a more nuanced view.

Reported Productivity ranges of 5-1 to 28-1
Why Does Productivity Range Matter?

Huge productivity ranges between programmers should affect

• Benchmarks – I want to be the best!
• Hiring - we want the best!
• Training – we make the best!
• Evaluation – we only keep the best!
• Promotion – we recognize the best!
• Pay – we pay the best!
• Planning – we need the best!
Questions

1. Does our data replicate the scale of performance range?
2. How much does programmer productivity vary?
3. What factors (experience, education, the individual) account for the variation?
4. What practical effect will the differences have?
Approach

Sponsored by the SCOPE SEI research project.

SCOPE focused on identifying causal relationships from observational data to investigate potentially controllable causes of software development cost.

Random Controlled Trials (RCT) are usually not practical in software engineering, so we looked for ways to infer causal relationships from observational data.

SCOPE included cutting edge ML techniques for causal inference sponsored by NIH. (Center for Causal Discovery)

Use the performance data collected during the Personal Software Process (PSP) course.
PSP Course Students Write 10 Programs

PSP components
- Record data with forms and logs
- Follow a process with defined
  - Scripts, and
  - Standards

Programs vary in size, but typically take an afternoon to complete.
PSP Data

We have a lot of data, and this is dataset is unique.
This version PSP course was taught for about 10 years, primarily to professionals.
Each programmer developed the same 10 programs.
The programming data includes
- 3140 developers and 31,140 programs
- 3,355,882 lines of code
- 123,996.53 hours of work
- 221,346 defects
We also collected demographic data for experience, education, and so forth.
Of these, 494 include complete sets of 10 programs written in “C”.
We will only look at these data.
Data Selection And Analysis

Selection
Limit context to reduce confounding.
• Focus only on Programming Effort (limit the scope).
• Limit sample to C Programmers (constrain confounders).
• Also have large samples of C++, C#, Java, and VB

Analysis
• Apply causal search techniques (find direction of causality).
• Apply Linear Hierarchical Models and SEM to measure effect size and quantify variation.
Causal Search Found “Something about Students”

1) There were no correlations (let alone causation) between experience, education level, or statistics background and productivity.

2) Evidence that Student (j) caused outcome on Assignment (i)

\[
\text{ProgramSize}[\text{LOC}]_{ij} = \text{ReqSize}_i \times \text{StudentSizeFactor}_j \\
\text{Defects}_{ij} = \text{ProgrammingEffort}_{ij} \times \text{StudentDefectRate}_j \\
\text{ProgramEffort}_{ij} = \text{RequirementSize}_i \times \text{StudentEffortFactor}_j
\]

3) But the “within programmer” variation was surprisingly strong.

Error terms from mixed model (ANOVA) and SEM showed individual programmer results were only almost as variable as between programmers.

Followed up with closer look at the variation.
Total Programmer Relative Effort

Distribution of Programmer's Relative Effort for all programs
( Min 23) (Mean 100.0+/-5) (Median 86.5+/-4) (Max 379)

Student Relative Effort = \( \frac{\text{Individual Effort}}{\text{Average Effort}} \)
Total for all programs

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Standard deviation</th>
<th>5%</th>
<th>25%</th>
<th>Median</th>
<th>75%</th>
<th>95%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relative effort</td>
<td>1</td>
<td>0.51</td>
<td>0.44</td>
<td>0.63</td>
<td>0.86</td>
<td>1.24</td>
<td>1.93</td>
</tr>
</tbody>
</table>
Total Programmer Relative Effort

Distribution of Programmer’s Relative Effort for all programs
( Min 23) (Mean 100.0+/-.5) (Median 86.5+/-.4) (Max 379)

Student Relative Effort = \frac{\text{Individual Effort}}{\text{Average Effort}}

Total for all programs
Some support for x10 range
But relies on outliers.

<table>
<thead>
<tr>
<th>Relative effort</th>
<th>Mean</th>
<th>Standard deviation</th>
<th>5%</th>
<th>25%</th>
<th>Median</th>
<th>75%</th>
<th>95%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>0.51</td>
<td>0.44</td>
<td>0.63</td>
<td>0.86</td>
<td>1.24</td>
<td>1.93</td>
</tr>
</tbody>
</table>
Total Programmer Relative Effort

Student Relative Effort = \frac{\text{Individual Effort}}{\text{Average Effort}}

Total for all programs

Some support for x10 range

But relies on outliers.

Most within x2

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Standard deviation</th>
<th>5%</th>
<th>25%</th>
<th>Median</th>
<th>75%</th>
<th>95%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relative effort</td>
<td>1</td>
<td>0.51</td>
<td>0.44</td>
<td>0.63</td>
<td>0.86</td>
<td>1.24</td>
<td>1.93</td>
</tr>
</tbody>
</table>
Total Programmer Relative Effort

Student Relative Effort = \( \frac{\text{Individual Effort}}{\text{Average Effort}} \)

Total for all programs

Some support for x10 range
But relies on outliers.
Most within x2 effort
Extremes about x5 effort

So where did the very wide ranges come from?
Quartiles of Relative Effort for the 10 Programs

Run Chart

Relative Effort = \frac{\text{Individual Effort}}{\text{Average Effort}}

Programmer Relative Productivity by Program

Normalized Effort by Program

Mean = 1.0
Min/Max of Relative Effort by Program Are Outliers

Program Max/Max extreme and variable

Someone always did really well
Someone always did poorly
Min/Max Effort by Program Are Outliers

Interquartile range is narrow and stable

Programmer performance category changes by program.
Caterpillar Plot of Effort Rank for Within and Between

90% confidence in individual rank spans half the sample.
Best Performers Usually Finished in the First Quintile

90% confidence in individual rank spans half the sample.
Weakest Performers Were a little More Variable

90% confidence in individual rank spans half the sample.
Most Individual **Effort Rank** Varied by 50% of the Population

90% confidence in individual rank spans half the sample.
Is there at least an Order of Magnitude range in programmer productivity?
Is there at least an Order of Magnitude range in programmer productivity?

The question either isn’t very precise, or doesn’t mean the same thing to all people.

What do we mean?

Programmers have about a x5 range, but

Most are within about x2

There may be very few much much slower, than average, but

There are not many more than twice average speed

With caveats.

BUT

This is for small programs that should not exceed normal capability.
Other findings

Experience didn’t matter – *I have 15 years of experience, (or maybe 1 year 15 times)*
We had years of experience and amount of code written, but no effect

Highest degree obtained, no effect

College level statistics class actually, no
Summary, Similar within and between productivity variation

1) IntraCluster Correlation 0.6 (40% of variation within developer)

2) **New** Within variation had not previously been reported

3) Wide min/max range remains, but

4) Vast majority of developers are within a much narrower “x5” total productivity range, best and average about “x2”, middle 50% about x2

5) Single point rather than repeated measures
   - Are not reliable (high variation)
   - Exaggerate range

6) Can get whatever Min/Max ratio you want by varying sample size and other assumptions. Min/Max isn’t very useful by itself.

7) Older studies replicated, but chased the noise
Limitations of This Study

This study only used C programmers (15% of the sample).

Programs are not broadly representative.
  • Mostly implement math and numeric
  • Small
  • Simple and well defined

Drop out bias? (common problem among the studies)

Selection bias for entry?

Some “re-use” affects individual program results.

Significant differences in productivity remain.
Implications

This has implications for Industry.

- Max/Min ratio doesn’t tell us much because it’s mostly from the low end. We are measuring outliers not the main effects.
- Hard to get enough “top” programmers to affect a large project performance.
- Don’t over react to short term variation.
  - Look for controllable sources of variation (test, design, reviews).
  - Recognize outlier events for intervention.
  - Short term variation swamps other productivity effects.
- Don’t focus just on productivity! Other factors matter too!, environment, design, tooling, reviews, and quality. These matter, are more controllable, and are trainable. Don’t focus excessively on short term productivity.
Future work

Linear mixed models are suggestive.

Relative Effort had the highest intra class correlation of these!

Look at effects across programming languages.

Test other long held beliefs

How much does programming language matter?

Is Quality free?

Data Link: http://ieee-dataport.org/1783
To Use the PSP and (some TSP) Data

PSP STUDENT ASSIGNMENT DATA

**Permalink:** http://ieee-dataport.org/open-access/psp-student-assignment-data

**DOI Link:** http://dx.doi.org/10.21227/a5vb-cf02

**Short Link:** http://ieee-dataport.org/1783


The PSP and TSP course materials are available on the SEI Digital Library at https://www.sei.cmu.edu/go/tsp for use under the terms of the Creative Commons license

TSP data used in the CESAW report is available as follows:

Acknowledgements

Mike Konrad and SCOPE project for supporting this work
Tim Menzies for his guidance and editing for the IEEE article
PSP Instructors and Students who contributed their data
References


Data Distributions Roughly Log-Normal

- **Size [LoC]**
- **Effort [Minutes]**
- **Defects**

**Student** $(j)$

**Assignment** $(i,j)$