Unreproducible tests

Successes, failures, and lessons in testing and verification

Michael D. Ernst
University of Washington

Presented at ICST
20 April 2012
Reproducibility: The linchpin of verification

A test should behave deterministically

- For detecting failures
- For debugging
- For providing confidence

A proof must be independently verifiable

Tool support: test frameworks, mocking, capture-replay, proof assistants, ...
Reproducibility: The linchpin of research

Research:
  – A search for scientific truth
  – Should be testable (falsifiable) - Karl Popper

Example: evaluation of a tool or methodology

Bad news: Much research in testing and verification fails this scientific standard
Industrial practice is little better

“Variability and reproducibility in software engineering: A study of four companies that developed the same system”, Anda et al., 2008
A personal embarrassment


Indicates bug-prone code
Outperforms competitors; 50x better than random
Solves open problem
Innovative methods
>100 citations
What went wrong

• Tried lots of machine learning techniques
  – Went with the one that worked
  – Output is actionable, but no explanatory power
  – Explanatory models were baffling
• Unable to reproduce
  – Despite availability of source code & experiments
• No malfeasance, but not enough care

How can we prevent such problems?
Outline

• Examples of non-reproducibility
• Causes of non-reproducibility
• Is non-reproducibility a problem?
• Achieving reproducibility
Random vs. systematic test generation

• Random is worse
  [Ferguson 1996, Csallner 2005, ...]

• Random is better
  [Dickinson 2001, Pacheco 2009]

• Mixed
  [Hamlet 1990, D’Amorim 2006, Pacheco 2007, Qu 2008]
Test coverage

• Test-driven development improves outcomes [Franz 94, George 2004]
• Unit testing ROI is 245%-1066% [IPL 2004]
• Abandoned in practice [Robinson 2011]
Type systems

• Static typing is better
  – the Haskell crowd

• Dynamic typing is better
  – [Hanenburg 2010]
  – the PHP/Python/JavaScript/Ruby crowd

• Many attempts to combine them
  – Soft typing, inference
  – Gradual/hybrid typing
Programming styles

• Introductory programming classes:
  – Objects first [Kolling 2001, Decker 2003, ...]
  – Objects later [Reges 2006, ...]
  – Makes no difference [Ehlert 2009, Schulte 2010, ...]

• Object-oriented programming

• Functional languages
  – Yahoo! Store originally in Lisp
  – Facebook chat widget originally in Erlang
More examples

• Formal methods from the beginning [Barnes 1997]
• Extreme programming [Beck 1999]
• Testing methodologies
Causes of non-reproducibility

1. Some other factor dominates the experimental effect

Threats to validity
• construct (correct measurements & statistics)
• internal (alternative explanations & confounds)
• external (generalize beyond subjects)
• reliability (reproduce)
We can learn a lot even from studies of college students

People

• Abilities
• Knowledge
• Motivation
Other experimental subjects
(besides people)

- “Subsetting the SPEC CPU2006 benchmark suite” [Phansalkar 2007]
- “Experiments with subsetting benchmark suites” [Vandierendonck 2005]
- “The use and abuse of SPEC” [Hennessey 2003]
Implementation

• Every evaluation is of an implementation
  – Tool, instantiation of a process such as XP or TDD, etc.
  – You hope it generalizes to a technique

• Your tool
  – Tuned to specific problems or programs

• Competing tool
  – Strawman implementation
    • Example: random testing
  – Tool is mismatched to the task
    • Example: clone detection [ICSE 2012]
  – Configuration/setup
    • Example: invariant detection
Interpretation of results

• Improper/missing statistical analysis
• Statistical flukes
  – needs to have an explanation
  – tried too many things
• Subjective bias
Biases

- Hawthorne effect (observer effect)
- Friendly users, underestimate effort
- Sloppiness
- Fraud
  - (Compare to sloppiness)
Reasons not to totemize reproducibility

Reproducibility is not always paramount
Reproducibility inhibits innovation

• Reproducibility adds cost
  – Small increment for any project

• Don’t over-engineer
  – If it’s not tested, it is not correct
  – Are your results important enough to be correct?

• Expectation of reproducibility affects research
  – Reproducibility is a good way to get your paper accepted
Our field is young

• It takes **decades** to transition from research to practice
  – True but irrelevant
• Lessons and generalizations will **appear in time**
  – **How** will they appear?
  – Do we want them to appear **faster**?
• The field is still developing & learning
  – Statistics? Study design?
A novel idea is worthy of dissemination...

... without evaluation

... without artifacts

Possibly true, but irrelevant

“Results, not ideas.”
-Craig Chambers
Positive deviance

• A difference in outcomes indicates:
  – an important factor
  – a too-general question

• Celebrate differences and seek lessons in them
  – Yes, but start understanding earlier
How to achieve reproducibility
Definitions

• **Reproducible**: an independent party can
  – follow the same steps, and
  – obtain similar results

• **Generalizable**: similar results, in a different context

• **Credible**: the audience believes the results
Give all the details

• Goal: a master's student can reproduce the results
  – Open-source tools and data
  – Use the Web or a TR as appropriate
• Takes extra work
  – Choice: science vs. extra publications vs. secrecy
• Don’t suppress unfavorable data

Unfortunately, license issues prevent the tool in its current form from being released as open source.
Admit non-generalizability

• You cannot to control for every factor
• What do you expect to generalize?
• **Why?**
• Did you try it?
  – Did you test your hypothesis?
“Threats to validity” section considered dangerous

“Our experiments use a suite of 7 programs and may not generalize to other programs.”

Often omits the real threats – cargo-cult science
It's better to discuss as you go along
Summarize in conclusions
Explain yourself

• No “I did it” research
• Explain each result/effect
  – or admit you don’t know
• What was hard or unexpected?
• Why didn’t others do this before?

• Make your conclusions actionable
Research papers are software too

• “If it isn’t tested, it’s probably broken.”

• Have you tested your code?
• Have you tested generalizability?

• Act like your results matter
Automate/script everything

There should be no manual steps (Excel, etc.)
- Except during exploratory analysis
- Prevents mistakes
- Enables replication
- Good if data changes

This costs no extra time in the long run
(Do you believe that? Why?)
Packaging a virtual machine

- Reproducibility, but not generalizability
- Hard to combine two such tools
- Partial credit
Measure and compare

• Actually measure
  – Compare to other work
  – Reuse data where possible
• Report statistical results, not just averages
• Explain differences

Look for measureable and repeatable effects
  – 1% programmer productivity would matter!
  – It won't be visible
Focus

• Don't bury the reader in details
• Don't report irrelevant measures
• Not every question needs to be answered
• Not every question needs to be answered numerically
Usability

• Is your setup only usable by the authors?
• Do you want others to extend the work?
• Pros and cons of realistic engineering
  – Engineering effort
  – Learning from users
  – Re-use (citations)
Reproducibility, not reproduction

• Not every research result must be reproduced
• All results should be reproducible

• Your research answers some specific (small) question
• Seek reproducibility in that context
Blur the lines

• Researchers should be practitioners
  – design, write, read, and test code!
  – and more besides, of course

• Practitioners should be open to new ways of working
  – Settling for “best practices” is settling for mediocrity
We are doing a great job

Research in testing and verification:
• Thriving research community
• Influence beyond this community
• Great ideas
• Practical tools
• Much good evaluation
• Transformed industry
• Helped society

We can do better
“If I have seen further it is by standing on the shoulders of giants.”

-Isaac Newton