Data Analytics in Software Engineering

Christian Kaestner
Learning Goals

• Understand importance of data-driven decision making also during software engineering
• Collect and analyze measurements
• Design evaluation strategies to evaluate the effectiveness of interventions
• Understand the potential of data analytics at scale for QA data
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
- barley water
- oranges

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?

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

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

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

Do cross-cutting concerns cause defects?

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

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

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

Should I be writing unit tests in my software project?

Is strong code ownership good or bad for software quality?

Does Distributed/Global software development affect quality?
How would you approach these questions with data?

- Where to focus testing effort?
- Is our review practice effective?
- Is the expensive static analysis tool paying off?
- Should we invest in security training?
Believes vs Evidence?

• “40% of major decisions are based not on facts, but on the manager’s gut” [Accenture survey among 254 US managers in industry]

• E.g., strong believes in survey among 564 Microsoft engineers
  • Code Reviews improve code quality
  • Coding Standards improve code quality
  • Static Analysis tools improve code quality

• Controversial believes from same survey
  • Code Quality depends on programming language
  • Fixing Defects is riskier than adding new features
  • Geographically distributed teams produce code of as good quality as non-distributed teams.

Source of Believes
Software Engineering is becoming more like modern medicine?
Measurement and Metrics

• Discussed throughout the semester
• Everything is measurable
• Define measures, be critical (precision, accuracy, ...)
• Be systematic in data collection (prefer automation)
How would you approach these questions with data?

• Where to focus testing effort?
• Is our review practice effective?
• Is the expensive static analysis tool paying off?
• Should we invest in security training?
Evaluate Effectiveness of an Intervention

• Controlled experiments
  • Compare group with intervention against control group without,
  • Randomized controlled trials, AB testing, ...
  • Ideally blinded

• Natural experiments, Quasi experiments
  • Compare similar groups that naturally only differ in the intervention
  • No randomized assignment of treatment condition

• Time series analyses
  • Compare measures before and after intervention, preferably across groups with the intervention at different times
On Experiments

• Understand experimental methods and limitations
  • Chose appropriate design (e.g., quasi experiment, vs timeseries, vs controlled)
  • Appropriate to research question and available subjects

• Design carefully, control confounds, avoid biases
• Use appropriate statistics to draw conclusions

• This requires sound understanding of quantitative research methods
• Many pitfalls
data science / analytics 101
Use of data, analysis, and systematic reasoning to [inform and] make decisions
the many names
software intelligence
software analytics
software development analytics
analytics for software development
empirical software engineering
mining software repositories
Data Scientist: The Sexiest Job of the 21st Century

by Thomas H. Davenport and D.J. Patil

When Jonathan Goldman arrived for work in June 2006 at LinkedIn, the business networking site, the place still felt like a start-up. The company had just under 8 million accounts, and the number was growing quickly as existing members invited their friends and colleagues to join. But users weren’t seeking out connections with the people who were already on the site at the rate executives had expected. Something was apparently missing in the social experience. As one LinkedIn manager put it, “It was like arriving at a conference reception and finding no knapsack to put your coffee and sandwiches in.”
Typical data science workflow

- Acquire data
  - Reformat and clean data
- Preparation
  - Explore alternatives
- Analysis
  - Edit analysis scripts
  - Debug
  - Execute scripts
  - Inspect outputs
- Dissemination
  - Make comparisons
  - Take notes
  - Hold meetings
  - Write reports
  - Deploy online
  - Archive experiment
  - Share experiment
- Reflection

Background of Data Scientists

Most CS, many interdisciplinary backgrounds
Many have higher education degrees
Strong passion for data

I love data, looking and making sense of the data. [P2]

I’ve always been a data kind of guy. I love playing with data. I’m very focused on how you can organize and make sense of data and being able to find patterns. I love patterns. [P14]

“Machine learning hackers”. Need to know stats

My people have to know statistics. They need to be able to answer sample size questions, design experiment questions, know standard deviations, p-value, confidence intervals, etc.
Abundance of Data
Abundance of Data

• Code history
• Developer activities
• Bug trackers
• Sprint backlog, milestones
• Continuous integration logs
• Static analysis and technical debt dashboards
• Test traces; dynamic analyses
• Runtime traces

• Crash reports from customers
• Server load, stats
• Customer data, interactions
• Support requests, customer reviews
• Working hours
• Team interactions in Slack/issue tracker/email/...
• ...

23
Measurement is Hard
Example: Performance
Twitter Case Study
Timer Overhead

• Measurement itself consumes time

Request time | Even starts | Event ends, request time

Time reported | Saved end time

Memory access and interaction with operating system | Measured event should be 100-1000x longer than measurement overhead
Confounding variables
Confounding variables

- Background processes
- Hardware differences
- Temperature differences
- Input data; random?
- Heap size
- System interrupts
- Single vs multi core systems
- Garbage collection
- Memory layout
- ...
Handling confounding variables

• Keep constant
• Randomize
  • -> Repeated measurements
  • -> Large, diverse benchmarks
• Measure and compute influence ex-post
Common approach: best result

- Repeat measurement
- Report best result (or second best, or worst)
Common approach: Mean values

• Repeat measurement (how often?)
• Report average

• Basic assumptions: Law of large numbers and central limit theorem
Means

- Arithmetic mean

\[ x_{arithm} = \frac{1}{n} \sum_{i=1}^{n} x_i = \frac{x_1 + x_2 + \ldots + x_n}{n} \]

- Median: The value in the middle
  - On even data sets, the arithmetic mean between the two values in the middle
  - Robust against outliers

- Truncated mean
  - Remove 10% outliers (on both ends), then arithm. mean

- Geometric mean

- ...
Median

- Median instead of arithmetic mean, if
  - ordinal data ("distance" has no meaning)
  - only few measurements
  - asymmetric distributions
  - expecting outliers
But

• How many measurements?
  • Are 3, 10, or 50 sufficient? Or 100 or 10000?
  • (to find the higgs boson, several million measurements were necessary)

• Measuring order?
  • AAABBB or ABABAB

• Iterate in a single batch or multiple batches?

• Are measurements independent?

• Is the average good enough?
Visualize data

• Get an overview
• Visually inspect distribution and outliers
 histograms
Reporting distributions

• Boxplot show
  • Median as thick line
  • Quartiles as box (50% of all values are in the box)
  • Whiskers
  • Outliers as dots

• Cumulative probability distributions
  plot(ecdf(c))

• Visual representation of distributions
  boxplot(c)
Error Models and Probability Distributions
Intuition: Error Model

• 1 random error, influence +/- 1
  • Real mean: 10
  • Measurements: 9 (50%) and 11 (50%)

• 2 random errors, each +/- 1
  • Measurements: 8 (25%), 10 (50%) and 12 (25%)

• 3 random errors, each +/- 1
  • Measurements: 7 (12.5%), 9 (37.5), 11 (37.5), 12 (12.5)
Normal distributions
Standard deviation

\[ s = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2} = \sqrt{\frac{(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \ldots + (x_n - \bar{x})^2}{n}} \]
Confidence intervals (formal)

\[
\left[ \bar{x} - z_{(1-\alpha/2)} \frac{\sigma}{\sqrt{n}} ; \bar{x} + z_{(1-\alpha/2)} \frac{\sigma}{\sqrt{n}} \right]
\]

\[
\left[ \bar{x} - t_{(1-\alpha/2; n-1)} \frac{s}{\sqrt{n}} ; \bar{x} + t_{(1-\alpha/2; n-1)} \frac{s}{\sqrt{n}} \right]
\]
Confidence intervals

Collect data until confidence interval at an expected size, e.g., +/- 10%
Confidence intervals

• Results of independent measurements are normally distributed (central limit theorem)

• Confidence level 95% => with 95% probability, the real mean is within the interval*
  • Mean of the measurements vs real mean of the statistical population

*Technically more correct: When repeating the experiment very often, in 95% of the repetitions the real mean will be within the confidence interval of that measurement
Accuracy vs Precision

**Accuracy:**
Deviations of the measured mean from the real mean
i.e., can we trust the results

**Precision:**
Distribution around the mean (repeatability)
Source of measurement error, usually not attributable

**Resolution:**
smallest measureable difference
Random vs. Systematic Errors

• Systematic errors: Error of experimental design or measurement technique
  • CPU Speed: Measuring at different temperatures
  • Forgot to reset counter for repeated measurement
  • -> Small variance over repeated measurements
  • -> Experience to exclude them during design
  • -> Accuracy

• Random errors
  • Cannot be controlled
  • Stochastic methods
  • -> Precision
Comparing Measurements
Comparing measurement results

- GenCopy faster than GenMS?
- GenCopy faster than SemiSpace?
Comparing Distributions
Different effect size, same deviations

small overlap
=> significant difference

large overlap
=> no significant difference
Same effect size, different deviations

small overlap
=> significant difference

large overlap
=> no significant difference
Dependent vs. independent measurements

• Pairwise (dependent) measurements
  • Before/after comparison
  • With same benchmark + environment
  • e.g., new operating system/disc drive faster

• Independent measurements
  • Repeated measurements
  • Input data regenerated for each measurement
Significance level

• Statistical change of an error
• Define before executing the experiment
  • use commonly accepted values
  • based on cost of a wrong decision
• Common:
  • 0.05 significant
  • 0.01 very significant

• Statistically significant result =!> proof
• Statistically significant result =!> important result
• Covers only alpha error (more later)
Compare confidence interval

• Rule of thumb: If the confidence intervals do not overlap, the difference is significant
t test

• Requires: normally distributed metric data
  • very large data sets almost always follow a normal distribution

• Compares to measurement

• Basic idea:
  • Assume that both measurements are from the same basis population (follow the same distribution)
  • t test computes the chance that both samples are from the same distribution
  • If probability is smaller than 5% (for significance level 0.05) the assumption is considered refuted
t test with R

> t.test(x, y, conf.level=0.9)

Welch Two Sample t-test

data:  x and y
t = 1.9988, df = 95.801, p-value = 0.04846
alternative hypothesis: true difference in means is not equal to 0
90 percent confidence interval:
  0.3464147  3.7520619
sample estimates:
mean of x  mean of y
 51.42307   49.37383

> t.test(x-y, conf.level=0.9) (paired)
• For causation
  • Provide a theory (from domain knowledge, independent of data)
  • Show correlation
  • Demonstrate ability to predict new cases (replicate/validate)

http://xkcd.com/552/
Big Code Data Science
Abundance of Data

- Code history
- Developer activities
- Bug trackers
- Sprint backlog, milestones
- Continuous integration logs
- Static analysis and technical debt dashboards
- Test traces; dynamic analyses
- Runtime traces

- Crash reports from customers
- Server load, stats
- Customer data, interactions
- Support requests, customer reviews
- Working hours
- Team interactions in Slack/issue tracker/email/...
- ...

...
Large Datasets now accessible

• Huge codebases in Google, Facebook, Microsoft, ...
• Public activates of open source projects, including hobby projects and industrial systems (e.g., GitHub
  • 27M contributors, 80M projects, 1B traces, 10 years
• Lots of data: Code, commits, commit messages, issues, bug-fixing patches, discussions, reviews, pull requests, teams, build logs, static analysis logs, coverage history, performance history
• Lots of noise: Multitasking, interruptions, offline communication, project and team cultures, ...
Data Science on Big Code

• Answer large, more general questions:
  • What team size is most productive or produces highest quality?
  • Is multitasking causing buggy code?
  • Do co-located teams perform better?
  • Does code review improve quality?

• Find trends in big noisy data sets using advanced statistics
• Find even small relationships with natural experiments: Compare similar projects that differ only in one aspect (given the size, there will be many pairs for most questions)
Example Results

• “Geographically distributed teams produce code whose quality (defect occurrence) is just as good as teams that are not geographically distributed”
  • No statistical difference detected at Microsoft

• “Defect probability increases if teams consist of members with large organizational distance”
  • Key predictor for defect density found at Microsoft

• “Multitaskers are more productive in open source projects, but not beyond 5 projects”
  • Confirmed on GitHub data by CMU Faculty Vasilescu
Example: Badges

<table>
<thead>
<tr>
<th>Basic Model</th>
<th>Full Model</th>
<th>RDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>response: freshness = 0</td>
<td>response: freshness = 0</td>
<td>response: log(freshness)</td>
</tr>
<tr>
<td>17.3% deviance explained</td>
<td>17.4% deviance explained</td>
<td>$R^2_m = 0.04, R^2_c = 0.35$</td>
</tr>
<tr>
<td>Coeffs (Err.)</td>
<td>LR ChiSq</td>
<td>Coeffs (Err.)</td>
</tr>
<tr>
<td>(Inter.) 3.54 (0.03)**</td>
<td>3.50 (0.03)**</td>
<td>1.45 (0.09)**</td>
</tr>
<tr>
<td>Dep. -1.78 (0.01)** 32077.8***</td>
<td>-1.79 (0.01)** 32292.8***</td>
<td>-0.04 (0.02)</td>
</tr>
<tr>
<td>RDep. 0.22 (0.01)** 610.3***</td>
<td>0.21 (0.01)** 560.6***</td>
<td>-0.01 (0.02)</td>
</tr>
<tr>
<td>Stars -0.08 (0.00)** 301.4***</td>
<td>-0.09 (0.00)** 311.2***</td>
<td>0.00 (0.01)</td>
</tr>
<tr>
<td>Contr. -0.24 (0.01)** 500.5***</td>
<td>-0.25 (0.01)** 548.7***</td>
<td>-0.04 (0.02)</td>
</tr>
<tr>
<td>lastU -0.65 (0.01)** 12080.9***</td>
<td>-0.64 (0.01)** 11537.9***</td>
<td>0.01 (0.02)</td>
</tr>
<tr>
<td>hasDM 0.24 (0.03)** 116.1***</td>
<td>0.45 (0.08)** 2.43</td>
<td></td>
</tr>
<tr>
<td>hasInf 0.11 (0.02)** 48.3***</td>
<td>0.04 (0.05)</td>
<td>0.45</td>
</tr>
<tr>
<td>hasDM:hasInf -0.05 (0.04) 1.9</td>
<td>-0.32 (0.10)**</td>
<td>0.03 (0.00)** 82.99***</td>
</tr>
<tr>
<td>hasOther 0.01 (0.01)</td>
<td>-0.93 (0.03)** 1373.22***</td>
<td></td>
</tr>
<tr>
<td>time 0.11 (0.00)</td>
<td>455.56***</td>
<td></td>
</tr>
<tr>
<td>time_after_intervention 0.10 (0.01)** 230.36***</td>
<td></td>
<td></td>
</tr>
<tr>
<td>time_after_intervention:hasDM -0.00 (0.01) 1.14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>time_after_intervention:hasInf 0.03 (0.01)** 10.62**</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$; 
Dep: dependencies; RDep: dependents; Contr: contributors; lastU: time since last update; 
hasDM: has dependency-manager badge; hasInf: has information badge; hasOther: adopts

Experimenting in Production
Canary Testing and AB Testing
Testing in Production

• Beta tests
• AB tests
• Tests across hardware/software diversity (e.g., Android)
• “Most updates are unproblematic”
• “Testing under real conditions, with real workloads”
• Avoid expensive redundant test infrastructure
Pipelines

Continuous Delivery

Unit Test  Platform Test  Deliver to Staging  Application Acceptance tests  Deploy to Production  Post deploy tests
Auto  Auto  Auto  Auto  Manual  Auto

Continuous Deployment

Unit Test  Platform Test  Deliver to Staging  Application Acceptance tests  Deploy to Production  Post deploy tests
Auto  Auto  Auto  Auto  Auto  Auto
Release cycle of Facebook’s apps
Real DevOps Pipelines are Complex

- Incremental rollout, reconfiguring routers
- Canary testing
- Automatic rolling back changes

Configuration management, Infrastructure as Code

• Scripts to change system configurations (configuration files, install packages, versions, ...); declarative vs imperative

• Usually put under version control

```
- hosts: all
  sudo: yes
  tasks:
    - apt: name={{ item }}
      with_items:
        - ldap-auth-client
        - nscd
    - shell: auth-client-config -t nss -p lac_ldap
    - copy: src=ldap/my_mkhomedir dest=... 
    - copy: src=ldap/ldap.conf dest=/etc/ldap.conf
    - shell: pam-auth-update --package
    - shell: /etc/init.d/nscd restart
```

```python
$nameservers = ['10.0.2.3']
file { '/etc/resolv.conf':
  ensure => file,
  owner => 'root',
  group => 'root',
  mode => '0644',
  content => template('resolver/resolv.conf.erb'),
}
```
Monitoring

• Many standard and custom tools for monitoring, aggregation and reporting

• Logging infrastructure at scale

• Open source examples
  • collectd/collect for gathering and storing statistics
  • Monit checks whether process is running
  • Nagios monitoring infrastructure, highly extensible
Why DevOps when testing in production

• Ability to quickly change configurations for different users
• Track configuration changes
• Track metrics at runtime in production system
• Track results per configuration; analysis dashboard to test effects
• Induce realistic fault scenarios (ChaosMonkey...)
• Ability to roll back bad changes quickly
From the authors of *The Visible Ops Handbook*

**The Phoenix Project**

A Novel About IT, DevOps, and Helping Your Business Win

Gene Kim, Kevin Behr, and George Spafford
Summary

• Pursue data-supported decisions, rather than relying on “belief”
• Learn from scientific methods, experiments, statistics
  • Experimental designs
  • Biases, confounding variables
  • Measurements, systematic vs random errors
• Big code provides new opportunities
• Measurement in production with DevOps
• Measurement is essential for software engineering professionals
Some slides with input from

• Bogdan Vasilescu, ISR/CMU
• Thomas Zimmermann, Microsoft Research:
  • https://speakerdeck.com/tomzimmermann
• Greg Wilson, Mozilla
  • https://www.slideshare.net/gvwilson/presentations