Lecture 2: Metrics and Measurement

17-313: Foundations of Software Engineering

Rohan Padhye and Michael Hilton

Administrivia

- Slack
  - Please add a profile picture.
  - Ask questions in #general or #technicalsupport
- Homework 1 is released. It is due Thu Sept 9, 11:59 pm (one week!)
  - This is an individual assignment; we will compose groups this week.
  - Get started early, ask for help, and check the #technicalsupport channel; chances are decent your questions have been asked by others! Office hours will be scheduled.
- Reading for next Tuesday will be posted shortly.
- If you haven't filled out the schedule survey, do so after class.

Learning Goals

- Use measurements as a decision tool to reduce uncertainty
- Understand difficulty of measurement; discuss validity of measurements
- Provide examples of metrics for software qualities and process
- Understand limitations and dangers of decisions and incentives based on measurements

Software Engineering: Principles, practices (technical and non-technical) for **confidently** building **high-quality** software.

What does this mean?
How do we know?
→ *Measurement* and metrics are **key** concerns.

CASE STUDY: AUTONOMOUS VEHICLE SAFETY

How can we judge the quality of AV software?

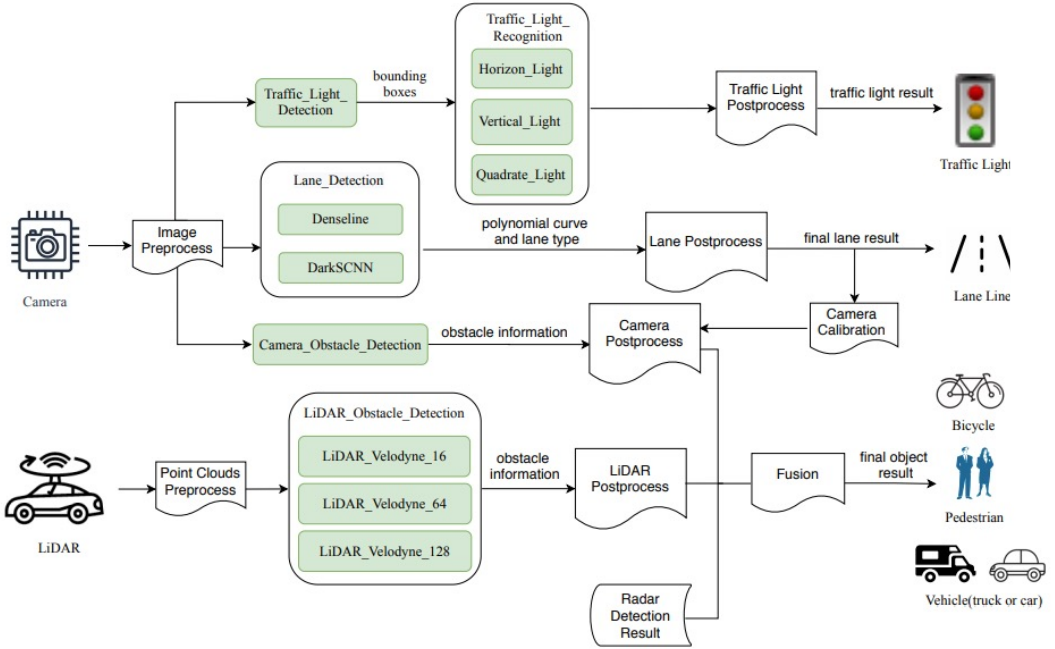
Test coverage

- Amount of code executed during testing.
- Statement coverage, line coverage, branch coverage, etc.
- E.g. 75% branch coverage → 3/4 if-else outcomes have been executed

```
        :          :
        :     1698 : const TrajectoryPoint& StGraphData::init_point() const { return init_point_; }
        :          :
        :     2264 : const SpeedLimit& StGraphData::speed_limit() const { return speed_limit_; }
        :          :
        :   212736 : double StGraphData::cruise_speed() const {
[ - + ]:   212736 :   return cruise_speed_ > 0.0 ? cruise_speed_ : FLAGS_default_cruise_speed;
        :          : }
        :          :
        :     1698 : double StGraphData::path_length() const { return path_data_length_; }
        :          :
        :     1698 : double StGraphData::total_time_by_conf() const { return total_time_by_conf_; }
        :          :
        :     1698 : planning_internal::STGraphDebug* StGraphData::mutable_st_graph_debug() {
        :     1698 :   return st_graph_debug_;
        :          : }
        :          :
        :      566 : bool StGraphData::SetSTDrivableBoundary(
        :          :     const std::vector<std::tuple<double, double, double>>& s_boundary,
        :          :     const std::vector<std::tuple<double, double, double>>& v_obs_info) {
[ + - ]:      566 :   if (s_boundary.size() != v_obs_info.size()) {
        :          :     return false;
        :          :   }
[ + + ]:    40752 :   for (size_t i = 0; i < s_boundary.size(); ++i) {
        :    80372 :     auto st_bound_instance = st_drivable_boundary_.add_st_boundary();
        :   160744 :     st_bound_instance->set_t(std::get<0>(s_boundary[i]));
        :   120558 :     st_bound_instance->set_s_lower(std::get<1>(s_boundary[i]));
        :   120558 :     st_bound_instance->set_s_upper(std::get<2>(s_boundary[i]));
[ - + ]:    40186 :     if (std::get<1>(v_obs_info[i]) > -kObsSpeedIgnoreThreshold) {
        :        0 :       st_bound_instance->set_v_obs_lower(std::get<1>(v_obs_info[i]));
        :          :     }
[ + + ]:    40186 :     if (std::get<2>(v_obs_info[i]) < kObsSpeedIgnoreThreshold) {
        :    50254 :       st_bound_instance->set_v_obs_upper(std::get<2>(v_obs_info[i]));
        :          :     }
        :          :   }
```

Model Accuracy

- Train machine-learning models on labelled data (sensor data + ground truth).
- Compute accuracy on a separate labelled test set
- E.g. 90% accuracy implies that object recognition is right for 90% of the test inputs.



Source: Peng et al. ESEC/FSE'20

Failure Rate

- Frequency of crashes/fatalities
- Per 1000 rides, per million miles, per month (in the news)

Mileage

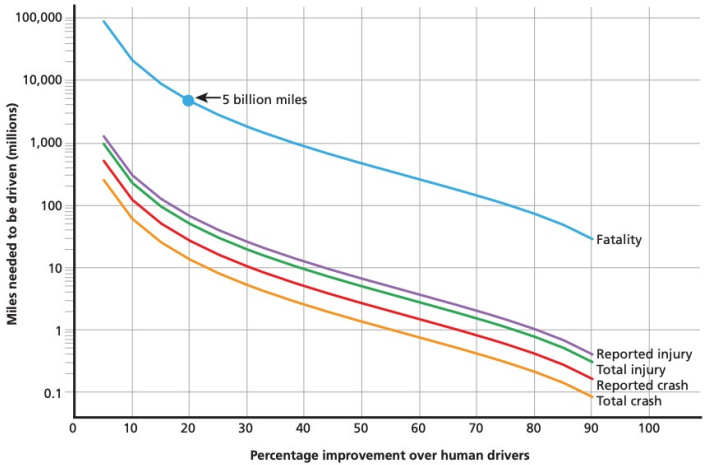
RAND CORPORATION

Driving to Safety

How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability?

Nidhi Kalra, Susan M. Paddock

Figure 3. Miles Needed to Demonstrate with 95% Confidence that the Autonomous Vehicle Failure Rate Is Lower than the Human Driver Failure Rate



Source: waymo.com/safety (September 2021)

Activity

Think of “pros” and “cons” for using various quality metrics to judge AV software.

- Test coverage
- Model accuracy
- Failure rate
- Mileage
- Size of codebase
- Age of codebase
- Time of most recent change
- Frequency of code releases
- Number of contributors
- Amount of code documentation

MEASUREMENT FOR DECISION MAKING IN SOFTWARE DEVELOPMENT

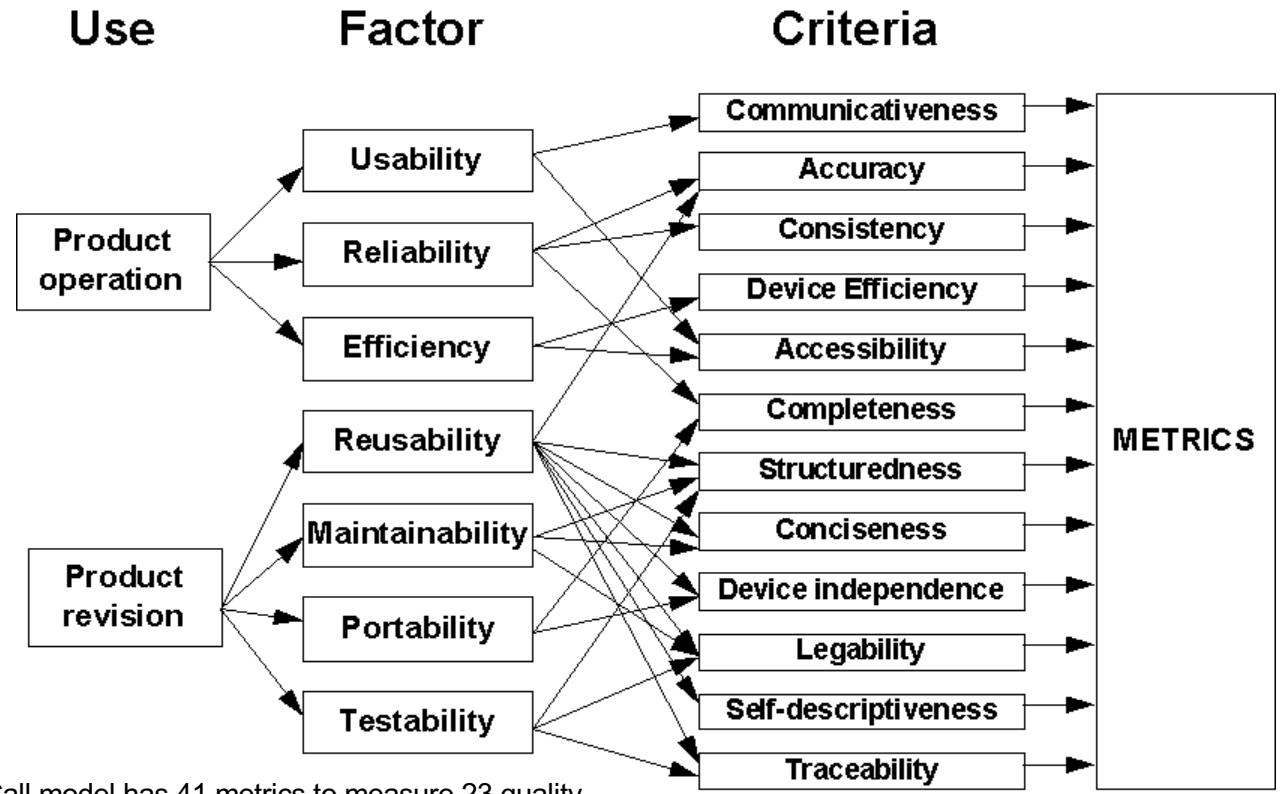
What is Measurement?

- Measurement is the empirical, objective assignment of numbers, according to a rule derived from a model or theory, to attributes of objects or events with the intent of describing them. – Craner, Bond, “Software Engineering Metrics: What Do They Measure and How Do We Know?”
- A quantitatively expressed reduction of uncertainty based on one or more observations. – Hubbard, “How to Measure Anything …”

Software Quality Metrics

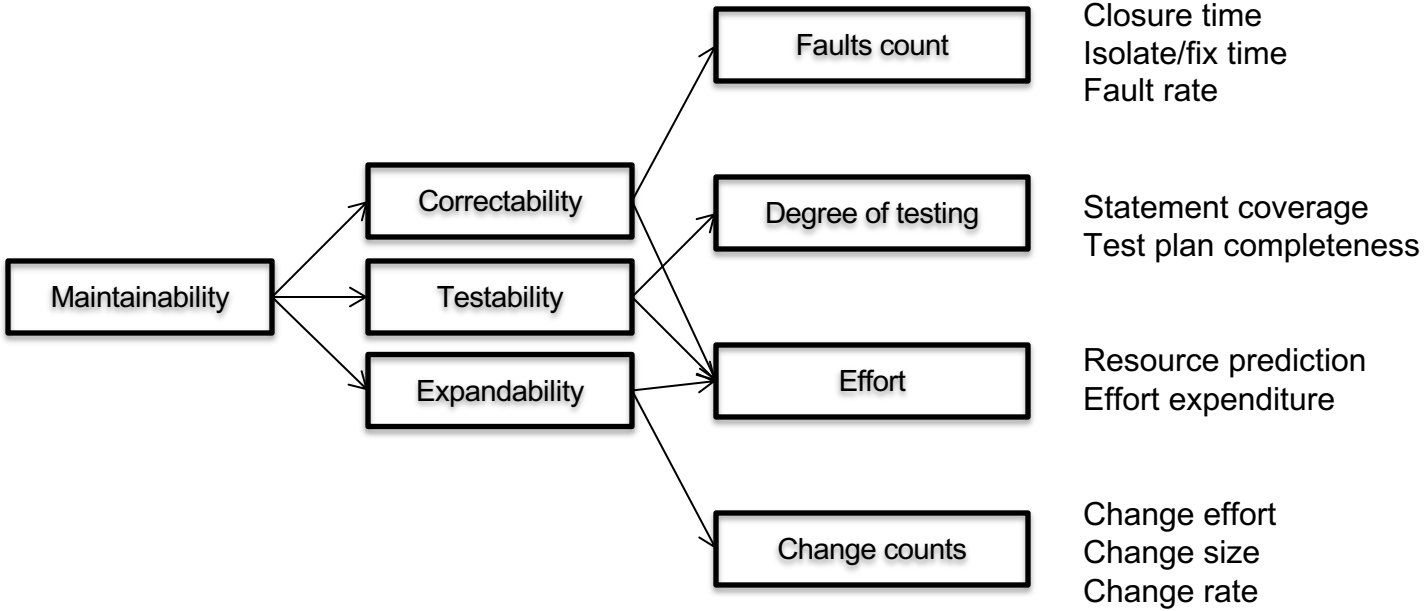
- IEEE 1061 definition: “A software quality metric is a function whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which the software possesses a given attribute that affects its quality.”
- Metrics have been proposed for many quality attributes; may define own metrics

External attributes: Measuring Quality



McCall model has 41 metrics to measure 23 quality criteria from 11 factors

Decomposition of Metrics



EXAMPLES:
CODE COMPLEXITY

Lines of Code

- Easy to measure

```
> wc –l file1 file2…
```

LOC	projects
450	Expression Evaluator
2,000	Sudoku
100,000	Apache Maven
500,000	Git
3,000,000	MySQL
15,000,000	gcc
50.000.000	Windows 10
2,000,000,000	Google (MonoRepo)

Normalizing Lines of Code

- Ignore comments and empty lines
- Ignore lines < 2 characters
- Pretty print source code first
- Count statements (logical lines of code)
- See also: cloc

```
for (i = 0; i < 100; i += 1) printf("hello"); /* How many lines of code is this? */
```

```
/* How many lines of code is this? */

for (
                    i = 0;
                    i < 100;
                    i += 1
    ) {
                    printf("hello");
}
```

Normalization per Language

Language	Statement factor (productivity)	Line factor
C	1	1
C++	2.5	1
Fortran	2	0.8
Java	2.5	1.5
Perl	6	6
Smalltalk	6	6.25
Python	6	6.5

Source: “Code Complete: A Practical Handbook of Software Construction“, S. McConnell, Microsoft Press (2004) and http://www.codinghorror.com/blog/2005/08/are-all-programming-languages-the-same.html u.a.

Halstead Volume

- Introduced by Maurice Howard Halstead in 1977
- Halstead Volume =
  number of operators/operands *
  log2(number of distinct operators/operands)
- Approximates size of elements and vocabulary

Halstead Volume - Example

```
main() {
        int a, b, c, avg;
        scanf("%d %d %d", &a, &b, &c);
        avg = (a + b + c) / 3;
        printf("avg = %d", avg);
}
```

Operators/Operands: main, (), {}, int, a, b, c, avg, scanf, (), "…", &, a, &, b, &, c, avg, =, a, +, b, +, c, (), /, 3, printf, (), "…", avg

Cyclomatic Complexity

- Proposed by McCabe 1976
- Based on control flow graph, measures linearly independent paths through a program
  - ~= number of decisions
  - Number of test cases needed to achieve branch coverage

M = edges of CFG – nodes of CFG + 2*connected components

```
if (c1) {
                f1();
} else {
                f2();
}
if (c2) {
                f3();
} else {
                f4();
}
```

"For each module, either limit cyclomatic complexity to [X] or provide a written explanation of why the limit was exceeded."
– NIST Structured Testing methodology

Object-Oriented Metrics

- Number of Methods per Class
- Depth of Inheritance Tree
- Number of Child Classes
- Coupling between Object Classes
- Calls to Methods in Unrelated Classes
- …

What software qualities do we care about? (examples)

- Scalability
- Security
- Extensibility
- Documentation
- Performance
- Consistency
- Portability
- Installability
- Maintainability
- Functionality (e.g., data integrity)
- Availability
- Ease of use

What process qualities do we care about? (examples)

- On-time release
- Development speed
- Meeting efficiency
- Conformance to processes
- Time spent on rework
- Reliability of predictions
- Fairness in decision making

- Measure time, costs, actions, resources, and quality of work packages; compare with predictions
- Use information from issue trackers, communication networks, team structures, etc…

Everything is measurable

- If X is something we care about, then X, by definition, must be detectable.
  - How could we care about things like “quality,” “risk,” “security,” or “public image” if these things were totally undetectable, directly or indirectly?
  - If we have reason to care about some unknown quantity, it is because we think it corresponds to desirable or undesirable results in some way.
- If X is detectable, then it must be detectable in some amount.
  - If you can observe a thing at all, you can observe more of it or less of it
- If we can observe it in some amount, then it must be measurable.

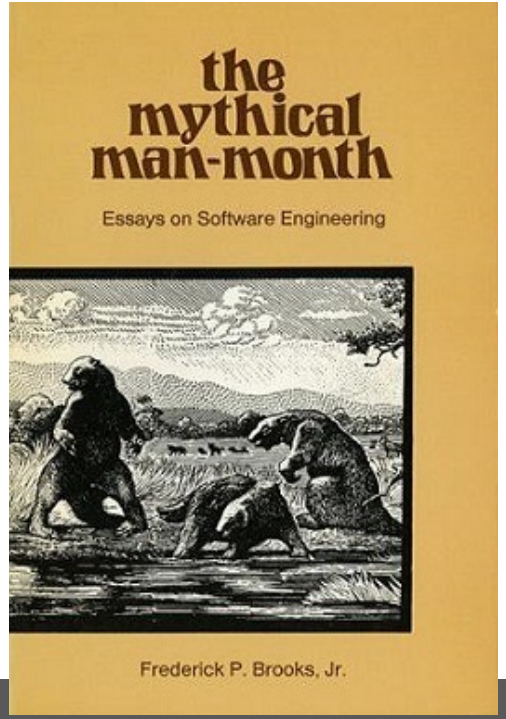
D. Hubbard, How to Measure Anything, 2010

Measurement for Decision Making

- Fund project?
- More testing?
- Fast enough? Secure enough?
- Code quality sufficient?
- Which feature to focus on?
- Developer bonus?
- Time and cost estimation? Predictions reliable?

Example: Antipattern in effort estimation

- IBM in the 60's: Would account in "person-months"
  e.g. Team of 2 working 3 months = 6 person-months
- LoC ~ Person-months ~ $$$
- Brooks: "*Adding manpower to a late software project makes it later*."



Questions to consider.

- What properties do we care about, and how do we measure it?
- What is being measured? Does it (to what degree) capture the thing you care about? What are its limitations?
- How should it be incorporated into process? Check in gate? Once a month? Etc.
- What are potentially negative side effects or incentives?

MEASUREMENT IS DIFFICULT

The streetlight effect

- A known observational bias.
- People tend to look for something only where it's easiest to do so.
    - If you drop your keys at night, you'll tend to look for it under streetlights.

What could possibly go wrong?

- Bad statistics: A basic misunderstanding of measurement theory and what is being measured.
- Bad decisions: The incorrect use of measurement data, leading to unintended side effects.
- Bad incentives: Disregard for the human factors, or how the cultural change of taking measurements will affect people.

Measurement scales

- Scale: the type of data being measured.
- The scale dictates what sorts of analysis/arithmetic is legitimate or meaningful.
- Your options are:
  - Nominal: categories
  - Ordinal: order, but no magnitude.
  - Interval: order, magnitude, but no zero.
  - Ratio: Order, magnitude, and zero.
  - Absolute: special case of ratio.

Nominal/categorical scale

- Entities classified with respect to a certain attribute. Categories are jointly exhaustive and mutually exclusive.
  - No implied order between categories!
- Categories can be represented by labels or numbers; however, they do not represent a magnitude, arithmetic operation have no meaning.
- Can be compared for identity or distinction, and measurements can be obtained by counting the frequencies in each category. Data can also be aggregated.

Entity	Attribute	Categories
Application	Purpose	E-commerce, CRM, Finance
Application	Language	Java, Python, C++, C#
Fault	Source	assignment, checking, algorithm, function, interface, timing

Ordinal scale

- Ordered categories: maps a measured attribute to an ordered set of values, but no information about the magnitude of the differences between elements.
- Measurements can be represented by labels or numbers, BUT: if numbers are used, *they do not represent a magnitude.*
    - Honestly, try not to do that. It eliminates temptation.
- You cannot: add, subtract, perform averages, etc (arithmetic operations are out).
- You can: compare with operators (like “less than” or “greater than”), create ranks for the purposes of rank correlations (Spearman’s coefficient, Kendall’s τ).

Entity	Attribute	Values
Application	Complexity	Very Low, Low, Average, High, Very High
Fault	Severity	1 – Cosmetic, 2 – Moderate, 3 – Major, 4 – Critical

Interval scale

- Has order (like ordinal scale) and magnitude.
  - The intervals between two consecutive integers represent equal amounts of the attribute being measured.
- Does NOT have a zero: 0 is an arbitrary point, and doesn't correspond to the absence of a quantity.
- Most arithmetic (addition, subtraction) is OK, as are mean and dispersion measurements, as are Pearson correlations. Ratios are not meaningful.
  - Ex: The temperature yesterday was 64 F, and today is 32 F. Is today twice as cold as yesterday?
- Incremental variables (quantity as of today – quantity at an earlier time) and preferences are commonly measured in interval scales.

Ratio scale

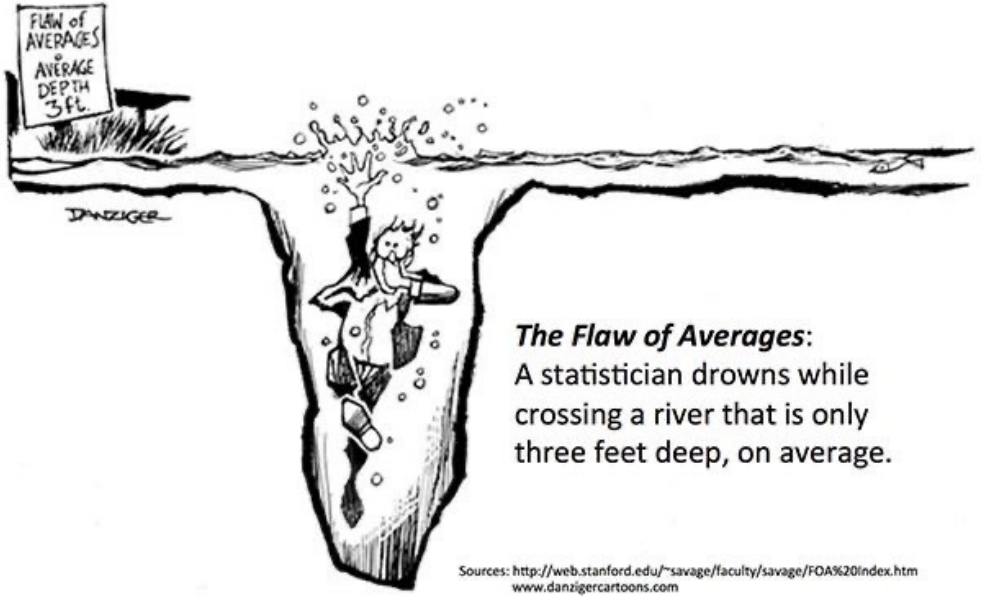
- An interval scale that has a true zero that actually represents the absence of the quantity being measured.
- All arithmetic is meaningful.
- Absolute scale is a special case, measurement simply made by counting the number of elements in the object.
    - Takes the form “number of occurrences of X in the entity.”

Entity	Attribute	Values
Project	Effort	Real numbers
Software	Complexity	Cyclomatic complexity

Summary of scales

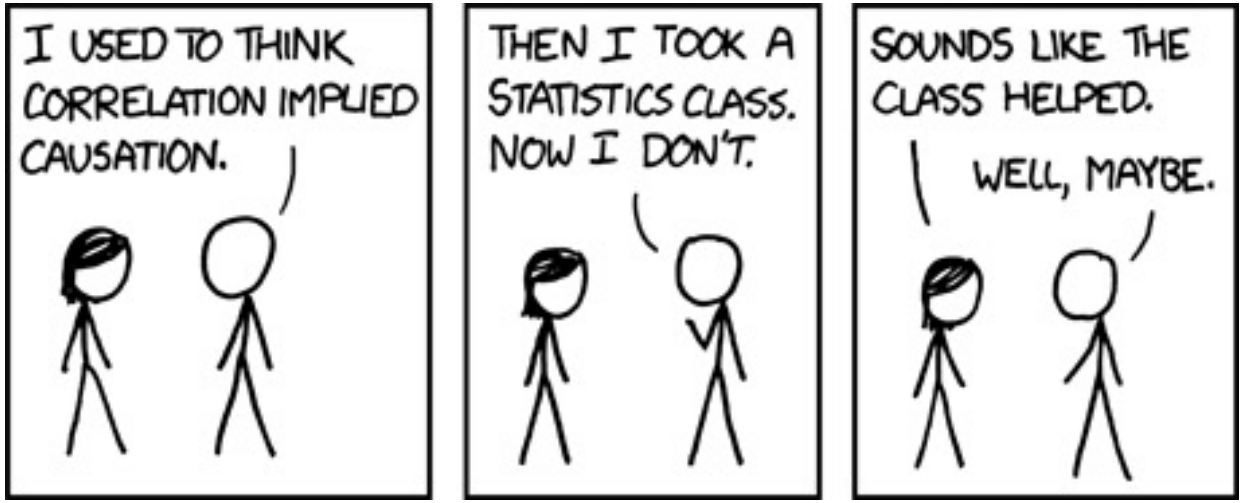
Scale level	Examples	Operators	Possible analyses
Quantitative scales			
Ratio	size, time, cost	$*, /, \log, \sqrt{}$	geometric mean, coefficient of variation
Interval	temperature, marks, judgement expressed on rating scales	$+, -$	mean, variance, correlation, linear regression, analysis of variance (ANOVA), ...
Qualitative scales			
Ordinal	complexity classes	$<, >$	median, rank correlation, ordinal regression
Nominal	feature availability	$=, \neq$	frequencies, mode, contingency tables

UNDERSTAND YOUR DATA



The Flaw of Averages:
A statistician drowns while crossing a river that is only three feet deep, on average.

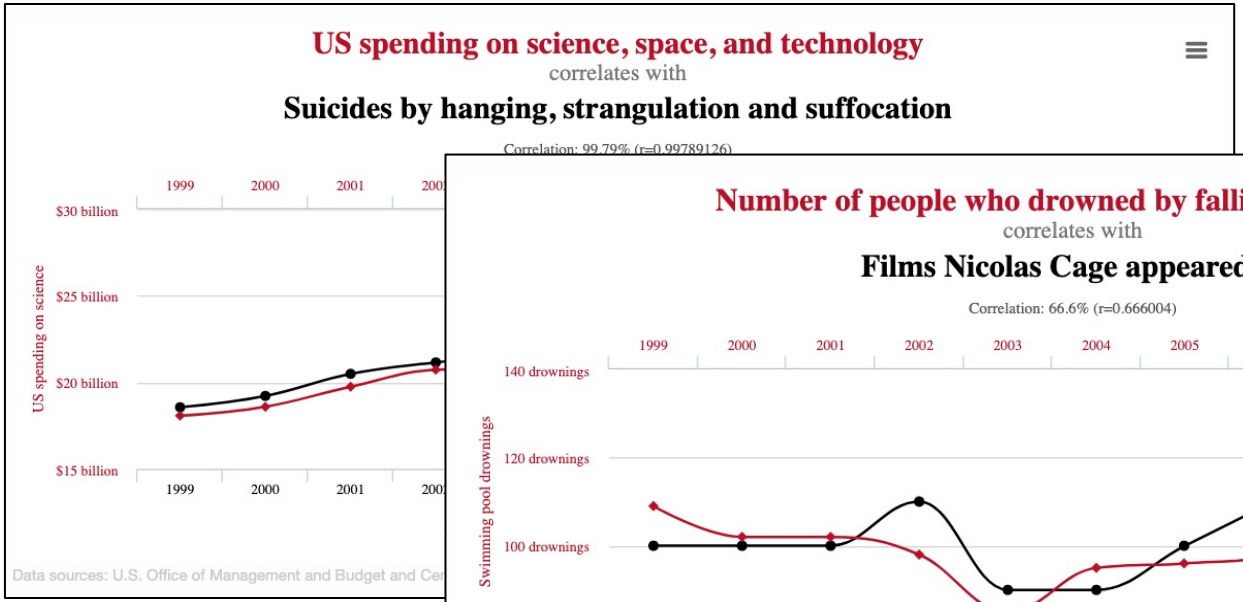
Sources: http://web.stanford.edu/~savage/faculty/savage/FOA%20Index.htm
www.danzigercartoons.com

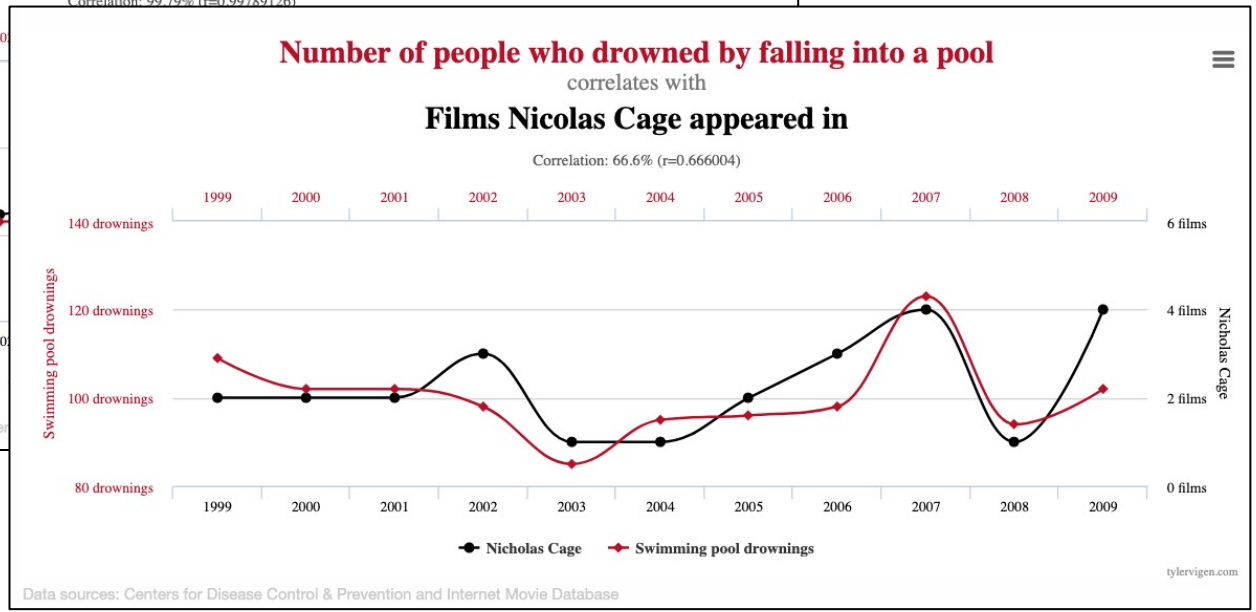


- For causation
  - Provide a theory (from domain knowledge, independent of data)
  - Show correlation
  - Demonstrate ability to predict new cases (replicate/validate)

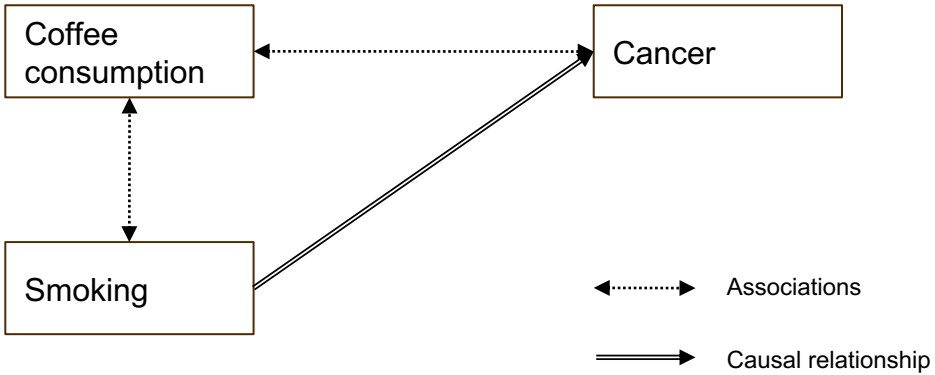
http://xkcd.com/552/

Spurious Correlations





Confounding variables



- If you look only at the coffee consumption → cancer relationship, you can get very misleading results
- Smoking is a confounder

RESEARCH-ARTICLE

Coverage is not strongly correlated with test suite effectiveness

Authors:  Laura Inozemtseva,  Reid Holmes Authors Info & Affiliations

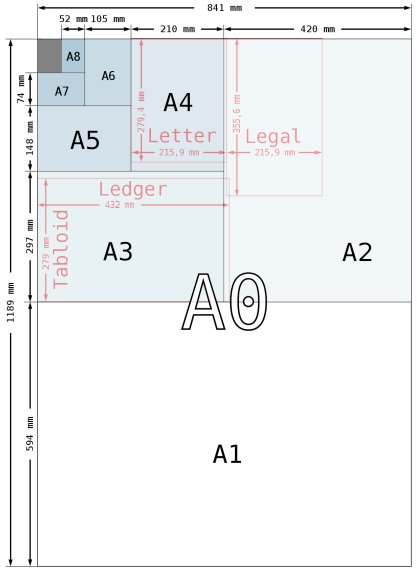
ICSE 2014: Proceedings of the 36th International Conference on Software Engineering • May 2014 • Pages 435–445 • https://doi.org/10.1145/2568225.2568271

"We found that there is a low to moderate correlation between coverage and effectiveness when the number of test cases in the suite is controlled for."

Measurements validity

- *Construct validity* – Are we measuring what we intended to measure?
- *Internal validity* – The extent to which the measurement can be used to explain some other characteristic of the entity being measured
- *External validity* – Concerns the generalization of the findings to contexts and environments, other than the one studied

Measurements reliability



44” (ish)

Measurements reliability

- Extent to which a measurement yields similar results when applied multiple times
- Goal is to teduce uncertainty, increase consistency
- Example: Performance
  - Time, memory usage
  - Cache misses, I/O operations, instruction execution count, etc.
- Law of large numbers
  - Taking multiple measurements to reduce error
  - Trade-off with cost

McNamara fallacy

- Measure whatever can be easily measured.
- Disregard that which cannot be measured easily.
- Presume that which cannot be measured easily is not important.
- Presume that which cannot be measured easily does not exist.

https://chronotopeblog.com/2015/04/04/the-mcnamara-fallacy-and-the-problem-with-numbers-in-education/

The McNamara Fallacy

- There seems to be a general misunderstanding to the effect that a mathematical model cannot be undertaken until every constant and functional relationship is known to high accuracy. This often leads to the omission of admittedly highly significant factors (most of the “intangibles” influences on decisions) because these are unmeasured or unmeasurable. To omit such variables is equivalent to saying that they have zero effect… Probably the only value known to be wrong…
  - J. W. Forrester, Industrial Dynamics, The MIT Press, 1961

DISCUSSION: MEASURING USABILITY

Example: Measuring usability.

- Automated measures on code repositories
- Use or collect process data
- Instrument program (e.g., in-field crash reports)
- Surveys, interviews, controlled experiments, expert judgment
- Statistical analysis of sample

METRICS AND INCENTIVES

Goodhart's law: "When a measure becomes a target, it ceases to be a good measure."

http://dilbert.com/strips/comic/1995-11-13/

Productivity Metrics

- Lines of code per day?
  - Industry average 10-50 lines/day
  - Debugging + rework ca. 50% of time
- Function/object/application points per month
- Bugs fixed?
- Milestones reached?

Incentivizing Productivity

- What happens when developer bonuses are based on
  - Lines of code per day?
  - Amount of documentation written?
  - Low number of reported bugs in their code?
  - Low number of open bugs in their code?
  - High number of fixed bugs?
  - Accuracy of time estimates?

Warning

- Most software metrics are controversial
  - Usually only plausibility arguments, rarely rigorously validated
  - Cyclomatic complexity was repeatedly refuted and is still used
  - “Similar to the attempt of measuring the intelligence of a person in terms of the weight or circumference of the brain”
- Use carefully!
- Code size dominates many metrics
- Avoid claims about human factors (e.g., readability) and quality, unless validated
- Calibrate metrics in project history and other projects
- Metrics can be gamed; you get what you measure

(Some) strategies

- Metrics tracked using tools and processes (process metrics like time, or code metrics like defects in a bug database).
- Expert assessment or human-subject experiments (controlled experiments, talk-aloud protocols).
- Mining software repositories, defect databases, especially for trend analysis or defect prediction.
  - Some success e.g., as reported by Microsoft Research
- Benchmarking (especially for performance).

Summary

- Measurement is difficult but important for decision making
- Software metrics are easy to measure but hard to interpret, validity often not established
- Many metrics exist, often composed; pick or design suitable metrics if needed
- Careful in use: monitoring vs incentives
- Strategies beyond metrics