



**CASE WESTERN RESERVE**  
UNIVERSITY

---

# **Metrics and Measurements**

CSDS 393/493: Software Engineering

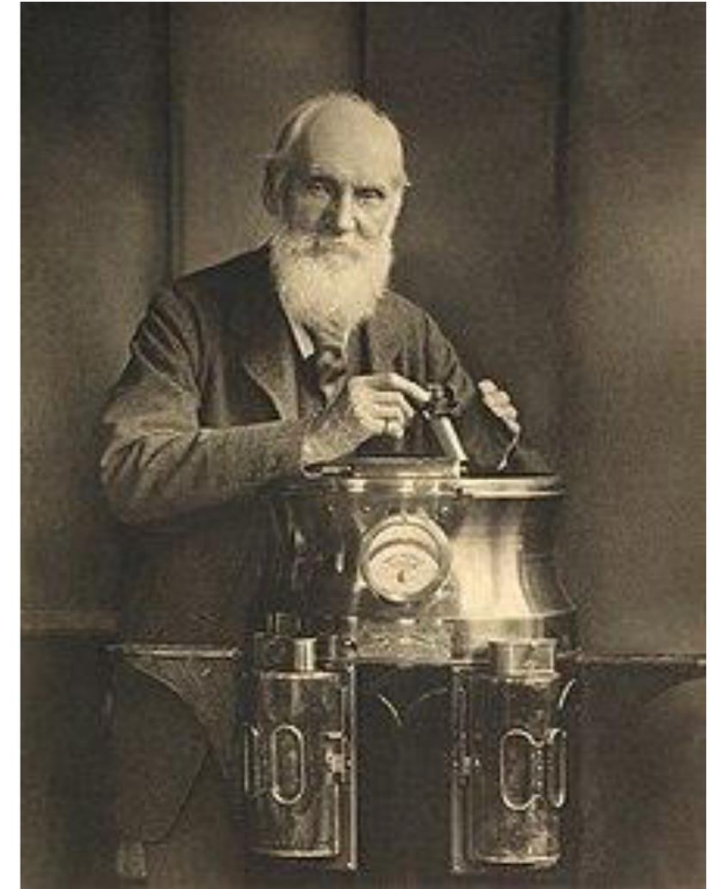
Spring 2025

# Logistics

- Quiz 1: February 6
  - See Week 2 lectures on Requirement
- Assignment 2: Due Feb 10
  - Software Requirements Specification (SRS)
- **Demo 1: Next week**
  - During the weekly meeting with the TA

*“To measure is to know;  
if you can not measure it,  
you can not improve it”*

William Thomson, Lord Kelvin

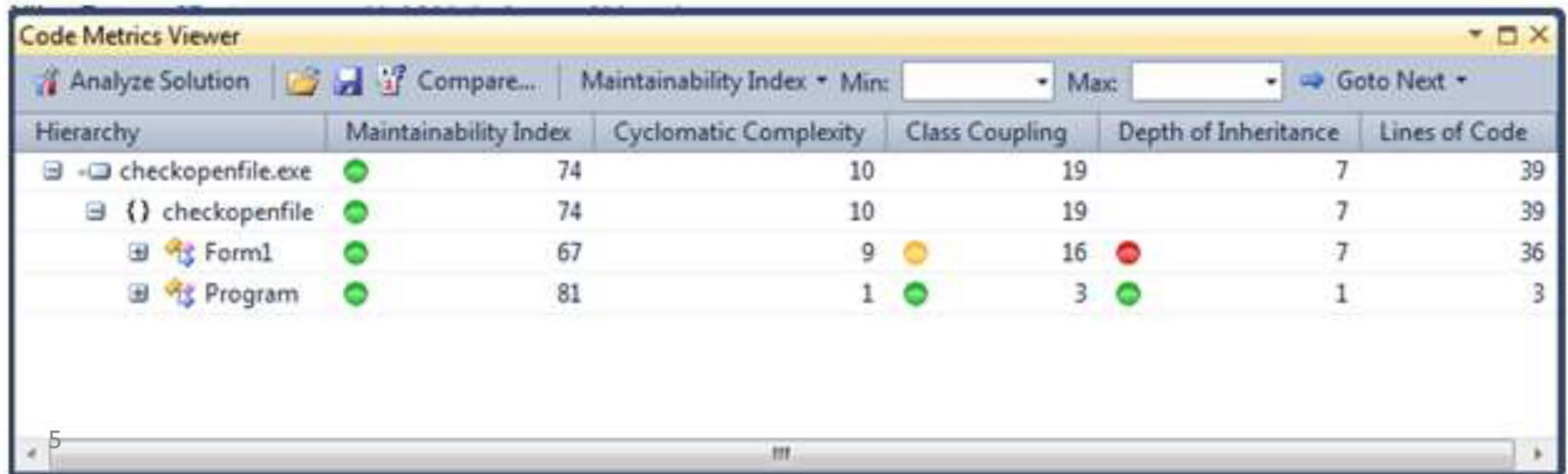


# Outline

- **Measurements and Metrics**
- How to use measurements and metrics?
- Case study: Autonomous Vehicle Software
- Risks and challenges
- Metrics and incentives

# Visual Studio

- Maintainability Index calculates an index value between 0 and 100 that represents the relative ease of maintaining the code.



The screenshot shows the 'Code Metrics Viewer' window in Visual Studio. The window has a toolbar with 'Analyze Solution', 'Compare...', and 'Goto Next' buttons. Below the toolbar is a table with columns: Hierarchy, Maintainability Index, Cyclomatic Complexity, Class Coupling, Depth of Inheritance, and Lines of Code. The table lists metrics for 'checkopenfile.exe', 'checkopenfile', 'Form1', and 'Program'. Each row includes a green circle icon next to the Maintainability Index value. The 'Form1' row also has a yellow circle icon next to the Class Coupling value and a red circle icon next to the Depth of Inheritance value.

Hierarchy	Maintainability Index	Cyclomatic Complexity	Class Coupling	Depth of Inheritance	Lines of Code
[-] checkopenfile.exe	74	10	19	7	39
[-] {} checkopenfile	74	10	19	7	39
[-] Form1	67	9	16	7	36
[-] Program	81	1	3	1	3

# 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 ...”

# The Index

Maintainability Index =

MAX(0,(171 –

5.2 \* log(Halstead Volume) –

0.23 \* (Cyclomatic Complexity) –

16.2 \* log(Lines of Code)

)\*100 / 171)

# Software Quality Metric

IEEE 1061 says:

*“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 software processes a given attribute that affects its quality.”*

**Entity**

**Attribute**

**Measurement**



# What Entities Do We Care About?

- Software product
- Modules
- Software development process
- People

# What software qualities do we care about?

- Functionality (e.g., data integrity)
- Scalability
- Security
- Extensibility
- Bugginess
- Documentation
- Performance
- Installability
- Availability
- Consistency
- Portability
- Regulatory compliance

# What process qualities do we care about?

- Development efficiency
- Meeting efficiency
- Conformance to processes
- Reliability of predictions
- Fairness in decision making
- Regulatory compliance
- On-time release

# What people qualities do we care about?

- Developers

- Maintainability
- Performance
- Satisfaction and well-being
- Communication
- Efficiency and flow
- Satisfaction with engineering
- Regulatory compliance

- Customers

- Satisfaction
- Ease of use
- Feature usage
- Regulatory compliance

# McNamara Fallacy

1. Measure whatever can be easily measured.
2. Disregard that which cannot be measured easily.
3. Presume that which cannot be measured easily does not exist or is not important.

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

# Non-trivial Qualities

- **Software**

- Code elegance
- Code maintainability

- **Process**

- Fairness in decision making

- **Team**

- Team collaboration
- Creativity

# Everything is Measurable (?)

- If  $X$  is something we care about, then  $X$ , by definition, must be **detectable**.
- 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**.

# 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)



# Code Complexity: Halstead Volume

Introduced by Maurice Howard Halstead in 1977

$$\text{Halstead Volume} = \text{number of operators/operands} * \log_2(\text{number of distinct operators/operands})$$

Approximates size of elements and vocabulary

# Example: Halstead Volume

```
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

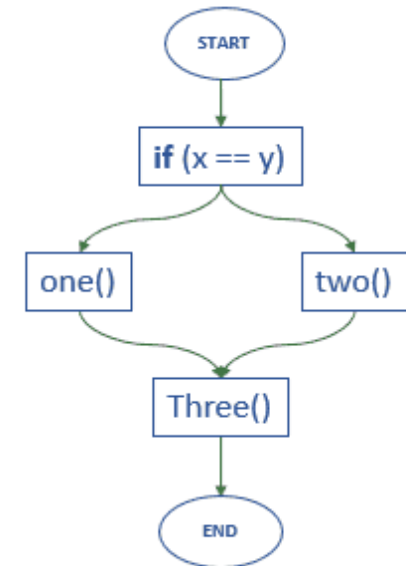
$$M = E - N + 2P,$$

where

- E = the number of edges of the graph.
- N = the number of nodes of the graph.
- P = the number of connected components

```
Void foo (void)
{
    if (x == y)
        one();
    else
        two();

    three();
}
```



Cyclomatic Complexity = 6 (Edges) - 6 (Nodes) + 2 = 2

# Code Complexity: 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
- ...

# Outline

- Measurements and Metrics
- **How to use measurements and metrics?**
- Case study: Autonomous Vehicle Software
- Risks and challenges
- Metrics and incentives

# A Goal-based Framework

“Every measurement action must be motivated by a particular goal or need that is **clearly defined and easily understandable.**”

- Software Metrics: A Rigorous and Practical Approach. N.Fenton, J.Bieman

# GQM: Defining Goals

**P: Purpose** (improve, evaluate, monitor, ...)

**I: Issue** (reliability, usability, effectiveness, ...)

**O: Object** (final product, component, process, activity)

**V: Viewpoint** (any stakeholder)

**Goal:**

**Evaluate** the **effectiveness** of the **organization's coding standard** from the **team's** perspective

**Questions:**

How comprehensible are the coding standards?

What is the impact of coding standards on the efficiency and productivity of the team?

**Metrics:**

Survey results measuring team members' understanding

Number of revisions required to achieve standard compliance

Code size: LOC, number of classes, number of functions



**Goal:**

**Monitor** the **performance** of the **web server** to enable the **ops team** to make decisions

**Questions:**

How quickly can users complete their tasks?

How many concurrent users can we support?

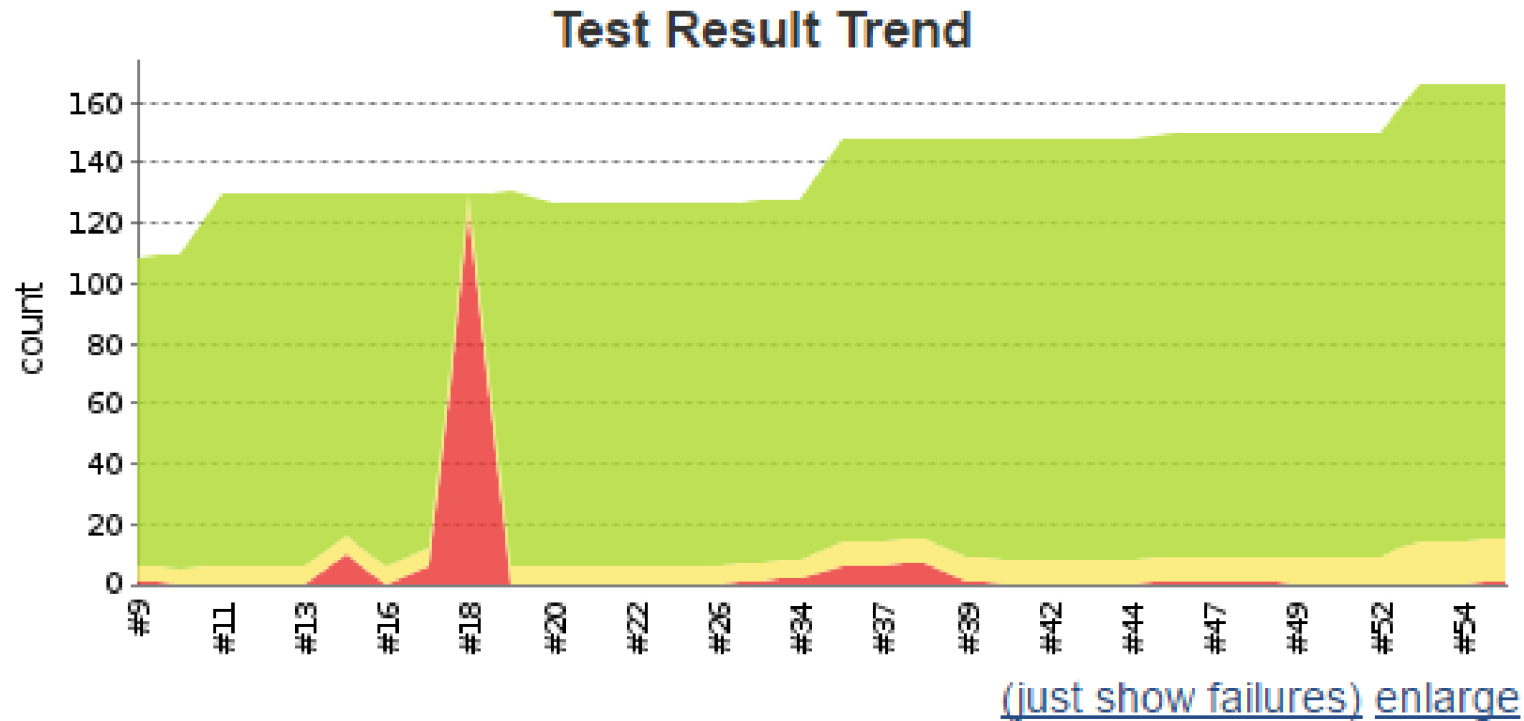
**Metrics:**

Average latency per request in milliseconds

Throughput:  
Number of requests served per second

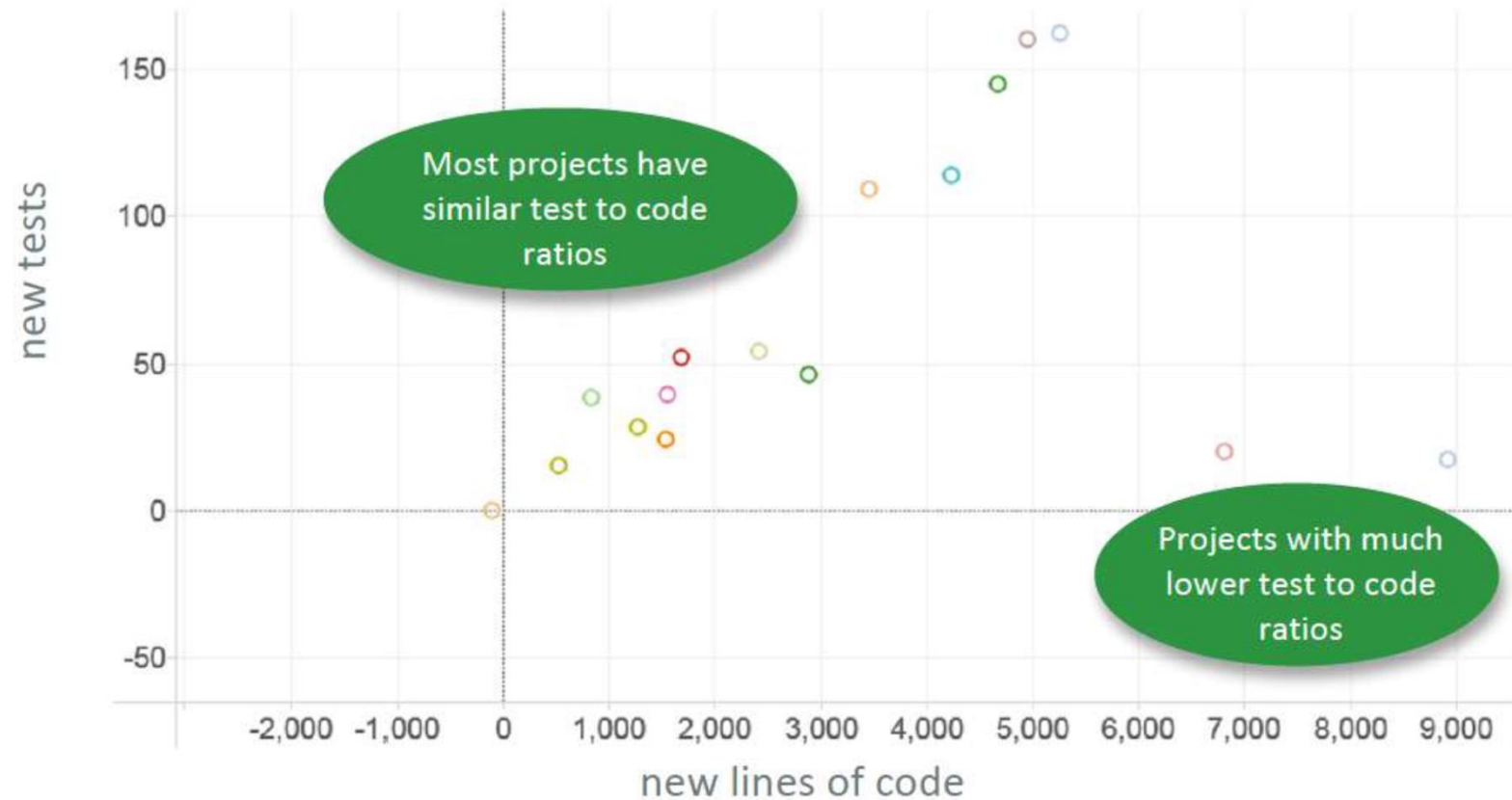
Peak memory consumption, as a % of max available

# Trend Analyses



Jenkins test result trend

# Benchmark-Based Metrics



<https://github.blog/news-insights/company-news/github-welcomes-semmle/>

# Outline

- Measurements and Metrics
- How to use measurements and metrics?
- **Case study: Autonomous Vehicle Software**
- Risks and challenges
- Metrics and incentives

# By what metrics can we judge AV software (e.g., safety)?



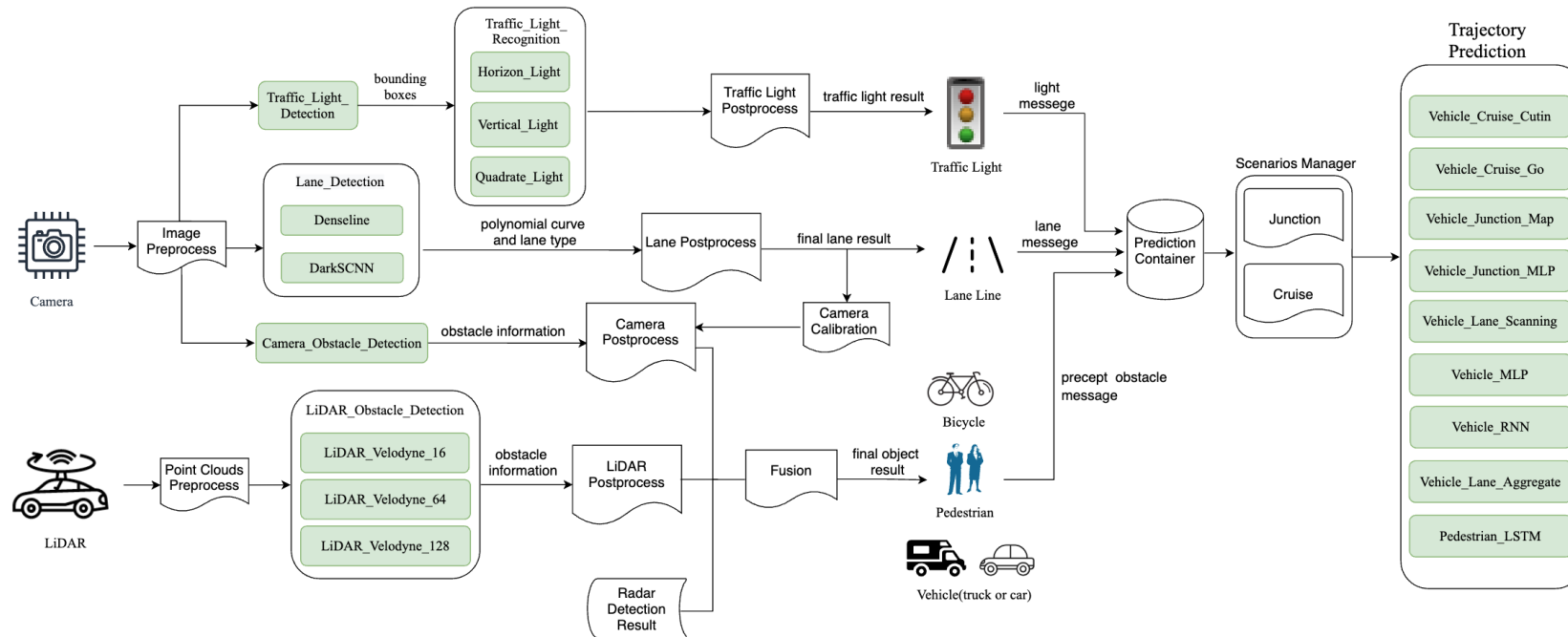
# (1) Code Coverage

- Amount of code executed during testing.
- Statement coverage, line coverage, branch coverage, etc.
  - E.g., 75% branch coverage  $\Rightarrow$  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]));
}
```

## (2) Model Accuracy

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





# (3) Failure Rate

- Frequency of crashes / fatalities
- Per 1,000 rides, per million miles, per month (in the news)

TRANSP

**Waymo's driverless cars were involved in two crashes and 18 'minor contact events' over 1 million miles**



Image: Allen J. Schaben / Los Angeles Times via Getty Images

/ The Alphabet-owned company pulls back the curtain on more stats from its public road testing. Of the 20 incidents, only two met the federal government's reporting criteria, and no one was injured.

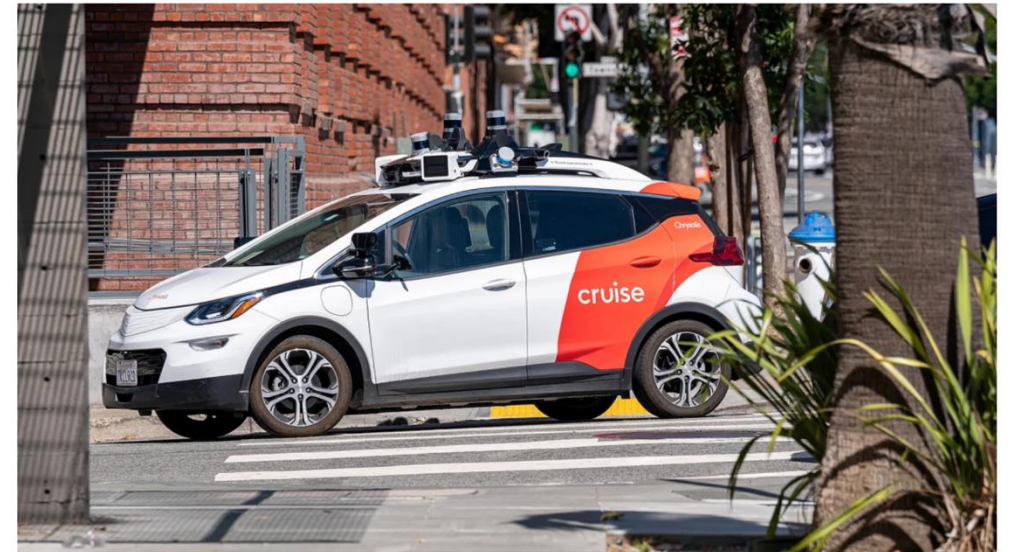
by **Andrew J. Hawkins**  
Feb 28, 2023, 12:00 PM EST

[Link](#) [Facebook](#) [Twitter](#) [Comment \(1 New\)](#)

**'Complete meltdown': Driverless cars in San Francisco stall causing a traffic jam**

By **Jordan Valinsky**, CNN Business

🕒 3 minute read · Updated 3:45 PM EDT, Mon August 14, 2023



A Cruise autonomous taxi in San Francisco. David Paul Morris/Bloomberg/Getty Images



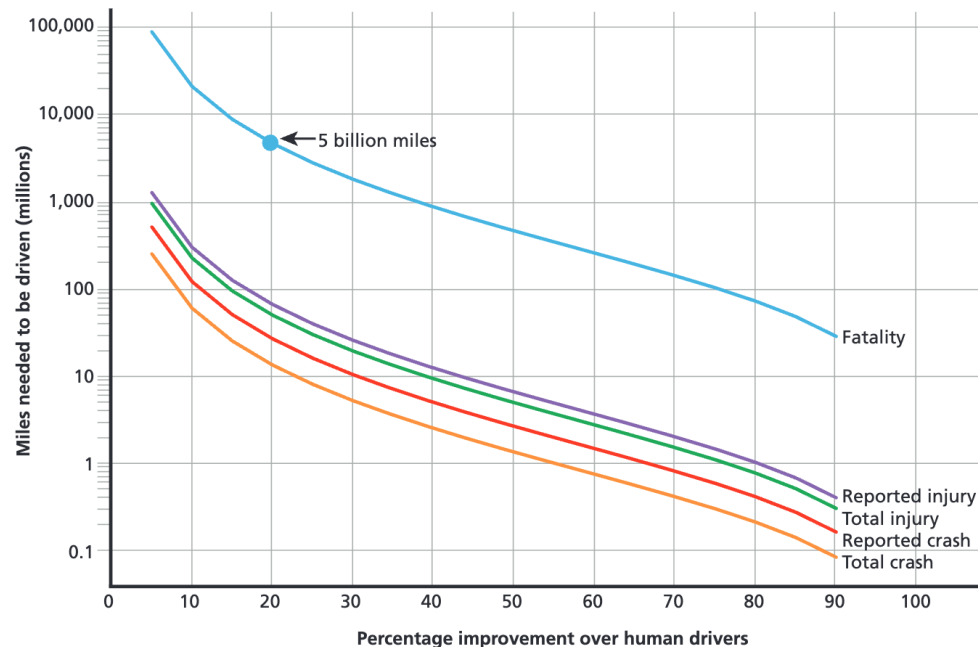
# (4) Mileage



## Driving to Safety

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

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



## Building the World's Most Experienced Driver™

The Waymo Driver gains experience with every mile, in each car.



10+

More than a Decade of Autonomous Driving in More than 10 States

5

Generations of Autonomously Driven Vehicles

15+

Billion Autonomously Driven Miles in Simulation

20+

Million Real-World Miles on Public Roads

Source: waymo.com/safety (September 2021)

# Example

**Goal:** Ensure energy efficiency and sustainability from the point of view of the organization and environmental analysts

**Question 1:** What is the vehicle's energy consumption under different driving conditions?

**Metrics:** Kilowatt-hours per 100 kilometers under city, highway, and mixed driving conditions.

**Question 2:** How efficient is the battery management system?

**Metrics:** Battery life in miles, number of charge cycles

# Outline

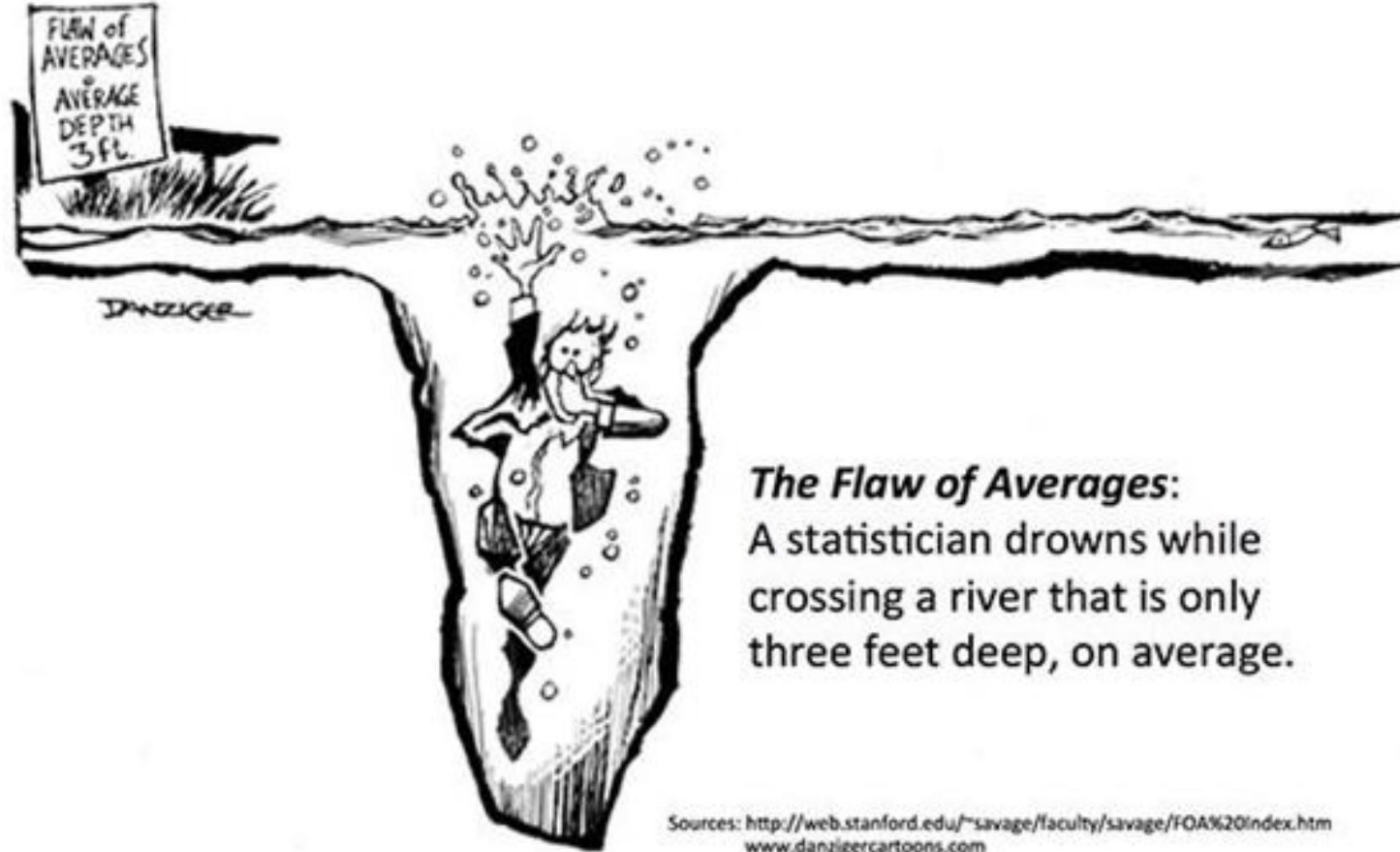
- Measurements and Metrics
- How to use measurements and metrics?
- Case study: Autonomous Vehicle Software
- **Risks and challenges**
- Metrics and incentives

# 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.



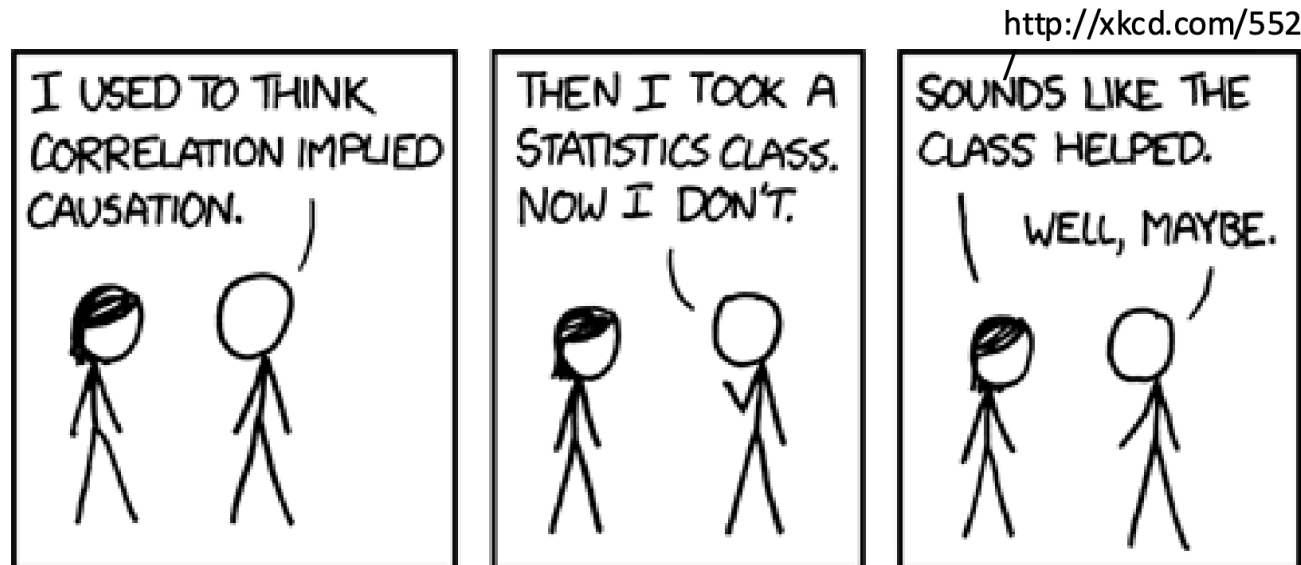
# Bad Statistics: What Could Possibly Go Wrong?



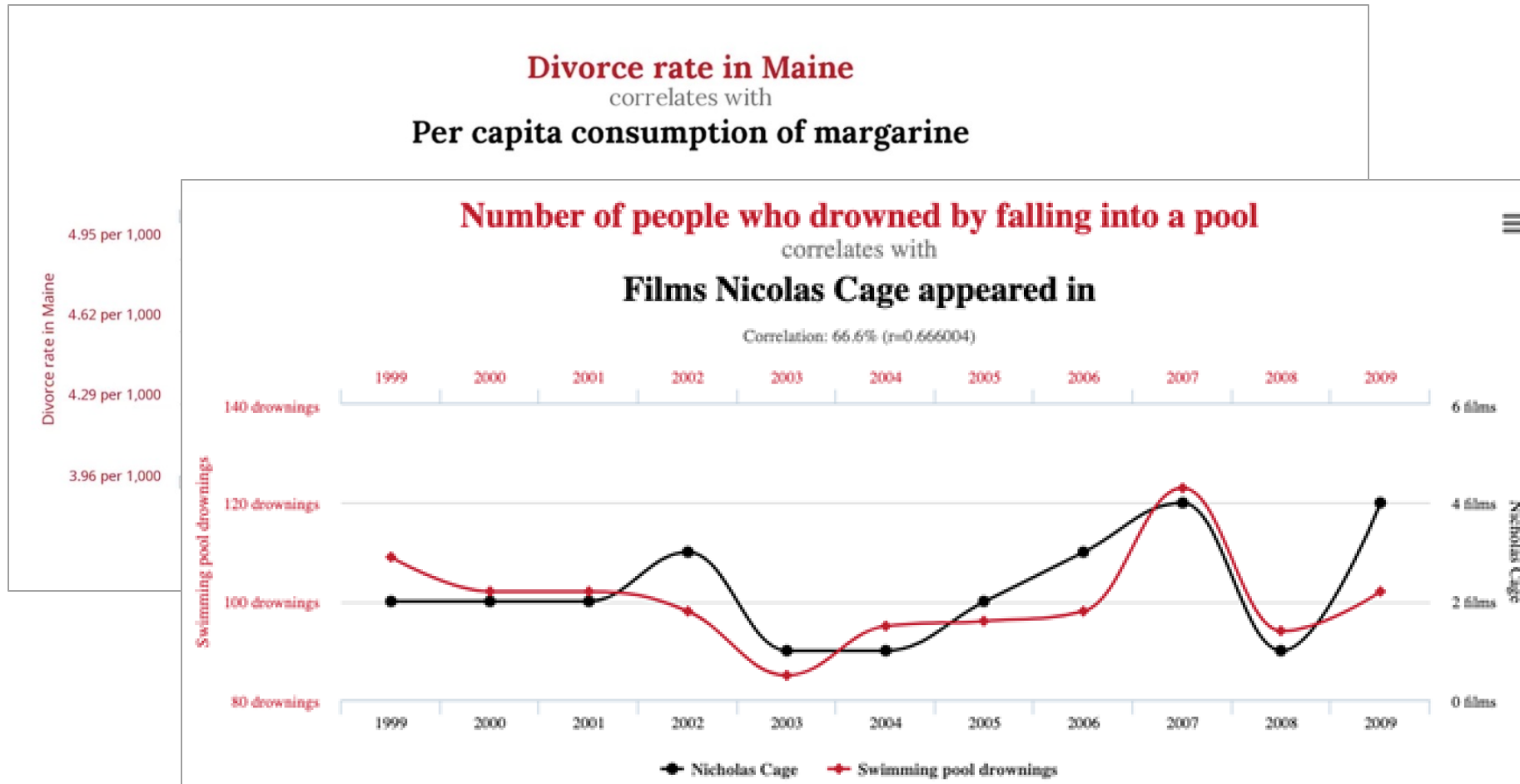
# Making Inferences

To infer causation:

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

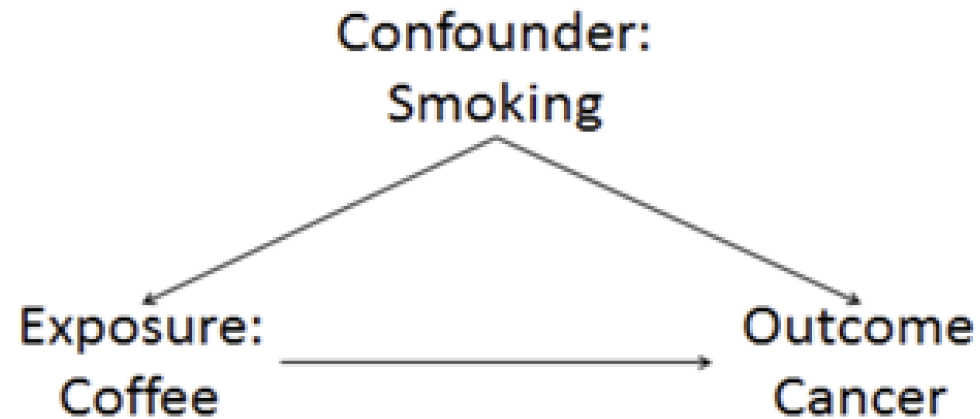


# Spurious Correlations



# Spurious Correlations: Confounding Variables

If you look only at the coffee consumption → cancer relationship, you can get very misleading results



Smoking is a confounder



# Outline

- Measurements and Metrics
- How to use measurements and metrics?
- Case study: Autonomous Vehicle Software
- Risks and challenges
- **Metrics and incentives**

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



# Productivity Metrics

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

# Takeaways



Metrics are important in Software Engineering



Apply goal-oriented approaches to software metrics



Provide examples of metrics for software qualities and process



Understand limitations and dangers of decisions and incentives based on measurements