# Test Selection Safety Evaluation Framework

Claire Leong

**Goal**: to make a generic framework for evaluating test scheduling algorithms at scale from the historical record.

# Project Overview

- Implementation:
    1. Determine safety information for historical changelists
    2. Evaluate the safety of test selection algorithms
    3. Implement optimistic, pessimistic and random test selection algorithms

# Project Overview

- Used over 2 datasets:

| Small Dataset | Large Dataset |
| --- | --- |
| 2 days of CL data (6-8 Dec 2017) | 1 month of CL data (October 2017) |
| 11k changelists | 900k changelists |
| 1k total targets | 4m total targets |
| 430k times targets were affected | 16b times targets were affected |

# Determining safety

- Safety = would skipping this test target miss a transition?
- Transition = a change in target results, either from failing->passing or passing->failing

# Safe Targets *skipping this target would not miss a transition*

**Time** →

| Changelist | CL1 | CL2 |
|---|---|---|
| Target Result | P | P |
| **Safety** | **-** | **Safe** |
| Transition | - | P->P |

** = affected*

# Safe Targets *skipping this target would not miss a transition*

**Time** →

| Changelist | CL1 | CL2 |
|---|---|---|
| Target Result | F | F |
| **Safety** | **-** | **Safe** |
| Transition | - | F->F |

** = affected*

# Safe Targets *skipping this target would not miss a transition*

**Time** →

| Changelist | CL1 | CL2 | CL3 |
|---|---|---|---|
| Target Result | P | * | P |
| **Safety** | **-** | **Safe** | **Safe** |
| Transition | - | P->P | P->P |

** = affected*

# Safe Targets *skipping this target would not miss a transition*

**Time** →

| Changelist | CL1 | CL2 | CL3 |
|---|---|---|---|
| Target Result | F | * | F |
| **Safety** | **-** | **Safe** | **Safe** |
| Transition | - | F->F | F->F |

** = affected*

# Unsafe Targets *skipping this target would definitely miss a transition*

**Time** ⟶

| Changelist | CL1 | CL2 |
|---|---|---|
| Target Result | P | F |
| **Safety** | **-** | **Unsafe** |
| Transition | - | P->F |

** = affected*

# Unsafe Targets *skipping this target would definitely miss a transition*

**Time** →

| Changelist | CL1 | CL2 |
|---|---|---|
| Target Result | F | P |
| **Safety** | **-** | **Unsafe** |
| Transition | - | F->P |

** = affected*

# Maybe Unsafe Targets *skipping this target might miss a transition*

**Time** →

| Changelist | CL1 | CL2 | CL3 |
|---|---|---|---|
| Target Result | P | * | F |
| **Safety** | **-** | **Maybe unsafe** | **Maybe unsafe** |
| Transition | - | P->F | P->F |

** = affected*

# Maybe Unsafe Targets *skipping this target might miss a transition*

**Time** →

| Changelist | CL1 | CL2 | CL3 |
|---|---|---|---|
| Target Result | F | * | P |
| **Safety** | **-** | **Maybe unsafe** | **Maybe unsafe** |
| Transition | - | F->P | F->P |

** = affected*

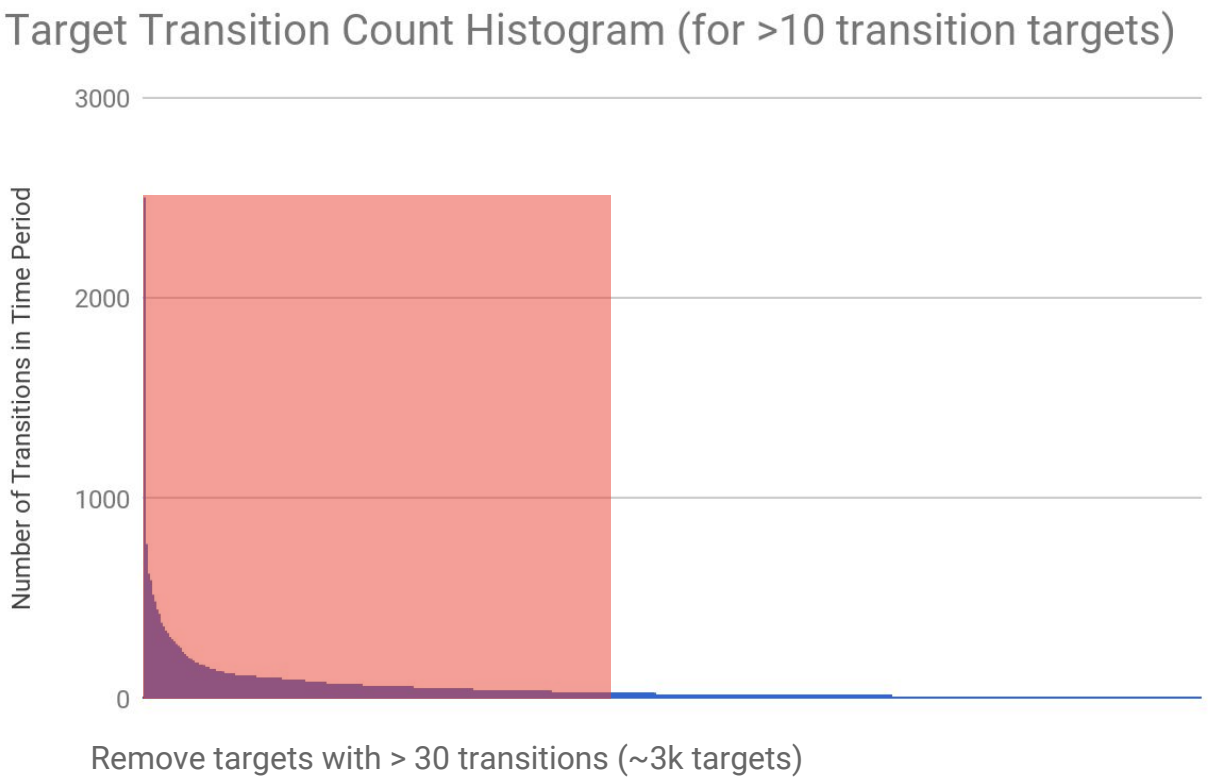
# Can we skip targets safely?

- This information is used to determine whether skipping a target is safe
- All non-definitive pass or fail results treated as affected
- We calculate the safety of skipping tests at rates from 0-100% for an algorithm

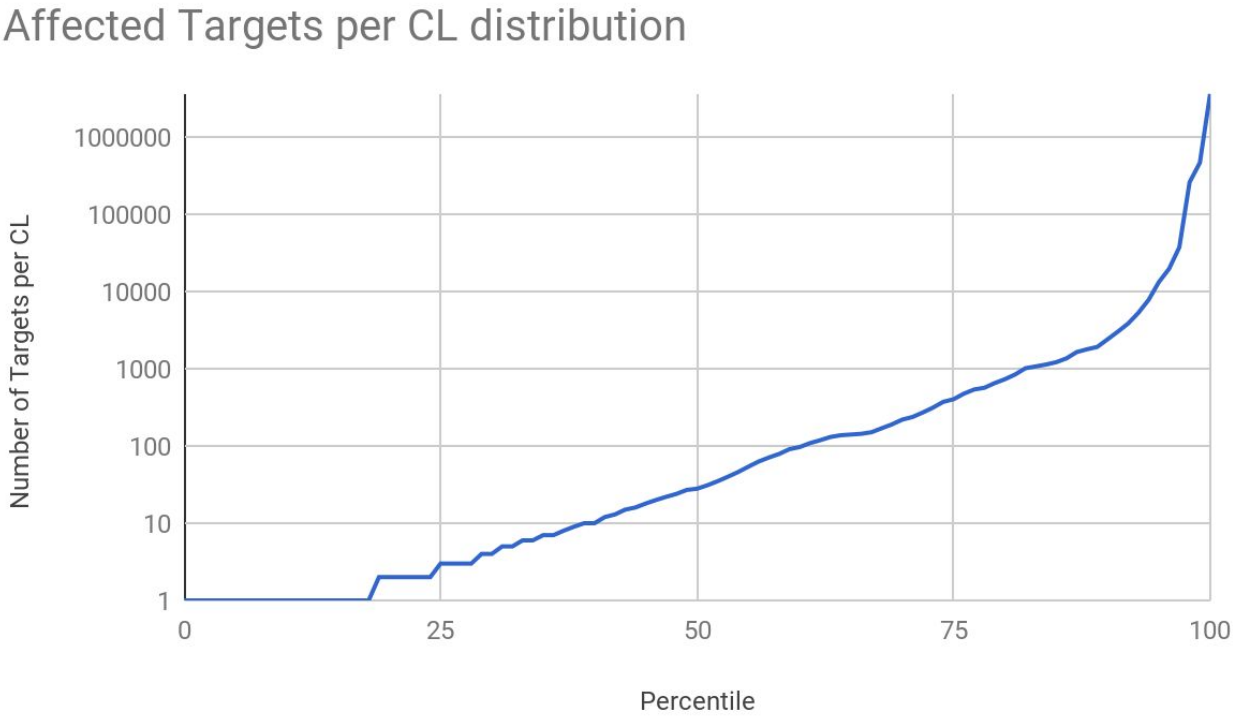
# Input data

- Input taken from Spanner backup’s Result and Affected tables
- Used 3 methods to eliminate flakes from the data
  - Only take pass and fail results
  - Removing target results identified as flaky by Kellogs
  - Removing targets with over X transitions in the time period

# Removing high transition count targets



# Targets per CL Distribution



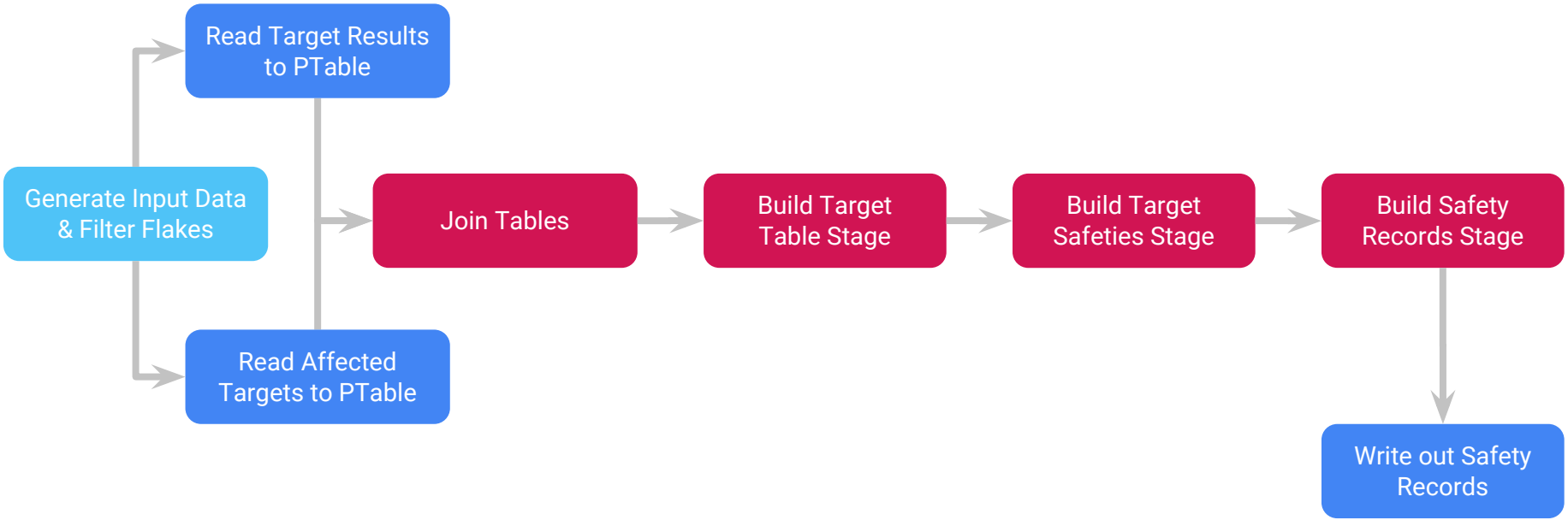

Stats:

- Median 38 tests!
- 90th percentile 2,604
- 95th percentile 4,702
- 99th percentile 55,730

# Implementation: Safety Data Builder

This package creates safety data given the historical changelist data as input.

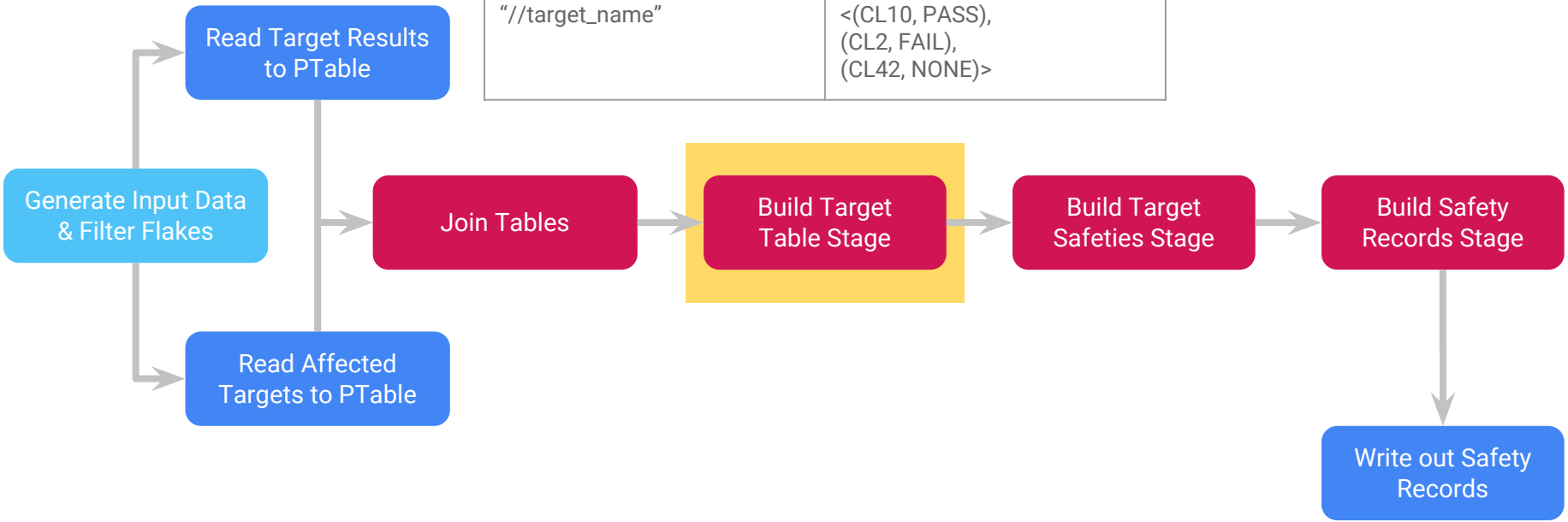
# Pipeline



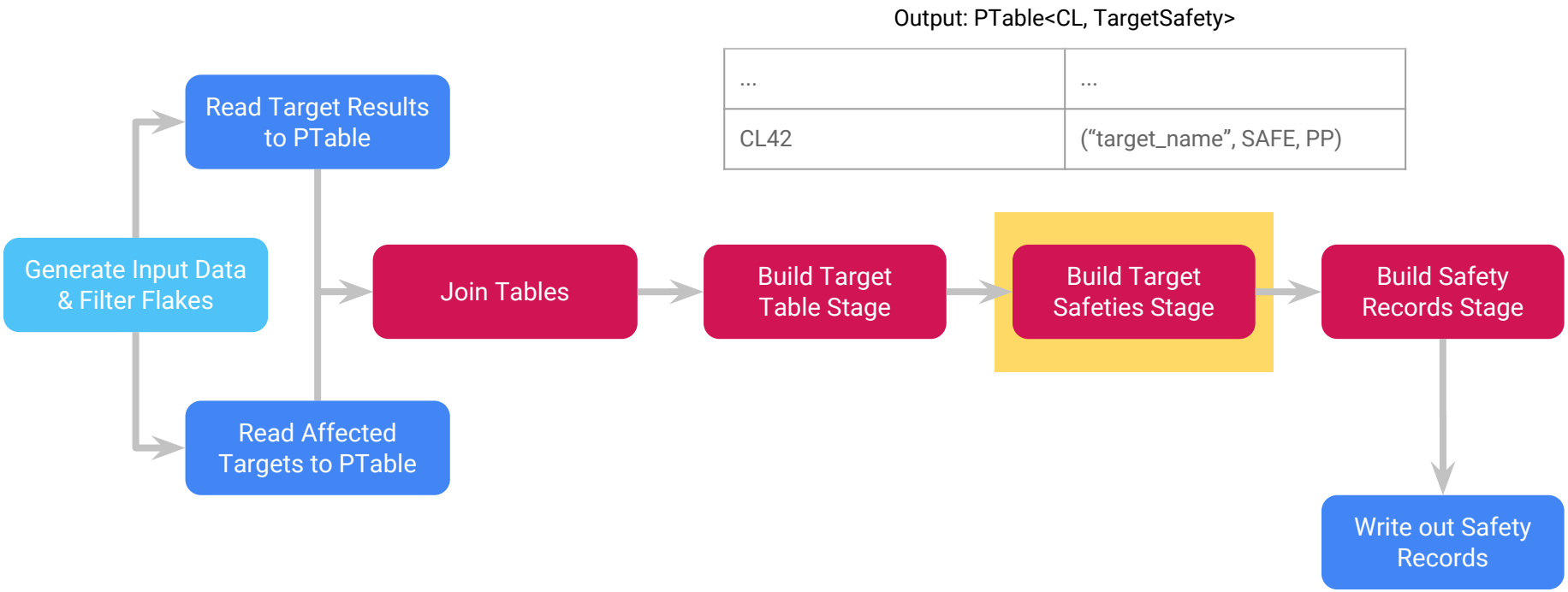
# Pipeline

Output: PTable<Target, Collection<ClAndResult>>

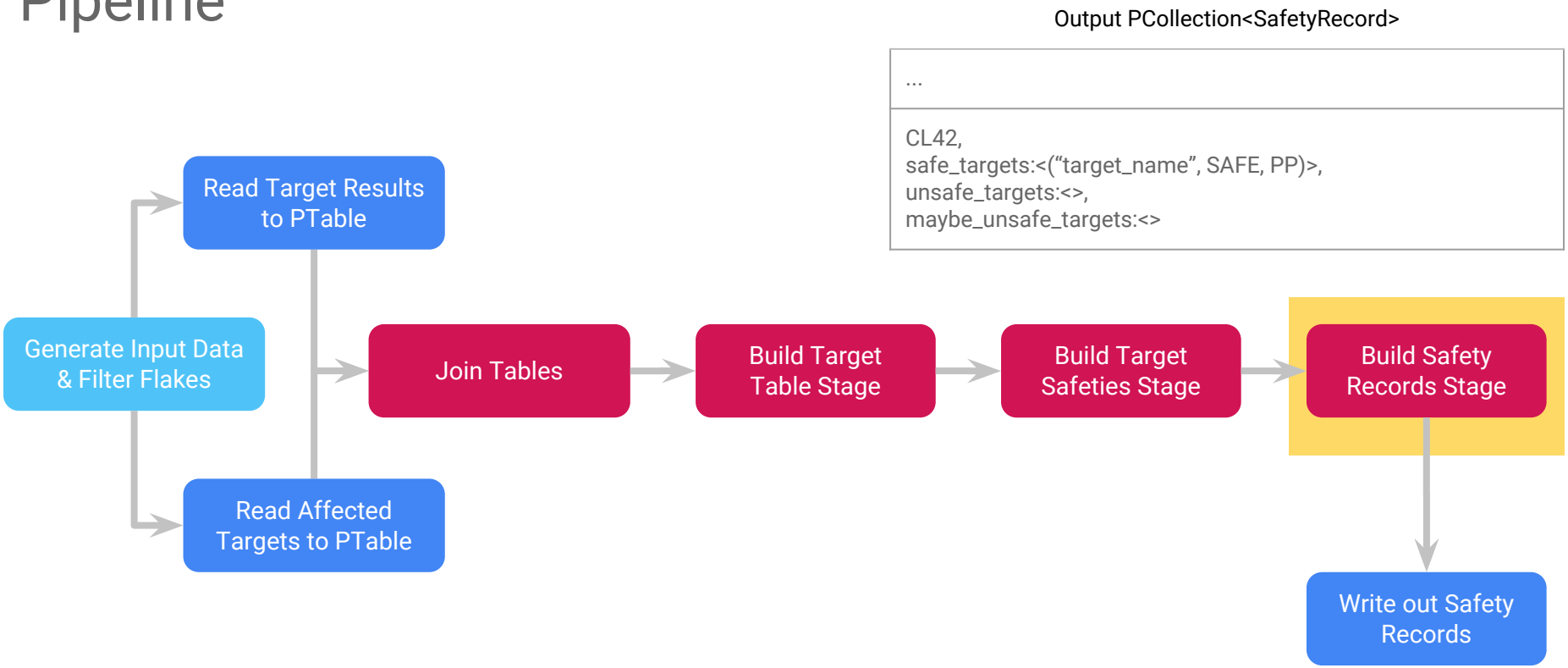
| | |
|---|---|
| ... | ... |
| “//target_name” | <(CL10, PASS),<br>(CL2, FAIL),<br>(CL42, NONE)> |



# Pipeline



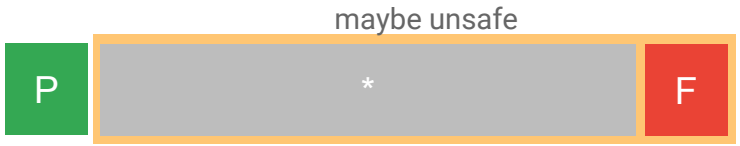
# Pipeline

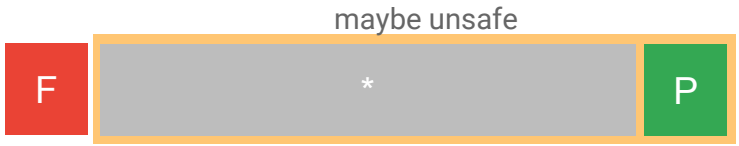


# Safety Data Results

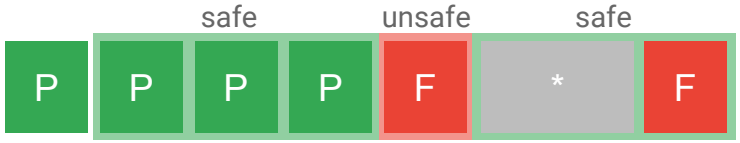
| | Small Data Set | Large Data Set |
|---|---|---|
| Total CLs | 10170 | 891,621 |
| CLs with only safe targets | 96.4% (9801) | 90.2% (804,160) |
| CLs with maybe unsafe | 3.4% (346) | 8.3% (73,897) |
| CLs with unsafe | 0.2% (25) | 1.5% (13,564) |
| Total target affecteds | 428,938 | 15,931,019,923 |
| Safe target affecteds | 99.9% (428,547) | 99.98% (15,927,853,638) |
| Maybe unsafe target affecteds | 0.09% (365) | 0.019% (3,054,667) |
| Unsafe target affecteds | 0.01% (26) | 0.0001% (111,618) |

# Culprit finding works!



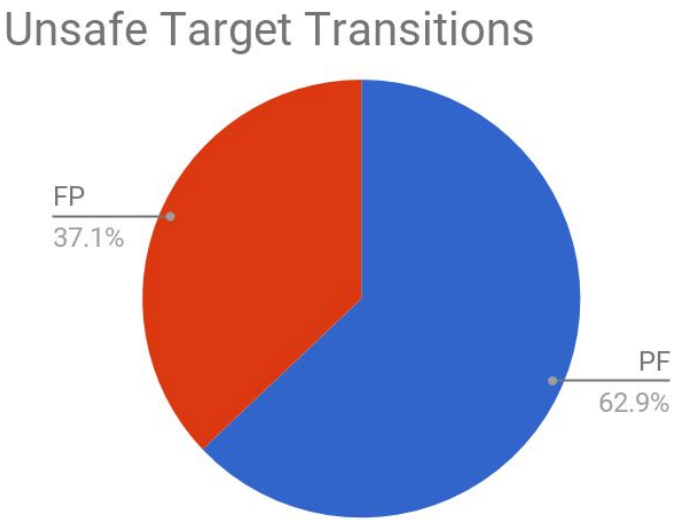



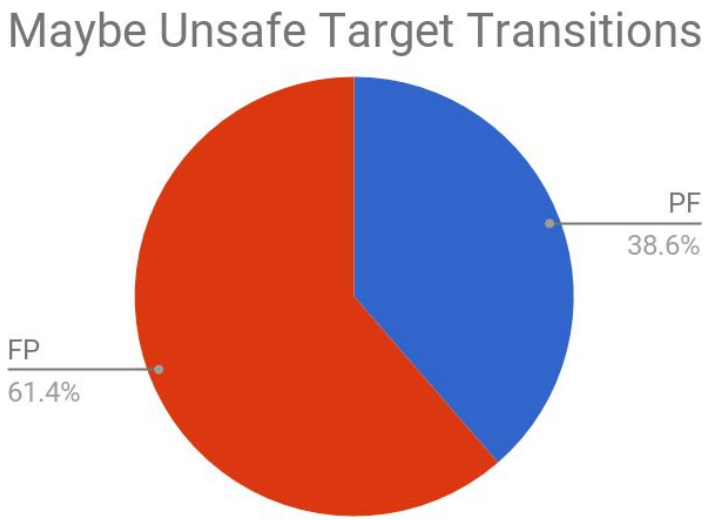

**With culprit finding:**




**We don’t do fix finding**

# Culprit finding works!



# Implementation: Algorithm Evaluator

This package evaluates the safety of using an algorithm to select tests to skip for a changelist.

# Evaluator Implementation

- For every changelist in the safety data, it will call an algorithm with skip rates from 0 to 100%
- Using the targets returned by the algorithm, determines if that selection was safe or not
  - Safe = no unsafe or maybe unsafe tests were skipped
  - Maybe unsafe = maybe unsafe tests were skipped but no unsafe tests
  - Unsafe = unsafe tests were skipped

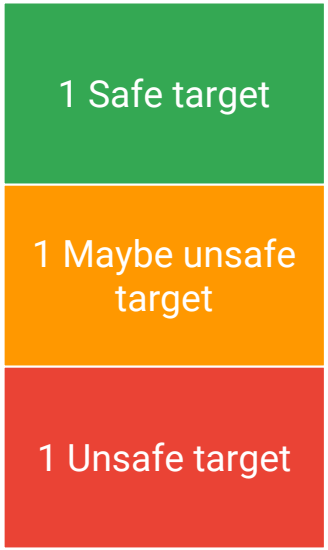
# Algorithms

- Algorithms are implementations of the interface TestSelectionAlgorithm which contains the method

```
ImmutableSet<Target> skipTargets(long cl, Iterable<Target> targets, int numToSkip)
```

# Algorithms - **Random**

Changelist's Affected Targets

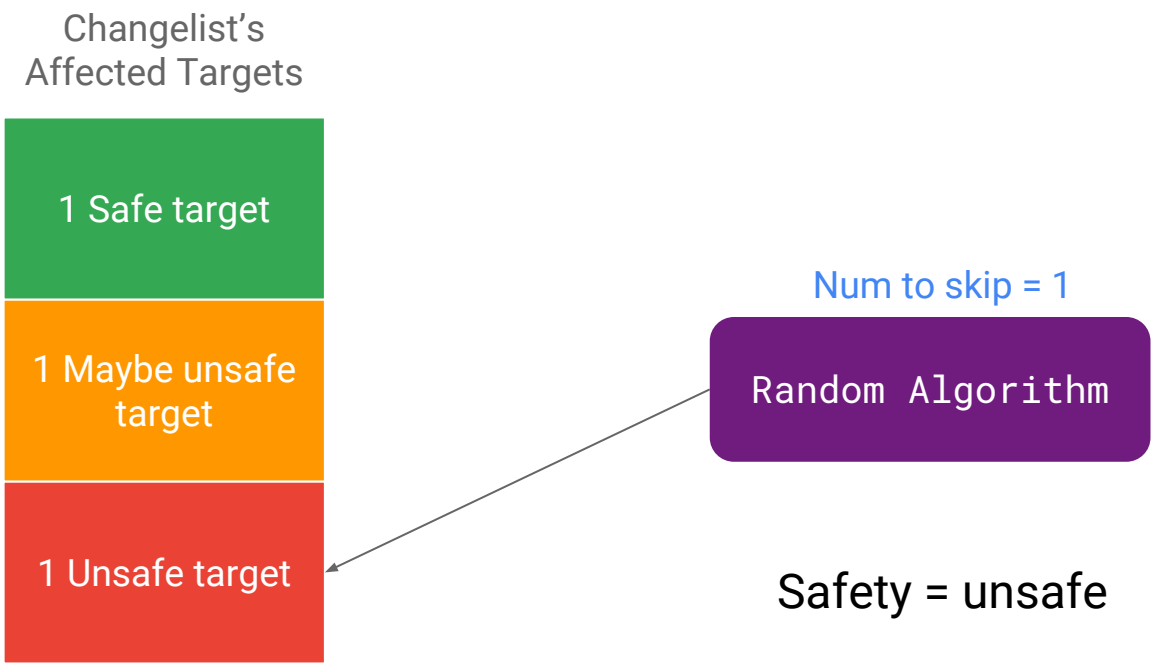



Num to skip = 0

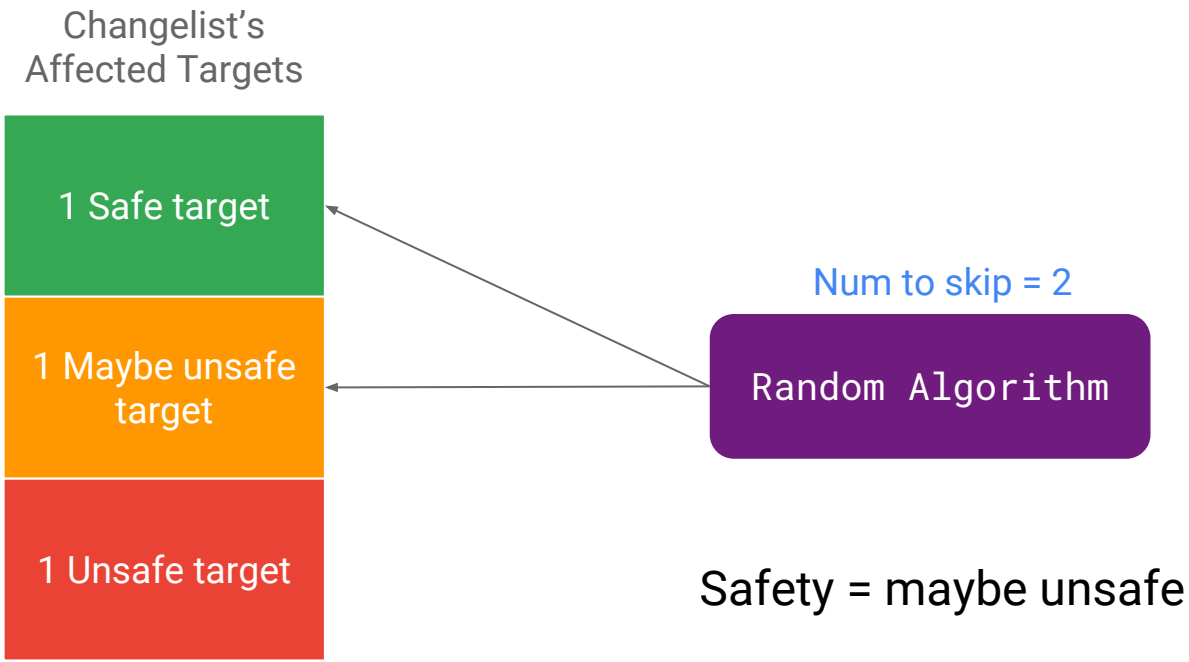
`Random Algorithm`

Safety = safe

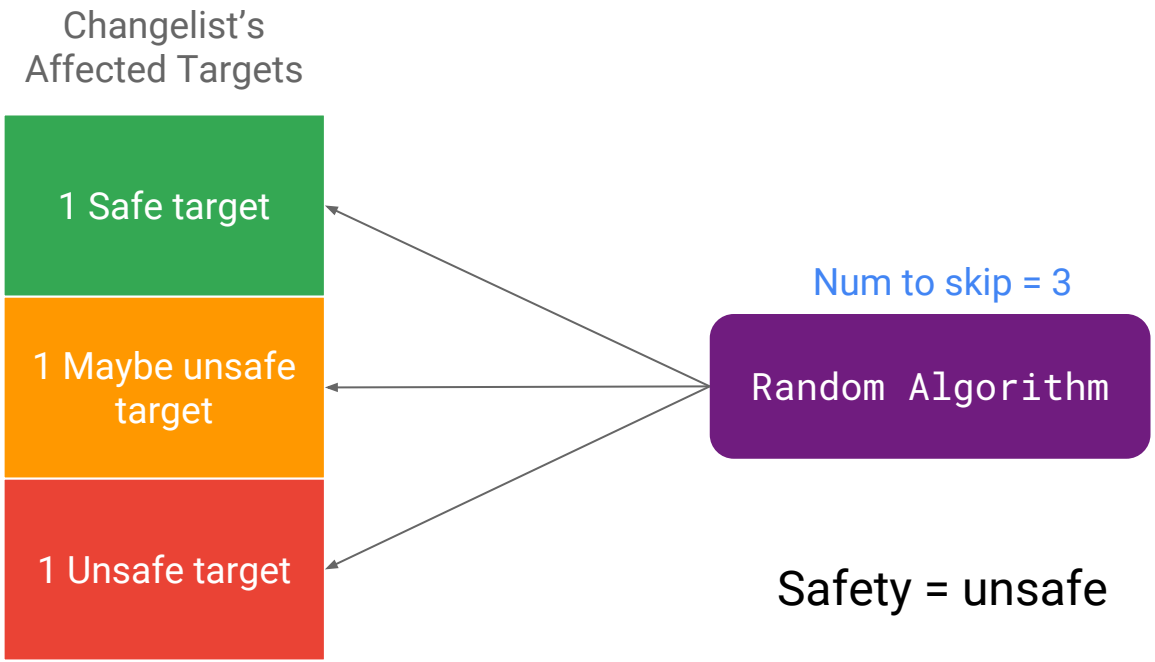
# Algorithms - **Random**



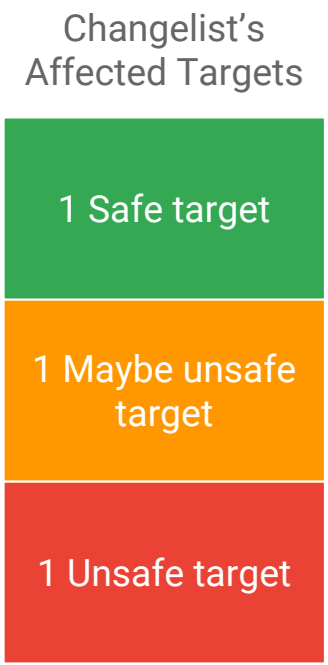
# Algorithms - **Random**



# Algorithms - **Random**



# Algorithms - **Optimistic**

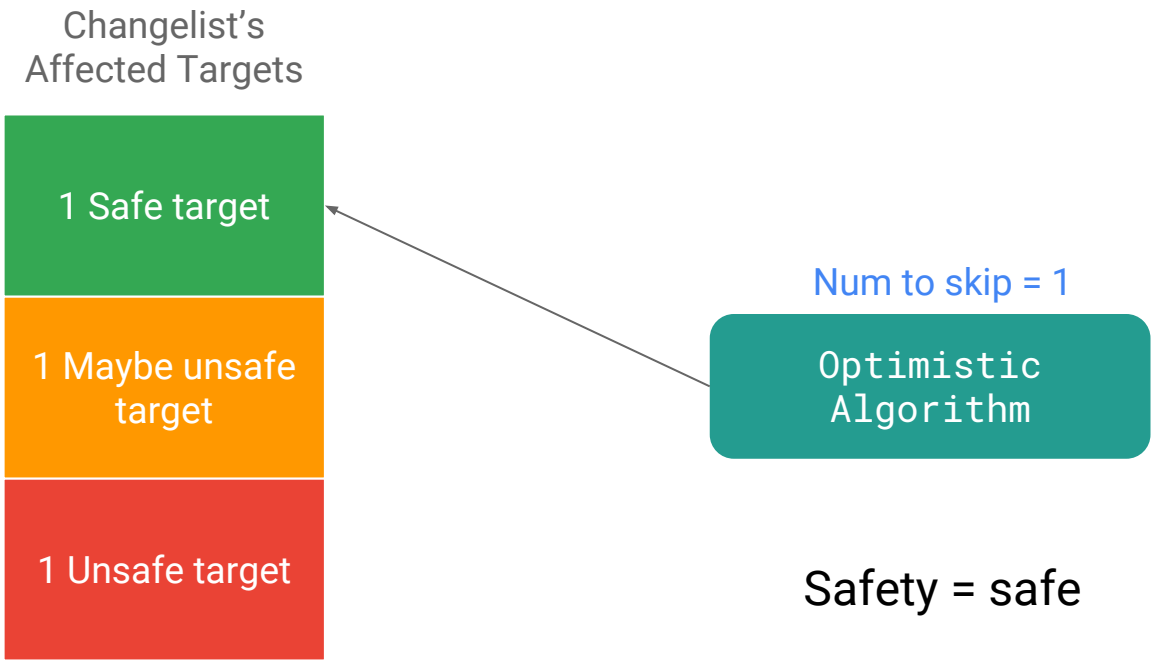



Num to skip = 0

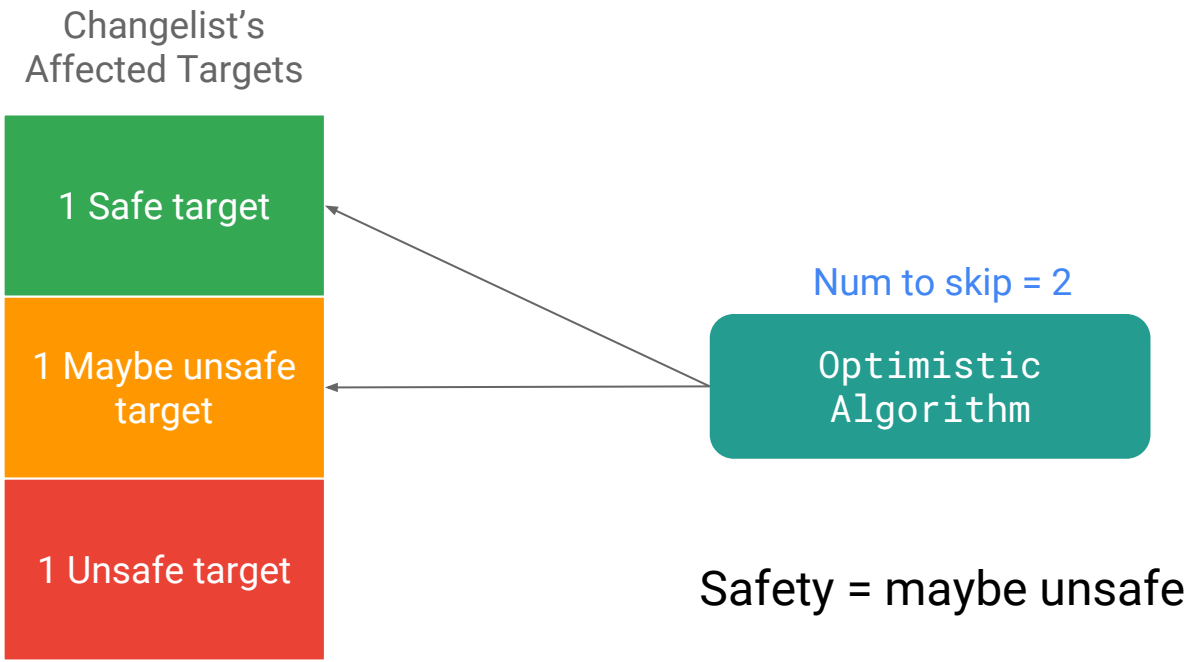
`Optimistic Algorithm`

Safety = safe

# Algorithms - **Optimistic**

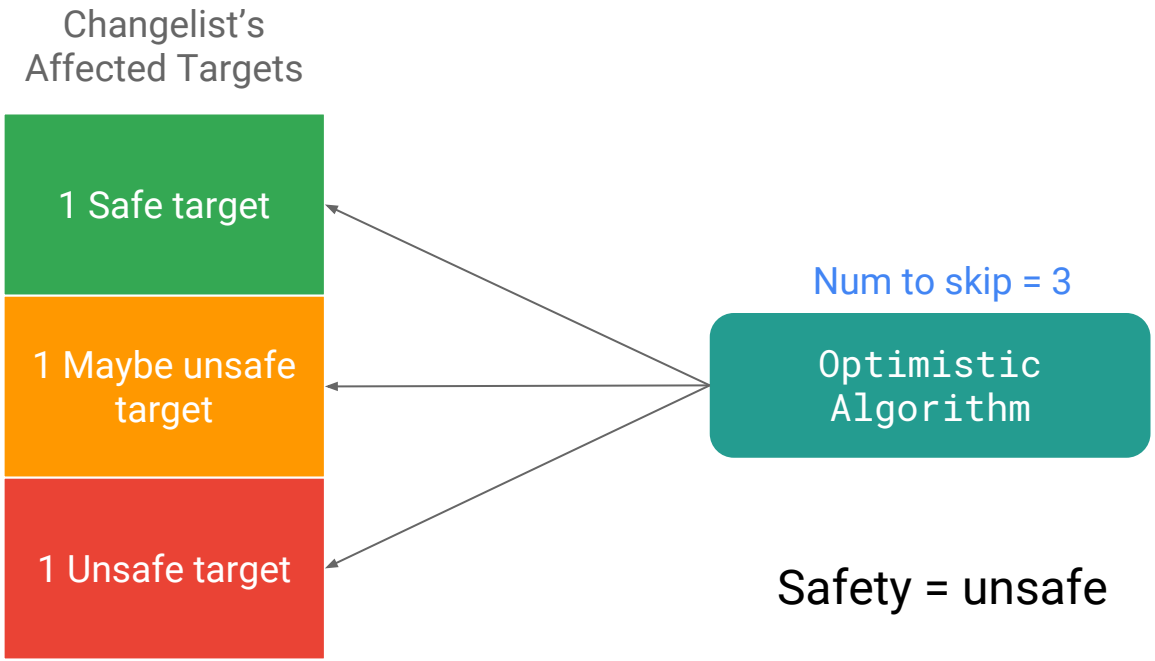


# Algorithms - **Optimistic**

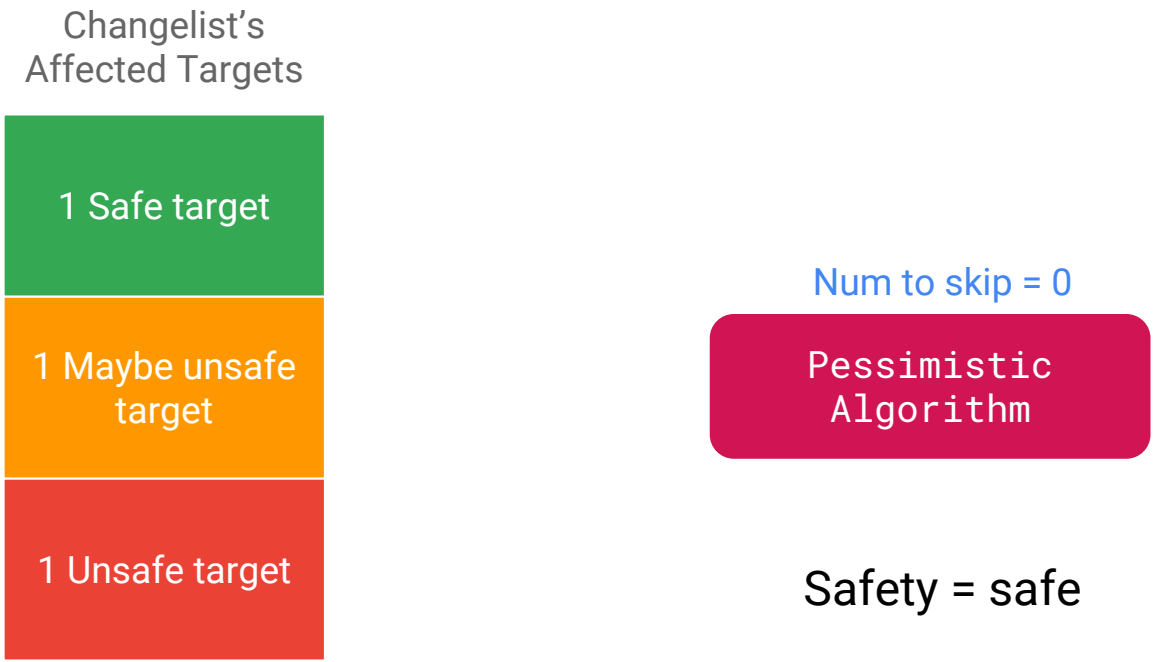



Safety = maybe unsafe

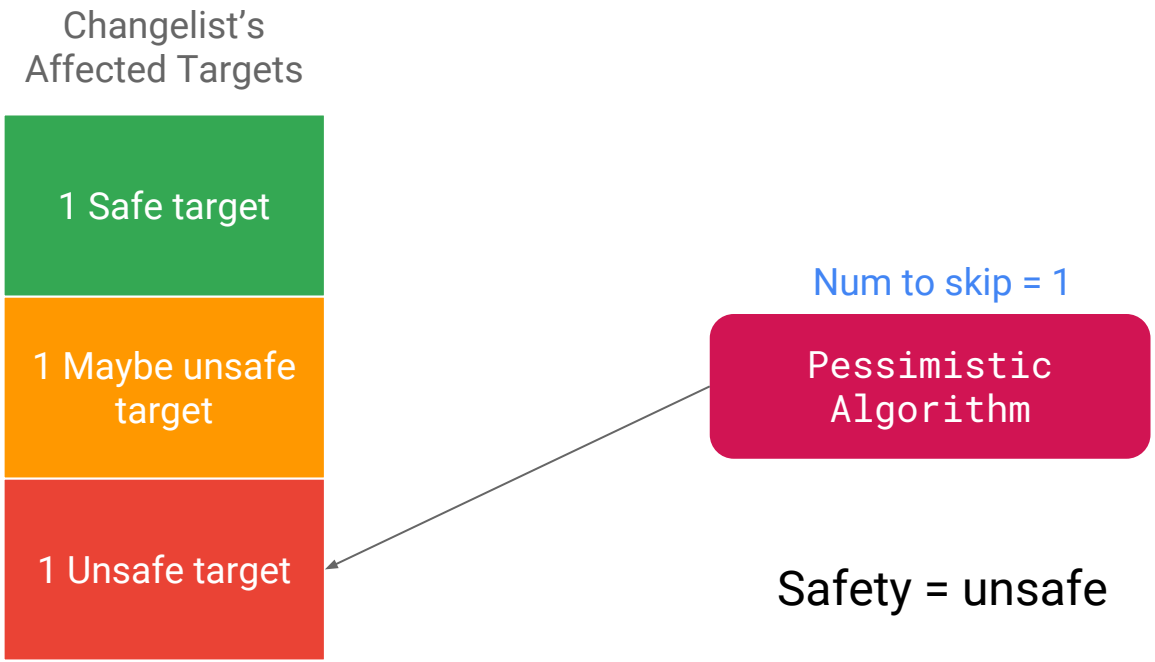
# Algorithms - **Optimistic**



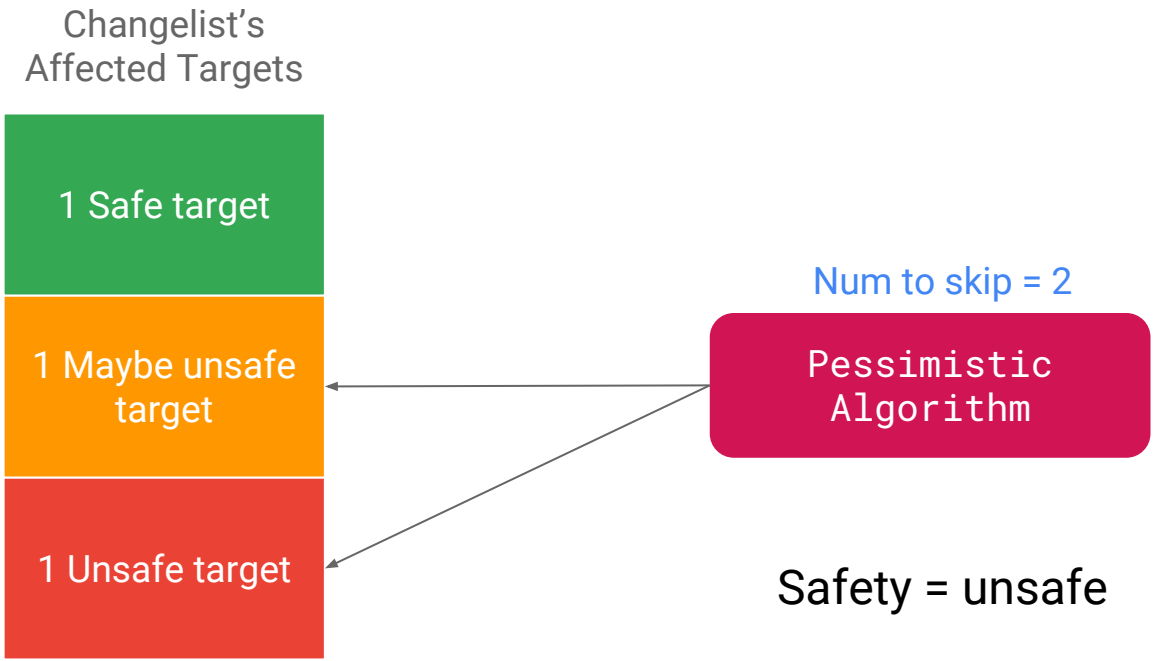
# Algorithms - **Pessimistic**



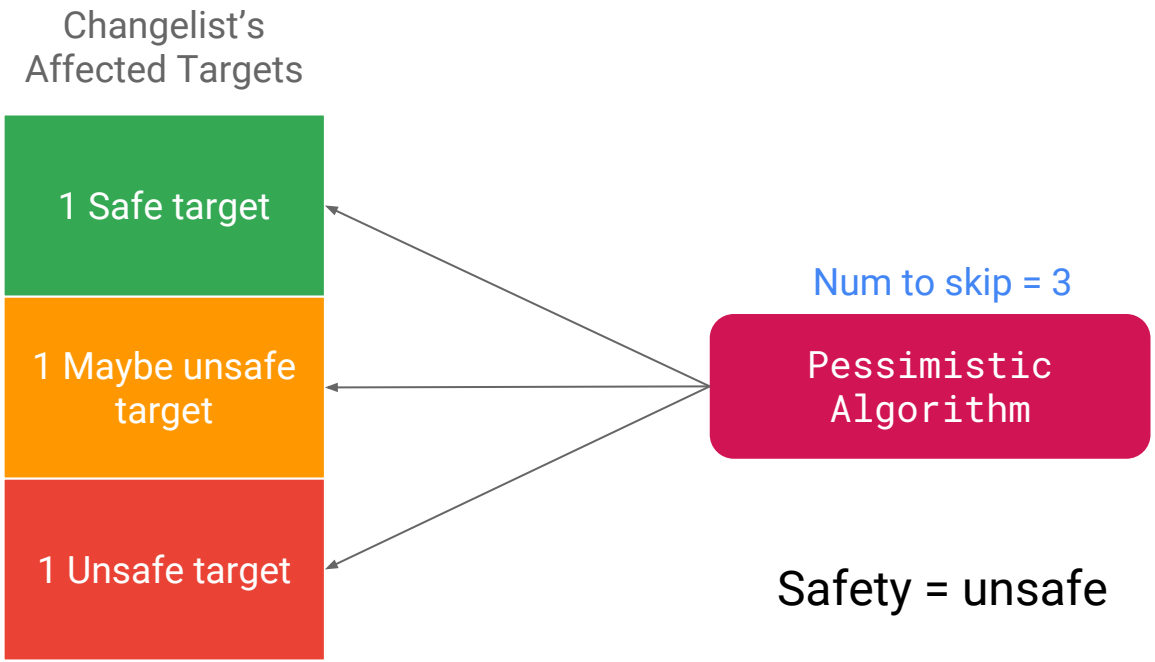
# Algorithms - **Pessimistic**



# Algorithms - **Pessimistic**



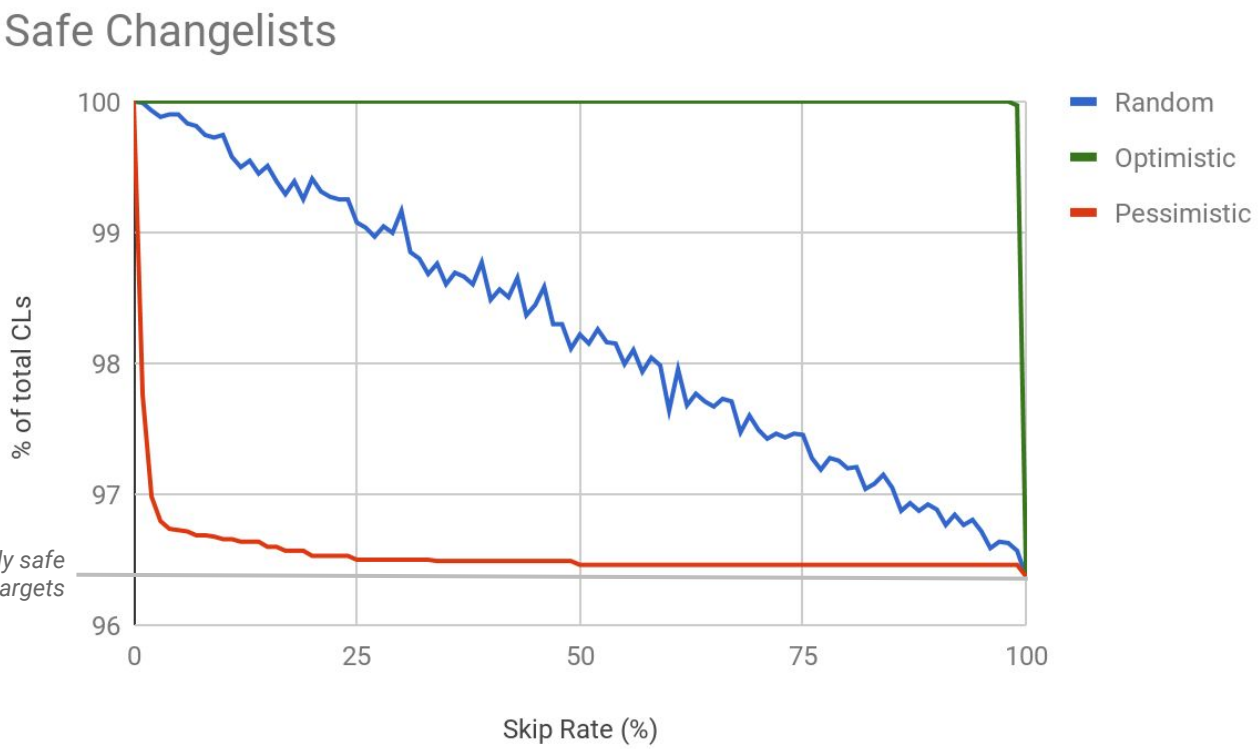
# Algorithms - **Pessimistic**



# Pipeline performance

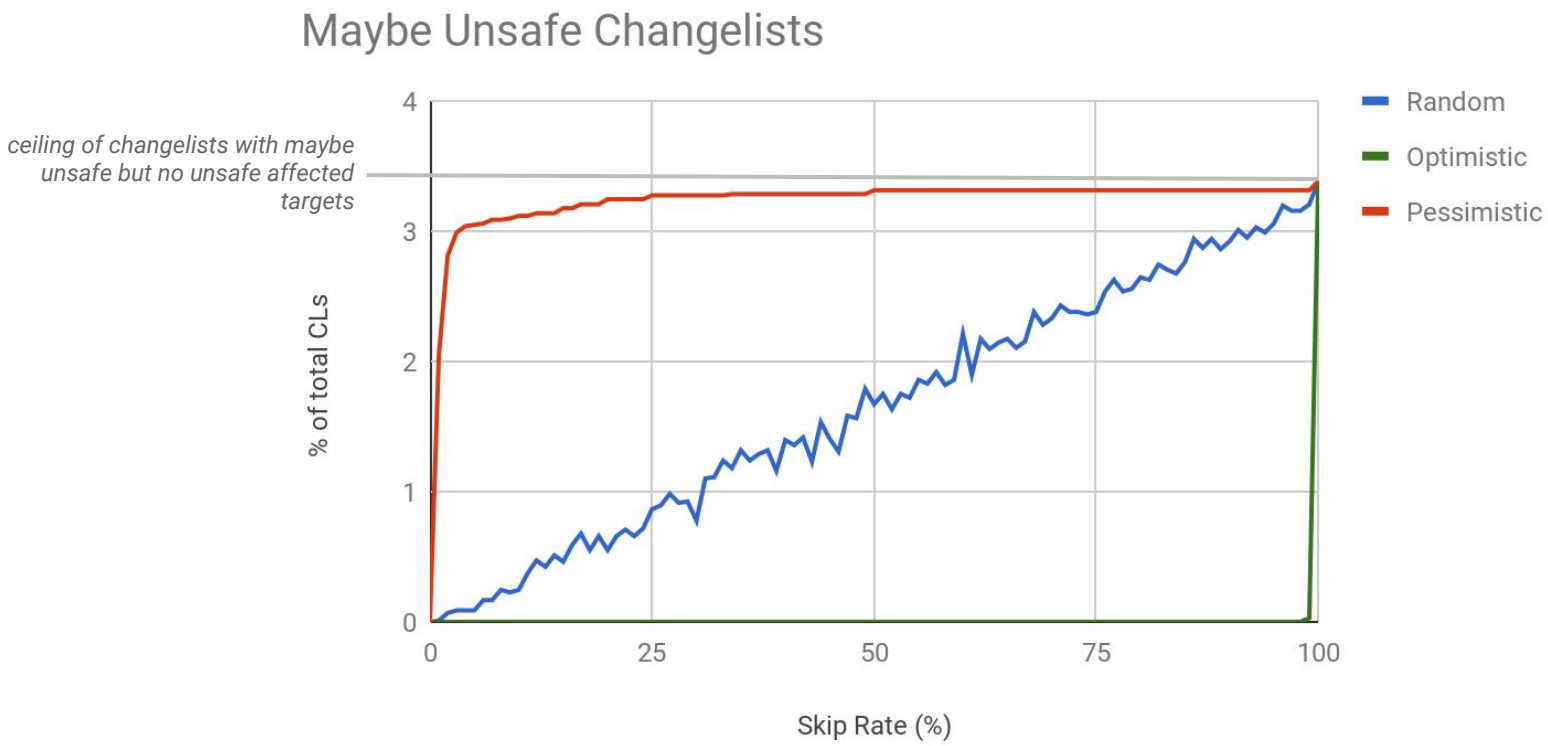
- Safety data builder ran in 35 mins
- Algorithm evaluator
  - Optimistic ran in 2h 40m
  - Pessimistic ran in 3h 5m
  - Random ran in 4h 40m

# Small dataset results

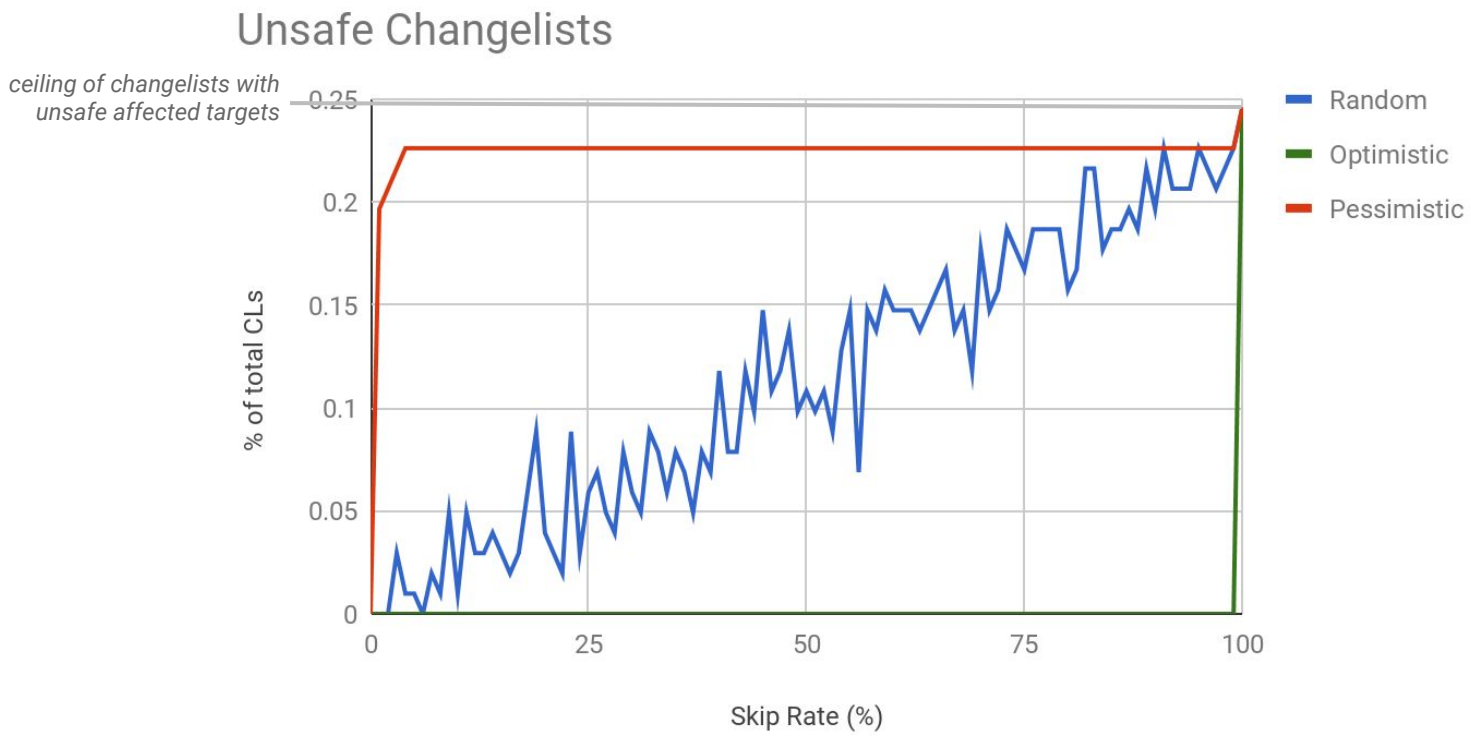



*floor of changelists with only safe affected targets*

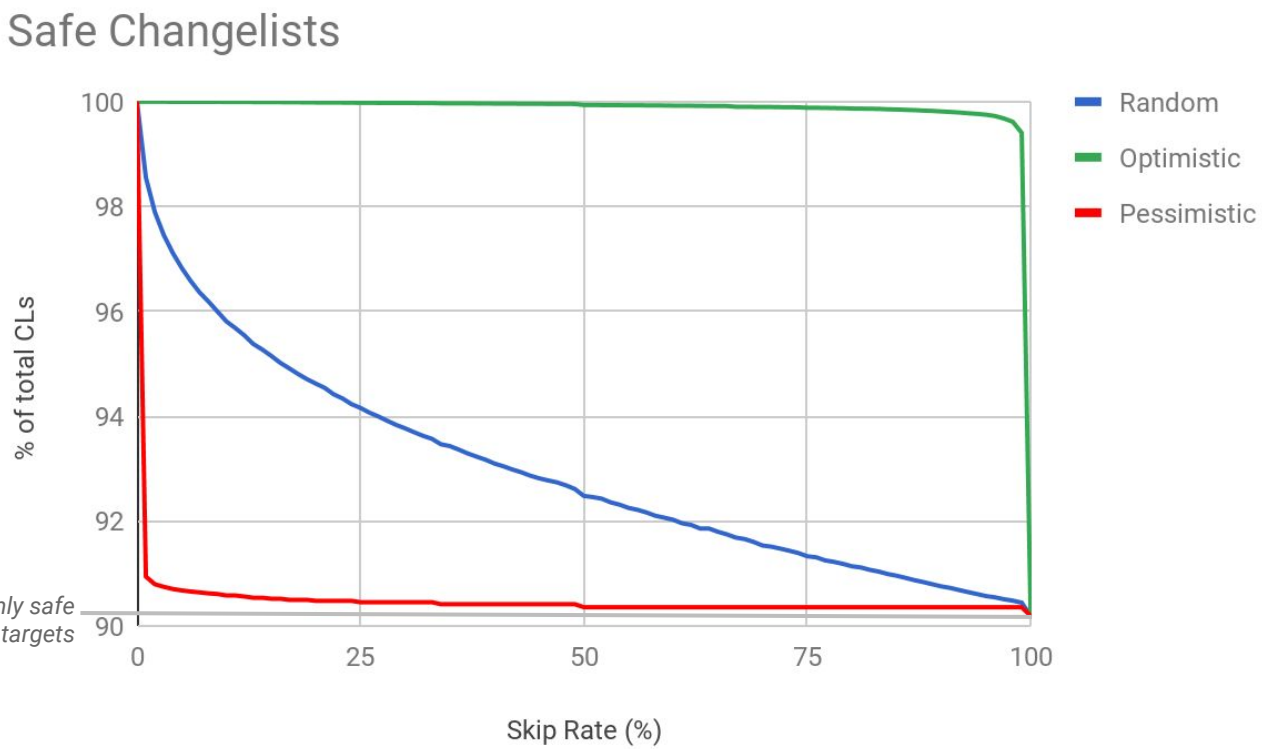
# Small dataset results



# Small dataset results



# Large dataset results



# Why is random a curve?

- Previously we had predicted a straight line for random
- Small data set has a straight line

# Probability Distribution

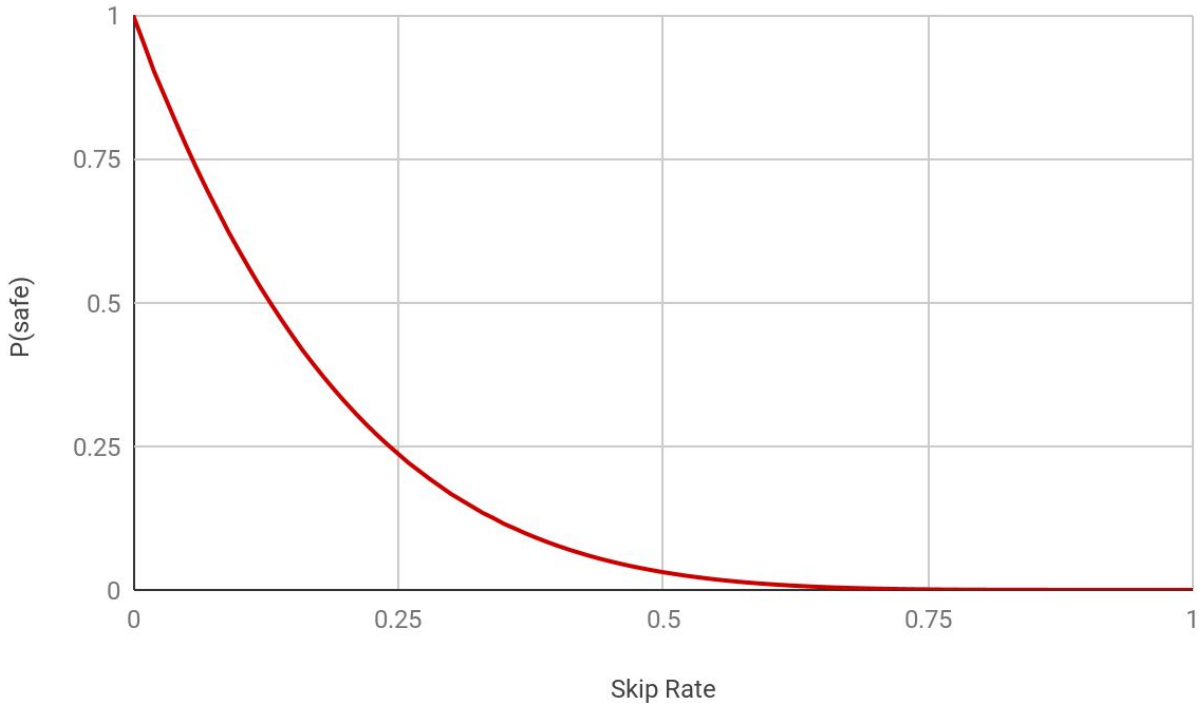
$$\binom{n}{k} = \text{number of ways to select k items from n total items}$$

$$\text{P(select k only safe targets)} = \frac{\text{number of ways to select k safe targets}}{\text{number of ways to select any k targets}}$$

$$= \frac{\binom{n}{Np}}{\binom{N}{Np}}$$

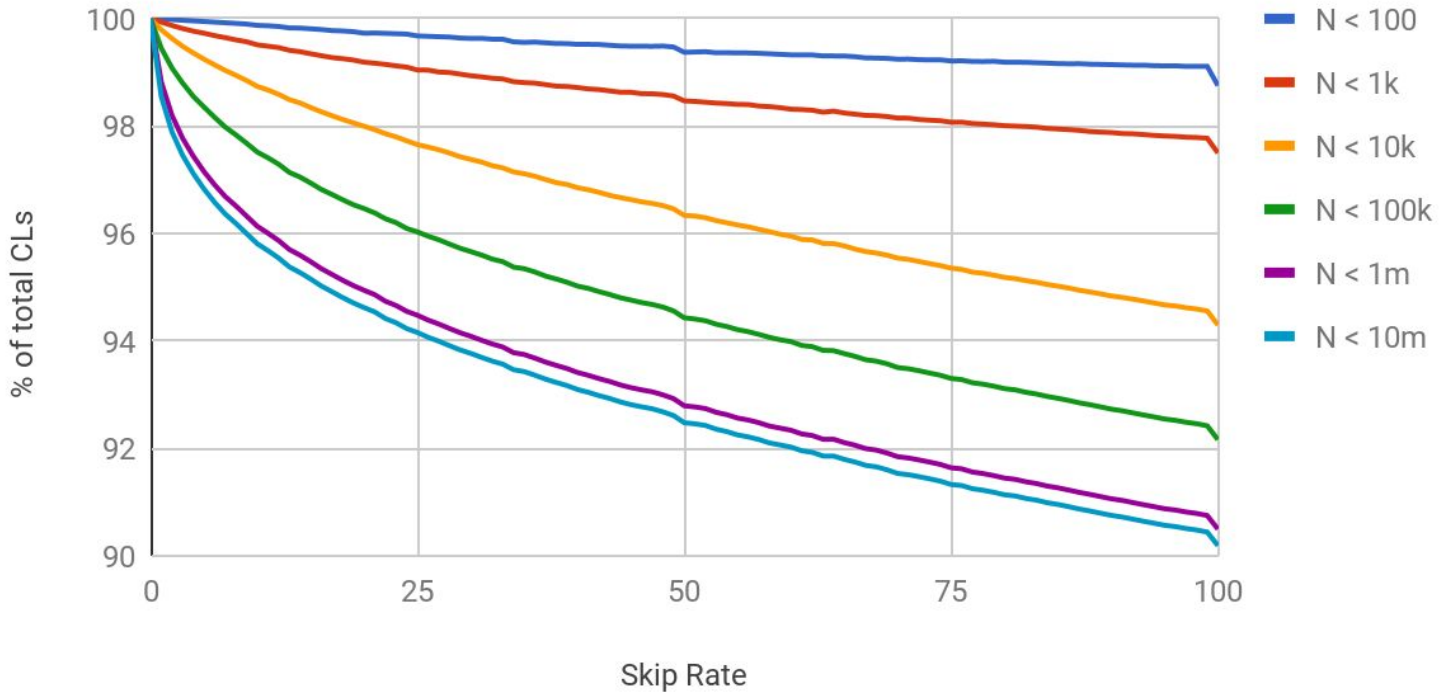
Where $n$ = number of safe affected targets
$N$ = total number of affected targets
$p$ = % of targets being selected

# Probability Distribution where N = 1000, n = 995



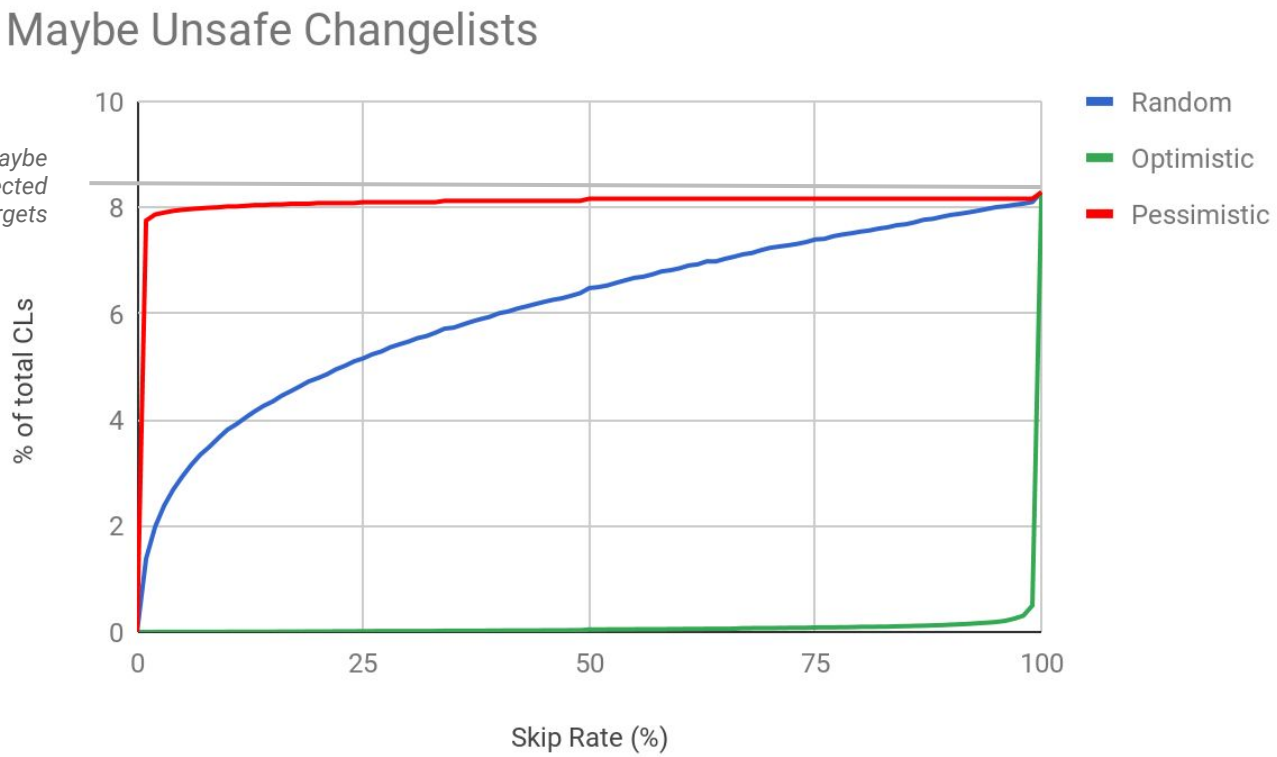
# Random Algorithm Safe Changelists

N = number of affected targets for a CL

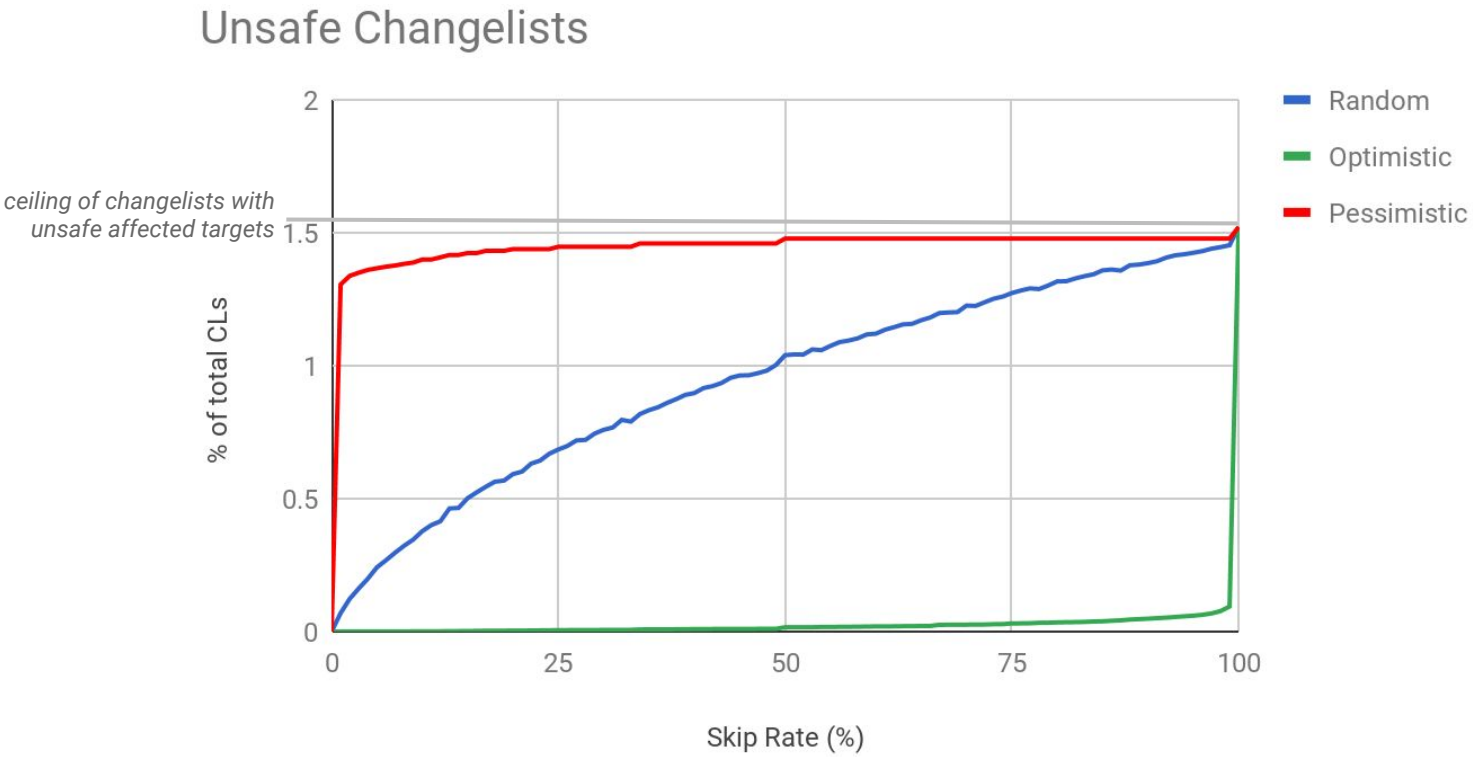


# Large dataset results

*ceiling of changelists with maybe unsafe but no unsafe affected targets*



# Large dataset results



# Conclusions

- The project was completed!
- We now have an offline method to evaluate test scheduling algorithms and a baseline for future comparison

# Continuing the project

- Better flake exclusion
  - Filter using ratio transitions:results
  - Find the point where Kellogs doesn’t identify the target as flaky
- Rerunning Elbaum experiments
  - An algorithm which prioritizes targets based on the number of transitions in some previous window of time
- Evaluating Efficacy machine learning model

# Questions?

# Creating safeties

```
for all targets {
     for (result in sorted target results) {
          if (result = affected) {
               add result to pending results
               continue;
          }

          if (previous result = result) {
               mark this result and all pending results as safe
          } else (if no pending results) {
               mark this result as unsafe
          } else {
               mark this result and all pending results as maybe safe
          }

          previous result = result;
     }
}
```

# Evaluating algorithms

```
for (changelist) {
     retrieve affected targets for changelist
     for (skip rate = 0..100%) {
          skipped targets = algorithm.skipTargets(affected targets, skip rate * num affected
          targets);

          if (skipped targets contain sunsafe targets) {
               mark this test selection as unsafe
          } else if (skipped targets contains maybe unsafe targets) {
               mark this test selection as maybe unsafe
          } else {
               mark this test selection as safe
          }
     }
}
```

# safety_record.proto

```
// Represents the safety information for all affected targets at a CL.
message SafetyRecord {
  optional int64 changelist = 1;
  repeated TargetSafety safe_targets = 2;
  repeated TargetSafety unsafe_targets = 3;
  repeated TargetSafety maybe_unsafe_targets = 4;
}

// Represents the safety of skipping a test target.
message TargetSafety {
  optional string target_name = 1;
  optional Safety safety = 2;
  optional Transition transition = 3;
}
```

# safety_result.proto

```
// Represents the safety information for using a test selection algorithm on a
// CL's test targets with a given target skip rate.
message SafetyResult {
  optional int64 changelist = 1;
  optional string algorithm_name = 2;
  optional int32 skip_rate = 3;

  optional tko.testselectionevaluation.TargetSafety.Safety safety = 4;

  repeated string unsafe_skipped = 5;
  repeated string maybe_unsafe_skipped = 6;

  optional int32 total_skipped = 7;
}
```

# Optimistic Algorithm Implementation

```
// Constructor takes a SafetyRecord as input
private final SafetyRecord record;

skipTargets(changelist, targetList, numToSkip) {
      remainingToSkip = numToSkip;

      skip targets from record.safe_targets with limit remainingToSkip
      remainingToSkip -= safe skipped targets

      skip targets from record.maybe_safe_targets with limit remainingToSkip
      remainingToSkip -= maybe unsafe skipped targets

      skip targets from record.unsafe_targets with limit remainingToSkip
      remainingToSkip -= unsafe skipped targets

      return skipped targets
}
```