UMass | Dartmouth UNIVERSITY OF MASSACHUSETTS DARTMOUTH

# Free and Open Source Tools to Assess Software Reliability and Security


Vidhyashree Nagaraju, Venkateswaran Shekar, Thierry Wandji[2] and Lance Fiondella[1]

[1]*University of Massachusetts*, North Dartmouth, MA 02747

[2]*Naval Air Systems Command*, Patuxent River, MD 20670

# Questions?

# Outline

- Year I deliverables summary
- Guidance
- Software Failure and Reliability Assessment Tool (SFRAT)
  - Architecture
  - Review of  Year I functionality
  - Year II functionality
- Software Defect Estimation Tool (SweET)
- Goals

# State of software reliability

- Software reliability studied for 50+ years
  - Methods have not gained widespread use
    - Disconnect between research and practice
- Diverse set of stakeholders
  - Reliability engineers
    - May lack software development experience
  - Software engineers
    - May be unfamiliar with methods to predict software reliability

# YEAR I (3/15-2/16)
# DELIVERABLE SUMMARY

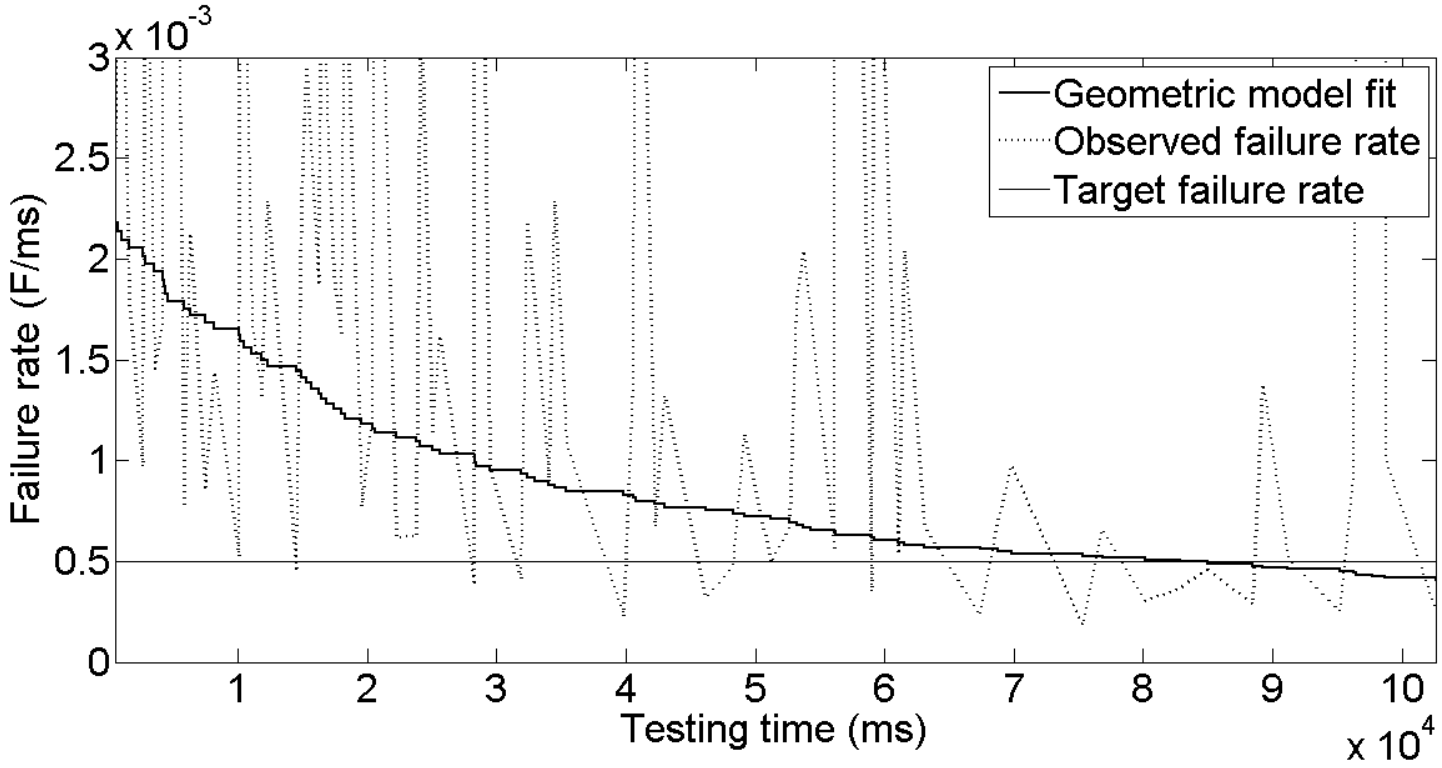
# Summary of Year I deliverables

- Implemented open source software reliability tool
  - Data conversion routines
  - Trend tests for reliability growth
  - Two failure rate models
    - Assume failure rate decreases as faults detected and removed
  - Three failure count models
    - Count faults detected as function of time
  - Tested on dozens of data sets
  - Two goodness of fit measures

# Estimates enabled by software reliability models

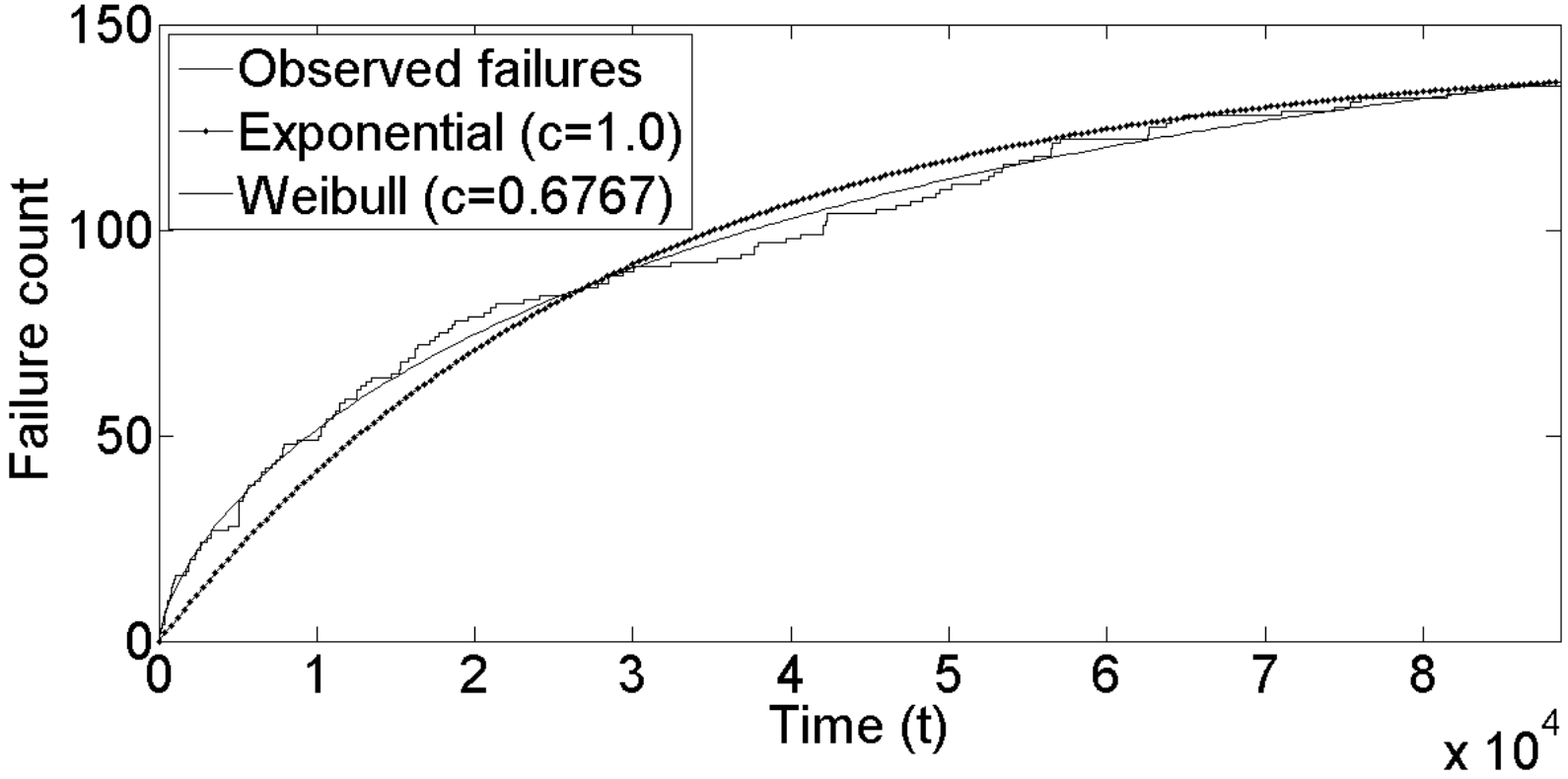
- Number of
  - Faults detected with additional testing
  - Remaining faults
- Mean time to failure (MTTF) of next fault
  - Testing time needed to remove next $k$ faults
- Probability software does not fail before completion of fixed duration mission

# Failure rate model




Model characterizes decreasing trend in failure rate

# Failure time/count models




Model characterizes fault discovery process

UMass | Dartmouth UNIVERSITY OF MASSACHUSETTS DARTMOUTH

# sasdlc.org/lab/projects/srt.html



# Software Failure and Reliability Assessment Tool (SFRAT)

## Description

The key to the success of all software is its reliability. The Software Failure and Reliability Assessment Tool (SFRAT) is an open source application to estimate and predict the reliability of a software system during test and operation. It allows users to answer the following questions about a software system during test:

1. Is the software ready to release (has it achieved a specified reliability goal)?
2. How much more time and test effort will be required to achieve a specified goal?
3. What will be the consequences to the system's operational reliability if not enough testing resources are available?

SFRAT runs under the R statistical programming framework and can be used on computers running Windows, Mac OS X, or Linux

## Resources

**WARNING: Web instance is for demonstration only. Please do not upload sensitive data to the site**

Web instance
Example failure data sets
SFRAT Github repository
User's Guide
Contributor's Guide

## Publications

Search:

| Year | Type | Publication |
|---|---|---|

# GUIDANCE

# Software Reliability Growth Modeling

- No single model characterizes all data sets best
- Models supplementary mathematical guidepost
  - Used in conjunction with SDLC activities to identify, implement, and test functional requirements
- Do not prescribe a single model
- Learn to track before planning in SEPs & TEMPs
- Emphasize
  - Effective communication between system, reliability, and software engineers
  - Frequent use of quantitative SRGM throughout DT and OT to assess progress toward software and system reliability goals

# Software Reliability Growth Tracking

- For reliability growth tracking to be effective
  - Failures and their severity must be clearly defined
  - Impact on mission and end-to-end capability in order to produce data suitable for reliability growth tracking
  - Will be impacted by updates to interacting subsystems including hardware, mechanical, sensing, and operator usage

# Data formats

- Based on data formats
  - Failure Rate models
    - Inter-failure times - time between $(i-1)^{st}$ and $i^{th}$ failure, defined as $t_i = (\mathbf{T}_i - \mathbf{T}_{i-1})$
    - Failure times – vector of failure times,
      $$\mathbf{T} =< t_1, t_2, \ldots, t_n >$$
  - Failure Counting models
    - Failure count data - length of the interval and number failures observed within it,

    $< \mathbf{T}, \mathbf{K} >=< (t_1, k_1), (t_2, k_2), \ldots, (t_n, k_n) >$
  - Possible to use change requests during DT

# Data quality

- Accuracy
  - Critically depends on availability of failure data
  - Inaccurate records of time make model fitting and prediction difficult
- Even when data available
  - Practitioner must know how to filter and organize data for use in models
    - Filter to exclude: non-software issues, duplicate failures, etc…

# SOFTWARE FAILURE AND RELIABILITY ASSESSMENT TOOL (SFRAT)

# ARCHITECTURE

# SFRAT user modes

- Graphical user interface
  - Web and intranet
- Developer mode
  - Incorporate additional models
- Power user
  - Incorporate into internal software testing processes
- Benefits
  - Can help contractors, FFRDCs, and government quantitatively assess software as part of data collection, reporting, and oversight

# SFRAT – File structure

install_script.R

server.R

ui.R

**utility**

- Data
  - a.Data_Tools.R
- Metrics
  - a.GOF.R
- Plots
  - a.PlotModelResults.R
  - b.Plot_Raw_Data.R
  - c.Plot_Trend_Tests.R
- Prediction
  - a.Detailed_prediction.R
- tables
  - a.DataAndTrendTables.R
  - b.ModelResultTable.R
- RunModels.R

**trend_tests**

1. Laplace_trend_test.R
2. RAA.R

**models**

- GO
- DSS
- Wei
- JM
- GM

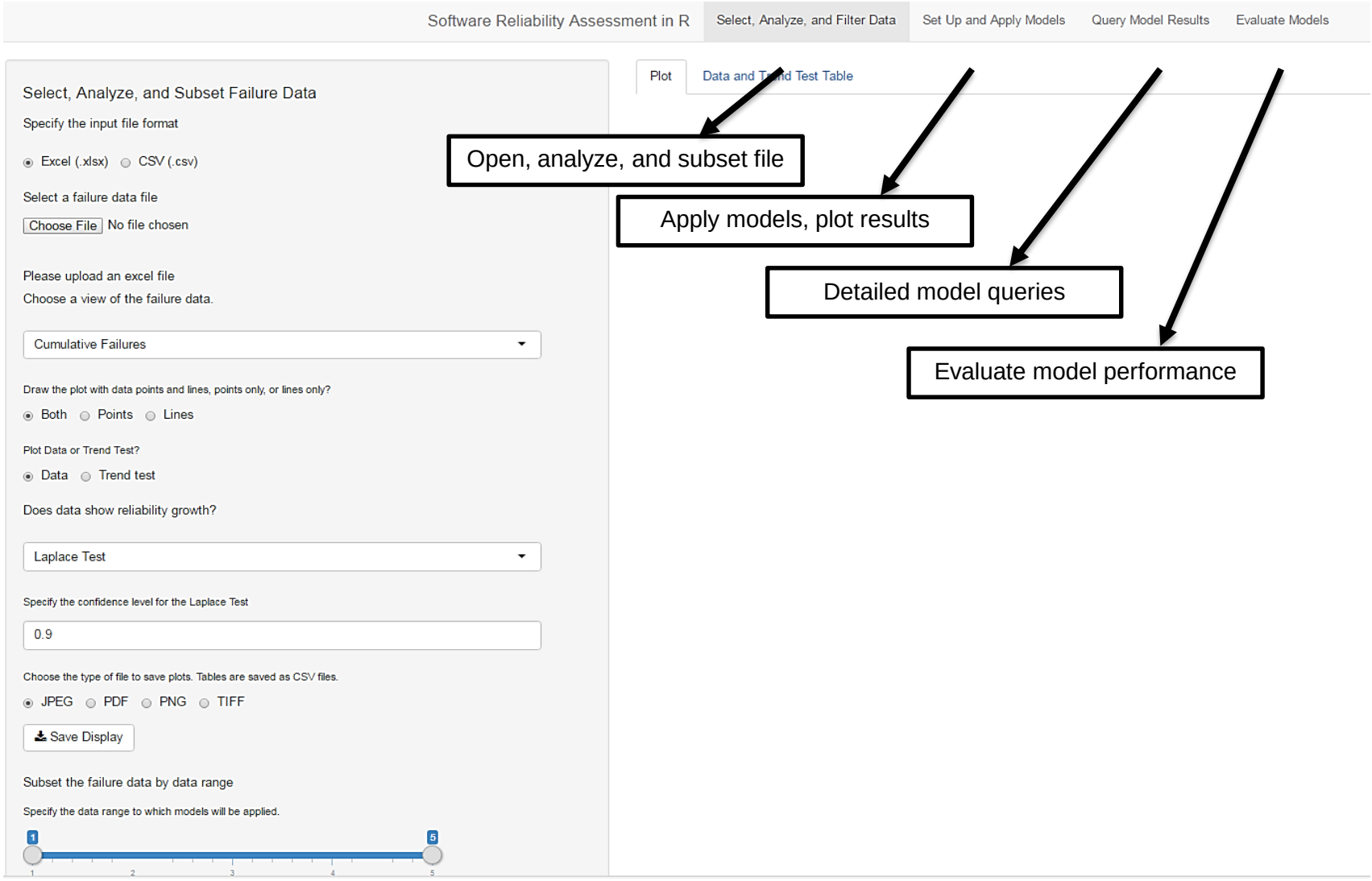
New models added in the “models” folder

# Power user mode

- Code can be tailored for internal use
  - Build into existing automated software testing procedures to provide near real-time feedback of reliability trends
  - Many industry standard programming languages can call R functions
    - Visual Basic, Java, C/C#/C++, and Fortran
    - Ensures tool will integrate smoothly

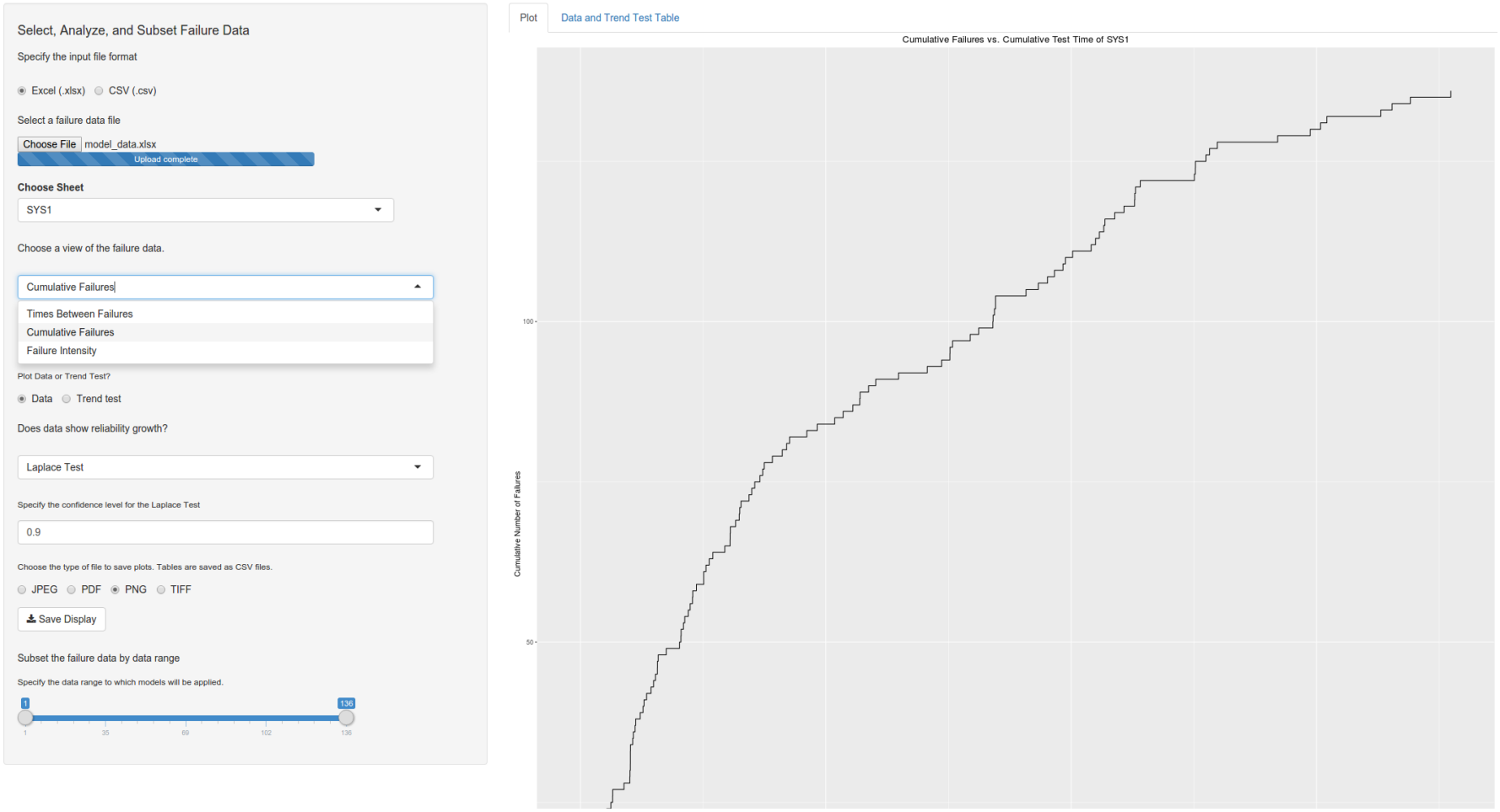
# REVIEW OF YEAR I FUNCTIONALITY

# SFRAT - Tab view



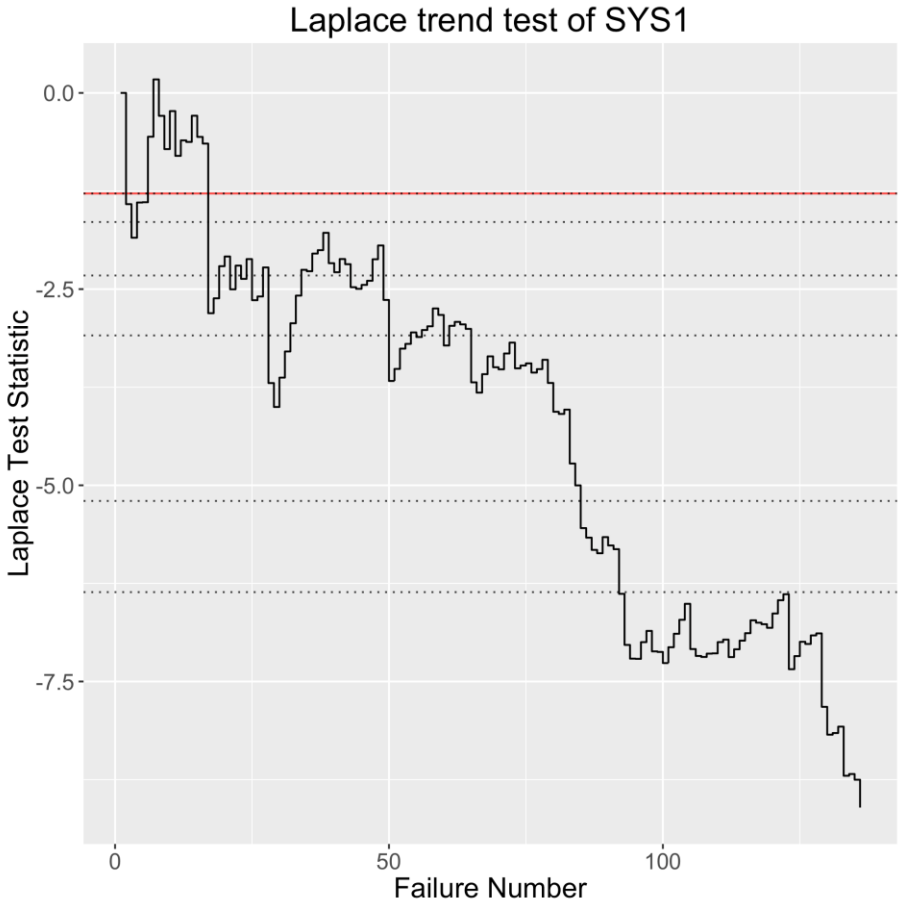
# Tab 1
# Select, Analyze, and Filter data

# Tab 1 – After data upload



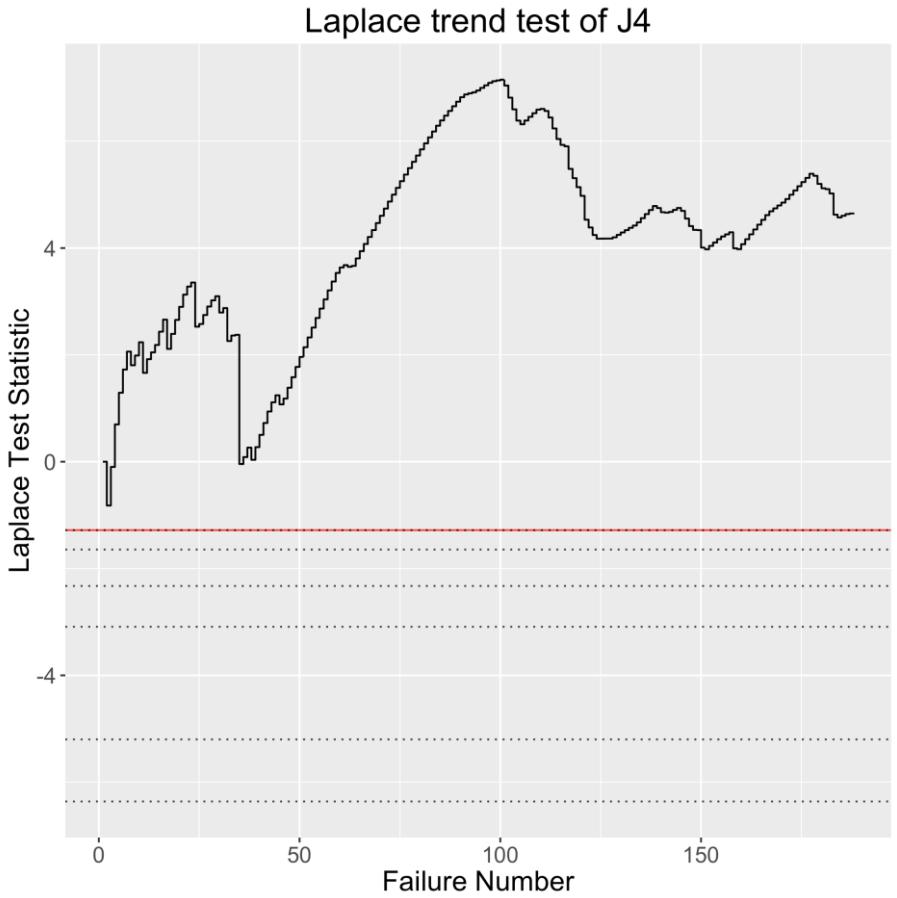

Cumulative failure data view

# Laplace trend test – SYS1 data



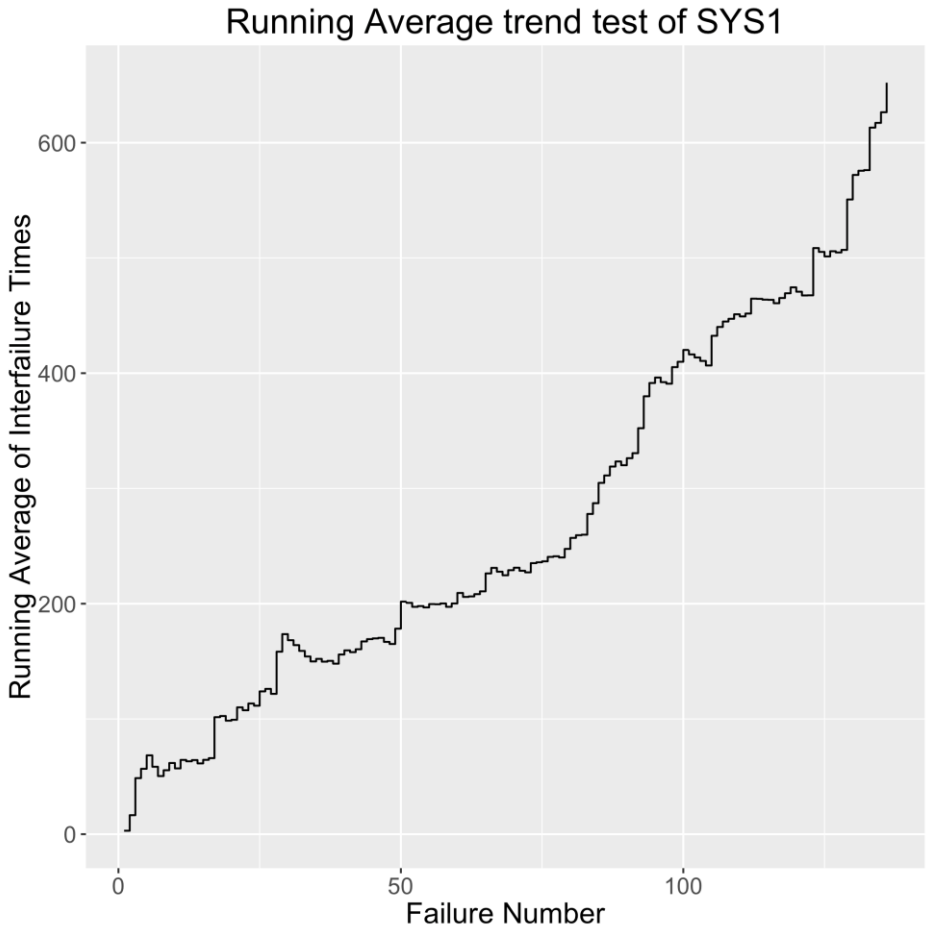

Decreasing trend indicates reliability growth
(Indicates application of SRGM appropriate)

# Laplace trend test – J4 data




Does not exhibit reliability growth
(Indicates additional testing required)

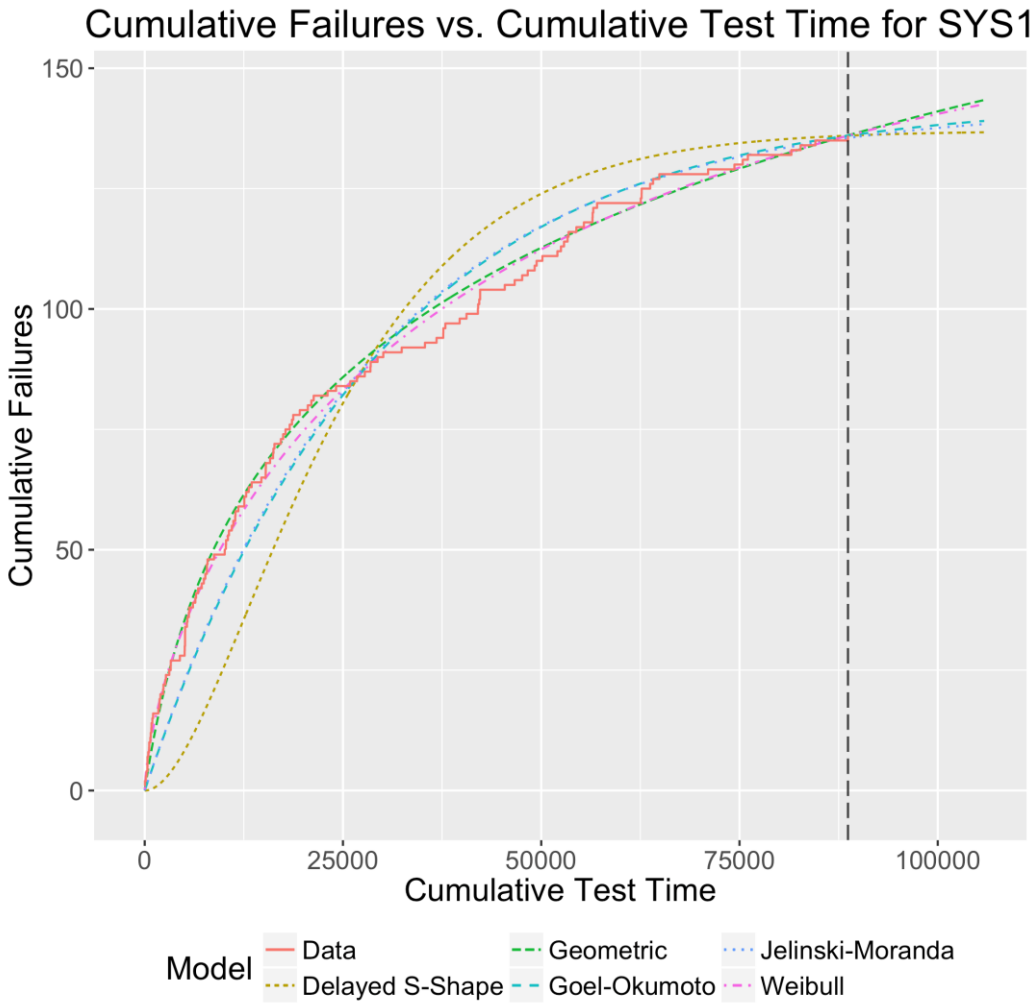
# Running Arithmetic Average – SYS1 data




Increasing trend indicates reliability growth

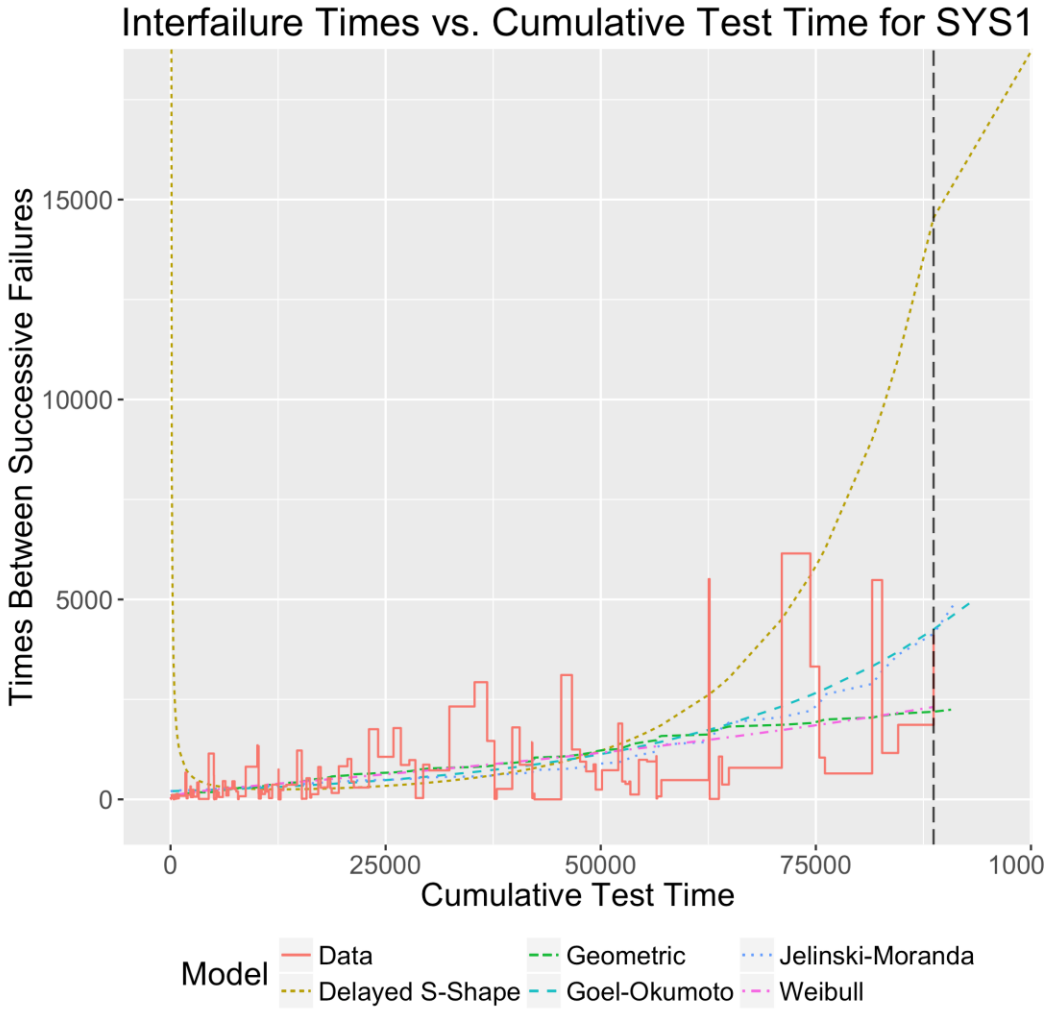
# Tab 2
# Set Up and Apply Models

# Cumulative failures



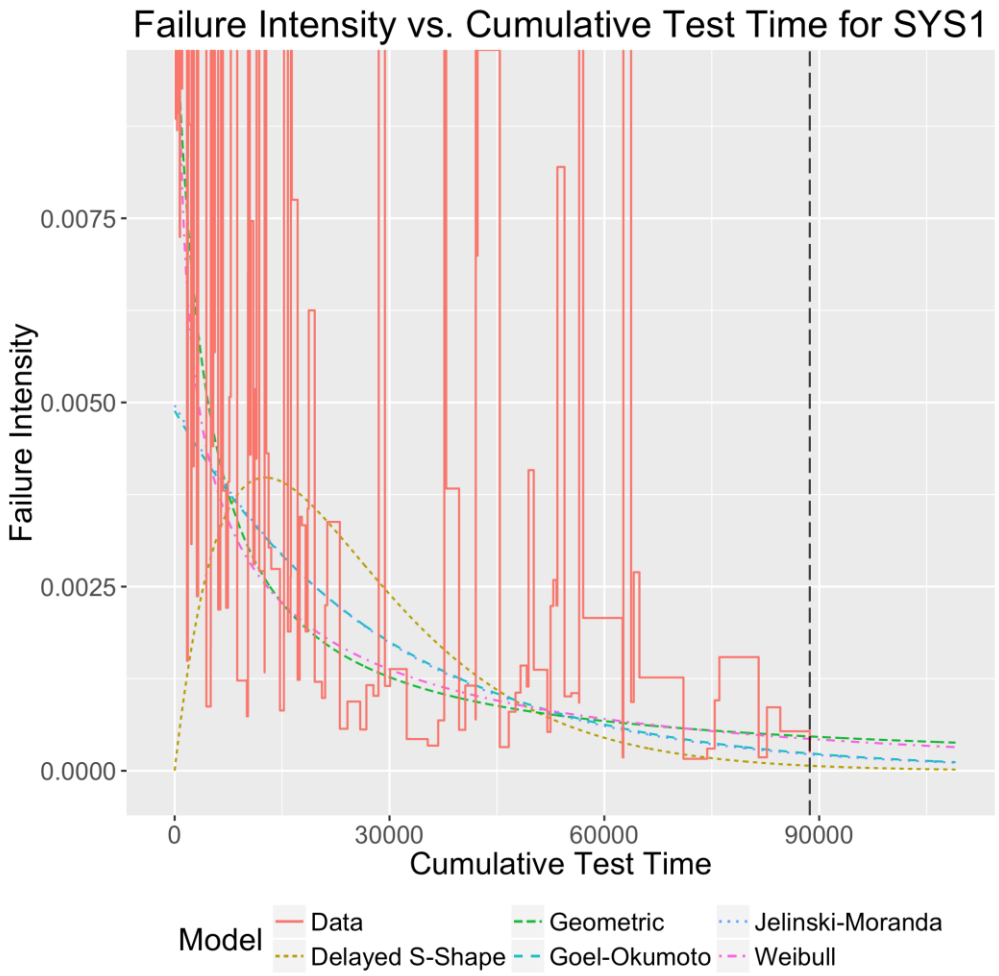

Plot enables comparison of data and model fits

# Time between failures



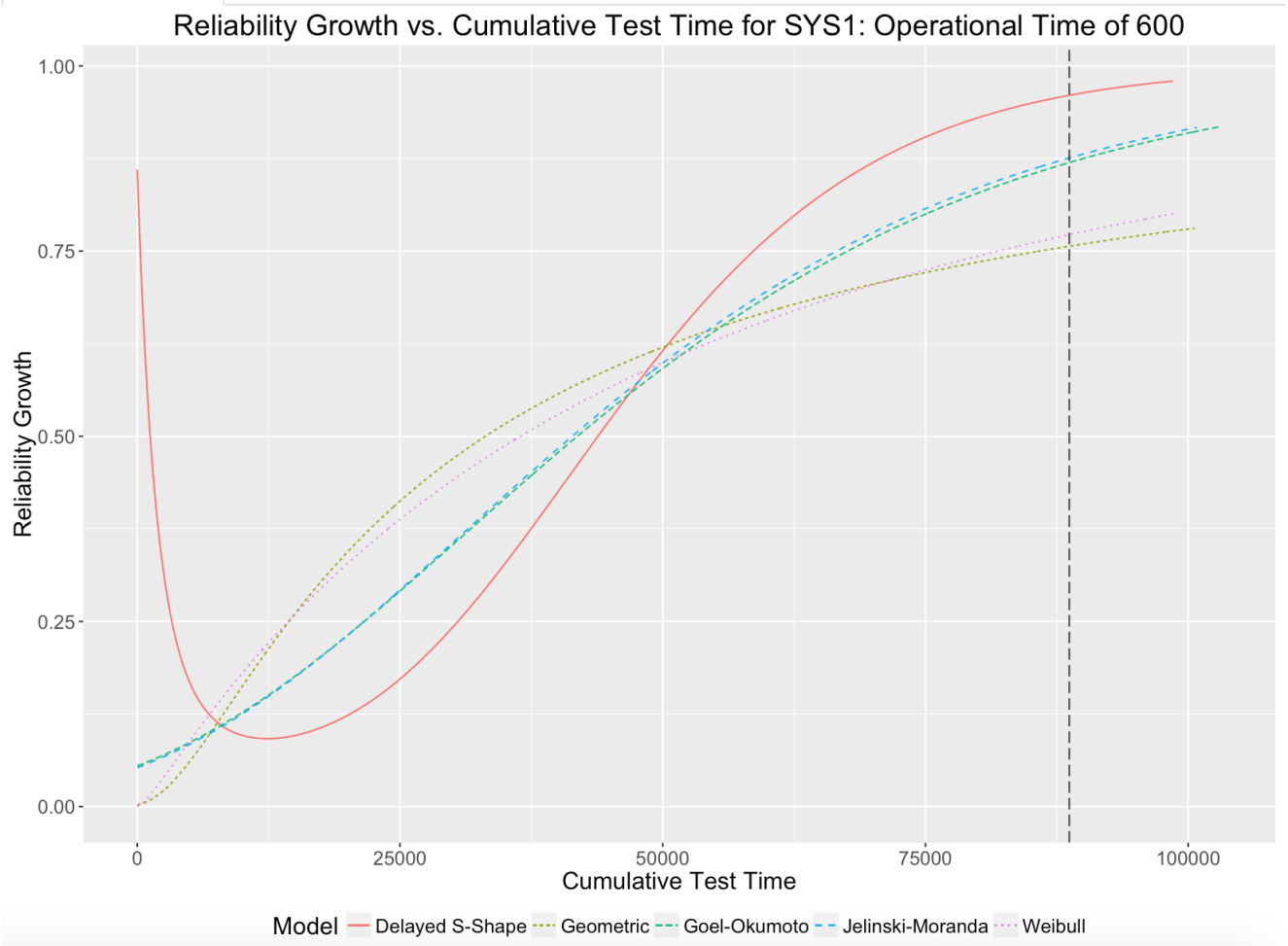

Times between failures should increase (indicates reliability growth)

# Failure intensity




Failure intensity should decrease (indicates reliability growth)

# Reliability growth curve




Can determine time to achieve target reliability

# Tab 3
# Query Model Results

# Failure Predictions

| | Model | Time to achieve R = 0.9 for mission of length 4116 | Expected # of failures for next 4116 time units | Nth failure | Expected times to next 1 failures |
|---|---|---|---|---|---|
| | All | All | All | All | All |
| 1 | Delayed S-Shape | 12401.1541529981 | 0.2468563 | 1 | NA |
| 2 | Geometric | 1592716.45936287 | 1.8774731 | 1 | 2170.03088926781 |
| 3 | Goel-Okumoto | 62829.7672027733 | 0.9036154 | 1 | 4591.28466949961 |
| 4 | Jelinski-Moranda | 59915.2917457156 | 0.8561255 | 1 | 4869.80650205625 |
| 5 | Weibull | 259865.770847692 | 1.7259537 | 1 | 2353.05254648438 |

Showing 1 to 5 of 5 entries

Previous 1 Next

Can identify potential schedule overruns

# Tab 4
# Evaluate Models

# AIC and PSSE

| | Model | AIC | PSSE |
|---|---|---|---|
| | All | All | All |
| 1 | Delayed S-Shape | 2075.146 | 296.34925 |
| 2 | Geometric | 1937.034 | 84.32708 |
| 3 | Goel-Okumoto | 1953.613 | 23.07129 |
| 4 | Jelinski-Moranda | 1950.534 | 19.60037 |
| 5 | Weibull | 1938.161 | 74.94496 |

Showing 1 to 5 of 5 entries

Previous 1 Next

Lower values preferred

# YEAR II (7/16-7/17) SFRAT FUNCTIONALITY

- Upper and lower confidence limits
  - Graphical and tabular values
- Model Evaluation Criteria
  - Prequential likelihood (PL) ratio
    - Identify model more likely to produce accurate estimates
      - Higher preferred
  - Model bias (MB) and MB trend
    - Indicate whether model over/underestimates times between failures
- Optimal release

~/srt.core - Shiny

http://127.0.0.1:4792 | Open in Browser | Publish

Software Reliability Assessment in R

Select, Analyze, and Filter Data | Set Up and Apply Models | Query Model Results | Evaluate Models

## Configure and Apply Models

**Specify the number of failures for which the models will make predictions**

**Specify the last data point for the initial parameter estimation interval.**

1 68 135

1 15 29 43 57 71 85 99 113 127 135

**Specify a confidence interval for parameter estimates.**

0.95

**Specify for how many failures into the future the models will predict**

1

**Choose one or more models to run, or exclude one or more models.**

Geometric | Goel-Okumoto | Jelinski-Moranda

Run Selected Models

## Display Model Results

**Choose one or more sets of model results to display.**

Goel-Okumoto

Model Result Plot | Model Result Table

**Show most likely parameter values or confidence bounds.** Low | Most Likely | High

~/srt.core - Shiny

http://127.0.0.1:4792 | Open in Browser | Publish

Software Reliability Assessment in R

Select, Analyze, and Filter Data | Set Up and Apply Models | Query Model Results | Evaluate Models

**Evaluate Model goodness of fit and Applicability**

**Choose one or more models for which the results will be evaluated.**

**Choose one or more sets of model results**

Geometric Goel-Okumoto Jelinski-Moranda

**Specify the Percent Data for PSSE**

0.9

**Draw the plot with data points and lines, points only, or lines only?**

(•) Both ( ) Points ( ) Lines

**Save model evaluations as PDF or CSV?**

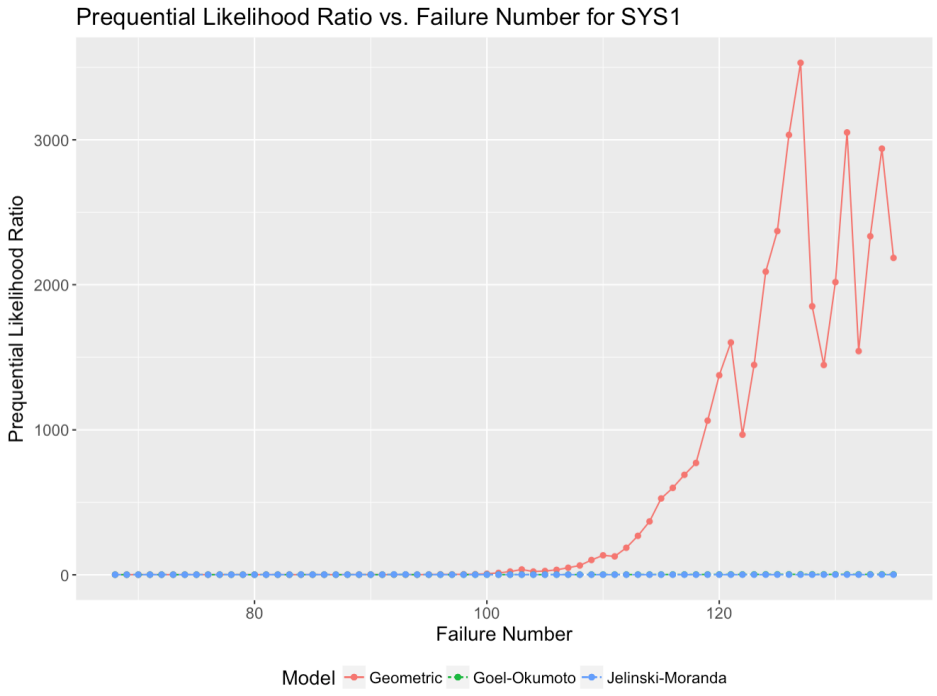
(•) CSV ( ) PDF

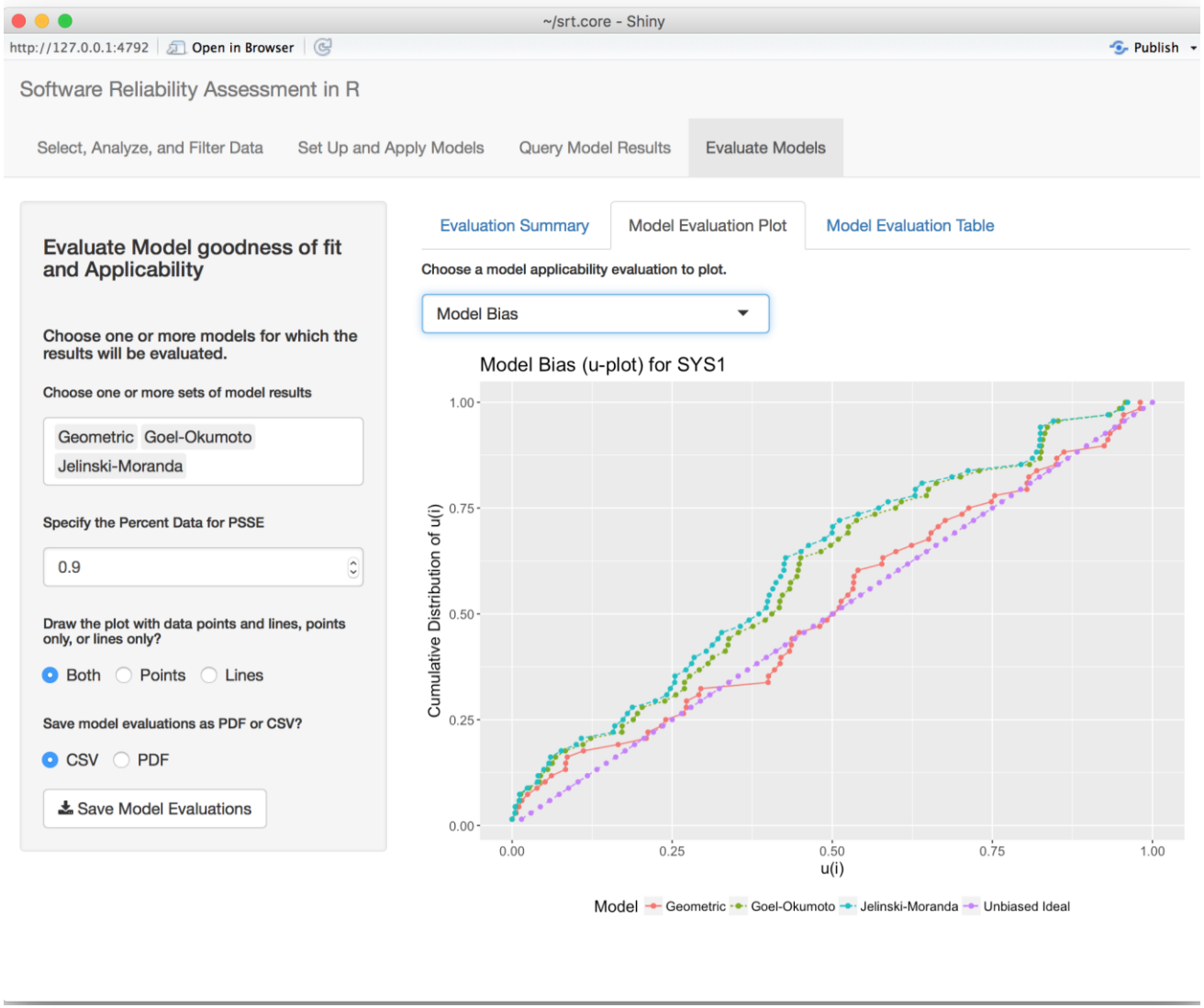
Save Model Evaluations

Evaluation Summary | Model Evaluation Plot | Model Evaluation Table

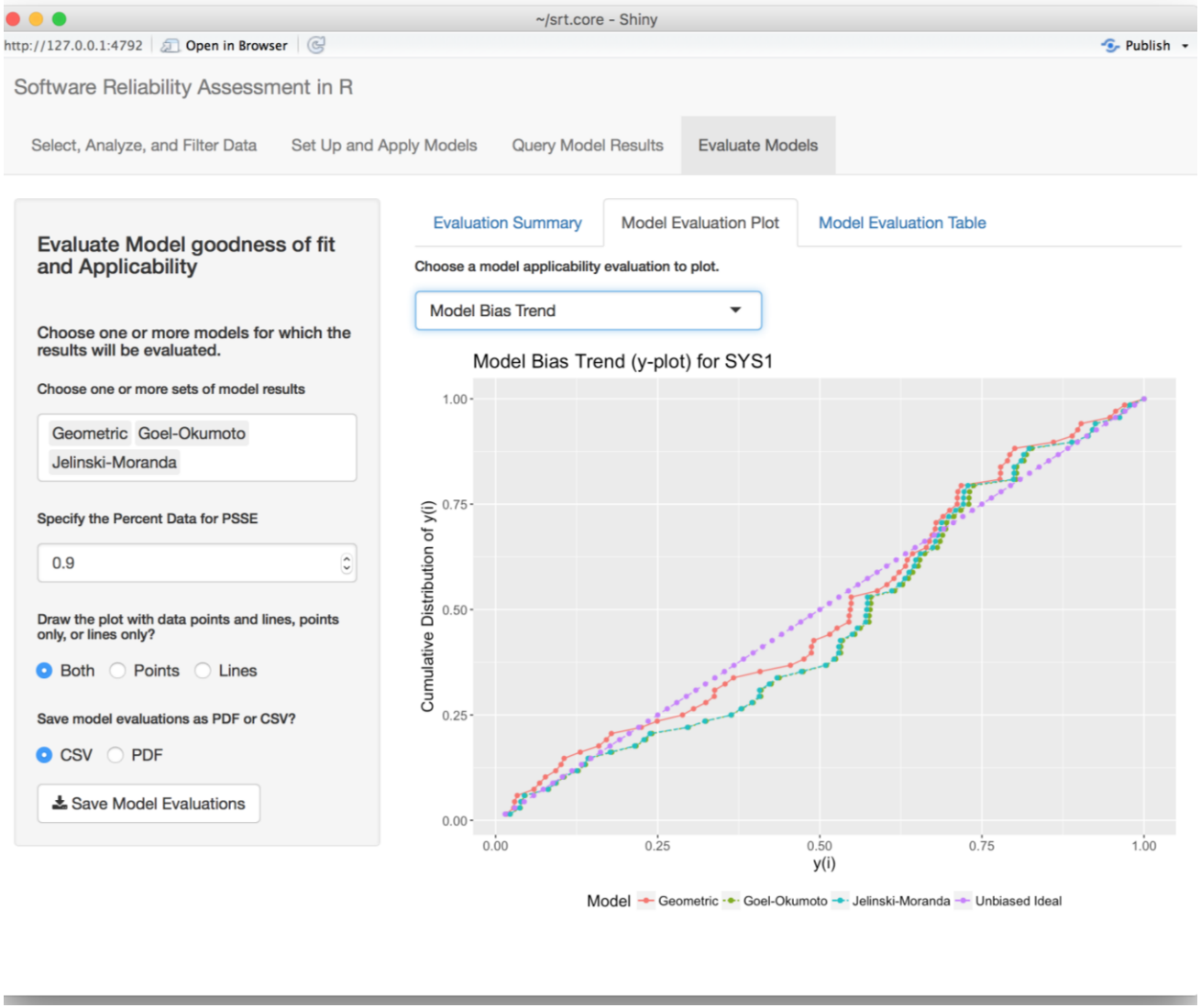
**Choose a model applicability evaluation to plot.**

Prequential Likelihood Ratio

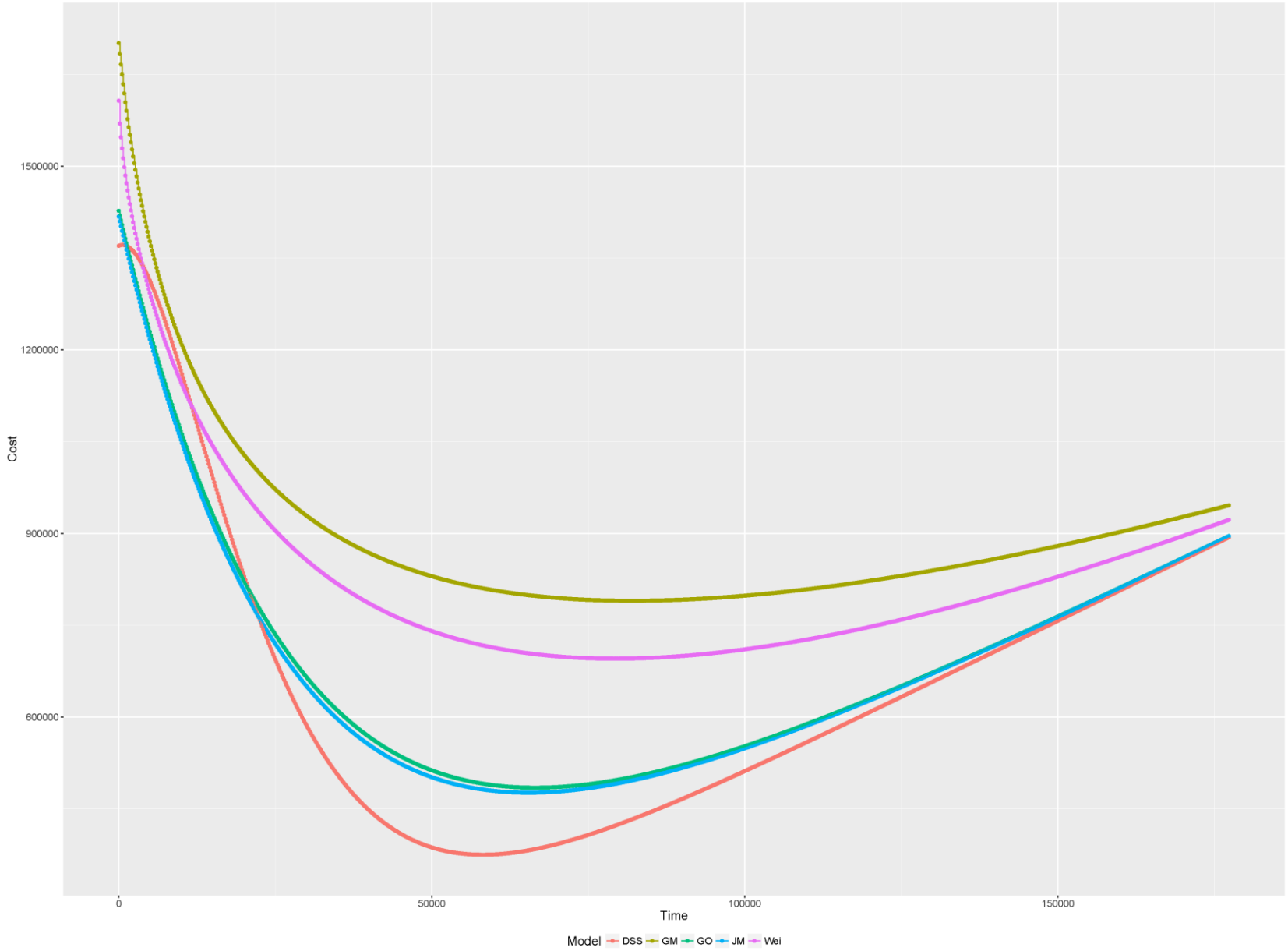





Models above line estimate more frequent times between failures than those observed




Models below line estimate more frequent times between failures than those observed


1500000
1200000
900000
600000
Cost
0
50000
100000
150000
Time
Model
DSS
GM
GO
JM
Wei

# SOFTWARE DEFECT ESTIMATION TOOL (SWEET)

# SWEEP (Software Error Estimation Program)

- Implemented four modes
  1. Time-based model
     - Estimates and tracks errors during system test and integration cycle
  2. Phase-based model
     - Provides defect information before running any code
  3. Planning aid
     - Generates an error discovery profile based on historical data
  4. Defect injection model
     - Allows user to understand probable defect injection profile

# Software Defect Estimation Tool (SweET)

## Description

The Software Defect Estimation Tool (SweET) is an open source application to track error identification and removal efforts during the software development lifecycle. SwEET is a free and open source version of the SoftWare Error Estimation Program (SWEEP) and SweET uses Weibull software reliability growth model utilizing Expectation Conditional Maximization algorithm to ensure stability and performance of the model fitting process. SweET simplifies four models of SWEEP into three modes:

1. **Mode A**: Time-based model: Estimates and tracks errors during system test and integration cycles.
2. **Mode B**: Phase-based and planning aid model: Predict and track defects for multiple phases and can provide defect information before running any code, whereas the planning aid model generates an error discovery profile based on the phase based historical data to help a software prohect achieve its objectives.
3. **Mode C**: Defect injection model: Allows the user to understand the probable defect injection profile and resulting efficiency and effectiveness of the verification process.

SweET runs under the Python 3.x programming framework and can be used on computers running Windows, Mac OS X, or Linux

## Resources

Example data sets
SweET Github repository
User's Guide (In preparation)

# GOALS

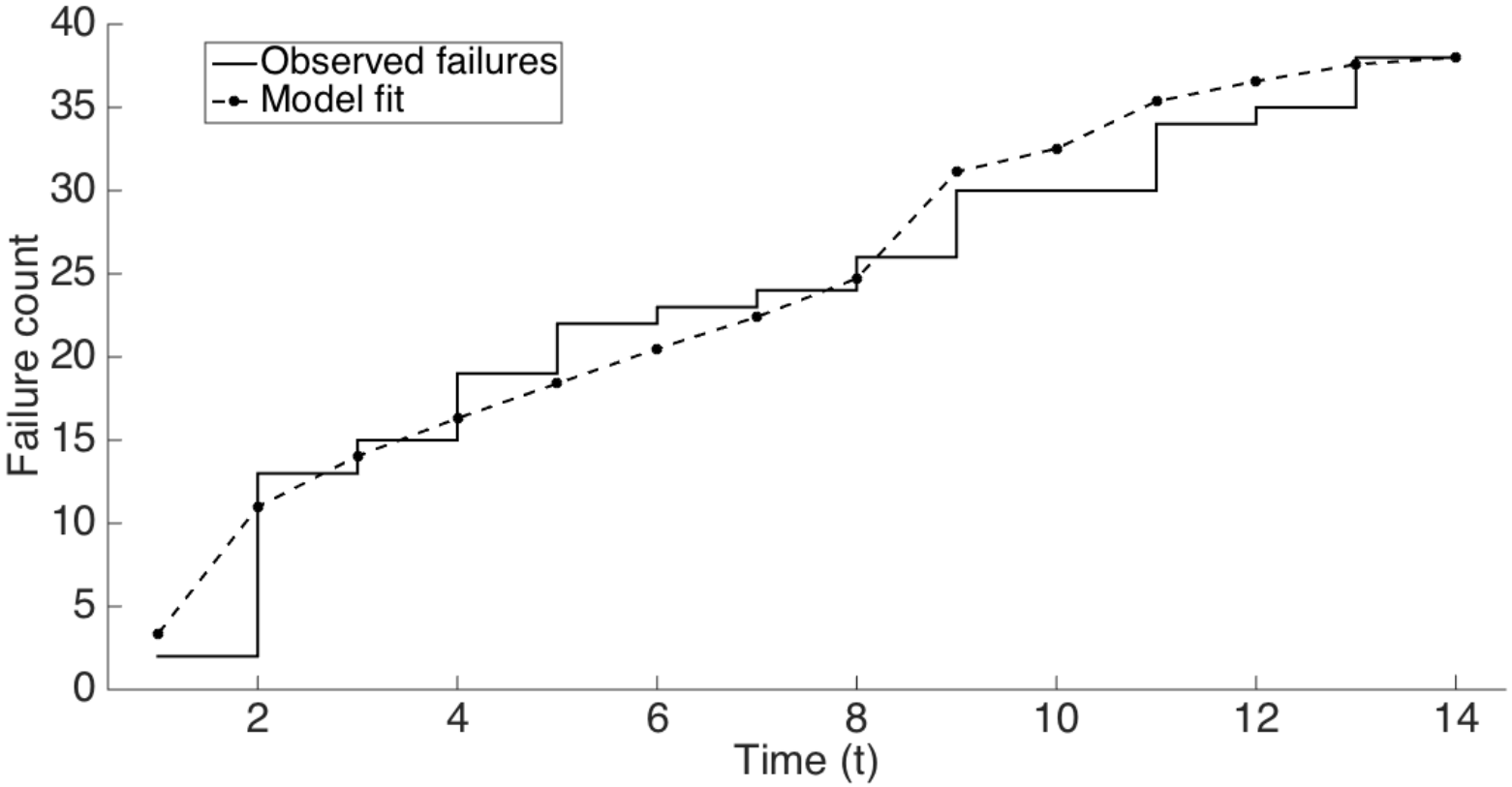
# Activities

- Update documentation
- Outreach, education, and training
  - Visit DoD labs and listen to practical concerns underlying modeling requirements
  - Work with existing users
- Coordinate contributions from developers
  - Failure severity decomposition
  - Software readiness metrics
  - Additional models, Bayesian, covariate
  - Expand architecture to additional stages of lifecycle

# Covariate data example

| week | Execution Time (hr) | Failure Identification Work (person hr) | Computer Time-Failure Ident. (hr) | Failure Identified |
|---|---|---|---|---|
| 1 | .0531 | 4 | 1.0 | 1 |
| 2 | .0619 | 20 | 0 | 1 |
| 3 | .1580 | 1 | 0.5 | 2 |
| 4 | .0810 | 1 | 0.5 | 1 |
| 5 | 1.0460 | 32 | 2.0 | 8 |
| 6 | 1.7500 | 32 | 5.0 | 9 |
| 7 | 2.9600 | 24 | 4.5 | 6 |
| 8 | 4.9700 | 24 | 2.5 | 7 |
| 9 | 0.4200 | 24 | 4.0 | 4 |
| 10 | 4.7000 | 30 | 2.0 | 3 |
| 11 | 0.9000 | 0 | 0 | 0 |
| 12 | 1.5000 | 8 | 4.0 | 4 |
| 13 | 2.0000 | 8 | 6.0 | 1 |
| 14 | 1.2000 | 12 | 4.0 | 0 |
| 15 | 1.2000 | 20 | 6.0 | 2 |
| 16 | 2.2000 | 32 | 10.0 | 2 |
| 17 | 7.6000 | 24 | 8.0 | 3 |
| total | 32.8000 | 296 | 60.0 | 54 |

# Covariate model data fit



# Stakeholder outreach




















# Acknowledgements

- This work was supported by (i) the Naval Air Warfare Center (NAVAIR) under contract N00421-16-T-0373 and (ii) the National Science Foundation (NSF) (#1526128).