Software Reliability and Security Assessment: Automation and Frameworks

Lance Fiondella[1] and Ying Shi[2]

University of Massachusetts Dartmouth[1]

NASA Goddard Space Flight Center[2]

Software Failure and Reliability Assessment Tool (SFRAT)

- Implements software reliability growth models and relevant inferences
- *Known users* limited to DoD (NAVAIR), including supporting UARCs (JHU APL) and FFRDCs (MITRE, Aerospace Corporation) as well as major defense contractors (GD, Raytheon)
- Automation script and documentation published on SFRAT Github

Primary outputs and potential to communicate risk to management

Primary SFRAT outputs

- Trends in
  - Faults discovered
  - Time between failures
  - Failure intensity
- Reliability growth curve
- Predictions
  - Time to achieve specified reliability
  - Number of failures in specified time
  - Time to next k failures

Potential benefits

- Visually and quantitatively identify progress toward software stability (less frequent/severe failure)
- Quantify probability of failure free operation for duration of mission
- Determine time required to achieve target reliability, time between failure, and failure intensity (corresponding schedule and cost risk)

SFRAT user modes

- Graphical user interface
  - Web and intranet
- Developer mode
  - Incorporate additional models
- Power user (present effort will support this class)
  - Streamline use for incorporation into internal software testing processes to encourage widespread application
  - Requires additional logic to remove human user from interface

Data requirements

- 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,

$$\mathbf{T} =< t_1, t_2, \dots, t_n >$$

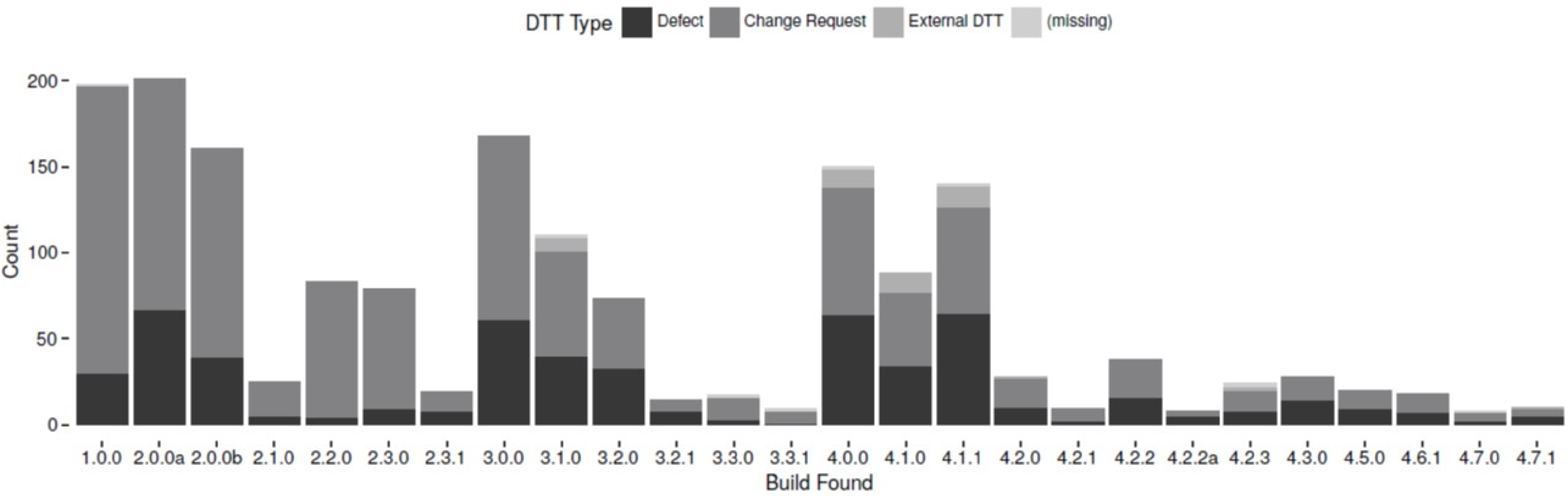
- Failure Counting models
  - Failure count data - length of the interval and number failures observed within it,

$$< \mathbf{T}, \mathbf{K} >=< (t_1, k_1), (t_2, k_2), \dots, (t_n, k_n) >$$

Data requirements (2)

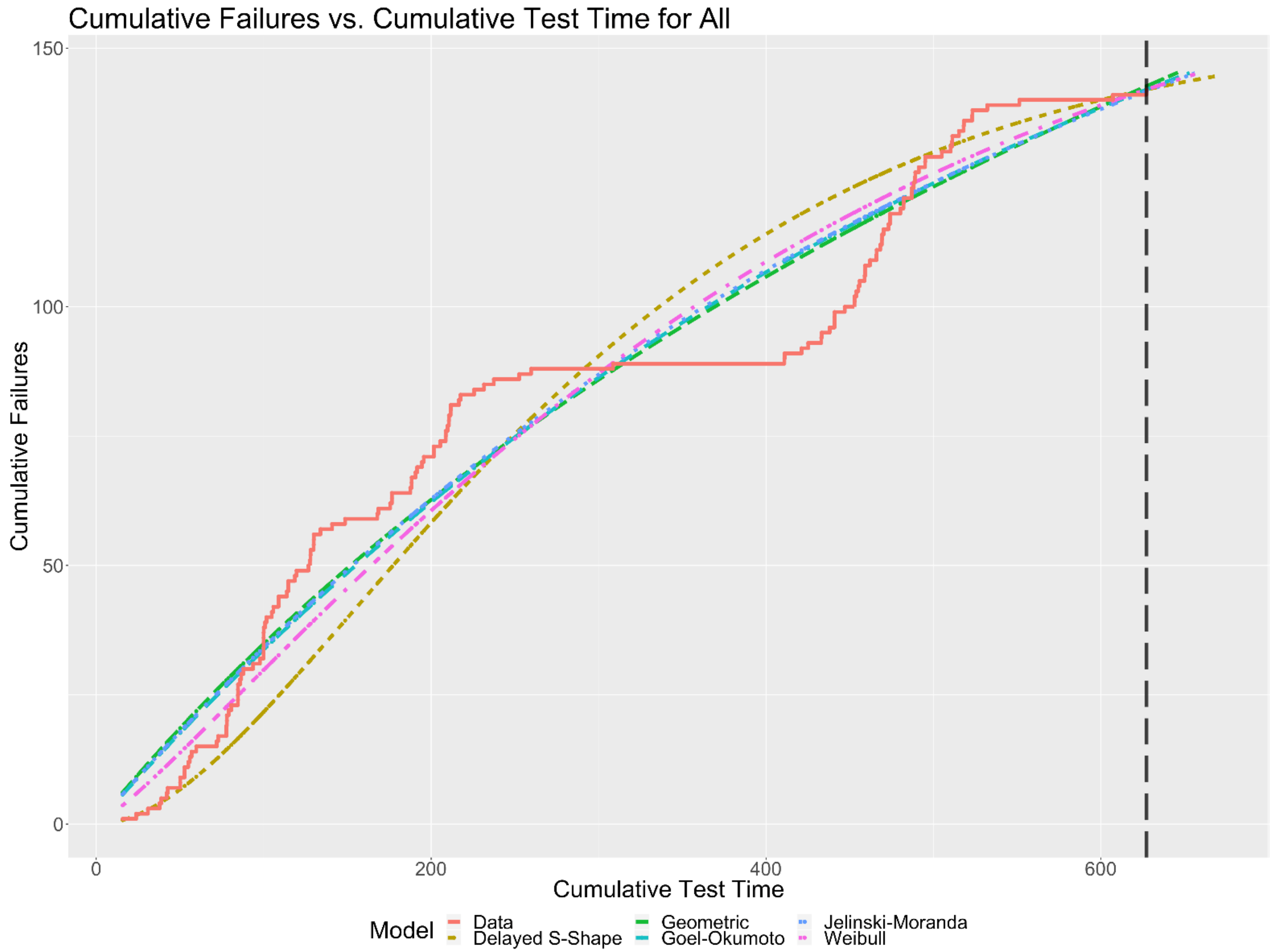
- The following will enable more accurate assessment and additional modeling
  - Time spent testing in each interval
  - Open and close times of defects
  - Severity
  - More detailed activity data in each interval
    - Execution time (hr), failure identification work (person hr), computer time failure identification (hr)
  - Cybersecurity
    - Penetration testing vs. vulnerabilities discovered

Summary of frequencies by DTT type and build found



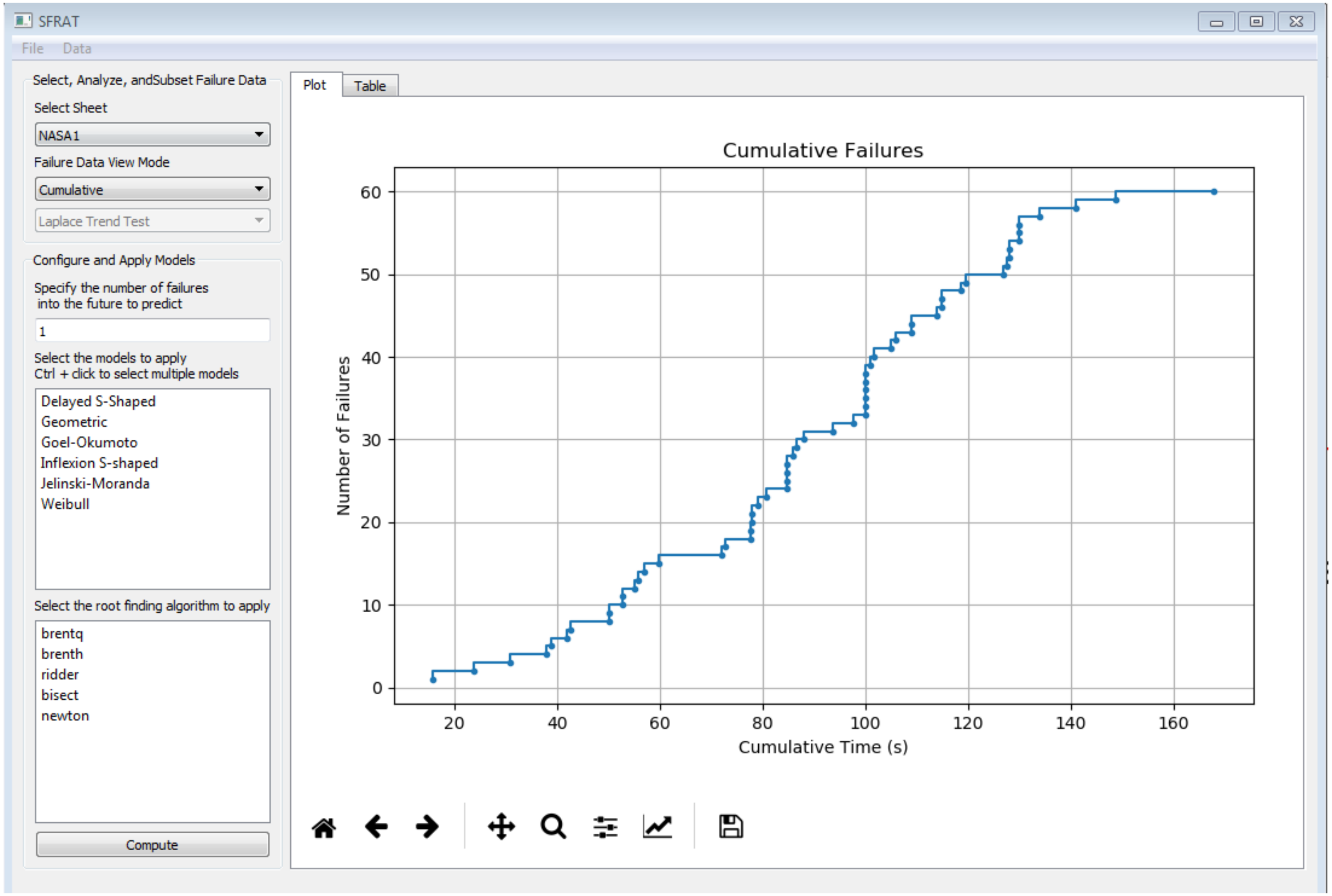
Extracted defects and change requests from major and minor versions exhibiting a large number of events

Concatenated data



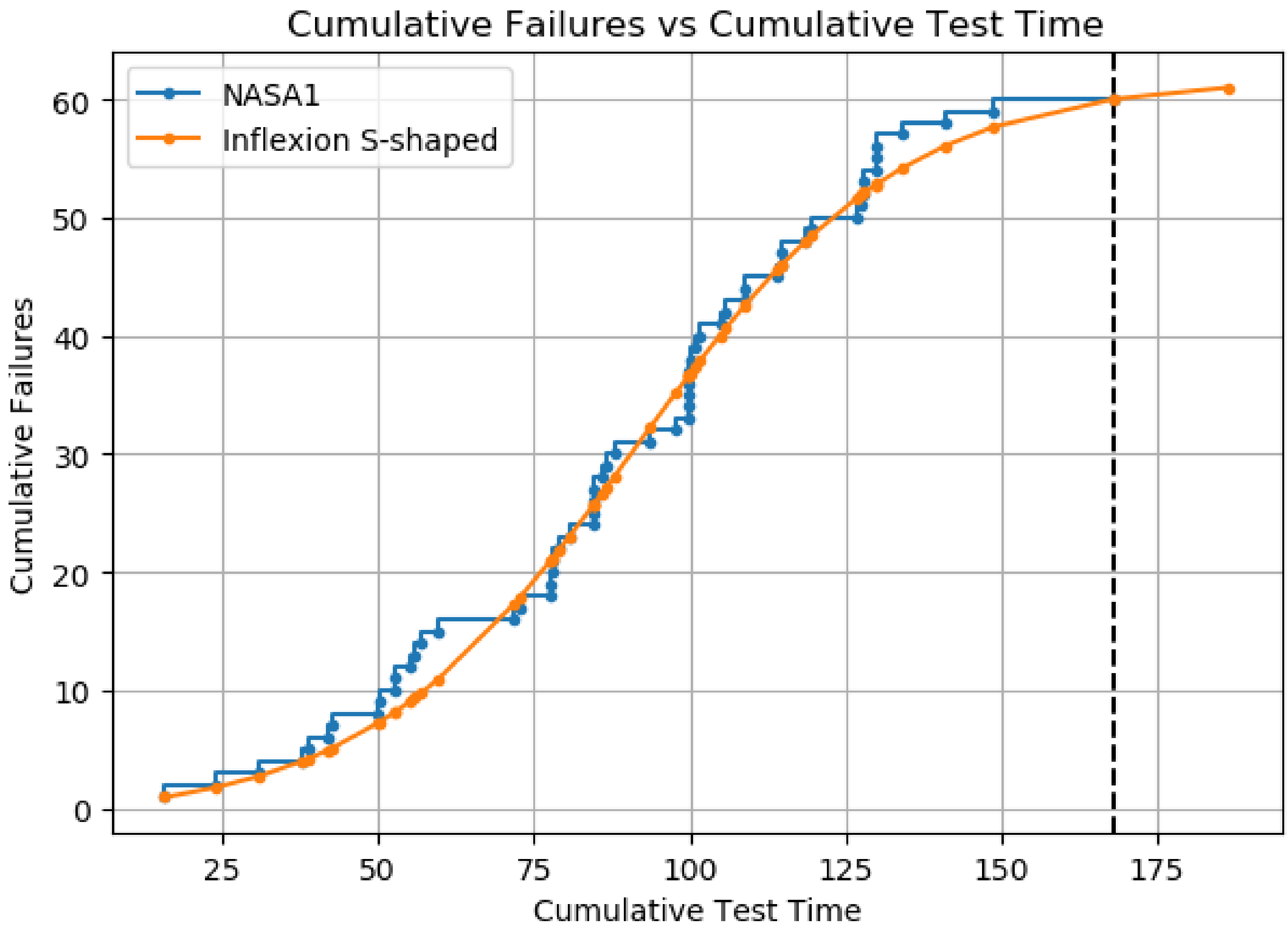
Concatenating data from successive minor versions without annotations lacks information about process

Tab 1 – After data upload



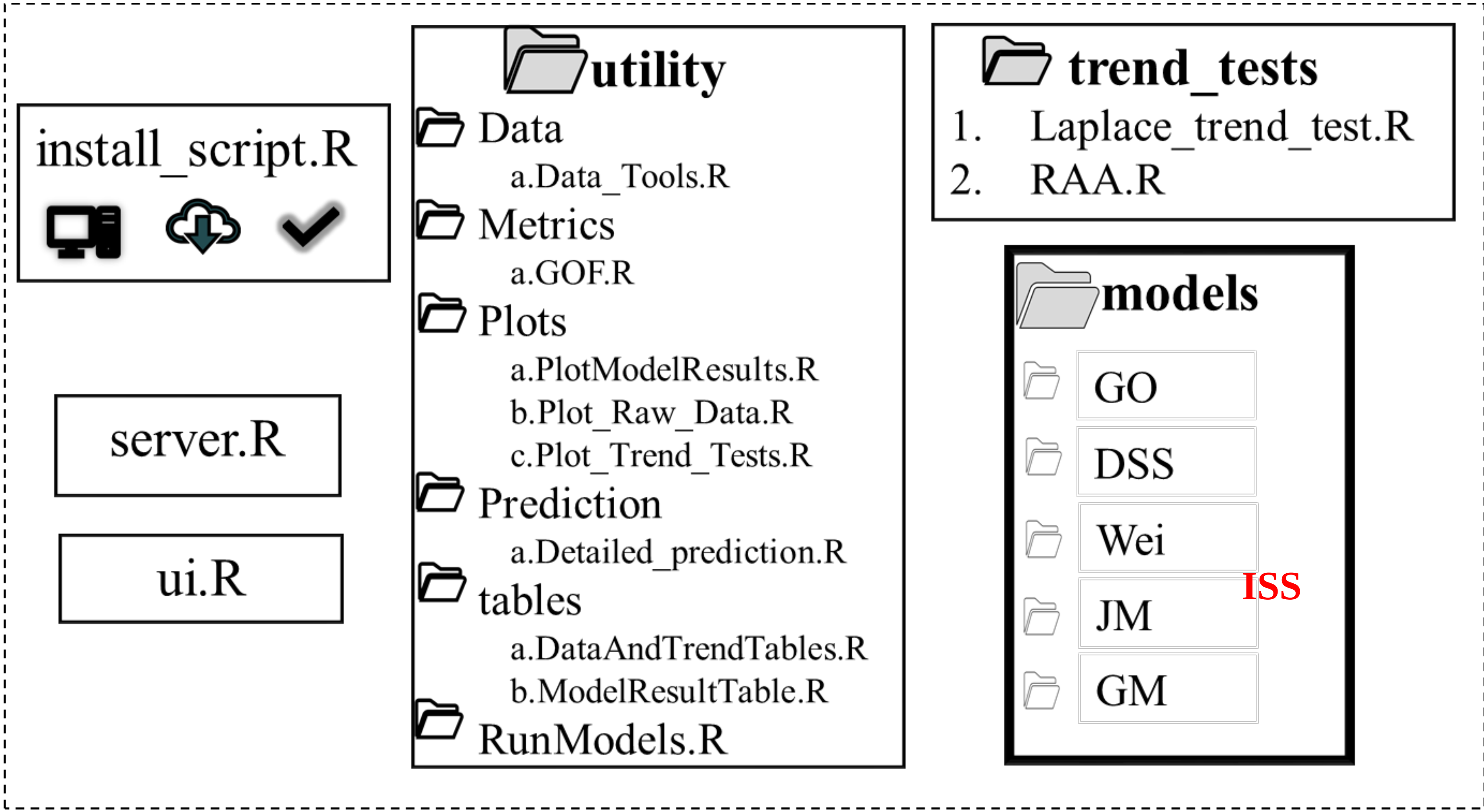
Cumulative failure data view

Cumulative failures

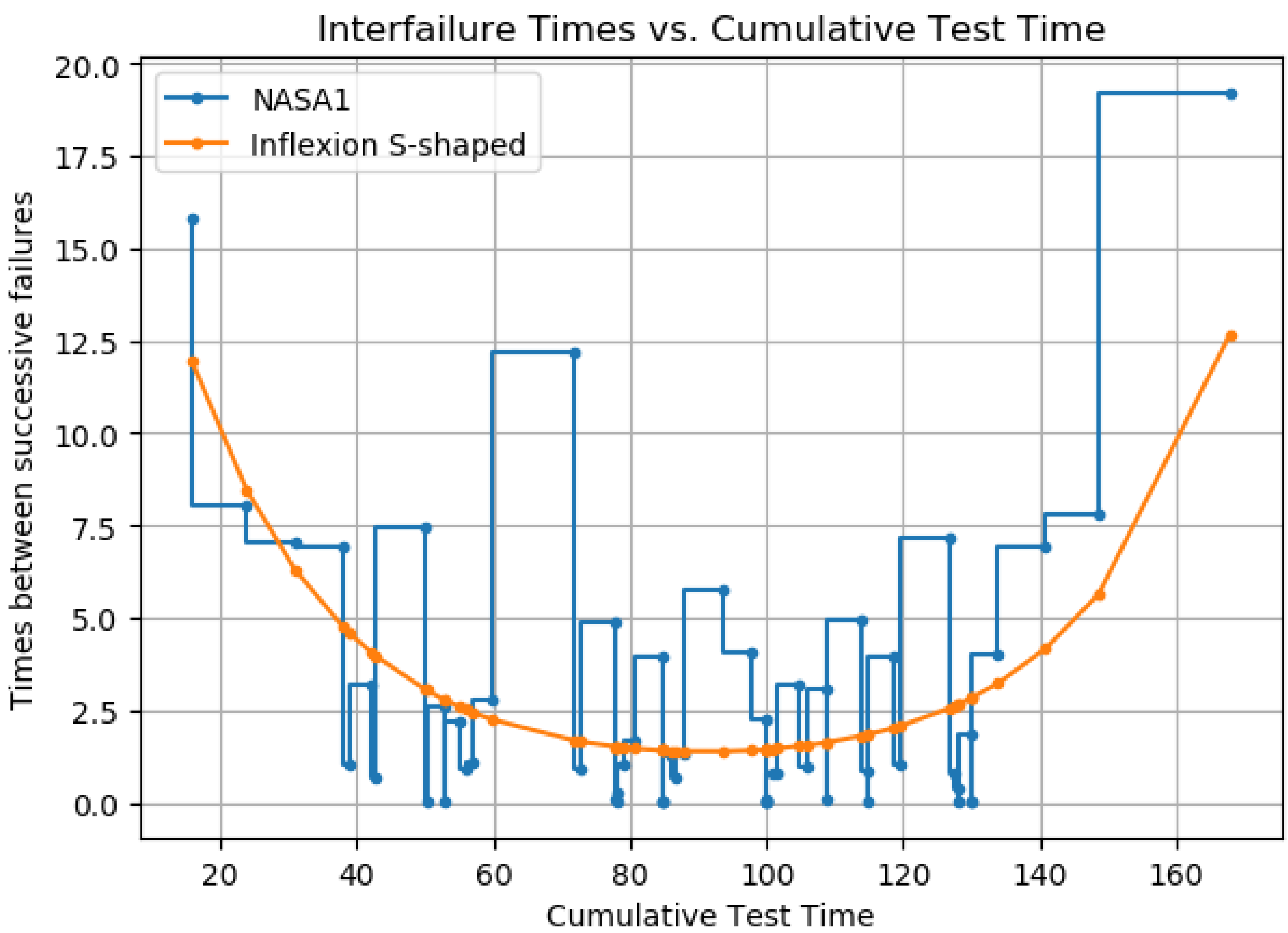


Plot enables comparison of data and model fits

SFRAT – File structure

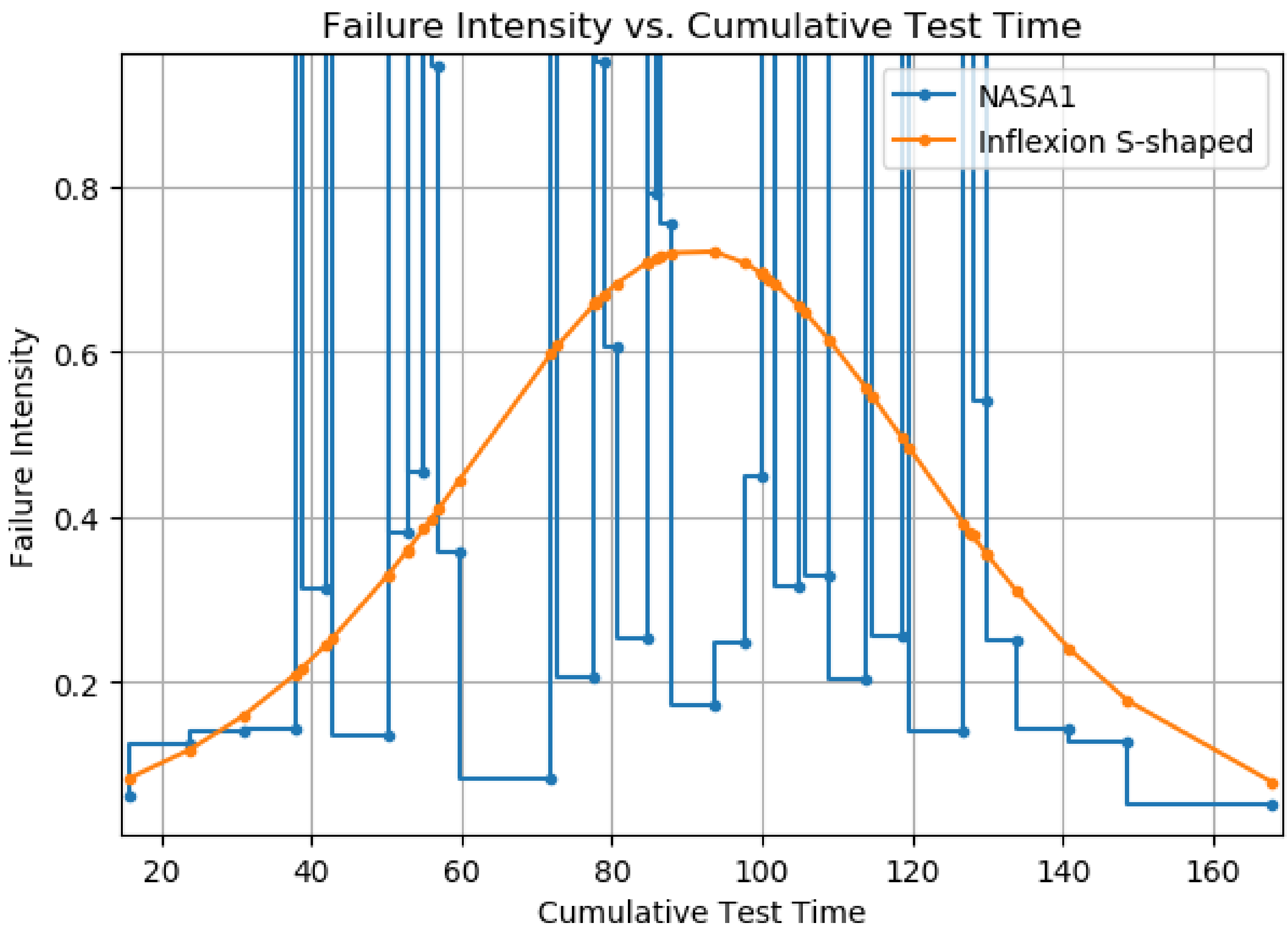


Time between failures



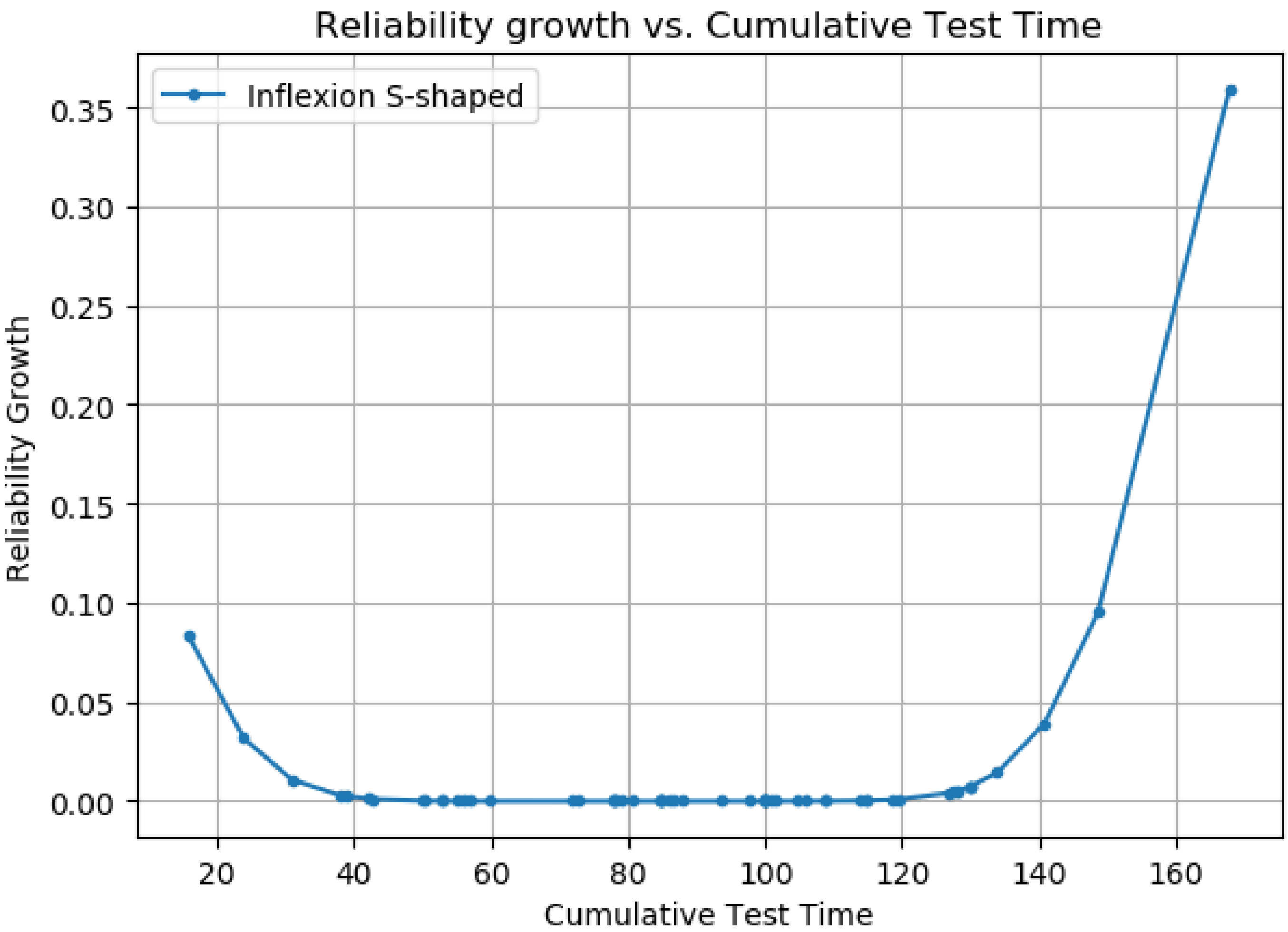
Times between failures should increase (indicates reliability growth)

Failure intensity



Failure intensity should decrease (indicates reliability growth)

Reliability growth curve



Reliability growth curve can estimate time to achieve target reliability

Failure Predictions

	Model	Time to achieve R = 0.9 for mission of length 10	Expected # of failures for next 10 time units	Nth failure	Expected times to next 1 failures
	All	All	All	All	All
1	Delayed S-Shape	342.43212	2.916884	1	3.349118
2	Goel-Okumoto		3.57428	1	2.797767
3	Inflexion S-Shape	207.73900	0.639518	1	2.12015

Time to achieve target reliability can help identify potential schedule overruns

Model goodness of fit – AIC and PSSE

	Model	AIC	PSSE
	All	All	All
1	Delayed S-Shape	232.587035	290.840042
2	Goel-Okumoto	247.458558	341.069591
3	Inflexion S-Shape	**223.002**	**41.8493**

Lower values preferred

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
  - Python port should further enhance opportunities to incorporate into organizational processes

SFRAT Automatic Report Generation

report-specifications.R

```r
# Edit below for script parameters
#1 for verbose report, 2 for non-verbose
verbose_report <- 2

#Specify the location of the input data file and the sheet to pick
filePath <- '/SFRAT/model_testing/NASAX.xlsx'
sheetNumber <- 1  #Selects NASA1 data in this case

#Tab 1 Parameters:
confidence_lvl <- 0.9 #Laplace test confidence level

#Tab 2 Parameters:
num_failures_future_prediction <- 2  #Number of future failures to predict
models_to_apply <- c('DSS','GM','Wei','GO','JM') #Pick models to include, by default is all
mission_time <- 10  #Mission time to compute reliability growth

#Tab 3 Parameters:
num_failures_to_predict <- 3  #Number of future failures to predict
additional_time_software_will_run <- 10 #Future prediction time
desired_reliability <- .9 #Between 0-1, desired software reliability
reliability_interval_length<- 10  #Interval size

#Tab 4 Parameters:
percent_data_for_PSSE <- .9#Predictive sum of squares, percentage

#DO NOT EDIT BELOW-----------------------------------------------------------------------------------
```

12:19 (Top Level) R Script

SFRATReport.Rmd

````
---
title: 'Software Failure and Reliability Assessment Tool: Report'
author: "xxx"
date: '`r format(Sys.time(), "%Y-%m-%d_%H:%M")`'
output:
  pdf_document: default
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
#opts_chunk$set(tidy.opts=list(width.cutoff=100),tidy=TRUE)
```

```{r, echo=FALSE}
source('./SFRAT/utility/data/Data_Tools.R')##DATA PREPROCESSING
d <- dataset #Input excel file with a single sheet for now
cnames <- colnames(d) # Read column names in the input excel file

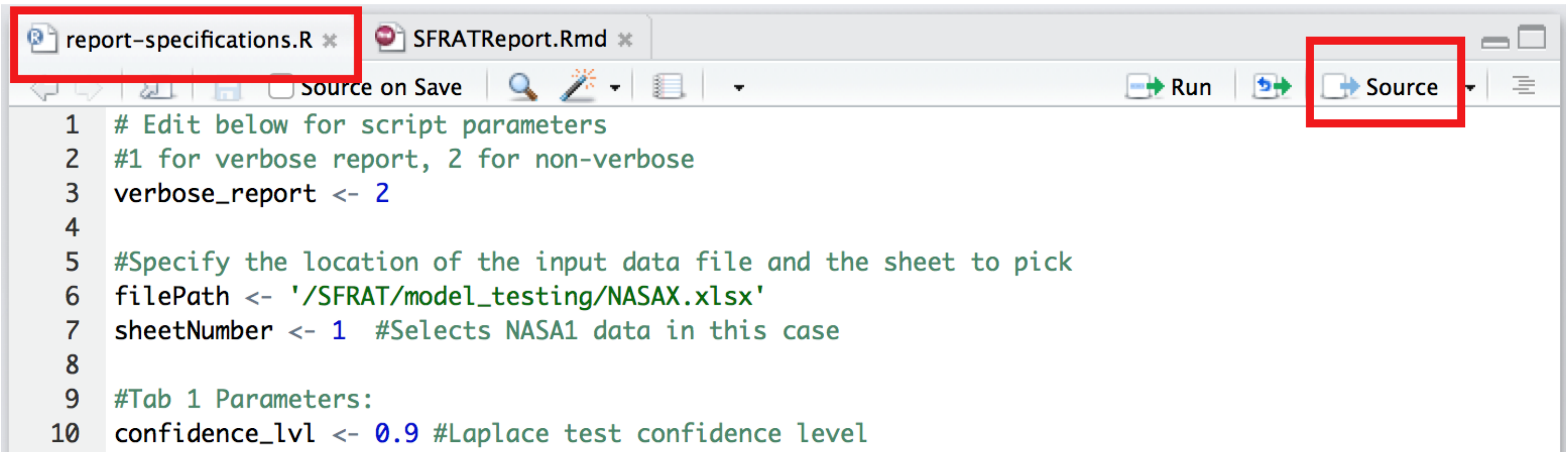
tryCatch({                      #Data conversion depending on the type of the input data
  if("FN" %in% cnames && "IF" %in% cnames && "FT" %in% cnames) {
    FT <- d$FT
    IF <- d$IF
    FN <- d$FN
  } else if("FN" %in% cnames && "IF" %in% cnames) {
    FT <- IF_to_FT(d$IF)
    IF <- d$IF
````

19:35 Chunk 2 R Markdown

.R file with SFRAT input specification

Markdown document to generate report
````

SFRAT Automatic Report Generation



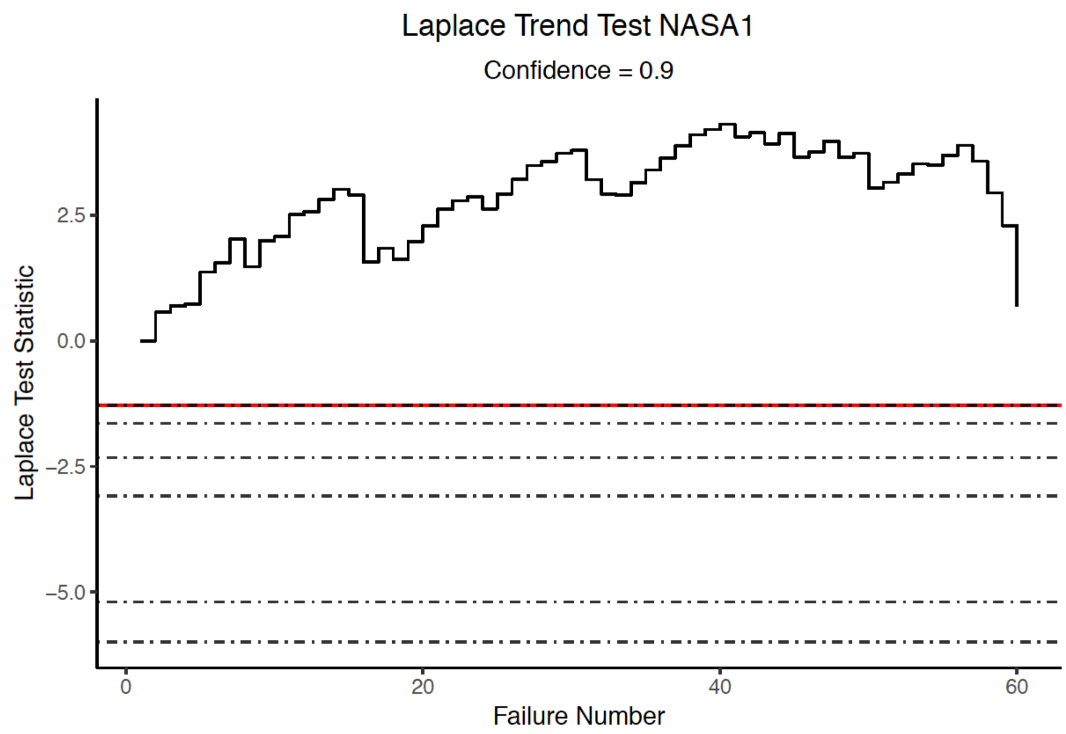
Report can be Knit to pdf, Word, or HTML format

Sample output

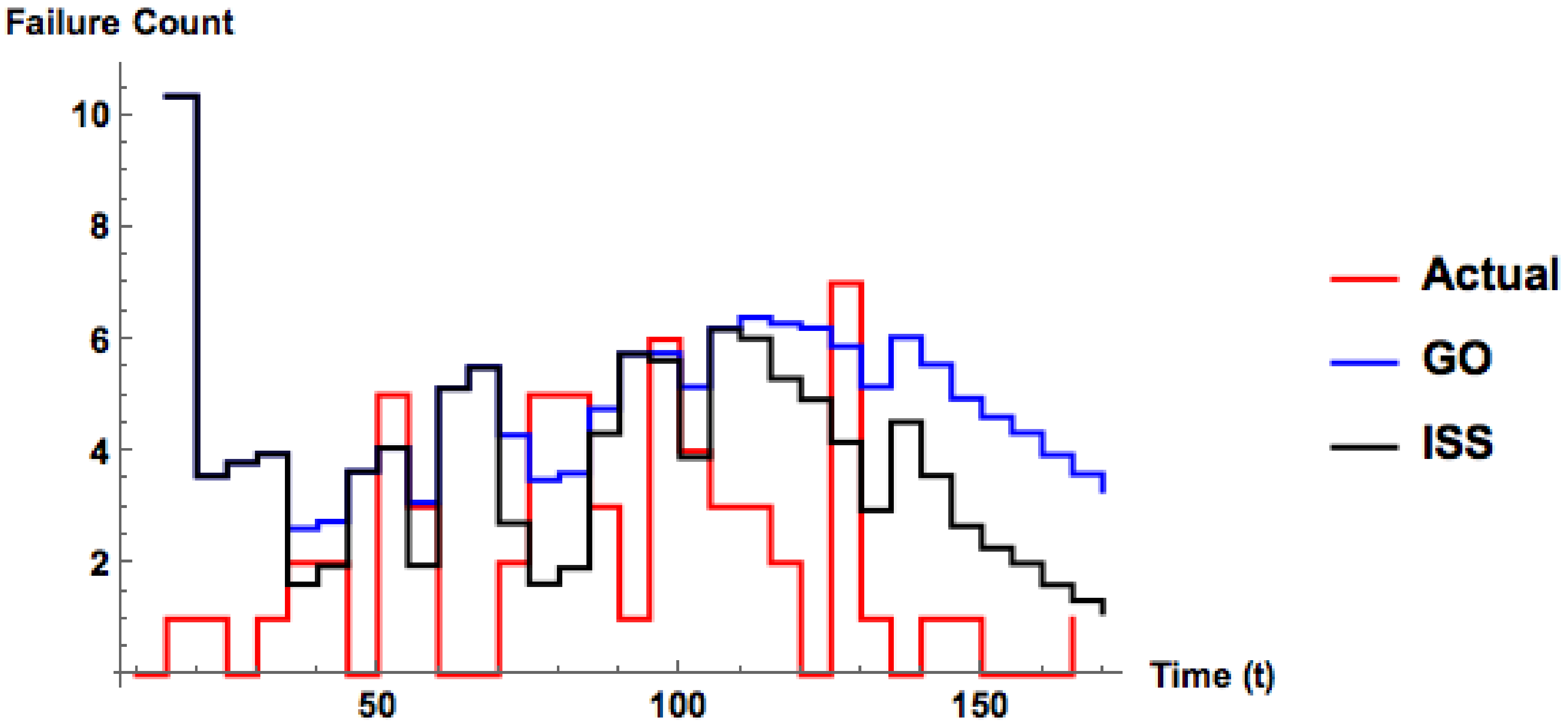
Bookmarks

- Tab 1: Select, Apply, and Analyze Data
  - Sample of the updated data (NASA1) in different formats:
  - Cumulative failures
  - Times between failures/ Interfailure times
  - Failure intensity
  - Laplace Trend Test
  - Running arithmetic average
- Tab2: Set Up and Apply Models
  - Cumulative failures
  - Times between failures
  - Failure intensity
  - Reliability growth
- Tab3: Query Model Results
- Tab4: Evaluate Models

Laplace Trend Test



Failure Count Predictions



Comparative performance based on 5-day ahead predictions

Potential future directions

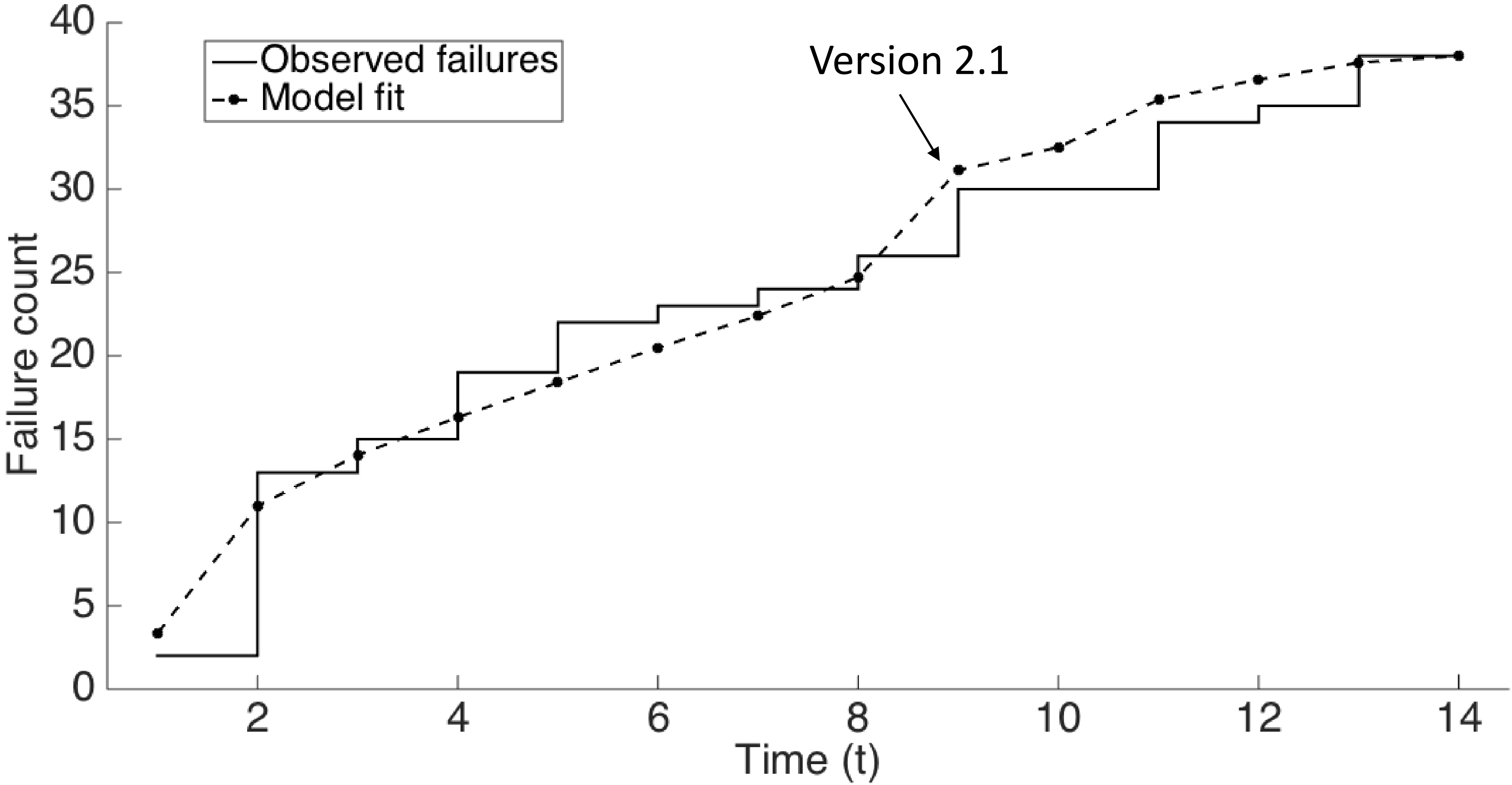
- Develop simple models to characterize data open/close time distributions considering severity with practical goal of
  - Identifying test effectiveness in successive stages as measured by leakage/escape (covariates data may be helpful)
  - Finding and fixing with appropriate resource allocation
- Develop simple models to characterize the fault lifecycle including additional events between open and close
- Work with a program to identify decisions driving effective testing and reliable software
  - What happens between major (1.0), minor (1.1), a patches (1.0.1) that can help?
- Automated extraction from JIRA databases (completed by MITRE) and undergoing public release process

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

Could inform activity effectiveness and process improvement because parameters explicitly linked to activities

Covariate model data fit



Acknowledgements

- This work was supported by the National Aeronautics and Space Administration (NASA) under Grant Number (#80NSSC18K0154) and NSF CAREER award (#1749635).