

Practical Software Reliability Modeling and Application: A Case Study

Vidhyashree Nagaraju, Shekar, Niti, and Lance Fiondella
Department of Electrical and Computer Engineering
University of Massachusetts, Dartmouth, MA, 02747 USA
Email: {vnagaraju,lfiondella}@umassd.edu

Ying Shi
Goddard Space Flight Center
National Aeronautics and Space Administration
Email: @

Abstract—This paper demonstrates the utility of a script to automatically apply a free and open source software failure and reliability assessment tool. The script generates a pdf report, eliminating the need to work with the user interface of the tool to manually assess the data and report. The reports can be used to summarize progress to both technical and non-technical leadership. Simplifying the assessment and reporting may encourage the software and reliability practitioners to apply the models and make decisions about the process.

Index Terms—Software reliability, software reliability growth model, R statistical programming language, GitHub, Software Failure and Reliability Assessment Tool, open source

I. INTRODUCTION

Software reliability growth models (SRGM) [1] are extremely useful in making decisions about software development by characterizing failure data collected during testing. Some of the predictions featured by SRGM include the remaining number of failures, failure intensities, mean time to failures, release time, as well as overall software reliability. However, practitioners may be reluctant to apply these models due to a lack of either the knowledge of the underlying mathematics or the time to develop expertise and implement models in their work. Therefore, several computer-aided software reliability tools [2]–[4] have been developed.

Previous computer-aided tools to apply software reliability methods include: Emerald [5], SRMP (Software Reliability Modeling Programs) [6], the AT&T SRE Toolkit [7], SoRel [8], SMERFS (Statistical Modeling and Estimation of Reliability Functions for Software) [2], and CASRE (Computer Aided Software Reliability Estimation) [4], Robust [9], SREPT [10], CARATS [11], SRATS [3], and M-SRAT [12]. Most of these tools are mostly spreadsheet based [3] or close source in nature. This inhibits the capability to integrate the existing tools into software testing work flows and update the tools with recent changes in the software reliability research. To overcome the limitation of existing tools, an open source Software Failure and Reliability Assessment Tool (SFRAT) [13] is developed with a flexible architecture to accommodate individual researchers models as well as methods.

In this paper, we present a script to automatically apply an open source SFRAT and generate report. This script eliminates the need to work with the graphical user interface to manually

prepare the report, thus conserving time. The script can be configured to produce custom reports; for example, the user can opt to include explanations of each result in the report. The script generates a pdf to visually assess the data and to ease the reporting process. The reports can then be used to summarize and visualize project status to both technical and non-technical leadership.

The remainder of the paper is organized as follows: Section II provides a brief overview of the SFRAT user interface and Section III provides a detailed discussion of the script to generate the report automatically. Section IV demonstrates the use of the script through a real data, while Section V provides conclusion and directions for future research.

II. SOFTWARE FAILURE AND RELIABILITY ASSESSMENT TOOL (SFRAT)

The Software Failure and Reliability Assessment Tool is a free and open source tool developed to promote quantitative assessment of software reliability as well as improved communications of such assessments. SFRAT is an application that estimates and predicts the reliability of a software system during test and operation. Some of the questions that the tool answers about a system undergoing test are:

- Is the software reliable enough to release?
- How long will it be before a specified goal is reached?
- What will be the consequences if testing resources are insufficient?

SFRAT is implemented in the R statistical programming language [], an open source environment for statistical computing and graphics that can be used on computers running Windows, OSX, or Linux, with the user interface implemented using the R Shiny library []. The code is accessible on GitHub at <https://github.com/LanceFiondella/srt.core>, which after downloading may run using RStudio or some other R environment. In doing so, an organization can easily perform information assurance of the code prior to use on sensitive failure data. For complete details, the reader is referred to <http://sasdlc.org/lab/projects/srt.html>. The architecture of the SFRAT combines the use of existing software reliability models into a single tool, enabling more systematic comparison of models than previously possible. Presently, data formats supported consist of inter failure time, failure time, and failure count. Functionality includes two trend tests for reliability

growth, two failure rate [1], Jelinski-Moranda and geometric, and three failure counting models, Goel-Okumoto [14], delayed S-shaped [15], and Weibull [16] models as well as two measures of goodness of fit. Also implemented are methods to predict time to reach a desired reliability as well as detecting a number of future failures. The free and open source nature of the tool architecture enables additional models and goodness of fit measures to be later implemented. In this regard, the tool is intended to serve as a shared environment for collaboration among researchers and practitioners.

SFRAT can be accessed by downloading the source code from GitHub and run using an R environment (e.g. RStudio), or through the web instance. Once the application is launched, the initial SFRAT screen is shown as seen in Figure 1.

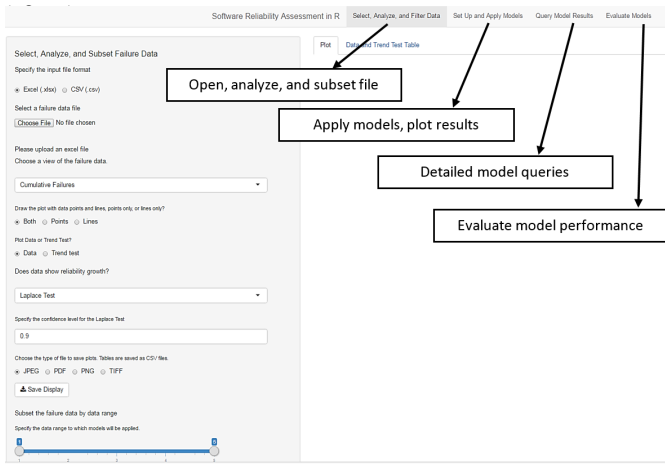


Fig. 1: Initial View within SFRAT

The SFRAT workflow is divided into four subtasks:

- **Select, Analyze, and Filter Data:** Import failure data history and determine model applicability of its reliability's growth.
- **Set Up and Apply Models:** View results of one or more reliability models, including reliability growth.
- **Query Model Results:** Query applied models to answer the following questions:
 - How many additional failures will occur during a continued period of testing?
 - How much time would be required to observe a number of additional failures?
 - At what point in testing will a reliability goal be reached?
- **Evaluate Models:** By model comparison, determine which is most likely to proficiently characterize the data and provide accurate predictions.

A. Tab 1: Select, Analyze, and Filter Data

Figure 2 shows the options in the first tab, which includes file selection, data visualization, and trend test analysis. The user can then use the tool by specifying the input file as either an Excel spreadsheet (.xlsx) or a CSV (comma separated value) (.csv) format. The input file must follow the format

shown in Figure 3, where failure number(FN), interfailure time(IF), and failure time(FT) are indicated.

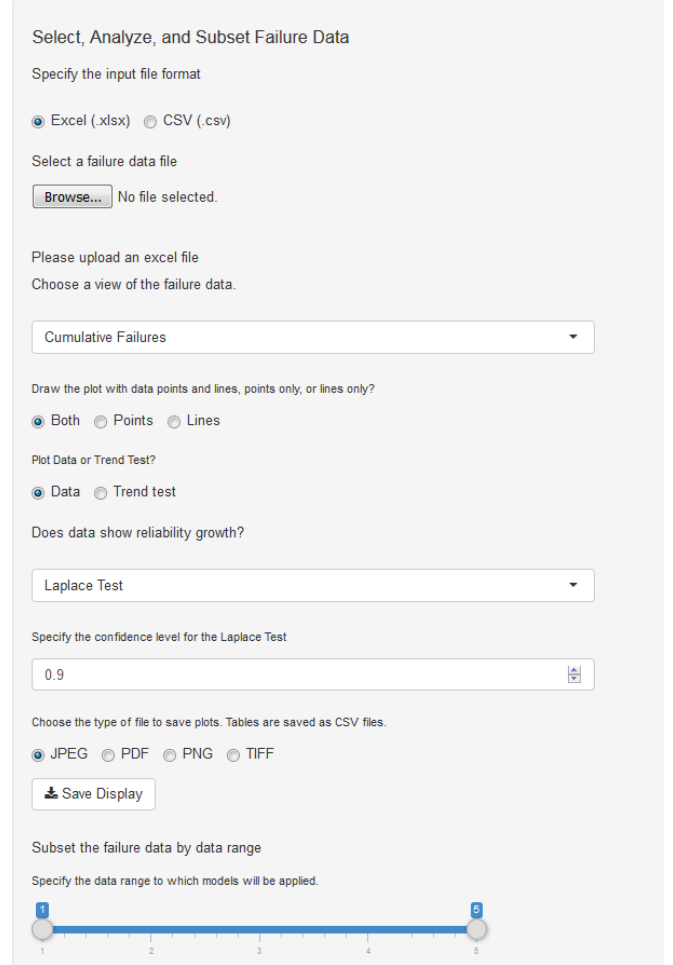


Fig. 2: Tab one view

	A	B	C
1	FN	IF	FT
2	1	3	3
3	2	30	33
4	3	113	146

Fig. 3: Example Excel input

Clicking on the **Browse...** button enables the user to search for and upload the input data file. The progress bar message "Upload complete" will indicate a successful file upload. By default, a plot of the cumulative failures for the uploaded data set is displayed. For example, Figure 4 shows the SYS1 data set [1].

From a drop-down menu below the progress bar, the user may choose from the provided data sets. CSV files may only contain only one data set, whereas Excel files will contain one data set per sheet. Data sets that do not comply with the input format will not be available in the dropdown menu. Regardless of the input's data format, the tool will provide

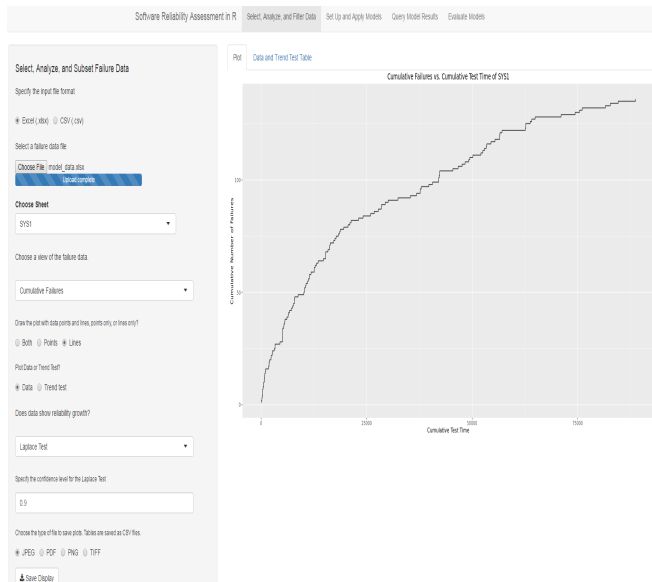


Fig. 4: Tab one view after upload

conversions to failure time, failure count, and inter failure data formats upon upload, allowing for the use of any data model. 31 data sets were taken from the Handbook of Software Reliability Engineering [1] and prepared in the file format. Of these, 10 are failure time data and the remaining 21 are failure counts. This enables a more comprehensive comparison between models than previously possible. Using the drop-down menu below “Choose a view of the failure data”, the user may select an alternative data view, including failure intensities and the times between failures. In addition, the option exists to draw plots with either points, lines or both.

Data and Trend test plotting is also possible by use of the radio buttons with the same labels. These trend tests have been placed before model fitting to ensure that the data exhibits reliability growth and is therefore appropriate for software reliability models and predictions. The two tests implemented include the Laplace trend test [17] as well as a running arithmetic average.

Figure 5 shows the Laplace trend test of the SYS1 data set. The red line seen is a user-specified input, entered into the text box just underneath the Laplace Test drop-down box, as shown in Figure 2. In this case, the value has been set to 0.9 or 90%. Additional default levels in black include 90, 95, 99, 99.9, 99.9999, and 99.999999. Values below these lines indicate that the data exhibits reliability growth with the specified level of statistical significance and is therefore suitable to apply software reliability models.

Figure 6 shows the running arithmetic average of the SYS1 data set. If the time between failures increases, the running arithmetic average then increases, indicating an increase in system reliability. A decreasing running arithmetic average indicates reliability deterioration. Both the Laplace trend test and running arithmetic average suggest that the example SYS1 data set exhibits reliability growth.

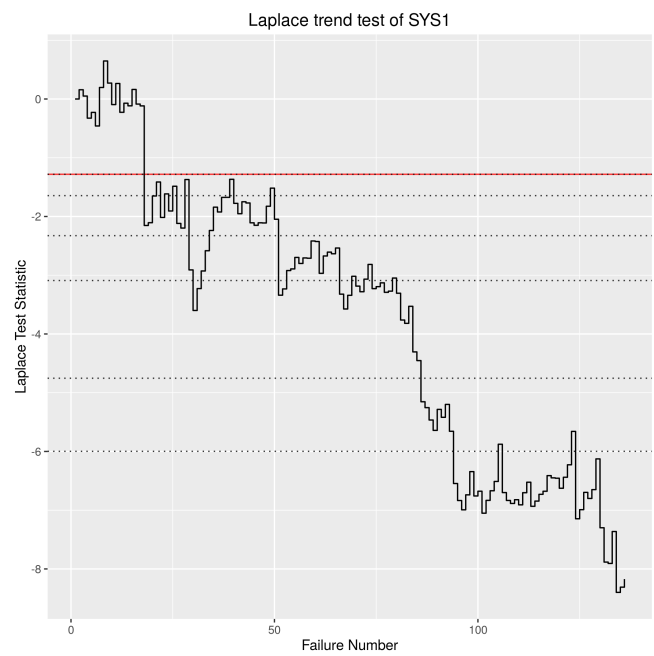


Fig. 5: Laplace trend test

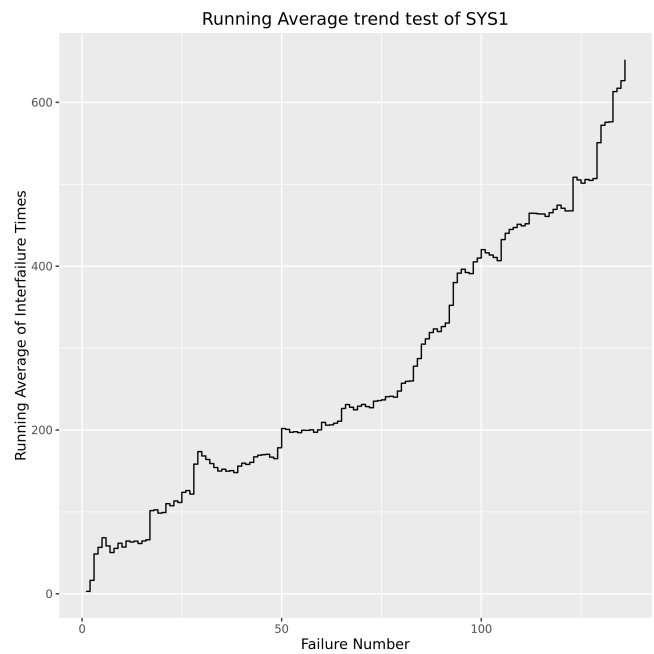


Fig. 6: Running arithmetic average

Figure 7 shows the Laplace trend test of the J4 data set, which does not experience any significant improvement in reliability. Thus, it is not appropriate to apply software reliability models to this data set until additional testing is performed to establish confidence in reliability growth. The slider at the bottom of Figure 2 ranges from 1 to n , where n denotes the number of data points contained in a data set. The sliders allow the user to specify a subset of the data for plotting, model fitting, and prediction. The default is to use

all n data points for model fitting.

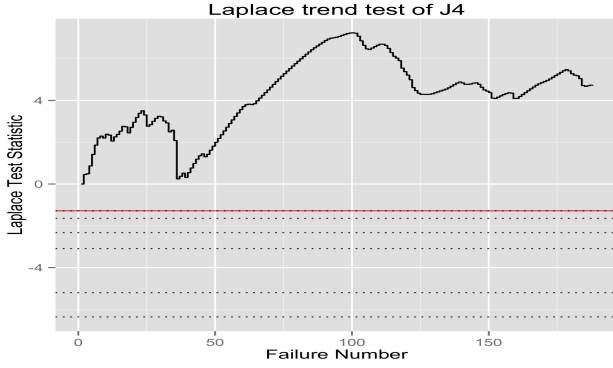


Fig. 7: Data set without reliability growth

The user can switch between the **Plot** and **Data and Trend Test Table** as shown in Figure 4. Selecting table view displays the raw numerical data used to draw the plots in a tabular format similar to Figure 3. Selecting a radio button **.jpeg**, **.pdf**, **.png**, or **.tiff** (Figure 2) and then clicking **Save Display** saves a plot in the selected image format, while numerical tables may be saved as a CSV or PDF file. Tabular data may be exported to input into graphing software to include images in reports and research papers.

B. Tab 2: Set up and Apply Models

Figure 8 shows the second tab's options, where model fitting will be performed. The first text box allows the user to "Specify the number of failures for which the models will make predictions". Here, the number of failures has been set to one for the sake of illustration. The second multi-select box allows the user to identify which models to fit with the data range chosen on Tab one. By default, the tool displays all available models. Clicking the **Run Selected Models** button executes the algorithms to fit the selected model.

Once model fitting completes, the multi-select box below "Choose one or more sets of model results to display" is populated with models that have completed successfully. Models that fail, however, will not be available. The results of successful models can be compared against the empirical data by selecting "Cumulative Failures", "Time between failures", "Failure intensity", or "Reliability growth" from the dropdown menu below "Choose a plot type".

Figure 9 shows a plot of the observed cumulative failures along with the fitted models. The black vertical line indicates the time at which the last failure occurred. Thus, points to its right indicate predicted failures. This line can be hidden by deselecting the **Show end of data on plot** checkbox. The legend at the bottom identifies the line that corresponds to each model fit. Figure 9 suggests that the geometric and Weibull models fit the observed data closely, whereas other models over-predict or under-predict the number of failures at various stages of testing.

Configure and Apply Models

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

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

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

☒ Delayed S-Shape
 ☒ Geometric
 ☒ Goel-Okumoto
 ☒ Jelinski-Moranda
 ☒ Weibull

Run Selected Models

Display Model Results

Choose one or more sets of model results to display

Choose the type of plot for model results.

Choose a plot type

Cumulative Failures

For how much time should the model results curve extend beyond the last prediction point?

☒ Show data on plot

☒ Show end of data on plot

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

☒ Both
 ☐ Points
 ☐ Lines

Choose the type of file to save plots. Tables are saved as CSV files.

☒ JPEG
 ☐ PDF
 ☐ PNG
 ☐ TIFF

Save

Fig. 8: Tab two options

Figures 10, 11, and 12 show the times between failures, failure intensities, and reliability growth data views respectively as well as with the corresponding model fits.

C. Tab 3: Query Model Results

Figure 13 shows the options on the third tab, which focus on a variety of predictive algorithms. The multi-select box below "Choose one or more sets of model results to display" allows the user to specify a model with which they would like to make predictions. To determine the time required to observe the next N failures, the user enters that number of failures in the text box just below. Alternatively, the user may "Specify the amount of additional time for which the software will run" to determine the number of faults that would be detected a specified interval. Entering numbers into these text boxes automatically generates a table (for example see Figure 14), which shows the model names and expected number of failures for the next t time units as well as the expected time required to detect the next N failures. In this case, Figure 14 shows predictions for the SYS1 data set, including the time for one

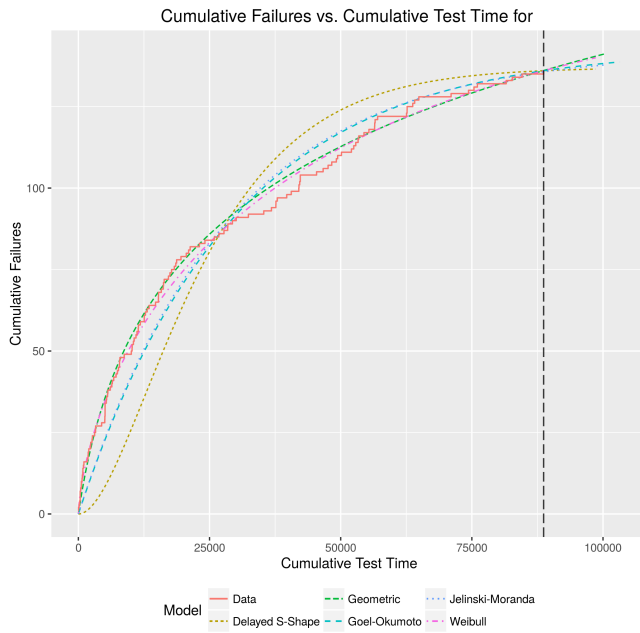


Fig. 9: SYS1 cumulative failures and model fits

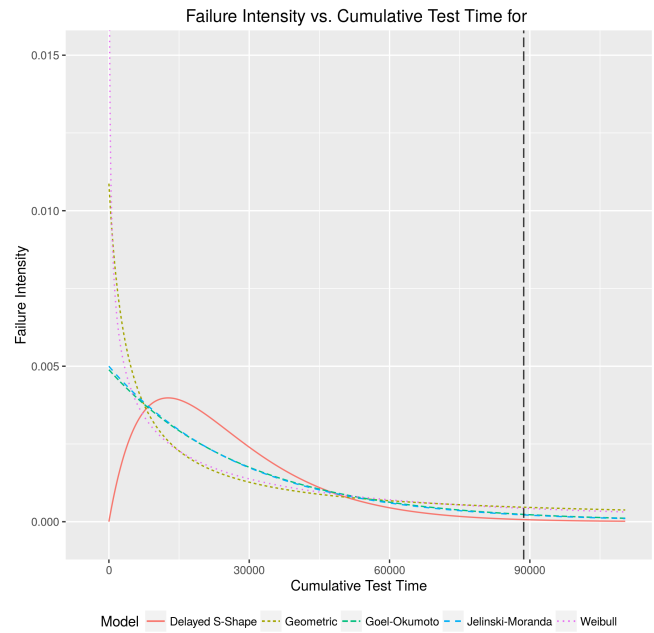


Fig. 11: SYS1 failure intensity and model fits

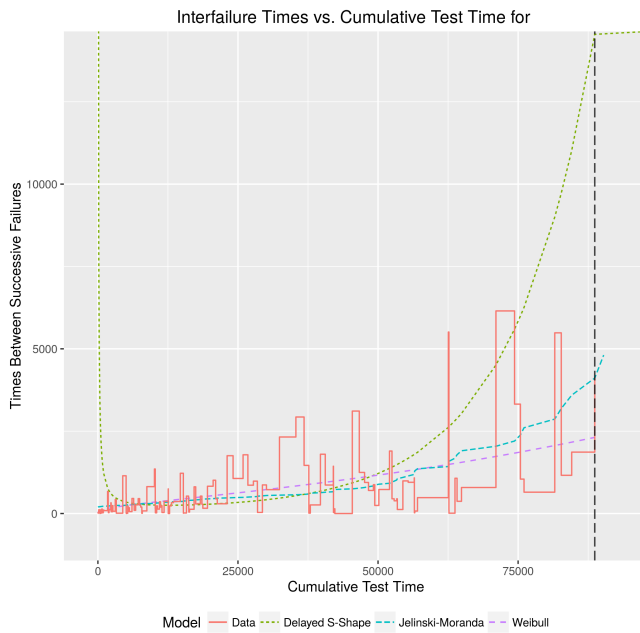


Fig. 10: SYS1 time between failures and model fits

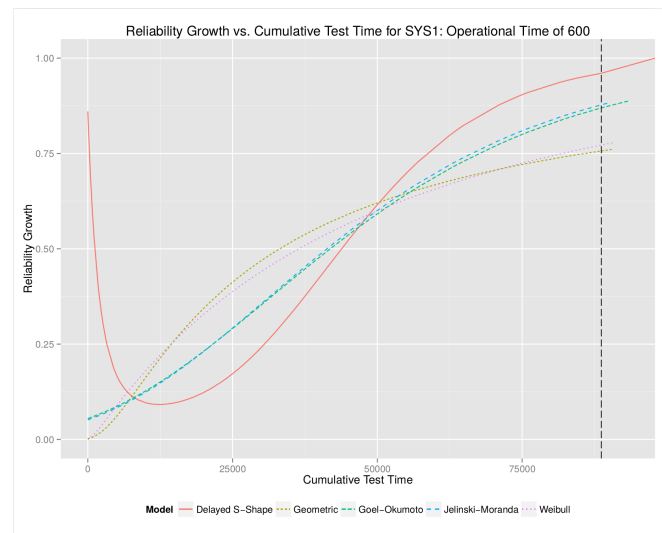


Fig. 12: SYS1 reliability growth and model fits

additional failure and the predicted number of failures to be observed in the next 100,000 seconds of additional testing time.

Tab three (Figure 13) also provides an option to estimate the testing time required to achieve a target reliability by entering the desired reliability and time for which the software must operate without failure in the text box below “Specify the desired reliability” and “How much more test time to achieve a specified reliability?” respectively.

D. Tab 4: Evaluate Models

Figure 15 shows tab four’s options, which provide methods to assess model goodness of fit. The multi-select box below **Choose one or more sets of model results** allows a user to specify models they would like to apply goodness of fit measures, including the Akaike information criterion (AIC) [18] and predictive sum of squares error (PSSE) [19]. The text box below “Specify the Percent Data for PSSE” allows specification of the fraction of data to be used to compute PSSE.

Figure 16 shows the AIC and PSSE values for the five models applied to SYS1 when 90% of data is used. The

Make Detailed Predictions From Model Results

Choose one or more sets of model results to display.

Delayed S-Shape Geometric Goel-Okumoto Jelinski-Moranda Weibull

How much time will be required to observe the next N failures

Specify the number of failures that are to be observed.

1

How many failures will be observed over the next N time units?

Specify the amount of additional time for which the software will run.

4116

How much more test time to achieve a specified reliability?

Specify the desired reliability.

0.9

Specify the length of the interval for which reliability will be computed

4116

Save detailed model results as PDF or CSV?

☐ CSV ☒ PDF

Save Model Predictions

Fig. 13: Tab three options

Model	Time to achieve R = 0.9 for mission of length 4116	Expected # of failures for next 10000 time units	Nth failure	Expected times to next 1 failure
Delayed S-Shape	12401.1541525981	0.9596777	1	NA
Geometric	1592716.45836287	31.8197191	1	2170.0308926781
Goel-Okumoto	62829.7672027733	6.6595071	1	4591.28468949961
Jelinski-Moranda	59023.265608516	6.0584251	1	4956.13815402745
Weibull	259865.770847692	23.5502173	1	2393.05254648438

Fig. 14: Failure predictions

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

Delayed S-Shape Geometric Goel-Okumoto Jelinski-Moranda Weibull

Specify the Percent Data for PSSE

0.9

Save model evaluations as PDF or CSV?

☐ CSV ☒ PDF

Save Model Evaluations

Fig. 15: Tab four options

up/down arrows next to the performance measures in Figure 16 sort the table based on rankings. Figure 16 indicates that the Geometric and Weibull models perform best with respect to the

AIC, while the Jelinski-Moranda and Goel-Okumoto models perform well with respect to PSSE.

	Model	S	AIC	PSSE
1	Delayed S-Shape		2076.146	296.34925
2	Geometric		1937.034	84.32708
3	Goel-Okumoto		1953.613	23.07129
4	Jelinski-Moranda		1950.541	24.39945
5	Weibull		1938.161	74.94496

Fig. 16: AIC and PSSE of all models

III. SFRAT REPORT GENERATION SCRIPT

This section discusses the installation and input specifications of the script. Generally, manually configuring the UI to then create a report explaining project status can be both tedious and unpleasant. As a result, a script has been written to automatically run the SFRAT and generate a PDF report of the recorded data. The script is relatively easy to configure and will assign the user's settings to the SFRAT before running and documenting the tool's results. This allows for easy report "writing", i.e. generation, and comes with the advantage of repeatability.

The script is written in the R programming language and utilizes the Markdown library to generate the report. The script consists of two files namely **report-specifications.R** and **SFRATReport.Rmd**. The **SFRATReport.Rmd** file is a R Markdown file used to create reports in different formats, whereas the **report-specifications.R** is an R script that allows the user to specify inputs necessary to run the SFRAT before generating a report.

A. Installation

An automated installation script is available from the GitHub repository at <https://github.com/LanceFiondella/SFRAT-Automated-Report>.

To install the script manually:

- Windows 7 (64-bit) or newer, Mac OS X 10.9 or newer, or Linux capable of running R or RStudio.
- Perl 5 version 16 or newer - Linux and Mac systems may already have this installed; on Windows machines, Perl may be downloaded from <http://strawberryperl.com/>
- If you are running Windows, any LaTeX text editor (e.g. MikTeX).
- Install R and RStudio on your machine. You can download both RStudio and R at rstudio.com. For Windows, Mac OS X, and Linux, you will need version R version 3.0 or newer and RStudio 0.99.482 or newer.
- Once RStudio has been installed, the following packages need to be installed:
 - **shiny**, a web application framework for R
 - **gdata**, a package providing data manipulation tools
 - **ggplot**, a graphical package capable of creating elegant and complex plots
 - **DT**, a package providing an R interface to the JavaScript library DataTables
 - **rootSolve**, a package used to find the roots of n nonlinear (or linear) equations

- **knitr**, a package providing a general-purpose tool for dynamic report generation in R using Literate Programming techniques
 - **rmarkdown**, a package allowing converting R Markdown documents into a variety of formats including HTML, MS Word PDF, and Beamer
 - **markdown**, a package for authoring HTML, PDF, and MS Word documents
 - **readxl**, a package to import excel files
 - **formatR**, a package designed to reformat R code to improve readability
- There is an automatic package installation script in the report generation script repository. Running this script is described below in section III-B. Otherwise, these packages may also be installed manually via the "Install packages..." option in RStudio's "Tools" menu, as shown in Figure 17:

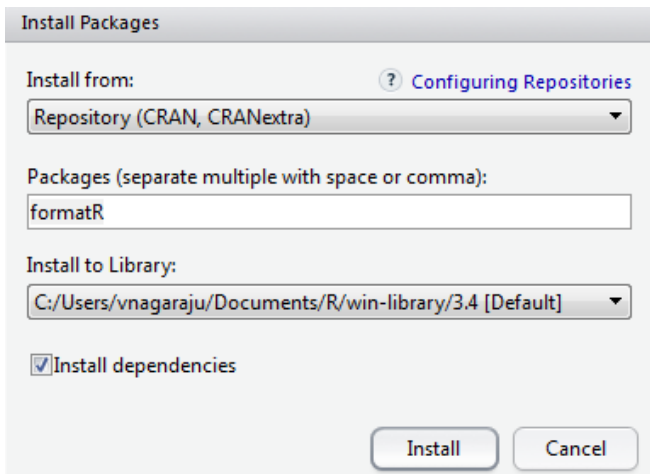


Fig. 17: Install packages dialog box

To install manually, enter the specified package names into the text input, and make sure the "Install dependencies" box is checked. It's unnecessary to change the installation source or location.

B. Configuring and running the script

To run the script using RStudio:

- 1) To download the script locally, navigate to github.com/LanceFiondella/SFRAT-Automated-Report, and from the "Clone or download" drop-down click "Download ZIP". Extract the folder to a location of your choosing.
- 2) Launch RStudio and set its working directory to the location where the folder is saved.
 - a) Click the menu options **Session** → **Set Working Directory** → **Choose Directory...**
 - b) Navigate to the location of the downloaded folder and click 'Open'.
- 3) If not previously done, install the packages described in Section III-A. This can be done automatically by

opening the 'installscript.R' file in RStudio and clicking 'Source'.

- 4) Open the file 'report-specifications.R'. Each variable in this script represents some parameter in the UI for the SFRAT. The script is designed to be easy to configure in that all the SFRAT settings will be found on this one script in one short section.

- Main input specifications:

- **verbose_report** - Shows or hides verbose descriptions on PDF report pages. Set to TRUE to enable such descriptions or FALSE to hide them.
- **sheetNumber** - If the input data file has multiple datasets arranged on different sheets, this option allows the user to specify a particular dataset. It is recommended to name the sheet to distinctly include the data set name in the report.
- **datapath** - Specify the path of the input data file.
- **colors** - Specify the set of colors used in the report. If nothing is specified, the default set of colors will be used to display graphical results.

- Tab 1: Select, Analyze, and Filter Data

- **confidence_lvl** - This allows the user to specify a confidence level between 0 and 1 (as a float value) to quantify a desired level of significance that the data exhibits reliability growth using the Laplace trend test.

- Tab 2: Set Up and Apply Models

- **num_failures_future_prediction** - The user can specify the number of failures that they would like to predict beyond the end of testing. The number should be an integer value.
- **models_to_apply** - This allows the user to specify which software reliability growth models they would like to apply to make predictions. The available models are: delayed s-shape (DSS), geometric (GM), Goel-Okumoto (GO), Jelinski-Moranda (JM), and Weibull (Wei) models.
- **mission_time** - The user can specify what amount of mission time beyond the testing period for which they would like to view the reliability growth trend.

- Tab 3: Query Model Results

- **num_failures_to_predict** - Specify the number of failures to predict beyond the end of testing. This is similar to 'num_failures_future_prediction'.
- **additional_time_software_will_run** - User can specify the additional time beyond the end of testing to predict the number of failures.
- **desired_reliability** - User can specify the target reliability between 0 to 1, as a float value, to estimate the time required to achieve that.
- **reliability_interval_length** - This is to specify the mission time beyond testing to estimate the reliability.

- Tab 4: Evaluate Models
 - **percent_data_for_PSSE** - The user can specify the percent of data to be used for model fitting. The remaining data will be used to assess model prediction capability using the predictive sum of squares error (PSSE).

5) After specifying the input parameters, save the file. There are two methods to run the script and generate the report:

- Click 'source' in the top-right corner as shown in Figure 18.
- In the console, run the following command:
`source('report-specifications.R')`

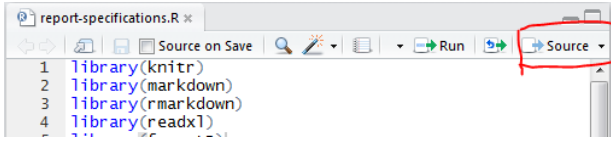


Fig. 18: Source script file

6) The generated reports will be stored in the 'Report' folder, which is located in the same directory as the script. Each report file will have the following naming convention:

SFRAT_report_dataName_YYYY-MM-DD.pdf

IV. ILLUSTRATIONS

This section illustrates the use of script to analyse the failure data using example SYS1 data [1]. The second example demonstrates the online assessment of the data using SYS1.

A. Script on SYS1 data

This example demonstrates the use of the script in the context of SYS1 example data, which consists of 136 failures.

Upon successfully installing all the required packages and downloading the script from the GitHub, open the 'report-specifications.R' file using RStudio. Be sure to set the working directory to the source file location. Prepare an excel file with the input data by following the formatting requirements of the SFRAT shown in Figure 3 and save it in your desired location. For the sake of illustration, we have taken the example data file provided in the downloaded content from GitHub. The first sheet of the 'model_data.xlsx' corresponds to the SYS1 data, which are specified below. All required inputs to all four tabs of the SFRAT tool are specified for SYS1 data as below.

```
# Input Parameters #

#1 for verbose report, 2 for non-verbose
verbose_report <- 2

#select which sheet to pick, in this case
gives SYS1
sheetNumber <- 1
#designate location of data file
filePath <-
  '/SFRAT/model_testing/model_data.xlsx'
```

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

```
#Tab 2 Parameters:
#number of future failures to predict
num_failures_future_prediction <- 2

#pick models to include, by default is all
models_to_apply <- c('DSS', 'GM',
  'Wei', 'GO', 'JM')
#mission time to compute reliability growth
mission_time <- 600
```

```
#Tab 3 Parameters:
#number of future failures to predict
num_failures_to_predict <- 3

#future prediction time
additional_time_software_will_run <- 4116

#between 0-1, desired software reliability
desired_reliability <- .9
```

```
#interval size
reliability_interval_length<- 600
```

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

Figure 19 shows the initial view of the 'SFRAT_report_SYS1_2018-10-11.pdf' with a sample and description of the SYS1 data.

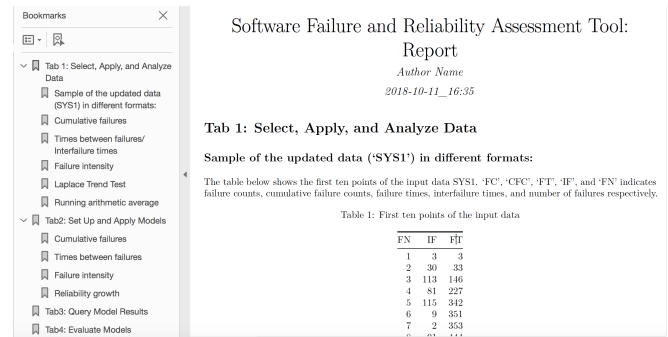


Fig. 19: Initial view of the report using SYS1 data

The left side of the Figure 19 shows the tab level details of all the available results in the report. The report consists of 12 pages with each result displayed in individual page for better presentation. The Verbose report includes a brief description before the table or graph presented in each page. The first page of the report allows the user to specify the author name and change the report name as well as the date and time in the first five lines of the 'SFRATReport.Rmd' file.

Similar graphs for times between failures and failure intensity as a function of the testing time is displayed on Page 3 and 4 respectively.

Similar plot and discussion for the running arithmetic average is included in the following page in the report.

Figure ?? shows the first page of the Tab 2 result, which the model fit for the SYS1 data with 2 failure predictions beyond the end of testing. The end of testing is indicated with a black vertical line.

Similarly, the times between failures, failure intensity, and reliability growth for all the models are displayed between Page 8-10.

In Figure ??, the * indicates the best model based on the corresponding goodness-of-fit measure.

B. Online analysis using the script on SYS1 data

example of application sequence that enables comparison of which model predicts best throughout the data

collection process and the assessments that might have been made if this were an ongoing activity. We can do this in the context of sys1 and then adapt to NASA data.

V. CONCLUSION AND FUTURE WORK

In this paper, we presented a script to automatically apply the Software Failure and Reliability Assessment Tool. An example of the SFRAT in the context of SYS1 dataset and a detailed description to run the tool is discussed. Automated pdf report generation is demonstrated using SYS1 data set. A second example is presented to illustrate the utilization of script to perform online assessment of the data.

Future work will extend this script to accommodate the updated version of the SFRAT. The script will be ported to python programming language considering the ease of use. In addition, options to perform online data assessment will be integrated with required functionalities and measures.

ACKNOWLEDGMENT

This work is supported by the Natioanl Aeronautics and Space Administration (NASA) SARP under Grant Number (#). The authors would also like to acknowledge Christian Ellis, Joshua Steakelum, and Melanie Luperon for their inputs on the manuscript.

REFERENCES

- [1] M. Lyu, Ed., *Handbook of Software Reliability Engineering*. New York, NY: McGraw-Hill, 1996.
- [2] W. Farr and O. Smith, "Statistical modeling and estimation of reliability functions for software (SMERFS) users guide," Naval Surface Warfare Center, Dahlgren, VA, Tech. Rep. NAVSWC TR-84-373, Rev. 2, 1984.
- [3] H. Okamura and T. Dohi, "SRATS: Software reliability assessment tool on spreadsheet (experience report)," in *International Symposium on Software Reliability Engineering*, Nov 2013, pp. 100–107.
- [4] M. Lyu and A. Nikora, "CASRE: A computer-aided software reliability estimation tool," in *IEEE International Workshop on Computer-Aided Software Engineering*, 1992, pp. 264–275.
- [5] J. Hudspeth, S. Aud, T. Khoshgoftaar, E. Allen, and J. Mayrand, "Emerald: Software metrics and models on the desktop," *IEEE Software*, vol. 13, no. 5, pp. 56–60, 1996.
- [6] Reliability and S. C. Ltd., "Software reliability modeling programs, Version 1.0," Tech. Rep., 1988.
- [7] A. B. Laboratories, "Draft software reliability engineering: Reliability estimation tools reference guide Version 3.7," Tech. Rep., 1990.

- [8] K. Kanoun, M. Kaaniche, J. Laprie, and S. Metge, "Sorel: A tool for reliability growth analysis and prediction from statistical failure data," in *IEEE International Symposium on Fault-Tolerant Computing*, 1993, pp. 654–659.
- [9] N. Li and Y. Malaiya, "ROBUST: A next generation software reliability engineering tool," in *IEEE International Symposium on Software Reliability Engineering*, 1995, pp. 375–380.
- [10] S. Ramani, S. Gokhale, and K. Trivedi, "SREPT: Software reliability estimation and prediction tool," *Performance evaluation*, vol. 39, no. 1–4, pp. 37–60, 2000.
- [11] C.-Y. Huang and M. Lyu, "Estimation and analysis of some generalized multiple change-point software reliability models," *IEEE Transactions on reliability*, vol. 60, no. 2, pp. 498–514, 2011.
- [12] K. Shibata, K. Rinsaka, and T. Dohi, "M-SRAT: Metrics-based software reliability assessment tool," *International Journal of Performability Engineering*, vol. 11, no. 4, pp. 369–379, 2015.
- [13] V. Nagaraju, K. Katipally, R. Muri, T. Wandji, and L. Fiondella, "An open source software reliability tool: A guide for users," in *International Conference on Reliability and Quality in Design*, CA, Aug 2016, pp. 132–137.
- [14] A. Goel, "Software reliability models: Assumptions, limitations, and applicability," *IEEE Transactions on Software Engineering*, no. 12, pp. 1411–1423, 1985.
- [15] S. Yamada, H. Ohtera, and H. Narihisa, "Software reliability growth models with testing-effort," *IEEE Transactions on Reliability*, vol. R-35, no. 1, pp. 19–23, apr 1986.
- [16] S. Yamada and S. Osaki, "Reliability growth models for hardware and software systems based on nonhomogeneous poisson process: A survey," *Microelectronics and Reliability*, vol. 23, no. 1, pp. 91–112, 1983.
- [17] O. Gaudoin, "Optimal properties of the laplace trend test for soft-reliability models," *IEEE Transactions on Reliability*, vol. 41, no. 4, pp. 525–532, 1992.
- [18] H. Akaike, "A new look at the statistical model identification," *IEEE Transactions on Automatic Control*, vol. 19, no. 6, pp. 716–723, 1974.
- [19] L. Fiondella and S. S. Gokhale, "Software reliability model with bathtub-shaped fault detection rate," in *Annual Reliability and Maintainability Symposium*, 2011, pp. 1–6.