

Practical Software Reliability Modeling and Application: A Case Study

Vidhyashree Nagaraju, Shekar, Niti and Lance Fiondella

Electrical and Computer Engineering

University of Massachusetts Dartmouth

North Dartmouth, MA, USA Ying Shi *Goddard Space Flight Center*

National Aeronautics and Space Administration

MD, USA

Abstract—This paper demonstrates the utility of a script to automatically apply a free and open source software failure and reliability assessment tool to generate a pdf. The script eliminates the need to work with the graphical user interface of the tool to manually assess the data and report. The reports can be used to summarize progress to technical and non-technical leadership. Simplifying the assessment and reporting may thus 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

I. INTRODUCTION

Software reliability growth models (SRGM) [1] are extremely useful in assisting the software and reliability practitioners to make practical decisions about the software by characterizing the failure data collected during testing. Some of the predictions enabled by the SRGM include remaining number of failures, failure intensity, mean time to failure, release time as well as software reliability. However practitioners are reluctant to apply these models due to 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] were developed in the past.

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 graphical user interface to manually prepare the a report, thus conserving time. The script can be configured to produce custom reports, for example, the user can opt to include explanation 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 be used to summarize progress to 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 and improved communication between organizations acquiring software and those conducting development and test. SFRAT is an application to estimate and predict the reliability of a software system during test and operation. Some of the questions it allows a user to answer about a software system undergoing test are:

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

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, Mac OSX, or Linux, with the user interface implemented in Shiny []. The code is accessible on GitHub at <https://github.com/LanceFiondella/srt.core>, which can be downloaded and run using RStudio [] to use the application locally. 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 enables incorporation of existing software reliability models into a single tool, enabling more systematic comparison of models than previously possible. Presently, inter failure time, failure time, and failure count data formats are supported. 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. Inferences such as the time to achieve a target reliability or detect a specified number of additional failures have also been implemented. The free and open source nature of the tool architecture enables additional models and goodness of fit measures to be included. In this regard, the tool is intended to serve as a shared environment for collaboration among researchers and practitioners.

SFRAT can be accessed either by downloading the source code from Github and run using RStudio or through the web instance. Once the application is launched, the initial SFRAT screen is shown in Figure 1.

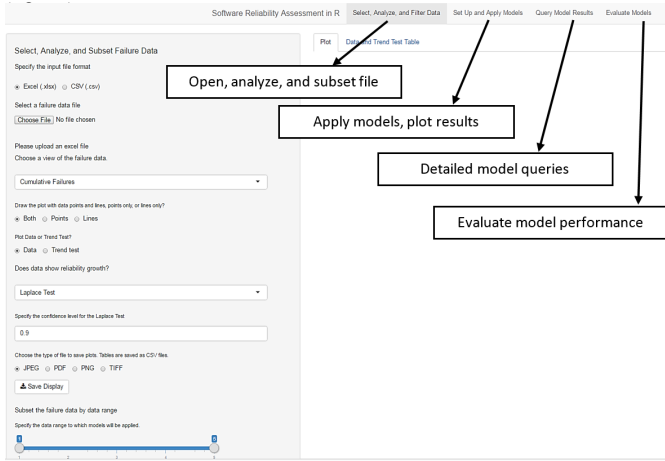


Fig. 1: Initial View within SFRAT

The SFRAT workflow is divided into four subtasks:

- **Select, Analyze, and Filter Data:** Open a failure data history file and determine whether the data exhibits reliability growth that would justify model application.
- **Set Up and Apply Models:** Apply one or more software reliability models and view the results including reliability growth.
- **Query Model Results:** Query applied models to answer the following questions:
 - How many additional failures will be observed if testing is continued for an additional amount of time?
 - How much additional time is required to observe a given number of additional failures?
 - How much additional testing time will be needed to achieve a specified reliability goal (or has that goal already been achieved)?
- **Evaluate Models:** Compare the models with one another to determine which one is the most likely to characterize the data well and provide accurate predictions.

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

Figure 2 shows the options in the first tab, including file selection, data visualization, and trend test analysis. User can begin to 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 (FN) indicates failure number, (IF) interfailure time, and (FT) failure time.

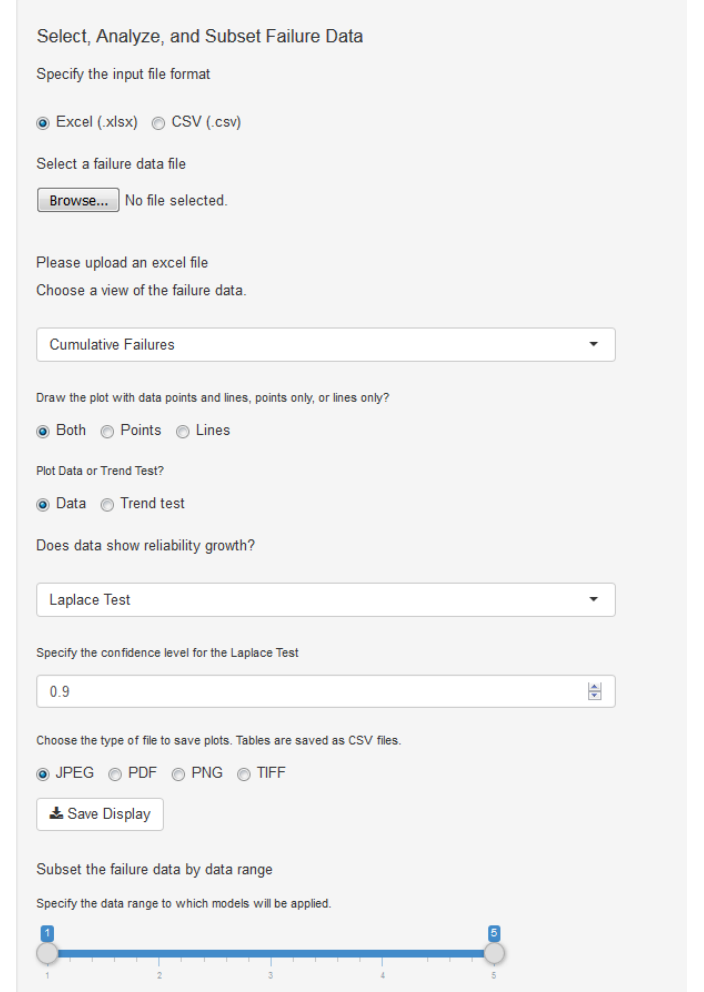


Fig. 2: Tab one options

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

Fig. 3: Excel input file format

Clicking on the **Choose File** button enables the user to browse and upload the input data file. The progress bar message "Upload complete" indicates a successful file upload. By default, a plot of the cumulative failures for the uploaded data set is displayed; Figure 4 shows the SYS1 data set [1].

Below the progress bar, the user is able to choose a sheet from a dropdown menu listing the data sets. csv files can

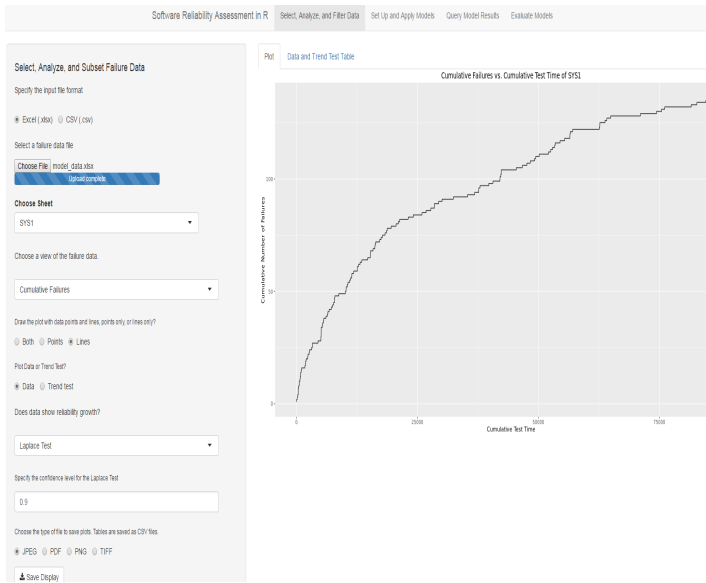


Fig. 4: Tab one after data upload

contain only one data set, while Excel files can contain one data set per sheet. Data sets that do not comply with the input format will not be available in the dropdown menu. During upload, the tool converts data into failure times, failure counts, and inter failure data formats regardless of the input format so that any model can be applied to any data set. 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 failure counts. This will enable more comprehensive comparison of models than ever before. Alternative data views include times between failures and failure intensity, which can be selected from the dropdown menu below the prompt to “Choose a view of the failure data”. All plots can be drawn using either points, lines or both.

The radio buttons **Data** and **Trend test** allow the user to switch between data and trend test plots. These trend tests have been placed before model fitting to ensure that the data exhibits reliability growth and it is therefore appropriate to apply software reliability models and make predictions. The two tests implemented include the Laplace trend test [17] and running arithmetic average.

Figure 5 shows the Laplace trend test of the SYS1 data set. The red line is a user specified input into the text box below the prompt to “Specify the confidence level for the Laplace Test” as shown in Figure 2. Here 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 it is therefore suitable to apply software reliability models.

Figure 6 shows the running arithmetic average of the SYS1 data set. Intuitively, if the time between failures increases then the running arithmetic average increases, indicating system reliability is improving. A decreasing running arithmetic average indicates reliability deterioration. Both the Laplace trend

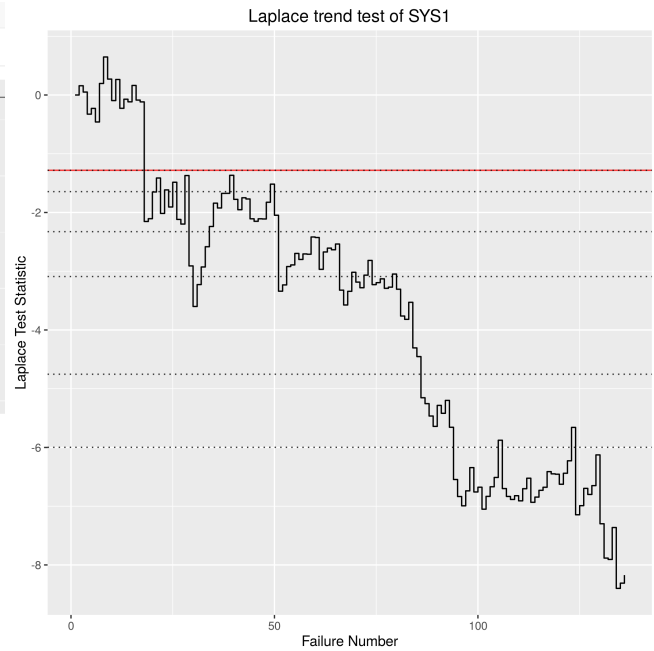


Fig. 5: Laplace trend test

test and running arithmetic average suggest the SYS1 data set exhibits reliability growth.

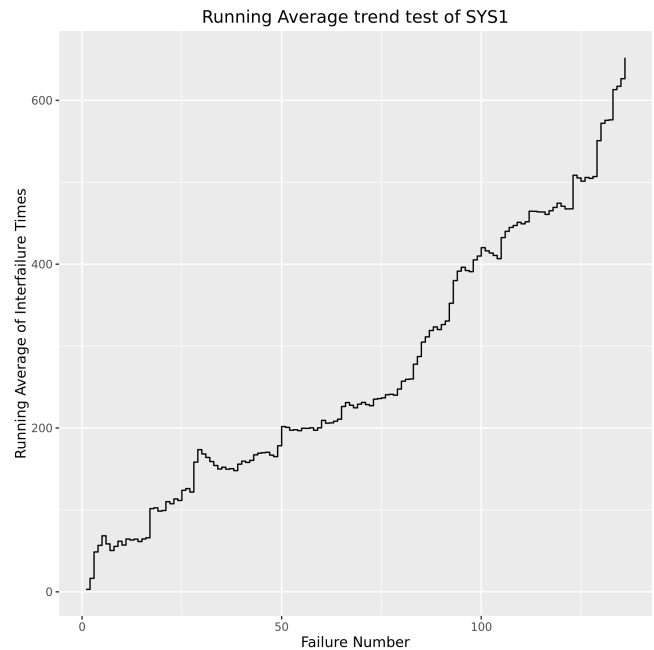


Fig. 6: Running arithmetic average

Figure 7 shows the Laplace trend test of the J4 data set which does not experience statistically significant reliability improvement. Thus, it is not appropriate to apply software reliability models to this data set until additional testing is performed that establishes 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.

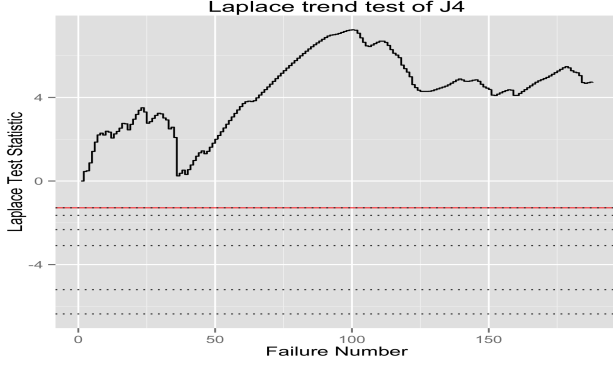


Fig. 7: Data set without reliability growth

The user can switch between the **Plot** and **Data and Trend Test Table** in Figure 4. Selecting table view displays the numerical data used to draw the plots in a tabular format similar to Figure 3. Selecting a radio button **.jpeg**, **.pdf**, **.png**, and **.tiff** (Figure 2) and then clicking **Save Display** saves a plot in the desired image format, while tables can be saved as csv or pdf. Tabular data can 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 options available on the second tab where model fitting is 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 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 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 model fits for all five models. The solid black vertical line indicates the time at which the last failure occurred. Thus, points to the right indicate predictions. 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 or under predict the number of failures at various stages of testing.

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

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

1

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.

No 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?

10000

☒ 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

C. Tab 3: Query Model Results

Figure 13 shows the options on the third tab, which enable a variety of predictions. 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 the number of failures in the text box below “Specify the number of failures that are to be observed.” 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 in this interval. Entering numbers into these text boxes automatically generates a table such as 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 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

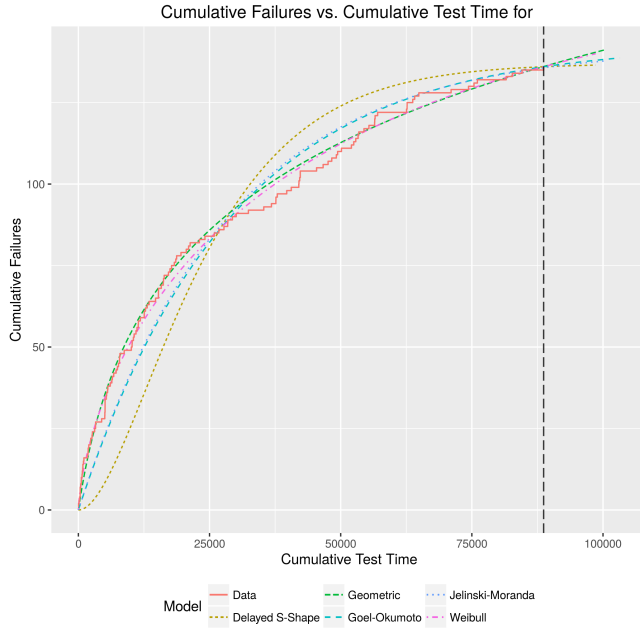


Fig. 9: SYS1 cumulative failures and model fits

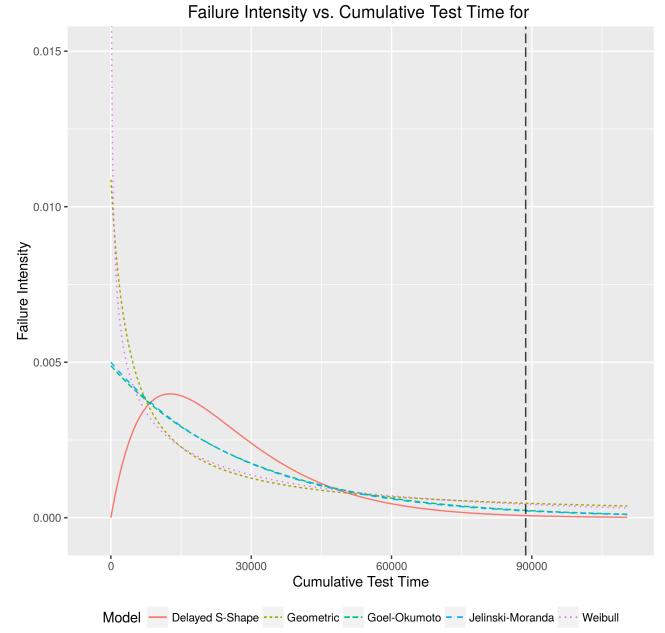


Fig. 11: SYS1 failure intensity and model fits

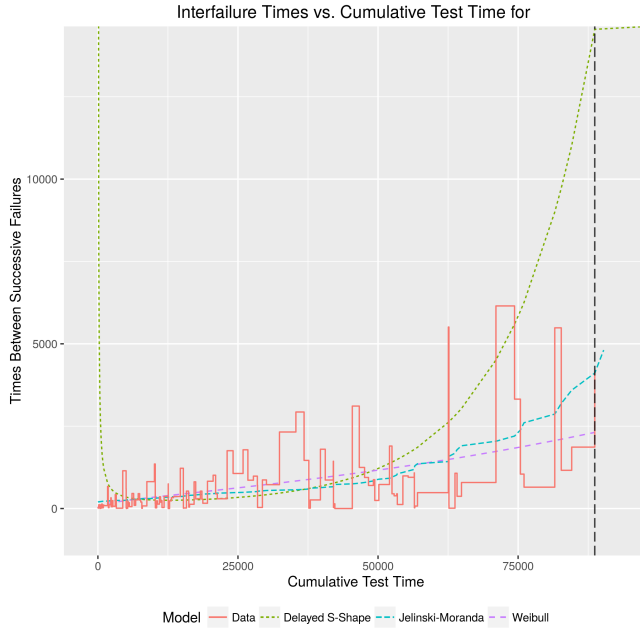


Fig. 10: SYS1 time between failures and model fits

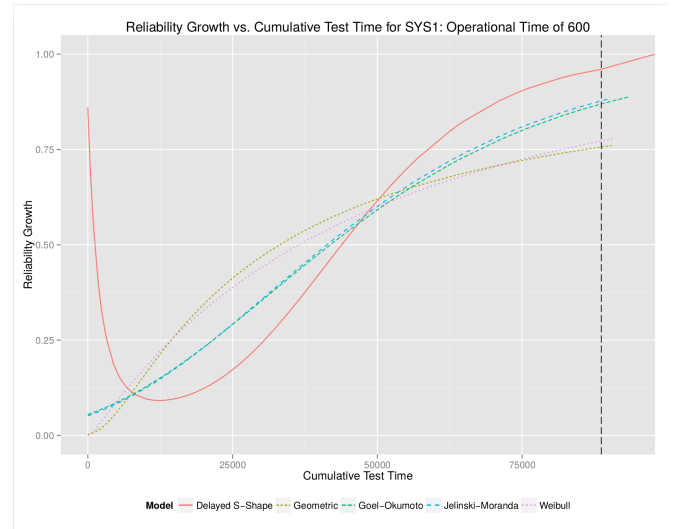


Fig. 12: SYS1 reliability growth and model fits

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 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 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.

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

III. AUTOMATED SCRIPT TO GENERATE SFRAT REPORT

The script is written in R programming language utilizing the Markdown library to generate the report. The script consists of 532 lines of code including comments and blank lines between each chunk. The script consists of two files namely **report-specifications.R** and **SFRATReport.Rmd**. The **SFRATReport.Rmd** is a R Markdown file used to create reports in different formats, which contains 493 lines of code. The **report-specifications.R** is an R script that allows the user to specify inputs necessary to run the SFRAT before generating a report, which contains 39 lines of code.

A. Installation

An automated installation script has been prepared and is available from the GitHub repository at: <https://github.com/LanceFiondella/SFRAT-Automated-Report/blob/master/installscript.R>.

The manual installation procedure is:

- Make sure that your platform is either 64-bit Windows 7 or later, Mac OS X 10.9 or later, or a version of Linux capable of running R Studio.
- Perl 5 version 16 or later. Linux and Mac computers usually have this installed, however, for Windows machines Perl can be downloaded from: <http://strawberryperl.com/>
- 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 later and RStudio 0.99.482 or later.

– R (<https://cran.r-project.org/>) needs to be installed before RStudio can be installed. Once RStudio has been installed, the following packages need to be installed.

- * **shiny** is a web application framework for R.
- * **gdata** is a package that provides various R programming tools for data manipulation.
- * **ggplot** is a graphical package, which offers a powerful graphics language for creating elegant and complex plots.
- * **DT** is a package that provides an R interface to the JavaScript library DataTables.
- * **rootSolve** is a package to find the roots of n nonlinear (or linear) equations.
- * **knitr** is a package that provides a general-purpose tool for dynamic report generation in R using Literate Programming techniques.
- * **rmarkdown** is a package that includes high level functions for converting R Markdown documents into a variety of formats including HTML, MS Word PDF, and Beamer.
- * **markdown** - is a package for authoring HTML, PDF, and MS Word documents.
- * **readxl** - is a package to import excel files.
- * **formatR** - is a package designed to reformat R code to improve readability.

– You can also install the packages using the Install packages menu item in RStudios Tools menu. This will bring up the dialog box in Figure 17:

Specify the package(s) you want to install in the first line of the dialog box. It is not necessary to change the installation location, so just leave the second line alone. Make sure that the “Install dependencies” box is checked if not, the packages you install may not work the way they should.

B. Script specifications

The user should then follow the steps described below to run the script:

- Download the contents from <https://github.com/LanceFiondella/SFRAT-Automated-Report> to your desired location on your computer and unzip the folder.
- Launch R Studio and set your working directory to the location where you have saved the downloaded folder. This can be done in one of the two following ways:
 - 1) Go to the menu option **Session**→**Set Working Directory**→**To source file location** to set the working directory to the file location.
 - 2) Run the following command on the console: `'setwd("/FileLocation")'`.
- Follow the steps described in Section III-A to make sure all the required packages are installed on your machine.
- Open the file ‘report-specifications.R and specify the required input parameters that you would have specified on the SFRAT user interface:

Model	Time to achieve R = 0.9 for mission of length 4116	Expected # of failures for next 100000 time units	Nth failure	Expected times to next 1 failures
All	All	All	All	All
Delayed S-Shape	12401.1541529981	0.9936777	1	NA
Geometric	1592716.45936287	31.6197191	1	2170.03088926781
Goel-Okumoto	62829.7672027733	6.6559071	1	4591.28466949961
Jelinski-Moranda	59023.2665608516	6.0584251	1	4956.13615407745
Weibull	259865.770847692	23.5552173	1	2353.05254648438

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

1) Input specifications:

- **datapath** - Specify the path to the location where input data is saved as an excel spreadsheet. This is on Line 12.
- **sheetNumber** - If the input data file has multiple datasets arranged on different sheets, then this option allows the user to specify a particular dataset. Otherwise, specify '1' on Line 13. It is recommended to name the sheet to distinctly include the data set name in the report.
- **colors** - User can specify the colors as a list on Line 17. If nothing is specified, default set of colors will be used to display graphical results.

2) Tab 1: Select, Analyze, and Filter Data

- **confidence_lvl** - Allows the user to specify a confidence level between 0 and 1 to quantify a desired level of significance that the data exhibits reliability growth using Laplace trend test. The variable is of type 'float' and can be specified on Line 20.

3) Tab2: Set Up and Apply Models

- **num_failures_future_prediction** - User can specify the number of failures that they would like to predict beyond the end of testing. The number of failures should be an integer value and can be specified in Line 23.

- **models_to_apply** - Allows the user to specify the list of software reliability growth models that they would like to apply to make predictions. Available models are delayed s-shape (DSS), geometric (GM), Goel-Okumoto (GO), Jelinski-Moranda (JM), and Weibull (Wei) models. Model selection should be specified as a list on Line 24.
- **mission_time** - User can specify the mission time beyond the end of testing for which they would like to see the reliability growth trend.

4) Tab3: 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' and should be specified on Line 28.
- **additional_time_software_will_run** - User can specify the additional time beyond end of testing to predict the number of failures.
- **desired_reliability** - User can specify the target reliability between 0 to 1 to estimate the time required to achieve that.
- **reliability_interval_length** - This is to specify the mission time beyond testing to estimate the reliability.

5) Tab4: Evaluate Models

- **percent_data_for_PSSE** - User can specify the percent of data to be used for model fitting to estimate the predictive capability of the specific model.

- After specifying the input parameters run the following command in the console to run the script: `source(report-specifications.R)` or you can click on source option on the topright corner as shown in Figure 18.
- Upon sourcing the file, an option to display verbose is offered as shown in Figure 19
Select 1 to display verbose report, '0' otherwise.
- Generated reports will be stored in the Report folder in the same location as the script. The file will have the following naming convention "SFRAT report_dataName_YYYY-MM-DD.pdf"

IV. ILLUSTRATIONS

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

	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	1950.541	24.39945
5	Weibull	1938.161	74.94496

Fig. 16: AIC and PSSE of all models

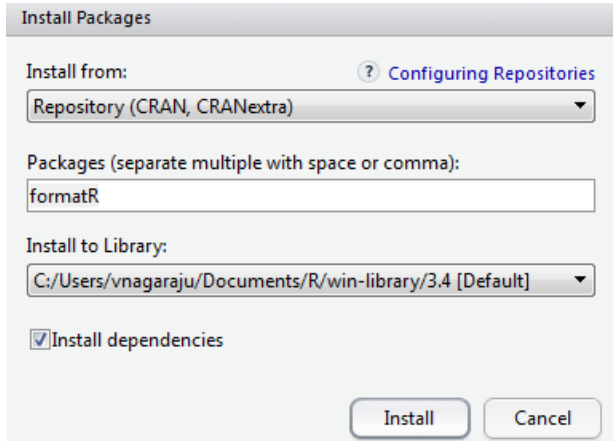


Fig. 17: Install packages dialog box

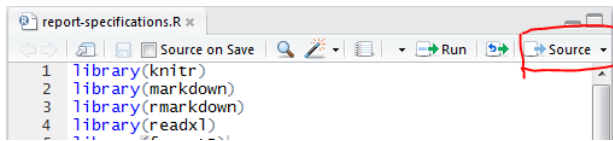


Fig. 18: Source script file

```

Display verbose report?

1: Type '1' for Yes
2: Type '2' for No

selection: |

```

Fig. 19: Verbose report

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.

A. Script on SYS1 data

B. Online analysis using the script on SYS1 data

V. CONCLUSION AND FUTURE WORK

ACKNOWLEDGMENT

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. Hudspohl, 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 Performance 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.