

Simple Plots Improve Software Reliability Prediction Models

John S. Lawson,^{1,*} Craig W. Wesselman,² and Del T. Scott¹

¹Department of Statistics, Brigham Young University, Provo, Utah, USA

²Ingenix Pharmaceutical Services, Basking Ridge, New Jersey, USA

ABSTRACT

Software reliability prediction is accomplished by fitting a nonhomogeneous Poisson process (NHPP) model to data from software testing. The data consist of the cumulative time and the cumulative number of failures found in software testing. The NHPP model can be used to predict the reliability of the software product at the time of release or to determine how much further testing must be done to reach a specified failure rate. Models are normally fitted to software testing data using Poisson regression by the method of maximum likelihood. We encountered difficulties fitting models when numerical algorithms failed to converge or when we were unable to discriminate among several models with the same number of parameters. These difficulties were the result of having no likelihood ratio test to compare models with the same number of parameters and anomalies in the data that caused numerical algorithms to fail. We found that a simple cumulative plot of the data (cumulative failures on the vertical axis vs. cumulative test time on the horizontal axis) helped in spotting anomalies in the data and selecting an appropriate model to fit. A second plot of running products of ratios of the probability densities for the predictions made from competing models, called the prequential likelihood ratio, helped in discriminating between models. Use of these plots helped resolve the difficulties we experienced in fitting models to the software testing data.

Key Words: Cumulative plot; Prequential likelihood ratio (PLR) plot; Non-homogeneous Poisson process (NHPP).

INTRODUCTION

The purpose of a software reliability prediction (SRP) is to predict the reliability of a software product at the time of release. Software is a critical component in systems that relate to nearly all aspects of modern life.

Software should be tested so that potential failures can be found and corrected until the field failure rate is reduced to an acceptable level. The SRP helps software producers predict how much more in-house testing is needed before their software products can be released to their customers. Along with cost considerations and functionality, SRP

*Correspondence: John S. Lawson, Department of Statistics, Brigham Young University, Provo, UT 84604, USA; Fax: (801) 378-7051; E-mail: lawson@byu.edu.

can help contractors in choosing between competing software products that perform similar functions.

To make predictions about software reliability, most SRP procedures use data from software testing. These data are modeled, and predictions are made. The data, collected from software testing, are usually in the form of the frequency of failures found and the time between each failure found. A nonhomogeneous Poisson process (NHPP), which has a varying interarrival time, is used to model this type of data since the failure intensity decreases as the software is tested and bugs are found and fixed.

For a homogeneous Poisson process (HPP), the probability of n events, or software failures, in the interval from t to $t + s$ is independent of t and is given by the Poisson distribution

$$P(n) = \frac{e^{-\lambda s} (\lambda s)^n}{n!} \tag{1}$$

where λ is called the *failure intensity function* or *failure rate*.

An NHPP differs from an HPP in that the failure rate or failure intensity function, $\lambda(t)$, of an NHPP varies over time. The expression for the probability of n events in the interval from t to $t + s$, for an NHPP, is a modification of Eq. (1) where the λs must be replaced by $\int_t^{t+s} \lambda(\tau) d\tau$. Healy, Jain, and Bennett^[1] propose several functional forms of $\lambda(t)$ that are useful in SRP (they are shown in Table 1 and include the Musa Basic, Duane, Littlewood, Littlewood–Verrall, and the Logarithmic Poisson model). These models are decreasing functions that approach zero asymptotically and are useful for modeling situations where the number of faults found and removed in the software decreases over time, since, to begin with, there is only a finite number.

Before and after fitting NHPP models to software testing data, we recommend making plots of the data. In this paper, we illustrate how a plot of the raw data from software testing can reveal problems with the data and help in selecting an appropriate model for the NHPP

intensity function $\lambda(t)$. We also illustrate how plots of the prequential likelihood ratio (PLR) can help in determining which of several models is making the best predictions of a data set.

CUMULATIVE PLOTS

A common plot that is used to represent data from repairable systems is the cumulative plot.^[2] In this plot, the cumulative number of repairs is plotted vs. the system age. This plot is useful in software reliability prediction. For software testing data this is a simple plot of the cumulative number of software faults discovered in testing on the vertical axis vs. the testing time on the horizontal axis. This plot is a staircase function and has discrete jumps on the vertical axis. But the population average of the cumulative function, denoted as $M(t)$, would be a smooth function of t and represents the *mean cumulative function*. The derivative of $M(t)$, $dM(t)/dt$ is $\lambda(t)$, the failure intensity function of the NHPP. The cumulative function then represents an estimate of the population mean cumulative function. Table 2 shows the population mean cumulative functions that would result from the models in Table 1.

The cumulative function can be used to determine if the data represents an HPP or an NHPP. If the cumulative plot forms a straight line, the failure intensity function $\lambda(t)$ is constant, indicating that the data represents a HPP. If, on the other hand, the cumulative plot looks like a concave or convex sequence of points, it indicates the data comes from an NHPP.

EXAMPLES OF CUMULATIVE PLOTS

Figs. 1 and 2 show the cumulative plots for two releases (3 and 4) of one software product. The most obvious characteristic of these plots is the gaps in the data. It is clear that testing was not performed every day. The gaps in the data correspond to weekends and

Table 1. Functional forms for failure intensity $\lambda(t)$.

Model	Parameters	Form of $\lambda(t)$
Musa Basic	a, b	ae^{-bt}
Duane	a, b	a/t^b
Littlewood	a, b, c	$\frac{a}{(bt+1)^c}$
Littlewood–Verrall	a, b	$\frac{a}{(bt+1)^{1/2}}$
Logarithmic Poisson	a, b	$\frac{a}{(bt+1)}$

Table 2. Functional forms for mean cumulative function $M(t)$.

Model	$M(t)$
Musa Basic	$\frac{a}{b}(1 - e^{-bt})$
Duane	$at^{1-b}/(1 - b)$
Littlewood	$(a[(bt + 1)^{1-c}/(b - bc)])$
Littlewood–Verrall	$(2a/b)[(bt + 1)^{1/2} - 1]$
Logarithmic Poisson	$\frac{a}{b} \log(bt + 1)$

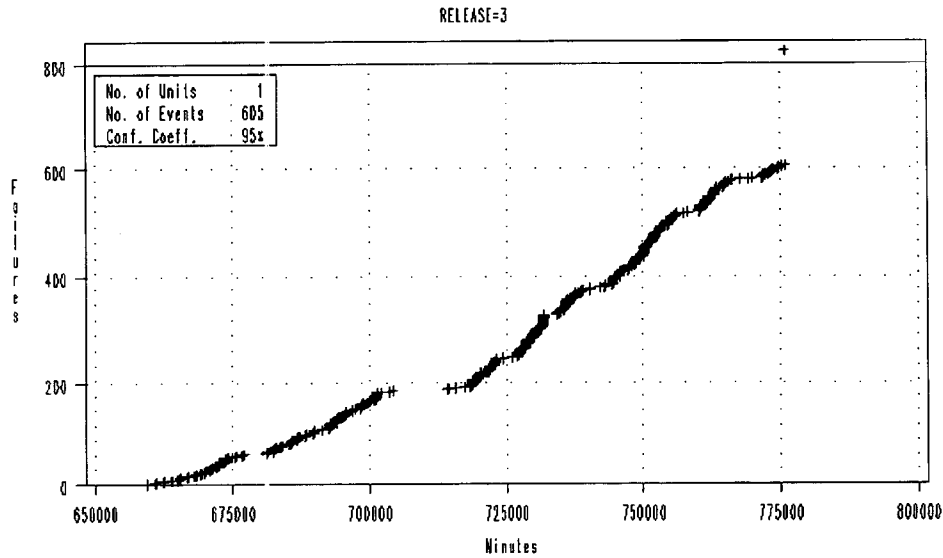


Figure 1. Cumulative plot for release 3.

holidays when testers did not work. After consulting a calendar to determine when the weekends and major holidays occurred, we removed from the data one day for each weekend day and holiday where no failures were found. If these gaps in the data are not removed, accurately fitting an NHPP model is more difficult and distinguishing between competing models is more complicated.

The second obvious characteristic of the cumulative plots in Figs. 1 and 2 is the S-shape that is particularly

evident in Fig. 2. The common NHPP intensity function models represented in Table 1 would have mean cumulative functions that are convex. However, the plots of the data show an initial concave appearance followed by a convex period. This may represent the fact that faults in the software are hard to find initially until the testers become accustomed to the software. Then failure detection intensity (or rate) increases until the removal of faults makes finding the few remaining faults difficult. Another reason this S-shape may occur is

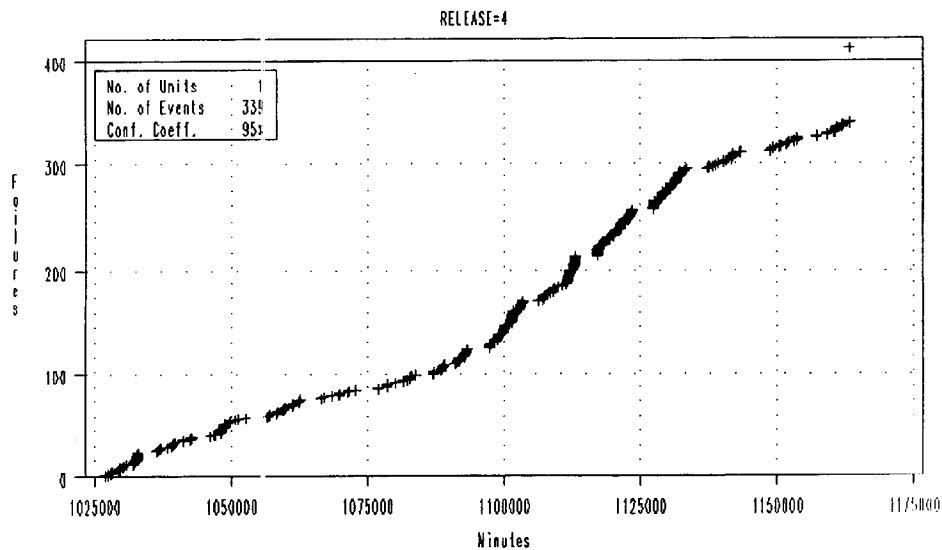


Figure 2. Cumulative plot for release 4.

because, at the beginning of testing, most of the effort in the production process is geared toward finishing the code of the program, which results in low initial test effort. Similarly, at the end of the production process, resources may be shifted from testing to preparing the product for shipping, decreasing the test effort.

Farr^[3] proposes an "S-shaped reliability growth model" (S-shaped), and its failure intensity and mean cumulative function are given in the following equations.

$$\lambda(t) = ab^2te^{-bt} \quad (2)$$

$$M(t) = a[1 - (bt + 1)e^{-bt}] \quad (3)$$

This model may be more appropriate for the data of Figs. 1 and 2 than the models given in Tables 1 and 2. The simple cumulative plot of the failure data indicates this fact.

FITTING NHPP MODELS TO SOFTWARE RELIABILITY DATA

Once a model has been chosen, the NHPP model can be fitted to the data by the method of maximum likelihood. The data (after removal of gaps like those shown in Figs. 1 and 2) consist of two columns. One column is the cumulative failure count and the other is the cumulative testing time. Table 3 shows an example of the first four failures from the data plotted in Fig. 1. In this example, the first failure occurred at 0.85 hours, the second failure, at 1.71 hours, etc.

Maximum likelihood fitting of a model, like the S-shaped shown in Eqs. (2) and (3), can be accomplished using software such as SAS[®] PROC NLMIXED, which can fit nonlinear mixed models with random effects following the normal, binomial, or Poisson distributions. An example of the SAS commands to fit the model to the data in Table 3 and Fig. 1 are:

```
Data Tab3;
input M t;
cards;
1 0.85
2 1.71
: :
proc nlmixed;
parms a 1000 b .01;
M = a*(1 - (b*t + 1)*exp(-b*t));
model count ~ poisson(M);
```

Table 3. First four lines of data plotted in Fig. 1.

$M(t) = \text{count}$	$t = \text{hours}$
1	0.85
2	1.71
3	1.85
4	1.87
⋮	⋮

Lawson and Meade^[4] show how the maximum likelihood fitting can be accomplished without special purpose software using the solver (or nonlinear maximization tool) available in common spreadsheet programs such as Microsoft EXCEL or Corel Quatro Pro.

COMPARING FITS OF VARIOUS MODELS

When fitting different NHPP models to the same data set, it is helpful to evaluate which model provides a better fit and will be more reliable for making predictions. No likelihood ratio tests are available to compare models like those in Table 1, Table 2, and Eqs. (2) and (3), since most of these models have the same number of parameters and neither is a subset of the other. Simple graphs of the actual and predicted values are sometimes helpful. For example, Fig. 3 shows that the S-shaped model provides a more reasonable fit to the data from release 3 than the Littlewood-Verrall model. But, in other cases, it is difficult to distinguish which model provides a better fit, as Fig. 4 shows.

In order to find out which model is making the best predictions, Brocklehurst and Littlewood^[5] have proposed a general technique, the PLR. This technique is used to compare two models, A and B, by predicting the next time to failure t_j and comparing how close each model is to the actual time of failure. In order to use this technique, a few observations (20–30) are removed from the end of the data set to be used for comparing predictions. For each model being compared, the parameter estimates made with the remaining data are used to get the predicted failure intensity function, $\hat{\lambda}(t_j)$, at each of the observations, t_j , left out of the fitting. These estimated failure intensities are used to calculate the estimated (pdf) $\hat{f}(t_j) = \hat{\lambda}(t_j)^{M_j} e^{-\hat{\lambda}(t_j)} / (M_j)!$ where M_j is the cumulative failures by time, t_j . These pdf's are calculated for each of the observation times, t_j , left out of the fitting. If model A's predictions are closer to the actual than model B's, model A's pdf, $\hat{f}_A(t_j)$ at t_j , will be larger, and thus the ratio of $\hat{f}_A(t_j) / \hat{f}_B(t_j)$ will be larger

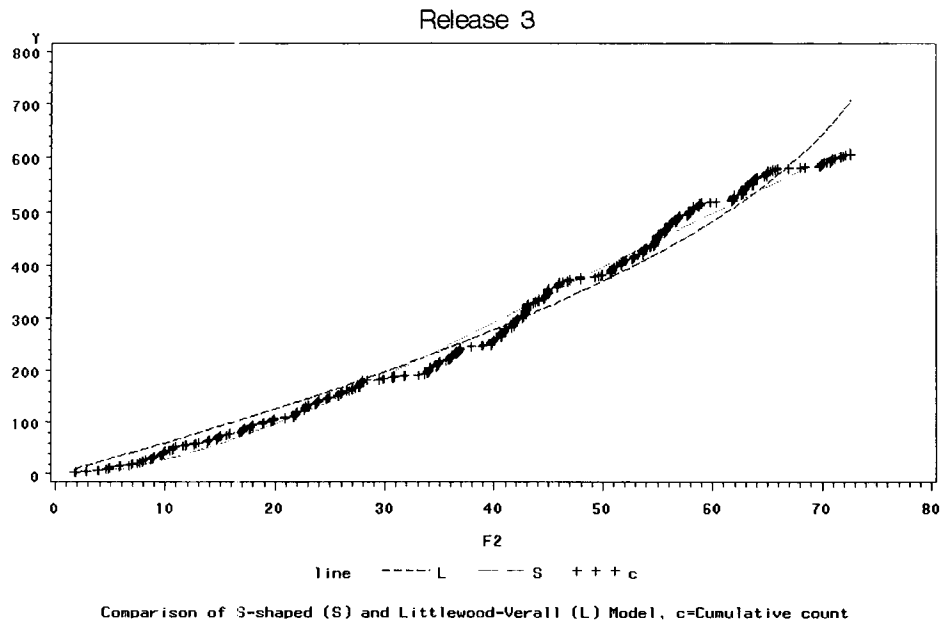


Figure 3. Comparison of fitted models.

than one. The PLR is a running product of the ratios of pdf's for many predictions:

$$PLR^{AB} = \prod_{j=1}^n \frac{\hat{f}_A(t_j)}{\hat{f}_B(t_j)} \quad (4)$$

where n is the number of observations reserved for predictions, and each better prediction for model A increases the PLR, and conversely each better prediction for model B decreases the PLR. Although the final PLR is useful, a plot of the intermediate

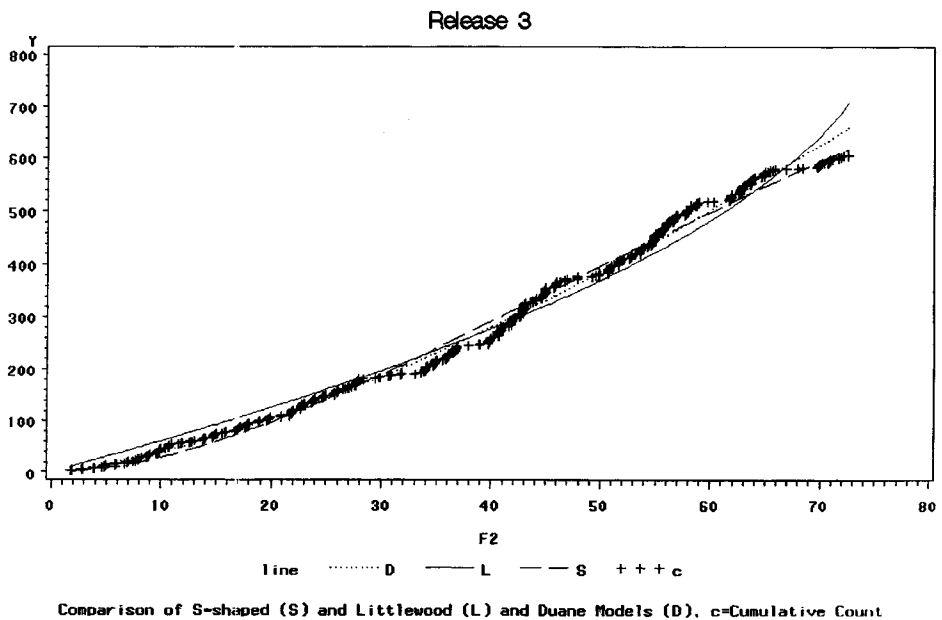


Figure 4. Comparison of fitted models.

Table 4. Intermediate calculations for PLR comparing S-shaped model to Duane model.

t_i Cum days	M_j Cum fail	λ_A Duane	λ_B S-shaped	$\text{Log}(f_A/f_B)$	(5) PLR_j
69.868	587	13.087	10.320	136.63	136.63
69.908	588	13.090	10.318	137.15	273.78
70.047	589	13.102	10.310	138.36	412.14
70.055	590	13.102	10.309	138.66	550.80
70.389	591	13.130	10.289	141.26	692.06
70.653	592	13.152	10.273	143.38	835.44
70.831	593	13.167	10.262	144.89	980.33
70.849	594	13.169	10.261	145.27	1125.60
70.860	595	13.170	10.261	145.61	1271.21
70.899	596	13.173	10.258	146.14	1417.35
70.937	597	13.176	10.256	146.66	1564.01
71.058	598	13.186	10.248	147.78	1711.79
71.653	599	13.235	10.211	152.35	1864.14
71.685	600	13.238	10.209	152.83	2016.97
71.723	601	13.241	10.207	153.37	2170.34
71.855	602	13.252	10.199	154.60	2324.94
71.856	603	13.252	10.199	154.86	2479.81
72.092	604	13.272	10.184	156.87	2636.67
72.622	605	13.315	10.150	161.04	2797.72
72.642	606	13.317	10.149	161.46	2959.18

predictions

$$\log(PLR_j^{AB}) = \sum_{i=1}^j [M_i \log \hat{\lambda}_A(t_i) - M_i \log \hat{\lambda}_B(t_i) + \hat{\lambda}_B(t_i) - \hat{\lambda}_A(t_i)] \quad (5)$$

over time shows how these intermediate values formed the final PLR. Many models can be compared using this technique by selecting one model as a reference and comparing all other models to it.

As an example of the calculations for the PLR, Table 4 shows the intermediate calculations for comparing the S-shaped model to the reference Duane model for 20 observations left out of the estimation for release 3. The estimated parameters used in these calculations were $\hat{a} = 1.955$, $\hat{b} = 0.4477$ for the Duane Model and $\hat{a} = 1490$, $\hat{b} = 0.02008$ for the S-shaped model. These were calculated by fitting the models to the data excluding the final 20 observations shown in Table 4.

EXAMPLE OF PLR PLOTS

When fitting models to the software testing data from releases 3 and 4, 20 and 30 observations, respectively, were removed from the data used to fit the models and

saved for comparing predictions of the models. Figures 5 and 6 show the plots of the log of the PLR for the removed data from release 3 and release 4. In these plots, we represent the reference model (Duane) by a straight line at zero. The calculations for the S-shaped model in Fig. 5 are shown in Table 4. Any model whose log(PLR) is more positive is making better predictions, while more negative log(PLR)s indicate worse predictions. In Fig. 5, it can be clearly seen that the S-shaped model fits the data of release 3 better, while Fig. 6 shows the S-shaped and reference Duane models seem to fit the release 4 data equally well.

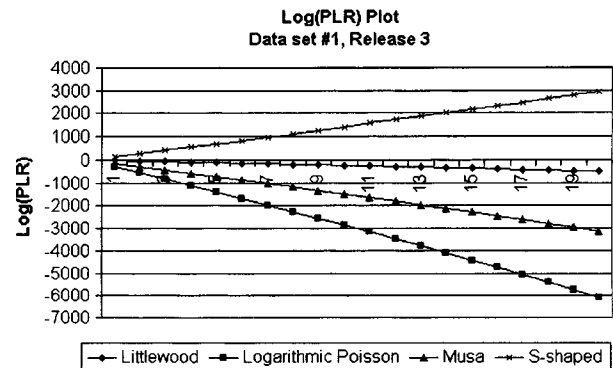


Figure 5. PLR plot for release 3.

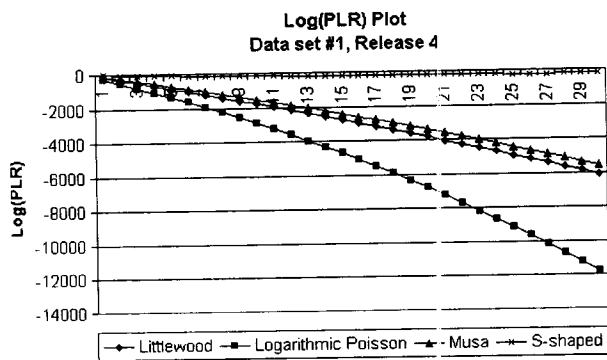


Figure 6. PLR plot for release 4.

SUMMARY AND CONCLUSIONS

In this paper, we describe two types of plots: 1) the cumulative plot and 2) the PLR plot, that were helpful to us in modeling data from software testing. The cumulative plot revealed gaps in the data that resulted when testing was not done every day. The cumulative plot also revealed the S-shaped appearance of the mean cumulative function, $M(t)$ and prompted us to consider models we previously had not considered. The PLR plot made it easier to judge which models fit the software testing data best.

ABOUT THE AUTHORS

Dr. John Lawson is an associate professor in the Department of Statistics at Brigham Young University. He received his Ph.D in applied statistics from the Polytechnic Institute in New York. He currently teaches courses in engineering statistics, reliability

engineering, and design of experiments. He is a senior member of ASQ.

Craig Wesselman is currently a statistician at Ingenix Pharmaceutical Services in Basking Ridge, New Jersey. He received his M.S. degree from Brigham Young University. He assists in designing clinical trials in the pharmaceutical industry and in analyzing and reporting the results of the trials to clients and the Food and Drug Administration.

Dr. Del T. Scott is currently a professor in the Department of Statistics at Brigham Young University. He received his Ph.D in statistics at Pennsylvania State University. He currently teaches courses in mathematical statistics and statistical computing. His interests lie in the area of statistical computing.

REFERENCES

1. Healy, J.; Jain, A.; Bennett, J. Reliability prediction. *Annual RELIABILITY and MAINTAINABILITY Symposium 1996 TUTORIAL NOTES*, Las Vegas, NV, 1996.
2. Tobias, P.; Trindade, D. *Applied Reliability*; Van Nostrand Reinhold: New York, N.Y., 1995; 306.
3. Farr, W. Software reliability modeling survey. In *Handbook of Software Reliability Engineering*; Lyu, M.R.Eds., Ed.; McGraw Hill: New York, 1996; 71–119.
4. Lawson, J.; Meade, D. Calculating maximum likelihood estimates of reliability parameters using spreadsheets. *Qual. Eng.* **1998–1999**, *11* (1), 44–53.
5. Brocklehurst, S.; Littlewood, B. Techniques for prediction analysis and recalibration. In *Handbook of Software Reliability Engineering*; Lyu, M.R.Eds., Ed.; McGraw Hill: New York, 1996; 119–164.