Software Reliability
&
Intervaldata

Dennis Timmers

23rd August 2004
Summary
## Contents

1 Introduction ........................................ 3

2 Geometric Model .................................. 4

3 Renewal model ...................................... 6

4 NHHP model ........................................ 8
   4.1 Estimating parameters ....................... 10
   4.2 The total number of tests ................... 10
   4.3 Number of tests between detections .......... 11
   4.4 Interpretation of model ..................... 12

5 Application of NHHP model on real datasets .... 13
   5.1 Dataset 1 .................................. 13
   5.2 Dataset 2 .................................. 14

6 Interval data ...................................... 15
   6.1 Theory ...................................... 15
   6.2 Program .................................... 15
   6.3 Results ..................................... 16

7 Conclusions ....................................... 17

A Mathematica code ................................. 18
   A.1 Plot of MLE .................................. 18
   A.2 Interval data ................................ 19

B Table of Notation ................................. 20
1 Introduction

De problemen die ik nog heb:

- type I censored data or type II censored data zie niet in waar ik dat kan behandelen bij de likelihood estimation
- krijg formule of pagina 6 niet rond
- bij $\chi^2$ test krijg ik geen voetnoten
- bij de tweede dataset krijg ik allemaal foutmeldingen bij het maken van plots van de likelihood en de punten die wel geplot worden geven allemaal een likelihood van 0.
2 Geometric Model

The idea originates from ??????. Suppose $a$ is the unknown number of errors in a software system. The software system is subjected to a series of tests. A test is a walk through the state space of the system. Each walk has a finite length. A walk ends either if an end state is reached or if an error has occurred. We assume that the walks have been designed in such a way that each undetected error has a probability $\theta$ to be on the walk. So each error has probability $\theta$ to be detected during a test. When there are $r$ remaining errors, the probability of not detecting an error is $(1-\theta)^r$.

The probability to detect the first error (during one test), $p_1$, equals $1-(1-\theta)^a$ since all errors are still undetected. The probability to detect the second error equals $1-(1-\theta)^a(1-\theta)$ since all errors are undetected but one (which has been repaired). This means that the probability to detect the $i^{th}$ error equals $1-(1-\theta)^a(i-1)$.

$X_i$ equals the number of tests to detect the $i^{th}$ error after the $(i-1)^{th}$ has been repaired. Clearly $X_i$ is geometrically distributed with succes parameter $p_i$.

The model above is restricted so that each error has the same probability to be detected. But what happens if the errors in the program are divided into several classes. Suppose there are $k$ classes of errors, each class has a different total number of errors $a_i$ and a different detection probability $\theta_i$. Define $T_{ij}$ as the number of tests necessary to detect an error from class $i$ when already $j-1$ errors of that class are found. Analogue with $X_i$ $T_{ij}$ is geometrically distributed with succes parameter $p_{ij} (= 1-(1-\theta_i)^{a_i-j+1})$

With this model $X_i$ is now the minimum of $k$ geometrically distributed random variables $T_{i(1)}, \ldots, T_{i(k)}$.

What is the distribution of $X_i$? It will turn out that the minimum of several geometrically distributed random variables is again geometrically distributed.

Suppose $S_i$ ($1 \leq i \leq l$) is geometrically distributed with succes parameter $\delta_i$ and $W$ is the minimum of $S_1, \ldots, S_l$ then

$$P(W > x) = P(S_1 > x \cap \ldots \cap S_l > x)$$

$$= \prod_{i=1}^{l} P(S_i > x)$$

$$= \prod_{i=1}^{l} (1-\delta_i)^x$$

$$= \left( \prod_{i=1}^{l} (1-\delta_i) \right)^x$$

then $W$ is geometrically distributed with succes parameter $1-\prod_{i=1}^{l} (1-\delta_i)$.

Since there are different classes there are different paths possible to detect errors, question is if the order in which errors are found is essential for the distribution of the $X_i$’s. With a simple example it is easy to show that this is indeed the case.

Suppose there are two classes which each have one error, so there are in total two errors. The error in class one has detection probability $\frac{1}{2}$, while the error in class two has detection probability $\frac{1}{4}$.

Make the following observations:

$$T_{11} \sim GEO(p_{11}) = GEO(1-(1-\theta_1)^{a_1-1+1}) = GEO(\frac{1}{2})$$

$$T_{21} \sim GEO(p_{21}) = GEO(1-(1-\theta_2)^{a_2-1+1}) = GEO(\frac{1}{4})$$

There are two paths possible to detect all errors, with both paths the distribution of $X_1$ and $X_2$ are calculated. $X_1$ is the minimum of $T_{11}$ and $T_{21}$ and thus geometrically distributed with succes
parameter $1 - \prod_{i=1}^{2}(1 - p_{1i})$.

**path 1** first the error of class 1 is detected and then the error of class 2

$X_1 \sim \text{GEO}(v)$

$v = 1 - \prod_{i=1}^{2}(1 - p_{1i}) = 1 - (1 - \frac{1}{2})(1 - \frac{1}{4}) = \frac{5}{8}$

Since there aren’t any errors left in class 1 and only one error of class 2

$X_2 = T_{21} \sim \text{GEO}(\frac{1}{4})$

**path 2** first the error of class 2 is detected and then the error of class 1

$X_1 \sim \text{GEO}(v)$

$v = 1 - \prod_{i=1}^{2}(1 - p_{1i}) = 1 - (1 - \frac{1}{2})(1 - \frac{1}{4}) = \frac{5}{8}$

Since there aren’t any errors left in class 2 and only one error of class 1

$X_2 = T_{11} \sim \text{GEO}(\frac{1}{4})$

It is hard to obtain results with this model since it matters in which order the errors are detected. A way to avoid this problem is to keep track from which class each found error is. This isn’t a very practical approach but there is another way. Take into account all possible ways to detect the total of $\sum_{i=1}^{k} a_i$ faults in the program. Compute the probability of each different path to occur. With this approach all possible $X_i$’s and the probability of every path have to be calculated.
3 Renewal model

An attempt is made to describe the geometric model with only one class with a Poisson process. First we will look at the simplification when all \(X_i\)’s are independent identically distributed.

A Poisson process is a counting process for which the times between successive events are independent and identically distributed exponential random variables. One possible generalization is to consider a counting process for which the times between successive events are independent and identically distributed with an arbitrary distribution. Such a counting process is called a renewal process.

For an arbitrary renewal process denote

\[
S_0 = 0, \quad S_n = \sum_{i=1}^{n} X_i, \quad n > 0
\]

Let \(\{N(t), t \geq 0\}\) be a counting process. With the geometric model all the \(X_i\)’s are geometrically distributed but all of them have another succes parameter. The renewal process requires that all the \(X_i\)’s are i.d.d., the model which comes the nearest by the geometric model is then when all \(X_i\) are i.d.d. \(\text{GEO}(p)\).

The distribution of \(N(t)\) has to be computed, note that

\[
P\{N(t) = n\} = P\{N(t) \geq n\} - P\{N(t) \geq n + 1\} \\
= P\{S_n \leq t\} - P\{S_{n+1} \leq t\}
\]

If the distribution of \(S_n\) is known in order to obtain the distribution of \(N(t)\). In this case is \(S_n\) a summation of geometrically distributed random variables. To find out which distribution fits to \(S_n\) Laplace transformation is used.

\[
\phi_{S_n}(t) = E[e^{tS_n}] \\
= E[e^{\sum_{i=1}^{n} X_i}] \\
= \prod_{i=1}^{n} E[e^{tX_i}] \\
= \prod_{i=1}^{n} E[e^{tX_i}]
\]

\(E[e^{tX_i}]\) is the Laplace transformation of a geometrically distributed random variable and equals \(\frac{pt}{1 - (1-p)t}\).

\[
\phi_{S_n}(t) = \prod_{i=1}^{n} \frac{pt}{1 - (1-p)t} = \left( \frac{pt}{1 - (1-p)t} \right)^n
\]

\(\phi_{S_n}(t)\) belongs to a negative binomial distribution so \(S_n \sim \text{NB}(n, p)\). There is one condition though, at each test only one error can be detected. So it is impossible to detect \(n\) errors in less then \(n\) tests. When this is combined to following distribution appears.

\[
P(S_n = k) = \begin{cases} 
(\binom{k-1}{n-1})p^n(1-p)^{k-n}, & k \geq n \\
0, & k < n
\end{cases}
\]

Thus we have

\[
P\{N(t) = n\} = \sum_{k=n}^{\lfloor t \rfloor} \binom{k-1}{n-1} p^n(1-p)^{k-n} - \sum_{k=n+1}^{\lfloor t \rfloor} \binom{k-1}{n} p^{n+1}(1-p)^{k-n-1}
\]
The outcome of this model is very simple but the problem is that the $X_i$’s have to be independent identically distributed. This assumption does not count in the geometric model with only one detection probability. The probability that an error is detected will be a decreasing function in time. A nonhomogeneous Poisson process allows this kind of behavior and will be the next model in line.

$$?? = \binom{\lfloor t \rfloor}{n} p^n (1 - p)^{\lfloor t \rfloor - n}$$
4 NHHP model

Consider the geometric model with only one class, so all errors in the program have the same detection probability \( \theta \). The probability that an error will be detected will be an decreasing function in time thus this process can not be described by a regular Poisson process. An attempt is made with the use of a nonhomogeneous Poisson process which allows that events may be more likely to occur at certain times than during other times.

First look at the case of a continuous process. Let \( \{N(t), t \geq 0\} \) denote a counting process representing the number of events occurred at time \( t \). Assume that this counting process is an inhomogeneous Poisson Process. Let \( C(t) \) denote \( E[N(t)] \) and \( \lambda(t) \) the intensity function, \( C(t) = \int_0^t \lambda(y)dy \), then

(i) \( N(0) = 0 \)

(ii) \( \{N(t), t \geq 0\} \) has independent increments, i.e. for any collection of the numbers of test runs \( 0 < t_1 < t_2 < \ldots < t_k \), the \( k \) random variables \( N(t_1), N(t_2) - N(t_1), \ldots, N(t_k) - N(t_{k-1}) \) are statistically independent.

(iii) \( P\{N(t+h) - N(t) \geq 2\} = o(h) \)

(iv) \( P\{N(t+h) - N(t) = 1\} = \lambda(t)h + o(h) \)

With the help of these four properties it can be shown that (see [2])

\[
P\{N(s+t) - N(s) = n\} = e^{-[C(s+t)-C(s)]} \frac{[C(s+t) - C(s)]^n}{n!}, \quad n \geq 0 \tag{1}
\]

In our model with only one detection probability the time will be measured in the number of tests done sofar. The time is discrete the counting process \( \{N(t), t = 0, 1, 2, \ldots\} \) denotes the cumulative number of errors detected after \( t \) tests. Not all four conditions will be necessary anymore but (1) will still hold.

According to (1) the distribution of \( N(t) \) is known when the function \( C(t) \) is known, therefore try to obtain the function \( C(t) \) with the help of a recurrent relation.

\[
C(t+1) - C(t) = E[N(t+1) - N(t)] = E[N(t+1)] - E[N(t)]
\]

During one test at most one error can be detected, so

\[
C(t+1) - C(t) = 1-P(\text{error is detected}) + 0-P(\text{no error is detected}) = P(\text{error is detected})
\]

In the interval \([0,t]\) sofar \( N(t) \) errors are already detected, thus

\[
P(\text{error is detected}) = p_{N(t)} = 1 - (1-\theta)^{a-N(t)+1}
\]

The recurrent relation becomes

\[
C(t+1) - C(t) = 1 - (1-\theta)^{a-N(t)+1} \tag{2}
\]

This recurrent relation can never be solved because of the term \( N(t) \) in the exponent. \( N(t) \) equals (1) and so there will be a term \( C(t) \) in the exponent. There is no known solution for these kind of recurrent relations.

Since the approach with the recurrent relation does not work the recurrent relation is adapted. The idea to use the following recurrent relation can be found in [3]. The expected number of software errors detected between the \( n \)-th and \( (n+1) \)-st test runs is assumed to be proportional to the expected number of remaining errors after the \( n \)-th test run, i.e.

\[
C(t+1) - C(t) = b(a - C(t)), \quad 0 < b < 1
\]
Where \( b \) is a proportional constant. The parameter \( b \) is interpreted as the error detection rate per error.

A consequence of this model it isn’t impossible anymore that during one test more than one error is detected. This recurrent relation is known as a first degree recurrent relation with constant coefficients. Solving the recurrent equation (3) is done with the help of generative functions. For more background information see ([1]).

\[
e_{m+1} = (1 - b)e_m + ab
\]  

(3)

Multiply both sides with \( x^{m+1} \)

\[
x^{m+1}e_{m+1} = x(x^m(1 - b)e_m + abx^m)
\]

This equation must count for all \( m = 0, 1, 2, \ldots \), sum up over all these values of \( m \)

\[
\sum_{m=0}^{\infty} x^{m+1}e_{m+1} = x \left( (1 - b) \sum_{m=0}^{\infty} c_m x^m + ab \sum_{m=0}^{\infty} x^m \right)
\]

Which is the same as

\[
\sum_{m=0}^{\infty} x^mc_m - c_0 = x \left( (1 - b) \sum_{m=0}^{\infty} c_m x^m + ab \sum_{m=0}^{\infty} x^m \right)
\]

\( \sum_{m=0}^{\infty} c_m x^m \) is called the generative function of \( c_m \) and will be denoted as \( c(x) \). Remember that \( c_0 = 0 \) since there can not be found an error before at least one test is executed. The generative function of \( \sum_{m=0}^{\infty} x^m = \frac{1}{1-x} \), then

\[
c(x) = \frac{x(1-b)c(x) + abx}{1-x}
\]

\[
c(x) = \frac{abx}{1-x} \cdot \frac{1}{1-x(1-b)}
\]

The translation back to \( c(x) = \sum_{m=0}^{\infty} c_m x^m \) is not trivial so try to write down the first 4 terms of the Taylor series, this is done with the help of the program Mathematica. Maybe a formula for \( c_m \) can be filtered out of these 4 coefficients.

<table>
<thead>
<tr>
<th>coefficient</th>
<th>formula</th>
<th>simplification</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>ab</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>-a(-2+b)b</td>
<td>a(1-(1-b)^2)</td>
</tr>
<tr>
<td>3</td>
<td>ab(3+(-3+b)b)</td>
<td>a(1-(1-b)^3)</td>
</tr>
</tbody>
</table>

We suspect that \( c_m = a(1 - (1-b)^m) \), substitute this form into (3) then

\[
e_{m+1} = a(1 - (1-b)^{m+1})
\]

\[
= a - a(1-b)^{m+1} - ab + ab
\]

\[
= a(1-b) - a(1-b)^{m+1} + ab
\]

\[
= (1-b)(a - a(1-b)^m) + ab
\]

\[
= (1-b)e_m + ab
\]

And the final conclusion is that

\[
C(t) = a(1 - (1-b)^t), \quad (t = 0, 1, 2, \ldots)
\]

(4)

Therefore this error detection process can be described by an Inhomogeneous Poisson Process with \( C(t) \) like in (4). The probability that at time \( t \) already \( n \) errors are detected is

\[
P(N(t) = n) = \frac{[a(1 - (1-b)^t)]^n}{n!}e^{-a(1-(1-b)^t)}
\]

(5)
4.1 Estimating parameters

For observed data the model parameters can be estimated. A way to do this is with the Maximum Likelihood Estimation. Suppose the data is available in the form of pairs \((t_i, x_i)(i = 1, 2, \ldots, k)\) where \(x_i\) is the total number of errors detected during \(t_i\) test runs.

The likelihood function is given by

\[
L = P[N(t_1) = x_1, N(t_2) = x_2, \ldots, N(t_k) = x_k] = 
\prod_{i=1}^{k} \frac{\{C(t_i) - C(t_{i-1})\}^{x_i-x_{i-1}}}{(x_i-x_{i-1})!} \exp[C(t_i) - C(t_{i-1})]
\]

\[= \exp[-C_0] \prod_{i=1}^{k} \frac{\{C(t_i) - C(t_{i-1})\}^{x_i-x_{i-1}}}{(x_i-x_{i-1})!}
\]

where \(t_0 = x_0 = 0\). Substitute to (4) then

\[
L = \exp[-a(1 - (1 - b)^{t_k})] \prod_{i=1}^{k} \frac{\{a((1 - b)^{t_{i-1}} - (1 - b)^{t_{i}})\}^{x_i-x_{i-1}}}{(x_i-x_{i-1})!}
\]

(6)

Then the log-likelihood function for given data \((t_i, x_i)(i = 1, 2, \ldots, k)\) is

\[
\ln L = \sum_{i=1}^{k} (x_i - x_{i-1}) \ln a + \sum_{i=1}^{k} (x_i - x_{i-1}) \ln[(1 - b)^{t_{i-1}} - (1 - b)^{t_{i}}]
\]

\[
- a(1 - (1 - b)^{t_k}) - \sum_{i=1}^{k} \ln[(x_i - x_{i-1})!]
\]

(7)

From (7) the maximum likelihood parameters \(\hat{a}\) and \(\hat{b}\) can be estimated by solving simultaneously the equations \(\frac{\delta \ln L}{\delta a} = \frac{\delta \ln L}{\delta b} = 0\).

\[
\frac{\delta \ln L}{\delta a} = \frac{x_k}{\hat{a}} - t_k(1 - b)^{t_k}
\]

\[
\frac{\delta \ln L}{\delta b} = \sum_{i=1}^{k} \frac{(x_i - x_{i-1})[t_i(1 - b)^{t_{i-1}} - t_{i-1}(1 - b)^{t_{i-1}}]}{(1 - b)^{t_{i-1}} - (1 - b)^{t_{i}}} - a \cdot t_k(1 - b)^{t_k-1}
\]

4.2 The total number of tests

When testing it is important to know how many tests are necessary to find all errors. Define \(T\) as the number of tests necessary until all errors are found. It is not enough if at time \(t\) all errors are found because that does not exclude the possibility that all errors already were found after \(t - 1\) tests. Only when all errors are found after \(t\) tests and at time \(t - 1\) there were still errors in the program it is sure that the total test time equals \(t\).

This means

\[
P(T = t) = P(N(t) = a \cap N(t - 1) < a)
\]

The formula found above can be modified

\[
P(N(t) = a \cap N(t - 1) < a) = \sum_{i=1}^{a} P(N(t) - N(t - 1) = i \cap N(t - 1) = a - i)
\]
According to the second property of a Nonhomogeneous Poisson model the number of errors found in \([0, t - 1]\) and the number of errors found in \([t - 1, t]\) are independent.

\[ P(N(t) - N(t - 1) = i) \cap N(t - 1) = a - i = P(N(t) - N(t - 1) = i) \cdot P(N(t - 1) = a - i) \]

With the help of formula (1) the chance can be expressed in \(a\) and \(b\). In the final step of the simplification Newton’s Binomial is used.

\[
P(T = t) = \sum_{i=1}^{a} \frac{(ab(1-b)^{t-1})^i \cdot (a(1 - (1 - b)^{t-1}))^{a-i}}{i!(a - i)!} e^{-a(1-(1-b)^t)}
\]

\[
= e^{-a(1-(1-b)^t)} \left\{ \sum_{i=0}^{a} \frac{(ab(1-b)^{t-1})^i \cdot (a(1 - (1 - b)^{t-1}))^{a-i}}{i!(a - i)!} - \frac{(a(1 - (1 - b)^{t-1}))^{a}}{a!} \right\}
\]

\[
= \frac{e^{-a(1-(1-b)^t)}}{a!} \left\{ \sum_{i=0}^{a} \frac{a^i (ab(1-b)^{t-1})^i \cdot (a(1 - (1 - b)^{t-1}))^{a-i}}{i!} - \frac{(a(1 - (1 - b)^{t-1}))^{a}}{a!} \right\}
\]

\[
= \frac{e^{-a(1-(1-b)^t)}}{a!} \left\{ (ab(1-b)^{t-1} + a(1 - (1 - b)^{t-1}))^a - (a(1 - (1 - b)^{t-1}))^a \right\}
\]

\[ (8) \]

The expectation of \(T\) is

\[
\mathbb{E}[T] = \sum_{t=1}^{\infty} P(T = t) \cdot t
\]

\[
= \sum_{t=1}^{\infty} t \cdot \frac{e^{-a(1-(1-b)^t)}}{a!} \left\{ (ab(1-b)^{t-1} + a(1 - (1 - b)^{t-1}))^a - (a(1 - (1 - b)^{t-1}))^a \right\}
\]

4.3 Number of tests between detections

Another aspect which can be looked at is \(S\), the number of tests between two consecutive detections of an error. If \(S = s\) then the following must hold:

(i) \(N(k) = i < a\)

(ii) \(N(k - 1) < i\)

(iii) \(N(k + s - 1) - N(k) = 0\)

(iv) \(1 \leq N(k + s) - N(k + s - 1) \leq a - i\)

(i) and (ii) make sure that the last error is found at time \(k\). The third remark states that during \(s - 1\) tests no fault is found. The fourth remark seems odd, \(S\) is the time between the detection of errors. But remember that in one test more errors can be found and the maximum number that can be found is \(a - i\). With the help of (1) the probability that the number of tests between the detection of two consecutive errors equals \(s\) can be expressed in \(a\) and \(b\).

\[
P(S = s) = \sum_{i=1}^{\infty} \sum_{j=1}^{i} \sum_{l=1}^{a-i} P(N(k + s) - N(k + s - 1) = l) \cap N(k - 1) = j \cap N(k) - N(k - 1) = i - j \cap N(k + s - 1) - N(k) = 0)
\]

With the help of the second property of a NHHP it can be stated that

\[
P(S = s) = \sum_{i=1}^{\infty} \sum_{j=1}^{i} \sum_{l=1}^{a-i} P(N(k + s) - N(k + s - 1) = l) \cdot P(N(k - 1) = j) \cdot P(N(k) - N(k - 1) = i - j) \cdot P(N(k + s - 1) - N(k) = 0)
\]

It is not possible to simplify the expression above.
4.4 Interpretation of model

There are only a errors in the program, so (1) is only valid when the number of errors found so far is less than a. With a normal nonhomogeneous Poisson process the number of events that can occur is unlimited. Question is if this assumption that there are only a errors in the program effects the results of this model. There are two ways possible to show that this is indeed the case.

Define $L$ as the time until the first error is found, so $L = \min\{i | N(i) = 0\}$.

$$P(L > t) = P(N(t) = 0) = e^{-C(t)}$$

$$\lim_{t \to \infty} P(L \leq t) = \lim_{t \to \infty} 1 - e^{-C(t)} = 1 - e^{-a}$$

There is a positive probability that not even one error is found even when the program is infinitely times tested. The second way to show that the model has strange consequences is with the help of (5).

The summation of the probabilities does not count up to 1.

In two ways it is possible to show that the assumption that there are only a errors has strange consequences. So maybe the formula given in (1) must be changed, a new counting process $\{N'(t), t = 0, 1, 2, \ldots \}$ is defined.

$$P\{N'(s + t) - N'(s) = n\} = \frac{e^{-[C(s+t) - C(s)]} [C(s+t) - C(s)]^n}{n!} \sum_{i=0}^{a} e^{-a(1-(1-b)^i)}$$

And the probability that $N'(t) equals n$

$$P\{N'(t) = n\} = \frac{e^{-C(t)} C(t)^n}{n!} \sum_{i=0}^{a} e^{-C(t)} C(t)^i$$

There is no closed expression since $\sum_{i=0}^{a} \alpha_i$ is a part from the series for $e^a$. 
5 Aplication of NHHP model on real datasets

There are two datasets available one from [3] and one from [4]. With both datasets we try to find the best estimators for the parameters $a$ and $b$. Extra work is done with the dataset from [3] because there are results in that paper so that a comparison with our result can be made. To estimate parameters $a$ and $b$ Maximum Likelihood Estimation is used. In order to obtain answers two equalities have to be numerically solved. Immediately the question arises if the solutions found with the numerical procedure are stable.

5.1 Dataset 1

The first thing that strikes the eye are the extremely small values for the likelihood function, the function is approximately 0.

With the contourplot an approximation of $(a,b)$ where the maximum of $L$ is found can be made. According to the figure this is something like $(87, 0.00237)$. In [3] the values for $a$ and $b$ where $L$ reaches its maximum are found by numerically solving simultaneously $\frac{\delta \ln L}{\delta a} = 0$, $\frac{\delta \ln L}{\delta b} = 0$, then $(\hat{a}, \hat{b}) = (81.95, 0.00236)$ is found.

Remembering that $C(t) = \mathbb{E}[N(t)]$ the $\chi^2$ test statistic can be calculated for both $a$ and $b$ obtained from the numerical procedure and from the contourplot.

$$\chi^2 = \sum_{i=1}^{9} \frac{(O_i - E_i)^2}{E_i}$$

<table>
<thead>
<tr>
<th>$n_{i-1}$, $n_i$</th>
<th>$O_i$</th>
<th>$E_i(1)^i$</th>
<th>$E_i(2)^i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 [0,50]</td>
<td>8</td>
<td>10.963</td>
<td>11.608</td>
</tr>
<tr>
<td>2 [50,100]</td>
<td>12</td>
<td>9.477</td>
<td>10.059</td>
</tr>
<tr>
<td>3 [100,150]</td>
<td>9</td>
<td>8.212</td>
<td>8.717</td>
</tr>
<tr>
<td>4 [150,200]</td>
<td>2</td>
<td>7.116</td>
<td>7.554</td>
</tr>
<tr>
<td>5 [200,250]</td>
<td>8</td>
<td>6.166</td>
<td>6.546</td>
</tr>
<tr>
<td>6 [250,325]</td>
<td>8</td>
<td>7.742</td>
<td>8.219</td>
</tr>
<tr>
<td>7 [325,400]</td>
<td>6</td>
<td>6.245</td>
<td>6.630</td>
</tr>
<tr>
<td>8 [400,500]</td>
<td>7</td>
<td>6.490</td>
<td>6.891</td>
</tr>
<tr>
<td>9 [500,773]</td>
<td>13</td>
<td>10.614</td>
<td>11.271</td>
</tr>
</tbody>
</table>

With this data $\chi^2_1 = 6.3537$ and $\chi^2_2 = 6.2441$ so according to the $\chi^2$ test statistic the estimation of total number of errors in the software with 87 is better then when it is estimated as 82. It is only a small reduction in the $\chi^2$ test statistic but the total number of faults are raised by no less then 5 errors, this is a significant difference.
5.2 Dataset 2

A contourplot and a normal plot of the likelihood-function (6) are made. The data is obtained from [3] and summarized in table 5.2.

<table>
<thead>
<tr>
<th>Week</th>
<th>Failures</th>
<th>Week</th>
<th>Failures</th>
<th>Week</th>
<th>Failures</th>
<th>Week</th>
<th>Failures</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>8</td>
<td>32</td>
<td>15</td>
<td>7</td>
<td>22</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>9</td>
<td>8</td>
<td>16</td>
<td>0</td>
<td>23</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>38</td>
<td>10</td>
<td>8</td>
<td>17</td>
<td>2</td>
<td>24</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>19</td>
<td>11</td>
<td>11</td>
<td>18</td>
<td>3</td>
<td>25</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>12</td>
<td>14</td>
<td>19</td>
<td>2</td>
<td>26</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>13</td>
<td>7</td>
<td>20</td>
<td>5</td>
<td>27</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>26</td>
<td>14</td>
<td>7</td>
<td>21</td>
<td>2</td>
<td>28</td>
<td>1</td>
</tr>
</tbody>
</table>
6 Intervaldata

When testing it is necessary to know when the testing can be stopped such that, with a certain probability, all errors are found. In previous studies the time between the detection of errors is taken as a criteria. In many cases it is not possible to record all the times when errors accurse. So another stopcriteria is necessary, here the number of errors found during an interval will be recorded and when this number is less than a certain bound the testing will stop.

6.1 Theory

First it is required to make some assumptions and definitions.

The first assumption is that the number of errors detected is always updated after \(n\) tests, so \(N(n), N(2n), \ldots\) are known. Let \(l\) be the bound such that if the number of errors found in an interval is less than \(l\) no more testing is compulsory.

\[
M_j = N((j+1)n) - N(j \cdot n) \quad R = \text{Min}\{M_j | j = 0, \ldots, \left\lfloor \frac{a}{r} \right\rfloor \}
\]

There are only a finite number of \(M_j\)’s because if there are more than \(\left\lfloor \frac{a}{r} \right\rfloor\) it is sure that in one interval less than \(l\) errors found.

Now the proper \(l\) can be calculated.

\[
P(R \geq r) = P(M_0 \geq r) \cdot P(M_2 \geq r) \cdots P(M_l \geq r)
\]
\[
= (1 - P(M_0 < r)) \cdot (1 - (P(M_2 < r)) \cdots (1 - P(M_l < r))
\]
\[
P(M_j < r) = \sum_{i=0}^{r-1} P(N((j+1)n) - N(j \cdot n) = i)
\]
\[
P(R \geq r) = \prod_{j=1}^{l} \left( 1 - \sum_{i=0}^{r-1} P(N((j+1)n) - N(j \cdot n) = i) \right)
\]

With the help that \(P(R \geq r) = 1 - \alpha\) the proper \(l\) can be found. Before this approach can be applied there has to be searched for an expression for \(R\) with the help of expression (1).

\[
P(M_j < r) = e^{a(1-(1-b)^j \cdot n) - a(1-(1-b)^{(j-1)}n)} \sum_{i=0}^{r-1} \frac{[a(1-(1-b)^j \cdot n) - a(1-(1-b)^{(j-1)}n)]^i}{i!}
\]
\[
= e^{a(1-(1-b)^j \cdot n) - a(1-(1-b)^{(j-1)}n)} \sum_{i=0}^{r-1} \frac{[a(1-(1-b)^j \cdot n) - a(1-(1-b)^{(j-1)}n)]^i}{i!}
\]

The expression above is from the form \(b \sum_{i=0}^{n} \frac{a^i}{i!}\), this is a part from the series for \(e^a\) so it is impossible to find a closed expression.

Since there isn’t a closed expression for (10) the results have to be attained numerically. Consecutive for \(r = 2, 3, \ldots\) the probability that \(R \geq r\) is calculated and as soon as \(P(R \geq r) \geq 1 - \alpha\) the proper bound is found.

6.2 Program

Since results can only be numerically attained a program is coded in Mathematica. This code can be found on page 19.

The number \(n\) is a known number and will be a constant in the program. The program works as follows

1. Start with \(r=2\)

2. Calculate the maximum number of intervals, this will be \(\left\lfloor \frac{a}{r} \right\rfloor\)
3. Compute $P(M_i \geq r)$ for $i = 1, \ldots, \lfloor \frac{n}{r} \rfloor$ and store them in a table.

4. Multiply all $P(M_i \geq r)$, this number will be stored as $P$.

5. If $P \geq 1 - \alpha$ stop and return $r$, else $r$ will be raised by 1 and step 2 until 5 are repeated.

6.3 Results

The program GetBound is tested with values $a = 87$, $b = 0.00237$, $n = 50$ and $\alpha = 0.05$. GetBound does not return a bound, the program does work properly so there must be an flaw in the chosen approach.

The proper bound is chosen upon the number of errors found in the intervals $M_0, \ldots, M_{\lfloor \frac{n}{r} \rfloor}$. The probability that in the last interval 0 errors are found is significant is can be seen in figure (3). The probability that not even one error is found is plotted against the intervals 1, \ldots, 87, where interval $i = (50 \cdot i, (i + 1)50)$ in the case of $a = 87$, $b = 0.00237$, $n = 50$ and $\alpha = 0.05$.

Figure 3:

As can ben seen the probability that at most one error is found stijgt very steep. So expression (10) won’t be greater than $1 - \alpha$ when there are too many intervals.

The number of intervals equals $\lfloor \frac{n}{r} \rfloor$, when $r$ is big the number of intervals will be small. But when $r$ is big $\left(1 - \sum_{i=0}^{r-1} P(N((j + 1)\cdot n) - N(j \cdot n) = i)\right)$ will be small which means that (10) will again be small.

Zit in de tang because if $r$ is too big the term $\left(1 - \sum_{i=0}^{r-1} P(N((j + 1)\cdot n) - N(j \cdot n) = i)\right)$ will be small and if $r$ is too small there are too many intervals.
7 Conclusions
A Mathematica code

A.1 Plot of MLE

The formulas necessary to calculate the MLE-estimators for a and b. NList is the list with the
n_t is number of tests sofar, X is the list of x_i, is number of observations during n_t tests. The
function C is the meanvalue-function, the two other functions Likelihood and LogLikelihood
speak for themselves.

\[ C[a_\_\_, b_\_\_, n_\_] := a(1 - (1 - b)^n) \]

Likelihood[a\_\_, b\_\_, X\_\_, NList\_] :=
\[ \text{Exp}[-C[a, b, \text{Last}[NList]]] \prod_{i=2}^{\text{Length}[X]} \frac{C[a, b, NList][i] - C[a, b, NList][i - 1]}{(X[i] - X[i - 1])} \]

LogLikelihood[a\_\_, b\_\_, X\_\_, NList\_] :=
\[ \sum_{i=2}^{\text{Length}[X]} (X[i] - X[i - 1]) \text{Log}[a] + \sum_{i=2}^{\text{Length}[X]} (X[i] - X[i - 1]) \text{Log}[(1 - b)^{NList[i - 1]} - (1 - b)^{NList[i]}] - a(1 - (1 - b)^{NList[[Length[NList]]]}) - \sum_{i=2}^{\text{Length}[X]} \text{Log}[(X[i] - X[i - 1])!] \]

To compute the estimators for a and b two equalities have to be solved, \( \frac{\partial L}{\partial a} = \frac{\partial L}{\partial b} = 0 \). The
expression for these expressions are called DLogLikelya and DLogLikelyb.

\[ \text{DLogLikelya[a\_\_, b\_\_, X\_\_, NList\_\_] := 1 - (1 - b)^{\text{Last}[NList]} - \frac{\text{Last}[X]}{a} \]

\[ \text{DLogLikelyb[a\_\_, b\_\_, X\_\_, NList\_\_] :=} \sum_{i=2}^{\text{Length}[X]} \left( \frac{(X[i] - X[i - 1])(1 - b)^{NList[i]} - 1 - NList[i](1 - b)^{NList[i - 1]} - (1 - b)^{NList[i - 1]}}{(1 - b)^{NList[i] - (1 - b)^{NList[i]}}(1 - b)^{NList[i - 1]}} \right) \]

\[ \text{a Last}[NList](1 - b)^{\text{Last}[NList]} - 1 \]

The method is used for both datasets but here the method will only be shown for the data
attained from [3].

NList1 = \{0, 50, 100, 150, 200, 250, 325, 400, 500, 773\};

X1 = \{0, 8, 20, 29, 31, 39, 47, 53, 60, 73\};

When we try to find the numerical estimators Mathemtica only returns errors.

FindRoot[\{DLogLikelya[a, b, X1, NList1], DLogLikelyb[a, b, X1, NList1]\}, \{a, 1\}, \{b, 0\}]}

Another way to estimate the maximum is with the help of graphics. The likelihood function is
plotted for 1 ≤ a ≤ 100 and 0 ≤ b ≤ 1. The result will be saved.

\[ q = \text{Plot3D}[\text{Likelihood[a, b, X1, NList1]}, \{a, 1, 100\}, \{b, 0, 1\}, \text{AxesLabel} \rightarrow \{"a", "b", "Likelihood"}, \text{ColorFunction} \rightarrow \text{Automatic}] \]

Export"D : /Mydocuments/Bachelor/verslag/likelihoodplot2.eps", q, "EPS"; Clear[q];
It is clear that the estimator for $b$ can easily be made but in the $a$-direction the function is not steep so it is hard to estimate $a$, this is shown in the next picture which is a rotated view of the last plot.

\[
\text{Plot3D[Likelihood[a, b, X1, NList1] * 10^{10}, \{a, 80, 95\}, \{b, 0, 0.005\}, ViewPoint → \{0, -3, 1\}];}
\]

There is an small area where the maximum of the likelihood lies. With a contourplot the area where the maximum lies can even be made smaller.

\[
q = \text{ContourPlot[Likelihood[a, b, X1, NList1] * 10^{10}, \{a, 84, 90\}, \{b, 0.0023, 0.0026\}, Contours → 40, PlotPoints → 200, AxesLabel → \{"a", "b"\}; Export["D:/Mydocuments/Bachelor/verslag/contourplot2 eps", q, "EPS"]}; \]

Clear[q];

A.2 Intervaldate

Suppose that after every $n$ tests the number of discoverd errors is checked. The number of faults are noted at the intervals $(x \cdot n; (x + 1)n)$. The probability to detect $y$ errors in one specific interval looks like

\[
P[N[(x+1)n] - N[xn] = y] = \frac{[a(1-(1-b)^{x+1}) - a(1-(1-b)^x)]^y \exp\left[-\left(a(1 - (1 - b)^{(x+1)n}) - a(1 - (1 - b)^{xn})\right)\right]}{y!}
\]

this expression can be simplified

\[
P[N[(x+1)n] - N[xn] = y] = \frac{a(1-b)^x (1 - (1-b)^{n})^y \exp[-a (1 - b)^x (1 - (1 - b)^n)]}{y!}
\]

Define the probability to detect $y$ errors in the interval $(x \cdot n; (x + 1)n)$ as

\[
P[a\_\_ , b\_\_ , n\_\_ , x\_\_ , y\_\_] := \frac{[a(1-b)^x (1 - (1-b)^{n})^y \exp[-a (1 - b)^x (1 - (1 - b)^n)]}{y!}
\]

The program code is

\[
\text{GetBound[a\_, b\_, n\_, \alpha\_] := Module[\{W, B, r, s, grens\},
\text{grens} = False;
\text{r} = 2;
\text{While}[\text{grens} \&\& r < a,}
\text{If}\left[\text{\frac{s}{r}} \in \text{Integers}, s = \frac{s}{r} - 1, s = \text{Floor}[\frac{s}{r}]\right];
W = \text{Table}[1 - \sum_{i=0}^{r-1} P[a, b, n, j, i], \{j, 0, s\}];
B = \prod_{i=1}^{\text{Length}[W]} W[[i]];\text{If}[B < 1 - \alpha, r = r + 1, \text{grens} = True; \text{Print["r = ", r]}]
\]

19
## B Table of Notation

<table>
<thead>
<tr>
<th>variable</th>
<th>definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>initial number of errors in program</td>
</tr>
<tr>
<td>$X_i$</td>
<td>time between $(i-1)^{th}$ and $i^{th}$ error</td>
</tr>
<tr>
<td>$a_i$</td>
<td>initial number of errors from class $i$</td>
</tr>
<tr>
<td>$\theta_i$</td>
<td>detection probability from class $i$</td>
</tr>
<tr>
<td>$T_{ij}$</td>
<td>number of tests before $j^{th}$ error from class $i$ is detected</td>
</tr>
<tr>
<td>$p_{ij}$</td>
<td>success parameter of $T_{ij} = 1 - (1 - \theta_i)^{a_i - j + 1}$</td>
</tr>
<tr>
<td>$S_n$</td>
<td>$S_n = \sum_{i=1}^{n} X_i$</td>
</tr>
<tr>
<td>$b$</td>
<td>error detection rate per error</td>
</tr>
<tr>
<td>$T$</td>
<td>total number of tests necessary to detect all errors</td>
</tr>
<tr>
<td>$S$</td>
<td>number of tests between two consecutive detections of errors</td>
</tr>
<tr>
<td>$n$</td>
<td>after each $n$ tests the number of errors is detected</td>
</tr>
<tr>
<td>$M_j$</td>
<td>$N((j+1)n) - N(jn)$</td>
</tr>
<tr>
<td>$R$</td>
<td>the minimum of all $M_j$'s</td>
</tr>
</tbody>
</table>
References


