Individual Project Courses in the Research School of Computer Science

Web Service for toxicant trigger valuation

Shichao Dong(u5135470)
Supervisor: Dr. Warren Jin

10/19/13

Australian National University
ANU College of Engineering and Computer Science
School of Computer Science
Contents

Abstract 1

1 Introduction 3
  1.1 Overview .................................................. 3
  1.2 Contributions ............................................ 4
  1.3 Structure .................................................. 4

2 Background 5
  2.1 Statistic models .......................................... 5
    2.1.1 Burr type III Distribution ............................ 5
    2.1.2 Log-logistic Distribution ............................ 6
    2.1.3 Weibull Distribution ................................ 7
    2.1.4 A Bayesian model for the NEC ....................... 7
    2.1.5 Markov Chain Monte Carlo (MCMC) ................... 9
  2.2 Software .................................................. 10
    2.2.1 BurrliOZ ............................................. 10
    2.2.2 R and shiny .......................................... 11
    2.2.3 BUGS, WinBUGS and OpenBUGS ........................ 13

3 Experiment 15
  3.1 Overview ................................................. 15
  3.2 BurrliOZ .................................................. 15
    3.2.1 Loading data ........................................ 15
    3.2.2 Check Graphics options .............................. 17
    3.2.3 Set fit options ..................................... 17
    3.2.4 Check and update Graph ............................. 17
    3.2.5 Export data ......................................... 17
    3.2.6 Perform calculation ................................ 17
  3.3 NEC Model ................................................ 18
    3.3.1 WinBUGS ............................................. 18
    3.3.2 OpenBUGS ............................................ 19

4 Web services 21
  4.1 Overview ................................................. 21
  4.2 BurrliOZ .................................................. 21
    4.2.1 Interface ............................................ 22
Abstract

In this project, we are focusing on building two web services based on revisiting and experimenting the previous work- Shao[15]'s and Fox[9]'s. Generally speaking, our work has been separated into three parts, the first is revisiting and experimenting the previous work. In this part, we just followed and repeated the previous work by using BurrliOZ MS Windows version, WinBUGS as well as OpenBUGS. The second part is designing the UI and implementing their theory as well as packages. In this part, we did the UI design by just using R code rather than widget or control providing by any certain OS, which makes it is accessible by any computer with any OS even on mobile devices like iPad. The last part of our project is modifying the original settings and solving the problem that we found in the first stage. In our project, we provide scientists or research an opportunity to experimenting and testing their data by using Shao’s and Fox’s model via web browser. Furthermore, we also extended their original work by providing more options like priors distributions or models. In the future work, we are going to compare this two methods or even combine them in order to draw some scientific conclusions.
1 Introduction

1.1 Overview

The toxicant concentration level is an important index in the field of eco-toxicology since it could be determined to protect biological species. In order to determine that index, Hazardous concentration ($HC_x$) comes to be a significant research topic. In this thesis, we mainly focus on two work with different aspect about this topic, the Shao [15] and Fox [9].

For the Shao’s [15] research, he used the laboratory toxicity data to estimate the hazardous concentrations and their confidence limits for toxic substances. He claims that the the lower tail of the NOEC(no observed effect concentrations) distribution is the main issue that is cared about. However, it is impossible to use more robust distribution-free approaches due to small sample sizes. To overcome that problem, Shao [15] considers previous work such as Aldenberg et al [1] and Burr [5], and claims to use statistical extrapolation techniques, for example he treats the underlying distribution of NOECs as a particular toxicant.

For Fox [9]’s research, he tried to estimate the no effect concentration ($NEC$) and the hazardous concentration ($HC_x$) under a Bayesian framework and treated this method as an alternative ways compared with the conventional methods based on NOECs. It can be seen that the major task of his research is to find a realistic threshold. As he argues, the combination of dose–response data and prior information generates posterior distributions for the parameters is the main advantage of his method.

Some tools have been developed to support the above work. For example, BurriLOZ is a implement software of Shao’s work, it provides the ability to fit several standard distributions and estimate the concentrations of chemicals. Fox also provided some WinBUGS code based on his paper. However, BurriLOZ is written in Microsoft Visual basic 2010 which uses the .NET framework, which makes it impossible for users who may not have Microsoft Operation system especially Linux user who widely exist in science filed and mobile device users whose number is increasing these years. Another problem is that using BurriLOZ requires R environment, which may increase the learning cost for ordinary users. The Fox’s code is written by WinBUGS, the same problem with BurriLOZ is the requirement of windows. Besides, running this code may need some programming background that the learning cost is even higher than BurriLOZ. Furthermore, WinBUGS has not updated since 2008, which means the bugs can not be fixed.
In this report, firstly, we revisited and experimented these two work and tried to evaluate differences between our experiment and the result come from papers. Secondly we developed two web services for each work respectively. Thirdly, we tried to modify some key parameters such as prior distributions and iteration times to compare with the paper’s work in order to draw some scientific conclusions.

1.2 Contributions

The contributions by this project include:
1. Revisited and experimented Shao’s and Fox’s work, found that Fox’s result is not repeatable with following his steps;
2. Unpack BurrliOZ’s R package, reorganized the API and interface;
3. Implement WinBUGS code into the R environment by using OpenBUGS;
4. Using R package called shiny building the web services;
5. Designing new UI for the applications;
6. Providing normal distribution as prior distribution to compare with Fox’s gamma distribution;
7. Testing the web applications on both Windows and Linux;
8. Testing the web applications with different data sets.

1.3 Structure

The rest report is organized as followed:

- Chapter 2 illustrates some background of this project;
- Chapter 3 describes the experiment that we did follow the paper and compares the result;
- Chapter 4 describes our web services including UI design and code structure;
- Chapter 5 shows modification based on the previous work and the result;
- Chapter 6 discusses some possible future work aspect gives the conclusion.
2 Background

2.1 Statistic models

2.1.1 Burr type III Distribution

Burr [5] in his paper claimed that the cumulative function could be directly used since it has its own advantage in finding the most proper function of an appropriate type and integration. He also gave a number of cumulative functions with examples in fitting data. According to Tadikamalla [16], the main purples of the distribution form selection is making it accessible for the mathematical analysis and attaining a reasonable approximation, and then Burr paid special attention to one of these forms, the distribution function $F(x)$ is given as

$$F(x;\theta) = 1 - (1 + x^c)^{-k}, \quad x > 0, \quad c > 0, \quad k > 0$$ (2.1)

Both $c$ and $k$ are shape parameters. Burr (1942) suggests that scale and location parameters could be considered in cumulative distribution function (CDF) for greater flexibility. In the year of 1996, Lindsay et al. [11] claim that the Burr type III known as distribution of the reciprocal of the type XII variable has a wider region of applicability than both type XII and the Weibull distribution, and then they gave Burr Type III CDF form. Nowadays the form is written as (Göve et al.[10]) :

$$F(x;\theta) = [1 + \left(\frac{b}{x-a}\right)^c]^k, \quad c > 0, \quad k > 0$$ (2.2)

The $a$ is location parameter and $b$ is scale parameter and $c$ and $k$ are shape parameters so that $\theta = (a, b, c, k)$ ; for $x > a$, when $F(x) = 0$ for $x \leq a$. For $x > a$, the corresponding probability density function (PDF) is :

$$F(x;\theta) = \frac{kc}{b} \left(\frac{b}{x-a}\right)^{c+1} (1 + \left(\frac{b}{x-a}\right)^c)^{-(k+1)}$$ (2.3)

In addition, Shao [15] point out the The $q$th quantile is:

$$x_q = b/\left[\left(\frac{1}{q}\right)^{1/k} - 1\right]^{1/c}$$ (2.4)
and the $q^{th}$ percentile is the ($q$ out of 100)th quartile.

He [15] also states the density function of Burr type III distribution (Three Parameter) as following:

$$f_B(x) = \frac{kc}{b} \cdot \frac{(\frac{x}{b})^{c-1}}{[1 + (\frac{x}{b})^{c+1}]^{k+1}}, \quad b, c, k > 0 \text{ for } 0 < x < \infty$$ (2.5)

Lindsay et al. [11] claim that one of the advantage of Burr type III distribution is that it cover a wider range of skewness-kurtosis values than Weibull distribution. Gove et al.[10] also agree with that and they add that Burr type III is flexible and easy to use. The reason that Shao used it is “its flexibility and ease-of-use” and he says that “our technique is easily extended to mixtures of the three-parameter Burr type III distributions, which can be used to model multi-model distributions”[15].

### 2.1.2 Log-logistic Distribution

The Log-logistic distribution has been studies the detail by Shah & Dave [14] as well as Tadikamalla & Johnson [17]. According to Tadikamalla & Johnson [17], the log-logistic distribution is a special case of the Burr type XII distribution and they gave the uni-modal density function as:

$$f_Y(y) = \delta e^{-\gamma y^{\delta-1}}(1 + e^{-\gamma y^{\delta}})^{-2}$$ (2.6)

Shao [15] also agrees with this point and gave the log-logistic distribution function as:

$$F_{LL}(x) = \frac{1}{1 + \exp\{[-\ln(x) - \mu]/\theta}\}, \quad \theta > 0, \text{ for } x > 0$$ (2.7)

He rewrote the log-logistic distribution function as followed:

$$F_{LL}(x) = \frac{1}{1 + (\frac{x}{\theta})^{1/\theta}}$$ (2.8)

As can be seen function (2.8) is the same form with function (2.2), which support that log-logistic distribution is a special case of the Burr type III distribution.

The reason that Shao [15] included log-logistic model is “it is guaranteed that the distribution fit using the three parameter Burr type III distribution to raw data, is at least as good as the log-logistic distribution.”
2.1 Statistic models

2.1.3 Weibull Distribution

As Culyer [8] says Weibull model is used for duration analysis, and this distribution is named after Waloddi Weibull since he described it in detail in 1951. As Weibull [20] description, the CDF is

\[
\begin{cases}
1 - e^{-(x/\lambda)^k} & x \geq 0 \\
0 & x \leq 0
\end{cases}
\]  

(2.9)

and PDF is:

\[f(x) = \begin{cases}
k(\frac{x}{\lambda})^{k-1} - e^{-(x/\lambda)^k} & x \geq 0 \\
0 & x \leq 0
\end{cases}\]  

(2.10)

Weibull [20] states that $\lambda$ and $k$ are two parameters and both of them are positive.

Shao [15] explains why he includes Weibull distribution. Before that, he gave the approximation as:

\[1 + \left(\frac{b}{x}\right)^c \approx 1 + \left(\frac{b^d}{x}\right)^{c/\delta} \]  

(2.11)

Under some condition, this may lead to high correlation which result in three numerical issues. “Firstly, when the underlying distribution is reciprocal Weibull and the three-parameter Burr type III distribution is used, the maximum likelihood estimation procedure will fail to converge because of an extra parameter. Secondly, when the true three parameter Burr type III distribution has very small $b$ and very large $k$, the surface of the likelihood function can be very flat. Finally, the limiting distributions may also imply that the distribution parameters cannot be identified because the data available for fitting distribution is deficient and therefore would not allow separate parameter estimation.” [15]

2.1.4 A Bayesian model for the NEC

In Fox’s paper [9], he mainly concentrated on one model from Pires et al. [12]. According to Pires et al. [12], to estimate a NEC, considering the data type, they included the response variable, $y$, the explanatory variable, $x$ by a function that has two steps at the point $c$, then they gave the simplest model with a linear decay:

\[y_j = l - m(x_j - c)I(x_j - c) + \varepsilon_j\]  

(2.12)
In this function, $\varepsilon_j$ is independent random errors with zero mean, $(x_j, y_j), j = 1, \ldots, n,$ are the pair of observations; $l, m$ and $c$ are parameters, $l$ is the constant level near $x = 0$, and $m$ is the slop of the decay, $I(x)$ is an indicator function that is given as:

$$I(x) = \begin{cases} 
1, & x \geq 0 \\
0, & x \leq 0 
\end{cases}$$

(2.13)

The mean value of this $E(y|x) = l - m(x - c)I(x - c)$, according to the model (2.12), we can have the plot (Fig. 2.1) to illustrate the model.

![Figure 2.1: Mean response function with a linear decay][9]

The weakness of this model is it seems to be unreasonable in some situations, therefore, Pires et al.[12] propose another model as following:

$$y_j = l \cdot \exp[-m \cdot (x_j - c)I(x_j - c)] \times \varepsilon_j$$

(2.14)

The difference is that $m$ is rate of decay and $\varepsilon_j$ is random errors approximately following a log-normal distribution. The plot (Fig. 2.2) below represents the new model.

Fox [9] tries to use the new model and gave the full model definition with arbitrary probability function $g_y(*)$:

$$y_i \sim g_y(*)$$

(2.15)
2.1 Statistic models

\[ E(Y_i|x_i) = \mu = \alpha \exp[-\beta(x_i - \gamma)I(x_i - \gamma)] \]  

(2.16)

With

\[ I(z) = \begin{cases} 
1, & z \geq 0 \\
0, & z \leq 0 
\end{cases} \]

2.1.5 Markov Chain Monte Carlo (MCMC)

Andrieu et al [2] state that MCMC could be widely used in machine learning, physics, statistics, econometrics and decision analysis to deal with the integration and optimization problems within large dimensional spaces and MCMC consists of two fundamental problems, including :

- Bayesian inference and learning
- Statistical mechanics

The basic steps of MCMC methodology could be summarize as :

1. Construct a Markov chain
2. Generate samples
3. Monte Carlo integration

The major step is the sampling algorithm. In this project, we used Gibbs Sampling.
2.2 Software

2.2.1 BurrliOZ

According to the CSIRO (Commonwealth Scientific and Industrial Research Organization) website[^7], BurrliOZ is a statistical software package to help environmental managers to figure out trigger values of toxicant in accordance with the Australian and New Zealand Guidelines for Fresh and Marine Water Quality. In our report, we used the 2.0 version of BurrliOZ, which is a free standing software application that is underpinned by statistical processing in the R statistical software (http://www.r-project.org/). As its updates manual[^6] says there are several standard distributions being fitted by BurrliOZ 2.0, and it also estimate the concentrations of chemicals such that a given percentage of species will not be unacceptably affected with a known probability. The concentration of this software is upon the protecting concentration referred to as this from herein that is the threshold concentration, and for the Australian and New Zealand Water Quality Guidelines, the threshold concentration is termed a trigger value (TV).

The interface part is written in Microsoft Visual basic 2010 which uses the .NET framework, and using the R statistical package to implement the core statistical calculations and plotting. To use BurrliOZ, user need both support of Microsoft operation system as well as R that could be downloaded from the internet (see http://www.r-project.org/) installed on the machine. BurrliOZ package is written by R but it can be treated as independent entities. Note that before execute BurrliOZ, it would look to find the location of R. Therefore, R should be install before the BurrliOZ’s first running.

The Fig. 2.3 is the main interface of BurrliOZ, which contains the main functions. At the top of this UI, some tabs could be found and each of the tabs represents different functions. The “input” and “fit options” are used to input data and set some key values or options such as choosing the distribution that can be used in the calculation. In the “Graph” tab, the BurrliOZ would display the result by plot and user could also export the plot as they preferred format like JPEG, PDF or EPS. In addition, “Graph” also provides the report that contains the whole result of the estimation. In the data tab, it would print the data set on the panel that is currently being used in the analysis. This could be used primarily as a tool to check if the data is correct. In the calculator tab, it would let the user calculate trigger values and the percentages of species protection levels and associated confidence intervals. In the log tab, the program would record the setting, processing and errors during the calculation, and then displays it in the panel.

For the statics calculation part, the BurrliOZ is using three distributions to implement, the Burr type III, log-logistic and the Weibull. The estimation of distributions is based on maximum likelihood. They use a percentile bootstrap to calculate the 95% CI’s and the original data is re-sampled and the model is refit using the identical algorithm. Some re-samples yield divergent estimates. All the data and values

[^7]: CSIRO website
[^6]: BurrliOZ updates manual
2.2 Software

Figure 2.3: Main UI

are plotted and reported after the calculations and exported via BurrliOZ report. The estimated trigger values in the calculation and report is estimated directly from the fitted line.[6]

2.2.2 R and shiny

According to the R’s official website[13], “R is a language and environment for statistical computing and graphics. And R is a GNU project that is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S. R provides a wide variety of statistical and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity. One of R’s strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formula where needed. Great care has been taken over the defaults for the minor design choices in graphics, but the user retains full control. R is available as Free Software under the terms of the Free Software Foundation’s GNU General Public License in source code form. It compiles and runs on a wide variety of UNIX
platforms and similar systems (including Free BSD and Linux), Windows and OS X.”

As can be seen, R is used to do the statistical computing and graphics. The most important part of R is packages. With the help of packages, R could implement different functions. “shiny” is one of the packages.

Shiny was developed by the Rstudio team and treated as a child project. Shiny is a R packages for R users who is not familiar with Web design such as HTML and JavaScript. Shiny has defined some controls like sliders, drop-downs, and text fields and help user to build a interactive web application by only using R. There are many examples on its official website: http://www.rstudio.com/shiny/.

Shiny has two versions, and the basic one that is for ordinary users is designed to run Shiny applications locally. If user wants to share their work, they have to send the whole project include their code, local R packages and the whole shiny structure. Besides this, shiny has a sever version that could enable developers to build their project on a Linux server which contains R and shiny server. For users who have no severs their own, the shiny team provides test sever for developers.

Figure 2.4: The shiny example

The Fig. 2.4 shows a example of a web application developed by shiny. As can be seen, the left side is the application’s interface in the web browser. The right side is the code. A normal shiny application structure consists of two R files, one is ui.R which is used to build the interface and the other is server.R which contains the core R code for computing values and running models. In some situations, the applications may have a globe.R file which is used to restore some globe various or functions.
2.2 Software

2.2.3 BUGS, WinBUGS and OpenBUGS

According to the official website, The BUGS is short for Bayesian inference Using Gibbs Sampling. The whole BUGS aims to using Markov chain Monte Carlo (MCMC) methods do the Bayesian analysis of complex statistical models. The project began in 1989 in the MRC Biostatistics Unit, Cambridge. There are two main versions of BUGS, namely WinBUGS and OpenBUGS. In the beginning, the project led initially to the Classic BUGS program, and then they transfer to the WinBUGS software developed jointly with the Imperial College School of Medicine at St Mary’s, London. For now, the team is focusing on developing the OpenBUGS project. [4]

Generally speaking, WinBUGS is a windows interface for BUGS for analysis. The advantage of this is that it enable users who are not familiar with coding or programming could construct the model by using a graphical interface call DooleBUGS. The latest version of WinBUGS is 1.4.3, and it has no longer required users to register. The disadvantage of WinBUGS is very clear that the development team is not interested in developing and pushing the project. This project or package is only for the windows users although it provide a R package called R2WinBUGS to call it from command line but Unix-like user such as Linux and OS X can not use it unless they install some software like wine to let them run windows applications on Unix-like operation system.

OpenBUGS is very similar to WinBUGS running on the most of the operating system. According to the official wiki, [19] “the specified model belongs to a class known as Directed Acyclic Graphs (DAGs). One of the main differences between OpenBUGS and WinBUGS is the way in which the expert system makes its decisions. WinBUGS defines one algorithm for each possible computation type whereas there is no limit to the number of algorithms that OpenBUGS can make use of, making for much greater flexibility and extensible.” The Fig. 2.5 displays two versions of BUGS, and it can be seen the UI is very similar. We would cover more difference of this two in the next chapter during the experiment.

![Image](image.png)

**Figure 2.5:** The interface of WinBUGS and OpenBUGS
In this chapter, we introduced two foundation parts of our project, models and software. For the models, we mainly focused on the testing and implement. For the software part, we introduced three major types of software tools that help us in the implement part. In the next chapter, we would do the test and experiment based on the models and we would cover more usage.
3 Experiment

3.1 Overview

In this chapter, we will talk about the experiment that we did followed the guild or the paper. The reason that we did this is we have to do the verification for the previous work as well as understanding the tech that we are going to use in our web service. The environment that we used is:

- CPU: Intel(R) Core(TM)2 Duo CPU E7500 @2.93GHz 2.93GHz
- RAM: 2.00GB
- OS: Windows 7(32Bit)

3.2 BurrliOZ

The basic usage of BurrliOZ consist of 6 steps, including:

1. loading data
2. check graphics options
3. set fit options
4. check and update graph
5. export data
6. perform calculation

3.2.1 Loading data

Tab.3.1 illustrates the data that we used in the experiment. As can be seen, the data consists of four columns, the first is the concentration data that is assumed by BurrliOZ, and then the details about the data that can be used in plotting stays rest of three columns.

As shown in Fig.3.1, we use the browse button to locate the file above, then choose the “Load Data” button that could load the data. Before really using the data in the workspace, we have to finish dialog box of requesting information. We would
<table>
<thead>
<tr>
<th>Data</th>
<th>Taxonomic group</th>
<th>Type</th>
<th>Species</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Alga</td>
<td>converted acute</td>
<td>Chlorella sp.</td>
</tr>
<tr>
<td>15</td>
<td>Gastropod</td>
<td>chronic</td>
<td>Amerianna cumingi</td>
</tr>
<tr>
<td>32</td>
<td>Macrophyte</td>
<td>chronic</td>
<td>Lemna aequinoctialis</td>
</tr>
<tr>
<td>32</td>
<td>Cnidarian</td>
<td>converted acute</td>
<td>Hydra viridissima</td>
</tr>
<tr>
<td>642</td>
<td>Fish</td>
<td>chronic</td>
<td>Mogurnda mogurnda</td>
</tr>
<tr>
<td>778</td>
<td>Fish</td>
<td>converted acute</td>
<td>Melanotaenia splendidia inornata</td>
</tr>
<tr>
<td>187</td>
<td>Macrophyte</td>
<td>chronic</td>
<td>Ceratophyllum demersum</td>
</tr>
<tr>
<td>12</td>
<td>Cladoceran</td>
<td>chronic</td>
<td>Moinodaphnia macleayi</td>
</tr>
</tbody>
</table>

Table 3.1: Test Data

complete by typing in the toxicant, specifying the units and typing a short note to remind us why we are doing the analysis. This part is optional for toxicant and note input. After this, the data is officially loaded.

In order to confirm the data loading process or if the data is the correct one that we need. We can check it in the data tab. In addition, Assume that we want to use different symbols for the acute/chronic tests and that we want to label the points to denote the Taxonomic groups represented. First we need to set the plotting symbols for the acute and chronic data.
3.2.2 Check Graphics options

The Option is in the main window either. As shown in Fig. 3.1, we can set the graph set by clicking the Options on the left top button on the main window. The graph options are options that a user can set such that the changes are saved when the program exits. In our experiment, we just chose the default setting.

3.2.3 Set fit options

After loading data and doing the presetting, the next step is to set the fit options. The options are provided in the second tab called fit options. The Fit section is disabled (Grey) to choose from us. According to the instruction [6], the choice of model is automatically selected depending on the number of observations, consistent with the TV derivation method ($n < 8$ – log logistic; $n \geq 8$ - Burr Type III). We could use different symbols for the the experiment and we could specify the Species that we want to label the points. In our experiment, we select Plot labels and highlight the “Taxonomic group” column and we select Plot symbols and highlight the “Type” column.

3.2.4 Check and update Graph

After setting the fit options, we pressed the “Run” button, and it will help to generate the result. Basically, it is shown as a graph in the third tab. After the process of running, the graph will be displayed in the graph tab like the XXXXTODO. In addition, we can also modify the options in the fit option tab and update the the new graph. In our experiment, we chose the 95% species protection level on the graph and modified a better x axis title. Also, we may use “change graphics options and rerun” button to color the plot.

3.2.5 Export data

In the same tab, we could use the buttons in the right side to get the data, the plot or the report about the process result. The result is based on the options that we have chosen. The report that we generated is shown in XXXXXXXXXXXTODO.

3.2.6 Perform calculation

In the last tab, the calculator tab, we could do some calculations. For example, if we want to calculate 90% protection of the trigger value, the BurriOZ allows us to input a concentration and calculate its level of protection. XXXXXXXXTODO is the result.
Chapter 3 Experiment

3.3 NEC Model

In the paper of Fox [9], he made examples based on the original model, and release the WinBUGS code. Therefore, for this part, we intended to follow Fox’s step and tried to copy the result. However, we found some problems during the experiment. Then, we decided to do the experiment on OpenBUGS to compare with the original one.

3.3.1 WinBUGS

The data we will use to test is shown in Tab. 3.2. The data is token from Biesinger et al. [3].

<table>
<thead>
<tr>
<th>Toxicant</th>
<th>Hg concentration ($\mu g/L$)</th>
<th>$n_i$</th>
<th>$y_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mercuric chloride</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>180</td>
<td>171</td>
</tr>
<tr>
<td></td>
<td>0.43</td>
<td>80</td>
<td>65</td>
</tr>
<tr>
<td></td>
<td>0.91</td>
<td>80</td>
<td>73</td>
</tr>
<tr>
<td></td>
<td>1.82</td>
<td>180</td>
<td>160</td>
</tr>
<tr>
<td></td>
<td>3.53</td>
<td>180</td>
<td>108</td>
</tr>
<tr>
<td></td>
<td>5.31</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>Methyl mercuric chloride</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>80</td>
<td>77</td>
</tr>
<tr>
<td></td>
<td>0.17</td>
<td>80</td>
<td>79</td>
</tr>
<tr>
<td></td>
<td>0.28</td>
<td>80</td>
<td>76</td>
</tr>
<tr>
<td></td>
<td>0.52</td>
<td>80</td>
<td>76</td>
</tr>
<tr>
<td></td>
<td>0.87</td>
<td>80</td>
<td>71</td>
</tr>
<tr>
<td></td>
<td>1.14</td>
<td>80</td>
<td>0</td>
</tr>
<tr>
<td>Phenyl mercuric acetate</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>35</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>0.35</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>0.54</td>
<td>35</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>1.12</td>
<td>35</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>1.90</td>
<td>35</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>3.00</td>
<td>35</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3.2: Daphnia magna data

For the model (2.16) that we mentioned in the previous part, Fox [9] made the 2.16 becoming:

\[ E[P_i] = \theta_i = \alpha \exp[-\beta(x_i - \gamma)I(x_i - \gamma)] \]  \hspace{1cm} (3.1)

Although the choice of prior distributions is flexibility, we still followed Fox’s step to choose gamma distribution (we will cover more distributions and modifications in the
following chapters). In the code, we use $\alpha \sim \text{dgamma}(0.0001, 0.0001)$ to make $\alpha$ fitting the gamma distribution. The parameters here is to keep the prior distribution non-informative. In WinBUGS, the function of dgamma is

The model here needs to run 100000 times. Before this, we did a initial 'burn-in' period with 10000 iteration. The code provided by Fox is used to run the second part of the data that is Methyl mercuric chloride. Fox still provided all results from this data. So we did all to compare with his result. However, the process of Fox’s experiment can not be followed since the WinBUGS would warn that gamma cannot be sliced. After research about that, we found the problem might be the parameters of the priors seems to be too small to slice. In this case, we tried to set the parameters as small as it could be to see the result. Fig. 5.2 shows the result compared with Fox’s.

As can be seen, the result is slightly different for the standard deviation. We thought the reason might be the parameters are not small enough, so we considered to test our data on the OpenBUGS.

### 3.3.2 OpenBUGS

In order to confirm that the priors is the same in both OpenBUGS and WinBUGS, we checked the definition of dgamma function in OpenBUGS is:

![Figure 3.2: OpenBUGS Result](image)

As can be seen, the function is the same one with WinBUGS. Then we followed the same step that we did in the WinBUGS, and Fig. 3.2 is our result. Compared with the result from Fox[9], there are still some differences between them.

To conclude, in this chapter, we followed the steps from previous work. For the BurrliOZ, it works reliable and stable. Therefore, we could build web services based on it. For the NEC model, we still cannot get a same result following the steps of Fox. But the result seems to be very close. We spend some time on solving problems that we met. We would give some ideas and conclusions in the discussion chapter.
4 Web services

4.1 Overview

After the experiment part, we will build two web services to provide end users to do the calculation in their web browser. The first web service is build for the BurrliOZ and the second one is for NEC. To build the BurrliOZ web service, the development is based on a R package called fromBurrliOz2 from CSIRO. The NEC service is built based on the the WinBUGS code with OpenBUGS environment. The tool we used to build web application is shiny that is a R package (see sec.2.2.2).

4.2 BurrliOZ

The web service is based on a R packages called fromBurrliOz2. In fact, this package is used to provide R users to do the calculation without interface. The advantage of this package is R user may easily to call the function in the package to finish their work regardless their operating system. However, for the people who is not familiar with R language or even coding things, using this seems even more difficult compared with the windows version. This is the motivation to build a web service. As can be seen in Fig.4.1, it provides some functions in the package. This package is the old version and the windows version that we talked about is the latest 2.0 version.

![Content of fromBurrliOz2](image)

**Figure 4.1:** Content of fromBurrliOz2

The functions from the package cover main features of the windows application. Generally, users could set plot options like xlab and ylab, additionally, users could
choose to fit different distributions ether. The packages provide burr and log-logistic and Weibull which is removed by the new version. For the output part, the package provides drawing part and the report part.

However, if the application is built based on the packages, the structure should be modified to fit the shiny package. As we mentioned in the sec. 2.2.2, the main structure consists of three parts, the input part, the sever part and the globe part. Therefore, the packages should be included in the sever part.

4.2 Interface

The main interface consists of two parts, the upload part and the feature tabs. As can be seen in Fig. 4.2, the upload part stays in the left part of the main window. In this part, the web service provides different option depend on the data. User could choose to include the data header, the quotation mark and the data separator (in this case, CSV file may be separated by common, semicolon or tab). In the function part, web service organized four main parts:

1. Data summary
2. Setting
3. report
4. burrplot

In the data summary tab, the data uploaded by user could be displayed here and the data is organized in the data summary. The Fig. 4.2 illustrate the result of the data summary. The second part is the setting. Users could do the setting after uploading their data. In this part, we provide some options for users to change the plot and the result based on their needs.

The Fig. 4.3 shows the setting options. Users could modify the plot symbols and labels. Then they could change the plot features like log scale 95% bounds and
4.2 BurrliOZ

Figure 4.3: The setting plot

probability. Furthermore, users could specify the X axis label and the Y axis label. The last option is the fit option. We provide three fit distribution rather than two from windows version. After the setting done, user could press apply button to implement the new options. The next two tabs shows the result of the output, the report and plot respectively.

Figure 4.4: Service Example

4.2.2 Use Example

We provide a example here to show the process. The data that we used is the same one in the experiment part (Tab. 3.1). The data summary and the setting are shown in the Fig. 4.4. For the plot set, we choose “Data” as plot symbols and “Species” as plot labels, then we choose 95% CI bounds. Furthermore, we type “Concentration” as X axis label and “df” as Y axis label. For the fit, we choose Burr type III as the distribution.
Chapter 4 Web services

Figure 4.5: Result

The Fig.4.5 is the result of the process. The left one is report and the right one is plot generated based on the setting we did above. The report and the plot are designed to fit the web browser that means end users could easily use the web services on different kind of devices such as PC, Mac or mobile device like iPad. In the report tab, we provide a function to end users to get a txt copy, and we also provide a function in the plot tab for end user getting a pdf copy of the plot.

4.2.3 Code and Structure

As we mentioned above, we arranged three parts to build the web application. For the UI R file, we separated the the window into two parts. For the upload, we put the interface in the left side by using “sidebarPanel()” function and “fileInput()” function. The example is:

\[
\text{fileInput('file1', 'ChooseCSV File', accept = c('text/csv', 'text/comma-separated-values, text/plain'))}
\]

To set the tabs, we used tabsetPanel() and tabPanel() functions. The example is:

\[
\text{tabsetPanel(tabPanel("DataSummary", tableOutput("contents")))}
\]

To use the BurrliOZ package, we use the function load('fromBurrliOz2.RData'); in the global.R file since this package should be seen by every file. In the same file, we also wrote a function to write the report since it would be called several times.

In the server.R file, the main function is to generate the model based on the data and the setting. The main function to do this are model <- fit(data, density); bs <- bootstrap(model). However, the package is designed for the windows interface or R code user, which is not easy to build the UI on it directly. Our solution is to open the package and extract the main functions code that call many basic functions and separate them into basic functions and call them directly from the web application. Therefore, generally speaking, the development is the process of re-organized the original R package. For example, in order to draw the plot, we rewrite the function plot.bo() to fit its parameters to the web interface.
4.3 NEC

There are two ways to implement the original WinBUGS code into R. The first is that install a WinBUGS software on a Microsoft Windows computer and then install a R package called R2WinBUGS, and the second is that user could install OpenBUGS on OS(Linux, OS X or MS Windows) and install a R package called R2OpenBUGS. The basic idea is the same with these two method, which is storing the code in a txt file and call it by a function from R2WinBUGS or R2OpenBUGS. However, some previous work should be done before running the program such as setting the priors distribution, iteration times and burning times. In the software, it can be done by menu options. We will provide these kind of options in our web service.

4.3.1 Interface

Just like the BurrliOZ web service, the window consists of three parts that are main panel which is used to display the main output, the left side panel which is used to make some options and upload data and the tabs which is used to display different windows.

![Figure 4.6: NEC interface](image)

As can be seen in the Fig. 4.6, the main interface has 4 parts. Instead of put the setting option as a single window, we put them in the side panel, which would be helpful to end users who may want to modify the setting based on the result. In the main interface, the first part is still the upload part, the basic is the same with the BurrliOZ web service. We enhance the function of the header option, since if the data is uploaded without header, we should add them since the data represent each parameters that would be used in the next step, therefore the header is extremely important. The second part is tabs as well as a short instruction about the usage of this service. We add this since the web service is new interface rather than the familiar one like BurrliOZ. The Third part is the MCMC options. It is clear that
Chapter 4
Web services

end users could set iteration times, the “burr in” times and the thin here. In our
design, our offer two distributions with their own parameters. The parameters would
change based on the choice of the distribution. Each time when users update their
settings, they just press update and the application would run a new process based
on the new settings. The main functions of this application is to generate plots to
shows the result. The fourth part here is to show the data summary, but this part
in the parameter tab, $i^{th}$ would show the plot. We would illustrate this part by a
example later.

4.3.2 Code and Structure

The main function in our application is to call the function from R2OpenBUGS(see
why not WinBUGS in the next chapter) by using :

```
bugs(data, inits, parameters, "NEC1.txt", n.chains = 1, n.iter = 110000, n.thin =
10, n.burnin = 10000)
```

As can be seen, to do this function, the application should prepare the data, the
initial data, the parameters from the data and the BUGS code. The rest of the
parameters of this function is the basic setting of OpenBUGS and we also provide
interface for users to modify them. We write a function called `bugrun <- reactive()
` to implement the bugs function. In this function, we used “reactive” function which
is widely used in this application. The application could react in real time to any
modification from the interface. After setting and running, the application would
get a model. To restore the parameters of the estimation, we use a function called
`attach.bugs()`, which may transfer all the parameters before to the R space that
could be visited by functions.

4.3.3 Use Example

Here is an example to show the process. The data we used is the same one in the
experiment part (Tab. 3.2). The Fig. 4.7 shows the data summary of the result based
on the setting in the left and the right one shows the plots that generated by the
MCMC process.

![Figure 4.7: Example](image-url)
In this chapter, we introduce the two web services that we build for end users. We talked about the interface, the function and the code structure of them. For now, our contribution mainly focus on providing new applications. In the next chapters, we will talk about the problems that we meet during the project and give our own opinion about them.
5 Modification and Discussion

5.1 WinBUGS and OpenBUGS

5.1.1 WinBUGS Error

As we mentioned above, when we tried to follow the process of Fox’s[9] experiment step. We found the error occurred as shown in left bottom of Fig. 5.1. It says “cannot bracket slice for node gamma”, which means we cannot update or run the process by using such parameters that is too small. However, Fox just did it by using them.

![WinBUGS Error](image)

**Figure 5.1: WinBUGS Error**

We tried different way to approach Fox’s result by setting the dgamma parameters as smaller as WinBUGS could accept. After testing on it, we found the number is 0.0032085. Therefore, we use gamma ~ dgamma(0.0001, 0.0032085) instead of gamma ~ dgamma(0.0001, 0.0001) as gamma’s priors distribution. The result is shown in Fig. 5.2. Most of the result is reasonable compared with Fox’s result except for the 2.5% of gamma. Fox’s result is 0.01 and our value is 0.8659.
Actually, the experiment above that just modify the second parameter of the gamma distribution may not be reasonable since we have change the variance of this distribution. This may lead the prior distribution not to be non-informative.

5.1.2 Comparison with OpenBUGS and WinBUGS

To solve the problem above that the priors distribution may be not biased on non-informative. We decide to use OpenBUGS instead of WinBUGS to do the process. Before doing this, we should confirm the two kind of software or package are replaceable to each other. We checked the website, the difference is slight accordingly[18]. The differences focus on interaction and efficient. We also checked the the distribution manual of these two software. As can be seen the gamma distribution there are same :

\[ x \sim dgamma(r, \mu) \Rightarrow \frac{\mu^r x^{r-1} e^{-\mu x}}{\Gamma(r)} ; \quad x > 0 \quad (5.1) \]

As shown in the 5.1, the mean is \( \frac{\mu}{\mu} \) and the variance is \( \frac{\mu^2}{\mu^2} \). Accordingly we think it is reasonable to use OpenBUGS instead of WinBUGS.

5.2 New Distribution and New structure

In the NEC web service, we provide a chance for end users by using normal distribution instead of gamma distribution as the priors. In Fox’s paper [9], he claims that the choice of distribution is flexible for users. In MCMC, we just need the priors to be non-informative. In this case, we though that, it could be possible for offering normal distribution non-informative. Therefore, we provide the normal distribution with changeable parameters to end users. In order to improve the performance, we change the whole structure of the code to make it possible to restore a log with user’s setting and restore the running result to decrease the iterate times.
For the problem we met, the WinBUGS that cannot bracket slice for node gamma comes first. In our opinion, the result comes that WinBUGS has not been updated for years, and the development team has abandoned it after doing the last modification. In another word, Fox may use a different version with us or after the last modification, some bugs of the software occurs. However, the OpenBUGS has been proved to be a reliable tool supporting variety platforms. We just take this advantage to build the application on a Linux server, which cannot be done by using WinBUGS.

For the result that cannot be repeated, especially the 2.5% of gamma, we thought it may be a mistake from Fox. Firstly, we followed every step that coming from Fox; secondly, as shown in Fig. 4.7, the difference among gamma is similar with alpha. The sd is similar either, both lower than 0.01. Therefor the number is reasonable to be 0.866. We also attribute this mistake to the version of WinBUGS. Finally, it could be that the typo of Fox could be mistaken.
6 Future work and Conclusion

6.1 Conclusion

To conclude, we have learned tested different models by using different tools and we also have tried to followed the previous work. Our contributions include:

1. Revisiting and experimenting Shao’s and Fox’s work, and we find there seems to be different from Fox’s result with ours, we try to explain the difference and give our own solutions;
2. implementing BurrliOZ into web service;
3. implementing NEC into web service;
4. providing figures with published quality
5. extending Fox’s work via providing more flexibility in the web service like normal distribution and Weibull distribution

To sum up, we have tested the correct of the previous work and give our own contribution to fill the blank or the work. In the future, we will try to improve our current work and develop more possibilities about this project.

6.2 Future work

Our work so far has shown a lot of possibilities in the future. Generally speaking, there were two kind of things could be done based on our current work, the improvement and the development. For the improvement part, we still have a lot to do to enhance our work including:

1. update the BurriOZ package;
2. improve the performance of BurrliOZ;
3. more possible priors distributions in NEC.
4. rewriting the front-end by using JavaScript;

For each of them above, the improvement is regarded as software update. We could use new technology to improve the experience of end users. On the other hand, we may develop more possibilities about the project like:
1. testing new possible models;
2. comparing NOEC. vs NEC or other models
3. using different tools and packages;
4. using the same way in different fields

For this part, we prefer to test the possibilities about the project and trying to implement our current work on different fields.
Acknowledgments

I would like to thank my parents first.

I would like to thank my supervisor, Dr. Warren Jin, he guided me with extremely patient and gave me courage when I feel upset about the project. Dr. Jin taught me not only the knowledge but the way to do research and find more possibilities in me.

Then I would like to thank DR. Weifa Liang for weekly meeting and being patient to our questions.

Finally I would like to thank ANU College of Engineering and Computer Science for giving this chance to experience research and enhance my abilities.
Bibliography


