2020

Bayesian Approach to Finding The Most Likely Circuit Structure

Shannon Harms
s.harmss21@gmail.com

ABSTRACT

BAYESIAN APPROACH TO FINDING THE MOST LIKELY CIRCUIT STRUCTURE

Shannon Harms, M.S.
Department of Statistics and Actuarial Science
Northern Illinois University, 2020
Dr. Alan M. Polansky, Director

Systems, and their reliabilities, depend on the reliabilities of the components that they are composed of, and in this paper we want to find the system structure that is the most likely given observed data. Bayesian methods were utilized in order to discover the posterior means, or observed reliabilities, of both the components and the systems. Assuming the serial and parallel system structures have independent components, we calculated system reliabilities based on observed component reliabilities by using the multiplication and addition probability rules. We are then able to expand upon the numerical comparison method through a maximum likelihood analysis that compares the computed system reliability to the observed system reliability. To account for any variation, we then simulate new data using the observed data that is passed through our maximum likelihood analysis in order to discover the most likely system structure. These functions are developed using R code; and our process is illustrated using an example dataset.

NORTHERN ILLINOIS UNIVERSITY
DE KALB, ILLINOIS

DECEMBER 2020

BAYESIAN APPROACH TO FINDING THE MOST LIKELY CIRCUIT STRUCTURE

BY

SHANNON HARMS

A THESIS SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE

MASTER OF SCIENCE

DEPARTMENT OF STATISTICS AND ACTUARIAL SCIENCE

Thesis Director:
Dr. Alan M. Polansky

ACKNOWLEDGEMENTS

I am exponentially grateful for Dr. Alan Polansky's enthusiasm, assistance, and guidance.

DEDICATION

To my family, for their unfaltering love and support

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Systems have a wide range of uses; from computer operations and software to mechanical networks, they are an integral part of the engineering world and beyond. Many functions both in the workforce and in everyday life can be stripped bare to reveal a system beneath. These systems are an essential aspect in the operation of both simple and complex functions.

1.1 System Structures and Reliability Assessments

For a system to be successful and functional, the performance of its elements, or components, must be successful on an individual level but also be cohesive within the system (Lu, 2019). This compatibility between the components is considered to be the reliability of the system's success (Lu, 2019). These assessments of reliability in terms of components contain valuable information about the systems and are often less expensive and less invasive to run (Lu, 2019). The two most common ways components can be connected within the system are by series and parallel structures (Lu, 2019).

As shown in Figure 1.1, in order for a system with a series structure to be successful, all four components must work (Lu, 2019). Similarly, Figure 1.2 illustrates that only one of the four components must work within a parallel structure for the system to be successful (Lu, 2019). This can only be achieved when we assume the components are independent, but not mutually exclusive (Lu, 2019). Additionally, components will be observed in the form of pass/fail data, and the reliabilities of the systems must be consistent (Lu, 2019).

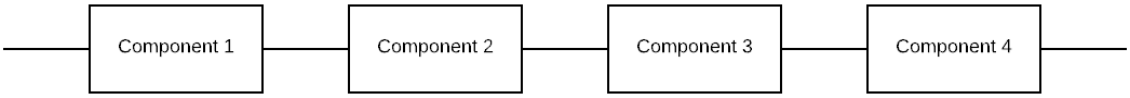

Figure 1.1: A series system with $n = 4$ components.

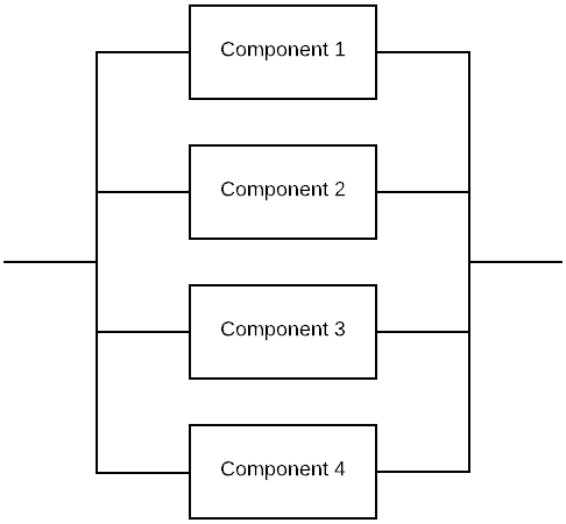

Figure 1.2: A parallel system with $n = 4$ components.

Using the rules of probability, the reliability for a system, p, with a series structure can be calculated as

$$p = \Pi_{k=1}^{m} p_k \tag{1.1}$$

when p_k is the reliability of the kth component in the system for $k \epsilon$ {1,...,m} (Lu, 2019). Likewise, the reliability for a parallel structure is computed by

$$p = 1 - \Pi_{k=1}^{m}(1 - p_k), \tag{1.2}$$

which concludes that at least one component works (Lu, 2019). To accurately estimate the system reliability, Anderson-Cook's (2009) frequentist hypothesis test will be used. This hypothesis test is based on the maximum likelihood estimates for the system and the component reliabilities (Lu, 2019).

1.2 Bayesian Evaluation

The Bayesian model we will use allows the differences between estimated and expected system reliabilities to be quantified, thus permitting helpful diagnostics and failure detection for the system structure (Lu, 2019). In addition, this method is adaptable and can be used for more complex system structures (Lu, 2019).

Circuits are the systems we will be focusing on. We let Y be the number of successful circuit functions in n replications. Similarly, we let Y_k represent the number of times component k, in the circuit, functions successfully during n_k replications for $k \epsilon$ {1,...,m}. We then assume that the observed binary data can be modeled by $Y \sim Binomial(n, p)$ and

$Y_k \sim Binomial(n_k, p_k)$ for the circuit and components respectively (Lu, 2019). Again, for this paper we assume that $Y, Y_1, ..., Y_k$ are mutually independent.

The variance, or discrepancy, between the observed system reliability and the computed system reliability, or the combined component reliability under the assumed structure, can be quantified in both the multiplicative and the additive forms (Lu, 2019). The multiplicative discrepancy parameter,

$$\delta_1 = \frac{p}{\prod_{k=1}^{m} p_k}, \tag{1.3}$$

quantifies the compatibility of the observed system and the computed system reliabilities (Lu, 2019).

The true system reliability is considered consistent with the computed system reliability under the assumed structure given the combined component reliabilities, if $\delta_1 = 1$ (Lu, 2019). Additionally, if $\delta_1 < 1$, then the computed system reliability is considered better than the true system reliability, meaning either a mock component is causing failure of the system, there is a connective issue between the components, or there is dependence between the components (Lu, 2019). Moreover, $\delta_1 > 1$ shows that the computed system reliability is worse than the true system reliability, indicating that the component reliabilities could be miscalculated (Lu, 2019). Similarly, the additive form can also be used to quantify the discrepancy (Lu, 2019).

Usually for a Bayesian analysis of reliability, distributions of model parameters are drawn based on the prior distributions before inference is made based on the parameter's posterior distributions given the observed data (Lu, 2019). These prior distributions tend to include knowledge or historical data on the performances given the parameter's reliabilities (Lu, 2019). However, not all data has historical information, and so a non-informative prior can be used instead (Lu, 2019).

The frequentist maximum likelihood approach for testing assumptions of system structures suggested by Anderson-Cook (2009) is comparable to Lu (2019) proposing to test the discrepancy measure, $H_0 : \delta_1 = 1$, using a Bayesian analysis with non-informative priors due to limited information on prior distributions (Lu, 2019). Thus, a conjugate prior distribution (Hamada et al., 2008, page 31), $p_k \sim Beta(a_k, b_k)$, with a probability density function (PDF) of

$$f(p_k|a_k, b_k) = \frac{\Gamma(a_k + b_k)}{\Gamma(a_k)\Gamma(b_k)} p_k^{a_k-1}(1 - p_k)^{b_k-1}, \tag{1.4}$$

and a Gamma function of $\Gamma(\alpha) = \int_0^\infty s^{\alpha-1}e^{-s}ds$, $\alpha > 0$, is chosen for the component reliability (Lu, 2019). The hyperparameters a_k and b_k are essentially one plus the number of observed successes and failures from historical data (Lu, 2019); however, if there is no historical data, then $a_k = b_k = 1$ for all $k \epsilon$ {1,...,m} (Lu, 2019).

The lognormal prior distribution, $\log(\delta_1) \sim N(\mu_1, \sigma_1^2)$, was chosen for the discrepancy parameter δ_1 (Lu, 2019). The PDF of this lognormal distribution is

$$f(\delta_1|\mu_1, \sigma_1^2) = \frac{1}{\delta_1\sigma_1\sqrt{2\pi}} exp\left\{-\frac{1}{2\sigma_1^2}[log(\delta_1) - \mu_1]^2\right\}, \tag{1.5}$$

with the mean and variance, μ_1 and σ_1, of the normal distribution of $\log(\delta_1)$ being the hyperparameters (Lu, 2019). Due to the lack of prior knowledge for the discrepancy, Lu (2019) chose $\mu_1 = 0$ and $\sigma_1 = 1$, which gave δ_1 a range of (0.1, 7.1) with a 95% probability. This range is quite large due to the non-informative priors; however, if historical information is available, then a smaller σ_1 can be used (Lu, 2019).

The observed data is then used to compute the posterior distribution of the model parameters,

$$\begin{aligned} &\pi(p_1, ..., p_m, \delta_1|y_1, ..., y_m, y) \\ &\propto f(y_1, ..., y_m, y|p_1, ..., p_m, \delta_1) \prod_{k=1}^{m} \pi(p_k)\pi(\delta_1), \end{aligned} \tag{1.6}$$

when $\pi(p_k)$ denotes the PDF given in Equation 1.4, $\pi(\delta_1)$ denotes the PDF given in Equation 1.5, and $f(y_1, ..., y_m, y|p_1, ..., p_m, \delta_1)$ is the likelihood function

$$\begin{aligned} & f(y_1, ..., y_m, y|p_1, ..., p_m, \delta_1) \\ & = f(y|p_1, ..., p_m, \delta_1) \cdot f(y_1, ..., y_m|p_1, ..., p_m) \\ & = \binom{n}{y} \left[\delta_1 \prod_{k=1}^{m} p_k\right]^y \left[1 - \delta_1 \prod_{k=1}^{m} p_k\right]^{n-y} \cdot \prod_{k=1}^{m} \left\{ \binom{n_k}{y_k} p_k^{y_k} (1-p_k)^{n_k - y_k} \right\}, \end{aligned} \tag{1.7}$$

where the series system reliability is written as $p = \delta_1 \prod_{k=1}^{m} p_k$ when δ_1 is used to adjust the discrepancy between the observed system reliability and the computed system reliability (Lu, 2019). The likelihood of the system data, $f(y|p) = \binom{n}{y} p^y [1-p]^{n-y}$, is equivalent to the first term in Equation 1.7 when p is substituted with $\delta_1 \prod_{k=1}^{m} p_k$ (Lu, 2019). Thus, the product of the likelihoods of the system and individual component data is equal to the joint likelihood in Equation 1.7 when independence is assumed between the components and system (Lu, 2019).

Similarly, when the additive discrepancy measure δ_2 is used, the normal prior distribution, $\delta_2 \sim N(\mu_2 = 0, \sigma_2^2)$, with the PDF

$$f(\delta_2|\mu_2, \sigma_2^2) = \frac{1}{\sigma_2\sqrt{2\pi}} exp\left\{ -\frac{1}{2\sigma_2^2} [\delta_2 - \mu_2]^2 \right\} \tag{1.8}$$

is recommended by Lu (2019). Without information on the discrepancy of the prior, σ_2^2 is then given a range of possible values (Lu, 2019). Thus, the joint likelihood from Equation 1.6, $f(y_1, ..., y_m, y|p_1, ..., p_m, \delta_2)$, can be used to calculate the posterior distribution

$$\binom{n}{y} \left[\delta_2 + \prod_{k=1}^{m} p_k\right]^y \left[1 - \delta_2 - \prod_{k=1}^{m} p_k\right]^{n-y} \cdot \prod_{k=1}^{m} \left\{ \binom{n_k}{y_k} p_k^{y_k} (1-p_k)^{n_k - y_k} \right\}, \tag{1.9}$$

where the reliability of the parallel system is adjusted for discrepancy through $p = \delta_2 + \prod_{k=1}^{m} p_k$ (Lu, 2019).

The Gibbs sampler approach (Gelfand & Smith, 1990; Gelman et al., 2013; Geman and Geman, 1984) and the Markov Chain Monte Carlo (MCMC) simulation are used to approximate a posterior distribution from Equation 1.6 (Lu, 2019). Lu (2019) used the rjags package to implement this calculation into R (R Development Core Team, 2016). This R-code provided an interface to the JAGS library (Plummer, 2003) for the purpose of implementing the Bayesian data analysis with the Gibbs sampler (Lu, 2019). Thus, summaries of the joint or marginal posterior distributions are used for the analysis after samples from the approximated posterior distribution are obtained (Lu, 2019).

Furthermore, a credible interval is used when estimating the multiplicative discrepancy, δ_1, in order to better comprehend the difference between the observed system data and the computed system data given the combined component data (Lu, 2019). For example, when using the MCMC simulation, the 0.025 and 0.975 quantiles of the posterior marginal distribution are used to find the 95% credible interval (Lu, 2019). Next, the posterior predictive p value (Gelman et al. 2013), which is the probability that the results of a test using the posterior distribution is more drastic than expected from the observed data, is computed (Lu, 2019).

When testing $H_0 : \delta_1 = 1$ versus $H_T : \delta_1 \neq 1$, the Bayesian p value is computed by

$$p_T = 2min\{Pr(\delta_1 > 1|y, y_1, ..., y_m), Pr(\delta_1 < 1|y, y_1, ..., y_m)\}; \tag{1.10}$$

however, when $H_U : \delta_1 < 1$, or the demand on a component is suspected to be lower than predicted, the Bayesian p value is calculated as $p_U = Pr(\delta_1 > 1|y, y_1, ..., y_m)$, and when $H_L : \delta_1 > 1$, or that the components are correlated in some way, the Bayesian p value is calculated as $p_L = Pr(\delta_1 < 1|y, y_1, ..., y_m)$ (Lu, 2019).

The five-component series system from Anderson-Cook (2009) is used to first have the Gibbs sampler approximate the joint posterior distribution, then three MCMC chains with 5,000 iterations run before 30,000 MCMC samples are drawn, and in order to reduce the autocorrelation between adjacent samples, only every 5th observation was reported (Lu, 2019).

Table 1.1 displays the series system data, including the Bayesian estimates of reliabilities with their 95% credible intervals (Lu, 2019). Additionally, Lu (2019) provides that the posterior mean of the marginal distribution, δ_1, is 0.85 with a 95% credible interval of (0.60, 1.08), and because this interval does contain 1, then it can be concluded that there is no significant evidence that the system and component data are inconsistent given the assumed system structure. Continuing with the analysis, testing the two-sided posterior, $H_0 : \delta_1 = 1$, Lu (2019) reports that $p_T = 0.22$, indicating there is no significant discrepancy. This is advantageous because this estimated discrepancy can be used to assess system reliability that will be estimated from eventual component test data (Lu, 2019).

Next, we assess a three-component parallel system from Anderson-Cook (2009) shown in Table 1.2. The multiplicative discrepancy formula,

$$\delta_1 = \frac{p}{1 - \prod_{k=1}^{m}(1 - p_k)}, \tag{1.11}$$

is given from the assumed structure in Equation 1.2 (Lu, 2019). While the same prior distributions from Equations 1.4, 1.5, and 1.8 are used for the component reliabilities, the

Table 1.1: The five-component series system data (Anderson-Cook, 2009) including the posterior estimates of the system and component reliabilities (Lu, 2019).

Data Type	Number of Successes	Sample Size	Posterior mean (95% Credible Interval)
System	17	25	0.657 (0.470, 0.821)
Component 1	111	120	0.918 (0.863, 0.960)
Component 2	96	100	0.951 (0.902, 0.984)
Component 3	76	80	0.939 (0.877, 0.980)
Component 4	98	100	0.971 (0.931, 0.994)
Component 5	79	80	0.976 (0.933, 0.997)

Table 1.2: The three-component parallel system data (Anderson-Cook, 2009) including the posterior estimates of the system and component reliabilities found by Lu (2019).

Data Type	Number of Successes	Sample Size	Posterior mean (95% Credible Interval)
System	33	40	0.806 (0.675, 0.909)
Component 1	136	200	0.678 (0.612, 0.742)
Component 2	54	80	0.671 (0.565, 0.768)
Component 3	31	40	0.516 (0.394, 0.638)

additive discrepancy measures, and the multiplicative discrepancy measures, a new joint likelihood,

$$f(y_1,...,y_m,y|p_1,...,p_m,\delta_1) = \binom{n}{y}\left\{\delta_1\left[1-\prod_{k=1}^{m}(1-p_k)\right]\right\}^{y} \left\{1-\delta_1\left[1-\prod_{k=1}^{m}(1-p_k)\right]\right\}^{n-y} \cdot \prod_{k=1}^{m}\left\{\binom{n_k}{y_k}p_k^{y_k}(1-p_k)^{n_k-y_k}\right\}, \tag{1.12}$$

is used based on the multiplicative discrepancy, δ_1, for the parallel system (Lu, 2019).

Similarly to the method used for the series system, the MCMC simulation is used to find the posterior mean and the credible intervals for the parallel system and component reliabilities which are summarized in Table 1.2. Additionally, Lu (2019) provides that the posterior mean of the multiplicative discrepancy, δ_1, is 0.85 with a 95% credible interval of (0.71, 0.96), and because this interval does not contain 1, then it can be concluded that there is a significant discrepancy between the system and component. Continuing with the analysis, testing the two-sided posterior, $H_0 : \delta_1 = 1$, Lu (2019) reports that $p_T = 0.003$, indicating there is an inconsistency between different test data when using the assumed structure.

With the Bayesian methods implemented by Lu (2019), we are able to expand upon the numerical comparison method through a maximum likelihood analysis in addition to a simulation of new data using the observed data in order to discover the most likely circuit structure. Lu's (2019) in-depth approach to finding the posterior mean, or reliability, and checking the discrepancy creates a building block for us to develop a function that will compare an observed circuit reliability to the observed component reliabilities, otherwise known as posterior means.

CHAPTER 2

STATISTICAL MODELS

Throughout this paper, we will denote the parallel building blocks for a circuit as $C_1 \oplus C_2$, requiring only one component to work properly, and the serial building blocks for a circuit as $C_1 \otimes C_2$, requiring all components within a circuit to function correctly. Please refer to Appendix: R-Code for the R-code written in 2019 by Dr. Alan Polansky and Shannon Harms that is described in this chapter and utilized in Chapter 3.

2.1 Creating Circuits in R

To begin, the function `addComponent` is used to create an object in R that compiles all possible circuits for a given quantity of components. This function repeats the processes generating a sequence of outcomes, and each outcome becomes the starting point of the next repetition.

The default argument for our function is the following: `addComponent <- function( circuit=NULL)`; thus, if the default is used, NULL will be passed through the function which will return the single circuit that exists for a component. The R output for the single circuit is shown in Figure 2.1. This output shows that `Circuit 1` is equal to 1, thus concluding that there is only one component. In addition, `Formula 1` assumes that the reliability of the component is given by $p[1]$, which is then used to describe how the reliability of the circuit is calculated. The attribute named *components* lists the number of components used in the

circuits, and similarly the attribute named *circuits* lists the number of circuits created when using the specified number of components. In this instance, both attributes are equal to 1.

```
$`Circuit 1`
[1] "1"

$`Formula 1`
[1] "p[1]"

attr(,"components")
[1] 1
attr(,"circuits")
[1] 1
```

Figure 2.1: R-output for `C1 <- addComponent(NULL)`.

Because the function `addComponent` is a repeating process, an object created by this function can then be passed back through the function. When this happens, the current component in a specified circuit is replaced in both the parallel and serial sub-circuits by the current component and the new component. Thus a new object is created that contains the additional circuits for the newly specified number of components. For example, after running `C1 <- addComponent(NULL)` to get the output in Figure 2.1, we then run our first object, `C1` back through the function in order to get the second object, `C2 <- addComponent(C1)`, which gives us the output in Figure 2.2.

In order to compute the circuit reliabilities, the formulas for the circuits are written as functions in the code. First, the addition rule of probability for non-mutually exclusive components, `aRule <- function(p,q) p+q-p*q`, is written as a function to represent parallel component structure. Additionally, we assume that the components within a circuit are independent; thus, multiplication rule of probability is also given as a function to represent the serial component structure, `mRule <- function(p,q) p*q`.

The output in Figure 2.2 shows that `Circuit 1` is composed of two parallel components `(1+2)`, representing the circuit $C_1 \oplus C_2$. Thus, `Formula 1` states that the addition rule of

probability, `aRule`, is used to compute the reliability of `Circuit 1` when the reliability of the components are given by $p[1]$ and $p[2]$. Similarly, `Circuit 2` contains two components `(1*2)`, but these components are sequential, so the circuit can also be written as $C_1 \otimes C_2$. Assuming the reliability of the components are $p[1]$ and $p[2]$, then `Formula 2` states that the multiplication rule of probability, `mRule`, is used to find the reliability of `Circuit 2`. In this case, both attributes, *components* and *circuits*, are equal to 2.

```
$‘Circuit 1‘
[1] "(1+2)"

$‘Formula 1‘
[1] "aRule(p[1],p[2])"

$‘Circuit 2‘
[1] "(1*2)"

$‘Formula 2‘
[1] "mRule(p[1],p[2])"

attr(,"components")
[1] 2
attr(,"circuits")
[1] 2
```

Figure 2.2: R-output for `C2 <- addComponent(C1)`.

To better understand the function's procedures, we can take it a step further and send `C2` through the function, `C3 <- addComponent(C2)`. The function begins by replacing each component within the object's circuits as shown in Table 2.1. Object `C2` has two circuits each containing two components, `(1+2)` and `(1*2)`. The first component in `Circuit 1` will be replaced by the new component parallel to the first component, `(1+3)`, in order to create the new circuit, `((1+3)+2)`, or $(C_1 \oplus C_3) \oplus C_2$. This new circuit structure can be seen in Figure 2.3. Then the first component in `Circuit 1` will be replaced by the new component serial to the first component, `(1*3)`, in order to create the new circuit, `((1*3)+2)`, or $(C_1 \otimes C_3) \oplus C_2$.

Table 2.1: Table explaining how new circuits are created in `C3 <- addComponent(C2)`.

Beginning Circuit	Component Being Replaced	New Component	New Circuit
(1+2)	Component 1	(1+3)	((1+3)+2)
(1+2)	Component 1	(1*3)	((1*3)+2)
(1+2)	Component 2	(2+3)	(1+(2+3))
(1+2)	Component 2	(2*3)	(1+(2*3))
(1*2)	Component 1	(1+3)	((1+3)*2)
(1*2)	Component 1	(1*3)	((1*3)*2)
(1*2)	Component 2	(2+3)	(1*(2+3))
(1*2)	Component 2	(2*3)	(1*(2*3))

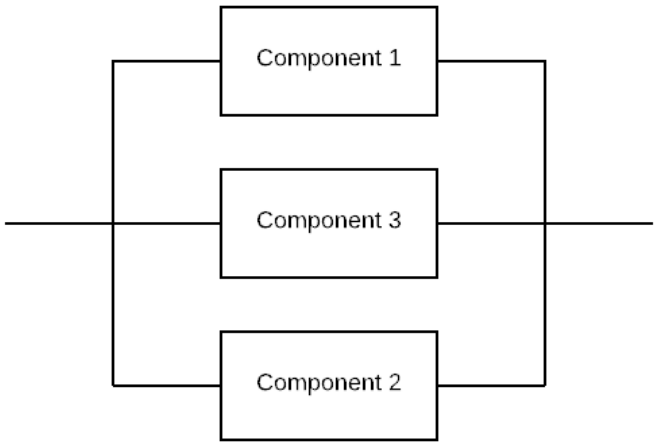

Figure 2.3: Circuit structure, $(C_1 \oplus C_3) \oplus C_2$.

Figure 2.4 shows the new circuit structure. This process continues for all components in all circuits. In this case, eight new circuits are created each with three components.

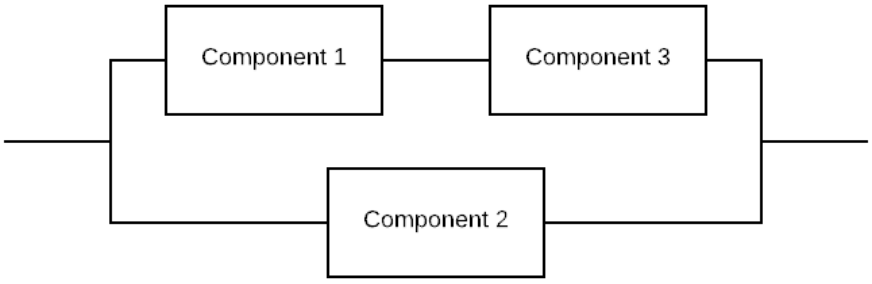

Figure 2.4: Circuit structure, $(C_1 \otimes C_3) \oplus C_2$.

The next step the function takes is to remove circuits that have corresponding structures, as they are considered equivalent. To check this, we begin by looking back at Table 2.1 in order to use the new circuits so that we may compute the formulas for the circuits. When considering the circuit $(C_1 \oplus C_3) \oplus C_2$, and recalling `aRule` defined previously as `function(p,q) p+q-p*q`, we can determine that the reliability formula is `aRule(aRule(p[1],p[3]),p[2])`. This formula translates into

$$(p[1] + p[3] - p[1] * p[3]) + p[2] - (p[1] + p[3] - p[1] * p[3]) * p[2]. \tag{2.1}$$

Each component is treated as either completely reliable or completely unreliable and assigned a value, $p = 1$ or $p = 0$ respectively. The output in Figure 2.5 first shows each possible combination of reliabilities for the three components and then gives the profile of each circuit for their corresponding reliability.

To better visualize the output, Table 2.2 was created to show the reliability results for circuit $(C_1 \oplus C_3) \oplus C_2$. So, when the component's reliabilities, `p[1], p[2], p[3]`, are

```
     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    0    1
[3,]    0    1    0
[4,]    0    1    1
[5,]    1    0    0
[6,]    1    0    1
[7,]    1    1    0
[8,]    1    1    1
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    0    1    1    1    1    1    1    1
[2,]    0    0    1    1    0    1    1    1
[3,]    0    0    0    1    0    0    1    1
[4,]    0    0    0    0    0    0    0    1
[5,]    0    1    1    1    1    1    1    1
[6,]    0    0    0    1    1    1    1    1
[7,]    0    0    0    0    0    1    1    1
[8,]    0    0    0    0    0    0    0    1
keepCircuits =  1 2 3 4 6 7 0 0
keepCircuits =  1 2 3 4 6 7
```

Figure 2.5: R-output of circuit profiles for the circuits created in `C3 <- addComponent(C2)`.

Table 2.2: Table of reliability results and circuit profile for circuit $(C_1 \oplus C_3) \oplus C_2$.

p[1]	p[2]	p[3]	aRule(aRule(p[1],p[3]),p[2])
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

plugged into the formula given in Equation 2.1, the circuit's reliabilities are listed under `aRule(aRule(p[1],p[3]),p[2])`.

For example, if we plug `p[1] = 0, p[2] = 0, p[3] = 1` into the formula from Equation 2.1, we get

$$(0 + 1 - 0 * 1) + 0 - (0 + 1 - 0 * 1) * 0, \tag{2.2}$$

which is equal to 1. Therefore, when all three components are unreliable, then the circuit will also be unreliable. Additionally, when at least one of the components is reliable, then the circuit will also be reliable. Hence, circuit $(C_1 \oplus C_3) \oplus C_2$'s profile is $(0, 1, 1, 1, 1, 1, 1, 1)$.

Next, we consider the circuit $(C_1 \otimes C_3) \oplus C_2$, which incorporates both the `aRule` and `mRule` that was previously defined as `function(p,q) p*q`. The reliability formula can then be shown as `aRule(mRule(p[1],p[3]),p[2])`, which translates into

$$(p[1] * p[3]) + p[2] - p[1] * p[3] * p[2]. \tag{2.3}$$

Similarly, if we use `p[1] = 0, p[2] = 0, p[3] = 1` to plug into the equation from Equation 2.3,

$$(0 * 1) + 0 - 0 * 1 * 0, \tag{2.4}$$

the result will be 0. This, and the other reliability results for the circuits are shown in Table 2.3. Thus, circuit $(C_1 \otimes C_3) \oplus C_2$'s profile is $(0, 0, 1, 1, 0, 1, 1, 1)$.

Finally, we consider circuit $C_1 \oplus (C_2 \oplus C_3)$. This circuit only depends on the `aRule`, and the reliability formula is `aRule(p[1],aRule(p[2]),p[3]))`. This formula translates into

$$p[1] + (p[2] + p[3] - p[2] * p[3]) - p[1] * (p[2] + p[3] - p[2] * p[3]). \tag{2.5}$$

Again, when `p[1] = 0, p[2] = 0, p[3] = 1` is plugged into the formula from Equation 2.5, we get

$$0 + (0 + 1 - 0 * 1) - 0 * (0 + 1 - 0 * 1), \tag{2.6}$$

which is equal to 1. The circuit reliabilities are given in Table 2.4 and show that circuit $C_1 \oplus (C_2 \oplus C_3)$'s profile is $(0, 1, 1, 1, 1, 1, 1, 1)$. Circuit $(C_1 \oplus C_3) \oplus C_2$ and circuit $C_1 \oplus (C_2 \oplus C_3)$ have identical reliability profiles, thus they are considered equivalent, and only one is kept in the final assemble of circuits.

Table 2.3: Table of reliability results and circuit profile for circuit $(C_1 \otimes C_3) \oplus C_2$.

p[1]	p[2]	p[3]	aRule(mRule(p[1],p[3]),p[2])
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Table 2.4: Table of reliability results and circuit profile for circuit $C_1 \oplus (C_2 \oplus C_3)$.

p[1]	p[2]	p[3]	aRule(p[1],aRule(p[2]),p[3]))
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Referring back to Figure 2.5, we can see that `keepCircuits = 1 2 3 4 6 7` is shown at the bottom of the output. This indicates that Circuit 5 and Circuit 8 are considered to have a reliability profile identical to a circuit profile that is kept. Finally, the output will provide all circuits with three components, sans circuits with duplicate reliability profiles, as shown in Figure 2.6.

The function `decimalToBinary` is used to generate all of the values of $p[1], ..., p[k]$ for the reliability formulas. The process of creating new components with the `addComponent` function can be repeated indefinitely; however, the R program we used does not have the adequate processing power necessary to compute and track circuits beyond about seven.

2.2 Maximum Likelihood Analysis

Once the circuits are created, we wish to find the most probable circuit given observed data, and this is done by comparing the observed component reliabilities to the observed circuit reliabilities using a maximum likelihood analysis.

Our next function, `apparentCircuitMLE`, ultimately returns the apparent circuit structure given the maximum likelihood analysis. To do this, the observed circuit reliability, `pCircuit`; a vector containing the observed component reliabilities, `pComponent`; the object created by the function `addComponent` that contains the circuits for the number of components implied by the length of `pComponent`, `C`; and a logical value determining whether or not a table of the independent circuit reliabilities is to be printed, `verbose`, are arguments required to pass through the function. The default for `verbose` is `FALSE`; therefore, our function is written as `apparentCircuitMLE <- function(pCircuit,pComponent,C,verbose =FALSE)`.

```
$‘Circuit 1‘
[1] "((1+3)+2)"

$‘Formula 1‘
[1] "aRule(aRule(p[1],p[3]),p[2])"

$‘Circuit 2‘
[1] "((1*3)+2)"

$‘Formula 2‘
[1] "aRule(mRule(p[1],p[3]),p[2])"

$‘Circuit 3‘
[1] "((1+3)*2)"

$‘Formula 3‘
[1] "mRule(aRule(p[1],p[3]),p[2])"

$‘Circuit 4‘
[1] "((1*3)*2)"

$‘Formula 4‘
[1] "mRule(mRule(p[1],p[3]),p[2])"

$‘Circuit 5‘
[1] "(1+(2*3))"

$‘Formula 5‘
[1] "aRule(p[1],mRule(p[2],p[3]))"

$‘Circuit 6‘
[1] "(1*(2+3))"

$‘Formula 6‘
[1] "mRule(p[1],aRule(p[2],p[3]))"

attr(,"components")
[1] 3
attr(,"circuits")
[1] 6
```

Figure 2.6: R-output for `C3 <- addComponent(C2)`.

For the specified number of components, `C`, the observed component reliabilities, the vector `pComponent`, are passed through each `Formula` that was provided in the `addComponent` output. These computed circuit reliabilities are then compared to the observed circuit reliability, `pCircuit`. The computed circuit reliability that has the smallest difference from the observed circuit reliability is considered to have the apparent circuit structure.

For example, we can consider a case with three components C_1, C_2, and C_3, with corresponding observed reliabilities, $p_1 = 0.100$, $p_2 = 0.200$, and $p_3 = 0.300$, and an observed circuit reliability of $p = 0.500$. If we would like the function to include a table of independent circuit reliabilities in the output, then our code would be the following: `apparentCircuitMLE(0.500,c(0.100,0.200,0.300),C3,TRUE)`. The list of formulas that would be used in this case are listed in Figure 2.6.

After running the code, the output would include the data found in Table 2.5. This data shows each circuit found in the `addComponent` function given there are three components, as well as the structure for each individual circuit, and finally the reliability for each circuit. Knowing that the observed circuit reliability is $p = 0.500$, we can see that the output seems to indicate that `Circuit 1`, $(C_1 \oplus C_3) \oplus C_2$, has the closest reliability with $p = 0.496$. Additionally, included in the output is the following line: `Apparent Circuit Structure: Circuit 1 with structure ((1+3)+2)`.

Table 2.5: Table of reliability results for `apparentCircuitMLE(0.500,c(0.100,0.200, 0.300),C3,TRUE)`.

Circuit	Structure	Reliability
Circuit 1	((1+3)+2)	0.496
Circuit 2	((1*3)+2)	0.224
Circuit 3	((1+3)*2)	0.074
Circuit 4	((1*3)*2)	0.006
Circuit 5	(1+(2*3))	0.154
Circuit 6	(1*(2+3))	0.044

If $(C_1 \oplus C_3) \oplus C_2$ is the true circuit structure, then we conclude that there is independence between the components in the circuit. However, if $(C_1 \oplus C_3) \oplus C_2$ is not the true circuit structure, then we can conclude that there is dependence between components in the circuit which makes the circuit act as if it had this structure. Unfortunately, we are unable to conclude either way with the information that we have.

2.3 Most Likely Circuit

Once the apparent circuit structure based on the observed data is found, the variation in the data must be accounted for. This variation, or discrepancy, is associated with the uncertainty between the observed circuit reliability and the reliability computed from the component data.

As stated in Section 1.2, we let Y be the number of successful circuit functions in n replications, and we let Y_k represent the number of times component k in the circuit functions successfully during n_k replications for $k \epsilon \{1,...,\mathrm{m}\}$. We then assume that the observed binary data can be modeled by $Y \sim Binomial(n, p)$ and $Y_k \sim Binomial(n_k, p_k)$ for the circuit and components respectively. We also let our estimated circuit reliability be $\hat{p} = n^{-1}Y$ and our estimated component reliability be $\hat{p}_k = n_k^{-1}Y_k$.

The final function, `apparentCircuitMLC`, returns the most likely circuit structure. To do this, the number of times that the circuit functions in n replications, `Y`; the number of replications of the circuit, `n`; a vector containing $Y_1, ..., Y_m$, `Ycomponents`; a vector containing $n_1, ..., n_m$, `nComponents`; and the object created by the function `addComponent` that contains the circuits corresponding to m components, `C`, are arguments required to pass through the function. Thus, the function is written as `apparentCircuitMLC <- function(Y,n, Ycomponents,nComponents,C)`.

Once the function computes the estimated reliabilities given the observed Y, Y_k, n, and n_k, the function then uses $\hat{p}$ and $\hat{p}_k$ to simulate $Y^* \sim Binomial(n, \hat{p})$ and $Y_k^* \sim Binomial(n_k, \hat{p}_k)$ for $k \in \{1,...,m\}$. This simulation repeats 10,000 times and uses the `apparentCircuitMLE` function from Section 2.2 to find the most likely circuit for $Y^*, Y_1^*, ..., Y_m^*$. This is done in order to account for the variation in the data and is based on the parametric bootstrap.

Continuing with our previous example where we consider a case with three components C_1, C_2, and C_3 with corresponding observed reliabilities $p_1 = 0.10$, $p_2 = 0.20$, and $p_3 = 0.30$ and an observed circuit reliability of $p = 0.50$, we can choose a simple case where `n` and `nComponents` are all equal to 100. Thus, in order for the component reliabilities to match, `Y` = 50 and the vector for the `Ycomponents` must be 10, 20, and 30 respectively. Therefore, the arguments in the function will be the following: `apparentCircuitMLC(50,100,c(10,20,30), c(100,100,100),C3)`.

The output is shown in Figure 2.7, which explains that `Circuit 1` is the most likely structure for the largest proportion of time, thus making it the most likely circuit structure overall. This matches our previous findings in the maximum likelihood analysis which expressed that the apparent circuit structure is `Circuit 1`. Therefore, we can conclude that this is the most evident circuit structure, given the information we have.

```
[1] "Proportion of time Circuit 1 is the most likely structure: 0.9881"
[2] "Proportion of time Circuit 2 is the most likely structure: 0.0118"
[3] "Proportion of time Circuit 5 is the most likely structure: 1e-04"
[1] "All Circuits not listed above are never the most likely Circuit."
[1] "The most likely circuit structure overall is Circuit 1"
```

Figure 2.7: R-output for `apparentCircuitMLC(50,100,c(10,20,30),c(100,100,100), C3)`.

CHAPTER 3

RESULTS AND DISCUSSION

Throughout this paper, as a simple example to express how our code works, we have used arbitrary data for our observed reliabilities, number of successes, and sample sizes. To better exhibit the processes of our code, we must compare our results with a small sample of data.

3.1 Expanding Upon Lu Lu's Evaluations

The five-component data first used by Anderson-Cook (2009) and then by Lu (2019) is shown in Table 3.1. The posterior means, or observed reliabilities, were found using the MCMC simulation and Gibbs sampler calculations implemented in R, as discussed in Chapter 1 (Lu, 2019). When running this data through our code, we first find that there are 174 unique circuits that contain five components. Then we plug the observed reliabilities for the system and the five components into our `apparentCircuitMLE` function, `apparentCircuitMLE(0.657,c(0.918,0.951,0.939,0.971,0.976),C5,TRUE)`.

The output reports a table for the independent circuit reliabilities that includes all 174 circuits; however, Table 3.2 only provides a short excerpt of the table in order to get an understanding of the results. In addition, the output also provides us with the output, `Apparent Circuit Structure: Circuit 16 with structure ((((1*5)*4)*3)*2)`, the serial structure $(((C_1 \otimes C_5) \otimes C_4) \otimes C_3) \otimes C_2$. `Circuit 16` has a computed reliability of 0.776887, which is the closest to the observed reliability, 0.657.

Table 3.1: The five-component data (Anderson-Cook, 2009) including the posterior estimates of the system and component reliabilities (Lu, 2019).

Data Type	Number of Successes	Sample Size	Posterior mean
System	17	25	0.657
Component 1	111	120	0.918
Component 2	96	100	0.951
Component 3	76	80	0.939
Component 4	98	100	0.971
Component 5	79	80	0.976

Table 3.2: R-output for `apparentCircuitMLE(0.657,c(0.918,0.951,0.939,0.971, 0.976),C5,TRUE)`.

Circuit	Structure	Reliability
Circuit 1	((((1+5)+4)+3)+2)	0.9999998
Circuit 2	((((1*5)+4)+3)+2)	0.999991
Circuit 3	((((1+5)*4)+3)+2)	0.9999076
Circuit 4	((((1*5)*4)+3)+2)	0.9996114
Circuit 5	((((1+5)+4)*3)+2)	0.9970084
Circuit 6	((((1*5)+4)*3)+2)	0.9968722
Circuit 7	((((1+5)*4)*3)+2)	0.9955888
Circuit 8	((((1*5)*4)*3)+2)	0.9910289
Circuit 9	((((1+5)+4)+3)*2)	0.9509967
Circuit 10	((((1*5)+4)+3)*2)	0.950825
Circuit 11	((((1+5)*4)+3)*2)	0.9492068
Circuit 12	((((1*5)*4)+3)*2)	0.9434577
Circuit 13	((((1+5)+4)*3)*2)	0.892938
Circuit 14	((((1*5)+4)*3)*2)	0.8902949
Circuit 15	((((1+5)*4)*3)*2)	0.8653859
Circuit 16	((((1*5)*4)*3)*2)	0.776887
Circuit 17	(((1+5)+4)+(2*3))	0.9999939
Circuit 18	(((1*5)+4)+(2*3))	0.9996772
Circuit 19	(((1+5)*4)+(2*3))	0.9966922
Circuit 20	(((1*5)*4)+(2*3))	0.986087

Next, looking back at the data from Table 3.1, we will use the number of successes and the sample sizes of both the system as a whole and each individual component to pass into our `apparentCircuitMLC` function in order to compute Y^* and Y_k^*. The function `apparentCircuitMLC(17,25,c(111,96,76, 98,79),c(120,100,80,100,80),C5)` reports the output pictured in Figure 3.1. From this output, we can see that the most likely circuit structure is `Circuit 16` due to the circuit structure being the most likely for the largest proportion of time, `0.9379`.

Thus, because `Circuit 16` gives the apparent circuit structure given the maximum likelihood analysis and is considered the most likely circuit given 10,000 simulations using the observed number of successes, sample sizes, and computed reliabilities, we can conclude that `Circuit 16` is the most likely circuit structure. The circuit structure for `Circuit 16` can be viewed in Figure 3.2.

3.2 Discussion

In this thesis, we compared the observed component reliabilities with the observed circuit reliabilities in order to assess what the structure of the circuit looks like given the reliability data. We expanded upon Anderson-Cook's numerical comparison method (Anderson-Cook, 2009) through a maximum likelihood analysis in addition to a simulation of new data using the observed data. We used Lu's (2019) Bayesian methods in order to compute the posterior means, or reliabilities, which we then used to simulate 10,000 maximum likelihood analyses, all to find the most likely circuit structure.

This maximum likelihood analysis is only appropriate if the individual components are independent (Anderson-Cook, 2009). However, our analysis does not take into account that dependence between components may exist in some instances. This dependence can cause

```
 [1] "Proportion of time Circuit 2 is the most likely structure: 1e-04"
 [2] "Proportion of time Circuit 9 is the most likely structure: 1e-04"
 [3] "Proportion of time Circuit 13 is the most likely structure: 9e-04"
 [4] "Proportion of time Circuit 14 is the most likely structure: 8e-04"
 [5] "Proportion of time Circuit 15 is the most likely structure: 0.0027"
 [6] "Proportion of time Circuit 16 is the most likely structure: 0.9379"
 [7] "Proportion of time Circuit 24 is the most likely structure: 0.0017"
 [8] "Proportion of time Circuit 33 is the most likely structure: 1e-04"
 [9] "Proportion of time Circuit 34 is the most likely structure: 4e-04"
[10] "Proportion of time Circuit 35 is the most likely structure: 4e-04"
[11] "Proportion of time Circuit 36 is the most likely structure: 0.0042"
[12] "Proportion of time Circuit 42 is the most likely structure: 4e-04"
[13] "Proportion of time Circuit 44 is the most likely structure: 5e-04"
[14] "Proportion of time Circuit 52 is the most likely structure: 0.004"
[15] "Proportion of time Circuit 56 is the most likely structure: 2e-04"
[16] "Proportion of time Circuit 65 is the most likely structure: 1e-04"
[17] "Proportion of time Circuit 67 is the most likely structure: 3e-04"
[18] "Proportion of time Circuit 68 is the most likely structure: 4e-04"
[19] "Proportion of time Circuit 69 is the most likely structure: 0.0016"
[20] "Proportion of time Circuit 70 is the most likely structure: 0.0088"
[21] "Proportion of time Circuit 77 is the most likely structure: 5e-04"
[22] "Proportion of time Circuit 78 is the most likely structure: 3e-04"
[23] "Proportion of time Circuit 88 is the most likely structure: 3e-04"
[24] "Proportion of time Circuit 89 is the most likely structure: 0.0036"
[25] "Proportion of time Circuit 90 is the most likely structure: 7e-04"
[26] "Proportion of time Circuit 95 is the most likely structure: 5e-04"
[27] "Proportion of time Circuit 98 is the most likely structure: 1e-04"
[28] "Proportion of time Circuit 103 is the most likely structure: 2e-04"
[29] "Proportion of time Circuit 116 is the most likely structure: 1e-04"
[30] "Proportion of time Circuit 118 is the most likely structure: 0.0062"
[31] "Proportion of time Circuit 122 is the most likely structure: 3e-04"
[32] "Proportion of time Circuit 127 is the most likely structure: 1e-04"
[33] "Proportion of time Circuit 141 is the most likely structure: 0.0018"
[34] "Proportion of time Circuit 142 is the most likely structure: 3e-04"
[35] "Proportion of time Circuit 153 is the most likely structure: 5e-04"
[36] "Proportion of time Circuit 154 is the most likely structure: 0.0179"
[37] "Proportion of time Circuit 164 is the most likely structure: 6e-04"
[38] "Proportion of time Circuit 172 is the most likely structure: 4e-04"
[1] "All Circuits not listed above are never the most likely Circuit."
[1] "The most likely circuit structure overall is Circuit 16"
```

Figure 3.1: R-output for `apparentCircuitMLC(17,25,c(111,96,76,98,79),c(120,100,80,100,80),C5)`.

Figure 3.2: Circuit 16, $(((C_1 \otimes C_5) \otimes C_4) \otimes C_3) \otimes C_2$.

components to imitate an incorrect circuit structure. Thus, further examination is necessary in order to account for the possible dependence between components.

Full-system tests are expensive, so finding an inexpensive way to check whether a system will function successfully is beneficial for all fields of professions that use systems. In addition to checking the success of the system, we can also check to see which system essentially has the highest success rate, given the reliability of the components of the system.

REFERENCES

Anderson-Cook, C. M. (2009). Evaluating the series or parallel structure assumption for system reliability. *Quality Engineering*, *21*(1), 88–95. https://doi.org/10.1080/08982110802317380

Gelfand, A. E., and A. F. M. Smith. (1990). Sampling-based approaches to calculating marginal densities. Journal of the American Statistical Association 8: 3–4.

Gelman, A., J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. 2013. *Bayesian data analysis*, 3rd ed. New York: Chapman and Hall/CRC.

Geman, S., and D. Geman. 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-6(6):721–41. https://doi.org/10.1109/TPAMI.1984.4767596

Hamada, M. S., G. W. Alyson, C. S. Reese, and H. F. Martz. 2008. *Bayesian reliability.* New York: Springer.

Lu, L. (2019). Bayesian evaluation of system structure for reliability assessment. *Quality Engineering*, *31*(4), 581–595. https://doi.org/10.1080/08982112.2019.1572901

Plummer, M. 2003. JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling. *Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003)* March 20–22, Vienna, Austria. ISSN 1609-395X.

R Development Core Team. 2016. *R: A language and environment for statistical computing.* Vienna, Austria: R Foundation for Statistical Computing. Available at: http://www.R-project.org.

APPENDIX

R-CODE

```
#
# Code written by Dr. Alan Polansky and Shannon Harms
#

set.seed(10211995)
#Use the set.seed in order to get the same output in the paper.
#Remove it in order for the function to work as intended.
decimalToBinary <- function(x,bits) {
  binaryNumber <- vector(mode="double", length=bits)
  k <- 0
  while(x>0) {
    k <- k + 1
    binaryNumber[bits-k+1] <- x%%2
    x <- x%/%2
  }
  return(binaryNumber)
}

aRule <- function(p,q) p+q-p*q
mRule <- function(p,q) p*q
addComponent <- function(circuit=NULL) {
#
# If circuit is NULL then create a circuit with a single component
#
  if(is.null(circuit)) {
    circuit <- list("Circuit 1"=c("1"), "Formula 1"=c("p[1]"))
    attr(circuit,"components") <- 1
    attr(circuit,"circuits") <- 1
    return(circuit)
  }
  numberOfComponents <- attr(circuit,"components")
  numberOfCircuits <- attr(circuit,"circuits")
  newNumberOfComponents <- numberOfComponents+1
  newCircuit <- list()
  attr(newCircuit,"components") <- newNumberOfComponents
  newComponent <- as.character(as.integer(newNumberOfComponents))
  newComponentFormula <- paste0("p[",as.character(as.integer(
  newNumberOfComponents)),"]")
  m <- 0
  for(k in 1:numberOfComponents) {
    replaceComponent <- as.character(k)
    replaceComponentFormula <- paste0("p[",k,"]")
    for(l in 1:numberOfCircuits) {
      m <- m + 1
      name <- paste("Circuit",m)
      fname <- paste("Formula",m)
      replaceCircuit <- paste0("(",as.character(k),"+",newComponent,")")
      replaceFormula <- paste0("aRule(",replaceComponentFormula,",",
      newComponentFormula,")")
      currentCircuit <- circuit[[paste("Circuit",l)]]
      currentFormula <- circuit[[paste("Formula",l)]]
      mcircuit <- gsub(as.character(k),replaceCircuit,currentCircuit)
      mFormula <- gsub(replaceComponentFormula,replaceFormula,currentFormula,
      fixed=TRUE)
      newCircuit[[name]] <- mcircuit
      newCircuit[[fname]] <- mFormula
      m <- m + 1
```

```
      name <- paste("Circuit",m)
      fname <- paste("Formula",m)
      replaceCircuit <- paste0("(",as.character(k),"*",newComponent,")")
      replaceFormula <- paste0("mRule(",replaceComponentFormula,",",
      newComponentFormula,")")
      mcircuit <- gsub(as.character(k),replaceCircuit,currentCircuit)
      mFormula <- gsub(replaceComponentFormula,replaceFormula,currentFormula,
      fixed=TRUE)

      newCircuit[[name]] <- mcircuit
      newCircuit[[fname]] <- mFormula
          }
  }
  attr(newCircuit,"circuits") <- m
#
# equivalent circuit detection
#
  numberOfSwitches <- 2^newNumberOfComponents
  binaryMatrix <- matrix(0,numberOfSwitches,newNumberOfComponents)
  for(k in 0:(numberOfSwitches-1))
      binaryMatrix[k+1,] <- decimalToBinary(k, bits = newNumberOfComponents)
  print(binaryMatrix)
#
# Build the switch matrix
#
  switchMatrix <- matrix(0,m,numberOfSwitches)
  for(k in 1:m)
    for(l in 1:numberOfSwitches) {
      p <- binaryMatrix[l,]
      name <- paste('Formula',k)
      switchMatrix[k,l] <- eval(parse(text=newCircuit[[name]]))
    }
  print(switchMatrix)
#
# Circuit Witch Comparisons
#
  keepCircuits <- vector(mode="integer", length=m)
  currentPosition <- 0
  for(k in 1:m) {
    if(k==1) {
      currentPosition <- currentPosition + 1
      keepCircuits[currentPosition] <- k
    }
    else {
      findEqual <- 0
      for(l in 1:(k-1)) {
        if(all(switchMatrix[k,]==switchMatrix[l,])) findEqual <- 1
      }
      if(findEqual == 0) {
        currentPosition <- currentPosition + 1
        keepCircuits[currentPosition] <- k
      }
    }
  }
  cat("keepCircuits = ",keepCircuits,"\n")
  keepCircuits <- keepCircuits[1:currentPosition]
  cat("keepCircuits = ",keepCircuits,"\n")
```

```
#
# Build Trimmed Circuit
#
  trimmedCircuit <- list()
  for(k in 1:currentPosition) {
    newCircuitName <- paste('Circuit',k)
    newFormulaName <- paste('Formula',k)
    circuitName <- paste('Circuit',keepCircuits[k])
    formulaName <- paste('Formula',keepCircuits[k])
    trimmedCircuit[[newCircuitName]] <- newCircuit[[circuitName]]
    trimmedCircuit[[newFormulaName]] <- newCircuit[[formulaName]]
  }

#
# Return Final Result
#
  attr(trimmedCircuit,"components") <- newNumberOfComponents
  attr(trimmedCircuit,"circuits") <- currentPosition
  return(trimmedCircuit)
}

C1 <- addComponent()
print(C1)

C2 <- addComponent(C1)
print(C2)

C3 <- addComponent(C2)
print(C3)

C4 <- addComponent(C3)
print(C4)

C5 <- addComponent(C4)
print(C5)

#
# Maximum Likelihood Analysis
#

apparentCircuitMLE <- function(pCircuit,pComponent,C,verbose=FALSE) {
  components = attr(C,"components")
  circuits = attr(C,"circuits")
  if(components != length(pComponent))
    stop("Incorrect Number of Components")
  if(length(pCircuit)!=1)
    stop("Length of pCircuit is not 1")

  pCircuitComponent <- vector(mode = "numeric", length = circuits)
  p = pComponent

  for(k in 1:circuits) {
    formulak = paste('Formula',k)
    pCircuitComponent[k] = eval(parse(text=C[[formulak]]))
  }
  apparentMLE <- which.min(abs(pCircuitComponent-pCircuit))
  if(verbose) {
```

```
    cat("Circuit Reliability: ", pCircuit,"\n\n")
    cat("Component Reliabilities: ", pComponent,"\n\n")
    cat("Independent Circuit Reliabilities: \n")
    for(k in 1:circuits) {
      circuitName <- paste('Circuit',k)
      cat(circuitName," ",C[[circuitName]]," ",pCircuitComponent[k],"\n")
    }
    apparentName <- paste('Circuit',apparentMLE)
    cat("\nApparent Circuit Structure: ",apparentName)
    cat(" with structure ",C[[apparentName]],"\n\n")
  }
  return(list(apparentMLE,pCircuitComponent))
}


### Example A ###
apparentCircuitMLE(0.500,c(0.100,0.200,0.300),C3,TRUE)

### Lu Lu's Example ###
apparentCircuitMLE(0.657,c(0.918,0.951,0.939,0.971,0.976),C5,TRUE)

#
# MLC
#

apparentCircuitMLC <- function(Y,n,Ycomponents,nComponents,C)
{
components = attr(C,"components")
circuits = attr(C,"circuits")
if(length(Ycomponents) != length(nComponents))
    stop("K's are not equal for the Yk Components")
if(components != length(Ycomponents))
    stop("Incorrect Number of Components")

m <- length(Ycomponents)
Yk <- Ycomponents
nk <- nComponents

phat <- (1/n)*Y
pkhat <- (1/nk)*Yk

if(phat > 1 | phat < 0)
    stop("Probability of phat has to be between 0 and 1")
if(any(pkhat > 1 | pkhat < 0))
    stop("Probability of pkhat has to be between 0 and 1")

b <- 10000
mlc <- cbind(rep(0,10))

for(l in 1:b)
{
  repeat {
    newY <- rbinom(1,n,phat)
    newYk <- rbinom(1*length(nk),nk,pkhat)
    newphat <- (1/n)*newY
    newpkhat <- (1/nk)*newYk
```

```
    if(newphat > 1 | newphat < 0) {
      stop("Probability of newphat has to be between 0 and 1")}
    if(any(newpkhat > 1 | newpkhat < 0)) {
      stop("Probability newpkhat has to be between 0 and 1")
    }

    mle <- apparentCircuitMLE(newphat,newpkhat,C,FALSE)
    mlc[l] <- mle[[1]]
    l <- l+1
    if(l > b) {
      break
    }
  }


  ptotal <- table(mlc)/b
  mostlikely <- names(which.max(ptotal))
  print(paste("Proportion of time Circuit",names(ptotal),
  "is the most likely structure:",ptotal))
  print(paste("All Circuits not listed above are never the most
  likely Circuit."))
  print(paste("The most likely circuit structure overall is Circuit",
  mostlikely))
  cat("Vector of MLC Circuit Structures:","\n\n")
  return(mlc)
}
}

### Example A ###
apparentCircuitMLC(50,100,c(10,20,30),c(100,100,100),C3)

### Lu Lu's Example ###
apparentCircuitMLC(17,25,c(111,96,76,98,79),c(120,100,80,100,80),C5)
```

(Polansky & Harms, 2019)