Université d'Ottawa · University of Ottawa

Université d'Ottawa · University of Ottawa

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Zhi WANG

AUTEUR DE LA THÈSE - AUTHOR OF THESIS

M. Sc. (Systems Science)

GRADE - DEGREE

Systems Science Program

FACULTÉ, ÉCOLE, DÉPARTEMENT - FACULTY, SCHOOL, DEPARTMENT

TITRE DE LA THÈSE - TITLE OF THE THESIS

On-Line Algorithms for Attribute Sampling

M. Alvo

DIRECTEUR DE LA THÈSE - THESIS SUPERVISOR

CO-DIRECTEUR DE LA THÈSE - THESIS CO-SUPERVISOR

EXAMINATEURS DE LA THÈSE - THESIS EXAMINERS

A. Dabrowski

M. Zarepour

J.-M. De Koninck, Ph.D.

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

DEAN OF THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

ON-LINE ALGORITHMS FOR ATTRIBUTE SAMPLING

By
Zhi WANG
October 2004

A Thesis
submitted to Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements
for the degree of
Master of Science in Systems Science

Your file Votre référence
ISBN: 0-494-01636-1
Our file Notre référence
ISBN: 0-494-01636-1

Abstract

Acceptance sampling is an important field of statistical quality control. This thesis proposed online algorithms to construct hypergeometric sampling plans using Chebyshev polynomials. The recurrence relationship of Chebyshev polynomials significantly reduces the combinatorial computations for hypergeometric distributions. This property can be extended to compute binomial probabilities.

The online tables generated by the proposed algorithms are easy to use and precise compared with the Odeh and Owen (1983)'s table. One can readily check the required sample size and acceptance number for pre-specified producer's risk, consumer's risk, proportion defective, and/or lot size for hypergeometric and binomial samplings. The algorithms also cover the confidence limits for hypergeometric and OC curve for binomial sampling. The programs written in R language are appended to show the implementation for the algorithms proposed.

Acknowledgements

I am grateful to Professor Mayer Alvo who guided me in the preparation of the thesis. Our weekly meetings motivated me to work on schedule. Through these meetings, we discussed the problems that I have encountered in my progress and his valuable guidance led to the completion of this thesis. I also took some statistical courses to equip my statistical background under Professor Alvo's suggestion. I am sincerely thankful for such experiences.

The Systems Science program gave me the opportunity to work on the thesis and to gain all the academic background in the period of study. For this I am grateful.

Contents

Abstract ii

Acknowledgements iii

1 Introduction 1

2 Background 3

2.1 Terminology 4

2.2 Notations 5

2.3 Lot Acceptance Sampling 6

2.4 Lot Acceptance Sampling Plans 6

2.4.1 Single sampling plans 7

2.4.2 Double sampling plans 7

2.4.3 Multiple sampling plans 8

2.4.4 Sequential sampling plans 9

2.5 The Probability of Acceptance 11

2.5.1 Hypergeometric distribution 12

2.5.2 Binomial distribution 13
2.5.3 Double sampling plan 13
2.6 Critical Quality Levels 14
2.6.1 Lot Tolerance Percentage Defective (LTPD) 14
2.6.2 Acceptable Quality Level (AQL) 15
2.6.3 Equilibrium Quality Level 15
2.7 Choosing a Single Sampling Plan 16
2.7.1 Two-point method 16
2.7.2 ISO2859/Military Standard 105D 18
2.7.3 Dodge and Romig's system of LTPD plans 19
2.7.4 Dodge and Romig's system of AOQL plans 19
2.8 Summary 20

3 Literature Review **21**

4 Chebyshev Polynomials **23**

4.1 Chebyshev Polynomials 23
4.2 Hypergeometric Distribution by Chebyshev Polynomials 24
4.3 Binomial Distribution by Chebyshev Polynomials 25

5 Algorithms **26**

5.1 On-line Algorithms for Hypergeometric Distribution 27
5.1.1 Mathematical model 27
5.1.2 On-line algorithms for Minitab macro 29

5.1.3 On-line algorithms for programs written in R 31
5.2 On-line Algorithms for Binomial Distribution 36
5.2.1 Mathematical model 36
5.2.2 On-line algorithms for programs written in R 37

6 Data Analysis **41**
6.1 Precision 42
6.1.1 Hypergeometric sampling 42
6.1.2 Binomial sampling 51
6.1.3 Summary 54
6.2 Execution Time 54

7 Conclusions and Recommendation for Future Study **57**

Bibliography **60**

Appendix **61**

A Minitab Macro **61**
A.1 Compute producer's risk α 61
A.2 Compute consumer's risk β 63

B R Code for Hypergeometric Sampling **65**
B.1 Compute (n, c) given $(\alpha, \beta, k_1, k_2, N)$ 65
B.2 Compute c given $(\alpha, \beta, k_1, k_2, n, N)$ 70

B.3 Compute upper and lower confidence limits 76

C R Code for Binomial Sampling **80**

C.1 Compute (n, c) given $(\alpha, \beta, p_1, p_2)$ 80

C.2 Draw OC curve given (n, c) 86

Chapter 1

Introduction

Acceptance sampling is an important field of statistical quality control which aims at continuous process and quality improvement. According to the definition from the ASQ standard (American Society for Quality) (2004), acceptance sampling is "inspection of a sample from a lot to decide whether to accept that lot" and an acceptance sampling plan is "a specific plan that indicates the sampling sizes and associated acceptance or non-acceptance criteria to be used".

Some of the reasons to use acceptance sampling are the high cost and/or destructive testing for 100% inspection and the possibilities for disastrously poor quality for no inspection at all. The basic concept behind acceptance sampling is that the samples randomly drawn from the lot are used to "sentence" the lot, which means to accept or reject the lot.

Acceptance sampling is a procedure to decide the disposition of the lot. The procedure can be simplified as (1) draw the sample, (2) inspect the sample, and (3) sentence the lot. When following the procedure to do inspection, one uses the scheme, called the acceptance sampling plan, to decide whether to accept or reject the lot. A sampling plan normally includes the parameters of sample size, acceptance number, proportional defective, and lot size. The detailed explanations for these terms are given in section (2.1).

When a lot is submitted for inspection, a sampling plan is used to decide how many samples should be drawn from the lot and how many defective items found in the sample are considered to be acceptable. Since the sample is not free of defectives, there are probabilities of accepting a lot which is actually not acceptable and of

rejecting one which is acceptable. We call these probabilities risks.

Risks follow different probability distributions. In this thesis, we compute two of them, one is hypergeometric and another is binomial distribution. Due to the combinatorial terms in both distributions, the computation becomes very cumbersome as the lot size becomes large.

There are normally two computational methods, one is exact computation and another is approximation to compute hypergeometric distribution. Exact computation can be performed by using the recursive relationship of hypergeometric distribution and/or the simplification of factorial terms under an appropriate algorithm on the computer. It can also be obtained by using a table, such as the one by Odeh and Owen (1983), with interpolations. An approximation utilizes the fact that the hypergeometric distribution can be approximated by a binomial distribution when the sample size is small comparing to the lot size.

In this thesis, an algorithm is proposed to generate a table online which displays the values of sample size and acceptance number given the values of proportional defective, associated risks and lot size. In this table, one can readily check the values for sample size and acceptance number according to every possible group of values of risks, proportional defective, and lot size. No interpolation is required anymore. In addition, the one-sided and two-sided confidence limits for the number of defectives in the submitted lot are supplied when the observed number of defectives, sample size and lot size are given.

The proposed algorithm can also be applied to compute the probabilities of the binomial distribution. The same sampling system, which computes sample size and acceptance number according to the values of risks and proportional defectives, will also be provided for the binomial case.

Chapter 2

Background

In constructing acceptance sampling plans, one usually should give answers for the following questions:

- What kind of sampling plan one should pick? - Single, Double, Multiple, or Sequential Sampling Plans;
- What kind of probability distribution the sampling procedure follows? - Hypergeometric or Binomial;
- What levels of quality have the consumer and/or producer agreed upon? - LTPD, AQL, Equilibrium Quality Level;
- What levels of risks do the producer and consumer expect? - Producer's risk and Consumer's risk;
- How many samples should one draw and how many numbers of defectives are considered to be acceptable for the lot? - Choosing a sampling plan (n, c).

In this chapter we provide a brief description of various sampling inspection schemes. For completeness, it includes:

- Terminology;
- Notations;
- Lot Acceptance Sampling;

- Lot Acceptance Sampling Plans (LASP);
- The Probability of Acceptance;
- Critical Quality Levels;
- Choosing a Single Sampling Plan;
- Summary.

2.1 Terminology

In a quality control regime, a **defect** is defined as any non-conformance with the specified requirements. A **defective item** or simply **defective** usually refers to an item which contains one or more defects. A detailed standard is always assumed to be available to qualify an inspected item as either acceptable or defective. The **proportion defective**, or *quality level* in acceptance sampling, means the number of defective items divided by the total number of items in the lot.

In terms of acceptance sampling, an **inspection lot** is a group of items on which the terminal decision of acceptance sampling is based. It bears the same meaning as what was mentioned before that a whole lot is accepted or rejected. Although it is not compulsorily required that the inspection lot is from the same production condition, e.g. shift, machine, material lot, it is strongly recommended that the inspection lot be as homogeneous as possible, especially for continuous production.

The term **sample** is always taken to mean a group of items selected individually at random from the lot without replacement.

The key point for sampling plan is that one can decide **the probability of acceptance** according to pre-specified proportion defective before the sampling inspection is taken place. The essence of quality control is to manipulate the whole process including all-aspects control of incoming materials, production process, and the final products. To produce the qualified final products, it is important that the incoming materials are satisfied and the process is stable enough. One of the most economic ways to achieve this goal is sampling inspection. Sampling inspection is normally applied in the following situations:

- When testing is destructive;

- When inspection costs are extremely high;
- When time or technology limitations are constraints;
- When lot sizes are very large and the probability of inspection errors is high;
- When supplier's quality history is good enough to justify less than 100% inspection;
- When potential liability risks are high enough to warrant some form of continuous monitoring.

The term **Producer** has the same meaning as everyday language, as well as the term **Consumer**. The *Producer* is the one who sells the products to whom buys them, the *Consumer*. Therefore, the words like seller, supplier, vendor, or manufacturer all come to mean Producer and buyer, user, or vendee means Consumer. Sometimes internal inspection happens for which the consumer and the producer come from the same firm. In this case, the actual cash flow may not happen but the sampling inspection is still in place to control the quality. It can be either done by the first production unit for the outgoing products or by the second one for incoming materials or both.

Analogous to not rejecting or rejecting the null hypothesis in a hypothesis test, attribute sampling plans are hypothesis tests and the conclusions are made towards accepting or rejecting a lot. There are two types of errors occurring in constructing sampling plans. Incorrectly rejecting a lot that is really acceptable is a *Type I error*. The risk of making a Type I error is called **Producer's Risk**, and is denoted by α. **Consumer's Risk**, usually denoted by β, is the risk of making a *Type II error*, which is incorrectly accepting a lot that is really unacceptable. It should be remembered that both of the producer's and consumer's risks are conditional probabilities, the real risks depend on the quality distribution of lots submitted for inspection.

2.2 Notations

Throughout this paper, the following notations are used:

- N - Lot size (the number of items in the lot submitted for inspection);
- n - Sample size (the number of items inspected);

- c - Acceptance number (the number of defective units allowed in a lot which is accepted);
- k_1 - The number of defective units in a satisfactory quality level;
- p_1 - Proportion defective, where $p_1 = k_1/N$;
- k_2 - The number of defective units in an unsatisfactory quality level;
- p_2 - Proportion defective, where $p_2 = k_2/N$;
- α - Producer's risk, which is the probability of rejecting a lot with proportion defective p_1. This is a conditional probability. In mathematical terms, it can be expressed as: $Pr(X > c|p = p_1) = \alpha$;
- β - Consumer's risk, which is the probability of accepting a lot with proportion defective p_2. It is also a conditional probability and can be expressed as $Pr(X \leq c|p = p_2) = \beta$ in mathematical terms.

2.3 Lot Acceptance Sampling

Sampling inspection can be either lot-by-lot or continuous. In lot-by-lot inspection, if the quality for inspected item is classified as defective or non-defective, we then call it lot-by-lot sampling inspection by attributes or *Sampling by Attributes*. If the item inspection leads to a continuous measurement, then we are *Sampling by Variables*.

In **Attribute Sampling**, the product is divided into inspection lots. A sample or several samples are then drawn at random from each lot. According to sampling plans specified, one decides to accept, reject, or take further actions, such as draw more samples for inspections.

In this thesis, only the former one, which is *Sampling by Attributes*, will be discussed.

2.4 Lot Acceptance Sampling Plans

A **Lot Acceptance Sampling Plan (LASP)** is a sampling scheme and a set of rules for making decisions. It can be categorized as:

- Single sampling plan;
- Double sampling plan;
- Multiple sampling plan;
- Sequential sampling plan.

2.4.1 Single sampling plans

A Single Sampling Plan is characterized by three parameters (n, c, N). The decision is made according to the following rule:

1. Take a random sample of size n from a lot sample of size N,
2. If the number of defectives in the sample is less than or equal to the acceptance number c, accept the whole lot, otherwise, reject the lot.

A single sampling plan is often called an (n, c) plan, which focuses on finding appropriate values for sample size n and acceptance number c. It is the easiest sampling plan in practice. The disadvantage for this plan is that the sample size may be larger than actually required, especially for the lots whose quality levels sit at two extremes, very good or very bad. Furthermore, a single sampling plan does not offer a second chance for doubtful lots. To compensate for this drawback, we have the double/multiple/sequential sampling plans as below.

2.4.2 Double sampling plans

After the first sample is tested, there are three possibilities: (1) accept the lot; (2) reject the lot; (3) no decision. If the result is (3), a second sample will be taken for further testing. This is called a *double sampling plan*. The terminal decision will be based on criteria pre-specified for each step. For example, an initial random sample of size n_1 is taken from the lot. The number of defectives d_1 is counted and then compared with the first sample's acceptance number c_1 and rejection number r_1. The following criteria apply:

- If $d_1 \leq c_1$, the lot is accepted;

- If $d_1 \geq r_1$, the lot is rejected;
- If $c_1 < d_1 < r_1$, a second sample is taken.

If a second sample of size n_2 is taken, the number of defectives d_2 is counted. The total number of defectives is then $D_2 = d_1 + d_2$. The acceptance number c_2 and rejection number r_2 of sample 2 are compared with D_2:

- If $D_2 \leq c_2$, the lot is accepted;
- If $D_2 \geq r_2$, the lot is rejected.

In double sampling plans, $r_2 = c_2 + 1$ to ensure a decision on the sample.

It is obvious that double sampling plans outperform single sampling plans if the incoming quality justifies a clear decision for a smaller sample size $n_1 (n_1 < n)$. So on average, the total number of inspections in double sampling plans would be less than the one required in single sampling plans. On the other hand, double sampling plans place more challenges on the inspectors. It is also possible that the maximum amount of inspections for a given lot exceeds that for the single sampling plans.

2.4.3 Multiple sampling plans

This is an extension of double sampling plans. It involves inspection of 1 to k successive samples as required to reach a terminal decision. Similar to double sampling plans, the advantage of multiple sampling is smaller sample sizes compared with the single sampling plan. But it imposes complicated inspection procedures for the inspectors. The number of samples could also be quite large if sampling is continuously made before reaching a terminal decision.

At stage 1, a sample of size n_1 is taken. The number of defectives d_1 is compared to acceptance number c_1 and rejection number r_1. The following rules apply:

- If $d_1 \leq c_1$, the lot is accepted;
- If $d_1 \geq r_1$, the lot is rejected;
- If $c_1 < d_1 < r_1$, the second sample is taken.

As in double sampling plans, if a second sample is required, the total number of defectives found in stages 1 and 2, say $D_2 = d_1 + d_2$, will be compared with acceptance number c_2 and rejection number r_2. Then if $c_2 < D_2 < r_2$, the third sample will be taken for inspection. The procedure continues until a terminal decision is made. Multiple sampling plans may sometimes not allow an acceptance decision in the early stages, but instead rejection may occur. When using multiple sampling plans, a parameter named Average Sample Number (ASN) is provided for measuring its efficiencies given the levels of consumer and producer's risks.

2.4.4 Sequential sampling plans

Sequential sampling plans are the most efficient type of sampling plans in terms of the power to discriminate between different quality levels at a given cost. They are valid and less costly alternatives to the military standard type of sampling plan. It is different from single, double or multiple sampling plans.

One takes a sequence of samples from a lot and the total number of samples taken is decided by a function of the results of the sampling process. The sampling process could be taken one sample at a time, called *item-by-item sequential sampling* or a group of samples for one time, named *group sequential sampling*. The first case is more popular in practice and will be described in detail here.

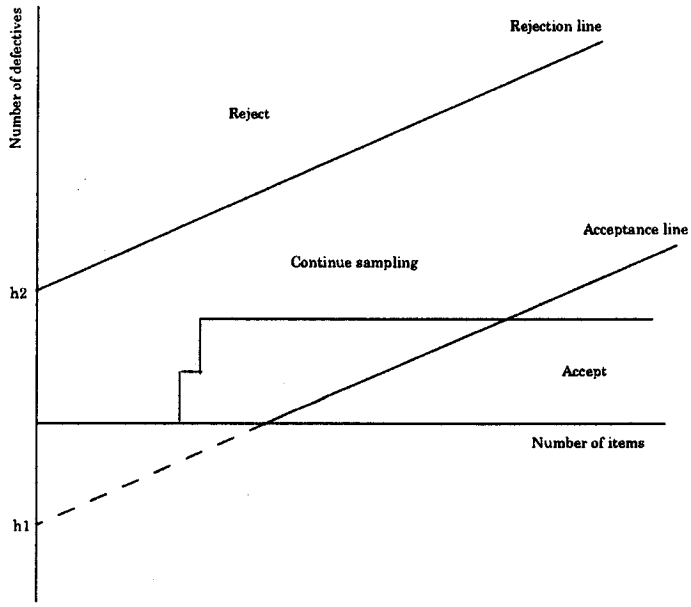

Figure 1: *Sequential Sampling Plan.*

The cumulative number of items selected and defectives are plotted on Figure 1. If the plotted point falls within the parallel lines the process continues by drawing another sample. Once a point falls above the upper line, the lot is rejected; if it falls on or below the lower line, the lot is accepted. Otherwise the sampling continues until the lot is 100% inspected in theory. However, in practice, to control the total number of inspected items in sequential sampling, the actual total number of items sampled should not exceed three times the number that would be used in a single sampling plan.

For the two limit lines shown in the graph, the following parameters apply:

Acceptance line: $x_a = -h_1 + sn$

Rejection line: $x_b = h_2 + sn$

Where: $h_1 = (\log \frac{1-\alpha}{\beta})/k$, $h_2 = (\log \frac{1-\beta}{\alpha})/k$, $k = \log \frac{p_2(1-p_1)}{p_1(1-p_2)}$, $s = (\log \frac{1-p_1}{1-p_2})/k$

For the meaning of α, β, p_1, p_2, please see section (2.2). Details for these derivations are found in the book by Bowker and Lieberman (1959, p464).

Each sampling plan has its own pros and cons. The decision on which sampling

plan to pick highly depends on economic considerations, the complexity for different plans, and level of uncertainty for the total number of samples taken in a day-by-day production. In this thesis, we will discuss *single sampling plans* and briefly touch upon *double sampling plans.*

2.5 The Probability of Acceptance

The probability of accepting a lot (Y-axis) versus the proportional defective per lot (X-axis) is plotted on a curve, called the **Operating Characteristic** (OC) curve. It is the primary tool for displaying and investigating the properties of LASP (Lot Acceptance Sampling Plan). Odeh and Owen (1983, p2) gave the definitions for different types of OC curve used in acceptance sampling plan:

> When sampling is from an isolated lot or inference is to a single lot, the operating characteristic curve is based on the hypergeometric distribution and in quality control literature is called a 'Type A' OC curve. If there is a sequence of lots so that the sampling continues over an indefinite period of time, the binomial model is used in quality control and the resulting OC curve is referred to as 'Type B'.

A typical OC curve shapes like Figure 2:

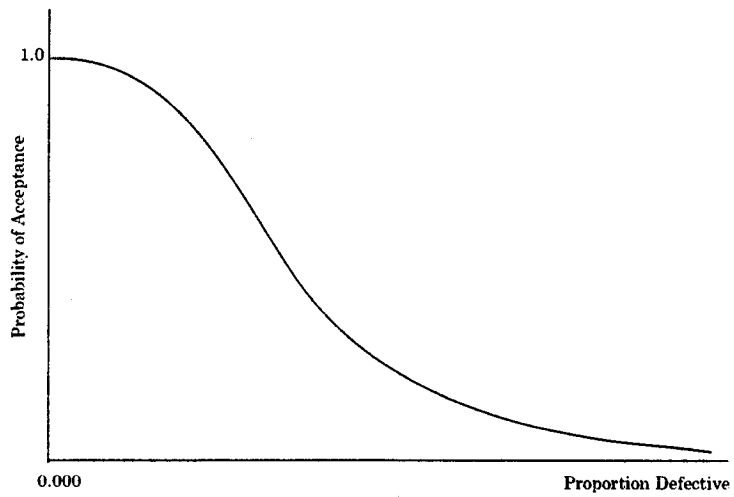

Figure 2: *Operating Characteristic Curve.*

The values of lot size N, sample size n and acceptance number c uniquely decide the shape of the *'Type A'* OC curve. For *'Type B'* OC curve, the characteristic of an OC curve is decided by the sample size n and acceptance number c since the interference due to the change of the lot size is negligible due to the small sample size.

As mentioned by Odeh and Owen (1983) in the table for attribute sampling plans, the *'Type A'* OC curve is based on the hypergeometric distribution and *'Type B'* OC curve is based on the binomial distribution. We consider these next.

2.5.1 Hypergeometric distribution

The hypergeometric probability of getting x defectives in a random sample of n items drawn without replacement from a lot of size N containing M defectives is:

$$h(x; n, M, N) = \frac{\binom{M}{x}\binom{N-M}{n-x}}{\binom{N}{n}} \tag{2.1}$$

where $max(0, n - N + M) \leq x \leq min(n, M)$

To draw an OC curve, we have:

$$p_a = \sum_{x=0}^{c} \frac{\binom{M}{x}\binom{N-M}{n-x}}{\binom{N}{n}} \tag{2.2}$$

The value for M is known and expressed on the X-axis as proportional defective M/N. Varying the values of M will yield different values for the probability of acceptance (p_a) in equation (2.2) when the values of n, c, and N are fixed. Joining those points together will form the *'Type A'* OC curve. The curve shifts in correspondence with the changes of the values n, c, and N.

2.5.2 Binomial distribution

The hypergeometric distribution may be approximated by the binomial distribution when the sample size n is equal to or less than $0.1N$.

If the probability of drawing a defective item in one time sampling is denoted by p, the probability of getting x defectives in a sample of size n with replacement is:

$$b(x;n,p) = \binom{n}{x} p^x(1-p)^{n-x} = \frac{n!}{x!(n-x)!}p^x(1-p)^{n-x} \tag{2.3}$$

It is always assumed that (1) the probability of drawing a defective item is a constant p and the probability of drawing a non-defective item is $(1-p)$; (2) the drawings are independent; (3) the sampling continues for an indefinite period of time.

To draw the *'Type B'* OC curve, we have:

$$p_a = \sum_{x=0}^{c} \binom{n}{x} p^x(1-p)^{n-x} = \sum_{x=0}^{c} \frac{n!}{x!(n-x)!}p^x(1-p)^{n-x} \tag{2.4}$$

The proportion of defectives, denoted by p in equation (2.4), is plotted on the X-axis. According to equation (2.4), when we fix the values of sample size n and acceptance number c, each value of p corresponds to the value of the probability of acceptance (p_a) on the Y-axis. Connecting these points obtains the *'Type B'* OC curve. As p becomes larger, the probability of acceptance (p_a) gets smaller. The unique pair of sample size n and acceptance number c determine the shape of the OC curve. Increasing the sample size n or decreasing the acceptance number c will lead to the inward shift of the OC curve.

2.5.3 Double sampling plan

For double sampling plans, the OC curve is based on the sampling criteria (n_1, n_2, c_1, c_2). Using the same notations as in Section 2.4.2, one can draw a double sampling OC curve for the binomial distribution using the following equation:

$$
\begin{aligned}
p_a &= P(d_1 \leq c_1|p) + P(d_1 + d_2 \leq c_2|c_1 < d_1 \leq c_2, p) \\
&= \sum_{d_1=0}^{c_1} \binom{n_1}{d_1} p^{d_1}(1-p)^{n_1-d_1} \\
&+ \sum_{d_1=c_1+1}^{c_2} \left[\binom{n_1}{d_1} p^{d_1}(1-p)^{n_1-d_1} \sum_{d_2=0}^{c_2-d_1} \binom{n_2}{d_2} p^{d_2}(1-p)^{n_2-d_2} \right] \qquad (2.5)
\end{aligned}
$$

The values of the first sample size n_1, acceptance number c_1 and second sample size n_2, acceptance number c_2 uniquely determine the shape of the OC curve for the double sampling plan. For every quality level (p), there is an OC curve for binomial double sampling plans. Normally there are fixed relationships between n_1 and n_2, for example $n_1 = n_2$ or $n_2 = 2n_1$ to control the Average Sample Number (ASN).

2.6 Critical Quality Levels

According to Hald (1981, p21), "*A critical quality level is a fraction defective determined for inspection purpose from technological and economical considerations such that product or lot (whichever is applicable) of better quality should be accepted and product or lots of poorer quality rejected*".

Sampling plans are indexed by the following critical quality levels which impose a restriction on the selected sampling plan:

- Lot Tolerance Percentage Defective (LTPD);
- Acceptable Quality Level (AQL);
- Equilibrium Quality Level.

2.6.1 Lot Tolerance Percentage Defective (LTPD)

The definition of LTPD in the ASQ standard(American Society for Quality) (2004) is: "expressed in percentage defective, the poorest quality in an individual lot that should be accepted".

The LTPD is the highest proportion defective that is considered acceptable for a given lot. The consumer would like to have a low probability to accept a lot with a defective level as high as the LTPD. Put it another way, the probability of accepting the lot with the quality level of LTPD is called the consumer's risk, and denoted by β.

The choice for LTPD is based on "the use of the product, consequences for the consumer of getting bad quality products, market quality expectation, the producer's own quality performance, consequences of loss of goodwill and other special technological and economical factors" (Hald 1981, p22).

2.6.2 Acceptable Quality Level (AQL)

The definition of acceptable quality level in the ASQ standard(American Society for Quality) (2004) is: "in a continuing series of lots, a quality level that, for the purpose of sampling inspection, is the limit of satisfactory process average".

The AQL is the highest proportion defective that is considered acceptable as a long-run average for the process. Since AQL is an average, a particular proportion defective for a lot has nothing to do with accepting or rejecting the lot. The risk of rejecting a satisfactory quality is the producer's risk α. The producer would like to have a sampling plan which has high probability of accepting a lot with defect level less than or equal to the AQL.

AQL is chosen according to the technological requirements for the product, economical consideration towards inspection costs and the price for the product.

2.6.3 Equilibrium Quality Level

One of the most popular methods of constructing sampling plans in practice consists in choosing two quality levels, p_1 and p_2 ($p_1 < p_2$), two risks, α and β ($0 < \beta < 1-\alpha < 1$). Quality levels, or proportion defective, p_1 and p_2 correspond with two parameters above, AQL and LTPD.

The equilibrium quality level occurs when both of the following two equations are

satisfied:

$$\begin{aligned} Pr(X > c|p = p_1) &\leq \alpha \\ Pr(X \leq c|p = p_2) &\leq \beta \end{aligned} \tag{2.6}$$

Since the numbers of acceptance number c and sample size n are integers, one should choose a stronger plan to satisfy the inequalities above.

Consumers expect to receive lots free of defectives, which for producers may entail an unacceptable high cost. A sampling plan may be derived depending if it is from the consumer's or the producer's perspective. But either way, a good sampling plan always means to meet its primary purpose of rejecting the unsatisfactory lot and accepting the satisfactory one with reasonable risks.

In this thesis, we will use equilibrium quality levels to find the appropriate acceptance number c and sample size n to satisfy equation (2.6). Given pre-specified α, β, p_1, p_2, and N, one can obtain a single sampling plan for (n, c). Details will be given in Section (2.7).

2.7 Choosing a Single Sampling Plan

As previously defined, a single sampling plan is specified by the pair of numbers (n, c). There are various ways to find appropriate values for n and c; the most commonly used methods are:

- Two-point method (Sampling plans with given producer's and consumer's risk);
- ISO2859/Military Standard 105D;
- Dodge and Romig's system of LTPD plans;
- Dodge and Romig's system of AOQL plans;

2.7.1 Two-point method

This is a method to design a sampling plan which satisfies both consumer and producer's perspectives, that is to reach equilibrium quality levels. Given the percent

defective p_1, the producer would expect the probability of acceptance is at least $1-\alpha$. On the other hand, the consumer would expect the probability of acceptance is at most β given the percent defective p_2. Thus, one will need to find a unique sampling plan, defined by (n, c), which passes through these two points, as shown in Figure 3 below.

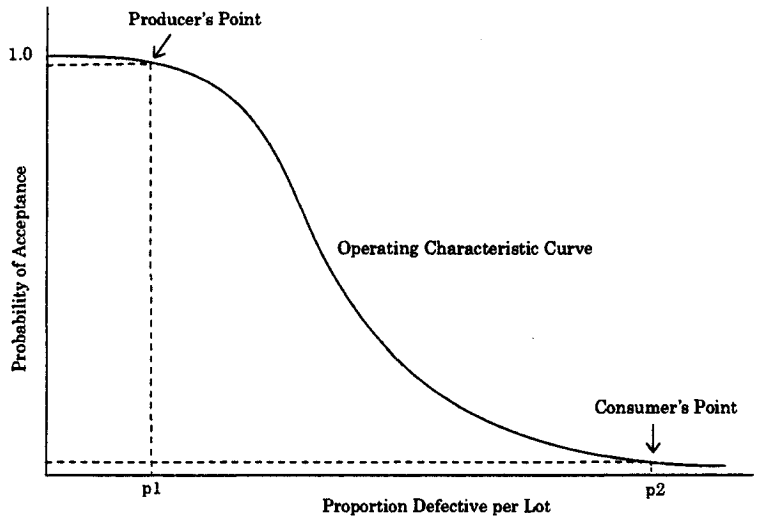

Figure 3: *Two-point Method.*

The producer's point is controlled to prevent rejecting satisfactory lots with proportional defective p_1 and the consumer's point is set up to prevent accepting unsatisfactory lots with proportional defective p_2. Typical choices for these points are: p_1 is the AQL; p_2 is the LTPD; producer's point is the producer's risk α (Type I error), and consumer's point is the consumer's risk β (Type II error).

If we assume that hypergeometric sampling is valid, then the sample size n and the acceptance number c can be solved by the equations (2.7) and (2.8):

$$P(X \leq c|k_1) = \sum_{x=0}^{c} \frac{\binom{k_1}{x}\binom{N-k_1}{n-x}}{\binom{N}{n}} = 1-\alpha \tag{2.7}$$

$$P(X \leq c|k_2) = \sum_{x=0}^{c} \frac{\binom{k_2}{x}\binom{N-k_2}{n-x}}{\binom{N}{n}} = \beta \tag{2.8}$$

where $k_1 < k_2$, $n \leq N, c \leq N, k_1 = p_1 \times N$, and $k_2 = p_2 \times N$.

If the binomial sampling is assumed to be appropriate, the sample size n and the acceptance number c are solved by:

$$\begin{aligned} P(X \leq c|p_1) &= \sum_{x=0}^{c} \binom{n}{x} p_1^x (1-p_1)^{n-x} \\ &= \sum_{x=0}^{c} \frac{n!}{x!(n-x)!} p_1^x (1-p_1)^{n-x} = 1-\alpha \\ P(X \leq c|p_2) &= \sum_{x=0}^{c} \binom{n}{x} p_2^x (1-p_2)^{n-x} \\ &= \sum_{x=0}^{c} \frac{n!}{x!(n-x)!} p_2^x (1-p_2)^{n-x} = \beta \end{aligned}$$

It is obvious that these two equations are nonlinear and therefore no simple, direct solution can be found. In this thesis, we will propose a computer algorithm to obtain the answers quickly and easily. There is no doubt that one can always get the solution from some existing tables, such as the one written by Odeh and Owen in 1983. But data in a book are limited. It cannot cover all the possible values for $(\alpha, \beta, k_1, k_2, N)$. Our on-line algorithm is superior in this regard and readily gives the solution for each set of $(\alpha, \beta, k_1, k_2, N)$. See Chapter 6 Data Analysis for details.

2.7.2 ISO2859/Military Standard 105D

This system is not based on an explicit mathematical model. It consists of a combination of a normal sampling plan, a tightened sampling plan, and a reduced sampling plan plus rules for switching from one to the other. Essentially this standard is used like a contract between a producer and a consumer. They both agree on an Average Quality Level (AQL). The producer is rewarded if he produces a reasonably low proportion of nonconforming units and punished if an unreasonably high proportion

of defective or nonconforming units are produced based on agreed-upon AQL. Cases above will lead to a switch from normal to reduced or normal to tightened sampling plan. In addition to AQL, one also needs to decide on inspection level, which determines the relationship between the lot size and the sample size.

Once decided on the AQL, the inspection level, and the lot size, one can check the table to find out the sample size code letter. According to the sample size code letter, one may start from normal inspection and enter the proper table to come up with appropriate sampling plans. The criteria for switching from normal to tightened or to reduced sampling plan or even to stop the inspection are specified on the table (NIST/SEMATECH 2000).

2.7.3 Dodge and Romig's system of LTPD plans

The Dodge-Romig (1929) system of LTPD plans fix the consumer's risk to 10 per cent. The sample size n and acceptance number c are decided by minimizing the average total amount of inspections for different levels of process average quality. For each LTPD level, there exist the paired values for n and c in accordance with the process average quality in percentage and the lot size.

2.7.4 Dodge and Romig's system of AOQL plans

According to the definition from ASQ (American Society for Quality) (2004), average outgoing quality limit is "the maximum average outgoing quality over all possible levels of incoming quality for a given acceptance sampling plan and disposal specification".

When sampling and inspection is non-destructive, a 100% inspection is conducted for rejected lots. All the bad units are then replaced by good ones. So the only defects left are those in the lot that was accepted. Average Outgoing Quality (AOQ) refers to the long term defect level for this combined LASP (Lot Acceptance Sampling Plan) and 100% inspection of rejected lots process. If the incoming quality is very good, then the Average Outgoing Quality (AOQ), defined as "the expected average quality level of outgoing product for a given value of incoming product quality" (ASQ 2004), is low. By the same rule, if the quality is bad and all the rejected items are replaced since a 100% inspection is conducted for the rejected lot, the AOQ is low too. The AOQ reaches its highest value, called AOQL to represent the worst possible quality

in which the rectifying program could result.

The Dodge and Romig's system of AOQL plans are similar to the LTPD plans. For any specific level of AOQL, the sample size n and acceptance number c are provided for different levels of process average quality in percentage and the lot size. Both of the Dodge and Romig's plans are designed to minimize the total amount of inspections.

2.8 Summary

In summary, this thesis deals with the following in Lot Acceptance Sampling:

- Sampling by Attributes;
- Single Sampling Plans and Double Sampling Plans;
- Hypergeometric and Binomial Samplings;
- Two-point Method according to AQL and LTPD.

Chapter 3

Literature Review

For computation of the density and the cumulative hypergeometric distribution functions, Lieberman and Owen (1961) gave the tables for values of N up to 100. Tables for sampling plans and confidence intervals were given by Odeh and Owen (1983). They both used the exact calculation for hypergeometric probabilities as described below.

The exact calculation for hypergeometric probabilities concerns the direct computation of the density and cumulative distribution functions. This can be approached by either minimizing the number of operations in use of the combinatorial properties or applying the recursive relationship for hypergeometric distribution.

$$\text{For } h(x;n,N,M) = \frac{\binom{M}{x}\binom{N-M}{n-x}}{\binom{N}{n}} = \frac{n!M!(N-M)!(N-n)!}{x!N!(M-x)!(N-x)!(N-M-n+x)!}$$

where $max(0, n-N+M) \leq x \leq min(n, M)$, we have

$$h(x+1;n,M,N) = \frac{(M-x)(n-x)}{(x+1)(N-M-n+x+1)} h(x;n,M,N) \qquad (3.9)$$

Freeman (1973) applied an algorithm that calculates the probability in equation (3.9) through successive multiplication and division of the minimum number of terms, using the relations: $h(x;n,N,M) = h(M-x;N-n,N,M) = h(x;M,N,n) = h(n-$

$x; N - M, N, n)$. Lund (1980) extended Freeman's (1973) algorithm to calculate the cumulative probability $\sum_{i=a}^{x} h(i; n, M, N)$, where $a = max(0, n - N + M)$ in merit of reducing multiplications and divisions when simply repetitively accumulate the probabilities by Freeman's method. It was reported by Lillestöl (1982) that both of the methods above (Freeman (1973) and Lund (1980)) can result in errors due to the overflow (or underflow) when n and M are large. To improve accuracy and to minimize the chance of underflow, Shea (1989) incorporated a scaling factor into the improved algorithm of Lund (1980) to calculate the point and cumulative probabilities for the hypergeometric distribution. Berger (1990) made further refinement of Shea's algorithm to improve its accuracy.

The IMSL library (1989) offered algorithms to calculate the density and cumulative probability functions of hypergeometric distributions. Wu (1993) adopted a method which applied prime number factorization to the factorials and then canceled the same terms appeared in numerator and denominator to reduce the computation complexity. The results are quite comparable to IMSL routine in terms of the accuracy.

Alvo and Cabilio (2000) proposed an exact computational approach which is quite different from those described above. For given values of n and N, the algorithm uses the recurrence property of Chebyshev polynomials to simultaneously calculate the hypergeometric distribution. This thesis is based on this method to obtain the point and cumulative probabilities of hypergeometric distributions in constructing the equations for solving n and c.

Chapter 4

Chebyshev Polynomials

Alvo and Cabilio (2000) provided the mathematical basis for the on-line algorithm developed in this thesis. The details are reproduced in this chapter for completeness.

In this chapter Chebyshev polynomials are defined and used to express hypergeometric and binomial distribution.

4.1 Chebyshev Polynomials

The Chebyshev polynomials defined over the integers $x = 0, 1, \ldots, n$ can be expressed in descending factorial form as

$$\begin{aligned} f_i(x,n) &= \sum_{m=0}^{i}(-1)^m \binom{i}{m}\binom{i+m}{m}\binom{x}{m}\binom{n}{m}^{-1} \\ &= \sum_{m=0}^{i}(-1)^m \binom{2m}{m}\binom{i+m}{i-m}\binom{x}{m}\binom{n}{m}^{-1} \end{aligned}$$

where $0 \leq i \leq n$ (Ralston 1965, p238; Abramowitz and Stegun 1968, p791). Such polynomials form an orthogonal basis over a $(n+1)$-dimensional space and are easily computed from the recurrence relationship:

$$
\begin{aligned}
f_0(x,n) &= 1 \\
f_1(x,n) &= 1 - \frac{2x}{n} \\
(i+1)(n-i)f_{i+1}(x,n) &= (2i+1)(n-2x)f_i(x,n) - i(n+i+1)f_{i-1}(x,n)
\end{aligned} \tag{4.10}
$$

Let $\boldsymbol{\varepsilon}_i = (f_i(0,n),\ldots,f_i(n,n))'$ for $i = 0,1,\ldots,n$ be the basis vectors in $(n+1)$-dimensional space determined from the Chebyshev polynomials. Then any function $g(x)$ defined over the integers $x = 0,1,\ldots,n$ can expressed in vector as $\mathbf{g} = \sum_{i=0}^{n} g_i\boldsymbol{\varepsilon}_i$ where the vector $\mathbf{g} = (g(0),g(1),\ldots,g(n))'$ and the g_i can be obtained from the relation $g_i = \frac{\mathbf{g}'\boldsymbol{\varepsilon}_i}{\|\boldsymbol{\varepsilon}_i\|^2}$, where $\|\boldsymbol{\varepsilon}_i\| = \sqrt{f_i^2(0,n) + f_i^2(1,n) + \cdots + f_i^2(n,n)}, i = 0,1,\ldots,n$. Alternatively, $g(x) = \sum_{i=0}^{n} g_i f_i(x,n)$ for $x = 0,1,\ldots,n$.

4.2 Hypergeometric Distribution by Chebyshev Polynomials

In Alvo and Cabilio (2000)'s paper, the following theorem and corollary were proved and used to compute the point hypergeometric distribution.

Theorem 4.1 $\sum_{x=0}^{n} f_i(x,n)h(x;n,M,N) = f_i(M,N)$ *for* $i = 0,1,\ldots,n$ *and* $M = 0,1,\ldots,N$

Proof:

$$
\begin{aligned}
&\sum_{x=0}^{n} f_i(x,n)h(x;n,M,N) \\
&= \sum_{m=0}^{i}(-1)^m \binom{i}{m}\binom{i+m}{m}\frac{(n-m)!}{n!}\sum_{x=0}^{n}\frac{x!}{(x-m)!}h(x;n,M,N)
\end{aligned} \tag{4.11}
$$

Given by Johnson, Kotz, and Kemp (1992, p249), the second sum is:

$$
\mu'_{[m]} = \frac{n!M!(N-m)!}{(n-m)!(M-m)!N!} = \sum_{x=0}^{n}\frac{x!}{(x-m)!}h(x;n,M,N) \tag{4.12}
$$

Substitute equation (4.12) back to equation (4.11) and simplify to get:

$$\sum_{x=0}^{n} f_i(x,n)h(x;n,M,N) = \sum_{m=0}^{i}(-1)^m \binom{i}{m}\binom{i+m}{m}\binom{M}{m}\binom{N}{m}^{-1}$$

Corollary 4.1 $Eg(X) = \sum_{i=0}^{n} g_i f_i(M,N)$.

Proof:

$$\begin{aligned} Eg(X) &= \sum_{x=0}^{n} g(x)h(x;n,M,N) = \sum_{x=0}^{n}\sum_{i=0}^{n} g_i f_i(x,n)h(x;n,M,N) \\ &= \sum_{i=0}^{n} g_i f_i(M,N) \end{aligned}$$

This corollary shows that for any arbitrary function $g(x)$, we may compute the expectation $Eg(X)$ from the knowledge of the coefficient g_i and the Chebyshev polynomials up to order n evaluated at M and N. In the special case where we wish to calculate the values of the probability mass function, these coefficients are given by $g_i = \frac{f_i(x,n)}{\|\varepsilon_i\|^2}$ so that

$$h(x;n,M,N) = \sum_{i=0}^{n} \frac{f_i(x,n)}{\|\varepsilon_i\|^2} f_i(M,N) \tag{4.13}$$

where $\varepsilon_i = (f_i(0,n),\ldots,f_i(n,n))', i = 0,1,\ldots,n$.

4.3 Binomial Distribution by Chebyshev Polynomials

For the binomial distribution with parameters n and p, Alvo and Cabilio (2000) have shown that the expectation of a random variable $g(X)$ when X has a binomial distribution is given by $\sum_{i=0}^{n} g_i H_i(p)$, where the functions $H_i(p)$ satisfy the recurrence relation:

$$\begin{aligned} &H_0(p) = 1 \\ &H_1(p) = 1 - 2p \\ &(i+1)H_{i+1}(p) = (2i+1)(1-2p)H_i(p) - iH_{i-1}(p) \end{aligned} \tag{4.14}$$

Chapter 5

Algorithms

This chapter gives detailed algorithms to compute (n, c) for both the hypergeometric and binomial distribution. Before doing so, we will give mathematical model for each distribution concerned. Based on the Alvo and Cabilio (2000) paper, both the global macro in Minitab and the programs written in R programming language (2003) are given in detail. According to the type of sampling distributions, this chapter is arranged as below:

- On-line Algorithm for Hypergeometric Distribution, which includes:
  - Mathematical model;
  - On-line algorithm for Minitab macro;
  - On-line algorithm for programs written in R.
- On-line Algorithm for Binomial Distribution, which includes:
  - Mathematical model;
  - On-line algorithm for programs written in R.

5.1 On-line Algorithms for Hypergeometric Distribution

5.1.1 Mathematical model

Lemma 5.1

$$\alpha = P(X < c|k_1) = 1 - \sum_{i=0}^{n} \frac{S_i(c,n)}{\|\boldsymbol{\varepsilon}_i\|^2} f_i(k_1, N)$$

$$\beta = P(X \geq c|k_2) = \sum_{i=0}^{n} \frac{S_i(c,n)}{\|\boldsymbol{\varepsilon}_i\|^2} f_i(k_2, N)$$

where $S_i(c,n) = \sum_{x=c}^{n} f_i(x,n), i = 0, 1, \ldots, n$

Proof:

Let $S_i(c,n) = \sum_{x=c}^{n} f_i(x,n), i = 0, 1, \ldots, n$ and note that for $i \geq 1, \sum_{x=0}^{n} f_i(x,n) = 0$, where for $i = 0, \sum_{x=0}^{n} f_i(x,n) = n+1$, we may express equation (2.7) and equation (2.8) in terms of Chebyshev polynomials:

$$\begin{aligned}
\alpha &= P(X < c|k_1) = 1 - P(X \geq c|k_1) \\
&= 1 - \sum_{x=c}^{n} h(x, n, k_1, N) \\
&= 1 - \sum_{i=0}^{n} \frac{\sum_{x=c}^{n} f_i(x,n)}{\|\boldsymbol{\varepsilon}_i\|^2} f_i(k_1, N) \\
&= 1 - \sum_{i=0}^{n} \frac{S_i(c,n)}{\|\boldsymbol{\varepsilon}_i\|^2} f_i(k_1, N) \\
\beta &= P(X \geq c|k_2) = \sum_{i=0}^{n} \frac{S_i(c,n)}{\|\boldsymbol{\varepsilon}_i\|^2} f_i(k_2, N) \qquad (5.15)
\end{aligned}$$

Next, we consider the calculation of $S_i(c,n)$.

Note that

$$f_i(x,n) = \sum_{m=0}^{i}(-1)^m \frac{\binom{2m}{m}\binom{i+m}{i-m}\binom{x}{m}}{\binom{n}{m}} I(x \geq m)$$

and consequently,

$$\begin{aligned} S_i(c,n) &= \sum_{x=c}^{n} f_i(x,n) \\ &= \sum_{m=0}^{c}\sum_{x=c}^{n}(-1)^m \binom{2m}{m}\binom{i+m}{i-m}\binom{x}{m}\binom{n}{m}^{-1} \\ &+ \sum_{m=c}^{i}\sum_{x=m}^{n}(-1)^m \binom{2m}{m}\binom{i+m}{i-m}\binom{x}{m}\binom{n}{m}^{-1} \\ &= \sum_{m=c}^{i}\sum_{x=m}^{n}(-1)^m \binom{2m}{m}\binom{i+m}{i-m}\binom{x}{m}\binom{n}{m}^{-1} \end{aligned} \tag{5.16}$$

The computational region for the double summation in equation (5.16) is therefore shown in Figure 4:

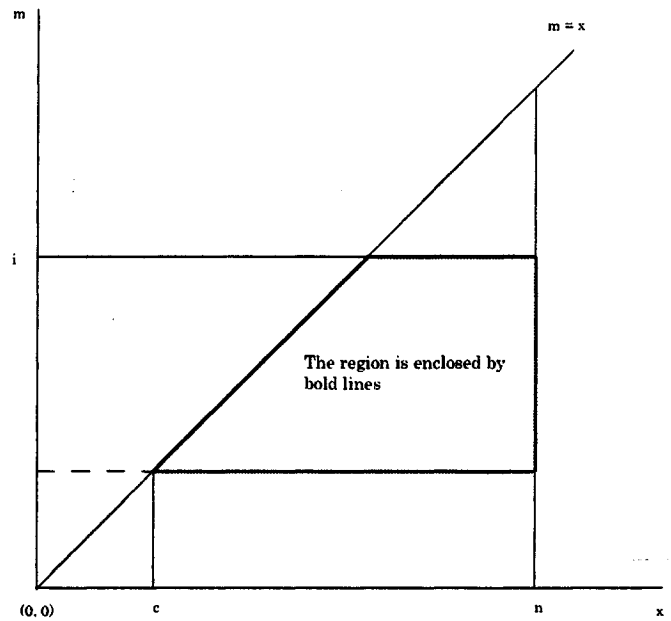

Figure 4: *Computational Region for* $S_i(c, n)$.

Once the range of (i, c, n) are known, the computation of $S_i(c, n)$ can be easily carried out under the summation region showed above since it is a sum of Chebyshev polynomials.

5.1.2 On-line algorithms for Minitab macro

Since the computations for (n, c) given $(\alpha, \beta, k_1, k_2, N)$ are rather cumbersome by Minitab macro, here only gives the algorithms for computing α and β when the values for n and c are floating. Due to the artificial restriction in Minitab, the data input for both values k_1 and k_2 can not be larger than 40. This gives a high limitation on constructing a handy sampling plan. However, the step here lays a good foundation for writing programs in R language.

- **The algorithm to compute producer's risk α given (k_1, N):**

  1. Compute for $\{f_i(x, n)\}$, for $x = 0, 1, \ldots, k_1$ and $i = 0, 1, \ldots, k_1$ by recurrence relationship of Chebyshev polynomials shown in equation (4.10);

2. Compute for $\{\|\boldsymbol{\epsilon}_i\|^2\}$, for $i = 0, 1, \ldots, k_1$
3. Compute for $\{f_i(k_1, N)\}$, for $x = 0, 1, \ldots, N$ and $i = 0, 1, \ldots, k_1$ by equation (4.10);
4. Compute for $h(x; n, k_1, N)$
5. Compute for producer's risk α by equation (2.7) and equation (4.13) and display as Table I below:

Producer's risk α				
n	$c = 0$	$c = 1$	$\cdots$	$c = n$
0				
1				
$\vdots$				
N				

The macro by Minitab is given as *Appendix A.1* Compute producer's risk α.

- **The algorithm to compute consumer's risk β given (k_2, N)**

1. Compute for $\{f_i(x, n)\}$, for $x = 0, 1, \ldots, k_2$ and $i = 0, 1, \ldots, k_2$ by recurrence relationship of Chebyshev polynomials shown in equation (4.10);
2. Compute for $\{\|\boldsymbol{\epsilon}_i\|^2\}$, for $i = 0, 1, \ldots, k_2$
3. Compute for $\{f_i(k_2, N)\}$, for $x = 0, 1, \ldots, N$ and $i = 0, 1, \ldots, k_2$ by equation (4.10);
4. Compute for $h(x; n, k_2, N)$
5. Compute for consumer's risk β by equation (2.8) and equation (4.13) and display as Table II below:

Consumer's risk β				
n	$c = 0$	$c = 1$	$\cdots$	$c = n$
0				
1				
$\vdots$				
N				

The macro by Minitab is given as *Appendix A.2* Compute consumer's risk β.

When invoking both macros, it should be noted the following:

- The data input for k_1 and k_2 should not be greater than 40 as mentioned before, otherwise the program will return missing values.
- The data outputs are read separately for α and β. After specifying (k_1, N) or (k_2, N), we will get two separate tables for α and β. When fixing the values of k_1, α, and N, one can find the values for n and c from Table I. Similarly, with fixed values of k_2, β, and N, one will be able to spot the values for n and c from Table II. Since the values for α and β are not together, attempting to read one unique pair of (n, c) from either of these two tables will certainly fail.
- The algorithms proposed here is a little different from the mathematical model presented in 5.1.1. It obtains the cumulative hypergeometric probabilities by simply adding point probabilities. The algorithms provided below for programming in the R language will be exactly the one shown in section 5.1.1.

5.1.3 On-line algorithms for programs written in R

Written in the R language (2003), the programs attempt to obtain the tables for the following programs separately:

- Compute (n, c) given $(\alpha, \beta, k_1, k_2, N)$;
- Compute c given $(\alpha, \beta, k_1, k_2, n, N)$;
- Compute upper and lower confidence limits for the number of defectives in a lot given (n, x, N) where x is the observed defective units.

- **Compute (n, c) given $(\alpha, \beta, k_1, k_2, N)$**

  This program is used to construct the acceptance sampling plan of sample size n and acceptance number c when given the pre-specified values of producer's risk α, consumer's risk β, proportion defective k_1 and k_2, and the lot size N.

  The algorithm to compute (n, c) given $(\alpha, \beta, k_1, k_2, N)$ is:

  1. Compute for $\{f_i(x, n)\}$, for $x = 0, 1, \ldots, k_1$ and $i = 0, 1, \ldots, k_1$ by recurrence relationship of Chebyshev polynomials shown in equation (4.10);

2. Compute for $\{S_i(c,n)\}$, for $c = 0, 1, \ldots, k_1$ and $i = 0, 1, \ldots, k_1$ by equation (5.16);

3. Compute for $\{f_i(k_1, N)\}$, for $x = 0, 1, \ldots, N$ and $i = 0, 1, \ldots, k_1$ by equation (4.10);

4. Compute for producer's real risk α' by equation (5.15);

5. Search for (n, c) satisfying that the real risk α' is equal to or less than pre-specified α;

6. Compute for $\{f_i(x,n)\}$, for $x = 0, 1, \ldots, k_2$ and $i = 0, 1, \ldots, k_2$ by recurrence relationship of Chebyshev polynomials shown in equation (4.10);

7. Compute for $\{S_i(c,n)\}$, for $c = 0, 1, \ldots, k_2$ and $i = 0, 1, \ldots, k_2$ by equation (5.16);

8. Compute for $\{f_i(k_2, N)\}$, for $x = 0, 1, \ldots, N$ and $i = 0, 1, \ldots, k_2$ by equation (4.10);

9. Compute for consumer's real risk β' by equation (5.15);

10. Search for (n, c) satisfying that the real risk β' is equal to or less than pre-specified β;

11. Search for (n, c) that meets both criteria and display as Table III:

	n	c	α'	β'	k_1	k_2
sampling.plan.1						
sampling.plan.2						
$\vdots$						

The program is appended as *Appendix B.1* Compute (n, c) given $(\alpha, \beta, k_1, k_2, N)$. The small letters of a and b in the function corresponds to pre-specified α and β. To narrow down the searching range and save execution time, two parameters, which are *a.prime* and *b.prime*, were added to reach the goals. Certainly, one can always set them to be zero. The function name for this program is *hyper*.

- **Compute c given $(\alpha, \beta, k_1, k_2, n, N)$**

  This algorithm intends to compute the acceptance number c when the sample size n is known. Although this approach is normally used in double/multiple sampling plans, it indeed saves a lot of time in finding appropriate acceptance number c.

  The algorithm to compute c given $(\alpha, \beta, k_1, k_2, n, N)$ is:

  1. Compute for $\{f_i(x, n)\}$, for $x = 0, 1, \ldots, k_1$ and $i = 0, 1, \ldots, k_1$ by recurrence relationship of Chebyshev polynomials shown in equation (4.10);

  2. Compute for $\{S_i(c, n)\}$, for $c = 0, 1, \ldots, k_1$ and $i = 0, 1, \ldots, k_1$ by equation (5.16);

  3. Compute for $\{f_i(k_1, N)\}$, for $x = 0, 1, \ldots, N$ and $i = 0, 1, \ldots, k_1$ by equation (4.10);

  4. Compute for producer's real risk α' by equation (5.15);

  5. Search for (n, c) satisfying that the real risk α' is equal to or less than pre-specified α;

  6. Compute for $\{f_i(x, n)\}$, for $x = 0, 1, \ldots, k_2$ and $i = 0, 1, \ldots, k_2$ by recurrence relationship of Chebyshev polynomials shown in equation (4.10);

  7. Compute for $\{S_i(c, n)\}$, for $c = 0, 1, \ldots, k_2$ and $i = 0, 1, \ldots, k_2$ by equation (5.16);

  8. Compute for $\{f_i(k_2, N)\}$, for $x = 0, 1, \ldots, N$ and $i = 0, 1, \ldots, k_2$ by equation (4.10);

  9. Compute for consumer's real risk β' by equation (5.15);

  10. Search for (n, c) satisfying that the real risk β' is equal to or less than pre-specified β;

11. Search for c that meets both criteria and display as Table IV below:

	n	c	α'	β'	k_1	k_2
sampling.plan.parameter						
sampling.plan.X						

The output table only has two rows. The first row lists all the parameters when running the program; the second row shows the required acceptance number given parameters in first row. One may find that the algorithm presented here is pretty similar with the one for computing (n, c). Actually it is the foundation to write the first program when n is floating. The detailed program can be found in *Appendix B.2* Compute c given $(\alpha, \beta, k_1, k_2, n, N)$. The function name is *hyper.fixedn*.

- **Compute upper and lower confidence limits for the number of defectives in a lot given (n, x, N)**

This program is used to compute upper and lower confidence limits for the number of defectives in a lot when the sample size n, the observed number of defectives in the sample x, and the lot size N are given. The program is able to compute any one-sided upper and lower confidence limits. Examples are shown in Section (6.1.1).

The algorithm to compute upper and lower confidence limits for the number of defectives in a lot given (n, x, N) is:

1. Compute for $\{f_i(x, n)\}$, for $x = 0, 1, \ldots, k_1$ and $i = 0, 1, \ldots, k_1$ by recurrence relationship of Chebyshev polynomials shown in equation (4.10);

2. Compute for $\{S_i(c, n)\}$, for $c = 0, 1, \ldots, k_1$ and $i = 0, 1, \ldots, k_1$ by equation (5.16);

3. Compute for $\{f_i(k_1, N)\}$, for $x = 0, 1, \ldots, N$ and $i = 0, 1, \ldots, k_1$ by equation (4.10);

4. Compute for producer's real risk α' by equation (5.15);

5. Compute for $\{f_i(x,n)\}$, for $x = 0,1,\ldots,k_2$ and $i = 0,1,\ldots,k_2$ by recurrence relationship of Chebyshev polynomials shown in equation (4.10);

6. Compute for $\{S_i(c,n)\}$, for $c = 0,1,\ldots,k_2$ and $i = 0,1,\ldots,k_2$ by equation (5.16);

7. Compute for $\{f_i(k_2,N)\}$, for $x = 0,1,\ldots,N$ and $i = 0,1,\ldots,k_2$ by equation (4.10);

8. Compute for consumer's real risk β' by equation (5.15);

9. Search for the probabilities satisfying that the upper limits are greater than the pre-specified upper confidence (γ), and lower limits are less than the pre-specified lower confidence ($1-\gamma$). Output data as Table V below:

	k	x	Conf.Limits
Upper.1	k_1	x	Conf.Limits.1
Upper.2			
⋮			
Lower.1	k_2	x	Conf.Limits.2
Lower.2			
⋮			

The program written in the R language is provided in *Appendix B.3* Compute upper and lower confidence limits. The function name is *hyper.CL*.

Since the real probability cannot be just equal to 0.95 and 0.05, or 0.975 and 0.025, $\cdots$, the ranges are then taken to allow the output to include all possible values.

The value of k is interpreted as:

- For upper confidence limits, one can be (Conf.Limits.1)% sure that the number of defective units in the lot is at most k_1;

- For lower confidence limits, one can be (1 - Conf.Limits.2)% sure that the number of defective units in the lot is at least k_2. Table V only provides one-sided confidence limits. For a lower confidence limit the confidence is

Lower; for an upper confidence limit the confidence is Upper; for a two-sided limit it is (Upper - Lower).

5.2 On-line Algorithms for Binomial Distribution

5.2.1 Mathematical model

For binomial distribution, the probability of observing exactly x defectives is given by:

$$b(x;n,p) = \binom{n}{x} p^x(1-p)^{n-x} = \frac{n!}{x!(n-x)!}p^x(1-p)^{n-x} \tag{5.17}$$

From Corollary (4.1) we know that we may compute the expectation $Eg(X)$ for any arbitrary function $g(x)$ from the knowledge of the coefficient g_i and the Chebyshev polynomials up to order n evaluated at p for the binomial distribution. In mathematical terms, we can express it by the binomial distribution as $b(x,n,p) = \sum_{i=0}^{n} g_i H_i(p)$ where $g_i = \frac{f_i(x,n)}{\|\epsilon_i\|^2}$, $f_i(x,n)$ follows the recurrence relationship in equation (4.10) and $H_i(p)$ is the same as in equation (4.14).

Lemma 5.2

$$\begin{aligned} \alpha &= P(X < c|p_1) = 1 - \sum_{i=0}^{n} \frac{S_i(c,n)}{\|\epsilon_i\|^2} H_i(p_1) \\ \beta &= P(X \geq c|p_2) = \sum_{i=0}^{n} \frac{S_i(c,n)}{\|\epsilon_i\|^2} H_i(p_2) \end{aligned} \tag{5.18}$$

$S_i(c,n)$ is the same as the one in equation (5.16).

From the equation (5.17), we know that for the binomial distribution, the probability of acceptance (p_a) is the probability that X, the number of defectives, is less than or equal to c, the acceptance number. Thus we have,

$$
\begin{aligned}
p_a &= P(X \le c) = \sum_{x=0}^{c} \binom{n}{c} p^x (1-p)^{n-x} \\
&= \sum_{x=0}^{c} \frac{\sum_{i=0}^{n} f_i(x,n)}{\|\varepsilon_i\|^2} H_i(p) \\
&= 1 - \sum_{i=0}^{n} \frac{S_i(c,n)}{\|\varepsilon_i\|^2} H_i(p)
\end{aligned}
$$

Where $S_i(c,n)$ is the same as the one in equation (5.16); p is the proportional defective per lot.

5.2.2 On-line algorithms for programs written in R

For binomial sampling plan, the programs written in R are intended to:

- **Compute for (n,c) according to pre-specified (α,β,p_1,p_2)**

  The algorithm and its output are shown below:

  For n from 2 to n.prime { },

  1. Compute for $\{f_i(x,n)\}$, for $x = 0,1,\ldots,n$ and $i = 0,1,\ldots,n$ by recurrence relationship of Chebyshev polynomials shown in equation (4.10);
  2. Compute for $\{\|\varepsilon_i\|^2\}$, for $i = 0,1,\ldots,n$;
  3. Compute for $\{H_i(p_1)\}$, for $i = 0,1,\ldots,n$ by equation (4.14);
  4. Compute for producer's real risk α' by equation (5.2);
  5. Search for (n,c) satisfying that the real risk α' is equal to or less than pre-specified α;
     Note: for{ } looping ends here.
  6. Compute for maximum n, named *maxn*, in the (n,c) table obtained in Step 5;

7. Compute for $\{f_i(x,n)\}$, for $x = 0,1,\ldots,maxn$ and $i = 0,1,\ldots,maxn$ by recurrence relationship of Chebyshev polynomials shown in equation (4.10);

8. Compute for $\{\|\varepsilon_i\|^2\}$, for $i = 0,1,\ldots,maxn$;

9. Compute for $\{H_i(p_2)\}$, for $i = 0,1,\ldots,maxn$ by equation (4.14);

10. Search for the real risk β' when n equals to $maxn$ and output a vector as $(p_1,p_2,c(1),n,\alpha'(1),\beta'(1))$,where (1) means the output obtained from step 1 to 10;

    For n from 2 to n.prime { },

11. Compute for $\{f_i(x,n)\}$, for $x = 0,1,\ldots,n$ and $i = 0,1,\ldots,n$ by recurrence relationship of Chebyshev polynomials shown in equation (4.10);
12. Compute for $\{\|\varepsilon_i\|^2\}$, for $i = 0,1,\ldots,n$,
13. Compute for $\{H_i(p_2)\}$, for $i = 0,1,\ldots,n$ by equation (4.14);
14. Compute for consumer's real risk α' by equation (5.2);
15. Search for (n,c) satisfying that the real risk β' is equal to or less than pre-specified β;
    Note: for{ } looping ends here.

16. Compute for minimum n, named $minn$, in the (n,c) table obtained in Step 15;

17. Compute for $\{f_i(x,n)\}$, for $x = 0,1,\ldots,minn$ and $i = 0,1,\ldots,minn$ by recurrence relationship of Chebyshev polynomials shown in equation (4.10);

18. Compute for $\{\|\varepsilon_i\|^2\}$, for $i = 0,1,\ldots,minn$;

19. Compute for $\{H_i(p_1)\}$, for $i = 0,1,\ldots,minn$ by equation (4.14);

20. Search for the real risk α' when n equals to $minn$ and output a vector as $(p_1,p_2,c(2),n,\alpha'(2),\beta'(2))$,where (2) means the output obtained from step 10 to 20;

21. Combine the vectors in step 10 and step 20, the output table (Table VI) looks like the following:

	p_1	p_2	c	n	α'	β'
Sampling.plan.1						
Sampling.plan.2						

Remarks:

- The variable *n.prime* in the algorithm above means the largest value for the sample size n;
- The values of p_1, p_2, and c are the same in sampling.plan.1 and sampling.plan.2;
- The value of n in sampling.plan.1 is the maximum n that satisfies the real risk α' is equal to or less than the pre-specified α.
- The value of n in sampling.plan.2 is the minimum n that satisfies the real risk β' is equal to or less than the pre-specified β.
- The program is appended as *Appendix C.1* Compute (n, c) given $(\alpha, \beta, p_1, p_2)$ with the function name *binom*;

- **Draw OC curve for given (n, c)**

The algorithm to draw OC curve given the sample size n and the acceptance number c is:

1. Compute for $\{f_i(x, n)\}$, for $x = 0, 1, \ldots, n$ and $i = 0, 1, \ldots, n$ by recurrence relationship of Chebyshev polynomials shown in equation (4.10);
2. Compute for $\{\|\varepsilon_i\|^2\}$, for $i = 0, 1, \ldots, n$,
3. Compute for $\{H_i(p)\}$, for $i = 0, 1, \ldots, n$ and $p = 0.01, 0.02, \ldots$ by equation (4.14):

4. Compute for the probability of acceptance by equation (5.17)

5. Output the graph for selected c.

The program written by R is shown in *Appendix C.2* Draw OC curve given (n, c). The name of the function is *binom.graph*. In the function, the variable *p.prime* is used to set up the upper limit for p ($p \leq 100$).

Chapter 6

Data Analysis

The analysis of results concerns two aspects: accuracy and execution time. Accuracy of results will be compared with Odeh and Owen (1983)'s table. The execution times are shown for different programs.

Compared to any existing table for checking (n, c) sampling plans, the algorithms proposed in this paper are superior in providing (n, c) for any pre-specified values of $(\alpha, \beta, k_1, k_2, N)$. Therefore, we call it ON-LINE here. We will soon show how this word - online works.

Since the magnitude of the largest coefficients of i are increasing roughly as 2^i, the calculation of their squared length may cause overflow, and division may cause loss of significance. The detailed comparisons will be made in the following sections to ensure the accuracy of the computations under the algorithms mentioned above. All the comparisons are made with Odeh and Owen (1983)'s table in terms of accuracy.

In the present high technology era, it is no doubt that the time consumed by the computation must be reasonably small. Execution time for each program will be shown for reference.

Simply put, this chapter conducts data analysis about:

- Precision with regard to Odeh and Owen (1983)'s table;
- Execution time.

6.1 Precision

For a clearer presentation, the comparison will be made according to the sequence of the programs presented in Chapter 5. The results shown here are under each function that is uniquely named.

To execute the functions written in R, one takes the following three steps:

1. According to the computational purpose, copy the required program into Notepad and save it as .txt file into the disk, e.g. save the function *hyper* as the file 'hyper.txt' to c:\;
2. Load the function with source( ) to the memory, e.g. source("c:/hyper.txt") if the file is saved in c:\;
3. Execute the function.

The third step will be explained in detail for different functions shown below.

6.1.1 Hypergeometric sampling

1. **Compute (n, c) for pre-specified $(\alpha, \beta, k_1, k_2, N)$,**

   The function for computing (n, c) given $(\alpha, \beta, k_1, k_2, N)$ is named as **hyper**. It includes the variables of $(a, a.prime, b, b.prime, k_1, k_2, N)$. The meanings for each variable are as follows:

a	pre-specified α;
$a.prime$	the lower bound for pre-specified α;
b	pre-specified β;
$b.prime$	the lower bound for pre-specified β;
k_1	proportion defective associated with α;
k_2	proportion defective associated with β;
N	lot size;

   After loading the function to the memory, one inputs hyper( ) in R console to execute it, just as executing any other functions in R. For example, when given $\alpha = 0.01, \beta = 0.05, k_1 = 20, k_2 = 40, N = 400$, one may execute the function

hyper as "hyper(0.01, 0.005, 0.05, 0.04, 20, 40, 400)". 0.005 is the value for the variable *a.prime* and 0.04 is the value for the variable *b.prime*. The sequence is important. If α is given as 0.01, then the first value should be 0.01. One may check the first line of the program to see how these variables are arranged.

The values for *a.prime* and *b.prime* can always set to be zero. However, to save the execution time, it is not suggested to set them too far from the pre-specified α and β. If the ranges are too narrow and no sampling plan is returned, the message "the ranges for alpha and/or beta are too narrow, please reselect them" will appear. Under this circumstance, one needs to widen the range for α or β or both.

To make the analysis comparable to Odeh and Owen (1983) table, we present this part according to whether the interpolation is required or not.

- **No interpolation required**

  In this case, all the values of $(\alpha, \beta, k_1, k_2, N)$ are clearly shown on Odeh and Owen (1983) table. One only needs to enter the appropriate table to find out the values for (n, c).

  Example 1: $\boldsymbol{N = 400, \alpha = 0.01, \beta = 0.05, k_1 = 20, k_2 = 40}$
  $\succ$ *hyper*(0.01, 0.005, 0.05, 0.04, 20, 40, 400)

	n	c	α'	β'	k_1	k_2
sampling.plan.1	205	15	0.006801390	0.04744154	20	40
sampling.plan.2	206	15	0.007242006	0.04422432	20	40
sampling.plan.3	207	15	0.007707426	0.04118369	20	40

  The result shown in Odeh and Owen (1983)'s table in Table 4.1.1 is:

	n	c	α'	k_2	β'
$k_1 = 20$	205	15	0.007	40	0.047

  Pick sampling.plan.1, we see that they are pretty much the same except that different numbers of digits are displayed.

Example 2: $N = 1200, \alpha = 0.01, \beta = 0.1, k_1 = 60, k_2 = 120$
$\succ$ *hyper*(0.01, 0.007, 0.10, 0.07, 60, 120, 1200)

	n	c	α'	β'	k_1	k_2
sampling.plan.1	271	21	0.007915019	0.09650641	60	120
sampling.plan.2	272	21	0.008290144	0.09274479	60	120
sampling.plan.3	273	21	0.008680008	0.08909892	60	120
sampling.plan.4	274	21	0.009085048	0.08556677	60	120
sampling.plan.5	275	21	0.009505706	0.08214629	60	120
sampling.plan.6	276	21	0.009942433	0.07883536	60	120
sampling.plan.7	286	22	0.007190396	0.08164052	60	120
sampling.plan.8	287	22	0.007527441	0.07838426	60	120
sampling.plan.9	288	22	0.007877674	0.07523234	60	120
sampling.plan.10	289	22	0.008241490	0.07218270	60	120

The result shown in Odeh and Owen (1983)'s table in Table 4.2.5 is:

	n	c	α'	k_2	β'
$k_1 = 60$	271	21	0.008	120	0.097

Example 3: $N = 2000, \alpha = 0.05, \beta = 0.2, k_1 = 200, k_2 = 400$, which is one of the highest value groups set up in Odeh and Owen (1983)'s table
$\succ$ *hyper*(0.050, 0.045, 0.20, 0.15, 200, 400, 2000)

	n	c	α'	β'	k_1	k_2
sampling.plan.1	72	11	0.04972725	0.1939184	200	400
sampling.plan.2	79	12	0.04596499	0.1723536	200	400

The result excerpted from Odeh and Owen (1983)'s table in Table 4.9.9:

	n	c	α'	k_2	β'
$k_1 = 200$	72	11	0.050	400	0.194

- **Interpolation required**

Due to limitations of the table, interpolations are required with respect to proportion defectives k_1, k_2 and lot size N. The function used in this part is exactly the same as the one without interpolation. Since there are no constraints on the values of k_1, k_2, and N in our program, one does not need to do any interpolation for any values of k_1, k_2, and N. We will

show different interpolation methods in Odeh and Owen (1983) table and compare them with our program's outputs.

⇒ Interpolation with respect to k_2:

Example: $N = 1,000, \alpha = 0.05, \beta = 0.05, k_1 = 35, k_2 = 72$
$\succ$ *hyper*(0.050, 0.040, 0.050, 0.040, 35, 72, 1000)

	n	c	α'	β'	k_1	k_2
sampling.plan.1	284	14	0.04448400	0.04981921	35	72
sampling.plan.2	285	14	0.04581390	0.04791016	35	72
sampling.plan.3	286	14	0.04717338	0.04606254	35	72
sampling.plan.4	287	14	0.04856280	0.04427491	35	72
sampling.plan.5	288	14	0.04998249	0.04254585	35	72

In Odeh and Owen (1983)'s table, an interpolation with respect to k_2 is given as below:

When $N = 1,000, \alpha = 0.05, \beta = 0.05, k_1 = 35, k_2 = 72$. Table 4.7.4 contains entries for $k_1 = 35$ as follows:

n	c	α'	k_2	β'
308	15	0.042	70	0.049
259	13	0.045	75	0.048

Interpolation and rounding up gives $\boldsymbol{n = 289}$, $\boldsymbol{c = 15}$. The actual computation gives the interpolated results for the following real risks:

	n	c	α'	k_2	β'
$k_1 = 35$	289	15	0.02349	72	0.073086

In consideration of Type I and Type II errors, we pick up sampling.plan.1, which gives $\boldsymbol{n = 284}$ and $\boldsymbol{c = 14}$. From the procedure listed above, we can see that there is no interpolation required in meeting pre-specified α, β, k_1, k_2, and N. The program covers every possible set of value for α, β, k_1, k_2, and N. We will show some more examples below.

⇒ Interpolation with respect to k_1:

Example: $N = 1,200, \alpha = 0.05, \beta = 0.20, k_1 = 40, k_2 = 84$
$\succ$ *hyper*(0.050, 0.040, 0.20, 0.15, 40, 84, 1200)

	n	c	α'	β'	k_1	k_2
sampling.plan.1	191	10	0.04111998	0.1890322	40	84
sampling.plan.2	192	10	0.04254417	0.1834154	40	84
sampling.plan.3	193	10	0.04400338	0.1779191	40	84
sampling.plan.4	194	10	0.04549801	0.1725427	40	84
sampling.plan.5	195	10	0.04702844	0.1672855	40	84
sampling.plan.6	196	10	0.04859505	0.1621466	40	84

An interpolation with respect to k_1 in Odeh and Owen (1983)'s table is given as below:

When $N = 1,200, \alpha = 0.05, \beta = 0.20, k_1 = 40, k_2 = 84$, we find two entries close in Table 4.9.5:

	n	c	α'	k_2	β'
$k_1 = 36$	158	8	0.038	84	0.198
$k_1 = 42$	205	11	0.042	84	0.198

Interpolation gives $\boldsymbol{n = 190}$ and $\boldsymbol{c = 10}$, where we have rounded up our interpolated answers to integer values.

Our program gives the answer $\boldsymbol{n = 191}$ and $\boldsymbol{c = 10}$, *without interpolation.*

$\Rightarrow$ Interpolation with respect to N:

Example: $N = 500, \alpha = 0.025, \beta = 0.20, k_1 = 6, k_2 = 20$
$\succ$ *hyper*(0.025, 0.015, 0.20, 0.15, 6, 20, 500)

	n	c	α'	β'	k_1	k_2
sampling.plan.1	162	4	0.01510857	0.1676464	6	20
sampling.plan.2	163	4	0.01555102	0.1629013	6	20
sampling.plan.3	164	4	0.01600326	0.1582534	6	20
sampling.plan.4	165	4	0.01646542	0.1537020	6	20

Interpolation with respect to N is given below as in Odeh and Owen (1983)'s table:

If $N = 500, \alpha = 0.025, \beta = 0.20, k_1 = 6, k_2 = 20$, first compute $p_1 = k_1/N = 0.012, p_2 = k_2/N = 0.04$. Table 4.6.1 has $N = 400$ and Table 4.6.2 has $N = 600$ for the required α and β. For entering Table 4.6.1 we compute $k_1 = p_1 \times N = 4.8$ and $k_2 = p_2 \times N = 16$. We round the k_1 to 5 and interpolate in Table 4.6.1:

	n	c	α'	k_2	β'
$k_1 = 4$	127	3	0.010	16	0.196
$k_1 = 6$	179	5	0.008	16	0.198

and obtain $n = 153, c = 4$ and $N = 400$. Turning to Table 4.6.2 we find for $k_1 = 600(0.012) = 7.2$ (rounded up to 8) and $k_2 = 600(0.04) = 24$:

	n	c	α'	k_2	β'
$k_1 = 6$	130	3	0.022	24	0.198
$k_1 = 9$	212	6	0.011	24	0.196

and hence the plan for $N = 600$ would be $n = 185, c = 5$. Finally, we interpolate between $N = 400$ and $N = 600$ and obtain $\boldsymbol{n = 169}$, $\boldsymbol{c = 5}$.

It takes a long procedure to get the result of $n = 169$ and $c = 5$. We can check the real risks when $n = 169$ and $c = 5$ given $N = 500, k_1 = 6$ and $k_2 = 20$:

	n	c	α'	k_2	β'
$k_1 = 6$	169	5	0.001405	20	0.277493

Obviously, interpolation does not give a sound sampling plan to satisfy the pre-specified criteria. Take the outputs from running our program, the sampling plan is $\boldsymbol{n = 162}$ and $\boldsymbol{c = 4}$.

2. **Compute c for pre-specified $(\alpha, \beta, k_1, k_2, n, N)$, n is known and fixed**:

This program is designed to compute the acceptance number c when the sample size n is known and fixed. The function name is **hyper.fixedn**. It has the variables (a, b, k_1, k_2, n, N), which mean exactly the same as in the function *hyper*.

Since the sample size n is fixed, this program return a unique sampling plan of (n, c) pair. This approach is usually used in double/multiple sampling plan to

compute for the acceptance number in different stages. The results are straightforward; we will give one example here:

Example: $N = 1,000, \alpha = 0.01, \beta = 0.10, k_1 = 40, k_2 = 80, n = 307$
$\succ$ *hyper.fixedn*(0.01, 0.10, 40, 80, 307, 1000)

	n	c	α'	k_2	β'
sampling.plan.parameter	307	40	0.010000000	80	0.10000000
sampling.plan.X	307	19	0.007168896	80	0.09870234

In Odeh and Owen (1983)'s table, the result is:

	n	c	α'	k_2	β'
$k_1 = 40$	307	19	0.007	80	0.099

The results are exactly the same except for the different number of digits.

3. **Compute upper and lower confidence limits for the number of defectives in a lot given (n, x, N)**

The function name for this program is **hyper.CL**. It has variables ($UCL.prime$, $Upper.CL$, $Lower.CL$, $LCL.prime$, n, x, N) in sequence, which mean:

- $UCL.prime$: upper bound for the one-sided upper confidence;
- $Upper.CL$: the one-sided upper confidence;
- $Lower.CL$: the one-sided lower confidence;
- $LCL.prime$: lower bound for the one-sided lower confidence;
- n: sample size;
- x: the observed number of defective items in a random sample;
- N: lot size.

When executing the function, one only needs to type hyper.CL( ) in R console after loading it to the memory. For example, when one observed 12 defective items in a sample sized at 160 taken from a lot size 1000, keying in "hyper.CL(0.98, 0.975, 0.025, 0.020, 160, 12, 1000)" in R console will give the results for the one-sided 97.5% upper and lower confidence limits for the number of defective items in the lot. More detailed interpretation for these limits are given in the

example below.

Example 1: $N = 1,000, n = 160, x = 12$:
$\succ$ *hyper.CL*(0.98, 0.975, 0.025, 0.020, 160, 12, 1000)

	k	x	*Conf.Limits*
Upper.1	124	12	0.97704147
Upper.2	125	12	0.97906486
Lower	41	12	0.02116213

$\succ$ *hyper.CL*(0.955, 0.950, 0.050, 0.040, 160, 12, 1000)

	k	x	*Conf.Limits*
Upper	116	12	0.95354971
Lower	45	12	0.04288032

$\succ$ *hyper.CL*(0.955. 0.950. 0.050, 0.045, 160, 12, 1000)

	k	x	*Conf.Limits*
Upper	"116"	"12"	"0.953549705044825"
Lower	"The range for lower confidence limits is too narrow, please reselect it"		

$\succ$ *hyper.CL*(0.905, 0.90, 0.10, 0.095, 160, 12, 1000)

	k	x	*Conf.Limits*
Upper	107	12	0.90478650
Lower	51	12	0.09883581

In Odeh and Owen (1983)'s table, the results in Table 5.4.2 are showed as below:

n	γ	$1-\alpha$	k	$x = 12$
160	0.025	0.95	41	0.0212
	0.975		124	0.9770
	0.050	0.900	45	0.0429
	0.950		116	0.9535

The results are quite comparable except for the number of digits. Since the searching criteria are of the ranges in nature, it therefore outputs a list of possible values which satisfy those criteria. We normally pick the results that are as nearly as possible to the one-sided confidence γ and $1-\gamma$. The values for each variable excluding (n, x, N) are suggested as below:

	$\gamma = 0.975$, $1-\gamma = 0.025$	$\gamma = 0.950$, $1-\gamma = 0.050$	$\gamma = 0.900$, $1-\gamma = 0.100$
UCL.Prime	0.980	0.955	0.905
Upper.CL	0.975	0.950	0.900
Lower.CL	0.025	0.050	0.100
LCL.Prime	0.020	0.045	0.095

If the ranges are too narrow and no confidence limits are returned, the message "The range for upper confidence limits is too narrow, please reselect it" and/or "The range for lower confidence limits is too narrow, please reselect it" will appear. Then one needs to widen the upper or the lower or both of the confidence range(s).

Take the example shown above, the interpretation for k is as follows:

If we observe $x = 12$ defective units in a sample of size 160 from a lot of 1,000,

(a) We can be 97.88% (1 - 0.0212 = 0.9788) sure that the number of defectives in the lot is at least 41.

(b) We can be 97.70% sure that the number of defective units in the lot is at most 124.

(c) We can be 95.58% (0.9770 - 0.0212 = 0.9558) sure that the number of defective units in the lot must be between 41 and 124.

Another example is similar but with higher N, n and x. We will look at the precision for the results.

Example 2: $\boldsymbol{N = 2,000, n = 400, x = 140}$, which is the highest (n, x, N) in Odeh and Owen (1983)'s table:
$\succ$ $hyper.CL(0.98, 0.975, 0.025, 0.020, 400, 140, 2000)$

	k	x	*Conf.Limits*
Upper.1	788	140	0.97524011
Upper.2	789	140	0.97652489
Upper.3	790	140	0.97775277
Upper.4	791	140	0.97892568
Lower.1	613	140	0.02089764
Lower.2	614	140	0.02217089
Lower.3	615	140	0.02350799
Lower.4	616	140	0.02491123

$\succ$ $hyper.CL(0.955, 0.950, 0.050, 0.045, 400, 140, 2000)$

	k	x	*Conf.Limits*
Upper.1	774	140	0.95013096
Upper.2	775	140	0.95242968
Upper.3	776	140	0.95464205
Lower.1	627	140	0.04539321
Lower.2	628	140	0.04777693

The result in Odeh and Owen (1983)'s table is:

n	γ	$1-\alpha$	k	$x = 140$
400	0.025	0.95	616	0.0249
	0.975		788	0.9752
	0.050	0.900	628	0.0478
	0.950		774	0.9501

Since a detailed probability is given for every value of k, one may select the one which is nearest to the pre-specified γ and $1-\gamma$. We normally take the first line for upper and the last line for the lower confidence limits.

6.1.2 Binomial sampling

1. **Compute (n, c) given $(\alpha, \beta, p_1, p_2)$**

The function name for this program is **binom**. It has the following variables to execute the function:

- *n.prime*: the largest value where n goes to in computing cumulative binomial probabilities;
- a and b: pre-specified values for α and β respectively;
- *a.prime* and *b.prime*: the lower bounds for pre-specified α and β respectively;
- p_1 and p_2: proportion defectives, pre-specified.

For example, when given $\alpha = 0.010, \beta = 0.050, p_1 = 0.025, p_2 = 0.114$, one may key in the function in R console as "binom(120, 0.025, 0.114, 0.010, 0.005, 0.050, 0.045)" to correspond with the given values of (*n.prime*, p_1, p_2, a, *a.prime*, b, *b.prime*). The values for *n.prime* could be any integer. When picking the values for *a.prime* and *b.prime*, it is not suggested to set them to be too far from the pre-specified values of α and β. For example, if α is 0.010, then *a.prime* is at least 0.005. The same rule applies to *b.prime*. The reason is that the the wide range for α or β could lead to incorrect result for maximum n, which in turn leads to all other values, such as c to be incorrect either.

Example: $\boldsymbol{\alpha = 0.010, \beta = 0.050, p_1 = 0.025, p_2 = 0.114}$
$\succ$ *binom*(120, 0.025, 0.114, 0.010, 0.005, 0.050, 0.045)

	p_1	p_2	c	n	α'	β'
sampling.plan.1	0.025	0.114	7	118	0.00984918	0.03429389
sampling.plan.2	0.025	0.114	7	113	0.00767669	0.04747355

The results in Odeh and Owen (1983)'s table is:

p_1	p_2	c	n	p_1'	p_2'	α'	β'
0.0250	0.114	7	118	0.0251	0.1085	0.010	0.034
			113	0.0262	0.1132	0.008	0.047

2. **Draw OC curve for given (n, c)**

For binomial sampling plan, an OC curve is drawn according to the probability of acceptance (p_a) versus the proportional defective (p). The probability of acceptance is decreasing when the proportional defective is increasing.

The function runs according to three variables: sample size n, acceptance number c, and maximum value for p ($p.prime$). The values of (n, c) uniquely determine the shape of OC curve. One can set $p.prime$ to be any value which is less than 100.

Example 1: $\boldsymbol{n = 52, c = 3}$
$\succ$ $binom.graph(52, 3, 15)$

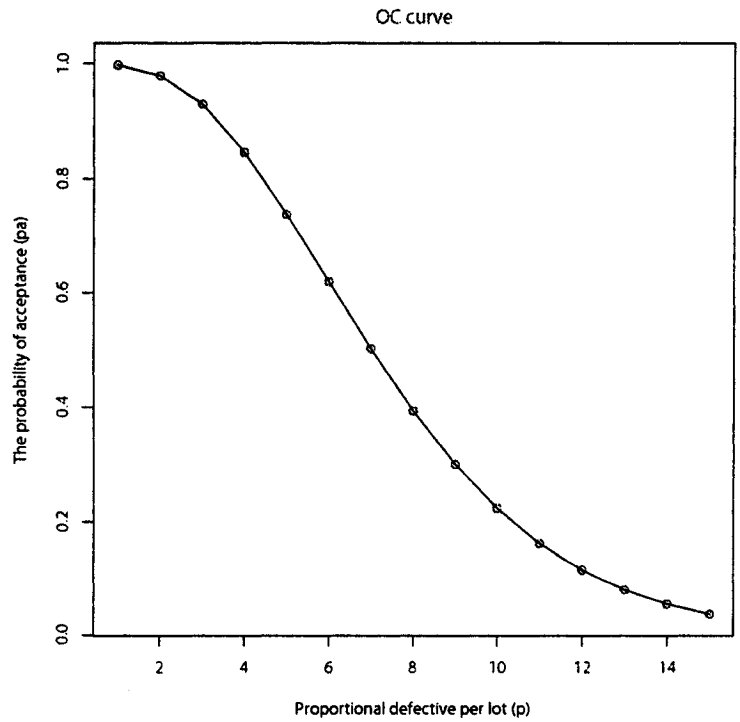

Figure 5: *OC Curve for Binomial Sampling (52, 3).*

Example 2: $\boldsymbol{n = 98, c = 4}$
$\succ$ $binom.graph(98, 4, 12)$

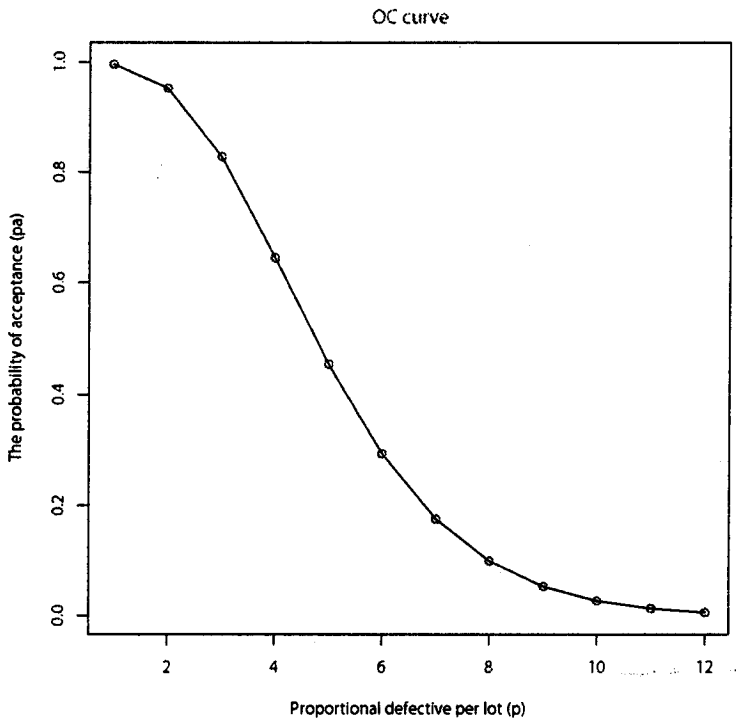

Figure 6: *OC Curve for Binomial Sampling (98, 4).*

6.1.3 Summary

In summary, we see that all the results outputted for the programs are accurate with respect to Odeh and Owen (1983)'s table. In addition to the ease of operation, one can also get results on-line with no interpolations required for any values of k_1, k_2 and N. Next, we will talk about the execution time.

6.2 Execution Time

Due to the different programming language, machines and operating systems, we will not compare our results with IMSL MDHYP routine. The comparison in Table VII below is made for different programs implementing different tasks.

Compute for	Given	Parameters	Time(in seconds)		
			User CPU	System CPU	Elapsed
Hypergeometric Sampling					
(n, c)	$(\alpha, \beta, k_1, k_2, N)$	(0.01, 0.005, 20, 40, 400)	6.46	0.03	6.66
		(0.01, 0.10, 60, 120, 1200)	96.31	1.29	101.42
		(0.05, 0.20, 200, 400, 2000)	663.41	15.81	701.95
		(0.05, 0.05, 35, 72, 1000)	32.91	0.07	33.83
		(0.05, 0.20, 40, 84, 1200)	56.86	0.02	58.56
		(0.025, 0.20, 6, 20, 500)	4.02	0.01	4.13
c	$(\alpha, \beta, k_1, k_2, n, N)$	(0.01, 0.10, 40, 80, 307, 1000)	75.60	0.03	77.30
Conf. Limits	$(UCL.prime, Upper.CL\ Lower.CL\ LCL.prime\ n, x, N)$	(0.955, 0.950 0.05, 0.04, 160, 12, 1000)	43.86	0.56	45.40
		(0.980, 0.975, 0.025, 0.020, 160, 12, 1000)	42.54	0.61	44.18
		(0.955, 0.950 0.05, 0.045, 400, 140, 2000)	414.39	12.42	437.80
		(0.980, 0.975 0.025, 0.020, 400, 140, 2000)	382.46	5.56	397.05
Binomial Sampling					
(n, c)	$(p_1, p_2, \alpha, \beta)$	(0.025, 0.114, 0.010, 0.050)	196.93	0.16	201.70
OC	(n, c)	(52, 3)	0.81	0.14	1.23
		(98, 4)	3.03	0.02	3.13
		(200, 7)	24.49	0.02	25.05

The execution times listed in the table above include: user CPU, system CPU and elapsed time. User CPU time refers to time spent executing database manager code and System CPU represents the time spent in system calls.

The configuration for the computer executing the programs in the table is:

- System: Microsoft XP Professional Version 2002
- Computer: Intel Celeron processor 851 MHz 248MB of RAM
- Programming language: R language

From the table above, we see the execution time is reasonably short when k_1, k_2, and N are small. As the values of k_1, k_2 and N get large, the execution times are increasing rapidly but all within acceptable range for hypergeometric sampling. The execution time for binomial sampling is much longer than the similar parameters under hypergeometric sampling even when the maximum n embedded in the program is small (e.g. n is 120). The reason for such a low speed is that there are four loopings in the program in order to obtain real α and β. The sample size n in binomial sampling is floating, in other words, it is a range, not a specific number as in the hypergeometric case.

Chapter 7

Conclusions and Recommendation for Future Study

The hypergeometric distribution can be represented in terms of Chebyshev polynomials. This representation is particularly useful in the calculation of expected values of hypergeometric variables, since the polynomials can be given in recurrence form and they form an orthogonal basis.

This recurrence relationship of Chebyshev polynomials can be easily computed by computer programs. One of the applications would be constructing attribute sampling plans. This thesis mainly shows the online algorithms of computing cumulative hypergeometric probabilities using Chebyshev polynomials. It also shows the extension to binomial samplings. Both computations return satisfactory computational precision and reasonable execution time for different programs.

Compared to the existing sampling tables (e.g. Odeh and Owen (1983)), the online algorithms discussed in the thesis provide all-scale computation capability, fast speed, and selectable results. With all computations, there are no interpolations required. For a lot size of around 500, it only takes about 4 seconds. The results are outputted in table format and one can decide which set of sample size and acceptance number to be selected from the table given the corresponding producer's and consumer's risks.

This thesis dealt primarily with hypergeometric sampling plans. For binomial plans, further studies could be focused on fixed sample size, double and/or multiple sampling plans. For binomial sampling plans, an OC curve for known sample size and acceptance number is critical. We briefly suggested an algorithm to draw an

OC curve for binomial samplings. One can express equation (2.5) using Chebyshev polynomials for constructing binomial double sampling plans. This would be a good start for all studies within binomial sampling regime.

Bibliography

[1] ABRAMOWITZ, M., AND STEGUN, I. A. (1968). *Handbook of Mathematical Functions*, Washington, DC: National Beureau of Standards.

[2] ALVO, M., AND CABILIO, P. (1998). Applications of Hamming Distance to the Analysis of Block Designs, *Asympototic methods in Probability and Statistics*, ed. B. Szyszkowicz, Amsterdam: Elsevier Science, pp. 787-799.

[3] ALVO, M., AND CABILIO, P. (2002). Calculation of Hypergeometric Probabilities Using Chebyshev Polynomials, *The American Statistican*, 54, 141-144.

[4] AMERICAN SOCIETY FOR QUALITY. http://www.asq.org/.

[5] BERGER, R.L. (1991). Algorithm AS R86-A Remark on Algorithm AS 152: Cumulative Hyergeometric Probabilities, *Applied Statistics*, 40, 374-375.

[6] BOWKER, A.H., AND LIEBERMAN, G.J. (1959). *Engineering Statistics*, New Jersey: Prentice-Hall.

[7] FREEMAN, P.R. (1973). Algorithm AS 59 Hypergeometric Probabilities, *Applied Statistics*, 22, 130-133.

[8] HALD, A. (1981). *Statistical Theory of Sampling Inspection by Attributes*, London: Academic Press.

[9] HILLIARD, S. (1999). *Tutorial on Modern Acceptance Sampling*, http://www.samplingplans.com/modern3.htm.

[10] LUND, R.E. (1980). Algorithm AS 152 Cumulative Hypergeometric Probabilities, *Applied Statistics*, 29, 221-223.

[11] NIST/SEMATECH (2000). *e-Handbook of Statistical Methods*, http://www.itl.nist.gov/div898/handbook/.

[12] ODEH, R.E., AND OWEN, D.B. (1983). *Attribute Sampling Plans, Tables of Tests and Confidence Limits for Proportions*, New York: Marcel Dekker.

[13] R DEVELOPMENT CORE TEAM (2003). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-00-3, URL http://www.R-project.org.

[14] RALSTON, A. (1965). *A first Course in Numerical Analysis*, New York: McGraw-Hill.

[15] SHEA, B.L. (1989), Remark AS R77-A Remark on Algorithm AS 152 Cumulative Hypergeometric Probabilities, *Applied Statistics*, 38, 199-204.

[16] USER'S MANUAL (1989). *IMSL Stat/Library: FORTRAN Subroutines for Statistical Analysis*, Version 1.1, Houston, TX: IMSL.

[17] WU, T. (1993). An Accurate Computation of the Hypergeometric Distribution Function, *ACM Transactions on Mathematical Software*, 19, 33-43.

Appendix A

Minitab Macro

A.1 Compute producer's risk α

This macro generates the value of the producer's risk alpha (α) for **hypergeometric samplings**, given proportion defective k_1 and lot size N.

```
GMACRO
HYPERTEST1
noecho
Note What is the proportion defective k1?
Note What is the size of the population N?
set c90;
file 'terminal';
nobs= 2.
copy c90 k1 k2
set c1
0:k1
end
name c1 'n'
let c2=c1*0+1
let c3=1-2*c1/k1
let k7=k1-1
  do k3=1:k7
  let k4=k3+3
  let k5=k3+2
  let k6=k3+1
```

```
  let ck4=((2*k3+1)*(k1-2*c1)*ck5-(k3)*(k1+k3+1)*ck6)/((k6)*(k1-k3))
  enddo
let k3=k1+2
  do k8=2:k3
  let ck8=ck8/1000
  enddo
  do k4=2:k3
  let k5=ssq(ck4)
  let ck4=ck4/k5
  let ck4=ck4/1000
  enddo
let k4=k1+2
let k5=k1+1
copy c2-ck4 m2
erase c2-ck4
transpose m2 m3
erase m2
set c1 0:k2
end
let c2=c1*0+1
let c3=1-2*c1/k2
  do k3=1:k7
  let k4=k3+3
  let k5=k3+2
  let k6=k3+1
  let ck4=((2*k3+1)*(k2-2*c1)*ck5-(k3)*(k2+k3+1)*ck6)/((k6)*(k2-k3))
  enddo
copy c2-ck4 m4
erase c2-ck4
multi m4 m3 m5
transpose m5 m6
let k8=k2+2
copy m6 c2-ck8
  let k9=k2+2
  do k10=2:k9
  let ck10=pars(ck10)
  let ck10=1-ck10
  enddo
let k11=k2+2
copy c2-ck11 m7
transpose m7 m8
erase c2-ck11
let k12=k1+2
```

```
copy m8 c2-ck12
set c1 0:k2
end
name c2 'c=0'
ENDMACRO
```

A.2 Compute consumer's risk β

This macro generates the value of the consumer's risk beta (β) for **hypergeometric samplings**, given proportion defective k_2 and lot size N.

```
GMACRO
HYPERTEST2
noecho
Note What is the proportion defective k2?
Note What is the size of the population N?
set c90;
file 'terminal';
nobs= 2.
copy c90 k1 K2
set c1 0:k1
end
name c1 'n'
let c2=c1*0+1
let c3=1-2*c1/k1
let k7=k1-1
  do k3=1:k7
  let k4=k3+3
  let k5=k3+2
  let k6=k3+1
  let ck4=((2*k3+1)*(k1-2*c1)*ck5-(k3)*(k1+k3+1)*ck6)/((k6)*(k1-k3))
  enddo
let k3=k1+2
  do k8=2:k3
  let ck8=ck8/1000
  enddo
  do k4=2:k3
  let k5=ssq(ck4)
  let ck4=ck4/k5
  let ck4=ck4/1000
```

```
  enddo
let k4=k1+2
let k5=k1+1
copy c2-ck4 m2
erase c2-ck4
transpose m2 m3
erase m2
set c1 0:k2
end
let c2=c1*0+1
let c3=1-2*c1/k2
  do k3=1:k7
  let k4=k3+3
  let k5=k3+2
  let k6=k3+1
  let ck4=((2*k3+1)*(k2-2*c1)*ck5-(k3)*(k2+k3+1)*ck6)/((k6)*(k2-k3))
  enddo
copy c2-ck4 m4
erase c2-ck4
multi m4 m3 m5
transpose m5 m6
let k8=k2+2
copy m6 c2-ck8
  let k9=k2+2
  do k10=2:k9
  let ck10=sum(ck10)-pars(ck10)
  let ck10=1-ck10
  enddo
let k11=k2+2
copy c2-ck11 m7
transpose m7 m8
erase c2-ck11
let k12=k1+2
copy m8 c2-ck12
set c1 0:k2
end
name c2 'c=0'
ENDMACRO
```

Appendix B

R Code for Hypergeometric Sampling

B.1 Compute (n, c) given $(\alpha, \beta, k_1, k_2, N)$

This program is used for obtaining sample size n and acceptance number c, that is (n, c), according to pre-specified producer's risk α, consumer's risk β, proportion defective k_1, and k_2 and lot size N $(\alpha, \beta, k_1, k_2, N)$ for **hypergeometric sampling**.

```
hyper <- function(a, a.prime, b, b.prime, k1, k2, N) {
# compute fi(x, n)
   M <- vector(length=k1+1)
# c <- 0
   x <- 0
   M[1] <- 1
   M[2] <- 1-2*x/k1
   for (i in 1:length(M)) {
      M[i+2] <- (((2*i+1)*(k1-2*x)*M[i+1])-(i*(k1+i+1)*M[i]))/((i+1)*(k1-i))
      length(M) <- k1+1
      i <- i+1
   }
   m <- vector(length=k1+1)
# c <- goes from 1 to k1
   for (x in 1:k1) {
      m[1] <- 1
      m[2] <- 1-2*x/k1
```

```
      for (i in 1:length(m)) {
         m[i+2] <- (((2*i+1)*(k1-2*x)*m[i+1])-(i*(k1+i+1)*m[i]))/((i+1)*(k1-i))
         length(m) <- k1+1
         i <- i+1
      }
# combine to form a (k1+1)*(k1+1) matrix
      M <- rbind(M,m)
      next
      x <- x+1
   }
# compute Si(x, n) <- partial sum of fi(x, n)
   S <- M
   for(i in 1:(k1+1)) {
      S[,i] <- cumsum(S[,i])
      i <- i+1
   }
# compute Si(x, n)/||ei^2||
   for (i in 1:(k1+1)) {
      S[,i] <- S[,i]/(sum(M[,i]^2))
      i <- i+1
   }
# transpose S
   S <- t(S)
# compute fi(M, N)
   V <- vector(length=k1+1)
# n <- 0
   x <- 0
   V[1] <- 1
   V[2] <- 1-2*x/N
   for (i in 1:(k1+1)) {
      V[i+2] <- (((2*i+1)*(N-2*x)*V[i+1])-(i*(N+i+1)*V[i]))/((i+1)*(N-i))
      length(V) <- k1+1
      i <- i+1
   }
   v <- vector(length=k1+1)
# n <- goes from 1 to N
   for (x in 1:N) {
      v[1] <- 1
      v[2] <- 1-2*x/N
      for (i in 1:(k1+1)) {
         v[i+2] <- (((2*i+1)*(N-2*x)*v[i+1])-(i*(N+i+1)*v[i]))/((i+1)*(N-i))
         length(v) <- k1+1
         i <- i+1
```

```
      }
# combine to form a (N+1)*(k1+1) matrix
      V <- rbind(V,v)
      next
      x <- x+1
   }
# compute for cumulative hypergeometric probabilities
   X <- V %*% S
   Y <- matrix(data=1, nr=N+1, nc=k1+1)
   Z <- Y-X
   Z[Z <= 0.000001] <- 0
# output data to make a table containing the values of (n, c, alpha)
   i <- 1
   j <- 1
   z <- as.vector(c(n=i-1, c=j-1, Z[i,j]))
   u <- vector(length=3)
   for (i in 1:(N+1)) {
      for (j in 1:(k1+1)) {
         if (Z[i, j] <= a & Z[i, j] >= a.prime) {
            u[1] <- (i-1)
            u[2] <- (j-1)
            u[3] <- Z[i, j]
            z <- rbind(z, u)
         }
         j <- j+1
      }
      next
      i <- i+1
   }
   z <- as.matrix(z)
# compute the consumer's risk beta
# compute fi(x, n)
   MA <- vector(length=k2+1)
# c <- 0
   x <- 0
   MA[1] <- 1
   MA[2] <- 1-2*x/k2
   for (i in 1:length(MA)) {
      MA[i+2] <- (((2*i+1)*(k2-2*x)*MA[i+1])-(i*(k2+i+1)*MA[i]))/((i+1)*(k2-i))
      length(MA) <- k2+1
      i <- i+1
   }
   ma <- vector(length=k2+1)
```

```
# c <- goes from 1 to k2
   for (x in 1:k2) {
      ma[1] <- 1
      ma[2] <- 1-2*x/k2
      for (i in 1:length(ma)) {
         ma[i+2] <- (((2*i+1)*(k2-2*x)*ma[i+1])-(i*(k2+i+1)*ma[i]))/((i+1)*(k2-i))
         length(ma) <- k2+1
         i <- i+1
      }
# combine to form a (k2+1)*(k2+1) matrix
      MA <- rbind(MA,ma)
      next
      x <- x+1
   }
# compute Si(x, n) <- partial sum of fi(x, n)
   SA <- MA
   for(i in 1:(k2+1)) {
      SA[,i] <- (sum(SA[,i]))-(cumsum(SA[,i]))
      i <- i+1
   }
# compute Si(x, n)/||ei^2||
   for (i in 1:(k2+1)) {
      SA[,i] <- SA[,i]/(sum(MA[,i]^2))
      i <- i+1
   }
   SA <- t(SA)
# compute fi(M, N)
   VE <- vector(length=k2+1)
# n <- 0
   x <- 0
   VE[1] <- 1
   VE[2] <- 1-2*x/N
   for (i in 1:(k2+1)) {
      VE[i+2] <- (((2*i+1)*(N-2*x)*VE[i+1])-(i*(N+i+1)*VE[i]))/((i+1)*(N-i))
      length(VE) <- k2+1
      i <- i+1
   }
   ve <- vector(length=k2+1)
# n <- goes from 1 to N
   for (x in 1:N) {
      ve[1] <- 1
      ve[2] <- 1-2*x/N
      for (i in 1:(k2+1)) {
```

```
        ve[i+2] <- (((2*i+1)*(N-2*x)*ve[i+1])-(i*(N+i+1)*ve[i]))/((i+1)*(N-i))
        length(ve) <- k2+1
        i <- i+1
      }
# combine to form a (N+1)*(k2+1) matrix
      VE <- rbind(VE,ve)
      next
      x <- x+1
   }
# compute for cumulative hypergeometric probabilities
   XB <- VE %*% SA
   YB <- matrix(data=1, nr=N+1, nc=k2+1)
   ZB <- YB-XB
   ZB[ZB <= 0.000001] <- 0
# output data to make a table containing the values of (n, c, beta)
   i <- 1
   j <- 1
   zb <- as.vector(c(n=i-1, c=j-1, ZB[i,j]))
   ub <- vector(length=3)
   for (i in 1:(N+1)) {
      for (j in 1:(k2+1)) {
         if (ZB[i, j] <= b & ZB[i, j] >= b.prime) {
            ub[1] <- as.numeric(i-1)
            ub[2] <- as.numeric(j-1)
            ub[3] <- ZB[i, j]
            zb <- rbind(zb, ub)
         }
         j <- j+1
      }
      next
      i <- i+1
   }
   zb <- as.matrix(zb)
# combine the table to show the acceptance plan (k1, n, c, alpha,
k2, beta)
   sampling.plan.X <- vector(length=6)
   sampling.plan <- as.vector(c(N, k1, a, b, k1, k2))
   for (i in 1:(nrow(z))) {
      for (j in 1:(nrow(zb))) {
         if (z[i,1] == zb[j,1] & z[i,2] == zb[j,2]) {
            sampling.plan.X[1] <- z[i,1]
            sampling.plan.X[2] <- z[i,2]
            sampling.plan.X[3] <- z[i,3]
```

```
            sampling.plan.X[4] <- zb[j,3]
            sampling.plan.X[5] <- k1
            sampling.plan.X[6] <- k2
            sampling.plan <- rbind(sampling.plan, sampling.plan.X)
         }
         j <- j+1
      }
      next
      i <- i+1
   }
   sampling.plan <- sampling.plan[-1,]
   nrow <- nrow(sampling.plan)
   if (nrow >= 4) {
   sampling.plan <- sampling.plan[(-1):(-2),]
   colnames(sampling.plan) <- c("n", "c", "alpha", "beta", "k1", "k2")
   rownames(sampling.plan) <- c(paste("sampling.plan.", 1:(nrow-2), sep=""))}
   else if (nrow == 3) {
   sampling.plan <- sampling.plan[(-1):(-2),]
   sampling.plan <- c("n"=sampling.plan.X[1], "c"=sampling.plan.X[2],
   "alpha"=sampling.plan.X[3], "beta"=sampling.plan.X[4],
   "k1"=sampling.plan.X[5], "k2"=sampling.plan.X[6])}
   else {
   sampling.plan <- c("The ranges for alpha and/or beta are too narrow,
   please reselect them")}
# output table to R console
   return(sampling.plan)
}
```

B.2 Compute c given $(\alpha, \beta, k_1, k_2, n, N)$

This program is used for obtaining acceptance number c according to pre-specified producer's risk α, consumer's risk β, proportion defective k_1, and k_2, sample size n, and lot size N $(\alpha, \beta, k_1, k_2, n, N)$ for **hypergeometric sampling**. Here the sample size n is known.

```
hyper.fixedn <- function (a, b, k1, k2, n, N) {
# compute fi(x, n)
   M <- vector(length=k1+1)
# c <- 0
    x <- 0
```

```
   M[1] <- 1
   M[2] <- 1-2*x/k1
   for (i in 1:length(M)) {
      M[i+2] <- (((2*i+1)*(k1-2*x)*M[i+1])-(i*(k1+i+1)*M[i]))/((i+1)*(k1-i))
      length(M) <- k1+1
      i <- i+1
   }
   m <- vector(length=k1+1)
# c <- goes from 1 to k1
   for (x in 1:k1) {
      m[1] <- 1
      m[2] <- 1-2*x/k1
      for (i in 1:length(m)) {
         m[i+2] <- (((2*i+1)*(k1-2*x)*m[i+1])-(i*(k1+i+1)*m[i]))/((i+1)*(k1-i))
         length(m) <- k1+1
         i <- i+1
      }
# combine to form a (k1+1)*(k1+1) matrix
      M <- rbind(M,m)
      next
      x <- x+1
   }
# compute Si(x, n) <- partial sum of fi(x, n)
   S <- M
   for(i in 1:(k1+1)) {
      S[,i] <- cumsum(S[,i])
      i <- i+1
   }
# compute Si(x, n)/||ei^2||
   for (i in 1:(k1+1)) {
      S[,i] <- S[,i]/(sum(M[,i]^2))
      i <- i+1
   }
# transpose S
   S <- t(S)
# compute fi(M, N)
   V <- vector(length=k1+1)
# n <- 0
   x <- 0
   V[1] <- 1
   V[2] <- 1-2*x/N
   for (i in 1:(k1+1)) {
      V[i+2] <- (((2*i+1)*(N-2*x)*V[i+1])-(i*(N+i+1)*V[i]))/((i+1)*(N-i))
```

```
      length(V) <- k1+1
      i <- i+1
   }
   v <- vector(length=k1+1)
# n <- goes from 1 to N
   for (x in 1:N) {
      v[1] <- 1
      v[2] <- 1-2*x/N
      for (i in 1:(k1+1)) {
         v[i+2] <- (((2*i+1)*(N-2*x)*v[i+1])-(i*(N+i+1)*v[i]))/((i+1)*(N-i))
         length(v) <- k1+1
         i <- i+1
      }
# combine to form a (N+1)*(k1+1) matrix
      V <- rbind(V,v)
      next
      x <- x+1
   }
# compute for cumulative hypergeometric probabilities
   X <- V %*% S
   Y <- matrix(data=1, nr=N+1, nc=k1+1)
   Z <- Y-X
   Z[Z <= 0.000001] <- 0
# output data to make a table containing the values of (n, c, alpha)
   i <- 1
   j <- 1
   z <- as.vector(c(n=i-1, c=j-1, Z[i,j]))
   u <- vector(length=3)
   for (i in 1:(N+1)) {
      for (j in 1:(k1+1)) {
         if (Z[i, j] <= a & Z[i, j] > 0) {
            u[1] <- (i-1)
            u[2] <- (j-1)
            u[3] <- Z[i, j]
            z <- rbind(z, u)
         }
         j <- j+1
      }
      next
      i <- i+1
   }
# compute the consumer's risk beta # compute fi(x, n)
   MA <- vector(length=k2+1)
```

```
# c <- 0
   x <- 0
   MA[1] <- 1
   MA[2] <- 1-2*x/k2
   for (i in 1:length(MA)) {
      MA[i+2] <- (((2*i+1)*(k2-2*x)*MA[i+1])-(i*(k2+i+1)*MA[i]))/((i+1)*(k2-i))
      length(MA) <- k2+1
      i <- i+1
   }
   ma <- vector(length=k2+1)
# c <- goes from 1 to k2
   for (x in 1:k2) {
      ma[1] <- 1
      ma[2] <- 1-2*x/k2
      for (i in 1:length(ma)) {
         ma[i+2] <- (((2*i+1)*(k2-2*x)*ma[i+1])-(i*(k2+i+1)*ma[i]))/((i+1)*(k2-i))
         length(ma) <- k2+1
         i <- i+1
      }
# combine to form a (k2+1)*(k2+1) matrix
      MA <- rbind(MA,ma)
      next
      x <- x+1
   }
# compute Si(x, n) <- partial sum of fi(x, n)
   SA <- MA
   for(i in 1:(k2+1)) {
      SA[,i] <- (sum(SA[,i]))-(cumsum(SA[,i]))
      i <- i+1
   }
# compute Si(x, n)/||ei^2||
   for (i in 1:(k2+1)) {
      SA[,i] <- SA[,i]/(sum(MA[,i]^2))
      i <- i+1
   }
   SA <- t(SA)
# compute fi(M, N)
   VE <- vector(length=k2+1)
# n <- 0
   x <- 0
   VE[1] <- 1
   VE[2] <- 1-2*x/N
   for (i in 1:(k2+1)) {
```

```
      VE[i+2] <- (((2*i+1)*(N-2*x)*VE[i+1])-(i*(N+i+1)*VE[i]))/((i+1)*(N-i))
      length(VE) <- k2+1
      i <- i+1
   }
   ve <- vector(length=k2+1)
# n <- goes from 1 to N
   for (x in 1:N) {
      ve[1] <- 1
      ve[2] <- 1-2*x/N
      for (i in 1:(k2+1)) {
         ve[i+2] <- (((2*i+1)*(N-2*x)*ve[i+1])-(i*(N+i+1)*ve[i]))/((i+1)*(N-i))
         length(ve) <- k2+1
         i <- i+1
      }
# combine to form a (N+1)*(k2+1) matrix
      VE <- rbind(VE,ve)
      next
      x <- x+1
   }
# compute for cumulative hypergeometric probabilities
   XB <- VE %*% SA
   YB <- matrix(data=1, nr=N+1, nc=k2+1)
   ZB <- YB-XB
   ZB[ZB <= 0.000001] <- 0
# output data to make a table containing the values of (n, c, beta)
   i <- 1
   j <- 1
   zb <- as.vector(c(n=i-1, c=j-1, ZB[i,j]))
   ub <- vector(length=3)
   for (i in 1:(N+1)) {
      for (j in 1:(k2+1)) {
         if (ZB[i, j] <= b & ZB[i, j] > 0) {
            ub[1] <- as.numeric(i-1)
            ub[2] <- as.numeric(j-1)
            ub[3] <- ZB[i, j]
            zb <- rbind(zb, ub)
         }
         j <- j+1
      }
      next
      i <- i+1
   }
   z <- as.matrix(z)
```

```
   zb <- as.matrix(zb)
# combine the table to show the acceptance plan (k1, n, c, alpha,
k2, beta)
   plan1.0 <- vector(length=3)
   plan1 <- as.vector(c(n, k1, a))
   for (i in 1:(nrow(z))) {
      if (z[i,1] == n) {
         plan1.0[1] <- n
         plan1.0[2] <- z[i,2]
         plan1.0[3] <- z[i,3]
         plan1 <- rbind(plan1, plan1.0)
      }
      i <- i+1
   }
   plan2.0 <- vector(length=3)
   plan2 <- as.vector(c(n, k2, b))
   for (j in 1:(nrow(zb))) {
      if (zb[j,1] == n) {
         plan2.0[1] <- n
         plan2.0[2] <- zb[j,2]
         plan2.0[3] <- zb[j,3]
         plan2 <- rbind(plan2, plan2.0)
      }
      j <- j+1
   }
   plan1 <- as.matrix(plan1)
   plan2 <- as.matrix(plan2)
   sampling.plan.X <- vector(length=5)
   sampling.plan.P <- as.vector(c(n, k1, a, k2, b))
   for (i in 1:(nrow(plan1))) {
      for (j in 1:(nrow(plan2))) {
         if (plan1[i,2] == plan2[j,2]) {
            sampling.plan.X[1] <- n
            sampling.plan.X[2] <- plan1[i,2]
            sampling.plan.X[3] <- plan1[i,3]
            sampling.plan.X[4] <- k2
            sampling.plan.X[5] <- plan2[j,3]
            sampling.plan.P <- rbind(sampling.plan.P, sampling.plan.X)
         }
         j <- j+1
      }
      next
      i <- i+1
```

```
   }
   colnames(sampling.plan.P) <- c("n", "c", "alpha", "k2", "beta")
# output the result to R console
   return(sampling.plan.P)
}
```

B.3 Compute upper and lower confidence limits

This program is used to compute upper and lower confidence limits for the number of defective units in the lot (k) given sample size n, the number of observed defectives in the sample x, and lot size N (n, x, N).

```
hyper.CL <- function (UCL.prime, Upper.CL, Lower.CL, LCL.prime, n,
x, N) {
# compute fi(x, n)
   M <- vector(length=n+1)
# c <- 0
   c <- 0
   M[1] <- 1
   M[2] <- 1-2*c/n
   for (i in 1:length(M)) {
      M[i+2] <- (((2*i+1)*(n-2*c)*M[i+1])-(i*(n+i+1)*M[i]))/((i+1)*(n-i))
      length(M) <- n+1
      i <- i+1
   }
   m <- vector(length=n+1)
# c <- goes from 1 to n
   for (c in 1:n) {
      m[1] <- 1
      m[2] <- 1-2*c/n
      for (i in 1:length(m)) {
         m[i+2] <- (((2*i+1)*(n-2*c)*m[i+1])-(i*(n+i+1)*m[i]))/((i+1)*(n-i))
         length(m) <- n+1
         i <- i+1
      }
# combine to form a (n+1)*(n+1) matrix
      M <- rbind(M,m)
      next
      c <- c+1
   }
```

```
# compute Si(x, n) <- partial sum of fi(x, n) for obtaining upper
and lower CL
   SA <- M
   SB <- M
   for(i in 1:(n+1)) {
      SA[,i] <- cumsum(SA[,i])
      SB[,i] <- (sum(SB[,i]))-(cumsum(SB[,i]))
      i <- i+1
   }
# compute Si(x, n)/||ei^2||
   for (i in 1:(n+1)) {
      SA[,i] <- SA[,i]/(sum(M[,i]^2))
      SB[,i] <- SB[,i]/(sum(M[,i]^2))
      i <- i+1
   }
   SA <- t(SA)
   SB <- t(SB)
# compute fi(M, N)
   V <- vector(length=n+1)
# n <- 0
   c <- 0
   V[1] <- 1
   V[2] <- 1-2*c/N
   for (i in 1:(n+1)) {
      V[i+2] <- (((2*i+1)*(N-2*c)*V[i+1])-(i*(N+i+1)*V[i]))/((i+1)*(N-i))
      length(V) <- n+1
      i <- i+1
   }
   v <- vector(length=n+1)
# n <- goes from 1 to N
   for (c in 1:N) {
      v[1] <- 1
      v[2] <- 1-2*c/N
      for (i in 1:(n+1)) {
         v[i+2] <- (((2*i+1)*(N-2*c)*v[i+1])-(i*(N+i+1)*v[i]))/((i+1)*(N-i))
         length(v) <- n+1
         i <- i+1
      }
# combine to form a (N+1)*(n+1) matrix
      V <- rbind(V,v)
      next
      c <- c+1
   }
```

```
# compute for cumulative hypergeometric probabilities
# compute for upper confidence limits
   Y <- matrix(data=1, nr=N+1, nc=n+1)
   X.upper <- V %*% SA
   Z.upper <- Y-X.upper
   Z.upper[Z.upper <= 0.000001] <- 0
# compute for lower confidence limits
   X.lower <- V %*% SB
   Z.lower <- Y-X.lower
   Z.lower <- Y-Z.lower
   Z.lower[Z.lower <= 0.000001] <- 0
# combine to form a table to show the results
   i <- 1
   j <- 1
   Upper <- as.matrix(c((i-1), (j-1), Z.upper[i,j]))
   CL.Upper <- vector(length=3)
   for (i in 1:(N+1)) {
      if (Z.upper[i,x+1] <= UCL.prime  & Z.upper[i,x+1] >= Upper.CL) {
         CL.Upper[1] <- i-1
         CL.Upper[2] <- x
         CL.Upper[3] <- Z.upper[i,x+1]
         Upper <- cbind(Upper, CL.Upper)
      }
      i <- i+1
   }
   ncol.upper <- ncol(Upper)
   if (ncol.upper > 2) {
   Upper <- Upper[,-1]
   colnames(Upper) <- c(paste("Upper.", 1:((ncol.upper)-1), sep=""))}
   else if (ncol.upper == 2) {
   Upper <- Upper[,-1]}
   else {Upper <- c("The range for upper confidence limits is too narrow,
   please reselect it")}
   i <- 1
   j <- 1
   Lower <- as.matrix(c((i-1), j, Z.lower[i,j]))
   CL.Lower <- vector(length=3)
   for (i in 1:(N+1)) {
      if (Z.lower[i,x] <= Lower.CL & Z.lower[i,x] >= LCL.prime) {
         CL.Lower[1] <- i-1
         CL.Lower[2] <- x
         CL.Lower[3] <- Z.lower[i,x]
         Lower <- cbind(Lower, CL.Lower)
```

```
      }
      i <- i+1
   }
   ncol.lower <- ncol(Lower)
   if (ncol.lower > 2) {
   Lower <- Lower[,-1]
   colnames(Lower) <- c(paste("Lower.", 1:((ncol.lower)-1), sep=""))}
   else if (ncol.lower == 2) {
   Lower <- Lower[,-1]}
   else {Lower <- c("The range for lower confidence limits is too narrow,
   please reselect it")}
   hyper.CL <- cbind(Upper, Lower)
   hyper.CL <- t(hyper.CL)
   colnames(hyper.CL) <- c("k", "x", "Conf.Limits")
# output data to R console
   return(hyper.CL)
}
```

Appendix C

R Code for Binomial Sampling

C.1 Compute (n, c) given $(\alpha, \beta, p_1, p_2)$

This program is used to compute the sample size n and acceptance number c (n, c) given proportional defective p_1 and p_2, producer's risk α, consumer's risk β $(p_1, p_2, \alpha, \beta)$ for **Binomial sampling**.

```
binom <- function (n.prime, p1, p2, a, a.prime, b, b.prime) {
# compute for maximum sample size n
   binoma <- vector(length=3)
   binoma <- c(0, 0, 0)
   for (n in 2:(n.prime)) {
# compute fi(x, n)
      M <- vector(length=n+1)
# c <- 0
      x <- 0
      M[1] <- 1
      M[2] <- 1-2*x/n
      for (i in 1:length(M)){
         M[i+2] <- (((2*i+1)*(n-2*x)*M[i+1])-(i*(n+i+1)*M[i]))/((i+1)*(n-i))
         length(M) <- n+1
         i <- i+1
      }
      m <- vector(length=n+1)
# c <- goes from 1 to n
      for (x in 1:n) {
```

```
        m[1] <- 1
        m[2] <- 1-2*x/n
        for (i in 1:length(m)) {
           m[i+2] <- (((2*i+1)*(n-2*x)*m[i+1])-(i*(n+i+1)*m[i]))/((i+1)*(n-i))
           length(m) <- n+1
           i <- i+1
        }
# combine to form a (n+1)*(n+1) matrix
        M <- rbind(M,m)
        next
        x <- x+1
     }
# compute fi(x, n)/||ei^2||
     for (i in 1:(n+1)) {
        M[,i] <- M[,i]/(sum(M[,i]^2))
        i <- i+1
     }
     M <- t(M)
# compute Hi(p1)
     HA <- vector(length=n+1)
     HA[1] <- 1
     HA[2] <- 1-2*(p1)
     for (i in 1:(n+1)) {
        HA[i+2] <- (((2*i+1)*(1-2*(p1))*HA[i+1])-(i*HA[i]))/(i+1)
        length(HA) <- n+1
        i <- i+1
     }
     HA <- rbind(HA)
# obtain point probability of binomial distribution
     XA <- HA %*% M
     XA <- cumsum(XA)
     YA <- rep(1, (n+1))
     ZA <- YA-XA
     ZA[ZA <= 0.00001] <- 0
     binoma.0 <- vector(length=3)
     for (i in 1:(n+1)) {
        if (ZA[i] <= a & ZA[i] >= a.prime) {
           binoma.0[1] <- i-1
           binoma.0[2] <- n
           binoma.0[3] <- ZA[i]
           binoma <- rbind(binoma, binoma.0)
        }
        i <- i+1
```

```
      }
      next
      n < n+1
   }
   maxn <- max(binoma[,2])
# compute fi(x, n)
   MA <- vector(length=((maxn)+1))
# c <- 0
   x <- 0
   MA[1] <- 1
   MA[2] <- 1-((2*x)/(maxn))
   for (i in 1:length(MA)){
      MA[i+2] <- (((2*i+1)*((maxn)-2*x)*MA[i+1])-(i*((maxn)+i+1)*MA[i]))/
      ((i+1)*((maxn)-i))
      length(MA) <- (maxn)+1
      i <- i+1
   }
   ma <- vector(length=((maxn)+1))
# c <- goes from 1 to (maxn)
   for (x in 1:(maxn)) {
      ma[1] <- 1
      ma[2] <- 1-((2*x)/(maxn))
      for (i in 1:length(ma)) {
         ma[i+2] <- (((2*i+1)*((maxn)-2*x)*ma[i+1])-(i*((maxn)+i+1)*ma[i]))/
         ((i+1)*((maxn)-i))
         length(ma) <- (maxn)+1
         i <- i+1
      }
# combine to form a (n+1)*(n+1) matrix
      MA <- rbind(MA,ma)
      next
      x <- x+1
   }
# compute fi(x, n)/||ei^2||
   for (i in 1:((maxn)+1)) {
      MA[,i] <- MA[,i]/(sum(MA[,i]^2))
      i <- i+1
   }
   MA <- t(MA)
   HB <- vector(length=((maxn)+1))
   HB[1] <- 1
   HB[2] <- 1-2*(p2)
   for (i in 1:((maxn)+1)) {
```

```
      HB[i+2] <- (((2*i+1)*(1-2*(p2))*HB[i+1])-(i*HB[i]))/(i+1)
      length(HB) <- (maxn)+1
      i <- i+1
   }
   HB <- rbind(HB)
   XB <- HB %*% MA
   XB <- (sum(XB))-(cumsum(XB))
   YB <- rep(1, ((maxn)+1))
   ZB <- YB-XB
   ZB[ZB <= 0.0000001] <- 0
   binom <- vector(length=6)
   for (i in 1:(nrow(binoma))) {
      if (maxn == binoma[i,2]) {
         j <- binoma[i,1]
         binom[1] <- p1
         binom[2] <- p2
         binom[3] <- j
         binom[4] <- maxn
         binom[5] <- binoma[i,3]
         binom[6] <- ZB[j+1]
      }
      i <- i+1
   }
# compute for minimum sample size n according to c obtained in the
steps above
   binomb <- vector(length=3)
   binomb <- c(0, 0, 0)
   for (n in 2:(n.prime)) {
# compute fi(x, n)
      MB <- vector(length=n+1)
# c <- 0
      x <- 0
      MB[1] <- 1
      MB[2] <- 1-2*x/n
      for (i in 1:length(MB)){
         MB[i+2] <- (((2*i+1)*(n-2*x)*MB[i+1])-(i*(n+i+1)*MB[i]))/((i+1)*(n-i))
         length(MB) <- n+1
         i <- i+1
      }
      mb <- vector(length=n+1)
# c <- goes from 1 to n
      for (x in 1:n) {
         mb[1] <- 1
```

```
        mb[2] <- 1-2*x/n
        for (i in 1:length(mb)) {
           mb[i+2] <- (((2*i+1)*(n-2*x)*mb[i+1])-(i*(n+i+1)*mb[i]))/((i+1)*(n-i))
           length(mb) <- n+1
           i <- i+1
        }
# combine to form a (n+1)*(n+1) matrix
        MB <- rbind(MB,mb)
        next
        x <- x+1
     }
# compute fi(x, n)/||ei^2||
     for (i in 1:(n+1)) {
        MB[,i] <- MB[,i]/(sum(MB[,i]^2))
        i <- i+1
     }
     MB <- t(MB)
# compute Hi(p2)
     HB.B <- vector(length=n+1)
     HB.B[1] <- 1
     HB.B[2] <- 1-2*(p2)
     for (i in 1:(n+1)) {
        HB.B[i+2] <- (((2*i+1)*(1-2*(p2))*HB.B[i+1])-(i*HB.B[i]))/(i+1)
        length(HB.B) <- n+1
        i <- i+1
     }
     HB.B <- rbind(HB.B)
# obtain point probability of binomial distribution
     XB.B <- HB.B %*% MB
     XB.B <- (sum(XB.B))-(cumsum(XB.B))
     YB.B <- rep(1, (n+1))
     ZB.B <- YB.B - XB.B
     ZB.B[ZB.B <= 0.00001] <- 0
     binomb.0 <- vector(length=3)
     for (i in 1:(n+1)) {
        if (ZB.B[i] <= b & ZB.B[i] >= b.prime) {
           binomb.0[1] <- i-1
           binomb.0[2] <- n
           binomb.0[3] <- ZB.B[i]
           binomb <- rbind(binomb, binomb.0)
        }
        i <- i+1
     }
```

```
      next
      n <- n+1
   }
   for (i in 1:(nrow(binomb))) {
      if (binomb[i,1] == binom[3]) {
         minn <- min(binomb[i,2])
         binomb.b <- binomb[i,3]
      }
      i <- i+1
   }
# after the minimum n is found, compute for alpha'(real risk alpha)
# compute fi(x, n)
   MA.A <- vector(length=((minn)+1))
# c <- 0
   x <- 0
   MA.A[1] <- 1
   MA.A[2] <- 1-((2*x)/(minn))
   for (i in 1:length(MA.A)) {
      MA.A[i+2] <- (((2*i+1)*((minn)-2*x)*MA.A[i+1])-(i*((minn)+i+1)*MA.A[i]))/
      ((i+1)*((minn)-i))
      length(MA.A) <- (minn)+1
      i <- i+1
   }
   ma.a <- vector(length=((minn)+1))
# c <- goes from 1 to (minn)
   for (x in 1:(minn)) {
      ma.a[1] <- 1
      ma.a[2] <- 1-((2*x)/(minn))
      for (i in 1:length(ma.a)) {
         ma.a[i+2] <- (((2*i+1)*((minn)-2*x)*ma.a[i+1])-(i*((minn)+i+1)*ma.a[i]))/
         ((i+1)*((minn)-i))
         length(ma.a) <- (minn)+1
         i <- i+1
      }
# combine to form a (n+1)*(n+1) matrix
      MA.A <- rbind(MA.A,ma.a)
      next
      x <- x+1
   }
# compute fi(x, n)/||ei^2||
   for (i in 1:((minn)+1)) {
      MA.A[,i] <- MA.A[,i]/(sum(MA.A[,i]^2))
      i <- i+1
```

```
   }
   MA.A <- t(MA.A)
   HA.A <- vector(length=((minn)+1))
   HA.A[1] <- 1
   HA.A[2] <- 1-2*(p1)
   for (i in 1:((minn)+1)) {
      HA.A[i+2] <- (((2*i+1)*(1-2*(p1))*HA.A[i+1])-(i*HA.A[i]))/(i+1)
      length(HA.A) <- (minn)+1
      i <- i+1
   }
   HA.A <- rbind(HA.A)
   XA.A <- HA.A %*% MA.A
   XA.A <- (cumsum(XA.A))
   YA.A <- rep(1, ((minn)+1))
   ZA.A <- YA.A - XA.A
   ZA.A[ZA.A <= 0.0000001] <- 0
   binom.a <- vector(length=6)
   k <- binom[3]
   binom.a <- c(p1, p2, k, minn, ZA.A[k+1], binomb.b)
   binom <- rbind(binom, binom.a)
   colnames(binom) <- c("p1", "p2", "c", "n", "alpha", "beta")
   rownames(binom) <- c("sampling.plan.1", "sampling.plan.2")
   return(binom)
}
```

C.2 Draw OC curve given (n, c)

This program is used to draw Operating Characteristic (OC) curve given sample size n and acceptance number c (n, c) for **binomial sampling**.

```
binom.graph <- function (n, c, p.prime) {
   binomb <- vector(length=3)
   binomb <- c(0, 0, 0)
# compute fi(x, n)
   M <- vector(length=n+1)
# c <- 0
   x <- 0
   M[1] <- 1
   M[2] <- 1-2*x/n
   for (i in 1:length(M)){
```

```
      M[i+2] <- (((2*i+1)*(n-2*x)*M[i+1])-(i*(n+i+1)*M[i]))/((i+1)*(n-i))
      length(M) <- n+1
      i <- i+1
   }
   m <- vector(length=n+1)
# c <- goes from 1 to n
   for (x in 1:n) {
      m[1] <- 1
      m[2] <- 1-2*x/n
      for (i in 1:length(m)) {
         m[i+2] <- (((2*i+1)*(n-2*x)*m[i+1])-(i*(n+i+1)*m[i]))/((i+1)*(n-i))
         length(m) <- n+1
         i <- i+1
      }
# combine to form a (n+1)*(n+1) matrix
      M <- rbind(M,m)
      next
      x <- x+1
   }
# compute fi(x, n)/||ei^2||
   for (i in 1:(n+1)) {
      M[,i] <- M[,i]/(sum(M[,i]^2))
      i <- i+1
   }
   M <- t(M)
# compute Hi(p)
   H <- vector(length=n+1)
   p <- 0.01
   H[1] <- 1
   H[2] <- 1-2*(p)
      for (i in 1:(n+1)) {
      H[i+2] <- (((2*i+1)*(1-2*(p))*H[i+1])-(i*H[i]))/(i+1)
      length(H) <- n+1
      i <- i+1
   }
   X <- H %*% M
   X <- (sum(X))-(cumsum(X))
   Y <- rep(1, (n+1))
   Z <- Y-X
   Z[Z <= 0.0000001] <- 0
   HP <- vector(length=n+1)
   for (p in 2:(p.prime)) {
      HP[1] <- 1
```

```
      HP[2] <- 1-2*(p/100)
      for (i in 1:(n+1)) {
         HP[i+2] <- (((2*i+1)*(1-2*(p/100))*HP[i+1])-(i*HP[i]))/(i+1)
         length(HP) <- n+1
         XP <- HP %*% M
         XP <- (sum(XP))-(cumsum(XP))
         YP <- rep(1, (n+1))
         ZP <- YP - XP
         ZP[ZP <= 0.0000001] <- 0
         i <- i+1
      }
      Z <- rbind(Z, ZP)
      next
      p <- p+1
   }
   colnames(Z) <- c(0:n)
   rownames(Z) <- c(1:(p.prime))
# obtain point probability of binomial distribution
   par(col.axis="blue", col.main="red", mar=c(4, 4, 2.5, 0.25))
   plot(Z[,(c+1)], xlab="Proportional defective per lot (p)",
   ylab="The probability of acceptance (pa)", pch=21, col="red",
   bg="yellow", main="OC curve")
   lines(Z[,(c+1)])
}
```