McMaster University
DigitalCommons@McMaster

Open Access Dissertations and Theses | Open Dissertations and Theses

10-1-2011

SELECTING THE MOST PROBABLE CATEGORY: THE R PACKAGE RS

Hong Jin
McMaster University, jinh5@math.mcmaster.ca

Follow this and additional works at: http://digitalcommons.mcmaster.ca/opendissertations

Part of the Statistics and Probability Commons

Recommended Citation
Jin, Hong, "SELECTING THE MOST PROBABLE CATEGORY: THE R PACKAGE RS" (2011). *Open Access Dissertations and Theses.* Paper 6106.

This Thesis is brought to you for free and open access by the Open Dissertations and Theses at DigitalCommons@McMaster. It has been accepted for inclusion in Open Access Dissertations and Theses by an authorized administrator of DigitalCommons@McMaster. For more information, please contact scom@mcmaster.ca.

SELECTING THE MOST PROBABLE CATEGORY: THE R PACKAGE RS

SELECTING THE MOST PROBABLE CATEGORY: THE R PACKAGE RS

By

Hong Jin, B.Sc.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree

Master of Science

McMaster University

@Copyright by Hong Jin, August 2011

MASTER OF SCIENCE (2011) McMaster University

(Statistics) Hamilton, Ontario

TITLE:	SELECTING THE MOST PROBABLE CATEGORY: THE R PACKAGE RS
AUTHOR:	Hong Jin, B.Sc. (Nankai University)
SUPERVISOR:	Dr. Aaron Childs
NUMBER OF PAGES:	x, 86

Acknowledgements

First of all, I owe my deepest gratitude to Dr. Aaron Childs for his supervision, guidance and help from the very early stage as well as encouragement and support in various ways throughout the project. My thesis would not have been possible without his enthusiasm, inspiration, good teaching, sound advice and lots of good ideas. Furthermore, his financial support for this project is really appreciated.

Besides my supervisor, I would like to express my great gratitude to professors Shui Feng and Narayanaswamy Balakrishnan for deciding to be the committee members, and taking time to read my thesis. Their insightful comments, questions, and encouragement are really appreciated.

Many thanks to professor Fred M. Hoppe, Dr. Angelo Canty and Dr. Amadou Sarr for giving me help and inspiration in statistics.

I would also like to thank the Department of Mathematics and Statistics for allowing me to join this master's program.

I gratefully thank Feng Su, Jianyi Xu, Tracy Liu, Tao Wang, Xiaojun Zhu, Xiaodi Yang, Ashley Bonner, Sathish Pichika, Surva Pal, Debanjan Mitra, Li Zhang, Amidu Raifu, Kirill Mayorov, Melissa Song, and other friends for studying statistics with me and making my graduate study an extremely valuable experience.

Finally, I would like to express special thanks to my family for their understanding, love and support.

Abstract

Selecting the most probable multinomial or multivariate hypergeometric category is a multiple-decision selection problem. In this package, fixed sampling and inverse sampling are used for selecting the most probable category. This package aims at providing functionality to calculate, display and plot the probabilities of correctly selecting the most probable category under the least favorable configuration for these two sampling types. A function for finding the specified smallest acceptable sample size (or cell quota and expected sample size) is included as well.

Keywords: multinomial distribution, negative multinomial distribution, multivariate hypergeometric distribution, multivariate inverse hypergeometric distribution, fixed sampling, inverse sampling, least favorable configuration, most probable category

Contents

1 Introduction **1**

1.1 General Introduction 1

1.2 Outline 4

2 Calculation Details **5**

2.1 Calculation for Multinomial Distributions 5

2.1.1 Multinomial Distribution 6

2.1.2 Negative Multinomial Distribution 6

2.2 Calculation for Hypergeometric Distributions 7

2.2.1 Multivariate Hypergeometric Distribution 7

2.2.2 Multivariate Inverse Hypergeometric Distribution 8

2.3 The Least Favorable Configuration 9

2.4 Large-Sample Approximations to the Probabilities of Correct Selection 10

2.4.1 Multinomial Distribution 10

2.4.2 Multivariate Hypergeometric Distribution 12

2.4.3 Negative Multinomial Distribution 13

2.4.4 Multivariate Inverse Hypergeometric Distribution 14

2.5 Smallest Sample Size or Cell Quota to Meet the PCS Requirement . . . 15

2.5.1 Exact Smallest Sample Size or Cell Quota 15

2.5.2 Approximate Smallest Sample Size or Cell quota 16

2.6 Exact and Asymptotic Evaluation of Expected Waiting Time 18

3 The RS Package **21**

3.1 Object classes 21

3.2 Validation methods 24

3.3 Show and Summary methods 25

3.4 Plot method 26

3.5 Finding the smallest sample size or cell quota 26

4 Examples **28**

4.1 Basic sampling information 28

4.2 Calculation summary 29

4.3 Plotting the PCS curve 33

4.4 Finding the smallest sample size (or cell quota and expected sample size) 38

5 Discussion **41**

Appendices **42**

A RS Package Code **43**

A.1 LF2c.R 43

A.2 find.n.R 69

B RS Package Help File **78**

List of Tables

3.1 Arguments and corresponding sampling distributions for the `LF2c`class. 24

List of Figures

3.1 Class structure 23

4.1 PCS curve for multinomial distribution 34

4.2 PCS curve for fixed sampling and inverse sampling 35

4.3 Expected sample size for inverse sampling 37

4.4 Comparison of calculations based on the multinomial distribution. . . . 38

Chapter 1

Introduction

1.1 General Introduction

Selecting the most probable category is an example of a multiple-decision selection problem. The goal is to select the category or *cell* with the largest probability of occurrence in a population with $k \geq 2$ categories. For example, we might be interested in which of the six faces of a die has the largest probability of landing up, or we might want to find out which of the 5 candidates has the largest voter support in a region with 2,000 voters. For further examples one may refer to Bechofer et al. (1995), and Sobel and Frankowski (1977, 1985). In these cases, we need to specify $P^\star$ and $\theta^\star$ before experimentation to satisfy the requirement that the probability of correct selection is greater than or equal to $P^\star$ whenever the ratio between the largest and second largest of the population probabilities or category sizes is greater than or equal to $\theta^\star$.

Certain sampling procedures have been developed to solve this multiple-decision selection problem. A fixed sampling procedure was first introduced by Bechhofer et al. (1959) to select the most probable multinomial category. This problem has been further studied by Gupta and Nagel (1967), Panchapakesan (1971), and Gupta and Huang

(1975) using a subset selection approach. An inverse sampling sequential procedure was first considered by Cacoullos and Sobel (1966) for selecting the most probable category in a multinomial distribution. Other sequential procedures were given by Alam (1971), Alam et al. (1971), Ramey and Alam (1979, 1980), and Bechhofer and Kulkarni (1984).

This paper aims at describing a new R package to find the probabilities of correctly selecting the most probable category based on fixed sampling and inverse sampling using the least favorable configuration, which is defined as the population probabilities or category sizes which give the minimum probability of correct selection for a given $\theta^\star$.

In fixed sampling, a sample of size n is selected for inspection and hence a decision will be made according to the observations. Let Y_i be the number of observations in the sample that come from the i-th cell, for i=1,2,...,k, and let $y_{[1]} \leq \cdots \leq y_{[k]}$ denote their ranked values. The cell corresponding to the largest sample frequency is selected to be the most probable cell; if more than one of the cell frequencies are tied with $y_{[k]}$, then one is selected by randomization. In an inverse sampling procedure, observations are selected one at a time until we observe m observations from any one cell. The number m is called a *cell quota*. This cell is then chosen as the most probable cell.

For fixed sampling, all n observations can be inspected at the same time, while inverse sampling can inspect just one observation at a time. Thus, inverse sampling is not realistic when the inspection is time consuming. However, generally speaking, inverse sampling has a smaller expected total number of observations than the corresponding fixed sample size procedure with the same $P^\star$ and $\theta^\star$, which means that it is more advantageous when sampling is relatively expensive.

Sampling with or without replacement is provided for each type of sampling procedure, hence the type of distribution can be assumed as multinomial or multivariate hypergeometric respectively. When sampling is without replacement, the population size N needs to be specified.

As well, all the calculations need $\theta^\star$ which for the multinomial distribution is the smallest value of the ratio between the highest and second highest probabilities and for the hypergeometric distribution is the smallest value of the ratio between the largest and second largest category sizes. Considering a multinomial population with probability vector $\mathbf{P} = (p_1, \ldots, p_k)$ ($0 \leq p_i \leq 1$ and $\sum_{i=1}^{k} p_i = 1$) and corresponding ranked probabilities $p_{[1]} \leq p_{[2]} \cdots \leq p_{[k]}$, we have that $p_{[k]}/p_{[k-1]} \geq \theta^\star$. For a hypergeometric population of size N with M_i objects in the i-th category ($\sum_{i=1}^{k} M_i = N$) and their corresponding ranked values $M_{[1]} \leq M_{[2]} \cdots \leq M_{[k]}$, the condition is that $M_{[k]}/M_{[k-1]} \geq \theta^\star$. In addition, we provide the option for two types of calculation for each type of sampling and assumed distribution. One is `exact`, which uses exact probabilities and the other is `approximate`, which uses a multivariate normal approximation. For the multinomial and multivariate hypergeometric distributions, the approximate calculation is based on the normal approximation with an arcsin transformation; for the negative multinomial and multivariate inverse hypergeometric distributions, an untransformed normal approximation is used.

After determining all of the necessary conditions, the probability of correctly selecting the most probable category based on the least favorable configuration can be calculated using the `LF2c` function. The PCS curve (probability of correct selection against $\theta^\star$) can then be plotted (e.g. Figures 4.1, 4.2, and 4.4).

It is obvious that when n or m increases, the probability of correct selection will increase. Thus, the user can increase the sample size n (in fixed sampling) or cell quota m (in inverse sampling) in order to get a higher probability of correct selection.

This new package also includes a function which allows the user to find the smallest sample size using fixed sampling or the smallest cell quota and expected sample size using inverse sampling such that a specified probability of correct selection is met, for a given $\theta^\star$.

For the theoretical details of this package the reader is referred to Bechhofer et al. (1959) and Cacoullos and Sobel (1966). For more details about ranking and selection

the reader is referred to Gupta and Panchapakesan (1979) and Santner and Duffy (1989).

1.2 Outline

Chapter 1 gives a general introduction to the new package **RS**. In Chapter 2, we first introduce the details of the exact calculations of probabilities of correct selection based on the multinomial and hypergeometric distributions. Then we give expressions for the least favorable configuration for these two types of distribution. Large-sample approximations to the probabilities of correct selection, smallest sample size or cell quota to meet the $P^\star$ requirement, and exact and asymptotic evaluation of expected waiting time are also discussed in Chapter 2.

We then describe the implementation of the **RS** package in Chapter 3 and give examples of its use in Chapter 4. Finally, in Chapter 5, we briefly discuss the restrictions of the package and possible areas for improvement. The R code for the package is given in Appendix A, and all of the help documents for the package can be found in Appendix B.

Chapter 2

Calculation Details

In this new R package, calculation of the probabilities of correctly selecting the most probable category are based on two major distributions, the multinomial distribution and multivariate hypergeometric distribution. And the calculations differ for the different sampling procedures: for fixed sampling, they are based on the multinomial and multivariate hypergeometric distributions; for inverse sampling, the negative multinomial and multivariate inverse hypergeometric distributions are used.

2.1 Calculation for Multinomial Distributions

Consider a multinomial population with unknown probability vector $\mathbf{p}$=(p_1,...,p_k) ($0 \leq p_i \leq 1$ and $\sum_{i=1}^{k} p_i = 1$). Let $p_{[1]} \leq p_{[2]} \cdots \leq p_{[k]}$ be the ranked probabilities. The cell associated with $p_{[k]}$ is called the most probable cell. Let $Y_{(i)}$ denote the number of observations in the sample which are associated with the category having probability $p_{[i]}$ $(i = 1, 2, \ldots, k)$.

2.1.1 Multinomial Distribution

Suppose n observations are selected. Bechhofer et al. (1959) introduced the probability of correct selection pcs, which can be calculated as

$$pcs = \sum_{y_{(i)} \leq y_{(k)}} \frac{1}{s} P(Y_{(1)} = y_{(1)}, \cdots, Y_{(k)} = y_{(k)}), \tag{2.1}$$

where $i = 1, 2, \ldots, k-1$, $\sum_{i=1}^{k} y_{(i)} = n$, and s is the number of $y_{(i)}$'s tied with $y_{(k)}$. (Note: for fixed sampling, if more than one of the cell frequencies are tied with $y_{[k]}$, then one is selected by randomization. So we have $1/s$ probability to correctly choose the cell with $y_{(k)}$ observations as the most probable cell.)

Since the probability mass function of the multinomial distribution is given by

$$f(y_{(1)}, \cdots, y_{(k)}; p_{[1]}, \cdots, p_{[k]}) = \frac{n!}{y_{(1)}! \cdots y_{(k)}!} p_{[1]}^{y_{(1)}} \cdots p_{[k]}^{y_{(k)}},$$

the exact probability of a correct selection is then given by

$$pcs = \sum_{y_{(i)} \leq y_{(k)}} \frac{1}{s} \cdot \frac{n!}{y_{(1)}! \cdots y_{(k)}!} p_{[1]}^{y_{(1)}} \cdots p_{[k]}^{y_{(k)}}.$$

2.1.2 Negative Multinomial Distribution

Suppose that we continue to sample until any one of the cells reaches m observations. Then the probability of correct selection is the probability that $Y_{(k)}$ reaches m before the other $Y_{(i)}$'s $(i < k)$. Cacoullos and Sobel (1966) calculated the probability of correct selection as

$$pcs = P(Y_{(1)} < m, \cdots, Y_{(k-1)} < m, \text{ when } Y_{(k)} \text{ reaches } m \text{ for the first time}). \tag{2.2}$$

Consider the time just prior to $Y_{(k)}$ reaching m observations. At that time, category

i has $y_{(i)}$ observations for $i = 1, \ldots, k-1$, and category k has $m-1$ observations. The probability of this event is $\frac{(\sum_{i=1}^{k-1} y_{(i)}+m-1)!}{(m-1)!y_{(1)}!\cdots y_{(k-1)}!} p_{[1]}^{y_{(1)}} \cdots p_{[k-1]}^{y_{(k-1)}} p_{[k]}^{m-1}$. And the probability that the next observation comes from the k-th category is $p_{[k]}$. Therefore,

$$\begin{aligned} P(Y_{(1)} = y_{(1)}, \ldots, Y_{(k-1)} = y_{(k-1)}, & \text{ when } Y_{(k)} \text{ reaches } m \text{ for the first time}) \\ &= \frac{(\sum_{i=1}^{k-1} y_{(i)} + m - 1)!}{(m-1)!y_{(1)}! \cdots y_{(k-1)}!} p_{[1]}^{y_{(1)}} \cdots p_{[k-1]}^{y_{(k-1)}} p_{[k]}^{m}, \end{aligned}$$

where $Y_{(1)}, \ldots, Y_{(k-1)}$ can be considered as negative multinomial random variables. Thus, the exact *pcs* is therefore given by

$$pcs = \sum_{y_{(i)} \leq m-1} \frac{(\sum_{i=1}^{k-1} y_{(i)} + m - 1)!}{(m-1)!y_{(1)}! \cdots y_{(k-1)}!} p_{[1]}^{y_{(1)}} \cdots p_{[k-1]}^{y_{(k-1)}} p_{[k]}^{m},$$

where $i = 1, 2, \ldots, k-1$.

2.2 Calculation for Hypergeometric Distributions

Consider a multivariate hypergeometric population of size N with M_i objects in the i-th category ($\sum_{i=1}^{k} M_i = N$). Let $M_{[1]} \leq M_{[2]} \cdots \leq M_{[k]}$ be the ranked values of M_i ($i = 1, 2, \ldots, k$). The cell associated with $M_{[k]}$ is called the most probable category. Let $Y_{(i)}$ denote the number of observations in the sample which are associated with $M_{[i]}$ ($i = 1, 2, \ldots, k$), so $Y_{(i)} \leq M_{[i]}$.

2.2.1 Multivariate Hypergeometric Distribution

Suppose n observations are selected. Then the probability of correct selection *pcs* is also given by (2.1). Since the observations are from a multivariate hypergeometric

population with probability mass function given by

$$f(y_{(1)}, \cdots, y_{(k)}; M_{[1]}, \cdots, M_{[k]})] = \frac{\binom{M_{[1]}}{y_{(1)}} \cdots \binom{M_{[k]}}{y_{(k)}}}{\binom{N}{n}},$$

where $y_{(i)} \leq M_{(i)}$ $(i = 1, 2, \ldots, k)$, the probability of correct selection can be obtained as follows

$$pcs = \sum_{y_{(i)} \leq y_{(k)}} \frac{1}{s} \cdot \frac{\binom{M_{[1]}}{y_{(1)}} \cdots \binom{M_{[k]}}{y_{(k)}}}{\binom{N}{n}},$$

where $\sum_{i=1}^{k} y_{(i)} = n$ and s is the number of $y_{(i)}$'s tied with $y_{(k)}$..

2.2.2 Multivariate Inverse Hypergeometric Distribution

Suppose that we continue to sample until any one of the categories reaches m observations. Then the probability of correct selection is the probability that $Y_{(k)}$ reaches m before the other $Y_{(i)}$'s $(i < k)$. The probability of correct selection is again given by (2.2). Consider the time just prior to $Y_{(k)}$ reaching m observations $(m \leq M_{[k]})$. At that time, category i has $y_{(i)}$ observations for $i = 1, \ldots, k-1$, and category k has $m-1$ observations. The probability of this event is $\binom{M_{[1]}}{y_{(1)}} \cdots \binom{M_{[k-1]}}{y_{(k-1)}} \binom{M_{[k]}}{m-1} / \binom{N}{m-1+\sum_{i=1}^{k-1} y_{(i)}}$. And the probability that the next observation comes from the k-th category is $\frac{M_{[k]}-m+1}{N-m+1-\sum_{i=1}^{k-1} y_{(i)}}$. Therefore,

$$\begin{aligned} P(Y_{(1)} = y_{(1)}, \ldots, Y_{(k-1)} = y_{(k-1)}, &\text{ when } Y_{(k)} \text{ reaches } m \text{ for the first time}) \\ &= \frac{\binom{M_{[1]}}{y_{(1)}} \cdots \binom{M_{[k-1]}}{y_{(k-1)}} \binom{M_{[k]}}{m-1}}{\binom{N}{m-1+\sum_{i=1}^{k-1} y_{(i)}}} \frac{M_{[k]} - m + 1}{N - m + 1 - \sum_{i=1}^{k-1} y_{(i)}} \end{aligned}$$

where $y_{(i)} \leq M_{(i)}$ $(i = 1, 2, \ldots, k-1)$, $m \leq M_{[k]}$, and $Y_{(1)}, \ldots, Y_{(k-1)}$ can be considered as multivariate inverse hypergeometric random variables.

So the probability of correct selection can be obtained as follows,

$$pcs = \sum_{y_{(i)} \leq min(m-1, M_{[i]})} \frac{\binom{M_{[1]}}{y_{(1)}} \cdots \binom{M_{[k-1]}}{y_{(k-1)}} \binom{M_{[k]}}{m-1}}{\binom{N}{m-1+\sum_{i=1}^{k-1} y_{(i)}}} \frac{M_{[k]} - m + 1}{N - m + 1 - \sum_{i=1}^{k-1} y_{(i)}}.$$

.

2.3 The Least Favorable Configuration

The least favorable configuration of the $p_{[i]}$'s from a multinomial or negative multinomial distribution is defined to be those values which minimize the probability of a correct selection when $p_{[k]}/p_{[k-1]} \geq \theta^\star$ for any given k, $\theta^\star$, and n or m. It is proved by Kesten and Morse (1959) that the least favorable configuration is independent of n or m and is given by $p_{[k]}/p_{[i]} = \theta^\star$ $(i = 1, 2, \ldots, k-1)$. Since $\sum_{i=1}^{k} p_i = 1$, the least favorable configuration is then given by

$$p_{[1]} = \cdots = p_{[k-1]} = \frac{1}{\theta^\star + k - 1} = q_0, \tag{2.3}$$

and

$$p_{[k]} = \frac{\theta^\star}{\theta^\star + k - 1} = p_0. \tag{2.4}$$

Similarly, we define the least favorable configuration of the $M_{[i]}$'s from a multivariate hypergeometric or multivariate inverse hypergeometric distribution as the $M_{[i]}$'s which for any given N, k, $\theta^\star$, and n or m, minimize the probability of a correct selection when $M_{[k]}/M_{[k-1]} \geq \theta^\star$. For the hypergeometric distributions, it is a little different from the multinomial distributions since the $M_{[i]}$'s need to be integers, and there is no specified formula for the least favorable configuration. But in this case, we can calculate the probability of correct selection for all possible configurations of the $M_{[i]}$'s for given N, k and $\theta^\star$. Then the corresponding $M_{[i]}$'s when pcs reaches its minimum is defined as the least favorable configuration of the $M_{[i]}$'s.

2.4 Large-Sample Approximations to the Probabilities of Correct Selection

We have tried the normal approximation with arcsin transformation and log transformation for calculating the approximate *pcs* as well as the untransformed normal approximation. Among the three methods, we find that the normal approximation with an arcsin transformation is the best for the multinomial and multivariate hypergeometric distributions, but for the negative multinomial and multivariate inverse hypergeometric distributions, an untransformed normal approximation is better than the other two.

2.4.1 Multinomial Distribution

Bechhofer et al. (1959) developed a normal approximation to (2.1) as

$$pcs \approx P(Y_{(k)} \geq Y_{(1)}, \cdots, Y_{(k)} \geq Y_{(k-1)}) \tag{2.5}$$

by considering the variables

$$Z_i = 2\arcsin\sqrt{\frac{Y_{(k)}}{n}} - 2\arcsin\sqrt{\frac{Y_{(i)}}{n}} \ (i = 1, 2, \cdots, k-1) \tag{2.6}$$

Then (2.5) can be written as

$$pcs \approx P\left(2\arcsin\sqrt{\frac{Y_{(k)}}{n}} \geq 2\arcsin\sqrt{\frac{Y_{(i)}}{n}}; i \neq k\right) = P(Z_1 \geq 0, \cdots, Z_{k-1} \geq 0). \tag{2.7}$$

The Z_i's $(i = 1, 2, \ldots, k-1)$ have a $(k-1)$-variate distribution which can be approximated by a multivariate normal distribution. Since it's hard to directly find the means, variances and covariances, we first expand $2\arcsin(Y_{(i)}/n)^{1/2}$ in a Taylor series around

$p_{[i]}$ as a function of $Y_{(i)}/n$ as follows,

$$2\arcsin\sqrt{\frac{Y_{(i)}}{n}} = f(p_{[i]}) + f'(p_{[i]})(\frac{Y_{(i)}}{n} - p_{[i]}) + O(\frac{1}{n^2}) \tag{2.8}$$

where

$$f(p_{[i]}) = 2\arcsin\sqrt{p_{[i]}} \tag{2.9}$$

$$f'(p_{[i]}) = \frac{1}{\sqrt{p_{[i]}(1-p_{[i]})}}. \tag{2.10}$$

Johnson et al. (1997) gave the expected values, variances and correlations of Y_i's $(i = 1, 2, \ldots, k)$ as

$$E(Y_i) = np_{[i]},$$
$$Var(Y_i) = np_{[i]}(1 - p_{[i]}),$$

and

$$Corr(Y_i, Y_j) = -\sqrt{\frac{p_{[i]}p_{[j]}}{(1-p_{[i]})(1-p_{[j]})}},$$

where $i = 1, 2, \ldots, k$; $j = 1, 2, \ldots, k$; and $i \neq j$. Then,

$$E(Z_i) = f(p_{[k]}) - f(p_{[i]}) + O(\frac{1}{n^2}) \ (i = 1, 2, \ldots, k-1)$$

and we approximate $E(Z_i)$ by

$$E(Z_i) \approx f(p_{[k]}) - f(p_{[i]}) \ (i = 1, 2, \ldots, k-1). \tag{2.11}$$

In the same way, $Var(Z_i)$ and $Corr_{i\neq j}(Z_i, Z_j)$ can be approximated as follows:

$$Var(Z_i) \approx (f'(p_{[k]})^2 p_{[k]} + f'(p_{[i]})^2 p_{[i]} - (f'(p_{[k]})p_{[k]} - f'(p_{[i]})p_{[i]})^2)/n, \tag{2.12}$$

and

$$Corr_{i\neq j}(Z_i, Z_j) \approx \frac{f'(p_{[k]})^2 p_{[k]} - (f'(p_{[k]})p_{[k]} - f'(p_{[i]})p_{[i]})^2}{\sqrt{Var(Z_i)}\sqrt{Var(Z_j)}}, \tag{2.13}$$

where $i = 1, 2, \ldots, k-1$ and $j = 1, \cdots, k-1$.

The function `pmvnorm` in the R package **mvtnorm** is used here to calculate (2.7) for given means, variances, and correlation matrix. It will also be used to calculate the approximate *pcs* for the other three distributions.

2.4.2 Multivariate Hypergeometric Distribution

As with the multinomial distribution, the probability of correct selection can also be approximated as $P(Z_1 \geq 0, \cdots, Z_{k-1} \geq 0)$ by considering the variables in (2.6).

In this case, we expand $2\arcsin(Y_{(i)}/n)^{1/2}$ in a Taylor series around $M_{[i]}/N$ as a function of $Y_{(i)}/n$. In this case, Johnson et al. (1997) gave the expected values, variances and covariances of Y_i's $(i = 1, 2, \ldots, k)$ as

$$E(Y_i) = \frac{nM_{[i]}}{N},$$

$$Var(Y_i) = \frac{nM_{[i]}}{N}\left(1 - \frac{M_{[i]}}{N}\right)\frac{N-n}{N-1},$$

and

$$Cov(Y_i, Y_j) = -\frac{nM_{[i]}M_{[j]}}{N^2}\frac{N-n}{N-1},$$

where $i = 1, 2, \ldots, k$; $j = 1, 2, \ldots, k$; and $i \neq j$.

Accordingly, $E(Z_i), Var(Z_i), Cov(Z_i, Z_j)_{i \neq j}$ can be approximated by

$$E(Z_i) \approx f(\frac{M_{[k]}}{N}) - f(\frac{M_{[i]}}{N}),$$

$$\begin{aligned} Var(Z_i) \quad &\approx f'(\tfrac{M_{[k]}}{N})^2 \tfrac{M_{[k]}}{N}\left(1 - \tfrac{M_{[k]}}{N}\right)\tfrac{N-n}{(N-1)n} \\ &+ f'(\tfrac{M_{[i]}}{N})^2 \tfrac{M_{[i]}}{N}\left(1 - \tfrac{M_{[i]}}{N}\right)\tfrac{N-n}{(N-1)n} \\ &+ 2f'(\tfrac{M_{[i]}}{N})f'(\tfrac{M_{[k]}}{N})\tfrac{M_{[i]}M_{[k]}}{nN^2}\tfrac{N-n}{N-1}, \end{aligned}$$

and

$$\begin{aligned} Cov(Z_i, Z_j)_{i\neq j} \approx\ & f'(\tfrac{M_{[k]}}{N})^2 \tfrac{M_{[k]}}{N}\left(1-\tfrac{M_{[k]}}{N}\right)\tfrac{N-n}{(N-1)n} \\ & -f'(\tfrac{M_{[i]}}{N})f'(\tfrac{M_{[j]}}{N})\tfrac{M_{[i]}M_{[j]}}{nN^2}\tfrac{N-n}{N-1} \\ & +f'(\tfrac{M_{[i]}}{N})f'(\tfrac{M_{[k]}}{N})\tfrac{M_{[i]}M_{[k]}}{nN^2}\tfrac{N-n}{N-1} \\ & +f'(\tfrac{M_{[j]}}{N})f'(\tfrac{M_{[k]}}{N})\tfrac{M_{[j]}M_{[k]}}{nN^2}\tfrac{N-n}{N-1}, \end{aligned}$$

where $i = 1, 2, \ldots, k-1$; $j = 1, 2, \ldots, k-1$; and $i \neq j$.

Thus, $Corr_{i\neq j}(Z_i, Z_j)$ can be obtained as

$$Corr_{i\neq j}(Z_i, Z_j) = \frac{Cov(Z_i, Z_j)_{i\neq j}}{\sqrt{Var(Z_i)}\sqrt{Var(Z_i)}}.$$

Then *pcs* can be approximated by calculating $P(Z_1 \geq 0, \cdots, Z_{k-1} \geq 0)$ using the corresponding multivariate normal distribution.

2.4.3 Negative Multinomial Distribution

Unlike the multinomial and multivariate hypergeometric distributions, we found that with inverse sampling an untransformed normal approximation gave more accurate results. Note that Cacoullos and Sobel (1966) also used an untransformed multivariate normal approximation in this case. Here we consider $Z_i = Y_{(i)}/m$ $(i = 1, 2, \ldots, k-1)$, so

$$pcs = P(Y_{(1)} < m, \cdots, Y_{(k-1)} < m) = P(Z_1 < 1, \cdots, Z_{k-1} < 1).$$

And $E(Y_i), Var(Y_i), Corr_{i\neq j}(Y_i, Y_j)$ are given by Le Gall (2006) as follows

$$E(Y_i) = \frac{mP_{[i]}}{p_{[k]}},$$

$$Var(Y_i) = \frac{mP_{[i]}(P_{[i]} + P_{[k]})}{p_{[k]}^2},$$

and

$$Corr_{i\neq j}(Y_i, Y_j) = \sqrt{\frac{P_{[i]}P_{[j]}}{(P_{[i]} + P_{[k]})(P_{[j]} + P_{[k]})}},$$

where ($i = 1, 2, \ldots, k-1$; $j = 1, 2, \ldots, k-1$; and $i \neq j$). So

$$E(Z_i) = \frac{P_{[i]}}{p_{[k]}}, \tag{2.14}$$

$$Var(Z_i) = \frac{P_{[i]}(P_{[i]} + P_{[k]})}{mp_{[k]}^2}, \tag{2.15}$$

and

$$Corr_{i\neq j}(Z_i, Z_j) = Corr_{i\neq j}(Y_i, Y_j). \tag{2.16}$$

Thus, we can calculate *pcs* using a $(k-1)$-variate multivariate normal distribution with the above means, variances, and correlation matrix.

2.4.4 Multivariate Inverse Hypergeometric Distribution

As with the negative multinomial distribution, we also consider $Z_i = Y_{(i)}/m$ ($i = 1, 2, \ldots, k-1$), so

$$pcs = P(Z_1 < 1, \cdots, Z_{k-1} < 1).$$

In this case, Janardan and Patil (1972) developed the expressions of $E(Y_i)$, $Var(Y_i)$, $Corr_{i\neq j}(Y_i, Y_j)$ as follows,

$$E(Y_i) = \frac{mM_{[i]}}{M_{[k]} + 1} \ (i = 1, 2, \cdots, k-1),$$

$$Var(Y_i) = \frac{mM_{[i]}}{M_{[k]} + 1}\left(1 + \frac{M_{[i]}}{M_{[k]} + 1}\right)\left(\frac{M_{[k]} - m + 1}{M_{[k]} + 2}\right) \ (i = 1, 2, \cdots, k-1),$$

and

$$Corr_{i\neq j}(Y_i, Y_j) = \sqrt{\frac{M_{[i]}M_{[j]}}{(M_{[i]} + M_{[k]} + 1)(M_{[j]} + M_{[k]} + 1)}} \; (i = 1, 2, \ldots, k-1; j = 1, \cdots, k-1).$$

So

$$E(Z_i) = \frac{M_{[i]}}{M_{[k]} + 1}, \tag{2.17}$$

$$Var(Z_i) = \frac{M_{[i]}}{m(M_{[k]} + 1)} \left(1 + \frac{M_{[i]}}{M_{[k]} + 1}\right) \left(\frac{M_{[k]} - m + 1}{M_{[k]} + 2}\right), \tag{2.18}$$

and

$$Corr_{i\neq j}(Z_i, Z_j) = Corr_{i\neq j}(Y_i, Y_j). \tag{2.19}$$

Thus, *pcs* can be approximated by using the multivariate normal distribution with the corresponding means, variances and correlation matrix.

2.5 Smallest Sample Size or Cell Quota to Meet the PCS Requirement

After introducing the exact and approximate calculations of probabilities of correct selection based on the least favorable configuration, we would like show how to determine proper values of the sample size or cell quota to meet the requirement that $pcs \geq P^\star$.

2.5.1 Exact Smallest Sample Size or Cell Quota

The exact smallest sample size and cell quota can be found through trial starting with n=1 or m=1 and increasing n or m until the condition, which is $pcs \geq P^\star$, is satisfied.

2.5.2 Approximate Smallest Sample Size or Cell quota

For approximate calculations, n or m can also be found when $pcs \geq P^\star$ is met. For the multinomial distribution, pcs is given by (2.7) with corresponding means, variances and correlations. The least favorable configuration is given by (2.3) and (2.4), so the means (2.11), or variances (2.12) are identical, say, $E(Z)$ and $Var(Z)$ respectively. Then we have that

$$pcs \approx \int_{-\frac{E(Z_1)}{\sqrt{Var(Z_1)}}}^{\infty} \cdots \int_{-\frac{E(Z_{k-1})}{\sqrt{Var(Z_{k-1})}}}^{\infty} g(t_1, \cdots, t_{k-1}) dt_1 \cdots d_{k-1} \tag{2.20}$$

$$= \int_{-\frac{E(Z)}{\sqrt{Var(Z)}}}^{\infty} \cdots \int_{-\frac{E(Z)}{\sqrt{Var(Z)}}}^{\infty} g(t_1, \cdots, t_{k-1}) dt_1 \cdots d_{k-1} \tag{2.21}$$

$$= P_{k-1}\left(-\frac{E(Z)}{\sqrt{Var(Z)}}\right). \tag{2.22}$$

where $g(t_1, \cdots, t_{k-1})$ is the (k-1)-variate normal density function with zero means, unit variances and correlation matrix (2.13). Let C satisfy $1 - P_{k-1}(C) = P^\star$. The function `qmvnorm` in the R package **mvtnorm** is used here to calculate C for the correlation matrix (2.13) and $P^\star$. By (2.22), we have that

$$C = -\frac{E(Z)}{\sqrt{Var(Z)}}$$

By (2.3), (2.4), (2.11) and (2.12),

$$C = -\frac{f(p_0) - f(q_0)}{\sqrt{(f'(p_0)^2 p_0 + f'(q_0)^2 q_0 - (f'(p_0)q_0 - f'(q_0)q_0)^2)/n^\star}} = \frac{A}{\sqrt{B/n^\star}}$$

Thus, $n^\star = BC^2/A^2$. And the smallest sample size is the smallest integer no less than $n^\star$.

For the negative multinomial distribution, the smallest cell quota which satisfies

$$pcs = P(Z_1 < 1, \cdots, Z_{k-1} < 1) \geq P^\star$$

can be obtained in the same way as the multinomial distribution. The least favorable configuration is also given by (2.3) and (2.4), and the means (2.14), or variances (2.15) are identical, say, $E(Z)$ and $Var(Z)$ respectively. Then pcs can be approximated by

$$\begin{aligned} pcs &\approx \int_{-\infty}^{\frac{1-E(Z_1)}{\sqrt{Var(Z_1)}}} \cdots \int_{-\infty}^{\frac{1-E(Z_{k-1})}{\sqrt{Var(Z_{k-1})}}} g(t_1, \cdots, t_{k-1}) dt_1 \cdots d_{k-1} && (2.23) \\ &= \int_{-\infty}^{\frac{1-E(Z)}{\sqrt{Var(Z)}}} \cdots \int_{-\infty}^{\frac{1-E(Z)}{\sqrt{Var(Z)}}} g(t_1, \cdots, t_{k-1}) dt_1 \cdots d_{k-1} && (2.24) \\ &= Q_{k-1}\left(\frac{1-E(Z)}{\sqrt{Var(Z)}}\right). && (2.25) \end{aligned}$$

where $g(t_1, \cdots, t_{k-1})$ is the (k-1)-variate normal density function with zero means, unit variance and correlation matrix (2.16). Let D satisfy $Q_{k-1}(D) = P^\star$. The function `qmvnorm` in the R package **mvtnorm** is used here to calculate D. By (2.25), we have that

$$D = \frac{1-E(Z)}{\sqrt{Var(Z)}}$$

By (2.3), (2.4), (2.14) and (2.15),

$$D = \left(1 - \frac{q_0}{p_0}\right) / \left(\sqrt{\frac{q_0(q_0+p_0)}{m^\star p_0^2}}\right)$$

Then, $m^\star$ can be obtained by

$$m^\star = \frac{D^2 q_0(q_0+p_0)}{(p_0-q_0)^2},$$

and the smallest cell quota is the smallest integer no less than $m^\star$.

As for the multivariate hypergeometric and multivariate inverse hypergeometric distributions, we cannot give specified formulas because the least favorable configuration is not known in general when the population is finite. But the calculation can be done in the same way as calculating the exact smallest sample size and cell quota, which is through trial starting with n=1 or m=1 and increasing n or m until $pcs \geq P^{\star}$ satisfied.

2.6 Exact and Asymptotic Evaluation of Expected Waiting Time

Using inverse sampling, the number of observations becomes a random variable. Thus, the expected sample size or waiting time $E(WT)$ can be determined. By definition,

$$E(WT) = \sum tP(WT = t).$$

For the negative multinomial distribution, we define a generalized least favorable configuration (GLFC) to be $p_{[1]} = \cdots = p_{[k-1]} = q$ and $p_{[k]} = p$. Let $E_{(i)}$ denote the category with probability $p_{[i]}$ that we observe m observations from before m observations from any of the other categories ($i = 1, 2, \ldots, k$). Cacoullos and Sobel (1966) pointed out that for any configuration,

$$E(WT) = m + \sum_{i=1}^{k} P(E_{(i)}) \left[\sum_{j \neq i} E(Y_{(j)} | E_{(i)}) \right]$$

As well, Cacoullos and Sobel (1966) gave the following exact expression for $E(WT)$ based on the GLFC with $q \leq p$,

$$E(WT|GLFC) = \frac{m}{q}[1 - (\frac{p-q}{p})D_{k-1}(m, m; q/p) - \frac{1}{2p} b_{2m}(m; q_1) D_{k-2}, (2m, m; q_1)],$$

where $q_1 = q/(p+q)$, and $D_s(M, N; a)$ is the Dirichlet integral, which was first con-

sidered by Sobel et al. (1985), defined by

$$D_s(M,N;a) = \frac{\Gamma(M+sN)}{\Gamma(M)\Gamma(N)^s}\int_a^\infty \cdots \int_a^\infty \frac{\prod_{i=1}^s [y_i^{N-1}dy_i]}{(1+\sum_{i=1}^s y_i)^{M+sN}}, (M,N>0),$$

and $b_n(x;a)$ is the binomial probability given by

$$b_n(x;a) = \binom{n}{x} a^x (1-a)^{n-x}, (x \le n, 0 < a < 1).$$

Cacoullos and Sobel (1966) also developed the following asymptotic expression for $E(WT)$ based on the GLFC with $q \le p$,

$$E(WT|GLFC) \approx \frac{m}{q}\left\{1 - \left(\frac{p-q}{p}\right)\int_{-\infty}^{\infty} F^m\left(x + m^{1/2}\log\frac{p}{q}\right) f(x)dx - \frac{1}{2p(m\pi)^{1/2}} \times \left[\frac{4pq}{(p+q)^2}\right]^m \times \int_{-\infty}^{\infty} F^{m-1}\left(\frac{x+m^{1/2}\log[(p+q)/2q]}{2^{1/2}}\right) f(x)dx\right\}.$$

where $f(x)$ is the standard univariate normal density function and $F(x)$ is the corresponding cumulative distribution function.

In order to give the corresponding formula for $E(WT)$ in the hypergeometric case, we first define the Dirichlet HD function. Suppose that we have $b+1$ different types of items in a population with sizes $M_1, \cdots, M_{b+1}$. Let $Y_{(i)}$ denote the number of observations associated with M_i $(i = 1, 2, \ldots, b+1)$. Let $\vec{M} = (M_1, \cdots, M_b)$ and $\vec{r} = (r_1, \cdots, r_b)$. Then $HD^{(b)}_{\vec{M},N}(\vec{r};m)$, the Dirichlet HD function, is defined to be the probability that category i (with M_i total elements in the population) will have fewer than r_i observations $(i = 1, 2, \ldots, b)$ when the $(b+1)$-th category reaches m observations for the first time. Let $\vec{M}_i = (M_1, \cdots, M_{i-1}, M_{i+1}, \cdots, M_k)$, $\bar{M} = \frac{1}{k}\sum_{i=1}^k M_i$, and $\vec{m} = \underbrace{(m, \cdots, m)}_{k-1}$. Then $HD^{(k-1)}_{\vec{M}_i, k\bar{M}+1}(\vec{m}; m+1)$ is considered as the probability that category j $(j = 1, 2, \ldots, k;\ j \ne i)$ will have fewer than m observations when the i-th category with $M_i + 1$ total elements in the population reaches $m+1$ observations for

the first time, and can be calculated as follows,

$$
\begin{aligned}
HD^{(k-1)}_{\vec{M}_i,k\bar{M}+1}(\vec{m};m+1) &= P(Y_{(1)} < m, \cdots, Y_{(i-1)} < m, Y_{(i+1)} < m, \cdots, Y_{(k)} < m) \\
&= \sum_{y_1=1}^{min(m-1,M_1)} \cdots \sum_{y_{i-1}=1}^{min(m-1,M_{i-1})} \sum_{y_{i+1}=1}^{min(m-1,M_{i+1})} \cdots \sum_{y_k=1}^{min(m-1,M_k)} \\
&\quad \frac{\binom{M_1}{y_1} \cdots \binom{M_{i-1}}{y_{i-1}} \binom{M_i+1}{m} \binom{M_{i+1}}{y_{i+1}} \cdots \binom{M_k}{y_k}}{\binom{N+1}{\sum_{j\neq i} y_j + m}} \frac{M_i - m + 1}{N - \sum_{j\neq i} y_j - m + 1}.
\end{aligned}
$$

For the multivariate inverse hypergeometric distribution, Childs (2010) introduced an easy way to calculate the exact $E(WT)$ as follows

$$
E(WT) = \sum_{i=1}^{k} \frac{m(N+1)}{M_i+1} HD^{(k-1)}_{\vec{M}_i,k\bar{M}+1}(\vec{m};m+1). \tag{2.26}
$$

As well, an asymptotic expression of $E(WT)$ can be derived using the multivariate normal approximation since $HD^{(k-1)}_{\vec{M}_i,k\bar{M}+1}(\vec{m};m+1)$ in (2.26) can be considered as follows

$$
HD^{(k-1)}_{\vec{M}_i,k\bar{M}+1}(\vec{m};m+1) = P(Y_{(j)} \leq m-1; j \neq i) = P(Z_j \leq 1 - \frac{1}{m};\ j \neq i)
$$

where $Z_j = \frac{Y_{(j)}}{m}$.

As with calculating the approximate *pcs* of the multivariate inverse hypergeometric distribution, $E(Z_j), Var(Z_j), Corr_{j\neq l}(Z_j, Z_l)$ $(j \neq i; l \neq i; j \neq l)$ are given by (2.17), (2.18), and (2.19), respectively. Thus, $HD^{(k-1)}_{\vec{M}_i,k\bar{M}+1}(\vec{m};m+1)$ can be approximated by using the multivariate normal distribution, hence the asymptotic $E(WT)$.

Chapter 3

The RS Package

The **RS** package is based on `S4` classes and methods (Chambers, 1999). This package provides tools for calculating, displaying and plotting the probabilities of correctly selecting the most probable category from a k-level population according to different distributional assumptions and sampling types using the least favorable configuration. Moreover, it also contains a function to find the specified smallest sample size (or cell quota and expected sample size) for given values of $\theta^\star$ and $P^\star$.

3.1 Object classes

The package consists of one virtual class, `LF2c`. It captures several parameters, namely the distribution, `dtype`, the sampling type, `stype`, the method of calculation, `ctype`, the number of categories, `k`, the smallest value of the ratio between the largest and second largest probabilities or category sizes, `thetastar`, and the probability of correct selection, `pcs`. The distributions that can be specified for `dtype` are

- `multinomial` for the multinomial distribution, which is used when the population is large or the sample is taken with replacement.

- `hypergeom` for the multivariate hypergeometric distribution, which is used when the population is finite and the sample is taken without replacement.

The sampling procedures that can be specified for `stype` are

- `fs` for fixed sampling. For `"fs"`, a sample size `n` must be provided and the calculations are based on the multinomial or multivariate hypergeometric distribution, depending on the distribution specified by `dtype`.
- `is` for inverse sampling. For `"is"`, the observations are selected one at a time and the calculations are based on the negative multinomial or multivariate inverse hypergeometric distribution, depending on the distribution specified by `dtype`.

The calculation methods specified for `ctype` are

- `exact` for exact calculation of the probability of correct selection (and exact expected sample size for `stype="is"`), which is based on the multinomial, negative multinomial, multivariate hypergeometric, or multivariate inverse hypergeometric distribution depending on the distribution `dtype` and sampling method `stype`. See Section 2.1 and 2.2 for details.
- `approximate` for the approximate calculation of the probability of correct selection (and expected sample size for `stype="is"`), which is also based on the above four distributions but uses multivariate normal approximations. For the multinomial and multivariate hypergeometric distributions, we use a normal approximation with arcsin transformation, while for the negative multinomial and multivariate inverse hypergeometric distributions, an untransformed normal approximation is used. See Section 2.4 for details.

`LF.multinomial` and `LF.hypergeom` are the two actual classes derived from the virtual class `LF2c` (Figure 3.1).

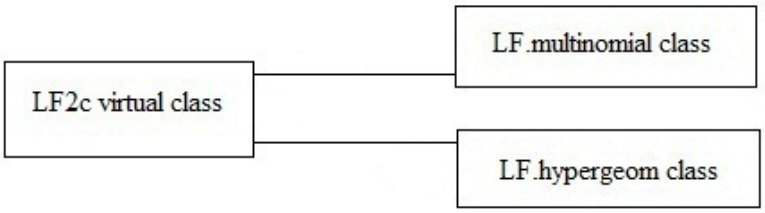

Figure 3.1: Class structure

Both derived classes contain the `LF2c` virtual class and its slots. Moreover, the `LF.multinomial` class also contains the additional slots for the sample size `n` and cell quota `m`; the `LF.hypergeom` class contains population size `N`, sample size `n`, and the cell quota `m`. Thus, new objects of these two classes can be produced by the LF2c function which takes the following arguments.

`dtype` : The type of distribution.

`stype` : The type of sampling.

`ctype` : The calculation method.

`k` : The number of categories.

`thetastar` : For `dtype="multinomial"`, this is a vector whose components represent the smallest ratio between the probabilities corresponding to the categories with the largest and second largest probabilities. For `dtype="hypergeom"`, this is a vector whose components represent the smallest ratio between the category sizes corresponding to the categories with largest and second largest sizes.

`...` : Additional arguments which depend on the distribution and sampling stype.

For the `stype="fs"`, `n` is needed, while `m` must be specified when `stype="is"`. When `dtype=hypergeom`, `N` must also be provided.

The required arguments depending on the different types of distribution and sampling procedure are listed in Table 3.1. If the value of `thetastar` is not provided, it will take a default value of `seq(1.1, 2, 0.05)`. The default values for `n`, `m` and `N` are 20, 8, and 100 respectively.

After defining these arguments, a new object can be created and returned after validation of the arguments.

dtype	stype	Arguments	distribution
`multinomial`	`fs`	`k, thetastar, n`	multinomial
	`is`	`k, thetastar, m`	negative multinomial
`hypergeom`	`fs`	`k, thetastar, N, n`	multivariate hypergeometric
	`is`	`k, thetastar, N, m`	multivariate inverse hypergeometric

Table 3.1: Arguments and corresponding sampling distributions for the `LF2c` class.

3.2 Validation methods

The validation functions are used to validate whether the given values of the arguments make sense for the specified distribution and sampling type. Because of inheritance, the validation functions for the virtual class `LF2c` are also applied to the derived classes `LF.multinomial` and `LF.hypergeom`.

The validation function for the virtual class checks that `k` is not NA , `k` only contains one value and `k` is an integer greater than 1.

In addition, the two actual classes also contain validation functions:

- `LF.multinomial` class:

  (1) For `stype "fs"`, we check that `thetastar` and `n` contain no NA's, each

component of `thetastar` is greater than 1, `n` only contains one value and is an integer greater than 0.

(2) For `stype "is"`, we check that `thetastar` and `m` contains no NA's, each component of `thetastar` is greater than 1, `m` only contains one value and is an integer greater than 0.

- `LF.hypergeom` class:

  (1) For `stype "fs"`, we check that `thetastar`, `n` and `N` contain no NA's, each component of `thetastar` is greater than 1, `n` or `N` only contains one value, `N` is an integer greater than `thetastar`+`k`-1, and `n` is an integer greater than 0 and no greater than `N`.

  (2) For `stype "is"`, we check that `thetastar`, `m` and `N` contain no NA's, each component of `thetastar` is greater than 1, `m` or `N` only contains one value, `N` is an integer greater than `thetastar`+`k`-1, and `m` is an integer greater than 0 and no greater than `N`-`k`+1.

3.3 Show and Summary methods

The generic `show` and `summary` methods are created to display information about the created objects. The `show` method provides a brief summary for a certain object. For `stype "fs"`, it displays the number of categories `k` and sample size `n`; for `stype "is"`, the number of categories `k` and cell quota `m` will be printed. If `ctype` is `"hypergeom"`, the population size `N` will also be shown.

The `summary` method shows all the information for the object, that is all the values of `thetastar` and the corresponding values of `pcs`, in addition to the information given by the `show` method. If `ctype` is `"hypergeom"`, `summary` will give the least favorable configuration, and if `stype` is `"is"`, the expected waiting time `EWT` will also be displayed. See Sections 4.1 and 4.2 for examples.

3.4 Plot method

The PCS curve shows how the probability of correct selection changes when `thetastar` varies. As a consequence, a plot method has been created for both classes with `thetastar` on the horizontal axis and `pcs` on the vertical axis. The signatures for these plot functions are

- `signature(x="LF.multinomial", y="missing");`
- `signature(x="LF.hypergeom", y="missing").`

Examples are provided in Section 4.3.

3.5 Finding the smallest sample size or cell quota

A function named `find.n` can be used to find the smallest sample size using fixed sampling or the expected smallest sample size using inverse sampling such that the specified probability of correct selection is met.

In the case of `dtype "multinomial"`, `p`, `k` and `thetastar` need to be provided. While for `dtype "hypergeom"`, the population size `N` must also be specified.

When `dtype` is `"multinomial"`, for `stype "fs"`, the function finds the smallest sample size `n`; and for `stype "is"`, it finds the smallest cell quota `m` and the corresponding expected waiting time `EWT`. But when `dtype` is "hypergeom", the additional parameter `M` (the least favorable configuration) will also be shown. Using `ctype "exact"`, the result gives an exact calculation of `n` or `m` and `EWT`; for `ctype "approximate"`, the function gives an approximate calculation of `n` or `m` and `EWT`.

When `ctype` is `"exact"`, the process of finding the smallest (expected) sample size is through trial starting with `n`=1 or `m`=1 and increasing `n` or `m` until the condition

is satisfied. When `ctype` is `"approximate"`, the function uses different approximations depending on different distributional assumptions and sampling procedures, as described in Sections 2.5 and 2.6.

Chapter 4

Examples

4.1 Basic sampling information

After loading the package, the basic sampling information can be shown by using the `LF2c` function. For instance, the information based on a multinomial distribution with $k = 3$, $n = 11$ and $\theta^\star = 1.6$ can be obtained as follows:

```
> library(RS)
> library(mvtnorm)
> k <- 3
> n <- 11
> thetastar <- 1.6
> x.mf <- LF2c(k = k, thetastar = thetastar, n = n)
> x.mf

Selecting the Most Probable Event for Multinomial

Number of categories: k = 3
Value(s) of thetastar: 1.6
Sample size: n = 11
```

For a finite population of size $N = 100$, `dtype = "hypergeom"` (can also be given

as "h") is needed. Here we take $n = 10$, $k = 3$ and $\theta^\star = (1.1, 1.2)$, then the basic information will be shown as follows:

```
> thetastar <- c(1.1, 1.2)
> k = 3
> n = 10
> N = 100
> x.hf <- LF2c(dtype = "h", thetastar = thetastar, N = N, k = k, n = n)
> x.hf

Selecting the Most Probable Event for Multivariate Hypergeometric

Number of categories: k = 3
Value(s) of thetastar: 1.1, 1.2
Population size: N = 100
Sample size: n = 10
```

Now consider the inverse sampling in which sampling continues until we obtain $m = 8$ observations from any of the categories. In this case, we take `stype "is"` (or just `"i"`).

```
> x.mi <- LF2c(dtype = "m", thetastar = 2, stype = "is", k = 3, m = 8)
> x.mi

Selecting the Most Probable Event for Negative Multinomial

Number of categories: k = 3
Value(s) of thetastar: 2
Cell quota: m = 8
```

4.2 Calculation summary

The `summary` method gives a summary with detailed probabilities of correct selection for specified conditions. The following examples show the detailed exact and approximate *pcs*'s for the multinomial distribution based on fixed sampling.

```
> x.mfe <- LF2c("m", stype = "fs", ctype = "exact", k = 3, n = 20,
+     thetastar = seq(1.2, 3, 0.3))
> summary(x.mfe)

Selecting the Most Probable Event for Multinomial

Number of categories: k = 3
Sample size: n = 20

 Detailed Probability(s) of Correct Selection

  Thetastar  P(CS).exact
        1.2   0.4775524
        1.5   0.6588646
        1.8   0.7864054
        2.1   0.8688125
        2.4   0.9198648
        2.7   0.9509115
        3.0   0.9696963
```

```
> x.mfa <- LF2c("m", stype = "fs", ctype = "approximate", k = 3,
+     n = 20, thetastar = seq(1.2, 3, 0.3))
> summary(x.mfa)

Selecting the Most Probable Event for Multinomial

Number of categories: k = 3
Sample size: n = 20

 Detailed Probability(s) of Correct Selection

  Thetastar  P(CS).approximate
        1.2         0.4791536
        1.5         0.6611310
        1.8         0.7876686
        2.1         0.8686669
        2.4         0.9186320
        2.7         0.9490776
        3.0         0.9676572
```

From the above output, we see that the exact and approximate *pcs*'s are really close for the multinomial distribution, even for a relatively small sample size.

As for the negative multinomial distribution (`dtype="m"` and `stype="is"`), the detailed exact and approximate *pcs*'s can be obtained as follows,

```
> x.nme <- LF2c("m", stype = "is", ctype = "exact", k = 4, m = 8,
+     thetastar = seq(1.1, 1.4, 0.05))
> summary(x.nme)

Selecting the Most Probable Event for Negative Multinomial

Number of categories: k = 4
Cell quota: m = 8

 Detailed Probability(s) of Correct Selection

  Thetastar  P(CS).exact      EWT
       1.10   0.3183419 27.98174
       1.15   0.3528709 27.71290
       1.20   0.3872058 27.41645
       1.25   0.4210719 27.09667
       1.30   0.4542359 26.75766
       1.35   0.4865048 26.40326
       1.40   0.5177241 26.03705



> x.nma <- LF2c("m", stype = "is", ctype = "approximate", k = 4,
+     m = 8, thetastar = seq(1.1, 1.4, 0.05))
> summary(x.nma)

Selecting the Most Probable Event for Negative Multinomial

Number of categories: k = 4
Cell quota: m = 8

 Detailed Probability(s) of Correct Selection

  Thetastar  P(CS).approximate      EWT
```

```
        1.10         0.3192254 27.81837
        1.15         0.3556613 27.60721
        1.20         0.3929276 27.35364
        1.25         0.4305656 27.05881
        1.30         0.4684932 26.72255
        1.35         0.5059924 26.35046
        1.40         0.5429047 25.94527
```

From the above output, we see that when $\theta^\star$ is small, the exact and approximate *pcs*'s are really close. But when $\theta^\star$ increases, the approximate *pcs* gets worse.

For a finite population size $N = 80$, the detailed probability of correct selection is shown as follows:

```
> x.mh <- LF2c("h", stype = "fs", ctype = "e", k = 3, N = 80, n = 10,
+     thetastar = c(1.1, 1.4))
> summary(x.mh)

Selecting the Most Probable Event for Multivariate Hypergeometric

Number of categories: k = 3
Population size: N = 80
Sample size: n = 10

 Detailed Probability(s) of Correct Selection

 Thetastar P(CS).exact M1 M2 M3
       1.1   0.4088886 25 26 29
       1.4   0.5724863 22 24 34
```

The least favorable configuration is also shown in the `summary` as M_i's.

The following example illustrates that inverse sampling has a smaller expected sample size than the corresponding fixed sample size with the approximately same $P^\star$ and $\theta^\star$.

```
> nm <- LF2c("m", "is", "e", k = 3, m = 8, thetastar = c(1.4, 1.6,
+     1.8))
> summary(nm)
```

```
Selecting the Most Probable Event for Negative Multinomial

Number of categories: k = 3
Cell quota: m = 8

 Detailed Probability(s) of Correct Selection

  Thetastar  P(CS).exact      EWT
        1.4   0.6072296 19.15739
        1.6   0.7093437 18.23985
        1.8   0.7872708 17.34150
```

```
> m1 <- LF2c("m", "fs", "e", k = 3, thetastar = 1.4, n = 20)
> m2 <- LF2c("m", "fs", "e", k = 3, thetastar = 1.6, n = 20)
> m3 <- LF2c("m", "fs", "e", k = 3, thetastar = 1.8, n = 20)
> x <- array(c(m1@thetastar, m2@thetastar, m3@thetastar, m1@pcs,
+      m2@pcs, m3@pcs, m1@n, m2@n, m3@n), dim = c(3, 3))
> colnames(x) <- c("Thetastar", "P(CS).exact", "Sample Size")
> rownames(x)<-rep("",3)
> x
```

```
 Thetastar P(CS).exact Sample Size
       1.4   0.6043679          20
       1.6   0.7071902          20
       1.8   0.7864054          20
```

4.3 Plotting the PCS curve

To visualize the PCS curve we use the `plot` function as follows

```
> thetastar <- seq(1.1, 3, 0.02)
> mf <- LF2c(dtype = "m", k = 3, thetastar = thetastarc, n = 10)
> plot(mf)
```

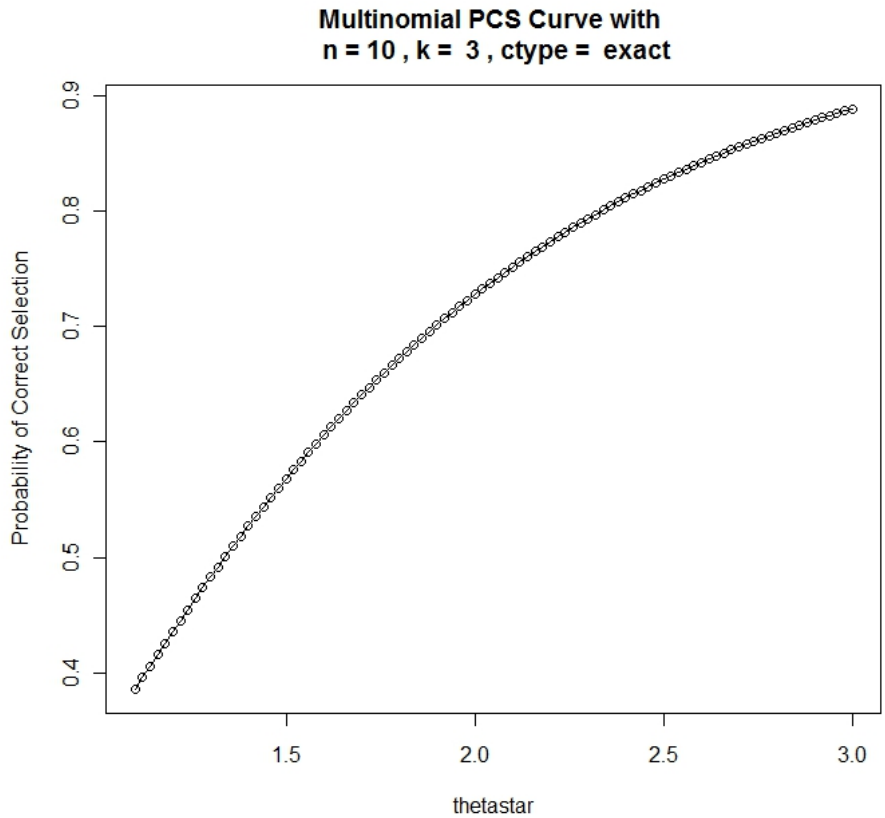

Figure 4.1: PCS curve for multinomial distribution

The plot is shown in Figure 4.1.

By default, the PCS curve for the multinomial or negative multinomial distribution is based on the standard plot type `type` = `"o"`, showing both lines and points, while the standard PCS curve for multivariate hypergeometric or multivariate inverse hypergeometric distribution is plotted only with points by using `type` = `"p"`. Other arguments for the plot method are also accepted.

Another example is used to see the probabilities of correct selection for all four combinations of the distribution and sampling type, which is given in Figure 4.2. Here we take $\theta^\star$ to be `seq(1.1,2.5,0.1)`, $n = 8$, $k = 3$, $m = 3$, and $N = 100$.

```
> x.m <- LF2c("m", "fs", "e", k = 3, thetastar = seq(1.1, 2.5,
+     0.1), n = 7)
```

```
> x.nm <- LF2c("m", "is", "e", k = 3, thetastar = seq(1.1, 2.5,
+     0.1), m = 3)
> x.mh <- LF2c("h", "fs", "e", k = 3, thetastar = seq(1.1, 2.5,
+     0.1), n = 7, N = 100)
> x.mih <- LF2c("h", "is", "e", k = 3, thetastar = seq(1.1, 2.5,
+     0.1), m = 3, N = 100)
> par(mfrow = c(1, 2))
> main = "Fixed Sampling with n=7, N=100"
> plot(x.m, type = "l", xlim = c(1.1, 2.5), ylim = c(0.35, 0.85),
+     main = main)
> grid(lty = "solid")
> points(x.mh@thetastar, x.mh@pcs, col = 3)
> legend(1.8, 0.5, c("multinomial", "hypergeometric"), col = c(1,
+     3), lty = c(1, -1), pch = c(-1, 1), bg = "gray95")
> main = "Inverse Sampling with m=3, N=100"
> plot(x.nm, type = "l", xlim = c(1.1, 2.5), ylim = c(0.35, 0.85),
+     main = main)
> grid(lty = "solid")
> points(x.mih@thetastar, x.mih@pcs, col = 3)
> legend(1.8, 0.5, c("negative multinomial", "inverse hypergeometric"),
+     col = c(1, 3), lty = c(1, -1), pch = c(-1, 1), bg = "gray96")
```

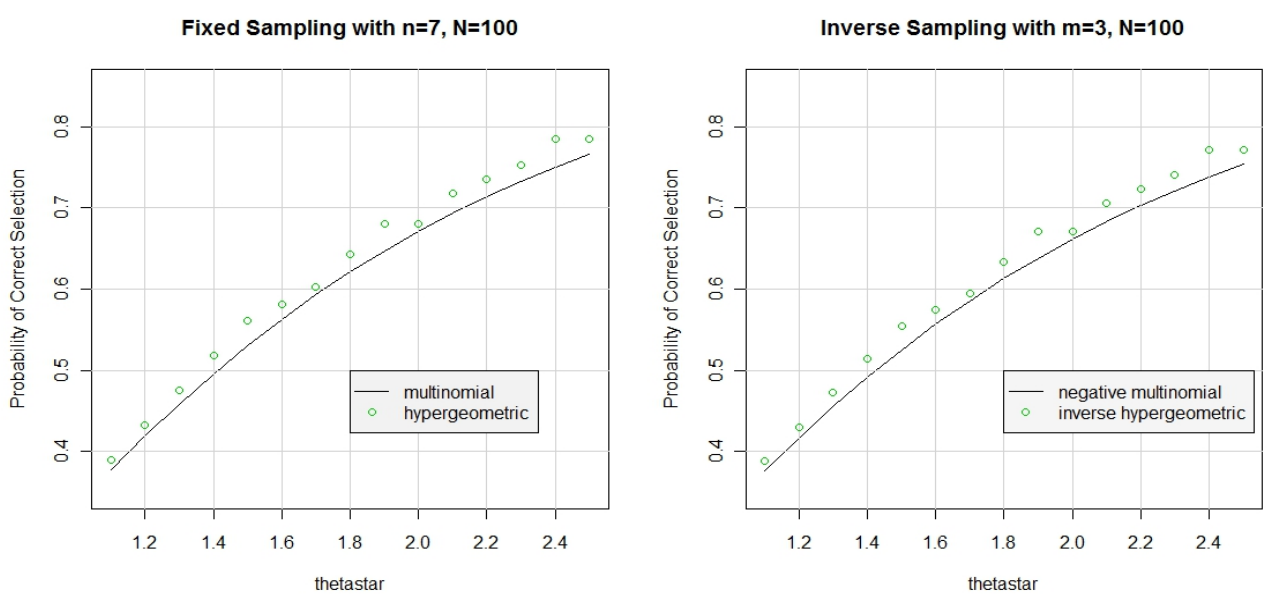

Figure 4.2: PCS curve for fixed sampling and inverse sampling

From Figure 4.2, we observe that the probabilities of correct selection for the hypergeometric distribution are greater than those for multinomial distribution, which means that sampling from a finite population will have a relatively smaller risk of making an incorrect selection. Note that for `thetastar`=1.9 and 2.0, the probabilities of correct selection are the same for both multivariate hypergeometric and multivariate inverse hypergeometric distributions because the least favorable configurations for `thetastar`=1.9 and `thetastar`=2.0 are identical. It is the same situation when `thetastar`=2.4 and 2.5. From Figure 4.2, we know that when n=7, m=3 and N=100, the probabilities of correct selection are almost the same for fixed and inverse sampling. The following example gives the expected sample sizes for the negative multinomial and multivariate inverse hypergeometric distributions using the same arguments as the last example.

```
> par(mfrow = c(2, 1))
> main = "EWT for Negative Multinomial with m=3"
> plot(x.nm@thetastar, x.nm@EWT, "o", xlab = "thetastar", ylab = "EWT",
+     main = main)
> main = "EWT for Multivariate Inverse Hypergeometric with m=3, N=100"
> plot(x.mih@thetastar, x.mih@EWT, "o", xlab = "thetastar", ylab = "EWT",
+     main = main)
```

From Figure 4.3, we see that the expected sample size is always less than 7 when m=3, which illustrates that inverse sampling has a smaller expected total number of observations than the corresponding fixed sample size procedure with the same $P^\star$ and $\theta^\star$.

The following example aims at comparing the probabilities of correct selection for exact and approximate calculation based on the multinomial distribution (see Figure 4.4).

```
> x.me <- LF2c("m", "fs", "e", k = 3, thetastar = seq(1.1, 3, 0.05),
+     n = 8)
> x.ma <- LF2c("m", "fs", "a", k = 3, thetastar = seq(1.1, 3, 0.05),
+     n = 8)
> main = "Comparison of Calculations with n=8, k=3 based on the
```

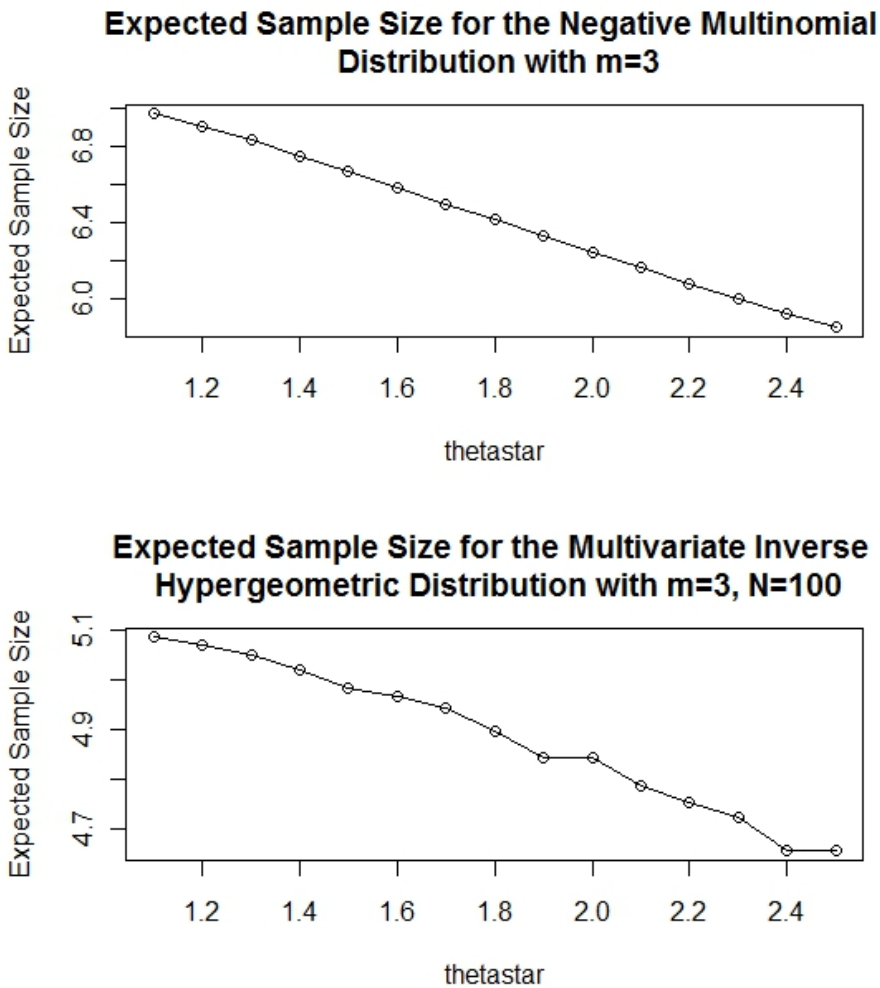

Figure 4.3: Expected sample size for inverse sampling

```
+     Multinomial Distribution"
> plot(x.me, type = "l", xlim = c(1.1, 3), main = main)
> grid(lty = "solid")
> points(x.ma@thetastar, x.ma@pcs, col = 10)
> legend(1.8, 0.5, c("Exact", "Approximate"), col = c(1, 10), lty = c(1,
+     -1), pch = c(-1, 1), bg = "gray95")
```

Figure 4.4 illustrates that the approximate probabilities of correct selection are really close to the exact probabilities of correct selection for the multinomial distribution.

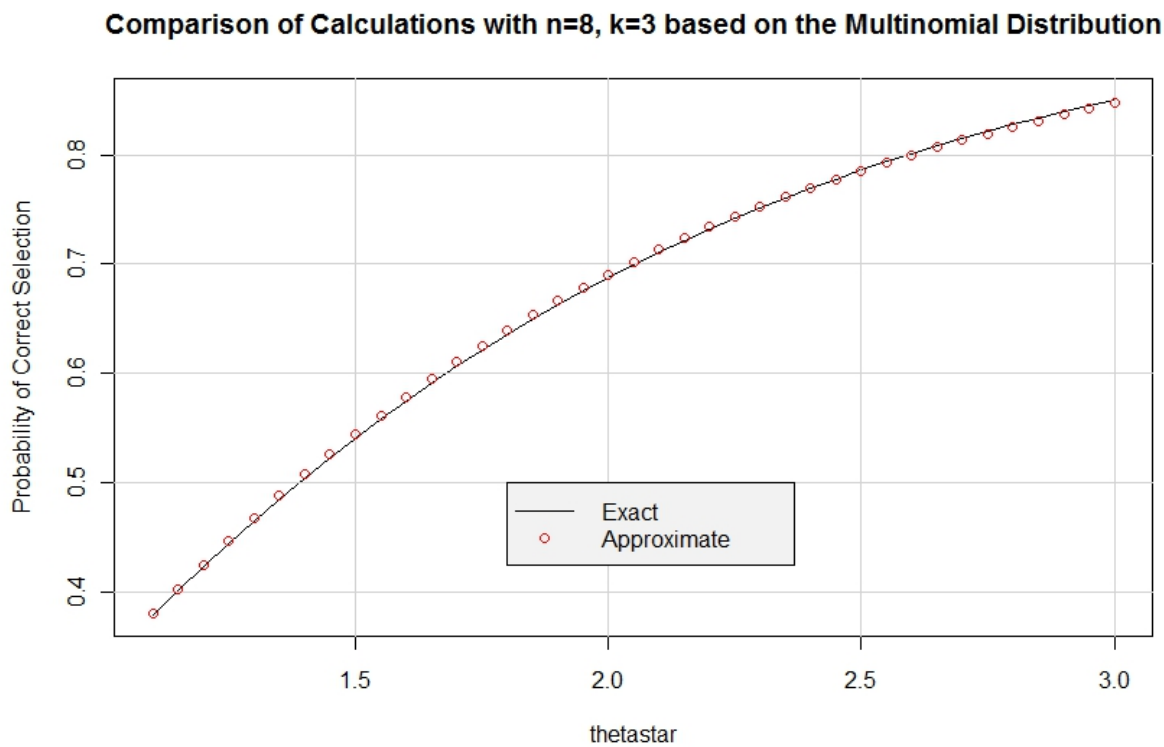

Figure 4.4: Comparison of calculations based on the multinomial distribution.

4.4 Finding the smallest sample size (or cell quota and expected sample size)

`Find.n` can be used to find the smallest acceptable sample size (or cell quota and expected sample size) which will meet the specified probability of correct selection. It can be implemented as follows:

```
> find.n(dtype = "multinomial", stype = "fs", ctype = "exact",
+     0.8, 3, 1.5)

Finding the exact sample size for multinomial

Number of categories: k = 3
Sample size: n.exact = 47
```

```
  Thetastar  P.star P(CS).exact
        1.5    0.8   0.8019867
```

The result means that, based on the multinomial distribution, fixed sampling and exact calculation, in order to meet probability of correct selection greater than 0.8 for $k = 3$ and $\theta^\star = 1.5$, a sample size of 47 should be taken and the probability of correct selection is 0.8019867. As well, if the calculation lasts too long, method of approximate calculation can give another choice:

```
> find.n(dtype = "multinomial", stype = "fs", ctype = "approximate",
+     0.8, 3, 1.5)

Finding the approximate sample size for multinomial

Number of categories: k = 3
Sample size: n.approximate = 47

  Thetastar  P.star P(CS).approximate
        1.5    0.8         0.8030263
```

We can also find the cell quota m to stop sampling and the corresponding expected sample size for inverse sampling as below:

```
> find.n(dtype = "h", stype = "is", ctype = "exact", p = 0.5, k = 3,
+     thetastar = 1.1, N = 80)

Finding the exact sample size for multivariate inverse hypergeometric

Number of categories: k = 3
Cell quota: m.exact = 13
Expected sample size: expected.n.exact = 31.36891

  Thetastar  M1 M2 M3 P.star P(CS).exact
        1.1 25 26 29    0.5    0.513029
```

The above output shows that based on the multivariate hypergeometric distribution, inverse sampling, and exact calculation, in order for the probability of correct selection

to be greater than 0.5 for $k = 3$, $\theta^\star = 1.1$ and $N = 80$, we should continue sampling until we get $m = 13$ of any of the category types, the expected sample size is 31.36891, and the probability of correct selection is 0.513029. If the calculation lasts too long, a method of approximate calculation is provided as well.

```
> find.n(dtype = "h", stype = "is", ctype = "approximate", p = 0.5,
+     k = 3, thetastar = 1.1, N = 80)

Finding the approximate sample size for multivariate inverse hypergeometric

Number of categories: k = 3
Cell quota: m.approximate = 10
Expected sample size: expected.n.approximate = 26.17803

  Thetastar  M1 M2 M3 P.star P(CS).approximate
        1.1 25 26 29    0.5         0.5102845
```

Chapter 5

Discussion

The **RS** package provides the users with tools to calculate, plot and display the probabilities of correctly selecting the most probable category using the least favorable configuration for both fixed and inverse sampling. A function for finding the specified smallest acceptable sample size (or cell quota and expected sample size) is also provided.

This package provides both exact and approximate calculations and is written entirely in R. As a result, when `ctype` is `"exact"`, the calculation of `LF2c` function will be slow for large values of k, m, n and N. Consequently, the `find.n` function can be slow if the acceptable probability of correct selection given is too close to 1. This could be improved by using FORTRAN or C programs for the *pcs* calculations, and incorporating the compiled code into the package.

When `ctype` is `"approximate"` and `dtype="multinomial"`, the calculations are really fast. But when `dtype="hypergeom"`, the calculations of `LF2c` and `find.n` function will be slow for large values of given N and k since we need to calculate the probability of correct selection for all possible $M_{[i]}$'s in order to find the least favorable configuration. Therefore the package could be improved by developing a more efficient algorithm for finding the least favorable configuration.

Moreover, the approximate calculations also have several restrictions. For the negative multinomial distribution, the approximate calculation will not be good when $\theta^\star$ is large and m is relatively small. For the multivariate inverse hypergeometric distribution, the approximate calculation is only good when N is large and m is relatively small. Therefore, the package could be improved by developing other possible approximations.

Appendix A

RS Package Code

The **RS** package code includes two parts, namely, `LF2c.R` and `find.n.R`.

A.1 LF2c.R

The `SetClass` function is used to create a virtual class `LF2c` and two actual classes including `LF.multinomial` and `LF.hypergeom`, and the function `new` is used to create an new object. We also use the `SetMethod` function to generate the generic functions involving `show`, `summary` and `plot`. The `LF2c` function is the major function to calculate the probabilities of correct selection. Other small functions are also created for certain uses.

```
## Class definitions
## Create a virtual class
setClass("LF2c",
  representation(k="numeric",
                 # The number of categories in the population
                 dtype="character",
                 # The type of distribution
                 stype="character", # The type of sampling
```

```
                ctype="character",# The type of calculation
                pcs="numeric",# Probability of correct selection
               "VIRTUAL"),
  validity=function(object){
  if(is.na(object@k))
   return("Missing value in 'k' not allowed")
  # Check that k is reasonable
  if (object@k!=as.integer(object@k))
   return(" 'k' must be an integer")
  # Check that k is an integer
  if (object@k <= 1)
   return(" 'k' must be greater than 1")
  # Check that k is greater than 1
   return(TRUE)
         })
## Create two classes
setClass("LF.multinomial",
  representation("LF2c",
                n="numeric",# The sample size
                m="numeric",# Cell quota
                EWT="matrix",# Expected sample size.
                thetastar="numeric"),
# The smallest value of the ratio between the highest and second highest
# probabilities for multinomial and the smallest value of the ratio between
# the largest and second largest category sizes for hypergeometric
   contains="LF2c",
   prototype=list("LF2c",n=20,m=8, dtype="multinomial",
    thetastar=seq(1.1,2,by=0.05)),
   validity=function(object){
   if (any(is.na(object@thetastar)))
    return("Missing values in 'thetastar' not allowed")
   # Check that thetastar is reasonable
   if(is.na(object@n)|is.na(object@m))
    return("Missing values in 'n'and 'm' not allowed")
   # Check that n and m are reasonable
   if(length(object@n)!=1|length(object@n)!=1)
    return("n or m must only contain one value")
   # Check that n or m only contians one value
   if (any(object@thetastar <=1 ))
    return("Thetastar must be greater than 1")
   # Check that thetastar is greater than 1
```

```
   if (object@n<=0)
    return(" 'n' must be greater than 0")
   # Check that n is greater than 0
   if (object@n!=as.integer(object@n))
    return(" 'n' must be an integer")
   # Check that n is an integer
   if (object@m<=0)
    return(" 'm' must be greater than 0")
   # Check that m is greater than 0
   if (object@m!=as.integer(object@m))
    return(" 'm' must be an integer")
   # Check that m is an integer
         });
setClass("LF.hypergeom",
   representation("LF2c",
                  N="numeric",
                  #The population size from which the sample is drawn
                  n="numeric",
                  m="numeric",
                  thetastar="numeric",
                  M="matrix",
                  # The population category sizes that minimize pcs
                  EWT="matrix"),
   contains="LF2c",
   prototype=list("LF2c", dtype="hypergeom",N=100,n=20,m=8,
    thetastar=seq(1.1,2,by=0.05)),
   validity=function(object){
   if (any(is.na(object@thetastar)))
     return("Missing values in 'thetastar' not allowed")
   if(is.na(object@N))
     return("Missing values in 'N' not allowed")
   # Check that N is reasonable
   if(is.na(object@n)|is.na(object@m))
     return("Missing values in 'n' and 'm' not allowed")
   if(length(object@n)!=1|length(object@n)!=1)
    return("n or m must only contain one value")
   # Check that n or m only contians one value
   if (any(object@thetastar <1 ))
     return("Thetastar must be greater than 1")
   if (any(object@N<(object@thetastar+object@k-1)))
     return("N must be no less than thetastar+k-1")
```

```
    # Check that N is no less than thetastar+k-1
    if (object@N!=as.integer(object@N))
      return(" 'N' must be an integer")
    # Check that N is an integer
    if (object@n<=0)
      return(" 'n' must be greater than 0")
    if (object@n!=as.integer(object@n))
      return(" 'n' must be an integer")
    if (object@n>object@N)
      return(" 'n' must be less than or equal to N")
    # Check that n is no greater than N
    if (object@m<=0)
      return(" 'm' must be greater than 0")
    if (object@m!=as.integer(object@m))
      return(" 'm' must be an integer")
    if (object@m>(object@N-object@k+1))
      return(" 'm' must be less than or equal to N-k+1")
    # Check that m is no greater than N-k+1
          });
## Creation of the object
LF2c <- function(dtype=c("multinomial","hypergeom"),
 stype=c("fs","is"),ctype=c("exact","approximate"),k,...){
 # Choose what 'type' to use
 dtype <- match.arg(dtype);
 stype <- match.arg(stype);
 ctype<-match.arg(ctype);
 LFtype <- paste("LF",dtype,sep=".");
 # a new object of that type
 obj <- new(LFtype, k=k,dtype=dtype,stype=stype,ctype=ctype,...)
 LFtype<- get(paste("calc",LFtype,sep=".."));
 #LFtype<- getFromNamespace(paste("calc",LFtype,sep=".."), ns="RS");
 if (dtype =="multinomial"&stype=="fs")
  {obj@pcs<-LFtype(k=obj@k, n=obj@n, thetastar=obj@thetastar,
   stype=obj@stype,ctype=obj@ctype)}
 else if (dtype =="multinomial"&stype=="is")
  {obj@pcs<- LFtype(k=obj@k,m=obj@m,thetastar=obj@thetastar,
   stype=obj@stype,ctype=obj@ctype);
   obj@EWT<- EWT(obj@m,obj@k,obj@thetastar,obj@ctype,obj@pcs);
  }
 else if (dtype =="hypergeom"&stype=="fs")
  {obj@pcs<- LFtype(k=obj@k, N=obj@N, thetastar=obj@thetastar,
```

```
   stype=obj@stype,n=obj@n,ctype=obj@ctype)[,k+1];
  temp<- LFtype(k=obj@k, N=obj@N,thetastar=obj@thetastar,
   stype=obj@stype, n=obj@n,ctype=obj@ctype)[,1:k];
   obj@M<-as.matrix(temp)}
 else if (dtype =="hypergeom" & stype=="is")
  {obj@pcs<- LFtype(k=obj@k, N=obj@N, thetastar=obj@thetastar,
   stype=obj@stype,m=obj@m,ctype=obj@ctype)[,k+1];
   temp<- LFtype(k=obj@k, N=obj@N,thetastar=obj@thetastar,
   stype=obj@stype,m=obj@m,ctype=obj@ctype)[,1:k];
   obj@M<-as.matrix(temp);
   obj@EWT<- ETW(obj@M,obj@m,obj@k,obj@ctype);
  }
 obj
 };
## Calculating the exact pcs for multinomial
calc.LF.multinomial.fs.exact<-function(k,n,thetastar)
{f<-function(k,n,thetastar)
{p<-rep(NA,k);
for (i in 1:(k-1)){
p[i]=1/(thetastar+k-1)};
p[k]=thetastar/(thetastar+k-1);
phi=0;
# Calculating the exact pcs when k is 2
if(k==2){
for (i in 0:floor(n/2))
{ s=1+sign(i==(n-i));
phi=phi+ 1/s*sum(dmultinom(c(i,n-i),n,c(p)));}};
# Calculating the exact pcs when k is 3
if (k==3)
{for (i in ceiling(n/3):n)
{
for (j in 0:(n-i))
{if (j<=i & (n-i-j)<=i)
{
s=1+sign(i==j)+sign((n-i-j)==i);
phi=phi+1/s*sum(dmultinom(c(j,n-i-j,i),n,c(p)));
}}}};
# Calculating the exact pcs when k is greater than 3
if (k>=4)
{for (i in ceiling(n/k):n)
{x<-combn(n-i+k-2,k-2);
```

```
n1<-ncol(x);
z<-array(rep(0,(k-1)*n1),dim=c(k-1,n1));
z[1,]=x[1,]-0;
for (j in 2:(k-2)){
z[j,]=x[j,]-x[j-1,];}
z[k-1,]=n-i+k-1-x[k-2,];
y<-z-1;
lo<-y<=i;
d<-rep(0,n1);
for(b in 1:n1){d[b]<-sign(sum(lo[,b])==(k-1))};
e<-which(d==1);
y<-array(y[,e],dim=c(k-1,length(e)));
s<-rep(0,ncol(y));
# Counting the number of Yi's tied for largest
for(j in 1:ncol(y))
{s[j]=1+sum(sign(y[,j]==i))
phi=phi+1/s[j]*dmultinom(c(as.vector(y[,j]),i),n,c(p));}}}
return(phi);}
l=length(thetastar);
p=rep(NA,l);
for (i in 1:l)
{p[i]=f(k,n,thetastar[i])};
return(p);
}
## Calculating the approximate pcs for multinomial
calc.LF.multinomial.fs.approximate<-function(k,n,thetastar)
{f<- function(k,N,thetastar){
p0<-rep(NA,k);
for (i in 1:(k-1)){
p0[i]=1/(thetastar+k-1)};
p0[k]=thetastar/(thetastar+k-1);
p<-p0[k];
q<-p0[1];
f0<-function(x){2*asin(sqrt(x))};
f1<-function(x){1/(sqrt(x*(1-x)))};
A<-f0(p)-f0(q);
B<-(f1(p)^2*p+f1(q)^2*q-(p*f1(p)-q*f1(q))^2)/N;
C<-(f1(p)^2*p-(p*f1(p)-q*f1(q))^2)/N;
corr.ij<-C/B;
J<-rep(corr.ij,(k-1)*(k-1));
corr<-diag(1-corr.ij,(k-1))+J;
```

```
lower<-rep(-A/sqrt(B),k-1);
# Calculating the approximate pcs when k is 2
if (k==2)
{p<-1-pnorm(lower);}
# Calculating the approximate pcs when k is greater than 2
else {p<-pmvnorm(lower,corr=corr);}
return(p[1]);
}
l=length(thetastar);
p=rep(NA,l);
for (i in 1:l)
{p[i]=f(k,n,thetastar[i])};
return(p);
}
## Calculating the pcs for multinomial
calc.LF.multinomial.fs<-function(k,n,thetastar,ctype)
{if (ctype=="exact") p<-calc.LF.multinomial.fs.exact(k,n,thetastar)
 else if (ctype=="approximate")
 p<-calc.LF.multinomial.fs.approximate(k,n,thetastar);
 return(p);
}
## Calculating the exact pcs for negative multinomial
calc.LF.multinomial.is.exact<-function(m,k,thetastar)
{f<-function(m,k,thetastar){
p<-rep(NA,k);
for (i in 1:(k-1)){
p[i]=1/(thetastar+k-1)};
p[k]=thetastar/(thetastar+k-1);
phik=0;
phi0=0;
phi=0;
f<-function(list,m){exp(m*log(p[k])+lgamma(m+sum(list))+
sum(list*log(p[1:(k-1)]))-sum(lgamma(list+1))- lgamma(m))}
cond<-function(x,list){return(c(x,list))};
sss<-function(i,list)
{if (i>(k-1)) phi0=0
else {for (j in (0:(m-1)))
{phi<-phi+ss(i,cond(list,j));}}
return(phi0+phi)
};
ss<-function(i,list)
```

```
{if (length(list)==k-1)ss=f(list,m)
else ss=sss(i+1,list);
return(ss);}
phik=phik+sss(1,c());
return(phik)
}
l=length(thetastar);
p=rep(NA,l);
for (i in 1:l)
{p[i]=f(m,k,thetastar[i])};
return(p);
}
## Calculating the approximate pcs for negative multinomial
calc.LF.multinomial.is.approximate<-function(m,k,thetastar)
{f<-function(m,k,thetastar){
p<-rep(NA,k);
for (i in 1:(k-1)){
p[i]=1/(thetastar+k-1)};
p[k]=thetastar/(thetastar+k-1);
l<-p[1:(k-1)]/p[k];
A<-l;
B<-l*(1+l)/m;
corr.ij<-l[1]/(1+l[1]);
J<-array(rep(corr.ij,(k-1)*(k-1)),dim=c(k-1,k-1));
for (i in 1:(k-1))
{J[i,i]=1};
lower<-(-1+A)/sqrt(B);
if (k==2)
{p<-1-pnorm(lower);}
else {p<-pmvnorm(lower=lower,corr=J);}
return(p[1]);
}
l=length(thetastar);
p=rep(NA,l);
for (i in 1:l)
{p[i]=f(m,k,thetastar[i])
};
return(p);
}
## Calculating the pcs for negative multinomial
calc.LF.multinomial.is<-function(k,m,thetastar,ctype)
```

```
{if (ctype=="exact") p<-calc.LF.multinomial.is.exact(m,k,thetastar)
 else if (ctype=="approximate")
 p<-calc.LF.multinomial.is.approximate(m,k,thetastar);
 return(p);
}
## Calculating the pcs for the multinomial distributions
calc.LF.multinomial<-function(k,n,m,thetastar,stype,ctype)
{ if (stype=="fs") p<-calc.LF.multinomial.fs(k,n,thetastar,ctype)
 else if (stype=="is") p<-calc.LF.multinomial.is(k,m,thetastar,ctype)
 else stop ("Method is unavailable")
return(p);}
## Calculating the exact pcs for multivariate hypergeometric
calc.LF.hypergeom.fs.exact<-function(N,k,n,thetastar)
{f<-function(N,k,n,thetastar)
{
# Calculating the exact pcs when k is 2
if (k==2)
{ ex0<-function(i,N,k,n,thetastar)
{M2=i;
M1=N-i;
M=rbind(M1,M2);
x1=ceiling(n/2):n;
x2=n-x1;
x<-rbind(x2,x1);
lo<-array(c(x[1:k,]<=M[1:k,]),dim=c(2,ncol(x)));
d<-rep(0,ncol(x));
for(b in 1:ncol(x)){d[b]<-sign(sum(lo[,b])==2)};
e<-which(d==1);
x<-array(x[,e],dim=c(k,length(e)));
s<-rep(0,ncol(x));
for(j in 1:ncol(x))
{s[j]=1+sum(sign(x[1:(k-1),j]==x[k,j]))
};
p=0;
for (i in 1:ncol(x))
{ p=p+1/s[i]*exp(sum(lgamma(M+1))-sum(lgamma(x[,i]+1))
 -sum(lgamma(M-x[,i]+1))-lgamma(N+1)+lgamma(n+1)+lgamma(N-n+1))
#prod(choose(M,x[,i]))/choose(N,n);
}
result<-rbind(M,p);
return(result);
```

```
}
Y<-array(rep(NA,k+1),dim=c(k+1,1));
for (i in ceiling(thetastar*N/(thetastar+k-1)):(N-k+1))
{ X<-ex0(i,N,k,n,thetastar);
Y<-cbind(Y,X);
}
Y=Y[,-1];
};
# Calculating the exact pcs when k is 3
if (k==3){
ex1<-function(i,N,k,n,thetastar){
x<-combn(N-i-1,1);
n0<-ncol(x);
y<-array(rep(0,(k-1)*n0),dim=c(k-1,n0));
y[1,]=x[1,]-0;
y[2,]=N-i-x[1,];
yk<-array(rep(i,n0),dim=c(1,n0));
M1<-rbind(y,yk);
lo1<-M1[1,]<=M1[2,];
lo2<-M1[1:2,]<=M1[k,]/thetastar;
lo1<-array(lo1,dim=c(1,length(lo1)));
lo2<-as.matrix(lo2);
lo<-rbind(lo1,lo2);
d<-rep(0,n0);
for(b in 1:n0){d[b]<-sign(sum(lo[,b])==(2*k-3))};
if (sum(d)!=0){
e<-which(d==1);
M<-array(M1[,e],dim=c(k,length(e)));
x1<-combn(n+k-1,k-1);
n1<-ncol(x1);
y1<-array(rep(0,k*n1),dim=c(k,n1));
y1[1,]=x1[1,]-0;
for (j in 2:(k-1)){
y1[j,]=x1[j,]-x1[j-1,];
}
y1[k,]=n+k-x1[k-1,];
y1=y1-1;
m<-ncol(M);
p<-rep(0,m);
for (i in 1:m){
lo3<-array(rep(NA,(k-1)*ncol(y1)),dim=c(k-1,ncol(y1)));
```

```
for (j in 1:(k-1)){
lo3[j,]<-y1[j,]<=y1[k,];}
lo4<-y1[1:k,]<=M[1:k,i];
lo5<-rbind(lo3,lo4);
d<-rep(0,ncol(y1));
for(b in 1:ncol(y1)){d[b]<-sign(sum(lo5[,b])==(2*k-1))};
e<-which(d==1);
y2<-array(y1[,e],dim=c(k,length(e)));
s<-rep(0,ncol(y2));
for(j in 1:ncol(y2))
{s[j]=1+sum(sign(y2[1:(k-1),j]==y2[k,j]))
};
for (j in 1:ncol(y2)){
p[i]=p[i]+1/s[j]*exp(sum(lgamma(M[,i]+1))-sum(lgamma(y2[,j]+1))
 -sum(lgamma(M[,i]-y2[,j]+1))-lgamma(N+1)+lgamma(n+1)+lgamma(N-n+1))
#prod(choose(M[,i],y2[,j]))/choose(N,n);
}
}
p<-array(p,dim=c(1,m))
result<-rbind(M,p);
}
else result=array(rep(0,k+1),dim=c(k+1,1));
return(result);
}
Y<-array(rep(NA,k+1),dim=c(k+1,1));
for (i in ceiling(thetastar*N/(thetastar+k-1)):(N-k+1))
{ X<-ex1(i,N,k,n,thetastar);
Y<-cbind(Y,X);
}
Y=as.matrix(Y[,-1]);
Y=Y[,which(apply(Y,2,sum)!=0)];
};
# Calculating the exact pcs when k is greater than 3
if (k>=4){
ex2<-function(i,N,k,n,thetastar){
x<-combn(N-i-1,k-2);
n0<-ncol(x);
y<-array(rep(0,(k-1)*n0),dim=c(k-1,n0));
y[1,]=x[1,]-0;
for (j in 2:(k-2)){
y[j,]=x[j,]-x[j-1,];}
```

```
y[k-1,]=N-i-x[k-2,];
yk<-array(rep(i,n0),dim=c(1,n0));
M1<-rbind(y,yk);
lo1<-M1[1:(k-2),]<=M1[2:(k-1),];
lo2<-M1[1:(k-1),]<=M1[k,]/thetastar;
lo1<-as.matrix(lo1);
lo2<-as.matrix(lo2);
lo<-rbind(lo1,lo2);
d<-rep(0,n0);
for(b in 1:n0){d[b]<-sign(sum(lo[,b])==(2*k-3))};
if (sum(d)!=0){
e<-which(d==1);
M<-array(M1[,e],dim=c(k,length(e)));
x1<-combn(n+k-1,k-1);
n1<-ncol(x1);
y1<-array(rep(0,k*n1),dim=c(k,n1));
y1[1,]=x1[1,]-0;
for (j in 2:(k-1)){
y1[j,]=x1[j,]-x1[j-1,];}
y1[k,]=n+k-x1[k-1,];
y1=y1-1;
m<-ncol(M);
p<-rep(0,m);
for (i in 1:m){
lo3<-array(rep(NA,(k-1)*ncol(y1)),dim=c(k-1,ncol(y1)));
for (j in 1:(k-1)){
lo3[j,]<-y1[j,]<=y1[k,];}
lo4<-y1[,1:ncol(y1)]<=M[,i];
lo5<-rbind(lo3,lo4);
d<-rep(0,ncol(y1));
for(b in 1:ncol(y1)){d[b]<-sign(sum(lo5[,b])==(2*k-1))};
e<-which(d==1);
y2<-array(y1[,e],dim=c(k,length(e)));
s<-rep(0,ncol(y2));
for(j in 1:ncol(y2))
{s[j]=1+sum(sign(y2[1:(k-1),j]==y2[k,j]))};
for (j in 1:ncol(y2)){
p[i]=p[i]+1/s[j]*exp(sum(lgamma(M[,i]+1))-sum(lgamma(y2[,j]+1))
 -sum(lgamma(M[,i]-y2[,j]+1))-lgamma(N+1)+lgamma(n+1)+lgamma(N-n+1))
#prod(choose(M[,i],y2[,j]))/choose(N,n);}}
p<-array(p,dim=c(1,m))
```

```
result<-rbind(M,p);}
else result=array(rep(0,k+1),dim=c(k+1,1));
return(result);
}
Y<-array(rep(NA,k+1),dim=c(k+1,1));
for (i in ceiling(thetastar*N/(thetastar+k-1)):(N-k+1))
{ X<-ex2(i,N,k,n,thetastar);
Y<-cbind(Y,X);}
Y=as.matrix(Y[,-1]);
Y=Y[,which(apply(Y,2,sum)!=0)];};
Y<-as.matrix(Y);
w<-which(Y[k+1,]==min(Y[k+1,]));
if (sum(sign(w))>1) Z=Y[,min(w)]
else Z=Y[1:(k+1),which(Y[k+1,]==min(Y[k+1,]))];
return (Z);}
if (N<n) stop(" 'N' must be greater than or equal to 'n' ")
else {l=length(thetastar);
Y<-array(rep(NA,(k+1)*l),dim=c(k+1,l))
for (i in 1:l)
{Y[,i]<-f(N,k,n,thetastar[i]);};
return(t(as.matrix(Y)));}}
## Calculating all the possible M[i]'s
M=function(N,k,thetastar){
# Calculating all the possible M[i]'s when k is 2
if (k==2)
{ex0<-function(i,N,k,thetastar)
{M2=i;
M1=N-i;
M=rbind(M1,M2);
return(M);}
Y<-array(rep(NA,k),dim=c(k,1));
for (i in ceiling(thetastar*N/(thetastar+k-1)):(N-k+1))
{ X<-ex0(i,N,k,thetastar);
Y<-cbind(Y,X);}
Y=Y[,-1];
};
# Calculating all the possible M[i]'s when k is 3
if (k==3)
{ex1<-function(i,N,k,thetastar){
x<-combn(N-i-1,1);
n<-ncol(x);
```

```
y<-array(rep(0,(k-1)*n),dim=c(k-1,n));
y[1,]=x[1,]-0;
y[2,]=N-i-x[1,];
yk<-array(rep(i,n),dim=c(1,n));
M1<-rbind(y,yk);
lo1<-M1[1,]<=M1[2,];
lo2<-M1[1:2,]<=M1[k,]/thetastar;
lo1<-array(lo1,dim=c(1,length(lo1)));
lo2<-as.matrix(lo2);
lo<-rbind(lo1,lo2);
d<-rep(0,n);
for(b in 1:n){d[b]<-sign(sum(lo[,b])==(2*k-3))};
if (sum(d)!=0){
e<-which(d==1);
M<-array(M1[,e],dim=c(k,length(e)));
return(M);}}
Y<-array(rep(NA,k),dim=c(k,1));
for (i in ceiling(thetastar*N/(thetastar+k-1)):(N-k+1))
{ X<-ex1(i,N,k,thetastar);
Y<-cbind(Y,X);}
Y=as.matrix(Y[,-1]);
};
# Calculating all the possible M[i]'s when k is greater than 3
if (k>=4)
{ex2<-function(i,N,k,thetastar){
x<-combn(N-i-1,k-2);
n<-ncol(x);
y<-array(rep(0,(k-1)*n),dim=c(k-1,n));
y[1,]=x[1,]-0;
for (j in 2:(k-2)){
y[j,]=x[j,]-x[j-1,];}
y[k-1,]=N-i-x[k-2,];
yk<-array(rep(i,n),dim=c(1,n));
M1<-rbind(y,yk);
lo1<-M1[1:(k-2),]<=M1[2:(k-1),];
lo2<-M1[1:(k-1),]<=M1[k,]/thetastar;
lo1<-as.matrix(lo1);
lo2<-as.matrix(lo2);
lo<-rbind(lo1,lo2);
d<-rep(0,n);
for(b in 1:n){d[b]<-sign(sum(lo[,b])==(2*k-3))};
```

```
if (sum(d)!=0){
e<-which(d==1);
M<-array(M1[,e],dim=c(k,length(e)));
return(M);}}
Y<-array(rep(NA,k),dim=c(k,1));
for (i in ceiling(thetastar*N/(thetastar+k-1)):(N-k+1))
{ X<-ex2(i,N,k,thetastar);
Y<-cbind(Y,X);}
Y=as.matrix(Y[,-1]);};
return(Y);
}
## Calculating the approximate pcs for multivariate hypergeometric
calc.LF.hypergeom.fs.approximate<-function(N,k,n,thetastar)
{f<-function(N,k,n,thetastar){
p.arc<-function(M,N,k,n)
{f0<-function(x){2*asin(sqrt(x))};
f1<-function(x){1/(sqrt(x*(1-x)))};
A<-rep(NA,k-1);
for (i in 1:(k-1)){
A[i]<-f0(M[k]/N)-f0(M[i]/N)};
B<-rep(NA,k-1);
for (i in 1:(k-1)){
B[i]<-(f1(M[k]/N))^2*M[k]/N*(1-M[k]/N)*n*(N-n)/(N-1)
+(f1(M[i]/N))^2*M[i]/N*(1-M[i]/N)*n*(N-n)/(N-1)
+2*f1(M[k]/N)*f1(M[i]/N)*n*M[i]*M[k]/N^2*(N-n)/(N-1);
B[i]<-B[i]/n^2;};
C<-array(rep(NA,(k-1)*(k-1)),dim=c(k-1,k-1));
for (i in 1:(k-1))
{for (j in 1:(k-1)){
C[i,j]<-(f1(M[k]/N))^2*M[k]/N*(1-M[k]/N)*n*(N-n)/(N-1)
+f1(M[k]/N)*f1(M[i]/N)*n*M[i]*M[k]/N^2*(N-n)/(N-1)
+f1(M[k]/N)*f1(M[j]/N)*n*M[j]*M[k]/N^2*(N-n)/(N-1)
-f1(M[i]/N)*f1(M[j]/N)*n*M[i]*M[j]/N^2*(N-n)/(N-1);
C[i,j]<-C[i,j]/n^2/sqrt(B[i]*B[j]);
}}
for (i in 1:(k-1)){
C[i,i]<-1;}
lower<--A/sqrt(B);
if (k==2)
{p<-1-pnorm(lower);
} else {
```

```
p<-pmvnorm(lower,corr=C);}
return(p[1]);}
M<-M(N,k,thetastar);
n1<-ncol(M);
p<-rep(NA,n1);
for (l in 1:n1){
p[l]<-p.arc(M[,l],N,k,n)}
M0<-M[,which(p==min(p))];
return(c(M0,min(p)));}
l=length(thetastar);
Y<-array(rep(NA,(k+1)*l),dim=c(k+1,l))
for (i in 1:l)
{Y[,i]<-f(N,k,n,thetastar[i]);
};
return(t(as.matrix(Y)));
}
## Calculating the pcs for multivariate hypergeometric
calc.LF.hypergeom.fs<-function(N,k,n,thetastar,ctype)
{if (ctype=="exact") p<-calc.LF.hypergeom.fs.exact(N,k,n,thetastar)
 else if (ctype=="approximate")
 p<-calc.LF.hypergeom.fs.approximate(N,k,n,thetastar);
return(p);
}
## Calculating the exact pcs for multivariate inverse hypergeometric
calc.LF.hypergeom.is.exact<-function(N,k,thetastar,m)
{f<-function(N,k,thetastar,m)
{
is.P<-function(M,m,k){
phik=0;
phi0=0;
phi=0;
f<-function(list,m)
{exp(sum(lgamma(M+1))-lgamma(m)-sum(lgamma(list+1))
-lgamma(M[k]-m+2)-sum(lgamma(M[1:(k-1)]-list+1))
-lgamma(sum(M)+1)+lgamma(m+sum(list))
+lgamma(sum(M)-m+2-sum(list)))*(M[k]-m+1)/(sum(M)-m+1-sum(list))}
cond<-function(x,list){return(c(x,list))};
sss<-function(i,list)
{if (i>(k-1)) phi0=0
else {for (j in (0:min(m-1,M[i])))
{phi<-phi+ss(i,cond(list,j));}}
```

```
return(phi0+phi)
};
ss<-function(i,list)
{if (length(list)==k-1){ss=f(list,m)}
else ss=sss(i+1,list);
return(ss);}
phik=phik+sss(1,c());
return(phik);}
Y=M(N,k,thetastar)
Y<-Y[,which(m<=Y[k,])];
n<-ncol(as.matrix(Y));
if (n==0)
# Check that m is no greater than M[i]'s
stop(cat("m must be less than or equal to",paste("M",k,sep=""),"\n"))
else {
Y<-as.matrix(Y);
p=rep(NA,n);
for (i in 1:n)
{p[i]=is.P(Y[,i],m,k)
}
Z=rbind(Y,p);
Y=as.matrix(Z);
w<-which(Y[k+1,]==min(Y[k+1,]));
if (sum(sign(w))>1) Z=Y[,min(w)]
else Z=Y[1:(k+1),which(Y[k+1,]==min(Y[k+1,]))];
return (Z);
}}
l=length(thetastar);
Y<-array(rep(NA,(k+1)*l),dim=c(k+1,l))
for (i in 1:l)
{Y[,i]<-f(N,k,thetastar[i],m);
};
return(t(as.matrix(Y)));
}
## Calculating the approximate pcs for multivariate inverse hypergeometric
calc.LF.hypergeom.is.approximate<-function(N,k,m,thetastar)
{f<-function(N,k,thetastar,m){
p.a<-function(M,m,k){
A<-rep(NA,k-1);
for (i in 1:(k-1)){
A[i]<-log(M[i]/(M[k]+1));}
```

```
B<-rep(NA,k-1);
for (i in 1:(k-1)){
B[i]<-1/m*1/exp(A[i])^2*M[i]/(M[k]+1)*(1+M[i]/(M[k]+1))*(M[k]-m+1)/(M[k]+2);}
C<-array(rep(NA,(k-1)^2),dim=c(k-1,k-1));
for (i in 1:(k-1)){
for (j in 1:(k-1)){
C[i,j]<-sqrt(M[i]*M[j]/((M[k]+M[i]+1)*(M[k]+M[j]+1)))}
}
for (i in 1:(k-1)){
C[i,i]=1}
temp<-array(NA,k-1);
for (i in 1:(k-1))
{temp[i]<-min(m,M[i])}
upper<-as.vector((log(temp/m)-A)/sqrt(B));
if (k==2)
{p<-pnorm(upper);
} else {
p<-pmvnorm(lower=-Inf,upper=upper,corr=C);}
return(p[1]);
}
M<-M(N,k,thetastar);
n<-ncol(M);
p<-rep(NA,n);
for (l in 1:n){
if (M[,l][k]>=m) p[l]<-p.a(M[,l],m,k)
else p[l]<-p.a(M[,l],M[,l][k],k)
}
M0<-M[,which(p==min(p))];
return(c(M0,min(p)));
}
l=length(thetastar);
Y<-array(rep(NA,(k+1)*l),dim=c(k+1,l))
for (i in 1:l)
{Y[,i]<-f(N,k,thetastar[i],m);
};
return(t(as.matrix(Y)));
}
## Calculating the pcs for multivariate inverse hypergeometric
calc.LF.hypergeom.is<-function(N,k,m,thetastar,ctype)
{if (ctype=="exact") p<-calc.LF.hypergeom.is.exact(N,k,thetastar,m)
 else if (ctype=="approximate")
```

```
p<-calc.LF.hypergeom.is.approximate(N,k,m,thetastar);
return(p);
}
## Calculating the pcs for the multivariate hypergeometric distributions
calc.LF.hypergeom<-function(N,k,n,m,thetastar,stype,ctype)
{  if (stype=="fs") p<-calc.LF.hypergeom.fs(N,k,n,thetastar,ctype)
 else if (stype=="is") p<-calc.LF.hypergeom.is(N,k,m,thetastar,ctype)
 else stop ("Method is unavailable")
return(p);
}
## Calculating the exact expected sample size for negative multinomial
LF.e<-function(N,k,thetastar)
{p<-rep(NA,k);
for (i in 1:(k-1)){
p[i]=1/(thetastar+k-1)};
p[k]=thetastar/(thetastar+k-1);
m=k-1;
exact<-function(M,N,a,m){
phik=0;
phi0=0;
phi=0;
f<-function(list){exp(M*log(p[1]/a)+lgamma(M+sum(list))-lgamma(M)
+sum(list)*log(p[1])-sum(lgamma(list+1)))}
cond<-function(x,list){return(c(x,list))};
sss<-function(i,list)
{if (i>m) phi0=0
else {for (j in (0:(N-1)))
{phi<-phi+ss(i,cond(list,j));}}
return(phi0+phi)};
ss<-function(i,list)
{if (length(list)==m)ss=f(list)
else ss=sss(i+1,list);
return(ss);}
phik=phik+sss(1,c());
return(phik)
};
q0=p[1];
p0=p[k];
q1=q0/(p0+q0);
if (k>2){
b<-dbinom(N,2*N,q1);
```

```
D1<-exact(N,N,1/thetastar,m);
D2<-exact(2*N,N,q1,m-1);
LF<-N/q0*(1-(p0-q0)/p0*D1-1/(2*p0)*b*D2)
};
if(k==2){
if (N>=2){
I1=0;
I2=0;
for (i in 0:(N-2))
{
I1=I1+p[k]^(N+1)*exp(lgamma(N+i+1)-lgamma(N+1)-lgamma(i+1))*(1-p[k])^i;
I2=I2+p[1]^(N+1)*exp(lgamma(N+i+1)-lgamma(N+1)-lgamma(i+1))*(1-p[1])^i;
}
LF<-N*(1+p[1]/p[k]*I1+p[k]/p[1]*I2)
}
else LF=1;
}
return(LF)
}
## Calculating the approximate expected sample size for negative multinomial
LF.a<-function(N,p,k,thetastar)
{m=k-1;
f<-function(x)
{
 pnorm((x+sqrt(N)*log((1+thetastar)/2))/sqrt(2))^(m-1)*dnorm(x)
};
LF.a<-N*(m+thetastar)/thetastar*(p+thetastar*(1-p))
-sqrt(N/pi)*(m+thetastar)^2/(2*thetastar)*
(4*thetastar/(1+thetastar)^2)^N*integrate(f,-Inf,Inf)$value;
return(LF.a)
}
## Calculating the expected sample size for negative multinomial
EWT<-function(N,k,thetastar,ctype,p){
l=length(thetastar);
Y<-array(rep(NA,l),dim=c(l,1))
for (i in 1:l)
{if (ctype=="exact") Y[i]<-LF.e(N,k,thetastar[i])
else Y[i]<- LF.a(N,p[i],k,thetastar[i])
};
return(Y);
}
```

```
## Calculating the exact expected sample size for
## multivariate inverse hypergeometric
ET.e<-function(M,m,k){
is.P<-function(M,m,k,d){
phik=0;
phi0=0;
phi=0;
M0<-M[-d];
f<-function(list,m,d){exp(sum(lgamma(M[-d]+1))+lgamma(M[d]+2)
-lgamma(m+1)-lgamma(M[d]-m+2)-sum(lgamma(M[-d]-list+1))-sum(lgamma(list+1))
-lgamma(sum(M)+2)+lgamma(sum(list)+m+1)
+lgamma(sum(M)-m+2-sum(list)))*(M[d]+1-m)/(sum(M)+1-m-sum(list))};
cond<-function(x,list){return(c(x,list))};
sss<-function(i,list)
{if (i>(k-1)) {phi0=0}
else {for (j in 0:min(m-1,M0[i])) phi<-phi+ss(i,cond(list,j));}
return(phi0+phi);
};
ss<-function(i,list)
{if (length(list)==k-1){ss=f(list,m,d)}
else ss=sss(i+1,list);
return(ss);
}
phik=phik+sss(1,c());
return(phik);
}
p<-rep(NA,k);
for (d in 1:k)
{if (M[d]>=m-1) p[d]<- m*(sum(M)+1)/(M[d]+1)*is.P(M,m,k,d)
 else p[d]=0
}
return(sum(p))
}
## Calculating the approximate expected sample size
## for multivariate inverse hypergeometric
ET.a<-function(M,m,k){
HD.norm<-function(M,m,k,d){
M[d]<-M[d]+1;
m<-m+1;
A<-M[-d]/(M[d]+1);
B<-(M[-d]/(M[d]+1))*(1+M[-d]/(M[d]+1))*(M[d]-m+1)/(M[d]+2)/m;
```

```
C<-array(rep(NA,(k-1)^2),dim=c(k-1,k-1));
M0<-M[-d];
C<-array(rep(NA,(k-1)^2),dim=c(k-1,k-1));
for (i in 1:(k-1)){
for (j in 1:(k-1)){
C[i,j]<-sqrt(M0[i]*M0[j]/((M[d]+M0[i]+1)*(M[d]+M0[j]+1)))
}
}
for (i in 1:(k-1)){
C[i,i]=1
}
upper<-as.vector((1-1/m-A)/sqrt(B));
lower<-as.vector(-A/sqrt(B));
if (k==2)
{p<-pnorm(upper)-pnorm(lower);
} else {
p<-pmvnorm(lower=-A/sqrt(B),upper=upper,corr=C);
}
return(p[1]);
}
p<-rep(NA,k);
for (d in 1:k)
{p[d]<- m*(sum(M)+1)/(M[d]+1)*HD.norm(M,m,k,d)
}
return(sum(p))
}
## Calculating the expected sample size
## for multivariate inverse hypergeometric
ETW<-function(M,m,k,ctype)
{if (length(M)==k) l=1 else l<-nrow(M);
Y<-array(rep(NA,l),dim=c(l,1))
for (i in 1:l)
{if (ctype=="exact") {if (l==1) Y[i]<-ET.e(M,m,k)
 else Y[i]<-ET.e(M[i,],m,k)} else {if (l==1) Y[i]<-ET.a(M,m,k)
 else Y[i]<-ET.a(M[i,],m,k)}
};
return(Y);
}
## Create show function
setMethod("show", "LF2c",
function(object){
```

```
cat("\n\n");
if (object@dtype=="multinomial"&object@stype=="fs")
 cat("Selecting the Most Probable Event for Multinomial","\n\n")
if (object@dtype=="multinomial"&object@stype=="is")
 cat("Selecting the Most Probable Event for Negative Multinomial","\n\n")
if (object@dtype=="hypergeom"&object@stype=="fs")
 cat("Selecting the Most Probable Event for Multivariate Hypergeometric",
  "\n\n")
if (object@dtype=="hypergeom"&object@stype=="is")
 cat("Selecting the Most Probable Event for Multivariate Inverse
 Hypergeometric","\n\n")
cat("Number of categories k: =",object@k,"\n");
 cat("Value(s) of thetastar: ");
if (length(object@thetastar)>1)
cat(paste(object@thetastar[1(length(object@thetastar)-1)],c(),sep=","),
 object@thetastar[length(object@thetastar)],"\n")
else cat(object@thetastar,"\n");
if (object@dtype=="multinomial"&object@stype=="fs"){
 cat("Sample size: n =",object@n,"\n");}
if (object@dtype=="multinomial"&object@stype=="is"){
 cat("Cell quota: m =",object@m,"\n");}
if (object@dtype=="hypergeom"&object@stype=="fs"){
cat("Population size: N =",object@N,"\n");
cat("Sample size: n =",object@n,"\n");}
if (object@dtype=="hypergeom"&object@stype=="is"){
cat("Population size: N =",object@N,"\n");
cat("Cell quota: m =",object@m,"\n")}
         })
## Create summary function
setMethod("summary", "LF2c",
function(object){
cat("\n\n");
if (object@dtype=="multinomial"&object@stype=="fs")
 cat("Selecting the Most Probable Event for Multinomial","\n\n")
if (object@dtype=="multinomial"&object@stype=="is")
 cat("Selecting the Most Probable Event for Negative Multinomial","\n\n")
if (object@dtype=="hypergeom"&object@stype=="fs")
 cat("Selecting the Most Probable Event for Multivariate Hypergeometric",
  "\n\n")
if (object@dtype=="hypergeom"&object@stype=="is")
 cat("Selecting the Most Probable Event for Multivariate Inverse
```

```
 Hypergeometric","\n\n")
cat("Number of categories k: =",object@k,"\n");
# Displaying pcs details
if (object@dtype=="multinomial"&object@stype=="fs"){
 cat("Sample size: n =",object@n,"\n");}
 if (object@dtype=="multinomial"&object@stype=="is"){
 cat("Cell quota: m =",object@m,"\n");}
 if (object@dtype=="hypergeom"&object@stype=="fs"){
 cat("Population size: N =",object@N,"\n");
 cat("Sample size: n =",object@n,"\n");}
 if (object@dtype=="hypergeom"&object@stype=="is"){
 cat("Population size: N =",object@N,"\n");
 cat("Cell quota: m =",object@m,"\n")}
 cat("\n","Detailed Probability(s) of Correct Selection","\n\n")
 if (object@dtype=="hypergeom"&object@ctype=="approximate"){
   if (length(object@thetastar)==1){
      if (object@stype=="is"){
       M<-array(object@M,dim=c(1,object@k));
       x<-cbind(as.vector(object@thetastar),as.vector(object@pcs),
       M,as.vector(object@EWT));
       colnames(x)<-c("Thetastar","P(CS).approximate",
       paste("M",1:length(object@M),sep=""),"EWT");
       rownames(x)<-rep("", length(object@pcs));show(x)}
      else{ M<-array(object@M,dim=c(1,object@k));
       x<-cbind(as.vector(object@thetastar),as.vector(object@pcs),M);
       colnames(x)<-c("Thetastar ","P(CS).approximate",
       paste("M",1:length(object@M),sep=""));
       rownames(x)<-rep("", length(object@pcs));show(x)} }
   else {
    if (object@stype=="is"){
       x<-cbind(as.vector(object@thetastar),as.vector(object@pcs),
       as.matrix(object@M),as.vector(object@EWT));
       colnames(x)<-c("Thetastar","P(CS).approximate",
       paste("M",1:ncol(object@M),sep=""),"EWT");
       rownames(x)<-rep("", length(object@pcs));show(x)}
    else {x<-cbind(as.vector(object@thetastar),as.vector(object@pcs),
          as.matrix(object@M));
          colnames(x)<-c("Thetastar","P(CS).approximate",
          paste("M",1:ncol(object@M),sep=""));
          rownames(x)<-rep("", length(object@pcs));show(x)}}}
   else if (object@dtype=="hypergeom"&object@ctype=="exact"){
```

```
if (length(object@thetastar)==1){
 if (object@stype=="is"){
   M<-array(object@M,dim=c(1,object@k));
   x<-cbind(as.vector(object@thetastar),as.vector(object@pcs),
      M,as.vector(object@EWT));
   colnames(x)<-c("Thetastar ","P(CS).exact",
   paste("M",1:length(object@M),sep=""),"EWT");
   rownames(x)<-rep("", length(object@pcs));show(x)}
 else {M<-array(object@M,dim=c(1,object@k));
   x<-cbind(as.vector(object@thetastar),as.vector(object@pcs),M);
   colnames(x)<-c("Thetastar ","P(CS).exact",
   paste("M",1:length(object@M),sep=""));
   rownames(x)<-rep("", length(object@pcs));show(x)}}
 else {if (object@stype=="is"){
   x<-cbind(as.vector (object@thetastar),as.vector(object@pcs),
   as.matrix(object@M),as.vector(object@EWT));
   colnames(x)<-c("Thetastar ","P(CS).exact",
   paste("M",1:ncol(object@M),sep=""),"EWT");
   rownames(x)<-rep("", length(object@pcs));show(x)}
else {x<-cbind(as.vector(object@thetastar),as.vector(object@pcs),
      as.matrix(object@M));
     colnames(x)<-c("Thetastar","P(CS).exact",
     paste("M",1:ncol(object@M),sep=""));
     rownames(x)<-rep("", length(object@pcs));show(x)}}}
     else if (object@dtype=="multinomial"&object@ctype=="approximate")
    {
  if (object@stype=="is"){
   x<-cbind(as.vector(object@thetastar),as.vector(object@pcs),
      as.vector(object@EWT));
   colnames(x)<-c("Thetastar ","P(CS).approximate",
                "EWT");
   rownames(x)<-rep(" ", length(object@pcs));show(x)}
  else {x<-cbind(as.vector(object@thetastar),as.vector(object@pcs));
   colnames(x)<-c("Thetastar ","P(CS).approximate");
   rownames(x)<-rep(" ", length(object@pcs));show(x)}}
   else if (object@dtype=="multinomial"&object@ctype=="exact") {
   if (object@stype=="is"){
    x<-cbind(as.vector(object@thetastar),as.vector(object@pcs),
       as.vector(object@EWT));
    colnames(x)<-c("Thetastar ","P(CS).exact", "EWT");
    rownames(x)<-rep(" ", length(object@pcs));show(x)}
```

```
        else {x<-cbind(as.vector(object@thetastar),as.vector(object@pcs));
              colnames(x)<-c("Thetastar ","P(CS).exact");
              rownames(x)<-rep(" ", length(object@pcs));show(x)}}
          })
## Create a new generic function for plot
setMethod("plot",signature(x="LF.multinomial",y="missing"),
          function(x,y,type="o",ylim=c(min(x@pcs),max(x@pcs)),
          main = main.plot(x),...)
           {
            plot(x@thetastar,x@pcs,type=type,xlab="thetastar",
         ylab="Probability of Correct Selection", ylim=ylim,main=main,...)
            }
         )
setMethod("plot",signature(x="LF.hypergeom",y="missing"),
          function(x,y,type="p",ylim=c(min(x@pcs),max(x@pcs)),
          main = main.plot(x),...)
           {
            plot(x@thetastar,x@pcs,type=type,xlab="thetastar",
         ylab="Probability of Correct Selection", ylim=ylim,main=main,...)
           }
         )
##The function for plot title
main.plot<-function(x)
  {
  if(x@dtype=="multinomial")
   {
    if (x@stype=="fs")
     return (paste("Multinomial PCS Curve with \n n =",x@n,", k = ",x@k,",
      ctype = ",x@ctype))
    else return(paste("Negative Multinomial PCS Curve with \n m =",x@m,",
      k = ",x@k,", ctype = ",x@ctype))
  }
  else
   {
    if(x@stype=="fs")
     return(paste("Hypergeometric PCS Curve with \n N =",x@N,", k = ",x@k,",
      n = ",x@n,", ctype = ",x@ctype))
    else
     return(paste("Negative Hypergeometric PCS Curve with \n N =",x@N,",
      k = ",x@k,", m = ",x@m,", ctype = ",x@ctype))
   }
```

```
  }
```

A.2 find.n.R

The following code is to find the smallest sample size (or cell quota and expected sample size) to meet the requirement that $pcs \geq P^{\star}$. Functions to calculate the approximate expected waiting time are also created.

```
## Finding the smallest sample size or cell quota
find.n<- function(dtype=c("multinomial","hypergeom"),stype=c("fs","is"),
ctype=c("exact","approximate"),p,k,thetastar,N)
{dtype<-match.arg(dtype);
 stype<-match.arg(stype);
 ctype<-match.arg(ctype);
  if (missing(p)|missing(k)|missing(thetastar))
  stop("p, k and thetastar must be provided")
  # Check p,k,thetastar are reasonable
  else if (p>1|p<0)
   stop("Probability of correct selection out of bound")
  # Check p is no less than 0 and no greater than 1
  else if (k<=1|thetastar<=1)
   stop("k and thetastar must be greater than 1")
  # Check k and thetastar are greater than 1
  else if (k!=as.integer(k))
   stop(" k must be an integer")
  # Check k is an integer
if (dtype=="multinomial")
{
# Finding the exact smallest sample size for multinomial
 if (stype=="fs"&ctype=="exact" )
   {
    N<-function(p,k,thetastar){
```

```
    n=1;
    ctype="exact";
    repeat
    {
     temp<-calc.LF.multinomial.fs(k,n,thetastar,ctype);
     if (temp<=p) n<-n+1
     else return(n);
    }
   }
result<-c();
result$p=p;
result$k=k;
result$thetastar=thetastar;
result$n.exact=N(p,k,thetastar);
result$pcs.exact=calc.LF.multinomial.fs.exact(k,result$n.exact,thetastar)
x<-cbind(as.vector(thetastar),as.vector(p),as.vector(result$pcs.exact));
colnames(x)<-c("Thetastar ","P.star", "P(CS).exact");
rownames(x)<-rep(" ", length(p));
cat("Finding the exact sample size for multinomial", "\n\n")
cat("Number of categories: k =",k , "\n" )
cat("Sample size: n.exact =", N(p,k,thetastar), "\n\n" )
show(x);
return(invisible(result))
  }
# Finding the approximate smallest sample size for multinomial
 if (stype=="fs"&ctype=="approximate")
   {
    N<-function(p,k,thetastar){
    p0<-rep(NA,k);
    for (i in 1:(k-1)){
    p0[i]=1/(thetastar+k-1)};
    p0[k]=thetastar/(thetastar+k-1);
    p1<-p0[k];
    q1<-p0[1];
    f0<-function(x){2*asin(sqrt(x))};
    f1<-function(x){1/(sqrt(x*(1-x)))};
    A<-f0(p1)-f0(q1);
    B<-(f1(p1)^2*p1+f1(q1)^2*q1-(p1*f1(p1)-q1*f1(q1))^2);
    C<-(f1(p1)^2*p1-(p1*f1(p1)-q1*f1(q1))^2);
    corr.ij<-C/B;
    corr<-array(rep(corr.ij,(k-1)*(k-1)),dim=c(k-1,k-1));
```

```
    for (i in 1:(k-1))
    {
    corr[i,i]=1
     };
    if (k==2)
     {
      temp<-qnorm(p,lower.tail =FALSE);
      m<-temp^2*B/A^2;
      }
    else
     {
      temp<-qmvnorm(p,corr=corr, tail = "upper.tail")$quantile;
      m<-(temp^2*B)/A^2;}
      return(ceiling(m));}
      result<-c();
result$p=p;
result$k=k;
result$thetastar=thetastar;
result$n.approximate=N(p,k,thetastar);
result$pcs.approximate=calc.LF.multinomial.fs.approximate(k,
result$n.approximate,thetastar)
x<-cbind(as.vector(thetastar),as.vector(p),as.vector(result$pcs.approximate))
colnames(x)<-c("Thetastar ","P.star", "P(CS).approximate");
rownames(x)<-rep(" ", length(p));
cat("Finding the approximate sample size for multinomial", "\n\n")
cat("Number of categories: k =",k , "\n" )
cat("Sample size: n.approximate =", N(p,k,thetastar), "\n\n" )
show(x);
return(invisible(result))
      }
# Finding the exact cell quota for negative multinomial
 if (stype=="is"&ctype=="exact")
   {
    N<-function(p,k,thetastar){
    m=1;
    repeat
    {
     temp<-calc.LF.multinomial.is.exact(m,k,thetastar);
     if (temp<=p) m<-m+1
     else return(m);
     }
```

```
     }
result<-c();
result$p=p;
result$k=k;
result$thetastar=thetastar;
result$m.exact=N(p,k,thetastar);
result$expected.n.exact=LF.e(result$m.exact,k,thetastar);
result$pcs.exact=calc.LF.multinomial.is.exact(result$m.exact,k,thetastar)
x<-cbind(as.vector(thetastar),as.vector(p),as.vector(result$pcs.exact));
colnames(x)<-c("Thetastar ","P.star", "P(CS).exact");
rownames(x)<-rep(" ", length(p));
cat("Finding the exact cell quota for negative multinomial", "\n\n")
cat("Number of categories: k =",k , "\n" )
cat("Cell quota: m.exact =", result$m.exact, "\n" )
cat("Expected sample size: expected.n.exact =",result$expected.n.exact)
cat("\n\n" )
show(x);
return(invisible(result))
    }
# Finding the approximate cell quota for negative multinomial
 if (stype=="is"&ctype=="approximate")
   {
    N.approximate<-function(p,k,thetastar){
    p0<-rep(NA,k);
    for (i in 1:(k-1)){
    p0[i]=1/(thetastar+k-1)};
    p0[k]=thetastar/(thetastar+k-1);
    l<-p0[1]/p0[k];
    A<-l;
    B<-l*(1+l);
    corr.ij<-l/(1+l);
    J<-array(rep(corr.ij,(k-1)*(k-1)),dim=c(k-1,k-1));
    for (i in 1:(k-1))
    {
     J[i,i]=1
     };
    if (k==2)
    {
     temp<-qnorm(p,lower.tail =TRUE);
     m0<-temp^2*B/(1-A)^2;
     }
```

```
    else
    {
     temp<-qmvnorm(p,corr=J, tail="upper.tail")$quantile;
     m0<-(temp^2*B)/(1-A)^2;}
     return(ceiling(m0));
     }
   result<-c();
result$p=p;
result$k=k;
result$thetastar=thetastar;
result$m.approximate=N.approximate(p,k,thetastar);
result$expected.n.approximate=LF.a(result$m.approximate,p,k,thetastar)
result$pcs.approximate
=calc.LF.multinomial.is.approximate(result$m.approximate,k,thetastar)
x<-cbind(as.vector(thetastar),as.vector(p),
as.vector(result$pcs.approximate));
colnames(x)<-c("Thetastar ","P.star", "P(CS).approximate");
rownames(x)<-rep(" ", length(p));
cat("Finding the approximate cell quota for negative multinomial", "\n\n")
cat("Number of categories: k =",k , "\n" )
cat("Cell quota: m.approximate =", result$m.approximate, "\n" )
cat("Expected sample size: expected.n.approximate =")
cat(result$expected.n.approximate)
cat("\n\n" )
show(x);
return(invisible(result))
   }
 }
if (dtype=="hypergeom")
{
 # Check that N is no NA.
 if (missing(N))
 stop ("Population size(N) must be provided.")
 else{
# Finding the exact sample size for multivariate hypergeometric
  if (stype=="fs"&ctype=="exact" )
   {
    n<-function(p,N,k,thetastar)
     {
    for (n in (k+1):N)
      {
```

```
        if (calc.LF.hypergeom.fs.exact(N,k,n,thetastar)[k+1]<p) next
        else break
       }
   answer<-c(n,calc.LF.hypergeom.fs.exact(N,k,n,thetastar)[1:k])
   return(answer)
     }
result<-c();
result$N=N;
result$p=p;
result$k=k;
result$thetastar=thetastar;
result$n.exact=n(p,N,k,thetastar)[1];
result$M=n(p,N,k,thetastar)[2:(k+1)];
result$pcs.exact=calc.LF.hypergeom.fs.exact(N,k,result$n.exact,thetastar)[k+1]
M0<-array(result$M,dim=c(1,k));
x<-cbind(as.vector(thetastar),M0,as.vector(p),
as.vector(result$pcs.exact));
colnames(x)<-c("Thetastar ",paste("M",1:k,sep=""),"P.star",
 "P(CS).exact");
rownames(x)<-rep(" ", length(p));
cat("Finding the exact sample size for multivariate hypergeometric")
cat( "\n\n")
cat("Number of categories: k =",k , "\n" )
cat("Sample size: n.exact =", result$n.exact, "\n\n" )
show(x);
return(invisible(result))
   }
# Finding the approximate sample size for multivariate hypergeometric
  if (stype=="fs"&ctype=="approximate" )
   {
    n<-function(p,N,k,thetastar)
    {
    for (n in (k+1):N)
     {
      if (calc.LF.hypergeom.fs.approximate(N,k,n,thetastar)[k+1]<p) next
      else break
      }
    answer<-c(n,calc.LF.hypergeom.fs.approximate(N,k,n,thetastar)[1:k])
    return(answer)
     }
result<-c();
```

```
result$N=N;
result$p=p;
result$k=k;
result$thetastar=thetastar;
result$n.approximate<-n(p,N,k,thetastar)[1];
result$M=n(p,N,k,thetastar)[2:(k+1)];
result$pcs.approximate
=calc.LF.hypergeom.fs.approximate(N,k,result$n.approximate,thetastar)[k+1];
M0<-array(result$M,dim=c(1,k));
x<-cbind(as.vector(thetastar),M0,as.vector(p),
as.vector(result$pcs.approximate));
colnames(x)<-c("Thetastar",paste("M",1:k,sep=""),"P.star",
"P(CS).approximate");
rownames(x)<-rep(" ", length(p));
cat("Finding approximate sample size for multivariate hypergeometric")
cat("\n\n")
cat("Number of categories: k =",k , "\n" )
cat("Sample size: n.approximate =", result$n.approximate, "\n\n" )
show(x);
return(invisible(result))
      }
# Finding the exact cell quota for multivariate inverse hypergeometric
  if (stype=="is"&ctype=="exact")
   {
    m<-function(p,N,k,thetastar)
    {
     for (m in 1:N)
     {
      if (calc.LF.hypergeom.is.exact(N,k,thetastar,m)[k+1]<p) next
      else break
      }
     answer<-c(m,calc.LF.hypergeom.is.exact(N,k,thetastar,m)[1:k])
     return(answer)
     }
result<-c();
result$N=N;
result$p=p;
result$k=k;
result$thetastar=thetastar;
result$m.exact<-m(p,N,k,thetastar)[1];
result$expected.n.exact=ET.e(m(p,N,k,thetastar)[2:(k+1)],result$m.exact,k);
```

```
result$M=m(p,N,k,thetastar)[2:(k+1)];
result$pcs.exact
=calc.LF.hypergeom.is.exact(N,k,thetastar,result$m.exact)[k+1];
M0<-array(result$M,dim=c(1,k));
x<-cbind(as.vector(thetastar),M0,as.vector(p),as.vector(result$pcs.exact));
colnames(x)<-c("Thetastar ",paste("M",1:k,sep=""),"P.star", "P(CS).exact");
rownames(x)<-rep(" ", length(p));
cat("Finding the exact sample size for multivariate inverse hypergeometric")
cat("\n\n")
cat("Number of categories: k =",k , "\n" )
cat("Cell quota: m.exact =", result$m.exact, "\n" )
cat("Expected sample size: expected.n.exact =", result$expected.n.exact)
cat("\n\n" )
show(x);
return(invisible(result))
   }
# Finding the approximate cell quota for multivariate inverse hypergeometric
  if (stype=="is"&ctype=="approximate")
   {
    m<-function(p,N,k,thetastar)
    {
     for (m in 1:N)
     {
      if (calc.LF.hypergeom.is.approximate(N,k,m,thetastar)[k+1]<p) next
      else break
      }
    answer<-c(m,calc.LF.hypergeom.is.approximate(N,k,m,thetastar)[1:k])
    return(answer)
     }
result<-c();
result$N=N;
result$p=p;
result$k=k;
result$thetastar=thetastar;
result$m.approximate=m(p,N,k,thetastar)[1];
result$expected.n.approximate
=ET.a(m(p,N,k,thetastar)[2:(k+1)],result$m.approximate,k)
result$M=m(p,N,k,thetastar)[2:(k+1)];
result$pcs.approximate
=calc.LF.hypergeom.is.approximate(N,k,thetastar,result$m.approximate)[k+1];
M0<-array(result$M,dim=c(1,k));
```

```
x<-cbind(as.vector(thetastar),M0,as.vector(p),
as.vector(result$pcs.approximate));
colnames(x)<-c("Thetastar",paste("M",1:k,sep=""),"P.star",
"P(CS).approximate");
rownames(x)<-rep(" ", length(p));
cat("Finding the approximate sample size for multivariate
inverse hypergeometric");
cat("\n\n")
cat("Number of categories: k =",k , "\n" )
cat("Cell quota: m.approximate =", result$m.approximate, "\n" )
cat("Expected sample size: expected.n.approximate =")
cat(result$expected.n.approximate, "\n\n" )
show(x);
return(invisible(result))
    }
  }
 }
}
```

Appendix B

RS Package Help File

R help files are written "R documentation" (Rd) format, which is a special markup language that resembles LaTeX. The compiled **rs** package help file is given on the following pages.

Package ‘RS’

August 3, 2011

Title Selecting the most probable multinomial or multivariate hypergeometric category based on fixed or inverse sampling

Version 1.0.0

Date 2011-08-03

Author Aaron Childs, Hong Jin

Maintainer Aaron Childs<childsa@mcmaster.ca>, Hong Jin<0610050@gmail.com>

Description This package provides tools to calculate, plot and display the probabilities of correctly selecting the most probable category in a population with k>=2 categories. A function for finding the specified smallest sample size or cell quota is also provided.

Depends methods, R(>= 1.9.0), stats

Imports mvtnorm,graphics

License GPL (>= 3)

LazyLoad yes

R topics documented:

find.n 1
LF2c 2
LF2c-class 4

Index **6**

`find.n` *A function for finding the smallest sample size or expected sample size.*

Description

Find the smallest sample size using fixed sampling or the smallest expected sample size using inverse sampling such that the specified probability of correct selection is met.

Arguments

`dtype`	The type of distribution on which the sampling is based; namely, `multinomial` or `hypergeom`.
`stype`	The type of sampling; namely, `fs` (fixed Sampling) or `is` (inverse sampling).
`ctype`	The type of calculation; namely, `exact` or `approximate`.
`p`	The minimum probability of correct selection.
`k`	The number of categories in the population.
`thetastar`	For `dtype="multinomial"` this is the smallest ratio between the probabilities corresponding to the categories with the largest and second largest probabilities. For `dtype="hypergeom"` this is the smallest ratio between the category sizes corresponding to the categories with the largest and second largest sizes.
`N`	Only for `dtype="hypergeom"`. An integer giving the population size from which the sample is drawn.

Usage

find.n(dtype="multinomial",stype="fs",ctype="exact",p,k,thetastar)

find.n(dtype="hypergeom",stype="is",ctype="appoximate",p,k,thetastar,N)

Author(s)

Aaron Childs, Hong Jin

See Also

`LF2c`

`LF2c` *Selecting the multinomial or hypergeometric category which has the highest probability*

Description

The preferred way of creating new objects from the family of `LF2c` classes.

Usage

```
LF2c(dtype=c("multinomial","hypergeom"),stype=c("fs", "is"),ctype=
c("exact","approximate"),k,thetastar,...)
```

Arguments

`dtype`	The type of distribution on which the sampling is based; namely, `multinomial` or `hypergeom`.
`stype`	The type of sampling; namely, `fs` or `is`.
`ctype`	The type of calculation; namely, `exact` or `approximate`.
`k`	The number of categories in the population

`thetastar` For `dtype="multinomial"`, this is a vector whose components represent the smallest ratio between the probabilities corresponding to the categories with the largest and second largest probabilities.

For `dtype="hypergeom"`, this is a vector whose components represent the smallest ratio between the category sizes corresponding to the categories with largest and second largest sizes.

`...` Additional parameters passed to the class generating function for each type. See Details for options.

Details

Typical usages are:

```
LF2c(k,thetastar,n)
LF2c(ctype="approximate",k,thetastar,n)
LF2c(stype="is",k)
LF2c(stype="is",ctype="approximate",k,thetastar,)
LF2c(dtype="hypergeom",stype="",ctype,k,thetastar,N,n)
LF2c(dtype="hypergeom",stype="is",ctype,k,thetastar,N,m)
```

The first and second forms use a default `dtype` of `multinomial` and `stype` of `fs`. The first form provides a default `ctype` of `exact` and default values for `n` and `thetastar`. The second form uses approximate calculations.

The third and fourth forms use a default `dtype` of `multinomial` and optional `stype` of `is`. The third form provides a default `ctype` of `exact` and default values for `m` and `thetastar` (`m=8` and `thetastar=seq(1.1,2,0.05)`). The fourth form uses approximate calculations.

The fifth and sixth forms state are for a finite population, so the population size `N` needs to be specified. The fifth form uses fixed sampling, so the sample size `n` needs to be specified. And the sixth form uses inverse sampling, so the cell quota `m` is needed. In these cases, `thetastar` is the smallest ratio of the category sizes for the categories with largest and second largest size. If `N`, `n`, `m` and `thetastar` are not specified, they take default values of `N=100`, `n=20`, `m=8` and `thetastar=seq(1.1, 2, 0.05)`.

Value

An object from the family of `LF2c-class`, namely of class `LF.multinomial` or `LF.hypergeom`.

See Also

`LF2c-class`

Examples

```
## P(correct selection) based on the multinomial distribution using fix sampling
## and exact calculation.
x <- LF2c(k=3,thetastar=1.1,n=10)
x ## print out a brief summary
plot(x) ## Plot the PCS curve

## P(correct selection) based on the multinomial distribution using
## inverse sampling and approximate calculation.
x <- LF2c(stype="is",ctype="approximate",thetastar=seq(1.1,2,0.05),k=4,m=4)
```

```
x
plot(x) ## Plot the PCS curve

## P(correct selection) based on the hypergeometric distribution using
## fixed sampling and exact calculation.
x <- LF2c(dtype="hypergeom",stype="fs",ctype="approximate",k=3,
thetastar=seq(1.2,3,0.1),N=20,n=8)
plot(x)
summary(x)#The summary
```

`LF2c-class` *Class Family "LF2c".*

Description

The family `LF2c` class provides methods for finding probabilities of correct selection based on the multinomial (`"LF.multinomial"`) and hypergeometric (`"LF.hypergeom"`) distributions using the least favorable configuration.

Objects from the Class

The `LF2c` class is a virtual Class: No objects may be created from it.

However, objects from the derived classes can be created by calls of the form such as `new("LF.multinomial", ...)`, or preferably using the creator function `LF2c`.

Slots

`dtype`**:** Object of class `character`. The type of distribution on which the sampling is based; namely, `multinomial` or `hypergeom`.

`stype`**:** Object of class `character`. The type of sampling; namely, `fs` or `is`.

`ctype`**:** Object of class `character`. The type of calculation; namely, `exact` or `approximate`.

`k`**:** Object of class `numeric`. The number of categories in the population.

`thetastar`**:** Object of class `numeric`.

For `dtype="multinomial"` this is a vector whose components represent the smallest ratio between the probabilities corresponding to the categories with the largest and second largest probabilities.

For `dtype="hypergeom"` this is a vector whose components represent the smallest ratio between the category sizes corresponding to the categories with largest and second largest sizes.

`n`**:** Object of class `numeric`. The sample size.

`m`**:** Object of class `numeric`. Only for stype `"is"`, a number giving the cell quota

`N`**:** Object of class `numeric`. Only for for class `"LF.hypergeom"`, an integer giving the population size from which the sample is drawn.

`M`**:** Object of class `matrix`. Only for for class `"LF.hypergeom"`, a matrix giving the population category sizes that minimize the probability of correct selection, i.e. the least favorable configuration

`EWT`**:** Object of class `matrix`. A numeric vector giving the expected sample size.

`pcs`**:** Object of class `numeric`. A numeric vector giving the probabilities of correct selection.

Methods

plot `signature(x="LF.multinomial", y="missing")`, `signature(x="LF.hypergeom", y="missing")` Plot the curve.

show `signature("LF2c")`: Show the basic sampling information.

summary `signature("LF2c")`: Summarise the sampling information and show all the details.

See Also

`LF2c`

Bibliography

[1] Alam, K. (1971). On selecting the most probable category. *Technometrics*, **13**, 843-850.

[2] Alam, K., Seo, K. and Thompson, J. R. (1971). A sequential sampling rule for selecting the most probable multinomial event. *Annals of the Institute of Statistical Mathematics*, **23**, 365-374.

[3] Bechhofer, R. E., Elmaghraby, S. and Morse, N. (1959). A single-sample multiple-decision procedure for selecting the multinomial event which has the highest probability. *The Annals of Mathematical Statistics*, **30**, 102-119.

[4] Bechhofer, R. E. and Kulkarni, R. V. (1984). Closed sequential procedures for selecting the multinomial events which have the highest probabilities. *Communications in Statistics - Theory and Methods A*, **13**(24), 2997-3031.

[5] Bechhofer, R. E., Santner, T. J. and Goldsman, D. M. (1995). *Design and Analysis of Experiments for Statistical Selection, Screening and Multiple Comparisons*. John Wiley and Sons, New York.

[6] Cacoullos, T. and Sobel, M. (1966). An inverse-sampling procedure for selecting the most probable event in a multinomial distribution. In PR Krishnaiah (ed.), *Multivariate Analysis*, pp.423-455. Academic Press, New York.

[7] Chambers, J. M. (1999). *Programming with Data*. Springer.

[8] Childs, A. (2010). Vector extensions of the Dirichlet HC and HD functions, with applications to the sharing problem. *Methodology and Computing in Applied Probability*, **12**(1), 91-109.

[9] Gupta, S. S. and Huang, D. Y. (1975). On subset selection procedures for Poisson populations and some applications to multinomial selection problems. In RP Gupta (ed.), *Applied Statistics*, pp. 97-109. North-Holland, Amsterdam.

[10] Gupta, S. S. and Nagel, K. (1967). On a subset selection procedure for the most probable event in a multinomial distribution. *Sankhya Series B*, **29**, 1-17.

[11] Gupta, S. S. and Panchapakesan, S. (1979). *Multiple Decision Procedures: Theory and Methodology of Selecting and Ranking Populations*. Wiley, New York.

[12] Janardan, K. G. and Patil, G. P. (1972). A unified approach for a class of multivariate hypergeometric models. *The Indian Journal of Statistics A*, **34**(4), 363-376.

[13] Johnson, N. L., Kotz, S. and Balakrishnan, N. (1997). *Discrete Multivariate Distributions*. Wiley.

[14] Kesten, H. and Morse, N. (1959). A property of the multinomial distribution. *The Annals of Mathematical Statistics*, **30**, 120-127.

[15] Le Gall, F. (2006). The modes of a negative multinomial distribution. *Statistics and Probability Letters*, **76**(6), 619-624.

[16] Panchapakesan, S. (1971). On selection and ranking procedures and order statistics from the multinomial distribution. In SS Gupta, J Yackel (eds.), *Statistical Decision Theory and Related Topics*, pp. 275-298. Academic Press, New York.

[17] Ramey, T. and Alam, K. (1979). A sequential procedure for selecting the most probable multinomial event. *Biometrika*, **66**, 171-173.

[18] Ramey, T. and Alam, K. (1980). A Bayes sequential procedure for selecting the most probable multinomial event. *Communications in Statistics - Theory and Methods A*, **9**(3), 265-276.

[19] Santner, T. J. and Duffy, D. E. (1989). *The Statistical Analysis of Discrete Data*. New York: Springer Verlag.

[20] Sobel, M., Uppuluri, V. R. R., and Frankowski, K. (1977). Dirichlet distributions type-1. *Selected Tables in Mathematical Statistics* **4**, American Mathematical Society, Providence, Rhode Island.

[21] Sobel, M., Uppuluri, V. R. R., and Frankowski, K. (1985). Dirichlet Integrals Type 2 and their application. *Selected Tables in Mathematical Statistics* **9**, American Mathematical Society, Providence, Rhode Island.