Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science and Statistics

Clustering Multivariate Categorical Data Using Exact ICL Method

Manasi Mohan Narsapur
[redacted]

A Dissertation submitted to the University of Dublin,
in partial fulfilment of the requirements for the degree of
Master of Science in Computer Science (Data Science)

Supervisor : Dr. Arthur White

2020

Declaration

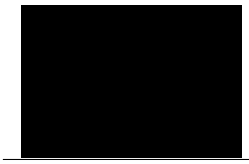
I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Manasi Mohan Narsapur

September 5, 2020

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.



Manasi Mohan Narsapur

September 5, 2020

Acknowledgments

I would like to express my gratitude towards the following individuals who have supported and guided me throughout this dissertation.

First and foremost, I would like to thank my supervisor, Dr. Arthur White for his mentorship, insight and knowledge that has steered me throughout this research. I would like to specially acknowledge his endless support and the latitude he has provided me during the programming stage of research.

Furthermore, I would like to express heartfelt appreciation towards my friends, Aishwarya and Mukesh for their advice, opinions, valuable feedbacks and for proofreading the dissertation drafts. Lastly, my deep and sincere gratitude towards my family for their continuous support and selflessly encouragement to explore new directions in life. This milestone is dedicated to them.

Thank you.

Manasi Mohan Narsapur

University of Dublin, Trinity College
September 2020

Clustering Multivariate Categorical Data Using Exact ICL Method

Manasi Mohan Narsapur, Master of Science in Computer Science

University of Dublin, Trinity College, 2020

Supervisor: Dr. Arthur White

Clustering, an integral part of data analysis, is a process of grouping together observations based on characteristics, which aids discovery of hidden information from a dataset. The latent class analysis (LCA) modelling algorithm is administered with the exact integrated complete likelihood (ICL) method, to build an algorithm named *exactICLforLCA* in this dissertation with the intent to cluster the multivariate categorical data utilizing the exact ICL method. Concepts such as Bayesian inference, beta distribution, Dirichlet distribution were applied to produce the notation obtained by administering ICL method on LCA modelling algorithm which was used to gauge the improvement in clustering. The algorithm was fit by utilizing real world data such as Alzheimer's dataset and simulated datasets. The results illustrated that the algorithm successfully assists in the improvement of clustering for a dataset by identification of the optimum number of groups, affirmed by the increase in the ICL value. The new matrices generated by the *exactICLforLCA* algorithm displayed an improvement in clustering and the average increase in the ICL value for simulated datasets was approx. 140.77 and 65.307 values for real data when compared to the fit by expectation–maximization algorithm for an experiment run for 20 iterations for both the datasets. Data clustered with the help of expectation–maximization algorithm were compared against the fit by the *exactICLforLCA* algorithm with the help of a cluster evaluation technique called `randIndex` to calculate the similarity among them.

Keywords— integrated complete likelihood, latent class analysis, Bayesian inference, cluster analysis

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

Clustering is a process of splitting the population or observations of a dataset into a number of subsets or groups, which correspondingly leads to observations or data points in the same group being similar to other data points in the same group and distinct from the observations or data points in other groups. Clustering is an important step in data analysis, which aims to extract information from a data set and transform it into an understandable format for later use. Cluster analysis is used to gain valuable insights from the data by observing the data points that form a group or a cluster. As an analytical activity listed out by data users, clustering is defined as finding groups of similar attribute or characteristic values from the given set of data cases.

Cluster analysis is a form of exploratory data analysis, where in, based on common characteristics, observations are divided into different groups. It is a type of multivariate data analysis technique, which is used to analyse data containing more than two variables, known as multivariate data. Some types of cluster analysis are conceptual clustering, spectral clustering, hierarchical clustering and model based clustering. Clustering techniques are methods used to identify groups of similar observations and some of these techniques utilised for clustering are k-nearest neighbor search, neighbourhood components analysis and latent class analysis.

Clustering approach can be categorised into two types, namely *deterministic* and *stochastic*. The deterministic clustering is a technique where the adjacency matrix or network is obtained by minimizing an objective function which measures discrepancy from an ideal structure of the model. The stochastic clustering approach also known as model based approach, assumes that the parameterized distribution help model the probability of links between the node sets in the network. The estimated parameters are utilized as a personification of the true network linking behaviour.

Latent class analysis (LCA) is a finite mixture model which offers a model based or stochastic clustering approach that generates clusters using a probabilistic model that describes the distribution of the data. The model outlines the distribution of the data and based on this model, the probabilities that certain data points or observations belong to a particular latent class or cluster is determined. The LCA model utilises this approach where the distribution such as number of clusters is described, unlike cluster analysis such as k-nearest neighbor search where the option is not available and starts off with examining the similarity between the observations followed by clustering them. Due to this structuring, the LCA model is used as the statistical model which helps to model the latent structure that underlies the structure of the data rather than just looking for similarities.

This dissertation aims to generates an algorithm to increase the quantity of clusters with the help of the integrated complete likelihood (ICL) value. The ICL value is obtained from the notation generated upon applying the Exact ICL method on the LCA model. Increase in value indicated the improvement in clustering. The research examines the model's application on different types of categorical datasets such as real world data and simulated datasets. Further analysis are run on the obtained results by utilizing a cluster evaluation technique such as rand index in the case of simulated dataset, to check the similarity between the datasets and retrieve the true structure of the matrix.

The rest of the paper is organised as follows, Chapter 2 puts forth the outline of the model specification for LCA in detail and provides a brief overview of other fundamental concepts such as multivariate categorical data, Bayesian inference and cluster of the LCA model by Exact ICL method. Chapter 3 discusses the implementation and

creation of the algorithm succeeded by the Chapter 4 which presents the application of the generated algorithm on to real and simulated data. Chapter 5 summarizes the dissertation, commencing with Chapter 6, which concludes the research and mentions the future work.

1.2 Background

The principle idea of utilizing the exact ICL method was taken from 'Estimation and prediction for stochastic blockstructures' authored by Nowicki, Krzysztof, and Tom A. B. Snijders' where in, the method was used to predict and estimate the stochastic block models(Côme and Latouche, 2015). The paper also stated that the largest value of the exact ICL, assuming it can be computed, provides the most preferable clustering of the nodes or observations in a dataset into subsets. The authors utilized the idea to optimize the ICL criterion using a greedy search over labels and the number of node clusters for application on the stochastic block models(SBM).

Defining the ICL with instance of numbers of clusters and labels was suggested by Rohe et al., 2016 (Rohe et al., 2016). The optimization of exact ICL through iterative cycles following smaller optimization techniques repeatedly until no further improvement in the ICL value is obtained was gathered from Wyse et al., 2017 (WYSE et al., 2017). This paper also provided inputs regarding the random assignment of nodes to the subsets and the convergence of model to a local maximum, which might not necessarily be the global maximum. Hence, the paper suggested running the greedy algorithm multiple number of times which would provide the highest exact ICL value.

1.3 Research Overview

The objective of the research is to cluster the multivariate categorical data using the exact integrated complete likelihood (ICL) approach. The research aims to apply the ICL method on the latent class analysis modelling algorithm in an attempt to obtain better clustering from datasets. In order to achieve this goal, the research involves formulation of an algorithm which aids in the enhancement of the ICL value, which indicates improvement quality in clustering.

In support of achieving the above mentioned aim, the Figure 1.1 illustrates in detail the thread of actions that were performed in the dissertation. Four main objectives that were undertaken to generate the algorithm to attain improved clustering are as follows:

- Understanding the Latent Class Analysis model.
- Derivation of Exact ICL method for the LCA model.
- Implementation of the model.
- Application of the algorithm to real and simulated data.

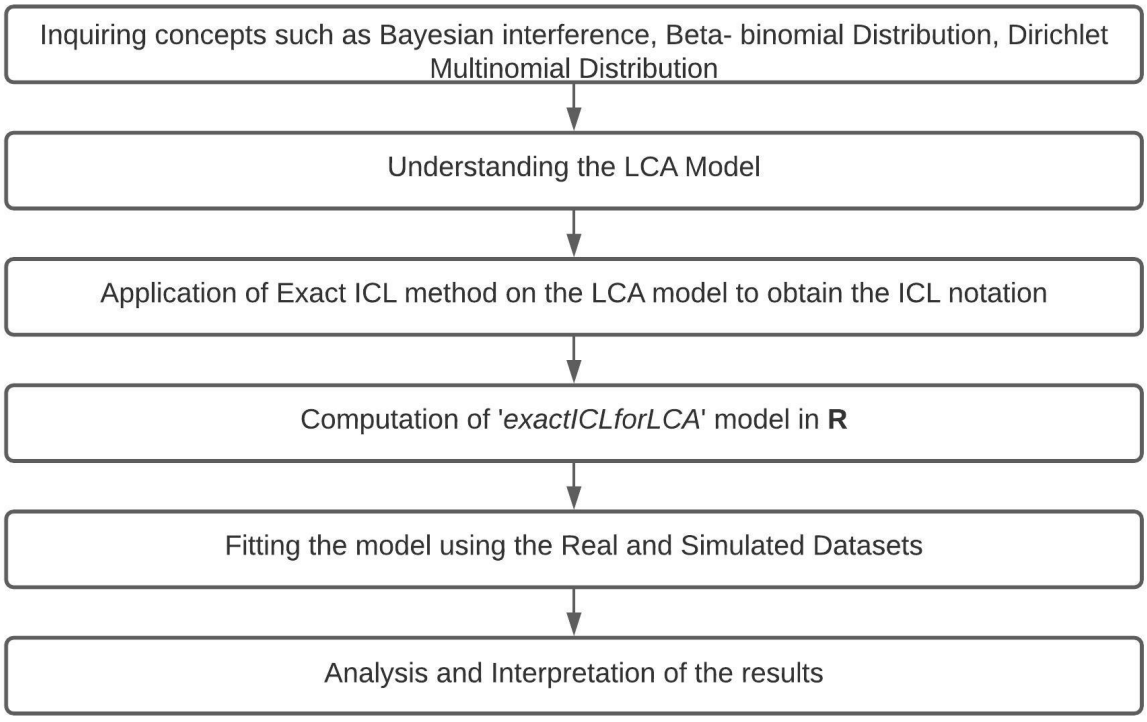


Figure 1.1: Different Phases of Implementation

1.4 Dataset Description

The section provides description regarding the datasets that have been used in this dissertation. The datasets are primarily fit using the BayesLCA package (White and Murphy, 2014) before being subjected to the model where the binary data is analysed by latent class analysis (LCA) model in an attempt to find the hidden clusters. The `blca.em` function applies the expectation-maximisation algorithm to find maximum a posteriori (MAP) estimates of the parameters and provides the prior values specific to the model, post which the model is subjected to the algorithm created for this research.

1.4.1 Alzheimer's Dataset

The Alzheimer's dataset (Moran et al., 2004) contains data regarding patients diagnosed with early onset of Alzheimer's disease, conducted in the Mercer Institute in St. James's Hospital, Dublin. The data documents the presence or absence of 6 symptoms of Alzheimer's disease in 240 patients. The patients were examined for Hallucination, Activity, Aggression, Agitation, Diurnal and Affective symptoms. The binary data consisting of 240 rows and 6 columns, marked with 1 to denote the presence of the symptom and 0 for the absence of it. Each row displays a patient and each column denotes the presence or absence of one of the 6 listed symptoms.

1.4.2 Simulated Dataset

Across industries and disciplines, simulation modeling provides valuable solutions by giving clear insights into complex systems. The simulated datasets is used in the dissertation as a step to validate the model by recovering the true structure of the data after it has been fit by the model that has been generated using the proposed algorithm called *exactICLforLCA*. The simulated set contains fictionalized data generated randomly with the help of `rlca` function (White and Murphy, 201) which generates binary dataset with respect to underlying latent classes. The `rlca` function has been slightly modified to return the labels along with the randomly generated matrix; and takes inputs such as the number of data points to be generated, combined vectors and the argument share among the groups as inputs to generate the simulated data from

the user. An example of the simulated dataset model for number of observations, N =1000 with 4 variables or features and parameters values, θ as 0.7 and 0.3 has been provided as shown below.

```
N <- 1000
type1 <- c(0.2, 0.8, 0.8, 0.2)
type2 <- c(0.8, 0.2, 0.2, 0.8)
sim_data_list <- rlca_fun(N, rbind(type1,type2), c(0.7,0.3))
```

Chapter 2

Exact ICL on LCA

The chapter illustrates the application of exact ICL method on the latent class analysis modelling algorithm. At the outset, concepts such as the multivariate categorical data, latent class analysis and Bayesian inference have been described which are utilized to generate the ICL notation when LCA is enforced with ICL method at the end of the chapter.

2.1 Multivariate Categorical Data

Multivariate categorical data is data involving three or more variables where they represent types of data which may be divided into groups, with no intrinsic ordering. A pattern of multivariate categorical type of data can be observed in the hospital records maintaining patient's symptoms, where in, each patient is marked for the existence of or for displaying the symptoms on daily basis which can be used to analyse the onset of the disease and predict the next stage of the disease for the patient. All problems, including the ones in multidimensional data can be solved by considering all the involved categorical data variables and analysing and testing them. To solve these issues involving three or more variables which are inherently multidimensional, requires the use of multivariate data analysis, which analyse and find patterns in multivariate data variables.

Multivariate data analysis refers to all statistical methods that simultaneously analyze multiple evaluations on each individual observation. Thus, any simultaneous analysis of more than two variables can be considered multivariate analysis. Multivariate data analysis techniques can be categorised as two, each pursuing a different type of relationship to the data, *dependence* and *interdependence*. Dependence relates to cause-effect situations and tries to see if a set of variables can describe or predict the values of other variable sets. Interdependence refers to structural intercorrelation and aims to understand the underlying patterns of the data. Cluster analysis, a type of interdependence technique, aims at detecting groups or clusters of data entities that have similar values. Clusters can be analysed to understand the distribution of entities, cause for the similarity and knowledge about the behaviour that drives the values of the analysed objects and find similar data points. Latent class analysis (LCA), a subset of structural equation modelling and a type of cluster analysis technique, is used to identify groups or clusters in multivariate categorical data. These subtypes are called *latent classes*.

2.2 LCA Description

Latent class analysis (LCA) model is a type of model-based clustering which aims to distinguish unobservable homogeneous sub groups or uncover hidden groupings in data. Specifically, it is a method to group subjects from multivariate data into latent classes. Latent class analysis also known as finite mixture modelling or mixture modelling based clustering, is used to develop classification systems which help group individuals or entities based on prevailing set of characteristics. Some examples of this type of data can include the symptoms displayed by subjects with major depressive disorder (Garrett and Zeger, 2000), the answers submitted during an exam which can be correct or incorrect (Bartholomew and Knott, 1999), or a disability index recorded by a long-term survey (Erosheva et al., 2007).

The Latent class analysis (LCA) modelling algorithm that describes the relationship between a set of observed variables and the latent categorical variable. LCA is a special case of modelling using categorical latent variables, generally known as finite modelling where latent variables represent a set of sub populations and the population membership is not known but is inferred from the data. LCA can be described using

latent models which primarily helps identify groups of individuals or entities they have some characteristics in common. Usually the array of observed data is too complex to identify groups through only inspection therefore multivariate classification procedures such as LCA is employed for this purpose. LCA hypothesizes and estimates a latent variable model and uncovers hidden patterns of association that can exist between observations. Conditional probability patterns, indicating the chance variables take on certain values and create the basis for latent class formation.

Latent class analysis involves performing inference with a mixture model frame-work and consists of latent classes which are the observed variables that are derived from the unobserved variables. Latent classes divide the cases or observations into their respective dimensions in relation to the variable. The clusters created by the cluster analysis are called as latent classes, these classes help obtain the latent variable or hidden variable or a construct which is a variable that is not observable or directly measurable. The observed variables in the data act as indicators to measure the latent variables.

The Latent class analysis relies on certain assumptions such as:

- The homogeneous sub-populations exist within the data. These subgroups have distant probability distributions and are mutually exclusive and as these sub-populations do not overlap, all classes together account to the total population.
- LCA assumes that the number of latent classes specified by latent model is correct.
- Conditional independence or local independence, LCA assumes that all relationships among the observed variables are accounted for by the latent class membership.

Types of Latent Class Analysis:

- Latent class cluster analysis: Latent class cluster analysis is based on the probability of classifying the class unlike cluster analysis methods which were based on the nearest distance such as nearest neighbors algorithm. It identifies clusters

that group people together, based on similar behaviours, characteristics, interests, or values. K-category latent variables represent the clusters and the number and size of the classes are not known beforehand.

- Latent class factor analysis identifies factors that group together variables with a common source of variation. The analysis is based on the class, each class shows one factor unlike traditional factor analysis method which was based on the rotated factor matrix.

- Latent class regression analysis predicts a dependent variable as a function of predictors. Single set of items is used to establish class memberships, and then additional covariates are used to model the variation in class memberships.

2.3 Bayesian Inference

Bayesian inference refers to the application of Bayes' theorem in determining the updated probability of a hypothesis given new information. It allows the posterior probability, which is the updated probability, to be calculated given the prior probability of a hypothesis and the likelihood function. It is a statistical inference in which the probability for a hypothesis or the posterior probability is updated as more information or evidence becomes available using the Bayes theorem. The theorem is based on the posterior distribution, given that the data is obtained from the two antecedents prior density and the likelihood function which is derived from a statistical model for the observed data using the following expression,

$$Posterior \quad \propto \quad Likelihood * Prior$$

Bayesian inference is a way to get sharper predictions from the data and is particularly useful when data is not available in abundance and information needs to be extracted from it. In simple terms, Bayesian inference technique specifies how one should update one's beliefs upon observing data. Bayesian inference computes the posterior probability according to Bayes' theorem and can be written as following,

$$P(H|E) = \frac{P(E|H) * P(H)}{P(E)}$$

- H stands for any hypothesis whose probability may be affected by data.
- *P(H)*, the prior probability, it is the estimate of the probability of the hypothesis H before the information E is observed.
- $P(H|E)$, the posterior probability, it is the probability of H given E, after it is observed.
- $P(E|H)$ is the likelihood and it is the probability of observing E given H.
- *P(E)*, the marginal likelihood or model evidence, it is the same for all possible hypotheses being considered so this factor does not enter into determining the relative probabilities of different hypotheses.

In the Bayesian approach, parameters are treated as random variables which can be described with a probability distribution where probability is simply the degree of belief, unlike that in the frequentist approach where the parameters are treated as fixed but unknown quantities. Bayesian inference is built on four key concepts as listed below,

- Conditional probability: For two events A and B, the conditional probability of A given B is denoted $P(A|B)$, and is defined by the formula

  $$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

  The probability P(A and B) is the joint probability of A and B.

- Bayes' theorem: For two events A and B, provided that

$$\begin{aligned} P(B) &\neq 0 \\ P(A|B) &= \frac{P(B|A) * P(A)}{P(B)} \end{aligned}$$

- Prior: The mathematical expression about the belief regarding the parameters.
- Likelihood: The joint probability density function of sample X = (X1,... X2) is $P(x|Y)$, where Y is a parameter and X = x is an observed sample point. Then the function of Y is the likelihood function which can be defined as

$$L(Y|x) = P(x|Y)$$

One of the primary ideas in Bayesian inference is that knowledge about anything unknown can be expressed in terms of probability. The posterior is influenced by the prior, likelihood and the sample size involved. The posterior value is similar to the prior if an informative prior is used in a relatively small sample size else is similar to the likelihood function when an uninformative prior is used. Prior's influence on the posterior is decided by the function's informative value. Ideally, best posteriors are obtained when the sample size is small and informative priors are used and large sample sizes lead to the posterior being influenced by likelihood.

2.4 LCA Model

Latent class analysis(LCA) model explains the relationship between a set of r number of observed variables also referred to as observed indicators and an underlined latent categorical variable. Observed indicators can be binary ordered or un-ordered categorical, censored count or continuous variables. For data with n number of observations where data y, of the form $y = y_1, y_2..., y_n$ and $y_{ij} \in \{0, 1\}$, a particular observation i can be written as $y_i = y_{i1}, y_{i2}..., y_{ir}$.

Suppose the data is divided into G number of clusters, the mixture model will contain $\pi = \pi_1, \pi_2..., \pi_G$ where $\pi_g = P(Group\ g)$, is the class probability or a prior probability or a weight parameter for the observation $i \in Group\ g$. For this LCA model with r observed variables, n number of observations and G clusters, the marginal item probability for item (i, j) being equal to 1 is provided as below,

$$P(ij = 1) = \sum_{g=1}^{G} P(i \in g) P(ij = 1 \mid i \in g)$$

$$\begin{aligned} P(y_{ij}) &= \sum_{g=1}^{G} P(y_{ij} | i \in g) P(i \in g) \\ &= \sum_{g=1}^{G} P(i \in g) \prod_{j=1}^{r} P(y_{ij} | i \in g) \\ &= \prod_{i=1}^{n} P(i \in g) \prod_{j=1}^{r} P(y_{ij} | i \in g)\ where \prod g = P(i \in g) \end{aligned}$$

$$\theta_{gj} = P(y_{ij} | i \in Group\ g)$$

where y_{ij} is Binomial and θ is the probability parameter

$$P(y_{ij} | Group\ g, \theta_{gj}) = \theta_{gj}^{y_{ij}} (1 - \theta_{gj})^{1 - y_{ij}} \tag{2.1}$$

Making the naive Bayes assumption, as mentioned in Section 2.3, which assumes that two random events A and B are conditionally independent given a third event C, provided the occurrence of A and B are independent events in their conditional probability distribution given the event C. Specifically, A and B are conditionally independent given C if and only if, occurrence of C and A provide no information on the likelihood of B occurring, and knowledge of whether B occurs provides no knowledge on the likelihood of event A occurring. In other words, *Conditional Independence*, which states or assumes that the effect of the value of a predictor on a given class is independent of the values of other predictors can be used to write the model as following,

$$p(y_{ij}|Group\ g, \theta_g) = \prod_{j=1}^{r} \theta_{gj}^{y_{ij}} (1-\theta_{gj})^{1-y_{ij}}$$

The full model $p(y_{ij} \mid \pi, \theta)$, for n number of observations, r number variables or features and G number of clusters, with priors θ and π which represent the probability parameter and the prior probability respectively can be equated as

$$\begin{aligned}
P(y \mid Group\ g) &= p(y_{ij} \mid \theta_{gj}) \\
P(y_{ij}, Group\ g \mid \pi_g, \theta_g) &= P(y_{ij} \mid Group\ g)\ P(Group\ g) \\
P(y_{ij} \mid \pi, \theta) &= \sum_{g=1}^{G} p(y_{ij}, Group\ g \mid \pi, \theta) \\
&= \sum_{g=1}^{G} \pi_g \prod_{j=1}^{r} \theta_{gj}^{y_{ij}} (1-\theta_{gj})^{1-y_{ij}} \\
p(y_{ij} \mid \pi, \theta) &= \prod_{i=1}^{n} \sum_{g=1}^{G} \pi_g \prod_{j=1}^{r} \theta_{gj}^{y_{ij}} (1-\theta_{gj})^{1-y_{ij}}
\end{aligned}$$

The full model $p(y \mid \pi, \theta)$ is the observed data likelihood where the likelihood function is constructed by considering all possible values of the latent variables, and the associated probabilities for the observed data. Introducing a missing data or latent variable z, where $z_i = z_{i1}, z_{i2}..., z_{iG}$ and

$$z_{ig} = \begin{cases} 1, & \text{if } i \in Group\ g \\ 0, & \text{otherwise} \end{cases}$$

The complete data likelihood can be written as following with θ and π as priors, which can be used to construct the likelihood function when the values of the latent random variables are known.

$$p(y, z \mid \theta, \pi) = \prod_{i=1}^{n} \prod_{g=1}^{G} \{\pi_g \prod_{j=1}^{r} \theta_{gj}^{y_{ij}} (1-\theta_{gj})^{1-y_{ij}}\}^{z_{ig}}$$

Setting α and β as hyperparameter for the θ prior using Bayesian inference, where the beta distribution is applied to model the behavior of random variables limited to intervals of finite length, the full model with respect to θ is formed.The beta distribution is a probability distribution on probabilities and is a family of continuous probability distributions, which is a mathematical function that gives the probabilities of occurrence of different possible outcomes for an experiment carried out and is defined on the interval [0,1]. The distribution is parametrized by two positive shape parameters, known as α and β, that act as exponents of the random variable and control the shape of the distribution. The general form of beta distribution for a random variable X distributed with respect to parameters α and β is denoted as below

$$X \sim \mathcal{B}(\alpha, \beta) \; and \; X \in [0, 1]$$

$$f(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\mathcal{B}(\alpha, \beta)}$$

$$where \; \mathcal{B}(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)} \; and \; \Gamma \; is \; Gamma$$

The full model using the beta distribution formed with respect to θ with α and β hyperparameters is written as shown below.

$$\begin{aligned} p(y, z, \theta, \pi) &= p(y, z|\theta, \pi) \; p(\theta) \; p(\pi) \\ p(\theta \mid \alpha, \beta) &= \prod_{g=1}^{G} \prod_{j=1}^{r} p(\theta_{gj}|\alpha, \beta) \\ p(\theta \mid \alpha, \beta) &= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \; \theta_{gj}^{\alpha-1} \; (1-\theta_{gj}^{\beta-1}) \end{aligned}$$

Therefore,

$$p(\theta \mid \alpha, \beta) \;=\; \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \prod_{g=1}^{G} \prod_{j=1}^{r} \theta_{gj}^{\alpha-1} \; (1-\theta_{gj}^{\beta-1})$$

$$p(y, z, \theta, \pi) = \mathcal{B}(\alpha, \beta) \prod_{g=1}^{G} \prod_{j=1}^{r} \theta_{gj}^{\alpha-1} \; (1-\theta_{gj}^{\beta-1}) p(y|\pi, z) \; p(\pi) \tag{2.2}$$

The Dirichlet distribution is used to generate the full model equation with respect to the π prior with δ as the hyperparameter. The Dirichlet distribution $\mathbf{Dir}(\delta)$ or $\boldsymbol{D(\delta)}$ is a family of continuous multivariate probability distributions parameterized by a vector δ of positive reals. The Dirichlet distribution is the conjugate prior to a number of probability distributions such as the categorical distribution and the multinomial distribution. Dirichlet distributions are probability distributions over multinomial parameter vectors and are called beta distributions when the number of outcomes is fixed to 2.

The multinomial distribution has two important properties,

- The sum of the probabilities for each entry must be equal to one.
- The probabilities should not be negative.

That is, $x_1, x_2..., x_K \; where \; x \in (0,1) \; or \; x_i \geq 0 \; and \; \sum_{j=1}^{K} x_j = 1$

Dirichlet distribution is a multivariate generalisation of the beta distribution which is commonly used as prior distributions in Bayesian statistics and is formally expressed as

$$X \sim \boldsymbol{D}(\delta)$$

$$f(x) = \frac{1}{\mathcal{B}(\delta)} \prod_{i=1}^{G} x_i^{\delta_i - 1}$$

$$where\ \mathcal{B}(\delta) = \frac{\prod_{j=1}^{G}\Gamma(\delta_j)}{\Gamma(\sum_{j=1}^{G}\delta_j)}\ ,\ \delta = (\delta_1, \delta_2..., \delta_G)\ and\ \Gamma\ is\ Gamma$$

$$\begin{aligned} p(y, z, \theta, \pi) &= p(y, z|\theta, \pi)\ p(\theta)\ p(\pi) \\ &= \prod_{g=1}^{G}\prod_{j=1}^{r}\ \pi_g^{z_{iG}}\ p(y|\theta, z)\ p(\theta)\ p(\pi) \\ &= \prod_{g=1}^{G}\prod_{j=1}^{r}\ \pi_g^{z_{iG}}\ \frac{1}{\mathcal{B}(\delta)}\ \prod_{g=1}^{G}\ \pi_g^{\delta_g - 1}\ p(y|\theta, z)\ p(\theta) \\ &= \prod_{g=1}^{G}\ \pi_g^{\sum_{i=1}^{n} z_{iG}}\ \frac{1}{\mathcal{B}(\delta)}\ \pi_g^{\delta_g - 1}\ p(y|\theta, z)\ p(\theta) \end{aligned}$$

$$p(y, z, \theta, \pi) = \frac{1}{\mathcal{B}(\delta)}\ \prod_{g=1}^{G}\ \pi_g^{ng+\delta_g - 1}\ p(y|\theta, z)\ p(\theta) \tag{2.3}$$

The equation 2.3 contains $\pi_g^{ng+\delta_g-1}$ term which is of the same form as the kernal of the Dirichlet distribution.

Combining the equations 2.2 and equation 2.3, the Latent class analysis(LCA) model for data y, containing n observations, r variables, divided into G clusters, with α, β as hyperparameters for θ prior and δ as hyperparameter for π priors is obtained.

$$p(y, z, \theta, \pi) = \frac{\mathcal{B}(\alpha, \beta)}{\mathcal{B}(\delta)}\ \prod_{g=1}^{G}\prod_{j=1}^{r}\ (\ \theta_{gj}^{\alpha-1}\ (1 - \theta_{gj}^{\beta-1}\ ))\ (\pi_g^{ng+\delta_g-1}) \tag{2.4}$$

The LCA model can be represented using the graphical diagram as shown in the Figure 2.1. The graphical diagram consists of nodes and edges, where nodes represent the data, parameters and the hyperparameters, while edges are used to depict the dependence between the nodes. The nature of dependency between the nodes are illustrated by the edges in the form of direction. The figure describes the hyperparameter nodes α, β and δ represented in rectangular cells, which are set or provided as an input by the analyst.

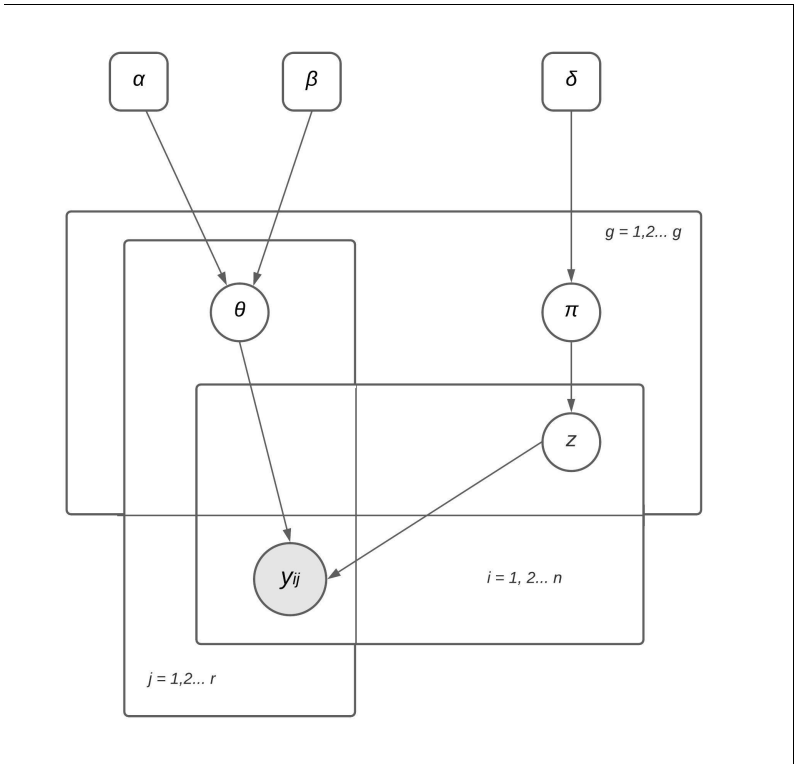


Figure 2.1: Graphical Diagram of LCA Model

The transparent cells such as priors θ and π and latent variable matrix z denote the values that are to be estimated, while the shaded cell y_{ij} represents the observed data. Plate notation is utilised to provide concise range of the variable, observations and cluster values. The diagram depicts the prior's dependency on the hyperparameters, that is, the reliance of θ on α and β and π on δ.

2.5 Exact ICL for LCA

Integrated complete-data likelihood(ICL) (Biernacki et al., 2000) (Biernacki et al., 2010) is a model selection criterion for clustering (Marbac and Sedki, 2017), for the Gaussian or normal mixture models. ICL is used because in the finite mixture model context the analysis is often carried out using a latent label vector which is difficult to integrate from the model. The introduced latent vector is often termed the allocation vector(Nobile and Fearnside, 2007) and provides clustering of the data to component

densities. Biernacki et al.,2000 research supports the argument that the evidence for the clustering should be considered while determining the number of mixture components that are required and hence suggest the importance of integrated completed data likelihood.

The ICL methods marginalizes the labels from the model and includes them as an integral of the information criterion. The number of clusters in the mixture, that is, the G value which provides the highest ICL value is most supported by the dataset. Using the notation introduced in the Equation 2.4 the integrated complete-data likelihood(ICL) for the full model can be calculated by solving the equation with respect to π and θ and can be written as

$$ICL \;\; = \;\; \int_{\pi}\int_{\theta} p(y, z, \theta, \pi) d\theta d\pi$$

Integrating the LCA model equation with respect to π, the following notation is obtained

$$\int p(y, z, \theta, \pi) \; \mathbf{d}\pi = \frac{1}{\mathcal{B}(\delta)} \int \prod_{g=1}^{G} \pi_g^{ng+\delta_g-1} \; C \; \mathbf{d}\pi$$

where C is the constant.

With the prior knowledge that the intergration of Dirichlet distribution is always equal to 1, deduction in the terms of $\mathcal{B}(\delta)$ is carried out.

Equating

$$\int \prod_{g=1}^{G} \pi_g^{ng+\delta_g-1} \; C \; \mathbf{d}\pi = \mathcal{B}(\delta')$$

Implying that

$$\int p(y, z, \theta, \pi) \; \mathbf{d}\pi = \frac{\mathcal{B}(\delta')}{\mathcal{B}(\delta)}$$

Similarly integrating with the full LCA model equation with respect to θ, the notation can be written as,

$$\int p(y, z, \theta, \pi)\ \mathbf{d}\theta = \frac{\prod_{g=1}^{G} \prod_{j=1}^{r}\ \mathcal{B}(\alpha_{gj}, \beta_{gj})}{\mathcal{B}(\alpha, \beta)^{Gr}}$$

where, $\alpha_{gj} = \sum_{i=1}^{n} y_{ij}\ z_{ig} + \alpha$ and $\beta_{gj} = \sum_{i=1}^{n}(1 - y_{ij})\ z_{ig} + \beta$

The ICL notation for LCA model integrated with respect to θ and π priors is written as

$$ICL \ = \ \int_{\pi}\int_{\theta} p(y, z, \theta, \pi) d\theta d\pi$$

$$ICL \ = \ \frac{\mathcal{B}(\delta')}{\mathcal{B}(\delta)} \times \frac{\prod_{g=1}^{G} \prod_{j=1}^{r}\ \mathcal{B}(\alpha_{gj}, \beta_{gj})}{\mathcal{B}(\alpha, \beta)^{Gr}} \tag{2.5}$$

Where $\delta' = (\delta'_1, \delta'_2, ...\delta'_G)$, and for $g = 1, 2, ..., G$ and $j = 1, 2, ..., r$
The hyperparameters α_{gj}, β_{gj} and δ' for the equation are as follows,

$$\alpha_{gj} = \alpha + \sum_{i=1}^{n} z_{ig} y_{ij} \tag{2.6}$$

$$\beta_{gj} = \beta + \sum_{i=1}^{n} z_{ig}(1 - y_{ij}) \tag{2.7}$$

$$\delta'_g = \delta_g + \sum_{i=1}^{n}\ z_{ig} \tag{2.8}$$

The ICL notation iteratively generated the δ'_g and α_{gj}, β_{gj} values which are dependent on z and z , y respectively, which helps generate a new matrix z. The z matrix is updated repeatedly to optimize the ICL value, that is, until the highest ICL value is attained.

Chapter 3

Algorithm Implementation

The chapter describes the architecture and code implementation of Exact ICL method applied on the latent class analysis (LCA) modelling algorithm. The workflow diagram illustrating the build and stages of model is elucidated along with the details in each of these stages and how it was achieved through programming . Lastly, the criteria for the termination of model will be explained.

For the purpose of this dissertation, a package named ***exactICLforLCA*** has been created in `R` language as a part of the implementation. The data to be fit along with the initial number of clusters, alpha, beta and delta values are provided as an input to the package for the fit to commence. As a primary step for the implementation the inputted dataset is first fit using the `blca.em` function which utilises an expectation-maximisation algorithm to find maximum a posteriori (MAP) estimates of the parameters. The generated matrix, post `blca.em` fit function [$\boldsymbol{z}$], the matrix form of the dataset provided as an input [$\boldsymbol{y}$], the cluster value [$\boldsymbol{G}$], hyperparameters values, alpha [*alpha_var*], beta [*beta_var*] and delta [*delta_var*] are all passed as parameters to the ICL_Fit to calculate the ICL value for the dataset.

The package *exactICLforLCA* has been created using two built-in libraries, *devtools* and *roxygen2* in `RStudio` (the steps to create a `R` package have been provided in appendix .1). The package is a set of accumulated functions stored under a directory called library in the `R` environment that helps in reusing the application of exact ICL method on the LCA model.

The *BayseLCA*(White and Murphy, 2014) package has been used to generate the initial fit using the `blca.em` function (Bayesian Latent Class Analysis Via An EM Algorithm). The latent class analysis (LCA) attempts to find G number of hidden clusters or groups in binary data and the function acquires the maximum a posteriori (MAP) estimates of the parameters by applying the expectation-maximisation algorithm. The expectation–maximization or EM algorithm, is an iterative method or approach for maximum likelihood estimation in the presence of latent variables(Bishop, 2006). The EM algorithm is an iterative approach that cycles between two modes. The first mode attempts to estimate the missing or latent variables, called the estimation-step or E-step, which creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters, and the second mode attempts to optimize the parameters of the model to best explain the data, called the maximization-step or M-step, which computes parameters maximizing the expected log-likelihood found on the E step. The obtained parameter-estimates in this M step are then used to determine the distribution of the latent variables in the next E step.

1. E-Step: Estimate the missing variables in the dataset.

2. M-Step: Maximize the parameters of the model in the presence of the data.

The matrix to be fit by the *exactICLforLCA* algorithm is generated by finding the Z-score value of the dataset fit by EM method through the `zscore` function. Z-score is a numerical measurement describing a value's relationship to the mean of a group of values, which is measured in terms of standard deviations from the mean. If the the data point's score is identical to the mean score, the Z-score value is 0. The Z-score equivalents obtained are then transformed using the `MAP` functions, by applying the function to each element and returning a list of the same length as the input which is then converted into a matrix of indicator variables using the `unMAP` function.

3.1 Primitive ICL Calculation

The basic ICL calculation contains the implementation of Equation 2.5. The equation contains 4 variables, which are individually calculated and then put together to generate the ICL value. The latent variable matrix z and matrix conversion of the dataset y along with parameters such as the initial cluster value G, α, β and δ values which are set by the analyser are taken as inputs to the function ICL_Calc to generate the primary ICL value of the dataset.

At the outset of the function, the α, β and δ values are utilized to generate the hyper-parameters required for the calculation of the ICL value which are α_{gj}, β_{gj} and δ'_g as denoted in Equations 2.6, 2.7 and 2.8 respectively. The α_{gj} and β_{gj} matrix of $G \times r$ are generated iteratively for all the values in G and r, where both z and y matrices are used to obtain the values. The δ'_g matrix values are generated using the initialised δ value and latent variable matrix.

The Equation 2.5 is broken down into two variables where the first variable is $\left[\frac{\mathcal{B}(\delta')}{\mathcal{B}(\delta)}\right]$ and the second variable is $\left[\frac{\prod_{g=1}^{G}\prod_{j=1}^{r}\mathcal{B}(\alpha_{gj},\beta_{gj})}{\mathcal{B}(\alpha,\beta)^{Gr}}\right]$.

The equations are generated using the log scale, by utilising the inbuilt `lgamma` and `lbeta` functions available in `R`. The `lgamma` function, log-gamma distribution provides the density, distribution function and gradient of density for the log-gamma distribution. The `lbeta` function, (log) beta approximations computes the log of the `beta` function, that is, `log(beta(a,b))`, in a straightforward or an asymptotic way. The beta function provides the density, distribution function, quantile function and random generation for the beta distribution with shape parameters, like α and β.

The components of the first variable are generated by creating a function which implements the following equation.

$$\mathcal{B}(\delta) = \frac{\prod_{j=1}^{G}\Gamma(\delta_j)}{\Gamma(\sum_{j=1}^{G}\delta_j)}$$

The numerator and denominator variable of this equation are computed programmatically using the log scale and with the use of `lgamma` and `sum` function. The `sum` of the `lgamma` function for the passed parameter is generated for the numerator variable and the `lgamma` function of the `sum` of the passed variable is computed for the denomination of the equation. The difference of the numerator and denominator values is taken, as the equation is computed in the log scale to obtain the value of each component of the first variable. The $\mathcal{B}(\delta')$ and $\mathcal{B}(\delta)$ values are obtained by computing the programmatically explained formula for each parameter, that is, δ' and δ. The first variable value is obtained by the difference between the computed values for δ' and δ parameter.

The second variable is generated with the use of `lbeta` function. The denominator is created by calculating the `lbeta` for the α_{gj} and β_{gj} shape parameters and multiplied with the cluster value and the variable value. The numerator utilizes a temp variable which is increased by the value of `lbeta` for the α_{gj} and β_{gj} for all variables in each group. The numerator and denominator are subtracted to obtain the second variable value. The ICL value is calculated by adding both the first and second variable as the log of the function converts multiplication to addition.

3.2 Enhancement of ICL Value

The section provides details regarding the enhancement steps taken to increase the ICL value, as the largest ICL value provides the most preferable clustering of the data points into subsets (Côme and Latouche, 2015). In other words, the model or algorithm iteratively updates the α_{gj}, β_{gj} and δ'_g values resulting in the generation of the updated matrix z which is initially fit with the help of EM algorithm, until an optimal value of ICL is reached.

The implementation of ICL enhancement comprises of three main stages, namely ICL Sweep, ICL Group Reduction and ICL Group Merge. The Figure 3.1 illustrates the improvement procedure of ICL value that takes place. The algorithm iteratively carries out these functions until convergence takes place, which then terminates the model, providing the updated matrix $[z]$, initial or original ICL value along with post processing increased ICL values and the optimal number of clusters for the dataset as the outputs.

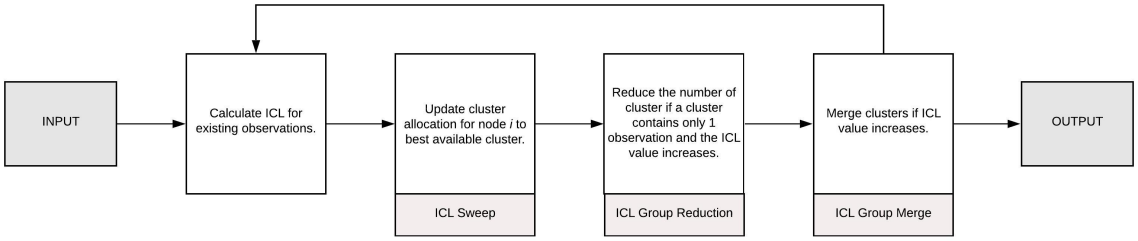


Figure 3.1: Workflow of the *exactICLforLCA* algorithm

The model takes the alpha, beta, delta, initial clusters value set by the analyser or the user along with the matrix form of dataset and fit matrix as input and calculates the ICL value for the existing set of observations that have been clustered to the number of subsets specified by the user. The enchantment is built on this ICL value and steps to revamp with the intent of increasing this ICL value is carried out. The sweep, group reduction and merge functions are continuously computed in the model until the optimum ICL value is obtained.

3.2.1 ICL Sweep

ICL is used as a clustering heuristic by updating the location of each observation to the best group G by comparing the observation's current ICL value with respect to the new cluster location with that of the previous ICL value. Suppose for a clustering configuration, called $z^{(0)}$, take an observation i at random, where $i \in Group\ g$, that is, $z_{ig}^{(0)} = 1$. A better cluster allocation is identified for this i with respect to the current allocation of all the other data points.

Suppose the new cluster allocation matrix is called $z^{(1)}$, the matrix will be the same as $z^{(0)}$ for all observations except i, that is, $z_{jg}^{(0)} = z_{jg}^{(1)}$ for all $g = 1, ..., G$, when $j \neq i$ and $z_{ig}^{(1)} = 0$ and $z_{ih}^{(1)} = 1$ are set for some $h \neq g$. The selection of the new cluster allocation is determined by computing the following,

$$\Delta_{g \to h} = ICL(Y, Z^{(1)}) - ICL(Y, Z^{(0)}).$$

The term $\Delta_{g\to h}$ is the difference in ICL the observation i is allocated to Group g versus Group h. Positive values indicate that the allocation is superior and negative values the opposite. The allocation of the observation back to the existing group provides a null value, that is, $\Delta_{g\to g} = 0$. $\Delta_{g\to h}$ is computed for all $h \neq g$ and if no positive values of Δ are obtained, i remains allocated to Group g, that is, $z^{(0)}$ matrix is retained. Otherwise, $z_{ig}^{(1)} = 0$ and $z_{ig'}^{(1)} = 1$ are set, where

$$g' = arg \max_{h=1,2,...G} \Delta_{g\to h}$$

Technical computation of the ICL sweep has been carried out in the package by creating a ICLSweep function which takes the fit matrix, matrix conversion of the dataset, g, alpha, beta, delta as inputs. As a primary step, the sweep function saves the ICL value which is calculated on the onset of the function. The goal of this stage is to place each node or observation in the most optimal cluster with respect to the matrix, which results in improvement of the ICL value.

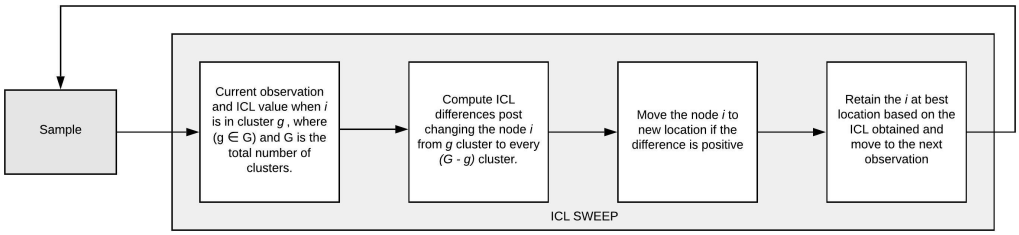


Figure 3.2: Workflow of the ICL Sweep function

The Figure 3.2 explains the functioning of the ICL sweep stage where, if the updated matrix z where the observation i has been moved from cluster g, where $g \in G$ to cluster h which is one of the $(G - g)$ clusters, that is, a cluster from the G set which is not g, provides a higher ICL when compared to matrix z where the observation i is in cluster g, the updated matrix is retained and the model moves on to the next observation.

For the purpose of individual observation's location update, a `sample()` is created of specified size, that is, `nrow(z)` from the elements of the matrix z and the sweep takes place in the order of placement in the generated sample. The model iteratively

runs the computation for all the observations in the order specified by the sample and generates new samples till the improvement in the ICL value halts. The ICL value for observation i being in cluster g, where $g \in G$ of matrix z, is calculated and checked against the value obtained when i is place in any of the *(G - g)* clusters, that is, all the clusters apart from the g cluster where it is originally placed and where all other observations are exactly in the same position as matrix z. If the gained ICL value is an improvement to the previous value, the observation i is moved to this new location.

3.2.2 ICL Group Reduction

Post computation of the sweep step, where the matrix z has all the observation at the most optimal allocations, if a particular cluster g, where $g \in G$, contains only a single observation, then the algorithm assumes that moving this singular observation i from the g cluster causes that subset to vanish, so that G value that is, the number of subsets or clusters is reduced by 1. In this case, the computation of the change in exact ICL is modified accordingly to

$$\Delta_{g \to h} = ICL(Y, Z', G-1) - ICL(Y, Z, G)$$

To compute this programmatically, the number of observations in each cluster was found using the `length()` function. If a cluster contained one node, then for this single observation i the sweep is carried out to find a location among other clusters under the condition that the ICL value increases. The ICL value for the original matrix with the cluster containing single observation is compared against the matrix where the observation is located in a different cluster changing the matrix z to *z'*, value for the particular observation while all other nodes remaining in the same location, along with a decrement in the total number of clusters from G to *(G-1)*.

3.2.3 ICL Group Merge

The updating process is repeated for each node i in z with the help of sweep and group reduction, post which the merging of clusters is computed, if it increases the exact ICL

value further. To merge two clusters, the ICL value is calculated where if attempting to merge clusters, for example, clusters g and h, such that all the labels or observations in g become labels of h. The new marginal likelihood calculations to get ICL where z' contains cluster h where observations of g are merged into it. Only if this difference is greater than 0, the merge between these two clusters is performed.

$$\Delta_{gh} = ICL(Y, Z', G-1) - ICL(Y, Z, G)$$

All pairwise merges are considered during the ICL merge group stage to make sure the optimum likelihood is obtained by the algorithm. Each group is merged with all other clusters from the total number of clusters individually to find the most profitable merge resulting in maximising the ICL value. The merged groups are then considered as a single group from next computation and the process is repeated again to enhance the ICL value.

The model breaks if all the clusters are merged down to 2, following which, a separate function to calculate the exact ICL value for a single cluster is carried out. If the value obtained is better than the ICL value for 2 clusters, the value is saved in a variable else, the single cluster is deemed as a bad fit and the value is ignored.

3.3 Termination of Algorithm

The *exactICLforLCA* algorithm is an iterative algorithm with specified conditions and guidelines to analyse if the model has fit the data and finished publishing it.

The algorithm or the model terminates once the ICL reaches convergence, or simply put, when the global maximum or the highest ICL is achieved. The algorithm runs the sweep and group reduction stages iteratively to obtain maximum ICL values and post achieving this, initiates the group merge stage. The model terminates if no further increase in the ICL value takes place post running the matrix through all possible pair combinations for all the clusters in the group merge stages.

The Δ value is calculated as a determinant to analyse the enhancement's profitability.

The Δ is the difference between the ICL value obtained at post computation and the ICL value before the the change was made. The changes are deemed as fit to be applied if the Δ is greater than 0 or positive. The positive Δ value indicates that the current exact ICL value is higher than the previous value and the changes made in favour of the objective of better clustering.

Another termination criteria is the number of maximum iterations that is specified based on the binary dataset being used. The model breaks when iteration count reaches the specified maximum iteration count.

Chapter 4

Application of *exactICLforLCA* Algorithm to Datasets

The chapter demonstrates the fit of the *exactICLforLCA* algorithm on the real world dataset, such as Alzheimer's dataset and simulated datasets. The section also investigates the performance of the model by attempting to retrieve the true structure of the data by comparing the pre-existing characteristics of data to the one fit by the model in the case of simulated dataset.

An `ICLSummary` function has been computed to summarize the outcome once the model is fit. The function returns the original ICL value calculated for the initial fit matrix generated using EM algorithm for the specified number of clusters G by the analyser. Post the processing carried out by the *exactICLforLCA* algorithm, the ICL value along with the number of clusters in the new fit matrix are returned. A dataframe with the cluster value and ICL value after each iteration of the model is also printed to illustrate the changes in number of clusters of the matrix and the increase in the exact ICL value.

4.1 Application to Alzheimer's Dataset

The application of the algorithm on real world data, such as, Alzheimer's dataset containing 240 observations and 6 variables, produced an optimally clustered matrix. The algorithm was fit using different number of clusters to observe the changes in the

clustering. All the results presented an increase in the ICL value, implicating that the data clustering improved successfully in the dataset. The details of the output, its interpretation and the mode of evaluation has been explained in the following section with the help of an example output, `ICLSummary`.

The optimum number of groups are estimated by the model as a part of the output upon application to Alzheimer's Dataset. The `ICLSummary` shown in Figure 4.1 illustrates the output obtained with an original ICL value of -877.02 when an initial clustering value is set to 9. The increased value of ICL post processing of - 793.03 is printed to show the improvement in the likelihood. The decreased number of clustering value, which is 2 has been displayed, depicting the changes that took place in the group reduction and group merge stages. The ICL value when the number of clusters, G is 1 is also printed, as it satisfies the condition that the ICL value for a single cluster is an improvement when compared to that of the likelihood in the presence of two clusters. The dateframe lists out the cluster and ICL value post each iteration step to view and analyse the optimization process carried out by the model.

```
> res <- ICLFit(Z, Y, G, alpha_var, beta_var, delta_var)
> ICLsummary()
[1] "Oringinal ICL value :  -877.02"
[1] "Initial number of clusters :  9"
[1] "ICL value post processing :  -793.03"
[1] "Number of clusters post processing :  2"
[1] "ICL value for a single cluster :  -789.21"
[1] "The cluster and ICL value after each iteration : "
  Cluster ICL.Value
1       9 -877.0195
2       8 -854.2385
3       7 -844.7850
4       6 -835.2719
5       5 -812.3223
6       4 -803.0231
7       3 -793.0287
8       2 -793.0287
>
```

Figure 4.1: An Example of result obtained for Alzheimer's Dataset

As a model evaluation process, the probability ratio distribution of observations among the clusters is executed between the model fit in the preprocessing step using expectation–maximization algorithm and the model fit using the *exactICLforLCA* algorithm. The `blca.em` function has been used to fit the model with the number of classes set for the LCA or the number of clusters set as 2, while the *exactICLforLCA* algorithm

Table 4.1: Probability of symptom occurrence in a cluster by expectation–maximization algorithm

	SYMPTOMS					
	Hallucination	Activity	Aggression	Agitation	Diurnal	Affective
1	0.07407407	0.5259259	0.08888889	0.05185185	0.1037037	0.5703704
2	0.08571429	0.8190476	0.40952381	0.74285714	0.4190476	0.9904762

Table 4.2: Probability of symptom occurrence in a cluster by exactICLforLCA algorithm

	SYMPTOMS					
	Hallucination	Activity	Aggression	Agitation	Diurnal	Affective
1	0.08839779	0.6906077	0.2596685	0.4254144	0.2707182	1
2	0.05084746	0.5423729	0.1355932	0.1355932	0.1525424	0

has been fit with initial clustering set as 7, which has optimized, reduced and merged the number of clusters to 2 post processing.

The Table 5.1 illustrates the probability parameters of symptoms being indicated by patients in the clusters fit by the `blca.em` model were number of classes was specified as 2. The generated clusters depict the possibility of a patient suggesting symptoms. The table displays the probability of a patient in cluster 2, showing 'Activity' symptom which is 81% and it relays that all cluster 2 observations are prone to have high 'Activity', 'Agitation' and 'Affective' symptoms.

The dataset when fit using *exactICLforLCA* algorithm, generates new clusters which have been optimised with the help of ICL value, each of the observation in the Alzheimer's dataset has been repeatedly computed to be placed in a cluster that will help produce a matrix divided in the best possible manner. Table 5.2, depicts the new clusters where in, the symptoms indication probability indicate that a patient in cluster 1 will definitely show 'Affective' symptom unlike in cluster 2, where none of the subjects will indicate the same symptom.

Bar chart depicting the probability of symptom occurrence in each cluster per symptom for both the expectation–maximization fit and *exactICLforLCA* fit are shown in Figures 4.2 and Figure 4.3 respectively. The plots show the difference in clusters generated by

both the fits. The enhancement of ICL values, an approach to obtain best possible clustering, results in the outcomes as depicted in the Figure 4.3.

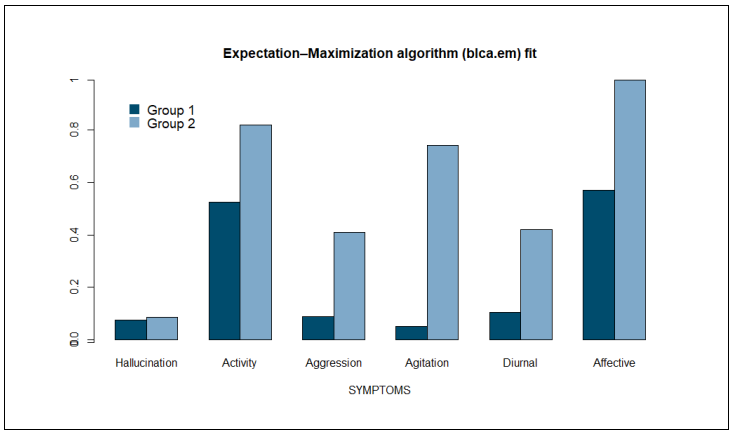


Figure 4.2: Bar chart depicting the division probability of nodes between 2 clusters when fit by the EM model

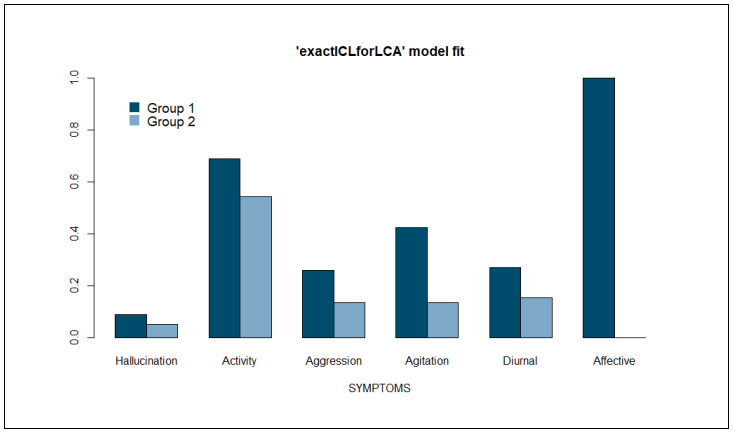


Figure 4.3: Bar chart depicting the division probability of nodes between 2 clusters when fit by the *exactICLforLCA* algorithm

4.2 Application to Simulated Dataset

The simulated datasets have been used as an important method of analysis which can be easily verified. The simulated data is created as explained in the section 1.4 and a `ICLSummary` is obtained for the same. An example of the summary for simulated data with probability parameters set as 0.7 and 0.3 has been shown in Figure 4.4. The summary provides the ICL value pre-processing and post processing along with clustered matrix value and iterative changes in both values in the form of a data frame. The initial ICL value of -4273.4 has been improved to -4078.6 by the *exactICLforLCA* algorithm, indicating that the clustering has improved. Results obtained from all the distinctive simulated datasets, depicted that ICL value increased suggesting that the data clustering improved successfully in all cases.

```
> res_final <- ICLFit(Z, Y, G, alpha_var, beta_var, delta_var)
> ICLsummary()
[1] "Oringinal ICL value :  -4273.4"
[1] "Initial number of clusters :  6"
[1] "ICL value post processing :  -4078.6"
[1] "Number of clusters post processing :  2"
[1] "The cluster and ICL value after each iteration : "
  Cluster ICL.Value
1       6 -4273.423
2       6 -4150.425
3       5 -4116.107
4       4 -4103.437
5       3 -4094.248
6       2 -4078.586
>
```

Figure 4.4: An Example of result obtained for Simulated Dataset

To evaluate the model, a rand Index value is calculated to examine the similarity between the the simulated clustered data generated by the user and the data clustered by the model. The comparison helps analyse the nature of both the models to verify if the true structure of the simulated data is returned by the model.

Adjusted Rand Index

Rand Index is a measure utilised particularly in data clustering, to measure of the similarity between two data clustering, that is, the simulated data matrix and matrix that was fit by the model. The `randIndex` value lies between 0 and 1, where 1 signifies perfect similarity between the two clustering outcomes, whereas, 0 depicts no similarity

Table 4.3: Simulated dataset matrix [*z_sim*] versus latent variable matrix [*z_new*]

	z_sim		
z_new		1	2
	1	786	12
	2	12	190

Table 4.4: ARI value for a simulated dataset

ARI
0.8927234

among them. Rand index, a measure of the percentage of correct decisions made by the algorithm, can be computed using the number of observations that are placed in similar clusters and those that are placed distinctly.

The `randIndex` function of `flexclust` library is used as a verification method to retrieve the true structure of the data and verify is the level of similarity among the clustering. The simulated data matrix [*z_sim*] is compared against the matrix fit by the *exactICLforLCA* algorithm [*z_new*]. The randIndex provides the `ari` value which is corrected for chance version of the Rand index value, helps verify the validity of the model for clustering the multivariate categorical data.

$$Adjusted\ Index\ (ari) = \frac{Index - ExpectedIndex}{MaxIndex - ExpectedIndex}$$

The `ari` value for the simulated data resulting in the output as shown in Figure 4.4 for 1000 data points out of which 976 observations have been placed similarly by the model as seen by Table 5.3, and the similarity rate is approximately 89.27% as depicted in Table 5.4.

The simulated dataset is run iteratively to obtain the mean of the `ari` and standard error, which are used as a marker to know the validity of the model. The number of times the simulation needs to be run is set to coagulate the individual `ari` value generated for each iteration to find the mean `ari` value illustrating the tendency of similarity between the models. A separate implementation has been programmed in the package to find the standard error and mean for any combination of simulated

data that is generated by the user to verify the average sync between the fits. For 100 iterations of a simulated dataset with 7 columns and initial clustering value set to 6, the mean `ari` of 0.883039 depicting the similarity among the models, with margin of error being 0.00801395 was obtained.

The average value of ICL obtained from 20 iterations of the `blca.em` algorithm is approx. -858.425, and -794.118 for *exactICLforLCA* for Alzheimer's dataset. For the same number of iterations, the approximate value for the `blca.em` and *exactICLforLCA* algorithm for simulated datasets are -4214.17 and -4073.4 (appendix .4). These values indicate that the *exactICLforLCA* algorithm produces a better ICL value by about 65.307 which is approx. a difference percentage of 7.79% when compared to `blca.em` for real dataset and the simulated dataset outputs indicate that the ICL value increases by approx. 140.77 (3.34%) when compared to `blca.em` irrespective of the algorithm's configuration.

Chapter 5

Discussion

The chapter discusses the research objective, architecture, background study and importance of the research. Model creation, core implementations, results and interpretation, along with a brief section on `RCPP` and its advantages when compared to `R` has been discussed.

The research presents an approach to cluster or group categorical data in the most suitable manner where the nodes in a subset are similar to each other while each subset is distinct from one another. The implementation is constructed on the goal to improve the quality of clustering which is the fundamental and one of most important steps of data analysis and to achieve this the integrated complete-data likelihood (ICL) method was applied on the latent class analysis (LCA) model to get the Exact ICL notation. The notation was computed programmatically and set as a benchmark to strengthen or increase the ICL value implicitly providing a matrix that has been optimally clustered.

The architecture of the dissertation involved understanding the LCA model and the ICL method. Bayesian inference, Beta distribution and Dirichlet distribution were utilised to obtain the equation generated by exposing the LCA model to the ICL method during the build of the model. LCA is a statistical modelling algorithm that is utilized to identify and describe latent classes, or hidden groups within a population of the dataset (Nylund-Gibson and Choi, 2018) and the ICL method helps to ameliorate the latent classes that are generated.

Computationally, `R` language was used to program the exact ICL method on LCA model. The programming was achieved using the bottom up approach, where in, the elementary function of calculating the ICL value was first generated and then functions to improve the ICL value were applied to the code. The build performed using the waterfall model, the sweep stage was added to obtain the result, then the group reduction functionality was the added, followed by the group merge methodology. Each stage played a significant role in the increment of the ICL and thus resulting in the improvement of the clustering for the introduced dataset. The execution of the ICL sweep stage was a laborious task as updating the cluster of each observation with respect to the sample's ICL value utilizing a greedy approach was tricky. As a solution, the original location was decided to be saved in a variable for retrieval and then computed along with all other cluster locations.

As a prerequisite, the model is primarily fit with the help EM algorithm to find the local maximum likelihood or maximum a posteriori (MAP) estimates of parameters and the model works on this fit model. Gibbs sampling or a gibbs sampler, a Markov chain Monte Carlo algorithm, Variational Bayes Algorithm, combination of EM Algorithm utilising empirical bootstrapping were some of the method could have been used, but the decision to apply EM algorithm using the BayseLAC library was taken at the beginning of research and was carried out as per the design. The model fit with this algorithm acts as a foundation on which the enhancement of the matrix is based upon.

A package named *exactICLforLCA* was created as the last step of bottom up approach. The creation of package was undertaken as a step to help users trying to achieve clustering on binary latent class analysis data. The functions were broken for easier access of code and understanding, separate files where created for real data and simulated data to implementation to aid the application of the algorithm on the data and to make it uncomplicated for any user. The algorithm written in R is available in the Git hub at `https://github.com/Manasimohan/exactICLforLCA`

The algorithm is centred around updating the fit matrix z, which is the clustered dataset generated by the mode. The z matrix is repeatedly generated until convergence, which is attaining the global maximum or maximum a posteriori (MAP) estimates of parameters with highest ICL value. Simply put, when all the nodes of the dataset have

been placed in the most optimal cluster during the clustering the process, the generation of new z is ceased. The algorithm is dependent on three main inputs α, β and δ which help generate the hyperparameters α_{gj}, β_{gj} and δ'. The model iteratively updates these hyperparameters to generate new z repeatedly. The hyperparameters α_{gj}, β_{gj} are generated utilising the inputted converted matrix $[y]$ and the newly generated matrix z, whereas the δ' is dependent solely on the z matrix. The hyperparameters help generate z and in turn this matrix generates new hyperparameter value. This codependent iterative generation halts when the latent variable matrix z reaches optimum clustering. The mechanism is aided by the ICL value whose increment indicates positive movement towards convergence.

The *exactICLforLCA* algorithm has been introduced with datasets which support LCA modeling algorithm that need to be clustered, these were converted into matrix and utilised as one of the inputs during the computation along with the initial number of clustering set by the analyser with the α, β and δ . The algorithm also accepts the primarily fit matrix using the EM algorithm with the number of clusters specified by the analyser. Termination of the algorithm generates the optimally clustered matrix and prints information about the matrix such as the current ICL value and the number of clusters it contains. List of cluster values and ICL values along with the primary ICL value and number of clusters are also printed to contemplate.

Application of the *exactICLforLCA* algorithm was carried out on two different type of datasets, real word data and simulated data. Real data implementation was conducted to illustrate algorithm's computation on real world data. Simulated data implementation was carried out to measure the of similarity between the simulated data and the fit data. The algorithm was run multiple times, for different number of clusters for real data and discrete datasets, generated by specifying the number of observations, initial cluster value and the number of iterations in the case of simulated data to run the evaluation of clusters. The generated fit matrix was subjected to comparison of number of labels, division of observations and length of clusters to study the difference between the initial fit and the final fit. To gain an understanding of average increase in the ICL value, the average value for the `blca.em` was subjected against the *exactICLforLCA* value for both real and simulated datasets, and the approx increase in ICL value for both the datasets and the difference percentage was found.

Results obtained in the form of `ICLsummary` assisted in the understanding of changes that the implemented stages brought about in the attempt to improve the ICL value and thus enhancing the clustering of the dataset. The real dataset and the simulated datasets, both illustrated an increase in the ICL value which implied the improvement in data clustering. The adjust rand index, a cluster evaluation method and a measure to check the level of similarity among two clusters was applied to the simulated data implementation as the knowledge regarding the clustering is known prior, which can be used to compare the new fit model that has been generated. Some of the other algorithm evaluation methods are accuracy, recall and precision where the presence of true label is required to evaluate and predicate labels, where as, rand index is purely used to compare clusters. The `randIndex` helped verify the similarity among was clusters and mean of the generated `ari` and standard error `std` was calculated for multiple fits to achieve clear knowledge regarding the general behavior and the probability of similarity among the models.

The behaviour analysis of the model in the presence of a categorical data that is not LCA needs to be undertaken along with application of algorithms other than the EM during the initial fit to check the robustness of the algorithm. Investigation of the algorithm under diverse situations, such as in the presence of a clustering where few of the clusters are closely similar to each other would help check the effectiveness of the algorithm. Execution time consumption is an area that needs to be refined since approx. 12 min was consumed to model Alzheimer's dataset with initial number of clustering set to 9 when run on the local machine. A possible solution for this would be the use of package called `Rcpp` which provides R functions along with C++ classes and offer a seamless integration of R and C++.

5.1 Rcpp Package

The `R` language is an interpreted language which is typically accessed through a command-line interpreter and was designed with the purpose to make data analysis and statistics easier. R is fundamentally used to understand data and although the programming in R is simple, it snaffles the computational power for the machine that it is executed from. Beyond performance limitations sourced due to the design and the implementa-

tion, programming in R code is supposed to be slow elementally because of its drafting. R was constructed valuing flexibility over performance but is a popular language which can ingest data in many formats, aggregate, summarize and visualize the data. It can also model in a diverse manner which can be extended further with use of packages.

The `Rcpp` package (Eddelbuettel et al., 2011)(Eddelbuettel, 2013)(Eddelbuettel and Balamuta, 2018) makes the connection from C++ to R easily, contrast to writing C for use in R which is a tedious task in comparison. Rcpp provides a clean, approachable API that allows programmer to produce a high-performance code, insulated by R's arcane C API.

Rcpp is easy to learn, implement and use as it avoids operation system complexities which are dealt by the R infrastructure. It deals with the typical bottlenecks that are faced by R, such as

- Loops, which cannot be easily vectorised as subsequent iterations depend on previous iteration values.
- Problems that require advanced data structures and algorithms that are not available in R.
- Recursive functions or functions that involve calling functions repeatedly. The overhead of a recursive function in C++ is much lower when compared to R.

The primitive ICL calculation function has been separately implemented in R and by utilising Rcpp package (appendix .2 and appendix .3 respectively) to view the difference. From the implementation it is explanatory that even though programming using Rcpp is complicated when compared to the R code, the execution time will decreased when compared to the time taken when only R is utilised. The time taken from each of the execution of these files containing primitive functionalities was timed, and the difference between both the execution was minute, yet it is clear that when the model is generated completely utilizing the package, the execution time will decrease when compared to present scenario.

R is user friendly and easy programming language and takes more execution time as the back end carries out more cycles to execute. R is utilised with the approach of

obtaining the solutions quickly than to develop a system that is diverse, meaning that it is relatively easy and fast to code in R. Whereas, the Rcpp package is a package that enables to implement R functions in C+, it is implemented in a manner such that the C++ code style is similar to R. Rcpp does not sacrifice execution speed for the ease of use unlike R, and is a go to package to obtain high performance outcome in less execution time.

Chapter 6

Conclusion

The dissertation presents an approach to improve the clustering obtained for the multivariate categorical data, analysed by modelling algorithm such as latent class analysis (LCA) with the help of exact ICL method. Description of the motivation for the research and the importance of clustering has been provided which is the foundation of this dissertation. Basic concepts such as Bayesian Inference, Beta Distribution, Dirichlet Distribution have been discussed and their involvement in the generation of the exact ICL method on LCA modelling algorithm has been explained.

Theoretical concepts of LCA, ICL and the clustering of LCA algorithm utilizing the ICL have been elucidated for better understanding of the elemental portions of the research. The algorithm was implemented with the goal to increase the ICL value calculated by the notation generated by implementation of ICL on LCA which results in improvement of clustering. *exactICLforLCA*, a package in R was created in fulfilling this objective, which is available at `https://github.com/Manasimohan/exactICLforLCA` .

Application of algorithm on real world data and simulated data resulted in production of matrices with high ICL value which indicated that the dataset clustering improved, accomplishing the aim of the research. Comparison was carried out between the matrix generated by the model to view the changes in the clusters against the initial matrix's cluster generated by the EM algorithm. Evaluation of the model was conduction utilizing the rand index in the case of simulated data to check the contrast between the randomly generated simulated data for the parameters provided and the matrix gen-

erated by the mode. The comparison was carried out iterative to obtain the margin of error and mean ari for simulated datasets and it was observed that the clusters are close to similar in most of the cases, illustrating that slight changes in the matrix can lead to great improvement in the clustering.

6.1 Future Work

As a part of future implementations, it would be interesting to apply the exact ICL method on clustering analysis models other than LCA, such as KNN. The expectation-maximization algorithm can be replaced by other algorithms to compare and analyse the importance of initial fit based on which the model operates. The computational processing and performance of exact ICL on LCA model can be introduced with salable computing which would compute parallel processing to generate multiple clustering models which could be utilised to improve and enhance the model. It would be appealing to generate the model with the application of Rcpp to compare the execution time and energy consumed by the machine.

Bibliography

Bartholomew, D. J. and M. Knott (1999). Latent variable models and factor analysis, volume 7 of kendall's library of statistics. *Arnold,*.

Biernacki, C., G. Celeux, and G. Govaert (2000). Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*(7), 719–725.

Biernacki, C., G. Celeux, and G. Govaert (2010). Exact and monte carlo calculations of integrated likelihoods for the latent class model. *Journal of Statistical Planning and Inference 140*(11), 2991–3002.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

Côme, E. and P. Latouche (2015). Model selection and clustering in stochastic block models based on the exact integrated complete data likelihood. *Statistical Modelling 15*(6), 564–589.

Eddelbuettel, D. (2013). *Seamless R and C++ integration with Rcpp*. Springer.

Eddelbuettel, D. and J. J. Balamuta (2018). Extending r with c++: a brief introduction to rcpp. *The American Statistician 72*(1), 28–36.

Eddelbuettel, D., R. François, J. Allaire, K. Ushey, Q. Kou, N. Russel, J. Chambers, and D. Bates (2011). Rcpp: Seamless r and c++ integration. *Journal of Statistical Software 40*(8), 1–18.

Erosheva, E. A., S. E. Fienberg, and C. Joutard (2007). Describing disability through individual-level mixture models for multivariate binary data. *The annals of applied statistics 1*(2), 346.

Garrett, E. S. and S. L. Zeger (2000). Latent class model diagnosis. *Biometrics 56*(4), 1055–1067.

Marbac, M. and M. Sedki (2017). Variable selection for model-based clustering using the integrated complete-data likelihood. *Statistics and Computing*, 1049–1063.

Moran, M., C. Walsh, A. Lynch, R. F. Coen, D. Coakley, and B. A. Lawlor (2004). Syndromes of behavioural and psychological symptoms in mild alzheimer's disease. *International Journal of Geriatric Psychiatry 19*(4), 359–364.

Nobile, A. and A. T. Fearnside (2007). Bayesian finite mixtures with an unknown number of components: The allocation sampler. *Statistics and Computing 17*(2), 147–162.

Nylund-Gibson, K. and A. Y. Choi (2018). Ten frequently asked questions about latent class analysis. *Translational Issues in Psychological Science 4*(4), 440.

Rohe, K., T. Qin, and B. Yu (2016). Co-clustering directed graphs to discover asymmetries and directional communities. *Proceedings of the National Academy of Sciences 113*(45), 12679–12684.

White, A. and T. B. Murphy (201). Bayesian variable selection for latent class analysis using a collapsed gibbs sampler. *Statistics and Computing*, 511–527.

White, A. and T. B. Murphy (2014). Bayeslca : An r package for bayesian latent class analysis.

WYSE, J., N. FRIEL, and P. LATOUCHE (2017). Inferring structure in bipartite networks using the latent blockmodel and exact icl. *Network Science 5*(1), 45–69.

Appendix

.1 Steps to Create R Package

1. Open a new project in RStudio. Go to the 'File' menu and click on `'New Project'`, select `'New Directory'` followed by *'R Package'* to create a new R package.

2. Enter name of the package under `Directory Name`, click on `Create Project`

3. Install packages *devtools* and *roxygen2*

4. Click on `Build` option on the main tab, followed by `Configure Build Tools`

5. Click the checkbox `Generate documentation with Roxygen` and enable all options when prompted.

6. Add or create .R files under the `R` directory of the package under Files tab.

7. Click on the `More` options under the `Build` tab of the package and choose `Clean and Rebuild` option which will build the package from source and load the package into the current `R` session.

.2 R Code for Primitive ICL Value Calculation

```
#' ICL Calculation
#'
#' @param Z, Y, G, alpha and beta
#' @return ICL

ICLCalc <- function(alpha_var, beta_var, G, Y, Z, delta_var) {
  # set variables
  delta <- rep(delta_var , G)
  r <- ncol(Y)

  # initalise alpha and beta matrix
  alpha_gj <- beta_gj <- matrix(0, nrow = G, ncol = r)

  # initalise delta matrix
  delta_prime <- delta + apply(Z, 2, sum)

  # calc alpha and beta of groups based on Y and Z
  for(j in 1:r){
    for(g in 1:G) {
      temp_alpha <- 0
      temp_beta <- 0
      for(i in 1:nrow(Z)) {
        temp_alpha <- temp_alpha + Z[i, g] * Y[i, j]
        temp_beta <- temp_beta + Z[i, g] * (1- Y[i, j])
      }
      alpha_gj[g, j] <- alpha_var + temp_alpha
      beta_gj[g, j] <- beta_var + temp_beta
    }
  }

  # log beta fun
  log_beta_vec <- function(delta) {
    l_num <- sum(lgamma(delta))
```

```
    l_denom <- lgamma(sum(delta))
    l_num - l_denom
  }

  # first eqn
  first_var <- log_beta_vec(delta_prime) - log_beta_vec(delta)

  # second eqn numerator value
  b_num <- 0
  for (g in 1:G)
  {
    for(j in 1:r)
    {
      b_num = b_num + lbeta(alpha_gj[g,j], beta_gj[g,j])
    }
  }

  # second eqn denomenarator value
  b_denom <- G * r * lbeta(alpha_var, beta_var)

  # second eqn
  sec_var <- b_num - b_denom

  # ICL calc
  ICL <- first_var + sec_var

  ICL
}
```

.3 Rcpp Code for Primitive ICL Value Calculation

```
#include <Rcpp.h>
using namespace Rcpp;

// This is a simple example of exporting a C++ function to R. You can
// source this function into an R session using the Rcpp::sourceCpp
// function (or via the Source button on the editor toolbar). Learn
// more about Rcpp at:
//
//   http://www.rcpp.org/
//   http://adv-r.had.co.nz/Rcpp.html
//   http://gallery.rcpp.org/
//

NumericMatrix repeat(int val, int times) {
  NumericMatrix result(1, times);

  for(int i = 0; i < times; i++) {
    result(0, i) = val;
  }
  return result;
}

NumericMatrix applysum(NumericMatrix mat) {
  NumericMatrix result(1, mat.ncol());

  for(int i = 0; i < mat.nrow(); i++) {
    for(int j = 0; j < mat.ncol(); j++) {
      result(0, j) = result(0, j) + mat(i, j);
    }
  }
  return result;
}
```

```
float applysum_vec(NumericVector vec) {
  float result = 0;

  for(int i = 0; i < vec.length(); i++) {
    result = result + vec[i];
  }
  return result;
}

NumericMatrix matsum(NumericMatrix mat1, NumericMatrix mat2) {
  NumericMatrix result(mat1.nrow(), mat1.ncol());

  for(int i = 0; i < mat1.nrow(); i++) {
    for(int j = 0; j < mat1.ncol(); j++) {
      result(i, j) = mat1(i, j) + mat2(i, j);
    }
  }

  return result;
}

float log_beta_vec(NumericMatrix delta) {
  // for l_num
  float l_num = applysum_vec(lgamma(delta));

  // for l_denom
  float l_denom = lgamma(applysum_vec(delta));

  return l_num - l_denom;
}


// [[Rcpp::export]]
float ICLCalc(int alpha_var, int beta_var, int G, NumericMatrix Y,
    NumericMatrix Z, int delta_var) {
```

```
NumericMatrix delta = repeat(delta_var, G);

// find ncol(Y)
int r = Y.ncol();

// initalise alpha and beta matrix
NumericMatrix alpha_gj( G , r );
NumericMatrix beta_gj( G , r );

//NumericMatrix delta_temp = ;
NumericMatrix delta_prime = matsum(applysum(Z) , delta);


// calc alpha and beta of groups based on Y and Z
for(int j = 0; j <= r; j++){
  for(int g = 0; g <= G; g++) {
    int temp_alpha = 0;
    int temp_beta = 0;
    for(int i = 0; i <= Z.nrow(); i++) {
      temp_alpha = temp_alpha + Z(i, g) * Y(i, j);
      temp_beta = temp_beta + Z(i, g) * (1- Y(i, j));
    }
    alpha_gj(g, j) = alpha_var + temp_alpha;
    beta_gj(g, j) = beta_var + temp_beta;
  }
}

// first eqn
float first_var = log_beta_vec(delta_prime) - log_beta_vec(delta);

float b_num = 0;

for (int g = 0; g<G; g++)
{
```

```
    for(int j = 0; j<r; j++)
    {
      b_num = b_num + R::lbeta(alpha_gj(g,j), beta_gj(g,j));
    }
  }

  // second eqn denomenarator value
  float b_denom = G * r * R::lbeta(alpha_var, beta_var);

  // second eqn
  float sec_var = b_num - b_denom;

  //ICL calc
  float ICL = first_var + sec_var;

  return ICL;
}
```

.4 Real and Simulated Dataset Outputs

Table 1: Results for Real Data Implementation

Initial ICL Value	Excat ICL Value
-890.967	-790.1847
-878.1975	-798.7279
-883.3904	-801.5593
-854.1605	-795.7536
-823.3164	-793.0287
-846.5878	-790.1847
-824.7602	-794.6988
-892.1219	-793.0287
-884.141	-793.0287
-845.0802	-793.0287
-862.27229	-794.32238
-879.5327	-793.0287
-890.2672	-795.7536
-870.9307	-793.0287
-865.0573	-795.7536
-849.8489	-795.7536
-846.5878	-790.1847
-823.2069	-795.7536
-874.9949	-793.0287
-831.5349	-794.6988
-823.3164	-793.0287
Average : -858.4251405	Average : -794.1176173
Difference percentage: 7.491337352	
Average Increase in ICL value : 64.30752318	

Table 2: Results for Simulated Data Implementation

Initial ICL Value	Excat ICL Value
-3787.1	-3659.5
-3694.2	-3677.5
-4653.6	-4483.4
-4745.5	-4503.1
-4745.5	-4503.1
-3888.1	-3676.1
-3878.2	-3855.8
-3889.8	-3795.8
-4528	-4266.3
-4491	-4339.8
-4487	-4319.7
-4296	-4236.6
-4562.7	-4405.1
-4562.7	-4405.1
-3833.6	-3694.9
-3948.6	-3755.4
-3840.8	-3802.3
-3914.6	-3848.9
-3775.3	-3656.4
-3794.4	-3657.3
-3865.6	-3835.9
Average : -4214.17	Average : -4073.4
Difference percentage: 3.340396804	
Average Increase in ICL value : 140.77	