ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF STATISTICAL SCIENCES

SECOND CYCLE DEGREE IN

STATISTICAL SCIENCES

FLEXIBLE LATENT CLASS MODELS WITH CONCOMITANT VARIABLES BASED ON B-SPLINES: A COMPARISON AMONG ESTIMATION METHODS

Dissertation in Statistical Models and Applications

Supervisor

Prof. Giuliano Galimberti

Co-Supervisor

Marco Berrettini, Ph.D.

Defended by

Lorenzo Bocchio

Academic Year 2022/2023

Abstract

The aim of the thesis is to compare different procedures for estimating latent class models with concomitant variables. More specifically, 1-Step, 2-Step and 3-Step estimation methods are examined in the context of flexible modelling, with covariate effects represented using B-splines. A simulation study was conducted to compare the three methods, revealing differences in terms of root mean square error and roughness of the estimated functions. The choice of the estimation method proved to have an impact also on the determination of the number of latent classes. Furthermore, the thesis describes an application to real data, focusing on the voting behaviour of UK Members of Parliament regarding several divisions related to Brexit.

Keywords: Maximum Likelihood, Stepwise Procedures, Cluster Analysis, Mixture Models, Brexit

Contents

Introduction **1**

1 Concomitant Variable Latent Class Models **3**

1.1 Latent Class Models 3

1.1.1 History of Latent Class Models 3

1.1.2 Latent Class Models: Basic Ideas and Main Features 4

1.1.3 Latent Class Models' Definition 5

1.2 Concomitant Variable Latent Class Models 6

2 Adding Flexibility with B-splines **11**

2.1 History of B-splines 11

2.2 Definition of B-splines 11

2.2.1 B-splines in Statistical Modeling 13

2.2.2 P-splines 13

3 Estimation Methods **15**

3.1 The Simplex Algorithm in Latent Class Analysis 15

3.2 1-Step Method 15

3.3 3-Step Method 20

3.3.1 Bias in 3-Step Procedures 22

3.4 2-Step Method 23

3.4.1 Properties of 2-Step Estimators 25

4 Simulation Study **27**

4.1 Data Generation 27

4.2 B-spline Generation 28

4.3 Creation and Specification of the Models 29

4.4 Results with a Fixed Number of Latent Classes 29

4.4.1 Comparative Analysis of f_2 and $\hat{f}_2$ Across Estimation Methods . . . 30
4.4.2 Comparative Analysis of Root Mean Square Errors Across Estimation Methods . . . 33
4.4.3 Roughness of the Estimated Function . . . 34
4.4.4 Selection of the Number of Internal Knots . . . 35
4.5 Results about the Selection of the Number of Latent Classes . . . 36
4.5.1 Comparative Analysis of the Optimal Number of Latent Classes Across Estimation Methods . . . 37
4.5.2 Comparative Analysis of Selected Latent Classes' Weights Across Estimation Methods . . . 38
4.5.3 Conclusions on the study with a variable number of latent classes . . . 40

5 Brexit Data Analysis 43
5.1 Introduction . . . 43
5.2 Description of the Data . . . 44
5.3 Optimal Number of Latent Classes . . . 46
5.4 Clustering with "Leave" as Concomitant Variable . . . 47
5.4.1 Results obtained using the 1-Step method . . . 50
5.4.2 Results obtained using the 2-Step method . . . 57
5.4.3 Results obtained using the 3-Step method . . . 62
5.5 Clustering with "ExpEntropy" as Concomitant Variable . . . 63
5.5.1 Results obtained using the 1-Step method . . . 66
5.5.2 Results obtained using the 2-Step and 3-Step methods . . . 71
5.6 Clustering Methods and Concomitant Variables Comparison . . . 75
5.6.1 The Adjusted Rand Index . . . 75
5.6.2 Comparative Analysis within "Leave" Clustering . . . 76
5.6.3 Comparative Analysis within "ExpEntropy" Clustering . . . 76
5.6.4 Inter-Variable Comparative Analysis . . . 77

6 Conclusions 79

Acknowledgements 81

Bibliography 83

A Simulation Study Code **87**
A.1 Data Generation 87
A.2 B-spline Generation in Simulation Study 88
A.3 Creation of the Model for a Fixed Number of Latent Classes 89
A.3.1 Dealing with Label Switching 95
A.3.2 Creation of f_2 95
A.3.3 Estimating f_2 with B-splines 95
A.3.4 Calculating Root Mean Square Error 95
A.3.5 Calculating the Roughness Measure 96

B Brexit Data Code **97**
B.1 Choice of Latent Classes' Number 97
B.2 B-spline Generation in Brexit Analysis 99
B.3 Creation of the Model for a Variable Number of Latent Classes . . . 100

C Simulation Study Figures **107**
C.1 Study with a Fixed Number of Latent Classes 107
C.1.1 Comparing f_2 and $\hat{f}_2$ with Sample Size = 500 and $\theta = 0.6$. 107
C.1.2 Comparing f_2 and $\hat{f}_2$ with Sample Size = 1000 and $\theta = 0.6$ 109
C.1.3 Comparing f_2 and $\hat{f}_2$ with Sample Size = 2000 and $\theta = 0.6$ 110
C.1.4 Comparing f_2 and $\hat{f}_2$ with Sample Size = 500 and $\theta = 0.7$. 112
C.1.5 Comparing f_2 and $\hat{f}_2$ with Sample Size = 1000 and $\theta = 0.7$ 113
C.1.6 Comparing f_2 and $\hat{f}_2$ with Sample Size = 2000 and $\theta = 0.7$ 115
C.1.7 Comparing f_2 and $\hat{f}_2$ with Sample Size = 500 and $\theta = 0.8$. 116
C.1.8 Comparing f_2 and $\hat{f}_2$ with Sample Size = 1000 and $\theta = 0.8$ 118
C.1.9 Comparing f_2 and $\hat{f}_2$ with Sample Size = 2000 and $\theta = 0.8$ 119
C.1.10 Comparing f_2 and $\hat{f}_2$ with Sample Size = 500 and $\theta = 0.9$. 121
C.1.11 Comparing f_2 and $\hat{f}_2$ with Sample Size = 1000 and $\theta = 0.9$ 122
C.1.12 Comparing f_2 and $\hat{f}_2$ with Sample Size = 2000 and $\theta = 0.9$ 124
C.2 Roughness measure 125
C.3 Root Mean Square Error 132
C.4 Internal Knots selected by Estimation Method 138

D Brexit Analysis Figures **141**
D.1 Relative Log-Odds with 1-Step and "Leave" Variable 141
D.2 1-Step Clusters with "Leave" as Concomitant Variable 146
D.3 Relative Log-Odds with 2-Step and "Leave" Variable 151

D.4 2-Step Clusters with “Leave” as Concomitant Variable 155
D.5 Relative Log-Odds with 3-Step and “Leave” Variable 160
D.6 3-Step Clusters with “Leave” as Concomitant Variable 164
D.7 Relative Log-Odds with 1-Step and “ExpEntropy” Variable 169
D.8 1-Step Clusters with “ExpEntropy” as Concomitant Variable 174
D.9 Relative Log-Odds with 2-Step and “ExpEntropy” Variable 179
D.10 2-Step Clusters with “ExpEntropy” as Concomitant Variable 182
D.11 Relative Log-Odds with 3-Step and “ExpEntropy” Variable 185
D.12 3-Step Clusters with “ExpEntropy” as Concomitant Variable 188

Introduction

The thesis aims at comparing different parameters estimation procedures of concomitant variable latent class models, investigating the effectiveness of each approach. In particular, it examines 1-Step, 2-Step, and 3-Step estimation methods and their application in the context of flexible models, where the effect of the covariate is represented using a B-spline function, allowing for more refined and adaptable modeling.

The exploration is driven by the need to understand and model complex data structures, characterized by hidden variability and the significant impact of concomitant variables on such variability.

Chapter 1 introduces latent class models, detailing their significance and utility in statistical analysis. It discusses the evolution of these models and the implementation of concomitant variables, setting the stage for the subsequent methodological exploration.

Chapter 2 presents B-spline functions and their application in statistical modeling, specifically how they are employed to enhance flexibility, and their possible penalization to balance the fit to the data and the smoothness of the curve.

Chapter 3 consists of a simulation study, designed to compare the 1-Step, 2-Step, and 3-Step estimation methods in the context of flexible concomitant variable latent class models. The investigation highlights the differences in root mean square error and the roughness of the estimated functions among the methods. Additionally, it examines the impact that each estimation procedure has on the determination of the number of latent classes.

Chapter 4 applies the discussed methodologies to a real-world scenario, analyzing the voting behaviour of UK Members of Parliament on Brexit-related divisions. This application demonstrates the ability of the proposed modeling approach, providing insights into complex political dynamics, and its capability to identify within-party views.

The thesis concludes with a discussion of the findings of the research, and a sug-

gestion of possible directions for future research. This concluding chapter aims to reflect on the effectiveness of the flexible concomitant variable latent class model and the considered estimation procedures, in both simulation and practical application.

Chapter 1

Concomitant Variable Latent Class Models

1.1 Latent Class Models

1.1.1 History of Latent Class Models

The history of Latent Class Models (LCMs) has found its main development during the second half of the 20th century. Between the '50s and the '60s the focus was predominantly on dichotomous observed variables, starting with the introduction of the so-called "latent structure models" by Paul Lazarsfeld in 1950 with "The logical and mathematical foundations of latent structure analysis", as cited in Hagenaars and McCutcheon (2002). During this era, several estimation methods for the parameters were introduced, though many faced limitations in consistency and efficiency, or were too computationally intensive for practical application.
This changed in the mid-1970s with the introduction of more general LCMs and simpler methods for maximum likelihood estimation, thanks to the work done by Goodman (1974). These new models were not only applicable to dichotomous data but also to polytomous observed variables and could incorporate multiple latent variables. Many important advancements in LCMs have been proposed since then, with the most important extension possibly being models including concomitant variables, thanks to the work of Dayton and Macready (1988).
Another significant improvement has been the LCM's integration with Item Response Theory's Rasch Model and latent classes, focusing on the relationship between individuals' latent traits and their performance to enable a more refined analysis in educational assessments. It allows for different sets of parameters between

the latent classes, with maximum likelihood estimates for each class, without any a priori assumption (Rost, 1990).
The development of the Expectation-Maximization (EM) algorithm by Dempster et al. (1977) revolutionized the computational efficiency of LCMs. Designed for models with latent variables, the algorithm alternates between estimating missing data and updating the model parameters to maximize the likelihood of observed data. The introduction of this algorithm marked a significant milestone in making LCMs more accessible and practicable for a wide range of applications.
Furthermore, the application of LCMs in longitudinal data analysis emerged, especially in social sciences and epidemiology, allowing the tracking and the analysis of behavioural patterns or disease progression over time (Nagin, 2005).
The evolution from early dichotomous models to more comprehensive frameworks has significantly enhanced the ability to uncover latent patterns and structures within the data, paving the way for recent insights and applications.

1.1.2 Latent Class Models: Basic Ideas and Main Features

Latent Class Models have become a useful tool whenever a construct is conceptually difficult to quantify and measure directly. This often happens when dealing with psychological traits, social attitudes, or medical conditions that are not directly observable and are conceptually difficult to define (Bandeen-Roche et al., 1997). In these instances, it can be interesting to search for the reasons behind the confounding between the observed variables, and to approximate their distribution in the population.
In these cases, the standard LCM can capture latent classes by means of multiple discrete variables (Hagenaars and McCutcheon, 2002) that act as indicators of implicit concepts, succeeding in segmenting the population into subgroups. The latent variable is often conceptualized as the underlying factor influencing the indicators, rather than considering these indicators as causally related. Essentially, the latent variable can be perceived either as a precursor that creates spurious relationships among the observed variables, or as a mediating factor that fully explains the observed association between the measured variables.
The introduction of the latent classes leads to the uncovering a simpler underlying structure that clarifies the relationships between the manifest variables by accounting for and thereby eliminating confounding influences. In fact, conditioning upon the latent variable, the responses of the manifest variables are assumed to be statis-

tically independent, with this assumption typically referred to as "conditional" or "local" independence (Linzer and Lewis, 2011).
A latent class model, although not inherently determining the number of latent classes in a dataset, can be evaluated using a range of measures for assessing model fit and parsimony, which then enable to make informed decisions about the number of classes that best represent the underlying structure in the data, balancing theoretical considerations with empirical evidence.
At its core, a latent class model is a form of finite mixture model, where the unseen latent variable represents class membership, functioning as a nominal variable. The model conceptualizes the observed data as a mixture of various component distributions, each corresponding to a different latent class. The model parameters include the proportion of the population in each latent class and the conditional probabilities of each manifest variable's response, given the latent class membership. By aggregating this information, the model approximates the distribution of cases in the observed data.
The latent class model effectively groups observations with similar response patterns on the manifest variables into the same latent classes, revealing the underlying subgroups within the data.

1.1.3 Latent Class Models' Definition

Consider a scenario where $y_i = \{y_{ij}\}_{j=1}^{J}$ is a vector-valued data point for the ith observation, where each y_{ij} representing a polytomous response, with $r = 1, \ldots, R_j$ denoting the possible categories for the jth response ($j = 1, \ldots, J$). Suppose that there are L latent classes c_l ($l = 1, 1 \ldots, L$). The basic Latent Class Model (LCM) can be articulated as:

$$Pr(Y \mid c_l) = \prod_{j=1}^{J} \prod_{r=1}^{R_j} [\alpha_{jrl}]^{\delta_{ij}} \tag{1.1.1}$$

accompanied by:

$$Pr(Y) = \sum_{l=1}^{L} \Theta_l Pr(Y \mid c_l). \tag{1.1.2}$$

In this framework, α_{jrl} indicates the probability of response r for variable j given latent class l, Θ_l represents the mixing parameters, and δ_{ij}, serving as the Kronecker delta, is defined by:

$$\delta_{ij} = \begin{cases} 1 & \text{iff } y_{ij} = r, \\ 0 & \text{otherwise.} \end{cases} \tag{1.1.3}$$

The model parameters are bound by the following conditions:

$$\sum_{r=1}^{R_j} \alpha_{jrl} = 1 \tag{1.1.4}$$

for each j, l, and also:

$$\sum_{l=1}^{L} \Theta_l = 1. \tag{1.1.5}$$

Latent class models rest on the foundational principle of conditional independence (Goodman, 1974), asserting that variables are statistically independent within any given latent class. Latent class models operate on the principle of conditional independence (Goodman, 1974). This means that within each latent class, the variables are considered to be statistically independent from one another. Additionally the latent classes are assumed to be mutually exclusive and exhaustive, as demonstrated through the aggregation of terms in equation 1.1.2 and constrained as per equation 1.1.5. From a theoretical standpoint, the model encapsulated in equations 1.1.1 to 1.1.5 represents a mixture of product-multinomial distributions with specified mixing parameters Θ_l, and distinct multinomial processes specified by the probabilities α_{jrl} (Dayton and Macready, 1988).
This model framework has been iteratively refined through the years, with the contribution of Goodman (1974), who explored methodologies for analyzing qualitative data systems incorporating unobservable variables, and Haberman (1974), with the development of log-linear models for categorical data tables derived through indirect observation.

1.2 Concomitant Variable Latent Class Models

In applications of the LCM, the interest often switches to employ the latent classes for more complex analysis. A crucial extension of the LCMs is the Concomitant Variable Latent Class Model (CVLCM), which aims at modeling the relationship between the basic parameters with concomitant variables, introduced by Dayton and Macready (1988). These concomitant variables are nonstochastic and assumed

to be known.
Adding concomitant variables to the latent class model leads to the possibility of testing hypothesis concerning the relationship between such variables and the classes, even finding a solution for the issue concerning an inadequate number of degrees of freedom (Dayton and Macready, 1988). The CVLCM presents two main components: the measurement model that focuses on how the latent classes are related to the manifest variables, and the structural model for the relationship between classes and concomitant variables (Bakk and Kuha, 2018).
In the basic LCM, each observation is initially assumed to have the same probability of being assigned to any latent class before considering the manifest variables. In contrast, the CVLCM approach allows these probabilities to differ for each observation, expressing them as function of the concomitant variables.
Consider a vector-valued observation $d_i = (y_i, x_i)$, with $y_i = \{y_{ij}\}$ as in Subsection 1.1.3 for the ith unit, where each x_i is a scalar-valued realization of the concomitant variable X considered, defined on the interval (a, b). Then, considering L latent classes, an L-class CVLCM is defined, conditioning the given multivariate response vector Y to the concomitant variable X, as:

$$Pr(Y \mid X) = \sum_{l=1}^{L} \Theta_{l|x} Pr(Y \mid c_l). \tag{1.2.1}$$

Here, $\Theta_{l|x}$ represents the conditional probability of an observation belonging to the lth latent class given the concomitant variable taking value equal to x.
In the model considered,

$$\Theta_{l|x} = g(X; \beta_l) \text{ with } \beta_l = \{\beta_{lk}\}, k = 0, \ldots, K-1 \tag{1.2.2}$$

indicating that the conditional probability of belonging to the lth latent class, given the concomitant variables X, is modeled by a function g. This function g is parameterized by the vector β_l, which contains the parameters specific to the lth latent class, and relates the concomitant variables to the probability of class membership. This means that the model uses the values of the concomitant variable X and the parameters β_l specific to each latent class to compute the probability that an observation with the concomitant variable $X = x$ belongs to the latent class l.
Since the range of $\Theta_{l|x}$ is $(0, 1)$ and the one of X is, theoretically, $(-\infty, \infty)$, the function $g(X; \beta_l,)$ must appropriately transform the unbounded real values of X to the bounded interval $[0, 1]$. While the choice of function g is flexible, it must ensure

that all real-valued inputs from X correspond to valid probabilities. (Dayton and Macready, 1988). Furthermore, there is a risk that some choices of the function could lead to nonidentified models, where multiple sets of parameter values give the same probability distribution for the observed data, making it impossible to distinguish them.

Additionally, $Pr(Y \mid c_l)$ is defined, as in the LCM, by 1.1.1, with restrictions on the probabilities of responses to the manifest variables in each latent class as in 1.1.4. Finally, for all possible values of X,

$$\sum_{l=1}^{L} \Theta_l = 1. \tag{1.2.3}$$

Having stated that the conditioning is done for Y on the concomitant variable X, which takes values in the interval (a, b), leads to:

$$Pr(Y) = \int_a^b Pr(Y \mid X) dX. \tag{1.2.4}$$

The parameters of the model are estimated through maximum likelihood methods. The likelihood is constructed based on the assumption that, given X, the probability of Y is the same for all cases with the same d_i. Thus, the likelihood for a given observation is a function of the observed patterns of responses and the concomitant variable X, with the assumption that the total probability across all latent classes equals one.

Focusing on Equation 1.2.1 exhibits how identical values of d_i lead to the same conditional probabilities $Pr(Y \mid X)$. As a result of this, the sufficient statistics is going to be the set of UV frequencies. U indexes the different levels or categories of the concomitant variable X, V refers to the unique response patterns observed in the manifest variables at each level $u = 1, \ldots, U$ of the concomitant variable. V is essentially the count of distinct ways in which the observed data can occur, corresponding to the uth level of X. This allows for the computation of probabilities for each combination of the concomitant and manifest variables.

For a dataset comprising N observations, the likelihood can be expressed as the product of probabilities associated with each observation's response given their concomitant variable data. This can be formally denoted by:

$$
\begin{aligned}
\mathcal{L} &= \prod_{i=1}^{N} Pr(y_i \mid x_i) \\
&= \prod_{u=1}^{U} \prod_{v=1}^{V} Pr(y_{uv} \mid x_u)^{N_{uv}} \\
&= \prod_{u=1}^{U} \prod_{v=1}^{V} \left[\sum_{l=1}^{L} \Theta_{l_u} \prod_{j=1}^{J} \prod_{r=1}^{R_j} [\alpha_{jrl}]^{\delta_{uvj}} \right]^{N_{uv}},
\end{aligned}
\tag{1.2.5}
$$

where $Pr(y_i \mid x_i)$ is the probability of observing the response pattern y_i given the concomitant variable x_i. The further decompositions consider the response patterns across different levels of the concomitant variable. In this representation, $y_{uv} = \{y_{uvj}\}$ symbolizes the set of V distinct response patterns at the uth level of the concomitant variable X, and N_{uv} denotes the count of cases for the uth response pattern at the given level of X. The Kronecker delta function is defined as:

$$
\delta_{uvj} = \begin{cases} 1 & \text{iff } y_{uvj} = r \\ 0 & \text{otherwise.} \end{cases}
\tag{1.2.6}
$$

The probability for a response pattern v at a particular level u of X within a latent class l is given by:

$$
a_{uvl} = \prod_{j=1}^{J} \prod_{r=1}^{R_j} [\alpha_{jrl}]^{\delta_{uvj}}.
\tag{1.2.7}
$$

The log-likelihood of the data, given the model parameters, can be simplified and expressed as:

$$
\begin{aligned}
\ln(\mathcal{L}) &= \sum_{u=1}^{U} \sum_{v=1}^{V} N_{uv} \ln \left[\sum_{l=1}^{L} \Theta_{l_u} \prod_{j=1}^{J} \prod_{r=1}^{R_j} [\alpha_{jrl}]^{\delta_{uvj}} \right] \\
&= \sum_{u=1}^{U} \sum_{v=1}^{V} N_{uv} \ln \left[\sum_{l=1}^{L} g(X_u, \beta_l) a_{uvl} \right].
\end{aligned}
\tag{1.2.8}
$$

The essence of the likelihood function in the context of latent-class analysis is to compute the product of probabilities for each unique combination of the manifest variables and the concomitant variable levels. This compound probability is raised to the power corresponding to the frequency of each unique response pattern. The

optimization goal is to maximize this likelihood (or equivalently, its logarithm), given the constraints on the parameters, which ensure that the probabilities sum to one and that the model parameters are identifiable.

Chapter 2

Adding Flexibility with B-splines

2.1 History of B-splines

Basis splines, better known as B-splines, constitute a fundamental concept in mathematics and statistics: they offer a versatile framework, able to represent curves and surfaces through piecewise polynomial functions. Splines were introduced by Isaac Jacob Schoenberg in the 1940s. The term "spline" was inspired by wooden flexible bars used to draw smooth curves (Schoenberg, 1946).
With Schoenberg originally listing ballistics, actuarial mathematics, and electronic computing machines as plausible fields to implement them, splines have transcended their original application (Schoenberg, 1946), now finding utility in statistical modeling, computer graphics, image processing, and financial mathematics.
Schoenberg introduced B-splines as a set of basis functions for crafting smooth and adaptable curves. B-splines enable precise and flexible modeling, representing functions accurately as a sum of basis functions, each weighted by specific coefficients.
The practical computation of B-splines was significantly improved by Carl de Boor in the early 1970s. De Boor's algorithm for efficiently evaluating B-spline functions made it possible to use B-splines in a wide range of applications, introducing a recursive formula for B-spline computation (de Boor, 1972).

2.2 Definition of B-splines

B-splines of order M are defined over a partition of the domain into intervals, separated by points known as knots. These knots determine where the pieces of piece-

wise polynomials of order M meet, influencing the shape and continuity of the resulting spline function. The B-spline function is constructed from a linear combination of B-spline basis functions of the same degree, which are defined recursively using the de Boor recursion formula (de Boor, 2001).
In order to define B-splines, it is necessary to introduce the knot vector. Assume that $\xi_0 < \xi_1$ and $\xi_T < \xi_{T+1}$ are boundary knots. Then, the augmented knot sequence τ is defined as follows:

- There are T internal knots $\xi_1 \dots \xi_T$, plus 2 boundary knots ξ_0 and ξ_{T+1}, and additionally, $2(M+1)$ external knots.
- The first $M+1$ elements of τ are the external knots such that $\tau_1 \leq \cdots \leq \tau_{M+1}$.
- The elements of τ from $M+2$ to $M+T+2$ correspond to the ξ knots, where it is possible to write $\tau_{M+j} = \xi_j$ for $j = 1, \dots, T$.
- The remaining $M+1$ last external knots are denoted as $\tau_{M+T+2} \leq \dots \leq \tau_{T+2M+2}$, ensuring there are a total of $T+2M+2$ knots.

The selection of knots beyond the boundaries may be arbitrary, but it is standard to set them equal to the boundary knots ξ_0 and ξ_{T+1}, respectively.
Consider $B_{i,m}(x)$ as the ith B-spline basis function of order m corresponding to the knot vector τ, where $0 \leq m \leq M$. These basis functions are defined recursively with the de Boor formula.
For the initial case, with the basis functions of the order zero:

$$B_{i,0}(x) = \begin{cases} 1 & \text{if } \tau_i \leq x < \tau_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{2.2.1}$$

for $i = 1, \dots, T+2M+1$.
For the recursive definition, where m is the order of the basis function:

$$B_{i,m}(x) = \frac{x - \tau_i}{\tau_{i+m-1} - \tau_i} B_{i,m-1}(x) + \frac{\tau_{i+m} - x}{\tau_{i+m} - \tau_{i+1}} B_{i+1,m-1}(x) \tag{2.2.2}$$

for $i = 1, \dots, T+2M+1-m$.
In order to avoid the issue of division by zero in these formulas, the convention is to set $B_{i,0} = 0$ if $\tau_i = \tau_{i+1}$, ensuring that the function is well-defined (Hastie et al., 2009).

2.2.1 B-splines in Statistical Modeling

B-splines find great application in non-parametric regression and smoothing techniques. By representing nonlinear relationships between variables through splines, it is possible to model data without having to assume any parametric form. This helps to capture subtle patterns and enhances the interpretability of the model.
In generalized additive models (GAMs), for example, B-splines are employed to estimate smooth functions of predictors, enabling the exploration of non-linear effects within an additive framework (Hastie and Tibshirani, 1990).
The flexibility and the fit to the data is affected by the choice of the number and the positioning of the knots. Selecting a number of knots too low may in fact lead to underfitting, while too large could make the model overfit. An optimal placement of the knots aims at ensuring the best possible fit. The positioning can be made at regular intervals (as selected in the simulation study of Chapter 4), quantile-based (as chosen in the Brexit Data Analysis of Chapter 5), or based on specific algorithms.
When comparing B-splines to direct polynomial representations, there are several advantages that make the former the better choice. Firstly, the modification of a control point affects only a local portion of the curve, enabling precise manipulation without altering the entire structure. Secondly, B-splines exhibit superior numerical stability in computation, particularly for high-degree polynomials. Thirdly, the possibility of choosing the knot vector in B-splines allows for adjusting the degree of continuity at different points, offering flexibility when designing the model. Lastly, B-splines enable efficient representation and evaluation of complex curves and surfaces with lower computational expense compared to other spline forms (Hastie et al., 2009).

2.2.2 P-splines

When fitting curves, it is crucial to balance the fit to the data with the smoothness of the curve to avoid overfitting. One approach to achieving this balance is through the use of P-splines, which stand for “penalized B-splines”. Introduced by Eilers and Marx, P-splines enhance the flexibility of B-splines by incorporating a penalty term that discourages excessive roughness in the fit (Eilers and Marx, 1996).
The penalization is implemented through the addition of a smoothness penalty to the least squares error function, which is traditionally used to estimate the spline coefficients. Considering the regression of n data points (x_i, y_i) on a set of K B-

splines $B_j()$, this can be expressed as:

$$L(\beta) = \sum_{i=1}^{n}(y_i - \sum_{j=1}^{K}\beta_j B_j(x_i))^2 + \lambda \int \sum_{j=1}^{K}\beta_j [B_j^{''}(x)]^2 dx, \tag{2.2.3}$$

where $L(\beta)$ represents the penalized loss function, β_j, $j = 1, \ldots K$, are the coefficients to be estimated for each spline, λ is the penalty term controlling the trade-off between smoothness and fit, and $B_j^{''}(x)$ is the second derivative of the jth spline function, indicating the level of smoothness to be imposed on the curve. As the penalty term increases, the smoothness of the spline increases, often at the expense of fitting the data closely, thus effectively managing the trade-off between overfitting and underfitting.

It is worth noting that the penalization based on the square of the second derivative can be interpreted as a measure of deviation from linearity, since it assumes a null value if and only if the function being differentiated is linear. Additionally, this measure does not change if a constant or a linear term is added to the function. Nevertheless, there is not a specific reason for choosing the second derivative: with smoothing splines, the first derivative leads to simple equations and piecewise linear fits, while the choice of higher derivatives may lead to complex equations (Eilers and Marx, 1996).

Chapter 3

Estimation Methods

3.1 The Simplex Algorithm in Latent Class Analysis

When introducing the Concomitant Variable Latent Class Model, Dayton and Macready advocated for the usage of the Simplex algorithm. It operates over an M-dimensional parameter space, utilizing a simplex (a figure with M+1 apexes) to navigate the space systematically. The movement rules within this space are crucial, involving the calculation of the objective function F for each apex, locating the best and worst apexes, and executing vertex operations depending on the nature of F. The algorithm iteratively moves towards the extremum of F, adjusting apexes by reflecting, expanding, or contracting to find better parameter estimates (Dayton and Macready, 1988).
The simplex algorithm was favored due to its ease of implementation, as it only requires the objective function and not its derivatives. Still, the algorithm needs precise and multiple starting values to avoid premature convergence and the choosing of local maxima. In summary, the simplex algorithm's ability to navigate parameter spaces efficiently and its adaptability to various constraints make it a suitable choice for multiparameter estimation problems in latent class analysis, as highlighted by Dayton and Macready.

3.2 1-Step Method

Focusing on the dual nature inherent in Concomitant Variable Latent Class Model is crucial for understanding the different ways of estimating the model. This duality is manifested in its division into a measurement model and a structural model, fa-

cilitating the exploration of latent constructs through observable variables and their subsequent interaction with external covariates.
The measurement model addresses the delineation of latent classes from the manifest variables, with the premise that the latter are manifestations of underlying constructs. The measurement model seeks to establish a probabilistic relationship between observable data patterns and latent class membership. This involves constructing a statistical framework where the distribution of observed variables, conditioned on latent class membership, is formulated. The selection of an appropriate distribution is based on the nature of the observed indicators.
Conversely, the structural model delves into explaining the relationship and interaction between the latent classes and concomitant variables.
The integration of the measurement and structural models in LCM poses a methodological challenge, with the need of heavy focus on their interaction. This interdependence unravels the importance of a methodologically sound approach in estimating these models. Today, the academic discourse revolves around three predominant estimation methodologies: the one-step, two-step, and three-step methods (Bakk and Kuha, 2018). Each method presents a unique approach to deal with the interaction between Measurement and Structural Models, influencing the interpretation.
In the 1-Step estimation method, both the measurement and structural components of the model are simultaneously considered to derive maximum likelihood estimates for all parameters. This method, often referred to as full-information maximum likelihood (FIML) estimation, is known for its efficiency and comprehensive approach. The expression of the Likelihood and its logarithm are given in 1.2.5 and 1.2.8. The objective of this estimation procedure is to maximize either $\mathcal{L}$ or $ln(\mathcal{L})$.
The partial derivatives of the log-likelihood function with respect to the parameters α_{jrl} and β_l are easily calculated, but they are intrinsically nonlinear. For this reason, the FIML estimation is iterative (Dayton and Macready, 1988), with the Newton-Raphson method and EM algorithm often employed. In applications utilizing the EM algorithm for deriving ML estimates of parameters, it is important to note that estimated probabilities cannot fall below zero or exceed one. When employing iterative processes, if an estimated probability hits these boundaries, its value remains unchanged in subsequent iterations. Consequently, the solution achieved for the entire set of parameters should be interpreted as a terminal solution or a conditional maximum likelihood solution. This interpretation is based on the assumption that these boundary estimates accurately represent the true population values, in the same way that happens when such constraints are predetermined (Hagenaars,

1993).
Following the notation used previously, the Measurement Model (MM) and the Structural Model (SM) are defined as follows.
For any realization of the vector-valued $d_i = (y_i, x_i)$, the MM is defined as the joint distribution of the manifest variables and the latent classes:

$$\text{MM: } Pr(Y_i, c_l; \Theta, \alpha) = Pr(c_l; \Theta_l) Pr(Y_i \mid c_l; \alpha_l) = \Theta_l \prod_{j=1}^{J} \prod_{r=1}^{R_j} [\alpha_{jrl}]^{\delta_{ij}}. \quad (3.2.1)$$

On the other hand, the SM represents the joint distribution of the latent classes and the concomitant variable, and can be expressed as:

$$\text{SM: } Pr(c_l, X; \beta_l) = Pr(X) Pr(c_l \mid X; \beta_l) = Pr(X) g(X; \beta_l) \quad (3.2.2)$$

,

where:

- Y_i represents the observed indicators.
- c_l represents the latent classes.
- X corresponds to the concomitant variable.
- Θ_l denotes the parameters for the latent class probabilities.
- α_{jrl} denotes the parameters for the probability of response r for variable j in latent class l.
- β_l denotes the parameters defining the impact of the concomitant variable on the probability associated with the lth latent class.

In the composite framework of the CVLCM, the Measurement Model and the Structural Model are integrated to form a Complete Model (CM). The CM delineates the probability distributions of observed variables and latent classes, incorporating concomitant variables. Formally, the model can be expressed as:

$$\begin{aligned} Pr(Y_i, c_l, X; \Theta, \alpha, \beta) &= Pr(X) Pr(Y_i, c_l \mid X; \Theta, \alpha, \beta_l) \\ &= Pr(X) Pr(c_l \mid X; \beta_l) Pr(Y_i \mid c_l; \alpha). \end{aligned} \quad (3.2.3)$$

When covariates are not present, the model simplifies, as the influence of concomitant variables is removed. The parameters specifying the distribution of latent

classes are excluded from the conditional probability calculations. This implies that the observed data is generated solely based on the parameters governing the relationship between the observed indicators and the latent classes.
In essence, the presence of covariates modifies the probability structure of the model, leading to a conditional estimation of the latent class distribution (Morin et al., 2023), and the model becoming defined for $P(Y_i \mid X_i)$ rather than for $(P(Y_i)$.
To clarify, when covariates are accounted for, the CM provides a conditional probability that is contingent on the covariate values. When covariates are not considered, the model reverts to a marginal probability that is independent of the concomitant variables. Therefore, the conditional distribution of latent classes within the CM can be characterized by the following conditional probabilities:

$$Pr(Y_i, c_l \mid X; \Theta, \alpha, \beta) = \begin{cases} Pr(Y_i, c_l; \Theta, \alpha) & \text{if } X = \emptyset, \\ Pr(c_l|X; \beta_l) \cdot Pr(Y_i|c_l, X; \Theta, \alpha) & \text{otherwise.} \end{cases} \tag{3.2.4}$$

This dual structure facilitates a flexible representation of the model, allowing for scenarios where the distribution of observed indicators Y_i is either considered independent of concomitant variables X, or it is conditioned on X when such covariates are included.
The one-step estimation based on the FIML approach for latent class analysis with covariates can be represented using the following notation. The probability of observing the set of responses Y_i, given the concomitant variable X is obtained by summing over the latent classes the product of the probability of the observation i belonging to latent class c_l, given their values on the concomitant variable X, by the conditional probability of observing the response pattern Y_i, knowing that the observation belongs to the latent class c_l:

$$P(Y_i \mid X) = \sum_{l=1}^{L} P(c_l \mid X; \beta_l) P(Y_i \mid c_l; \alpha_l). \tag{3.2.5}$$

It is assumed that Y_i is independent of X conditional on c_l (Vermunt, 2010). Following the ideas introduced in the previous chapter, in this Thesis the probability of belonging to a latent class given the concomitant variable is modeled using B-splines as:

$$P(c_l \mid X; \beta_l) = \frac{\exp\left(\sum_{k=1}^{K} \beta_{lk} B_k(X)\right)}{\sum_{l=1}^{L} \exp\left(\sum_{k=1}^{K} \beta_{lk} B_k(X)\right)}. \tag{3.2.6}$$

The Full Information Maximum Likelihood estimates of the B-splines and multinomial parameters are obtained by maximizing the loglikelihood based on the conditional probability of $Pr(Y_i \mid X_i, \beta_l)$,

$$\log \mathcal{L}_{FIML} = \sum_{i=1}^{N} \log \left[\sum_{l=1}^{L} P(c_l \mid X_i; \beta_l) P(Y_i \mid c_l; \alpha_l) \right]. \tag{3.2.7}$$

The maximization of the log-likelihood is performed by applying an Expectation-Maximization algorithm (Bandeen-Roche et al., 1997).

The one-step approach, while effective, presents several drawbacks (Vermunt, 2010). Firstly, it can be challenging, particularly in exploratory studies where numerous potential covariates are involved. Each addition or removal of a covariate necessitates re-estimation not just of the structural model but also of the measurement model, consequently affecting the interpretation and comparability of the analysis results. This can be a time-consuming and complex process that suffers of computational inefficiency. In practical scenarios, this may undermine the meaningful comparison of structural model estimations.

Secondly, this method introduces complexity in the model construction, such as deciding on the number of classes in models with or without covariates. In CVLCM, the addition of covariates can influence the class structure itself, raising questions about the stability and interpretability of the class definitions.

Thirdly, this approach often contradicts the typical methodology that considers the integration of covariates as a subsequent step following the development of the classification model. This sequential approach allows for a clearer understanding of the underlying Measurement model before examining the influence of external variables. The one-step approach, which integrates these steps, can be counterintuitive to this traditional workflow, potentially leading to confusion or misinterpretation of results.

Fourthly, it presupposes that the development of the Measurement model and the Structural model predicting class membership occur simultaneously within a study, which may not always be the case (Vermunt, 2010).

Lastly, when estimating the entire model through a single step, inaccuracies in cer-

tain sections of the model can adversely affect the precision of parameter estimations in other segments, even if these sections are correctly specified (G. A. Marcoulides, 2002).

3.3 3-Step Method

In numerous cases, adopting a Stepwise methodology is not only more intuitive but sometimes represents the only viable strategy. The conventional methodology is composed of three phases.
The first step develops a latent class model for a set of manifest variables. This step requires deciding on the selection of manifest variables, the number of latent classes for the classification model, and addressing model specification details like the distribution of the manifest variables within classes. The model for the probability of Y_i given the latent class c_l is defined as:

$$P(Y_i) = \sum_{l=1}^{L} P(c_l) P(Y_i \mid c_l) \tag{3.3.1}$$

With the assumption of independence of the manifest variables within each class, the probability of the response vector Y_i given the latent class c_l is:

$$P(Y_i \mid c_l) = \prod_{j=1}^{J} P(Y_{ij} \mid c_l) = \prod_{j=1}^{J} \prod_{r=1}^{R_j} \alpha_{jrl}^{\delta_{ijr}}, \tag{3.3.2}$$

where δ_{ijr} equals 1 if the observation i takes response r on the manifest variable j and 0 otherwise. The parameters that need to be estimated are the proportions $Pr(c_l)$ associated with each class, and α_{jrl}, the probability of each manifest variable j to assume a specific value r, given its class l.
The estimation of these parameters is conducted through the maximization of the following log-likelihood function:

$$\ln \mathcal{L} = \sum_{i=1}^{N} \log Pr(Y_i) = \sum_{i=1}^{N} \log \left[\sum_{l=1}^{L} P(c_l) \prod_{j=1}^{J} \prod_{r=1}^{R_j} \alpha_{jrl}^{\delta_{ijr}} \right], \tag{3.3.3}$$

In the second step, observations are allocated to the latent classes, based on their posterior class membership probabilities, derived from their observed responses and

the parameters estimated in the first phase. The assignment of observation i to a class is represented by w_i, with the critical factor for class assignment being the posterior class membership probability $Pr(c_l \mid Y_i)$, which can be calculated using Bayes' theorem (Vermunt, 2010):

$$P(c_l \mid Y_i) = \frac{P(c_l)P(Y_i \mid c_l)}{P(Y_i)} \tag{3.3.4}$$

Classification rules most frequently employed are modal and proportional assignment. According to the modal assignment, w_i is chosen as the value of l that maximizes the posterior probability just defined. The proportional assignment gives a fractional classification with weights $w_{il} = P(c_l \mid Y_i)$, offering a soft classification for each subject over all classes. This method acknowledges that observations may not fit perfectly into a single class and allows for a degree of uncertainty in class membership.

In the third step, a multinomial logistic regression model with the class assignments from the second phase is implemented to study the effect of concomitant variables on the estimated class membership (Vermunt, 2010).

$$P(c_l \mid X_i; \beta_l) = \frac{\exp\left(\sum_{k=1}^{K} \beta_{lk} B_k(X)\right)}{\sum_{l'=1}^{L} \exp\left(\sum_{k=1}^{K} \beta_{l'k} B_k(X)\right)} \tag{3.3.5}$$

The parameters of interest are β_{lk}, where l indexes the latent classes and k indexes the B-spline basis functions. These coefficients determine the influence of the B-spline basis functions for the covariate on the log-odds of membership in the latent class c_l. In particular, they show how the probability of an observation belonging to a particular latent class is affected by the value of the concomitant variable. These parameters are obtained by maximizing the following weighted log-likelihood function:

$$\log \mathcal{L}_{\text{Step3}} = \sum_{i=1}^{N} \sum_{l=1}^{L} w_{il} \log P(c_l \mid X_i; \beta_l), \tag{3.3.6}$$

This involves performing a multinomial logistic regression analysis that uses L records for each observation, except when using a modal assignment: here w_{il} will be 1 for the assigned class and 0 otherwise (Vermunt, 2010).

This approach offers greater control and specificity at each stage, potentially lead-

ing to a more nuanced classification and a deeper understanding of the relationships. However, it also introduces complexity in implementation and the potential for cumulative errors across the steps.

3.3.1 Bias in 3-Step Procedures

Dealing with the relationship between predicted and actual latent scores within any LCM composed of a measurement and structural part could be complex. The distribution of the predicted scores is linked to the true latent scores through an integral equation, indicating that the former cannot perfectly represent the latter. Also, the variety of methods available for calculating predicted latent scores makes it impossible to mathematically identify them (G. A. Marcoulides, 2002).
Having considered these drawbacks, the main concern of of 3-Step procedures is that the chosen method for computing predicted latent scores can introduce bias into the estimates of parameters for the relationships between latent and observed structural variables, and into their standard errors. This bias is still present even asymptotically (Bolck et al., 2004).
The classification rules proposed, either modal or deterministic, are both based on the posterior class membership probability $Pr(c_l|Y_i)$, and face uncertainty and potential misclassification. During the second stage of the 3-Step estimation method, it is possible to prove that $Pr(\hat{c}_l|Y_i)$, with $\hat{c}_l$ being the estimated class membership for unit i is not identical to $Pr(c_l|Y_i)$, obtained in the 1-Step estimation procedure. This leads to the fact that the estimates of the parameters in the structural model will be biased in the 3-Step approach. More specifically, a bigger classification error in the second steps will lead to a larger bias in the parameter estimates. For further readings on the topic, is it possible to consult Bolck et al. (2004).This paper outlines a method designed to correct the bias, referred to as the BCH method, named after its developers Bolck, Croon, and Hagenaars. Vermunt (Vermunt, 2010) attributes the method's development to these authors, who introduced it as a correction for the three-step approach in their work (Bolck et al., 2004). First, the BCH 3-Step method creates a multidimensional frequency table containing the information regarding covariates for the structural model and class allocations. Then, the frequency counts' weights from the table are recalculated through a process involving multiplication by the inverse of the classification error matrix. Finally, a logistic regression model is developed, treating the reweighted frequency table as if it were the actual observed data.

However, several issues arise with this approach. First of all, there is a requirement for covariates to be in a categorical format for effective inclusion in a frequency table. Secondly, the need for complex matrix multiplications during the preparation of data becomes particularly laborious as it must be repeated each time a new set of covariates is introduced. Lastly, the logistic regression model significantly underestimates standard errors (Vermunt, 2010).
The resolution of the three outlined challenges can be achieved by formulating a pseudo log-likelihood that accounts for individual data points as opposed to weighted covariate patterns. This methodology takes the name of Extended BCH. Vermunt has also introduced an alternative for correcting the 3-Step estimation method, based on maximum likelihood. Including the concomitant variables in a LCM where $Pr(\hat{c_{l'}} \mid c_l)$ are treated as known error probabilities will lead to precise estimations of the covariate effects. This method yields maximum likelihood estimates for $Pr(c_l \mid X_i)$ and for the B-spline coefficients β_{lk}. Both methodologies introduced by Vermunt may have standard errors that are slightly underestimated since the classification error probabilities are considered as known, when they can only be calculated thanks to the LCM estimations obtained without considering the concomitant variable. The maximum likelihood method boasts a significant benefit over the BCH method in terms of efficiency, almost comparable to the 1-Step method. A drawback of these corrections is the case when class distinctions are extremely vague; here, the estimated parameters and the standard errors are not reliable. (Vermunt, 2010). For this reason, the analyses in the following chapters will be based on the standard estimation in the 3-Step case.

3.4 2-Step Method

Beyond the BCH and ML corrections for the 3-Step estimation technique, a 2-Step method has been proposed by Bakk and Kuha (Bakk and Kuha, 2018) to overcome potential bias in the structural component' estimates.
The first step mirrors the 3-Step approach, focusing solely on fitting the latent class measurement model. The second step, however, involves maximizing the joint likelihood with the parameters of the measurement model fixed at their first-step estimates. This means that the second step now focuses only on the estimation of the structural model parameters.
The authors' reasoning behind the methodology stems from the understanding that the key element of a Stepwise approach lies in the independent estimation of the

measurement model, rather than the necessity for a distinct classification phase, especially since the bias in the 3-Step estimation is introduced in the second stage. By omitting this stage, the need for correcting the bias is also omitted. For this reason, the 2-Step approach is more direct than the adjusted versions of the 3-Step method. According to simulation studies conducted by the authors, the structural model estimates are consistent, positioning second in terms of efficiency after the 1-Step estimates, but with better performance than the bias-adjusted 3-Step ones (Bakk and Kuha, 2018).

In the first step, the parameters that are going to be estimated are the ones related to the measurement model, that have been denoted as α_{jrl} in the previous paragraphs. The second step will focus on the structural model' parameters β_{lk}.

The first step is still based on the maximization of the log-likelihood function of a latent class model without concomitant variables:

$$\mathcal{L} = \sum_{i=1}^{N} \log Pr(Y_i) = \sum_{i=1}^{N} \log \left[\sum_{l=1}^{L} P(c_l) \prod_{j=1}^{J} \prod_{r=1}^{R_j} \alpha_{jrl}^{\delta_{ijr}} \right]. \tag{3.4.1}$$

The estimation of parameters from the first phase, denoted as $\hat{\alpha}_{jrl}$, sets the stage for the second phase's application of the log likelihood $\ln \mathcal{L}_2(\hat{\alpha}_{jrl}, \beta_{lk})$, which is the summation for all the observations of the logarithm of the probabilities

$$\sum_{i=1}^{N} Pr(Y_i \mid X_i) \tag{3.4.2}$$

which is a function of β_{lk} and assessed at $\alpha_{jrl} = \hat{\alpha}_{jrl}$. The maximization of this sum, with respect to β_{lk} yields the estimates for the second step parameters, denoted by $\hat{\beta}_{lk}$.

The 2-Step estimation procedure maintains the Stepwise idea, since it estimates the structural model parameters by fixing the measurement ones. A modification to the structural model would not lead to changes for $\hat{\alpha}_{jrl}$, since only the second step would need to be repeated. This complete separation between the two steps also enables the chance of employing two different samples, useful when dealing with missing data, or with resource constraints (Bakk and Kuha, 2018).

3.4.1 Properties of 2-Step Estimators

Dividing the estimation into two phases leads to the measurement model estimators being consistent, and to then considering these estimates as know when dealing with the structural model parameters. The possible misspecification of the α_{jrl}, leads to the maximization of the loglikelihood in terms of pseudo-maximum-likelihood (PML). The estimators will have the same properties as the ones obtained from PML: they are consistent and asymptotically normal under general regularity conditions. These conditions are satisfied since the first step estimates are obtained simply by maximization of the likelihood, and because the two steps are independent.

Attention is directed towards the Fisher information matrix relevant to the model parameters $\phi = (\alpha, \beta)$:

$$I(\phi^*) = \begin{bmatrix} I_{11} & I_{12} \\ I_{12}^T & I_{22} \end{bmatrix}, \tag{3.4.3}$$

with ϕ^* representing the true value of ϕ, and the partition of the matrix being based on the parameters being estimated in the first and in the second step. Let's assume that the estimation in the first step is based on m observation, while the second step on n.

Similarly to what happens in the case of 1-Step estimations, the first step's estimates of the proposed 2-Step are obtained thanks to maximum likelihood. Let Σ_{11}/m be the asymptotic variance matrix of the first step estimator $\hat{\alpha}$, evaluated at the true values of the parameters (α, β), and estimated by substituting Step 1 estimates for these parameters.

The asymptotic variance matrix of the second step estimator β is then $\frac{V}{n}$, with V being:

$$V = I_{22}^{-1} + I_{22}^{-1} I_{12}^T \left(\frac{n}{m} \Sigma_{11} \right) I_{12} I_{22}^{-1} = V_2 + V_1. \tag{3.4.4}$$

The adjustment factor (n/m) compensates for potential discrepancies in sample size across the two stages of the process. Within this framework, V_2 represents the variability in $\hat{\beta}$ under the hypothetical scenario where the parameters from step 1, denoted α, are known. Conversely, V_1 denotes the added variability that stems from the actual scenario where α are not directly observed but are inferred from $\hat{\alpha}$.

The variance V is approximated by $\hat{V} = \hat{V}_2 + \hat{V}_1$, this is achieved by replacing $\hat{\phi}$ for the true parameter values in the relevant components of the information matrix, specifically I_{22} and I_{12}. This approximation utilizes the n observations from the second step, and the earlier estimates from the first for Σ_{11}.

Chapter 4

Simulation Study

The following chapter contains a simulation study designed to investigate the behaviour of the three estimation methods for the Concomitant Variable Latent Class Model, when the log-odds of the latent class probabilities are modelled using B-splines, in order to allow a flexible representation of the concomitant variable effect. In particular, this simulation study will focus on the impact on the choice for the number of internal knots characterising the B-spline functions and on the selection of the number of latent classes. The simulation is executed through the use of the Python programming language.

4.1 Data Generation

The objective of the simulation study is to analyze data generated from a Concomitant Variable Latent Class Model (CVLCM) with two latent classes and incorporating ten binary variables. The data generation begins by defining a non-linear function f_2 that is involved in the definition of the probability distribution of group membership in the data. This group membership is determined using a multinomial process. The next step involves creating manifest variables based on the group membership. These manifest variables are subjected to a probability alteration mechanism, which is controlled by a parameter and a randomly determined binary switch. This methodology not only ensures the creation of diverse data sets but also mirrors the complexity and variability inherent in real-world data, providing a robust platform for testing and validating the CVLCM approach.
In the simulation framework, the generation of data begins with the initialization of

a seed to ensure the reproducibility of the analysis. The function f_2 is defined as:

$$f_2(x) = \sin(3\pi x) \cdot \exp(-x), \tag{4.1.1}$$

combining a sine wave with exponential decay. After this, a concomitant variable x is generated using a uniform distribution. Subsequently, the assignment of latent classes in *group* is based on the transformation $f_2(x)$. Probabilities for each latent class are computed, using a logistic transformation, and defined as:

$$\begin{aligned} Pr(c_2 \mid x) &= \frac{\exp(f_2(x))}{1 + \exp(f_2(x))} \\ Pr(c_1 \mid x) = 1 - Pr(c_2 \mid x) &= \frac{1}{1 + \exp(f_2(x))} \end{aligned} \tag{4.1.2}$$

The *rmulti* function then assigns each observation to a latent class based on these probabilities.
For the binary manifest variables, identified as *Y*, a binomial variable introduces variability in the conditional binary outcome probabilities, so that $\alpha_{ij1} = Pr(Y_{ij} = 1 \mid c_1) = \theta$ and $\alpha_{ij2} = Pr(Y_{ij} = 1 \mid c_2) = 1 - \theta$ or viceversa. The parameter θ controls the separation between the conditional distributions of the manifest variables. In the chapter, the focus will be on it being equal to 0.6, 0.7, 0.8, 0.9, with a higher value corresponding to a higher separation. Once each manifest variable's probability structure is determined, the variables are generated accordingly, taking into account the assigned latent class.
To conclude the generation process, a dataframe is created to contain the manifest variables, the concomitant variable, and the latent class information.

4.2 B-spline Generation

The generation of B-splines for the concomitant variable is based on the modification and creation of various Python functions. It begins with the *B* function, adapted from the existing implementation in the SciPy library (Virtanen et al., 2020). This function is modified to handle the special case when x is equal to 1. Such an adjustment is needed for assuring that only the last B-spline will be different than 0 and equal to 1. The *B* function essentially calculates the value of a B-Spline basis at a given point, considering its degree and position within the knot sequence.
Further, the *bspline_bases* function is crafted to return a list of B-spline bases in x.

This function iterates through the knot sequence, applying the B function to calculate the value of each B-spline basis. The positioning of the knots is based on a even spacing between 0 and 1, adding three external knots at 0 and three external knots at 1. This extension is crucial for ensuring the B-spline curve's smoothness at both extremities.
The B-splines are then added to the dataset by the function *add_bspline_columns*. This function computes the B-spline bases for each observation, and then merges these values into the original dataset.

4.3 Creation and Specification of the Models

The analysis conducted in this chapter was facilitated by the use of the Python package StepMix (Morin et al., 2023). StepMix is an open-source software designed for the pseudo-likelihood estimation of generalized finite mixture models, including latent class analysis with the addition of concomitant variables. The package permits the choice of different distributions for the measurement and the structural models. For the sake of the simulation study, a various number of models will be created by varying the estimation method, the separation between the conditional distributions of the variables, the sample size, the number of internal knots, and in some instances the number of latent classes. A constant in the analysis will be the measurement model, which will be composed of 10 manifest variables all having a Bernoulli distribution, and the structural model, where class probabilities will be evaluated by a multinomial logistic model of the concomitant variable.
The procedure will be repeated several times to ensure robustness and reliability. These repetitions are obtained by varying the starting seed, which plays a role in both data generation and model fit to the data. This practice overcomes the possible data sensitivity to random fluctuations and overfitting.

4.4 Results with a Fixed Number of Latent Classes

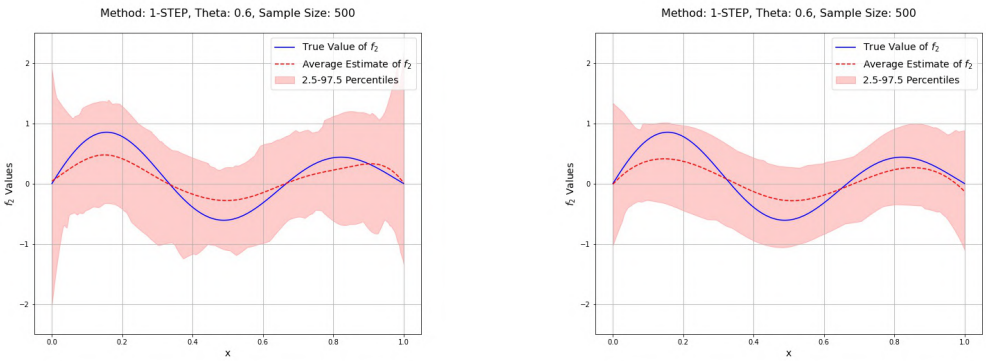
The first part of the simulation study will consider the number of latent classes as known and equal to two. For this reason, for each one of the 100 iterations performed, the parameters that are going to vary are the estimation method (three possible options), the separation between the conditional distributions of the manifest variables (four possible options), the sample size (three possible options), for a total

of 36 different scenarios. In each scenario, a number of internal knot ranging from one to seven has been considered.
The *evaluate_methods_integral* function is going to perform the analysis. It begins by creating a series of models, each differing in their parameters' configurations. For every combination of these and every iteration, it calculates the Root Mean Square Error (RMSE), the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC) for model selection. The AIC is then used to select the number of internal knots in each iteration that leads to a smaller information criteria. The function *integrate_second_derivative* is called to quantify the roughness of the estimated function $\hat{f}_2$ via its integrated squared second derivative, in order to keep track of the trade-off between the necessity of fitting the data closely and maintaining a smooth model.
After fitting each models to the simulated data, the models' coefficients are checked by the function *correct_coefficients* to ensure that label switching is not affecting the current iteration.
Consequently, for all possible combinations of estimation method, sample size and separation between conditional distributions of the manifest variables, the true f_2 is plotted against the average estimated function $\hat{f}_2$ and its 2.5% and 97.5% percentiles.

4.4.1 Comparative Analysis of f_2 and $\hat{f}_2$ Across Estimation Methods

After performing the analysis with 100 iterations, it is possible to examine how class separation, sample size and estimation methods affect $\hat{f}_2$. It is noteworthy that, independently from the possible parameters' choices, the 2.5% and 97.5% percentiles present unexpected spikes that clash with the expected smoothness. The underlying cause is the fact that each iteration loops over a different range of internal knots, going from one to seven. As can be observed in Appendix C.4, in the 100 models selected for each parameters' combination by picking the one with the smaller value of AIC, there is always variability in the number of internal knots.
Figure 4.1 illustrates the effect of selecting the number of internal knots on the true and estimated f_2 using the 1-Step estimation method with the class separation set very low (0.6) and with the sample size equal to 500. It becomes apparent that employing various values for the internal knots (in Figure 4.1a the selected numbers were different than one in 42 cases out of 100) leads to fluctuations in the per-

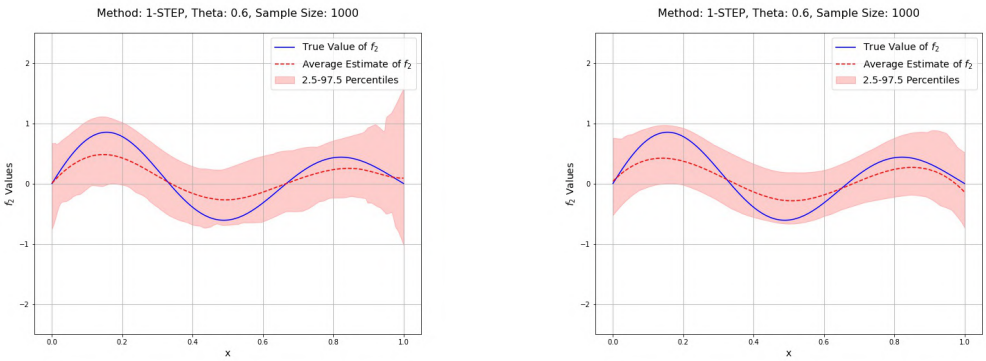
centiles. Conversely, this effect is not present when a constant number of internal knots is implemented. It is also discernible how a varying number of knots leads to larger percentiles, especially at the domain margins.



(a) (b)

Figure 4.1: Comparison of True vs. Estimated f_2 using 1-Step with $\theta = 0.6$ and Sample Size = 500: impact of the number of internal knots (selected according to AIC vs fixed at 1)

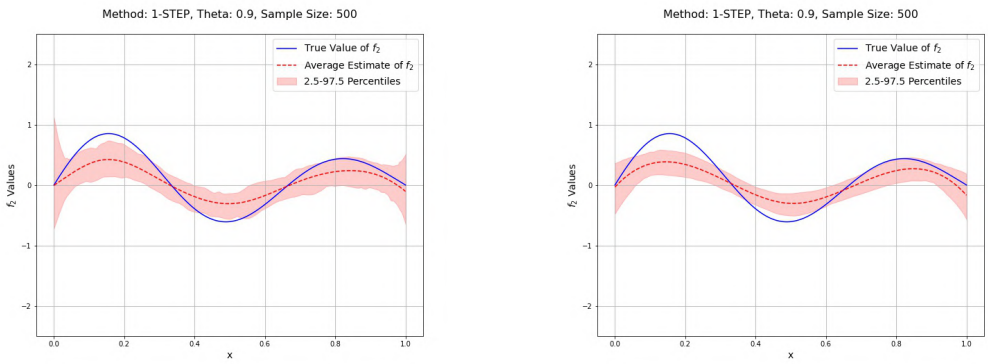
The following case, portrayed in Figure 4.2, depicts the effect of selecting the number of internal knots on the true and estimated f_2 using the 1-Step estimation method with the same class separation (0.6), but with the sample size equal to 1000. The amount of spikes in the percentiles is greatly reduced. This can be attributed to the higher sample size, that leads to a more accurate calculation of the 2.5% and 97.5% percentiles.



(a) (b)

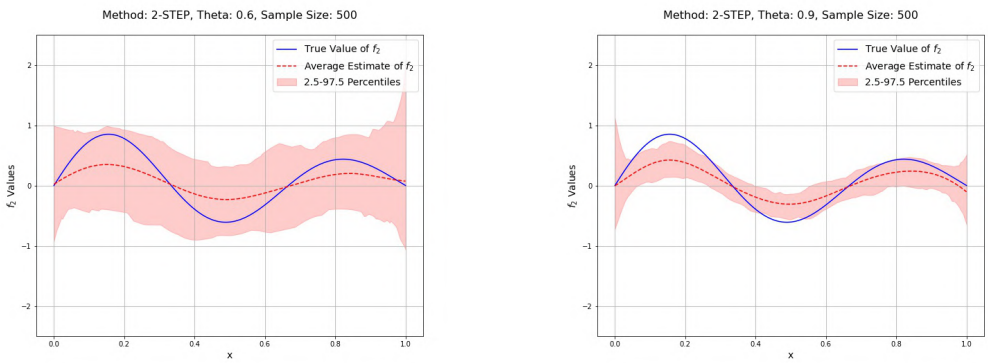
Figure 4.2: Comparison of True vs. Estimated f_2 using 1-Step with $\theta = 0.6$ and Sample Size = 1000: impact of the number of internal knots (selected according to AIC vs fixed at 1)

Continuing the analysis and referring to Figure 4.3, the focus is on the 1-Step technique with $\theta = 0.9$ and a sample size of 500. The comparison is based on $\hat{f}_2$ being estimated with an optimal number of knots against a scenario with a single knot. While the amount of bumps remains substantial, a higher class separation is observed to reduce the width of the pointwise percentile intervals, particularly when contrasted with the observations in Figure 4.1a.



(a) (b)

Figure 4.3: Comparison of True vs. Estimated f_2 using 1-Step with $\theta = 0.9$ and Sample Size = 500: impact of the number of internal knots (selected according to AIC vs fixed at 1)

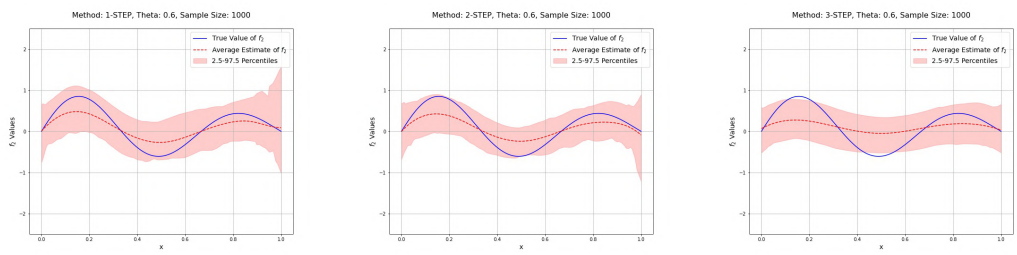


(a) (b)

Figure 4.4: Comparison of True vs. Estimated f_2 using 2-Step and Sample Size = 1000: Variability with θ set to 0.6 and 0.9

The analysis now exhibits a comparison aiming at evaluating the effect of a higher class separation, while keeping the sample size and estimation method constant. With the 2-Step estimation procedure and sample size set to 500 in both

Figure 4.4a and Figure 4.4b, the difference is based on the group separation, that goes from very low (0.6) to high (0.9). The effect of this is clear, with the percentiles closely aligning to $\hat{f}_2$ when θ is bigger. The presence of bumps is greatly reduced. The next analysis aims at drawing a comparison among the three estimation procedures. Figure 4.5 includes the results of the analyses with class separation equal to 0.6 and sample size set to 1000. The 1-Step pointwise percentile intervals appear to be larger than the ones based on the 2 and 3-Step methods. These two are similar, with the 3-Step having slightly narrower intervals.

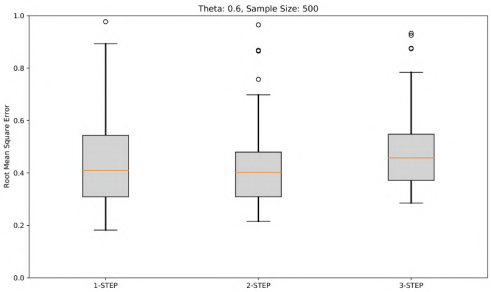


(a) (b) (c)

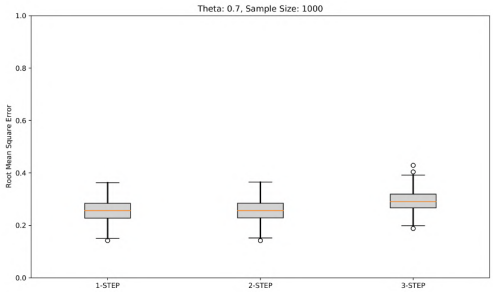
Figure 4.5: Comparative analysis of True vs. Estimated f_2 using different step methods with $\theta = 0.6$ and Sample Size = 1000

4.4.2 Comparative Analysis of Root Mean Square Errors Across Estimation Methods

A more quantitative comparison between the estimated and the true function can be based on the root mean square errors (see Appendix C.3 for a detailed report of the results obtained in each scenario). Generally speaking, RMSE tends to show larger values when the class separation is poor ($\theta = 0.6$ or $\theta = 0.7$). It is worth noting that with a small sample size the 2-Step method seems to achieve a slightly better performance (Figure 4.6a). Differences between estimation procedures seems to vanish as the sample size incresases, even though the 3-Step procedure tends to be characterised by slightly larger values for RMSE (Figure 4.6b).



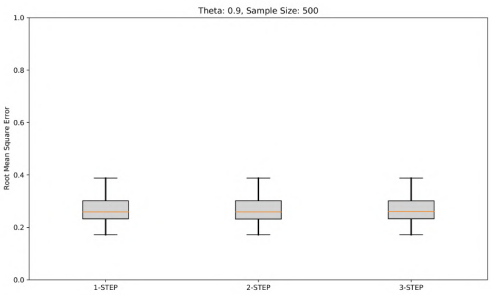
(a) RMSE across Estimation Procedures with $\theta = 0.6$, Sample Size = 500



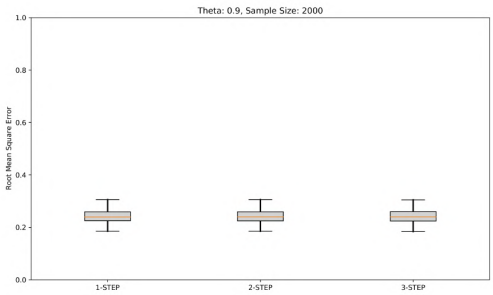
(b) RMSE across Estimation Procedures with $\theta = 0.7$, Sample Size = 1000

Figure 4.6: Comparison of RMSE across Estimation Procedures with different Sample Sizes and θ

As shown in Figure 4.7, the three procedures seem to provide similar performances when the conditional distributions of the manifest variables become more separated ($\theta = 0.8$ or $\theta = 0.9$), with clear overall improvements achieved when the sample size increases.



(a) RMSE across Estimation Procedures with $\theta = 0.9$, Sample Size = 500



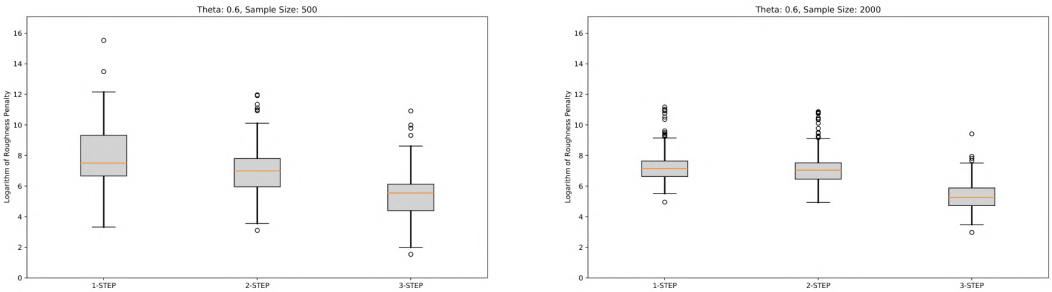
(b) RMSE across Estimation Procedures with $\theta = 0.9$, Sample Size = 1000

Figure 4.7: Comparison of RMSE across Estimation Procedures with different Sample Sizes and θ = 0.9

4.4.3 Roughness of the Estimated Function

This Section focuses on a comparison among the estimated functions based on their roughness, measured by their integrated squared second partial derivatives. Detailed results are reported in Appendix C.2. In scenarios characterized by minimal class separation, quantified as $\theta = 0.6$, the 1-Step procedure seems to produce estimated functions characterised by a larger roughness, while those obtained using the 3-Step procedures tends to be smoother. The 2-Step procedure appears to produce intermediate results (see Figure 4.8a). When the sample size increases to 1000 or

to 2000, the discrepancy between 1 and 2-Step narrows, alongside with the dispersion of the roughness values. At the same time, the 3-Step procedure continues to produce estimated functions that are smoother than those obtained with the other procedures (see Figure 4.8b).

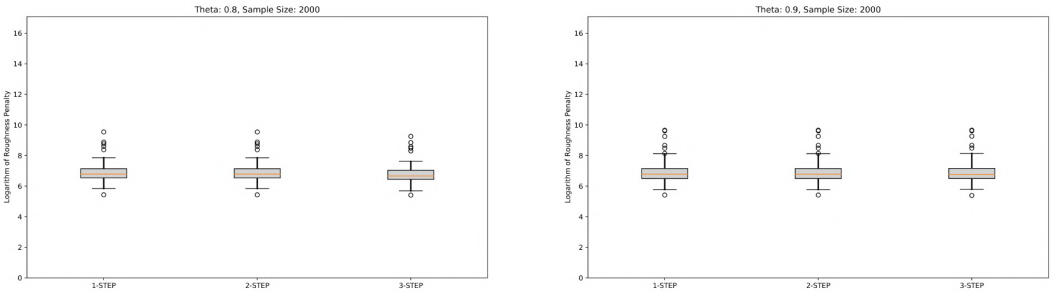


(a) Roughness of the estimated functions with $\theta = 0.6$, Sample Size = 500

(b) Roughness of the estimated functions with $\theta = 0.6$, Sample Size = 2000

Figure 4.8: Comparison of the estimated functions based on their roughness with $\theta = 0.6$ and different Sample Sizes

Conversely, when class separation increases to $\theta \geq 0.7$, the roughness across the different estimation procedures do not show noticeable variations.



(a) Roughness of the estimated functions with $\theta = 0.8$, Sample Size = 2000

(b) Roughness of the estimated functions with $\theta = 0.9$, Sample Size = 2000

Figure 4.9: Comparison of the estimated functions based on their roughness with Sample Sizes = 2000 and different θ

4.4.4 Selection of the Number of Internal Knots

The three estimation methods can be compared also in terms of possible differences in the optimal number of internal knots selected according to AIC and BIC. Detailed results are reported in appendix C.4.

When examining the 1-Step estimation for sample size set to 500 or 1000, independently from the θ values, the knots' choices through AIC shows consistency,

with a predominant preference (in at least 55 of 100 iterations) for one internal knot. A discrepancy is instead found for a sample of 2000 observations. Here, less than half picks one knot, and a bigger class separation seems to lead to a higher number of replications selecting two, three or four internal knots. As the number of observations increases, a corresponding increase in the number of internal knots is observed. This seems a distinctive feature of the results obtained using the AIC, which has a propensity to favor more complex models when a larger dataset provides more information to employ for model selections. On the contrary, BIC mostly selects one internal knots, independently from θ and sample size, in over 90% of iterations.
The attention is now directed toward the 2-Step estimation procedure. In scenarios with a very low class separation (0.6), a higher proportion of iterations selects one internal knot. As θ values increase, the choices of selected knots aligns closely with the 1-Step estimation method.
In the case of the 3-Step procedure, the interval $0.6 \leq \theta \leq 0.7$ leads to even more iterations favoring one internal knot. With greater class separations, the pattern of knot selection mirrors that observed in both the 1-Step and 2-Step methods.

4.5 Results about the Selection of the Number of Latent Classes

The second part of the simulation study is focused on analysing possible impacts of the choice of the estimation procedure on the ability in accurately determining the number of latent classes. In each of the scenarios obtained, combining different levels of class separation and different sample sizes, models with a number of latent classes ranging from 1 to 4 have been fitted using the three estimation procedures.. The data are generated using the *gendati* and *add_bspline_columns* functions, with the true number of latent classes set at two. This process is conducted across 100 replications and seeds for each combination of estimation method, sample size, class separation, and number of latent classes, while allowing the number of internal knots number to range from one to seven.
The objective of predicting the number of latent classes is achieved through the function *evaluate_methods_groups*. This function generates the data under diverse parametric conditions, and fits models by varying estimation procedure and latent classes' number.

Within each replication and scenario, the function calculates the AIC as the metric for model comparison. Following the identification of the model with the lowest AIC value, the function records the minimum weight associated with the fitted latent classes. This step ensures that no class is left empty and examines whether the classes are of negligible size. This approach facilitates the comparison among estimation procedures and their consistency, while assessing their strengths and weaknesses in accurately uncovering the underlying latent class structure.

4.5.1 Comparative Analysis of the Optimal Number of Latent Classes Across Estimation Methods

Consider, for the three estimation method, the number of latent classes selected for every combination of class separation and sample size, as shown in Figure 4.10.
Focusing on the 1-Step estimation method, it is apparent its poor performance in identifying the correct number of latent classes, especially when both the sample size and θ assume small values. For example, in the scenario with 500 observations and a class separation of 0.6, the correct number of latent classes is selected only for 25% of the replications, while for majority of replications the selected value is 4.
When the separation θ is increased, the performance of the 1-Step procedure seems to improve. This trend is more pronounced when the parameter θ passes from very low (0.6) to low (0.7).
Conversely, the transition to higher class separations seems to lead to a detriment in the performance of the 1-Step procedure, with an decrease in the percentage of replications having a correctly selected number of latent classes. This phenomenon, which may be caused by the tendency of the Akaike Information Criteria to attempt the fit of more complex models when the amount of information available is big, is especially evident when the class separation passes from low to medium, and the sample size is 2000.
The 1-Step's ability into choosing the correct number of latent classes is low when compared to the 2 and 3-Step estimation methods. This is likely due to the potential susceptibility of the 1-Step method to initial class overlap.
The 2-Step and 3-Step procedures make considerable improvements, probably thanks to their intermediate step that refines class assignments after the initial fitting of the measurement model. Even with a very low class separation and with 500 observations, the 2-Step and 3-Step procedures correctly identifies the correct number of

latent classes in 86 and 90 replications, respectively. Fixing the class separation and increasing the sample size bring a further increase, with both techniques with over 95% of correct selection of the number of classes.
Nevertheless, a discernible difference can be spotted between the two methodologies. In fact, the most substantial gains are seen with the 3-Step method, which steadily improves its performance with greater class separation and sample size. On the contrary, the 2-Step seems to suffer the same issue as the 1-Step technique, with a higher chance of being selected for more complex models. The reason for this difference may be the 3-Step's introduction of an additional phase, capitalizing on the improved class separation to enhance model performance.
The results seem to prove that the Stepwise procedures highly improve the selection of the number of LCs. Still, the effect of increased sample sizes shows that the 2-Step method is not totally capable of adhering to a parsimonious model selection criteria, as seen in the 1-Step results. The 3-Step is instead greatly able to avoid overfitting.

Class Separation	Sample Size	1-Step				2-Step				3-Step			
		1 LC	2 LC	3 LC	4 LC	1 LC	2 LC	3 LC	4 LC	1 LC	2 LC	3 LC	4 LC
VERY LOW (θ = 0.6)	500	0	25	25	50	3	86	9	2	4	90	5	1
	1000	0	59	17	24	0	95	5	0	0	97	3	0
	2000	0	78	15	7	0	95	5	0	0	95	5	0
L0W (θ = 0.7)	500	0	37	24	39	0	87	9	4	0	95	4	1
	1000	0	55	21	24	0	88	11	1	0	95	5	0
	2000	0	75	13	12	0	97	2	1	0	100	0	0
MEDIUM (θ = 0.8)	500	0	39	31	30	0	90	9	1	0	96	4	0
	1000	0	54	25	21	0	94	5	1	0	98	2	0
	2000	0	62	25	13	0	89	9	2	0	100	0	0
HIGH (θ = 0.9)	500	0	40	33	27	0	85	13	2	0	95	5	0
	1000	0	57	27	16	0	88	8	4	0	97	2	1
	2000	0	67	20	13	0	91	7	2	0	99	1	0

Figure 4.10: Frequency of selected latent classes by estimation methods across various class separations and sample sizes

4.5.2 Comparative Analysis of Selected Latent Classes' Weights Across Estimation Methods

The following analysis presents a detailed evaluation of the minimum weights associated with the function *evaluate_methods_groups* determining the number of latent

classes, for various values of class separation and sample size. The examination is repeated for the three distinct estimation approaches.

Let's begin by saying that the data generation process leads to two latent classes, having on average 47% and 53% of the simulated observations. The iterations performed applying the 1-Step estimation method, for $\theta = 0.6$, seem to include on average 38.9%, 41.9%, and 44.4% in the smaller class, when choosing the correct number of latent classes and changing the sample size 500, 1000 and 2000, respectively. With higher level of class separations, the average gets closer to the correct fraction of observations, even with the smaller value of sample size. This shows how crucial is the class separation and its influence of the manifest variables in classifying the observations with the 1-Step method. When this method picks instead three latent classes, the percentages of observations sorted in the smallest classes seem to get smaller as the class separation increases, going from 24.5% for $\theta = 0.6$ to 8.1% for $\theta = 0.9$, with the sample size fixed at 2000. Another major difference is the fact that the absolute minimum value for each parameter combinations gets smaller the more the class separation increases. The reason behind this may be again the improved ability of the 1-Step method to correctly detect the number of latent classes, reducing the amount of observations inserted in the superfluous classes. Similar conclusions can be reached when the number of LC chosen is four. For this instance, the average minima and absolute minima unsurprisingly get even smaller, being respectively 7% and 0.4% with high class separation and 2000 observations.

The 2-Step estimation process, when two is the number of latent classes selected, performs similarly to the 1-Step, both when considering the average minima and absolute minima. In fact, the average minima are all very close to the expected one (0.46), with a discernible increase in the absolute minima as class separation advances from very low to high, for every combination of class separation and sample size.

When the selected number of latent classes is three, the average minima are again similar to the 1-Step techniques, while the absolute ones tend to be less distant from the averages when $0.6 \leq \theta \leq 0.7$. Stronger class separations tend to create a third class with small amounts of observations. With four LC, the weights are similar to the 1-Step, but the low amount of replications selecting four, as shown in Figure 4.10 makes any assessment's reliability compromised due to insufficient data.

The 3-Step method, for two latent classes, exhibits minima almost identical to the ones of the 2-Step technique. The fact that so few iterations have chosen three

and four classes undermines any assessment's credibility. What could be said, focusing on three classes, is the presence of bigger average and absolute minima for $0.6 \leq \theta \leq 0.8$ with respect to the 2-Step ones.
An examination of the impact of sample size, with a fixed class separation, reveals that a greater number of observations does not uniformly translate to increased minimum weights for all methods. This phenomenon suggests that while an abundance of data yields more informational content, it does not lead to an indiscriminate inflation in the number of latent classes identified by these methods. The instance where a bigger sample size leads to an improvement in the minimum weights is with a very low class separation (0.6).

Class Separation	Sample Size	1-Step				2-Step				3-Step			
		1 LC	2 LC	3 LC	4 LC	1 LC	2 LC	3 LC	4 LC	1 LC	2 LC	3 LC	4 LC
Very Low (0.6)	500	None	0.389 (0.313)	0.185 (0.079)	0.117 (0.050)	1.000 (1.000)	0.399 (0.212)	0.210 (0.112)	0.152 (0.124)	1.000 (1.000)	0.401 (0.212)	0.183 (0.112)	0.124 (0.124)
	1000	None	0.419 (0.210)	0.221 (0.132)	0.153 (0.044)	None	0.426 (0.306)	0.272 (0.204)	None	None	0.427 (0.306)	0.264 (0.204)	None
	2000	None	0.444 (0.260)	0.245 (0.203)	0.201 (0.165)	None	0.450 (0.273)	0.298 (0.250)	None	None	0.453 (0.362)	0.294 (0.236)	None
Low (0.7)	500	None	0.460 (0.401)	0.140 (0.055)	0.102 (0.020)	None	0.460 (0.372)	0.162 (0.055)	0.122 (0.076)	None	0.461 (0.372)	0.190 (0.115)	0.139 (0.139)
	1000	None	0.466 (0.406)	0.145 (0.047)	0.096 (0.025)	None	0.465 (0.374)	0.144 (0.070)	0.087 (0.087)	None	0.465 (0.374)	0.123 (0.080)	None
	2000	None	0.464 (0.404)	0.149 (0.078)	0.122 (0.056)	None	0.465 (0.405)	0.191 (0.183)	0.091 (0.091)	None	0.465 (0.405)	None	None
Medium (0.8)	500	None	0.468 (0.419)	0.108 (0.014)	0.069 (0.016)	None	0.469 (0.419)	0.110 (0.023)	0.176 (0.176)	None	0.469 (0.419)	0.125 (0.027)	None
	1000	None	0.467 (0.434)	0.108 (0.013)	0.070 (0.008)	None	0.469 (0.432)	0.158 (0.014)	0.039 (0.039)	None	0.468 (0.411)	0.172 (0.130)	None
	2000	None	0.466 (0.432)	0.125 (0.011)	0.070 (0.013)	None	0.466 (0.431)	0.172 (0.083)	0.019 (0.017)	None	0.466 (0.431)	None	None
High (0.9)	500	None	0.462 (0.420)	0.109 (0.006)	0.068 (0.015)	None	0.466 (0.418)	0.081 (0.008)	0.013 (0.005)	None	0.467 (0.418)	0.031 (0.008)	None
	1000	None	0.467 (0.415)	0.107 (0.003)	0.055 (0.005)	None	0.468 (0.415)	0.106 (0.010)	0.042 (0.011)	None	0.468 (0.415)	0.030 (0.010)	0.004 (0.004)
	2000	None	0.465 (0.433)	0.081 (0.004)	0.070 (0.003)	None	0.466 (0.433)	0.117 (0.007)	0.133 (0.107)	None	0.466 (0.433)	0.007 (0.007)	None

Figure 4.11: Average minimum weights of latent classes, with the absolute minimum for each category in parentheses, by estimation methods across various class separations and sample sizes

4.5.3 Conclusions on the study with a variable number of latent classes

The results of the study have detected noticeable differences in accuracy, with the Stepwise methods outperforming the 1-Step method, especially when the class separation is low, and the sample sizes are small. These improved performances may

be possible thanks to the extra steps in the 2 and 3-Steps techniques, which make the groups clearer, reducing the overlap, while leading to more accurate identification. Among these, the 3-Step method stands out as particularly effective, consistently identifying the correct number of groups across different scenarios. Its additional step seems to effectively help its performance, avoiding the overfitting problem seen in the 1-Step method and, to a lesser degree, in the 2-Step method.
The study has also showed that having a higher amount of observations in the samples does not always lead to better results for all methodologies, especially when group distinctions are very slight. On the contrary, class separation is crucial, since it seems to affect the relationship between the various manifest variables, and the ability of the estimation methods in detecting the correct number of latent classes.

Chapter 5

Brexit Data Analysis

5.1 Introduction

The following chapter aims at applying to real world data the Concomitant Variable Latent Class Model and its flexibility, gained from the expression of the effects of the concomitant variables on the latent class probabilities by means of B-splines. The emphasis will be on identifying voting behaviour trends within the UK House of Commons through an examination of specific voting that happened between March 25th and April 1st 2019, during Brexit negotiations.
The British electoral system is lead by the prime minister and the party they are a member of. The remaining parties become the opposition, whose role is to be opposed to the government, not by following an coordinated strategy, but by just confronting the administration's plans. In the UK political system, a Member of Parliament (MP) is elected to represent a constituency in the House of Commons, which comprises 650 elected officials.
In 2010, Spirling and Quinn applied a Dirichlet process mixture model for analyzing the voting record of the House of Commons. Their model relied on the presence of highly disciplined parties to obtain an acceptable performance. The parties' stances in the House of Commons are denoted by a great cohesiveness when it comes to voting. The authors note that, during the first term of Prime Minister Blair's administration, approximately 99% of parliamentary decisions were aligned with the directives of each party's leadership. Deviating from the party line typically resulted in demotion (Spirling and Quinn, 2010).
Conversely, this idea of cohesiveness of the House of Commons has been diminished since the beginning of Brexit consultations. Aidt, Grey and Savu have investi-

gated the reasons behind politicians going against the line of their party, focusing on the Brexit votes held in the House of Commons in early 2019 (Aidt et al., 2021). In the votes, significant dissent was observed within the Conservative Party. The study outlines three pivotal factors that persuaded MPs to dissent: their personal ideological beliefs, the political leanings of their voters, and considerations regarding their political futures.

Politicians' ideological stance has played a critical role in MPs' decision-making that lead to rebel against their party line. Ideology influences how politicians interpret policy implications, aligning or not with their core beliefs and principles. To exemplify this concept, the authors have demonstrated how MPs that supported the "Leave" during the 2016 Brexit Referendum, were more inclined to go against any deal perceived as compromising the Country's sovereignty, despite party directives. Regarding the preferences of an MP's constituency, this seems to significantly impact their willingness to diverge from party lines. Politicians' awareness of their electoral survival leads them to acutely follow the views of their constituents. When there is a clear divergence between party policy and constituency sentiment, as was evident in areas with strong Leave or Remain leanings during the Brexit Referendum, MPs may choose to vote against the party line to align with their voters' preferences.

Career concerns also influence MPs' decisions to rebel, though this factor can both deter and motivate rebellion. On one hand, aligning with the party can be crucial for career advancement within the political hierarchy, deterring rebellion. On the other hand, taking a stand on principle, especially on pivotal issues like Brexit, can enhance an MP's reputation among constituents and within certain party factions (Aidt et al., 2021).

5.2 Description of the Data

The study utilizes a dataset composed of information about the results of 16 divisions held between March 25th and April 1st 2019 in the House of Commons. The data contains the votes of 638 MPs, compiled through the use of the *hansard* R package (Odell, 2017), and two different variables, including information about each MPs and their constituencies.

Specifically, the attention will be focused on the proportion of votes in favor of "Leave" in the Brexit Referendum within their constituencies, and the effective

number of competing candidates in each constituency. For the former piece of information, the proportion has been estimated through a demographic model by Hanretty (Hanretty, 2017). Following Berrettini et al. (To Appear, 2024), the number of competing candidates is calculated by first considering the votes received by the 12 main parties (Apostolova et al., 2017), except in constituencies where a party outside of these main 12 won. There, their votes are categorized separately, bringing the total number of categories to 14 (Berrettini et al., To Appear, 2024). Then, the variable that assesses the diversity of vote distribution is calculated using Shannon's Entropy of the vote shares. This value provides a basis for determining the effective number of competing candidates, ranging from a minimum of 0 (indicating no diversity) to a maximum equal to the logarithm of the number of parties, in cases of equal vote distribution:

$$H(\omega_i) = -\sum_{k=1}^{P} \omega_{ki} \log(\omega_{ki}). \tag{5.2.1}$$

By considering the exponential of Shannon's Entropy, $\exp(H(\omega_i))$, the variable will range from a minimum of 1 to a maximum set to the number of parties (Berrettini et al., To Appear, 2024). The study will omit certain Members of Parliament from the analysis: this includes the four individuals holding the office of Speaker who traditionally do not engage in voting, representatives from the Irish political party Sinn Féin who choose not to take their seats, and an MP who deceased within the time frame of the investigation.

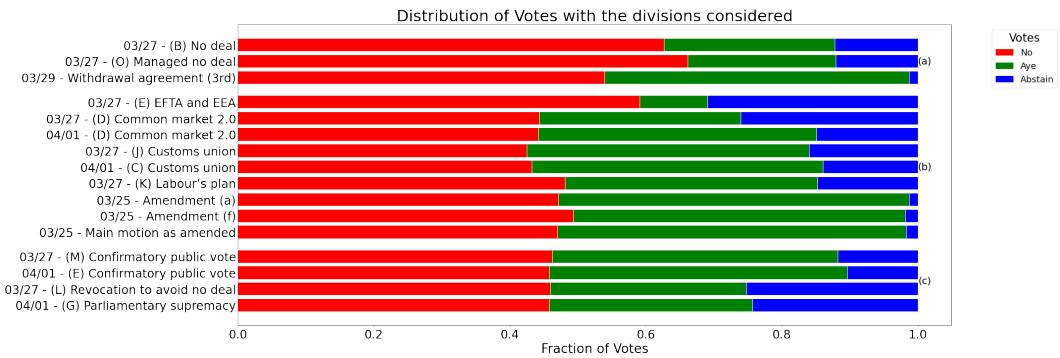


Figure 5.1: Vote fractions for the 16 divisions considered. Panels: (a) "hard Brexit"; (b) "soft Brexit"; (c) "pro-Remain"

The 16 divisions considered were aimed at exploring potential Brexit options,

which ultimately led to a postponement of the Brexit date. These 16 votes are categorized based on their implications for Brexit, ranging from “hard Brexit” options to “soft Brexit” and “pro-Remain” stances, as can be observed in Figure 5.1. The 16 divisions are categorical variables, possibly taking values equal to “Aye”, “No” and “Abstain”.
The analysis seeks to identify clusters of MPs with similar voting patterns found in the data, utilizing concomitant variables to improve the interpretation of these patterns and to refine the model accuracy. In particular, this chapter aims at investigating whether the inclusion of different concomitant variables or the use of different estimation methods for the Concomitant Variable Latent Class Model yields contrasting clustering outcomes, in order to understand if the model is robust or adaptable enough to capture the nuances of voting behaviours between and within parties.

5.3 Optimal Number of Latent Classes

In order to determine the optimal number of LC, the function *evaluateLC* is designed for the exploration of model configurations. It operates through an iterative process that evaluates the performance of models across a spectrum of component counts (ranging from 7 to 20 latent classes) and knot numbers (from 1 to 6) for the B-spline expression of the concomitant variable, within a specified seed range to assure reproducibility.
The evaluation employs the function *Stepmix* included in the Python package *stepmix* (Morin et al., 2023), and applies the three estimation techniques discussed in Chapter 3. For each iteration, these three procedures are employed for every combination of number of components and internal knots. To ensure convergence, the function initializes the analysis 20 times, with a maximum of 100 iterative instances. *evaluateLC* picks the best number of classes by selecting the combination of parameters with the lowest AIC. The analysis proceeds smoothly due to a custom mechanism that bypasses iterations encountering singular value decomposition (SVD) errors. This ensures continuity despite the *Stepmix* function’s current limitation in handling SVD issues, preventing a complete halt of the analysis process.
The full analysis is repeated with a different seed to ensure robustness and stability of the numbers of latent classes chosen by the three estimation methods.
In the analysis, political parties will be referred to as follows: C for Conservative Party, DUP for Democratic Unionist Party, GP for Green Party, Ind for Indepen-

dents, Lab for Labour, LD for Liberal Democrats, PC for Plaid Cymru, and SNP for Scottish National Party.

5.4 Clustering with "Leave" as Concomitant Variable

The first part of the analysis is conducted by applying the Concomitant Variable Latent Class Model, with "Leave" as concomitant variable. For every MPs, "Leave" measures the proportion of votes in favor of "Leave" in the Brexit Referendum in their constituencies.

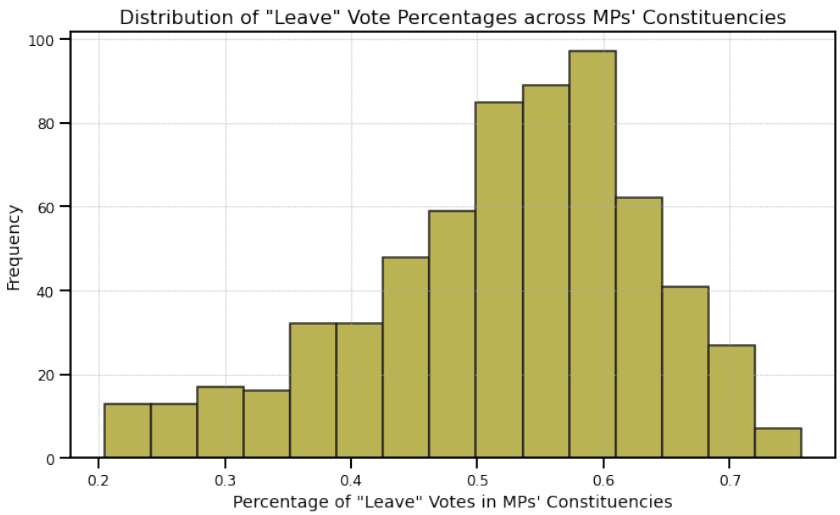


Figure 5.2: Distribution of "Leave" Vote Percentages across MPs' Constituencies

Given the asymmetric distribution of the data, as can be seen in Figure 5.2, the construction of B-splines will utilize quantiles for partitioning the domain, rather than segmenting it into equal intervals, as it was done in the Simulation of Chapter 4. This approach ensures a more accurate representation of the underlying distribution by aligning the spline adjustments with the data actual spread.
The function *evaluateLC* is applied to the dataset twice, for every combination of estimation method, number of latent classes, and internal knots. The following heatmaps in Figure 5.3 contain the lowest values of AIC of the combinations. It can be noticed how, for the 2-Step procedure with 16 components and 6 knots the cell is empty, indicating non-convergence for the selected seeds. This specific scenario has been reanalysed with alternative seeds (with the minimum obtained being 41366.41), confirming that the resulting AIC values align closely with those of neighboring cells, suggesting consistency in the model performance across similar

configurations.
The lowest levels are reached with 14 classes and 1 internal knot for the 1-Step estimation method, and with 13 classes and 1 internal knot for both Stepwise procedures. The heatmap generated from the 1-Step technique reveals an interesting trend: a broad range of combinations between the number of latent classes and internal knots yield AIC values that are close to the minimum. This observation suggests robustness in the model performance, as it reaches comparable results across different parameter configurations. Conversely, such flexibility is not observed with Stepwise procedures. These methods exhibit a more constrained behaviour, where AIC values close to their minimum are found in a more limited number of combinations of latent class counts and knot configurations.

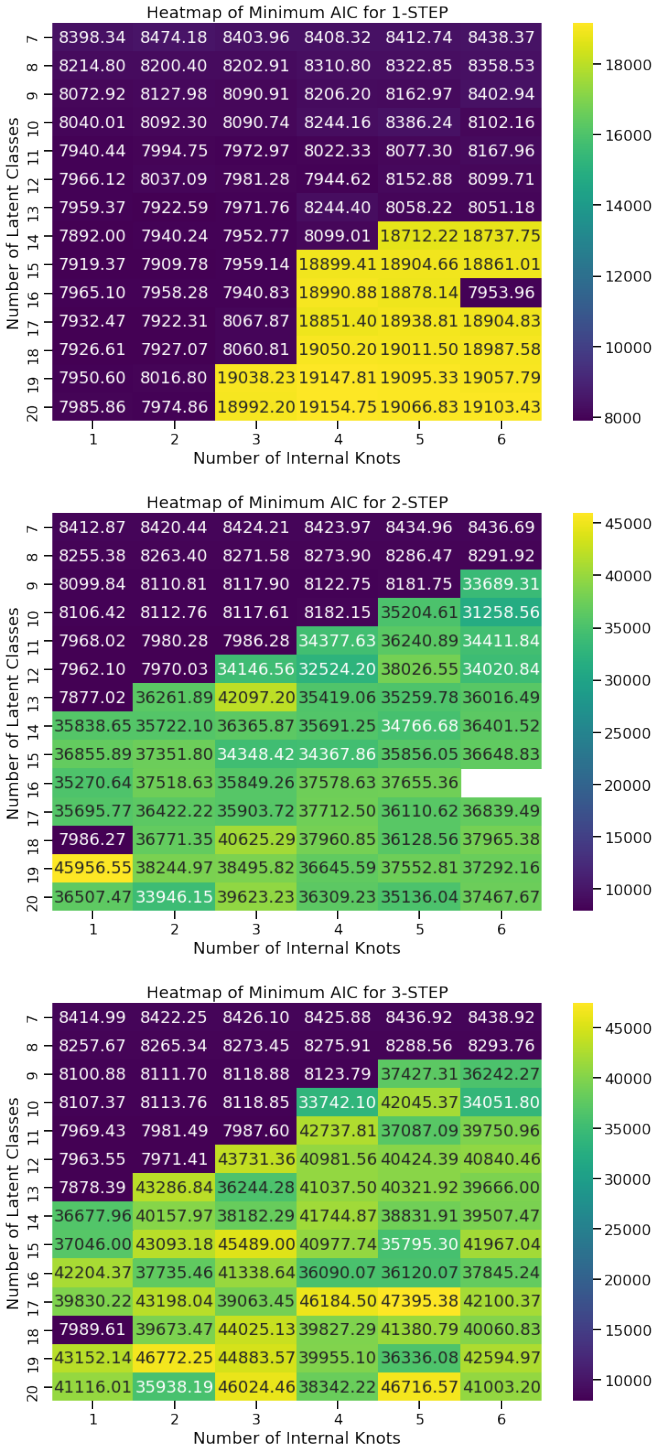


Figure 5.3: Heatmaps of AIC across MPs' Constituencies for Different Estimation Methods using "Leave" as Concomitant Variable

5.4.1 Results obtained using the 1-Step method

The following analysis will be conducted employing a Concomitant Variable Latent Class Model with 14 components, 1 internal knot, and utilizing the variable “Leave” as concomitant variable. The model is fit to the data, and used to predict the clusters of the MPs.

Predicted Groups	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Party														
C	21	8	152	3	53	13	0	0	0	22	28	10	0	3
DUP	0	10	0	0	0	0	0	0	0	0	0	0	0	0
GP	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Ind	0	0	0	12	1	0	0	1	0	0	0	0	0	2
LD	1	0	0	1	0	0	0	1	0	0	0	0	0	9
Lab	2	2	1	6	2	1	16	91	97	0	0	3	24	2
PC	0	0	0	0	0	0	0	0	0	0	0	0	0	4
SNP	0	0	0	0	0	0	0	0	0	0	1	0	0	34

Figure 5.4: Predicted Groups vs Actual Parties with 1-Step estimation and “Leave” as concomitant variable

As depicted in Figure 5.4, Group 2 appears noteworthy, comprising 152 Conservative Party members out of 313. This cluster is significant for encapsulating members who predominantly support a hard Brexit stance, highlighting a specific ideological segment within the Conservative Party. This focus allows for a deeper understanding of the party’s internal divisions regarding Brexit.

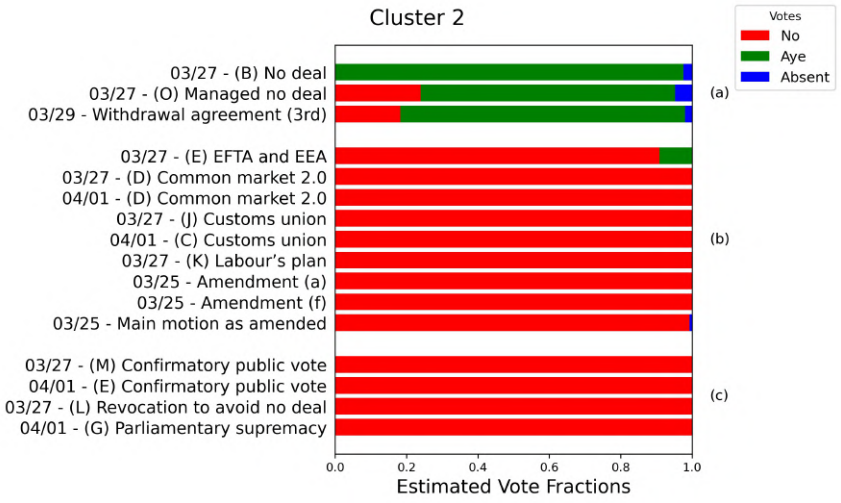


Figure 5.5: Clusters 2 with 1-Step estimation and "Leave" as concomitant variable

As can be seen in Figure 5.5, Cluster 2 contains in fact 149 Conservative MPs who supported the "No deal" vote on March 27th, out of a total of 157. This insight demonstrates the model capability to distinguish varying viewpoints within the party, noting that only 3 MPs included in this cluster chose to abstain from this particular motion. Since the CVLCM appears to distinguish easily this cluster, it will be chosen as baseline for the analysis.

The model identifies ideological alignments within major parties and also clusters smaller parties effectively. It places the majority of Conservative MPs into Group 2, then into Cluster 4 (having 53 members of the party), and the rest in Groups 0, 5, 9, 10, 11.

By analysing the relative log odds of Cluster 4 with 2 as baseline in Figure 5.6, it is possible to understand how the opinion towards Brexit of the respectively represented constituencies influences the MPs' votes. The probability for an MP to belong to this cluster gets very low as the percentage of Leave votes at the Brexit Referendum gets bigger than 50%.

In fact, the main difference between Cluster 2 and 4 is how the former contains MPs in favour of a No deal Brexit, whilst the latter has mostly votes against the "No deal" and "Managed no deal" proposals, as Figure 5.7 depicts.

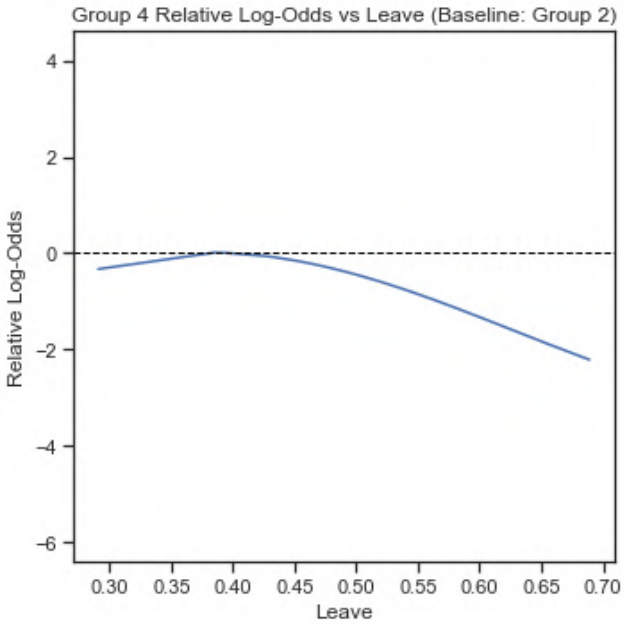


Figure 5.6: Relative Log-Odds in Cluster 4 (Cluster 2 as baseline)

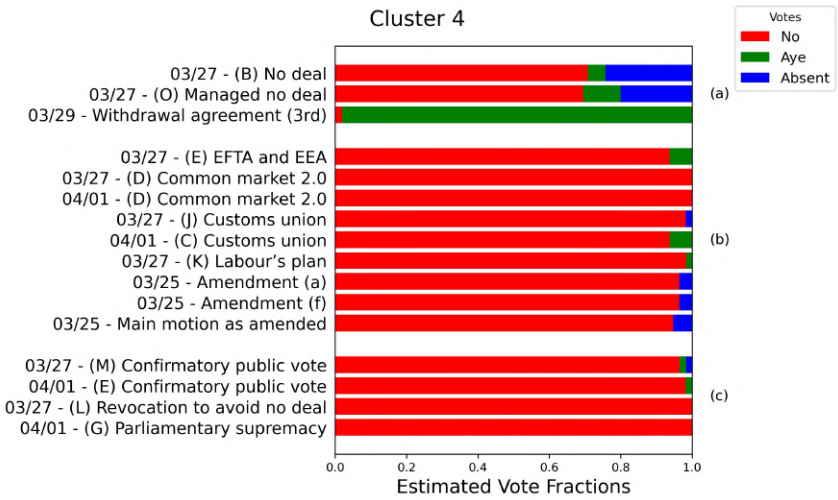


Figure 5.7: Clusters 4 with 1-Step estimation and “Leave” as concomitant variable

Clusters 0, 5, 9 and 11 predominantly consist of Conservative MPs, yet discerning patterns within these groups is challenging due to the heterogeneous structures of the estimated vote fractions for the divisions, as detailed in Appendix D.1. Conversely, Cluster 10 contains Conservative MPs who abstained on pro-Remain divisions, either voted against or abstained on soft Brexit options, and refrained from voting on the No Deal and Managed No Deal divisions, but aligned with the party’s stance by supporting the Withdrawal Agreement on March 29, as illustrated

in Figure 5.9. Cluster 10 includes Prime Minister Theresa May and twenty other members of the second May ministry (70% of the ministry is part of this group). This explains why the MPs in the cluster voted in favour of the third Withdrawal agreement, proposed by PM May. The relative log-odds of the cluster, as depicted in Figure 5.8, show how the probability of a Member of Parliament of belonging to the group decreases as the Leave vote percentage in their constituency exceeds 45% in the Brexit Referendum. Considering the information that the MPs in Cluster 10 are from constituencies with a lower inclination towards Brexit, and the fact that 72.41% are part of the ministry, this explains the reason behind their abstention.

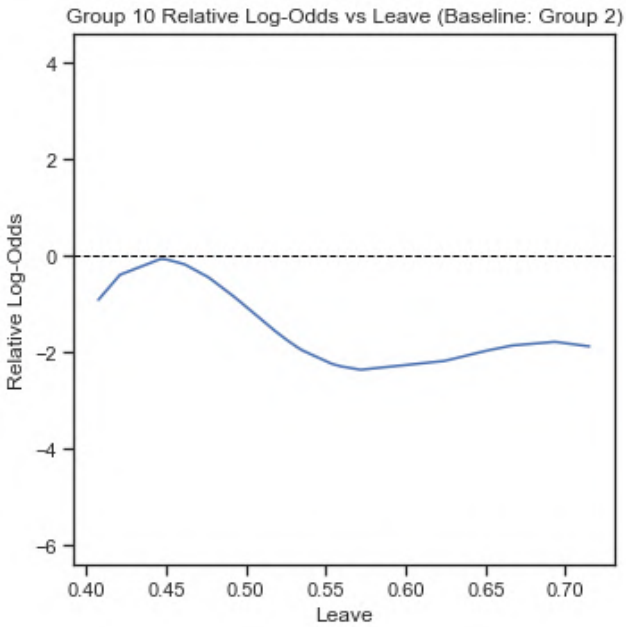


Figure 5.8: Relative Log-Odds in Cluster 10 (Cluster 2 as baseline)

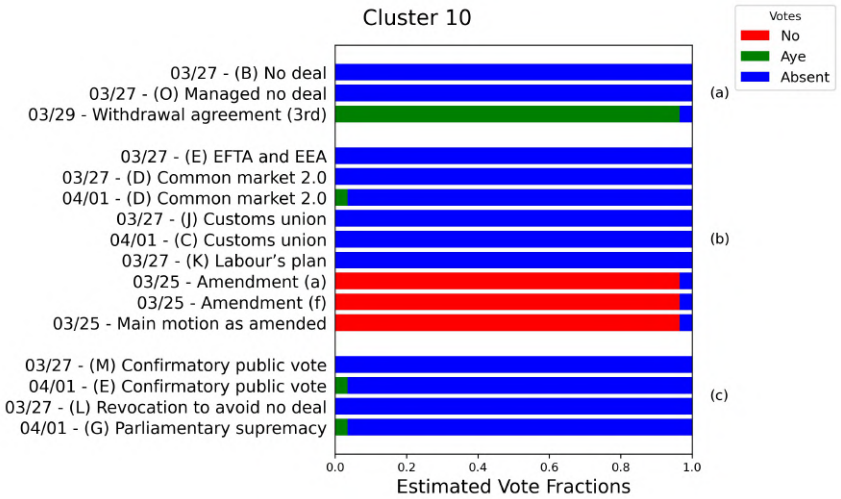


Figure 5.9: Clusters 10 with 1-Step estimation and “Leave” as concomitant variable

Conversely, the Labour Party primarily occupies Clusters 7 and 8, respectively including 36.84% and 39.27% of the party’s members. As depicted in Figure 5.10, Cluster 8 leans heavily towards Remain, showing unanimous support in the divisions on the topic, whereas Cluster 7 includes a significant number of abstentions. Both Clusters voted against hard Brexit divisions, while being in favour of a soft alternative.

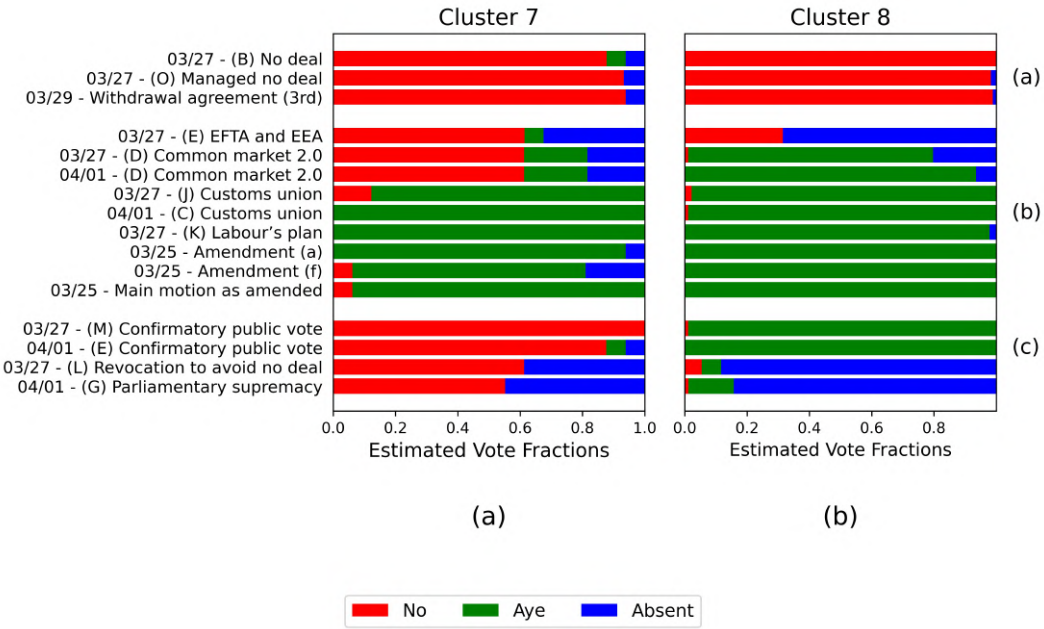


Figure 5.10: Clusters 7 and 8 with 1-Step estimation and "Leave" as concomitant variable

The remaining Labour MPs are included in Groups 6 and 12. Analysing Figure 5.11, Cluster 6 uniformly opposes remaining in the EU, as evidenced by their votes against or abstentions on pro-Remain measures. Similarly to Cluster 7, Cluster 12 is composed of Labour MPs who chose to abstain from voting on issues related to the UK's potential continued membership in the EU.

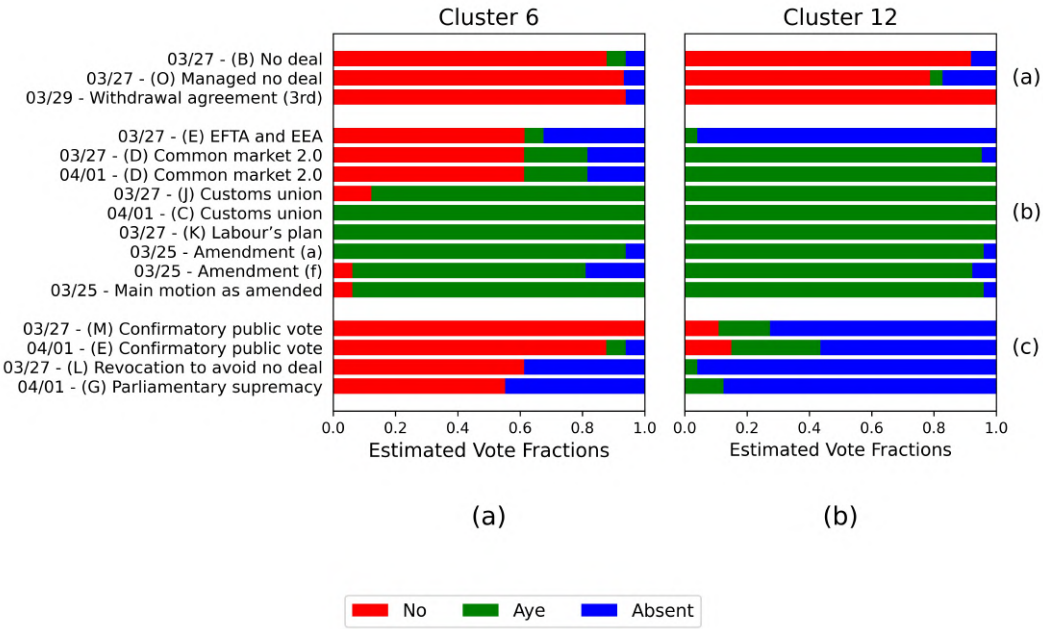


Figure 5.11: Clusters 6 and 12 with 1-Step estimation and “Leave” as concomitant variable

Cluster 13 has variety in the parties that it contains, but it includes almost the totality of MPs from the Scottish National Party (SNP), the Liberal Democrats (LD), and all the MPs from Plaid Cymru (PC). By observing the estimated vote fractions in Figure 5.12, it is evident how the group has strong pro-Remain stances, with nearly all members supporting the four divisions related to the possibility of the UK staying in the EU.

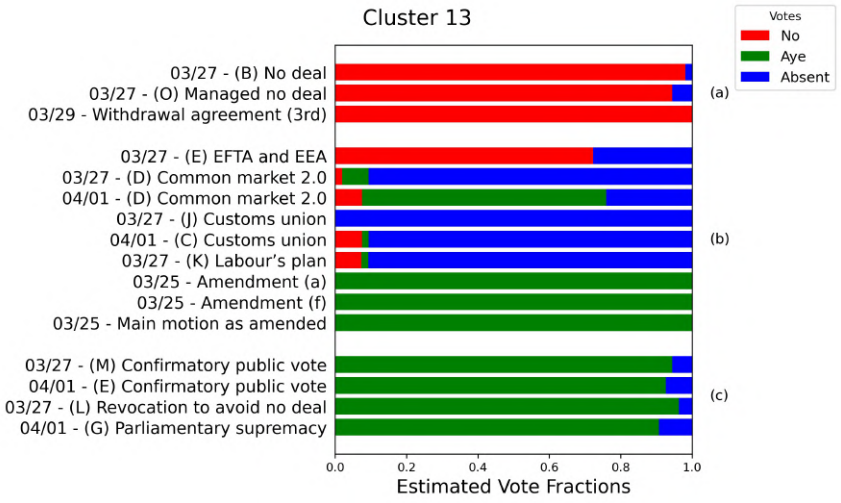


Figure 5.12: Clusters 13 with 1-Step estimation and "Leave" as concomitant variable

Clusters 1 and 3 have variety in them, but still present respectively the totality of Democratic Unionist Party (DUP) and almost all Independents (IND).
The remaining images depicting vote fractions and relative log-odds of the clusters can be observed in Appendix D.1.

5.4.2 Results obtained using the 2-Step method

The Concomitant Variable Latent Class Model reaches the lowest AIC value with 13 components and 1 internal knot, when employing the 2-Step estimation method and using "Leave" as concomitant variable. This specific configuration is fitted to the data to predict MPs' groups.

Predicted Groups	0	1	2	3	4	5	6	7	8	9	10	11	12
Party													
C	156	28	0	0	52	23	3	16	3	0	12	6	14
DUP	0	0	0	0	0	0	0	0	0	0	10	0	0
GP	0	0	0	0	0	0	0	0	1	0	0	0	0
Ind	0	0	0	2	1	0	1	0	12	0	0	0	0
LD	0	0	0	1	0	1	9	0	1	0	0	0	0
Lab	1	0	103	106	4	0	2	1	6	20	2	1	1
PC	0	0	0	0	0	0	4	0	0	0	0	0	0
SNP	0	0	0	0	0	0	34	0	0	0	0	1	0

Figure 5.13: Predicted Groups vs Actual Parties with 2-Step estimation and “Leave” as concomitant variable

Considering Figure 5.13, it becomes evident how the CVLCM selects a cluster made almost solely of 156 Conservative MPs out of the total of 313, similarly to the clustering obtained with the application of the 1-Step estimation method.

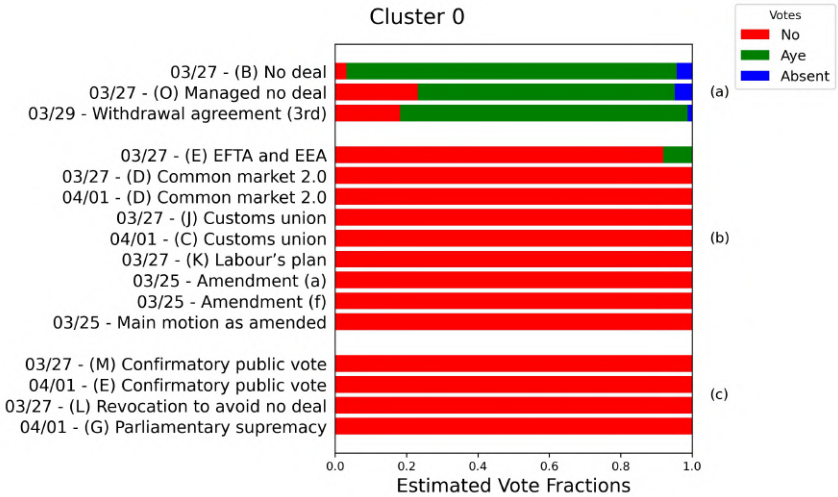


Figure 5.14: Clusters 0 with 2-Step estimation and “Leave” as concomitant variable

By examining Figure 5.14 with the estimated vote fractions of the 16 divisions in Cluster 0, it suggests that Cluster 0 mostly contains MPs advocating for a stringent approach to Brexit. In fact, the group contains the 94.27% of Conservative MPs who voted in favour of the No deal proposal, and the 89.68% who were in favour of the Managed no deal of March 27. This indicates that the 2-Step estimation method is proficient in capturing specific perspectives within political parties, though with a marginally reduced efficacy compared to the 1-Step approach. Specifically, the

2-Step Cluster that identifies Conservative MPs supporting a hard Brexit stance includes MPs who either opposed or abstained from voting on the No Deal motion, accounting for 5.73% of the group's composition. In contrast, the 1-Step technique manages to lower this figure to merely 2.61%, with only four MPs abstaining during the division, and none voting against.

Neverthless, the 2-Step method shows commendable results in clustering such a substantial percentage of Conservative MPs in favour of a hard Brexit within the same group. For the following analyses, Cluster 0 will serve as baseline.

The focus is now shifted to the clusters which contains the remaining Conservative MPs. Cluster 4 includes 16.61% of Conservative MPs, while smaller proportions are distributed among Clusters 1, 5, 7, 10, 11, and 12, with percentages ranging from 8.95% to 1.92%. Similar to what has been observed with the 1-Step estimation method, these clusters predominantly consist of Conservative party members, underscoring the model's capacity to distinguish between the parties within the House of Commons. This further supports the notion that opposition parties tend to vote contrary to the government's stance (Spirling and Quinn, 2010).

Figure 5.15 represents the vote fractions of Clusters 4, 1 and 5, respectively the second, third and fourth clusters having the most Conservative MPs inside of them. Focusing first on the hard Brexit divisions, Cluster 4 and 5 have similar fractions, with the latter having higher percentages of votes against the No deal and the Managed no deal of March 27. Conversely, all MPs included in Cluster 1 have abstained from these two divisions. The three clusters then all had favorable positions towards the Withdrawal agreement (3rd) of March 29. Focusing on soft Brexit divisions, Cluster 1 maintains its abstaining positions in six out of nine votes, while Group 4 is mostly voting against, and Cluster 5 has voted in favour of soft Brexit divisions seven times out of nine. None of these three clusters present Remain views, since Group 1 abstained in the four relevant votes, Group 4 voted against, and Group 5 was split between voting against and abstaining.

These patterns suggest that Cluster 5 comprises Conservative MPs who were against a hard Brexit yet supportive of a softer approach. Conversely, members of Cluster 4 appeared to advocate for a hard Brexit while rejecting no-deal options. The abstentionism of Cluster 1, particularly regarding most divisions except for the third Withdrawal Agreement, can be explained by the fact that it contains PM Theresa May and 70% of her ministry.

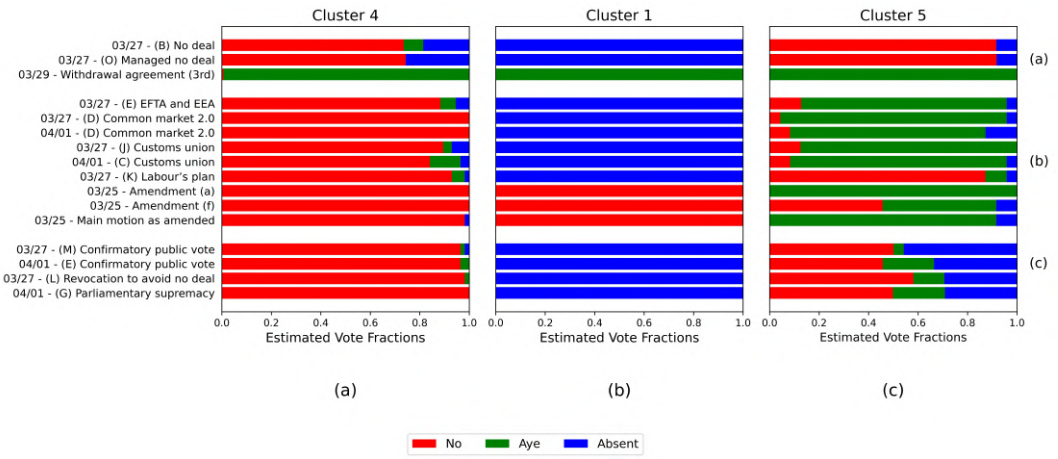


Figure 5.15: Clusters 4, 1 and 5 with 2-Step estimation and “Leave” as concomitant variable

A significant improvement achieved through the 2-Step method, compared to the 1-Step estimation procedure, is its capacity to distinguish a group (Cluster 5), predominantly composed of Conservative MPs in favour of a soft Brexit stance. This was in fact not discernible in the analysis conducted with the 1-Step method. Within the framework of the data under consideration, this outcome is particularly impactful: it demonstrates the CVLCM’s proficiency in distinguishing between political parties and highlights its ability to identify differences within each party. The latter result holds substantial relevance in the context of Brexit, since the cohesiveness of HoC parties was deeply reduced during the negotiations (Aidt et al., 2021), and previously proposed models needed highly disciplined MPs to obtain acceptable results (Spirling and Quinn, 2010).

Shifting the attention to the clustering of the Labour Party, it is mainly clustered in Groups 2 and 3, respectively holding the 41.7% and the 42.92% of the party’s MPs. An additional 8.1% is present in Cluster 9. The voting behaviours are depicted in Figure 5.16. Unsurprisingly, the three clusters all present opposition towards hard Brexit views. Notably, Cluster 9 is different than Groups 2 and 3 because of its voting pattern concerning pro-Remain divisions, with its MPs either voting against these motions or abstaining.

The primary difference between the two main Labour Party clusters is found in their responses to the EFTA and EEA division on March 27, where Cluster 2 opposed, while and Cluster 3 abstained. The divergence is also extended in the votes regarding the Remain-related votes. Cluster 2 has in fact predominantly voted in favour of the four divisions aimed at remaining in the EU, whereas Cluster 3 supported two

and abstained from the other two.

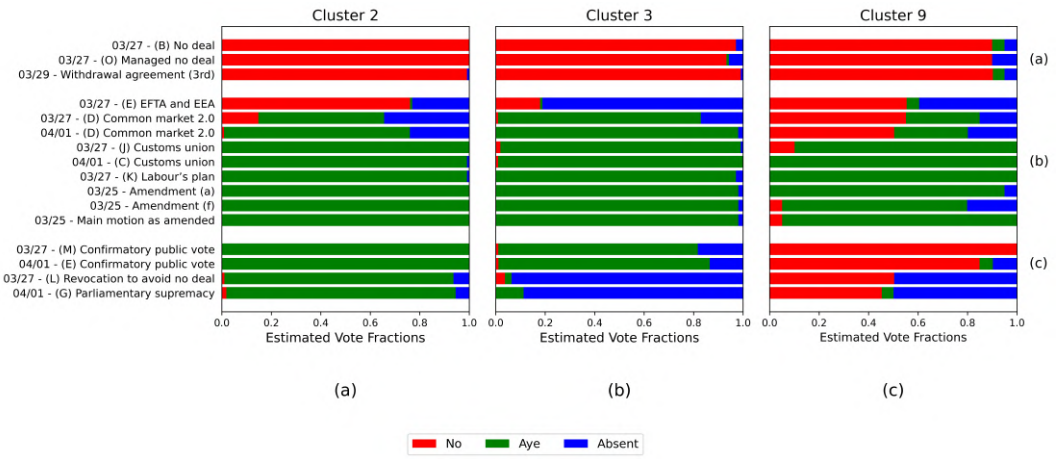
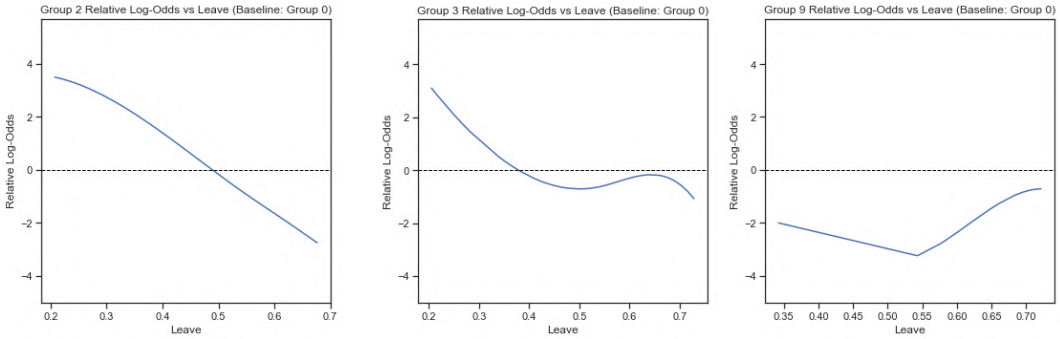


Figure 5.16: Clusters 2, 3 and 9 with 2-Step estimation and "Leave" as concomitant variable

The voting patterns of the three groups can be better understood by examining the relative log-odds depicted in Figure 5.17. For MPs representing constituencies where the Leave vote was below 50% in the Brexit Referendum, there's a high probability of them being included in Cluster 2. Similarly, With Leave vote percentages under 40%, MPs have higher probabilities in belonging to Cluster 3. This correlation offers insight into why MPs within these clusters predominantly supported pro-Remain motions. Conversely, the dynamics for Cluster 9 is different: MPs in this cluster are more likely to come from constituencies with over 50% Leave votes, which could explain their tendency to vote against remaining in the EU, even when being members of the Labour Party. This observation demonstrate the idea for which MPs often align their voting behaviour with the preferences of their constituents, a strategy aimed at securing their political longevity and career prospects.



(a) (b) (c)

Figure 5.17: Relative Log-Odds in Clusters 2, 3, 9 (Cluster 0 as baseline)

Concluding the analysis on the 2-Step estimation method, Cluster 6 contains almost the totality of Members of Parliament of SNP and LD, along wit every member of PC. Meanwhile, Cluster 8 is predominantly made up of nearly all Independents, and the sole elected GP MP. As seen with the 1-Step technique, DUP members are grouped alongside Conservative politicians in Cluster 10.

5.4.3 Results obtained using the 3-Step method

The clustering results obtained with the 3-Step estimation method are identical to those from the 2-Step method, with the sole exception of two MPs switching clusters. This consistency between the two methods underscores the robustness of the clustering analysis when performed with a Stepwise estimation procedure, with the groups and interpretations remaining stable.

Given this high similarity, the results and discussions presented in the previous paragraph for the 2-Step estimation method hold true for the 3-Step method as well. The minor adjustment involving the switching of two MPs does not materially affect the overall analysis or conclusions drawn from the clustering. Therefore, the insights and implications discussed previously are equally applicable and relevant when considering the outcomes of the 3-Step estimation method. The relative log-odds and the vote fractions of the clusters can be observed in Appendix D.5 and Appendix D.6.

Predicted Groups	0	1	2	3	4	5	6	7	8	9	10	11	12
Party													
C	156	28	0	0	52	23	3	16	3	0	12	6	14
DUP	0	0	0	0	0	0	0	0	0	0	10	0	0
GP	0	0	0	0	0	0	0	0	1	0	0	0	0
Ind	0	0	0	2	1	0	1	0	12	0	0	0	0
LD	0	0	0	1	0	1	9	0	1	0	0	0	0
Lab	1	0	105	104	4	0	2	1	6	20	2	1	1
PC	0	0	0	0	0	0	4	0	0	0	0	0	0
SNP	0	0	0	0	0	0	34	0	0	0	0	1	0

Figure 5.18: Predicted Groups vs Actual Parties with 3-Step estimation and “Leave” as concomitant variable

5.5 Clustering with “ExpEntropy” as Concomitant Variable

The following section will try to cluster the Members of Parliament by applying Concomitant Variable Latent Class Model, with “ExpEntropy” selected as concomitant variable. For every MPs’ constituency, “ExpEntropy” is a measure of the effective number of competing candidates, assessing the diversity of vote distribution. It is calculated with the exponential of Shannon’s Entropy of the vote shares.

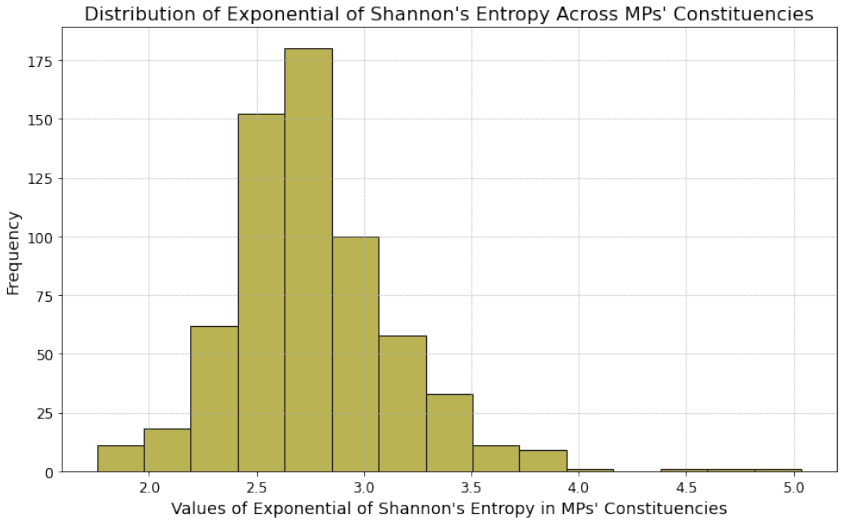


Figure 5.19: Vote fractions for the 16 divisions considered.Panels: (a) “hard Brexit”; (b) “soft Brexit”; (c) “pro-Remain”

Given the asymmetric nature of the distribution of the data, as can be seen in

Figure 5.19, the B-splines will be created by utilizing the quantiles of the domain, rather than segmenting it into equal intervals. This adjustment is employed to ensure a more accurate representation of the underlying distribution, since it aligns the splines to the variable they try to represent.
The results are obtained by applying twice the function *evaluateLC* is applied to the dataset, for every combination of estimation method, number of latent classes, and internal knots. The following heatmaps in Figure 5.20 contain the lowest values of AIC of the combinations.
The presence of empty cells in the heatmaps for certain combinations of estimation procedures, number of components, and internal knots is attributed to the failure of the algorithms to converge using the initially selected seeds. The analysis is then repeated for these specific combinations. For the 1-Step method, the minimum AIC obtained with 10 classes and 4 knots is 8059.50, while with 19 classes and 3 knots is 19486.26. The missing cell in the 2-Step procedure is the one with 9 components and 3 internal knots, having a AIC equal to 33822.10 when repeating the analysis with a different set of seeds. A comparison of these outcomes with adjacent values in the heatmap illustrates that they are numerically consistent, demonstrating how the model estimation may encounter problems with convergence for certain parameters, but the repetition of the analysis generally does not lead to bring highly different results. This consistency confirms the robustness of the model estimation process across varied parameter configurations.
The optimal combinations identified are 14 classes and 1 internal knot for the 1-Step estimation method, and 8 components and 4 internal knots for both the Stepwise procedures. An interesting observation emerges when these configurations are compared with those obtained with the “Leave” concomitant variable. Notably, the 1-Step procedure consistently selects the same number of latent classes and internal knots for both variables, demonstrating its robustness across different contexts. Conversely, for the 2 and 3-Step techniques, there is a shift from 13 classes with one knot under the “Leave” variable, to 8 classes with 4 knots in the “ExpEntropy” analysis.
Additionally, the 1-Step procedure yields a wide range of component-knot combinations that produce AIC values near the minimum. In contrast, the Stepwise estimations reveal a much narrower set of close combinations.

Figure 5.20: Heatmaps of AIC across MPs' Constituencies for Different Estimation Methods using "ExpEntropy" as Concomitant Variable

5.5.1 Results obtained using the 1-Step method

The following analysis will be executed using a Concomitant Variable Latent Class Model with 14 components, 1 internal knot, and utilizing the variable “ExpEntropy” serving as concomitant variable. The model is fit to the data, and used to predict the clusters of the Members of Parliament.

Predicted Groups	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Party														
C	9	2	60	149	3	17	5	33	21	3	0	10	1	0
DUP	10	0	0	0	0	0	0	0	0	0	0	0	0	0
GP	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Ind	0	12	0	0	1	0	1	0	0	0	0	0	1	1
LD	0	1	0	0	9	0	0	0	0	0	0	1	0	1
Lab	1	6	1	1	2	1	6	0	0	18	104	21	1	85
PC	0	0	0	0	4	0	0	0	0	0	0	0	0	0
SNP	0	0	0	0	34	0	0	0	0	0	0	0	1	0

Figure 5.21: Predicted Groups vs Actual Parties with 1-Step estimation and “ExpEntropy” as concomitant variable

As depicted in Figure 5.21, Group 3 is composed of 149 Conservative Party MPs out of 313. The cluster represents the same hard Brexit stances found in the previous models using the “Leave” variable, and picked as baseline. The same choice is taken in this context, since the CVLCM has distinguished easily this cluster in all the possible analyses.

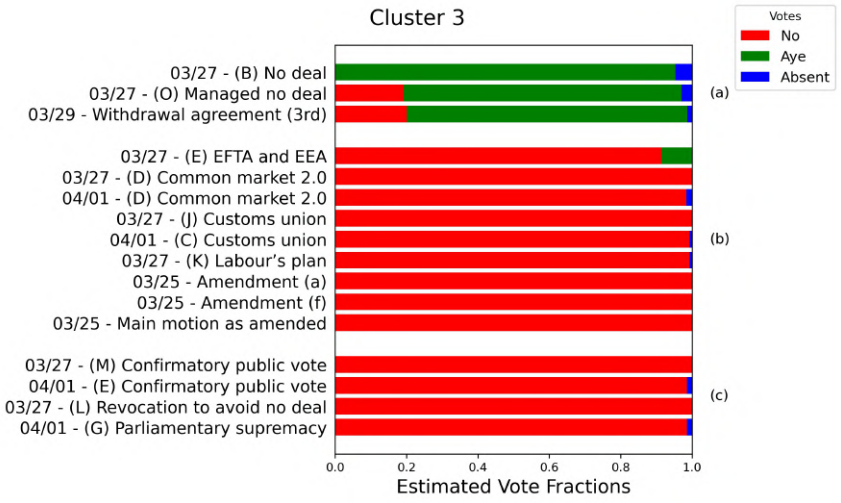


Figure 5.22: Clusters 3 with 1-Step estimation and "ExpEntropy" as concomitant variable

As illustrated in Figure 5.22, Cluster 3 contains in fact 144 Conservative MPs who supported the "No deal" vote on March 27, out of a total of 157. The remaining six MPs in the cluster chose to abstain from the vote. This insight seems to confirm how the 1-Step estimation procedure is capable of clearly distinguishing the hard Brexit stance inside of the Conservative Party, independently of the concomitant variable.

The majority of Conservative MPs has been put into Group 3, then into Cluster 2 (having 60 members, corresponding to the 19.17% of the party's MPs), and the remaining mostly in Groups 7 (10.54%), 8 (6.71%) and 5 (5.43%). Figures 5.23 and 5.24 contain respectively the estimated vote fractions of Clusters 2 and 5, and of Clusters 7 and 8. By first analysing the divisions concerning the possibility of a hard Brexit, Clusters 2, 5 and 8 contain MPs who voted against the No deal and the Managed no deal proposals. Cluster 7 members abstained in the two votes. Nevertheless, the four groups were all in favour of the third Withdrawal agreement, indicating a conditional alignment towards a negotiated exit from the EU.

Cluster 7 MPs appear to have abstained on most of the divisions concerning soft Brexit and pro Remain proposals. Distinguishing between Clusters 5 and 8 presents a more complex challenge, as their voting patterns on soft Brexit issues are quite similar. The main difference is found in the pro Remain divisions, where Cluster 5 is split between voting against or abstaining. Conversely, Cluster 8 is nearly unanimously against.

The abstention rate in Cluster 7, especially in all divisions except the third With-

drawal Agreement is easily explained, since it contains PM May and 70% of her ministry. Interpreting Clusters 5 and 8 poses challenges, similar to those encountered with the 1-Step estimation technique and the “Leave” variable. Cluster 2 appears to include Conservative MPs who are inclined towards a hard Brexit yet reject the extreme measures of the No Deal and Managed No Deal options, as evidenced by their unanimous opposition to soft Brexit proposals. This cluster’s voting behaviour underscores a preference for a decisive yet regulated withdrawal from the EU, reflecting the difficult balance MPs seek between party loyalty, constituency preferences, and personal beliefs (Aidt et al., 2021).

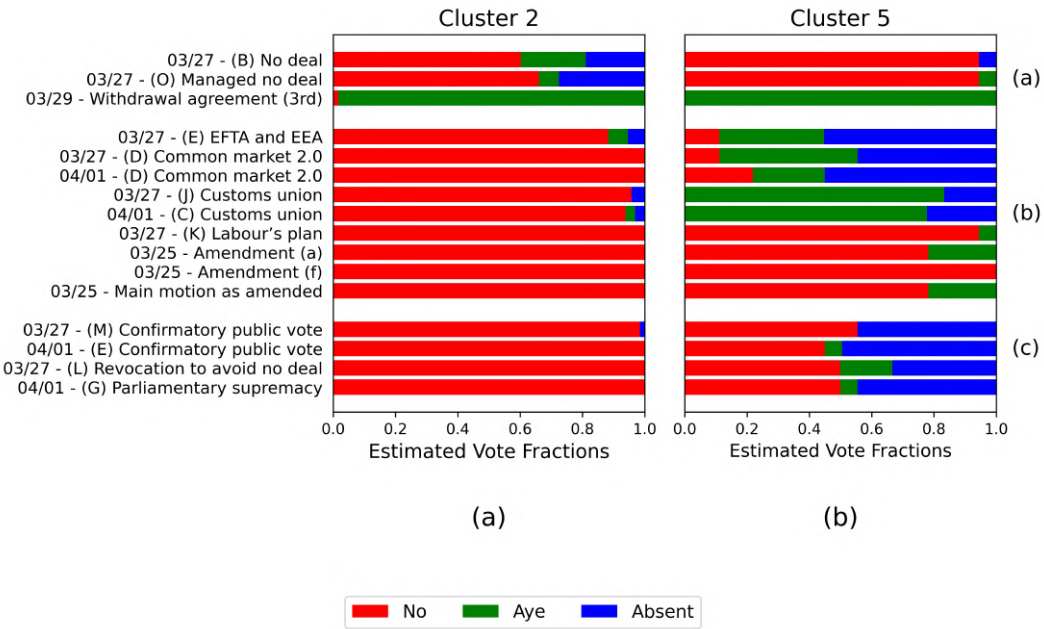


Figure 5.23: Clusters 2 and 5 with 1-Step estimation and “ExpEntropy” as concomitant variable

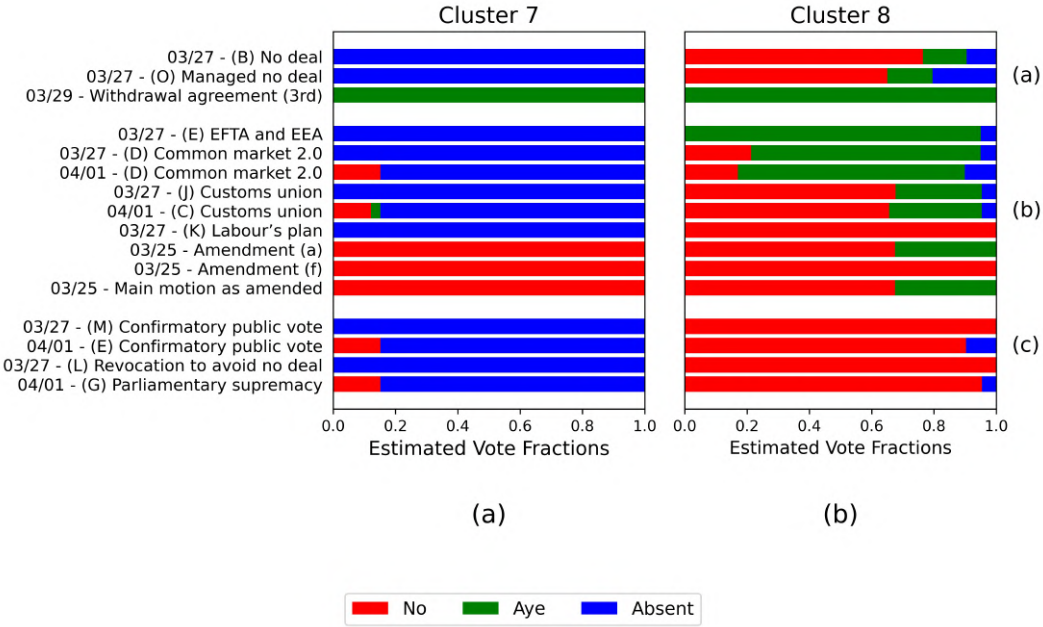


Figure 5.24: Clusters 7 and 8 with 1-Step estimation and "ExpEntropy" as concomitant variable

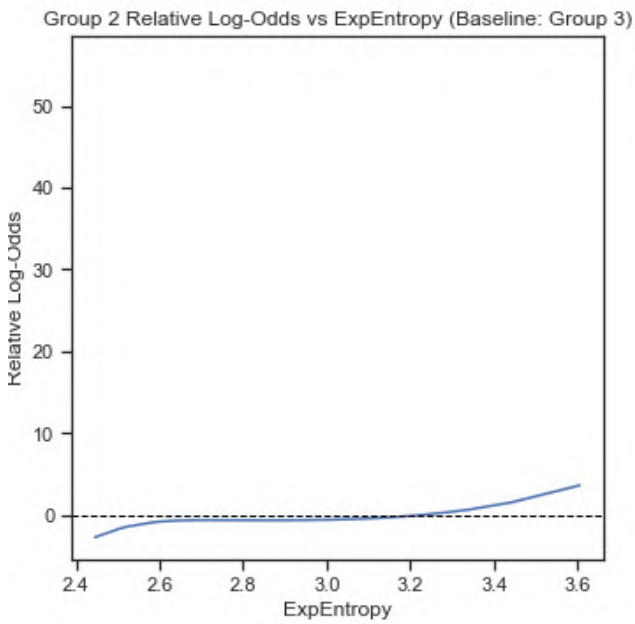


Figure 5.25: Relative Log-Odds in Cluster 2 (Cluster 3 as baseline)

The relative log-odds of Cluster 2, as depicted in Figure 5.25, explains an interesting dynamic within Cluster 2: the probability of belonging to the cluster increases as the exponential of the Entropy exceeds 3.2. This suggests that in constituencies where votes are more evenly distributed across multiple parties between parties, Conservative MPs find it challenging to unequivocally support a hard Brexit stance.

The clustering analysis of the Labour Party reveals a significant concentration of its MPs within Groups 10 and 13, holding 42.11% and 34.41% of the party’s representation, respectively, with an additional 7.29% falling into Cluster 9. The voting behaviours are represented in Figure 5.26. The three Labour clusters are all against the hard Brexit divisions.

When examining their stance on soft Brexit proposals, both Clusters 10 and 13 show support for the majority, favoring eight out of the nine proposals presented. This demonstrates a clear preference for a softer approach to Brexit within these groups. In contrast, Cluster 9, while still mostly supportive, favors only six out of the nine soft Brexit proposals.

Concerning the positions on pro Remain stances, Cluster 10 unanimously supports them, reflecting a strong preference against leaving the EU without a deal. Cluster 13 shows a more complex view, abstaining from two motions aimed at preventing a no-deal exit. This may suggest that while members of Cluster 13 lean towards a soft Brexit, they might not be as opposed to a hard Brexit as the MPs in the other clusters. On the other hand, Cluster 9 is distinctly opposed to the idea of remaining in the EU.

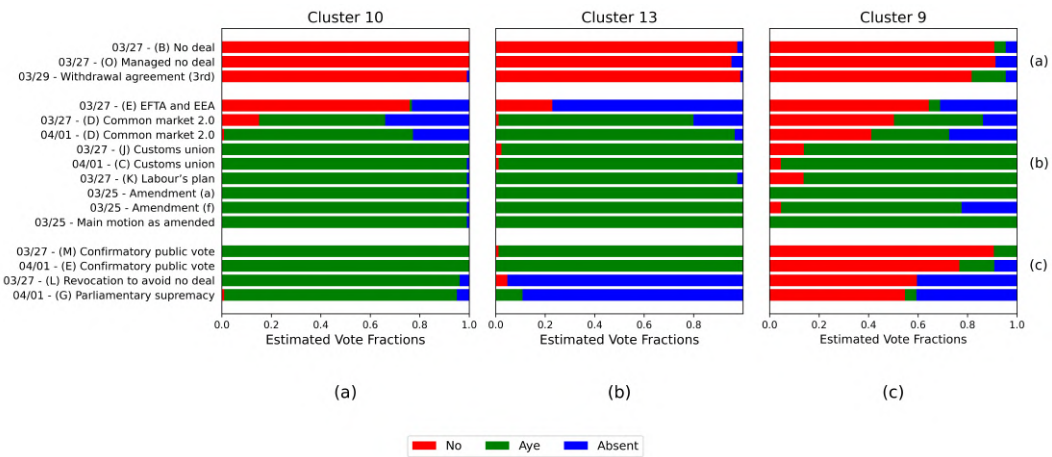


Figure 5.26: Clusters 10, 13 and 9 with 1-Step estimation and “ExpEntropy” as concomitant variable

Focusing on the remaining groups, the composition of Cluster 11 is interesting, since it is composed of Conservative MPs by 31.25% and Labour MPs by 65.63%. This peculiar group was not found in the analysis made with the "Leave" variable. Analysing the estimated vote fractions, the MPs seem to be in favour of a soft Brexit, but when the shift focuses on hard Brexit's Withdrawal agreement and pro Remain divisions, the fractions seem to be split, making a possible interpretation difficult. Cluster 0 contains the totality of DUP members, along with nine Conservative MPs.

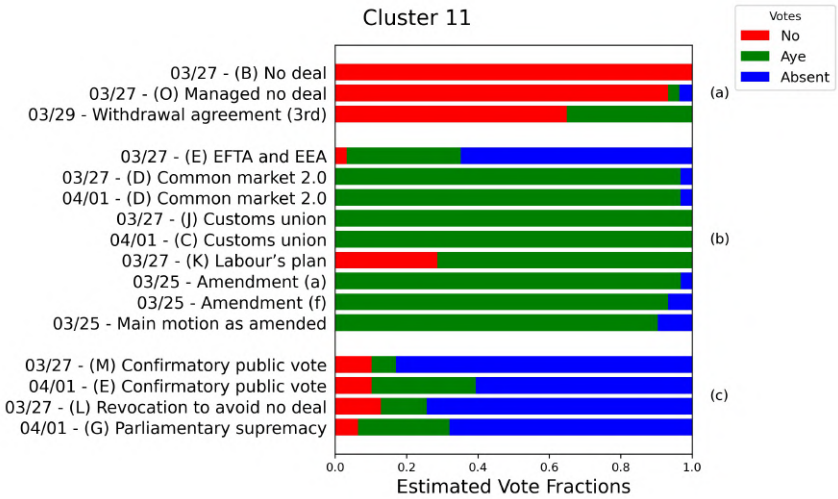


Figure 5.27: Cluster 11 with 1-Step estimation and "ExpEntropy" as concomitant variable

Cluster 1 is distinguished by including the most Independents in it. Cluster 4 predominantly has most of LD and SNP MPs, and the totality of PC members. Cluster 6 is a small group, containing only twelve individuals, half Conservative and half part of Labour Party; it includes four out of the total six Labour MPs who voted yes for the Withdrawal agreement of March 29. Cluster 12 comprises only four MPs, each from four different parties; given its limited size and the diversity of its members, this cluster does not present a coherent stance on Brexit and may not be particularly informative for analysis purposes.

5.5.2 Results obtained using the 2-Step and 3-Step methods

The 2-Step and 3-Step estimation procedures, when applying a CVLCM to the data with "ExpEntropy" used as concomitant variable, lead to completely identical clustering. The analysis will employ 8 latent classes and 4 internal knots. This is the only case in which the model selection leads to a number of knots different than

one. Adding knots leads to an increase in flexibility, allowing the B-splines to depart more sensibly from linearity. This leads to capturing more complex patterns in the data.

Predicted Groups	0	1	2	3	4	5	6	7
Party								
C	17	33	3	213	0	41	3	3
DUP	0	0	0	10	0	0	0	0
GP	0	0	1	0	0	0	0	0
Ind	0	0	12	1	0	0	2	1
LD	0	0	1	0	0	0	9	2
Lab	20	1	6	5	111	4	2	98
PC	0	0	0	0	0	0	4	0
SNP	0	1	0	0	0	0	34	0

Figure 5.28: Predicted Groups vs Actual Parties with Stepwise estimation and “ExpEntropy” as concomitant variable

Figure 5.28 shows the composition of the predicted groups. Group 3 is composed of 68.51% of Conservative Party MPs. The cluster represents the same hard Brexit stances found in the previous models, but in this case the fraction of votes against the corresponding hard Brexit divisions are higher than the clusters selected as baseline in the previous analyses. This is to be expected since the cluster contains around sixty more Conservative MPs, and the totality of DUP members.

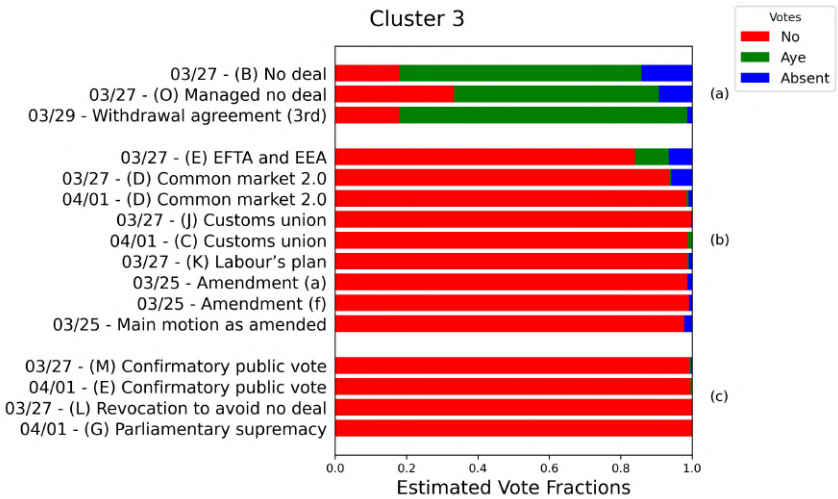


Figure 5.29: Clusters 3 with Stepwise estimation and “ExpEntropy” as concomitant variable

Other clusters containing the remaining Conservative MPs are Groups 5 and 1, respectively composed of 13.10% and 10.54%. Figure 5.30 contains the estimated vote fractions of the two groups.
Cluster 5 includes Conservative MPs who predominantly oppose to the idea of remaining in the EU, but with a response to soft Brexit proposals that is mixed, with vote fractions indicating division within the group. The total opposition to No deal and Managed No deal proposals by MPs in this cluster suggests a reluctance to endorse a hard Brexit. This voting pattern complicates the interpretation of this group's overall position.
Cluster 1 is composed of Conservative MPs who decided to abstain or reject the soft Brexit divisions, and abstained in pro Remain votes, in the No deal and Managed no deal divisions, but ultimately voted in favour of the Withdrawal agreement of March 29.

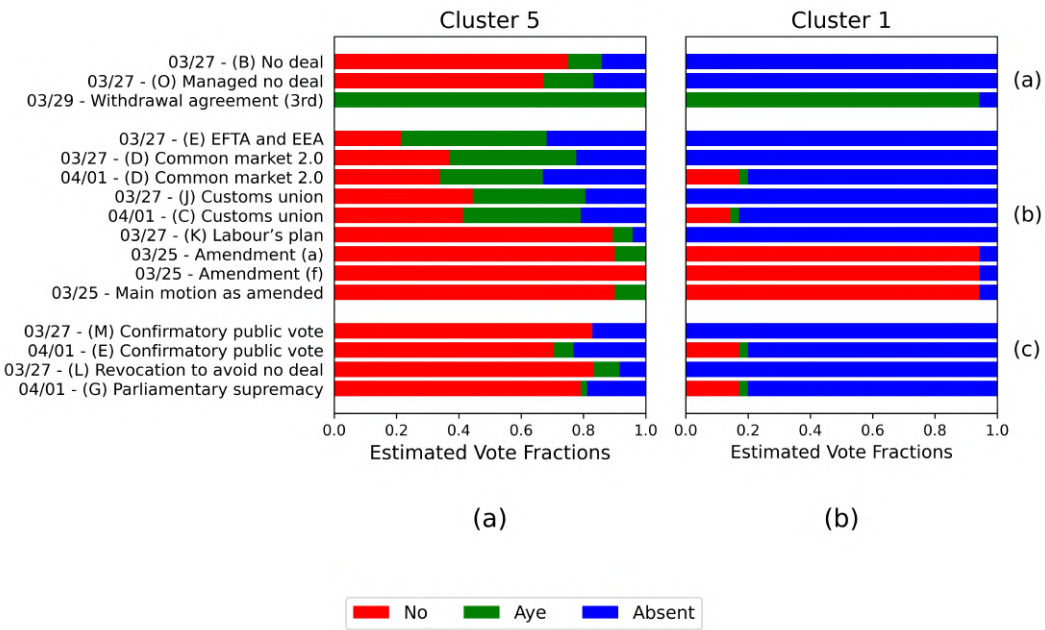


Figure 5.30: Clusters 5 and 1 with Stepwise estimation and "ExpEntropy" as concomitant variable

The clustering of the Labour Party seems similar to what was observed in models employing a Stepwise procedure and the "Leave" concomitant variable. Clusters 4, 7 and 0 include respectively 44.94%, 39.68% and 8.10% of the party. The voting behaviours are represented in Figure 5.31. The three Labour clusters are all against the hard Brexit divisions. Cluster 0 appears to be less informative due to its mixed composition, consisting equally of Conservative and Labour MPs. This dilutes the

cluster’s ability to represent a clear stance on Brexit, particularly evident in the split decision over the Withdrawal Agreement, where only one Labour MP supported it, and the rest voted against. This diversity within the cluster makes it challenging to derive a coherent interpretation of Brexit preferences from this group.
Conversely, Cluster 4 is characterized by Labour MPs who are uniformly opposed to a hard Brexit, while showing strong support for soft Brexit options and pro-Remain positions. This cluster’s voting behaviour clearly delineates a group within the Labour Party that is firmly against leaving the EU without a comprehensive deal. Cluster 7, while also unanimously opposing a hard Brexit, shares much of Cluster 4’s perspective on soft Brexit, except for a divergence in their approach to the EFTA and EEA division, where Cluster 7 chose to abstain. Considering this factor and the abstention from votes aimed at preventing an EU exit without a deal, this signals a more moderate approach towards opposing Brexit.

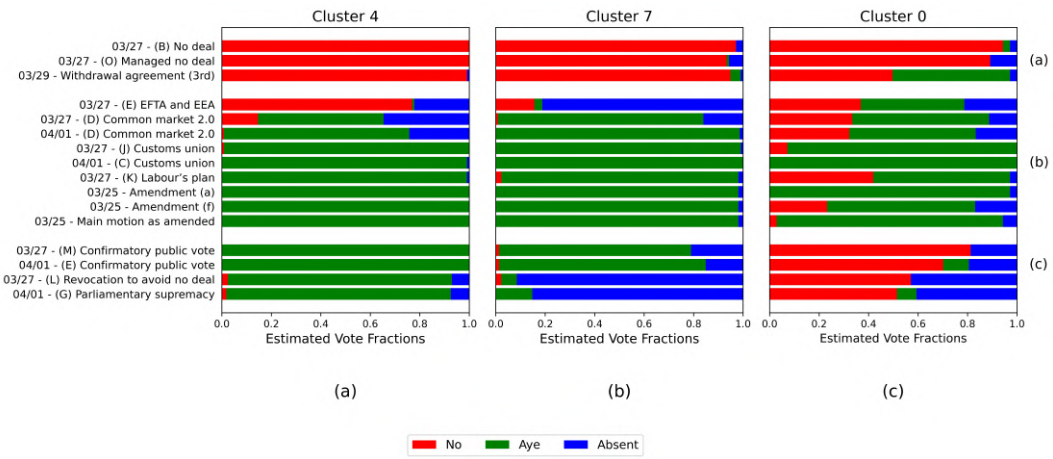


Figure 5.31: Clusters 4, 7 and 0 with Stepwise estimation and “ExpEntropy” as concomitant variable

Turning the attention to the remaining clusters, Cluster 2 predominantly comprises Independent Members of Parliament along with MPs from various parties. Meanwhile, Cluster 6 is made up of most of the Members from SNP and LD, and it includes all representatives from PC. It is a cluster with strong pro Remain views.

5.6 Clustering Methods and Concomitant Variables Comparison

5.6.1 The Adjusted Rand Index

In order to quantify the similarity between clustering results, obtained from the application of the Concomitant Variable Latent Class Model using "Leave" and "ExpEntropy" as variables, the Adjusted Rand Index (ARI) is calculated. It provides a normalized measure of the similarity between two clustering solutions (Rand, 1971), adjusting for the chance grouping of elements. It is an enhanced version of the Rand Index, which simply calculates the percentage of pairing of elements in clusters that are correct (Hubert and Arabie, 1985).
The formula for ARI is given by:

$$ARI = \frac{\sum_{ij} \binom{n_{ij}}{2} - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}]/\binom{n}{2}}. \tag{5.6.1}$$

The numerator represents the difference between the actual number of pair agreements and the expected number of pair agreements by chance. The denominator ensures that the ARI ranges between -1 and 1, normalizing the index to correct for chance agreement. A score of 0 indicates a random clustering agreement, and a score of 1 indicates a perfect agreement.
In the formula, n_{ij} represents the number of elements in common between cluster i of the first clustering and cluster j of the second clustering. a_i is the number of observations in cluster i of the first clustering, and b_j is the number of observations in cluster j of the second clustering. n is the total number of elements.
$\sum_{ij} \binom{n_{ij}}{2}$ calculates the sum of the combinations of pairs of elements that are in the same group in both clusterings. The binomial coefficient $\binom{n_{ij}}{2}$ is the number of possible ways to choose two elements from n_{ij}, indicating the number of element pairs that are correctly grouped together in both clusterings. $\sum_i \binom{a_i}{2}$ and $\sum_j \binom{b_j}{2}$ calculate the sum of the combinations of pairs within each cluster for each clustering.
$\binom{n}{2}$ represents the total number of possible pairs of elements given n elements, serving as the normalization factor for the expected number of chance agreements between the two clusterings (Hubert and Arabie, 1985).

5.6.2 Comparative Analysis within "Leave" Clustering

The ARI scores from comparing "Leave" clustering across different estimation methods reveal strong consistencies. Specifically, the 1-Step method compared to the 2-Step and 3-Step methods leads to scores of 0.870 and 0.864, respectively. This suggests a high degree of similarity in the clusterings, particularly evident in the categorization of smaller parties and Conservative Members of Parliament. Additionally, the content of nine out of thirteen clusters in the 2-Step and 3-Step find a correspondence with groups obtained with the 1-Step technique.
The Stepwise procedures fail to identify a cluster of Labour MPs fully abstaining from pro-Remain votes, as was instead observed in the 1-Step clustering (refer to Figure D.11a in the Appendix). However, both the 2-Step and 3-Step methods manage to cluster Conservative MPs expressing preferences for a soft Brexit (see Figures D.23b and D.36b in the Appendix), while this was not present in the 1-Step results. The exceptionally high ARI of 0.991 between the 2-Step and 3-Step methods is primarily due to the identical clustering of all but two MPs.
Comparisons with actual party affiliations result in ARI scores of 0.32 for the 1-Step method and 0.35 for the Stepwise methods. These scores reveal the models' capability to dissect the major parties based on internal Brexit viewpoints, despite a limited correlation with formal party structures.

5.6.3 Comparative Analysis within "ExpEntropy" Clustering

The "ExpEntropy" variable analysis exhibits a perfect ARI score of 1 between the Stepwise methods, indicating identical clusterings. However, when the 2-Step and 3-Step are compared to the 1-Step method, the ARI score decreases to 0.658. This reduction reflects the reduction in the selected number of latent classes (14 for 1-Step vs. 8 for Stepwise). Furthermore, only three groups from the 2-Step and 3-Step methods show a clear correspondence with the content of the 1-Step groups.
Specifically, clusters comprising Conservative MPs abstaining in pro-Brexit divisions, Independent MPs, and a group including SNP, PC, and LD members are identifiable in all three models. Surprisingly, a cluster representing Conservative MPs with No deal views is absent in the Stepwise clustering when utilizing "ExpEntropy", which is instead found in all the other CVLCMs.
Comparing the "ExpEntropy" clusters to party affiliations results in scores of 0.321 for the 1-Step method, and 0.439 for the Stepwise methods. The higher scores for Stepwise methods are attributed to the CVLCM's selection of eight components,

equal to the number of parties analyzed, and its limitations to differentiate within parties.

5.6.4 Inter-Variable Comparative Analysis

The inter-variable comparison reveals an ARI of 0.88 for the 1-Step method models, with clear similarities in the interpretation of eight out of fourteen groups.
This is less pronounced in the Stepwise methods, with “ExpEntropy” clustering less capable of uncovering contrasting within-party views. This results in clustering Conservative MPs in three groups instead of the seven found with “Leave”, and Labour MPs in two instead of three. The “ExpEntropy” Stepwise models are not capable of differentiating the Labour members in favour, against or abstaining to pro Remain divisions. Nevertheless, the ARI scores for the 2-Step and 3-Step are respectively 0.713 and 0.72.
The analysis demonstrates that the “Leave” vote percentages in MPs’ constituencies are more useful at capturing the views within each party, compared to the exponential of the entropy. Although “ExpEntropy” distinguishes between parties, it fails to capture internal nuances, leading to clusters with ambiguous interpretations.

Chapter 6

Conclusions

The objective of this thesis was to conduct a comparative analysis of various procedures for estimating parameters in latent class models with concomitant variables. Through detailed examination, the thesis explored the efficacy of 1-Step, 2-Step, and 3-Step estimation methods within the framework of flexible models, particularly focusing on the incorporation of B-spline functions to represent the covariate effect. This exploration exposed the applicabilities of each estimation method, offering insights into their strengths and limitations in modeling complex relationships within data.

The simulation study, divided into two main parts, focused on the impact of method selection, sample size, class separation, and the choice of internal knots on the estimation accuracy and model performance. Initially, the study has emphasized the fluctuation of the 2.5% and 97.5% percentiles of $\hat{f}_2$ with varying internal knots, stabilizing with a fixed knot count, particularly at domain margins. Larger sample sizes and higher class separations enhance percentile stability and reduce fluctuations, narrowing percentile intervals. Among the estimation methods, the 1-Step approach exhibits larger percentiles compared to the more refined 2-Step and 3-Step methods, with the latter slightly outperforming the former in terms of interval narrowness.

The analysis reveals that poor class separation ($\theta \leq 0.7$) increases RMSE values, with the 2-Step method showing superior performance under such conditions. However, differences between methods diminish as sample size increases, although the 3-Step method slightly leads in RMSE values. The estimation techniques have similar performance with larger sample sizes and greater class separation, highlighting the overall improvement in accuracy.

Regarding the estimation function's roughness, the 1-Step method results in rougher estimates compared to the smoother estimates produced by the 3-Step method, with the 2-Step method yielding intermediate outcomes. The disparity between the 1-Step and 2-Step methods lessens with increased sample sizes, whereas the 3-Step method consistently presents smoother function estimates regardless of class separation levels.

The second part of the simulation study assesses the efficacy of different estimation methods in accurately identifying the number of latent classes, revealing the 1-Step method's limitations under small sample sizes and low class separation. In contrast, Stepwise methods significantly enhance performance through an intermediate refinement step, particularly the 3-Step method, which excels with larger sample sizes and greater class separation.

The empirical application to UK Members of Parliament's voting behavior illustrates the modeling approach's capability to distinguish between party lines and internal divisions, with the model being strongly influenced by the choice of the concomitant variable. The "Leave" variable demonstrates consistency across estimation methods in identifying Brexit-related viewpoints, while the "ExpEntropy" variable highlights the 1-Step method's relative advantage in capturing within-party variations compared to Stepwise techniques.

A potential continuation of this thesis could involve comparing the estimation methods with an emphasis on the variability of the estimates. Specifically, the software utilized for simulations and analyses does not provide estimates for the variability of the maximum likelihood estimators. This aspect could be explored in future research activities.

Acknowledgements

Desidero esprimere il mio sincero ringraziamento al mio relatore, il Prof. Giuliano Galimberti, ed al mio correlatore, il Dott. Marco Berrettini, per il loro indispensabile sostegno nella stesura della tesi. La loro guida insostituibile è stata fondamentale per il raggiungimento degli obiettivi di questo progetto.

Bibliography

T. Aidt, F. Grey, and A. Savu. The meaningful votes: Voting on brexit in the british house of commons. *Public Choice*, 186(3):587–617, Mar 2021. ISSN 1573-7101. doi: 10.1007/s11127-019-00762-9. URL https://doi.org/10.1007/s11127-019-00762-9.

V. Apostolova, L. Audickas, C. Baker, A. Bate, R. Cracknell, N. Dempsey, O. Hawkins, R. McInnes, T. Rutherford, and E. Uberoi. General election 2017: Results and analysis. Briefing Paper CBP 7979, House of Commons Library, 2017.

Z. Bakk and J. Kuha. Two-step estimation of models between latent classes and external variables. *Psychometrika*, 83:871–892, 2018. doi: 10.1007/s11336-017-9592-7.

K. Bandeen-Roche, D. L. Miglioretti, S. L. Zeger, and P. J. Rathouz. Latent variable regression for multiple discrete outcomes. *Journal of the American Statistical Association*, 92(440):1375–1386, 1997. ISSN 01621459. URL http://www.jstor.org/stable/2965407.

M. Berrettini, G. Galimberti, S. Ranciati, and T. B. Murphy. Identifying brexit voting patterns in the british house of commons: An analysis based on bayesian mixture models with flexible concomitant covariate effects. *Journal of the Royal Statistical Society - Series C*, To Appear, 2024.

A. Bolck, M. Croon, and J. Hagenaars. Estimating latent structure models with categorical variables: One-step versus three-step estimators. *Political Analysis*, 12(1):3–27, 2004. ISSN 10471987, 14764989. URL http://www.jstor.org/stable/25791751.

C. M. Dayton and G. B. Macready. Concomitant-variable latent-class models.

Journal of the American Statistical Association, 83(401):173–178, 1988. ISSN 01621459.

C. de Boor. On calculating with b-splines. *Journal of Approximation Theory*, 6(1):50–62, 1972. ISSN 0021-9045. doi: https://doi.org/10.1016/0021-9045(72)90080-9. URL `https://www.sciencedirect.com/science/article/pii/0021904572900809`.

C. de Boor. *A Practical Guide to Splines*. Springer-Verlag, New York, revised edition, 2001. ISBN 978-0387953663.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. ISSN 00359246. URL `http://www.jstor.org/stable/2984875`.

P. H. C. Eilers and B. D. Marx. Flexible smoothing with B-splines and penalties. *Statistical Science*, 11(2):89 – 121, 1996. doi: 10.1214/ss/1038425655. URL `https://doi.org/10.1214/ss/1038425655`.

I. Moustaki G. A. Marcoulides. *Latent Variable and Latent Structure Models*. Quantitative Methodology Series. Psychology Press, illustrated edition edition, 2002. ISBN 080584046X; 9780805840469; 9780585418391; 058541839X.

L. A. Goodman. Exploratory latent structure analysis using both identifiable and unidentifiable models. *Biometrika vol. 61 iss. 2*, 61, 1974. doi: 10.1093/biomet/61.2.215.

S. J. Haberman. Log-Linear Models for Frequency Tables Derived by Indirect Observation: Maximum Likelihood Equations. *The Annals of Statistics*, 2(5):911 – 924, 1974. doi: 10.1214/aos/1176342813. URL `https://doi.org/10.1214/aos/1176342813`.

J. Hagenaars. *Loglinear Models with Latent Variables*. SAGE Publications, Inc., Thousand Oaks, California, 1993. doi: 10.4135/9781412984850. URL `https://methods.sagepub.com/book/loglinear-models-with-latent-variables`.

J. A. Hagenaars and A. L. McCutcheon. *Applied Latent Class Analysis*. Cambridge University Press, 1 edition, 2002. ISBN 0521594510; 9780521594516.

C. Hanretty. Areal interpolation and the uk's referendum on eu membership. *Journal of Elections, Public Opinion and Parties*, 27 (4):466–483, 2017. doi: 10.1080/17457289.2017.1287081. URL https://doi.org/10.1080/17457289.2017.1287081.

T. Hastie and R. Tibshirani. *Generalized Additive Models*. Chapman Hall/CRC, 1990.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, Stanford, CA, 2 edition, 2009. ISBN 978-0-387-84857-0. doi: 10.1007/978-0-387-84858-7. URL https://web.stanford.edu/ hastie/Papers/ESLII.pdf.

L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2 (1):193–218, 1985. ISSN 1432-1343. doi: 10.1007/BF01908075. URL https://doi.org/10.1007/BF01908075.

D. A. Linzer and J. B. Lewis. polca: An r package for polytomous variable latent class analysis. *Journal of Statistical Software*, 42(10), 2011.

S. Morin, R. Legault, Z. Bakk, C. Giguère, R. de la Sablonnière, and E. Lacourse. Stepmix: A python package for pseudo-likelihood estimation of generalized mixture models with external variables. 04 2023.

D. S. Nagin. *Group-Based Modeling of Development*. Harvard University Press, 2005.

E. Odell. *hansard: Provides Easy Downloading Capabilities for the UK Parliament API*, 2017. URL https://cran.r-project.org/package=hansard. R package version 0.8.0.

W.M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971. ISSN 01621459. URL http://www.jstor.org/stable/2284239.

J. Rost. Rasch models in latent classes: An integration of two approaches to item analysis. *Applied Psychological Measurement*, 14 (3):271–282, 1990. doi: 10.1177/014662169001400305. URL https://doi.org/10.1177/014662169001400305.

I. J. Schoenberg. Contribution to the problem of approximation of equidistant data by analytic functions. *Quarterly of Applied Mathematics*, 4(1):45–99, 112–141, 1946.

A. Spirling and K. Quinn. Identifying intraparty voting blocs in the u.k. house of commons. *Journal of the American Statistical Association*, 105(490):447–457, 2010. ISSN 01621459. URL `http://www.jstor.org/stable/29747055`.

J. K. Vermunt. Latent class modeling with covariates: Two improved three-step approaches. *Political Analysis*, 18(4):450–469, 2010. ISSN 10471987, 14764989. URL `http://www.jstor.org/stable/25792024`.

P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C J Carey, İ Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Appendix A

Simulation Study Code

A.1 Data Generation

```
def rmulti(p):
    n = p.shape[0]
    p = p[:, :-1]
    thresh = np.cumsum(p, axis=1)
    vals = np.random.rand(n, 1)
    return np.sum(vals > thresh, axis=1) + 1



def gendati(n=2000, seed=303, teta=0.90,
    num_manifest_variables=6):
    np.random.seed(seed)
    f2 = lambda x: np.sin(3*np.pi*x)*np.exp(-x)
    x = np.random.uniform(size=n)
    eta2 = f2(x)

    p = np.exp(np.column_stack((np.zeros(n), eta2)))
    p = p / np.sum(p, axis=1, keepdims=True)
    group = rmulti(p)

    swap_probs = np.random.randint(0, 2,
        num_manifest_variables)
```

```
    probs = []
    for j in range(num_manifest_variables):
        if swap_probs[j] == 0:
            probs.append(np.array([[teta, 1-teta],
            [1-teta, teta]]))
        else:
            probs.append(np.array([[1-teta, teta],
            [teta, 1-teta]]))

    y = [rmulti(probs[j][group-1, :]) - 1
        for j in range(num_manifest_variables)]
    y = np.column_stack(y)

    y_df = pd.DataFrame(y, columns=[f'Y{j+1}'
        for j in range(num_manifest_variables)])
    x_df = pd.DataFrame(x, columns=['x'])
    group_df = pd.DataFrame(group, columns=['group'])

    dati = pd.concat([y_df, x_df, group_df], axis=1)
    dati['group'] = -dati['group'] +2
    return dati
```

A.2 B-spline Generation in Simulation Study

```
def B(x, degree, i, t):
    if x == 1.0:
        return 1.0 if i == len(t) - degree - 2 else 0.0
    if degree == 0:
        return 1.0 if t[i] <= x < t[i+1] else 0.0
    if t[i+degree] == t[i]:
        c1 = 0.0
    else:
        c1 = (x - t[i])/(t[i+degree] - t[i]) *
            B(x, degree-1, i, t)
    if t[i+degree+1] == t[i+1]:
        c2 = 0.0
    else:
```

```
        c2 = (t[i+degree+1] - x)/(t[i+degree+1] - t[i+1]) *
            B(x, degree-1, i+1, t)
    return c1 + c2


def bspline_bases(x, t, degree):
    n = len(t) - degree - 1
    return [B(x, degree, i, t) for i in range(n)]


def add_bspline_columns(data, x_colname, t, degree):
    bases_values = [bspline_bases(x, t, degree)
        for x in data[x_colname]]
    bases_df = pd.DataFrame(bases_values, columns=[f"B{i+1}"
        for i in range(len(t) - degree - 1)])
    result_df = pd.concat([data, bases_df], axis=1)
    return result_df
```

A.3 Creation of the Model for a Fixed Number of Latent Classes

```
def evaluate_methods_integral(covariate_params, sample_sizes,
    teta_values, knots_list, n_iterations=100,
    num_manifest_variables=10, save_plots=True,
    output_dir='.'):
results = []
x_values = np.linspace(0, 1, 333)

def create_models(seed):
    return {
        '1-STEP': StepMix(n_components=Groups_Number,
            n_steps=1,
            measurement="binary",
            structural="covariate", verbose=0,
            random_state=seed,
            structural_params=covariate_params,
            max_iter=100),
```

```
        '2-STEP': StepMix(n_components=Groups_Number,
            n_steps=2,
            measurement="binary",
            structural="covariate", verbose=0,
            random_state=seed,
            structural_params=covariate_params,
            max_iter=100),

        '3-STEP': StepMix(n_components=Groups_Number,
            n_steps=3,
            measurement="binary",
            structural="covariate", verbose=0,
            random_state=seed,
            structural_params=covariate_params,
            max_iter=100)
    }
if save_plots:
    pdf_filename = f"
    {output_dir}/All_Plots_integral_labelswitch.pdf"
    pdf_pages = PdfPages(pdf_filename)


global_min_f2_value = -2.5
global_max_f2_value = 2.5

for teta in teta_values:
    for method_name, model in create_models(0).items():
        for sample_size in sample_sizes:
            aics = []
            bics = []
            rmses = []
            estimated_f2_values_per_all_iterations = []
            best_knots_counter = {k: 0 for k in knots_list}
            best_knots_counter_bic = {k: 0 for k in knots_list}
            roughness_penalties = []

            for iteration in range(n_iterations):
```

```
best_aic = float('inf')
best_bic = float('inf')
best_knot_bic = None
best_estimated_f2_values_for_iteration = None
best_knot = None
dati = gendati(n=2000, seed=iteration,
    teta=teta,
    num_manifest_variables=
        num_manifest_variables)

for knot in knots_list:
    t = np.linspace(0, 1, knot - 2)
    t = np.pad(t, pad_width=3,
        mode='constant',
        constant_values=(0, 1))

    dati_spline = add_bspline_columns(dati,
        'x', t,
        degree)
    fit = model.fit(X=dati_spline.iloc[:
        sample_size,
        0:num_manifest_variables],
        Y=dati_spline.iloc[:sample_size,
        num_manifest_variables+2:])
    aic = fit.aic(X=dati_spline.iloc[:
        sample_size,
        0:num_manifest_variables],
        Y=dati_spline.iloc[:
        sample_size,
        num_manifest_variables+2:])
    bic = fit.bic(X=dati_spline.iloc[:
        sample_size,
        0:num_manifest_variables],
    Y=dati_spline.iloc[:sample_size,
    num_manifest_variables+2:])

    if aic < best_aic:
        best_aic = aic
```

```
                best_knot = knot
                betas = model.
                get_parameters()
                ["structural"]
                    ["beta"]
                betas = betas.ravel()[:knot]
                pred = fit.predict(X=
                dati_spline.iloc[
                    :sample_size,
                    0:num_manifest_variables],
                    Y=dati_spline.iloc[
                    :sample_size,
                    num_manifest_variables+2:])
                correct_proportion =
                    np.sum(dati_spline.
                iloc[:sample_size,]
                    ['group'] == pred)
                    / len(pred)

                corrected_betas =
                correct_coefficients(
                    betas, correct_proportion)
                best_estimated_f2_values
                    _for_iteration =
                    estimate_f2(x_values,
                        corrected_betas,
                        t, degree)
                rmse = calculate_rmse(x_values,
                    t, corrected_betas, degree)
            if bic < best_bic:
                best_bic = bic
                best_knot_bic = knot

        if best_knot:
            best_knots_counter[best_knot] += 1
            best_knots_counter_bic[best_knot_bic] += 1

        integral_d2_f, roughness_penalty =
```

```
            integrate_second_derivative(
                best_estimated_f2_values_for_iteration,
                x_values)
        roughness_penalties.append(roughness_penalty)
        estimated_f2_values_per_all_iterations.append(
            best_estimated_f2_values_for_iteration)
        aics.append(best_aic)
        bics.append(bic)
        rmses.append(rmse)

    estimated_f2_values_array = np.array(
        estimated_f2_values_per_all_iterations)
    average_estimated_f2_values = np.mean(
        estimated_f2_values_array, axis=0)
    ci_lower = np.percentile(
        estimated_f2_values_array, 2.5, axis=0)
    ci_upper = np.percentile(
        estimated_f2_values_array, 97.5, axis=0)


    global_min_f2_value = min(global_min_f2_value,
        ci_lower.min())
    global_max_f2_value = max(global_max_f2_value,
        ci_upper.max())

    if save_plots:
        plt.figure(figsize=(10, 8))
        plt.plot(x_values, f2(x_values), label=
        'True Value of $f_2$', color='blue')
        plt.plot(x_values, average_estimated_f2_values,
            label='Average Estimate of $f_2$',
            color='red', linestyle='--')
        plt.fill_between(x_values, ci_lower, ci_upper,
            color='red', alpha=0.2, label=
            '2.5-97.5 Percentiles')

        knot_counts_str = ", ".join([f"{k-4}:
            {best_knots_counter[k]}" for k in
```

```
            sorted(best_knots_counter.keys())])
        knot_counts_str_bic = ", ".join([f"{k-4}:
            {best_knots_counter_bic[k]}" for k in
            sorted(best_knots_counter_bic.keys())])

        plt.title(f"Method: {method_name},
            Theta: {teta},
            Sample Size: {sample_size}\n",
            fontsize = 16)
        plt.xlabel('x', fontsize = 14)
        plt.ylabel('$f_2$ Values',
        fontsize = 14)
        plt.ylim([global_min_f2_value,
            global_max_f2_value])
        plt.legend(fontsize = 14)
        plt.grid(True)
        dpi_value = 300
        jpeg_filename = f"{output_dir}/
            Method_{method_name}_Theta_{teta}
            _SampleSize_{sample_size}.jpg"

        plt.savefig(jpeg_filename, format='jpg')
        pdf_pages.savefig()
        plt.close()

    results.append({
        'Method': method_name,
        'Theta': teta,
        'Sample Size': sample_size,
        'Knots Choices (AIC)': knot_counts_str,
        'Knots Choices (BIC)': knot_counts_str_bic,
        'RMSE': [round(r, 3) for r in rmses],
        'Average RMSE': round(np.mean(rmses), 3),
        'AIC': [round(a, 3) for a in aics],
        'Average AIC': round(np.mean(aics), 3),
        'BIC': [round(b, 3) for b in bics],
        'Average BIC': round(np.mean(bics), 3),
        'Roughness Penalty': [round(rp, 3)
```

```
                    for rp in roughness_penalties],
                'Average Roughness Penalty': round(np.mean(
                    roughness_penalties), 3)
            })
if save_plots:
    pdf_pages.close()

results_df = pd.DataFrame(results).sort_values(by=['Method',
    'Theta', 'Sample Size'])
return results_df
```

A.3.1 Dealing with Label Switching

```
def correct_coefficients(coefficients, correct_proportion):
    if correct_proportion < 0.5:
        corrected_coefficients = [-coef for coef in
            coefficients]
        return corrected_coefficients
    return coefficients
```

A.3.2 Creation of f_2

```
def f2(x):
    return np.sin(3*np.pi*x)*np.exp(-x)
```

A.3.3 Estimating f_2 with B-splines

```
def estimate_f2(x_values, coefficients, t, degree):
    return [bspline(x, t,coefficients, degree)
        for x in x_values]
```

A.3.4 Calculating Root Mean Square Error

```
def calculate_rmse(x_values, t, coefficients, degree):
    true_values = f2(x_values)
    estimated_values =
```

```
    estimate_f2(x_values, coefficients, t, degree)
    squared_errors = [(true - estimated)**2 for true,
        estimated in zip(true_values, estimated_values)]
    mean_square_error = sum(squared_errors) /
        len(squared_errors)

    return mean_square_error**0.5
```

A.3.5 Calculating the Roughness Measure

```
def integrate_second_derivative(f_values, x_values):
    h = x_values[1] - x_values[0]
    d2_f_values = np.zeros_like(f_values)
    for i in range(1, len(f_values) - 1):
        d2_f_values[i] = (f_values[i - 1] - 2 * f_values[i] +
        f_values[i + 1]) / h**2

    sel = np.arange(0, len(x_values) - 1, 2)

    integral_d2_f = np.sum(2 * h * d2_f_values[sel])
    integral_d2_f_squared = np.sum(2 * h * d2_f_values[sel]**2)

    return integral_d2_f, integral_d2_f_squared
```

Appendix B

Brexit Data Code

B.1 Choice of Latent Classes' Number

```
def evaluateLC(data, x_colname, knots_list, seed_start,
    seed_end, min_comp, max_comp):

    def create_models(n_components, seed):
        return {
            '1-STEP': StepMix(n_components=n_components,
                n_steps=1, measurement="categorical",
                structural="covariate", verbose=0,
                random_state=seed,
                structural_params=covariate_params,
                max_iter=100, n_init = 20),

            '2-STEP': StepMix(n_components=n_components,
                n_steps=2, measurement="categorical",
                structural="covariate", verbose=0,
                random_state=seed,
                structural_params=covariate_params,
                max_iter=100, n_init = 20),

            '3-STEP': StepMix(n_components=n_components,
                n_steps=3, measurement="categorical",
                structural="covariate", verbose=0,
                random_state=seed,
```

```
                structural_params=covariate_params,
                max_iter=100, n_init = 20)
        }


    iterations_results = []

    for seed in range(seed_start, seed_end):
        for n_components in range(min_comp, max_comp):
            print(f"Start of analysis of Number of Components:
            {n_components}")
            for num_knots in knots_list:
                data_with_bspline = add_bspline_columns(data,
                    x_colname, num_knots, 3)
                models_to_evaluate =
                    create_models(n_components, seed)
                for method_name, model in
                models_to_evaluate.items():
                    try:
                        fit = model.fit(
                        X=data_with_bspline.iloc[:,
                        3:19],
                        Y=data_with_bspline.iloc[:,
                        19:])

                        aic =
                        fit.aic(X=data_with_bspline.iloc[:,
                        3:19],Y=data_with_bspline.iloc[:,
                        19:])

                        min_weight = np.min(fit.weights_)

                        iterations_results.append({
                            'Method': method_name,
                            'AIC': aic,
                            'Knot Number': num_knots,
                            'Seed': seed,
                            'Number of Components':
```

```
                        n_components,
                        'Min Weights': min_weight
                    })
                except np.linalg.LinAlgError as e:
                    if "SVD did not converge" in str(e):
                        print(f"Iteration {seed} skipped
                        for {method_name} at knot
                        {num_knots} due to SVD
                        convergence issue: {e}")
                        continue

            temp_df = pd.DataFrame(iterations_results)
            temp_df.to_csv(f'evaluation_results_up_to_comp_
            {n_components}_seed_{seed}.csv', index=False)
            print(f"End of analysis of Number of Components:
            {n_components}")

    results_df = pd.DataFrame(iterations_results, columns=
    ['Method', 'AIC', 'Knot Number', 'Seed', 'Number of
    Components', 'Min Weights'])

    return results_df
```

B.2 B-spline Generation in Brexit Analysis

```
def add_bspline_columns(data, x_colname, num_knots, degree):
    quantile_positions = np.linspace(1/(num_knots+1),
    num_knots/(num_knots+1), num_knots)
    quantiles = data[x_colname].quantile(quantile_positions)

    min_value = data[x_colname].min()
    max_value = data[x_colname].max()

    t = quantiles.values
    t = np.pad(t, pad_width=degree+1, mode='constant',
    constant_values=(min_value, max_value))
```

```
    bases_values = [bspline_bases(x, t, degree) for x in
        data[x_colname]]

    bases_df = pd.DataFrame(bases_values, columns=[f"B{i+1}"
    for i in range(len(t) - degree - 1)])

    data.reset_index(drop=True, inplace=True)
    bases_df.reset_index(drop=True, inplace=True)
```

B.3 Creation of the Model for a Variable Number of Latent Classes

```
def evaluate_methods_groups(covariate_params,
    sample_sizes, teta_values, knots_list, n_iterations=100,
    num_manifest_variables=10):

    def create_models(n_components, seed):
        return {
            '1-STEP': StepMix(n_components=n_components,
            n_steps=1,
                                  measurement="binary",
                                structural="covariate",
                                verbose=0,
                                random_state=seed,
                                structural_params=
                                covariate_params,
                                max_iter=100),

            '2-STEP': StepMix(n_components=n_components,
            n_steps=2,
                                  measurement="binary",
                                structural="covariate",
                                verbose=0,
                                random_state=seed,
                                structural_params=
                                covariate_params,
                                max_iter=100),
```

```
        '3-STEP': StepMix(n_components=n_components,
        n_steps=3,
                                  measurement="binary",
                                structural="covariate",
                                verbose=0,
                                random_state=seed,
                                structural_params=
                                covariate_params,
                                max_iter=100)
    }

def create_models2(seed):
    return {
        '1-STEP': StepMix(n_components=1,
        n_steps=1,
        measurement="binary", verbose=0,
        random_state=seed, max_iter=100),

        '2-STEP': StepMix(n_components=1,
        n_steps=2,
        measurement="binary", verbose=0,
        random_state=seed, max_iter=100),

        '3-STEP': StepMix(n_components=1,
        n_steps=3,
        measurement="binary", verbose=0,
        random_state=seed, max_iter=100)
    }


results_counts = {
    (teta, sample_size): {method: {i: 0 for
    i in range(1, 5)}
                                for method in
                                ['1-STEP',
                                '2-STEP', '3-STEP']}
    for teta in teta_values for
```

```
    sample_size in sample_sizes
}

min_weights = {
    (teta, sample_size): {method: {i:
    [] for i in range(1, 5)}
                                for method in
                                ['1-STEP',
                                '2-STEP', '3-STEP']}
    for teta in teta_values
    for sample_size in sample_sizes
}

for teta in teta_values:
    for sample_size in sample_sizes:
        temp_min_weights = {}
        for method in ['1-STEP', '2-STEP',
        '3-STEP']:
            min_weights[(teta, sample_size)]
            [method][1] = []
        for iteration in range(n_iterations):
            best_aic_per_method =
            {'1-STEP': float('inf'),
            '2-STEP': float('inf'),
            '3-STEP': float('inf')}
            best_group_per_method =
            {'1-STEP': None,
            '2-STEP': None,
            '3-STEP': None}
            best_knot_per_method =
            {'1-STEP': None,
            '2-STEP': None,
            '3-STEP': None}


            for n_components in range(1, 5):
                dati = gendati(n=2000,
                seed=iteration, teta=teta,
```

```
num_manifest_variables=
num_manifest_variables)

if n_components == 1:
    models_to_evaluate =
    create_models2(iteration)
    for method_name,
        model in
        models_to_evaluate.items():
        fit =
        model.fit(X=dati.iloc[:sample_size,
        0:num_manifest_variables])
        aic =
        fit.aic(X=dati.iloc[:
        sample_size,
       0:num_manifest_variables])

        if aic <
        best_aic_per_method[method_name]:
            best_aic_per_method[
            method_name] =
            aic
            best_group_per_method[
            method_name]
            =
            n_components
            best_knot_per_method[
            method_name] = None
            temp_min_weights[method_name] =
                        np.min(
                        fit.weights_)

else:
    for knot in knots_list:
        t = np.linspace(0, 1, knot - 2)
        t = np.pad(t, pad_width=3,
        mode='constant',
            constant_values=(0, 1))
```

```
dati_with_bspline =
    add_bspline_columns(dati,
    'x', t, 3)

models_to_evaluate = create_models(
        n_components, iteration)
for method_name, model in
    models_to_evaluate.items():
    fit = model.fit(X=
        dati_with_bspline.iloc[
        :sample_size,
        0:num_manifest_variables],
        Y=dati_with_bspline.iloc[:
        sample_size,
        num_manifest_variables+2:])
    aic = fit.aic(X=
        dati_with_bspline.iloc[:
        sample_size,
        0:num_manifest_variables],
        Y=dati_with_bspline.iloc[:
        sample_size,
        num_manifest_variables+2:])

    # Update best AIC, groups,
    #and knots for each method
    if aic < best_aic_per_method[
            method_name]:
        best_aic_per_method[
            method_name] = aic
        best_group_per_method[
            method_name] =
            n_components
        best_knot_per_method[
            method_name] = knot
        temp_min_weights[
            method_name] =
            np.min(fit.weights_)
```

```
            # Update the count for the chosen number of
            #components for each method
            for method_name in ['1-STEP',
            '2-STEP', '3-STEP']:
                chosen_components = best_group_per_method[
                    method_name]
                if chosen_components is not None:
                    results_counts[(teta, sample_size)][
                    method_name][chosen_components] += 1
                    if method_name in temp_min_weights:
                        min_weights[(teta, sample_size)]
                        [method_name][
                            chosen_components].append(
                            temp_min_weights[method_name])

df_rows = [(teta, sample_size) for teta in teta_values
    for sample_size in sample_sizes]
df_index = pd.MultiIndex.from_tuples(df_rows,
    names=["Theta", "Sample Size"])
df_columns = [f"{method} {comp}-Comp" for
    method in ['1-STEP', '2-STEP', '3-STEP']
    for comp in range(1, 5)]
results_df = pd.DataFrame(index=df_index,
    columns=df_columns)

for (teta, sample_size), method_counts in
    results_counts.items():
    for method, comp_counts in method_counts.items():
        for comp, count in comp_counts.items():
            min_weight_values = min_weights[(teta,
                sample_size)][method][comp]
            min_weight_str = ', '.join([f"{mw:.3f}"
                for mw in min_weight_values])
            results_df.loc[(teta, sample_size),
                f"{method} {comp}-Comp"] =
                    f"{count} ({min_weight_str})"
return results_df
```

Appendix C

Simulation Study Figures

C.1 Study with a Fixed Number of Latent Classes

C.1.1 Comparing f_2 and $\hat{f}_2$ with Sample Size = 500 and $\theta = 0.6$

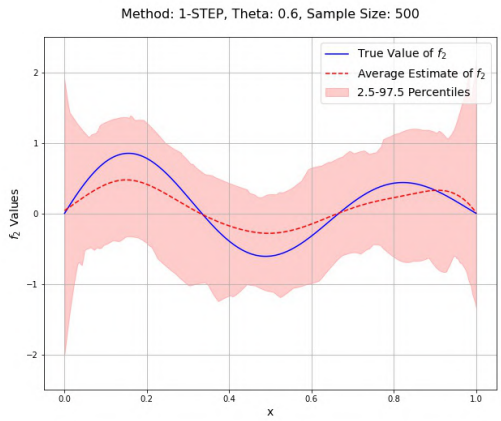


Figure C.1: True vs. Estimated f_2 using 1-Step with $\theta = 0.6$, Sample Size = 500

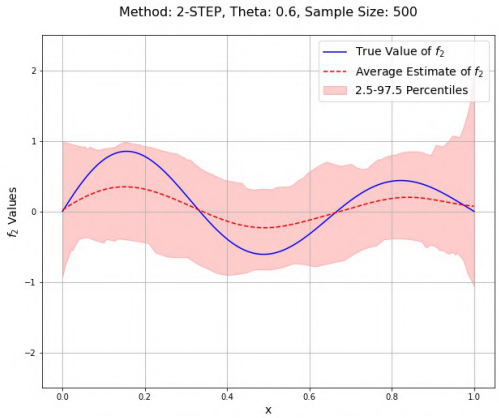


Figure C.2: True vs. Estimated f_2 using 2-Step with $\theta = 0.6$, Sample Size = 500

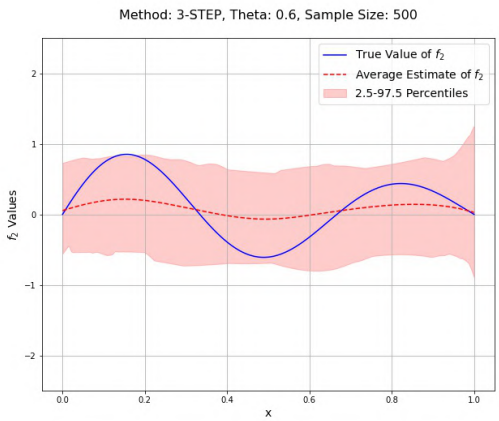


Figure C.3: True vs. Estimated f_2 using 3-Step with $\theta = 0.6$, Sample Size = 500

C.1.2 Comparing f_2 and $\hat{f}_2$ with Sample Size = 1000 and $\theta = 0.6$

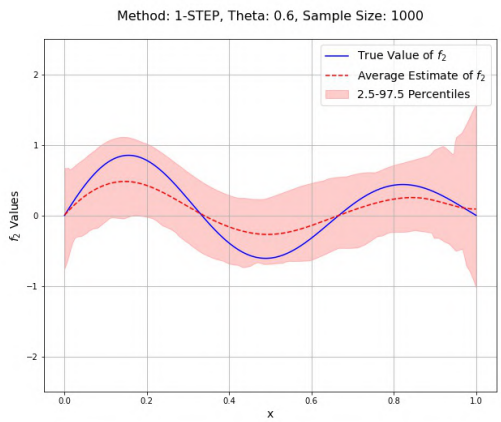


Figure C.4: True vs. Estimated f_2 using 1-Step with $\theta = 0.6$, Sample Size = 1000

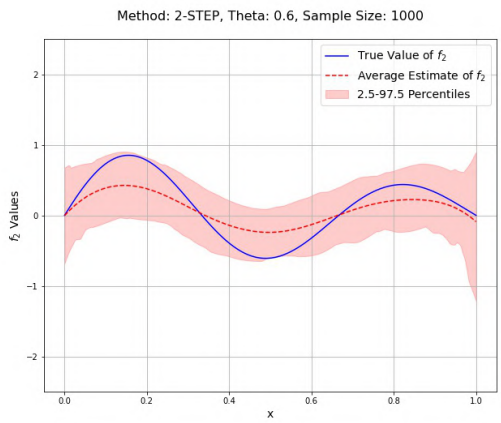


Figure C.5: True vs. Estimated f_2 using 2-Step with $\theta = 0.6$, Sample Size = 1000

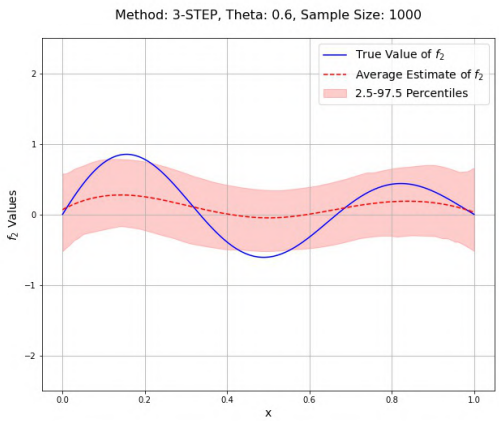


Figure C.6: True vs. Estimated f_2 using 3-Step with $\theta = 0.6$, Sample Size = 1000

C.1.3 Comparing f_2 and $\hat{f}_2$ with Sample Size = 2000 and $\theta = 0.6$

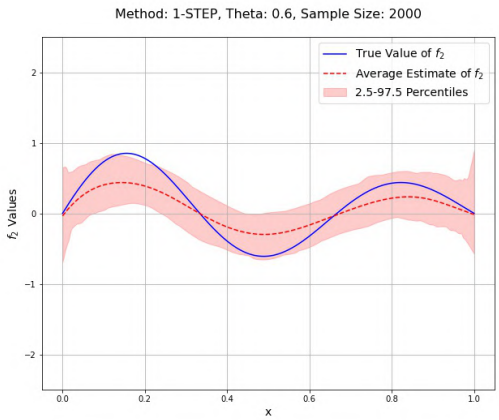


Figure C.7: True vs. Estimated f_2 using 1-Step with $\theta = 0.6$, Sample Size = 2000

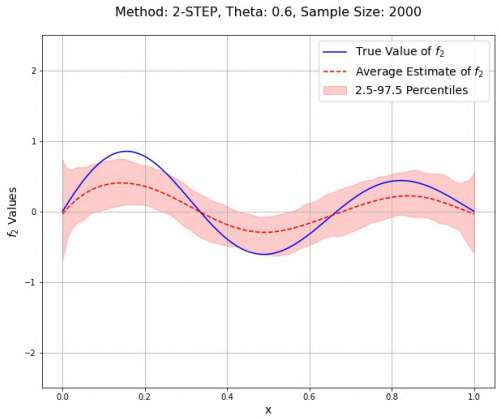


Figure C.8: True vs. Estimated f_2 using 2-Step with $\theta = 0.6$, Sample Size = 1000

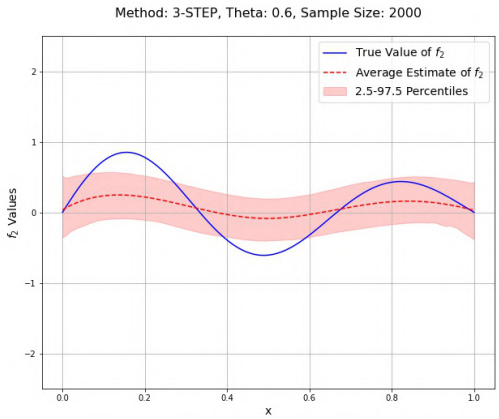


Figure C.9: True vs. Estimated f_2 using 3-Step with $\theta = 0.6$, Sample Size = 2000

C.1.4 Comparing f_2 and $\hat{f}_2$ with Sample Size = 500 and $\theta = 0.7$

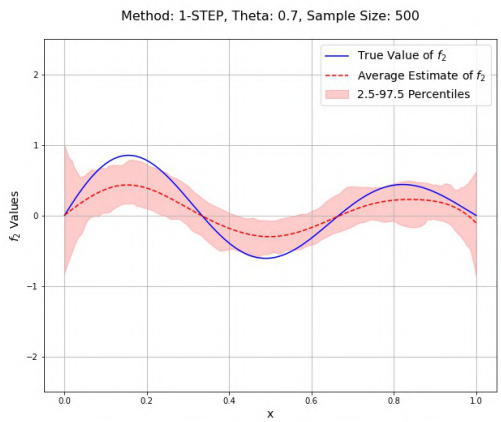


Figure C.10: True vs. Estimated f_2 using 1-Step with $\theta = 0.7$, Sample Size = 500

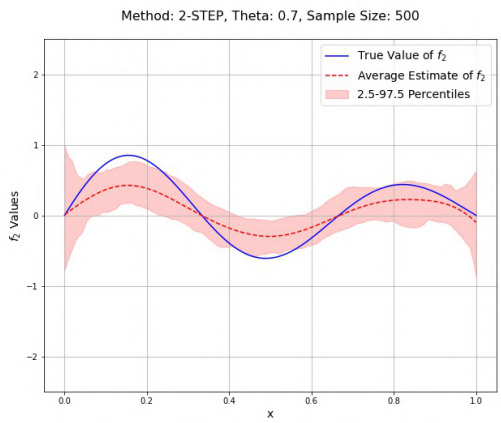


Figure C.11: True vs. Estimated f_2 using 2-Step with $\theta = 0.7$, Sample Size = 500

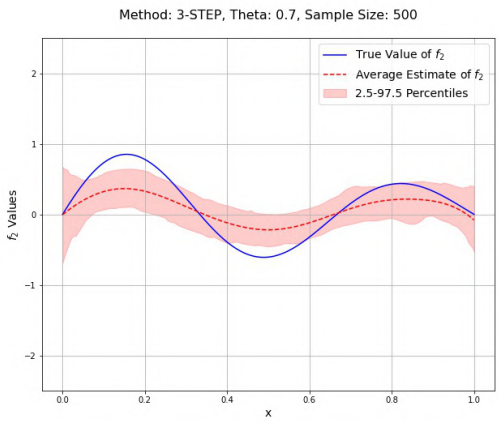


Figure C.12: True vs. Estimated f_2 using 3-Step with $\theta = 0.7$, Sample Size = 500

C.1.5 Comparing f_2 and $\hat{f}_2$ with Sample Size = 1000 and $\theta = 0.7$

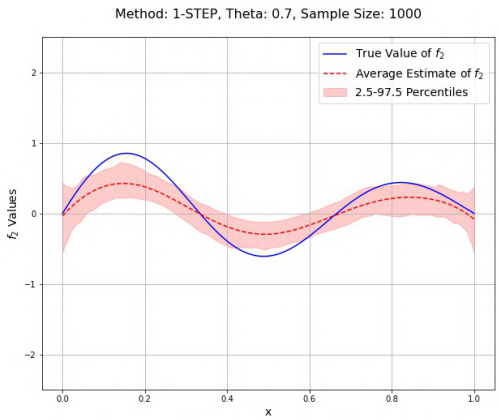


Figure C.13: True vs. Estimated f_2 using 1-Step with $\theta = 0.7$, Sample Size = 1000

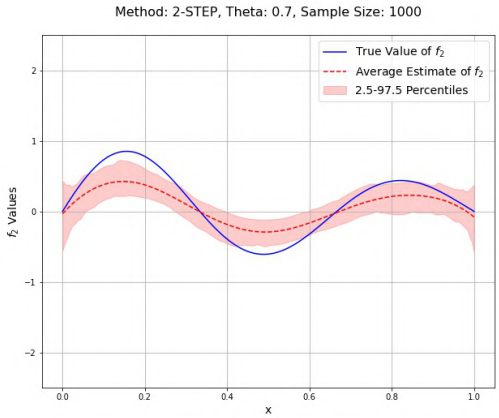


Figure C.14: True vs. Estimated f_2 using 2-Step with $\theta = 0.7$, Sample Size = 1000

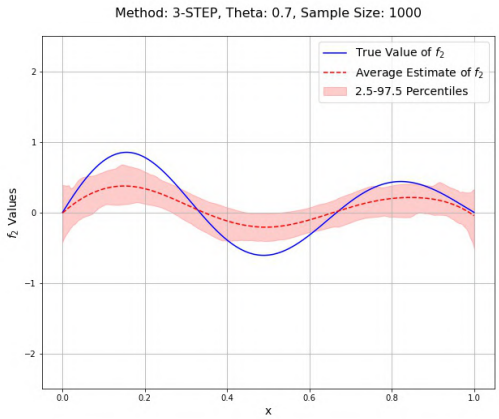


Figure C.15: True vs. Estimated f_2 using 3-Step with $\theta = 0.7$, Sample Size = 1000

C.1.6 Comparing f_2 and $\hat{f}_2$ with Sample Size = 2000 and $\theta = 0.7$

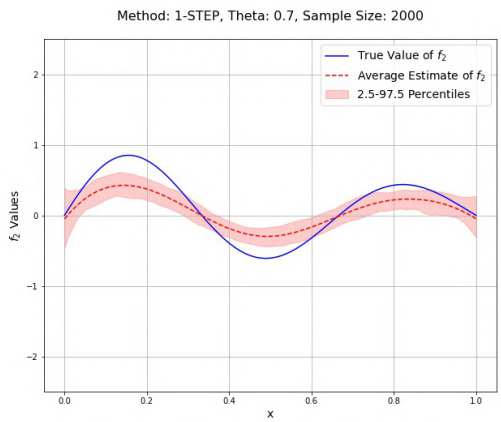


Figure C.16: True vs. Estimated f_2 using 1-Step with $\theta = 0.7$, Sample Size = 2000

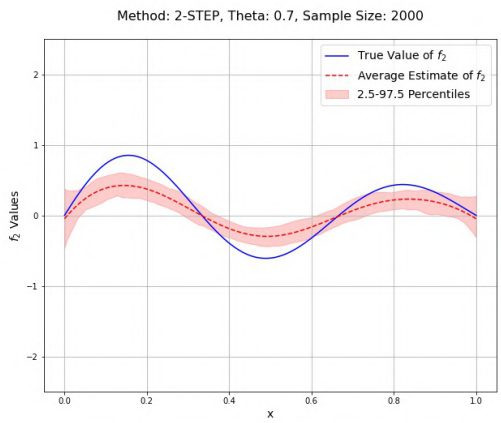


Figure C.17: True vs. Estimated f_2 using 2-Step with $\theta = 0.7$, Sample Size = 1000

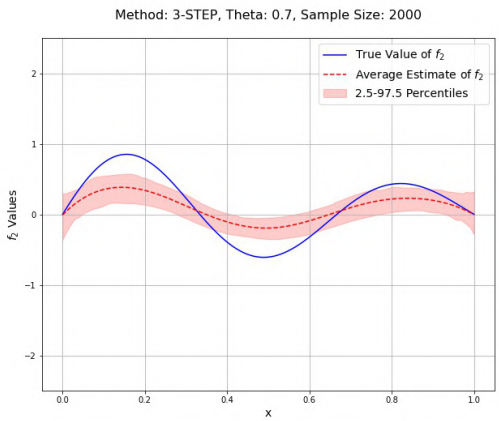


Figure C.18: True vs. Estimated f_2 using 3-Step with $\theta = 0.7$, Sample Size = 2000

C.1.7 Comparing f_2 and $\hat{f}_2$ with Sample Size = 500 and $\theta = 0.8$

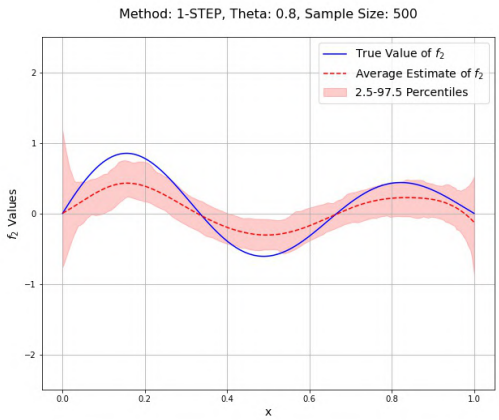


Figure C.19: True vs. Estimated f_2 using 1-Step with $\theta = 0.8$, Sample Size = 500

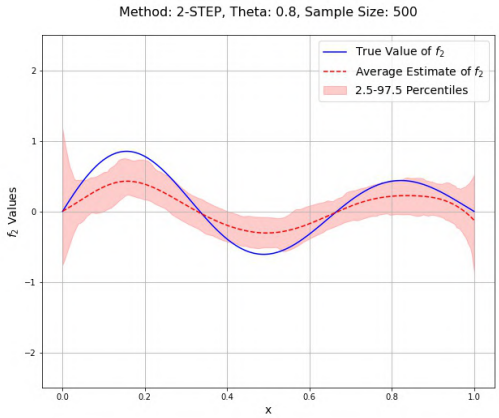


Figure C.20: True vs. Estimated f_2 using 2-Step with $\theta = 0.8$, Sample Size = 500

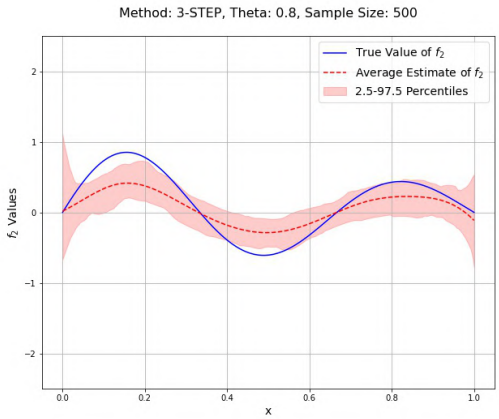


Figure C.21: True vs. Estimated f_2 using 3-Step with $\theta = 0.8$, Sample Size = 500

C.1.8 Comparing f_2 and $\hat{f}_2$ with Sample Size = 1000 and $\theta = 0.8$

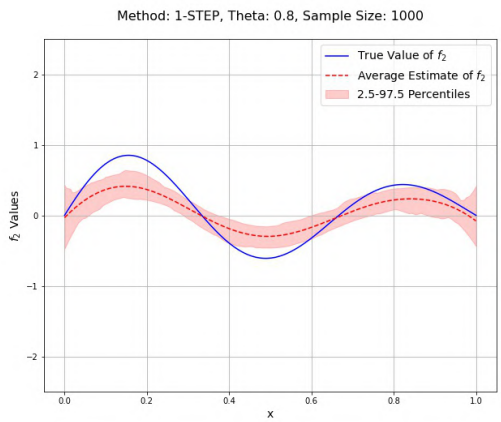


Figure C.22: True vs. Estimated f_2 using 1-Step with $\theta = 0.8$, Sample Size = 1000

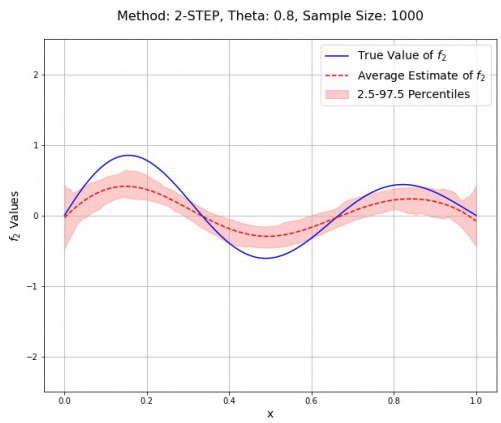


Figure C.23: True vs. Estimated f_2 using 2-Step with $\theta = 0.8$, Sample Size = 1000

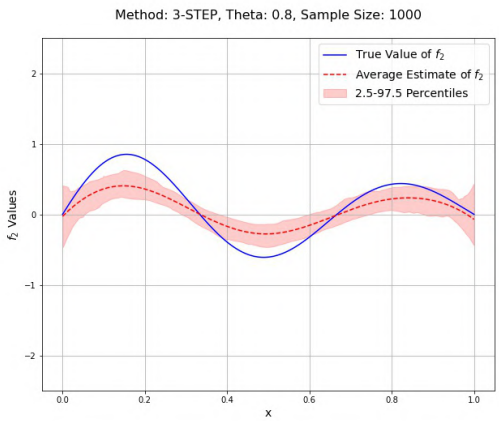


Figure C.24: True vs. Estimated f_2 using 3-Step with $\theta = 0.8$, Sample Size = 1000

C.1.9 Comparing f_2 and $\hat{f}_2$ with Sample Size = 2000 and $\theta = 0.8$

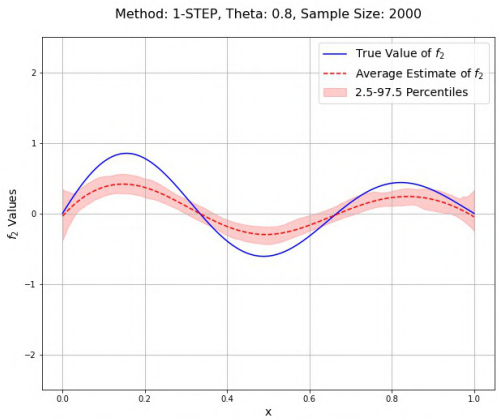


Figure C.25: True vs. Estimated f_2 using 1-Step with $\theta = 0.8$, Sample Size = 2000

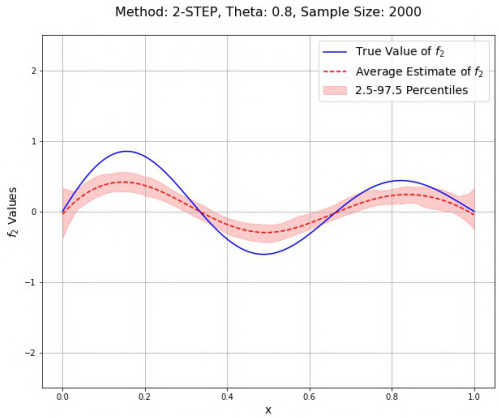


Figure C.26: True vs. Estimated f_2 using 2-Step with $\theta = 0.8$, Sample Size = 1000

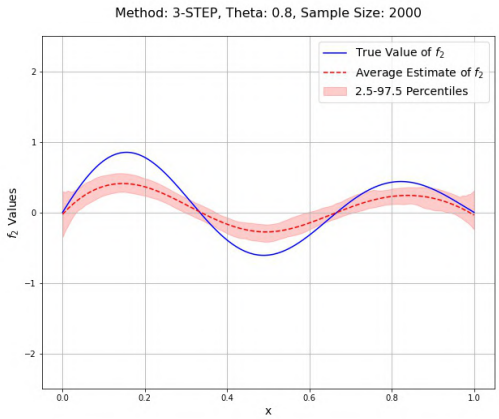


Figure C.27: True vs. Estimated f_2 using 3-Step with $\theta = 0.8$, Sample Size = 2000

C.1.10 Comparing f_2 and $\hat{f}_2$ with Sample Size = 500 and $\theta = 0.9$

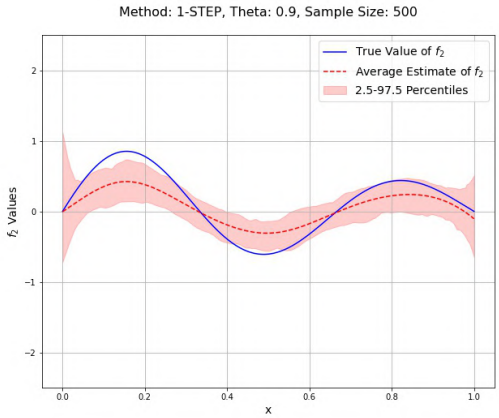


Figure C.28: True vs. Estimated f_2 using 1-Step with $\theta = 0.9$, Sample Size = 500

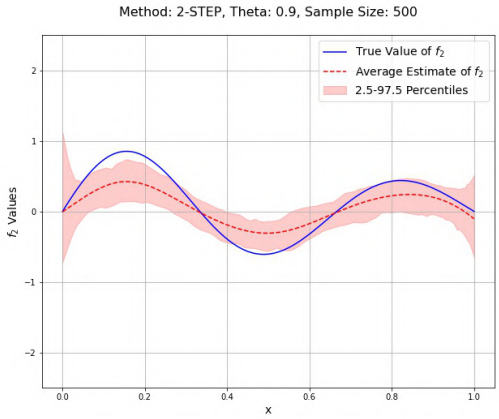


Figure C.29: True vs. Estimated f_2 using 2-Step with $\theta = 0.9$, Sample Size = 500

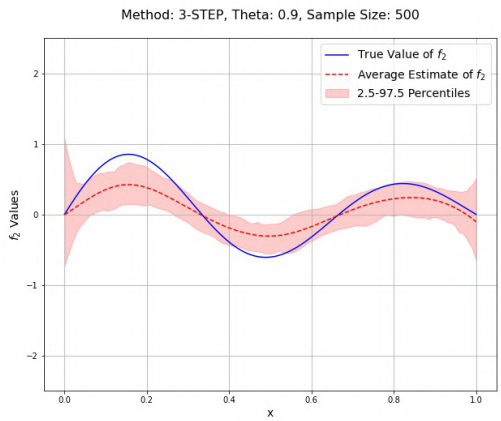


Figure C.30: True vs. Estimated f_2 using 3-Step with $\theta = 0.9$, Sample Size = 500

C.1.11 Comparing f_2 and $\hat{f}_2$ with Sample Size = 1000 and $\theta = 0.9$

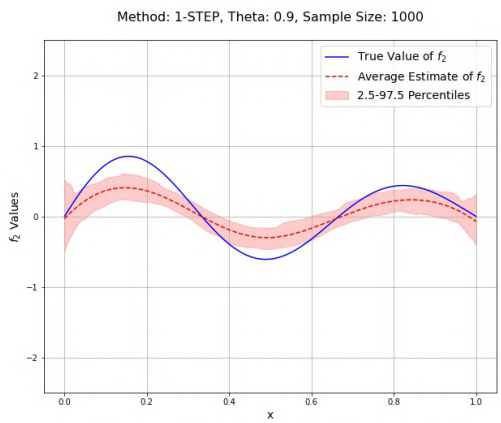


Figure C.31: True vs. Estimated f_2 using 1-Step with $\theta = 0.9$, Sample Size = 1000

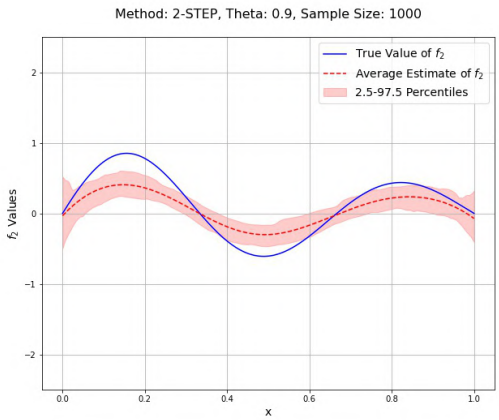


Figure C.32: True vs. Estimated f_2 using 2-Step with $\theta = 0.9$, Sample Size = 1000

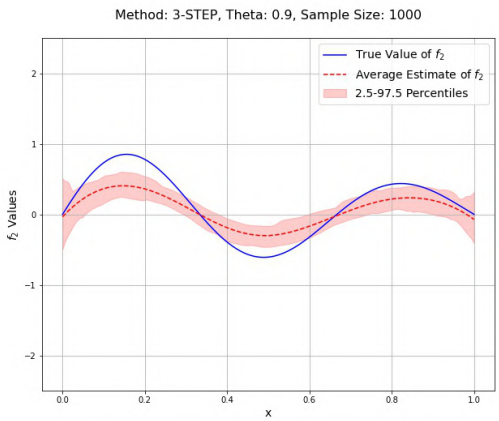


Figure C.33: True vs. Estimated f_2 using 3-Step with $\theta = 0.9$, Sample Size = 1000

C.1.12 Comparing f_2 and $\hat{f}_2$ with Sample Size = 2000 and $\theta = 0.9$

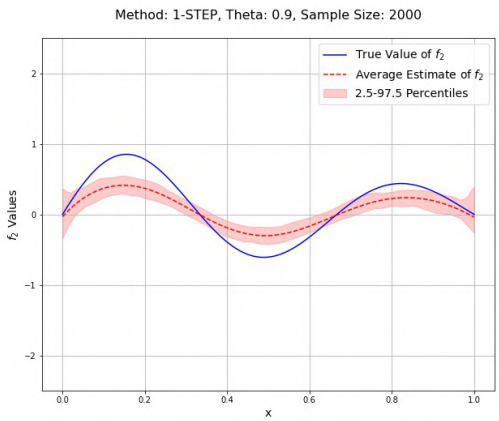


Figure C.34: True vs. Estimated f_2 using 1-Step with $\theta = 0.9$, Sample Size = 2000

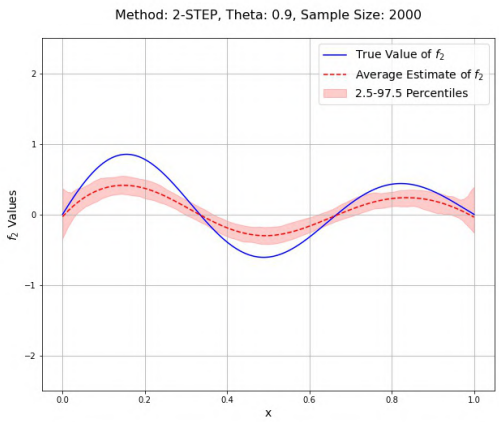


Figure C.35: True vs. Estimated f_2 using 2-Step with $\theta = 0.9$, Sample Size = 1000

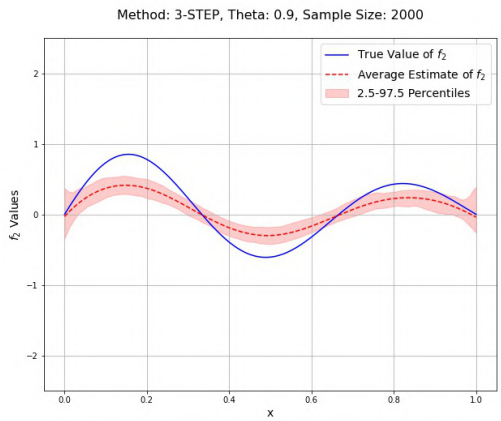


Figure C.36: True vs. Estimated f_2 using 3-Step with $\theta = 0.9$, Sample Size = 2000

C.2 Roughness measure

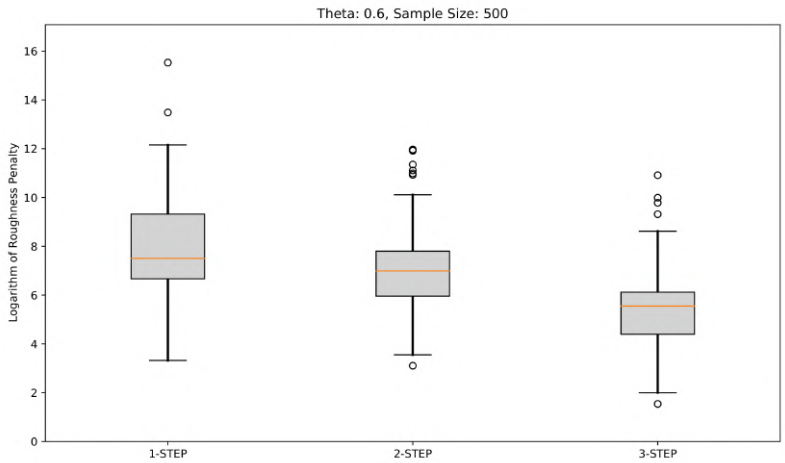


Figure C.37: Boxplot comparison of roughness penalty logarithms for estimation methods, with $\theta = 0.6$, Sample Size = 500

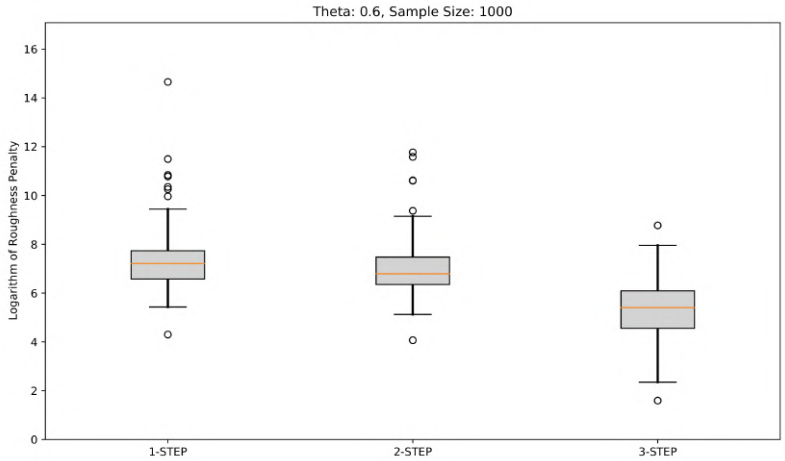


Figure C.38: Boxplot comparison of roughness penalty logarithms for estimation methods, with $\theta = 0.6$, Sample Size = 1000

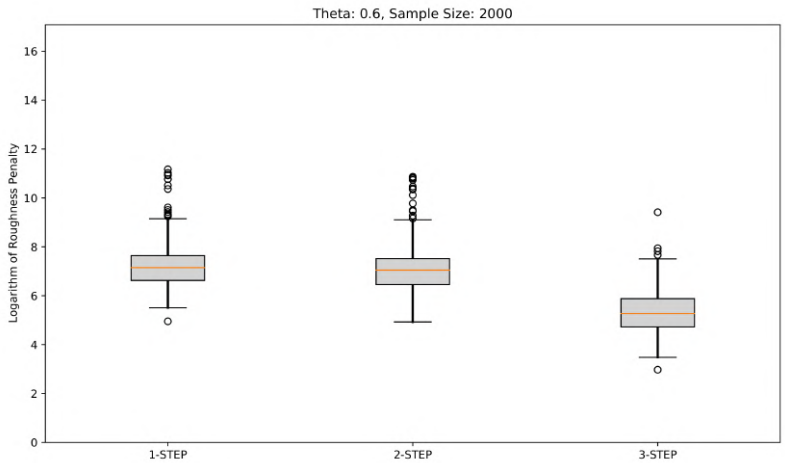


Figure C.39: Boxplot comparison of roughness penalty logarithms for estimation methods, with $\theta = 0.6$, Sample Size = 2000

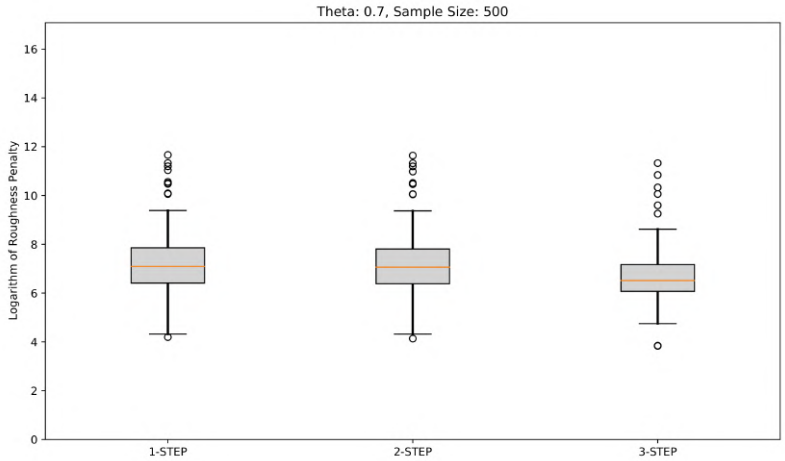


Figure C.40: Boxplot comparison of roughness penalty logarithms for estimation methods, with $\theta = 0.7$, Sample Size = 500

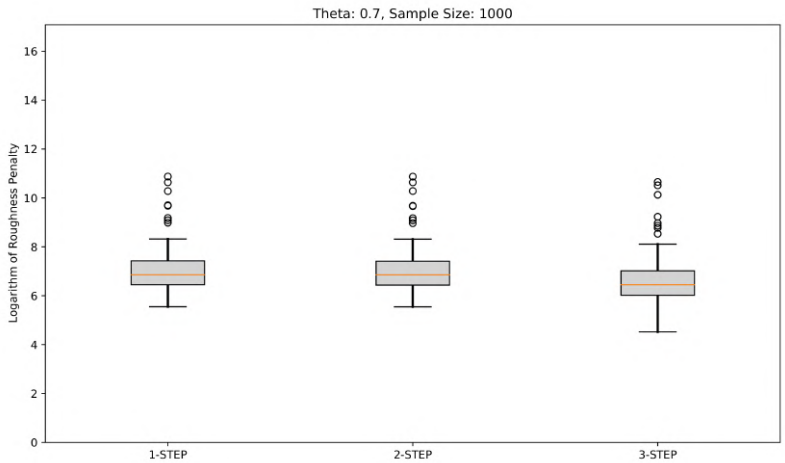


Figure C.41: Boxplot comparison of roughness penalty logarithms for estimation methods, with $\theta = 0.7$, Sample Size = 1000

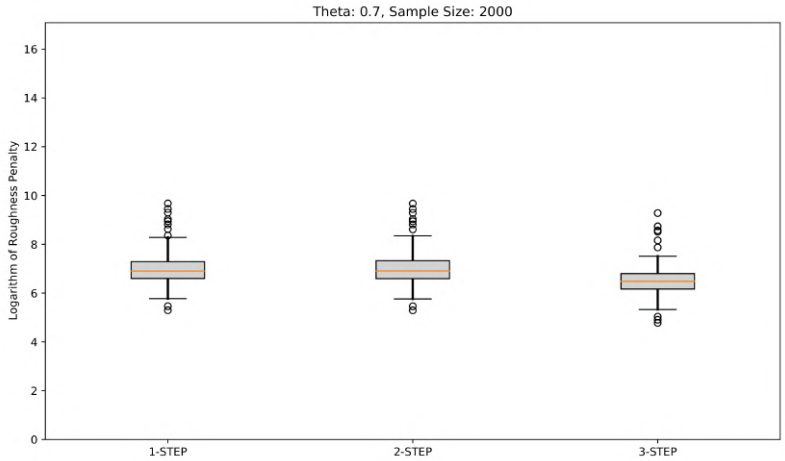


Figure C.42: Boxplot comparison of roughness penalty logarithms for estimation methods, with $\theta = 0.7$, Sample Size = 2000

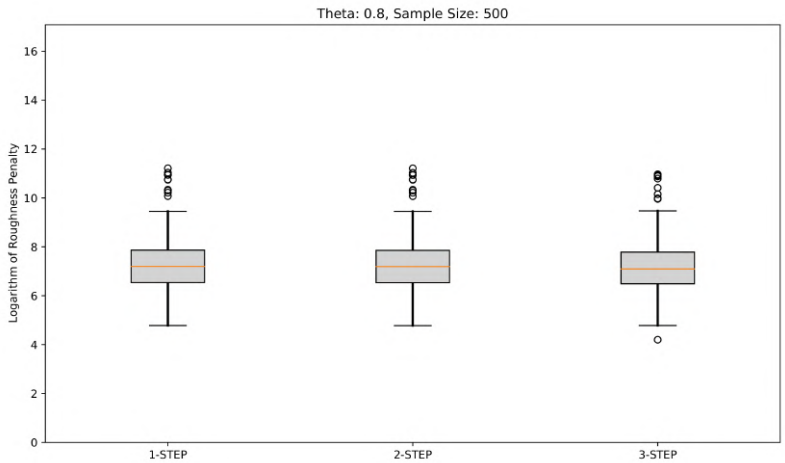


Figure C.43: Boxplot comparison of roughness penalty logarithms for estimation methods, with $\theta = 0.8$, Sample Size = 500

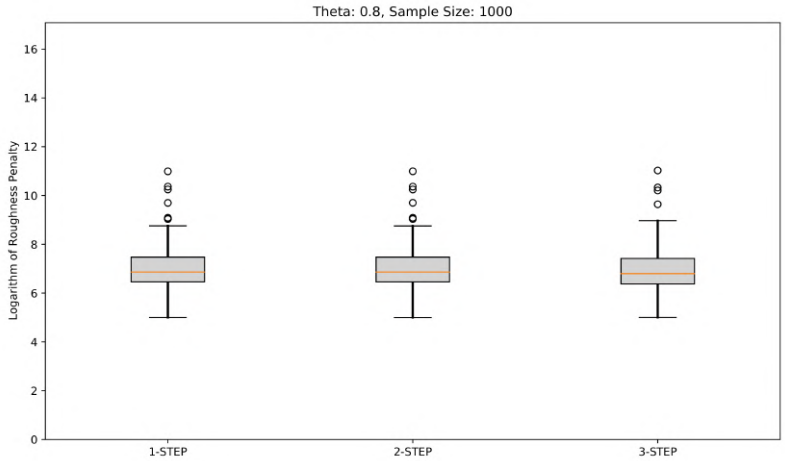


Figure C.44: Boxplot comparison of roughness penalty logarithms for estimation methods, with $\theta = 0.8$, Sample Size = 1000

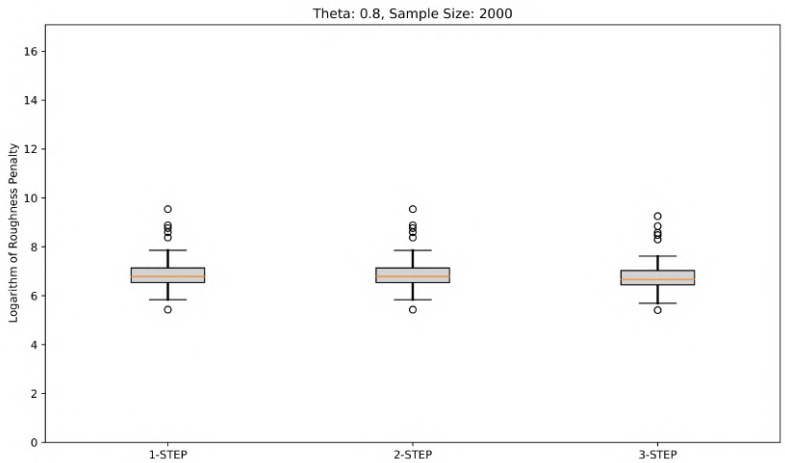


Figure C.45: Boxplot comparison of roughness penalty logarithms for estimation methods, with $\theta = 0.8$, Sample Size = 2000

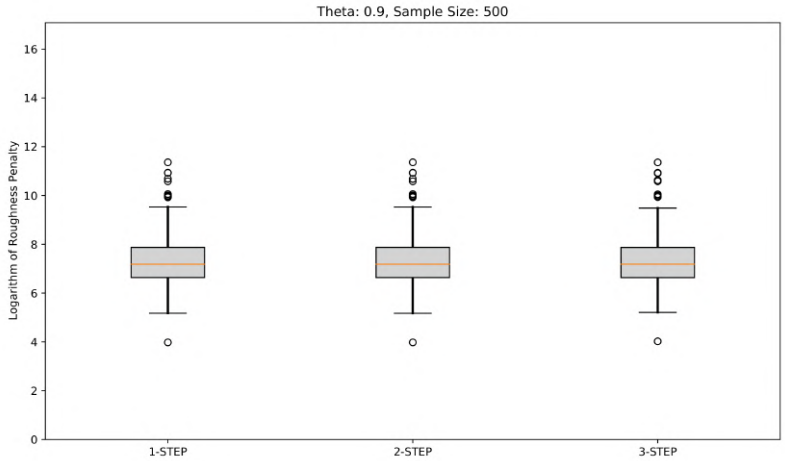


Figure C.46: Boxplot comparison of roughness penalty logarithms for estimation methods, with $\theta = 0.9$, Sample Size = 500

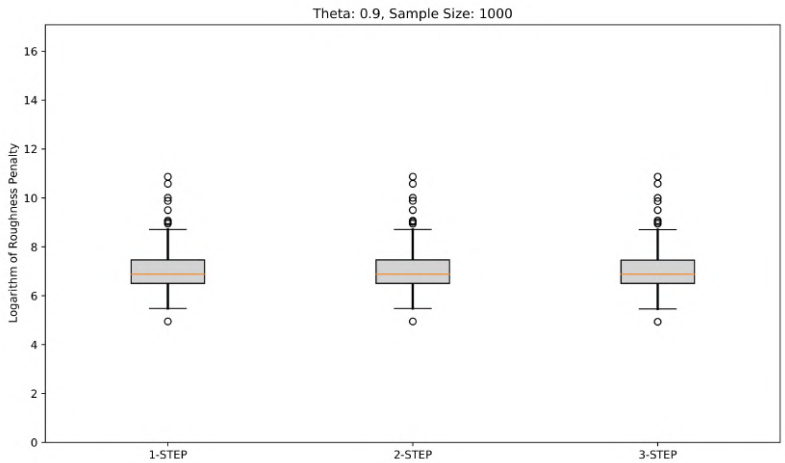


Figure C.47: Boxplot comparison of roughness penalty logarithms for estimation methods, with $\theta = 0.9$, Sample Size = 1000

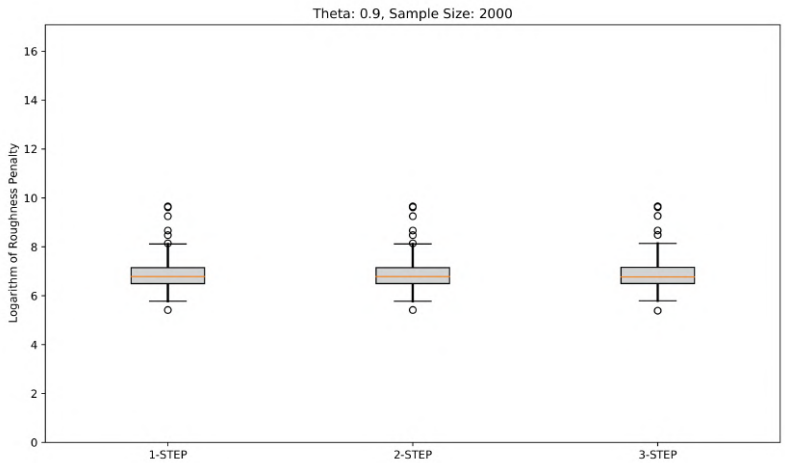


Figure C.48: Boxplot comparison of roughness penalty logarithms for estimation methods, with $\theta = 0.9$, Sample Size = 2000

C.3 Root Mean Square Error

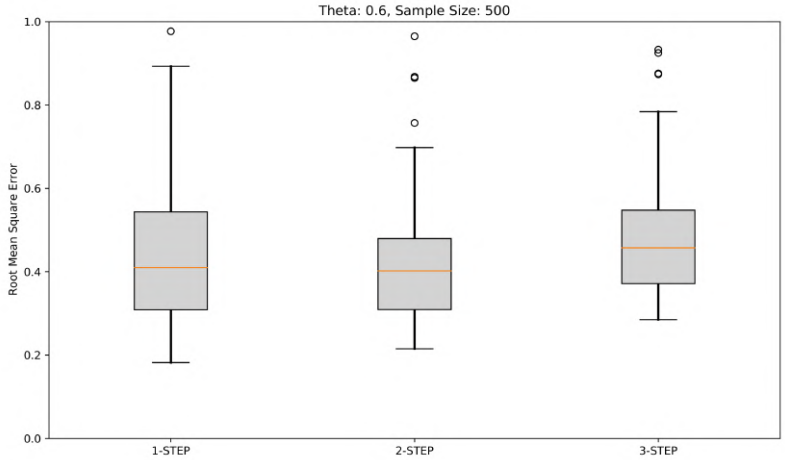


Figure C.49: Boxplot comparison of root mean square errors for estimation methods, with $\theta = 0.6$, Sample Size = 500

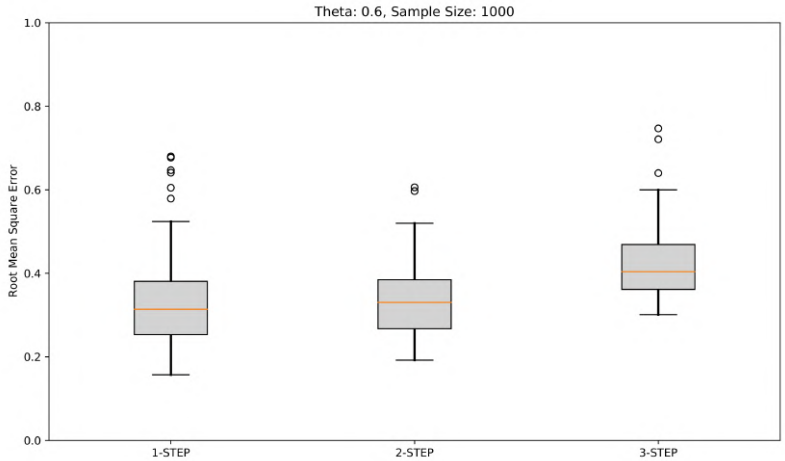


Figure C.50: Boxplot comparison of root mean square errors for estimation methods, with $\theta = 0.6$, Sample Size = 1000

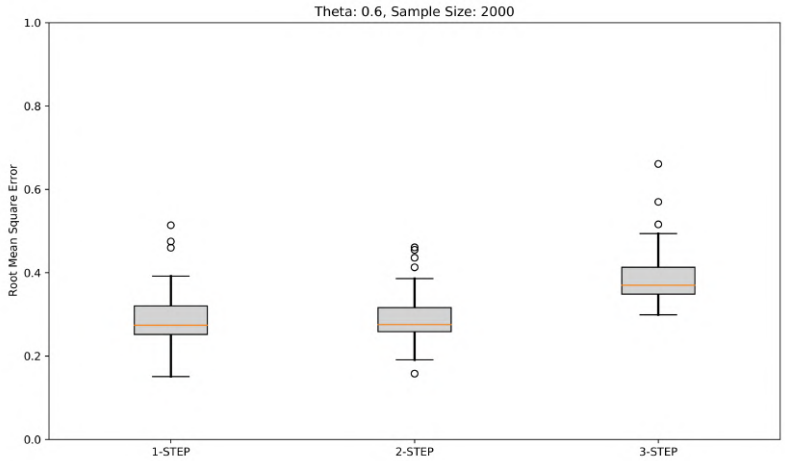


Figure C.51: Boxplot comparison of root mean square errors for estimation methods, with $\theta = 0.6$, Sample Size = 2000

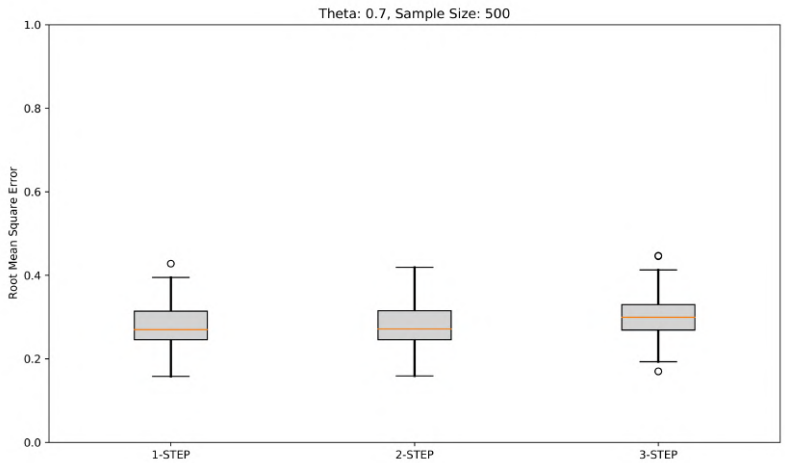


Figure C.52: Boxplot comparison of root mean square errors for estimation methods, with $\theta = 0.7$, Sample Size = 500

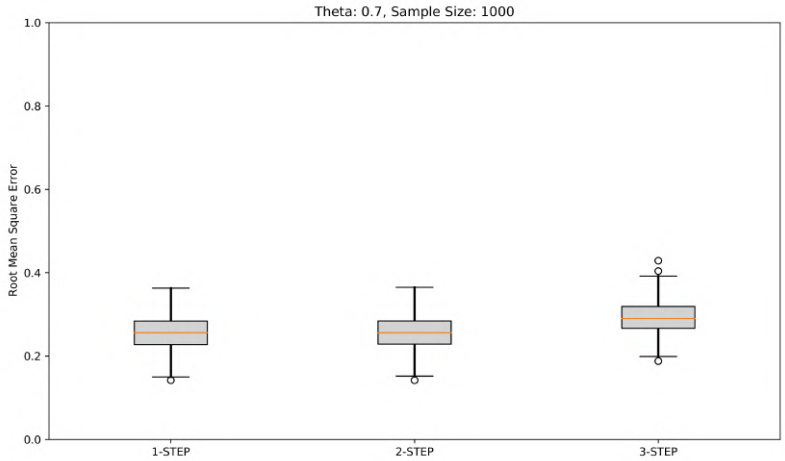


Figure C.53: Boxplot comparison of root mean square errors for estimation methods, with $\theta = 0.7$, Sample Size = 1000

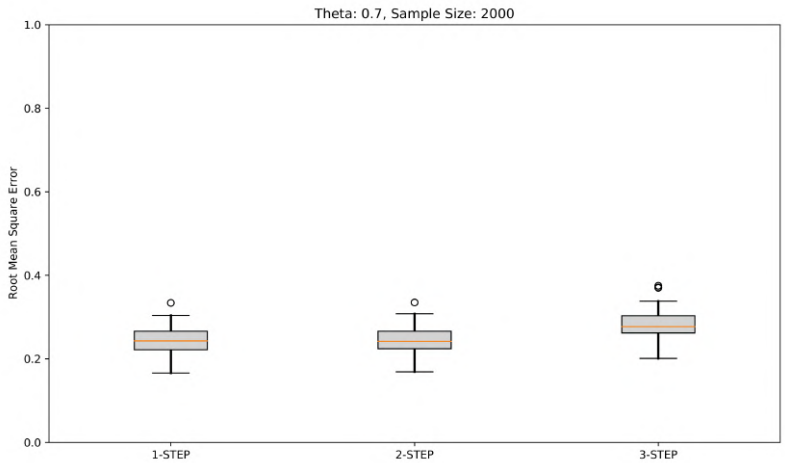


Figure C.54: Boxplot comparison of root mean square errors for estimation methods, with $\theta = 0.7$, Sample Size = 2000

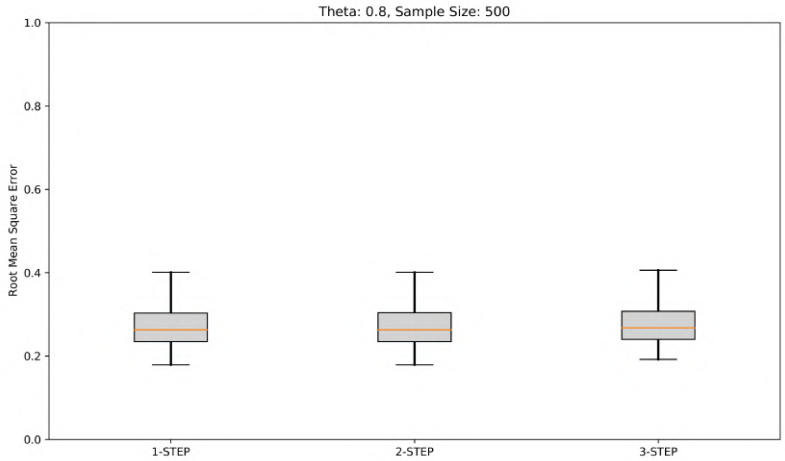


Figure C.55: Boxplot comparison of root mean square errors for estimation methods, with $\theta = 0.8$, Sample Size = 500

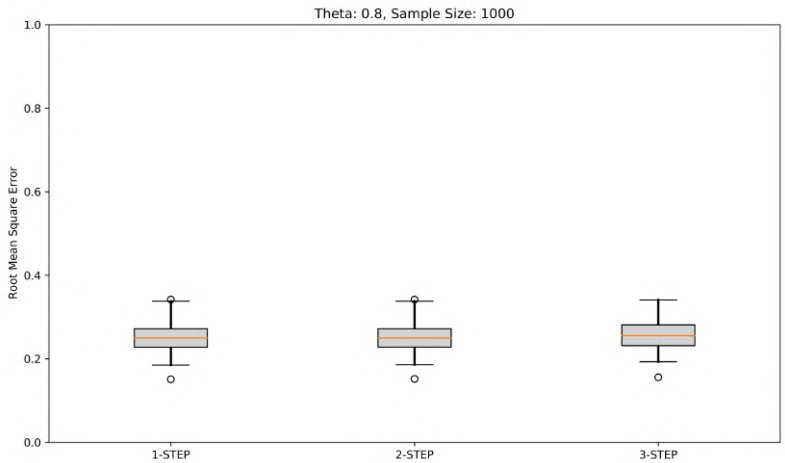


Figure C.56: Boxplot comparison of root mean square errors for estimation methods, with $\theta = 0.8$, Sample Size = 1000

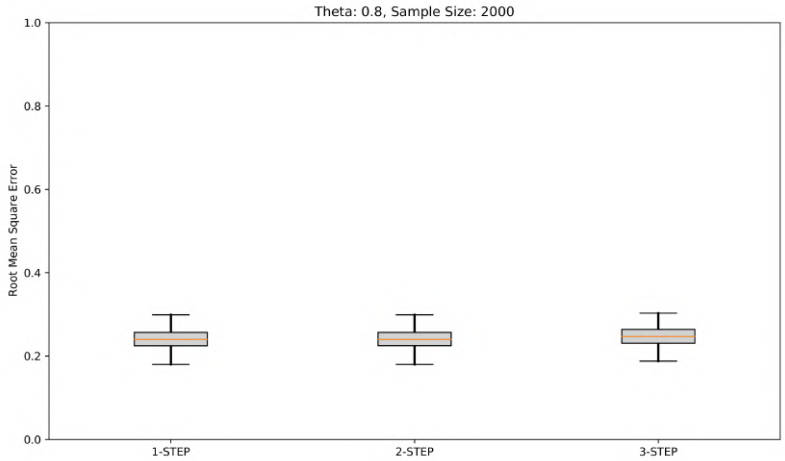


Figure C.57: Boxplot comparison of root mean square errors for estimation methods, with $\theta = 0.8$, Sample Size = 2000

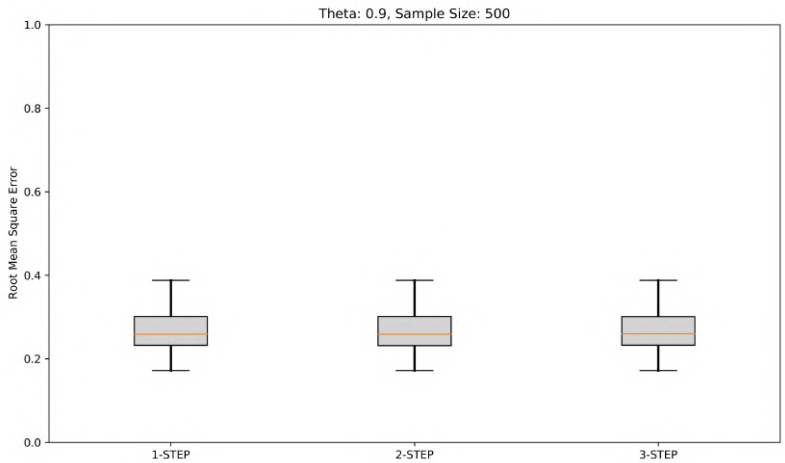


Figure C.58: Boxplot comparison of root mean square errors for estimation methods, with $\theta = 0.9$, Sample Size = 500

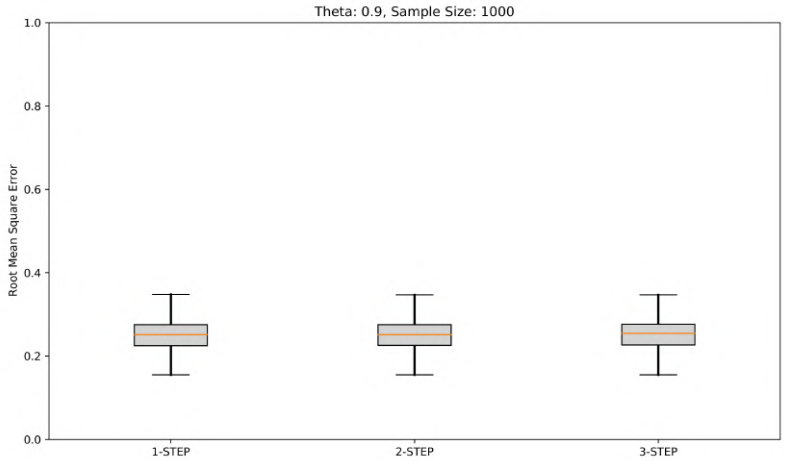


Figure C.59: Boxplot comparison of root mean square errors for estimation methods, with $\theta = 0.9$, Sample Size = 1000

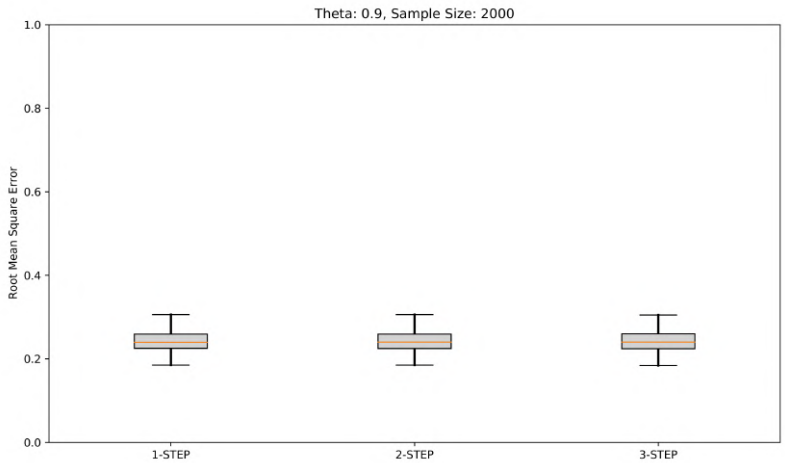


Figure C.60: Boxplot comparison of root mean square errors for estimation methods, with $\theta = 0.9$, Sample Size = 2000

C.4 Internal Knots selected by Estimation Method

Class Separation	Sample Size	AIC's Internal Knots Selected							BIC's Internal Knots Selected						
		1	2	3	4	5	6	7	1	2	3	4	5	6	7
VERY LOW (θ = 0.6)	500	58	16	7	5	4	6	4	94	5	1	0	0	0	0
	1000	62	22	6	2	2	5	1	98	2	0	0	0	0	0
	2000	48	23	8	5	5	6	5	99	1	0	0	0	0	0
LOW (θ = 0.7)	500	62	16	8	5	2	5	2	98	2	0	0	0	0	0
	1000	64	16	7	5	2	5	1	98	1	1	0	0	0	0
	2000	47	22	14	9	3	4	1	94	4	2	0	0	0	0
MEDIUM (θ = 0.8)	500	55	15	10	7	2	7	4	97	3	0	0	0	0	0
	1000	61	16	7	6	5	4	1	98	1	1	0	0	0	0
	2000	48	20	13	12	4	2	1	96	2	2	0	0	0	0
HIGH (θ = 0.9)	500	57	14	10	7	2	7	3	99	1	0	0	0	0	0
	1000	59	13	11	6	4	6	1	98	1	1	0	0	0	0
	2000	40	20	17	16	3	2	2	97	2	1	0	0	0	0

Figure C.61: Internal knots selected by AIC and BIC with 1-Step Estimation

Class Separation	Sample Size	AIC's Internal Knots Selected							BIC's Internal Knots Selected						
		1	2	3	4	5	6	7	1	2	3	4	5	6	7
VERY LOW (θ = 0.6)	500	71	15	6	2	1	3	2	96	3	1	0	0	0	0
	1000	67	20	7	1	1	3	1	99	1	0	0	0	0	0
	2000	53	19	8	4	6	5	5	99	1	0	0	0	0	0
LOW (θ = 0.7)	500	62	16	8	5	2	5	2	98	2	0	0	0	0	0
	1000	64	16	7	5	2	5	1	98	1	1	0	0	0	0
	2000	46	22	14	10	3	4	1	94	4	2	0	0	0	0
MEDIUM (θ = 0.8)	500	55	15	10	7	2	7	4	97	3	0	0	0	0	0
	1000	61	16	7	6	5	4	1	98	1	1	0	0	0	0
	2000	48	20	13	12	4	2	1	96	2	2	0	0	0	0
HIGH (θ = 0.9)	500	57	14	10	7	2	7	3	99	1	0	0	0	0	0
	1000	59	13	11	6	4	6	1	98	1	1	0	0	0	0
	2000	40	20	17	16	3	2	2	97	2	1	0	0	0	0

Figure C.62: Internal knots selected by AIC and BIC with 2-Step Estimation

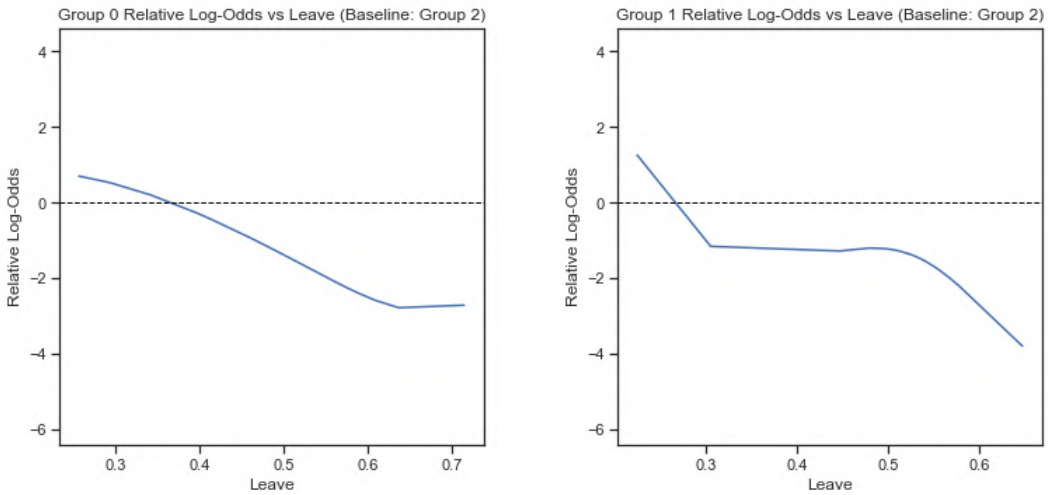
Class Separation	Sample Size	AIC's Internal Knots Selected							BIC's Internal Knots Selected						
		1	2	3	4	5	6	7	1	2	3	4	5	6	7
VERY LOW (θ = 0.6)	500	81	12	2	2	0	2	1	98	2	0	0	0	0	0
	1000	78	16	5	0	0	1	0	100	0	0	0	0	0	0
	2000	73	15	7	3	1	1	0	100	0	0	0	0	0	0
LOW (θ = 0.7)	500	72	12	5	5	2	2	2	98	2	0	0	0	0	0
	1000	66	14	7	6	2	4	1	98	1	1	0	0	0	0
	2000	50	26	12	7	3	2	0	94	4	2	0	0	0	0
MEDIUM (θ = 0.8)	500	57	13	10	7	2	7	4	97	3	0	0	0	0	0
	1000	61	16	8	5	5	4	1	97	2	1	0	0	0	0
	2000	48	20	14	11	4	2	1	96	2	2	0	0	0	0
HIGH (θ = 0.9)	500	57	14	10	7	2	7	3	99	1	0	0	0	0	0
	1000	59	13	11	6	4	6	1	98	1	1	0	0	0	0
	2000	40	20	17	17	3	1	2	97	2	1	0	0	0	0

Figure C.63: Internal knots selected by AIC and BIC with 3-Step Estimation

Appendix D

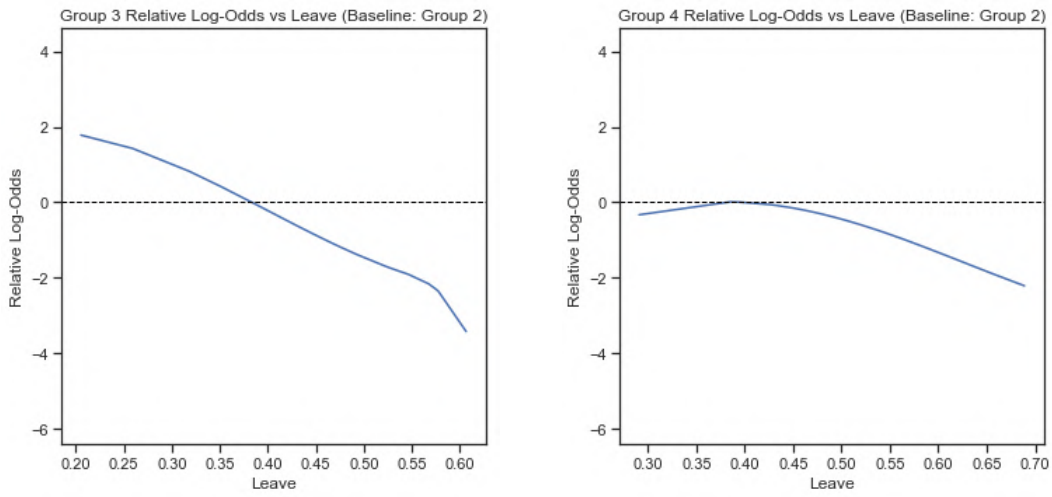
Brexit Analysis Figures

D.1 Relative Log-Odds with 1-Step and "Leave" Variable



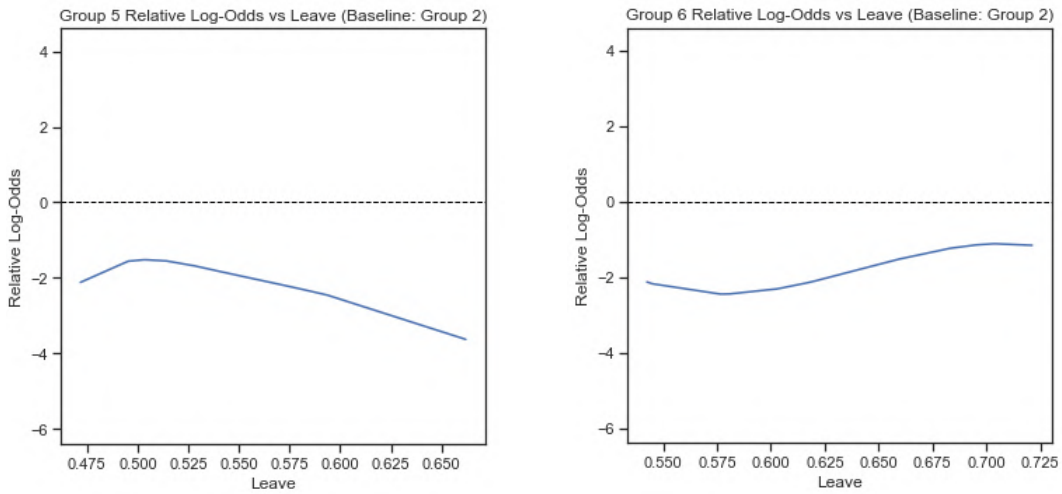
(a) (b)

Figure D.1: Relative Log-Odds in Cluster 0 and 1 with 1-Step estimation and "Leave" as concomitant variable



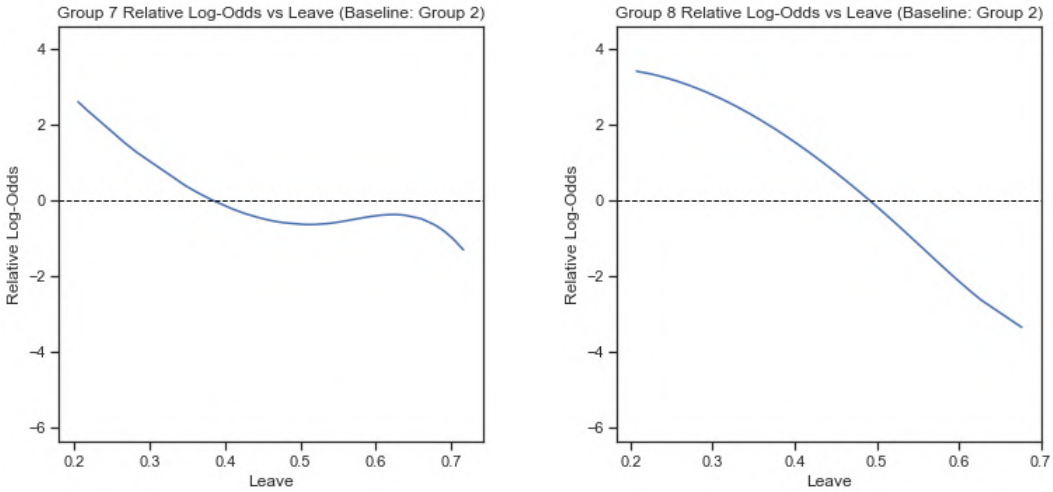
(a) (b)

Figure D.2: Relative Log-Odds in Cluster 3 and 4 with 1-Step estimation and “Leave” as concomitant variable



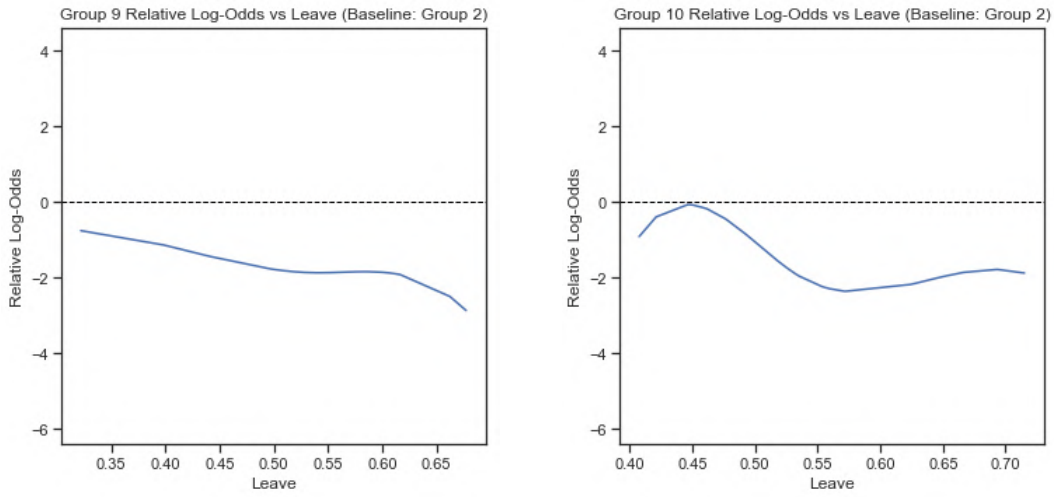
(a) (b)

Figure D.3: Relative Log-Odds in Cluster 5 and 6 with 1-Step estimation and “Leave” as concomitant variable



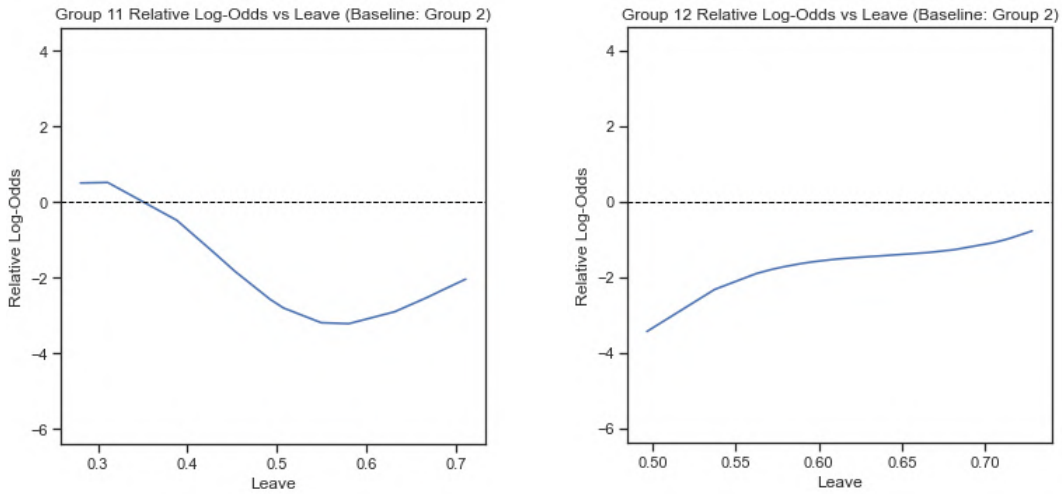
(a) (b)

Figure D.4: Relative Log-Odds in Cluster 7 and 8 with 1-Step estimation and "Leave" as concomitant variable



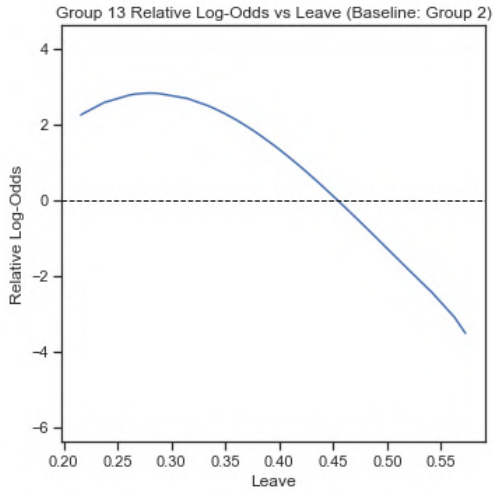
(a) (b)

Figure D.5: Relative Log-Odds in Cluster 9 and 10 with 1-Step estimation and “Leave” as concomitant variable



(a) (b)

Figure D.6: Relative Log-Odds in Cluster 11 and 12 with 1-Step estimation and “Leave” as concomitant variable



(a)

Figure D.7: Relative Log-Odds in Cluster 13 with 1-Step estimation and “Leave” as concomitant variable

D.2 1-Step Clusters with “Leave” as Concomitant Variable

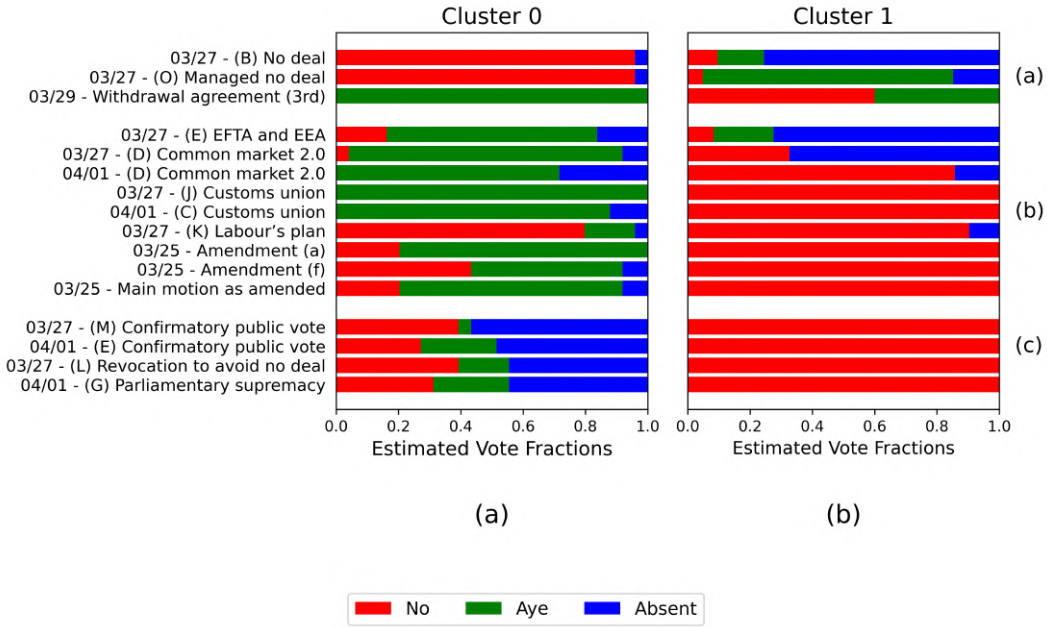


Figure D.8: Clusters 0 and 1 with 1-Step estimation and “Leave” as concomitant variable

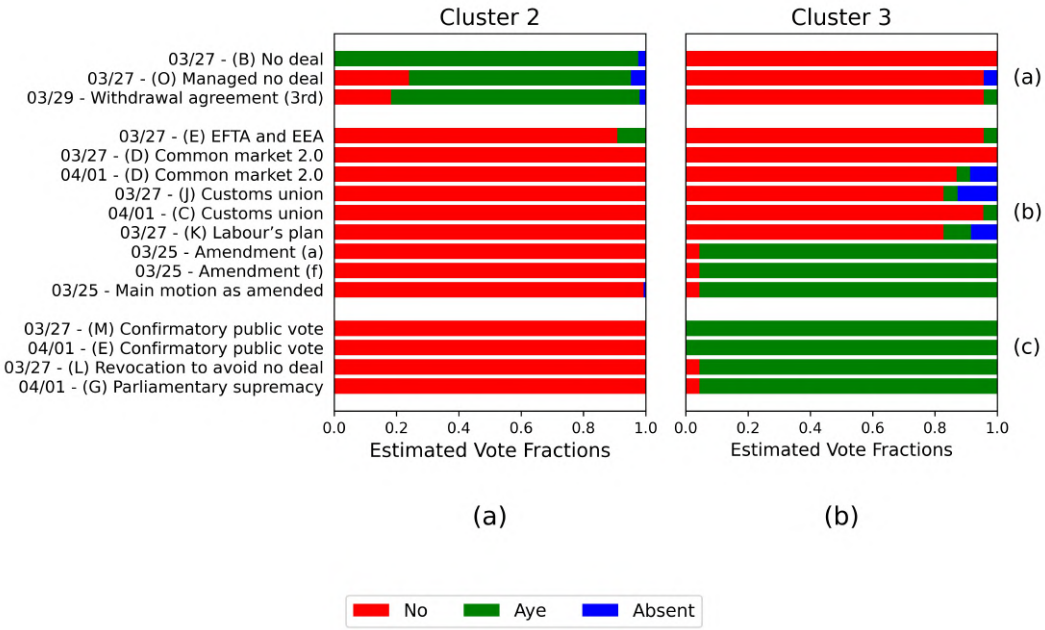


Figure D.9: Clusters 2 and 3 with 1-Step estimation and "Leave" as concomitant variable

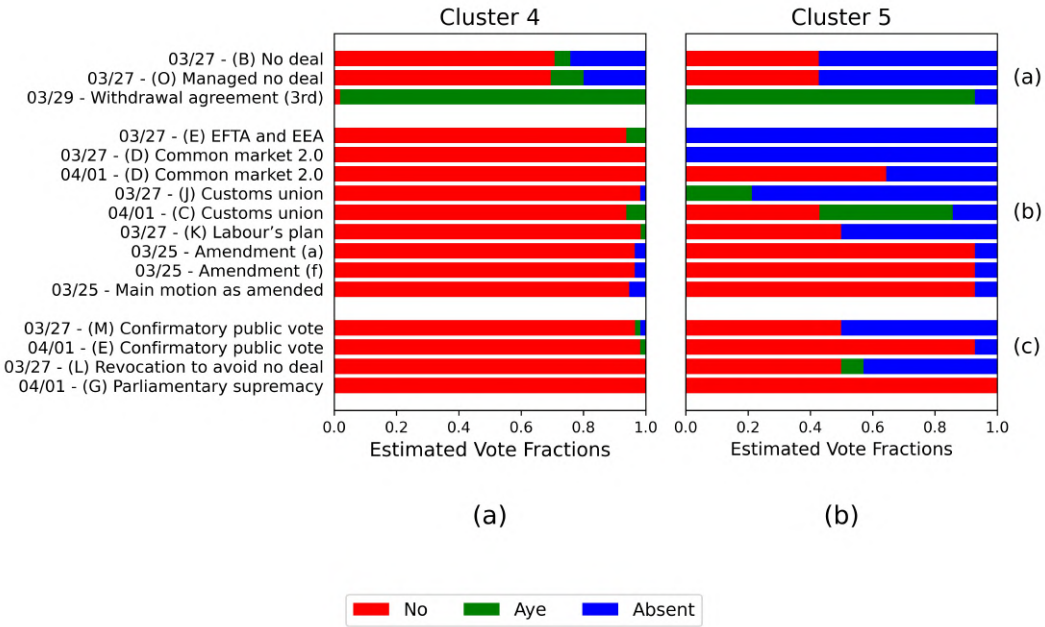


Figure D.10: Clusters 4 and 5 with 1-Step estimation and "Leave" as concomitant variable

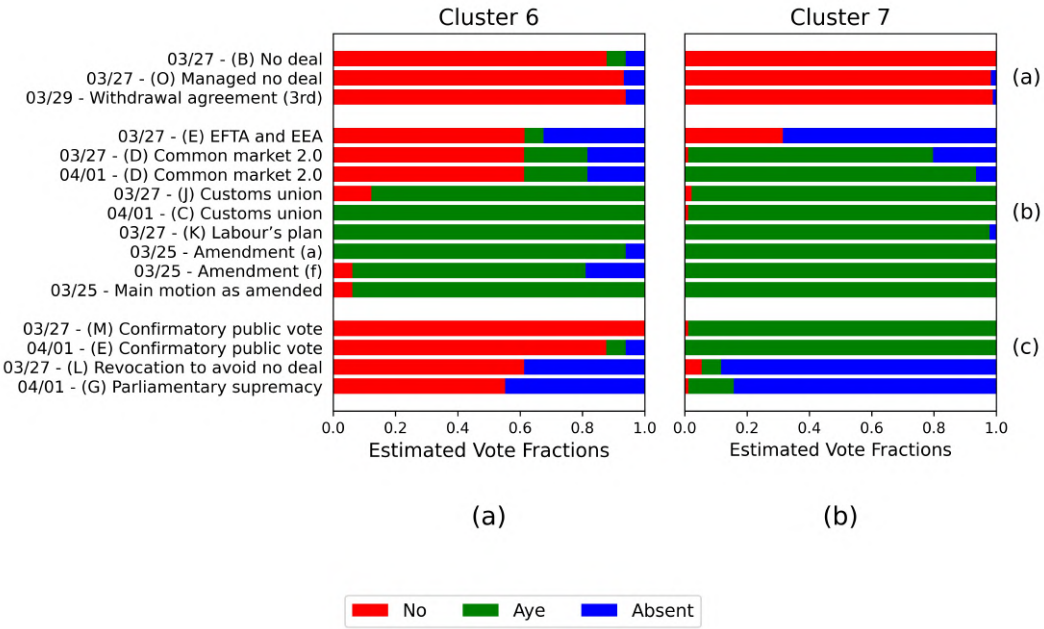


Figure D.11: Clusters 6 and 7 with 1-Step estimation and “Leave” as concomitant variable

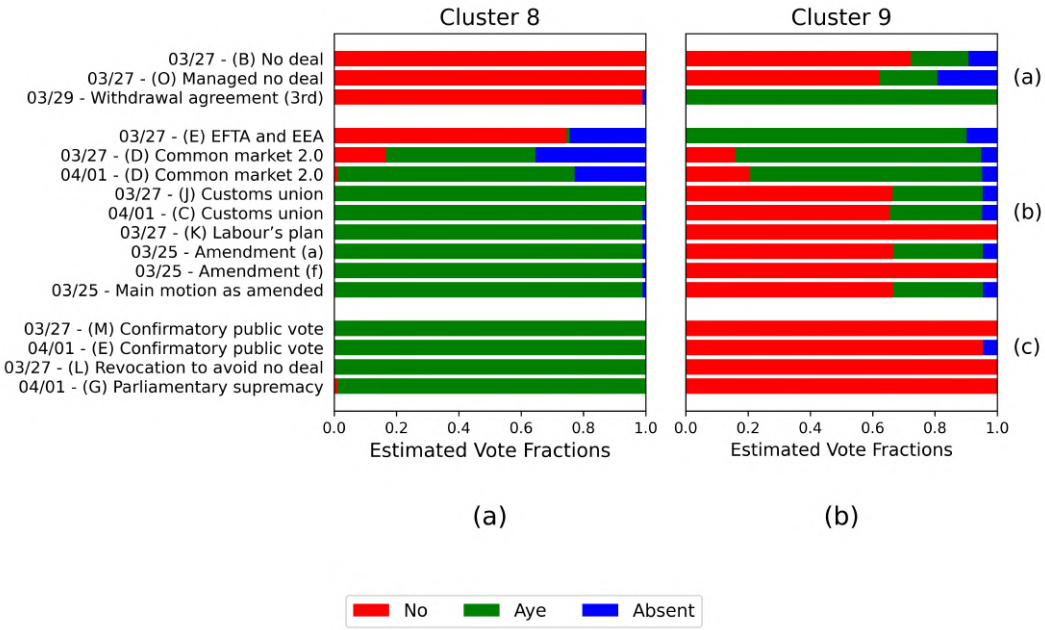


Figure D.12: Clusters 8 and 9 with 1-Step estimation and "Leave" as concomitant variable

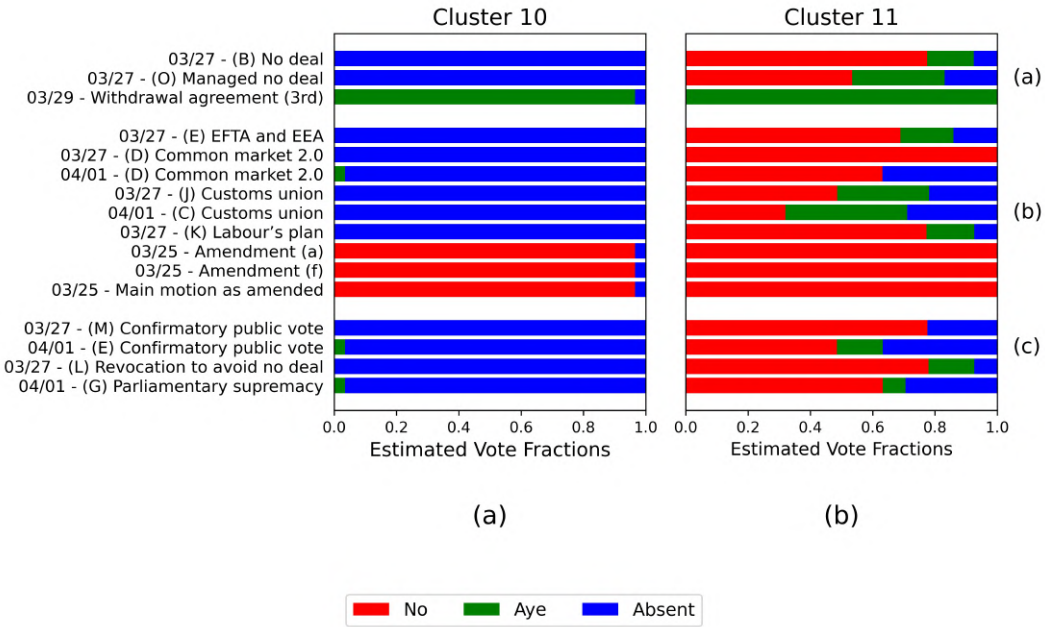


Figure D.13: Clusters 10 and 11 with 1-Step estimation and "Leave" as concomitant variable

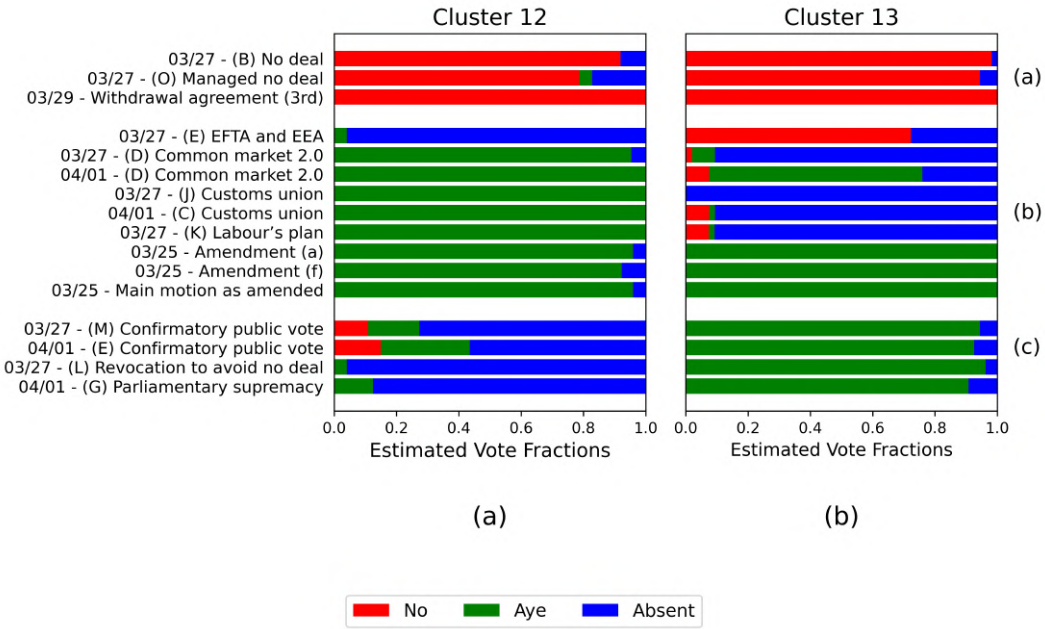
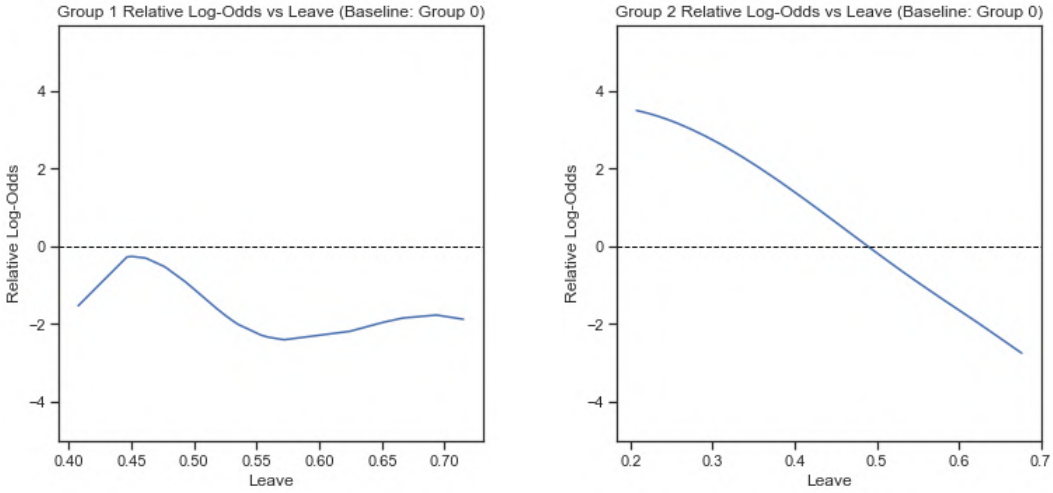


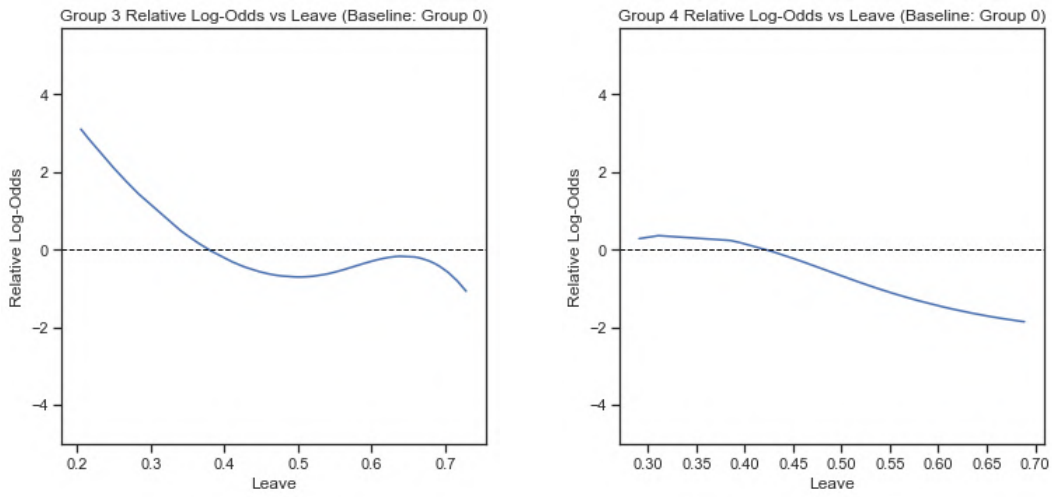
Figure D.14: Clusters 12 and 13 with 1-Step estimation and “Leave” as concomitant variable

D.3 Relative Log-Odds with 2-Step and "Leave" Variable



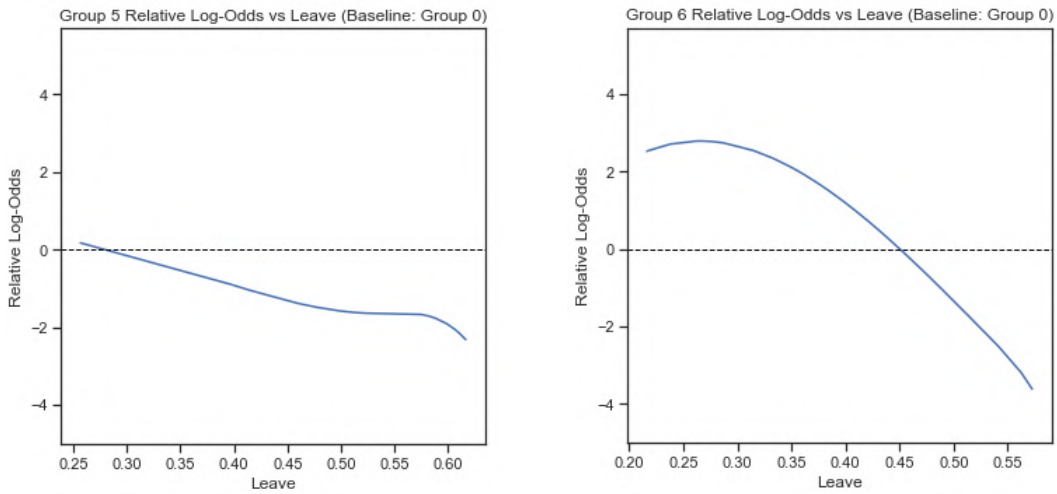
(a) (b)

Figure D.15: Relative Log-Odds in Cluster 1 and 2 with 2-Step estimation and "Leave" as concomitant variable



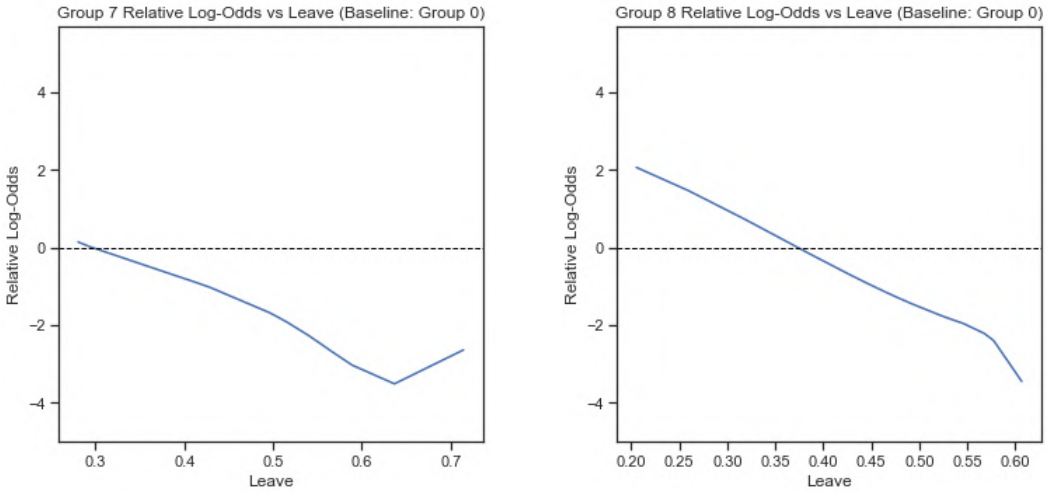
(a) (b)

Figure D.16: Relative Log-Odds in Cluster 3 and 4 with 2-Step estimation and “Leave” as concomitant variable



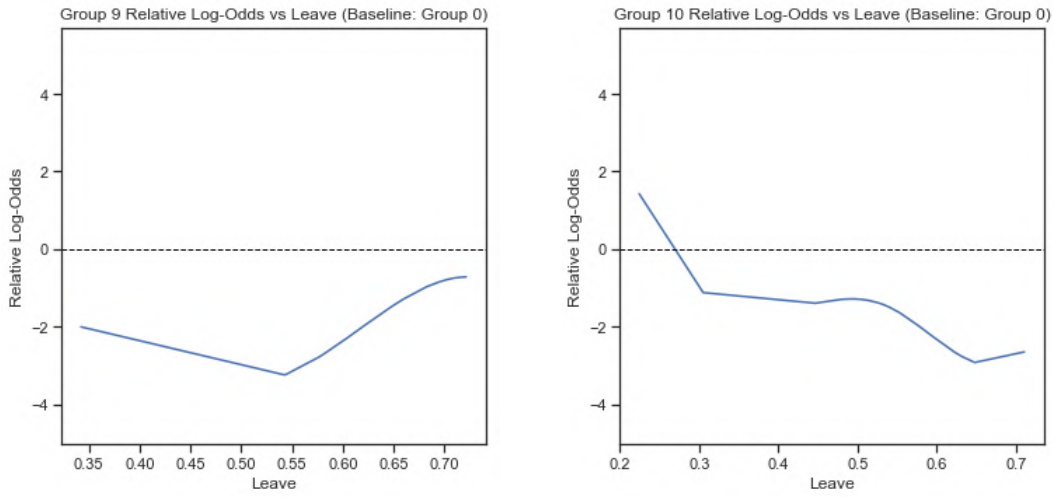
(a) (b)

Figure D.17: Relative Log-Odds in Cluster 5 and 6 with 2-Step estimation and “Leave” as concomitant variable



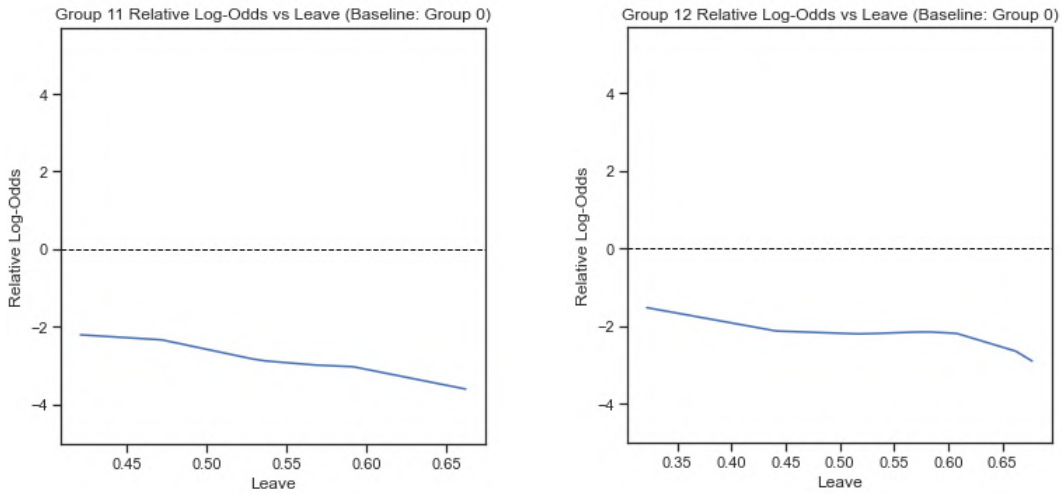
(a) (b)

Figure D.18: Relative Log-Odds in Cluster 7 and 8 with 2-Step estimation and "Leave" as concomitant variable



(a) (b)

Figure D.19: Relative Log-Odds in Cluster 9 and 10 with 2-Step estimation and “Leave” as concomitant variable



(a) (b)

Figure D.20: Relative Log-Odds in Cluster 11 and 12 with 2-Step estimation and “Leave” as concomitant variable

D.4 2-Step Clusters with "Leave" as Concomitant Variable

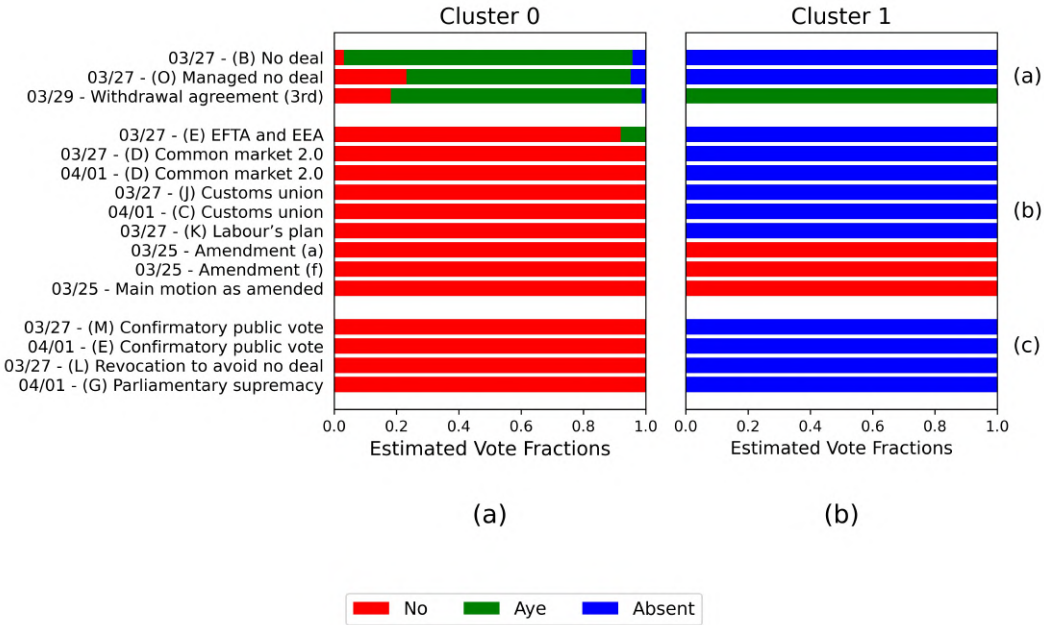


Figure D.21: Clusters 0 and 1 with 2-Step estimation and "Leave" as concomitant variable

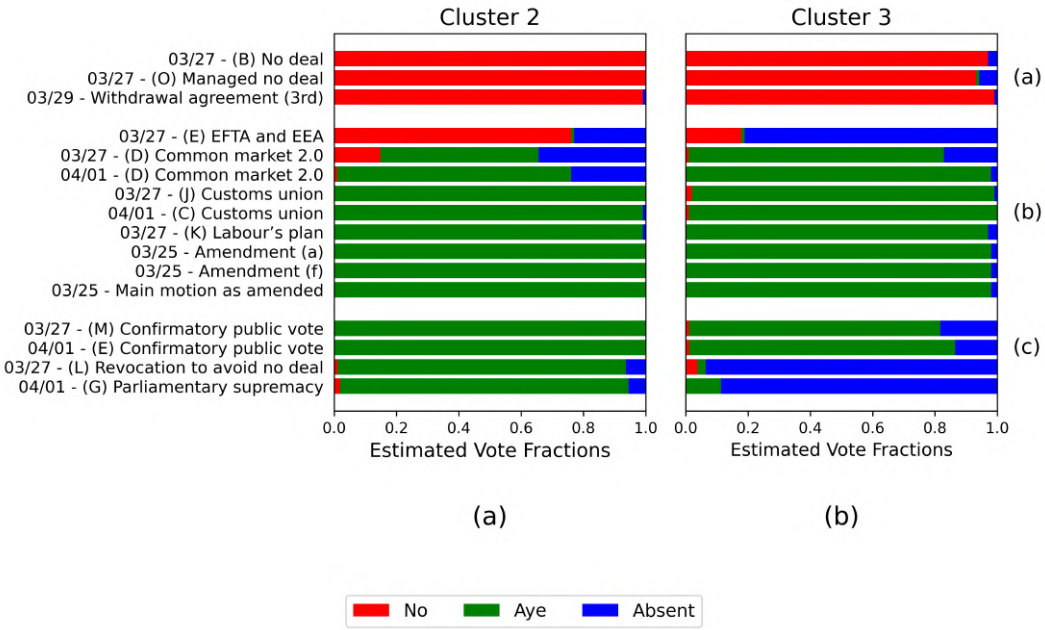


Figure D.22: Clusters 2 and 3 with 2-Step estimation and “Leave” as concomitant variable

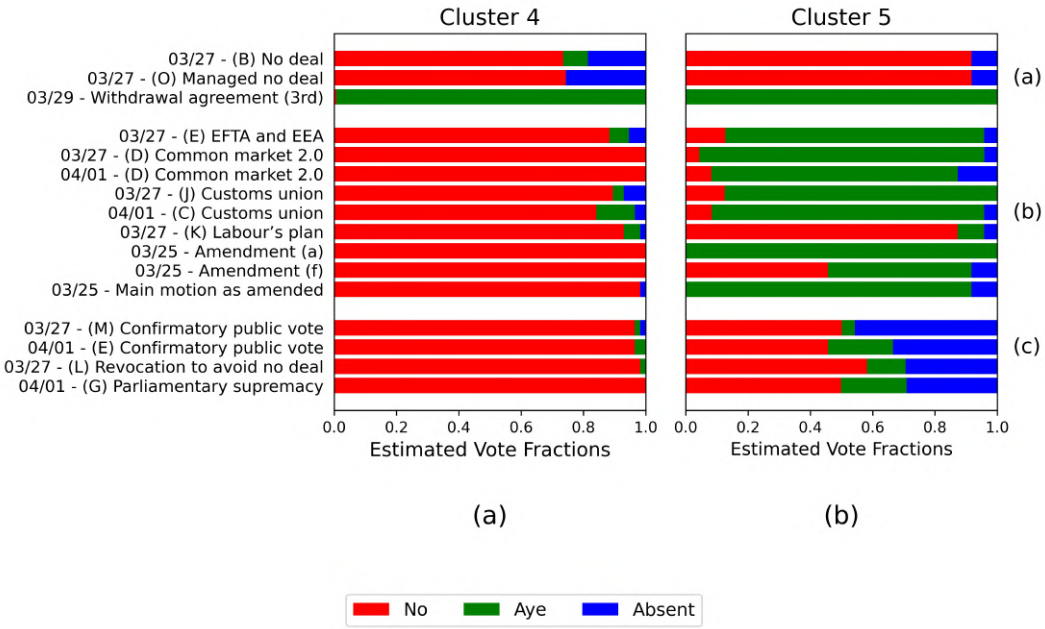


Figure D.23: Clusters 4 and 5 with 2-Step estimation and “Leave” as concomitant variable

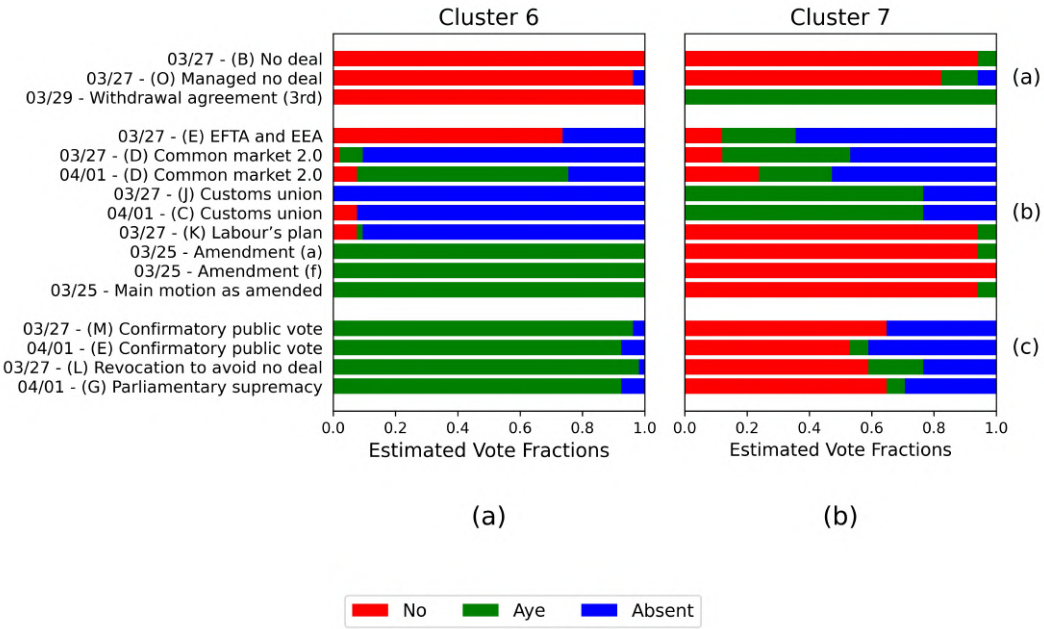


Figure D.24: Clusters 6 and 7 with 2-Step estimation and "Leave" as concomitant variable

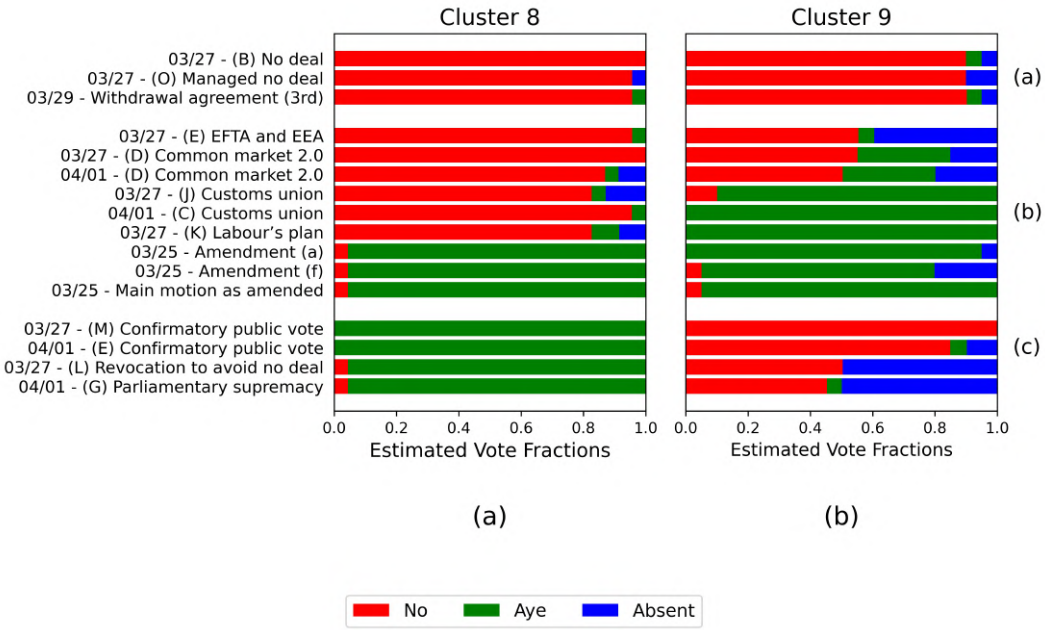


Figure D.25: Clusters 8 and 9 with 2-Step estimation and “Leave” as concomitant variable

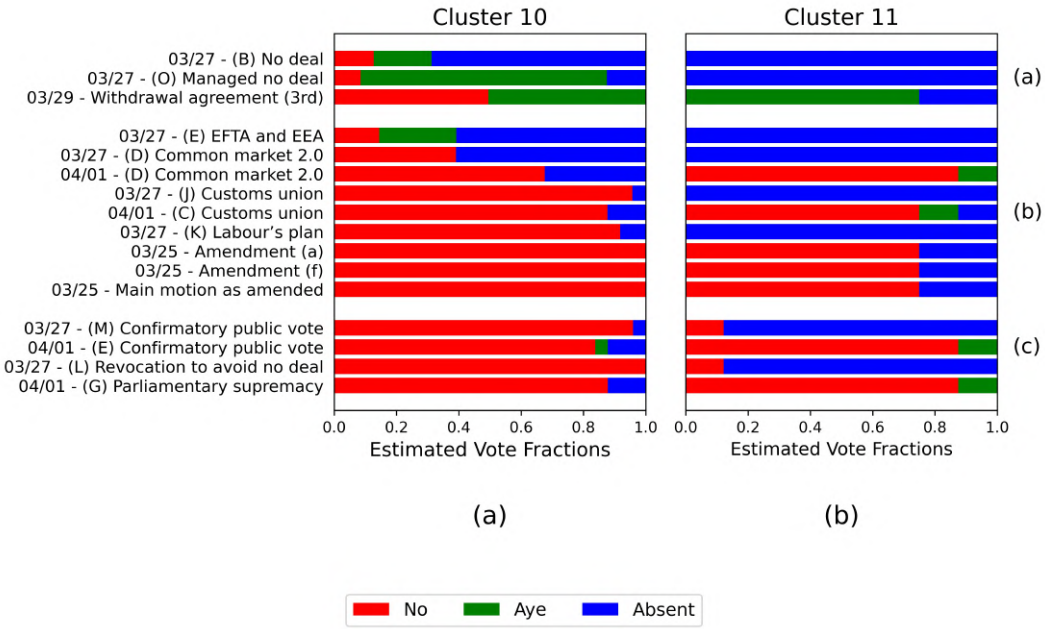


Figure D.26: Clusters 10 and 11 with 2-Step estimation and “Leave” as concomitant variable

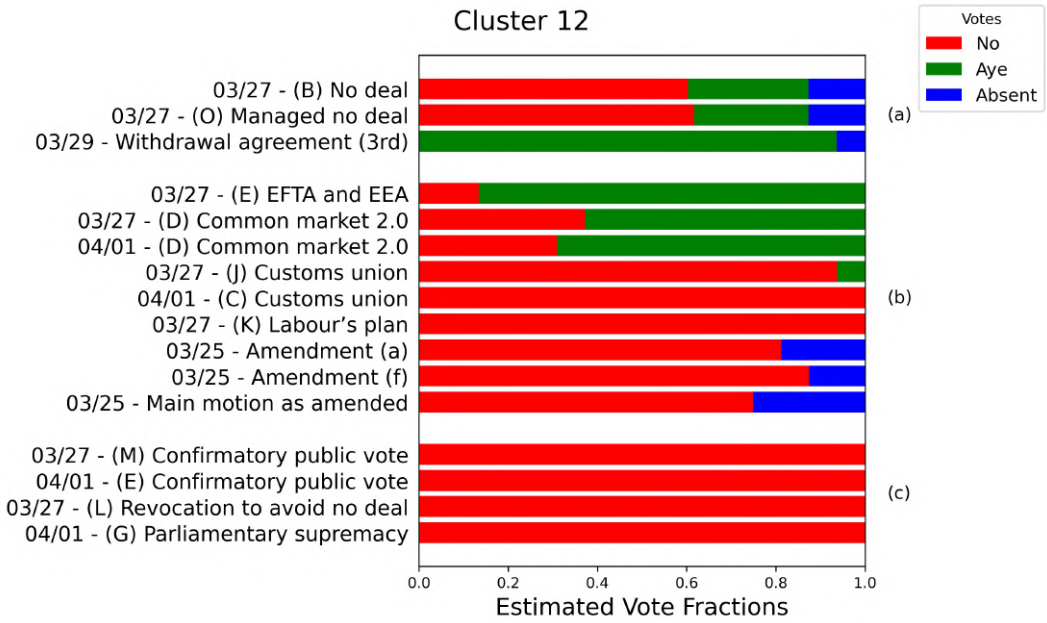
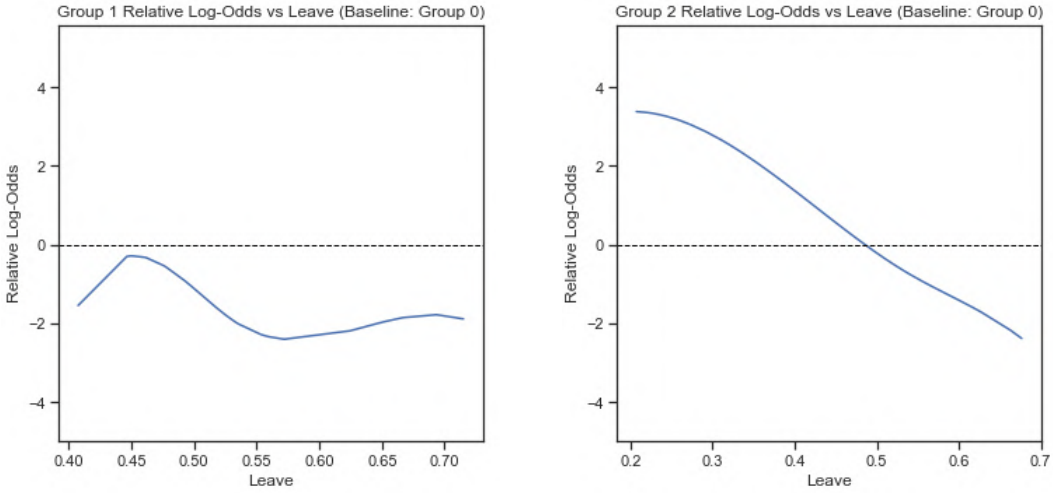


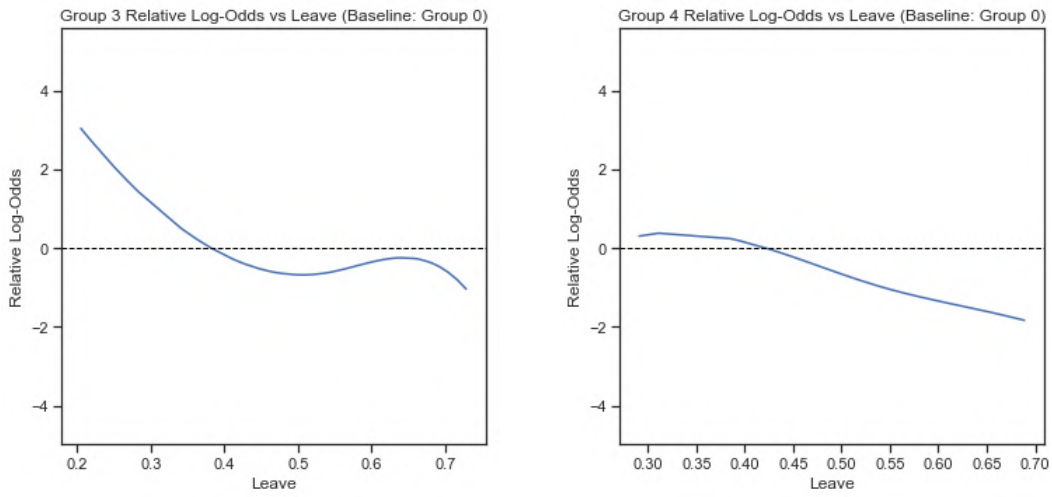
Figure D.27: Cluster 12 with 2-Step estimation and "Leave" as concomitant variable

D.5 Relative Log-Odds with 3-Step and “Leave” Variable



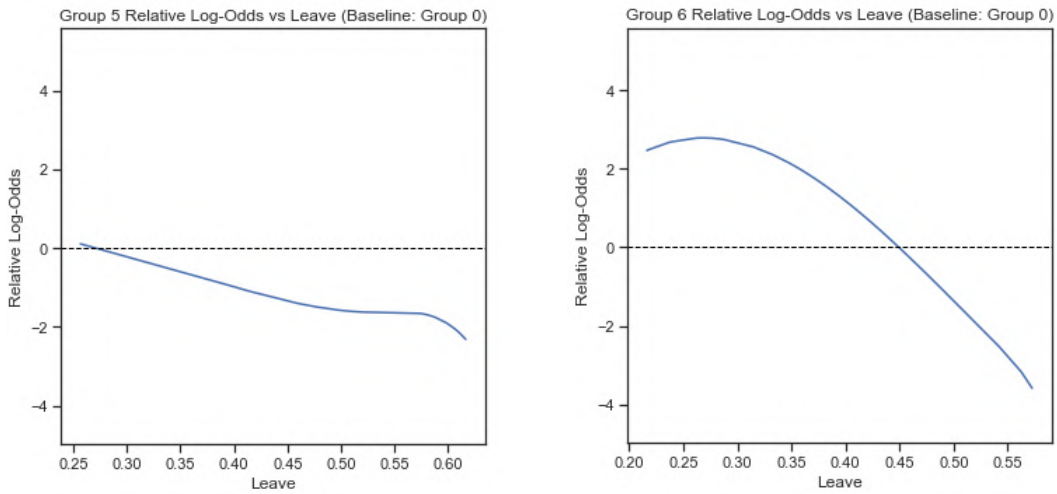
(a) (b)

Figure D.28: Relative Log-Odds in Cluster 1 and 2 with 3-Step estimation and “Leave” as concomitant variable



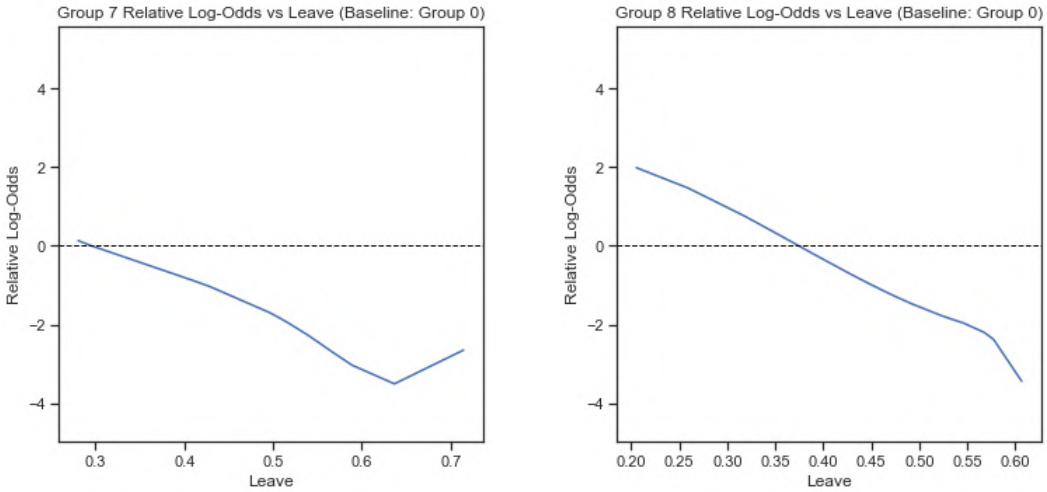
(a) (b)

Figure D.29: Relative Log-Odds in Cluster 3 and 4 with 3-Step estimation and "Leave" as concomitant variable



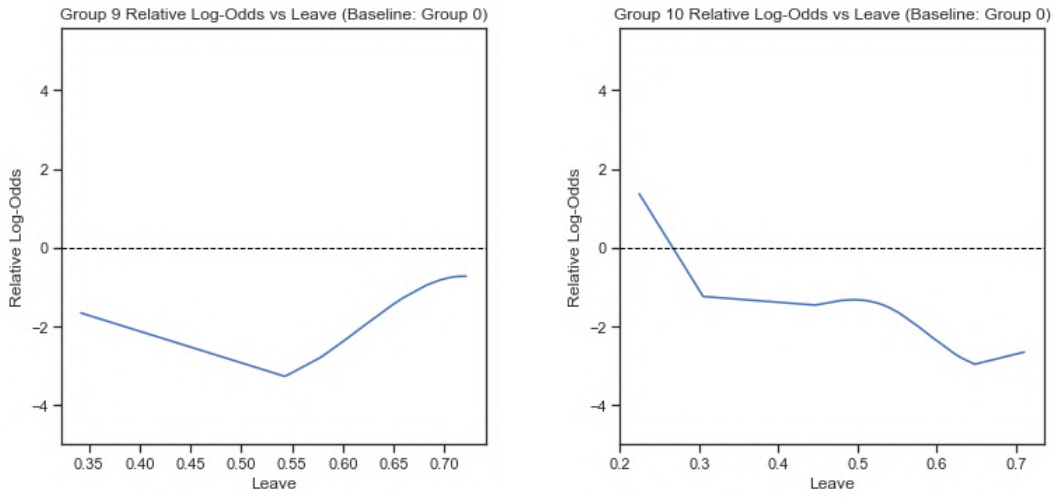
(a) (b)

Figure D.30: Relative Log-Odds in Cluster 5 and 6 with 3-Step estimation and "Leave" as concomitant variable



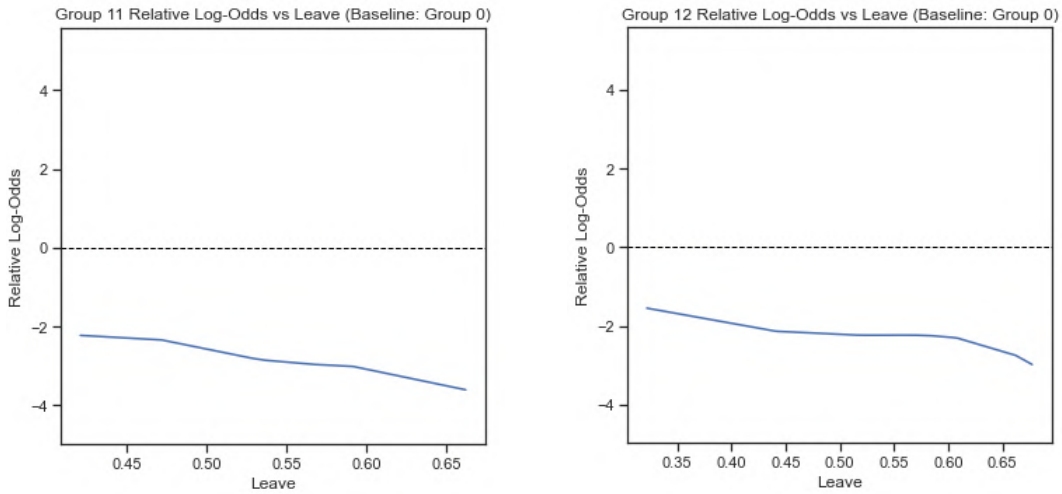
(a) (b)

Figure D.31: Relative Log-Odds in Cluster 7 and 8 with 3-Step estimation and “Leave” as concomitant variable



(a) (b)

Figure D.32: Relative Log-Odds in Cluster 9 and 10 with 3-Step estimation and "Leave" as concomitant variable



(a) (b)

Figure D.33: Relative Log-Odds in Cluster 11 and 12 with 3-Step estimation and "Leave" as concomitant variable

D.6 3-Step Clusters with “Leave” as Concomitant Variable

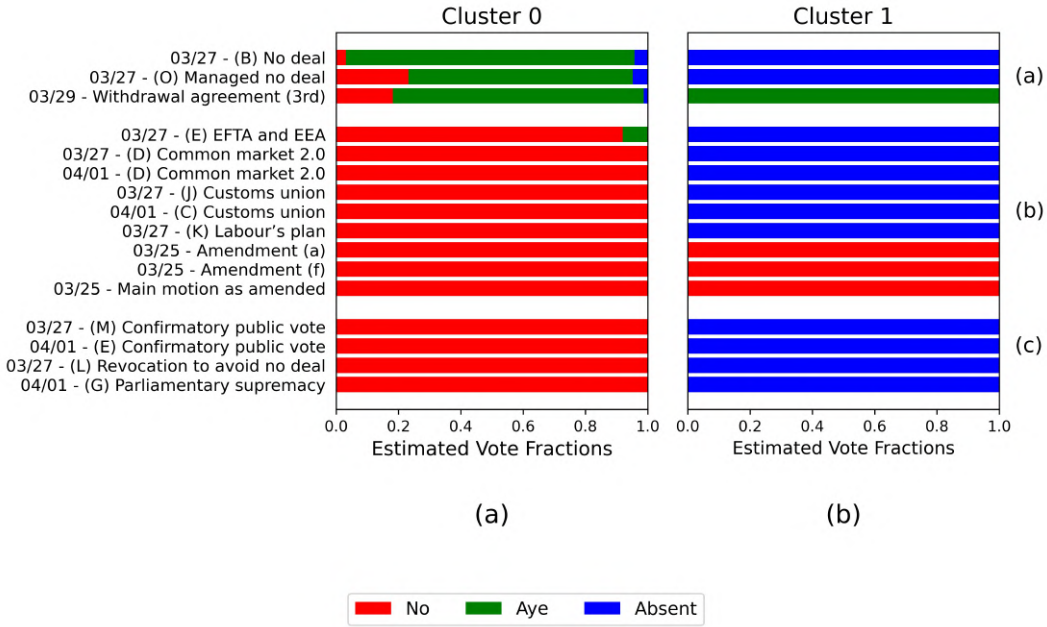


Figure D.34: Clusters 0 and 1 with 3-Step estimation and “Leave” as concomitant variable

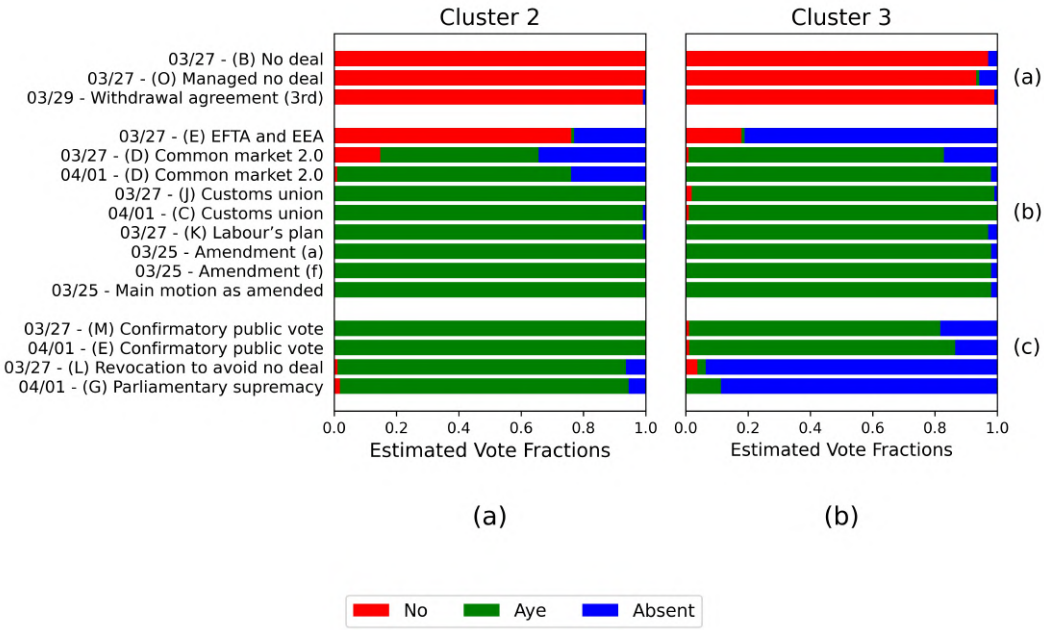


Figure D.35: Clusters 2 and 3 with 3-Step estimation and "Leave" as concomitant variable

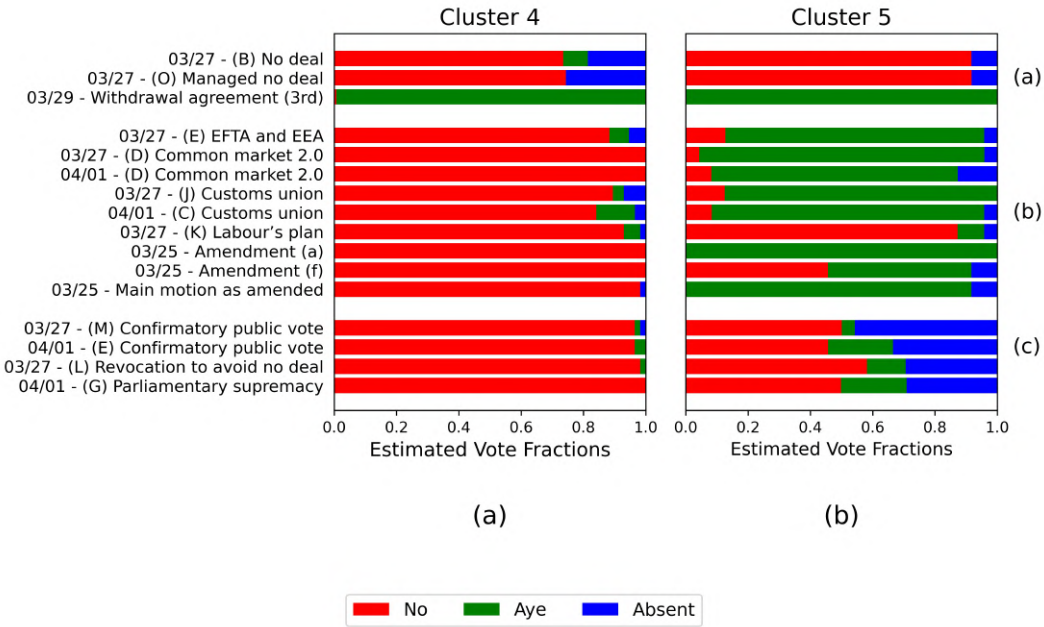


Figure D.36: Clusters 4 and 5 with 3-Step estimation and "Leave" as concomitant variable

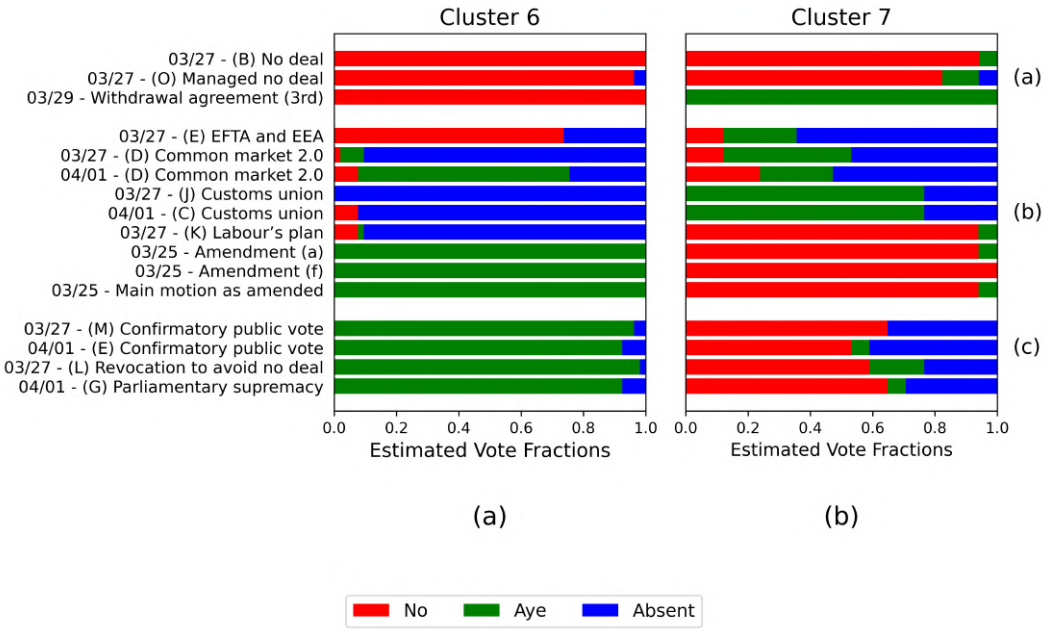


Figure D.37: Clusters 6 and 7 with 3-Step estimation and “Leave” as concomitant variable

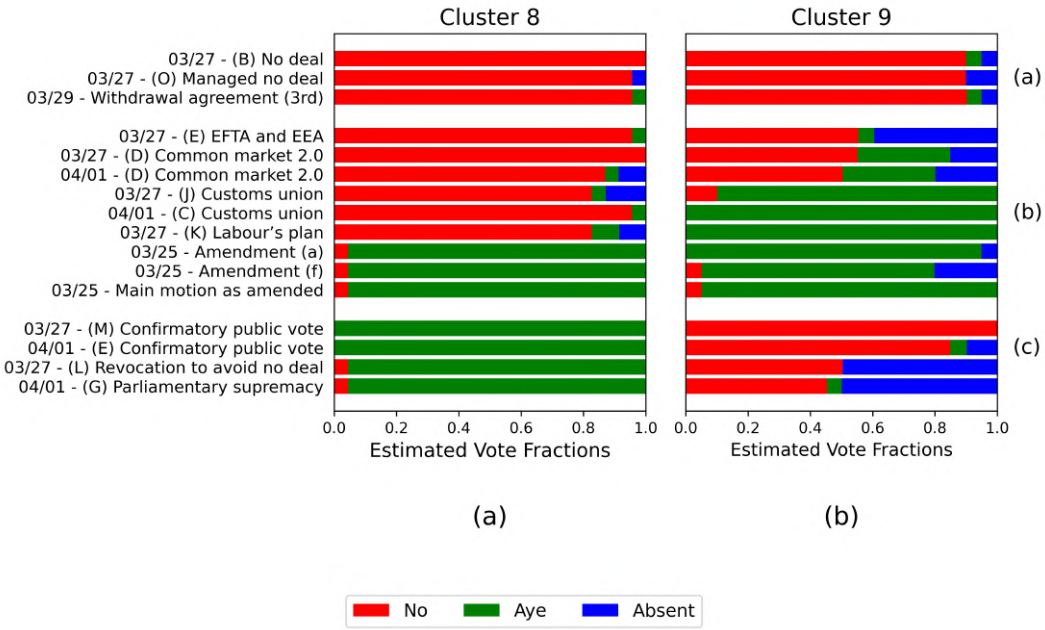


Figure D.38: Clusters 8 and 9 with 3-Step estimation and "Leave" as concomitant variable

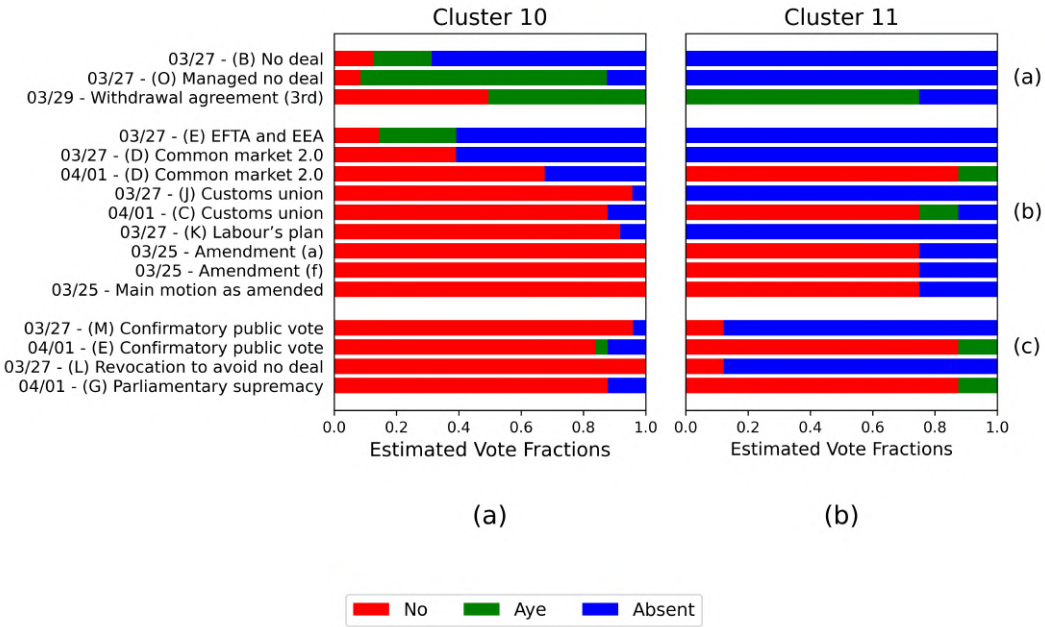


Figure D.39: Clusters 10 and 11 with 3-Step estimation and "Leave" as concomitant variable

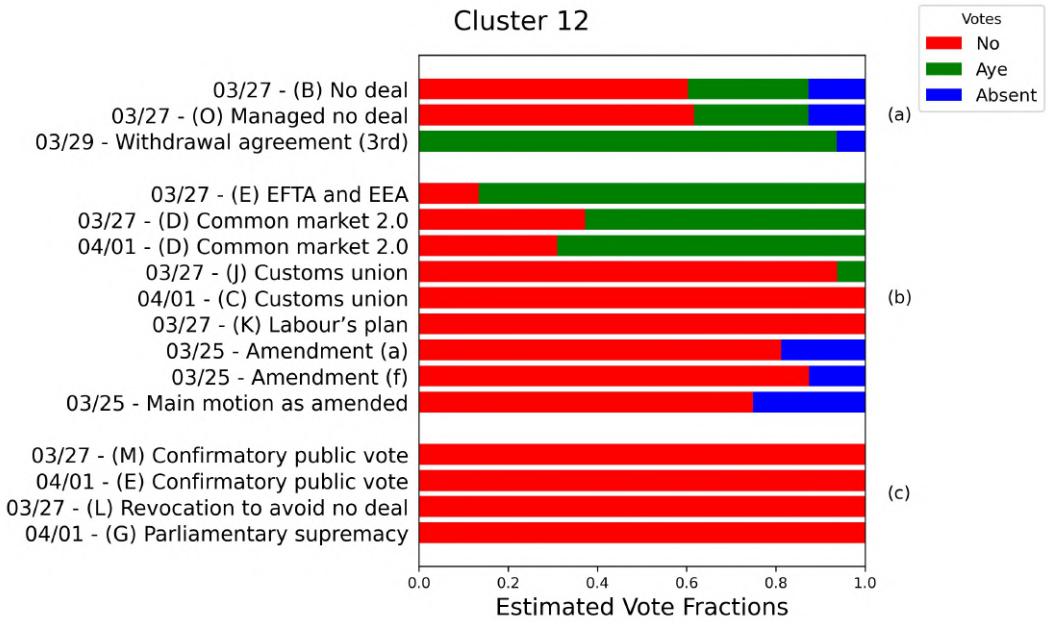
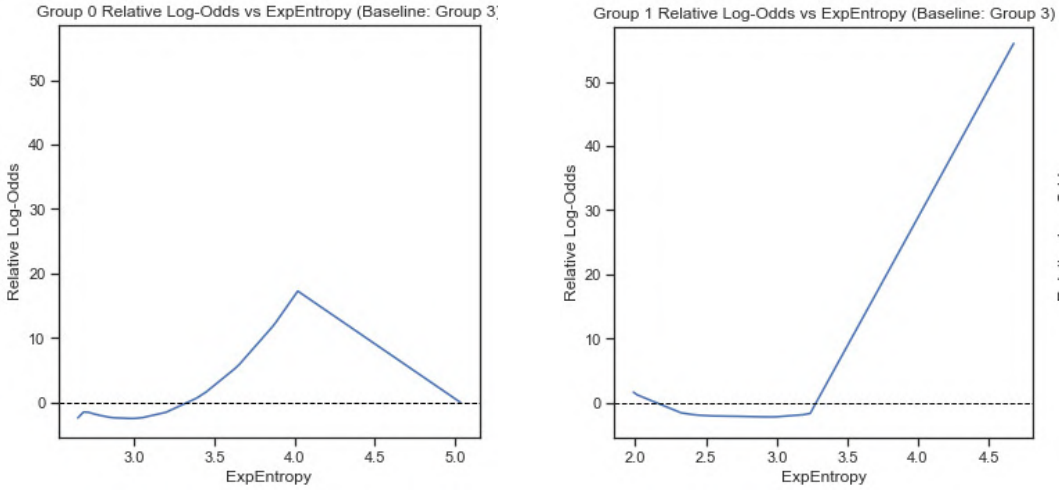


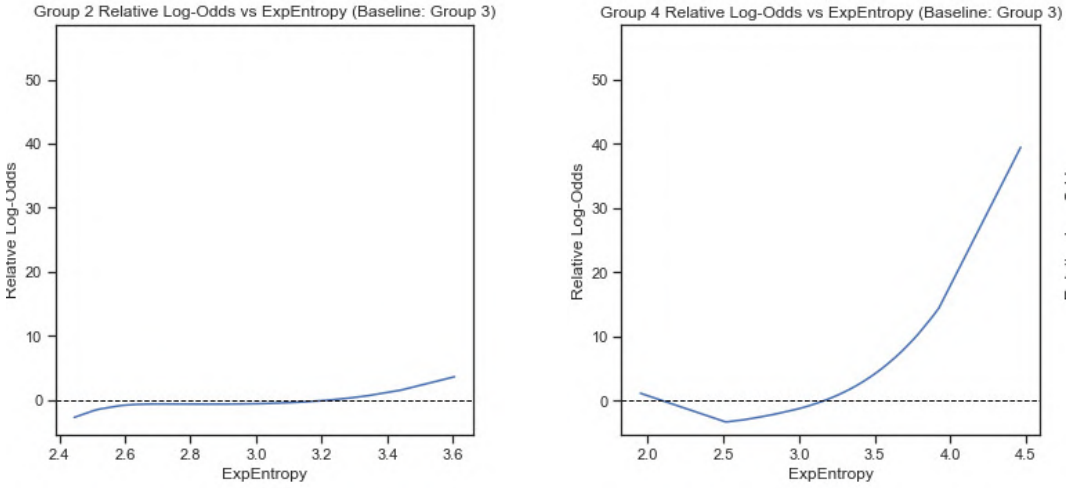
Figure D.40: Cluster 12 with 3-Step estimation and “Leave” as concomitant variable

D.7 Relative Log-Odds with 1-Step and "ExpEntropy" Variable



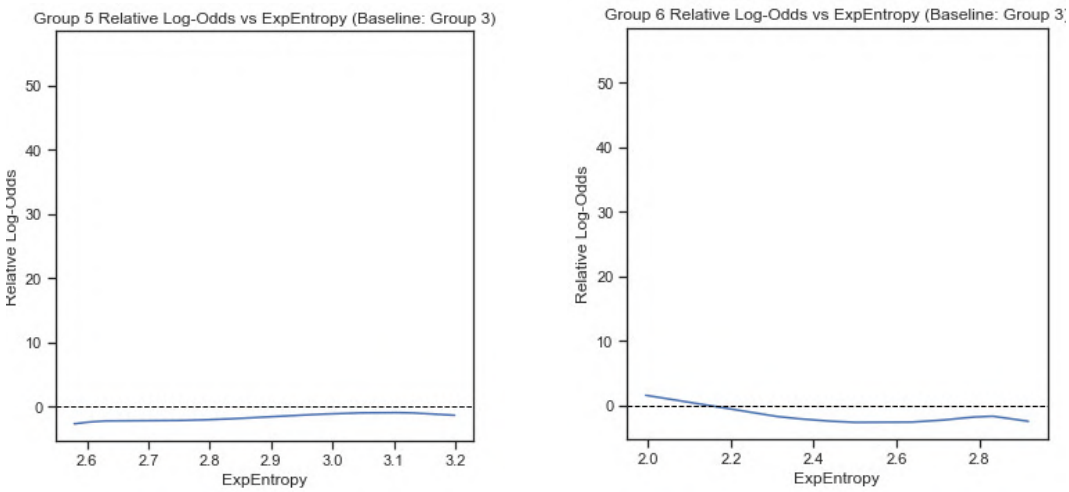
(a) (b)

Figure D.41: Relative Log-Odds in Cluster 0 and 1 with 1-Step estimation and "ExpEntropy" as concomitant variable



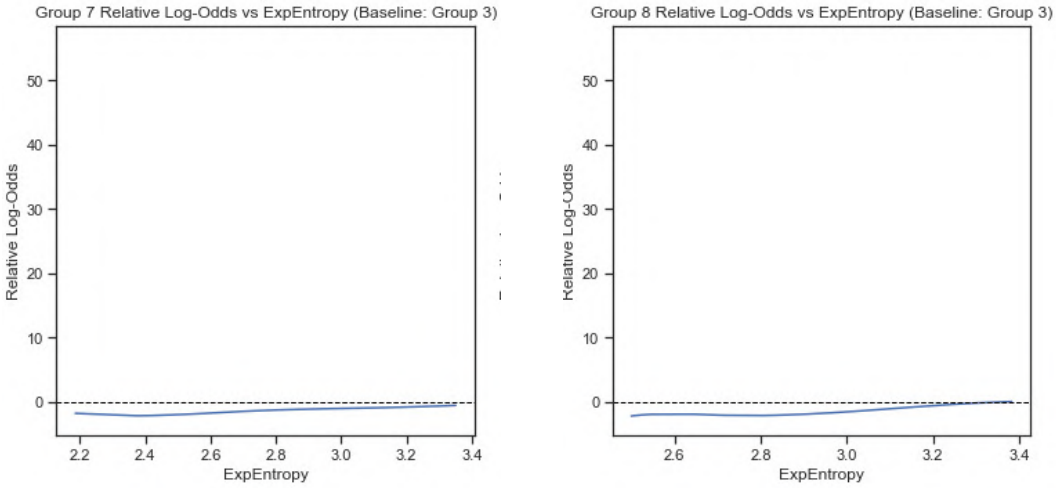
(a) (b)

Figure D.42: Relative Log-Odds in Cluster 2 and 4 with 1-Step estimation and “ExpEntropy” as concomitant variable



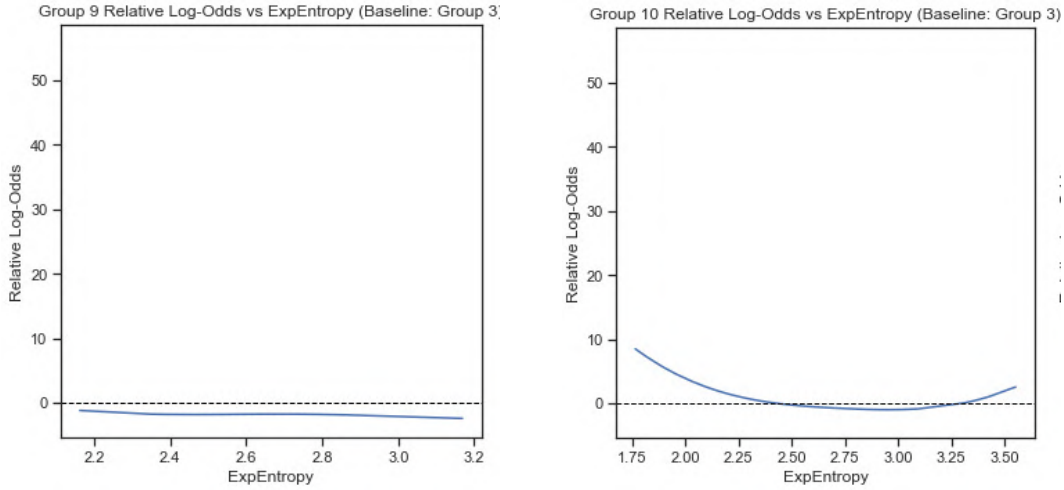
(a) (b)

Figure D.43: Relative Log-Odds in Cluster 5 and 6 with 1-Step estimation and “ExpEntropy” as concomitant variable



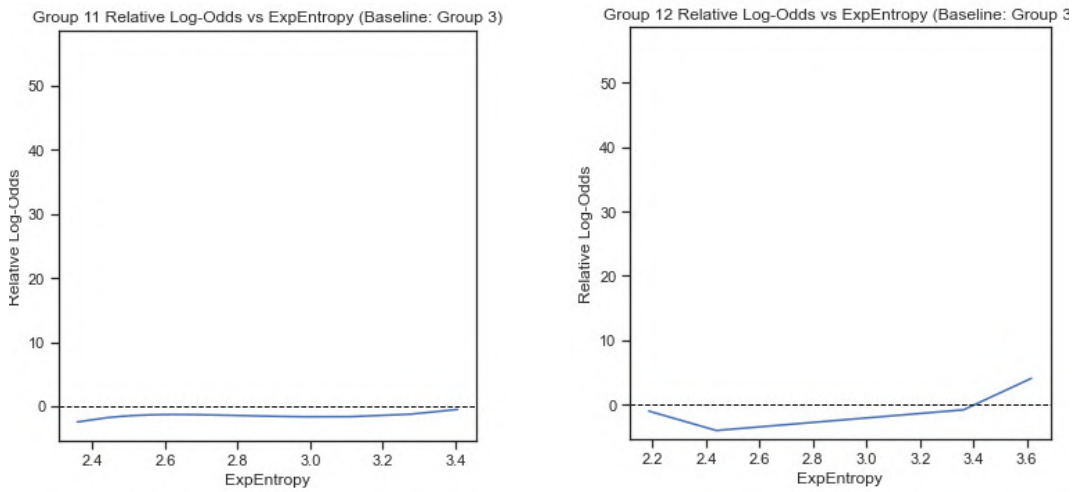
(a) (b)

Figure D.44: Relative Log-Odds in Cluster 7 and 8 with 1-Step estimation and "ExpEntropy" as concomitant variable



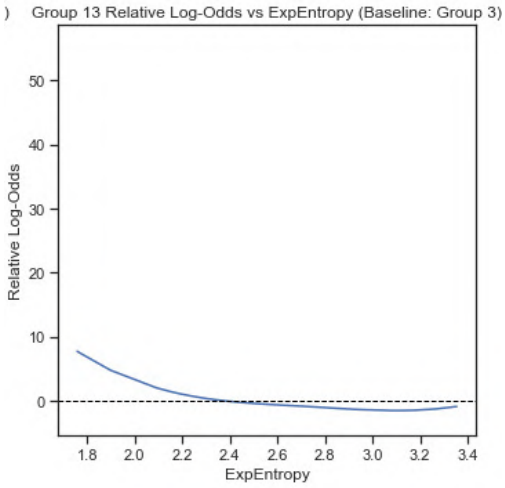
(a) (b)

Figure D.45: Relative Log-Odds in Cluster 9 and 10 with 1-Step estimation and “ExpEntropy” as concomitant variable



(a) (b)

Figure D.46: Relative Log-Odds in Cluster 11 and 12 with 1-Step estimation and “ExpEntropy” as concomitant variable



(a)

Figure D.47: Relative Log-Odds in Cluster 13 with 1-Step estimation and "ExpEntropy" as concomitant variable

D.8 1-Step Clusters with “ExpEntropy” as Concomitant Variable

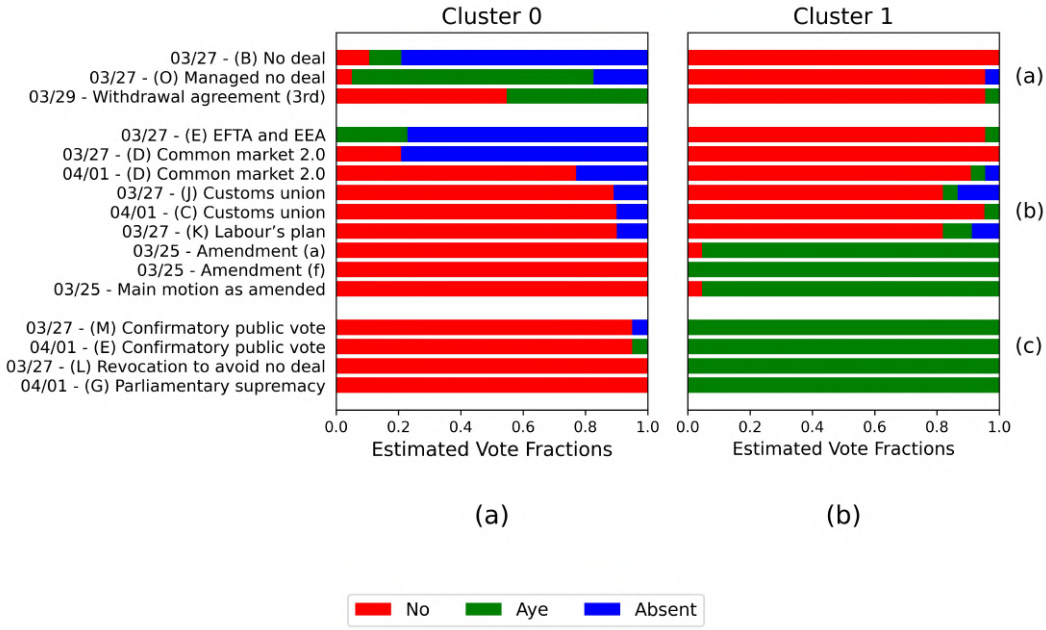


Figure D.48: Clusters 0 and 1 with 1-Step estimation and “ExpEntropy” as concomitant variable

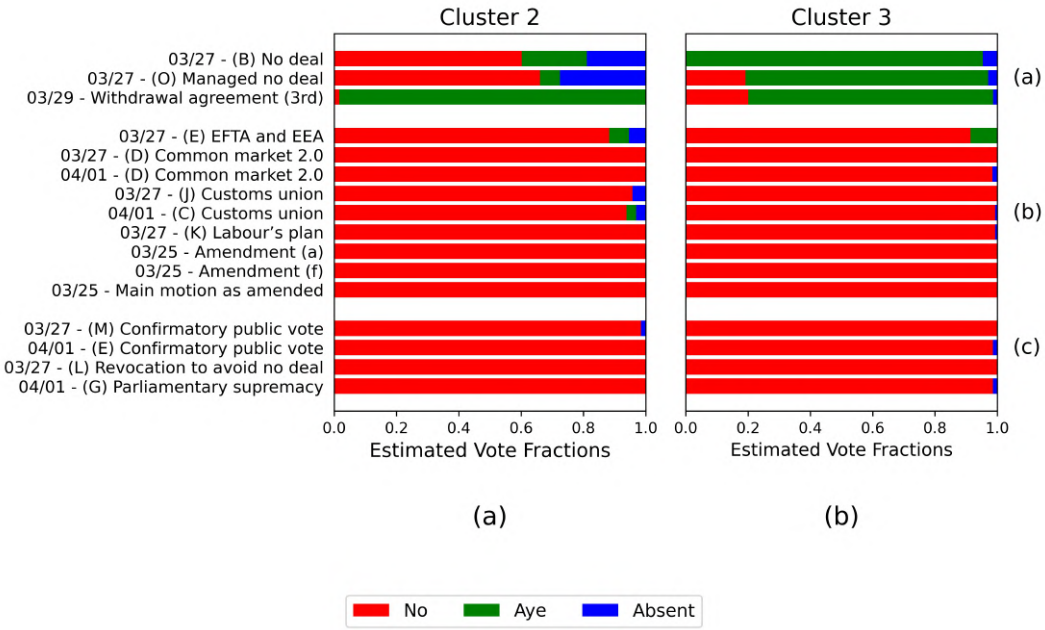


Figure D.49: Clusters 2 and 3 with 1-Step estimation and "ExpEntropy" as concomitant variable

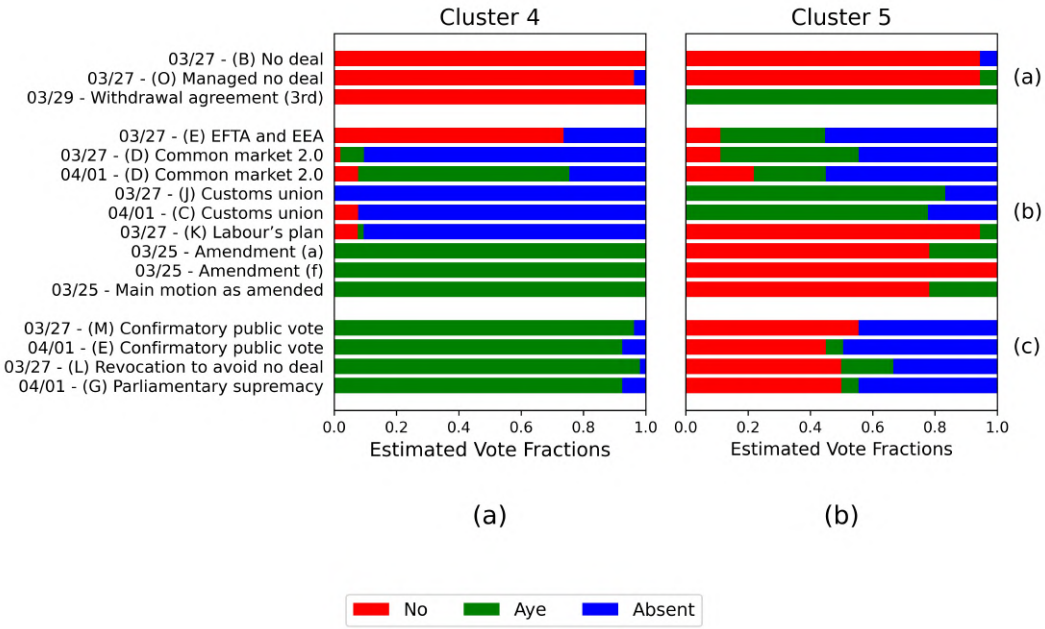


Figure D.50: Clusters 4 and 5 with 1-Step estimation and "ExpEntropy" as concomitant variable

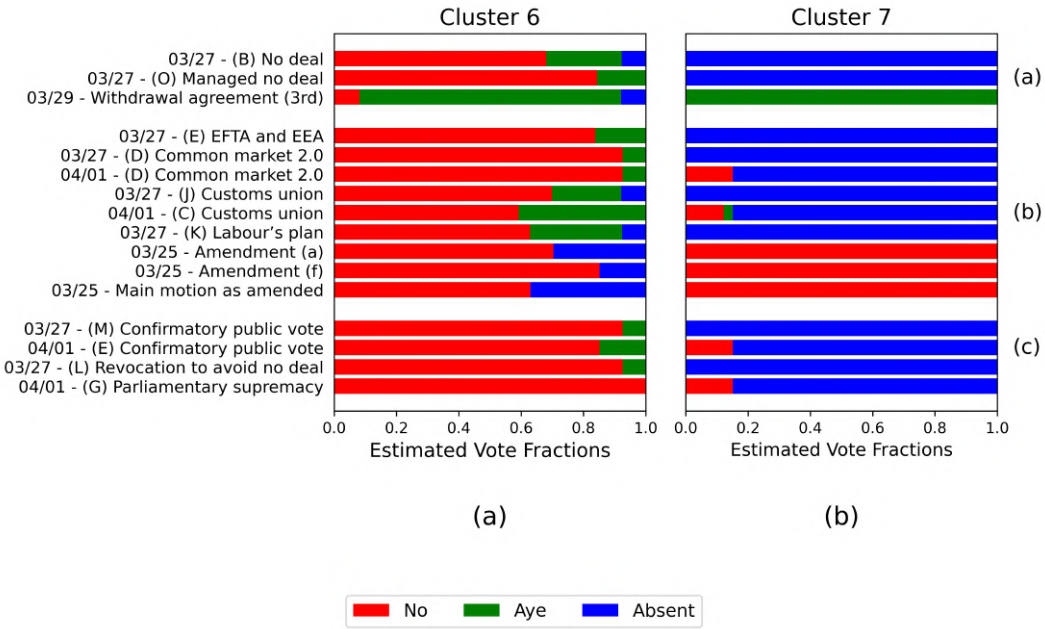


Figure D.51: Clusters 6 and 7 with 1-Step estimation and “ExpEntropy” as concomitant variable

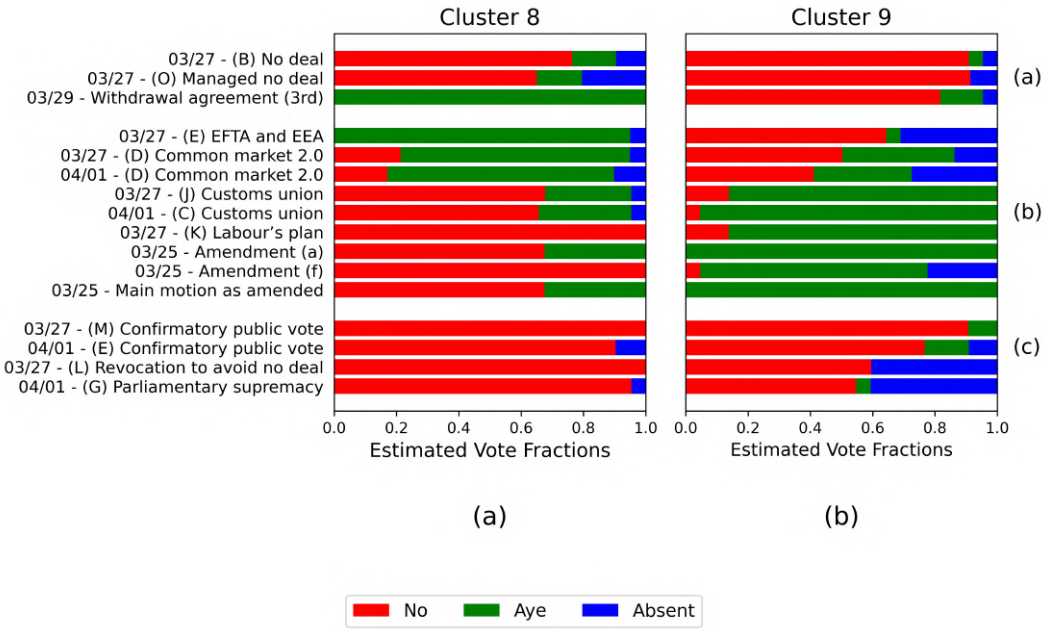


Figure D.52: Clusters 8 and 9 with 1-Step estimation and "ExpEntropy" as concomitant variable

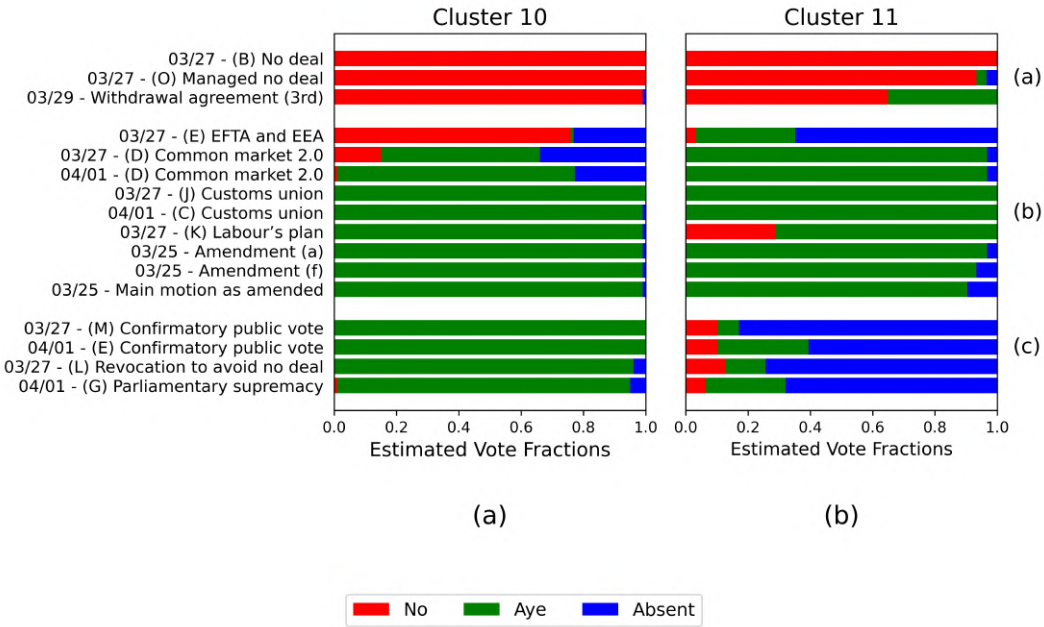


Figure D.53: Clusters 10 and 11 with 1-Step estimation and "ExpEntropy" as concomitant variable

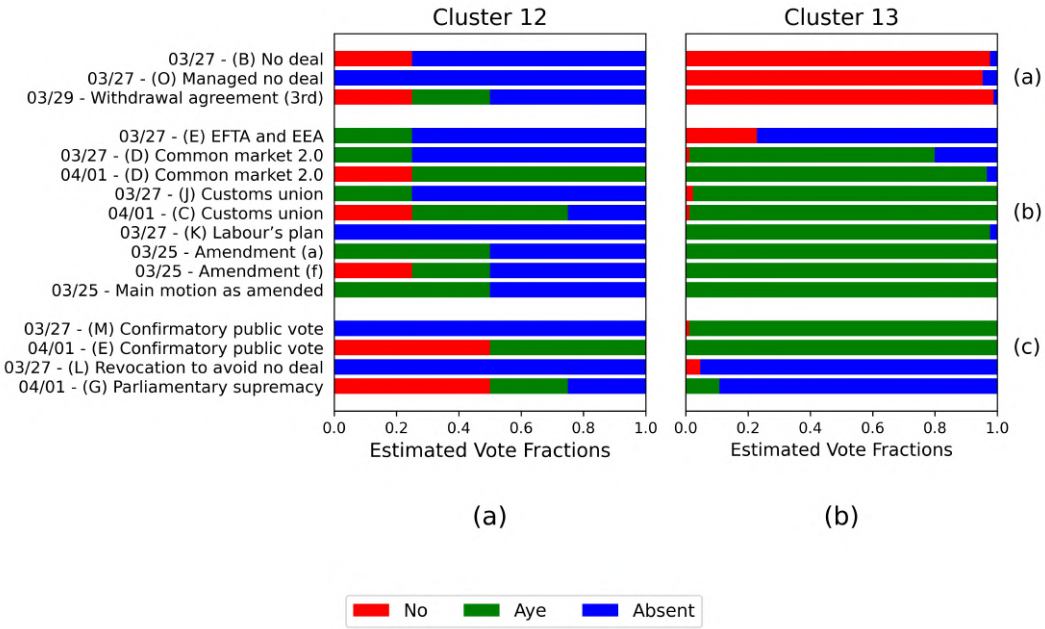
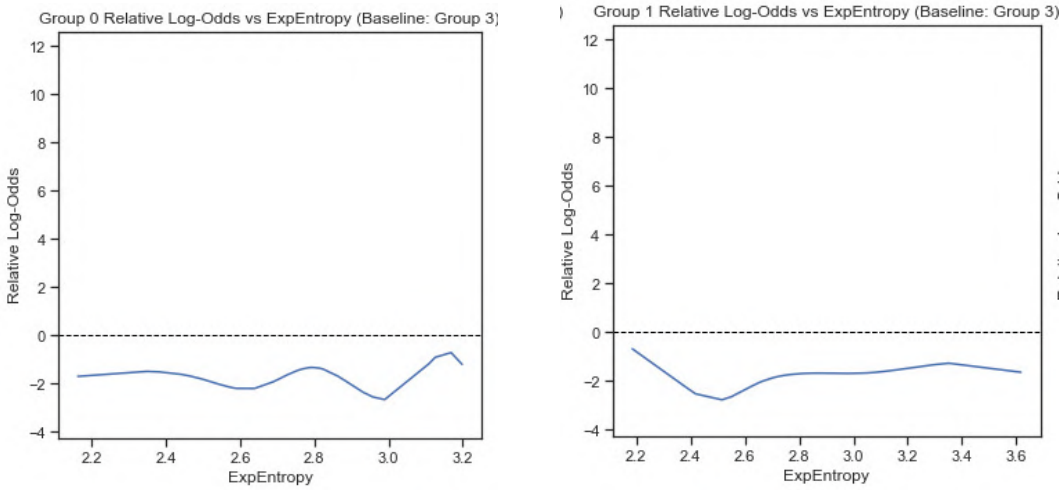


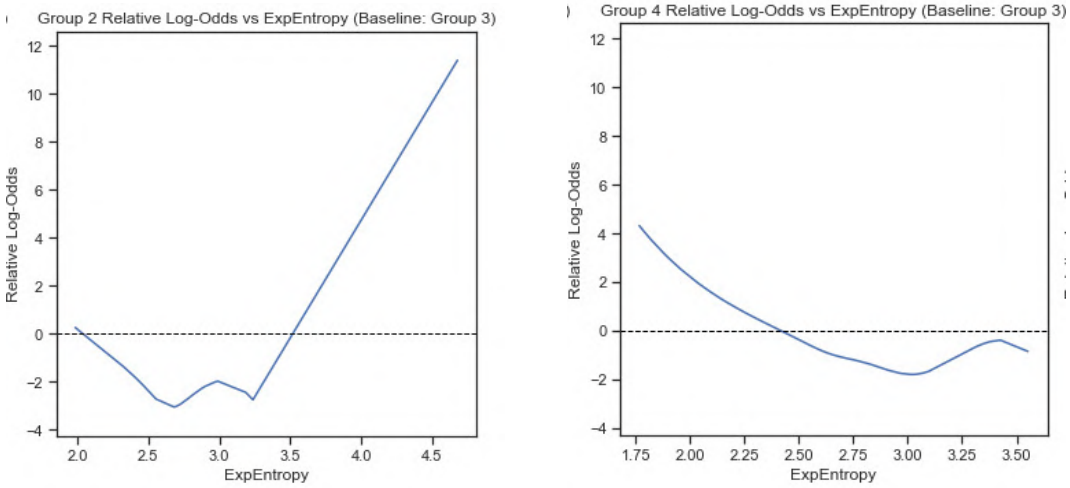
Figure D.54: Clusters 12 and 13 with 1-Step estimation and “ExpEntropy” as concomitant variable

D.9 Relative Log-Odds with 2-Step and "ExpEntropy" Variable



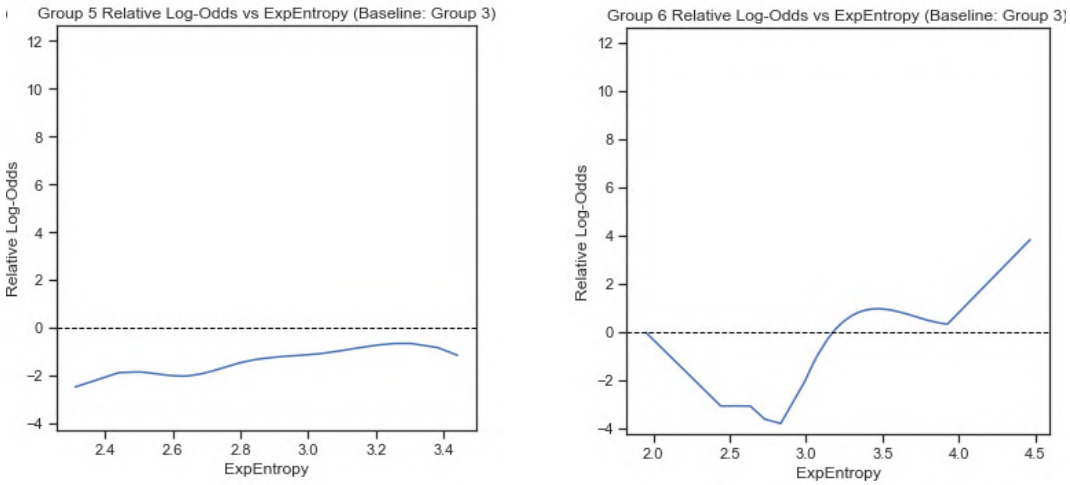
(a) (b)

Figure D.55: Relative Log-Odds in Cluster 0 and 1 with 2-Step estimation and "ExpEntropy" as concomitant variable



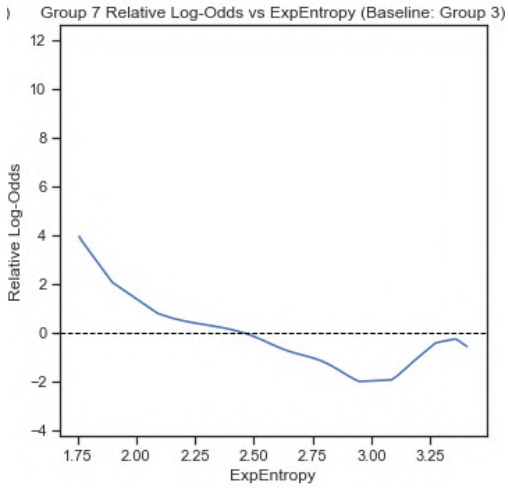
(a) (b)

Figure D.56: Relative Log-Odds in Cluster 2 and 4 with 2-Step estimation and “ExpEntropy” as concomitant variable



(a) (b)

Figure D.57: Relative Log-Odds in Cluster 5 and 6 with 2-Step estimation and “ExpEntropy” as concomitant variable



(a)

Figure D.58: Relative Log-Odds in Cluster 7 with 2-Step estimation and “ExpEntropy” as concomitant variable

D.10 2-Step Clusters with “ExpEntropy” as Concomitant Variable

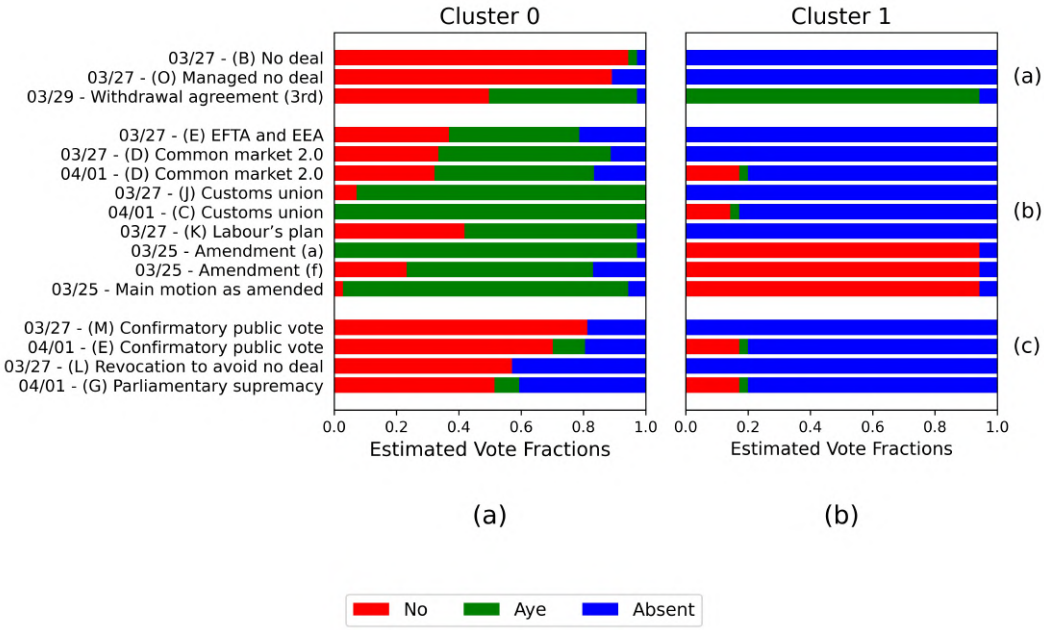


Figure D.59: Clusters 0 and 1 with 2-Step estimation and “ExpEntropy” as concomitant variable

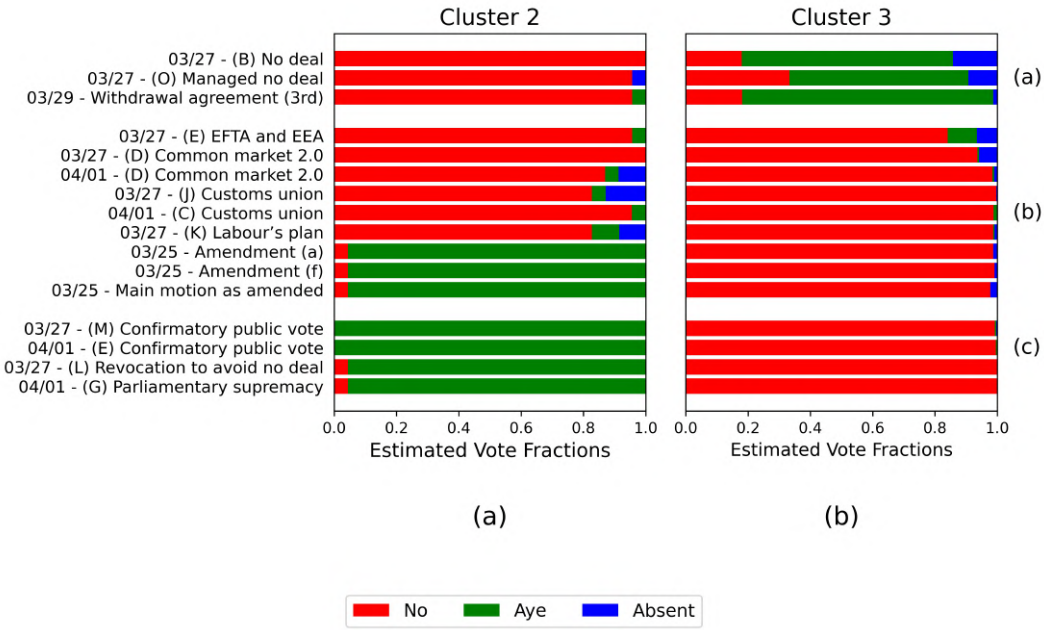


Figure D.60: Clusters 2 and 3 with 2-Step estimation and "ExpEntropy" as concomitant variable

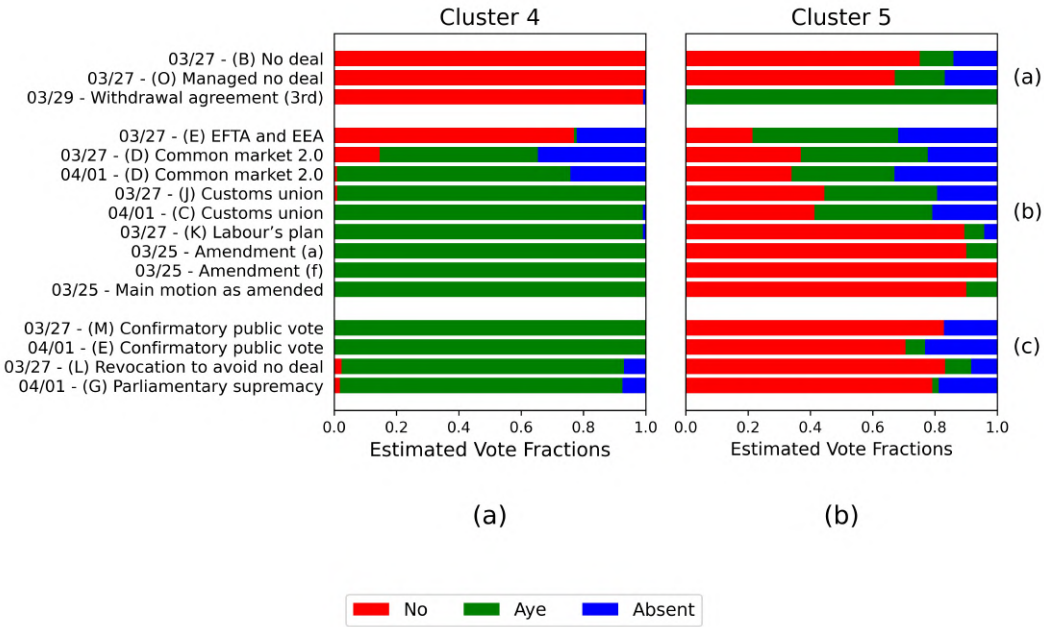


Figure D.61: Clusters 4 and 5 with 2-Step estimation and "ExpEntropy" as concomitant variable

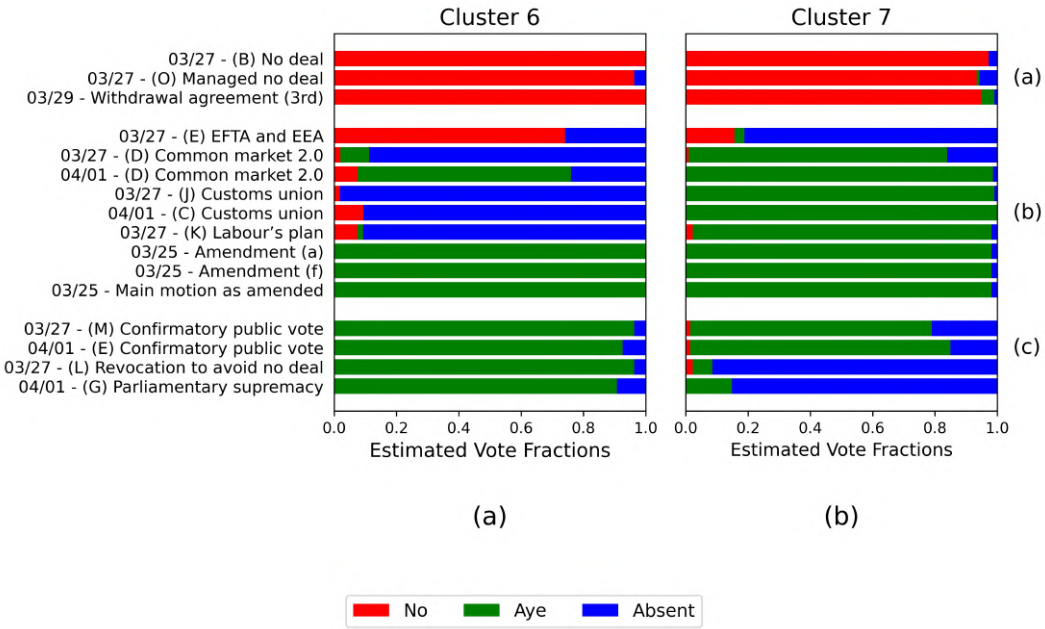
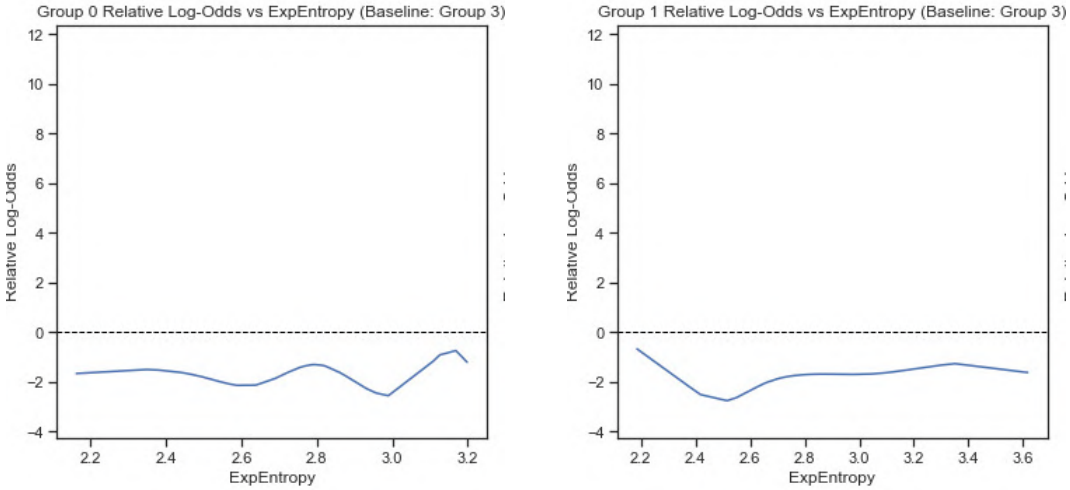


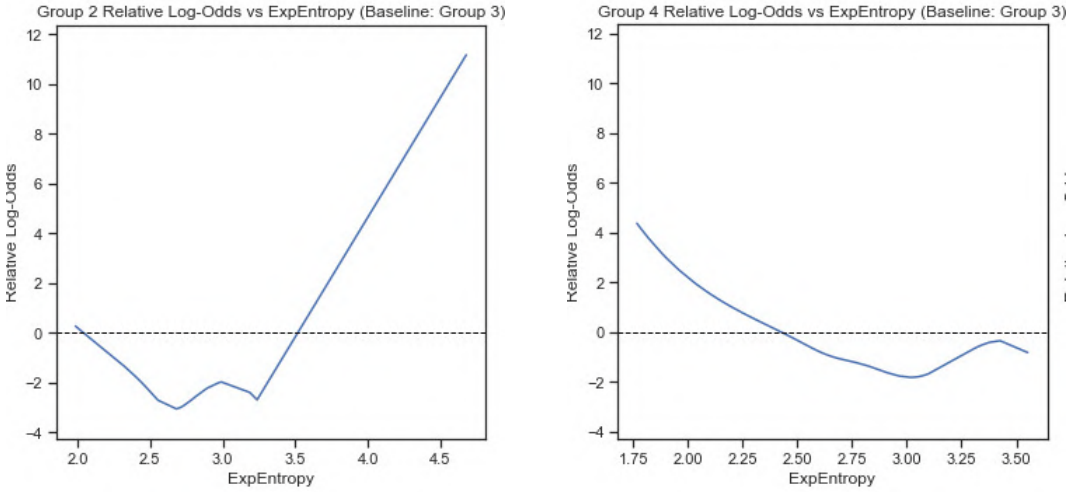
Figure D.62: Clusters 6 and 7 with 2-Step estimation and “ExpEntropy” as concomitant variable

D.11 Relative Log-Odds with 3-Step and "ExpEntropy" Variable



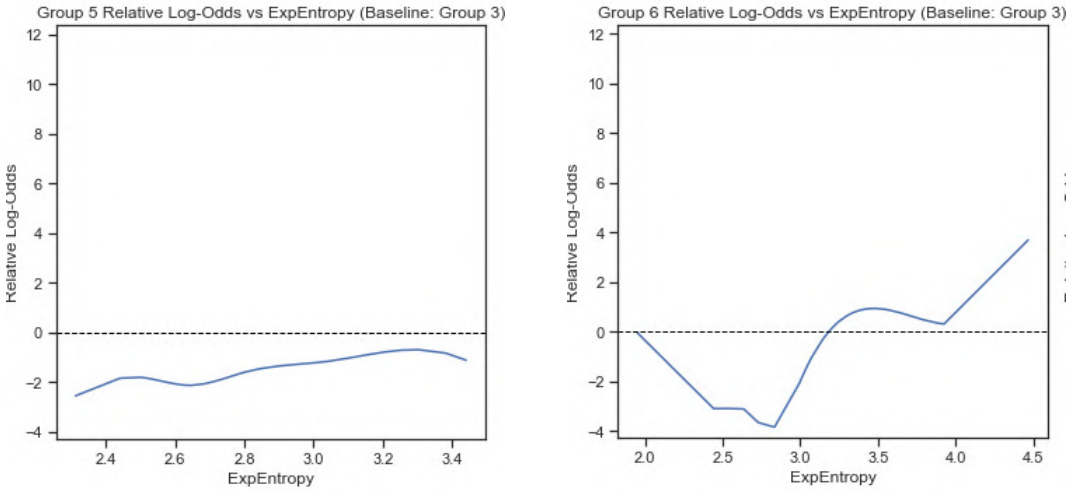
(a) (b)

Figure D.63: Relative Log-Odds in Cluster 0 and 1 with 3-Step estimation and "ExpEntropy" as concomitant variable



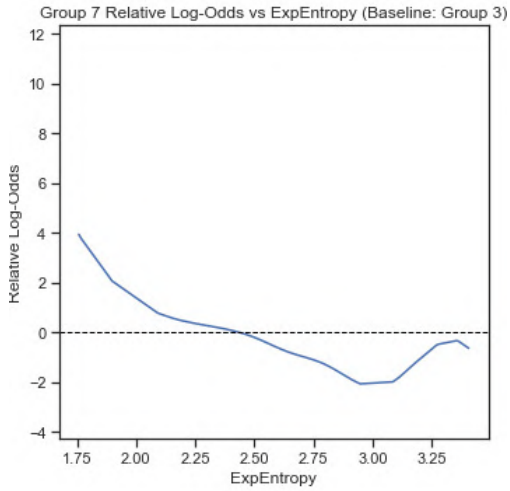
(a) (b)

Figure D.64: Relative Log-Odds in Cluster 2 and 4 with 3-Step estimation and “ExpEntropy” as concomitant variable



(a) (b)

Figure D.65: Relative Log-Odds in Cluster 5 and 6 with 3-Step estimation and “ExpEntropy” as concomitant variable



(a)

Figure D.66: Relative Log-Odds in Cluster 7 with 3-Step estimation and “ExpEntropy” as concomitant variable

D.12 3-Step Clusters with “ExpEntropy” as Concomitant Variable

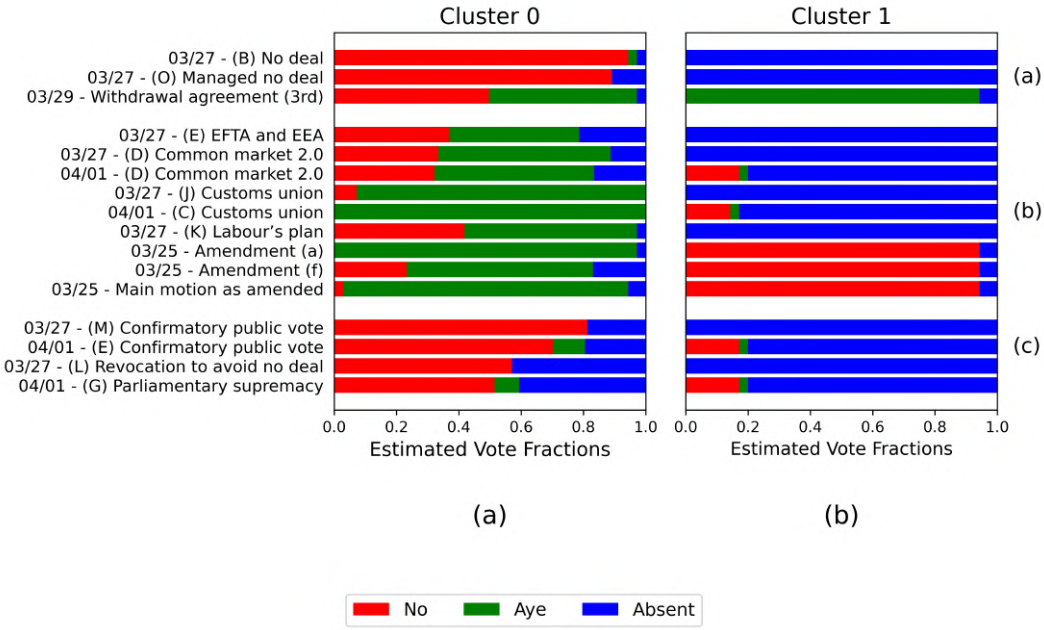


Figure D.67: Clusters 0 and 1 with 3-Step estimation and “ExpEntropy” as concomitant variable

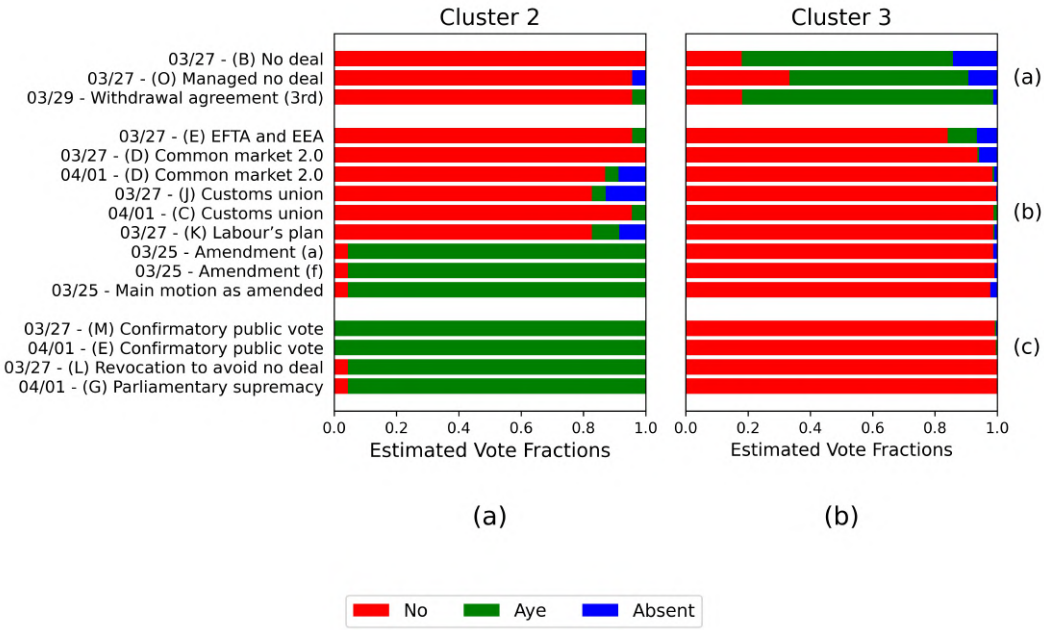


Figure D.68: Clusters 2 and 3 with 3-Step estimation and "ExpEntropy" as concomitant variable

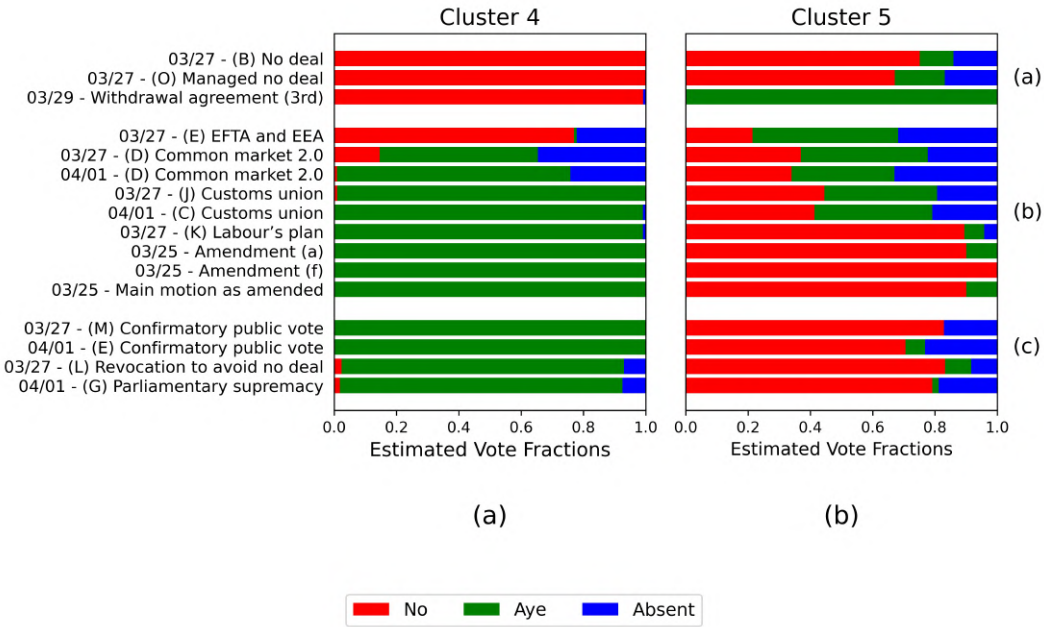


Figure D.69: Clusters 4 and 5 with 3-Step estimation and "ExpEntropy" as concomitant variable

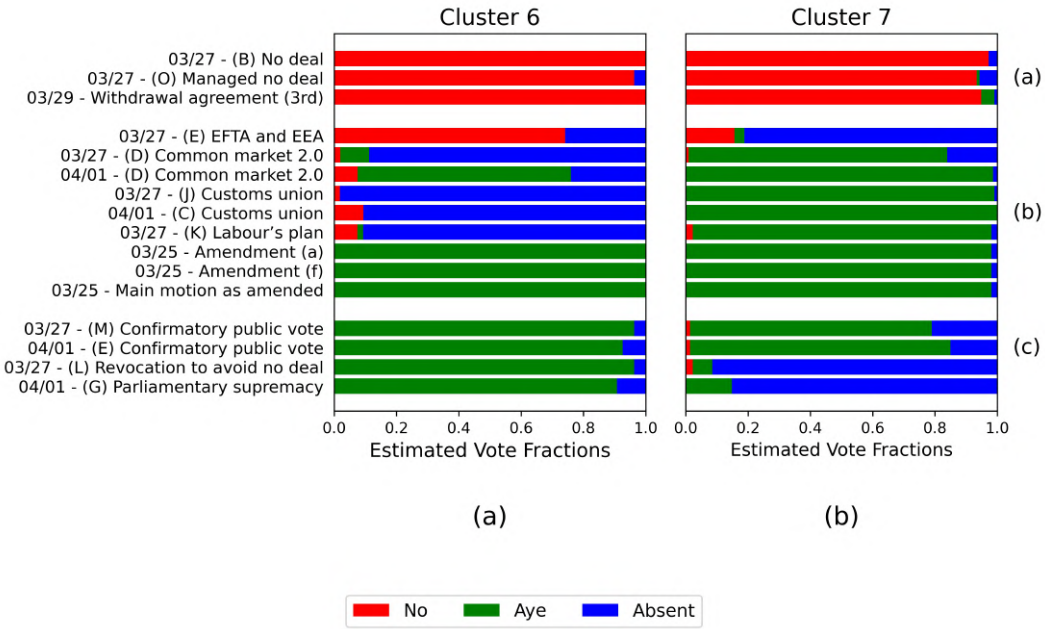


Figure D.70: Clusters 6 and 7 with 3-Step estimation and “ExpEntropy” as concomitant variable