An MM Algorithm for Fixed Effects Multinomial Logit Models

Mingli Chen
University of Warwick

Ian Meeker
Charles River Associates

Marc Rysman
Boston University

Shuang Wang
Charles River Associates*

August 12, 2025

Abstract

Estimating multinomial models with many fixed effects faces prohibitive computational challenges. We develop a new method based on the Minorization-Maximization (MM) algorithm to address this issue. The method linearizes the logit model, allowing for the use of linear methods for handling fixed effects. We demonstrate the usefulness of our method through simulations and an empirical application to hospital choice. We show that our method also works well if the researcher wants to specify a model with individual-specific coefficients on an explanatory variable, even with large numbers of individuals.

l**Keywords:** MM algorithm, multinomial logit, fixed effects, jack-knife bias correction, acceleration methods

*The opinions expressed in this article are the authors' own and do not reflect the views of Charles River Associates. We thank Devesh Raval and Ted Rosenbaum for their feedback and for sharing the data used in the empirical application. We also thank Keith Brand and Daniel Grodzicki for their comments.

1 Introduction

Implementing models with many fixed effects is critical to modern empirical work and plays a central role in what Angrist and Pischke (2010) term the credibility revolution in economics. In linear models, methods for addressing large numbers of unobserved individual effects, such as demeaning, are well-known. However, these approaches do not directly extend to non-linear models, such as the multinomial logit. Instead, standard implementations of multinomial estimation typically rely on estimating dummy variable coefficients, which becomes computationally slow and memory intensive, or even infeasible, as the number of fixed effects grows large. While there are a number of methods for applying fixed effects approaches to binary outcome models, extending these methods to multinomial models has remained a significant challenge.

In this paper, we provide a new method for calculating maximum likelihood estimates for a multinomial logit model with fixed effects. Our approach is based on the Minorization-Maximization (MM) algorithm, which can be seen as a generalization of the Expectation-Maximization (EM) algorithm. The MM algorithm, originally developed in the statistics literature (Hunter and Lange, 2004; Lange, 2016), has seen little application in econometrics to date. We utilize the MM algorithm to minorize the logit model in a way that leads to a linear estimation problem, allowing us to apply standard techniques to handle the fixed effects. Our method is simple to program, achieves numerically identical estimates to standard techniques, and is significantly faster and more memory-efficient than previously available alternatives. Furthermore, we demonstrate that our approach remains effective even when the model includes individual-specific coefficients on an explanatory variable, making it feasible for applications with large numbers of individuals. We provide a theoretical convergence result but focus on demonstrating performance in numerical simulations.

Because we consider a fixed effects estimation approach that treats the unobserved effects as parameters to be estimated, the resulting estimators are generally subject to the incidental parameters problem due to the high dimensionality of the estimated parameters, as is well known in the nonlinear panel data literature (Neyman and Scott, 1948). We address this issue using ex-post bias reduction via the jackknife method proposed by Dhaene and Jochmans (2015).

We demonstrate the computational advantages of our estimator over competing alternative methods with simulations. The first alternative is Newton's method. Standard im-

plementations of multinomial logit estimation, such as those available in Stata or R, rely on Newton's method or related gradient-based methods. These implementations handle fixed effects by introducing indicator variables for each group, the so-called dummy variables approach. However, these implementations are extremely slow or run into memory problems in our simulations. To advantage Newton's method, we implement our own algorithm that exploits the sparsity and structure of the Hessian, following Hospido (2012). We further enhance Hospido's algorithm with several computational techniques specifically exploiting the data structure of our simulations, as described below. The second alternative uses the optimization algorithm Adam, as recommended for the multinomial logit by Du, Kanodia, and Athey (2023). We explore how these methods scale with the number of fixed effects and the number of individual-specific coefficients on a continuous explanatory variable. In terms of coefficients, all methods return essentially the same coefficients. In terms of speed, our method outperforms competing ones when handling fixed effects, and is comparable or better than alternatives when handling large numbers of individual-specific coefficients. Not surprisingly, Newton's method is a strong competitor. While our method outperforms Newton's method in the case of fixed effects, Newton's method is slightly better on average in the case of individual-specific coefficients. However, we view the speed improvement as small relative to the level of bespoke programming required to achieve it. We consider two acceleration methods designed for MM and EM algorithms, and we find that they pair well with our algorithm. Generally applicable code for implementing our MM algorithm with and without acceleration methods is available on author websites.[1]

We conclude with an empirical application to hospital choice based on Raval, Rosenbaum, and Tenn (2017). Recent research emphasizes highly flexible choice models in order to accurately determine the relevant market for competition policy purposes, which can be implemented with large numbers of fixed effects. In part because of the estimation challenge of estimating so many fixed effects, Raval, Rosenbaum, and Tenn (2017) estimate separate multinomial logit models for each sub-population in the data with just an intercept for each hospital as the only explanatory variables. This approach means the researcher cannot hold any coefficients constant across the whole population, and cannot include any other explanatory variables, such as a continuous variable or multi-way fixed effects. In this context, we show our estimator accommodates population-wide estimation of all of their parameters, and can easily handle additional explanatory variables as well.

[1]See in particular: `imeeker.com/code`

Overall, our paper makes several contributions. We provide an attractive new approach to estimate the multinomial discrete choice models with fixed effects. Within the MM literature, we provide a minorization for the multinomial logit model and show how it can be used to efficiently address panel data estimation. Our approach could likely be extended to a number of linear-index likelihood models, allowing a wide range of non-linear models to handle many fixed effects.

Literature Review Our paper contributes to the literature on fixed effects estimation of nonlinear panel data models. Several other papers precede us in this regard. A classic contribution is Chamberlain's conditional logit model (Chamberlain, 1980). Typical implementations handle only the binary outcome case, and extending the model to multinomial outcomes creates significant combinatoric complexities. Furthermore, the model does not naturally deliver estimates for the fixed effects, which our approach does.

D'Haultfœuille and Iaria (2016) and Hospido (2012) introduce efficient methods for computing the dummy variables model and the latter provides a bias-corrected likelihood approach. D'Haultfœuille and Iaria (2016) rely on simulation (which introduces integration error) of the choice set to cheaply compute the Hessian of the objective function, while Hospido (2012) exploits the sparsity of the fixed effects. When calculating the Hessian for Newton's method, we adapt Hospido (2012) to our setting.

Stammann, Heiß, and McFadden (2016) is perhaps closest to us in that they advocate for iterative demeaning to obtain estimates, and then ex-post bias correction, in their case based on Hahn and Newey (2004). Several techniques rely on concentrating out fixed effects so the researcher can maximize over the remaining parameters, such as Hinz, Hudle, and Wanner (2019) and Stammann (2018). Our understanding is that all three of these approaches have been developed only for binary outcomes and do not easily extend to multinomial settings.

Another approach relies on differencing out fixed effects in a way that leads to estimation with moment inequalities, such as Ho and Pakes (2014) and Shum, Song, and Shi (2018). Our approach differs in that it generates point identification and delivers estimates of the fixed effects, and we view our approach as computationally less challenging than estimation with moment inequalities. It is possible to replace the multinomial logit model with the Poisson model, as in Guimaraes, Figueirdo, and Woodward (2003), which is attractive because the Poisson model more easily accommodates fixed effects. However, this technique requires groups of agents to have the same explanatory variables.

Our discussion focuses on estimation methods. For identification, Mugnier and Wang, 2024 study identification in a linear index model with a non-parametric link function. Similar to our paper, they study the cases of fixed effects, two-way fixed effects, and individual-specific slope coefficients, and so cover our applications.

The Minorization-Maximization algorithm, which is sometimes called the Majorization-Minimization algorithm but is the MM algorithm in either case, has a long history in statistics dating about to the time of the introduction of the EM algorithm. The EM algorithm can be regarded as a special case of the MM algorithm. In general, the MM algorithm expands the set of functions that can be used in the E-step of the EM algorithm, and has appeared under many names in different papers, often depending on what function was used. Böhning and Lindsay (1988) is an important early citation, while Hunter and Lange (2004) and Lange (2016) provides a helpful overview and history. The MM literature has focused on cross-sectional applications, and we believe we are the first paper to apply the MM algorithm to panel data settings. We are aware of only one paper in the econometrics literature the precedes us: James (2017) shows that the MM algorithm can be advantageous in the context of the mixed multinomial logit model of McFadden and Train (2000) but does not discuss fixed effects or provide an empirical application. We are aware of several papers making use of our algorithm. Sahai (2023) embeds our MM algorithm into an EM algorithm to allow for the estimation of many fixed effects in a mixed logit model. He applies his EM algorithm to estimate a model of school choice in India. Rysman, Wang, and Wozniak, 2025 applies our model to study a panel of consumers choosing between payment type (such as card, cash, or check). They estimate a multinomial logit model with more than 2 million fixed effects.

Our paper is closely related to that of Chen (2019). She proposes a new EM algorithm for nonlinear panel data models with interactive effects. She uses the EM algorithm to transform the model into a linear form (with generalized residuals), enabling the application of existing techniques for handling interactive fixed effects in the linear case. Additionally, she derives the bias formula, and with its known analytic form, one can perform ex-post bias correction. Notably, her implementation of the EM algorithm closely resembles the MM algorithm. However, she does not consider multinomial models. Building on her ideas, our model could be extended to accommodate interactive fixed effects in a multinomial logit model. However, caution is required, as the log-likelihood function would no longer be concave, and the MM algorithm would, at best, be expected to converge to a local maximum.

2 The MM Algorithm

The central idea in our paper is to use the MM algorithm to estimate a nonlinear discrete choice model in an iterative series of two steps. First, in each iteration, we construct a simpler concave surrogate function that minorizes the complicated log-likelihood function, such that the surrogate function is less than the log-likelihood function everywhere but at the current best guess of the parameters, where it is equal. Second, we maximize the surrogate function instead of the log-likelihood function. The ascent of the log-likelihood is guaranteed by the property of minorization. By alternating between these steps of minorization and maximization, the MM algorithm finds the parameters that maximize the original log-likelihood function.

Before diving into the details of our multinomial logit model, we first discuss the definition of the MM algorithm in general and provide conditions required for convergence. Relative to the standard mathematical definition of a minorizing function, we add an extra condition that makes the function suitable for use in a maximization problem. We refer to our minorization as a *transfer minorization.* Our name is based on the terminology of Lange, Hunter, and Yang (2000), which refers to the minorization as the *transfer function.* That is, we transfer optimization from the function of interest to the minorization of this function.

Definition 1. *Suppose $\mathcal{L}$ is a real-valued function on Ω that is twice differentiable and S is a real-valued function on $\Omega \times \Omega$, we say that S is a* transfer minorization *of $\mathcal{L}$ if:*

(a) $S(\boldsymbol{\theta}; \boldsymbol{\vartheta}) \leq \mathcal{L}(\boldsymbol{\theta})$ *for all* $\boldsymbol{\theta}$ *and* $\boldsymbol{\vartheta}$*;*

(b) $S(\boldsymbol{\vartheta}; \boldsymbol{\vartheta}) = \mathcal{L}(\boldsymbol{\vartheta})$ *for all* $\boldsymbol{\vartheta}$*;*

(c) $\nabla^2_{\theta} S(\boldsymbol{\theta}; \boldsymbol{\vartheta})$*—which denotes the second derivative of S with respect to $\boldsymbol{\theta}$—exists and is negative definite at* $\boldsymbol{\theta}$*.*

Analogously, S is a *transfer majorization* of $\mathcal{L}$ if $-S$ is a transfer minorization of $-\mathcal{L}$. The first two conditions of Definition 1 are from Leeuw and Lange (2009) and are standard for defining a minorization. The third condition ensures that the minorization is well-behaved around the focal point. Arguably, we could use a less strict condition, such as that the minorization has the same sign as $\mathcal{L}$ in some region around $\boldsymbol{\theta}$, but in practice, we are not aware of any implementations of the MM algorithm that do not satisfy the third condition. As shown in Leeuw and Lange (2009), an implication of Definition 1 is:

Corollary 1. *If S is a transfer minorization of $\mathcal{L}$, then for all $\boldsymbol{\theta} \in \Omega$:*

$$\nabla_\theta S(\boldsymbol{\theta}; \boldsymbol{\theta}) = \nabla \mathcal{L}(\boldsymbol{\theta}).$$

This follows because, at $\boldsymbol{\theta} = \boldsymbol{\vartheta}$, the gradients of S and $\mathcal{L}$ must coincide for S to be a minorization.

Intuitively, when faced with a likelihood function $\mathcal{L}$ that is difficult to maximize, we instead maximize another function S. Its maximizer should always be closer to a local optima than the current guess, and we choose this surrogate function S so that it is easier to maximize than $\mathcal{L}$. Figure 1 provides an example. In the figure, we seek to maximize $\mathcal{L}$ over the scalar parameter θ. Because we are considering a scalar, we write θ rather than $\boldsymbol{\theta}$. We start with a guess $\theta^{(k)}$. Rather than seek to optimize $\mathcal{L}$ directly, we construct a transfer minorization $S(\theta; \theta^{(k)})$. As a minorization, it is always below $\mathcal{L}$ and is equal to $\mathcal{L}$ at $S(\theta^{(k)}; \theta^{(k)})$. As a transfer minorization, S is well-behaved around $\theta^{(k)}$, i.e. it is differentiable and concave. It is optimized at the point $\theta^{(k+1)}$. At this point, we will construct a new transfer minorization $S(\theta; \theta^{(k+1)})$. Under suitable regularity conditions, iterative application of this process converges to a local maximum of $\mathcal{L}$.

The general MM algorithm starts from an initial guess $\boldsymbol{\theta}^{(0)}$. The MM algorithm iteratively applies the following two-step procedure. In the first step, *the minorization step*, the surrogate function is computed $S(\boldsymbol{\theta}; \boldsymbol{\theta}^{(k)})$ for the current iterate $\boldsymbol{\theta}^{(k)}$. In the second step, *the maximization step*, the next iterate $\boldsymbol{\theta}^{(k+1)}$ is value $\boldsymbol{\theta} \in R^p$ that maximize $S(\boldsymbol{\theta}; \boldsymbol{\theta}^{(k)})$. At each step, let $\boldsymbol{\theta}^{(k)} = \boldsymbol{\theta}^{(k+1)}$ and repeat the procedure until $\boldsymbol{\theta}^{(k)}$ converges.

Lange (2013, Chapter 8) establishes convergence of the MM algorithm.[2] Böhning (1992) discusses the special case of applying the MM algorithm to the multinomial logit model without unobserved effects. Thus, any function that satisfies Definition 1 for some function $\mathcal{L}$ can be used in the MM algorithm to find an optimum to $\mathcal{L}$. This approach allows substantial freedom in selecting the transfer minorization. From the perspective of the MM algorithm, the EM algorithm is a special case and it works because the conditional expectation of $\mathcal{L}$ used in the EM algorithm is a transfer minorization. The MM algorithm is more general because the researcher is free to use any minorizing function. Hunter and Lange (2004) discuss several approaches for constructing MM algorithms. In the next section, we focus on a Taylor expansion, which delivers a least-squares optimization

[2]Lange, 2013 does not include part (c) of Definition 1 in the definition of a minorization but then later imposes it for the proof of convergence.

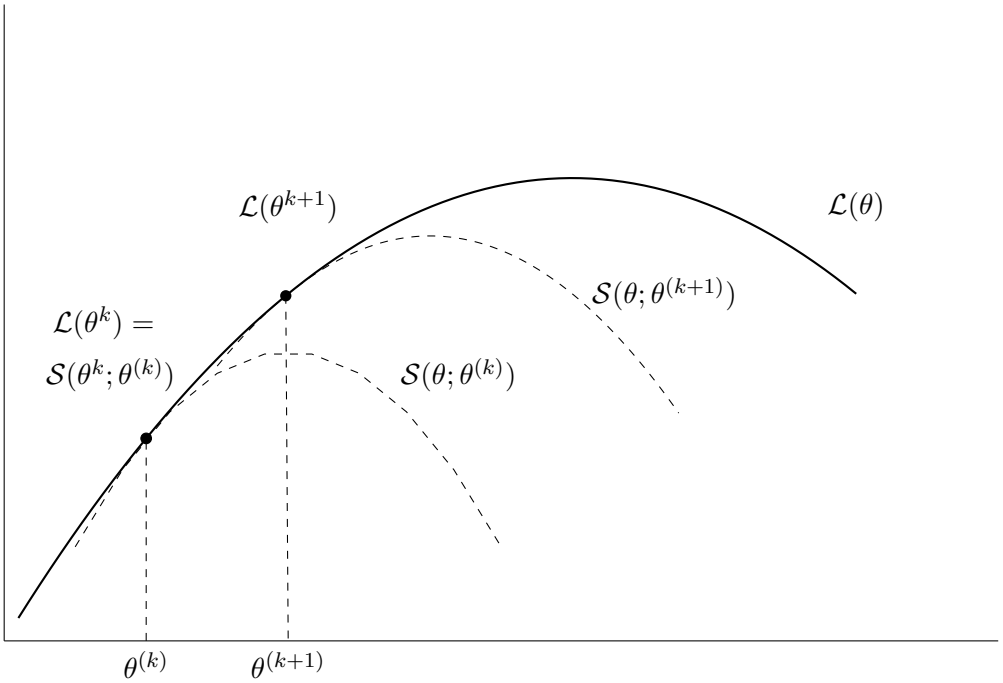


Figure 1: One-dimensional illustration of MM algorithm as a minorization or optimization transfer strategy.

problem in our context.

3 Model

In the multinomial logit model, an individual makes a discrete choice among several alternatives, each of which draws an Extreme Value error. The distinguishing feature of the model is the closed-form logistic function for the probability of each choice.

We observe I individuals make a discrete choice among J alternatives in each of T time periods.[3] Individual i in period t who chooses alternative j obtains utility u_{ijt}. Utility is defined as:

$$u_{ijt} = x_{it}\beta_j + w_{jt}\gamma_i + \alpha_{ij} + \varepsilon_{ijt}, \tag{1}$$

where x_{it} is a set of individual-specific observable characteristics; w_{jt} is a set of alternative-specific characteristics; and α_{ij} is a set of fixed effects at the household-by-alternative

[3]The number of alternatives can differ across time and consumers. For ease of presentation, we consider a balanced panel.

level. For instance, if we observe households choosing the method of transportation to work, x_{it} might represent household income, w_{jt} might represent the cost of various transportation modes, and α_{ij} would be separate fixed effects for every household and alternative combination. We typically normalize one alternative to have a utility of zero, so α_{ij} represents $I \times (J-1)$ fixed effects. It is possible to allow w_{jt} to vary across individuals (i.e., w_{ijt}), for instance, if prices were specific to the household.

Continuing with Equation 1, ε_{ijt} is a scalar $i.i.d$ idiosyncratic shock distributed according to a Type I Extreme Value distribution. The variable y_{ijt} is a binary indicator for the alternative that consumer i chooses in t, where:

$$y_{ijt} = \mathbb{1}[u_{ijt} \geq u_{ikt}, \forall k], \ \forall i, t.$$

We collect the parameter vectors as $\boldsymbol{\theta} = \{\beta_j, \gamma_i, \alpha_{ij}, i = 1, \ldots, I, j = 1, \ldots, J\}$. These are to be estimated. The log-likelihood function is then:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i,t} \log \ l(x_{it}\boldsymbol{\beta} + \boldsymbol{w_t}\gamma_i + \boldsymbol{\alpha_i}; \boldsymbol{y_{it}}) \tag{2}$$

where $\boldsymbol{\beta} = \{\beta_1, \ldots, \beta_J\}$, $\boldsymbol{w}_t = \{w_{1t}, \ldots, w_{Jt}\}$, $\boldsymbol{\alpha}_i = \{\alpha_{i1}, \ldots, \alpha_{iJ}\}$, $\boldsymbol{y}_{it} = \{y_{i1t}, \ldots, y_{iJt}\}$ and:

$$\begin{aligned} l(x_{it}\boldsymbol{\beta} + \boldsymbol{w}_t\gamma_i + \boldsymbol{\alpha}_i; \boldsymbol{y}_{it}) &= \textstyle\prod_{j=1}^{J} p_{ijt}(x_{it}\boldsymbol{\beta} + \boldsymbol{w}_t\gamma_i + \boldsymbol{\alpha}_i)^{y_{ijt}}, \\ p_{ijt}(x_{it}\boldsymbol{\beta} + \boldsymbol{w}_t\gamma_i + \boldsymbol{\alpha}_i) &= \frac{\exp(x_{it}\beta_j + w_{jt}\gamma_i + \alpha_{ij})}{\sum_{k=1}^{J} \exp(x_{it}\beta_k + w_{kt}\gamma_i + \alpha_{ik})}. \end{aligned}$$

In practice, we normalize the mean utility of one of the alternatives to be zero, as is standard in the multinomial logit.

4 An MM Algorithm for the Multinomial Logit

4.1 The Transfer Minorization

This subsection derives a function S that satisfies the conditions of Definition 1 to be a transfer minorization of $\mathcal{L}$.

Theorem 1. *Let $\mathcal{L}(\boldsymbol{\theta})$ be the log-likelihood function for a multinomial logit model as defined in Eq.(2). Let $S(\boldsymbol{\theta};\boldsymbol{\theta}^{(k)})$ be defined as:*

$$S(\boldsymbol{\theta};\boldsymbol{\theta}^{(k)}) = \mathcal{L}(\boldsymbol{\theta}^{(k)}) + \frac{1}{2}\sum_{i,j,t} h_j(\boldsymbol{\psi}_{it}^{(k)};\boldsymbol{y}_{it})^2 - \frac{1}{2}\sum_{i,j,t}\left(x_{it}\beta_j^{(k)} + w_{jt}\gamma_i^{(k)} + \alpha_{ij}^{(k)} + h_j(\boldsymbol{\psi}_{it}^{(k)};\boldsymbol{y}_{it}) - x_{it}\beta_j - w_{jt}\gamma_i - \alpha_{ij}\right)^2, \quad (3)$$

where

$$\begin{aligned} h_j(\boldsymbol{\psi}_{it}^{(k)};\boldsymbol{y}_{it}) &= \frac{\partial \log l(\boldsymbol{\psi}_{it}^{(k)};\boldsymbol{y}_{it})}{\partial \psi_{ijt}^{(k)}} = y_{ijt} - p_{ijt}(\boldsymbol{\psi}_{it}^{(k)}), \\ p_{ijt}(\boldsymbol{\psi}_{it}) &= \frac{\exp(\psi_{ijt})}{\sum_{k=1}^{J}\exp(\psi_{ikt})}, \\ \psi_{ijt}^{(k)} &= x_{it}\beta_j^{(k)} + w_{jt}\gamma_i^{(k)} + \alpha_{ij}^{(k)}, \\ \boldsymbol{\psi}_{it}^{(k)} &= \{\psi_{i1t},\ldots,\psi_{iJt}\}, \end{aligned}$$

then $S(\boldsymbol{\theta};\boldsymbol{\theta}^{(k)})$ is a transfer minorization of $\mathcal{L}(\boldsymbol{\theta})$.

Therefore $\boldsymbol{\theta}^{(k+1)} = \arg\max_{\boldsymbol{\theta}} S(\boldsymbol{\theta};\boldsymbol{\theta}^{(k)}), k = 0, 1, 2, ...$ converges to a local, if not global, maximum of $\mathcal{L}(\boldsymbol{\theta})$.

Eq.(3) is a first-order Taylor expansion of the likelihood function. Note that the parameters that we search over in the iterative MM algorithm, $\boldsymbol{\theta}$, appear only in the third part of the right-hand side of Eq.(3). Focusing on this third part, we can view the optimization of $S(\boldsymbol{\theta},\boldsymbol{\theta}^{(k)})$ as a linear regression of $v_{ijt}^{(k)} = x_{it}\beta_j^{(k)} + w_{jt}\gamma_i^{(k)} + \alpha_{ij}^{(k)} + h_j(\boldsymbol{\psi}_{it}^{(k)};y_{it})$ on x_{it}, w_{jt}, and household-alternative fixed effects. This is the central benefit of our MM approach to the multinomial logit. We convert a non-linear optimization problem to a series of sequential linear optimization, which is particularly attractive when there are many fixed effects.

The functional form of $h_j(\boldsymbol{\psi}_{it}^{(k)};\boldsymbol{y}_{it})$ is clearly important to our technique. For the multinomial logit, this function takes on a particularly simple form: $y_{ijt} - p_{ijt}(\boldsymbol{\psi}_{it}^{(k)})$. Thinking of $x_{it}\beta_j^{(k)} + w_{jt}\gamma_i^{(k)} + \alpha_{ij}^{(k)}$ as the expectation of u_{ijt} at iteration k, our iterative linear regression uses a dependent variable above the expectation for observations with $y_{ijt} = 1$ and below the expectation for observations with $y_{ijt} = 0$.

If we could observe u_{ijt}, we could estimate our coefficients and fixed effects directly by linear techniques rather than relying on discrete outcome methods such as the multinomial logit. In this sense, the minorization step of the MM algorithm has the feel of

data augmentation (Tanner, 1996). That is, we calculate v_{ijt} as an approximation of the unobserved u_{ijt}. That is the intuition behind many applications of the EM algorithm.

In the multinomial logit, v_{ijt} does not coincide with the expectation of u_{ijt}. That is, $v_{ijt}^{(k)} \neq E[u_{ijt}|x_{it}, w_{jt}, \alpha_{ij}, y_{ijt}, \boldsymbol{\theta}_j^{(k)}]$. Indeed, substituting $v_{ijt}^{(k)}$ with $E[u_{ijt}|x_{it}, w_{jt}, \alpha_{ij}, y_{ijt}, \boldsymbol{\theta}_j^{(k)}]$ would lead to biased results because the likelihood of the expectation is not equal to the expectation of the likelihood. Generating the expectation of the likelihood function for the case of the multinomial logit is more complicated than our minorization approach, so the EM algorithm is relatively unattractive in our context.

Remark 1 (A Digression on the Binary Probit)**.** *In the case of the binary probit, the MM and EM algorithms coincide. Chen (2019) estimates the binary probit by the EM algorithm and derives a functional form equivalent to our MM algorithm. Ruud, 1991 and Greene (2018) also discuss applying the EM to the binary probit model. In the case of binary probit:*

$$l(x_{it}\beta + w_t\gamma_i + \alpha_i; y_{it}) = \begin{cases} \Phi(x_{it}\beta + w_t\gamma_i + \alpha_i), & y_{it} = 1 \\ 1 - \Phi(x_{it}\beta + w_t\gamma_i + \alpha_i), & y_{it} = 0 \end{cases}$$

where $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution, $\phi(\cdot)$ is its probability density function. We have suppressed j subscripts relative to the previous equations as there are only two alternatives.

For ease of notation, let $\psi_{it} = x_{it}\beta + w_t\gamma_i + \alpha_i$. Accordingly:

$$\begin{aligned} h(\psi_{it}; y_{it}) &= \frac{\partial \log(\psi_{it}; y_{it})}{\partial(\psi_{it})} = \begin{cases} \dfrac{\phi(\psi_{it})}{\Phi(\psi_{it})}, & y_{it} = 1 \\ \dfrac{-\phi(\psi_{it})}{1 - \Phi(\psi_{it})}, & y_{it} = 0 \end{cases} \\ &= \frac{(y_{it} - \Phi(\psi_{it}))\,\phi(\psi_{it})}{\Phi(\psi_{it})\,(1 - \Phi(\psi_{it}))} \end{aligned}$$

Thus, we see that $h(\psi_{it}; y_{it})$ corresponds to the well-known Mills ratio and, as a result, $h(\psi_{it}; y_{it}) = E[\epsilon_{it}|\psi_{it}, y_{it}]$, where ϵ_{it} corresponds to the normally distributed error from the probit model. In this sense, the MM and EM algorithms correspond in the case of the binary probit.

4.2 The MM Algorithm

Our MM algorithm starts from an initial guess $\boldsymbol{\theta}^{(0)}$. From this initial guess, each iteration follows a two-step procedure. In the minorization step, we compute $v_{ijt} = x_{it}\beta_j^{(k)} + w_{jt}\gamma_i^{(k)} + \alpha_{ij}^{(k)} + h_j(\boldsymbol{\psi}_{it}^{(k)}; \boldsymbol{y}_{it})$. In the maximization step, we update the parameters by linearly regressing v_{ijt} on x_{it}, w_{jt}, and household-choice fixed effects. These steps repeat until convergence.

For our base algorithm, we use the dummy variables approach to handle fixed effects. Let d_{ij} be an $I \times (J-1)$ vector of dummy variables indicating observation ij. We combine the regressors x_{it}, w_{jt}, and d_{ij} into a single vector $z_{ijt} = [x_{it}, w_{jt}, d_{ij}]$ and then stack the observations to form the matrix $\boldsymbol{Z}$. We write out the steps in Algorithm 1.

Algorithm 1 (Base Algorithm)

Require: initial guess $\boldsymbol{\theta}^{(0)} = \{\beta_j^{(0)}, \gamma_i^{(0)}, \alpha_{ij}^{(0)}\}_{i=1,\ldots,I\ j=1,\ldots,J}$

repeat

$k \leftarrow k+1$

Minorization Step:

$v_{ijt}^{(k)} \leftarrow x_{it}\beta_j^{(k)} + w_{jt}\gamma_i^{(k)} + \alpha_{ij}^{(k)} + h_j(\boldsymbol{\psi}_{it}^{(k)}; \boldsymbol{y}_{it})$

$\boldsymbol{v}^{(k)} \leftarrow \{v_{ijt}^{(k)}\}_{i=1,\ldots I;\ j=1,\ldots,J;\ t=1,\ldots,T}$

Maximization Step: (Update θ^k by OLS)

$\boldsymbol{\theta}^{(k+1)} \leftarrow (\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{Z}\boldsymbol{v}^{(k)}$

until convergence

There are several ways to improve the efficiency of the algorithm at the cost of clarity. First note that the independent variables in the linear regressions do not change from iteration to iteration. As a result, we need to compute $(\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{Z}$ only once at the beginning, reducing the computational burden of the maximization step. Also note that the logit probabilities depend on $x_{it}\beta_j^{(k)} + w_{jt}\gamma_i^{(k)} + \alpha_{ij}^{(k)}$, so we need to compute this term only once per minorization step.

In the algorithm, we did not specify a convergence criterion. The MM algorithm accommodates a number of different convergence criteria. The researcher should use the criterion that makes sense with their application. A natural choice is the difference in the successive values of the parameters $\left\|\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^{(k)}\right\|$. Another possible choice is the difference in successive values of the log-likelihood $\left|\mathcal{L}\left(\boldsymbol{\theta}^{(k+1)}\right) - \mathcal{L}\left(\boldsymbol{\theta}^{(k)}\right)\right|$. This criterion requires an additional computational step since the MM algorithm does not compute the

likelihood.

4.3 Handling Fixed Effects

If some of the fixed effects have many categories, updating the parameters using OLS with dummy variables is inefficient or even infeasible. As the model is linear at each iteration, we make use of the Frisch-Waugh-Lovell theorem to absorb the fixed effects. Partialling out the fixed effects greatly reduces the dimensionality of the problem and hence the computational cost of the maximization step.

In the case of one-way fixed effects, we can residualize the regressors by demeaning the variables. In the maximization step, we first recover a subset of the parameters $\boldsymbol{\eta} = \{\{\beta_j\}_{j=1,\ldots,J}, \{\gamma_i\}_{i=1,\ldots,I}\}$ using demeaned OLS. The asymptotic runtime of OLS helps us understand why demeaned OLS can lead to large speed improvements. OLS has a complexity of $\mathcal{O}(K^2(N+K))$ where N is the number of observations and K is the number of regressors (i.e., the dimensions of x_{it} and w_{jt}).[4] Because runtime is cubic in the number of regressors, absorbing the fixed effects has a major impact on the speed of the algorithm relative to the dummy variables approach, which introduces $I \times (J-1)$ regressors. This suggests that our method will be most effective when the number of additional covariates is small and the number of fixed effects is large.

After recovering $\boldsymbol{\eta}$, we then recover the fixed effects in a separate, additional step. The computational cost of this additional step is small relative to the gain from using partitioned regression.

For ease of notation, we combine the variables x_{jt} and w_{it} into a single matrix and stack observations to form the matrix $\boldsymbol{\mathcal{Z}}$. We denote the demeaned variables with a tilde, such as $\tilde{\boldsymbol{\mathcal{Z}}}$. We present the steps for the case of demeaning in Algorithm 2.

The update for the fixed effects returns a vector with a length equal to the number of observations N. This vector repeats α_{ij} for each time period. The advantage of this update step is that it extends naturally to multi-way fixed effects. We can also save the values $\boldsymbol{\mathcal{Z}}\boldsymbol{\eta}^{(k+1)}$ and $\tilde{\boldsymbol{\mathcal{Z}}}\boldsymbol{\eta}^{(k+1)}$ for use on the next iteration. An alternative way to update

[4] Our Big-O calculation assumes schoolbook matrix multiplication for all multiplications and Gauss-Jordan elimination for computing the inverse.

Algorithm 2 (Fixed Effects)

Require: initial guess $\boldsymbol{\theta}^{(0)} = \{\beta_j^{(0)}, \gamma_i^{(0)}, \alpha_{ij}^{(0)}\}_{i=1,\ldots,I\ j=1,\ldots,J}$

repeat

$k \leftarrow k + 1$

Minorization Step:

$v_{ijt}^{(k)} \leftarrow x_{it}\beta_j^{(k)} + w_{jt}\gamma_i^{(k)} + \alpha_{ij}^{(k)} + h_j(\boldsymbol{\psi}_{it}^{(k)}; y_{it})$

$\boldsymbol{v}^{(k)} \leftarrow \{v_{ijt}^{(k)}\}_{i=1,\ldots N;\ j=1,\ldots,J;\ t=1,\ldots,T}$

$\tilde{\boldsymbol{v}}^{(\boldsymbol{k})} \leftarrow$ Demean $\boldsymbol{v}^{(k)}$

Maximization Step:

1. Update $\eta^{(k)}$ by demeaned OLS:

   $\boldsymbol{\eta}^{(k+1)} \leftarrow \left(\tilde{\boldsymbol{\mathcal{Z}}}'\tilde{\boldsymbol{\mathcal{Z}}}\right)^{-1}\tilde{\boldsymbol{\mathcal{Z}}}\tilde{\boldsymbol{v}}^{(\boldsymbol{k})}$

2. Update the fixed effects: $\boldsymbol{\alpha}$

   $\boldsymbol{\alpha}^{(k+1)} \leftarrow \boldsymbol{v}^{(k)} - \boldsymbol{\mathcal{Z}}\boldsymbol{\eta}^{(k+1)} - \left(\tilde{\boldsymbol{v}}^{(k)} - \tilde{\boldsymbol{\mathcal{Z}}}\boldsymbol{\eta}^{(k+1)}\right)$

until convergence

the fixed effects is to average the residuals:

$$\alpha_{ij}^{(k+1)} = \frac{1}{T}\sum_t v_{ijt}^{(k)} - z_{ijt}\boldsymbol{\eta}^{(k+1)} \tag{4}$$

This update does not repeat the values of α_{ij} since we are averaging across time periods. Both updates yield equivalent values for α_{ij}.

Note that the independent variables $\boldsymbol{\mathcal{Z}}$ do not change from iteration to iteration. To avoid repetition, we can then pull out the term $\left(\tilde{\boldsymbol{\mathcal{Z}}}'\tilde{\boldsymbol{\mathcal{Z}}}\right)^{-1}\tilde{\boldsymbol{\mathcal{Z}}}$ and save it for later iterations.

The above algorithm also applies to the case of multi-way fixed effects with minor modification. The algorithm can accommodate both individual-alternative fixed effects and alternative-time fixed effects. For example, utility can have the form:

$$u_{ijt} = x_{it}\beta_j + w_{jt}\gamma_i + \alpha_{ij} + \tau_{jt} + \varepsilon_{ijt}, \tag{5}$$

where τ_{jt} are the alternative-time fixed effects. The other variables are the same as in (1).

The main difference with multi-way fixed effects is the method for residualizing the regressors. With multi-way fixed effects, the annihilator matrix for projecting out the fixed

effects is not equivalent to demeaning and beyond two fixed effects, directly computing the annihilator matrix is normally impossible. A number of papers provide methods for absorbing multi-way fixed effects in linear models. Notably, Guimaraes and Portugal (2010), Gaure (2013), and Correia (2017) provide iterative estimators based on the method of alternating projections. Fong and Saunders (2011) provide an alternative estimator for solving sparse linear systems more generally. Finally, Somaini and Wolak (2016) provide a method for computing the annihilator matrix in the case of two-way fixed effects. These methods are available in many commonly-used statistical software.[5] Researchers can use any of these methods to residualize the regressors in the maximization step.

In the case of multi-way fixed effects, our algorithm recovers the sum of the fixed effects $\alpha_{ij}+\tau_{jt}$ rather than the actual coefficients. So in the maximization step, we are updating the sum as:

$$[\alpha_{ij} + \tau_{jt}]^{(k+1)} = v_{ijt}^{(k)} - z_{ijt}\hat{\eta}^{(k+1)} - \left(\tilde{v}_{ijt}^{(k)} - \tilde{z}_{ijt}\hat{\eta}^{(k+1)}\right) \tag{6}$$

Recovering the coefficients requires solving the system defined by:

$$\alpha_{ij} + \tau_{jt} = v_{ijt} - z_{ijt}\hat{\eta} - (\tilde{v}_{ijt} - \tilde{z}_{ijt}\hat{\eta}) \tag{7}$$

where $\hat{\eta}$ is the final estimates. With high-dimensional fixed effects, this system will be too large to solve directly and must be solved using numerical routines. Gaure (2013) and Stammann (2018) describes how to solve this particular system using the Kaczmarz method (Kaczmarz, 1993).[6] Solving for the coefficients ex-post will reduce some of the computational gains from our MM algorithm.

5 Further refinements

5.1 Handling Individual-Specific Coefficients

The dimension of the individual-specific coefficients γ_i can also pose computational problems. Updating these parameters using OLS with interactions between regressors and dummies is inefficient and at worst infeasible. Just as we did with the fixed effects,

[5] All of the methods mentioned are available in the python package PyHDFE. Gaure (2013) is also available in the R packages lfe and collapse, and Correia (2017) is available in the Stata package reghdfe.

[6] This is the English translation of Kaczmarz's original 1937 paper in German.

we can solve this problem by partialling out the alternative-specific variables to avoid directly computing the individual-specific coefficients. Reducing the dimensionality of the problem offers significant speed advantages.

To partial out the alternative-specific variables, we need to use a generalized demeaning algorithm that can handle heterogeneous slopes, such as Correia (2017). Projecting out the individual-specific variables in addition to the fixed effects requires only a slight modification to algorithm 2. As with the fixed effects, demeaning affects the maximization step, not the minorization step. In the maximization step, we first recover the parameters $\boldsymbol{\beta} = (\{\beta_j\}_{j=1,\ldots,J})$ using demeaned OLS. After recovering $\boldsymbol{\beta}$, we then recover the contribution of alternative-specific variables and the fixed effects in an additional step. The estimation procedure can be found in Algorithm 3.

Algorithm 3 (Fixed Effects and Individual-Specific Coefficients)

Require: initial guess $\boldsymbol{\theta}^{(0)} = \{\beta_j^{(0)}, \gamma_i^{(0)}, \alpha_{ij}^{(0)}\}_{i=1,\ldots,I\ j=1,\ldots,J}$

repeat

$k \leftarrow k + 1$

Minorization Step:

$v_{ijt}^{(k)} \leftarrow x_{it}\beta_j^{(k)} + w_{jt}\gamma_i^{(k)} + \alpha_{ij}^{(k)} + h_j(\boldsymbol{\psi}_{it}^{(k)}; \boldsymbol{y}_{it})$

$\boldsymbol{v}^{(k)} \leftarrow \{v_{ijt}^{(k)}\}_{i=1,\ldots N;\ j=1,\ldots,J;\ t=1,\ldots,T}$

$\tilde{\boldsymbol{v}}^{(\boldsymbol{k})} \leftarrow$ Demean $\boldsymbol{v}^{(k)}$

Maximization Step:

1. Update $\beta^{(k)}$ by demeaned OLS:

$$\boldsymbol{\beta}^{(k+1)} \leftarrow \left(\tilde{\boldsymbol{X}}'\tilde{\boldsymbol{X}}\right)^{-1} \tilde{\boldsymbol{X}}\tilde{\boldsymbol{v}}^{(\boldsymbol{k})}$$

2. Update the contribution $\boldsymbol{w\gamma} + \boldsymbol{\alpha}$:

$$\boldsymbol{w\gamma}^{(k+1)} + \boldsymbol{\alpha}^{(k+1)} \leftarrow \boldsymbol{v}^{(k)} - \boldsymbol{X\beta}^{(k+1)} - \left(\tilde{\boldsymbol{v}}^{(k)} - \tilde{\boldsymbol{X}}\boldsymbol{\beta}^{(k+1)}\right)$$

until convergence

As with multi-way fixed effects, our algorithm recovers the overall contribution of alternative-specific variables and the fixed effects rather than the coefficients. Finding the coefficients requires solving a large sparse system numerically, which can be computationally expensive. This step is necessary only if the researcher cares about the specific value of the coefficients.

5.2 Acceleration Methods

The low cost per iteration of the MM algorithm comes at the expense of more iterations. Many MM algorithms can exhibit slow convergence close to the maximum (Zhou, Alexander, and Lange, 2011). In these cases, acceleration methods can be used to improve the rate of convergence.

Acceleration methods use additional information to improve the parameter updates. There are many acceleration methods for MM algorithms. We highlight two iterative methods well-suited to our algorithm, SQUAREM (Varadhan and Roland, 2008) and Zhou, Alexander, and Lange (2011) (hereafter ZAL). Both methods are applicable to algorithms 1, 2, and 3.

SQUAREM and ZAL approach the MM algorithm from the perspective of a fixed point equation.[7] We can view the MM algorithm as a mapping F from the current iterate $\boldsymbol{\theta}^{(k)}$ to the next iterate $\boldsymbol{\theta}^{(k+1)}$, $\boldsymbol{\theta}^{(k+1)} = F(\boldsymbol{\theta}^{(k)})$. In this formulation, the goal of the MM algorithm is to find the fixed point $\theta^* = F(\theta^*)$ or equivalently to find the root of the function $G(\boldsymbol{\theta}) = \theta - F(\boldsymbol{\theta})$. Finding the root of $G(\boldsymbol{\theta})$ can be done with Newton's method which updates the parameters according to:

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \nabla G(\boldsymbol{\theta}^{(k)})^{-1} G(\boldsymbol{\theta}^{k}) \tag{8}$$

Rather than compute $\nabla G(\boldsymbol{\theta}^{(k)})^{-1}$ directly, both methods rely on an approximation. The two methods use different approximations, resulting in different updating formulas.

Before giving the update formula, we first define the quantities $u_k = F(\boldsymbol{\theta}^{(k)}) - \boldsymbol{\theta}^{(k)}$, and $w_k = F \circ F(\boldsymbol{\theta}^{(k)}) - F(\boldsymbol{\theta}^{(k)})$. These quantities are differences in the parameter values between successive steps of the MM algorithm. With these defined, the SQUAREM update formula is:

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - 2s_k u_k + s_k^2 (w_k - u_k) \tag{9}$$

where $s_k = -\frac{||u_k||}{||w_k - u_k||}$.[8] The update formula for ZAL is:

$$\boldsymbol{\theta}^{(k+1)} = (1 - c_k) F(\boldsymbol{\theta}^{(k)}) + c_k F \circ F(\boldsymbol{\theta}^{(k)}) \tag{10}$$

[7] The discussion of these algorithms is close to that in Zhou, Alexander, and Lange (2011).

[8] There are three variants of SQUAREM that calculate the step size s_k in different ways (Varadhan and Roland, 2008) We focus on the third variant, which generally performs the best (Du and Varadhan, 2020).

where $c_k = -\frac{u_k^T u_k}{u_k^T (w_k - u_k)}$. As the update formulas show, a single step of each acceleration method really consists of three steps. The method first performs two steps of the MM algorithm, saving the parameter values at each step. Rather than taking another step of the MM algorithm, these methods extrapolate the next values from the two MM iterations. By using information from two MM steps, these methods can produce larger steps or better search directions.

While these acceleration methods can result in noticeable speed improvements over the base algorithm, they break the ascent property of the MM algorithm. As a result, these methods will not necessarily give parameter values that increase the likelihood at every iteration. In these cases, the researcher can fall back to the second MM update $F \circ F(\boldsymbol{\theta}^{(k)}))$ to ensure reliability. Note that SQUAREM and ZAL are designed for EM and MM-type algorithms that exhibit monotone convergence and are not applicable to Adam or Newton-type methods.

Remark 2. *Estimates from the MM algorithm for nonlinear panel data models may still be biased due to the incidental parameters problem. To reduce the bias, we recommend that researchers use split-panel jackknife estimation following Dhaene and Jochmans (2015). The split-panel jackknife removes the leading bias and reduces the bias from high-order terms. Dhaene and Jochmans (2015) show that the split-panel jackknife minimizes the higher-order bias.*

Split-panel jackknife estimation involves splitting the dataset into two subpanels. In the simplest case, the first subpanel runs from time $t = 0$ *to* $t = \frac{T}{2}$ *and the second runs from* $t = \frac{T}{2} + 1$ *to* $t = T$*. We use the MM algorithm to obtain an estimate for the full sample* $\hat{\boldsymbol{\theta}}$*, an estimate for the first subpanel* $\hat{\boldsymbol{\theta}}_1$*, and an estimate for the second subpanel* $\hat{\boldsymbol{\theta}}_2$*. The split-panel jackknife estimator* $\tilde{\boldsymbol{\theta}}$ *is then* $\tilde{\boldsymbol{\theta}} = 2\hat{\boldsymbol{\theta}} - \frac{1}{2}\left(\hat{\boldsymbol{\theta}}_1 + \hat{\boldsymbol{\theta}}_2\right)$.

6 Alternative Estimation Methods

6.1 Line Search Methods

In later simulations, we assess the performance of the MM algorithm against commonly used first and second-order line search methods. Specifically, we consider Adam (Kingma

and Ba, 2014) and Newton's method.[9] These particular methods are available in existing software implementations of logit models. The R package mlogit and the Stata commands cmlogit use Newton's method. The recent Python package torch-choice described in Du, Kanodia, and Athey, 2023 includes a variety of first-order methods, including Adam, which the paper recommends for large-scale problems.

Line search methods use an iterative procedure to find the maximum of the log-likelihood. Starting from the current iterate $\boldsymbol{\theta}^{(k)}$, these methods determine a search direction ρ_k and the amount to move along that search direction σ_k (Nocedal and Wright, 2006). Line search methods update the parameter values according to:

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \sigma_k \rho_k \tag{11}$$

Line search methods differ in how they calculate the search direction ρ_k and the step size σ_k.

6.2 First-Order Methods

First-order methods use information about the gradient to find the search direction and adjust the step size. In the case of the multinomial logit, the gradient is easy to compute, so these methods have a low cost per iteration and hence scale well with the number of fixed effects.

Standard gradient ascent searches in the direction of the gradient $\rho = \nabla \mathcal{L}(\boldsymbol{\theta}^{(k)})$ and uses a constant step size $\sigma_k = \sigma$. Standard gradient ascent can be sensitive to the step size. If the step size is too large, the method can overshoot the maximum. Too small a step size, the method can be slow to converge. Adaptive methods, like Adam, adjust the step size across iterations and across different dimensions of the parameters. Adam takes an exponentially weighted average over gradients and the squared gradients. It updates the parameters as:

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \sigma \frac{\boldsymbol{m}^{(k)}}{\sqrt{\boldsymbol{v}^{(k)} + \epsilon}} \tag{12}$$

where $\boldsymbol{m}^{(k)}$ is the weighted average of the gradients $\boldsymbol{m}^{(k)} = \phi_1 \boldsymbol{m}^{(k-1)} + (1-\phi_1)\nabla \mathcal{L}(\boldsymbol{\theta}^{(k)})$; $\boldsymbol{v}^{(k)}$ is the weighted average of the squared gradients $\boldsymbol{v}^{(k)} = \phi_2 \boldsymbol{v}^{(k-1)} + (1-\phi_2)\nabla \mathcal{L}(\boldsymbol{\theta}^{(k)})^2$;

[9] Here, we are referring to Newton's method for optimization, which is distinct from Newton's method for root finding mentioned in section 5.4.

σ is the step size; ϕ_1 and ϕ_2 govern the exponential decay; and ϵ is a hyperparameter to avoid division by zero.

The choices of the hyperparameters σ, ϕ_1, ϕ_2, and ϵ affect the performance of Adam. While the default values recommended by Kingma and Ba (2014) often perform well in practice, they are not appropriate for every problem. Adam can fail to converge due to a poor choice of hyperparameters (Reddi, Kale, and Kumar, 2019; Zhang et al., 2022). Appendix B describes the choices for the hyperparameters for Adam that we use in our simulations. From the perspective of researcher time, the time spent tuning the hyperparameters for optimal performance can negate any speed advantage. In contrast to first-order methods, the MM algorithm does not depend on any hyperparameters. The ascent property of the MM algorithm guarantees that each iteration moves closer to the maximum and gives our algorithm good numerical stability.

6.3 Second-Order Methods

Second-order methods use information on the first and second-order derivatives to improve the search direction. Newton's method, for example, computes the search direction as:

$$\rho^{(k)} = \left(\nabla^2 \mathcal{L}(\boldsymbol{\theta}^{(k)})\right)^{-1} \nabla \mathcal{L}(\boldsymbol{\theta}^{(k)}) \tag{13}$$

When the number of parameters is large, brute-force calculation and inversion the Hessian is costly. The cost of inverting the Hessian scales cubically with the number of parameters.[10] With fixed effects, the Hessian has a sparse structure that facilitates direct calculation of the Newton (Hospido, 2012). The form of the multinomial logit imposes additional structure on the Hessian that facilitates direct computation.

As we will show later, even exploiting the structure of the Hessian, each iteration of Newton's method is considerably more expensive than an iteration of the MM algorithm. Quasi-Newton methods, like BFGS, replace the inverse Hessian in (13) with an approximation. By avoiding the inversion, BFGS is normally less computationally burdensome than Newton's method. However, due to the structure of the Hessian, we found the BFGS update to be more expensive than directly calculating the Newton step, so we study Newton's method rather than quasi-Newton methods for the remainder of the

[10] Calculating the inversion through Gauss-Jordan elimination has a complexity of $\mathcal{O}(P^3)$ where P is the number of parameters.

paper.

Second-order methods often come with substantial memory overhead. In particular, when the number of fixed effects is large, the inverse Hessian or an approximation to it can occupy a great deal of memory. Unlike the Hessian, the inverse Hessian is not sparse because the block containing the fixed effects is dense. For Newton's method, clever use of block matrix multiplication can mitigate memory issues at the cost of increased code complexity.

Although second-order methods have a high cost per iteration, they exhibit faster convergence. Newton's method converges at a quadratic rate, whereas MM algorithms typically converge at a linear rate. The MM algorithm therefore takes more iterations, but each iteration is less costly. We investigate this tradeoff in the next section.

While Newton's method is commonly used with multinomial logit in standard statistical software such as Stata and R, these implementations are very general and do not utilize the structure of the Hessian in the presence of fixed effects. Moreover, these methods tend to store the entire Hessian and invert it like any other matrix. The end result is long runtimes and incredibly high memory usage. Existing implementations for estimating multinomial models that rely on Newton's method are not well-suited to large-scale problems.

In order to better advantage Newton's method, we exploit the sparsity in the Hessian resulting from fixed-effects specifications following Hospido (2012). Hospido (2012) uses the structure of the Hessian to make use of block matrix multiplication and partitioned inversion. In addition to exploiting the structure of the Hessian, we tailor the block matrix multiplication to the specific structure of the data in our simulation. All of this allows us to directly compute the Newton step at a low cost while using little memory.

We are not aware of an available implementation of Hospido's approach to multinomial logit models and coded it ourselves. Appendix B describes the implementation details for Newton's method. Our implementation of Newton steps is far superior to what appears in standard implementations, such as in R or Stata. For instance, in our experience, a single iteration of a standard implementation with 10,000 fixed effects can take more time than our implementation takes to complete estimation with 1,000,000 fixed effects. In addition, our implementation has much lower memory requirements, and so can more easily accommodate the large number of fixed effects we are interested in. Because

existing implementations of multinomial logit estimation were so slow or simply infeasible because of memory issues, we compare the MM algorithm to our bespoke implementation of Newton's method rather than an existing implementation.

7 Performance of the MM Algorithm

In this section, we demonstrate the performance of the MM algorithm relative to Adam and Newton's method through two simulations. The first simulation considers how our method scales with the number of fixed effects and the second considers how our method scales with the number of individual-specific coefficients.

7.1 Fixed Effects

We consider a setting where the number of covariates is small and the number of fixed effects grows with the sample size. In this simulation, I consumers choose from three alternatives, $J = \{1, 2, 3\}$ in $T = 20$ time periods. Consumers choose the alternative that yields the highest utility. Consumer i's utility from choosing option j in time period t is:

$$u_{ijt} = x_{it}\beta_j + \alpha_{ij} + \varepsilon_{ijt} \tag{14}$$

where x_{it} is a single individual-level characteristic; α_{ij} are individual-alternative fixed effects; and ε_{ijt} is a scalar $i.i.d$ idiosyncratic shock distributed according to a Type I Extreme Value distribution. We draw x_{it} and α_{ij} from standard normals and set $\boldsymbol{\beta} = (\beta_1, \beta_2, \beta_3) = (0, 0.5, 1)$. We normalize the utility of the base option to be zero so $\alpha_{i1} = 0,\ \forall i$.

We allow the number of individuals I to vary from 500 to 500,000. Since the model contains individual-alternative fixed effects, the number of fixed effects is twice the number of individuals $2I$. Thus, in our most extreme scenario, we are estimating 1,000,000 fixed effects. For each value of I, we simulate 100 trials. We drop individuals who failed to choose all options at least once as these can create problems for convergence, as would be the case for any fixed effects estimation.

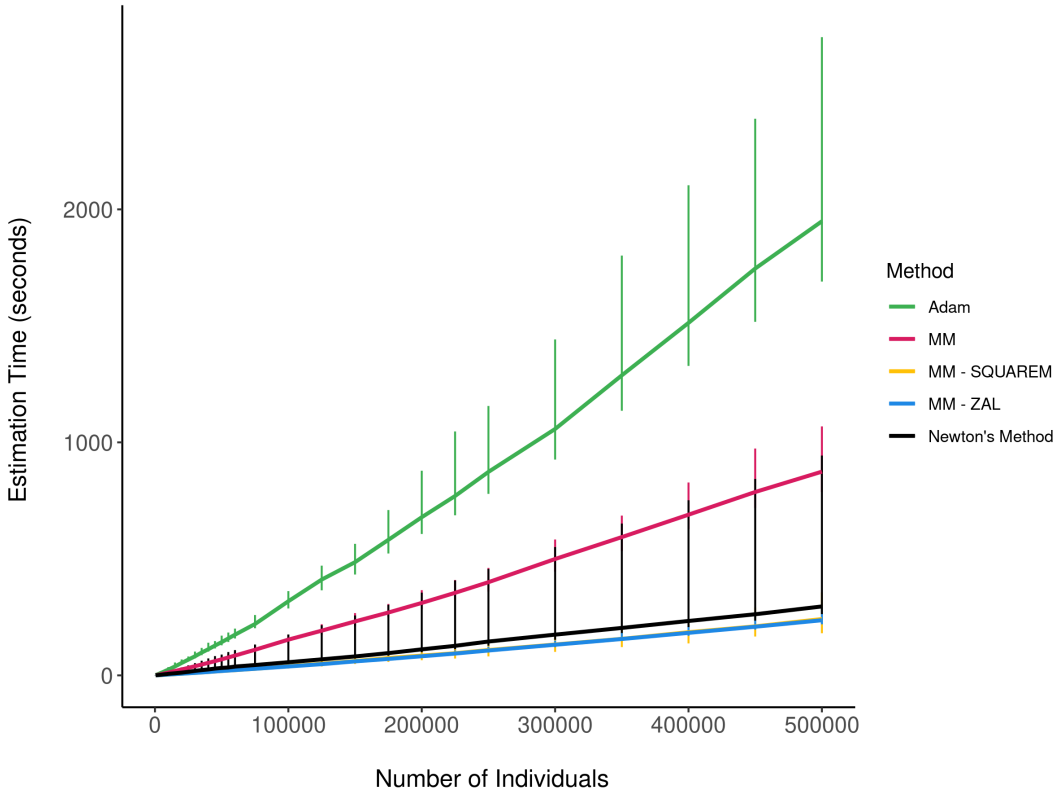
We estimate the model parameters using Newton's method, Adam, the MM algorithm,

and the MM algorithm paired with SQUAREM and ZAL.[11]

We use the same convergence criterion across all of the methods. In particular, we stop the algorithms when the difference in successive values of the log-likelihood is less than 10^{-8}. The choice of the stopping condition does not have a major effect on the results. In addition, we start all of the algorithms from the same initial guess of $\mathbf{0}$.

Figure 2 shows the runtimes for the different methods against the number of individuals I. The figure shows the median runtime for each method. The vertical bars give the 2.5th and 97.5th percentiles of runtimes across the 100 trials. The variation in estimation times comes from the variation in values of the individual-level characteristic x_{it} and the values of fixed effects α_{ij}. Since we drop individuals who chose an alternative with zero probability, there is some variation in the number of observations across trials.

Figure 2



Notes: The main line shows the median runtime. The bottom and top of vertical bars represents the 2.5th percentile and the 97.5th percentile of runtimes, respectively.

[11]We performed all calculations on a 2.6 GHz Intel Xeon E5-2670 using 8 cores with 8 GB of memory per code.

All of the methods yield roughly the same values across runs. The differences between the estimates are less than 10^{-6} and lack any practical significance. The minuscule differences confirm that the MM algorithms are indeed converging to the same estimates as Newton's method and Adam, the maximum likelihood estimates. Thus, the performance of the different algorithms is reflected in the differences in runtimes.

Newton's method is the faster of the maximum likelihood estimators in this simulation. Newton's method is around more than 7 times faster than Adam on average. Newton's method take only 4 minutes to estimate 1,000,000 fixed effects. In contrast, Adam takes nearly 30 minutes.

Our MM algorithm outperforms Adam, but not Newton's method. The base algorithm is roughly 2 times faster than Adam, but 3 times slower than Newton's method on average.[12] Newton's method, however, exhibits a fair amount of variability so there are trials where our base algorithm outperforms Newton's method.

Accelerating the MM algorithm using SQUAREM or ZAL offers additional speed improvements over the base algorithm and allows the MM algorithm to slightly outperform Newton's method. The two acceleration methods perform equally well on average. However, SQUAREM results in slightly more variable run times across trials in this simulation.

The number of iterations to convergence provides additional clarity on the performance of the different methods. Figure 3 shows the number of iterations taken by each method plotted against the number of individuals. As before, the vertical bars show the variation between trials, with the bottom representing the 2.5th percentile and the top representing the 97.5th percentile.

Newton's method takes the fewest iterations. In this simulation, Newton's method always takes 6 iterations to converge. Newton's method therefore exhibits faster convergence, but a high cost per iteration. In contrast, Adam has a low cost per iteration but takes the most iterations to converge. On average, Adam takes almost 300 iterations, 70 more than the MM algorithm on average.

The figure also shows that the speed improvement from acceleration methods comes from

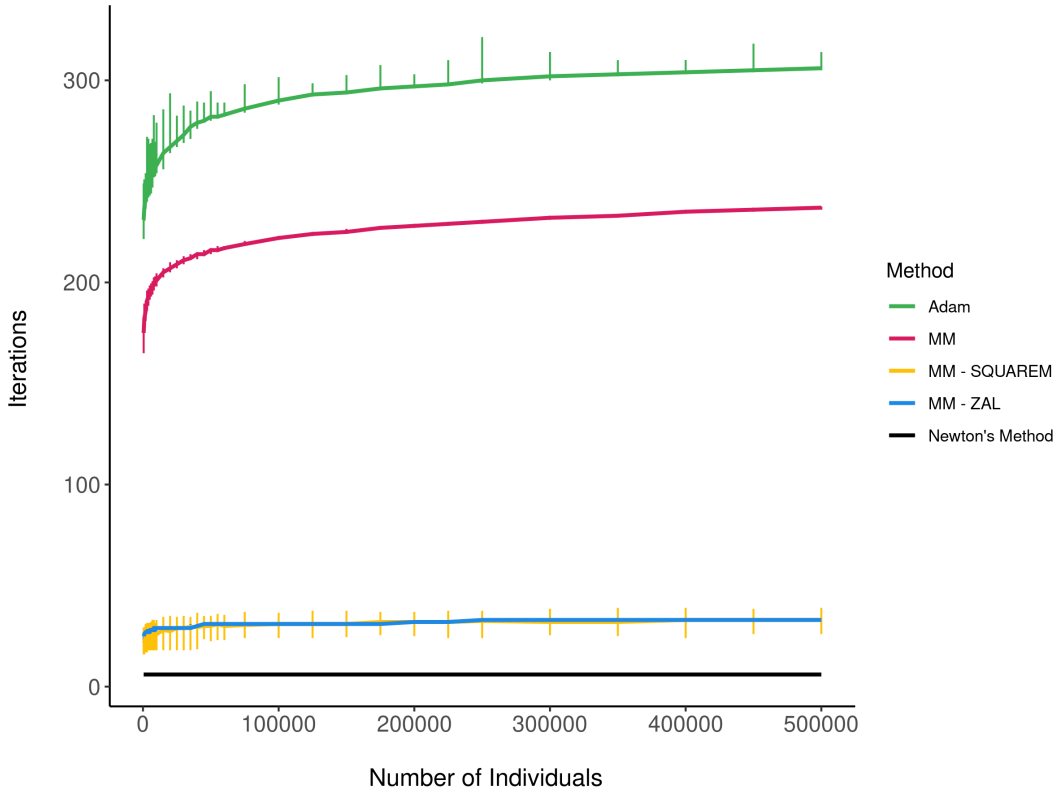
[12]There may exist a step size (or schedule of step sizes) that allows Adam to outperform the MM algorithm. However, the time spent finding this step size would likely far exceed any reduction in computation time.

the drastic reduction in the number of iterations. SQUAREM and ZAL take around 25 iterations on average. We should note that a single iteration of SQUAREM and ZAL involves two iterations of the MM algorithm, followed by an extrapolation step. SQUAREM and ZAL are therefore performing around 50 iterations of the MM algorithm and 50 extrapolations. The extrapolation step therefore saves roughly 130 iterations of the MM algorithm.

Thus far, we have evaluated the methods based solely on their runtimes. However, programming complexity is another important consideration. The extra time spent coding a faster algorithm may outweigh any runtime savings.

Out of the algorithms considered, we found Newton's method to be the most involved to code. For Newton's method to perform well, the calculation of the Hessian needs to be efficient, which means using analytical expressions for the building blocks of the Hessian and avoiding any unnecessary calculations.

Figure 3



Notes: The main line shows the median number of iterations until convergence. The bottom and top of vertical bars represents the 2.5th percentile and the 97.5th percentile of iterations, respectively.

Simply following the approach to exploiting the sparsity of the Hessian described in Hospido (2012) is not enough for Newton's method to outperform the MM algorithm. In fact, our early implementations performed worse than the base MM algorithm.[13] These less-optimized versions of Newton's method calculated the individual building blocks of the Hessian using loops and matrix operations. In comparison, our final version relies more heavily on vectorization and calculates individual elements of Hessian using group sums of vectors. While faster, the final version is less general. Whereas the preliminary version accommodates different data shapes, the final version works only with three alternatives.

In our experience, the time that we spent optimizing and testing Newton's method far exceeded the eventual improvement in runtime over the base MM algorithm. Moreover, in far less time, we were able to match or do better than the well-optimized version of Newton's method by adding acceleration, which is easy and straightforward to implement.

The extensibility of the code also matters to researchers. With our optimized version of Newton's method, any changes to the specification, such as changing the number of alternatives, requires recoding the building blocks of the Hessian, which would be involved. In contrast, changing the specification for the MM algorithm does not affect the code for the algorithm, only the inputs. From our experience, the time spent optimizing and adapting Newton's method to new specifications outweighs any additional performance that Newton's method has over the MM algorithm. Overall, the MM algorithm and its accelerated versions deliver excellent performance while being simple to code and extend.

7.2 Individual-Specific Coefficients

We now consider how the MM algorithm scales with the number of individual-specific coefficients. This simulation is similar to the previous one, but we replace the fixed effect with an alternative-specific variable that has individual-specific coefficients. As before, I consumers choose from three alternatives, $J = \{0, 1, 2\}$, but now in $T = 50$ time periods.[14] Consumers choose the alternative that yields the highest utility. Consumer

[13] A simulation showing the performance of an early, less-optimized Newton's method against the MM algorithm is available upon request.

[14] We increase the length of the time-series to reduce the number of times that a consumer chooses an alternative no more than one time. These cases can cause convergence problems for all of the algorithms.

i's utility from choosing option j in time period t is now:

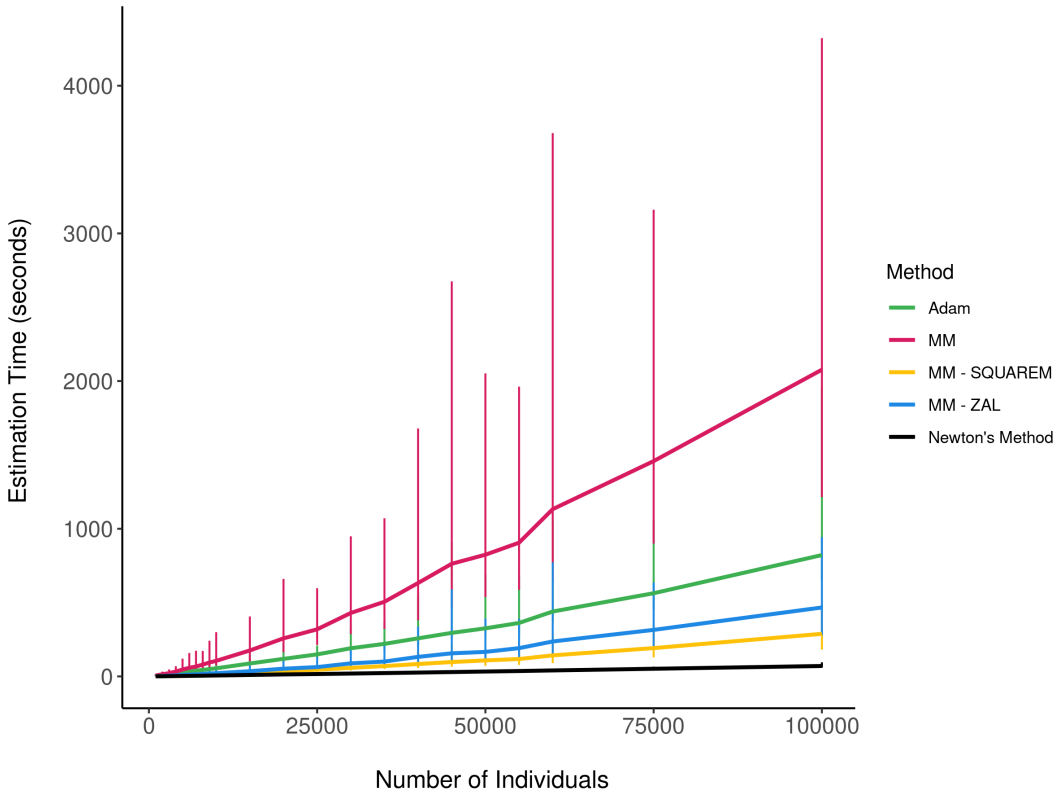
$$u_{ijt} = x_{it}\beta_j + w_{jt}\gamma_i + \varepsilon_{ijt} \tag{15}$$

where x_{it} is a single individual-level characteristic; w_{jt} is a single alternative-level characteristic; and ε_{ijt} is a scalar $i.i.d$ idiosyncratic shock distributed according to a Type I Extreme Value distribution. We draw x_{it} and w_{jt} from standard normals. The individual-specific coefficients γ_i is drawn from a normal $\mathcal{N}(0, 0.5^2)$. As in the previous simulation, we set $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2) = (0, 0.5, 1)$.

We allow the number of individuals I to vary from 1,000 to 100,000. For each value of I, we simulate 100 trials. As in the previous simulation, we drop individuals who do not choose all of the options at least once.

We estimate the model parameters using the same algorithms as before. Figure 4 shows the runtimes for the various methods. The main line gives the median runtime across the 100 simulations. As before, the vertical bar captures the variation across trials. The bottom of the vertical bars represents the 2.5th percentile and the top represents the 97.5th percentile.

Figure 4



Notes: The main line shows the median runtime. The bottom and top of vertical bars represents the 2.5th percentile and the 97.5th percentile of runtimes, respectively.

In this simulation, the MM algorithm is slower than Adam on average. The slower speed stems from the generalized demeaning. Projecting out the alternative-specific variable with its individual-specific coefficients is more intensive than subtracting off group means. So it is not surprising that the MM algorithm loses some of its computational advantage. The accelerated versions are still faster than Adam on average, with SQUAREM being faster than ZAL.

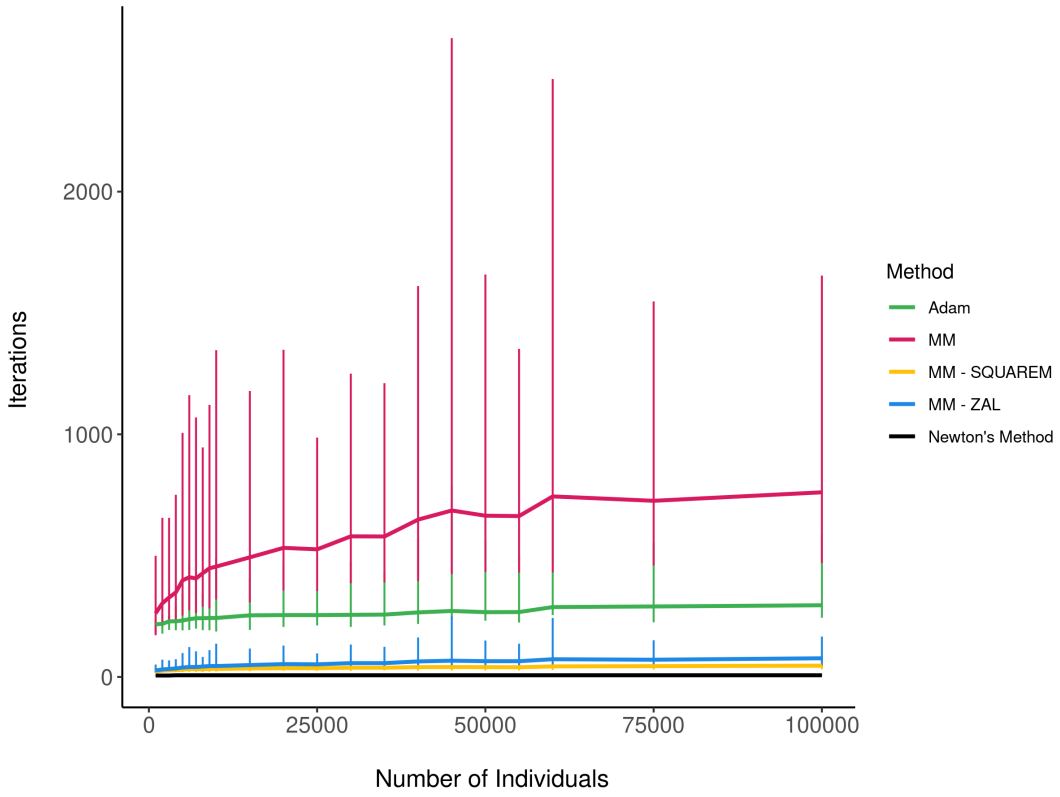
As before, Newton's method outperforms the base algorithm. However, unlike in the first simulation, Newton's method now outperforms the accelerated versions of the MM algorithm. This is again due to the increased burden from generalized demeaning. For 100,000 individuals, Newton's method is approximately 6.5 times faster than the accelerated algorithms.

While Newton's method is faster, it also takes significantly longer to program and optimize. For this simulation, we found that the increased programming time far outweighed

any runtime savings. As the previous simulation showed, Newton's method is faster only when using well-optimized code, optimized for the specific data structure in a way that limits extensibility. Less optimized coding can eliminate the marginal speed improvement that Newton's method provides in this case.

The number of iterations to convergence provides additional insights about the runtimes. Figure 5 shows the number of iterations taken by each method plotted against the number of individuals with the vertical bars showing the variation across trials.

Figure 5



Notes: The main line shows the median number of iterations until convergence. The bottom and top of vertical bars represents the 2.5th percentile and the 97.5th percentile of iterations, respectively.

The MM algorithm takes between 200 and 500 more iterations than Adam. Moreover, the number of iterations for the MM algorithm is far more variable. In some cases, the MM algorithm requires more than 2000 iterations to converge. This variability in the number of iterations is reflected in the variability in runtimes. In contrast, Newton's method consistently converges in 6 to 7 iterations. Each iteration is costly, but there are not many iterations.

Figure 5 also highlights the value of acceleration methods. Using accelerations methods results in approximately a 10-fold decrease in the number of iterations for the MM algorithm. The extrapolation step allows the MM algorithm to jump ahead and avoid areas of slow convergence.

8 Empirical Application

8.1 Modeling Hospital Choice

In this section, we further demonstrate the utility of our method by providing an empirical application to hospital choice. Economists frequently rely on discrete choice models to identify the degree of substitution between hospitals and study competition in healthcare markets (e.g. Gaynor, Kleiner, and Vogt, 2013; Gowrisankaran, Nevo, and Town, 2015; Garmon, 2017). When evaluating hospital mergers, antitrust practitioners use multinomial logit models to estimate diversion ratios that measure the level of competition between the merging parties (e.g. Capps et al., 2018). In a hospital setting, these diversions represent the share of patients who would switch from one hospital to another if the first hospital went out of network. Due to the availability of patient-level discharge data, practitioners often use individual-level logit models.

In the logit model, the functional form of the utilities dictates the allowed substitution patterns. Greater flexibility is therefore crucial for recovering realistic substitution patterns. Due to the shortcomings of existing estimation methods, (Raval, Rosenbaum, and Tenn, 2017) provide a semiparametric bin estimator that can capture rich substitution patterns.[15] Their method partitions individuals into groups based on a set of G characteristics and a minimum group size. They estimate separate multinomial logit models for each group, with hospital intercepts as the only explanatory variables. Their estimator is equivalent to a multinomial logit with hospital-group fixed effects.

The semiparametric approach of (Raval, Rosenbaum, and Tenn, 2017) has two main drawbacks. The first is that it can accommodate only discrete variables. Continuous variables, like travel time or the acuity weight, need to be discretized. The second drawback comes from the binning process, which prevents groups from overlapping.

[15]The method has seen use in litigated hospital mergers, e.g. Advocate-Northshore (Tenn and Vandergrift, 2017).

This process effectively rules out some forms of multi-way fixed effects.

These drawbacks could be addressed by estimating on the whole population simultaneously and treating intercepts as hospital-group fixed effects. However, doing so leads to a large number of fixed effects. This leads to the use of the MM algorithm, we allows for multinomial logit models that have continuous variables, many fixed effects, and multi-way fixed effects. The MM algorithm, therefore, expands the set of models that practitioners can consider and gives additionally flexibility beyond what the semiparametric method can offer.

8.2 Diversion from Natural Disasters

We revisit Raval, Rosenbaum, and Wilson (RRW, 2022), which compares diversion ratios estimated from logit models to observed diversions from hospital closures due to natural disasters. We consider one of the four natural disasters considered in that paper, the Americus Tornado. The Americus Tornado swept through Americus, GA on March 1, 2007, damaging Sumter Regional Hospital. The hospital remained closed for a year, eventually reopening on April 1, 2008. Prior to closure, Sumter hospital accounted 50.4% of patient admissions within its service area (RWW). The Americus Tornado therefore forced a large number of patients to divert to the surrounding eight hospitals or to the outside option.

The data consists of all hospital admissions from the state of Georgia in 2006, 2007, and 2008. The data set includes information about each patient such as age, zip code, and gender, among other variables. It also includes information about treatment including the area of diagnosis and length of stay. RRW merge the discharge data with American Hospital Association data on hospital characteristics. We do not use the full dataset, but instead use the estimation sample from RRW. This sample consists of admissions within the 90% service area of Sumter hospital. The 90% service area is the smallest set of zip codes that accounts for 90% of patient admissions to that hospital.

Following RRW, we calculate the observed diversions using the difference in shares between the period prior to closure and the period just after closure. The pre-period runs from January 1, 2006 to February 28, 2007 and the post-period runs from March 1, 2007

to March 31, 2008. The observed diversions from Sumter hospital to hospital k are:

$$D^k_{observed} = \frac{s^{post}_k - s^{pre}_k}{s^{pre}_{Sumter}} \tag{16}$$

where s is the observed share of admissions. We compare the observed diversions to predicted diversions from a logit model estimated on the pre-period data. Given the logit form, the predicted diversions from Sumter to hospital k are:

$$\hat{D}^k = \sum_i \frac{\hat{s}_{ik}}{1 - \hat{s}_{i,Sumter}} \frac{\hat{s}_{i,Sumter}}{\sum_i \hat{s}_{i,Sumter}} \tag{17}$$

where i consists of patients who choose Sumter hospital and $\hat{s}$ are the predicted shares from the logit model. We examine how well different models perform in predicting the observed diversions.

8.3 Models

We consider four logit models. We take the first three models from RRW. The first model *Semipar* uses the semiparametric method, grouping individuals according to their zip code, disease acuity, age group, and area of diagnosis. Under *Semipar*, we estimate a logit model with no explanatory variables, just hospital-specific constant terms, separately for each group. The model is equivalent to a logit model with 816 hospital-group indicators. The second model *GNT* follows Gowrisankaran, Nevo, and Town (2015) and includes travel time, travel time squared, hospital fixed effects, and interactions of demographic and hospital characteristics with travel time. The third model *Ho* follows Ho (2006) and includes time, time squared, hospital fixed effects, and interactions between hospital and patient characteristics.

The final model *Time+ZIP* is our own specification. It includes travel time, travel time squared, and hospital fixed effects. It also includes zip code-hospital indicators for three zip codes. These three zip codes had a large number of patients who received treatment at Sumter hospital in the pre-period. Together with hospital fixed effects, our model has 32 indicator variables. While not as many as *Semipar*, *Time+ZIP* has continuous variables, which the semiparametric method cannot accommodate.

We estimate *Ho*, *GNT*, and *Time+ZIP* using the MM algorithm paired with SQUAREM. The flexibility of the MM algorithm allows us to accommodate all of these different

specifications. While we could have estimated *Semipar* with the MM algorithm, we chose to use the estimation approach in Raval, Rosenbaum, and Tenn (2017) since it was easily available to us.

8.4 Predictive Performance

We evaluate the four models based on their prediction errors, $e_k = \hat{D}^k - D^k_{observed}$. In particular, we focus on the mean squared prediction errors. We use bootstrapping to compute 95% confidence intervals. As in RRW, we boostrap the pre- and post-period datasets to account for variation in the observed and predicted diversions. While the predicted diversions depend only on the variation in the pre-period data, the observed diversions depend on the variation in both periods.

Figure 6

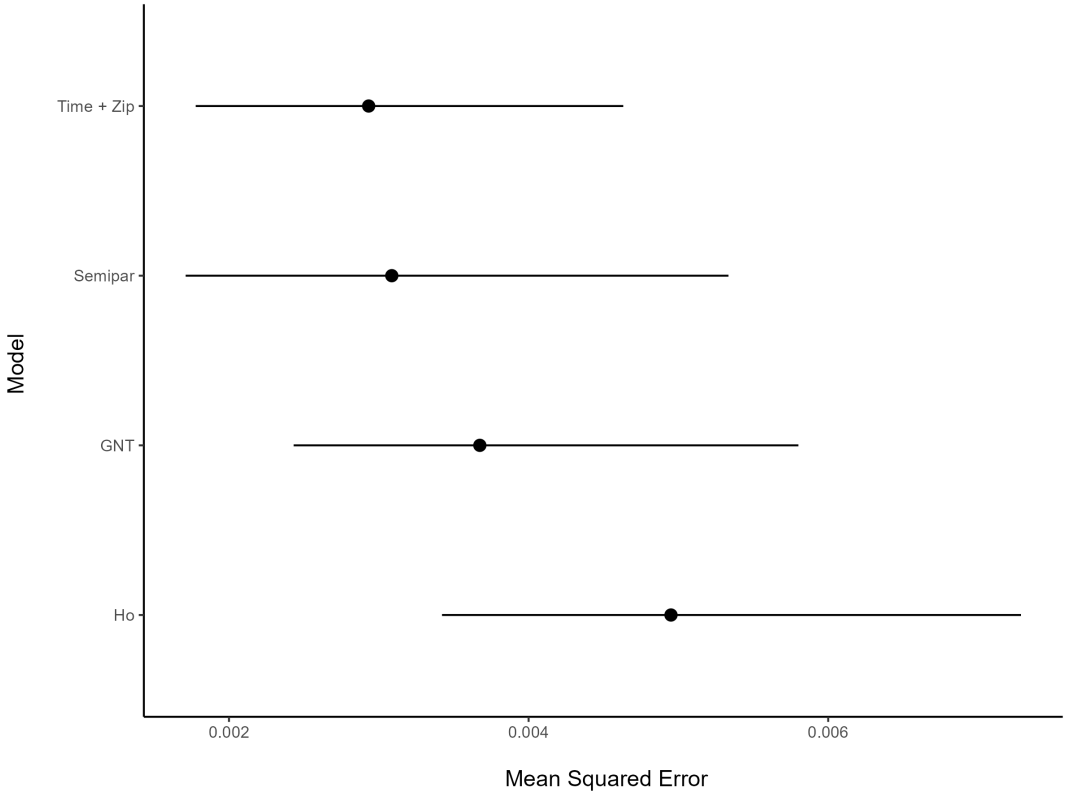


Figure 6 shows the 95% bootstrapped confidence intervals for the different models. From the figure, we see that *Time+ZIP* is best performing model, slightly edging out *Semipar*. *GNT* and *Ho* are the worst performing models on average.

As RRW note, including patient characteristics and interactions does not necessarily help us predict diversions. *GNT* and *Ho* perform the worst even though they include many patient characteristics and various interactions. The results also highlight that we do not need hundreds of group-hospital indicators to capture rich substitution patterns. A model that incorporates travel time with many fixed effects may perform just as well as the semiparametric approach. Another approach we could take would be to add a continuous term, such as patient distance interacted with age, to the 816 hospital-group interactions in RRW, which naturally would allow us to improve fit.

The MM algorithm allows practitioners to estimate a wide range of logit models. We were able to estimate models with a variety of covariates and fixed effects. In this specific example, the MM algorithm allowed us to estimate a logit model that incorporated continuous variables and fixed effects, allowing us to outperform the semiparametric method.

9 Conclusion

We present a new method for estimating multinomial models with many fixed effects. Existing implementations rely on non-linear estimation of many dummy variable coefficients, which is computationally slow and memory intensive. We found these methods to be infeasible for the applications we have in mind. Our approach relies on the Minorization-Maximization (MM) Algorithm, which can be seen as a generalization of the better-known Expectation-Maximization (EM) algorithm. Our implementation maximizes the likelihood through an iterative series of linear regressions, allowing fixed effects to be handled through the use of linear panel methods. Our method is substantially faster and more memory efficient than previously available implementations.

We show the performance of the MM algorithm through two Monte Carlo simulations. Our method dramatically faster and more efficient than standard estimation packages, such as Stata or R. Our simulations focus on comparing our method to two alternatives: a highly-optimized Newton step algorithm exploiting both the sparsity of the Hessian (as in Hospido, 2012) and the specific data structure of our simulation problem, and Adam, a line search method recommended by Du, Kanodia, and Athey, 2023. Our method outperforms the alternatives in estimating fixed effects. We also consider models with individual-specific coefficients on a continuous explanatory variable. In this case,

our method outperforms the approach based on Adam substantially. Newton's method offers an improvement on average in this case, but the found the improvement is small relative to the level of bespoke programming required to achieve it. For our method, we consider acceleration methods designed for MM and EM algorithms. The use of acceleration methods provides noticeable speed improvements over the base algorithm, and we recommend that practitioners use our method in conjunction with SQUAREM acceleration.

Our method provides a new way to estimate multinomial logit models in a variety of settings. We demonstrate one potential use to estimating models of hospital choice and show how our method provides greater flexibility for estimating diversions relative to Raval, Rosenbaum, and Tenn, 2017, a recent popular approach. Overall, our method expands the set of specifications that practitioners can consider given their limited time and computational resources.

Appendices

Appendix A Proof of Theorem 1

We show that $S(\boldsymbol{\theta};\boldsymbol{\theta}^{(k)})$ is a minorization of $\mathcal{L}(\boldsymbol{\theta}^{(k)})$ at $\boldsymbol{\theta}^{(k)}$ by verifying:

(1) $\underline{S(\boldsymbol{\theta};\boldsymbol{\theta}^{(k)}) \leq \mathcal{L}(\boldsymbol{\theta})}$

We start from a representative individual i. Let $\boldsymbol{\psi}_{it} = x_{it}\boldsymbol{\beta} + \boldsymbol{w}_t\gamma_i + \boldsymbol{\alpha}_i$. The multinomial logit likelihood for observation (i,t) is:

$$l(\boldsymbol{\psi}_{it};\boldsymbol{y}_{it}) = \prod_{j=1}^{J}\left(\frac{\exp(\psi_{ijt})}{\sum_{k=1}^{J}\exp(\psi_{ikt})}\right)^{y_{ijt}}.$$

The score function and Hessian are:

$$h_j(\boldsymbol{\psi}_{it};\boldsymbol{y}_{it}) = y_{ijt} - p_{ijt}(\psi_{ijt}),$$
$$h_{jk}(\boldsymbol{\psi}_{it};\boldsymbol{y}_{it}) = \begin{cases} -p_{ijt}(\psi_{ijt})(1-p_{ijt}(\psi_{ijt})), & j=k \\ p_{ijt}(\psi_{ijt})p_{ikt}(\psi_{ikt}), & j\neq k \end{cases}$$

The second-order Taylor expansion of $\log l(\boldsymbol{\psi}_{it};\boldsymbol{y}_{it})$ at $\tilde{\boldsymbol{\psi}}_{it} = \boldsymbol{\psi}_{it}^{(k)}$ is:

$$\log\, l(\boldsymbol{\psi}_{it};\boldsymbol{y}_{it}) = \log\, l(\tilde{\boldsymbol{\psi}}_{it};\boldsymbol{y}_{it}) + (\boldsymbol{\psi}_{it}-\tilde{\boldsymbol{\psi}}_{it})'\nabla \log\, l(\tilde{\boldsymbol{\psi}}_{it};\boldsymbol{y}_{it}) + \frac{1}{2}(\boldsymbol{\psi}_{it}-\tilde{\boldsymbol{\psi}}_{it})'\nabla^2 \log\, l(\boldsymbol{\psi}_{it}^{*};\boldsymbol{y}_{it})(\boldsymbol{\psi}_{it}-\tilde{\boldsymbol{\psi}}_{it}) \tag{18}$$

Given that

$$\nabla^2 \log\, l(\boldsymbol{\psi}_{it}^{*};\boldsymbol{y}_{it}) = \begin{bmatrix} h_{11}(\boldsymbol{\psi}_{it}^{*};\boldsymbol{y}_{it}) & h_{12}(\boldsymbol{\psi}_{it}^{*};\boldsymbol{y}_{it}) & \dots & h_{1J}(\boldsymbol{\psi}_{it}^{*};\boldsymbol{y}_{it}) \\ h_{21}(\boldsymbol{\psi}_{it}^{*};\boldsymbol{y}_{it}) & h_{22}(\boldsymbol{\psi}_{it}^{*};\boldsymbol{y}_{it}) & \dots & h_{2J}(\boldsymbol{\psi}_{it}^{*};\boldsymbol{y}_{it}) \\ \vdots & \vdots & & \vdots \\ h_{J1}(\boldsymbol{\psi}_{it}^{*}) & h_{J2}(\boldsymbol{\psi}_{it}^{*};\boldsymbol{y}_{it}) & \dots & h_{JJ}(\boldsymbol{\psi}_{it}^{*};\boldsymbol{y}_{it}) \end{bmatrix} \tag{19}$$

$$= \begin{bmatrix} -p_{i1t}(1-p_{i1t}) & p_{i1t}p_{i2t} & ... & p_{i1t}p_{iJt} \\ p_{i2t}p_{i1t} & -p_{i2t}(1-p_{i2t}) & ... & p_{i2t}p_{iJt} \\ \vdots & \vdots & & \vdots \\ p_{iJt}p_{i1t} & p_{i1t}p_{i2t} & ... & -p_{iJt}(1-p_{iJt}) \end{bmatrix}, \quad (20)$$

we have $\nabla^2 \log\ l(\boldsymbol{\psi}_{it}^*; \boldsymbol{y}_{it}) \succeq -\boldsymbol{I}$, meaning $\nabla^2 \log\ l(\boldsymbol{\psi}_{it}^*; \boldsymbol{y}_{it}) + I$ is positive semi-definite.

Therefore:

$$\begin{aligned} \log\ l(\boldsymbol{\psi}_{it}; \boldsymbol{y}_{it}) &\geq \log\ l(\tilde{\boldsymbol{\psi}}_{it}; \boldsymbol{y}_{it}) + (\boldsymbol{\psi}_{it} - \tilde{\boldsymbol{\psi}}_{it})' \nabla \log\ l(\tilde{\boldsymbol{\psi}}_{it}; \boldsymbol{y}_{it}) - \frac{1}{2}(\boldsymbol{\psi}_{it} - \tilde{\boldsymbol{\psi}}_{it})'(\boldsymbol{\psi}_{it} - \tilde{\boldsymbol{\psi}}_{it}) \\ &= \log\ l(\tilde{\boldsymbol{\psi}}_{it}; \boldsymbol{y}_{it}) - \sum_j h_j(\tilde{\boldsymbol{\psi}}_{it}; \boldsymbol{y}_{it})(\tilde{\psi}_{ijt} - \psi_{ijt}) - \frac{1}{2}\sum_j (\tilde{\psi}_{ijt} - \psi_{ijt})^2 \\ &= \log\ l(\tilde{\boldsymbol{\psi}}_{it}; \boldsymbol{y}_{it}) + \frac{1}{2}\sum_j h_j(\tilde{\boldsymbol{\psi}}_{it}; \boldsymbol{y}_{it})^2 - \frac{1}{2}\sum_j (\tilde{\psi}_{ijt} + h_j(\tilde{\boldsymbol{\psi}}_{it}; \boldsymbol{y}_{it}) - \psi_{ijt})^2 \end{aligned} \quad (21)$$

After algebraic manipulation and summing over all observations, we obtain:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}) \geq \mathcal{L}(\boldsymbol{\theta}^{(k)}) &+ \frac{1}{2}\sum_{i,j,t} h_j(x_{it}\boldsymbol{\beta}^{(k)} + \boldsymbol{w}_t\gamma_i^{(k)} + \boldsymbol{\alpha}_i^{(k)}; \boldsymbol{y}_{it})^2 \\ &- \frac{1}{2}\sum_{i,j,t}(x_{it}\boldsymbol{\beta}^{(k)} + w_{jt}\gamma_i^{(k)} + \alpha_{ij}^{(k)} + h_j(x_{it}\boldsymbol{\beta}^{(k)} + \boldsymbol{w}_t\gamma_i^{(k)} + \boldsymbol{\alpha}_i^{(k)}; \boldsymbol{y}_{it}) \\ &- (x_{it}\boldsymbol{\beta} + w_{jt}\gamma_i + \alpha_{ij}))^2 \end{aligned} \quad (22)$$

where the RHS of the above inequality as $S(\boldsymbol{\theta}; \boldsymbol{\theta}^{(k)})$.

(2) $\underline{S(\boldsymbol{\theta}^{(k)}; \boldsymbol{\theta}^{(k)}) = \mathcal{L}(\boldsymbol{\theta}^{(k)})}$

Direct substitution shows:

$$\begin{aligned} S(\boldsymbol{\theta}^{(k)}; \boldsymbol{\theta}^{(k)}) &= \mathcal{L}(\boldsymbol{\theta}^{(k)}) + \frac{1}{2}\sum_{i,j,t} h_j(\boldsymbol{\psi}^{(k)}; \boldsymbol{y}_{it})^2 - \\ &\qquad \frac{1}{2}\sum_{i,j,t}(\boldsymbol{\psi}^{(k)} + h_j(\boldsymbol{\psi}^{(k)}; \boldsymbol{y}_{it}) - \boldsymbol{\psi}^{(k)})^2 \\ &= \mathcal{L}(\boldsymbol{\theta}^{(k)}) + \frac{1}{2}\sum_{i,j,t} h_j(\boldsymbol{\psi}^{(k)}; \boldsymbol{y}_{it})^2 - \frac{1}{2}\sum_{i,j,t} h_j(\boldsymbol{\psi}^{(k)}; \boldsymbol{y}_{it})^2 \\ &= \mathcal{L}(\boldsymbol{\theta}^{(k)}) \end{aligned}$$

(3) $\nabla^2_\theta S(\boldsymbol{\theta}; \boldsymbol{\theta}^{(k)})$ exists, and $\nabla^2_\theta S(\boldsymbol{\theta}^{(k+1)}; \boldsymbol{\theta}^{(k)})$ is negative definite

To ease notation, we consider $\boldsymbol{\alpha}_i$ as coefficients on indicator variables, combine these indicator variables with x_{it} and w_{jt} and denote the combined vector as z_{ijt}. Then by the definition of $S(\boldsymbol{\theta}; \boldsymbol{\theta}^{(k)})$ in Eq.(22), $\nabla^2_\theta S(\boldsymbol{\theta}; \boldsymbol{\theta}^{(k)}) = -\boldsymbol{Z}'\boldsymbol{Z}$ where $\boldsymbol{Z}$ is the matrix with $z_{ijt},\ \forall i, j, t$ stacked by rows, and does not depend on $\boldsymbol{\theta}$. Because $\boldsymbol{Z}'\boldsymbol{Z}$ is positive definite under sufficient variation of x_{it} across time and individuals, we have our result.

Appendix B Implementation Details for Simulation 1

This section describes the implementation details for Newton's method and Adam used in the simulation in section 7.1.

B.1 Newton's Method

Our implementation of Newton's method follows Hospido (2012) and exploits structure and sparsity of the Hessian to efficiently calculate the Newton step by making use of block inversion and block matrix multiplication. We first show the structure of the Hessian and then show how we exploit that structure to calculate the Newton step.

B.1.1 The Gradient and Hessian

For background, recall that the parameters to be estimated are $\boldsymbol{\theta} = (\beta_2, \beta_3, \{\alpha_{i2}, \alpha_{i3}\}_{i=1,\dots I})$. We rewrite the likelihood function as $l(x_{it}\boldsymbol{\beta} + \boldsymbol{\alpha}_i; \boldsymbol{y}_{it}) = l(\boldsymbol{\psi}_{it}; \boldsymbol{y}_{it})$. We suppress w_{jt} and γ_i from our notation because we do not use these terms in this simulation. For each individual i and each time period t,

$$\frac{\partial \log l(\boldsymbol{\psi}_{it}; \boldsymbol{y}_{it})}{\partial \boldsymbol{\theta}} = \frac{\partial \log l(\boldsymbol{\psi}_{it}; \boldsymbol{y}_{it})}{\partial \boldsymbol{\psi}_{it}} \times \frac{\partial \boldsymbol{\psi}_{it}}{\partial \boldsymbol{\theta}} = (\boldsymbol{y}_{it} - \boldsymbol{p}_{it})' \times \frac{\partial \boldsymbol{\psi}_{it}}{\partial \boldsymbol{\theta}} \tag{23}$$

where $\dfrac{\partial \boldsymbol{\psi}_{it}}{\partial \boldsymbol{\theta}}$ is a $2 \times (1 + I)$ matrix.

Let $\text{diag}_2(a)$ be a 2×2 matrix with diagonal elements as a and off-diagonal elements as 0,

$$\begin{aligned}
\left[\frac{\partial \boldsymbol{\psi}_{it}}{\partial \boldsymbol{\theta}}\right]_{.;1:2} &= \frac{\partial \boldsymbol{\psi}_{it}}{\partial \boldsymbol{\beta}} &= \text{diag}_2(x_{it}) \\
\left[\frac{\partial \boldsymbol{\psi}_{it}}{\partial \boldsymbol{\theta}}\right]_{.;2\times i+1:2} &= \frac{\partial \boldsymbol{\psi}_{it}}{\partial \boldsymbol{\alpha}_i} &= \text{diag}_2(1)
\end{aligned}$$

while the other elements are zero because $\boldsymbol{\alpha}_h$ is not in $l(\boldsymbol{\psi}_{it}; \boldsymbol{y}_{it})$ if $h \neq i$.

Therefore, Equation 33 can be rewritten as:

$$((y_{i2t} - p_{i2t})x_{it}, (y_{i3t} - p_{i3t})x_{it}, 0, \cdots, 0, y_{i2t} - p_{i2t}, y_{i3t} - p_{i3t}, 0, \cdots, 0)$$

The gradient of the likelihood $\nabla\mathcal{L}(\theta)$ is:

$$\nabla\mathcal{L}(\theta) = \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}}; \frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}_1}, \ldots, \frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}_I}\right) = \tag{24}$$

$$\left(\sum_{i,t}(y_{i2t} - p_{i2t})x_{it}, \sum_{i,t}(y_{i3t} - p_{i3t})x_{it}; \sum_t y_{12t} - p_{12t}, \sum_t y_{13t} - p_{13t}, \ldots, \sum_t y_{I2t} - p_{I2t}, \sum_t y_{I3t} - p_{I3t}\right) \tag{25}$$

For ease of notation, we define:

$$H_{\boldsymbol{\psi}_{it}} = \begin{bmatrix} -p_{i2t}(1 - p_{i2t}) & p_{i2t}p_{i3t} \\ p_{i3t}p_{i2t} & -p_{i3t}(1 - p_{i3t}) \end{bmatrix} = \begin{bmatrix} H_{i,11} & H_{i,12} \\ H_{i,21} & H_{i,22} \end{bmatrix} \tag{26}$$

Then the matrix $\dfrac{\partial^2 \log l(\boldsymbol{\psi}_{it}(\boldsymbol{\theta}); \boldsymbol{y}_{it})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}'}$ can be written as:

$$\begin{bmatrix} H_{\boldsymbol{\psi}_{it}} x_{it}^2 & \mathbf{0} & \cdots & \mathbf{0} & H_{\boldsymbol{\psi}_{it}} x_{it} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ H_{\boldsymbol{\psi}_{it}} x_{it} & \mathbf{0} & \cdots & \mathbf{0} & H_{\boldsymbol{\psi}_{it}} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix}$$

And then the Hessian matrix $\nabla^2 \mathcal{L}(\theta)$ is:

$$\frac{1}{I \times T} \sum_{i,t} \frac{\partial^2 \log l(\boldsymbol{\psi}_{it}; \boldsymbol{y}_{it})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}'} =$$

$$\frac{1}{I \times T} \left[\begin{array}{c|cccccc} \sum_{i,t} H_{\boldsymbol{\psi}_{it}} x_{it}^2 & \sum_t H_{\boldsymbol{\psi}_{1t}} x_{1t} & \sum_t H_{\boldsymbol{\psi}_{2t}} x_{2t} & \cdots & \sum_t H_{\boldsymbol{\psi}_{it}} x_{it} & \cdots & \sum_t H_{\boldsymbol{\psi}_{It}} x_{It} \\ \hline \sum_t H_{\boldsymbol{\psi}_{1t}} x_{1t} & \sum_t H_{\boldsymbol{\psi}_{1t}} & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \sum_t H_{\boldsymbol{\psi}_{2t}} x_{2t} & \mathbf{0} & \sum_t H_{\boldsymbol{\psi}_{2t}} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\ \sum_t H_{\boldsymbol{\psi}_{it}} x_{it} & \mathbf{0} & \mathbf{0} & \cdots & \sum_t H_{\boldsymbol{\psi}_{it}} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \cdots & \vdots & \ddots & \vdots \\ \sum_t H_{\boldsymbol{\psi}_{It}} x_{It} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \sum_t H_{\boldsymbol{\psi}_{It}} \end{array}\right] \tag{27}$$

We partition the Hessian into blocks according to the lines in (27). The blocks are:

$$A = \sum_{i,t} H_{\boldsymbol{\psi}_{it}} x_{it}^2 = \sum_i A_i$$

$$B = \left[\sum_t H_{\boldsymbol{\psi}_{1t}} x_{1t} \ \sum_t H_{\boldsymbol{\psi}_{2t}} x_{2t} \cdots \sum_t H_{\boldsymbol{\psi}_{it}} x_{it} \cdots \sum_t H_{\boldsymbol{\psi}_{it}} x_{It} \right]$$

$$= [B_1 \; B_2 \cdots B_i \cdots B_I]$$

$$C = B'$$

$$D = \begin{bmatrix} \sum_t H_{\psi_{1t}} & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \sum_t H_{\psi_{2t}} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \sum_t H_{\psi_{it}} & \cdots & \mathbf{0} \\ \vdots & \vdots & \cdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \sum_t H_{\psi_{It}} \end{bmatrix} \tag{28}$$

$$= \begin{bmatrix} D_1 & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & D_2 & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & D_i & \cdots & \mathbf{0} \\ \vdots & \vdots & \cdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \cdots & D_I \end{bmatrix}$$

B.1.2 Block Inversion

To calculate the inverse of the Hessian $\mathcal{L}(\boldsymbol{\theta})^{-1}$, we use block inversion. With block inversion, we can write the inverse Hessian as:

$$\nabla^2 \mathcal{L}(\theta)^{-1} = \begin{bmatrix} (A - BD^{-1}C)^{-1} & -(A - BD^{-1}C)^{-1}BD^{-1} \\ -D^{-1}C(A - BD^{-1}C)^{-1} & -D^{-1} + D^{-1}C(A - BD^{-1}C)^{-1}BD^{-1} \end{bmatrix} \tag{29}$$

Because D is block diagonal, we can find its inverse by inverting each of the 2×2 diagonal matrices D_i. This simplification greatly reduces the computation burden of inversion.

B.1.3 Newton Step

The Newton step is $\Delta\boldsymbol{\theta} = -\sigma\nabla^2\mathcal{L}(\boldsymbol{\theta})^{-1}\nabla\mathcal{L}$. For the step size, we use $\sigma = 1$. This is the optimal step size in a neighborhood of the solution (Nocedal and Wright, 2006). Outside of that area, the step size must satisfy the Wolfe conditions to guarantee convergence (Nocedal and Wright, 2006). Most implementations of Newton's method start with the unit step size and then use backtracking line search in the event that the Wolfe conditions do not hold. In our simulations, we did not find a need for backtracking so we hard-code the step size as one. The Newton step is simply:

$$\begin{aligned}\Delta\boldsymbol{\theta} &= -\nabla^2\mathcal{L}(\boldsymbol{\theta})^{-1}\nabla\mathcal{L} \qquad (30)\\ &= \begin{bmatrix} (A - BD^{-1}C)^{-1} & -(A - BD^{-1}C)^{-1}BD^{-1} \\ -D^{-1}C(A - BD^{-1}C)^{-1} & -D^{-1} + D^{-1}C(A - BD^{-1}C)^{-1}BD^{-1} \end{bmatrix} \begin{bmatrix} \frac{\partial\mathcal{L}}{\partial\boldsymbol{\beta}} \\ \frac{\partial\mathcal{L}}{\partial\boldsymbol{\alpha}_i} \\ \vdots \\ \frac{\partial\mathcal{L}}{\partial\boldsymbol{\alpha}_I} \end{bmatrix}\end{aligned}$$

Brute-force multiplication is inefficient and slow. Instead, Hospido (2012) uses block matrix multiplication to simplify the Newton step into:

$$\Delta\boldsymbol{\beta} = \left(\sum_i A_i - B_i D_i^{-1} C_i\right)^{-1} \left(\frac{\partial\mathcal{L}}{\partial\boldsymbol{\beta}} - \sum_i B_i D_i^{-1} \frac{\partial\mathcal{L}}{\partial\boldsymbol{\alpha}_i}\right) \qquad (31)$$

and

$$\Delta\boldsymbol{\alpha_i} = D_i^{-1}\left(\frac{\partial\mathcal{L}}{\partial\boldsymbol{\alpha}_i} + B_i\Delta\boldsymbol{\beta}\right) \qquad (32)$$

The matrices A_i, B_i, C_i and D_i^{-1} are all 2×2. We never construct the full inverse Hessian so this formulation is also memory efficient.

To compute the Newton step, we start by calculating the choice probabilities as these are the common building elements of both the gradient and the Hessian. With the probabilities, the calculation of the gradient is straightforward and follows (25). We then compute the building blocks of the inverse Hessian.

Rather than construct $H_{\boldsymbol{\psi}_{it}}$ for each individual, we instead construct the individual elements separately using vector operations. We end up with three vectors: one for $H_{i,11}$, one for $H_{i,12}$, and one for $H_{i,22}$. We do not compute $H_{i,21}$ as $H_{i,12} = H_{i,21}$. We then multiply these vectors by the corresponding values of x_{it} and x_{it}^2 and sum over time to obtain the individual elements of the matrices A_i, B_i, and D_i. To sum over time, we

make use of fast grouped sums for vectors.

We loop over individuals and piece together the individual vector elements to form the final matrices A_i, B_i and D_i. From these components, we can calculate $\Delta\boldsymbol{\beta}$ using (39). With $\Delta\boldsymbol{\beta}$, we loop over individuals again to calculate the steps for the fixed effects using (40).

B.2 Adam

Our implementation of Adam uses the full data set to compute the gradient. This is slightly unusual. Most implementations computes the gradient from a random subsample of data or mini-batch, often consisting of 32 to 512 observations. The small sample reduces the computational burden of each iteration. Mini-batching also improves generalizability by avoiding sharp solutions (Keskar et al., 2017). That is great when the goal is prediction, but here the goal is estimation. In our context, small subsamples lead to slow convergence since each iteration updates only a limited number of individual-alternative fixed effects.

The performance of Adam ultimately depends on the choice of its hyperparameters. Out of the hyperparameters, the step size σ usually has the most influence on the speed of convergence. We use a step size of $\sigma = 0.1$.

While an exhaustive grid search might yield a better choice, performing a grid search for each trial of the simulation is simply not practical. To determine the step size, we instead started with the recommended step size from Kingma and Ba (2014) of $\sigma = 0.001$ and the considered alternative step sizes based on powers of 10. We evaluated the step sizes 0.001, 0.01, 0.1, and 1 on the 100 datasets generated for $I = 1000$. We capped the maximum number of iterations at 5000. Table 1 shows the mean, minimum, and maximum number of iterations that Adam took before terminating.

Table 1: Adam: Number of Iterations

σ	Mean	Min	Max
0.001	5000	5000	5000
0.01	2332.87	2066	2492
0.1	239.03	228	267
1	-	-	-

With the default step size $\sigma = 0.001$, Adam fails to converge before reaching the iteration limit of 5000. This step size is simply too small. Larger step sizes reduce the number of iterations needed to achieve convergence. For $\sigma = 0.01$, Adam takes around 2300 iterations and for $\sigma = 0.1$, takes around 200 iterations. Increasing the step size further to 1 results in Adam diverging. Based on powers of 10, we settled a step size $\sigma = 0.1$, which seems to work well in practice.

Appendix C Implementation Details for Simulation 2

This section describes the implementation details for Newton's method used in the simulation in section 7.2. As in the first simulation, we rely on Hospido (2012). While the individual-specific coefficients change the structure of Hessian, they do not alter the basic approach.

C.1 Newton's Method

C.1.1 The Gradient and Hessian

In second simulation, we estimate the parameters $\boldsymbol{\theta} = (\beta_2, \beta_3, \{\gamma_i\}_{i=1,\dots I})$. As before, we rewrite the likelihood function as $l(x_{it}\boldsymbol{\beta} + w_{jt}\gamma_i; \boldsymbol{y}_{it}) = l(\boldsymbol{\psi}_{it}; \boldsymbol{y}_{it})$. We suppress the fixed effects α_{ij} as they do not appear in this simulation. For each individual i and each time period t,

$$\frac{\partial \log l(\boldsymbol{\psi}_{it}; \boldsymbol{y}_{it})}{\partial \boldsymbol{\theta}} = \frac{\partial \log l(\boldsymbol{\psi}_{it}; \boldsymbol{y}_{it})}{\partial \boldsymbol{\psi}_{it}} \times \frac{\partial \boldsymbol{\psi}_{it}}{\partial \boldsymbol{\theta}} = (\boldsymbol{y}_{it} - \boldsymbol{p}_{it})' \times \frac{\partial \boldsymbol{\psi}_{it}}{\partial \boldsymbol{\theta}} \tag{33}$$

which reduces to:

$$\left((y_{i2t} - p_{i2t})x_{it}, (y_{i3t} - p_{i3t})x_{it}, 0, \cdots, 0, \sum_j (y_{ijt} - p_{ijt})w_{jt}, 0, \cdots, 0\right)$$

The gradient of the likelihood $\nabla\mathcal{L}(\theta)$ is therefore:

$$\nabla\mathcal{L}(\theta) = \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}}; \frac{\partial \mathcal{L}}{\partial \gamma_1}, \dots, \frac{\partial \mathcal{L}}{\partial \gamma_I}\right) = \tag{34}$$

$$\left(\sum_{i,t}(y_{i2t} - p_{i2t})x_{it}, \sum_{i,t}(y_{i3t} - p_{i3t})x_{it}; \sum_{j,t}(y_{1jt} - p_{1jt})w_{jt}, \dots, \sum_{j,t}(y_{Ijt} - p_{Ijt})w_{jt}\right) \tag{35}$$

Then the matrix $\dfrac{\partial^2 \log l(\boldsymbol{\psi}_{it}(\boldsymbol{\theta}); \boldsymbol{y}_{it})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}'}$ can be written as:

$$\begin{bmatrix}
-p_{i2t}(1-p_{i2t})x_{it}^2 & p_{i2t}p_{i3t}x_{it}^2 & \cdots & \mathbf{0} & (p_{i2t}\overline{w}_{it} - p_{i2t}w_{2t})x_{it} & \mathbf{0} & \cdots & \mathbf{0} \\
p_{i3t}p_{i2t}x_{it}^2 & -p_{i3t}(1-p_{i3t})x_{it}^2 & \cdots & \mathbf{0} & (p_{i3t}\overline{w}_{it} - p_{i3t}w_{3t})x_{it} & \mathbf{0} & \cdots & \mathbf{0} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\
(p_{i2t}\overline{w}_{it} - p_{i2t}w_{2t})x_{it} & (p_{i3t}\overline{w}_{it} - p_{i3t}w_{3t})x_{it} & \cdots & \mathbf{0} & (\overline{w}_{it})^2 - \overline{w_{it}^2} & \mathbf{0} & \cdots & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0}
\end{bmatrix}$$

where

$$\overline{w}_{it} = \sum_j p_{ijt} w_{jt}$$
$$\overline{w_{it}^2} = \sum_j p_{ijt} w_{jt}^2$$

And then the Hessian matrix $\nabla^2 \mathcal{L}(\theta)$ is:

$$\frac{1}{I \times T} \sum_{i,t} \frac{\partial^2 \log l(\boldsymbol{\psi}_{it}; \boldsymbol{y}_{it})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}'} =$$

$$\frac{1}{I \times T} \left[\begin{array}{cc|ccccc}
\sum_{i,t} -p_{i2t}(1-p_{i2t})x_{it}^2 & \sum_{i,t} p_{i2t}p_{i3t}x_{it}^2 & \sum_t (p_{12t}\overline{w}_{it} - p_{12t}w_{2t})x_{1t} & \cdots & \sum_t (p_{i2t}\overline{w}_{it} - p_{i2t}w_{2t})x_{it} & \cdots & \sum_t (p_{I2t}\overline{w}_{It} - p_{I2t}w_{2t})x_{It} \\
\sum_{i,t} p_{i3t}p_{i2t}x_{it}^2 & \sum_{i,t} -p_{i3t}(1-p_{i3t})x_{it}^2 & \sum_t (p_{13t}\overline{w}_{it} - p_{13t}w_{3t})x_{1t} & \cdots & \sum_t (p_{i3t}\overline{w}_{it} - p_{i3t}w_{3t})x_{it} & \cdots & \sum_t (p_{I3t}\overline{w}_{It} - p_{I3t}w_{3t})x_{It} \\
\hline
\sum_t (p_{12t}\overline{w}_{it} - p_{12t}w_{2t})x_{1t} & \sum_t (p_{13t}\overline{w}_{it} - p_{13t}w_{3t})x_{1t} & \sum_t (\overline{w}_{1t})^2 - \overline{w_{1t}^2} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\
\sum_t (p_{i2t}\overline{w}_{it} - p_{i2t}w_{2t})x_{it} & \sum_t (p_{i3t}\overline{w}_{it} - p_{i3t}w_{3t})x_{it} & \mathbf{0} & \cdots & \sum_t (\overline{w}_{it})^2 - \overline{w_{it}^2} & \cdots & \mathbf{0} \\
\vdots & \vdots & \vdots & \cdots & \vdots & \ddots & \vdots \\
\sum_t (p_{I2t}\overline{w}_{It} - p_{I2t}w_{2t})x_{It} & \sum_t (p_{I3t}\overline{w}_{It} - p_{I3t}w_{3t})x_{It} & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \sum_t (\overline{w}_{It})^2 - \overline{w_{It}^2}
\end{array}\right] \tag{36}$$

We partition the Hessian into blocks according to the lines in (36). Note that the top left block is the same as in the previous simulation. So the blocks are:

$$A = \sum_{i,t} H_{\boldsymbol{\psi}_{it}} x_{it}^2 = \sum_i A_i$$

$$B = \begin{bmatrix}
\sum_t (p_{12t}\overline{w}_{it} - p_{12t}w_{2t})x_{1t} & \cdots & \sum_t (p_{12t}\overline{w}_{it} - p_{12t}w_{2t})x_{1t} & \cdots & \sum_t (p_{12t}\overline{w}_{it} - p_{12t}w_{2t})x_{1t} \\
\sum_t (p_{13t}\overline{w}_{it} - p_{13t}w_{2t})x_{1t} & \cdots & \sum_t (p_{13t}\overline{w}_{it} - p_{13t}w_{3t})x_{1t} & \cdots & \sum_t (p_{13t}\overline{w}_{it} - p_{13t}w_{3t})x_{1t}
\end{bmatrix}$$

$$= [B_1 \cdots B_i \cdots B_I]$$

$$C = B'$$

$$D = \begin{bmatrix} \sum_t (\overline{w}_{1t})^2 - \overline{w_{1t}^2} & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \sum_t (\overline{w}_{2t})^2 - \overline{w_{2t}^2} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \sum_t (\overline{w}_{it})^2 - \overline{w_{it}^2} & \cdots & \mathbf{0} \\ \vdots & \vdots & \cdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \sum_t (\overline{w}_{It})^2 - \overline{w_{It}^2} \end{bmatrix} \tag{37}$$

$$= \begin{bmatrix} d_1 & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & d_2 & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & d_i & \cdots & \mathbf{0} \\ \vdots & \vdots & \cdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \cdots & d_I \end{bmatrix}$$

Note that the matrix D is a diagonal matrix so inverting it is trivial, which makes block inversion an efficient strategy. We can apply the same formula as in (29).

C.1.2 Newton Step

The Newton step is:[16]

$$\Delta\boldsymbol{\theta} = \nabla^2\mathcal{L}(\boldsymbol{\theta})^{-1}\nabla\mathcal{L} \tag{38}$$

$$= \begin{bmatrix} (A - BD^{-1}C)^{-1} & -(A - BD^{-1}C)^{-1}BD^{-1} \\ -D^{-1}C(A - BD^{-1}C)^{-1} & -D^{-1} + D^{-1}C(A - BD^{-1}C)^{-1}BD^{-1} \end{bmatrix} \begin{bmatrix} \frac{\partial\mathcal{L}}{\partial\boldsymbol{\beta}} \\ \frac{\partial\mathcal{L}}{\partial\gamma_i} \\ \vdots \\ \frac{\partial\mathcal{L}}{\partial\gamma_I} \end{bmatrix}$$

From Hospido (2012), the Newton step simplifies to:

$$\Delta\boldsymbol{\beta} = \left(\sum_i A_i - \frac{1}{d_i}B_iC_i\right)^{-1}\left(\frac{\partial\mathcal{L}}{\partial\boldsymbol{\beta}} - \sum_i \frac{1}{d_i}B_i\frac{\partial\mathcal{L}}{\partial\gamma_i}\right) \tag{39}$$

and

$$\Delta\gamma_i = \frac{1}{d_i}\left(\frac{\partial\mathcal{L}}{\partial\gamma_i} + B_i\Delta\boldsymbol{\beta}\right) \tag{40}$$

[16] As in the previous simulation, we set $\sigma = 1$.

As in the previous simulation, we first compute the individual components and then find $\Delta\beta$ which we use to calculate each $\Delta\gamma_i$.

References

Angrist, J. D. and J.-S. Pischke (2010). "The Credibility Revolution in Empirical Economics: How Better Research Design Is Taking the Con out of Econometrics". In: *Journal of Economic Perspectives* 24, pp. 3–30.

Böhning, D. (1992). "Multinomial Logistic Regression Algorithm". In: *Annals of the Institute of Statistical Mathematics* 44, pp. 197–200.

Böhning, D. and B. G. Lindsay (1988). "Monotonicity of Quadratic-Approximation Algorithms". In: *Annals of the Institute of Statical Mathematics* 40, pp. 641–663.

Capps, C., L. Kmitch, Z. Zabinski, and S. Zayats (2018). "The Continuing Saga of Hospital Merger Enforcement". In: *Antitrust LJ* 82, p. 441.

Chamberlain, G. (1980). "Analysis of Covariance with Qualitative Data". In: *Review of Economic Studies* 47, pp. 225–238.

Chen, M. (2019). *Estimation of Nonlinear Panel Models with Multiple Unobserved Effects*. Unpublished manuscript, University of Warwick.

Correia, S. (2017). "Linear models with high-dimensional fixed effects: An efficient and feasible estimator". In: *Working Paper*.

D'Haultfœuille, X. and A. Iaria (2016). "A Convenient Method for the Estimation of the Multinomial Logit Model with Fixed Effects". In: *Economics Letters* 141, pp. 77–70.

Dhaene, G. and K. Jochmans (2015). "Split-panel Jackknife Estimation of Fixed-effect Models". In: *Review of Economic Studies* 82, pp. 991–1030.

Du, T., A. Kanodia, and S. Athey (2023). "Torch-Choice: A PyTorch Package for Large-Scale Choice Modelling with Python". In: *arXiv preprint arXiv:2304.01906*.

Du, Y. and R. Varadhan (2020). "SQUAREM: An R Package for Off-the-Shelf Acceleration of EM, MM and Other EM-Like Monotone Algorithms". In: *Journal of Statistical Software* 92.7, pp. 1–41. DOI: 10.18637/jss.v092.i07. URL: https://www.jstatsoft.org/index.php/jss/article/view/v092i07.

Fong, D. C.-L. and M. Saunders (2011). "LSMR: An iterative algorithm for sparse least-squares problems". In: *SIAM Journal on Scientific Computing* 33.5, pp. 2950–2971.

Garmon, C. (2017). "The accuracy of hospital merger screening methods". In: *The RAND Journal of Economics* 48.4, pp. 1068–1102.

Gaure, S. (2013). "OLS with multiple high dimensional category variables". In: *Computational Statistics & Data Analysis* 66, pp. 8–18.

Gaynor, M. S., S. A. Kleiner, and W. B. Vogt (2013). "A structural approach to market definition with an application to the hospital industry". In: *The Journal of Industrial Economics* 61.2, pp. 243–289.

Gowrisankaran, G., A. Nevo, and R. Town (2015). "Mergers When Prices Are Negotiated: Evidence from the Hospital Industry." In: *American Economic Review* 105, pp. 172–203. ISSN: 00028282. URL: http://search.ebscohost.com.ezproxy.bu.edu/login.aspx?direct=true&db=ecn&AN=1467554&site=ehost-live&scope=site.

Greene, W. H. (2018). *Econometric Analysis*. 8th. Prentice Hall.

Guimaraes, P., O. Figueirdo, and D. Woodward (2003). "A Tractable Approach to the Firm Location Decision Problem". In: *The Review of Economics and Statistics*, pp. 201–204.

Guimaraes, P. and P. Portugal (2010). "A simple feasible procedure to fit models with high-dimensional fixed effects". In: *The Stata Journal* 10.4, pp. 628–649.

Hahn, J. and W. Newey (2004). "Jackknife and Analytical Bias Reduction for Nonlinear Panel Models". In: *Econometrica* 72, pp. 1295–1319.

Hinz, J., A. Hudle, and J. Wanner (2019). *Separating the Wheat from the Chaff: Fast Estimation of GLMs with High-Dimensional Fixed Effects*. Unpublished manuscript, European University Institute.

Ho, K. and A. Pakes (2014). "Hospital Choices, Hospital Prices, and Financial Incentives to Physicians". In: *American Economic Review* 104, pp. 3841–3884.

Ho, K. (2006). "The welfare effects of restricted hospital choice in the US medical care market". In: *Journal of Applied Econometrics* 21.7, pp. 1039–1079. DOI: https://doi.org/10.1002/jae.896. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/jae.896.

Hospido, L. (2012). "Estimating nonlinear models with multiple fixed effects: A computational note". In: *Oxford Bulletin of Economics and Statistics* 74.5, pp. 760–775.

Hunter, D. R. and K. Lange (2004). "A Tutorial on MM Algorithms". In: *The American Statistician* 58, pp. 30–37.

James, J. (2017). "MM Algorithm for General Mixed Multinomial Logit Models". In: *Journal of Applied Econometrics* 32, pp. 841–857.

Kaczmarz, S. (1993). "Approximate solution of systems of linear equations". In: *International Journal of Control* 57.6, pp. 1269–1271. DOI: 10.1080/00207179308934446. eprint: https://doi.org/10.1080/00207179308934446. URL: https://doi.org/10.1080/00207179308934446.

Keskar, N. S., D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang (2017). *On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima*. arXiv: 1609.04836 [cs.LG].

Kingma, D. P. and J. Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

Lange, K. (2013). *Optimization*. Second Edition. New York: Springer.

Lange, K. (2016). *MM Optimization Algorithms*. SIAM.

Lange, K., D. R. Hunter, and I. Yang (2000). "Optimization transfer using surrogate objective functions". In: *Journal of Computational and Graphical Statistics* 9, pp. 1–20.

Leeuw, J. de and K. Lange (2009). "Sharp Quadratic Majorization in One Dimension". In: *Computational Statistics & Data Analysis* 53, pp. 2471–2484.

McFadden, D. and K. Train (2000). "Mixed MNL models for discrete response". In: *Journal of Applied Econometrics* 15, pp. 447–470.

Mugnier, M. and A. Wang (2024). *Fixed Effects Nonlinear Panel Models with Heterogeneous Slopes: Identification and Consistency*. SSRN Paper No. 4186349.

Neyman, J. and E. L. Scott (1948). "Consistent estimates based on partially consistent observations". In: *Econometrica: journal of the Econometric Society*, pp. 1–32.

Nocedal, J. and S. J. Wright (2006). *Numerical optimization*. Spinger.

Raval, D., T. Rosenbaum, and S. A. Tenn (2017). "A semiparametric discrete choice model: An application to hospital mergers". In: *Economic Inquiry* 55.4, pp. 1919–1944.

Raval, D., T. Rosenbaum, and N. E. Wilson (2022). "Using disaster-induced closures to evaluate discrete choice models of hospital demand". In: *The RAND Journal of Economics* 53.3, pp. 561–589.

Reddi, S. J., S. Kale, and S. Kumar (2019). "On the convergence of Adam and beyond". In: *arXiv preprint arXiv:1904.09237*.

Ruud, P. A. (1991). "Extensions of Estimation Methods Using the EM Algorithm". In: *Journal of Econometrics* 49, pp. 305–341.

Rysman, M., S. Wang, and K. Wozniak (2025). *Empirics of Payment Method Choice in Scanner Data*. Unpublished manuscript, Boston University.

Sahai, H. (2023). "The Welfare Implications of School Voucher Design: Evidence from India". In.

Shum, M., W. Song, and X. Shi (2018). "Estimating Semi-parametric Panel Multinomial Choice Models using Cyclic Monotonicity". In: *Econometrica* 86, pp. 737–761.

Somaini, P. and F. A. Wolak (2016). "An algorithm to estimate the two-way fixed effects model". In: *Journal of Econometric Methods* 5.1, pp. 143–152.

Stammann, A. (2018). *Fast and Feasible Estimation of Generalized Linear Models with High-Dimensional k-way Fixed Effects*. Unpublished manuscript, arXiv:1707.01815.

Stammann, A., F. Heiß, and D. McFadden (2016). *Estimating Fixed Effects Logit Models with Large Panel Data*. Beiträge zur Jahrestagung des Vereins für Socialpolitik 2016: Demographischer Wandel - Session: Microeconometrics, No. G01-V3. URL: `http://hdl.handle.net/10419/145837`.

Tanner, M. (1996). *Tools for Statistical Inference: Methods for the Exploration of Posterior Distributions and Likelihood Functions*. Springer.

Tenn, S. and S. Vandergrift (2017). "Geographic Market Definition in Urban Hospital Mergers: Lessons from the Advocate-NorthShore Litigation". In: *Antitrust Source* 17.3, pp. 1–10.

Varadhan, R. and C. Roland (2008). "Simple and globally convergent methods for accelerating the convergence of any EM algorithm". In: *Scandinavian Journal of Statistics* 35.2, pp. 335–353.

Zhang, Y., C. Chen, N. Shi, R. Sun, and Z.-Q. Luo (2022). "Adam can converge without any modification on update rules". In: *Advances in Neural Information Processing Systems* 35, pp. 28386–28399.

Zhou, H., D. Alexander, and K. Lange (2011). "A quasi-Newton acceleration for high-dimensional optimization algorithms". In: *Statistics and Computing* 21, pp. 261–273.