MATLAB implementation of the Constrained Zero Inflated Joint Modeling (CEZIJ) framework of Banerjee et al. (2019).

March 18, 2019

1 Introduction

CEZIJ (Banerjee et al., 2019) is a novel framework for parameter estimation in joint models with multiple longitudinal outcomes along with a time-to-event analysis. Longitudinal data from modern datasets usually exhibit a large set of potential predictors and choosing the relevant set of predictors is highly desirable for various purposes including improved predictability. To achieve this goal, CEZIJ conducts simultaneous selection of fixed and random effects in high-dimensional penalized generalized linear mixed models and maintains the hierarchical congruity of the fixed and random effects, thus producing models with interpretable composite effects. It not only accommodates extreme zero-inflation in the responses in a joint model setting but also incorporates domain-specific, convex structural constraints on the model parameters. For analyzing such large-scale datasets, variable selection and estimation is conducted via a distributed computing based split-and-conquer approach (Chen and Xie, 2014) that massively increases scalability.

2 Installation requirements

The GitHub repository holds the MATLAB toolbox `cezij.mltbx` that provides an implementation of the CEZIJ procedure developed in Banerjee et al. (2019). To install this toolbox, simply download `cezij.mltbx` in your computer and double click to install it. For a successful installation, please make sure that the following system requirements are met.

- Access to 32GB RAM and at least 8 CPU cores for parallel computing
- MATLAB 2016b or higher with the following toolboxes (and their dependencies):
  - Statistics toolbox
  - Optimization toolbox
  - Parallel Computing toolbox
  - Data Acquisition toolbox
- CVX for MATLAB (version 2.1 or higher)

3 A numerical example

In this section, we will use a numerical example to illustrate the use of the `cezij` toolbox. Our goal is to assess the model selection performance of CEZIJ under the longitudinal and Dropout models discussed in Section 3.1 of Banerjee et al. (2019). To do that, we use the simulation example of setting I discussed in section C.2 of the supplementary materials and indicate which files in the toolbox should be edited to test a different dataset. Figure 1 presents three MATLAB scripts that should be edited to prepare the cezij toolbox for analyses. In what follows, we discuss these scripts.

3.1 Generating simulated data - `simulate_data.m`

To run the `cezij` variable selection algorithm on a dataset of your own choice, please edit the m file `simulate_data.m` and ensure that your data is in the same format as the output of this m file. In the default setting, `simulate_data.m` generates a simulated dataset that we describe below.

Consider a sample of $n = 500$ players and for each player i, let $\mathbb{X}_i = (\boldsymbol{X}_{i.1}, \ldots, \boldsymbol{X}_{i.p})$ denote the $m \times p$ matrix of candidate predictors where $\boldsymbol{X}_{i.k} = (x_{i1k}, \ldots, x_{imk})^T$, $m = 30$

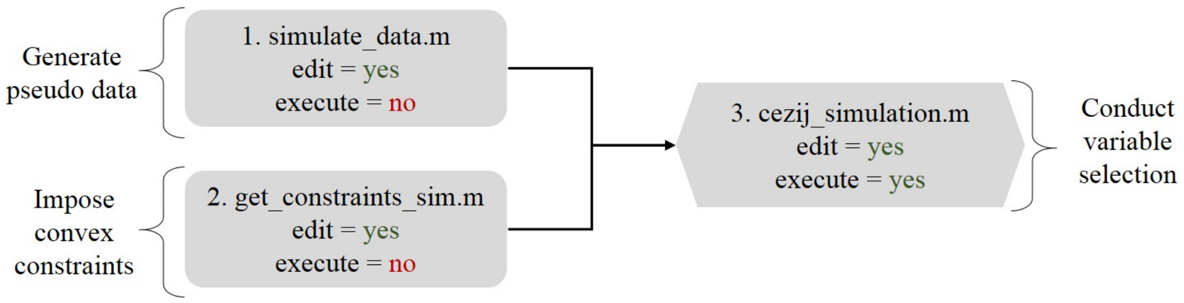

Figure 1: There are three MATLAB scripts, numbered 1-3, that must be edited to prepare the `cezij` toolbox for analyses. To reproduce table 4 in the supplementary file or table 1, script 3 must be executed without making any changes to the default parameters in scripts 1-3.

denotes the number of time points for which we observe each player and $p = 10$ is number of candidate predictors. We take $\mathcal{I}_f = \{1, \ldots, 8\}$ as the indices of the fixed effects and, $\mathcal{I}_c = \{9, 10\}$ as the indices of the composite effects so that $p_f = |\mathcal{I}_f| = 8$, $p_c = |\mathcal{I}_c| = 2$. Thus, the first 8 columns of $\mathbb{X}_i$ represent fixed effects while the last 2 represent composite effects. The five responses $[\alpha_i, \mathbb{A}_i, \epsilon_i, \mathbb{E}_i, \mathbb{D}_i]$ corresponding to `AI`, positive Activity, `EI`, Positive Engagement and Dropout are generated from the following models: $\mathsf{logit}(\pi_{ij}) = \beta_0^{(1)} + x_{ij1}\beta_1^{(1)} + b_{i1}$, $\mu_{ij} = \beta_0^{(2)} + x_{ij2}\beta_1^{(2)} + b_{i2}$ with $\sigma_1 = 0.5$, $\mathsf{logit}(q_{ij}) = \beta_0^{(3)} + x_{ij3}\beta_1^{(3)} + b_{i3}$, $\gamma_{ij} = \beta_0^{(4)} + x_{ij4}\beta_1^{(4)} + b_{i4}$ with $\sigma_2 = 0.5$ and $\text{logit}(\lambda_{ij}) = \beta_0^{(5)} + x_{ij5}\beta_1^{(5)} + \eta_1 b_{i1} + \ldots + \eta_4 b_{i4}$ where the true values of the fixed effect coefficients are: $\boldsymbol{\beta}^{(1)} = (1, -1.5)$, $\boldsymbol{\beta}^{(2)} = (3.5, -2)$, $\boldsymbol{\beta}^{(3)} = (1, -1)$, $\boldsymbol{\beta}^{(4)} = (3, -3)$, $\boldsymbol{\beta}^{(5)} = (-1, 2)$ and, $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_4) = (-0.1, 0.2, 0.1, -0.2)$. Thus this setting presents a scenario wherein there are no composite effects in the true model. The random effects $\boldsymbol{b}_i = (b_{i1}, \ldots, b_{i4})$ are sampled from $N_4(\mathbf{0}, \boldsymbol{\Sigma})$, independently for each i, where

$$\boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0.2 & 0.4 & 0.5 \\ 0.2 & 3 & 0.9 & 0.7 \\ 0.4 & 0.9 & 0.8 & 0.5 \\ 0.5 & 0.7 & 0.5 & 4 \end{pmatrix}$$

Finally, to complete the specification, we sample $(x_{ij1}, \ldots, x_{ij4})$ from $N_4(\mathbf{0}, 4\boldsymbol{I}_4)$, independently for each $i = 1, \ldots, n$, $j = 1, \ldots, m$. To ensure that the generated sample contains players that have not churned for at least the first 7 to 10 days, we let $\boldsymbol{X}_{i.5}$ to be an m dimensional ordered sample from `Unif`$(-1,1)$ so that $\boldsymbol{X}_{i.5} = (x_{i15} \leq \cdots \leq x_{im5})$ and, generate the remaining predictors independently from `Unif`$(-1,1)$. Although the MATLAB file `simulate_data.m` stores the above simulation setting, it can easily be modified to test different settings.

3.2 Imposing convex constraints - `get_constraints_sim.m`

The CEZIJ framework can incorporate convexity constraints on the fixed effect coefficients and this MATLAB file stores the following default constrains:

$$\beta_0^{(1)} > 0,\ \beta_1^{(1)} < 0;\ \beta_1^{(3)} < 0;\ \beta_0^{(4)} > 0,\ \beta_1^{(4)} < 0.$$

Please modify this file to enforce constraints specific to your application or leave this file unchanged to reproduce table 4 of the supplementary materials or table 1 in `cezij help.pdf`.

3.3 Running the joint model - `cezij_simulation.m`

Recall that CEZIJ uses the split-and-conquer approach of Chen and Xie (2014) to split the full set of n players into K non-overlapping groups and conducts variable selection separately in each group by solving K parallel maximization problems represented by equation (9) of Banerjee et al. (2019). The selected fixed and random effects are then determined using a majority voting scheme across all the K groups as described in Section 5 of the above paper.

In the MATLAB file `cezij_simulation.m`, lines `15-27` can be used to specify a number of user defined parameters. For this example, we fix $K = 5$ so that n/K is 100 while $q = 3$ indicates the number of random effects including a random intercept. We generate `nsets` $=$ 25 datasets and run `cezij_simulation.m` to assess the model selection performance in terms of the average False Negatives (in-model predictors falsely identified as being out of model) and average False Positives (out of model predictors falsely identified as being in-model) for the fixed effects (composite or not) and the random effects. To evaluate

Table 1: $(n = 500, m = 30, p = 10, K = 5)$ - average False Negatives(FN), average False Positives (FP) for fixed (composite or not) and random effects and, % datasets with non-hierarchical selection.

	Fixed Effects		Random Effects		
Model	FN	FP	FN	FP	% Non Hier. Selec.
AI	0	2.76	0.16	0.36	0
Activity Time	0	1.44	0	0.04	0
EI	0	4.48	0	1.12	0
Engage. Time	0	1.52	0	0.04	0
Dropout	0	1.52	-	-	-

the hierarchical selection property of our framework, the code also reports the percentage of datasets where `cezij` conducted non-hierarchical selection and chose predictors with random effects only.

Running `cezij_simulation.m` with the default parameters generates Table 1 that reports the results of the simulation exercise under setting I that is discussed in section C.2 of the supplementary materials. We see that CEZIJ selects the correct in-model predictors for the five models. The relatively higher fixed effects False positives for the `AI` and `EI` models possibly indicate some over-fitting due to the prevalence of large number of zeros in these models. However, CEZIJ selects the fixed and random effects in a hierarchical fashion such that no random effect predictor appears in any of the four models without their fixed effect counterparts. This is not surprising given the way CEZIJ updates the adaptive weights $(c_{sr}^{(t)}, d_{sr}^{(t)})$ are after each iteration.

4 Simulation flow

In figure 2, we present a simulation flow diagram that depicts the main scripts that are called when `cezij_simulation.m` is executed. The scripts highlighted in blue are editable and can be used to run the analyses on a different data set as described in section 3. The

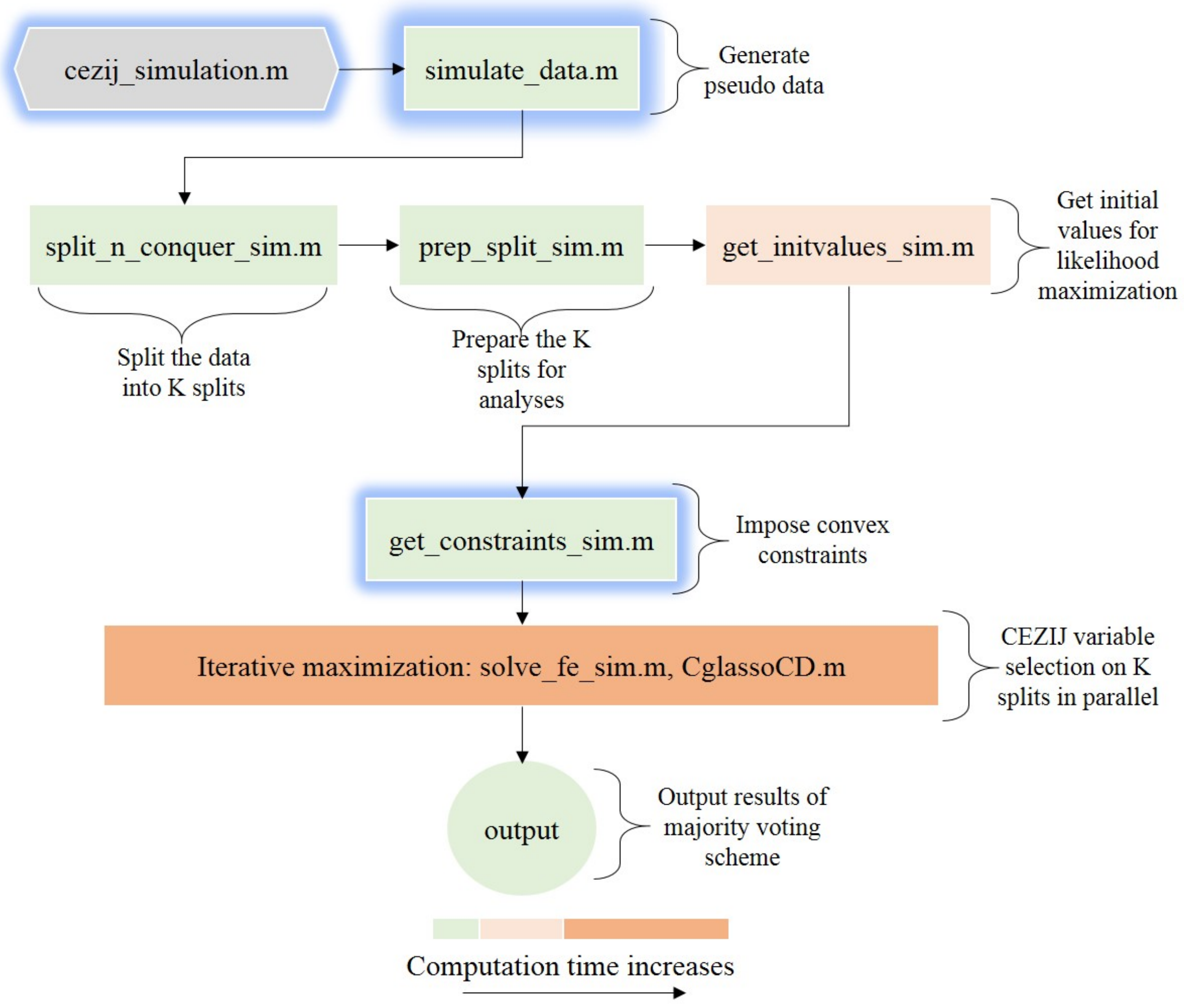
cezij_simulation.m
simulate_data.m
Generate pseudo data
split_n_conquer_sim.m
prep_split_sim.m
get_initvalues_sim.m
Get initial values for likelihood maximization
Split the data into K splits
Prepare the K splits for analyses
get_constraints_sim.m
Impose convex constraints
Iterative maximization: solve_fe_sim.m, CglassoCD.m
CEZIJ variable selection on K splits in parallel
output
Output results of majority voting scheme
Computation time increases

Figure 2: Simulation flow diagram that depicts the main scripts that are called when `cezij_simulation.m` is executed. The scripts highlighted in blue are editable while the color coding of the scripts indicate their relative contribution to total computation time.

color coding of the scripts indicate their relative contribution to total computation time.
For instance, the iterative maximization step that is executed in parallel across the K splits is the most computationally intensive step of the `cezij` algorithm and, as discussed in section C.2 of the supplementary material, relies on the specific system configuration and the number of computation cores available. In the default setting that is used to reproduce table 1, this step takes approximately 5 minutes to execute (see figure 7 of the supplementary materials). Depending on the number of splits K, availability of additional

computational cores may further reduce the overall computation time.

References

Banerjee, T., Mukherjee, G., Dutta, S., and Ghosh, P. (2019). A large-scale constrained joint modeling approach for predicting user activity, engagement and churn with application to freemium mobile games. *under review*.

Chen, X. and Xie, M.-g. (2014). A split-and-conquer approach for analysis of extraordinarily large data. *Statistica Sinica*, pages 1655–1684.