Code Tutorial for the Social Identity Voting model

Scott Pauls, Greg Leibon, and Daniel Rockmore

Contents of the zip file

The zip file contains a directory structure and the following files:

```
clean_data.m
read_data.m
make_holdout.m
SIV_layer_one.m
SIV_layer_two.m
wnominate_script.R
wnom_out_of_sample_acc.m

demo_script.m

convert_votes.m
partition_representation.m
```

The files in the first group are described in detail below. A sample script is included (`demo_script.m`) is included to demonstrate the construction of the SIV for the 113th House of Representatives. The last two MATLAB files are commands used by more than one of the commands in the first group.

Tutorial

In this tutorial, we describe in detail the process for constructing the SIV model. We also reference the MATLAB code and scripts which execute each of the steps. All of these steps are put together in the `demo_script.m` MATLAB file. We assume there are three subdirectories – data, holdout samples, and results.

We also assume that you have downloaded ORD files from Voteview.com for the sessions you wish to analyze. For the purposes of the demonstration script, we further assume that data for the 113th House of Representatives is available in a file names H113.ORD in the data subdirectory.

1. *Read the data*

Data files from voteview are in special format. We read the data into MATLAB using the *read_data.m* which parses the format and returns matrices of the encoded data.

Syntax:

```
[congressnum,icpsr,statecode,district,statename,partycode,meansoffice,name,votes]=read_data(filename)
```

This returns nine objects: vectors containing the congress numbers for each legislator, their ICPSR codes, their state codes, districts, state names, their party codes (as listed on the voteview website), codes describing how they entered office, their names, and a matrix of all the votes cast.

2. *Clean the data*

As discussed in the methods, we remove votes with less than 2.5% of the body in the minority and legislators who vote on less than two thirds of the votes. The function *clean_data.m* achieves this.

Syntax: `[votesm,missed,lopsided_votes]=clean_data(votes)`

The command returns, in order, the cleaned votes matrix, a list of the legislators who missed too many votes, and a list of the lopsided votes.

3. Create a MATLAB data file with the cleaned data

For ease of use, at this point we save the initial and cleaned data in a MATLAB data file named either `house-xxx.mat` or `senate-xxx.mat` in the data subdirectory.

4. Create holdout data sets

We create 10 modified data sets where we remove 10% of the combined yea/nay votes at random and treat them as missing data. If V is our matrix of cleaned votes, we denote by $\{V^1, \ldots, V^{10}\}$ the ten holdout data sets. To create these in MATLAB, we use the function *make_holdout.m.*

Syntax: `make_holdout(congress_number,1:10,flag)`

Executing this command will create the holdouts for the specified Congress for either the House of Representatives (flag = 1) or the Senate (flag =0). The function saves the files in MATLAB format as well as CSV. The latter may be used by the R implementation of W-NOMINATE.

For each holdout sample, we perform steps 5 and 6.

5. Construct the first layer of SIV

For a given legislature, we form two groups, the indices of members who caucus with the Democrats, $\mathcal{D}$, and the indices of those who caucus with the Republicans, $\mathcal{R}$. We compute the motivations from the vote matrix, V^i:

$$M_1 = (v_1^1, \dots, v_n^1)$$

where $v_j^1 = \frac{1}{|\mathcal{D}|}\sum_{l\in\mathcal{D}} V_{lj}^i$ and

$$M_2 = (v_1^2, \dots, v_n^2)$$

where $v_j^2 = \frac{1}{|\mathcal{R}|}\sum_{l\in\mathcal{R}} V_{lj}^i$. The one layer representation for legislator l is then given as

$$L_1^l = sign(w_1(l)M_1 + w_2(l)M_2).$$

Using equation (x), we computed the weights for each legislator using least squares and form the residual data,

$$(\hat{v}_1(l), \dots, \hat{v}_n(l)) = (v_1(l), \dots, v_n(l)) - \sum_i w_i(l)M_i,$$

for legislator l.

The MATLAB function *SIV_layer_one.m* executes these computations.

Syntax:

```
[layer_one,R,motivations,oos_lay1]=SIV_layer_one(num,flag)
```

The command takes two inputs, the Congress number and a flag indicating House (1) or Senate (0). It returns the first layer approximation of the votes (the $\hat{v}$), the residual data, a matrix of the motivation vectors for the two parties, and the out-of-sample accuracy for the first layer of the SIV.

6. Construct the second layer of SIV

To form the second layer of SIV, we first construct a matrix of correlations between the residual voting profiles of each pair of legislators:

$$c_{ij} = corr\big((\hat{v}_1(i), \dots, \hat{v}_n(i)), (\hat{v}_1(j), \dots, \hat{v}_n(j))\big).$$

We then convert the correlations to spherical distances using the formula $d_{ij} = \sin(\cos^{-1} c_{ij})$ and then rescale the distances, defining $a_{ij} = e^{-d_{ij}^2}$. The rescaling provides values in the matrix $A = (a_{ij})$ which reflect adjacency – values close to one when distances are small and values close to zero when distances are large.

We then perform spectral clustering, first constructing the normalized graph Laplacian:

$$\mathcal{L} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

where I is the identity matrix, and D is the diagonal matrix with $D_{ii} = \sum_j A_{ji}$. We then find the eigendata for $\mathcal{L}$, (λ_i, μ_i) with $0 = \lambda_0 < \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_m$. The first parameter in spectral clustering is the parameter r which determines how many eigenvectors associated to nonzero eigenvalues we use in the clustering step. Once choosing the value for r, we form the spectral embedding:

$$emb: \{1, \dots, m\} \to \mathbb{R}^k$$

$$i \to \left(\mu_1(i), \ldots, \mu_k(i)\right).$$

Last, we perform k-means on the points of this embedding to form k clusters of the nodes of the network.

To aid in estimating the corect parameters, we repeat this algorithm for parameter values $r, k \in \{1, \ldots, 10\}$ where k is the number of clusters and r is the number of eigenvectors of the graph Laplacian to use for k-means. In each case, we form the motivation vector for each group via Equation (2) using the residual data, which we denote by $\widehat{M}_i = \left(\hat{v}_1^i, \ldots, \hat{v}_n^i\right)$, and find the weights, $\widehat{w}_i(l)$, using Equation (x). Then, the two layer representation approximation of the voting matrix is given as,

$$L_2^l = sign(w_1(l)M_1 + w_2(l)M_2 + \sum_j \hat{v}_j(l)\widehat{M}_j\ ).$$

The MATLAB command *SIV_layer_two.m* performs these steps.

Syntax: `[layer_two,motivations,oos_lay2,r,k]=SIV_layer_two(num,flag)`

The two arguments are the same as in the previous step. The outputs are the layer two approximation of the vote matrix, the matrix of motivation vectors for the groups identified by the algorithm, the out-of-sample accuracy of the two layer SIV model, and the two parameters $((r, k))$ from spectral clustering for the model that achieves the best mean out-of-sample accuracy.

a. *Out-of-sample accuracy for each of the holdout data samples*

In the previous two steps, out-of-sample accuracy is computed for the two models. We briefly discuss how this computation is completed. For each of the holdout samples, we have constructed the one and two layer models for each pair of parameters (r, k). We can then measure the out of sample accuracy in each case,

$$oos_1(j) = \frac{1}{|\mathcal{H}|} \sum_{h \in \mathcal{H}} \sum_{l=1}^{m} \psi(L_1^l(v) - V_{lh}^j)$$

$$oos_2(j, r, k) = \frac{1}{|\mathcal{H}|} \sum_{h \in \mathcal{H}} \sum_{l=1}^{m} \phi(L_2^l(v) - V_{lh}^j)$$

where $\mathcal{H}$ is the list of held out votes and ψ is the function which is one at zero and zero everywhere else. The mean out-of-sample accuracies are given by

$$moos_1 = mean(oos_1), \text{ and } moos_2(r, k) = \frac{1}{10} \sum_{j=1}^{10} oos_2(j, r, k).$$

The MATLAB function *oos_accuracy.m* performs this computation.

b. *Model Selection*

To determine our choice of r and k for the two layer model, we find the values of the parameters where $moos_2$ reaches its maximum. If there are multiple pairs, we select the one with the smallest k and then, if necessary, the smallest r. To choose between the one layer model and the selected two layer model, we simply choose the model with the highest mean out of sample accuracy.

7. W-NOMINATE

In the paper, we compare the results of SIV to those of W-NOMINATE. Using the saved holdout files (in CSV format), the R implementation of W-NOMINATE easily generates ideal point estimates for the votes and voters (the R software is well documented and we will not provide additional discussion here). We provide an R script, `wnominate_script.R`, which processes the holdout samples from step 4. The script writes two files for each holdout sample. As written, the script uses data from the 113th House, and the files are named `house-113-x-leg-nd.csv` and `house-113-x-rollcall-nd.csv` where *x* is the holdout number and *n* is the dimension of the W-NOMINATE model. These files are placed in the directory `results_nd` where, again, *n* is the dimension of the W-NOMINATE model.

8. Measuring out-of-sample accuracy for W-NOMINATE models

Next, we compute the out-of-sample accuracy for the values estimated by the W-NOMINATE models.

Syntax:

```
wnom_oos=wnom_out_of_sample_acc(session,dimension,sample,flag)
```

The arguments are *session* number of the Congress, the *dimension* of the W-NOMINATE model, a vector of *sample* indices, and a *flag* indicating the House (1) or Senate (0). The output *wnom_oos* is the mean out-of-sample accuracy over all samples.

For example, to calculate the out-of-sample accuracy for the one dimensional W-NOMINATE model for the 113th House, we use:

```
wnom_oos=wnom_out_of_sample_acc(113,1,1:10,1).
```