Using Poole's Optimal Classification in R

September 19, 2024

1 Introduction

This package estimates Poole's Optimal Classification scores from roll call votes supplied though a `rollcall` object from package `pscl`.[1] Optimal Classification fits a Euclidean spatial model that places legislators in a specified number of dimensions (usually one or two). It maximizes the correct classification of legislative choices, whereas W-NOMINATE maximizes the probabilities of legislative choices given its error framework. It also differs from W-NOMINATE because it is a non-parametric procedure that requires no assumptions about the parametric form of the legislators' preference functions, other than assuming that they are symmetric and single–peaked. However, legislator coordinates recovered using OC are virtually identical to those recovered by parametric procedures.

The R version of Optimal Classification improves upon the earlier software in three ways. First, it is now considerably easier to input new data for estimation, as the current software no longer relies exclusively on the old *ORD* file format for data input. Secondly, roll call data can now be formatted and subsetted more easily using R's data manipulation capabilities. Finally, the `oc` package includes a full suite of graphics functions to analyze the results.

This section briefly outlines the method by which OC scores are calculated. For a full description, readers are referred to chapters 1 through 3 of Keith Poole's *Spatial Models of Parliamentary Voting.* and Poole's article in *Political Analysis* [1] [2].

[1]Production of this package is supported by NSF Grant SES-0611974.

We begin this discussion by considering how OC works in one dimension. The one-dimensional optimal classification method can be summarized as follows:

1. Generate a starting estimate of the legislator rank ordering using singular value decomposition.
2. Holding the legislator rank ordering fixed, use the *Janice* algorithm (described below) to find the optimal cutting point ordering.
3. Holding the cutting point ordering fixed, use the *Janice* algorithm to find the optimal legislator ordering.
4. Return to step 2 to iterate if the results change from the previous iteration

Together, steps 2-4 constitute what is know as the *Edith* algorithm.

To understand the Janice algorithm, we provide a useful example. Suppose we have six legislators who vote on 5 roll calls as follows:

Legislators	1	2	3	4	5
One	Y	Y	N	Y	Y
Two	N	Y	Y	Y	Y
Three	N	N	Y	Y	Y
Four	N	N	N	Y	Y
Five	N	N	N	N	Y
Six	N	N	N	N	N

To recover the legislator ideal points $X_1 ... X_6$ with voting error, we generate an agreement score matrix and extract the first eigenvector from the double–centered agreement score matrix (not shown) as follows. The result here transposes the ideal points of legislators X_1 and X_2, demonstrating how *Janice* orders the legislator ideal points.

Agreement Scores					
1					
.6	1				
.4	.8	1			
.6	.6	.8	1		
.4	.4	.6	.8	1.0	
.2	.2	.4	.6	.8	1.0

First Eigenvector
$X_1 = -.44512$
$X_2 = -.48973$
$X_3 = -.11093$
$X_4 = .04431$
$X_5 = .34859$
$X_6 = .65288$

To see how *Janice* chooses the cutting lines to minimize the number of classification errors, we first observe that the ideal points of the legislators are ordered such that $X_2 < X_1 < X_3 < X_4 < X_5 < X_6$. From this ordering, we take the predicted votes conditional on the jth cutline z_j as follows, and simply measure the actual number of errors on the roll calls. In this example, we show only a table with Yeas on the Left and Nays on the Right; however, this procedure is usually also repeated with a table where Yeas are on the Right and Nays are on the Left.

Cutline placement	Predicted Vote	Errors on Roll calls
		1 2 3 4 5
$Z_j < X_2$	N N N N N N	1 2 2 4 5
$X_2 < Z_j < X_1$	Y N N N N N	2 1 1 3 4
$X_1 < Z_j < X_3$	Y Y N N N N	**1** **0** 2 2 3
$X_3 < Z_j < X_4$	Y Y Y N N N	2 1 **1** 1 2
$X_4 < Z_j < X_5$	Y Y Y Y N N	3 2 2 **0** 1
$X_5 < Z_j < X_6$	Y Y Y Y Y N	4 3 3 1 **0**
$X_6 < Z_j$	Y Y Y Y Y Y	5 4 4 2 1

The underlined placements of the five roll call cutlines thus minimize the number of classification errors in this example, and the application of the *Janice* algorithm produces the following joint ordering of legislators and cutting points:

$$X_1 < X_2 < Z_1 = Z_2 < X_3 < Z_3 < X_4 < Z_4 < X_5 < Z_5 < X_6$$

In multiple dimensions, OC shares the same core algorithm as its single-dimension counterpart. The major difference in multiple dimensions is that cutting lines can no longer be tested exhaustively as in the one-dimensional case. Instead, the optimal cutting lines N_j for a roll call matrix with p legislators, s roll calls, and d dimensions are derived through projection onto a least squares line as follows:

1. Obtain a starting estimate of N_j, usually through least squares regression.

2. Calculate the correct classifications associated with N_j.

3. Construct Ψ^*, where:

   - $\Psi_i = X_i + (c_j - w_i)N_j$ if correctly classified and $\Psi_i = X_i$ if incorrectly classified. Ψ_i is the $sx1$ vector that is the ith row of Ψ, X_i is the ideal point of legislator i, c_j is the midpoint of roll call j, and $w_i = X_i' N_j$.
   - Ψ^* = Ψ - $J_p u'$, where J_p is a $px1$ vector of 1s and u is a $dx1$ vector of means.

4. Perform the singular value decomposition $U\Lambda V'$ of Ψ^*.

5. Use the sth singular vector (sth column) v_s of V as the new estimate of N_j.

The starting value generator, cutting line algorithm, and legislator ordering algorithm collectively consitute the OC algorithm that is implemented in this R package.

2 Usage Overview

The `oc` package was designed for use in one of three ways. First, users can estimate ideal points from a set of Congressional roll call votes stored in the traditional *ORD* file format. Secondly, users can generate a vote matrix of their own, and feed it directly into `oc` for analysis. Finally, users can also generate test data with ideal points and bill parameters arbitrarily specified as arguments by the user for analysis with `oc`. Each of these cases are supported by a similar sequence of function calls, as shown in the diagrams below:

ORD file $\xrightarrow{\texttt{readKH()}}$ *rollcall* object $\xrightarrow{\texttt{oc()}}$ *OCobject*

Vote matrix $\xrightarrow{\texttt{rollcall()}}$ *rollcall* object $\xrightarrow{\texttt{oc()}}$ *OCobject*

Following generation of an `OCobject`, the user then analyzes the results using the *plot* and *summary* methods, including:

- **plot.OCcoords()**: Plots ideal points in one or two dimensions.
- **plot.OCangles()**: Plots a histogram of cut lines.
- **plot.OCcutlines()**: Plots a specified percentage of cutlines (a Coombs mesh).
- **plot.OCskree()**: Plots a Skree plot with the first 20 eigenvalues.
- **plot.OCobject()**: S3 method for an `OCobject` that combines the four plots described above.
- **summary.OCobject()**: S3 method for an `OCobject` that summarizes the estimates.

Examples of each of the three cases described here are presented in the following sections.

3 Optimal Classification with ORD files

This is the use case that the majority of `oc` users are likely to fall into. Roll call votes in a fixed width format *ORD* format for all U.S. Congresses are stored online for download at:

- *https://legacy.voteview.com/*
- *https://voteview.com* (updates votes in real time)

`oc` takes `rollcall` objects from Simon Jackman's `pscl` package as input. The package includes a function, *readKH()*, that takes an *ORD* file and automatically transforms it into a `rollcall` object as desired. Refer to the documentation in `pscl` for more detailed information on *readKH()* and *rollcall()*. Using the 90th Senate as an example, we can download the file *sen90kh.ord* and read the data in R as follows:

```
> library(oc)
> #sen90 <- readKH("https://voteview.com/static/data/out/votes/S090_votes.ord")
> data(sen90)     #Does same thing as above
> sen90
```

```
Source:                  C:/sen90kh.ord
Number of Legislators:         102
Number of Votes:          596

Using the following codes to represent roll call votes:
Yea:                  1 2 3
Nay:                  4 5 6
Abstentions:          7 8 9
Not In Legislature:          0

Legislator-specific variables:
[1] "state"      "icpsrState" "cd"         "icpsrLegis" "party"
[6] "partyCode"
Detailed information is available via the summary function.
```

To make this example more interesting, suppose we were interested in applying *oc()* only to bills that pertained in some way to agriculture. Keith Poole and Howard Rosenthal's VOTEVIEW software allows us to quickly determine which bills in the 90th Senate pertain to agriculture.[2] Using this information, we create a vector of roll calls that we wish to select, then select for them in the `rollcall` object. In doing so, we should also take care to update the variable in the `rollcall` object that counts the total number of bills, as follows:

```
> selector <- c(21,22,44,45,46,47,48,49,50,53,54,55,56,58,59,60,61,62,65,66,67,68,
> sen90$m <- length(selector)
> sen90$votes <- sen90$votes[,selector]
```

oc() takes a number of arguments described fully in the documentation. Most of the arguments can (and probably should) be left at their defaults, particularly when estimating ideal points from U.S. Congresses. The default options estimate ideal points in two dimensions without standard errors, using the same beta and weight parameters as described in the introduction. Votes where the losing side has less than 2.5 per cent of the vote, and legislators who vote less than 20 times are excluded from analysis.

The most important argument that *oc()* requires is a set of legislators who have positive ideal points in each dimension. This is the *polarity* argument to *oc()*. In two dimensions, this might mean a fiscally conservative

[2]VOTEVIEW for Windows can be downloaded at *legacy.voteview.com*.

legislator on the first dimension, and a socially conservative legislator on the second dimension. Polarity can be set in a number of ways, such as a vector of row indices (the recommended method), a vector of names, or by any arbitrary column in the *legis.data* element of the `rollcall` object. Here, we use Senators Sparkman and Bartlett to set the polarity for the estimation. The names of the first 12 legislators are shown, and we can see that Sparkman and Bartlett are the second and fifth legislators respectively.

```
> rownames(sen90$votes)[1:12]

 [1] "JOHNSON (D USA)"  "SPARKMAN (D AL)"  "HILL (D AL)"      "GRUENING (D AK)"
 [5] "BARTLETT (D AK)"  "HAYDEN (D AZ)"    "FANNIN (R AZ)"    "FULBRIGHT (D AR)"
 [9] "MCCLELLAN (D AR)" "KUCHEL (R CA)"    "MURPHY (R CA)"    "DOMINICK (R CO)"

> result <- oc(sen90, polarity=c(2,5))

Preparing to run Optimal Classification...

        Checking data...

                ... 1 of 102 total members dropped.

                Votes dropped:
                ... 36 of 208 total votes dropped.

        Running Optimal Classification...

                Generating Start Coordinates...
                Running Edith Algorithm...
                Getting normal vectors...
                Getting legislator coordinates...
                Getting normal vectors...
                Getting legislator coordinates...
                Getting normal vectors...
                Getting legislator coordinates...
                Getting normal vectors...
                Getting legislator coordinates...
                Getting normal vectors...
```

```
Getting legislator coordinates...
Getting normal vectors...
Getting legislator coordinates...
Getting normal vectors...
Getting legislator coordinates...
Getting normal vectors...
Getting legislator coordinates...
Getting normal vectors...
Getting legislator coordinates...
Getting normal vectors...
Getting legislator coordinates...
```

```
Optimal Classification completed successfully.
Optimal Classification took 1.308 seconds to execute.
```

`result` now contains all of the information from the OC estimation, the details of which are fully described in the documentation for *oc()*. `result$legislators` contains all of the information from the `PERF25.DAT` file from the old Fortran *oc()*, while `result$rollcalls` contains all of the information from the old `PERF21.DAT` file. The information can be browsed using the *fix()* command as follows (not run):

```
> legisdata <- result$legislators
> fix(legisdata)
```

For those interested in just the ideal points, a much better way to do this is to use the *summary()* function:

```
> summary(result)

SUMMARY OF OPTIMAL CLASSIFICATION OBJECT
---------------------------

Number of Legislators:         101 (1 legislators deleted)
Number of Votes:         172 (36 votes deleted)
Number of Dimensions:         2
Predicted Yeas:                6845 of 7732 (88.5%) predictions correct
```

```
Predicted Nays:                6745 of 7570 (89.1%) predictions correct

The first 10 legislator estimates are:
                 coord1D coord2D
JOHNSON (D USA)   -0.497  -0.330
SPARKMAN (D AL)    0.220   0.682
HILL (D AL)        0.427   0.681
GRUENING (D AK)   -0.533   0.594
BARTLETT (D AK)   -0.463   0.621
HAYDEN (D AZ)      0.059   0.841
FANNIN (R AZ)      0.629  -0.225
FULBRIGHT (D AR)   0.103   0.882
MCCLELLAN (D AR)   0.455   0.332
KUCHEL (R CA)     -0.095  -0.447
```

`result` can also be plotted, with a basic summary plot achieved as follows as shown Figure 1:

This basic plot splits the window into 4 parts and calls *plot.OCcoords()*, *plot.OCangles()*, *plot.OCskree()*, and *plot.OCcutlines()* sequentially. Each of these four functions can be called individually. In this example, the coordinate plot on the top left plots each legislator with their party affiliation. A unit circle is included to illustrate how OC scores are constrained to lie within a unit circle. Observe that with agriculture votes, party affiliation does not appear to be a strong predictor on the first dimension, although the second dimension is largely divided by party line. The skree plot shows the first 20 eigenvalues, and the rapid decline after the second eigenvalue suggests that a two-dimensional model describes the voting behavior of the 90th Senate well. The final plot shows 50 random cutlines, and can be modified to show any desired number of cutlines as necessary.

Three things should be noted about the use of the *plot()* functions. First, the functions always plot the results from the first two dimensions, but the dimensions used (as well as titles and subheadings) can all be changed by the user if, for example, they wish to plot dimensions 2 and 3 instead. Secondly, plots of one dimensional `oc` objects work somewhat differently than in two dimensions and are covered in the example in the final section. Finally, *plot.OCcoords()* can be modified to include cutlines from whichever votes the user desires. The cutline of the 14th agricultural vote (corresponding to the 58th actual vote) from the 90th Senate with ideal points is plotted below

```
> plot(result)

NULL
```

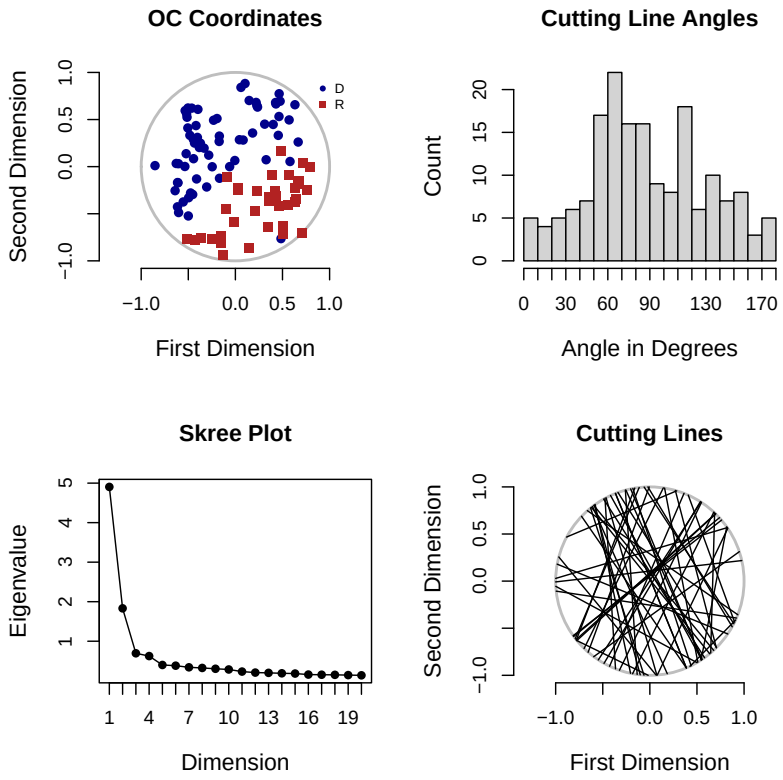

Figure 1: Summary Plot of 90th Senate Agriculture Bill OC Scores

in Figure 2, showing that the vote largely broke down along partisan lines.

4 Optimal Classification with arbitrary vote matrix

This section describes an example of OC being used for roll call data not already in *ORD* format. The example here is drawn from the first three sessions of the United Nations, discussed further as Figure 5.8 in Keith Poole's *Spatial Models of Parliamentary Voting* [1].

To create a `rollcall` object for use with *oc()*, one ideally should have three things:

- A matrix of votes from some source. The matrix should be arranged as a *legislators* x *votes* matrix. It need not be in 1/6/9 or 1/0/NA format, but users must be able to distinguish between Yea, Nay, and missing votes.
- A vector of names for each member in the vote matrix.
- OPTIONAL: A vector describing the party or party-like memberships for the legislator.

The `oc` package includes all three of these items for the United Nations, which can be loaded and browsed with the code shown below. The data comes from Eric Voeten at George Washington University. In practice, one would prepare a roll call data set in a spreadsheet, like the one available one *legacy.voteview.com/k7ftp/dtaord/UN.csv*, and read it into R using *read.csv()*. The csv file is also stored in this package and can be read using:

UN<-read.csv("library/oc/data/UN.csv",header=FALSE,strip.white=TRUE)

The line above reads the exact same data as what is stored in this package as R data, which can be obtained using the following commands:

```
> rm(list=ls(all=TRUE))
> data(UN)
> UN<-as.matrix(UN)
> UN[1:5,1:6]
```

```
> par(mfrow=c(1,1))
> plot.OCcoords(result,cutline=14)
```

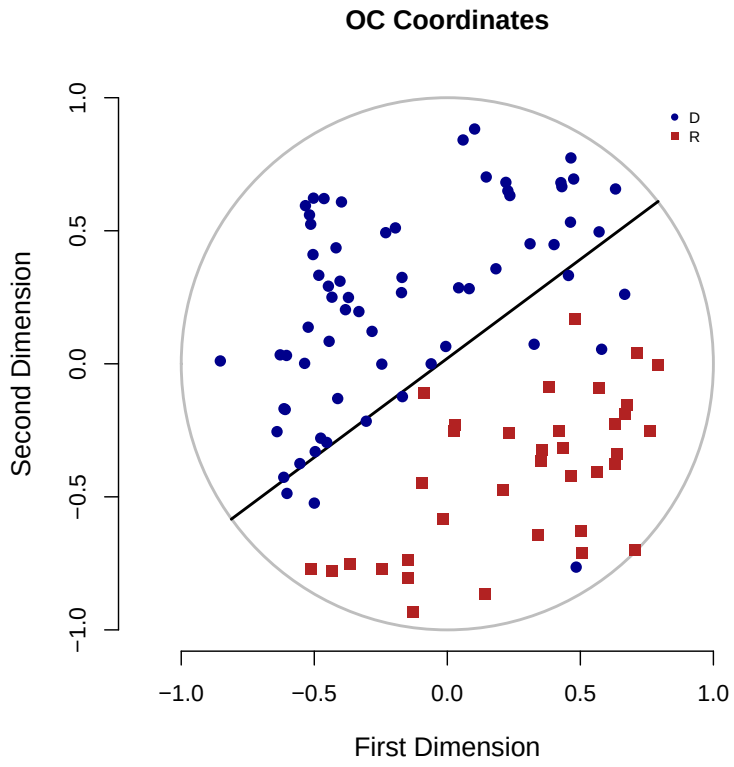

Figure 2: 90th Senate Agriculture Bill OC Scores with Cutline

```
  V1               V2      V3  V4  V5  V6
1 "United States" "Other" "1" "6" "6" "6"
2 "Canada"        "Other" "6" "6" "6" "6"
3 "Cuba"          "Other" "1" "6" "1" "1"
4 "Haiti"         "Other" "1" "6" "6" "9"
5 "Dominican Rep" "Other" "1" "6" "6" "7"
```

Observe that the first column are the names of the legislators (in this case, countries), and the second column lists whether a country is a "Warsaw Pact" country or "Other", which in this case can be thought of as a 'party' variable. All other observations are votes. Our objective here is to use this data to create a `rollcall` object through the *rollcall* function in `pscl`. The object can then be used with *oc()* and its plot/summary functions as in the previous *ORD* example.

To do this, we want to extract a vector of names (*UNnames*) and party memberships (*party*), then delete them from the original matrix so we have a matrix of nothing but votes. The *party* variable must be rolled into a matrix as well for inclusion in the `rollcall` object as follows:

```
> UNnames<-UN[,1]
> legData<-matrix(UN[,2],length(UN[,2]),1)
> colnames(legData)<-"party"
> UN<-UN[,-c(1,2)]
```

In this particular vote matrix, Yeas are numbered 1, 2, and 3, Nays are 4, 5, and 6, abstentions are 7, 8, and 9, and 0s are missing. Other vote matrices are likely different so the call to *rollcall* will be slightly different depending on how votes are coded. Party identification is included in the function call through `legData`, and a `rollcall` object is generated and applied to OC as follows. A one dimensional OC model is fitted, and result is summarized below and plotted in Figure 3:

```
> rc <- rollcall(UN, yea=c(1,2,3), nay=c(4,5,6),
+ missing=c(7,8,9),notInLegis=0, legis.names=UNnames,
+ legis.data=legData,
+ desc="UN Votes",
+ source="legacy.voteview.com")
> result<-oc(rc,polarity=1,dims=1)
```

```
Preparing to run Optimal Classification...

        Checking data...

                All members meet minimum vote requirements.

                Votes dropped:
                ... 18 of 237 total votes dropped.

        Running Optimal Classification...

                Generating Start Coordinates...
                Running Edith Algorithm...
                Permuting adjacent legislator pairs...
                Permuting adjacent legislator triples...


Optimal Classification completed successfully.
Optimal Classification took 0.306 seconds to execute.

> summary(result)

SUMMARY OF OPTIMAL CLASSIFICATION OBJECT
---------------------------

Number of Legislators:         59 (0 legislators deleted)
Number of Votes:         219 (18 votes deleted)
Number of Dimensions:         1
Predicted Yeas:                 4758 of 5039 (94.4%) predictions correct
Predicted Nays:                 4068 of 4488 (90.6%) predictions correct

The first 10 legislator estimates are:
              coord1D
United States    50.0
Canada           57.0
Cuba             27.0
Haiti            24.0
Dominican Rep    40.5
```

```
> plot(result)
```

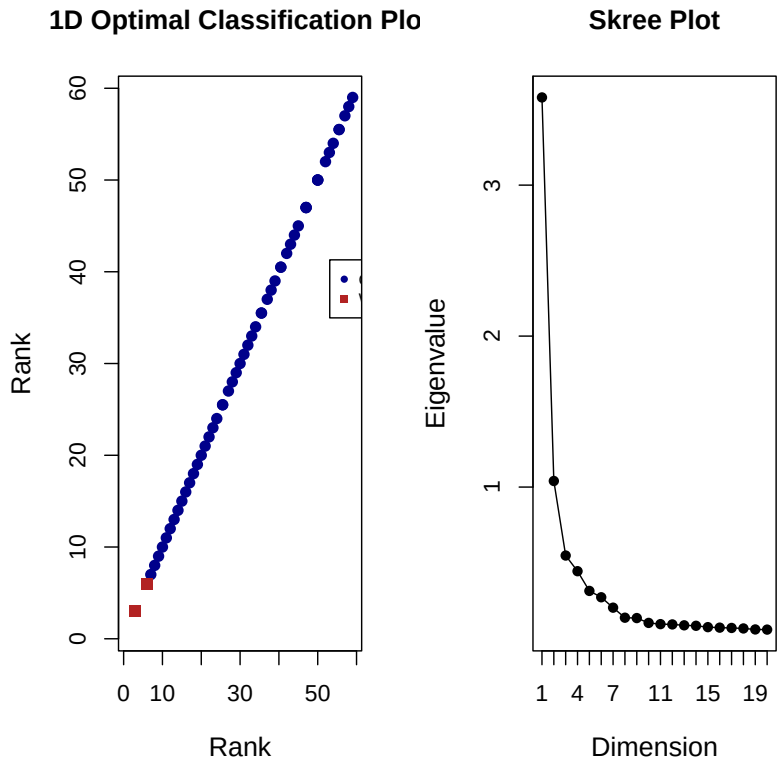

Figure 3: Summary Plot of UN Data

```
Mexico          25.5
Guatemala        9.0
Honduras        29.0
El Salvador     38.0
Nicaragua       39.0
```

Note that the one dimensional plot differs considerably from the previous two dimensional plots, since only a coordinate plot and a Skree plot are shown. This is because in one dimension, all cutlines are angled at 90°, so there is no need to plot either the cutlines or a histogram of cutline angles. Also, the plot appears to be compressed, so users need to expand the image

manually by using their mouse and dragging along the corner of the plot to expand it.

References

[1] Poole, Keith (2000) "Non-Parametric Unfolding of Binary Choice Data." *Political Analysis* 8: 211-237.

[2] Poole, Keith (2005) *Spatial Models of Parliamentary Voting.* Cambridge: Cambridge University Press.