JudgeIt II: A Program for Evaluating Electoral Systems and Redistricting Plans[1]

Gary King[2] Andrew C. Thomas[3]

Version 1.0.4
March 28, 2007

[1]Available from http://GKing.Harvard.edu/judgeit.

[2]David Florence Professor of Government, Harvard University (Institute for Quantitative Social Science, 1737 Cambridge Street, Harvard University, Cambridge MA 02138; http://GKing.Harvard.Edu, King@Harvard.Edu, (617) 495-2027).

[3]PhD Candidate, Department of Statistics, and Graduate Associate, Institute for Quantitative Social Science, Harvard University. http://www.fas.harvard.edu/∼acthomas/; acthomas@fas.harvard.edu.

Contents

1 Introduction

JudgeItII brings the analytical routines of the original version of JudgeItto the R Project for Statistical Computing, while also greatly simplifying its interface and control system.[1] The methods implemented in this software were developed in Gelman and King (1990*b*,*a*, 1994*b*,*a*); King and Gelman (1991); Gelman, Katz and King. (2004).

JudgeItallows a user to construct a model of a two-party election system over multiple election cycles, derive quantities of interest about the system through statistical estimation and simulation, and produce output summary statistics and graphical plots of those quantities. Some of the quantities of interest are based on partisan symmetry as a standard of fairness in legislative redistricting, such as *partisan bias* as the deviation from fairness and *electoral responsiveness* which indexes how party control of legislative seats responds to changes in a party's success at the polls even in a fair system. (A uniform consensus has existed in the academic literature since at least King and Browning (1987) on partisan symmetry as a standard for fairness, and even the U.S. Supreme Court now appears to agree; see Grofman and King 2007.) JudgeItalso estimates and graphs seats-votes curves, make specific vote and seat predictions for individual districts, and calculate numerous other relevant statistics.

The program can evaluate electoral systems in three general situations:

1. When an election already has taken place,
2. When an election has not been held yet but a new redistricting plan (or plans) has been proposed or implemented, and
3. When you wish to assess what an election would have been like if held under certain specified counterfactual conditions (such as if no minority districts had been drawn, or term limitations had prevented incumbents from running for reelection).

For bias, responsiveness, seats-votes curves, and virtually every other estimate, JudgeItprovides quantitative estimates of uncertainty (i.e., standard errors or confidence intervals). This are recognized as essential by social scientists and even the Supreme Court (see Castaneda v. Partida, 430 U.S. 482, 1977, and Hazelwood School District v. United States, 433 U.S. 299, 1977).

2 Installation Instructions and Requirements

As this version of JudgeIthas been built as a package for the mathematical software R, it is necessary to install this software before beginning.

The latest version of R is available for free from `http://www.r-project.org`. After installation, retrieve the JudgeItpackage through the following command:

```
> install.packages("JudgeIt", repos = "http://gking.harvard.edu")
```

To use the commands within, type

```
> library(JudgeIt)
```

JudgeItrequires the R package `mvtnorm` to function properly, and will load it automatically when needed.

2.1 Case Syntax

In all instances, the stylized JudgeItrefers to the software; the mixed case **JudgeIt** refers explicitly to the library name; and the lower case `judgeit`is used for the primary analysis function and any contained terms, such as `judgeit.object`.

[1] The original version of JudgeIt, written by Gary King and Andrew Gelman in versions available for DOS and in the language Gauss, is still available at `http://gking.harvard.edu/judgeit/judgeitI`.

3 Usage Overview

JudgeItcan conduct three types of analyses:

1. Evaluation, wherein the information gathered from one year's elections is applied to itself. This is done to suggest the underlying structure of an electoral system.

2. Prediction, which takes a set of observed covariates and predicts the outcome of an election. This can be done with or without the previous election's results and covariates included explicitly.

3. Counterfactual analysis, which estimates the results if the election had been run under different circumstances (i.e. with different predictors.)

The JudgeItinterface requires only **one** command to perform analyses, called `judgeit`. The standard auxiliary functions `summary`, `print` and `plot` have been adapted to display the object that results.

Two steps are required to perform a JudgeItanalysis: election data must be loaded into a JudgeItobject, and the routine for analysis must be specified along with its required options.

4 Formatting Data Sets

JudgeItrequires each election to be reported in an object of type `data.frame`. This is natural if your data is in a comma-separated or tab-delimited file, since operations like

```
data <- read.csv("ohio.csv")
data2 <- read.table("cali98.txt")
```

will automatically put these tables into the data frame format. It is important to make sure that each variable is labeled before using it in the function `judgeit`, either by making the first line of the data file contain variable names, or by assigning them afterwards. For example:

```
data <- read.csv("ohio.csv",header=FALSE)
colnames(data) <- c("district","vote","inc")
```

Since most JudgeItanalyses will involve a series of consecutive elections, `judgeit`accepts such a series as an object of type `list`. Suppose we have three elections, say Ohio's State Legislature in 1992, 1994 and 1996, and that each election is stored in a data frame as above. To integrate the separate data frames, use the command `elections <- list(oh1992,oh1994,oh1996)` which will create an object named "elections". Each year's election data *must* use the same variable names, so that JudgeItcan recognize the same quantity across different elections.

5 Included Data Sets

JudgeItcomes with data set ICSPR 6311, coded as `house6311`, containing the results for the U.S. House of Representatives from 1896 through 1992. It is formatted as a list of 49 data frames, each named for their election year, and containing six vectors:

- `STATE`, a numerical ICPSR state code for the state containing the district;
- `DIST`, a numerical indicator for the number of the district within the state;
- `INC`, an incumbency indicator, -1 for Republican and 1 for Democrat;
- `VOTE`, the share of the vote received by the Democratic candidate;
- `TURNOUT`, the number of votes for the two candidates combined, and
- `DELSOUTH`, an indicator for whether the district is located in the South.

6 Step 1: Create a JudgeItobject

Now that there is an object containing the elections we wish to analyze, we can load this object into JudgeIt. To do this:

```
judg.obj <-
judgeit(model.formula=vote ~ predictor1 + predictor2 + modfun(vote),
    data=elections, vote.formula=cbind(turnout,eligible)~seats, ...) }
```

where the ellipsis "..." represents other possible commands used by the R command `model.frame`.

Model Formula

The model formula, `model.formula`, specifies the system's outcome variable — the *proportion* of the two-party vote received by the candidate for a specified party — and any desired predictors, such as incumbency status. The predictor names must be the same as the variable names within the data list (referred to here as `elections`.)

Functions can also be applied to the variables within the formula. For example, if we wish to create an indicator for whether a win is decisive (for example, the winner more than triples the opponent's votes), we first create such a function,

```
modfun <- function(arg) -1*(arg<0.25)+1*(arg>0.75)
```

and then use it in the formula. For example, with data set `house6311` this formula works: `model.formula=VOTE~INC+modfun(VOTE)`.

Voter Formula

We use the voter formula, `vote.formula`, to indicate the *number* of actual voters (also known as the turnout), the number of eligible voters, and/or the number of seats each district elects to the main assembly. If this is omitted, the number of seats is assumed to be one per district and the turnout is assumed to be equal in all districts.

For example,

- `vote.formula=cbind(turnout,eligible)~seats` specifies all three quantities.
- `vote.formula=cbind(turnout,eligible)~1` specifies turnout and eligible voters, and one seat per district.
- `vote.formula=turnout~1` specifies turnout. The number of eligible voters is ignored, and seats are specified at one per district.
- `vote.formula=~1` is the default option.

6.1 Other Options

6.1.1 Selecting a data subset

For example, the data set `house6311` contains an indicator `DELSOUTH`, which is 1 if a district is located in the south and 0 otherwise. To perform analyses on non-southern districts only, initialize a JudgeIt object (in this example, `jud.ob`) with the expression

```
jud.ob <- judgeit(...,subset=DELSOUTH==0,...,data=house6311)
```

6.1.2 Automatic Use of Previous Year's Results

The results of a previous year's election is often among the best predictors of a current contest, and they are included by default. To remove this option, include the option `use.last.votes=F` in your `judgeit`statement.

6.1.3 Redistricting

At some point during a state's existence, the electoral map is redrawn to adjust for changes in the demographics of the districts. Specifying whether the districts in one election are identical to the previous one are vital to an accurate representation of the system.

By default, `judgeit`assumes that if two consecutive years have the same number of districts in the map, then no redistricting has taken place.

To set a redistricting occurrence manually, `judgeit`accepts as an option `same.districts`, which indicates whether the previous election used the same districting plan. In many states in the American system, the easiest way to do this is to have a variable specifying the election years and note whether the year ends in 2 (`year %% 10 == 2`). For other situations, you will need to add a specially coded variable to the data frame and set `same.districts` to it.

6.1.4 Uncontested Districts

JudgeItincludes several routines to deal with uncontested districts. By default JudgeItassumes that a vote proportion below 0.05 or above 0.95 indicates an uncontested district and are imputed; these levels can be changes using the options `uncontested.low` and `uncontested.high`.

The option `uncontesteds.method` indicates which method is to be used to deal with districts tagged as uncontested:

- `"default"`, which replaces uncontested values with assumed vote totals specified by **uncontested.low.new** and **uncontested.high.new** (default values are 25% and 75%);
- `"impute"`, which uses the behaviour of contested districts to estimate unknown vote shares,
- `"remove"`, which simply eliminates uncontested districts from the analysis, or
- `"nochange"` which leaves the vote variable as entered.

6.1.5 Simulation Parameters

JudgeIt analyzes election systems by simulating a number of hypothetical draws of the election of interest, and computing quantities of interest from these simulations. The default number of simulated elections is 1000; to change this, include the option `simulations` in the `judgeit`command.

6.1.6 Applying District Weights: Some Count More than Others

Districts may have different influence than others when determining the parameters of a system. To reflect this, we can assign a `weight` parameter to command `judgeit`:

- `weight=constant` (default) in which all districts have equal weight;
- `weight=turnout`, corresponding to the two-party votes cast in each district;
- `weight=eligible.voters`, corresponding to the total number of eligible voters in each district, or
- `weight=seats`, the number of seats in each district.

This will have two distinct impacts on the analysis. First, in the linear modelling, each district will have a variance equal to a constant (σ^2) times the specified weight. Second, the mean vote will be calculated with respect to this weight; a `constant` weight will yield the average district vote, while setting the weight according to `turnout` will yield the grand average or "popular" vote.

7 Step 2: Conduct Analyses

Once an election system has been loaded into a JudgeItobject, quantities of interest can be simulated. There are three types of analyses than can be performed:

7.1 Evaluation

The default setting of JUDGEITfor analysis, evaluation mode examines an election under the actual conditions that took place. Under these conditions, the systematic error component in simulation (the amount estimated by the parameter λ) is identical, and total error estimates are therefore smaller.

By default, all substantive outputs from `judgeit`use evaluation mode. For example, using a previously created JUDGEITobject `judgeit.object`, we can obtain a seats-votes curve using the command

```
plot(judgeit(routine="seats",judgeit.object=judgeit.object))
```

7.2 Prediction

Prediction takes the information gleaned from one election and uses it to forecast the results of a new election. In this case, since the new election hasn't taken place, the systematic error component is inestimable, and the total error component is generated.

To enable predictive mode, add the option `predict=T` to a `judgeit`analysis command. To specify new covariates for this election, add the option `new.covariates` to specify the new conditions, or use `new.covariate.matrix` to draw a new electoral map entirely.

7.3 Counterfactual Evaluation

This option supposes what might have happened if an election were rerun under different circumstances. The systematic error component is once again held constant to reflect that this is the same "voting apparatus" as was used originally.

To enable a counterfactual analysis, specify new covariates by adding the option `new.covariates` and changing the relevant values.

7.4 Using new.covariates to recode data

In order to recode the data for counterfactual analysis or prediction, the option `new.predictors` must be added to a `judgeit`analysis command.

As an example, suppose we wish to find out what were happened if term limits were imposed and no incumbent could run for re-election, and that the incumbency indicator is labelled `INC`. The option

```
new.covariates=list("INC",0)
```

would instruct each routine to use this counterfactual data.

Suppose it came to pass that, instead, a group of representatives were forced to resign shortly before the election. If the new incumbency indicators are stored in a variable `new.incs`,

```
new.covariates=list("INC",new.incs)
```

would make that substitution. Make sure, however, that the substitution variable has the same number of districts as the model for that year; to confirm this, run the command

```
summary(judgeit.object,year).
```

7.5 distreport: Examine the estimation

The output of this routine is a reckoning of each district, including the observed and hypothetical vote shares, the standard deviation of the hypothetical estimate, and the probability that a voter in this district would be able to change the outcome of the entire election by reversing their vote.

7.6 seats: Given a vote share, determine the seat share

This is the basic tool used to determine the conversion between seats and votes in this electoral system.

Example: `judgeit(routine=‘‘seats",judgeit.object=j.ob)` will produce estimations and errors for the fraction of seats received using the default options: in particular, in the final election in the system, at various values between 0.45 and 0.55.

Example: `judgeit(routine="seats", judgeit.object=j.ob,voterange=c(0.1,0.9))` will now produce estimates over a much wider vote range. Note that the farther we get from the actual outcome, the less reliable the model will become.

7.6.1 Plotting a seats-votes curve

If the output in a JudgeItobject was produced by `seats`, the `plot()` command will recognize this and produce a seats-votes curve.

7.7 prob: Given a vote share, determine the probability of an electoral win

As above, but replace "seats" with "likelihood of a majority for party 1".

Example: `judgeit(routine="prob",judgeit.object=j.ob)` will produce estimations and errors for the fraction of seats received using the default options: in particular, in the final election in the system, at various values between 0.45 and 0.55.

Example: `judgeit(routine="prob",judgeit.object=j.ob,voterange=c(0.1,0.9))` will now produce estimates over a much wider vote range. Note that the farther we get from the actual outcome, the less reliable the model will become.

7.8 winprob: Given a vote share or deviation, determine the probability that the seat share is within a particular margin

Example: `judgeit(routine="winprob",judgeit.object=j.ob)` does the default: For the vote as it is, what is the probability that the seat share will be between the default range of 0.45 to 0.55?

Example:

```
judgeit(routine="winprob",judgeit.object=j.ob,
voteorshift="shift",voteshares=-0.05)
```

determines what the probability is that the seat share will be between the default range of 0.45 to 0.55, given that the resulting vote share was 5 points lower than actually experienced.

Example:

```
judgeit(routine="winprob",judgeit.object=j.ob,
voterange=c(0,0.5),voteorshift="vote",voteshares=0.5)
```

determines what the probability is that the seat share will be below 50 percent, given that the resulting vote share was 50 percent for each party.

7.9 svsum: Determine partisan bias and responsiveness

The title says it all. The output is a 4-by-2 table containing the estimates and errors for four quantities: Partisan bias at an even vote, both instantaneous and averaged over a 10-point swing, and responsiveness at the midpoint and the observed vote percentage.

Example:

```
judgeit(routine="svsum",judgeit.object=j.ob,year=which(elecyears==1976))
```

will output those properties for the election held in 1976. (This assumes there is a variable called `elecyears` which encodes the calendar years of each election.)

7.10 winvote: Given a seat share, determine the total vote share required to attain it

The reverse procedure of `seats`. **Example:**

```
judgeit(routine="winvote", judgeit.object=j.ob,winvote=0.7)
```

outputs the expected vote percentage needed to get 70% of the seats.

7.11 voting.power: Determine the power that distinct groups have in changing an election result

Suppose there exists a distinct number of groups spread across the districts of an electoral map (by race, language, or some other delineation.) The routine `voting.power` calculates the power of an electoral group by estimating the probability that a vote from this group has in changing the result of an election, by flipping the outcome of districts in which they live.

Example:

```
judgeit(routine="voting.power", judgeit.object=j.ob,voting.groups=matrix1,all.groups=matri
```

`matrix1` represents the voters in each group within each district; `matrix2` represents the eligible voters respectively. Each must have the same number of rows as there are districts in the system.

8 Further JudgeItoutput options

8.1 Plotting output to file

JudgeItoutputs can be printed to a PNG graphics file by giving a file name in the plot command, like so:

```
plot(jud.obj,filename="txseatsvotes")
```

producing the file `txseatsvotes.png`.

8.2 System summaries

Using the command `summary()` on a JudgeItobject will give one of two results. Without a year given, the output will be the number of years, as well as the values of model parameters λ and σ.

With a year given, a report of the vote outcomes, predictors, seats and populations will be the output.

9 Accessing Raw judgeit.object Data

All data are stored in an object of class `judgeit`. If desired, the user can access each component within. Here is a list of components and their attributes:

- `covars` is a list of data frames comprising the predictors for each election in the system. So `judgeit.object$covars[[25]]` is a data frame with the covariates from the 25th election.
- `voteshare` is a list of vectors comprising the vote shares for each election in the system. So `judgeit.object$voteshare[[25]]` is a vector of the results of the 25th election.
- `turnout,elgvotes,seats` are lists of vectors comprising the actual turnout, the number of eligible voters, and the seats per district in the system for each election.
- `fullrow` is a list of vectors containing those rows whose primary elements (covariates, vote shares, eligible and actual voters and seats) contain complete data.
- `uncL,uncLR,uncU,uncUR` are the uncontested election detection thresholds and imputations as listed above.

- `svexpected.value.only` is the value of `expected.value.only` as given above.
- `simulations` is the number of simulations conducted by `JudgeIt` during each analysis.
- `weight` is the option selected by the user to indicate what weights should be used in the linear model, as described above.
- `distweights` is a list of the actual values of these weights.
- `covarsnew` is a list of data frames of counterfactual or future predictors as manipulated by the option `new.covariates`. It must have the same data type in each column as `covars` though not necessarily the same number of rows.
- `same.dists` is a vector indicating whether the previous election's district map is identical to the current one, as described above.
- `output` contains the output of the last analytical routine, and is displayed with the command `print(JudgeIt.object)`.
- `outputyear, outputclass` indicate the year and type of the last analysis conducted. These are used mainly in `plot(judgeit.object)`.
- `beta,vc` are the estimates given by the linear model for the system for the coefficients of the covariates and their covariance matrix.
- `sigma,lambda,sind,lind` are, respectively, the mean and year-by-year estimates of the standard error and systematic error fraction of the system.
- `years` is a vector of the names of each election variable in the inputted data frame list. In the case of `house6311`, this is a vector of the election years between 1896 and 1992.

A Several demos

After loading the JudgeIt library, you may run these commands to check that it's in working order.

In addition, the demonstrations `seatsdemo, probdemo, distreportdemo` and `svsumdemo` are available through the command `demo`.

```
data(house6311)
#columns: STATE,DIST,INC,VOTE,TURNOUT,DELSOUTH

#operators:
unc <- function(inp) -1*(inp<0.05)+1*(inp>0.95)

years <- seq (1896,1992,by=2)
same.dists <- 1*(yrs%%10!=2)

j.ob <- judgeit(model.formula=VOTE~unc(VOTE)+INC,vote.formula=TURNOUT~1,
               data=house6311,
               use.last.votes=T,subset=DELSOUTH==0,same.d=same.dists)

summary(j.ob)
summary(j.ob,which(house6311$years==1942))

j.ob <- judgeit(routine="distreport",judgeit.object=j.ob,year=which(years==1962),new.covariates
j.ob

#seats-votes curve
j.ob <- judgeit(routine="seats",jud=j.ob,year=which(years==1986),vote.range=c(0.2,0.8))
plot(j.ob)
```

References

Gelman, Andrew and Gary King. 1990*a*. "Estimating Incumbency Advantage Without Bias." *American Journal of Political Science* 34(4, November):1142–1164. http://gking.harvard.edu/files/abs/inc-abs.shtml.

Gelman, Andrew and Gary King. 1990*b*. "Estimating the Electoral Consequences of Legislative Redistricting." *Journal of the American Statistical Association* 85(410, June):274–282. http://gking.harvard.edu/files/abs/svstat-abs.shtml.

Gelman, Andrew and Gary King. 1994*a*. "Enhancing Democracy Through Legislative Redistricting." *American Political Science Review* 88(3, September):541–559. http://gking.harvard.edu/files/abs/red-abs.shtml.

Gelman, Andrew and Gary King. 1994*b*. "A Unified Method of Evaluating Electoral Systems and Redistricting Plans." *American Journal of Political Science* 38(2, May):514–554. http://gking.harvard.edu/files/abs/writeit-abs.shtml.

Gelman, Andrew, Jonathan Katz and Gary King. 2004. *Empirically Evaluating the Electoral College*. New York: Oxford University Press chapter 5, pp. 75–88. http://gking.harvard.edu/files/abs/rethink-abs.shtml.

Grofman, Bernard and Gary King. 2007. "The Future of Partisan Symmetry as a Judicial Test for Partisan Gerrymandering after LULAC v. Perry." *Election Law Journal* 6(1, January):2–35. http://gking.harvard.edu/files/abs/jp-abs.shtml.

King, Gary and Andrew Gelman. 1991. "Systemic Consequences of Incumbency Advantage in the U.S. House." *American Journal of Political Science* 35(1, February):110–138. http://gking.harvard.edu/files/abs/sysconseq-abs.shtml.

King, Gary and Robert X Browning. 1987. "Democratic Representation and Partisan Bias in Congressional Elections." *American Political Science Review* 81(4, December):1252–1273. http://gking.harvard.edu/files/abs/sv-abs.shtml.