JUDGEM: The Next Generation of Electoral Analysis Software[1]

Gary King[2] Andrew C. Thomas; acthomas@fas.harvard.edu).

Version 1.0-1
September 22, 2006

[1]Available from http://GKing.Harvard.Edu/judgem.

[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).

Contents

1 Introduction

Judgem brings the analytical routines of JudgeIt, the electoral analysis software by Gary King and Andrew Gelman, to the modern R programming interface while highly simplifying its interface and control system.

2 Installation Instructions

If you haven't downloaded the source distribution, do so. Once you have it, Beginning with the source distribution for Judgem, enter the following command into your favorite Terminal program:

```
R CMD INSTALL judgem
```

This will install the package into your version of R. To use the commands within, type

```
>library(judgem)
```

Note that judgem requires package `mvtnorm` to function properly.

3 Theory

See instructions for Judgeit to determine the model. More to come.

4 Usage Overview

There are three general types of analyses Judgem can conduct:

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

The Judgem interface requires only **one** command to perform analyses, called (appropriately enough) `judgem`. The standard auxiliary functions `summary`, `print` and `plot` have been adapted to display the analysis as neatly as possible as well.

5 Formatting Data Sets

Judgem requires each election to be reported in the `data.frame` type. This is natural if your data is in a comma-separated or tab-delimited file, since the functions

```
data <- read.csv("ohio.csv")

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

automatically output data frames. It is important to make sure that each variable is labeled before using it in `judgem`, either by making the first line of the data file contain variable names, or by assigning them afterwards like so:

```
data <- read.csv("ohio.csv",header=FALSE)

colnames(data) <- c("district","vote","inc")
```

Since most Judgem analyses will involve a series of consecutive elections, `judgem` 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 them into one, use the command

`elections <- list(oh1992,oh1994,oh1996)`

which will create the object (appropriately) named "elections".

It is highly important that each year's election data use the same variable names, so that Judgem can recognize the same quantity across different elections.

6 Enclosed Data Sets

At this time, Judgem comes loaded 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 indicator 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.

7 Step 1: Create a Judgem object

Now that there is an object containing the elections we wish to analyze, we can load this object into Judgem in order to perform preliminary analyses and light scrutiny.

To do this, the command takes the following form

```
judg.obj <-
judgem(modelform=vote~predictor1+predictor2+modfun(vote),
    data=elections,
    voteform=cbind(turnout,eligible)~seats,
    ...) }
```

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

Model Formula

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

It is also possible to apply functions to these predictors 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 make reference to it as in the above example.

One example that will work with the data set `house6311` is the formula

`modelform=VOTE INC+modfun(VOTE)`

Voter Formula

We use the voter formula to indicate the number of actual voters (or 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.

Several examples:

- `voteform=cbind(turnout,eligible) seats` specifies all three quantities.
- `voteform=cbind(turnout,eligible) 1` specifies turnout and eligible voters, and one seat per district.
- `voteform=turnout 1` specifies turnout. The number of eligible voters is ignored, and seats are specified at one per district.
- `voteform= 1` does nothing. Please refrain from entering this command.

7.1 Other Options

7.1.1 Selecting a data subset

As Judgem relies partly on the equation notation of R to arrange inputted data, the option `subset` is available to it.

For example, the data set `house6311` contains an indicator `DELSOUTH`, which highlights whether a district is located in the south. To perform analyses on non-southern districts only, initialize a Judgem object with the expression

```
judgem(...,subset=DELSOUTH==0,...,data=house6311
```

7.1.2 Automatic Use of Previous Year's Results

The results of a previous year's election are among the best predictors of a current contest. To include these automatically, no extra commands are required; to preclude them, include the option `use.last.votes=F` in your `judgem` statement.

7.1.3 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, `judgem` 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, `judgem` accepts as an option `same.districts`, which indicates whether the previous election used the same districting plan. In the American system, the easiest way to do this is to have a variable specifying the election years (a good thing to have anyway) and note whether the year ends in 2 `(year %% 10 == 2)`.

7.1.4 Uncontested Districts

Judgem includes several routines to deal with uncontested districts. The model assumes that a vote share below 5% or above 95% indicates an uncontested district; these levels can be changes using the options `uncons.low` and `uncons.high`.

The option `uncons.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 `uncons.low.new` and **uncons.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"`.

7.1.5 Withholding Preliminary Analysis

By default, the routine `judgem` will conduct a preliminary analysis of the electoral system in order to determine estimates of several properties. These include the fraction of the error which can be considered systematic (the quantity `lambda`), the standard error for each vote estimate (`sigma`), and the coefficients for each predictor in the linear model.

To disable this feature on initial run, specify `prelim=F`. The preliminary analysis will be conducted automatically when a `routine` is chosen, or can be run at any time on a Judgem object with the command `jud.obj <- judgem (judgem.object=jud.obj`.

7.1.6 Simulation Parameters

Judgem analyzes election systems by simulating a number of elections and computing quantities of interest from this simulation. These are set using options `simnum` and `simdepth`.

7.1.7 Extra Options

Since `judgem` uses the R routine `model.frame`, options used by that routine, such as `subset`, can be used here as well. This is ideal for selecting a subset of districts within the system.

8 Step 2: Conduct Analyses

Once an election system has been loaded into a Judgem object, quantities of interest can be simulated. At this time, we can specify the particular conditions for analysis.

- **Evaluation:** The default setting of Judgem analyses, evaluation examines the election as it was. Since the parameter λ estimates the systematic component of the error, simulations hold this part constant while only changing the stochastic error component.
- **Prediction:** Prediction takes the information from one election and uses it to (guess what) predict a new election's results. The entire error is estimated, and new predictors are used to determine the potential outcomes.

  To enable the predictive mode, add the option `predict=T` to a `judgem` analysis command. New predictors can be added as described below.
- **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" being used twice.

8.1 Using new.predictors to recode data

In order to recode the data for counterfaction or prediction, the option `new.predictors` must be added to a `judgem` 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.predictor=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.predictor=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(judgem.object,year)`.

8.2 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.

8.3 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: `judgem(routine=‘‘seats’’,judgem.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: The command `judgem(routine=‘‘seats’’, judgem.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.

8.3.1 Plotting a seats-votes curve

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

8.4 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: `judgem(routine=‘‘prob’’,judgem.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: The command `judgem(routine=‘‘prob’’,judgem.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.

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

Example: `judgem(routine=‘‘winprob’’,judgem.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: `judgem(routine=‘‘winprob’’,judgem.object=j.ob, voteorshift=‘‘shift’’,voteshares=-0.`
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: `judgem(routine=‘‘winprob’’,judgem.object=j.ob,voterange=c(0,0.5),voteorshift=‘‘vote`
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.

8.6 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: `judgem(routine=‘‘svsum’’,judgem.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.)

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

The reverse procedure of `seats`. **Example:** `judgem(routine=''winvote'', judgem.object=j.ob,winvote=0.7)` outputs the expected vote percentage needed to get 70% of the seats.

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

This option is currently not supported. Stay tuned.

9 Further Judgem output options

9.1 Plotting output to file

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

`plot(jud.obj,filename=''drewsfile'')`,
producing the file `drewsfile.png`.

9.2 System summaries

Using the command `summary()` on a Judgem object 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 if a preliminary analysis has been run.

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

10 Accessing Raw judgem.object Data

All data are stored in an object of class `judgem`. 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 `judgem.object$covars[[25]]` is a data frame with the covariates from the 25th election.
- `voteshare` is a list of vectors comprising the vote shares for each election in the system. So `judgem.object$voteshare[[25]]` is a vector of the results of the 25th election.
- `actvotes,elgvotes,seatsper` 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.
- `svEvar` is the value of `Evar` as given above.
- `sims,simd` are the depths of simulation conducted by `judgem` as given above.
- `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.predictors`. They are identical in dimension to `covars`.

- `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(judgem.object)`.
- `outputyear,outputclass` indicate the year and type of the last analysis conducted. These are used mainly in `plot(judgem.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.

11 References

Andrew Gelman, Gary King. "A Unified Method of Evaluating Electoral Systems and Redistricting Plans". American Journal of Political Science, Vol. 38, No. 2, May 1994, Pp. 514-54

Andrew Gelman, Gary King. JudgeIt: A Program for Evaluating Electoral Systems and Redistricting Plans.

A Several demos

After loading the Judgem 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)

j.ob <- judgem(modelform=VOTE~unc(VOTE)+INC,data=house6311,
               use.last.votes=T,subset=DELSOUTH==0)

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

years <- seq (1896,1992,by=2)

j.ob <- judgem(routine="distreport",judgem.object=j.ob,year=which(years==1962),new.predictors=l
j.ob

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

References