

Package ‘JudgeIt’

February 14, 2012

Version 1.3.3

Date 2008-08-26

Title JudgeIt

Author Andrew Gelman <gelman@stat.columbia.edu>, Gary King
<king@harvard.edu>, Andrew C. Thomas <acthomas@fas.harvard.edu>

Maintainer Andrew C. Thomas <acthomas@fas.harvard.edu>

Depends R (>= 2.4)

Description Calculates bias, responsiveness, and other characteristics
of two-party electoral systems, with district-level electoral and other data.

License GPL-2

URL <http://gking.harvard.edu/>

Repository CRAN

Date/Publication 2008-08-27 07:35:41

R topics documented:

house6311	2
judgeit	2
seats	5
v2s	6
vshar	7

Index	8
--------------	----------

house6311

*ICPSR Data Set 6311: U.S. House Elections, 1896-1992***Description**

This set contains data on the elections to the U.S. House of Representatives, every two years between 1896 and 1992.

Usage

house6311

Format

A list containing 49 data frames. Each data frame has the following vectors: STATE (identifier), DIST (identifier), INC (incumbency: -1 for Rep, 1 for Dem), VOTE (fraction of vote for Democratic candidate), TURNOUT (total number of votes), DELSOUTH (1 if in the South, 0 otherwise).

References

ICPSR.

judgeit

*judgeit***Description**

Two-party election modelling and simulation for R.

Usage

```
judgeit (model.formula=~1,vote.formula=NULL,same.districts=NULL,
        data,pop.groups=NULL,

        uncontested.method="default",uncontested.low=0.05,
        uncontested.low.new=0.25,uncontested.high=0.95,
        uncontested.high.new=0.75,

        use.last.votes=TRUE,
        simulations=201,

        weight="constant",
        years=NULL,

        routine=NULL,year=NULL,judgeit.object=NULL,
```

...
)

Arguments

<code>model.formula</code>	The model formula; left side contains the vote outcome as one party's share of the total vote, right side contains predictors.
<code>vote.formula</code>	The vote distribution formula. Left side contains one or two columns with actual and/or eligible turnout, right side contains the number of seats per district. ~~Describe vote.formula here~~
<code>same.districts</code>	A vector noting whether an election had the same district map as the previous one.
<code>data</code>	A list containing all the elections to be modelled in the system. Each item of the list should be an election variable; each election variable should be a data frame containing the information for each district.
<code>pop.groups</code>	A formula indicating groups of subpopulations to be considered in the voting.power routine.
<code>uncontested.method</code>	A choice between "nochange","default","impute" and "delete" for the way to deal with uncontested districts.
<code>uncontested.low</code>	The value below which a district should be considered uncontested by party 1, so that party 2 wins the district's seats.
<code>uncontested.low.new</code>	If method is "default", districts uncontested by party 1 will be considered to have this amount of the vote share for the purposes of the analysis.
<code>uncontested.high</code>	The value above which a district should be considered uncontested by party 2.
<code>uncontested.high.new</code>	If method is "default", districts uncontested by party 2 will be considered to have this amount of the vote share for the purposes of the analysis.
<code>use.last.votes</code>	T/F whether a previous election's votes should be automatically included as a current election's predictor. This is overridden if the district maps are not identical due to the information in 'same.districts', or an unequal number of districts between two elections.
<code>simulations</code>	The number of simulations of election parameters for each analysis.
<code>weight</code>	A choice between "constant","elgvotes","actvotes" and "seats" for the type of weights used in the linear model of each election.
<code>years</code>	A vector indicating the years considered in the analysis. This entry is redundant if the names of the entries in the data list are given as the years.
<code>routine</code>	If desired, the routine to run in order to obtain desired quantities of interest.
<code>year</code>	If desired, the election for which the above routine will be run.
<code>judgeit.object</code>	If desired, a pre-existing object from which judgeit() will obtain the model.
...	Extra commands to be used with model.frame(), a routine used in assembling the linear model for each year.

Value

judgeit() loads several years of electoral history into a judgeit object for further analysis.

Alternatively, judgeit() will perform the routine requested in option "routine", and the function will return a judgeit object with judgeit.object\$output containing the output from the chosen routine.

Author(s)

Andrew Gelman, Gary King & Andrew C. Thomas

References

See JudgeIt website (<http://gking.harvard.edu/judgeit>) for more information.

Examples

```
#Demo files are available through the following commands:
demo(judgeit.primer)
demo(seatsdemo)
demo(probdemo)
demo(svsumdemo)
demo(distreportdemo)

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 <- rep(1,49); same.dists[seq(4,49,by=5)] <- 0

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(years==1942))

d.rep <- district.report(j.ob,year=1962,new.covariates=list("INC",0),vote.range=c(0.1,0.9))
d.rep

#seats-votes curve
seating <- seats(j.ob,year=1986,vote.range=c(0.2,0.8))
plot(seating)
```

Description

`seats`: Given an average vote, determine the distribution of seats received in a two-party election system.

`voting.power`: Determine the average probability of changing an election result given membership in a particular population subgroup.

`seats.with.prob` (or `prob`): Given a probability range, determine the fraction of seats whose probability of victory lies within that range.

`results.prob` (or `winprob`): Given a seats range, determine the probability that the election outcome lies in that range.

`district.report` (or `distreport`): Output information about the system at the district level.

`conditional.seats`: Given a vote range, output the fraction of seats whose outcomes lie within this range.

`votes.for.result` (or `winvote`): Determine the mean vote required to get a given probability of victory/majority for Party 1.

`bias.resp` (or `svsum`): Output the partisan bias and responsiveness of the electoral system under consideration.

Usage

```
seats(judgeit.object,...) #mean.votes=NULL,
voting.power(judgeit.object,...) #new.pop.groups=NULL,
seats.with.prob(judgeit.object,...) #prob.range=c(0.5,1),
results.prob(judgeit.object,...) #seat.range=c(0.5,1),
district.report(judgeit.object,...)
conditional.seats(judgeit.object,...) #vote.range=c(0.4,0.6),
votes.for.result(judgeit.object,...) #prob.win=c(0.1,0.5,0.9),
bias.resp(judgeit.object,...)
```

Arguments

`judgeit.object` An election object produced by `judgeit()`.

... Additional options, including:

`year`: The year to be analyzed. Defaults to the last year in the system.

`predict`: T/F whether prediction, rather than analysis, should be executed.

`new.covariates`: A list of variable names and their replacement values for counterfactual analysis or prediction.

`new.covariate.matrix`: A new matrix of covariates for prediction or counterfactual analysis. This matrix need not have the same number of rows as the old covariate matrix, but must have the identical column setup.

`new.seats`, `new.actual.voters`, `new.eligible.voters`: If a new covariate matrix is used, these are vectors of the new seats per district, turnout, and eligible voters respectively. (Default: one seat per district and identical populations.)

`extra.districts`: Extra districts to be added to the system after analysis but before reporting, typically uncontested districts whose results are unlikely to change. Should be three column matrix, with columns representing vote share, turnout/eligible voters and seats per district respectively.

`district.select`: If a subset of districts is desired for one analysis, this is a numerical vector identifying these districts.

`vote.range`, `prob.range`, `seat.range`: Two-element vectors specifying a mean vote range, desired probability range, or fraction of seats. Their use depends on the routine called.

`mean.votes`: A vector of mean votes to be used as the independent variable in analyses.

`shift.in.votes`: A vector of shifts in the mean vote from the observed value, defining the independent variables in an analysis.

`prob.win`: The desired probability/ies of victory for Party 1.

`new.pop.groups`: Population groups to be used in the `voting.power` routine. These will override groups specified in the `JudgeIt` object.

Value

An object of a `judgeit` sub-class.

Author(s)

Andrew C. Thomas

v2s

v2s - votes to seats

Description

Indicates whether a vote share variable is greater than 0.5.

Usage

`v2s(inp)`

Arguments

`inp` The vote share variable.

Value

`1*(inp>0.5)`

Note

For aid in JudgeIt routines.

Author(s)

Andrew C. Thomas

vshar

vshar - vote share given two vote totals

Description

Given two vote totals (or fractions), outputs the share of the vote received by the first party.

Usage

vshar(r1, r2)

Arguments

r1, r2 vectors of total votes or fractions.

Value

$r1/(r1+r2)$

Author(s)

Andrew C. Thomas

Index

*Topic **array**

seats, [5](#)

v2s, [6](#)

vshar, [7](#)

*Topic **datasets**

house6311, [2](#)

*Topic **models**

judgeit, [2](#)

bias.resp (seats), [5](#)

conditional.seats (seats), [5](#)

distreport (seats), [5](#)

district.report (seats), [5](#)

house6311, [2](#)

JudgeIt (judgeit), [2](#)

judgeit, [2](#)

prob (seats), [5](#)

results.prob (seats), [5](#)

seats, [5](#)

svsum (seats), [5](#)

v2s, [6](#)

votes.for.result (seats), [5](#)

voting.power (seats), [5](#)

vshar, [7](#)

winprob (seats), [5](#)

winvote (seats), [5](#)