R documentation

of ‘`/Users/malecki/Library/Application`’ etc.

January 4, 2013

R topics documented:

`getNumberWays-methods` *Get Number of Ways for MRP analysis*

Description

A cross-classified dataset for multilevel regression and poststratification is an N-dimensional array. Each dimension in the array is one of the “ways” by which it can later be poststratified. For various reasons it may be useful to query a `mrp-class` object or an `NWayData-class` object for these dimensions and their names.

Methods

`signature(object = "mrp")` Returns a vector of length 2 with names ‘poll’ and ‘pop’. In the special case where no population data has been assigned (and an array of 1s is used to return aggregated fitted values from the multilevel model) the value returned for ‘pop’ is 0.

`signature(object = "NWayData")` Returns the number of ways and an attribute ‘ways’ containing the character vector of names of the “ways” by which it is constructed.

mr — *Run Multilevel Regression step of MRP Analysis*

Description

Run a (binomial) multilevel regression in survey data for later poststratification.

Usage

```
  ## S4 method for signature 'mrp'
mr(object, mr.formula, ...)
```

Arguments

`object`	A `mrp` object.
`mr.formula`	A formula specification for the multilevel model to run in the prepared data. The left-hand side should always be ‘`response`’. For convenience, the formula is handled by `update.formula` so that `.` indicates the current formula contents on either side of the `~`, e.g., `.~.+newVar`. The initial default formula is constructed as just an intercept term for each of the variables in the main formula specification (`(1|way1)+(1|way2)` etc.)
`...`	Additional arguments to be passed to the multilevel regression step, which uses `glmer` by default.

See Also

`mrp-class` for an example. `mrp-class` for other methods on the objects produced by `mrp()`; `plotmrp` for how to plot poststratified results onto maps.

mrp — *Multilevel regression and poststratification*

Description

Set up survey and population data, and a multilevel regression model used for poststratifying by an arbitrary number of strata or “ways”.

Usage

```
mrp (formula.cell,
     data,
     population=NULL,
     pop.weights=NULL,
     grouplevel.data.frames=NULL,
     grouplevel.expressions=NULL,
     formula.model.update=NULL,
     poll.weights=1,
     formula.pop.update=formula.cell,
     pop.margin=NULL,
     ...)
```

Arguments

`formula.cell`
A formula representation of the binary outcome variable and the desired eventual poststratification; i.e., the "ways" by which to break down the poll and population data, which should always be given as factor variables with matching names and levels in both `data` and `population`. By default, this formula will also be used to construct the multilevel model, with an intercept estimated for each named stratum. See `formula` below to easily fit more complex models.

`data`
A `data.frame` representing a survey, containing (at least) the named variables in `formula.cell`. Those variables should be `factor`s. The LHS response is expected to be dichotomous, and will be coerced to binary-logical (if factor, 1 for 'yes', 0 for 'no').

`population`
A `data.frame` containing population (e.g. census) data with variable names and factor levels matching those in `data` and specified in `cell.formula`. *Note:* As in `data`, the cell formula variables should be of type `factor`.

`pop.weights`
character. The column of the `population` data that contains frequencies or proportions (of the entire population) for the cells defined in `formula`.

`grouplevel.data.frames`
A `list` of `data.frames` to be left-joined onto the data via `join`. An example is mrp.regions, which contains two columns, 'state' (the matched key) and 'region', a column being added. Multiple keys (e.g., age and education) are supported.

`grouplevel.expressions`
A `list` of `expressions` to be evaluated in the data (with the grouplevel.data.frames already joined). In the example below we construct two types: interaction terms and a linear prior mean for a factor: `expression(interaction1and2 <- interaction(term1,ter`
`expression(z.age <- rescale(age))`

`formula.model.update`
A formula specification for the multilevel model to run in the prepared data. The left-hand side should always be 'response'. For convenience, the formula is handled by `update.formula` so that `.` indicates the current formula contents on either side of the `~`, e.g., `.~.+newVar`. The initial default formula is constructed as just an intercept term for each of the variables in the main formula specification (`(1|way1)+(1|way2)` etc.)

`formula.pop.update`
Any modifications to be made to the `formula.cell` above. In the example below, we control for poll, but the population is the same for all values of *poll*. If used, should be of the `update.formula` template form, `.~.-var`.

This replicates the population data for all levels of the variable 'excluded' in this fashion. *Note:* This argument **will change**, to `constant.population.dimensions="chr"`.

`poll.weights`
Name of variable of the survey weights for respondents in the poll. This is used to compute the effective N, weighted $\bar{Y}$, and Design Effect. Default is to make all weights equal.

Ideally, the dimensions specified by `formula.cell` would account for all of the variation of survey weights in respondents in all cells. Sometimes, survey researchers design samples that leave some variation in the design weights of poststratification cells. Using `poll.weights` inflates or deflates the effective N of respondents in poststratification cells based on the average variance of design weights in all cells and each cell's deviation from that overall design effect.

If multiple polls are included and contain poll.weights, they must be normalized within each poll before *mrp* attempts to normalize the weights across all polls for all cells.

`pop.margin`	Margin of population data on which to sum other cells. Used in modeling PartyID, but implementation is not stable.
`...`	Additional arguments to be passed to the multilevel regression `bglmer` step.

See Also

`mrp-class` for other methods on the objects produced by `mrp()`; `plotmrp` for how to plot poststratified results onto maps.

Examples

```
library(mrpdata)
library(mrp)

## Load example data.
data(CCES.complete)

## Helper datasets for other US applications of MRP:
data(spmap.states) # projected US state map
data(mrp.census)   # census with common demo strata
data(mrp.regions)  # regions data.frame with DC separate

## To ensure matching of strata between poll and population,
## both should be factors with identical names and levels.
CCES.complete <- within (CCES.complete, {
  education <- factor(education,exclude=NA)
  female <- factor(sex=="Female", labels=c("Male","Female"), exclude=NA)
  race <- factor(race,exclude=NA)
  f.race <- interaction(female,race)
})

## Poll has four levels of education, so we need to combine
## the top two levels in the census data. We'll also go ahead
## and trim it down to just the variables used here.

mrp.census <- within(mrp.census,{
    age <- factor(age,exclude=NA,labels=c("18-29","30-44","45-64","65+"))
    education[education=="postgraduate"] <- "college graduate"
    education <- factor(education,exclude=NA)
    edu <- factor(education,exclude=NA,labels=c("< High School",
                                     "High School",
                                     "Some College",
                                     "Graduated College"))
    state <- factor(state,exclude=NA)
    race <- factor(race,exclude=NA)
    f.race <- interaction(sex,race)
})
mrp.census <- na.omit(mrp.census)

## Ready to run simple mrp with poll and population:
mrp.simple <- mrp(ban.gaymarr ~ state+age+education+race,
                  data=CCES.complete,
                  population=mrp.census,
                  pop.weights="weighted2004")
print(100*poststratify(mrp.simple, ~ education+age), digits=2)
```

```
## Not run:
## Fit a fuller model, adding state-level predictors:
## This model is also used in the not-run example
## for plotting maps.
mrp.statelevel <- mrp(ban.gaymarr~
                      state+f.race+age+education,
                      data=CCES.complete,
                      population=mrp.census, pop.weights="weighted2008",
                      #population.formula= .~.-age,
                      grouplevel=list(Statelevel,
                        mrp.regions,
                        expression(age.edu <- interaction(age,education)),
                        ## an ordered factor, we use a normalized
                        ## continuous z.age as the prior mean for
                        ## varying intercepts of the 'age' groups.
                        ## That is, the prior mean for
                        ## age cat 1 of 4 (18-29) becomes (-.58)
                        expression(z.age <- rescale(age)))
                      )
## Note: the formula is expanded from the condensed version in "formula" to
##  an expanded version.
getFormula(mrp.statelevel)

## Update the mr.formula on already-prepared mrp object and re-fit:
mrp.statelevel <- mr(mrp.statelevel, .~.+(1|region)+ (1|age.edu)+
                     z.age+p.relig.full+p.kerry.full)

## Fine plot control is shown with this example in plotmrp documentation!

## End(Not run)
```

`mrp-class` *Multilevel regression and poststratification*

Description

The `mrp` class holds N-dimensional cross-classified arrays for a survey and a population, a 2-dimensional summary of survey data used to fit a multilevel model, and a fitted model. Methods described here operate on `mrp` objects which are typically created by calling the `mrp` function.

Slots

`data`**:** the `data.frame` used in the multilevel regression step. This is created by summarizing the NWayData array in poll, and optionally joining additional columns of other predictors. **Note:** the row order of the data object, which will be the row order of the fitted values, **must be preserved** because it corresponds to the NWayData arrays used in poststratification. To add new columns onto the data.frame in the `mrp@data` slot, take care to preserve this ordering. Base `merge()` will almost invariably permute it in unpredictable ways: **use** `join` from plyr instead. This is done automatically when data.frames are joined by the add argument to `mrp`.

`formula`**:** The formula used in the multilevel regression. The left-hand side is always 'response'.

`multilevelModel`**:** The multilevel regression model (class `mer` created by `glmer`)

`outcome`**:** character. The name of the outcome variable.

`poll`**:** An N-way array (`NWayData-class`) constructed from a survey.

`population`**:** The population distribution, also of `NWayData-class`, with dimensions matching those of `poll`. This is intended to be a probability mass table (all entries sum to 1), but the package will also work with unnormalized population distributions. The population is not used for the multilevel regression step. It is used only in poststratification.

Methods

mr `signature(object = "mrp")`: update the formula and re-run the multilevel regression in the object.

poststratify `signature(object = "mrp")`: perform poststratification using the fitted values of the multilevel model slot and the population frequencies contained in the population slot. When used with a `poststratification.specification` (formula, character, or logical; see above), gives values in each of the given cells of the N-way array.

getThetaHat `signature(object = "mrp")`: estimates from the multilevel model in an N-way array corresponding to the dimension of the NWayData contained in slot `poll`.

getEstimates alias for `getThetaHat`

setPopulation `signature(object = "mrp")`: set population frequency data used for poststratification. This method checks that the dimension and names of the array correspond to those of slot `poll`.

setPopulationOnes `signature(object = "mrp")`: To display model results without adjusting to a population frequency table, all dimensions can be set to 1.

getNumberWays `signature(object = "mrp")`: returns the dimension of both `poll` and `population`. When a “real” population array is not set (all=1), value for pop shows 0 here for.

getFormula `signature(object = "mrp")`: returns the formula used to fit the multilevel regression.

setFormula `signature(object = "mrp")`: set the formula; but you really should use the `mr` method instead.

getPopulation `signature(object = "mrp")`: returns the `NWayData` population array.

getData `signature(object = "mrp")`: return the `data.frame` used in the multilevel regression step.

getResponse `signature(object = "mrp")`: returns a matrix containing the two-column “yes/no” response used to fit the multilevel regression.

See Also

`mrp`, which produces `mrp` objects.

`NWayData-class` *N-Dimensional Arrays for Multilevel Regression and Poststratification*

Description

Arrays used in multilevel regression and poststratification are low-dimensional (usually 2 or 3) slices of survey respondents or population frequencies. For the multilevel regression step, dimensions of such arrays are then summarized using methods here taking into account survey weights and associated design effects.

Slots

.Data An array.

type Character, either 'poll' or 'pop' indicating the origin of the data contained in the array.

levels When the array is collapsed back to a two-dimensional form, array dimension labels become character and any ordering of the original factor levels is lost. To restore level names and order, those attributes (in a `list`) are stored here.

Methods

getNumberWays

getYbarWeighted

getDesignEffect

getN

getNEffective

getDesignEffectByCell

Author(s)

Andrew Gelman <gelman@stat.columbia.edu>, Daniel Lee <bearlee@alum.mit.edu>, Yu-Sung Su <ys463@columbia.edu>, Michael Malecki <malecki@wustl.edu>

See Also

`mrp-class`

`plotmrp` *Plot poststratified estimates onto sp map objects*

Description

Poststratify and plot a fitted `mrp-class` object onto a SpatialPolygonsDataFrame. Note that package *mrp* masks the *sp* function `panel.polygonsplot`: in particular it plots `NA` values in a color, and enables a custom stroke on selected polygons in the map.

Usage

```
## S4 method for signature 'mrp'
spplot(obj, formula,
                        spmap=spmap.states, FID="STATE",
                        exclude=NULL, stroke=NULL, subset=TRUE,
                        at, cuts=63, pretty=FALSE,
                        center=0.5, between=list(x=0.25, y=0.25),
                        add.settings, ...)
```

Arguments

`obj`	A `mrp-class` object.
`formula`	The desired poststratification, where the left-hand side indicates the name of the variable in the `mrp` data that identifies the polygons to plot. In the special case where the desired poststratification is *only* by geographic unit, place it on both sides of the equation (e.g., `state ~ state`).
`spmap`	A map of class `SpatialPolygonsDataFrame`, such as is produced by `readShapeSpatial` for reading ESRI "shapefiles." This package includes a US map called spmap.states, which is the default here.
`FID`	The name of the column in the `spmap` to use as unique "feature IDs", whose values to the left-hand-side of `formula`. In the included spmap.states map, the column with two-letter state abbreviations is called 'STATE', which is the default here.
`exclude`	A vector of feature IDs to exclude from the plot. *Note:* any rows not present in the `mrp` data itself will automatically be excluded.
`stroke`	A list of feature IDs (**character**, **logical**, or **integer** with respect to the order of `spmap`) to plot with a distinct stroke; or an **expression** to evaluate in the `mrp@data`. Expressions allow data pertaining to geographic units, joined via the add argument to `mrp` to be used in plotting. The type, width, and color of the stroke(s) are taken from the *superpose.line* list, which can be customized as described below. In the event that a feature ID appears multiple times in the list, the last value is used; e.g., if it is in both the first and second vectors of the 'stroke' list, its 'type' will be `superpose.line$lty[2]`.
`add.settings`	A list of lattice graphical parameters, suitable for `trellis.par.set()`. These will override the MRP defaults, which bear mention for two reasons: 1) top and left strip titles are drawn only in panels on the top and left of the full lattice. 2) any `NA`-valued parts of any panel will be filled in the color given by 'reference.line$col', which in our theme defaults to a 68% opaque black, or `"#00000044"`.
`subset`	Expression that evalues to a logical or integer indexing vector, evaluated in the `melt`ed poststratified results. N.B. because of partial argument matching, to place a subheading blow the plot using the argument 'sub' you have to include `subset=TRUE`.
`at`	See levelplot. If a vector of values is provided to `at`, it overrides `cuts`, `pretty`, and `center`.
`cuts`	See levelplot. If using a `RColorBrewer` palette, should be one less than the number of colors in the call to `brewer.pal`.
`pretty`	See levelplot
`center`	Value at which to center the calculation of limits. Default=0.5
`between`	Documented for `xyplot`. A named list giving space between panels (in character heights). Default is 0.25 for both x and y.
`...`	further arguments to be passed to `spplot` which in turn passes them to `levelplot`.

Value

An object of class `"trellis"`. See `xyplot` for details.

See Also

`spplot` for the parent function of the `spplot,mrp-method`; `spplot` in turn extends `levelplot`.

Several packages produce gradients of colors which can be supplied to `spplot` as `regions=list(col=...)` in the 'add.settings' list. Among them are RColorBrewer, `colorRamps`, and `colorspace`, along with the basic grDevices palettes such as `heat.colors` and the function `colorRampPalette`.

A wrapper for all these nice palettes (including spline interpolation for the Brewer palettes) is in package *fBasics*; it is used in the example below

Examples

```
## plot the example from mrp()
example(mrp)
## Dimension of plot are by order of terms -- y,x so
## this will plot age along x and poll along y.
spplot(mrp.simple, formula=state~ age,
       spmap=spmap.states, FID="STATE",
   exclude=c("AK","DC","HI"),
       stroke=list(c("IA","NH","VT","CT","MA","ME"), "CA"),
       colorkey=list(
         space="bottom",height=.5,width=.5)
       )
## Not run:
spplot(mrp.statelevel, state ~ age+edu, cuts=50,
       subset=TRUE,
       spmap.states, "STATE", exclude=c("AK","DC","HI"),
       stroke=list(expression(hasmarriage2010==TRUE),
         "CA"),
       sub=paste("National average:",
         format(poststratify(mrp.statelevel),digits=2)),
       add.settings=list(
         regions=list(col=fBasics:::divPalette(51,"BrBG"))
         ),
       colorkey=list(
         space="bottom",height=.5,width=.5,
         labels=list(at=c(.1,.5,.9),labels=c("10
         )
       )



## End(Not run)
```

`poststratify` *Poststratification method*

Description

Poststratify multilevel regression model by an arbitrary number of strata or "ways". By default this method returns a single poststratified predicted value.

Usage

```
  ## S4 method for signature 'mrp'
poststratify(object, formula)
  ## S4 method for signature 'NWayData'
poststratify(object, formula, fun, population)
  ## S4 method for signature 'jagsNWayData'
poststratify(object, formula = NULL, fun = mean, population =
                 Census.NWay)
```

Arguments

`object`	An `mrp`, `NWayData`, or `jagsNWayData` object.
`formula`	A formula representation of the desired poststratification. The formula is `NULL` on the left-hand side and right-hand side variable names corresponding to the "ways" in the population data by which to poststratify. The right-hand side can also be a character vector of such names or a logical vector of length "ways". See example in `mrp`.
`fun`	The function (default=*mean*) to summarize the collapsed dimensions.
`population`	An array or `NWayData` with dimensions matching `object`, used to produce population-weighted estimates from `jagsNWayData`.

See Also

`mrp-class` for an example. `mrp-class` for other methods on the objects produced by `mrp()`; `plotmrp` for how to plot poststratified results onto maps.

`serialPaste` *serial paste*

Description

Function to paste together a list of items, separated by commas (if more than 2), and with the last one having the collapse string.

Usage

```
serialPaste(x, collapse = "and")
```

Arguments

`x`	vector or list
`collapse`	default="and"

xval — *Run k-fold cross validations on mrp model*

Description

Run a (binomial) multilevel regression in survey data for later poststratification.

Usage

```
    ## S4 method for signature 'mrp'
xval(object,
        formula, folds, loss.type,
    ...)
```

Arguments

`object`	A `mrp` object.
`formula`	A formula specification for the multilevel model on which the k-folds cross validation is run. The default is the formula of the `mrp` object. The left-hand side should always be '`response`'. For convenience, the formula is handled by `update.formula` so that `.` indicates the current formula contents on either side of the `~`, e.g., `.~.+newVar`. The initial default formula is constructed as just an intercept term for each of the variables in the main formula specification (`(1|way1)+(1|way2)` etc.)
`folds`	Number of folds for cross valiation. Default is 4.
`loss.type`	Type of loss measure used in cross validation. Currently "logloss" is supported.
`...`	Additional arguments to be passed to the multilevel regression step, which uses `bglmer` by default.

Index