Creating Models

David J Dorer

Creating Models

Model Components

Models are lists with 4 components:

1. name: A name for the model without spaces.
2. variables: A named list of functions that can be used to compute derived PUMS variables. By convention the function name ends in ".v" and the list element name does not include the ".v" This convention is used to provide conceptional clarity and to avoid name collisions when defining the variable and marginal table functions.
3. marginal.tables: A named list of marginal tables. By convention the marginal table function names end in ".m" and the name of the list element does not have the ".m"
4. parameters: A named list with parameters. Required:

   4a. model: the model name from the "name" component.

   4b. model.type: Values "person" or "house." A "person" model.type means that the PUMS variables and model table are based on the PUMS person level data. A "house" model.type uses the PUMS house level data.

   4c. geotype: A value of "tract" or "blockgroup" A "tract" geotype means that synthetic data will be produced for census tracts. A "blockgroup" geotype means that block group level synthetic data will be produced.

   4d. nraces: Number of race categories for variables and marginal tables that use race. See help(PAT.race.code).

   4e. nages: Number of age categories for variables and marginal tables that use age. See help(PAT.age.code).

   4f. comment: A comment to be used for outputs.

   4g. ... Optional user defined parameters to be passed to the variable functions and marginal table functions.

Model Component variables

Before the model list can be created the variable functions and marginal table functions must be defined (created) so that they can be used. A simple example illustrates this process:

```
Sex.v<-function(SEX) {
   sex<-as.numeric(as.character(SEX))
   factor(sex,levels=c(1,2),labels=c("Male","Female"))
};
```

A few things to notice.

1. the functions is named "Sex.v" To keep things straight use the convention that the function starts with a capital letter and ends with a ".v"
2. SEX is a PUMS person level variable. Review the PUMS codebook, e.g. PUMS documentation page
3. The return value of the Sex.v function is a factor variable. When PUMS data is downloaded via the API (Application Programming Interface) the resulting R data.frame has character variable. To be sure we use "as.character" in our code.
4. The derived variable in the PUMS data.frame will be "Sex" the name of the model list component "variables."
5. To keep things organized the factor labels should the following convention:

   5a. The factor label starts with a capitalized letter (or a numeral) and then uses lower case.

   4b. The factor labels use "_" not " " so that there are no blanks in the factor levels. This will save a lot of headaches later.
6. The variable function can optionally use arguments from the defining model list. For example when you are using AGEP as one of your arguments, you might want to add an "nages" argument. You can use any additional parameters that you might like as long as your parameter appears in the model "parameters" list component.
7. When the PUMS data is downloaded the variables that you have defined will be added to the PUMS data.frame with the name of the model "variables" list component. This is why when you construct the model "list" you use the convention that the function used to define the PUMS "Sex" variable is named "Sex.v".

Model Component marginal.tables

1. Like the variable name/function definition the convention that is used should use the marginal table name correspond to the marginal table function except that the marginal table function ends in ".m". For example you might like to have a marginal table that corresponds to the model PUMS variable function "Sex." That marginal table function should be a function named "Sex.m".
2. An improtant point using Iterative Proportinal Fitting is to adjust a PUMS table/cross tabulation at the PUMA geography to an ACS "B", "S", and "DP" table at the tract or block group geography. To do the adjustment the marginal table dimnames variable names need to correspond to model PUMS variable names. Also the marginal table "dimnames" need to have PUMS variable as names and the dimnames of the marginal table categories (labels component of the factor variable) that agree exactly with the PUMS variables factor labels.

Here is an example of a marginal table function:

```
Sex.m<-function (state = "25", puma = "", county = "021", tract = "400100",
        blockgroup = "", csd = "", place = "", vintage = PAT.vintage(),
        key = PAT.census.key(), period = PAT.period(), download = PAT.download(),
        debug = PAT.verbose())
      {
        gat <- PAT.acs.table("B01001", vintage = vintage, period = period,
               state = state, county = county, tract = tract, blockgroup = blockgroup,
               puma = puma, csd = csd, place = place, download = download,
               debug = debug);
       dat <- gat$data;
       rownames(dat) <- toupper(rownames(dat));
       dat <- dat[c("MALE", "FEMALE"), c("Est", "MoE")];
       dimanmes(dat) <- list(Sex = c("Male", "Female"), stat = c("Est",  "MoE"));
       dat
};
```

Try the function on a PUMA (this is what PAT.test.model does.)

```
Sex.m(state=25,puma="03400");
         stat
Sex        Est  MoE
  Male   71006 1164
  Female 80261 1161
```

A few things to notice about the Sex.m return value.

1. The rownames are the categories are the same as the value of Sex.v()
2. The names of the dimnames are "Sex" and "stat." "Sex" is the same as the name in variables list component in the defining model (not shown). The "stat" part of the dimnames is c("Est","MoE"). These correspond to the column names of an ACS ("B", "S" or "DP") table downloaded via the API.

Testing Your Model

The function PAT.test.model will test your model by running your model on a PUMA, the state and puma FIPS codes are arguments.

PAT.test.model downloads the PUMS data for the selected geography. Next PAT.test.model downloads your defined marginal tables for the same PUMA geography. When you are synthizing data at the tract level, the PAT.synth.data function will use tract level geography for marginal tables.

For the testing phase we want to check that PUMA marginal tables are close to the corresponding PUMS variable cross tabulations. The marginal tables will not match exactly with the PUMS cross tabulation as the PUMS data is a sub-sample of the ACS data records that go into the marginal tables. If there are inconsistencies in variable names or differences in factor labels between the PUMS variables and the marginal tables, these errors will show up at this testing step.

If PAT.test.model produces errors. Go back and revise your PUMS variable definitions and marginal table computations. Also if the PUMA marginal tables are too different from the PUMS data table cross tabulations, you should be suspicious and go back and check your model definition functions. This is a matter of judgement because of the sampling used in constructing the PUMS data sets and the PAT.test.model step will not produce an exact agreement.

7 Dec 2023 15:26