EES2019 Stacking Process

Synthetic Variables Estimation

Giuseppe Carteny

20.09.2021

Contents

1 Introduction

Our next tasks consist in (1) the estimation of generic **synthetic** variables and (2) the preparation of the SDM codebook.

Again, the workflow follows up the one used for our previous task: we will modify the 'genvars' country-specific scripts developed during the last weeks, adding the computation of the synthetic variables.

2 Estimating Synthetic Variables and Codebook Preparation

Synthetic variables consists in variables measuring the affinity between a set of individual characteristics and the set relevant parties identified in each political context. These variables are determined by (1) taking a set of independent variables and using them in a series of regression analyses that links these variables to

each relevant party in turn, and then (2) estimating the linear predictions of said regression analyses. For this reason, such variables are often called '**y-hats**'.

As mentioned earlier during our talks, the reason why such variables are estimated is because many variables of interest for analysing voting behaviour do not directly concern the relationships investigated with an SDM, namely the voter-party relationships. Individual features such as individuals' age, socio-economic status, or religiosity, are individual characteristics that do not vary across the pivotal voter-party dyadic relationships of the SDM approach. Consequently, by computing said y-hats we try to link said characteristics to these relationships of interest.

2.1 The `gensyn.fun` function

As for previous generic variables, I implemented an *ad-hoc* function for estimating the synthetic variables briefly presented few lines above.

In short, the function (1) selects the dependent and independent variables of interest from the current SDM, and then (2) creates a set of data frames for each regression model. For instance, in a situation like the Italian EES 2019 voter study, with 1000 respondents, and 7 relevant parties (thus, an SDM with 7000 dyadic relationships), a generic dependent variable and a set of categorical and/or continuous predictors, the function would first create 7 data frames.
Then, (3) it will estimate a *linear* regression model for each data frame, and then (4) it will return the *linear* predictions of such models. Thus, in our example it would first compute 7 regression models, and then it would return the linear predictions for said models.
In the end (5) the function will stack the predictions of each model, creating thus an SDM with three columns: the respondent identification code (`respid`), the party code (`party`), and the syntehtic variable linked to each dyadic relationship. In our fictional case, the output of the function would be a three columns SDM with 7000 observations.

The arguments of the function are the following:

. `data` = The main data basis for computing our variables. Essentially it takes only one value, that is the 'EES2019' (country-specific) **stacked** data frame;

. `depvar`= The dependent variable of our regression models. Since the EES voter study provides a continuous (namely, the propensity to vote variable) and a dichotomous dependent variable (namely, the vote choice variable), this argument essentially takes two values, that refer to the generic variables `Q10_gen` (PTV, continuous) and `Q7_gen` (vote choice, dichotomous). By specifying the dependent variable the function will automatically estimate an OLS model (for the PTV variable) or a logit model (for the vote choice variable), and eventually compute the predicted values (see the `regsum` argument below);

. `cat.indvar` = Refers to the **categorical** independent variables of our regression models. Thus, you must supply a string vector containing the names of said independent variables;

. `cont.indvar` = Refers to the **continuous** independent variables of our regression models. Also in this case, you must specify a string vector containing the names of said independent variables;

. `yhat.name` = The name of our synthetic variable. The function will take said name as a string vector and then will complete it with a suffix conditional on the dependent variable specified ("`_ptv`" if `depvar` is the

PTV variable, or "_vc" if depvar is the vote choice variable);

. regsum = Consists in a logical argument that will return the prediction if set to FALSE. If set on TRUE it will return a list object containing the regression models output, in order to check the regression analyses.

2.2 Procedure for computing synthetic variables

The procedure for computing our distance variables is rather similar (in terms of code) to those used previously for estimating the other generic variables. For implementing your scripts you just have to run first the 'EES2019_stack.R' without the last section ('Estimate the generic variables'), and then run your country-specific 'genvars.R' script, without the last section (that is, without 'tidying' the environment).

In our routine we will compute **two synthetic variables** for each set of predictors, one using the PTV (Q10_gen) variable and one for the vote choice (Q7_gen) variable as dependent variables. Consequently, the two synthetic variables will be the linear predictions of an OLS model (for the PTV variable) and a logit model (for the vote choice variable). The resulting predictions will be, thus, a linear prediction on the same scale of the dependent variable for the OLS model, and a log-odds value for the logit[1].

If you want to estimate the generic variable and check the results, you can just run the function, setting to TRUE the regsum argument, and save the output (regression outputs) in an object. Then you can access the list entries one by, one or all together with another apply family function, to check the model summary or more specific statistics (such as the variance inflaction factor, aka 'VIF', for checking predictors collinearity), as in the example below.

```
fit_lst <-
  gensyn.fun(data = EES2019_it_stack,
             depvar = 'Q10_gen',
             cat.indvar =  c('D3_rec', 'D8_rec',  'D5_rec', 'EDU_rec', 'D6_une'),
             cont.indvar =  c('D4_age', 'D10_rec'),
             yhat.name = 'socdem',
             regsum = T)

# e.g. #

fit_lst[[3]] %>% summary
```

```
##
## Call:
## lm(formula = x$frml, data = x$data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

[1]The reason why we do not compute the predicted probabilities for the logit models is because for most of the analyses in which synthetic variables are used such variables are centered. In the case of logit models' predictions is done on the log-odds, that only subsequently are transformed in predicted probabilities.

```
## -0.66031 -0.37121 -0.00165  0.36138  0.71708
##
## Coefficients:
##               Estimate Std. Error t value             Pr(>|t|)
## (Intercept)  0.5744075  0.0668463   8.593 < 0.0000000000000002 ***
## D3_rec2      0.0088228  0.0259704   0.340               0.7341
## D8_rec1     -0.0473456  0.0278735  -1.699               0.0897 .
## D5_rec1      0.0122108  0.0287932   0.424               0.6716
## EDU_rec2    -0.0955460  0.0448626  -2.130               0.0335 *
## EDU_rec3    -0.1868018  0.0463085  -4.034            0.0000596 ***
## D6_une1     -0.0547979  0.0504636  -1.086               0.2778
## D4_age      -0.0011196  0.0008405  -1.332               0.1832
## D10_rec      0.0218204  0.0055549   3.928            0.0000922 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3884 on 892 degrees of freedom
##   (99 observations deleted due to missingness)
## Multiple R-squared:  0.04366,    Adjusted R-squared:  0.03509
## F-statistic: 5.091 on 8 and 892 DF,  p-value: 0.000003264
```

```
fit_lst[[3]] %>% car::vif(.)
```

```
##             GVIF Df GVIF^(1/(2*Df))
## D3_rec  1.006237  1        1.003114
## D8_rec  1.007089  1        1.003538
## D5_rec  1.080578  1        1.039508
## EDU_rec 1.029725  2        1.007350
## D6_une  1.089658  1        1.043867
## D4_age  1.091334  1        1.044669
## D10_rec 1.032862  1        1.016298
```

About regression diagnostic, one thing must be said: it is *very, very, very* unlikely that the regression models that we are estimating will satisfy most of the OLS or logit models assumptions. For instance, talking about OLS, it is very unlikely that the regression models that we will estimate will show normally distributed residuals, or linear residuals' patterns. All the models will suffer from model specification issues. These models are clearly omitting many relevant variables, and in most cases they will include irrelevant variables. Nonetheless, we actually don't care that much about such violations, at least at this stage. What we care the most about is that the regression models converge (especially logit models) and that there's no collinearity among the predictors.

In any case, checking model assumptions is definitely not worthless exercise, thus I encourage you to take a look at regression diagnostic results, and add some notes to your code about said results if you consider them particularly relevant.

After checking all the issues mentioned few lines above, and if everything works (more or less) fine, you can develop an additional paragraph just below the one dedicated to the generic distance/proximity variables.

```
EES2019_it_stack %<>%
  left_join(.,
        lapply(data = EES2019_it_stack,
               cat.indvar =  c('D3_rec', 'D8_rec',  'D5_rec', 'EDU_rec'),
               cont.indvar =  c('D4_age', 'D10_rec'),
               yhat.name = 'socdem_synt',
               regsum = F,
               X = list('Q10_gen','Q7_gen'),
               FUN = gensyn.fun) %>%
          do.call('left_join',.),
        by = c('respid', 'party')) %>%
  as_tibble()
```

```
## Joining, by = c("respid", "party")
```

Once computed the variables we just have to tidy up the environment as already specified in the current version of our scripts.

2.3 Codebook development

Once finished the estimation of the variable(s) of interest, then the next step is to insert new entries in the SDM codebook, that you can find as a R Markdown script ('Codebook.Rmd') in the '~/EESstacked/Docs/docs_scripts' subdirectory.

About the **content** of the paragraphs concerning the **synthetic variables**: in this step we will essentially estimate the affinity between respondents' socio-demographic characteristics and their propensity to vote for one of the stack party and their vote choice. After this introduction, I suggest you to just specify whether the prediction consists in a linear prediction of an OLS or logit model, and then which are the variables used to estimate the regression models from which such scores are predicted.

About the content of the predictors, just take a look at the following paragraph.

2.4 Which Variables

The list of variables that we are going to use for computing our synthetic variables is shown below. Note that the list of predictors is, first, temporary (since more predictors will be included in the following weeks) and, second, refer only to socio-demographic characteristic of the EES2019 voter stud respondents.

Dependent variables:

- `Q7_gen`: A variable measuring whether the respondent voted or not for the stack party;
- `Q10_gen`: Respondent's propensity to vote for the stack party;

Independent variables:

- `'D3_rec'`: Respondent's gender (0 = Male, 1 = Female), recoded from the original `D3` EES2019 variable (categorical);
- `'D5_rec'`: Whether the respondent is married/remarried/single living with a partner (1) or single/divorced/separated/widowed (0), recoded from the original `D5` EES2019 variable (categorical);
- `'D8_rec'`: Whether the respondent lives in a rural (0) or urban area (1), recoded from the original `D8` variable (categorical);
- `EDU_rec`: Respondent's years of formal education (1 = 15 years or less, 2 = 16-19 years, 3 = 20+);
- `D4_age`: Respondent's age, recoded from the original `D4_1` (year of birth) EES2019 variable (ordinal treated as continuous);
- `D10_rec`: Respondent's religiosity, recoded from the original `D10` EES2019 variable (ordinal treated as continuous). In particular, the values (min = 0, max = 6) are inverted, so that higher values indicate stronger religiosity and lower values indicate low/none religiosity.

3 Who Does What

Like for the last task, I will take care of Belgium, Bulgaria, Cyprus, and Italy, while you will take care of the following countries:

- **Willie**: Denmark, Estonia, Germany, Luxembourg, Malta, Netherlands, Spain, United Kingdom;
- **Julian**: Czech Rep., Finland, Greece, Hungary, Lithuania, Slovakia, Poland, Sweden;
- **Matthias**: Austria, Croatia, France, Ireland, Latvia, Portugal, Romania, Slovenia.

This time **Willie** will fill the codebook with a set of entries referring to the synthetic variables generated from the routine described in the previous pages.

4 The Deadline

I believe that also in this case a few days should be enough to finish the tasks listed above (deadline: **27.09.2021**).