EES2019 Stacking Process

Synthetic Variables Evaluation (Pt.2)

Giuseppe Carteny

29.10.2021

Contents

1 Introduction **1**

2 Reviewing our evaluation scripts, and creating our `r-markdown` scripts **2**

2.1 Review and changes of the country-specific `synteval` scripts 2

2.2 Summarizing evaluations with an `r-markdown` script 6

3 Who Does What **9**

4 The Deadline **9**

1 Introduction

The second part of the evaluation of synthetic variables estimation consists in creating a set of summary documents that will part of an appendix to the SDM codebook. These documents will be created with `r-markdown` and related packages (e.g. `bookdown`, `knitr`, `kableExtra`) using the information collected in our previous step. For

The creation of documents summarizing synthetic variables estimation is needed for transparency and reproducibility sake. Synthetic variables are the only variables of the SDM created with procedures that bo beyond basic transformations of existing variables. Hence, we must provide a summary of the statistical models to the prospective SDM users, including workarounds for solving some issues faced in estimating our synthetic variables.

2 Reviewing our evaluation scripts, and creating our `r-markdown` scripts

The first goal of our new step is to improve our `synteval` scripts in order make them less redundant, clearer and more concise. The second goal, then, is to include them in our `r-markdown` scripts, creating a set of summary documents. The next section explains how to achieve the first goal, and the following one explains how to achieve the second one.

Before starting, however, a **cautionary note**: the workflow of our evaluation of synthetic variables estimation has been changed, and it has been changed exactly because the plan is to integrate them in our `r-markdown` scripts. For running the evaluation scripts now you'll have to run the `Synteval_gen` script in the `~/Scripts/synteval_scripts/` subdirectory. After that you'll be able to work on your country-specific scripts.

2.1 Review and changes of the country-specific `synteval` scripts

Our new task on the evaluation scripts consists in determining which information should be included and shape them in a way that will allow us to include them straighforwardly in our `r-markdown` scripts.
As you saw during the last two weeks, the existing evaluation scripts provide a lot of information. However, some of them are actually not needed for our documents. For instance, if partial models are needed just to deal with 'problematic' models, then creating partial models for *all* the regression models that have been estimated, and moreover adding their fit statistics to our summary data frames is a useless exercise. This situation was created because the evaluation procedure was conceived when the structure of the summary was still under development.
To exemplify how we should review our scripts I will make again reference to the one dedicated to the Cypriot sample (`EES2019_cy_synteval.R` in the `~/Scripts/synt_eval_scripts/country_spec_scripts/` subdirectory).

We start by running our country-specific script until the '*Syntvars evaluation: logit models fit stats*'. By doing so, we organize our country-specific data frames, compute the dichotomous and proximity/distance generic variables, source the functions, select the variables of interest, organize the variables and data frames in lists, fit the full and null models, and finally create data frames dedicated to model fit statistics.

The few differences of the current version of the script compared to the previous one are that (1) the '*Syntvars evaluation: ... models summary*' sections have been removed since the `stargazer` tables are going to be used in the `r-markdown` scripts and that (2) the '*Syntvars evaluation: logit models fit stats*' section is slightly changed, because differently from before a single data frame including fit statistics for both our full and null logit models is created.

Moreover, notice that an **additional variable** has to be included in our regression models, that is the `D6_une` variable (a dichotomous variable that has value 0 for respondents having an employment and 1 for unemployed ones). Obviously, once concluded our evaluation, this variable must be specified also in the related country-specific `genvars` script.

These steps are more or less the same that we used in our previous task and already create more information than what we might actually need in our documents. However, at this stage I would not drop anything

from the environment. Even if some of said information won't be actually used in our `r-markdown` scripts, these might return useful in any case. What we will definitely include in our summary documents (at least cccording to the current version of them) is:

1. A data frame with information concerning the relevant parties (the `relprty_df` object);
2. A set of two tables for our regression models (OLS and logistic);
3. A data frame summarizing the Akaike Information Criterion scores for the full and null models;

The relevant parties frame (1) is already formatted in a way allowing us to include it in our `r-markdown` script. As stated few lines above, the regression tables (2) can be created directly in the `r-markdown` file. What needs to be created is a data frame for the AIC scores (3) in a way that will allow `r-markdown` to format it, and we can do it just selecting the AIC scores for the full and null models and reshaping it in a wide format.

```
# OLS AIC df

ols_aic <-
  ols_df %>%
  pivot_wider(id_cols = c('depvar', 'partycode', 'partyname_eng'),
              values_from = 'AIC',
              names_from = 'model') %>%
  mutate(diff = full - null) %>%
  mutate(across(c('full', 'null', 'diff'), ~round(.,3))) %>%
  dplyr::select(-c(partyname_eng))

# Logit AIC df

logit_aic <-
  logit_df %>%
  pivot_wider(id_cols = c('depvar', 'partycode', 'partyname_eng'),
              values_from = 'AIC',
              names_from = 'model') %>%
  mutate(diff = full - null) %>%
  mutate(across(c('full', 'null', 'diff'), ~round(.,3))) %>%
  dplyr::select(-c(partyname_eng))
```

Once created said data frames, we move to the sections that need to be reviewed the most, namely those dedicated to the **estimation and evaluation of the partial models**. As mentioned earlier, during the last weeks I realised that we definitely do not need to create said constrained models (and, thus, a summary of them) for those full models showing reliable estimates. Thus we can implement said sections of our 'synteval' scripts removing some steps, and changing others.

First, after the last section dedicated to full and null models fit statistics ('*Syntvars evaluation: logit models fit stats*') I would insert a brief comment section. All of us put comments on our scripts, and this is very

good, but I would like to 'standardize' them. My idea is that the first comment section should be like the following one:

```
# Full models evaluation # ============================================================ #

# logit models 3, 5, and 6 show inflated SE on some predictors, more specifically:
# Model 3: D7_rec (only for category 2)
# Model 5: D8_rec, D5_rec, EDU_rec, D7_rec (only for category 2), D6_une
# Model 6: D6_une

# Model 3 and 6 constant terms are not affected by D7_rec and D6_une inflated SE, whereas
# Model 6  constant is affected showing unusual values. We deal only with model 6 affected
# by separation issue.
```

Then we crosstab the dependent variable with the problematic predictors. I realized that the former ad hoc function developed for this purpose was neither appropriate for our tasks nor proper for creating tables to be included in our `r-markdown` scripts. Thus I created another one, called `tab.auxfun`[1], that is more specific and creates tables that can be easily included in our `r-markdown` scripts.

This is the exemplary routine for creating our crosstabs, plus comments about them :

```
# Syntvars evaluation: evaluating the source of misfit # ============================== #

# Model 5 # - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -#

mdl  <- 5
df   <- regdf_lst$logit[[mdl]]
cols <- c('D8_rec', 'D5_rec', 'EDU_rec', 'D7_rec', 'D1_rec', 'D6_une')

tabs <- lapply(data=df, y='stack_505', na=F, X = cols, FUN = tab.auxfun)

# No respondents from rural areas, not married or in partnership, with low education,
# with high subjective social status, members of trade unions, and unemployed did vote
# for party 505 (voted by only 5 respondents of the Cypriot sample).
```

After this step, then, we implement our constrained model and we evaluate them against our full models using the LR test strategy, performed using the `anova` base function, then commenting the results:

```
# Syntvars evaluation: partial logit models # ========================================= #

# Get the df for and estimate the partial models # - - - - - - - - - - - - - - - - - - - #
```

[1]The function takes the following arguments: `data` that should be one of our standard data frames for the regression models; `y` that is our dependent variable (character); `x` the predictor of interest (character); `na` whether the function should include also missing values (logical, default `TRUE`); `perc` whether you want also the percentages values for the cells along with the absolute frequencies (logical, default `FALSE`); `which_perc` if included, whether the percentages should be computed considering `all` the cells, `rows` or `columns` (character, default `all`).

```
vrbls_2_drop <- c('D5_rec', 'D8_rec', 'EDU_rec', 'D1_rec', 'D7_rec', 'D6_une')

regdf_lst_part <-
  regdf_lst$logit %>%
  lapply(., function(x){
    x %<>% na.omit() %>% dplyr::select(-c(all_of(vrbls_2_drop)))
    })

partmod_lst <-
  lapply(regdf_lst_part, function(x){
    y    <- names(x)[startsWith(names(x), 'stack')]
    xs   <- names(x)[3:length(x)]
    frml <- paste(y, paste0(xs, collapse = ' + '), sep = " ~ ") %>% as.formula

    fit <- glm(data = x, formula = frml, family = binomial)

    return(fit)
  })

# LR test (Chisq) # - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -#

mdls <- c(5)

anova_lst <-
  anova.auxfun(mdl_lst1 = partmod_lst[c(mdls)],
               mdl_lst2 = fullmod_lst$logit[c(mdls)],
               table = F)

# According to the LR test for Model 5 we cannot reject H0.
```

After this passage we include the partial model in a final list of models of interest, updating also the AIC data frame referring to our logit models :

```
# Syntvars evaluation: logit models summary # ========================================== #


finalmod_lst <- list()
finalmod_lst[['OLS']] <- fullmod_lst[['OLS']]
finalmod_lst[['logit']] <- fullmod_lst[['logit']]

finalmod_lst[['logit']][[7]] <- finalmod_lst[['logit']][[6]]
finalmod_lst[['logit']][[6]] <- partmod_lst[[mdls]]
```

```
# Syntvars evaluation: Updating AIC data frames (logit only) # ========================= #

# logit AIC df

logit_aic %<>%
  rbind(.,
        tibble('depvar'    = 'stack_505',
               'partycode' = 505,
               'full'      = partmod_lst[[mdls]] %>% AIC,
               'null'      = nullmod_lst$logit[[mdls]] %>% AIC,
               ) %>%
          mutate(diff = full-null)) %>%
  .[order(.$depvar, .$partycode),]
```

Again, once finished the evaluation, we must modify our 'genvars' country-specific scripts accordingly. Remember that we have a new variable to be included, thus we need to modify such scripts even if no issues are found. The following step, then, is dedicated to the creation of the `r-markdown` scripts.

2.2 Summarizing evaluations with an `r-markdown` script

For creating our `r-markdown` scripts we move then to the `~/Docs/Synteval/scripts` subdirectory. Here we can find the exemplary script `EES2019_synteval_gc.Rmd`, with '`gc`' referring to my name and surname first letters. This script *is not* country specific, but rather is conceived for including a summary of the evaluation of synthetic variables estimation of all the countries that each of us analysed during the last few weeks. This will allow us, later on, to create a single `r-markdown` script formatted as a pdf document. Now let's turn to the script.

First, the `YAML` header. You can copy-paste it from the exemplary script, and then modify the `title`, `subtitle`, `author`, and `date` arguments.

Second, there's the first chunk of code that will source the `Synteval_gen` script, after loading the only package required for doing so (that is `here`)[2].

````
```{r echo=FALSE, warning=FALSE, message=FALSE}

# Load 'here' for sourcing # - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - #
library('here')

# Source the general workflow # - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -#
source(here('Scripts', 'synteval_scripts', 'Synteval_gen.R'))

```
````

[2]We load this package because, as you might now, everytime we knit a document with `r-markdown` we launch an `R` session.

After these two chunks then we have the sections dedicated to our country-specific evaluations. After the title (that can be simply the name of the country of interest) we include a second chunk of `R` code that sources the country-specific `synteval` script.

```
```{r echo=FALSE, warning=FALSE, message=FALSE}
# Source the Cypriot synteval script # - - - - - - - - - - - - - - - - - - - - - - - - - - - - #

source(here('Scripts', 'synteval_scripts', 'country_spec_scripts', 'EES2019_cy_synteval.R'))

```
```

At this point we are ready to start typing and formatting the info of interest in our script.

In terms of content, I would like to maintain the documents as concise and clear as possible. Thus after a first introductory paragraph, and after printing the relevant parties table, I would simply mention whether the regression models did produce unusual results, the min and max R^2 (or pseudo R^2) values[3], and then comment differences in the AIC scores between full and null models, printing then the related table. Of course, if we encountered problems with our models, then we need to mention them first when summarising the regression output information, and then dedicating some lines and tables to show how we dealt with such issues. Overall, I would rely on the comments that we already put in the `synteval` scripts. We don't need to explain the strategy everytime. I will explain it when I will write down the introductory section of the appendix.

For formatting tables we can use the functions provided by `knitr` and `kableExtra` packages, whereas for formatting our regression tables we can use `stargazer`.
This is an example of the code for formatting a table:

```
```{r echo=FALSE, warning=FALSE, message=FALSE}
options(knitr.kable.NA = '')
options(knitr.table.format = "latex")

names(relprty_df) <- c('Dep. Var.', 'Party', 'Party name (eng)')

relprty_df %>%
  kable(caption = "Cypriot relevant parties \\label{table:relprty_tab_cy}",
        booktabs = T,
        align = c('l', 'c', 'l')) %>%
  kable_styling(latex_options = c('striped', 'hold_position'))

```
```

The first two lines of the chunk are general options for knitr (namely, what to print in the table cells when we have `NA` values, and that the tables needs to be formatted for a pdf document). Then we rename the columns

[3] As you will notice, in the script I use `r` inline chunks for reporting such values. I think that this is a convenient way to do it because reporting manually such values is prone to human error, but this is up to you.

of our data frames and finally we print them. Notice the `caption` argument of the `kable` function: we label the table by adding `\\label{...}` in the string vector. This will allow us to reference the table in directly into the document. Moreover, be aware that the labels *must* be country-specific (e.g. `table:full_logit_cy`) while the related `R` objects do not need to. Then, the remaining options allow us to style a bit our table.

All the tables are formatted in a similar fashion. Nonetheless, if we dealt with 'problematic' regression models, then we must insert extra information in our tables. For instance, in the Cypriot case the AIC values table for the logistic regression models now includes also the AIC values of the partial model. Hence, we must specify in our table that the column referring to full models' AIC values includes one value that actually referes to a constrained model. This can be easily achieved adjusting the AIC data frame and adding a footnote to our table, as shown below:

```
```{r echo=FALSE, warning=FALSE, message=FALSE}

logit_aic[6,1] <- 'stack_505*'

names(logit_aic) <- c('Dep. Var.', 'Party', 'Full Mod.', 'Null Mod.', 'Diff. (Full-Null)')

logit_aic %>%
  kable(caption = "Akaike Information Criterion values for logistic full and null models
        \\label{table:logit_aic_cy}", booktabs = T,
        align = c('l', 'c', rep('r',3))) %>%
  kable_styling(latex_options = c('striped', 'hold_position')) %>%
  footnote(symbol = 'AIC value refers to Model 11b (constrained).',
           threeparttable = T,
           footnote_as_chunk = T)
```
```

Turning to the regression tables formatted with `stargazer`, the logit models tables require some adjustements.`stargazer` auomatically assigns model names that are simply based on the list of model results that we process with the function (that is, if we supply a list of 3 model results, `stargazer` will label them 'Model 1', 'Model 2', and 'Model 3'). This is very convenient for our OLS models, but when we turn to the logistic ones we need to rename them. Unfortunately, despite its flexibility, `stargazer` does not allow us to modify such names with a function argument. So we need a workaround. First we assign our `stargazer` table to an object. This will create a character vector, whose entries refer to the LaTeX strings created by `stargazer`. Then you can modify it, changing the names of the model with the `gsub` function. After doing so you can finally "print" the table using the `cat` function.

```
```{r, results='asis', echo=F}

logit_regtab <-
  stargazer::stargazer(finalmod_lst$logit,
                       title = "Vote choice for a relevant party according to respondents'
                       socio-demographic characteristics (Logistic regression models)",
```
```

```
                        label = 'table:full_logit_cy',
                        type = 'latex',
                        column.labels = c('501', '502', '503', '504','505','505', '507'),
                        dep.var.labels.include = F,
                        star.cutoffs = c(0.05, 0.01, 0.001),
                        omit.stat=c("f", "ser"),
                        header = F,
                        style = 'ajps') %>%
  capture.output()

logit_regtab[9] %<>%
  gsub('Model 7', 'Model 12',.) %>%
  gsub('Model 1$', 'Model 7',.) %>%
  gsub('Model 6', 'Model 11b',.) %>%
  gsub('Model 5', 'Model 11a',.) %>%
  gsub('Model 4', 'Model 10',.) %>%
  gsub('Model 3', 'Model 9',.) %>%
  gsub('Model 2', 'Model 8',.)

cat(logit_regtab, sep = "\n")


```

This is it. If you spot any error, typo, or if you have any additional idea please just drop me an email as usual!

3 Who Does What

The list of countries on which we will work is not changed. 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.

4 The Deadline

I believe that we will need a bit more than one week of work, especially because you'll need to fight with R Markdown and all its dependencies. So the next deadline is **12.11.2021**.