Documentation of the The Swiss case: Reactions of the real economy to the discontinuation of the exchange rate floor in 2015. A synthetic control approach project

Martin Dohmen

16 March 2018

CONTENTS

CHAPTER

ONE

INTRODUCTION

This project estimates the real consequences of the Swiss appreciation from January 2015 using the synthetic control method. The idea is to construct a synthetic Switzerland as a control unit, which describes how Switzerland would have developed in absence of a large appreciation. The difference in development after the decision of the SNB to unpeg the franc between the actual and the synthetic Switzerland is the treatment effect.

The method is based on Abadie et al. (2010, 2014). *[1] [2]* On the basis of pre-treatment data a weighted average of control countries is constructed to work as a synthetic control for the treated country. The method is described shortly below.

The algorithm used to calculate the synthetic control is based on Becker and Klößner (2018) *[3]*. The implementation and algorithm suggested by them is explained briefly below.

The project is based on the Templates for Reproducible Research Projects in Economics by von Gaudecker. *[4]*

1.1 Structure

The project is structured in different folders. Every folder contains files with dictinct functions. The folders are:

- original_data
- data_management
- analysis
- final
- paper
- model_code
- model_specs

Following the folders, the project is organised in different steps:

1. data management: take the raw data from **original_data** and clean it and bring it in the structure we need (second normal form)
2. analysis: Taking the functions defining the "model", in my case implementing the algorithm of Becker and Klößner (2018) *[3]* from **model_code**, and the specifications defined for the models in **model_specs**, the main analysis is done. The synthetic control unit and all necessary values are calculated and the results are stored.

3. final: In this step the results from the analysis step are visualised. This means tables and figures are constructed.
4. paper: This is the last step. Figures and tables are incorporated and the paper is compiled.

Detailed discribtions of what exactly every step and file is doing can be found in the following chapters.

The project runs 6 different specifications contained in **model_specs**. These are:

1. a baseline specification for GDPPC: outcome is GDPPC and predictors are yearly averages of the outcome for 2008, 2010, 2012, 2013, 2014
2. a all-year averages specification for GDPPC: outcome is GDPPC and predictors are yearly averages of the outcome for 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014
3. a with-covariates specification: outcome is GDPPC and predictors are an averages of the outcome for 2012, 2013, 2014 and other covariates: trade opennes, inflation, industry share and schooling
4. a current-account specification: outcome is CA as % of GDP and predictors are an averages of the outcome for 2012, 2013, 2014 and other covariates: trade opennes, inflation, industry share and schooling
5. a time placebo specification: as baseline, but treatment time changes to 2005Q1 and accordingly sample period and predictor years are ten years earlier
6. a country-placebo specification: as baseline, but treated country changes to Australia and Switzerland is excluded

For every specification the analysis is run and tables and figures are produced.

1.2 The Synthetic Control Method

In this section I want to give a short description of the synthetic control method. It follows mainly Abadie (2010) *[1]* . For details please refer to this paper and the ones cited above.

The synthetic control method is a data-driven approach to construct a suitable control unit in comparative case studies. The idea is that a combination of control units will often provide a better comparison for the treated unit than a single control unit alone. Therefore, the synthetic control is constructed as a weighted average of so called donors, a sample of suitable control units, with non-negative weights that sum to one. These weights, collected in a vector W, are calculated by minimizing the difference of selected predictor variables between the treated unit and the synthetic control. The predictor variables can be linear combinations of the outcome variable prior to treatment as well as other covariates with explanatory power for the outcome of interest, which need to be measured prior to or unaffected by the treatment. In the optimization the predictors are weighted by a predictor weighting matrix V, which is chosen to result in optimal weights W that yield the lowest possible RMSPE between the outcome of the treated unit and the synthetic control prior to treatment. This structure leads to a nested optimization problem. In the inner optimization the optimal donor weights W are determined, which construct the synthetic control unit. These optimization depends on the predictor weights V, which are determined in the outer optimization to guarantee the best possible pre-treatment fit. The resulting optimal weights define the synthetic control unit. The treatment effect consists of the difference in the outcome variable after treatment between the treated unit and the synthetic control.

1.3 The Implementation and the Algorithm

The structure of the synthetic control method described above poses some challenges to the implementation. The problem is that the nested optimization is not only computational intensive and therefore slow with larger data sets, but it might also be quite unstable and unreliable with numerical optimizers. The reason for that is that the objective function of the outer optimization contains a minimization problem, which results in a noisy function that might be ill behaved and can fool the outer optimizer. Becker and Klößner (2018) *[3]* provide an algorithm that tries to reduce these problems. It starts with detecting important special cases that are easy to compute and then tries to reduce the dimension of the nested optimization problem.

The basis of Becker and Klößner's argumentation consists of some theory concerning the optimization problems that have to be solved for applying the synthetic control method. They start with separating the donor pool in sunny and shady donors. A shady donor is a control unit, whose difference in predictor values to the treated unit multiplied by α with $0 < \alpha < 1$ lies inside the convex hull of the differences of all donor units. They show that if a donor is shady, it will not be part of an optimal synthetic control unit. Furthermore, they give simple solutions in cases with no sunny donors, which means exact fit is possible, or only one sunny donor, which will then be the unique donor with positive weight. If none of these special cases occur, the algorithm tests whether the unrestricted outer optimum is feasible. This means it searches for predictor weights V which result in donor weights W that constitute the global minimum of the outer optimization problem. Only if finding such predictor weights is not possible, the nested optimization is performed. In order to do this, the dimension of the problem is reduced by excluding all shady donors. A detailed description of the algorithm can be found in Becker and Klößner (2018) *[3]* and Figure 2 in their paper illustrate it's structure in a simple way.

1.4 Testing

I do different forms of testing:

For the functions defined in the model code that construct the algorithm explained in the last section I do unit and integration tests using pytest. For the function putting everything together I do system tests. These tests confirm that I implemented all formulas from Becker and Klößner correctly.

All data work was firstly done and tested using Jupyter notebooks.

Furtherore, as a kind of regression testing, I compared some results to the implementation of the synthetic control method in stata (the synth package by Abadie et al. (2010) *[1]*) for one specification and the results are similar.

CHAPTER

TWO

ORIGINAL DATA

This section stores the raw data, which you should not manipulate to ensure reproducibility.

Documentation of the different datasets in *original_data*:

`Current_account_percentage_GDP_OECD_quarterly_1990_2017.csv` : A dataset from OECD.Stat containing quarterly data on current account in % of GDP from 1990 to 2017 for all OECD countries.

`GDPPC_OECD_1990_2017.csv` : A dataset from the quarterly national accounts in OECD.Stat containing quarterly GDP per capita data from 1990 to 2017 for all OECD countries.

`predictor_data_WDI.csv` : A dataset containing data for the covariates used as predictors (trade opennes, inflation, industry share and schooling) for all OECD data from 2010 to 2015.

CHAPTER

THREE

DATA MANAGEMENT

This section does include all code doing the data management. This means it cleans the data, replaces missing values and brings it in the second normal form. Only the data needed for the later analysis is extracted.

The second normal form is violated in one case, namely in the created table containing the predictor variables. In this table also the country names are included. This is done to make the data easier to read for humans in between, which was necessary in some stages of the project. As the dataset is only small, in my opinion the violation is not that much of a concern.

The data management is done in the file `clean_raw_data.py` in the folder *src.data_management*.

It does and contains: Clean the raw data to use it for the analysis and bring it in the second normal form: Create a table for countries with IDs, one for time periods with IDs and one for every data series we do have and use later on. Furthermore, clean and replace missings for predictors and calculate averages for predictors over the years 2012-2014 and safe them in a seperate datafile.

All tested using Jupyter notebook and regression tests.

`create_ca_percentage_gdp_table` (*data*, *countries*, *periods*)
Create table of current accout in percentage of GDP for every country only including country ID and period ID.

`create_country_table` (*data*)
Create table of countries sorted alphabetically with new IDs.

`create_gdppc_const_prices_table` (*data*, *countries*, *periods*)
Create table of GDPPC in constant prices (OECD reference year) with fixed PPP for every country only including country ID and period ID.

`create_gdppc_current_prices_table` (*data*, *countries*, *periods*)
Create table of GDPPC in current prices with current PPP for every country only including country ID and period ID.

`create_period_table` (*data*)
Create table of Periods with IDs.

`create_table_of_predictors` (*country_table*, *predictor_variables*)
Create the table of predictor variables for every country only including country id and the different predictors.

`read_in_data` (*filename*)
Read in source data.

CHAPTER FOUR

MAIN MODEL ESTIMATIONS / SIMULATIONS

In this section the main analysis is done. This means the code computes the synthetic control unit and all relevant values for every model. To do so it defines numpy matrices from the data and calculate x_tilde and z_tilde which are needed as inputs for the algorithm used for the model code. Then it calls the main function from the *model_code*, **determine_synthetic_control_weights**, to calculate the synthetic control. Afterwards, the data is formated as needed for plots and graphs and all results are saved in pickle file as a dictionary.

All this is done in the file `synthetic_control.py` in the folder *src.analysis*.

It does and contains: Run the synthetic control method to define the weights for a synthetic Switzerland.

All developed and tested using Jupyter notebooks.

`get_x_one_and_x_zero_from_data`(*dep_variable*, *predictor_data*, *predictors*, *index_treated*, *index_donors*)

Extraxt X_1 and x_0, the matrices of predictor values for treated unit and donors from the data. Build up matrices recursivly looping over predictor definitions.

`get_x_tilde`(*dep_variable*, *predictor_data*, *predictors*, *index_treated*, *index_donors*)

Calculate X_tilde = X_0 - X_1 * vector_ones', the input data for the inner minimization of the synthetic control method.

`get_z_one_from_data`(*dep_variable*, *index_treated*, *index_pre_treatment_start*, *index_treatment_start*)

Calculate the array Z_one, the array of pre-treatment data of the dependent variable for the treated unit.

`get_z_tilde`(*dep_variable*, *index_treated*, *index_donors*, *index_pre_treatment_start*, *index_treatment_start*)

Calculate Z_tilde = Z_0 - Z_1 * vector_ones', the input data for the outer minimization of the synthetic control method.

`get_z_zero_from_data`(*dep_variable*, *index_donors*, *index_pre_treatment_start*, *index_treatment_start*)

Calculate the array Z_one, the array of pre-treatment data of the dependent variable for the treated unit.

`standarize_data`(*data*)

Standarize data to have mean zero and unit variance.

CHAPTER

FIVE

VISUALISATION AND RESULTS FORMATTING

In this section the tables and figures are constructed from the results contained in the pickle file saved in the main analysis. The code is located in the directory *src.final*.

5.1 Tables

For every specification the file `sc_tables.py` construct three tables:

1. a table containing the weights for the countries in the donor pool constituting the synthetic control.
2. a table containing the predictors for the treated country and the synthetic control.
3. a table containing the outcomes for the treated country and the synthetic control as well as the constraint weighting matrix V and the status of the optimization. (This table is constructed just for inspection and to calculate some values for the paper, but is not included in the paper)

5.2 Figures

Furthermore, for every specification the file `plot_sc_graph.py` construct a figure showing the development of the outcome variable for the treated unit and the synthetic control. To do so it defines and uses to functions.

The file does and contains: Plot the graph describing the solution of the synthetic control method. This means plot a line graph with time on x-achsis and the dependent variable on y-achsis and lines for the treated unit and the synthetic control.

`get_dates_for_x_axsis`(*start_year*, *end_year*, *start_quarter=1*, *end_quarter=1*)
    Prepare the dates on the x-axis of the plot.

`plot_dep_var`(*z_one*, *z_sc*, *model_name*, *start_date*, *end_date*, *name*, *treatment_date*)
    Create a line graph of the dependent variable for the treated unit and the synthetic control unit with time on x-axis.

CHAPTER

SIX

RESEARCH PAPER / PRESENTATIONS

The file: `research_paper.tex` contains the actual paper. This includes all text as well as the latex code to include figures and tables.

It is located in the directory *src.paper*.

Furthermore the directory contains the references cited in the paper in `refs.bib` and an image which is included as a figure in the paper as well.

CHAPTER

SEVEN

MODEL CODE

This section contains a collection of functions implementing the algorithm by Becker and Klößner (2018). *[3]*

This is done in the file `synth_control_functions.py` in the folder *src.model_code*:

The file contains: Provide functions to calculate optimal weights for synthetic control unit.

`calculate_inner_part_of_inequ_constr_for_linprog_16` (*x_tilde*, *w_outer*)
Calculate the inner part of the inequality constraints for the linear + program in (20) from Becker and Klößner (2018). This is, calculate (e_j - v) ‘ B_k w for k=1,...,K and stack them into a vector. Do that for all countries J, so j = 1,..., J and stack them into a matrix to get the matrix defining the inequality constraint of the linear program A_ub.

`create_matrix_v_from_v_k_tilde` (*v_k_tilde*, *k*)
Create matrix V from v_k_tilde about which to optimize, see (20) in Becker and Klößner (2018).

`determine_synthetic_control_weights` (*x_tilde*, *z_tilde*, *lower_bound=1e-08*)
Put everything together to determine the optimal country weights w and the optimal constraint weighting matrix v defining the synthetic control. As inputs use only numpy matrixes! The function works as described in the paper by Becker and Klößner (2018), especially it uses the algorithm shown in figure 2 of the paper. The difference to the paper so far is that it does not allow to pass a mapping V, but builds up V as a diagonal matrix of constraint weights. This means, so far it does not allow for time series as constraints with stable weights for the whole time series. Instead it always calculates for every single constraint the optimal weight to solve the outer minimization problem. This means do:

1.determine sunny donors

2. test of no sunny donors, if so solve outer minimization constraint to exact fit and stop.

3. test of one sunny donor, if so just give him a positive weight and stop.

4. try if unrestricted outer minimization feasible, if so choose weights solving unrestricted outer minimization and stop.

5.Perform the nested optimization task and choose weights solving it.

`determine_unrestricted_outer_optimum` (*z_tilde*)
Solve euqtion (13) in Becker and Klößner (2018) to get weights solving the unrestricted outer optimization problem. Return these weights.

`find_sunny_donors` (*x_tilde*)
Find sunny donors from a numpy array of all donors according to (9) in Becker and Klößner (2018). Return a numpy array only including the data remaining for sunny donors and the number

of sunny donors. Additionaly return the index in the matrix of columns for sunny and shady donors.

function_to_solve_for_outer_optimum(*w*, *z_tilde*)
Function to solve for the outer optimum as used in (7) or (13) of Becker and Klößner (2018).

inner_optimization(*x_tilde*, *v*)
Perform the inner optimization of the synthetic control problem, corresponding to equation (8) in Becker and Klößner (2018). Return optimal weights for countries w.

inner_optimization_function(*w*, *x_tilde*, *v*)
Construct the function to minimize by the inner optimization, corresponding to the part after min in (8) in Becker and Klößner (2018)

outer_optimization(*z_tilde*, *x_tilde*, *lower_bound=1e-08*)
Perform the outer optimization of equation (7) in Becker and klößner (2018). Use K optimizations of dimension K-1 as described in the paper in 3.5.. Return the optimal constrains weighting matrix v.

outer_optimization_function(*v_k_tilde*, *z_tilde*, *x_tilde*, *k*, *K*)
Construct the function to minimize in the K subproblems of the outer optimization. For that consruct V according to (20) and than construct function used in (7) in Becker and klößner (2018).

solve_case_of_no_sunny_donors(*x_tilde*)
Solve the case when we have no sunny donors, so a perfect fit of the treated unit is possible. In this case I deviate from the paper by Becker and Klößner (2018). In their paper they solve the linear programm (10). This does not work with scipy.optimize as the minimizer does not support problems with more equality constraints than independent variables. As I did not find a good optimizer for python so far solving this problem, I will just choose a weights solving the inner minimization, so leading to a perfect fit for the synthetic control unit regarding the predictor variables. This might not be optimal concerning the outer optimization! The reason for this shortcut is that the case of perfect fit is very unprobable to begin with. If this case happens, the program will print a warning. In this case a good solution would be to choose more predictor variables. In an extrem case all variables/data points defining the outer optimization can be used as predictors. This would lead to the problem being unimportant as outer and inner optimization become the same!

ten_to_the_power_of_x(*x*)
Calculate 10 to the power of x.

try_if_unrestricted_outer_optimum_feasible(*x_tilde*, *z_tilde*, *lower_bound=1e-08*)
Solve linear program in (16) in Becker and Klößner (2018). This means, determine if the outer optimum is feasible. If so, return a status telling that and the weights forming the outer optimum as well as the normalized constraint weighting matrix V belonging to the outer optimum. If not, return a status telling the outer optimum is infeasible and give numpy.nan as weights w and V.

All functions are tested using pytest within the file `synth_control_functions_test.py`!

CHAPTER

EIGHT

MODEL SPECIFICATIONS

The directory *src.model_specs* contains JSON files with model specifications. These files define the different model specifications used to calculate synthetic controls for different szenarios.

You can input the treatment country, the treatment date, the period you are considering, the donor pool, the outcome variable and all predictor variables.

Thereby everything except the predictor variables are defined in the files named synth_control_parameters_*specification_name*, and the predictors in predictors_*specification_name*. In the later one also the names of the predictors to input in the table are defined. It is important that for every specification both are specified.

These specifications are, as explained in the introduction:

1. a baseline specification for GDPPC: outcome is GDPPC and predictors are yearly averages of the outcome for 2008, 2010, 2012, 2013, 2014
2. a all-year averages specification for GDPPC: outcome is GDPPC and predictors are yearly averages of the outcome for 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014
3. a with-covariates specification: outcome is GDPPC and predictors are an averages of the outcome for 2012, 2013, 2014 and other covariates: trade opennes, inflation, industry share and schooling
4. a current-account specification: outcome is CA as % of GDP and predictors are an averages of the outcome for 2012, 2013, 2014 and other covariates: trade opennes, inflation, industry share and schooling
5. a time placebo specification: as baseline, but treatment time changes to 2005Q1 and accordingly sample period and predictor years are ten years earlier
6. a country-placebo specification: as baseline, but treated country changes to Australia and Switzerland is excluded

CHAPTER NINE

REFERENCES

BIBLIOGRAPHY

[1] Alberto Abadie, Alexis Diamond, and Jens Hainmueller. Synthetic control methods for comparative case studies: estimating the effect of california's tobacco control program. *Journal of the American Statistical Association*, 105(490):493–505, 2010. URL: https://doi.org/10.1198/jasa.2009.ap08746, arXiv:https://doi.org/10.1198/jasa.2009.ap08746, doi:10.1198/jasa.2009.ap08746.

[2] Alberto Abadie, Alexis Diamond, and Jens Hainmueller. Comparative politics and the synthetic control method. *American Journal of Political Science*, 59(2):495–510, apr 2014. doi:10.1111/ajps.12116.

[3] Martin Becker and Stefan Klößner. Fast and reliable computation of generalized synthetic controls. *Econometrics and Statistics*, 5:1–19, jan 2018. doi:10.1016/j.ecosta.2017.08.002.

[4] Hans-Martin von Gaudecker. Templates for reproducible research projects in economics. https://github.com/hmgaudecker/econ-project-templates, 2014.

PYTHON MODULE INDEX