Documentation of the Skills and Wages project

Poooja Bansal

02 March 2019

CONTENTS

CHAPTER

ONE

INTRODUCTION

Documentation on the rationale, Waf, and more background is at http://hmgaudecker.github.io/econ-project-templates/

The Python version of the template uses a modified version of Stachurski's and Sargent's code accompanying their Online Course [2] for Schelling's (1969, [1]) segregation model as the running exmaple.

1.1 Getting started

This assumes you have completed the steps in the README.md file **and everything worked.**

The logic of the project template works by step of the analysis:

1. Data management
2. The actual estimations / simulations / ?
3. Visualisation and results formatting (e.g. exporting of LaTeX tables)
4. Research paper and presentations.

It can be useful to have code and model parameters available to more than one of these steps, in that case see sections model_specifications, model_code, and *Code library*.

First of all, think about whether this structure fits your needs – if it does not, you need to adjust (delete/add/rename) directories and files in the following locations:

- Directories in **src/**;
- The list of included wscript files in **src/wscript**;
- The documentation source files in **src/documentation/** (Note: These should follow the directories in **src** exactly);
- The list of included documentation source files in **src/documentation/index.rst**

Later adjustments should be painlessly possible, so things won't be set in stone.

Once you have done that, move your source data to **src/original_data/** and start filling up the actual steps of the project workflow (data management, analysis, final steps, paper). All you should need to worry about is to call the correct task generators in the wscript files. Always specify the actions in the wscript that lives in the same directory as your main source file. Make sure you understand how the paths work in Waf and how to use the auto-generated files in the language you are using particular language (see the section *Project paths* below).

1.2 Project paths

A variety of project paths are defined in the top-level wscript file. These are exported to header files in other languages. So in case you require different paths (e.g. if you have many different datasets, you may want to have one path to each of them), adjust them in the top-level wscript file.

The following is taken from the top-level wscript file. Modify any project-wide path settings there.

```
def set_project_paths(ctx):
    """Return a dictionary with project paths represented by Waf nodes."""

    pp = OrderedDict()
    pp["PROJECT_ROOT"] = "."
    pp["IN_DATA"] = "src/original_data/"
    pp["LIBRARY"] = "src/library"
    pp["BLD"] = ""
    pp["OUT_DATA"] = f"{out}/out/data"
    pp["OUT_ANALYSIS"] = f"{out}/out/analysis"
    pp["OUT_FINAL"] = f"{out}/out/final"
    pp["OUT_FIGURES"] = f"{out}/out/figures"
```

As should be evident from the similarity of the names, the paths follow the steps of the analysis in the `src` directory:

1. **data_management** → **OUT_DATA**
2. **analysis** → **OUT_ANALYSIS**
3. **final** → **OUT_FINAL**, **OUT_FIGURES**, **OUT_TABLES**

These will re-appear in automatically generated header files by calling the `write_project_paths` task generator (just use an output file with the correct extension for the language you need – `.py`, `.r`, `.m`, `.do`)

By default, these header files are generated in the top-level build directory, i.e. `bld`. The Python version defines a dictionary `project_paths` and a couple of convencience functions documented below. You can access these by adding a line:

```
from bld.project_paths import XXX
```

at the top of you Python-scripts. Here is the documentation of the module:

bld.project_paths

Define a dictionary *project_paths* with path definitions for the entire project.

This module is automatically generated by Waf, never change it!

If paths need adjustment, change them in the root wscript file.

project_paths_join(*key*, **args*)

Given input of a *key* in the *project_paths* dictionary and a number of path arguments *args*, return the joined path constructed by:

```
os.path.join(project_paths[key], *args)
```

project_paths_join_latex(*key*, **args*)

Given input of a *key* in the *project_paths* dictionary and a number of path arguments *args*, return the joined path constructed by:

```
os.path.join(project_paths[key], *args)
```

and backslashes replaced by forward slashes.

CHAPTER

TWO

ORIGINAL DATA

Documentation of the different datasets in *original_data*.

In the original data section you would store the raw data, which you should not manipulate to ensure reproducibility.

If you want to include multiple data sets, you can also create subfolders for the sake of a clear structure.

CHAPTER

THREE

DATA MANAGEMENT

Documentation of the code in *src.data_management*.

We start with merging all the relevant data files. SOEP has provided with the data for two different waves using different questionnaires. We first merge all of the datasets and then keep only the useful variables. Now, all our data for skill variables, be it cognitive or non-cognitive was categorical. So we first converted the categorical figures into numeric to perform mathematical operations on them. We then rename the important variables to be used later for our convenience. Then we restrict our data with individuals of age between 20 and 60 and drop individuals with occupations not useful in our analysis. After this, we dropped all the missing values by first replacing them nan. The variables for Cognitive abilities were standardised using the sklearn library. For personality we had 15 variables to be reduced to 5 be taking averages of 3 variables corresponding to a particular personality trait. Out of those 15, 4 had to be reversed since they represented the opposite quality corresponding the respective trait. We then created 5 variables by taking the average and standardising them. We then generated the experience and experience squared variables for the Mincer equation. We then took a log of wages to improve our regression model. For occupations, we combined similar categories and sorted them in order.

CHAPTER

FOUR

MAIN MODEL ESTIMATIONS / SIMULATIONS

Documentation of the code in *src.analysis*. This is the core of the project.

4.1 Regression

We run the regression using statsmodel python package.

First we investigate the returns to earnings by using the basic Mincer Equation which used 3 variables [Years of education, Years of Experience and Square of years of experience.] We will then expand the basic specification by adding the cognitive skills : Fluency and Symbol test score which we standardised. We then introduce only non cognitive skills in the basic specification : Openness, Conscientiousness, Extraversion , Agreeableness and Neuroticism.

We then add both skills (Cognitive and Non-Cognitive)to the specification.

We define our independent variables as X and dependent variables as Y, For each specification, we generate a new matrix. We fit our linear model using the OLS module offered by the sm library and then print the summary which contains the regression output for all the defined model.

After the first 4 regression, we perform regression for different occupational groups by defining a loop for the values in the our column occupation which contains : 1,2,3,4 thus generates results for 4 different outputs for all categories.

4.2 Decision Tree

CHAPTER

FIVE

VISUALISATION AND RESULTS FORMATTING

Documentation of the code in *src.final*.

5.1 Schelling example

CHAPTER

SIX

RESEARCH PAPER / PRESENTATIONS

Purpose of the different files (rename them to your liking):

- `research_paper.tex` contains the actual paper.
- `research_pres_30min.tex` contains a typical conference presentation.
- `research_pres_90min.tex` contains a full-length seminar presentation (add by yourself).
- `formulas` contains short files with the LaTeX formulas – put these into a library for re-use in paper and presentations.

CHAPTER

SEVEN

CODE LIBRARY

The directory *src.library* provides code that may be used by different steps of the analysis. Little code snippets for input / output or stuff that is not directly related to the model would go here.

The distinction from the model_code directory is a bit arbitrary, but I have found it useful in the past.

CHAPTER

EIGHT

REFERENCES

BIBLIOGRAPHY

[1] Thomas C. Schelling. Models of segregation. *The American Economic Review*, 59(2):488–493, 1969.

[2] John Stachurski and Thomas J. Sargent. Quantitative economics. Available at http://quant-econ.net/index.html, 2013.

PYTHON MODULE INDEX

INDEX