XBUILD - EXTENDED G7 REGRESSION TRANSLATOR SOFTWARE

23th INFORUM Conference, Bangkok, August 2015

Frank Hohmann
PortableDyme Model Building Framework

► Complete model building framework (software + basic model)

► Mainly INFORUM software (esp. G7, Dyme) plus add-ons, e.g.
  ☞ PSPad (professional multi-file editor)
  ☞ Excel VBA tools (data management, scenario evaluation)
  ☞ Various scripts (batch, G7)

► Portable: Preconfigured to run from any storage media (i.e. USB devices) without installation

► Two versions of PortableDyme
  ☞ „Plain Vanilla“: Software only; no model, no data
  ☞ Basic IO model with SNAB based on WIOD data
    ▪ May be easily adopted to 40 WIOD countries
    ▪ Has to be customized to become a sophisticated model
PortableDyme Model Building Framework (cont.)

- **Main steps of model building:**
  1. Building the historical database
  2. Performing regressions
  3. Writing model code
  4. Performing impact analysis and evaluation

- **PortableDyme reflects these steps both on disk and in the project editor**

- **Each step contains preconfigured scripts and instructions**

- **Model building is an iterative process!**
PortableDyme Model Building Framework (cont.)

▶ One important „to do“:
Smoothen transition from
step 2 „regressions“ to
step 3 „model programming“:
⇒ Simplify vector regressions
⇒ Enable „log-log“ regressions
⇒ Improve error handling
⇒ Provide runtime checks
Regressions and Definitions (Step 2)

► Needed Software: G7

► Data banks need to be loaded first (either in g.cfg) or by commands:
  
  bank calcdada
  vam calcdada b
  dvam b

► Definitions are written using the „id“ command, e.g. for net domestic product B1NT
  
  id B1NT = B1GT – K1UT

► Appropriate limits have to be given for each regression, e.g.
  
  lim 1996 2008

► The „r“ command performs a regression, e.g.

  Macro: r P1RT = !OUT
  Vector: r im01 = !out01 („!“ omits the constant term)
Results are saved using the "save" command, e.g.

```
save snab.sav
```

Sample output

**Macro:**

```
r P1RT = !OUT becomes
r P1RT = 1.011268*OUT
d
```

**Vector:**

```
r im01 = !out01 becomes
r im01 = 0.387989*out01
d
```
Model Programming (Step 3)

- Needed software: idbuild, C++ compiler
- Important task: Integrate regression results into the model
  - idbuild translates .sav files into C++ code (i.e. heart.cpp)
  - Regression functions need to be called in the C++ model code
  - Compiler generates executable from C++ files
- idbuild translates list of files given in „master“ file by iadd commands, e.g.
  
  iadd SNAB.sav

- What about „rho“-adjustment and fixes?
  - rho-adjustment: difference between historical and calculated value in the last historical year („error term“)
  - fixes: overriding calculated values by given values, i.e. necessary for scenario analysis
Model Programming (Step 3) (cont.)

- For macro regressions, translation is straightforward:
  \[ r \ P1RT = 1.011268 \times \text{OUT} \ d \]
  becomes
  \[ /* P1RT */ \text{depend} = \text{coef}[2][0] \times \text{OUT}[t]; \]
  \[ \text{P1RT}.modify(\text{depend}); \]
  \[ \Rightarrow \text{“modify“ handles both rho-ajustment and fixes} \]

- For vector regressions, vector names need to be given with “isvector” command, e.g.
  \[ \text{isvector } \text{im} \]
  \[ \text{isvector } \text{out} \]
  \[ \text{iadd } \text{im.sav} \]

- Example of vector regression translation:
  \[ r \ \text{im01} = 0.387989 \times \text{out01} \ d \]
  becomes
  \[ \text{depend} = \text{im}[1]; \]
  \[ \text{im}[1] = \text{coef}[22][0] \times \text{out}[1]; \]
  \[ \Rightarrow \text{rho-ajustment and fixes are missing} \]
Model Programming (Step 3) (cont.)

Options for dealing with missing rho-adjustment and fixes

   - „punch“ coefficients into .eqn files
   - Create vector regression handlers in C++
   - For advanced users only

2. By hand in .reg files (current PortableDyme approach)
   Example of a log-log regression:
   \[
   r \log(hcesr01) = \log(B6GT / PHCES), \log(phces01/PHCES)
   \]
   has to be written as
   \[
   f lhcesr = \log(hcesr01) \\
   r lhcesr = \log(B6GT/PHCES), \log(phces01/PHCES) \\
   cc hcesr[1] = hcesrEQN.rhoadj(exp(depend), hcesr[1], 1); \\
   cc hcesr.fix(t, 1); \\
   \]

TODOs:
   - Handle log-log regressions automatically
   - Include rho-adjustment and fixes in translation
Problem: Why does the model behave erroneous although regression results looked fine?

- Missing statements/equations for RHS variables
- Missing/weired values in the database
- Erroneous statements cause problems while model iterates
- Math errors like log(0), division by zero, over-/underflows, etc.
- ...

High degree of interdependency is a problem per se!

Standard C function „assert“ helps to detect errors:

- `void assert(boolean expression);`  //defined in assert.h
- If `expression` evaluates to `false`, program terminates giving the module name and line number causing the problem
Example for a „secured“ regression:

\[ r \text{ lhcesr} = \log(B6GT / \text{PHCES}) + \log(phces01 / \text{PHCES}) \]

requires the following asserts in the model:

- `assert(PHCES[t] != 0);` // check division by zero
- `assert(B6GT[t] / PHCES[t] > 0);` // check log()
- `assert(phces[1] / PHCES[t] > 0);` // check log()

**TODO:** Automatically insert „assert“ statements to detect math errors

Model builder should use „assert“ statements to secure other parts of the model
Xbuild Features

- xbuild addresses the aforementioned TODOs
  - Improve readability of equations by including coefficient values
  - Fully translate log-log regressions
  - Include rho-adjustment and "fix"-statements
  - Provide runtime checks ("assert")
  - [Provide static checks (e.g. index errors) !?]

Demonstration: xbuild in action...
Xbuild Configuration

- Xbuild uses a configuration file in .ini format:

  ```ini
  [xbuild]  General settings (just some examples)
  asserts=1
  lineBreaks=0
  include=./user includes

  [banks]  Banks to use
  vam=dyme.vam
  bnk=dyme.bnk

  [files]  Files to translate
  SNAB.sav
  FD.sav
  im.sav
  hcesr.sav
  ```

- Processing starts by giving the .ini file as a parameter:

  ```bash
  xbuild xbuild.conf
  ```

- Integrates nicely into PortableDyme’s idmodel.bat
Thank you for your attention

Frank Hohmann
Gesellschaft für Wirtschaftliche Strukturforschung mbH
Heinrichstr. 30
49080 Osnabrück
Tel + 49 (0) 541 40933-130
Fax + 49 (0) 541 40933-110
hohmann@gws-os.com