Introduction to package `isismdl`

Rob van Harrevelt

2018-01-22

Contents

1 Introduction

This introduction shows how package `isismdl` can be used to solve a simple example model, a dynamical version of the ISLM model.

[Here follows a description of the ISLM model]

2 The model file

The model file should be defined on an external ASCII file. A detailed discussion about the syntax of this file is provided in the Reference Manual of Isis.

A model file for the ISLM model is included in the example models in directory `models` of the package directory. To copy this file to you working directory, use

```
> mdl_file <- system.file("models", "islm.mdl", package = "isismdl")
> file.copy(mdl_file, "islm.mdl")
```

The model file `islm.mdl` has the following contents:

```
param c0 100 c1 0.7  c2 20 c3 0.5;
param i0 100 i1 0.2  i2 40 i3 1.5;
param m0  75 m1 0.23 m2 35 m3 1.5;
param t0 -25 t1 0.22;

? behavourial equations
frml c = c0 + c1 * (0.9 * yd + 0.1 * yd[-1]) - c2 * r + c3 * r**2;
frml i = i0 + i1 * (0.2 * y + 0.8 * y[-1]) - i2 * r[-1] + i3 * r[-1]**2;
frml md = m0 + m1 * y - m2 * r + m3 * r**2;
frml t = t0 + t1 * y;

? ident equations
ident yd = y - t;
ident y = c + i + g;
ident r = r + (ms - md) / ms;
```

3 Creating the model

The function `isis_mdl` parses the model file and creates an `IsisMdl` object

```
> mdl <- isis_mdl("islm.mdl")
```

```
Isis Model Compiler 3.00
Compiling model...
       7 equations processed
Ordering equations...
Checking redundant feedback variables
       0 redundant feedback variables detected
Generating feedback variable ordering
Writing MIF file...
Writing cross-reference file...
End compilation
Reading mif file...
Model with     7 equations read
Checking Model-code...
Model is ok...
Feedback ordering generated ...
```

This function generates the file `islm.mrf`, the so called model reference file (mrf file), a text file with information about the structure of the model. If the compiler detects an error in the model file, an additional file `islm.err` is created. This file contains a list of errors.

The mrf file for our ISLM looks like this:

```
 ISIS model compilation : islm 2018-01-22  11:48:33

 Model reference table

 Each variable name is followed by its
 maximum lag (with minus sign) and maximum lead in the model,
 its type:  E(xogenous), B(ehavioral), I(dentity).
 Exogenous variables listed first.

 Parameters are listed separately.
 Each parameter name is followed by its length.
```

```
Equations are listed in solution order,
followed by a list of feedback variables.

*** Statistics ***

      9 variables of which
             2 exogenous
             4 behavioral
             3 identity

      3 total number of lags and leads with
             1 maximum lag
             0 maximum lead
             0 variables with leads

     14 parameters
            14 total length of parameter values

      7 equations of which
             0 in prologue
             7 in simultaneous block
             0 in epilogue

      2 feedback variables
               4 (****%) structural non zero's in jacobian


Statistics of feedback ordering
              7 words of memory used
            2   minimum feedback chain (excl. fb equations)
            5   maximum feedback chain (excl. fb equations)
            3.5 average feedback chain (excl. fb equations)
            2   average newton steps (rounded up)


*** Variables ***

*** Exogenous ***

g       0   0  E
ms      0   0  E

*** Endogenous ***

c       0   0  B
i       0   0  B
md      0   0  B
r      -1   0  I
t       0   0  B
y      -1   0  I
yd     -1   0  I

*** Parameters ***
```

```
c0      1
c1      1
c2      1
c3      1
i0      1
i1      1
i2      1
i3      1
m0      1
m1      1
m2      1
m3      1
t0      1
t1      1

*** Equations (in solution order) ***

    0 Prologue equations

    7 Simultaneous equations

i    md   t    yd   c    y    r

    0 Epilogue equations


*** Feedback variables ***

    2 Feedback variables

r    y


*** Additional information for feedback variables
    Name of feedback variable followed by how it was chosen: fixed, diagonal, heuristic,
    and the length of the feedback cycle (excluding feedback variables)


r    diagonal          2
y    diagonal          5
```

The mrf file contains technical information about the model. For example, it shows which variables are exogenous or endogenous, and which variables are feedback variables. A brief explanation of these concepts:

- Exogenous variables are variables that only occur at the right hand side of the equation.
- Endogenous variables are variables on the left hand side of the equations.
- A feedback variable is a variable that directly or indirectly depends on itself. In order to solve the model, initial guess values for the feedback variables have to be specified.

4 IsisMdl objects

An `IsisMdl` object is an R6 class object. R6 classes behave quite differently than the more familiar S3 and S4 classes. For example, for R6 classes methods are part of the object itself and not of generic functions. R6 classes behave in a similar way as classes in object oriented languages such as Java, C++ or Python.

For example, the method `get_params()` can be used to obtain the values of the model parameters. For example, to obtains the values of parameters `m0` and `c1`, use

```
> mdl$get_param(names = c("m0", "c1"))
```

```
$m0
[1] 75

$c1
[1] 0.7
```

Methods starting with `get_`, are often called "getter" methods. There are also corresponding "setter" methods. For example

```
> mdl$set_param(list(m0 = 100))
> mdl$get_param("m0")
```

```
$m0
[1] 100
```

Input for method `set_params()` is a named list.

4.1 Copying IsisMdl objects

Consider the following assignment:

```
> mdl2 <- mdl
```

Now variables `mdl2` and `mdl` refer to the same object. If you modify `mdl2`, then also `mdl` is modified.

```
> mdl2$set_param(list(m0 = 75))
> mdl$get_param("m0")
```

```
$m0
[1] 75
```

The usual copy-on-modify semantics that are used for convential R objects such as S3 or S4 classes do not apply to R6 classes.

If you want to create a copy of the model, use the `copy()` method with argument `deep = TRUE` :

```
> mdl2 <- mdl$copy()
> mdl2$set_param(list(m0 = -9999))
> mdl$get_param("m0")
```

```
$m0
[1] 75
```

5 Solving the model

Suppose that we want to solve the model for the period from `2017Q1` to `2018Q2`. Then we need the values of the exogenous variables for that period. For the feedback variables `y` and `r` we also need initial starting

values for the model period 2017Q1/2018Q2'. Sincer,yandy_dare lagged, we also need values for these variables in2016Q4‘.

In typical applications, the input timeseries are read from an external file, for example a csv file. For this tutorial we will create a timeseries object manually by employing function `regts` of the `regts` package. For the exogenous variables `g` and `ms` we assume an annual growth of 1.5% after `2016Q4`.

```
> # exogenous variables:
> g  <- regts(210 * cumprod(rep(1.015, 6)), start = "2017Q1")
> ms <- regts(200 * cumprod(rep(1.015, 6)), start = "2017Q1")
> # feedback variables (with lag):
> r <- regts(3.4, period = "2016Q4/2018Q2")
> y <- regts(980, period = "2016Q4/2018Q2")
> # lagged variable
> yd <- regts(790, start = "2016Q4")
> data <- cbind(g, ms, r, y, yd)
> data
```

```
              g       ms   r   y  yd
2016Q4       NA       NA 3.4 980 790
2017Q1 213.1500 203.0000 3.4 980  NA
2017Q2 216.3472 206.0450 3.4 980  NA
2017Q3 219.5925 209.1357 3.4 980  NA
2017Q4 222.8863 212.2727 3.4 980  NA
2018Q1 226.2296 215.4568 3.4 980  NA
2018Q2 229.6231 218.6887 3.4 980  NA
```

To transfer these variables to the model, we first have to define the model period. This is the period for which the model will be solved. For this we use the method `set_period()`:

```
> mdl$set_period(period_range("2017Q1", "2018Q2"))
> mdl
```

```
IsisModel object
Model index:                                                    1
Number of variables:                                            9
Maximum lag:                                                    1
Maximum lead:                                                   0
Model period:                                                   2017Q1/2018Q2
Model data period:                                              2016Q4/2018Q2
```

The data period is the model period extended with the lag and lead period. You can also retrieve the model period and model data period with the methods `get_period()` and `get_data_period`.

Now we can transfer the values in `regts` object data to the model with method `set_data()`:

```
> mdl$set_data(data)
```

To check the model data, we use

```
> mdl$get_data()
```

```
        c        g  i md       ms   r  t   y  yd
2016Q4 NA       NA NA NA       NA 3.4 NA 980 790
2017Q1 NA 213.1500 NA NA 203.0000 3.4 NA 980  NA
2017Q2 NA 216.3472 NA NA 206.0450 3.4 NA 980  NA
2017Q3 NA 219.5925 NA NA 209.1357 3.4 NA 980  NA
2017Q4 NA 222.8863 NA NA 212.2727 3.4 NA 980  NA
2018Q1 NA 226.2296 NA NA 215.4568 3.4 NA 980  NA
```

```
2018Q2 NA 229.6231 NA NA 218.6887 3.4 NA 980  NA
```

Model variables that we have not explicitly set all value `NA`. These values are not needed to solve the model.

We are now ready to solve the model:

```
> mdl$solve()
```

```
Model Solve Options
Solution period               2017Q1/2018Q2
Simulation mode               dynamic
Feedback starting values      current period
Maximum iterations per period 50
Relaxation minimum            0.500E-01
           maximum            1.00
           shrinkage          0.500
Criteria stepback             1.30
          matrix              0.900
Maximum updates Newton matrix per period                10
Maximum number of line searches with old Jacobian       5
Criterion for line search decisions etc.                geometric

   1 steps for Newton matrix at iteration   0 (1/condscal =  1.43E-01)
Convergence for 2017Q1 in    4 iterations
Convergence for 2017Q2 in    5 iterations
Convergence for 2017Q3 in    4 iterations
Convergence for 2017Q4 in    4 iterations
Convergence for 2018Q1 in    4 iterations
Convergence for 2018Q2 in    4 iterations
Total number of iterations   26
Solve model used     0.00 CPU secs
```

The method `get_data()` can be used to retrieve the solution.

```
> mdl$get_data()
```

```
              c        g        i       md       ms        r        t
2016Q4       NA       NA       NA       NA       NA 3.400000       NA
2017Q1 595.2722 213.1500 177.5801 203.0000 203.0000 3.284697 191.9205
2017Q2 602.9867 216.3472 182.6351 206.0450 206.0450 3.309682 195.4332
2017Q3 610.2563 219.5925 184.9507 209.1357 209.1357 3.304113 198.2559
2017Q4 617.5631 222.8863 187.7053 212.2727 212.2727 3.301510 201.1941
2018Q1 624.9866 226.2296 190.4614 215.4568 215.4568 3.298568 204.1691
2018Q2 632.5206 229.6231 193.2628 218.6887 218.6887 3.295613 207.1894
               y       yd
2016Q4  980.0000 790.0000
2017Q1  986.0023 794.0818
2017Q2 1001.9690 806.5358
2017Q3 1014.7995 816.5436
2017Q4 1028.1548 826.9607
2018Q1 1041.6776 837.5085
2018Q2 1055.4065 848.2171
```

6 Writing and reading `IsisMdl` objects

To write an `IsisMdl`object to a file, use for example

```
> mdl$write_mdl(file = "islm_mdl.ismdl")
```

This methods serializes the model equations, parameters, model data, solve options, etc. You can read this model back with the command

```
> mdl <- read_mdl("islm_mdl.ismdl")
```

Do not use the standard functions for writing objects to a file and to restore them, such as `save`, `saveRDS`, `load` and `readRDS`. These functions cannot handle the complex structure of `IsisMdl` objects.

7 Fixing variables

TODO

8 The fit procedure

TODO

9 Modellen bewerken

9.1 Vergelijkingen uitschakelen

TODO

9.2 Modeldata aanvullen

TODO

9.3 Losse vergelijkingen draaien

10 Bijsturen van het modeloplossen

TODO