RSLinearDiscreteExample

August 19, 2016

0.1 Introduction

This example illustrates how to fit a discrete-time regime-switching linear model in the `dynr` package.

0.2 Data

First, create a dynr data object using *dynr.data*. Here our observed variable is called "EMG" . In additon, we are to use a covariate, "self", in our measurement model, thus it also needs to be specified in the data step.

```
require(dynr)

data(EMGsim)
dd <- dynr.data(EMGsim, id='id', time='time', observed='EMG', covariates='self')
```

0.3 Measurement Model

Next, we specify the measurement model using *prep.measurement*. Parameters are indicated by parameter names (e.g. `mu_0` and `beta_0` in the following code), and fixed values are indicated by "fixed". In this case, our model has two regimes and the measurement models differ in these two regimes. The `values.*` arguments specify the starting values and fixed values. The `params.*` arguments specify the free parameter names or the reserved word "fixed" for fixed parameters. Hence `values.int` gives the values of the *intercept* and `params.int` gives the parameter names of the *intercept*. Parameters with the same name are constrained to be equal. Likewise `values.exo` specifies the values of the loadings associated with the *exogenous* covariate(s) and `params.exo` specifies the fixed values and parameter names in the covariate loadings. These arguments are lists lists of two matrices: the first matrix is for the measurement model of the first regime; the second matrix is for the second regime. The measurement models for the regimes are written out as follows:

Regime 1: $EMG(t) = \mu_0 + lEMG(t) + beta_0 \times self$
Regime 2: $EMG(t) = \mu_1 + lEMG(t) + beta_1 \times self$

$lEMG$ is the latent state EMG variable.

```
recMeas <- prep.measurement(
    values.load=rep(list(matrix(1, 1, 1)), 2),
    values.int=list(matrix(0, 1, 1), matrix(1, 1, 1)),
    params.int=list(matrix('mu_0', 1, 1), matrix('mu_1', 1, 1)),
    values.exo=list(matrix(0, 1, 1), matrix(1, 1, 1)),
    params.exo=list(matrix('beta_0', 1, 1), matrix('beta_1', 1, 1)),
    obs.names = c('EMG'),
    state.names=c('lEMG'),
    exo.names=c("self"))
```

0.4 Dynamic Model

In the next chuck, we specify our dynamic models by first specifying the covariance matrices of measurement errors and dynamic noises using *prep.noise*, and then specifying the dynamic functions using *prep.matrixDynamics*. The dynamic models are:

Regime 1: $lEMG(t+1) = \phi_0 \times lEMG(t) + w(t)$
Regime 2: $lEMG(t+1) = \phi_1 \times lEMG(t) + w(t)$
$w(t) \sim N(0, dynNoise)$

We assume the same dynamic noise process applies to both regimes, hence we only have one matrix for dynamic noise specification (`*.latent`). We also assume there is no measurement error in this model, shown by fixing `values.observed` to a 0 matrix in *prep.noise*.

```
recNoise <- prep.noise(
    values.latent=matrix(1, 1, 1),
    params.latent=matrix('dynNoise', 1, 1),
    values.observed=matrix(0, 1, 1),
    params.observed=matrix('fixed', 1, 1))

recDyn <- prep.matrixDynamics(
    values.dyn=list(matrix(.1, 1, 1), matrix(.8, 1, 1)),
    params.dyn=list(matrix('phi_0', 1, 1), matrix('phi_1', 1, 1)),
    isContinuousTime=FALSE)
```

0.5 Regime Probabilities

In the next step, we specify transition probability matrix between the regimes using *prep.regimes*. This transition probabilities from time *t* to time *t+1* are:

	Regime1(t+1)	**Regime2(t+1)**
Regime1(t)	$\frac{exp(p11)}{exp(p11)+exp(0)}$	$\frac{exp(0)}{exp(p11)+exp(0)}$
Regime2(t)	$\frac{exp(p21)}{exp(p21)+exp(0)}$	$\frac{exp(0)}{exp(p21)+exp(0)}$

(Here, $p11$ and $p21$ are model parameters.)

```
recReg <- prep.regimes(
    values=matrix(0, 2, 2),
    params=matrix(c('p11', 'p21', 'fixed', 'fixed'), 2, 2))
```

0.6 Initial Values

After that, we specify values at time *t=0* using *prep.initial*. These values are used to initialize the recursive algorithm (extended Kalman filter) that dynr uses. The `*.inistate` arguments specify the initial (starting) states of the latent state variables. The `*.inicov` arguments specify the starting covariance matrix of the latent state variables. The `*.regimep` specifies the initial probabilities of the two regimes.

```
recIni <- prep.initial(
    values.inistate=matrix(0, 1, 1),
    params.inistate=matrix('fixed', 1, 1),
    values.inicov=matrix(1, 1, 1),
    params.inicov=matrix('fixed', 1, 1),
    values.regimep=c(1, 0),
    params.regimep=c('fixed', 'fixed'))
```

0.7 Dynr Model

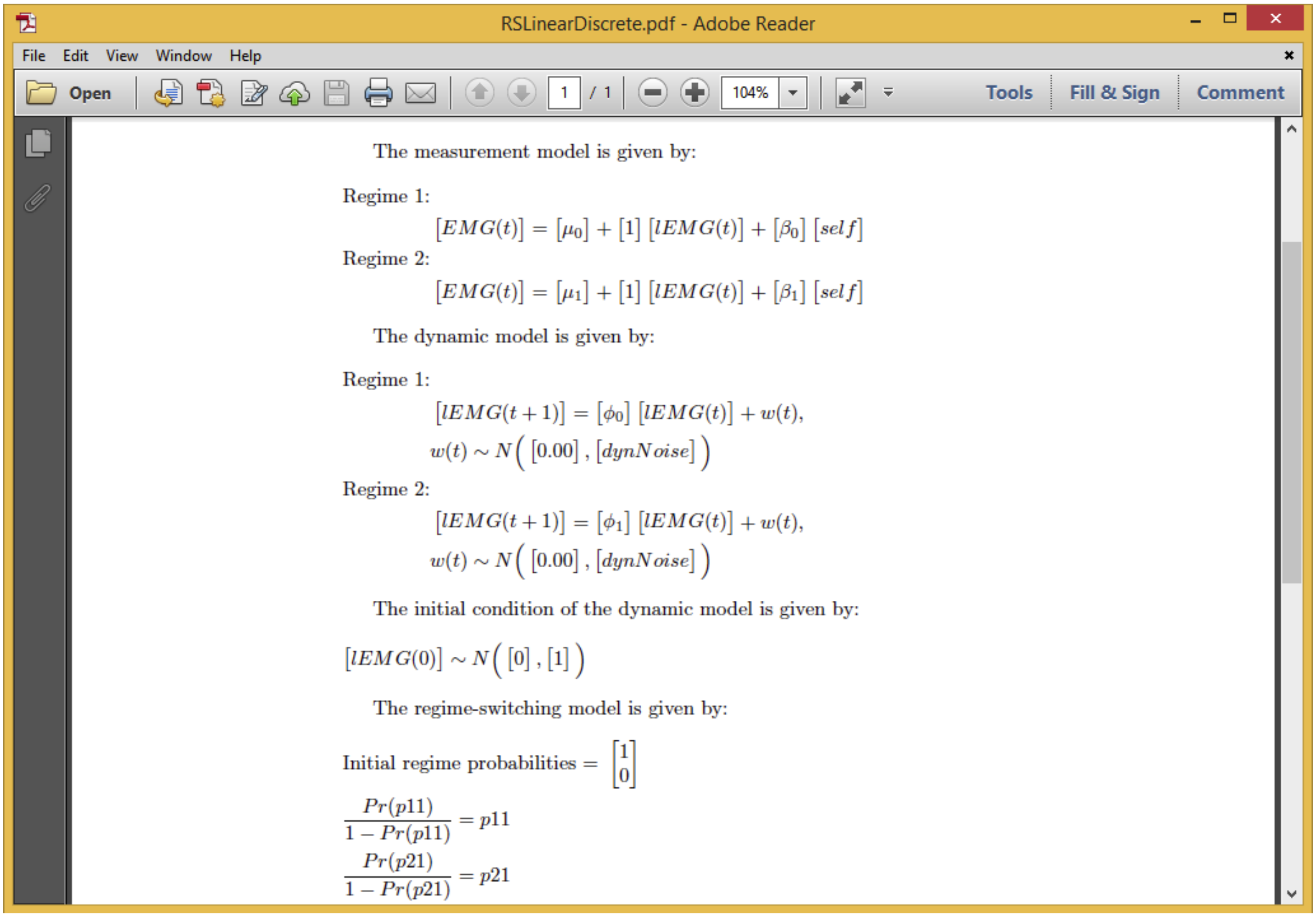
Now we put together everything we've previously specified in *dynr.model*. This code connects the recipes we've written up with our data and writes a c file in our working directory. We can inspect c functions that go with each recipe in the c file.

```
rsmod <- dynr.model(dynamics=recDyn, measurement=recMeas, noise=recNoise,
initial=recIni, regimes=recReg,
                    data=dd, outfile="RSLinearDiscrete.c")
```

0.8 Tex Options

We can check our model specifications in a neatly printed pdf file using the following code.

The *printex* command is used to write the model into a Latex file, with a name given by the `outFile` argument. Then, the *tools::texi2pdf* command generates a pdf file from the latex file we just created. The *system* command prints out the pdf file:



The measurement model is given by:

Regime 1:

$$[EMG(t)] = [\mu_0] + [1]\,[lEMG(t)] + [\beta_0]\,[self]$$

Regime 2:

$$[EMG(t)] = [\mu_1] + [1]\,[lEMG(t)] + [\beta_1]\,[self]$$

The dynamic model is given by:

Regime 1:

$$[lEMG(t+1)] = [\phi_0]\,[lEMG(t)] + w(t),$$
$$w(t) \sim N\Big(\,[0.00]\,,\,[dynNoise]\,\Big)$$

Regime 2:

$$[lEMG(t+1)] = [\phi_1]\,[lEMG(t)] + w(t),$$
$$w(t) \sim N\Big(\,[0.00]\,,\,[dynNoise]\,\Big)$$

The initial condition of the dynamic model is given by:

$$[lEMG(0)] \sim N\Big(\,[0]\,,\,[1]\,\Big)$$

The regime-switching model is given by:

$$\text{Initial regime probabilities} = \begin{bmatrix}1\\0\end{bmatrix}$$

$$\frac{Pr(p11)}{1-Pr(p11)} = p11$$
$$\frac{Pr(p21)}{1-Pr(p21)} = p21$$

We can also print out the model in R, instead of generating a Latex file, using the command *plotFormula*.

```
printex(rsmod, ParameterAs=rsmod$param.names, printInit=TRUE,printRS=TRUE,
        outFile="RSLinearDiscrete.tex")
tools::texi2pdf("RSLinearDiscrete.tex")
system(paste(getOption("pdfviewer"), "RSLinearDiscrete.pdf"))
```

0.9 Optimization Step and Results

Finally, it is time to cook dynr (i.e. fit our model through parameter optimization)!

```
yum <- dynr.cook(rsmod)
```

And serve!

```
summary(yum)
```

```
##             names  parameters       s.e.    t-value    ci.lower
## phi_0       phi_0  0.31414604 0.05506106  5.7054119  0.20622834
## phi_1       phi_1  0.92725308 0.02727326 33.9986131  0.87379847
## beta_0     beta_0  0.01190633 0.03974690  0.2995536 -0.06599617
## beta_1     beta_1  0.45385563 0.17534348  2.5883805  0.11018873
## mu_0         mu_0  2.98574402 0.08881840 33.6162779  2.81166315
## mu_1         mu_1  4.17992677 0.47141965  8.8666791  3.25596124
## dynNoise dynNoise  0.23228631 0.01500866 15.4768173  0.20286987
## p11           p11  4.49335395 0.58626479  7.6643762  3.34429607
## p21           p21 -4.50750828 0.71461456 -6.3076077 -5.90812708
##             ci.upper
## phi_0     0.42206373
## phi_1     0.98070770
## beta_0    0.08980882
## beta_1    0.79752253
## mu_0      3.15982488
## mu_1      5.10389231
## dynNoise  0.26170275
## p11       5.64241183
## p21      -3.10688949
##
## -2 log-likelihood value at convergence = 748.24
## AIC = 766.24
## BIC = 804.18
```