# DynareR: A Seamless Integration of R and Dynare

Sagiru Mati (PhD)

## About DynareR

DynareR is an R package that can run `Dynare` program from R Markdown.

## Requirements

Users need the following in order to knit this document:

1. Dynare 4.6.1 or above
2. Octave 5.2.0 or above
3. Dynare is installed in the standard location as follows:

- `/usr/lib/dynare/matlab` for `Linux`
- `/usr/lib/dynare/matlab` for `macOS`
- `c:/dynare/x.y/matlab` for `Windows`, where `x.y` is `Dynare` version number.

If `dynare` and `Octave` are installed in standard location, `DynareR` package will take care of the configurations, which include adding `matlab` directory to path, using the latest installed `dynare` and so on. Otherwise, users have to specify the `matlab` folder using `add_path` function, set the `Octave` path using the `set_octave_path` function, or set `dynare` version using the `set_dynare_version` function.

## Installation

DynareR can be installed using the following commands in R.

```r
install.packages("DynareR")

          OR

devtools::install_github('sagirumati/DynareR')
```

## Usage

Please load the DynareR package as follows:

````
```{r DynareR}
library(DynareR)
```
````

Then create a chunk for `dynare` (adopted from Dynare example file `bkk`) as shown below:

````
```{dynare bkk,eval=T}
/*
 * This file implements the multi-country RBC model with time to build,
 * described in Backus, Kehoe and Kydland (1992): "International Real Business
 * Cycles", Journal of Political Economy, 100(4), 745-775.
 *
 * The notation for the variable names are the same in this file than in the paper.
 * However the timing convention is different: we had to taken into account the
 * fact that in Dynare, if a variable is denoted at the current period, then
 * this variable must be also decided at the current period.
 * Concretely, here are the differences between the paper and the model file:
 * - z_t in the model file is equal to z_{t+1} in the paper
 * - k_t in the model file is equal to k_{t+J} in the paper
 * - s_t in the model file is equal to s_{J,t}=s_{J-1,t+1}=...=s_{1,t+J-1} in the paper
 *
 * The macroprocessor is used in this file to create a loop over countries.
 * Only two countries are used here (as in the paper), but it is easy to add
 * new countries in the corresponding macro-variable and completing the
 * calibration.
 *
````
````

```
 * The calibration is the same than in the paper. The results in terms of
 * moments of variables are very close to that of the paper (but not equal
 * since the authors a different solution method).
 *
 * This implementation was written by Sebastien Villemot. Please note that the
 * following copyright notice only applies to this Dynare implementation of the
 * model.
 */

/*
 * Copyright (C) 2010 Dynare Team
 *
 * This file is part of Dynare.
 *
 * Dynare is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * Dynare is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with Dynare.  If not, see <http://www.gnu.org/licenses/>.
 */

@#define countries = [ "H", "F" ]
@#define J = 4

@#for co in countries
var C_@{co} L_@{co} N_@{co} A_@{co} K_@{co} Z_@{co} X_@{co} LAMBDA_@{co} S_@{co} NX_@{co} Y_@

varexo E_@{co};

parameters beta_@{co} alpha_@{co} eta_@{co} mu_@{co} gamma_@{co} theta_@{co} nu_@{co} sigma_@
@#endfor

// Lagrange multiplier of aggregate constraint
var LGM;

parameters rho_@{countries[1]}_@{countries[2]} rho_@{countries[2]}_@{countries[1]};
```

```
model;
@#for co in countries

Y_@{co} = ((LAMBDA_@{co}*K_@{co}(-@{J})^theta_@{co}*N_@{co}^(1-theta_@{co}))^(-nu_@{co}) + si
K_@{co} = (1-delta_@{co})*K_@{co}(-1) + S_@{co};
X_@{co} =
@# for lag in (-J+1):0
          + phi_@{co}*S_@{co}(@{lag})
@# endfor
;

A_@{co} = (1-eta_@{co})*A_@{co}(-1) + N_@{co};
L_@{co} = 1 - alpha_@{co}*N_@{co} - (1-alpha_@{co})*eta_@{co}*A_@{co}(-1);

// Utility multiplied by gamma
# U_@{co} = (C_@{co}^mu_@{co}*L_@{co}^(1-mu_@{co}))^gamma_@{co};

// FOC with respect to consumption
psi_@{co}*mu_@{co}/C_@{co}*U_@{co} = LGM;

// FOC with respect to labor
// NOTE: this condition is only valid for alpha = 1
psi_@{co}*(1-mu_@{co})/L_@{co}*U_@{co}*(-alpha_@{co}) = - LGM * (1-theta_@{co})/N_@{co}*(LAMB

// FOC with respect to capital
@# for lag in 0:(J-1)
 +beta_@{co}^@{lag}*LGM(+@{lag})*phi_@{co}
@# endfor
@# for lag in 1:J
 -beta_@{co}^@{lag}*LGM(+@{lag})*phi_@{co}*(1-delta_@{co})
@# endfor
 = beta_@{co}^@{J}*LGM(+@{J})*theta_@{co}/K_@{co}*(LAMBDA_@{co}(+@{J})*K_@{co}^theta_@{co}*N_

// FOC with respect to stock of inventories
 LGM=beta_@{co}*LGM(+1)*(1+sigma_@{co}*Z_@{co}^(-nu_@{co}-1)*Y_@{co}(+1)^(1+nu_@{co}));

// Shock process
@# if co == countries[1]
@#  define alt_co = countries[2]
@# else
@#  define alt_co = countries[1]
@# endif
```

```
 (LAMBDA_@{co}-1) = rho_@{co}_@{co}*(LAMBDA_@{co}(-1)-1) + rho_@{co}_@{alt_co}*(LAMBDA_@{alt_

NX_@{co} = (Y_@{co} - (C_@{co} + X_@{co} + Z_@{co} - Z_@{co}(-1)))/Y_@{co};

@#endfor

// World ressource constraint
@#for co in countries
  +C_@{co} + X_@{co} + Z_@{co} - Z_@{co}(-1)
@#endfor
    =
@#for co in countries
  +Y_@{co}
@#endfor
    ;

end;

@#for co in countries
beta_@{co} = 0.99;
mu_@{co} = 0.34;
gamma_@{co} = -1.0;
alpha_@{co} = 1;
eta_@{co} = 0.5; // Irrelevant when alpha=1
theta_@{co} = 0.36;
nu_@{co} = 3;
sigma_@{co} = 0.01;
delta_@{co} = 0.025;
phi_@{co} = 1/@{J};
psi_@{co} = 0.5;
@#endfor

rho_H_H = 0.906;
rho_F_F = 0.906;
rho_H_F = 0.088;
rho_F_H = 0.088;

initval;
@#for co in countries
LAMBDA_@{co} = 1;
NX_@{co} = 0;
Z_@{co} = 1;
```

````
A_@{co} = 1;
L_@{co} = 0.5;
N_@{co} = 0.5;
Y_@{co} = 1;
K_@{co} = 1;
C_@{co} = 1;
S_@{co} = 1;
X_@{co} = 1;

E_@{co} = 0;
@#endfor

LGM = 1;
end;

shocks;
var E_H; stderr 0.00852;
var E_F; stderr 0.00852;
corr E_H, E_F = 0.258;
end;

steady;
check;

stoch_simul(order=1, hp_filter=1600);
```
````

The above chunk creates a Dynare program with the chunk's content, then automatically run Dynare, which will save Dynare outputs in the current directory.

Please note that DynareR uses the chunk name as the model name. So, the outpus of Dynare are saved in a folder with its respective chunk name. Thus a new folder `bkk/` will be created in your current working directory.

By default, `dynare` chunk imports log output as a list of dataframes, which can be accessed via `dynare$modelName`. Therefore to access the outputs of the `bkk` model produced by the `dynare` chunk, use `dynare$bkk`.

Use inline code `` `r dynare$bkk$moments[2,3]` `` to access the value of second row and third column of the `moments`, which is 0.0024.
````

## Plotting the IRF

The Impulse Response Function (IRF) is saved by default in `bkk/bkk/graphs/` folder with the IRF's name `bkk_IRF_E_H2.pdf`, where `bkk` is the Dynare model's name. Therefore, you need to add `stoch_simul(graph_format = (pdf))` to change the default saving behaviour of `Dynare` from `eps` to `pdf`.

## DynareR functions for base R

The DynareR package is also designed to work with base R. The following functions show how to work with DynareR outside the R Markdown or Quarto documents.

### The include_IRF function

Use this function to embed the graphs Impulse Response Function (IRF) in R Markdown or Quarto document.

The Impulse Response Function (IRF) of the `bkk` model can be fetched using the following R chunk. Note that only the last part of the IRF's name (`E_H2`) is needed, that is `bkk_IRF_` is excluded. Also note that `out.extra='trim={0cm 7cm 0cm 7cm},clip'` is used to trim the white space above and below the IRF.

````
```{r IRF,fig.cap="Another of figure generated from Dynare software"}
include_IRF("bkk","E_H2")

# Alternatively, use the path argument

```
````

```r
include_IRF(model="bkk",IRF = "E_H2")
#> PDF error: No display font for 'Symbol'
#> PDF error: No display font for 'ArialUnicode'

# Alternatively, use the path argument

include_IRF(path="bkk/bkk/graphs/bkk_IRF_E_H2.pdf")
#> PDF error: No display font for 'Symbol'
#> PDF error: No display font for 'ArialUnicode'
```

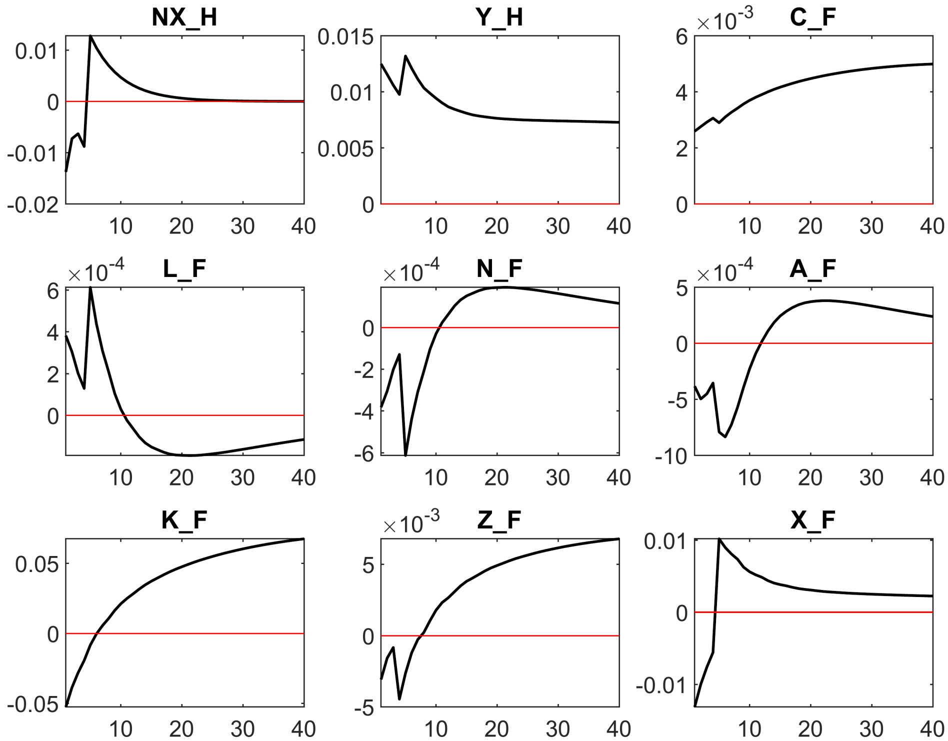


Figure 1: Another of figure generated from Dynare software

However, Dynare figure can only be dynamically included if the output format is pdf as Dynare produces pdf and eps graphs only.

### The write_dyn function

This function writes a new `dyn` file.

Use `write_dyn(code="code",model="someModel")` if you want the `Dynare` file to live in the current working directory. Use `write_dyn(code="code",model="path/to/someDirectory/someModel")` if you want the Dynare file to live in the path different from the current working directory.

```
dynareCodes='var y, c, k, a, h, b;
varexo e, u;
parameters beta, rho, alpha, delta, theta, psi, tau;
alpha = 0.36;
rho   = 0.95;
tau   = 0.025;
beta  = 0.99;
delta = 0.025;
```

```
psi   = 0;
theta = 2.95;
phi   = 0.1;
model;
c*theta*h^(1+psi)=(1-alpha)*y;
k = beta*(((exp(b)*c)/(exp(b(+1))*c(+1)))
          *(exp(b(+1))*alpha*y(+1)+(1-delta)*k));
y = exp(a)*(k(-1)^alpha)*(h^(1-alpha));
k = exp(b)*(y-c)+(1-delta)*k(-1);
a = rho*a(-1)+tau*b(-1) + e;
b = tau*a(-1)+rho*b(-1) + u;
end;
initval;
y = 1.08068253095672;
c = 0.80359242014163;
h = 0.29175631001732;
k = 11.08360443260358;
a = 0;
b = 0;
e = 0;
u = 0;
end;

shocks;
var e; stderr 0.009;
var u; stderr 0.009;
var e, u = phi*0.009*0.009;
end;

stoch_simul;'


write_dyn(code=dynareCodes, model="example1")

write_dyn(code=dynareCodes,model="DynareR/write_dyn/example1")
```

### The write_mod function

This function writes a new `mod` file.

Use `write_mod(code="code",model="someModel")` if you want the `Dynare` file to live in the current working directory. Use `write_mod(code="code",model="path/to/someDirectory/someModel")` if you want the Dynare file to live in the path different from the current working directory.

```
DynareCodes='var y, c, k, a, h, b;
varexo e, u;
parameters beta, rho, alpha, delta, theta, psi, tau;
alpha = 0.36;
rho   = 0.95;
tau   = 0.025;
beta  = 0.99;
delta = 0.025;
psi   = 0;
theta = 2.95;
phi   = 0.1;
model;
c*theta*h^(1+psi)=(1-alpha)*y;
k = beta*(((exp(b)*c)/(exp(b(+1))*c(+1)))
          *(exp(b(+1))*alpha*y(+1)+(1-delta)*k));
y = exp(a)*(k(-1)^alpha)*(h^(1-alpha));
k = exp(b)*(y-c)+(1-delta)*k(-1);
a = rho*a(-1)+tau*b(-1) + e;
b = tau*a(-1)+rho*b(-1) + u;
end;
initval;
y = 1.08068253095672;
c = 0.80359242014163;
h = 0.29175631001732;
k = 11.08360443260358;
a = 0;
b = 0;
e = 0;
u = 0;
end;

shocks;
var e; stderr 0.009;
var u; stderr 0.009;
var e, u = phi*0.009*0.009;
end;

stoch_simul;'


write_mod(model="example1",code=dynareCodes)
```

```r
write_mod(code=dynareCodes,model="DynareR/write_mod/example1")
```

### The run_dynare function

Create and run Dynare `mod` file

Use this function to create and run Dynare mod file. Use `run_dynare(code="code",model="someModel")` if you want the Dynare files to live in the current working directory. Use `run_dynare(code="code",model="path` if you want the Dynare files to live in the path different from the current working directory. Use `import_log=T` argument to return the `dynare` log file as list of dataframes in an environment `dynare`, which can be accessed via `dynare$modelName`.

```r
DynareCodes='var y, c, k, a, h, b;
varexo e, u;
parameters beta, rho, alpha, delta, theta, psi, tau;
alpha = 0.36;
rho   = 0.95;
tau   = 0.025;
beta  = 0.99;
delta = 0.025;
psi   = 0;
theta = 2.95;
phi   = 0.1;
model;
c*theta*h^(1+psi)=(1-alpha)*y;
k = beta*(((exp(b)*c)/(exp(b(+1))*c(+1)))
          *(exp(b(+1))*alpha*y(+1)+(1-delta)*k));
y = exp(a)*(k(-1)^alpha)*(h^(1-alpha));
k = exp(b)*(y-c)+(1-delta)*k(-1);
a = rho*a(-1)+tau*b(-1) + e;
b = tau*a(-1)+rho*b(-1) + u;
end;
initval;
y = 1.08068253095672;
c = 0.80359242014163;
h = 0.29175631001732;
k = 11.08360443260358;
a = 0;
b = 0;
e = 0;
u = 0;
```

```
end;

shocks;
var e; stderr 0.009;
var u; stderr 0.009;
var e, u = phi*0.009*0.009;
end;

stoch_simul;'

run_dynare(code=DynareCodes,model="example1",import_log = T)
run_dynare(code=DynareCodes,model="DynareR/run_dynare/example1")
```

### The run_models function

Run multiple existing `mod` or `dyn` files.

Use this function to execute multiple existing Dynare files. Use `run_models(model="someModel")` if the Dynare files live in the current working directory. Use `run_models(model="path/to/someDirectory/some` if the Dynare files live in the path different from the current working directory. Use `run_models()` to exectute all the `dynare` models in the current working directory. Use `run_models("path/to/someDirectory*)` to run all the `dynare` models in `path/to/someDirectory`.

Where `agtrend.mod`, `bkk.mod` and `example1.mod` are the Dynare model files (with `mod` or `dyn` extension), which live in the current working directory.

```
demo(agtrend)
#>
#>
#>  demo(agtrend)
#>  ---- ~~~~~~~
#>
#> > # We use "agtrend" of the Dynare example files to illustrate
#> > #how to use this function
#> >
#> >
#> > library(DynareR)
#>
#> > DynareCodes='/*
#> +  * This file replicates the model studied in:
#> +  * Aguiar, Mark and Gopinath, Gita (2004): "Emerging Market Business Cycles:
```

```
#> +  * The Cycle is the Trend" (NBER WP 10734). It is different from version published
#> +  * in the Journal of Political Economy.
#> +  *
#> +  * This model file is intended to show the capabilities of the Dynare macro
#> +  * languange. It is not intended to provide a full replication of the original
#> +  * paper due to some differences in model calibration. In
#> +  * particular, this mod-file does not calibrate the share of debt to GDP
#> +  * to 0.1 as this would require the use of a steady state file. Rather, the
#> +  * absolute value of debt is set to 0.1. Given that output is close to 1 in
#> +  * the benchmark specification, this results in only a small difference to
#> +  * the working paper.
#> +  * The mod-file reproduces Figure 4 of the working paper, which displays the
#> +  * model response to 1 percent shock to trend and cyclical TFP.
#> +  *
#> +  * This implementation was written by S?bastien Villemot and Johannes Pfeifer.
#> +  * Please note that the following copyright notice only applies to this Dynare
#> +  * implementation of the model.
#> +  */
#> +
#> + /*
#> +  * Copyright (C) 2012-13 Dynare Team
#> +  *
#> +  * This file is part of Dynare.
#> +  *
#> +  * Dynare is free software: you can redistribute it and/or modify
#> +  * it under the terms of the GNU General Public License as published by
#> +  * the Free Software Foundation, either version 3 of the License, or
#> +  * (at your option) any later version.
#> +  *
#> +  * Dynare is distributed in the hope that it will be useful,
#> +  * but WITHOUT ANY WARRANTY; without even the implied warranty of
#> +  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#> +  * GNU General Public License for more details.
#> +  *
#> +  * You should have received a copy of the GNU General Public License
#> +  * along with Dynare.  If not, see <http://www.gnu.org/licenses/>.
#> +  */
#> +
#> +
#> + // Set the following variable to 0 to get Cobb-Douglas utility
#> + @#define ghh = 1
#> + // Set the following variable to 0 to get the calibration for Canada
```

```
#> + @#define mexico = 1
#> +
#> + var c k y b q g l u z uc ul f c_y tb_y i_y;
#> +
#> + varexo eps_z eps_g;
#> +
#> + parameters mu_g sigma rho_g sigma_g delta phi psi b_star alpha rho_z sigma_z r_star beta
#> +
#> + // Benchmark parameter values (table 3)
#> + @#if ghh == 1
#> + parameters tau nu;
#> + tau = 1.4;
#> + nu = 1.6;
#> + @#else
#> + parameters gamma;
#> + gamma = 0.36;
#> + @#endif
#> +
#> + alpha = 0.68;
#> + sigma = 2;
#> + delta = 0.03;
#> + beta = 0.98;
#> + psi = 0.001;
#> + b_star = 0.1; //taken here as the steady state value of debt; in the original paper, thi
#> +
#> + // Estimated parameters (table 4)
#> + @#if mexico == 1
#> + @# if ghh == 1
#> + mu_g = 1.006;
#> + sigma_z = 0.0041;
#> + rho_z = 0.94;
#> + sigma_g = 0.0109;
#> + rho_g = 0.72;
#> + phi = 3.79;
#> + @# else
#> + mu_g = 1.005;
#> + sigma_z = 0.0046;
#> + rho_z = 0.94;
#> + sigma_g = 0.025;
#> + rho_g = 0.06;
#> + phi = 2.82;
#> + @# endif
```

```
#> + @#else
#> + // Canada
#> + @# if ghh == 1
#> + mu_g = 1.007;
#> + sigma_z = 0.0057;
#> + rho_z = 0.88;
#> + sigma_g = 0.0014;
#> + rho_g = 0.94;
#> + phi = 2.63;
#> + @# else
#> + mu_g = 1.007;
#> + sigma_z = 0.0072;
#> + rho_z = 0.96;
#> + sigma_g = 0.0044;
#> + rho_g = 0.50;
#> + phi = 3.76;
#> + @# endif
#> + @#endif
#> +
#> + @#if ghh == 1
#> + r_star = mu_g^sigma/beta - 1;
#> + @#else
#> + r_star = mu_g^(1-gamma*(1-sigma))/beta - 1;
#> + @#endif
#> +
#> + model; //equation numbers refer to numbers in the working paper version
#> + y=exp(z)*k(-1)^(1-alpha)*(g*l)^alpha; // Production technology (1)
#> + z = rho_z*z(-1)+sigma_z*eps_z; // Transitory shock (2)
#> + log(g) = (1-rho_g)*log(mu_g)+rho_g*log(g(-1))+sigma_g*eps_g; // Trend shock
#> + @#if ghh == 1
#> + u = (c-tau*l^nu)^(1-sigma)/(1-sigma); // GHH utility (3)
#> + uc = (c - tau*l^nu)^(-sigma);
#> + ul = -tau*nu*l^(nu-1)*(c - tau*l^nu)^(-sigma);
#> + f = beta*g^(1-sigma);
#> + @#else
#> + u = (c^gamma*(1-l)^(1-gamma))^(1-sigma)/(1-sigma); // Cobb-Douglas utility (4)
#> + uc = gamma*u/c*(1-sigma);
#> + ul = -(1-gamma)*u/(1-l)*(1-sigma);
#> + f = beta*g^(gamma*(1-sigma));
#> + @#endif
#> + c+g*k=y+(1-delta)*k(-1)-phi/2*(g*k/k(-1)-mu_g)^2*k(-1)-b(-1)+q*g*b; // Resource constrai
#> + 1/q = 1+r_star+psi*(exp(b-b_star)-1); // Price of debt (6)
```

```
#> + uc*(1+phi*(g*k/k(-1)-mu_g))*g=f*uc(+1)*(1-delta+(1-alpha)*y(+1)/k+phi/2*(g(+1)*k(+1)/k-m
#> + ul+uc*alpha*y/l=0; // Leisure-consumption arbitrage (11)
#> + uc*g*q=f*uc(+1); // Euler equation (12)
#> +
#> + //definition of auxilary variables to be plotted
#> + tb_y = (b(-1)-g*q*b)/y; // Trade balance to GDP ratio, not logged as it can be negative
#> + c_y = log(c/y); // Consumption to GDP ratio, logged to be in percent
#> + i_y = log((g*k-(1-delta)*k(-1)+phi/2*(g*k/k(-1)-mu_g)^2*k(-1))/y); // Investment to GDP
#> + end;
#> +
#> + initval;
#> + q = 1/(1+r_star);
#> + b = b_star;
#> + z = 0;
#> + g = mu_g;
#> +
#> + c = 0.583095;
#> + k = 4.02387;
#> + y = 0.721195;
#> + l = 0.321155;
#> +
#> + @#if ghh == 1
#> + u = (c-tau*l^nu)^(1-sigma)/(1-sigma);
#> + uc = (c - tau*l^nu)^(-sigma);
#> + ul = -tau*nu*l^(nu-1)*(c - tau*l)^(-sigma);
#> + f = beta*g^(1-sigma);
#> + @#else
#> + u = (c^gamma*(1-l)^(1-gamma))^(1-sigma)/(1-sigma);
#> + uc = gamma*u/c*(1-sigma);
#> + ul = -(1-gamma)*u/(1-l)*(1-sigma);
#> + f = beta*g^(gamma*(1-sigma));
#> + @#endif
#> +
#> + tb_y = (b-g*q*b)/y;
#> + c_y = c/y;
#> + i_y = (g*k-(1-delta)*k)/y;
#> + end;
#> +
#> + shocks;
#> + var eps_g; stderr 1/sigma_g/100; // use a 1 percent shock
#> + var eps_z; stderr 1/sigma_z/100; // use a 1 percent shock
#> + end;
```

```
#> +
#> + steady;
#> +
#> + check;
#> +
#> + // Plot impulse response functions (Figure 4)
#> + stoch_simul(order=1) tb_y c_y i_y;'
#>
#> > run_dynare(code=DynareCodes,model="agtrend")
#>
#> > # You can create an absolute or relative path for the DynareR files.
#> > # The following writes and run mod file in "DynareR/run_dynare/"  folder
#> > # relative to the current path.
#> >
#> >
#> > run_dynare(code=DynareCodes,model="DynareR/run_dynare/agtrend")
demo(bkk)
#>
#>
#>  demo(bkk)
#>  ---- ~~~
#>
#> > # We use "bkk" of the Dynare example files to illustrate
#> > #how to use this function
#> >
#> >
#> > library(DynareR)
#>
#> > DynareCodes='/*
#> +  * This file implements the multi-country RBC model with time to build,
#> +  * described in Backus, Kehoe and Kydland (1992): "International Real Business
#> +  * Cycles", Journal of Political Economy, 100(4), 745-775.
#> +  *
#> +  * The notation for the variable names are the same in this file than in the paper.
#> +  * However the timing convention is different: we had to taken into account the
#> +  * fact that in Dynare, if a variable is denoted at the current period, then
#> +  * this variable must be also decided at the current period.
#> +  * Concretely, here are the differences between the paper and the model file:
#> +  * - z_t in the model file is equal to z_{t+1} in the paper
#> +  * - k_t in the model file is equal to k_{t+J} in the paper
#> +  * - s_t in the model file is equal to s_{J,t}=s_{J-1,t+1}=...=s_{1,t+J-1} in the paper
#> +  *
```

```
#> +  * The macroprocessor is used in this file to create a loop over countries.
#> +  * Only two countries are used here (as in the paper), but it is easy to add
#> +  * new countries in the corresponding macro-variable and completing the
#> +  * calibration.
#> +  *
#> +  * The calibration is the same than in the paper. The results in terms of
#> +  * moments of variables are very close to that of the paper (but not equal
#> +  * since the authors a different solution method).
#> +  *
#> +  * This implementation was written by Sebastien Villemot. Please note that the
#> +  * following copyright notice only applies to this Dynare implementation of the
#> +  * model.
#> +  */
#> +
#> + /*
#> +  * Copyright (C) 2010 Dynare Team
#> +  *
#> +  * This file is part of Dynare.
#> +  *
#> +  * Dynare is free software: you can redistribute it and/or modify
#> +  * it under the terms of the GNU General Public License as published by
#> +  * the Free Software Foundation, either version 3 of the License, or
#> +  * (at your option) any later version.
#> +  *
#> +  * Dynare is distributed in the hope that it will be useful,
#> +  * but WITHOUT ANY WARRANTY; without even the implied warranty of
#> +  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#> +  * GNU General Public License for more details.
#> +  *
#> +  * You should have received a copy of the GNU General Public License
#> +  * along with Dynare.  If not, see <http://www.gnu.org/licenses/>.
#> +  */
#> +
#> + @#define countries = [ "H", "F" ]
#> + @#define J = 4
#> +
#> + @#for co in countries
#> + var C_@{co} L_@{co} N_@{co} A_@{co} K_@{co} Z_@{co} X_@{co} LAMBDA_@{co} S_@{co} NX_@{co
#> +
#> + varexo E_@{co};
#> +
#> + parameters beta_@{co} alpha_@{co} eta_@{co} mu_@{co} gamma_@{co} theta_@{co} nu_@{co} si
```

```
#> + @#endfor
#> +
#> + // Lagrange multiplier of aggregate constraint
#> + var LGM;
#> +
#> + parameters rho_@{countries[1]}_@{countries[2]} rho_@{countries[2]}_@{countries[1]};
#> +
#> + model;
#> + @#for co in countries
#> +
#> + Y_@{co} = ((LAMBDA_@{co}*K_@{co}(-@{J})^theta_@{co}*N_@{co}^(1-theta_@{co}))^(-nu_@{co}
#> + K_@{co} = (1-delta_@{co})*K_@{co}(-1) + S_@{co};
#> + X_@{co} =
#> + @# for lag in (-J+1):0
#> +           + phi_@{co}*S_@{co}(@{lag})
#> + @# endfor
#> + ;
#> +
#> + A_@{co} = (1-eta_@{co})*A_@{co}(-1) + N_@{co};
#> + L_@{co} = 1 - alpha_@{co}*N_@{co} - (1-alpha_@{co})*eta_@{co}*A_@{co}(-1);
#> +
#> + // Utility multiplied by gamma
#> + # U_@{co} = (C_@{co}^mu_@{co}*L_@{co}^(1-mu_@{co}))^gamma_@{co};
#> +
#> + // FOC with respect to consumption
#> + psi_@{co}*mu_@{co}/C_@{co}*U_@{co} = LGM;
#> +
#> + // FOC with respect to labor
#> + // NOTE: this condition is only valid for alpha = 1
#> + psi_@{co}*(1-mu_@{co})/L_@{co}*U_@{co}*(-alpha_@{co}) = - LGM * (1-theta_@{co})/N_@{co}
#> +
#> + // FOC with respect to capital
#> + @# for lag in 0:(J-1)
#> +  +beta_@{co}^@{lag}*LGM(+@{lag})*phi_@{co}
#> + @# endfor
#> + @# for lag in 1:J
#> +  -beta_@{co}^@{lag}*LGM(+@{lag})*phi_@{co}*(1-delta_@{co})
#> + @# endfor
#> +  = beta_@{co}^@{J}*LGM(+@{J})*theta_@{co}/K_@{co}*(LAMBDA_@{co}(+@{J})*K_@{co}^theta_@{
#> +
#> + // FOC with respect to stock of inventories
#> +  LGM=beta_@{co}*LGM(+1)*(1+sigma_@{co}*Z_@{co}^(-nu_@{co}-1)*Y_@{co}(+1)^(1+nu_@{co}));
```

```
#> +
#> + // Shock process
#> + @# if co == countries[1]
#> + @#  define alt_co = countries[2]
#> + @# else
#> + @#  define alt_co = countries[1]
#> + @# endif
#> +  (LAMBDA_@{co}-1) = rho_@{co}_@{co}*(LAMBDA_@{co}(-1)-1) + rho_@{co}_@{alt_co}*(LAMBDA_@
#> +
#> +
#> + NX_@{co} = (Y_@{co} - (C_@{co} + X_@{co} + Z_@{co} - Z_@{co}(-1)))/Y_@{co};
#> +
#> + @#endfor
#> +
#> + // World ressource constraint
#> + @#for co in countries
#> +   +C_@{co} + X_@{co} + Z_@{co} - Z_@{co}(-1)
#> + @#endfor
#> +     =
#> + @#for co in countries
#> +   +Y_@{co}
#> + @#endfor
#> +     ;
#> +
#> + end;
#> +
#> + @#for co in countries
#> + beta_@{co} = 0.99;
#> + mu_@{co} = 0.34;
#> + gamma_@{co} = -1.0;
#> + alpha_@{co} = 1;
#> + eta_@{co} = 0.5; // Irrelevant when alpha=1
#> + theta_@{co} = 0.36;
#> + nu_@{co} = 3;
#> + sigma_@{co} = 0.01;
#> + delta_@{co} = 0.025;
#> + phi_@{co} = 1/@{J};
#> + psi_@{co} = 0.5;
#> + @#endfor
#> +
#> + rho_H_H = 0.906;
#> + rho_F_F = 0.906;
```

```
#> + rho_H_F = 0.088;
#> + rho_F_H = 0.088;
#> +
#> + initval;
#> + @#for co in countries
#> + LAMBDA_@{co} = 1;
#> + NX_@{co} = 0;
#> + Z_@{co} = 1;
#> + A_@{co} = 1;
#> + L_@{co} = 0.5;
#> + N_@{co} = 0.5;
#> + Y_@{co} = 1;
#> + K_@{co} = 1;
#> + C_@{co} = 1;
#> + S_@{co} = 1;
#> + X_@{co} = 1;
#> +
#> + E_@{co} = 0;
#> + @#endfor
#> +
#> + LGM = 1;
#> + end;
#> +
#> + shocks;
#> + var E_H; stderr 0.00852;
#> + var E_F; stderr 0.00852;
#> + corr E_H, E_F = 0.258;
#> + end;
#> +
#> + steady;
#> + check;
#> +
#> + stoch_simul(order=1, hp_filter=1600,graph_format=pdf);
#> + '
#>
#> > run_dynare(code=DynareCodes,model="bkk")
#>
#> > # You can create an absolute or relative path for the DynareR files.
#> > # The following writes and run mod file in "DynareR/run_dynare/"  folder
#> > # relative to the current path.
#> >
#> >
```

```
#> > run_dynare(code=DynareCodes,model="DynareR/run_dynare/bkk")
demo(example1)
#>
#>
#>  demo(example1)
#>  ---- ~~~~~~~~
#>
#> > # We use "example1" of the Dynare example files to illustrate
#> > #how to use this function
#> >
#> > FileName<-"example1"
#>
#> > library(DynareR)
#>
#> > DynareCodes='/*
#> +  * Example 1 from F. Collard (2001): "Stochastic simulations with DYNARE:
#> +  * A practical guide" (see "guide.pdf" in the documentation directory).
#> +  */
#> +
#> + /*
#> +  * Copyright (C) 2001-2010 Dynare Team
#> +  *
#> +  * This file is part of Dynare.
#> +  *
#> +  * Dynare is free software: you can redistribute it and/or modify
#> +  * it under the terms of the GNU General Public License as published by
#> +  * the Free Software Foundation, either version 3 of the License, or
#> +  * (at your option) any later version.
#> +  *
#> +  * Dynare is distributed in the hope that it will be useful,
#> +  * but WITHOUT ANY WARRANTY; without even the implied warranty of
#> +  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#> +  * GNU General Public License for more details.
#> +  *
#> +  * You should have received a copy of the GNU General Public License
#> +  * along with Dynare.  If not, see <http://www.gnu.org/licenses/>.
#> +  */
#> +
#> +
#> + var y, c, k, a, h, b;
#> + varexo e, u;
#> +
```

```
#> + parameters beta, rho, alpha, delta, theta, psi, tau;
#> +
#> + alpha = 0.36;
#> + rho   = 0.95;
#> + tau   = 0.025;
#> + beta  = 0.99;
#> + delta = 0.025;
#> + psi   = 0;
#> + theta = 2.95;
#> +
#> + phi   = 0.1;
#> +
#> + model;
#> + c*theta*h^(1+psi)=(1-alpha)*y;
#> + k = beta*(((exp(b)*c)/(exp(b(+1))*c(+1)))
#> +     *(exp(b(+1))*alpha*y(+1)+(1-delta)*k));
#> + y = exp(a)*(k(-1)^alpha)*(h^(1-alpha));
#> + k = exp(b)*(y-c)+(1-delta)*k(-1);
#> + a = rho*a(-1)+tau*b(-1) + e;
#> + b = tau*a(-1)+rho*b(-1) + u;
#> + end;
#> +
#> + initval;
#> + y = 1.08068253095672;
#> + c = 0.80359242014163;
#> + h = 0.29175631001732;
#> + k = 11.08360443260358;
#> + a = 0;
#> + b = 0;
#> + e = 0;
#> + u = 0;
#> + end;
#> +
#> + shocks;
#> + var e; stderr 0.009;
#> + var u; stderr 0.009;
#> + var e, u = phi*0.009*0.009;
#> + end;
#> +
#> + stoch_simul;'
#>
#> > run_dynare(code=DynareCodes,model = "example1")
```

```
#>
#> > # You can create an absolute or relative path for the DynareR files.
#> > # The following writes and run mod file in "DynareR/run_dynare/"  folder
#> > # relative to the current path.
#> >
#> >
#> >
#> > run_dynare(code=DynareCodes,model = "DynareR/run_dynare/example1")

# Provide the list of the `Dynare` files in a vector
# Ensure that "agtrend.mod", "bkk.mod" and "example1.mod"
# live in the current working directory

# Copy the dynare files to the current working directory

lapply(c("agtrend","bkk","example1"),\(x) file.copy(paste0(x,"/",x,".mod"),"."))
#> [[1]]
#> [1] TRUE
#>
#> [[2]]
#> [1] TRUE
#>
#> [[3]]
#> [1] TRUE

run_models(c("agtrend","bkk","example1")) # Run the models in the vector.
```

To run all `Dynare` models that live in the current working directory, use the following:

```
run_models() # Run all models in Current Working Directory.
```

To run all `Dynare` models that live in particular path (for example 'DynareR/run_dynare/' folder), use the following:

```
# Copy the dynare files to the 'DynareR/run_dynare' directory

lapply(c("agtrend","bkk","example1"),\(x) file.copy(paste0(x,".mod"),"DynareR/run_dynare"))
#> [[1]]
#> [1] TRUE
#>
#> [[2]]
```

```
#> [1] TRUE
#>
#> [[3]]
#> [1] TRUE

run_models(model = 'DynareR/run_dynare*') # notice the * at the end
```

## import_log function

This function returns the `dynare` log output as a list of dataframes, which include `summary`, `shocks`, `policy`, `moments`, `decomposition`, `correlation` and `autocorrelation`. The list is accessible via `dynare$modelName`. if the model name is `bkk`, the policy variables can be obtained via `dynare$bkk$policy` as a dataframe.

```
import_log(model="bkk")
#> NULL

import_log(path="bkk/bkk.log")
#> NULL

knitr::kable(dynare$bkk$autocorrelation)  %>% kableExtra::kable_styling(latex_options = c("ba
 kableExtra::footnote(general="Some footnote with equation $\\alpha x^2+\\beta x+c=0$", gene
kableExtra::row_spec(0,bold=T)
#> Warning in styling_latex_scale(out, table_info, "down"): Longtable cannot be
#> resized.
```

| **Order** | **X1** | **X2** | **X3** | **X4** | **X5** |
|---|---|---|---|---|---|
| C_H | 0.7277 | 0.4968 | 0.3105 | 0.1655 | 0.0268 |
| L_H | 0.6843 | 0.4628 | 0.3519 | 0.3262 | 0.0676 |
| N_H | 0.6843 | 0.4628 | 0.3519 | 0.3262 | 0.0676 |
| A_H | 0.8817 | 0.7001 | 0.5277 | 0.3600 | 0.1320 |
| K_H | 0.6471 | 0.3737 | 0.1679 | 0.0176 | -0.0905 |
| Z_H | 0.7119 | 0.5339 | 0.4710 | 0.1904 | -0.0219 |
| X_H | 0.6202 | 0.2743 | -0.0522 | -0.3742 | -0.3032 |
| LAMBDA_H | 0.6933 | 0.4421 | 0.2405 | 0.0824 | -0.0377 |
| S_H | -0.1118 | -0.0955 | -0.0784 | -0.0598 | -0.0506 |
| NX_H | 0.5027 | 0.2569 | 0.0621 | -0.3409 | -0.2835 |
| Y_H | 0.6868 | 0.4605 | 0.3238 | 0.2646 | 0.0490 |

| | | | | | |
|---|---|---|---|---|---|
| C_F | 0.7277 | 0.4968 | 0.3105 | 0.1655 | 0.0268 |
| L_F | 0.6843 | 0.4628 | 0.3519 | 0.3262 | 0.0676 |
| N_F | 0.6843 | 0.4628 | 0.3519 | 0.3262 | 0.0676 |
| A_F | 0.8817 | 0.7001 | 0.5277 | 0.3600 | 0.1320 |
| K_F | 0.6471 | 0.3737 | 0.1679 | 0.0176 | -0.0905 |
| Z_F | 0.7119 | 0.5339 | 0.4710 | 0.1904 | -0.0219 |
| X_F | 0.6202 | 0.2743 | -0.0522 | -0.3742 | -0.3032 |
| LAMBDA_F | 0.6933 | 0.4421 | 0.2405 | 0.0824 | -0.0377 |
| S_F | -0.1118 | -0.0955 | -0.0784 | -0.0598 | -0.0506 |
| NX_F | 0.5027 | 0.2569 | 0.0621 | -0.3409 | -0.2835 |
| Y_F | 0.6868 | 0.4605 | 0.3238 | 0.2646 | 0.0490 |
| LGM | 0.7367 | 0.5080 | 0.3158 | 0.1585 | 0.0289 |

* Some footnote with equation $alphax^2 + betax + c = 0$

## set_dynare_version function

On Windows, you can set the version of dynare you want to use. By default, `DynareR` package does this for you if the dynare version ranges from 4.6.1 to 9.9. However, if you are using the development version of `dynare`, for example version `6-unstable-2022-04-03-0800-700a0e3a`, you can override the default as follows

```
set_dynare_version("6-unstable-2022-04-03-0800-700a0e3a")
```

## set_octave_path function

You can use this function if `Octave` is not installed in the standard location

```
set_octave_path('C:/Program Files/GNU Octave/Octave-6.4.0/mingw64/bin/octave20.exe')
```

## add_path function

This function is a wrapper of `addpath` in `Octave`. If `dynare` is not installed in the standard location, use this function to add the `matlab` subdirectory. By default, `DynareR` does this for if `dynare` is installed in the standard location.

```
add_path('/usr/lib/dynare/matlab')#  Default for Linux

add_path('c:/dynare/5.1/matlab') # Default for Windows, but 5.1 can change if later version o
# `Dynare` is installed.

add_path('/usr/lib/dynare/matlab') # Default for macOS
```

## Demo

The demo files are included and can be accessed via demo(package="DynareR")

```
demo(run_dynare)
#>
#>
#>  demo(run_dynare)
#>  ---- ~~~~~~~~~~
#>
#> > # We use "example1" of the Dynare example files to illustrate
#> > #how to use this function
#> >
#> >
#> > library(DynareR)
#>
#> > DynareCodes='var y, c, k, a, h, b;
#> + varexo e, u;
#> +
#> + parameters beta, rho, alpha, delta, theta, psi, tau;
#> +
#> + alpha = 0.36;
#> + rho   = 0.95;
#> + tau   = 0.025;
#> + beta  = 0.99;
#> + delta = 0.025;
#> + psi   = 0;
#> + theta = 2.95;
#> +
#> + phi   = 0.1;
#> +
#> + model;
#> + c*theta*h^(1+psi)=(1-alpha)*y;
```

```
#> + k = beta*(((exp(b)*c)/(exp(b(+1))*c(+1)))
#> +           *(exp(b(+1))*alpha*y(+1)+(1-delta)*k));
#> + y = exp(a)*(k(-1)^alpha)*(h^(1-alpha));
#> + k = exp(b)*(y-c)+(1-delta)*k(-1);
#> + a = rho*a(-1)+tau*b(-1) + e;
#> + b = tau*a(-1)+rho*b(-1) + u;
#> + end;
#> +
#> + initval;
#> + y = 1.08068253095672;
#> + c = 0.80359242014163;
#> + h = 0.29175631001732;
#> + k = 11.08360443260358;
#> + a = 0;
#> + b = 0;
#> + e = 0;
#> + u = 0;
#> + end;
#> +
#> + shocks;
#> + var e; stderr 0.009;
#> + var u; stderr 0.009;
#> + var e, u = phi*0.009*0.009;
#> + end;
#> +
#> + stoch_simul;'
#>
#> > # You can create an absolute or relative path for the DynareR files.
#> > # The following writes and run mod file in "DynareR/run_dynare/"  folder
#> > # relative to the current path.
#> >
#> >
#> >
#> > run_dynare(code=DynareCodes,model = "run_dynare")
#>
#> > run_dynare(code=DynareCodes,model = "DynareR/run_dynare/run_dynare")
demo(run_models)
#>
#>
#>  demo(run_models)
#>  ---- ~~~~~~~~~~
#>
```

```
#> > library(DynareR)
#>
#> > # This file should be the last to execute
#> >
#> > library(DynareR)
#>
#> > # Copy the dynare files to the current working directory
#> >
#> > demo(agtrend)
#>
#>
#>  demo(agtrend)
#>  ---- ~~~~~~~
#>
#> > # We use "agtrend" of the Dynare example files to illustrate
#> > #how to use this function
#> >
#> >
#> > library(DynareR)
#>
#> > DynareCodes='/*
#> +  * This file replicates the model studied in:
#> +  * Aguiar, Mark and Gopinath, Gita (2004): "Emerging Market Business Cycles:
#> +  * The Cycle is the Trend" (NBER WP 10734). It is different from version published
#> +  * in the Journal of Political Economy.
#> +  *
#> +  * This model file is intended to show the capabilities of the Dynare macro
#> +  * languange. It is not intended to provide a full replication of the original
#> +  * paper due to some differences in model calibration. In
#> +  * particular, this mod-file does not calibrate the share of debt to GDP
#> +  * to 0.1 as this would require the use of a steady state file. Rather, the
#> +  * absolute value of debt is set to 0.1. Given that output is close to 1 in
#> +  * the benchmark specification, this results in only a small difference to
#> +  * the working paper.
#> +  * The mod-file reproduces Figure 4 of the working paper, which displays the
#> +  * model response to 1 percent shock to trend and cyclical TFP.
#> +  *
#> +  * This implementation was written by S?bastien Villemot and Johannes Pfeifer.
#> +  * Please note that the following copyright notice only applies to this Dynare
#> +  * implementation of the model.
#> +  */
#> +
```

```
#> + /*
#> +  * Copyright (C) 2012-13 Dynare Team
#> +  *
#> +  * This file is part of Dynare.
#> +  *
#> +  * Dynare is free software: you can redistribute it and/or modify
#> +  * it under the terms of the GNU General Public License as published by
#> +  * the Free Software Foundation, either version 3 of the License, or
#> +  * (at your option) any later version.
#> +  *
#> +  * Dynare is distributed in the hope that it will be useful,
#> +  * but WITHOUT ANY WARRANTY; without even the implied warranty of
#> +  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#> +  * GNU General Public License for more details.
#> +  *
#> +  * You should have received a copy of the GNU General Public License
#> +  * along with Dynare.  If not, see <http://www.gnu.org/licenses/>.
#> +  */
#> +
#> +
#> + // Set the following variable to 0 to get Cobb-Douglas utility
#> + @#define ghh = 1
#> + // Set the following variable to 0 to get the calibration for Canada
#> + @#define mexico = 1
#> +
#> + var c k y b q g l u z uc ul f c_y tb_y i_y;
#> +
#> + varexo eps_z eps_g;
#> +
#> + parameters mu_g sigma rho_g sigma_g delta phi psi b_star alpha rho_z sigma_z r_star beta
#> +
#> + // Benchmark parameter values (table 3)
#> + @#if ghh == 1
#> + parameters tau nu;
#> + tau = 1.4;
#> + nu = 1.6;
#> + @#else
#> + parameters gamma;
#> + gamma = 0.36;
#> + @#endif
#> +
#> + alpha = 0.68;
```

```
#> + sigma = 2;
#> + delta = 0.03;
#> + beta = 0.98;
#> + psi = 0.001;
#> + b_star = 0.1; //taken here as the steady state value of debt; in the original paper, thi
#> +
#> + // Estimated parameters (table 4)
#> + @#if mexico == 1
#> + @# if ghh == 1
#> + mu_g = 1.006;
#> + sigma_z = 0.0041;
#> + rho_z = 0.94;
#> + sigma_g = 0.0109;
#> + rho_g = 0.72;
#> + phi = 3.79;
#> + @# else
#> + mu_g = 1.005;
#> + sigma_z = 0.0046;
#> + rho_z = 0.94;
#> + sigma_g = 0.025;
#> + rho_g = 0.06;
#> + phi = 2.82;
#> + @# endif
#> + @#else
#> + // Canada
#> + @# if ghh == 1
#> + mu_g = 1.007;
#> + sigma_z = 0.0057;
#> + rho_z = 0.88;
#> + sigma_g = 0.0014;
#> + rho_g = 0.94;
#> + phi = 2.63;
#> + @# else
#> + mu_g = 1.007;
#> + sigma_z = 0.0072;
#> + rho_z = 0.96;
#> + sigma_g = 0.0044;
#> + rho_g = 0.50;
#> + phi = 3.76;
#> + @# endif
#> + @#endif
#> +
```

```
#> + @#if ghh == 1
#> + r_star = mu_g^sigma/beta - 1;
#> + @#else
#> + r_star = mu_g^(1-gamma*(1-sigma))/beta - 1;
#> + @#endif
#> +
#> + model; //equation numbers refer to numbers in the working paper version
#> + y=exp(z)*k(-1)^(1-alpha)*(g*l)^alpha; // Production technology (1)
#> + z = rho_z*z(-1)+sigma_z*eps_z; // Transitory shock (2)
#> + log(g) = (1-rho_g)*log(mu_g)+rho_g*log(g(-1))+sigma_g*eps_g; // Trend shock
#> + @#if ghh == 1
#> + u = (c-tau*l^nu)^(1-sigma)/(1-sigma); // GHH utility (3)
#> + uc = (c - tau*l^nu)^(-sigma);
#> + ul = -tau*nu*l^(nu-1)*(c - tau*l^nu)^(-sigma);
#> + f = beta*g^(1-sigma);
#> + @#else
#> + u = (c^gamma*(1-l)^(1-gamma))^(1-sigma)/(1-sigma); // Cobb-Douglas utility (4)
#> + uc = gamma*u/c*(1-sigma);
#> + ul = -(1-gamma)*u/(1-l)*(1-sigma);
#> + f = beta*g^(gamma*(1-sigma));
#> + @#endif
#> + c+g*k=y+(1-delta)*k(-1)-phi/2*(g*k/k(-1)-mu_g)^2*k(-1)-b(-1)+q*g*b; // Resource constrai
#> + 1/q = 1+r_star+psi*(exp(b-b_star)-1); // Price of debt (6)
#> + uc*(1+phi*(g*k/k(-1)-mu_g))*g=f*uc(+1)*(1-delta+(1-alpha)*y(+1)/k+phi/2*(g(+1)*k(+1)/k-m
#> + ul+uc*alpha*y/l=0; // Leisure-consumption arbitrage (11)
#> + uc*g*q=f*uc(+1); // Euler equation (12)
#> +
#> + //definition of auxilary variables to be plotted
#> + tb_y = (b(-1)-g*q*b)/y; // Trade balance to GDP ratio, not logged as it can be negative
#> + c_y = log(c/y); // Consumption to GDP ratio, logged to be in percent
#> + i_y = log((g*k-(1-delta)*k(-1)+phi/2*(g*k/k(-1)-mu_g)^2*k(-1))/y); // Investment to GDP
#> + end;
#> +
#> + initval;
#> + q = 1/(1+r_star);
#> + b = b_star;
#> + z = 0;
#> + g = mu_g;
#> +
#> + c = 0.583095;
#> + k = 4.02387;
#> + y = 0.721195;
```

```
#> + l = 0.321155;
#> +
#> + @#if ghh == 1
#> + u = (c-tau*l^nu)^(1-sigma)/(1-sigma);
#> + uc = (c - tau*l^nu)^(-sigma);
#> + ul = -tau*nu*l^(nu-1)*(c - tau*l)^(-sigma);
#> + f = beta*g^(1-sigma);
#> + @#else
#> + u = (c^gamma*(1-l)^(1-gamma))^(1-sigma)/(1-sigma);
#> + uc = gamma*u/c*(1-sigma);
#> + ul = -(1-gamma)*u/(1-l)*(1-sigma);
#> + f = beta*g^(gamma*(1-sigma));
#> + @#endif
#> +
#> + tb_y = (b-g*q*b)/y;
#> + c_y = c/y;
#> + i_y = (g*k-(1-delta)*k)/y;
#> + end;
#> +
#> + shocks;
#> + var eps_g; stderr 1/sigma_g/100; // use a 1 percent shock
#> + var eps_z; stderr 1/sigma_z/100; // use a 1 percent shock
#> + end;
#> +
#> + steady;
#> +
#> + check;
#> +
#> + // Plot impulse response functions (Figure 4)
#> + stoch_simul(order=1) tb_y c_y i_y;'
#>
#> > run_dynare(code=DynareCodes,model="agtrend")
#>
#> > # You can create an absolute or relative path for the DynareR files.
#> > # The following writes and run mod file in "DynareR/run_dynare/"  folder
#> > # relative to the current path.
#> >
#> >
#> > run_dynare(code=DynareCodes,model="DynareR/run_dynare/agtrend")
#>
#> > demo(bkk)
#>
```

```
#>
#>  demo(bkk)
#>  ---- ~~~
#>
#> > # We use "bkk" of the Dynare example files to illustrate
#> > #how to use this function
#> >
#> >
#> > library(DynareR)
#>
#> > DynareCodes='/*
#> +  * This file implements the multi-country RBC model with time to build,
#> +  * described in Backus, Kehoe and Kydland (1992): "International Real Business
#> +  * Cycles", Journal of Political Economy, 100(4), 745-775.
#> +  *
#> +  * The notation for the variable names are the same in this file than in the paper.
#> +  * However the timing convention is different: we had to taken into account the
#> +  * fact that in Dynare, if a variable is denoted at the current period, then
#> +  * this variable must be also decided at the current period.
#> +  * Concretely, here are the differences between the paper and the model file:
#> +  * - z_t in the model file is equal to z_{t+1} in the paper
#> +  * - k_t in the model file is equal to k_{t+J} in the paper
#> +  * - s_t in the model file is equal to s_{J,t}=s_{J-1,t+1}=...=s_{1,t+J-1} in the paper
#> +  *
#> +  * The macroprocessor is used in this file to create a loop over countries.
#> +  * Only two countries are used here (as in the paper), but it is easy to add
#> +  * new countries in the corresponding macro-variable and completing the
#> +  * calibration.
#> +  *
#> +  * The calibration is the same than in the paper. The results in terms of
#> +  * moments of variables are very close to that of the paper (but not equal
#> +  * since the authors a different solution method).
#> +  *
#> +  * This implementation was written by Sebastien Villemot. Please note that the
#> +  * following copyright notice only applies to this Dynare implementation of the
#> +  * model.
#> +  */
#> +
#> + /*
#> +  * Copyright (C) 2010 Dynare Team
#> +  *
#> +  * This file is part of Dynare.
```

```
#> +  *
#> +  * Dynare is free software: you can redistribute it and/or modify
#> +  * it under the terms of the GNU General Public License as published by
#> +  * the Free Software Foundation, either version 3 of the License, or
#> +  * (at your option) any later version.
#> +  *
#> +  * Dynare is distributed in the hope that it will be useful,
#> +  * but WITHOUT ANY WARRANTY; without even the implied warranty of
#> +  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#> +  * GNU General Public License for more details.
#> +  *
#> +  * You should have received a copy of the GNU General Public License
#> +  * along with Dynare.  If not, see <http://www.gnu.org/licenses/>.
#> +  */
#> +
#> + @#define countries = [ "H", "F" ]
#> + @#define J = 4
#> +
#> + @#for co in countries
#> + var C_@{co} L_@{co} N_@{co} A_@{co} K_@{co} Z_@{co} X_@{co} LAMBDA_@{co} S_@{co} NX_@{c
#> +
#> + varexo E_@{co};
#> +
#> + parameters beta_@{co} alpha_@{co} eta_@{co} mu_@{co} gamma_@{co} theta_@{co} nu_@{co} s
#> + @#endfor
#> +
#> + // Lagrange multiplier of aggregate constraint
#> + var LGM;
#> +
#> + parameters rho_@{countries[1]}_@{countries[2]} rho_@{countries[2]}_@{countries[1]};
#> +
#> + model;
#> + @#for co in countries
#> +
#> + Y_@{co} = ((LAMBDA_@{co}*K_@{co}(-@{J})^theta_@{co}*N_@{co}^(1-theta_@{co}))^(-nu_@{co}
#> + K_@{co} = (1-delta_@{co})*K_@{co}(-1) + S_@{co};
#> + X_@{co} =
#> + @# for lag in (-J+1):0
#> +          + phi_@{co}*S_@{co}(@{lag})
#> + @# endfor
#> + ;
#> +
```

```
#> + A_@{co} = (1-eta_@{co})*A_@{co}(-1) + N_@{co};
#> + L_@{co} = 1 - alpha_@{co}*N_@{co} - (1-alpha_@{co})*eta_@{co}*A_@{co}(-1);
#> +
#> + // Utility multiplied by gamma
#> + # U_@{co} = (C_@{co}^mu_@{co}*L_@{co}^(1-mu_@{co}))^gamma_@{co};
#> +
#> + // FOC with respect to consumption
#> + psi_@{co}*mu_@{co}/C_@{co}*U_@{co} = LGM;
#> +
#> + // FOC with respect to labor
#> + // NOTE: this condition is only valid for alpha = 1
#> + psi_@{co}*(1-mu_@{co})/L_@{co}*U_@{co}*(-alpha_@{co}) = - LGM * (1-theta_@{co})/N_@{co}*
#> +
#> + // FOC with respect to capital
#> + @# for lag in 0:(J-1)
#> +  +beta_@{co}^@{lag}*LGM(+@{lag})*phi_@{co}
#> + @# endfor
#> + @# for lag in 1:J
#> +  -beta_@{co}^@{lag}*LGM(+@{lag})*phi_@{co}*(1-delta_@{co})
#> + @# endfor
#> +  = beta_@{co}^@{J}*LGM(+@{J})*theta_@{co}/K_@{co}*(LAMBDA_@{co}(+@{J})*K_@{co}^theta_@{c
#> +
#> + // FOC with respect to stock of inventories
#> +  LGM=beta_@{co}*LGM(+1)*(1+sigma_@{co}*Z_@{co}^(-nu_@{co}-1)*Y_@{co}(+1)^(1+nu_@{co}));
#> +
#> + // Shock process
#> + @# if co == countries[1]
#> + @#  define alt_co = countries[2]
#> + @# else
#> + @#  define alt_co = countries[1]
#> + @# endif
#> +  (LAMBDA_@{co}-1) = rho_@{co}_@{co}*(LAMBDA_@{co}(-1)-1) + rho_@{co}_@{alt_co}*(LAMBDA_@
#> +
#> +
#> + NX_@{co} = (Y_@{co} - (C_@{co} + X_@{co} + Z_@{co} - Z_@{co}(-1)))/Y_@{co};
#> +
#> + @#endfor
#> +
#> + // World ressource constraint
#> + @#for co in countries
#> +   +C_@{co} + X_@{co} + Z_@{co} - Z_@{co}(-1)
#> + @#endfor
```

```
#> +     =
#> + @#for co in countries
#> +   +Y_@{co}
#> + @#endfor
#> +     ;
#> +
#> + end;
#> +
#> + @#for co in countries
#> + beta_@{co} = 0.99;
#> + mu_@{co} = 0.34;
#> + gamma_@{co} = -1.0;
#> + alpha_@{co} = 1;
#> + eta_@{co} = 0.5; // Irrelevant when alpha=1
#> + theta_@{co} = 0.36;
#> + nu_@{co} = 3;
#> + sigma_@{co} = 0.01;
#> + delta_@{co} = 0.025;
#> + phi_@{co} = 1/@{J};
#> + psi_@{co} = 0.5;
#> + @#endfor
#> +
#> + rho_H_H = 0.906;
#> + rho_F_F = 0.906;
#> + rho_H_F = 0.088;
#> + rho_F_H = 0.088;
#> +
#> + initval;
#> + @#for co in countries
#> + LAMBDA_@{co} = 1;
#> + NX_@{co} = 0;
#> + Z_@{co} = 1;
#> + A_@{co} = 1;
#> + L_@{co} = 0.5;
#> + N_@{co} = 0.5;
#> + Y_@{co} = 1;
#> + K_@{co} = 1;
#> + C_@{co} = 1;
#> + S_@{co} = 1;
#> + X_@{co} = 1;
#> +
#> + E_@{co} = 0;
```

```
#> + @#endfor
#> +
#> + LGM = 1;
#> + end;
#> +
#> + shocks;
#> + var E_H; stderr 0.00852;
#> + var E_F; stderr 0.00852;
#> + corr E_H, E_F = 0.258;
#> + end;
#> +
#> + steady;
#> + check;
#> +
#> + stoch_simul(order=1, hp_filter=1600,graph_format=pdf);
#> + '
#>
#> > run_dynare(code=DynareCodes,model="bkk")
#>
#> > # You can create an absolute or relative path for the DynareR files.
#> > # The following writes and run mod file in "DynareR/run_dynare/"  folder
#> > # relative to the current path.
#> >
#> >
#> > run_dynare(code=DynareCodes,model="DynareR/run_dynare/bkk")
#>
#> > demo(example1)
#>
#>
#>  demo(example1)
#>  ---- ~~~~~~~~
#>
#> > # We use "example1" of the Dynare example files to illustrate
#> > #how to use this function
#> >
#> > FileName<-"example1"
#>
#> > library(DynareR)
#>
#> > DynareCodes='/*
#> +  * Example 1 from F. Collard (2001): "Stochastic simulations with DYNARE:
#> +  * A practical guide" (see "guide.pdf" in the documentation directory).
```

```
#> +  */
#> +
#> + /*
#> +  * Copyright (C) 2001-2010 Dynare Team
#> +  *
#> +  * This file is part of Dynare.
#> +  *
#> +  * Dynare is free software: you can redistribute it and/or modify
#> +  * it under the terms of the GNU General Public License as published by
#> +  * the Free Software Foundation, either version 3 of the License, or
#> +  * (at your option) any later version.
#> +  *
#> +  * Dynare is distributed in the hope that it will be useful,
#> +  * but WITHOUT ANY WARRANTY; without even the implied warranty of
#> +  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#> +  * GNU General Public License for more details.
#> +  *
#> +  * You should have received a copy of the GNU General Public License
#> +  * along with Dynare.  If not, see <http://www.gnu.org/licenses/>.
#> +  */
#> +
#> +
#> + var y, c, k, a, h, b;
#> + varexo e, u;
#> +
#> + parameters beta, rho, alpha, delta, theta, psi, tau;
#> +
#> + alpha = 0.36;
#> + rho   = 0.95;
#> + tau   = 0.025;
#> + beta  = 0.99;
#> + delta = 0.025;
#> + psi   = 0;
#> + theta = 2.95;
#> +
#> + phi   = 0.1;
#> +
#> + model;
#> + c*theta*h^(1+psi)=(1-alpha)*y;
#> + k = beta*(((exp(b)*c)/(exp(b(+1))*c(+1)))
#> +     *(exp(b(+1))*alpha*y(+1)+(1-delta)*k));
#> + y = exp(a)*(k(-1)^alpha)*(h^(1-alpha));
```

```
#> + k = exp(b)*(y-c)+(1-delta)*k(-1);
#> + a = rho*a(-1)+tau*b(-1) + e;
#> + b = tau*a(-1)+rho*b(-1) + u;
#> + end;
#> +
#> + initval;
#> + y = 1.08068253095672;
#> + c = 0.80359242014163;
#> + h = 0.29175631001732;
#> + k = 11.08360443260358;
#> + a = 0;
#> + b = 0;
#> + e = 0;
#> + u = 0;
#> + end;
#> +
#> + shocks;
#> + var e; stderr 0.009;
#> + var u; stderr 0.009;
#> + var e, u = phi*0.009*0.009;
#> + end;
#> +
#> + stoch_simul;'
#>
#> > run_dynare(code=DynareCodes,model = "example1")
#>
#> > # You can create an absolute or relative path for the DynareR files.
#> > # The following writes and run mod file in "DynareR/run_dynare/"  folder
#> > # relative to the current path.
#> >
#> >
#> >
#> > run_dynare(code=DynareCodes,model = "DynareR/run_dynare/example1")
#>
#> > lapply(c("agtrend","bkk","example1"),\(x) file.copy(paste0("DynareR/run_dynare/",x,"/",
#> [[1]]
#> [1] TRUE
#>
#> [[2]]
#> [1] TRUE
#>
#> [[3]]
```

```
#> [1] TRUE
#>
#>
#> > run_models(c("agtrend","bkk","example1")) # This should be executed after running the de
#>
#> > run_models() # Run all models in Current Working Directory.
#>
#> > # Copy the dynare files to the 'DynareR/run_dynare' directory
#> >
#> > lapply(c("agtrend","bkk","example1"),\(x) file.copy(paste0(x,".mod"),"DynareR/run_dynare
#> [[1]]
#> [1] TRUE
#>
#> [[2]]
#> [1] TRUE
#>
#> [[3]]
#> [1] TRUE
#>
#>
#> > run_models("DynareR/run_dynare*") # Run all models in 'DynareR/run_dynare' folder
demo(import_log)
#>
#>
#>  demo(import_log)
#>  ---- ~~~~~~~~~~
#>
#> > library(DynareR)
#>
#> > demo(bkk)
#>
#>
#>  demo(bkk)
#>  ---- ~~~
#>
#> > # We use "bkk" of the Dynare example files to illustrate
#> > #how to use this function
#> >
#> >
#> > library(DynareR)
#>
#> > DynareCodes='/*
```

```
#> +  * This file implements the multi-country RBC model with time to build,
#> +  * described in Backus, Kehoe and Kydland (1992): "International Real Business
#> +  * Cycles", Journal of Political Economy, 100(4), 745-775.
#> +  *
#> +  * The notation for the variable names are the same in this file than in the paper.
#> +  * However the timing convention is different: we had to taken into account the
#> +  * fact that in Dynare, if a variable is denoted at the current period, then
#> +  * this variable must be also decided at the current period.
#> +  * Concretely, here are the differences between the paper and the model file:
#> +  * - z_t in the model file is equal to z_{t+1} in the paper
#> +  * - k_t in the model file is equal to k_{t+J} in the paper
#> +  * - s_t in the model file is equal to s_{J,t}=s_{J-1,t+1}=...=s_{1,t+J-1} in the paper
#> +  *
#> +  * The macroprocessor is used in this file to create a loop over countries.
#> +  * Only two countries are used here (as in the paper), but it is easy to add
#> +  * new countries in the corresponding macro-variable and completing the
#> +  * calibration.
#> +  *
#> +  * The calibration is the same than in the paper. The results in terms of
#> +  * moments of variables are very close to that of the paper (but not equal
#> +  * since the authors a different solution method).
#> +  *
#> +  * This implementation was written by Sebastien Villemot. Please note that the
#> +  * following copyright notice only applies to this Dynare implementation of the
#> +  * model.
#> +  */
#> +
#> + /*
#> +  * Copyright (C) 2010 Dynare Team
#> +  *
#> +  * This file is part of Dynare.
#> +  *
#> +  * Dynare is free software: you can redistribute it and/or modify
#> +  * it under the terms of the GNU General Public License as published by
#> +  * the Free Software Foundation, either version 3 of the License, or
#> +  * (at your option) any later version.
#> +  *
#> +  * Dynare is distributed in the hope that it will be useful,
#> +  * but WITHOUT ANY WARRANTY; without even the implied warranty of
#> +  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#> +  * GNU General Public License for more details.
#> +  *
```

```
#> +  * You should have received a copy of the GNU General Public License
#> +  * along with Dynare.  If not, see <http://www.gnu.org/licenses/>.
#> +  */
#> +
#> + @#define countries = [ "H", "F" ]
#> + @#define J = 4
#> +
#> + @#for co in countries
#> + var C_@{co} L_@{co} N_@{co} A_@{co} K_@{co} Z_@{co} X_@{co} LAMBDA_@{co} S_@{co} NX_@{c
#> +
#> + varexo E_@{co};
#> +
#> + parameters beta_@{co} alpha_@{co} eta_@{co} mu_@{co} gamma_@{co} theta_@{co} nu_@{co} s
#> + @#endfor
#> +
#> + // Lagrange multiplier of aggregate constraint
#> + var LGM;
#> +
#> + parameters rho_@{countries[1]}_@{countries[2]} rho_@{countries[2]}_@{countries[1]};
#> +
#> + model;
#> + @#for co in countries
#> +
#> + Y_@{co} = ((LAMBDA_@{co}*K_@{co}(-@{J})^theta_@{co}*N_@{co}^(1-theta_@{co}))^(-nu_@{co}
#> + K_@{co} = (1-delta_@{co})*K_@{co}(-1) + S_@{co};
#> + X_@{co} =
#> + @# for lag in (-J+1):0
#> +           + phi_@{co}*S_@{co}(@{lag})
#> + @# endfor
#> + ;
#> +
#> + A_@{co} = (1-eta_@{co})*A_@{co}(-1) + N_@{co};
#> + L_@{co} = 1 - alpha_@{co}*N_@{co} - (1-alpha_@{co})*eta_@{co}*A_@{co}(-1);
#> +
#> + // Utility multiplied by gamma
#> + # U_@{co} = (C_@{co}^mu_@{co}*L_@{co}^(1-mu_@{co}))^gamma_@{co};
#> +
#> + // FOC with respect to consumption
#> + psi_@{co}*mu_@{co}/C_@{co}*U_@{co} = LGM;
#> +
#> + // FOC with respect to labor
#> + // NOTE: this condition is only valid for alpha = 1
```

```
#> + psi_@{co}*(1-mu_@{co})/L_@{co}*U_@{co}*(-alpha_@{co}) = - LGM * (1-theta_@{co})/N_@{co}*
#> +
#> + // FOC with respect to capital
#> + @# for lag in 0:(J-1)
#> +  +beta_@{co}^@{lag}*LGM(+@{lag})*phi_@{co}
#> + @# endfor
#> + @# for lag in 1:J
#> +  -beta_@{co}^@{lag}*LGM(+@{lag})*phi_@{co}*(1-delta_@{co})
#> + @# endfor
#> +  = beta_@{co}^@{J}*LGM(+@{J})*theta_@{co}/K_@{co}*(LAMBDA_@{co}(+@{J})*K_@{co}^theta_@{c
#> +
#> + // FOC with respect to stock of inventories
#> +  LGM=beta_@{co}*LGM(+1)*(1+sigma_@{co}*Z_@{co}^(-nu_@{co}-1)*Y_@{co}(+1)^(1+nu_@{co}));
#> +
#> + // Shock process
#> + @# if co == countries[1]
#> + @#  define alt_co = countries[2]
#> + @# else
#> + @#  define alt_co = countries[1]
#> + @# endif
#> +  (LAMBDA_@{co}-1) = rho_@{co}_@{co}*(LAMBDA_@{co}(-1)-1) + rho_@{co}_@{alt_co}*(LAMBDA_@
#> +
#> +
#> + NX_@{co} = (Y_@{co} - (C_@{co} + X_@{co} + Z_@{co} - Z_@{co}(-1)))/Y_@{co};
#> +
#> + @#endfor
#> +
#> + // World ressource constraint
#> + @#for co in countries
#> +   +C_@{co} + X_@{co} + Z_@{co} - Z_@{co}(-1)
#> + @#endfor
#> +     =
#> + @#for co in countries
#> +   +Y_@{co}
#> + @#endfor
#> +     ;
#> +
#> + end;
#> +
#> + @#for co in countries
#> + beta_@{co} = 0.99;
#> + mu_@{co} = 0.34;
```

```
#> + gamma_@{co} = -1.0;
#> + alpha_@{co} = 1;
#> + eta_@{co} = 0.5; // Irrelevant when alpha=1
#> + theta_@{co} = 0.36;
#> + nu_@{co} = 3;
#> + sigma_@{co} = 0.01;
#> + delta_@{co} = 0.025;
#> + phi_@{co} = 1/@{J};
#> + psi_@{co} = 0.5;
#> + @#endfor
#> +
#> + rho_H_H = 0.906;
#> + rho_F_F = 0.906;
#> + rho_H_F = 0.088;
#> + rho_F_H = 0.088;
#> +
#> + initval;
#> + @#for co in countries
#> + LAMBDA_@{co} = 1;
#> + NX_@{co} = 0;
#> + Z_@{co} = 1;
#> + A_@{co} = 1;
#> + L_@{co} = 0.5;
#> + N_@{co} = 0.5;
#> + Y_@{co} = 1;
#> + K_@{co} = 1;
#> + C_@{co} = 1;
#> + S_@{co} = 1;
#> + X_@{co} = 1;
#> +
#> + E_@{co} = 0;
#> + @#endfor
#> +
#> + LGM = 1;
#> + end;
#> +
#> + shocks;
#> + var E_H; stderr 0.00852;
#> + var E_F; stderr 0.00852;
#> + corr E_H, E_F = 0.258;
#> + end;
#> +
```

```
#> + steady;
#> + check;
#> +
#> + stoch_simul(order=1, hp_filter=1600,graph_format=pdf);
#> + '
#>
#> > run_dynare(code=DynareCodes,model="bkk")
#>
#> > # You can create an absolute or relative path for the DynareR files.
#> > # The following writes and run mod file in "DynareR/run_dynare/"  folder
#> > # relative to the current path.
#> >
#> >
#> > run_dynare(code=DynareCodes,model="DynareR/run_dynare/bkk")
#>
#> > import_log(model="bkk")
#> NULL
#>
#> > # Alternatively, use the path to the log file
#> >
#> > import_log(path="bkk/bkk.log")
#> NULL
#>
#> > # Access the mported list
#> >
#> > dynare$bkk
#> $steady
#>         C_H X0.826091
#> 1       L_H  0.696782
#> 2       N_H  0.303218
#> 3       A_H  0.606436
#> 4       K_H 11.014800
#> 5       Z_H  1.098700
#> 6       X_H  0.275370
#> 7  LAMBDA_H  1.000000
#> 8       S_H  0.275370
#> 9      NX_H  0.000000
#> 10      Y_H  1.101460
#> 11      C_F  0.826091
#> 12      L_F  0.696782
#> 13      N_F  0.303218
#> 14      A_F  0.606436
```

```
#> 15      K_F 11.014800
#> 16      Z_F  1.098700
#> 17      X_F  0.275370
#> 18 LAMBDA_F  1.000000
#> 19      S_F  0.275370
#> 20     NX_F  0.000000
#> 21      Y_F  1.101460
#> 22      LGM  0.278732
#>
#> $eigenvalues
#>          Modulus       Real       Imaginary
#> 1      3.709e-15 -3.709e-15               0
#> 2      1.157e-14 -1.157e-14               0
#> 3      3.667e-06 -3.667e-06               0
#> 4      3.667e-06  1.834e-06       3.176e-06
#> 5      3.667e-06  1.834e-06      -3.176e-06
#> 6      4.913e-06  4.913e-06               0
#> 7      4.913e-06 -2.456e-06       4.255e-06
#> 8      4.913e-06 -2.456e-06      -4.255e-06
#> 9      0.0006354 -0.0006354               0
#> 10     0.0006357 -2.917e-07       0.0006357
#> 11     0.0006357 -2.917e-07      -0.0006357
#> 12      0.000636   0.000636               0
#> 13           0.5        0.5               0
#> 14           0.5        0.5               0
#> 15        0.7265    -0.7265               0
#> 16        0.7929    0.05082          0.7913
#> 17        0.7929    0.05082         -0.7913
#> 18         0.818      0.818               0
#> 19         0.969      0.969               0
#> 20         0.994      0.994               0
#> 21         1.042      1.042               0
#> 22         1.274    0.08164           1.271
#> 23         1.274    0.08164          -1.271
#> 24          1.39      -1.39               0
#> 25          3673       2597            2597
#> 26          3673       2597           -2597
#> 27          3673      -2597            2597
#> 28          3673      -2597           -2597
#> 29     1.942e+04 -1.942e+04               0
#> 30     1.942e+04     0.9036       1.942e+04
#> 31     1.942e+04     0.9036      -1.942e+04
```

```
#> 32     1.942e+04  1.942e+04               0
#> 33     3.003e+33  3.003e+33               0
#> 34     7.789e+33 -7.789e+33               0
#> 35           Inf        Inf               0
#> 36           Inf        Inf               0
#> 37         There        are              16
#> 38 eigenvalue(s)     larger            than
#> 39             1         in         modulus
#> 40           for         16 forward-looking
#> 41   variable(s)
#> 42           The       rank       condition
#> 43            is  verified.
#>
#> $summary
#>            Number of variables 42
#> 1   Number of stochastic shocks  2
#> 2     Number of state variables 20
#> 3             Number of jumpers 16
#> 4    Number of static variables  8
#>
#> $shocks
#>   Variables     E_H     E_F
#> 1       E_H 7.3e-05 1.9e-05
#> 2       E_F 1.9e-05 7.3e-05
#>
#> $policy
#>               X       C_H       L_H       N_H       A_H       K_H       Z_H
#> 1      Constant  0.826091  0.696782  0.303218  0.606436 11.014795  1.098697
#> 2       A_H(-1)  0.000000  0.000000  0.000000  0.500000  0.000000  0.000000
#> 3       K_H(-1)  0.013152  0.005880 -0.005880 -0.005880  0.349915  0.007711
#> 4       Z_H(-1)  0.023611 -0.002992  0.002992  0.002992  1.079008  0.209348
#> 5       S_H(-1) -0.010263 -0.004589  0.004589  0.004589 -0.387216 -0.007373
#> 6       A_F(-1)  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
#> 7       K_F(-1)  0.013152  0.005880 -0.005880 -0.005880  0.349915  0.007711
#> 8       Z_F(-1)  0.017337  0.007751 -0.007751 -0.007751  1.079008  0.209348
#> 9       S_F(-1) -0.010263 -0.004589  0.004589  0.004589 -0.387216 -0.007373
#> 10      K_H(-2)  0.000490  0.000219 -0.000219 -0.000219 -0.009893  0.001478
#> 11      K_H(-3)  0.000463  0.000207 -0.000207 -0.000207 -0.018435  0.033007
#> 12      K_H(-4)  0.006231 -0.008799  0.008799  0.008799  0.053936  0.010465
#> 13      S_H(-2) -0.007461 -0.003336  0.003336  0.003336 -0.358712 -0.020184
#> 14      S_H(-3) -0.004214 -0.001884  0.001884  0.001884 -0.262264 -0.050884
#> 15      K_F(-2)  0.000490  0.000219 -0.000219 -0.000219 -0.009893  0.001478
```

```
#> 16      K_F(-3)  0.000463  0.000207 -0.000207 -0.000207 -0.018435 -0.025667
#> 17      K_F(-4)  0.000867  0.000387 -0.000387 -0.000387  0.053936  0.010465
#> 18      S_F(-2) -0.007461 -0.003336  0.003336  0.003336 -0.358712 -0.020184
#> 19      S_F(-3) -0.004214 -0.001884  0.001884  0.001884 -0.262264 -0.050884
#> 20 LAMBDA_H(-1)  0.356099 -0.161927  0.161927  0.161927  8.865347  0.746282
#> 21 LAMBDA_F(-1)  0.221839  0.067992 -0.067992 -0.067992 -7.142393 -0.454954
#> 22          E_H  0.372779 -0.187789  0.187789  0.187789 10.651359  0.880795
#> 23          E_F  0.208648  0.093286 -0.093286 -0.093286 -8.918005 -0.587709
#>          X_H LAMBDA_H       S_H      NX_H       Y_H       C_F       L_F
#> 1   0.275370    1.000  0.275370  0.000000  1.101461  0.826091  0.696782
#> 2   0.000000    0.000  0.000000  0.000000  0.000000  0.000000  0.000000
#> 3  -0.156271    0.000 -0.625085  0.110649 -0.013533  0.013152  0.005880
#> 4   0.269752    0.000  1.079008  0.466904  0.016987  0.017337  0.007751
#> 5   0.153196    0.000 -0.387216 -0.113486  0.010560 -0.010263 -0.004589
#> 6   0.000000    0.000  0.000000  0.000000  0.000000  0.000000  0.000000
#> 7   0.087479    0.000  0.349915 -0.110649 -0.013533  0.013152  0.005880
#> 8   0.269752    0.000  1.079008 -0.466904 -0.017839  0.023611 -0.002992
#> 9  -0.096804    0.000 -0.387216  0.113486  0.010560 -0.010263 -0.004589
#> 10 -0.002473    0.000 -0.009893  0.000000 -0.000505  0.000490  0.000219
#> 11 -0.004609    0.000 -0.018435 -0.026635 -0.000476  0.000463  0.000207
#> 12  0.013484    0.000  0.053936  0.023339  0.055887  0.000867  0.000387
#> 13  0.160322    0.000 -0.358712 -0.113486  0.007677 -0.007461 -0.003336
#> 14  0.184434    0.000 -0.262264 -0.113486  0.004336 -0.004214 -0.001884
#> 15 -0.002473    0.000 -0.009893  0.000000 -0.000505  0.000490  0.000219
#> 16 -0.004609    0.000 -0.018435  0.026635 -0.000476  0.000463  0.000207
#> 17  0.013484    0.000  0.053936 -0.023339 -0.000892  0.006231 -0.008799
#> 18 -0.089678    0.000 -0.358712  0.113486  0.007677 -0.007461 -0.003336
#> 19 -0.065566    0.000 -0.262264  0.113486  0.004336 -0.004214 -0.001884
#> 20  2.216337    0.906  8.865347 -1.777810  1.360531  0.221839  0.067992
#> 21 -1.785598    0.088 -7.142393  1.777810 -0.060526  0.356099 -0.161927
#> 22  2.662840    1.000 10.651359 -2.173361  1.522542  0.208648  0.093286
#> 23 -2.229501    0.000 -8.918005  2.173361 -0.214691  0.372779 -0.187789
#>          N_F       A_F       K_F       Z_F       X_F LAMBDA_F       S_F
#> 1   0.303218  0.606436 11.014795  1.098697  0.275370    1.000  0.275370
#> 2   0.000000  0.000000  0.000000  0.000000  0.000000    0.000  0.000000
#> 3  -0.005880 -0.005880  0.349915  0.007711  0.087479    0.000  0.349915
#> 4  -0.007751 -0.007751  1.079008  0.209348  0.269752    0.000  1.079008
#> 5   0.004589  0.004589 -0.387216 -0.007373 -0.096804    0.000 -0.387216
#> 6   0.000000  0.500000  0.000000  0.000000  0.000000    0.000  0.000000
#> 7  -0.005880 -0.005880  0.349915  0.007711 -0.156271    0.000 -0.625085
#> 8   0.002992  0.002992  1.079008  0.209348  0.269752    0.000  1.079008
#> 9   0.004589  0.004589 -0.387216 -0.007373  0.153196    0.000 -0.387216
```

```
#> 10 -0.000219 -0.000219 -0.009893  0.001478 -0.002473    0.000 -0.009893
#> 11 -0.000207 -0.000207 -0.018435 -0.025667 -0.004609    0.000 -0.018435
#> 12 -0.000387 -0.000387  0.053936  0.010465  0.013484    0.000  0.053936
#> 13  0.003336  0.003336 -0.358712 -0.020184 -0.089678    0.000 -0.358712
#> 14  0.001884  0.001884 -0.262264 -0.050884 -0.065566    0.000 -0.262264
#> 15 -0.000219 -0.000219 -0.009893  0.001478 -0.002473    0.000 -0.009893
#> 16 -0.000207 -0.000207 -0.018435  0.033007 -0.004609    0.000 -0.018435
#> 17  0.008799  0.008799  0.053936  0.010465  0.013484    0.000  0.053936
#> 18  0.003336  0.003336 -0.358712 -0.020184  0.160322    0.000 -0.358712
#> 19  0.001884  0.001884 -0.262264 -0.050884  0.184434    0.000 -0.262264
#> 20 -0.067992 -0.067992 -7.142393 -0.454954 -1.785598    0.088 -7.142393
#> 21  0.161927  0.161927  8.865347  0.746282  2.216337    0.906  8.865347
#> 22 -0.093286 -0.093286 -8.918005 -0.587709 -2.229501    0.000 -8.918005
#> 23  0.187789  0.187789 10.651359  0.880795  2.662840    1.000 10.651359
#>         NX_F       Y_F       LGM
#> 1   0.000000  1.101461  0.278732
#> 2   0.000000  0.000000  0.000000
#> 3  -0.110649 -0.013533 -0.007499
#> 4  -0.466904 -0.017839 -0.009885
#> 5   0.113486  0.010560  0.005852
#> 6   0.000000  0.000000  0.000000
#> 7   0.110649 -0.013533 -0.007499
#> 8   0.466904  0.016987 -0.009885
#> 9  -0.113486  0.010560  0.005852
#> 10  0.000000 -0.000505 -0.000280
#> 11  0.026635 -0.000476 -0.000264
#> 12 -0.023339 -0.000892 -0.000494
#> 13  0.113486  0.007677  0.004254
#> 14  0.113486  0.004336  0.002403
#> 15  0.000000 -0.000505 -0.000280
#> 16 -0.026635 -0.000476 -0.000264
#> 17  0.023339  0.055887 -0.000494
#> 18 -0.113486  0.007677  0.004254
#> 19 -0.113486  0.004336  0.002403
#> 20  1.777810 -0.060526 -0.118252
#> 21 -1.777810  1.360531 -0.118252
#> 22  2.173361 -0.214691 -0.118966
#> 23 -2.173361  1.522542 -0.118966
#>
#> $moments
#>    VARIABLE    MEAN STD. DEV. VARIANCE
#> 1       C_H  0.8261    0.0053   0.0000
```

```
#> 2        L_H  0.6968    0.0024   0.0000
#> 3        N_H  0.3032    0.0024   0.0000
#> 4        A_H  0.6064    0.0039   0.0000
#> 5        K_H 11.0148    0.1261   0.0159
#> 6        Z_H  1.0987    0.0131   0.0002
#> 7        X_H  0.2754    0.0437   0.0019
#> 8  LAMBDA_H  1.0000    0.0108   0.0001
#> 9        S_H  0.2754    0.1047   0.0110
#> 10      NX_H  0.0000    0.0440   0.0019
#> 11       Y_H  1.1015    0.0174   0.0003
#> 12       C_F  0.8261    0.0053   0.0000
#> 13       L_F  0.6968    0.0024   0.0000
#> 14       N_F  0.3032    0.0024   0.0000
#> 15       A_F  0.6064    0.0039   0.0000
#> 16       K_F 11.0148    0.1261   0.0159
#> 17       Z_F  1.0987    0.0131   0.0002
#> 18       X_F  0.2754    0.0437   0.0019
#> 19 LAMBDA_F  1.0000    0.0108   0.0001
#> 20       S_F  0.2754    0.1047   0.0110
#> 21      NX_F  0.0000    0.0440   0.0019
#> 22       Y_F  1.1015    0.0174   0.0003
#> 23       LGM  0.2787    0.0021   0.0000
#>
#> $decomposition
#>            X   E_H   E_F
#> 1        C_H 80.86 19.14
#> 2        L_H 67.89 32.11
#> 3        N_H 67.89 32.11
#> 4        A_H 67.87 32.13
#> 5        K_H 40.83 59.17
#> 6        Z_H 49.75 50.25
#> 7        X_H 42.38 57.62
#> 8  LAMBDA_H 93.94  6.06
#> 9        S_H 45.49 54.51
#> 10      NX_H 37.10 62.90
#> 11       Y_H 89.35 10.65
#> 12       C_F 41.78 58.22
#> 13       L_F 12.14 87.86
#> 14       N_F 12.14 87.86
#> 15       A_F 11.94 88.06
#> 16       K_F 34.93 65.07
#> 17       Z_F 27.41 72.59
```

```
#> 18       X_F 32.56 67.44
#> 19 LAMBDA_F 14.07 85.93
#> 20       S_F 29.30 70.70
#> 21      NX_F 37.10 62.90
#> 22       Y_F  5.81 94.19
#> 23       LGM 62.90 37.10
#>
#> $correlations
#>    Variables     C_H     L_H     N_H     A_H     K_H     Z_H     X_H LAMBDA_H
#> 1        C_H  1.0000 -0.5460  0.5460  0.5203  0.3194  0.4186  0.1688   0.8965
#> 2        L_H -0.5460  1.0000 -1.0000 -0.9213 -0.7998 -0.8780 -0.2580  -0.7287
#> 3        N_H  0.5460 -1.0000  1.0000  0.9213  0.7998  0.8780  0.2580   0.7287
#> 4        A_H  0.5203 -0.9213  0.9213  1.0000  0.6512  0.8448  0.0896   0.6216
#> 5        K_H  0.3194 -0.7998  0.7998  0.6512  1.0000  0.8243  0.7304   0.6599
#> 6        Z_H  0.4186 -0.8780  0.8780  0.8448  0.8243  1.0000  0.3762   0.6273
#> 7        X_H  0.1688 -0.2580  0.2580  0.0896  0.7304  0.3762  1.0000   0.5215
#> 8   LAMBDA_H  0.8965 -0.7287  0.7287  0.6216  0.6599  0.6273  0.5215   1.0000
#> 9        S_H  0.1259 -0.2550  0.2550 -0.0279  0.4447  0.1974  0.4276   0.3194
#> 10      NX_H  0.0031 -0.0116  0.0116  0.1838 -0.5324 -0.1758 -0.9503  -0.3174
#> 11       Y_H  0.8043 -0.9368  0.9368  0.8659  0.7035  0.7961  0.2598   0.8936
#> 12       C_F  0.8775 -0.0799  0.0799  0.0895 -0.0858 -0.0103  0.0527   0.6549
#> 13       L_F -0.0799 -0.7742  0.7742  0.7185  0.7424  0.7547  0.1841   0.1907
#> 14       N_F  0.0799  0.7742 -0.7742 -0.7185 -0.7424 -0.7547 -0.1841  -0.1907
#> 15       A_F  0.0895  0.7185 -0.7185 -0.7826 -0.5816 -0.7161 -0.0127  -0.1134
#> 16       K_F -0.0858  0.7424 -0.7424 -0.5816 -0.9397 -0.7500 -0.7070  -0.4965
#> 17       Z_F -0.0103  0.7547 -0.7547 -0.7161 -0.7500 -0.8852 -0.3321  -0.3112
#> 18       X_F  0.0527  0.1841 -0.1841 -0.0127 -0.7070 -0.3321 -0.9710  -0.3354
#> 19  LAMBDA_F  0.6549  0.1907 -0.1907 -0.1134 -0.4965 -0.3112 -0.3354   0.3106
#> 20       S_F  0.0056  0.2027 -0.2027  0.0608 -0.4335 -0.1742 -0.4165  -0.2041
#> 21      NX_F -0.0031  0.0116 -0.0116 -0.1838  0.5324  0.1758  0.9503   0.3174
#> 22       Y_F  0.4230  0.5145 -0.5145 -0.4679 -0.5665 -0.5392 -0.1151   0.1365
#> 23       LGM -0.9685  0.3203 -0.3203 -0.3146 -0.1236 -0.2125 -0.1143  -0.7972
#>          S_H    NX_H     Y_H     C_F     L_F     N_F     A_F     K_F     Z_F
#> 1    0.1259  0.0031  0.8043  0.8775 -0.0799  0.0799  0.0895 -0.0858 -0.0103
#> 2   -0.2550 -0.0116 -0.9368 -0.0799 -0.7742  0.7742  0.7185  0.7424  0.7547
#> 3    0.2550  0.0116  0.9368  0.0799  0.7742 -0.7742 -0.7185 -0.7424 -0.7547
#> 4   -0.0279  0.1838  0.8659  0.0895  0.7185 -0.7185 -0.7826 -0.5816 -0.7161
#> 5    0.4447 -0.5324  0.7035 -0.0858  0.7424 -0.7424 -0.5816 -0.9397 -0.7500
#> 6    0.1974 -0.1758  0.7961 -0.0103  0.7547 -0.7547 -0.7161 -0.7500 -0.8852
#> 7    0.4276 -0.9503  0.2598  0.0527  0.1841 -0.1841 -0.0127 -0.7070 -0.3321
#> 8    0.3194 -0.3174  0.8936  0.6549  0.1907 -0.1907 -0.1134 -0.4965 -0.3112
#> 9    1.0000 -0.4680  0.2471  0.0056  0.2027 -0.2027  0.0608 -0.4335 -0.1742
```

```
#> 10 -0.4680  1.0000  0.0014 -0.0031  0.0116 -0.0116 -0.1838  0.5324  0.1758
#> 11  0.2471  0.0014  1.0000  0.4230  0.5145 -0.5145 -0.4679 -0.5665 -0.5392
#> 12  0.0056 -0.0031  0.4230  1.0000 -0.5460  0.5460  0.5203  0.3194  0.4186
#> 13  0.2027  0.0116  0.5145 -0.5460  1.0000 -1.0000 -0.9213 -0.7998 -0.8780
#> 14 -0.2027 -0.0116 -0.5145  0.5460 -1.0000  1.0000  0.9213  0.7998  0.8780
#> 15  0.0608 -0.1838 -0.4679  0.5203 -0.9213  0.9213  1.0000  0.6512  0.8448
#> 16 -0.4335  0.5324 -0.5665  0.3194 -0.7998  0.7998  0.6512  1.0000  0.8243
#> 17 -0.1742  0.1758 -0.5392  0.4186 -0.8780  0.8780  0.8448  0.8243  1.0000
#> 18 -0.4165  0.9503 -0.1151  0.1688 -0.2580  0.2580  0.0896  0.7304  0.3762
#> 19 -0.2041  0.3174  0.1365  0.8965 -0.7287  0.7287  0.6216  0.6599  0.6273
#> 20 -0.9773  0.4680 -0.1542  0.1259 -0.2550  0.2550 -0.0279  0.4447  0.1974
#> 21  0.4680 -1.0000 -0.0014  0.0031 -0.0116  0.0116  0.1838 -0.5324 -0.1758
#> 22 -0.1542 -0.0014 -0.1876  0.8043 -0.9368  0.9368  0.8659  0.7035  0.7961
#> 23 -0.0666  0.0000 -0.6311 -0.9685  0.3203 -0.3203 -0.3146 -0.1236 -0.2125
#>        X_F LAMBDA_F     S_F    NX_F     Y_F     LGM
#> 1   0.0527   0.6549  0.0056 -0.0031  0.4230 -0.9685
#> 2   0.1841   0.1907  0.2027  0.0116  0.5145  0.3203
#> 3  -0.1841  -0.1907 -0.2027 -0.0116 -0.5145 -0.3203
#> 4  -0.0127  -0.1134  0.0608 -0.1838 -0.4679 -0.3146
#> 5  -0.7070  -0.4965 -0.4335  0.5324 -0.5665 -0.1236
#> 6  -0.3321  -0.3112 -0.1742  0.1758 -0.5392 -0.2125
#> 7  -0.9710  -0.3354 -0.4165  0.9503 -0.1151 -0.1143
#> 8  -0.3354   0.3106 -0.2041  0.3174  0.1365 -0.7972
#> 9  -0.4165  -0.2041 -0.9773  0.4680 -0.1542 -0.0666
#> 10  0.9503   0.3174  0.4680 -1.0000 -0.0014  0.0000
#> 11 -0.1151   0.1365 -0.1542 -0.0014 -0.1876 -0.6311
#> 12  0.1688   0.8965  0.1259  0.0031  0.8043 -0.9685
#> 13 -0.2580  -0.7287 -0.2550 -0.0116 -0.9368  0.3203
#> 14  0.2580   0.7287  0.2550  0.0116  0.9368 -0.3203
#> 15  0.0896   0.6216 -0.0279  0.1838  0.8659 -0.3146
#> 16  0.7304   0.6599  0.4447 -0.5324  0.7035 -0.1236
#> 17  0.3762   0.6273  0.1974 -0.1758  0.7961 -0.2125
#> 18  1.0000   0.5215  0.4276 -0.9503  0.2598 -0.1143
#> 19  0.5215   1.0000  0.3194 -0.3174  0.8936 -0.7972
#> 20  0.4276   0.3194  1.0000 -0.4680  0.2471 -0.0666
#> 21 -0.9503  -0.3174 -0.4680  1.0000  0.0014  0.0000
#> 22  0.2598   0.8936  0.2471  0.0014  1.0000 -0.6311
#> 23 -0.1143  -0.7972 -0.0666  0.0000 -0.6311  1.0000
#>
#> $autocorrelation
#>       Order      X1      X2      X3      X4      X5
#> 1       C_H  0.7277  0.4968  0.3105  0.1655  0.0268
```

```
#> 2        L_H  0.6843  0.4628  0.3519  0.3262  0.0676
#> 3        N_H  0.6843  0.4628  0.3519  0.3262  0.0676
#> 4        A_H  0.8817  0.7001  0.5277  0.3600  0.1320
#> 5        K_H  0.6471  0.3737  0.1679  0.0176 -0.0905
#> 6        Z_H  0.7119  0.5339  0.4710  0.1904 -0.0219
#> 7        X_H  0.6202  0.2743 -0.0522 -0.3742 -0.3032
#> 8   LAMBDA_H  0.6933  0.4421  0.2405  0.0824 -0.0377
#> 9        S_H -0.1118 -0.0955 -0.0784 -0.0598 -0.0506
#> 10      NX_H  0.5027  0.2569  0.0621 -0.3409 -0.2835
#> 11       Y_H  0.6868  0.4605  0.3238  0.2646  0.0490
#> 12       C_F  0.7277  0.4968  0.3105  0.1655  0.0268
#> 13       L_F  0.6843  0.4628  0.3519  0.3262  0.0676
#> 14       N_F  0.6843  0.4628  0.3519  0.3262  0.0676
#> 15       A_F  0.8817  0.7001  0.5277  0.3600  0.1320
#> 16       K_F  0.6471  0.3737  0.1679  0.0176 -0.0905
#> 17       Z_F  0.7119  0.5339  0.4710  0.1904 -0.0219
#> 18       X_F  0.6202  0.2743 -0.0522 -0.3742 -0.3032
#> 19  LAMBDA_F  0.6933  0.4421  0.2405  0.0824 -0.0377
#> 20       S_F -0.1118 -0.0955 -0.0784 -0.0598 -0.0506
#> 21      NX_F  0.5027  0.2569  0.0621 -0.3409 -0.2835
#> 22       Y_F  0.6868  0.4605  0.3238  0.2646  0.0490
#> 23       LGM  0.7367  0.5080  0.3158  0.1585  0.0289
#>
#>
#> > dynare$bkk$moments
#>     VARIABLE    MEAN STD. DEV. VARIANCE
#> 1        C_H  0.8261    0.0053   0.0000
#> 2        L_H  0.6968    0.0024   0.0000
#> 3        N_H  0.3032    0.0024   0.0000
#> 4        A_H  0.6064    0.0039   0.0000
#> 5        K_H 11.0148    0.1261   0.0159
#> 6        Z_H  1.0987    0.0131   0.0002
#> 7        X_H  0.2754    0.0437   0.0019
#> 8   LAMBDA_H  1.0000    0.0108   0.0001
#> 9        S_H  0.2754    0.1047   0.0110
#> 10      NX_H  0.0000    0.0440   0.0019
#> 11       Y_H  1.1015    0.0174   0.0003
#> 12       C_F  0.8261    0.0053   0.0000
#> 13       L_F  0.6968    0.0024   0.0000
#> 14       N_F  0.3032    0.0024   0.0000
#> 15       A_F  0.6064    0.0039   0.0000
#> 16       K_F 11.0148    0.1261   0.0159
```

```
#> 17      Z_F  1.0987    0.0131   0.0002
#> 18      X_F  0.2754    0.0437   0.0019
#> 19 LAMBDA_F  1.0000    0.0108   0.0001
#> 20      S_F  0.2754    0.1047   0.0110
#> 21     NX_F  0.0000    0.0440   0.0019
#> 22      Y_F  1.1015    0.0174   0.0003
#> 23      LGM  0.2787    0.0021   0.0000
#>
#> > knitr::kable(dynare$bkk$decomposition,format='pandoc')
#>
#>
#> X             E_H     E_F
#> ---------  ------  ------
#> C_H         80.86   19.14
#> L_H         67.89   32.11
#> N_H         67.89   32.11
#> A_H         67.87   32.13
#> K_H         40.83   59.17
#> Z_H         49.75   50.25
#> X_H         42.38   57.62
#> LAMBDA_H    93.94    6.06
#> S_H         45.49   54.51
#> NX_H        37.10   62.90
#> Y_H         89.35   10.65
#> C_F         41.78   58.22
#> L_F         12.14   87.86
#> N_F         12.14   87.86
#> A_F         11.94   88.06
#> K_F         34.93   65.07
#> Z_F         27.41   72.59
#> X_F         32.56   67.44
#> LAMBDA_F    14.07   85.93
#> S_F         29.30   70.70
#> NX_F        37.10   62.90
#> Y_F          5.81   94.19
#> LGM         62.90   37.10
```

## Template

Template for R Markdown is created. Go to `file->New File->R Markdown-> From Template->DynareR`.

Please download the example files from Github.