###Description of the package

This package allows to simulate Post-Keynesian Stock-Flow Consistent Models, following the approach of Godley, W. and M. Lavoie, 2007: Monetary Economics An Integrated Approach to Credit, Money, Income, Production and Wealth. Palgrave MacMillan, New York. The package uses the Gauss-Seidel algorithm to solve linear systems of equations, following the approach found in Kinsella, Stephen and O'Shea, Terence, Solution and Simulation of Large Stock Flow Consistent Monetary Production Models Via the Gauss Seidel Algorithm (December 21, 2010). Journal of Policy Modeling, Forthcoming. Available at SSRN: http://ssrn.com/abstract=1729205 or http://dx.doi.org/10.2139/ssrn.1729205

This package works with R, more information about R can be found here: http://www.r-project.org. I recomend to use R studio (http://www.rstudio.com) for its simplicity, but any other solution will work.

I thank David O'Sullivan and Hamid Raza for their valuable help in developping the package. I gratefully acknowledge the support of the Institute for New Economic Thinking.

###Installation

The package can be downloaded here and is still in a beta version. Note that the package needs a version of R above or equal to 3.1.1. Furthermore, you will need to install expm and igraph. For those who never installed a package in R, you will need to use the following command.

```
install.packages("expm")
install.packages("igraph")
```

Once those packages have been installed, download the PK-SFC package on your computer and store it in a folder of your choice. Make sure that the name of the package is "PKSFC_1.6.tar.gz". Then run the following comand line. It will install the package from your local folder where 'pathToYourFolder' represent the path to the folder where you downloaded the package.

```
## Installing package into 'C:/Users/godina.GROUPE/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)
```

```
install.packages("pathToYourFolder/PKSFC_1.6.tar.gz",repos = NULL, type="source")
```

You are almost set, now you need to load the package. In order to do so, you should run the following line. You might have some output but if no error message is being printed this means you are now ready to use the PK-SFC.

```
library(PKSFC)
```

```
## Loading required package: expm
## Warning: package 'expm' was built under R version 3.5.3
## Loading required package: Matrix
##
## Attaching package: 'expm'
## The following object is masked from 'package:Matrix':
##
##     expm
## Loading required package: igraph
## Warning: package 'igraph' was built under R version 3.5.3
##
## Attaching package: 'igraph'
## The following objects are masked from 'package:stats':
##
```

```
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union

## Loading required package: networkD3

## Warning: package 'networkD3' was built under R version 3.5.1
```

###Testing

In order to test wether the package was correctly installed, we will run the simulation with the model SIM of the third chapter of Godley and Lavoie's Monetary Economics. To do so you need to download the source code for the SIM model (SIM.txt) in the resource tab of this page and make sure to save it in your current directory. Once this is done you should be able to run the following comand line and load the model. The code loads the model described in SIM.txt in the object 'sim', note that you can also specify a more elaborate model name using the modelName option.

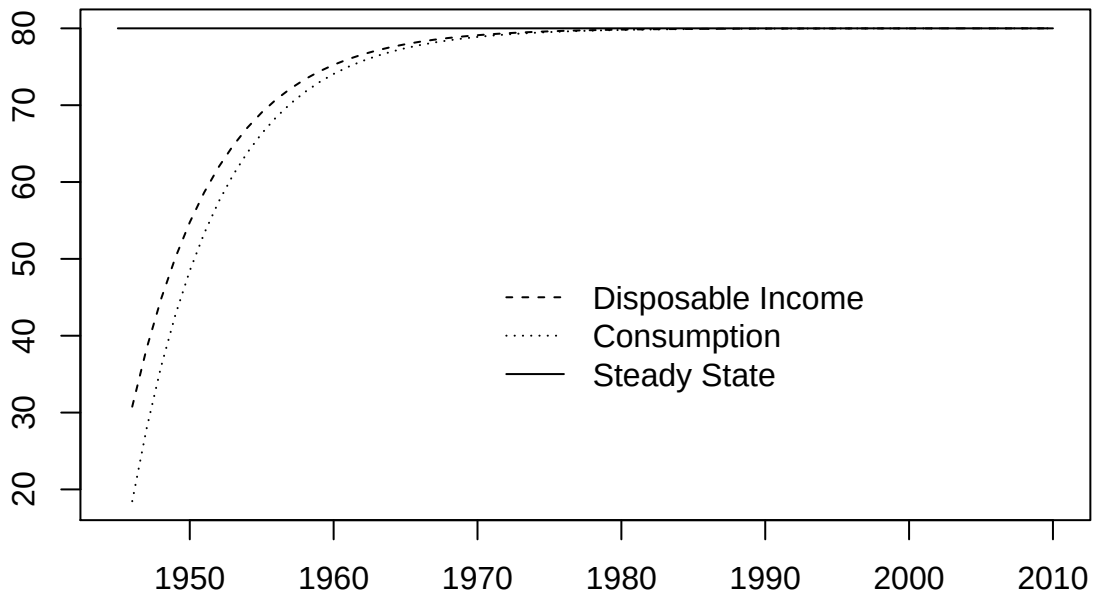
```
sim<-sfc.model("SIM.txt",modelName="SIMplest model from chapter 3 of Godley and Lavoie (2007)")
```

Once the model has been loaded, it can be simulated. The following line allows you to do that.

```
datasim<-simulate(sim)
```

Finally, to control that we have the good results, we can replicate figure 3.2 of page 73.

```
plot(sim$time,datasim$baseline[,"Yd"],type="l", xlab="", ylab="", lty=2,
     ylim=range(datasim$baseline[,c("Yd","C_s")],na.rm=T))
lines(sim$time,datasim$baseline[,"C_s"],lty=3)
lines(sim$time,vector(length=length(sim$time))+datasim$baseline["2010","C_s"])
legend(x=1970,y=50,legend=c("Disposable Income","Consumption","Steady State"),lty=c(2,3,1),bty="n")
```



###Source code of a model

Before starting with the various models of Godley and Lavoie (2007), let's have a look at the structure of the source code we used to run the SIM model. The source code can be divided in three parts:

The equation list, note that lags are represented with the '(-x)' parenthesis where 'x' is the lag.

The values for the various parameters and the initial values for the endogenous variables that need one (all variables that are used with a lag in the equation list).

The timeline. This is mainly used for data fetching and plots. But you need to specify an initial period and a last period, otherwise R doesn't know how many periods need to be simulated.

```
#1. EQUATIONS
C_s = C_d
G_s = G_d
T_s = T_d
N_s = N_d
Yd = W*N_s - T_s
T_d = theta*W*N_s
C_d = alpha1*Yd + alpha2*H_h(-1)
H_s = H_s(-1) + G_d - T_d
H_h = H_h(-1) + Yd - C_d
Y = C_s + G_s
N_d = Y/W
#2. PARAMETERS
alpha1=0.6
alpha2=0.4
theta=0.2
```

```
#EXOGENOUS
G_d=20
W=1
#INITIAL VALUES
H_s=0
H_h=0
#3. Timeline
timeline 1945 2010
```

A few important points regarding the model source code:

The file should not contain any empty lines.

You should avoid naming your variables with reserved names in R such as 'in' or 'max'.

###Direct Acyclical Graph

The following paper Visualising Stock Flow Consistent Models as Directed Acyclic Graphs, shows how any PK-SFC model can be seen as a directed graph (DG) and allows for a different representation than the traditional Transaction-Flow and Balance Sheet Matrices. Obviously these representations are more useful for large models than small models. The package includes the functions needed to obtain the DG representation. However, in order to use these functionalities, the package `Rgraphviz` needs to be installed. Because it is not stored in CRAN, you need to use the following commands to install it.

According to the latest README:

```
Rgraphviz now comes bundles with Graphviz. This should greatly simplify installation
on all platforms, compared with earlier versions.
```

Bioconductor 2.11 contains a lot of libraries that you might not want or need, but it does seem to be the easiest path to achieving what you want. These are the instructions on the Rgraphviz homepage:

```
source("http://bioconductor.org/biocLite.R")
biocLite("Rgraphviz")
```

In order to test for the success of the installation:

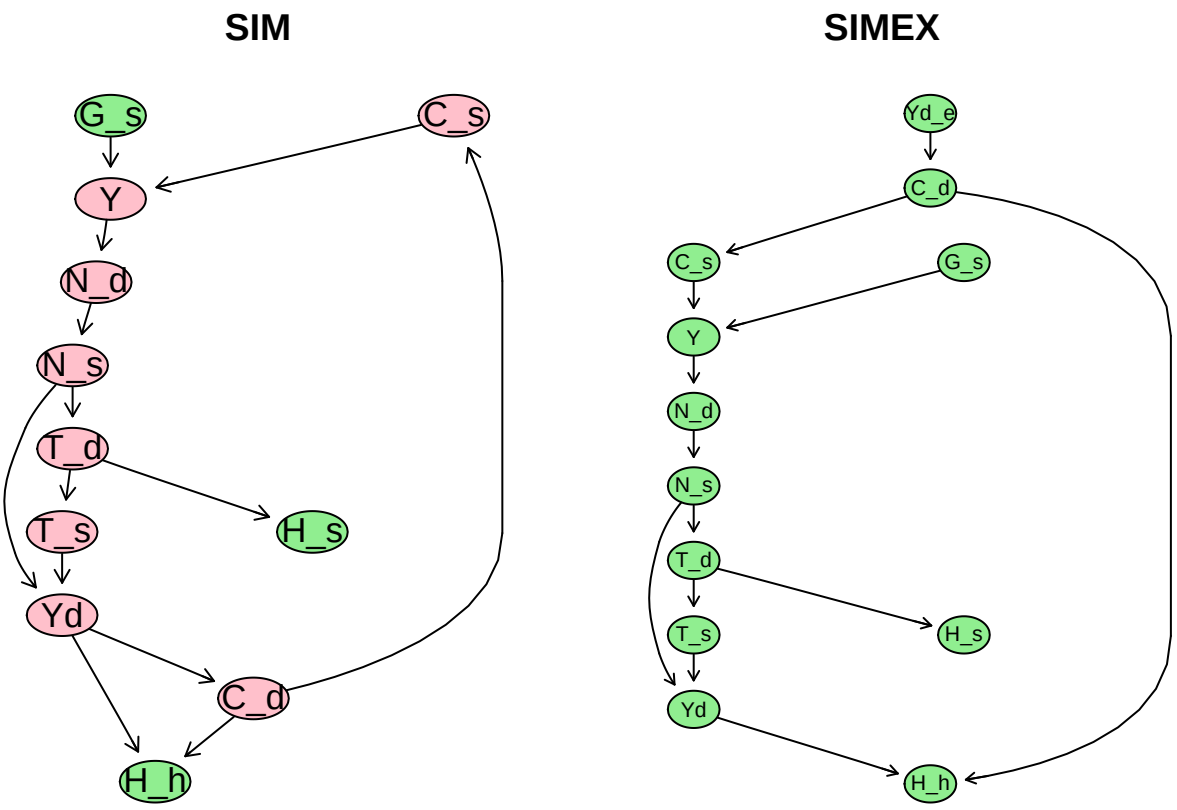
```
library("Rgraphviz")
```

```
## Loading required package: graph

## Loading required package: BiocGenerics

## Loading required package: parallel

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB

## The following objects are masked from 'package:igraph':
##
##     normalize, path, union

## The following objects are masked from 'package:Matrix':
##
##     colMeans, colSums, rowMeans, rowSums, which
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind,
##     colMeans, colnames, colSums, dirname, do.call, duplicated,
##     eval, evalq, Filter, Find, get, grep, grepl, intersect,
##     is.unsorted, lapply, lengths, Map, mapply, match, mget, order,
##     paste, pmax, pmax.int, pmin, pmin.int, Position, rank, rbind,
##     Reduce, rowMeans, rownames, rowSums, sapply, setdiff, sort,
##     table, tapply, union, unique, unsplit, which, which.max,
##     which.min

##
## Attaching package: 'graph'

## The following objects are masked from 'package:igraph':
##
##     degree, edges, intersection

## Loading required package: grid
```

We can now test these functionalities and compare the DGs of model SIM and SIMEX:

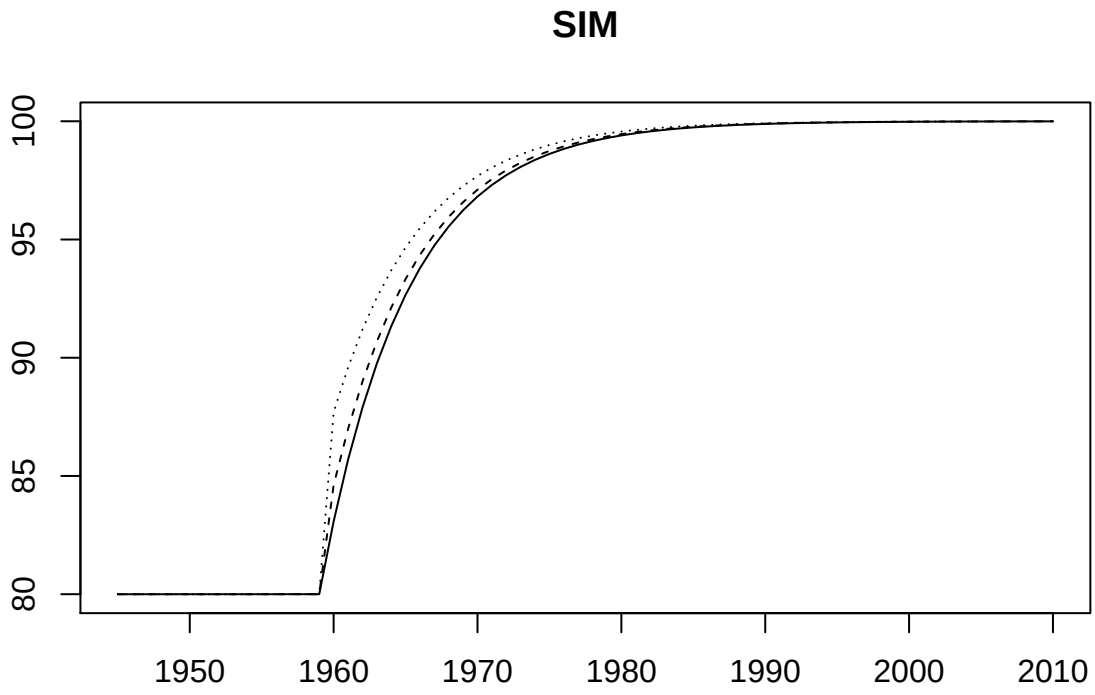
```
simex<-sfc.model("SIMEX.txt",modelName="SIMplest model with expectation")
layout(matrix(c(1,2),1,2))
plot.dag(sim, main="SIM" )
plot.dag(simex, main="SIMEX" )
```



A close look at the two generated figures shows the fundamental difference between SIM and SIMEX. The SIM graph contains a cycle (all the nodes in red), that is it is composed by a system of dependent equations that determine GDP, Employment, Consumption, Taxes and Disposable income at once. Aside from the mathematical implication that a system of equation represent, it also has an economic meaning. It implies that the economy represented by SIM will adjust in one period to any shock applied to government spending.

On the other hand, SIMEX is a directed acyclical grapg (DAG). The cycle present in SIM has been broken by having consumption depend on expected disposable income which is equal to previous period disposable income. In this case, the economy represented by SIMEX will adjust slowly to a shock applied to government spending. This can be observed in the following plots.

```
datasimex<-simulate(simex)
init = datasimex$baseline[66,]
simex<-sfc.addScenario(simex,"G_d",25,1960,2010,init)
datasimex<-simulate(simex)
datasim<-simulate(sim)
init = datasim$baseline[66,]
sim<-sfc.addScenario(sim,"G_d",25,1960,2010,init)
datasim<-simulate(sim)
plot(sim$time,datasim$scenario_1[,"H_s"],type="l",xlab="",ylab="",main="SIM")
lines(sim$time,datasim$scenario_1[,"C_d"],lty=2)
lines(sim$time,datasim$scenario_1[,"Yd"],lty=3)
legend(x=1944,y=130,legend=c("Wealth","Consumption","Disposable Income"),
       lty=c(1,2,3),bty="n")
```




```
plot(simex$time,datasimex$scenario_1[,"H_s"],type="l",xlab="",ylab="",main="SIMEX")
lines(simex$time,datasimex$scenario_1[,"C_d"],lty=2)
lines(simex$time,datasimex$scenario_1[,"Yd"],lty=3)
lines(simex$time,datasimex$scenario_1[,"Yd_e"],lty=4)
legend(x=1944,y=130,legend=c("Wealth","Consumption","Disposable Income",
                             "Expecetd Disposable Income"),lty=c(1,2,3,4),bty="n")
```