How to simulate simple economic models in *R*

Karsten Kohler

Leeds University Business School

k.kohler@leeds.ac.uk

https://karstenkohler.com

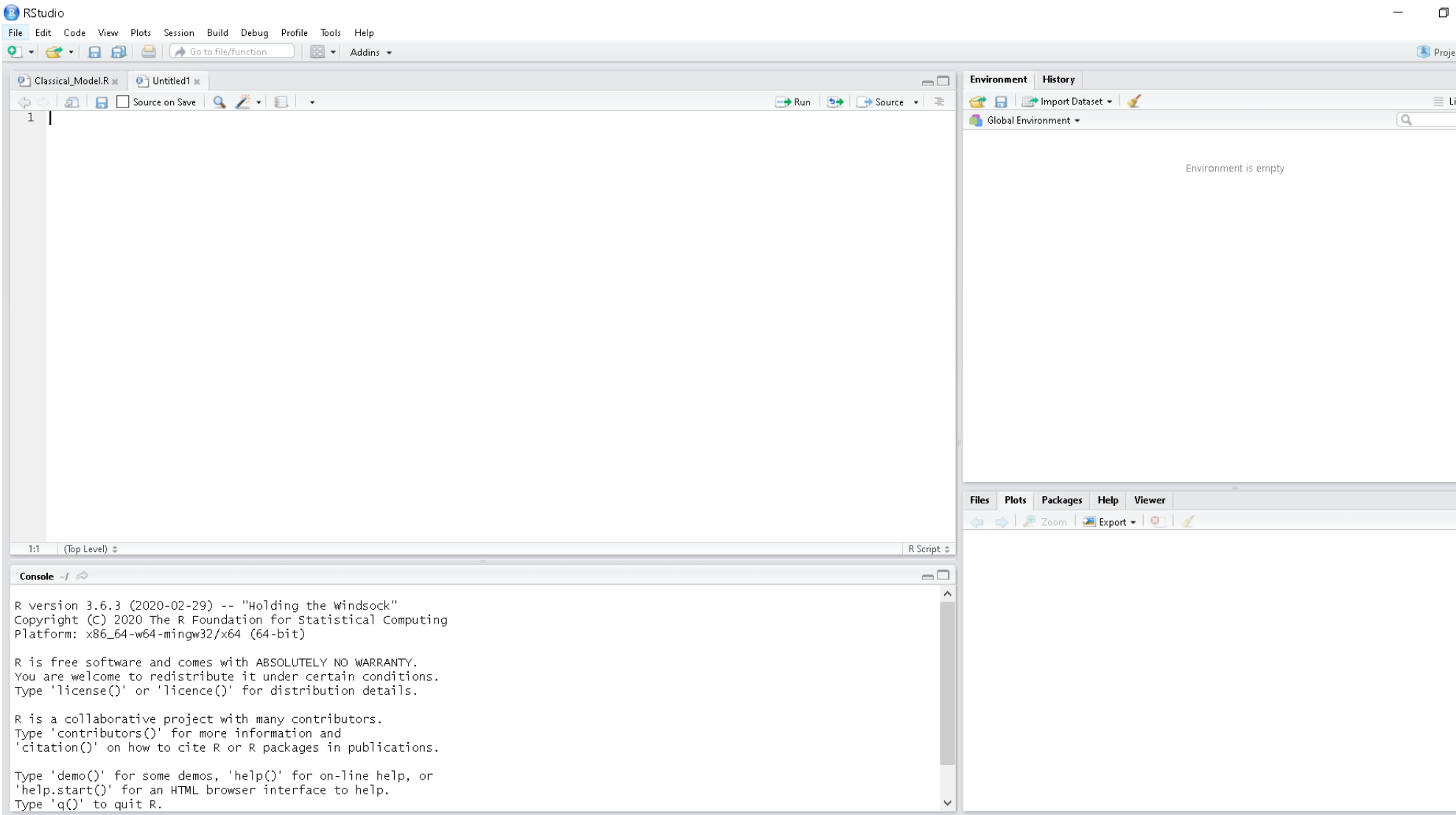
September 2022

What is *R*?

- a programming language and free software environment for statistical computing and graphics (see here)
- popular across many sciences for statistical and numerical analyses
- unlike many other packages (e.g. *EViews*, *MATLAB*, *Mathematica* and *Stata*), **R is free**
- its functionality continuously grows thanks to numerous user-written libraries and packages
- *RStudio* provides an interface for *R*, which facilitates its use

How to get *R* running on your machine

- download and install both R and RStudio
- Open *R-Studio*
- you can now use *R*


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Classical_Model.R
Untitled1
Source on Save
Run
Source
1
1:1
(Top Level)
R Script
Console ~/
R version 3.6.3 (2020-02-29) -- "Holding the Windsock"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
Environment
History
Import Dataset
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Zoom
Export

The main components of *RStudio*

- the top-left window is the script in which you type and save your code
- the top-right window displays objects and data you create
- the bottom-left windows is the console in which you can enter code that will be executed directly; it also displays some output
- the bottom-right window displays various things, most importantly plots that you create and help files

Numerical simulations in *R*

- numerical simulation of simple economic models doesn't require deep knowledge of *R*
- we will almost exclusively use the built-in functions of *Base R* (see here for a list of key commands)
- the key tool we will use to solve models is the *for-loop* (see below)

Numerical simulations of economic models: some general remarks (1/2)

- every economic model is essentially a system of N equations
- these N equations determine an equivalent number of N endogenous variables
- economic models can be:
  - *static*: time plays no role and all endogenous variables are determined simultaneously
  - *dynamic*: time matters and the endogenous variables change gradually over time
- the endogenous variables are typically interrelated: e.g. x determines y, but y also determines x
- these interrelationships can be:
  - *simultaneous*: x and y determine each other within the same period
  - *recursive*: x affects y only in $t+1$ and/or vice versa (only possible in dynamic models)

Numerical simulations of economic models: some general remarks (2/2)

- many simple economic models can be solved analytically without a computer
- numerical simulation can be useful:
  - if a model has more than 3 dimensions ($N > 3$) (then it's tedious to compute analytical solutions)
  - if a model has nonlinearities that preclude the computation of analytical solutions
  - to examine the dynamic adjustment of endogenous variables in dynamic models
- however, numerical solution requires the choice of a (possibly arbitrary) set of numerical values for the models' parameters
- it is thus less general than analytical solution

Numerical simulations of economic models: how (1/2)

- if a (dynamic) model exclusively contains recursive relationships, it can be solved iteratively by sequentially updating the endogenous variables from initial conditions
- by contrast, if interrelationships are simultaneous, solving the system for the endogenous variables is less trivial (e.g. finding the solution for x requires the solution of y, but the latter requires in turn the solution for x)
- one approach is to use linear algebra: cast the system in matrix form ($b = Ax$) and let the computer find $x^* = A^{-1}b$ through some algorithm (e.g. the Gauss-Seidel method)

Numerical simulations of economic models: how (2/2)

We will use an approach that is simpler and based on iteration:

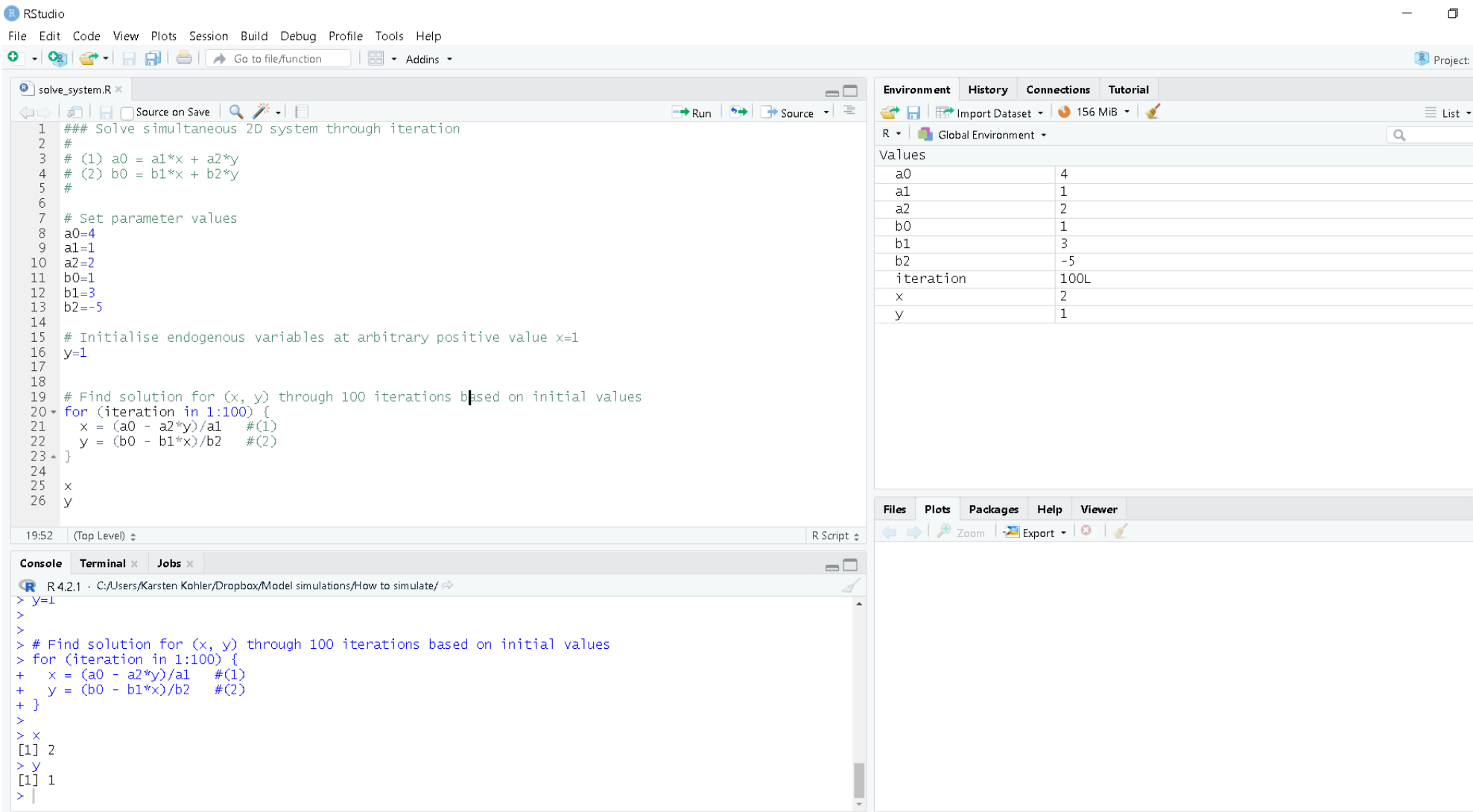
- choose (arbitrary) non-zero values for the (initial) endogenous variables
- then solve the equations many times through iteration using a for-loop
- in this way, the solution gets approximated successively

Numerical solution of a simultaneous system through iteration: example (1/3)

- consider the two-dimensional simultaneous system

$$a_0 = a_1 x + a_2 y \tag{1}$$
$$b_0 = b_1 x + b_2 y. \tag{2}$$

- suppose the parameters are given by $a_0 = 4, a_1 = 1, a_2 = 2, b_0 = 1, b_1 = 3, b_2 = -5$
- now you want to find the solution $(x^*, y^*) = (2, 1)$ through simulation


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project:
solve_system.R
Source on Save
Run
Source
Solve simultaneous 2D system through iteration
#
(1) a0 = a1*x + a2*y
(2) b0 = b1*x + b2*y
#
# Set parameter values
a0=4
a1=1
a2=2
b0=1
b1=3
b2=-5
# Initialise endogenous variables at arbitrary positive value x=1
y=1
# Find solution for (x, y) through 100 iterations based on initial values
for (iteration in 1:100) {
x = (a0 - a2*y)/a1 #(1)
y = (b0 - b1*x)/b2 #(2)
}
x
y
19:52
(Top Level)
R Script
Environment History Connections Tutorial
Import Dataset
156 MiB
List
R
Global Environment
Values
a0 4
a1 1
a2 2
b0 1
b1 3
b2 -5
iteration 100L
x 2
y 1
Files Plots Packages Help Viewer
Zoom
Export
Console Terminal Jobs
R 4.2.1 · C:/Users/Karsten Kohler/Dropbox/Model simulations/How to simulate/
> y=1
>
>
> # Find solution for (x, y) through 100 iterations based on initial values
> for (iteration in 1:100) {
+ x = (a0 - a2*y)/a1 #(1)
+ y = (b0 - b1*x)/b2 #(2)
+ }
>
> x
[1] 2
> y
[1] 1
>

Numerical solution of a simultaneous system through iteration: example (2/3)

- the code above solves the simultaneous system numerically through iteration
- you first set the parameter values
- then you initialise the endogenous variables at an arbitrary value (different from zero)
- write down the equations, solved for the endogenous variables, inside a for-loop
- then you solve the system iteratively through the for-loop
- the loop says: repeat the segment of code insights the curly brackets 100 times

Numerical solution of a simultaneous system through iteration: example (3/3)

- what happens is the following: in the first iteration, x and y are calculated based on the initial values and the parameter values
- in the second iteration, the values for x and y are then overwritten based on the results from the first iteration
- this process continues 100 times
- in this way, the correct solution $(x^*, y^*) = (2, 1)$ is successively approximated

Numerical solution of recursive system through iteration: example (1/2)

- now consider the two-dimensional recursive dynamic system

$$x_t = a_0 + a_1 x_{t-1} + a_2 y_{t-1} \tag{3}$$
$$y_t = b_0 + b_1 x_{t-1} + b_2 y_{t-1}. \tag{4}$$

- suppose the parameters are given by $a_0 = 1, a_1 = 0.3, a_2 = 0.6, b_0 = 1, b_1 = 0.3, b_2 = -0.2$
- now you want to find the solution $(x^*, y^*) \approx (2.73, 1.52)$ through simulation

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins | Project:

solve_system.R

Source on Save | Run | Source

```
27
28  ## Solve recursive 2D system through iteration
29
30  #
31  # (1) x(t) = a0 + a1*x(t-1) + a2*y(t-1)
32  # (2) y(t) = b0 + b1*x(t-1) + b2*y(t-1)
33  #
34
35  # Set parameter values
36  a0=1
37  a1=0.3
38  a2=0.6
39  b0=1
40  b1=0.3
41  b2=-0.2
42
43  # Set number of periods
44  T=100
45
46  # Construct matrices in which values for different periods will be stored; initialise at 1
47  y=matrix(data=1, nrow=1, ncol=T)
48  x=matrix(data=1, nrow=1, ncol=T)
49
50  for (t in 2:T){
51      x[,t]=a0 + a1*x[,t-1] + a2*y[,t-1]
52      y[,t]=b0 + b1*x[,t-1] + b2*y[,t-1]
53  }
54
55  x[,T]
56  y[,T]
57
58  plot(x[2:T], type="l", ylab="X", xlab="time")
59
60
```

61:1 (Top Level) R Script

Console | Terminal | Jobs

R 4.2.1 · C:/Users/Karsten Kohler/Dropbox/Model simulations/How to simulate/

```
> x[,T]
[1] 2.727273
> y[,T]
[1] 1.515152
>
> plot(x[2:T], type="l", ylab="X", xlab="time")
>
```

Environment | History | Connections | Tutorial

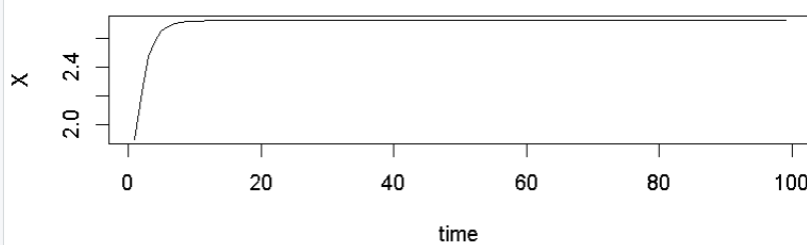
Import Dataset | 148 MiB | List

R | Global Environment

Data	
x	num [1, 1:100] 1 1.9 2.23 2.48 2.58 ...
y	num [1, 1:100] 1 1.1 1.35 1.4 1.46 ...
Values	
a0	1
a1	0.3
a2	0.6
b0	1
b1	0.3
b2	-0.2
t	100L
T	100

Files | Plots | Packages | Help | Viewer

Zoom | Export | Publish



Numerical solution of recursive system through iteration: example (2/2)

- this code solves the recursive system numerically through iteration
- we need to set a number of periods for which we want to simulate the model (here $T = 100$)
- we also create some matrices in which the solutions for each time step will be saved
- otherwise, the approach is the same: we iterate over the equations using a for-loop
- we can then plot the adjustment of the endogenous variables from the initial values to the solution

Getting started

- you can find a series of codes to simulate seminal economic models on my website
- all codes are accompanied by short notes that explain the models and also provide some analytical discussion
- feel free to get in touch if you have questions