Package: Recon (via r-universe)

September 6, 2024

Title Computational Tools for Economics

Version 0.3.0.0

Description Implements solutions to canonical models of Economics such as Monopoly Profit Maximization, Cournot's Duopoly, Solow (1956, <doi:10.2307/1884513>) growth model and Mankiw, Romer and Weil (1992, <doi:10.2307/2118477>) growth model.

Depends R (>= 3.4.0)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Imports rootSolve, stats

Suggests plot3D, dplyr, ggplot2

Repository https://pedrocava.r-universe.dev

RemoteUrl https://github.com/pedrocava/recon

RemoteRef HEAD

RemoteSha 0cb0860a8ddc3649adda00fe49a9ebec21d6b5fa

Contents

`cobb_douglas` *Cobb-Douglas Model*

Description

This function allows you to compute a Cobb-Douglas production/ utility function with n inputs/goods.

Usage

```
cobb_douglas(I, Elas = rep(1/length(I), times = length(I)), K = 1)
```

Arguments

`I`	is a vector of inputs
`Elas`	is a vector of elasticities, must be the same length as I. Defaults to equal elasticities to all inputs, with sum of elasticities equal to 1.
`K`	is the constant of the model. Defaults to 1.

Details

cobb_douglas_2 computes what - mathematically - is a particular case of this function, but computationally there are differentes. Here, the user must input two vectors, one for elasticies and one for quantities, whereas in cobb_douglas_2, the user specifies only quantities and elasticities are taken as parameters.

Value

A list with output, function's degree of homogeneity.

Author(s)

Pedro Cavalcante Oliveira, Department of Economics, Fluminense Federal University

Examples

```
I <- c(3, 4, 5)

cobb_douglas(I)
```

`cobb_douglas_2` *2 inputs Cobb-Douglas Model*

Description

This function allows you to compute a Cobb-Douglas production/ utility function with two inputs/goods.

Usage

```
cobb_douglas_2(x, TFP = 1, alpha = 0.5, beta = 1 - alpha)
```

Arguments

`x`	is a data frame with two columns.
`TFP`	is the constant of the model. Defaults to 1.
`alpha`	is the first input's elasticity. Defaults to a random number between 0 and 1, rounded to two digits.
`beta`	is the second input's elasticity. Defaults to 1 - alpha.

Value

Returns a list object with compued y and elasticities.

Author(s)

Pedro Cavalcante Oliveira, Department of Economics, Fluminense Federal University

Examples

```
x <- c(3, 4, 5)
y <- c(1, 4, 2)

data <- data.frame(x = x, y = y)

cobb_douglas_2(data)
```

`cournot_solver` *Cournot Duopoly with numeric solution*

Description

This function numerically finds the equilibrium in a Cournot duopoly model with quadratic functions. For guaranteed existence of equilibrium, cost parameters should be non-negative.

Usage

```
cournot_solver(firm1 = c(0, 1, 0), firm2 = c(0, 1, 0), demand = c(0,
  -1, 0))
```

Arguments

`firm1`	a vector of cost curve coefficients, which must be in order: intercept of firm 1's cost function, linear term's parameter of firm 1's cost function and quadratic term's parameter of firm 1's cost function
`firm2`	a vector of cost curve coefficients, which must be in order: intercept of firm 2's cost function, linear term's parameter of firm 2's cost function and quadratic term's parameter of firm 2's cost function
`demand`	a vector of demand curve coefficients, which must be in order: intercept of inverse demand function, linear coefficient, secon degree coefficient

Value

List with market price, firm output, profits and market share

Author(s)

Diego S. Cardoso, Dyson School of Applied Economics & Management, Cornell University <mail@diegoscardoso.com>

Examples

```
d = c(20,-1,0)
cournot_solver(demand = d)
```

grid2 *Cartesian coordinates generator*

Description

This function creates a grid (more especifically, a 2-cell) of coordinates in R^2. Useful for plotting and generating data points with which to apply some functions.

Usage

```
grid2(a = 0, b = 100, c = 0.5)
```

Arguments

a	is the grid's lower bound. Defaults to 0.
b	is the grid's upper bound. Defaults to 100.
c	is the "by" parameter, the grid's density. Defaults to .5.

Value

Data Frame with a grid

Examples

```
grid2(a = 0, b = 10, c = .1)
```

monopoly_solver *Monopoly Profit Maximization*

Description

This function numerically finds the profit-maximizing output for a monopolist with linear and non-linear cost and demand curves. For guaranteed existence of feasible solution (in which both price and output are positive), a linear demand curve might be necessary.

Usage

```
monopoly_solver(cost = c(0, 1, 0), demand = c(0, -1, 0), q0 = 0)
```

Arguments

`cost`	a vector of cost curve coefficients, which must be in order: intercept of the cost function, linear term's parameter of the cost function and quadratic term's parameter of the cost function
`demand`	a vector of demand curve coefficients, which must be in order: intercept of inverse demand function, linear coefficient, secon degree coefficient
`q0`	Initial guess for monopolist's output. Defaults to 0. Strongly advise not to set this parameter unless you are very aware of what you're doing.

Value

A list with market price, output, profits, markup, profitrate.

Author(s)

Pedro Cavalcante Oliveira, Department of Economics, Fluminense Federal University `<pedrocolrj@gmail.com>`

Examples

```
c = c(50, 3, 1)
p = c(500, -8, -1)
monopoly_solver(cost = c, demand = p)
```

`MRW_steady_state` *Mankiw-Romer-Weil Growth Model Steady State*

Description

This function computes steady state income, capital and human capital per worker given relevant parameters according to the MRW model.

Usage

```
MRW_steady_state(n = 0.01, g = 0.01, alpha = 0.33, beta = 0.33,
  sk = 0.01, sh = 0.01, delta = 0.01, gamma = 0)
```

Arguments

`n`	is population growth rate. Defaults to .01.
`g`	is the technological growth rate. Defaults to .01.
`alpha`	is capital-output elasticity. Defaults to .33 as estimated by Mankiw, Romer and Weil.
`beta`	is the human capital-output elascitticy. Defatults to .33 as estimated by Mankiw, Romer and Weil.

`sk`	is the savings rate devoted to physical capital. Defaults to .01.
`sh`	is the savings rate devoted to human capital. Defaults to 0.1.
`delta`	is the physical capital stock's depreciation rate. Defaults to .01.
`gamma`	is the human capital stock's depreciation rate. Defaults to 0.

Value

List with steady state capital, human capital and income per capita

Author(s)

Pedro Cavalcante Oliveira, Department of Economics, Fluminense Federal University

Examples

```
MRW_steady_state(gamma = .005)
```

`sim_nasheq` *Simultaneous Games Strategies Nash Equilibria*

Description

This function finds the Nash equilibrium in mixed or pure strategies of a 2-person simultaneous game.

Usage

```
sim_nasheq(a, b, type = "pure")
```

Arguments

`a`	The row player's payoff matrix.
`b`	The column player's payoff matrix.
`type`	The type of equilibrium to calculate. Can be either "pure" or "mixed". Defaults to “pure“.

Value

List with all Nash Equilibria

Author(s)

Marcelo Gelati, National Institute of Pure and Applied Mathematics (IMPA) <`marcelogelati@gmail.com`>

Examples

```
a = matrix(c(-8, -10, 0, -1), nrow = 2)
b = matrix(c(-8, 0, -10, -1), nrow = 2)
sim_nasheq(a, b)
sim_nasheq(a, b, "mixed")
```

`solow_steady_state` *Solow Growth Model Steady State*

Description

This function computes steady state income and capital per worker given relevant parameters according to Solow-Swan Model.

Usage

```
solow_steady_state(n = 0.01, g = 0.01, alpha = 0.5, s = 0.01,
  delta = 0.01)
```

Arguments

`n`	is population growth rate. Defaults to .01.
`g`	is the technological growth rate. Defaults to .01.
`alpha`	is capital-output elasticity. Defaults to .5.
`s`	is the savings rate. Defaults to .01.
`delta`	is the capital stock's depreciation rate. Defaults to .01.

Value

List with steady state capital and income per capita

Author(s)

Pedro Cavalcante Oliveira, Department of Economics, Fluminense Federal University

Examples

```
solow_steady_state()
```

`stackelberg_solver` *Stackelberg Duopoly with numeric solution*

Description

This function numerically finds the equilibrium in a Stackelberg duopoly model with linear functions. For guaranteed existence of equilibrium, cost parameters should be non-negative. The general functional form for a function of argument x is $f(x) = p_0 + p_1 x$. Parameters p refer to the inverse demand function. The firm indexed by "l" is the leader, and the one indexed by "f" is the follower.

Usage

```
stackelberg_solver(leader = c(0, 1), follower = c(0, 1),
  demand = c(0, -1), l0 = 0, f0 = 0)
```

Arguments

`leader`	vector of coefficients of the leader's cost function which in order must be: intercept of leader's cost function and linear term's parameter of leader's cost function
`follower`	vector of coefficients of the follower's cost function which in order must be: intercept of intercept of follower's cost function linear term's parameter of follower's cost function
`demand`	vector of coefficients of the market demand curve. Must be, in order, intercept and linear coefficient.
`l0`	Initial guess for leader's output. Defaults to 0. Strongly advised not to set this parameter unless you are very aware of what you're doing.
`f0`	Initial guess for follower's output. Defaults to 0. Strongly advised not to set this parameter unless you are very aware of what you're doing.

Value

A list with market price, firm output, profits and market share

Author(s)

Pedro Cavalcante Oliveira, Department of Economics, Fluminense Federal University <`pedrocolrj@gmail.com`>

Examples

```
l = c(100, 4)
f = c(120, 5)
p = c(300, -10)
stackelberg_solver(leader = l, follower = f, demand = p)
```

Index