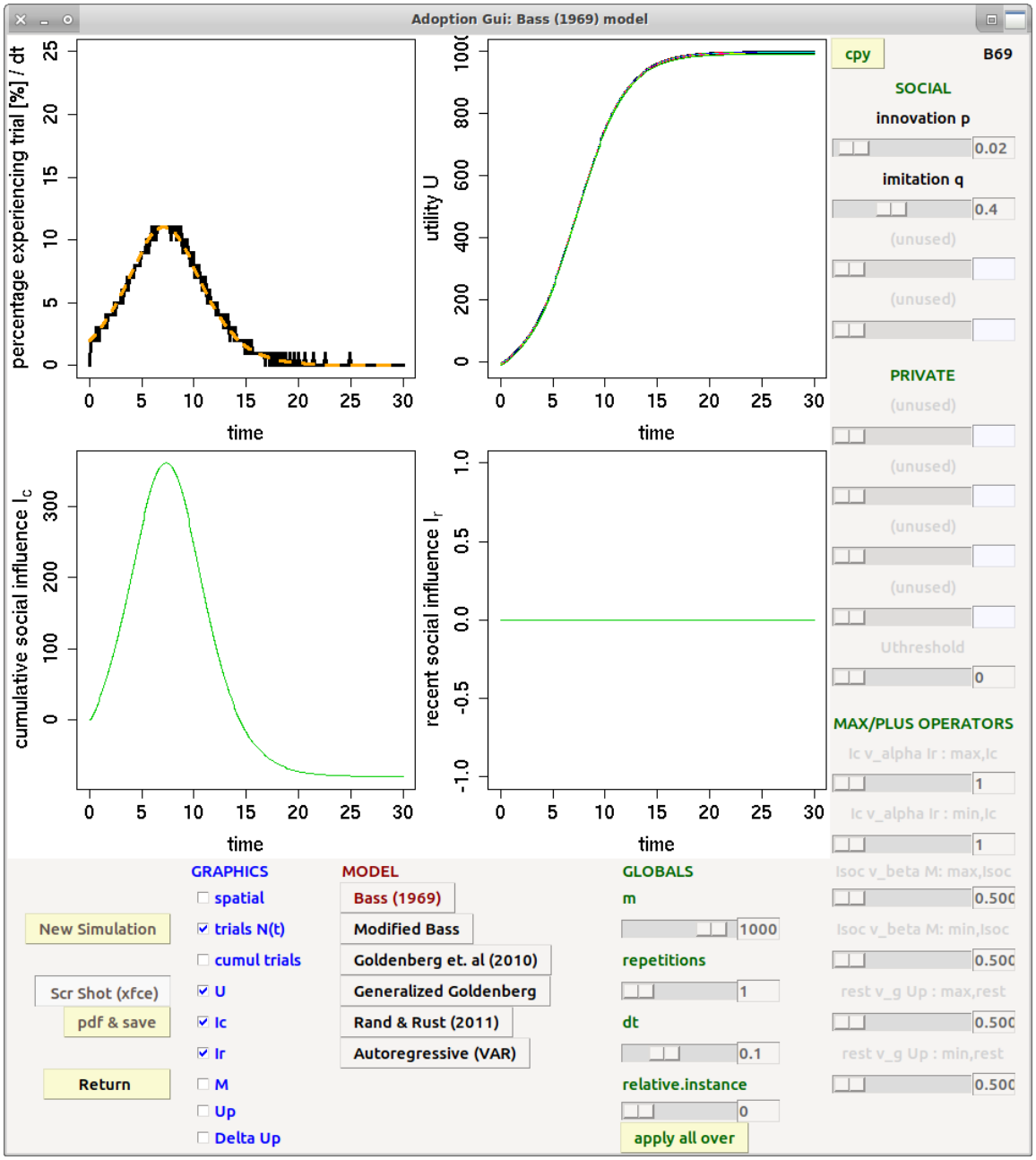
Introduction to the GUI of the R package `adoption`

Martin Schlather
Institute for Mathematics, Universität Mannheim
B6, D-68131 Mannheim
schlather@math.uni-mannheim.de

March 20, 2019



Contents

1 Introduction

Anonymous et al. (2019) propose a novel, physically motivated framework that integrates various classes of models for adoption processes such as the Bass model and its variants or agent-based models. The R package `adoption` implements the framework suggested in Anonymous et al. (2019). This tool might be particularly useful for research and educational purposes as well as managerial planning.
The gui has been developed under Ubuntu 16.04 and R version 3.5.0. It has been tested under Windows 10 on Lenovo Yoga 710 14IKB and on a MAC "El Capitan", see also Section 7 for installation details. All screen shots have been made on a Linux machine; deviations between the below pictures appear due to the different operating systems and further developments of the package.

1.1 Overview

Disclaimer: Except for Section 7, the context of the sections becomes more and more specialized. In particular, a beginner will probably loose interest within section 4 the latest, while Section 6 might be of interest only for those who aim to program their own adoption package.
At the end of a subsection there might be a technical note which gives further details on the program that might be of interest only for more specialized users.

2 Diffusion Modeling in Marketing

This section is a summary of Anonymous et al. (2019).

2.1 Introduction

Understanding the product adoption process is of considerable scholarly and managerial interest in marketing and innovation management. While early adoption models, most notably the Bass model (Bass, 1969), have considered social influence mainly in terms of social learning and imitation, agent-based models (ABM) specify the role of influentials and network effects in product adoption. For an overview, see Rand and Rust (2011), for instance. Notably, the Bass model takes a macro perspective and ABM a micro perspective, which has contributed to fragmentation and misalignment in literature.
This R package provides an integrated framework that unifies the fragmented approaches of Bass and ABM. To this end we model the adoption utility instead of a binary adoption decision. We adopt approaches by Tanny and Derzko (1988), Brock and Durlauf (2001), Van den Bulte and Lilien (2001), Risselada et al. (2014) and others and decompose the utility function twice into two complementary parts:

- aggregated utility and recent utility
- social utility and private utility (in a broad sense)

so that we obain 4 disjoint utilities:

1. the aggregated social utility I^c, also called the cumulative social utility; it captures normative pressure;
2. the recent social utility I^c; it captures word-of-mouth (WOM);
3. the aggregated private utility M, also called memory effect which keeps track of one's former utility function;
4. the recent private utility U^p, also called private utility in the narrow sense or briefly private utility; it captures external influences such as mass-media communication.

Typically, the variants of the Bass model do not consider any recent social influence I^r, while ABM do not consider cumulative social influence I^r and any memory effect M (see, for instance, Goldenberg et al. (2007)).

2.2 Modeling Framework

Typically, the utility U is defined as a weighted sum of the four elements I^c, I^r, M and U^p (e.g., the Bass model) or through maxima and minima, see the Goldenberg et al. (2010) model below. A weighted sum means that some elements (e.g., high private utility) can compensate for others (e.g., weak recent social influence), whereas the minimum function requires all elements to be above a certain threshold. Conversely, the maximum function indicates that one element may suffice (e.g., strong recent social influence) to gain enough utility to make an adoption decision. We denote by the maximum sign indexed by a greek letter, $\vee_\alpha$ for instance, any of the three binary operators: (weighted) sum, maximum or minimum. As we model the utility function at discrete time points $t_k = k\Delta t$, $k = 0, 1, 2, \ldots$ only, the utility U can be given by

$$U(t) \;=\; I^c(t-\Delta t)\Delta t \;\vee_\alpha\; I^r(t-\Delta t)\Delta t \;\vee_\beta\; M(t-\Delta t) \;\vee_\gamma\; \Delta U^p(t-\Delta t). \tag{1}$$

where the chain of operators is evaluated from the left, i.e., $x \;\vee_\alpha\; y \;\vee_\beta\; z = (x \;\vee_\alpha\; y) \;\vee_\beta\; z$. The operator $\vee_\alpha$ can be generalized to a smoothly parameterized class of operators as follows. For $\alpha = (\alpha_1, \alpha_2) \in [0,1]^2$ we define

$$x \;\vee_\alpha\; y = y + \alpha_1(x-y)_+ - \alpha_2(y-x)_+$$

where x_+ equals x if $x > 0$ and 0 otherwise. Important special cases are the convex combinations of x and y, the minimum of x and y and the maximum, which are obtained by $\alpha \in \{(a,a) : a \in [0,1]\}$, $\alpha = (0,1)$, and $\alpha = (1,0)$, respectively. Since for $c \in [0,1]$, $\alpha, \beta \in [0,1]^2$ and $\gamma = c\alpha + (1-c)\beta$ the equation $x \;\vee_\gamma\; y = c\,(x \;\vee_\alpha\; y) + (1-c)\,(x \;\vee_\beta\; y)$ holds, the ensemble of functions $\{(x,y) \mapsto x \;\vee_\alpha\; y : \alpha \in [0,1]^2\}$ equals the ensemble of all convex combinations of x, y, $\min\{x,y\}$ and $\max\{x,y\}$.
If the memory effect M is naught, some of the effects $I^c\Delta t$, $I^r\Delta t$ and U^p should be interpreted as absolute effects. If the memory effect M equals the utility $U(t-\Delta t)$ at the preceding time point, they are all rather incremental utilities. The border is nonetheless fuzzy since the memory effect can be weak. The border line is blurred further by choosing $\Delta t = 1$, what is a typical choice in ABM. Note that in our setup the modeling of the utility function is continued after adoption; in the period after adoption the utility corresponds to the degree of satisfaction.
Our framework posits that an individual becomes an adopter when their utility function U reaches or exceeds a given threshold θ, which is 0 by default.

2.3 Framework Specification

The above framework can further be specified whilst still including many models for adoption. So, I^c shall depend on t only through the total number $N(t)$ of adopters up to time t, i.e.,

$$I^c(t) = I^c(N(t)).$$

The recent influence I_i^r of an individual i is given by a linear combination of all the utilities within the market,

$$I_i^r(t) = \sum_j w_{ij} f_r(U_j(t)). \tag{2}$$

Here, the set of social weights w_{ij} (Van den Bulte and Lilien, 2001) reflects the specific social network of an individual, excluding the individual herself, i.e., $w_{ii} = 0$. Based on the notion that individuals communicate their product utility, the function f_r is included to model the transition from one's genuine utility to what is perceived by another individual. For simplicity we assume that f_r is the

same between all pairs of individuals (and independent of time).

The memory effect $M_i(t)$ is a function of one's own utility, i.e.,

$$M_i(t) = f_m(U_i(t))) \tag{3}$$

for some function f_m that is the same for all individuals. In line with standard models we even choose $f_m = f_r = f$. In this special case, M could be included in (2) choosing $w_{ii} = 1$, but we will keep I^r and M separate in favor of clearer interpretation.

In the following subsections, we give details for the implemented models; the list of models can be extended by the user, see Section 5. We assume here that we have m individuals and that the individuals are numbered from 1 to m.

2.4 Bass (1969)

2.4.1 Specification

The Bass model can be written as

$$dN(t)/dt = (m - N(t))(p + qN(t)/m) = mp + qN(t) - N(t)(p + qN(t)/m)$$

where the market potential m denotes the maximum number of adopters. The term mp refers to the individual's private appreciation of the product. The terms qN and $-N(t)(p+qN)$ refer to the increasing and decreasing effect on an individual, respectively, when having many other adopters. The decreasing effect is also necessary to control for the total number of adopters (Bass, 1969). The Bass model fits into our framework in a discretized version,

$$N(t) = -N(t - \Delta t)(p - q + qN(t - \Delta t)/m)\Delta t + N(t - \Delta t) + mp\Delta t \tag{4}$$

for discrete time instances $t_0 + k\Delta t$. The idea for a physical interpretation of the Bass model through microscopic modeling is to assume that the utility of all individuals is proportional to N, but that individuals start with different (negative) utilities at time $t_0 = 0$. More precisely, for individual i we define

$$U_i(t) = N(t) - i$$

so that all individuals have the same dynamic. Let the social influence I^c be $I^c(N) = -4N(p - q + qN/m)$ and the private utility be linearly increasing, $U_i^p(t) = 2mpt - i$. Then for any i equation (4) is equivalent to

$$U_i(t) = I^c(N(t - \Delta t))\Delta t \ \vee_\alpha \ I^r \ \vee_\beta \ 4U_i(t - \Delta t) \ \vee_\gamma \ \Delta U_i^p(t - \Delta t) \tag{5}$$

where $\alpha = (1, 1)$, $\beta = \gamma = (1/2, 1/2)$ and I^r is unimportant due to the choice of α. Hence, equation (5) obeys (1) and (3) with $f(x) = 4x$. Since $U_i(t) \geq 0$ if and only if $N(t) \geq i$, and $U_1 > U_2 > \ldots > U_n$, the number of adopters $N(t)$ equals the number of individuals with utility at least 0.

2.4.2 Implementation details

Since for each i the dynamics in (5) is the same as for the discretized Bass model, model (5) converges to the Bass model as $m \to \infty$ and $\Delta t \to 0$. A close approximation is already obtained for $m = 1000$ and $\Delta t = .1$.

2.5 Modified Bass Model

2.5.1 Specification

The approach of the Bass model through an agent based model as given in Anonymous et al. (2019) allows for various modifications and generalizations. First, the parameter p appears in both the social utility I^c and the private utility U^p. It might be then more natural to introduce a further parameter s:

$$dN(t)/dt = sm - N(t)(p - q + qN(t)/m).$$

Next, replacing the deterministic starting value $U_i^p(0) = -i$ by independent uniformly distributed $U_i^p(0)$ on $[-m, 0]$ shows that this minor modification introduces a visibly greater variability of $N(t)$ even when m is rather large. We have fixed that an individual will buy the product when her utility exceeds the threshold 0 for the first time. Allowing different values for the threshold shows that the Bass model is not too sensitive to this parameter. Finally, allowing for general operators $\vee_\beta$ and $\vee_\gamma$ shows that the value of U is very sensitive to β and γ whereas $N(t)$ is rather sensitive only to β_1 and γ_2.

2.5.2 Implementation details

The default values are the same as for the Bass (1969) model; particularly $s = p$.
As for the Bass, the theoretical curve is shown for which $s = p$.

2.6 Goldenberg et al. (2010)

2.6.1 Specification

In their agent-based model, Goldenberg et al. (2010) place individuals on a two-dimensional grid, whereby every utility is primarily influenced by its 8 nearest neighbours on the grid, the so-called eight-cell Moore neighborhood. Adoption of an individual occurs with probability $p_i(t)$

$$p_i(t) = \begin{cases} 1 - (1-a)(1-b)^{n_i(t)}, & \text{if } N(t)/m > h_i \\ 0, & \text{otherwise.} \end{cases}, \qquad t \in \mathbb{N}$$

where h_i is an individual Gaussian threshold with mean μ and variance $\sigma^2\mu^2$, $n_i(t)$ is the number of adopters in the Moore neighborhood, and $a, b \in [0, 1]$ are constants.

The Goldenberg et al. (2010) model can be described by the following formula

$$U_i(t) = I_i^c(N) \ \vee_\alpha \ \sum_j w_{ij} f(U_j(t-1)) \ \vee_\beta \ f(U(t-1)) \ \vee_\gamma \ \Delta U_i^p(t-1),$$

where $\alpha = (0, 1)$, $\beta = (1, 0)$ and $\gamma = (0.5, 0.5)$. The utility at starting point $t_0 = 0$ equals $U(0) = U_i^p(0) = c$ for some arbitrary number $c < 0$ so that nobody is an adopter at t_0. Further, $I_i^c(N) = \infty$ if $N > mh_i$ and $-\infty$ otherwise, so that the utility is negative whenever the condition $N(t)/m > h_i$ is not satisfied. If the latter is satisfied, we have

$$U_i(t) = \sum_j w_{ij} f(U_j(t-1)) \ \vee_\beta \ f(U(t-1)) \ \vee_\gamma \ \Delta U_i^p(t-1).$$

Let $w_{ij} = 1/C$ for some $C > 0$ if individuals i and j are in their mutual Moore neighbourhood and 0 otherwise. Let $f(U) = C\mathbf{1}_{U \geq 0}$. Then,

$$\sum_j w_{ij} f(U_j(t-1)) = n_i(t).$$

The interpretation of the Goldenberg et al. (2010) model is that each adopter in the neighborhood of an individual contributes positively and with the same value to the value of $U_i(t)$. Assume that ΔU^p is bounded from below by some number z and $C > -z$. Then $U(t) > 0$ whenever $U(t-1) \geq 0$. So we finally consider the case where $U(t-1) < 0$ (and $N(t)/m > h_i$). Let $n = 8$ be the number of neighbours in the Moore neighbourhood and $z < -n$. Define

$$\Delta U_i^p(t) = \max\{z, -\log(Z_{it}/(1-a))/\log(1-b)\}$$

for i.i.d. random variables $Z_{it} \sim U[0,1]$. Then

$$\begin{aligned}
\mathbb{P}(n_i(t) + \Delta U_1^p(t) \geq 0) &= \mathbb{P}(-\log(Z_{it}/(1-a))/\log(1-b) \geq -n_i(t)) \\
&= \mathbb{P}(\log(Z_{it}/(1-a)) \geq n_i(t)\log(1-b)) \\
&= \mathbb{P}(Z_{it} \geq (1-a)(1-b)^{n_i(t)})
\end{aligned}$$

so that the probability p_i in the Goldenberg et al. (2010) model is met. Note that the contribution by the private utility $\Delta U_i^p(t)$ is positive only with probability a, while the society contributes only positively.

2.6.2 Implementation details

We place the individuals on a square grid with grid points $1, 2, \ldots, \sqrt{N}$ in each direction. Let p_i be the position of individual i on the grid. The weights w_{ij} equal $1/C$ if $i \neq j$ and the Euclidean distance between p_i and p_j is less than or equal to d. For the Moore neighborhood we have $d = \sqrt{2}$.

2.7 Generalized Goldenberg et al. (2010) model

As in the modified Bass model the parameters of the operators $\vee_\beta$ and $\vee_\gamma$ allow arbitrary values in the generalized Goldenberg et al. (2010) model. Furthermore, the threshold for the Euclidean distance as described in the preceding subsection can be varied. As a genuine novel parameter the probability of aversion is introduced, i.e., the sign of w_{ij} in the weight matrix is flipped independently with probability `"prob of aversion"` for each $i, j = 1, \ldots, m$.

2.8 Rand and Rust (2011)

2.8.1 Specification

The general requirements of an ABM model outlined in Section 6 of Rand and Rust (2011) can be met by our generalized framework. Rand and Rust (2011) suggest that adoption occurs at time t if

$$X_1 < z_2 \text{ or } X_2 < \frac{z_1}{n_i^*} n_i(t)$$

for two independent, on $[0,1]$ uniformly distributed random variables X_1 and X_2, n_i^* the number of individuals in the neighborhood and $n_i(t)$ the number of adopters at time t in the neighborhood. It is easily seen that the adoption probability $p_i(t)$ equals

$$p_i(t) = z_2 + \frac{z_1}{n_i^*} n_i(t) - \frac{z_2 z_1}{n_i^*} n_i(t).$$

Hence, in order to obtain an example of a model that fits the framework of Rand and Rust (2011), it suffices to take the model of Goldenberg et al. (2010) with $\Delta U_i^p(t)$ replaced by

$$\Delta U_i^p(t) = \left(\frac{Z_{it}}{1 - z_1} - 1 \right) \frac{n_i^*}{z_2}$$

for constants $z_1, z_2 \in (0, 1)$. Assuming that n_i^* is the same for all i we get with $\tilde{z}_2 = n_i^*/z_2$ that

$$\Delta U_i^p(t) = \tilde{z}_2 \left(\frac{Z_{it}}{1 - z_1} - 1 \right).$$

Again, the contribution of ΔU_i^p is positive with a probability less than 1, here z_1.

2.8.2 Implementation details

The Rand and Rust (2011) approach is very general and we picked out a specification that is very close to the Goldenberg et al. (2010) model. So, for instance, as in the Goldenberg et al. (2010) model, we model the neighbourhood by means of the Euclidian distance.

2.9 A Vector-Autoregressive Approach

2.9.1 Specification

Vector auto-regressive models are typically used to model multivariate time series in economics and finance, such as joint price developments of different products. For instance, Dekimpe and Hanssens (1999) use such a (non-spatial) VAR model to jointly model the a brand's sales performance, its marketing budget, and the competitiors marketing spending. The spatial VAR model is used in economics (see Beenstock and Felsenstein (2007), for instance) to model spatially located agents, e.g. markets, but also to model the spatial spread of a utility, see Durlauf and Ioannides (2010).
The general model for VAR is given by

$$U(t + 1) = WU(t) + \varepsilon(t) \tag{6}$$

Here, the vectors $\varepsilon(t)$ are independent in time and consist of independent variables, frequently chosen as being Gaussian (Johansen, 1991). A row of the weight matrix W reflects the neighborhood of an individual, cf. Section 2.3.
The VAR model can be included in our framework if the diagonal elements of W are all the same. If $W_{11} \neq 0$, we have for w_{ij} in (2) that $w_{ii} = 0$, and $w_{ij} = \frac{W_{ij}}{4W_{11}}$ for $i \neq j$. Further, $f(U) = 4W_{11}U$, $\Delta t = 1$, $\alpha = (1, 0)$, $\beta = \gamma = (1/2, 1/2)$, $\Delta U^P = 2\varepsilon$. If $W_{11} = 0$, a similar but simpler model representations can be given.

To conclude we note that linear systems of differential equations are well-known for modeling the spread of information (Mahajan et al., 1995; Peres et al., 2010). Our VAR model can be seen as a discretized version of a linear system of stochastic differential equations:

$$dU_i(t)/dt \;=\; \sum_j w_{i,j} U_j(t) + dU_i^p(t)/dt, \qquad i = 1, \ldots, m. \tag{7}$$

2.9.2 Implementation details

In the R package `adoption`, $w_{ij} = \exp(-\kappa \operatorname{dist}(i,j))$ for some $\kappa \in \mathbb{R}$. Here, $\operatorname{dist}(i,j)$ denotes the Euclidean distance between individuals i and j. Hence, $w_{ij} \to 0$ as $\kappa \to \infty$ and the eigenvalues of W are all within $(-1,1)$ for κ large enough and β any fixed value in the interval $(0,2)$. In this case we have a stationary model for the utilities.

3 The gui

The package `adoption` primarily provides a real-time graphical user interface (gui) for modeling adoption processes, which is described in the following.

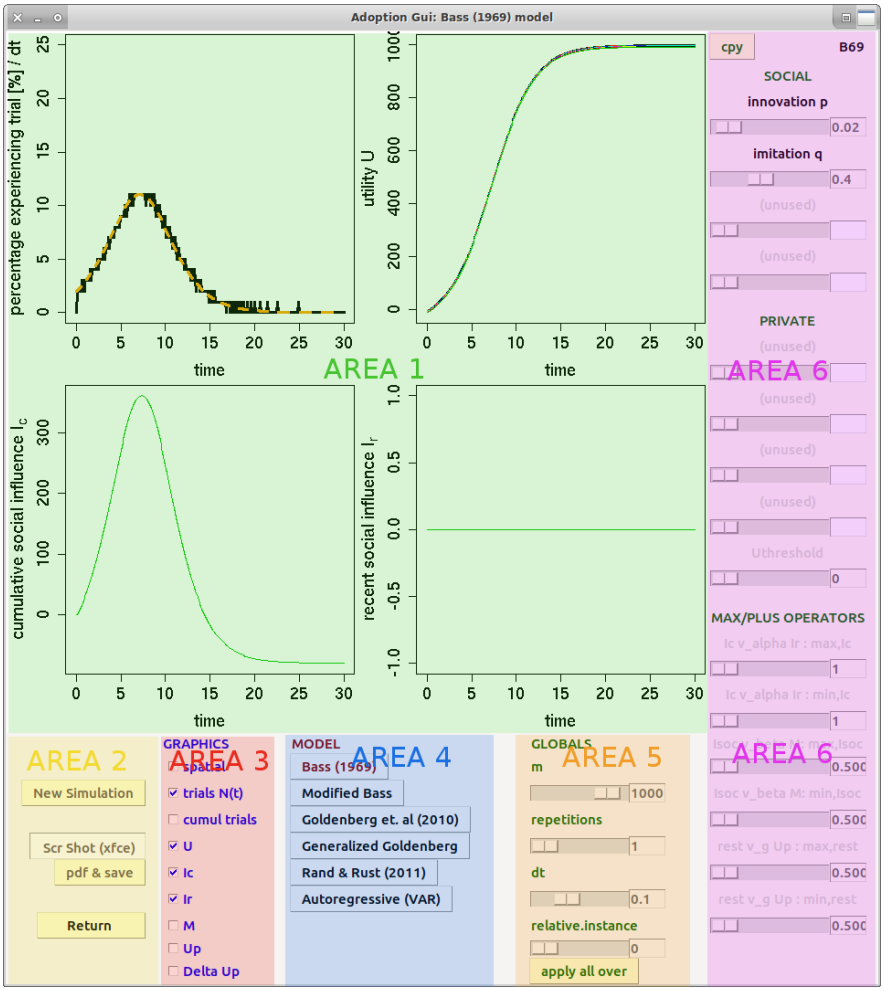


Figure 1: The 6 different areas of the gui

3.1 Start with

The gui is simply started with

```
library(adoption)
a <- adoption()
```

You will get the gui presented on the cover sheet, showing Bass (1969) model. On the first right column, see Figure 1 (AREA 6), only the very first sliders are active, namely those for the two parameters of the Bass model. When slipping the sliders the resulting number of trials per time instance (and all the other figures in the simulation area (AREA 1) will change in real time.
If there is a speed problem when slipping the sliders call

```
a <- adoption(quantiles=NULL)
```

so that some minor, but time consuming details will not be shown. Or use parallel computing by calling

```
a <- adoption(cores=2) ## or more than 2
```

if you system supports OMP.
The gui is devided into the following areas, which are indicated in Figure 1:

1. general stuff such as return button and a button for restarting simulations with a different seed (AREA 2);
2. model choice (AREA 4);
3. general parameters: parameters that belong to all the models such as the size m of the market or the number of repetitions of simulations (AREA 5);
4. model specific parameters (AREA 6);
5. up to 4 pictures can used to depict the temporal or spatial development of adoptors and utilities (AREA 1), depending on the user's choice (AREA 3).

Technical note: When a slider is moved, typical reaction times are $0.1-0.2$ seconds so that about 5-10 frames per second become visible. The exact reaction time depends, of course, on the model, its parameter specifications and the computer itself.
The sliders and buttons on the bottom (AREA 5) or on the very right column (on Windows systems) change fundamental parameters and can lead to considerably higher reaction times. These parameter should be changed with care.

3.2 General buttons (AREA 2)

AREA 4 has four or six buttons, which are described in the following subsections.

3.2.1 New Simulation

This button causes the gui to perform the calculations on a new set of realisations of the involved random variables.

Technical note: Particularly in agent based models, random variables are involved at different parts of the modeling process, e.g. as random initial value for the private utilties and as random decisions whether the product will be adopted given that a certain number of neighbours has already adopted. To ensure comparability of the realisations based on different parameter values of the model, at least the random seed at the very beginning must be stored. For efficiency, the random seed for all parts of the model are stored in `adoption`, so that only that only a part has to be resimulated, when a parameter value is changed. The button "New simulation" defines a new seed for every part of the model and resimulates all the random variables used in the model.

3.2.2 Fit 1 step (not shown in Figure 1)

When data have been passed to adoption, this button appear in this area and row of tick boxes appears to select the parameters that shall be fitted (AREA 6). The button "fit" performs an ordinary least squares fit with the selected parameters. When the button is pressed a second time for the same model, then the fitting result is deleted. Since the fitting can take a rather long time, information upon the progress of the fitting is shown. While under Linux (and alike) the values are shown in the xterm, an additional window opens under Windows. When closing the gui, all output is shown on the console also under Windows.
By default, all parameters have a hook in the box to left of its slider and will be fitted. An exception is the parameter `Uthreshold` and the parameters for the operators which are never suggested to estimated. All parameters that are not fitted take their values from the gui. The current values in the gui also build the starting values of the fitting process.

Technical note: There are many additional options to direct the fitting. Among others these are `fit_m`, `cumfit`, `fit_repetitions`, `fit_window`, `pgtol`, `factr`. See the manual pages of `RFoption` for details on the options and Section 6.3 for details on the fitting procedure.
The parameter dt is given by the data and should never be changed in the gui when data are given. In most cases, a higher value of `fit_repetitions` entails higher precision, but also longer calculation times. Integer valued model specific parameters are never fitted with one exception. If the `adoption` is called with the argument `fit_m=TRUE`, then the market size m is fitted.

3.2.3 Complete fit (not shown in Figure 1)

The fitting procedure uses a local approximation of the RSS which is necessary because of the generally high collinearities among the parameters. The button `fit 1 step` performs such an optimization. Since the approximation is good only locally, but fixed for the whole fitting procedure, the result is not optimal. Because of this the `complete fit` iterates the `fit 1 step` until an improvement cannot be seen anymore (through a simple criterion). Clearly, the `complete fit` is time consuming.

Technical note: the stopping criterion is given by the optional argument `max_increasing`.

3.2.4 Screen shot

This button takes a screen shot of the gui, see Figure 2 for an example. This button is implemented mainly for Linux users of Xfce.

3.2.5 Save & PDF

This button does two things:

1. it saves all possible graphics in separate pdf files;
2. it saves the whole session in an `rda` file. A session can be restored if `adoption` is restarted with argument `user` being the file name.

Important: The button is programmed in a rather simple way, so that it is best to use empty directories for saving.

3.2.6 Return

The gui is terminated and the adjustments of the whole session including all sets is returned (i.e., the same information as for "PDF and save" above).

Technical note: The precise behaviour of the return button depends on the `wait` argument of the function `adoption`. If `wait` is negative, the current session model is stored in `.adoption.exit` in the `.GlobalEnv` environment. If `wait` is non-negative, the gui will return the current session model (after at most `wait` milli seconds).

3.3 Model choice (AREA 4)

In AREA 4, the user may choose a model among a default list of models, where the currently chosen model is given in red. Furthermore, on Linux systems, on the very top right of AREA 6, an abbreviation of the model name is shown. According to the model choice, different parameters become relevant and the names of the parameters may change.

Technical note: for each model, the currently used model parameters are stored, so that the former parameter values reappear when switching back to a model.

3.4 General parameters (AREA 5)

The gui knows 4 general parameters,

1. the market size m
2. the number of repetitions in the simulations. Note that the Bass model is deterministic; changing this value will not lead to different results in this case.
3. the time increment dt and
4. the instance of time in the spatial representations, see Section 3.6.9.

Technical note: The first three parameters are fundamental and trigger a complete resimulation of all the random variables when their value is changed. So, `adoption` might need a considerable amount of time to react when these values are changed.
Although the first three parameters are common to all models, different frozen parameter sets (see Section 3.5.1) may have different values. To make them equal for all sets the last buttom "apply all over" must be pressed. This button has not been tested intensively yet.

3.5 Model parameters (AREA 6)

In AREA 6, the user can change the parameter values of the current model either by slipping the slider or by changing the value in the entry box.

Technical note: Here, we only describe how the gui works; for the meaning of the parameters we refer to Section 2. The interface for the parameters (AREA 6) has the following characteristics:

1. Grey scales signify importance. The lighter the names of the parameters are printed the less important are the paramters considered.

2. For each part of the model, space for a certain number of parameters is reserved. If the current model does not use the full space the place holder "(unused)" appears; changing the value of the place holder does not have any effect.

3. If the parameter used in the model is fixed, then the slider cannot be moved. The latter happens especially for the parameters of the operators $\vee_\alpha$, $\vee_\beta$ and $\vee_\gamma$ which are fixed for most models given in literature.

4. Sliders have different underlying scales depending on the fact whether the variable is integer valued, positive or real valued.

5. Entry boxes allow to supply arbitrary numbers, even outside the mathematical domain of the parameter. Parameter values outside their domain can indeed be of interest, e.g., the definition of the parameter α of the operator $\vee_\alpha$ can be extended beyond $[0, 1]^2$. An additional effect of the entry box is that the boundary of the slider might be changed so that invalid values might be obtained subsequently through the slider as well.

6. Note that the entry boxes react instantaneously when the contents is changed. This is mostly the desired behaviour as one can see directly the effect of the changings.

   This instantaneous reaction might be stopped (e.g. when a completely new number is typed in) by first typing a non-interpretable sign, e.g. a '#', which must be removed at the end.

7. At the very top left (AREA 6) there is a button called "cpy" which will store the current model and its parameters in a separate column. See Section 3.5.1 for details; on Windows systems this button is on the very top on the right hand side.

3.5.1 Sets

With the button "cpy" on the top left of the right column (AREA 6) (on Windows systems: on the top right), the current model and its parameters and realization are frozen. Figure 2 shows an example with one frozen model. Up to 4 frozen models are possible. The frozen models and the current models are called sets. The column(s) with the frozen model(s) have the following characteristics:

1. On the top an abbreviation of the model is given.

2. The color given on the top corresponds to the color in the graphics. Note that the color scheme is always kept in the following way: the current model is in black, first frozen model is in red, second frozen model is in blue, etc.

3. The button "d" on the top deletes the frozen model.

4. The button "1" ("1st") on the top switches the current model and the frozen model.

5. Frozen models do not have sliders, but do have entry boxes.

6. The graphics show the current model and the frozen model(s). Note that the scale of a graphic is always given by the current model so that the graph of a frozen model might be even not visible.

Technical note: When the value of a entry boxes is changed in a frozen box, the gui visibly first exchanges the current column and the frozen column, then performs a new simulation, and then switches back. This is slow.

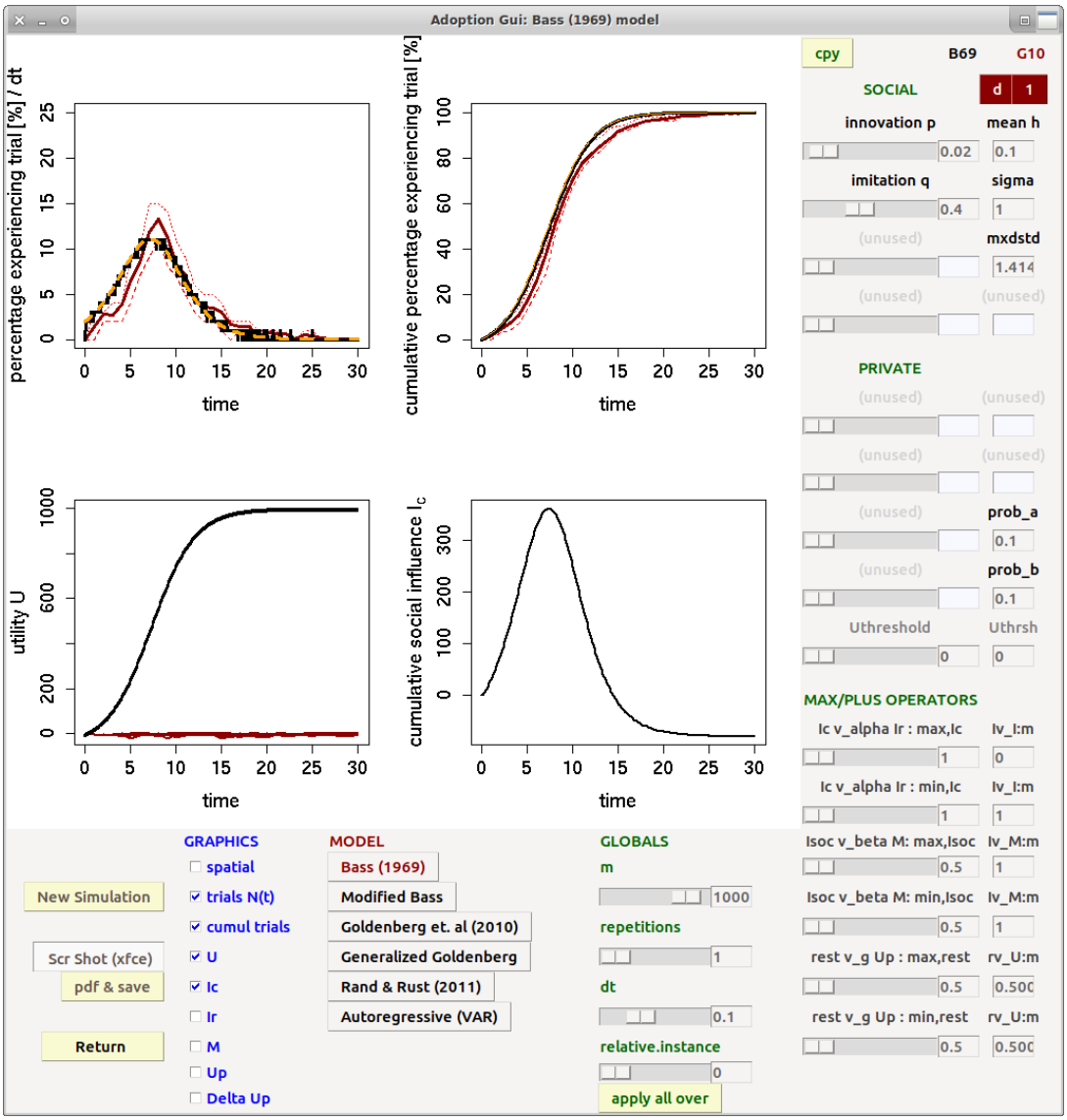


Figure 2: Screen shot for Linux and MAC system, which shows one additional fixed parameter set (second column from the right). Black curves: Bass (1969) model; red curves: Goldenberg et al. (2010) model.

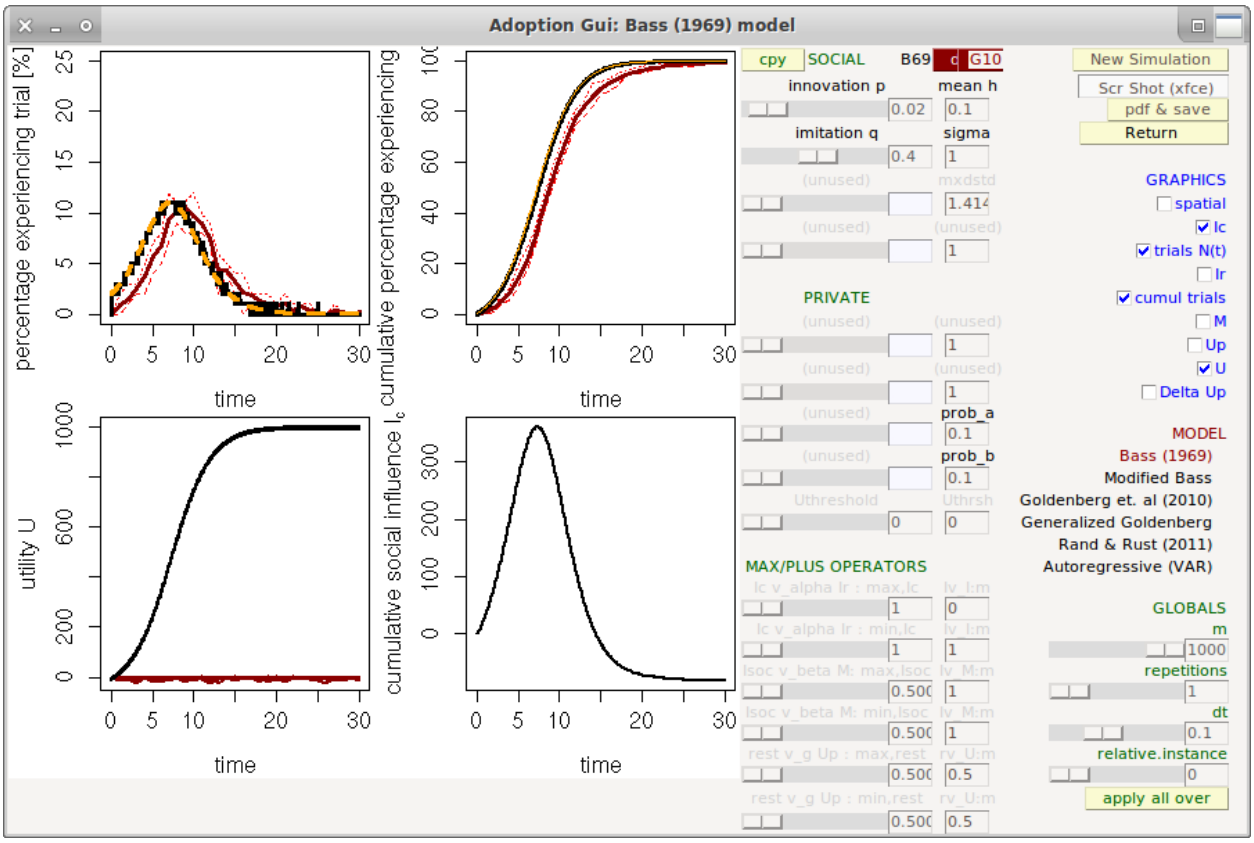


Figure 3: Screen shot typical for Windows systems, which shows one additional fixed parameter set (second column from the right). Black curves: Bass (1969) model; red curves: Goldenberg et al. (2010) model.

3.6 Graphics (AREA 1, AREA 3)

Graphics are drawn until the place for four plots is filled or the gui runs out of the list of ticked boxes. The plotting always starts at the top left. In the following the meaning of the tick boxes are described.

In each of the below figures we give realisations for the figures visible at starting point when choosing one of the main four models of the package, the Bass (1969) model, the Goldenberg et al. (2010) model, the Rand and Rust (2011) model and the VAR model.

Technical note: The figures have been obtained by the following code:

```
library(adoption)
RFoptions(fontsize = 12)  # on windows choose 'fontsize = 8'
set.seed(0)
g <- function(sw, fn) adoption(startwith=sw, gui=FALSE, filename=fn)
g(1, "bass")
g(3, "goldenberg")
g(5, "randrust")
g(6, "VAR")
```

3.6.1 Cumulative number of trials N

The cumulative number of trials in most of the adoption models show an S-shaped curve, which is caused by a small number of adopters at the beginning and at the end of the period.

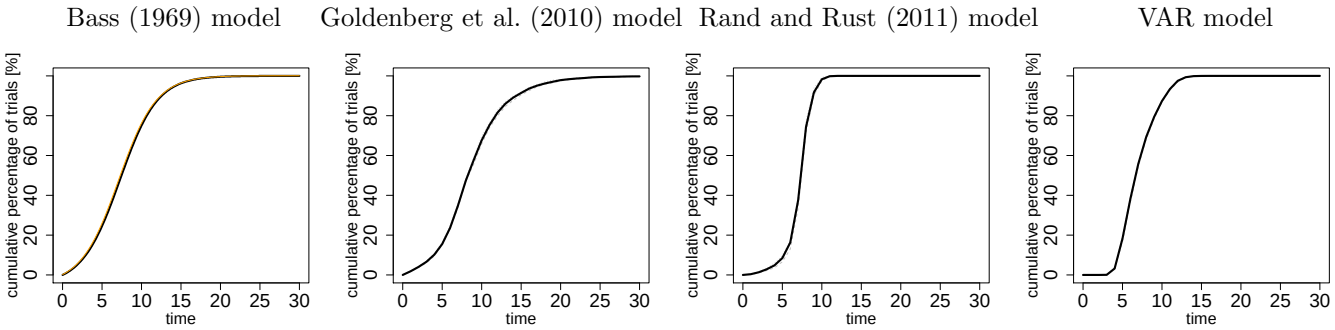


Figure 4: Cumulative number of trials. All figures are based on 1000 individuals. The orange curve on the very left figure shows the theoretical curve in the Bass (1969) model; obviously, the theoretical curve and the black curve obtained through the approximative model given in Section 2.4 can hardly be distinguished.

Technical note: In the gui, the S-shaped graph of the cumulative trials is always given in percent. If `repetitions` is larger than 1, the black curve is the mean value of all individuals in all repeated simulations. If, additionally, `quantiles` is not `NULL` in function `adoption`, the quantiles among the repetitions are also plotted, namely in a lighter color and in different line type. Note that calculating the quantiles is very time consuming, taking more than 60 % of the total computing time.

3.6.2 Number of trials per unit of time dN

The number of trials dN gives the increments of N. In most models dN is small at the beginning and at the end of a period and is (essentially) unimodal.

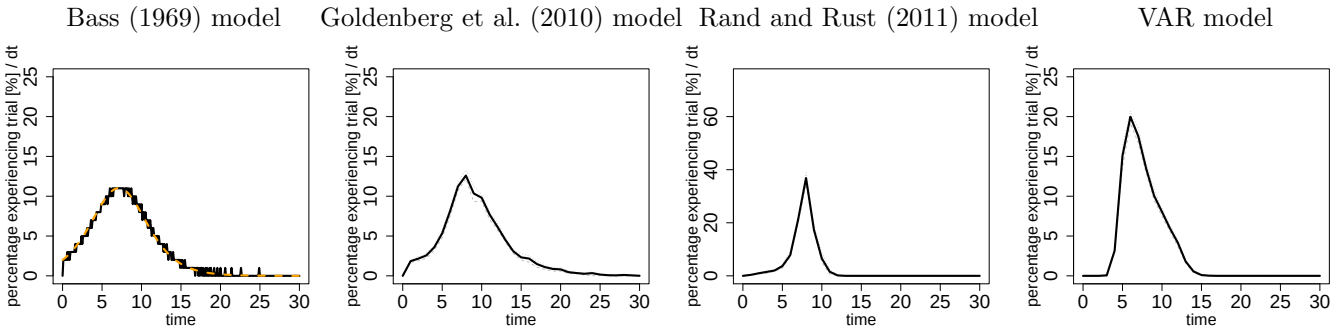


Figure 5: Number of trials per unit of time.

Technical note: In the gui, the orange curve on the very left figure shows the theoretical curve in the Bass (1969) model; obviously, the the black curve obtained through the approximative model given in Section 2.4 is an excellent discretization of the theoretical curve. The intensity of the trials, not the trials themselves are given, so that the result of the Bass model remains stable when the slider "dt" is moved.

If `repetitions` is larger than 1, the black curve is the mean value of all individuals in all repeated simulations. If furhter `quantiles` is not `NULL` in function `adoption`, the quantiles among the repetitions are also plotted, namely in a lighter color and in different line type. Note that calculating the quantiles is very time consuming, taking more than 60 % of the total computing time.

3.6.3 Utility U

The utility $U(t)$ is based of the two social influence I^c and I^r, the memory effect M and the private utility U^p. It is the key quantity in the model for two reasons:

1. The first time $U(t)$ exceeds a certain threshold θ, here 0, the product is adopted. Hence, by θ, the non-observable $U(t)$ is related to the observable fact of a trial.

2. The utility $U(t)$ enters into I^c and M in the next time step.

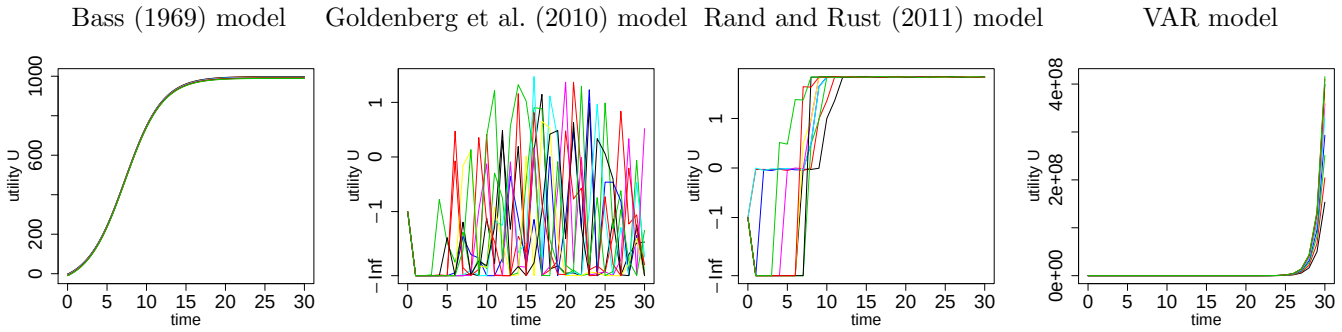


Figure 6: Utility. Here, the four models show very different behaviour. In our representation of the Bass (1969) model the utility U equals the number of trials N, see Figure 4. The Goldenberg et al. (2010) model leads eratic paths for the default parameter values. In the Rand and Rust (2011) model the curves are nearly monotonic and level out. In the VAR model the values are rapidly increasing. In the Bass (1969) model the curves of first ten individuals start at -1 to -10, what is hard to distinguish from 0 so that only one curve seems to be present.

Technical note: In the gui, the temporal development of the characteric for `show.n.indiv = 10` individuals is depicted. If only one set is given, the graphs are shown in a wide range of colors. In case of several sets, the individual courses are shown in the color of the set.

3.6.4 Cumulative social influence I^c

The cumulative social influence is a function of the number of trials $N(t)$ made so far. Generally, one might assume that I^c is an increasing function of N. A function I^c that decreases for small N signifies aversion against a new product. If it decreases for large N it signifies aversion against an established product.

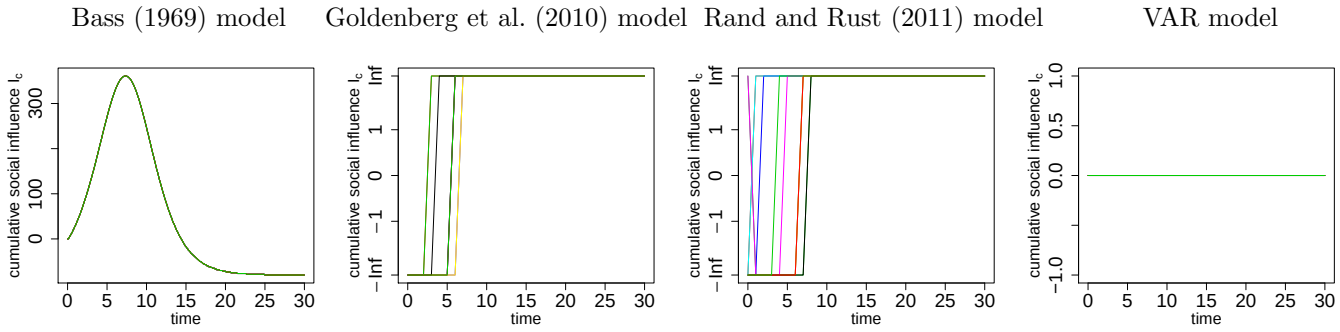


Figure 7: Cumulative social influence. In the Bass (1969) model it argued that this quantity must become negative, particularly to prevent the cumulative number of trials to tend to infinity. In our representation of the Goldenberg et al. (2010) model and the Rand and Rust (2011) model, the cumulative social influence jumps from $-\infty$ to ∞. The vector autoregressive model (VAR) does not depend on I^c; as a default, I^c is set to 0.

Technical note: In the gui, the temporal development of the characteric for `show.n.indiv = 10` individuals is depicted. If only one set is given, the graphs are shown in a wide range of colors. In case of several sets, the individual courses are shown in the color of the set.

3.6.5 Recent social influence I^r

The recent social influence reflects the effect of word of mouth to an individual. This is modeled mainly by ABMs.

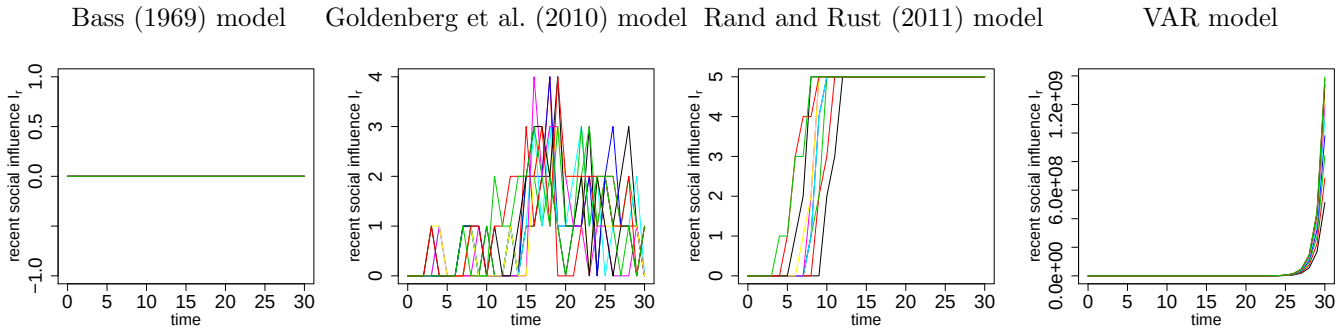


Figure 8: Recent social influence. Here, the four models show very different behaviour. In our representation of the Bass (1969) model the recent social influence I^r is not included. Hence the value of I^r is set to 0 by default. The Goldenberg et al. (2010) model leads to non-monotonic paths except for values a and b close to 1. In the Rand and Rust (2011) model the curves are increasing and level out, while in the VAR model the values are rapidly increasing.

Technical note: In the gui, the temporal development of the characteric for `show.n.indiv = 10` individuals is depicted. If only one set is given, the graphs are shown in a wide range of colors. In case of several sets, the individual courses are shown in the color of the set.

3.6.6 Private Utility U^p

The private utility U^p is considered as influenceable by external factors such as mass media. Hence, it is natural to consider U^p as a random variable with time dependent mean.

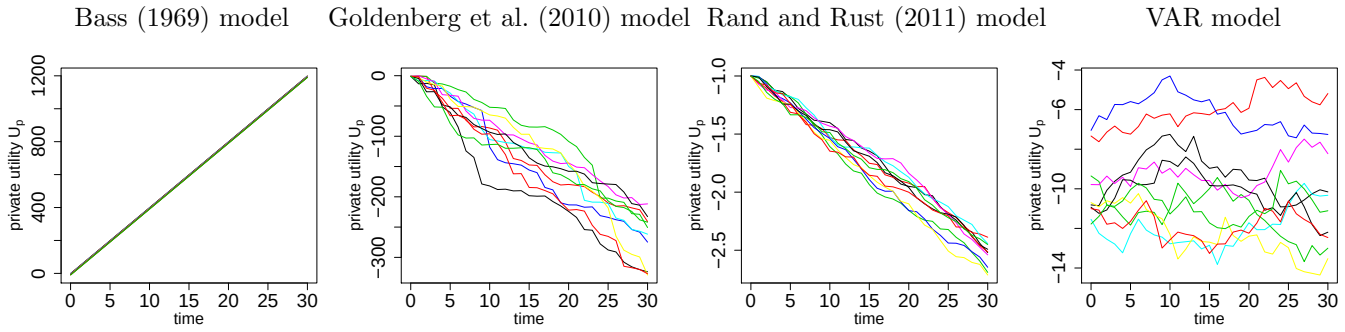


Figure 9: Private Utility. Here, the models show three different kinds of behaviour. In our representation of the Bass (1969) model the private utility U^p is linearly increasing, whilst for the the models of Goldenberg et al. (2010) and Rand and Rust (2011), the curves are essentially linearly decreasing. In the VAR model the curves seem to fluctuate only.

Technical note: In the gui, the temporal development of the characteric for `show.n.indiv = 10` individuals is depicted. If only one set is given, the graphs are shown in a wide range of colors. In case of several sets, the individual courses are shown in the color of the set.

3.6.7 Difference of the private utility between two time steps ΔU^p

The increment ΔU^p gives a clearer picture about the jumps of the private utility U^p between subsequent instances of time.

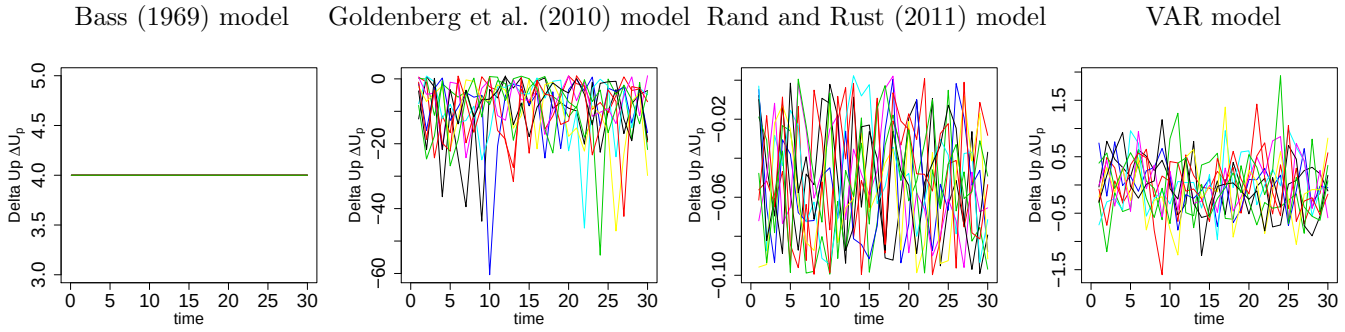


Figure 10: Difference of the private utility between two time steps. These curves are essentially the derivatives of the private utility U^p shown in Figure 9. The two figures in the middle show that the Goldenberg et al. (2010) model and Rand and Rust (2011) model are based on two different kinds of distribution.

Technical note: In the gui, the temporal development of the characteric for `show.n.indiv = 10` individuals is depicted. If only one set is given, the graphs are shown in a wide range of colors. In case of several sets, the individual courses are shown in the color of the set.

3.6.8 Memory effect M

Models based on differential or difference equations obviously include a memory effect. In our representation of ABMs, memory effects appear as well and play an important role of control.

Technical note: In the gui, the temporal development of the characteric for `show.n.indiv = 10` individuals is depicted. If only one set is given, the graphs are shown in a wide range of colors. In case of several sets, the individual courses are shown in the color of the set.

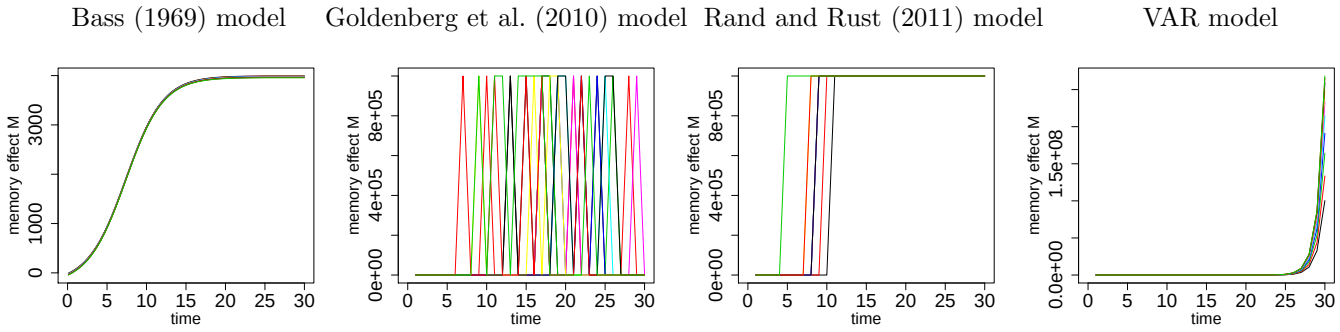


Figure 11: Memory effect. Since the memory effect is a function of the utility U, the four models show very different behaviour as the utility U does. Being based on a differential equation, the memory effect M of the Bass (1969) model equals $N(t-1)$, so that the curves here and in figure 4 are essentially the same. In the Goldenberg et al. (2010) model, the value of M jumps between 0 and some very high value; the Rand and Rust (2011) model jumps only once. Since the VAR model is based on a difference equation, the curves here and in figure 4 are essentially the same.

3.6.9 Spatial distribution of adopters

Most models have an underlying spatial definition, so that the spatial spreading of the adopters can be made visible.

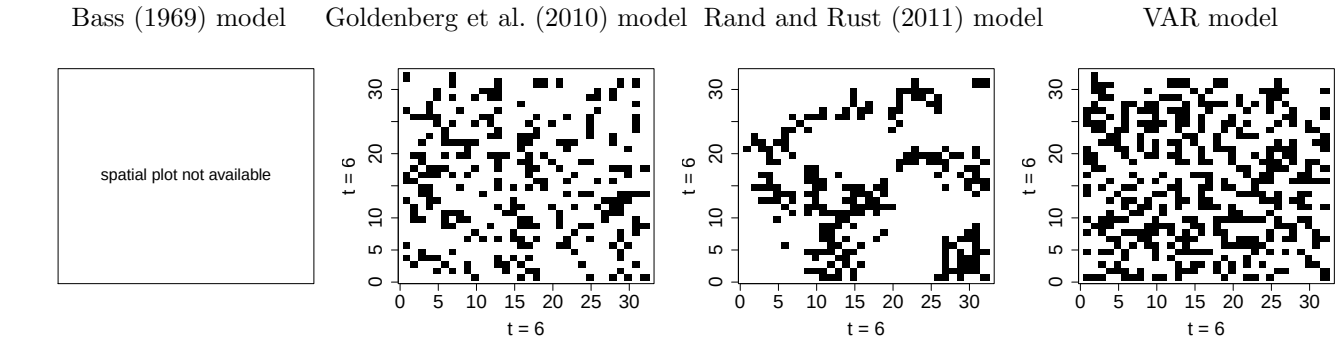


Figure 12: Spatial distribution of adopters. The individuals are assumed to be placed on a square grid. In case the coordinates `coord`, see section 5, are not given, such as in the Bass (1969) model, the spatial distribution of the adopters is not defined and no image is shown.

Technical note: In the gui, the time point for which the spatial pattern is shown is determined by the button "relative.instance", a value between 0 and 1, where 0 and 1 refer to the argument `Tstart` and `Tend`, respectively, of the function `adoption`. The button "spatial" is the only tick box that can trigger more than one graphic, namely as many graphics as there are sets.

3.7 Colors

The curves for the trials and cumulative trials are plotted in the stong colors 'black', 'darkred', 'darkblue','orange', 'forestgreen', according to their set number. The set colors build also the background color for the buttons [d]elete und [1]st once the parameters have being [cpy]-ed. The light colors 'grey', 'red', 'lightblue', 'yellow', 'lightgreen' are used respectively for the quantiles.
For all the other graphs the colors used depends on the number of sets. If the number of set is 1, some nice 7 colors are used for a bundle of curves. If 2 or more sets are present, the curves take the strong colors of the sets.

Data are given in 'pink' in graphics for trials and cumulative trials, the fit to the data is given in 'purple', If a theoretical curve is known, it is plotted in 'orange'.
The sliders are given in different grey scales. In general, the more important a parameter seems to be, the darker the title of the slider isdrawn. Sliders that cannot be used are drawn only in a very light, nearly unreadable, grey color.

4 Arguments of `adoption`

Here an commented overview over the parameters are given. For technical details and a comprehensive list, see the manual pages of the package and the package `RandomFieldsUtils`.

The definition of the function `adoption` is

```
adoption(user = NULL,
         Tend=30,
         quantiles = c(0.25, 0.75),
         included.models =c("Bass (1969)",
                            "Modified Bass",
                            "Goldenberg et. al (2010)",
                            "Generalized Goldenberg",
                            "Rand & Rust (2011)",
                            "Autoregressive (VAR)"),
         dt = NULL,
         data = NULL,
         cumdata = NULL,
         ...)
```

The simplest arguments are `Tend` and `quantiles`. `Tend` gives the endpoint of the simulation. (The simulation starts at time `Tstart` which is 1 by default.) If `quantiles` is given (i.e. not `NULL`), in some of the plots, quantiles are depicted, additionally to the mean curves.

The argument `user` has a threefold meaning:

1. called with the output of the last session, `adoption` restores the former session except for the random seed of the simulations. A sample code is the following:

   ```
   a <- adoption()
   b <- adoption(a)
   ```

2. called with a filename, `adoption` restores the session saved by the button `pdf & store` in the gui.

3. called with a list or a list of lists, `adoption` assumes that the user passes new model definition(s) which are then displayed as additional models (in AREA 4 of the gui, see Figure 1).

   The following code takes the definition of the Bass (1969) model and defines ΔU^p as a waving function leading to a waving behaviour of the trials.

   ```
   a <- adoption(gui=FALSE, printlevel=0)    ## get all model definitions
   my_model <- a[["Bass (1969)"]] ## get the Bass model definition
   my_model $ Up <- function(param, dt, m, nT, rep)
   ```

```
    rep(2 * dt * m * Value("Ic.param", 1) * sin((1:nT) * dt * 0.3),
        rep * m)
m <- adoption(list("Bass funny waving" = my_model))
```

The number of default models can be reduced by the argument `included.models`. Here, the model names might be abbreviated as long as the abbreviations are unique. For instance, starting the gui with

```
a <- adoption(included.models=c("Ba", "Mo"))
```

will offer only the Bass (1969) model and its modification.

Data can be passed as an integer valued vector through the arguments `data` or `cumdata`. The instances when the data have been measured are assumed to be gridded. The time span between subsequent measurements is passed through the argument `dt`. Note that `cumdata` accepts missing values (`NA`).

Important: If the system supports OMP (OpenMP, Open Multi-Processing), the speed of the program can be accelerated by the argument `cores`.

4.1 Options passed by "..." (advanced)

The following options can be set within `adoption()` or globally by `RFoptions`. There are three different kinds of options.

4.1.1 Optional arguments of general purpose

Two general parameters are used in `adoption`,

- `cores` which sets the number of cores used in parallel
- `printlevel` a higher number results in a higher entertainment on the xterm of Linux machines with additional informations.

Technical note: The list of general parameters can be obtained by `RFoptions()$basic`.

4.1.2 Optional arguments for fitting to the data

The following arguments control the fitting to the data:

- `fit_m` Default is that integer values are never fitted by `adoption`. An exception builds the market size m which is never known in real life and should be estimated, too, If `fit_m = TRUE`, then also the market size is fitted at a rather high price of time.

  Technical note: The fitting procedure for the market size m is seen to be in a rudimentary stage and might need further work of research.

- `fit_repetitions` The minimum number of simulations used to calculate $N(t)$ from the model. If the value in gui for repetitions is higher than `fit_repetitions`, this value is taken for the calculation of $N(t)$.
- `pgtol`, `factr` see `optim` for their meaning.

- `tracefit` If the value is 1 or 2, more entertainment of the user is given whilst fitting.
- `max_increasing` If the number of non-decreasing iterations equal `max_increasing` the complete fitting algorithm ends.

Technical note: The list of fitting parameters is a subset of `RFoptions()$adoption`.

4.1.3 Other optional arguments

- The appearance of the gui can be adjusted, for instance, the placement of certain buttons by `button2right`, the `fontsize`, the model number to `startwith`, the starting time point `Tstart` and how often simulations are updated when the entry box is used (`simuOnTheFly`).
- `wait` This argument influences the return value of the gui, see Section 3.2.6, but also the amount of time the CPU is idling. The lower the value the more time is spend in doing nothing productive.
- `ymax` When plotting a series of plots of the number of trials ΔN, there is a trade off between changing the scale all the time (leading to eye irritation) and imprecision (keeping the same scale for a too long time). This trade off can be modified by the argument `ymax`.
- `gui` The option `gui=FALSE` runs `adoption` in an non-interactive mode where figures are plotted and saved before `adoption` is left. If additionally `printlevel=0` then only the definitions of the default models are returned; i.e., this option returns templates for one's own model definition or modification.

Technical note: The above list of arguments is a subset of `RFoptions()$adoption`.

5 Model definitions

Beyond the six predefined models in `adoption`, the user may define one's own models. In this rather technical section we give details about coding new models. If you are happy with the six predefined models, you might skip this section.
To give an idea how such a model definition might look like, we give examplarily the definition of the Bass model as defined in the file `adoption.R` of the package:

```
Bass69 <- list( m = 1000,
    repetitions=1L,
    repetitions.max=1,
    dt = 0.1,
    relative.instance = 0,
    max.relative.instance = 0,

    SOCIAL = c(1, 3, 3),
    Ic.start = NULL,
    Ic =  function(param, Nt, m, ...) {
      rep(- 4 * Nt * (param[1] - param[2] + param[2] * Nt / m), each=m)
    },
    Ic.param = c("innovation p" =  0.02,
                 "imitation q" = 0.4),
    Ic.param.min = c(0, 0),
    Ic.param.max = c(1, 1),
```

```
  Utrafo = function(U, ...) 4 * U,

  PRIVATE = rep(5, 3),
  Up.start = function(param, m, rep) base::rep(-(1:m), rep),
  CrossReferences = function() TRUE
  Up = function(param, dt, m, nT, rep)
    rep(2 * dt * m * Value("Ic.param", 1), nT * rep * m),
  Uthreshold = 0,
  Uthreshold.min = 0,
  Uthreshold.max = 0,

  "MAX/PLUS OPERATORS" = rep(5, 3),
  alpha = c("alpha_1"=1, "alpha_2"=1),
  alpha.min = c(1, 1),
  alpha.max = c(1, 1),
  beta = c("beta_1"=0.5, "beta_2"=0.5),
  beta.min = c(0.5, 0.5),
  beta.max = c(0.5, 0.5),
  gamma = c("gamma_1"=0.5, "gamma_2"=0.5),
  gamma.min = c(0.5, 0.5),
  gamma.max = c(0.5, 0.5),

  theor.dN = function(Ic.param, t, m) {
    p <- Ic.param[1]
    q <- Ic.param[2]
    E <- exp((q+p) * (t + log(p / m)/(p+q)))
    Nt <- (m * E - p) / (E + q/ m)
    (m - Nt) * (p + q * Nt / m)
  }
)
```

In such a model definition the ordering of the elements of the list is irrelevant, since the elements are reordering according to an internal list which includes also some default values and the titles within the gui:

```
list("m",
     "repetitions",
     "dt",
     "relative.instance",
     SOCIAL = c(1, 1, 3),
     "Ic.param",
     "weight.param",
     "coord.param" = FALSE,
     PRIVATE = c(1, 1, 3),
     "Up.start.param",
     "Up.param",
     "Uthreshold",
     "MAX/PLUS OPERATORS" = c(2, 2, 2),
     "alpha",
     "beta",
     "gamma")
```

This order gives the order presented in the gui and cannot be changed. The model definition of the user may leave out some parameters, which are considered as "(unused)" then. The user must

include additionally several function definitions that are described below. The above list, which is taken from the internal R code in `adoption.R`, has the following interpretations: if a parameter is `FALSE`, here `coord.param`, then this parameter is invisible in the gui and hence can never be changed by the gui user. Titles are followed by a vector of integers. These values give the importance of the subsequent parameters (1 for high and 4 for low; 5 for unused or fixed parameters). The parameters before the first title ("SOCIAL") are the global parameters. All the parameters can be vectors except for the global ones, which are always scalars. The parameters are given as vectors of named default elements, e.g. `Ic.param = c("innovation p" = 0.02, "imitation q" = 0.4)` in the Bass model. These names are printed in the gui. Additionally to the default values, values for the minimum and the maximum can be given, e.g. `Ic.param.min = c(0, 0)`. The following functions do not have default definitions, so most of them must be given. Some of them can be `NULL` and that model part is ignored then. The argument `Nt` equals $N(t)$, `m` is the market size, `rep` the number of repetitions, `dt` the time increment, and `nT` is the number of performed time steps. We have

1. `Ic.start = function(Ic.param, m, rep)` : This function has no direct correspondance in the model definition of Anonymous et al. (2019), but is used to create additional random parameters for I^c. The function returns a vector of length `m * rep`, which is used as argument `start` in the function `Ic`. Here, the function will be called with parameter `Ic.param`; the parameter `Ic.start.param` does not exist.

2. `Ic = function(Ic.param, Nt, m, start)` : This function corresponds to I^c in Anonymous et al. (2019). It returns a vector of length `start`. The latter is created by function `Ic.start`.

3. `coord = function(coord.param, m)` : The output of this function is used to define the weights w_{ij} in Anonymous et al. (2019). It creates a matrix of `m` rows. The number of columns equals the dimension of the space and is given by `coord.param[1]`.

4. `weight = function(weight.param, dist)` : This function returns the matrix of weights $(w_{ij})_{i,j=1,\ldots,m}$ in Anonymous et al. (2019). The function returns a matrix of the same size as `dist`. Here, `dist` is a matrix of distances calculated as Euclidean distances from the coordinates given by `coord`. Note that there is no check whether diagonal values are zero as required in the theoretical approach by Anonymous et al. (2019).

5. `Utrafo = function(U, Uthreshold, m)` : This is the function f in the model defintion in Anonymous et al. (2019). The function returns a vector of the same length as `U`. The argument `Uthreshold` equals the parameter θ in the model definition of Anonymous et al. (2019) and is 0 by default.

6. `Up.start = function(Up.start.param, m, rep)` : This function delivers the starting values $U^p(0)$ of the private utility U^p. The function returns a vector of length `m * rep`.

7. `Up = function(Up.param, dt, m, nT, rep)` : This function defines either U^p or ΔU^p in Anonymous et al. (2019). If the function returns a vector of length `nT * rep * m`, then the result is considered as values ΔU^p. If the function returns `NULL`, then U^p is considered as being a Brownian motion with variance given by `Up.param[1]^2`. (If it returns a `RMmodel` of the R package `RandomFields`, then this model defines U^p – this will be implemented in future.)

8. `theor.dN = function(Ic.param, t, m)` : This function gives the theoretical number of adoptors. It returns a vector of size `t`. Here, `t` is a vector of time instances.

9. `CrossReferences = function() TRUE` : This technical function does not have an immediate correspondance in the theoretical framework. `CrossReferences` is either not given or looks exactly like this. If given, functions may use the parameters of other functions as well. For instance, in the Bass model, the parameter p appears in both I^c and U^p so that the definition of U^p is the following:

```
Up = function(Up.param, dt, m, nT, rep)
     rep(2 * dt * m * Value("Ic.param", 1), nT * rep * m)
```

Here, `Value(par, i)` returns the ith component of the parameter `par`, where `par` must be given as a character. Note that if `CrossReferences` is given, the simulation time might increase considerably as the system has to redo all the simulation parts for any change of the parameter values in the gui, as the system does not know which parts are affected by cross-referencing. Second note is that the R environment of the model definition is changed so that no variables of the R session (in particular no global variables of R) may be used in the model definition. (E.g., the use of the variable `Goldenberg_C <- 1e6` in the internal definition of the Goldenberg et al. (2010) model in the R function `adoption` would not be possible.)

One's own models are passed to `adoption` through the argument `user` as a lost of a list of lists, see also Section 4.

6 Further implementation details

6.1 Evaluation of the operator

Note that we have

$$x \ \vee_\alpha \ y = y + \xi_\alpha(x, y)(x - y).$$

with $\xi_\alpha(x, y) = \alpha_1 \mathbf{1}_{x>y} + \alpha_2 \mathbf{1}_{x<y}$. Since the value of x or y could be $\pm\infty$ and $x = y$ implies $\xi_\alpha = 0$, we define $0 \cdot \infty = 0$. This is the form the operator is implemented in the C code of the gui.

6.2 Simulation

The code is a mixure of R code, where the model definition is given, and C code, where the temporal development of the model is calculated.
For efficiency reasons of code of R (R Development Core Team, 2019) all the initial and boundary values, e.g. U^p are simulated at once at the very beginning. A slider or entry change triggers only a partial recalculation of random variables and initial and boundary values. The temporal development of the model must always be recalculated.
The calculation of I^r is very time consuming so that three cases are distinguished: no weight matrix is given or the weight matrix consists of zeros only, weight matrix is sparse (containing at least 80 % of zero), and matrix is full.
Since R tends to copy values and since in the R language only functions are defined, and not procedures, special functions for calculation the distances between individuals in the Goldenberg et al. (2010) and in the VAR model are defined that can modify globally defined variables without any copying. These functions may be used only with great care.
Creating random variables is time expensive. So, the simulated random variables are reused until the button `new simulation` is pressed.

6.3 Fitting

We fit the model to the data by least squares so that the RSS between the empirical cumulative distribution function and the distribution function calculated from the model (by enough simulations from the model) is the decisive criterion.
Most parameters influence heavily the location of the density function so that the parameter have a collinear behavior and `optim` is not able to estimate the parameters correctly, in general. We define the location of a density function as its median. The fitting consists of several steps.

1. Global search versus local search for the optimum
   We determine the medians for a grid of values of the parameters that roughly cover the space of the parameters (3 points in each direction so in total 3^k points where k is the number of parameters are to be estimated). We also determine the RSS for all the grid points in the parameter space. If all RSS are greater than the RSS of the user's suggestion, this global grid is discarded and replaced by a grid with the user's suggestion (i.e. the parameter values given by the current slider positions in the gui) in the middle and a small lattice spacing around.

2. Determination of the collinearity correction
   Instead of performing the fitting with the genuine model parameters, The fitting is performed with a set of dummy parameters that are (locally) algrebraically independent. The relation between the genuine model parameters and the dummy parameters are assumed to be a (locally) linear relation. This linear relation is calculated from the above parameter grid.

   So, we fit a hyperplane by LSQ to the values of the medians of the grid in the parameter space. Now, we determine the closest point (by the square of the Euclidean distance) to the starting point suggested by the user under the constraint that the solution is on the intersection of the hyperplane and the median of the data (considered as a hyperplane of constant value). If one of the obtained values is outside the range of the parameters, a genuine quadratic program is solved by solve.QP of the R package `quadprog` (Berwin and Weingessel, 2013). Still it can happend that `quadprog` fails (for theoretical reasons). Then the absolute value of the leading coefficient of the hyperplane (i.e. the coefficient with the highest absolute value except the intercept) is increased. And the fitting by `quadprog` is redone. This part might be even repeated for a few times.

3. Optimisation of the dummy parameters
   The optimizer `optim` will optimize the above mentioned dummy parameters. Of course these dummy parameters have first to be transformed to the genuine model parameters before the RSS is calculated. Boundaries of the parameter values have to taken into account.

4. Complete fit
   Not surprisingly, the above algorithm (which is run in gui through the button `fit 1 step`) suffers also from the collinearity problems when our (locally optimized) correction for the collinearity is not good anymore. To circumvent this problem we iterate the above procedure with an updated starting value until the fit looks satisfying. Since the local correction causes a waving behaviour of the RSS, the stopping criterion is rather crude. The algorithms stopps if no improvement is obtained for a (short) run of iterations, determined by the optional argument `max_iterations`.

5. Fitting of the market size
   Here, the base is a simple RSS optimization of the parameters for a fixed market size m
   A correction for the collinearity as described above is currently not performed. The lowest

possible value for m is given by the empirical data by the maximum value of N. Starting with this minimal value a search for the optimal m is performed by first doubling the value of m until an improvement is possible any more. Then the interval is determined where the optimal value for m must be. Then a standard search in the interval is performed. In a last step, the model parameters with collinearity correction and fixed market size might be estimated.

Technical note: (i) See Section 4.1.2 for argument controlling the fitting procedure.
(ii) Not surprisingly from description, the fitting result will depend on the starting values given by user (through the current parameter values in the gui). Since the starting value given by the user is compared with further potential starting values, we firstly reduce the dependence on the user's values and secondly obtain a complex dependency of the result on the user's values. This dependency is further blurred and reduced by the iteration steps in the complete fit. Overall, the dependency on the user's values seem to be acceptable low. (iii) In Anonymous et al. (2019), the presented results are based, whenever applicable, on the same starting values to ensure maximum comparability among the results. Furthermore, an extensive testing in Anonymous et al. (2019) indicated that the variation of the starting values may not be able to improve the model fit by more than one percent.

7 Installation details

Here some details and potential difficulties are listed when installing the package `adoption`:

- `adoption` is based on the package `RandomFieldsUtils` (Schlather et al., 2019) which itself needs a fortran compiler, e.g. `gfortran`
- `tcltk` and `tk` must be installed on the OS system, additionally to the R package `tcltk2`
- newer MAC OS systems need additional installation of `XQuartz` on the system. This program must be running when adoption is used.
- On some systems `openmp` does not work although `openmp` is recognized. Then versions of `RandomFieldsUtils` and `adoption` with no `$(SHLIB_OPENMP_CXXFLAGS)` flags in `/src/Makevars` must be installed.

8 Acknowledgement

The author is grateful to Anonymous and Anonymous for hints and comments, which improved considerably the presentation of this technical report.

References

Anonymous, Anonymous, and Anonymous. Toward a generalized adoption modeling framework. 2019. In preparation.

F.M. Bass. A new product growth for model consumer durables. *Management Science*, 15(5): 215–227, 1969.

M. Beenstock and D. Felsenstein. Spatial vector autoregressions. *Spatial Economic Analysis*, 2(2): 167–196, 2007.

A. Berwin and A. Weingessel. *quadprog: Functions to solve Quadratic Programming Problems.*, 2013. URL `https://CRAN.R-project.org/package=quadprog`. R package version 1.5-5.

W.A. Brock and S.N. Durlauf. Discrete choice with social interactions. *Review of Economic Studies*, 68(2):235–260, 2001.

M.G. Dekimpe and D.M. Hanssens. Sustained spending and persistent response: A new look at long-term marketing profitability. *Journal of Marketing Research*, pages 397–412, 1999.

S.N. Durlauf and Y.M. Ioannides. Social interactions. *Annual Review of Economics*, 2(1):451–478, 2010.

J. Goldenberg, B. Libai, S. Moldovan, and E. Muller. The npv of bad news. *International Journal of Research in Marketing*, 24(3):186–200, 2007.

J. Goldenberg, B. Libai, and E. Muller. The chilling effects of network externalities. *International Journal of Research in Marketing*, 27(1):4–15, 2010.

S. Johansen. Estimation and hypothesis testing of cointegration vectors in gaussian vector autoregressive models. *Econometrica*, pages 1551–1580, 1991.

V. Mahajan, E. Muller, and F.M. Bass. Diffusion of new products: Empirical generalizations and managerial uses. *Marketing Science*, 14(3):G79–G88, 1995.

R. Peres, E. Muller, and V. Mahajan. Innovation diffusion and new product growth models: A critical review and research directions. *International Journal of Research in Marketing*, 27(2): 91–106, 2010.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2019. URL `http://www.R-project.org`. ISBN 3-900051-07-0.

W. Rand and R.T. Rust. Agent-based modeling in marketing: Guidelines for rigor. *International Journal of Research in Marketing*, 28(3):181–193, 2011.

H. Risselada, P.C. Verhoef, and T.H.A. Bijmolt. Dynamic effects of social influence and direct marketing on the adoption of high-technology products. *Journal of Marketing*, 78(2):52–68, 2014.

M. Schlather, R. Furrer, and M. Kroll. *RandomFieldsUtils: Utilities for the Simulation and Analysis of Random Fields*, 2019. URL `http://CRAN.R-project.org/package=RandomFieldsUtils`. R package version 0.4.2.

S.M. Tanny and N.A. Derzko. Innovators and imitators in innovation diffusion modelling. *Journal of Forecasting*, 7(4):225–234, 1988.

C. Van den Bulte and G.L. Lilien. Medical innovation revisited: Social contagion versus marketing effort. *American Journal of Sociology*, 106(5):1409–1435, 2001.