The Routes & Rumours Model

Martin Hinsch, Jakub Bijak

October 18, 2021

model version 0.1.0, document version 1.0

Contents

1 Introduction

We model the formation of migration routes and how it is affected by the availability and exchange of information. In our model agents attempt to traverse a - for them - mostly unknown landscape, having to rely on either local exploration or communication with other agents to find the best path across.

This document describes the full version of the model, including in particular risk and resources. Both of these aspects of the model remain unused in some of the publications that are based on it. Since they can be switched off completely (using model parameters) only the active parts of the model are documented in publications outside of this document.

We deliberately restrict this documentation to a technical description of the working of the model itself. This is very much a qualitative, not a predictive model (and should not be treated as such). Consequently purpose, parameterisation and analysis depend to a large degree on the context in which the model is used (Hinsch & Bijak, in press). For a number of different applications of versions of the model see Bijak (in press).

Scope

Any documentation of a computational model has to make a trade-off between accuracy and accessibility. A fully accurate and complete document will necessarily be as complex as the source code it describes and therefore at best only slightly easier to read.

Documentation therefore has to simplify. This can happen in two ways. One, by leaving out "obvious" details, that is by not describing technical implementation details that can be expected to be inferred by a reasonably competent reader. For example when we write that all cities that are closer than a given distance will be connected by a link we implicitly assume that the reader understands that this means that - if they were to replicate the model - they needed to implement a loop over all combinations of cities, calculate their Euclidean distance, test if that distance is lower than the given threshold, then, if so, create a new link object, initialise it accordingly, add the link to both cities, and so on. This does not necessarily mean that there is only one possible implementation of the mechanism we describe und the implicit assumption is that any implementation will produce the same result.

The second way to "abbreviate" is to not specify semantic details of the model itself that we assume will not affect its behaviour. We do for example not specify which random number generator we are using. Similarly we do not describe for each calculation the exact steps we are performing and their order. In both cases different interpretations will lead to different numerical results (due to different algorithms and different rounding errors, respectively), however, we assume that these are irrelevant for the behaviour of the model.

Ideally, this documentation should therefore be the minimum of information that would be required to recreate the model (without having access to the source code) while producing sufficiently similar results that the same conclusions would be reached as based on the original. In practice, even with best efforts, documentation of software is very rarely fully complete and accurate, ultimately making the source code the authoritative reference.

This leads us to the other purpose of documenting a model, which is to make the source code more accessible. Reading and understanding somebody else's program code is generally not easy, but knowing beforehand what the code is supposed to do can help substantially. To reduce the hurdle even further we have attempted to use a notation, layed out in the next section, that makes connections between description and implementation as obvious as possible.

A note on notation

Most of the time mathematical expressions are more concise and more accessible to a broader audience than program code. We therefore defaulted to using mathematical formulae as a formalism for expressing calculations in the model. On the other hand, the large number of variables and parameters in the model makes the mathematical convention to use single letter variable names unviable. Since we also wanted to make it as easy as possible to find and understand the source code belonging to a part of the documentation, we therefore attempted to use names exactly as they occur in the code wherever possible. To further clarify the relation between the text and the source code we use the following conventions.

Variable, type and function names that are copied verbatim from the source code are shown in sans serif font in mathematical formulae as well as in the text. Where the name is also a model parameter we additionally set it in **bold face** in the text (a full list of parameters also appears in the Appendix). Names that are used only in the documentation (for clarity or as abbreviations) are shown in *italics* in the text as well as in mathematical expressions, as are helper variables. Important model concepts are set in the standard font but underlined when mentioned for the first time or for emphasis. For example, there is a difference if we talk about the concept of an agent or the type Agent in the program code. Similarly we have to distinguish between the concept of an obstacle in the simulated landscape and the model parameter **obstacle**.

Source files or modules that contain the implementation of a given section are referenced in margin notes next to the section headings.

2 Entities

`base/entities.jl`

Entities directly represented in the model are agents, cities and transport links.

2.1 Agents

At any time agents are either present at a city or a transport link or they have arrived at an exit.

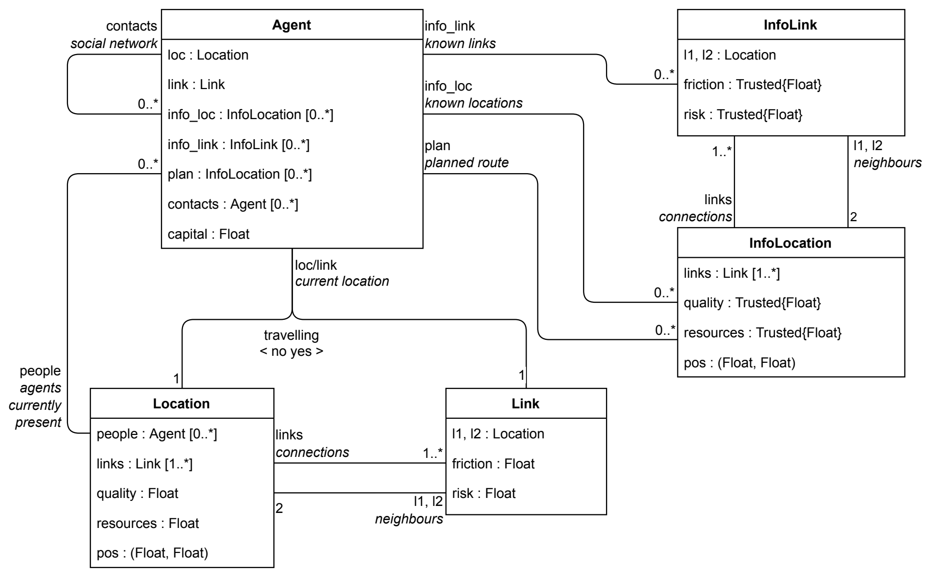

Figure 1: Types of entities in the model and their relationships.

Contacts Each agent has a list of other agents that it stands in contact and can exchange information with (see Section 5.2).

Knowledge Each agent has a potentially incomplete and inaccurate set of knowledge items concerning the world. Each item describes the properties and topology (i.e. connections to other entities) of a city or a transport link.

Plan Agents maintain a list of cities they plan to move to on their way to the exit.

Risk Each agent has properties slope and int that determine slope and intercept, respectively of its reaction to risk.

2.2 Cities

Cities are located at a specific position on the map and differ in quality and resources. Cities are connected amongst each other by randomly generated transport links (see Section 3). There are a number of designated entry and exit cities at which agents arrive and leave the world, respectively.

2.3 Links

Links always connect two cities. In the basic version of the model the only property of links is friction which represents a combination of distance and difficulty of travel. In the risk version of the model links also have risk associated which is used to calculate the probability for an agent to die when crossing that link.

3 Setup

base/setup.jl

3.1 Map

Before the start of the simulation a map of cities and links is generated and their property values assigned. To generate the default topology we use a random geometric graph: **n_cities** cities are placed at random positions, then cities that are closer than a given threshold **link_thresh** are connected with a transport link. In addition we place a fixed number of **n_entry** (or **n_exit** respectively) entry and exit cities at the respective edge of the map (either at random positions or spaced out regularly - parameters **regular_entries**, **regular_exits**). Entries and exits are connected with the **n_nearest_entry** (**n_nearest_exit**) nearest “regular” cities. Furthermore, entries (exits) connect to all cities that are closer than **entry_dist** (**exit_dist**) to the entry (exit) edge of the map, but using slow links (see below).

Cities as well as links differ with respect to whether agents can have knowledge about them before they start their journey. Belonging to either category is determined by **p_unknown_city** and **p_unknown_link**, respectively.

Links

Links come in two types (typ). Links that connect randomly generated cities as well as those that are generated in a fixed number per entry and exit, respectively, are fast (i.e. have low friction). Links that are generated to connect entries and exits to all cities within a certain range of the respective edge of the map are slow (i.e. have high friction).

The parameter **dist_scale** determines how the length of a link (distance) scales to friction for both types. Given random value $\nu \sim \mathcal{U}_{[0,\text{frict\_range}]}$ (parameter **frict_range**), the friction of links is calculated as $\text{friction} = \text{distance} \cdot \text{dist\_scale}[\text{typ}] \cdot (1 + \nu)$.

Links’ risk value is set to **risk_normal** unless they intersect a rectangle whose coordinates are given by **obstacle** in which case risk is set to **risk_high**.

Cities

Cities’ quality and resources are both set to random values out of $\mathcal{U}_{[0,1]}$. The values of both properties for entries and exits are set by the parameters **qual_entry**, **res_entry**, **qual_exit** and **res_exit**.

3.2 Agents

base/model.jl

At the beginning of the simulation no agents are present. Agents are created by the create agents process and added to a randomly selected entry point.

Risk

Agent properties int and slope are determined by drawing from bivariate normal distributions with configurable variance-covariance matrix and mean values:

$$\Sigma = \begin{bmatrix} \text{risk\_sd\_i} & \text{risk\_cov\_i\_s} \\ \text{risk\_cov\_i\_s} & \text{risk\_sd\_s} \end{bmatrix}$$
$$\mu = \begin{pmatrix} \text{risk\_i} \\ \text{risk\_s} \end{pmatrix}$$
$$\text{int}, \text{slope} \sim \mathcal{N}(\mu, \Sigma)$$

Contacts

The number of initial contacts an agent has is determined as

$$n = \min(\#agents/10, \text{n\_ini\_contacts}).$$

Contacts are picked at random from the entire population (including arrived agents, but excluding dead ones).

Knowledge

Newly created agents explore their current location with speed **speed_expl_stay** and indirect discovery (see Section 5.3). They know exit cities with probability **p_know_target** and regular cities/links (out of those that can be known to agents before setting off, see map creation above) with a probability **p_know_city** and **p_know_link**, respectively. They explore each known city using exploration speed **speed_expl_ini** and no indirect discovery. Finally they explore known links if they already know one of the end points (also with speed **speed_expl_init**).

4 Processes

`base/model_agents.jl`

The model is implemented as an event-based simulation. That means that updates to the model state do not happen in discrete time steps but instead as a result of asynchronous events (see Table 1). All events in the model are assumed to be Poisson processes in continuous time (Gillespie, 1976).

With the exception of the creation of new agents all changes of model state are the result of the actions of agents which are also the only entitites that change state. Which actions an agent can perform and their rates of occurence depends on its state, in particular on whether it is currently travelling on a link or staying in a city. Agents can exchange information with other agents either in their contact list or present at the same location as they are. They can travel along transport links and collect information on their current and neighbouring cities.

4.1 Events

`base/processes.jl`

Most processes are changes of state in single agents. Whether they can apply is usually dependent on whether an agent is travelling (= present at a transport link) or not (= present at a city). It is important to note that *every agent* in the population at any time that it fulfills the respective conditions can potentially experience the event in question.

Table 1: Events, their effects and rates.

event	*entity*	*condition*	*rate depends on*	*affects*
create agent	world	always	constant	number of agents
plan	agent	at city	up to date inf.	plan
explore	agent	at city	constant	information
add contact	agent	at city	#local agents	contacts
forget contact	agent	always	#contacts	contacts
exchange information	agent	always	#contacts	information
depart	agent	at city	constant	location
arrive	agent	travelling	friction	location
die	agent	travelling	friction	number of agents
costs	agent	at city	constant	capital

Create agents

Agents are created with rate **rate_dep**. They enter the world at a randomly selected departure location. For details see Section 3.

Plan

During planning an agent decides where to go next, depending on its current knowledge. Agents plan with a rate **rate_plan** if their knowledge about the world has changed and 0 otherwise. During planning agents decide on where to go next, either based on local optimization or on finding the best route to an exit (see section 5.4 for details).

Explore

An exploring agent gains new knowledge about closeby cities and links. Agents that do not travel explore their current location at a fixed rate **rate_explore_stay**, using speed **speed_expl_stay** and employing indirect discovery (see Section 5.3 for details).

Add contact

An agent adds a new agent that is currently situated in the same city to its list of contacts. Agents meet new contacts at a rate *#other agents* · p_keep_contact if their current number of contacts is below **n_contacts_max**. A randomly selected agent from the focal agent's current location is added to the contact list. If the other agent has space in its contact list as well the contact becomes mutual. Then both agents exchange information (see Section 5.2).

Forget contact

Agents lose contacts at a rate of $\#contacts \cdot \text{p\_drop\_contact}$. Forgotten contacts are only removed unilaterally, that is the other agents will keep its contact.

Exchange information

An agent communicates with one of its contacts and exchanges information about the world topology, i.e. the existence and connectedness of cities and links, as well as their properties. Agents communicate at rate $\#contacts \cdot \text{p\_info\_contacts}$. See Section 5.2 for details.

Depart from a location

An agent departs from its current location and starts travelling to the next location in its plan. Departure rate is 1 unless the agents capital is lower than **save_thresh** and its expected income is greater than **save_income**.

On departure an agent leaves its current location and is now located on the link to the next location. It immediately explores the link with speed **speed_expl_move** (see Section 5.3). It also loses $\text{costs\_move} \cdot \text{friction}$ resources which are substracted from its capital.

Arrive at a destination

A travelling agent arrives at its destination. Agents arrive at a rate $\text{move\_rate} + \text{move\_speed}/\text{friction}$. Arrived agents change their location to the destination.

Die

Agents travelling on links can die. For a link with arrival rate r_a the mortality rate of agents travelling on that link is $r_a \frac{\text{risk}}{1-\text{risk}}$. Therefore the probability for an agent to die before arrival is just the risk value associated with the link. Deaths can affect the perceived risk of other agents (see Section 5.5 for details).

Income/costs

When staying at a location agents can lose or gain resources. The change in resources happens at a rate **rate_costs_stay** and is calculated as

$$\text{ben\_resources} \cdot \text{resources} - \text{rate\_costs\_stay}$$

(with resources the resource availability at the current location).

5 Information and decisions

Information is a crucial part of the model. Agents have subjective - i.e. potentially wrong and incomplete - information about the world. They obtain that knowledge either by exploration or by communicating with other agents. Nearly all decisions agents make are based on their subjective knowledge.

5.1 Belief Dynamics

Beliefs

Each item of knowledge an agent has - for example the quality of a specific city - is described by a value estimate (called value in the model) and a certainty (trust). That is, an agent has an idea of the numerical value of a given property and how certain it is that the value is correct. For a given agent these numbers change either when the agent explores its environment or when it exchanges information with other agents. When collecting information from the environment the estimate becomes more accurate while the certainty increases. Information exchange is a bit more complicated. Generally speaking the more certain an agent is (i.e. the higher its certainty value) the stronger the effect on the other agent's estimate. At the same time agents with similar beliefs (i.e. similar value estimates) will reinforce each other and their certainty will increase while for very dissimilar beliefs certainty can decrease.

In the following we give a general description of the belief dynamics sub-model. Since some of the expressions are quite long we have - for the sake of readability - kept to mathematical notation in this case. Names of the corresponding variables in the source code are given in parentheses.

We based our information model on the well-known mass action dynamics. To understand the model it is best to imagine that an agent's belief consists of two "substances", certainty and doubt, in proportions t (trust) and $d = 1 - t$, respectively. When two agents interact a "reaction" between their respective belief components takes place, potentially transforming them (see Table 2). Doubt reacting with doubt produces doubt. Certainty of one agent interacting with the other agent's doubt can "convince" the latter, changing parts of its doubt into certainty. Depending on the difference in estimate certainty interacting with certainty can lead to confusion and increased doubt or just change the estimate.

Table 2: Interactions between certainty and doubt of two communicating agents.

interacting with	*doubt*	*certainty*
doubt	doubt	-
certainty	certainty	doubt+certainty

More formally, for an interaction between agents A and B with an estimate $v_.$ (value) we define difference in estimate as

$$\delta_v := \frac{|v_A - v_B|}{v_A + v_B}.$$

Using parameters c_i (**convince**), c_u (**confuse**) and c_e (**convert**) we then calculate the new doubt value d'_A based on the previous values of certainty $t_.$ and doubt $d_.$ as

$$d'_A = d_A d_B + (1 - c_i) d_A t_B + c_u t_A t_B \delta_v.$$

The estimate v_A changes accordingly:

$$v'_A = \frac{t_A d_B v_A + c_i d_A t_B v_B + t_A t_B (1 - c_u \delta_v)((1 - c_e) v_A + c_e v_B)}{1 - d'_A}$$

As we can see, if different opinions do not lead to doubt, i.e. $c_u = 0$, doubt will disappear, i.e. d will approach 0 (as long as $c_i > 0$), and the model reverts to a simple weighted mean [citations for eqv models]:

$$v'_A = (1 - c_e) v_A + c_e v_B$$

Following common programming language convention we will use dot notation to denote the respective properties of an estimate (e.g. risk.trust). In some cases we are interested in the "discounted" estimate which we define as

$$\text{disc}(x) := x.\text{value} \cdot x.\text{trust}.$$

5.2 Information Exchange

In a single communication event, information about each information item known by at least one of the participants is exchanged with probability **p_transfer_info**. Arrived agents generally do not receive information and do not learn about links or cities unknown to them, so if one of the participants has already arrived then items that are unknown to that participant will be skipped. Otherwise information items with default values are added accordingly (for exchanges that take place) so that both participants know the item in question prior to the actual exchange of information.

During the exchange of information, beliefs about all properties of the item in question (i.e. friction and risk for links and quality and resources for cities) are exchanged as described in Belief Dynamics. Depending on parameter values, two additional effects can apply compared to the basic model described earlier:

Errors

Communication can be configured to be noisy. Each agent perceives the value and the trust component of the communicated property with an added error $\epsilon \sim \mathcal{U}_{[-\delta,\delta]}$ (with δ being **error_frict** for the friction value, **error_risk** for the risk value and **error** otherwise).

Success bias

Agents that have arrived can confer increased authority when communicating their experience. Agents interacting with an arrived agent use modified values for **convince** and **convert** (with parameter **weight_arr**):

$$c_i = \text{convince}^{1/\text{weight\_arr}}$$
$$c_o = \text{convert}^{1/\text{weight\_arr}}$$

5.3 Exploration

During exploration agents discover new cities and links and refine their knowledge about those they are already aware of. There are several situations in the model that let agents explore their environment (see Section 4), the basic mechanics are identical, however.

Unless mentioned otherwise the properties of newly discovered links and cities are initialized with "expected" default values **risk_exp**, **res_exp**, etc.

Update

When exploring links or cities an agent always improves its estimate of the respective entities' properties. An agent's belief is updated by calculating the new value v' and trust t' as simple weighted means using a learning speed s (which can differ dependent on situation) and target or true value v_t (the target value for trust is 1):

$$v' = v(1-s) + v_t s$$
$$t' = t(1-s) + s$$

Indirect discovery

When exploring a city with indirect discovery enabled then it finds and immediately explores links connected to that city with a probability **p_find_links**. Neighbouring cities connected through these links are always discovered (i.e. known and properties set to default values) and with a probability **p_find_dest** also explored (with half learning speed and no indirect discovery enabled).

5.4 Travel Decisions

Agents start out at entry cities at one edge of the map and attempt to reach exit cities at the other edge. Agents decide if and where to go purely based on the subjective information they have available. Travel decisions are made in one of two ways. If an agent has enough topological information to be able to find a fully connected route from its current city to an exit, it will plan a path and follow it. Otherwise it will select a destination among all immediately accessible cities.

Properties affecting the decisions

In both cases the decision is affected by a number of properties of the cities and links involved.

attractiveness The attractiveness of a location is calculated as a weighted sum of its quality and the amount of resources it provides (parameter **qual_weight_res**):

$$a = \mathsf{disc(quality)} \cdot (1 - \mathsf{qual\_weight\_res}) + \mathsf{disc(resources)} \cdot \mathsf{qual\_weight\_res}$$

effective friction As trust in an agents's estimate of friction decreases the effective value of friction, on which decisions are based increases:

$$f_{\text{eff}} = \mathsf{frict.value} + \mathsf{frict.value} \cdot (1 - \mathsf{frict.trust}).$$

perceived safety Risk is perceived and communicated in the same way as other properties of cities and links. However, the way risk influences decisions is not as straightforward, as we used the results of an empirical study to calibrate the effects of perceived risk on travel decisions. First we derive the agent's perceived probability to be safe from the raw risk risk.value:

$$p_s = \mathsf{risk.trust} \cdot (1 - \mathsf{risk.value})^{\mathsf{risk\_scale}}$$

Here, **risk_scale** is a parameter that determines how objective risk of dying scales to a general subjective perception to be safe. As with other subjective properties the safety estimate is discounted by the trust an agent has in the quality of the estimate.

Using agent-specific parameters int and slope (see Section 3.2) we can then calculate a safety score s (strictly speaking the probability for an agent to decide in favour of a potentially unsafe action):

$$s = \frac{e^{\mathsf{int} + 100 p_s \mathsf{slope}}}{1 + e^{\mathsf{int} + 100 p_s \mathsf{slope}}}$$

Local travel

If agents are not able to find a route to an exit they will pick a destination at random from all immediately reachable cities (i.e. the current location and all its known neighbours). The probability to pick a location i, p_i is its relative suitability modified by **qual_bias**: $p_i = (l_i / \sum l)^{\mathsf{qual\_bias}}$. Suitability of a location j, reachable by link i depends on proximity to the exit x, attractiveness a and the difficulty of reaching it due to friction and risk (both of which are assumed to be 0 for the current location):

$$l_j = (\mathsf{qual\_weight\_x} \cdot \mathsf{x}_j + a_j) \frac{\mathsf{qual\_tol\_frict}}{\mathsf{qual\_tol\_frict} + f_{\text{eff},i}} s_i$$

Path planning

If an agent knows enough to find a complete route it will attempt to travel the route with the lowest costs. The cost of a route is simply the sum of the costs of all its segments which again are a function of travel effort and properties of the waypoints. Specifically, the costs of a segment consisting of a link i leading to a city j are calculated from effective friction f_{eff} and safety score s of the link (see above), and (quality dependent) cost of the destination as

$$e_i = f_{\text{eff},i}\, e_{q,j} + \mathsf{path\_penalty\_risk} \cdot (1 - s_i).$$

The cost of a waypoint then decreases with increasing attractiveness:

$$e_{q,j} = \frac{1 + \mathsf{path\_penalty\_loc}}{1 + \mathsf{path\_penalty\_loc} \cdot a_j}$$

5.5 Perceived risk

Travel in the model can be risky and agents can perceive and react to that risk. Risk is perceived and communicated by agents in the same way as other properties of the world. In addition, however, perceived risk can be directly affected by deaths either of contacts or by nearby agents. Every death has a chance of being noticed by contacts of the dead agent as well as agents travelling on the same link with probabilities **p_notice_death_c** and **p_notice_death_o**, respectively. If an agent noticing a death knows the link it has occured on, it adjusts its risk estimate for that link towards 1 with learning speeds **speed_risk_obs** and **speed_risk_indir** (for direct observation and contacts, respectively), following the same rules as knowledge updates during exploration (see Section 5.3).

For the effects of risk on decision making see Section 5.4.

Acknowledgements

This work was funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant no. 725232) for 2017-2022, and executed by the University of Southampton, with the University of Rostock.

References

Bijak, J. (in press). *Towards Bayesian Model-Based Demography: Agency, Complexity and Uncertainty in Migration Studies* (Vol. 17). Dordrecht: Springer.

Gillespie, D. T. (1976, December). A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, *22*(4), 403–434. doi: 10.1016/0021-9991(76)90041-3

Hinsch, M., & Bijak, J. (in press). Principles and State of the Art of Agent-Based Migration Modelling. In *Towards Bayesian Model-Based Demography: Agency, Complexity and Uncertainty in Migration Studies.* Dordrecht: Springer.

A Model parameters

`base/params.jl`

World setup		
parameter	*range*	*explanation*
n_cities	$[1,\infty[$	number of (regular) cities
link_thresh	$]0.0,1.0[$	maximum distance to connect two cities
n_exits	$[1,\infty[$	number of exit cities
regular_exits	{true, false}	whether to space exits out regularly
n_entries	$[1,\infty[$	number of entry cities
regular_entries	{true, false}	whether to space entries out regularly
exit_dist	$[0.0,1.0]$	minimum distance to entry to connect city to all exist
entry_dist	$[0.0,1.0]$	maximum distance to entry to connect city to all entries
n_nearest_entry	$[0,\infty[$	number of nearest cities to connect each entry to
n_nearest_exit	$[0,\infty[$	number of nearest cities to connect each exit to
qual_entry	$[0.0,1.0]$	quality of entry cities
res_entry	$[0.0,1.0]$	resources of entry cities
qual_exit	$[0.0,1.0]$	quality of exit cities
res_exit	$[0.0,1.0]$	resources of exit cities
obstacle	$[0.0,1.0]^4$	position of obstacle rectangle (x1, y1, x2, y2)
dist_scale	$[1.0,\infty[^2$	scale between distance and friction (fast, slow)
frict_range	$[0.0,\infty[$	maximum stochastic increase of friction
risk_normal	$[0.0,1.0]$	risk on normal links
risk_high	$[0.0,1.0]$	risk on obstacle links
p_unkown_city	$[0.0,1.0]$	probability for a city to be unknown to all starting agents
p_unknown_link	$[0.0,1.0]$	probability for a link to be unknown to all starting agents

Agent setup		
parameter	*range*	*explanation*
n_ini_contacts	$[0,\infty[$	number of initial contacts
ini_capital	$[0.0,\infty[$	amount of initial capital
p_know_target	$[0.0,1.0]$	probability for a starting agent to already know a given exit
p_know_city	$[0.0,1.0]$	probability for a starting agent to already know a given city
p_know_link	$[0.0,1.0]$	probability for a starting agent to already know a given link
speed_expl_ini	$[0.0,1.0]$	learning speed during initial setup

	Process rates	
parameter	*range*	*explanation*
rate_dep	$[0.0, \infty[$	rate at which new agents are created
rate_costs_stay	$[0.0, \infty[$	rate at which agents accrue costs while not travelling
rate_explore_stay	$[0.0, \infty[$	rate at which agents explore while not travelling
move_rate	$[0.0, \infty[$	base rate at which agents cross links
move_speed	$[0.0, \infty[$	movement speed while crossing links
p_keep_contact	$[0.0, \infty[$	rate (per co-located agent) at which agents acquire new contacts
p_drop_contact	$[0.0, \infty[$	rate (per contact) at which agents lose contacts
p_info_contacts	$[0.0, \infty[$	rate (per contact) at which agents initiate communication
rate_plan	$[0.0, \infty[$	rate at which agents revise their planned route

	Exploration	
parameter	*range*	*explanation*
res_exp	$[0.0, 1.0]$	expected value of resources for newly discovered cities
qual_exp	$[0.0, 1.0]$	expected value of quality for newly discovered cities
frict_exp	$[1.0, \infty[^2$	expected value of friction for newly discovered links
risk_exp	$[0.0, 1.0]$	expected value of risk for newly discovered links
p_find_links	$[0.0, 1.0]$	probability to find links connected to newly explored cities
p_find_dests	$[0.0, 1.0]$	probability to explore cities on the other side of found links
speed_expl_stay	$[0.0, 1.0]$	learning speed while not travelling
speed_expl_move	$[0.0, 1.0]$	learning speed while travelling
p_notice_death_c	$[0.0, 1.0]$	probability contacts become aware of an agent's death
p_notice_death_o	$[0.0, 1.0]$	probability co-located agents become aware of an agent's death
speed_risk_indir	$[0.0, 1.0]$	speed of learning a link's risk from death of contact
speed_risk_obs	$[0.0, 1.0]$	speed of learning a link's risk from observed death
speed_expl_risk	$[0.0, 1.0]$	learning speed for risk

	Resources	
parameter	*range*	*explanation*
costs_stay	$[0.0, \infty[$	resource costs while staying
ben_resources	$[0.0, \infty[$	effect of a city's resources on income
costs_move	$[0.0, \infty[$	resource costs while moving
save_thresh	$[0.0, \infty[$	minimum capital an agent needs to travel

save_income	$[0.0,\infty[$	minimum income an agent needs to stop travelling

Decisions

parameter	*range*	*explanation*
qual_weight_x	$[0.0,\infty[$	effect of geographical location on (local) attractiveness of a city
qual_weight_res	$[0.0,1.0]$	weight of resources relative to quality for attractiveness of a city
qual_tol_frict	$[0.0,\infty[$	insensitivity of (local) attractiveness of a city against friction
qual_bias	$[0.0,\infty[$	bias towards higher (local) attractiveness
path_penalty_loc	$[0.0,\infty[$	reduction in path costs by city attractiveness
path_penalty_risk	$[0.0,\infty[$	effect of risk on path costs
risk_scale	$[0.0,\infty[$	how real probability to survive scales to perceived safety
risk_i	$]-\infty,\infty[$	intercept of map from perceived safety to decision
risk_s	$]-\infty,\infty[$	slope of map from perceived safety to decision
risk_sd_i	$[0.0,\infty[$	standard deviation of intercept
risk_sd_s	$[0.0,\infty[$	standard deviation of slope
risk_cov_i_s	$]-\infty,\infty[$	covariance between intercept and slope

Information exchange

parameter	*range*	*explanation*
n_contacts_max	$[0,\infty[$	maximum number of contacts an agent can have
p_transfer_info	$[0.0,1.0]$	probability for an information item to be exchanged during communication
convince	$[0.0,1.0]$	effect of certainty on doubt, resulting in certainty
convert	$[0.0,1.0]$	effect of certainty on certainty, resulting in certainty
confuse	$[0.0,1.0]$	effect of certainty on certainty, resulting in doubt
error	$[0.0,1.0]$	communication error
error_risk	$[0.0,1.0]$	communication error for risk
error_frict	$[0.0,1.0]$	communication error for friction
weight_arr	$]0,\infty[$	weight of opinion of arrived agents during communication