Package: rsocsim (via r-universe)

June 6, 2026

Type Package

Title SOCSIM

Version 1.9.18

Date 2026-06-01

Maintainer Tom Theile <`theile@demogr.mpg.de`>

Description Tools for preparing inputs, running SOCSIM (SOCial SIMulator) demographic kinship microsimulations, and reading simulation outputs from R. The package includes helpers for creating simulation folders, downloading demographic rate schedules, starting simulations, and loading population and marriage result files.

License GPL-3

Imports Rcpp (>= 1.0.5), magrittr, dplyr, tidyr, utils

LinkingTo Rcpp

Suggests future, parallel, testthat (>= 3.0.0)

Config/testthat/edition 3

URL `https://github.com/MPIDR/rsocsim`, `https://mpidr.github.io/rsocsim/`

BugReports `https://github.com/MPIDR/rsocsim/issues`

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.3

Config/pak/sysreqs libicu-dev

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

Date/Publication 2026-06-01 06:26:55 UTC

RemoteUrl https://github.com/mpidr/rsocsim

RemoteRef HEAD

RemoteSha 125450cff968bb3ff762fa99d4c926a8136062a2

Contents

`create_simulation_folder`

Create a directory structure for the simulation

Description

Create a two-level directory structure. If the first-level argument is NULL, we look for and, if needed, create the directory 'socsim' in the current temporary directory. If the second-level argument is NULL, we create a directory named like 'socsim_sim_' followed by a random component in the first-level directory.

Usage

```
create_simulation_folder(basedir = NULL, simdir = NULL)
```

Arguments

`basedir`	A string. Optional. First-level directory where the simulation-specific directory will be created. Defaults to `file.path(tempdir(), "socsim")`.
`simdir`	A string. Optional. Simulation-specific directory which will be created within 'basedir'. Defaults to 'socsim_sim_' plus a random component created with `tempfile()`.

Value

A string. The full path to the simulation-specific directory.

`create_sup_file` *Create a basic .sup file for a simulation*

Description

The simulation is only a simple one. The file will be saved into the directory 'simdir'.

Usage

```
create_sup_file(simdir, simname = "socsim")
```

Arguments

`simdir`	A string. The directory where the .sup file will be saved.
`simname`	A string. The base name of the simulation. Defaults to `"socsim"`.

Value

A string. The basename of the created supervisory file, for example `"socsim.sup"`. The file is written to `simdir`, and the function also copies the bundled rate and initial-population input files into `simdir`.

`estimate_fertility_rates` *Estimate yearly age-specific fertility rates (ASFR) from a SOCSIM-generated population file*

Description

Given a population file ('opop') generated by rsocsim, the function estimates age-specific fertility rates.

Usage

```
estimate_fertility_rates(
  opop,
  final_sim_year,
  year_min,
  year_max,
  year_group = 5,
  age_min_fert = 15,
  age_max_fert = 50,
  age_group = 5
)
```

Arguments

`opop`	An R object from SOCSIM microsimulation output (population file).
`final_sim_year`	numeric. Final simulated year in 'real world' time (used to convert 'SOCSIM time' to 'real world' time.)
`year_min`	numeric. Lower-bound year for which rate should be estimated.
`year_max`	numeric. Upper-bound year for which rate should be estimated.
`year_group`	numeric. Size of year groups to estimate rate (year_group=1 will produce single-year estimates)
`age_min_fert`	numeric. Lower-bound age of female reproductive period
`age_max_fert`	numeric. Upper-bound age of female reproductive period
`age_group`	numeric. Size of age groups to estimate rate (age_group=1 will produce single-age estimates)

Details

The `final_sim_year` can be obtained from the .sup file and must refer to to a real-world year.

Grouped year and age ranges (i.e., if `year_group > 1` or `age_group > 1`) are created as [year;year+year_group).

Value

A data frame with columns `year`, `age`, and `socsim`. `year` is a factor describing the grouped calendar-year interval, age is a factor describing the maternal-age interval, and `socsim` is the estimated fertility rate for that cell.

Examples

```
opop <- data.frame(
  pid = 1:6,
  fem = c(1, 0, 1, 0, 1, 1),
  group = 1,
  nev = 0,
  dob = c(120, 120, 336, 348, 180, 360),
  mom = c(0, 0, 1, 1, 0, 5),
  pop = c(0, 0, 2, 2, 0, 2),
  nesibm = 0,
  nesibp = 0,
  lborn = 0,
  marid = 0,
  mstat = 0,
  dod = c(0, 300, 0, 0, 0, 0),
  fmult = 0
)

asfr <- estimate_fertility_rates(opop = opop,
                     final_sim_year = 2021,
                     year_min = 1998,
                     year_max = 2000,
                     year_group = 5,
```

```
                       age_min_fert = 15,
                       age_max_fert = 50,
                       age_group = 5)
head(asfr)
```

`estimate_mortality_rates` *Estimate yearly age-specific mortality rates (ASMR) from a SOCSIM-generated population file*

Description

Given a population file ('opop') generated by rsocsim, the function estimates (yearly) age-specific mortality rates.

Usage

```
estimate_mortality_rates(
  opop,
  final_sim_year,
  year_min,
  year_max,
  year_group,
  age_max_mort,
  age_group
)
```

Arguments

`opop`	An R object from SOCSIM microsimulation output (population file).
`final_sim_year`	numeric. Final simulated year in 'real world' time (used to convert 'SOCSIM time' to 'real world' time.)
`year_min`	numeric. Lower-bound year for which rate should be estimated.
`year_max`	numeric. Upper-bound year for which rate should be estimated.
`year_group`	numeric. Size of year groups to estimate rate (year_group=1 will produce single-year estimates)
`age_max_mort`	numeric. Maximum age for estimating mortality.
`age_group`	numeric. Size of age groups to estimate rate (age_group=1 will produce single-age estimates)

Details

The `final_sim_year` can be obtained from the .sup file and must refer to to a real-world year.

Grouped year and age ranges (i.e., if `year_group > 1` or `age_group > 1`) are created as [year;year+year_group). For `age_group > 1`, mortality rates are split into an infant group [0,1) and then grouped ages [1, age_group), [age_group, age_group + age_group), and so on.

Value

A data frame with columns `year`, `sex`, `age`, and `socsim`. `year` is a factor describing the grouped calendar-year interval, `sex` is `"male"` or `"female"`, `age` is a factor describing the age interval, and `socsim` is the estimated mortality rate for that cell.

Examples

```
opop <- data.frame(
  pid = 1:6,
  fem = c(1, 0, 1, 0, 1, 1),
  group = 1,
  nev = 0,
  dob = c(120, 120, 336, 348, 180, 360),
  mom = c(0, 0, 1, 1, 0, 5),
  pop = c(0, 0, 2, 2, 0, 2),
  nesibm = 0,
  nesibp = 0,
  lborn = 0,
  marid = 0,
  mstat = 0,
  dod = c(0, 300, 0, 0, 0, 0),
  fmult = 0
)

asmr <- estimate_mortality_rates(opop = opop,
                     final_sim_year = 2021,
                     year_min = 1995,
                     year_max = 2000,
                     year_group = 5,
                     age_max_mort = 100,
                     age_group = 5)
head(asmr)
```

`read_omar` *Read output marriage file into a data frame*

Description

When `fn` contains multiple file paths, or when `seed` contains multiple values and `fn` is `NULL`, the matching result files are read and row-bound into a single data frame. To keep identifiers unique across simulations, positive ID columns are offset by `(index - 1) * id_offset`, while sentinel zeros remain unchanged.

Usage

```
read_omar(
  folder = NULL,
  supfile = "socsim.sup",
  seed = 42,
```

```
  suffix = "",
  fn = NULL,
  id_offset = 10000000L,
  quiet = FALSE
)
```

Arguments

`folder`	simulation base folder ("~/socsim/simulation_235/")
`supfile`	name of supervisory-file ("socsim.sup")
`seed`	random number seed (42)
`suffix`	optional suffix for the results-directory (default="")
`fn`	complete path to the file. If not provided, it will be created from the other arguments
`id_offset`	positive integer stride used to offset IDs when combining multiple files. Ignored for single-file reads. Default is 10 million, which allows combining up to 214 files with a total population of 10 million each.
`quiet`	logical. If `FALSE`, emit a message with the file path being read.

Details

1	mid	Marriage id number (unique sequential integer)
2	wpid	Wife's person id
3	hpid	Husband's person id
4	dstart	Date marriage began
5	dend	Date marriage ended or zero if still in force at end of simulation
6	rend	Reason marriage ended 2 = divorce; 3 = death of one partner
7	wprior	Marriage id of wife's next most recent prior marriage
8	hprior	Marriage id of husband's next most recent prior marriage

you can either provide the complete `path` to the file or the `folder`, supfilename, seed and suffix with which you started the simulation

Value

A data frame with columns `mid`, `wpid`, `hpid`, `dstart`, `dend`, `rend`, `wprior`, and `hprior`, matching the SOCSIM `result.omar` file. If the file is missing or empty, a zero-row data frame with these columns is returned.

`read_opop` *Read output population file into a data frame*

Description

When `fn` contains multiple file paths, or when `seed` contains multiple values and `fn` is `NULL`, the matching result files are read and row-bound into a single data frame. To keep identifiers unique across simulations, positive ID columns are offset by `(index - 1) * id_offset`, while sentinel zeros remain unchanged.

Usage

```
read_opop(
  folder = NULL,
  supfile = "socsim.sup",
  seed = 42,
  suffix = "",
  fn = NULL,
  id_offset = 10000000L,
  quiet = FALSE
)
```

Arguments

`folder`	simulation base folder ("~/socsim/simulation_235/")
`supfile`	name of supervisory-file ("socsim.sup")
`seed`	random number seed (42)
`suffix`	optional suffix for the results-directory (default="")
`fn`	complete path to the file. If not provided, it will be created from the other arguments
`id_offset`	positive integer stride used to offset IDs when combining multiple files. Ignored for single-file reads. Default is 10 million, which allows combining up to 214 files with a total population of 10 million each.
`quiet`	logical. If `FALSE`, emit a message with the file path being read.

Details

after the end of the simulation, socsim writes every person of the simulation into a file called result.opop |

1	pid	Person id unique identifier assigned as integer in birth order
2	fem	1 if female 0 if male
3	group	Group identifier 1..60 current group membership of individual
4	nev	Next scheduled event
5	dob	Date of birth integer month number

6	mom	Person id of mother
7	pop	Person id of father
8	nesibm	Person id of next eldest sibling through mother
9	nesibp	Person id of next eldest sibling through father
10	lborn	Person id of last born child
11	marid	Id of marriage in .omar file
12	mstat	Marital status at end of simulation integer 1=single;2=divorced; 3=widowed; 4=married
13	dod	Date of death or 0 if alive at end of simulation
14	fmult	Fertility multiplier

This table explains the columns of the opop file and the columns of the output data frame. You can either provide the complete `path` to the file or the `folder`, supfilename, seed and suffix with which you started the simulation

Value

A data frame with columns `pid`, `fem`, `group`, `nev`, `dob`, `mom`, `pop`, `nesibm`, `nesibp`, `lborn`, `marid`, `mstat`, `dod`, and `fmult`, matching the SOCSIM `result.opop` file. If the file is missing or empty, a zero-row data frame with these columns is returned.

`retrieve_kin` *Identify members of a kin network for an individual or individuals of interest.*

Description

Identify members of a kin network for an individual or individuals of interest.

Usage

```
retrieve_kin(
  opop,
  omar,
  pid,
  extra_kintypes = character(),
  kin_by_sex = FALSE,
  KidsOf = NULL
)
```

Arguments

`opop`	An R object from SOCSIM microsimulation output (population file). Create this object with the function read_opop().
`omar`	An R object from SOCSIM microsimulation output (marriage file). Create this object with the function read_omar().

`pid`	A vector of person IDs, indicating persons of interest for whom these kin networks should be identified.
`extra_kintypes`	A vector of character values indicating which additional types of kin should be obtained. For reasons of computational efficiency, the function will by default only identify an individual's great-grandparents ("ggparents" in function output), grandparents ("gparents"), parents, siblings, spouse, children, and grandchildren ("gchildren"). However, by selecting one or more of the following kin types, the kin network generated will also include these individuals: • "gunclesaunts": Great-uncles and great-aunts • "unclesaunts": Uncles and aunts • "firstcousins": First cousins (Children of uncles and aunts) • "niblings": Nieces and nephews (Children of siblings) • "inlaws": Parents-in-law (parents of spouse) and brothers and sisters in law (siblings of spouse and spouse of siblings)
`kin_by_sex`	A logical value indicating whether output should include kin relations additionally disaggregated by the sex of the relative. Setting this value to TRUE will result in additional objects being generated to identify individuals' relatives by sex.
`KidsOf`	An optional precomputed list object containing the children of each person in the population. If `NULL`, it is built from opop.

Value

A named list whose components are kinship categories such as `parents`, `siblings`, or `children`. Each component is itself a named list of integer person IDs, organized by relationship. These person ID values will be named based on the person of interest with whom they are associated. For example, for a list named "parents", the values will be person IDs of the parents of individuals of interest. These values will be named according to their children's IDs (given that their children are, in this case, the persons of interest provided to the function input). With kin_by_sex set to TRUE and extra_kintypes set to c(c("gunclesaunts", "unclesaunts", "firstcousins", "niblings", "inlaws")), the full list of kin relations identified are:

- "ggparents": great-grandparents
- "ggmothers": great-grandmothers
- "ggfathers": great-grandfathers
- "gparents": grandparents
- "gmothers": grandmothers
- "gfathers": grandfathers
- "gunclesaunts": great-uncles and great-aunts
- "guncles": great-uncles
- "gaunts": great-aunts
- "parents": parents
- "mother": mother
- "father": father

- "unclesaunts": uncles and aunts (siblings of parents)
- "uncles": uncles
- "aunts": aunts
- "siblings": siblings
- "sisters": sisters
- "brothers": brothers
- "firstcousins": first cousins
- "firstcousinsfemale": female first cousins
- "firstcousinsmale": male first cousins
- "children": children
- "daughters": daughters
- "sons": sons
- "gchildren": grandchildren
- "gdaughters": granddaughters
- "gsons": grandsons
- "niblings": nephews and nieces
- "nieces": nieces
- "nephews": nephews
- "spouse": spouse (based on final marriage, in the case of multiple marriages)
- "parentsinlaw": parents-in-law
- "motherinlaw": mother-in-law
- "fatherinlaw": father-in-law
- "siblingsinlaw": brothers and sisters in law
- "sistersinlaw": sisters-in-law
- "brothersinlaw": brothers-in-law

Examples

```
opop <- data.frame(
  pid = 1:4,
  fem = c(1, 0, 1, 0),
  group = 1,
  nev = 0,
  dob = c(120, 120, 300, 300),
  mom = c(0, 0, 1, 1),
  pop = c(0, 0, 2, 2),
  nesibm = 0,
  nesibp = 0,
  lborn = 0,
  marid = c(1, 1, 0, 0),
  mstat = c(4, 4, 1, 1),
  dod = 0,
  fmult = 0
```

```
)
omar <- data.frame(
  mid = 1,
  wpid = 1,
  hpid = 2,
  dstart = 0,
  dend = 0,
  rend = 0,
  wprior = 0,
  hprior = 0
)

kin_network <- retrieve_kin(
  opop = opop,
  omar = omar,
  pid = 3,
  extra_kintypes = c("niblings", "inlaws"),
  kin_by_sex = TRUE
)
kin_network$parents[[1]]
```

`simulation_time_to_years`

Calculate for how many years the simulation ran

Description

Calculate for how many years the simulation ran

Usage

```
simulation_time_to_years(simulation_time, pre_simulation_time, start_year)
```

Arguments

`simulation_time`	An integer. The number of periods (months) the simulation ran.
`pre_simulation_time`	An integer. The number of periods (months) the simulation ran before getting to a stable population. This is subtracted from 'simulation_time' in order to arrive at the "real" simulation time
`start_year`	An integer. The year the simulation started.

Value

A numeric scalar giving the calendar year reached at the end of the simulated period after subtracting `pre_simulation_time / 12`. The value can include a fractional year.

`socsim` *Run a single Socsim simulation with a given supervisory file and directory*

Description

Run a single Socsim simulation with a given supervisory file and directory

Usage

```
socsim(
  folder,
  supfile,
  seed = "42",
  process_method = "inprocess",
  compatibility_mode = "1",
  suffix = ""
)
```

Arguments

`folder`	A string. This is the base directory of the simulation. Every .sup and rate file should be named relative to this directory.
`supfile`	A string. The name of the .sup file to start the simulation, relative to the directory.
`seed`	A string. The seed for the RNG, so expects an integer. Defaults to "42".
`process_method`	A string. Whether and how SOCSIM should be started in its own process or in the running R process. Defaults to "inprocess". Use one of: • "future" - the safest option. A new process will be started via the "future" package • "inprocess" - SOCSIM runs in the R-process. Beware if you run several different simulations, they may affect later simulations. • "clustercall" - if the future package is not available, try this method instead.
`compatibility_mode`	A string.
`suffix`	A string.

Value

Returns `1L` when the simulation finishes successfully. If the simulation errors before completion, the function returns `NULL` after issuing warnings. Result files are written to the directory `sim_results_<basename(supfile)>_<seed>_<suffix>` inside `folder`.

Index