

# Package ‘phylopomp’

October 30, 2024

## Contents

|                                       |    |
|---------------------------------------|----|
| as.data.frame . . . . .               | 2  |
| curtail . . . . .                     | 2  |
| diagram . . . . .                     | 3  |
| gendat . . . . .                      | 4  |
| geneal . . . . .                      | 5  |
| genealogy diagram internals . . . . . | 5  |
| getInfo . . . . .                     | 7  |
| lbdp . . . . .                        | 8  |
| lineages . . . . .                    | 10 |
| moran . . . . .                       | 11 |
| newick . . . . .                      | 12 |
| parse_newick . . . . .                | 13 |
| phylopomp . . . . .                   | 14 |
| reexports . . . . .                   | 14 |
| s2i2r2 . . . . .                      | 15 |
| seir . . . . .                        | 16 |
| si2r . . . . .                        | 19 |
| siir . . . . .                        | 21 |
| simulate . . . . .                    | 23 |
| sir . . . . .                         | 24 |
| treeplot . . . . .                    | 26 |
| twospecies . . . . .                  | 28 |
| yaml . . . . .                        | 31 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>32</b> |
|--------------|-----------|

---

|                            |                               |
|----------------------------|-------------------------------|
| <code>as.data.frame</code> | <i>Coerce to a Data Frame</i> |
|----------------------------|-------------------------------|

---

### Description

Functions to coerce an object to a data frame.

### Usage

```
## S3 method for class 'gpln'
as.data.frame(x, ...)
```

### Arguments

|                  |   |
|------------------|---|
| <code>x</code>   | any R object.   |
| <code>...</code> | additional arguments to be passed to or from methods. |

### Details

An object of class ‘gpln’ is coerced to a data frame by means of `as.data.frame`.

---

|                      |  |
|----------------------|--|
| <code>curtail</code> | <i>Curtail a genealogy to the given time</i> |
|----------------------|--|

---

### Description

Discards all nodes beyond the given time.

### Usage

```
curtail(object, time = NA, prune = TRUE, obscure = TRUE)
```

### Arguments

|                      |                                   |
|----------------------|-----------------------------------|
| <code>object</code>  | gpsim object.                     |
| <code>time</code>    | logical; return the current time? |
| <code>prune</code>   | logical; prune the genealogy?     |
| <code>obscure</code> | logical; obscure the demes?       |

### Value

A curtailed genealogy object.

**Examples**

```

library(ggplot2)

simulate("SIIR",time=5) -> x

plot_grid(
  x |>
    plot(prune=FALSE,points=TRUE),
  x |>
    curtail(time=3) |>
    plot(prune=FALSE,points=TRUE)+
    expand_limits(x=5),
  ncol=1,align="h",axis="tblr"
)

plot_grid(
  x |>
    plot(prune=TRUE,points=TRUE)+
    geom_vline(xintercept=3),
  x |> curtail(time=3) |>
    plot(prune=TRUE,points=TRUE)+
    geom_vline(xintercept=3)+
    expand_limits(x=5),
  ncol=1,align="h",axis="tblr"
)

```

---

diagram

*Genealogy process diagram*


---

**Description**

Produces a diagram of the genealogy process state.

**Usage**

```

diagram(
  object,
  prune = TRUE,
  obscure = TRUE,
  m = NULL,
  n = NULL,
  ...,
  digits = 1,
  palette = scales::hue_pal(l = 80, c = 20, h = c(220, 580))
)

## S3 method for class 'gpdiag'
print(x, newpage = is.null(vp), vp = NULL, ...)

```

**Arguments**

|         |  |
|---------|--|
| object  | gpsim object.  |
| prune   | logical; prune the genealogy?  |
| obscure | logical; obscure the demes?  |
| m       | width of plotting window, in nodes. By default, the nodes will be adjusted in width to fit the window.   |
| n       | height of the pockets, in balls. By default, the balls will be adjusted in size to fit the space available.  |
| ...     | other arguments, ignored.  |
| digits  | non-negative integer; number of decimal digits to print in the node time   |
| palette | color palette for indicating demes. This can be furnished either as a function or a vector of colors. If this is a function, it should take a single integer argument, the number of colors required. If it is a vector, it should have at least as many elements as there are demes in the genealogy. |
| x       | An R object.   |
| newpage | draw new empty page first?   |
| vp      | viewport to draw plot in   |

**Value**

A **grid** graphics object (grob), invisibly.

**Examples**

```
runSIR(Beta=3,gamma=0.1,psi=0.2,S0=100,I0=5,R0=0,time=2,t0=0) -> x
plot(x,points=TRUE,prune=FALSE)
plot_grid(plotlist=list(plot(x,points=TRUE)[[1]],diagram(x)),
  ncol=1,rel_heights=c(4,1))
```

---

gendat

*Genealogy as a data frame*


---

**Description**

Converts a given genealogy to a data frame.

**Usage**

```
gendat(object)
```

**Arguments**

|        |                   |
|--------|-------------------|
| object | a 'gpgen' object. |
|--------|-------------------|

**Value**

A list of objects containing the information pertinent for filtering.

---

|        |                       |
|--------|-----------------------|
| geneal | <i>Bare genealogy</i> |
|--------|-----------------------|

---

**Description**

Extracts the bare genealogy from a Markov genealogy process simulation

**Usage**

```
geneal(object)
```

**Arguments**

object            a ‘gpger’ object.

**Value**

A bare genealogy object.

---

|                             |                              |
|-----------------------------|------------------------------|
| genealogy diagram internals | <i>Diagramming internals</i> |
|-----------------------------|------------------------------|

---

**Description**

Facilities to produce diagrammatic representations of genealogy process states.

**Usage**

```
genealogyGrob(object, m = NULL, n = NULL, vp = NULL, palette, ...)
```

```
nodeGrob(object, digits = 1, palette, n = NULL, vp = NULL)
```

```
pocketGrob(object, n, vp = NULL)
```

```
ballGrob(object, vp = NULL)
```

```
resizingTextGrob(..., vp = NULL)
```

```
## S3 method for class 'resizingTextGrob'
drawDetails(x, recording = TRUE)
```

```
## S3 method for class 'resizingTextGrob'
preDrawDetails(x)
```

```
## S3 method for class 'resizingTextGrob'
postDrawDetails(x)

## S3 method for class 'ballGrob'
drawDetails(x, recording = TRUE)

## S3 method for class 'ballGrob'
preDrawDetails(x)

## S3 method for class 'ballGrob'
postDrawDetails(x)

## S3 method for class 'gpsim'
print(x, ...)

## S3 method for class 'gpngen'
print(x, ...)

## S3 method for class 'gpyaml'
print(x, ...)
```

### Arguments

|           |  |
|-----------|--|
| object    | list; pocket structure   |
| m         | width of plotting window, in nodes. By default, the nodes will be adjusted in width to fit the window.   |
| n         | length of longest genealogy  |
| vp        | viewport to draw plot in   |
| palette   | color palette for indicating demes. This can be furnished either as a function or a vector of colors. If this is a function, it should take a single integer argument, the number of colors required. If it is a vector, it should have at least as many elements as there are demes in the genealogy. |
| ...       | arguments to be passed to <a href="#">textGrob</a> .   |
| digits    | non-negative integer; number of decimal digits to print in the node time   |
| x         | An R object.   |
| recording | A logical value indicating whether a grob is being added to the display list or redrawn from the display list.   |

### Details

Code for the resizing text adapted from a blog post by Mark Heckmann (<https://ryouready.wordpress.com/2012/08/01/creating-a-text-grob-that-automatically-adjusts-to-viewport-size/>).

getInfo

*getInfo***Description**

Retrieve information from genealogy process simulation

**Usage**

```
getInfo(
  object,
  prune = TRUE,
  obscure = TRUE,
  t0 = FALSE,
  time = FALSE,
  description = FALSE,
  structure = FALSE,
  yaml = FALSE,
  ndeme = FALSE,
  lineages = FALSE,
  newick = FALSE,
  nsample = FALSE,
  genealogy = FALSE,
  gendat = FALSE
)
```

**Arguments**

|             |  |
|-------------|--|
| object      | gpsim object.  |
| prune       | logical; prune the genealogy?                            |
| obscure     | logical; obscure the demes?                              |
| t0          | logical; return the zero-time?                           |
| time        | logical; return the current time?                        |
| description | logical; return the description?                         |
| structure   | logical; return the structure in R list format?          |
| yaml        | logical; return the structure in YAML format?            |
| ndeme       | logical; return the number of demes?                     |
| lineages    | logical; return the lineage-count function?              |
| newick      | logical; return a Newick-format description of the tree? |
| nsample     | logical; return the number of samples?                   |
| genealogy   | logical; return the lineage-traced genealogy?            |
| gendat      | logical; return the data-frame format?                   |

**Value**

A list containing the requested elements, including any or all of:

**t0** the initial time (a numeric scalar)  
**time** the final time (a numeric scalar)  
**ndeme** the number of demes (an integer)  
**nsample** the number of samples (an integer)  
**newick** the genealogical tree, in Newick format  
**description** a human readable description of the state of the genealogy process  
**yaml** the state of the genealogy process in YAML format  
**structure** the state of the genealogy process in R list format  
**lineages** a [tibble](#) containing the lineage count function through time  
**gendat** a [tibble](#) containing the (obscured) genealogy in a data-frame format  
**genealogy** the lineage-traced genealogy (as a raw vector)

**Examples**

```
simulate("SIIR",time=3,psi1=1,psi2=0) |>
  simulate(Beta1=2,gamma=2,time=10,psi1=10,psi2=1) |>
  plot()

runSIIR(Beta1=10,Beta2=8,
  S0=200,I1_0=10,I2_0=8,R0=0,time=0,t0=-1) |>
  simulate(psi1=10,time=2) |>
  plot(points=TRUE,obscure=FALSE)

simulate("SIIR",Beta1=2,Beta2=50,gamma=1,psi1=2,
  S0=300,I1_0=20,I2_0=2,time=5) |>
  lineages() |>
  plot()
```

---

lbdp

---

*Linear birth-death-sampling model*


---

**Description**

The genealogy process induced by a simple linear birth-death process with constant-rate sampling.

**Usage**

```
runLBDP(time, t0 = 0, lambda = 2, mu = 1, psi = 1, n0 = 5)

continueLBDP(object, time, lambda = NA, mu = NA, psi = NA)

lbdp_exact(x, lambda, mu, psi, n0 = 1)

lbdp_pomp(x, lambda, mu, psi, n0 = 1, t0 = 0)
```



## Arguments

|        |   |
|--------|---|
| time   | final time  |
| t0     | initial time  |
| lambda | per capita birth rate   |
| mu     | per capita recovery rate.   |
| psi    | per capita sampling rate.   |
| n0     | initial population size   |
| object | either the name of the model to simulate <i>or</i> a previously computed ‘gpsim’ object |
| x      | genealogy in <b>phylopomp</b> format (i.e., an object that inherits from ‘gpgen’).      |

## Details

`lbdp_exact` gives the exact log likelihood of a linear birth-death process, conditioned on  $n_0 = 0$  (Stadler, 2010, Thm 3.5). The derivation is also given in comments in the code.

`lbdp_pomp` constructs a **pomp** object containing a given set of data and a linear birth-death-sampling process.

## Value

`runLBDP` and `continueLBDP` return objects of class ‘gpsim’ with ‘model’ attribute “LBDP”.

`lbdp_exact` returns the log likelihood of the genealogy. Note that the time since the most recent sample is informative.

## References

- A. A. King, Q. Lin, and E. L. Ionides. Exact phylodynamic likelihood via structured Markov genealogy processes. *arXiv* 2405.17032, 2024. doi:10.48550/arxiv.2405.17032.
- A. A. King, Q. Lin, and E. L. Ionides. Markov genealogy processes. *Theoretical Population Biology* **143**, 77–91, 2022. doi:10.1016/j.tpb.2021.11.003.
- T. Stadler. Sampling-through-time in birth-death trees. *Journal of Theoretical Biology* **267**, 396–404, 2010. doi:10.1016/j.jtbi.2010.09.010.
- T. Stadler. Sampling-through-time in birth-death trees. *Journal of Theoretical Biology* **267**, 396–404, 2010. doi:10.1016/j.jtbi.2010.09.010.

## See Also

More example genealogy processes: [moran](#), [s2i2r2](#), [seir](#), [si2r](#), [siir](#), [simulate\(\)](#), [sir](#), [twospecies](#)

## Examples

```
simulate("LBDP",time=4) |> plot(points=TRUE)

simulate("LBDP",lambda=2,mu=1,psi=3,n0=1,time=1) |>
  simulate(time=10,lambda=1) |>
  plot()
```

```
simulate("LBDP",time=4) |>
  lineages() |>
  plot()
```

---

|          |                               |
|----------|-------------------------------|
| lineages | <i>Lineage-count function</i> |
|----------|-------------------------------|

---

### Description

Lineage-counts, saturations, and event-codes.

### Usage

```
lineages(object, prune = TRUE, obscure = TRUE)

## S3 method for class 'gplin'
plot(x, ..., palette = scales::hue_pal(l = 30, h = c(220, 580)))
```

### Arguments

|         |  |
|---------|--|
| object  | gpsim object.  |
| prune   | logical; prune the genealogy?  |
| obscure | logical; obscure the demes?  |
| x       | object of class 'gpger'  |
| ...     | passed to <a href="#">theme</a> .  |
| palette | color palette for branches. This can be furnished either as a function or a vector of colors. If this is a function, it should take a single integer argument, the number of colors required. If it is a vector, it should have at least as many elements as there are demes in the genealogy. |

### Details

This function extracts from the specified genealogy several important time-varying quantities. These include:

**lineages** number of lineages through time  
**saturation** the number of lineages emerging from the event  
**event\_type** an integer coding the type of event

If the genealogy has been obscured (the default), the number in the `lineages` returned is the total number of lineages present at the specified time and the saturation is the total saturation. If the genealogy has not been obscured (`obscure = FALSE`), the deme-specific data are returned. In this case, the `deme` column specifies the pertinent deme.

The event types are:

- 0** no event,
- 1** a root,
- 1** a sample event,
- 2** a non-sample event,
- 3** the end of the time interval, which may or may not coincide with the latest tip of the genealogy.

### Value

A [tibble](#) containing information about the genealogy. See Details for specifics. The [tibble](#) returned by `lineages` has a [plot](#) method.

### Examples

```
library(tidyverse)

pal <- c("#00274c", "#ffcb05")

simulate("SIIR", time=3) -> x
plot_grid(
  x |> plot(),
  x |> lineages() |> plot(),
  x |> plot(obscurse=FALSE, palette=pal),
  x |> lineages(obscurse=FALSE) |>
    plot(palette=pal, legend.position=c(0.8, 0.9)),
  align="v", axis="b",
  ncol=2, byrow=FALSE
)
```

---

moran

*The classical Moran model*


---

### Description

The Markov genealogy process induced by the classical Moran process, in which birth/death events occur at a constant rate and the population size remains constant.

### Usage

```
runMoran(time, t0 = 0, mu = 1, psi = 1, n = 100)

continueMoran(object, time, mu = NA, psi = NA)

moran_exact(x, n = 100, mu = 1, psi = 1)
```

**Arguments**

|        |   |
|--------|---|
| time   | final time  |
| t0     | initial time  |
| mu     | per capita event rate   |
| psi    | per capita sampling rate  |
| n      | population size   |
| object | either the name of the model to simulate <i>or</i> a previously computed ‘gpsim’ object |
| x      | genealogy in <b>phylopomp</b> format (i.e., an object that inherits from ‘gpgen’).      |

**Details**

`moran_exact` gives the exact log likelihood of a genealogy under the uniformly-sampled Moran process.

**Value**

`runMoran` and `continueMoran` return objects of class ‘gpsim’ with ‘model’ attribute “Moran”.  
`moran_exact` returns the log likelihood of the genealogy.

**References**

P.A.P. Moran. Random processes in genetics. *Mathematical Proceedings of the Cambridge Philosophical Society* **54**, 60–71, 1958. doi:[10.1017/s0305004100033193](https://doi.org/10.1017/s0305004100033193).

**See Also**

More example genealogy processes: [lbdp](#), [s2i2r2](#), [seir](#), [si2r](#), [siir](#), [simulate\(\)](#), [sir](#), [twospecies](#)

---

newick

*Newick output*


---

**Description**

Extract a Newick-format description of a genealogy.

**Usage**

```
newick(object, prune = TRUE, obscure = TRUE)
```

**Arguments**

|         |                               |
|---------|-------------------------------|
| object  | gpsim object.                 |
| prune   | logical; prune the genealogy? |
| obscure | logical; obscure the demes?   |

**Value**

A string in Newick format.

**Examples**

```
simulate("SIIR",time=1) |> newick()
```

---

|              |                              |
|--------------|------------------------------|
| parse_newick | <i>parse a Newick string</i> |
|--------------|------------------------------|

---

**Description**

Parses a Newick description and returns a binary version of the genealogy.

**Usage**

```
parse_newick(x, t0 = 0, tf = NA)
```

**Arguments**

|    |   |
|----|---|
| x  | character; the Newick description. See Details for specifics. |
| t0 | numeric; the root time.                                       |
| tf | numeric; the current or final time.                           |

**Details**

parse\_newick can only handle a subset of the full Newick specification. In particular, labels are assumed to be of the form <TYPE>\_<DEME>\_<LABEL>, i.e., each label has three parts, separated by underscores ('\_'). The parts are as follows.

- TYPE must be a single character from among the following: 'b', 'g', 'm', 'o'.
  - 'b' signifies a sample.
  - 'g' signifies an internal node.
  - 'm' signifies a root.
  - 'o' indicates an extant lineage.
- DEME must be a non-negative integer, specifying the deme in which the branch resides. If deme information is not present, use 0.
- LABEL is ignored and may be left out.

**Value**

An object of class "gpgen".

---

|           |                                      |
|-----------|--------------------------------------|
| phylopomp | <i>Phylodynamics for POMP models</i> |
|-----------|--------------------------------------|

---

## Description

Simulation and inference of Markov genealogy processes.

## Author(s)

Aaron A. King, Qianying Lin

## References

A. A. King, Q. Lin, and E. L. Ionides. Exact phylodynamic likelihood via structured Markov genealogy processes. *arXiv* 2405.17032, 2024. doi:[10.48550/arxiv.2405.17032](https://doi.org/10.48550/arxiv.2405.17032).

A. A. King, Q. Lin, and E. L. Ionides. Markov genealogy processes. *Theoretical Population Biology* **143**, 77–91, 2022. doi:[10.1016/j.tpb.2021.11.003](https://doi.org/10.1016/j.tpb.2021.11.003).

---

|           |   |
|-----------|---|
| reexports | <i>Objects exported from other packages</i> |
|-----------|---|

---

## Description

These objects are imported from other packages. Follow the links below to see their documentation.

**cowplot** [plot\\_grid](#)

**foreach** [%dopar%](#), [foreach](#), [registerDoSEQ](#)

**grid** [viewport](#)

**pomp** [bake](#), [freeze](#), [stew](#)

**yaml** [as.yaml](#), [read\\_yaml](#)

s2i2r2

*Two-host infection model with waning, immigration, and demography.***Description**

The population is structured by infection progression and host species.

**Usage**

```
runS2I2R2(
  time,
  t0 = 0,
  Beta11 = 4,
  Beta12 = 0,
  Beta22 = 4,
  gamma1 = 1,
  gamma2 = 1,
  psi1 = 1,
  psi2 = 0,
  omega1 = 0,
  omega2 = 0,
  b1 = 0,
  b2 = 0,
  d1 = 0,
  d2 = 0,
  iota1 = 0,
  iota2 = 0,
  S1_0 = 100,
  S2_0 = 100,
  I1_0 = 0,
  I2_0 = 10,
  R1_0 = 0,
  R2_0 = 0
)

continueS2I2R2(
  object,
  time,
  Beta11 = NA,
  Beta12 = NA,
  Beta22 = NA,
  gamma1 = NA,
  gamma2 = NA,
  psi1 = NA,
  psi2 = NA,
  omega1 = NA,
  omega2 = NA,
```

```

    b1 = NA,
    b2 = NA,
    d1 = NA,
    d2 = NA,
    iota1 = NA,
    iota2 = NA
  )

```

Arguments

|                |   |
|----------------|---|
| time           | final time  |
| t0             | initial time  |
| Beta11, Beta22 | transmission rates within species 1 and 2, respectively                                 |
| Beta12         | transmission from species 2 to species 1  |
| gamma1, gamma2 | recovery rates for species 1 and 2, respectively  |
| psi1, psi2     | per capita sampling rates   |
| omega1, omega2 | rates of waning of immunity   |
| b1, b2         | per capita birth rates  |
| d1, d2         | per capita death rates  |
| iota1, iota2   | infection importation rates   |
| S1_0, S2_0     | initial sizes of susceptible populations  |
| I1_0, I2_0     | initial sizes of infected populations   |
| R1_0, R2_0     | initial sizes of immune populations   |
| object         | either the name of the model to simulate <i>or</i> a previously computed ‘gpsim’ object |

Value

runS2I2R2 and continueS2I2R2 return objects of class ‘gpsim’ with ‘model’ attribute “S2I2R2”.

See Also

More example genealogy processes: [lbdp](#), [moran](#), [seir](#), [si2r](#), [siir](#), [simulate\(\)](#), [sir](#), [twospecies](#)

---

|      |   |
|------|---|
| seir | <i>Classical susceptible-exposed-infected-recovered model</i> |
|------|---|

---

Description

The population is structured by infection progression.



**Usage**

```
runSEIR(  
  time,  
  t0 = 0,  
  Beta = 4,  
  sigma = 1,  
  gamma = 1,  
  psi = 1,  
  omega = 0,  
  S0 = 100,  
  E0 = 5,  
  I0 = 5,  
  R0 = 0  
)  
  
runSEIRS(  
  time,  
  t0 = 0,  
  Beta = 4,  
  sigma = 1,  
  gamma = 1,  
  psi = 1,  
  omega = 0,  
  S0 = 100,  
  E0 = 5,  
  I0 = 5,  
  R0 = 0  
)  
  
continueSEIR(  
  object,  
  time,  
  Beta = NA,  
  sigma = NA,  
  gamma = NA,  
  psi = NA,  
  omega = NA  
)  
  
continueSEIRS(  
  object,  
  time,  
  Beta = NA,  
  sigma = NA,  
  gamma = NA,  
  psi = NA,  
  omega = NA  
)
```

```
seirs_pomp(x, Beta, sigma, gamma, psi, omega = 0, S0, E0, I0, R0)
```

### Arguments

|                |   |
|----------------|---|
| time           | final time  |
| t0             | initial time  |
| Beta           | transmission rate   |
| sigma          | progression rate  |
| gamma          | recovery rate   |
| psi            | per capita sampling rate  |
| omega          | rate of waning of immunity  |
| S0, E0, I0, R0 | initial sizes of S, E, I, R compartments, respectively.                                 |
| object         | either the name of the model to simulate <i>or</i> a previously computed ‘gpsim’ object |
| x              | genealogy in <b>phylopomp</b> format.   |

### Details

seirs\_pomp constructs a ‘pomp’ object containing a given set of data and an SEIRS model.

### Value

runSEIR and continueSEIR return objects of class ‘gpsim’ with ‘model’ attribute “SEIR”.

seirs\_pomp returns a ‘pomp’ object.

### References

A. A. King, Q. Lin, and E. L. Ionides. Exact phylodynamic likelihood via structured Markov genealogy processes. *arXiv* 2405.17032, 2024. doi:10.48550/arxiv.2405.17032.

### See Also

More example genealogy processes: [lbdp](#), [moran](#), [s2i2r2](#), [si2r](#), [siir](#), [simulate\(\)](#), [sir](#), [twospecies](#)

### Examples

```
simulate("SEIR",Beta=2,sigma=2,gamma=1,psi=2,S0=1000,I0=5,time=5) |>
  simulate(Beta=5,gamma=2,time=10,psi=3) |>
  plot()

runSEIR(Beta=3,gamma=1,psi=2,S0=20,I0=5,R0=0,time=5,t0=-1) |>
  plot(points=TRUE,obscure=FALSE)

runSEIR(Beta=3,gamma=0.1,psi=0.2,S0=100,I0=5,R0=0,time=2,t0=0) -> x
plot_grid(plotlist=list(plot(x,points=TRUE),diagram(x)),
  ncol=1,rel_heights=c(4,1))
```

```
simulate("SEIR",sigma=1,omega=1,time=20,I0=4) |> plot(obscure=FALSE)

simulate("SEIR",sigma=1,omega=1,time=20,I0=4) |>
  lineages(obscure=FALSE) |>
  plot()
```

---

si2r

*Two-deme model of superspreading*


---

## Description

Deme 2 consists of "superspreaders" who engender clusters of infection in "superspreading events".

## Usage

```
runSI2R(
  time,
  t0 = 0,
  Beta = 5,
  mu = 5,
  gamma = 1,
  omega = 0,
  psi1 = 1,
  psi2 = 0,
  sigma12 = 1,
  sigma21 = 3,
  S0 = 500,
  I0 = 10,
  R0 = 0
)

continueSI2R(
  object,
  time,
  Beta = NA,
  mu = NA,
  gamma = NA,
  omega = NA,
  psi1 = NA,
  psi2 = NA,
  sigma12 = NA,
  sigma21 = NA
)
```

## Arguments

time                      final time

|                  |   |
|------------------|---|
| t0               | initial time  |
| Beta             | transmission rate   |
| mu               | mean superspreading-event cluster size  |
| gamma            | recovery rate   |
| omega            | rate of waning of immunity  |
| psi1, psi2       | sampling rates for demes 1 and 2, respectively  |
| sigma12, sigma21 | movement rates from deme 1 to 2 and 2 to 1, respectively                                |
| S0               | initial size of susceptible population  |
| I0               | initial size of I1 population (I2 = 0 at t = 0)   |
| R0               | initial size of recovered population  |
| object           | either the name of the model to simulate <i>or</i> a previously computed ‘gpsim’ object |

## Details

Superspreaders (deme 2) behave differently than ordinary infections: transmission events occur at the same rate (Beta), but at each event, a superspreader infects  $N$  individuals, where

$$N \sim 1 + \text{Geometric}(1/\mu).$$

Thus, assuming susceptibles are not limiting, the mean number of infections resulting from a superspreading event is  $\mu$  and the variance in this number is  $\mu^2 - \mu$ . If susceptibles are limiting, i.e., if the number of susceptibles is not greater than  $N$ , then all remaining susceptibles are infected.

## Value

runSI2R and continueSI2R return objects of class ‘gpsim’ with ‘model’ attribute “SI2R”.

## See Also

More example genealogy processes: [lbdp](#), [moran](#), [s2i2r2](#), [seir](#), [siir](#), [simulate\(\)](#), [sir](#), [twospecies](#)

## Examples

```
simulate("SI2R",time=1) |>
  plot(obscure=FALSE)

runSI2R(Beta=10,S0=2000,time=1,psi1=0) |>
  simulate(time=2,psi1=1) |>
  plot(points=TRUE,obscure=FALSE)

simulate("SI2R",time=5) |>
  lineages() |>
  plot()

simulate("SI2R",time=2) |>
  diagram(m=30)
```

```

simulate("SI2R",time=20,omega=0.2,mu=20) -> x
plot_grid(
  x |> plot(obscure=FALSE),
  x |> lineages(obscure=FALSE) |> plot(),
  ncol=1,
  align="v",axis="b"
)

```

---

siir

*Two-strain SIR model.*


---

## Description

Two distinct pathogen strains compete for susceptibles.

## Usage

```

runSIIR(
  time,
  t0 = 0,
  Beta1 = 5,
  Beta2 = 5,
  gamma = 1,
  psi1 = 1,
  psi2 = 0,
  sigma12 = 0,
  sigma21 = 0,
  omega = 0,
  S0 = 500,
  I1_0 = 10,
  I2_0 = 10,
  R0 = 0
)

continueSIIR(
  object,
  time,
  Beta1 = NA,
  Beta2 = NA,
  gamma = NA,
  psi1 = NA,
  psi2 = NA,
  sigma12 = NA,
  sigma21 = NA,
  omega = NA
)

```

**Arguments**

|                  |   |
|------------------|---|
| time             | final time  |
| t0               | initial time  |
| Beta1, Beta2     | transmission rates from each of the infectious classes.                                 |
| gamma            | recovery rate.  |
| psi1, psi2       | sampling rates.   |
| sigma12, sigma21 | movement rates from deme 1 to 2 and 2 to 1, respectively                                |
| omega            | rate of loss of immunity  |
| S0               | initial size of susceptible population.   |
| I1_0             | initial size of I2 population.  |
| I2_0             | initial size of I2 population.  |
| R0               | initial size of recovered population.   |
| object           | either the name of the model to simulate <i>or</i> a previously computed 'gpsim' object |

**Value**

runSIIR and continueSIIR return objects of class 'gpsim' with 'model' attribute "SIIR".

**See Also**

More example genealogy processes: [lbdp](#), [moran](#), [s2i2r2](#), [seir](#), [si2r](#), [simulate\(\)](#), [sir](#), [twospecies](#)

**Examples**

```
simulate("SIIR",time=3,psi1=1,psi2=0) |>
  simulate(Beta1=2,gamma=2,time=10,psi1=10,psi2=1) |>
  plot()

runSIIR(Beta1=10,Beta2=8,
  S0=200,I1_0=10,I2_0=8,R0=0,time=0,t0=-1) |>
  simulate(psi1=10,time=2) |>
  plot(points=TRUE,obscure=FALSE)

simulate("SIIR",Beta1=2,Beta2=50,gamma=1,psi1=2,
  S0=300,I1_0=20,I2_0=2,time=5) |>
  lineages() |>
  plot()
```

---

|          |                 |
|----------|-----------------|
| simulate | <i>simulate</i> |
|----------|-----------------|

---

## Description

Simulate Markov genealogy processes

## Usage

```
simulate(object, ...)

## Default S3 method:
simulate(object, ...)

## S3 method for class 'character'
simulate(object, time, ...)

## S3 method for class 'gpsim'
simulate(object, time, ...)
```

## Arguments

|        |   |
|--------|---|
| object | either the name of the model to simulate <i>or</i> a previously computed ‘gpsim’ object |
| ...    | additional arguments to the model-specific simulation functions                         |
| time   | end timepoint of simulation   |

## Details

When object is of class ‘gpsim’, i.e., the result of a genealogy-process simulation, `simulate` acts to continue the simulation to a later timepoint. Note that, one cannot change initial conditions or `t0` when continuing a simulation.

## Value

An object of ‘gpsim’ class.

## References

A. A. King, Q. Lin, and E. L. Ionides. Exact phylodynamic likelihood via structured Markov genealogy processes. *arXiv* 2405.17032, 2024. doi:[10.48550/arxiv.2405.17032](https://doi.org/10.48550/arxiv.2405.17032).

A. A. King, Q. Lin, and E. L. Ionides. Markov genealogy processes. *Theoretical Population Biology* **143**, 77–91, 2022. doi:[10.1016/j.tpb.2021.11.003](https://doi.org/10.1016/j.tpb.2021.11.003).

## See Also

More example genealogy processes: [lbdp](#), [moran](#), [s2i2r2](#), [seir](#), [si2r](#), [siir](#), [sir](#), [twospecies](#)

---

**sir***Classical susceptible-infected-recovered model*

---

**Description**

A single, unstructured population of hosts.

**Usage**

```
runSIR(  
  time,  
  t0 = 0,  
  Beta = 2,  
  gamma = 1,  
  psi = 1,  
  omega = 0,  
  S0 = 100,  
  I0 = 2,  
  R0 = 0  
)
```

```
runSIRS(  
  time,  
  t0 = 0,  
  Beta = 2,  
  gamma = 1,  
  psi = 1,  
  omega = 0,  
  S0 = 100,  
  I0 = 2,  
  R0 = 0  
)
```

```
continueSIR(object, time, Beta = NA, gamma = NA, psi = NA, omega = NA)
```

```
runSIRS(  
  time,  
  t0 = 0,  
  Beta = 2,  
  gamma = 1,  
  psi = 1,  
  omega = 0,  
  S0 = 100,  
  I0 = 2,  
  R0 = 0  
)
```



```
continueSIRS(object, time, Beta = NA, gamma = NA, psi = NA, omega = NA)
```

```
sir_pomp(x, Beta, gamma, psi, omega = 0, S0, I0, R0, t0 = 0)
```

```
sirs_pomp(x, Beta, gamma, psi, omega = 0, S0, I0, R0, t0 = 0)
```

### Arguments

|            |   |
|------------|---|
| time       | final time  |
| t0         | initial time  |
| Beta       | transmission rate.  |
| gamma      | recovery rate.  |
| psi        | sampling rate.  |
| omega      | immunity waning rate  |
| S0, I0, R0 | initial sizes of susceptible, infected, and recovered populations, respectively.        |
| object     | either the name of the model to simulate <i>or</i> a previously computed ‘gpsim’ object |
| x          | genealogy in <b>phylopomp</b> format (i.e., an object that inherits from ‘gpgen’).      |

### Details

sir\_pomp constructs a ‘pomp’ object containing a given set of data and a SIR model.

### Value

runSIR and continueSIR return objects of class ‘gpsim’ with ‘model’ attribute “SIR”.

sir\_pomp and sirs\_pomp return ‘pomp’ objects.

### References

A. A. King, Q. Lin, and E. L. Ionides. Exact phylodynamic likelihood via structured Markov genealogy processes. *arXiv* 2405.17032, 2024. doi:10.48550/arxiv.2405.17032.

A. A. King, Q. Lin, and E. L. Ionides. Markov genealogy processes. *Theoretical Population Biology* 143, 77–91, 2022. doi:10.1016/j.tpb.2021.11.003.

### See Also

More example genealogy processes: [lbdp](#), [moran](#), [s2i2r2](#), [seir](#), [si2r](#), [siir](#), [simulate\(\)](#), [twospecies](#)

### Examples

```
simulate("SIR",Beta=2,gamma=1,psi=2,S0=1000,I0=5,time=5) |>
  simulate(Beta=5,gamma=2,time=10,psi=3) |>
  plot()
```

```
runSIR(Beta=3,gamma=1,psi=2,S0=20,I0=5,R0=0,time=5,t0=-1) |>
  plot(points=TRUE)
```

```
runSIR(Beta=3,gamma=0.1,psi=0.2,S0=100,I0=5,R0=0,time=2,t0=0) -> x
plot_grid(plotlist=list(plot(x,points=TRUE),diagram(x)),
  ncol=1,rel_heights=c(4,1))

simulate("SIRS",omega=1,time=20,I0=4) |> plot()
simulate("SIRS",omega=1,time=20,I0=4) |> lineages() |> plot()
```

treeplot

*Fancy tree plotter***Description**

Plots a genealogical tree.

**Usage**

```
## S3 method for class 'gpgen'
plot(x, ..., time, t0, prune = TRUE, obscure = TRUE)

treeplot(
  tree,
  time = NULL,
  t0 = 0,
  ladderize = TRUE,
  points = FALSE,
  ...,
  palette = scales::hue_pal(l = 30, h = c(220, 580))
)
```

**Arguments**

|           |  |
|-----------|--|
| x         | object of class ‘gpgen’  |
| ...       | plot passes extra arguments to <a href="#">treeplot</a> . treeplot passes extra arguments to <a href="#">theme</a> .   |
| time      | numeric; time of the genealogy.  |
| t0        | numeric; time of the root.   |
| prune     | logical; prune the genealogy?  |
| obscure   | logical; obscure the demes?  |
| tree      | character; tree representation in Newick format.   |
| ladderize | Ladderize?   |
| points    | Show nodes and tips?   |
| palette   | color palette for branches. This can be furnished either as a function or a vector of colors. If this is a function, it should take a single integer argument, the number of colors required. If it is a vector, it should have at least as many elements as there are demes in the genealogy. |

**Value**

A printable ggplot object.

**Examples**

## Not run:

```
library(ggplot2)
library(phylopomp)
times <- seq(from=0,to=8,by=0.1)[-1]

png_files <- sprintf(
  file.path(tempdir(),"frame%05d.png"),
  seq_len(2*length(times))
)

pb <- utils::txtProgressBar(0,2*length(times),0,style=3)
x <- simulate("SIIR",time=0,Beta1=5,Beta2=10,gamma=1,omega=0.5,
  psi1=0.2,psi2=0.1,sigma12=1,sigma21=1,S0=200,I1_0=3,I2_0=2)

img <- 1
for (k in seq.int(from=1,to=length(times),by=1)) {
  x <- simulate(x,time=times[k])
  ggsave(
    filename=png_files[img],
    plot=plot(
      x, t0=0, time=max(times),
      points=FALSE, prune=FALSE, obscure=FALSE,
      palette=c("#ffcb05", "#dddddd"),
      axis.line=element_line(color="white"),
      axis.ticks=element_line(color="white"),
      axis.text=element_blank(),
      plot.background=element_rect(fill=NA,color=NA),
      panel.background=element_rect(fill=NA,color=NA)
    ),
    device="png",dpi=300,
    height=2,width=3,units="in"
  )
  setTxtProgressBar(pb,img)
  img <- img+1
}

for (k in seq.int(from=length(times),to=1,by=-1)) {
  x <- curtail(x,time=times[k])
  ggsave(
    filename=png_files[img],
    plot=plot(
      x, t0=0, time=max(times),
      points=FALSE, prune=FALSE, obscure=FALSE,
      palette=c("#ffcb05", "#dddddd"),
      axis.line=element_line(color="white"),
      axis.ticks=element_line(color="white"),
```

```

        axis.text=element_blank(),
        plot.background=element_rect(fill=NA,color=NA),
        panel.background=element_rect(fill=NA,color=NA)
    ),
    device="png",dpi=300,
    height=2,width=3,units="in"
)
setTxtProgressBar(pb,img)
img <- img+1
}

library(gifski)
gif_file <- "movie1.gif"
gifski(png_files,gif_file,delay=0.08,loop=TRUE)
unlink(png_files)

## End(Not run)

```

twospecies

*Two-host infection model with waning, immigration, demography, and spillover. Hosts are culled upon sampling with a given probability.*

## Description

The population is structured by infection progression and host species.

## Usage

```

runTwoSpecies(
  time,
  t0 = 0,
  Beta11 = 4,
  Beta12 = 0,
  Beta21 = 0,
  Beta22 = 4,
  gamma1 = 1,
  gamma2 = 1,
  psi1 = 1,
  psi2 = 0,
  c1 = 1,
  c2 = 1,
  omega1 = 0,
  omega2 = 0,
  b1 = 0,
  b2 = 0,
  d1 = 0,
  d2 = 0,

```

```
    iota1 = 0,  
    iota2 = 0,  
    S1_0 = 100,  
    S2_0 = 100,  
    I1_0 = 0,  
    I2_0 = 10,  
    R1_0 = 0,  
    R2_0 = 0  
  )  
  
  continueTwoSpecies(  
    object,  
    time,  
    Beta11 = NA,  
    Beta12 = NA,  
    Beta21 = NA,  
    Beta22 = NA,  
    gamma1 = NA,  
    gamma2 = NA,  
    psi1 = NA,  
    psi2 = NA,  
    c1 = NA,  
    c2 = NA,  
    omega1 = NA,  
    omega2 = NA,  
    b1 = NA,  
    b2 = NA,  
    d1 = NA,  
    d2 = NA,  
    iota1 = NA,  
    iota2 = NA  
  )  
  
  twospecies_pomp(  
    x,  
    Beta11,  
    Beta12,  
    Beta21,  
    Beta22,  
    gamma1,  
    gamma2,  
    psi1,  
    psi2,  
    c1,  
    c2,  
    omega1,  
    omega2,  
    b1,
```

```

    b2,
    d1,
    d2,
    S1_0,
    S2_0,
    I1_0,
    I2_0,
    R1_0,
    R2_0
  )

```

### Arguments

|        |   |
|--------|---|
| time   | final time  |
| t0     | initial time  |
| Beta11 | transmission rate within species 1  |
| Beta12 | transmission from species 2 to species 1  |
| Beta21 | transmission from species 1 to species 2  |
| Beta22 | transmission rate within species 2  |
| gamma1 | species 1 recovery rate   |
| gamma2 | species 2 recovery rate   |
| psi1   | per capita sampling rate for species 1  |
| psi2   | per capita sampling rate for species 2  |
| c1     | probability that a sampled (positive) host of species 1 is culled                       |
| c2     | probability that a sampled (positive) host of species 2 is culled                       |
| omega1 | rate of waning of immunity for species 1  |
| omega2 | rate of waning of immunity for species 2  |
| b1     | per capita birth rate for species 1   |
| b2     | per capita birth rate for species 2   |
| d1     | per capita death rate for species 1   |
| d2     | per capita death rate for species 2   |
| iota1  | imported infections for species 1   |
| iota2  | imported infections for species 2   |
| S1_0   | initial size of species 1 susceptible population  |
| S2_0   | initial size of species 2 susceptible population  |
| I1_0   | initial size of species 1 infected population   |
| I2_0   | initial size of species 2 infected population   |
| R1_0   | initial size of species 1 immune population   |
| R2_0   | initial size of species 2 immune population   |
| object | either the name of the model to simulate <i>or</i> a previously computed ‘gpsim’ object |
| x      | genealogy in <b>phylopomp</b> format.   |

**Details**

`twospecies_pomp` constructs a ‘pomp’ object containing a given set of data and a `TwoSpecies` model. Note that, for the moment, `twospecies_pomp` assumes that there is no importation of infection into the populations (i.e., `iota1 = iota2 = 0`).

**Value**

`runTwoSpecies` and `continueTwoSpecies` return objects of class ‘gpsim’ with ‘model’ attribute “TwoSpecies”.

`twospecies_pomp` returns a ‘pomp’ object.

**See Also**

More example genealogy processes: [lbdp](#), [moran](#), [s2i2r2](#), [seir](#), [si2r](#), [siir](#), [simulate\(\)](#), [sir](#)

---

`yaml`*YAML output*

---

**Description**

Human- and machine-readable description.

**Usage**

```
yaml(object)
```

**Arguments**

`object`                  gpsim object.

**Value**

A string in YAML format, with class “gpyaml”.

**Examples**

```
simulate("SIIR",time=1) |> yaml()
```

# Index

## \* Genealogy processes

- lbdp, 8
- moran, 11
- s2i2r2, 15
- seir, 16
- si2r, 19
- siir, 21
- simulate, 23
- sir, 24
- twospecies, 28

## \* internals

- genealogy diagram internals, 5

## \* internal

- as.data.frame, 2
- genealogy diagram internals, 5
- getInfo, 7
- reexports, 14

%dopar% (reexports), 14

%dopar%, 14

as.data.frame, 2

as.yaml, 14

as.yaml (reexports), 14

bake, 14

bake (reexports), 14

ballGrob (genealogy diagram internals),  
5

continueLBDP (lbdp), 8

continueMoran (moran), 11

continueS2I2R2 (s2i2r2), 15

continueSEIR (seir), 16

continueSEIRS (seir), 16

continueSI2R (si2r), 19

continueSIIR (siir), 21

continueSIR (sir), 24

continueSIRS (sir), 24

continueTwoSpecies (twospecies), 28

curtail, 2

diagram, 3

drawDetails.ballGrob (genealogy  
diagram internals), 5

drawDetails.resizingTextGrob  
(genealogy diagram internals),  
5

foreach, 14

foreach (reexports), 14

freeze, 14

freeze (reexports), 14

gendat, 4

geneal, 5

genealogy diagram internals, 5

genealogyGrob (genealogy diagram  
internals), 5

getInfo, 7

LBDP (lbdp), 8

lbdp, 8, 12, 16, 18, 20, 22, 23, 25, 31

lbdp\_exact (lbdp), 8

lbdp\_pomp (lbdp), 8

lineages, 10

Moran (moran), 11

moran, 9, 11, 16, 18, 20, 22, 23, 25, 31

moran\_exact (moran), 11

newick, 12

nodeGrob (genealogy diagram internals),  
5

parse\_newick, 13

phylopomp, 14

phylopomp, package (phylopomp), 14

phylopomp-package (phylopomp), 14

plot, 11

plot.gp (treeplot), 26

plot.gp (lineages), 10

plot\_grid, 14



plot\_grid (reexports), 14  
 pocketGrob (genealogy diagram  
     internals), 5  
 postDrawDetails.ballGrob (genealogy  
     diagram internals), 5  
 postDrawDetails.resizingTextGrob  
     (genealogy diagram internals),  
     5  
 preDrawDetails.ballGrob (genealogy  
     diagram internals), 5  
 preDrawDetails.resizingTextGrob  
     (genealogy diagram internals),  
     5  
 print.gpdia (diagram), 3  
 print.gpgen (genealogy diagram  
     internals), 5  
 print.gpsim (genealogy diagram  
     internals), 5  
 print.gpyaml (genealogy diagram  
     internals), 5  
  
 read\_yaml, 14  
 read\_yaml (reexports), 14  
 reexports, 14  
 registerDoSEQ, 14  
 registerDoSEQ (reexports), 14  
 resizingTextGrob (genealogy diagram  
     internals), 5  
 runLBDP (lbdp), 8  
 runMoran (moran), 11  
 runS2I2R2 (s2i2r2), 15  
 runSEIR (seir), 16  
 runSEIRS (seir), 16  
 runSI2R (si2r), 19  
 runSIIR (siir), 21  
 runSIR (sir), 24  
 runSIRS (sir), 24  
 runTwoSpecies (twospecies), 28  
  
 S2I2R2 (s2i2r2), 15  
 s2i2r2, 9, 12, 15, 18, 20, 22, 23, 25, 31  
 SEIR (seir), 16  
 seir, 9, 12, 16, 16, 20, 22, 23, 25, 31  
 seirs\_pomp (seir), 16  
 SI2R (si2r), 19  
 si2r, 9, 12, 16, 18, 19, 22, 23, 25, 31  
 SIIR (siir), 21  
 siir, 9, 12, 16, 18, 20, 21, 23, 25, 31  
 simulate, 9, 12, 16, 18, 20, 22, 23, 25, 31  
  
 SIR (sir), 24  
 sir, 9, 12, 16, 18, 20, 22, 23, 24, 31  
 sir\_pomp (sir), 24  
 SIRS (sir), 24  
 sirs\_pomp (sir), 24  
 stew, 14  
 stew (reexports), 14  
  
 textGrob, 6  
 theme, 10, 26  
 tibble, 8, 11  
 treeplot, 26, 26  
 TwoSpecies (twospecies), 28  
 twospecies, 9, 12, 16, 18, 20, 22, 23, 25, 28  
 twospecies\_pomp (twospecies), 28  
  
 viewport, 14  
 viewport (reexports), 14  
  
 yaml, 31