Package 'SCIFER'

June 12, 2024

Title Package to compute neutral mutation accumulation during drift and selection

Version 0.0.0.9000

Description Package to compute neutral mutation accumulation during drift and selection.

License `use_mit_license()`

Encoding UTF-8

Roxygen list(markdown=TRUE)

RoxygenNote 7.2.3

Depends R (>= 2.10)

LazyData true

Suggests knitr,
rmarkdown,
ggplot2,
HDInterval,
ggranges,
ggridges

VignetteBuilder knitr

R topics documented:

`.approximate.delta` *Approximate deltas to approximate non-linear decline with a linear b-d-process parametrized by the number of death events*

Description

Approximate deltas to approximate non-linear decline with a linear b-d-process parametrized by the number of death events

Usage

```
.approximate.delta(lambda, N, t, D)
```

Arguments

`lambda`	the division rate
`N`	the clone size at the start of contraction
`t`	the time span
`D`	the number of death events

Value

the death rate yielding the same number of death events if modeling exponential decay

.clonal.combinations *Clonal combinations*

Description

Clonal combinations

Usage

```
.clonal.combinations(clone.ids)
```

Arguments

clone.ids	IDs of daughter clones from the same mother

Value

All clonal combinations in which a mutation acquired in the mother can end up.

.clonal_dynamics *Dynamics of normal cells and j selected clones*

Description

Dynamics of normal cells and j selected clones

Usage

```
.clonal_dynamics(N, init, lambda, delta, s, t)
```

Arguments

N	the carrying capacity
init	vector of length j with the initial condition of the system (number of cells per clone)
lambda	cell division rate
delta	differentiation rate
s	vector with selective advantage associated with the j-th driver. Selection is modeled as a reduction of the differentiation rate, so 0 <= s <= 1
t	the time point of evaluation

Value

The system state at time t

`.compute_actual_size` *Compute the final size of the clones from the input parameters*

Description

Compute the final size of the clones from the input parameters

Usage

```
.compute_actual_size(
  t.s,
  mother.daughter,
  N,
  lambda.ss,
  delta.ss,
  lambda.exp,
  delta.exp,
  size,
  t.end
)
```

Arguments

`t.s`	vector of length j with the time points at which the selected advantages were introduced
`mother.daughter`	mother-daughter relationships. Matrix with 2 columns encoding mother and daughter for each pair
`N`	the compartment size
`lambda.ss`	cell division rate during homeostasis
`delta.ss`	differentiation rate during homeostasis
`lambda.exp`	division rate during initial expansion
`delta.exp`	loss rate during initial expansion
`size`	vector of length j with the input size of the selcted clones. Using exponential growth approximation, the selective advantage, `s` will be computed from `size`. The actual clone sizes will then be computed using the values of `s` and a clonal competition model.
`t.end`	the time point of evaluation

Value

A vector reporting the system state at the final time point for each clone

.differentiation.s.p *Cell differentiation*

Description

Simulate differentiation of a stem cell into a progenitor cell

Usage

```
.differentiation.s.p(tree, nr = 1)
```

Arguments

tree	object of class phylo; the current tree. Needs the following additional list elements: tip.class, specifying the cell type and sel.adv., specifying the selective advantage of each tip
nr	number of reactions to simulate; defaults to 1

Value

the updated tree

.division.s.p *Cell division*

Description

Simulate division of a cell together with acquisition of neutral and driver mutations

Usage

```
.division.s.p(
  cell.type,
  tree,
  mut.rate.D = 0,
  s.shape = 1.5,
  s.rate = 35,
  mutation.mode,
  mut.rate,
  symmetric = T,
  nr = 1
)
```

Arguments

cell.type	the cell type that is to divide (1, stem cell; 2, progenitor cell)
tree	object of class phylo; the current tree. Needs the following additional list elements: tip.class, specifying the cell type and sel.adv., specifying the selective advantage of each tip
mut.rate.D	integer; driver mutation rate per cell division
s.shape	shape parameter of the gamma distribution from which the selective advantage of a new driver is drawn
s.rate	rate parameter of the gamma distribution from which the selective advantage of a new driver is drawn
mutation.mode	should the number of new mutations be "constant" or "Binomial"ly distributed?
mut.rate	average number of neutral mutations per daughter cell and division
symmetric	logical, is the division a symmetric division (2 new daughter cells) or an asymmetric division (1 daughter cell differentiates); defaults to T
nr	number of reactions to simulate; defaults to 1

Value

the updated tree

.driver.mutation.acquisition

Driver mutation acquisition

Description

Simulate the acquisition of new driver mutations in both daughter cells

Usage

```
.driver.mutation.acquisition(mut.rate.D, s.shape, s.rate)
```

Arguments

mut.rate.D	integer; driver mutation rate per cell division
s.shape	shape parameter of the gamma distribution from which the selective advantage of a new driver is drawn
s.rate	rate parameter of the gamma distribution from which the selective advantage of a new driver is drawn

Value

the additional selective advantage acquired during the cell division

`.forward_dynamics` *Forward dynamics of j clones*

Description

Forward dynamics of j clones

Usage

```
.forward_dynamics(
  N,
  init.cells,
  lambda.ss,
  delta.ss,
  lambda.exp,
  delta.exp,
  s,
  t.s,
  mother.daughter,
  t,
  resolution = 0.01
)
```

Arguments

`N`	the compartment size
`init.cells`	vector with the initial condition of the system (number of cells per clone)
`lambda.ss`	cell division rate during homeostasis
`delta.ss`	differentiation rate during homeostasis
`lambda.exp`	division rate during initial expansion
`delta.exp`	loss rate during initial expansion
`s`	vector with selective advantages associated with the j-th driver. Selection is modeled as a reduction of the differentiation rate, so 0 <= s <= 1
`t.s`	vector of length j with the time points at which the selected advantages were introduced
`mother.daughter`	mother-daughter relationships. Matrix with 2 columns encoding mother and daughter for each pair
`t`	the time point of evaluation
`resolution`	the time resolution of the simulation

Value

A data.frame reporting the system state in the following order: time, cell count for each clone

.get.progeny *Get all subclonal descendants from a given clone*

Description

Get all subclonal descendants from a given clone

Usage

```
.get.progeny(mother.daughter, id)
```

Arguments

mother.daughter	mother-daughter relationships. Matrix with 3 columns encoding mother, daughter and birth-compartment for each pair
id	IDs of the clone of interest

Value

A vector with the progeny IDs.

.initialize.sim.s.p *System initialization*

Description

This function initializes the simulation

Usage

```
.initialize.sim.s.p()
```

Arguments

N	integer; number of stem cells at homeostasis
NP	integer; number of progenitor cells at homeostasis
mut.rate	integer; number of mutations per cell division and daughter cell
time.max	time of simulation
parms.steady	named vector of steady state parameters; must contain "lambda.s"/"lambda.p" the stem and progenitor division rate; "delta.s"/"delta.p" the stem and progenitor loss rate and "alpha.s" the stem cell differentiation rate

Value

returns the state list; initialized with a single stem cell and a single mutation

.loss *Cell loss*

Description

Simulate loss of a stem cell or a progenitor cell

Usage

```
.loss(cell.type, tree, nr = 1)
```

Arguments

cell.type	integer; which cell type is dividing? 1, stem cell, 2, progenitor cell
tree	object of class phylo; the current tree. Needs the following additional list elements: tip.class, specifying the cell type and sel.adv., specifying the selective advantage of each tip
nr	number of reactions to simulate; defaults to 1

Value

the updated tree

.mutation.acquisition *Acquisition of neutral mutations*

Description

This function simulates the acquisition of new mutations

Usage

```
.mutation.acquisition(mut.rate, mutation.mode = "Binomial")
```

Arguments

mut.rate	integer; average number of mutations per division and daughter cell
mutation.mode	character vector; should the number be based on a binomial distribution ("Binomial") or should a constant number be introduced ("constant"); defaults to "Binomial".

Value

the number of new mutations per daughter cell

.props.2c *Propensity function for 2 compartments ("S"tem cells and "P"rogenitors)*

Description

This function computes the propensities for a linear, hierarchical system of cell division, death and differentiation

Usage

```
.props.2c(cell.count, parms)
```

Arguments

cell.count	a vector with cell numbers (S, P)
parms	a named parameter vector containing the rates of cell division ("lambda.s"/"lambda.p") and differentiation ("alpha.s") or loss ("delta.s"/"delta.p"), where "s" and "p" indicate stem and progenitor cells, respectively

Value

the propensities of the possible reactions

.simulate.sampling *Sample a random subset of a phylogenetic tree*

Description

Sample a random subset of a phylogenetic tree

Usage

```
.simulate.sampling(tree, sample.size)
```

Arguments

tree	object of class phylo; the current tree. Needs the following additional list elements: tip.class, specifying the cell type and sel.adv., specifying the selective advantage of each tip.
sample.size	the number of cells to be sampled

Value

the updated state list

.simulated.scWGS.data *Simulate sampling as in single-cell sequencing by simulating the actual sampling of cells (i.e. it's not the expected value but a sampling instance)*

Description

Simulate sampling as in single-cell sequencing by simulating the actual sampling of cells (i.e. it's not the expected value but a sampling instance)

Usage

```
.simulated.scWGS.data(
  clone.sizes,
  expected.mutations,
  ncells = 100,
  min.vaf = 0.05
)
```

Arguments

clone.sizes	vector of clone sizes at which cumulative mutation counts were measured
expected.mutations	expected number of mutations at each clone size
ncells	sequenced cells

Value

A vector of simulated VAFs

.simulated.wgs.data *Simulate read sampling in WGS by Binomial sampling*

Description

Simulate read sampling in WGS by Binomial sampling

Usage

```
.simulated.wgs.data(
  clone.sizes,
  expected.mutations,
  depth = 90,
  sensitivity = T,
  false.negative.per.vaf,
  min.vaf = 0.05
)
```

Arguments

clone.sizes	vector of clone sizes at which cumulative mutation counts were measured
expected.mutations	expected number of mutations at each clone size
depth	sequencing depth
sensitivity	logical, if sensitivity of sequencing method should be taken into account in addition to binomial noise. Requires a specification for false.negative.per.vaf.
false.negative.per.vaf	optional, a matrix with columns corresponding to the measured VAFs and rows corresponding to individual measurements of the false negative rate at this VAF in addition to binomial noise. Must be provided if sensitivity=T

Value

A vector of simulated VAFs

.subset.cell.type *Subset a phylogenetic tree on a particular cell type*

Description

Subset a phylogenetic tree on a particular cell type

Usage

```
.subset.cell.type(tree, cell.type)
```

Arguments

tree	object of class phylo; the current tree. Needs the following additional list elements: tip.class, specifying the cell type and sel.adv., specifying the selective advantage of each tip.
cell.type	cell type of interest

Value

the updated stat list

`density.a.b.exact` *Non-critical clone size distribution (exact).*

Description

Exact probability to grow from a clone of size "a" to a clone of size "b" within time "t" according to a non-critical birth-death process.

Usage

```
## S3 method for class 'a.b.exact'
density(lambda, delta, t, a, b)
```

Arguments

`lambda`	proliferation rate
`delta`	loss rate
`t`	time
`a`	clone size at t=0
`b`	clone size at t=t

Value

The probability that a clone of size a grows to size "b" within "t".

References

Bailey, NTJ (1964). The elements of stochastic processes with applications to the natural sciences, Wiley (New York).

Examples

```
density.a.b.exact(1, 0, 10, 1, 2)
```

`Extract.info.from.vcf` *Extracts information from a vcf file.*

Description

Extracts information from a vcf file.

Usage

```
Extract.info.from.vcf(
  vcf,
  info = "readcounts",
  type = "snvs",
  mutationcaller = "Strelka",
  tumor.col.mutect = 10,
  normal.col.mutect = 11,
  sample.col.mpileup = NA,
  tumor.id = NULL
)
```

Arguments

vcf	Mutation information in VCF format represented as a list (as returned by read.vcf from package bedR).
info	Variant information to be retrieved. Possible values are
	readcounts returns the number of reference and variant reads for each variant position.
	varCounts the number of variant reads (if mutationcaller is 'Strelka' or 'Manta').
	depth returns the sequencing depths for each variant position.
	VAF returns the variant allele frequency at each variant position.
	VAF.control returns the variant allele frequency in the germline control at each variant position.
	depth.control returns the sequencing depth in the germline control at each variant position
	AA_change returns the amino acid change at each variant position (only works if vcf-file had been annotated with annovar).
	cDNA_change returns the cDNA change at each variant position (only works if vcf-file had been annotated with annovar).
	Exon returns the exon targeted by each variant (only works if vcf-file had been annotated with annovar).
	Gene returns the gene targeted by each variant (only works if vcf-file had been annotated with annovar).
	annovar_function returns the consequence of the variant (e.g. "intronic"; only works if vcf-file had been annotated with annovar).
	exonic_function returns the exonic consequence of each variant (e.g. 'non-synonymous SNV'; only works if vcf-file had been annotated with annovar).
	svtype type of structural variant at each variant position; only works on output generated with Manta.
	somaticScore the somatic score computed by Manta for each variant; only works on output generated with Manta.
	End the end position of a structural variant; only works with output generated by Manta.
type	Specify the variant type ("snvs" or "indel", defaults to "snvs"). Ignored if mutationcaller = "Mutect2" or mutationcaller = "Manta".
mutationcaller	The mutation caller that generated the vcf-files. Must be either "Strelka", "Mutect2", "mpileup" or "Manta". Defaults to "Strelka".

`tumor.col.mutect`
: Column index or name of the tumor information when `mutationcaller="Mutect2"`.

`normal.col.mutect`
: Column index or name of the germline control information when `mutationcaller="Mutect2"`.

`tumor.id`
: The name of the tumor sample. Only used if vcf-file was generated with Manta.

Value

The requested information for each variant position.

`get_matrix_from_tree` *Get a binary mutation matrix from a phylogenetic tree*

Description

Get a binary mutation matrix from a phylogenetic tree

Usage

```
get_matrix_from_tree(tree)
```

Arguments

`tree`
: object of class `phylo`; the current tree. Needs the following additional list elements: `tip.class`, specifying the cell type and `sel.adv.`, specifying the selective advantage of each tip.

Value

a matrix with rows corresponding to mutations and columns to cells; entries are binaries of 0 or 1, indicating, respectively, absence or presence of the mutation.

`get_mutations_per_tip` *Compute the number of mutations per tip cell in a phylogenetic tree*

Description

Compute the number of mutations per tip cell in a phylogenetic tree

Usage

```
get_mutations_per_tip(tree)
```

Arguments

`tree`
: object of class `phylo`; the current tree. Needs the following additional list elements: `tip.class`, specifying the cell type and `sel.adv.`, specifying the selective advantage of each tip.

Value

a vector of mutation counts

get_vaf_from_tree — *Compute the VAF of a mutation in the population from a phylogenetic tree*

Description

Compute the VAF of a mutation in the population from a phylogenetic tree

Usage

```
get_vaf_from_tree(tree)
```

Arguments

tree	object of class phylo; the current tree. Needs the following additional list elements: `tip.class`, specifying the cell type and `sel.adv.`, specifying the selective advantage of each tip.

Value

a vector of VAFs

gillespie.sim.s.p — *Tree simulation*

Description

Simulate the phylogenetic tree of a physiological population that grows in 2 regimes: initial exponential expansion and subsequent homeostasis. The function can simuate stem cells only or stem and progenitor cells

Usage

```
gillespie.sim.s.p(
  parms.exp,
  parms.steady,
  time.max = 50,
  time.samples = c(0, 5, 25, 50),
  N = 1000,
  NP = 10000,
  report.at.f = NA,
  mut.rate = 3,
  mutation.mode = "Binomial",
  driver.mode = "random",
  t.driver = NA,
  mut.rate.D = 0,
  s.shape = 1.5,
  s.rate = 35,
  tau = 1
)
```

Arguments

parms.exp	expansion parameters, vector that must contain lambda.s (stem cell division rate), lambda.p (progenitor cell division rate), alpha.s (stem cell differentiation rate), delta.s (stem cell loss rate), delta.p (progenitor cell differentiation rate)
parms.steady	same as parms.exp, but for the homeostatic phase
time.max	maximal simulation time
time.samples	time points at which simulation results are stored
N	number of stem cells during homeostasis
NP	number of progenitor cells during homeostasis
report.at.f	frequency of a selected clone at which the simulation should be stopped. Defaults to NA - don't stop at a certain frequency.
mut.rate	average number of mutation per division and daughter cell
mutation.mode	should mutations be "constant" or "Binomial"ly distributed?
driver.mode	string; "fixed_time" if driver is acquired at a fixed time point, "random", if driver is randomly acquired governed by the parameters mu_D and s
t.driver	integer, the time point at which the driver is acquired if mode is "fixed_time"
mut.rate.D	integer; driver mutation rate per cell division
s.shape	shape parameter of the gamma distribution from which the selective advantage of a new driver is drawn. If the function is run with driver.mode="fixed", the selective advantage is not randomyl drawn but computed as s.shape/d.rate.
s.rate	rate parameter of the gamma distribution from which the selective advantage of a new driver is drawn
tau	tau leaping parameter: how many steps should be merged during homeostasis? Defaults to 1, no leaping.

Value

a list of state.lists at the desired time samples

---

histogram.drift *Computes the expected histogram of variants in a clone of interest at time t, given a VAF histogram at time zero*

Description

Computes the expected histogram of variants in a clone of interest at time t, given a VAF histogram at time zero

Usage

```
histogram.drift(
  lower.bins.1,
  n.muts,
  bin.p1 = 1,
  bin.p2,
```

```
  lower.bins.2,
  N,
  lambda,
  delta,
  t
)
```

Arguments

lower.bins.1	vector of bin sizes of the histogram at t0
n.muts	vector of mutation counts per bin at t0
bin.p1	vector of probabilities that the variants in each bin will remain in the clone of interest
bin.p2	as bin.p1 but for the upper border of the bin
lower.bins.2	vector of clone sizes at t
N	the number of cells in the system
lambda	the division rate
delta	the loss rate
t	the time point of evaluation

Value

The cumulative VAF histogram at time t

---

mutational.burden *Mutation accumulation during exponential expansion followed by homeostasis.*

Usage

```
mutational.burden(
  mu,
  N,
  lambda.exp,
  delta.exp,
  lambda.ss,
  t.end,
  b,
  accuracy.a = 0.05,
  phase = "both"
)
```

Arguments

mu	mutation rate per cell division
N	population size
lambda.exp	proliferation rate during expansion
delta.exp	loss rate during expansion

lambda.ss	proliferation and loss rate during homeostasis
t.end	end point (starting from homeostasis)
b	minimal clone size of interest. Number or vector.
accuracy.a	step size in which mutations accumulated during expansion are evaluated (evaluation runs between 5 and 100\
	\itemphasereturn variants from "both" phases, or from "expansion" or "homeostasis" only
	This function returns the approximate number of mutations in clones of at least b cells, by first computing the distribution at the transition time within intervals, then averaging the fate of each interval during homeostasis and adding newly acquired mutations.
	Mutation accumulation during exponential expansion followed by homeostasis.

---

mutational.burden.general *Mutation accumulation during exponential expansion followed by second phase of either expansion, homeostasis or decline*

Usage

```
mutational.burden.general(
  mu,
  N.1,
  lambda.1,
  delta.1,
  lambda.2,
  delta.2 = NULL,
  t.end,
  b,
  accuracy.a = 0.05,
  phase = "both"
)
```

Arguments

mu	mutation rate per cell division
N.1	population size after first phase
lambda.1	proliferation rate during initial expansion
delta.1	loss rate during initial expansion
lambda.2	proliferation rate during second phase
delta.2	loss rate during second phase
t.end	end point (starting after initial expansion)
b	minimal clone size of interest. Number or vector.
accuracy.a	step size in which mutations accumulated during expansion are evaluated (evaluation runs between 5 and 100\
	\itemphasereturn variants from "both" phases, or from "first" or "second" only

This function returns the approximate number of mutations in clones of at least b cells, by first computing the distribution at the transition time within intervals, then averaging the fate of each interval during homeostasis and adding newly acquired mutations.

Mutation accumulation during exponential expansion followed by second phase of either expansion, homeostasis or decline

---

`mutational.burden.multiclone` *Mutation accumulation during exponential expansion followed by homeostasis.*

Description

Mutation accumulation during exponential expansion followed by homeostasis.

Usage

```
mutational.burden.multiclone(
  mu,
  N,
  lambda.exp,
  delta.exp,
  lambda.ss,
  t.end,
  t.s,
  s,
  mother.daughter,
  b,
  min.clone.size = 0.05,
  accuracy.a = 0.05
)
```

Arguments

`mu`	mutation rate per cell division
`N`	population size
`lambda.exp`	proliferation rate during expansion
`delta.exp`	loss rate during expansion
`lambda.ss`	proliferation and loss rate during homeostasis
`t.end`	end point (starting from homeostasis)
`t.s`	vector of time points at which selective advantages are acquired.
`s`	vector of selective advantages associated with driver mutations
`mother.daughter`	a matrix containing the mother (1st column) - daughter (2nd column) relationships between the subclones
`b`	minimal clone size of interest. Number or vector.

Value

This function returns an approximation by first computing the distribution at the transition time within intervals, then averaging the fate of each interval during homeostasis and adding newly acquired mutations in a scenario with 2 nested clonal selections (clone starts growing at t.s >t.ss and grows with a selective advantage s; clone 2 starts. Returns the number of mutations present in at least b cells

---

`mutational.burden.selection.expansion`
Mutation accumulation during exponential expansion with clonal selection.

Description

Mutation accumulation during exponential expansion with clonal selection.

Usage

```
mutational.burden.selection.expansion(mu, lambda, delta, s, t.s, t.end, b)
```

Arguments

`mu`	mutation rate per cell division
`lambda`	proliferation rate
`delta`	loss rate
`s`	selective advantage
`t.s`	time point at which selective advantage is acquired.
`t.end`	end point
`b`	minimal clone size of interest. Number or vector.
`N`	population size

Value

This function returns an approximation by first computing the distribution at the transition time within intervals, then averaging the fate of each interval during homeostasis and adding newly acquired mutations in a scenario where a subpopulation is under positive selection. Returns the number of mutations present in at least b cells.

`mutational.burden.with.selection` *Mutation accumulation during exponential expansion followed by homeostasis with clonal selection.*

Usage

```
mutational.burden.with.selection(
  mu,
  N,
  lambda.exp,
  delta.exp,
  lambda.ss,
  t.end,
  t.s,
  s,
  b,
  accuracy.a = 0.05,
  min.clone.size = 0.05
)
```

Arguments

`mu`	mutation rate per cell division
`N`	population size
`lambda.exp`	proliferation rate during expansion
`delta.exp`	loss rate during expansion
`lambda.ss`	proliferation and loss rate during homeostasis
`t.end`	end point (starting from homeostasis)
`t.s`	time point at which selective advantage is acquired.
`s`	selective advantage
`b`	minimal clone size of interest. Number or vector.
`accuracy.a`	step size in which mutations accumulated during expansion are evaluated between 5 and 100\
	\itemmin.clone.sizethe lower detection limit for selected clones
	This function returns an approximation by first computing the distribution at the transition time within intervals, then averaging the fate of each interval during homeostasis and adding newly acquired mutations in a scenario where a subpopulation is under positive selection. Returns the number of mutations present in at least `b` cells.
	Mutation accumulation during exponential expansion followed by homeostasis with clonal selection.

`mutational.burden.with.selection.no.size.compensation` *Mutation accumulation during exponential expansion followed by a second phase with clonal selection.*

Usage

```
mutational.burden.with.selection.no.size.compensation(
  mu,
  N,
  lambda.exp,
  delta.exp,
  lambda.ss,
  t.end,
  t.s,
  s,
  b,
  accuracy.a = 0.05,
  min.clone.size = 0.05
)
```

Arguments

`mu`	mutation rate per cell division
`N`	population size during homeostasis in absence of CH
`lambda.exp`	proliferation rate during expansion
`delta.exp`	loss rate during expansion
`lambda.ss`	proliferation and loss rate during homeostasis
`t.end`	end point (starting from homeostasis)
`t.s`	time point at which selective advantage is acquired.
`s`	selective advantage
`b`	minimal clone size of interest. Number or vector.
`accuracy.a`	step size in which mutations accumulated during expansion are evaluated between 5 and 100\
	\itemmin.clone.sizethe lower detection limit for selected clones
	This function returns an approximation by first computing the distribution at the transition time within intervals, then averaging the fate of each interval during homeostasis and adding newly acquired mutations in a scenario where a subpopulation is under positive selection. Returns the number of mutations present in at least `b` cells.
	Mutation accumulation during exponential expansion followed by a second phase with clonal selection.

mutations.during.steady.state

Neutral mutation accumulation during steady state.

Description

Neutral mutation accumulation during steady state.

Usage

```
mutations.during.steady.state(lambda, N, mu, n.min, t.end)
```

Arguments

lambda	proliferation rate
N	population size
mu	mutation rate per cell division
n.min	minimal clone size
t.end	time at end point

Value

The number of mutations that were acquired during steady state and are present in at least `n.min` cells.

mutations.noncritical.bd

Mutation accumulation in a growing tissue

Description

Expected number of neutral mutations that are present in at least `n.min` cells at `t.end` in an exponentially growing or contracting tissue.

Usage

```
mutations.noncritical.bd(
  lambda,
  delta,
  t.end,
  mu,
  n.min,
  N0 = 1,
  N = N,
  mode = "approx"
)
```

Arguments

lambda	proliferation rate
delta	loss rate
t.end	time
mu	mutation rate per cell division
n.min	minimal clone size at t.end; can be a value or a vector
N0	initial population size
N	final population size
mode	if "approx" the sum is approximated by integration. If "exact" the sum is exactly computed for clone sizes between 1 and 10 but beyond that also approximated.

Details

The expected number of mutations present in at least n.min cells is computed as
$M(n_{\min}) = \sum_{n_{\min}}^{N} \mu\lambda \int_0^t e^{(\lambda-\delta)(t-t')} P(1, n_{min}, t-t')dt'$, which is approximated to
$M(n_{\min}) \approx \mu\lambda \int_0^t e^{(\lambda-\delta)(t-t')} \frac{P(1,N,t-t') - P(1,n_{\min},t-t')}{\log y(t-t')} dt'$,
where $y(t) = \frac{\lambda e^{(\lambda-\delta)t} - \lambda}{\lambda e^{(\lambda-\delta)t} - \delta}$, if mode=="approx" or if $n_{\min} \leq 10$

Value

The expected number of mutations present in at least n.min cells at t.end in an exponentially growing tissue. The function assumes that mutations are continuously acquired at a constant rate.

References

Bailey, NTJ (1964). The elements of stochastic processes with applications to the natural sciences, Wiley (New York).

---

p.a.b *Clone size distribution in a noncritical birth-death process (approximate).*

Description

Probability of a clone of size "a" to grow to size "b" within "t" according to a noncritical linear birth-death process.

Usage

```
p.a.b(lambda, delta, t, a, b, mode = "cumulative", approx = "highnumbers")
```

Arguments

lambda	proliferation rate
delta	loss rate
t	time
a	clone size at t=0
b	clone size at t=t
mode	"density" if density distribution is to be returned , "cumulative" if cumulative distribution is to be returned. Defaults to "cumulative"
approx	Approximation to be used. Defaults to "highnumbers"; i.e. the distribution is approximated with a gamma distribution if a and b are large.

Details

If approx="highnumbers", the function is approximated with a Γ-distribution if a+b>100 and mode="density" or if a+b>10 and mode="cumulative". The Γ-distribution is parametrized with $shape = \mu^2/\sigma, scale = \sigma/\mu$, where $\mu = ae^{(\lambda-\delta)t}, \sigma = a\frac{\lambda+\delta}{\lambda-\delta}e^{(\lambda-\delta)t}(e^{(\lambda-\delta)t} - 1)$

Value

The probability of growing from size a to size b within t. The Function switches between the exact solution and an approximate solution according to a parametrized gamma distribution.

References

Bailey, NTJ (1964). The elements of stochastic processes with applications to the natural sciences, Wiley (New York).

---

p.ss *Clone size distribution in a critical b-d process.*

Description

Function to compute the probability to grow from a to b in a critical b-d process using automatic switching between exact and approximate solution.

Usage

```
p.ss(lambda, a, b, t)
```

Arguments

lambda	proliferation rate
a	clone size at t=0
b	clone size at t=t; single value or vector
t	time

Details

The function automatically switches between the exact solution and an approximation with a parametrized Γ-distribution at a cutoff criterion of $a * p * (1 - p) >= 9 \& b * p * (1 - p) >= 9$

Value

The probability to grow from a to b within t

`p.ss.approx` *Approximate solution to grow from a clone of size a to a clone of size b within t in a critical birth-death process using gamma distribution*

Description

Approximate solution to grow from a clone of size a to a clone of size b within t in a critical birth-death process using gamma distribution

Usage

```
p.ss.approx(lambda, a, b, t, mode = "density")
```

Arguments

`lambda`	proliferation rate
`a`	clone size at t=0
`b`	clone size at t=t
`t`	time
`mode`	either 'density' if density distribution is to be returned or 'cumulative'.

Value

The approximate probability to grow from a to b within t.

`p.ss.exact` *Exact solution to grow from a clone of size a to a clone of size b within t in a critical birth-death process*

Description

Exact solution to grow from a clone of size a to a clone of size b within t in a critical birth-death process

Usage

```
p.ss.exact(lambda, a, b, t)
```

Arguments

`lambda`	proliferation rate
`a`	clone size at t=0
`b`	clone size at t=t
`t`	time

Value

The probability to growth from a to b within t.

`probability.this.combination` *Probability of a variant present in a given bin size ends up in a particular combination of selected daughters if a driver is acquired in a random cell of the mother clone*

Description

Probability of a variant present in a given bin size ends up in a particular combination of selected daughters if a driver is acquired in a random cell of the mother clone

Usage

```
probability.this.combination(
  bin.size,
  clone.size.mother,
  n.daughters.present,
  n.daughters.absent
)
```

Arguments

`bin.size`	vector of bin sizes the variant are present in
`clone.size.mother`	the size of the mother clone
`n.daughters.present`	the number of daughters the variant ends up in
`n.daughters.absent`	the number of daughters the variant does not end up in

Value

Computes the probability that a given variant that is present in `bin.size` cells of the mother clone ends up in `n.daughters.present` daughter clones, but not in the remaining `n.daughters.absent` if the cells giving rise to the selected daughters are randomly sampled.

`simulated.data` *Wrapper function to simulate sequencing either by bulk WGS or by scWGS*

Description

Wrapper function to simulate sequencing either by bulk WGS or by scWGS

Usage

```
simulated.data(
  seqtype,
  clone.sizes,
  expected.mutations,
  depth = 90,
  ncells = 100,
  sensitivity = T,
  false.negative.per.vaf,
  min.vaf = 0.05
)
```

Arguments

`seqtype`	string, specifying the sequencing method. Must be either "bulk" or "sc"
`clone.sizes`	vector of clone sizes at which cumulative mutation counts were measured
`expected.mutations`	expected number of mutations at each clone size
`depth`	sequencing depth, only specify for bulk WGS
`ncells`	the number of sequenced cells, only specify if `seqtype=="sc"`.
`sensitivity`	logical, if sensitivity of sequencing method should be taken into account in addition to binomial noise. Requires a specification for `false.negative.per.vaf`.
`false.negative.per.vaf`	optional, a matrix with columns corresponding to the measured VAFs and rows corresponding to individual measurements of the false negative rate at this VAF in addition to binomial noise.
`min.vaf`	the minimal VAF to return

Value

A vector of simulated VAFs

`simulate_vaf_upon_sequencing` *Simulate the measured VAF after sequencing*

Description

Simulate the measured VAF after sequencing

Usage

```
simulate_vaf_upon_sequencing(vaf, depth)
```

Arguments

`vaf`	a vector of true VAFs in the population
`depth`	average coverage in sequencing

Value

a vector of simulated VAFs after sequencing

`snvs` *SNV data from individual A1*

Description

Exemplary SNV data from individual A1 of the study Körber et al., Detecting and quantifying clonal selection in somatic mosaicism. The dataset is a list object, containing variant information in vcf format.

Usage

```
snvs
```

Format

`snvs`**:**

A list containing a data frame with 447 rows and 45 columns:

Chr Chromosome

Start, End Start and end position of the variant

Ref, Alt Reference and alternative base par

VAF, Depth, varCounts Variant allele frequency, read depth and number of variant reads

VAF.control Variant allele frequency in the control data set

Func.refGene Annovar annotation of the functional change (e.g., exonic, intergentic)

GeneDetail.refGene Annovar annotation of the gene ID.

ExonicFunc.refGene Annovar annotation of the exonic change in the gene (e.g. nonsynonymous)

Gene.refGene Annovar annotation of the gene symbol

AAChange.refGene Annovar annotation of the amino acid substitution

avsnp150 dbSNP identifier

ExAC_ALL, ExAC_AFR, ExAC_AMR, ExAC_EAS, ExAC_FIN, ExAC_NFE, ExAC_OTH, ExAC_SAS exome aggregation consortium information

AF, AF_popmax, AF_male, AF_female, AF_raw, AF_afr, AF_sas, AF_amr, AF_eas, AF_nfe, AF_fin, AF_asj, A GnomAD annotated population-wide allele frequencies

CLINALLELEID, CLNDN, CLNDISDB, CLNREVSTAT, CLNSIG Clinvar annotation

Index