Package: RestoreNet (via r-universe)

October 13, 2024

Title Random-Effects Stochastic Reaction Networks

Version 1.0.1

Description A random-effects stochastic model that allows quick detection of clonal dominance events from clonal tracking data collected in gene therapy studies. Starting from the Ito-type equation describing the dynamics of cells duplication, death and differentiation at clonal level, we first considered its local linear approximation as the base model. The parameters of the base model, which are inferred using a maximum likelihood approach, are assumed to be shared across the clones. Although this assumption makes inference easier, in some cases it can be too restrictive and does not take into account possible scenarios of clonal dominance. Therefore we extended the base model by introducing random effects for the clones. In this extended formulation the dynamic parameters are estimated using a tailor-made expectation maximization algorithm. Further details on the methods can be found in L. Del Core et al., (2022) <doi:10.1101/2022.05.31.494100>.

License GPL-3

Encoding UTF-8

RoxygenNote 7.2.3.9000

Imports Matrix, xtable, scales, stringr, ggplot2, scatterpie, RColorBrewer

Depends R (>= 2.10)

LazyData true

Suggests R.rsp

VignetteBuilder R.rsp

NeedsCompilation no

Author Luca Del Core [aut, cre, cph] (<https://orcid.org/0000-0002-1672-6995>), Marco Grzegorczyk [aut, ths] (<https://orcid.org/0000-0002-2604-9270>), Ernst Wit [aut, ths] (<https://orcid.org/0000-0002-3671-9610>)

Maintainer Luca Del Core <l.del.core@rug.nl>

Date/Publication 2024-02-15 11:00:02 UTC

Repository https://delcore-luca.r-universe.dev

RemoteUrl https://github.com/cran/RestoreNet

RemoteRef HEAD

RemoteSha cf0167c84e61acd6c253890525b511ec9b2f46b9

Contents

`fit.null` *Fit the base (null) model*

Description

This function builds the design matrix of the null model and returns the fitted values and the corresponding statistics.

Usage

```
fit.null(
  Y,
  rct.lst,
  maxit = 10000,
  factr = 1e+07,
  pgtol = 1e-08,
  lmm = 100,
  trace = TRUE,
  verbose = TRUE
)
```

Arguments

`Y`	A 3-dimensional array whose dimensions are the time, the cell type and the clone respectively.

rct.lst	list of biochemical reactions. A differentiation move from cell type "A" to cell type "B" must be coded as "A->B" Duplication of cell "A" must be coded as "A->1" Death of cell "A" must be coded as "A->0"
maxit	maximum number of iterations for the optimization step. This argument is passed to optim() function. Details on "maxit" can be found in "optim()" documentation page.
factr	controls the convergence of the "L-BFGS-B" method. Convergence occurs when the reduction in the objective is within this factor of the machine tolerance. Default is 1e7, that is a tolerance of about 1e-8. This argument is passed to optim() function.
pgtol	helps control the convergence of the "L-BFGS-B" method. It is a tolerance on the projected gradient in the current search direction. This defaults to zero, when the check is suppressed. This argument is passed to optim() function.
lmm	is an integer giving the number of BFGS updates retained in the "L-BFGS-B" method, It defaults to 5. This argument is passed to optim() function.
trace	Non-negative integer. If positive, tracing information on the progress of the optimization is produced. This parameter is also passed to the optim() function. Higher values may produce more tracing information: for method "L-BFGS-B" there are six levels of tracing. (To understand exactly what these do see the source code: higher levels give more detail.)
verbose	(defaults to TRUE) Logical value. If TRUE, then information messages on the progress of the algorithm are printed to the console.

Value

A 3-length list. First element is the output returned by "optim()" function (see "optim()" documentation for details). Second element is a vector of statistics associated to the fitted null model:

nPar	number of parameters of the base(null) model
cll	value of the conditional log-likelihood, in this case just the log-likelihood
mll	value of the marginal log-likelihood, in this case just the log-likelihood
cAIC	conditional Akaike Information Criterion (cAIC), in this case simply the AIC.
mAIC	marginal Akaike Information Criterion (mAIC), in this case simply the AIC.
Chi2	value of the χ^2 statistic $(y - M\theta)' S^{-1} (y - M\theta)$.
p-value	p-value of the χ^2 test for the null hypothesis that Chi2 follows a χ^2 distribution with n - nPar degrees of freedom.

The third element, called "design", is a list including:

M	A $n \times K$ dimensional (design) matrix.

V A $p \times K$ dimensional net-effect matrix.

Examples

```
rcts <- c("A->1", "B->1", "C->1", "D->1",
          "A->0", "B->0", "C->0", "D->0",
          "A->B", "A->C", "C->D") ## set of reactions
ctps <- head(LETTERS,4)
nC <- 3 ## number of clones
S <- 10 ## trajectory length
tau <- 1 ## for tau-leaping algorithm
u_1 <- c(.2, .15, .17, .09*5,
         .001, .007, .004, .002,
         .13, .15, .08)
u_2 <- c(.2, .15, .17, .09,
         .001, .007, .004, .002,
         .13, .15, .08)
u_3 <- c(.2, .15, .17*3, .09,
         .001, .007, .004, .002,
         .13, .15, .08)
theta_allcls <- cbind(u_1, u_2, u_3) ## clone-specific parameters
rownames(theta_allcls) <- rcts
s20 <- 1 ## additional noise
Y <- array(data = NA,
           dim = c(S + 1, length(ctps), nC),
           dimnames = list(seq(from = 0, to = S*tau, by = tau),
                           ctps,
                           1:nC)) ## empty array to store simulations
Y0 <- c(100,0,0,0) ## initial state
names(Y0) <- ctps
for (cl in 1:nC) { ## loop over clones
  Y[,,cl] <- get.sim.tl(Yt = Y0,
                        theta = theta_allcls[,cl],
                        S = S,
                        s2 = s20,
                        tau = tau,
                        rct.lst = rcts,
                        verbose = TRUE)
}
null.res <- fit.null(Y = Y,
                     rct.lst = rcts,
                     maxit = 0, ## needs to be increased (>=100) for real applications
                     lmm = 0, ## needs to be increased (>=5) for real applications
) ## null model fitting
```

`fit.re` *Fit the random-effects model*

Description

This function builds the design matrix of the random-effects model and returns the fitted values and the corresponding statistics.

Usage

```
fit.re(
  theta_0,
  Y,
  rct.lst,
  maxit = 10000,
  factr = 1e+07,
  pgtol = 1e-08,
  lmm = 100,
  maxemit = 100,
  eps = 1e-05,
  trace = TRUE,
  verbose = TRUE
)
```

Arguments

theta_0	A p-dimensional vector parameter as the initial guess for the inference.
Y	A 3-dimensional array whose dimensions are the time, the cell type and the clone respectively.
rct.lst	list of biochemical reactions. A differentiation move from cell type "A" to cell type "B" must be coded as "A->B" Duplication of cell "A" must be coded as "A->1" Death of cell "A" must be coded as "A->0"
maxit	maximum number of iterations for the optimization step. This argument is passed to optim() function. Details on "maxit" can be found in "optim()" documentation page.
factr	controls the convergence of the "L-BFGS-B" method. Convergence occurs when the reduction in the objective is within this factor of the machine tolerance. Default is 1e7, that is a tolerance of about 1e-8. This argument is passed to optim() function.
pgtol	helps control the convergence of the "L-BFGS-B" method. It is a tolerance on the projected gradient in the current search direction. This defaults to zero, when the check is suppressed. This argument is passed to optim() function.
lmm	is an integer giving the number of BFGS updates retained in the "L-BFGS-B" method, It defaults to 5. This argument is passed to optim() function.
maxemit	maximum number of iterations for the expectation-maximization algorithm.
eps	relative error for the value x and the objective function f(x) that has to be optimized in the expectation-maximization algorithm.
trace	Non-negative integer. If positive, tracing information on the progress of the optimization is produced. This parameter is also passed to the optim() function. Higher values may produce more tracing information: for method "L-BFGS-B"

there are six levels of tracing. (To understand exactly what these do see the source code: higher levels give more detail.)

`verbose` (defaults to TRUE) Logical value. If TRUE, then information messages on the progress of the algorithm are printed to the console.

Value

A 3-length list. First element is the output returned by "optim()" function (see "optim()" documentation for details) along with the conditional expectation $E[u|y]$ and variance $V[u|y]$ of the latent states u given the observed states y from the last step of the expectation-maximization algorithm. Second element is a vector of statistics associated to the fitted random-effects model:

`nPar`	number of parameters of the base(null) model
`cll`	value of the conditional log-likelihood, in this case just the log-likelihood
`mll`	value of the marginal log-likelihood, in this case just the log-likelihood
`cAIC`	conditional Akaike Information Criterion (cAIC), in this case simply the AIC.
`mAIC`	marginal Akaike Information Criterion (mAIC), in this case simply the AIC.
`Chi2`	value of the χ^2 statistic $(y - M\theta)' S^{-1} (y - M\theta)$.
`p-value`	p-value of the χ^2 test for the null hypothesis that Chi2 follows a χ^2 distribution with n - nPar degr
`KLdiv`	Kullback-Leibler divergence of the random-effects model from the null model.
`KLdiv/N`	Rescaled Kullback-Leibler divergence of the random-effects model from the null model.
`BhattDist_nullCond`	Bhattacharyya distance between the random-effects model and the null model.
`BhattDist_nullCond/N`	Rescaled Bhattacharyya distance between the random-effects model and the null model.

The third element, called "design", is a list including:

`M`	A $n \times K$ dimensional (design) matrix.
`M_bdiag`	A$n \times Jp$ dimensional block-diagonal design matrix.
`V`	A $p \times K$ dimensional net-effect matrix.

Examples

```
rcts <- c("A->1", "B->1", "C->1", "D->1",
          "A->0", "B->0", "C->0", "D->0",
          "A->B", "A->C", "C->D") ## set of reactions
```

```
ctps <- head(LETTERS,4)
nC <- 3 ## number of clones
S <- 10 ## trajectory length
tau <- 1 ## for tau-leaping algorithm
u_1 <- c(.2, .15, .17, .09*5,
         .001, .007, .004, .002,
         .13, .15, .08)
u_2 <- c(.2, .15, .17, .09,
         .001, .007, .004, .002,
         .13, .15, .08)
u_3 <- c(.2, .15, .17*3, .09,
         .001, .007, .004, .002,
         .13, .15, .08)
theta_allcls <- cbind(u_1, u_2, u_3) ## clone-specific parameters
rownames(theta_allcls) <- rcts
s20 <- 1 ## additional noise
Y <- array(data = NA,
           dim = c(S + 1, length(ctps), nC),
           dimnames = list(seq(from = 0, to = S*tau, by = tau),
                           ctps,
                           1:nC)) ## empty array to store simulations
Y0 <- c(100,0,0,0) ## initial state
names(Y0) <- ctps
for (cl in 1:nC) { ## loop over clones
  Y[,,cl] <- get.sim.tl(Yt = Y0,
                        theta = theta_allcls[,cl],
                        S = S,
                        s2 = s20,
                        tau = tau,
                        rct.lst = rcts,
                        verbose = TRUE)
}
null.res <- fit.null(Y = Y,
                     rct.lst = rcts,
                     maxit = 0, ## needs to be increased (>=100) for real applications
                     lmm = 0, ## needs to be increased (>=5) for real applications
) ## null model fitting

re.res <- fit.re(theta_0 = null.res$fit$par,
                 Y = Y,
                 rct.lst = rcts,
                 maxit = 0, ## needs to be increased (>=100) for real applications
                 lmm = 0, ## needs to be increased (>=5) for real applications
                 maxemit = 1 ## needs to be increased (>= 100) for real applications
) ## random-effects model fitting
```

`get.boxplots` *Clonal boxplots*

Description

Draw clonal boxplots of a random-effects reaction network.

Usage

```
get.boxplots(re.res)
```

Arguments

re.res	output list returned by fit.re().

Details

This function generates the boxplots of the conditional expectations

$$w_k = E_{u|\Delta Y;\hat{\psi}}[u^k_{\alpha_l}] - E_{u|\Delta Y;\hat{\psi}}[u^k_{\delta_l}]$$

, computed from the estimated parameters $\hat{\psi}$ for the clone-specific net-duplication in each cell lineage l (different colors). The whiskers extend to the data extremes.

Value

No return value.

Examples

```
rcts <- c("A->1", "B->1", "C->1", "D->1",
          "A->0", "B->0", "C->0", "D->0",
          "A->B", "A->C", "C->D") ## set of reactions
ctps <- head(LETTERS,4)
nC <- 3 ## number of clones
S <- 10 ## trajectory length
tau <- 1 ## for tau-leaping algorithm
u_1 <- c(.2, .15, .17, .09*5,
         .001, .007, .004, .002,
         .13, .15, .08)
u_2 <- c(.2, .15, .17, .09,
         .001, .007, .004, .002,
         .13, .15, .08)
u_3 <- c(.2, .15, .17*3, .09,
         .001, .007, .004, .002,
         .13, .15, .08)
theta_allcls <- cbind(u_1, u_2, u_3) ## clone-specific parameters
rownames(theta_allcls) <- rcts
s20 <- 1 ## additional noise
Y <- array(data = NA,
           dim = c(S + 1, length(ctps), nC),
           dimnames = list(seq(from = 0, to = S*tau, by = tau),
                           ctps,
                           1:nC)) ## empty array to store simulations
Y0 <- c(100,0,0,0) ## initial state
names(Y0) <- ctps
for (cl in 1:nC) { ## loop over clones
  Y[,,cl] <- get.sim.tl(Yt = Y0,
                        theta = theta_allcls[,cl],
```

```
                        S = S,
                        s2 = s20,
                        tau = tau,
                        rct.lst = rcts,
                        verbose = TRUE)
}
null.res <- fit.null(Y = Y,
                     rct.lst = rcts,
                     maxit = 0, ## needs to be increased (>=100) for real applications
                     lmm = 0, ## needs to be increased (>=5) for real applications
) ## null model fitting

re.res <- fit.re(theta_0 = null.res$fit$par,
                 Y = Y,
                 rct.lst = rcts,
                 maxit = 0, ## needs to be increased (>=100) for real applications
                 lmm = 0, ## needs to be increased (>=5) for real applications
                 maxemit = 1 ## needs to be increased (>= 100) for real applications
) ## random-effects model fitting

get.boxplots(re.res)
```

`get.rescaled` *Rescaling a clonal tracking dataset*

Description

Rescales a clonal tracking dataset based on the sequencing depth.

Usage

```
get.rescaled(Y)
```

Arguments

`Y`	A 3-dimensional array whose dimensions are the time, the cell type and the clone respectively.

Details

This function rescales a clonal tracking dataset Y according to the formula

$$Y_{ijk} \leftarrow Y_{ijk} \cdot \frac{min_{ij} \sum_c Y_{ijc}}{\sum_c Y_{ijc}}$$

Value

A rescaled clonal tracking dataset.

Examples

```
get.rescaled(Y_RM[["ZH33"]])
```

`get.scatterpie` *Clonal pie-chart*

Description

Draw a clonal pie-chart of a random-effects reaction network.

Usage

```
get.scatterpie(re.res, txt = FALSE, legend = FALSE)
```

Arguments

`re.res`	output list returned by fit.re().
`txt`	logical (defaults to FALSE). If TRUE, barcode names will be printed on the pies.
`legend`	logical (defaults to FALSE). If TRUE, the legend of the pie-chart will be printed.

Details

This function generates a clonal pie-chart given a previously fitted random-effects model. In this representation each clone k is identified with a pie whose slices are lineage-specific and weighted with w_k, defined as the difference between the conditional expectations of the random-effects on duplication and death parameters, that is

$$w_k = E_{u|\Delta Y;\hat{\psi}}[u^k_{\alpha_{lin}}] - E_{u|\Delta Y;\hat{\psi}}[u^k_{\delta_{lin}}]$$

, where `lin` is a cell lineage. The diameter of the k-th pie is proportional to the euclidean 2-norm of w_k. Therefore, the larger the diameter, the more the corresponding clone is expanding into the lineage associated to the largest slice.

Value

No return value.

Examples

```
rcts <- c("A->1", "B->1", "C->1", "D->1",
          "A->0", "B->0", "C->0", "D->0",
          "A->B", "A->C", "C->D") ## set of reactions
ctps <- head(LETTERS,4)
nC <- 3 ## number of clones
S <- 10 ## trajectory length
tau <- 1 ## for tau-leaping algorithm
u_1 <- c(.2, .15, .17, .09*5,
         .001, .007, .004, .002,
```

```
         .13, .15, .08)
u_2 <- c(.2, .15, .17, .09,
         .001, .007, .004, .002,
         .13, .15, .08)
u_3 <- c(.2, .15, .17*3, .09,
         .001, .007, .004, .002,
         .13, .15, .08)
theta_allcls <- cbind(u_1, u_2, u_3) ## clone-specific parameters
rownames(theta_allcls) <- rcts
s20 <- 1 ## additional noise
Y <- array(data = NA,
           dim = c(S + 1, length(ctps), nC),
           dimnames = list(seq(from = 0, to = S*tau, by = tau),
                           ctps,
                           1:nC)) ## empty array to store simulations
Y0 <- c(100,0,0,0) ## initial state
names(Y0) <- ctps
for (cl in 1:nC) { ## loop over clones
  Y[,,cl] <- get.sim.tl(Yt = Y0,
                        theta = theta_allcls[,cl],
                        S = S,
                        s2 = s20,
                        tau = tau,
                        rct.lst = rcts,
                        verbose = TRUE)
}
null.res <- fit.null(Y = Y,
                     rct.lst = rcts,
                     maxit = 0, ## needs to be increased (>=100) for real applications
                     lmm = 0, ## needs to be increased (>=5) for real applications
) ## null model fitting

re.res <- fit.re(theta_0 = null.res$fit$par,
                 Y = Y,
                 rct.lst = rcts,
                 maxit = 0, ## needs to be increased (>=100) for real applications
                 lmm = 0, ## needs to be increased (>=5) for real applications
                 maxemit = 1 ## needs to be increased (>= 100) for real applications
) ## random-effects model fitting

get.scatterpie(re.res, txt = TRUE)
```

`get.sim.tl` *τ-leaping simulation algorithm*

Description

Simulate a trajectory of length S for a stochastic reaction network.

Usage

```
get.sim.tl(Yt, theta, S, s2 = 0, tau = 1, rct.lst, verbose = TRUE)
```

Arguments

Yt	starting point of the trajectory
theta	vector parameter for the reactions.
S	length of the simulated trajectory.
s2	noise variance (defaults to 0).
tau	time interval length (defaults to 1).
rct.lst	list of biochemical reactions.
verbose	(defaults to TRUE) Logical value. If TRUE, then information messages on the simulation progress are printed to the console.

Details

This function allows to simulate a trajectory of a single clone given an initial conditions Y_0 for the cell counts, and obeying to a particular cell differentiation network defined by a net-effect (stoichiometric) matrix V and an hazard function $h()$. The function allows to consider only three cellular events, such as cell duplication ($Y_{it} \to 1$), cell death ($Y_{it} \to \emptyset$) and cell differentiation ($Y_{it} \to Y_{jt}$) for a clone-specific time counting process

$$Y_t = (Y_{1t}, \ldots, Y_{Nt})$$

observed in N distinct cell lineages. In particular, the cellular events of duplication, death and differentiation are respectively coded with the character labels `"A->1"`, `"A->0"`, and `"A->B"`, where `A` and `B` are two distinct cell types. The output is a 3-dimensional array Y whose ijk-entry Y_{ijk} is the number of cells of clone k for cell type j collected at time i. More mathematical details can be found in the vignette of this package.

Value

A $S \times p$ dimensional matrix of the simulated trajectory.

Examples

```
rcts <- c("A->1", "B->1", "C->1", "D->1",
          "A->0", "B->0", "C->0", "D->0",
          "A->B", "A->C", "C->D") ## set of reactions
ctps <- head(LETTERS,4)
nC <- 3 ## number of clones
S <- 10 ## trajectory length
tau <- 1 ## for tau-leaping algorithm
u_1 <- c(.2, .15, .17, .09*5,
         .001, .007, .004, .002,
         .13, .15, .08)
u_2 <- c(.2, .15, .17, .09,
         .001, .007, .004, .002,
```

```
         .13, .15, .08)
u_3 <- c(.2, .15, .17*3, .09,
         .001, .007, .004, .002,
         .13, .15, .08)
theta_allcls <- cbind(u_1, u_2, u_3) ## clone-specific parameters
rownames(theta_allcls) <- rcts
s20 <- 1 ## additional noise
Y <- array(data = NA,
           dim = c(S + 1, length(ctps), nC),
           dimnames = list(seq(from = 0, to = S*tau, by = tau),
                           ctps,
                           1:nC)) ## empty array to store simulations
Y0 <- c(100,0,0,0) ## initial state
names(Y0) <- ctps
for (cl in 1:nC) { ## loop over clones
  Y[,,cl] <- get.sim.tl(Yt = Y0,
                        theta = theta_allcls[,cl],
                        S = S,
                        s2 = s20,
                        tau = tau,
                        rct.lst = rcts,
                        verbose = TRUE)
}
Y
```

Y_RM *Rhesus Macaque clonal tracking dataset*

Description

A dataset containing clonal tracking cell counts from a Rhesus Macaque study.

Usage

```
Y_RM
```

Format

A list containing clonal tracking data for each animal (ZH33, ZH17, ZG66). Each clonal tracking dataset is a 3-dimensional array whose dimensions identify

1 time, in months

2 cell types: T, B, NK, Macrophages(M) and Granulocytes(G)

3 unique barcodes (clones)

Source

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3979461/bin/NIHMS567927-supplement-02.xlsx

Index