A quick introduction to RestoreNet

L. Del Core

l.del.core@rug.nl

February 15, 2024

Abstract

This document reviews the key functionalities of RestoreNet package. Section 1 shows how to simulate a clonal tracking dataset from a stochastic quasi-reaction network. In particular, we show how to simulate clone-specific trajectories following a given set of biochemical reactions. Sections 2 and 3 show how to fit the null (base) model and the random-effects model to a simulated clonal tracking dataset. Finally in Section 4 we show how to visualize the results at clonal level.

1 Simulating clonal tracking datasets

A clonal tracking dataset compatible with RestoreNet's functions must be formatted as a 3-dimensional array $\boldsymbol{Y}$ whose ijk-entry y_{ijk} is the number of cells of clone k for cell type j collected at time i. The function `get.sim.tl()` can be used to simulate a trajectory of a single clone given an initial conditions $\boldsymbol{Y}_0$ for the cell counts, and obeying to a particular cell differentiation network defined by a list `rct.lst` of biochemical reactions. Consistently with [1], our package considers only three cellular events, such as cell duplication, cell death and cell differentiation for a time counting process

$$\boldsymbol{x}_t = (x_{1t}, \dots, x_{nt}) \tag{1}$$

of a single clone in n distinct cell lineages, whose measurements are indicated with $\boldsymbol{y}_t = (y_{1t}, \dots, y_{nt})$. Following [1], we assume that the time counting process $\boldsymbol{x}_t$ for a single clone in a time interval $(t, t + \Delta t)$ evolves according to a set of reactions $\{\boldsymbol{V}_{\cdot k}\}_k$ and hazard functions $\{h_k\}_k$ defined as

$$\boldsymbol{V}_{\cdot k} = \begin{cases} (0 \dots 1_i \dots 0)' \\ (0 \dots -1_i \dots 0)' \\ (0 \dots -1_i \dots 2_j \dots 0)' \end{cases} \qquad h_k(\boldsymbol{x}_t, \boldsymbol{\theta}) = \begin{cases} x_{it}\alpha_i & \text{for duplication} \\ x_{it}^2\delta_i & \text{for death} \\ x_{it}\lambda_{ij} & \text{for differentiation} \end{cases} \tag{2}$$

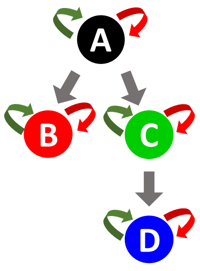


Figure 1: Cell differentiation structure of four synthetic cell types A, B, C and D. Duplication, death and differentiation moves are indicated with green, red and grey arrows respectively.

which contains a linear growth term with a duplication rate parameter $\alpha_i > 0$, and a linear term to describe cell differentiation from lineage i to lineage j with differentiation rate $\lambda_{ij} > 0$ for each $i \neq j = 1, \ldots, n$. Finally we employ a quadratic term for cell death with a death rate parameter $\delta_i > 0$, and the vector parameter

$$\boldsymbol{\theta} = (\cdots \alpha_i \cdots \delta_i \cdots \lambda_{ij} \cdots) \tag{3}$$

is the concatenation of all the dynamics parameters. Thus, the net-effect matrix and the hazard vector are defined as

$$V = \left[\boldsymbol{V}_{\cdot 1} \cdots \boldsymbol{V}_{\cdot k}\right]; \qquad \boldsymbol{h}(\boldsymbol{x}_t, \boldsymbol{\theta}) = (h_1(\boldsymbol{x}_t, \boldsymbol{\theta}), \ldots, h_K(\boldsymbol{x}_t, \boldsymbol{\theta}))' \tag{4}$$

The cellular events of duplication, death and differentiation are respectively coded by the package functions with the character labels "A->1", "A->0", and "A->B", where A and B are two distinct cell types. The following R code chunk shows how to simulate clone-specific trajectories of cells via a τ-leaping simulation algorithm. In particular, as an illustrative example we focus on a simple cell differentiation network structure of four synthetic cell types A, B, C and D, as illustrated in Figure 1, and only one clone.

```
> 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
> S <- 100 ## trajectory length
> tau <- 1 ## for tau-leaping algorithm
> theta <- c(.2,.15,.17,.09*5,
             .001 , .007 , .004 , .002 ,
             .13, .15, .08) ## parameters
> names(theta) <- rcts
> Y0 <- c(100,0,0,0) ## initial state names(Y0) <- rownames(V)
> names(Y0) <- head(LETTERS ,4)
> s20 <- 1 ## noise variance
> Y <- get.sim.tl(Yt = Y0,
                  theta = theta,
                    S = S,
                    s2 = s20,
                    tau = tau,
                    rct.lst = rcts) ## simulation
> head(Y) ## look at the simulated data
          A           B          C          D
0 100.61983  0.06136727  0.7714631  0.3255576
1  82.64798 25.80389091 30.2276346  0.0000000
2  67.38059 44.75329724 52.8111779  4.9761676
3  59.22818 57.88492115 64.9075555 15.2798701
4  49.95502 57.19943051 73.4204752 32.5405621
5  43.79580 56.15629549 73.4675043 57.1191486
```

2 Fitting the base model

Following [1], the base model is defined as

$$\underbrace{\begin{bmatrix} \Delta \boldsymbol{y}_{t_0} \\ \vdots \\ \Delta \boldsymbol{y}_{t_{T-1}} \end{bmatrix}}_{\Delta \boldsymbol{y}} = \underbrace{\begin{bmatrix} \boldsymbol{M}_{t_0} \\ \vdots \\ \boldsymbol{M}_{t_{T-1}} \end{bmatrix}}_{\boldsymbol{M}} \boldsymbol{\theta} + \boldsymbol{\varepsilon}; \quad \boldsymbol{\varepsilon} \sim \mathcal{N}\left(\boldsymbol{0}, \overbrace{\underbrace{\begin{bmatrix} \boldsymbol{W}_{t_0}(\boldsymbol{\theta}) & & \\ & \ddots & \\ & & W_{t_{T-1}}(\boldsymbol{\theta}) \end{bmatrix}}_{\boldsymbol{W}(\boldsymbol{\theta})} + \sigma^2 \boldsymbol{I}_{nT}}^{\boldsymbol{\Sigma}(\boldsymbol{\theta},\sigma^2)} \right) \tag{5}$$

where

$$\underbrace{\Delta \boldsymbol{y}_t}_{\boldsymbol{y}_{t+1}-\boldsymbol{y}_t} = \boldsymbol{V} \overbrace{\begin{bmatrix} \prod_{i=1}^n \binom{y_{it}}{r_{1i}} & & \\ & \ddots & \\ & & \prod_{i=1}^n \binom{y_{it}}{r_{Ki}} \end{bmatrix}}^{\boldsymbol{M}_t} \Delta t \underbrace{\begin{bmatrix} \theta_1 \\ \vdots \\ \theta_K \end{bmatrix}}_{\boldsymbol{\theta}} + \left(\underbrace{\boldsymbol{V} \begin{bmatrix} h_1(\boldsymbol{y}_t,\boldsymbol{\theta}) & & \\ & \ddots & \\ & & h_1(\boldsymbol{y}_t,\boldsymbol{\theta}) \end{bmatrix} \boldsymbol{V}' \Delta t}_{\boldsymbol{W}_t(\theta)} + \sigma^2 \boldsymbol{I}_n \right)^{1/2} \Delta \boldsymbol{W}(t) \tag{6}$$

$$\Delta \boldsymbol{W}(t) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_n)$$

The package RestoreNet allows to infer the parameters $(\boldsymbol{\theta}, \sigma^2)$ of (5) with a maximum likelihood approach, that is by solving the following constrained optimization problem

$$\hat{\boldsymbol{\theta}}_{ML} \leftarrow \underset{\boldsymbol{\theta} \geq \boldsymbol{0}; \sigma^2 \geq 0}{\texttt{argmin}} f(\boldsymbol{\theta}, \sigma^2) \tag{7}$$

where the objective function f is the negative log-likelihood of the multivariate normal distribution $\mathcal{N}_{nT}\left(\boldsymbol{M}\boldsymbol{\theta}, \boldsymbol{\Sigma}(\boldsymbol{\theta}, \sigma^2)\right)$. Details on the inference procedure can be found in [1]. The following R code chunk shows how to accomplish this on a clonal tracking dataset simulated from the same differentiation network structure of previous section. In this case we simulate the trajectories of three independent clones following different dynamics of clonal dominance, that is we use clone-specific values for the vector parameter $\boldsymbol{\theta}$.

```
> 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 <- 100 ## 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)
> }
> null.res <- fit.null(Y = Y, rct.lst = rcts) ## null model fitting
> null.res$ fit ## model fitting results
$par
 [1] 6.788801e-02 2.125983e-02 9.192739e-03 2.753155e-03
        1.000000e-07 2.102263e-03 8.510596e-05 7.137124e-05
 [9] 7.727499e-02 1.147283e-01 3.631258e-02 1.297511e+00

$value
[1] 3419.932

$counts
function gradient
     673      673

$convergence
```

```
[1] 0

$message
[1] "CONVERGENCE:␣REL_REDUCTION_OF_F␣<=␣FACTR*EPSMCH"
> null.res$stats ## model statistics
nPar    cll     mll     cAIC    mAIC    Chi2    p-value
12.000  -2812.692 -2812.692 5649.384 5651.691 337324.840 0.000

> head(null.res$design$M) ## design matrix
6 x 11 sparse Matrix of class "dgCMatrix"
1 100.61983 .   .       .       -10124.350      .       .
1       .       0.06136727      .       .       .       -0.003765942
1       .       .       0.7714631       .       .       .
1       .       .       .       0.3255576       .       .
1  82.64798     .       .       .       -6830.688       .

> null.res$design$V ## net-effect matrix
  A->1 B->1 C->1 D->1 A->0 B->0 C->0 D->0 A->B A->C C->D
A    1    0    0    0   -1    0    0    0   -1   -1    0
B    0    1    0    0    0   -1    0    0    2    0    0
C    0    0    1    0    0    0   -1    0    0    2   -1
D    0    0    0    1    0    0    0   -1    0    0    2
```

3 Fitting the random-effects model

Consistently with [1], the random-effects model is defined as

$$\Delta\boldsymbol{y} = \underbrace{\begin{bmatrix} \boldsymbol{M}_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \boldsymbol{M}_J \end{bmatrix}}_{\mathbb{M}\in\mathbb{R}^{n\times Jp}} \boldsymbol{u} + \boldsymbol{\varepsilon} \qquad \boldsymbol{u} \sim \mathcal{N}_{Jp}\left(\underbrace{\mathbf{1}_J \otimes \boldsymbol{\theta}}_{\boldsymbol{\theta}_u}, \boldsymbol{I}_J \otimes \underbrace{\begin{bmatrix} \tau_1^2 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \tau_p^2 \end{bmatrix}}_{\boldsymbol{\Delta}_u} \right)$$
$$\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}(\boldsymbol{\theta}, \sigma^2)) \tag{8}$$

where $\mathbb{M}$ is the block-diagonal design matrix for the random effects $\boldsymbol{u}$ centered in $\boldsymbol{\theta}$, and each block $\boldsymbol{M}_j$ is clone-specific. As in the case of the null model (5), to explain additional noise of the data and to avoid singularity of the stochastic covariance matrix $\boldsymbol{W}(\boldsymbol{\theta})$ we added to its diagonal a small unknown quantity σ^2 which we infer from the data. Under this framework (see [1] for details) the conditional distribution of the random effects $\boldsymbol{u}$ given the data $\Delta\boldsymbol{y}$ has the following explicit formulation

$$\boldsymbol{u}|\Delta\boldsymbol{y} \sim \mathcal{N}_{Jp}(E_{\boldsymbol{u}|\Delta\boldsymbol{y};\boldsymbol{\psi}}[\boldsymbol{u}], V_{\boldsymbol{u}|\Delta\boldsymbol{y};\boldsymbol{\psi}}(\boldsymbol{u})) \tag{9}$$

where

$$\begin{aligned} E_{\boldsymbol{u}|\Delta\boldsymbol{y};\boldsymbol{\psi}}[\boldsymbol{u}] &= V_{\boldsymbol{u}|\Delta\boldsymbol{y};\boldsymbol{\psi}}(\boldsymbol{u}) \left(\mathbb{M}'\boldsymbol{\Sigma}^{-1}(\boldsymbol{\theta},\sigma^2)\Delta\boldsymbol{y} + \boldsymbol{\Delta}_u^{-1}\boldsymbol{\theta}_u\right) \\ V_{\boldsymbol{u}|\Delta\boldsymbol{y};\boldsymbol{\psi}}(\boldsymbol{u}) &= \left(\mathbb{M}'\boldsymbol{\Sigma}^{-1}(\boldsymbol{\theta},\sigma^2)\mathbb{M} + \boldsymbol{\Delta}_u^{-1}\right)^{-1} \end{aligned} \tag{10}$$

provide clone-specific mean and variance of the (random) reaction rates. The package RestoreNet allows to infer the vector parameter $\boldsymbol{\psi} = (\boldsymbol{\theta}, \sigma^2, \tau_1^2, \dots, \tau_p^2)$, and in turn to get the corresponding conditional first two-order moments $E_{\boldsymbol{u}|\Delta\boldsymbol{y};\boldsymbol{\psi}}[\boldsymbol{u}]$ and $V_{\boldsymbol{u}|\Delta\boldsymbol{y};\boldsymbol{\psi}}(\boldsymbol{u})$, by the means of an efficient tailor-made expectation maximization algorithm where $\Delta\boldsymbol{y}$ and $\boldsymbol{u}$ take the roles of the observed and latent states respectively. Further details on the inference procedure can be found in [1]. The following R code chunk shows how to accomplish this on the simulated clonal tracking dataset of previous section. In this example we use the optimal parameter vector $\hat{\boldsymbol{\theta}}_0$ estimated for the null model in the previous section, as initial guess for the corresponding parameters in the random-effects model.

```
> re.res <- fit.re(theta_0 = null.res$fit$par,
                   Y = Y,
                   rct.lst = rcts,
                   maxemit = 100) ## random-effects model fitting
> re.res$fit$par ## estimated parameters
 [1] 1.000000e-07 1.843245e-03 1.000000e-07 1.036969e-04
        5.255077e-04 1.000000e-07 1.000000e-07 1.000000e-07
 [9] 1.026921e-03 5.080835e-03 1.000000e-07 3.837475e-02
        2.862468e-02 7.111302e-02 6.109796e-02 1.000000e-07
[17] 4.675422e-05 1.550055e-05 4.952111e-06 1.416910e-02
        2.576975e-02 1.106758e-02 1.720079e+00

> re.res$fit$VEuy$euy ## conditional expected values of u|y
33 x 1 Matrix of class "dgeMatrix"
              [,1]
 [1,] 0.1693522400
 [2,] 0.1478834088
 [3,] 0.1643743275
 [4,] 0.4553735855
 [5,] 0.0006527738
.
.
.
> re.res$fit$VEuy$vuy ## conditional covariance matrix of u|y
33 x 33 sparse Matrix of class "dsCMatrix"

 [1,]  3.552098e-04  2.910616e-05  2.925707e-05 .        .        .
 [2,]  2.910616e-05  2.095979e-04 -3.311544e-07 .        .        .
 [3,]  2.925707e-05 -3.311544e-07  1.478458e-04 .        .        .
        .             .             .
        .             .             .
        .             .             .
```

4 Visualizing results

The main graphical output of RestoreNet is a clonal piechart. Consistently with [1], in this representation each clone k is identified with a pie whose slices are lineage-specific and weighted with w_k^l, defined as the difference between the conditional expectations of the duplication and death parameters, that is

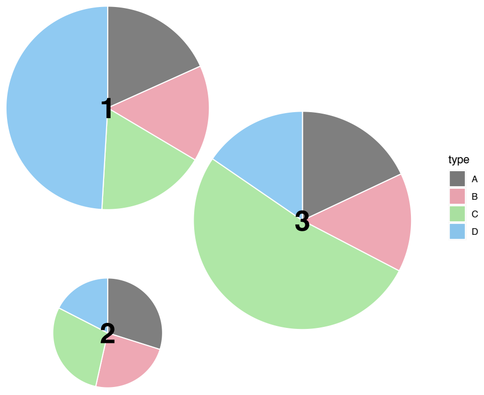
$$w_k^l = E_{\boldsymbol{u}|\Delta\boldsymbol{y};\hat{\boldsymbol{\psi}}}[u_{\alpha_l}^k] - E_{\boldsymbol{u}|\Delta\boldsymbol{y};\hat{\boldsymbol{\psi}}}[u_{\delta_l}^k] \tag{11}$$

where $u_{\alpha_l}^k$ and $u_{\delta_l}^k$ are the random-effects respectively for duplication and death of cell l for clone k. The diameter of the k-th pie is proportional to the euclidean 2-norm of

$$\boldsymbol{w}_k = (w_k^{l_1}, \dots, w_k^{l_n}) \tag{12}$$

where n is the number of cell types. Therefore, the larger the diameter, the more the corresponding clone is expanding into the lineage associated to the largest slice. The package RestoreNet includes the function `get.scatterpie()` which returns a clonal piechart given a fitted random-effects model previously obtained with the function `fit.re()`. The following R code chunk illustrates how to obtain a clonal piechart with few lines of R code.

```
> re.res <- fit.re(theta_0 = null.res$fit$par,
                   Y = Y,
                   rct.lst = rcts,
                   maxemit = 100) ## random-effects model fitting
> get.scatterpie(re.res, txt = TRUE) ## get the clonal piechart
```



References

[1] L. Del Core, M. A. Grzegorczyk, and E. C. Wit, "Stochastic inference of clonal dominance in gene therapy studies," *bioRxiv*, 2022. doi:10.1101/2022.05.31.494100.