Phylodynamics Simulation of Phylogenetic Trees

System analysis/flow

Hsiang-Yu Yuan

Date: July 22, 2013

Version: 1.2

Update: 13 July 2015

Section:

Background

System Architecture

System Design

Package Descriptions

Program Flow

The document of System analysis/flow describes the physical location of the files, flow of the algorithms at the system level

Background:

This document describes the system design and program flow of the new project based on the previous project entitled *critical analysis of binding avidity hypothesis*. The previous project ends in early, 2014. The files are located in ‘BindingSim’ and ‘binding\_v3’.

Package Descriptions:

There are two packages. One is for developing and the other is for running on the clusters.

Each package is stored in a separate folder. For example, the “BindingAvidity/mat/Mbindingavidity/” stores the files for ‘binding avidity simulation’, and the package “InferPhylo/make\_infection\_tree/” stores the code to reconstruct the virus transmission tress.

Packages for modules developing:

* BindingAvidity/mat/Mbindingavidity: Phylodynamic simulation of viral infection using Individual-based model.
* InferPhylo/make\_infection\_tree: Inferring phylogenetic trees based on finding MRCA in each lineage.
* tradeoff: Show the viral within host fitness

Packages for release version:

* binding\_v3
  + binding\_ode: ODE to quickly calculate stationary status
  + main
    - calbinding: calculate binding avidity on the transmission tree
    - infertree: infer the transmission tree
    - job: script for job submitting on clusters
    - plotfig: plot simulated lineages
    - main\_simulation.m
  + mainout
    - ibms: output of stochastic simulation
    - ode: output of deterministic simulation
    - tree: output of transmission tree
  + make\_infection\_tree: resample and generate the transmission tree. Packaged is distributed from “InferPhylo/make\_infection\_tree/”
  + Mbindingavidity: The packaged is distributed from “BindingAvidity/mat/Mbindingavidity/”
  + tradeoff

System Architecture (for modules developing)

3 modules in maintained in 3 separate folders. Once the packages are done, copy to the integrated release version folder.

BindingSim

tradeoff

InferPhylo

BindingAvidity

make\_infection\_tree

Mbindingavidity

trees

Viruses

System produces 3 objects into ‘out’ folder:

1. dat\_x\_trans\_tmp.mat: to reconstruct the transmission tree
2. DataTLSIR.mat: to store SIR dynamics
3. virus\_traits.mat: to store virus traits

System Flow

Mbindingavidity generate virus array

make\_infection\_tree produce the virus transmission tree first.

Using the transmission tree and virus traits, we can produce virus annotated tree.

**main\_binding.m**

*make\_infection\_tree*

*Mbindingavidity*

Nexus, Newick, mat

*copy*

Tree

**main\_simulate\_genealogy\_indiv\_binding**

dat\_x\_trans\_tmp.mat

virus\_traits.mat

Traits

viruses

*tau\_leap*

virus\_traits.mat

Mbinding/cal\_meanbding.m

Mean\_binding.mat

plot\_simulation.m

Flow of tauleap:

main\_binding.m

tauleap\_singlesir\_ibm\_matrix.m

Step1: Declare global variables

Step2: Simulation of the events

Step2.1: Iteration of Tau-leap

Step2.2: Output

tauleap\_singlesir\_ibm\_matrix.m

**Rate properties:**

|  |  |
| --- | --- |
| Rate\_Birth | Birth rate |
| Rate\_Death\_S | Death rate for S |
| Rate\_Death\_I | Death rate for I |
| Rate\_Death\_R |  |
| Rate\_Recovery |  |
| Rate\_Wanning |  |

Iteration of Tao Leap Algorithm

1. Setup the number of current viruses
2. Updating Viruses phenotypes
   1. Add mutational effect to each current virus
   2. Calculate binding changes
   3. obtain new immune status k from total antigenic change
3. Calculate transmission rate beta
4. Update epidemiological events
   1. Birth rate
   2. Death rate for S, I and R
   3. Recovery rate
   4. Wanning rate
5. Calculate the number of events
6. Update disease statuses
7. Updating transmission for each viral strain

Detailed steps:

Initialize variables and parameters

* Declare global variables
* Load deltaV Matrix
* Initialize Parameters
* Initialize virus strains

if metadata.ibms.initFlag == 1

create viruses

if metadata.ibms.initFlag == 2

use previous viruses

Simulation of the events

Iteration of Tao Leap Algorithm

1. Setup the number of current viruses
2. Updating Viruses phenotypes
   1. Update mutational effect to each current virus
   2. Update binding changes
   3. Update new immune status k from total antigenic change
3. Calculate transmission rate beta
4. Update epidemiological events
   1. Birth rate
   2. Death rate for S, I and R
   3. Recovery rate
   4. Infection
   5. Wanning rate
5. Calculate the number of events
6. Update disease statuses
7. Updating transmission for each viral strain

Auxiliary functions

Producing Output