

**CLASSES**

Notation: camelcase with capitalization on the first letter denotes the class, and camelcase with a lower case first letter denotes an instantiation of the class (i.e. ImmuneState is the class, immuneState is the particular immune state within a human object)

**PfPedigree**

The PfPedigree contains a list of attributes of every clonal lineage of Pf parasite, specifically the pfid, genotype, phenotype, a list of Human ids (ixh) of every Human infected with this specific clonal variant, the times of the beginning and end of each Human infection, as well as a list of Mosquito ids (ixm) and times of infection (they are assumed to be infected for the duration of their lives)

The pedigree is referenced at the creation of a new Pf object – the ids of the micro- and macrogametocytes are used to look up the parent genotypes, which are then recombined following our genetic algorithm to create the genotype of the new clonal variant.

Fields:

gtype – list of real-valued vectors within the unit n-cube ordered by pfid; vector is of

length nAntigenLoci

ptype – list of integer-valued vectors with component values for component i contained

in (1,nptype[i])

ixh – list of human ids infected by a particular parasite clonal lineage with given pfid

ixm – list of mosquito ids infected by a particular parasite clonal lineage with given pfid

mac – list of ids of macrogametocyte lineage for a given pfid

mic – list of ids of microgametocyte lineage for a given pfid

nAntigenLoci – integer, number of antigens we’re considering in the genotype vector

nptypes – integer-valued vector, number of distinct phenotypes for each of the antigens

of interest

PedLength – integer, current number of distinct clonal lineages on the part of the

landscape the pfPedigree is instantiated in

sib – sibling type (may not be used)

th – list of times a particular human was infected by a clonal lineage with given pfid

thEnd – list of times a particular human cleared a clonal lineage with given pfid

tm – list of times a particular mosquito was infected by a clonal lineage with given pfid

tmEnd – list of times a particular mosquito died/cleared a clonal lineage with a given

pfid

Methods:

initialize( ) – called automatically at instantiation of object, sets default values for

nAntigenLoci and nptypes

get\_\*field\*( ) - accessors for each of the above fields, no input necessary. Output is the

value of the field listed in the method

**Human**

The Human class acts as an environmental container for the immuneState, healthState, and Pathogen objects. To instantiate a human object, one needs to input the unique human id (ixH), the human’s age, the sex (M for male, F for female), and the location id. Creating a human object also queues the creation of a set of internal fields that are instantiations of ImmuneState, HealthState, and Pathogen (see diagram). Any updates can be queued in the Human, and the cross-talk between the different internal objects are done through function calls in the Human with inputs using values accessed within the Human.

Fields:

ixh – unique human id

age – age of human

sex – sex of human (‘M’ for male, ‘F’ for female)

locH – current location id of human

immuneState – immuneState object, see ImmuneState class description

healthState - healthState object, see HealthState class description

pathogen - pathogen object, see Pathogen class description

history – list of past values for various fields

Methods:

initialize( ixh, age, sex, locH ) – initialization function, sets id, age, sex, and location to

given values at declaration

get\_\*field\*( ) – accessor method that returns value of a given field

set\_\*field\*( newFieldValue ) – setter method that replaces field value with a new given

value

infectHuman( t, pfid ) – infect human with a given pf object listed in the pfPedigree

clearPathogen( t, pfid ) – remove the pf object when densities go to zero

infectMosquito( t, pfid, ixm ) – infect mosquito with a given pf object listed in pfPedigree

moveHuman( newlocH ) – move human to new location

updateHuman( t ) – updates human given current time t – updates all internal fields,

including the health/immune states and pathogens

**ImmuneState**

An immuneState object contains lists of immune counters for general immunity, as well as an array of times when a particular phenotype was last seen within the human.

Fields:

nBSImmunecounters – number of different blood-stage immune counters

BSImm – current value of each blood-stage immune counter

wx – vector of waxing rates for each blood-stage immune counter

wn – vector of waning rates for each blood-stage immune counter

dxp - parameter that weights effect of phenotypic difference on type-specific immunity

dtp – parameter that weights effect of temporal difference on type-specific immunity

GenImm – value of total general immune effect of all the counters

ptypesTime – matrix/array of previous time(s) a phenotype was seen by the human

typeImm – overall effect of type-specific immunity to a given type

nptypes – vector of integers, number of phenotypes – read from pfPedigree

history – list of all previous values of the fields

Methods:

initialize( ) – initializing function, automatically called when object is instantiated

get\_\*field\*( ) - accessor method that returns value of a given field

set\_\*field\*( newFieldValue ) – setter method that replaces field value with a new given

value

update\_immuneState( t, Ptot ) – updates all fields of the immune state given current

time and total parasite density

update\_history( ) – updates the historical record of field values

sigmoidX( X, X50, Xs, atMax ) – sigmoidal function – takes in x-value, halfway point,

slope parameter, and first x-value at which it reaches its maximum; domain for x is R

dynamicXdt ( X, P, PAR ) -

daysSinceUnder( X, P, PAR )

daysSinceOver( X, P, PAR )

antibodyRegister( t, pfid )

dynamicCounter( P, PAR )

gImPAR( wx, wn, P50, Ps, atMax, b, sigma )

ptype2Mat( ptype, nptypes)

shift( v, places, dir )

update\_typeImmunity( t, ptype )

crossImm( ptype, nptypes )

**HealthState**

[ insert HealthState class description here ]

Fields:

Fever – current fever state (either binary or actual temperature)

feverThresh – threshold Ptot above which a fever response will be induced

HRP2 – current log10 HRP2 levels

RBC – current RBC count

pLDH – current log10 pLDH levels

Rx – list of possible drugs a person can take

history

Methods:

initialize( ) – initializing function, automatically called when object is instantiated

get\_\*field\*( ) - accessor method that returns value of a given field

set\_\*field\*( newFieldValue ) – setter method that replaces field value with a new given

value

update\_healthState( Ptot, RBCHist )

update\_Fever( Ptot )

update\_HRP2( Ptot )

update\_pLDH( Ptot )

update\_RBC( Ptot, RBCHist )

update\_history( )

RDT( )

HSRDT( )

PCR( )

LAMP( )

LightMic( )

sigmoidX( X, X50, Xs, atMax )

Treat( Rx )

**Pathogen**

[ insert Pathogen class description here ]

Fields:

PfPathogen – list containing pf objects (see Pf class description) that classify current pf

infection state within human

Ptot – sum of all individual asexual clonal lineages currently infecting human

Gtot – sum of all individual gametocyte clonal lineages currently infecting human

PfMOI – current multiplicity of infection within human object (i.e. sum of activeP’s)

history -

Methods:

initialize( ) – initializing function, automatically called when object is instantiated

get\_\*field\*( ) – accessor method that returns value of a given field

set\_\*field\*( newFieldVal ) – setter method that replaces field value with a new given

value

add\_Pf( t, pfid) ## currently include more inputs, but can look up from pfped

remove\_Pf( t, pfid )

update\_pathogen( t )

update\_Ptot( t )

update\_Gtot( )

update\_history( )

log10sum( x )

log10vals( x )

**Pf**

[ insert Pf class description here ]

Fields:

pfid

PAR

gdk

activeP – binary value – 1 if asexual parasite numbers of a particular clonal strain within

the human host are nonzero, 0 if they’ve been cleared

active – binary value – 1 if gametocyte numbers of a particular clonal strain within the

human host are nonzero, 0 if they’ve been cleared

Pt – asexual parasite numbers

Ptt – by default a 10-day rolling record of previous Pt values, used to simulate the delay

to correspond with the Gt maturation period

Gt – gametocyte parasite numbers

mic – microgametocyte id, used to look up the microgametocyte genotype in the

pfPedigree when calculating the new genotype for a pf

mac – macrogametocyte id, used to look up the macrogametocyte genotype in the

pfPedigree when calculating the new genotype for a pf

gtype – vector of length n=nAntigenLoci within the unit n-cube, genotype of the clonal

lineage

ptype – integer-valued vector of length n=nAntigenLoci, computed using the gtype

mu – parameter dictating the per-locus mutation rate

Methods:

initialize( mic, mac, pfid, seed=F ) - initializing function, automatically called when object

is instantiated – sets mic, mac, pfid. If seed=F, will use mic/mac to create new

gtype; otherwise it will create one at random to help ‘seed’ the pf population

get\_\*field\*( ) - accessor method that returns value of a given field

set\_\*field\*( newFieldVal ) – setter method that replaces field value with a new given

value

getGtype( mic, mac, mu, seed=F )

getPtype( gtype, nptypes )

mutate( gtype, mu )

update\_Pf( t ) – calls the update methods for Pt, Ptt, Gt – usually called from the

pathogen object, which is part of an update\_pathogen method from the human

level

update\_Pt( t ) – updates the asexual parasite population – default uses dPdt\_tent, a

dynamically updated tent function

update\_Ptt( t ) – updates a historical list of asexual parasite population – shifts the list

back by a day and adds the new value. By default holds 10 days of history. This is

used to update Gametocyte populations

update\_Gt( t ) – updates the gametocyte population using the rolling record of asexual

parasite densities Ptt

GamCyGen( t, P, PAR ) – gametocytogenesis, called by the update\_Gt method to

determine the relationship between the rolling record of asexual parasite

densities and gametocyte creation

Pf.MaxPD( N, mn, vr )

Pf.PeakD( min )

Pf.MZ0( )

Pf.Duration( peakD, N, mn )

tentPAR( t, pfid )

gr\_tent( t, PAR )

dPdt\_tent( t, P, PAR, PD=0, IM = 0 )

shift( v, places, dir=”right”)

log10sum( x )

log10vals( x )

immuneMod\_tent( BSImm )

sigmoid01( x, xh, b, max )

Typical Pf lifecycle in PfLOME:

1. Mosquito probes host and bites an infected human
2. Gametocytes are sampled from the human, weighted by their relative Gt numbers
3. The mic and mac ids are retrieved, looked up in the pfpedigree, and a new pf object is created using the mic and mac gtypes
4. \*\* current idea – a similar tent function is queued in the mosquito that is delayed from the biting event corresponding to sporozoite densities in the mouth of the mosquito – only the pfids of the new pf objects are stored in the mosquito
5. When the mosquito bites another human, n pfids are sampled with relative weights of their sporozoite values – the n is randomly drawn with a mean increasing with sporozoite densities
6. This queues a tent function in the human of asexual Pt, and corresponding dynamics of gametocytes Gt, biomarkers, and immune counters
7. The tent function parameters are modified at the start of the new infection by the current levels of immune counters
8. Fever events are currently tied to parasite densities – if Pt exceeds feverThreshold, then this can queue care-seeking behavior and increases probabilities of diagnostic test taking/Rx administration to curb the densities
   1. Rx has a pharmacodynamic (PD) daily log10 killing effect specific to each class of drug
9. If the human is bitten again while they have a nonzero number of gametocytes, go to step 1

Example of use (walkthrough of object\_practice.R)

1. Create a pedigree on the tile:

pfped = PfPedigree$new( )

This instructs the PfPedigree class to create a new object we call pfped – no input is required for this instantiation

1. Populate your human population:

someGuy = Human$new( ixh = 1, age = 20, sex = ’M’, locH = 7)

This creates an object from the human class with an id of 1, sets their age to be 20 years old, their sex to be male, and puts them in the location with id 7. This automatically initiates the instantiation within the newly created human environment objects called pathogen, immuneState, and healthState from their respective classes

1. Populate your parasite population:

pf = Pf$new( mic = 1, mac = 1, pfid = 1, seed = T)

This creates an object from the parasite class with a pfid of 1; seed = T means we’re seeding our population, so it creates a genotype randomly without using any prior data from mic and mac information, but values are still required for now. If seed = F (which is the default) then mic and mac ids will be used to look up the parents’ genotypes, and the new genotype will be calculated from those. Tent function parameters are also drawn at instantiation of the object, which will be modified by the current immune state of whichever human host it eventually infects.

Importantly we must also add the newly created pf object to the pedigree:

pfped$add2Pedigree(pf)

This uses the pedigree’s method and takes the pf object as input to store its information – it can be referenced as necessary by any other object, and some components (such as when a human is infected by that object) can be modified.

1. We can infect someGuy from the human population at time t with our pf:

pfid = pf$get\_pfid( )

someGuy$infectHuman( t, pfid )

First we accessed our pfid through the publicly accessible method in the pf object. Then we used the current time and the pfid of the pathogen to infect the human; both the t and ixH will be recorded in pfped, and a tent function will be queued in the human corresponding to the asexual parasite population.

1. We can update the infection status of the human (including Pt, Gt, biomarkers, fever, immune markers, etc) through the command

someGuy$updateHuman( t )

This uses the current time to update Pt through dPdt, and all of the other components are updated in turn. The history for each internal object is also updated automatically, so we have a rolling list of every past value of each field of interest.

1. Finally we can plot the results after the human object has been modified by accessing the history list inside the human. For example, if we want to plot Pt over time, we can plot it through the following command:

history = someGuy$get\_history( )

Pt = history$Pt

plot( time, Pt )

This extracts the history list from the human, then extracts Pt from history, then plots it; of course this can also be done on a single line:

plot( time, someGuy$get\_history( )$Pt )