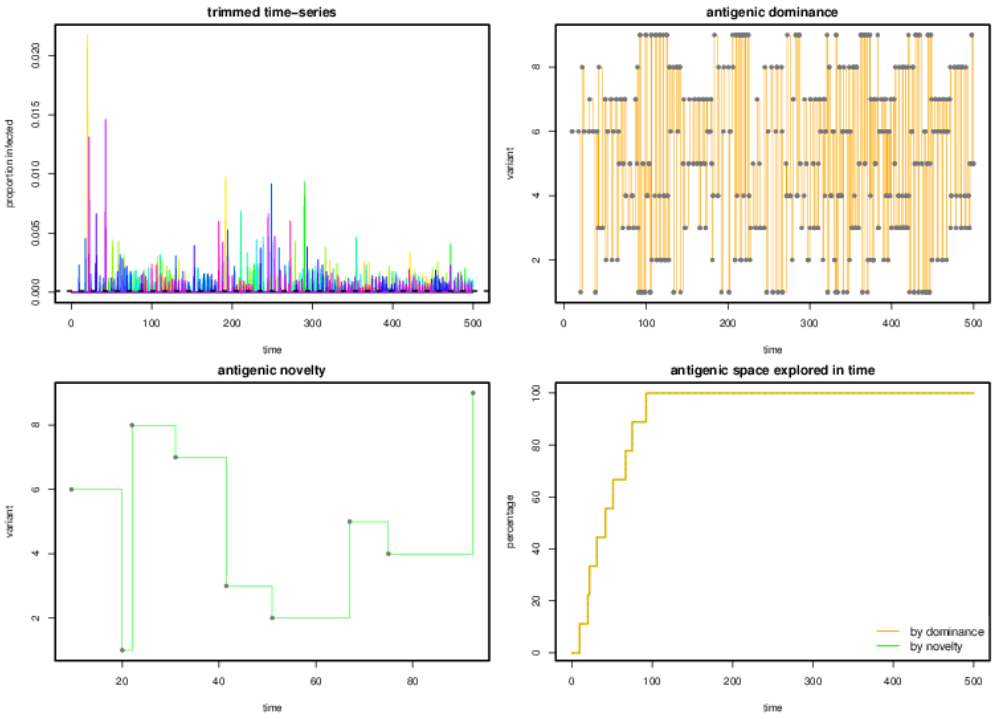
Upgrade to V3.0 (red egg) - summary of new functionalities.

Optimized C code and compilation:

MANTIS's C-core code has been optimized and is now set by default to be compiled using the -O3 option during installation. This has resulted in significant reductions in computation time. If a developer, make sure your Makevars config file is set properly (read src/Makevars) within the MANTIS source package.

Starting code to generate CSS dynamics:

```
library(MANTIS)

#define epitope structure
epiStruc<- c(3,3)

#get number of strains in the system
nStrains<- extractNumberStrains(epiStruc)
#time at which the solver will stop
tMax<- 500
#solver step
tInt<- 0.005
#record solution every tObsPer steps
tObsPer<- 100
#random initial conditions for infected
infInitCond <- runif(nStrains, 0, 1)*1e-9

#parameter values here defined by year
#thus output time scale will be years
beta<- 292
sigma<- (1/5)*365 #5 days infectious period
mu<- 1/50 #50 years of life-span

#running model with weak cross-immunity
gamma<- 0.75
simdata<- runMANTIS(epiStruc, tMax, tObsPer, tInt, infInitCond, beta, gamma, sigma, mu)

#plotting antigenic dynamics
plotAntigenicSpaceDyn(simdata, extThreshold=1e-4, fileout='antigenicspace.pdf')
```



(*New*) Extracting the 'allelic matrix':

The 'allelic matrix' is a matrix that presents the strain number and 'sequence' of each variant. The 'sequence' is the unique combination of alleles at each loci that defines each variant. This matrix can now be extracted from the output of the function runMANTIS (in the example below, the output is in the variable simdata). This can be done by calling the function extractAllelicMatrix. The result is a matrix in which the columns are the exsiting loci. In each locus' column, integers are used to represent the alleles.

```
#extract 'allelic matrix'
allelicMatrix= extractAllelicMatrix(simdata)

#extract the sequence of one strain (ex: strain 3)
allelicMatrix[3,]

result:

     locus1  locus2

3      3          1
```

(*New*) Exporting the 'allelic matrix':

Exporting the allelic matrix is possible with the following function:

```
#export the allelic matrix
exportAllelicMatrix(simdata, fileout='allelic.matrix.csv')
```

A new file, in this case "allelic.matrix.csv', is created with the information described in “Extracting the allelic matrix'.

(*New*) Extracting the sequences of dominant strains in time:

The sequences of the dominance strains can now be extracted from the output of the function runMANTIS (in the example below, the output is in the variable simdata). This can be done by calling the function `extractSeqOfDomVariant`. The result is a matrix in which the first column is time, the second the variant that is dominating and the remaining columns the existing loci. In each locus' column, integers are used to represent the alleles.

```
#build matrix that has the sequences of each dominant strain in time
seqsOfDomVariant= extractSeqOfDomVariant(simdata, extThreshold=1e-50)

#have a look at which sequences were dominant in time (first 10 for simplicity)
seqsOfDomVariant[1:10,]

result:

        time           domvariant     locus1  locus2

1       0.000          9              3       3

2       0.500          6              3       2

3       1.000          6              3       2

4       1.505          6              3       2

5       2.005          6              3       2

6       2.505          6              3       2

7       3.005          6              3       2
```

```
8       3.505           6               3       2
9       4.005           6               3       2
10      4.505           6               3       2
```

(*New*) Exporting the sequences of dominant strains in time:

Exporting the sequences of the dominant strains in time is possible with the following function:

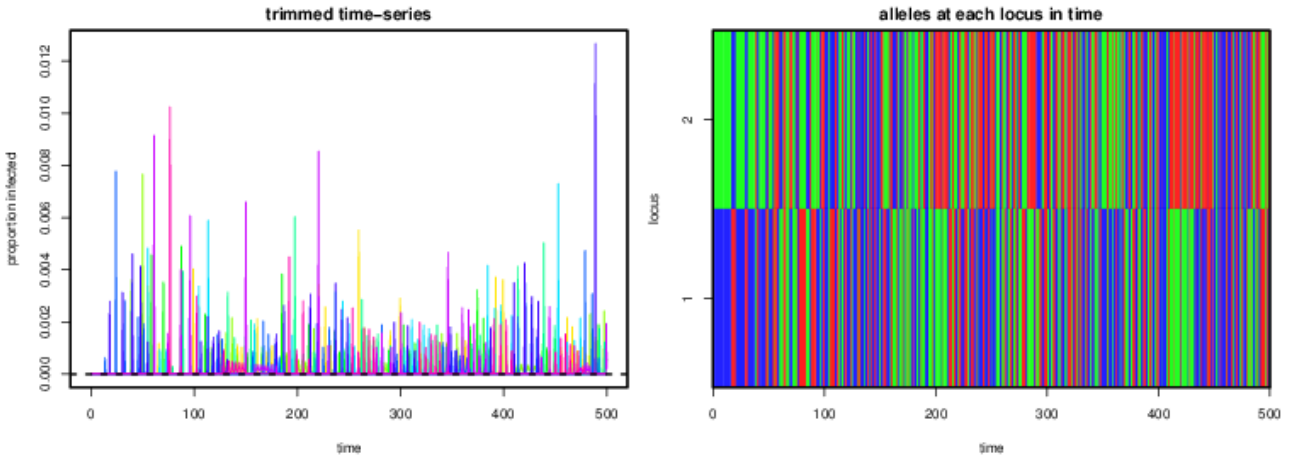
```
#exporting the sequences
exportSeqOfDomVariant(simdata, extThreshold=1e-50,  fileout='dominant.sequences.time.csv')
```

A new file, in this case "dominant.sequences.time.csv', is created with the information described in “Extracting the sequences of the dominant strains in time'.

(*New*) Plotting the sequences of dominant strains in time:

The sequences of the dominance strains can now be plotted from the output of the function runMANTIS (in the example below, the output is in the variable simdata). This can be done by calling the function `plotSeqsOfDomVariant`. The result is a composed figure with two subplots. The first subplot is the time series of the Y calls for all strains. The second plot is a colour map in which each row is an existing locus and colours represent alleles.

```
##plot matrix that has the sequences of each dominant strain in time
extThreshold=1e-50
plotSeqsOfDomVariant(simdata, extThreshold=extThreshold, epiStruc=epiStruc, fileout=
'allelesintime.pdf' )
```

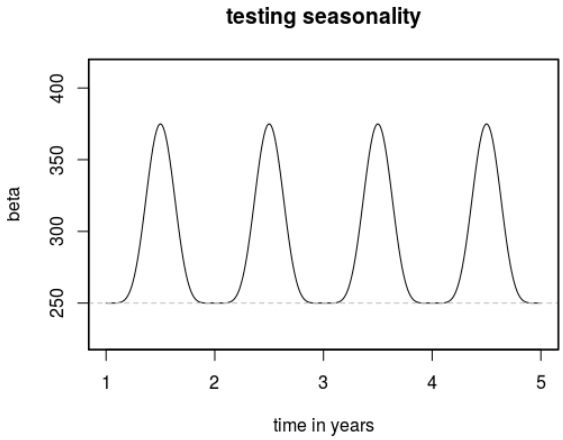


(*New*) Seasonal forcing in transmission rates

It is now possible to introduce seasonal forcing into transmission rates. All strains suffer the same forcing, although they can still have different transmission base lines (beta). The parameter epsilon has been added to the call of runMANTIS; it represents the desired increases to the betas during seasons; if epsilon=0 then seasonal effects are not modelled, which is the default (see example below).

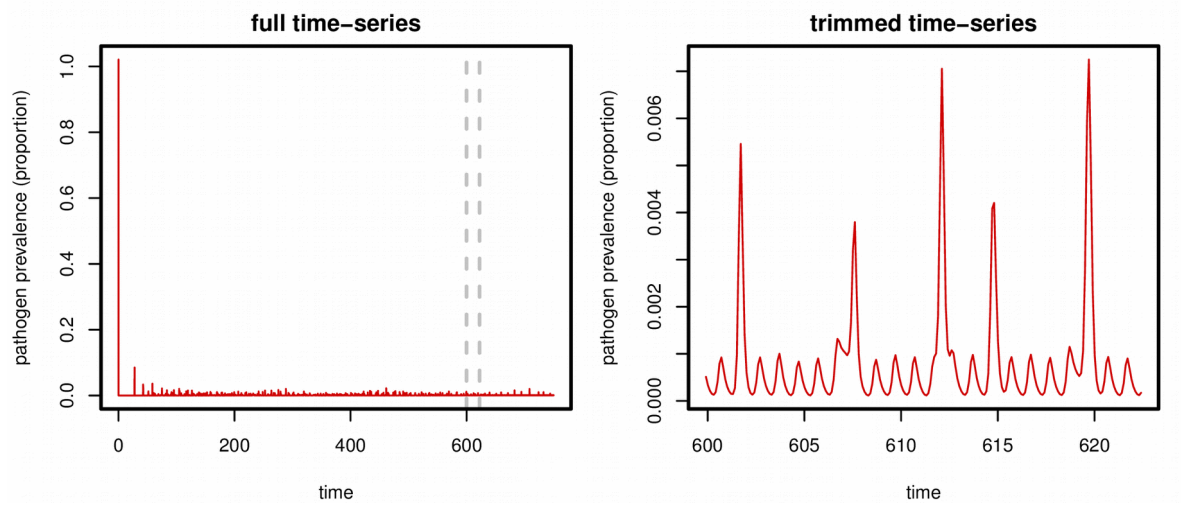
The function testSeasonality has been added to visualize the assumed seasonal dynamics of a given beta. This function plots, by default, the resulting seasonal dynamics:

```
#generates and increase of 50% to a baseline beta of 250
testSeasonality(beta=250, epsilon=0.5, time=seq(1,5,0.01))
```



A full example of dynamics with seasonality can be found below under NSS dynamics.

```
#define antigenic structure
epiStruc<- c(2,2)
#time at which the solver will stop
tMax<- 750
#solver step
tInt<- 0.01

#record solution every tObsPer steps
tObsPer<- 10

#parameter values here defined by year
#thus output time scale will be years
beta<- 292
sigma<- (1/5)*365 #5 days infectious period
mu<- 1/50 #50 years of life-span

#running model with weak cross-immunity
gamma<- 0.01
epsilon<- 0.25
nLocus<- length(epiStruc)
nStrains<- extractNumberStrains(epiStruc)
infInitCond <- runif(nStrains, 0, 1)*1e-4

simdata<- runMANTIS(epiStruc, tMax, tObsPer, tInt, infInitCond, beta, gamma, sigma, mu, epsilon)
plotPathogenPrevalence(simdata, xiObs=0.8, xfObs=0.83, fileout='example.season.pdf')
```



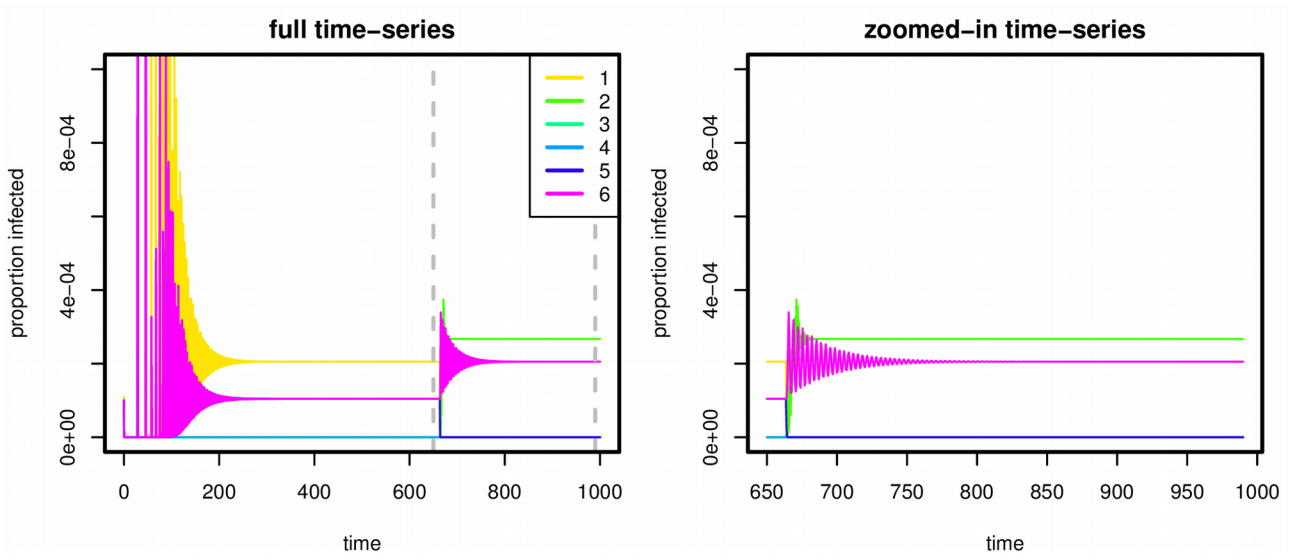
(*New*) Changes in the transmission phenotype of a strain

In versions V1 and V2, all parameters had to be fixed for the course of a simulation. It is now possible to run simulations in which the transmission potential of a strain changes at a desired time point. For this, the following function has been developed:

```
runInvasionWithOneBetaChange(epiStruc, tMax, tObsPer, tInt, initCondY, beta, gamma, sigma, mu,
tBetaChange, changedBeta, changedStrain)
```

Here, most of the parameters are the same as when calling runMANTIS (see ?runMANTIS). However, the last 3 parameters - tBetaChange, changedBeta, changedStrain – are used to choose a time step for the change, the new value of beta and the strain which will suffer the change, respectively (for details, see the documentation by running ?runInvasionWithOneBetaChange). The following code runs a complete example of this new functionality:

```
epiStruc<- c(3,2) #define epitope structure, with 2 loci
nStrains<- extractNumberStrains(epiStruc) #get number of strains in the system
tMax<- 1000 #maximum time for the simulation
tInt<- 0.005 #time step for the simulation
tObsPer<- 100 #record solution every tObsPer steps

initCondY <- rep(1e-4, nStrains) #random initial conditions for infected
initCondY[1]<- initCondY[1]+ 0.00001 #strain 1 set with a slightly higher prevalence

beta<- rep(292, nStrains) #default transmission for all strains
gamma<- 0.98 #cross immunity for this simulation
sigma<- (1/5)*365 #5 days infectious period
mu<- 1/50 #50 years of life-span

#define here the extra parameters for this experiment
tBetaChange= 650 #time at which beta will be changed
changedStrain= 2 #which strain to change beta for
changedBeta= 292*10

#run MANTIS
simdata<- runInvasionWithOneBetaChange(epiStruc, tMax, tObsPer, tInt, initCondY, beta, gamma, sigma,
mu, tBetaChange, changedBeta, changedStrain)

#plot entire simulation
plotY(simdata, xiObs=0.65, xfObs=0.99, ymax=0.001, addLegend=TRUE)
```



Thanks for your support,

EEID group.