dynamAedes vignette

Daniele Da Re, Matteo Marcantonio

15th December 2021

This tutorial provides a step-by-step guide to the **dynamAedes** package, a unified modelling framework for invasive *Aedes* mosquitoes. In this package, the users can apply the stochastic, time-discrete and spatially-explicit population dynamical model developed in Da Re et. al 2021 (DOI: https://doi.org/10.1016/j.ecoinf.2020.101180) for *Aedes aegypti* mosquitoes and now expanded for other three species: *Ae. albopictus*, *Ae. japonicus* and *Ae. koreicus*

This stage-based model is informed by temperature and photoperiod and can be applied to three different spatial scales: punctual, local and regional. These spatial scales consider different degrees of spatial complexity and data availability, by accounting for both the active and passive dispersal of the modeled mosquito species as well as for the heterogeneity of temperature data.

We will present the application of each scale of the model using a simulated temperature dataset for the species *Ae. albopictus*

```
#Packages for processing
library(raster)
library(sp)
library(gstat)
library(spatstat)
library(maptools)
library(rgeos)
library(parallel)
library(eesim)
library(tidyverse)
library(dynamAedes)

#Packages for plotting
library(ggplot2)
library(geosphere)
```

Punctual scale model

At punctual scale the model only requires a weather station temperature time series provided as a numerical matrix with temperatures in Celsius. For the purpose of this tutorial, we will simulate a three-year long temperature time series.

Simulate temperature data with seasonal trend

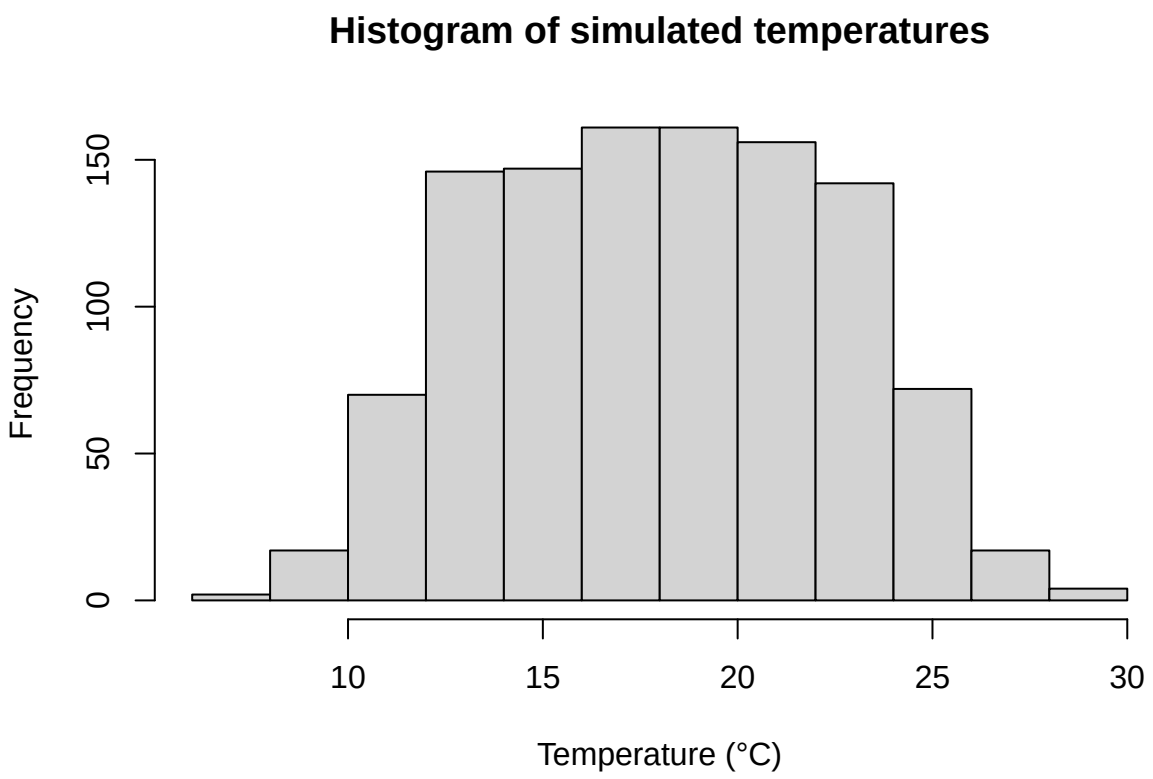
Next, we simulate a two-year temperature time series with seasonal trend. For the time series we consider a mean value of 18°C and standard deviation of 3°C.

```
ndays = 365*3 #length of the time series in days
set.seed(123)
sim_temp <- create_sims(n_reps = 1,
```

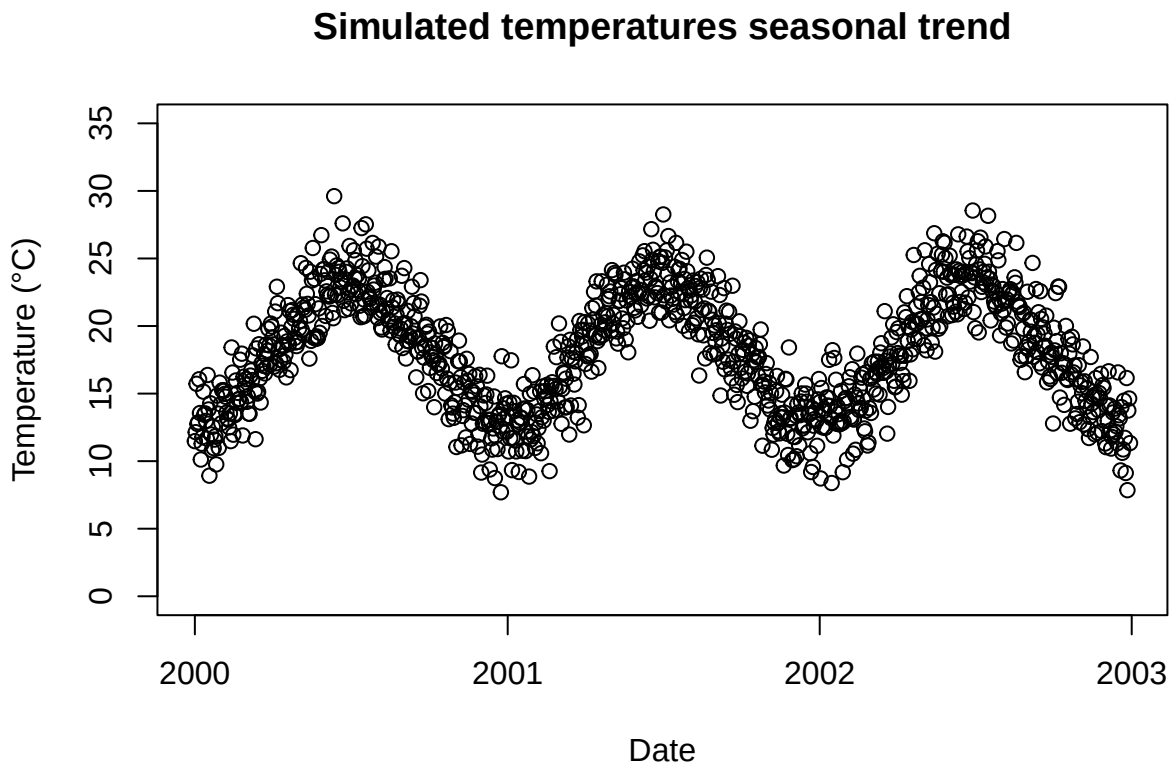
```
                        n = ndays,
                        central = 18,
                        sd = 2,
                        exposure_type = "continuous",
                        exposure_trend = "cos1", exposure_amp = -.3,
                        average_outcome = 12,
                        outcome_trend = "cos1",
                        outcome_amp = 0.8,
                        rr = 1.0005)
```

We can now observe the temperature values distribution and the its temporal trend.

```
hist(sim_temp[[1]]$x,
     xlab="Temperature (°C)",
     main="Histogram of simulated temperatures")
```




```
plot(sim_temp[[1]]$date,
     sim_temp[[1]]$x,
     main="Simulated temperatures seasonal trend",
     xlab="Date", ylab="Temperature (°C)", ylim=c(0,35))
```



Format the simulated input datasets and run the model

Model settings

Float numbers in the temperature matrix would slow the computational speed, thus we first multiply them by 1000 and then transform them in integer numbers. We also transpose the matrix from long to wide format, because we conceptualize the module structure considering the rows as the spatial component (here = 1) and the columns as the temporal one.

```
df_temp=data.frame("Date" = sim_temp[[1]]$date, "temp" = sim_temp[[1]]$x)
w <- t(as.integer(df_temp$temp*1000))
```

We can now define the model parameters.

```
## Define the day of introduction (day 1)
str = 121
## Define the end-day of life cycle
endr = 121+365
## Define the number of eggs to be introduced
ie = 500
## Define the number of model iterations
it = 10 # The higher the number of simulation the better
## Define the number of liters for the larval density-dependent mortality
habitat_liters=1
#Define latitude and longitude for the diapause process
myLat=42
myLon=7
## Define the number of parallel processes (for sequential itarations set nc=1)
cl = 5
## Set output name for the *.RDS output will be saved
outname= paste0("dynamAedes_albo_ws_dayintro_",str,"_end",endr,"_niters",it,"_neggs",ie)
```

Run the model

Running the model takes around 10 minutes with the settings specified in this example.

```
simout=dynamAedes.m(species="albopictus",
                    scale="ws",
                    ihwv=habitat_liters,
                    temps.matrix=w,
                    startd=str,
                    endd=endr,
                    n.clusters=cl,
                    iter=it,
                    intro.eggs=ie,
                    compressed.output=TRUE,
                    lat=myLat,
                    long=myLon,
                    suffix=outname,
                    verbose = FALSE)
```

Analyze the results

We first explore the model output structure: the *simout* object is a nested list.

The **first** level corresponds to the number of model iterations

```
print(it)
```

```
## [1] 10
```

```
print(length(simout))
```

```
## [1] 10
```

The **second** level corresponds to the simulated days. So if we inspect the first iteration, we will observe that the model has computed 366 days, as we had specified above in the object *endr*.

```
length(simout[[1]])
```

```
## [1] 366
```

The **third** level corresponds to the amount of individuals for each stage (rows). So if we inspect the 1st and the 50th day within the first iteration, we will obtain a matrix having

```
dim(simout[[1]][[1]])
```

```
## [1] 4 1
```

```
simout[[1]][[1]]
```

```
##               [,1]
## egg            500
## juvenile       378
## adult            0
## diapause_egg     0
```

```
simout[[1]][[50]]
```

```
##               [,1]
## egg          10874
## juvenile      6208
## adult          402
```

```
## diapause_egg     0
```

We can now use the auxiliary functions of the model to analyze the results.

Derive probability of a successfull introduction at the end of the simulated period

First, we can retrieve the probability of successful introduction, computed as the proportion of model iterations that resulted in a viable mosquito population at a given date.

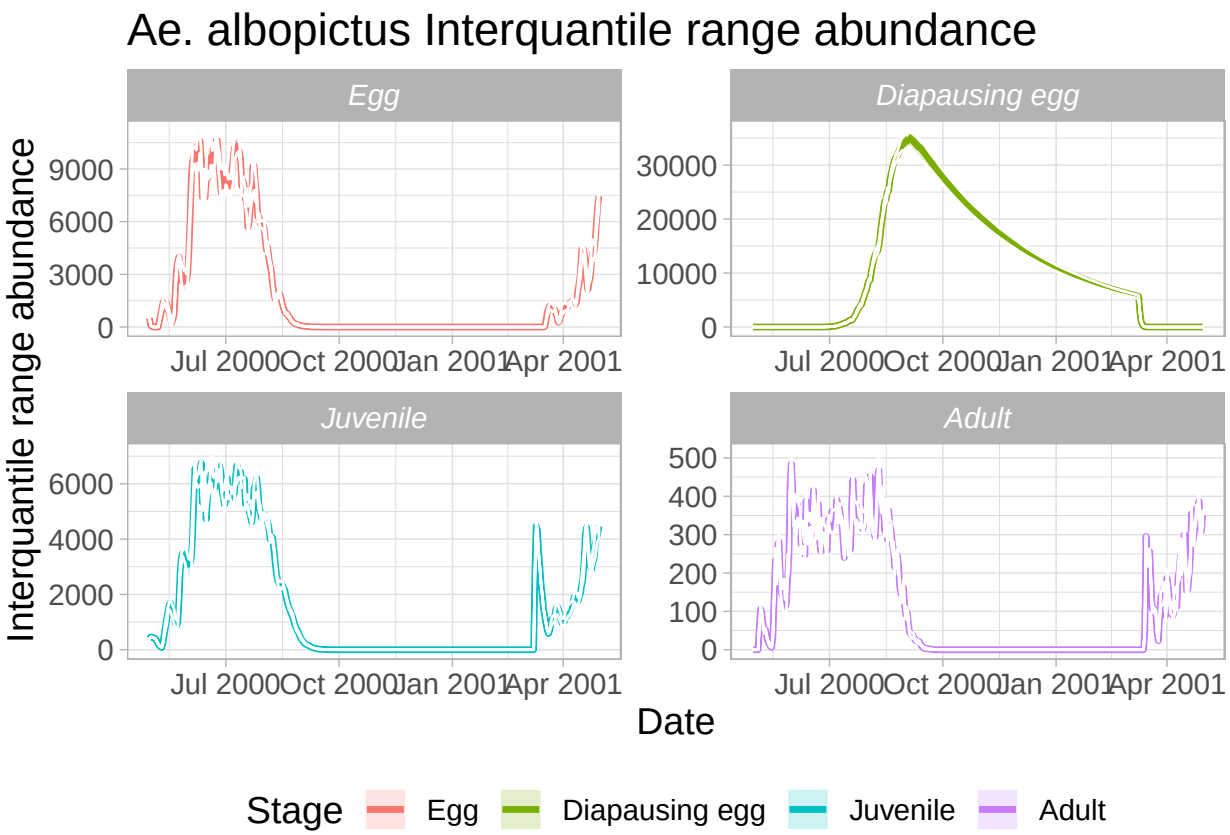
```
rbind.data.frame(psi(input_sim = simout, eval_date = 5),
                 psi(input_sim = simout, eval_date = 365))
```

```
##   Days_after_intro p_success      stage
## 1            Day 5         1 Population
## 2          Day 365         1 Population
```

Derive abundance 95% CI for each life stage and in each day

We can now compute the interquantile range abundance of the simulated population using the function *adci*.

```
dd <- max(sapply(simout, function(x) length(x)))#retrieve the maximum number of simulated days
egg<-as.data.frame(adci(simout, eval_date=1:dd, breaks=c(0.25,0.50,0.75), stage=1))
juv<-as.data.frame(adci(simout, eval_date=1:dd, breaks=c(0.25,0.50,0.75), stage=2))
ad<-as.data.frame(adci(simout, eval_date=1:dd, breaks=c(0.25,0.50,0.75), stage=3))
eggd<-as.data.frame(adci(simout, eval_date=1:dd, breaks=c(0.25,0.50,0.75), stage=4))

egg$myStage="Egg"
egg$Date=seq.Date(sim_temp[[1]]$date[str], sim_temp[[1]]$date[endr], by="day")
juv$myStage="Juvenile"
juv$Date=seq.Date(sim_temp[[1]]$date[str], sim_temp[[1]]$date[endr], by="day")
ad$myStage="Adult"
ad$Date=seq.Date(sim_temp[[1]]$date[str], sim_temp[[1]]$date[endr], by="day")
eggd$myStage="Diapausing egg"
eggd$Date=seq.Date(sim_temp[[1]]$date[str], sim_temp[[1]]$date[endr], by="day")

outdf=bind_rows(egg, juv, ad, eggd) %>%
  as_tibble()

outdf %>%
  mutate(myStage=factor(myStage, levels= c("Egg", "Diapausing egg", "Juvenile", "Adult"))) %>%
  ggplot( aes(y=(`50%`),x=Date, group=factor(myStage),col=factor(myStage))) +
  ggtitle("Ae. albopictus Interquantile range abundance")+
  geom_line(size=1.2)+
  geom_ribbon(aes(ymin=`25%`,ymax=(`75%`),fill=factor(myStage)),
              col="white",
              alpha=0.2,
              outline.type="full")+
  labs(x="Date", y="Interquantile range abundance", col="Stage", fill="Stage")+
  facet_wrap(~myStage, scales = "free")+
  theme_light()+
  theme(legend.pos="bottom",  text = element_text(size=14) , strip.text = element_text(face = "italic"))
```



Local scale model

The local scale allows the model to account for both active and passive dispersal of the mosquitoes. With this setting, the model requires three input datasets: a numerical matrix with temperatures in Celsius defined in space and time (space in the rows, time in the columns), a two-column numerical matrix reporting the coordinates (in meters) of each space-unit (cell) and a numerical *distance matrix* which reports the distance in meters between the cells connected through a road network. For the purpose of this tutorial, we will use the following simulated datasets:

1. A 10 km lattice grid with 250 m cell size;
2. A 2-year long spatially and temporally correlated temperature time series;
3. A matrix of distances between cells connected through a simulated road network;

Prepare input data

Create lattice arena

First, we define the physical space where the introduction of our mosquitoes will happen. We define a squared lattice arena having 10 km side and 250 m resolution (40 colums and 40 rows, 1600 total cells).

```
gridDim <- 40 # 10000m/250 m = 40 columns and rows
xy <- expand.grid(x=1:gridDim, y=1:gridDim)
```

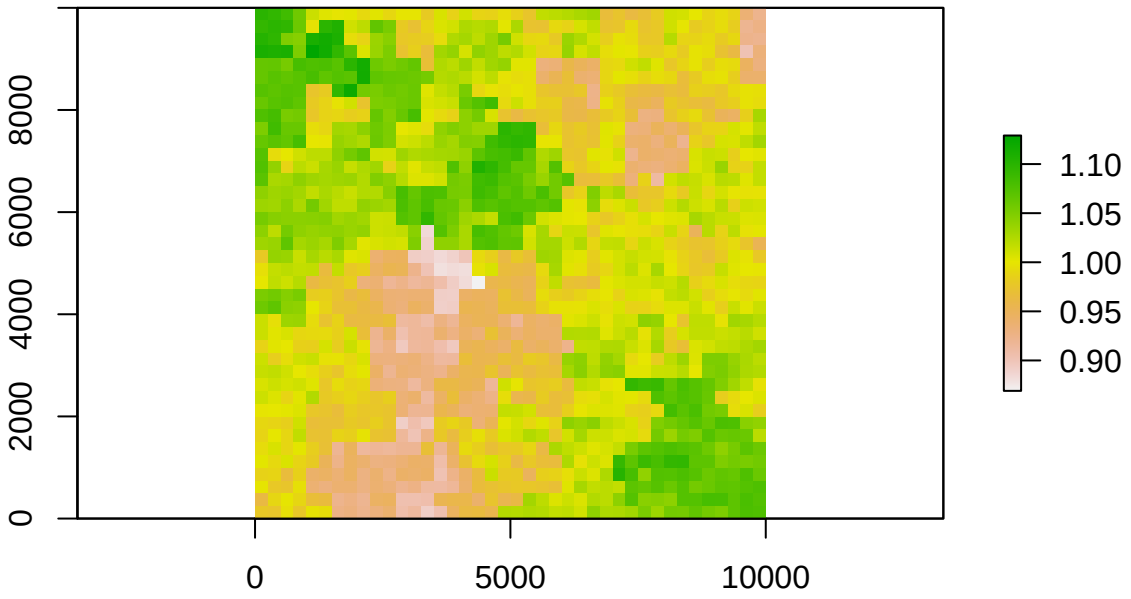
We then add a spatial pattern into the lattice area. This spatial pattern will be used later to add spatial correllation (SAC) to the temperature time series. The spatial autocorrelated pattern will be obtained using a semivariogram model with defined sill (value that the semivarion attains at the range) and range (distance of 0 spatial correlation) and then predicting the semivariogram model over the lattice grid using unconditional Gaussian simulation.

```
varioMod <- vgm(psill=0.005, range=100, model='Exp') # psill = partial sill = (sill-nugget)
# Set up an additional variable from simple kriging
zDummy <- gstat(formula=z~1,
                locations = ~x+y,
                dummy=TRUE,
                beta=1,
                model=varioMod,
                nmax=1)
# Generate a randomly autocorrelated predictor data field
set.seed(123)
xyz <- predict(zDummy, newdata=xy, nsim=1)
```

```
## [using unconditional Gaussian simulation]
```

We generate a spatially autocorrelated raster adding the SAC variable (*xyz$sim1*) to the RasterLayer object. The autocorrelated surface could for example represent the distribution of vegetation cover in a urban landscape.

```
utm32N <- "+proj=utm +zone=32 +ellps=WGS84 +datum=WGS84 +units=m +no_defs"
r <- raster(nrow=40, ncol=40, crs=utm32N, ext=extent(0,10000, 0,10000))
values(r)=xyz$sim1
plot(r)
```




```
df <- data.frame("id"=1:nrow(xyz), coordinates(r))
bbox <- as(extent(r), "SpatialPolygons")

# Store Parameters for autocorrelation
autocorr_factor <- values(r)
```

Simulate temperature data with seasonal trend

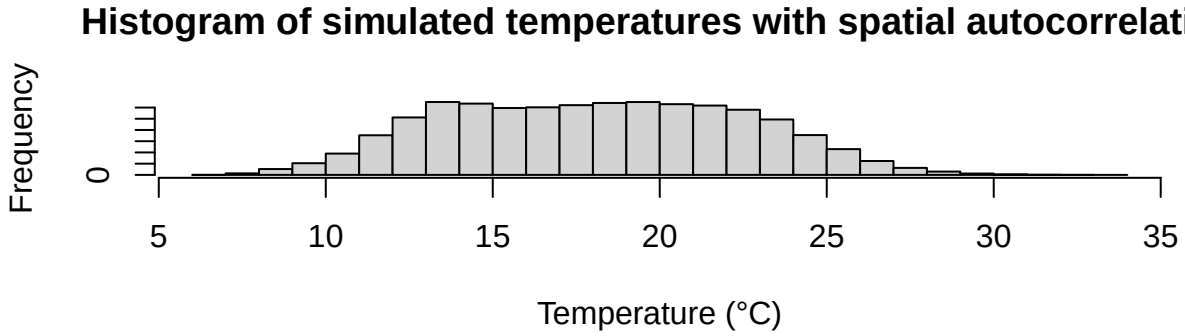
We take advantage of the temperature dataset simulated for the punctual scale modelling exercise. We can then "expand onto space" the temperature time series by multiplying it with the autocorrelated surface simulated above.

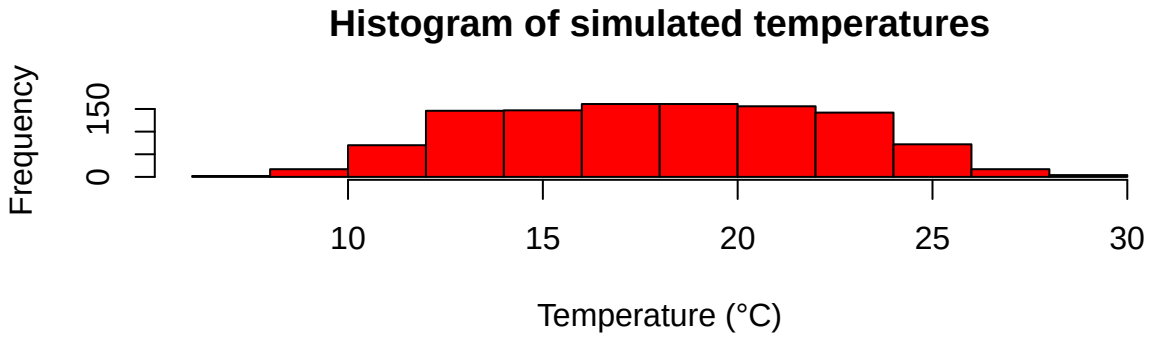
```
mat <- mclapply(1:ncell(r), function(x) {
    d_t <- sim_temp[[1]]$x*autocorr_factor[[x]]
```

```
    return(d_t)
},mc.cores=1) #mc.cores=1 in Windows OS, which does not support mclapply function

mat <- do.call(rbind,mat)
```

We can compare the distribution of the initial temperature time series with autocorrelated temperature surface one.

```
par(mfrow=c(2,1))
hist(mat, xlab="Temperature (°C)", main="Histogram of simulated temperatures with spatial autocorrelati
hist(sim_temp[[1]]$x, xlab="Temperature (°C)", main="Histogram of simulated temperatures", col="red")
```



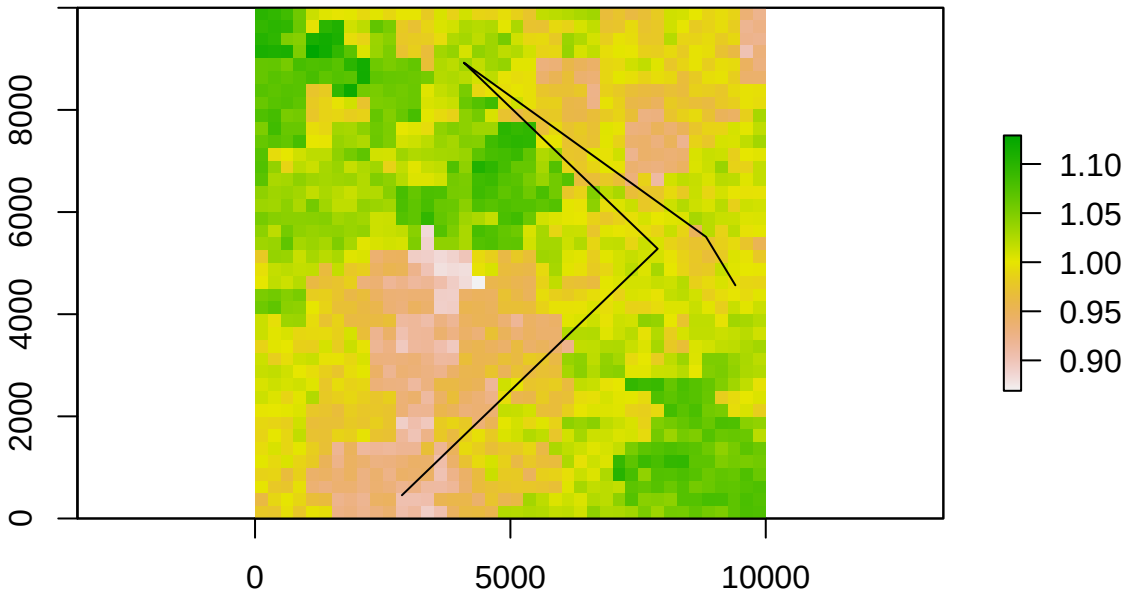



```
par(mfrow=c(1,1))

# Format temperature data
names(mat) <- paste0("d_", 1:ndays)
df_temp <- cbind(df, mat)
```

Simulate an arbitrary road segment for medium-range dispersal

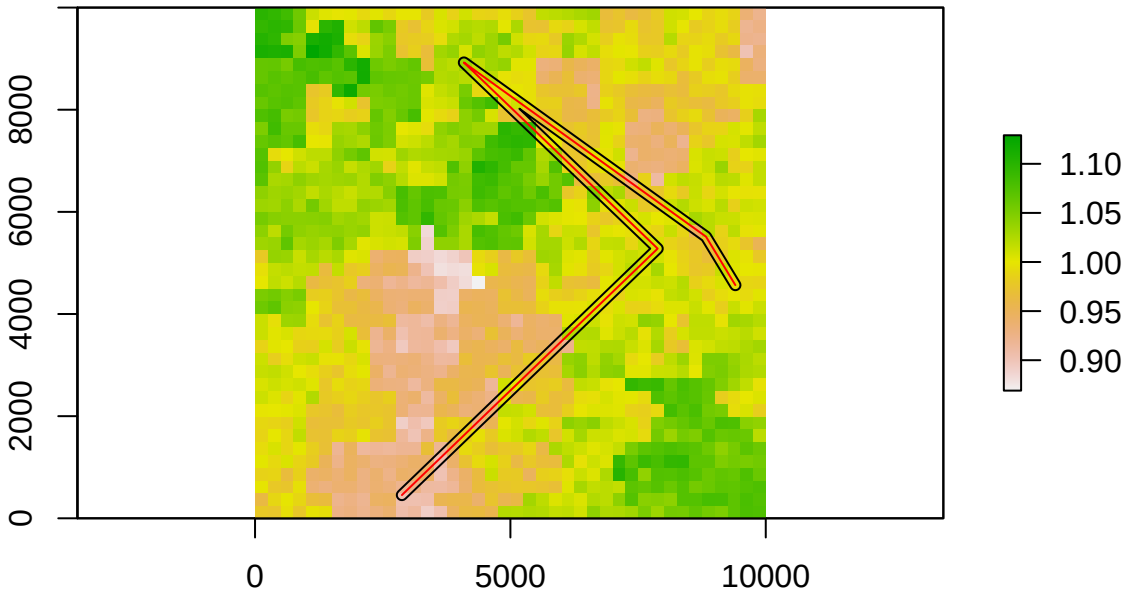
In the model we have considered the possiblity of medium-range passive dispersal. Thus, we will simulate an arbitrary road segment along which adult mosquitoes can disperse passively (i.e., thorugh car traffic).

```
set.seed(123)
pts <- spsample(bbox, 5, type="random")
roads <- spLines(pts)

# Check simulated segment
raster::plot(r)
raster::plot(roads, add=T)
```



After defining the road segment we will add a "buffer" of 100 m around the road segment. Adult mosquitoes that reach or develop into cells comprised in the 100 m buffer around roads will be able to undergo passive dispersal.

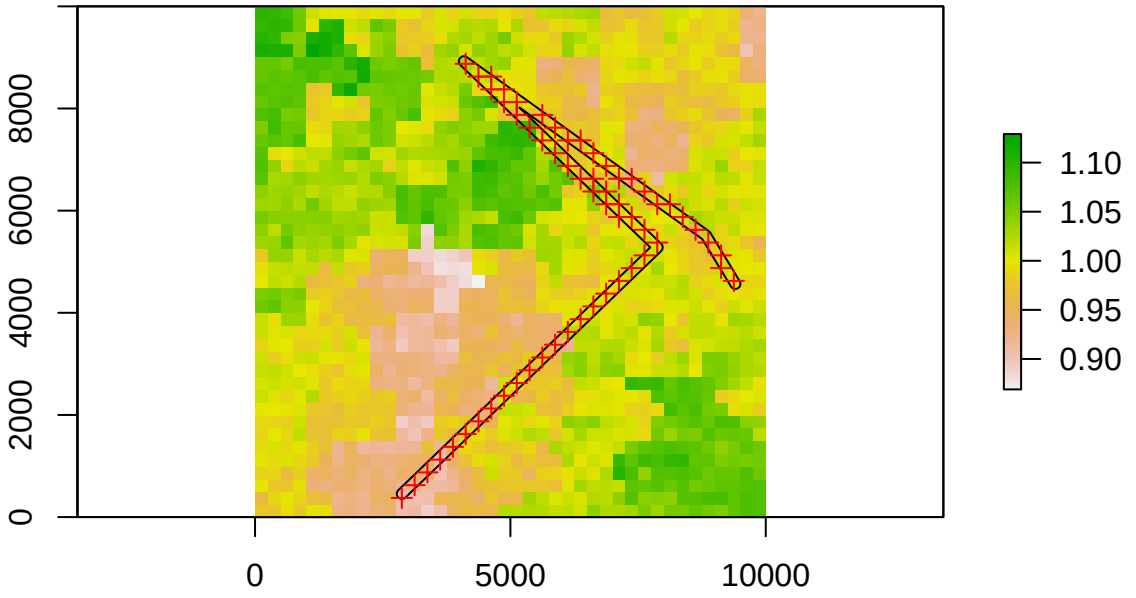
```
buff <- buffer(roads, width=100)
crs(buff) <- crs(r)
# Check grid, road segment and buffer
raster::plot(r)
raster::plot(buff, add=T)
raster::plot(roads, add=T, col="red")
```



Next, we derive a distance matrix between cells comprised in the spatial buffer along the road network. First, we select the cells.

```
df_sp=df
coordinates(df_sp)=~x+y
df_sp=raster::intersect(df_sp,buff)
```

```
# Check selected cells
raster::plot(r)
raster::plot(buff, add=T)
raster::plot(df_sp, add=T, col="red")
```



Then, we compute the Euclidean distance between each selected cell.

```
dist_matrix <- as.matrix(dist(coordinates(df_sp)))
```

Format the simulated input datasets and run the model

Model settings

Float numbers in the temperature matrix would slow the computational speed, thus we first multiply them by 1000 and then transform them in integer numbers.

```
w <- sapply(df_temp[,-c(1:3)], function(x) as.integer(x*1000))
```

We can now define a two-column matrix of coordinates to identify each cell in the lattice grid.

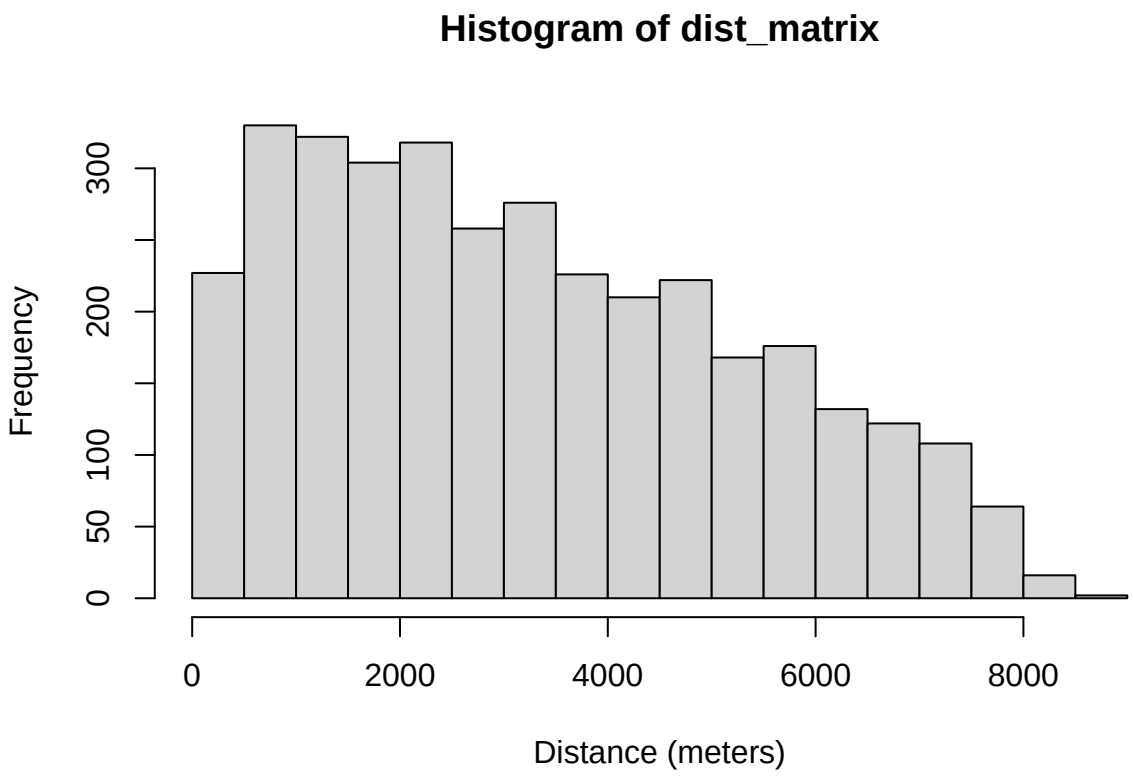
```
cc <- df_temp[,c("x","y")]
```

As for model requirement, the distance matrix must have column names equal to row names.

```
colnames(dist_matrix) <- row.names(dist_matrix)
```

Moreover, distances in the distance matrix must be rounded to the thousands.

```
dist_matrix <- apply(dist_matrix,2,function(x) round(x/1000,1)*1000)

# An histogram showing the distribution of distances of cells along the road network
hist(dist_matrix, xlab="Distance (meters)")
```



We can now define the model parameters.

```
## Define cells into which introduce propagules on day 1
intro.vector <- sample(as.numeric(row.names(dist_matrix)),1)
## Define the day of introduction (day 1)
str = 121
## Define the end-day of life cycle
endr = 121+(365*2)
## Define the number of eggs to be introduced
ie = 500
## Define the number of model iterations
it = 10 # The higher the number of simulation the better
## Define the number of liters for the larval density-dependent mortality
habitat_liters=1
#Define latitude and longitude for the diapause process
myLat=42
myLon=7
##Define average trip distance
myCountry="fra"
## Define the number of parallel processes (for sequential itarations set nc=1)
cl = 5
## Set output name for the *.RDS output will be saved
outname= paste0("dynamAedes_albo_lc_dayintro_",str,"_end",endr,"_niters",it,"_neggs",ie)
```

Run the model

Running the model takes around 15 minutes with the settings specified in this example.

```
simout=dynamAedes.m(species="albopictus",
            scale="lc",
```

```
            ihwv=habitat_liters,
            temps.matrix=w,
            cells.coords=cc,
            road.dist.matrix=dist_matrix,
            intro.cells=intro.vector,
            startd=str,
            endd=endr,
            n.clusters=cl,
            iter=it,
            intro.eggs=ie,
            compressed.output=TRUE,
            lat=myLat,
            long=myLon,
            country=myCountry,
            suffix=outname,
            verbose = FALSE)
```

Analyze the results

We first explore the model output structure: the *simout* object is a nested list.

The **first** level corresponds to the number of model iterations

```
print(it)
```

```
## [1] 10
```

```
print(length(simout))
```

```
## [1] 10
```

The **second** level corresponds to the simulated days. So if we inspect the first iteration, we will observe that the model has computed 366 days, as we had specified above in the object *endr*.

```
length(simout[[1]])
```

```
## [1] 731
```

The **third** level corresponds to the amount of individuals for each stage (rows) within each grid cell of the landscape (columns). So if we inspect the first day within the first iteration, we will obtain a matrix having

```
dim(simout[[1]][[1]])
```

```
## [1]    4 1600
```

We can now use the auxiliary functions of the model to analyze the results.

Derive probability of a successfull introduction at the end of the simulated period

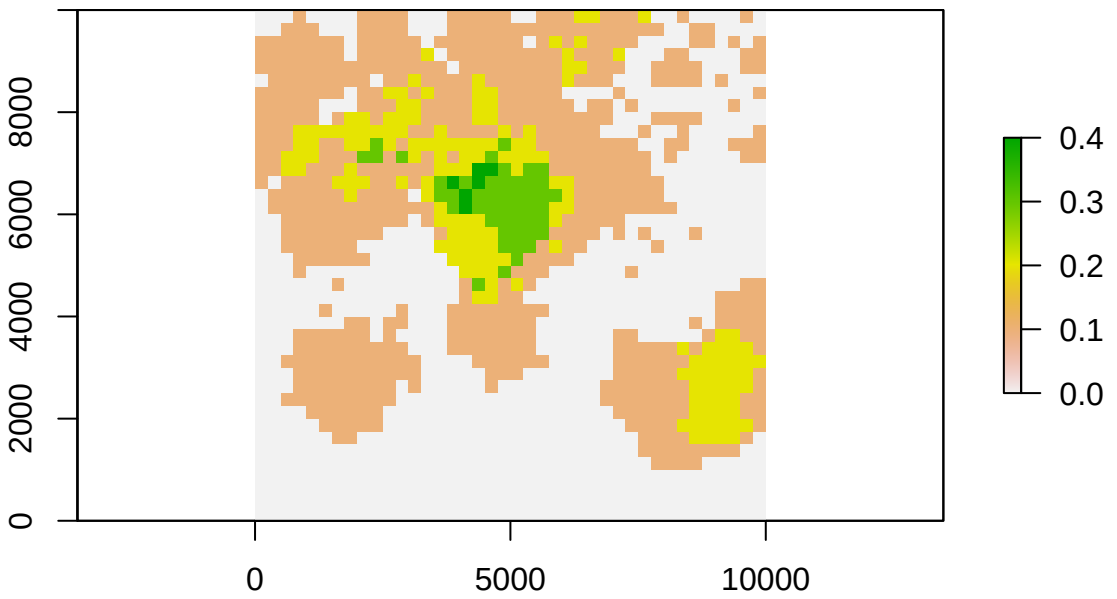
First, we can retrieve the probability of successful introduction, computed as the proportion of model iterations that resulted in a viable mosquito population at a given date.

```
rbind.data.frame(psi(input_sim = simout, eval_date = 300),
                 psi(input_sim = simout, eval_date = 700))
```

```
##   Days_after_intro p_success      stage
## 1          Day 300         1 Population
## 2          Day 700         1 Population
```

We can also get a spatial output, using the function *psi_sp*, which require as additional input only the matrix of the pixels coordinates,

```
plot(psi_sp(coords = cc, input_sim = simout, eval_date = 600, n.clusters=cl))
```



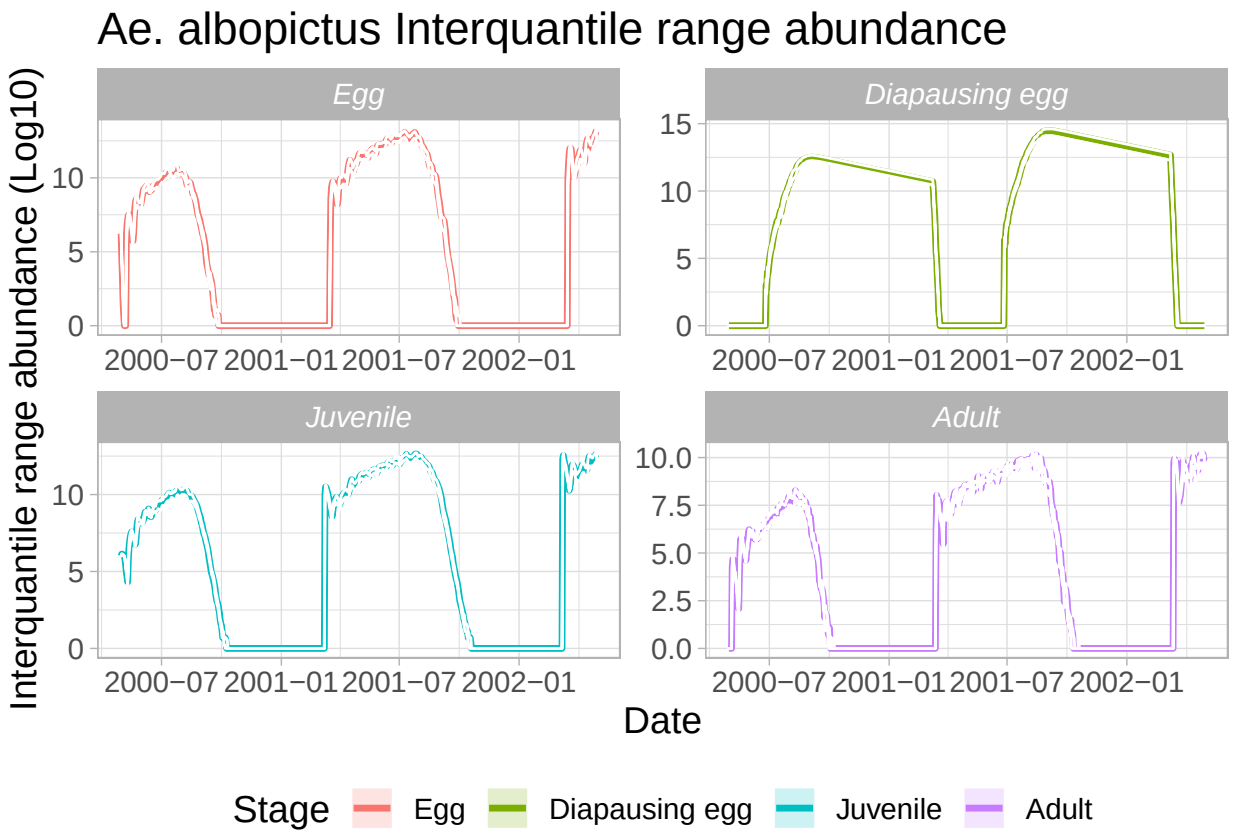
At local scale, this output has a double interpretation: a pixel having psi=0 can be a pixel where all the simulations resulted in an extinction or where the species has not arrived yet through dispersal.

Derive abundance 95% CI for each life stage and in each day

We can now compute the interquantile range abundance of the simulated population over the whole landscape using the function *adci* over the whole landscape.

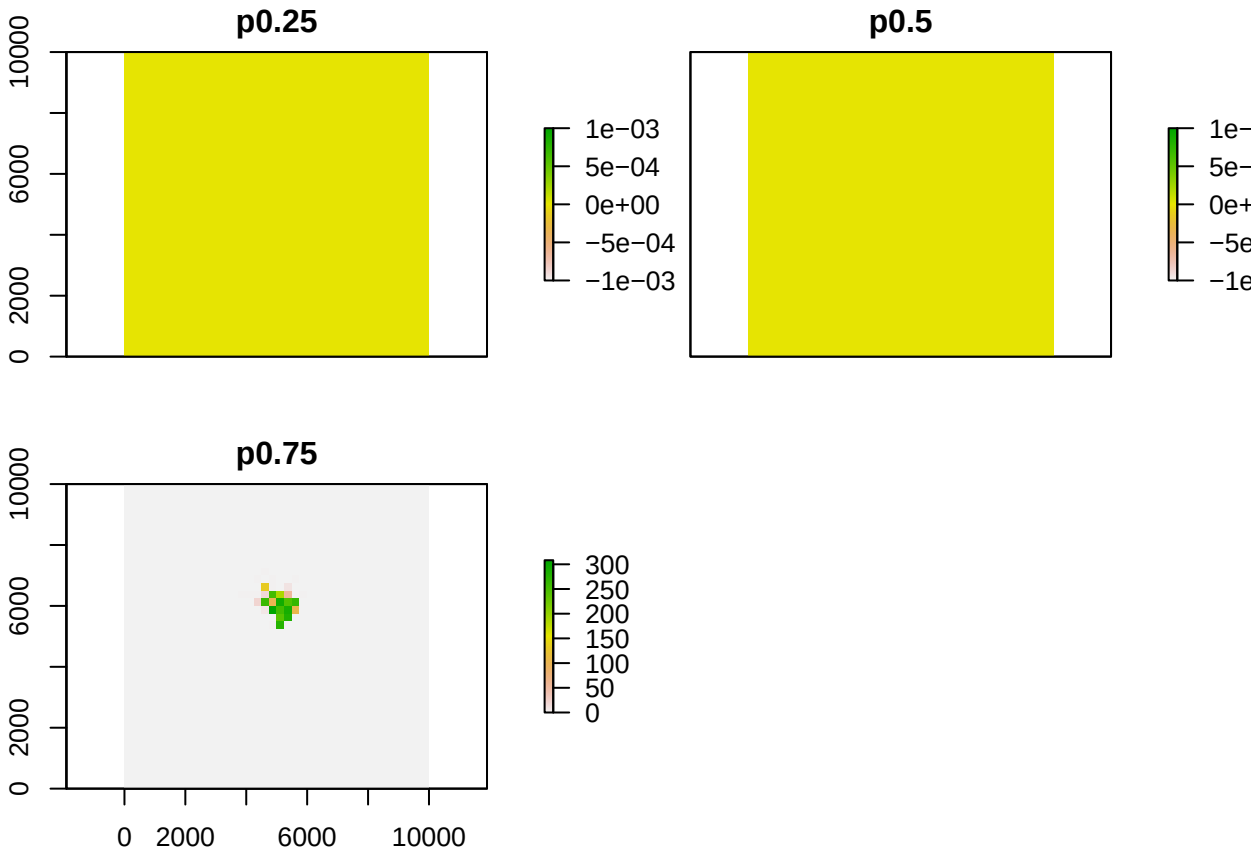
```
dd <- max(sapply(simout, function(x) length(x)))#retrieve the maximum number of simulated days
egg<-as.data.frame(adci(simout, eval_date=1:dd, breaks=c(0.25,0.50,0.75), stage=1))
juv<-as.data.frame(adci(simout, eval_date=1:dd, breaks=c(0.25,0.50,0.75), stage=2))
ad<-as.data.frame(adci(simout, eval_date=1:dd, breaks=c(0.25,0.50,0.75), stage=3))
eggd<-as.data.frame(adci(simout, eval_date=1:dd, breaks=c(0.25,0.50,0.75), stage=4))

egg$myStage="Egg"
egg$Date=seq.Date(sim_temp[[1]]$date[str], sim_temp[[1]]$date[endr], by="day")
juv$myStage="Juvenile"
juv$Date=seq.Date(sim_temp[[1]]$date[str], sim_temp[[1]]$date[endr], by="day")
ad$myStage="Adult"
ad$Date=seq.Date(sim_temp[[1]]$date[str], sim_temp[[1]]$date[endr], by="day")
eggd$myStage="Diapausing egg"
eggd$Date=seq.Date(sim_temp[[1]]$date[str], sim_temp[[1]]$date[endr], by="day")

outdf=bind_rows(egg, juv, ad, eggd) %>%
  as_tibble()

outdf %>%
  mutate(myStage=factor(myStage, levels= c("Egg", "Diapausing egg", "Juvenile", "Adult"))) %>%
  ggplot( aes(y=log1p(`50%`),x=Date, group=factor(myStage),col=factor(myStage))) +
  ggtitle("Ae. albopictus Interquantile range abundance")+
  geom_line(size=1.2)+
```

```
  geom_ribbon(aes(ymin=log1p(`25%`),ymax=log1p(`75%`),fill=factor(myStage)),
              col="white",
              alpha=0.2,
              outline.type="full")+
  labs(x="Date", y="Interquantile range abundance (Log10)", col="Stage", fill="Stage")+
  facet_wrap(~myStage, scales = "free")+
  theme_light()+
  theme(legend.pos="bottom",  text = element_text(size=14) , strip.text = element_text(face = "italic"))
```



We can also have a spatial output of the estimated interquantile range abundance of a given life stage using the function *adci_sp* just specifiying the pixels coordinates.

```
adult_sp_ab=adci_sp(simout, coords=cc, eval_date=450, breaks=c(0.25,0.50,0.75), stage=3)
plot(adult_sp_ab)
```



Note that if only a small number of mosquitoes are present in the pixels, the lower quantiles (e.g., the 1st and the 2nd quartiles) will be zero.

Number of invaded cells

Compute a summary of the number of invaded cells over model iterations

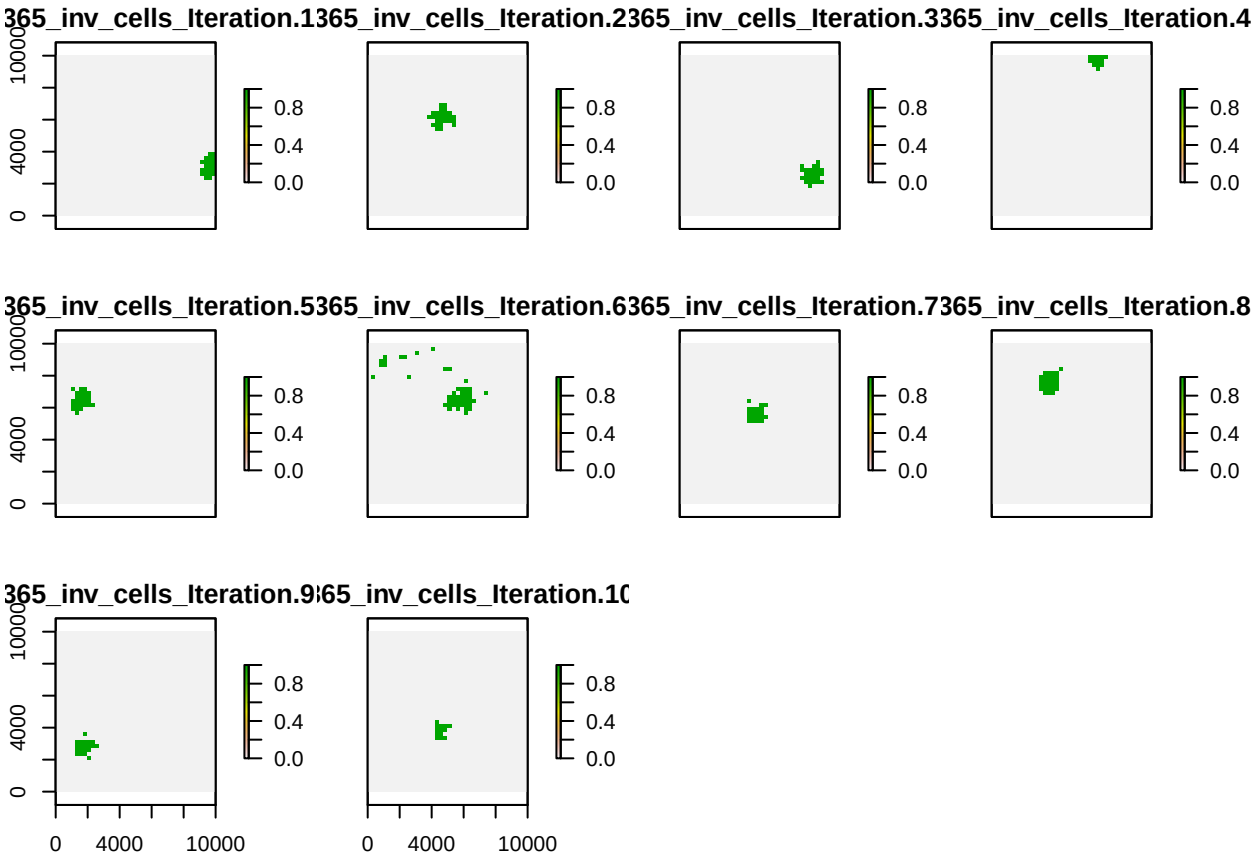
```
x=icci(simout, eval_date=700, breaks=c(0.25,0.50,0.75))
x
```

```
##    25% 50% 75% day
## V1   1   1   1   1
```

Estimates of mosquito dispersal spread (in km2)

Estimates of mosquito dispersal spread (in km2) of the simulated mosquito populations when scale = “lc”

```
x=dici(simout, coords=cc, eval_date=365, breaks=c(0.25,0.50,0.75), space=TRUE)
plot(x)
```



Regional scale model

Essentialy, the regional scale model run a punctual model within each grid cell of the landscape, without accounting for both active and passive dispersal of the mosquitoes. With this setting, the model requires two input datasets: a numerical matrix with temperatures in Celsius defined in space and time (space in the rows, time in the columns), and a two-column numerical matrix reporting the coordinates (in meters) of each space-unit (cell). For the purpose of this tutorial, we will use the following simulated datasets:

1. A 10 km lattice grid with 250 m cell size;
2. A 2-year long spatially and temporally correlated temperature time series;

Model settings

We take advantage of the spatial temperature dataset simulated for the local scale modelling exercise.

Float numbers in the temperature matrix would slow the computational speed, thus we first multiply them by 1000 and then transform them in integer numbers.

```
w <- sapply(df_temp[,-c(1:3)], function(x) as.integer(x*1000))
```

We can now define a two-column matrix of coordinates to identify each cell in the lattice grid.

```
cc <- df_temp[,c("x","y")]
```

We can now define the model parameters.

```
## Define the day of introduction (day 1)
str = 121
## Define the end-day of life cycle
```

```
endr = 121+365
## Define the number of eggs to be introduced
ie = 500
## Define the number of model iterations
it = 10 # The higher the number of simulation the better
## Define the number of liters for the larval density-dependent mortality
habitat_liters=1
#Define latitude and longitude for the diapause process
myLat=42
myLon=7
## Define the number of parallel processes (for sequential iterations set nc=1)
cl = 5
## Set output name for the *.RDS output will be saved
outname= paste0("dynamAedes_albo_rg_dayintro_",str,"_end",endr,"_niters",it,"_neggs",ie)
```

Run the model

Running the model takes around 10 minutes with the settings specified in this example.

```
simout=dynamAedes.m(species="albopictus",
            scale="rg",
            ihwv=habitat_liters,
            temps.matrix=w,
            cells.coords=cc,
            startd=str,
            endd=endr,
            n.clusters=cl,
            iter=it,
            intro.eggs=ie,
            compressed.output=TRUE,
            lat=myLat,
            long=myLon,
            suffix=outname,
            verbose = FALSE)
```

Analyze the results

We first explore the model output structure: the *simout* object is a nested list.

The **first** level corresponds to the number of model iterations

```
print(it)
```

```
## [1] 10
```

```
print(length(simout))
```

```
## [1] 10
```

The **second** level corresponds to the simulated days. So if we inspect the first iteration, we will observe that the model has computed 366 days, as we had specified above in the object *endr*.

```
length(simout[[1]])
```

```
## [1] 366
```

The **third** level corresponds to the amount of individuals for each stage (rows) within each grid cell of the landscape (columns). So if we inspect the first day within the first iteration, we will obtain a matrix having

```
dim(simout[[1]][[1]])
```

```
## [1]    4 1600
```

We can now use the auxiliary functions of the model to analyze the results.

Derive probability of a successfull introduction at the end of the simulated period

First, we can retrieve the probability of successful introduction, computed as the proportion of model iterations that resulted in a viable mosquito population at a given date for a given life stage.

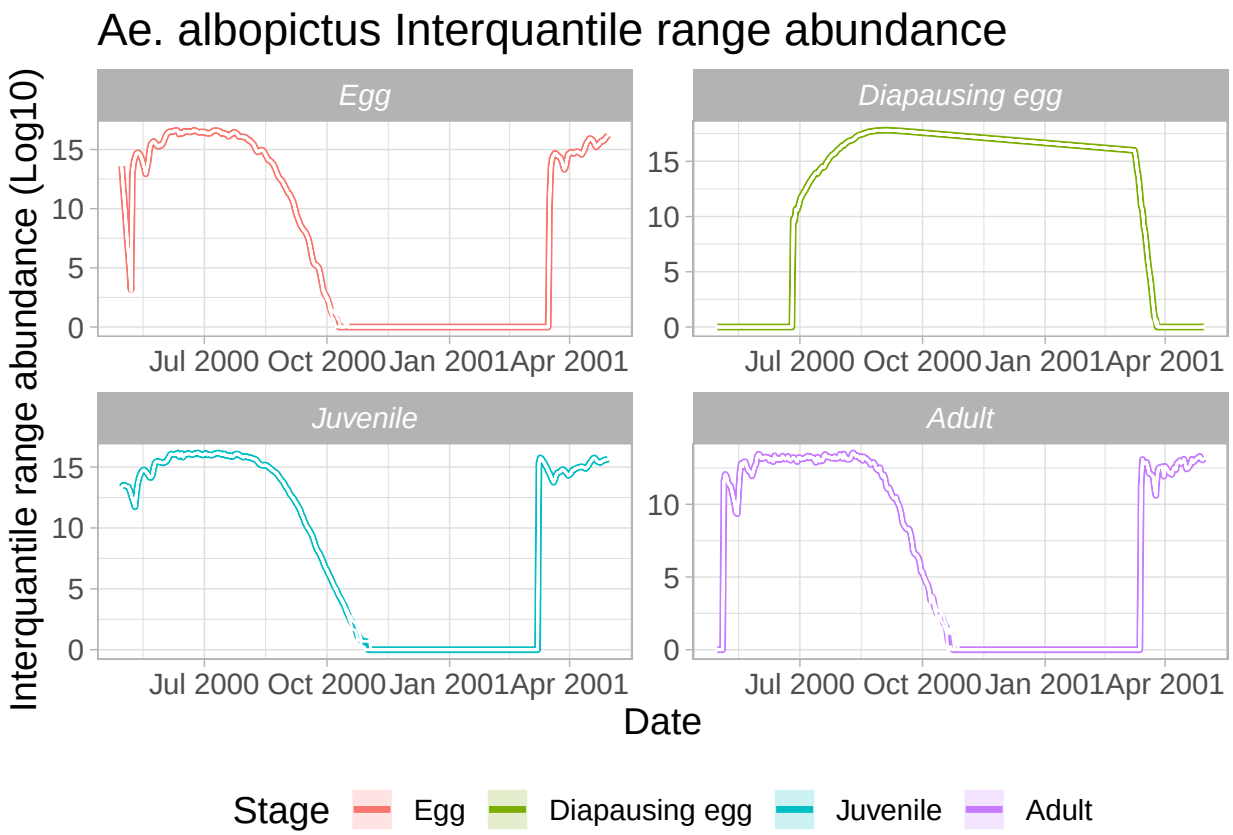
```
rbind.data.frame(psi(input_sim = simout, eval_date = 365))
```

```
##   Days_after_intro p_success      stage
## 1          Day 365         1 Population
```

Derive abundance 95% CI for each life stage and in each day

We can now compute the interquantile range abundance of the simulated population using the function *adci* over the whole landscape.

```
dd <- max(sapply(simout, function(x) length(x)))#retrieve the maximum number of simulated days
egg<-as.data.frame(adci(simout, eval_date=1:dd, breaks=c(0.25,0.50,0.75), st=1))
juv<-as.data.frame(adci(simout, eval_date=1:dd, breaks=c(0.25,0.50,0.75), st=2))
ad<-as.data.frame(adci(simout, eval_date=1:dd, breaks=c(0.25,0.50,0.75), st=3))
eggd<-as.data.frame(adci(simout, eval_date=1:dd, breaks=c(0.25,0.50,0.75), st=4))

egg$myStage="Egg"
egg$Date=seq.Date(sim_temp[[1]]$date[str], sim_temp[[1]]$date[endr], by="day")
juv$myStage="Juvenile"
juv$Date=seq.Date(sim_temp[[1]]$date[str], sim_temp[[1]]$date[endr], by="day")
ad$myStage="Adult"
ad$Date=seq.Date(sim_temp[[1]]$date[str], sim_temp[[1]]$date[endr], by="day")
eggd$myStage="Diapausing egg"
eggd$Date=seq.Date(sim_temp[[1]]$date[str], sim_temp[[1]]$date[endr], by="day")

outdf=bind_rows(egg, juv, ad, eggd) %>%
  as_tibble()

outdf %>%
  mutate(myStage=factor(myStage, levels= c("Egg", "Diapausing egg", "Juvenile", "Adult"))) %>%
  ggplot( aes(y=log1p(`50%`),x=Date, group=factor(myStage),col=factor(myStage))) +
  ggtitle("Ae. albopictus Interquantile range abundance")+
  geom_line(size=1.2)+
  geom_ribbon(aes(ymin=log1p(`25%`),ymax=log1p(`75%`),fill=factor(myStage)),
              col="white",
              alpha=0.2,
              outline.type="full")+
  labs(x="Date", y="Interquantile range abundance (Log10)", col="Stage", fill="Stage")+
  facet_wrap(~myStage, scales = "free")+
  theme_light()+
  theme(legend.pos="bottom",  text = element_text(size=14) , strip.text = element_text(face = "italic"))
```


Ae. albopictus Interquantile range abundance
Egg
Diapausing egg
Juvenile
Adult
Interquantile range abundance (Log10)
0
5
10
15
Jul 2000
Oct 2000
Jan 2001
Apr 2001
Date
Stage
Egg
Diapausing egg
Juvenile
Adult