Climate variation simulation

2022-11-23

Testing mu simulation

This document show how to simulate a species with mu growth pre-computation and not integrate a classical IPM. For now, the package is in experimental state and use the branch *fast_int*. For this example we will use the species *Picea abies* as an example.

```
# Libraries
library(ggplot2)
library(dplyr)
```

```
##
## Attachement du package : 'dplyr'
```

```
## Les objets suivants sont masqués depuis 'package:stats':
##
##     filter, lag
```

```
## L'objet suivant est masqué depuis 'package:testthat':
##
##     matches
```

```
## Les objets suivants sont masqués depuis 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(devtools)

# Loading all functions of the package
devtools::load_all()
```

```
## i Loading treeforce
```

```
species <- "Picea_abies"
data(list = paste0("fit_", species))
fit <- eval(parse(text=paste0("fit_", species)))
climate <- subset(climate_species, sp == species, select = -sp)
```

Get Mu values

The first step consist in computing all mu values for a species in its climatic and basal area range. For this, we compute the mu range of the species and simulate a matrix for each values of mu, with a small step. Keep in mind that the time of this computation rely mostly on the step and this step will influence the precision of the simulation later. The mu range step is done inside the `make_mu_gr` function call, but it's possible to check it before.

```
mu_range <- getRangemu(
    climate = climate, fit = fit, BA = seq(0, 200, by = 10),
    mesh = seq(90, get_maxdbh(fit_Picea_abies) * 1.1, by = 2))
mu_range
```

```
##        min        max        sig
## -4.4163638  2.5839773  0.5813021
```

Note : Low values highly depend on maximum basal area, so, one can limit the mu computation time if he expect the basal area to stay low during simulation.

Making the mu matrix takes more or less the same arguments as the function `make_IPM` in term of integration levels for Gauss-Legendre and midbin methods. The important argument that limit computation time is stepMu.

```
mu_Picea_abies <- make_mu_gr(
    species = "Picea_abies", fit = fit_Picea_abies,
    mesh = c(m = 700, L = 90, U = get_maxdbh(fit_Picea_abies) * 1.1),
    verbose = TRUE, stepMu = 0.01)
```

```
## Mu range done

## Launching mu computation loop

## GL integration occur on 32 cells

## midbin integration occur on 25 cells

## Loop done.

## Time difference of 1.81 secs
```

This object is a matrix with as many row as integrated cells for a given mu, and a range of mu in column. During simulation, the loop will compute the mu needed for climate and basal area for a given time and extract from the matrix the required columns.

This step is 5 times longer than extracting the IPM matrix, this is the bottleneck during simulation.

```
step_climate <- subset(climate, N ==2, select = -N)
step_climate <- drop(as.matrix(step_climate))
Picea_IPM_BA20 <- get_step_IPM(x = mu_Picea_abies, BA = 20,
                               climate = step_climate, sim_corr = "cut")
Picea_IPM_BA20[1:5, 1:5]
```

```
## 5 x 5 sparse Matrix of class "dtCMatrix"
##
## [1,] 0.14293258 .          .         .         .
## [2,] 0.54639766 0.13991750 .         .         .
## [3,] 0.22480400 0.54383487 0.1339809 .         .
## [4,] 0.05652551 0.22800705 0.5383695 0.1282001 .
## [5,] 0.01530974 0.05811602 0.2343373 0.5325677 0.1253771
```

```
dim(Picea_IPM_BA20)
```

```
## [1] 700 700
```

Simulations

Once the mu matrix is created, creating species and simulating is really close to simulating with IPM. Because the mu matrix are not defined for a specific climate, we need to set one to simulate on it. For now we only set a single climate that will not change during simulation.

```
time <- 1000

Picea_abies <- new_species(IPM = mu_Picea_abies, init_pop = def_initBA(20),
```

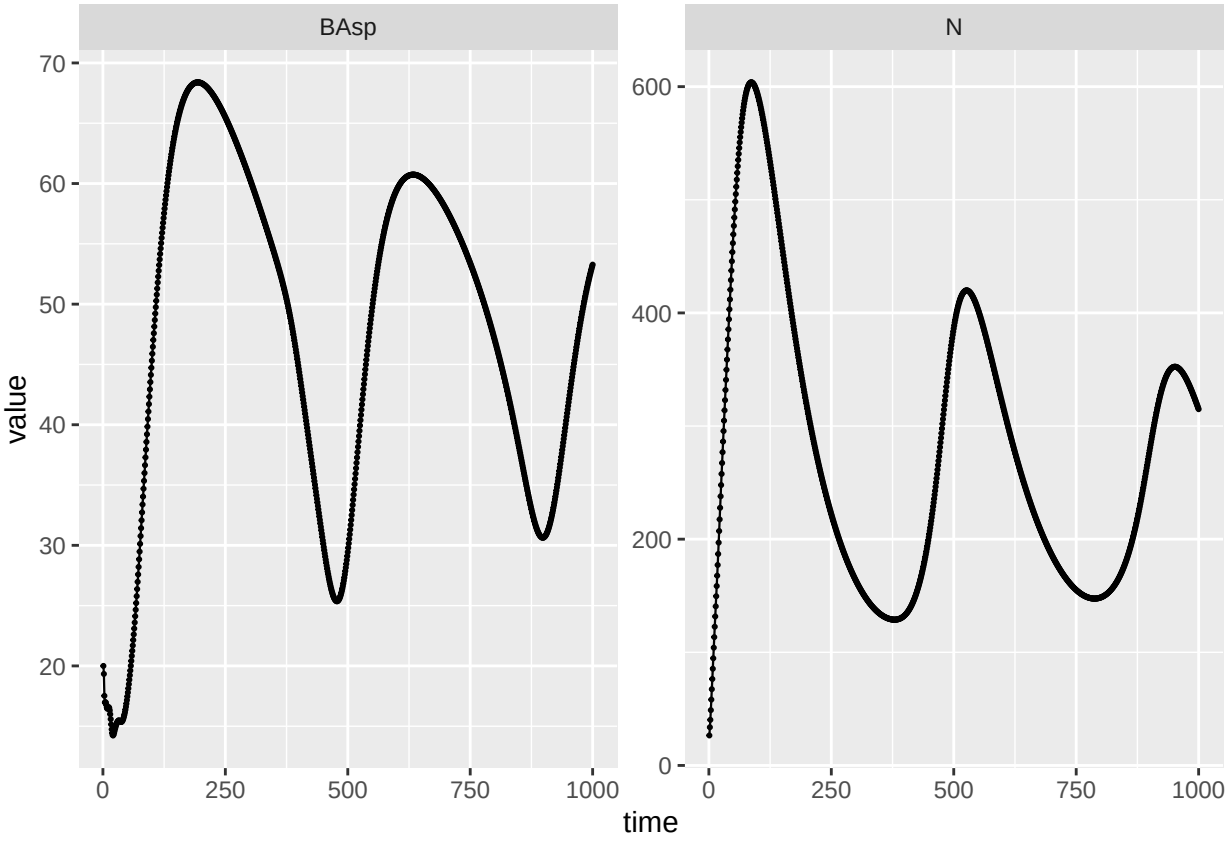
```
                              harvest_fun = def_harv)

forest <- new_forest(species = list(mu_Picea = Picea_abies))
load_all()
```

```
## i Loading treeforce
```

```
set.seed(42)
memor_mu <- sim_deter_forest.forest(forest, tlim = time, climate = step_climate,
                                    equil_dist = time, equil_time = time,
                                    verbose = TRUE, correction = "cut") %>%
    tree_format()
```

```
## apply a IPM cut correction
## Starting while loop. Maximum t = 1000
## time 500 | BA diff : 54.16
## time 1000 | BA diff : 54.16
## Simulation ended after time 1000
## BA stabilized at 53.27 with diff of 54.16 at time 1000
## Time difference of 13.8 secs
```

```
memor_mu %>%
    filter(var %in% c("BAsp", "H", "N"), ! equil, value != 0) %>%
    ggplot(aes(x = time, y = value)) +
    facet_wrap(~ var, scales = "free_y") +
    geom_line(size = .4) +
    geom_point(size = .4) +
    NULL
```



```
ipm_Picea <- make_IPM(
    "Picea_abies", step_climate, "opt_Picab_clim", fit = fit_Picea_abies,
    mesh = c(m = 700, L = 90, U = get_maxdbh(fit_Picea_abies) * 1.1),
    BA = 0:100, verbose = TRUE
)
Picea_abis_ipm <- species(IPM = ipm_Picea, init_pop = def_initBA(20),
                          harvest_fun = def_harv)
forest <- new_forest(species = list(ipm_Picea = Picea_abies_ipm))
load_all()
set.seed(42)
memor_ipm <- sim_deter_forest.forest(forest, tlim = time,
                                     equil_dist = time, equil_time = time,
                                     verbose = TRUE, correction = "none") %>%
    tree_format()
```

```
e_memor <- dplyr::bind_rows(ipm = memor, mu = memor_mu, .id = "meth")
e_memor %>%
    filter(var %in% c("BAsp", "H", "N"), ! equil, value != 0) %>%
    # dplyr::na_if(0) %>%
    ggplot(aes(x = time, y = value, color = meth)) +
    facet_wrap(meth ~ var, scales = "free_y") +
    geom_line(size = .4, linetype = "dotted") +
    geom_point(size = .4) +
    NULL
```