# 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 = 0.001)
```

```
## Mu range done

## Launching mu computation loop

## GL integration occur on 32 cells

## midbin integration occur on 25 cells

## Integration ====>---------------------------   14% | ETA: 12sIntegration ====>------------------------
## Time difference of 12.9 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,] 0.14232207 .         .         .         .
## [2,] 0.54587574 0.1372175 .         .         .
## [3,] 0.22543914 0.5413825 0.1328084 .         .
## [4,] 0.05683980 0.2308491 0.5372208 0.1282001 .
## [5,] 0.01542374 0.0595642 0.2355892 0.5325677 0.1236919
```

```
dim(Picea_IPM_BA20)
```

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

## Simulations

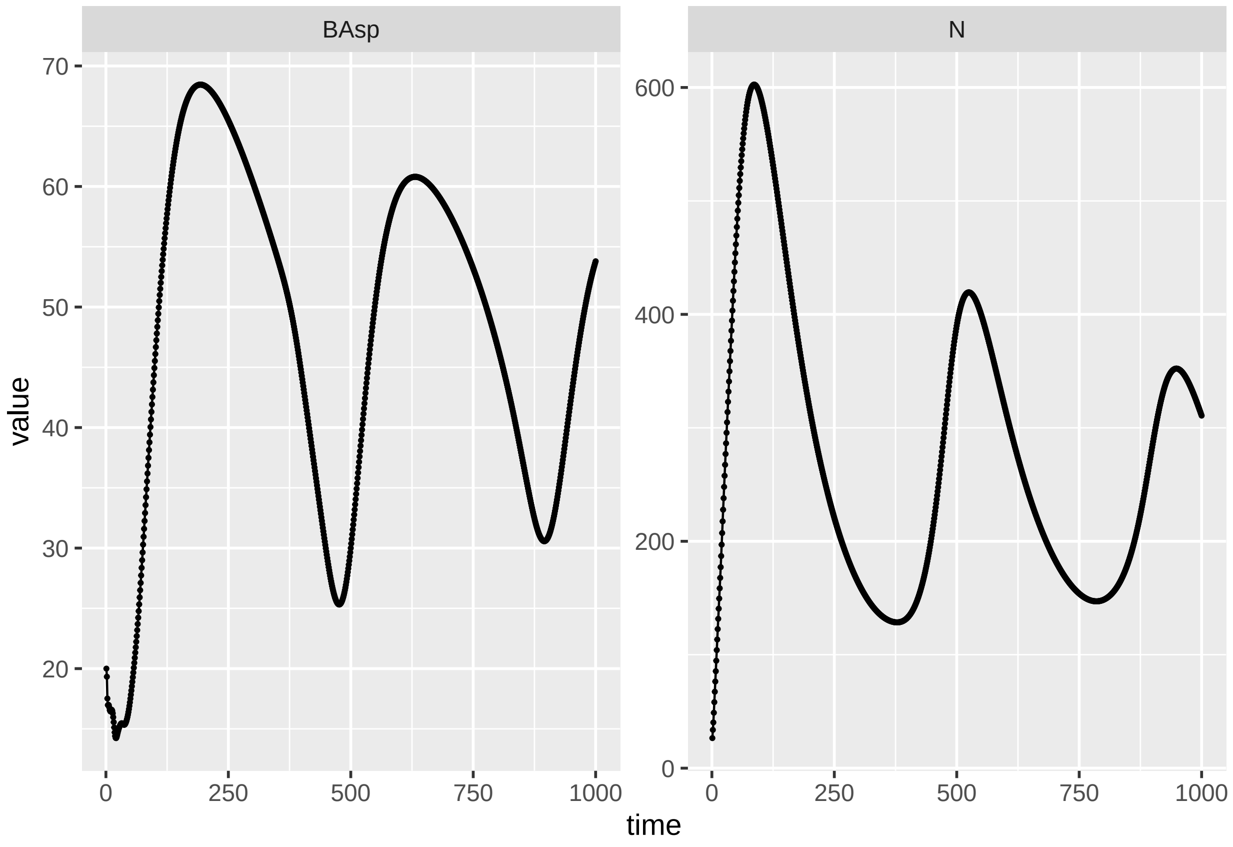
Once the mu matrix is created, creating species and simulating is really close to simulating with IPM. Because the mu matrix are not defined for a specific climate, we need to set one to simulate on it. For now we only set a single climate that will not change during simulation.

```
time <- 1000

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

```r
forest <- new_forest(species = list(mu_Picea = Picea_abies))
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 : 54.22

## time 1000 | BA diff : 54.22

## Simulation ended after time 1000

## BA stabilized at 53.80 with diff of 54.22 at time 1000

## Time difference of 12 secs
```

```r
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
```



We can compare this with a classical IPM simulation to see that it's not so different. Obviously this takes some times since integration takes longer. To limit this, we only integrate for basal area up to 100.

```r
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
)
```

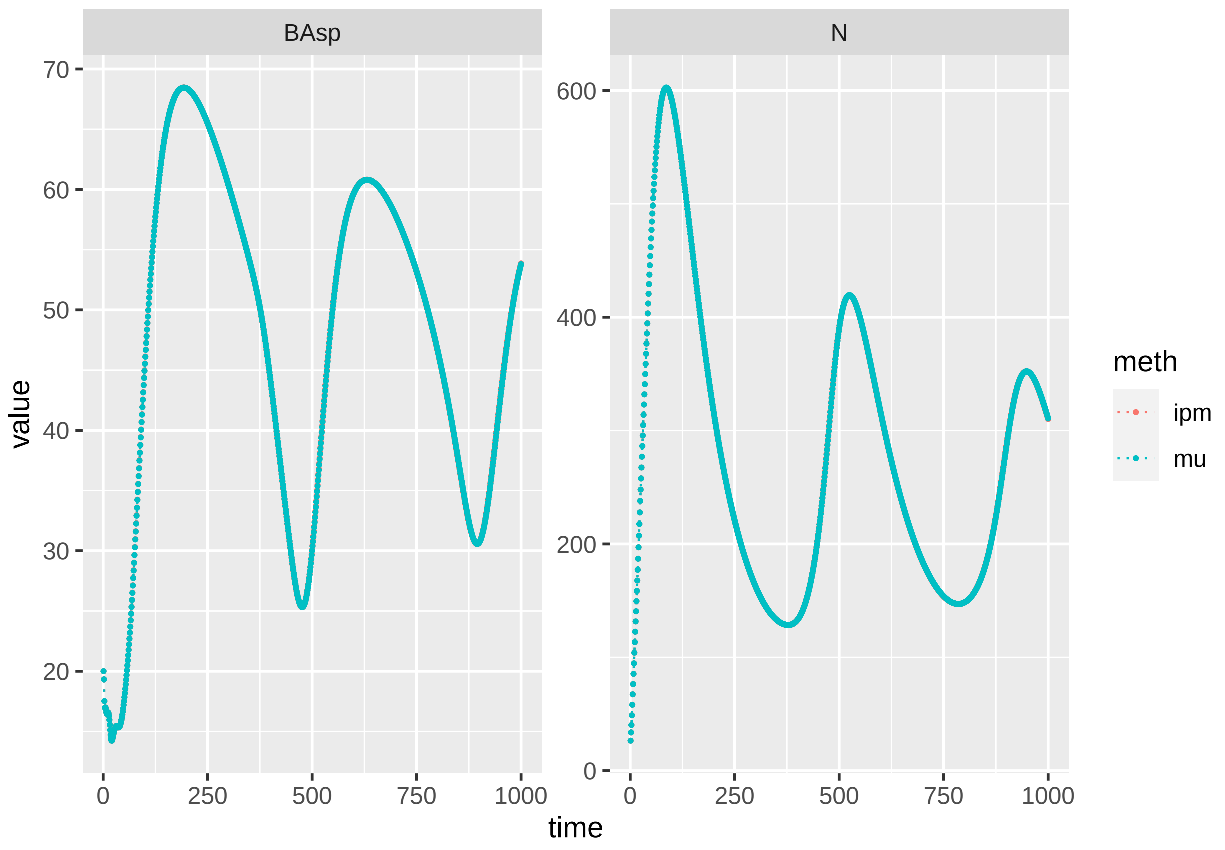
```
## Launching integration loop

## GL integration occur on 32 cells

## midbin integration occur on 25 cells

## Integration =>-----------------------------    3% | ETA:  1mIntegration =>---------------------------
## Time difference of 57.9 secs
```

```r
Picea_abies_ipm <- species(IPM = ipm_Picea, init_pop = def_initBA(20),
                          harvest_fun = def_harv)
forest_ipm <- new_forest(species = list(ipm_Picea = Picea_abies_ipm))
set.seed(42)
memor_ipm <- sim_deter_forest.forest(forest_ipm, tlim = time,
                                    equil_dist = time, equil_time = time,
                                    verbose = TRUE, correction = "cut") %>%
    tree_format()
```

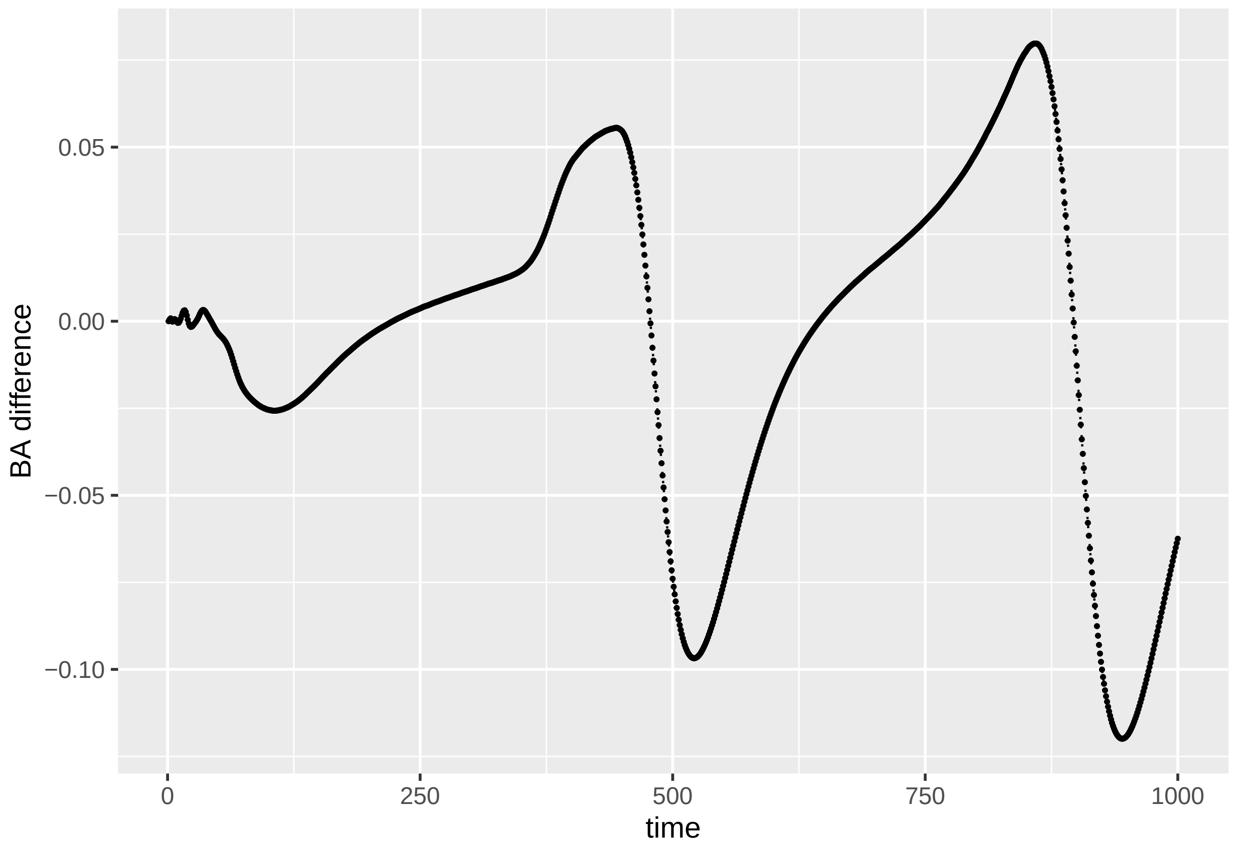
```
## apply a IPM cut correction
## Starting while loop. Maximum t = 1000
## time 500 | BA diff : 54.22
## time 1000 | BA diff : 54.22
## Simulation ended after time 1000
## BA stabilized at 53.86 with diff of 54.22 at time 1000
## Time difference of 3.67 secs
```

Below is a plot that combine both simulations and compute the difference along time.

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



```
e_memor %>%
    filter(var %in% c("BAsp"), ! equil) %>%
    group_by(time) %>% summarise(value = diff(value)) %>%
    ggplot(aes(x = time, y = value)) +
    ylab("BA difference") +
    geom_line(size = .4, linetype = "dotted") +
    geom_point(size = .4) +
    NULL
```



## Climate evolution in time

Previous simulation specified a single climate that is used from start to finish, with only a variation on BA. The interest of using precomputed values is too create new IPM matrix for a lot more climates. But as integrating an IPM takes around 2 or minutes, integrating a climate gradient would takes way too long, and only to use few basal area values when we integrate between 0 and 200.

Of course we now need to define a climate at each time step of the future simulation. For example, the simulation will start on cold edge and migrate to hot edge in a thousand years, with a small random effect.

**This code is an example and some functions may be added to the package later.**

```r
# future function for the package treeforce.
# I may need some help on what's needed or usefull

#' @param x a named values for a climatic variable.
#' @param inv_null If TRUE, the inverse variable is \code{varb = 1/(var+1)}, in
#' case of var can take 0 for value.
#' Example : \code{x = c(sgdd = 2000)}
value2sqb <- function(x, inv_null = FALSE){

    nms <- names(x)
    res <- numeric(3)
    names(res) <- c(nms, paste0(nms, c("2", "b")))
    res[nms] <- x
    res[paste0(nms, "2")] <- x^2
    res[paste0(nms, "b")] <- 1/(x + inv_null)
    res
}

#' @param climate Named vector of climatic variables.
#' @param inv_null If TRUE, the inverse variable is \code{varb = 1/(var+1)}, in
#' case of var can take 0 for value.
#' Example : \code{x = c(sgdd = 2000)}
#' Example : \code {climate = c(sgdd = 2000, wai = 0.16)}
expand_clim <- function(climate, inv_null){

    assertNumeric(climate, any.missing = FALSE)
    assertLogical(inv_null, any.missing = FALSE)

    nms <- names(climate)
    res <- vector("list", length(climate))
    for(x in seq_along(nms)){
        res[[x]] <- value2sqb(climate[nms[x]], inv_null[nms[x]])
    }
    res <- unlist(res)

    return(res)
}
# May be usefull to create the same function but to work on data.frame
# or matrices likes below code
```

```
n <- time
tmp <- subset(climate, N != 2, select = c(sgdd,wai, N))

wai <- seq(tmp$wai[2], tmp$wai[1], length.out = n) + runif(n, -0.05, 0.05)
wai <- pmax(wai, 0) ; wai <- pmin(wai, 1) # range the values after runif
sim_clim <- as.matrix(data.frame(
    sgdd = seq(tmp$sgdd[2], tmp$sgdd[1], length.out = n) + runif(n, -50, 50),
    wai = wai,
    t = 1:n
))
# We need to specify if variables can be set 0 for variable compution of
# varb = 1 / var. If true varb = 1 / (var + 1)
inv_null <- c(sgdd = FALSE, wai = TRUE)

exp_clim <- vector("list", n)
for(i in 1:n){
    exp_clim[[i]] <- c(expand_clim(sim_clim[i, -3], inv_null), t = i)
}
exp_clim <- do.call("rbind", exp_clim)
dim(exp_clim)
```

```
## [1] 1000    7
```

```
exp_clim[c(1:3, 997:1000),]
```

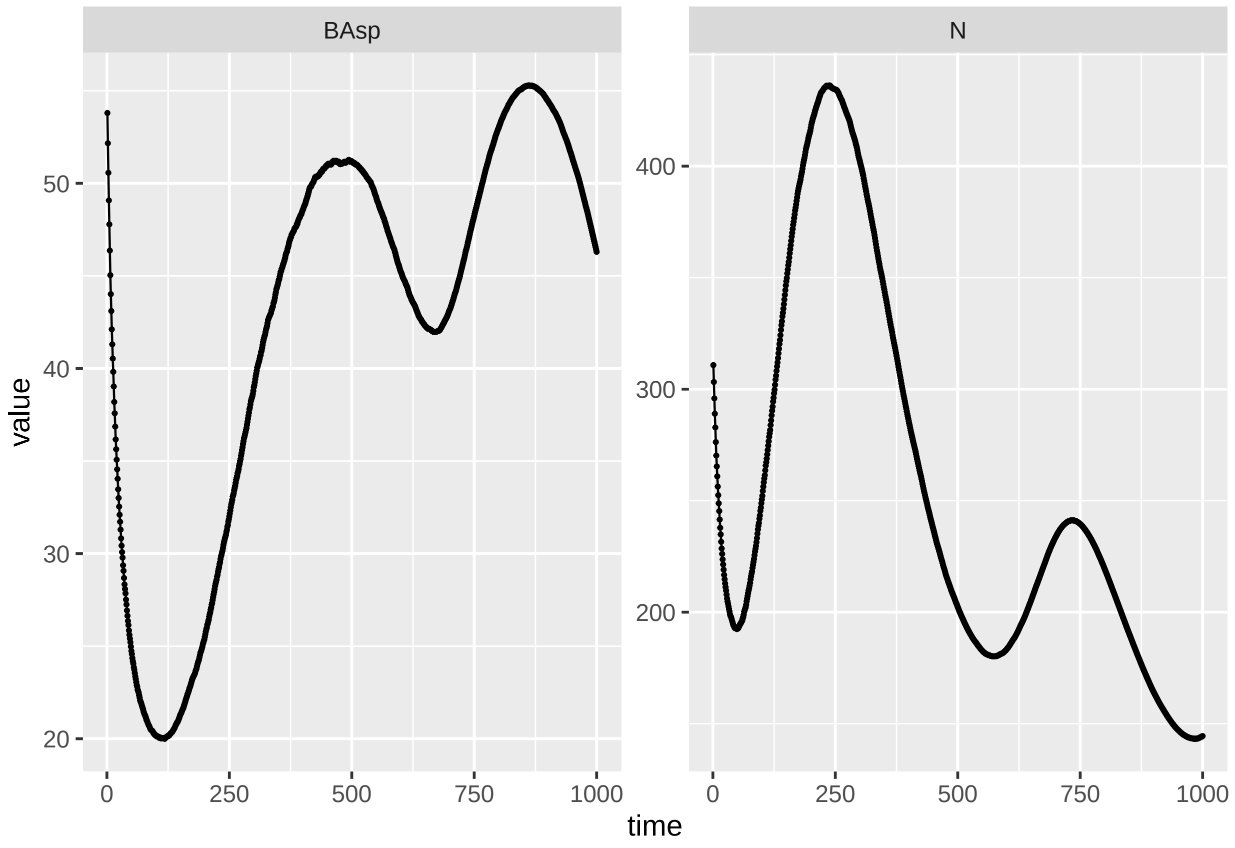
```
##            sgdd     sgdd2        sgddb        wai         wai2      waib    t
## [1,]   489.2361  239351.9 0.0020440030 0.97465608 0.9499544712 0.5064173    1
## [2,]   465.4413  216635.6 0.0021484987 1.00000000 1.0000000000 0.5000000    2
## [3,]   461.4521  212938.1 0.0021670720 1.00000000 1.0000000000 0.5000000    3
## [4,]  2163.6127 4681219.8 0.0004621899 0.06240819 0.0038947817 0.9412578  997
## [5,] 2230.3893 4974636.4 0.0004483522 0.01238347 0.0001533504 0.9877680  998
## [6,] 2167.8538 4699589.9 0.0004612857 0.00000000 0.0000000000 1.0000000  999
## [7,] 2204.7134 4860761.2 0.0004535737 0.00000000 0.0000000000 1.0000000 1000
```

Once the climate is set, we can simulate a species. I recreate the species to start with a size distribution closer to the equilibrium reached in previous simulations. Don't forget to modify this distribution with the surface sampled (`SurfEch`).

```
distrib <- filter(memor_mu, equil, var == "m") %>% pull(value) * 0.03
Picea_abies <- new_species(IPM = mu_Picea_abies, init_pop = def_init_k(distrib),
                           harvest_fun = def_harv)
clim_forest <- new_forest(species = list(clim_Picea = Picea_abies))
set.seed(42)
memor_clim <- sim_deter_forest.forest(clim_forest, tlim = time, climate = exp_clim,
                                      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 : 33.80

## time 1000 | BA diff : 35.29

## Simulation ended after time 1000

## BA stabilized at 46.30 with diff of 35.29 at time 1000

## Time difference of 9.32 secs
```

```
memor_clim %>%
    filter(var %in% c("BAsp", "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
```



I think this fast decay at start is because the distribution comes from an equilibrium for the "optimal" climate and not the the cold edge, but past this point an increase in *sgdd* and *wai* seems to increase basal area along with a decrease of population size. However, the climate variation is relatively quick and we may need to test more hypothesis. Lastly, the evolution of metrics is not smooth, so the random effect on climate may be effective.

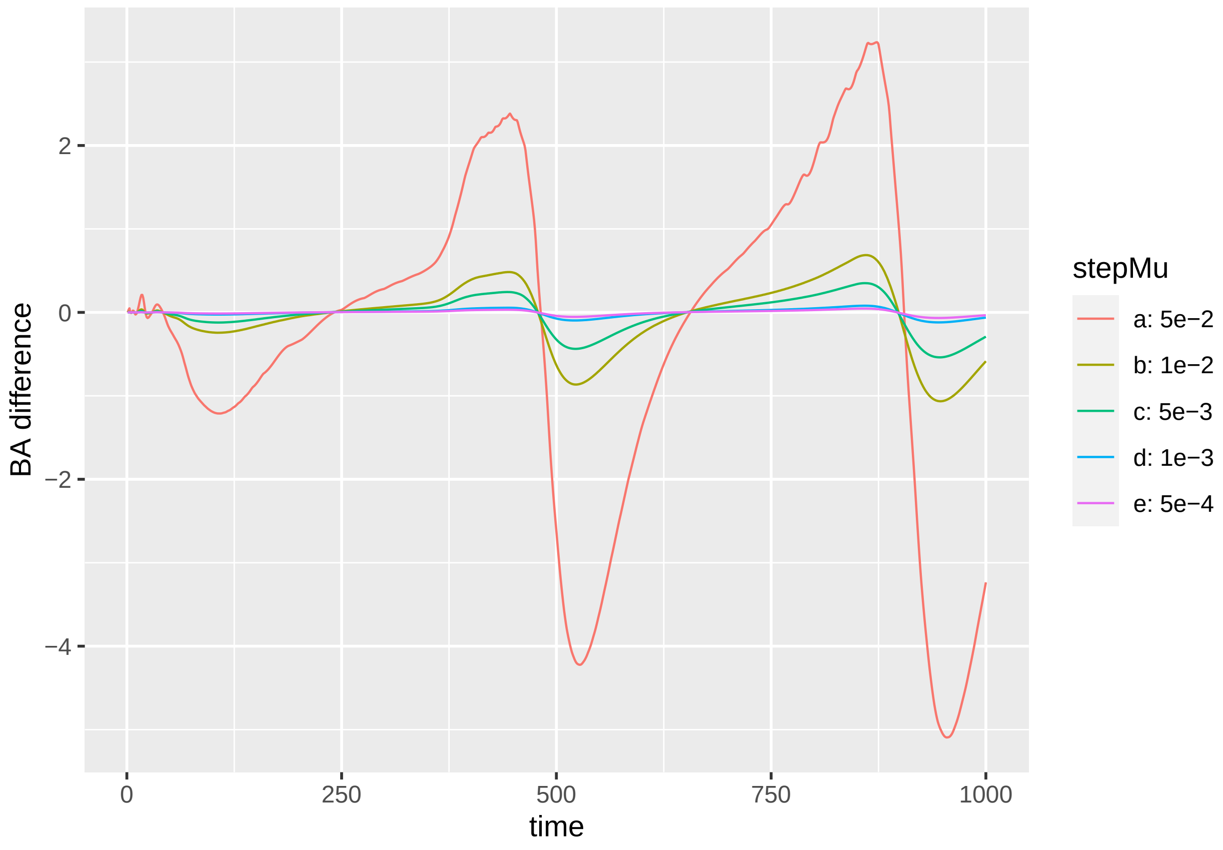
## Testing effect of stepMu

The BA difference is already pretty low between IPM and mu simulations, but this section will illustrate that it's directly linked with `stepMu` value. Error is negligeable for stepMu of 0.0001, but computing mu values takes around 2 minutes.

```r
steps <- c(0.05, 0.01, 0.005, 0.001, 0.0005)
diff_list <- vector("list", length(steps))

for(s in seq_along(steps)){
    cat(sprintf(" %.4f", steps[s]))
    mu_Picea <- make_mu_gr(
        species = "Picea_abies", fit = fit_Picea_abies,
        mesh = c(m = 700, L = 90, U = get_maxdbh(fit_Picea_abies) * 1.1),
        stepMu = steps[s])

    Picea <- new_species(IPM = mu_Picea, init_pop = def_initBA(20),
                         harvest_fun = def_harv)

    forest <- new_forest(species = list(Picea = Picea))
    set.seed(42)
    memor <- sim_deter_forest.forest(forest, tlim = time,
                                         climate = step_climate,
                                         equil_dist = time, equil_time = time,
                                         correction = "cut") %>%
        tree_format()

    diff_list[[s]] <- dplyr::bind_rows(ipm = memor_ipm, mu = memor,
                                        .id = "meth") %>%
    filter(var %in% c("BAsp"), ! equil) %>%
    group_by(time) %>% summarise(value = diff(value))
}
```

```
##  0.0500 0.0100 0.0050 0.0010 0.0005
```

```
names(diff_list) <- c("a: 5e-2", "b: 1e-2", "c: 5e-3", "d: 1e-3", "e: 5e-4")
res <- do.call("bind_rows", c(diff_list, list(.id = "stepMu")))
res %>%
    ggplot(aes(x = time, y = value, color = stepMu)) +
    ylab("BA difference") +
    geom_line(size = .4) +
    NULL
```



```
res %>% group_by(stepMu) %>%
    summarise(min = min(value), max = max(value), d = max - min)
```

```
## # A tibble: 5 x 4
##   stepMu      min    max     d
##   <chr>     <dbl>  <dbl> <dbl>
## 1 a: 5e-2 -5.09   3.24   8.33
## 2 b: 1e-2 -1.07   0.686  1.75
## 3 c: 5e-3 -0.539  0.350  0.889
## 4 d: 1e-3 -0.120  0.0798 0.200
## 5 e: 5e-4 -0.0671 0.0458 0.113
```

## Future steps in dev

- Testing before simulation if all species are `IPM` or `mu` and if climate evolve.
- Test `climate` dimensions with `equil.time`.