RFate

Modelling vegetation spatially and temporally, through plant functional groups

Version 1.0

User Manual

Laboratoire d'Ecologie Alpine, UGA, CNRS, Grenoble, France,

May 2021

TABLE OF CONTENTS

1. Introduction

2. Plant Functional Groups

a. Principle

« The recurring suggestions are that models should explicitly (i) include spatiotemporal dynamics; (ii) consider multiple species in interactions and (iii) account for the processes shaping biodiversity distribution. » FATE is a « a biodiversity model that meets this challenge at regional scale by combining phenomenological and process-based approaches and using well-defined **plant functional group**. » (Boulangeat, 2014)

A plant functional group, or **PFG**, is « *a set of representative species [that] is classified based on key biological characteristics, to determine groups of species sharing ecological strategies.* » (Boulangeat, 2012)

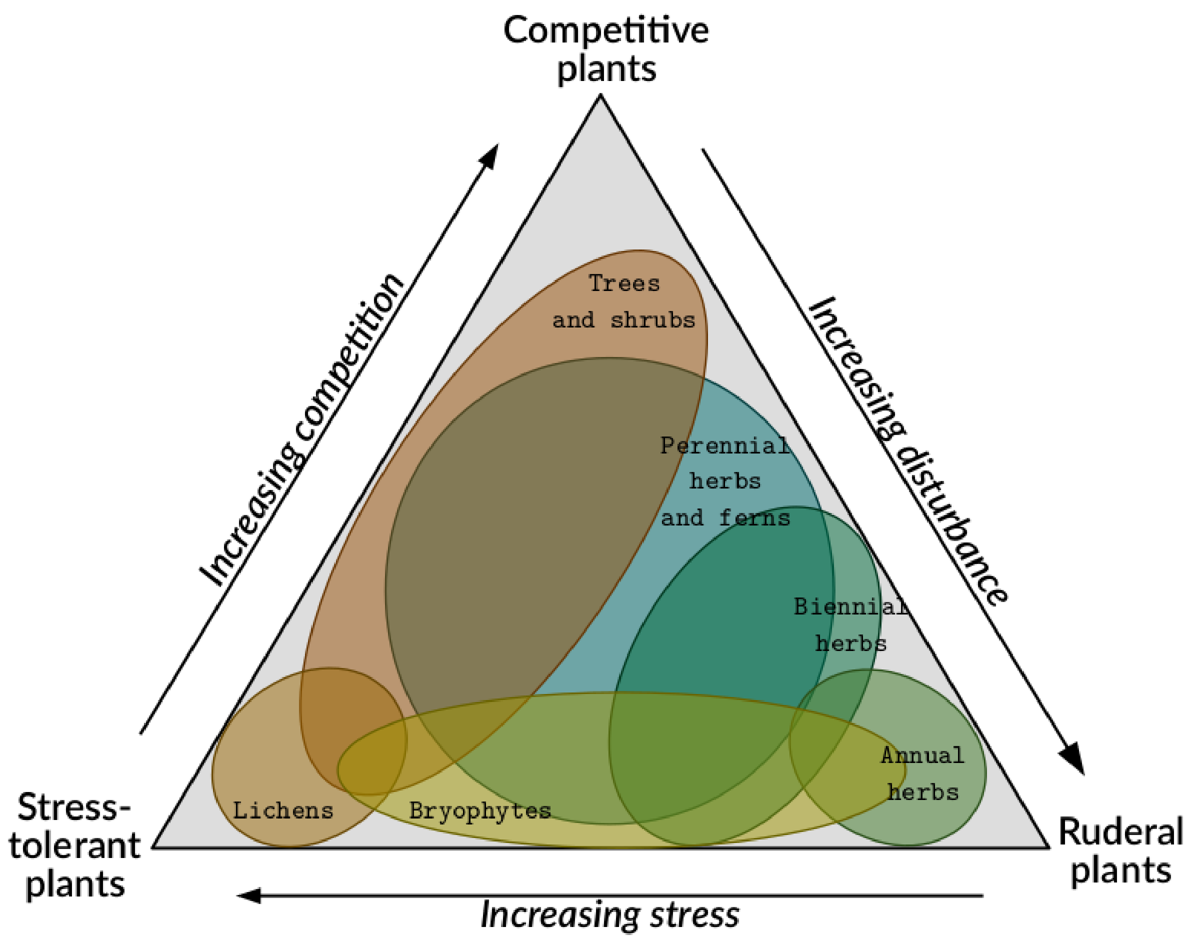


Figure 1: C-S-R triangle theory

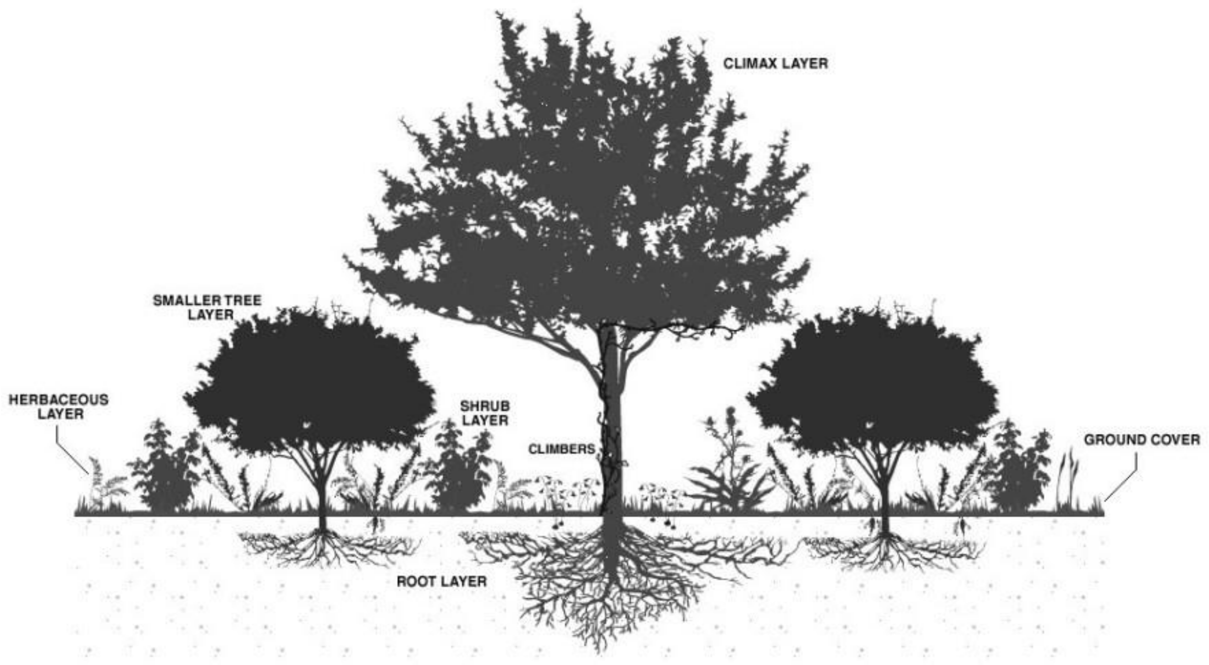
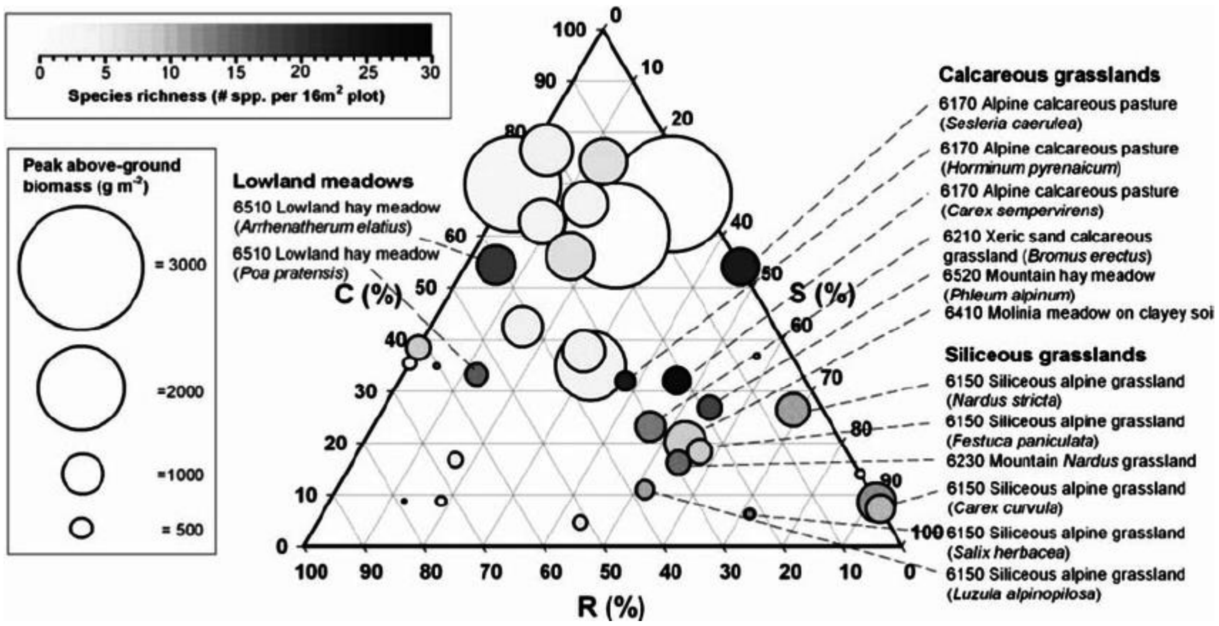


Figure 2: Vegetation layers

« ***Dominant species** are usually seen as the main drivers of vegetation dynamics and ecosystem functioning ('Biomass ratio hypothesis' (Grime, 1998),* Fig1*). Moreover, according to the well-known species-abundance distribution (Whittaker, 1965), just a few species produce most of the community's biomass.* ***In each vegetation strata (herbaceous, shrub, trees)*** (Fig2)*, these species are the most important, not only for structuring the landscape, but also explaining patterns of functional diversity.* » (Boulangeat, 2012)

Figure 3: Extract from Cerabolini 2014, Fig4



Building Plant Functional Group therefore consists of **bringing together plants that have similar traits and strategies** (Fig3, Fig4). Emphasis is placed on **dominant plants** that are supposed to structure the community, and groups are formed according to vegetation strata (and thus height). Soil conditions can also be taken into account (through trait measurement), as well as environmental or climatic conditions (through niche overlap).

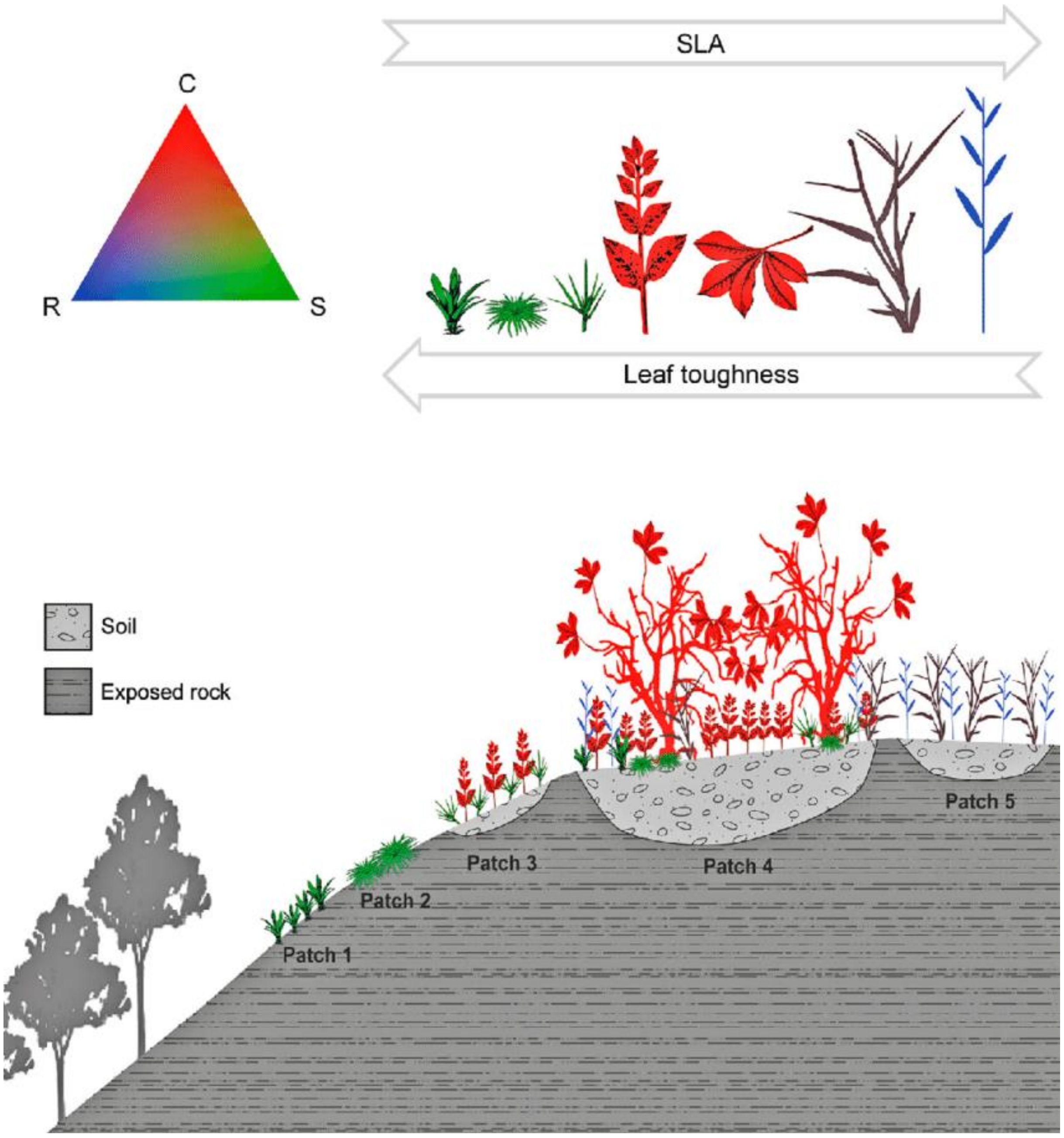


Figure 4: Extract from de Paula 2015, Fig4

Example of data required ?

Of course, anyone can build functional groups with any trait he wishes, as long as they reflect the strategies he wants to see emerge and study between groups. *Here is a naive example : focusing on desert plants, traits to build the groups could include the height (to distinguish cactus and trees from wildflowers), soil preferences (sand, rocks, tundra), leaf dry matter content (water storage strategy), the flowering type, etc.*

Here is an example of functional traits that can be used to build Plant Functional Group (and parametrize a FATE simulation) (Boulangeat, 2012, Boulangeat, 2014).

- Demographic characteristics :
  - Age at maturity
  - Longevity
  - Raunkiaer's life forms (Raunkiaer, 1934)
  - Height to represent a proxy for biomass
- Dispersal ability
  - Dispersal classes (Vittoz & Engler, 2007)
- Response to interactions
  - Height to represent the competitive ability for light
  - Indicator value for light requirement (Ellenberg et al. 1991, Landolt et al. 2010)
  - Indicator value for soil requirement (Ellenberg et al. 1991, Landolt et al. 2010)
- Response to disturbances
  - For grazing : palatability index based on pastoral values (Jouglet 1999)
  - For mowing : mowing tolerance

What are the key steps of this process ?

Since the basic idea of building Functional Group is to gather a lot of elements into a few, this implies two requirements :

1. that these elements are not too numerous
2. and that they are representative of the studied area, meaning not rare or outlier elements.
   This is the first step : the selection of dominant species.

In order to identify similarities between selected dominant species in terms of habitat, the climatic or environmental niche of each species is calculated and is compared with all the other dominant species niches.

The overlap of species environmental niches is obtained in second step.

Functional traits related to the fundamental process of growth are retrieved for each dominant species and mixed together to calculate functional distances between species.

Overlap of environmental niches and functional distances are combined to form a matrix of species pairwise distances.

Finally, based on this distance matrix, **species are clustered to find the best combination and obtain Functional Groups.**

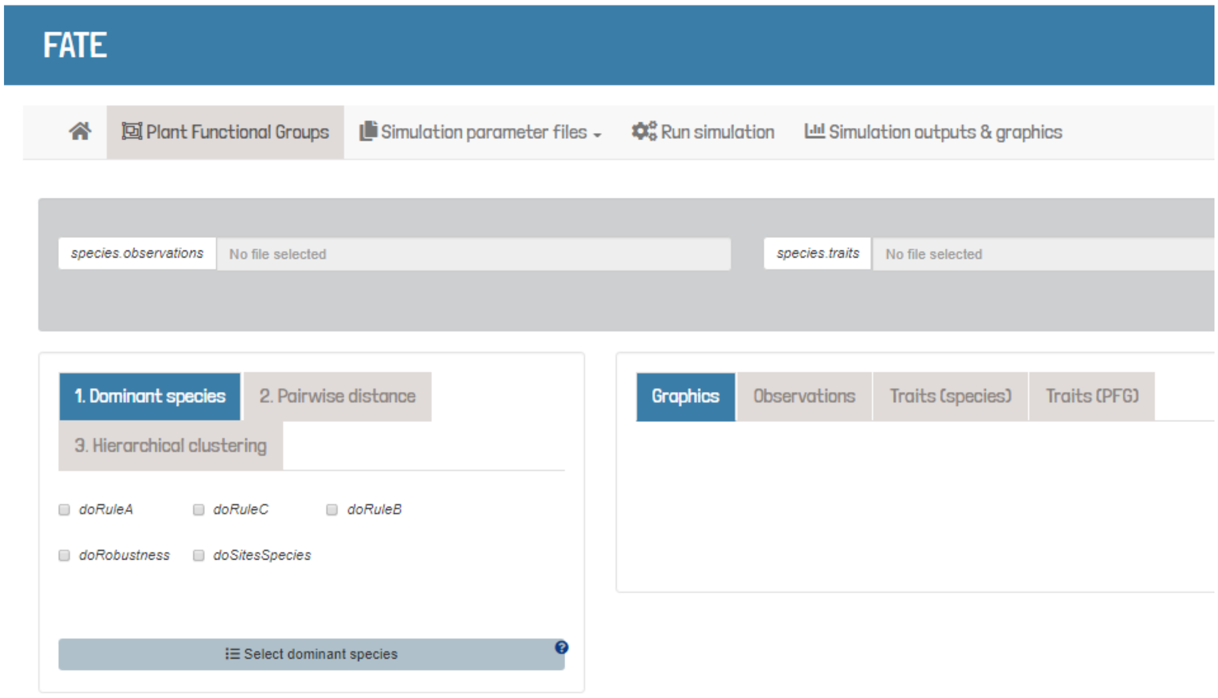
b. Building PFG

i. Selection of dominant species

- Gather **occurrences** for all species within the studied area
- Identify dominant species based on **abundances and frequençy of sampling**

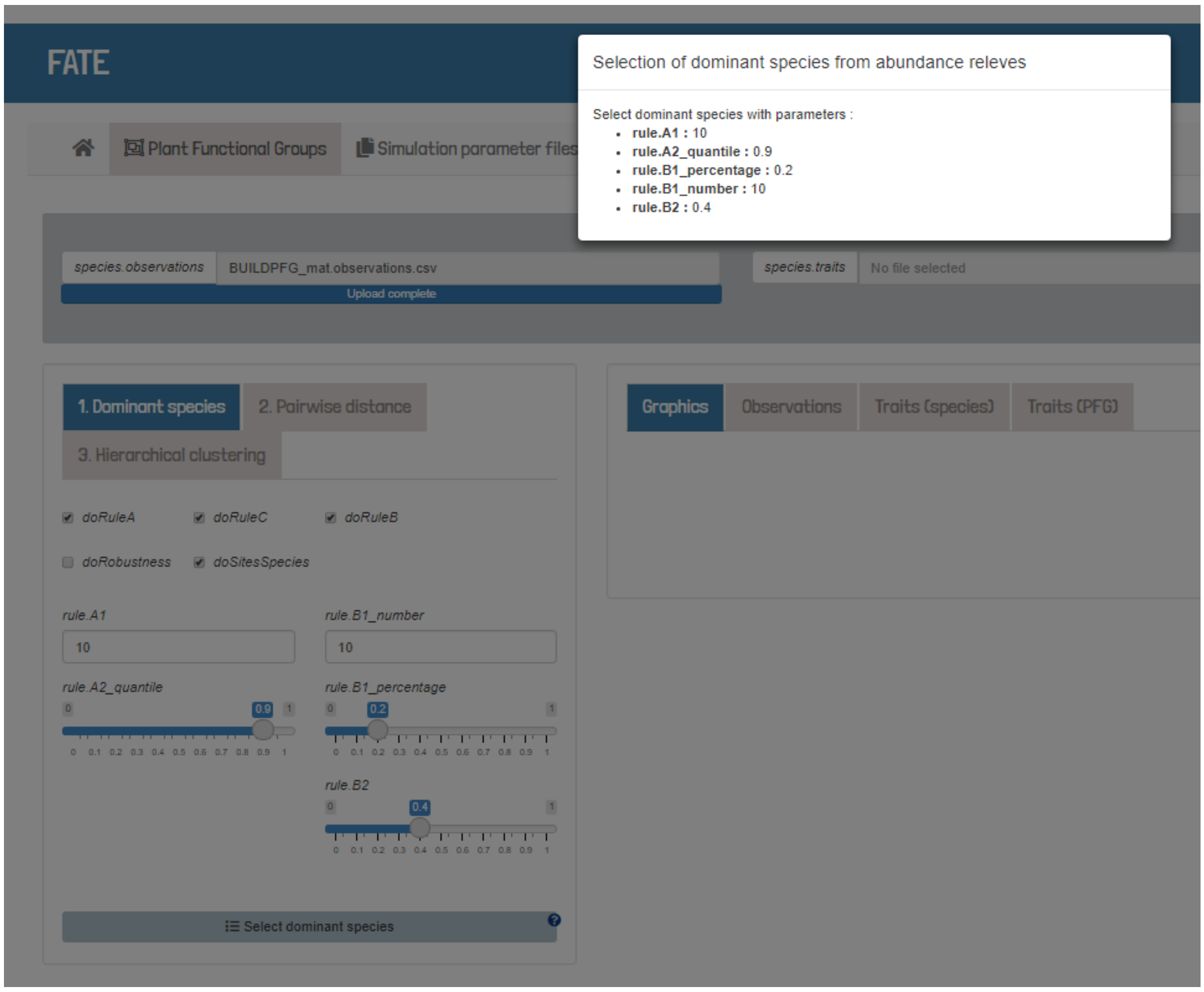
with the function PRE_FATE.selectDominant

```
## Load example data
Champsaur_PFG = .loadData("Champsaur_PFG", "RData")

## Species observations
tab = Champsaur_PFG$sp.observations

## No habitat, no robustness -------------------------------------------------
tab.occ = tab[, c("sites", "species", "abund")]
sp.SELECT = PRE_FATE.selectDominant(mat.observations = tab.occ)
names(sp.SELECT)
str(sp.SELECT$tab.rules)
plot(sp.SELECT$plot.A)
plot(sp.SELECT$plot.B$abs)
plot(sp.SELECT$plot.B$rel)

## Habitat, change parameters, no robustness (!quite long!) --------------------
tab.occ = tab[, c("sites", "species", "abund", "habitat")]
sp.SELECT = PRE_FATE.selectDominant(mat.observations = tab.occ
                                    , doRuleA = TRUE
                                    , rule.A1 = 10
                                    , rule.A2_quantile = 0.9
                                    , doRuleB = TRUE
                                    , rule.B1_percentage = 0.2
                                    , rule.B1_number = 10
                                    , rule.B2 = 0.4
                                    , doRuleC = TRUE)
names(sp.SELECT)
str(sp.SELECT$tab.rules)
plot(sp.SELECT$plot.C)
plot(sp.SELECT$plot.pco$Axis1_Axis2)
plot(sp.SELECT$plot.pco$Axis1_Axis3)
```



species.observations: No file selected

species.traits: No file selected



Selection of dominant species from abundance releves

Select dominant species with parameters :

- **rule.A1 :** 10
- **rule.A2_quantile :** 0.9
- **rule.B1_percentage :** 0.2
- **rule.B1_number :** 10
- **rule.B2 :** 0.4

species.observations: BUILDPFG_mat.observations.csv

species.traits: No file selected

[✓ doRuleA] [✓ doRuleC] [✓ doRuleB] doRobustness [✓ doSitesSpecies]

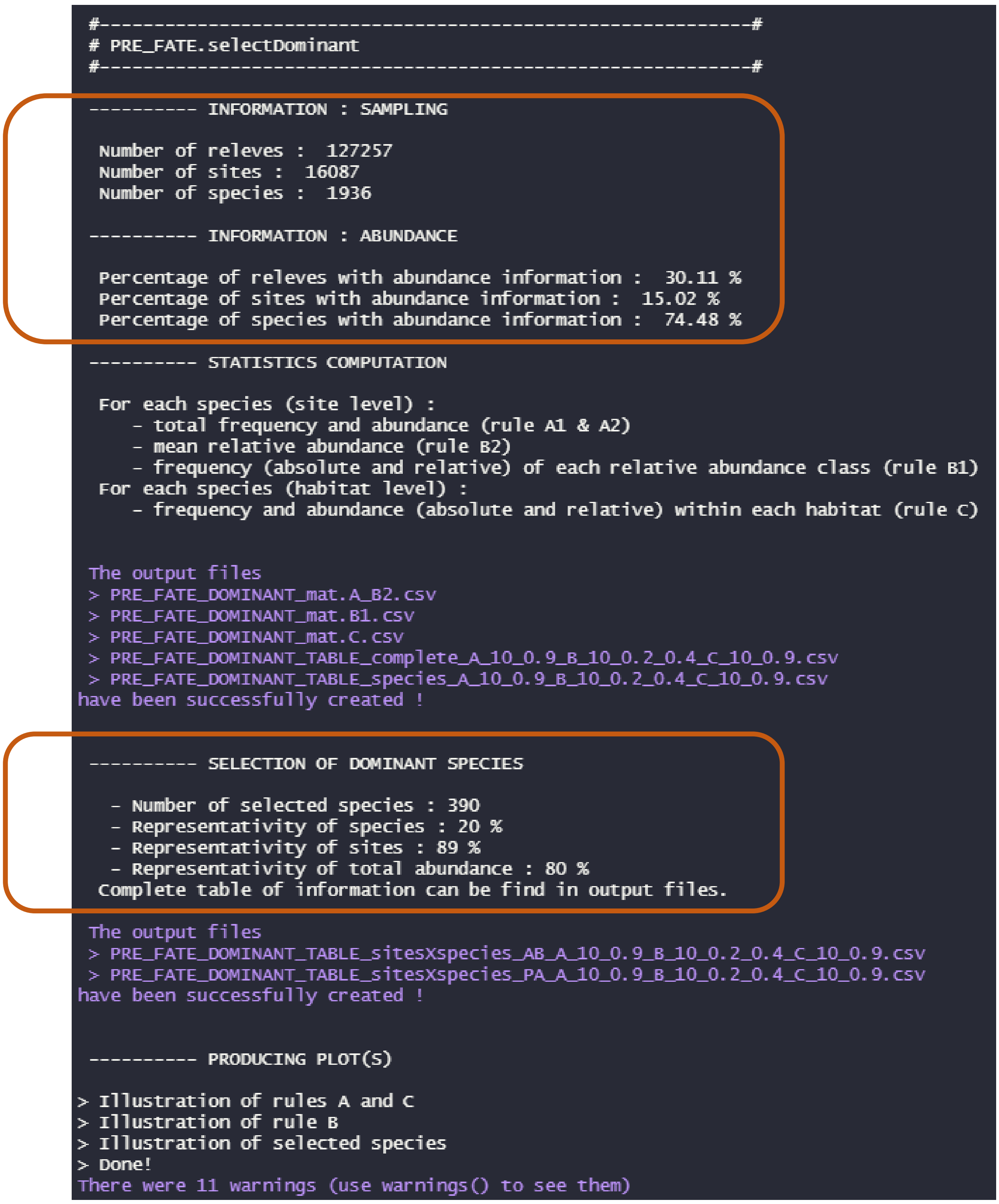
rule.A1: 10

rule.B1_number: 10

rule.A2_quantile: 0.9

rule.B1_percentage: 0.2

rule.B2: 0.4

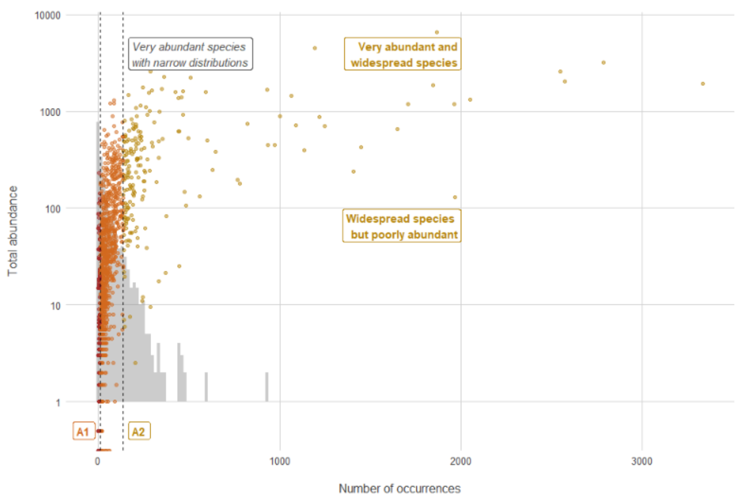


```
#-------------------------------------------------------------#
# PRE_FATE.selectDominant
#-------------------------------------------------------------#

 ---------- INFORMATION : SAMPLING

  Number of releves :  127257
  Number of sites :  16087
  Number of species :  1936

 ---------- INFORMATION : ABUNDANCE

  Percentage of releves with abundance information :  30.11 %
  Percentage of sites with abundance information :  15.02 %
  Percentage of species with abundance information :  74.48 %

 ---------- STATISTICS COMPUTATION

  For each species (site level) :
     - total frequency and abundance (rule A1 & A2)
     - mean relative abundance (rule B2)
     - frequency (absolute and relative) of each relative abundance class (rule B1)
  For each species (habitat level) :
     - frequency and abundance (absolute and relative) within each habitat (rule C)


 The output files
 > PRE_FATE_DOMINANT_mat.A_B2.csv
 > PRE_FATE_DOMINANT_mat.B1.csv
 > PRE_FATE_DOMINANT_mat.C.csv
 > PRE_FATE_DOMINANT_TABLE_complete_A_10_0.9_B_10_0.2_0.4_C_10_0.9.csv
 > PRE_FATE_DOMINANT_TABLE_species_A_10_0.9_B_10_0.2_0.4_C_10_0.9.csv
have been successfully created !


 ---------- SELECTION OF DOMINANT SPECIES

   - Number of selected species : 390
   - Representativity of species : 20 %
   - Representativity of sites : 89 %
   - Representativity of total abundance : 80 %
  Complete table of information can be find in output files.

 The output files
 > PRE_FATE_DOMINANT_TABLE_sitesXspecies_AB_A_10_0.9_B_10_0.2_0.4_C_10_0.9.csv
 > PRE_FATE_DOMINANT_TABLE_sitesXspecies_PA_A_10_0.9_B_10_0.2_0.4_C_10_0.9.csv
have been successfully created !


 ---------- PRODUCING PLOT(S)

> Illustration of rules A and C
> Illustration of rule B
> Illustration of selected species
> Done!
There were 11 warnings (use warnings() to see them)
```

Information provided by the function can help you explore your dataset, both in terms of « *what's there* » (Fig, 1st orange box : `INFORMATION : SAMPLING` and `INFORMATION : ABUNDANCE` parts) and « *what do I select* » (Fig, 2nd orange box : `SELECTION OF DOMINANT SPECIES`). If you want to use the selection criteria B, it requires abundance information and might be applied only on part of your data. It is up to the user to decide whether this part is sufficiently representative of the whole dataset. As for the species selected, final decision of « *is this set of species representative of dominance inside my dataset* », although a common recommandation for calculating community weighted mean is to have at least a representativity of 80% of the total abundance of the studied area and such a threshold could be similarly used here.

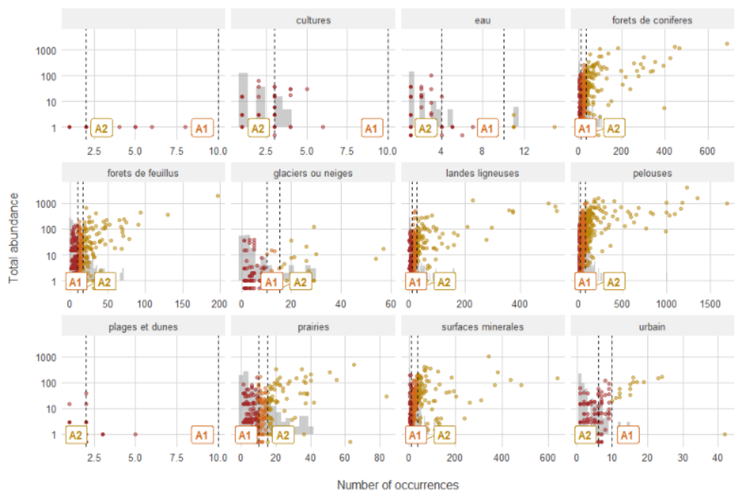
STEP 1 : Selection of dominant species - rule A

Criteria concerning occurrences within all sites :
 > A1 = minimum number of releves required : 10
 > A2 = minimum number of occurrences required : quantile(90%) = 138



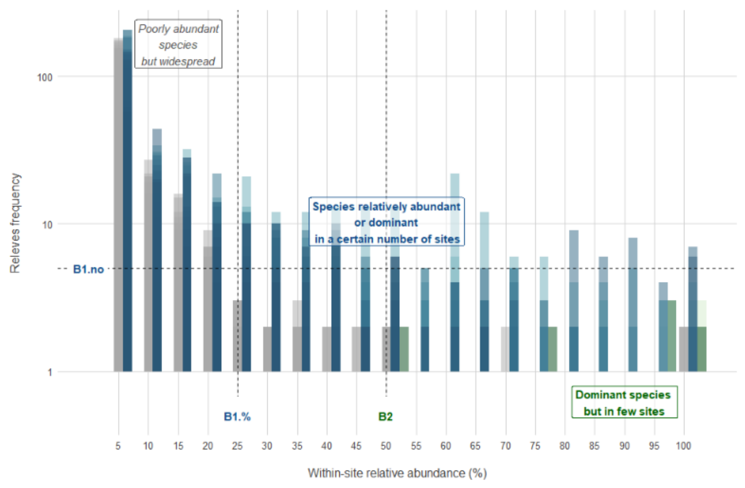
STEP 1 : Selection of dominant species - rule C

Criteria concerning occurrences within all habitats :
 > A1 = minimum number of releves required : 10
 > A2 = minimum number of occurrences required : quantile(90%) = : 2 / cultures: 3 / eau: 4 / forets de coniferes: 34 / forets de feuillu



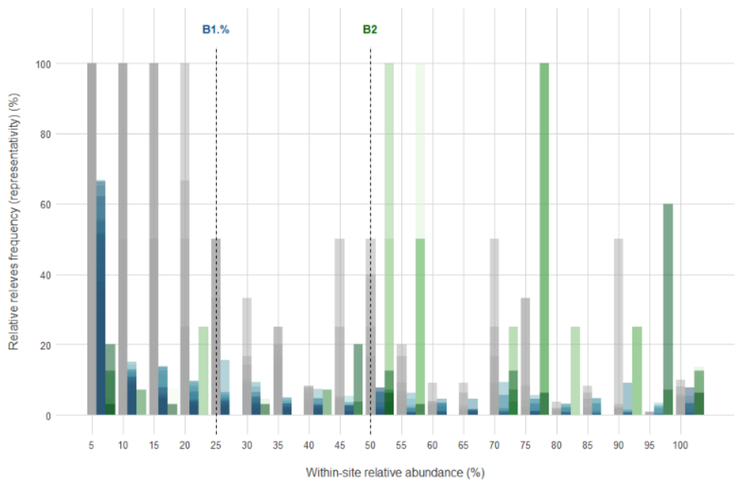
STEP 1 : Selection of dominant species - rule B

Criteria concerning abundances within all concerned sites :
 > B1.no = minimum number of sites required : 5
 > B1.% = with a minimum relative abundance of : 25%
 > B2 = minimum mean relative abundance required : 50%



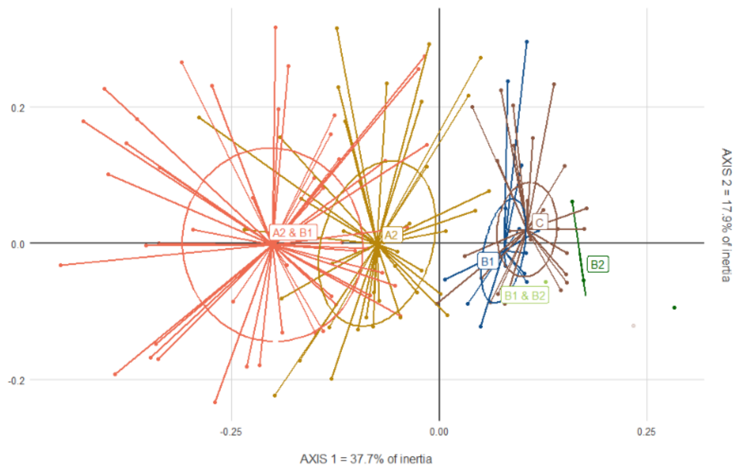
STEP 1 : Selection of dominant species - rule B

Criteria concerning abundances within all concerned sites :
 > B1.no = minimum number of sites required : 5
 > B1.% = with a minimum relative abundance of : 25%
 > B2 = minimum mean relative abundance required : 50%



STEP 2 : Selected dominant species

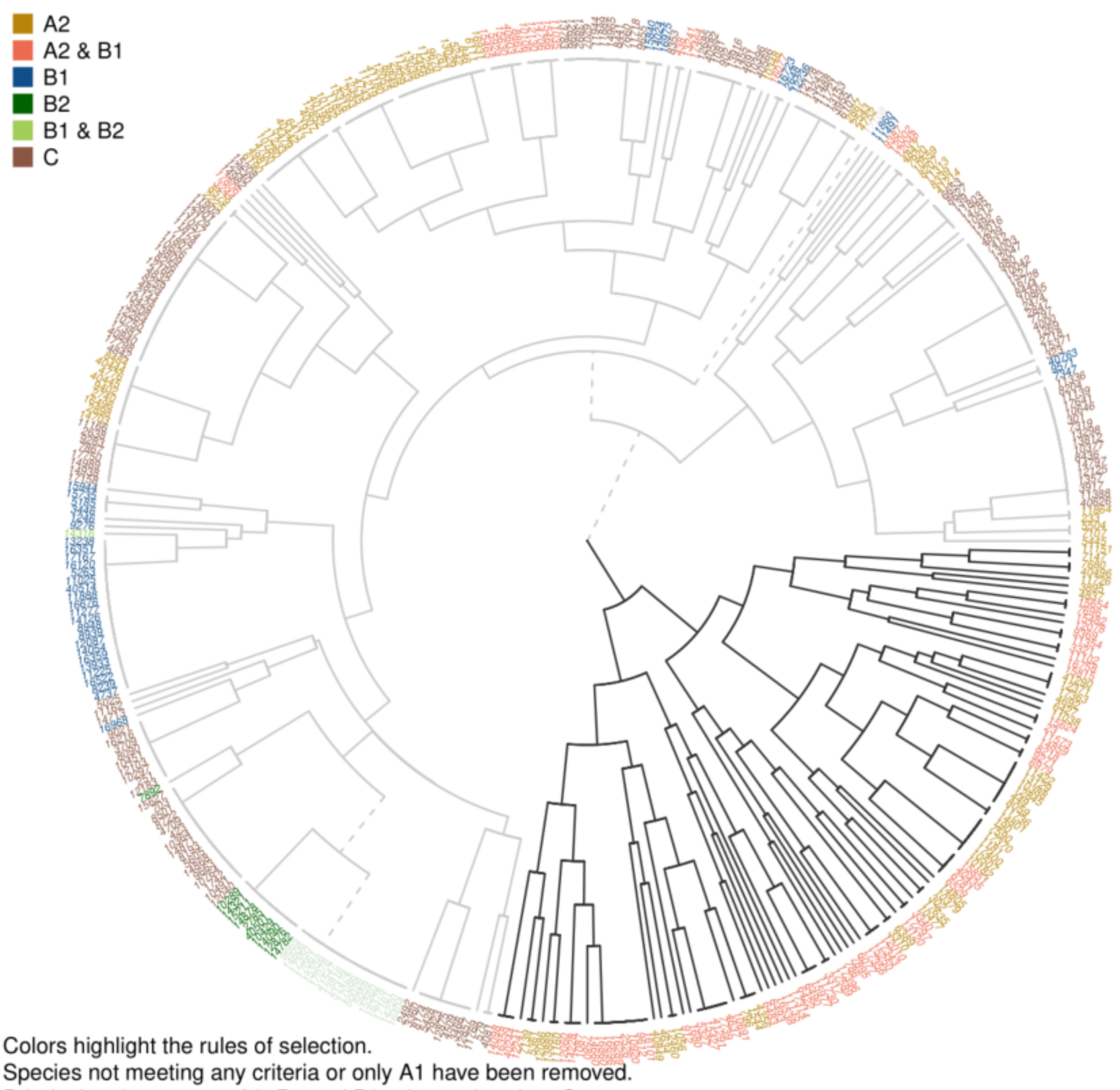
Colors highlight the rules of selection.
Species not meeting any criteria or only A1 have been removed.
Priority has been set to A2, B1 and B2 rules, rather than C.
Hence, species selected according to A2, B1 and/or B2 can also meet criterion C
while species selected according to C do not meet any of the three criteria.
Species selected according to one (or more) criterion but not meeting criterion A1 are transparent.



Graphics are produced to help you visualize your species selection, as well as the tables containing the values calculated and used to obtain such selection.

They are automatically saved, as `.csv` and `.pdf` files, but the production of plots can be desacti-vated to save some computation time.

STEP 2 : Selected dominant species



Colors highlight the rules of selection.
Species not meeting any criteria or only A1 have been removed.
Priority has been set to A2, B1 and B2 rules, rather than C.
Hence, species selected according to A2, B1 and/or B2 can also meet criterion C
while species selected according to C do not meet any of the three criteria.
Species selected according to one (or more) criterion but not meeting criterion A1 are transparent.

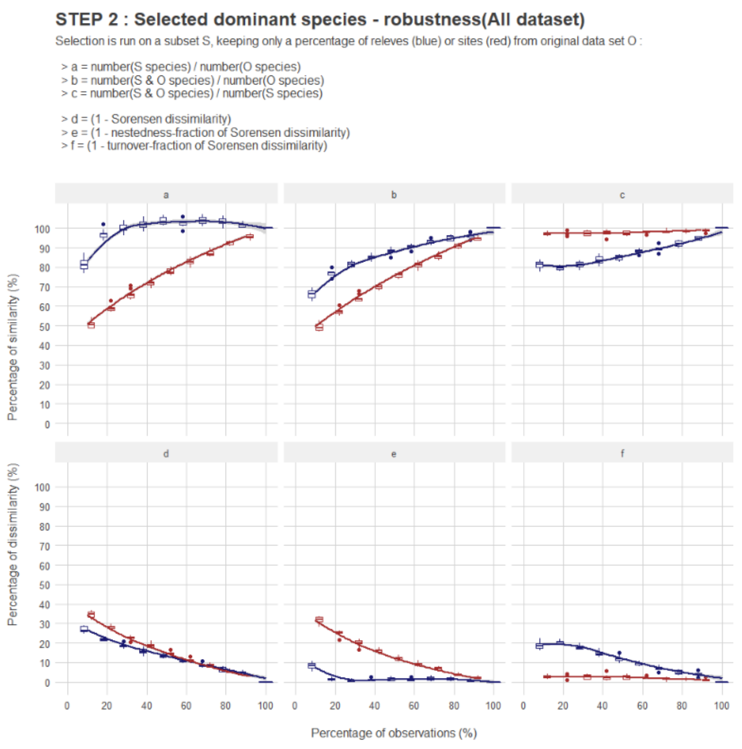
Representation into ordination space or phylogeny can help exploring by which criterion the species are selected, and maybe adjusting selection parameter values. *For example, if you decide to use selection criterion C, but no species are selected with it, you might want to change your parameter values. It might also mean that either you do not have species specific to your habitat level, or they might already be selected by A or B criteria.*

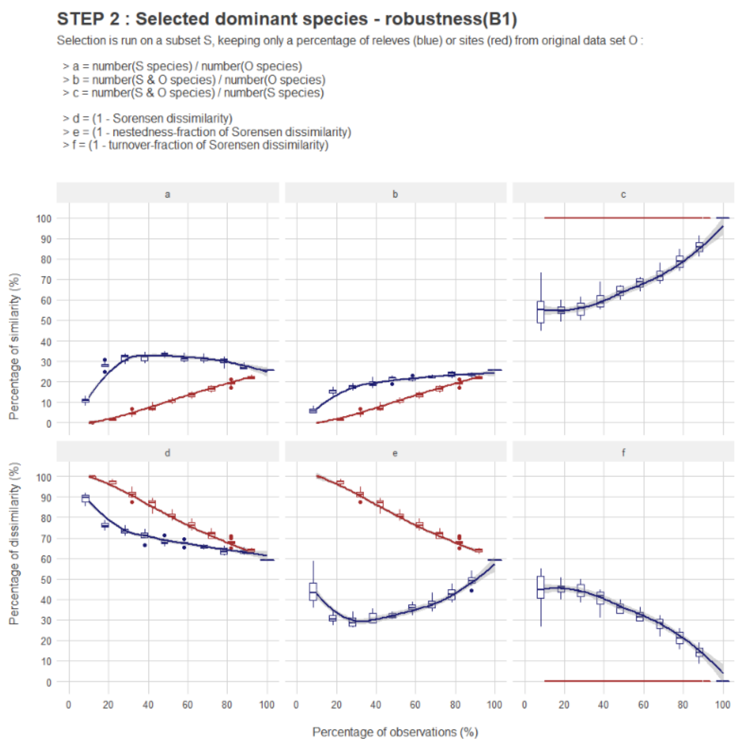
```
## No habitat, robustness (!quite long!) --------------------
tab.occ = tab[, c("sites", "species", "abund")]
sp.SELECT = PRE_FATE.selectDominant(mat.observations = tab.occ
                                    , opt.doRobustness = TRUE
                                    , opt.robustness_percent = seq(0.1,0.9,0.1)
                                    , opt.robustness_rep = 10)
names(sp.SELECT)
str(sp.SELECT$tab.robustness)
names(sp.SELECT$plot.robustness)
plot(sp.SELECT$plot.robustness$`All dataset`)
```

The exploratory robustness module was created following a questioning about the robustness of the dominant species selection against changes in the observation dataset. The underlying question was : *for example, if I get my data from an observatory program that updates the dataset every year, is my selection going to change a lot each time I add more observations ?*

Dominant selection can therefore be run, taking only a subset of the observation dataset, removing either single species observations (blue), or complete sites observations (red). The larger (smaller) the values for a/b/c (d/e/f) measures, the lower the impact of removing observations.

a.	100%	= same number of selected species in S and O
b.	100%	= species in O are all in S (O is included into S)
c.	100%	= species in S are all in O (S is included into O)
d.	0%	= S is identical to O (same number of species + same species)
	> 0%	= difference between S and O
e.	0%	= difference between S and O is enterely due to turnover
	> 0%	= same species than O, but a subset
f.	0%	= difference between S and O is enterely due to nestedness
	> 0%	= same number of species but some that are different





ii. Overlap of species environmental niches

- *Option 1: Principal Component analysis*
  - Gather **environmental data** for the studied area
  - Compute **PCA** over environment to create a *climatic/habitat space*
  - Calculate the **density of each species** within this *climatic/habitat space* from the PCA
  - For each pair of species, compute the **overlap** of the 2 considered species within the *climatic/habitat space*

- *Option 2: Species Distribution Models*
  - Gather **environmental data** for the studied area
  - For each dominant species, compute a **species distribution model** (SDM)
  - combining environmental data and occurrences to determine the *climatic/habitat niche* of the species
  - With these SDMs, calculate the **niche overlap** of each pair of species

This step represents a complete other subject (species distribution modelling and overlap) and other packages are dedicated to these computations (`ecospat`, `biomod2`, `phyloclim`, etc). However, some computations can be made through the PRE_FATE.speciesDistance function presented here-after.

- *Option 1* can be realized with the help of the `ecospat` package, giving a sites by species occurrence matrix (as produced by the PRE_FATE.selectDominant function) together with a corresponding sites by environment matrix.
- *Option 2* can only be partially executed through the RFate package : a list of species distribution maps can be given to the function and a matrix of niche overlap distances will be calculated with the help of the `phyloclim` package.

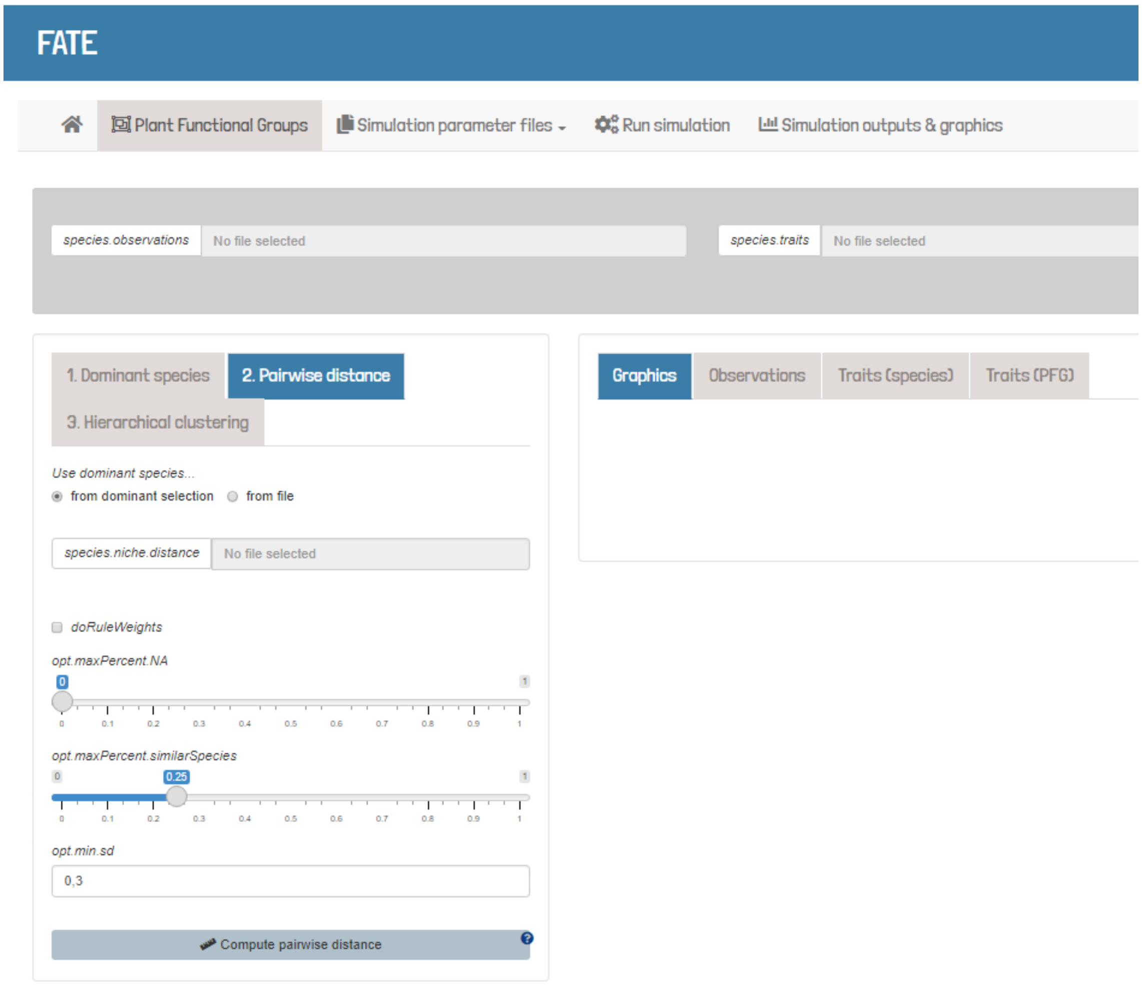
If you do not want to fullfill this step, you can simply build a species by species matrix and fill it with 1 everywhere (meaning that all species share the same environmental niche), and use it for the next step.

iii. Species pairwise distance

by combining overlap and functional distances with the function PRE_FATE.speciesDistance

- Gather **traits data** for all dominant species within the studied area (traits need to be related to fundamental process of growth : light tolerance, dispersal, height…)
- Compute **dissimilarity distances** between pairs of species based on these traits and taking also into account the overlap of the 2 species within the *environmental space* (see previous step)

```
## Load example data
Champsaur_PFG = .loadData("Champsaur_PFG", "RData")

## Species traits
tab.traits = Champsaur_PFG$sp.traits
tab.traits = tab.traits[, c("species", "GROUP", "MATURITY", "LONGEVITY"
                            , "HEIGHT", "DISPERSAL", "LIGHT", "NITROGEN")]
str(tab.traits)

## Species niche overlap (dissimilarity distances)
tab.overlap = 1 - Champsaur_PFG$mat.overlap ## transform into similarity
tab.overlap[1:5, 1:5]

## Give warnings -----------------------------------------------------------
sp.DIST = PRE_FATE.speciesDistance(mat.traits = tab.traits
                                   , mat.overlap.option = "dist"
                                   , mat.overlap.object = tab.overlap)
str(sp.DIST)

## Change parameters to allow more NAs (and change traits used) --------------
sp.DIST = PRE_FATE.speciesDistance(mat.traits = tab.traits
                                   , mat.overlap.option = "dist"
                                   , mat.overlap.object = tab.overlap
                                   , opt.maxPercent.NA = 0.05
                                   , opt.maxPercent.similarSpecies = 0.3
                                   , opt.min.sd = 0.3)
str(sp.DIST)

require(foreach); require(ggplot2); require(ggdendro)
pp = foreach(x = names(sp.DIST$mat.ALL)) %do%
{
  hc = hclust(sp.DIST$mat.ALL[[x]])
  pp = ggdendrogram(hc, rotate = TRUE) +
    labs(title = paste0("Hierarchical clustering based on species distance "
                        , ifelse(length(names(sp.DIST$mat.ALL)) > 1
                        , paste0("(group ", x, ")")
                        , "")))
  return(pp)
}
plot(pp[[1]])
plot(pp[[2]])
plot(pp[[3]])
```



FATE

Plant Functional Groups | Simulation parameter files | Run simulation | Simulation outputs & graphics

species.observations No file selected | *species.traits* No file selected

1. Dominant species | 2. Pairwise distance | 3. Hierarchical clustering

Use dominant species...
[✓ from dominant selection] from file

species.niche.distance No file selected

doRuleWeights

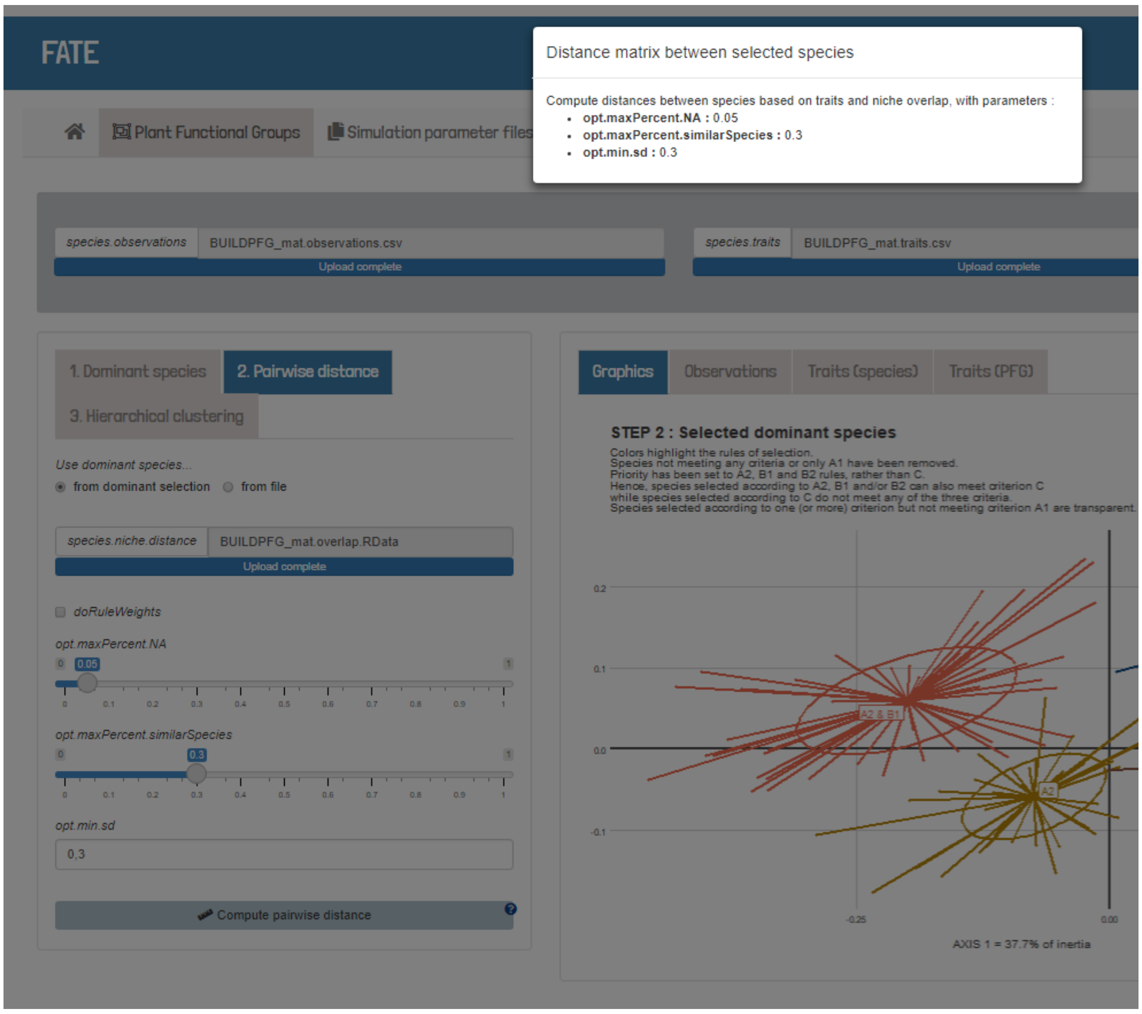
opt.maxPercent.NA 0

opt.maxPercent.similarSpecies 0.25

opt.min.sd 0,3

Compute pairwise distance

Graphics | Observations | Traits (species) | Traits (PFG)



Distance matrix between selected species

Compute distances between species based on traits and niche overlap, with parameters :

- **opt.maxPercent.NA :** 0.05
- **opt.maxPercent.similarSpecies :** 0.3
- **opt.min.sd :** 0.3

species.observations BUILDPFG_mat.observations.csv — Upload complete

species.traits BUILDPFG_mat.traits.csv — Upload complete

species.niche.distance BUILDPFG_mat.overlap.RData — Upload complete

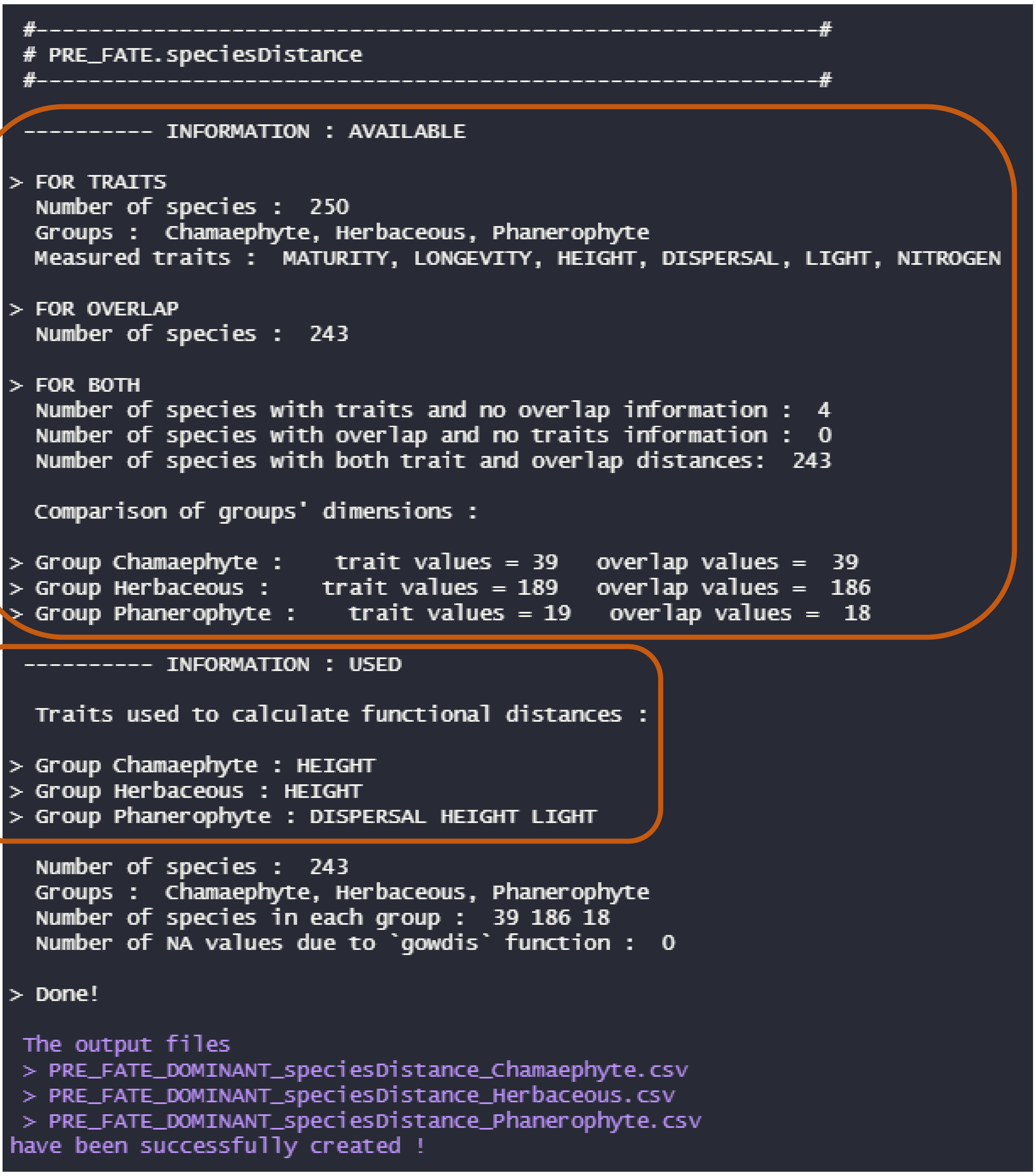
opt.maxPercent.NA 0.05

opt.maxPercent.similarSpecies 0.3

opt.min.sd 0,3

STEP 2 : Selected dominant species

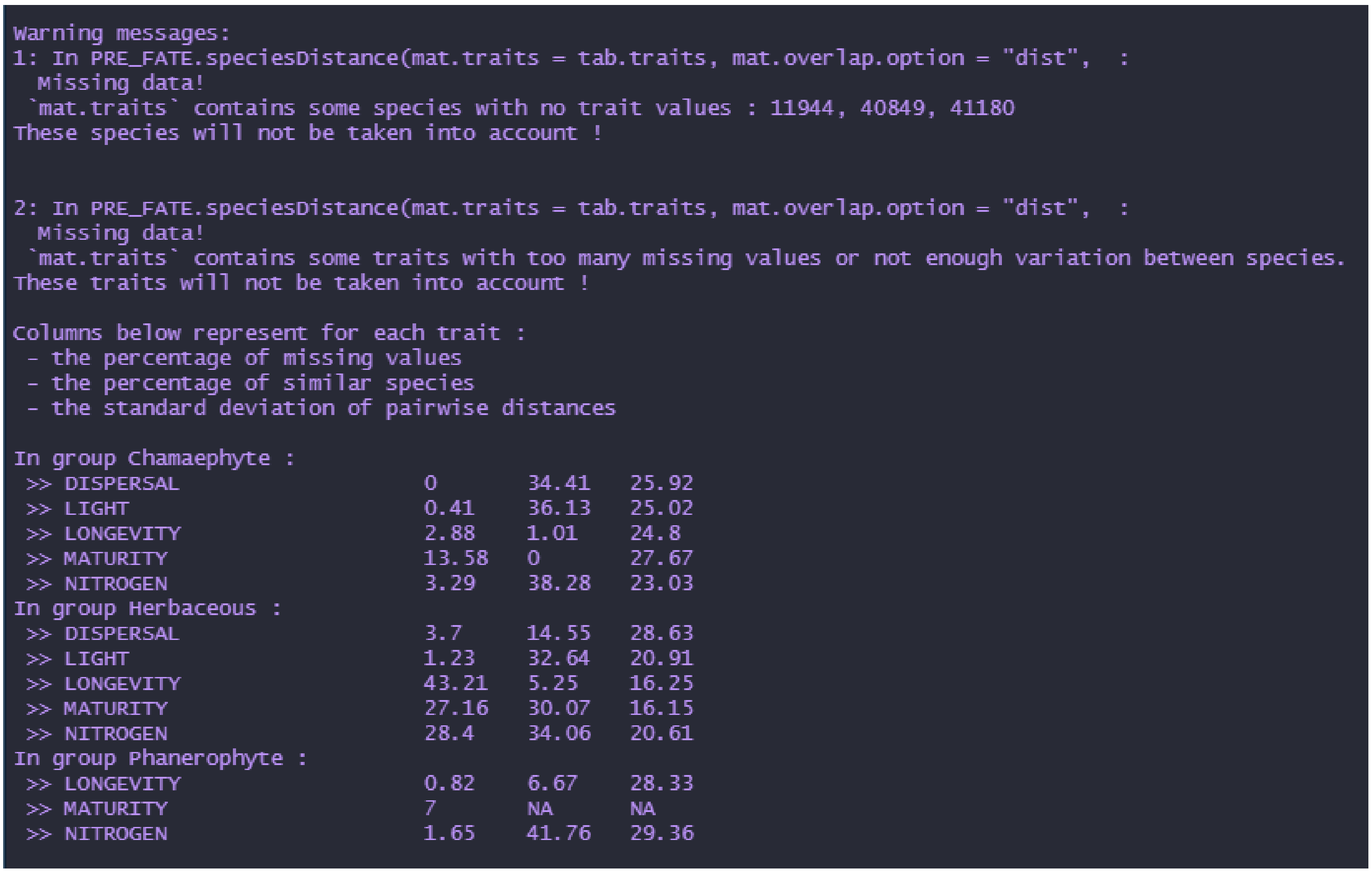
Colors highlight the rules of selection.
Species not meeting any criteria or only A1 have been removed.
Priority has been set to A2, B1 and B2 rules, rather than C.
Hence, species selected according to A2, B1 and/or B2 can also meet criterion C
while species selected according to C do not meet any of the three criteria.
Species selected according to one (or more) criterion but not meeting criterion A1 are transparent.


```
 #--------------------------------------------------------------#
 # PRE_FATE.speciesDistance
 #--------------------------------------------------------------#

 ---------- INFORMATION : AVAILABLE

> FOR TRAITS
  Number of species :  250
  Groups :  Chamaephyte, Herbaceous, Phanerophyte
  Measured traits :  MATURITY, LONGEVITY, HEIGHT, DISPERSAL, LIGHT, NITROGEN

> FOR OVERLAP
  Number of species :  243

> FOR BOTH
  Number of species with traits and no overlap information :  4
  Number of species with overlap and no traits information :  0
  Number of species with both trait and overlap distances:  243

  Comparison of groups' dimensions :

> Group Chamaephyte :    trait values = 39   overlap values =  39
> Group Herbaceous :    trait values = 189   overlap values =  186
> Group Phanerophyte :    trait values = 19   overlap values =  18
 ---------- INFORMATION : USED

  Traits used to calculate functional distances :

> Group Chamaephyte : HEIGHT
> Group Herbaceous : HEIGHT
> Group Phanerophyte : DISPERSAL HEIGHT LIGHT

  Number of species :  243
  Groups :  Chamaephyte, Herbaceous, Phanerophyte
  Number of species in each group :  39 186 18
  Number of NA values due to `gowdis` function :  0

> Done!

 The output files
 > PRE_FATE_DOMINANT_speciesDistance_Chamaephyte.csv
 > PRE_FATE_DOMINANT_speciesDistance_Herbaceous.csv
 > PRE_FATE_DOMINANT_speciesDistance_Phanerophyte.csv
have been successfully created !
```

As for the selection of dominant, the computation of species distances returns lot of information to help you explore your traits dataset for your selected species. Once again, in terms of « what's there » (Fig, 1st orange box : `INFORMATION : AVAILABLE` part) and « *what do I select* » (Fig, 2nd orange box : `INFORMATION : USED`).

A distance can only be calculated for a specific species if it has both trait(s) and overlap information. Missing one or the another causes the species to be removed from the calculation. This is reflected into the `INFORMATION : AVAILABLE` part.

The `INFORMATION : USED` part indicates the trait(s) *really* used for each group to calculate functional distances. *For example here, only* `HEIGHT` *is kept for Chamaephyte, while 6 traits were given as inputs.*



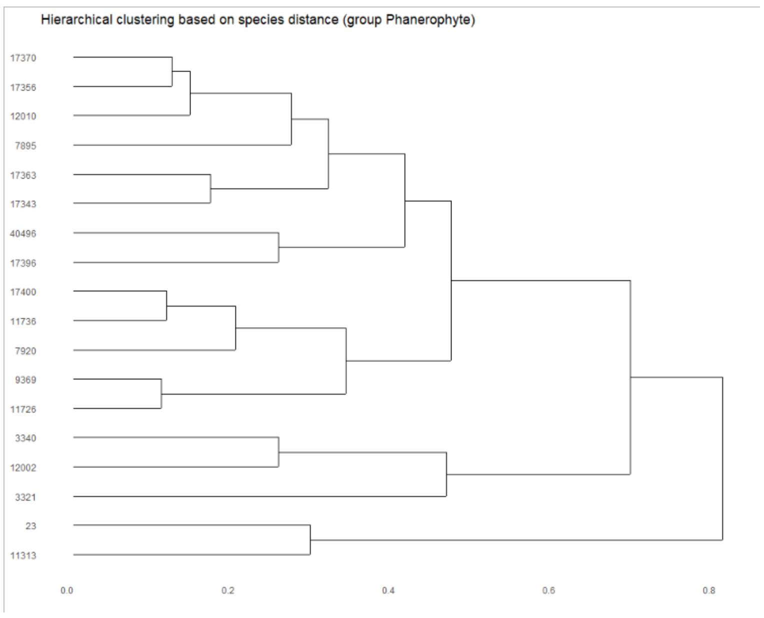
```
Warning messages:
1: In PRE_FATE.speciesDistance(mat.traits = tab.traits, mat.overlap.option = "dist",  :
  Missing data!
 `mat.traits` contains some species with no trait values : 11944, 40849, 41180
These species will not be taken into account !


2: In PRE_FATE.speciesDistance(mat.traits = tab.traits, mat.overlap.option = "dist",  :
  Missing data!
 `mat.traits` contains some traits with too many missing values or not enough variation between species.
These traits will not be taken into account !

Columns below represent for each trait :
 - the percentage of missing values
 - the percentage of similar species
 - the standard deviation of pairwise distances

In group Chamaephyte :
 >> DISPERSAL                        0       34.41    25.92
 >> LIGHT                            0.41    36.13    25.02
 >> LONGEVITY                        2.88    1.01     24.8
 >> MATURITY                         13.58   0        27.67
 >> NITROGEN                         3.29    38.28    23.03
In group Herbaceous :
 >> DISPERSAL                        3.7     14.55    28.63
 >> LIGHT                            1.23    32.64    20.91
 >> LONGEVITY                        43.21   5.25     16.25
 >> MATURITY                         27.16   30.07    16.15
 >> NITROGEN                         28.4    34.06    20.61
In group Phanerophyte :
 >> LONGEVITY                        0.82    6.67     28.33
 >> MATURITY                         7       NA       NA
 >> NITROGEN                         1.65    41.76    29.36
```

This is a good example why warning messages ARE important and should ALWAYS be carefully checked.

First warning indicates species with no trait values that will be removed from the distance calculation. Second warning gives a table of values for each trait in each group : the percentages of missing values and similar species for the concerned trait, as well as the standard deviation of pairwise distances. They are all compared to parameters of the PRE_FATE.speciesDistance function : `opt.maxPercent.NA`, `opt.maxPercent.similarSpecies` and `opt.min.sd` respectively. If a trait does not fullfill the required condition, it is removed from the computation for this specific group. Hence, it is possible not having the same set of traits used between groups. If this is something that you do not want, you just have to fix the function parameters to low tolerance values (*for example : 1, 1 and 0 respectively*).

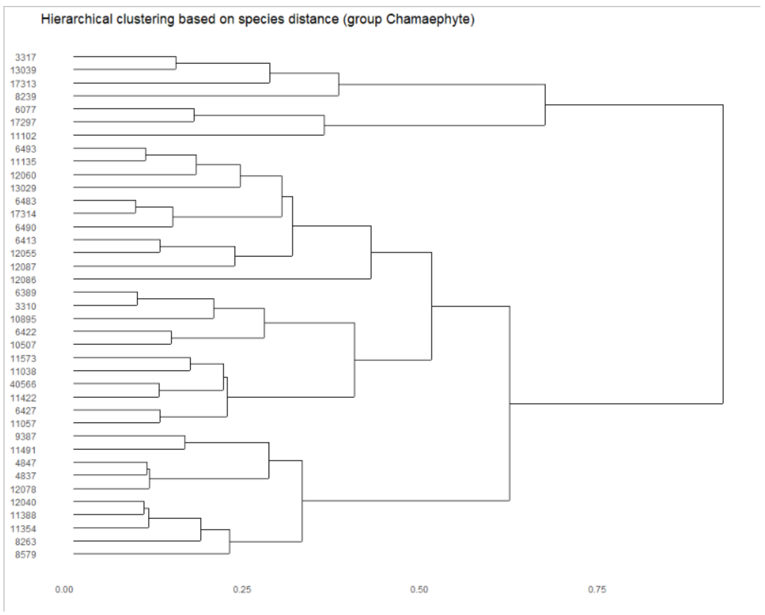
Once species pairwise distances calculated, they are in the form of species by species matrix, but can be easily represented through dendrograms : it gives an overview of what is going to be done in the following steps.

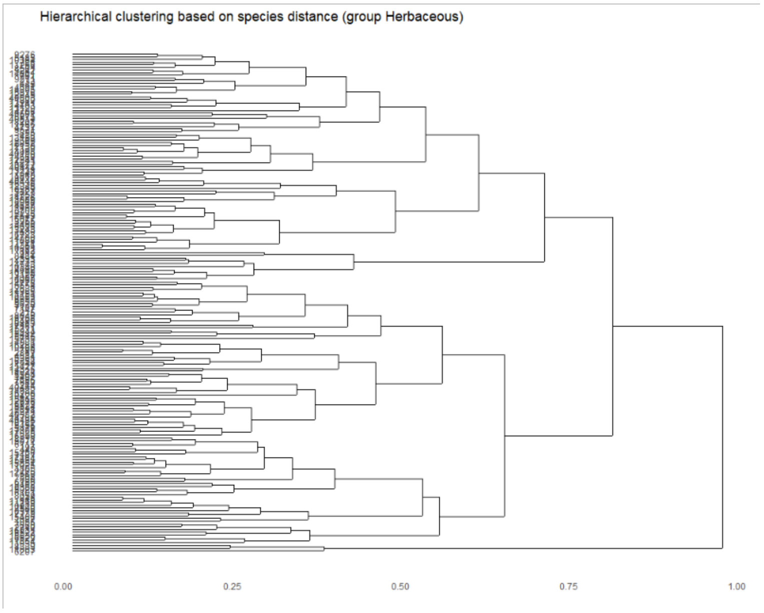
The distance matrix obtained (one for each given category, *here 3 : Phanerophyte, Chamaephyte and Herbaceous*) is classifying species according to the traits used, the overlap distances, and the weights given to these two sources of information.



Clustering and choice of functional group will just consist in cutting these distance trees at a certain height, thus giving a certain number of groups for each category.

The next function realize this cut and give some information to help the user decide the final number of groups to keep.



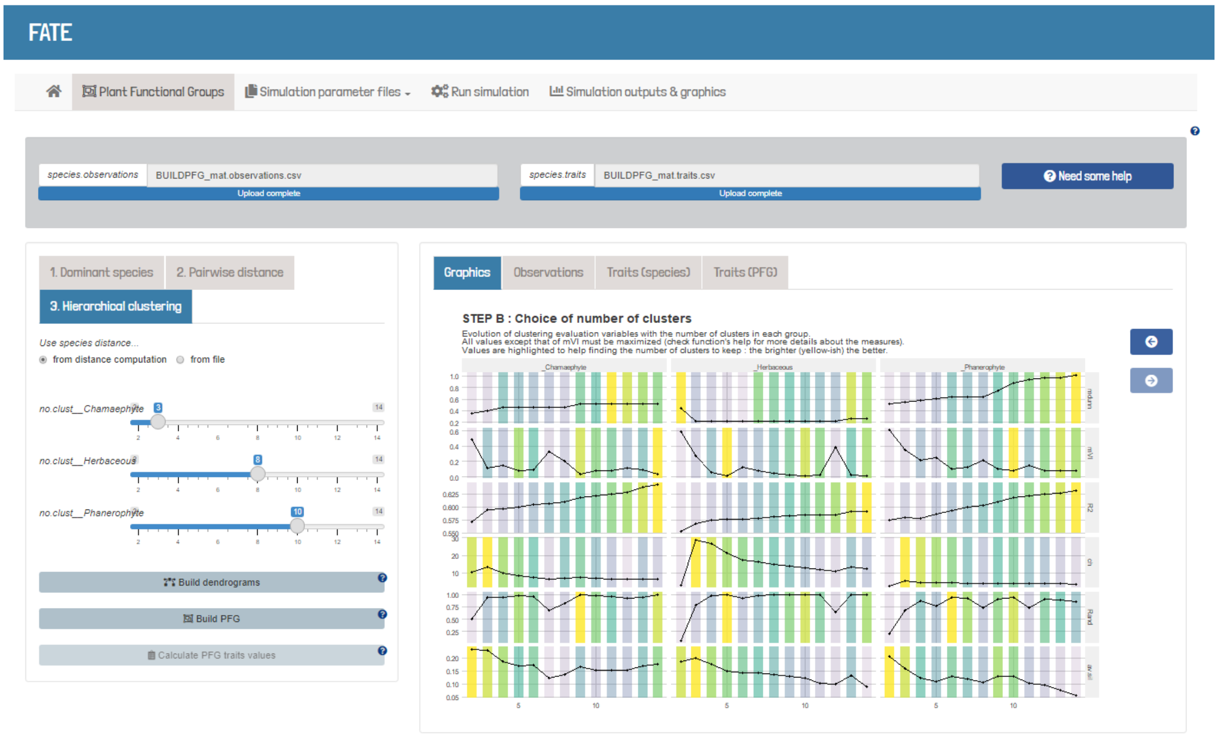


iv. Clustering

using the **dissimilarity distances** from previous step :

- calculate all possible **clusters**, and the corresponding evaluation metrics with the function PRE_FATE.speciesClustering_step1
- choose the best number of clusters from the previous step and find **determinant species** with the function PRE_FATE.speciesClustering_step2
- combine traits data and clustering to calculate mean / median trait values per PFG with the function PRE_FATE.speciesClustering_step3

```r
## Load example data
Champsaur_PFG = .loadData("Champsaur_PFG", "RData")

## Species dissimilarity distances (niche overlap + traits distance)
tab.dist = list("Phanerophyte" = Champsaur_PFG$sp.DIST.P$mat.ALL
                , "Chamaephyte" = Champsaur_PFG$sp.DIST.C$mat.ALL
                , "Herbaceous" = Champsaur_PFG$sp.DIST.H$mat.ALL)
str(tab.dist)
as.matrix(tab.dist[[1]])[1:5, 1:5]

## Build dendrograms -----------------------------------------------------
sp.CLUST = PRE_FATE.speciesClustering_step1(mat.species.DIST = tab.dist)
names(sp.CLUST)
str(sp.CLUST$clust.evaluation)
plot(sp.CLUST$plot.clustMethod)
plot(sp.CLUST$plot.clustNo)
```



Use species distance...
[✓ from distance computation] from file

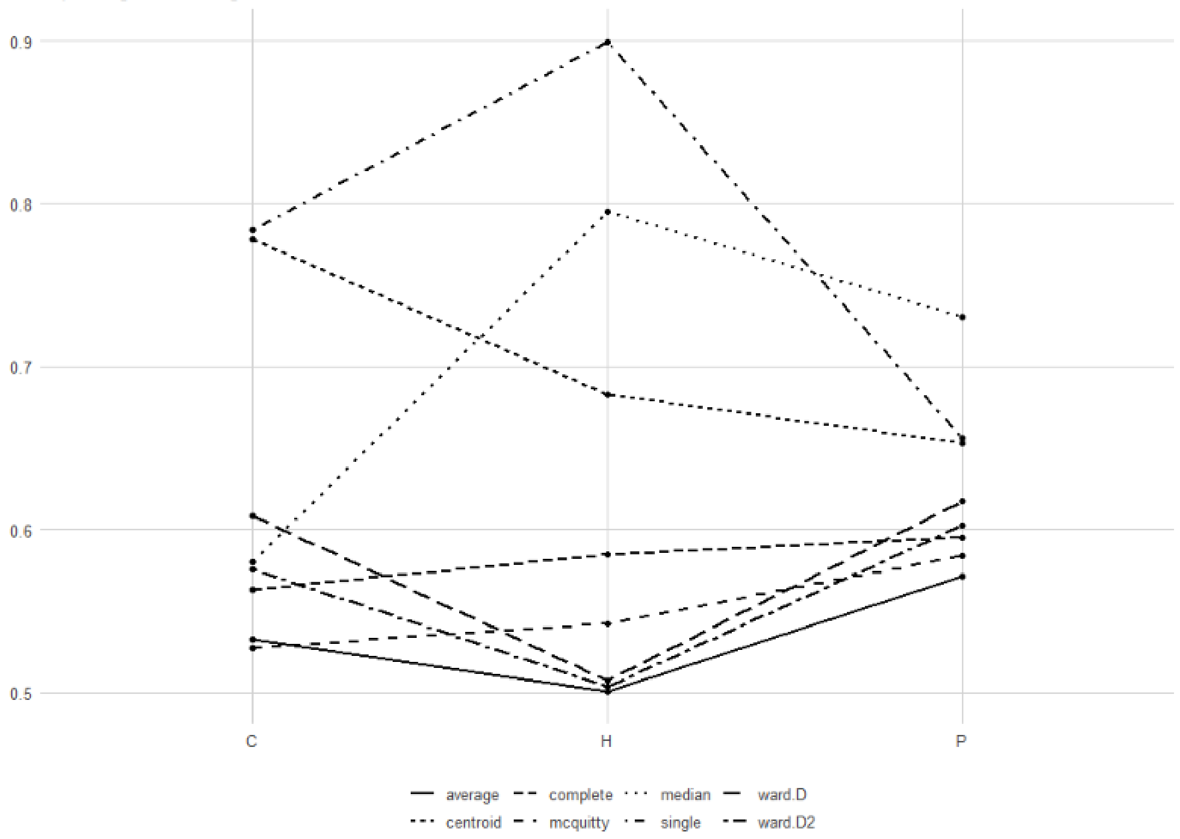
no.clust__Chamaephyte 3
no.clust__Herbaceous 8
no.clust__Phanerophyte 10

STEP B : Choice of number of clusters

Evolution of clustering evaluation variables with the number of clusters in each group.
All values except that of mVI must be maximized (check function's help for more details about the measures).
Values are highlighted to help finding the number of clusters to keep : the brighter (yellow-ish) the better.

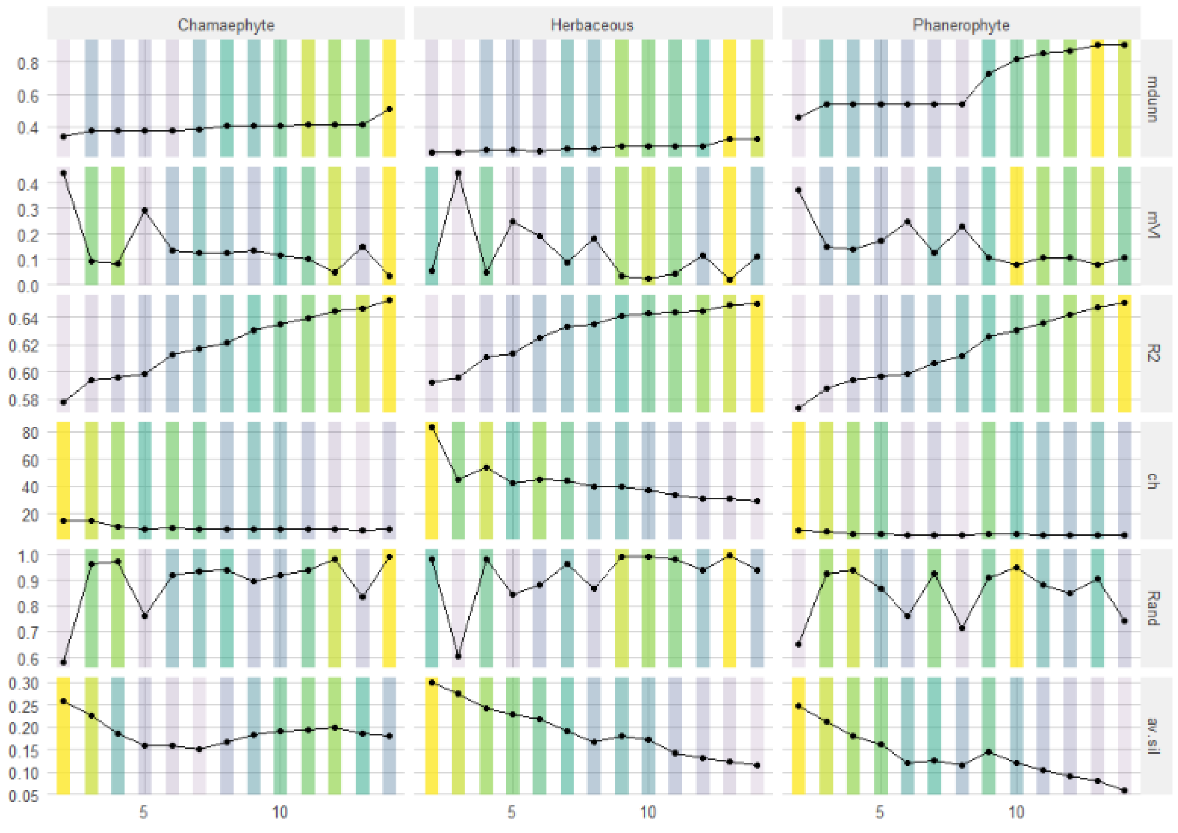
STEP A : Choice of clustering method

Similarity between input and clustering distances (must be minimized, Mouchet et al. 2008) depending on clustering method.



STEP B : Choice of number of clusters

Evolution of clustering evaluation variables with the number of clusters in each group.
All values except that of mVI must be maximized (check function's help for more details about the measures).
Values are highlighted to help finding the number of clusters to keep : the brighter (yellow-ish) the better.



Unfortunately, there are several clustering methods, and several ways of evaluating the quality of the clusters obtained according to the number of groups created. Hence, it is quite difficult to build a method indicating directly by itself the « best » groups that you can get from your matrix of species distances.

The graphic A is purely informative, as you do not have control on this step : the selection of the clustering method (see function's help for details). The graphic B, however, is produced to help you decide about how many groups you want to keep in the end (and this will be done with the PRE_FATE.speciesClustering_step2 function). Indeed, when using a distance matrix to do clustering, it is to be decided « at which level » of the hierarchical tree obtained you want to cut and get the subsequent groups.

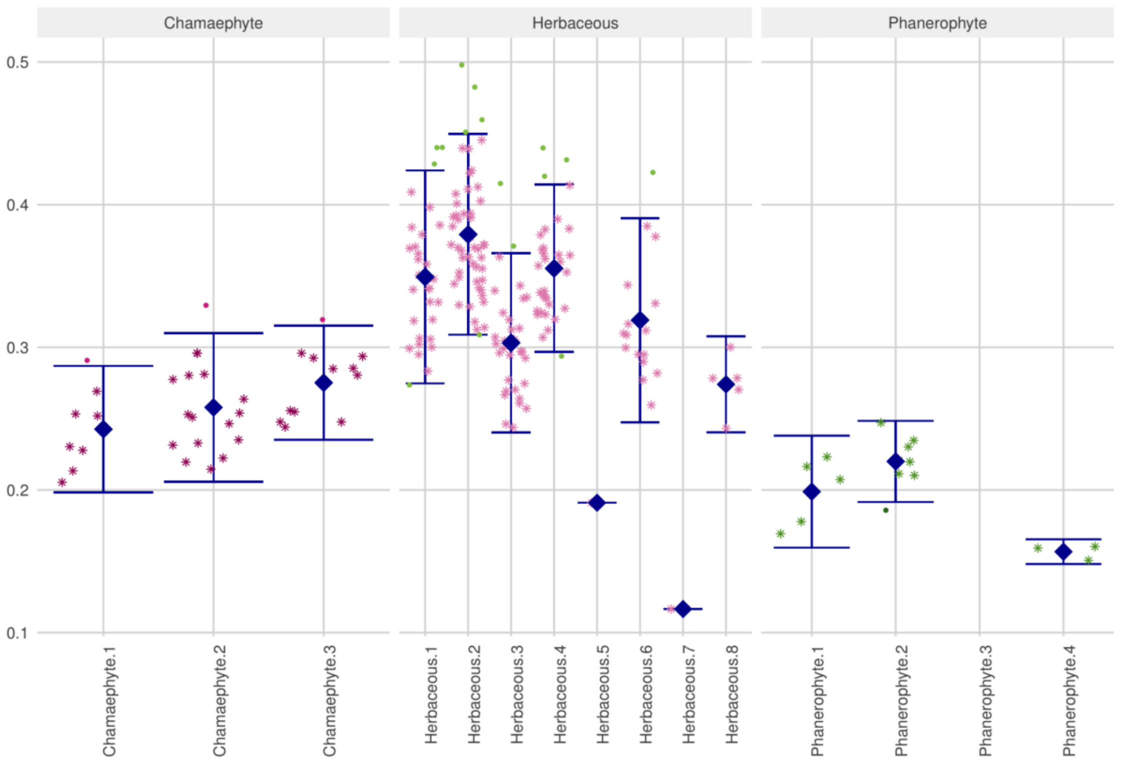
```
## Number of clusters per group
plot(sp.CLUST$plot.clustNo)
no.clusters = c(4, 3, 8) ## Phanerophyte, Chamaephyte, Herbaceous

## Find determinant species --------------------------------------------------
sp.DETERM = PRE_FATE.speciesClustering_step2(
clust.dendrograms = sp.CLUST$ clust.dendrograms
, no.clusters = no.clusters
, mat.species.DIST = tab.dist)

names(sp.DETERM)
str(sp.DETERM$determ.sp)
str(sp.DETERM$determ.all)
```

An additional step of group refinement is also realized with the PRE_FATE.species Clustering_step2 function. Once the hierarchical trees are cut according to the specified number of groups wanted, within-group distances are calculated to try and identified « outliers » species that will be tagged as **« non determinant » species** (see function's help for details).

Principal Coordinate Analysis graphics give insights on how the resulting groups are well differencied from each other according to the final distance matrix combining both traits and overlap informations.

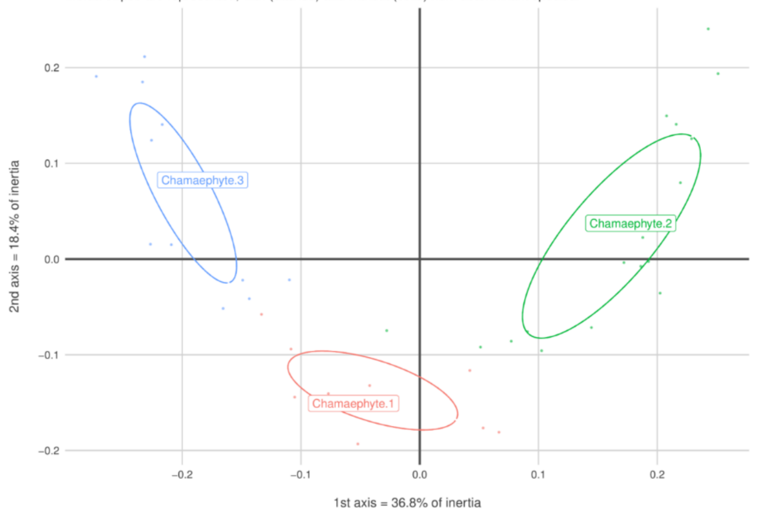
STEP C : Removal of distant species

Only species whose mean distance to other species is included in the distribution
of all PFG's species mean distances to other species are kept.
Species indicated with * will be removed from PFGs.
Non–represented PFG might be one–species–only.



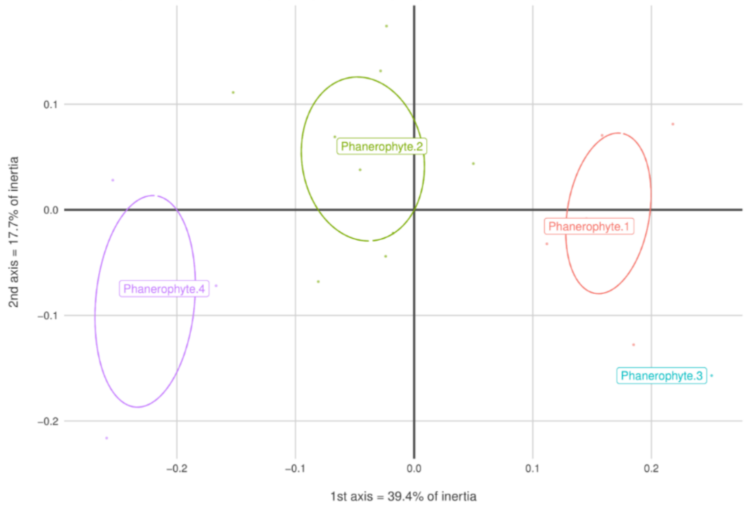
STEP C : Removal of distant species : group Chamaephyte

Only species whose mean distance to other species is included in the distribution
of all PFG's species mean distances to other species are kept.
Species indicated with * will be removed from PFGs.
Inertia ellipse are represented, with (dashed) and without (solid) non–determinant species.



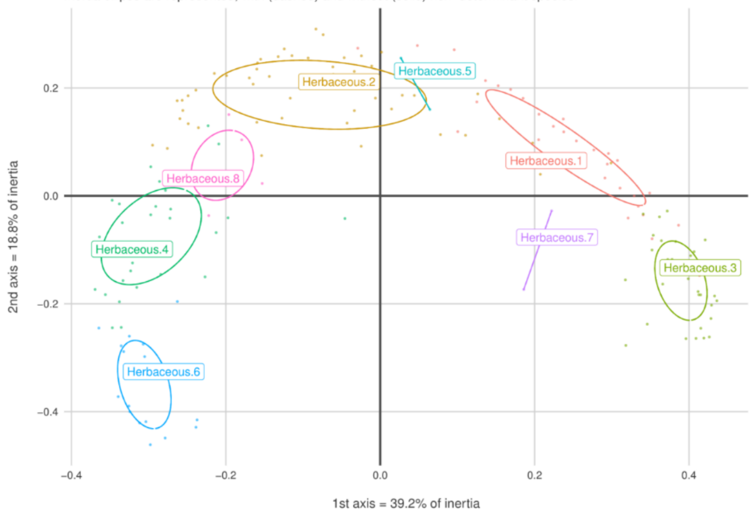
STEP C : Removal of distant species : group Phanerophyte

Only species whose mean distance to other species is included in the distribution
of all PFG's species mean distances to other species are kept.
Species indicated with * will be removed from PFGs.
Inertia ellipse are represented, with (dashed) and without (solid) non–determinant species.



STEP C : Removal of distant species : group Herbaceous

Only species whose mean distance to other species is included in the distribution
of all PFG's species mean distances to other species are kept.
Species indicated with * will be removed from PFGs.
Inertia ellipse are represented, with (dashed) and without (solid) non–determinant species.



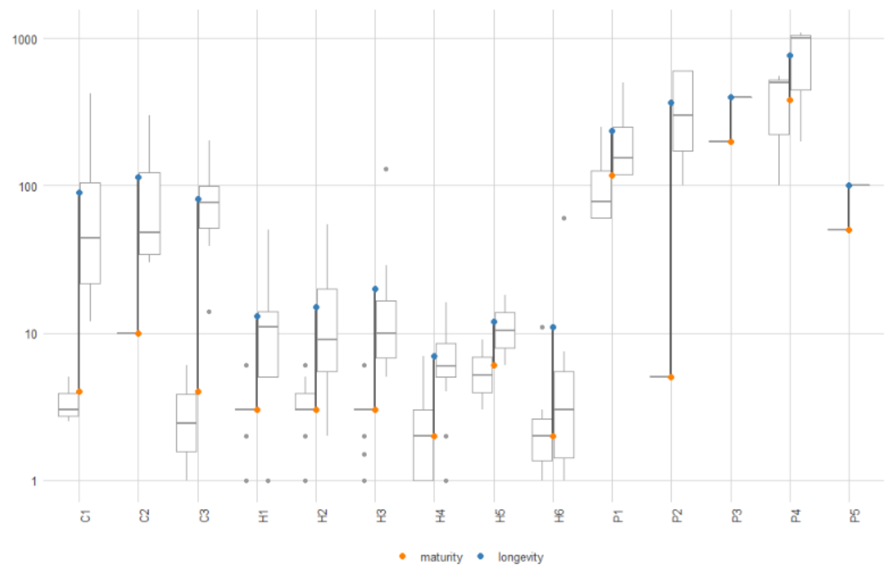
```r
## Load example data
Champsaur_PFG = .loadData("Champsaur_PFG", "RData")

## Species traits
tab.traits = Champsaur_PFG$sp.traits
str(tab.traits)

## Determinant species
tab.PFG = Champsaur_PFG$PFG.species
str(tab.PFG)

## Merge traits and PFG informations
mat.traits = merge(tab.PFG[which(tab.PFG$DETERMINANT==TRUE), c("species","PFG")]
                   , tab.traits
                   , by = "species", all.x = TRUE)
str(mat.traits)

## Keep only traits of interest
mat.traits = mat.traits[, c("PFG", "species", "MATURITY", "LONGEVITY"
                        , "HEIGHT", "LIGHT", "DISPERSAL"
                        , "NITROGEN", "NITROGEN_TOLERANCE", "LDMC", "LNC")]
colnames(mat.traits) = c("PFG", "species", "maturity", "longevity"
                         , "height", "light", "dispersal"
                         , "soil_contrib", "soil_tolerance", "LDMC", "LNC")
mat.traits$soil_contrib = as.numeric(mat.traits$soil_contrib)
mat.traits$soil_tolerance = ifelse(mat.traits$soil_tolerance == 1, 0.5, 1)


## Compute traits per PFG : with one specific graphic ------------------------
PFG.traits = PRE_FATE.speciesClustering_step3(mat.traits = mat.traits)

names(PFG.traits)
str(PFG.traits$tab)
names(PFG.traits$plot)
plot(PFG.traits$plot$maturity_longevity)
plot(PFG.traits$plot$height_light)
plot(PFG.traits$plot$soil)
```

PFG traits values are computed as the mean (for numeric traits) or the median (for categorical traits) of the group's species values. The more missing values, the more biased the group values are.

Graphics can help visualize how different (or not) the groups are in the end. *For example, if all PFG have similar light preferences, then you should not expect to the light interaction module to have a huge impact on your simulated vegetation communities.*

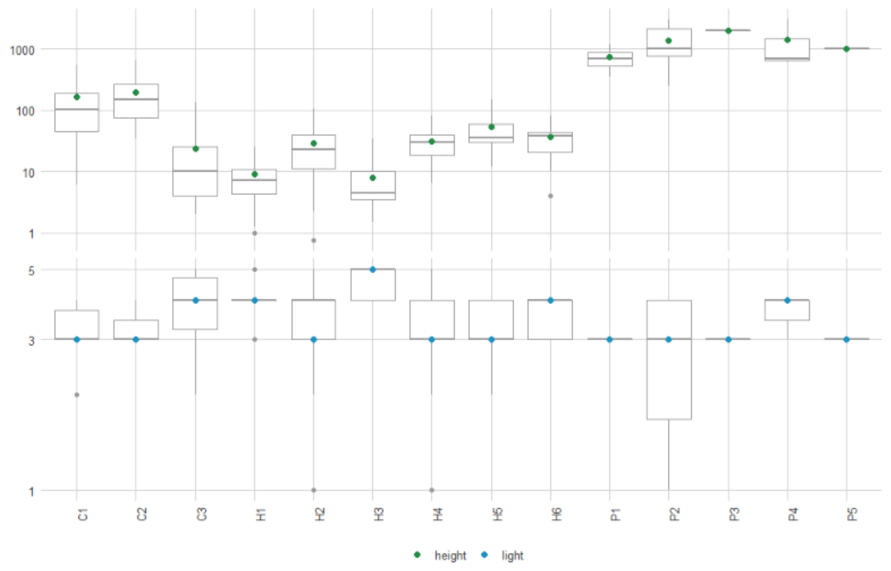
STEP D : Computation of PFG traits values : longevity & maturity

PFG traits values are calculated as the average of the PFG determinant species traits values.
If the trait is factorial or categorical, median value is taken.
Light-grey boxplot represent determinant species values.
Colored points represent the PFG calculated values.



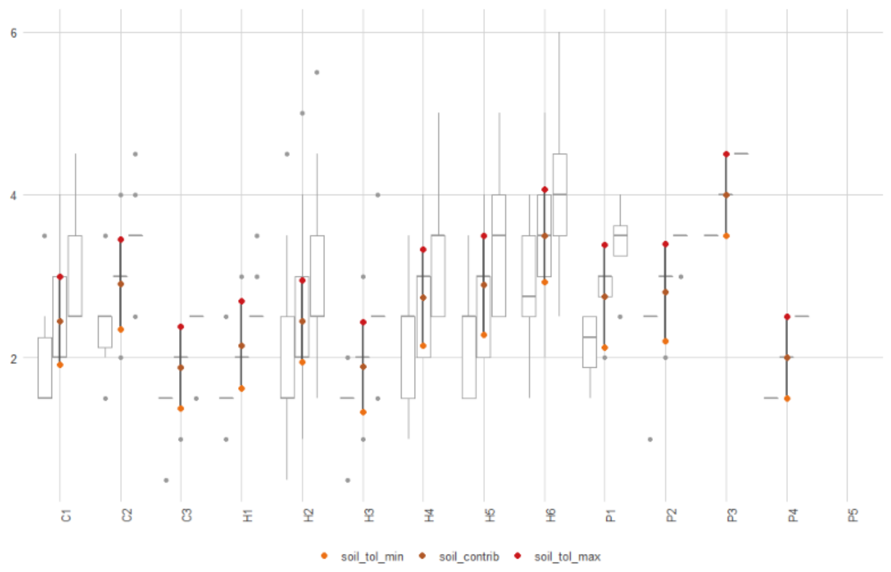
STEP D : Computation of PFG traits values : height & light

PFG traits values are calculated as the average of the PFG determinant species traits values.
If the trait is factorial or categorical, median value is taken.
Light-grey boxplot represent determinant species values.
Colored points represent the PFG calculated values.



STEP D : Computation of PFG traits values : soil contribution & tolerance

PFG traits values are calculated as the average of the PFG determinant species traits values.
If the trait is factorial or categorical, median value is taken.
Light-grey boxplot represent determinant species values.
Colored points represent the PFG calculated values.



3. Modelling framework

a. Parameter files

FATE requires a quite large number of parameters, which are stored into `.txt` files, presented to and recovered by the software. These **parameters** can be of 3 types :

1. **Filenames**, to guide the application to other parameter files that should be read
2. These filenames either correspond to :
    - other parameter files that contain **values** to be actually read and used
    - **raster files**, with the extension `.tif` (lighter) or `.img`

To enumerate these settings, 2 types of flag can be found and used within the parameter files:

1. To give one or several links to other files containing parameter values or to raster files : `--PARAM_NAME—`

```
--GLOBAL_PARAMS--
FATE_simulation/DATA/GLOBAL_PARAMETERS/Global_parameters_V1.txt
--MASK--
FATE_simulation/DATA/MASK/mask.tif
--PFG_PARAMS_LIFE_HISTORY--
FATE_simulation/DATA/PFGS/SUCC/SUCC_PFG1.txt
FATE_simulation/DATA/PFGS/SUCC/SUCC_PFG2.txt
...
```

In this way, each parameter can have several values (filenames), and **each line corresponds to a value**. The transition to a new parameter is made thanks to the presence of a new flag on the next line.

2. To give parameter values : `PARAM_NAME`

```
NAME H2_dryGrass
MATURITY 3
LONGEVITY 11
MAX_ABUNDANCE 1
CHANG_STR_AGES 0 10000 10000 10000 10000
...
```

Each line corresponds to a parameter, given by the **flag** (parameter name in capital letters) **followed by all values linked to this flag on the same line**. Each value has to be separated from another by a **space**.

The FATE friendly directory organization

As FATE needs quite a lot of parameters given into different parameter files, it is important to organize them in a clear and intuitive way. While this is not mandatory to run a FATE simulation, a specific and methodical folder architecture can be obtained with the PRE_FATE.skeletonDirectory function from the RFate package.

A good way to start is to create this skeleton directory, and then copy or create the needed files inside to have all simulation data into the same place.

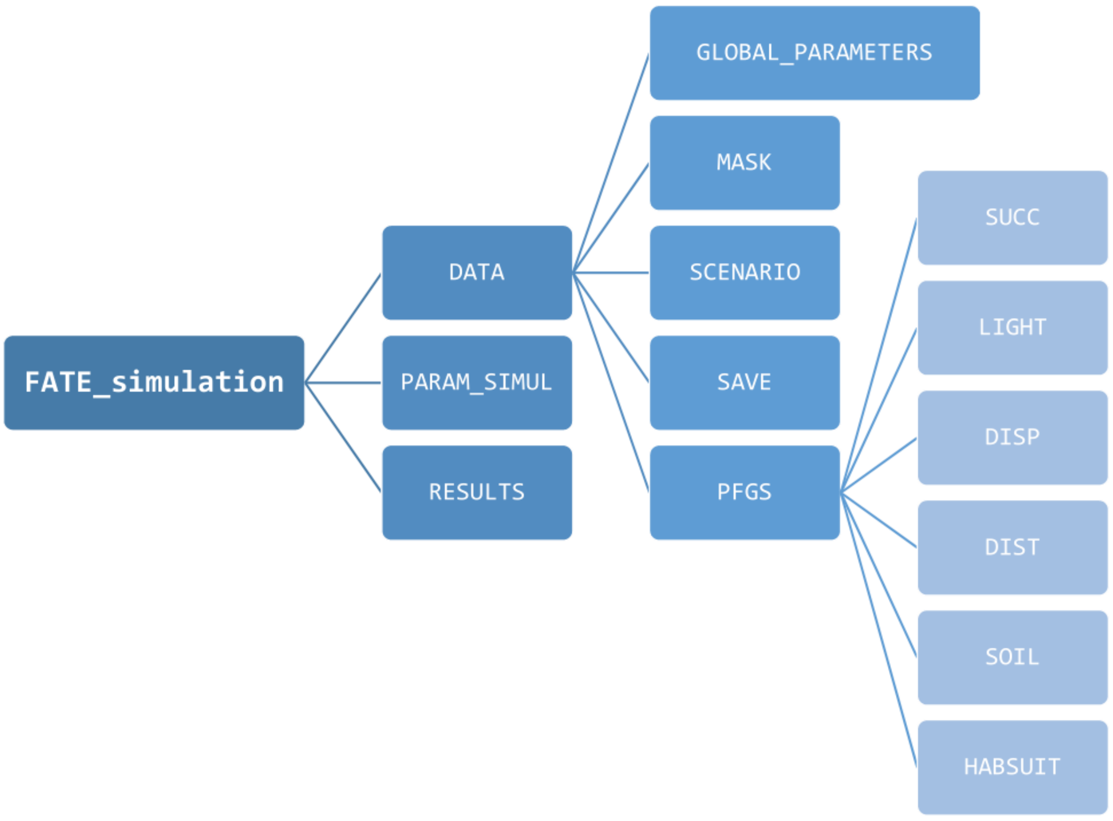


Figure 5: a FATE simulation directory

The FATE modules

FATE model is built like a LEGO tool : it has a **core module**, which corresponds to the fundamental succession, and then can be complemented with **other modules**. Some of these modules can be regarded as equally important as the core module (from an ecological likelihood point of view, like light, dispersal, soil, etc), and some others are more specific (drought, aliens, fire, etc).

All of these modules require

```
mandatory parameters
```

and sometimes, some others that are

```
optional parameters
```

that can all be created with RFate functions.

Which files for which settings ?

The function PRE_FATE.skeletonDirectory allows to create a user-friendly directory tree to store all parameter files and data.

1. *Simulation parameterization*

- **Global parameters :** related to the simulation definition (number of PFG and strata, simulation duration, computer resources, manage abundance values, modules loaded…) with the function PRE_FATE.params_globalParameters
- **Years to save abundance raster maps and simulation outputs** with the function PRE_FATE.params_saveYears
- **Years and files to change rasters** for the succession, habitat suitability or disturbance modules with the function PRE_FATE.params_changingYears

2. *For each PFG : behavior and characteristics*

- **Succession files :** related to the life history with the function PRE_FATE.params_PFGsuccession
- **Dispersal files :** related to the dispersal ability with the function PRE_FATE.params_PFGdispersal
- **Light files :** related to the light interaction with the function PRE_FATE.params_PFGlight
- **Soil files :** related to the soil interaction with the function PRE_FATE.params_PFGsoil
- **Disturbance files :** related to the response to perturbations in terms of resprouting and mortality with the function PRE_FATE.params_PFGdisturbance
- **Drought files :** related to the response to drought in terms of resprouting and mortality with the function PRE_FATE.params_PFGdrought

3. *Parameter management*

- **SimulParameters file :** containing all links to the files created with the previous functions. This is the file that will be given as the only argument to the FATE function. It can be created with the function PRE_FATE.params_simulParameters
- **Multiple set of files :** to duplicate simulation folder and scan global parameters space with Latin Hypercube Sampling.
  Folders can be created with the function PRE_FATE.params_multipleSet

a. Core module (succession)

« *Based on the 'FATE' model (Moore & Noble, 1990), it describes the within-pixel succession dynamics in an annual time step. […] Five processes describe PFG demography (germination, recruitment, growth, survival and fecundity, see Table 1).* » (Boulangeat, 2014)

i. Conceptual diagram

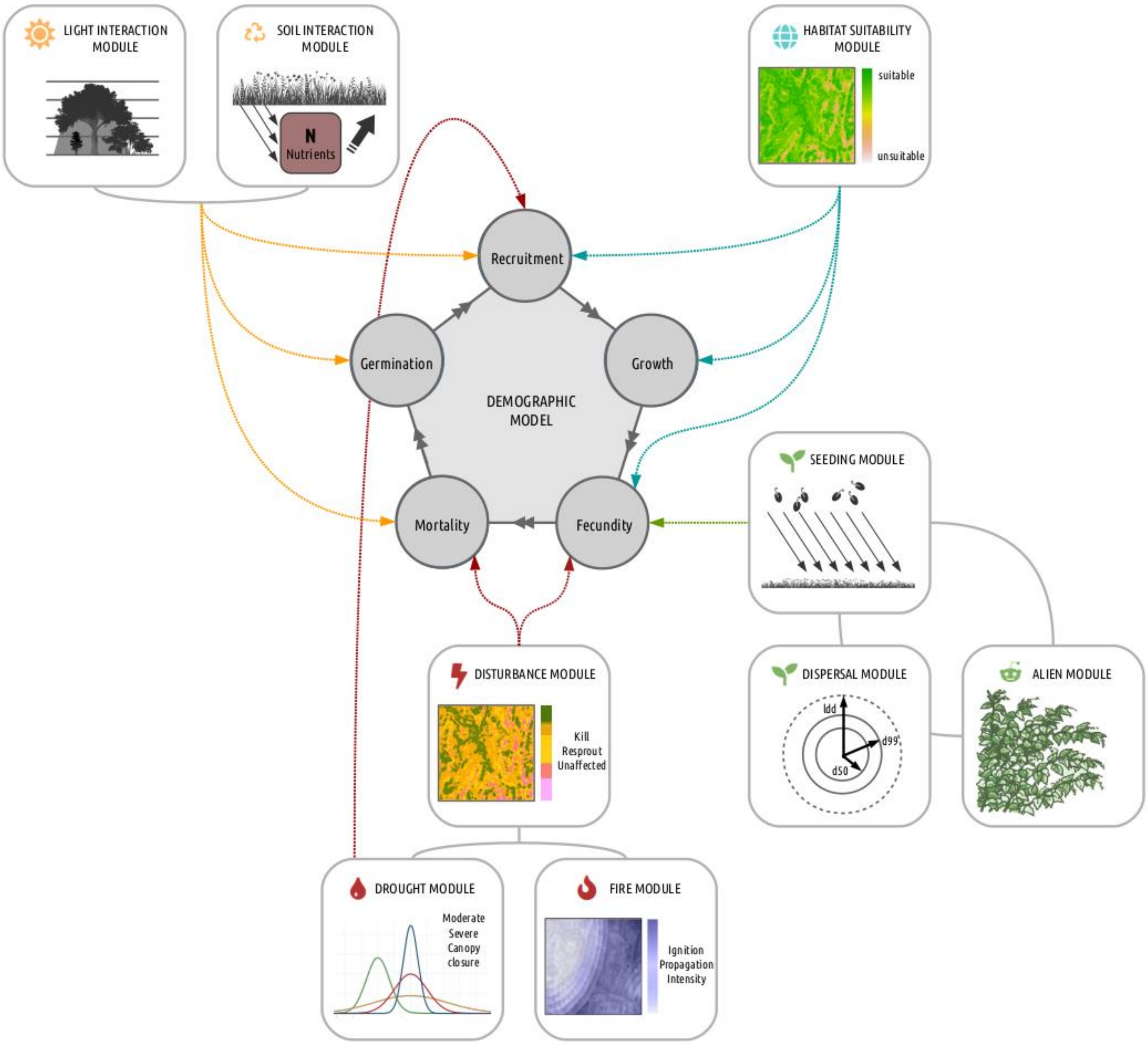


Figure 6: FATE conceptual diagram

The whole area is divided in grid-cells in which an independent **demographic model** regulates the PFG life cycle. PFG abundances are structured by age into cohorts and each cohort is attributed to a height stratum according to the growth parameters.

Different submodels affect this cycle at various levels (Fig6) :

- ⌘ **Interaction through light and/or soil resources** regulates interactions between cohorts affecting germination, recruitment and survival.
- ⌘ **Habitat suitability** affects the recruitment and fecundity rates.
- ⌘ The **seed dispersal** model makes FATE spatially explicit by connecting grid-cells. It depends on the amount of seeds produced by mature plants and affects each PFG's seed bank in each cell.
- ⌘ **Disturbances** affect PFG survival and fecundity.

ii. Life cycle of each PFG and influences from sub-models

Only three age classes are considered : **germinant**, **juvenile** and **mature**, and can be impacted by the different sub-models activated during the simulation (Fig7) :

- ⅄ The **recruitment** is influenced by the habitat suitability and the biotic interactions.
- ⅄ **Mortality** occurs when light or soil conditions are not favorable or when the PFG completes its life span.
- ⅄ In addition, the disturbance regime directly affects juvenile or mature PFG and may for instance result in PFG death, impede **seed production** by reducing mature PFG age to N-1, or **revitalize senescents** by reducing their age to M-1.

The **timestep** is at the **year** level : seasonality is not included within each timestep, but communities go through 4 states (Fig8) :

1. **Check of survival** : what are the pixel resources (in terms of light and soil) of the previous year, and can the PFG stand them ?
2. All PFG **grow one-year older**, and too old PFG die.
3. **New pixel resources** (in terms of light and soil) are calculated with the actual community, as well as the **seeds produced**. **Recruitment** of new individuals from the previous pool of seeds occurs.
4. If some perturbations are defined, community is impacted in function of the **PFG responses to the disturbance(s)**.

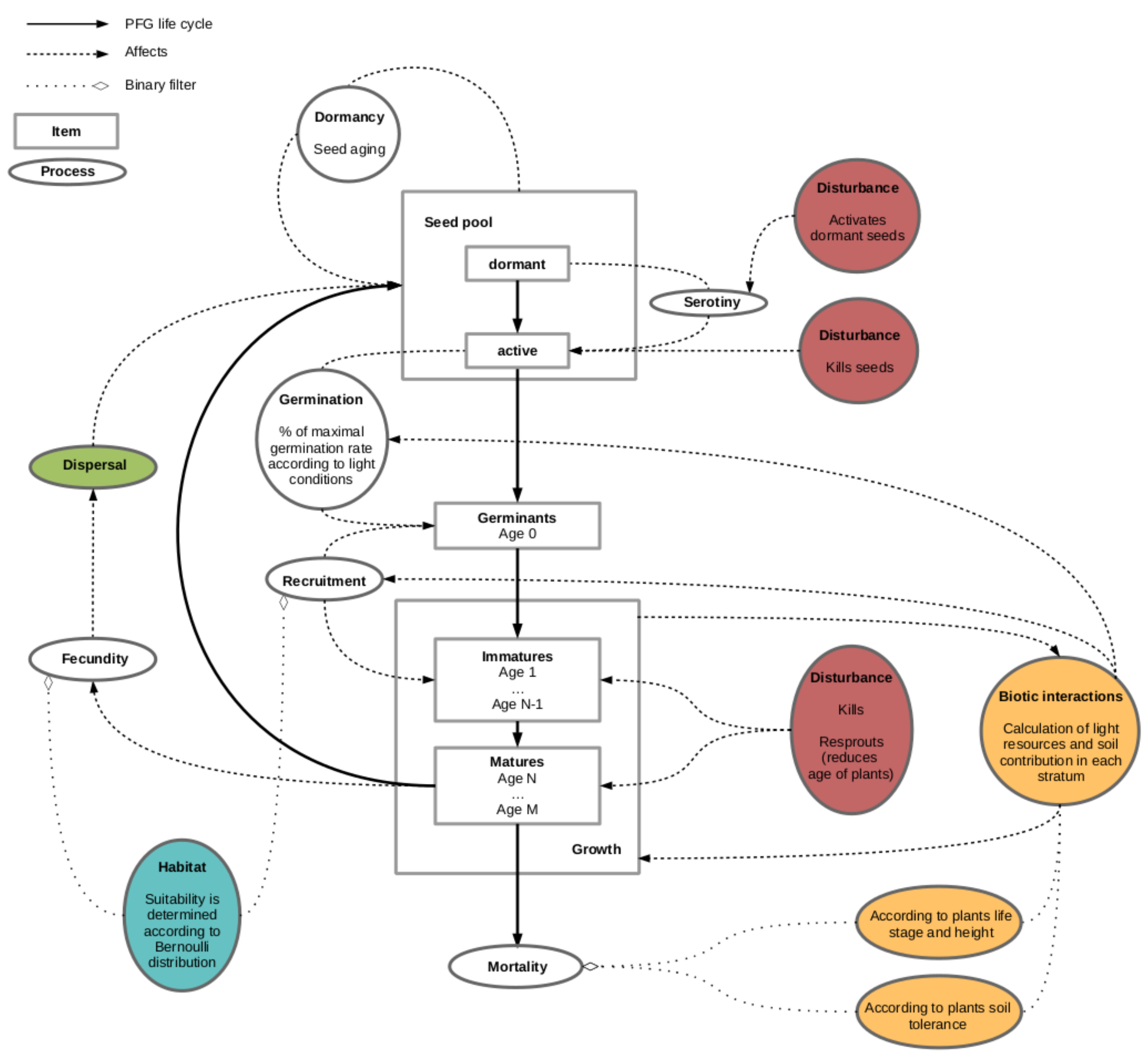


Figure 7: FATE life cycle

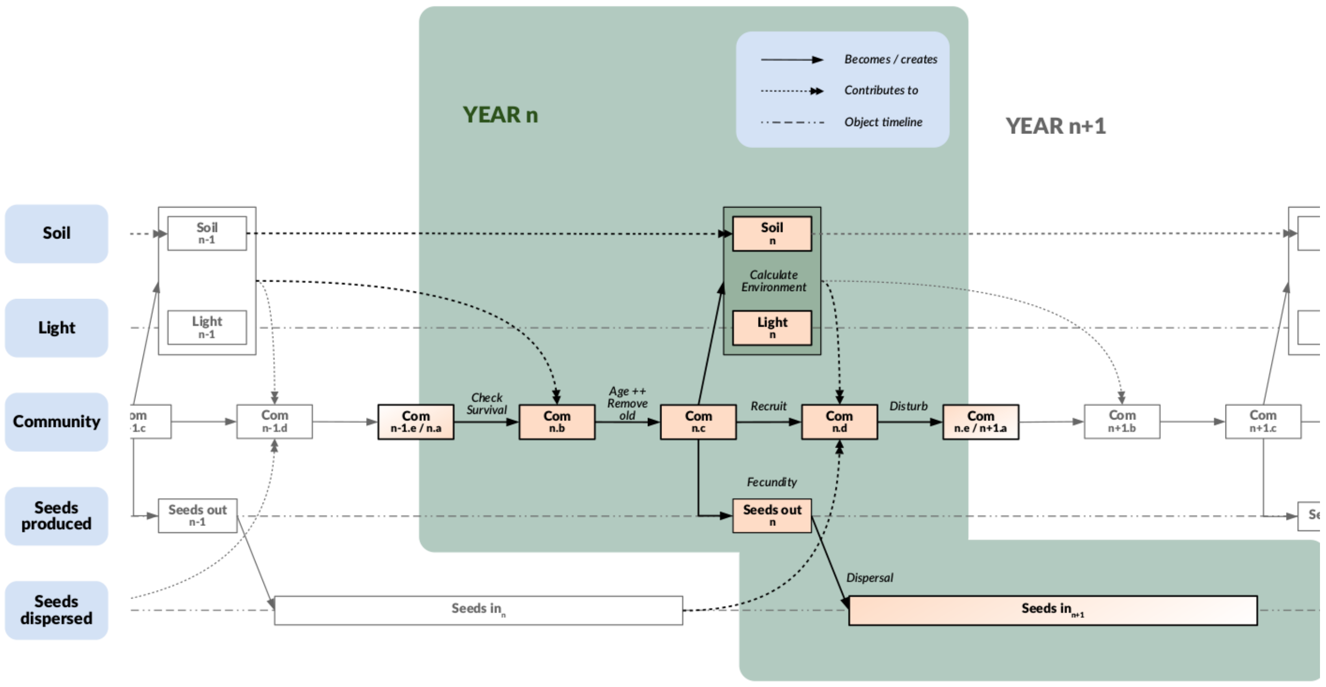


Figure 8: FATE timeline diagram

iii. Calculation of pixel resources

Light and soil resources are **proportional to the abundance of the PFG community** of the pixel. They are both converted into **qualitative classes** (Low, Medium or High), for each **height stratum** (concerning light) and for each **PFG** (according to its tolerance, regarding soil). The response of each PFG to each resource level is defined **in function of age** (Germinant, Immature and Mature), in a semi-binary way for light (Fig9), and in a more quantitative way for soil (Fig10).

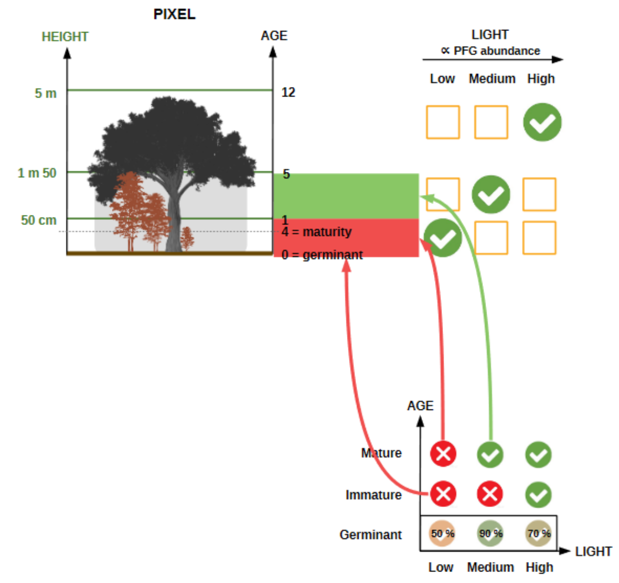


Figure 9: Calculation of FATE resources (light)

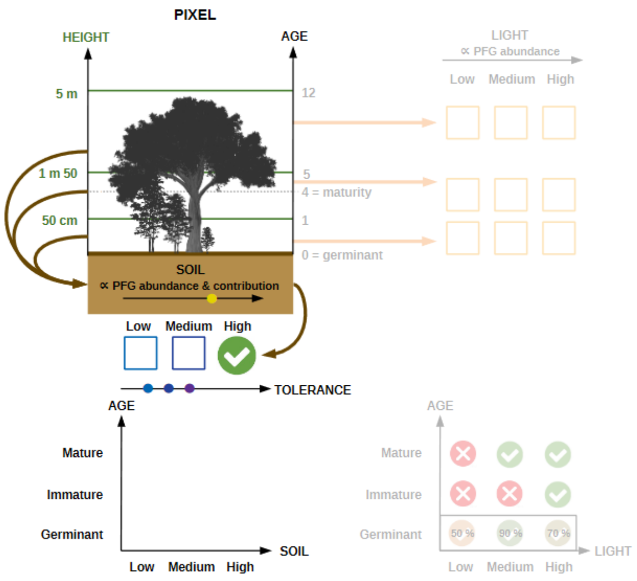


Figure 10: Calculation of FATE resources (soil)

iv. Structural equations

Internal processes within FATE are presented on Fig11. Its full understanding by the user is not required or mandatory for the proper use of the software. However, it can help understand the few structural equations integrated in FATE and bring a better appreciation of the overall functioning. Key parameters and functions are progressively presented below in order to appreciate the structural equations.

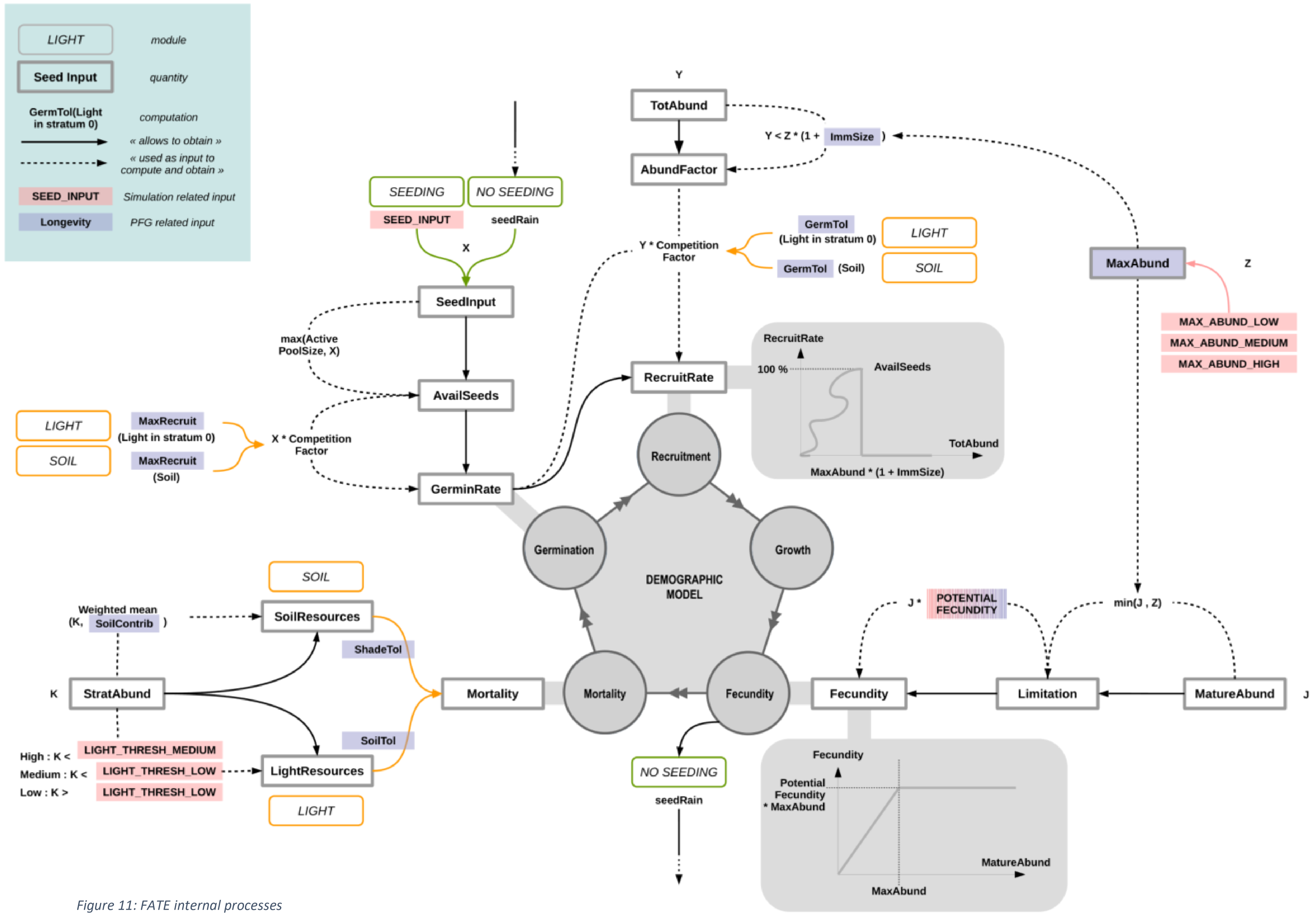


Figure 11: FATE internal processes

Influence of environment (habitat suitability)

- **$getEnv$ ... () functions :**
  All the $getEnv...()$ functions represent the influence of the habitat suitability if the module is selected (see `DO_HABITAT_SUITABILITY` parameter in GlobalParameter file).
  They can have effect on different processes, such as mortality, recruitment or fecundity, depending on whether the habitat within the pixel is suitable or not for the considered PFG.

- **Is the habitat suitable ?**
  Each year (timestep), the values contained in each PFG habitat suitability maps will be compared to a reference value :
  - if superior, the environment is considered suitable for the PFG (hence $getEnv...()$ functions will return 1)
  - otherwise, the environment is considered unsuitable for the PFG (hence $getEnv...()$ functions will return 0)

Depending on the parameterisation chosen, (see `HABSUIT_OPTION` parameter in GlobalParameter file), the reference value can be set in two different ways.

Lifespan & maturity :

- **Lifespan :**
  In theory, the lifetime of a species could be influenced by the environment, but this is currently not the case. Hence, habitat or not, $getEnvMort() = 1$.

$$LifeSpan * getEnvMort()$$

- **Maturity time :**
  The time from which a PFG is able to produce seeds can also be influenced by its habitat, in a negative way :

$$maturityTime = (LifeSpan - Maturity) * (1.0 - getEnvGrowth()) + Maturity$$

$$maturityTime = ceil(maturityTime)$$

If `DO_HABITAT_SUITABILITY` model is NOT selected, or the habitat is suitable, then $maturityTime = Maturity$. Otherwise, it the habitat is NOT suitable, $maturityTime = LifeSpan$, which means there will be no fecundity, and then no seeds produced.

Carrying capacity (mature vs immature) :

- ⌘ **Immature :**
  Depending on the PFG life-form (herbaceous, chamaephytes, phanerophytes), immature individuals may no take as much space as mature individuals (e.g. young tree vs old tree). Hence, when calculating total abundance of plants, which is used as a proxy of space occupation, abundance of immature individuals is weighted by their relative size compare to mature individuals : $ImmSize$ (see `IMM_SIZE` parameter in Succession files).

- ⌘ **Mature - Global carrying capacity :**
  $MaxAbund \in 1,2,3$ defines the maximum abundance that can be reached by a mature PFG. It should be inversely proportional to the space that the PFG can occupy, with taller PFG generally having fewer individuals than for example herbaceous within the same space (see `MAX_ABUNDANCE` parameter in Succession files). It is converted to abundance-related values when contributing to structural equations (see `MAX_ABUND_{...}` parameters in GlobalParameter file).

Germination :

- ⅄ **Condition on carrying capacity :**
  A condition is set to help regulate populations : new individuals only grow if there is not yet too many individuals within the pixel, i.e. if the total abundance of the PFG does not exceed its global carrying capacity :

$$totAbund = MatureAbund + ImmatureAbund * ImmSize$$

$$globalCC = MaxAbund + MaxAbund * ImmSize$$

- ⅄ **Condition on pixel resources :**
  $MaxRecruit$ corresponds to percentage of seeds that will germinate depending on the pixel resources (light, soil) in stratum 0 (which represents the enforced dormancy) (see `ACTIVE_GERM` parameter in Light and Soil files). The number of germinating seeds is obtained by weighting the number of available seeds by this germination rate. If the module is selected (see `DO_HABITAT_SUITABILITY` parameter in Global Parameter file), the habitat must be suitable, otherwise the recruitment will be null.

$$RecruitmentRate = GerminationRate * getEnvRecrRate()$$

$$= AvailSeeds * MaxRecruitment * getEnvRecrRate()$$

Fecundity :

⌘ **Potential fecundity :**
Each PFG can produce a fixed amount of seeds per individual (see `POTENTIAL_FECUNDITY` parameter in Succession files). Due to lack of empirical data, this amount is often set at the same value for all PFG.

⌘ **Produced seeds :**
At each time step, the number of seeds that will be produced by a PFG depends both on the number of mature individuals of this PFG within the considered pixel, and on the suitability of the pixel if the module is selected (see `DO_HABITAT_SUITABILITY` parameter in GlobalParameter file) (no seeds produced if the habitat is not suitable) :

$$Fecundity = min(MatureAbund, MaxAbund) * PotentialFecund * getEnvFecund()$$

If $MatureAbund \geqslant MaxAbund$, the PFG has reached its annual carrying capacity : it is in optimal conditions and will produce its maximum amount of seeds ($MaxAbund * PotentialFecund$). Otherwise, this amount will be reduced in proportion.

ANNUAL SEED CYCLE : (*combining all previous information*)

Germination occurs depending on the current abundance of the PFG inside the pixel : if it reaches the carrying capacity of the PFG $= MaxAbund * (1 + ImmSize)$, no seed germinates. The number of produced seeds is proportional to the current abundance of mature individuals only.

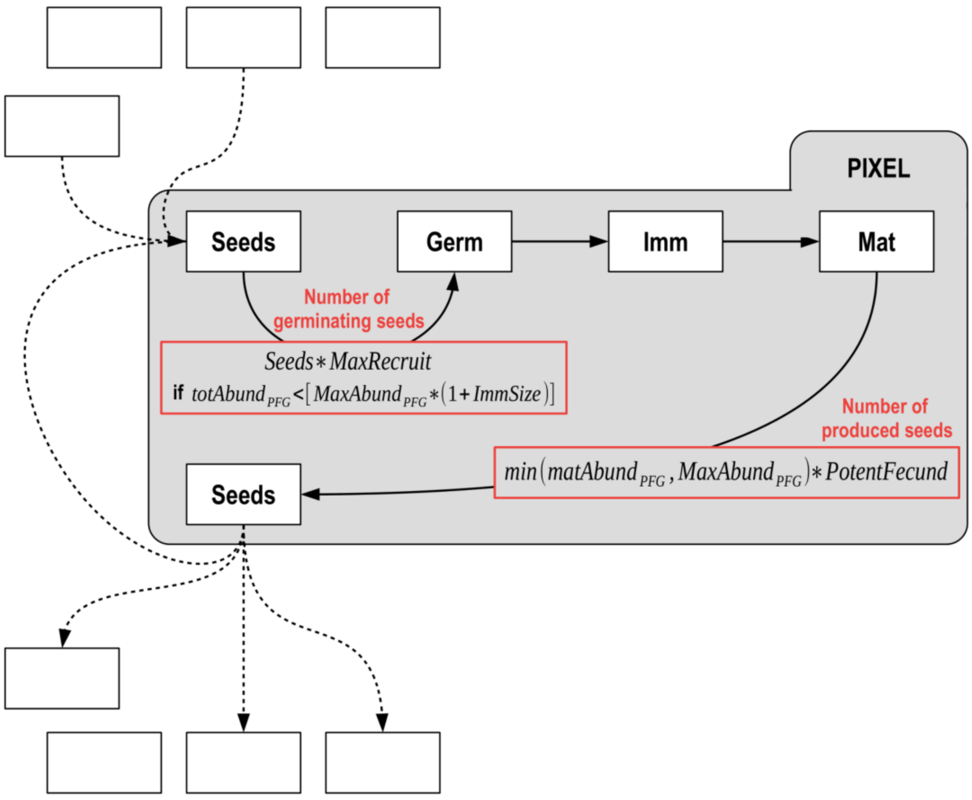

PIXEL
Seeds
Germ
Imm
Mat
Number of germinating seeds
$Seeds * MaxRecruit$
if $totAbund_{PFG} < [MaxAbund_{PFG} * (1 + ImmSize)]$
Number of produced seeds
$min(matAbund_{PFG}, MaxAbund_{PFG}) * PotentFecund$
Seeds

Figure 12: FATE annual seed cycle

Required parameters :

- In GlobalParameters file :

```
NO_PFG 16
NO_STRATA 6
SIMULATION_DURATION 950
SEEDING_DURATION 300
SEEDING_TIMESTEP 1
SEEDING_INPUT 100
POTENTIAL_FECUNDITY 10
MAX_ABUND_LOW 1000
MAX_ABUND_MEDIUM 5000
MAX_ABUND_HIGH 8000
```

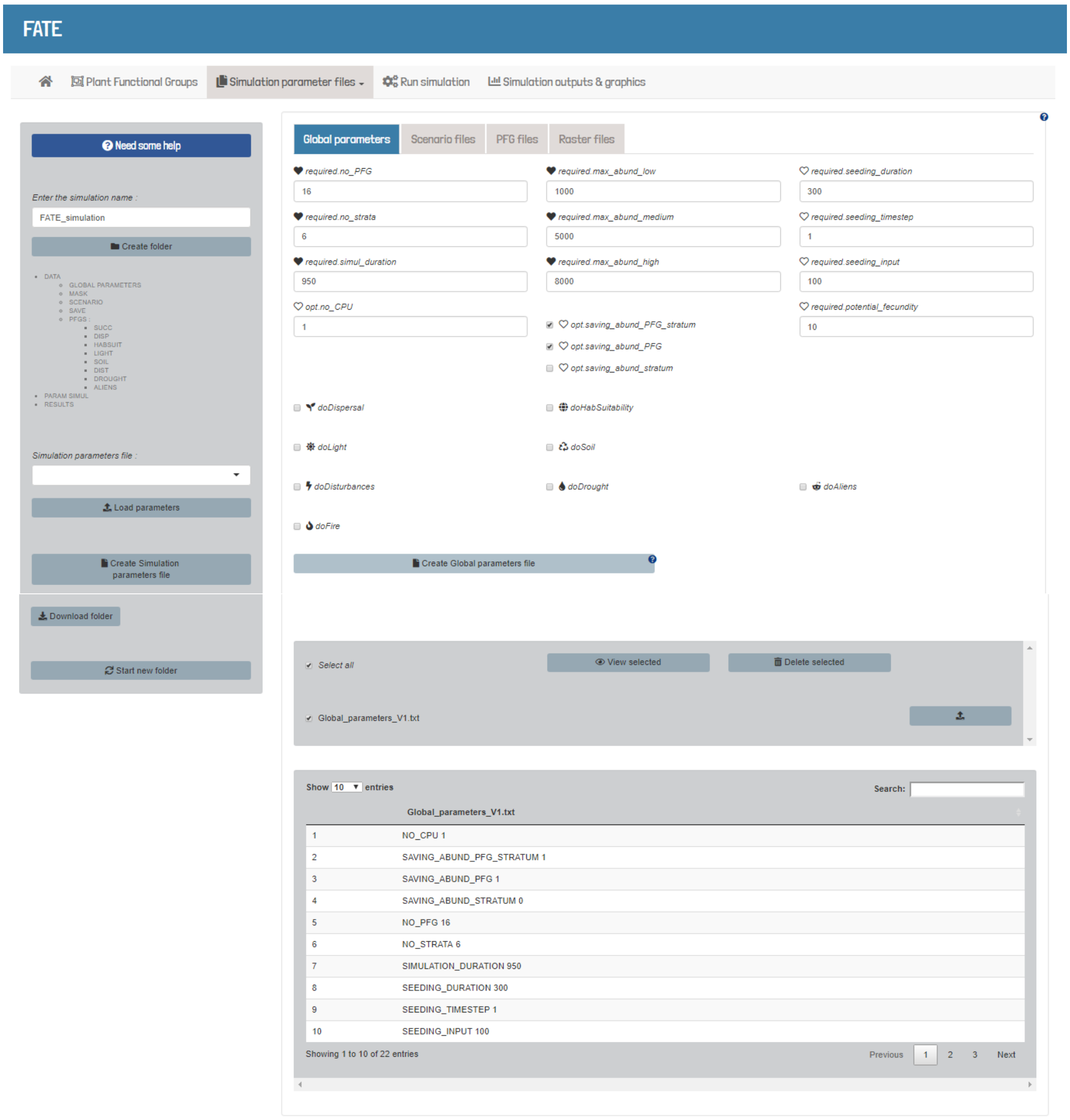
The number of computer resources can also be given. If so, some parts of the main loop of the code will be parallelized on the amount of indicated resources.

```
NO_CPU 6
```

Finally, several parameters are available to select which outputs should be saved on when running the simulation :

```
SAVING_ABUND_PFG_STRATUM 1
SAVING_ABUND_PFG 1
SAVING_ABUND_STRATUM 0
```

```
## Create a skeleton folder with the default name ('FATE_simulation') --------
PRE_FATE.skeletonDirectory()

## Create a Global_parameters file--------------------------------------------
PRE_FATE.params_globalParameters(name.simulation = "FATE_simulation"
                                 , opt.saving_abund_PFG_stratum = TRUE
                                 , opt.saving_abund_PFG = TRUE
                                 , opt.saving_abund_stratum = FALSE
                                 , required.no_PFG = 16
                                 , required.no_strata = 6
                                 , required.simul_duration = 950
                                 , required.seeding_duration = 300
                                 , required.seeding_timestep = 1
                                 , required.seeding_input = 100
                                 , required.potential_fecundity = 10
                                 , required.max_abund_low = 1000
                                 , required.max_abund_medium = 5000
                                 , required.max_abund_high = 8000)
```



FATE

Plant Functional Groups | Simulation parameter files | Run simulation | Simulation outputs & graphics

Need some help

Enter the simulation name :

FATE_simulation

Create folder

- DATA
  - GLOBAL PARAMETERS
  - MASK
  - SCENARIO
  - SAVE
  - PFGS :
    - SUCC
    - DISP
    - HABSUIT
    - LIGHT
    - SOIL
    - DIST
    - DROUGHT
    - ALIENS
- PARAM SIMUL
- RESULTS

Simulation parameters file :

Load parameters

Create Simulation parameters file

Download folder

Start new folder

Global parameters | Scenario files | PFG files | Raster files

required.no_PFG: 16	*required.max_abund_low*: 1000	*required.seeding_duration*: 300
required.no_strata: 6	*required.max_abund_medium*: 5000	*required.seeding_timestep*: 1
required.simul_duration: 950	*required.max_abund_high*: 8000	*required.seeding_input*: 100
opt.no_CPU: 1	[✓] *opt.saving_abund_PFG_stratum* [✓] *opt.saving_abund_PFG* [ ] *opt.saving_abund_stratum*	*required.potential_fecundity*: 10

[ ] *doDispersal* [ ] *doHabSuitability*

[ ] *doLight* [ ] *doSoil*

[ ] *doDisturbances* [ ] *doDrought* [ ] *doAliens*

[ ] *doFire*

Create Global parameters file

[✓] *Select all* View selected Delete selected

[✓] Global_parameters_V1.txt

Show 10 entries Search:

	Global_parameters_V1.txt
1	NO_CPU 1
2	SAVING_ABUND_PFG_STRATUM 1
3	SAVING_ABUND_PFG 1
4	SAVING_ABUND_STRATUM 0
5	NO_PFG 16
6	NO_STRATA 6
7	SIMULATION_DURATION 950
8	SEEDING_DURATION 300
9	SEEDING_TIMESTEP 1
10	SEEDING_INPUT 100

Showing 1 to 10 of 22 entries Previous 1 2 3 Next

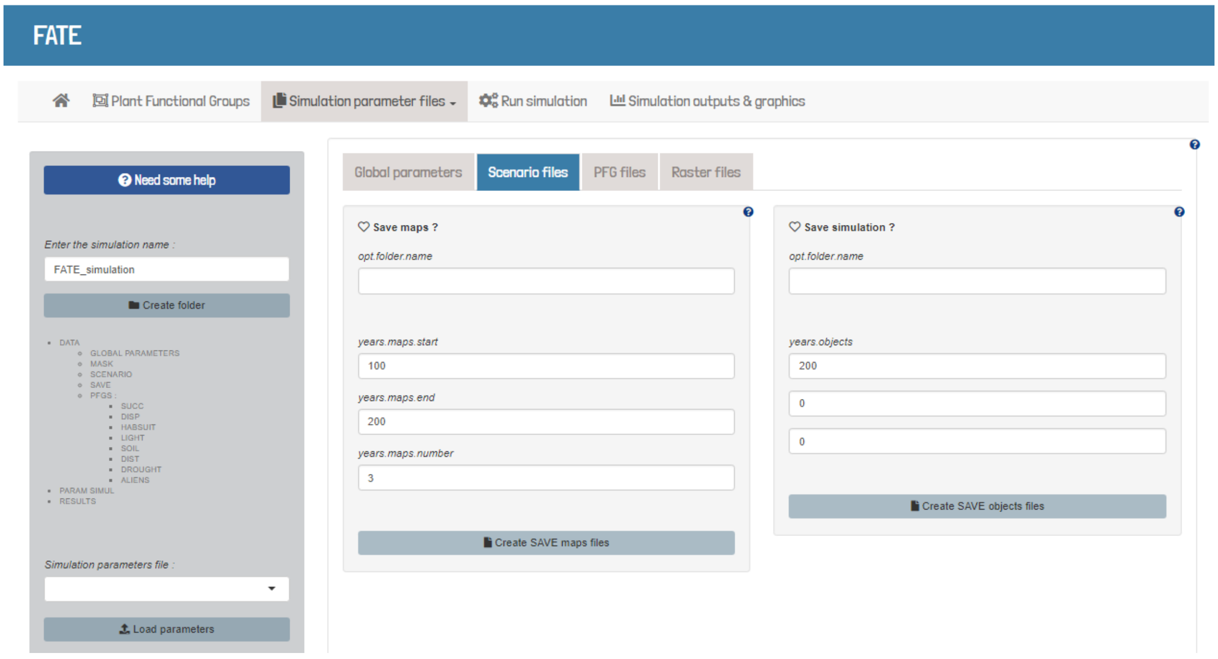
➢ In SimulParameters file :

```
--GLOBAL_PARAMS--
FATE_simulation/DATA/GLOBAL_PARAMETERS/Global_parameters_V1.txt
--SAVING_DIR--
FATE_simulation/RESULTS/SIMUL_1/
--MASK--
FATE_simulation/DATA/MASK/mask.tif
--PFG_PARAMS_LIFE_HISTORY--
FATE_simulation/DATA/PFGS/SUCC/SUCC_PFG1.txt
FATE_simulation/DATA/PFGS/SUCC/SUCC_PFG2.txt
--END_OF_FILE--
```

The years for which must be saved abundance maps for each PFG can also be indicated, as well as years to save a copy of the simulation object :

```
--SAVING_YEARS_ARRAYS--
FATE_simulation/DATA/SAVE/SAVE_YEARS_maps.txt
--SAVING_YEARS_OBJECTS--
FATE_simulation/DATA/SAVE/SAVE_YEARS_objects.txt
```

```
## Create a skeleton folder with the default name ('FATE_simulation') --------
PRE_FATE.skeletonDirectory()

## Create a SAVE_year_maps or/and SAVE_year_objects parameter file -----------
PRE_FATE.params_savingYears(name.simulation = 'FATE_simulation'
                            , years.maps = c(100, 150, 200)
                            , years.objects = 200)
```



Enter the simulation name : FATE_simulation

- DATA
  - GLOBAL PARAMETERS
  - MASK
  - SCENARIO
  - SAVE
  - PFGS :
    - SUCC
    - DISP
    - HABSUIT
    - LIGHT
    - SOIL
    - DIST
    - DROUGHT
    - ALIENS
- PARAM SIMUL
- RESULTS

Simulation parameters file :

Save maps ?
opt.folder.name
years.maps.start: 100
years.maps.end: 200
years.maps.number: 3

Save simulation ?
opt.folder.name
years.objects: 200
0
0

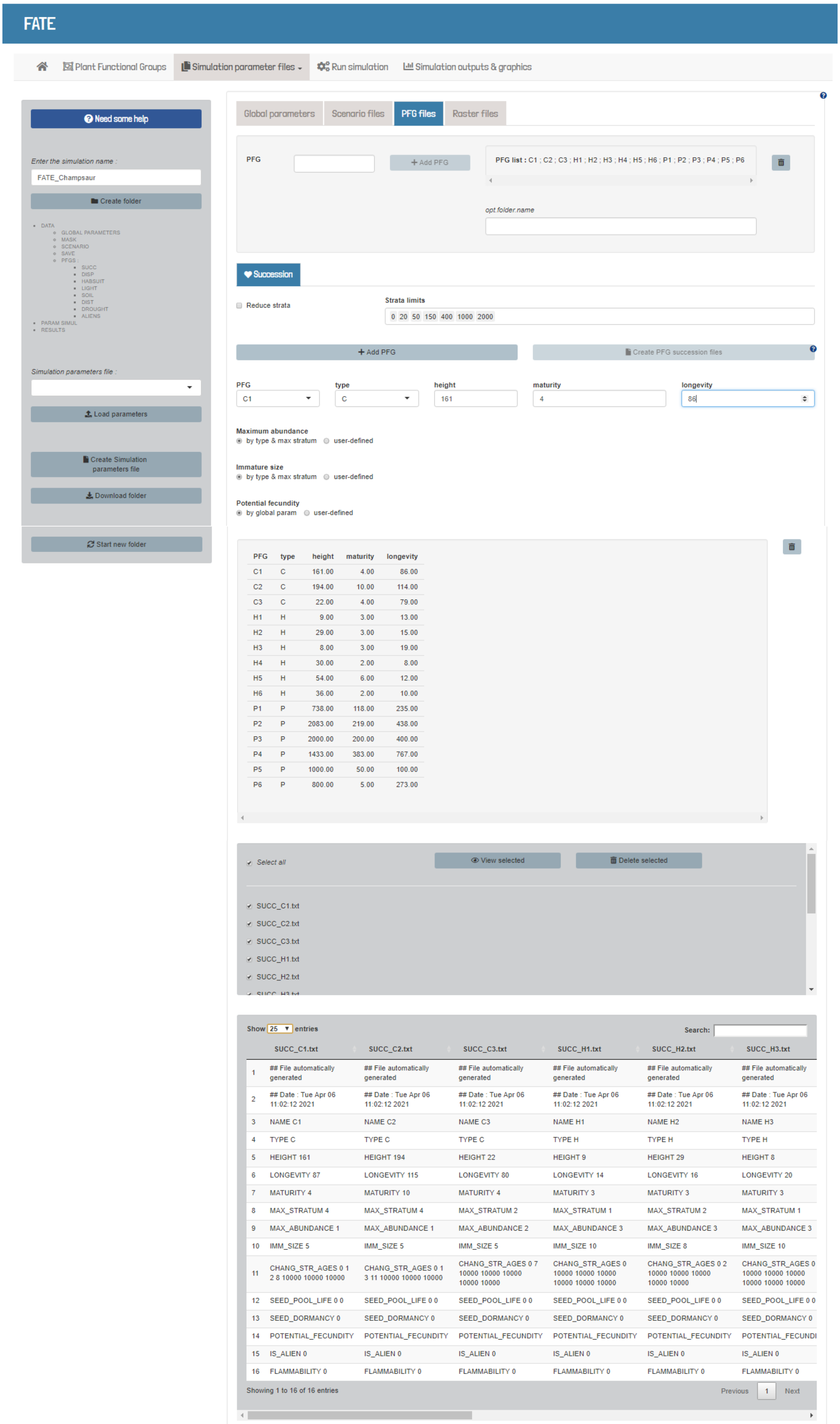
➢ In SUCCESSION files (given with the `--PFG_PARAMS_LIFE_HISTORY--` flag in *SimulParameters* file) :

```
NAME PFG1
TYPE P
HEIGHT 1200
MATURITY 45
LONGEVITY 451
MAX_STRATUM 5
MAX_ABUNDANCE 3
IMM_SIZE 1
CHANG_STR_AGES 0 14 38 110 344
SEED_POOL_LIFE 0 0
SEED_DORMANCY 0
```

The maximum number of seeds produced each year can also be specified per PFG :

```
POTENTIAL_FECUNDITY 50
```

```
## Load example data
Champsaur_params = .loadData("Champsaur_params", "RData")

## Create a skeleton folder
PRE_FATE.skeletonDirectory(name.simulation = 'FATE_Champsaur')



## PFG traits for succession
tab.succ = Champsaur_params$tab.SUCC
str(tab.succ)

## Create PFG succession parameter files -------------------------------------
PRE_FATE.params_PFGsuccession(name.simulation = 'FATE_Champsaur'
                              , mat.PFG.succ = tab.succ)

## Create PFG succession parameter files (fixing strata limits) --------------
PRE_FATE.params_PFGsuccession(name.simulation = 'FATE_Champsaur'
                              , mat.PFG.succ = tab.succ
                              , strata.limits = c(0, 20, 50, 150, 400, 1000, 2000)
                              , strata.limits_reduce = FALSE)
```



Enter the simulation name : FATE_Champsaur

Simulation parameters file :

PFG

PFG list : C1 ; C2 ; C3 ; H1 ; H2 ; H3 ; H4 ; H5 ; H6 ; P1 ; P2 ; P3 ; P4 ; P5 ; P6

opt.folder.name

Reduce strata

Strata limits: 0 20 50 150 400 1000 2000

PFG: C1 — type: C — height: 161 — maturity: 4 — longevity: 86

Maximum abundance
[✓ by type & max stratum] user-defined

Immature size
[✓ by type & max stratum] user-defined

Potential fecundity
[✓ by global param] user-defined

PFG	type	height	maturity	longevity
C1	C	161.00	4.00	86.00
C2	C	194.00	10.00	114.00
C3	C	22.00	4.00	79.00
H1	H	9.00	3.00	13.00
H2	H	29.00	3.00	15.00
H3	H	8.00	3.00	19.00
H4	H	30.00	2.00	8.00
H5	H	54.00	6.00	12.00
H6	H	36.00	2.00	10.00
P1	P	738.00	118.00	235.00
P2	P	2083.00	219.00	438.00
P3	P	2000.00	200.00	400.00
P4	P	1433.00	383.00	767.00
P5	P	1000.00	50.00	100.00
P6	P	800.00	5.00	273.00

✓ Select all

✓ SUCC_C1.txt
✓ SUCC_C2.txt
✓ SUCC_C3.txt
✓ SUCC_H1.txt
✓ SUCC_H2.txt
✓ SUCC_H3.txt

Show 25 entries — Search:

	SUCC_C1.txt	SUCC_C2.txt	SUCC_C3.txt	SUCC_H1.txt	SUCC_H2.txt	SUCC_H3.txt
1	## File automatically generated	## File automatically generated	## File automatically generated	## File automatically generated	## File automatically generated	## File automatically generated
2	## Date : Tue Apr 06 11:02:12 2021	## Date : Tue Apr 06 11:02:12 2021	## Date : Tue Apr 06 11:02:12 2021	## Date : Tue Apr 06 11:02:12 2021	## Date : Tue Apr 06 11:02:12 2021	## Date : Tue Apr 06 11:02:12 2021
3	NAME C1	NAME C2	NAME C3	NAME H1	NAME H2	NAME H3
4	TYPE C	TYPE C	TYPE C	TYPE H	TYPE H	TYPE H
5	HEIGHT 161	HEIGHT 194	HEIGHT 22	HEIGHT 9	HEIGHT 29	HEIGHT 8
6	LONGEVITY 87	LONGEVITY 115	LONGEVITY 80	LONGEVITY 14	LONGEVITY 16	LONGEVITY 20
7	MATURITY 4	MATURITY 10	MATURITY 4	MATURITY 3	MATURITY 3	MATURITY 3
8	MAX_STRATUM 4	MAX_STRATUM 4	MAX_STRATUM 2	MAX_STRATUM 1	MAX_STRATUM 2	MAX_STRATUM 1
9	MAX_ABUNDANCE 1	MAX_ABUNDANCE 1	MAX_ABUNDANCE 2	MAX_ABUNDANCE 3	MAX_ABUNDANCE 3	MAX_ABUNDANCE 3
10	IMM_SIZE 5	IMM_SIZE 5	IMM_SIZE 5	IMM_SIZE 10	IMM_SIZE 8	IMM_SIZE 10
11	CHANG_STR_AGES 0 1 2 8 10000 10000 10000	CHANG_STR_AGES 0 1 3 11 10000 10000 10000	CHANG_STR_AGES 0 7 10000 10000 10000 10000 10000	CHANG_STR_AGES 0 10000 10000 10000 10000 10000 10000	CHANG_STR_AGES 0 2 10000 10000 10000 10000 10000	CHANG_STR_AGES 0 10000 10000 10000 10000 10000 10000
12	SEED_POOL_LIFE 0 0	SEED_POOL_LIFE 0 0	SEED_POOL_LIFE 0 0	SEED_POOL_LIFE 0 0	SEED_POOL_LIFE 0 0	SEED_POOL_LIFE 0 0
13	SEED_DORMANCY 0	SEED_DORMANCY 0	SEED_DORMANCY 0	SEED_DORMANCY 0	SEED_DORMANCY 0	SEED_DORMANCY 0
14	POTENTIAL_FECUNDITY	POTENTIAL_FECUNDITY	POTENTIAL_FECUNDITY	POTENTIAL_FECUNDITY	POTENTIAL_FECUNDITY	POTENTIAL_FECUNDI
15	IS_ALIEN 0	IS_ALIEN 0	IS_ALIEN 0	IS_ALIEN 0	IS_ALIEN 0	IS_ALIEN 0
16	FLAMMABILITY 0	FLAMMABILITY 0	FLAMMABILITY 0	FLAMMABILITY 0	FLAMMABILITY 0	FLAMMABILITY 0

Showing 1 to 16 of 16 entries — Previous 1 Next

b. Optional modules

i. Dispersal

« *The quantity of produced seeds depends on the abundances of mature PFGs and their habitat suitability. A seed dispersal model determines seed inflow in each pixel (Fig. 1c). From the source, three circles of influence are defined using distance parameters. In the first circle, 50% of the seeds are distributed uniformly. In the second circle, 49% of the seeds are distributed with the same concentration as in the first circle but by pairs of pixels, simulating the spatial autocorrelation of dispersed seeds. In the third circle, 1% of the seeds fall into a random pixel. This seed dispersal model behaves similar to a continuous kernel function (see Fig.S1a) but is very effective and requires only a few parameters (Vittoz & Engler, 2007).* » (Boulangeat, 2014)

Required parameters :

- In GlobalParameters file :

```
DO_DISPERSAL 1
DISPERSAL_MODE 1
```

```
DISPERSAL_SAVING 0
```

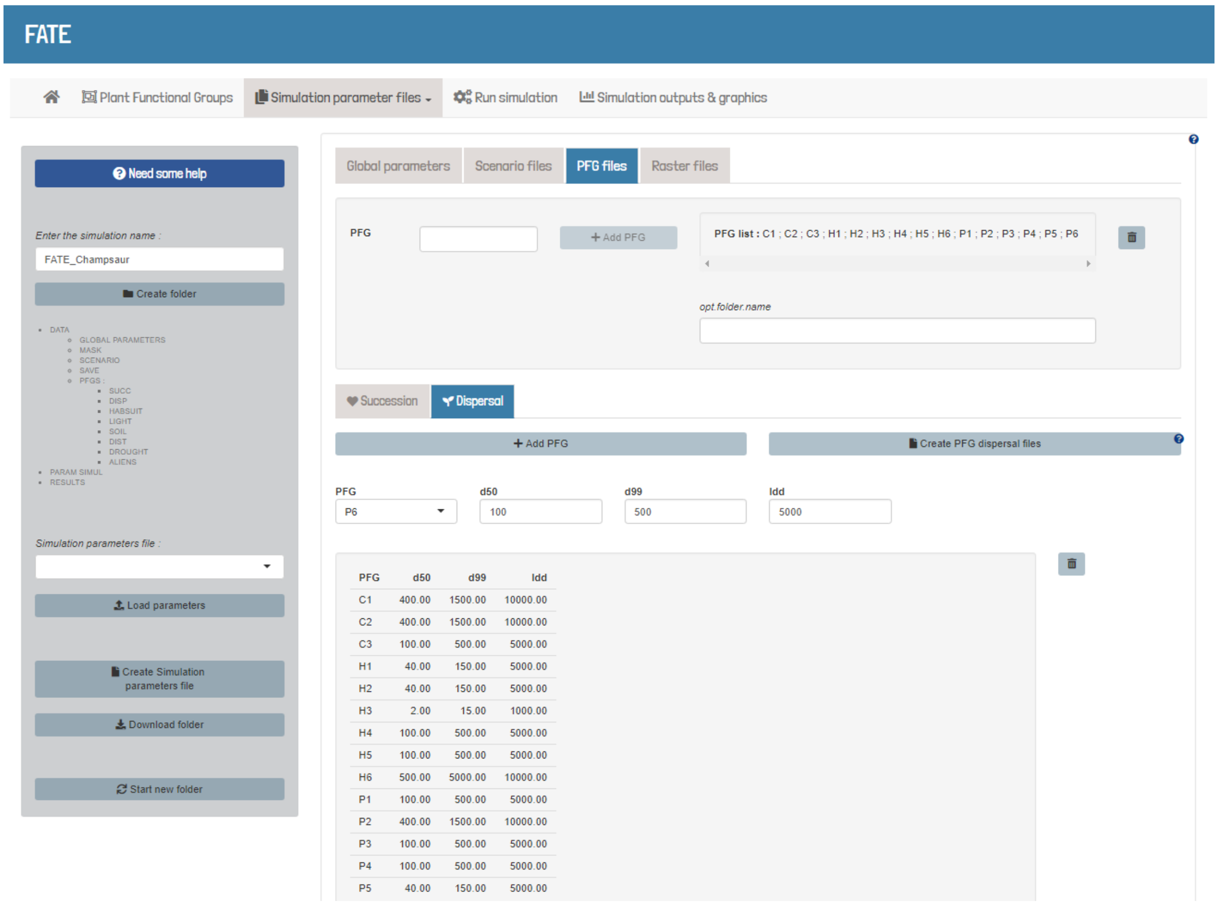
- In SimulParameters file :

```
--PFG_PARAMS_DISPERSAL--
FATE_simulation/DATA/PFGS/DISP/DISP_PFG1.txt
FATE_simulation/DATA/PFGS/DISP/DISP_PFG2.txt
```

- In DISPERSAL files (given with the `--PFG_PARAMS_DISPERSAL--` flag in *SimulParameters* file) :

```
NAME PFG1
DISPERS_DIST 100 500 79000
```

```r
# ## Load example data
# Champsaur_PFG = .loadData("Champsaur_PFG", "RData")
#
# ## Build PFG traits for dispersal
# tab.traits = Champsaur_PFG$PFG.traits
# ## Dispersal values
# ##    = Short: 0.1-2m;     Medium: 40-100m;     Long: 400-500m
# ##    = Vittoz correspondance : 1-3: Short;     4-5: Medium;    6-7:Long
# corres = data.frame(dispersal = 1:7
#                      , d50 = c(0.1, 0.5, 2, 40, 100, 400, 500)
#                      , d99 = c(1, 5, 15, 150, 500, 1500, 5000)
#                      , ldd = c(1000, 1000, 1000, 5000, 5000, 10000, 10000))
# tab.traits$d50 = corres$d50[tab.traits$dispersal]
# tab.traits$d99 = corres$d99[tab.traits$dispersal]
# tab.traits$ldd = corres$ldd[tab.traits$dispersal]
# str(tab.traits)


## Load example data
Champsaur_params = .loadData("Champsaur_params", "RData")

## Create a skeleton folder
PRE_FATE.skeletonDirectory(name.simulation = 'FATE_Champsaur')


## PFG traits for dispersal
tab.disp = Champsaur_params$tab.DISP
str(tab.disp)

## Create PFG dispersal parameter files -------------------------------------
PRE_FATE.params_PFGdispersal(name.simulation = 'FATE_Champsaur'
                             , mat.PFG.disp = Champsaur_params$tab.DISP)
```



Enter the simulation name : FATE_Champsaur

Simulation parameters file :

PFG

PFG list : C1 ; C2 ; C3 ; H1 ; H2 ; H3 ; H4 ; H5 ; H6 ; P1 ; P2 ; P3 ; P4 ; P5 ; P6

opt.folder.name

PFG	d50	d99	ldd
P6	100	500	5000

PFG	d50	d99	ldd
C1	400.00	1500.00	10000.00
C2	400.00	1500.00	10000.00
C3	100.00	500.00	5000.00
H1	40.00	150.00	5000.00
H2	40.00	150.00	5000.00
H3	2.00	15.00	1000.00
H4	100.00	500.00	5000.00
H5	100.00	500.00	5000.00
H6	500.00	5000.00	10000.00
P1	100.00	500.00	5000.00
P2	400.00	1500.00	10000.00
P3	100.00	500.00	5000.00
P4	100.00	500.00	5000.00
P5	40.00	150.00	5000.00

ii. Habitat suitability

« *Modelling how habitat suitability affects species population dynamics is tricky given the limited knowledge on the type and form of this relationship. Gallien et al. (2010) suggested a parsimonious approach using only presence-absences or a linear link. In FATE-HD, the probability for recruitment and seed production occurring is calculated every year according to the habitat suitability of the PFG in the pixel in question. Over time, the probability of presence is thus linearly related to fecundity and establishment. Accounting for interannual variability allows species coexistence via temporal niches. Mortality does not depend on habitat suitability, as the immediate effects of annual abiotic conditions on plant mortality are not clear in the literature. Habitat suitability for each PFG can be obtained from various sources such as correlative species distribution models (Guisan & Thuiller, 2005) or mechanistic niche models (Chuine & Beaubien, 2001).* » (Boulangeat, 2014)

Required parameters :

- In GlobalParameters file :

```
DO_HAB_SUITABILITY 1
HABSUIT_MODE 1
```

- In SimulParameters file :

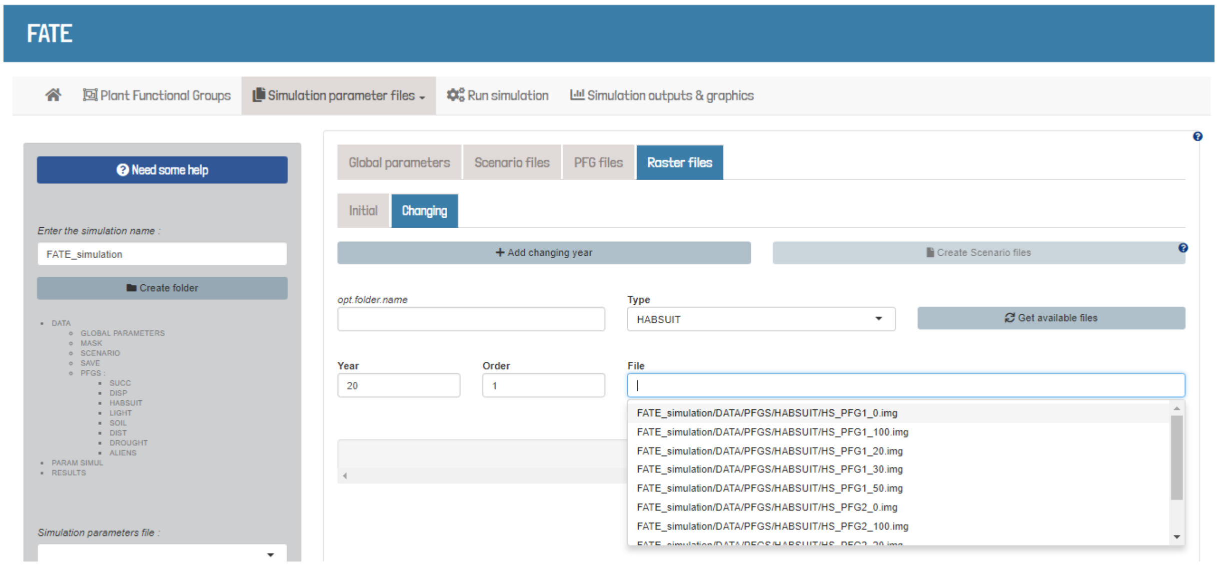
```
--PFG_MASK_HABSUIT--
FATE_simulation/DATA/PFGS/HABSUIT/HS_CA/HS_PFG1_0.tif
FATE_simulation/DATA/PFGS/HABSUIT/HS_CA/HS_PFG2_0.tif
```

Habitat suitability maps can be changed through simulation time.

Two supplementary type of files are then needed :

1. a file with each row indicating each simulation year of change
2. as many files as the number of years indicated in the previous file, and inside them, as many lines as PFG, with a path for a new habitat suitability for each of them

```
--HABSUIT_CHANGEMASK_YEARS--
FATE_simulation/DATA/SCENARIO/HABSUIT/HABSUIT_changingmask_years.txt
--HABSUIT_CHANGEMASK_FILES--
FATE_simulation/DATA/SCENARIO/HABSUIT/HABSUIT_changingmask_files_t20.txt
FATE_simulation/DATA/SCENARIO/HABSUIT/HABSUIT_changingmask_files_t30.txt
FATE_simulation/DATA/SCENARIO/HABSUIT/HABSUIT_changingmask_files_t50.txt
FATE_simulation/DATA/SCENARIO/HABSUIT/HABSUIT_changingmask_files_t100.txt
```



Enter the simulation name : FATE_simulation

Simulation parameters file :

opt.folder.name

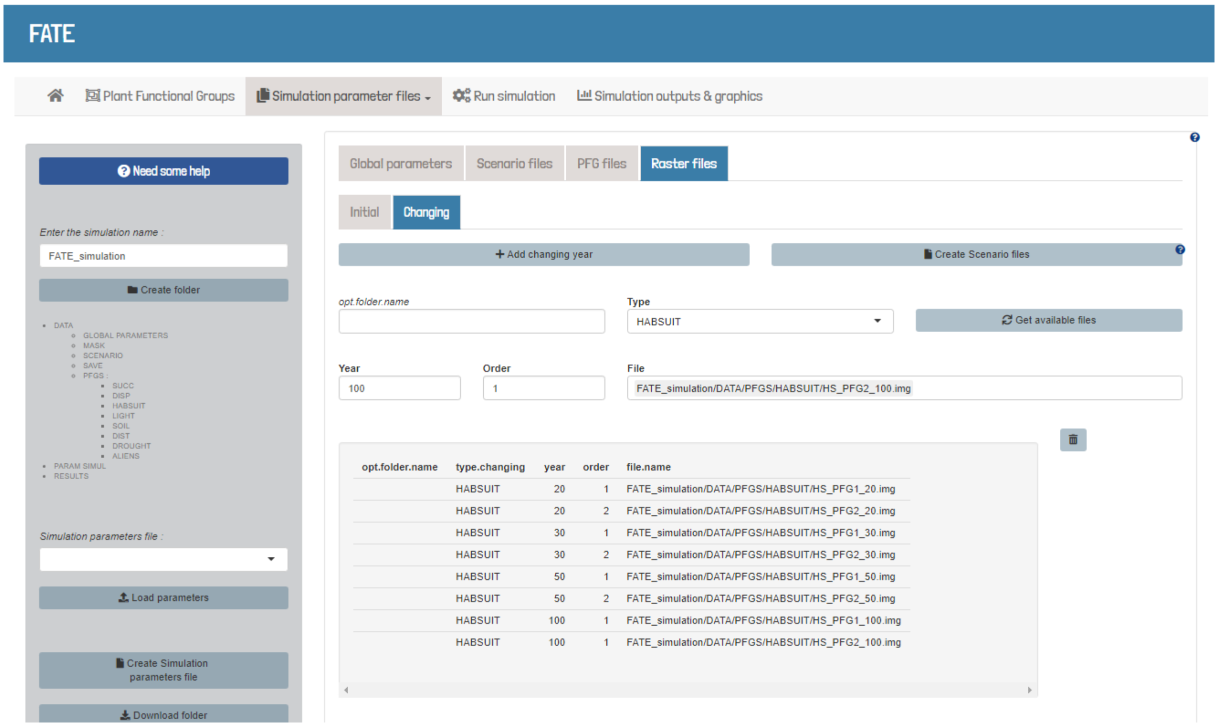
Type: HABSUIT

Year: 20

Order: 1

File: |

FATE_simulation/DATA/PFGS/HABSUIT/HS_PFG1_0.img
FATE_simulation/DATA/PFGS/HABSUIT/HS_PFG1_100.img
FATE_simulation/DATA/PFGS/HABSUIT/HS_PFG1_20.img
FATE_simulation/DATA/PFGS/HABSUIT/HS_PFG1_30.img
FATE_simulation/DATA/PFGS/HABSUIT/HS_PFG1_50.img
FATE_simulation/DATA/PFGS/HABSUIT/HS_PFG2_0.img
FATE_simulation/DATA/PFGS/HABSUIT/HS_PFG2_100.img



Enter the simulation name : FATE_simulation

Simulation parameters file :

opt.folder.name

Type: HABSUIT

Year: 100

Order: 1

File: FATE_simulation/DATA/PFGS/HABSUIT/HS_PFG2_100.img

opt.folder.name	type.changing	year	order	file.name
	HABSUIT	20	1	FATE_simulation/DATA/PFGS/HABSUIT/HS_PFG1_20.img
	HABSUIT	20	2	FATE_simulation/DATA/PFGS/HABSUIT/HS_PFG2_20.img
	HABSUIT	30	1	FATE_simulation/DATA/PFGS/HABSUIT/HS_PFG1_30.img
	HABSUIT	30	2	FATE_simulation/DATA/PFGS/HABSUIT/HS_PFG2_30.img
	HABSUIT	50	1	FATE_simulation/DATA/PFGS/HABSUIT/HS_PFG1_50.img
	HABSUIT	50	2	FATE_simulation/DATA/PFGS/HABSUIT/HS_PFG2_50.img
	HABSUIT	100	1	FATE_simulation/DATA/PFGS/HABSUIT/HS_PFG1_100.img
	HABSUIT	100	1	FATE_simulation/DATA/PFGS/HABSUIT/HS_PFG2_100.img

iii. Light interaction

« *Vegetation height is represented by a limited number of strata to incorporate the shading process (Fig. 1a). Within a pixel, the light resource for each stratum is calculated from the total abundance of all PFGs across all the upper strata. Within-pixel spatial heterogeneity in light resources is not taken into consideration* » (Boulangeat, 2014)

Required parameters :

- In GlobalParameters file :

```
DO_LIGHT_INTERACTION 1
LIGHT_THRESH_MEDIUM 13000
LIGHT_THRESH_LOW 19000
```

```
LIGHT_SAVING 1
```

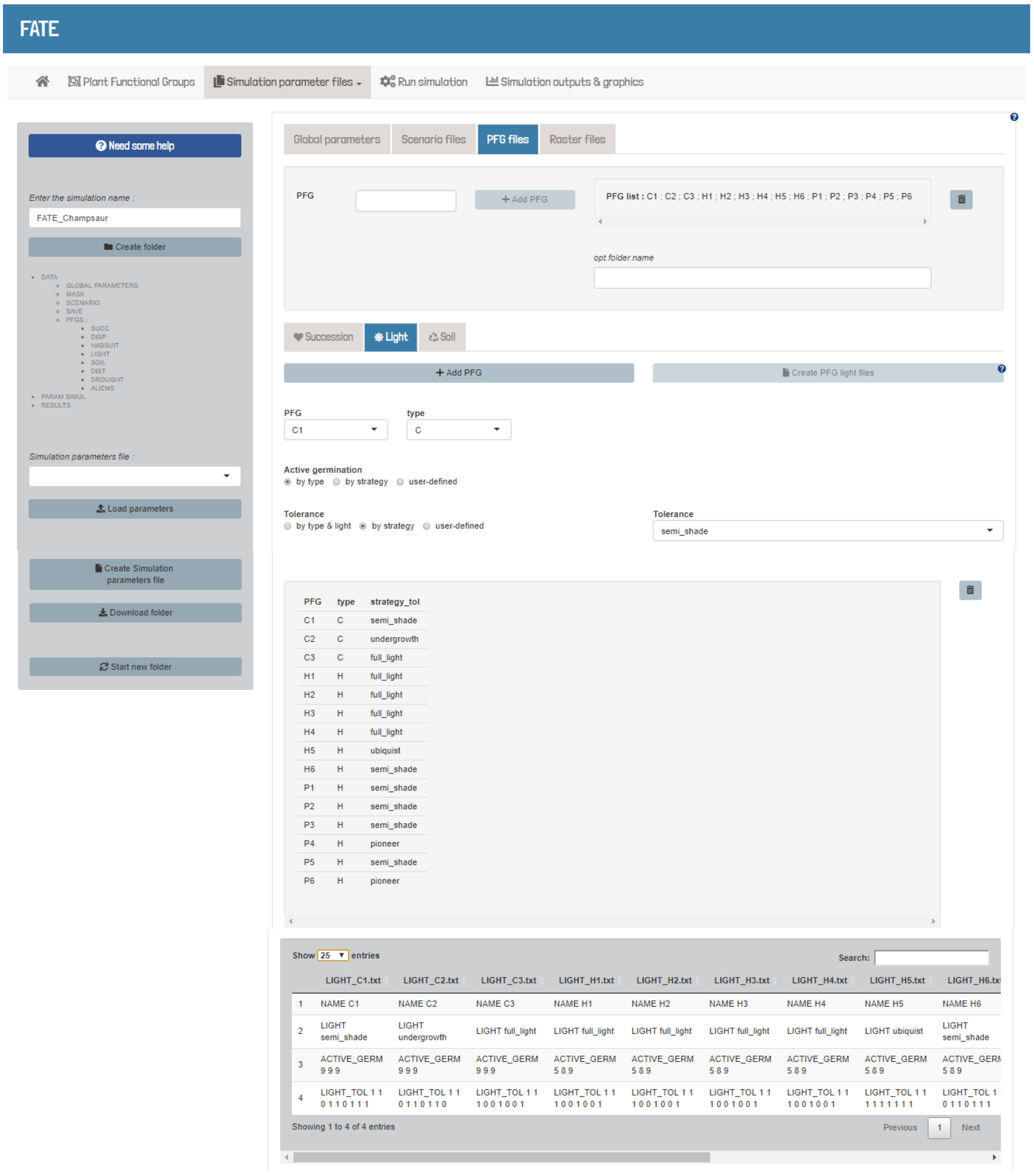
- In SimulParameters file :

```
--PFG_PARAMS_LIGHT--
FATE_simulation/DATA/PFGS/LIGHT/LIGHT_PFG1.txt
FATE_simulation/DATA/PFGS/LIGHT/LIGHT_PFG2.txt
```

- In LIGHT files (given with the `--PFG_PARAMS_LIGHT--` flag in *SimulParameters* file) :

```
NAME PFG1
ACTIVE_GERM 9 9 9
LIGHT 4
LIGHT_TOL 1 1 1 1 1 1 1 1 1
```

```
## Load example data
Champsaur_params = .loadData("Champsaur_params", "RData")

## Create a skeleton folder
PRE_FATE.skeletonDirectory(name.simulation = 'FATE_Champsaur')


## PFG traits for light
tab.light = Champsaur_params$tab.LIGHT
str(tab.light)

## Create PFG light parameter files -----------------------------------------
PRE_FATE.params_PFGlight(name.simulation = 'FATE_Champsaur'
                         , mat.PFG.light = tab.light[, c("PFG", "type")]
                         , mat.PFG.tol = tab.light[, c("PFG", "strategy_tol")])
```



Enter the simulation name : FATE_Champsaur

Simulation parameters file :

PFG

PFG list : C1 ; C2 ; C3 ; H1 ; H2 ; H3 ; H4 ; H5 ; H6 ; P1 ; P2 ; P3 ; P4 ; P5 ; P6

opt.folder.name

PFG: C1
type: C

Active germination
[✓ by type] by strategy user-defined

Tolerance
by type & light [✓ by strategy] user-defined

Tolerance: semi_shade

PFG	type	strategy_tol
C1	C	semi_shade
C2	C	undergrowth
C3	C	full_light
H1	H	full_light
H2	H	full_light
H3	H	full_light
H4	H	full_light
H5	H	ubiquist
H6	H	semi_shade
P1	H	semi_shade
P2	H	semi_shade
P3	H	semi_shade
P4	H	pioneer
P5	H	semi_shade
P6	H	pioneer

Show 25 entries Search:

	LIGHT_C1.txt	LIGHT_C2.txt	LIGHT_C3.txt	LIGHT_H1.txt	LIGHT_H2.txt	LIGHT_H3.txt	LIGHT_H4.txt	LIGHT_H5.txt	LIGHT_H6.txt
1	NAME C1	NAME C2	NAME C3	NAME H1	NAME H2	NAME H3	NAME H4	NAME H5	NAME H6
2	LIGHT semi_shade	LIGHT undergrowth	LIGHT full_light	LIGHT full_light	LIGHT full_light	LIGHT full_light	LIGHT full_light	LIGHT ubiquist	LIGHT semi_shade
3	ACTIVE_GERM 9 9 9	ACTIVE_GERM 9 9 9	ACTIVE_GERM 9 9 9	ACTIVE_GERM 5 8 9	ACTIVE_GERM 5 8 9	ACTIVE_GERM 5 8 9	ACTIVE_GERM 5 8 9	ACTIVE_GERM 5 8 9	ACTIVE_GERM 5 8 9
4	LIGHT_TOL 1 1 0 1 1 0 1 1 1	LIGHT_TOL 1 1 0 1 1 0 1 1 0	LIGHT_TOL 1 1 1 0 0 1 0 0 1	LIGHT_TOL 1 1 1 0 0 1 0 0 1	LIGHT_TOL 1 1 1 0 0 1 0 0 1	LIGHT_TOL 1 1 1 0 0 1 0 0 1	LIGHT_TOL 1 1 1 0 0 1 0 0 1	LIGHT_TOL 1 1 1 1 1 1 1 1 1	LIGHT_TOL 1 0 1 1 0 1 1 1

Showing 1 to 4 of 4 entries Previous 1 Next

iv. Soil interaction

"To be written" ()

Required parameters :

- In GlobalParameters file :

```
DO_SOIL_INTERACTION 1
SOIL_INIT 2.5
SOIL_RETENTION 0.8
```

```
SOIL_SAVING 1
```

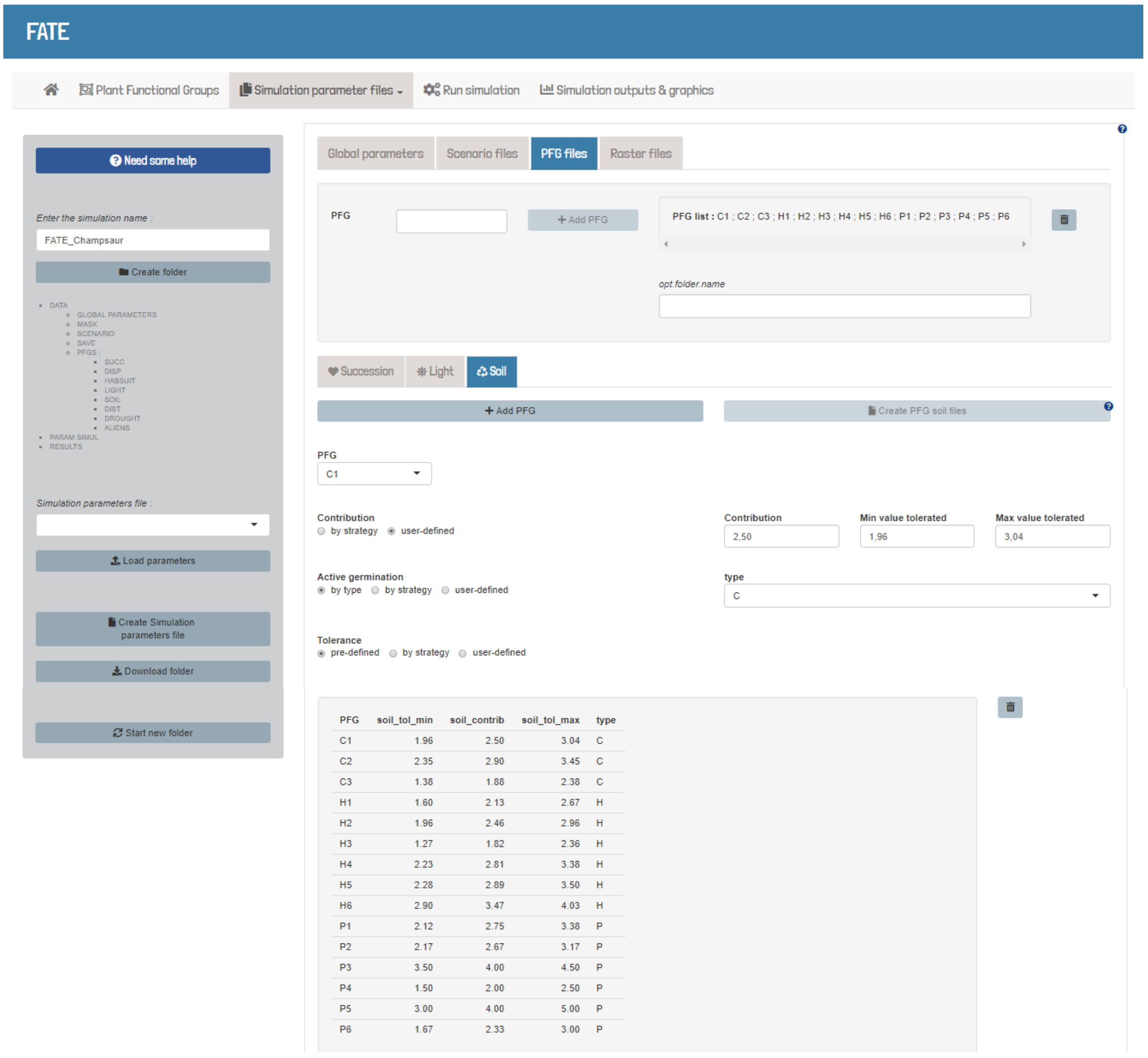
- In SimulParameters file :

```
--PFG_PARAMS_SOIL--
FATE_simulation/DATA/PFGS/SOIL/SOIL_PFG1.txt
FATE_simulation/DATA/PFGS/SOIL/SOIL_PFG2.txt
```

- In SOIL files (given with the `--PFG_PARAMS_SOIL--` flag in *SimulParameters* file) :

```
NAME PFG1
ACTIVE_GERM 8 10 5
SOIL_CONTRIB 2.4
SOIL_LOW 1
SOIL_HIGH 4
SOIL_TOL 1 10 0 5 10 4 9 10 8
```

```
## Load example data
Champsaur_params = .loadData("Champsaur_params", "RData")

## Create a skeleton folder
PRE_FATE.skeletonDirectory(name.simulation = 'FATE_Champsaur')


## PFG traits for soil
tab.soil = Champsaur_params$tab.SOIL
str(tab.soil)

## Create PFG soil parameter files -------------------------------------------
PRE_FATE.params_PFGsoil(name.simulation = 'FATE_Champsaur'
                        , mat.PFG.soil = tab.soil)
```



Enter the simulation name : FATE_Champsaur

Simulation parameters file :

PFG

PFG list : C1 ; C2 ; C3 ; H1 ; H2 ; H3 ; H4 ; H5 ; H6 ; P1 ; P2 ; P3 ; P4 ; P5 ; P6

opt.folder.name

PFG: C1

Contribution: by strategy [✓ user-defined]

Contribution: 2,50 — Min value tolerated: 1,96 — Max value tolerated: 3,04

Active germination: [✓ by type] by strategy user-defined

type: C

Tolerance: [✓ pre-defined] by strategy user-defined

PFG	soil_tol_min	soil_contrib	soil_tol_max	type
C1	1.96	2.50	3.04	C
C2	2.35	2.90	3.45	C
C3	1.38	1.88	2.38	C
H1	1.60	2.13	2.67	H
H2	1.96	2.46	2.96	H
H3	1.27	1.82	2.36	H
H4	2.23	2.81	3.38	H
H5	2.28	2.89	3.50	H
H6	2.90	3.47	4.03	H
P1	2.12	2.75	3.38	P
P2	2.17	2.67	3.17	P
P3	3.50	4.00	4.50	P
P4	1.50	2.00	2.50	P
P5	3.00	4.00	5.00	P
P6	1.67	2.33	3.00	P

v. Disturbances

« *Several disturbance models can be parameterized to remove vegetation, affect fecundity, kill seeds or activate dormant seeds according to each PFG's tolerance or sensitivity to the given disturbance. (Fig. 1d).* » (Boulangeat, 2014)

Required parameters :

- In GlobalParameters file :

```
DO_DISTURBANCES 1
DIST_NO 4
DIST_NOSUB 4
DIST_FREQ 1 1 1 1
```

- In SimulParameters file :

```
--PFG_PARAMS_DISTURBANCES--
FATE_simulation/DATA/PFGS/DIST/DIST_PFG1.txt
FATE_simulation/DATA/PFGS/DIST/DIST_PFG2.txt
--DIST_MASK--
FATE_simulation/DATA/MASK/mask_noPerturb.tif
FATE_simulation/DATA/MASK/mask_mowing.tif
FATE_simulation/DATA/MASK/mask_grazing_level1.tif
FATE_simulation/DATA/MASK/mask_grazing_level2.tif
```

Like habitat suitability maps, disturbances maps can be changed through simulation time.

Two supplementary type of files are then needed :

1. a file with each row indicating each simulation year of change
2. as many files as the number of years indicated in the previous file, and inside them, as many lines as disturbances, with a path for a new mask for each of them

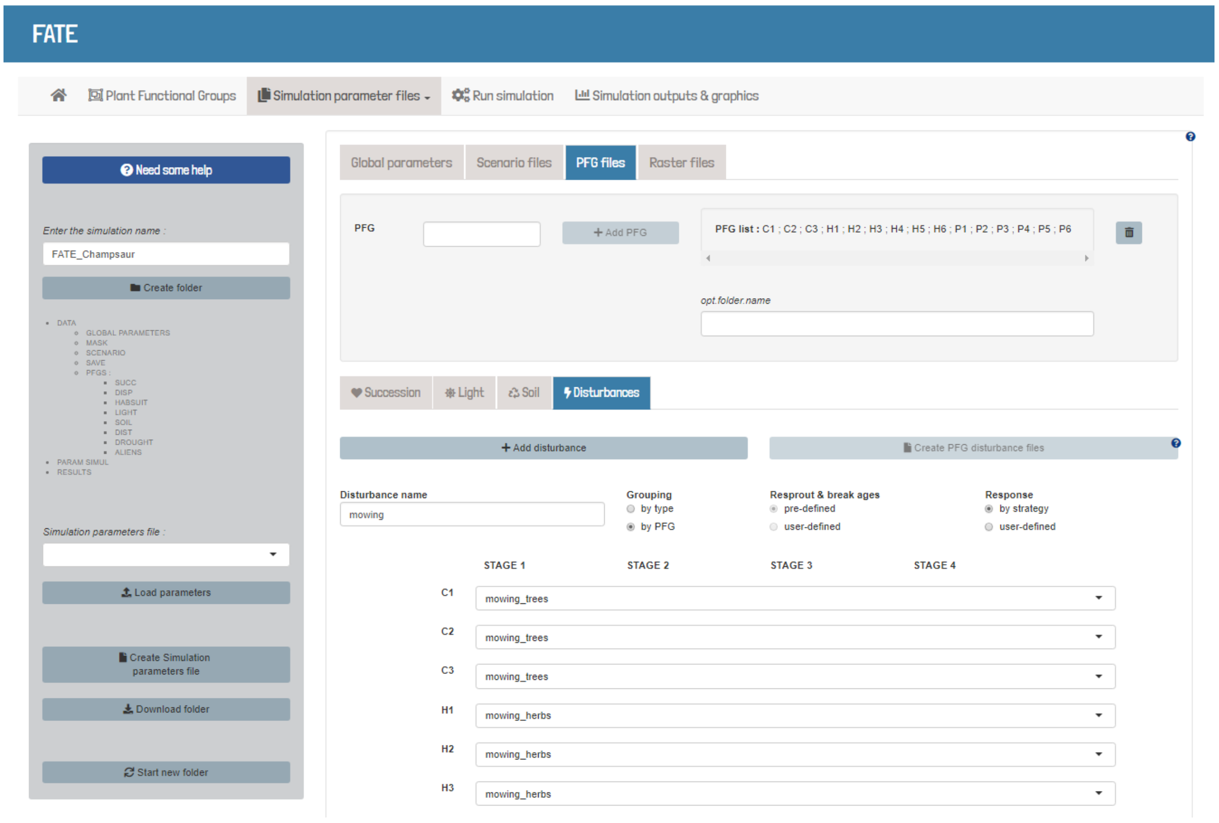
```
--DIST_CHANGEMASK_YEARS--
FATE_simulation/DATA/SCENARIO/DIST_scen1/DIST_changingmask_years.txt
--DIST_CHANGEMASK_FILES--
FATE_simulation/DATA/SCENARIO/DIST_scen1/DIST_changingmask_files_t50.txt
FATE_simulation/DATA/SCENARIO/DIST_scen1/DIST_changingmask_files_t100.txt
FATE_simulation/DATA/SCENARIO/DIST_scen1/DIST_changingmask_files_t150.txt
```

```
## Load example data
Champsaur_params = .loadData("Champsaur_params", "RData")

## Create a skeleton folder
PRE_FATE.skeletonDirectory(name.simulation = 'FATE_Champsaur')


## Changing_times table for disturbance
tab.changing = data.frame(year = c(600, 601, 800, 801)
                          , order = rep(1, 4)
                          , new.value = c("MASK_mowing.img"
                                          , "MASK_noDisturb.img"
                                          , "MASK_mowing.img"
                                          , "MASK_noDisturb.img"))

## Create a Changing_times parameter file -----------------------------------
PRE_FATE.params_changingYears(name.simulation = 'FATE_Champsaur'
                              , type.changing = 'DIST'
                              , mat.changing = tab.changing)
```

- In DISTURBANCE files (given with the `--PFG_PARAMS_DISTURBANCES--` flag in *SimulParameters* file) :

```
NAME PFG1
BREAK_AGE 2 4 10 2 4 10 2 4 10 2 4 10
RESPR_AGE 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
FATES 0 0 0 0 5 2 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 2 0 1
PROP_KILLED 0 0 0 0
ACTIVATED_SEED 0 0 0 0
```

```
## Load example data
Champsaur_params = .loadData("Champsaur_params", "RData")

## Create a skeleton folder
PRE_FATE.skeletonDirectory(name.simulation = 'FATE_Champsaur')


## PFG traits for succession
tab.succ = Champsaur_params$tab.SUCC
str(tab.succ)

## Create PFG succession parameter files (fixing strata limits) --------------
PRE_FATE.params_PFGsuccession(name.simulation = 'FATE_Champsaur'
                              , mat.PFG.succ = tab.succ
                              , strata.limits = c(0, 20, 50, 150, 400, 1000, 2000)
                              , strata.limits_reduce = FALSE)

tmp = fread("FATE_Champsaur/DATA/PFGS/SUCC_COMPLETE_TABLE.csv")
tab.succ = Champsaur_params$tab.SUCC
tab.succ$age_above_150cm = tmp$CHANG_STR_AGES_to_str_4_150

## PFG traits for disturbance
tab.dist = Champsaur_params$tab.DIST
str(tab.dist)

## Create PFG response to disturbance parameter files (give warnings) --------
PRE_FATE.params_PFGdisturbance(name.simulation = 'FATE_Champsaur'
                               , mat.PFG.dist = tab.succ
                               , mat.PFG.tol = tab.dist)
```



Enter the simulation name : FATE_Champsaur

Simulation parameters file :

PFG

PFG list : C1 ; C2 ; C3 ; H1 ; H2 ; H3 ; H4 ; H5 ; H6 ; P1 ; P2 ; P3 ; P4 ; P5 ; P6

opt.folder.name

Disturbance name: mowing

Grouping: by type [✓ by PFG]

Resprout & break ages: [✓ pre-defined] user-defined

Response: [✓ by strategy] user-defined

	STAGE 1	STAGE 2	STAGE 3	STAGE 4
C1	mowing_trees			
C2	mowing_trees			
C3	mowing_trees			
H1	mowing_herbs			
H2	mowing_herbs			
H3	mowing_herbs			

vi. Drought

« *Fig. 1. Drought simulation scheme. For each year i, a PFG's habitat suitability (HS; step 1) and drought effects (step 2) are evaluated within a pixel j. If HS ij or Din ij are below reference values, PFG fecundity and recruitment are set to 0. Additionally, if Din ij crosses the reference value, one drought year is added to the PFG's cumulative drought effects counter. Severe drought effects occur if conditions 2.3.1 ii or 2.3.2 are met, consisting in immediate and post-drought effects. Otherwise, only moderate drought effects are caused (2.1 and 2.3.1 i). Drought recovery is simulated by subtracting one or two drought events from the cumulative drought effects counter. […] See Table S3 in Appendix S2 for full parameter list and refer to main text for further details.* » (Barros, 2017)

Required parameters :

- In GlobalParameters file :

```
DO_DROUGHT_DISTURBANCE 1
DROUGHT_NOSUB 4
```

- In SimulParameters file :

```
--PFG_PARAMS_DROUGHT--
FATE_simulation/DATA/PFGS/DROUGHT/DROUGHT_PFG1.txt
FATE_simulation/DATA/PFGS/DROUGHT/DROUGHT_PFG2.txt
--DROUGHT_MASK--
FATE_simulation/DATA/MASK/DROUGHT_init.tif
```

Like habitat suitability or disturbances maps, the drought index map can be changed through simulation time.

Two supplementary type of files are then needed :

1. a file with each row indicating each simulation year of change
2. as many files as the number of years indicated in the previous file, and inside them, one line with a path for a new map of drought index

```
--DROUGHT_CHANGEMASK_YEARS--
FATE_simulation/DATA/SCENARIO/DROUGHT_changingmask_years.txt
--DROUGHT_CHANGEMASK_FILES--
FATE_simulation/DATA/SCENARIO/DROUGHT_changingmask_files_t15.txt
FATE_simulation/DATA/SCENARIO/DROUGHT_changingmask_files_t30.txt
FATE_simulation/DATA/SCENARIO/DROUGHT_changingmask_files_t45.txt
```

- In DROUGHT files (given with the `--PFG_PARAMS_DROUGHT--` flag in *SimulParameters* file) :

```
NAME PFG1
BREAK_AGE 1 5 26 1 5 26
RESPR_AGE 0 0 3 26 0 0 3 26
FATES 6 0 2 0 5 0 6 0 2 0 5 0 1 0 2 0
PROP_KILLED 0 0
ACTIVATED_SEED 0 0
THRESHOLD_MOD -11.50406848
THRESHOLD_SEV -12.3335733
COUNTER_RECOVERY 1
COUNTER_SENS 3
COUNTER_CUM 3
```

vii. Aliens

« *We then simulated the introduction of the alien PFGs through annual seeding. The sites of simulated introduction were based on a map of the Human Footprint […] an index combining information on land-use, population density and transportation network (including mountain footpaths). As such it represents an excellent proxy of potential local propagule pressure for introduced species […] In the current propagule pressure scenario, introductions were a proportion of a set maximum number of seeds depending on the human footprint value in each pixel (i.e. highest introduction intensity in the most densely populated centres, and lowest introduction intensity along mountain footpaths; see Appendix S2 for maps and for details). In the increased propagule pressure scenario, the maximum introduction level was applied in all areas that had a non-zero human footprint (simulating a maximum exploitation of all areas suitable to humans).* » (Carboni, 2017)

Required parameters :

- In GlobalParameters file :

```
DO_ALIENS_INTRODUCTION 1
ALIENS_NO 4
ALIENS_FREQ 2 2 2 2
```

- In SimulParameters file :

```
--PFG_MASK_ALIENS--
FATE_simulation/DATA/PFGS/ALIENS/NoIntroduction.tif
FATE_simulation/DATA/PFGS/ALIENS/Introduction_ALIEN2.tif
FATE_simulation/DATA/PFGS/ALIENS/Introduction_ALIEN3.tif
FATE_simulation/DATA/PFGS/ALIENS/Introduction_ALIEN4.tif
```

Like habitat suitability or disturbances maps, aliens introduction masks can be changed through simulation time.

Two supplementary type of files are then needed :

1. a file with each row indicating each simulation year of change
2. as many files as the number of years indicated in the previous file,
   and inside them, as many lines as PFG, with a path for a new mask for each of them

```
--ALIENS_CHANGEMASK_YEARS--
FATE_simulation/DATA/SCENARIO/ALIENS_changingmask_years.txt
--ALIENS_CHANGEMASK_FILES--
FATE_simulation/DATA/SCENARIO/ALIENS_changingmask_files_t20.txt
FATE_simulation/DATA/SCENARIO/ALIENS_changingmask_files_t25.txt
```

Once introduction maps have been set, frequency of introduction can also be changed through simulation time.

Two supplementary type of files are then needed :

1. a file with each row indicating each simulation year of change
2. as many files as the number of years indicated in the previous file,
   and inside them, as many lines as PFG, with a number for each of them representing its introduction frequency

```
--ALIENS_CHANGEFREQ_YEARS--
FATE_simulation/DATA/SCENARIO/ALIENS_changingfreq_years.txt
--ALIENS_CHANGEFREQ_FILES--
FATE_simulation/DATA/SCENARIO/ALIENS_changingfreq_files_t20.txt
FATE_simulation/DATA/SCENARIO/ALIENS_changingfreq_files_t25.txt
```

- In SUCCESSION files (given with the `--PFG_PARAMS_LIFE_HISTORY--` flag in *SimulParameters* file) :

```
IS_ALIEN 1
```

viii. Fire

“To be written” ()

Required parameters :

- In GlobalParameters file :

```
DO_FIRE_DISTURBANCE 1
FIRE_NO 1
FIRE_NOSUB 4
FIRE_FREQ 2
FIRE_IGNIT_MODE 1
FIRE_NEIGH_MODE 2
FIRE_PROP_MODE 4
FIRE_QUOTA_MODE 2
```

Depending on the values given for the `FIRE_IGNIT_MODE`, `FIRE_NEIGH_MODE`, `FIRE_PROP_MODE` and `FIRE_QUOTA_MODE` parameters, more information might be needed :

```
FIRE_IGNIT_NO 12

FIRE_IGNIT_NOHIST 5 8 12 5 9 0 3 11 5 7 4

FIRE_IGNIT_LOGIS 0.6 2.5 0.05
FIRE_IGNIT_FLAMMMAX 10

FIRE_NEIGH_CC 4 3 4 3

FIRE_PROP_INTENSITY 0.5

FIRE_PROP_LOGIS 0.6 2.5 0.05

FIRE_QUOTA_MAX 1000
```

➢ In SimulParameters file :

```
--PFG_PARAMS_FIRE--
FATE_simulation/DATA/PFGS/FIRE/FIRE_PFG1.txt
FATE_simulation/DATA/PFGS/FIRE/FIRE_PFG2.txt
--FIRE_MASK--
FATE_simulation/DATA/MASK/FIRE_init.tif
```

Like habitat suitability or disturbances maps, fire masks can be changed through simulation time.

Two supplementary type of files are then needed :

1. a file with each row indicating each simulation year of change
2. as many files as the number of years indicated in the previous file,
   and inside them, as many lines as fire disturbances, with a path for a new mask for each of them

```
--FIRE_CHANGEMASK_YEARS--
FATE_simulation/DATA/SCENARIO/FIRE_changingmask_years.txt
--FIRE_CHANGEMASK_FILES--
FATE_simulation/DATA/SCENARIO/FIRE_changingmask_files_t20.txt
FATE_simulation/DATA/SCENARIO/FIRE_changingmask_files_t25.txt
```

Once introduction maps have been set, frequency of fires can also be changed through simulation time.

Two supplementary type of files are then needed :

1. a file with each row indicating each simulation year of change
2. as many files as the number of years indicated in the previous file,
   and inside them, as many lines as fire disturbances, with a number for each of them representing its occuring frequency

```
--FIRE_CHANGEFREQ_YEARS--
FATE_simulation/DATA/SCENARIO/FIRE_changingfreq_years.txt
--FIRE_CHANGEFREQ_FILES--
FATE_simulation/DATA/SCENARIO/FIRE_changingfreq_files_t20.txt
FATE_simulation/DATA/SCENARIO/FIRE_changingfreq_files_t25.txt
```

Depending on the value given for the `FIRE_PROP_MODE` parameter, more information might be needed :

```
--ELEVATION_MASK--
FATE_simulation/DATA/MASK/elevation.tif
--SLOPE_MASK--
FATE_simulation/DATA/MASK/slope.tif
```

- In SUCCESSION files (given with the `--PFG_PARAMS_LIFE_HISTORY--` flag in *SimulParameters* file) :

```
FLAMMABILITY 6
```

- In FIRE files (given with the `--PFG_PARAMS_FIRE--` flag in *SimulParameters* file) :

```
NAME PFG1
BREAK_AGE 1 4 20
RESPR_AGE 0 1 3 12
FATES 8 0 6 4 5 5 4 6
PROP_KILLED 0 0
ACTIVATED_SEED 0 0
```

3. Run a FATE simulation

When all data and parameter files have been produced and correctly referred in a SimulParameters file, a FATE simulation can be run using the FATE function :

- ✓ the simulation will start, and the software will **print messages into the console** indicating what the software is doing.
- ✓ depending on the simulation duration, the size and the resolution of the study area, the number of Plant Functional Groups (PFG) involved, … running the full simulation **could take a while**.

Note : the folder from which the command is sent must be adapted based on how paths to files have been given within the SimulParameters file :

- ➢ If all paths are **absolute**
  (i.e. including the root, such as */home/username/FATE_simulation/DATA/ GLOBAL_PARAMETERS/Global_parameters_V1.txt*),
  there should not be any problem. The only requirement then is to also give absolute path to the simulation folder, if not in the current directory.

- ➢ If all paths are **relative**
  (i.e. starting from a specific folder, such as *FATE_simulation/DATA/GLOBAL_PARAMETERS/Global_parameters_V1.txt*),
  then the FATE simulation must be run from this specific folder (i.e. here from the folder containing the FATE_simulation folder).

4. Outputs & analyses

a. FATE outputs

Once the simulation is completed, the directory defined under the flag `--SAVING_DIR--` within the SimulParameters file must contain the following directories (depending on the parameters selected and the modules activated within the GlobalParameters file) :

- *ABUND_allPFG_perStrata/*
- *ABUND_perPFG_allStrata/*
- *ABUND_perPFG_perStrata/*

Each of them contains raster files with the abundance of Plant Functional Groups (PFG) (which should be considered as a proxy for the vegetation coverage/abundance) contained within each pixel of the study area. The files within each folder show different informations :

- *ABUND_perPFG_perStrata/* :
  one specific year, and the abundance of **a specific PFG** in **a specific stratum**
- *ABUND_allPFG_perStrata/* :
  one specific year, and the abundance of **all PFGs** in **a specific stratum**
- *ABUND_perPFG_allStrata/* :
  one specific year, and the abundance of **a specific PFG** within **all strata**

If the corresponding modules were activated within the GlobalParameters file, the following folders may also exist and contain output files :

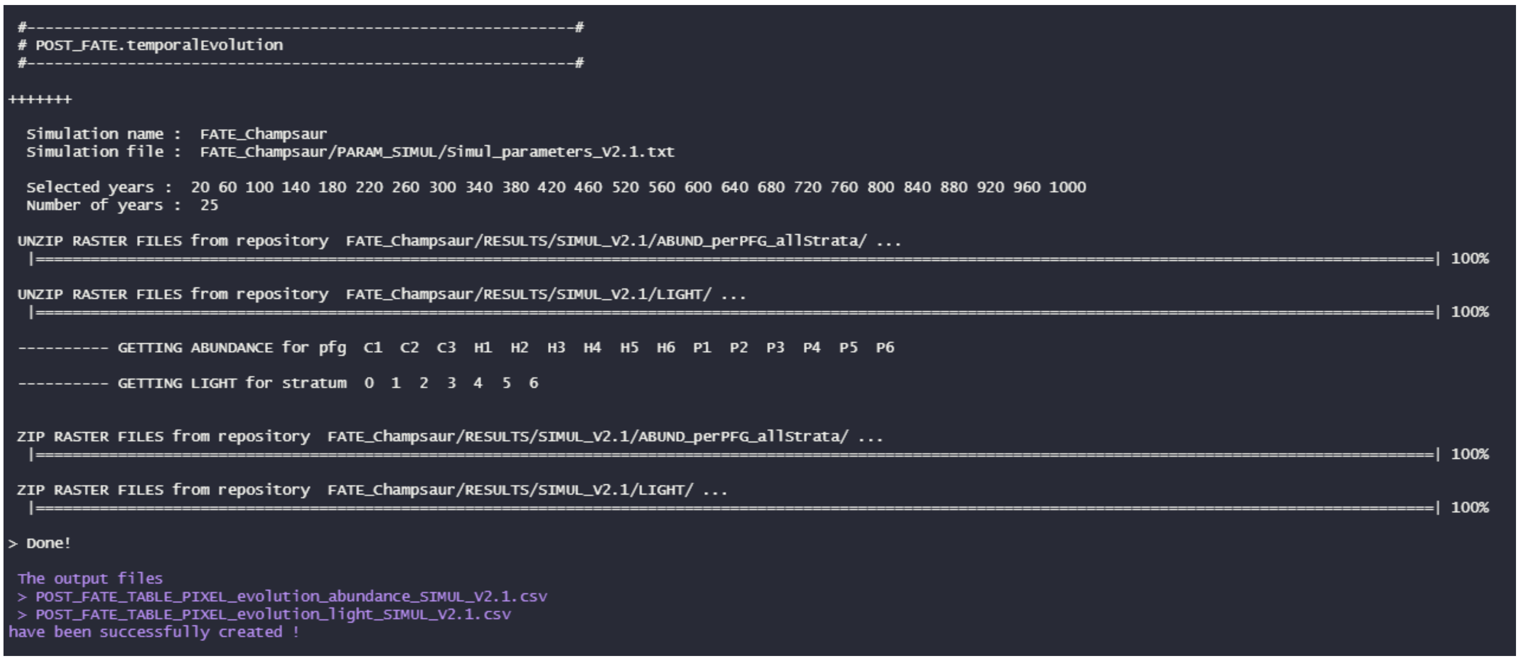
- *DISPERSAL/*
- *LIGHT/*
- *SOIL/*

These ouputs can then be used as is, or post-treated with functions from the RFate package to obtain more specific results (e.g. evolution of abundance curves, cover or diversity maps, etc).

b. Temporal evolution

Results are saved as raster files, and archived to save some place. Extracting and retrieving informations for each PFG, pixel or height stratum can take some time. The [POST_FATE.temporalEvolution]() function takes care of the **extraction** (which can be parallelized) and concatenates the values in one table file.

```
## Load example data
## .loadData("Champsaur_params", "7z")

## Select a parameter file
# param = "Simul_parameters_V1.1.txt"
param = "Simul_parameters_V2.1.txt"
# param = "Simul_parameters_V3.1.txt"
# param = "Simul_parameters_V4.1.txt"

simul.name = "FATE_Champsaur"
simul.param = paste0(simul.name, "/PARAM_SIMUL/", param)


## TEMPORAL EVOLUTION ------------------------------------------------------
POST_FATE.temporalEvolution(name.simulation = simul.name
                            , file.simulParam = simul.param
                            , no_years = 30)
```

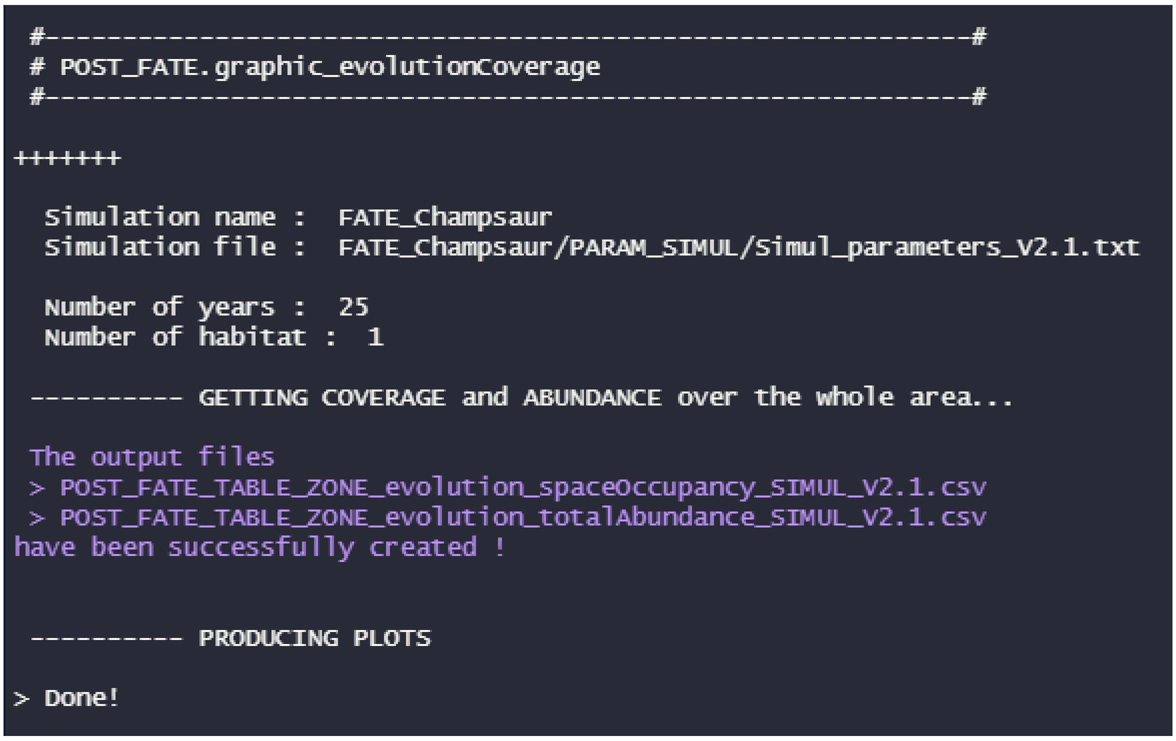


```
 #------------------------------------------------------------#
 # POST_FATE.temporalEvolution
 #------------------------------------------------------------#

+++++++

  Simulation name :  FATE_Champsaur
  Simulation file :  FATE_Champsaur/PARAM_SIMUL/Simul_parameters_V2.1.txt

  Selected years :  20 60 100 140 180 220 260 300 340 380 420 460 520 560 600 640 680 720 760 800 840 880 920 960 1000
  Number of years :  25

 UNZIP RASTER FILES from repository  FATE_Champsaur/RESULTS/SIMUL_V2.1/ABUND_perPFG_allStrata/ ...
  |======================================================================================================| 100%

 UNZIP RASTER FILES from repository  FATE_Champsaur/RESULTS/SIMUL_V2.1/LIGHT/ ...
  |======================================================================================================| 100%

 ---------- GETTING ABUNDANCE for pfg  C1  C2  C3  H1  H2  H3  H4  H5  H6  P1  P2  P3  P4  P5  P6

 ---------- GETTING LIGHT for stratum  0  1  2  3  4  5  6


 ZIP RASTER FILES from repository  FATE_Champsaur/RESULTS/SIMUL_V2.1/ABUND_perPFG_allStrata/ ...
  |======================================================================================================| 100%

 ZIP RASTER FILES from repository  FATE_Champsaur/RESULTS/SIMUL_V2.1/LIGHT/ ...
  |======================================================================================================| 100%

> Done!

 The output files
 > POST_FATE_TABLE_PIXEL_evolution_abundance_SIMUL_V2.1.csv
 > POST_FATE_TABLE_PIXEL_evolution_light_SIMUL_V2.1.csv
have been successfully created !
```

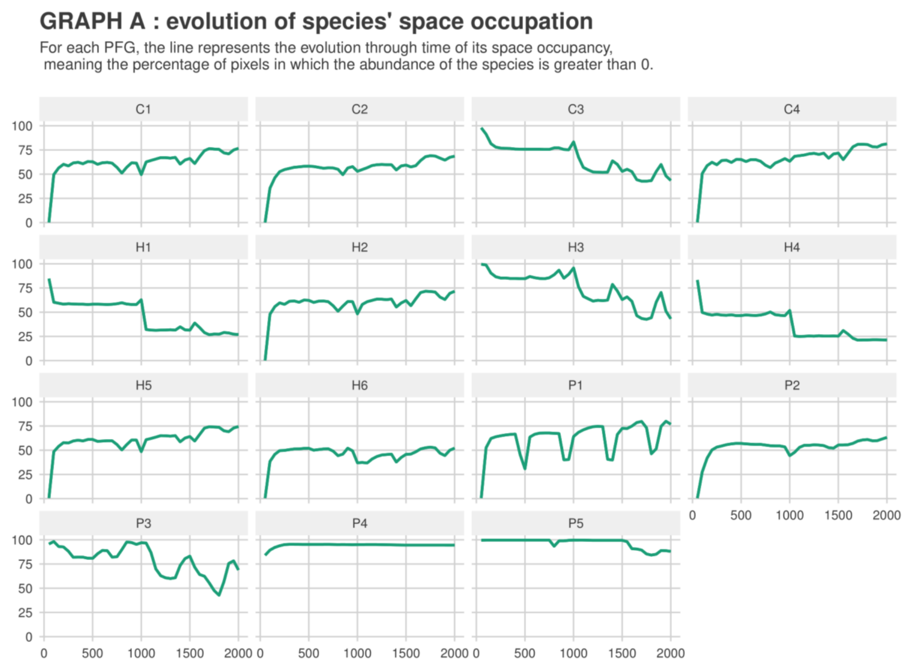
Once this table is obtained, three other functions are available to **represent the evolution of these results through the simulation time** :

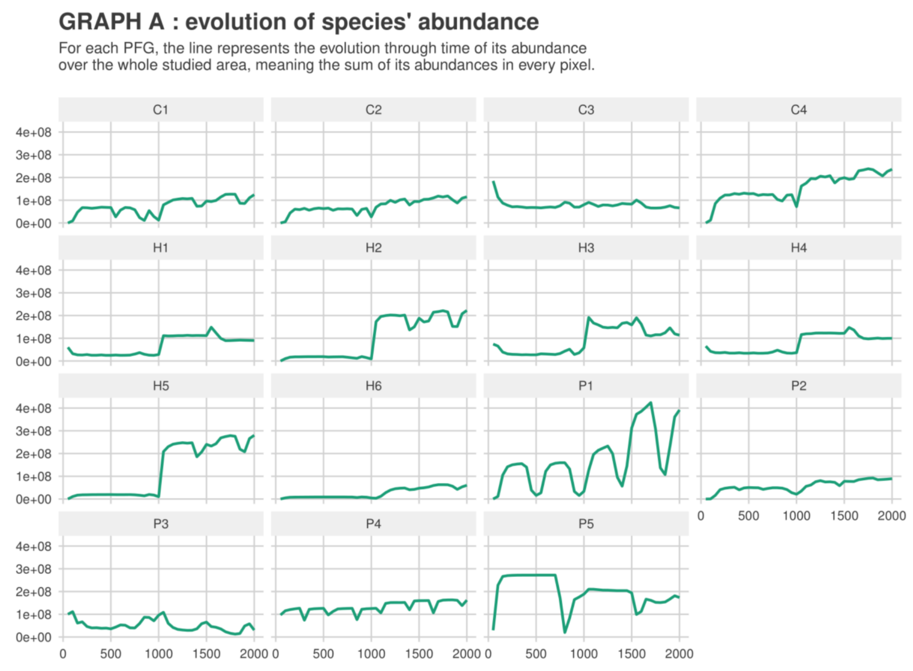
➢ The POST_FATE.graphic_evolutionCoverage function draws, for each PFG, its space occupation and its total abundance over the whole simulation area.

```
POST_FATE.graphic_evolutionCoverage(name.simulation = simul.name
                                    , file.simulParam = simul.param)
```



```
 #---------------------------------------------------------------#
 # POST_FATE.graphic_evolutionCoverage
 #---------------------------------------------------------------#

+++++++

  Simulation name :  FATE_Champsaur
  Simulation file :  FATE_Champsaur/PARAM_SIMUL/Simul_parameters_V2.1.txt

  Number of years :  25
  Number of habitat :  1

 ---------- GETTING COVERAGE and ABUNDANCE over the whole area...

 The output files
 > POST_FATE_TABLE_ZONE_evolution_spaceOccupancy_SIMUL_V2.1.csv
 > POST_FATE_TABLE_ZONE_evolution_totalAbundance_SIMUL_V2.1.csv
have been successfully created !


 ---------- PRODUCING PLOTS

> Done!
```



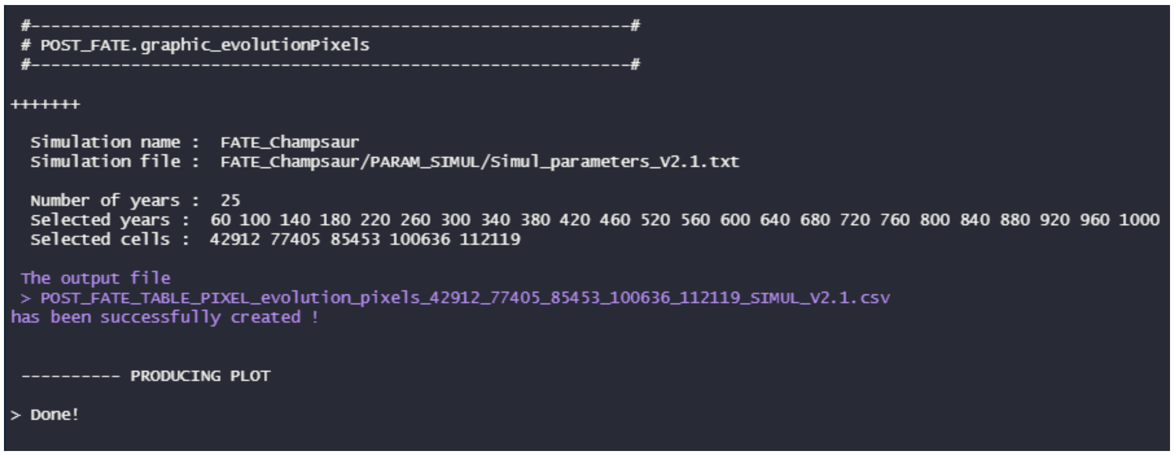


The evolution of the different PFG over the whole simulation area and timline can help to detect global effects, *such as the seeding timing (here it stops at 1000 simulation years), the survival of the plant functional groups, the colonization of space, etc.*

The same graphics can be obtained per habitat if a raster file is given to the POST_FATE.temporalEvolution function, to visualize global dynamic of each PFG within what should be different environment classes.

- The POST_FATE.graphic_evolutionPixels function focuses on some pixels (either given or randomly drawn). The abundance of PFG through time within these pixels is represented, together with the pixel resources (light, soil) if the corresponding modules were activated within the GlobalParameters file.

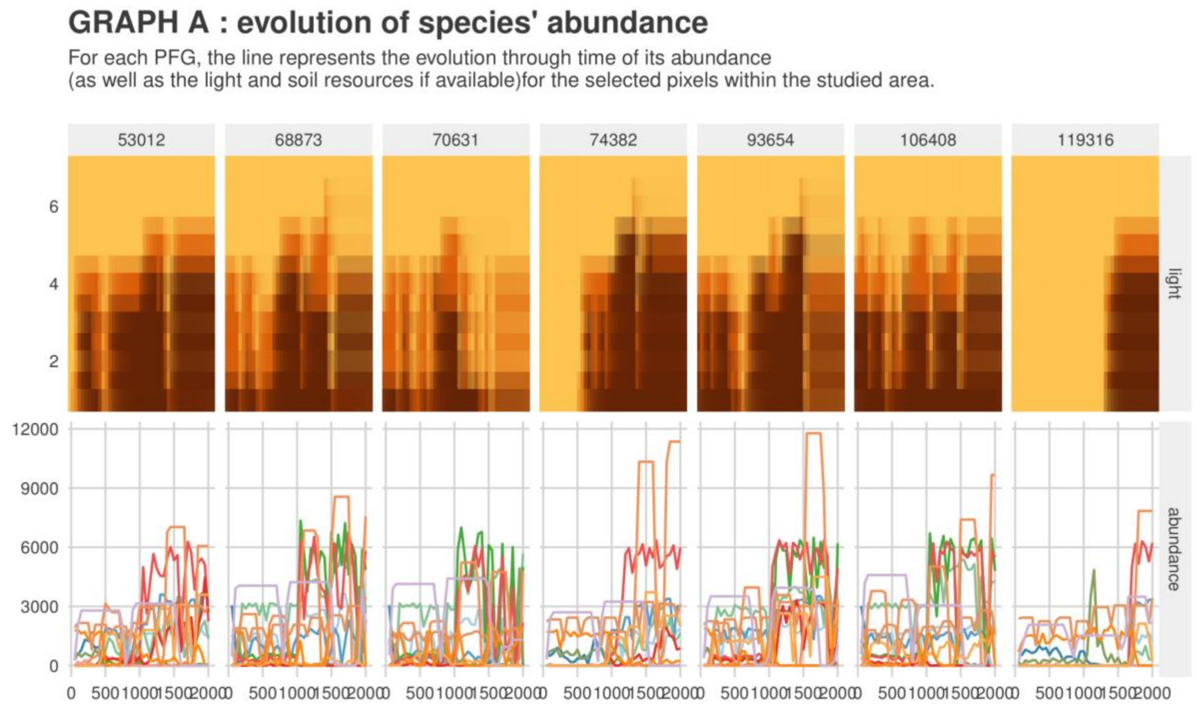
```
POST_FATE.graphic_evolutionPixels(name.simulation = simul.name
                                  , file.simulParam = simul.param)
```

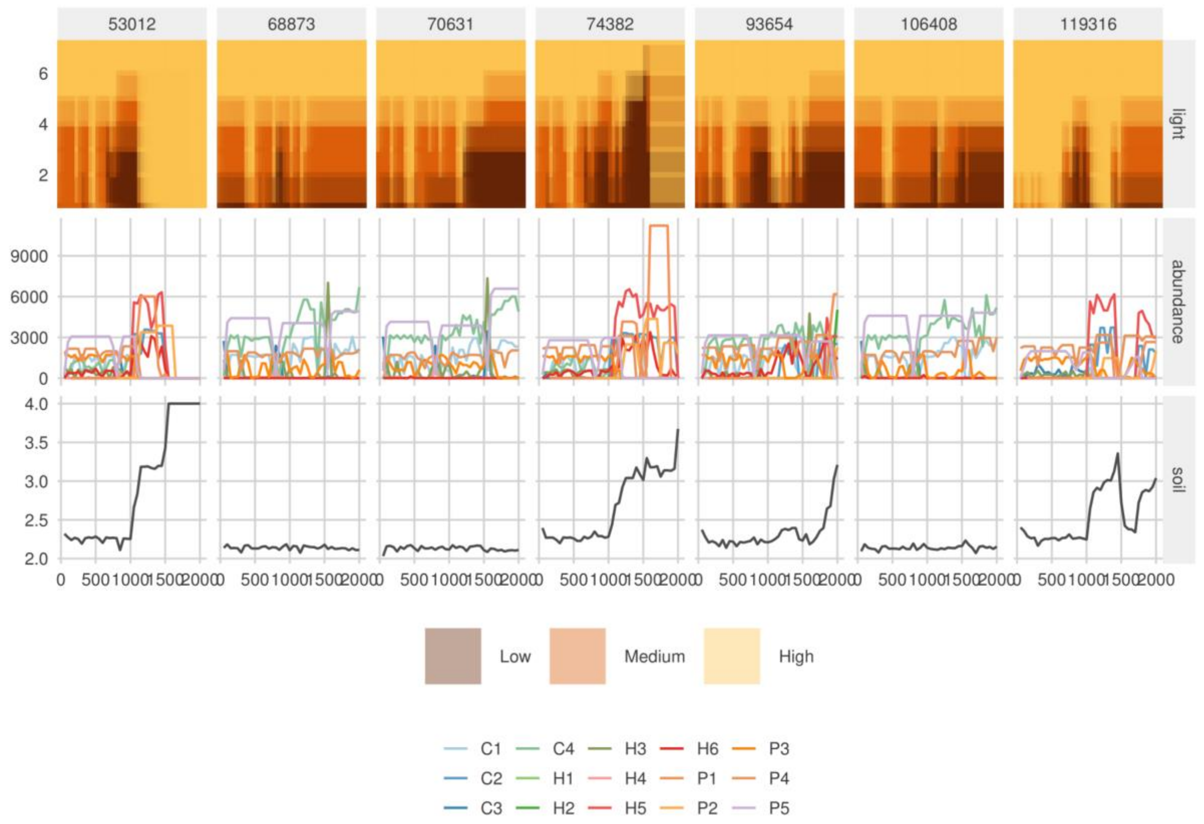


```
 #-------------------------------------------------------------#
 # POST_FATE.graphic_evolutionPixels
 #-------------------------------------------------------------#

+++++++

  Simulation name :  FATE_Champsaur
  Simulation file :  FATE_Champsaur/PARAM_SIMUL/Simul_parameters_V2.1.txt

  Number of years :  25
  Selected years :  60 100 140 180 220 260 300 340 380 420 460 520 560 600 640 680 720 760 800 840 880 920 960 1000
  Selected cells :  42912 77405 85453 100636 112119

 The output file
 > POST_FATE_TABLE_PIXEL_evolution_pixels_42912_77405_85453_100636_112119_SIMUL_V2.1.csv
has been successfully created !


 ---------- PRODUCING PLOT

> Done!
```

Zooming on pixel(s) allow to really « see » and explore the abundance dynamics between functional groups. This is particularly useful when interaction modules (light, soil) are activated : it is then that PFG really have an impact on each other.

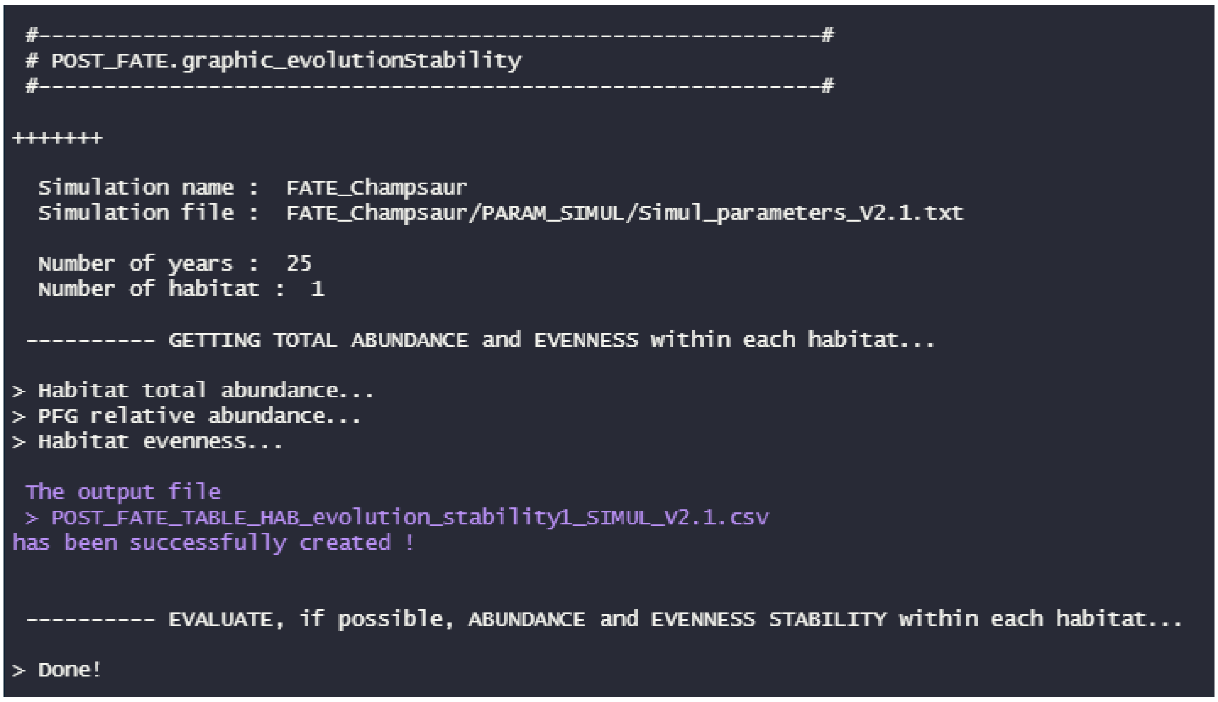
It can also be used to detect differences in PFG dynamics between different habitats classes ; or to compare the effect of the activation of different modules (by always selecting the same pixels between simulations).



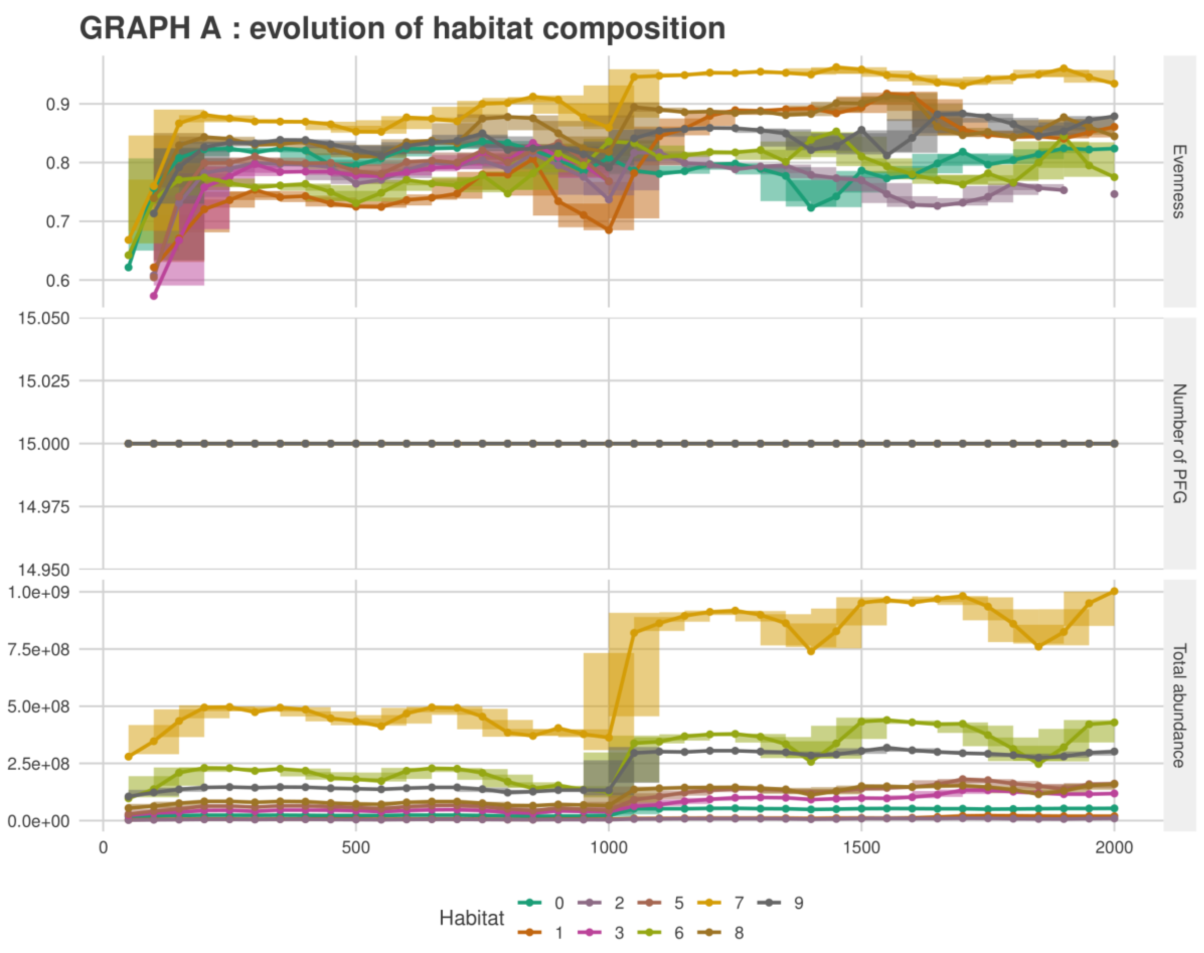


➢ The POST_FATE.graphic_evolutionStability function is a bit more specific, and was built to study the habitat composition stability over the whole simulation area.

```
POST_FATE.graphic_evolutionStability(name.simulation = simul.name
                                     , file.simulParam = simul.param)
```



```
 #-------------------------------------------------------------#
 # POST_FATE.graphic_evolutionStability
 #-------------------------------------------------------------#

+++++++

  Simulation name :  FATE_Champsaur
  Simulation file :  FATE_Champsaur/PARAM_SIMUL/Simul_parameters_V2.1.txt

  Number of years :  25
  Number of habitat :  1

 ---------- GETTING TOTAL ABUNDANCE and EVENNESS within each habitat...

> Habitat total abundance...
> PFG relative abundance...
> Habitat evenness...

 The output file
 > POST_FATE_TABLE_HAB_evolution_stability1_SIMUL_V2.1.csv
has been successfully created !


 ---------- EVALUATE, if possible, ABUNDANCE and EVENNESS STABILITY within each habitat...

> Done!
```



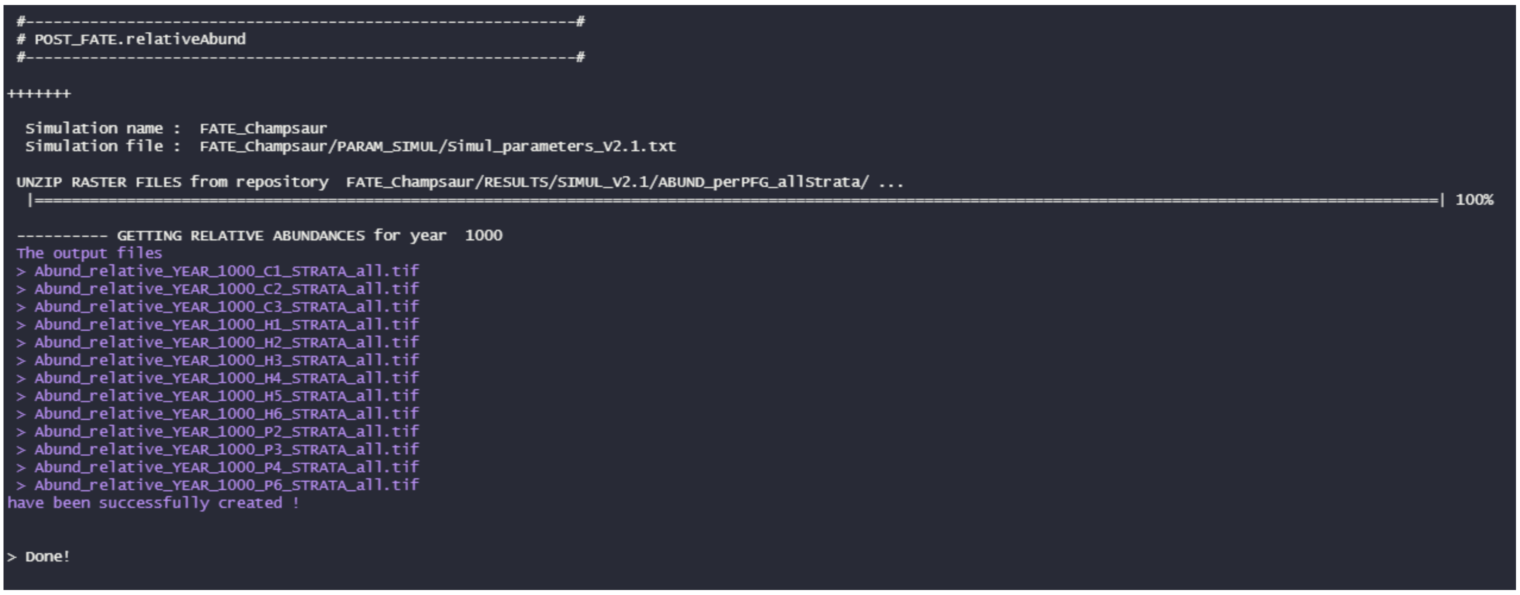
The evenness of an habitat represents the uniformity of its species composition. Close to 1 : all the species contained within the habitat have similar abundances. Close to 0 : the species present are very unbalanced in terms of proportions. *Thus, an habitat seeing its evenness decreasing may be colonized by a new dominant species (or simply one species already present takes precedence over the others).*

c. From abundance to presence/absence

As FATE abundances are quite arbitrary (no direct link in terms of number of individuals), results are more easily interpreted when transformed as **relative abundances** or **presence/absence** values. Several methods are available to perform this final transformation at the user's convenience.

- The POST_FATE.relativeAbund function creates, for the required years, raster maps of PFG relative abundances into the *ABUND_perPFG_allStrata/* folder.

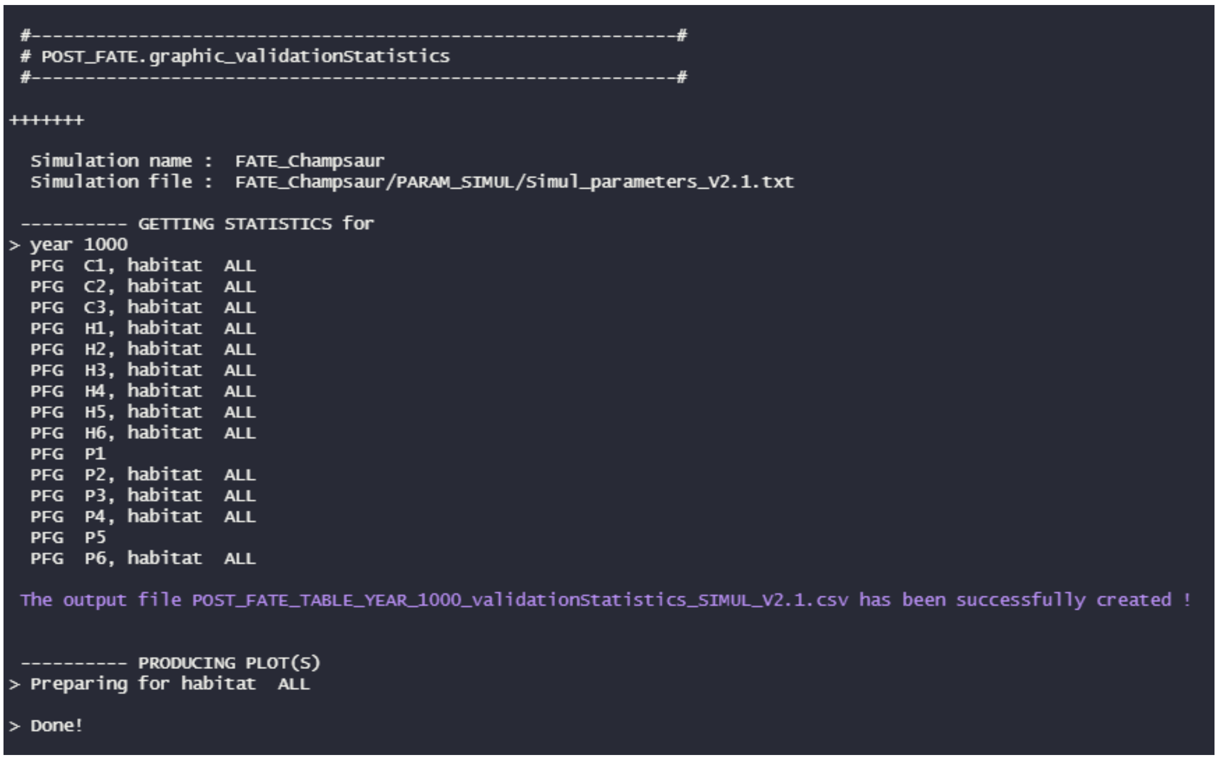
```
## RELATIVE ABUNDANCE / BINARY MAPS & VALIDATION -----------------------------
POST_FATE.relativeAbund(name.simulation = simul.name
                        , file.simulParam = simul.param
                        , years = 1000)
```



```
 #-------------------------------------------------------------#
 # POST_FATE.relativeAbund
 #-------------------------------------------------------------#

+++++++

  Simulation name :  FATE_Champsaur
  Simulation file :  FATE_Champsaur/PARAM_SIMUL/Simul_parameters_V2.1.txt

 UNZIP RASTER FILES from repository  FATE_Champsaur/RESULTS/SIMUL_V2.1/ABUND_perPFG_allStrata/ ...
  |=====================================================================================================| 100%

 ---------- GETTING RELATIVE ABUNDANCES for year  1000
 The output files
 > Abund_relative_YEAR_1000_C1_STRATA_all.tif
 > Abund_relative_YEAR_1000_C2_STRATA_all.tif
 > Abund_relative_YEAR_1000_C3_STRATA_all.tif
 > Abund_relative_YEAR_1000_H1_STRATA_all.tif
 > Abund_relative_YEAR_1000_H2_STRATA_all.tif
 > Abund_relative_YEAR_1000_H3_STRATA_all.tif
 > Abund_relative_YEAR_1000_H4_STRATA_all.tif
 > Abund_relative_YEAR_1000_H5_STRATA_all.tif
 > Abund_relative_YEAR_1000_H6_STRATA_all.tif
 > Abund_relative_YEAR_1000_P2_STRATA_all.tif
 > Abund_relative_YEAR_1000_P3_STRATA_all.tif
 > Abund_relative_YEAR_1000_P4_STRATA_all.tif
 > Abund_relative_YEAR_1000_P6_STRATA_all.tif
have been successfully created !


> Done!
```

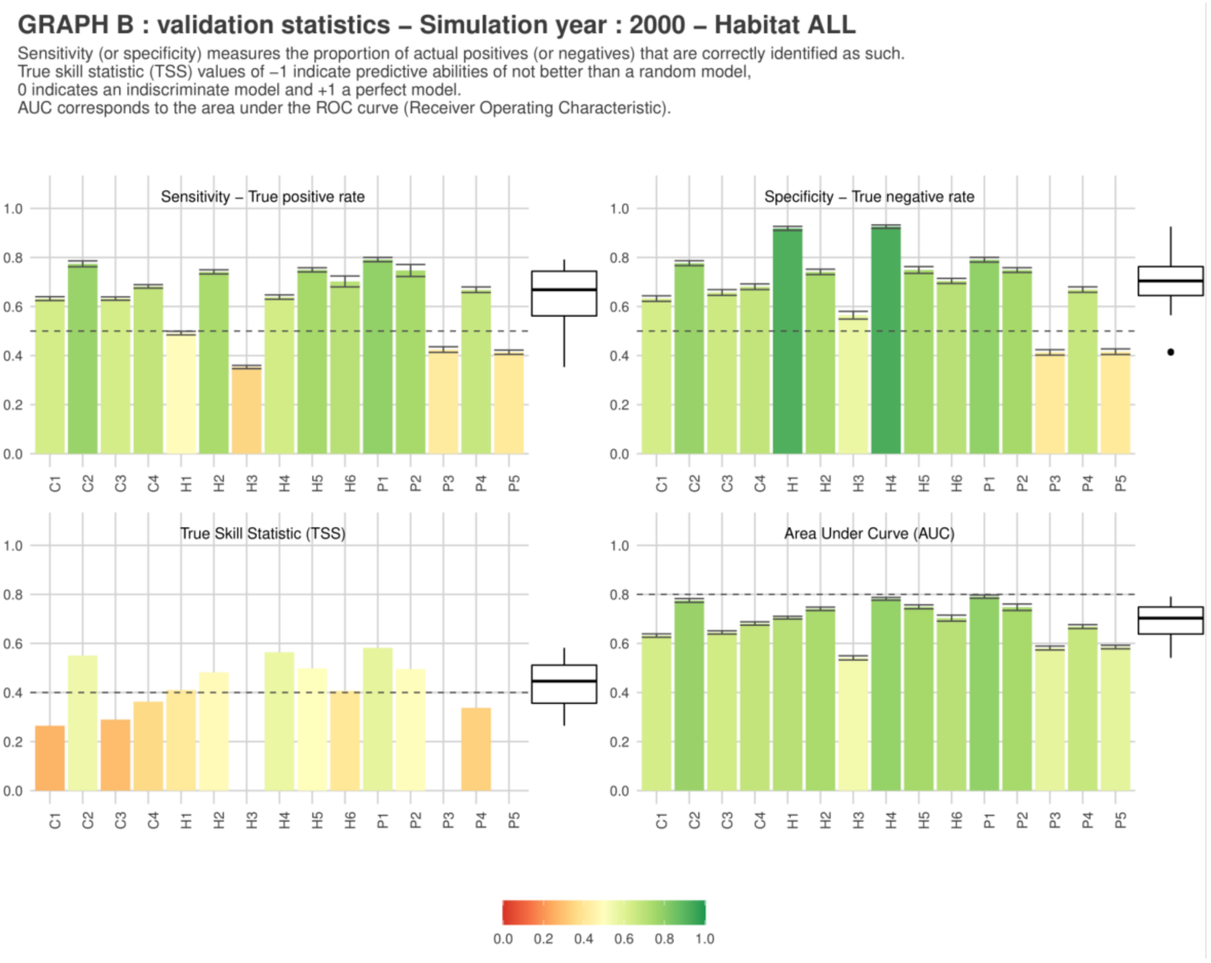
- The POST_FATE.graphic_validationStatistics function can then be used, to calculate and represent TSS and AUC values for each PFG. The threshold to convert relative abundances into presence/absence values is automatically chosen to maximize the TSS values.

```
Champsaur_params = .loadData("Champsaur_params", "RData")

## PFG observations
tab.occ = Champsaur_params$tab.occ
tab.xy = Champsaur_params$tab.xy


## Merge observations and sites coordinates
tab.obs = merge(tab.occ, tab.xy, by = "row.names")
tab.obs = melt(tab.obs, id.vars = c("Row.names", "X", "Y"))
colnames(tab.obs) = c("sites", "X", "Y", "PFG", "obs")
tab.obs = na.exclude(tab.obs)
head(tab.obs)

tab = tab.obs[, c("PFG", "X", "Y", "obs")]
POST_FATE.graphic_validationStatistics(name.simulation = simul.name
                                       , file.simulParam = simul.param
                                       , years = 1000
                                       , mat.PFG.obs = tab)
```



```
 #-----------------------------------------------------------#
 # POST_FATE.graphic_validationStatistics
 #-----------------------------------------------------------#

+++++++

  Simulation name :  FATE_Champsaur
  Simulation file :  FATE_Champsaur/PARAM_SIMUL/Simul_parameters_V2.1.txt

 ---------- GETTING STATISTICS for
> year 1000
  PFG  C1, habitat  ALL
  PFG  C2, habitat  ALL
  PFG  C3, habitat  ALL
  PFG  H1, habitat  ALL
  PFG  H2, habitat  ALL
  PFG  H3, habitat  ALL
  PFG  H4, habitat  ALL
  PFG  H5, habitat  ALL
  PFG  H6, habitat  ALL
  PFG  P1
  PFG  P2, habitat  ALL
  PFG  P3, habitat  ALL
  PFG  P4, habitat  ALL
  PFG  P5
  PFG  P6, habitat  ALL

 The output file POST_FATE_TABLE_YEAR_1000_validationStatistics_SIMUL_V2.1.csv has been successfully created !


 ---------- PRODUCING PLOT(S)
> Preparing for habitat  ALL

> Done!
```

Even if TSS or AUC measures are not at all adapted to validate the whole temporal dynamic of a FATE simulation, they can still provide interesting information about a specific state at a specific simulation time of the studied community.



- The POST_FATE.binaryMaps function creates, for the required years, raster maps of PFG presence/absence into the *BIN_perPFG_allStrata/* folder and *BIN_perPFG_ perStrata/* folders. Two methods are available to perform such transformation :

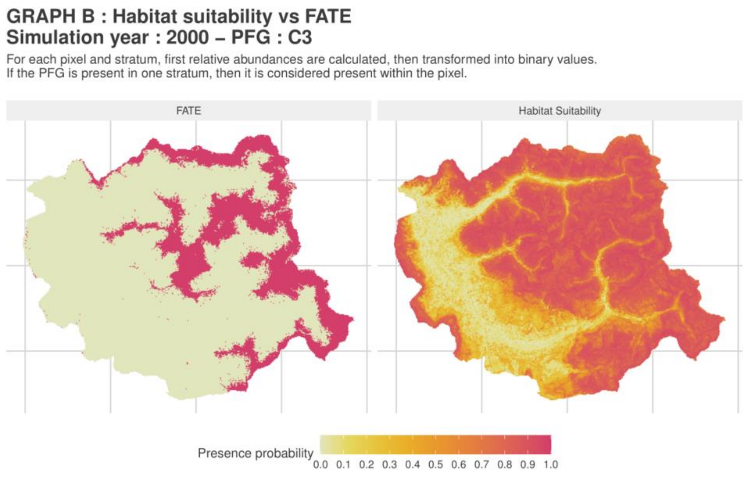
   1. By setting manually a conversion threshold for the relative abundance maps (*e.g. : all PFG representing more than 10% of a pixel total abundance are considered as present within this pixel*)

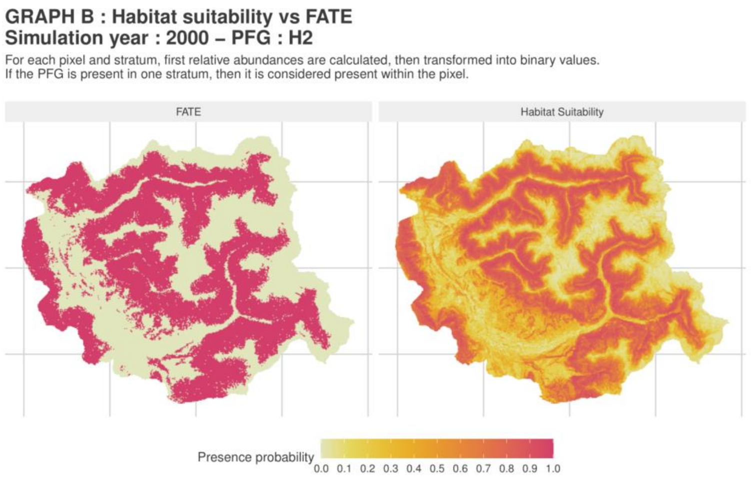
```
POST_FATE.binaryMaps(name.simulation = simul.name
                     , file.simulParam = simul.param
                     , years = 1000
                     , method = 1
                     , method1.threshold = 0.05)
```

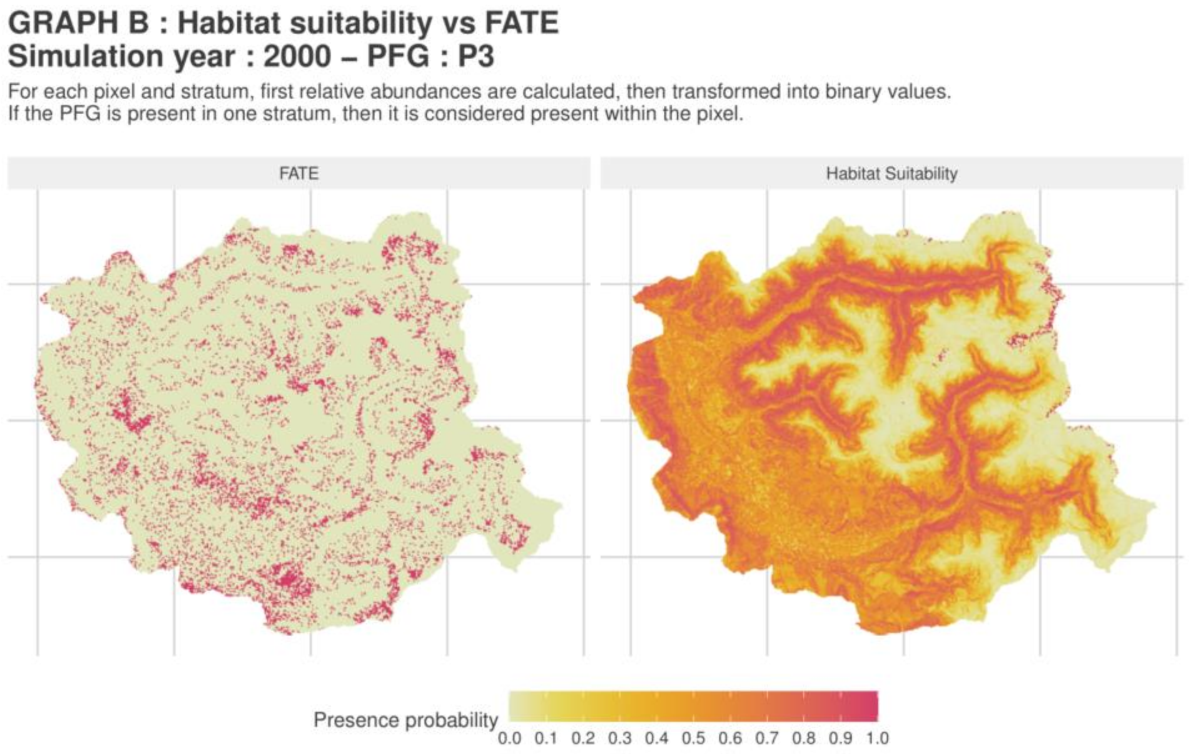
   2. By using specific cutoffs for each PFG, that can be obtained from the POST_FATE.graphic_validationStatistics function for example.

- The [POST_FATE.graphic_mapPFGvsHS] function represents together, for each PFG, its habitat suitability map (if the module was activated within the [GlobalParameters] file) and its presence/absence map obtained from the FATE simulation.

```
POST_FATE.graphic_mapPFGvsHS(name.simulation = simul.name
                            , file.simulParam = simul.param
                            , years = 2000)
```


GRAPH B : Habitat suitability vs FATE
Simulation year : 2000 – PFG : C3
For each pixel and stratum, first relative abundances are calculated, then transformed into binary values.
If the PFG is present in one stratum, then it is considered present within the pixel.
FATE
Habitat Suitability
Presence probability
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0


GRAPH B : Habitat suitability vs FATE
Simulation year : 2000 – PFG : H2
For each pixel and stratum, first relative abundances are calculated, then transformed into binary values.
If the PFG is present in one stratum, then it is considered present within the pixel.
FATE
Habitat Suitability
Presence probability
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0


GRAPH B : Habitat suitability vs FATE
Simulation year : 2000 – PFG : P3
For each pixel and stratum, first relative abundances are calculated, then transformed into binary values.
If the PFG is present in one stratum, then it is considered present within the pixel.
FATE
Habitat Suitability
Presence probability
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

d. Visualization of outputs

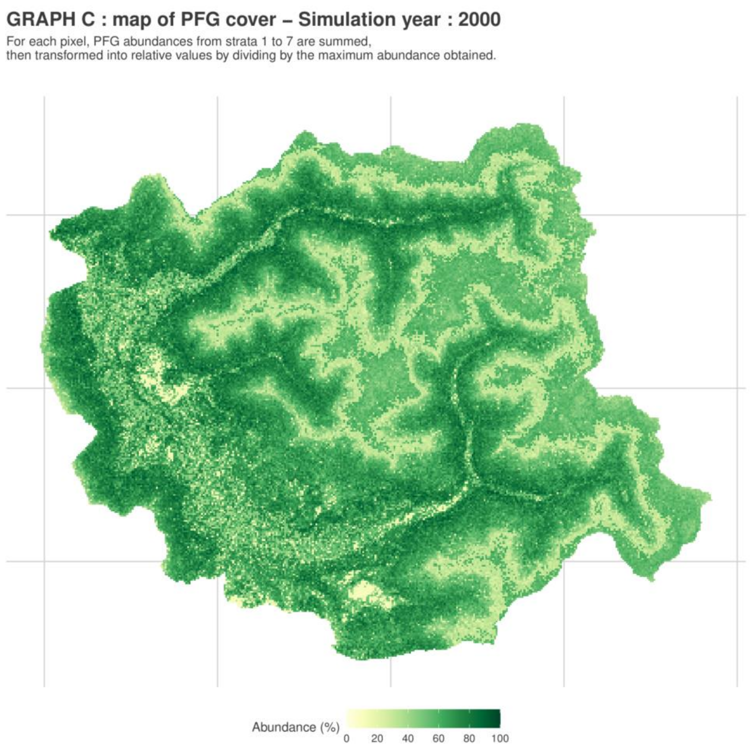
Finally, some simple representations can be obtained with the POST_FATE.graphic_mapPFG function, for the required years :

- The PFG relative cover : the total abundance of all PFG within each pixel, divided by the maximum total abundance found in a pixel over the whole simulation area ;
- The PFG richness : the sum of all PFG present in each pixel ;
- The resources community weighted mean (light and soil, if the corresponding modules were activated within the GlobalParameters file) ;
- The simulated resources (light and soil, if the corresponding modules were activated within the GlobalParameters file) ;

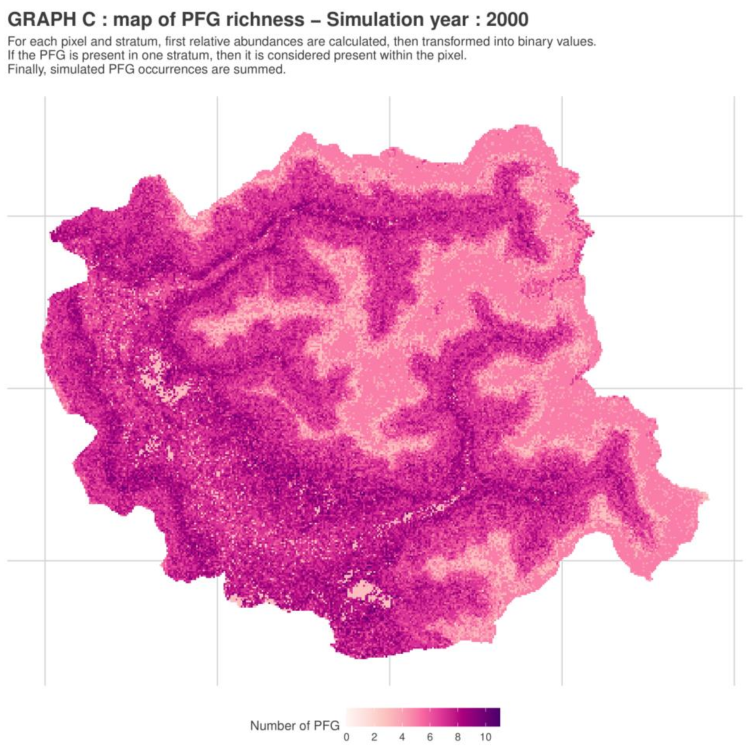
Specific height strata (minimum and maximum to be considered) can be selected to perform these graphics, to be able to focus on specific vegetation layer.

```
POST_FATE.graphic_mapPFG(name.simulation = simul.name
                         , file.simulParam = simul.param
                         , years = 2000
                         , opt.stratum_min = 1
                         , opt.stratum_max = 6)
```

PFG cover should be considered with precaution, especially if all height strata are selected, given the fact that herbaceous PFG (or at least the ones with `IMM_SIZE` close to 100%) can be more abundant than phanerophyte PFG, potentially misleading the interpretation of the PFG « cover ».


GRAPH C : map of PFG cover – Simulation year : 2000

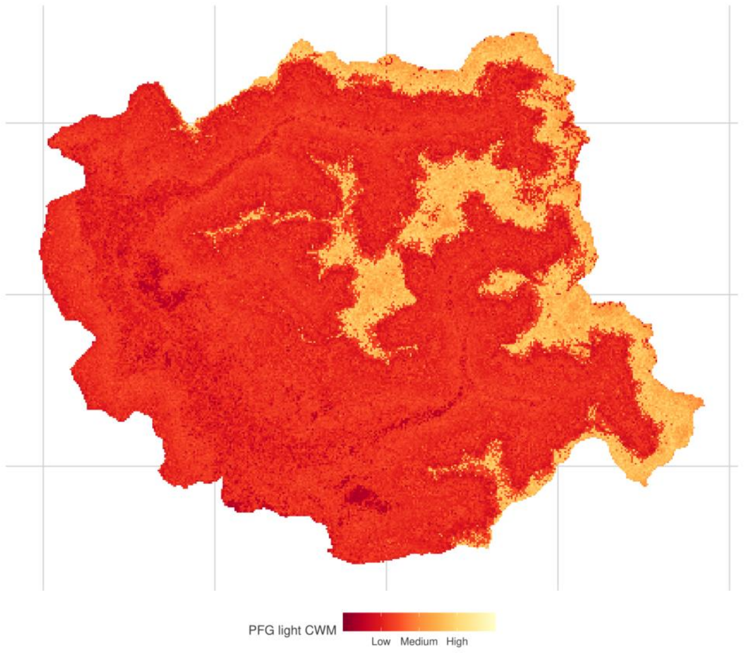
For each pixel, PFG abundances from strata 1 to 7 are summed, then transformed into relative values by dividing by the maximum abundance obtained.


GRAPH C : map of PFG richness – Simulation year : 2000

For each pixel and stratum, first relative abundances are calculated, then transformed into binary values. If the PFG is present in one stratum, then it is considered present within the pixel. Finally, simulated PFG occurrences are summed.

GRAPH C : map of light CWM – Simulation year : 2000

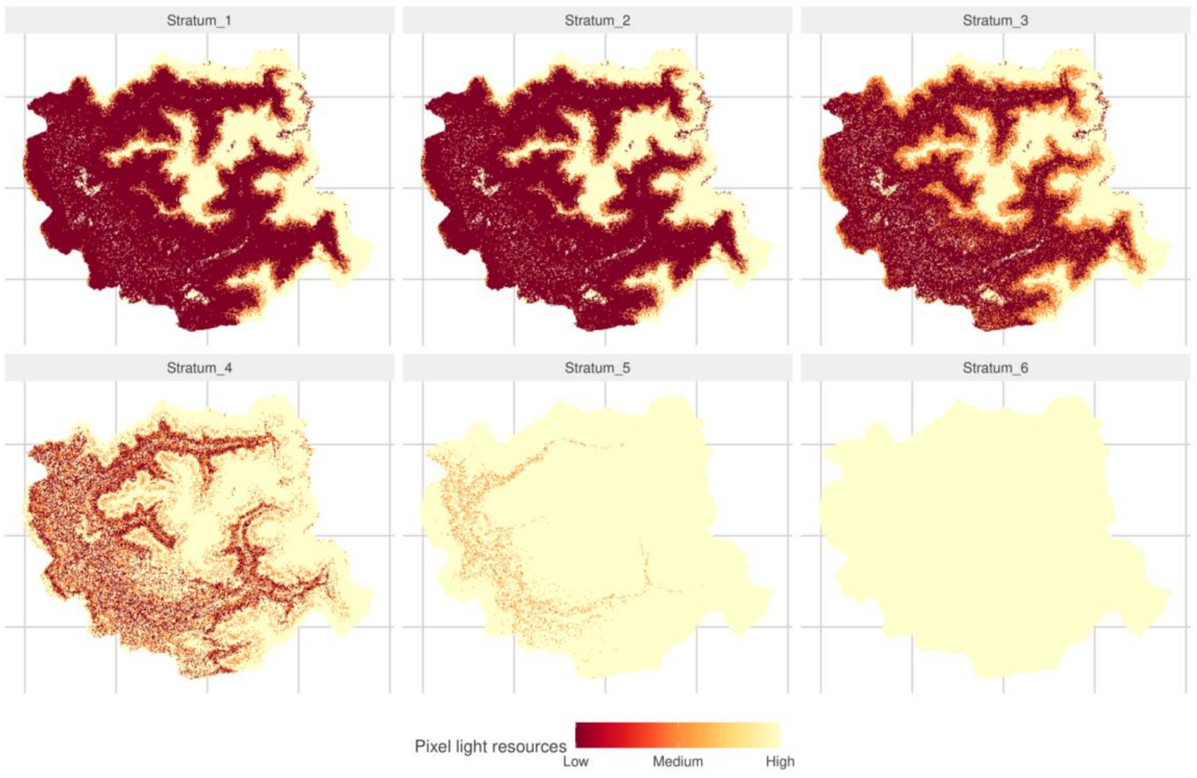
For each pixel, PFG abundances from strata 1 to 7 are summed,
then transformed into relative values by dividing by the maximum abundance obtained.
Community Weighted Mean is then calculated with observed values of light for each PFG.



This map does NOT represent the simulated light resources, but rather what you expect them to be, given the parametrization you have made of your PFG.

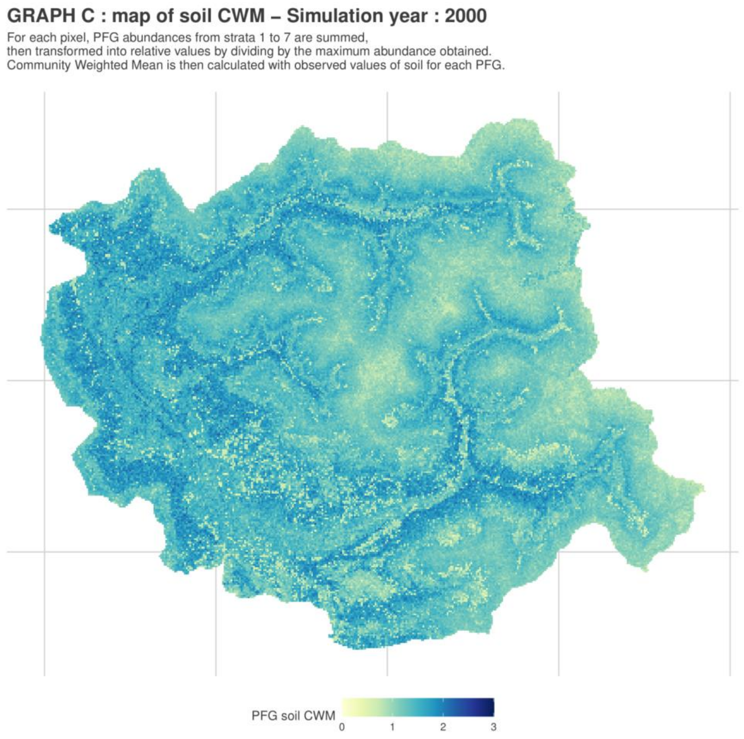
The maps below (one for each height considered height stratum) show the « real » light resources, meaning the one that were actually calculated and used during the simulation.

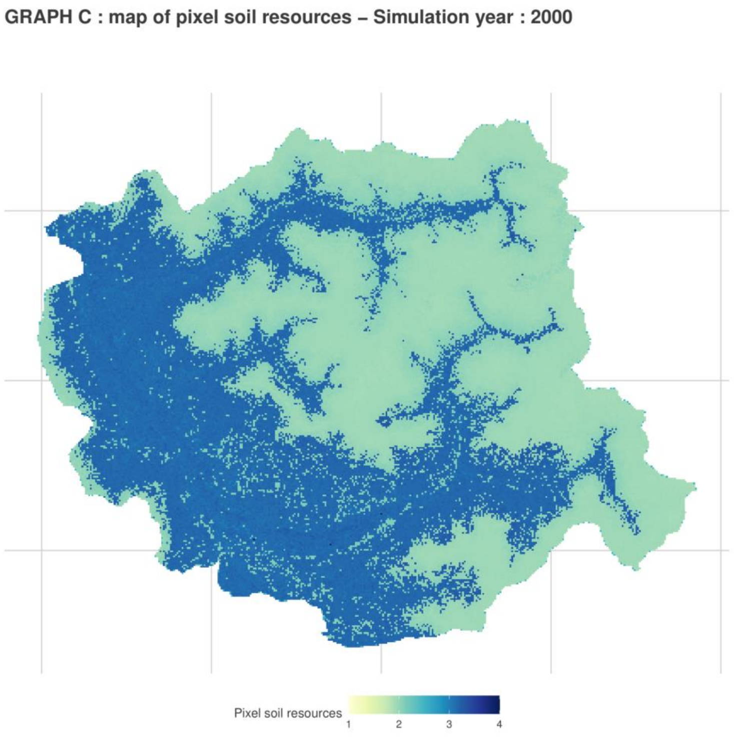
GRAPH C : map of pixel light resources – Simulation year : 2000



The same can be obtained for soil resources :

They can be seen as « *what you expect to have* », in terms of pixel soil composition, according to the PFG community present within the pixel ; and « *what is actually simulated* ».


GRAPH C : map of soil CWM – Simulation year : 2000
For each pixel, PFG abundances from strata 1 to 7 are summed,
then transformed into relative values by dividing by the maximum abundance obtained.
Community Weighted Mean is then calculated with observed values of soil for each PFG.
PFG soil CWM
0
1
2
3


GRAPH C : map of pixel soil resources – Simulation year : 2000
Pixel soil resources
1
2
3
4

Publications

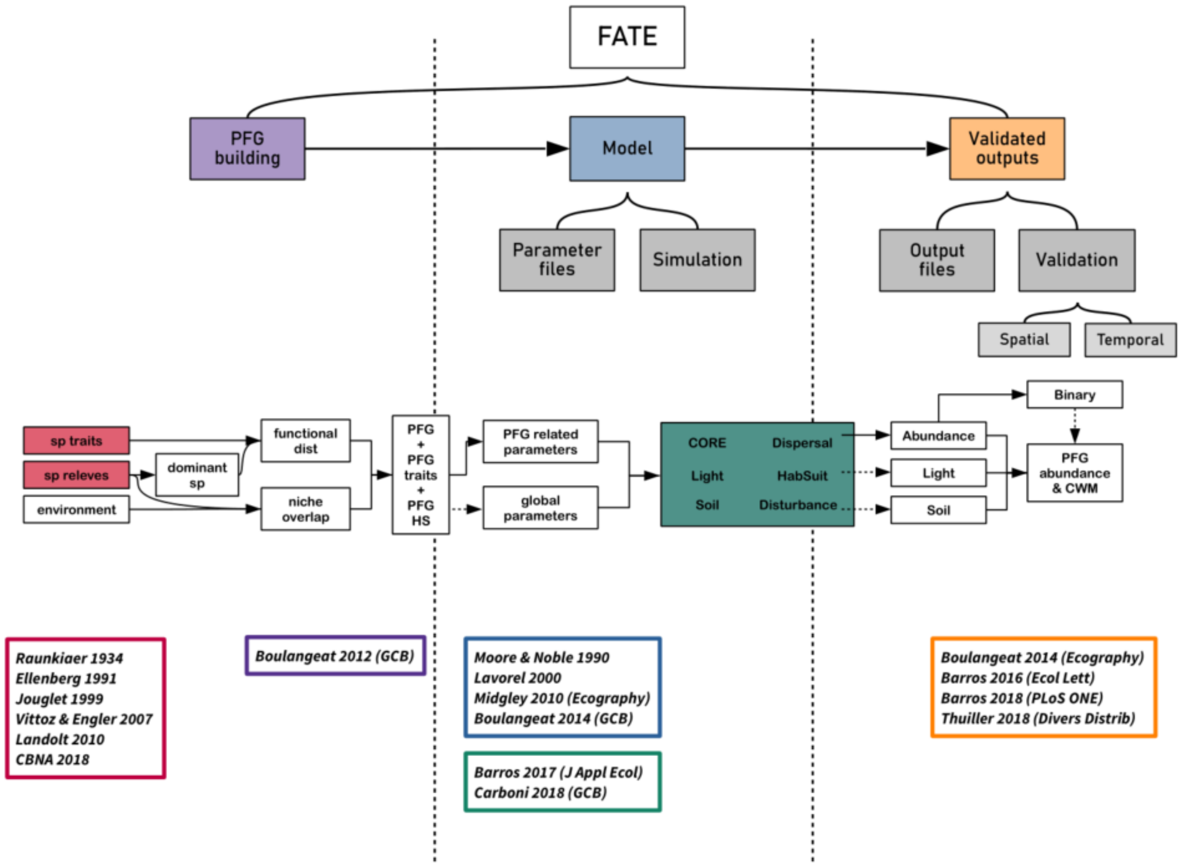


Figure 13: FATE publications workflow

Origins (FATE / LAMOS / BioMove)

- ⌘ Moore A. D. and Noble I. R. (1990), An individualistic model of vegetation stand dynamics. https://doi.org/10.1016/s0301-4797(05)80015-5
- ⌘ Lavorel S., Davies I. D. and Noble I. R. (2000). LAMOS: a landscape modelling shell. Landscape Fire Modeling-Challenges and Opportunities. Natural Resources Canada, Canadian Forest Service, Vancouver, BC, Canada, 25-28.
- ⌘ Midgley G. F., Davies I. D., Albert C. H., Altwegg R., Hannah L., Hughes G. O., O'Halloran L. R., Seo C., Thorne J. H. and Thuiller W. (2010), BioMove – an integrated platform simulating the dynamic response of species to environmental change. Ecography, 33: 612-616. https://doi.org/10.1111/j.1600-0587.2009.06000.x

FATE model

⌘ Boulangeat I., Philippe P., Abdulhak S., Douzet R., Garraud L., Lavergne S., Lavorel S., Van Es J., Vittoz P. and Thuiller W. (2012), Improving plant functional groups for dynamic models of biodiversity: at the crossroads between functional and community ecology. Glob Change Biol, 18: 3464-3475. https://doi.org/10.1111/j.1365-2486.2012.02783.x

⌘ Boulangeat I., Georges D. and Thuiller W. (2014), FATE-HD: a spatially and temporally explicit integrated model for predicting vegetation structure and diversity at regional scale. Glob Change Biol, 20: 2368-2378. https://doi.org/10.1111/gcb.12466

Applications

⌘ Boulangeat I., Georges D., Dentant C., Bonet R., Van Es J., Abdulhak S., Zimmermann N. E. and Thuiller W. (2014), Anticipating the spatio-temporal response of plant diversity and vegetation structure to climate and land use change in a protected area. Ecography, 37: 1230-1239. https://doi.org/10.1111/ecog.00694

⌘ Barros C., Thuiller W., Georges D., Boulangeat I., Münkemüller T. and Bellwood D. (2016), N-dimensional hypervolumes to study stability of complex ecosystems. Ecol Lett, 19: 729-742. https://doi.org/10.1111/ele.12617

⌘ Barros C., Guéguen M., Douzet R., Carboni M., Boulangeat I., Zimmermann N. E., Münkemüller T., Thuiller W. and Mori A. (2017), Extreme climate events counteract the effects of climate and land-use changes in Alpine tree lines. J Appl Ecol, 54: 39-50. https://doi.org/10.1111/1365-2664.12742

⌘ Barros C., Thuiller W. and Münkemüller T. (2018), Drought effects on the stability of forest-grassland ecotones under gradual climate change. PLoS ONE 13(10): e0206138. https://doi.org/10.1371/journal.pone.0206138

⌘ Carboni M., Guéguen M., Barros C., Georges D., Boulangeat I., Douzet R., Klonner G., Van Kleunen M., Essl F., Bossdorf O., Haeuser E., Talluto M. V., Moser D., Block S., Conti L., Dullinger I., Münkemüller T. and Thuiller W. (2018). Simulating plant invasion dynamics in mountain ecosystems under global change scenarios. Glob Change Biol, 24:e289–e302. https://doi.org/10.1111/gcb.13879

⌘ Thuiller W, Guéguen M, Bison M, et al (2018). Combining point-process and landscape vegetation models to predict large herbivore distributions in space and time—A case study of Rupicapra rupicapra. Divers Distrib, 24:352–362. https://doi.org/10.1111/ddi.12684

Supplementary material : COMPLETE EXAMPLE : Champsaur

a. Building Plant Functional Group

```
library(RFate)
Champsaur_PFG = .loadData("Champsaur_PFG", "RData")


###################################################################################################
## DOMINANT SPECIES
###################################################################################################
## Species observations
tab.occ = Champsaur_PFG$sp.observations
str(tab.occ)

## Run selection ---------------------------------------------------------------------------------
sp.SELECT = PRE_FATE.selectDominant(mat.observations = tab.occ[, c("sites", "species", "abund")]
                                    , doRuleA = TRUE
                                    , rule.A1 = 10
                                    , rule.A2_quantile = 0.88
                                    , doRuleB = TRUE
                                    , rule.B1_percentage = 0.25
                                    , rule.B1_number = 10
                                    , rule.B2 = 0.5
                                    , doRuleC = FALSE
                                    , opt.doRobustness = TRUE
                                    , opt.robustness_percent = seq(0.1, 0.9, 0.1)
                                    , opt.robustness_rep = 10
                                    , opt.doSitesSpecies = TRUE
                                    , opt.doPlot = TRUE)

## Explore results
names(sp.SELECT)
str(sp.SELECT[1:5])
str(sp.SELECT$tab.rules)
plot(sp.SELECT$plot.A)
plot(sp.SELECT$plot.B$abs)
plot(sp.SELECT$plot.B$rel)
plot(sp.SELECT$plot.pco$Axis1_Axis2)
# plot(sp.SELECT$plot.pco$Axis1_Axis3)
str(sp.SELECT$tab.robustness)
names(sp.SELECT$plot.robustness)
plot(sp.SELECT$plot.robustness$`All dataset`)


## Prepare data to calculate pairwise species distance -------------------------------------------

# ## Calculate mat.overlap matrix
# ## Add absences in community sites
# tab.dom.PA = sp.SELECT$tab.dom.PA
# for (si in sites$sites[which(sites$TYPE == "COMMUNITY")])
# {
#   ind = which(rownames(tab.dom.PA) == si)
#   tab.dom.PA[ind, which(is.na(tab.dom.PA[ind, ]))] = 0
# }
#
# ## Prepare environmental (and traits) table
# tab.env = Champsaur_PFG$tab.env
# tmp.traits = Champsaur_PFG$sp.traits[, c("species", "HEIGHT", "HEIGHT_log")]
# sp.DIST = PRE_FATE.speciesDistance(mat.traits = tmp.traits
#                                     , mat.overlap.option = "PCA"
#                                     , mat.overlap.object = list(tab.dom.PA, tab.env))
# mat.overlap = sp.DIST$mat.OVERLAP
# (mat.overlap[1:5, 1:5])
```

```
## Transform dissimilarity matrices into similarity distances
mat.habitat = Champsaur_PFG$mat.habitat
mat.overlap = Champsaur_PFG$mat.overlap

mat.habitat = 1 - mat.habitat
mat.overlap = 1 - mat.overlap


## Load selected traits for each group
tab.traits.P = Champsaur_PFG$sp.traits.P
tab.traits.C = Champsaur_PFG$sp.traits.C
tab.traits.H = Champsaur_PFG$sp.traits.H

str(tab.traits.P)
str(tab.traits.C)
str(tab.traits.H)



###################################################################################################
## PHANEROPHYTE
###################################################################################################

## Calculate pairwise species distance ------------------------------------------------------------
sp.DIST.P = PRE_FATE.speciesDistance(mat.traits = tab.traits.P
                                     , mat.overlap.option = "dist"
                                     , mat.overlap.object = mat.habitat
                                     , opt.weights = c(0.5, 0.5)
                                     , opt.maxPercent.NA = 0.25
                                     , opt.maxPercent.similarSpecies = 0.5
                                     , opt.min.sd = 0.5)
str(sp.DIST.P)
{
  require(foreach)
  require(ggplot2)
  require(ggdendro)
  pp = foreach(x = names(sp.DIST)) %do%
  {
    hc = hclust(sp.DIST[[x]])
    pp = ggdendrogram(hc, rotate = TRUE) +
      labs(title = paste0("Hierarchical clustering based on species distance "
                          , ifelse(length(names(sp.DIST)) > 1
                          , paste0("(group ", x, ")")
                          , "")))
    return(pp)
  }
  plot(pp[[1]])
  plot(pp[[2]])
  plot(pp[[3]])
}

## Build clusters and choose final groups number --------------------------------------------------
sp.CLUST1.P = PRE_FATE.speciesClustering_step1(sp.DIST.P$mat.ALL)
sp.CLUST2.P = PRE_FATE.speciesClustering_step2(clust.dendrograms = sp.CLUST1.P$clust.dendrograms
                                               , no.clusters = 5
                                               , mat.species.DIST = sp.DIST.P$mat.ALL)
names(sp.CLUST2.P)
str(sp.CLUST2.P$determ.sp)
str(sp.CLUST2.P$determ.all)
plot(sp.DETERM$plot.distance)
plot(sp.DETERM$plot.PCO$Phanerophyte)
```

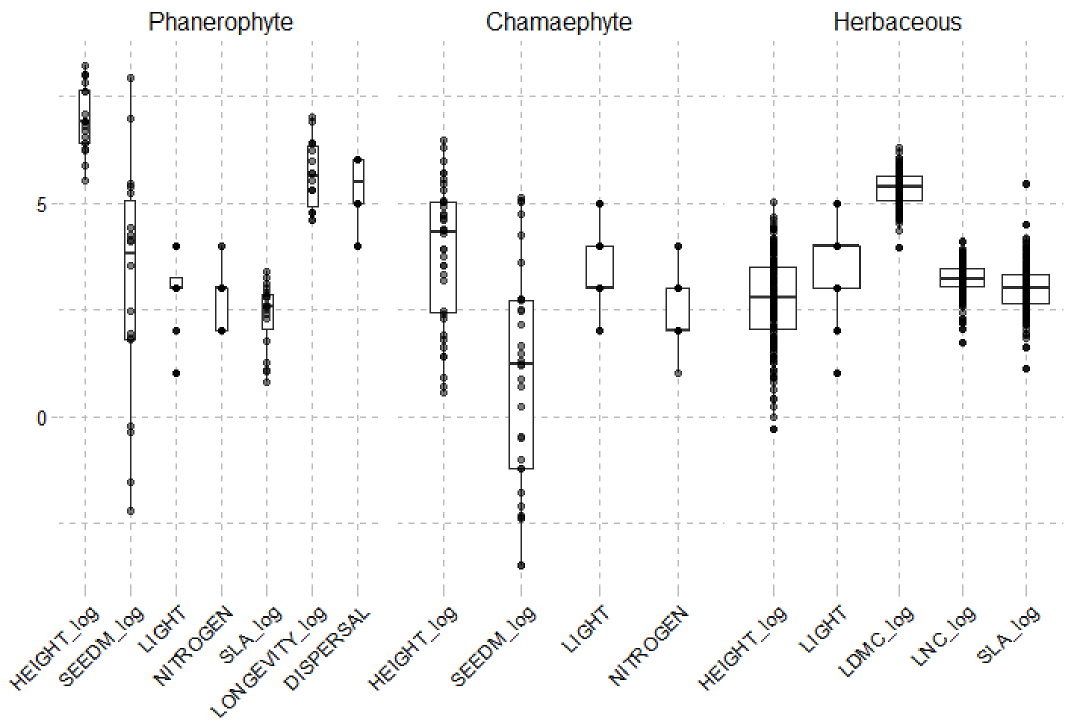
```
   ###################################################################################################
   ## CHAMAEPHYTE
   ###################################################################################################

   ## Combine distances of habitat preferences and niche overlap
   ## to have one matrix to reflect the species niches
   wei.habi = 0.5
   wei.over = 0.5
   ind.sp = intersect(colnames(mat.overlap), colnames(mat.habitat))
   mat.env  =  (wei.habi  *  mat.habitat[ind.sp,  ind.sp]  +  wei.over  *  mat.overlap[ind.sp,  ind.sp])  /
(wei.habi + wei.over)

   (mat.habitat[ind.sp, ind.sp][1:5, 1:5])
   (mat.overlap[ind.sp, ind.sp][1:5, 1:5])
   (mat.env[1:5, 1:5])

   ## Calculate pairwise species distance -------------------------------------------------------------
   sp.DIST.C = PRE_FATE.speciesDistance(mat.traits = tab.traits.C
                                       , mat.overlap.option = "dist"
                                       , mat.overlap.object = mat.env
                                       , opt.weights = c(0.4, 0.6)
                                       , opt.maxPercent.NA = 0.25
                                       , opt.maxPercent.similarSpecies = 0.5
                                       , opt.min.sd = 0.5)

   ## Build clusters and choose final groups number --------------------------------------------------
   sp.CLUST1.C = PRE_FATE.speciesClustering_step1(sp.DIST.C$mat.ALL)
   sp.CLUST2.C = PRE_FATE.speciesClustering_step2(clust.dendrograms = sp.CLUST1.C$clust.dendrograms
                                                 , no.clusters = 4
                                                 , mat.species.DIST = sp.DIST.C$mat.ALL)



   ###################################################################################################
   ## HERBACEOUS
   ###################################################################################################

   ## Rearrange data (more difficult to distinguish groups among herbaceous species) -----------------
   ## Separate some species quite different from the rest
   SP_wetlands = c("15735", "15211", "10429", "17167", "40501"
                   , "16522", "40514", "14782", "40445", "14316")
   SP_outliers = c("11223", "5445", "14024", "11035", "10477")
   tab.traits.H = tab.traits.H[-which(tab.traits.H$species %in% SP_wetlands), ]
   tab.traits.H = tab.traits.H[-which(tab.traits.H$species %in% SP_outliers), ]

   ## Put more weights on overlap matrix
   wei.habi = 0.4
   wei.over = 0.6
   ind.sp = intersect(colnames(mat.overlap), colnames(mat.habitat))
   mat.env  =  (wei.habi  *  mat.habitat[ind.sp,  ind.sp]  +  wei.over  *  mat.overlap[ind.sp,  ind.sp])  /
(wei.habi + wei.over)


   ## Calculate pairwise species distance -------------------------------------------------------------
   sp.DIST.H = PRE_FATE.speciesDistance(mat.traits = tab.traits.H
                                       , mat.overlap.option = "dist"
                                       , mat.overlap.object = mat.env
                                       , opt.weights = c(0, 1)
                                       , opt.maxPercent.NA = 0.25
                                       , opt.maxPercent.similarSpecies = 0.5
                                       , opt.min.sd = 0.5)


   ## Build clusters and choose final groups number --------------------------------------------------
   sp.CLUST1.H = PRE_FATE.speciesClustering_step1(sp.DIST.H$mat.ALL)
   sp.CLUST2.H = PRE_FATE.speciesClustering_step2(clust.dendrograms = sp.CLUST1.H$clust.dendrograms
                                                 , no.clusters = 7
                                                 , mat.species.DIST = sp.DIST.H$mat.ALL)

   ## Groups H6 (adventices) and H7 (outlier species) will be removed
   ## SP_wetlands will become PFG H6
```

```
##############################################################################################
## PLANT FUNCTIONAL GROUPS
##############################################################################################

tab.PFG = Champsaur_PFG$PFG.species
tab.PFG = tab.PFG[which(tab.PFG$DETERMINANT == "TRUE"), ]

tab.traits = Champsaur_PFG$sp.traits
tab.summary = tab.traits[, c("species", "MATURITY", "LONGEVITY", "HEIGHT", "LIGHT"
                             , "DISPERSAL", "NITROGEN", "NITROGEN_TOLERANCE", "LDMC", "LNC")]
colnames(tab.summary) = c("species", "maturity", "longevity", "height", "light"
                          , "dispersal", "soil_contrib", "soil_tolerance", "LDMC", "LNC")
tab.summary$soil_contrib = as.numeric(tab.summary$soil_contrib)
tab.summary$soil_tolerance = ifelse(tab.summary$soil_tolerance == 1, 0.5, 1)
tab.summary = merge(tab.PFG[, c("PFG", "species")], tab.summary, by = "species", all.x = TRUE)
tab.summary$soil_tolerance[which(tab.summary$PFG %in% c("H2", "P2") &
                                 tab.summary$soil_tolerance == 0.5)] = 1
head(tab.summary)

load("BUILDPFG_tab.dom.PA.RData")
tab.dom.PA = as.data.frame(tab.dom.PA)
tab.dom.PA = tab.dom.PA[, which(colnames(tab.dom.PA) %in% tab.summary$species)]

## Calculate trait values per PFG -----------------------------------------------------------

PFG.traits = PRE_FATE.speciesClustering_step3(mat.traits = tab.summary, opt.mat.PA = tab.dom.PA)
```

Traits used to build functional groups should reflect the community strategies that are taken into account in FATE, such as :

- life cycle (through maturity, longevity...)
- strategy for dispersal
- strategy for light (through height, light preference...)
- strategy for soil (through LNC, nitrogen, soil preference...)
- habitat and overall strategy (through LDMC, SLA, CSR strategy...)

They should be standardized (scale, log-transformation...) to be sure that they will all have similar importance. Traits selected were the ones with enough values, and enough variation.



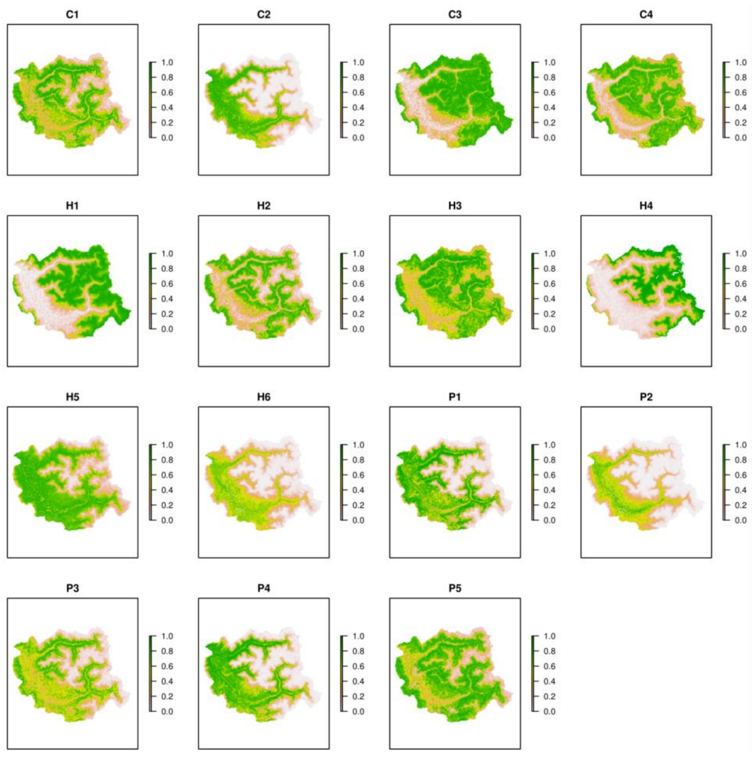
A brief description of the Plant Functional Groups obtained for the Champsaur dataset is presented below. Among selected dominant species, ~~some species~~ have been pointed out as *Not determinant* by the PRE_FATE.speciesClustering_step2 function (difference between dominant and determinant species is explained in the *Details* section of this function).

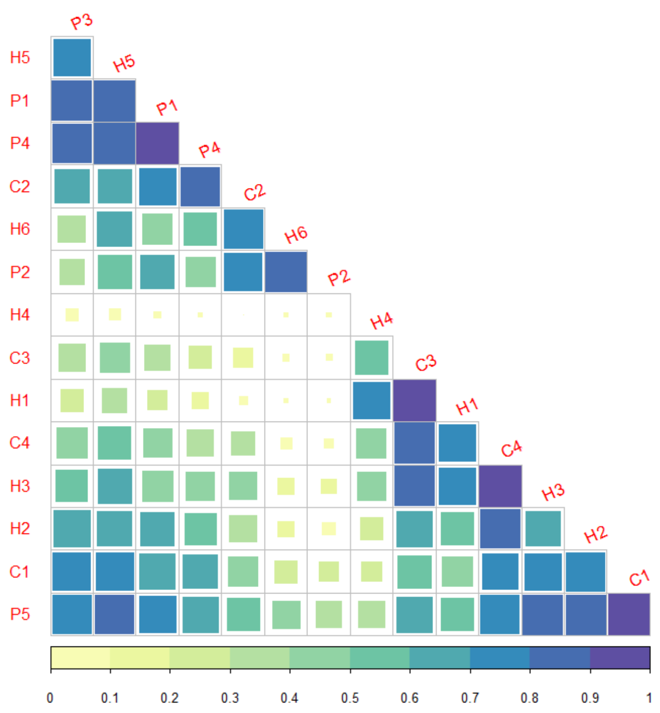
GROUP	Habitat distance	Overlap distance	Traits distance	Traits used ?	PFG	Brief description	Light preference	Soil preference	Dominant / determinant species
PHANEROPHYTES	100% of niche distance (50% in total)	~~0% of niche distance (0% in total)~~	50% in total	Longevity (log), Height (log), Seed mass (log), SLA (log), Dispersal, Light, Nitrogen	P1	Mixed forest	undergrowth (corrected)	moist / medium infertile-fertile	*Abies alba, Acer campestre, Acer pseudoplatanus, Fagus sylvatica, Picea abies*
					P2	Black poplar grove	semi-shade	very moist / fertile	*Alnus incana, Populus nigra*
					P3	Deciduous montane forest	pioneer	indifferent / medium infertile	*Betula pendula, Populus tremula*
					P4	Southern chestnut grove	ubiquist	moderately dry / infertile	*Castanea sativa, Corylus avellana, Fraxinus excelsior, Prunus avium, Quercus pubescens, Sorbus aria, Sorbus mougeotii*
					P5	Alpine forest	pioneer	moderately moist / infertile	*Larix decidua, Pinus cembra, Pinus sylvestris, Pinus uncinata*
CHAMAEPHYTES	20% of niche distance (12% in total)	80% of niche distance (48% in total)	40% in total	Height (log), Light, Nitrogen, LNC (log)	C1	Bearberry heaths	undergrowth	fresh / infertile	*Amelanchier ovalis, Aruncus dioicus, Berberis vulgaris, Calluna vulgaris, Clematis alpina, ~~Daphne mezereum~~, Juniperus communis, Rhamnus alpina, Rosa pendulina, Vaccinium vitis-idaea*
					C2	Hedges, edges, open woods	semi-shade	fresh / medium fertile	*Crataegus monogyna, ~~Hippophae rhamnoides subsp. fluviatilis~~, Lavandula angustifolia subsp. angustifolia, Ligustrum vulgare, Lonicera xylosteum, Prunus spinosa, Ribes uva-crispa, Rosa canina, Rubus caesius, Salix caprea, Salix purpurea, Viburnum lantana*
					C3	Snowbed heaths	full light	moist / infertile	*Dryas octopetala, Empetrum nigrum subsp. hermaphroditum, Globularia cordifolia, Juniperus sibirica, Salix glaucosericea, Salix herbacea, Salix reticulata, Salix retusa, Vaccinium uliginosum subsp. microphyllum*
					C4	Subalpine and boreal heatlands	semi-shade	moderately moist / infertile	*Arctostaphylos uva-ursi, Juniperus sabina, Rhododendron ferrugineum, Rubus idaeus, Vaccinium myrtillus*
HERBACEOUS	40% of niche distance (40% in total)	60% of niche distance (60% in total)	~~*0%*~~	~~*LDMC (log), Height (log), SLA (log), Light, LNC (log)*~~	H1	Alpine lean grasslands	full light	medium infertile	*Alchemilla alpina, Alchemilla transiens, Alopecurus alpinus, Antennaria dioica, Anthoxanthum odoratum subsp. nipponicum, Astrantia minor, Bartsia alpina, Carduus defloratus subsp. defloratus, Carex sempervirens subsp. sempervirens, Cerastium arvense subsp. strictum, Coincya richeri, Euphrasia minima, Festuca violacea, Geum montanum, Homogyne alpina, Juncus trifidus, ~~Lotus alpinus~~, Luzula nutans, Nardus stricta, Plantago alpina,Polygonum viviparum, Potentilla crantzii, Potentilla grandiflora, Primula hirsuta, Sagina glabra, Saxifraga aizoides,~~Sedum anacampseros~~, ~~Sibbaldia procumbens~~, Silene rupestris, Soldanella alpina, Trifolium alpinum, Trifolium thalii, Veronica fruticans, Viola calcarata L.*
					H2	Montane, fresh undergrowth	undergrowth	fresh / medium fertile	*~~Aconitum anthora~~, Asphodelus albus subsp. delphinensis, Calamintha grandiflora, Campanula rhomboidalis, Chaerophyllum villarsii, Digitalis grandiflora, ~~Dryopteris filix-mas~~, Gentiana lutea, Geranium sylvaticum, Hieracium prenanthoides, Imperatoria ostruthium, Lilium bulbiferum var. croceum, Lilium martagon, Luzula nivea, Oxalis acetosella, Prenanthes purpurea, Rumex pseudalpinus*
					H3	Subalpine grasslands	pioneer	moist / medium infertile-fertile	*Achnatherum calamagrostis, Anthoxanthum odoratum, Asplenium septentrionale, Bunium bulbocastanum, Campanula rotundifolia, Carlina acanthifolia subsp. acanthifolia, Carlina acaulis subsp. caulescens, Centranthus angustifolius, Cryptogramma crispa, Cystopteris fragilis, Deschampsia flexuosa, Epilobium dodonaei Vill. subsp. fleischeri, Euphorbia cyparissias, Festuca acuminata, Festuca laevigata, Galium pumilum, Helianthemum nummularium, Helictotrichon sempervirens, Laserpitium gallicum, Laserpitium latifolium, Minuartia laricifolia subsp. laricifolia, Phyteuma betonicifolium, Polystichum lonchitis, Primula veris subsp. columnae, Rumex scutatus, ~~Saxifraga paniculata~~, Sempervivum arachnoideum, Sempervivum tectorum, Sesleria caerulea, Silene nutans subsp. nutans, Stachys recta, Stipa eriocaulis, Teucrium chamaedrys, Trifolium badium, Valeriana montana, Viola biflora*
					H4	Ridge, barked alpine grasslands	full light	dry / infertile	*Achillea nana, Androsace pubescens, Arabis alpina, Berardia subacaulis, Cacalia leucophylla, Campanula cenisia, Campanula cochleariifolia, Campanula scheuchzeri subsp. scheuchzeri, Cardamine resedifolia, Carex curvula subsp. rosae, Cirsium spinosissimum, Doronicum grandiflorum, Festuca halleri, Galium pseudohelveticum, Geum reptans, Helictotrichon sedenense subsp. sedenense, Kobresia myosuroides, Leucanthemopsis alpina, Linaria alpina subsp. alpina, ~~Myosotis alpestris~~, Noccaea rotundifolia, Phyteuma hemisphaericum, Poa alpina, Poa cenisia, Ranunculus glacialis, Saxifraga oppositifolia, ~~Senecio incanus~~, Silene acaulis, Trisetum distichophyllum*
					H5	Montane to early subalpine species	semi-shade	medium fertile	*Achillea millefolium, Agrostis capillaris, Arrhenatherum elatius subsp. elatius, Artemisia absinthium, Brachypodium rupestre, Briza media, Bupleurum falcatum, Cirsium arvense, Clinopodium vulgare, Dactylis glomerata subsp. glomerata, Epilobium angustifolium, Euphorbia dulcis subsp. incompta, ~~Fragaria vesca~~, Galium aparine, Galium mollugo, ~~Genista sagittalis~~, Geranium robertianum, Geum urbanum, Gymnadenia conopsea, Helleborus foetidus, Hypericum perforatum, Lathyrus pratensis, ~~Leontodon hispidus~~, Listera ovata, Lotus corniculatus, Melica nutans, Mercurialis perennis, Mycelis muralis, Phyteuma spicatum, Plantago lanceolata, Platanthera bifolia, Poa nemoralis, Potentilla neumanniana, Rhinanthus alectorolophus, Senecio ovatus subsp. alpestris, Silene vulgaris, Trifolium montanum, Trifolium pratense, Trifolium repens, Trisetum flavescens subsp. flavescens, ~~Tussilago farfara~~, Veronica chamaedrys*
					H6	Wetlands, humid grasslands	semi-shade	very moist, flooded / medium fertile	*Adonis aestivalis, Allium ursinum, Calamagrostis pseudophragmites, Carex elata, Dactylorhiza fuchsii, Gagea fragifera, Molinia caerulea, Platanthera chlorantha, ~~Trichophorum cespitosum subsp. cespitosum~~, Typha latifolia*

b. Creating FATE parameter files

Build PFG habitat suitability maps (`biomod2` package)

```
library(RFate)
Champsaur_params = .loadData("Champsaur_params", "RData")


###################################################################################################
## BUILD PFG HABSUIT MAPS (biomod2 package)
###################################################################################################

require(biomod2)

## Species observations
tab.occ = Champsaur_params$tab.occ
str(tab.occ)

## Sites environmental table
tab.env = Champsaur_params$tab.env
str(tab.env)

## Sites coordinates table
tab.xy = Champsaur_params$tab.xy
str(tab.xy)

## Raster stack for projection
stk.var = Champsaur_params$stk.var

## Run species distribution models ----------------------------------------------------------------

for(pfg in colnames(tab.occ))
{
  nrep = 5
  sp.name = pfg
  sp.occ = tab.occ[which(!is.na(tab.occ[, pfg])), pfg]
  sp.xy = tab.xy[which(!is.na(tab.occ[, pfg])), c("X", "Y")]
  sp.var = tab.env[which(!is.na(tab.occ[, pfg])), ]

  ########################################################################################
  ## BELOW, MOST CHANGES WILL BE FOR MODELS PARAMETERS OR ADAPT DATA
  ## ALL NEEDED DATA HAS BEEN PRESENTED PREVIOUSLY

  ## formating data in a biomod2 friendly way -----------------------------------
  bm.form <- BIOMOD_FormatingData(resp.var = as.matrix(sp.occ)
                                  , expl.var = sp.var
                                  , resp.xy = sp.xy
                                  , resp.name = sp.name)

  ## define models options -------------------------------------------------------
  bm.opt <- BIOMOD_ModelingOptions(GLM = list(type = "quadratic", interaction.level = 0, test = "AIC")
                                   , GAM = list(k = 3))

  bm.mod <- BIOMOD_Modeling(data = bm.form
                            , models = c('RF', 'GLM', 'GAM')
                            , models.options = bm.opt
                            , NbRunEval = nrep
                            , DataSplit = 70
                            , Prevalence = 0.5
                            , VarImport = 3
                            , models.eval.meth = c('TSS','ROC')
                            , do.full.models = FALSE
                            , modeling.id = 'mod1')
  ## run ensemble models ---------------------------------------------------------
  bm.em <- BIOMOD_EnsembleModeling(modeling.output = bm.mod
                                   , chosen.models = "all"
                                   , em.by = "all"
                                   , eval.metric = c('TSS')
                                   , eval.metric.quality.threshold = 0.4
                                   , models.eval.meth = c('TSS', 'ROC')
                                   , prob.mean = FALSE
                                   , prob.mean.weight = TRUE
                                   , prob.mean.weight.decay = 'proportional'
                                   , committee.averaging = TRUE
                                   , VarImport = 3)

  ## project ensemble models -----------------------------------------------------
  bm.ef <- BIOMOD_EnsembleForecasting(EM.output = bm.em
                                      , new.env = stk.var
                                      , output.format = ".img"
                                      , proj.name = "CURRENT_100m"
                                      , selected.models = "all"
                                      , binary.meth = c('TSS'))
}

## ---------------------------------------------------------------------------------------
## This code gives a quick and rough idea of the
## PFG communities that should be found within the simulation area
## For example here :
##  H6 is mainly found with : C2 and P2 (edges, very humid and fertile groups)
##  H5 is mainly found with : P1, P3 and P4 / H2, C1 and P5 (montane and subalpine forests)

# library(phyloclim)
# type.mod = "wmean" # "ca"
# list.fi = paste0(sort(colnames(tab.occ)), "/proj_CURRENT_100m/individual_projections/"
#          , sort(colnames(tab.occ)), "_EM", type.mod, "ByTSS_mergedAlgo_mergedRun_mergedData.img")
# stk.wmean = stack(list.fi)
# stk.wmean = stk.wmean / 1000
# names(stk.wmean) = sub("_.*", "", names(stk.wmean))
# stk.wmean[which(stk.wmean[] < 0)] = 0
# stk.wmean[which(stk.wmean[] > 1)] = 1
# HS.stk[] = ifelse(stk.wmean[] > 0.5, 1, 0)
# HS.list = lapply(1:nlayers(HS.stk), function(x) as(HS.stk[[x]], 'SpatialGridDataFrame'))
# nich = as.matrix(niche.overlap(HS.list))
# colnames(nich) = rownames(nich) = sapply(names(HS.stk), function(x) strsplit(x, "_")[[1]][2])
# library(corrplot)
# colo = colorRampPalette(c('#9e0142','#d53e4f','#f46d43','#fdae61','#fee08b','#ffffbf'
#                           ,'#e6f598','#abdda4','#66c2a5','#3288bd','#5e4fa2'))
# corrplot(nich, method = "square", type = "lower", diag = FALSE
#          , col = colo(20), cl.lim = c(0,1), order = "hclust"
#          , tl.srt = 25, tl.offset = 1)
```



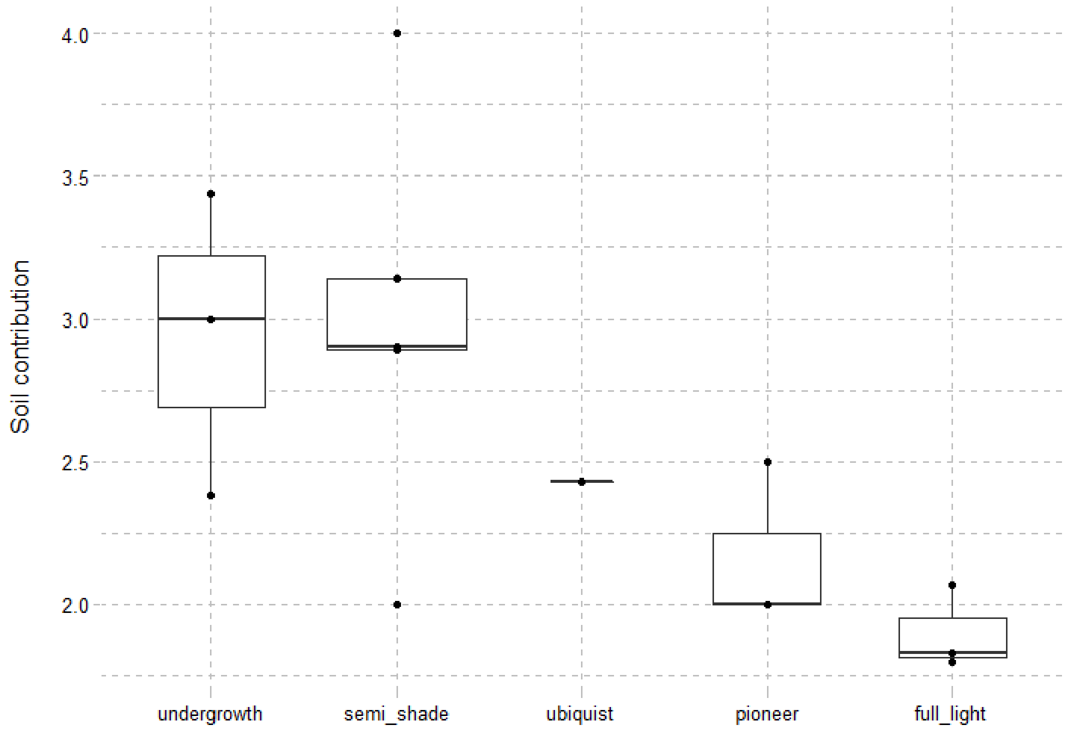


Create FATE simulation folder and parameter files

```
library(RFate)
Champsaur_params = .loadData("Champsaur_params", "RData")

###################################################################################################
## CREATE FATE PARAMETER FOLDER
###################################################################################################

PRE_FATE.skeletonDirectory(name.simulation = "FATE_Champsaur")


## Create PFG parameter files ---------------------------------------------------------------------

## SUCCESSION -------------------------------------------------------------------------------
PRE_FATE.params_PFGsuccession(name.simulation = "FATE_Champsaur"
                              , strata.limits = c(0, 20, 50, 150, 400, 1000, 2000)
                              , strata.limits_reduce = FALSE
                              , mat.PFG.succ = Champsaur_params$tab.SUCC)


## DISPERSAL --------------------------------------------------------------------------------
# ## Load example data
# Champsaur_PFG = .loadData("Champsaur_PFG", "RData")
#
# ## Build PFG traits for dispersal
# tab.traits = Champsaur_PFG$PFG.traits
# ## Dispersal values
# ##   = Short: 0.1-2m;    Medium: 40-100m;    Long: 400-500m
# ##   = Vittoz correspondance : 1-3: Short;    4-5: Medium;   6-7:Long
# corres = data.frame(dispersal = 1:7
#                     , d50 = c(0.1, 0.5, 2, 40, 100, 400, 500)
#                     , d99 = c(1, 5, 15, 150, 500, 1500, 5000)
#                     , ldd = c(1000, 1000, 1000, 5000, 5000, 10000, 10000))
# tab.traits$d50 = corres$d50[tab.traits$dispersal]
# tab.traits$d99 = corres$d99[tab.traits$dispersal]
# tab.traits$ldd = corres$ldd[tab.traits$dispersal]
# str(tab.traits)

PRE_FATE.params_PFGdispersal(name.simulation = "FATE_Champsaur"
                             , mat.PFG.disp = Champsaur_params$tab.DISP)


## LIGHT ------------------------------------------------------------------------------------
PRE_FATE.params_PFGlight(name.simulation = "FATE_Champsaur"
                         , mat.PFG.light = Champsaur_params$tab.LIGHT[, c("PFG", "type")]
                         , mat.PFG.tol = Champsaur_params$tab.LIGHT[, c("PFG", "strategy_tol")])
.setParam(params.lines = "FATE_Champsaur/DATA/PFGS/LIGHT/LIGHT_P1.txt"
          , flag = "LIGHT_TOL"
          , flag.split = " "
          , value = "1 1 0 1 1 1 1 1 1")


## SOIL -------------------------------------------------------------------------------------
PRE_FATE.params_PFGsoil(name.simulation = "FATE_Champsaur"
                        , mat.PFG.soil = Champsaur_params$tab.SOIL)



## Create simulation related parameter files -----------------------------------------------------

## SAVING YEARS -----------------------------------------------------------------------------
PRE_FATE.params_savingYears(name.simulation = "FATE_Champsaur"
                            , years.maps = seq(50, 2000, 50))
```

```
## GLOBAL PARAMS -----------------------------------------------------------------------
combi = expand.grid(doLight = c(FALSE, TRUE), doSoil = c(FALSE, TRUE))
for (ii in 1:nrow(combi))
{
  PRE_FATE.params_globalParameters(name.simulation = "FATE_Champsaur"
                                   , opt.saving_abund_PFG_stratum = TRUE
                                   , opt.saving_abund_PFG = TRUE
                                   , opt.saving_abund_stratum = FALSE
                                   , required.no_PFG = 15
                                   , required.no_strata = 7
                                   , required.simul_duration = 2000
                                   , required.seeding_duration = 1000
                                   , required.seeding_timestep = 1
                                   , required.seeding_input = 100
                                   , required.potential_fecundity = 1
                                   , required.max_abund_low = 1000
                                   , required.max_abund_medium = 2000
                                   , required.max_abund_high = 3000
                                   , doLight = combi$doLight[ii]
                                   , LIGHT.thresh_medium = 8000 #6000
                                   , LIGHT.thresh_low = 12000 #10000
                                   , LIGHT.saving = TRUE
                                   , doSoil = combi$doSoil[ii]
                                   , SOIL.init = 2.5
                                   , SOIL.retention = 0.8
                                   , SOIL.saving = TRUE
                                   , doDispersal = TRUE
                                   , DISPERSAL.mode = 1
                                   , DISPERSAL.saving = FALSE
                                   , doHabSuitability = TRUE
                                   , HABSUIT.mode = 1)
}
.setParam(params.lines = "FATE_Champsaur/DATA/GLOBAL_PARAMETERS/Global_parameters_V4.txt"
          , flag = "LIGHT_THRESH_MEDIUM"
          , flag.split = " "
          , value = "6000")
.setParam(params.lines = "FATE_Champsaur/DATA/GLOBAL_PARAMETERS/Global_parameters_V4.txt"
          , flag = "LIGHT_THRESH_LOW"
          , flag.split = " "
          , value = "10000")


## SIMUL_PARAM ---------------------------------------------------------------------------
writeRaster(Champsaur_params$stk.mask
            , filename = paste0("FATE_Champsaur/DATA/MASK/MASK_"
                                , names(Champsaur_params$stk.mask), ".tif")
            , bylayer = TRUE)
writeRaster(Champsaur_params$stk.wmean
            , filename = paste0("FATE_Champsaur/DATA/PFGS/HABSUIT/HS_"
                                , names(Champsaur_params$stk.wmean), "_0.tif")
            , bylayer = TRUE)

.adaptMaps(name.simulation = "FATE_Champsaur", extension.old = "tif", extension.new = "tif")

PRE_FATE.params_simulParameters(name.simulation = "FATE_Champsaur"
                                , name.MASK = "MASK_Champsaur.img")
```

```
## ---------------------------------------------------------------------------------------
## This code represents the repartition of interaction strategies for soil and light

# Champsaur_params = .loadData("Champsaur_params", "RData")
# tab = merge(Champsaur_params$tab.LIGHT, Champsaur_params$tab.SOIL, by = c("PFG", "type"))
# tab$strategy_tol = factor(tab$strategy_tol, c("undergrowth", "semi_shade"
#                                               , "ubiquist", "pioneer", "full_light"))
# library(ggplot2)
# library(ggthemes)
# ggplot(tab, aes(x = strategy_tol, y = soil_contrib)) +
#   geom_boxplot(varwidth = TRUE) +
#   labs(x = "", y = "Soil contribution\n") +
#   theme_pander()
```



c. Running a FATE simulation

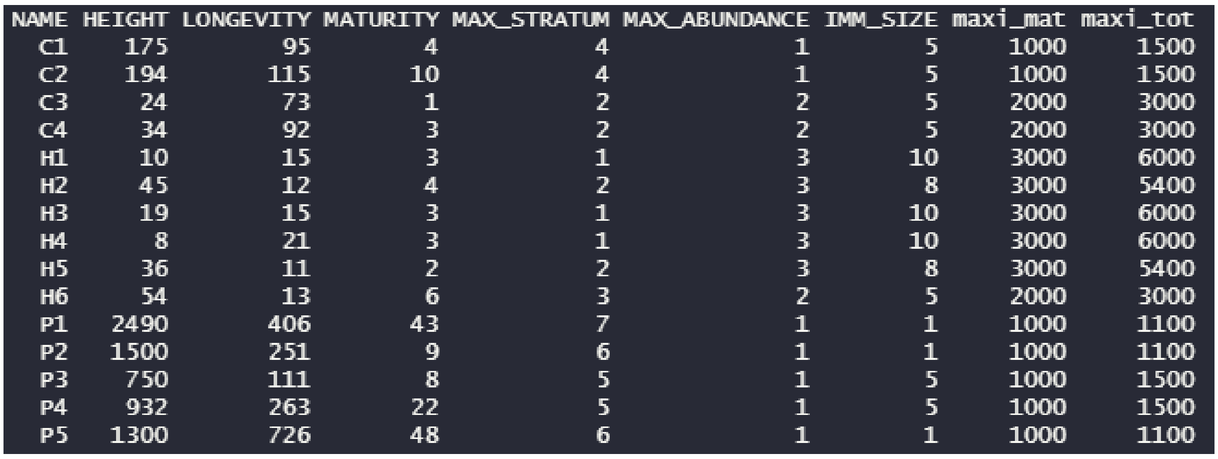
```
library(RFate)

## Select a parameter file
param = "Simul_parameters_V1.1.txt"
# param = "Simul_parameters_V2.1.txt"
# param = "Simul_parameters_V3.1.txt"
# param = "Simul_parameters_V4.1.txt"

## Run the simulation
## (here parallelized on 4 resources, and showing only warning and error messages)
FATE(simulParam = paste0("FATE_Champsaur/PARAM_SIMUL/", param)
     , no_CPU = 4
     , verboseLevel = 2)
```

Explore, understand and tune the parameters

When using the Light module, the `LIGHT_THRESH_MEDIUM` and `LIGHT_THRESH_LOW` parameters will depend on the PFG abundances within a pixel. *For example, a simulation in which a pixel will contain 5 or 6 PFG, each with a maximum total abundance around 5000, will have different parameter values than a simulation in which a pixel will contain 10 PFG with a maximum total abundance set to 1000.*

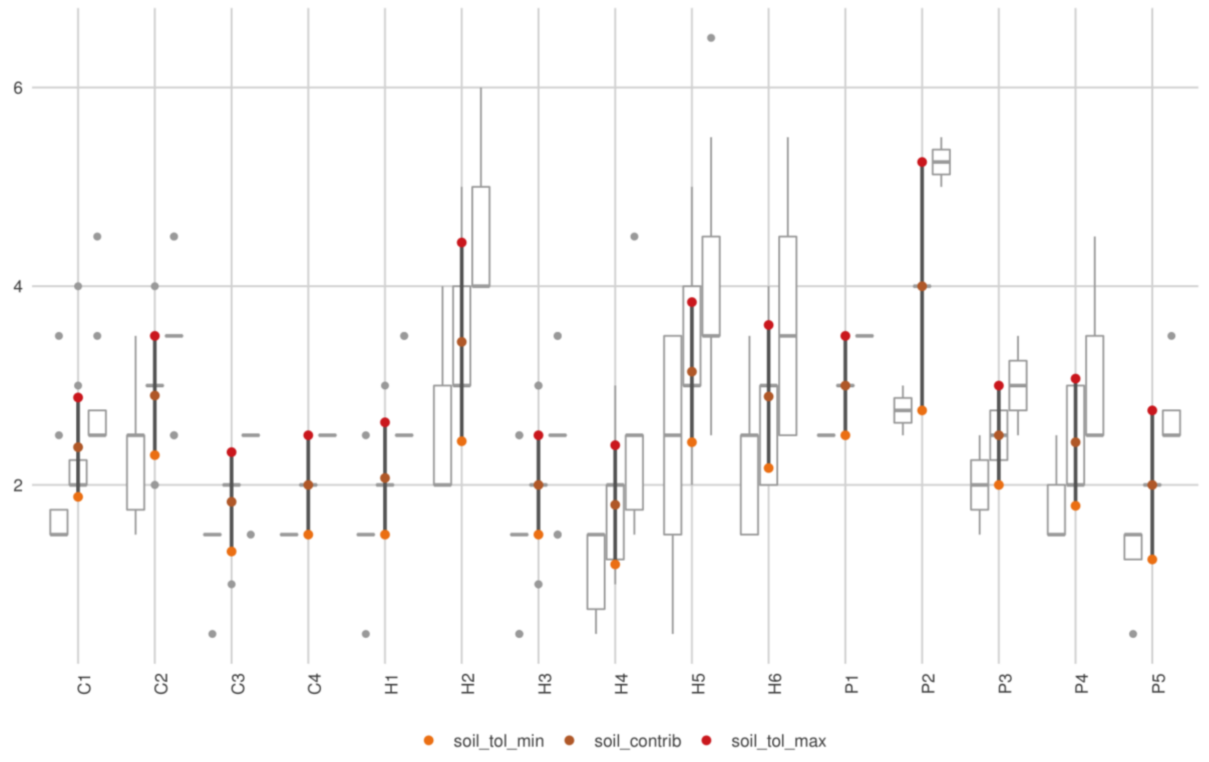
In the same way, these parameters can change depending on the module activated. *Here, the values of `LIGHT_THRESH_MEDIUM` and `LIGHT_THRESH_LOW` change between the simulations 2 (only Light) and 4 (Light + Soil).*

```
## ---------------------------------------------------------------------------------------
## This code calculates for each PFG its maximum total abundance :
##   MaxAbund * (1 + ImmSize)

# ma_low = .getParam(params.lines = "FATE_Champsaur/DATA/GLOBAL_PARAMETERS/Global_parameters_V1.txt"
#                    , flag = "MAX_ABUND_LOW", flag.split = " ", is.num = TRUE)
# ma_med = .getParam(params.lines = "FATE_Champsaur/DATA/GLOBAL_PARAMETERS/Global_parameters_V1.txt"
#                    , flag = "MAX_ABUND_MEDIUM", flag.split = " ", is.num = TRUE)
# ma_hig = .getParam(params.lines = "FATE_Champsaur/DATA/GLOBAL_PARAMETERS/Global_parameters_V1.txt"
#                    , flag = "MAX_ABUND_HIGH", flag.split = " ", is.num = TRUE)
# tab = as.data.frame(fread("FATE_Champsaur/DATA/PFGS/SUCC_COMPLETE_TABLE.csv"))
# tab = tab[, c("NAME", "HEIGHT", "LONGEVITY", "MATURITY"
#               , "MAX_STRATUM", "MAX_ABUNDANCE", "IMM_SIZE")]
# tab$maxi_mat = sapply(tab$MAX_ABUNDANCE
#                       , function(x) switch(x, "1" = ma_low, "2" = ma_med, "3" = ma_hig))
# tab$maxi_tot = tab$maxi_mat * (1 + tab$IMM_SIZE / 10)
# head(tab)
```



```
NAME HEIGHT LONGEVITY MATURITY MAX_STRATUM MAX_ABUNDANCE IMM_SIZE maxi_mat maxi_tot
  C1    175        95        4           4             1        5     1000     1500
  C2    194       115       10           4             1        5     1000     1500
  C3     24        73        1           2             2        5     2000     3000
  C4     34        92        3           2             2        5     2000     3000
  H1     10        15        3           1             3       10     3000     6000
  H2     45        12        4           2             3        8     3000     5400
  H3     19        15        3           1             3       10     3000     6000
  H4      8        21        3           1             3       10     3000     6000
  H5     36        11        2           2             3        8     3000     5400
  H6     54        13        6           3             2        5     2000     3000
  P1   2490       406       43           7             1        1     1000     1100
  P2   1500       251        9           6             1        1     1000     1100
  P3    750       111        8           5             1        5     1000     1500
  P4    932       263       22           5             1        5     1000     1500
  P5   1300       726       48           6             1        1     1000     1100
```

When using the Soil module, the ranges of soil tolerance (`SOIL_LOW` and `SOIL_HIGH`) of all PFG should overlap in a balanced way : not too much, otherwise no effect of the soil module will be seen, but enough to ensure PFG survival.



In any case, it is good to keep in mind that **it does not exist ONE good parametrization, but rather a gradient of parameter values** along which more or less effect will be seen.

For example, Light module can be activated, but if the thresholds are well above pixel abundances, it will have no impact on the communities. As the thresholds are lowered, the interaction for light will become increasingly important, leading at some point to the crash of some PFG.

d. Analyzing results

```
## .loadData("Champsaur_results", "7z") :
##    table outputs and graphic pdf files only


library(RFate)

## Select a parameter file
param = "Simul_parameters_V1.1.txt"
# param = "Simul_parameters_V2.1.txt"
# param = "Simul_parameters_V3.1.txt"
# param = "Simul_parameters_V4.1.txt"

simul.name = "FATE_Champsaur"
simul.param = paste0("FATE_Champsaur/PARAM_SIMUL/", param)


###################################################################################################
## TEMPORAL EVOLUTION
###################################################################################################

POST_FATE.temporalEvolution(name.simulation = simul.name
                            , file.simulParam = simul.param
                            , no_years = 40)

POST_FATE.graphic_evolutionCoverage(name.simulation = simul.name
                                    , file.simulParam = simul.param)

POST_FATE.graphic_evolutionPixels(name.simulation = simul.name
                                  , file.simulParam = simul.param)

POST_FATE.graphic_evolutionStability(name.simulation = simul.name
                                     , file.simulParam = simul.param)


###################################################################################################
## RELATIVE ABUNDANCE / BINARY MAPS & VALIDATION
###################################################################################################

Champsaur_params = .loadData("Champsaur_params", "RData")

## PFG observations
tab.occ = Champsaur_params$tab.occ
tab.xy = Champsaur_params$tab.xy

## Merge observations and sites coordinates
tab.obs = merge(tab.occ, tab.xy, by = "row.names")
tab.obs = melt(tab.obs, id.vars = c("Row.names", "X", "Y"))
colnames(tab.obs) = c("sites", "X", "Y", "PFG", "obs")
tab.obs = na.exclude(tab.obs)
head(tab.obs)


POST_FATE.relativeAbund(name.simulation = simul.name
                        , file.simulParam = simul.param
                        , years = 2000)

POST_FATE.graphic_validationStatistics(name.simulation = simul.name
                                       , file.simulParam = simul.param
                                       , years = 2000
                                       , mat.PFG.obs = tab.obs[, c("PFG", "X", "Y", "obs")])
```

```
POST_FATE.binaryMaps(name.simulation = simul.name
                     , file.simulParam = simul.param
                     , years = 2000
                     , method = 1
                     , method1.threshold = 0.05)

POST_FATE.graphic_mapPFGvsHS(name.simulation = simul.name
                             , file.simulParam = simul.param
                             , years = 2000)


###################################################################################################
## VISUALIZATION OF OUTPUTS
###################################################################################################

POST_FATE.graphic_mapPFG(name.simulation = simul.name
                         , file.simulParam = simul.param
                         , years = 2000
                         , opt.stratum_min = 1
                         , opt.stratum_max = 6)
```

The evolutionCoverage graphic allows a rapid evaluation of the presence and the importance of PFG at the end of the simulation.

The mapPFG graphics give spatial representations.

The evolutionPixels can help understand the dynamics between different modules when selecting the same pixels.