RFate

Modelling vegetation spatially and temporally, through plant functional groups

Version 1.0

User Manual

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

February 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)

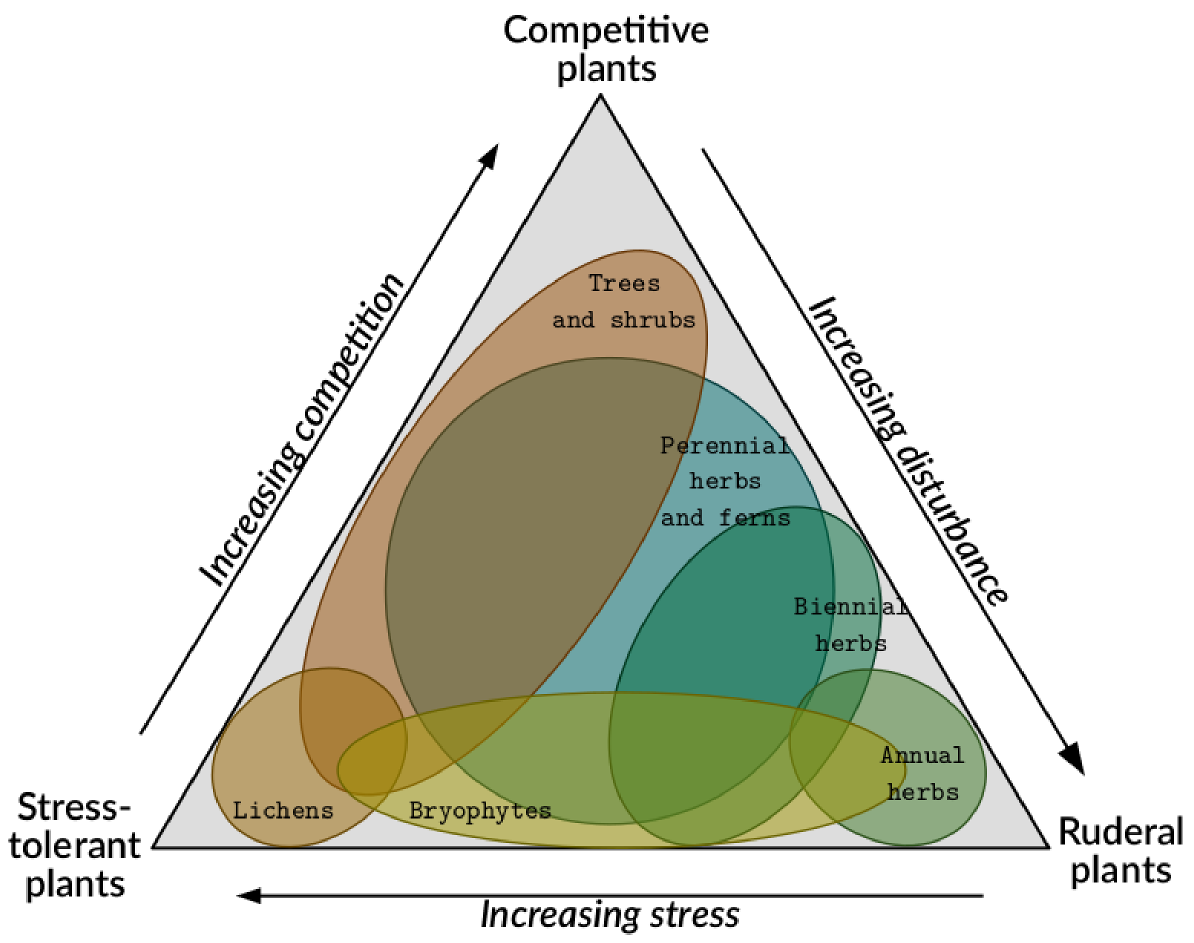


Figure 1: C-S-R triangle theory

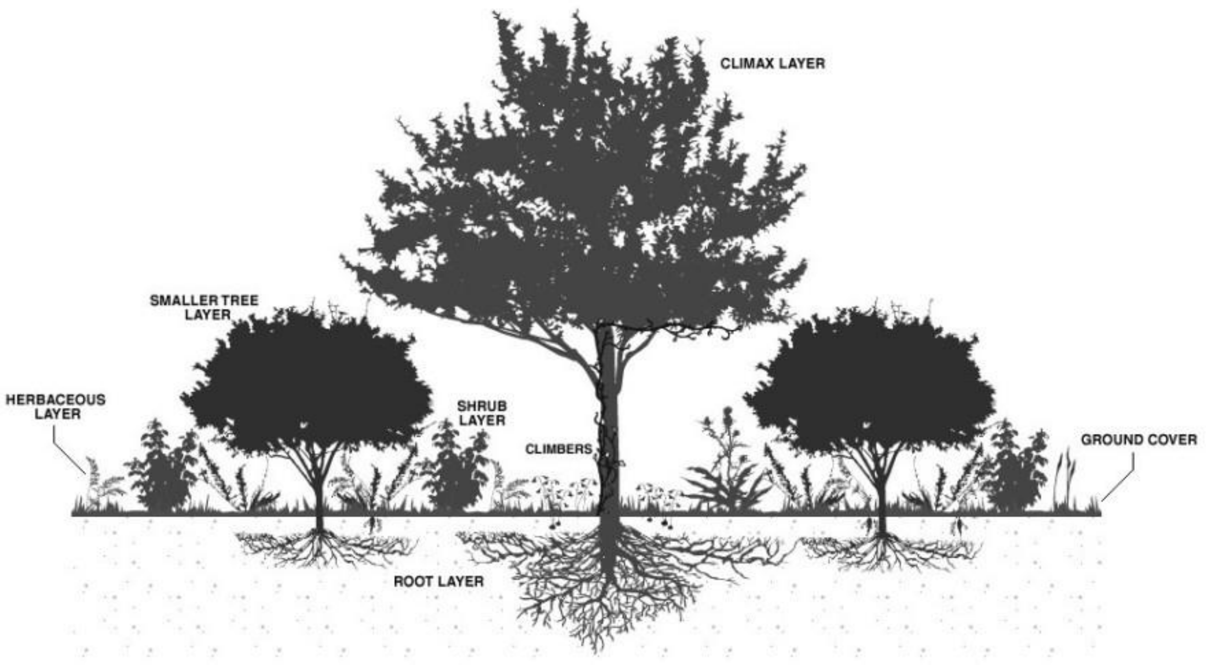
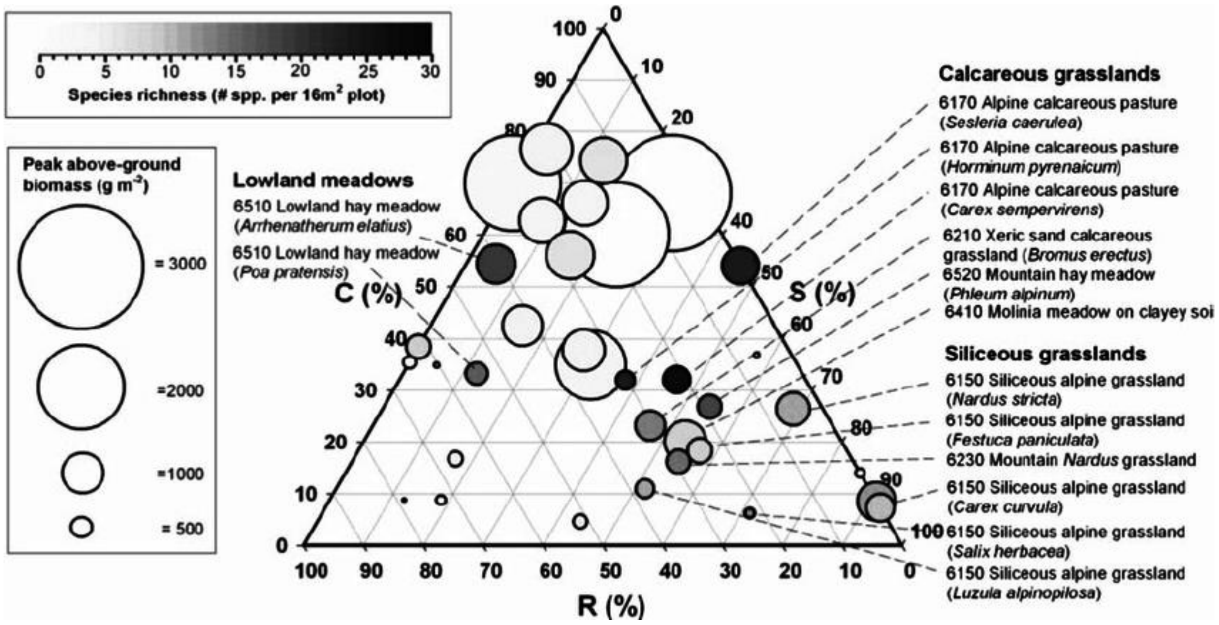


Figure 2: Vegetation layers

« ***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). 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).

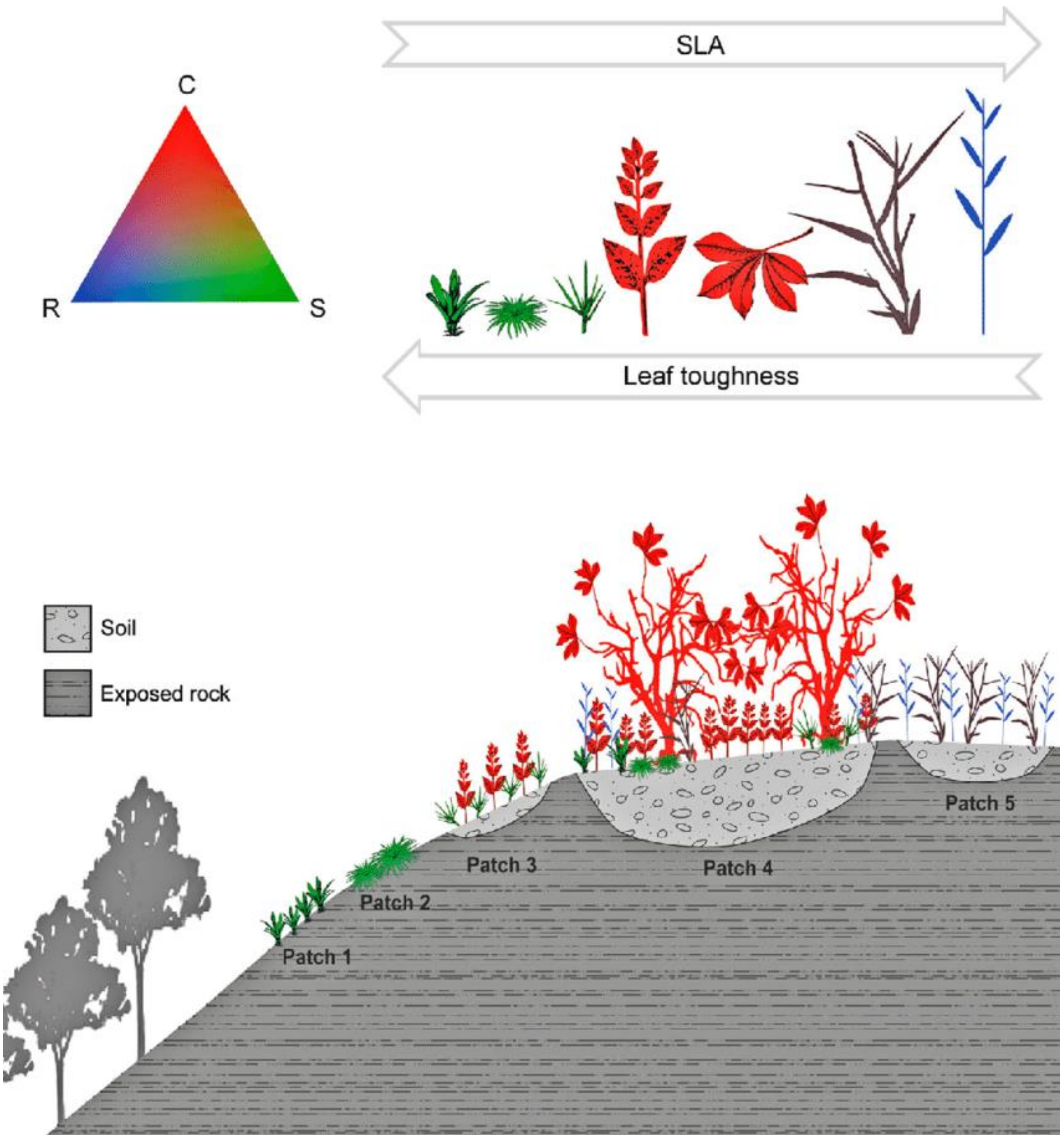


Figure 4: Extract from de Paula 2015, Fig4

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

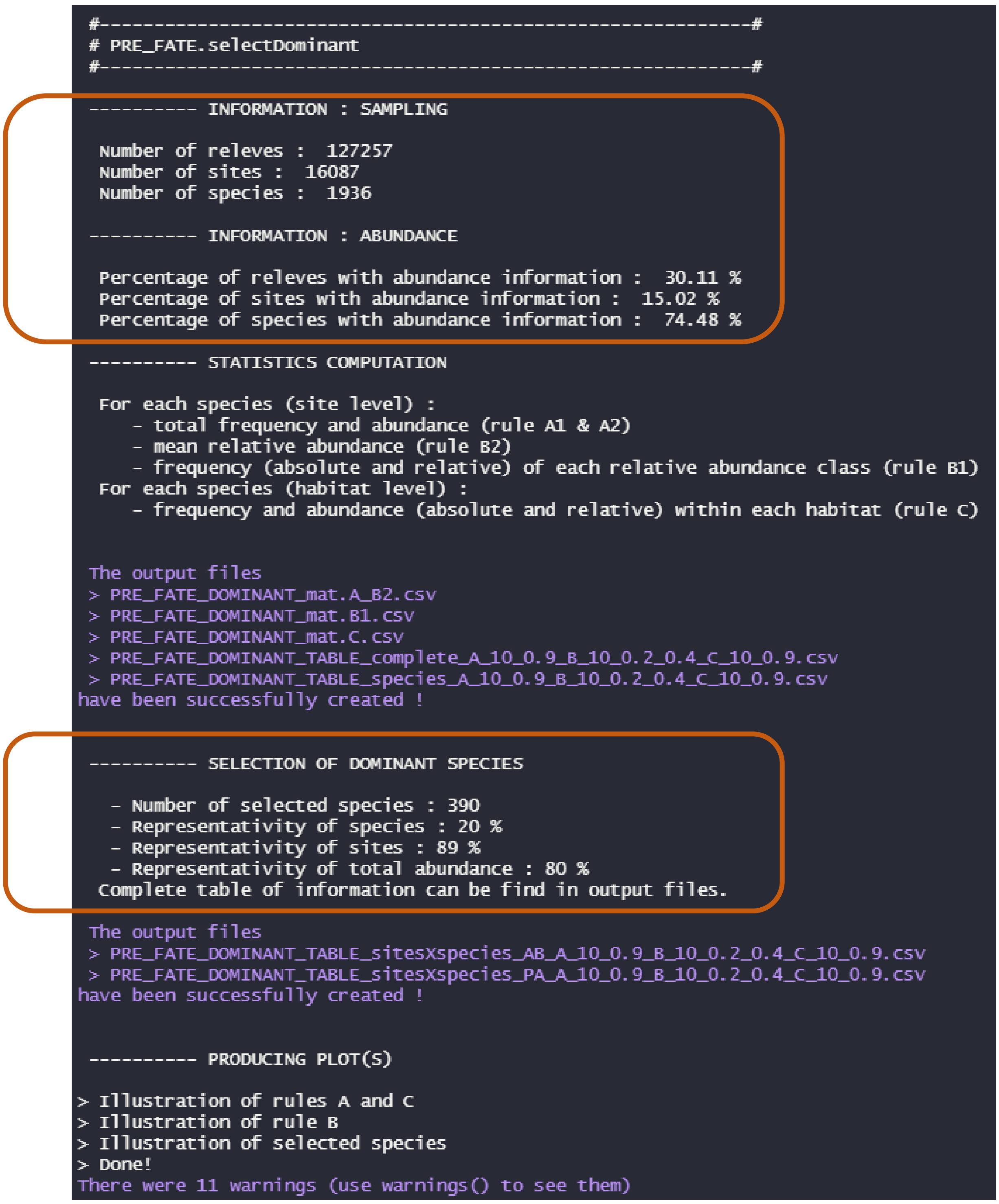
i. Selection of dominant species

- 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")

## 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)
```

XXX (interface example)

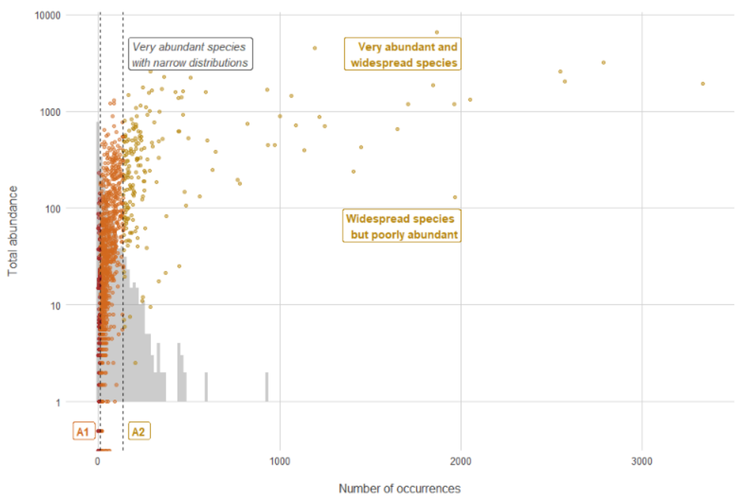


```
#---------------------------------------------------------------#
# PRE_FATE.selectDominant
#---------------------------------------------------------------#

 ---------- 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.

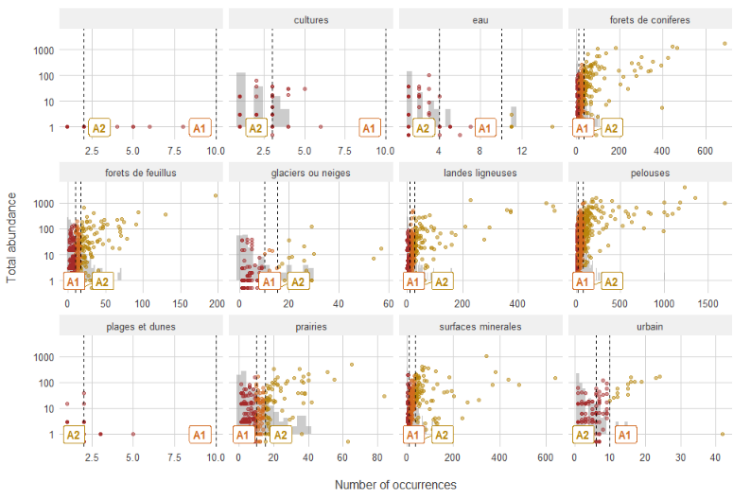
STEP 1 : Selection of dominant species - rule A

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



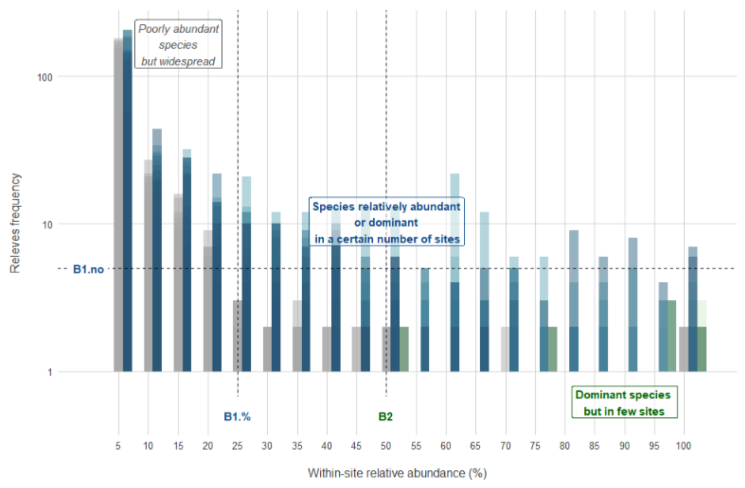
STEP 1 : Selection of dominant species - rule C

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



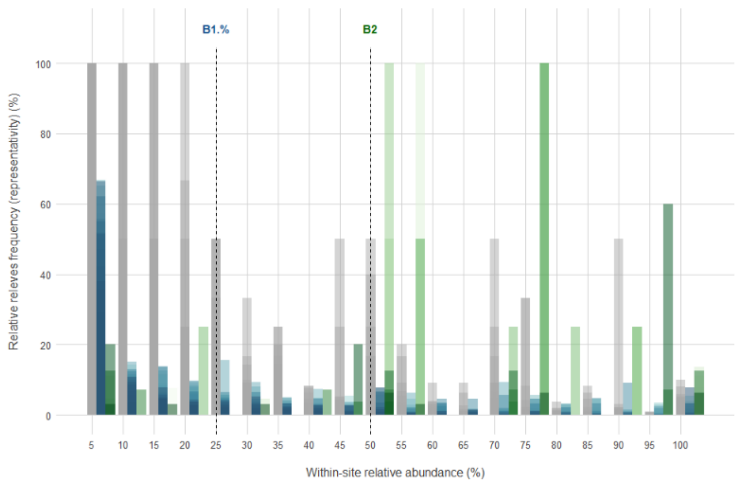
STEP 1 : Selection of dominant species - rule B

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%



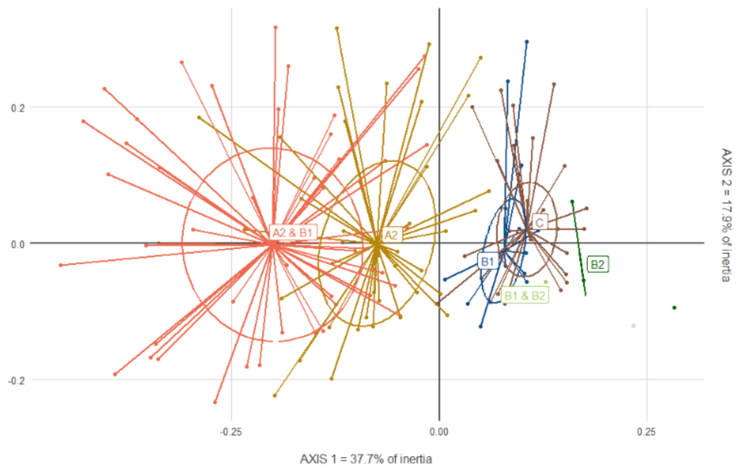
STEP 1 : Selection of dominant species - rule B

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%



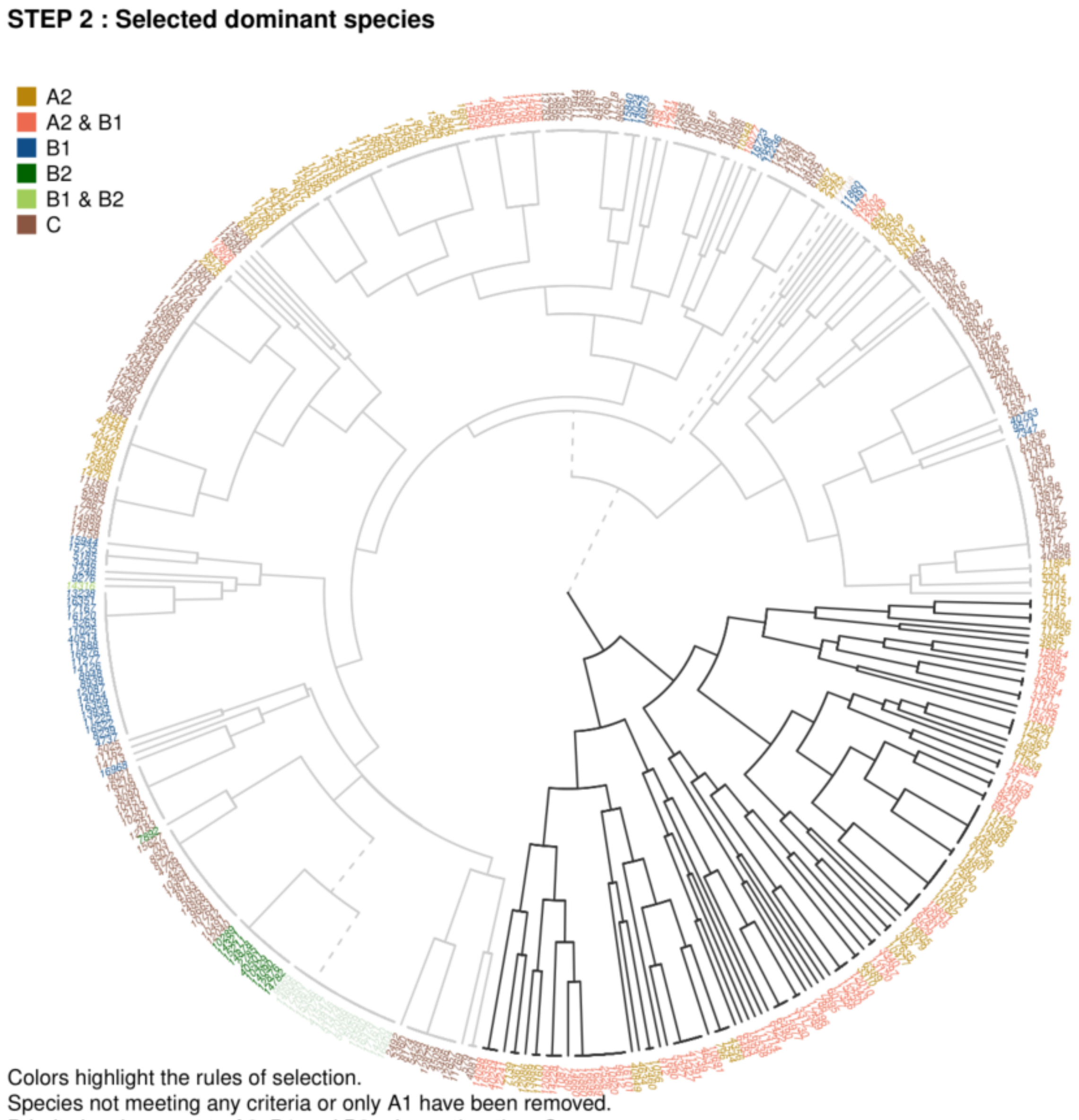
STEP 2 : Selected dominant species

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.



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.

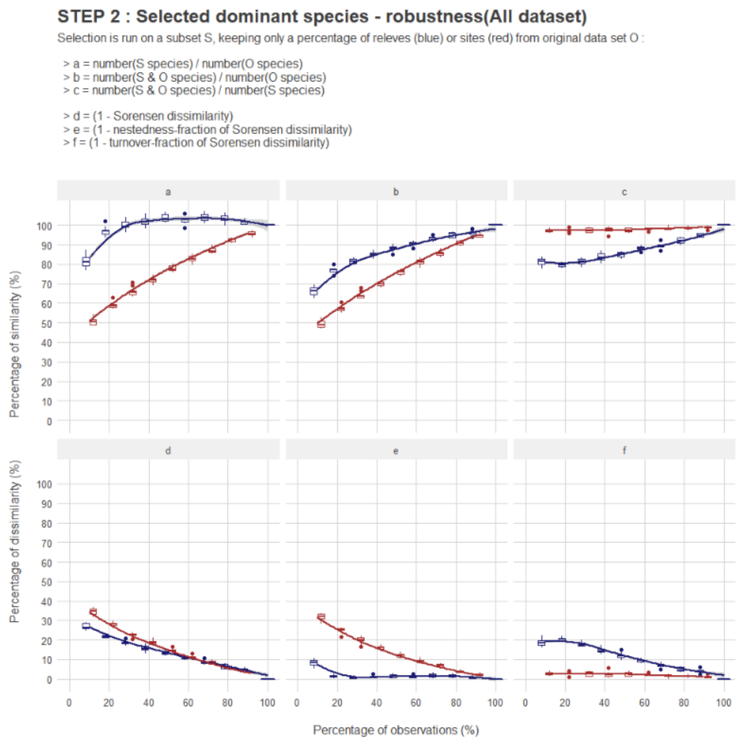
Representation into ordination space or phylogeny can help exploring by which criterion the species are selected, and maybe adjusting selection parameter values. *For example, if you decide to use selection criterion C, but no species are selected with it, you might want to change your parameter values. 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.*

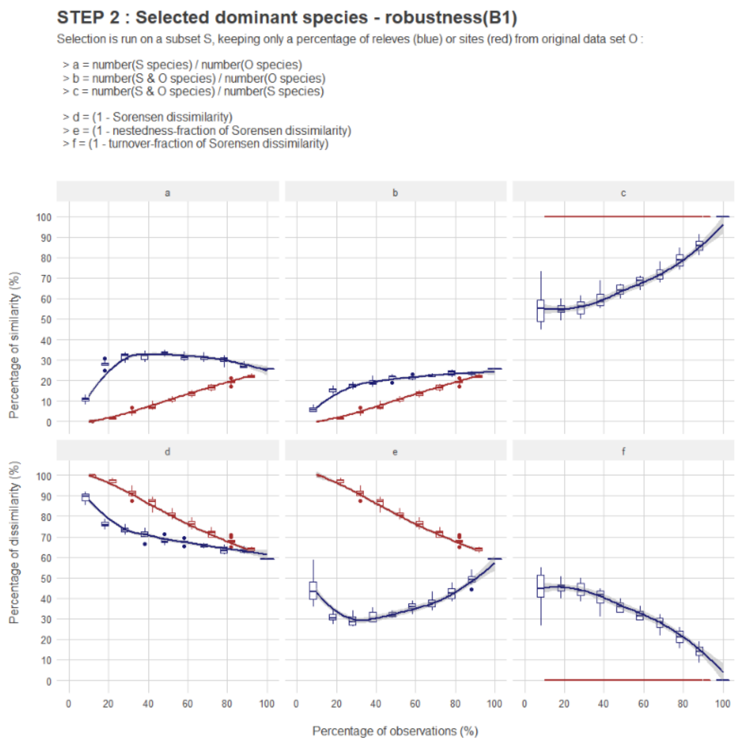
```
## No habitat, robustness (!quite long!) --------------------
tab.occ = tab[, c("sites", "species", "abund")]
sp.SELECT = PRE_FATE.selectDominant(mat.observations = tab.occ
                                    , opt.doRobustness = TRUE
                                    , opt.robustness_percent = 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.

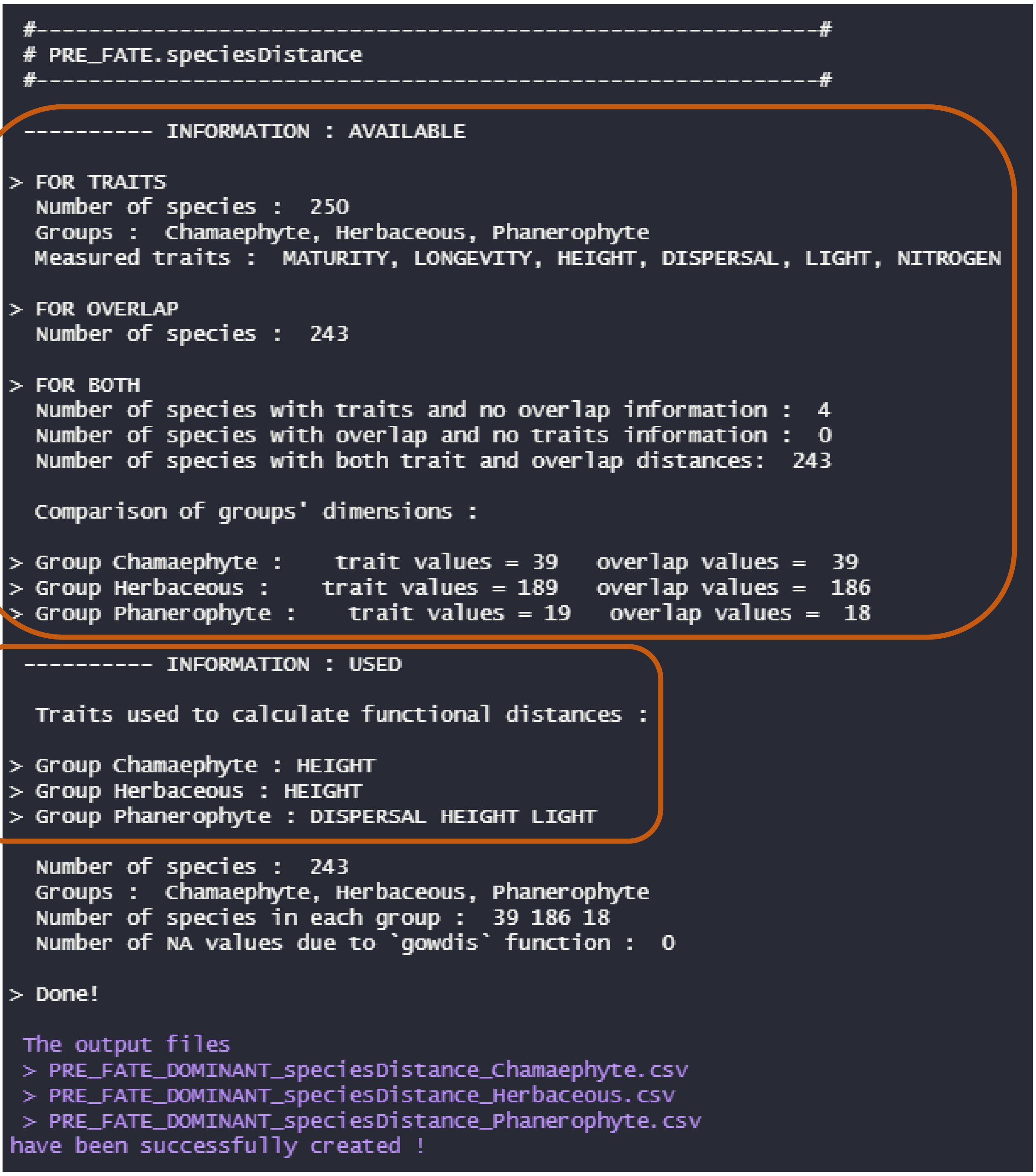
iii. Species pairwise distance

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")

## 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]])
```

XXX (interface example)

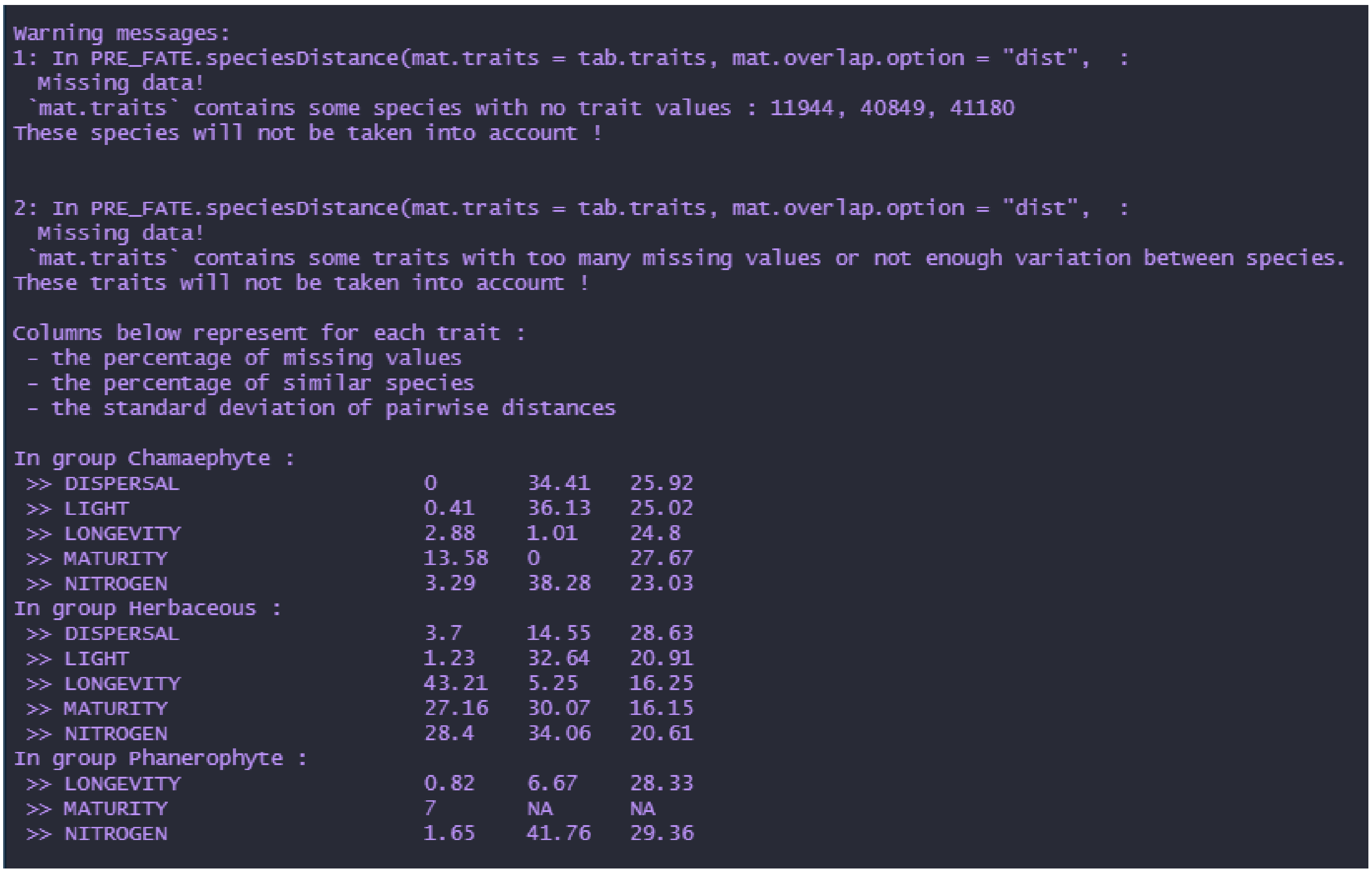


```
 #---------------------------------------------------------------#
 # PRE_FATE.speciesDistance
 #---------------------------------------------------------------#

 ---------- 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.*

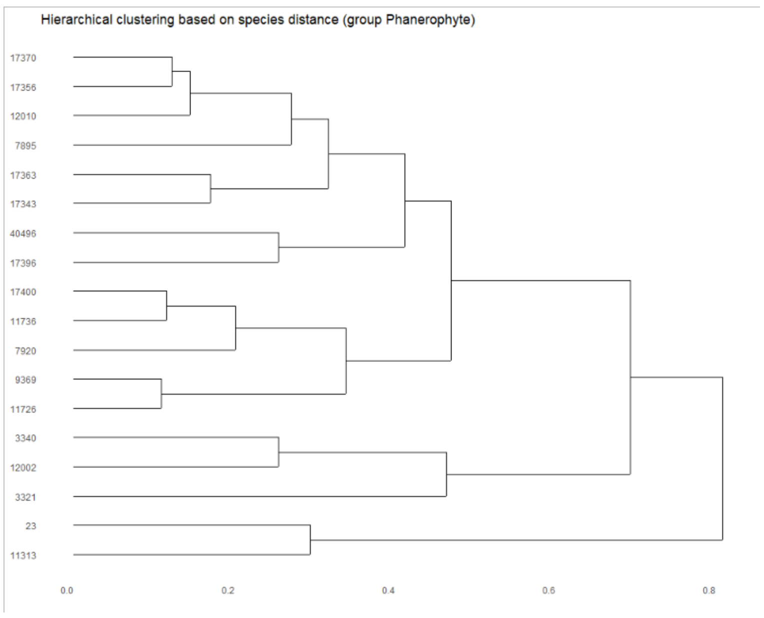

```
Warning messages:
1: In PRE_FATE.speciesDistance(mat.traits = tab.traits, mat.overlap.option = "dist",  :
  Missing data!
 `mat.traits` contains some species with no trait values : 11944, 40849, 41180
These species will not be taken into account !


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*).

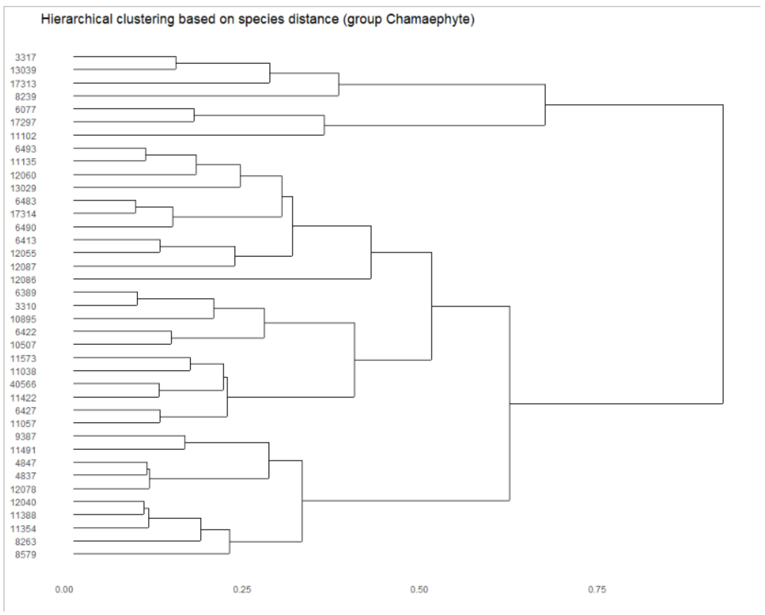
Once species pairwise distances calculated, they are in the form of species by species matrix, but can be easily represented through dendrograms : it gives an overview of what is going to be done in the following steps.

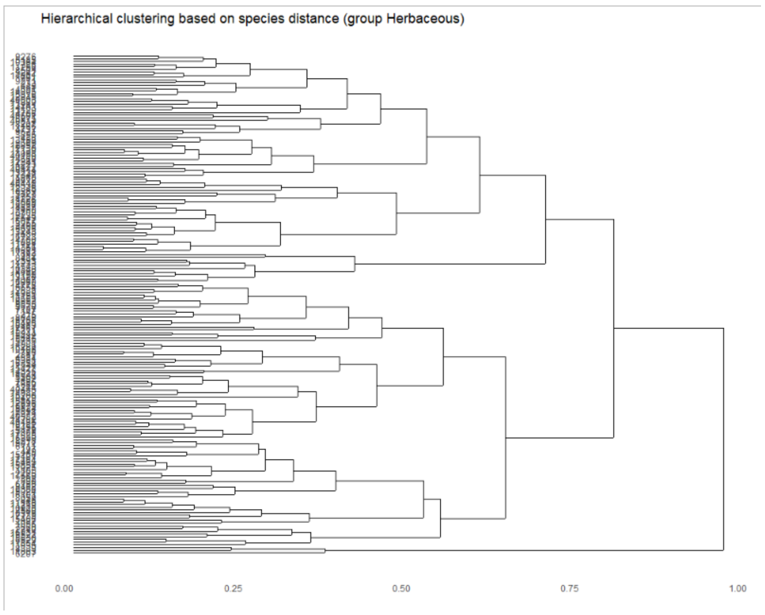
The distance matrix obtained (one for each given category, *here 3 : Phanerophyte, Chamaephyte and Herbaceous*) is classifying species according to the traits used, the overlap distances, and the weights given to these two sources of information.



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 :

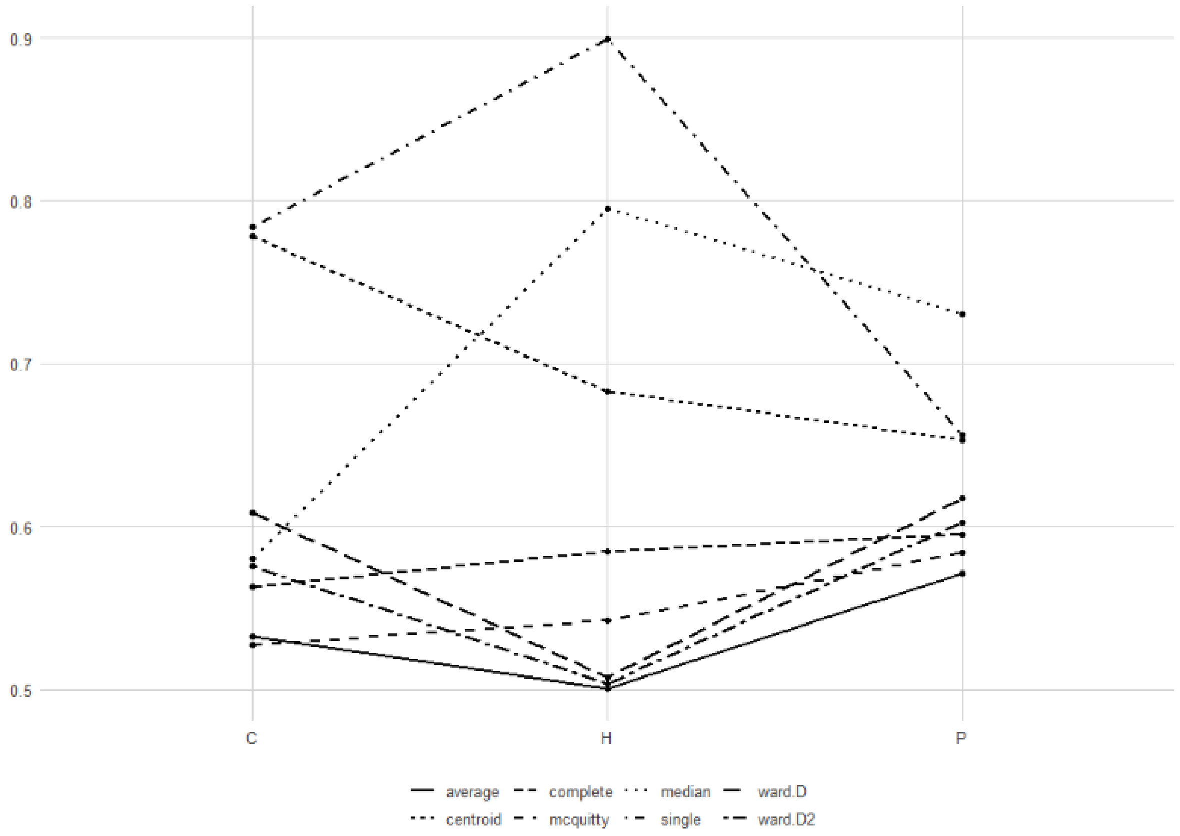
- calculate all possible **clusters**, and the corresponding evaluation metrics with the function PRE_FATE.speciesClustering_step1
- choose the best number of clusters from the previous step and find **determinant species** with the function PRE_FATE.speciesClustering_step2
- combine traits data and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

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

## 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)
```

XXX

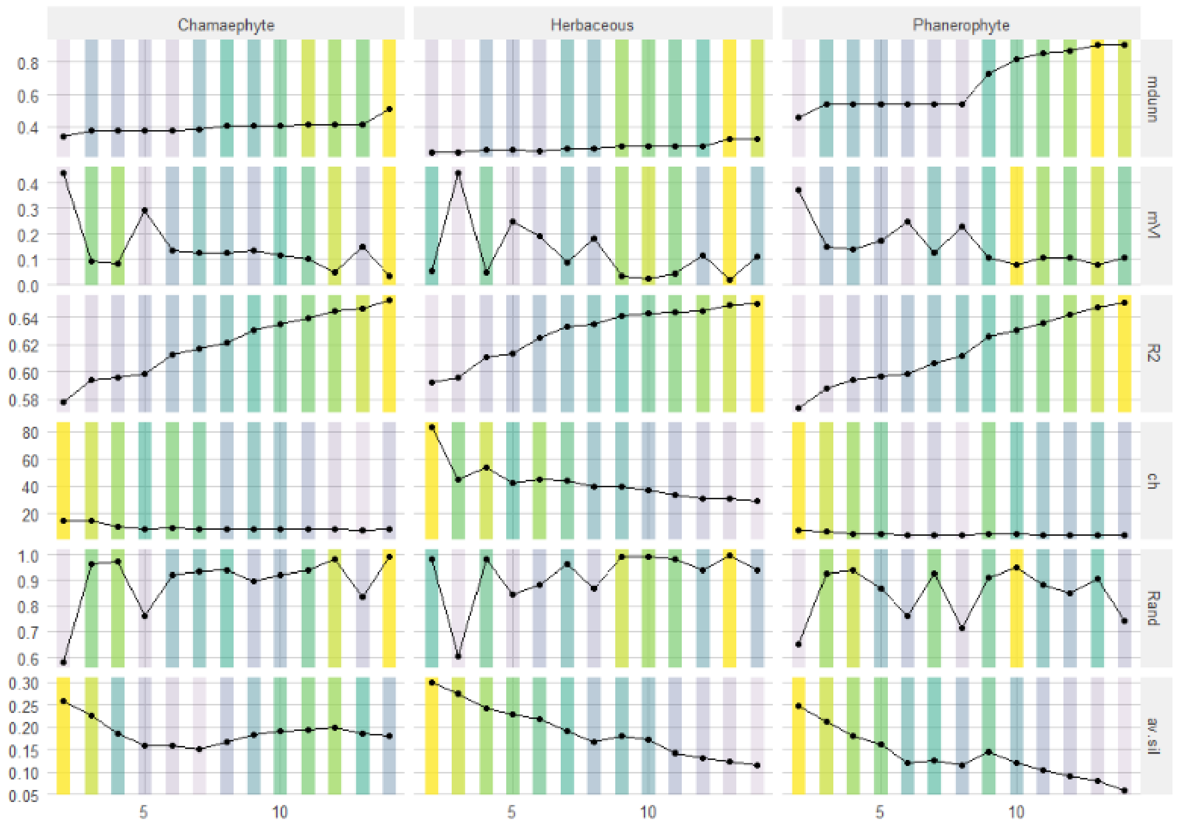
STEP A : Choice of clustering method

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.

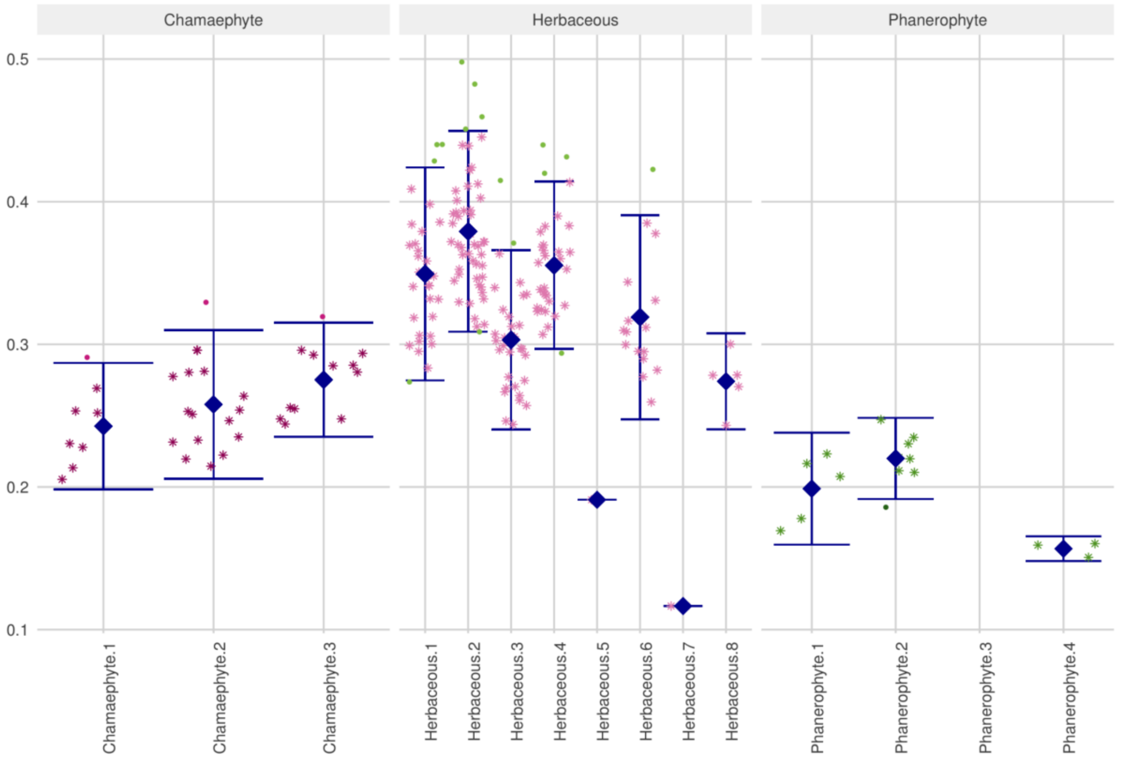
```
## Number of clusters per group
plot(sp.CLUST$plot.clustNo)
no.clusters = c(4, 3, 8) ## Phanerophyte, Chamaephyte, Herbaceous

## 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.

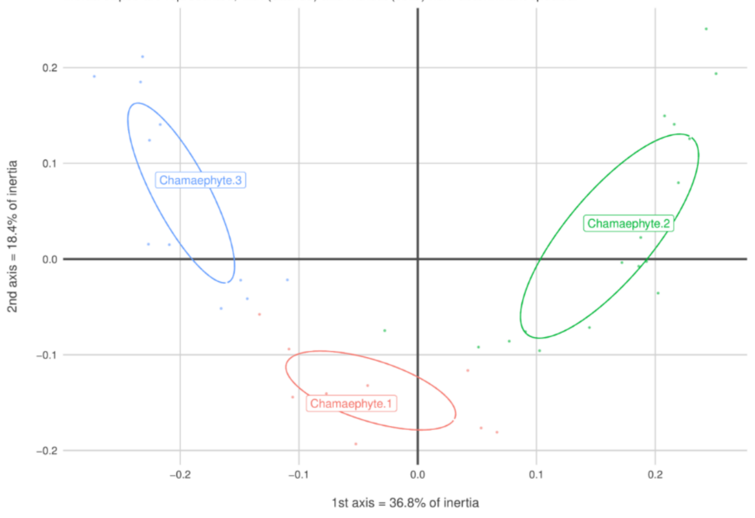
STEP C : Removal of distant species

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.



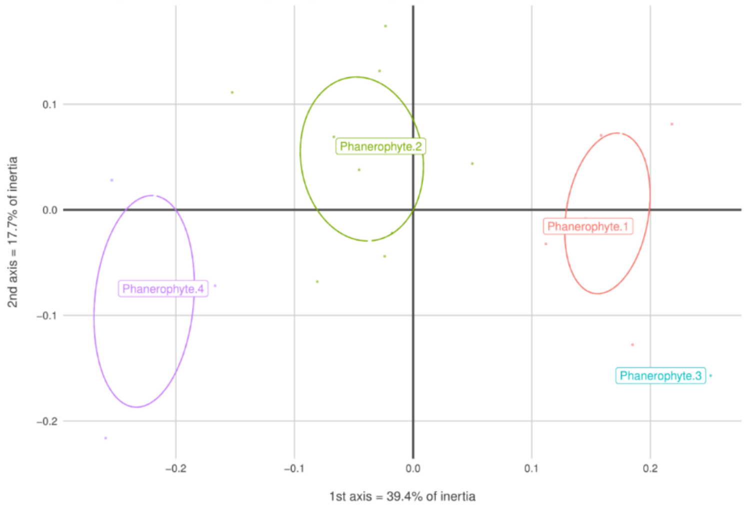
STEP C : Removal of distant species : group Chamaephyte

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.



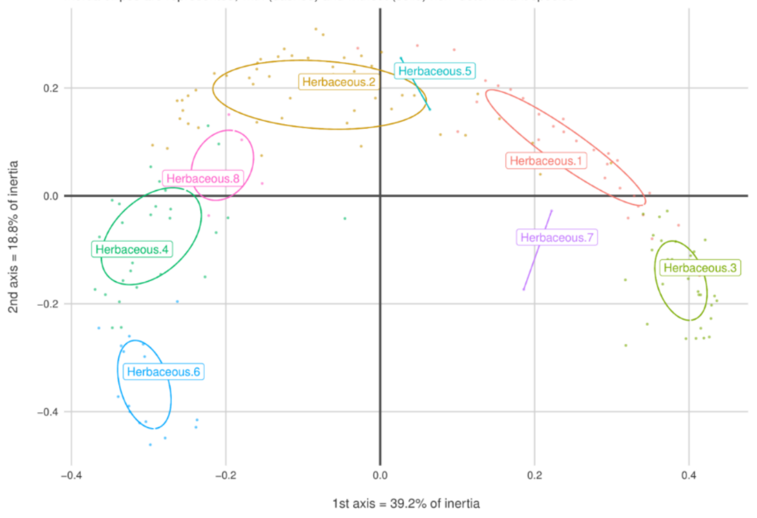
STEP C : Removal of distant species : group Phanerophyte

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.



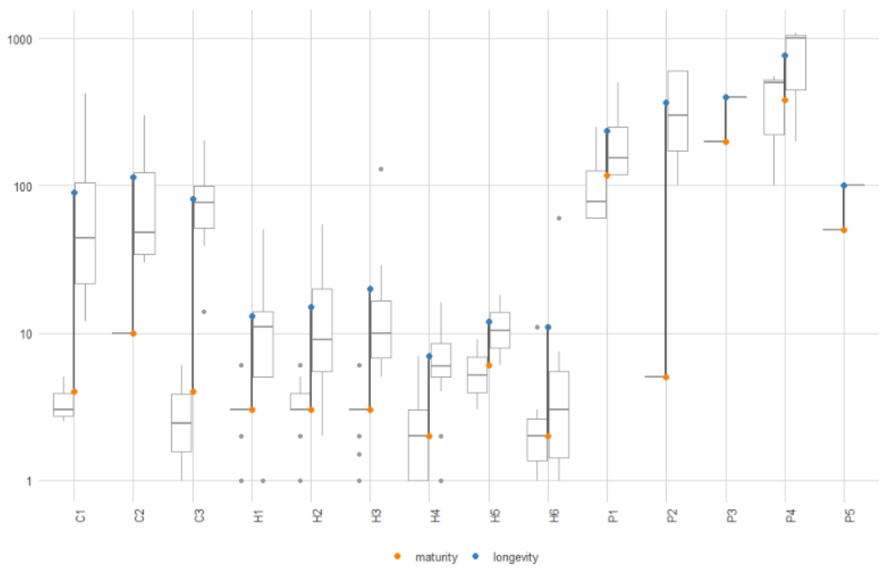
```
## Load example data
Champsaur_PFG = .loadData("Champsaur_PFG")

## 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.*

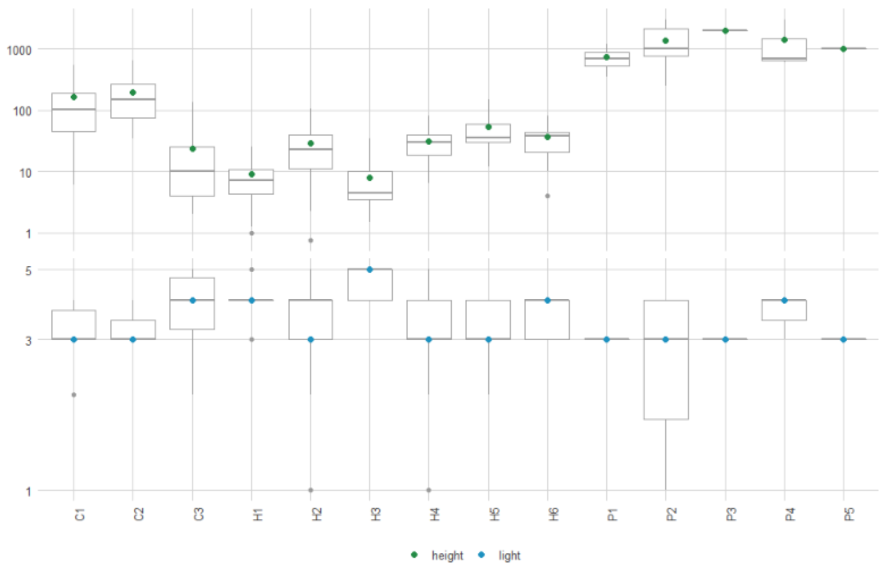
STEP D : Computation of PFG traits values : longevity & maturity

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.



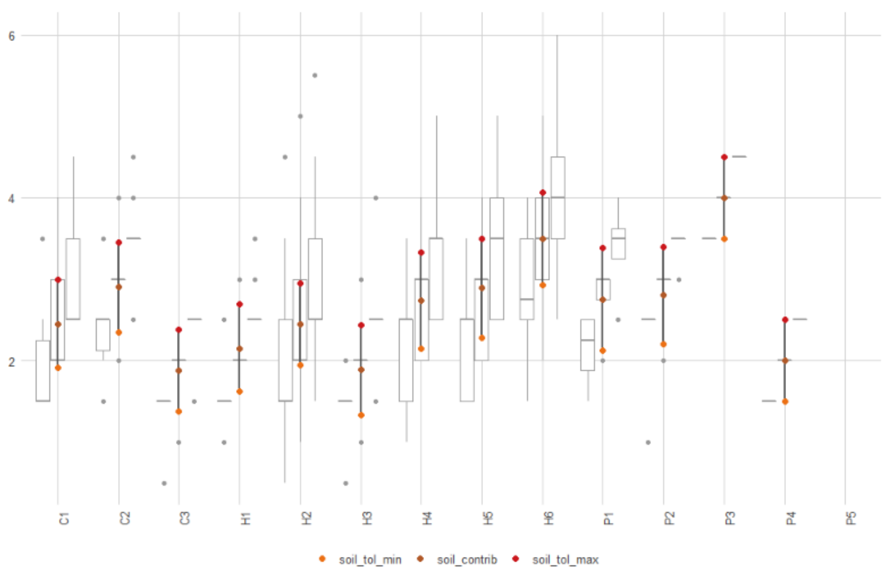
STEP D : Computation of PFG traits values : height & light

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.

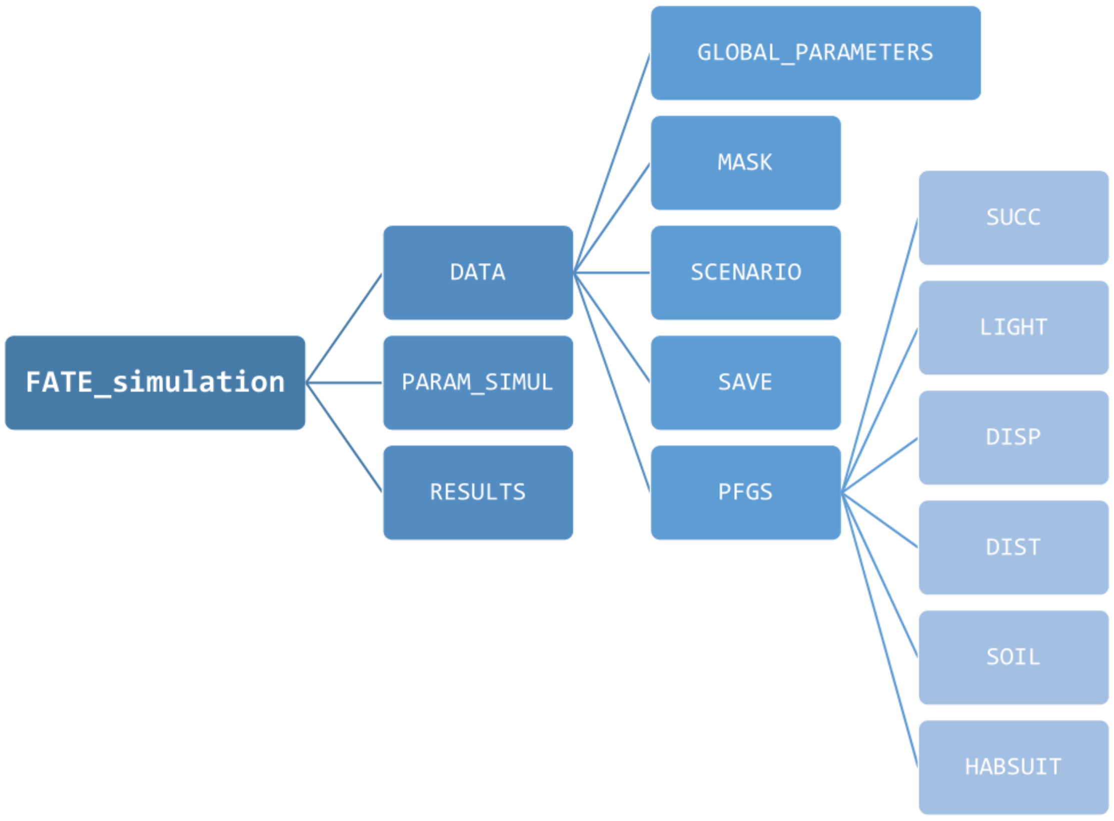


Figure 5: a FATE simulation directory

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

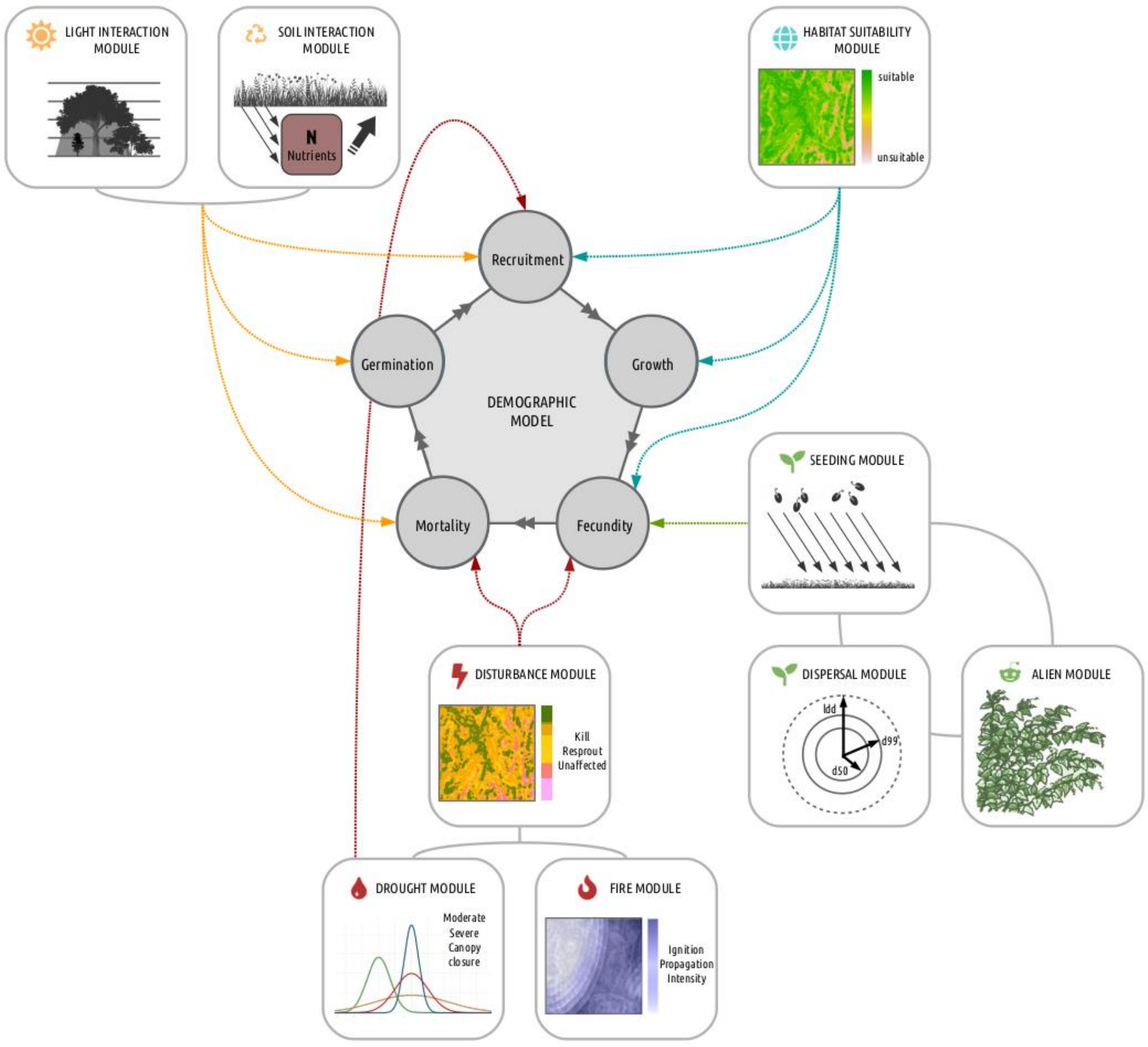


Figure 6: FATE conceptual diagram

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 (Fig5) :

- ⌘ **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 (Fig6) :

- ⅄ 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 (Fig7) :

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)**.

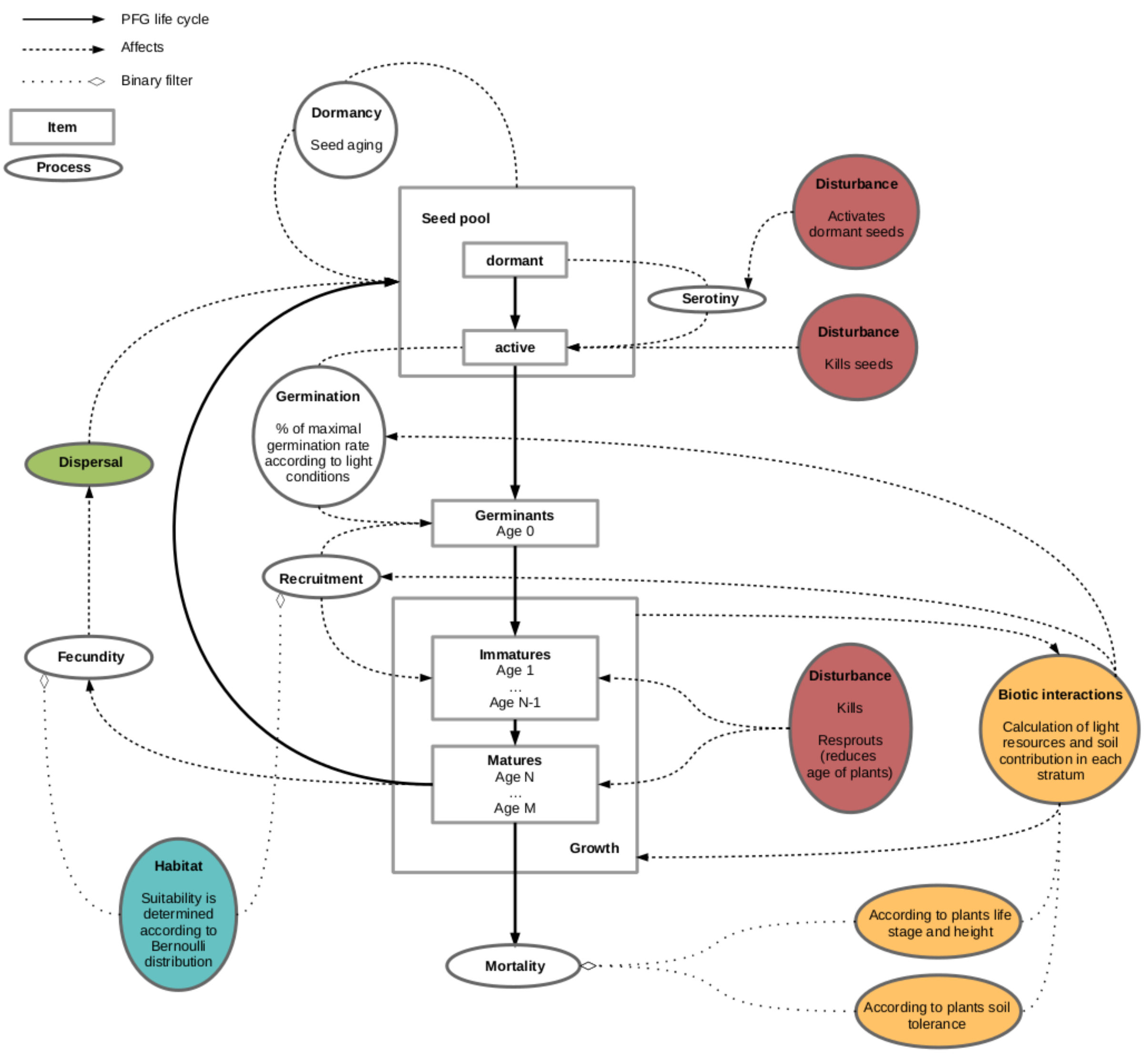


Figure 7: FATE life cycle

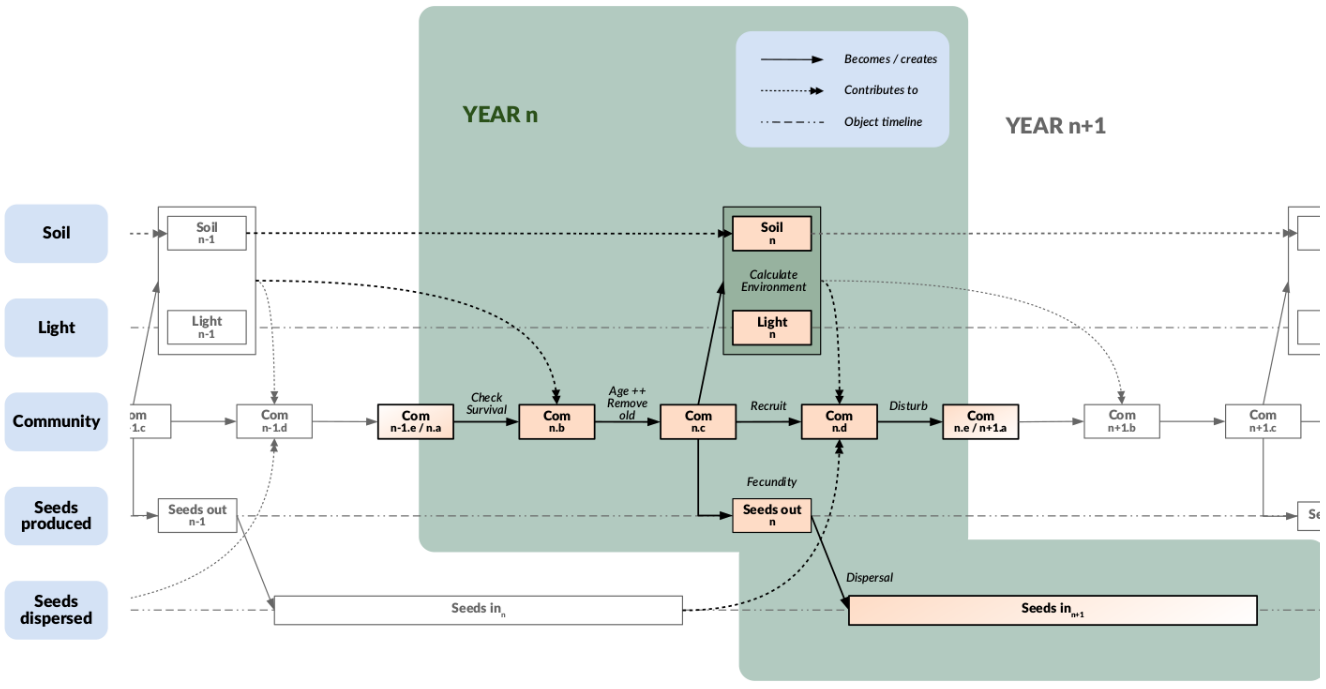


Figure 8: FATE timeline diagram

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 (Fig8), and in a more quantitative way for soil (Fig9).

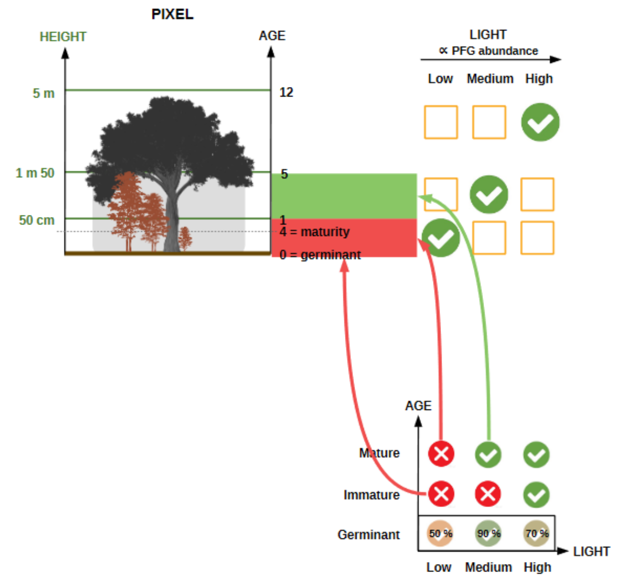


Figure 8: Calculation of FATE resources (light)

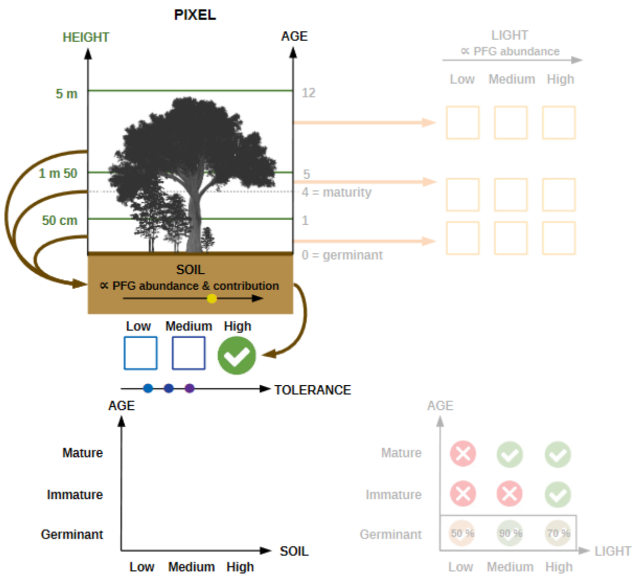


Figure 9: Calculation of FATE resources (soil)

iv. Structural equations

Internal processes within FATE are presented on Fig10. 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.

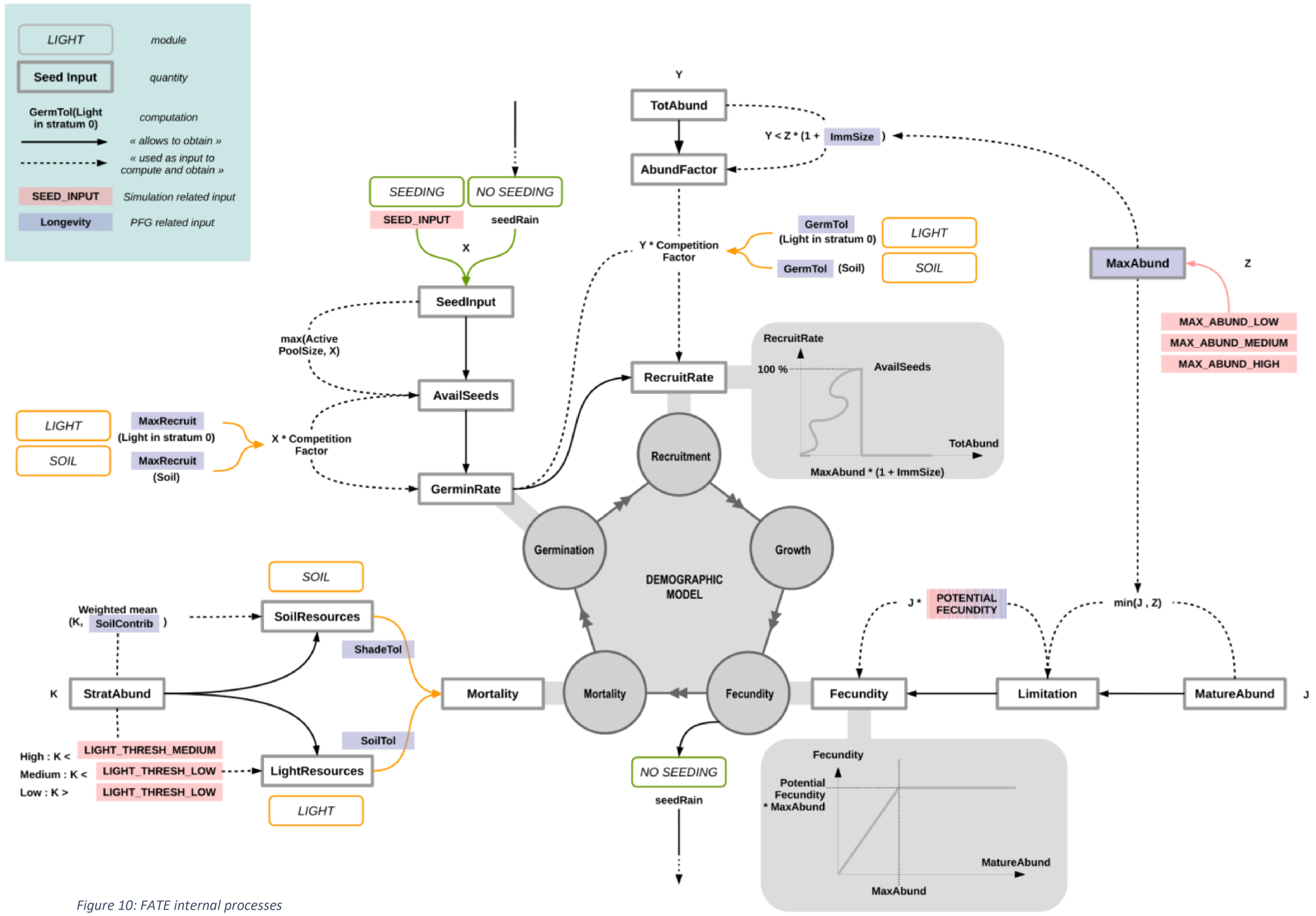


Figure 10: FATE internal processes

Influence of environment (habitat suitability)

- **$getEnv$ … () functions :**
  All the $getEnv\ldots()$ 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 $getEnv\ldots()$ functions will return 1)
  - otherwise, the environment is considered unsuitable for the PFG (hence $getEnv\ldots()$ 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.

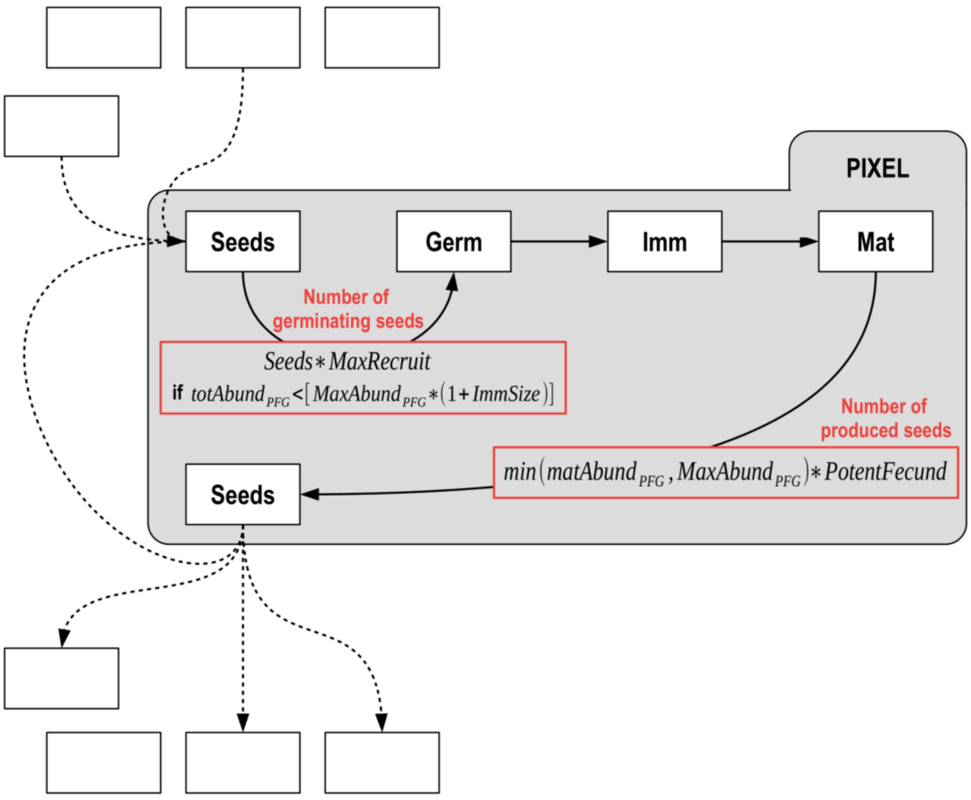

PIXEL
Seeds
Germ
Imm
Mat
Number of germinating seeds
$Seeds * MaxRecruit$
if $totAbund_{PFG} < [MaxAbund_{PFG} * (1 + ImmSize)]$
Number of produced seeds
$min(matAbund_{PFG}, MaxAbund_{PFG}) * PotentFecund$
Seeds

Figure 11: 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)
```

- 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)
```

➢ 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")

## 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)
```

XXX

XXX

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
```

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

## 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)
```

XXX

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)

<u>Required parameters :</u>

- 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
```

XXX

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")

## 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")])
```

XXX

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")

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


## PFG traits for light
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)
```

XXX

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_scenario1/DIST_changingmask_years.txt
  --DIST_CHANGEMASK_FILES--
  FATE_simulation/DATA/SCENARIO/DIST_scenario1/DIST_changingmask_files_t50.
txt
  FATE_simulation/DATA/SCENARIO/DIST_scenario1/DIST_changingmask_files_t100
.txt
  FATE_simulation/DATA/SCENARIO/DIST_scenario1/DIST_changingmask_files_t150
.txt
```

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

## 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 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")

## 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)
```

XXX

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
```

XXX

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
```

XXX

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
```

XXX

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

c. From abundance to presence/absence

Publications

Supplementary material