GCBM Chile Data Preprocessing

Release v0.9.1

moja global docs team

Sep 28, 2021

CONTENTS:

Moja global's GCBM Chile data pre-processing project aims to host the data pre-processing algorithms used by Chile to pre-process datasets to be used by the Generic Carbon Budget Model (GCBM). The dataset has been sourced from Forest Cadastre maps, and the National Forest Inventory, with the Los Rios Region, in southern Chile, utilized as a proof of concept for demonstrating the implementation of the model.

GCBM has been utilized as an open-source modelling framework that implements the IPCC gain-loss method to estimate stocks and stock changes. With the help of GCBM, the overall ecosystem carbon balance from all sources and sinks at each annual time step is modelled. The model then tracks carbon mass transfers in and out of the forest ecosystem to ensure the carbon mass balance.

The preprocessing algorithms are designed to replicate the data preparation conducted by Chile in the elaboration of its Forest Reference Emissions Level / Forest Reference Level, FREL/FRL, submitted on August 31st, 2016. The results derived from the use of these algorithms do not reflect the positions of the Government of Chile for REDD+ accounting or any other purpose.

The methods and results of this work were compiled into the technical document **"Modelling forest carbon dynamics for REDD+ using the Generic Carbon Budget Model (GCBM)"**, where more details can be found.

CHAPTER

ONE

MOJA GLOBAL CHILE DATA PRE-PROCESSING PROJECT SETUP

The following section describes how to set up the project on your local machine. The following are minimum requirements:

1. R programming language
2. R Studio
3. Git

1.1 Setting up the project

The following instructions describe how to install all the required tools to do live data pre-processing on a Windows 10 system.

After installing, R Studio and R programming language, start the R Studio IDE and follow these steps:

1. Click on `File` and select `New Project`.
2. Click on `Version Control` to checkout a project from a version control repository.
3. Click on `Git` to clone the project from our GitHub repository.
4. Add the repository URL: `https://github.com/moja-global/GCBM.Chile.Data_Preprocessing`. Select the subdirectory as per your needs and click on `Create Project`.
5. After the repository is cloned and a workspace is initialized, download the Input data required for the data pre-processing. You can download it from the GitHub releases:
    1. Visit the GitHub releases on the GCBM Chile Data Preprocessing repository and check out the latest release.
    2. Scroll down to find the `Input_Files.rar` which contains the dataset used by the pre-processing algorithm.
    3. Click on `Input_Files.rar` and download it on your local machine. Extract the `Input_Files` directory on the root of the cloned GCBM Chile Data Preprocessing repository.
    4. **Optional**: You can download the `Output_Files.rar` as well for reproducibility and checking the results of the pre-processing algorithm.
6. Ensure the following directory structure for the project:

```
├── Input_Files
│   ├── Growth                # Excel spreadsheet with growth data
│   ├── LUC                   # Trazabilidad (Land use) data
│   ├── SOC                   # Soil Organic carbon data
|   └── Temperature           # Temperature raw data (NetCDF)
├── Output_Files
```

(continues on next page)

(continued from previous page)

```
│   ├── input_database
│   └── layers
│       └── raw
│           ├── disturbances
│           ├── environment
│           └── inventory
├── Processing_codes
├── README.md
├── docs/
├── run_all.R
└── ...
```

1.2 Running the project

With the initial setup complete, you are now ready to run the pre-processing algorithm code.

1. From the R Studio IDE, select the `run_all.R` file and click on `Run` to run all the pre-processing code.
2. Optionally, choose the individual files in the `Preprocessing_Codes` to run them singularly. Make sure to install the R packages as suggested by the R Studio.

With the repository and tools set up on your workstation, you can now either edit existing code or prepare local datasets for a GCBM run using R.

CHAPTER

TWO

UNDERSTANDING PREPROCESSING CODES

The Preprocessing codes host all the preprocessing algorithms utilized by the GCBM Chile Data Pre-Processing project. The code is hosted on the Preprocessing_Codes sub-directory of the repository and allows developers to concentrate on running individual simulations on the input dataset and retrieve the necessary output. In this section, we would delve deep into the individual code file and understand them.

2.1 Listing all Preprocessing Codes

All the individual code files should be run using R Studio which abstracts away complex configurations and package management. You can primarily run `run_all.R` present on the repository root, to run all the processing codes.

Preprocessing Code Name	Description	Preprocessing Code Source
Install packages	Install the necessary packages using the checkpoint package.	00-Install_packages.R
Temperature layer	Process the CR2 temperature product to obtain the mean temperature layer (1997-2016).	01-Temperature_layer.R
Growth Curves	Build the Growth curves using the Annual Growth values for each forest type.	02-Growth_curves.R
Inventory layer	Create the inventory layer (initial layer).	03-Inventory_layer.R
Afforestation layer	Create the afforestation layer: Land use changes from non-forest to forest.	04-Afforestation_layer.R
Forest loss layer	Create the deforestation and substitution layers.	05-Forest_loss_layer.R
Forest degradation layer	Create the degradation layers and disturbance matrix values.	06-Forest_degradation_layer.R
Forest enhancement layer	Create the forest enhancement layers and add the new growth curves.	07-Forest_enhancement_layer.R

2.2 Official Preprocessing Codes

The Preprocessing Code files describe the packages needed, folder paths, parameters, layers and the core algorithm organized in a separate single unit. You can run each of the Preprocessing codes individually, provided the folder paths are configured correctly as described in the installation.

2.2.1 Install Packages

The Install Packages code installs all the necessary packages using the `checkpoint` package. The `checkpoint` package is used to secure reproducibility and downloads the package version of that specific date.

```
if (!require(checkpoint)) install.packages("checkpoint")
library("checkpoint")
checkpoint("2019-10-01")
```

Here `2019-10-01` is the date of the compatibility packages.

2.2.2 Temperature Layer

The Temperature Layer code processes the CR2 temperature product to obtain the mean temperature layer between 1997 to 2016. The `initial_year` and `final_year` defines the initial and last year of execution respectively. The `initial_year_temp` and the `final_year_temp` as the initial and last year of execution of the CR2 product respectively.

The `layer_temp` is defined as the product of the average mean temperature defined by the CR2 climate research centre. After the definition of the Temperature raster folder for the `input_temp`, we read the first layer of the netCDF file. Each netCDF band is a month and the band corresponding to January of the first year of the simulation is calculated.

```
temp <- raster(paste0(input_temp, "/", layer_temp), band = 1)
dif_year <- initial_year - initial_year_temp
initial_band <- (dif_year * 12) + 1
final_band <- ((final_year_temp - initial_year_temp) * 12) + 12
```

Later we calculate the mean temperature by adding all the bands and dividing them by the number of bands.

```
for (i in initial_band:final_band) {
  temp <- raster(paste0(input_temp, "/", layer_temp), band = i)
  if (i == initial_band) {

    sum_temp <- temp
    count <- 0
  } else {
    sum_temp <- sum_temp + temp
    count <- count + 1
  }
  setTxtProgressBar(pb, i)
}

mean_temp <- sum_temp / count
```

The maximum and minimum temperature is calculated and the output is written to a raster file.

2.2.3 Growth Curves

The Growth Curves Code file builds the Growth curves using the Annual Growth values for each forest type. The `age_stop_growth` and `age_max_curves` parameters define the age at which the growth of the forest stops and the maximum age reflected in the growth curves respectively. The `growth_table` defines the annual periodic increment tables which increment in commercial wood. The `regional_vol` and `regional_vol_ma` defines the regional average volume and the average volume for the Matorral Arborescente forest type respectively.

The growth curves are constructed with the help of forest types, forest structures defined as a list in the R programming language. A growth curve is constructed for each combination of forest type and forest structure. Taking an example here where the forest type is Tipofor and the structure is Estruc:

```
for (Tipofor in forest_types){
  for (Estruc in structures){
    if (Estruc %in% c("Renoval","Adulto Renoval")){
      growth_year <- dplyr::select(filter(growth_table,TipoFor_classifier == Tipofor),
↪Annual_Growth_Renoval)
    }
    if (Estruc %in% c("Adulto","Achaparrado")){
      growth_year <- dplyr::select(filter(growth_table,TipoFor_classifier == Tipofor),
↪Annual_Growth_AdultoRenoval)
    }
    seq_growth_first <- seq(from= 0, by=as.numeric(unname(growth_year)),length.out = age_
↪stop_growth+1)
    seq_growth_last <- rep(seq_growth_first[length(seq_growth_first)],age_max_curves -␣
↪age_stop_growth)
    seq_growth <- c(seq_growth_first,seq_growth_last)
    growth_curve <- c(Tipofor,Estruc,Origen,AIDBSPP,seq_growth)
    growth_curve
    if (counter == 1){
      growth <- rbind(growth_curve)
    } else {
      growth <- rbind(growth,growth_curve)
    }
    counter = counter + 1
  }
}
```

Furthermore, extra classifiers are added. The Bosque mixto is a mix of several forest types and has its own classifier:

```
growth_year <- dplyr::select(filter(growth_table,TipoFor_classifier == "Bosque Mixto"),
↪Annual_Growth_Renoval)

seq_growth_first <- seq(from= 0, by=as.numeric(unname(growth_year)),length.out = age_
↪stop_growth+1)
seq_growth_last <- rep(seq_growth_first[length(seq_growth_first)],age_max_curves - age_
↪stop_growth)
seq_growth <- c(seq_growth_first,seq_growth_last)

growth_curve <- c("Bosque Mixto","Bosque Mixto",Origen,AIDBSPP,seq_growth)
growth <- rbind(growth,growth_curve)
```

The Matorral Arborescente is not properly a forest type but due to a change in the forest definition it is included in the FREL:

```
growth_year <- dplyr::select(filter(growth_table,TipoFor_classifier == "Matorral␣
↪Arborescente"),Annual_Growth_Renoval)

seq_growth_first <- seq(from= 0, by=as.numeric(unname(growth_year)),length.out = age_
↪stop_growth+1)
seq_growth_last <- rep(seq_growth_first[length(seq_growth_first)],age_max_curves - age_
↪stop_growth)
seq_growth <- c(seq_growth_first,seq_growth_last)

growth_curve <- c("Matorral Arborescente","Matorral Arborescente",Origen,"Western larch",
↪seq_growth)
growth <- rbind(growth,growth_curve)
```

The Promedio Regional is an "artificial" forest type created when the forest type is not possible to determine (included in the REDD+ Annex):

```
growth_year <- dplyr::select(filter(growth_table,TipoFor_classifier == "Promedio Regional
↪"),Annual_Growth_Renoval)

seq_growth_first <- seq(from= 0, by=as.numeric(unname(growth_year)),length.out = age_
↪stop_growth+1)
seq_growth_last <- rep(seq_growth_first[length(seq_growth_first)],age_max_curves - age_
↪stop_growth)
seq_growth <- c(seq_growth_first,seq_growth_last)

growth_curve <- c("Promedio Regional","Promedio Regional",Origen,AIDBSPP,seq_growth)
growth <- rbind(growth,growth_curve)
```

We can then create the growth curves for the Bosque Inicial (Initial Forest). Right now we are using assign the Red Alder species, which will be replaced afterwards with Chilean generic species parameters. We go for all combinations of forest type and structure and the growth curve will give a fixed value of volume for all the forests in the area:

```
Origen <- "Bosque Inicial"
AIDBSPP <- "Red alder"

for (Tipofor in forest_types){
  for (Estruc in structures){
    growth_year <- regional_vol / age_stop_growth
    seq_growth_first <- seq(from= 0, by=as.numeric(unname(growth_year)),length.out = age_
↪stop_growth+1)
    seq_growth_last <- rep(seq_growth_first[length(seq_growth_first)],age_max_curves -␣
↪age_stop_growth)
    seq_growth <- c(seq_growth_first,seq_growth_last)
    growth_curve <- c(Tipofor,Estruc,Origen,AIDBSPP,seq_growth)
    growth <- rbind(growth,growth_curve)
  }
}
```

In a similar manner, we can add extra classifiers curves for the initial forest. For non-stocked (non-forest) pixels, we can use the "non stocked" species so the GCBM will ignore the growth:

```
seq_growth <- rep(0,age_max_curves + 1 )
growth_curve <- c("No forestal","No forestal","No forestal","Not stocked",seq_growth)
growth <- rbind(growth,growth_curve)
```

The names are finally assigned to the dataframe and the volume values are converted to numeric and appended to the dataframe. The CSV files are written and the growth curve is created.

2.2.4 Inventory Layer

The Inventory Layer file creates the initial inventory layer. The `input_traza`, `input_SOC` and the `output_gcbm` parameters define the Trazabilidad file where the land usage data is stored, the folder where the soil organic carbon data is stored and the folder of the output data respectively.

Apart from this, we define the `inventory_year` which denotes the initial year of model execution. The `layer_traza` and `layer_SOC` defines the land use data (Trazabilidad) and the Soil organic carbon data from FAO respectively.

We will be generating the inventory layer from the trazabilidad file by reading the same and optionally filtering the database to accommodate the polygons that were forest at a particular point in the time-series data. Further, the classifiers are assigned, which includes the Tipofor classifier, Forest structure classifier and Origin classifier.

The Tipofor classifier gets the forest types from the 1997 map and assigns the "Bosque Mixto" and "Matorral Arborescente" forest type from the T1 column. Polygons that are "non forest" will be left as No Forestal.

```
traza$Tipofor <- as.character(traza$T_F_97)

traza$Tipofor <- ifelse(traza$T1 == "0403", "Bosque Mixto", as.character(traza$Tipofor))

traza$Tipofor <- ifelse(traza$T1 == "0304", "Matorral Arborescente", as.character(traza
↪$Tipofor))

traza$Tipofor <- ifelse(is.na(traza$Tipofor), "No forestal", as.character(traza$Tipofor))
```

The Forest structure classifier follows the same steps, albeit we add a filter to leave only 4 columns. The origin classifier is used to distinguish between forests that comes from the initial inventory (bosque inicial) for accounting purposes.

Initially, the shapefile is reprojected and the SOC is calculated by calculating the mean of each polygon. Later the age and SOC is determined by reprojecting the input shapefile to latitude and longitude and calculating the mean data in each polygon. If the GCBM detects a SOC greater than 0 in a forest pixel it will not work.

Finally, the historic and current land use assumptions are made, taking into account, FL if it is a native forest, and CL if not. The shapefile is written and the inventory layer is published in the `Output_files` directory.

2.2.5 Afforestation Layer

The afforestation layer denotes the land-use changes from non-forest to the forest and creates the afforestation layer. The `input_traza` and `output_gcbm` parameters define the folder where the land-use change data is (Trazabilidad file) and the output data folder respectively.

The `layer_traza` parameter defines the land use data (Trazabilidad) with the Trazabilidad layer having four fields (T1, T2, T3, T4) with land use data for 4 distinct years. Initially, the LUC shapefile is read and a filter is used to identify polygons that went from non-forest to forest between T1 and T2.

After assigning the year and name of the disturbance, we assign a random year between T1 and T2 for the disturbance. We calculate the post-disturbance forest type for Bosque Mixto forest type, Matrorral Arborescente forest type including the ones without forest type which will have the regional average growth, according to the REDD+ Technical annexe.

```
afor_t2$Tipofor_pa <- as.character(afor_t2$T_F_06)

afor_t2$Tipofor_pa <- ifelse(afor_t2$T2 == "0403", "Bosque Mixto", as.character(afor_t2
↪$Tipofor_pa))
```

(continues on next page)

(continued from previous page)

```
afor_t2$Tipofor_pa <- ifelse(afor_t2$T2 == "0304", "Matorral Arborescente", as.
↪character(afor_t2$Tipofor_pa))

afor_t2$Tipofor_pa <- ifelse(is.na(afor_t2$Tipofor_pa), "Promedio Regional", as.
↪character(afor_t2$Tipofor_pa))
```

In a similar manner, we calculate the post-disturbance forest structure:

```
afor_t2$Estruc_pa <- as.character(recode(afor_t2$ID_EST_06,
                                  "01" = "Adulto",
                                  "02" = "Renoval",
                                  "03" = "Adulto Renoval",
                                  "04" = "Achaparrado"
                                    ))

afor_t2$Estruc_pa <- ifelse(afor_t2$T2 == "0403", "Bosque Mixto", as.character(afor_t2
↪$Estruc_pa))

afor_t2$Estruc_pa <- ifelse(afor_t2$T2 == "0304", "Matorral Arborescente", as.
↪character(afor_t2$Estruc_pa))

afor_t2$Estruc_pa <- ifelse(is.na(afor_t2$Estruc_pa), "Promedio Regional", as.
↪character(afor_t2$Estruc_pa))

afor_t2<-afor_t2[,c("year","Perturb","Tipofor_pa","Estruc_pa")]
```

We will further repeat the same process with T3 and T4. Once completed, all the afforestation layers are bind together in a single file. We add the origin classier type to distinguish the afforestation forest and finally write the shapefile.

2.2.6 Forest Loss Layer

The Forest Loss Layer code creates the forest loss (deforestation and substitution) layer. The `input_traza` and `output_gcbm` parameters define the folder where the land-use change data is (Trazabilidad file) and the output data folder respectively.

The `layer_traza` parameter defines the land use data (Trazabilidad) with the Trazabilidad layer having four fields (T1, T2, T3, T4) with land use data for 4 distinct years. Initially, the LUC shapefile is read and a filter is used to identify polygons that went from non-forest to forest between T1 and T2. The year and name of the disturbance are assigned and a random year is assigned between T1 and T2 for the disturbance.

The important columns are left out, with 2014, 2015 and 2016 years being included and the partial layers bind together. The substitution layer is further developed, from forest to exotic plantations. The polygons that went from forest to plantations between T1 and T2 are filtered and the year and type of substitution are assessed.

```
sust_t2<- dplyr::filter(
  traza,
  T1 %in% c("0403", "040203", "040202", "040204", "040201", "0402","0304"),
  T2 %in% c("0401"))

sust_t2$year<-year_t2
sust_t2$Perturb<-"Sustitucion"
```

(continues on next page)

(continued from previous page)

```
sust_t2$year<-sample(year_t1:(year_t2),nrow(sust_t2),replace=TRUE)

sust_t2<-sust_t2[,c("year","Perturb")]
```

The disturbance year is randomized, along with disturbances from T3 to T4. The partial layers are bind together and the forest loss disturbances layer is bound together. We add the project to the WGS84 latitude and longitude and finally write the shapefile.

2.2.7 Forest Degradation Layer

The Forest Degradation layer code creates the degradation layers and disturbance matrix values, including the conservation areas. The parameters `input_pf` and `output_gcbm` define the folders where the permanent forest files are and the folder of the output data for the model.

The other parameters include the permanent forest layer in the FREL for two years, 2001 and 2010, defined as `year_t1` and `year_t2`. The number of quantiles to represent the degradation is defined as `n_quantiles`.

We read the permanent forest shapefile and the codes that indicate degradation are 2,3,4 and 10000, 10001 in the "Carta". Do note that the change in CO2 (CAM_CO2) has to be negative.

```
pf <- st_read(dsn = input_pf, layer = layer_pf)

degradation <- dplyr::filter(
  pf,
  Carta %in% c("2", "3", "4", "10000", "10001"),
  CAM_CO2 < 0)
```

A random year is defined for the degradation between 2002 and 2010 and the regional average of CO2 is calculated. The percentage of the regional CO2 that is lost in each degradation pixel is also calculated and the shapefile is divided into two parts: degradation that occurs in conservation areas and degradation that occurs outside of conservation areas.

```
degradation$year<-sample((year_t1+1):year_t2,nrow(degradation),replace=TRUE)
degradation$year

regional_CO2 <- (375.2900742 * 1.75 * 0.5)  * 0.5 * (44/12)

degradation$p_regional <- (degradation$CAM_CO2 / regional_CO2) * (-1)

degradation_c <- dplyr::filter(degradation,ca_ras_erp>0)

degradation <- dplyr::filter(degradation,ca_ras_erp==0)
```

The quantiles are further calculated and the mean of each quantile is calculated. The name of the disturbance according to the quantile (intensity level) is also fetched.

```
degradation$quantile <- cut(degradation$p_regional , breaks = quantile(degradation$p_
↪regional, seq(0,1,length.out = n_quantiles+1)),labels=1:n_quantiles, include.
↪lowest=TRUE)

means_quantile <- group_by(as.data.frame(degradation), quantile) %>% summarize(mean_quan␣
↪= mean(p_regional))
```

(continues on next page)

(continued from previous page)

```
degradation$Perturb <- paste0("Forest ", "Degradation Chile"," intensity lvl ",
↪degradation$quantile)
```

The same operation is performed with the degradation in conservation areas. Both of the degradations are joined in a single shape, and a shapefile is written which will be used as input for the tiler (GCBM).

```
degradation <- rbind(degradation,degradation_c)

degradation<-degradation[,c("year","Perturb")]

degradation<-st_transform(degradation, "+proj=longlat +datum=WGS84 +ellps=WGS84␣
↪+towgs84=0,0,0")

degradation$year<-as.integer(degradation$year)
```

We then use the information of each quantile to make the disturbance matrices to insert them into the `gcbm_input` database. For the same, we create the disturbance type CSV and start the id counting from 9003 (9001 and 9002 are deforestation and substitution).

In a for loop, we will calculate the disturbance in each iteration, with the first degradation outside conservation and then inside conservation areas.

```
for (i in 1:(n_quantiles*2)) {
  if (i==n_quantiles+1){
    lvl <- 1
  }
  if (i<= n_quantiles){
    name <- paste0("Forest Degradation Chile intensity lvl ",lvl)
  } else {
    name <- paste0("Forest Conservation Degradation Chile intensity lvl ",lvl)
  }
  code <- id
  disturbance_type <- cbind(id,disturbance_category_id,transition_land_class_id,name,
↪code)
  if (i==1){
    disturbance_types_full <- disturbance_type
  } else {
    disturbance_types_full <- rbind(disturbance_types_full,disturbance_type)
  }
  id <- id + 1
  lvl <- lvl + 1
}
```

The same process is repeated for the disturbance matrix CSV where the data frame just assigns an id and name to the disturbance matrix. It starts from 903 (901 and 902 are deforestation and substitution) and the first degradation is calculated outside conservation and then inside conservation areas.

Finally, the CSV is inputted into the `gcbm_input` database and the disturbance matrix association CSV is created. The disturbance matrix corresponds to spatial unit 36 (British Columbia and Pacific maritime). The disturbance type ID starts from 9003 while the disturbance matrix ID starts from 903. Similar to prior methods, the first degradation is calculated outside conservation and then inside conservation areas.

```
for (i in 1:(n_quantiles*2)) {
  disturbance_matrix_association <- cbind(spatial_unit_id,disturbance_type_id,
↪disturbance_matrix_id)
  if (i==1){
    disturbance_matrix_association_full <- disturbance_matrix_association
  } else {
    disturbance_matrix_association_full <- rbind(disturbance_matrix_association_full,
↪disturbance_matrix_association)
  }
  disturbance_type_id <- disturbance_type_id + 1
  disturbance_matrix_id <- disturbance_matrix_id + 1
}
```

In a similar manner, we calculate the disturbance matrix values CSV where the dataframe includes the proportion of each reservoir that goes to CO2.

```
for (i in 1:(n_quantiles*2)) {
  disturbance_matrix_id <- rep(dist_id,6)
  source_pool_id <- c(1,2,3,6,7,8)
  sink_pool_id <- rep(22,6)
  if (i<= n_quantiles){
    proportion <- rep(means_quantile$mean_quan[i],6)
  } else {
    proportion <- rep(means_quantile_c$mean_quan[i-n_quantiles],6)
  }

  disturbance_matrix <- cbind(disturbance_matrix_id,source_pool_id,sink_pool_id,
↪proportion)

  if (i==1){
    disturbance_matrix_full <- disturbance_matrix
  } else {
    disturbance_matrix_full <- rbind(disturbance_matrix_full,disturbance_matrix)
  }
  dist_id <- dist_id + 1
}
```

Finally, we write the CSV that is inserted into the `gcbm_input` database.

2.2.8 Forest Enhancement Layer

The Forest Enhancement layer code creates the forest enhancement layers and adds the new growth curves, including enhancement in conservation areas. The `input_pf` and `output_gcbm` parameters provide the path of the input folder, where the permanent forest files are, and the output data for GCBM respectively.

The other parameters include `year_t1` and `year_t2` which consists of the permanent forest layer in the FREL for 2001 and 2010 respectively. `n_quantiles` denote the number of quantiles to represent the forest enhancement. On the other hand, `age_stop_growth` and `age_max_curves` represent the age at which the growth of the forest stops and the maximum age reflected in the growth curves respectively.

A shapefile is generated with all the enhancements in the permanent forest. We read from the permanent forest shapefile and the codes that indicate enhancement are 10002, 19999, 20000 and 4, 10001 in the "Carta" field. Do note that the change in CO2 (CAM_CO2) has to be positive.

```
pf <- st_read(dsn = input_pf, layer = layer_pf)

enhancement <- dplyr::filter(
  pf,
  Carta %in% c("10002", "19999", "20000", "4", "10001"),
  CAM_CO2 > 0)

enhancement$year <- year_t1+1
```

Based on the yearly growth in volume we will create the new growth curves, which includes the yearly growth of each enhancement pixel, where we will consider the stock difference like 9 years (as in the FREL excel files). We will further divide the shapefiles into two, one for the enhancement that occurs in conservation areas while the other for the enhancement that occurs outside of conservation areas.

```
enhancement$year_grow <- enhancement$CAM_CO2 / 9

enhancement_c <- dplyr::filter(enhancement,ca_ras_erp>0)

enhancement <- dplyr::filter(enhancement,ca_ras_erp==0)
```

The quantiles are calculated, along with the mean growth of each quantile and the name of the enhancement disturbance according to the quantile (intensity level). The same process is repeated for the conservation enhancement.

```
enhancement$quantile <- cut(enhancement$year_grow , breaks = quantile(enhancement$year_
↪grow, seq(0,1,length.out = n_quantiles+1)),labels=1:n_quantiles, include.lowest=TRUE)

means_quantile <- group_by(as.data.frame(enhancement), quantile) %>% summarize(mean_quan␣
↪= mean(year_grow))

enhancement$Origen_pa <- paste0("Forest ", "Enhancement Chile"," intensity lvl ",
↪enhancement$quantile)

enhancement$Perturb <- "Enhancement"
```

Both of the enhancement shapefiles are put together and we select the necessary columns for our use case. After 9 years the forest will return to the "Bosque Inicial" (Initial forest) Origin and thus we will create a copy of the enhancement shape and change the origin classifier to Bosque Inicial. The enhancement stops one year after the FREL period (2001-2013) and we put everything together and write the shapefile as input for the tiler (GCBM).

```
enhancement_stop <- enhancement

enhancement_stop$Origen_pa <- "Bosque Inicial"

enhancement_stop$year <- year_t2+1

enhancement <- rbind(enhancement,enhancement_stop)

enhancement<-st_transform(enhancement, "+proj=longlat +datum=WGS84 +ellps=WGS84␣
↪+towgs84=0,0,0")

enhancement$year<-as.integer(enhancement$year)

write_sf(enhancement, paste0(output_gcbm, "/forest_enhancement_LosRios.shp"))
```

In a similar manner, we are going to modify the growth curves to reflect the enhancements and create the growth curves including all the forest enhancement curves. A loop that starts with the enhancement outside conservation areas and then with the enhancement inside degradation areas is initiated.

```
for (i in 1:(n_quantiles*2)) {
  if (i<= n_quantiles){
    enhancement_name <- paste0("Forest ", "Enhancement Chile"," intensity lvl ",i)
  } else {
    enhancement_name <- paste0("Forest ", "Conservation Enhancement Chile"," intensity␣
↪lvl ",i-20)
  }

  growth_base <- filter(growth,Origen %in% c("Bosque Inicial"))
  growth_base$Origen <- enhancement_name
  if (i<= n_quantiles) {
    year_grow <- unname(means_quantile$mean_quan[i])
  } else {
    year_grow <- unname(means_quantile_c$mean_quan[i-n_quantiles])
  }
  year_grow_vol <- year_grow / (1.75 * 0.5 * 0.5 * (44/12))
  growth_curve <- seq(from= 0, by=year_grow_vol,length.out = ncol(growth_base)-4)
  growth_curve <- unname(rbind(growth_curve))
  growth_curve <- growth_curve[rep(seq_len(nrow(growth_curve)), nrow(growth_base)), ]
  growth_base[,5:ncol(growth_base)] <- growth_curve
  if (i==1){
    growth_full <- rbind(growth, growth_base)
  } else {
    growth_full <- rbind(growth_full,growth_base)
  }
}
```

In a similar manner to create the growth curves for "No forestal" (non-forest) regions, a loop is initiated that starts with the enhancement outside conservation areas and then with the enhancement inside degradation areas.

```
for (i in 1:(n_quantiles*2)) {
  if (i<= n_quantiles){
    enhancement_name <- paste0("Forest ", "Enhancement Chile"," intensity lvl ",i)
  } else {
    enhancement_name <- paste0("Forest ", "Conservation Enhancement Chile"," intensity␣
↪lvl ",i-20)
  }

  growth_curve <- c("No forestal","No forestal",enhancement_name,"Not stocked",rep(as.
↪numeric(0), ncol(growth_base)-4))
  growth_curve
  growth_full <- rbind(growth_full,growth_curve)
}
```

Finally, we use the "non stocked" species so that the GCBM will ignore the growth, and append the non-stocked growth curves to the dataframe. The volume values are converted to numeric data and classifiers are converted to text. We finally write the CSV with the growth curves including the enhancement curves.