spadesCBM

Celine Boisvenue

Ian Eddy

Alex Chubaty

September 2020

# Overview

This SpaDES deck (four SpaDES modules) presents a transparent and spatially explicit implementation of the logic, pools structure, equations, and default assumptions of the Carbon Budget Model of the Canadian Forest Sector (CBM-CFS3) (Kurz et al. 2009). This was developed as a R&D tool for improving CBM and carbon modelling in the CFS. In this implementation of CBM, the entire landscape in the provided study area is simulated on a yearly basis.  
The Generic Carbon Budget Model (GCBM) (Smyth et al. 2007) simulates each pixel through the time horizon independently. The yearly processing of the entire landscape permits the link to/use of other SpaDES simulation modules that affect that landscape, making this an easier exploration tool for R&D. Other models that modify the landscape (fire, insects, harvest) that simulate the whole landscape at each time step can be used to modify the landscape and assess the carbon implication of these modifications. Contagion processes (examples: fire, seed dispersal, insect propagation) can therefore be simulated. Further, all scripts are R-based, providing a parameter and data handling platform, and a clear understanding of the model structure and parameters to any R-proficient scientist. Results will eventually be linked to an R-shiny-based interface to present results more easily to non-R proficient users.

## Background

This SpaDES-deck was developed on the SpaDES platform (a package in R; <https://cran.r-project.org/package=SpaDES>) to make it transparent, spatial explicit, and able to link to other modules/models in SpaDES. This SpaDES-deck enables the inclusion of CFS-type carbon modelling in cumulative effects evaluation, and provides an environment in which science improvements can be explored and tested. SpaDES is a Spatial Discrete Event Simulation. It is an R-package that functions as a scheduler through space and time. Being an R-based platform, is makes modelling transparent and accessible to a large community of researchers across disciplines. Note that modules or models do not have to be writting in R, but callable from R. More information on SpaDES and other openly available SpaDES modules can be found here <http://spades.predictiveecology.org/>.

# Four-module family CBM

spadesCBM modifies the carbon pools via matrix multiplications (multiplication is used in CBM-CFS3). These multiplications happen via a C++ script originally created by Scott Morken and since modified by this team to correct errors or clarify the scripts. Being in the SpaDES environment, it is meant to be run spatially explicitly which assumes that the required inputs are spatially explicit. Knowledge of the SpaDES structure would help an R-knowledgeable user to manipulate simulations but is not necessary to run the current simulations. Prior knowledge of CBM-CFS3 would also help users understand the structure of these modules, the default parameters used, but is not necessary to run simulations. All modules being written in R and the publicly available description of the SpaDES R-package imply that any R-user can learn how to run these modules and simulate carbon on a landscape. In this document, we describe all four modules necessary for simulations using spadesCBM in the traditional way as described in (Kurz et al. 2009) with simular parameters as described in (**Stinson:2011?**), which is with growth curves as the main change-agent () for the study area.

The four modules SpaDES-deck is called from a global script (below). In this project, four modules are run: CBM\_defaults, CBM\_dataPrep\_RIA\_scenario, CBM\_vol2biomass\_RIA, and CBM\_core. The code environment is on a public repository here: <https://github.com/cboisvenue/spadesCBM_RIA>. You need a GitHub account to access this and for now, **please don’t distribute this code.**

Several core utilities are provided by the [CBMutils](https://github.com/PredictiveEcology/CBMutils) package, available on GitHub.

## CBM\_defaults

This module loads all the CBM-CFS3 default parameters (Canadian defaults that is akin to the ArchiveIndex access database in CBM-CFS3). These parameters are then stored in an S4 object called cbmData and accessed throughout the simulations. This object has the following slot names:

* “turnoverRates” (15byb13 full);
* “rootParameters” (48by7 full);
* “decayParameters” (11X6 full);
* “spinupParameters”(48by4 full);
* “classifierValues”(0X0);
* “climate” (48by2 full - mean annual temp);
* “spatialUnitIds” (48by3 full);
* “slowAGtoBGTransferRate”(1by1 0.006);
* “biomassToCarbonRate”(1by1 0.5);
* “ecoIndices” (0by0);
* “spuIndices” (0by0);
* “stumpParameters” (48by5 full);
* “overmatureDeclineParameters” (48by4 full);
* “disturbanceMatrix” (426X3 - character matrix with word descriptions of disturbances [“id” “name” “description”]).

The whole sqlite db that contains the defaults is stored in this RStudio project associated with the current simulations (spadesCBM\_RIA.Rproj) in the data folder spadesCBM\_RIA/data/modules/CBM\_defaults/data/cbm\_defaults. *All parameters used in these simulations are the general Canadian defaults, and are searchable/changeable with common R functionality*. In the SpaDES environment, this module has one event (init) and does not schedule anything else. It requires the dbPath and sqlDir to run (specified in the global script below).

## CBM\_dataPrep\_studyArea\_specifyScenario

This module reads in information that is expected to be provided by the user similarly to CBM-CFS3. User provided/expected input include: - the ages of the stands/pixels (raster), - study location information (raster or shapefile) - disturbance information the user wants applied in the simulations - the growth curves and where they should be applied (which pixels) on the land base, - growth curve meta data with includes at a minimum growth curve identification and leading species from which a six column table will be built by the scripts in this module, OR the user can provide the six-column meta data directly. These column names are: + growth\_curve\_id - the identification to the spatial application of this growth curve, + growth\_curve\_component\_id (same value as above), + species - the common name for the leading species, + canfi\_species - a numeric code used to identify species for correct Boudewyn *et al.* (2007) parameter match, + genus - a four-leter codeidentifying the genus of the species, again, for correct Boudewyn *et al.* (2007) parameter matching. + forest\_type\_id - a numeric code used in CBM-CFS3 to identify if the forest is a conifer forest (1), a mixed forest (2) or a hardwood forest (3). Note that the list of canfi\_species and genus are provided in the default example and stored in the cloud-based folder to assist with links between species names, canfi\_species, and genus, or create them.

The user-provided study area is used to make a sim$masterRaster on which all maps and other calculations are based. The spatial unit raster as well as an ecozone raster are created using the sim$masterRaster. Spatial units (SPUs) are an overlay of administrative boundaries (provinces and territories) and ecozones. SPUs are the link back to the default ecological parameters assembled for CBM-CFS3 simulations in Canada. These parameters are necessary to be able to perform a simulation, you either use the defaults or have to provide alternative values for all the parameters. The location information provided by SPU is used to narrow the parameter options from CBM\_default to the ones that are specific to this study area. The CBM\_default modules needs to have been run before CBM\_dataPrep\_studyArea\_scenario. User can take the module provided here as a base and adapt it to their specific study area (example: CBM\_dataPrep\_myStudyArea). Information and data provided are also used to create a table of similar pixels to increase processing speeds (pixelGroup). The data.table produced for initial representation of the land base is saved in the simList as sim$level3DT . All necessary vectors for annual processes (simulations) are created in this module. These vectors need to be in a specific format for the C++ functions processing. A table stored in the simList (sim$mySpuDmid) links the user-provided disturbance information to the disturbance matrix identification numbers in CBM\_defaults. The .inputObjects function at this end of this module provides automatic read-in of all the necessary rasters and tables for current study area.

This module is the most specific to a study area. Users should expect this module to contain all idiosyncratic data manipulations specific to the study area and to each simulated scenario.

## CBM\_vol2biomass\_studyArea

This module is a translation module from the user-provided growth curves (cumulative ) into the biomass increments that drive simulations in CBM. It is an implementation of the stand-level biomass conversion parameters published in Boudewyn *et al.* (2007). Similarly to the CBM\_dataPrep\_myStudyArea module, this module is specific to your study area and needs your modifications.

Following the CBM-CFS3 approach, the user provides growth curves of cumulative . Each curves needs to have an identification number permitting the linking to its spatial application (user-provide in CBM\_dataPrep\_studyArea\_specifyScenario); it needs the range of ages from 0 to the oldest ages represented on the landscape; and it needs the volume associated with that age vector. The user also provides meta data for each curve (again in CBM\_dataPrep\_studyArea\_specifyScenario). All curves provided are plotted for visual inspection in a simList object named sim$volCurves. The unaltered translation of the curves into three above ground carbon pools, directly out of the application of the Boudewyn *et al.* (2007) parameters and caps, can be saved as plots in the simList object sim$plotsRawCumulativeBiomass. These translations most likely will result in non-smooth curves, or curves with odd shapes. Resulting curves do not necessarily go through a 0 intercept. For these reason, the user is obliged to smooth the curves. This seems to be done internally in CBM-CFS3. We proposed a smoothing procedure that uses a Chapman-Richards function to correct for non-plausible shapes and wiggles in the curves resulting from the translation process. Note that the purpose of the present SpaDES-deck is to emulate the CBM-CFS3 approach. Modification to this approach will be ongoing via advancements and improvements enabled by the transparency this SpaDES-deck provides. Examples of this are the use of biomass increments from other sources than translations of cumulative growth curves using the Boudewyn *et al.* (2007) algorithms. This module saves figures for users to evaluate in spadesCBM\_RIA/data/modules/CBM\_vol2biomass\_RIA/figures.

For simulations emulating CBM-CFS3, the CBM\_default, CBM\_dataPrep\_yourStudyArea\_specifyScenario, need to be run prior to running this module. This module, however, can be run independently for translation of stand-level, single-species translation into carbon per ha values (**this needs to be re-tested**). Important: it is the user’s responsability to ensure that the increments driving the simulation produced by this current module (CBM\_vol2Biomass\_studyArea) are as realistic as possible. These curves are the engine of CBM-CFS3 simulation and of the SpaDES-deck simulations that emulate them.

## CBM\_core

This module completes the simulations of the spadesCBM SpaDES deck. The CBM\_default, and CBM\_dataPrep\_studyArea\_specifyScenario or CBM\_datPrep\_studyArea need to be run prior to this module. The CBM\_vol2biomass\_studyArea either needs to be run or the user must provide annual increments, halved, for the three above ground pools (merch, foliage, other) by forest type (hardwood and softwood), hashed in the same format as produced by this module for simulations to be possible (see spadesCBMrunsSK$growth\_increments in the simList created below which gets hashed in CBM\_vol2Biomass\_SK where spadesCBMrunsSK$gcHash gets created). The module has six SpaDES-events: spinup, postSpinup, saveSpinup, annual, plot, and savePools, with saveSpinup and plot being optional. The spinup event is the init event run by default in SpaDES modules. The init event (spinup fonction) runs the traditional spinup of CBM-CFS3: where each stand (pixel or pixel group in our case) is disturbed using the disturbance specified in sim$historicDMIDs (usually wildfire for the ecozone) and re-grown using the provided above ground biomass pools, repeatedly, until the dead organic matter (DOM) pools values stabilize or the maximum number of iteration is reached (sim$maxRotations). A user can set a minimum and a maximum number of rotations, and the disturbance return interval (sim$minRotations, sim$maxRotations, sim$returnIntervals) for the spinup. In the spinup event, carbon increment estimates from the biomass estimate of Boudewyn *et al.* (2007)’s translation of the curves are used for each of the pools. The bark, branches, biomass nonmerch (equation 2 in Boudewyn) pools, and biomass sap (equation 3 in Boudewyn) are grouped under “other” in CBM. Biomass in coarse and fine roots are estimated using the above ground estimates from the increments and default parameters, one set for softwood and one set for hardwood (see root\_parameter table in the SQLite default database). To estimate carbon in all other pools, the burn-grow cycle is repeated as described above. In all spatial units in Canada, the historical disturbance is set to fire. The CBM\_default module has fire return intervals for each ecozone in Canada that can be match with the ecozone of the study area via the ecozone raster, which is either provided by the user, or create in the CBM\_dataPrep\_yourStudyArea\_specifyScenario module. Once the DOM pools have stabilized, the spinup event grows the stand/pixel/pixelGroup (still using the same growth curve) to the user-provided age of that stand/pixel/pixelGroup (“ages” defined by the age raster the user provided in CBM\_dataPrep\_yourStudyArea\_specifyScenario).

In the postSpinup event, matrices are set up for the processes that will happen in the annual event. In order the processes are: 1. disturbance (sim$allProcesses$Disturbance), 2. half growth (sim$allProcesses$Growth1), 3. dead organic matter (DOM) turnover (sim$allProcesses$DOMturnover), 4. biomass turnover (sim$allProcesses$BioTurnover) 5. OverMatureDecline (sim$allProcesses$OverMatureDecline), 6. second half of growth (sim$allProcesses$Growth1), 7. DOM decay (sim$allProcesses$DOMDecay), 8. slow decay(sim$allProcesses$SlowDecay), 9. slow mixing of dead pools’ carbon(sim$allProcesses$SlowMixing).

The annual event is where all the processes are applied. Most carbon transactions are done via C++ functions compiled via Rcpp in this module.

The ‘plot’ event uses three parameters: the initial plot time (.plotInitialTime), the interval to plot (.plotInterval), and the carbon pools to plot (poolsToPlot). The parameter poolsToPlot accepts a character vector consisting of any individual pools in cbmPools as well as totalCarbon for the sum of below ground and above ground carbon.

The event savePools is scheduled last. It currently creates a .csv file (cPoolsPixelYear.csv) that contains the carbon pool values for each unique stand/pixel type (pixelGroup) at the end of each simulation year, for all simulation years.

The event spinupDebug is currently a place holder to explore spinup results. If spinupDebug is set to FALSE, the spinup event provides a line for each stand with the initial pool values to initialize the stands/pixels for the annual simulations. The event spinupDebug was put in place to explore the results of the spinup and if TRUE, it saves **ALL** the disturbed-grow cycles and will not permit simulations.

Note: there is a special simList parameter called P(sim)$spinupDebug. This parameter is a logical parameter defined in the metadata of the CBM\_core.R module. This parameter is by default set to FAlSE. If this parameter is set to TRUE, annual simulations will not happen. This parameter modifies the output of the C++ function called Spinup() to output all the spinup runs performed to get to the maximum rotations or the stabilizing of the DOM pools. It determines if the results from the spinup will be saved as an external file. If this is set to TRUE, only saveSpinup event is schedule, not the postSpinup which starts the process of the annual simulation by scheduling the annual event. This parameter is solely for exploring the spinup procedure used in CBM.

# Current Simulations

The global script in this document will run by default, simulations for of managed forests in the northeastern corner of the province of British Colombia, Canada.

Running the following script, with all the defaults parameters, for four sceanrios. These scenarios were developed for a publication (**Boisvenue:in?** prep) where the carbon carrying capacity of the study area (FRI for Fire Return Interval) is compared to the current carbon on the landscape (presentDay), and to two levels of timber removal (harvest1 and harvest2). The presentDay runs the disturbances from 1985-2015 as presented in Hermosilla *et al.* (**Hermosilla:2016?**). The FRI scenario runs on the same landscape, 500 years of spatially explicit fire disturbance on the respective fire return interval as represented by the fire model described Armstrong & Cumming (**Armstrong:2003?**). This scenario represent the carbon carrying capacity of the landscape (**Liang:2017?**). The final two scenarios (harvest1 and harvest2) are a spatial representation of harvest and fires at contrasting levels of harvesting represented by the Ws3 model (**Paradis:2018?**).

Note that all scenarios use the same CBM\_vol2biomass\_RIA module, and that both the CBM\_default and CBM\_core modules are non-changing between any study area or scenario. Each scenario requires its own dataPrep modules. These are names as follows: - CBM\_dataPrep\_RIAfri - CBM\_dataPrep\_RIApresentDay - CBM\_dataPrep\_RIAharvest1 - CBM\_dataPrep\_RIAharvest2

Meta data for each scenario are created in the script below and this script runs four seperate simulations.

library(Require)  
Require("magrittr") # this is needed to use "%>%" below  
Require("SpaDES.core")  
  
#install\_github("PredictiveEcology/CBMutils@development")  
#load\_all("~/GitHub/PredictiveEcology/CBMutils")  
Require("PredictiveEcology/CBMutils (>= 0.0.6)")  
  
options("reproducible.useRequire" = TRUE)  
  
cacheDir <- reproducible::checkPath("cache", create = TRUE)  
moduleDir <- reproducible::checkPath("modules")  
inputDir <- reproducible::checkPath("inputs", create = TRUE)  
outputDir <- reproducible::checkPath("outputs", create = TRUE)  
scratchDir <- file.path(tempdir(), "scratch", "CBM") %>% reproducible::checkPath(create = TRUE)  
  
## TODO fix this so we can run all the times with the appropriate sims below  
timesFRI <- list(start = 2020.00, end = 2540.00)  
timesPresentDay <- list(start = 1985.00, end = 2015.00)  
timesHarvest1 <- list(start = 2020.00, end = 2099.00)  
  
  
parameters <- list(  
 CBM\_defaults = list(  
 .useCache = TRUE  
 ),  
 CBM\_vol2biomass\_RIA = list(  
 .useCache = TRUE  
 )  
)  
#Fire Return Interval  
parametersFRI <- parameters  
parametersFRI$CBM\_dataPrep\_RIAfri <- list(  
 .useCache = TRUE  
 )  
parametersFRI$CBM\_core <- list(  
 .useCache = FALSE, #"init", #c(".inputObjects", "init")  
 # .plotInterval = 5,  
 .plotInitialTime = timesFRI$start,  
 poolsToPlot = c("totalCarbon"),  
 spinupDebug = FALSE ## TODO: temporary  
 )  
#Present Day 1985-2015  
parametersPresentDay <- parameters  
parametersPresentDay$CBM\_dataPrep\_RIApresentDay <- list(  
 .useCache = TRUE  
 )  
parametersPresentDay$CBM\_core <- list(  
 .useCache = FALSE, #"init", #c(".inputObjects", "init")  
 # .plotInterval = 5,  
 .plotInitialTime = timesPresentDay$start,  
 poolsToPlot = c("totalCarbon"),  
 spinupDebug = FALSE ## TODO: temporary  
 )  
# harvest1 is the base case  
parametersHarvest1 <- parameters  
parametersHarvest1$CBM\_dataPrep\_RIAharvest1 <- list(  
 .useCache = TRUE  
 )  
  
parametersHarvest1$CBM\_core <- list(  
 #.useCache = TRUE, #"init", #c(".inputObjects", "init")  
 # .plotInterval = 5,  
 .plotInitialTime = timesHarvest1$start,  
 poolsToPlot = c("totalCarbon"),  
 spinupDebug = FALSE) ## TODO: temporary  
  
modulesFRI <- list("CBM\_defaults", "CBM\_dataPrep\_RIAfri", "CBM\_vol2biomass\_RIA", "CBM\_core")  
modulesPresentDay <- list("CBM\_defaults", "CBM\_dataPrep\_RIApresentDay", "CBM\_vol2biomass\_RIA", "CBM\_core")  
modulesHarvest1 <- list("CBM\_defaults", "CBM\_dataPrep\_RIAharvest1", "CBM\_vol2biomass\_RIA", "CBM\_core")  
#harvest2 is the less-than-base case  
modulesHarvest2 <- list("CBM\_defaults", "CBM\_dataPrep\_RIAharvest2", "CBM\_vol2biomass\_RIA", "CBM\_core")  
  
objects <- list(  
 dbPath = file.path(inputDir, "cbm\_defaults", "cbm\_defaults.db"),  
 sqlDir = file.path(inputDir, "cbm\_defaults")  
)  
  
paths <- list(  
 cachePath = cacheDir,  
 modulePath = moduleDir,  
 inputPath = inputDir,  
 rasterPath = scratchDir  
)  
  
pathsFRI <- paths  
pathsFRI$outputPath <- file.path(outputDir,"FRI")  
  
pathsPresentDay <- paths  
pathsPresentDay$outputPath <- file.path(outputDir,"presentDay")  
  
pathsHarvest1 <- paths  
pathsHarvest1$outputPath <- file.path(outputDir,"harvest1")  
  
pathsHarvest2 <- paths  
pathsHarvest2$outputPath <- file.path(outputDir,"harvest2")  
  
  
# this sets options globally  
# setPaths(  
# cachePath = cacheDir,  
# modulePath = moduleDir,  
# inputPath = inputDir,  
# outputPath = outputDir,  
# rasterPath = scratchDir  
# )  
  
quickPlot::dev.useRSGD(FALSE)  
dev()  
clearPlot()  
options(spades.moduleCodeChecks = FALSE,  
 reproducible.useMemoise = FALSE,  
 spades.recoveryMode = FALSE)  
 #reproducible.useNewDigestAlgorithm = 2)  
  
# this is the whens for the 540 years of fire runs  
whensFRI <- sort(c(timesFRI$start, timesFRI$start + 1:4 \* 100, timesFRI$end - 2:0))  
# this is for the presentDay runs  
whensPresentDay <- sort(c(timesPresentDay$start, timesPresentDay$start + c(5, 10, 15, 20, 25),   
 timesPresentDay$end - 2:0))  
# this is for the harvest runs  
whensHarvest1 <- sort(c(timesHarvest1$start, timesHarvest1$start + c(10, 30, 50, 60),   
 timesHarvest1$end - 1:0))  
  
  
outputsFRI <- as.data.frame(expand.grid(objectName = c("cbmPools", "NPP"), saveTime = whensFRI))  
outputsPresentDay <- as.data.frame(expand.grid(objectName = c("cbmPools", "NPP"), saveTime = whensPresentDay))  
outputsHarvest1 <- as.data.frame(expand.grid(objectName = c("cbmPools", "NPP"), saveTime = whensHarvest1))  
# tool for debugging, this runs only the simInit fnct, running the .inputObjects for each module.  
# presemtDaySimList <- simInit(times = timesPresentDay,  
# params = parametersPresentDay,  
# modules = modulesPresentDay,  
# objects = objects,  
# paths = pathsPresentDay,  
# outputs = outputsPresentDay,  
# loadOrder = unlist(modulesPresentDay))  
  
RIApresentDayRuns <- simInitAndSpades(times = timesPresentDay,  
 params = parametersPresentDay,  
 modules = modulesPresentDay,  
 objects = objects,  
 paths = pathsPresentDay,  
 outputs = outputsPresentDay,  
 loadOrder = unlist(modulesPresentDay),  
 debug = TRUE)  
  
RIAfriRuns <- simInitAndSpades(times = timesFRI,  
 params = parametersFRI,  
 modules = modulesFRI,  
 objects = objects,  
 paths = pathsFRI,  
 outputs = outputsFRI,  
 loadOrder = unlist(modulesFRI),  
 debug = TRUE)  
  
RIAharvest1Runs <- simInitAndSpades(times = timesHarvest1,  
 params = parametersHarvest1,  
 modules = modulesHarvest1,  
 objects = objects,  
 paths = pathsHarvest1,  
 outputs = outputsHarvest1,  
 loadOrder = unlist(modulesHarvest1),  
 debug = TRUE)  
  
RIAharvest2Runs <- simInitAndSpades(times = timesHarvest1,  
 params = parametersHarvest1,  
 modules = modulesHarvest2,  
 objects = objects,  
 paths = pathsHarvest2,  
 outputs = outputsHarvest1,  
 loadOrder = unlist(modulesHarvest2),  
 debug = TRUE)  
## Tools for debuging FRI runs, running the .inputObjects of all modules  
# RIAfriSim <- simInit(times = timesFRI,  
# params = parameters,  
# modules = modulesFRI,   
# objects = objects,   
# outputs = outputsFRI,  
# paths = pathsFRI, ## not needed when using setPaths() above  
# loadOrder = unlist(modulesFRI))  
#   
# RIAfriCBMout <- spades(RIAfriSim, debug = TRUE)

# Functions

## Disturbance matrices

A series of functions were built to help get details on CBM-CFS3 default disturbances. These are part of the R-package CBMutils required to perform simulations with this SpaDES-deck.

# Spatial Unit 27 is one of the SPU in the SK default example  
spuDist(27, "~/GitHub/spadesCBM/modules/CBM\_defaults/data/cbm\_defaults/cbm\_defaults.db")

The CBMutils::spuDist() function identifies the ID number of the CBM-CFS3 matrices that are possible in the specific spatial unit you are in. You give the spatial units id(s) you are targeting (note: a raster of the spatial units is created in CBM\_dataPrep\_yourStudyArea module from which a vector is created sim$spatialUnits) and it gives you the disturbance matrix id(s) that are possible/default in that specific SPU and a descriptive name of that disturbance matrix. It returns an R data.frame. The function CBMutils::histDist(), identifies the stand-replacing wildfire disturbance in each spatial unit. By default the most recent is selected, but the user can change that. As per CBMutils::spuDist(), you need to specify your spatial unit. Historical disturbances in CBM-CFS3 are used for “filling-up” the soil-related carbon pools in the spinup process. CBMutils::simDist() is an R function that requires a simulation list (from the SpaDES functions spades(); see ?simList) and returns a list of data.frames. Each data.frame has the descriptive name of a disturbance used in the simulation. Each data.frame has the disturbance matrix identification number from cbm\_defaults, the pool from which carbon is taken (source pools) in this specific disturbance, the pools into which carbon goes, and the proportion in which the carbon-transfers are completed. You give the CBMutils::seeDist() function one or more disturbance matrix id, and it will return the descriptive name of the disturbance, the source pools, the sink pools, and the proportions transferred. It returns a list of data.frames, one data.frame per disturbance matrix id, similarly to CBMutils::simDist().

# Other important information

The standIndex comes out of the C++ functions. This is the same as the pixelGroup in the R scripts. It is always on the right-hand side of the of assignments in the R scripts. To make sure that standIndex matches pixelGroup, rows are ordered by pixelGroup before going into C++ functions. sim$pixelGroupC: this is the data.table that gets updated every year and from which the sim$pixelGroupForAnnual is created. sim$pixelGroupForAnnual is fed into the Rcpp functions (matrix operations of annual processes and disturbances). Each line represents a pixelGroup, *i.e.*, a group of unique pixels. It has unique combinations of ages, spatial\_unit\_id, growth\_curve\_component\_id, growth\_curve\_id, ecozones, and events (which represent disturbances). pixelGroup the unique identifier for the each group. pixelGroup are recalculated every simulation year and therefore each simulation year assigns pixels to pixelGroup and these do not necessarily match between years, only non-disturbed pixels maintain their pixelGroup number from one simulation year to the next. pixelGroupC and pixelGroupForAnnual populates the vectors needed for the C++ functions (ages, gcids, spatialUnits, etc.). The data.table pixelGroupForAnnual is remade annually because annual disturbances reset the age and modify the amount of carbon in each pool. The sim$spatialDT is the long form of the pixelGroup table where each pixel individually is listed. sim$spatialDT lists all the simulated pixels in the sim$masterRaster. Not all pixels in the sim$masterRaster are usually simulated. The sim$pixelKeep is a data.table that has tracks what pixelGroup each pixel is in at each simulation year. The sim$pixelKeep gets build 1 column at a time for each year of the simulation. sim$spatialDT is rebuilt every year because disturbances change ages, one of the unique identifiers. sim$spatialDT is used to add a column called events that identify which pixels get disturbed each year from the disturbance raster information (sim$disturbanceRasters).

*NOTE* the first sim$spatialDT, used in CBM\_core spinup event does not have an events column, as fire is applied on a reoccurring schedule.

# Units

The user provides growth curves in of cumulative over time. Those curves are fed into the Boudewyn algorithms (CBM\_vol2biomass module) with its results multiplied by 0.5 to give carbon. That gives us the cumPools object that is the cumulative biomass for the three above-ground live pools in tonnes of carbon/ha. The sim$growth\_increments are the final increments (differences between the years) driving simulation which are hashed to speed up the processing time. These are in metric tonnes of carbon per ha. All other pools and fluxes are also in metric tonnes of carbon per ha.

# References

Boudewyn, P, X Song, S Magnussen, and M D Gillis. 2007. “Model-Based, Volume-to-Biomass Conversion for Forested and Vegetated Land in Canada.” BC-X-411. Victoria, BC: Natural Resource Canada, Pacific Forestry Centre.

Kurz, W. A., C. C. Dymond, T. M. White, G. Stinson, C. H. Shaw, G. J. Rampley, C. Smyth, et al. 2009. “CBM-CFS3: A Model of Carbon-Dynamics in Forestry and Land-Use Change Implementing IPCC Standards.” *Ecological Modelling* 220 (4): 480–504. <https://doi.org/10.1016/j.ecolmodel.2008.10.018>.

Smyth, C. E., A. J. Dugan, M. Olguin, R. Birdsey, C. Wayson, A. Alanís, and W. A. Kurz. 2007. “A Synthesis of Climate Change Mitigation Options Based on Regional Case Studies of the North American Forest Sector Using a Harmonized Modeling Approach.” BC-X-455. Victoria, BC: Natural Resource Canada, Pacific Forestry Centre. <https://doi.org/10.13140/RG.2.2.29177.70247>.