Supplement. Quantifying natural disturbances using a large-scale dendrochronological reconstruction to guide forest management

Cada V., Trotsiuk V., ….

2/8/2019

Table of Contents

# Libraries and settings

library(dplyr)   
library(tibble)  
library(tidyr)  
library(zoo)  
library(pracma)  
library(readr)

# Functions

priorGrowth <- function(x, windowLength = 10){  
 rollapply( x,   
 width = windowLength,  
 FUN = mean,  
 fill = NA,  
 align = "right",  
 na.rm = T,  
 partial = TRUE)  
}  
  
followGrowth <- function(x, windowLength = 10){  
 rollapply( lead(x, 1),   
 width = windowLength,  
 FUN = mean,  
 fill = NA,  
 align = "left",  
 na.rm = T,  
 partial = TRUE)  
}  
  
peakDetection <- function(x, threshold, mindist = 20, nups = 2){  
 #' @description identify the index of year when release event occur  
 #' @param x a vector of absolute increase change  
 #' @param threshold a minimum ai value in mm  
 #' @param mindist minimum distance between two consecutive peaks in years  
 #' @param nups number of increasing steps before the peak  
   
 x <- ifelse(is.na(x), -0.2, x)  
   
 x <- findpeaks(x,   
 minpeakheight = threshold,  
 minpeakdistance = mindist,  
 nups = nups)   
   
 if(is.null(x)){  
 NA  
 }else{  
 matrix(x, ncol = 4)[,2]  
 }  
}  
  
keepRelease <- function(year, type, n = 20){  
 #' @description calculate the distance between gap origin and releases  
 #' @param year the vector of years for event  
 #' @param type type of the event (release or gap)  
 #' @param n number of years to be checked  
   
 keep <- rep('yes', length(year))  
   
 if(any(type %in% 'gap')){  
 diffyear <- year - year[type %in% 'gap']  
 keep[diffyear < n & type %in% 'release'] <- 'no'  
 }  
 keep  
}  
  
growthCalculate <- function(data = data, windowLength = 10){  
 #' @description take the list of data prepared by 'dist\_get\_data' function and calculate the growth change, plus age and dbh of the trees  
 #' @param data a list of tree tables  
 #' @param windowLength the length of the window for ai calculation  
   
 # data quality check  
 options(error = NULL) # not to enter debug mode  
   
 # perform the checks  
 if(!is.list(data)) stop('The input data is not a list of three tables')  
 if(!identical(c('core',"dist\_param","ring"), ls(data))) stop('The input data tables dont match with required')  
   
 # calculate the age, dbh, and the growth change  
 inner\_join(  
 data$ring,  
 data$core,  
 by = 'core\_id'  
 ) %>%  
 arrange(core\_id, year) %>%  
 group\_by(core\_id) %>%  
 mutate(dbh\_growth = ifelse(row\_number() == 1, incr\_mm + missing\_mm, incr\_mm),  
 dbh\_growth = cumsum(dbh\_growth) \* 2,  
 dbh\_mm = ifelse(is.na(dbh\_mm), max(dbh\_growth), dbh\_mm),  
 dbh\_coef = max(dbh\_mm) / max(dbh\_growth),  
 dbh\_growth = dbh\_growth \* dbh\_coef,  
 age = year - min(year) + missing\_years + 1,  
 pg = priorGrowth(incr\_mm, windowLength = windowLength),  
 fg = followGrowth(incr\_mm, windowLength = windowLength),  
 ai = fg - pg) %>%  
 select(dist\_param, tree\_id, core\_id, ring\_id, year, incr\_mm, age, dbh\_mm = dbh\_growth, ai, fg, pg) ->  
 data$ring  
   
 return(data)  
}  
  
releaseCalculate <- function(data = NULL, gapAge = c(5:15), nprol = 7){  
 #' @description function calculate the releases for individual trees  
 #' @param data a list of three dataframes, output of growthCalculate function  
 #' @param nprol number of years to consider that release is sustaind  
 #' @param gapAge age of the tree when it shall be tested for gap origin  
   
 # data quality check  
 options(error = NULL) # not to enter debug mode  
   
 # perform the checks  
 if(!is.list(data)) stop('The input data is not a list of three tables')  
 if(!identical(c('core',"dist\_param","ring"), ls(data))) stop('The input data tables dont match with required')  
   
 aith <- data$dist\_param %>% select(dist\_param, ai\_mm) %>% deframe()  
 gapth <- data$dist\_param %>% select(dist\_param, gap\_mm) %>% deframe()  
   
 # calculate releases  
 data$ring %>%  
 arrange(year) %>%  
 group\_by(core\_id) %>%  
 mutate(event = ifelse(row\_number() %in% peakDetection(x = ai, threshold = aith[first(as.character(dist\_param))], nups = 1, mindist = 30), 'release', NA),  
 event = ifelse(lead(fg, nprol) <= pg, NA, event),  
 event = ifelse(lag(pg, nprol) >= fg, NA, event)) %>%  
 filter(!is.na(event)) %>%  
 select(core\_id, year, event) ->  
 release.event  
   
 # calculate the gap origin   
 data$ring %>%   
 filter(age %in% gapAge) %>%  
 arrange(year) %>%  
 group\_by(core\_id) %>%  
 summarise(dist\_param = first(dist\_param),  
 gapGrowth = mean(incr\_mm, na.rm = T),  
 N = n(),  
 year = min(year)) %>%  
 filter(N >= 5,  
 gapGrowth >= gapth[as.character(dist\_param)]) %>%  
 mutate(event = 'gap') %>%  
 select(core\_id, year, event) ->  
 gap.event  
   
 # add those that don't have any event  
 data$ring %>%  
 filter(!core\_id %in% c(unique(gap.event$core\_id), unique(release.event$core\_id))) %>%  
 group\_by(core\_id) %>%  
 summarise(year = min(year)) %>%  
 mutate(event = 'no event') ->  
 no.event  
   
 # add together the events  
 bind\_rows(release.event, gap.event, no.event) %>%  
 arrange(year) %>%  
 group\_by(core\_id) %>%  
 mutate(keeprel = keepRelease(year, event, n = 30)) %>%  
 ungroup() %>%  
 filter(keeprel %in% 'yes') %>%  
 inner\_join(., data$ring, by = c('core\_id', 'year')) %>%  
 select(ring\_id, dist\_param, year, age, dbh\_mm, ai, event) ->  
 data$event  
   
 return(data)  
   
}  
  
mdsFun <- function(ca, k = 30, bw = 5, st = 7){  
 #' @description return a vector of the fited KDE function  
 #' @param ca arranged vector of the canopy area values  
 #' @param k a windows length, default 30  
 #' @param bw a smoothing bandwidth to be used, default = 5  
 #' @param st a standartization value, to scale back to canopy area  
   
 rollapply( ca,   
 width = k,  
 FUN = function(x){n <- length(x); density(1:n, weights = x, bw = bw, n = n)$y[round((n+1)/2)]\* 100/st},  
 fill = 0,  
 align = "center",  
 partial = TRUE)  
}

# Data

plots\_clean <- read\_csv("plots\_clean\_app.csv")  
  
dist\_patches <- read\_csv("dist\_patches\_app.csv") %>% filter(stand\_size > 20)  
  
data\_list <- list(  
 dist\_param = read\_csv("dist\_param\_app.csv"),  
 core = read\_csv("core\_app.csv"),  
 ring = read\_csv("ring\_app.csv")  
)  
  
data\_all <- read\_csv("data\_all\_app.csv")

# Calculate the growth change and releases at the tree level

data.growth <- growthCalculate(data = data\_list, windowLength = 10)  
  
data.release <- releaseCalculate(data = data.growth, gapAge = c(5:15), nprol = 7)

# Calculate the Kernel density and find peaks at the plot level

## Disturbance history data

data\_all %>%   
 rowwise() %>%  
 mutate(ca = eval(parse(text = dbh\_ca\_f))) %>%   
 ungroup() %>%  
 do({  
 x <- .  
 inner\_join(  
 x %>% group\_by(plotid, species, event, year) %>% summarise(ca = sum(ca)),  
 x %>% distinct(tree\_id, .keep\_all = T) %>% group\_by(plotid) %>% summarise(ca\_f = sum(ca), n = n()) %>% filter(n >= 5),  
 by = 'plotid'  
 )   
 }) %>%  
 ungroup() %>%  
 mutate(ca = ca \* 100 / ca\_f) %>%  
 arrange(plotid, year) %>%  
 filter(year %in% c(1600:2010)) %>%  
 select(plotid, species, event, year, ca) %>%  
 gather(plot\_type, value, -year, -ca, -plotid) %>%  
 mutate(plot\_type = factor(plot\_type, levels = c('species', 'event')),  
 Species = factor(value, levels = c('Picea', 'Fagus', 'Abies', 'Acer', 'Pinus', 'Others', 'gap', 'release', 'no event')))->  
 data.dist

## Calculate the Kernel density (MDS) and the moving sums

data.dist %>%  
 filter(plot\_type == 'species') %>%  
 group\_by(plotid, year) %>%  
 summarise(ca = sum(ca)) %>%  
 group\_by(plotid) %>%  
 complete(year = 1600:2030, fill = list(ca = 0)) %>%  
 mutate(value = mdsFun(ca, k = 30, bw = 5, st = 7),  
 ca = round(ca, digits = 2),  
 severity = rollapply(ca, width = 11, FUN = sum, fill = 0)) %>%  
 ungroup() %>%  
 filter(year %in% c(1600:2010)) ->  
 data.mds

## Detect the peaks in Kernel density

data.mds %>%  
 group\_by(plotid) %>%  
 filter(row\_number() %in% peakDetection(x = value, threshold = 10, nups = 5, mindist = 10)) %>%   
 mutate(method = '10\_10\_5') %>%  
 ungroup() ->  
 data.peaks

# Calculate the Kernel density and find peaks at the stand level

## Data

data.peaks %>%   
 filter(severity > 10) %>%  
 mutate(value = round(value, digits = 5)) %>%  
 inner\_join(., plots\_clean, by = "plotid") %>%  
 select(plotid, Xjtsk, Yjtsk, country, newstand, year, ca\_per = ca, kde = value, severity) ->  
 dist\_eventsb

## Calculate and bootstrap density function for chronologies of plot level events

dist\_eventsb %>%   
 select(country, newstand, plotid, year) %>%  
 filter(!is.na(newstand)) %>%  
 group\_by(country, newstand) %>%   
 mutate(nplots = length(unique(plotid))) %>%  
 filter(year %in% c(1811:1989)) %>%   
 slice(rep(1:n(), each = 1000)) %>%  
 ungroup() %>%   
 mutate(rep = rep(1:1000, times = nrow(.) / 1000)) ->  
 standlevel\_densityf\_boot  
  
standlevel\_densityf\_boot %>%  
 distinct(., country, newstand, plotid, rep) %>%   
 group\_by(country, newstand, rep) %>%  
 sample\_n(., size = 10, replace = TRUE) %>%  
 left\_join(., standlevel\_densityf\_boot, by = c("country", "newstand", "plotid", "rep")) %>%  
 group\_by(country, newstand, year, rep) %>%  
 summarise(nevents = n(),  
 nplots = mean(nplots),  
 freq = nevents / nplots) %>%  
 group\_by(country, newstand, rep) %>%  
 complete(year = c(1780:2020), fill = list(freq = 0)) %>%  
 mutate(density\_pre = mdsFun(freq, k = 30, bw = 5, st = 7),  
 density = rollapply(density\_pre, width = 5, FUN = mean, fill = 0)) %>%   
 ungroup() ->  
 standlevel\_densityf\_boot

## Find peaks in all (bootstrapped) density functions and select most frequent peaks

standlevel\_densityf\_boot %>%  
 filter(dplyr::row\_number() %in% peakDetection(x = density, threshold = 0.00001, nups = 5, mindist = 10),  
 year %in% c(1812:1988)) %>%  
 group\_by(country, newstand, year) %>%  
 summarise(freq = n()/100) %>%  
 group\_by(country, newstand) %>%  
 complete(year = c(1780:2020), fill = list(freq = 0)) %>%  
 mutate(freqsmooth = mdsFun(freq, k = 11, bw = 1, st = 7)/10) %>%  
 filter(row\_number() %in% peakDetection(x = freqsmooth, threshold = 0.00001, nups = 0, mindist = 10)) %>%  
 unite(peakid, c("country", "newstand", "year"), sep = "-", remove = FALSE) %>%  
 select(peakid, country, newstand, year, freqsmooth) %>%  
 ungroup() ->   
 standlevel\_peaks

## Group plot level events (join plot level events with closest stand level peaks)

dist\_events\_dt <- data.table::data.table(subset(subset(dist\_eventsb, !is.na(newstand)), year %in% c(1811:1989)),  
 key = c("country", "newstand", "year"))  
  
standlevel\_peaks\_dt <- data.table::data.table(standlevel\_peaks, key = c("country", "newstand", "year"))  
  
dist\_eventsb %>%  
 left\_join(.,  
 data.frame(standlevel\_peaks\_dt[dist\_events\_dt,  
 list(country, newstand, year, peakid, plotid),  
 roll = "nearest"]) %>%  
 mutate(peakyear = as.numeric(substr(peakid, nchar(peakid) - 3, nchar(peakid)))) %>%   
 select(plotid, year, peakid, peakyear),   
 by = c("plotid", "year")) %>%  
 select(plotid, X = Xjtsk, Y = Yjtsk, country, newstand, year, ca\_per, kde, peakid, peakyear, severity) ->  
 standlevel\_dist\_events\_joined

# Calculate the disturbance characteristics and their rotation periods

## Stand proportion disturbed

standlevel\_dist\_events\_joined %>%   
 filter(!is.na(peakyear)) %>%  
 group\_by(newstand, peakyear) %>%  
 summarize(nplots\_disturb = n()) %>%  
 left\_join(.,  
 plots\_clean %>%   
 filter(!is.na(newstand)) %>%  
 group\_by(locality, country, landscape, newstand) %>%  
 summarize(nplots = n()),  
 by = "newstand") %>%  
 mutate(plotsprop\_disturb = nplots\_disturb/nplots) ->  
 plotsprop\_disturb

## Rotation period of disturbance severity

standlevel\_dist\_events\_joined %>%   
 group\_by(country, newstand, plotid) %>%   
 summarize(nevents = n()) %>%  
 slice(rep(1:n(), each = 1000)) %>%  
 ungroup() %>%   
 mutate(rep = rep(1:1000, times = nrow(.) / 1000)) %>%  
 group\_by(rep) %>%  
 sample\_n(., size = 50, replace = TRUE) %>%  
 left\_join(., standlevel\_dist\_events\_joined, by = c("country", "newstand", "plotid")) %>%  
 mutate(severityclass = floor(severity/5) \* 5) %>%  
 group\_by(plotid, rep) %>%   
 mutate(ch\_length = 1990 - min(year)) ->  
 rotation\_severity\_boot  
  
rotation\_severity\_boot %>%   
 group\_by(rep, plotid) %>%   
 summarize(ch\_length = mean(ch\_length)) %>%  
 group\_by(rep) %>%   
 summarize(rep\_length = sum(ch\_length)) ->  
 rep\_length  
  
rotation\_severity\_boot %>%   
 group\_by(rep, severityclass) %>%  
 summarize(distevents\_n = n()) %>%  
 group\_by(rep) %>%  
 complete(severityclass = seq(10, 150, 5), fill = list(distevents\_n = 0)) %>%  
 left\_join(., rep\_length, by = "rep") %>%  
 arrange(rep, desc(severityclass)) %>%  
 group\_by(rep) %>%  
 mutate(rotation\_s = rep\_length/cumsum(distevents\_n)) %>%  
 group\_by(severityclass) %>%   
 summarize(conf95 = quantile(rotation\_s, probs = 0.975),   
 conf05 = quantile(rotation\_s, probs = 0.025)) %>%  
 filter(severityclass < 100) ->  
 rotation\_severity\_confidence  
  
standlevel\_dist\_events\_joined %>%  
 left\_join(., plots\_clean, by = "plotid") %>%  
 group\_by(landscape, plotid) %>%   
 summarize(ch\_length = 1990 - min(year)) %>%  
 group\_by(landscape) %>%   
 summarize(length = sum(ch\_length)) ->  
 length\_landscapes  
  
standlevel\_dist\_events\_joined %>%   
 select(plotid, year, severity) %>%   
 mutate(severityclass = floor(severity/5) \* 5) %>%  
 left\_join(., plots\_clean, by = "plotid") %>%  
 group\_by(landscape, severityclass) %>%  
 summarize(distevents\_n = n()) %>%  
 arrange(landscape, desc(severityclass)) %>%  
 left\_join(., length\_landscapes, by = "landscape") %>%  
 mutate(rotation\_s = length/cumsum(distevents\_n)) %>%  
 select(-distevents\_n) %>%   
 filter(severityclass < 100) %>%  
 ungroup()->  
 rotation\_severity\_landscapes  
  
length <- sum((standlevel\_dist\_events\_joined %>% group\_by(plotid) %>% summarize(ch\_length = 1990 - min(year)))$ch\_length)  
  
standlevel\_dist\_events\_joined %>%   
 select(plotid, year, severity) %>%   
 mutate(severityclass = floor(severity/5) \* 5) %>%  
 group\_by(severityclass) %>%   
 summarize(distevents\_n = n()) %>%  
 arrange(desc(severityclass))%>%  
 mutate(rotation\_s = length/cumsum(distevents\_n)) %>%  
 select(-distevents\_n) ->  
 rotation\_severity

## Rotation period of disturbance patch size

dist\_patches %>%   
 group\_by(newstand) %>%   
 summarize(nevents = n()) %>%  
 slice(rep(1:n(), each = 1000)) %>%  
 ungroup() %>%   
 mutate(rep = rep(1:1000, times = nrow(.) / 1000)) ->  
 rotation\_patchsize\_boot  
  
rotation\_patchsize\_boot %>%  
 group\_by(rep) %>%  
 sample\_n(., size = 20, replace = TRUE) %>%  
 left\_join(., dist\_patches, by = "newstand") %>%  
 mutate(patchsizeclass = floor(patch\_area/5) \* 5) %>%  
 group\_by(newstand, rep) %>%   
 mutate(ch\_length = 1990 - min(peakyear)) ->  
 rotation\_patchsize\_boot  
  
rotation\_patchsize\_boot %>%   
 group\_by(rep, newstand) %>%   
 summarize(ch\_length = mean(ch\_length)) %>%  
 group\_by(rep) %>%   
 summarize(rep\_length = sum(ch\_length)) ->  
 rep\_length  
  
rotation\_patchsize\_boot %>%   
 group\_by(rep, patchsizeclass) %>%  
 summarize(distevents\_n = n()) %>%  
 group\_by(rep) %>%  
 complete(patchsizeclass = seq(0, 60, 5), fill = list(distevents\_n = 0)) %>%  
 left\_join(., rep\_length, by = "rep") %>%  
 arrange(rep, desc(patchsizeclass)) %>%  
 group\_by(rep) %>%  
 mutate(rotation\_s = rep\_length/cumsum(distevents\_n)) %>%  
 group\_by(patchsizeclass) %>%   
 summarize(conf95 = quantile(rotation\_s, probs = 0.975),  
 conf05 = quantile(rotation\_s, probs = 0.025)) ->  
 rotation\_patchsize\_confidence  
  
dist\_patches %>%  
 left\_join(.,   
 plots\_clean %>%   
 filter(!is.na(newstand)) %>%  
 group\_by(locality, country, landscape, newstand) %>%  
 summarize(nplots = n()),  
 by = "newstand") %>%  
 group\_by(landscape, newstand) %>%   
 summarize(ch\_length = 1990 - min(peakyear)) %>%  
 group\_by(landscape) %>%   
 summarize(lengthp = sum(ch\_length))->  
 length\_patch\_landscapes  
  
dist\_patches %>%   
 select(newstand, peakyear, patch\_area) %>%   
 mutate(patchsizeclass = floor(patch\_area/5) \* 5) %>%  
 left\_join(.,   
 plots\_clean %>%   
 filter(!is.na(newstand)) %>%  
 group\_by(locality, country, landscape, newstand) %>%  
 summarize(nplots = n()),   
 by = "newstand") %>%  
 group\_by(landscape, patchsizeclass) %>%  
 summarize(distevents\_n = n()) %>%  
 complete(patchsizeclass = seq(0, 60, 5), fill = list(distevents\_n = 0)) %>%  
 arrange(landscape, desc(patchsizeclass)) %>%  
 left\_join(., length\_patch\_landscapes, by = "landscape") %>%  
 mutate(rotation\_patch = lengthp/cumsum(distevents\_n)) %>%  
 select(-distevents\_n) %>%  
 ungroup()->  
 rotation\_patchsize\_landscapes  
  
length\_patch <- sum((dist\_patches %>% group\_by(newstand) %>% summarize(ch\_length = 1990 - min(peakyear)))$ch\_length)  
  
dist\_patches %>%   
 select(newstand, peakyear, patch\_area) %>%   
 mutate(patchsizeclass = floor(patch\_area/5) \* 5) %>%  
 group\_by(patchsizeclass) %>%   
 summarize(distevents\_n = n()) %>%  
 arrange(desc(patchsizeclass))%>%  
 mutate(rotation\_s = length\_patch/cumsum(distevents\_n)) %>%  
 select(-distevents\_n) ->  
 rotation\_patchsize

## Rotation period of stand proportion disturbed

plotsprop\_disturb %>%   
 group\_by(newstand) %>%   
 summarize(nevents = n()) %>%  
 slice(rep(1:n(), each = 1000)) %>%  
 ungroup() %>%   
 mutate(rep = rep(1:1000, times = nrow(.) / 1000)) ->  
 rotation\_prop\_boot  
  
rotation\_prop\_boot %>%  
 group\_by(rep) %>%  
 sample\_n(., size = 20, replace = TRUE) %>%  
 left\_join(., plotsprop\_disturb, by = "newstand") %>%  
 mutate(proportionclass = floor(100 \* plotsprop\_disturb/5) \* 5) %>%  
 group\_by(newstand, rep) %>%   
 mutate(ch\_length = 1990 - min(peakyear)) ->  
 rotation\_prop\_boot  
  
rotation\_prop\_boot %>%   
 group\_by(rep, newstand) %>%   
 summarize(ch\_length = mean(ch\_length)) %>%  
 group\_by(rep) %>%   
 summarize(rep\_length = sum(ch\_length)) ->  
 rep\_length  
  
rotation\_prop\_boot %>%   
 group\_by(rep, proportionclass) %>%  
 summarize(distevents\_n = n()) %>%  
 group\_by(rep) %>%  
 complete(proportionclass = seq(0, 125, 5), fill = list(distevents\_n = 0)) %>%  
 left\_join(., rep\_length, by = "rep") %>%  
 arrange(rep, desc(proportionclass)) %>%  
 group\_by(rep) %>%  
 mutate(rotation\_s = rep\_length/cumsum(distevents\_n)) %>%  
 group\_by(proportionclass) %>%   
 summarize(conf95 = quantile(rotation\_s, probs = 0.975),  
 conf05 = quantile(rotation\_s, probs = 0.025)) ->  
 rotation\_prop\_confidence  
  
plotsprop\_disturb %>%  
 group\_by(landscape, newstand) %>%   
 summarize(ch\_length = 1990 - min(peakyear)) %>%  
 group\_by(landscape) %>%   
 summarize(lengthpr = sum(ch\_length))->  
 length\_prop\_landscapes  
  
plotsprop\_disturb %>%   
 select(newstand, landscape, peakyear, plotsprop\_disturb) %>%  
 mutate(proportionclass = floor(100 \* plotsprop\_disturb/5) \* 5) %>%  
 group\_by(landscape, proportionclass) %>%   
 summarize(distevents\_n = n()) %>%  
 complete(proportionclass = seq(0, 125, 5), fill = list(distevents\_n = 0)) %>%  
 arrange(landscape, desc(proportionclass))%>%  
 left\_join(., length\_prop\_landscapes, by = "landscape") %>%  
 mutate(rotation\_prop = lengthpr/cumsum(distevents\_n)) %>%  
 select(-distevents\_n) %>%   
 filter(proportionclass < 105) %>%  
 ungroup()->  
 rotation\_prop\_landscapes  
  
length\_prop <- sum((plotsprop\_disturb %>% group\_by(newstand) %>% summarize(ch\_length = 1990 - min(peakyear)))$ch\_length)  
  
plotsprop\_disturb %>%   
 select(newstand, peakyear, plotsprop\_disturb) %>%  
 mutate(proportionclass = floor(100 \* plotsprop\_disturb/5) \* 5) %>%  
 group\_by(proportionclass) %>%   
 summarize(distevents\_n = n()) %>%  
 arrange(desc(proportionclass))%>%  
 mutate(rotation\_s=length\_prop/cumsum(distevents\_n)) %>%  
 select(-distevents\_n) ->  
 rotation\_prop