Fetching FIA data for sugarbush growth models

Neal Maker

February 02, 2024

Introduction

Here I pull together a dataset of remeasured trees from the US Forest Service's Forest Inventory and Analysis (FIA) records for the Northern Forest region, which will allow us to build sugarbush tree growth, mortality, and ingrowth models. It will be filtered to only include subplots with sufficient sugar maple stocking such that they could be viable sugarbushes. We would like to have at least 3 measurements for each subplot (they're remeasured about every 5 years) but also would like to only use data that was collected using the current standard plot design, which was created in the mid 1990s. At least 3 measurements for each subplot will allow us to build an ingrowth/regeneration model that accounts for logging removals that were made ahead of time. (As in, we see how much basal area was removed from year 0 to year 5, which will become a predictor for ingrowth that happens between years 5 and 10, and maybe also for ingrowth between years 10 and 15.) I'll see how many plots there are with 3, 4, and 5 measurements, and how ingrowth changes through time following a logging operation, and then We can decide how many measurements to use for the ingrowth model. We expect to use dbh, bal, and previous logging removals as independent variables for all the models.

The Northern Forest region was chosen because it covers a fairly large geographic extent while still representing a coherent ecological region, in which trees can be expected to follow a similar set of behaviors. Models developed with the dataset should be relatively unbiased for individual sugarbushes within the region, while still being trained on enough data (we hope) to paint an accurate picture. The US Northern Forest is defined here as including Oswego, Oneida, Lewis, Jefferson, Saint Lawrence, Herkimer, Fulton, Hamilton, Franklin, Essex, Clinton, and Warren Counties in New York; Franklin, Orleans, Essex, Chittenden, Lamoille, Caledonia, Washington, Addison, Orange, and Grand Isle Counties in Vermont; Coos, Grafton, and Carroll Counties in New Hampshire; and Oxford, Franklin, Somerset, Androscoggin, Kennebec, Waldo, Hancock, Washington, Penobscot, Piscataquis, and Aroostook Counties in Maine.

```r
library("dplyr")
library("lubridate")
library("english")

# Define States & counties (FIPS codes) in Northern Forest region --------

states <- c("NY", "VT", "NH", "ME")

NY_counties <- c(75, 65, 49, 45, 89, 43, 35, 41, 33, 31, 19, 113)
VT_counties <- c(11, 19, 9, 7, 15, 5, 23, 1, 17, 13)
NH_counties <- c(7, 9, 3)
ME_counties <- c(17, 7, 25, 1, 11, 27, 9, 29, 19, 21, 3)
```

Methods

FIA tree, plot, condition, and GRM data are downloaded from the FIA DataMart[1] in the form of state-specific csv files, which were generated by the Forest Service from the FIA Oracle database tables. Most of the necessary data is from the trees table, but the conditions table has some useful site information and the plots table has measurement dates and plot IDs that we'll need. The GRM table has growth data, which is mostly modeled. I think we don't want any of that, but I can't remember, so we'll download it in case and probably discard it later.

Potentially useful fields are retained in all the tables. In the trees table, seedlings with diameters measured at the root collar instead of at breast height get thrown out right off the bat, since they're hard to work with alongside the data from larger trees. All the downloaded data are current as of February 02, 2024.

```r
################################################################################
# Import FIA data
################################################################################

# Fetch FIA tree, growth, plot, & condition data for Northern Forest states
# and filter to keep only northern forest counties
# (this may take a while; ~140MB of downloads + reading)

temp <- tempfile()

for(state in states){
  download.file(paste("https://apps.fs.usda.gov/fia/datamart/CSV/",
                      state, "_TREE.zip", sep = ""),
                temp, mode = "wb")
  unzip(temp, paste(state, "_TREE.csv", sep = ""))
}

TREE <- lapply(states, function(x){
  read.csv(paste(x, "_TREE.csv", sep = ""), header = T) %>%
    filter(COUNTYCD %in% eval(as.name(paste(x, "_counties", sep = ""))),
           DIAHTCD == 1) %>% # excludes seedlings measured at root collar
    select(CN, PLT_CN, SUBP, PREV_TRE_CN, CONDID, DIA, SPCD, STATUSCD) %>%
    mutate(ba_ac = if_else(DIA >= 5,
                           # poles & larger from 24' radius subplots
                           # saplings from 6.8' radius microplots
                           0.005454*DIA^2*(43560/(pi*24^2)),
                           0.005454*DIA^2*(43560/(pi*6.8^2))))
})

## SUBPLOTS MATTER!!!!
# Subplots may have different sizes depending on the plot design, found in
# PLOT$DESIGNCD (see database guide, appendix i)
# I can just keep DESIGNCD == 1 (the main standard) and lose some data,
# or I can account for the various designs when I calculate TREE$ba_ac
# (above; which would mean calculating ba_ac after combining
# states' data and joining nf_trees to nf_plots).
# Update: it's probably harder b/c they severed the plot codes to older
# inventories to keep people from seeing real coordinates

# DESIGN CODES:
```

[1] https://apps.fs.usda.gov/fia/datamart/datamart.html

```r
# 1:4 used 1999 - present
# 11:15 used 1994 - 1996
# 100 used 1982 & 1983
# 101:104 used 1991 - 1998
# 101 was continued through 2008
# 105:120 variousy used 1991 - 1993


for(state in states){
  download.file(paste("https://apps.fs.usda.gov/fia/datamart/CSV/", state,
                      "_PLOT.zip", sep = ""),
                temp, mode = "wb")
  unzip(temp, paste(state, "_PLOT.csv", sep = ""))
}

PLOT <- lapply(states, function(x){
  read.csv(paste(x, "_PLOT.csv", sep = ""), header = T) %>%
    filter(COUNTYCD %in% eval(as.name(paste(x, "_counties", sep = "")))) %>%
    select(CN, PREV_PLT_CN, DESIGNCD, MEASYEAR, MEASMON,
           MEASDAY, LAT, LON) %>%
    rename(PLT_CN = CN)
})


for(state in states){
  download.file(paste("https://apps.fs.usda.gov/fia/datamart/CSV/", state,
                      "_COND.zip", sep = ""),
                temp, mode = "wb")
  unzip(temp, paste(state, "_COND.csv", sep = ""))
}

COND <- lapply(states, function(x){
  read.csv(paste(x, "_COND.csv", sep = ""), header = T) %>%
    filter(COUNTYCD %in% eval(as.name(paste(x, "_counties", sep = "")))) %>%
    select(PLT_CN, CONDID, ALSTKCD, SITECLCD)
})


for(state in states){
  download.file(paste("https://apps.fs.usda.gov/fia/datamart/CSV/", state,
                      "_TREE_GRM_COMPONENT.zip", sep = ""),
                temp, mode = "wb")
  unzip(temp, paste(state, "_TREE_GRM_COMPONENT.csv", sep = ""))
}

GRM <- lapply(states, function(x){
  read.csv(paste(x, "_TREE_GRM_COMPONENT.csv", sep = ""), header = T) %>%
    select(TRE_CN, STATECD, DIA_BEGIN, DIA_MIDPT, DIA_END,
           ANN_DIA_GROWTH, ANN_HT_GROWTH)
})

# Combine states' data
```

```r
nf_trees <- do.call(rbind, TREE)
nf_plots <- do.call(rbind, PLOT)
nf_conds <- do.call(rbind, COND)
nf_grms <- do.call(rbind, GRM)

# delete temporary objects and downloaded files

unlink(temp)

remove(TREE, PLOT, COND, GRM, temp, state)

for(state in states){
  file.remove(paste(state, "_TREE.csv", sep = ""))
  file.remove(paste(state, "_PLOT.csv", sep = ""))
  file.remove(paste(state, "_COND.csv", sep = ""))
  file.remove(paste(state, "_TREE_GRM_COMPONENT.csv", sep = ""))
}
```

```r
################################################################################
# Group species
################################################################################

# Make names and factor levels more intuitive ------------------------------

species_codes <-
  c(12, 43, 68, 70, 71, 91, 94, 95, 96, 97, 105, 123, 125, 126, 129,
    130, 136, 202, 221, 241, 261, 310, 313, 314, 315, 316, 317, 318,
    319, 320, 331, 341, 355, 356, 357, 367, 370, 371, 372, 373, 375,
    379, 391, 400, 402, 403, 407, 409, 421, 462, 491, 500, 531, 540,
    541, 543, 544, 546, 552, 601, 602, 621, 651, 655, 660, 661, 663,
    680, 693, 701, 712, 731, 741, 742, 743, 744, 746, 760, 761, 762,
    763, 764, 771, 802, 804, 806, 816, 823, 832, 833, 837, 901, 920,
    922, 923, 926, 934, 935, 936, 937, 950, 951, 970, 972, 975, 977,
    999)

species <-
  c("fir", "other softwood", "cedar", "tamarack", "tamarack",
    "norway spruce", "white spruce", "other softwood",
    "other softwood", "red spruce", "other softwood", "other softwood",
    "red pine", "other softwood", "white pine",

    "scots pine", "other softwood", "other softwood", "other softwood",
    "cedar", "hemlock", "other hardwood", "other hardwood",
    "hard maple", "striped maple", "red maple", "silver maple",
    "hard maple",

    "shrubs", "hard maple", "other hardwood", "other hardwood",
    "shrubs", "shrubs", "shrubs", "other hardwood", "other hardwood",
    "yellow birch", "other hardwood", "other hardwood", "paper birch",

    "gray birch", "shrubs", "hickory", "hickory", "hickory", "hickory",
    "hickory", "other hardwood", "other hardwood", "shrubs", "shrubs",
    "beech", "ash",
    "ash", "black ash", "ash", "ash", "other hardwood", "butternut",
    "other hardwood", "other hardwood", "other hardwood",
    "other hardwood", "apple", "apple", "apple",

    "other hardwood", "other hardwood", "hophornbeam",
    "other hardwood", "other hardwood", "aspen", "cottonwood", "aspen",
    "cottonwood", "aspen", "shrubs", "pin cherry", "black cherry",

    "shrubs", "shrubs", "shrubs", "white oak", "white oak", "red oak",
    "white oak", "white oak", "white oak", "red oak", "red oak",
    "other hardwood", "shrubs",

    "black willow", "shrubs", "shrubs", "shrubs", "shrubs", "shrubs",
    "shrubs", "basswood", "basswood", "elm", "elm", "elm", "elm",

    "other hardwood")

names(species) <- as.character(species_codes)

nf_trees$SPCD <- factor(unname(species[as.character(nf_trees$SPCD)]),
                     levels = levels(factor(species))) # standardize levels


################################################################################
# Calculate BAL & plot BA for each tree
################################################################################

# Calculates overtopping basal area (BAL) assuming all input trees are in
# same plot and ba is adjusted based on tpa:
pbal <- function(dbh, ba){
  sapply(dbh, function(x){
    index <- dbh > x
    return(sum(ba[index]))
  })
}


# Add BAL
nf_trees <- nf_trees %>%
  mutate(bal = NA,
         ba = NA)

# Note that this only calculates ending basal areas for trees that lived
nf_trees[nf_trees$STATUSCD == 1,] <- nf_trees[nf_trees$STATUSCD == 1,] %>%
  group_by(PLT_CN, SUBP) %>%
  mutate(bal = pbal(DIA, ba_ac),
         ba = sum(ba_ac, na.rm = T)) %>%
  ungroup()
```

Some of the variables retained from the FIA tables were recorded in the field, while others were determined remotely by the FIA Program. I also calculate plot basal area and tree overtopping basal area here after the fact, by grouping trees into their respective plots and subplots.

To start, I'll sort out 5 measurements for each plot, working backwards in time and only keeping records from earlier measurements in plots that can be linked to later measurements:

```
################################################################################
# Make linkable tables for multiple measurements
################################################################################

# make ending data, which will be used to find data from earlier measurements of the same subplots (but

nf4 <- nf_trees %>%
  left_join(nf_plots, by = "PLT_CN") %>%
  left_join(nf_conds, by = c("PLT_CN", "CONDID")) %>%
  rename(cn_4 = CN,
         cn_3_from_4 = PREV_TRE_CN,
         plt_cn_4 = PLT_CN,
         plt_cn_3_from_4 = PREV_PLT_CN,
         subp_4 = SUBP,
         condid_4 = CONDID,
         spp_4 = SPCD,
         dbh_4 = DIA,
         statuscd_4 = STATUSCD,
         MEASYEAR_4 = MEASYEAR,
         MEASMON_4 = MEASMON,
         MEASDAY_4 = MEASDAY,
         ba1_4 = ba,
         bal1_4 = bal,
         stocking_4 = ALSTKCD,
         site_class_4 = SITECLCD,
         designcd_4 = DESIGNCD) %>%
  select(-LAT, -LON) #%>% # we'll just use starting values
  # filter(!is.na(plt_cn_3_from_4)) # remove those that weren't already measured

# now need data from next-to last measurement
nf3 <- nf_trees %>%
  left_join(nf_plots, by = "PLT_CN") %>%
  left_join(nf_conds, by = c("PLT_CN", "CONDID")) %>%
  filter(PLT_CN %in% nf4$plt_cn_3_from_4) %>%
  rename(cn_3 = CN,
         cn_2_from_3 = PREV_TRE_CN,
         plt_cn_3 = PLT_CN,
         plt_cn_2_from_3 = PREV_PLT_CN,
         subp_3 = SUBP,
         condid_3 = CONDID,
         spp_3 = SPCD,
         dbh_3 = DIA,
         statuscd_3 = STATUSCD,
         MEASYEAR_3 = MEASYEAR,
         MEASMON_3 = MEASMON,
         MEASDAY_3 = MEASDAY,
         ba1_3 = ba,
         bal1_3 = bal,
         stocking_3 = ALSTKCD,
         site_class_3 = SITECLCD,
         designcd_3 = DESIGNCD) %>%
  select(-LAT, -LON) #%>% # we'll just use starting values
```

```r
  # filter(!is.na(plt_cn_2_from_3)) # remove those that weren't already measured

# data from third-to-last measurement
nf2 <- nf_trees %>%
  left_join(nf_plots, by = "PLT_CN") %>%
  left_join(nf_conds, by = c("PLT_CN", "CONDID")) %>%
  filter(PLT_CN %in% nf3$plt_cn_2_from_3) %>%
  rename(cn_2 = CN,
         cn_1_from_2 = PREV_TRE_CN,
         plt_cn_2 = PLT_CN,
         plt_cn_1_from_2 = PREV_PLT_CN,
         subp_2 = SUBP,
         condid_2 = CONDID,
         spp_2 = SPCD,
         dbh_2 = DIA,
         statuscd_2 = STATUSCD,
         MEASYEAR_2 = MEASYEAR,
         MEASMON_2 = MEASMON,
         MEASDAY_2 = MEASDAY,
         ba1_2 = ba,
         bal1_2 = bal,
         stocking_2 = ALSTKCD,
         site_class_2 = SITECLCD,
         designcd_2 = DESIGNCD) %>%
  select(-LAT, -LON) #%>% # we'll just use starting values
  # filter(!is.na(plt_cn_1_from_2)) # remove those that weren't already measured

# data from fourth-to-last measurement
nf1 <- nf_trees %>%
  left_join(nf_plots, by = "PLT_CN") %>%
  left_join(nf_conds, by = c("PLT_CN", "CONDID")) %>%
  filter(PLT_CN %in% nf2$plt_cn_1_from_2) %>%
  rename(cn_1 = CN,
         cn_0_from_1 = PREV_TRE_CN,
         plt_cn_1 = PLT_CN,
         plt_cn_0_from_1 = PREV_PLT_CN,
         subp_1 = SUBP,
         condid_1 = CONDID,
         spp_1 = SPCD,
         dbh_1 = DIA,
         statuscd_1 = STATUSCD,
         MEASYEAR_1 = MEASYEAR,
         MEASMON_1 = MEASMON,
         MEASDAY_1 = MEASDAY,
         ba1_1 = ba,
         bal1_1 = bal,
         stocking_1 = ALSTKCD,
         site_class_1 = SITECLCD,
         designcd_1 = DESIGNCD) %>%
  select(-LAT, -LON) #%>% # we'll just use starting values
  # filter(!is.na(plt_cn_0_from_1)) # remove those that weren't already measured

# ...and data from fifth-to-last measurement (starting data if we use them all)
```

```r
nf0 <- nf_trees %>%
  left_join(nf_plots, by = "PLT_CN") %>%
  left_join(nf_conds, by = c("PLT_CN", "CONDID")) %>%
  filter(PLT_CN %in% nf1$plt_cn_0_from_1) %>%
  rename(cn_0 = CN,
         plt_cn_0 = PLT_CN,
         subp_0 = SUBP,
         condid_0 = CONDID,
         spp_0 = SPCD,
         dbh_0 = DIA,
         statuscd_0 = STATUSCD,
         MEASYEAR_0 = MEASYEAR,
         MEASMON_0 = MEASMON,
         MEASDAY_0 = MEASDAY,
         ba1_0 = ba,
         bal1_0 = bal,
         stocking_0 = ALSTKCD,
         site_class_0    = SITECLCD,
         designcd_0 = DESIGNCD,
         lat_0 = LAT,
         lon_0 = LON)

################################################################################
# Potential Sugarbush Parameters (for next part)
################################################################################
bamin <- 50
pcthm <- 60
maxsite <- 5
```

Now we can explore data available for the ingrowth model, by starting with the oldest data, filtering out non-sugarbushes, seeing how much data is left, seeing how many plots had some logging, and doing a cursory exploration of ingrowth patterns through time. Based on what we find, we can decide weather to keep the oldest data for the model creation, or discard it and start over with the next oldest. Potential sugarbushes (which we'll keep in the data) need to have at least 50 sqft/ac ba, at least 60% of ba in hard maple, and a site class not over 5 (growing at least 50 cuft/ac/yr; VT class II or better).

```r
################################################################################
# Find sugarbush-potential subplots and evaluate logging done
################################################################################
sugar_plots <- nf0 %>% filter(designcd_0 ==1, # only current plot design
                              statuscd_0 == 1) %>% # only live
  mutate(plt = paste0(plt_cn_0, subp_0)) %>% # unique subplot IDs
  group_by(plt) %>%
  summarise(ba = mean(ba1_0), # I verified that this matches sum(ba_ac)
            hm = 100 * sum(ba_ac[spp_0 == "hard maple"]) / sum(ba_ac),
            site = mean(site_class_0)) %>%
  filter(ba >= bamin, hm >= pcthm, site <= maxsite)

sugar_plots <- sugar_plots$plt

nf01 <- nf0 %>% full_join(nf1, by = c("cn_0" = "cn_0_from_1")) %>%
  mutate(plt = case_when(!is.na(plt_cn_0) & !is.na(subp_0) ~
                         paste0(plt_cn_0, subp_0),
                         !is.na(plt_cn_0_from_1) & !is.na(subp_1) ~
```

```
                           paste0(plt_cn_0_from_1, subp_1),
                         TRUE ~ "5")) %>%
  filter(statuscd_0 == 1, # don't care about trees that started dead
         !is.na(statuscd_1), # newer records missing for live trees TO: bad data or stumps that couldn'
         plt != "5", # removes newer records that don't tie back to old plots
         plt %in% sugar_plots) %>%
  mutate(cut_01 = if_else(statuscd_0 == 1 & statuscd_1 == 3,
                          TRUE, FALSE), # logging removals
         died_01 = if_else(statuscd_0 == 1 & statuscd_1 == 2,
                           TRUE, FALSE)) # natural mort.

################################################################################
# Link all data to look at ingrowth patterns
################################################################################
```

There are 54 potential-sugarbush subplots with at least 5 measurements, of which 9 had at least some cutting during the first remeasurement period (from measurement 0 to measurement 1).