Fetching FIA data for sugarbush growth models

Neal Maker

February 06, 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.

```
library("dplyr")
library("lubridate")
library("english")
library("ggplot2")

# 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 06, 2024.

```
################################################################################
# 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
```

```
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 = ""))
}
```

```
################################################################################
# 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 note that this can contain multiple measurements from
# the same subplot; for a plot that's been measured 4 times, we'll get four
# different 'records' in this table, three of which will have PREV_PLT_CN. Most
# or all of these multiples will be filtered out later when we remove plots with
# insufficient measurements, depending on how many measurements we decide to
# use.)

nf4 <- nf_trees %>%
  left_join(nf_plots, by = "PLT_CN") %>%
  left_join(nf_conds, by = c("PLT_CN", "CONDID")) %>%
  filter(DESIGNCD == 1) %>%
  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,
         ba_ac_4 = ba_ac,
         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(DESIGNCD == 1) %>%
  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,
         ba_ac_3 = ba_ac,
```

```
         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
  # 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(DESIGNCD == 1) %>%
  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,
         ba_ac_2 = ba_ac,
         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(DESIGNCD == 1) %>%
  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,
```

```
         ba_ac_1 = ba_ac,
         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)
nf0 <- nf_trees %>%
  left_join(nf_plots, by = "PLT_CN") %>%
  left_join(nf_conds, by = c("PLT_CN", "CONDID")) %>%
  filter(DESIGNCD == 1) %>%
  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,
         ba_ac_0 = ba_ac,
         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) %>%
  select(-PREV_TRE_CN, -PREV_PLT_CN)

##############################################################################
# Make index that connects all subplots through time & common subplot id to all
##############################################################################
nf0$sub0 <- paste0(nf0$plt_cn_0, nf0$subp_0)
nf1$sub1 <- paste0(nf1$plt_cn_1, nf1$subp_1)
nf2$sub2 <- paste0(nf2$plt_cn_2, nf2$subp_2)
nf3$sub3 <- paste0(nf3$plt_cn_3, nf3$subp_3)
nf4$sub4 <- paste0(nf4$plt_cn_4, nf4$subp_4)
nf1$sub0from1 <- paste0(nf1$plt_cn_0_from_1, nf1$subp_1)
nf2$sub1from2 <- paste0(nf2$plt_cn_1_from_2, nf2$subp_2)
nf3$sub2from3 <- paste0(nf3$plt_cn_2_from_3, nf3$subp_3)
nf4$sub3from4 <- paste0(nf4$plt_cn_3_from_4, nf4$subp_4)

# have to work from oldest forward to join
```

```
subs <- data.frame(s0 = unique(nf0$sub0))
subs$s1 <- sapply(subs$s0, function(i){
  nf1$sub1[which(nf1$sub0from1 == i)][1]
})
# but add in newer plots that didn't have previous measurements
subs <- rbind(subs, data.frame(s0 = NA,
                               s1 = unique(nf1$sub1[!(nf1$sub1 %in% subs$s1)])))
subs$s2 <- sapply(subs$s1, function(i){
  nf2$sub2[which(nf2$sub1from2 == i)][1]
})
subs <- rbind(subs, data.frame(s0 = NA,
                               s1 = NA,
                               s2 = unique(nf2$sub2[!(nf2$sub2 %in% subs$s2)])))
subs$s3 <- sapply(subs$s2, function(i){
  nf3$sub3[which(nf3$sub2from3 == i)][1]
})
subs <- rbind(subs, data.frame(s0 = NA,
                               s1 = NA,
                               s2 = NA,
                               s3 = unique(nf3$sub3[!(nf3$sub3 %in% subs$s3)])))
subs$s4 <- sapply(subs$s3, function(i){
  nf4$sub4[which(nf4$sub3from4 == i)][1]
})
subs <- rbind(subs, data.frame(s0 = NA,
                               s1 = NA,
                               s2 = NA,
                               s3 = NA,
                               s4 = unique(nf4$sub4[!(nf4$sub4 %in% subs$s4)])))
# one unique subplot id that's independent of plot cn & subp
subs$id <- 1:nrow(subs)

################################################################################
# Remove entries from later tables that are already in earlier tables
################################################################################
nf1 <- nf1[which(!(nf1$cn_1 %in% nf0$cn_0)), ]
nf2 <- nf2[which(!(nf2$cn_2 %in% nf0$cn_0)), ]
nf2 <- nf2[which(!(nf2$cn_2 %in% nf1$cn_1)), ]
nf3 <- nf3[which(!(nf3$cn_3 %in% nf0$cn_0)), ]
nf3 <- nf3[which(!(nf3$cn_3 %in% nf1$cn_1)), ]
nf3 <- nf3[which(!(nf3$cn_3 %in% nf2$cn_2)), ]
nf4 <- nf4[which(!(nf4$cn_4 %in% nf0$cn_0)), ]
nf4 <- nf4[which(!(nf4$cn_4 %in% nf1$cn_1)), ]
nf4 <- nf4[which(!(nf4$cn_4 %in% nf2$cn_2)), ]
nf4 <- nf4[which(!(nf4$cn_4 %in% nf3$cn_3)), ]

################################################################################
# Make one master, multitemporal table
################################################################################
nf <-  # na_matches = "never" keeps it from joining NAs
  full_join(nf0, nf1, by = c("cn_0" = "cn_0_from_1"), na_matches = "never") %>%
  full_join(nf2, by = c("cn_1" = "cn_1_from_2"), na_matches = "never") %>%
  full_join(nf3, by = c("cn_2" = "cn_2_from_3"), na_matches = "never") %>%
  full_join(nf4, by = c("cn_3" = "cn_3_from_4"), na_matches = "never")
```

```
# sub is our unique subplot id
nf$sub <- NA
nf$sub[is.na(nf$sub) & !is.na(nf$sub4)] <-
  subs$id[match(nf$sub4[is.na(nf$sub) & !is.na(nf$sub4)], subs$s4)]
nf$sub[is.na(nf$sub) & !is.na(nf$sub3)] <-
  subs$id[match(nf$sub3[is.na(nf$sub) & !is.na(nf$sub3)], subs$s3)]
nf$sub[is.na(nf$sub) & !is.na(nf$sub2)] <-
  subs$id[match(nf$sub2[is.na(nf$sub) & !is.na(nf$sub2)], subs$s2)]
nf$sub[is.na(nf$sub) & !is.na(nf$sub1)] <-
  subs$id[match(nf$sub1[is.na(nf$sub) & !is.na(nf$sub1)], subs$s1)]
nf$sub[is.na(nf$sub) & !is.na(nf$sub0)] <-
  subs$id[match(nf$sub0[is.na(nf$sub) & !is.na(nf$sub0)], subs$s0)]

################################################################################
# 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 subplots that have been measured five times, 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).

```
# Do all records for each later plot point to the same earlier plot?
all((nf4 %>% filter(!is.na(plt_cn_3_from_4)) %>%
  group_by(plt_cn_4) %>%
  summarize(same_pre = length(unique(plt_cn_3_from_4)) == 1))$same_pre)
```

```
## [1] TRUE
```

```
# Do all records with the same earlier plot point to the same later plot?
all((nf4 %>% filter(!is.na(plt_cn_3_from_4)) %>%
  group_by(plt_cn_3_from_4) %>%
  summarize(same_post = length(unique(plt_cn_4)) == 1))$same_post)
```

```
## [1] TRUE
```

```
# Both should be TRUE.
```

```
################################################################################
# Find sugarbush-potential subplots (at start) and evaluate logging done
################################################################################
sugar_plots <- nf %>%
  group_by(sub) %>%
  summarise(sub = sub[1],
            ba = mean(ba1_0, na.rm = T),
            ba2 = sum(ba_ac_0[statuscd_0 == 1], na.rm = T), # to verify data isn't screwed up
```

```
            hm = 100 * sum(ba_ac_0[spp_0 == "hard maple" & statuscd_0 == 1], na.rm = T) /
              sum(ba_ac_0[statuscd_0 == 1], na.rm = T),
            site = mean(site_class_0, na.rm = T))
sugar_plots$ba[is.na(sugar_plots$ba)] <- 0

# should be TRUE
all(sugar_plots$ba == sugar_plots$ba2)
```

```
## [1] TRUE
```

```
# plots w/ 5 measurements
sub5 <- subs$id[!is.na(subs$s0) & !is.na(subs$s1) & !is.na(subs$s2) &
                  !is.na(subs$s3) & !is.na(subs$s4)]

sugar_plots <- sugar_plots %>%
  filter(ba >= bamin, hm >= pcthm, site <= maxsite, sub %in% sub5)

sugar_plots <- unique(sugar_plots$sub)
sugar_plots <- sugar_plots[!is.na(sugar_plots)]

sugar5 <- nf %>%
  filter(sub %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.
         cut_01 = if_else(is.na(cut_01), FALSE, cut_01),
         died_01 = if_else(is.na(died_01), FALSE, died_01),
         ingrowth_01 = if_else(is.na(cn_0) & !is.na(cn_1), TRUE, FALSE),
         cut_12 = if_else(statuscd_1 == 1 & statuscd_2 == 3,
                          TRUE, FALSE), # logging removals
         died_12 = if_else(statuscd_1 == 1 & statuscd_2 == 2,
                           TRUE, FALSE), # natural mort.
         cut_12 = if_else(is.na(cut_12), FALSE, cut_12),
         died_12 = if_else(is.na(died_12), FALSE, died_12),
         ingrowth_12 = if_else(is.na(cn_1) & !is.na(cn_2), TRUE, FALSE),
         cut_23 = if_else(statuscd_2 == 1 & statuscd_3 == 3,
                          TRUE, FALSE), # logging removals
         died_23 = if_else(statuscd_2 == 1 & statuscd_3 == 2,
                           TRUE, FALSE), # natural mort.
         cut_23 = if_else(is.na(cut_23), FALSE, cut_23),
         died_23 = if_else(is.na(died_23), FALSE, died_23),
         ingrowth_23 = if_else(is.na(cn_2) & !is.na(cn_3), TRUE, FALSE),
         cut_34 = if_else(statuscd_3 == 1 & statuscd_4 == 3,
                          TRUE, FALSE), # logging removals
         died_34 = if_else(statuscd_3 == 1 & statuscd_4 == 2,
                           TRUE, FALSE), # natural mort.
         cut_34 = if_else(is.na(cut_34), FALSE, cut_34),
         died_34 = if_else(is.na(died_34), FALSE, died_34),
         ingrowth_34 = if_else(is.na(cn_3) & !is.na(cn_4), TRUE, FALSE)) %>%
  select(sub, spp_0, dbh_0, statuscd_0, ba_ac_0, bal1_0, ba1_0, MEASYEAR_0, MEASMON_0, MEASDAY_0,
         spp_1, dbh_1, statuscd_1, ba_ac_1, bal1_1, ba1_1, MEASYEAR_1, MEASMON_1, MEASDAY_1,
         spp_2, dbh_2, statuscd_2, ba_ac_2, bal1_2, ba1_2, MEASYEAR_2, MEASMON_2, MEASDAY_2,
```

```
          spp_3, dbh_3, statuscd_3, ba_ac_3, bal1_3, ba1_3, MEASYEAR_3, MEASMON_3, MEASDAY_3,
          spp_4, dbh_4, statuscd_4, ba_ac_4, bal1_4, ba1_4, MEASYEAR_4, MEASMON_4, MEASDAY_4,
          cut_01, died_01, ingrowth_01, cut_12, died_12, ingrowth_12,
          cut_23, died_23, ingrowth_23, cut_34, died_34, ingrowth_34)

# newer records missing for live trees T0: bad data or stumps that couldn't be
# found? I think we just throw them out:
# sugar5 <- sugar5[!is.na(sugar5$statuscd_1), ]
```

There are 53 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). Here is the average ingrowth (stems/subplot) in different remeasurement periods for plots that had some logging in the first period, and plots that had no logging:

```
# plot ingrowth by time in plots w/ & w/o cutting
sugar5 %>% group_by(sub) %>%
  summarize(cutting01 = any(cut_01), in01 = sum(ingrowth_01),
            in12 = sum(ingrowth_12), in23 = sum(ingrowth_23),
            in34 = sum(ingrowth_34)) %>%
  group_by(cutting01) %>%
  summarize(nplots = n(), ingrowth01 = mean(in01), ingrowth12 = mean(in12),
            ingrowth23 = mean(in23), ingrowth34 = mean(in34))
```

cutting01	nplots	ingrowth01	ingrowth12	ingrowth23	ingrowth34
FALSE	44	0.7500000	1.022727	1.045454	1.340909
TRUE	9	0.6666667	1.222222	1.777778	1.555556

There's some evidence of increased ingrowth in the remeasurement periods following logging (ingrowth12, ingrowth23, and ingrowth34), but there isn't a lot of data. Definitely not enough to see how ingrowth is affected by different cutting intensities. I'll start over and include plots that have only 4 measurements, to increase the amount of data we have to work with.

```
##############################################################################
# Find sugarbush-potential subplots (at start) and evaluate logging done
##############################################################################
sugar_plots1 <- nf %>%
  group_by(sub) %>%
  summarise(sub = sub[1],
            ba = mean(ba1_1, na.rm = T),
            ba2 = sum(ba_ac_1[statuscd_1 == 1], na.rm = T), # to verify data isn't screwed up
            hm = 100 * sum(ba_ac_1[spp_1 == "hard maple" & statuscd_1 == 1], na.rm = T) /
              sum(ba_ac_1[statuscd_1 == 1], na.rm = T),
            site = mean(site_class_1, na.rm = T))
sugar_plots1$ba[is.na(sugar_plots1$ba)] <- 0

# should be TRUE
all(sugar_plots1$ba == sugar_plots1$ba2)
```

```
## [1] TRUE
```

```
# plots w/ 4 measurements
sub4meas <- subs$id[!is.na(subs$s1) & !is.na(subs$s2) & !is.na(subs$s3) &
                  !is.na(subs$s4)]

sugar_plots1 <- sugar_plots1 %>%
  filter(ba >= bamin, hm >= pcthm, site <= maxsite, sub %in% sub4meas)

sugar_plots1 <- unique(sugar_plots1$sub)
sugar_plots1 <- sugar_plots1[!is.na(sugar_plots1)]

sugar4 <- nf %>%
  filter(sub %in% sugar_plots1) %>%
  mutate(cut_12 = if_else(statuscd_1 == 1 & statuscd_2 == 3,
                          TRUE, FALSE), # logging removals
         died_12 = if_else(statuscd_1 == 1 & statuscd_2 == 2,
                           TRUE, FALSE), # natural mort.
         cut_12 = if_else(is.na(cut_12), FALSE, cut_12),
         died_12 = if_else(is.na(died_12), FALSE, died_12),
         ingrowth_12 = if_else(is.na(cn_1) & !is.na(cn_2), TRUE, FALSE),
         cut_23 = if_else(statuscd_2 == 1 & statuscd_3 == 3,
                          TRUE, FALSE), # logging removals
         died_23 = if_else(statuscd_2 == 1 & statuscd_3 == 2,
                           TRUE, FALSE), # natural mort.
         cut_23 = if_else(is.na(cut_23), FALSE, cut_23),
         died_23 = if_else(is.na(died_23), FALSE, died_23),
         ingrowth_23 = if_else(is.na(cn_2) & !is.na(cn_3), TRUE, FALSE),
         cut_34 = if_else(statuscd_3 == 1 & statuscd_4 == 3,
                          TRUE, FALSE), # logging removals
         died_34 = if_else(statuscd_3 == 1 & statuscd_4 == 2,
                           TRUE, FALSE), # natural mort.
         cut_34 = if_else(is.na(cut_34), FALSE, cut_34),
         died_34 = if_else(is.na(died_34), FALSE, died_34),
         ingrowth_34 = if_else(is.na(cn_3) & !is.na(cn_4), TRUE, FALSE)) %>%
  select(sub,
         spp_1, dbh_1, statuscd_1, ba_ac_1, bal1_1, ba1_1, MEASYEAR_1, MEASMON_1, MEASDAY_1,
         spp_2, dbh_2, statuscd_2, ba_ac_2, bal1_2, ba1_2, MEASYEAR_2, MEASMON_2, MEASDAY_2,
         spp_3, dbh_3, statuscd_3, ba_ac_3, bal1_3, ba1_3, MEASYEAR_3, MEASMON_3, MEASDAY_3,
         spp_4, dbh_4, statuscd_4, ba_ac_4, bal1_4, ba1_4, MEASYEAR_4, MEASMON_4, MEASDAY_4,
         cut_12, died_12, ingrowth_12,
         cut_23, died_23, ingrowth_23, cut_34, died_34, ingrowth_34)
```

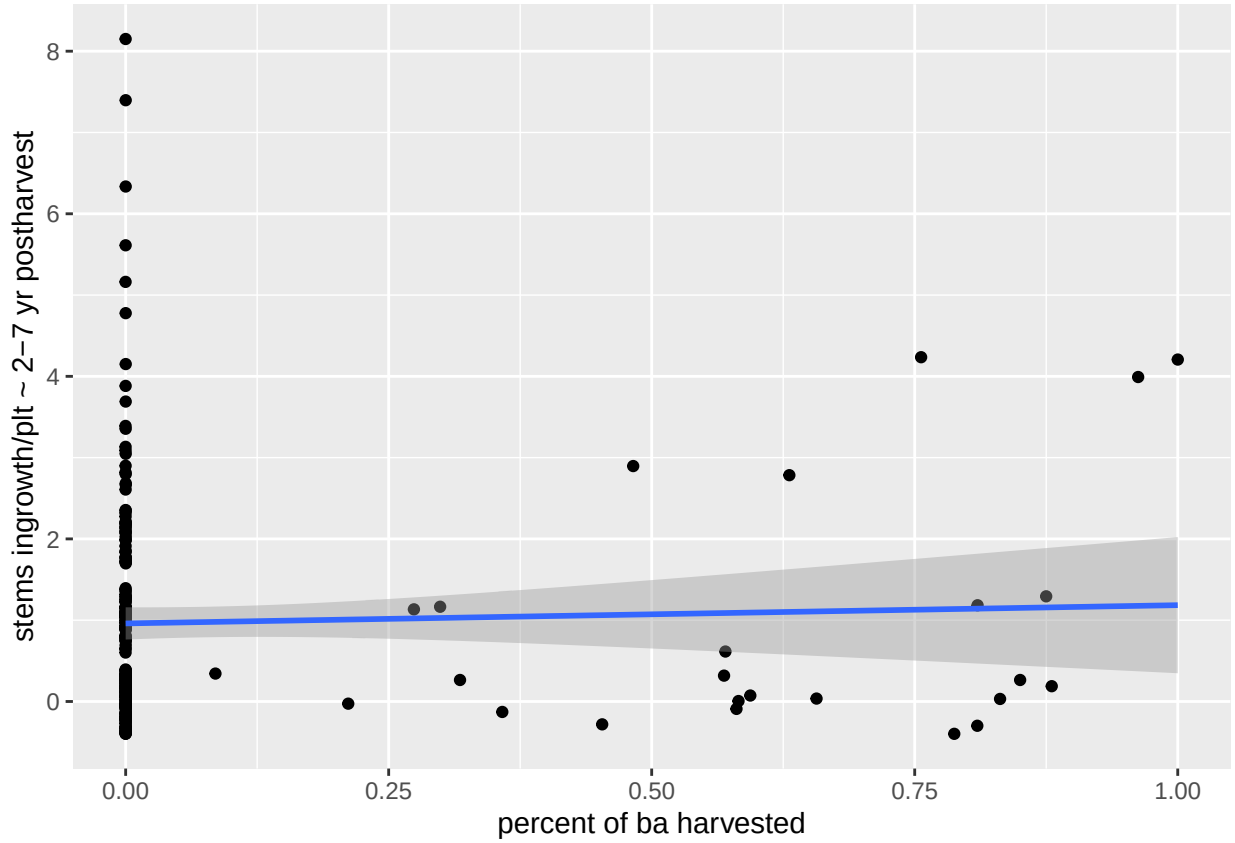
There are 212 potential-sugarbush subplots with at least 4 measurements, of which 25 had at least some cutting during the first remeasurement period (from measurement 1 to measurement 2). Here is the average ingrowth (stems/subplot) in different remeasurement periods for plots that had some logging in the first period, and plots that had no logging:

```
# ingrowth by time in plots w/ & w/o cutting
sugar4 %>% group_by(sub) %>%
  summarize(cutting12 = any(cut_12), in12 = sum(ingrowth_12),
            in23 = sum(ingrowth_23), in34 = sum(ingrowth_34)) %>%
  group_by(cutting12) %>%
  summarize(nplots = n(), ingrowth12 = mean(in12),
            ingrowth23 = mean(in23), ingrowth34 = mean(in34))
```

cutting12	nplots	ingrowth12	ingrowth23	ingrowth34
FALSE	187	0.8716578	0.9839572	1.010695
TRUE	25	0.8400000	0.9200000	3.400000

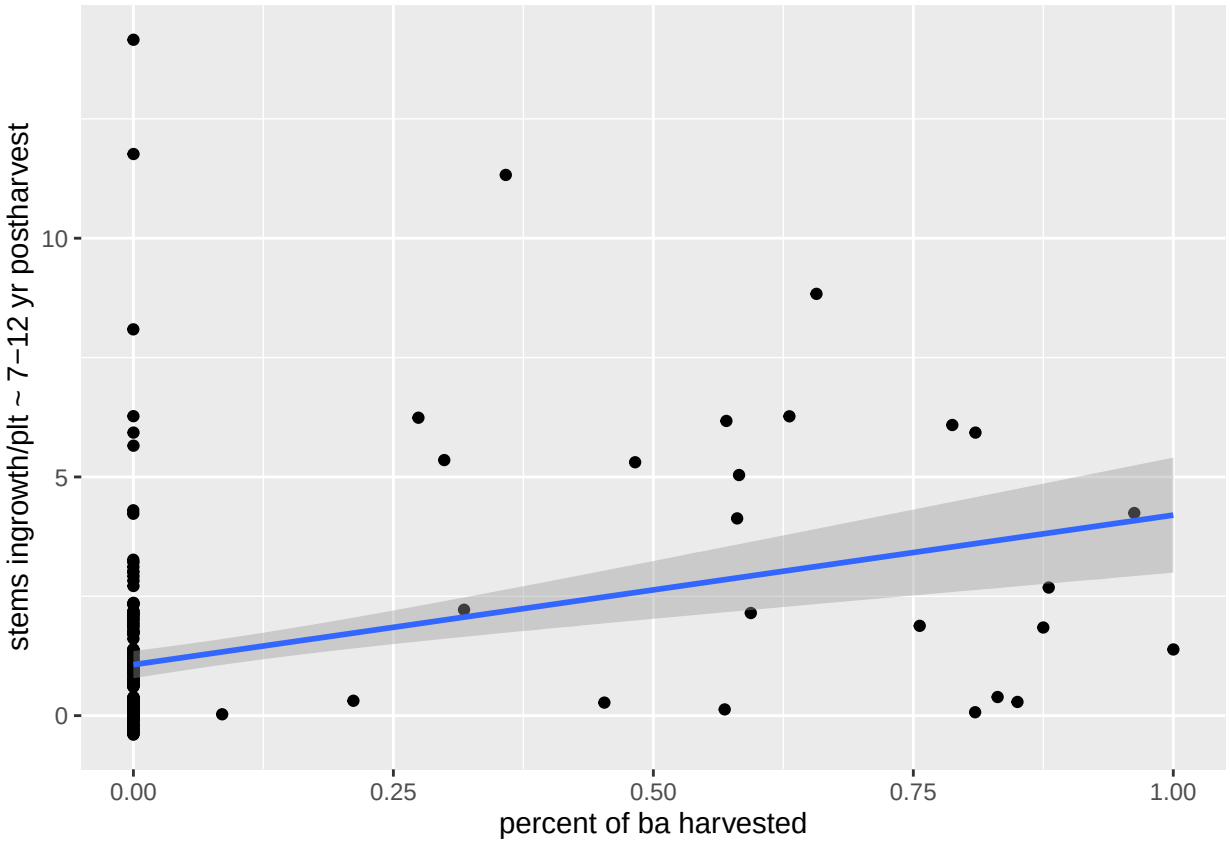
Exploring the effects of logging intensity:

```
# plot ingrowth vs. pct cut
sugar4 %>% group_by(sub) %>%
  summarize(pctcut = sum(ba_ac_1[cut_12])/
              sum(ba_ac_1[statuscd_1 == 1], na.rm = T),
            ingrowth12 = sum(ingrowth_12),
            ingrowth23 = sum(ingrowth_23), ingrowth34 = sum(ingrowth_34)) %>%
  ggplot2::ggplot(ggplot2::aes(pctcut, ingrowth23)) +
  ggplot2::geom_jitter() +
  ggplot2::geom_smooth(method = "lm") +
  scale_y_continuous("stems ingrowth/plt ~ 2-7 yr postharvest") +
  scale_x_continuous("percent of ba harvested")
```

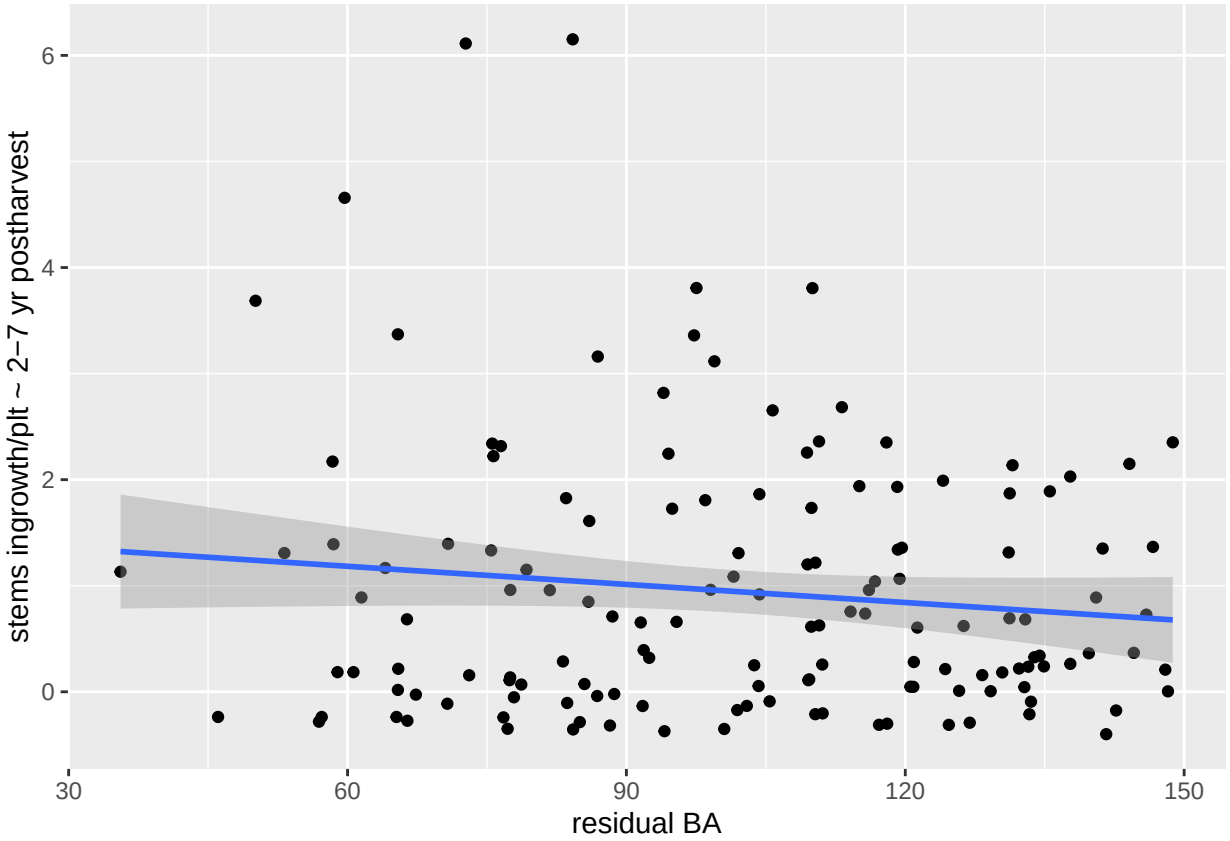



```
sugar4 %>% group_by(sub) %>%
  summarize(pctcut = sum(ba_ac_1[cut_12])/
              sum(ba_ac_1[statuscd_1 == 1], na.rm = T),
            ingrowth12 = sum(ingrowth_12),
            ingrowth23 = sum(ingrowth_23), ingrowth34 = sum(ingrowth_34)) %>%
  ggplot2::ggplot(ggplot2::aes(pctcut, ingrowth34)) +
```

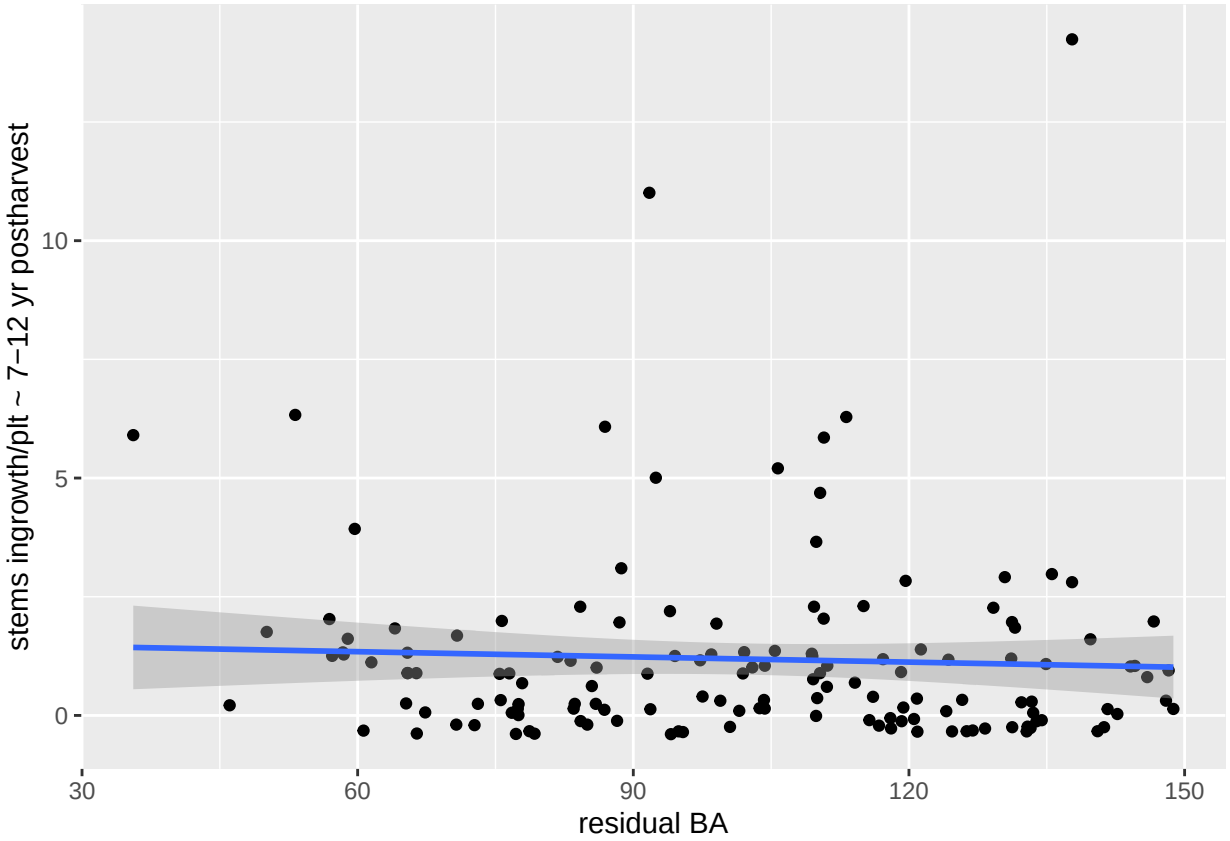
```
  ggplot2::geom_jitter() +
  ggplot2::geom_smooth(method = "lm") +
  scale_y_continuous("stems ingrowth/plt ~ 7-12 yr postharvest") +
  scale_x_continuous("percent of ba harvested")
```




```
# plot ingrowth vs. res ba (really only care about res. ba ~30 - 150)
sugar4 %>% group_by(sub) %>%
  summarize(resba = sum(ba_ac_2[statuscd_2 == 1], na.rm = T),
            ingrowth12 = sum(ingrowth_12),
            ingrowth23 = sum(ingrowth_23), ingrowth34 = sum(ingrowth_34)) %>%
  filter(resba > 30, resba < 150) %>%
  ggplot2::ggplot(ggplot2::aes(resba, ingrowth23)) +
  ggplot2::geom_jitter() +
  ggplot2::geom_smooth(method = "lm") +
  scale_y_continuous("stems ingrowth/plt ~ 2-7 yr postharvest") +
  scale_x_continuous("residual BA")
```



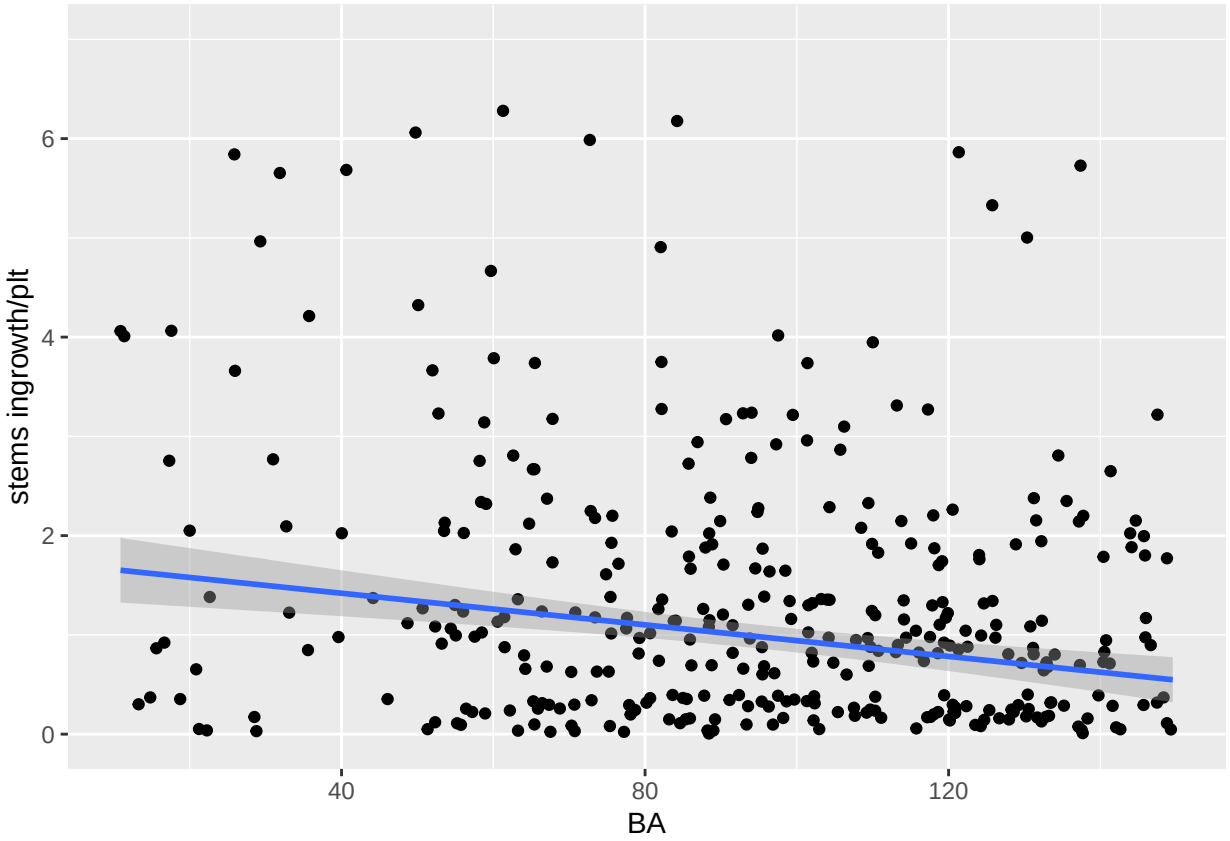

```
sugar4 %>% group_by(sub) %>%
  summarize(resba = sum(ba_ac_2[statuscd_2 == 1], na.rm = T),
            ingrowth12 = sum(ingrowth_12),
            ingrowth23 = sum(ingrowth_23), ingrowth34 = sum(ingrowth_34)) %>%
  filter(resba > 30, resba < 150) %>%
  ggplot2::ggplot(ggplot2::aes(resba, ingrowth34)) +
  ggplot2::geom_jitter() +
  ggplot2::geom_smooth(method = "lm") +
  scale_y_continuous("stems ingrowth/plt ~ 7-12 yr postharvest") +
  scale_x_continuous("residual BA")
```



Exploring the effects of basal area on ingrowth, irregardless of logging:

```
temp <- sugar4 %>% group_by(sub) %>%
  summarize(ba1 = sum(ba_ac_1[statuscd_1 == 1], na.rm = T),
            ba2 = sum(ba_ac_2[statuscd_2 == 1], na.rm = T),
            ba3 = sum(ba_ac_3[statuscd_3 == 1], na.rm = T),
            ingrowth12 = sum(ingrowth_12),
            ingrowth23 = sum(ingrowth_23), ingrowth34 = sum(ingrowth_34))

rbind(data.frame(ba = temp$ba1, ingrowth = temp$ingrowth12),
      data.frame(ba = temp$ba2, ingrowth = temp$ingrowth23),
      data.frame(ba = temp$ba3, ingrowth = temp$ingrowth34)) %>%
  filter(ba > 10, ba < 150) %>%
  ggplot2::ggplot(ggplot2::aes(ba, ingrowth)) +
  ggplot2::geom_jitter() +
  ggplot2::geom_smooth(method = "lm") +
  scale_y_continuous("stems ingrowth/plt", limits = c(0, 7)) +
  scale_x_continuous("BA")
```



It looks like ingrowth changes through time following logging, with residual basal area being a better predictor in the first timestep following logging, and percent removed being a better predictor in the second timestep. It would be nice to have different ingrowth models for each of the three timesteps following logging (assuming we set the problem up with 5 year timesteps and a 15 year cutting cycle). The data with 4 measurements can be used for the second timestep following logging (between measurements 3 and 4 if logging was between 1 and 2), but we're probably better off going back and making a bigger dataset that includes data with only 3 measurements to get a better model of ingrowth in the first timestep after logging. We could use the 5 measurement data for a model of the third timestep after logging, but it's so limited that it might be more accurate to just use a "background" model of ingrowth based on basal area (ignoring logging), using a bigger dataset.

We should also think about how to deal with different species in the ingrowth. Here's a broad overview of the species breakdown we're seeing:

```
sugar4 %>% filter(ingrowth_12 | ingrowth_23 | ingrowth_34) %>%
  group_by(spp_4) %>% summarise(n = n()) %>% filter(!is.na(spp_4)) %>%
  arrange(desc(n))
```

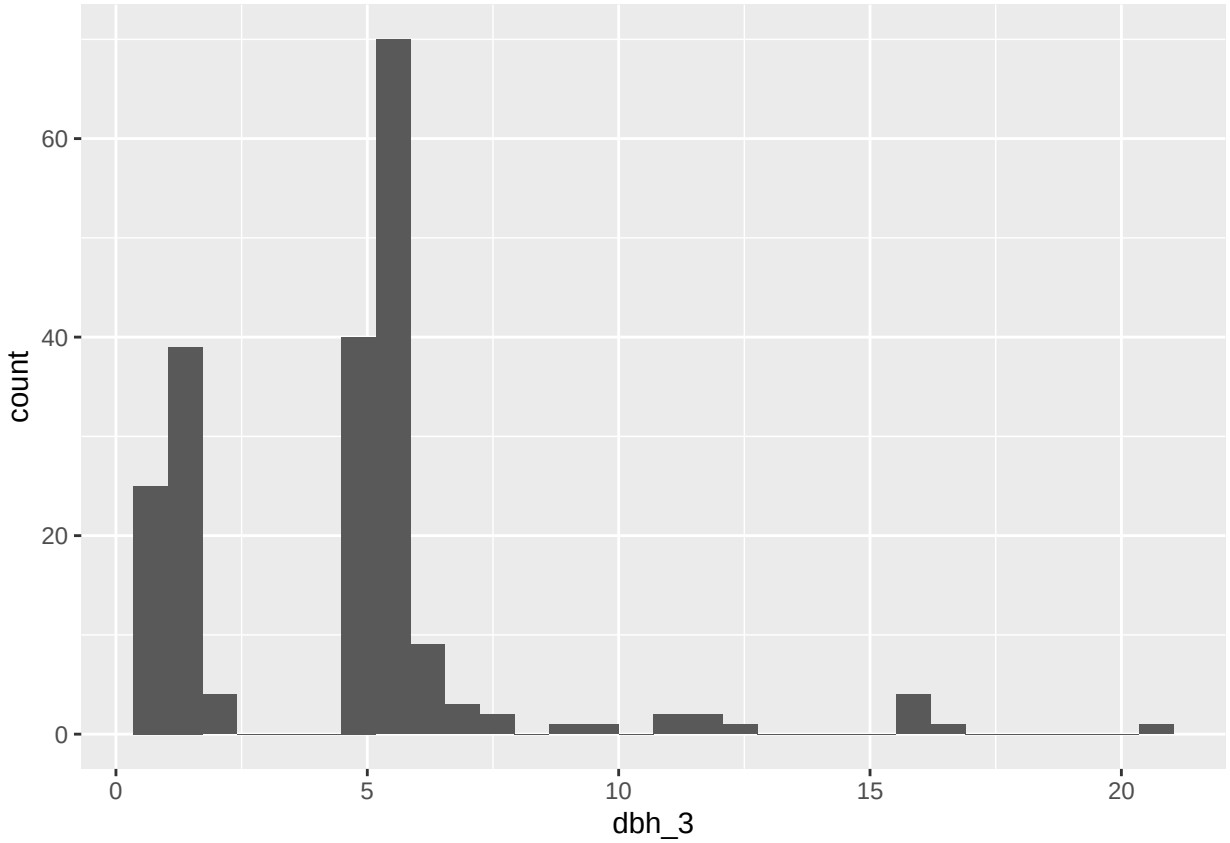
spp_4	n
beech	185
hard maple	164
yellow birch	64
striped maple	58
fir	48
aspen	24

spp_4	n
red spruce	23
red maple	20
shrubs	12
paper birch	11
hophornbeam	9
ash	7
black ash	5
pin cherry	4
hemlock	3
black cherry	2
red oak	2
white pine	2
basswood	1
gray birch	1
white spruce	1

Maybe we just remove the ingrowth that's not hard maple, assuming it will get thinned out at the next entry. That would affect our tree growth a bit, because we would be ignoring the additional stocking, but probably only minimally, since 15 year's growth of new regen doesn't add much stocking.

Here's a new problem: look at a histogram of the DBHs of ingrowth that showed up between measurements 2 and 3:

```
sugar4 %>% filter(ingrowth_23, statuscd_3 == 1) %>% select(spp_3, dbh_3) %>%
  ggplot(aes(dbh_3)) + geom_histogram()
```



There are a bunch of trees that are way too big to have just become measurable. The big pulse of trees just over 5" must be trees that were outside of the microplot (on which trees 1-5" are tallied) and just got big enough to be measured on the main subplot (which is bigger, and on which trees 5" + are tallied). I think we could remove those and deal with it by using TPA numbers based on the initial measurement plot size (so regen that joins the microplot would forevermore have higher TPAs associated with them based on that small plot size, trees that started out in subplots would keep their lower TPAs, and trees that were outside the microplots and joined the subplots when they passed 5" would be discarded). I don't understand the sawtimber-sized "ingrowth." Did the cruisers think those trees were out of the plot initially, and then the next cruisers decided they were in? They must be errors in some respect, and we can probably just remove them from the data. Also worth looking to see if the opposite problem occurs: trees that started out in the data and then suddenly disappeared without being recorded as having died or been cut.