masting2–define mast years, compute mast periods

Gleb Kozienko

2023-06-30

This document has initial versions of masting threshold functions, UPDATED versions of masting threshold functions, function to compute mast periods (intervals between mast years), as well as visual exploration of how well these functions work.

Upload useful functions

```
library(readr)
library(dplyr)
library(stringr)
library(tidyverse)
library(lubridate)
library(ggplot2)
library(gridExtra) #allows me to plot multiple graphs together
library(ggpubr) #allows me to plot multiple graphs together

library(Ckmeans.1d.dp)
library(RColorBrewer)
```

Get your data

```
MASTREE_climate_2 <- read_csv("CURI/R code/Mastree datasets/MASTREE_climate_2.csv")
```

```
MASTREE_climate_1 <- read_csv("CURI/R code/Mastree datasets/MASTREE_climate_1.csv")
```

```
MASTREE<-read_csv("CURI/R code/Mastree datasets/MASTREE+_selection_2023-06-01.csv")
```

Get some more data

```
Picea_species <- MASTREE_climate_1|>
  filter(Alpha_Number=="0234")
```

The provided R code performs data manipulation and filtering on the "mastree_continuous" dataset to create subsets for different tree species. The code then groups and summarizes the data based on the species, alpha number, and length, and prints the top 50 results for each combination. Finally, the code creates separate datasets for each tree species (Fagus, Quercus, Fraxinus, Picea, and Pinus) using the filtered data

```
# Filter "mastree_continuous" dataset to get only continuous variables with Unit not equal to "index"
mastree_continuous <- MASTREE_climate_2 |>
  filter(VarType == "C", Unit != "index")

# Get unique species from the "mastree_continuous" dataset
all_species <- mastree_continuous$Species |> unique()

# Group and summarize the "mastree_continuous" data based on Species, Alpha_Number, and Length,
# and print the top 50 results sorted by Length in descending order
mastree_continuous |>
```

```
  group_by(Species, Alpha_Number, Length) |>
  summarise(n = n()) |>
  arrange(desc(Length)) |>
  print(n = 50)

# Filter unique species to get those containing "Fagus" in their name
fagus <- all_species[str_detect(all_species, "Fagus")]

# Filter unique species to get those containing "Quercus" in their name
quercus <- all_species[str_detect(all_species, "Quercus")]

# Filter unique species to get those containing "Fraxinus" in their name
fraxinus <- all_species[str_detect(all_species, "Fraxinus")]

# Filter unique species to get those containing "Picea" in their name
picea <- all_species[str_detect(all_species, "Picea")]

# Filter unique species to get those containing "Pinus" in their name
pinus <- all_species[str_detect(all_species, "Pinus")]

# Create subsets for each tree species based on the filtered unique species
fagus_data <- mastree_continuous |>
  filter(Species %in% fagus)

quercus_data <- mastree_continuous |>
  filter(Species %in% quercus)

fraxinus_data <- mastree_continuous |>
  filter(Species %in% fraxinus)

picea_data <- mastree_continuous |>
  filter(Species %in% picea)

pinus_data <- mastree_continuous|>
  filter(Species %in% pinus)



# Group and summarize the "fagus_data" dataset based on Coords, Alpha_Number, and Species,
# calculate the count (n) for each group, and print the top 25 results sorted by count in descending or
fagus_data |>
  group_by(Coords, Alpha_Number, Species) |>
  summarise(n = n()) |>
  arrange(desc(n)) |>
  print(n = 25)

 #   Coords     Alpha_Number Species             n
 #   <chr>      <chr>        <chr>           <int>
 # 1 49, 11.4   3024         Fagus sylvatica   912
 # 2 49.8, 22.2 6013         Fagus sylvatica   828
 # 3 49.9, 20.5 6013         Fagus sylvatica   828
 # 4 50.4, 18.6 6013         Fagus sylvatica   828
 # 5 51, 20.9   6013         Fagus sylvatica   828
```

```
quercus_data|>
  group_by(Coords, Alpha_Number, Species)|>
  summarise(n=n())|>
  arrange(desc(n))|>
  base::print(n=75)

#    Coords       Alpha_Number Species                n
#    <chr>        <chr>        <chr>              <int>
#  1 27.2, -81.3  5001         Quercus chapmanii    972
#  2 27.2, -81.3  5001         Quercus geminata     972
#  3 27.2, -81.3  5001         Quercus myrtifolia   936
# 21 27.2, -81.3  5001         Quercus laevis       648
# 22 36.4, -121.6 5092         Quercus agrifolia    492
# 23 36.4, -121.6 5092         Quercus chrysolepis  492

fraxinus_data|>
  group_by(Coords, Alpha_Number, Species)|>
  summarise(n=n())|>
  arrange(desc(n))|>
  print(n=25)

#   Coords      Alpha_Number Species                n
#   <chr>       <chr>        <chr>              <int>
# 1 42.4, 128.1 2284         Fraxinus mandshuri...    96
# 2 43.9, -71.8 0378         Fraxinus americana       84

picea_data|>
  group_by(Coords, Alpha_Number, Species)|>
  summarise(n=n())|>
  arrange(desc(n))|>
  print(n=25)

 #   Coords       Alpha_Number Species              n
 #   <chr>        <chr>        <chr>            <int>
 # 1 39.9, -105.9 0234         Picea engelmannii  5904
 # 2 64.7, -148.3 5071         Picea glauca       1596
 # 3 49.8, 22.2   6013         Picea abies         828


pinus_data|>
  group_by(Coords, Alpha_Number, Species)|>
  summarise(n=n())|>
  arrange(desc(n))|>
  print(n=25)

 #   Coords       Alpha_Number Species              n
 #   <chr>        <chr>        <chr>            <int>
#  2 50.1, 17.2   2552         Pinus mugo        2376
#  3 49.8, 22.2   6013         Pinus sylvestris   828
#  4 49.9, 20.5   6013         Pinus sylvestris   828
#  21 61.8, 29.3   5164         Pinus sylvestris   312
# 22 39.5, -121.2 2531         Pinus ponderosa    288
```

```
# 23 61.8, 29.3   5164         Pinus cembra       168
```

The provided R code creates different subsets of data for specific tree species and locations from the previously filtered datasets. Each subset corresponds to a specific species and location identified by "Coords" and "Alpha_Number" values.

```
# Create a subset for Fagus sylvatica at Coords "49, 11.4" and Alpha_Number "3024" from fagus_data
fagus_sylvatica_at_49_11.4_from_3024 <- fagus_data |>
  filter(Coords == "49, 11.4", Alpha_Number == "3024")

# Create a subset for Fagus sylvatica at Coords "49.8, 22.2" and Alpha_Number "6013" from fagus_data
fagus_sylvatica_at_49.8_22.2_from_6013 <- fagus_data |>
  filter(Coords == "49.8, 22.2", Alpha_Number == "6013")

# Create a subset for Quercus chapmanii at Coords "27.2, -81.3", Alpha_Number "5001" from quercus_data
quercus_chapmanii_at_27.2_minus81.3_from_5001 <- quercus_data |>
  filter(Coords == "27.2, -81.3", Alpha_Number == "5001", Species == "Quercus chapmanii")

# Create a subset for Quercus geminata at Coords "27.2, -81.3", Alpha_Number "5001" from quercus_data
quercus_geminata_at_27.2_minus81.3_from_5001 <- quercus_data |>
  filter(Coords == "27.2, -81.3", Alpha_Number == "5001", Species == "Quercus geminata")

# Create a subset for Picea engelmannii at Coords "39.9, -105.9", Alpha_Number "0234" from picea_data
picea_engelmannii_at_39.9_minus105.9_from_0234 <- picea_data |>
  filter(Coords == "39.9, -105.9", Alpha_Number == "0234")

# Create a subset for Picea glauca at Coords "64.7, -148.3", Alpha_Number "5071" from picea_data
picea_glauca_at_64.7_minus148.3_from_5071 <- picea_data |>
  filter(Coords == "64.7, -148.3", Alpha_Number == "5071")

# Create a subset for Picea abies at Coords "49.8, 22.2", Alpha_Number "6013" from picea_data
picea_abies_at_49.8_22.2_from_6013 <- picea_data |>
  filter(Coords == "49.8, 22.2", Alpha_Number == "6013")

# Create a subset for Pinus mugo at Coords "50.1, 17.2", Alpha_Number "2552" from pinus_data
pinus_mugo_at_50.1_17.2_from_2552 <- pinus_data |>
  filter(Coords == "50.1, 17.2", Alpha_Number == "2552")

# Create a subset for Pinus sylvestris at Coords "49.8, 22.2", Alpha_Number "6013" from pinus_data
pinus_sylvestris_at_49.8_22.2_from_6013 <- pinus_data |>
  filter(Coords == "49.8, 22.2", Alpha_Number == "6013")
```

Threshold functions.

As a part of our research, we encountered the need to define years as mast years (those years when lots of seeds are being produced) and non-mast years (years between mast years, when small amount of seeds is being produced)

Note: codes for ADM method and MSD method here are almost identical to the codes from "masting3 – initial versions of masting threshold and visualizations – final.Rmd", with one exception– now you have to supply supply datasets that have all 12 monthly rows for each year

Also, I add changes to ADCY method so it can process multiple sites. Also, I am implementing code for k-medians clustering that I introduced in "masting3.5–k-means vs k-medians – final.Rmd"

1) Average distance to mean:

Average distance to the mean, aka mean deviation: https://www.mathsisfun.com/data/mean-deviation.html#:~:text=1.,each%20value%2C%20ignore%20minus%20signs)

Function computes mean deviation, and sets threshold to the value of mean plus mean deviation threshold=mean+(mean deviation)

```
# ADM_vec is a function that returns ADM threshold value

  # ADM_vec calculates the threshold  of seed production values.
  ADM_vec <- function(seed_prod_vals) {
    mean <- mean(seed_prod_vals)        # Calculate the mean of seed production values.
    abs_val_diff <- abs(seed_prod_vals - mean)    # Calculate the absolute distances from the mean.
    sum <- sum(abs_val_diff)                       # Calculate the sum of absolute distances.
    n <- length(seed_prod_vals)                   # Get the number of seed production values.
    result <- mean + sum / n                      #compute threshold
    return(result)                                # Return threshold
  }
```

ADM_funct is a function that classifies years as “mast year” or “non-mast year” based on a threshold value, then adds column of mast vs. nonmast values into the dataset

```
# ADM_funct is a function that classifies years as "mast year" or "non-mast year" based on a threshold


#according to profs. suggestion, this function uses same threshold computed across all sites rather tha
#If you need one site only, first supply all sites into function and then filter the outcome


# ADM_funct is a function that classifies years as "mast year" or "non-mast year" based on a threshold

# Arguments:
# Dataset: A data frame containing a column "Value" representing seed production values.

ADM_funct <- function(Dataset) {

  # Filter the dataset to retain only rows with month equal to 1 (same_month_only_data).
  same_month_only_data <- Dataset |>
    filter(month == 1)

  # Extract the unique values of "Value" from the filtered data.
  vals_unique <- same_month_only_data$Value

  # Calculate the threshold value for classification using the ADM_vec function on the unique values.
  treshold <- ADM_vec(vals_unique)

  # Create a logical vector (mast_boolean) indicating whether each value in the original dataset is abo
  mast_boolean <- Dataset$Value > treshold

  # Initialize a vector (mast_val) to store the classification results.
  mast_val <- rep(NA, length(mast_boolean))

  # Loop through the mast_boolean vector and assign "mast year" or "non-mast year" based on the logical
  for (i in 1:length(mast_boolean)) {
    if (mast_boolean[i] == TRUE) {
      mast_val[i] <- "mast year"
    } else {
```

```
      mast_val[i] <- "non-mast year"
    }
  }

  # Add the "mast_val" vector as a new column to the original dataset.
  Dataset$mast_val <- mast_val

  # Return the modified dataset with the new "mast_val" column.
  return(Dataset)
}
```

2.1) Average distance in consecutive years

With this approach, we define mast years as years that shown increase in seed production from the previous year that is greater than average positive incerease in seed production.

Or, described in steps: 1) Calculate differences in seed production in consecutive years (both positive and negative differences) 2) Ignoring negative differences completely (we won't need them), calculte the mean positive difference. obtained value will be a mean increase in seed production (not a mean change in seed production, because we are ignoring negarive differences, but exactly the mean increase) 3) whatever year has an increase from previous year higher than mean increase, is a mast year.

```
#apply only for a dataset with one site only (if used on dataset with multiple sites, be super cautious


# ADCY_simple is a function that classifies years as "mast year" or "non-mast year" based on the positi

# Arguments:
# Dataset: A data frame containing a column "Value"  representing seed production values.

ADCY_simple <- function(Dataset) {

  vals <- Dataset$Value           # Extract the "Value" column from the dataset.
  diffs <- rep(NA, length(vals))  # Initialize a vector to store differences between consecutive values

  # Calculate the differences between consecutive values and store them in the 'diffs' vector.
  for (i in 2:length(vals)) {
    diffs[i] <- vals[i] - vals[i - 1]
  }

  diffs_positive <- diffs[which(diffs > 0)]    # Filter out positive differences.

  sum_diffs_positive <- sum(diffs_positive)    # Calculate the sum of positive differences.
  num_diffs_positive <- length(diffs_positive) # Get the number of positive differences.

  sig_diff <- sum_diffs_positive / num_diffs_positive  # Calculate the average of positive differences.

  mast_boolean <- diffs > sig_diff         # Create a boolean vector indicating if the difference is gr

  mast_val <- rep(NA, length(mast_boolean))  # Initialize a vector to store the classification of each

  # Classify years based on whether the difference is greater than the average positive difference.
  for (i in 2:length(mast_boolean)) {
    if (mast_boolean[i] == TRUE) {
      mast_val[i] <- "mast year"           # If the difference is greater, classify as "mast year".
```

```
    } else {
      mast_val[i] <- "non-mast year"       # If the difference is not greater, classify as "non-mast ye
    }
  }

  Dataset$mast_val <- mast_val   # Add the classification column "mast_val" to the original dataset.

  return(Dataset)   # Return the dataset with the "mast_val" column added.

}
```

2.2) Average distance in consecutive years+remove in-row mast years

This approach is the same as the previous approach, but, after all the steps from the previous approach, we also eliminate consecutive mast years. I.e., if we have three mast years in a row as defined by mean increase, which could mean that we had a significant increase in seed production for three years in a row, function will label first two years as non-mast years, and keep only the last year, that has highest seed production, as a mast year

Or, described in steps: 1) Calculate differences in seed production in consecutive years (both positive and negative differences) 2) Ignoring negative differences completely (we won't need them), calculte the mean positive difference. obtained value will be a mean increase in seed production (not a mean change in seed production, because we are ignoring negarive differences, but exactly the mean increase) 3) whatever year has an increase from previous year higher than mean increase, is a mast year. 4) if you have more than one mast year in the row, label each year in this row except for last one as a non-mast year

```
#apply only for a dataset with one site only


#simple ADCY_NMIR -- does ADCY, then makes sure there are No Mast years In a Row (in a row of mast years

# Arguments:
# Dataset: A data frame containing a column "Value"  representing seed production values.

ADCY_NMIR <- function(Dataset) {

  vals <- Dataset$Value           # Extract the "Value" column from the dataset.
  diffs <- rep(NA, length(vals))  # Initialize a vector to store differences between consecutive values

  # Calculate the differences between consecutive values and store them in the 'diffs' vector.
  for (i in 2:length(vals)) {
    diffs[i] <- vals[i] - vals[i - 1]
  }

  diffs_positive <- diffs[which(diffs > 0)]    # Filter out positive differences.

  sum_diffs_positive <- sum(diffs_positive)    # Calculate the sum of positive differences.
  num_diffs_positive <- length(diffs_positive) # Get the number of positive differences.

  sig_diff <- sum_diffs_positive / num_diffs_positive  # Calculate the average of positive differences.

  mast_boolean <- diffs > sig_diff         # Create a boolean vector indicating if the difference is gr

  # This step ensures that there is no consecutive mast years
  for (i in 2:(length(mast_boolean) - 1)) {
```

```
    if (mast_boolean[i + 1] == TRUE) {
      mast_boolean[i] <- FALSE   # If the next year is classified as "mast year," set the current year
    } else {
      next   # Otherwise, skip to the next iteration.
    }
  }

  mast_val <- rep(NA, length(mast_boolean))   # Initialize a vector to store the classification of each

  # Classify years based on the mast_boolean vector.
  for (i in 2:length(mast_boolean)) {
    if (mast_boolean[i] == TRUE) {
      mast_val[i] <- "mast year"           # If the year is classified as "mast year," assign "mast yea
    } else {
      mast_val[i] <- "non-mast year"       # If the year is classified as "non-mast year," assign "non-
    }
  }

  Dataset$mast_val <- mast_val   # Add the classification column "mast_val" to the original dataset.

  return(Dataset)   # Return the dataset with the "mast_val" column added.

}
```

2.3) ADCY_simple and ADCY_NMIR are set up in a way that they can only be applied to the dataset with one site only. This is problematic as other function (ADM, MSD, KMDN) use thresholds calculated across all sites.

Thus, i am modifying my ADCY functions so they can now work across all sites

Next four chunks are supposed to work together: pre_ADCY_1, ADCY_simple2, and ADCY_NMIR2 are all used in ADCY_all_sites function

ADCY_all_sites function does the same thing as ADCY_simple, and ADCY_NMIR, but for all sites in the study. You can choose to keep consecutive mast years by setting ADCY_type argument to "simple", or you can remove consecutive mast years using "NMIR"

pre_ADCY_1: This function calculates the differences between consecutive masting values at one particular site and returns positive differences.

```
pre_ADCY_1 <- function(Dataset) {

 # Step 1: Filter the input Dataset to keep only yearly values(i achieve it by filtering for  January (
  same_month_only_data <- Dataset |>
    filter(month == 1)

  # Step 2: Extract the yearly values
  vals <- same_month_only_data$Value

  # Step 3: Initialize a vector of NA values to store the differences between consecutive values
  diffs <- rep(NA, length(vals))

  # Step 4: Calculate the differences between consecutive values using a for loop
  for (i in 2:length(vals)) {
    diffs[i] <- vals[i] - vals[i - 1]
  }
```

```
  # Step 5: Select only the positive differences using the which function
  diffs_positive <- diffs[which(diffs > 0)]

  # Step 6: Return the vector containing positive differences of yearly values
  return(diffs_positive)

}
```

ADCY_simple2: This function applies the ADCY method to a dataset with a single site (assuming one species and variable). It calculates the differences between consecutive values, identifies mast years based on a threshold (sig_diff), and assigns "mast year" or "non-mast year" labels accordingly.

```
#apply only for a dataset with one site only


ADCY_simple2 <- function(Dataset, sig_diff) {

  # Step 1: Filter the input Dataset to keep only yearly values(i achieve it by filtering for  January
  same_month_only_data <- Dataset |>
    filter(month == 1)

  # Step 2: Extract the yearly values
  vals <- same_month_only_data$Value

  # Step 3: Initialize a vector of NA values to store the differences between consecutive values
  diffs <- rep(NA, length(vals))

  # Step 4: Calculate the differences between consecutive values using a for loop
  for (i in 2:length(vals)) {
    diffs[i] <- vals[i] - vals[i - 1]
  }

  # Step 5: Create a boolean vector to indicate whether the differences are greater than the threshold
  mast_boolean <- diffs > sig_diff

  # Step 6: Initialize a vector to store the mast year labels (mast year or non-mast year)
  mast_val <- rep(NA, length(mast_boolean))

  # Step 7: Assign "mast year" or "non-mast year" labels based on the mast_boolean values
  for (i in 2:length(mast_boolean)) {
    if (mast_boolean[i] == TRUE) {
      mast_val[i] <- "mast year"
    } else {
      mast_val[i] <- "non-mast year"
    }
  }

  # Step 8: Add the mast_val column to the same_month_only_data data frame
  same_month_only_data$mast_val <- mast_val

  # Step 9: Select only the Year and mast_val columns from same_month_only_data
  MastVal_year <- same_month_only_data |>
    select(Year, mast_val)
```

```
  # Step 10: Merge the mast_val information back to the original Dataset based on the Year column
  Dataset_return <- Dataset |>
    left_join(MastVal_year, by = "Year", relationship = "many-to-many")

  # Step 11: Return the Dataset with the mast_val information added
  return(Dataset_return)

}
```

ADCY_NMIR2: This function is similar to ADCY_simple2 but additionally ensures that there are no consecutive mast years. It identifies mast years using the ADCY method, and then if there are consecutive mast years, it changes the label to "non-mast year" for all of them but the last one.

```
#apply only for a dataset with one site only


#simple ADCY_NMIR -- does ADCY, makes sure there are No Mast years In a Row
ADCY_NMIR2 <- function(Dataset, sig_diff) {

  # Step 1: Filter the input Dataset to keep only yearly values(i achieve it by filtering for  January
  same_month_only_data <- Dataset |>
    filter(month == 1)

  # Step 2: Extract the yearly values
  vals <- same_month_only_data$Value

  # Step 3: Initialize a vector of NA values to store the differences between consecutive values
  diffs <- rep(NA, length(vals))

  # Step 4: Calculate the differences between consecutive values using a for loop
  for (i in 2:length(vals)) {
    diffs[i] <- vals[i] - vals[i - 1]
  }

  # Step 5: Create a boolean vector to indicate whether the differences are greater than the threshold
  mast_boolean <- diffs > sig_diff

  # Step 6: Remove in-row mast years
  for (i in 2:length(mast_boolean) - 1) {
    if (mast_boolean[i + 1] == TRUE) {
      mast_boolean[i] <- FALSE
    } else {
      next
    }
  }

  # Step 7: Initialize a vector to store the mast year labels (mast year or non-mast year)
  mast_val <- rep(NA, length(mast_boolean))

  # Step 8: Assign "mast year" or "non-mast year" labels based on the mast_boolean values
  for (i in 2:length(mast_boolean)) {
    if (mast_boolean[i] == TRUE) {
      mast_val[i] <- "mast year"
    } else {
```

```
      mast_val[i] <- "non-mast year"
    }
  }

  # Step 9: Add the mast_val column to the same_month_only_data data frame
  same_month_only_data$mast_val <- mast_val

  # Step 10: Select only the Year and mast_val columns from same_month_only_data
  MastVal_year <- same_month_only_data |>
    select(Year, mast_val)

  # Step 11: Merge the mast_val information back to the original Dataset based on the Year column
  Dataset_return <- Dataset |>
    left_join(MastVal_year, by = "Year", relationship = "many-to-many")

  # Step 12: Return the Dataset with the mast_val information added
  return(Dataset_return)

}
```

ADCY_all_sites function does the same thing as ADCY_simple, and ADCY_NMIR, but for all sites in the study. You can choose to keep consecutive mast years by setting ADCY_type argument to "simple", or "NMIR"

ADCY_all_sites: This function applies either ADCY_simple2 or ADCY_NMIR2 method for all sites in the study (assuming one species and variable with the same Alpha_Number). It calculates the threshold (sig_diff) based on all positive differences from all sites and then applies the specified ADCY method.

```
#function assumes single species from particular study/coordinate
#i.e. supply dataset with only one species, variable, but most importantly, SAME Alpha_Number(it means

# Function to identify mast years using the ADCY (Average Distance Consecutive Years) method for all si
# Input: Dataset - a data frame containing the relevant data
#        ADCY_type - a character string specifying the type of ADCY method ("simple" or "NMIR")
ADCY_all_sites <- function(Dataset, ADCY_type) {

  # Step 1: Check if the ADCY_type parameter is missing; if missing, stop the function with an error me
  if (missing(ADCY_type))
    stop(
      'Second parameter is missing. Use only \"simple\" or \"NMIR\" as values of the second parameter.'
    )

  # Step 2: Check if the ADCY_type is "simple"
  if (ADCY_type == "simple") {
    # Step 3: Get the unique site numbers from the Dataset
    sites <- Dataset$Site_number |> unique()

    # Step 4: Initialize a vector to store all positive differences between consecutive values from all
    all_diffs <- c()

    # Step 5: Loop through each site to calculate the positive differences between consecutive values
    for (i in 1:length(sites)) {
      current_site <- sites[i]

      # Step 6: Filter the data for the current site
```

```
    data_one_site <- Dataset |>
      filter(Site_number == current_site)

    # Step 7: Calculate the positive differences for the current site and store them in diff_one_site
    diff_one_site <- pre_ADCY_1(data_one_site)

    # Step 8: Add the positive differences from the current site to the all_diffs vector
    all_diffs <- c(all_diffs, diff_one_site)

  }

  # Step 9: Calculate the sum of all positive differences and the total number of positive difference
  sum_diffs_positive <- sum(all_diffs)
  num_diffs_positive <- length(all_diffs)

  # Step 10: Calculate the threshold (sig_diff) as the average of all positive differences
  sig_diff <- sum_diffs_positive / num_diffs_positive

  # Step 11: Initialize a data frame (Dataset_return) to store the results for all sites
  Dataset_return <- NULL

  # Step 12: Loop through each site again to identify mast years using the ADCY_simple2 method
  for (i in 1:length(sites)) {
    current_site <- sites[i]

    # Step 13: Filter the data for the current site
    data_one_site <- Dataset |>
      filter(Site_number == current_site)

    # Step 14: Identify mast years using the ADCY_simple2 method and store the results in pre_Dataset
    pre_Dataset_return <- ADCY_simple2(data_one_site, sig_diff)

    # Step 15: Add the results for the current site to the Dataset_return data frame
    Dataset_return <- rbind(Dataset_return, pre_Dataset_return)
  }

  #if method is NMIR:
} else if (ADCY_type == "NMIR") {
  # Step 16: Get the unique site numbers from the Dataset
  sites <- Dataset$Site_number |> unique()

  # Step 17: Initialize a vector to store all positive differences between consecutive values from al
  all_diffs <- c()

  # Step 18: Loop through each site to calculate the positive differences between consecutive values
  for (i in 1:length(sites)) {
    current_site <- sites[i]

    # Step 19: Filter the data for the current site
    data_one_site <- Dataset |>
      filter(Site_number == current_site)

    # Step 20: Calculate the positive differences for the current site and store them in diff_one_sit
```

```
      diff_one_site <- pre_ADCY_1(data_one_site)

      # Step 21: Add the positive differences from the current site to the all_diffs vector
      all_diffs <- c(all_diffs, diff_one_site)

    }

    # Step 22: Calculate the sum of all positive differences and the total number of positive differenc
    sum_diffs_positive <- sum(all_diffs)
    num_diffs_positive <- length(all_diffs)

    # Step 23: Calculate the threshold (sig_diff) as the average of all positive differences
    sig_diff <- sum_diffs_positive / num_diffs_positive

    # Step 24: Initialize a data frame (Dataset_return) to store the results for all sites
    Dataset_return <- NULL

    # Step 25: Loop through each site again to identify mast years using the ADCY_NMIR2 method
    for (i in 1:length(sites)) {
      current_site <- sites[i]

      # Step 26: Filter the data for the current site
      data_one_site <- Dataset |>
        filter(Site_number == current_site)

      # Step 27: Identify mast years using the ADCY_NMIR2 method and store the results in pre_Dataset_r
      pre_Dataset_return <- ADCY_NMIR2(data_one_site, sig_diff)

      # Step 28: Add the results for the current site to the Dataset_return data frame
      Dataset_return <- rbind(Dataset_return, pre_Dataset_return)

    }

  } else {
    # Step 29: If the ADCY_type is neither "simple" nor "NMIR", return an error message
    return(cat("Use only \"simple\" or \"NMIR\" as a parameter for ADCY_type"))
  }

  # Step 30: Return the final data frame (Dataset_return) containing the mast year information for all
  return(Dataset_return)

}
```

3) Mean standard deviates While two previous methods were designed and implemented by me, this method comes from the existing paper on masting:

LaMontagne, Jalene M., and Stan Boutin. "Quantitative methods for defining mast-seeding years across species and studies." Journal of Vegetation Science 20.4 (2009): 745-753.

Link:https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1654-1103.2009.01068.x

Look for the section in the paper titled "Six Methods for Identifying Mast Years" In that section, look for "(4) Years when seed production is greater than the mean by a designated amount based on standardized deviates" There, you will find the description of the method implemented here.

3.1) funtion that simply returns numeric threshold

```r
#apply only for a dataset with one month only


# MSD_vec is a function that calculates and returns the threshold value based on the Mean Standardized

# Arguments:
# vals: A numeric vector containing the values for which the threshold is to be calculated.

MSD_vec <- function(vals) {

  mean_seed_prod = mean(vals)          # Calculate the mean of the input values.
  SD_seed_prod = sd(vals)              # Calculate the standard deviation of the input values.

  # Calculate annual deviates by standardizing values using mean and standard deviation.
  annual_deviates <- (vals - mean_seed_prod) / SD_seed_prod

  treshold <- abs(min(annual_deviates))   # Calculate the threshold as the absolute value of the minimu

  # Adjust the threshold by scaling it back to the original data scale.
  treshold_adj <- treshold * SD_seed_prod + mean_seed_prod

  return(treshold_adj)   # Return the adjusted threshold value.

}
```

3.2) function that adds mast vs. non-mast column to the dataset

```r
#according to profs. suggestion, this function uses same threshold computed across all sites rather tha
#If you need one site only, first supply all sites into function and then filter the outcome

#apply only for a dataset with one month only


# MSD_orig is a function that classifies years as "mast year" or "non-mast year" based on the Mean Stan

# Arguments:
# Dataset: A data frame containing a column "Value" representing seed production.
# Define a function called "MSD_orig" that takes a dataset (Dataset) as input.

MSD_orig <- function(Dataset) {

  # Filter the dataset to retain only rows with month equal to 1 (same_month_only_data).
  same_month_only_data <- Dataset |>
    filter(month == 1)

  # Extract the "Value" data from the original dataset.
  vals <- Dataset$Value

  # Extract the unique "Value" data from the filtered data (same_month_only_data).
  vals_unique <- same_month_only_data$Value

  # Calculate the mean and standard deviation of the "Value" data for the same month (mean_seed_prod an
  mean_seed_prod = mean(vals_unique)
  SD_seed_prod = sd(vals_unique)
```

```
  # Calculate the annual deviations using the mean and standard deviation.
  annual_deviates <- (vals - mean_seed_prod) / SD_seed_prod

  # Calculate the threshold for classification as the absolute minimum of the annual deviations.
  treshold <- abs(min(annual_deviates))

  # Create a logical vector (mast_boolean) indicating whether each value's annual deviation is above th
  mast_boolean <- annual_deviates > treshold

  # Initialize a vector (mast_val) to store the classification results.
  mast_val <- rep(NA, length(mast_boolean))

  # Loop through the mast_boolean vector and assign "mast year" or "non-mast year" based on the logical
  for (i in 1:length(mast_boolean)) {
    if (mast_boolean[i] == TRUE) {
      mast_val[i] <- "mast year"
    } else {
      mast_val[i] <- "non-mast year"
    }
  }

  # Add the "mast_val" vector as a new column to the original dataset.
  Dataset$mast_val <- mast_val

  # Return the modified dataset with the new "mast_val" column.
  return(Dataset)
}
```

4) k-medians clustering for two medians https://en.wikipedia.org/wiki/K-medians_clustering https://machinelearningjourney.com/index.php/2020/02/07/k-means-k-medians/

```
# Define a function called "KMDN" that takes a dataset (Dataset) as input.
KMDN <- function(Dataset) {

  # Filter the dataset to retain only rows with month equal to 1 (same_month_only_data).
  same_month_only_data <- Dataset |>
    filter(month == 1)

  # Extract the "Value" data from the filtered data (same_month_only_data).
  vals <- same_month_only_data$Value

  # Perform k-medians clustering analysis on the "Value" data with 2 clusters.
  cluster_analysis <- Ckmedian.1d.dp(vals, 2)

  # Extract the cluster assignments from the clustering results.
  cluster_vals <- cluster_analysis$cluster

  # Initialize a vector (mast_val) to store the classification results.
  mast_val <- rep(NA, length(cluster_vals))

  # Loop through the cluster_vals vector and assign "mast year" or "non-mast year" based on the cluster
  for (i in 1:length(mast_val)) {
    if (cluster_vals[i] == 2) {
      mast_val[i] <- "mast year"
```

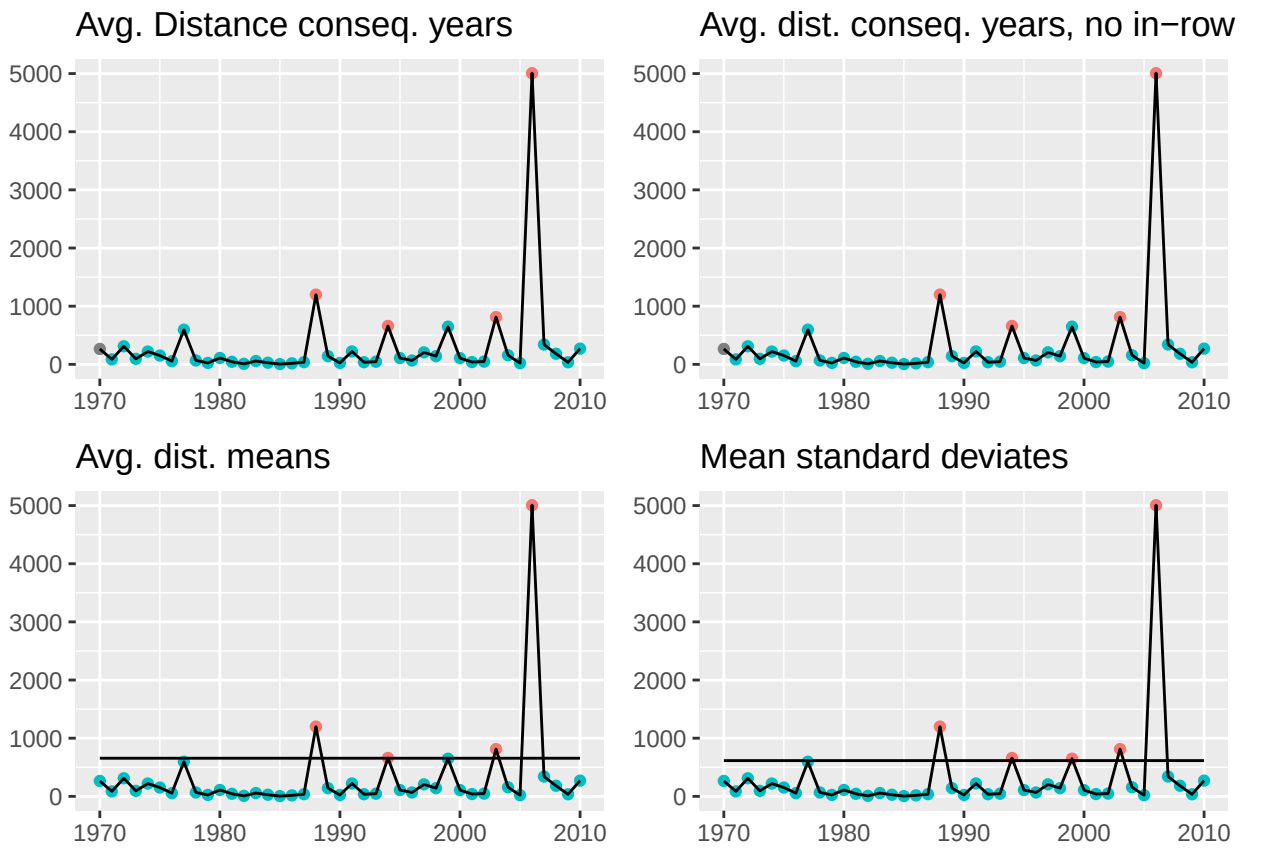
```
    } else {
      mast_val[i] <- "non-mast year"
    }
  }

  # Add the "mast_val" vector as a new column to the filtered dataset (same_month_only_data).
  same_month_only_data$mast_val <- mast_val

  # Select only the "Site_number," "Year," and "mast_val" columns from the filtered data (same_month_on
  MastVal_year <- same_month_only_data |>
    select(Site_number, Year, mast_val)

  # Merge the "mast_val" column back to the original dataset based on "Site_number" and "Year" columns.
  Dataset_return <- Dataset |>
    left_join(MastVal_year,
              by = c("Site_number", "Year"),
              relationship = "many-to-many")

  # Return the modified dataset (Dataset_return) with the "mast_val" column added.
  return(Dataset_return)
}
```

EXPLORATION OF MAST THRESHOLD FUNCTIONS Here you can see and compare how different threshold methods identify years as mast vs. non-mast

The provided R code visually explores and compares different threshold methods for identifying mast years versus non-mast years in the "Dataset1," which is related to Picea species and filtered for the month of January and Site_number "002". The code calculates the mast years using different methods, including ADM (Average Distance to Mean), ADCY (Average Distance to Consecutive Years), ADCY_NMIR (Average Distance to Consecutive Years with No In-Row Mast Years), and MSD (Mean standard deviates). The resulting mast years are then plotted against the temperature data for comparison. 1)

```
# Filter the Picea_species dataset "Dataset1" to consider only data for January and Site_number "002"
Dataset1 <- Picea_species |> filter(month == 1, Site_number == "004")

# Create a plot to compare mast identification using ADM method
ADM_plot <- ggplot(ADM_funct(Dataset1)) +
  geom_point(aes(x = Year, y = Value, color = mast_val)) +
  geom_line(aes(x = Year, y = Value)) +
  geom_line(aes(x = Year, y = ADM_vec(Value)))+
  ggtitle("Avg. dist. means") +
  theme(legend.position = "none",
        axis.title.x = element_blank(), # Remove x axis labels
        axis.title.y = element_blank())

# Create a plot to compare mast identification using ADCY_simple method
ADCY_simple_plot <- ggplot(ADCY_simple(Dataset1)) +
  geom_point(aes(x = Year, y = Value, color = mast_val)) +
  geom_line(aes(x = Year, y = Value)) +
  ggtitle("Avg. Distance conseq. years") +
  theme(legend.position = "none",
        axis.title.x = element_blank(), # Remove x axis labels
        axis.title.y = element_blank())

# Create a plot to compare mast identification using ADCY_NMIR method
```

```
ADCY_NMIR_plot <- ggplot(ADCY_NMIR(Dataset1)) +
  geom_point(aes(x = Year, y = Value, color = mast_val)) +
  geom_line(aes(x = Year, y = Value)) +
  ggtitle("Avg. dist. conseq. years, no in-row masy. years") +
  theme(legend.position = "none",
        axis.title.x = element_blank(), # Remove x axis labels
        axis.title.y = element_blank())

# Create a plot to compare mast identification using MSD method
MSD_plot <- ggplot(MSD_orig(Dataset1)) +
  geom_point(aes(x = Year, y = Value, color = mast_val)) +
  geom_line(aes(x = Year, y = Value)) +
  geom_line(aes(x = Year, y = MSD_vec(Value)))+
  ggtitle("Mean standard deviates") +
  theme(legend.position = "none",
        axis.title.x = element_blank(),
        axis.title.y = element_blank())

# Arrange the plots in a 2x2 grid for visualization and comparison
grid.arrange(ADCY_simple_plot, ADCY_NMIR_plot, ADM_plot, MSD_plot, nrow = 2)
```

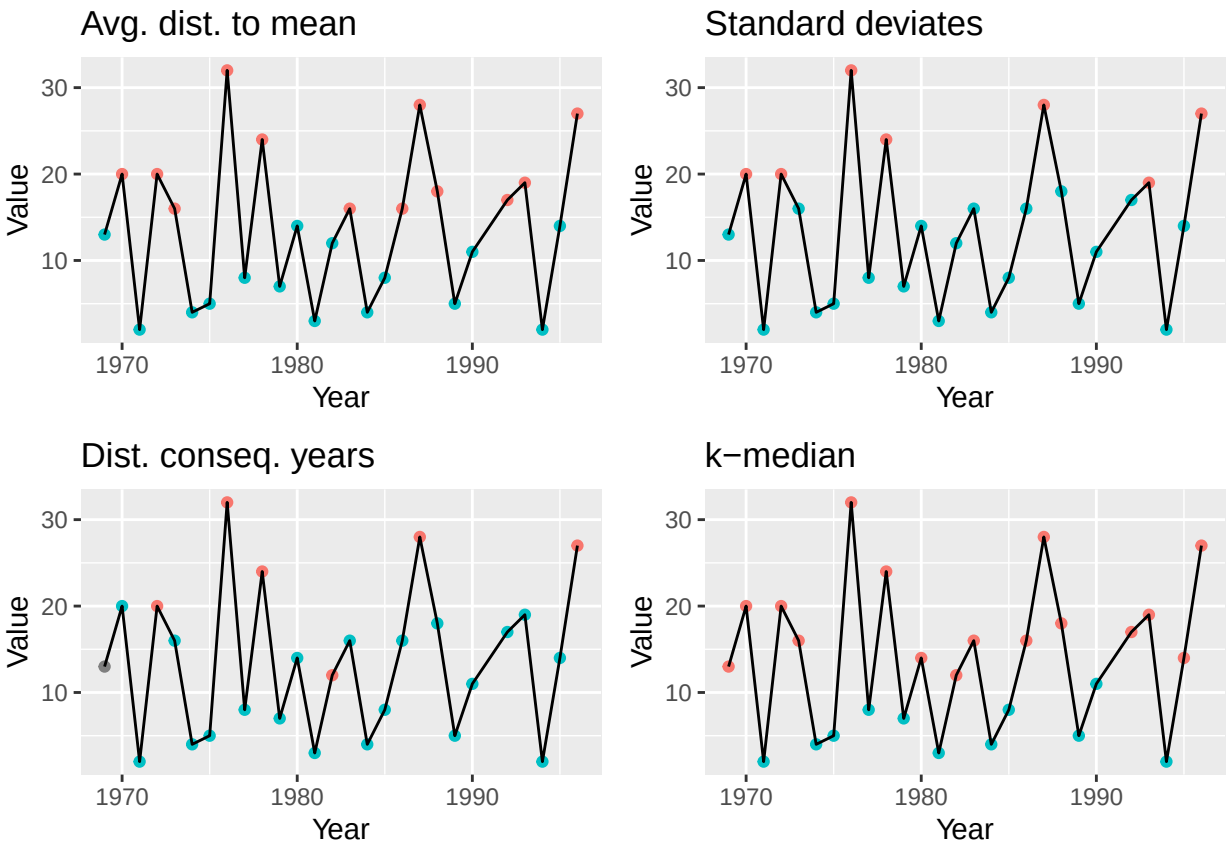


2)

```
Dataset3<-quercus_chapmanii_at_27.2_minus81.3_from_5001

#see that sites are out there
Dataset3$Site_number|>unique()
```

```
## [1] "001" "002" "003"
```

```
#pick the site of interest
s_n<-"003"

#define plots for different methods
ADM_plot <-ggplot(ADM_funct(Dataset3)|>
                    filter( month==1, Site_number==s_n)) +
  geom_point(aes(x = Year, y = Value, color=mast_val)) +
  geom_line(aes(x = Year, y = Value))+
  ggtitle("Avg. dist. to mean")+
  theme(legend.position = "none")


ADCY_plot <- ggplot(ADCY_all_sites(Dataset3, "NMIR")|>
                    filter( month==1, Site_number==s_n)) +
  geom_point(aes(x = Year, y = Value, color=mast_val)) +
  geom_line(aes(x = Year, y = Value))+
  ggtitle("Dist. conseq. years")+
  theme(legend.position = "none")

MSD_plot <- ggplot(MSD_orig(Dataset3)|>
                    filter( month==1, Site_number==s_n)) +
  geom_point(aes(x = Year, y = Value, color=mast_val)) +
  geom_line(aes(x = Year, y = Value))+
  ggtitle("Standard deviates")+
  theme(legend.position = "none")


KMDN_plot <- ggplot(KMDN(Dataset3)|>
                    filter( month==1, Site_number==s_n)) +
  geom_point(aes(x = Year, y = Value, color=mast_val)) +
  geom_line(aes(x = Year, y = Value))+
 ggtitle("k-median")+
  theme(legend.position = "none")



#plot all plots together
grid.arrange(ADM_plot,MSD_plot,ADCY_plot, KMDN_plot, nrow = 2)
```



FIND MASTING PERIOD It might be useful to consider the distribution of distances between mast years (hm years are there between consecutive mast years). This function allows one to get the distances between mast years from the dataset

The provided R code defines a function called "mast_intervals_from_study" that calculates the time intervals (differences) between consecutive "mast years" for each site in the input dataset. The function filters the dataset to only include "mast year" data for a specified month and then computes the time intervals between consecutive mast years for each site.

!!!!ATTENTION!!!! If there are fewer than 2 "mast years" for some site, function will record time interval to be 9999. I.e., if you see number 9999 within your intervals, then there are fewer than 2 "mast years" for some site, and thus there is no option to calculate the difference.

```
# Define a function called "mast_intervals_from_study" that takes a dataset (Dataset) and an optional p
mast_intervals_from_study <- function(Dataset, set_month = 1) {

  # Check if the set_month parameter is valid (between 1 and 12), otherwise raise an error.
  if (!(set_month %in% c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12))) {
    stop('Second parameter is mis-used. Second parameter allows to pick desired month number. Use only
  }

  # Filter the dataset to retain only "mast year" data for the specified month.
  only_mast <- Dataset |>
    filter(mast_val == "mast year", month == set_month)

  # Get unique site numbers from the original dataset.
  sites <- Dataset$Site_number |> unique()

  # Create an empty vector to store the time intervals between consecutive "mast years" for each site.
```

```
  mast_year_diffs <- c()

  # Loop through each site and calculate the time intervals between consecutive "mast years."
  for (i in 1:length(sites)) {
    current_site <- sites[i]

    # Filter the "mast year" data for the current site.
    only_mast_one_site <- only_mast |>
      filter(Site_number == current_site)

    # Get the years corresponding to "mast years" for the current site.
    mast_years <- only_mast_one_site$Year

    # If there are fewer than 2 "mast years" for the current site, set the time interval to 9999.
    if (length(mast_years) < 2) {
      mast_year_diffs <- c(mast_year_diffs, 9999)
    } else {
      # Calculate the time interval (difference) between consecutive "mast years" and store it in mast_
      for (j in 1:length(mast_years) - 1) {
        difference <- mast_years[j + 1] - mast_years[j]
        mast_year_diffs <- c(mast_year_diffs, difference)
      }
    }
  }

  # Return the vector containing the time intervals between consecutive "mast years" for each site.
  return(mast_year_diffs)
}
```

VISUALIZATION OF MASTING INTERVALS at some point we were wondering how intervals between mast years are distributed, so we did this:

The provided R code visualizes the distribution of mast intervals (time intervals between consecutive mast years) for a specific dataset "Dataset4" related to Fagus sylvatica at 49.11.4 from 3024. The code calculates mast intervals using different methods, including ADM (Average Distance to Mean), ADCY (Average Distance to Consecutive Years), MSD (Mean Standard deviates), and KMDN (K-Medians Clustering). The resulting mast intervals are then used to create histograms for each method and arrange them in a 2x2 grid for comparison.

```
#set the dataset for analysis
Dataset4<-picea_engelmannii_at_39.9_minus105.9_from_0234

#set desired number of bins. If NULL, ggplot will use default option, i.e. 30
num_bins<-20


#Obtain vectors of mast intervals
ADM_stuff<-Dataset4|>
        ADM_funct()|>
        mast_intervals_from_study()

ADCY_stuff<-Dataset4|>
        ADCY_all_sites("NMIR")|>
        mast_intervals_from_study()
```

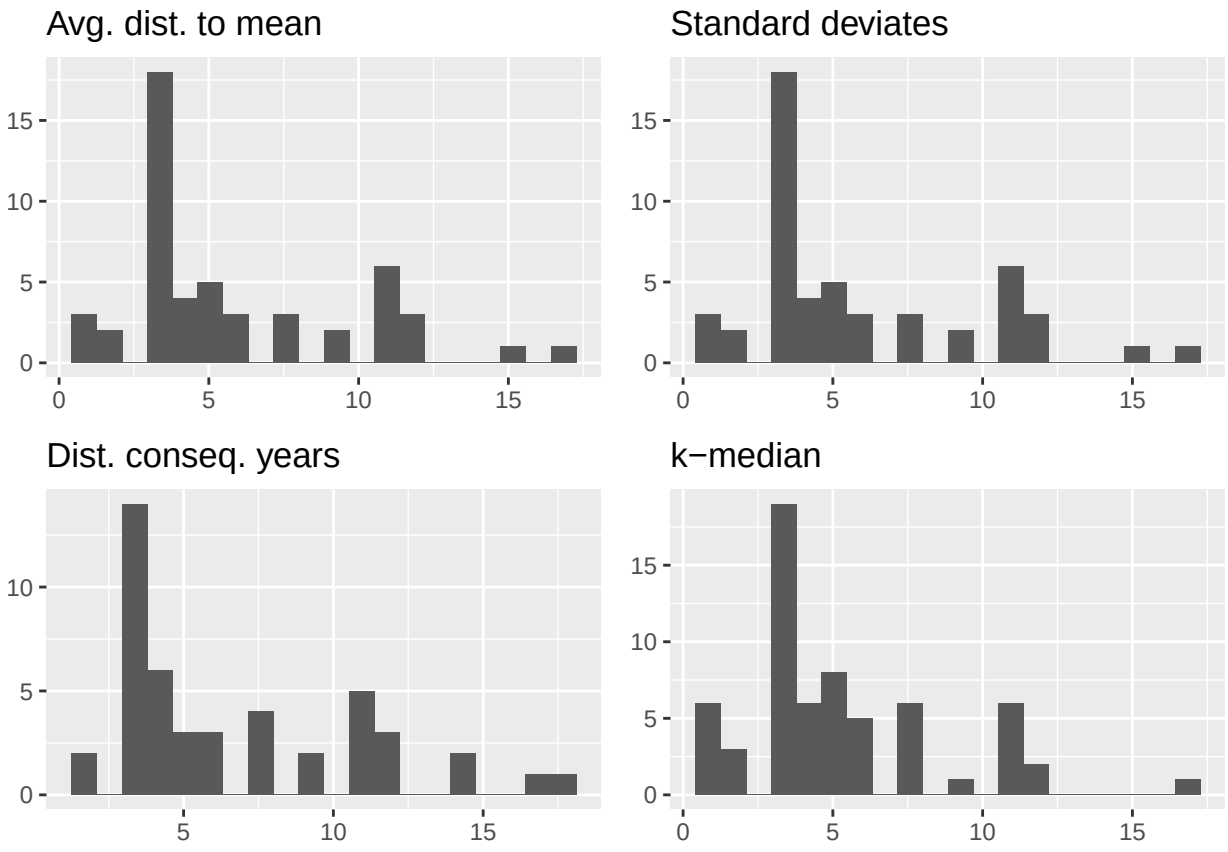
```r
MSD_stuff<-Dataset4|>
        MSD_orig()|>
        mast_intervals_from_study()

KMDN_stuff<-Dataset4|>
        KMDN()|>
        mast_intervals_from_study()

as.tibble(KMDN_stuff)
```

```
## Warning: `as.tibble()` was deprecated in tibble 2.0.0.
## i Please use `as_tibble()` instead.
## i The signature and semantics have changed, see `?as_tibble`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## # A tibble: 63 x 1
##    value
##    <dbl>
##  1     6
##  2    12
##  3     3
##  4     8
##  5     3
##  6     3
##  7     8
##  8     1
##  9     3
## 10     5
## # i 53 more rows
```

```r
#create histograms to show distribution of mast intervals for different thresholds
ADM_plot <-ggplot()+
  aes(
    ADM_stuff[ADM_stuff!=9999]
  ) +
  geom_histogram(bins = num_bins) +
  ggtitle("Avg. dist. to mean")+
  theme(legend.position = "none",
        axis.title.x=element_blank(),
        axis.title.y=element_blank())


ADCY_plot <- ggplot()+
  aes(
    ADCY_stuff[ADCY_stuff!=9999]
          ) +
  geom_histogram(bins = num_bins)+
  ggtitle("Dist. conseq. years")+
  theme(legend.position = "none",
        axis.title.x=element_blank(),
        axis.title.y=element_blank())

MSD_plot <- ggplot()+
```

```
  aes(
    MSD_stuff[MSD_stuff!=9999]
          ) +
  geom_histogram(bins = num_bins) +
  ggtitle("Standard deviates")+
  theme(legend.position = "none",
        axis.title.x=element_blank(),
        axis.title.y=element_blank())


KMDN_plot <- ggplot()+
  aes(
    KMDN_stuff[KMDN_stuff!=9999]
  ) +
  geom_histogram(bins = num_bins) +
 ggtitle("k-median")+
  theme(legend.position = "none",
        axis.title.x=element_blank(),
        axis.title.y=element_blank())




#Show all histograms together
grid.arrange(ADM_plot,MSD_plot,ADCY_plot, KMDN_plot, nrow = 2)
```


Avg. dist. to mean
Standard deviates
Dist. conseq. years
k−median
0
5
10
15