# Stats–pinus sylvestris

Gleb Kozienko

2023-07-18

Upload useful libraries

```
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)

library(EnvStats)

library(infer)

library(lme4)

library(boot)

library(ggResidpanel)
```

Get dataset

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

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)


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

fraxinus_data|>
```

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

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

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

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.

```r
# 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")
```

Set the dataset you want to work with

```
current_dataset<-pinus_sylvestris_at_49.8_22.2_from_6013
```

At some point in our research we noticed the correlations between temperature in some summer months and seed production values. Sara Clifton, our mentor, suggested that we should consider the correlation between the temperature in the hottest month and masting value. This approach worked, yielding good p-values n stuff. Therefore, for all our species, we are testing the correlation between the temperature in the hottest month and masting value.

However, temperature in the hottest month is not the only variable we consider. Multiple articles pointed to the fact that masting correlates with the differences in temperatures in previous years: 1) Vacchiano, Giorgio, et al. “Spatial patterns and broad-scale weather cues of beech mast seeding in Europe.” New Phytologist 215.2 (2017): 595-608. https://nph.onlinelibrary.wiley.com/doi/full/10.1111/nph.14600

2) Kelly, Dave, et al. “Of mast and mean: differential-temperature cue makes mast seeding insensitive to climate change.” Ecology Letters 16.1 (2013): 90-98. https://onlinelibrary.wiley.com/doi/abs/10.1111/ele.12020

Therefore, we incorporate two other variables: * difference between the max temperature in the current year and a year before (i.e., if current year is #0 and year before is #-1, we are looking at (temp diff)=(temp in year#0)-(temp in year #-1) )

- difference between the max temperature in the year before current year and a year two years before current year (i.e., if current year is #0, year before is #-1, and year two years before current year is #-2, we are looking at (temp diff)=(temp in year#-1)-(temp in year #-2) )

Start by obtaining the hottest temp in the year Run chunk below only if you have one site in the dataset

```
current_dataset_hottest_month<-current_dataset|>
  group_by(Year, Value)|>
  summarise(max_t=max(temp_2m))
```

OR

Sometimes you will have a dataset with many sites. Then, before you can proceed with your analysis, you need to average across sites. To do so, use the steps below instead of running the chunk above.

1) First check data for how many sites is present for each year. It might be the case that some years have observations for more sites than other years (it may or may not affect the results you get)

```
current_dataset|>
  group_by(Year)|>
  summarise(num_sites=length(unique(Site_number)))

check_num_sites <- current_dataset|>
  group_by(Year)|>#group by year
  summarise(num_sites=length(unique(Site_number)))#count how many unique site numbers are there for eac



#if the line below outputs one number that is not "1", it means that you have more than one site, and a

#also, if the line below outputs more than one number, it means that you have more than one site, and s
check_num_sites$num_sites|>unique()
```

2) Average across sites

```
# current_dataset_avg_of_sites<-current_dataset|>
#   group_by(Year, month, temp_2m)|>
#   summarise(Value=mean(Value))
```

3) obtaining the hottest temp in the year

```
# current_dataset_hottest_month<-current_dataset_avg_of_sites|>
#   group_by(Year, Value)|>
#   summarise(max_t=max(temp_2m))
```

Set your differences in temperature

```
#create storege for difference values
dT.01<-rep(NA, nrow(current_dataset_hottest_month))

dT.12<-rep(NA, nrow(current_dataset_hottest_month))



#find and record differences
for (i in 1:(nrow(current_dataset_hottest_month)-1)) {

  dT.01[i + 1] <- current_dataset_hottest_month$max_t[i + 1] - current_dataset_hottest_month$max_t[i]

}

for (i in 1:(nrow(current_dataset_hottest_month)-2)) {

  dT.12[i + 2] <- current_dataset_hottest_month$max_t[i + 1] - current_dataset_hottest_month$max_t[i]

}

#add columns with differences to the dataset

current_dataset_hottest_month_dT<-current_dataset_hottest_month

  current_dataset_hottest_month_dT$dT.01<-dT.01

  current_dataset_hottest_month_dT$dT.12<-dT.12
```

Run useful function. See part 3 of the analysis below for explanation why its useful

```
ADM_vec <- function(seed_prod_vals) {
  mean <- mean(seed_prod_vals)
  abs_val_diff <- abs(seed_prod_vals - mean)
  sum <- sum(abs_val_diff)
  n <- length(seed_prod_vals)
  result <- mean + sum / n
  return(result)
}
```

For each variable, we are doing three types of analysis: 1) Assess fitness of the data a) break the connections between explanatory and response variables by randomly reassigning response values to explanatory values. We do it by permutating over the vector of response values. b) for each permutated data, build linear regression model, assess R^2, store it c)Use permutated R^2 distributions to assess the p-value for the R^2 value of the original, not permutated data. We asses p-value using cumulative distribution functions and percentiles.

2) Analyse linear regression model or original data and its slope

3) Analyze the distribution of values of explanatory variables that correlate to a particular responce variable: we are interested to know that temperature value would cause the size of seed output to exceed masting threshold

a) resample original data with replacement
b) for each resample, build a linear model
c) determine masting threshold value; then, using coefficients from linear regression of that particular resampling, find the temperature(or difference in temps) that corresponds to the masting threshold
d) consider the distribution of such temperatures (or differences in temps)

So, lets apply these types of analysis to our three explanatory variables(current hottest temp, and two differences discussed above)

I) correlation between the temperature in the hottest month and masting value.

1) Assess fitness of the data

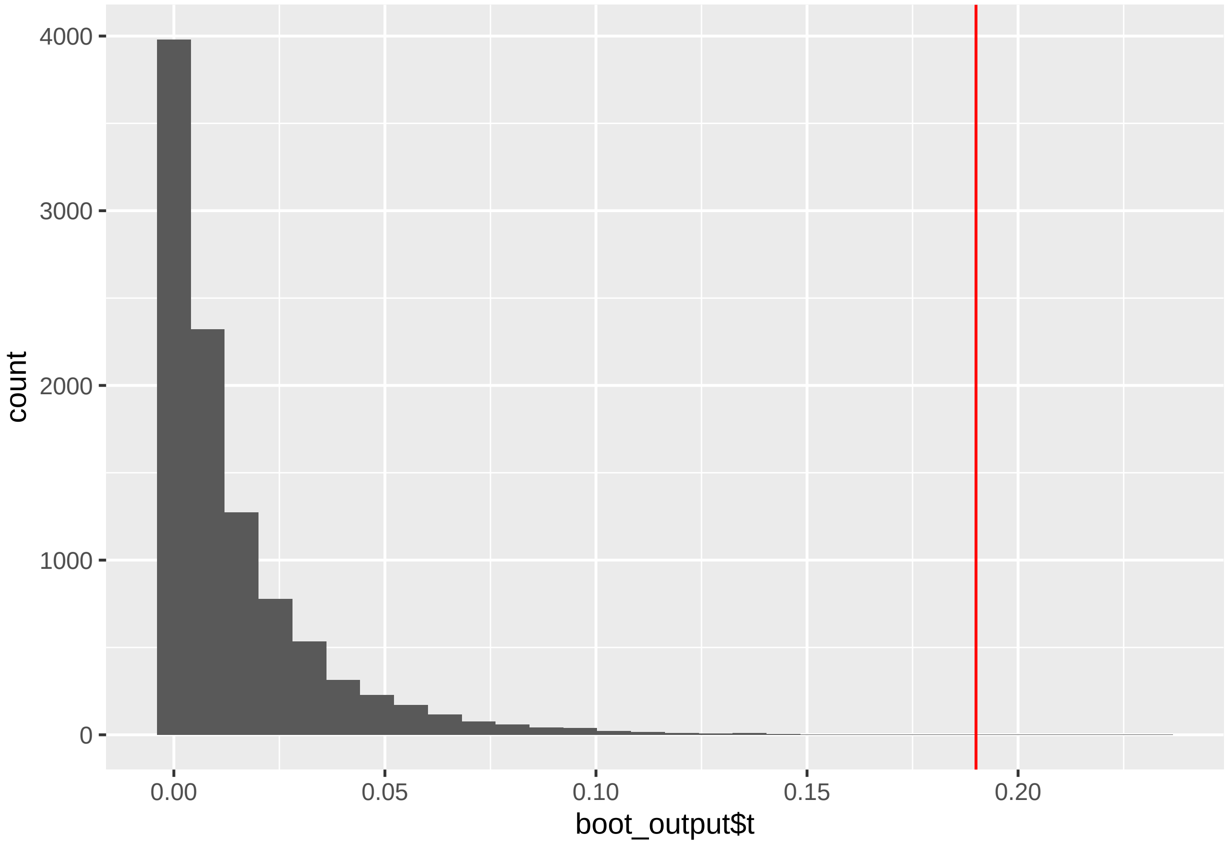
a) This function is used in the boot() function below. It takes permutated the values of response variable from boot() function, makes linear regression of permuted values

```
Rsquares3<- function(Data, idx) {
  #idx parameter is passed into this function by the boot() function
  #each repetition of boot() function permutates indices and supplies permutated indices here
  vals_permutated<-Data[idx,2]$Value #access values in permutated order
  Data_permutated<-Data
  Data_permutated$Value<-vals_permutated #rewrite original values with permutated values



  #build linear regression model
  model<-lm(Value~max_t, data= Data_permutated)
  model_output<-summary(model)
  return(model_output$r.squared)#return R^2 value of the model
}
```

b) do permutations here

```
#conduct 10000 permutations
set.seed(228)#makes sure that boot() generates same output for the same input (because seed defines pse
boot_output<-boot::boot(current_dataset_hottest_month_dT, Rsquares3,10000, sim = "permutation")

#make a histograom of R^2 values
ggplot()+
  aes(boot_output$t)+
  geom_histogram()+
  geom_vline(xintercept = boot_output$t0 ,colour="red")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```r
#confidence interval for permutated R^2 values
boot::boot.ci(boot_output, type = "perc")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot::boot.ci(boot.out = boot_output, type = "perc")
##
## Intervals :
## Level     Percentile
## 95%   ( 0.0000,  0.0715 )
## Calculations and Intervals on Original Scale
```

```r
#original R^2 value
boot_output$t0
```

```
## [1] 0.190026
```

```r
#get p-value
percentile <- ecdf(boot_output$t) #create empirical cumulative distribution function
percentile(boot_output$t0) #assess what percentile of original distribution that corresponds to the ori
```

```
## [1] 0.9998
```

```r
1-percentile(boot_output$t0) #get p-value
```

```
## [1] 2e-04
```

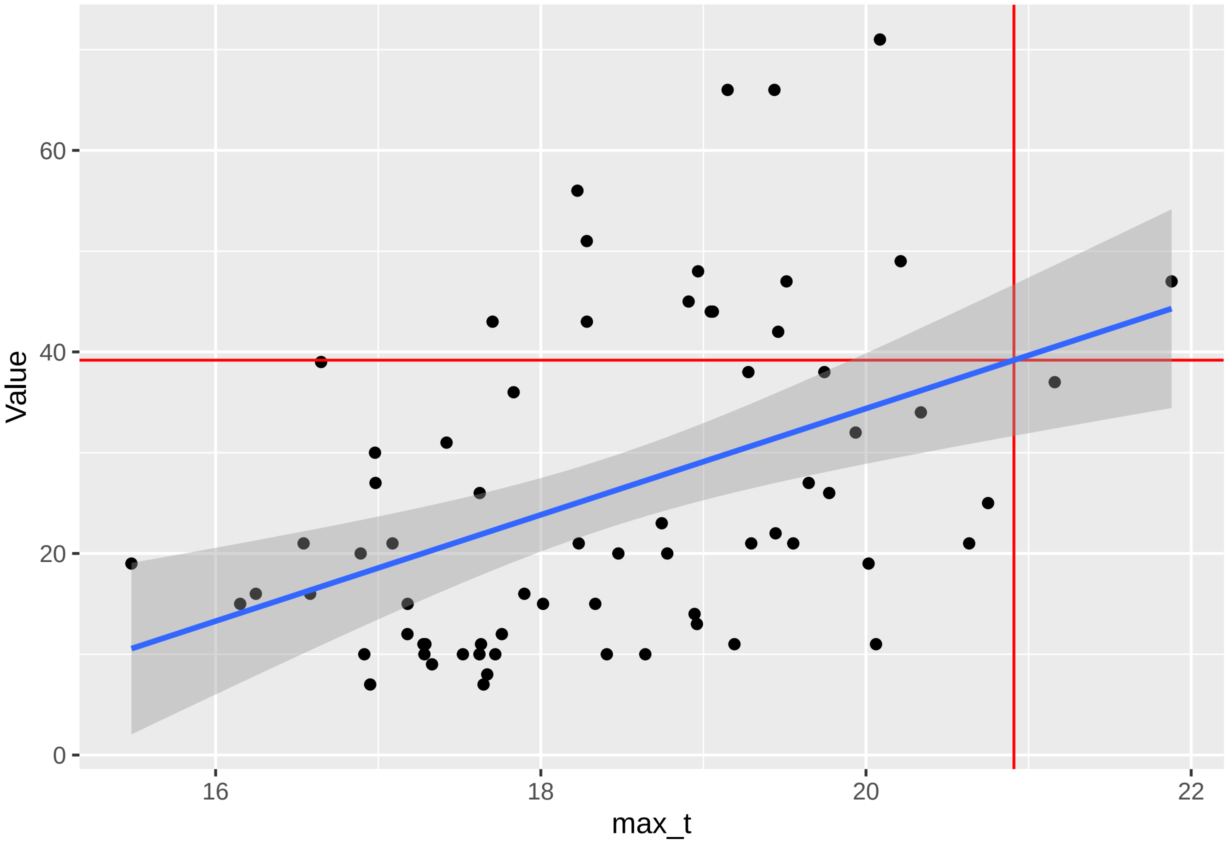
2) Analyse linear regression model or original data and its slope

```
#do linear regressions
lm_0<-lm(Value~max_t, data= current_dataset_hottest_month_dT)
summary(lm_0) #show coeffs and p-vals
```

```
##
## Call:
## lm(formula = Value ~ max_t, data = current_dataset_hottest_month_dT)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -23.709 -10.898  -4.517  10.220  36.163
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -71.093     24.544  -2.896 0.005092 **
## max_t          5.274      1.330   3.965 0.000181 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.52 on 67 degrees of freedom
## Multiple R-squared:   0.19,  Adjusted R-squared:  0.1779
## F-statistic: 15.72 on 1 and 67 DF,  p-value: 0.0001811
```

```
#resid_panel(lm_0)#check how well data fits linear regression model
#extract coefficients of linear regression
b_0=lm_0$coefficients[1]
b_1=lm_0$coefficients[2]

yval<-ADM_vec(current_dataset_hottest_month_dT$Value) #get masting threshold value
xval<-(yval-b_0)/b_1 #get temperature that corresponds with masting threshold value


#present results in the form of a graph
ggplot(current_dataset_hottest_month_dT, mapping= aes(x=max_t, y=Value))+
  geom_point() +
  geom_hline(yintercept = yval,colour="red")+
  geom_vline(xintercept = xval,colour="red")+
  geom_smooth(method="lm")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



3) Analyze the distribution of values of explanatory variables that correlate to a particular responce variable:

a) This function is used in the boot() function below. It takes resampled values from boot() function, makes linear regression of the sample, outputs temp value that corresponds to masting threshold

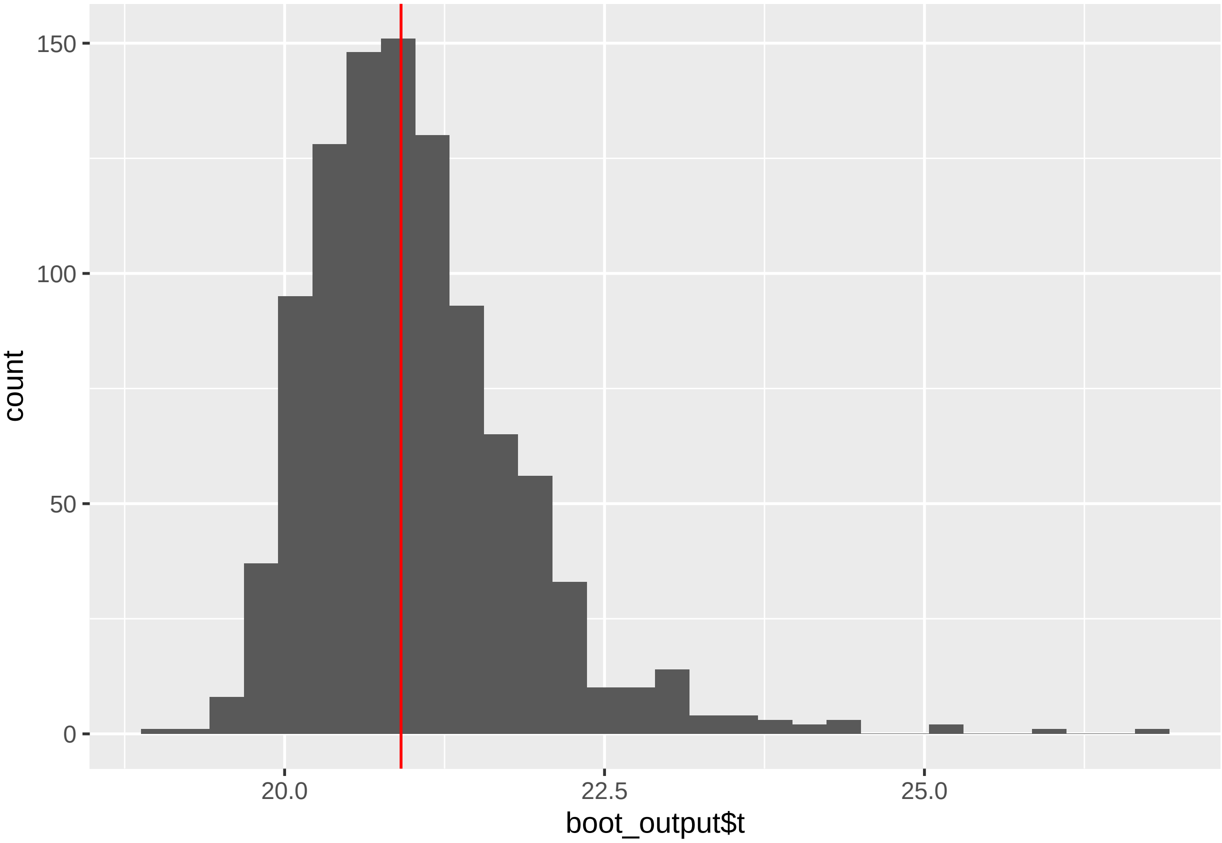
```
temp_dist_stat0<- function(Data, idx) {

  #while its called "Data_permutated", it is actually a resampling with replacement
  Data_permutated<-Data[idx,]
  #build model
  model<-lm(Value~max_t, data= Data_permutated)
  #get coefficients
  b_0=model$coefficients[1]
  b_1=model$coefficients[2]

#get threshold value
yval<-ADM_vec(current_dataset_hottest_month_dT$Value)
xval<-(yval-b_0)/b_1 #get temp value that corresponds with threshold value

  return(xval)
}
```

b) bootstrapping happens here

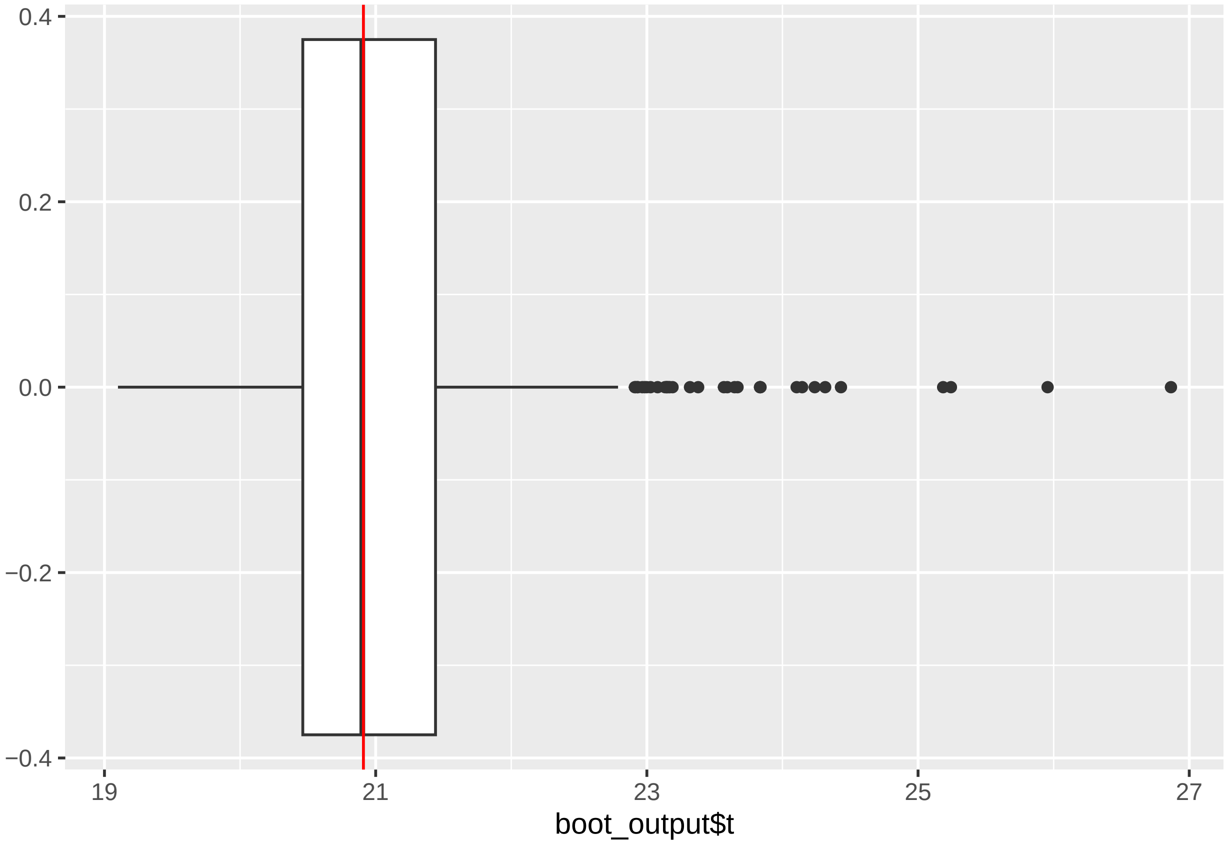
```
set.seed(228)#makes sure that boot() generates same output for the same input (because seed defines pse

#conduct resampling with replacement, collect  temp values that correspond to masting threshold
boot_output<-boot::boot(current_dataset_hottest_month_dT, temp_dist_stat0,1000)
```

```
#temperature distributions as a histogram
ggplot()+
  aes(boot_output$t)+
  geom_histogram()+
  geom_vline(xintercept = boot_output$t0 ,colour="red")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



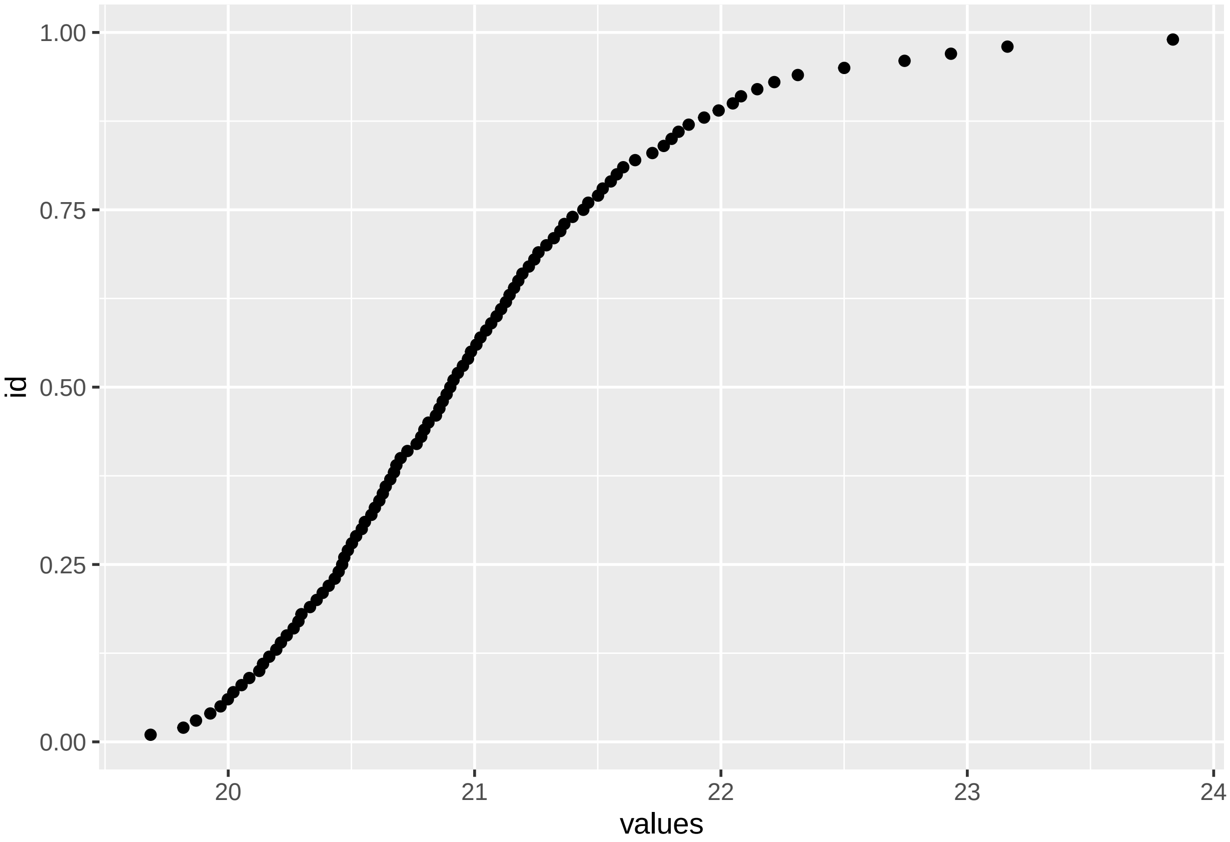
```
# temp distribution as a boxplot
ggplot()+
  geom_boxplot(aes(boot_output$t))+
  geom_vline(xintercept = boot_output$t0 ,colour="red")
```



```
#quantiles of distribution
quantile(boot_output$t, probs =c(0.05, 0.25, 0.5, 0.75, 0.95))
```

```
##       5%      25%      50%      75%      95%
## 19.96894 20.46259 20.90144 21.44136 22.50040
```

```
#build a cumulative distribution function
test<-quantile(boot_output$t, probs = seq(0.01, 0.99, by = 0.01))

y <- data.frame(id = seq(0.01, 0.99, by = 0.01), values = test)

#output of cumulative distribution function
ggplot()+
  geom_point(data = y,aes(y=id, x=values))
```



II) correlation between the temp difference(max temperature in the current year and a year before) and masting value.

1) Assess fitness of the data

a) This function is used in the boot() function below. It takes permutated the values of response variable from boot() function, makes linear regression of permuted values

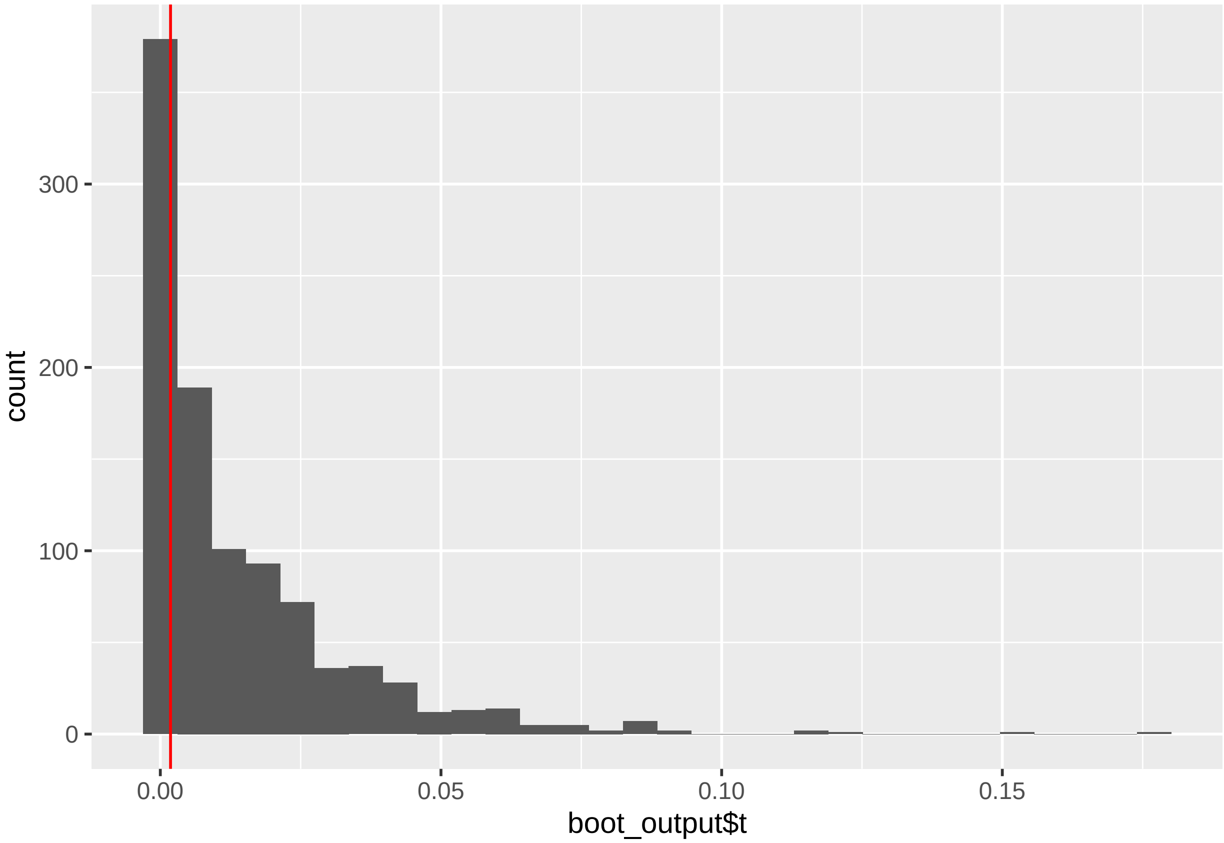
```
Rsquares4<- function(Data, idx) {
  #idx parameter is passed into this function by the boot() function
  #each repetition of boot() function permutates indices and supplies permutated indices here
  vals_permutated<-Data[idx,2]$Value #access values in permutated order
  Data_permutated<-Data
  Data_permutated$Value<-vals_permutated #rewrite original values with permutated values



  #build linear regression model
  model<-lm(Value~dT.01, data= Data_permutated)
  model_output<-summary(model)
  return(model_output$r.squared)#return R^2 value of the model
}
```

b) do permutations here

```
#conduct 10000 permutations
set.seed(228)#makes sure that boot() generates same output for the same input (because seed defines pse
boot_output<-boot::boot(current_dataset_hottest_month_dT, Rsquares4,1000, sim = "permutation")
```

```
#make a histograom of R^2 values
ggplot()+
  aes(boot_output$t)+
  geom_histogram()+
  geom_vline(xintercept = boot_output$t0 ,colour="red")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
#confidence interval for permutated R^2 values
boot::boot.ci(boot_output, type = "perc")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot::boot.ci(boot.out = boot_output, type = "perc")
##
## Intervals :
## Level     Percentile
## 95%   ( 0.0000,  0.0665 )
## Calculations and Intervals on Original Scale
```

```
#original R^2 value
boot_output$t0
```

```
## [1] 0.001817505
```

```
#get p-value
percentile <- ecdf(boot_output$t) #create empirical cumulative distribution function
```

```
percentile(boot_output$t0) #assess what percentile of original distribution that corresponds to the ori
```

```
## [1] 0.308
```

```
1-percentile(boot_output$t0) #get p-value
```

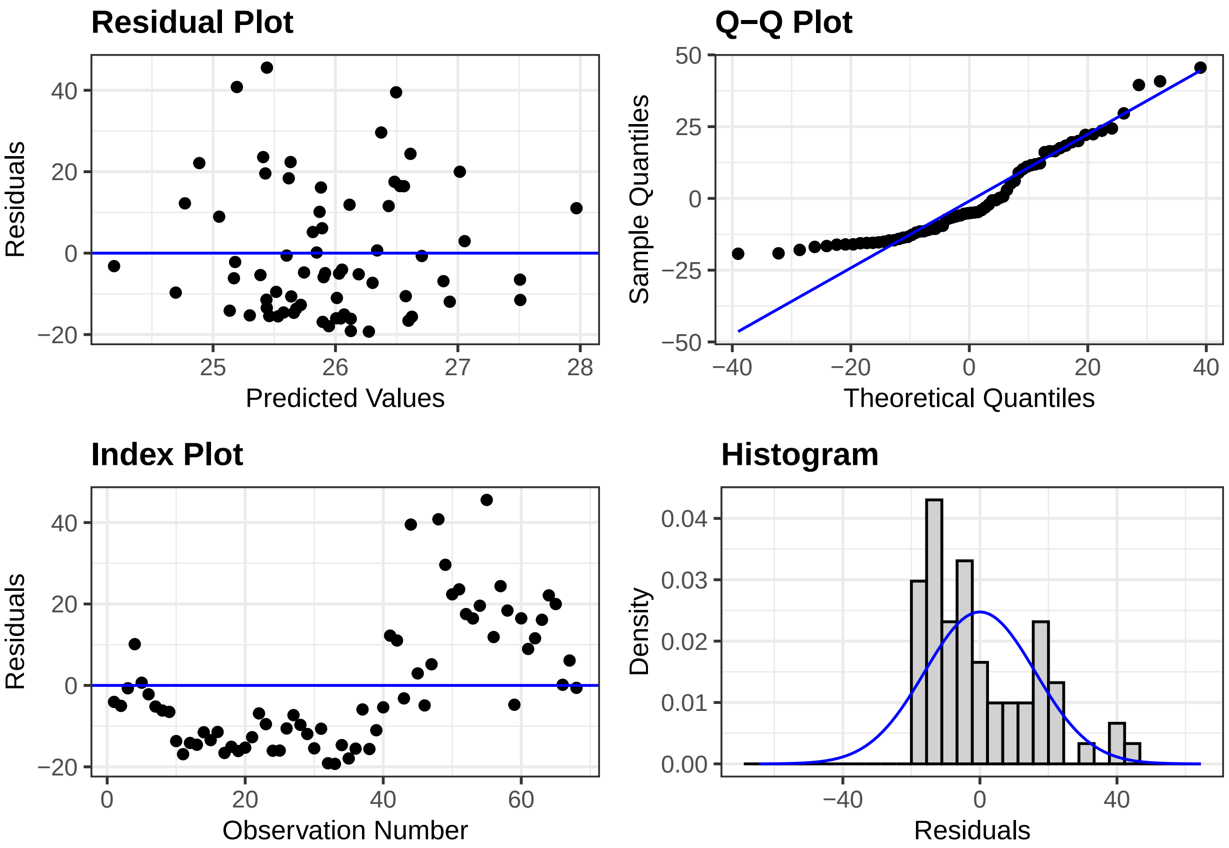
```
## [1] 0.692
```

2) Analyse linear regression model or original data and its slope

```
#do linear regression
lm_01<-lm(Value~dT.01, data= current_dataset_hottest_month_dT)
summary(lm_01)#show coeffs and p-vals
```

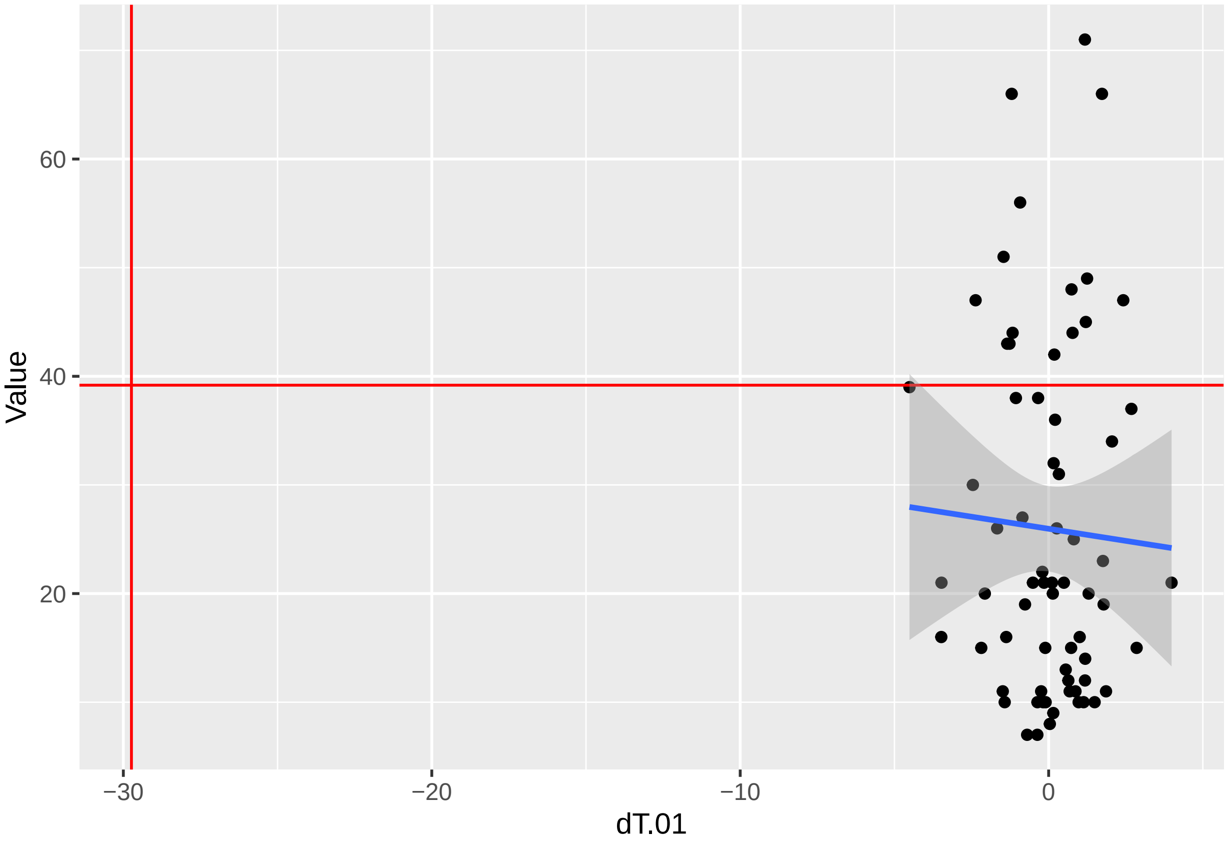
```
##
## Call:
## lm(formula = Value ~ dT.01, data = current_dataset_hottest_month_dT)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -19.27 -13.50  -5.11  11.64  45.56
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  25.9631     1.9691  13.185   <2e-16 ***
## dT.01        -0.4446     1.2824  -0.347     0.73
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.24 on 66 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.001818,   Adjusted R-squared:  -0.01331
## F-statistic: 0.1202 on 1 and 66 DF,  p-value: 0.7299
```

```
resid_panel(lm_01)#check how well data fits linear regression model
```



```r
#extract coefficients of linear regression
b_0=lm_01$coefficients[1]
b_1=lm_01$coefficients[2]

yval<-ADM_vec(current_dataset_hottest_month_dT$Value) #get masting threshold value
xval<-(yval-b_0)/b_1 #get temperature that corresponds with masting threshold value


#present results in the form of a graph
ggplot(current_dataset_hottest_month_dT, mapping= aes(x=dT.01, y=Value))+
  geom_point() +
 geom_hline(yintercept = yval,colour="red")+
  geom_vline(xintercept = xval,colour="red")+
  geom_smooth(method="lm")
```

```
## `geom_smooth()` using formula = 'y ~ x'

## Warning: Removed 1 rows containing non-finite values (`stat_smooth()`).

## Warning: Removed 1 rows containing missing values (`geom_point()`).
```



We do not attempt to assess the distribution of values of explanatory variables that correlate to the masting threshold value for temp difference(max temperature in the current year and a year before), because for all species we had examined, this variable did not show a significant correlation with masting

III) correlation between the temp difference(difference between the max temperature in the year before current year and a year two years before current year) and masting value.

1) Assess fitness of the data

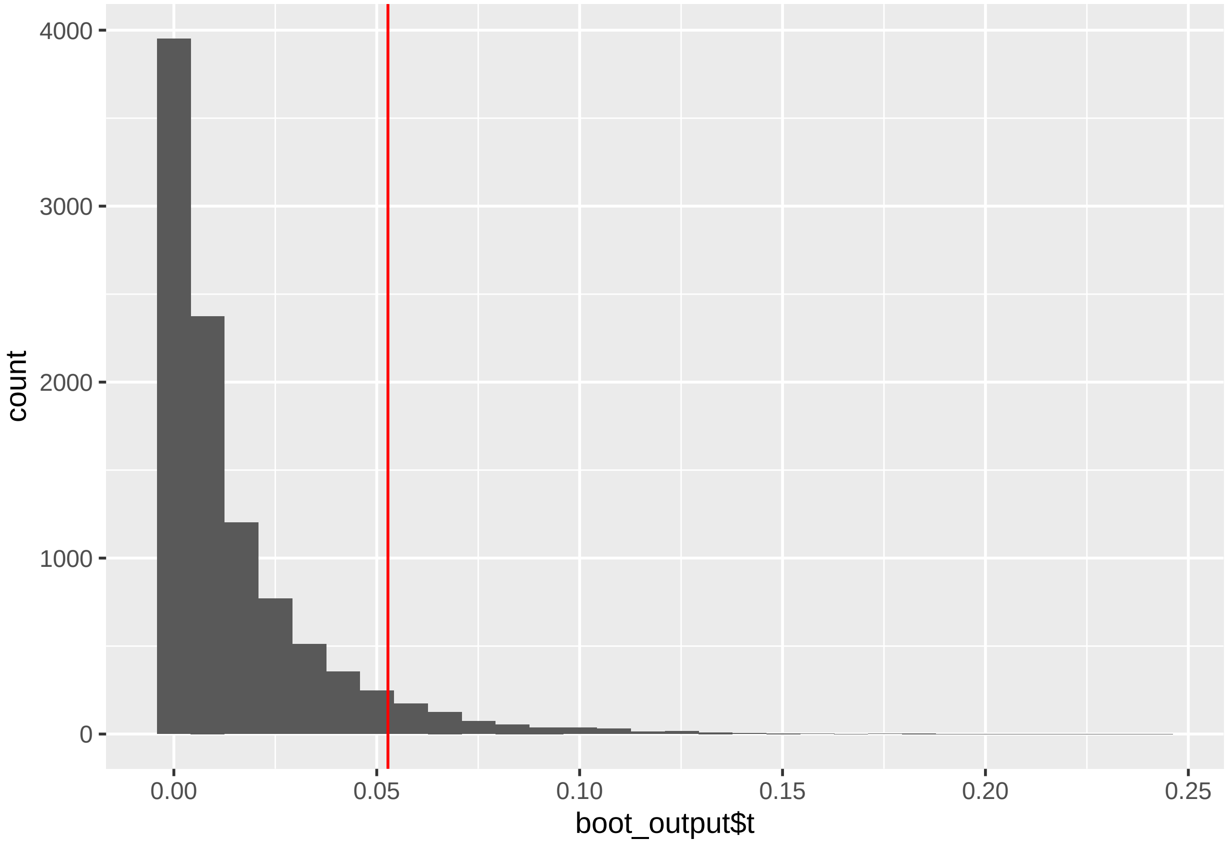
a) This function is used in the boot() function below. It takes permutated the values of response variable from boot() function, makes linear regression of permuted values

```
Rsquares5<- function(Data, idx) {
  #idx parameter is passed into this function by the boot() function
  #each repetition of boot() function permutates indices and supplies permutated indices here
  vals_permutated<-Data[idx,2]$Value #access values in permutated order
  Data_permutated<-Data
  Data_permutated$Value<-vals_permutated #rewrite original values with permutated values



  #build linear regression model
  model<-lm(Value~dT.12, data= Data_permutated)
  model_output<-summary(model)
  return(model_output$r.squared)#return R^2 value of the model
}
```

b) do permutations here

```r
#conduct 10000 permutations
set.seed(228)#makes sure that boot() generates same output for the same input (because seed defines pse
boot_output<-boot::boot(current_dataset_hottest_month_dT, Rsquares5,10000, sim = "permutation")

#make a histograom of R^2 values
ggplot()+
  aes(boot_output$t)+
  geom_histogram()+
  geom_vline(xintercept = boot_output$t0 ,colour="red")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```r
#confidence interval for permutated R^2 values
boot::boot.ci(boot_output, type = "perc")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot::boot.ci(boot.out = boot_output, type = "perc")
##
## Intervals :
## Level     Percentile
## 95%   ( 0.0000,  0.0754 )
## Calculations and Intervals on Original Scale
```

```r
#original R^2 value
boot_output$t0
```

```
## [1] 0.05275816
```

```
#get p-value
percentile <- ecdf(boot_output$t) #create empirical cumulative distribution function
percentile(boot_output$t0) #assess what percentile of original distribution that corresponds to the ori
```

```
## [1] 0.938
```

```
1-percentile(boot_output$t0) #get p-value
```

```
## [1] 0.062
```

2) Analyse linear regression model or original data and its slope

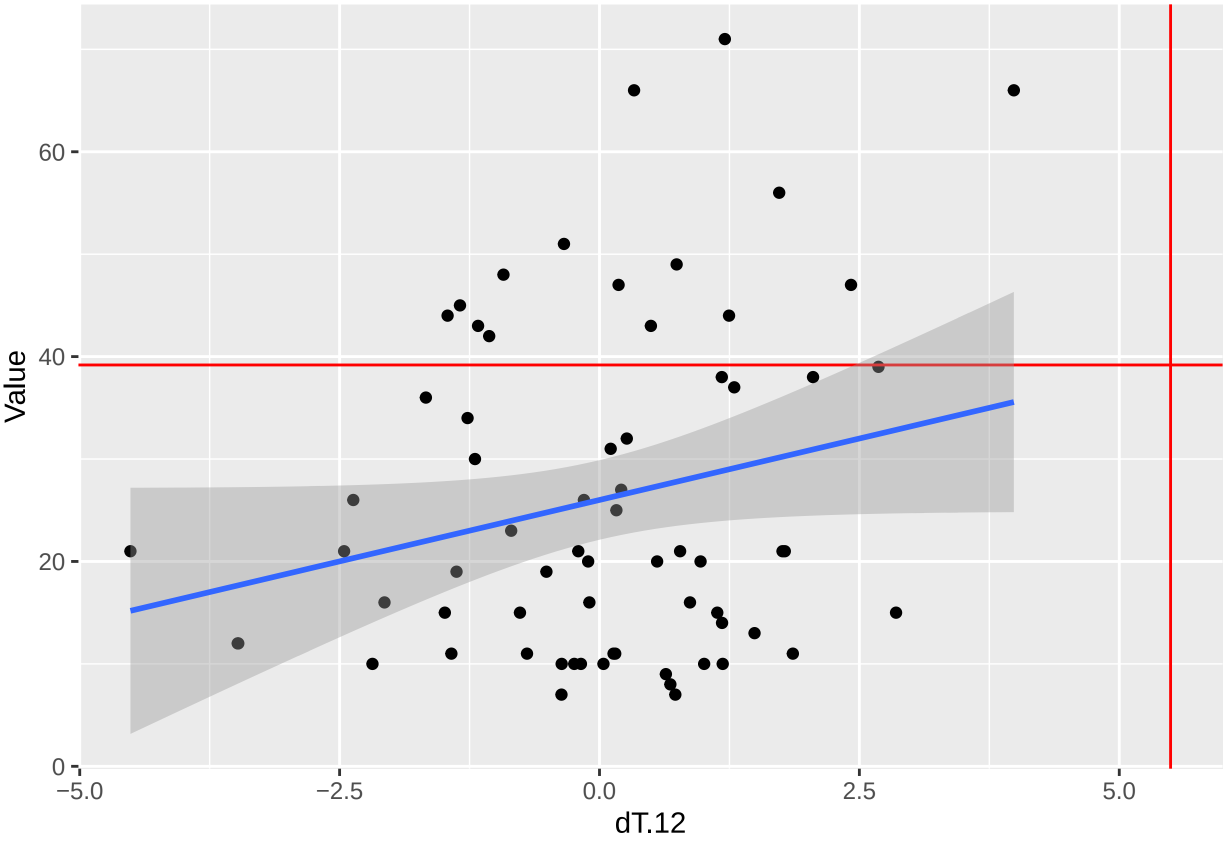
```
#do linear regression
lm_12<-lm(Value~dT.12, data= current_dataset_hottest_month_dT)
summary(lm_12)#show coeffs and p-vals
```

```
##
## Call:
## lm(formula = Value ~ dT.12, data = current_dataset_hottest_month_dT)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -20.755 -13.527  -5.043  10.106  42.103
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   26.005      1.946  13.361   <2e-16 ***
## dT.12          2.399      1.261   1.903   0.0615 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.93 on 65 degrees of freedom
##   (2 observations deleted due to missingness)
## Multiple R-squared:  0.05276,    Adjusted R-squared:  0.03819
## F-statistic:  3.62 on 1 and 65 DF,  p-value: 0.06151
```

```
#resid_panel(lm_12)#check how well data fits linear regression model
#extract coefficients of linear regression
b_0=lm_12$coefficients[1]
b_1=lm_12$coefficients[2]

yval<-ADM_vec(current_dataset_hottest_month_dT$Value) #get masting threshold value
xval<-(yval-b_0)/b_1 #get temperature that corresponds with masting threshold value


#present results in the form of a graph
ggplot(current_dataset_hottest_month_dT, mapping= aes(x=dT.12, y=Value))+
  geom_point() +
 geom_hline(yintercept = yval,colour="red")+
  geom_vline(xintercept = xval,colour="red")+
  geom_smooth(method="lm")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## Warning: Removed 2 rows containing non-finite values (`stat_smooth()`).
## Warning: Removed 2 rows containing missing values (`geom_point()`).
```



3) Analyze the distribution of values of explanatory variables that correlate to a particular responce variable:

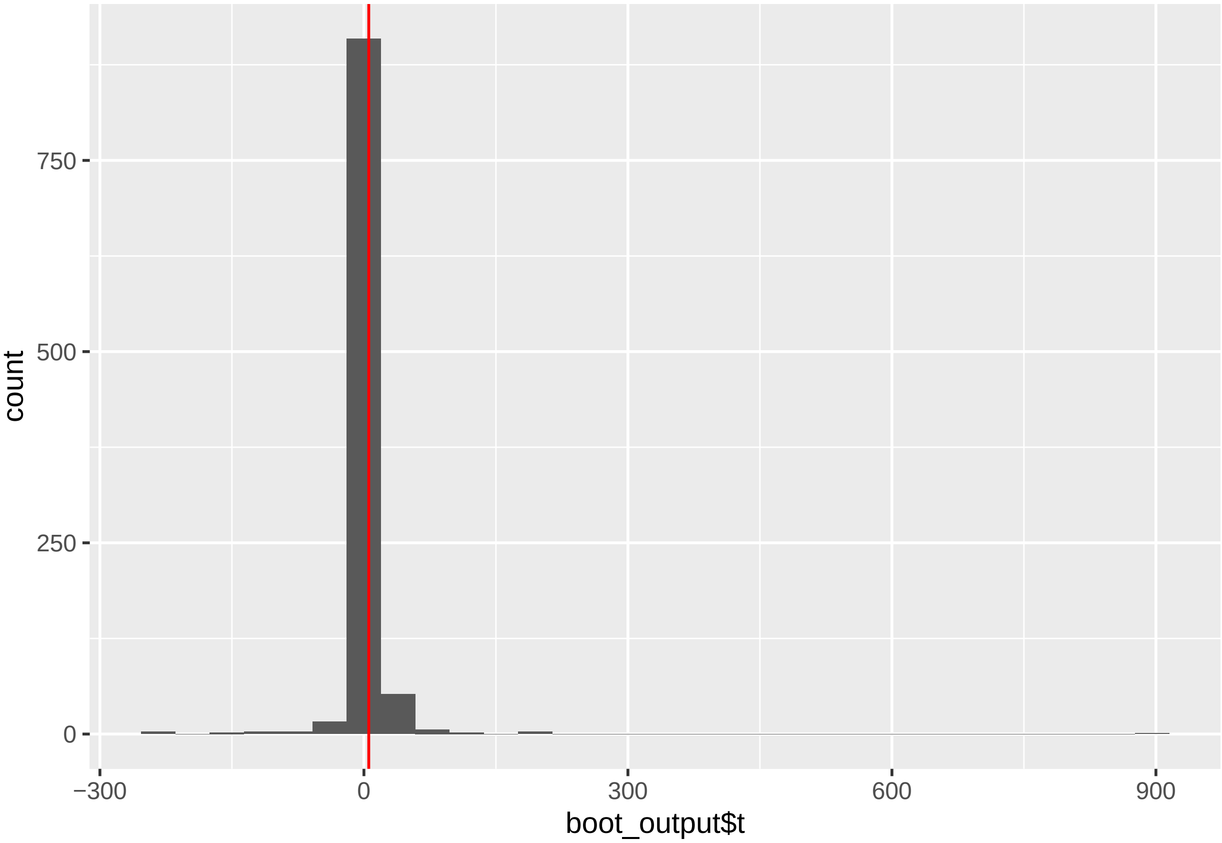
a) This function is used in the boot() function below. It takes resampled values from boot() function, makes linear regression of the sample, outputs temp value that corresponds to masting threshold

```
temp_dist_stat12<- function(Data, idx) {

  #while its called "Data_permutated", it is actually a resampling with replacement
  Data_permutated<-Data[idx,]
  #build model
  model<-lm(Value~dT.12, data= Data_permutated)
  #get coefficients
  b_0=model$coefficients[1]
  b_1=model$coefficients[2]

#get threshold value
yval<-ADM_vec(current_dataset_hottest_month_dT$Value)
xval<-(yval-b_0)/b_1 #get temp value that corresponds with threshold value

  return(xval)
}
```

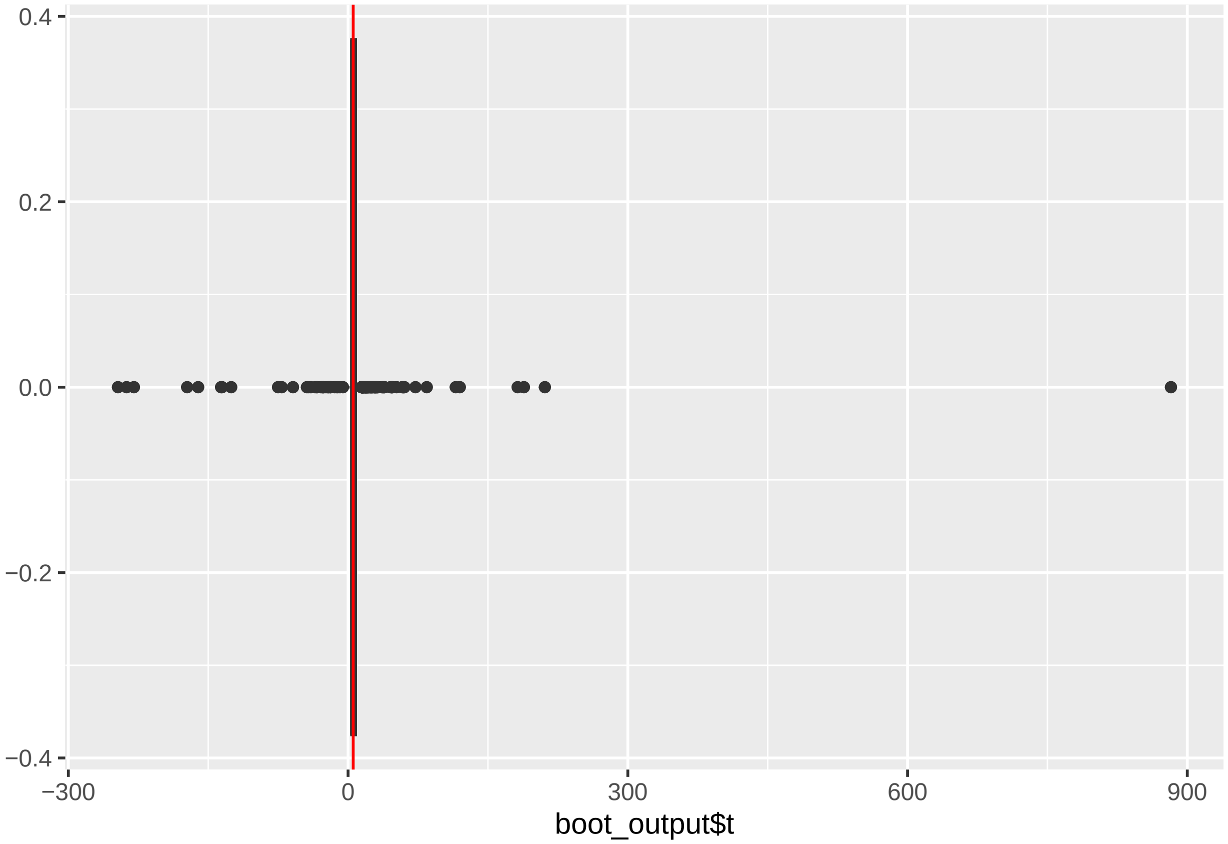
b) bootstrapping happens here

```
set.seed(228)#makes sure that boot() generates same output for the same input (because seed defines pse
#conduct resampling with replacement, collect  temp values that correspond to masting threshold
boot_output<-boot::boot(current_dataset_hottest_month_dT, temp_dist_stat12,1000)


#temperature distributions as a histogram
ggplot()+
  aes(boot_output$t)+
  geom_histogram()+
  geom_vline(xintercept = boot_output$t0 ,colour="red")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



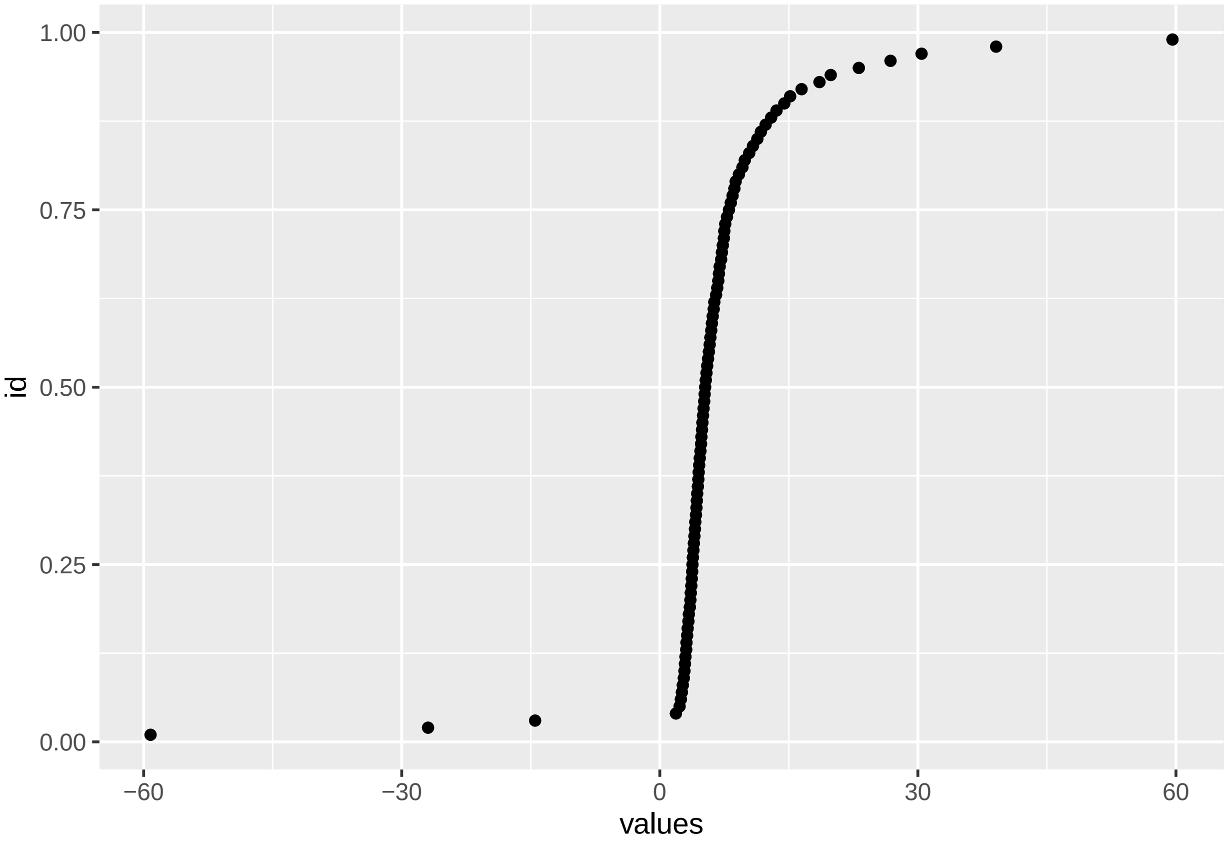
```
# temp distribution as a boxplot
ggplot()+
  geom_boxplot(aes(boot_output$t))+
  geom_vline(xintercept = boot_output$t0 ,colour="red")
```



```r
#quantiles of distribution
quantile(boot_output$t, probs =c(0.05, 0.25, 0.5, 0.75, 0.95))
```

```
##        5%       25%       50%       75%       95%
##  2.298367  3.800620  5.273703  8.040912 23.136990
```

```r
#build a cumulative distribution function
test<-quantile(boot_output$t, probs = seq(0.01, 0.99, by = 0.01))

y <- data.frame(id = seq(0.01, 0.99, by = 0.01), values = test)

#output of cumulative distribution function
ggplot()+
  geom_point(data = y,aes(y=id, x=values))
```



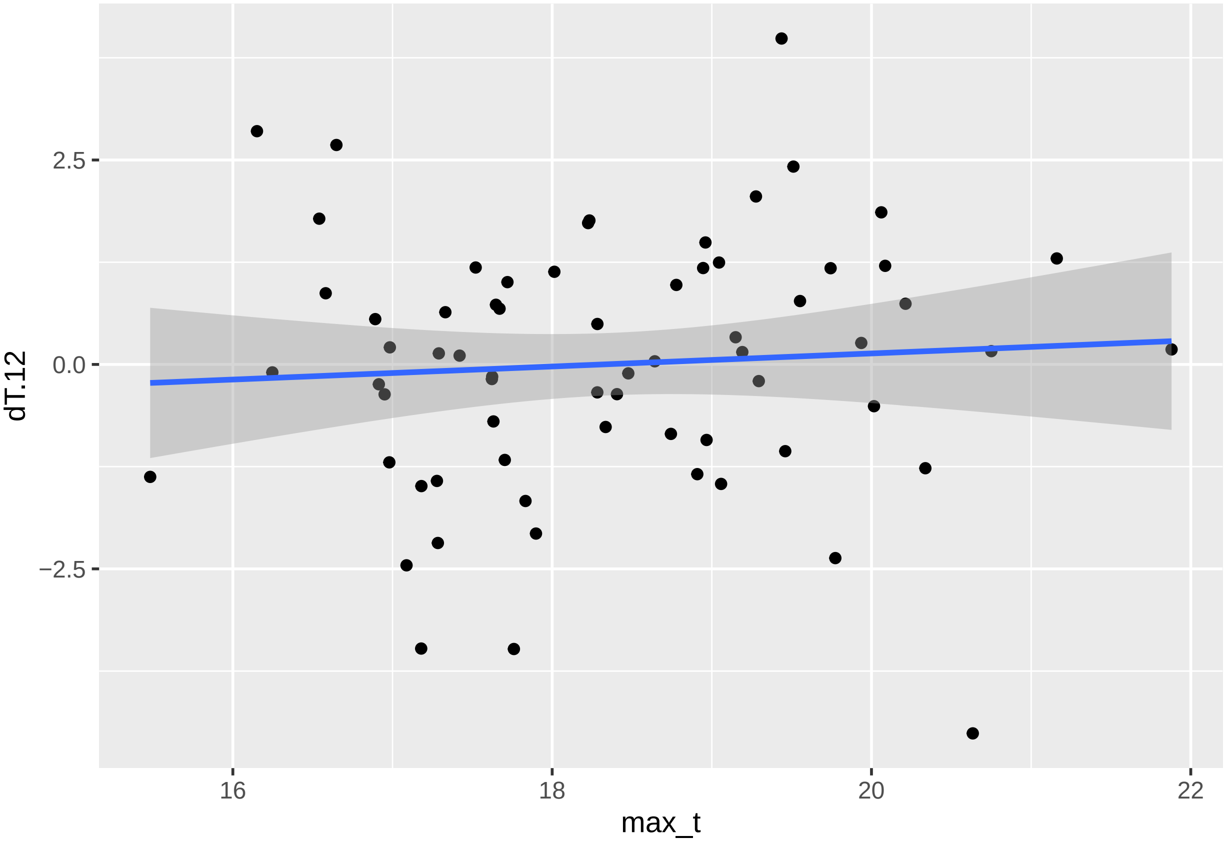
EXTRA: SOME ANALYSIS THAT WE DID, BUT DID NOT FIND IT Particularly HELPFUL TAKE A LOOK IF CURIOUS

I) For some species, we found that both maximum temp in a given month and temp difference(difference between the max temperature in the year before current year and a year two years before current year) show correlation with seed production. Thus, we decided to check if there is a correlation between maximum temp in a given month and temp difference(difference between the max temperature in the year before current year and a year two years before current year). For species that we looked for, such correlation was absent.

You can assess this correlation with the code below.

```
ggplot(current_dataset_hottest_month_dT, mapping= aes(y=dT.12, x=max_t))+
  geom_point() +
  geom_smooth(method="lm")
```

```
## `geom_smooth()` using formula = 'y ~ x'

## Warning: Removed 2 rows containing non-finite values (`stat_smooth()`).

## Warning: Removed 2 rows containing missing values (`geom_point()`).
```



II) After we observed significant slope for correlation between the max temp in a year and seed production, we decided to see if the trend will still be there if we consider mast years and nonmast years separately. For species that we checked, after we correlation between the max temp in a year and seed production was shown to be insignificant if we consider mast and nonmast years separately.

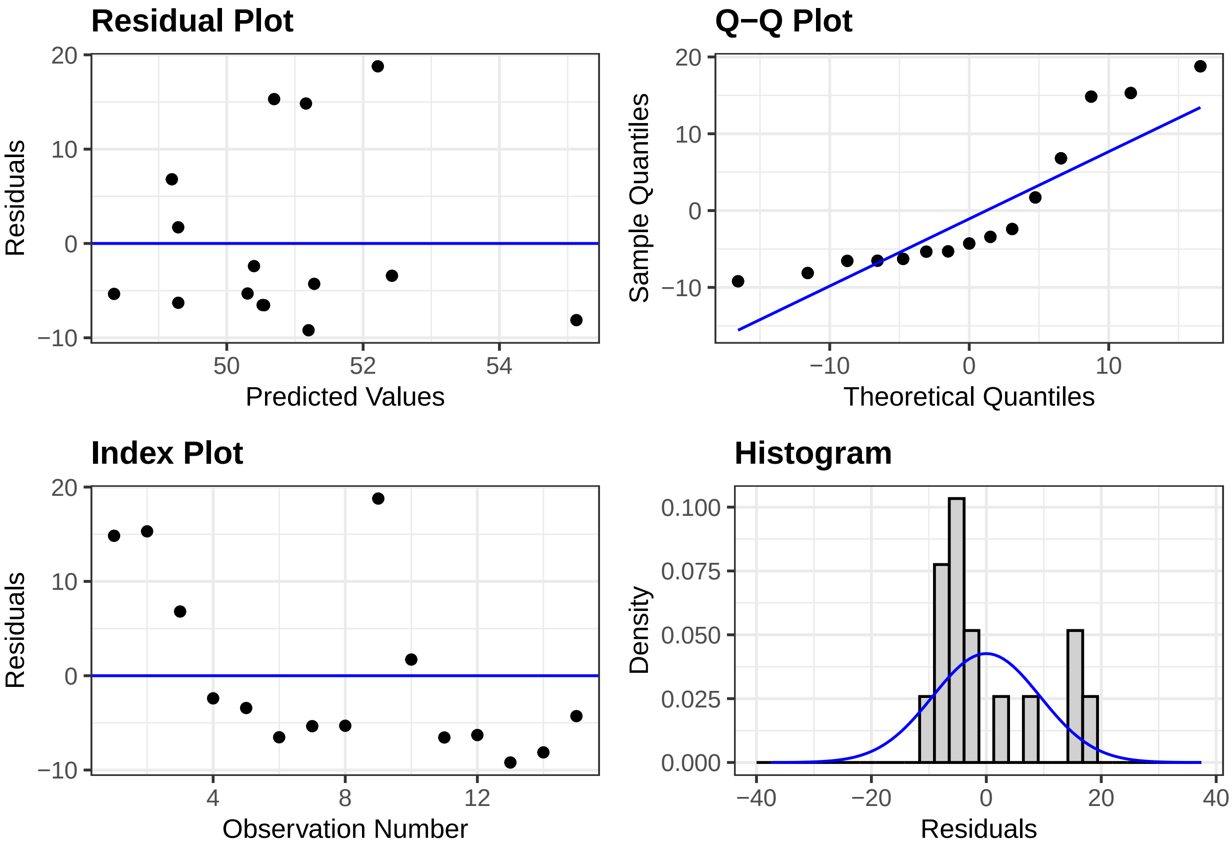
Here is the code:

```
mast_treshold<-ADM_vec(current_dataset_hottest_month_dT$Value) #define threshold

#get data for mast years only
current_dataset_hottest_month_dT_mast<-current_dataset_hottest_month_dT|>
  filter(Value>=mast_treshold)

#get data for non-mast years only
current_dataset_hottest_month_dT_nonmast<-current_dataset_hottest_month_dT|>
  filter(Value<mast_treshold)



# for mast years:

#conduct linear regression
lm_0<-lm(Value~max_t, data= current_dataset_hottest_month_dT_mast)
summary(lm_0)
```

```
##
## Call:
## lm(formula = Value ~ max_t, data = current_dataset_hottest_month_dT_mast)
```

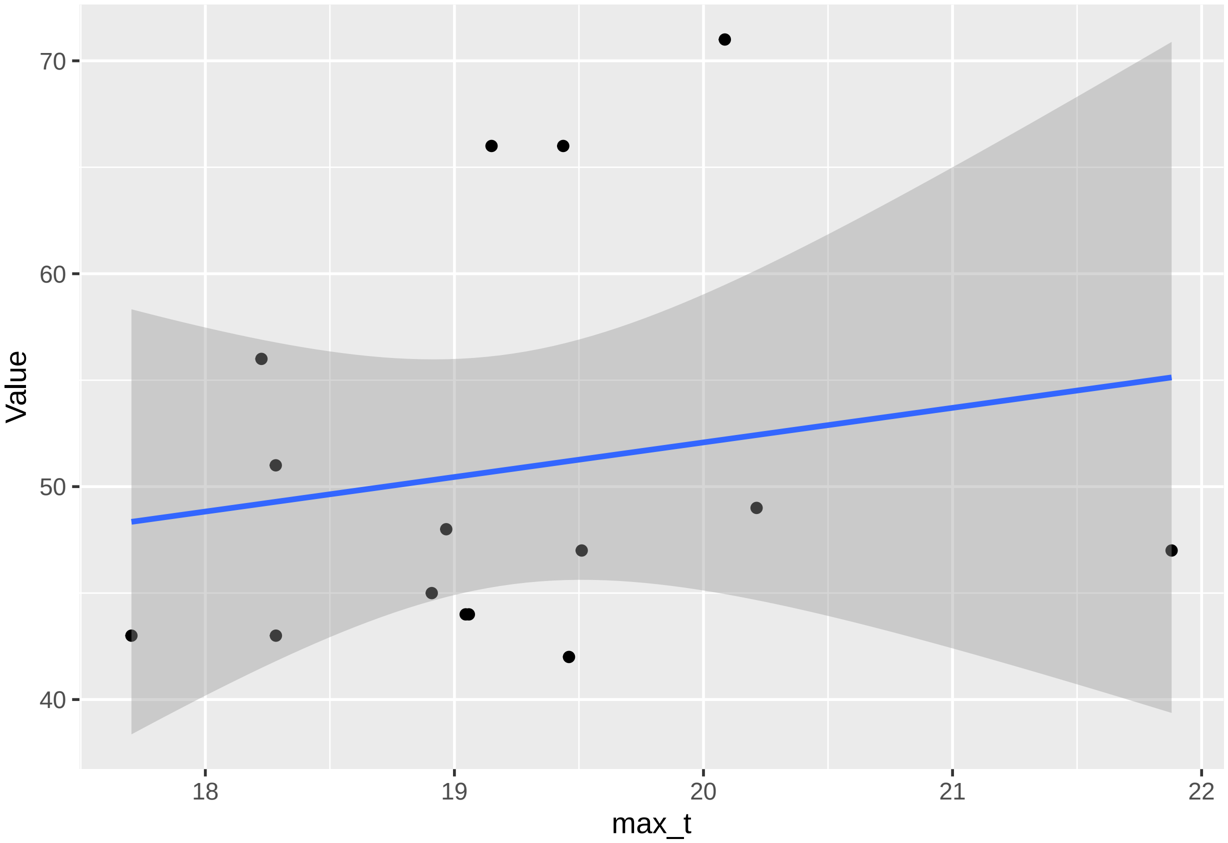
```
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.200 -6.407 -4.283  4.259 18.784
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   19.595     49.439   0.396    0.698
## max_t          1.624      2.570   0.632    0.538
##
## Residual standard error: 9.706 on 13 degrees of freedom
## Multiple R-squared:  0.02981,    Adjusted R-squared:  -0.04482
## F-statistic: 0.3994 on 1 and 13 DF,  p-value: 0.5383
```

```
#assess residuals
resid_panel(lm_0)
```



```
#see scatterplot of the relationship
ggplot(current_dataset_hottest_month_dT_mast, mapping= aes(x=max_t, y=Value))+
  geom_point() +
  geom_smooth(method="lm")
```

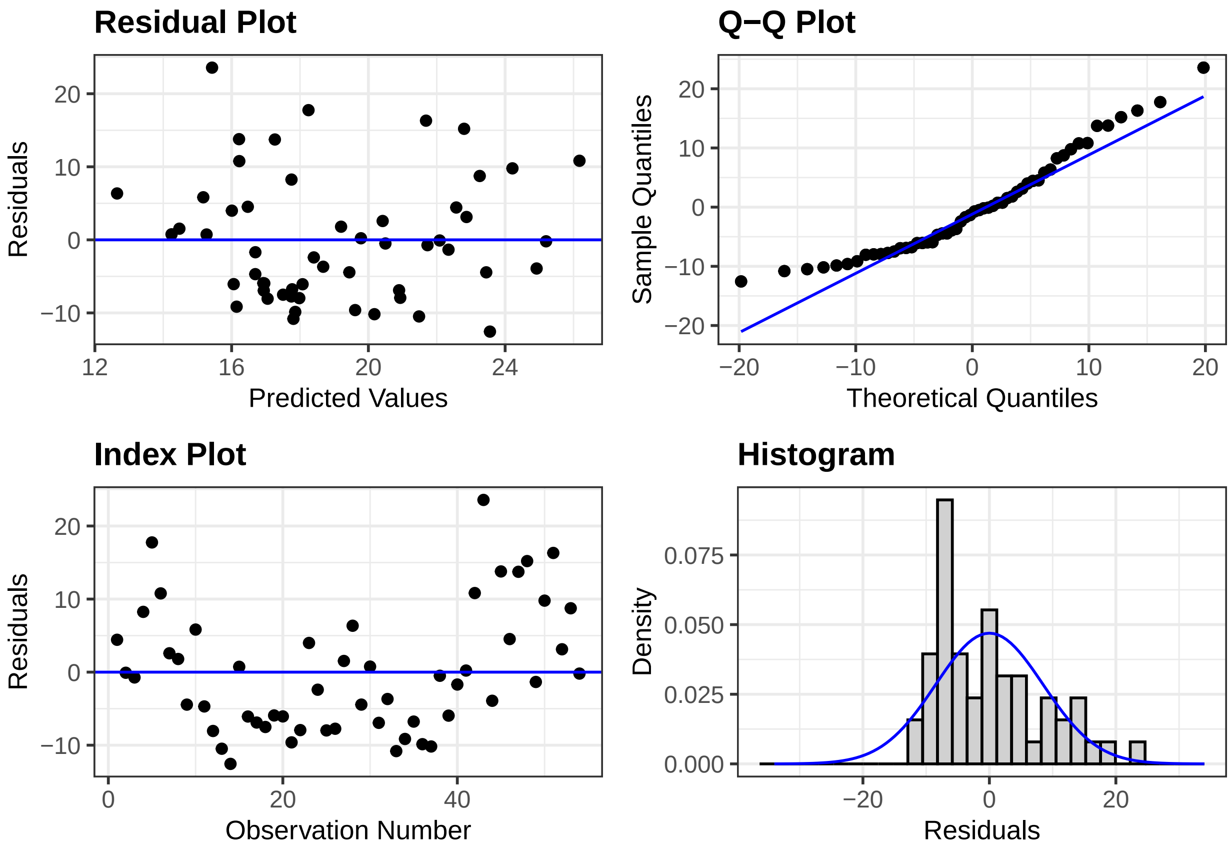
```
## `geom_smooth()` using formula = 'y ~ x'
```



```
#for nonmast years

#conduct linear regression
lm_0<-lm(Value~max_t, data= current_dataset_hottest_month_dT_nonmast)
summary(lm_0)
```

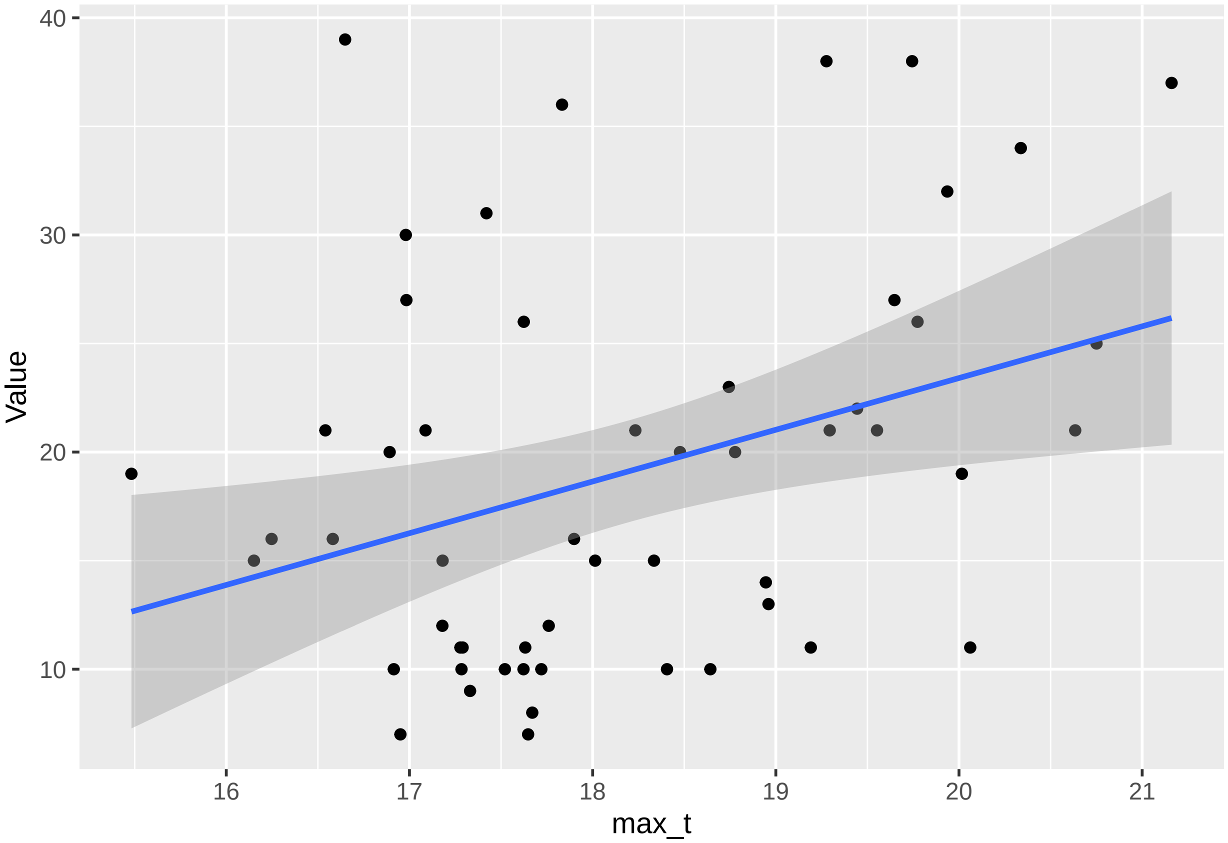
```
##
## Call:
## lm(formula = Value ~ max_t, data = current_dataset_hottest_month_dT_nonmast)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.556  -6.866  -1.036   4.503  23.573
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -24.2259    16.2847  -1.488   0.1429
## max_t         2.3818     0.8935   2.666   0.0102 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.582 on 52 degrees of freedom
## Multiple R-squared:  0.1202, Adjusted R-squared:  0.1033
## F-statistic: 7.107 on 1 and 52 DF,  p-value: 0.01021
```

```
#assess residuals
resid_panel(lm_0)
```



```
#see scatterplot of the relationship
ggplot(current_dataset_hottest_month_dT_nonmast, mapping= aes(x=max_t, y=Value))+
  geom_point() +
  geom_smooth(method="lm")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



III) At some point we were wondering if seed production correlates with years since last masting event. Basically it is a test of the resource storage hypothesis, also known as resource budget model (description of the resource budget model https://nph.onlinelibrary.wiley.com/doi/abs/10.1111/nph.14114)

1)Create another column in the dataset that classifies years as mast and non-mast

```r
mast_val_num<-ADM_vec(current_dataset_hottest_month_dT$Value)
current_dataset_hottest_month_dT<-current_dataset_hottest_month_dT|>
  mutate(mast_val=ifelse(Value>=mast_val_num,"mast","nonmast"))
```

2)compute years since last masting event for each year if first year is mast year, it will be assigned with "NA", because assigning any numerical value would be methodologically incorrect (would it?) if first year is not mast year, it is assigned with 0. For consecutive non-mast years assigned value increases by one up until the mast year. Mast year is assigned with the largest number in the sequence, and year right after mast year is assigned with 0, counting begins again

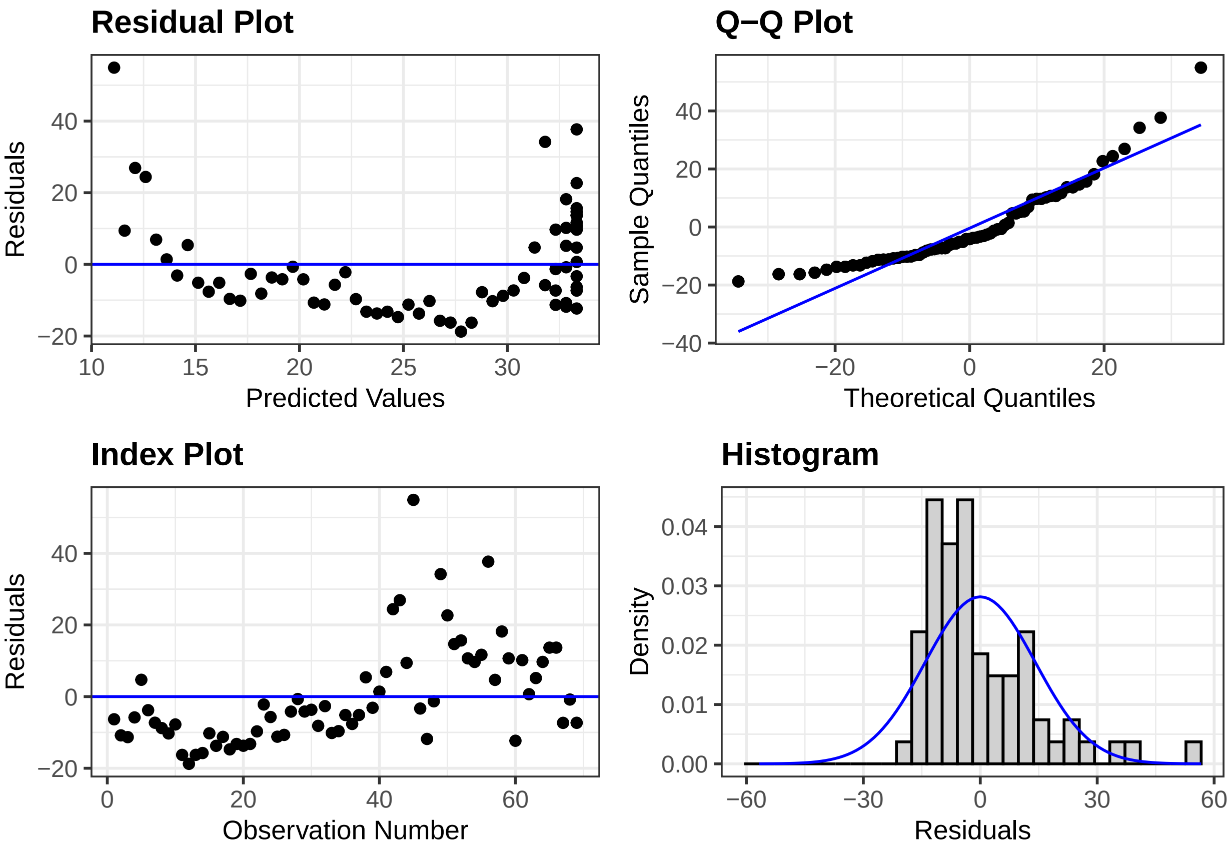
```r
count=0 #counts years since last mast event
period=c() #stores count values

#go through each row
for (i in 1:nrow(current_dataset_hottest_month_dT)){

  #if first year is mast year, it will be assigned with "NA", because assigning any numerical value wou
  if (current_dataset_hottest_month_dT$mast_val[i]=="mast" & length(period)==0) {

period=c(period, NA)
count=0

#Mast year is assigned with the largest number in the sequence
} else if ( current_dataset_hottest_month_dT$mast_val[i]=="mast") {
  period=c(period, count)
  count=0# and year right after mast year is assigned with 0

  #For consecutive non-mast years assigned value increases by one up until the mast year.
} else {
  period=c(period, count)
  count=count+1
}

}

current_dataset_hottest_month_dT$period<-period
```

3) assess the relationsip

```
#make linear regression model
lm_period<-lm(Value~period, data= current_dataset_hottest_month_dT)
#obseve your model
summary(lm_period)
```

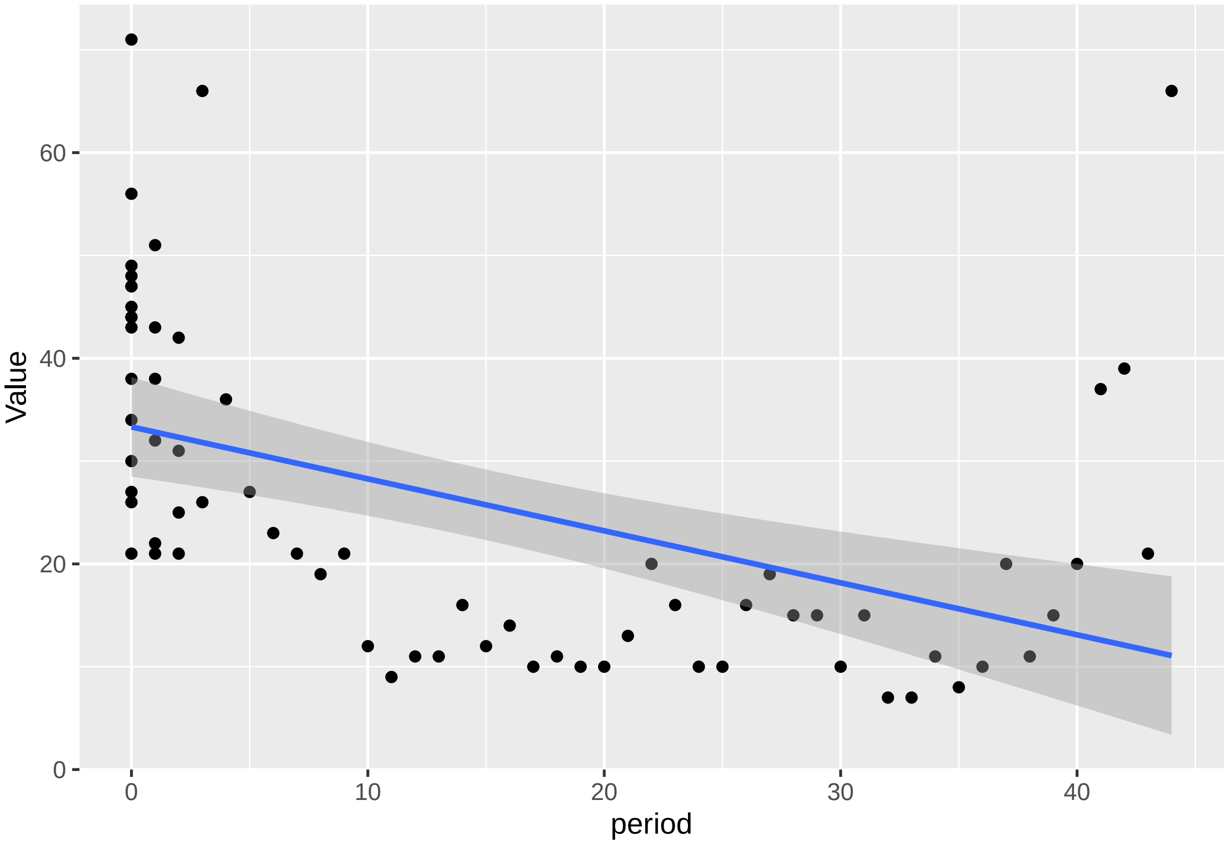
```
##
## Call:
## lm(formula = Value ~ period, data = current_dataset_hottest_month_dT)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.766 -10.249  -4.172   9.410  54.916
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  33.3264     2.4230  13.754  < 2e-16 ***
## period       -0.5055     0.1174  -4.306 5.55e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.28 on 67 degrees of freedom
## Multiple R-squared:  0.2168, Adjusted R-squared:  0.2051
## F-statistic: 18.54 on 1 and 67 DF,  p-value: 5.551e-05
```

```
#assess residuals
resid_panel(lm_period)
```



```
#scatterplot of  seed production vs years since last masting event
ggplot(current_dataset_hottest_month_dT, mapping= aes(x=period, y=Value))+
  geom_point() +
  geom_smooth(method="lm")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



IV) At some point we decided to conduct multivariable analysis (multiple linear regression) of climatic variables. Here we consider temperature in a certain month and precipitation at a certain month as separate variables (such approach is questionable because climate is autocorrelated and stuff… There are ways to assess autocorrelation: https://cran.r-project.org/web/packages/lme4/vignettes/lmer.pdf but we did not go that far[and, tbh, our statistical knowledge was not sufficient as well])

Start by preparing data Our main data has questionable format: we have 12 rows (one for each month) for each year that are identical in everything but climatic variables (we have a column "month", and each value in climate column corresponds to the month from the same row). Someone might say that better format would be to widen our data so it is 12 times shorter (only one row per year), but 12 times wider (climatic variable in each month is its own column). For the multivariable analysis, we need our data to have the latter format. Thus, we widen it.

```
# Extract columns 1 to 39 (excluding column 40) from the 'current_dataset' data frame.
# Then, pivot the data wider, with the column values of 'temp_2m' spread across multiple columns based
# The new columns will be named with the prefix "t" followed by the corresponding month number.
current_dataset_main_and_temp <- current_dataset[, c(1:39)] |>
  pivot_wider(
    names_from = c(month),
    values_from = c(temp_2m),
    names_prefix = "t"
  )

# Extract columns 1 to 38 and column 40 from the 'current_dataset' data frame.
# Then, pivot the data wider, with the column values of 'tot_precip' spread across multiple columns bas
# The new columns will be named with the prefix "p" followed by the corresponding month number.
current_dataset_main_and_precip <- current_dataset[, c(1:38, 40)] |>
  pivot_wider(
    names_from = c(month),
    values_from = c(tot_precip),
    names_prefix = "p"
  )

# Extract columns 38 to 49 (these contain the monthly precipitation values) from 'current_dataset_main_
current_dataset_precip <- current_dataset_main_and_precip[, c(38:49)]

# Combine the 'current_dataset_main_and_temp' data frame (containing temperature values) and the 'curre
current_dataset_main_temp_precip <- cbind(current_dataset_main_and_temp, current_dataset_precip)
```

Multivariable analysis with explanatory variables being temperatures in particular months

```
model <- lm(Value ~ t1 + t2 + t3+t4 + t5 + t6+t7 + t8 + t9+t10 + t11 + t12, data = current_dataset_main_
summary(model)
```

```
##
## Call:
## lm(formula = Value ~ t1 + t2 + t3 + t4 + t5 + t6 + t7 + t8 +
##     t9 + t10 + t11 + t12, data = current_dataset_main_temp_precip)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -19.294  -9.758  -2.756   8.395  36.354
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -107.1379    34.2966  -3.124  0.00283 **
## t1             0.7652     0.7198   1.063  0.29230
## t2             0.1753     0.5729   0.306  0.76077
## t3            -0.2791     0.7170  -0.389  0.69861
## t4            -0.4610     1.2514  -0.368  0.71395
## t5            -0.3275     1.2644  -0.259  0.79659
## t6             1.2649     1.4158   0.893  0.37548
## t7             4.2979     1.5109   2.845  0.00620 **
## t8             2.7021     1.5965   1.693  0.09609 .
## t9            -1.1162     1.3145  -0.849  0.39941
## t10            2.0973     1.1898   1.763  0.08339 .
## t11           -0.5304     0.9050  -0.586  0.56019
## t12            0.7845     0.7538   1.041  0.30252
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.22 on 56 degrees of freedom
## Multiple R-squared:  0.3504, Adjusted R-squared:  0.2112
## F-statistic: 2.518 on 12 and 56 DF,  p-value: 0.01006
```

Multivariable analysis with explanatory variables being precipitations in particular months

```
model <- lm(Value ~ p1 + p2 + p3+p4 + p5 + p6+p7 + p8 + p9+p10 + p11 + p12, data = current_dataset_main_
summary(model)
```

```
##
## Call:
## lm(formula = Value ~ p1 + p2 + p3 + p4 + p5 + p6 + p7 + p8 +
##     p9 + p10 + p11 + p12, data = current_dataset_main_temp_precip)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -25.596 -10.328  -0.465   9.780  42.405
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    28.27      16.45   1.719   0.0912 .
## p1             49.28    2986.97   0.016   0.9869
## p2           4792.89    2911.32   1.646   0.1053
## p3           2374.29    3029.46   0.784   0.4365
## p4           4000.41    2698.19   1.483   0.1438
## p5           -129.36    1850.08  -0.070   0.9445
## p6          -1142.37    1617.19  -0.706   0.4829
## p7           -526.56    1330.66  -0.396   0.6938
## p8          -1933.22    1924.95  -1.004   0.3196
## p9            -13.30    1712.71  -0.008   0.9938
## p10          1624.23    1778.79   0.913   0.3651
## p11         -2979.53    3107.53  -0.959   0.3418
## p12         -5139.03    2531.95  -2.030   0.0472 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.71 on 56 degrees of freedom
## Multiple R-squared:  0.2071, Adjusted R-squared:  0.03725
## F-statistic: 1.219 on 12 and 56 DF,  p-value: 0.2935
```

Multivariable analysis with explanatory variables being temperatures and precipitations in particular months (assume that in a particular year seed production is defined by climate and precipitation separately )

```
model <- lm(Value ~t1 + t2 + t3+t4 + t5 + t6+t7 + t8 + t9+t10 + t11 + t12+ p1 + p2 + p3+p4 + p5 + p6+p7
summary(model)
```

```
##
## Call:
## lm(formula = Value ~ t1 + t2 + t3 + t4 + t5 + t6 + t7 + t8 +
##     t9 + t10 + t11 + t12 + p1 + p2 + p3 + p4 + p5 + p6 + p7 +
##     p8 + p9 + p10 + p11 + p12, data = current_dataset_main_temp_precip)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -22.191  -7.606  -1.202   7.064  27.089
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -135.3349    44.5551  -3.037  0.00400 **
## t1              0.2416     0.7975   0.303  0.76334
## t2             -0.2443     0.6178  -0.395  0.69448
## t3             -0.1952     0.8229  -0.237  0.81364
## t4             -1.0690     1.3263  -0.806  0.42456
## t5             -0.9603     1.4861  -0.646  0.52148
## t6              1.8212     1.5645   1.164  0.25067
## t7              4.9636     1.5851   3.131  0.00309 **
## t8              3.3701     1.9530   1.726  0.09143 .
## t9             -1.0324     1.6163  -0.639  0.52630
## t10             2.0060     1.2372   1.621  0.11207
## t11            -1.0267     1.0512  -0.977  0.33408
## t12             1.3305     0.8189   1.625  0.11134
## p1          -1487.3069  2702.0990  -0.550  0.58481
## p2           4145.2630  2777.9505   1.492  0.14278
## p3           3125.3634  3133.4743   0.997  0.32402
## p4           4528.5253  2554.7916   1.773  0.08322 .
## p5           -333.8517  1941.4179  -0.172  0.86426
## p6            394.7568  1667.7844   0.237  0.81399
## p7           1178.6593  1299.9170   0.907  0.36949
## p8            324.8193  2099.9134   0.155  0.87778
## p9            250.5454  1910.7678   0.131  0.89628
## p10          1533.3650  1708.8828   0.897  0.37445
## p11         -5550.7739  3486.9911  -1.592  0.11858
## p12         -5967.9672  2411.3691  -2.475  0.01725 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.59 on 44 degrees of freedom
## Multiple R-squared:  0.5341, Adjusted R-squared:   0.28
## F-statistic: 2.102 on 24 and 44 DF,  p-value: 0.01605
```

Multivariable analysis with explanatory variables being temperatures and precipitations in particular months (assume that in a particular year seed production is defined by combination of climate and precipitation in a particular month)

```
model <- lm(Value ~ (t1 * p1) + (t2*p2) + (t3 * p3)+(t4 * p4) + (t5*p5) + (t6 * p6)+( t7 * p7) +( t8*p8
summary(model)
```

```
##
## Call:
## lm(formula = Value ~ (t1 * p1) + (t2 * p2) + (t3 * p3) + (t4 *
##     p4) + (t5 * p5) + (t6 * p6) + (t7 * p7) + (t8 * p8) + (t9 *
##     p9) + (t10 * p10) + (t11 * p11) + (t12 * p12), data = current_dataset_main_temp_precip)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.196  -5.739   0.943   4.866  18.468
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.061e+02  1.504e+02  -1.370 0.180131
## t1          -3.068e-01  1.845e+00  -0.166 0.869010
## p1          -3.195e+03  4.578e+03  -0.698 0.490350
## t2          -2.736e+00  1.571e+00  -1.742 0.091169 .
## p2           7.047e+03  3.174e+03   2.220 0.033600 *
## t3          -4.219e-01  2.595e+00  -0.163 0.871902
## p3           2.804e+03  3.688e+03   0.760 0.452594
## t4          -5.893e+00  3.243e+00  -1.817 0.078545 .
## p4          -1.300e+04  1.048e+04  -1.240 0.223908
## t5          -9.012e-01  4.211e+00  -0.214 0.831910
## p5           2.666e+03  1.764e+04   0.151 0.880778
## t6          -2.074e+00  3.557e+00  -0.583 0.563938
## p6          -1.893e+04  1.709e+04  -1.108 0.276080
## t7           4.894e+00  3.872e+00   1.264 0.215408
## p7          -2.933e+02  1.856e+04  -0.016 0.987490    
## t8           5.859e+00  3.506e+00   1.671 0.104443    
## p8           1.918e+04  2.364e+04   0.811 0.423213    
## t9           2.538e+00  2.901e+00   0.875 0.388160    
## p9           1.446e+04  1.468e+04   0.985 0.332024    
## t10          9.321e+00  2.451e+00   3.802 0.000608 ***
## p10          3.194e+04  9.760e+03   3.273 0.002558 ** 
## t11          4.849e-01  2.489e+00   0.195 0.846752    
## p11         -3.075e+03  5.090e+03  -0.604 0.550098    
## t12          1.401e+00  2.822e+00   0.496 0.623037    
## p12         -4.833e+03  2.845e+03  -1.699 0.099045 .  
## t1:p1        5.269e+02  9.433e+02   0.559 0.580337    
## t2:p2        1.370e+03  8.088e+02   1.694 0.099886 .  
## t3:p3       -2.633e+02  1.191e+03  -0.221 0.826443    
## t4:p4        2.064e+03  1.302e+03   1.585 0.122784    
## t5:p5       -1.702e+02  1.373e+03  -0.124 0.902144    
## t6:p6        1.288e+03  1.083e+03   1.189 0.243280    
## t7:p7        1.204e+02  1.035e+03   0.116 0.908179    
## t8:p8       -9.802e+02  1.337e+03  -0.733 0.468965    
## t9:p9       -9.705e+02  1.175e+03  -0.826 0.414928    
## t10:p10     -3.941e+03  1.210e+03  -3.257 0.002668 ** 
## t11:p11     -5.335e+02  1.262e+03  -0.423 0.675179    
## t12:p12      1.511e+02  1.474e+03   0.102 0.919012    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 12.25 on 32 degrees of freedom
## Multiple R-squared:  0.7245, Adjusted R-squared:  0.4145 
## F-statistic: 2.337 on 36 and 32 DF,  p-value: 0.008379
```