Exercise 5: Feature extraction for tree segments

1) Open and examine the given segmentation (TrainingTrees.shp) in ArcMap

2) Read the Lidar point data to RStudio (script provided)

3) Read the shapefile to RStudio (script provided)

4) Use spatial overlay analysis to detect which Lidar points are located inside the tree segments (script provided)

subset >2 and <50 because no tree above 50 and should be above 2 to avoid small grases

5) Create a code which calculates the following Lidar features for individual tree segments. Use only first and single echoes while calculating the features:

□ Tree position

o X-coordinate of maximum height observation (X)

o Y-coordinate of maximum height observation (Y)

□ Maximum height of the observations (Hmax)

□ Mean height of the observations (Hmean)

□ Standard deviation of the height observations (Hstd)

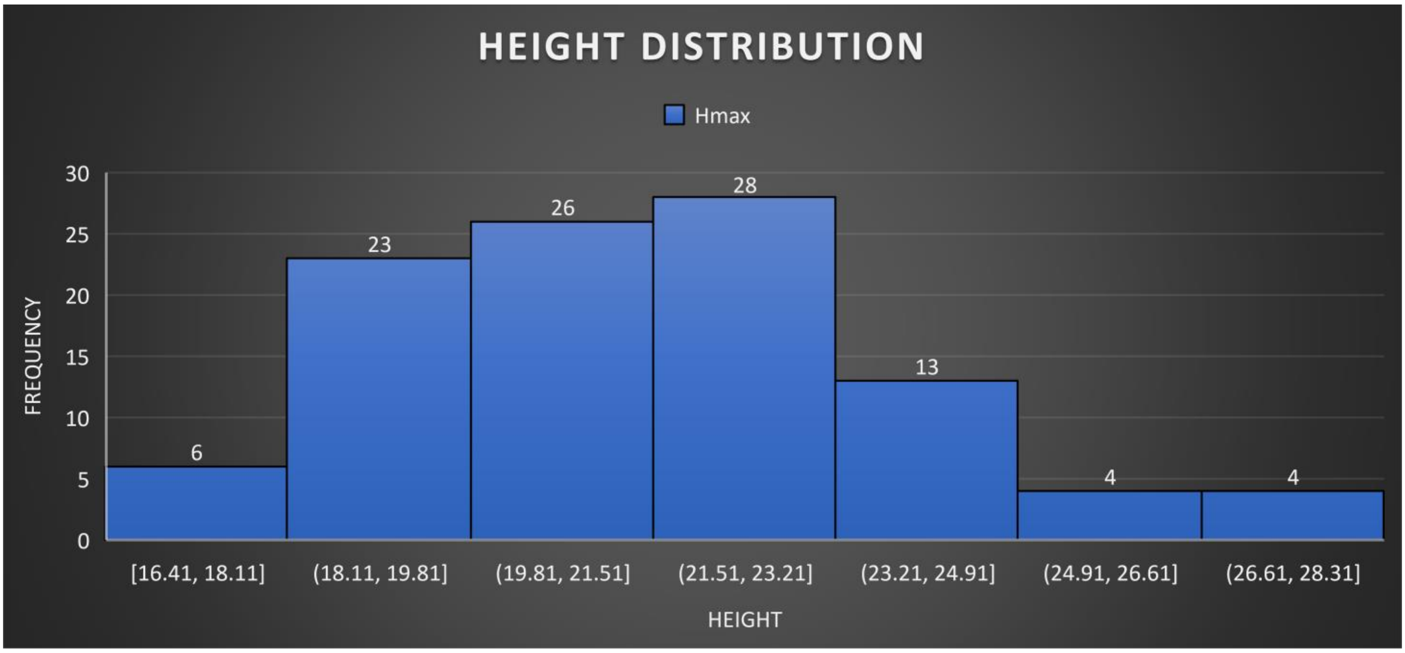
□ Coefficient of variation (CV)

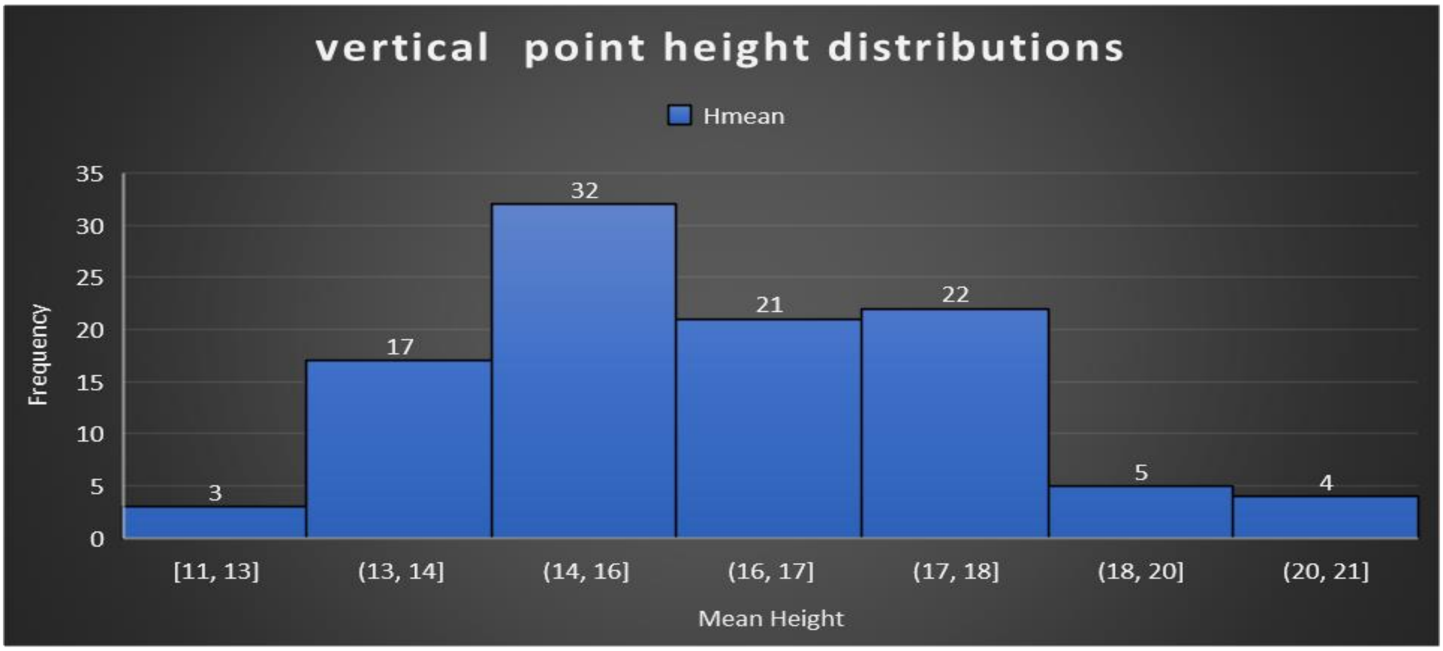
□ Vegetation density, i.e. percentage of observations above 2 meters (VD)

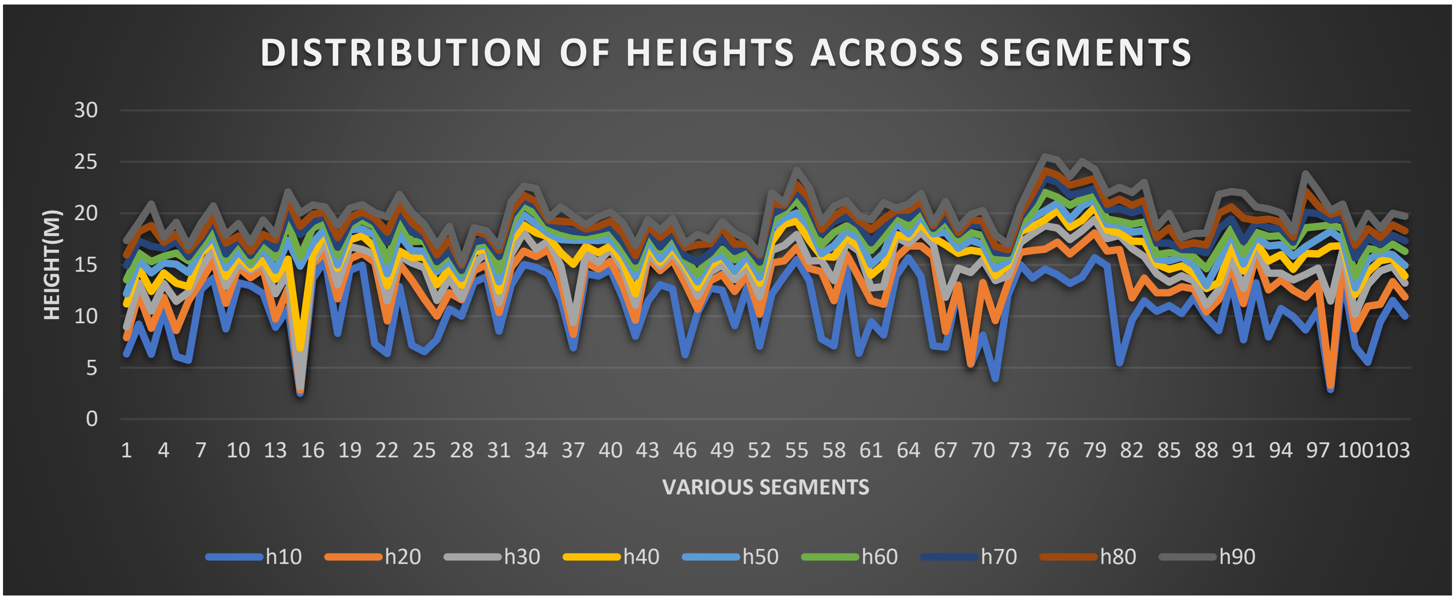
□ Quantiles of the height observations (h10-h90)

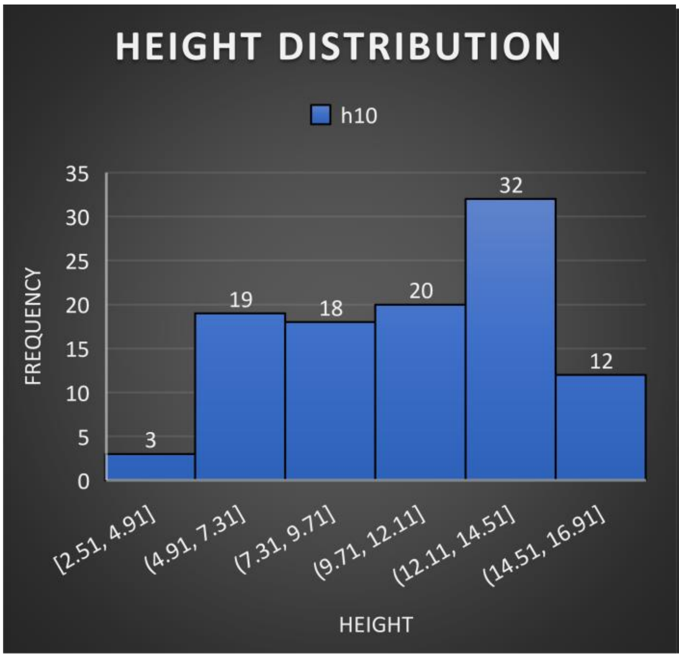
Hint: When you test the code, use only one tree segment. When the code works well, create a loop to calculate the features for all the tree segments.

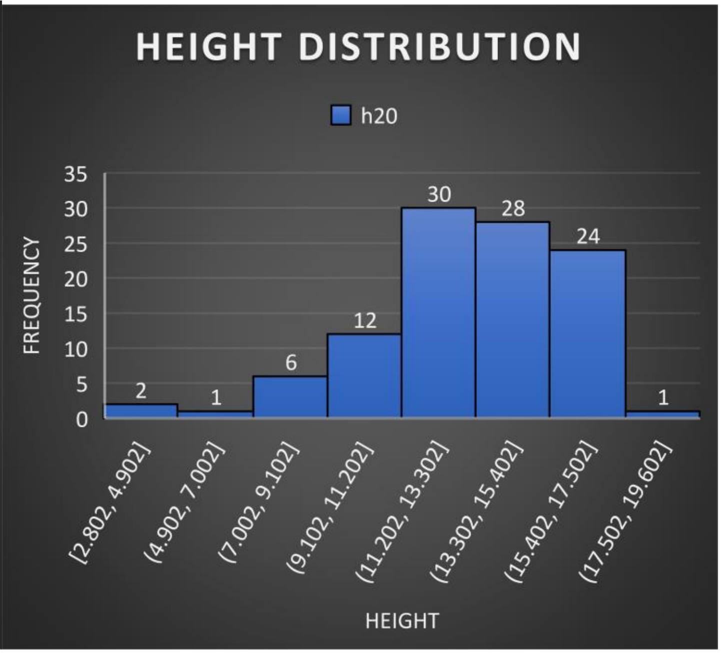
6) Export your features as a csv-file. Draw vertical point height distributions in Excel.

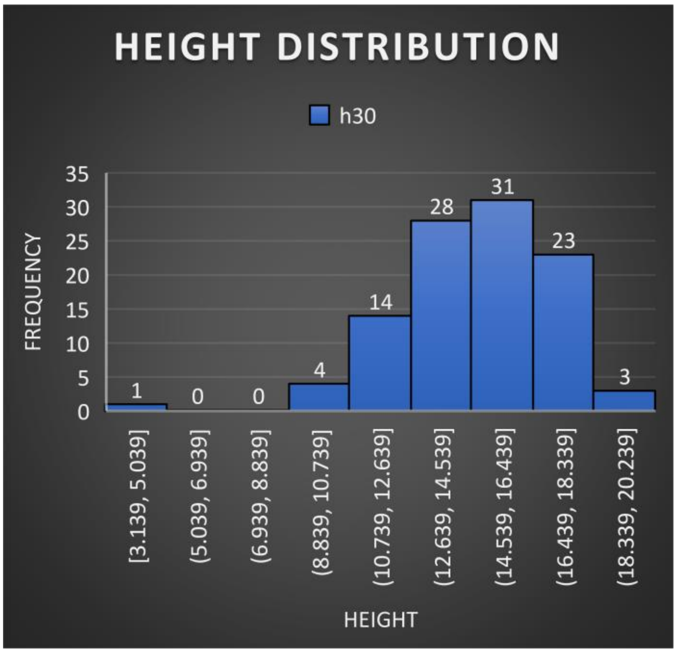


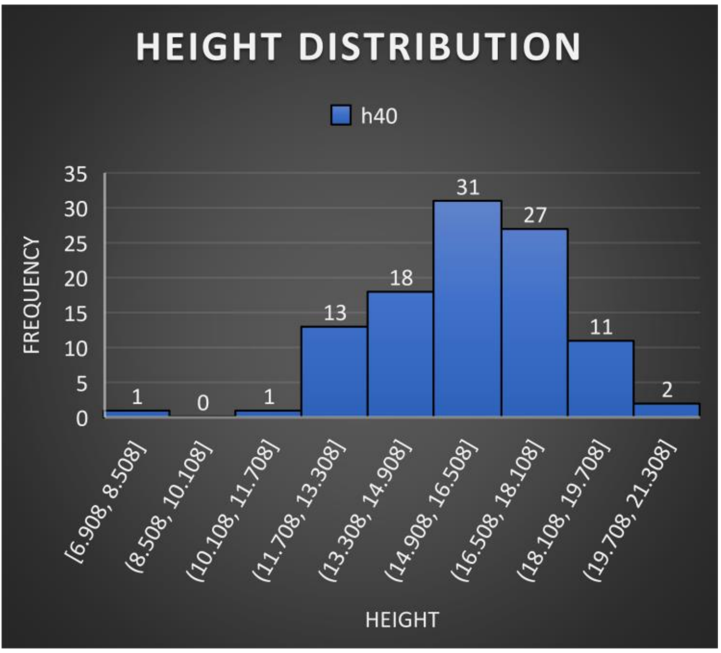

vertical point height distributions
Hmean
Frequency
35
30
25
20
15
10
5
0
3
17
32
21
22
5
4
[11, 13]
(13, 14]
(14, 16]
(16, 17]
(17, 18]
(18, 20]
(20, 21]
Mean Height

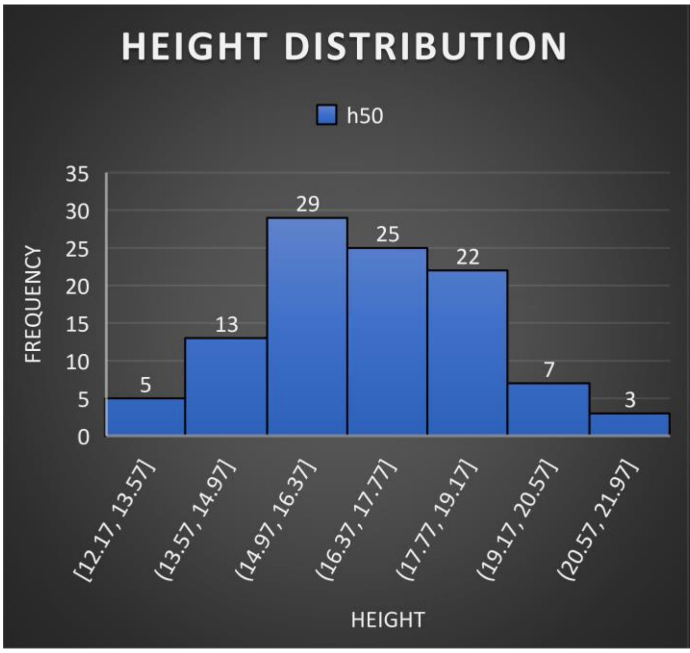

DISTRIBUTION OF HEIGHTS ACROSS SEGMENTS
HEIGHT(M)
30
25
20
15
10
5
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97 100103
VARIOUS SEGMENTS
h10
h20
h30
h40
h50
h60
h70
h80
h90

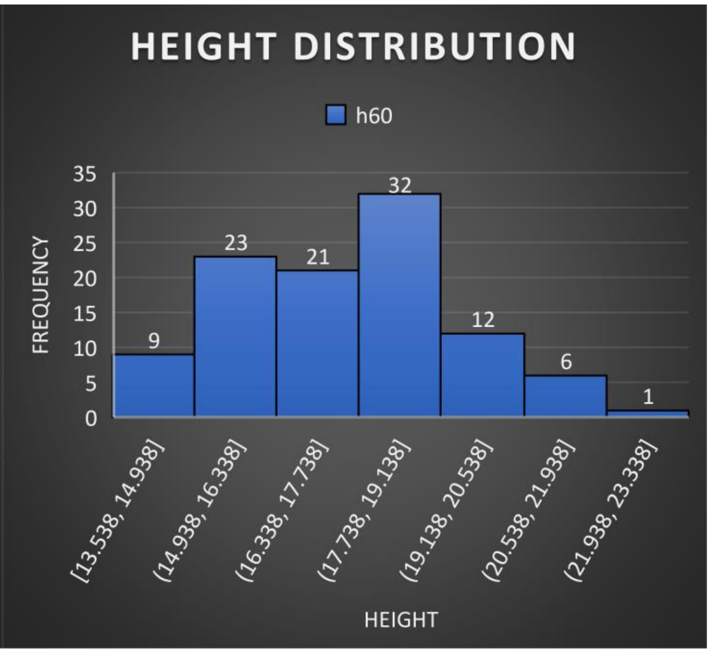

HEIGHT DISTRIBUTION
h10
FREQUENCY
35
30
25
20
15
10
5
0
3
19
18
20
32
12
[2.51, 4.91]
(4.91, 7.31]
(7.31, 9.71]
(9.71, 12.11]
(12.11, 14.51]
(14.51, 16.91]
HEIGHT

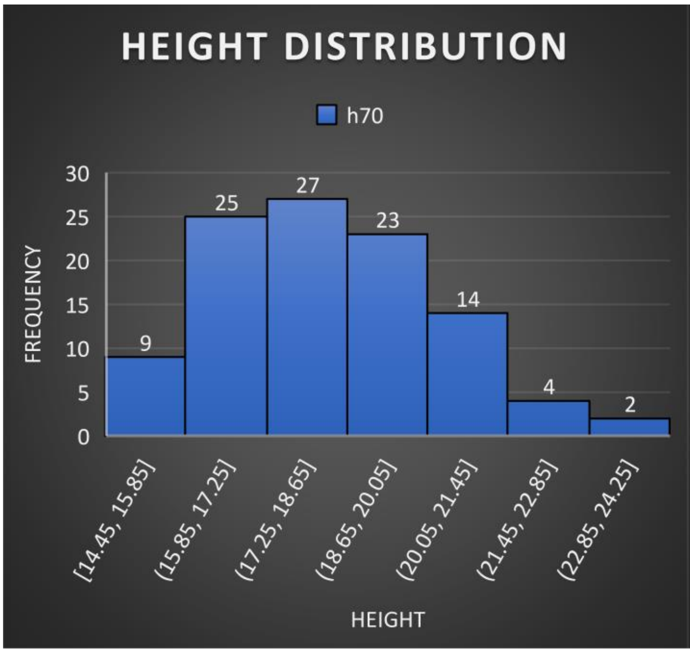

HEIGHT DISTRIBUTION
h20
FREQUENCY
35
30
25
20
15
10
5
0
2
1
6
12
30
28
24
1
[2.802, 4.902]
(4.902, 7.002]
(7.002, 9.102]
(9.102, 11.202]
(11.202, 13.302]
(13.302, 15.402]
(15.402, 17.502]
(17.502, 19.602]
HEIGHT

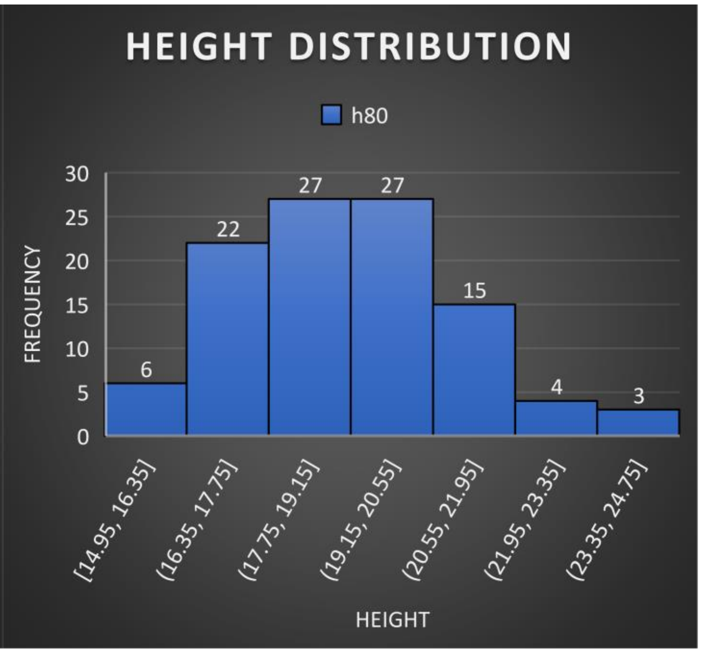

HEIGHT DISTRIBUTION
h30
FREQUENCY
35
30
25
20
15
10
5
0
1
0
0
4
14
28
31
23
3
[3.139, 5.039]
(5.039, 6.939]
(6.939, 8.839]
(8.839, 10.739]
(10.739, 12.639]
(12.639, 14.539]
(14.539, 16.439]
(16.439, 18.339]
(18.339, 20.239]
HEIGHT

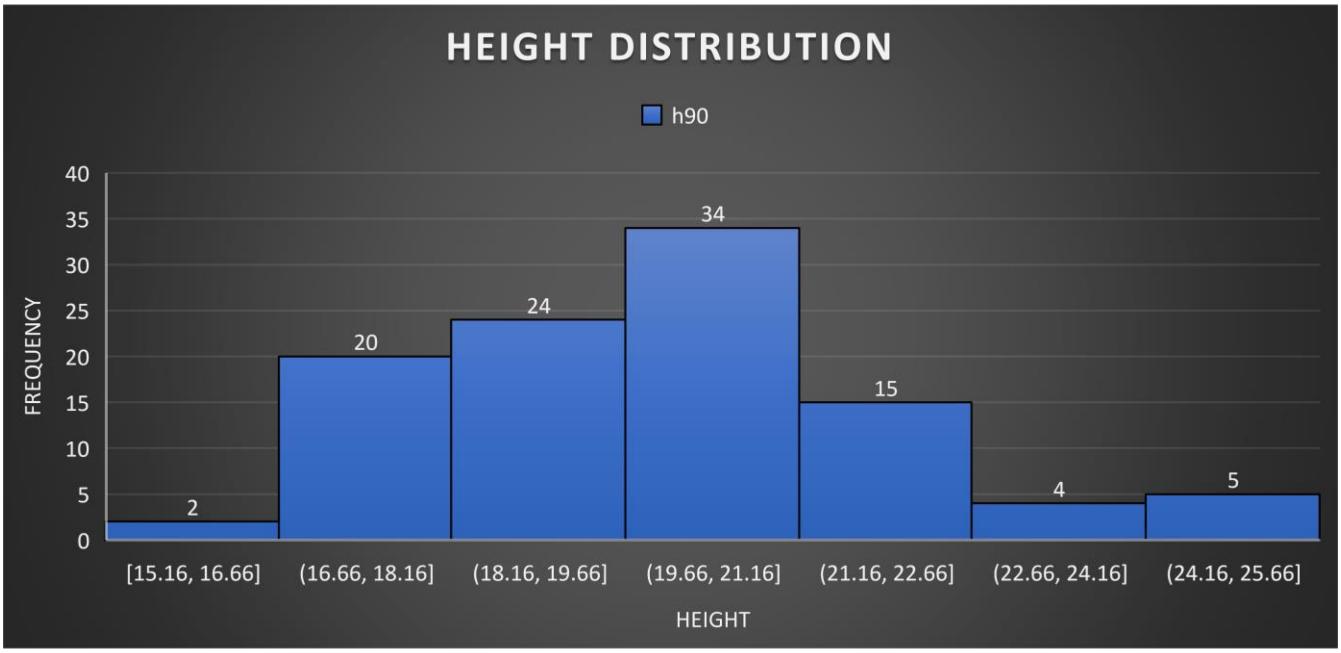

HEIGHT DISTRIBUTION
h40
FREQUENCY
35
30
25
20
15
10
5
0
1
0
1
13
18
31
27
11
2
[6.908, 8.508]
(8.508, 10.108]
(10.108, 11.708]
(11.708, 13.308]
(13.308, 14.908]
(14.908, 16.508]
(16.508, 18.108]
(18.108, 19.708]
(19.708, 21.308]
HEIGHT

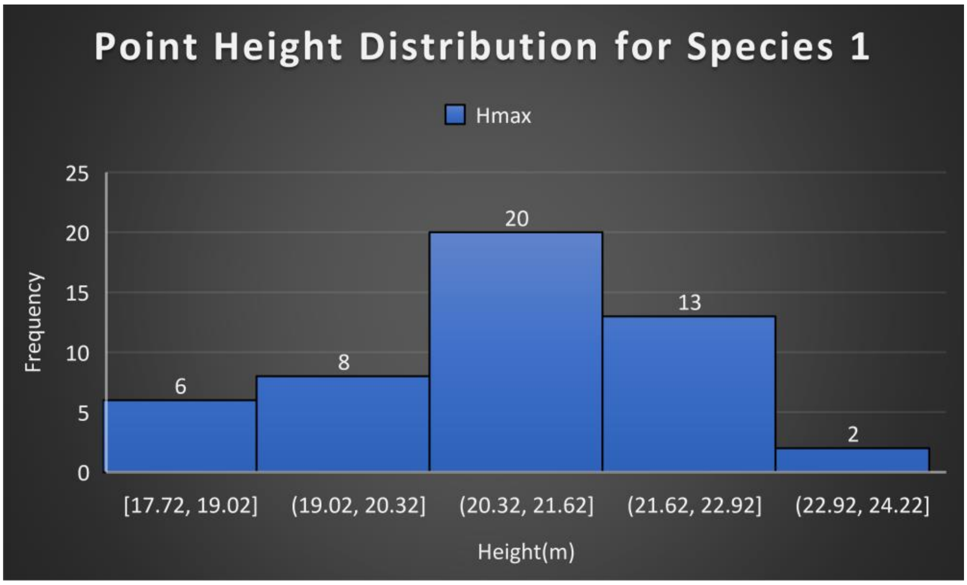

HEIGHT DISTRIBUTION
h50
FREQUENCY
35
30
25
20
15
10
5
0
5
13
29
25
22
7
3
[12.17, 13.57]
(13.57, 14.97]
(14.97, 16.37]
(16.37, 17.77]
(17.77, 19.17]
(19.17, 20.57]
(20.57, 21.97]
HEIGHT

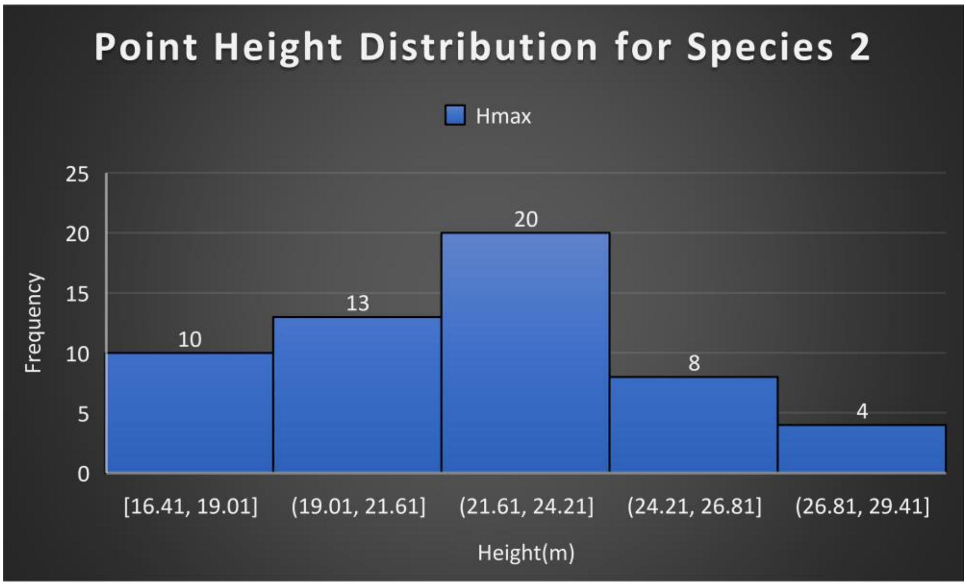

HEIGHT DISTRIBUTION
h60
FREQUENCY
35
30
25
20
15
10
5
0
9
23
21
32
12
6
1
[13.538, 14.938]
(14.938, 16.338]
(16.338, 17.738]
(17.738, 19.138]
(19.138, 20.538]
(20.538, 21.938]
(21.938, 23.338]
HEIGHT


HEIGHT DISTRIBUTION
h70
FREQUENCY
30
25
20
15
10
5
0
9
25
27
23
14
4
2
[14.45, 15.85]
(15.85, 17.25]
(17.25, 18.65]
(18.65, 20.05]
(20.05, 21.45]
(21.45, 22.85]
(22.85, 24.25]
HEIGHT

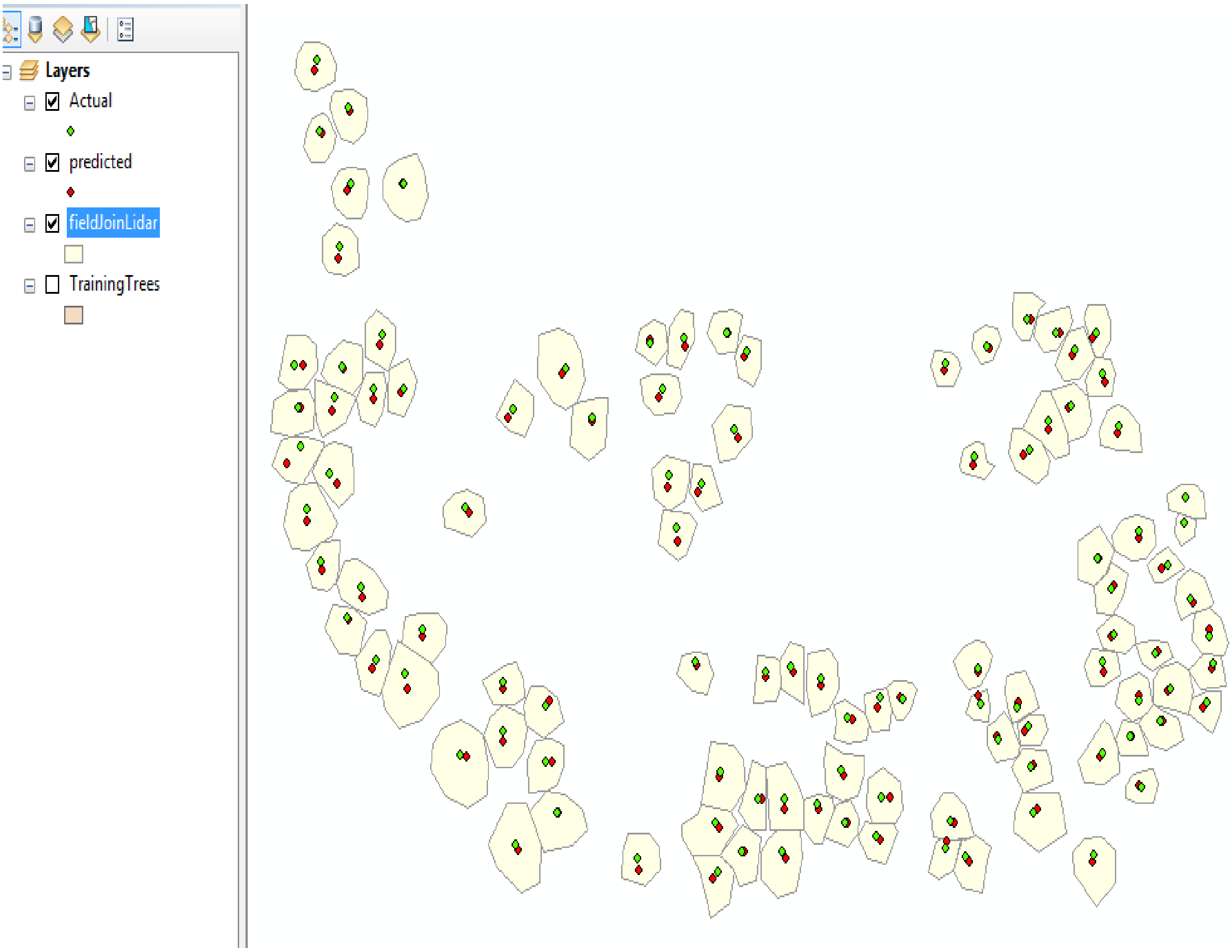

HEIGHT DISTRIBUTION
h80
FREQUENCY
30
25
20
15
10
5
0
6
22
27
27
15
4
3
[14.95, 16.35]
(16.35, 17.75]
(17.75, 19.15]
(19.15, 20.55]
(20.55, 21.95]
(21.95, 23.35]
(23.35, 24.75]
HEIGHT


HEIGHT DISTRIBUTION
h90
FREQUENCY
0
5
10
15
20
25
30
35
40
2
20
24
34
15
4
5
[15.16, 16.66]
(16.66, 18.16]
(18.16, 19.66]
(19.66, 21.16]
(21.16, 22.66]
(22.66, 24.16]
(24.16, 25.66]
HEIGHT

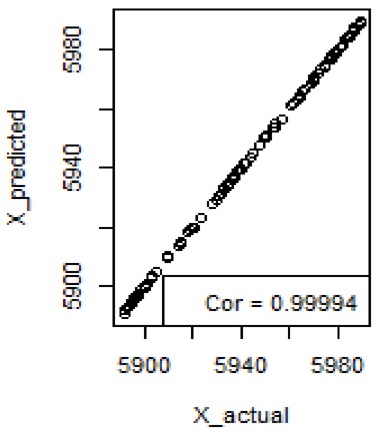

Point Height Distribution for Species 1
Hmax
Frequency
0
5
10
15
20
25
6
8
20
13
2
[17.72, 19.02]
(19.02, 20.32]
(20.32, 21.62]
(21.62, 22.92]
(22.92, 24.22]
Height(m)

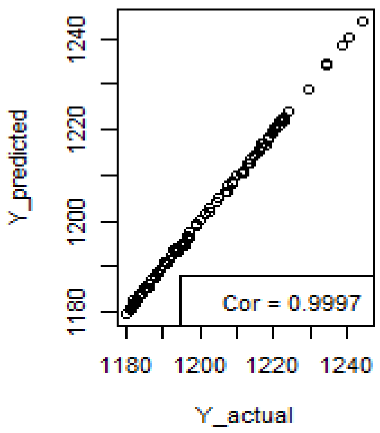

Point Height Distribution for Species 2
Hmax
Frequency
0
5
10
15
20
25
10
13
20
8
4
[16.41, 19.01]
(19.01, 21.61]
(21.61, 24.21]
(24.21, 26.81]
(26.81, 29.41]
Height(m)

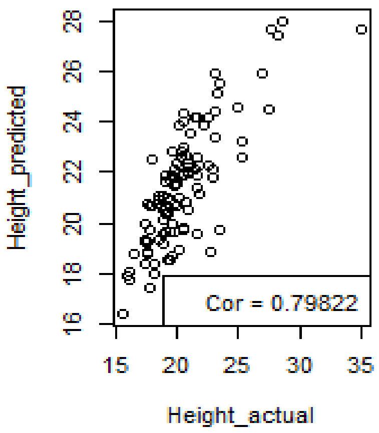
7) Import your trees and calculated features to ArcGIS. Calculate positioning error and error in tree height for all the trees.



As it can be seen in the map above, there is a slight mismatch between the positions.


Predicted X vs Actual X


Predicted Y vs Actual Y


Predicted Height vs Actual Hei

As shown in the scatter diagram above, and also the correlation, the accuracies are quite high but not perfect. The positional accuracies are better than that of the height. Latitude has the highest accuracy, followed by the longitude and the height.

RMSE_X	RMSE_Y	RMSE_H	MAE_X	MAE_Y	MAE_H
0.34281	0.507849	2.008496	0.264904	0.412337	1.656411

Where RMSE is root mean square error;

MAE is mean absolute error.

The table shows the root mean square errors and the mean absolute errors of the various variables. Latitude has the least error, followed by the longitude and the height has the most error, although not so much. This further corroborates the scatter diagram and correlation earlier.

Analyse, are there some differences between different tree species?

ID_Sp	rmse_X	rmse_Y	rmse_H	mae_X	mae_Y	mae_H
1	0.265166	0.453207	1.80725	0.219388	0.376898	1.551655
2	0.399468	0.551993	2.172139	0.305455	0.443909	1.749739

In the above table, it is shown that species 2 has more error than species 1. In other words, the positional accuracies and the height accuracies of species 1 are higher than those of species 2.

Return a pdf-file including:

□ The modified R script

```
#by: Oyedayo Oyelowo

#student number: 014717208

#University of Helsinki


##########################################

# Feature extraction for tree segments #

##########################################


rm(list=ls())

setwd("C:/Users/oyeda/Desktop/ADV_REM_SENS/assignment4")

.libPaths("C:/Users/oyeda/Desktop/ADV_REM_SENS/library")


######################################################################################


# Read 3D point data (TEXAS_lidar.txt)


Lidar <- read.table(file="TEXAS_lidar.txt", head=FALSE, sep="\t")
```

```
colnames(Lidar) <- c("Year","Pulse","X","Y","Z","h","range","int","agc")

summary(Lidar)
library(sp)
library(maptools)
library(rgdal)

# Specify column names of the coordinates
coordinates(Lidar) = c("X", "Y")
####################################################################################

# Read the shapefile (TrainingTrees.shp)
#install.packages("sp")
#install.packages("maptools")
S <- readShapePoly("TrainingTrees.shp") #Tree segments
SP <- as(S, "SpatialPolygons")
#?readShapePoly

# Plot tree segments

plot(SP, col="red")

####################################################################################

# Analyse, which Lidar points are located inside the segments

Lidar$In <- over(Lidar,SP)



# Select the 2010 Lidar data, and remove the points located outside of the segments

Lidar10 <- subset(Lidar, Lidar$Year==10 & Lidar$In!="NA")
Lidar10<-data.frame(Lidar10)

tail(Lidar10)
```

```
summary (Lidar10)

######################################################################################

####################################################################
### TEE ALLA OLEVAAN KOODIIN TARVITTAVAT LISÄYKSET JA MUOKKAUKSET ###
####################################################################
#5) Create a code which calculates the following Lidar features for individual tree
#segments. Use only first and single echoes while calculating the features:
#  . Tree position

#use first or single only which are have height more than 2m but and less than 50m
#these are the defined thresholds for the trees. Because the trees are assumed to be not
#not more than 50m, and anything more than is taken as error. also, below 2m is assumed
#not to be a tree.



#o X-coordinate of maximum height observation (X)
#o Y-coordinate of maximum height observation (Y)
#. Maximum height of the observations (Hmax)
#. Mean height of the observations (Hmean)
#. Standard deviation of the height observations (Hstd)
#. Coefficient of variation (CV)
#. Vegetation density, i.e. percentage of observations above 2 meters (VD)
#. Quantiles of the height observations (h10-h90)

#I tried two ways.
#create empty objects to be filled later
#instead of using the long method below, i used a more efficient means
#which follows immediately after
#ID <- 0; X <- 0; Y <- 0; Hmax <- 0; Hmean <- 0;
#Hstd <- 0; CV <- 0; VD <- 0;
#h10 <- 0; h20 <- 0; h30 <- 0; h40 <- 0; h50 <- 0;
#h60 <- 0; h70 <- 0; h80 <- 0; h90 <- 0;
```

```
#the second method below:
ID<-X<-Y<-Hmax<-Hmean<-Hstd<-CV<-VD<-h10<-h20<-h30<-h40<-h50<-h60<-
  h70<-h80<-h90<-0

# subset only first and single echoes!
lfisi2 <- subset(Lidar10, (Lidar10$Pulse=="Fi" | Lidar10$Pulse=="Si") )

#subset height greater than 2m and less than 50. I could have executed
#the above and below on a single line but I did them separately
#for a calculation in the loop afterwards, where I needed to calculate
#the percentage of the trees(i.e above 2m), covering the area
lfisi<- subset(lfisi2, lfisi2$h>2 &lfisi2$h<50)



#min(lfisi$In):max(lfisi$In) this can also be used instead of
#sort(unique(lfisi$In)) in this case
for (i in sort(unique(lfisi$In))) {

  #subset only first and single echoes!
  lfisi2 <- subset(Lidar10, (Lidar10$Pulse=="Fi" | Lidar10$Pulse=="Si") )
  lfisi<- subset(lfisi2, lfisi2$h>2 &lfisi2$h<50)
  lfisi<-subset(lfisi, lfisi$In == i)
  ID[i]<- i

  #the below is done for the joining operation in Arcgis for the
  #field data column with numbers running from 0 to 103
  #also note:  you can also use #can also use: ID2<-seq(0, 103)
  #or ID2<- seq(0, i-1) but when using the sequence, avoid the vector [i]
  #as it will generate 0 throughout the column instead of
  #sequencing from 0 to 103. This is because it returns
  #NB: I decided not to use this anymore because I joined
  #the field data and lidar data directly here
  #ID2[i]<- i-1
  X[i] <- lfisi[which.max(lfisi[,"h"]),"X"] #Returns "X" from the point that has the maximum "h" value
  Y[i] <- lfisi[which.max(lfisi[,"h"]),"Y"]   #Returns "Y" from the point that has the maximum "h" value
  Hmax[i] <- lfisi[which.max(lfisi[,"h"]),"h"]
  Hmean[i]<-mean(lfisi$h)
  Hstd[i]<- sd(lfisi$h)
  #CV[i]<- (Hstd[i]/ Hmean[i])
  #or
  CV[i]<-sd(lfisi$h)/mean(lfisi$h)

  h10[i]<-quantile(lfisi$h, probs = c(0.1))
  h20[i]<-quantile(lfisi$h, probs = c(0.2))
  h30[i]<-quantile(lfisi$h, probs = c(0.3))
  h40[i]<-quantile(lfisi$h, probs = c(0.4))
  h50[i]<-quantile(lfisi$h, probs = c(0.5))
  h60[i]<-quantile(lfisi$h, probs = c(0.6))
  h70[i]<-quantile(lfisi$h, probs = c(0.7))
  h80[i]<-quantile(lfisi$h, probs = c(0.8))
  h90[i]<-quantile(lfisi$h, probs = c(0.9))



  VD[i]<- (nrow(lfisi)*100/nrow(lfisi2))
  #0r
  #VD[i]<- (length(lfisi$h)*100/length(lfisi2$h))



  output <- cbind.data.frame(ID,X,Y, Hmax, Hmean, Hstd, CV,VD,h10,h20,
                   h30,h40,h50,h60,h70,h80,h90)
}

#I already converted the cbind directly to dataframe above in
#the loop by using cbind.data.frame() instead of just cbind
#output<-data.frame(output) #convert vector to dataframe

#the vertical point height distribution
hist(output$Hmean, main = "vertical point height distributions",
     xlab = "Height", col = "purple")

##################################################################
#THIS PART IS NOT NEEDED ANYMORE. Just keeping for reference
#purpose. I did this, when i had to use arcgis to join the field
#and lidar data and then reimported, the shapefile which had been
#joined with the lidar output.

##field and lidar data have been joined in Arcgis, here, I will
#import the shp file and caculate the errors
#fieldLidar<- readShapePoly("C:/Users/oyeda/Desktop/ADV_REM_SENS/assignment4/fieldJoinLidar")

#this is not really necessary now, as I only need to import
#the shapfile and convert to dataframe to calculate the errors
#only did it to try it out to see how it works. If used, data.frame
#function cannot coerce the spatialPolygons into dataframe
#fieldLidar <- as(S, "SpatialPolygons")
#plot(fieldLidar, col= "blue")  #plot the shapefile
#convert the shapefile to dataframe to allow manipulation and analysis.
#fieldLidarDf <- data.frame(fieldLidar)
#####################################################################

#join the field data imported and defined as S earlier and also the
#output of the lidar analysis. Do this directly into a dataframe
fieldLidarDf<-cbind.data.frame(S,output)

#function that returns the RMSE(root mean square error)
rmse <- function(actual, predicted)
{
  sqrt(mean((actual - predicted)^2))
}

#function that returns the mean absolute error
mae <- function(actual, predicted)
{
  mean(abs(actual - predicted))
}

#It can also be broken into steps but I prefer the first
#because it is more straightforward

# Function that returns Root Mean Squared Error
#rmse <- function(error)
#{
#  sqrt(mean(error^2))
#}

# Function that returns Mean Absolute Error
#mae <- function(error)
#{
#  mean(abs(error))
#}

# Calculate error
#where, error <- actual - predicted

#Here, I will explore the various columns and compare them to calculate
#errors
summary(fieldLidarDf)

#rename the columns to make them different
colnames(fieldLidarDf)[7]<-c("X1")
colnames(fieldLidarDf)[19]<-c("X2")
colnames(fieldLidarDf)[8]<-c("Y1")
colnames(fieldLidarDf)[20]<-c("Y2")

fieldLidarDf2<-fieldLidarDf
#decided not to use this, because, the number seem vague
#instead, I renamed those columns with same name, as above
#RMSE_X<-rmse(fieldLidarDf[,7], fieldLidarDf[,19])
```

```
#couldn't use the below either because it was a case of two columns
#having thesame name
#names(df)[names(df) == 'old.var.name'] <- 'new.var.name
#names(fieldLidarDf)[names(fieldLidarDf) == '"X"'] <- '"X2"'



#calculate the erros of the positions and height
RMSE_X<-rmse(fieldLidarDf2$X1, fieldLidarDf2$X2)
RMSE_Y<-rmse(fieldLidarDf2$Y1, fieldLidarDf2$Y2)
RMSE_H<-rmse(fieldLidarDf2$h, fieldLidarDf2$Hmax)

MAE_X<-mae(fieldLidarDf2$X1, fieldLidarDf2$X2)
MAE_Y<-mae(fieldLidarDf2$Y1, fieldLidarDf2$Y2)
MAE_H<-mae(fieldLidarDf2$h, fieldLidarDf2$Hmax)

errors<-cbind.data.frame(RMSE_X,RMSE_Y,RMSE_H,MAE_X,MAE_Y,MAE_H)

par(mfrow=c(1,3))
#create a matrix plot to compare them
plot(fieldLidarDf2$X1, fieldLidarDf2$X2, xlab="X_actual",
     ylab="X_predicted", main= "Predicted X vs Actual X")
legend(x='bottomright', legend=paste('Cor =',
      round(cor(fieldLidarDf2$X1, fieldLidarDf2$X2),5)))



plot(fieldLidarDf2$Y1, fieldLidarDf2$Y2,xlab="Y_actual",
     ylab="Y_predicted", main= "Predicted Y vs Actual Y")
legend(x='bottomright', legend=paste('Cor =',
round(cor(fieldLidarDf2$Y1, fieldLidarDf2$Y2),5)))

plot(fieldLidarDf2$h, fieldLidarDf2$Hmax, xlab="Height_actual",
   ylab="Height_predicted", main= "Predicted Height vs Actual Height")
legend(x='bottomright', legend=paste('Cor =',
round(cor(fieldLidarDf2$h, fieldLidarDf2$Hmax),5)))
```

```
fieldLidarDf2<-fieldLidarDf
#the below is mean to calculate the errors in both species into a dataframe
ID_Sp<-rmse_X<-rmse_Y<-rmse_H<-mae_X<-mae_Y<-mae_H<-0
#fieldLidarDf2<-fieldLidarDf2
for (i in sort(unique(fieldLidarDf2$Species))){
  #fieldLidarDf2<-fieldLidarDf2
  b<- subset(fieldLidarDf2, fieldLidarDf2$Species==i)
  ID_Sp[i]<-i
  rmse_X[i]=rmse(b$X1, b$X2)
  rmse_Y[i]<-rmse(b$Y1, b$Y2)
  rmse_H[i]<-rmse(b$h, b$Hmax)
  mae_X[i]=mae(b$X1, b$X2)
  mae_Y[i]<-mae(b$Y1, b$Y2)
  mae_H[i]<-mae(b$h, b$Hmax)
  res=cbind.data.frame(ID_Sp,rmse_X,rmse_Y,rmse_H,mae_X, mae_Y, mae_H)
  }



#subset species 1 and 2
sp1<- subset(fieldLidarDf2, fieldLidarDf2$Species==1)
sp2<- subset(fieldLidarDf2, fieldLidarDf2$Species==2)

write.table(output, file = "ouput.csv", dec=".", sep=",",row.names=F)
write.table(output, file="ouput.txt", dec=".", sep="\t",row.names=F)
write.table(errors, file = "errors.csv", dec=".", sep=",",row.names=F)
write.table(res, file = "res.csv", dec=".", sep=",",row.names=F)
```

□ Point height distributions of different tree species

□ Conclusions of your error calculations

□ Positioning error map