```
#-------------------------------------------------------------------------------#
##########---------------------- Title ----------------------------------##########
# R-script for Quality Assurance and Quality Checking of Bangladesh Forest Inventory Data
#-------------------------------------------------------------------------------#


##########---------------------- Contributors ----------------------------##########
# 1. Md. Akhter Hossain, Food and Agriculture Organization of United Nations and
#    Institute of Forestry and Environmental Sciences, University of Chittagong, Bangladesh
# 2. Oswaldo Carrillo, Food and Agriculture Organization of United Nations
# 3. Gael Sola, Food and Agriculture Organization of United Nations
# 4. Laurent Saint-Andre, Food and Agriculture Organization of United Nations
# 5. Luca Birigazzi, Food and Agriculture Organization of United Nations
#-------------------------------------------------------------------------------#

##                          April 2019                          ##


# Introduction
# -------------------------------------------------------------------------------#
# The script is intended to check the Bangladesh Forest Inventory (BFI) raw and processed data
```

in order to assurance and checking the quality of BFI data. The whole script is composed of

a list of quality checks, importing the data, codes for preparing basic data frames,

checking the data quality on different quality parameter and finally exporting the outputs.

In the codes of data quality control, firstly fild data of biophysical survey of BFI was checked,

then the estimates for the BFI data were checked for outliers or extreme values and then soil and litter

data was checked for different data quality parameters. The inputs used here are the BFI raw/cleaned field

data, soil and litter data from the laboratory and some secondary data (i.e. wood density and

species names) as mentioned in the data import section. The users can navigate the desired checks

easily by using the document outline (section). To ensure the checks work well it is necessry to install the

necessary packages and run the basic codes first. At the end of each check the outputs were exported to

a folder named "QAQC_outputs" inside the directory. At the end of this document the quality check outputs

were exported in group by category as well as alltogether. The users need to create a folder named "QAQC_outputs"

inside the working directory because the codes is prepared to save all the outputs to that folder.

##########---------- List of checks ----------##########

Check with field data----------#

CP 1 (3): Plot, subplot and land feature related checks

CP 1.1 (3.1): Non sampled and inaccessible plots with reasons

CP 1.2 (3.2): LF recorded but missing LF object

(i.e. vegetated, non-vegetated etc.)

CP 1.3 (3.3): Trees were recorded in non-vegetated LF object

CP 1.4 (3.4): Trees present in herb, grass, shrub dominated LF

CP 1.5 (3.5): Strange relationship between leaf cover and crown cover

CP 1.6 (3.6): Presence of Reference Point (RP)

CP 1.7 (3.7): Consistency in distance of reference point

CP 1.8 (3.8): Outlier in the subplot slope by forest zone

(hill>100%, village>15%,Sal forest<20%, coastal and sundarbans >10%)

CP 1.9 (3.10): Inconsistency between LF ownership and legal status

CP 1.10: Mismatching subplot between subplot info and LF subplot proportion

CP 1.11: Missing vegetation management information for "Vegetated" LF object

CP 1.12: Missing LF issue information

CP 1.13: Non-uniform information recorded under LF issue

CP 1.14: CWD data missing (for important variables only)

CP 1.15: FWD data missing (for important variables only)

CP 2 (4): Tree and sapling related checks

CP 2.1 (4.1): NA in tree DBH

CP 2.2 (4.1.1): Saplings having DBH <2 cm

CP 2.3 (4.1.2): Trees having DBH <10 cm

CP 2.4: Outliters (extreme values) in DBH per species

CP 2.5 (4.2): NA in tree/sapling total length

CP 2.6 (4.3): Trees smaller than 1.3m total length

CP 2.7: Extreme values in tree/sapling total length per species

CP 2.8 (4.4): Tree and Saplings measured outside their subplot

CP 2.9 (4.5): Number of trees per plot for which species name is "blank"

CP 2.10 (4.6): Number of trees with unknown species

CP 2.11 (4.7): Outliers in dbh-h relationship

CP 2.12 (4.8): Percentage of tree lengths measured by estimation

CP 2.13 (4.9.1): LF Proportioning completed correctly

CP 2.14 (4.9.2): Tree WO missed in the tree list

CP 2.15 (4.11): Dead standing trees having < 1.3m total length

CP 2.16 (4.14): Trees with no LF information

CP 2.17 (4.15): Consistency between tree density and crown cover

CP 2.18 (4.18): Inconsistency in the distribution of sapling and tree

distribution per team number

CP 3: Standing dead trees related checks

CP 3.1 Outliers (extreme values) in standing dead tree DBH

CP 3.2: Outliers (extreme values) in standing dead tree total length

CP 4: Stumps related checks

CP 4.1 (14.): Outliers (extreme values) in stump (dead and live) height

CP 4.2 (14.): Outliers (extreme values) in stump (dead and live) diameter

CP 5: Bamboo related checks

CP 5.1: Outliers (extreme values) in Bamboo DBH

CP 5.2: Outliers (extreme values) in Bamboo Height

CP 6 (11.1) Consistency in time interval between arrival and departure

Checks with estimated attributes ----------#

CP 7.1: Outliers (extreme values) in tree/ha by plot

CP 7.2 (4.15): Extreme values in AGB/ha by plot for large trees (DBH >=30cm)

CP 7.3 (4.15): Extreme values in AGB/ha by plot for Medium trees (DBH >= 10cm)

CP 7.4 (4.15): Extreme values in AGB/ha by plot for Saplings (10cm > DBH >= 2cm)

CP 7.5 (4.16): Plots of strange values in biomass-leaf cover relationship

CP 7.6: Eextreme values in Large (>=30cm DBH) tree volume/hectare by plot

CP 7.7: Extreme values in Medium (30 >DBH >=10cm) tree volume/ha by plot

CP 7.8: Outliers (extreme values) in standing dead tree volume/ha

CP 7.9 (14.): Outliers (extreme values) in standing dead tree AGB/ha

CP 7.10 (8.1): Outliers (extreme values) in dead matter ton/hectare for CWD

CP 7.11 (8.2): Outliers (extreme values) in dead matter/hectare for FWD

CP 7.12: Strange relationships between Trees/hectare vs Dead Matter

CP 7.13: Outliers (extreme values) in Bamboo volume/hectare

CP 7.14: Outliers (extreme values) in Stump per hectare

CP 7.15: Extreme values in Seedling number/hectare by plot

Checks with soil and litter data ----------#

CP 08: Soil bulk density related checks

CP 8.1: NAs in bulk density database

CP 8.2: Blank cells in bulk density database

CP 8.3: Duplicate values in BD database

CP 8.4: Check the outlier in bulk density data

CP 8.5: Consistency of bulk density data depth with Zone

CP 8.6: Whether the number of bulk density data reported from 3 subplots

CP 8.7: Mismatching lf_id in bulk density and lf database

CP 9 : Soil organic carbon related checks

CP 9.1: NAs in soil organic carbon data

CP 9.2: Identify the blanks in SOC data

CP 9.3: Find the duplicate values in soil organic carbon data

CP 9.4: Outliers (extreme values) in soil organic carbon/ha

CP 9.5: Mismatching lf_id in bulk density and soil carbon

CP 9.6: Relationship of bulk density and soil carbon

CP 9.7: Mismatching lf_id of SOC and LF database

CP 10 : Soil texture

CP 10.1: NAs in soil texture

CP 10.2: Blank in texture data

CP 10.3: Find the duplicate values in soil texture data

CP 10.4: Outliers in percentages of clay, silt and sandy particles

CP 10.5: Mismatching lf_id in Soil Texture and LF database

CP 11: Litter carbon data related checks

CP 11.1: NAs in litter carbon

CP 11.2: Blank cells in litter carbon data

CP 11.3: Find the duplicate values

CP 11.4: Outlier in the litter carbon (%) by LCC

CP 11.5: Mismatching LF ID in Litter Carbon and LF database

CP 11.6: Outliers (extreme values) in litter carbon per hectare

CP 12: Litter dry wt related checks

CP 12.1: Identify the NAs in cells

CP 12.2: Blanks in litter dry weight data

CP 12.3: Duplicate values in Litter Dry Weight data

CP 12.4: Consistency between canopy cover and leaf dry weight

CP 12.5: Mismatched lf_id in litter carbon and litter dry weight data

CP 13: Electric conductivity and pH related checks

CP 13.1: NAs in electricity conductivity and pH data

CP 13.2: Identify the blank cells

CP 13.3: Duplicate values in salinity and pH data

CP 13.4: Missing sample or having sample less than 10g

CP 13.5: Mismatching lf_id in salinity_pH and LF database

#----------#

```
# Check working directory
getwd()

# remove plots
dev.off()

#Erage memory
rm(list=ls())

#install.packages()

library(ggplot2)
library(nlme)



# Define working directory
setwd("C:/Users/User/Google Drive/5_Training Materials/R_Dhaka/Nacessary Materials")



# Import data ----------------------------------------------------------------------
```

```
# BFI field data
tree <- read.csv("./BFI-DB/tree.csv",header=T,stringsAsFactors = FALSE)
names(tree)[1] <- "plot_plot_id"
lf <- read.csv("./BFI-DB/lf.csv", header=T, stringsAsFactors = FALSE)
plot_info <- read.csv("./BFI-DB/plot.csv", header=T, stringsAsFactors = FALSE)
lf_object <- read.csv("./BFI-DB/lf_object.csv", header=T, stringsAsFactors = FALSE)
names(lf_object)[1] <- "plot_plot_id"
lf_issue <- read.csv("./BFI-DB/lf_issues.csv", header=T, stringsAsFactors = F)
subplot_info <- read.csv("./BFI-DB/subplot.csv", header=T, stringsAsFactors = FALSE)
subplot_prop<-read.csv("./BFI-DB/lf_subp_prop.csv", header=T, stringsAsFactors = FALSE)
saplings <- read.csv("./BFI-DB/sapling.csv", header=T, stringsAsFactors = FALSE)
wo <- read.csv("./BFI-DB/wo.csv", header = T, stringsAsFactors = F)
cwd <- read.csv("./BFI-DB/cwd.csv", header=T, stringsAsFactors = F)
fwd <- read.csv("./BFI-DB/fwd.csv", header=T, stringsAsFactors = F)
bamboo <- read.csv("./BFI-DB/bamboo.csv", header=T, stringsAsFactors = FALSE)
seedling <- read.csv("./BFI-DB/seedling.csv", header = T, stringsAsFactors = F)
# BFI soil data
soil_bulk <- read.csv("./BFI_soil_data/bulk_density.csv", header=T, stringsAsFactors = F)
soil_car <- read.csv("./BFI_soil_data/soil_carbon.csv", header=T, stringsAsFactors = F)
lit_dry_wt <- read.csv("./BFI_soil_data/litter_dry_weight.csv", header =T, stringsAsFactors = F)
lit_car <- read.csv("./BFI_soil_data/litter_carbon.csv", header = T, stringsAsFactors = F)
```

```
soil_tex <- read.csv("./BFI_soil_data/soil_texture.csv", header = T, stringsAsFactors = F)
salinity_ph <- read.csv("./BFI_soil_data/ec_and_ph.csv", header = T, stringsAsFactors = F)

# Auxiliary data
species <- read.csv("./SPECIES/species_list_20170201.csv", header=T, stringsAsFactors = F)
plot_zone <- read.csv("./BFI-DB/Plot_distribution_bfi.csv", header = T, stringsAsFactors = F)

# Run basic functions necessary in different portions of the r-script ----------
#-------------------------------------------------------------------------------#
# Run funciton to create plot circles
# For drawing circles with the tree lcoations as per BFI sample plot design
circleFun <- function(xcenter,ycenter,radius){
  tt <- seq(0,2*pi,length.out = 1000)
  xx <- xcenter + radius * cos(tt)
  yy <- ycenter + radius * sin(tt)
  return(data.frame(x = xx, y = yy))
}

# Function to add subplots to graphs
subplot_design <- function(xcoord,ycoord){
  a <- xcoord
```

```
  b <- ycoord
  sp <- list(
    annotate("path", x=circleFun(a,b,19)[,1], y=circleFun(a,b,19)[,2]),
    annotate("path", x=circleFun(a,b,8)[,1], y=circleFun(a,b,8)[,2]),
    annotate("path", x=circleFun(a+5.5,b,2.5)[,1], y=circleFun(a+5.5,b,2.5)[,2])
  )
  return(sp)
}

# To add full plot design to ggplot
plot_design <- list(
  subplot_design(0,0), geom_point(aes(x=0,y=0),cex=4),
  subplot_design(0,38), geom_point(aes(x=0,y=38),cex=3),
  subplot_design(38,0), geom_point(aes(x=38,y=0),cex=3),
  subplot_design(0,-38), geom_point(aes(x=0,y=-38),cex=3),
  subplot_design(-38,0), geom_point(aes(x=-38,y=0),cex=3)
)
#-------------------------------------------------------------------------------#

# Primary codes for preparing some basic data frame ---------------------------------
# Run following codes to prepare some primary data frames which will be used frequently across the script
```

```
# Rename team name column
names(plot_info)[33] <- "team"

# Create team data frame
team_name <- plot_info[,c("plot_id","team")]
names(team_name)[1] <- "plot_plot_id"

# Merge zone from plot info
tree <- merge(tree, plot_info[,c("plot_id", "location_zone")],
              by.x="plot_plot_id", by.y="plot_id", all.x=T)

# Merge: species fom species_list
species$tree_spp_code <- species$code
tree_sub <- merge(tree, species[,c("family", "tree_spp_code")], by="tree_spp_code", all.x=T)

# Merge: team name fom team
tree_sub <- merge(tree_sub, team_name, by="plot_plot_id", all.x = T)

# Prepare a common output data table
output <- data.frame(cp="CP",cat="Category",plot_id=9999,issue="Issue", "team" = 99,  stringsAsFactors = F)
#-------------------------------------------------------------------------------#
```

```
#-------------------------------------------------------------------------------#
# CP 1 (3): Plot, subplot and land feature related checks
#-------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------#
# CP 1.1 (3.1): Non sampled and inaccessible plots with reasons ----------------------
# Input: plot.csv
# Output: list of plots not sampled and inaccessible
#-------------------------------------------------------------------------------#
# A new Database is created using the key variabbles at plot level
plot_infoD<-plot_info[plot_info$plot_details_plot_status == 4,c("plot_id",
                                                                "plot_details_plot_nonsample_reason",
                                                                "plot_details_plot_nonsample_reason_label",
                                                                "plot_details_plot_nonsample_reason_qualifier",
                                                                "team")]
# A plot details issues are summary in a single variable
plot_infoD$issue<-paste(plot_infoD$plot_details_plot_nonsample_reason,"-",plot_infoD$plot_details_plot_nonsample_reason_label,"-
",plot_infoD$plot_details_plot_nonsample_reason_qualifier)
```

```
# New descriptve variables are created

plot_infoD$cp<-"CP 3.1"

plot_infoD$cat<-"Plot non sampled"

plot_infoD$plot_id<-9999

# The final table are saved

Output11ns<-plot_infoD[,c("cp","cat","plot_id","issue","team")]


####### prepare the table of partially accessible

# A new Database is created using the key variabbles at plot level

plot_infoD<-plot_info[plot_info$plot_details_plot_status == 2,c("plot_id",
                                                                "plot_details_plot_nonsample_reason",
                                                                "plot_details_plot_nonsample_reason_label",
                                                                "plot_details_plot_nonsample_reason_qualifier",
                                                                "team")]

# A plot details issues are summary in a single variable

plot_infoD$issue<-paste(plot_infoD$plot_details_plot_nonsample_reason,"-",plot_infoD$plot_details_plot_nonsample_reason_label,"-
",plot_infoD$plot_details_plot_nonsample_reason_qualifier)

# New descriptve variables are created

plot_infoD$cp<-"CP 3.1"

plot_infoD$cat<-"Paritally accessible plot"

plot_infoD$plot_id<-9999
```

```
# The final table are saved

Output11pa<-plot_infoD[,c("cp","cat","plot_id","issue","team")]


####### prepare the table of inaccessible plot

# A new Database is created using the key variabbles at plot level

plot_infoD<-plot_info[plot_info$plot_details_plot_status == 3,c("plot_id",
                                                                "plot_details_plot_nonsample_reason",
                                                                "plot_details_plot_nonsample_reason_label",
                                                                "plot_details_plot_nonsample_reason_qualifier",
                                                                "team")]

# A plot details issues are summary in a single variable

plot_infoD$issue<-paste(plot_infoD$plot_details_plot_nonsample_reason,"-",plot_infoD$plot_details_plot_nonsample_reason_label,"-
",plot_infoD$plot_details_plot_nonsample_reason_qualifier)

# New descriptve variables are created

plot_infoD$cp<-"CP 3.1"

plot_infoD$cat<-"Inaccessible plot"

plot_infoD$plot_id<-9999

# The final table are saved

Output11ia<-plot_infoD[,c("cp","cat","plot_id","issue","team")]


# rbind all plots that are non-sampled, partially accessible and inaccessible
```

```
output1.1 <- rbind(Output11ns, Output11pa, Output11ia)
write.csv(output1.1,
        file = "./QAQC_outputs/1.1_non-sampled, partially accessible and inaccessible plots.csv",
        row.names = F)

#-------------------------------------------------------------------------------#
# CP 1.2 (3.2): LF recorded but missing LF object ----------------------------------
#              (i.e. vegetated, non-vegetated etc.)
# Input: lf.csv and lf_object.csv
# Output: list of plots missing LF objects
#-------------------------------------------------------------------------------#

#identify the plot and lf missing lf_object
lf_object$plot_lf <- paste(lf_object$plot_plot_id, lf_object$lf_lf_id, sep = "_")
test <- merge(lf[,c("plot_plot_id", "lf_id")], unique(lf_object[,c("plot_plot_id", "lf_lf_id", "plot_lf" )]),
            by.x= c("plot_plot_id", "lf_id"), by.y=c("plot_plot_id", "lf_lf_id"), all.x=T)
#prepare the data frame to incorporate with output table
test <- merge(test, team_name, by="plot_plot_id", all.x = T)
names(test)[1] <- "plot_id"
test$cat <- "Plot not completed"
test$issue <- paste("lf_object of lf", test$lf_id,  "is missing", sep = " ")
```

```
test$cp <- "CP 3.2"
test <- test[which(is.na(test$plot_lf)),]

output1.2 <- test[,c("cp","cat","plot_id","issue", "team")]
write.csv(output1.2,
        file = "./QAQC_outputs/1.2_LF recorded but LF object missing.csv",
        row.names = F)

#----------------------------------End of the check 1.2--------------------------------#



#-------------------------------------------------------------------------------------#
# CP 1.3 (3.3): Trees were recorded in non-vegetated LF object ------------------------
# Input: lf.csv; lf_object.csv and tree.csv
# Output: list of plots with number of trees where trees were recorded in
# non-vegetated plots
#-------------------------------------------------------------------------------------#
####### Create a table with plot, lf, object and tree_no
# select the min object by plot and lf level
plo <- aggregate(object~plot_plot_id+lf_lf_id, data=lf_object, FUN=min) #plo = plot_lf_object
# aggregate number of trees measured per plot
```

```
tree_sub$count <- 1
test <- aggregate(count~plot_plot_id+tree_lfid, data=tree_sub, FUN=sum)
test <- merge(test, plo, by.x=c("plot_plot_id","tree_lfid"),
        by.y=c("plot_plot_id", "lf_lf_id"), all.x = T)

####### Identify plot and lf with non_vegetated objects but with trees and place in output
# Identify plot and lf with non-vegetated object
library(dplyr)
test <- filter(test, object>1 & count>1)
#prepare the table as output table
test <- merge(test, team_name, by="plot_plot_id", all.x = T)
names(test)[1] <- "plot_id"
test$cp <- "CP 1.3"
test$cat <- "Inconsistency LF/Trees"
test$issue <- paste(test$count, " Trees in a no-veg object of LF ", test$tree_lfid, "- wrong lf object?")
output1.3 <- test[,c("cp","cat","plot_id","issue", "team")]
write.csv(output1.3,
      file = "./QAQC_outputs/1.3_Trees recorded in non-vegetated LF object.csv",
      row.names = F)

#-------------------------------------------------------------------------------#
```

```
# CP 1.4 (3.4): Trees present in herb, grass, shrub dominated LF ----------------------
# Input: lf.csv and tree.csv
# Output: List of plots with trees in non-tree vegetation type
#-------------------------------------------------------------------------------------#
####### Create a table with plot, lf, veg_type and tree_no
# select the min veg_type by plot and lf
pl_veg <- aggregate(vegetation_veg_type~plot_plot_id+lf_lf_id,
                    data=lf_object, min) #pl_veg = plot veg type
# aggregate and merge number of trees measured per plot
test <- aggregate(count~plot_plot_id+tree_lfid, data=tree_sub, FUN=sum)
test <- merge(test, pl_veg, by.x=c("plot_plot_id","tree_lfid"),
              by.y=c("plot_plot_id", "lf_lf_id"), all.x = T)

####### Identify plot and lf vegetated shrubs or herbs but having trees and
#       place in output
# Identify plot and lf with non-vegetated object
library(dplyr)
test <- filter(test, vegetation_veg_type>1 & count>1)
#prepare the table as output table
test <- merge(test, team_name, by="plot_plot_id", all.x = T)
names(test)[1] <- "plot_id"
```

```
test$cp <- "CP 3.4"
test$cat <- "Inconsistency veg_type Vs trees"
test$issue <- paste(test$count, " trees in veg_type ", test$vegetation_veg_type,
                    " of LF", test$tree_lfid, "- wrong veg_type?")
#The final table is saved
output1.4 <- test[,c("cp","cat","plot_id","issue", "team")]
write.csv(output1.4,
          file = "./QAQC_outputs/1.4_Trees recorded in herb dominated LF object.csv",
          row.names = F)


#-------------------------------------------------------------------------------#
# CP 1.5 (3.5): Strange relationship between leaf cover and crown cover ----------
# Input: subplot.csv, lf.csv
# Output: List of plots with strange relationship between leaf cover and crown cover
#-------------------------------------------------------------------------------#
# Read of Lf and Subplot Databases including key variables
lfr<-lf[,c("plot_plot_id","lf_id","lf_details_cover_min","lf_details_cover_max")]
subplotR<-subplot_info[,c("plot_plot_id","subp_id","subp_details_subp_leaf_pc")]
#Estimation of average of Leave Coverage
TableLcPl<-aggregate(subp_details_subp_leaf_pc~plot_plot_id, data = subplotR,FUN=mean)
#Merge of "lfr" and "TableLcPl" data frames
```

```
TableLcPlm<-merge(TableLcPl,lfr,by.x = "plot_plot_id", by.y = "plot_plot_id")
#Identifiction of plot where Leaf cover is reported and no Crown cover
TableLcPlm$CanopyErrror<-ifelse(TableLcPlm$subp_details_subp_leaf_pc>0 & TableLcPlm$lf_details_cover_max==0,1,0)
TableLcPlmF<-TableLcPlm[TableLcPlm$CanopyErrror==1,]
#Merge of "TableLcPlmF" and "team_name" data frames
TableLcPlmF<-merge(TableLcPlmF,team_name,by.x = "plot_plot_id", by.y = "plot_plot_id")
# New descriptve variables are created
TableLcPlmF$cp<-"CP 3.5"
TableLcPlmF$cat<-"Inconsistency in Crown-Leaf cover"
TableLcPlmF$plot_id<-TableLcPlmF$plot_plot_id
TableLcPlmF$issue<-"Leaf cover is reported and no Crown cover"
# The final table are saved
output1.5<-TableLcPlmF[,c("cp","cat","plot_id","issue","team")]
write.csv(output1.5,
        file = "./QAQC_outputs/1.5_Leaf cover is reported and no Crown cover.csv",
        row.names = F)



#-------------------------------------------------------------------------------------#
# CP 1.6 (3.6): Presence of Reference Point (RP) ----------------------------------------
# Input: plot.csv
```

```
# Output: List of plots missing RP
#-------------------------------------------------------------------------------#

#plot_info_rp <- plot_info[which(plot_info$plot_details_plot_status <=2),]
plot_info_rp <- plot_info[which(is.na(plot_info$rp_attributes_rp_type)),c("plot_id",
                                                    "plot_details_plot_status",
                                                    "rp_attributes_rp_type")]
names(plot_info_rp)[1]<-"plot_plot_id"
plot_info_rp<- merge(plot_info_rp, team_name, by="plot_plot_id", all.x = T)
test<-plot_info_rp
names(test)[1] <- "plot_id"
test$cp <- "CP 3.6"
test$cat <- "Missing value"
test$issue<- "Plot sampled but reference point is absent"
test <- test[which(is.na(test$rp_attributes_rp_type) &
                           test$plot_details_plot_status!=4),]
# save the outputs
output1.6 <- test[,c("cp","cat","plot_id","issue", "team")]
write.csv(output1.6,
        file = "./QAQC_outputs/1.6_Plot sampled but reference point is absent.csv",
        row.names = F)
```

```
#-------------------------------------------------------------------------------#
# CP 1.7 (3.7): Consistency in distance of reference point-----------------------------
# Input: plot.csv
# Output: List of plots having RP at more than 300 m distance
#-------------------------------------------------------------------------------#
plot_info_rpdist <- plot_info[c("plot_id","rp_attributes_rp_dist")]
plot_info_rpdist <- plot_info_rpdist[which(plot_info_rpdist$rp_attributes_rp_dist> 300),] #RP in more than 200 m is considered inconsistent
names(plot_info_rpdist)[1]<-"plot_plot_id"
plot_info_rpdist<- merge(plot_info_rpdist, team_name, by="plot_plot_id", all.x = T)
test<-plot_info_rpdist
test$issue<- paste("Reference point is ", test$rp_attributes_rp_dist, "m away from plot centre", sep = "")
names(test)[1] <- "plot_id"
test$cp <- "CP 1.7"
test$cat <- "Potential inconsistency in RP"
output1.7 <- test[,c("cp","cat","plot_id","issue", "team")]
write.csv(output1.7,
        file = "./QAQC_outputs/1.7_Ref point is more than 300m away.csv",
        row.names = F)
```

```
#-------------------------------------------------------------------------------#
# CP 1.8 (3.8): Outlier in the subplot slope by forest zone ----------------------------------
# (hill>100%, village>15%,Sal forest<20%, coastal and sundarbans >10%)
# Input: plot.csv and subplot.csv
# Output: List of plots with extreme values of slopes in different zones as mentioned
#-------------------------------------------------------------------------------#

# Merge with BFI zone and team name
subplot_info <- merge(subplot_info, plot_zone[,c("Zone","Plot_ID")],
                      by.x="plot_plot_id", by.y="Plot_ID", all.x=TRUE)
subplot_info<-merge(subplot_info, team_name, by="plot_plot_id", all.x=TRUE)

# Identify the plots subplots having unusual slopes in different zones
subplot_info$issue <- ifelse((subplot_info$Zone=="Coastal"|subplot_info$Zone=="Sundarbans") & abs(subplot_info$subp_details_subp_slope
>10),
                             paste("Coastal/Subdarbans zone but subplot slope ", abs(subplot_info$subp_details_subp_slope)),
                             ifelse(subplot_info$Zone=="Hill" & abs(subplot_info$subp_details_subp_slope>100),
                                    paste("Hill zone but Subplot slope ", abs(subplot_info$subp_details_subp_slope)),
                                    ifelse(subplot_info$Zone=="Sal" & abs(subplot_info$subp_details_subp_slope>20),
                                           paste("Sal zone but suplot slope", abs(subplot_info$subp_details_subp_slope)),
                                           ifelse(subplot_info$Zone=="Villages" & abs(subplot_info$subp_details_subp_slope>15),
```

```
                                    paste("Village zone but subplot slope", abs(subplot_info$subp_details_subp_slope)), "Slope is OK"))))

#prepare output table with extreme slopes
test <- subplot_info
test$cp <- "CP 3.8"
test$cat <- "Extreme slopes in subplots"
names(test)[1] <- "plot_id"
output1.8 <- test[subplot_info$issue!="Slope is OK"& !is.na(test$subp_details_subp_slope),
                  c("cp","cat","plot_id","issue", "team")]

write.csv(output1.8,
      file = "./QAQC_outputs/1.8_extreme slopes in subplots.csv",
      row.names = F)

#-------------------------------------------------------------------------------#
# CP 1.9 (3.10): Inconsistency between LF ownership and legal status ----------
# Input: lf.csv
# Output: list of Plot and LF with ownership and legal status inconsistency
#-------------------------------------------------------------------------------#
lf$owner_issue <- ifelse(lf$lf_details_lf_owner==2 & is.na(lf$lf_details_land_legal),
                  "legal status BFD land is NA",
```

```
                ifelse(lf$lf_details_lf_owner==2 & lf$lf_details_land_legal==99,
                    "BFD land legal status is Unknown!",
                    ifelse(lf$lf_details_lf_owner!=2 & !is.na(lf$lf_details_land_legal),
                        paste("Not BFD land but legal status is ",lf$lf_details_land_legal_label),
                        ifelse(is.na(lf$lf_details_lf_owner), "ownership info missing",
                        "Legal status is OK"))))
owner_incons <- lf[,c("plot_plot_id", "lf_id", "lf_details_lf_owner","lf_details_land_legal", "owner_issue")]
owner_incons <- merge(owner_incons, team_name, by="plot_plot_id", all.x=T)
names(owner_incons)[2] <- "plot_id"
owner_incons$cp <- "CP3.10"
owner_incons$cat <- "LF Ownership and legal status inconsistent"
owner_incons$issue <- owner_incons$owner_issue
# save the outputs
output1.9 <- owner_incons[owner_incons$owner_issue!="Legal status is OK",
                c("cp", "cat", "plot_id","issue","team")]
write.csv(output1.9,
      file = "./QAQC_outputs/1.9_LF Ownership and legal status inconsistent.csv",
      row.names = F)



#-------------------------------------------------------------------------------#
```

```
# CP 1.10: Mismatching subplot between subplot info and LF subplot proportion ---------
# Input: lf_subp_prop.csv and subplot.csv
# Output: List of plots with mismatching subplot between subplot info and LF subplot
# proportion
#-------------------------------------------------------------------------------------#
# Identify key variables from subplot proportion (lf_subp_prop.csv)
spKV <- subplot_prop[,c("plot_plot_id", "subplot_subp_id")]
# Identify key variables from subplot info
sKV <- subplot_info[subplot_info$subp_details_subp_status==1,][,c("plot_plot_id", "subp_id")]
### Check mismatches in plot and subplot
#install.packages("sqldf")
require(sqldf)
# plot and completely sampled subplots present in subplot.csv but absent in lf_subp_prop.csv
sKV_NotIn_spKV <- sqldf('SELECT * FROM sKV EXCEPT SELECT * FROM spKV')
sKV_NotIn_spKV <- sKV_NotIn_spKV[!is.na(sKV_NotIn_spKV$plot_plot_id),]

# Prepare the output table
if(dim(sKV_NotIn_spKV)[1]==0){
  ouptut1.10 <- data.frame(cp = "CP 1.10",
                           cat = "Mismatching subplot in subplot.csv and lf_subp_prop.csv",
                           plot_id= NA, issue = "No mismatch found", team = NA)
```

```
}else{
 sKV_NotIn_spKV$cp <- "CP 1.10"
 sKV_NotIn_spKV$cat <- "Mismatching subplot in subplot.csv and lf_subp_prop.csv"
 sKV_NotIn_spKV$issue <- paste("Plot-", sKV_NotIn_spKV$plot_plot_id, ", Subp-",
                        sKV_NotIn_spKV$subp_id,
                        ", present in subplot but absent in lf_sup_prop")
 sKV_NotIn_spKV <- merge(sKV_NotIn_spKV, team_name, by="plot_plot_id", all.x=T)
 names(sKV_NotIn_spKV)[1] <- "plot_id"
 output1.10 <- sKV_NotIn_spKV[,c("cp", "cat", "plot_id", "issue", "team")]
 }
# export the outputs
write.csv(output1.10, file="./QAQC_outputs/1.10_Mismatched plots and subplots in subplot and subp proportion.csv",
        row.names = F)


#-------------------------------------------------------------------------------------#
# CP 1.11: Missing vegetation management information for "Vegetated" LF object---------
# Input: lf.csv, lf_object.csv
# Output: List of plots and LF object where management information is missing
#-------------------------------------------------------------------------------------#
mgt_blank <- lf_object[lf_object$object==1 &
                    is.na(lf_object$vegetation_management),]
```

```
mgt_blank <- merge(mgt_blank, plot_info[,c("plot_id", "team")], by.x="plot_plot_id", by.y= "plot_id", all.x=T)
mgt_blank <- mgt_blank[,c("plot_plot_id", "lf_lf_id", "object_label",
                          "vegetation_management","location_zone","team")]
mgt_blank$issue <- "Missing vegetation management info"
mgt_blank$cp <- "CP 1.11"
mgt_blank$cat <- "Missing vegetation management info"
names(mgt_blank)[1] <- "plot_id"
output1.11 <- mgt_blank[,c("cp","cat","plot_id","issue", "team")]
write.csv(output1.11,
        file = "./QAQC_outputs/1.11_Missing vegetation management info.csv",
        row.names = F)


#-------------------------------------------------------------------------------#
# CP 1.12: Missing LF issue information ----------------------------------------------
# Input: lf_issue.csv
# Output: List of plots and LF missing lf_issue information
#-------------------------------------------------------------------------------#
lfi <- merge(lf[,c("plot_plot_id", "lf_id")],lf_issue, by.x=c("plot_plot_id", "lf_id"),
           by.y= c("plot_plot_id", "lf_lf_id"), all.x=T)#lfi = lf_issue
```

```
if(dim(lfi[is.na(lfi$X_lf_issues_position),])[1]==0){
  output1.12 <- data.frame(cp="CP 1.12", cat= "Missing lf_issue info", plot_id = NA,
                           issue="No lf_issue information is missed", team=NA)
}else{
  lfi_missing <- lfi[is.na(lfi$X_lf_issues_position),]
  lfi_missing$cp <- "CP 1.12"
  lfi_missing$cat <- "Missing lf_issue info"
  lfi_missing$issue <- "Missing lf_issue info"
  lfi_missing <- merge(lfi_missing, team_name, by="plot_plot_id", all.x=T)
  names(lfi_missing)[1] <- "plot_id"
  output1.12 <- lfi_missing[,c("cp", "cat", "plot_id", "issue", "team")]
}
# Save the output
write.csv(output1.12, file = "./QAQC_outputs/1.12_missing lf_issue info.csv",
          row.names = F)


#-------------------------------------------------------------------------------#
# CP 1.13: Non-uniform information recorded under LF issue ----------------------
# Input: lf_issue.csv
# Output: List of plots with LF issues recorded under other category that need
# generalization
```

```
#-------------------------------------------------------------------------------------#
lfi_other <- lf_issue[lf_issue$lf_issues_label=="Other",]
lfi_other <- lfi_other[,c("plot_plot_id", "lf_lf_id", "lf_issues", "lf_issues_qualifier", "lf_issues_label")]
lfi_other <- merge(lfi_other, plot_info[,c("plot_id", "location_zone", "team")],
                   by.x="plot_plot_id", by.y="plot_id", all.x=T)
lfi_other$issue <- "lf_issues under 99 category need generalization"
lfi_other$cp <- "CP 1.13"
lfi_other$cat <- "LF issues non-uniform under 99 category"
names(lfi_other)[1] <- "plot_id"
# save the output
output1.13 <- lfi_other[,c("cp", "cat", "plot_id", "issue", "team")]
write.csv(output1.13,
          file = "./QAQC_outputs/1.13_LF issues under 99 category are non-uniform.csv",
          row.names = F)


#-------------------------------------------------------------------------------------#
# CP 1.14: CWD data missing (for important variables only) ----------------------------
# Input: cwd.csv
# Output: List of plots with CWD missing lf id
#-------------------------------------------------------------------------------------#
# select the key variables
```

```
cwdKV <- cwd[cwd$cwd_tran!=0,][,c("plot_plot_id", "subplot_subp_id", "cwd_tran",
                          "cwd_lf", "cwd_sdist", "cwd_dia", "cwd_decay")]
# identify cwd missing LF ID, Distance, Diameter and Decay class info
CwdDM <- cwdKV[is.na(cwdKV$cwd_lf)|
               is.na(cwdKV$cwd_sdist)|
               is.na(cwdKV$cwd_dia)|
               is.na(cwdKV$cwd_decay),] #CwdDM = cwd data missed


# prepare the output table
if(dim(CwdDM)[1]==0){
  output1.14 <- data.frame(cp="CP 1.14", cat= "CWD data missing", plot_id = NA,
                     issue="No data on important variables missed", team=NA)
  }else{
  CwdDM$cp <- "CP 1.14"
  CwdDM$cat <- "CWD data missing"
  CwdDM$issue <- ifelse(is.na(CwdDM$cwd_lf), "LF ID missed",
                    ifelse(is.na(CwdDM$cwd_sdist), "Distance missed",
                         ifelse(is.na(CwdDM$cwd_dia), "Diameter missed",
                         "Decay class missed")))
  CwdDM <- merge(CwdDM, team_name, by="plot_plot_id", all.x=T)
  names(CwdDM)[1] <- "plot_id"
```

```
 output1.14 <- CwdDM[,c("cp", "cat", "plot_id", "issue", "team")]
 }
# save the output
write.csv(output1.14, file = "./QAQC_outputs/1.14_CWD data missing.csv", row.names = F)


#-------------------------------------------------------------------------------#
# CP 1.15: FWD data missing (for important variables only) ----------------------
# Input: fwd.csv
# Output: List of plots with CWD missing lf id
#-------------------------------------------------------------------------------#
# select the key variables
fwdKV <- fwd[fwd$fwd_tran!=0,][,c("plot_plot_id", "subplot_subp_id", "fwd_tran",
                          "fwd_lfid", "fwd_smallct", "fwd_medct", "fwd_lrgct")]


# identify Fwd missing LF ID
FwdDM <- fwdKV[is.na(fwdKV$fwd_lfid),]


# prepare the output table
if(dim(FwdDM)[1]==0){
 output115 <- data.frame(cp="CP 1.15", cat= "FWD data missing", plot_id = NA,
                    issue="No data on important variables missed", team=NA)
```

```
}else{
  FwdDM$cp <- "CP 1.15"
  FwdDM$cat <- "FWD data missing"
  FwdDM$issue <- "FWD LF id is missed"
  FwdDM <- merge(FwdDM, team_name, by="plot_plot_id", all.x=T)
  names(FwdDM)[1] <- "plot_id"
  output1.15 <- FwdDM[,c("cp", "cat", "plot_id", "issue", "team")]
}
# save the output
write.csv(output1.15, file = "./QAQC_outputs/1.15_FWD data missing.csv", row.names = F)



#-----------------------------------------------------------------------------------#
# CP 2 (4): Tree and sapling related checks ----------------------------------------
#-----------------------------------------------------------------------------------#
####### Prepare data
### Prepare LF data with updated/assigned nlcl
#import new lcc and update in lf
lcc_new <- read.csv("./BFI-DB/lf_class_assigned_20190414.csv", header=T, stringsAsFactors = F)
lf <- merge(lf, lcc_new[,c("plot_plot_id", "lf_id", "nlcl_assigned")], by=c("plot_plot_id", "lf_id"), all.x=T)
names(lf)[32] <- "nlcl_field"
```

```
lf$nlcl <- ifelse(is.na(lf$nlcl_assigned), lf$nlcl_field,lf$nlcl_assigned)
lf$nlcl <- ifelse(lf$nlcl=="PCm"|lf$nlcl=="PCs", "PC",
                  ifelse(lf$nlcl=="FEh", "FH",
                         ifelse(lf$nlcl=="B", "BS", lf$nlcl)))

##### Create a single table containing trees and sapling
### Prepare sapling data
#merge saplings_live and lf_id from subplot_prop
saplings_live <- saplings[which((!is.na(saplings$sap_dia)) & (!is.na(saplings$sap_total_lgt)) &
                                saplings$sap_sp_scientific_name!=""),]
saplings_live <- merge(saplings_live,subplot_prop[subplot_prop$lf_s_percent==100,
                                                  c("plot_plot_id","subplot_subp_id","lf_subp_lfno")],
                       by = c("plot_plot_id","subplot_subp_id"), all.x=TRUE)

#to sort the plots
QC_h=sort(table(saplings_live[is.na(saplings_live$lf_subp_lfno),1]),decreasing = TRUE)
saplings_live=saplings_live[!is.na(saplings_live$lf_subp_lfno),]

saplings_live= merge(saplings_live,  lf[,c("plot_plot_id","lf_id","nlcl")],
                     by.x=c("plot_plot_id","lf_subp_lfno"),
                     by.y= c("plot_plot_id","lf_id"), all.x=TRUE)
```

```
# new sapling table and renames of the columns to merge with tree dataset
new_saplings <- saplings_live
names(new_saplings)[2]="tree_lfid"
names(new_saplings)[4]="tree_id"
names(new_saplings)[5]="tree_spp_code"
names(new_saplings)[6]="tree_spp_scientific_name"
names(new_saplings)[10]="tree_bearing"
names(new_saplings)[12]="tree_dist"
names(new_saplings)[14]="tree_dia"
names(new_saplings)[18]="tree_total_lgt"
new_saplings$tree_status=99 #99 for saplings

str(new_saplings)
#create dataframe with some selected column
new_saplings <- new_saplings[,c("plot_plot_id","tree_lfid","subplot_subp_id","tree_id",
                        "tree_spp_code","tree_spp_scientific_name",
                        "tree_status","tree_bearing","tree_dist",
                        "tree_dia","tree_total_lgt", "nlcl")]

### prepare tree data
```

```
# add nlcl to tree data
tree <- merge(tree, lf[,c("plot_plot_id", "lf_id", "nlcl")],
              by.x= c("plot_plot_id", "tree_lfid"), by.y =c("plot_plot_id", "lf_id"), all.x=TRUE)

### merge tree and sapling in one data frame
library(gtools)
treesap <- smartbind(tree, new_saplings) #smartbind will add the data of same col names under the data of first dataframe

#Use trees/saplings from completely sampled subplots only
treesap <- merge(treesap, subplot_info[,c("plot_plot_id","subp_id","subp_details_subp_status_label")],
                 by.x=c("plot_plot_id", "subplot_subp_id"), by.y=c("plot_plot_id", "subp_id"), all.x=T)
treesap <- treesap[treesap$subp_details_subp_status_label=="Sampled completely",]



#-------------------------------------------------------------------------------#
# CP 2.1 (4.1): NA in tree DBH ----------------------------------------------------------
# Input: tree.csv and sapling.csv
# Output: List of plots and trees with NA in DBH
#-------------------------------------------------------------------------------#
# number of NA in DBH column for tree data
dim(tree[which(is.na(tree$tree_dia)),])[1]
```

```
dim(tree_sub[which(is.na(tree_sub$tree_dia)),])[1]


if (dim(tree_sub[which(is.na(tree_sub$tree_dia)),])[1] == 0){


  output21t <- data.frame(cp="CP 4.1",cat="Missing value in DBH",plot_id=9999,issue="DBH passed successfully for trees", "team" = 99,
stringsAsFactors = F)


} else {
  tree_sub$issue= paste("dbh is NA in tree: tree_id-", tree_sub$tree_id, "subp_id-",tree_sub$subplot_subp_id)
  tree_missdbh <-  data.frame(cp="CP 2.1",cat= "Missing value in DBH",
                              plot_plot_id = unique(tree_sub[which(is.na(tree_sub$tree_dia)),]$plot_plot_id),
                              issue=tree_sub[is.na(tree_sub$tree_dia),]$issue)
  tree_missdbh <- merge(tree_missdbh, team_name, by="plot_plot_id", all.x = T)
  names(tree_missdbh)[1] <- "plot_id"
  output21t <- tree_missdbh[,c("cp", "cat", "plot_id", "issue", "team")]


}


# no NA in DBH column for sapling data
nb_pb=dim(saplings[which((is.na(saplings$sap_dia)) & (saplings$sap_notes!="No sapling found")),])[1]
```

```
if (nb_pb == 0){


  output21s <- data.frame(cp="CP 4.1",cat="Missing value in DBH",plot_id=9999,issue="DBH passed successfully for saplings", "team" = 99,
stringsAsFactors = F)


} else {
  saplings$issue= ifelse(is.na(saplings$sap_dia) & saplings$sap_notes!="No sapling found",
                         paste("dbh is NA in sapling: sapling_id-", saplings$sapling_id, "subp_id-",saplings$subplot_subp_id),
                         NA)
  sap_missdbh <-  data.frame(cp="CP 4.1",cat= "Missing value in DBH",
                             plot_plot_id = unique(saplings[which((is.na(saplings$sap_dia)) & (saplings$sap_notes!="No sapling found")),]$plot_plot_id),
                             issue="Sapling DBH is NA")
  sap_missdbh <- merge(sap_missdbh, team_name, by="plot_plot_id", all.x = T)
  names(sap_missdbh)[1] <- "plot_id"
  output21s <- sap_missdbh[,c("cp", "cat", "plot_id", "issue", "team")]


}


# bind outputs from both trees and saplings
output2.1 <- rbind(output21t, output21s)
write.csv(output2.1, file = "./QAQC_outputs/2.1_Tree and sapling DBH missing.csv",
```

```
        row.names = F)


#-------------------------------------------------------------------------------#
# CP 2.2 (4.1.1): Saplings having DBH <2 cm ----------------------------------------
# Input: sapling.csv
# Output: list of plots and saplings with DBH < 2 cm
#-------------------------------------------------------------------------------#


sap_dbhsmall <- saplings[!is.na(saplings$sap_dia) &
                          saplings$sap_dia<2,]


if(dim(sap_dbhsmall)[1]==0){
  output2.2 <- data.frame(cp="CP 22",
                          cat="Saplings having DBH <2 cm",
                          plot_id=9999,
                          issue="All sapling dbh are within specified range",
                          team = 99, stringsAsFactors = F)
  }else{
  sap_dbhsmall$cp <- "CP 22"
  sap_dbhsmall$cat <- "Saplings having DBH <2 cm"
  sap_dbhsmall$issue <- paste("sapling but dbh ",
```

```
                    sap_dbhsmall$sap_dia,
                    "cm (subp_id ",sap_dbhsmall$subplot_subp_id,
                    ", sap_id ", sap_dbhsmall$sapling_id, ")",
                    sep = "")
 sap_dbhsmall <- merge(sap_dbhsmall, team_name, by="plot_plot_id", all.x=T)
 names(sap_dbhsmall)[1] <- "plot_id"
 output2.2 <- sap_dbhsmall[,c("cp", "cat","issue", "plot_id","team")]
 }

# save the outputs
write.csv(output2.2, file = "./QAQC_outputs/2.2_Sapling having less than 2cm DBH.csv",
        row.names = F)

#-------------------------------------------------------------------------------#
# CP 2.3 (4.1.2): Trees having DBH <10 cm ----------------------------------------
# Input: tree.csv
# Output: list of plots and trees having DBH < 10cm
#-------------------------------------------------------------------------------#

tree_dbhsmall <- tree[!is.na(tree$tree_dia) &
                    tree$tree_dia<10,]
```

```
if(dim(tree_dbhsmall)[1]==0){
  output2.3 <- data.frame(cp="CP 23",
                          cat="trees having DBH <10 cm",
                          plot_id=9999,
                          issue="All tree dbh are within specified range",
                          team = 99, stringsAsFactors = F)
}else{
  tree_dbhsmall$cp <- "CP 23"
  tree_dbhsmall$cat <- "Trees having DBH <10 cm"
  tree_dbhsmall$issue <- paste("Tree but dbh ",
                               tree_dbhsmall$tree_dia,
                               "cm (subp_id ",tree_dbhsmall$subplot_subp_id,
                               ", tree_id ", tree_dbhsmall$tree_id, ")", sep = "")
  tree_dbhsmall <- merge(tree_dbhsmall, team_name, by="plot_plot_id", all.x=T)
  names(tree_dbhsmall)[1] <- "plot_id"
  output2.3 <- tree_dbhsmall[,c("cp", "cat", "issue", "plot_id","team")]
  }

# save the outputs
write.csv(output2.3, file = "./QAQC_outputs/2.3_Trees having less than 10cm DBH.csv",
```

```
        row.names = F)


#-------------------------------------------------------------------------------#
# CP 2.4: Outliters (extreme values) in DBH per species ------------------------------
# Input: tree.csv and sapling.csv
# Output: list of plots and trees/saplings species haivng extreme values in DBH
#-------------------------------------------------------------------------------#
### Data preparation
treesap_live <- treesap[(treesap$tree_status==1|
                         treesap$tree_status==99),]
# Tree DB with key variables
treesap_liveKV <- treesap_live[,c("plot_plot_id","subplot_subp_id","tree_lfid",
                                  "tree_status","tree_spp_scientific_name","tree_dia",
                                  "tree_total_lgt")]
# Live trees are filtered
# Missing values and mistakes are eliminated at scientific_name, tree_dia, length
treesap_liveKVf <- treesap_liveKV[!is.na(treesap_liveKV$tree_dia) &
                                    !treesap_liveKV$tree_dia<2 &
                                    !is.na(treesap_liveKV$tree_total_lgt) &
                                    !treesap_liveKV$tree_total_lgt<=1.3 &
                                    treesap_liveKV$tree_spp_scientific_name!="",]
```

```
### A table of statistics (mean and sd) are done
f24 <- function(x) c(n = length(x), Mean = mean(x), SD = sd(x))
BasicSat <- aggregate(tree_dia ~ tree_spp_scientific_name, data=treesap_liveKVf, FUN=f24)
BasicSat$n<-BasicSat$tree_dia[,1]
BasicSat$Mean<-BasicSat$tree_dia[,2]
BasicSat$SD<-BasicSat$tree_dia[,3]

### Tree database and Statistical database are merged by sientific name
treesap_liveKVfSt<-merge(treesap_liveKVf,BasicSat[,c("tree_spp_scientific_name","n", "Mean", "SD")],
                         by.x="tree_spp_scientific_name",
                         by.y="tree_spp_scientific_name",all.x=T)
treesap_liveKVfStF<- treesap_liveKVfSt[treesap_liveKVfSt$n>3,]

### The Z value is estimated and Possible Outliers are identified
# Z is estimated
treesap_liveKVfStF$Z<-((treesap_liveKVfStF$tree_dia-treesap_liveKVfStF$Mean)/
                         treesap_liveKVfStF$SD)
# Possible Outliers are identify
treesap_liveKVfStF$PO<-ifelse(abs(treesap_liveKVfStF$Z)>4,1,0)
```

```
### Possible Outliers are summarized and saved
# A Database of possible OL is created
treesap_liveKVfStF$issue <- paste("Spp:", treesap_liveKVfStF$tree_spp_scientific_name,
                          "DBH:", treesap_liveKVfStF$tree_dia,
                          "Mean:",round(treesap_liveKVfStF$Mean,1),
                          "Z:",round(treesap_liveKVfStF$Z,1),sep = " ")
treesap_liveKVfStF$cp <- "CP 2.4"
treesap_liveKVfStF$cat <- "Outliers in DBH"
treesap_liveKVfStF <- merge(treesap_liveKVfStF, team_name, by="plot_plot_id", all.x=T)
names(treesap_liveKVfStF)[1] <- "plot_id"

# The database of Possible OL is created
output2.4 <- treesap_liveKVfStF[treesap_liveKVfStF$PO==1,
                         c("cp", "cat", "plot_id", "issue", "team")]
# Save the outputs
write.csv(output2.4, file = "./QAQC_outputs/2.4_Possible outliers in tree DBH.csv",
        row.names = F)

#-------------------------------------------------------------------------------------#
# CP 2.5 (4.2): NA in tree/sapling total length ----------------------------------------
# Input: tree.csv and saplings.csv
```

```
# Output: List of plots and trees/saplings having NA in total tree length
#-------------------------------------------------------------------------------#
#Filter the tree, saplings and dead standing trees
tree_missh <- treesap[(treesap$tree_status <= 2|treesap$tree_status==99),]

#NA in tree/sapling/standing dead trees total length is identified
treeHNa <- tree_missh[which(is.na(tree_missh$tree_total_lgt)),]

#prepare the table to export as output
treeHNa$cp <- "CP 2.5"
treeHNa$cat <- "Total length is missing"
treeHNa$issue <- paste("Tree status: ", treeHNa$tree_status, ", Total length is NA", sep = "")
treeHNa <- merge(treeHNa, team_name, by="plot_plot_id", all.x=T)
names(treeHNa)[1] <- "plot_id"
output2.5 <- treeHNa[,c("cp","cat","plot_id","issue", "team")]
# save the output
write.csv(output2.5,
        file = "./QAQC_outputs/2.5_Tree and sapling total height missing.csv",
        row.names = F)
```

```
#-------------------------------------------------------------------------------------#
# CP 2.6 (4.3): Trees smaller than 1.3m total length ---------------------------------
# Input: tree.csv
# Output: List of plots and trees having height <1.3m
#-------------------------------------------------------------------------------------#
# Live trees are filtered only
tree_live <- tree_sub[which(tree_sub$tree_status == 1),]
tree_lessthan1 <- tree_live[which(is.na(tree_live$tree_total_lgt<1.3)),]
# Filter trees less than 1.3m height and prepare the output table
if(dim(tree_lessthan1)[1]==0){
  output4.3 <- data.frame(cp = "CP 2.6", cat="Inconsistency - small trees",
                          plot_id= NA, issue= "All trees have height > 1m",
                          team=NA)
}else{
  tree_lessthan1 <- data.frame(cp="CP 2.6",cat= "Inconsistency - small trees",
                               plot_plot_id = unique(tree_live[which(is.na(tree_live$tree_total_lgt<1.3)),]$plot_plot_id),
                               issue= "Tree length < 1m")
  tree_lessthan1 <- merge(tree_lessthan1, team_name, by="plot_plot_id", all.x = T)
  names(tree_lessthan1)[1] <- "plot_id"
  output2.6 <- tree_lessthan1[,c("cp","cat","plot_id","issue", "team")]}
```

```
# save the output
write.csv(output2.6,
          file = "./QAQC_outputs/2.6_Live trees having less than 1m total length.csv",
          row.names = F)


#-------------------------------------------------------------------------------#
# CP 2.7: Extreme values in tree/sapling total length per species ----------------
# Input: tree.csv and sapling.csv
# Output: List of plots and tree/sapling species having outliers in their total length
#-------------------------------------------------------------------------------#
### Data preparation
treesap_live <- treesap[(treesap$tree_status==1|
                         treesap$tree_status==99),]
# Tree DB with key variables
treesap_liveKV <- treesap_live[,c("plot_plot_id","subplot_subp_id","tree_lfid",
                                  "tree_status","tree_spp_scientific_name","tree_dia",
                                  "tree_total_lgt")]
# Live trees are filtered
# Missing values and mistakes are eliminated at scientific_name, tree_dia, length
treesap_liveKVf <- treesap_liveKV[!is.na(treesap_liveKV$tree_dia) &
                                    !treesap_liveKV$tree_dia<2 &
```

```
                    !is.na(treesap_liveKV$tree_total_lgt) &
                    !treesap_liveKV$tree_total_lgt<=1.3 &
                    treesap_liveKV$tree_spp_scientific_name!="",]

### A table of statistics (mean and sd) are done
f27 <- function(x) c(n = length(x), Mean = mean(x), SD = sd(x))
BasicSat <- aggregate(tree_total_lgt ~ tree_spp_scientific_name, data=treesap_liveKVf, FUN=f27)
BasicSat$n<-BasicSat$tree_total_lgt[,1]
BasicSat$Mean<-BasicSat$tree_total_lgt[,2]
BasicSat$SD<-BasicSat$tree_total_lgt[,3]

### Tree database and Statistical database are merged by sientific name
treesap_liveKVfSt<-merge(treesap_liveKVf,
                BasicSat[,c("tree_spp_scientific_name","n","Mean","SD")],
                by.x="tree_spp_scientific_name",
                by.y="tree_spp_scientific_name",all.x=T)
treesap_liveKVfStF<- treesap_liveKVfSt[treesap_liveKVfSt$n>3,]

### The Z value is estimated and Possible Outliers are identified
# Z is estimated
treesap_liveKVfStF$Z<-((treesap_liveKVfStF$tree_total_lgt-treesap_liveKVfStF$Mean)/
```

```
                    treesap_liveKVfStF$SD)
# Possible Outliers are identify
treesap_liveKVfStF$PO<-ifelse(abs(treesap_liveKVfStF$Z)>4,1,0)

### Possible Outliers are summarized and saved
# A Database of possible OL is created
treesap_liveKVfStF$issue <- paste("Spp:", treesap_liveKVfStF$tree_spp_scientific_name,
                          "Status:",treesap_liveKVfStF$tree_status,
                          "Ht:", treesap_liveKVfStF$tree_total_lgt,
                          "n:", treesap_liveKVfStF$n,
                          "Mean:",round(treesap_liveKVfStF$Mean,1),
                          "Z:",round(treesap_liveKVfStF$Z,1),sep = " ")
treesap_liveKVfStF$cp <- "CP 2.7"
treesap_liveKVfStF$cat <- "Outliers in tree/sapling length"
treesap_liveKVfStF <- merge(treesap_liveKVfStF, team_name, by="plot_plot_id", all.x=T)
names(treesap_liveKVfStF)[1] <- "plot_id"

# The database of Possible OL is created
output2.7 <- treesap_liveKVfStF[treesap_liveKVfStF$PO==1,
                         c("cp", "cat", "plot_id", "issue", "team")]
```

```
write.csv(output2.7,
        file = "./QAQC_outputs/2.7_Possible outliers in tree and sapling total length.csv",
        row.names = F)


#-------------------------------------------------------------------------------#
# CP 2.8 (4.4): Tree and Saplings measured outside their subplot ----------------
# Input: tree.csv and sapling.csv
# Output: List of plots and trees/saplings measured outside their subplot
#-------------------------------------------------------------------------------#


# FLAG tree dbh outside subplot


# flag 1: dbh < 2,
# flag 2: dbh < 10 & radius > 2.5 m
# flag 3: dbh < 30 & radius > 8 m
# flag 100: tree dist missing (NA)


tree_sub$flag_tree_dist <-ifelse(is.na(tree_sub$tree_dist),100,
                        ifelse(tree_sub$tree_dia < 2,1,
                          ifelse(tree_sub$tree_dia < 10 & tree_sub$tree_dist > 2.5, 2,
                            ifelse(tree_sub$tree_dia < 30 & tree_sub$tree_dist > 8, 3, 0))))
```

```
tree_sub$flag_tree_dist <- factor(tree_sub$flag_tree_dist)
summary(tree_sub$flag_tree_dist)


tree_dbhout <- tree_sub[which(!is.na(tree_sub$tree_dia) &
                        tree_sub$flag_tree_dist != 0 &
                        (tree_sub$flag_tree_dist != 100)),][,c("plot_plot_id","tree_id", "tree_dist",
                                                              "tree_dia", "tree_spp_scientific_name",
                                                              "tree_status","flag_tree_dist", "team")]


# Prepare the output table
if (dim(tree_dbhout)[1]== 0){
  output2.8 <- data.frame(cp="CP 2.8",cat="Tree measured outside its subplot",plot_id=9999,
                    issue="passed successfully", team = 99,  stringsAsFactors = F)
} else {
  # Prepare a table trees recorded outside the subplot
  tree_dbhout$cp <- "CP 2.8"
  tree_dbhout$cat <- "Tree measured outside its subplot"
  tree_dbhout$plot_id <- tree_dbhout$plot_plot_id
  tree_dbhout$issue <- ifelse(tree_dbhout$flag_tree_dist == 100,
                        "tree distance not measured and tree_status=1",
```

```
                    paste("tree dbh is",tree_dbhout$tree_dia, "cm and tree measured at",
                        tree_dbhout$tree_dist,"m", sep=" "))
  output2.8 <- tree_dbhout[,c("cp", "cat", "plot_id", "issue", "team")]}

# save the output
write.csv(output2.8,
        file = "./QAQC_outputs/2.8_Trees and saplings measured outside their subplot.csv",
        row.names = F)



#-----------------------Map tree location -------------------------------#
# The following codes will prepare 2 graphs with the location of the trees which showed the
# trees measured in different locations of the plot. Graph 1 ploted all the trees with their
# location in plot. Graph 2 showed the trees measured outside their subplot.

##### Graph 1: Plot all trees with their locaiton in plot
# tree coordinates from subplot center
tree$tree_loc_x <- cos(tree$tree_bearing*pi/180)*tree$tree_dist
tree$tree_loc_y <- sin(tree$tree_bearing*pi/180)*tree$tree_dist

# tree coordinates from plot center
```

```
tree$tree_x <- ifelse(tree$subplot_subp_id == 3,
                      tree$tree_loc_x + 38,
                      ifelse(tree$subplot_subp_id == 5,
                             tree$tree_loc_x - 38,
                             tree$tree_loc_x))


tree$tree_y <- ifelse(tree$subplot_subp_id == 2,
                      tree$tree_loc_y + 38,
                      ifelse(tree$subplot_subp_id == 4,
                             tree$tree_loc_y - 38,
                             tree$tree_loc_y))


# dbh class
tree$dbh_class <- ifelse(tree$tree_dia < 10, 1,
                         ifelse(tree$tree_dia < 30, 2,
                                ifelse(tree$tree_dia < 100, 3, 4)))


tree$dbh_class <- factor(tree$dbh_class, levels=c(1,2,3,4), labels=c("dbh < 10", "10 <= dbh < 30", "30 <= dbh < 100", "dbh > 100"))


# # all plots overlaid (+plot design)
p <- ggplot(tree)
```

```
p <- p + geom_point(aes(x=tree_x, y=tree_y, size = dbh_class, fill=dbh_class), shape=21, position="jitter")
p <- p + scale_size_manual(values=c(0.2,0.5,1,4), drop=FALSE)
p <- p + plot_design
p <- p + coord_fixed(ratio=1)
p <- p + theme(legend.position="none")
p

##### Graph 2: Plot only trees measured outside their subplot
# plot with trees outside their subplot
if(dim(tree_dbhout)[1]!=0){

 vec <- unique(tree_dbhout$plot_plot_id)
 list_plot <- list(a="starter")
 for(i in 1:length(vec)){

  p <- ggplot(tree[which(tree$plot_plot_id==vec[i]),])
  p <- p + geom_point(aes(x=tree_x, y=tree_y, size = dbh_class, fill=dbh_class), shape=21, position="jitter")
  p <- p + scale_size_manual(values=c(1,2,4,8), drop=FALSE)
  p <- p + plot_design + coord_fixed(ratio=1)
  p <- p + theme(legend.position="none")
  p <- p + xlab(paste("plot",vec[i], sep=" ")) + ylab("tree coordinates")
```

```
   list_plot <- list(list_plot,assign(paste("plot",vec[i], sep="_"),p))
  }

  list_plot

}



#-------------------------------------------------------------------------------------#
# CP 2.9 (4.5): Number of trees per plot for which species name is "blank" ------------
# Input: tree.csv and sapling.csv
# Output List of plots with number of trees/sapling for which species name is "blank"
#-------------------------------------------------------------------------------------#

# Number of trees with missing species
dim(tree[which(tree$tree_spp_scientific_name ==""),])[1]

# Prepare the output table

if (dim(tree[which(tree$tree_spp_scientific_name ==""),])[1]==0){
```

```
 output29t <- data.frame(cp="CP 2.9",cat="Species name missing",plot_id=9999,issue="passed successfully for trees (species always filled)",
"team" = 99,  stringsAsFactors = F)
 } else {
 # Prepare a table with missing species to join to the output table
 count_msp <- tree[tree$tree_spp_scientific_name == "",]
 count_msp <- count_msp[,c("plot_plot_id","tree_spp_scientific_name","unk_spp_tree_spp_other")]
 count_msp <- merge(count_msp, team_name, by = "plot_plot_id", all.x = T)
 count_msp$count <- 1
 count_msp <- aggregate(count~plot_plot_id, data=count_msp, FUN=sum, na.rm=TRUE)
 count_msp$issue <- paste(count_msp$count, "species missing in trees", sep = " ")
 count_msp$count <- NULL
 missing_sp <- data.frame(cp="CP 2.9",cat = "Species name missing",
                  plot_id = unique(tree[which(tree$tree_spp_scientific_name ==""),]$plot_plot_id))
 missing_sp <- merge(missing_sp, team_name, by.x = "plot_id", by.y = "plot_plot_id", all.x = T)
 missing_sp <- merge(missing_sp, count_msp, by.x = "plot_id", by.y = "plot_plot_id", all.x = T)
 output29t <- missing_sp[,c("cp", "cat", "plot_id", "issue", "team")]
 }

# Number of saplings with missing species
nb_pb=dim(saplings[which((saplings$sap_sp_scientific_name =="") & (saplings$sap_notes !="No sapling found") &
                  (!is.na(saplings$sap_dia))),])[1]
```

```
# Prepare the output table


if (nb_pb==0){
  output29s <- data.frame(cp="CP 2.9",cat="Species name missing in sapling",plot_id=9999,issue="passed successfully for saplings (species always filled)", "team" = 99,  stringsAsFactors = F)
  } else {
  # Prepare a table with missing species to join to the output table
  count_msp <- saplings[saplings$sap_sp_scientific_name == "",]
  count_msp <- count_msp[count_msp$sap_notes !="No sapling found",]
  count_msp <- count_msp[,c("plot_plot_id","sap_sp_scientific_name")]
  count_msp <- merge(count_msp, team_name, by = "plot_plot_id", all.x = T)
  count_msp$count <- 1
  count_msp <- aggregate(count~plot_plot_id, data=count_msp, FUN=sum, na.rm=TRUE)
  count_msp$issue <- paste(count_msp$count, "species missing in saplings", sep = " ")
  count_msp$count <- NULL
  missing_sp <- data.frame(cp="CP 2.9",cat = "species name missing in sapling",
                           plot_id = unique(saplings[which((saplings$sap_sp_scientific_name =="") & (saplings$sap_notes !="No sapling found")),]$plot_plot_id))
  missing_sp <- merge(missing_sp, team_name, by.x = "plot_id", by.y = "plot_plot_id", all.x = T)
  missing_sp <- merge(missing_sp, count_msp, by.x = "plot_id", by.y = "plot_plot_id", all.x = T)
```

```
  output29s <- missing_sp[,c("cp", "cat", "plot_id", "issue", "team")]}


# rbind outputs for both trees and sapling
output2.9 <- rbind(output29t, output29s)
#save outputs
write.csv(output2.9,
          file = "./QAQC_outputs/2.9_Trees and saplings species name missing.csv",
          row.names = F)


#-------------------------------------------------------------------------------#
# CP 2.10 (4.6): Number of trees with unknown species ---------------------------------
# Input: tree.csv and sapling.csv
# Output: List of plots with number of unknown tree/sapling
#-------------------------------------------------------------------------------#

# Number of trees with unknown species
dim(tree[which(tree$tree_spp_code == "UNK"),])[1]

# Prepare the output table
if (dim(tree[which(tree$tree_spp_code == "UNK"),])[1]==0){
```

```
 output210t <- data.frame(cp="CP 2.10",cat="Unknown species in trees",plot_id=9999,issue="passed successfully for trees (all species known)",
"team" = 99,  stringsAsFactors = F)

 } else {

 # Prepare a table with unknown species to join to the output table

 count_unk <- tree[tree$tree_spp_code == "UNK",]

 count_unk <- count_unk[,c("plot_plot_id","tree_spp_code")]

 count_unk <- merge(count_unk, team_name, by = "plot_plot_id", all.x = T)

 count_unk$count <- 1

 count_unk <- aggregate(count~plot_plot_id, data=count_unk, FUN=sum, na.rm=TRUE)

 count_unk$issue <- paste(count_unk$count, "unknown species in trees", sep = " ")

 count_unk$count <- NULL

 unk_sp <- data.frame(cp="CP 4.6",cat = "Unknown species in trees",

                      plot_id = unique(tree[which(tree$tree_spp_code == "UNK"),]$plot_plot_id))

 unk_sp <- merge(unk_sp, team_name, by.x = "plot_id", by.y = "plot_plot_id", all.x = T)

 unk_sp <- merge(unk_sp, count_unk, by.x = "plot_id", by.y = "plot_plot_id", all.x = T)

 output210t <- unk_sp[,c("cp", "cat", "plot_id", "issue", "team")]}



# Number of saplings with unknown species

nb_pb=dim(saplings[which((saplings$sap_sp_code == "UNK") | ((is.na(saplings$sap_sp_code)) & (saplings$sap_notes != "No sapling found"))
),])[1]
```

```
# Prepare the output table


if (nb_pb==0){
  output210s <- data.frame(cp="CP 2.10",cat="Unknown species in sapling",plot_id=9999,issue="passed successfully for saplings (all species known)", "team" = 99,  stringsAsFactors = F)
  } else {
  # Prepare a table with unknown species to join to the output table
  count_unk <- saplings[which((saplings$sap_sp_code == "UNK") | ((is.na(saplings$sap_sp_code)) & (saplings$sap_notes != "No sapling found"))),]
  count_unk <- count_unk[,c("plot_plot_id","sap_sp_code")]
  count_unk <- merge(count_unk, team_name, by = "plot_plot_id", all.x = T)
  count_unk$count <- 1
  count_unk <- aggregate(count~plot_plot_id, data=count_unk, FUN=sum, na.rm=TRUE)
  count_unk$issue <- paste(count_unk$count, "unknown species in sapling", sep = " ")
  count_unk$count <- NULL
  unk_sp <- data.frame(cp="CP 2.10",cat = "Unknown species in sapling",
                       plot_id = unique(saplings[which((saplings$sap_sp_code == "UNK") | ((is.na(saplings$sap_sp_code)) & (saplings$sap_notes != "No sapling found"))),]$plot_plot_id))
  unk_sp <- merge(unk_sp, team_name, by.x = "plot_id", by.y = "plot_plot_id", all.x = T)
  unk_sp <- merge(unk_sp, count_unk, by.x = "plot_id", by.y = "plot_plot_id", all.x = T)
  output210s <- unk_sp[,c("cp", "cat", "plot_id", "issue", "team")]
```

```
 }
#rbind outputs for both trees and saplings
output2.10 <- rbind(output210t, output210s)
write.csv(output2.10,
        file = "./QAQC_outputs/2.10_Unknown species in trees and saplings.csv",
        row.names = F)



#-------------------------------------------------------------------------------#
# CP 2.11 (4.7): Outliers in dbh-h relationship ----------------------------------------
# (to be updated by use of the power model h = a*(dbh)**(b) to check outliers)
# from confidence interval for individuals
# Input: tree.csv and saplings
# Output: List of plots with outliers in DBH-H relationship
#-------------------------------------------------------------------------------#
####### Looking for better approach for this check



# Remove stump/dead trees from the table tree
tree_live <- tree[which(tree$tree_status == 1),]
length(unique(tree_live$plot_plot_id))
```

```
x_lim = range(tree_live$tree_dia,na.rm=T)
y_lim = range(tree_live$tree_total_lgt,na.rm=T)

# Remove stump/dead trees from the table saplings
saplings_live <- saplings[which((!is.na(saplings$sap_dia)) & (!is.na(saplings$sap_total_lgt))),]

#complete datasets including trees and saplings
saplings_live$combined_plot_subplot<-paste(saplings_live$plot_plot_id,saplings_live$subplot_subp_id, sep="_")
tree_live$combined_plot_subplot<-paste(tree_live$plot_plot_id,tree_live$subplot_subp_id, sep="_")
treesap_live <- merge(tree_live, saplings_live,by ="combined_plot_subplot", all.x=T)


# plot dbh - h for saplings and trees
p_hd <- ggplot(treesap_live)
p_hd <- p_hd + xlim(0,max(treesap_live$tree_dia)) + ylim(0,max(treesap_live$tree_total_lgt))
p_hd <- p_hd + geom_point(aes(x=sap_dia, y=sap_total_lgt,col="Saplings"))
p_hd <- p_hd + geom_point(aes(x=tree_dia, y=tree_total_lgt,col="Trees"))
p_hd <- p_hd + xlab("tree diameter at breast height (cm)") + ylab("tree total height (m)")
p_hd
```

```
# Export plot dbh - h graph per team

# Merge team name

treesap_live <- merge(tree_live,saplings_live,by ="combined_plot_subplot", all.x=T)

names(treesap_live)[2] <- "plot_plot_id"

names(treesap_live)[4] <- "subplot_subp_id"

treesap_live <- merge(treesap_live, team_name, by = "plot_plot_id", all.x = T)

treesap_live <- treesap_live[which(treesap_live$team == 3),] #### Change here the team number ###


#vec <- unique(treesap_live$tree_lgt_method_label)

vec <- unique(treesap_live$plot_plot_id)

list_plot <- list()


#table(treesap_live$tree_lgt_method_label, treesap_live$team)

table(treesap_live$plot_plot_id, treesap_live$team)



for (i in 1:length(vec)){


  #estimate <- treesap_live[which(treesap_live$tree_lgt_method_label == vec[i]),]
```

```
  estimate <- treesap_live[which(treesap_live$plot_plot_id == vec[i]),]
  p_hd <- ggplot(estimate)
  p_hd <- p_hd + xlim(0,max(estimate$tree_dia)) + ylim(0,max(estimate$tree_total_lgt))
  p_hd <- p_hd + geom_point(aes(x=sap_dia, y=sap_total_lgt,col="Saplings",shape=tree_lgt_method_label,size=3))
  p_hd <- p_hd + geom_point(aes(x=tree_dia, y=tree_total_lgt,col="Trees",shape=tree_lgt_method_label,size=3))
  p_hd <- p_hd + xlab("tree diameter at breast height (cm)") + ylab("tree total height (m)")
  p_hd <- p_hd + facet_wrap(~team)
  p_hd <- p_hd + ggtitle(as.character(vec[i]))
  p_hd

  list_plot[[i]] = p_hd
}



# Save plots to tiff. Makes a separate file for each plot.
for (i in 1:length(vec)) {
  file_name = paste("./SaveGraphs/height_estimate/team3/", vec[i], ".tiff", sep="_") ## Change here the team number##
  ppi <- 70
  tiff(file_name, width=10*ppi, height=6*ppi, res=ppi)
  print(list_plot[[i]])
  dev.off()
```

```
}



#------------finding the Ht-DBH outliers using Ht & DBH threshold--------#
#-------------------------------------------------------------------------#

#create a table with the plots having outliers considering the following threshof for height and diameter
tree_outliers <- tree[which(tree$tree_total_lgt > 40 | tree$tree_dia > 120 |
                            tree$tree_total_lgt > 30 & tree$tree_dia < 20 |
                            tree$tree_dia >50 & tree$tree_total_lgt < 7),]
#plot the trees
#install.packages("ggrepel")
tree_outliers <- tree_outliers[!is.na(tree_outliers$tree_total_lgt),]
library(ggrepel)
p_hd <- ggplot(tree_outliers[!is.na(tree_outliers$tree_total_lgt),])
p_hd <- p_hd + xlim(0,max(tree_outliers$tree_dia)) + ylim(0,max(tree_outliers$tree_total_lgt, na.rm = T))
p_hd <- p_hd + geom_point(aes(x=tree_dia, y=tree_total_lgt,col="Trees"))
# p_hd <- p_hd+ geom_label_repel(aes(tree_dia, tree_total_lgt,
#                                label=tree_spp_scientific_name, fill=factor(tree_spp_scientific_name)))
p_hd <- p_hd + xlab("tree diameter at breast height (cm)") + ylab("tree total height (m)")
```

```
p_hd




# save the findings in output
if(dim(tree_outliers)[1] == 0){
  output2.11 <- data.frame(cp="CP 2.11",cat="Inconsistency H-D",plot_id=9999,issue="H-D check successfully, no outliers", "team" = 99,
stringsAsFactors = F)
  } else {
  tree_outliers <- merge(tree_outliers,team_name, by="plot_plot_id", all.x = T)
  tree_outliers$cp <- "CP 2.11"
  tree_outliers$cat <- "Inconsistency H-D"
  tree_outliers$plot_id <- tree_outliers$plot_plot_id
  tree_outliers$issue <- "Outlier in Height-Diameter relationship"


  output2.11 <- tree_outliers[,c("cp","cat","plot_id","issue", "team")]}
#save the output
write.csv(output2.11,
        file = "./QAQC_outputs/2.11_Possible outliers in H-DBH relationship.csv",
        row.names = F)
```

```
#-------------------------------------------------------------------------------#
# CP 2.12 (4.8): Percentage of tree lengths measured by estimation ----------
# Input: tree.csv and sapling.csv
# Outputs: List of plots in which 80-100% tree lengths were  measured by estimation]
#-------------------------------------------------------------------------------#

# Prepare a table with live trees and saplings
treesap_live <- treesap[(treesap$tree_status==1|
                         treesap$tree_status==2|
                         treesap$tree_status==99) &
                         (!is.na(treesap$tree_dia)&
                         !is.na(treesap$tree_total_lgt)),]

# select key variables only
treesap_liveKV <- treesap_live[,c("plot_plot_id", "tree_lgt_method_label")]
treesap_liveKV <- unique(treesap_liveKV)
#lets summerise the tree numbers by ht estimation method
treesap_liveKV$count <- 1
treesap_liveKVS <-aggregate(count~plot_plot_id, data=treesap_liveKV, sum)
```

```
names(treesap_liveKVS)[2]<-"count_sum"

#add the quanitity of estimation methods by plot_plot_id to unique plot and count list

treesap_liveKVSf <- merge(treesap_liveKV[,c("plot_plot_id","tree_lgt_method_label")],

                          treesap_liveKVS, by = "plot_plot_id", all.x=T)

treesap_liveKVSf$issue<-ifelse(treesap_liveKVSf$tree_lgt_method_label=="Estimate" & treesap_liveKVSf$count_sum==1, "tree lengths completely measurement by estimation",

                               "Not completely measured by estimation")


#incroporate the list of plots with necessary information in the output table

test <-treesap_liveKVSf[which(treesap_liveKVSf$issue=="tree lengths completely measurement by estimation"),]

test <- merge(test, team_name, by = "plot_plot_id")

test$cp <- "CP 2.12"

test$cat <- "tree length measurement method"

names(test)[1] <-"plot_id"


# put the result into output

if(dim(test)[1] == 0){

  output2.12 <- data.frame(cp="CP 2.12",cat="Category",plot_id=9999,issue="tree_length_measurement method Ok", "team" = 99, stringsAsFactors = F)

  } else {

  output2.12 <- test[,c("cp","cat","plot_id","issue", "team")]
```

```
 }


# save the output
write.csv(output2.12,
        file = "./QAQC_outputs/2.12_Plots where all tree lengths were estimation.csv",
        row.names = F)


#-------------------------------------------------------------------------------#
# CP 2.13 (4.9.1): LF Proportioning completed correctly ------------------------------
# Input: subplot_prop.csv and subplot.csv
# Output: List of plots and subplots in which lf proportion is not 100%
#-------------------------------------------------------------------------------#
# consider completely sampled subplots only
subplot_prop_SSt <- merge(subplot_prop, subplot_info[,c("plot_plot_id", "subp_id",
                "subp_details_subp_status")],by.x=c("plot_plot_id", "subplot_subp_id"),
                by.y=c("plot_plot_id", "subp_id"), all.x=T) #subplot_prop_SSt = subplot status
subplot_prop_SSt <- subplot_prop_SSt[subplot_prop_SSt$subp_details_subp_status==1,]

# summarize the LF proportions by subplot for each S, M and L plot to see whether their sum is 100
lf_prop_prob <- aggregate(data=subplot_prop_SSt, cbind(lf_s_percent, lf_l_percent)~plot_plot_id+subplot_subp_id,
                    sum)
```

```
library(dplyr)

lf_prob <- filter(lf_prop_prob, lf_s_percent != 100 | lf_l_percent != 100)

lf_prob$issue <- ifelse(lf_prob$lf_s_percent<100 | lf_prob$lf_l_percent<100,

                paste("LF-Subp proportion is incomplete: subp_id-",lf_prob$subplot_subp_id), "LF-subp proportion is complete")


# prepare the table of non sampled plot and add it to output

if(dim(lf_prob)[1] == 0){

  output2.13 <- data.frame(cp="CP 2.13",cat="LF-Subplot proportioning",plot_id=9999,issue="LF Proportioning is OK", "team" = 99,
stringsAsFactors = F)

  } else {

  lf_prob$cp <- "CP 4.9.1"

  lf_prob$cat <- "LF-Subplot Proportioning"

  lf_prob <- merge(lf_prob, team_name, by="plot_plot_id", all.x=T)

  names(lf_prob)[1] <- "plot_id"

  output2.13 <- lf_prob[,c("cp","cat","plot_id","issue", "team")]}


# save the output

write.csv(output2.13,

        file = "./QAQC_outputs/2.13_LF proportions not completed.csv", row.names = F)


#-------------------------------------------------------------------------------#
```

```
# CP 2.14 (4.9.2): Tree WO missed in the tree list ------------------------------------
# Note: Trees >10cm diameter and within 8 m distance from centre of plot and greater than
# 30 cm diameter wthin 19 m from plot centre recorded as witness object but not
# measured in plot
# Input: wo.csv and tree.csv
# Output: List of WO trees missed in tree table
#-------------------------------------------------------------------------------------#

# Plots with WO as Tree that fit within the distance - dia thresholds
wo_sub <- wo
wo_sub$new_id <- paste(wo_sub$plot_plot_id, wo_sub$subplot_subp_id, sep = "_")
wo_sub <- wo_sub[which(wo_sub$wo_type_label == "Tree" & wo_sub$wo_dia > 10),]
wo_sub <- wo_sub[which(wo_sub$wo_dia <= 30 & wo_sub$wo_dist <= 8 | wo_sub$wo_dia > 30 & wo_sub$wo_dist <= 19),]
#wo_sub <- wo_sub[,c("plot_plot_id", "new_id", "wo_type_label", "plot_wo_notes", "wo_dia", "wo_dist")]
wo_sub <- wo_sub[,c("plot_plot_id", "new_id")]
wo_sub <- unique(wo_sub)

# Prepare tree table
no_tree_plot <- tree
no_tree_plot$new_id <- paste(tree$plot_plot_id, tree$subplot_subp_id, sep = "_")
no_tree_plot <- no_tree_plot[,c("plot_plot_id","new_id")]
```

```
no_tree_plot <- unique(no_tree_plot)

# Find trees in WO table not in Tree Table
no_tree <- wo_sub[!wo_sub$new_id %in% no_tree_plot$new_id,]
no_tree <- unique(no_tree)
no_tree <- merge(no_tree, team_name, by = "plot_plot_id")
names(no_tree)[1] <- "plot_id"
no_tree_count <- dim(no_tree)

# prepare the table of non sampled plot and add it to output
if(dim(no_tree)[1] == 0){
 output2.14 <- data.frame(cp="CP",cat="WO data missing",plot_id=9999,issue="WO trees OK", "team" = 99,  stringsAsFactors = F)
 } else {
 no_tree$cp <- "CP 214"
 no_tree$cat <- "WO data missing"
 no_tree$issue <- paste("WO Tree missing in Plot_Subplot:", no_tree$new_id, sep = " ")
 output2.14 <- no_tree[,c("cp","cat","plot_id","issue", "team")]
 }

# save the output
write.csv(output2.14,
```

```
        file = "./QAQC_outputs/2.14_WO missed in tree list.csv",
        row.names = F)


#-------------------------------------------------------------------------------------#
# CP 2.15 (4.11): Dead standing trees having < 1.3m total length ----------------------
# Input: tree.csv
# Output: List of plots having dead standing trees <1.3 m
#-------------------------------------------------------------------------------------#


dead_tree <- tree
dead_tree <- dead_tree[which(dead_tree$tree_status_label == "Dead standing tree" & dead_tree$tree_total_lgt <= 1.3),]
dim(dead_tree)[1]


# prepare the table of non sampled plot and add it to output
if(dim(dead_tree)[1] == 0){
  output2.15 <- data.frame(cp="CP 215",cat="Dead standing tree Ht",plot_id=9999,issue="dead standing tree heights OK", "team" = 99,
stringsAsFactors = F)
} else {
  dead_tree$cp <- "CP 215"
  dead_tree$cat <- "dead standing or stump identification"
  dead_tree$issue <- "dead standing trees < 1.3m height"
```

```
  output2.15 <- no_tree[,c("cp","cat","plot_id","issue", "team")]}


# save the output
write.csv(output2.15,
          file = "./QAQC_outputs/2.15_Dead standing trees less than 1.3m total length.csv",
          row.names = F)



#-------------------------------------------------------------------------------#
# CP 2.16 (4.14): Trees with no LF information ----------------------------------------
# Input: tree.csv
# Output: List of plots having trees missing lf_id
#-------------------------------------------------------------------------------#

# This should be corrected in OpenForis before export
tree_no_lf <- as.data.frame(unique(treesap[which(is.na(treesap$tree_lfid)),]))
tree_no_lf <- tree_no_lf[,c("plot_plot_id", "subplot_subp_id")]
tree_no_lf <- unique(tree_no_lf)
tree_no_lf <- merge(tree_no_lf, team_name, by = "plot_plot_id")

# Prepare the output table
```

```
if(dim(tree_no_lf)[1] == 0){
  output2.16 <- data.frame(cp="CP 2.16",cat="LF ID",plot_id=9999,issue="All trees and saplings have lf_id", "team" = 99,  stringsAsFactors = F)
  } else {
    tree_no_lf$cp <- "CP 2.16"
    tree_no_lf$cat <- "Tree and sapling missing lf_id"
    tree_no_lf$issue <- paste("No lf_id in subplot", tree_no_lf$subplot_subp_id, sep=" ")
    names(tree_no_lf)[1] <- "plot_id"
    output2.16 <- tree_no_lf[,c("cp","cat","plot_id","issue", "team")]}

# save the output
write.csv(output2.16,
          file = "./QAQC_outputs/2.16_Trees and sapling missing LF id.csv",
          row.names = F)



#-------------------------------------------------------------------------------------#
# CP 2.17 (4.15): Consistency between tree density and crown cover ----------------
# Note: Compare crown cover with tree and sapling stem density to see plots having
# higher crown cover but no/very less trees or vice-versa
# Input: lf.csv and tree.csv
# Output: List of plots and LF with inconsistent stem density and crown cover
```

```
#-------------------------------------------------------------------------------------#


ts_live <- treesap[which(treesap$tree_status == 1|
                         treesap$tree_status==99),] # ts_live=tree and sapling live
ts_live$stem_count <- 1
#expand the stem number from S and M plot to L plot
ts_live$stem_countF <- ifelse(ts_live$tree_dia<10, ts_live$stem_count*(19^2/2.5^2),
                              ifelse(ts_live$tree_dia <=30, ts_live$stem_count*(19^2/8^2),
                                     ts_live$stem_count))
# sum the stem number by plot, subplot and tree_lf
ts_liveSpl <- aggregate(stem_countF~plot_plot_id+subplot_subp_id+tree_lfid, data=ts_live, sum)
# add the sampled area by plot and subplot
ts_liveSplF <- merge(ts_liveSpl, subplot_prop[,c("plot_plot_id", "subplot_subp_id","lf_subp_lfno","lf_l_percent")],
                                    by.x=c("plot_plot_id","subplot_subp_id", "tree_lfid"),
                                    by.y=c("plot_plot_id","subplot_subp_id", "lf_subp_lfno"),
                                    all.x=T)
ts_liveSplF$ai <- ((ts_liveSplF$lf_l_percent/100)*pi*19^2)/10000
#compute the tree density at plot and lf level
ts_liveSplFf <- aggregate(data=ts_liveSplF, cbind(stem_countF, ai)~plot_plot_id+tree_lfid, sum)
ts_liveSplFf$stem_density <- ts_liveSplFf$stem_countF/ts_liveSplFf$ai
# add the canopy cover
```

```
ts_liveSplFf <- merge(ts_liveSplFf, lf[,c("plot_plot_id", "lf_id", "lf_details_cover_max",
                                          "lf_details_cover_min")],
                      by.x=c("plot_plot_id", "tree_lfid"),
                      by.y=c("plot_plot_id", "lf_id"), all.x=T)
# compute avg CC
ts_liveSplFf$cc_mean <- (ts_liveSplFf$lf_details_cover_max+ts_liveSplFf$lf_details_cover_min)/2
#delete the infinitity cells in stem density
ts_liveSplFfF <- ts_liveSplFf[ts_liveSplFf$stem_density!=Inf,]
# identify inconsistencies
ts_liveSplFfF$issue <- ifelse(ts_liveSplFfF$cc_mean==0 & ts_liveSplFfF$stem_density>0,
                       paste("CC is 0 but stem density is", round(ts_liveSplFfF$stem_density,0)),
                       "Passed successfully")


ts_cc_incons <- ts_liveSplFfF[ts_liveSplFfF$issue!="Passed successfully",]


# Prepare the output table
if(dim(ts_cc_incons)[1] == 0){
  output2.17 <- data.frame(cp="CP 2.17",cat="Inconsistency between Canopy Cover and stem density",
                           plot_id=9999,issue="No inconsistencies found", "team" = 99,  stringsAsFactors = F)
} else {
  ts_cc_incons$cp <- "CP 2.17"
```

```
  ts_cc_incons$cat <- "Inconsistency between Canopy Cover and stem density"
  ts_cc_incons <- merge(ts_cc_incons, team_name, by="plot_plot_id", all.x=T)
  names(ts_cc_incons)[1] <- "plot_id"
  output2.17 <- ts_cc_incons[,c("cp","cat","plot_id","issue", "team")]}



# save the output
write.csv(output2.17,
          file = "./QAQC_outputs/2.17_Inconsistency between crown cover and stem density.csv",
          row.names = F)

#-------------------------------------------------------------------------------#
# CP 2.18 (4.18): Inconsistency in the distribution of sapling and tree ----------
# distribution per team number
# Input: tree.csv, sapling.csv, plot.csv
# Output: Visual intepretation of the tree-sapling distribution based on Ht-DBH
#-------------------------------------------------------------------------------#
# Read the sapling and plot info data
sap <- read.csv("./BFI-DB/sapling.csv", header = T, stringsAsFactors = F)
plot_info <- read.csv("./BFI-DB/plot.csv", header=T, stringsAsFactors = FALSE)
```

```
# prepare sapling table
#1. check NA in sapling (in 5 columns when sapling_id == 0, which seem to mean that there are no sapling in the plot)
temp=table(
  sap[(sap$sap_sp_scientific_name==""|is.na(sap$sap_bearing)|
        is.na(sap$sap_dist)|is.na(sap$sap_dia)|is.na(sap$sap_total_lgt))
      &sap$sapling_id!=0,"plot_plot_id"])
QC_g=as.data.frame(temp[temp>1]) #plots which at least 2 missing data in saplings
QC_g$type="at least 2 saplings with missing data"

#dbh-h of saplings together with the trees
#1. remove the NA
sap_QC=sap[!is.na(sap$sap_dia)&!is.na(sap$sap_total_lgt)
           &sap$sapling_id!=0,];nrow(sap_QC)

#2. try to assign a lf to each sapling

sap_QC=
  merge(sap_QC,subplot_prop[subplot_prop$lf_s_percent==100,
                            c("plot_plot_id","subplot_subp_id","lf_subp_lfno")],
        by = c("plot_plot_id","subplot_subp_id"),
        all.x=TRUE)
```

```
unique(sap_QC[is.na(sap_QC$lf_subp_lfno),1:2])#this is the list of the supbplot for which no
# info about the land feature of s plot is provided but sapling have been measured

QC_h=sort(table(sap_QC[is.na(sap_QC$lf_subp_lfno),1]),decreasing = TRUE)
sap_QC=sap_QC[!is.na(sap_QC$lf_subp_lfno),]

sap_QC= merge(
  sap_QC,
  lf[,c("plot_plot_id","lf_id","nlcl")],
  by.x=c("plot_plot_id","lf_subp_lfno"),
  by.y= c("plot_plot_id","lf_id"),
  all.x=TRUE)


#3, create a joint table of tree and saplings
sap_QC=sap_QC[,c("plot_plot_id","subplot_subp_id","sapling_id",
                 "lf_subp_lfno",
                 #"tree_status", #this is not present
                 "sap_sp_scientific_name",
```

```
              "sap_dist","sap_dia","sap_total_lgt")]
names(sap_QC)=c("plot_plot_id","subplot_subp_id","tree_id",
              "tree_lfid","tree_spp_scientific_name",
              "tree_dist","tree_dia","tree_total_lgt")

# select only live trees and saplings
sap_QC$tree_status=1;sap_QC$type="sapling"
tree_sub=tree[,c("plot_plot_id","subplot_subp_id","tree_id",
              "tree_lfid","tree_status","tree_spp_scientific_name",
              "tree_dist","tree_dia","tree_total_lgt")]
tree_sub$type="tree"
head(tree_sub);head(sap_QC)
#bind the two tables
tree_sap=rbind( tree_sub,
             sap_QC)
summary(tree_sap)

#ADD team no and location_zone
tree_sap=merge(tree_sap,
             plot_info[,c("plot_id","location_zone","plot_details_plot_teamnbr")],
             by.x="plot_plot_id",
```

```
            by.y= "plot_id",
            all.x=TRUE)

#add zone from another data frame
tree_sapx=merge(tree_sap,plot_zone[,c("Plot_ID", "Zone")],
            by.x="plot_plot_id", by.y="Plot_ID", all.x=T)

tree_sap= merge(
  tree_sap,
  lf[,c("plot_plot_id","lf_id","nlcl")],
  by.x=c("plot_plot_id","tree_lfid"),
  by.y= c("plot_plot_id","lf_id"),
  all.x=TRUE)


#GRAPHIC IT
# a function for making quick visual analysis of the dbh-h distribution in trees and saplings
plot_team= function (team=unique(tree_sap$plot_details_plot_teamnbr),
                  lf=unique(tree_sap$nlcl)) {
  data= tree_sap[tree_sap$tree_status==1 &
                tree_sap$plot_details_plot_teamnbr %in% team &
```

```
                tree_sap$nlcl %in% lf,]
  p= ggplot(data,
           aes(x=tree_dia, y=tree_total_lgt,
               color=type)) +
    geom_point(cex=1) +
    xlab("dbh") + ylab("height (m)")+
    theme(plot.title = element_text(size=22))
  if(missing(team)){
    p=p+facet_wrap(~plot_details_plot_teamnbr)+
      ggtitle("Dbh against Height by Team")
  } else {
    if (missing(lf)){
      p=p+facet_wrap(~nlcl)+
        ggtitle(paste("Dbh against Height of all lf by plot. Team ", team))
    } else {
      p=p+facet_wrap(~plot_plot_id)+
        ggtitle(paste("Dbh against Height of",lf,"lf by plot. Team ", team))
    }
  }
  p
}
```

```
plot_plot=function (plot_plot_id,full=TRUE){
  data=tree_sap[tree_sap$tree_status==1&
                  tree_sap$plot_plot_id==plot_plot_id,]
  lf=unique(data$nlcl)
  team= unique(data$plot_details_plot_teamnbr)
  p=ggplot(data,
           aes(x=tree_dia, y=tree_total_lgt,
               color=type)) +
    geom_point(cex=3) +
    #geom_text(hjust=0, vjust=-1)+
    xlab("dbh (cm)") + ylab("height (m)")+
    ggtitle(paste("Dbh against Height of plot ",plot_plot_id,", in ",lf, ". Team:",team))+
    theme(plot.title = element_text(size=22))
  if (full==TRUE){
    p=p+ facet_wrap(~subplot_subp_id)
  }
  p
}
```

```
#let's look all the team together
plot_team()

#let's look in detail team 7
plot_team(7)

#team 7 in RS is strange

# now let's look in detail what they have done in RS
plot_team(7,"RS")



#now let's look a specific plot
plot_plot(627)

###LET"S GO
plot_team(10)
plot_plot(1101,full=FALSE)
```

```
#-------------------------------------------------------------------------------#
# CP 3: Standing dead trees related checks ------------------------------------------
#-------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------#
# CP 3.1 Outliers (extreme values) in standing dead tree DBH ------------------------
# Input: tree.csv
# Output: List of plots with outliers (extreme values) in DBH of standing dead trees
#-------------------------------------------------------------------------------#
# Dead Tree DBH with key variables
treesapKV <- treesap[,c("plot_plot_id","subplot_subp_id","tree_lfid",
                        "tree_status","tree_spp_scientific_name","tree_dia",
                        "tree_total_lgt")]
# Trees missing DBH, DBH <2cm and missing length are filtered out
# Missing values and mistakes are eliminated at scientific_name, tree_dia, length
treesapKVf <- treesapKV[!is.na(treesapKV$tree_dia) &
                          !treesapKV$tree_dia<2 &
                          !is.na(treesapKV$tree_total_lgt) &
                          !treesapKV$tree_total_lgt<=1.3,]


### A table of statistics (mean and sd) are done
```

```
f31 <- function(x) c(n = length(x), Mean = mean(x), SD = sd(x))
BasicSat <- aggregate(tree_dia ~ tree_status, data=treesapKVf[treesapKVf$tree_status==2,], FUN=f31)
BasicSat$n<-BasicSat$tree_dia[,1]
BasicSat$Mean<-BasicSat$tree_dia[,2]
BasicSat$SD<-BasicSat$tree_dia[,3]

### Tree database and Statistical database are merged by sientific name
treesapKVfSt<-merge(treesapKVf,BasicSat[,c("tree_status", "n", "Mean", "SD")],by.x="tree_status"
                    ,by.y="tree_status",all.x=T)

#select the dead trees only
treesapKVfStF <- treesapKVfSt[treesapKVfSt$tree_status==2,]

### The Z value is estimated and Possible Outliers are identified
# Z is estimated
treesapKVfStF$Z<-((treesapKVfStF$tree_dia-treesapKVfStF$Mean)/
                  treesapKVfStF$SD)
# Possible Outliers are identified
treesapKVfStF$PO<-ifelse(abs(treesapKVfStF$Z)>4,1,0)

### Possible Outliers are summarized and saved
```

```
# A Database of possible OL is created
treesapKVfStF$issue <- paste("DeadTree-DBH:", treesapKVfStF$tree_dia,
                             "Mean:",round(treesapKVfStF$Mean,1),
                             "Z:",round(treesapKVfStF$Z,1),sep = " ")
treesapKVfStF$cp <- "CP 3.1"
treesapKVfStF$cat <- "Outliers in dead tree DBH"
treesapKVfStF <- merge(treesapKVfStF, team_name, by="plot_plot_id", all.x=T)
names(treesapKVfStF)[1] <- "plot_id"

# The database of Possible OL is created
output3.1 <- treesapKVfStF[treesapKVfStF$PO==1, c("cp", "cat", "plot_id", "issue", "team")]

# save the output
write.csv(output3.1,
          file = "./QAQC_outputs/3.1_Possible outlier in dead tree DBH.csv",
          row.names = F)

#-------------------------------------------------------------------------------#
# CP 3.2: Outliers (extreme values) in standing dead tree total length ----------
# Input: tree.csv
# Output: List of plots with outliers (extreme values) in total length of standing
```

```
# dead trees
#-------------------------------------------------------------------------------#
# Work with the database with the standing dead tree related key variables only
treesapKV <- treesap[,c("plot_plot_id","subplot_subp_id","tree_lfid",
                        "tree_status","tree_spp_scientific_name","tree_dia",
                        "tree_total_lgt")]
# Trees missing DBH, DBH <2cm and missing length are filtered out
# Missing values and mistakes are eliminated at scientific_name, tree_dia, length
treesapKVf <- treesapKV[!is.na(treesapKV$tree_dia) &
                         !treesapKV$tree_dia<2 &
                         !is.na(treesapKV$tree_total_lgt) &
                         !treesapKV$tree_total_lgt<=1.3,]

### A table of statistics (mean and sd) are done
f32 <- function(x) c(n = length(x), Mean = mean(x), SD = sd(x))
BasicSat <- aggregate(tree_total_lgt ~ tree_status, data=treesapKVf[treesapKVf$tree_status==2,], FUN=f32)
BasicSat$n<-BasicSat$tree_total_lgt[,1]
BasicSat$Mean<-BasicSat$tree_total_lgt[,2]
BasicSat$SD<-BasicSat$tree_total_lgt[,3]

### Tree database and Statistical database are merged by sientific name
```

```
treesapKVfSt<-merge(treesapKVf,BasicSat[,c("tree_status", "n", "Mean", "SD")],by.x="tree_status"
                ,by.y="tree_status",all.x=T)

#select the dead trees only
treesapKVfStF <- treesapKVfSt[treesapKVfSt$tree_status==2,]

### The Z value is estimated and Possible Outliers are identified
# Z is estimated
treesapKVfStF$Z<-((treesapKVfStF$tree_total_lgt-treesapKVfStF$Mean)/
                treesapKVfStF$SD)
# Possible Outliers are identified
treesapKVfStF$PO<-ifelse(abs(treesapKVfStF$Z)>4,1,0)

### Possible Outliers are summarized and saved
# A Database of possible OL is created
treesapKVfStF$issue <- paste("DeadTree-H:", treesapKVfStF$tree_total_lgt,
                        "Mean:",round(treesapKVfStF$Mean,1),
                        "Z:",round(treesapKVfStF$Z,1),sep = " ")
treesapKVfStF$cp <- "CP 3.2"
treesapKVfStF$cat <- "Outliers in dead tree length"
treesapKVfStF <- merge(treesapKVfStF, team_name, by="plot_plot_id", all.x=T)
```

```
names(treesapKVfStF)[1] <- "plot_id"


# The database of Possible OL is created
output3.2 <- treesapKVfStF[treesapKVfStF$PO==1,
                           c("cp", "cat", "plot_id", "issue", "team")]


# save the output
write.csv(output3.2,
          file = "./QAQC_outputs/3.2_Possible outlier in dead tree total length.csv",
          row.names = F)


#-------------------------------------------------------------------------------#
# CP 4: Stumps related checks ---------------------------------------------------
#-------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------#
# CP 4.1 (14.): Outliers (extreme values) in stump (dead and live) height -------
# Input: tree.csv
# Output: List of plots with number of stump having height outliers
#-------------------------------------------------------------------------------#
###prepare the data
```

```
stump <-tree[which(tree$tree_status_label=="Stump (alive)"|
                   tree$tree_status_label=="Stump (dead)"),]

# Select the key variables
summpKV <- stump[,c("plot_plot_id", "tree_lfid", "tree_spp_scientific_name",
                    "tree_status", "tree_htdmp", "tree_total_lgt")]

#number of stump missing height data
length(is.na(summpKV$tree_total_lgt))

#Since, the number missing height data is very hight, add the stump diameter
#measurement height
summpKV$tree_total_lgt <- ifelse(is.na(summpKV$tree_total_lgt), summpKV$tree_htdmp,
                                 summpKV$tree_total_lgt)

###identify the stumps having height>1.3m
stumpKvOT <- summpKV[summpKV$tree_total_lgt>1.3,]

### Prepare the data for output
stumpKvOT$cp <- "CP 4.1"
stumpKvOT$issue <- paste("Stump length", stumpKvOT$tree_total_lgt, "cm")
```

```
stumpKvOT<- merge(stumpKvOT, team_name, by="plot_plot_id", all.x= TRUE)
stumpKvOT$cat <- "Stump total length > 1.3m"
names(stumpKvOT)[1] <- "plot_id"

# prepare the output table
if(dim(stumpKvOT)[1] == 0){
  output4.1 <- data.frame(cp="CP 4.1",cat="Stump total length > 1.3m",plot_id=NA,
                          issue="No stumps have length > 1.3m", team = NA,
                          stringsAsFactors = F)} else {
  output4.1 <- stumpKvOT[,c("cp","cat","plot_id","issue", "team")]}

# save the output
write.csv(output4.1,
          file = "./QAQC_outputs/4.1_Possible outlier in stump total length.csv",
          row.names = F)

#-------------------------------------------------------------------------------#
# CP 4.2 (14.): Outliers (extreme values) in stump (dead and live) diameter ----------
# Input: tree.csv
# Output: List of plots with number of stump having outliers in diameter
#-------------------------------------------------------------------------------#
```

```
### Data preparation
# Select the key variables
stumpKV <- stump[,c("plot_plot_id", "tree_lfid", "tree_spp_scientific_name",
                    "tree_status", "tree_dia")]

### Identification of outliers
## A table of statistics (mean and sd) are done
f42 <- function(x) c(n = length(x), Mean = mean(x), SD = sd(x))
BasicSat <- aggregate(tree_dia ~ tree_status,
                      data=stumpKV[stumpKV$tree_status==3|stumpKV$tree_status==4,], FUN=f42)
BasicSat$n<-BasicSat$tree_dia[,1]
BasicSat$Mean<-BasicSat$tree_dia[,2]
BasicSat$SD<-BasicSat$tree_dia[,3]

### stump database and Statistical database are merged by tree status
stumpKVSt<-merge(stumpKV,BasicSat[,c("tree_status", "n", "Mean", "SD")],
                 by="tree_status", all.x=T)

### The Z value is estimated and Possible Outliers are identified
# Z is estimated
stumpKVSt$Z<-((stumpKVSt$tree_dia-stumpKVSt$Mean)/stumpKVSt$SD)
```

```
# Possible Outliers are identified
stumpKVSt$PO<-ifelse(abs(stumpKVSt$Z)>4,1,0)

### Possible Outliers are summarized and saved
# A Database of possible OL is created
stumpKVSt$issue <- paste("Status:",stumpKVSt$tree_status,
                         "Stump-DBH:", stumpKVSt$tree_dia,
                         "Mean:",round(stumpKVSt$Mean,1),
                         "Z:",round(stumpKVSt$Z,1),sep = " ")
stumpKVSt$cp <- "CP 4.2"
stumpKVSt$cat <- "Outlier in stump DBH"
stumpKVSt <- merge(stumpKVSt, team_name, by="plot_plot_id", all.x=T)
names(stumpKVSt)[1] <- "plot_id"

# Prepare the output table
if(dim(stumpKVSt[stumpKVSt$PO==1,])[1] == 0){
  output4.2 <- data.frame(cp="CP 4.2",cat="Outlier in stump DBH",plot_id=NA,
                          issue="No outlier is found in Stump DBH", team = NA,
                          stringsAsFactors = F)} else {
  output4.2 <- stumpKVSt[stumpKVSt$PO==1,][,c("cp","cat","plot_id","issue", "team")]}
```

```
# save the output
write.csv(output4.2,
          file = "./QAQC_outputs/4.2_Possible outlier in stump diameter.csv",
          row.names = F)


#-------------------------------------------------------------------------------#
# CP 5: Bamboo related checks ----------------------------------------------------
#-------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------#
# CP 5.1: Outliers (extreme values) in Bamboo DBH -------------------------------
# Input: bamboo.csv
# Output: List of plots with outliers (extreme values) in bamboo DBH
#-------------------------------------------------------------------------------#
### Data preparation
# Bamboo DB with key variables
bambooKV <- bamboo[,c("plot_plot_id","subplot_subp_id","bamboo_id",
                      "bamboo_sp_scientific_name","bamboo_dia","bamboo_lgt")]


# Missing values are eliminated at scientific_name and tree_dia
```

```
bambooKVf <- bambooKV[!is.na(bambooKV$bamboo_dia)&
                      !is.na(bambooKV$bamboo_sp_scientific_name)&
                      bambooKV$bamboo_sp_scientific_name!="",]

### A table of statistics (mean and sd) are done
f51 <- function(x) c(n = length(x), Mean = mean(x), SD = sd(x))
BasicSat <- aggregate(bamboo_dia ~ bamboo_sp_scientific_name,data=bambooKVf, FUN=f51)
BasicSat$n<-BasicSat$bamboo_dia[,1]
BasicSat$Mean<-BasicSat$bamboo_dia[,2]
BasicSat$SD<-BasicSat$bamboo_dia[,3]

### Tree database and Statistical database are merged by sientific name
bambooKVfSt<-merge(bambooKVf, BasicSat[,c("bamboo_sp_scientific_name","n","Mean", "SD")],
                   by="bamboo_sp_scientific_name", all.x=T)
bambooKVfStF<- bambooKVfSt[bambooKVfSt$n>3,]

### The Z value is estimated and Possible Outliers are identified
# Z is estimated
bambooKVfStF$Z<-((bambooKVfStF$bamboo_dia-bambooKVfStF$Mean)/bambooKVfStF$SD)

# Possible Outliers are identify
```

```
bambooKVfStF$PO<-ifelse(abs(bambooKVfStF$Z)>4,1,0)


### Possible Outliers are summarized and saved
# A Database of possible OL is created
bambooKVfStF$issue <- paste("Spp:", bambooKVfStF$bamboo_sp_scientific_name,
                            "DBH:", bambooKVfStF$bamboo_dia,
                            "Mean:",round(bambooKVfStF$Mean,1),
                            "Z:",round(bambooKVfStF$Z,1),sep = " ")
bambooKVfStF$cp <- "CP 5.1"
bambooKVfStF$cat <- "Outlier in bamboo DBH"
bambooKVfStF <- merge(bambooKVfStF, team_name, by="plot_plot_id", all.x=T)
names(bambooKVfStF)[1] <- "plot_id"

# Prepare the output table
if(dim(bambooKVfStF[bambooKVfStF$PO==1,])[1] == 0){
  output5.1 <- data.frame(cp="cp 5.1",cat="Outlier in Bamboo DBH",plot_id=NA,
                          issue="No outlier is found in Bamboo DBH", team = NA,
                          stringsAsFactors = F)} else {
  output5.1 <- bambooKVfStF[bambooKVfStF$PO==1,][,c("cp","cat","plot_id","issue", "team")]}

# save the output
```

```
write.csv(output5.1, file = "./QAQC_outputs/5.1_Possible outlier in bamboo diameter.csv",
          row.names = F)


#-------------------------------------------------------------------------------------#
# CP 5.2: Outliers (extreme values) in Bamboo Height ---------------------------------
# Input: bamboo.csv
# Output: List of plots with outliers (extreme values) in bamboo Height
#-------------------------------------------------------------------------------------#
### Data preparation
# Bamboo DB with key variables
bambooKV <- bamboo[,c("plot_plot_id","subplot_subp_id","bamboo_id",
                      "bamboo_sp_scientific_name","bamboo_dia","bamboo_lgt")]

# Missing values are eliminated at scientific_name and bamboo height
bambooKVf <- bambooKV[!is.na(bambooKV$bamboo_lgt)&
                        !is.na(bambooKV$bamboo_sp_scientific_name)&
                        bambooKV$bamboo_sp_scientific_name!="",]

### A table of statistics (mean and sd) are done
f52 <- function(x) c(n = length(x), Mean = mean(x), SD = sd(x))
BasicSat <- aggregate(bamboo_lgt ~ bamboo_sp_scientific_name,data=bambooKVf, FUN=f52)
```

```
BasicSat$n<-BasicSat$bamboo_lgt[,1]
BasicSat$Mean<-BasicSat$bamboo_lgt[,2]
BasicSat$SD<-BasicSat$bamboo_lgt[,3]

### Bamboo database and Statistical database are merged by sientific name
bambooKVfSt<-merge(bambooKVf, BasicSat[,c("bamboo_sp_scientific_name","n","Mean", "SD")],
                   by="bamboo_sp_scientific_name", all.x=T)
bambooKVfStF<- bambooKVfSt[bambooKVfSt$n>3,]

### The Z value is estimated and Possible Outliers are identified
# Z is estimated
bambooKVfStF$Z<-((bambooKVfStF$bamboo_lgt-bambooKVfStF$Mean)/bambooKVfStF$SD)

# Possible Outliers are identify
bambooKVfStF$PO<-ifelse(abs(bambooKVfStF$Z)>4,1,0)

### Possible Outliers are summarized and saved
# A Database of possible OL is created
bambooKVfStF$issue <- paste("Spp:", bambooKVfStF$bamboo_sp_scientific_name,
                            "lgt:", bambooKVfStF$bamboo_lgt,
                            "Mean:",round(bambooKVfStF$Mean,1),
```

```
                    "Z:",round(bambooKVfStF$Z,1),sep = " ")
bambooKVfStF$cp <- "CP 5.2"
bambooKVfStF$cat <- "Outlier in bamboo lgt"
bambooKVfStF <- merge(bambooKVfStF, team_name, by="plot_plot_id", all.x=T)
names(bambooKVfStF)[1] <- "plot_id"

# Prepare the output table
if(dim(bambooKVfStF[bambooKVfStF$PO==1,])[1] == 0){
  output5.2 <- data.frame(cp="CP 5.2",cat="Outlier in Bamboo length",plot_id=NA,
                    issue="No outlier is found in Bamboo length", team = NA,
                    stringsAsFactors = F)} else {
  output5.2 <- bambooKVfStF[bambooKVfStF$PO==1,][,c("cp","cat","plot_id","issue", "team")]}

# save the output
write.csv(output5.2, file = "./QAQC_outputs/5.2_Possible outlier in bamboo total length.csv",
          row.names = F)



#-------------------------------------------------------------------------------------#
# CP 6 (11.1) Consistency in time interval between arrival and departure ----------
# (Note: Volume of work by zone (times are manually entered, time data may not be real)
```

```
# Input: plot.csv
# Output: List of plots with inconsistent time interval between arrival and departure
#-------------------------------------------------------------------------------#
time_on_plot <- plot_info[,c("plot_id", "plot_record_plot_date",
                             "plot_record_time_dep",
                             "plot_record_time_start",
                             "plot_end_end_time")]
t_names <- c("plot_plot_id", "p_date", "p_time", "s_time", "e_time")
names(time_on_plot) <- t_names

# Extract Subplot time
time_on_subp <- subplot_info[,c("plot_plot_id", "subp_id",
                                "sp_start_time_collection")]
names(time_on_subp)[3] <- "sp_time"

# Extract LF time
time_on_lf <- lf[,c("plot_plot_id", "lf_start_time_collection")]
names(time_on_lf)[2] <- "lf_time"
time_on_lf <- aggregate(lf_time~plot_plot_id, data = time_on_lf, FUN = max)

# # Merge
```

```
# time_on_plot <- merge(time_on_plot, time_on_lf, by = "plot_plot_id")
# time_on_plot <- merge(time_on_plot, time_on_subp, by = "plot_plot_id")
# time_on_plot <- unique(time_on_plot)

# Transform date/time fields into data/time format
time_on_plot$p_date <- strptime(time_on_plot$p_date, "%d/%m/%Y")
time_on_plot$p_time <- paste(time_on_plot$p_date, time_on_plot$p_time, sep = " ")
time_on_plot$p_time <- strptime(time_on_plot$p_time, "%Y-%m-%d %H:%M")
time_on_plot$s_time <- paste(time_on_plot$p_date, time_on_plot$s_time, sep = " ")
time_on_plot$s_time <- strptime(time_on_plot$s_time, "%Y-%m-%d %H:%M")
time_on_plot$e_time <- paste(time_on_plot$p_date, time_on_plot$e_time, sep = " ")
time_on_plot$e_time <- strptime(time_on_plot$e_time, "%Y-%m-%d %H:%M")
# time_on_plot$lf_time <- paste(time_on_plot$p_date, time_on_plot$lf_time, sep = " ")
# time_on_plot$lf_time <- strptime(time_on_plot$lf_time, "%Y-%m-%d %H:%M")

# Calculate time on plot in minutes
time_on_plot$travel_time <- difftime(time_on_plot$s_time, time_on_plot$p_time,units = "min")
time_on_plot$plot_time <- difftime(time_on_plot$e_time, time_on_plot$s_time,units = "min")
time_on_plot$total_time <- difftime(time_on_plot$e_time, time_on_plot$p_time,units = "min")

# tranform minutes to clock format (ie 1:10 = one hour and ten minutes)
```

```
time_on_plot$travel_time <- as.numeric(time_on_plot$travel_time, units = "mins")
time_on_plot$travel_time <- sprintf("%.02d:%.02d", time_on_plot$travel_time %/% 60, round(time_on_plot$travel_time %% 60))

time_on_plot$plot_time <- as.numeric(time_on_plot$plot_time, units = "mins")
time_on_plot$plot_time <- sprintf("%.02d:%.02d", time_on_plot$plot_time %/% 60, round(time_on_plot$plot_time %% 60))

time_on_plot$total_time <- as.numeric(time_on_plot$total_time, units = "mins")
time_on_plot$total_time <- sprintf("%.02d:%.02d", time_on_plot$total_time %/% 60, round(time_on_plot$total_time %% 60))

# Find time errors
time_errors <- time_on_plot[which(time_on_plot$total_time < 0 | time_on_plot$total_time > 12),]

time_errors <- merge(team_name, time_errors, by = "plot_plot_id")

# Add issues to output
if(dim(time_errors)[1] == 0){

  output6 <- data.frame(cp="CP 6",cat="Time on plot",
                        plot_id=9999,issue="Time entered ok",
                        "team" = 99,  stringsAsFactors = F)
  } else {
```

```
  time_errors$cp <- "CP 6"
  time_errors$cat <- "Time on plot"
  time_errors$plot_id <- time_errors$plot_plot_id
  time_errors$issue <- ifelse(time_errors$total_time < 0,
                              "Negative value for time",
                              "Over 12 hours - check times")
  output6 <- time_errors[,c("cp","cat","plot_id","issue", "team")]}

# save the output
write.csv(output6, file = "./QAQC_outputs/6_Inconsistency in time spend for the plot survey.csv",
          row.names = F)

#-------------------------------------------------------------------------------#
#-------------------------------------------------------------------------------#
# Quality control of the estimated attributes  ---------------------------------------
#-------------------------------------------------------------------------------#
#-------------------------------------------------------------------------------#
####### Estimation process
##### Data Management
### Estimation at tree level
```

```
####### Prepare area data ------------------------------------------------------------
# table with lf area proportion and nlcl
sup= merge(subplot_prop,lf[,c("plot_plot_id","lf_id","nlcl")],
        by.x=c("plot_plot_id","lf_subp_lfno"),
        by.y= c("plot_plot_id","lf_id"),all.x=TRUE)
# use completely sampled subplots only
sup=sup[sup$subp_details_subp_status_label=="Sampled completely",] #i remove the plots partially accesible and not accesible

# calculate the areas at plot level
sup$L_area=sup$lf_l_percent/100*(19^2*pi/10000)
sup$m_area=sup$lf_l_percent/100*(8^2*pi/10000)
sup$s_area=sup$lf_l_percent/100*(2.5^2*pi/10000)

# sampled area by plot, subplot and nlcl
ai_sn <- aggregate(list(ai = sup$L_area), by=list(plot_plot_id=sup$plot_plot_id,
                                    subp_id = sup$subplot_subp_id,
                                    nlcl = sup$nlcl), sum) #ai_sn = sampled area by subp and nlcl
# sampled area by plot
ai_p <- aggregate(list(ai = sup$L_area), by=list(plot_plot_id=sup$plot_plot_id), sum)
#sampled area by nlcl
ai_n <- aggregate(list(ai = sup$L_area), by=list(nlcl=sup$nlcl), sum) #ai_n = area by nlcl
```

```
####### Growing stock estimation ----------------------------------------------------------
##### Process: Species specific equation were used for 28 species and quarter girth formula with form factor
#     is used for the remaining species
# live trees trees having >=10 cm DBH are filtered
tree_live <- treesap[treesap$tree_status==1&
                     treesap$tree_dia>=10 &
                     treesap$tree_spp_scientific_name!=""&
                     !is.na(treesap$tree_dia)&
                     !is.na(treesap$tree_total_lgt),] #select only trees >=10 cm DBH
tree_live$tree_dia_m <- tree_live$tree_dia/100
tree_live$GBH_cm <- pi*tree_live$tree_dia
# Convert DBH and Ht from cm and m to inch and ft for Tectona grandis
tree_live$teak_dia_inch <- ifelse(tree_live$tree_spp_scientific_name=="Tectona grandis",
                          tree_live$tree_dia*0.393701, NA)
tree_live$teak_lgt_ft <- ifelse(tree_live$tree_spp_scientific_name=="Tectona grandis",
                        tree_live$tree_total_lgt*3.28084, NA)
# Apply the volume models for estimating the tree volume
tree_live$sp_volume_m3 <- ifelse(tree_live$tree_spp_scientific_name=="Acacia auriculiformis",
                          exp(-11.506528+1.973377*log(tree_live$GBH_cm)+0.623823*log(tree_live$tree_total_lgt)),
```

```
ifelse(tree_live$tree_spp_scientific_name=="Acacia mangium",
    exp(-10.7488+2.2178*log(tree_live$GBH_cm)),
    ifelse(tree_live$tree_spp_scientific_name=="Acacia nilotica",
        exp(-11.875835+1.8823999*log(tree_live$GBH_cm)+1.0819988*log(tree_live$tree_total_lgt)),
        ifelse(tree_live$tree_spp_scientific_name=="Samanea saman",
            exp(-11.37623+2.26924*log(tree_live$GBH_cm)),
            ifelse(tree_live$tree_spp_scientific_name=="Albizia richardiana",
                exp(-10.996396+2.247808*log(tree_live$GBH_cm)),
                ifelse(tree_live$tree_spp_scientific_name=="Albizia procera",
                    exp(-11.6632+1.941989*log(tree_live$GBH_cm)+0.754839*log(tree_live$tree_total_lgt)),
                    ifelse(tree_live$tree_spp_scientific_name=="Albizia sp.",
                        exp(-11.19651+1.85690*log(tree_live$GBH_cm)+0.67878*log(tree_live$tree_total_lgt)),
                        ifelse(tree_live$tree_spp_scientific_name=="Aphanamixis polystachya",
                            exp(-11.25528+1.98544*log(tree_live$GBH_cm)+0.47163*log(tree_live$tree_total_lgt)),
                            ifelse(tree_live$tree_spp_scientific_name=="Artocarpus chama",
                                exp(-8.9449526+1.82851*log(tree_live$tree_dia)+0.735381*log(tree_live$tree_total_lgt)),
                                ifelse(tree_live$tree_spp_scientific_name=="Artocarpus heterophyllus",
                                    exp(-10.99533+1.80823*log(tree_live$GBH_cm)+0.68951*log(tree_live$tree_total_lgt)),
                                    ifelse(tree_live$tree_spp_scientific_name=="Avicennia officinalis",
                                        (0.0089+0.0000264*(tree_live$tree_dia^2)*tree_live$tree_total_lgt),
                                        ifelse(tree_live$tree_spp_scientific_name=="Azadirachta indica",
```

```
                                                                exp(-
11.42823+1.89235*log(tree_live$GBH_cm)+0.71493*log(tree_live$tree_total_lgt)),
                                                                ifelse(tree_live$tree_spp_scientific_name=="Neolamarckia cadamba",
                                                                    exp(-
10.4647+2.3911*log(tree_live$tree_dia)+0.6373*log(tree_live$tree_total_lgt)),
                                                                    ifelse(tree_live$tree_spp_scientific_name=="Dalbergia sissoo",
                                                                        exp(-
12.5189939+1.9800535*log(tree_live$GBH_cm)+1.0775148*log(tree_live$tree_total_lgt)),
                                                                        ifelse(tree_live$tree_spp_scientific_name=="Dipterocarpus turbinatus",
                                                                            exp(-8.5116354+2.35556*log(tree_live$tree_dia)),
                                                                            ifelse(tree_live$tree_spp_scientific_name=="Eucalyptus camaldulensis"
& tree_live$tree_dia<=18,
                                                                                exp(-
9.3520+1.8055*log(tree_live$tree_dia)+0.8590*log(tree_live$tree_total_lgt)),
                                                                                ifelse(tree_live$tree_spp_scientific_name=="Eucalyptus
camaldulensis" & tree_live$tree_dia>18,
                                                                                    (0.076339-
0.00058066*(tree_live$tree_total_lgt)+0.000016216*((tree_live$GBH_cm)^2)+0.0000032565*((tree_live$GBH_cm)^2)*(tree_live$tree_total_lg
t)),
                                                                                    ifelse(tree_live$tree_spp_scientific_name=="Gmelina arborea",
                                                                                        exp(-
8.4687076+1.63502*log(tree_live$tree_dia)+0.784847*log(tree_live$tree_total_lgt)),
                                                                                        ifelse(tree_live$tree_spp_scientific_name=="Hevea
brasiliensis",
```

```
exp(-
11.2768+1.8795*log(tree_live$GBH_cm)+0.6928*log(tree_live$tree_total_lgt)),
ifelse(tree_live$tree_spp_scientific_name=="Lagerstroemia speciosa",
exp(-
9.6744+2.1065*log(tree_live$tree_dia)+0.6675*log(tree_live$tree_total_lgt)),
ifelse(tree_live$tree_spp_scientific_name=="Lannea
coromandelica",
exp(-
11.519102+2.01724*log(tree_live$GBH_cm)+0.56356*log(tree_live$tree_total_lgt)),
ifelse(tree_live$tree_spp_scientific_name=="Mangifera indica",
exp(-
11.25377+1.96697*log(tree_live$GBH_cm)+0.52237*log(tree_live$tree_total_lgt)),
ifelse(tree_live$tree_spp_scientific_name=="Pinus caribaea" & tree_live$tree_dia<=25,
exp(-
9.39412+1.867386*log(tree_live$tree_dia)+0.839034*log(tree_live$tree_total_lgt)),
ifelse(tree_live$tree_spp_scientific_name=="Pinus caribaea" & tree_live$tree_dia>25,
exp(-
11.5317+1.867386*log(tree_live$GBH_cm)+0.839034*log(tree_live$tree_total_lgt)),
ifelse(tree_live$tree_spp_scientific_name=="Senna siamea",
```

```
exp(-
11.6557+1.871*log(tree_live$GBH_cm)+0.897*log(tree_live$tree_total_lgt)),
ifelse(tree_live$tree_spp_scientific_name=="Shorea robusta",
exp(-
12.0554+2.5178944*log(tree_live$GBH_cm)),
ifelse(tree_live$tree_spp_scientific_name=="Sonneratia apetala",
exp(-
9.29715+1.70514*log(tree_live$tree_dia)+0.95088*log(tree_live$tree_total_lgt)),
ifelse(tree_live$tree_spp_scientific_name=="Swietenia macrophylla",
exp(-
11.27102+1.88064*log(tree_live$GBH_cm)+0.64629*log(tree_live$tree_total_lgt)),
ifelse(tree_live$tree_spp_scientific_name=="Syzygium cumini",
exp(-
11.24854+2.24804*log(tree_live$GBH_cm)),
ifelse(tree_live$tree_spp_scientific_name=="Tectona grandis",
((0.000465*(tree_live$teak_dia_inch^(1.58))*(tree_live$teak_lgt_ft^1.603))*0.0283168),
ifelse(tree_live$tree_spp_scientific_name=="Terminalia arjuna",
```

```
                                                                                                    exp(-
11.3794+1.896423*log(tree_live$GBH_cm)+0.653558*log(tree_live$tree_total_lgt)),

ifelse(tree_live$tree_spp_scientific_name=="Xylia xylocarpa",

                                                                                                        exp(-
9.4303+2.0988*log(tree_live$tree_dia)+0.6042*log(tree_live$tree_total_lgt)),

((tree_live$tree_dia_m^2)/4)*pi*tree_live$tree_total_lgt*0.693)))))))))))))))))))))))))))))))))


# volume expansion for L plot size from M and S plots

tree_live$exp_volume_m3 <- ifelse(tree_live$tree_dia<10, tree_live$sp_volume_m3*((19^2*pi)/(2.5^2*pi)), #change the equation here if needed

                    ifelse(tree_live$tree_dia<30, tree_live$sp_volume_m3*((19^2*pi)/(8^2*pi)),

                        tree_live$sp_volume_m3*((19^2*pi)/(19^2*pi)))) # Yhid = volume (Y) in i plot of d domain of h strata



######## Estimate the biomass at tree level ----------------------------------------------

##### WD addition at tree level using species, genus, family

# Read wood density tables and renames column

wd_sp_bd <- read.csv("./WD_WORLD/wd_bd.csv", header=T, stringsAsFactors = F)

wd_fam <- read.table("./WD_WORLD/wd_fam.txt", header=T, sep="\t", stringsAsFactors = FALSE)

wd_gen <- read.table("./WD_WORLD/wd_gen.txt", header=T, sep="\t", stringsAsFactors = FALSE)
```

```
wd_sp <- read.table("./WD_WORLD/wd_sp.txt", header=T, sep="\t", stringsAsFactors = FALSE)

# changes the column names to match with treesap data
names(wd_sp)[1]="tree_spp_scientific_name"
names(wd_fam)[1]="family"
wd_gen$tree_spp_scientific_name=paste(wd_gen$Genus,"sp.", sep=" ") # to match genus names with treesap data

#merging with the treesap data set
treesapWd <- merge(treesap, wd_sp_bd[,c("tree_spp_scientific_name", "wd_bd")], by="tree_spp_scientific_name", all.x = T)
treesapWd <- merge(treesapWd, wd_sp[,c("tree_spp_scientific_name","wd_sp")], by = "tree_spp_scientific_name", all.x =TRUE)
treesapWd <- merge(treesapWd, wd_gen[,c("tree_spp_scientific_name","wd_gen")], by = "tree_spp_scientific_name", all.x =TRUE)
treesapWd <- merge(treesapWd, species[,c("code","family")],by.x = "tree_spp_code", by.y="code", all.x=TRUE)
treesapWd <- merge(treesapWd, wd_fam[,c("family","wd_fam")],by = "family", all.x=TRUE)

# first filling of wd column
treesapWd$wd <- ifelse(!is.na(treesapWd$wd_bd), treesapWd$wd_bd,
                       ifelse(!is.na(treesapWd$wd_sp), treesapWd$wd_sp,
                              ifelse(!is.na(treesapWd$wd_gen), treesapWd$wd_gen,
                                     ifelse(!is.na(treesapWd$wd_fam), treesapWd$wd_fam,
                                            NA))))
#Checking trees with no wd
```

```
treesapWd_NAWD <-treesapWd[which(is.na(treesapWd$wd)),]


#filling procedure checking genus name in species name
for (i in 1:dim(treesapWd)[1]){
  if (is.na({treesapWd$wd[i]})){
    for (j in 1:dim(wd_gen)[1]){
      if (regexpr(wd_gen$Genus[j],treesapWd$tree_spp_scientific_name[i])[1]>0){
        treesapWd$wd_gen[i]=wd_gen$wd_gen[j]
      }
    }
  }
}


# checking improvements
treesapWd$wd <- ifelse(!is.na(treesapWd$wd_bd),treesapWd$wd_bd,
                       ifelse(!is.na(treesapWd$wd_sp), treesapWd$wd_sp,
                              ifelse(!is.na(treesapWd$wd_gen), treesapWd$wd_gen,
                                     ifelse(!is.na(treesapWd$wd_fam), treesapWd$wd_fam,
                                            NA))))
treesap_NAWD <-treesapWd[which(is.na(treesapWd$wd)),]
unique(treesap_NAWD$family)
```

```
unique(treesap_NAWD$tree_spp_scientific_name)


# final filling, the rest of the trees (unknown species) take the overall average WD
treesapWd$wd <- ifelse(!is.na(treesapWd$wd_bd),treesapWd$wd_bd,
                       ifelse(!is.na(treesapWd$wd_sp), treesapWd$wd_sp,
                              ifelse(!is.na(treesapWd$wd_gen), treesapWd$wd_gen,
                                     ifelse(!is.na(treesapWd$wd_fam), treesapWd$wd_fam,
                                            0.6133782)))) #here, wd in g/cm3 unit
# prepare a columen with WD in kg/m3 unit
treesapWd$wd_kg_m3 <- treesapWd$wd*1000



# Process: for trees with both DBH and Height, AGB at tree level is calculated using species specific
#   equation used for 7 species and 4 common zone specific equation is used for respective zone
#   (following the AE selection decision tree).

# Live trees and saplings are filtered with dia, length and scientific name
treesap_live <- treesapWd[(treesapWd$tree_status==1|
                           treesapWd$tree_status==99) &
                          !is.na(treesapWd$tree_dia) &
                          !treesapWd$tree_dia<2 &
```

```
                !treesapWd$tree_total_lgt<=1.3 &
                treesapWd$tree_spp_scientific_name!="",]


# Apply the equations for biomass estimation
treesap_live$tree_D2H=treesap_live$tree_total_lgt*(treesap_live$tree_dia)^2
treesap_live$tree_agb_kg=ifelse(treesap_live$tree_spp_scientific_name=="Sonneratia apetala",
                        exp(-1.7608+2.0077*log(treesap_live$tree_dia)+0.2981*log(treesap_live$tree_total_lgt)),
                        ifelse(treesap_live$tree_spp_scientific_name=="Excoecaria agallocha",
                            10^(1.0996*log10((treesap_live$tree_dia)^2)-0.8572),
                            ifelse(treesap_live$tree_spp_scientific_name=="Acacia auriculiformis",
                                (10^(-0.475 + 0.614*log10 (treesap_live$tree_dia^2)))^2,
                                ifelse(treesap_live$tree_spp_scientific_name=="Acacia mangium",
                                    (10^(-0.497 + 0.606*log10 (treesap_live$tree_dia^2)))^2,
                                    ifelse(treesap_live$tree_spp_scientific_name=="Heritiera fomes",
                                        exp(-2.1324 + 2.3895*log(treesap_live$tree_dia) + 0.1367*log(treesap_live$tree_total_lgt)),
                                        ifelse(treesap_live$tree_spp_scientific_name=="Shorea robusta",
                                            exp(-3.3592 + 2.1830*log(treesap_live$tree_dia) + 0.6787*log(treesap_live$tree_total_lgt)),
                                            ifelse(treesap_live$tree_spp_scientific_name=="Gmelina arborea",
                                                exp(-3.028+0.925*log(treesap_live$tree_D2H)),
                                                ifelse(treesap_live$location_zone=="Hill",
```

```
                                        exp(-6.9531+ 0.825*log(treesap_live$tree_D2H*treesap_live$wd_kg_m3)),
                                        ifelse(treesap_live$location_zone=="Sal",
                                            exp(-2.460 + 2.171*log(treesap_live$tree_dia) + 0.367*log(treesap_live$tree_total_lgt) +
0.161*log(treesap_live$wd_kg_m3)),
                                            ifelse(treesap_live$location_zone=="Sundarbans",
                                                exp(-6.7189 + 2.1634*log(treesap_live$tree_dia) + 0.3752*log(treesap_live$tree_total_lgt)
+ 0.6895*log(treesap_live$wd_kg_m3)),
                                                ifelse((treesap_live$location_zone=="Villages" | (treesap_live$location_zone=="Coastal")),
                                                    exp(-6.0325 +
1.9715*log(treesap_live$tree_dia)+0.8193*log(treesap_live$wd_kg_m3)),
                                                    0.0673*(treesap_live$wd*treesap_live$tree_D2H)^(0.976))))))))))))


# Expand tree and sapling AGB from S and M plot to L plot

treesap_live$exp_tree_agb_kg <- ifelse(treesap_live$tree_dia<10, treesap_live$tree_agb_kg*((19^2*pi)/(2.5^2*pi)), #biomss expansion for
different plot size
                                ifelse(treesap_live$tree_dia<30, treesap_live$tree_agb_kg*((19^2*pi)/(8^2*pi)),
                                    treesap_live$tree_agb_kg*((19^2*pi)/(19^2*pi))))
```

```
### CWD volume and biomass estimation ------------------------------------------------

# select the completely sampled subplots only

cwd_sub <- cwd[cwd$subp_details_subp_status_label=="Sampled completely",]


# Filter out the CWD that not having diameter and transect id

cwd_sub <- cwd_sub[which(!is.na(cwd_sub$cwd_dia) & cwd_sub$cwd_tran!=0 & !is.na(cwd_sub$cwd_lf)),] #since one row under cwd_tran is 0


#calculate vol per ha using the formula, biomass (t/ha)= (pi^2/(8+L)*summation (D^2*WD) (following Marshall et al. 2000)

#add a column for CWD dia square

cwd_sub$D2 <- cwd_sub$cwd_dia*cwd_sub$cwd_dia #D2 = CWD diameter square


#considerting number of transact 4 each with 8 m length

#get sum of D2wdF per plot, subplot, nlcl and transact

cwdTranVol <- aggregate(D2~plot_plot_id+subplot_subp_id+cwd_lf+cwd_tran+cwd_decay,

                        data=cwd_sub, sum) #cwdTranVol = CWD transact volume

cwdTranVol$cwd_vol_ha <- (pi*pi)/(8*8)*cwdTranVol$D2


### add wood density to compute biomass

cwdTranVol$wd <- 0.6133782 #average wood density is used for CWD


#creating a column with reduction factor for wood density by Harmon et al. 2011
```

```
cwdTranVol$red_fac <- ifelse(cwdTranVol$cwd_decay==1,0.95,
                        ifelse(cwdTranVol$cwd_decay==2,0.74,
                            ifelse(cwdTranVol$cwd_decay==3,0.51,
                                ifelse(cwdTranVol$cwd_decay==4,0.29, 0.22))))

# multiply the red_fac with wd to get the final wd (wdF)
cwdTranVol$wdF <- cwdTranVol$red_fac*cwdTranVol$wd

# compute the biomass at transact and decay status level
cwdTranVol$cwd_biom_ha <- cwdTranVol$cwd_vol_ha*cwdTranVol$wdF

# Compute bioamss at plot, subplot and nlcl level
cwdBiomSpl <- aggregate(data=cwdTranVol, cwd_biom_ha~plot_plot_id+subplot_subp_id+cwd_lf, sum)
cwdBiomSpl$cwd_biom_ha <- cwdBiomSpl$cwd_biom_ha/4 # the biomass is divided by 4 to get mean biomass of each transact

# Add nlcl to the cwdBiomSpl
cwdBiomSplLf <- merge(cwdBiomSpl, lf[,c("plot_plot_id", "lf_id", "nlcl")],
                  by.x=c("plot_plot_id", "cwd_lf"),
                  by.y=c("plot_plot_id", "lf_id"), all.x=T)

# Add sampled sampled area per plot, subplot and lf
```

```
cwdBiomSplLfF <- merge(cwdBiomSplLf, ai_sn, by.x = c("plot_plot_id", "subplot_subp_id", "nlcl"),
                       by.y=c("plot_plot_id", "subp_id", "nlcl"), all=T)



# FWD volume and biomass estimation ----------------------------------------------------
# Add subplot sampling status to FWD
fwd_sub <- merge(fwd, subplot_info[,c("plot_plot_id", "subp_id", "subp_details_subp_status_label")],
                 by.x=c("plot_plot_id", "subplot_subp_id"), by.y=c("plot_plot_id", "subp_id"), all.x=T)
# select the completely sampled subplots only
fwd_sub <- fwd_sub[fwd_sub$subp_details_subp_status_label=="Sampled completely",]

#select transacts with fwd recorded and lf id
fwd_subF <- fwd_sub[which(fwd_sub$fwd_tran_label!="No woody debris observed" &
                          fwd_sub$fwd_lfid>0),]

#get total number of fwd of in the classes
fwd_subF$tot_fwdct <- fwd_subF$fwd_smallct+fwd_subF$fwd_medct+fwd_subF$fwd_lrgct

#merge nlcl
fwd_subFLf <- merge(fwd_subF, lf[,c("plot_plot_id", "lf_id", "nlcl")], by.x = c("plot_plot_id", "fwd_lfid"),
```

```
                by.y = c("plot_plot_id", "lf_id"),all.x=TRUE)

#select only those transacts with fwd recorded
fwd_subFLf <- fwd_subFLf[which(fwd_subFLf$tot_fwdct>0),]

#calculate the biomass
#step 1: put avg diameter, get diameter square and multiply with wd (0.6133782) for each class
# here, transact length 3 for small and medium FWD, 8 for Large FWD; avg dia 0.5, 2 and 5.5 cm for
# small, medium and large FWD respectively
fwd_subFLf$Sfwd_biom_t_ha <- 0.5*0.5*0.6133782*0.51*fwd_subFLf$fwd_smallct*(pi*pi)/(8*3) #uses the mean value for reduction factor
fwd_subFLf$Mfwd_biom_t_ha <- 2.00*2.00*0.6133782*0.51*fwd_subFLf$fwd_medct*(pi*pi)/(8*3)
fwd_subFLf$Lfwd_biom_t_ha <- 5.50*5.50*0.6133782*0.51*fwd_subFLf$fwd_lrgct*(pi*pi)/(8*8)

###sum the fwd biomass for all the size categories for each transact
# converte the NA into zero
fwd_subFLf[,16:18][is.na(fwd_subFLf[,16:18])] <- 0
fwd_subFLf$fwd_biom_t_ha <- fwd_subFLf$Sfwd_biom_t_ha + fwd_subFLf$Mfwd_biom_t_ha + fwd_subFLf$Lfwd_biom_t_ha

# get the subplot level FWD biomass
fwdBiomSpl <- aggregate(data=fwd_subFLf, fwd_biom_t_ha~plot_plot_id+subplot_subp_id+nlcl, FUN=sum)
fwdBiomSpl$fwd_biom_t_ha <- fwdBiomSpl$fwd_biom_t_ha/4 # the value is divided into 4 to get transact mean
```

```
# add sampled are for at plot, subplot and nlcl level
fwdBiomSplf <- merge(fwdBiomSpl, ai_sn, by.x = c("plot_plot_id", "subplot_subp_id", "nlcl"),
                     by.y=c("plot_plot_id", "subp_id", "nlcl"), all=T)



#-------------------------------------------------------------------------------------#
#-------------------------------------------------------------------------------------#
# Quality control of the estimated attributes  ---------------------------------------
#-------------------------------------------------------------------------------------#

#-------------------------------------------------------------------------------------#
# CP 7.1: Outliers (extreme values) in tree/ha by plot --------------------------------
# Input: tree.csv and sapling.csv
# Output: List of plots with extreme tree/sapling density
#-------------------------------------------------------------------------------------#
# select only key variables
treesap_liveKV <- treesap_live[,c("plot_plot_id", "subplot_subp_id", "nlcl",
                                  "tree_status", "tree_dia")]
# add a column named "tree_count" inn treesasp_live
treesap_liveKV$tree_count <- 1
```

```
# Expand tree and sapling count from S and M plot to L plot

treesap_liveKV$exp_stem_no <- ifelse(treesap_liveKV$tree_dia<10, treesap_liveKV$tree_count*((19^2*pi)/(2.5^2*pi)), #biomss expansion for
different plot size

                        ifelse(treesap_liveKV$tree_dia<30, treesap_liveKV$tree_count*((19^2*pi)/(8^2*pi)),

                            treesap_liveKV$tree_count*((19^2*pi)/(19^2*pi))))

# tree stem number at plot, subplot and nlcl level

stemdataSpl <- aggregate(data=treesap_liveKV, exp_stem_no~plot_plot_id+subplot_subp_id+nlcl, sum)


#  Merge of "stemdataSpl" and "ai_sn" (subplot level sampled area) to get Stem_Ha at subplot-lf level

stemdataSplF <-merge(stemdataSpl, ai_sn, by.x=c("plot_plot_id","subplot_subp_id","nlcl"),

          by.y=c("plot_plot_id","subp_id","nlcl"),all = T)


## Computing of "Stem_Ha"

stemdataSplF$Stem_Ha<- (stemdataSplF$exp_stem_no)/stemdataSplF$ai

# convert the NAs in "Stem_Ha" of stemdata into 0

stemdataSplF[,6][is.na(stemdataSplF[,6])] <- 0



### Identification of Outliers

# A table of statistics (mean and sd) is done
```

```
f71 <- function(x) c(n = length(x), Mean = mean(x), SD = sd(x))
BasicSat <- aggregate(Stem_Ha ~ nlcl,
                      data=stemdataSplF[!is.na(ifelse(stemdataSplF$Stem_Ha==Inf,NA,
                                                      stemdataSplF$Stem_Ha)),], FUN=f71)
BasicSat$n<-BasicSat$Stem_Ha[,1]
BasicSat$Mean<-BasicSat$Stem_Ha[,2]
BasicSat$SD<-BasicSat$Stem_Ha[,3]

# Stem database and Statistical database are merged by nlcl
stemdataSplFSt<-merge(stemdataSplF,BasicSat[,c("nlcl", "n", "Mean", "SD")],
                      by.x="nlcl", by.y="nlcl",all.x=T)
stemdataSplFStF <- stemdataSplFSt[stemdataSplFSt$n>3,]
# The Z value is estimated and Possible Outliers are identified
stemdataSplFStF$Z<-((stemdataSplFStF$Stem_Ha-stemdataSplFStF$Mean)/
                      stemdataSplFStF$SD)
stemdataSplFStF$PO<-ifelse(stemdataSplFStF$Z>4,1,0)
dim(stemdataSplFStF)
# Identification od Outliers
stemdataSplFStFf<-stemdataSplFStF[!is.na(stemdataSplFStF$PO) & stemdataSplFStF$PO==1,]

### Possible Outliers are summarized and saved
```

```
# A Database of possible OL is created
stemdataSplFStFf$issue <- paste("LCC:", stemdataSplFStFf$nlcl,
                          "Stem_Ha:", round(stemdataSplFStFf$Stem_Ha,1),
                          "Mean:",round(stemdataSplFStFf$Mean,1),
                          "Z:",round(stemdataSplFStFf$Z,1),sep = " ")
stemdataSplFStFf$cp <- "CP 7.1"
stemdataSplFStFf$cat <- "Outlier in tree Stem per hectare"
stemdataSplFStFf <- merge(stemdataSplFStFf, team_name, by="plot_plot_id", all.x=T)
names(stemdataSplFStFf)[1] <- "plot_id"

# Prepare the output table
if(dim(stemdataSplFStFf)[1] == 0){
  output7.1 <- data.frame(cp="CP 7.1",cat="Outlier in tree Stem per hectare",plot_id=NA,
                          issue="No outlier in tree Stem per hectare", team = NA,
                          stringsAsFactors = F)} else {
  output7.1 <- stemdataSplFStFf[,c("cp","cat","plot_id","issue", "team")]}

# save the output
write.csv(output7.1, file = "./QAQC_outputs/7.1_Possible outlier in tree stem per hectare.csv",
          row.names = F)
```

```
#-------------------------------------------------------------------------------#
# CP 7.2 (4.15): Extreme values in AGB/ha by plot for large trees (DBH >=30cm) --------
# Input: tree.csv, plot.csv, lf.csv, AE, WD
# Output: List of plots with extreme values in AGB/hectare
#-------------------------------------------------------------------------------#
####### Outlier analysis for Medium trees >=30 cm DBH
# Seleccion of Key variabes of Tree database
treesap_liveKV<-treesap_live[,c("plot_plot_id","subplot_subp_id","tree_lfid", "nlcl",
                          "tree_status","exp_tree_agb_kg","tree_dia")]
# Selecction of trees with DBH>30cm
treesap_liveKV<-treesap_liveKV[treesap_liveKV$tree_dia>=30,]
dim(treesap_liveKV)
# Compute of AGB at sub-plot level
AGBdataSpl <- aggregate(exp_tree_agb_kg~plot_plot_id+subplot_subp_id+nlcl,
                    data=treesap_liveKV, sum)

#  Merge of "AGBdataSpl" and "ai_sn" (subplot level sampled area) to get AGB at subplot-lf level
AGBdataSplF <-merge(AGBdataSpl, ai_sn, by.x=c("plot_plot_id","subplot_subp_id","nlcl"),
                by.y=c("plot_plot_id","subp_id","nlcl"),all = T)

## Computing of "AGB a SupPlot-lf" and "AGB_tonHa"
```

```
AGBdataSplF$AGB_tonHa<- (AGBdataSplF$exp_tree_agb_kg/1000)/AGBdataSplF$ai
# convert the NAs in AGB_tonHa into 0
AGBdataSplF[,6][is.na(AGBdataSplF[,6])] <- 0


### Identification of Outliers
# A table of statistics (mean and sd) is done
f72 <- function(x) c(n = length(x), Mean = mean(x), SD = sd(x))
BasicSat <- aggregate(AGB_tonHa ~ nlcl,
                      data=AGBdataSplF[!is.na(ifelse(AGBdataSplF$AGB_tonHa==Inf,NA,
                                                     AGBdataSplF$AGB_tonHa)),], FUN=f72)
BasicSat$n<-BasicSat$AGB_tonHa[,1]
BasicSat$Mean<-BasicSat$AGB_tonHa[,2]
BasicSat$SD<-BasicSat$AGB_tonHa[,3]

# Tree database and Statistical database are merged by sientific name
AGBdataSplFSt<-merge(AGBdataSplF,BasicSat[,c("nlcl", "n", "Mean", "SD")],
                     by.x="nlcl", by.y="nlcl",all.x=T)
AGBdataSplFStF <- AGBdataSplFSt[AGBdataSplFSt$n>3,]
# The Z value is estimated and Possible Outliers are identified
AGBdataSplFStF$Z<-((AGBdataSplFStF$AGB_tonHa-AGBdataSplFStF$Mean)/
```

```
                AGBdataSplFStF$SD)
AGBdataSplFStF$PO<-ifelse(AGBdataSplFStF$Z>4,1,0)
dim(AGBdataSplFStF)
# Identification od Outliers
AGBdataSplFStFf<-AGBdataSplFStF[!is.na(AGBdataSplFStF$PO) & AGBdataSplFStF$PO==1,]


### Possible Outliers are summarized and saved
# A Database of possible OL is created
AGBdataSplFStFf$issue <- paste("LCC:", AGBdataSplFStFf$nlcl,
                       "agb:", round(AGBdataSplFStFf$AGB_tonHa,1),
                       "Mean:",round(AGBdataSplFStFf$Mean,1),
                       "Z:",round(AGBdataSplFStFf$Z,1),sep = " ")
AGBdataSplFStFf$cp <- "CP 7.2"
AGBdataSplFStFf$cat <- "Outlier in large tree (DBH >=30cm) AGB density"
AGBdataSplFStFf <- merge(AGBdataSplFStFf, team_name, by="plot_plot_id", all.x=T)
names(AGBdataSplFStFf)[1] <- "plot_id"

# Prepare the output table
if(dim(AGBdataSplFStFf)[1] == 0){
  output7.2 <- data.frame(cp="CP 7.2",cat="Outlier in large tree (DBH>=30cm) AGB density",plot_id=NA,
                    issue="No outlier is found in large tree (DBH>=30cm) AGB density", team = NA,
```

```
                    stringsAsFactors = F)} else {
  output7.2 <- AGBdataSplFStFf[,c("cp","cat","plot_id","issue", "team")]}

# save the output
write.csv(output7.2,
        file = "./QAQC_outputs/7.2_Possible outlier in large tree (more than 30 cm) AGB per hectare.csv",
        row.names = F)

#-------------------------------------------------------------------------------#
# CP 7.3 (4.15): Extreme values in AGB/ha by plot for Medium trees (DBH >= 10cm) ------
# Input: tree.csv, plot.csv, lf.csv, AE, WD
# Output: List of plots with extreme values in AGB/hectare
#-------------------------------------------------------------------------------#

####### Outlier analysis for Medium trees 30cm > DBH >=10cm
# Seleccion of Key variabes of Tree database
treesap_liveKV<-treesap_live[,c("plot_plot_id","subplot_subp_id","tree_lfid", "nlcl",
                        "tree_status","exp_tree_agb_kg","tree_dia")]
# Selecction of trees with DBH>30cm
treesap_liveKV<-treesap_liveKV[treesap_live$tree_dia<30 & treesap_liveKV$tree_dia>=10,]
dim(treesap_liveKV)
```

```
# Compute of AGB at sub-plot level
AGBdataSpl <- aggregate(exp_tree_agb_kg~plot_plot_id+subplot_subp_id+nlcl,
                        data=treesap_liveKV, sum)

#  Merge of "AGBdataSpl" and "ai_sn" (subplot level sampled area) to get AGB at subplot-lf level
AGBdataSplF <-merge(AGBdataSpl, ai_sn, by.x=c("plot_plot_id","subplot_subp_id","nlcl"),
                    by.y=c("plot_plot_id","subp_id","nlcl"),all = T)

## Computing of "AGB a SupPlot-lf" and "AGB_tonHa"
AGBdataSplF$AGB_tonHa<- (AGBdataSplF$exp_tree_agb_kg/1000)/AGBdataSplF$ai
# convert the NAs in AGB_tonHa into 0
AGBdataSplF[,6][is.na(AGBdataSplF[,6])] <- 0


### Identification of Outliers
# A table of statistics (mean and sd) is done
f73 <- function(x) c(n = length(x), Mean = mean(x), SD = sd(x))
BasicSat <- aggregate(AGB_tonHa ~ nlcl,
                      data=AGBdataSplF[!is.na(ifelse(AGBdataSplF$AGB_tonHa==Inf,NA,
                                                     AGBdataSplF$AGB_tonHa)),], FUN=f73)
BasicSat$n<-BasicSat$AGB_tonHa[,1]
```

```
BasicSat$Mean<-BasicSat$AGB_tonHa[,2]
BasicSat$SD<-BasicSat$AGB_tonHa[,3]

# Tree database and Statistical database are merged by sientific name
AGBdataSplFSt<-merge(AGBdataSplF,BasicSat[,c("nlcl", "n", "Mean", "SD")],
                     by.x="nlcl", by.y="nlcl",all.x=T)
AGBdataSplFStF <- AGBdataSplFSt[AGBdataSplFSt$n>3,]
# The Z value is estimated and Possible Outliers are identified
AGBdataSplFStF$Z<-((AGBdataSplFStF$AGB_tonHa-AGBdataSplFStF$Mean)/
                     AGBdataSplFStF$SD)
AGBdataSplFStF$PO<-ifelse(AGBdataSplFStF$Z>4,1,0)

# Identification od Outliers
AGBdataSplFStFf<-AGBdataSplFStF[!is.na(AGBdataSplFStF$PO) & AGBdataSplFStF$PO==1,]


### Possible Outliers are summarized and saved
# A Database of possible OL is created
AGBdataSplFStFf$issue <- paste("LCC:", AGBdataSplFStFf$nlcl,
                               "agb:", round(AGBdataSplFStFf$AGB_tonHa,1),
                               "Mean:",round(AGBdataSplFStFf$Mean,1),
```

```
                    "Z:",round(AGBdataSplFStFf$Z,1),sep = " ")
AGBdataSplFStFf$cp <- "cp 7.3"
AGBdataSplFStFf$cat <- "Outlier in Medium tree (30 > DBH >=10cm) AGB density"
AGBdataSplFStFf <- merge(AGBdataSplFStFf, team_name, by="plot_plot_id", all.x=T)
names(AGBdataSplFStFf)[1] <- "plot_id"

# Prepare the output table
if(dim(AGBdataSplFStFf)[1] == 0){
  output7.3 <- data.frame(cp="CP 7.3",cat="Outlier in Medium tree (30cm>DBH>=10cm) AGB density",plot_id=NA,
                    issue="No outlier is found in Medium tree (30cm>DBH>=10cm) AGB density", team = NA,
                    stringsAsFactors = F)} else {
  output7.3 <- AGBdataSplFStFf[,c("cp","cat","plot_id","issue", "team")]}

# save the output7.3,
write.csv(output7.3,
        file = "./QAQC_outputs/7.3_Possible outlier in medium tree (10 to 30 cm DBH) AGB per hectare.csv",
        row.names = F)

#-------------------------------------------------------------------------------------#
# CP 7.4 (4.15): Extreme values in AGB/ha by plot for Saplings (10cm > DBH >= 2cm)-----
# Input: tree.csv, plot.csv, lf.csv, AE, WD
```

```
# Output: List of plots with extreme values in AGB/hectare
#-------------------------------------------------------------------------------#

####### Outlier analysis for Sapling DBH 10cm>DBH>=2cm
# Seleccion of Key variabes of Tree database
treesap_liveKV<-treesap_live[,c("plot_plot_id","subplot_subp_id","tree_lfid", "nlcl",
                          "tree_status","exp_tree_agb_kg","tree_dia")]
# Selecction of trees with DBH>30cm
treesap_liveKV<-treesap_liveKV[treesap_liveKV$tree_dia<10,]
dim(treesap_liveKV)
# Compute of AGB at sub-plot level
AGBdataSpl <- aggregate(exp_tree_agb_kg~plot_plot_id+subplot_subp_id+nlcl,
                    data=treesap_liveKV, sum)

#  Merge of "AGBdataSpl" and "ai_sn" (subplot level sampled area) to get AGB at subplot-lf level
AGBdataSplF <-merge(AGBdataSpl, ai_sn, by.x=c("plot_plot_id","subplot_subp_id","nlcl"),
                by.y=c("plot_plot_id","subp_id","nlcl"),all = T)

## Computing of "AGB a SupPlot-lf" and "AGB_tonHa"
AGBdataSplF$AGB_tonHa<- (AGBdataSplF$exp_tree_agb_kg/1000)/AGBdataSplF$ai
# convert the NAs in AGB_tonHa into 0
```

```
AGBdataSplF[,6][is.na(AGBdataSplF[,6])] <- 0



### Identification of Outliers
# A table of statistics (mean and sd) is done
f74 <- function(x) c(n = length(x), Mean = mean(x), SD = sd(x))
BasicSat <- aggregate(AGB_tonHa ~ nlcl,
                      data=AGBdataSplF[!is.na(ifelse(AGBdataSplF$AGB_tonHa==Inf,NA,
                                                     AGBdataSplF$AGB_tonHa)),], FUN=f74)
BasicSat$n<-BasicSat$AGB_tonHa[,1]
BasicSat$Mean<-BasicSat$AGB_tonHa[,2]
BasicSat$SD<-BasicSat$AGB_tonHa[,3]

# Tree database and Statistical database are merged by sientific name
AGBdataSplFSt<-merge(AGBdataSplF,BasicSat[,c("nlcl", "n", "Mean", "SD")],
                     by.x="nlcl", by.y="nlcl",all.x=T)
AGBdataSplFStF <- AGBdataSplFSt[AGBdataSplFSt$n>3,]
# The Z value is estimated and Possible Outliers are identified
AGBdataSplFStF$Z<-((AGBdataSplFStF$AGB_tonHa-AGBdataSplFStF$Mean)/
                     AGBdataSplFStF$SD)
AGBdataSplFStF$PO<-ifelse(AGBdataSplFStF$Z>4,1,0)
```

```
# Identification od Outliers
AGBdataSplFStFf<-AGBdataSplFStF[!is.na(AGBdataSplFStF$PO) & AGBdataSplFStF$PO==1,]
dim(AGBdataSplFStFf)



### Possible Outliers are summarized and saved
# A Database of possible OL is created
AGBdataSplFStFf$issue <- paste("LCC:", AGBdataSplFStFf$nlcl,
                               "agb:", round(AGBdataSplFStFf$AGB_tonHa,1),
                               "Mean:",round(AGBdataSplFStFf$Mean,1),
                               "Z:",round(AGBdataSplFStFf$Z,1),sep = " ")
AGBdataSplFStFf$cp <- "CP 7.4"
AGBdataSplFStFf$cat <- "Outlier in Saplings (10cm>DBH>=2cm) AGB density"
AGBdataSplFStFf <- merge(AGBdataSplFStFf, team_name, by="plot_plot_id", all.x=T)
names(AGBdataSplFStFf)[1] <- "plot_id"

# Prepare the output table
if(dim(AGBdataSplFStFf)[1] == 0){
  output7.4 <- data.frame(cp="CP 7.4",cat="Outlier in Saplings (10cm>DBH>=2cm) AGB density",plot_id=NA,
                          issue="No outlier is found in Saplings (10cm>DBH>=2cm) AGB density", team = NA,
```

```
                    stringsAsFactors = F)} else {
  output7.4 <- AGBdataSplFStFf[,c("cp","cat","plot_id","issue", "team")]}


# save the output,
write.csv(output7.4,
        file = "./QAQC_outputs/7.4_Possible outlier in Sapling AGB per hectare.csv",
        row.names = F)
#-------------------------------------------------------------------------------#
# CP 7.5 (4.16): Plots of strange values in biomass-leaf cover relationship ----------
# Input: tree.csv, sapling.csv, lf.csv
# Output: list of plots and lf with strange AGB-CC relationship
#-------------------------------------------------------------------------------#


#calculate biomass in M plot because leaf cover is also measured within M plot
ps_agb<- aggregate(tree_agb_kg~plot_plot_id+subplot_subp_id, data= treesap_live[treesap_live$tree_dist<=8,], sum)


##### calculate leaf cover
# consider completely sampled subplots only
subpKV <- subplot_info[subplot_info$subp_details_subp_status==1,][,c("plot_plot_id",
                                                "subp_id",
                                                "subp_details_subp_leaf_pc")]
```

```
# add leaf cover from subplot info
ps_agbLc <- merge(ps_agb, subpKV, by.x=c("plot_plot_id", "subplot_subp_id"),
                  by.y=c("plot_plot_id", "subp_id"), all.x=T)
# add sampled area by M plot
ps_agbLc$ai <- (pi*8^2)/10000
#compute agb per hectare
ps_agbLc$agb_t_ha <- (ps_agbLc$tree_agb_kg/1000)/ps_agbLc$ai
# plot LC Vs AGB t/ha
#graphical verification of biomass per plot (AGB of trees in Kg/plot) vs leaf cover
ggplot(ps_agbLc) +
  geom_point(aes(x=subp_details_subp_leaf_pc, y=tree_agb_kg))+
  xlab("Leaf cover percentage") + ylab("Tree above ground biomass in Kg")
# identify plots and subplots with 0 leaf cover but agb density>0
ps_agbLcF <- filter(ps_agbLc, subp_details_subp_leaf_pc==0 & agb_t_ha>0)

#prepare output table
ps_agbLcF$cp <- "CP 7.5"
ps_agbLcF$cat <- "Inconsistency between leaf cover and AGB t/ha"
ps_agbLcF$issue <- "Leaf cover is zero but AGB t/ha >0"
ps_agbLcF <- merge(ps_agbLcF, team_name, by="plot_plot_id", all.x = T)
names(ps_agbLcF)[1] <- "plot_id"
```

```
# Prepare the output table
if(dim(ps_agbLcF)[1] == 0){
  output7.5 <- data.frame(cp="CP 2.21",cat="Inconsistency between leaf cover and AGB t/ha",
                          plot_id=NA, issue="Relationship between leaf cover and AGB t/ha is consistent",
                          team = NA, stringsAsFactors = F)} else {
  output7.5 <- ps_agbLcF[,c("cp","cat","plot_id","issue", "team")]}

# save the output,
write.csv(output7.5,file = "./QAQC_outputs/7.5_Inconsistency between leaf cover and AGB.csv",
          row.names = F)



#-------------------------------------------------------------------------------#
# CP 7.6: Eextreme values in Large (>=30cm DBH) tree volume/hectare by plot ----------
# Input: tree.csv, plot.csv, lf.csv, AE
# Output: List of plots with extreme values in tree volume/hectare
#-------------------------------------------------------------------------------#

# Seleccion of Key variabes of Tree database
tree_liveKV <-tree_live[,c("plot_plot_id","subplot_subp_id","tree_lfid", "nlcl",
                           "tree_status","exp_volume_m3","tree_dia")]
```

```
# Selecction of trees with DBH>30cm
tree_liveKV<-tree_liveKV[tree_liveKV$tree_dia>=30,]
dim(tree_liveKV)
# Compute of Volume at sub-plot level
VOLdataSpl <- aggregate(exp_volume_m3~plot_plot_id+subplot_subp_id+nlcl,
                        data=tree_liveKV, sum)

#  Merge of "VOLdataSpl" and "ai_sn" (subplot level sampled area) to get Volume at subplot-lf level
VOLdataSplF <-merge(VOLdataSpl, ai_sn, by.x=c("plot_plot_id","subplot_subp_id","nlcl"),
                    by.y=c("plot_plot_id","subp_id","nlcl"),all = T)

## Computing of "VOL at SupPlot-lf" and "VOL_m3Ha"
VOLdataSplF$VOL_m3Ha<- (VOLdataSplF$exp_volume_m3)/VOLdataSplF$ai
# convert the NAs in VOL_tonHa into 0
VOLdataSplF[,6][is.na(VOLdataSplF[,6])] <- 0


### Identification of Outliers
# A table of statistics (mean and sd) is done
f76 <- function(x) c(n = length(x), Mean = mean(x), SD = sd(x))
BasicSat <- aggregate(VOL_m3Ha ~ nlcl,
```

```
                data=VOLdataSplF[!is.na(ifelse(VOLdataSplF$VOL_m3Ha==Inf,NA,
                                               VOLdataSplF$VOL_m3Ha)),], FUN=f76)
BasicSat$n<-BasicSat$VOL_m3Ha[,1]
BasicSat$Mean<-BasicSat$VOL_m3Ha[,2]
BasicSat$SD<-BasicSat$VOL_m3Ha[,3]

# Tree database and Statistical database are merged by sientific name
VOLdataSplFSt<-merge(VOLdataSplF,BasicSat[,c("nlcl", "n", "Mean", "SD")],
                by.x="nlcl", by.y="nlcl",all.x=T)
VOLdataSplFStF <- VOLdataSplFSt[VOLdataSplFSt$n>3,]
# The Z value is estimated and Possible Outliers are identified
VOLdataSplFStF$Z<-((VOLdataSplFStF$VOL_m3Ha-VOLdataSplFStF$Mean)/
                VOLdataSplFStF$SD)
VOLdataSplFStF$PO<-ifelse(VOLdataSplFStF$Z>4,1,0)
dim(VOLdataSplFStF)
# Identification od Outliers
VOLdataSplFStFf<-VOLdataSplFStF[!is.na(VOLdataSplFStF$PO) & VOLdataSplFStF$PO==1,]

### Possible Outliers are summarized and saved
# A Database of possible OL is created
VOLdataSplFStFf$issue <- paste("LCC:", VOLdataSplFStFf$nlcl,
```

```
                    "vol:", round(VOLdataSplFStFf$VOL_m3Ha,1),
                    "Mean:",round(VOLdataSplFStFf$Mean,1),
                    "Z:",round(VOLdataSplFStFf$Z,1),sep = " ")
VOLdataSplFStFf$cp <- "CP 7.6"
VOLdataSplFStFf$cat <- "Outlier in large tree (DBH >=30cm) Volume density"
VOLdataSplFStFf <- merge(VOLdataSplFStFf, team_name, by="plot_plot_id", all.x=T)
names(VOLdataSplFStFf)[1] <- "plot_id"

# Prepare the output table
if(dim(VOLdataSplFStFf)[1] == 0){
  output7.6 <- data.frame(cp="CP 7.6",cat="Outlier in large tree (DBH>=30cm) Vol density",plot_id=NA,
                    issue="No outlier is found in large tree (DBH>=30cm) Vol density", team = NA,
                    stringsAsFactors = F)} else {
  output7.6 <- VOLdataSplFStFf[,c("cp","cat","plot_id","issue", "team")]}

# save the output,
write.csv(output7.6,
        file = "./QAQC_outputs/7.6_Possible outlier in large tree (DBH 30cm or above) vol density.csv",
        row.names = F)

#-------------------------------------------------------------------------------#
```

```
# CP 7.7: Extreme values in Medium (30 >DBH >=10cm) tree volume/ha by plot ----------
# Input: tree.csv, plot.csv, lf.csv, AE
# Output: List of plots with extreme values in tree volume/hectare
#-------------------------------------------------------------------------------------#

# Seleccion of Key variabes of Tree database
tree_liveKV <-tree_live[,c("plot_plot_id","subplot_subp_id","tree_lfid", "nlcl",
                           "tree_status","exp_volume_m3","tree_dia")]



# Selecction of trees with 30cm > DBH >= 10cm
tree_liveKV<-tree_liveKV[tree_liveKV$tree_dia<30|tree_liveKV$tree_dia>=10,]
dim(tree_liveKV)

# Compute of Volume at sub-plot level
VOLdataSpl <- aggregate(exp_volume_m3~plot_plot_id+subplot_subp_id+nlcl,
                        data=tree_liveKV, sum)

#  Merge of "VOLdataSpl" and "ai_sn" (subplot level sampled area) to get Volume at subplot-lf level
VOLdataSplF <-merge(VOLdataSpl, ai_sn, by.x=c("plot_plot_id","subplot_subp_id","nlcl"),
                    by.y=c("plot_plot_id","subp_id","nlcl"),all = T)
```

```
## Computing of "VOL at SupPlot-lf" and "VOL_m3Ha"
VOLdataSplF$VOL_m3Ha<- (VOLdataSplF$exp_volume_m3)/VOLdataSplF$ai
# convert the NAs in VOL_tonHa into 0
VOLdataSplF[,6][is.na(VOLdataSplF[,6])] <- 0


### Identification of Outliers
# A table of statistics (mean and sd) is done
f77 <- function(x) c(n = length(x), Mean = mean(x), SD = sd(x))
BasicSat <- aggregate(VOL_m3Ha ~ nlcl,
                      data=VOLdataSplF[!is.na(ifelse(VOLdataSplF$VOL_m3Ha==Inf,NA,
                                                     VOLdataSplF$VOL_m3Ha)),], FUN=f77)
BasicSat$n<-BasicSat$VOL_m3Ha[,1]
BasicSat$Mean<-BasicSat$VOL_m3Ha[,2]
BasicSat$SD<-BasicSat$VOL_m3Ha[,3]

# Tree database and Statistical database are merged by sientific name
VOLdataSplFSt<-merge(VOLdataSplF,BasicSat[,c("nlcl", "n", "Mean", "SD")],
                     by.x="nlcl", by.y="nlcl",all.x=T)
VOLdataSplFStF <- VOLdataSplFSt[VOLdataSplFSt$n>3,]
```

```
# The Z value is estimated and Possible Outliers are identified
VOLdataSplFStF$Z<-((VOLdataSplFStF$VOL_m3Ha-VOLdataSplFStF$Mean)/
                   VOLdataSplFStF$SD)
VOLdataSplFStF$PO<-ifelse(VOLdataSplFStF$Z>4,1,0)
dim(VOLdataSplFStF)
# Identification od Outliers
VOLdataSplFStFf<-VOLdataSplFStF[!is.na(VOLdataSplFStF$PO) & VOLdataSplFStF$PO==1,]


### Possible Outliers are summarized and saved
# A Database of possible OL is created
VOLdataSplFStFf$issue <- paste("LCC:", VOLdataSplFStFf$nlcl,
                               "vol:", round(VOLdataSplFStFf$VOL_m3Ha,1),
                               "Mean:",round(VOLdataSplFStFf$Mean,1),
                               "Z:",round(VOLdataSplFStFf$Z,1),sep = " ")
VOLdataSplFStFf$cp <- "CP 7.7"
VOLdataSplFStFf$cat <- "Outlier in Medium tree (30cm > DBH >= 10cm) Volume density"
VOLdataSplFStFf <- merge(VOLdataSplFStFf, team_name, by="plot_plot_id", all.x=T)
names(VOLdataSplFStFf)[1] <- "plot_id"

# Prepare the output table
if(dim(VOLdataSplFStFf)[1] == 0){
```

```
 output7.7 <- data.frame(cp="CP 7.7",cat="Outlier in Medium tree (30cm>DBH>=10cm) Vol density",plot_id=NA,
                         issue="No outlier in Medium tree (30cm>DBH>=10cm) Vol density", team = NA,
                         stringsAsFactors = F)} else {
 output7.7 <- VOLdataSplFStFf[,c("cp","cat","plot_id","issue", "team")]}


# save the output,
write.csv(output7.7,
          file = "./QAQC_outputs/7.7_Possible outlier in Medium tree (DBH 10cm or above) vol density.csv",
          row.names = F)


#-------------------------------------------------------------------------------#
# CP 7.8: Outliers (extreme values) in standing dead tree volume/ha ----------
# Input: tree.csv, volume models
# Output: List of plots with outliers (extreme values) in volume/ha of standing
# dead trees
#-------------------------------------------------------------------------------#
# Select the key variables from treesap data
sdtKV <- treesap[treesap$tree_status==2,][,c("plot_plot_id", "subplot_subp_id", "nlcl",
                                             "tree_dia", "tree_status",
                                             "tree_spp_scientific_name", "tree_total_lgt")]
```

```
# Filter out the trees not having diameter or length
sdtKVf <- sdtKV[!is.na(sdtKV$tree_dia) &
                !is.na(sdtKV$tree_total_lgt),]

# Compute volume at the tree level
sdtKVf$tree_dia_m <- sdtKVf$tree_dia/100
sdtKVf$Vol_m3 <- pi*((sdtKVf$tree_dia_m^2)/4)*sdtKVf$tree_total_lgt*0.5

# expand the volume from M plot to L plot
sdtKVf$exp_Vol_m3 <- ifelse(sdtKVf$tree_dia<30, sdtKVf$Vol_m3*((19^2*pi)/(8^2*pi)), #volume expansion for different plot size
                            sdtKVf$Vol_m3*1)
# Dead tree volume at subplot level
sdtVolSpl <- aggregate(data=sdtKVf, exp_Vol_m3~plot_plot_id+subplot_subp_id+nlcl, sum)
# Merge the sampled area for standing dead trees
sdtVolSplF <- merge(sdtVolSpl, ai_sn, by.x=c("plot_plot_id", "subplot_subp_id", "nlcl"),
                    by.y = c("plot_plot_id", "subp_id", "nlcl"), all=T)
# compute the dead tree volume per hectare
sdtVolSplF$sdtVol_Ha <- sdtVolSplF$exp_Vol_m3/sdtVolSplF$ai
# convert the NAs in "bamVol_Ha" of bamVolSplF into 0
sdtVolSplF[,6][is.na(sdtVolSplF[,6])] <- 0
```

```
### Identification of Outliers
# A table of statistics (mean and sd) is done
f78 <- function(x) c(n = length(x), Mean = mean(x), SD = sd(x))
BasicSat <- aggregate(sdtVol_Ha ~ nlcl,
                      data=sdtVolSplF[!is.na(ifelse(sdtVolSplF$sdtVol_Ha==Inf,NA,
                                                    sdtVolSplF$sdtVol_Ha)),], FUN=f78)
BasicSat$n<-BasicSat$sdtVol_Ha[,1]
BasicSat$Mean<-BasicSat$sdtVol_Ha[,2]
BasicSat$SD<-BasicSat$sdtVol_Ha[,3]

# Dead tree database and Statistical database are merged by nlcl
sdtVolSplFSt<-merge(sdtVolSplF, BasicSat[,c("nlcl", "n", "Mean", "SD")],
                    by.x="nlcl", by.y="nlcl",all.x=T)
sdtVolSplFStF <- sdtVolSplFSt[sdtVolSplFSt$n>3,]
# The Z value is estimated and Possible Outliers are identified
sdtVolSplFStF$Z<-((sdtVolSplFStF$sdtVol_Ha-sdtVolSplFStF$Mean)/
                    sdtVolSplFStF$SD)
sdtVolSplFStF$PO<-ifelse(sdtVolSplFStF$Z>4,1,0)
dim(sdtVolSplFStF)
```

```
# Identification of Outliers
sdtVolSplFStFf<-sdtVolSplFStF[!is.na(sdtVolSplFStF$PO) & sdtVolSplFStF$PO==1,]


### Possible Outliers are summarized and saved
# A Database of possible OL is created
sdtVolSplFStFf$issue <- paste("LCC:", sdtVolSplFStFf$nlcl,
                        "SdtVol_Ha:", round(sdtVolSplFStFf$sdtVol_Ha,1),
                        "Mean:",round(sdtVolSplFStFf$Mean,1),
                        "Z:",round(sdtVolSplFStFf$Z,1),sep = " ")
sdtVolSplFStFf$cp <- "CP 7.8"
sdtVolSplFStFf$cat <- "Outlier in dead tree volume per hectare"
sdtVolSplFStFf <- merge(sdtVolSplFStFf, team_name, by="plot_plot_id", all.x=T)
names(sdtVolSplFStFf)[1] <- "plot_id"

# Prepare the output table
if(dim(sdtVolSplFStFf)[1] == 0){
  output7.8 <- data.frame(cp="CP 7.8",cat="Outlier in dead tree volume per hectare",plot_id=NA,
                    issue="No outlier in sdt Volume per hectare", team = NA,
                    stringsAsFactors = F)} else {
  output7.8 <- sdtVolSplFStFf[,c("cp","cat","plot_id","issue", "team")]}
```

```
# export the outputs
write.csv(output7.8, file = "./QAQC_outputs/7.8_outliners in Standing dead tree volume per hectare.csv",
          row.names = F)



#-------------------------------------------------------------------------------#
# CP  7.9 (14.): Outliers (extreme values) in standing dead tree AGB/ha----------
# Input: tree.csv
# Output: List of plots with outliers (extreme values) of standing tree biomass/hectare
#-------------------------------------------------------------------------------#
sdt_biom <- treesap[which(treesap$tree_status==2&
                          !is.na(treesap$tree_total_lgt)),] #sdt_biom = standing dead tree biomass
sdt_biom <- merge(sdt_biom, plot_info[,c("plot_id", "location_zone")],
                  by.x="plot_plot_id", by.y="plot_id", all.x=T)

#add reduced wood density for decay class of std biomass
#prepare the final WD using the reduction factor #followed Harmon et al. 2011 and AE raw data
sdt_biom$wd_red_fac <- ifelse(sdt_biom$tree_decay==1, 0.99,
                              ifelse(sdt_biom$tree_decay==2, 0.80, 0.54))
sdt_biom$wd_final <- sdt_biom$wd*sdt_biom$wd_red_fac
sdt_biom$wd_final_kg_m3 <- sdt_biom$wd_final*1000  #converting wd g/cm3 to kg/m3
```

```
# Estimating Standing dead tree AGB using Chaves et al. 2014 and WD reduction factor----#
#calculate sdt biomass
sdt_biom$tree_D2H=sdt_biom$tree_total_lgt*(sdt_biom$tree_dia)^2
sdt_biom$tree_agb_kg <- 0.0673*(sdt_biom$wd_final*sdt_biom$tree_D2H)^(0.976)


# expand the biomass from M plot to L plot
sdt_biom$exp_tree_agb_kg <- ifelse(sdt_biom$tree_dia<30, sdt_biom$tree_agb_kg*(19^2/8^2),
                           sdt_biom$tree_agb_kg*1)
# sum biomass at subplot level
sdt_biomSpl <- aggregate(data=sdt_biom, exp_tree_agb_kg~plot_plot_id+subplot_subp_id+tree_lfid, sum)
# add sampled area at subplot level and nlcl level
sdt_biomSpl <- merge(sdt_biomSpl, subplot_prop[,c("plot_plot_id", "subplot_subp_id", "lf_subp_lfno", "lf_l_percent")],
                by.x=c("plot_plot_id", "subplot_subp_id","tree_lfid"),
                by.y = c("plot_plot_id", "subplot_subp_id", "lf_subp_lfno"),
                all.x=T)
# compute sampled area by plot, subplot and nlcl
sdt_biomSpl$ai <- ((sdt_biomSpl$lf_l_percent/100)*(pi*19^2))/10000
# compute biomass density
sdt_biomSpl$sdt_agb_t_ha <- sdt_biomSpl$exp_tree_agb_kg/sdt_biomSpl$ai
# add nlcl to sdt_biomSpl
```

```
sdt_biomSplF <- merge(sdt_biomSpl, lf[,c("plot_plot_id", "lf_id", "nlcl")],
                  by.x=c("plot_plot_id", "tree_lfid"), by.y=c("plot_plot_id", "lf_id"),
                  all.x=T)


### Identification of Outliers
# A table of statistics (mean and sd) is done
f79 <- function(x) c(n = length(x), Mean = mean(x), SD = sd(x))
BasicSat <- aggregate(sdt_agb_t_ha ~ nlcl,
                  data=sdt_biomSplF[!is.na(ifelse(sdt_biomSplF$sdt_agb_t_ha==Inf,NA,
                                        sdt_biomSplF$sdt_agb_t_ha)),], FUN=f79)
BasicSat$n<-BasicSat$sdt_agb_t_ha[,1]
BasicSat$Mean<-BasicSat$sdt_agb_t_ha[,2]
BasicSat$SD<-BasicSat$sdt_agb_t_ha[,3]

# Dead tree database and Statistical database are merged by nlcl
sdt_biomSplF<-merge(sdt_biomSplF, BasicSat[,c("nlcl", "n", "Mean", "SD")],
                by.x="nlcl", by.y="nlcl",all.x=T)
sdt_biomSplF <- sdt_biomSplF[sdt_biomSplF$n>3,]
# The Z value is estimated and Possible Outliers are identified
sdt_biomSplF$Z<-((sdt_biomSplF$sdt_agb_t_ha-sdt_biomSplF$Mean)/
               sdt_biomSplF$SD)
```

```
sdt_biomSplF$PO<-ifelse(sdt_biomSplF$Z>4,1,0)
dim(sdt_biomSplF)

# Identification of Outliers
sdt_biomSplFf<-sdt_biomSplF[!is.na(sdt_biomSplF$PO) & sdt_biomSplF$PO==1,]

### Possible Outliers are summarized and saved
# A Database of possible OL is created
sdt_biomSplFf$issue <- paste("LCC:", sdt_biomSplFf$nlcl,
                             "SdtAgb_Ha:", round(sdt_biomSplFf$sdt_agb_t_ha,1),
                             "Mean:",round(sdt_biomSplFf$Mean,1),
                             "Z:",round(sdt_biomSplFf$Z,1),sep = " ")
sdt_biomSplFf$cp <- "CP 7.9"
sdt_biomSplFf$cat <- "Possible outlier in dead tree AGB per hectare"
sdt_biomSplFf <- merge(sdt_biomSplFf, team_name, by="plot_plot_id", all.x=T)
names(sdt_biomSplFf)[1] <- "plot_id"

# Prepare the output table
if(dim(sdt_biomSplFf)[1] == 0){
  output7.9 <- data.frame(cp="CP 7.9",cat="Possible outlier in dead tree AGB per hectare",plot_id=NA,
                          issue="No outlier in sdt AGB per hectare", team = NA,
```

```
                    stringsAsFactors = F)} else {
  output7.9 <- sdt_biomSplFf[,c("cp","cat","plot_id","issue", "team")]}

# export the outputs
write.csv(output7.9,
        file = "./QAQC_outputs/7.9_outliners in Standing dead tree AGB per hectare.csv",
        row.names = F)



#-------------------------------------------------------------------------------------#
# CP 7.10 (8.1): Outliers (extreme values) in dead matter ton/hectare for CWD ---------
# Input: cwd.csv
# Output: List of plots with outliers in dead matter/hectare for CWD
#-------------------------------------------------------------------------------------#
# use the dead matter in CWD computed in the data management section
# convert the NAs into zero
cwdBiomSplLfF[,5][is.na(cwdBiomSplLfF[,5])] <- 0

### Identification of Outliers
# A table of statistics (mean and sd) is done
f710 <- function(x) c(n = length(x), Mean = mean(x), SD = sd(x))
```

```
BasicSat <- aggregate(cwd_biom_ha ~ nlcl,
                      data=cwdBiomSplLfF[!is.na(ifelse(cwdBiomSplLfF$cwd_biom_ha==Inf,NA,
                                                       cwdBiomSplLfF$cwd_biom_ha)),], FUN=f710)
BasicSat$n<-BasicSat$cwd_biom_ha[,1]
BasicSat$Mean<-BasicSat$cwd_biom_ha[,2]
BasicSat$SD<-BasicSat$cwd_biom_ha[,3]

# CWD database and Statistical database are merged by nlcl
cwdBiomSplLfFSt<-merge(cwdBiomSplLfF, BasicSat[,c("nlcl", "n", "Mean", "SD")],
                       by.x="nlcl", by.y="nlcl",all.x=T)
cwdBiomF <- cwdBiomSplLfFSt[cwdBiomSplLfFSt$n>3,]

# The Z value is estimated and Possible Outliers are identified
cwdBiomF$Z<-((cwdBiomF$cwd_biom_ha-cwdBiomF$Mean)/cwdBiomF$SD)
cwdBiomF$PO<-ifelse(cwdBiomF$Z>4,1,0)
dim(cwdBiomF)

# Identification of Outliers
cwdBiomFf<-cwdBiomF[!is.na(cwdBiomF$PO) & cwdBiomF$PO==1,]

### Possible Outliers are summarized and saved
```

```
# A Database of possible OL is created
cwdBiomFf$issue <- paste("LCC:", cwdBiomFf$nlcl,
                         "cwdBiom_Ha:", round(cwdBiomFf$cwd_biom_ha,1),
                         "Mean:",round(cwdBiomFf$Mean,1),
                         "Z:",round(cwdBiomFf$Z,1),sep = " ")
cwdBiomFf$cp <- "CP 7.10"
cwdBiomFf$cat <- "Outlier in CWD biomass per hectare"
cwdBiomFf <- merge(cwdBiomFf, team_name, by="plot_plot_id", all.x=T)
names(cwdBiomFf)[1] <- "plot_id"

# Prepare the output table
if(dim(cwdBiomFf)[1] == 0){
  output7.10 <- data.frame(cp="CP 7.10",cat="Outlier in CWD biomass per hectare",plot_id=NA,
                           issue="No outlier in CWD biomass per hectare", team = NA,
                           stringsAsFactors = F)} else {
  output7.10 <- cwdBiomFf[,c("cp","cat","plot_id","issue", "team")]}

# export the outputs
write.csv(output7.10, file ="./QAQC_outputs/7.10_outliners in CWD biomass per hectare.csv",
          row.names = F)
```

```
#-------------------------------------------------------------------------------#
# CP 7.11 (8.2): Outliers (extreme values) in dead matter/hectare for FWD ----------
# Input: fwd.csv
# Output: List of plots with outliers in dead matter/hectare for FWD
#-------------------------------------------------------------------------------#
# use the dead matter in FWD computed in the data management section
head(fwdBiomSplf)

### Identification of Outliers
# A table of statistics (mean and sd) is done
f711 <- function(x) c(n = length(x), Mean = mean(x), SD = sd(x))
BasicSat <- aggregate(fwd_biom_t_ha ~ nlcl,
                      data=fwdBiomSplf[!is.na(ifelse(fwdBiomSplf$fwd_biom_t_ha==Inf,NA,
                                                     fwdBiomSplf$fwd_biom_t_ha)),], FUN=f711)
BasicSat$n<-BasicSat$fwd_biom_t_ha[,1]
BasicSat$Mean<-BasicSat$fwd_biom_t_ha[,2]
BasicSat$SD<-BasicSat$fwd_biom_t_ha[,3]

# CWD database and Statistical database are merged by nlcl
fwdBiomSplfSt<-merge(fwdBiomSplf, BasicSat[,c("nlcl", "n", "Mean", "SD")],
```

```
                    by.x="nlcl", by.y="nlcl",all.x=T)
fwdBiomF <- fwdBiomSplfSt[fwdBiomSplfSt$n>3,]

# The Z value is estimated and Possible Outliers are identified
fwdBiomF$Z<-((fwdBiomF$fwd_biom_t_ha-fwdBiomF$Mean)/fwdBiomF$SD)
fwdBiomF$PO<-ifelse(fwdBiomF$Z>4,1,0)
dim(fwdBiomF)

# Identification of Outliers
fwdBiomFf<-fwdBiomF[!is.na(fwdBiomF$PO) & fwdBiomF$PO==1,]

### Possible Outliers are summarized and saved
# A Database of possible OL is created
fwdBiomFf$issue <- paste("LCC:", fwdBiomFf$nlcl,
                    "fwdBiom_Ha:", round(fwdBiomFf$fwd_biom_t_ha,1),
                    "Mean:",round(fwdBiomFf$Mean,1),
                    "Z:",round(fwdBiomFf$Z,1),sep = " ")
fwdBiomFf$cp <- "CP 7.11"
fwdBiomFf$cat <- "Outlier in CWD biomass per hectare"
fwdBiomFf <- merge(fwdBiomFf, team_name, by="plot_plot_id", all.x=T)
names(fwdBiomFf)[1] <- "plot_id"
```

```
# Prepare the output table
if(dim(fwdBiomFf)[1] == 0){
  output7.11 <- data.frame(cp="CP 7.11",cat="Possible outlier in FWD biomass per hectare",plot_id=NA,
                           issue="No outlier in FWD biomass per hectare", team = NA,
                           stringsAsFactors = F)} else {
  output7.11 <- fwdBiomFf[,c("cp","cat","plot_id","issue", "team")]}

# export the outputs
write.csv(output7.11,
          file = "./QAQC_outputs/7.11_Possible outliners in FWD biomass per hectare.csv",
          row.names = F)



#-------------------------------------------------------------------------------------#
# CP 7.12: Strange relationships between Trees/hectare vs Dead Matter ----------------
# Input: cwd.csv, fwd.csv, tree.csv
# Output: List of plots with strange relationships between trees/hectare Vs Dead Matter
#-------------------------------------------------------------------------------------#
####### data preparation
```

```
### compute the stem density from treesap

# select only key variables

treesap_liveKV <- treesap_live[,c("plot_plot_id", "subplot_subp_id", "nlcl",

                          "tree_status", "tree_dia")]

# add a column named "tree_count" inn treesasp_live

treesap_liveKV$tree_count <- 1


# Expand tree and sapling count from S and M plot to L plot

treesap_liveKV$exp_stem_no <- ifelse(treesap_liveKV$tree_dia<10, treesap_liveKV$tree_count*((19^2*pi)/(2.5^2*pi)), #biomss expansion for
different plot size

                              ifelse(treesap_liveKV$tree_dia<30, treesap_liveKV$tree_count*((19^2*pi)/(8^2*pi)),

                                  treesap_liveKV$tree_count*((19^2*pi)/(19^2*pi))))

# tree stem number at plot, subplot and nlcl level

stemdataSpl <- aggregate(data=treesap_liveKV, exp_stem_no~plot_plot_id+subplot_subp_id+nlcl, sum)


#  Merge of "stemdataSpl" and "ai_sn" (subplot level sampled area) to get Stem_Ha at subplot-lf level

stemdataSplF <-merge(stemdataSpl, ai_sn, by.x=c("plot_plot_id","subplot_subp_id","nlcl"),

                by.y=c("plot_plot_id","subp_id","nlcl"),all = T)


## Computing of "Stem_Ha"

stemdataSplF$Stem_Ha<- (stemdataSplF$exp_stem_no)/stemdataSplF$ai
```

```
# convert the NAs in "Stem_Ha" of stemdata into 0
stemdataSplF[,6][is.na(stemdataSplF[,6])] <- 0



#### comopute the dry matter from cwd and fwd biomass t/ha
DrMat <- merge(cwdBiomSplLfF[,c("plot_plot_id", "subplot_subp_id", "nlcl","cwd_biom_ha")],
               fwdBiomSplf[,c("plot_plot_id", "subplot_subp_id", "nlcl","fwd_biom_t_ha")],
               by=c("plot_plot_id", "subplot_subp_id", "nlcl"), all.x=T)

# convert the NA ddMat into zero
DrMat[,4:5][is.na(DrMat[,4:5])] <- 0

# sum up the dry matter
DrMat$ddMat_tonHa <- DrMat$cwd_biom_ha+DrMat$fwd_biom_t_ha

### merge the dry matter and tree density data
TdDrMat <- merge(stemdataSplF[,c("plot_plot_id", "subplot_subp_id", "nlcl", "Stem_Ha")],
                 DrMat[,c("plot_plot_id", "subplot_subp_id", "nlcl", "ddMat_tonHa")],
                 by=c("plot_plot_id", "subplot_subp_id", "nlcl"),
                 all.x=T) # TdDrMat = Tree density and dry matter
```

```
### identify the possible inconsistencies between tree/ha and dead matter/ha
# dead matter/ha > 0 but tree/ha = 0; in the forest only
TdDrMatIC <- TdDrMat[TdDrMat$ddMat_tonHa>0 & TdDrMat$Stem_Ha==0 &
                     (TdDrMat$nlcl=="FH" | TdDrMat$nlcl == "NMF"|TdDrMat$nlcl == "NMF"|
                       TdDrMat$nlcl == "FDp"|TdDrMat$nlcl=="FMp"|TdDrMat$nlcl=="FP"|
                       TdDrMat$nlcl=="FPr"|TdDrMat$nlcl=="H"),]

# prepare the table with possible inconsistencies
TdDrMatIC$issue <- "Tree/ha = 0 but dead matter/ha > 0"
TdDrMatIC$cp <- "CP 7.12"
TdDrMatIC$cat <- "Inconsistencies btwn dead matter and tree density"
TdDrMatIC <- merge(TdDrMatIC, team_name, by="plot_plot_id", all.x=T)
names(TdDrMatIC)[1] <- "plot_id"

# Prepare the output table
if(dim(TdDrMatIC)[1] == 0){
  output7.12 <- data.frame(cp="CP 7.12",cat="Inconsistencies btwn dead matter and tree density",plot_id=NA,
                           issue="No inconsistencies in Tree/ha and dead matter", team = NA,
                           stringsAsFactors = F)} else {
  output7.12 <- TdDrMatIC[,c("cp","cat","plot_id","issue", "team")]}
```

```
# export the outputs
write.csv(output7.12,
        file = "./QAQC_outputs/7.12_Possible inconsistencies in tree per ha and dead matter per ha.csv",
        row.names = F)



#-------------------------------------------------------------------------------------#
# CP 7.13: Outliers (extreme values) in Bamboo volume/hectare -------------------------
# Input: bamboo.csv
# Output: List of plots with outliers (extreme values) in bamboo volume/hectare
#-------------------------------------------------------------------------------------#
####### Prepare the data
#work on bamboos from completely sampled subplots only
bambooF <- merge(bamboo, subplot_info[,c("plot_plot_id", "subp_id", "subp_details_subp_status_label")],
              by.x=c("plot_plot_id", "subplot_subp_id"), by.y=c("plot_plot_id", "subp_id"), all.x=T)
bambooF <- bambooF[bambooF$subp_details_subp_status_label=="Sampled completely",]

# select the key variables
bambooFKV <- bambooF[,c("plot_plot_id", "subplot_subp_id", "bamboo_sp_scientific_name",
                    "bamboo_lf", "bamboo_dia", "bamboo_lgt", "bamboo_number")]
# Compute bamboo volume following NFA
```

```
bambooFKV$bamboo_dia_m <- bambooFKV$bamboo_dia/100

bambooFKV$bv_m3 <- ((bambooFKV$bamboo_dia_m^2-
(bambooFKV$bamboo_dia_m*0.7)^2)/4)*pi*bambooFKV$bamboo_lgt*0.8*bambooFKV$bamboo_number #bv_m3 = bamboo volume; bamboo form factor=0.8


#expand the bamboo volume from M plot to L plot

bambooFKV$exp_bv_m3 <- bambooFKV$bv_m3*((19^2*pi)/(8^2*pi)) #exp_bv_m3  = expanded bamboo volume for L plot as bamboo is recorded only from M plot


# Merge nlcl into bambooFKV

bambooFKVLf <- merge(bambooFKV, lf[,c("plot_plot_id", "lf_id", "nlcl")],

        by.x=c("plot_plot_id", "bamboo_lf"),

        by.y=c("plot_plot_id", "lf_id"), all.x=T)


# Bamboo volume at subplot level

bamVolSpl <- aggregate(data=bambooFKVLf, exp_bv_m3~plot_plot_id+subplot_subp_id+nlcl+bamboo_sp_scientific_name, sum)


# Merge the sampled area for bamboo

bamVolSplF <- merge(bamVolSpl, ai_sn, by.x=c("plot_plot_id", "subplot_subp_id", "nlcl"),

        by.y = c("plot_plot_id", "subp_id", "nlcl"), all=T)


# compute the bamboo volume per hectare
```

```
bamVolSplF$bamVol_Ha <- bamVolSplF$exp_bv_m3/bamVolSplF$ai
# convert the NAs in "bamVol_Ha" of bamVolSplF into 0
bamVolSplF[,6][is.na(bamVolSplF[,6])] <- 0


### Identification of Outliers
# A table of statistics (mean and sd) is done
f713 <- function(x) c(n = length(x), Mean = mean(x), SD = sd(x))
BasicSat <- aggregate(bamVol_Ha ~ nlcl,
                      data=bamVolSplF[!is.na(ifelse(bamVolSplF$bamVol_Ha==Inf,NA,
                                                    bamVolSplF$bamVol_Ha)),], FUN=f713)
BasicSat$n<-BasicSat$bamVol_Ha[,1]
BasicSat$Mean<-BasicSat$bamVol_Ha[,2]
BasicSat$SD<-BasicSat$bamVol_Ha[,3]

# Bamboo database and Statistical database are merged by nlcl
bamVolSplFSt<-merge(bamVolSplF, BasicSat[,c("nlcl", "n", "Mean", "SD")],
                    by.x="nlcl", by.y="nlcl",all.x=T)
bamVolSplFStF <- bamVolSplFSt[bamVolSplFSt$n>3,]
# The Z value is estimated and Possible Outliers are identified
bamVolSplFStF$Z<-((bamVolSplFStF$bamVol_Ha-bamVolSplFStF$Mean)/
```

```
                bamVolSplFStF$SD)
bamVolSplFStF$PO<-ifelse(bamVolSplFStF$Z>4,1,0)
dim(bamVolSplFStF)

# Identification of Outliers
bamVolSplFStFf<-bamVolSplFStF[!is.na(bamVolSplFStF$PO) & bamVolSplFStF$PO==1,]

### Possible Outliers are summarized and saved
# A Database of possible OL is created
bamVolSplFStFf$issue <- paste("LCC:", bamVolSplFStFf$nlcl,
                        "BamVol_Ha:", round(bamVolSplFStFf$bamVol_Ha,1),
                        "Mean:",round(bamVolSplFStFf$Mean,1),
                        "Z:",round(bamVolSplFStFf$Z,1),sep = " ")
bamVolSplFStFf$cp <- "CP 7.13"
bamVolSplFStFf$cat <- "Outlier in bamboo volume per hectare"
bamVolSplFStFf <- merge(bamVolSplFStFf, team_name, by="plot_plot_id", all.x=T)
names(bamVolSplFStFf)[1] <- "plot_id"

# Prepare the output table
if(dim(bamVolSplFStFf)[1] == 0){
  output7.13 <- data.frame(cp="CP 7.13",cat="Outlier in bamboo Volume per hectare",plot_id=NA,
```

```
                issue="No outlier in bam Vol per hectare", team = NA,
                stringsAsFactors = F)} else {
  output7.13 <- bamVolSplFStFf[,c("cp","cat","plot_id","issue", "team")]}

# export the outputs
write.csv(output7.13, file = "./QAQC_outputs/7.13_Possible outliners in Bamboo volume per hectare.csv",
          row.names = F)

#-------------------------------------------------------------------------------#
# CP 7.14: Outliers (extreme values) in Stump per hectare ------------------------------
# Input: tree.csv, lf.csv, lf_subp_prop.csv
# Output: List of plots with outliers (extreme values) in Stump per hectare
#-------------------------------------------------------------------------------#
###prepare the data
stump <-treesap[which(treesap$tree_status_label=="Stump (alive)"|
                treesap$tree_status_label=="Stump (dead)"),]

# Select the key variables
stumpKV <- stump[,c("plot_plot_id", "subplot_subp_id", "tree_lfid", "tree_spp_scientific_name",
                "tree_status", "tree_dia", "nlcl")]
```

```
# tree seedling number at plot, subplot and nlcl level
stumpKV$stump_no <- 1
#expand the stump number from M plot to L plot
stumpKV$exp_stump_no <- ifelse(stumpKV$tree_dia<30, stumpKV$stump_no*((pi*19^2)/(pi*8^2)),
                        stumpKV$stump_no*1)

#compute the number of stump at plot, subplot and nlcl level
stumpKVSpl <- aggregate(data=stumpKV, exp_stump_no~plot_plot_id+subplot_subp_id+nlcl, sum)

### Merge the sampled area for L plot "ai" from ai_sn
stumpKVSplF <- merge(stumpKVSpl, ai_sn, by.x = c("plot_plot_id", "subplot_subp_id", "nlcl"),
                  by.y = c("plot_plot_id", "subp_id", "nlcl"), all = T)

## Computing "Stump_Ha"
stumpKVSplF$Stump_Ha<- (stumpKVSplF$exp_stump_no)/stumpKVSplF$ai
# convert the NAs in "Stem_Ha" of stemdata into 0
stumpKVSplF[,6][is.na(stumpKVSplF[,6])] <- 0



### Identification of Outliers
# A table of statistics (mean and sd) is done
```

```
f714 <- function(x) c(n = length(x), Mean = mean(x), SD = sd(x))
BasicSat <- aggregate(Stump_Ha ~ nlcl,
                      data=stumpKVSplF[!is.na(ifelse(stumpKVSplF$Stump_Ha==Inf,NA,
                                                     stumpKVSplF$Stump_Ha)),], FUN=f714)
BasicSat$n<-BasicSat$Stump_Ha[,1]
BasicSat$Mean<-BasicSat$Stump_Ha[,2]
BasicSat$SD<-BasicSat$Stump_Ha[,3]

# Stump database and Statistical database are merged by nlcl
stumpKVSplFSt<-merge(stumpKVSplF, BasicSat[,c("nlcl", "n", "Mean", "SD")],
                     by.x="nlcl", by.y="nlcl",all.x=T)
stumpKVSplFStF <- stumpKVSplFSt[stumpKVSplFSt$n>3,]
# The Z value is estimated and Possible Outliers are identified
stumpKVSplFStF$Z<-((stumpKVSplFStF$Stump_Ha-stumpKVSplFStF$Mean)/
                   stumpKVSplFStF$SD)
stumpKVSplFStF$PO<-ifelse(stumpKVSplFStF$Z>4,1,0)
dim(stumpKVSplFStF)

# Identification of Outliers
stumpKVSplFStFf<-stumpKVSplFStF[!is.na(stumpKVSplFStF$PO) & stumpKVSplFStF$PO==1,]
```

```
### Possible Outliers are summarized and saved
# A Database of possible OL is created
stumpKVSplFStFf$issue <- paste("LCC:", stumpKVSplFStFf$nlcl,
                        "Stump_Ha:", round(stumpKVSplFStFf$Stump_Ha,1),
                        "Mean:",round(stumpKVSplFStFf$Mean,1),
                        "Z:",round(stumpKVSplFStFf$Z,1),sep = " ")
stumpKVSplFStFf$cp <- "CP 7.14"
stumpKVSplFStFf$cat <- "Possible outlier in Stump per hectare"
stumpKVSplFStFf <- merge(stumpKVSplFStFf, team_name, by="plot_plot_id", all.x=T)
names(stumpKVSplFStFf)[1] <- "plot_id"

# Prepare the output table
if(dim(stumpKVSplFStFf)[1] == 0){
  output7.14 <- data.frame(cp="CP 7.14",cat="Outlier in Stump per hectare",plot_id=NA,
                    issue="No outlier in Stump per hectare", team = NA,
                    stringsAsFactors = F)} else {
  output7.14 <- stumpKVSplFStFf[,c("cp","cat","plot_id","issue", "team")]}

# export the outputs
write.csv(output7.14, file = "./QAQC_outputs/7.14_Possible outliers in Stump per hectare.csv",
          row.names = F)
```

```
#-------------------------------------------------------------------------------#
# CP 7.15: Extreme values in Seedling number/hectare by plot ---------------------------
# Input: seedling.csv
# Output: List of plots with extreme values in seedling/hectare
#-------------------------------------------------------------------------------#
####### Prepare seedling data
# Select completely sampled seedlings plots only
seedlingF <- merge(seedling, subplot_info[,c("plot_plot_id", "subp_id", "subp_details_subp_status_label")],
                                  by.x=c("plot_plot_id", "subplot_subp_id"),
                                  by.y=c("plot_plot_id", "subp_id"), all.x=T)
seedlingF <- seedlingF[seedlingF$subp_details_subp_status_label=="Sampled completely",]

#merge seedlings and lf_id from subplot_prop
seedlingFLf <- merge(seedlingF[which((!is.na(seedlingF$seedling_count))&
                              seedlingF$seedling_spp_scientific_name!=""),],
                subplot_prop[subplot_prop$lf_s_percent==100,
                          c("plot_plot_id","subplot_subp_id","lf_subp_lfno")],
                by = c("plot_plot_id","subplot_subp_id"), all.x=TRUE)
```

```
#extract plots missing lf_id
seedlingNoLf <- seedlingFLf[is.na(seedlingFLf$lf_subp_lfno),]
write.csv(seedlingNoLf[,c("plot_plot_id", "subplot_subp_id","seedling_spp_scientific_name",
                          "seedling_count", "lf_subp_lfno")],
          file="./QAQC_outputs/seedlings_plots and subplots_missed in lf_subp_prop.csv", row.names = F)

#Add nlcl to seedlingFLf
seedlingFLf <- merge(seedlingFLf, lf[,c("plot_plot_id", "lf_id", "nlcl")],
                     by.x=c("plot_plot_id", "lf_subp_lfno"),
                     by.y=c("plot_plot_id", "lf_id"), all.x=T)



# select only key variables
seedlingKV <- seedlingFLf[,c("plot_plot_id", "subplot_subp_id", "nlcl", "seedling_count")]

# tree seedling number at plot, subplot and nlcl level
seedlingKVSpl <- aggregate(data=seedlingKV, seedling_count~plot_plot_id+subplot_subp_id+nlcl, sum)

### Merge the sampled area for s plot "ai" from sup
# s plot area by plot, subplot and nlcl
s_area <- aggregate(data=sup, s_area~plot_plot_id+subplot_subp_id+nlcl, sum)
```

```
seedlingKVSplF <- merge(seedlingKVSpl, s_area, by = c("plot_plot_id", "subplot_subp_id", "nlcl"),
                        all = T)

## Computing of "Stem_Ha"
seedlingKVSplF$Seedling_Ha<- (seedlingKVSplF$seedling_count)/seedlingKVSplF$s_area
# convert the NAs in "Stem_Ha" of stemdata into 0
seedlingKVSplF[,6][is.na(seedlingKVSplF[,6])] <- 0



### Identification of Outliers
# A table of statistics (mean and sd) is done
f715 <- function(x) c(n = length(x), Mean = mean(x), SD = sd(x))
BasicSat <- aggregate(Seedling_Ha ~ nlcl,
                      data=seedlingKVSplF[!is.na(ifelse(seedlingKVSplF$Seedling_Ha==Inf,NA,
                                                        seedlingKVSplF$Seedling_Ha)),], FUN=f715)
BasicSat$n<-BasicSat$Seedling_Ha[,1]
BasicSat$Mean<-BasicSat$Seedling_Ha[,2]
BasicSat$SD<-BasicSat$Seedling_Ha[,3]

# Stem database and Statistical database are merged by nlcl
seedlingKVSplFSt<-merge(seedlingKVSplF, BasicSat[,c("nlcl", "n", "Mean", "SD")],
```

```
                by.x="nlcl", by.y="nlcl",all.x=T)
seedlingKVSplFStF <- seedlingKVSplFSt[seedlingKVSplFSt$n>3,]
# The Z value is estimated and Possible Outliers are identified
seedlingKVSplFStF$Z<-((seedlingKVSplFStF$Seedling_Ha-seedlingKVSplFStF$Mean)/
                 seedlingKVSplFStF$SD)
seedlingKVSplFStF$PO<-ifelse(seedlingKVSplFStF$Z>4,1,0)
dim(seedlingKVSplFStF)
# Identification od Outliers
seedlingKVSplFStFf<-seedlingKVSplFStF[!is.na(seedlingKVSplFStF$PO) & seedlingKVSplFStF$PO==1,]

### Possible Outliers are summarized and saved
# A Database of possible OL is created
seedlingKVSplFStFf$issue <- paste("LCC:", seedlingKVSplFStFf$nlcl,
                        "Sdling_Ha:", round(seedlingKVSplFStFf$Seedling_Ha,1),
                        "Mean:",round(seedlingKVSplFStFf$Mean,1),
                        "Z:",round(seedlingKVSplFStFf$Z,1),sep = " ")
seedlingKVSplFStFf$cp <- "CP 7.15"
seedlingKVSplFStFf$cat <- "Possible outlier in Seedling per hectare"
seedlingKVSplFStFf <- merge(seedlingKVSplFStFf, team_name, by="plot_plot_id", all.x=T)
names(seedlingKVSplFStFf)[1] <- "plot_id"
```

```
# Prepare the output table
if(dim(seedlingKVSplFStFf)[1] == 0){
  output7.15 <- data.frame(cp="CP 7.15",cat="Outlier in Seedling per hectare",plot_id=NA,
                           issue="No outlier in Seedling per hectare", team = NA,
                           stringsAsFactors = F)} else {
  output7.15 <- seedlingKVSplFStFf[,c("cp","cat","plot_id","issue", "team")]}

# export the outputs
write.csv(output7.15, file = "./QAQC_outputs/7.15_Possible outliers in seedling per hectare.csv",
          row.names = F)



#-------------------------------------------------------------------------------#
# CP  08: Soil bulk density related checks ----------------------------------------
#-------------------------------------------------------------------------------#

#-------------------------------------------------------------------------------#
# CP  8.1: NAs in bulk density database -------------------------------------------
# input: soil bulk density data;
# output: list of plots, subplot with NAs in cells for bulk density
#-------------------------------------------------------------------------------#
```

```
# Select the key variables
bdKV <- soil_bulk[,c("plot_id", "subplot_id", "lf_id", "soil_depth_cm", "bulk_density_g_cc")]


# Identify the NAs and prepare for output table
if(dim(bdKV[(is.na(bdKV$plot_id) | is.na(bdKV$subplot_id) |
        is.na(bdKV$soil_depth_cm) | is.na(bdKV$lf_id) |
        (is.na(bdKV$bulk_density_g_cc) & bdKV$remarks!="No sample")),])[1]==0){
  output8.1 <- data.frame(cp= "CP 8.1", cat="NA in data frame",
                    plot_id = NA, issue= "No NAs found", team=NA)
}else{
  bd_na <- bdKV[(is.na(bdKV$plot_id) | is.na(bdKV$subplot_id) |
                is.na(bdKV$soil_depth_cm) | is.na(bdKV$lf_id) |
                (is.na(bdKV$bulk_density_g_cc) & bdKV$remarks!="No sample")),]
  bd_na$cp <- "CP 8.1"
  bd_na$cat <- "NA in BD database"
  bd_na$issue <- paste(ifelse(is.na(bdKV$plot_id), "NA in plot_no",
                        ifelse(is.na(bdKV$subplot_id), "NA in subplot_id",
                               ifelse(is.na(bdKV$soil_depth_cm), "NA in soil depth",
                                      ifelse(is.na(bdKV$lf_id),
                                             "Na in lf","NA in bulk density")))))
  bd_na$team <- merge(bd_na, team_name, by.x="plot_id", by.y="plot_plot_id", all.x=T)
```

```
  names(bd_na)[1]<- "plot_id"
  output8.1 <- bd_na[,c("cp", "cat", "plot_id", "issue", "team")]
}

# export the outputs
write.csv(output8.1, file = "./QAQC_outputs/8.1_NAs in bulk density data.csv",row.names = F)

#-------------------------------------------------------------------------------#
# CP 8.2: Blank cells in bulk density database ----------------------------------------
# Input: soil bulk density data;
# Output: list of plots, subplot with blank cells for bulk density
#-------------------------------------------------------------------------------#
# Select the key variables
bdKV <- soil_bulk[soil_bulk$remarks!="No sample", c("plot_id", "subplot_id", "lf_id",
                                    "soil_depth_cm", "bulk_density_g_cc")]
# identify the blank cells and prepare in the output table
if(dim(bdKV[(bdKV$plot_id=="" | bdKV$subplot_id=="" |
            bdKV$soil_depth_cm=="" | bdKV$lf_id=="" |
            bdKV$bulk_density_g_cc==""),])[1]==0){
  output8.2 <- data.frame(cp= "CP 8.2", cat="Blank cells BD database",
                          plot_id = NA, issue= "No blank cells found", team=NA)
```

```
}else{
  bd_blank <- bdKV[(bdKV$plot_id=="" | bdKV$subplot_id=="" |
                    bdKV$soil_depth_cm=="" | bdKV$lf_id=="" |
                    bdKV$bulk_density_g_cc==""|bdKV$Bulk_density_g_cc==0),]
  bd_blank$cp <- "CP 8.2"
  bd_blank$cat <- "Blank cells in data frame"
  bd_blank$issue <- paste(ifelse(bdKV$plot_id=="", "Blank in plot_no",
                                 ifelse(bdKV$subplot_id=="", "Blank in subplot_no",
                                        ifelse(bdKV$soil_depth_cm=="", "Blank in soil depth",
                                               ifelse(bdKV$lf_id=="", "Blank in lf",
                                                      "Blank in bulk density")))))
  bd_blank <- merge(bd_blank, team_name, by.x="plot_id", by.y="plot_plot_id", all.x=T)
  names(bd_blank)[1]<- "plot_id"
  output8.2 <- bd_blank[,c("cp", "cat", "plot_id", "issue", "team")]}


# export the outputs
write.csv(output8.2, file = "./QAQC_outputs/8.2_Blanks in bulk density data.csv",
          row.names = F)


#-------------------------------------------------------------------------------#
# CP 8.3: Duplicate values in BD database ----------------------------------------------
```

```
#         [input: soil bulk density data;
#         output: list of plots with duplicate values.
#-------------------------------------------------------------------------------#
# Select the key variables
bdKV <- soil_bulk
bdKV$count <- 1

# CP 8.3.1: find duplicate upto bulk density
#-------------------------------------------------------------------------------#
bd_dp <-aggregate(count~plot_id+subplot_id+soil_depth_cm+lf_id+bulk_density_g_cc,
                  data = bdKV, sum)
bd_dp_1 <- bd_dp[bd_dp$count>1, ]

# add results to output
if(dim(bd_dp_1)[1]==0){
  output8.3.1 <- data.frame(cp= "CP 8.3.1", cat="Duplicate values in bulk density",
                plot_id = NA, issue= "No duplicates found up to bulk density", team=NA)
}else{
  bd_dp <- bd_dp[-as.integer(rownames(bd_dp_1)),]
  bd_dp_1$cp <- "CP 8.3.1"
  bd_dp_1$cat <- "Duplicate values in bulk density"
```

```
  bd_dp_1$issue <- "values duplicated up to BD"
  bd_dp_1 <- merge(bd_dp_1, team_name, by.x = "plot_id", by.y="plot_plot_id", all.x=T)
  names(bd_dp_1)[1] <- "plot_id"
  output8.3.1 <- bd_dp_1[,c("cp", "cat", "plot_id", "issue", "team")]
}



# CP 8.3.2 find duplicates upto lf level in BD database
#-------------------------------------------------------------------------------#
bd_dp <-aggregate(count~plot_id+subplot_id+soil_depth_cm+lf_id, data = bdKV, sum)
bd_dp_2 <- bd_dp[bd_dp$count>1,]

#add results to output
if(dim(bd_dp_2)[1]==0){
  output8.3.2 <- data.frame(cp= "CP 8.3.2", cat="Duplicate values in bulk density",
              plot_id = NA, issue= "No duplicates found up to lf", team=NA)
}else{
  bd_dp <- bd_dp[-as.integer(rownames(bd_dp_2)),]
  bd_dp_2$cp <- "CP 8.3.2"
  bd_dp_2$cat <- "Duplicate values in bulk density"
  bd_dp_2$issue <- "values duplicated up to lf"
```

```
  bd_dp_2 <- merge(bd_dp_2, team_name, by.x="plot_id", by.y = "plot_plot_id", all.x=T)
  names(bd_dp_2)[1] <- "plot_id"
  output8.3.2 <- bd_dp_2[,c("cp", "cat", "plot_id", "issue","team")]
}

# CP 8.3.3 find duplicates upto depth level in BD database
#-------------------------------------------------------------------------------------#
# sum the counts of depths per plot and subplot
bd_dp <-aggregate(count~plot_id+subplot_id+soil_depth_cm, data = bdKV, sum)
# identify more thab one depths per plot and subplot
bd_dp_3 <- bd_dp[bd_dp$count>1,]

#add results to output
if(dim(bd_dp_3)[1]==0){
  output8.3.3 <- data.frame(cp= "CP 8.3.3", cat="Duplicate values in bulk density",
              plot_id = NA, issue= "No duplicates found up to depth", team=NA)
}else{
  bd_dp <- bd_dp[-as.integer(rownames(bd_dp_3)),] #[here the problem occurs]
  bd_dp_3$cp <- "CP 8.3.3"
  bd_dp_3$cat <- "Duplicate values in bulk density"
  bd_dp_3$issue <- "values duplicated up to soil depth"
```

```
  bd_dp_3 <- merge(bd_dp_3, team_name, by.x="plot_id", by.y = "plot_plot_id", all.x=T)
  names(bd_dp_3)[1] <- "plot_id"
  output8.3.3 <- bd_dp_3[,c("cp", "cat", "plot_id", "issue", "team")]
}

# Get all the duplicates in BD database
output8.3 <- rbind(output8.3.1,output8.3.2, output8.3.3)

# export the outputs
write.csv(output8.3, file = "./QAQC_outputs/8.3_Duplicates in bulk density data.csv",
          row.names = F)

#-------------------------------------------------------------------------------#
# CP 8.4: Check the outlier in bulk density data ------------------------------------
#         [input: soil bulk density data;
#         output: list of plots, subplot with strange value of bulk density data]
#-------------------------------------------------------------------------------#
# Select the key variables
bdKV <- soil_bulk[, c("plot_id", "subplot_id", "lf_id", "soil_depth_cm", "bulk_density_g_cc")]

# 15 highest values of bulk densities
```

```
bdKV[order(-bdKV$bulk_density_g_cc),][1:15,]$bulk_density_g_cc
# Add the NLCL from LF database
bdKVLf <- merge(bdKV, lf[,c("plot_plot_id", "lf_id", "nlcl")],
                by.x=c("plot_id", "lf_id"), by.y=c("plot_plot_id", "lf_id"), all.x=T)
### A table of statistics (mean and sd) are done
f84 <- function(x) c(n = length(x), Mean = mean(x), SD = sd(x))
BasicSat <- aggregate(bulk_density_g_cc ~ nlcl, data=bdKVLf, FUN=f84)
BasicSat$n<-BasicSat$bulk_density_g_cc[,1]
BasicSat$Mean<-BasicSat$bulk_density_g_cc[,2]
BasicSat$SD<-BasicSat$bulk_density_g_cc[,3]

### BD database and Statistical database are merged by NLCL
bdKVLfSt<-merge(bdKVLf, BasicSat[,c("nlcl","n", "Mean", "SD")],by="nlcl", all.x=T)
bdKVLfStF<- bdKVLfSt[bdKVLfSt$n>3,]

### The Z value is estimated and Possible Outliers are identified
# Z is estimated
bdKVLfStF$Z<-((bdKVLfStF$bulk_density_g_cc-bdKVLfStF$Mean)/bdKVLfStF$SD)
# Possible Outliers are identify
bdKVLfStF$PO<-ifelse(abs(bdKVLfStF$Z)>4,1,0)
```

```
### Possible Outliers are summarized and saved
# A Database of possible OL is created and output table prepared
if(dim(bdKVLfStF[bdKVLfStF$PO==1,])[1]==0){
outpu8.4 <- data.frame(cp = "CP 8.4", cat = "Outliers in Bulk Density",
                       issue = "No outliers in Bulk Density", plot_id = NA, team = NA)
}else{
bdKVLfStF$cp <- "CP 8.4"
bdKVLfStF$cat <- "Outliers in Bulk Density"
bdKVLfStF$issue <- paste("Possible outliers in Bulk Density: subp_id-", bdKVLfStF$subplot_id,
                         ", LCC-", bdKVLfStF$nlcl, ", Depth-", bdKVLfStF$soil_depth_cm,
                         ", BD-",round(bdKVLfStF$bulk_density_g_cc,2),sep = "")
bdKVLfStF <- merge(bdKVLfStF, team_name, by.x="plot_id", by.y="plot_plot_id", all.x=T)
names(bdKVLfStF)[1] <- "plot_id"
output8.4 <- bdKVLfStF[!is.na(bdKVLfStF$plot_id) & bdKVLfStF$PO==1,
                       c("cp", "cat", "plot_id", "issue", "team")]}

# Export the output
write.csv(output8.4, file = "./QAQC_outputs/8.4_Possible outliers in Bulk density.csv",
          row.names = F)

#-------------------------------------------------------------------------------#
```

```
# CP 8.5: Consistency of bulk density data depth with Zone ----------------------------
#       [Note: Check if the number of bulk density data for different soil
#       depths/layer in different forest zones is appropriate or not as
#       per BFI manual]
#       [input: bulk density data;
#       output: the number of soil depth/layer not appropriate as per BFI manual]
#-------------------------------------------------------------------------------------#
# Select the key variables
bdKV <- soil_bulk[, c("plot_id", "subplot_id", "lf_id", "soil_depth_cm", "bulk_density_g_cc")]
bdKV$depth_count <- 1
# Get the number of depth by plot and subplot
bd_depths <- aggregate(depth_count~plot_id+subplot_id, data=bdKV, sum)

# Add the location zone
bd_depths <- merge(bd_depths, plot_info[, c("plot_id", "location_zone")], by= "plot_id", all.x=T)
#plot/subplot in which bulk density data found for only one layer
a <- bd_depths[which(bd_depths$depth_count < 2),]
#plot/subplot in which bulk density data found for more than 3 depths
b <- bd_depths[which(bd_depths$depth_count > 3),]
#if the bulk density data in less than 3 depths in plots/subplots of
#     sundarbans and coastal zones
```

```
c <- bd_depths[which((bd_depths$location_zone== "Sundarbans" |
                      bd_depths$location_zone=="Coastal") &
                     bd_depths$depth_count < 3),]
#If the bulk density data is in more than 2 depths in plots/subplots of
#      forest zones other than sundarbans and coastal
d <- bd_depths[which((bd_depths$location_zone!= "Sundarbans" &
                      bd_depths$location_zone!="Coastal") & bd_depths$depth_count >2),]


#prepare a data frame for all soil layer inconsistency
bd_incons <- rbind(a,b,c,d)


#add results to output
if(dim(bd_incons)[1]==0){
  output8.5 <- data.frame(cp= "CP 8.5", cat="Consistency betwn BD soil layer and Zone",
                          plot_id = 9999, issue= "Passed successfully", team=99)
}else{
  bd_incons$cp <- "CP 8.5"
  bd_incons$cat <- "Consistency betwn BD soil layer and Zone"
  bd_incons$issue <- ifelse((bd_incons$location_zone!= "Sundarbans" &
                             bd_incons$location_zone!="Coastal") &
                            bd_incons$depth_count > 2,
```

```
                    paste("BD data in 3 soil layers of subp",
                          bd_incons$Subplot_no, "in",
                          bd_incons$location_zone, "zone", sep=" "),
                    ifelse((bd_incons$location_zone== "Sundarbans" |
                              bd_incons$location_zone=="Coastal") &
                             bd_incons$depth_count < 3,
                           paste("BD data in 2 soil layers under subp",
                                 bd_incons$Subplot_no, "in",
                                 bd_incons$location_zone, "zone", sep=" "),
                           ifelse(bd_incons$depth_count>3,
                                  paste("BD data in only in 4 soil layers of subp",
                                        bd_incons$Subplot_no, "in",
                                        bd_incons$location_zone, "zone", sep=" "),
                                  paste("BD data in only 1 soil layer under subp",
                                        bd_incons$Subplot_no, "in", bd_incons$location_zone,
                                        "zone", sep=" "))))
  bd_incons <- merge(bd_incons, team_name, by.x="plot_id", by.y = "plot_plot_id", all.x=T)
  names(bd_incons)[1] <- "plot_id"
  output8.5 <- bd_incons[,c("cp", "cat", "plot_id", "issue", "team")]}


# Export output
```

```
write.csv(output8.5, file = "./QAQC_outputs/8.5_Inconsistency in Bulk density layer number and zone.csv",
          row.names = F)


#-------------------------------------------------------------------------------------#
# CP 8.6: Whether the number of bulk density data reported from 3 subplots ------------
#         [input: bulk density data;
#         output: number of completely sampled plots with < 3 subplots from which
#         soil bulk density was reported and list of those plots]
#-------------------------------------------------------------------------------------#
# select the Key variables
bd_subp <- soil_bulk[,c("plot_id", "subplot_id")]
bd_subp<- unique(bd_subp)
# Count the number of subplots per plots
bd_subp$subplot_count <- 1
bd_subp <- aggregate(subplot_count~plot_id, data=bd_subp, sum)

#merge the plot status
bd_subp<- merge(bd_subp, plot_info[, c("plot_id", "plot_details_plot_status_label")],
                by.x ="plot_id", by.y ="plot_id", all.x=T)
# select the completely sampled plots in which soil sample was collected
# from less than two subplots
```

```
bd_subpT <- bd_subp[which(bd_subp$subplot_count <3 & bd_subp$plot_details_plot_status_label==
                    "Accessible - Sampled completely"),]


# Prepare the output table
if(dim(bd_subpT)[1]==0){
  output8.6 <- data.frame(cp= "CP 8.6", cat="bulk density sample collected subp_no",
                    plot_id = NA, issue= "Passed successfully", team=NA)
}else{
  bd_subpT$cp <- "CP 8.6"
  bd_subpT$cat <- "Consistency betwn BD and soil layer"
  bd_subpT$issue <- "plot sampled completely but BD collected from <3 subplots"
  bd_subpT <- merge(bd_subpT, team_name, by.x="plot_id", by.y = "plot_plot_id", all.x=T)
  names(bd_subpT)[1] <- "plot_id"
  output8.6 <- bd_subpT[,c("cp", "cat", "plot_id", "issue", "team")]
}


# Export output
write.csv(output8.6, file = "./QAQC_outputs/8.6_plot sampled but Bulk Ddensity in less than 3 subplots.csv",
        row.names = F)
```

```
#-------------------------------------------------------------------------------#
# CP 8.7: Mismatching lf_id in bulk density and lf database -------------------------
#        [input: bulk density data; lf.csv;
#        output: number of plots with mismatching LF ID
#-------------------------------------------------------------------------------#
# Select the key variables
bdKV <- unique(soil_bulk[,c("plot_id", "subplot_id", "lf_id")])
# prepare a seperate variable for plot and lf in "bdKV"
bdKV$plot_lf_bd <- paste(bdKV$plot_id, bdKV$lf_id, sep = "_")
# prepare a seperate variable for plot and lf in "lf"
lf$plot_lf_lf <- paste(lf$plot_plot_id, lf$lf_id, sep="_")
# plot_lf in bdKV database not present LF database
lf_mismatch <- bdKV[!bdKV$plot_lf_bd %in% lf$plot_lf_lf,]
# add the plot_id that are entered in LF database
lf$count <- 1
lf_mismatch <- merge(lf_mismatch, lf[,c("plot_plot_id", "count")], by.x="plot_id",
                     by.y="plot_plot_id", all.x=T)

# prepare the outputs
if(dim(lf_mismatch)[1]==0){
  output8.7 <- data.frame(cp= "CP 8.7", cat="lf_id of bulk density mismatching with lf database",
```

```
                    plot_id = NA, issue= "No lf_id mismatch found", team=NA)
}else{
  lf_mismatch$issue <- ifelse(is.na(lf_mismatch$count), "plot is not found in LF database",
                              "plot present in LF database but LF ID mismatched")
  lf_mismatch$cat <- "lf_id of bulk density mismatching with lf database"
  lf_mismatch$cp <- "CP 87"
  lf_mismatch <- merge(lf_mismatch, team_name, by.x="plot_id", by.y = "plot_plot_id", all.x=T)
  names(lf_mismatch)[1] <- "plot_id"
  output8.7 <- lf_mismatch[,c("cp", "cat", "plot_id", "issue", "team")]}

# Export output
write.csv(output8.7, file = "./QAQC_outputs/8.7_lf_id mismatching in bulk density and lf database.csv",
          row.names = F)

#-------------------------------------------------------------------------------------#
# CP 8.8: BD data said collected by FT but data not present in databae ----------------
#         [input: BD data; subplot.csv;
#         Output: number of plots, subplots from which soil BD collected
#         mismatching with BD data]
#-------------------------------------------------------------------------------------#
# Prepare subplot data
```

```
subp_bdKV <- subplot_info[,c("plot_plot_id", "subp_id", "soil_soil_bd.1.",
                             "soil_soil_bd.2.", "soil_soil_bd.3.")]
# Remove NAs
subp_bdKV <- subp_bdKV[!is.na(subp_bdKV$soil_soil_bd.1.),]

#add BD data from data received from KU: for depth class 1
subp_bdKVC1 <- merge(subp_bdKV, soil_bulk[soil_bulk$soil_depth_cm==
                                    "5 to 10",][,c("plot_id", "subplot_id",
                                     "bulk_density_g_cc")],
                    by.x=c("plot_plot_id", "subp_id"),
                    by.y=c("plot_id", "subplot_id"), all=T)

#add BD data from data received from KU: for depth class 1
subp_bdKVC2 <- merge(subp_bdKVC1, soil_bulk[soil_bulk$soil_depth_cm==
                                    "20 to 25",][,c("plot_id", "subplot_id",
                                                "bulk_density_g_cc")],
                    by.x=c("plot_plot_id", "subp_id"),
                    by.y=c("plot_id", "subplot_id"), all=T)
#add BD data from data received from KU: for depth class 1
subp_bdKVC3 <- merge(subp_bdKVC2, soil_bulk[soil_bulk$soil_depth_cm==
                                     "65 to 70",][,c("plot_id", "subplot_id",
```

```
                                    "bulk_density_g_cc")],
                by.x=c("plot_plot_id", "subp_id"),
                by.y=c("plot_id", "subplot_id"), all=T)

names(subp_bdKVC3)[6:8] <- c("bulk_density_depth_class_1", "bulk_density_depth_class_2", "bulk_density_depth_class_3")
# remove 0 in soil_soil_bd.1. column
subp_bdKVC3 <- subp_bdKVC3[(subp_bdKVC3$soil_soil_bd.1.>0),]
# remove NA
subp_bdKVC3 <- subp_bdKVC3[!is.na(subp_bdKVC3$plot_plot_id),]
# Select cells field teams collected but we don't have bd data or field teams haven't
# about data collection but we have bd data
bd_prob <- subp_bdKVC3[is.na(subp_bdKVC3$soil_soil_bd.1.)|
                   is.na(subp_bdKVC3$bulk_density_depth_class_1)|
                   (!is.na(subp_bdKVC3$bulk_density_depth_class_3)&
                     is.na(subp_bdKVC3$soil_soil_bd.3.))|
                   (is.na(subp_bdKVC3$bulk_density_depth_class_3)&
                     !is.na(subp_bdKVC3$soil_soil_bd.3.)),]

# prepare the outputs
if(dim(bd_prob)[1]==0){
  output8.8 <- data.frame(cp= "CP 8.8", cat="FT BD sample collection and BD data availability mismatching",
```

```
                plot_id = NA, issue= "No lf_id mismatch found", team=NA)
}else{
  bd_prob$issue <- ifelse(is.na(bd_prob$soil_soil_bd.1.),
                "FT havnt collected BD sample from Depth class 1 or 2 but we have BD data",
                ifelse(is.na(bd_prob$bulk_density_depth_class_1),
                    "FT collected BD sample from Depth class 1 or 2 but we dont have BD data",
                    ifelse(is.na(bd_prob$bulk_density_depth_class_3),
                        "FT collected BD sample from Depth class 3 but we dont have BD data",
                        "FT havnt collected BD sample from Depth class 3 but we have BD data")))
  bd_prob$cat <- "FT BD sample collection and BD data availability mismatching"
  bd_prob$cp <- "CP 88"
  bd_prob <- merge(bd_prob, team_name, by.x="plot_plot_id", by.y = "plot_plot_id", all.x=T)
  names(bd_prob)[1] <- "plot_id"
  output8.8 <- bd_prob[,c("cp", "cat", "plot_id", "issue", "team")]}

# Export output
write.csv(output8.8, file = "./QAQC_outputs/8.8_FTs BD sample collection and BD data availability mismatching.csv",
        row.names = F)



#-------------------------------------------------------------------------------#
```

```
# CP  9 : Soil organic carbon related checks ----------------------------------------
#-------------------------------------------------------------------------------#

#-------------------------------------------------------------------------------#
# CP 9.1: NAs in soil organic carbon data ---------------------------------------
#        [input: soil organic carbon data;
#        Output: list of plots, subplot with NAs in soil organic carbon table]
#-------------------------------------------------------------------------------#
# Prepare the database with key variables
scKV <- soil_car[,c("plot_id", "soil_depth_cm", "lf_id", "organic_carbon_perc")]
# Identify the NAs in soil organic data and prepare the output table
if(dim(scKV[(is.na(scKV$plot_id) | is.na(scKV$soil_depth_cm) |
         is.na(scKV$lf_id) | is.na(scKV$organic_carbon_perc)),])[1]==0){
  output9.1 <- data.frame(cp= "CP 9.1", cat="NA in soil carbon data",
                          plot_id = NA, issue= "No NAs found", team=NA)
}else{
  sc_na <- scKV[(is.na(scKV$plot_id) | is.na(scKV$soil_depth_cm) |
                 is.na(scKV$lf_id) | is.na(scKV$organic_carbon_perc)),]
  sc_na$cp <- "CP 9.1"
  sc_na$cat <- "NA in soil carbon data"
  sc_na$issue <- paste(ifelse(is.na(scKV$plot_id), "NA in plot_id",
```

```
                    ifelse(is.na(scKV$soil_depth_cm), "NA in soil layer",
                        ifelse(is.na(scKV$lf_id),
                            "NA in land feature",
                            "NA in soil organic carbon"))))
  sc_na <- merge(sc_na, team_name, by.x="plot_id", by.y="plot_plot_id", all.x=T)
  output9.1 <- sc_na[,c("cp", "cat", "plot_id", "issue", "team")]}

# Export output
write.csv(output9.1, file = "./QAQC_outputs/9.1_NA in soil organic carbon data.csv",
          row.names = F)

#-------------------------------------------------------------------------------#
# CP 9.2: Identify the blanks in SOC data ----------------------------------------
#         [input: soil organic carbon data;
#         output: list of plots, subplot with blank cells in SOC data frame]
#-------------------------------------------------------------------------------#
# Prepare the database with key variables
scKV <- soil_car[,c("plot_id", "soil_depth_cm", "lf_id", "organic_carbon_perc")]
# Identify the blank data and prepare the output table
if(dim(scKV[(scKV$plot_id=="" | scKV$soil_depth_cm=="" |
          scKV$lf_id=="" | scKV$organic_carbon_perc==""),])[1]==0){
```

```
  output9.2 <- data.frame(cp= "CP 9.2", cat="Blank cells SOC data table",
                          plot_id = NA, issue= "No blank cells found", team=NA)
}else{
  sc_blank <- scKV[(scKV$plot_id=="" | scKV$Soil_layer=="" |
                      scKV$lf_id=="" | scKV$organic_carbon_perc==0),]
  sc_blank$cp <- "CP 9.2"
  sc_blank$cat <- "Blank cells in SOC data frame"
  sc_blank$issue <- paste(ifelse(scKV$plot_id=="", "Blank in plot_no",
                                 ifelse(scKV$Soil_layer=="", "Blank in soil layer",
                                        ifelse(scKV$Land_feature_no=="",
                                               "Blank in lf", "Blank in SOC"))))
  sc_blank <- merge(sc_blank, team_name, by.x="plot_id", by.y="plot_plot_id", all.x=T)
  names(sc_blank)[1]<- "plot_id"
  output9.2 <- sc_blank[,c("cp", "cat", "plot_id", "issue", "team")]}

# Export output
write.csv(output9.2, file = "./QAQC_outputs/9.2_Blanks in soil organic carbon data.csv",
          row.names = F)

#-------------------------------------------------------------------------------#
# CP 9.3: Find the duplicate values in soil organic carbon data ----------------------
```

```
#       [input: soil carbon data;
#       utput: list of plots with duplicate values
#-------------------------------------------------------------------------------#
# Prepare the database with key variables
scKV <- soil_car[,c("plot_id", "soil_depth_cm", "lf_id", "organic_carbon_perc")]
# add count to sum the number of data at different levels
scKV$count <- 1

# CP 9.3.1 find duplicate upto SOC
#-------------------------------------------------------------------------------#
# Find the number of duplicates at plot, soil layer, lf and organic data level
sc_dp <-aggregate(count~plot_id+soil_depth_cm+lf_id+organic_carbon_perc, data = scKV, sum)
sc_dp_1 <- sc_dp[sc_dp$count>1, ]

#add results to output
if(dim(sc_dp_1)[1]==0){
  output9.3.1 <- data.frame(cp= "CP 9.3.1", cat="Duplicate values in SOC", plot_id = NA,
                            issue= "No duplicates found up to SOC data table", team=NA)
}else{
  sc_dp <- sc_dp[-as.integer(rownames(sc_dp_1)),]
  sc_dp_1$cp <- "CP 9.3.1"
```

```
  sc_dp_1$cat <- "Duplicate values in SOC"
  sc_dp_1$issue <- "values duplicated up to SOC"
  sc_dp_1 <- merge(sc_dp_1, team_name, by.x="plot_id", by.y = "plot_plot_id", all.x=T)
  names(sc_dp_1)[1] <- "plot_id"
  output9.3.1 <- sc_dp_1[,c("cp", "cat", "plot_id", "issue", "team")]}


# CP 9.3.2 find duplicate upto lf level
#-------------------------------------------------------------------------------#
sc_dp <-aggregate(count~plot_id+soil_depth_cm+lf_id, data = scKV, sum)
sc_dp_2 <- sc_dp[sc_dp$count>1,]

#add results to output
if(dim(sc_dp_2)[1]==0){
  output9.3.2 <- data.frame(cp= "CP 9.3.2", cat="Duplicate values in SOC",
                            plot_id=NA,issue= "No duplicates found up to lf", team=NA)
}else{
  sc_dp <- sc_dp[-as.integer(rownames(sc_dp_2)),]
  sc_dp_2$cp <- "CP 9.3.2"
  sc_dp_2$cat <- "Duplicate values in SOC"
  sc_dp_2$issue <- "values duplicated up to lf"
```

```
  sc_dp_2 <- merge(sc_dp_2, team_name, by.x="plot_id", by.y = "plot_plot_id", all.x=T)
  names(sc_dp_2)[1] <- "plot_id"
  output9.3.2 <- sc_dp_2[,c("cp", "cat", "plot_id", "issue", "team")]}

# Put all duplicates together
output9.3 <- rbind(output9.3.1, output9.3.2)

# Export output
write.csv(output9.3, file = "./QAQC_outputs/9.3_Duplicates in SOC data.csv", row.names = F)

#-------------------------------------------------------------------------------#
# CP 9.4: Outliers (extreme values) in soil organic carbon/ha -------------------------
#         [Iinput: Soil organic carbon data; bulk density data
#         Output: soil layers and plots in which outliers (strange) data was found]
#-------------------------------------------------------------------------------#
# Note: WHAT LITERATURE SAYS ABOUT OC%
# Banglapedia: Young floodplain ridge soils generally have 1% organic matter in the topsoil,
# so do well-aerated ridge soils on older floodplains. Older floodplain soils,
# except those on the highest ridges, generally have 1-2% organic matter in the
# surface layer, and topsoils in deep basin centres generally have 2-5%.
# Black Terai soils commonly have 3-6% organic matter in their thick topsoils.
```

```
# Available reports indicate that about 70% of the net cultivable area in high and
# medium-high lands have a soil organic matter content of less than 2%.

####### Preparation of data
# Select the key variables from bulk density tables
bdKV <- soil_bulk[,c("plot_id", "subplot_id", "lf_id", "soil_depth_cm", "bulk_density_g_cc")]
# create a column in bulk density database for soil depth class
bdKV$depth_class <- ifelse(bdKV$soil_depth_cm=="5 to 10", 1,
                           ifelse(bdKV$soil_depth_cm=="20 to 25", 2, 3))

# Selct the key variables from SOC table
scKV <- soil_car[,c("plot_id", "lf_id", "soil_depth_cm", "organic_carbon_perc")]
#In the soil_car table create a column with soil layer for soil carbon
scKV$depth_class <- ifelse(scKV$soil_depth_cm=="0 to 15", 1,
                           ifelse(scKV$soil_depth_cm=="15 to 30", 2, 3))
# compute mean SOC per plot, lf and depth class
soc <- aggregate(organic_carbon_perc~plot_id+lf_id+depth_class,
                 data=scKV, FUN=mean) #soc=soil  organic carbon by plot, lf and depth class
#merge bulk density with soil carbon table
socBd <- merge(bdKV, soc, by.x=c("plot_id", "lf_id", "depth_class"),
               by.y=c("plot_id", "lf_id", "depth_class"), all.x=TRUE)
```

```
#remove the zero values and NAs in organic carbon
socBdF <- socBd[!is.na(socBd$organic_carbon_perc),]

# Add a column with soil depth interval (cm)
socBdF$depth_cm <- ifelse(socBdF$depth_class==1, 15, ifelse(socBdF$depth_class==2, 15, 70)) #depth_cm = soil depth interval
# Add locations zone
socBdFZ <- merge(socBdF, plot_info[,c("plot_id", "location_zone")],by="plot_id", all.x=T)
# remove unnecessary soil layer data (30-100cm) from village zone
socBdFZ <- socBdFZ[!(socBdFZ$location_zone=="Villages"& socBdFZ$depth_class==3),]

#calculate soil carbon t/ha
# Process: soil organic carbon (t/ha) = Bulk density (g/cm3) ɔOC (%) ɔSoil depth interval (cm) ɔ100
# Here, soil depth interval is 15 cm for 0-15 cm depth interval, 15 cm for
# soil depth 15-30 cm depth interval, and 70 cm for 30-100 cm interval. %OC
# is expressed as a decimal fraction (e.g., 5% is expressed as 0.05) and
# 100 is a conversion factor to convert the units to t/ha.
# The calculation method followed the protocol of Donato et al. (2009) .

socBdFZ$soc_t_ha <- socBdFZ$bulk_density_g_cc*(socBdFZ$organic_carbon_perc/100)*socBdFZ$depth_cm*100
```

```
######## Outlier identificaiton for soil depth 0-30 cm
# Select the key variables
socKV <- socBdFZ[,c("plot_id","subplot_id", "lf_id","depth_class","soc_t_ha")]
#select soil depth upto 30 cm
socKV <- socKV[socKV$depth_class==1|socKV$depth_class==2,]
#add nlcl
socKV <- merge(socKV, lf[,c("plot_plot_id", "lf_id", "nlcl")], by.x=c("plot_id","lf_id"),
          by.y=c("plot_plot_id", "lf_id"), all.x=T)

### Identification of Outliers
# A table of statistics (mean and sd) is done
f94 <- function(x) c(n = length(x), Mean = mean(x), SD = sd(x))
BasicSat <- aggregate(soc_t_ha ~ nlcl, data=socKV[!is.na(ifelse(socKV$soc_t_ha==Inf,NA,
                                        socKV$soc_t_ha)),], FUN=f94)
BasicSat$n<-BasicSat$soc_t_ha[,1]
BasicSat$Mean<-BasicSat$soc_t_ha[,2]
BasicSat$SD<-BasicSat$soc_t_ha[,3]

# SOC database and Statistical database are merged by nlcl
socKVSt<-merge(socKV, BasicSat[,c("nlcl", "n", "Mean", "SD")],by.x="nlcl", by.y="nlcl",all.x=T)
socKVStF <- socKVSt[socKVSt$n>3,]
```

```
# The Z value is estimated and Possible Outliers are identified
socKVStF$Z<-((socKVStF$soc_t_ha-socKVStF$Mean)/socKVStF$SD)
socKVStF$PO<-ifelse(socKVStF$Z>4,1,0)
dim(socKVStF)

# Identification of Outliers
socKVStFf<-socKVStF[!is.na(socKVStF$PO) & socKVStF$PO==1,]

### Possible Outliers are summarized and saved
# A Database of possible OL is created
socKVStFf$issue <- paste("LCC:", socKVStFf$nlcl,
                         "C_tHa:", round(socKVStFf$soc_t_ha,1),
                         "Mean:",round(socKVStFf$Mean,1),
                         "Z:",round(socKVStFf$Z,1),sep = " ")
socKVStFf$cp <- "CP 7.15"
socKVStFf$cat <- "Possible outlier in SOC per hectare"
socKVStFf <- merge(socKVStFf, team_name, by.x = "plot_id", by.y="plot_plot_id", all.x=T)
names(socKVStFf)[1] <- "plot_id"

# Prepare the output table
if(dim(socKVStFf)[1] == 0){
```

```
  output9.4 <- data.frame(cp="CP 7.15",cat="Outlier in SOC per ha",plot_id=NA,
                          issue="No outlier in SOC per ha", team = NA,
                          stringsAsFactors = F)
  } else {
  output9.4 <- socKVStFf[,c("cp","cat","plot_id","issue", "team")]}


# export the outputs
write.csv(output9.4, file = "./QAQC_outputs/9.4_Possible outliners in SOC per ha.csv",
          row.names = F)


#-------------------------------------------------------------------------------#
# CP 9.5: Mismatching lf_id in bulk density and soil carbon ---------------------
#         [input: Soil BD and organic carbon data,
#         output: list of plots with only organic carbon data but no BD data,
#         plots with nonmatching land feature number]
#-------------------------------------------------------------------------------#
### Prepare the BD table
# Select the key variables in bulk density database
bdKV <- soil_bulk[,c("plot_id", "soil_depth_cm", "lf_id")]
# assign a class for each depth
bdKV$depth_class <- ifelse(bdKV$soil_depth_cm=="5 to 10", 1,
```

```
                    ifelse(bdKV$soil_depth_cm=="20 to 25", 2,
                        ifelse(bdKV$soil_depth_cm=="65 to 70",3, NA)))
# prepare a column with plot, lf and depth
bdKV$bd_plot_lf_depth <- paste(bdKV$plot_id, bdKV$lf_id, bdKV$depth_class, sep = "_")

### Prepare the soil organic carbon table
# Select the Key variables
scKV <- soil_car[,c("plot_id", "lf_id", "soil_depth_cm" )]
# assign a class for each depth
unique(scKV$soil_depth_cm)
scKV$depth_class <- ifelse(scKV$soil_depth_cm=="0 to 15", 1,
                    ifelse(scKV$soil_depth_cm=="15 to 30",2,
                        ifelse(scKV$soil_depth_cm=="30 to 100", 3, NA)))
# prepare a column with plot, lf and lf
scKV$sc_plot_lf_depth <- paste(scKV$plot_id, scKV$lf_id, scKV$depth_class, sep = "_")

##### Check the mismatching plot_lf_depth
### plot_lf_depth present in SOC table but not in BD table
missing_in_soc <- scKV[!scKV$sc_plot_lf_depth %in% bdKV$bd_plot_lf_depth,]
dim(missing_in_soc)
# add the plot_id that are entered in BD database
```

```
bdKV$count <- 1
missing_in_soc <- merge(missing_in_soc, bdKV[,c("plot_id", "count")], by.x="plot_id", by.y="plot_id", all.x=T)

# Prepare the output table for PLOT_LF_DEPTH present in SOC table but absent in BD table
if(dim(missing_in_soc)[1]==0){
  output9.5.1 <- data.frame(cp= "CP 9.5.1", cat="PLOT_LF_DEPTH mismatching in SOC and BD data",
                            plot_id = NA, issue= "No mismatch found", team=NA)
}else{
  missing_in_soc$issue <- ifelse(is.na(missing_in_soc$count), "plot present in SOC but not in BD database",
                                 "plot present in SOC but -PLOT_LF_DEPTH- mismatched with BD")
  missing_in_soc$cat <- "PLOT_LF_DEPTH mismatching in SOC and BD data"
  missing_in_soc$cp <- "CP 9.5.1"
  missing_in_soc <- merge(missing_in_soc, team_name, by.x="plot_id", by.y = "plot_plot_id", all.x=T)
  names(missing_in_soc)[1] <- "plot_id"
  output9.5.1 <- missing_in_soc[,c("cp", "cat", "plot_id", "issue", "team")]}

### plot_lf_depth present in SOC table but not in BD table
missing_in_BD <- bdKV[!bdKV$bd_plot_lf_depth %in% scKV$sc_plot_lf_depth,]
dim(missing_in_BD)
# add the plot_id that are entered in SOC database
scKV$sc_count <- 1
```

```
missing_in_BD <- merge(missing_in_BD, scKV[,c("plot_id", "sc_count")], by="plot_id", all.x=T)

# Prepare the output table for PLOT_LF_DEPTH present in SOC table but absent in BD table
if(dim(missing_in_BD)[1]==0){
  output9.5.2 <- data.frame(cp= "CP 9.5.2", cat="PLOT_LF_DEPTH mismatching in SOC and BD", plot_id = 9999,
                            issue= "No mismatch found", team=99)
}else{
  missing_in_BD$issue <- ifelse(is.na(missing_in_BD$count), "plot present in SOC but not in SOC database",
                                "plot present in BD but -PLOT_LF_DEPTH- mismatched with SOC")
  missing_in_BD$cat <- "PLOT_LF_DEPTH mismatching in SOC and BD"
  missing_in_BD$cp <- "CP 9.5.2"
  missing_in_BD <- merge(missing_in_BD, team_name, by.x="plot_id", by.y = "plot_plot_id", all.x=T)
  names(missing_in_BD)[1] <- "plot_id"
  output9.5.2 <- missing_in_BD[,c("cp", "cat", "plot_id", "issue", "team")]}

# Put all mismatchs together
output9.5 <- rbind(output9.5.1, output9.5.2)

# export the outputs
write.csv(output9.5, file = "./QAQC_outputs/9.4_Possible outliners in SOC per ha.csv",
          row.names = F)
```

```
#-------------------------------------------------------------------------------#
# CP 9.6: Relationship of bulk density and soil carbon ---------------------------------
#       [Note: To evaluate visually if there is any abnormal relationship]
#       [input: Soil organic carbon data;
#       Output: graphs indicating the relationship between SPC and bulk density]
#-------------------------------------------------------------------------------#
# Select the key variables from Bulk Density
bdKV <- soil_bulk[,c("plot_id", "subplot_id", "lf_id", "soil_depth_cm", "bulk_density_g_cc")]
# Assign a depth for each soil depth
bdKV$depth_class <- ifelse(bdKV$soil_depth_cm=="5 to 10", 1,
                    ifelse(bdKV$soil_depth_cm=="20 to 25", 2,
                        ifelse(bdKV$soil_depth_cm=="65 to 70",3, NA)))
# Select key variables from SOC
scKV <- soil_car[,c("plot_id", "lf_id", "soil_depth_cm", "organic_carbon_perc")]
# Assign a depth for each soil depth
scKV$depth_class <- ifelse(scKV$soil_depth_cm=="0 to 15", 1,
                    ifelse(scKV$soil_depth_cm=="15 to 30",2,
                        ifelse(scKV$soil_depth_cm=="30 to 100", 3, NA)))
# merge BD and SOC data
```

```
sc_bd <- merge(bdKV[,c("plot_id", "subplot_id", "lf_id", "depth_class", "bulk_density_g_cc")],
               scKV[,c("plot_id", "lf_id", "depth_class", "organic_carbon_perc")],
               by=c("plot_id", "lf_id", "depth_class"), all.x=T)
#merge forest zone with soil car data table
sc_bd <- merge(sc_bd, plot_info[,c("plot_id", "location_zone")], by="plot_id", all.x=TRUE)
#merge team number
sc_bd<- merge(sc_bd, team_name, by.x="plot_id", by.y="plot_plot_id", all.x = TRUE)


#plot organic carbon perc Vs bulk density
ggplot(sc_bd)+
  aes(y=bulk_density_g_cc, x=organic_carbon_perc)+
  geom_point()+
  ylab("Bulk density g/cc")+xlab("Soil organic carbon (%)")+
  #  facet_wrap(~depth_class)+
  #  geom_smooth(method=lm)+
  ggtitle("Bulk density Vs Soil organic carbon")
#  facet_wrap(~team)


#plot organic carbon perc Vs bulk density by different layers
ggplot(sc_bd[which(sc_bd$bulk_density_g_cc <=2 & sc_bd$organic_carbon_perc <= 5),])+
  aes(y=bulk_density_g_cc, x=organic_carbon_perc)+
```

```
  geom_point()+
  ylab("Bulk density g/cc")+xlab("Soil organic carbon (%)")+
  #  facet_wrap(~depth_class)+
  geom_smooth(method=lm)+
  ggtitle("Bulk density and organic carbon when apparent outliers removed
        (BD </=2 g/cc & SoC </=5 perc removed)")
#  facet_wrap(~team)


#-------------------------------------------------------------------------------#
# CP 9.7: Mismatching lf_id of SOC  and LF database ----------------------------------
#         [input: SOC data; lf.csv;
#         Output: number of plots with mismatching LF ID]
#-------------------------------------------------------------------------------#
# Select the key variabes
scKV <- soil_car[,c("plot_id", "lf_id", "soil_depth_cm","organic_carbon_perc")]
# prepare a seperate variable for plot and lf in "bdKV"
scKV$plot_lf_sc <- paste(scKV$plot_id, scKV$lf_id, sep = "_")
# prepare a seperate variable for plot and lf in "lf" database
lf$plot_lf_lf <- paste(lf$plot_plot_id, lf$lf_id, sep="_")
# plot_lf in scKV database not present LF database
lf_mismatch <- scKV[!scKV$plot_lf_sc %in% lf$plot_lf_lf,]
```

```
# add the plot_id that are entered in LF database
lf$count <- 1
lf_mismatch <- merge(lf_mismatch, lf[,c("plot_plot_id", "count")], by.x="plot_id", by.y="plot_plot_id", all.x=T)

# prepare the outputs
if(dim(lf_mismatch)[1]==0){
  output9.10 <- data.frame(cp= "CP 9.10", cat="LF id mismatching between SOC and LF database",
                           plot_id = NA, issue= "No Mismatch found", team=NA)
}else{
  lf_mismatch$issue <- ifelse(is.na(lf_mismatch$count), "plot not present in LF database",
                              "plot present in SOC and LF database but LF ID mismatched")
  lf_mismatch$cat <- "LF id mismatching between SOC and LF database"
  lf_mismatch$cp <- "CP 9.10"
  lf_mismatch <- merge(lf_mismatch, team_name, by.x="plot_id", by.y = "plot_plot_id", all.x=T)
  names(lf_mismatch)[1] <- "plot_id"
  output9.7 <- lf_mismatch[,c("cp", "cat", "plot_id", "issue", "team")]}

# export the outputs
write.csv(output9.7, file = "./QAQC_outputs/9.7_SOC lf_id not matching with LF database.csv",
          row.names = F)
```

```
#-------------------------------------------------------------------------------#
# CP 9.8: Texture data said collected by FT but SOC data not present in databae ----------
#         [input: Texture data; subplot.csv;
#         Output: number of plots, subplots from which soil Texture collected
#         but mismatching with SOC data]
#-------------------------------------------------------------------------------#
# Prepare subplot data
subp_TexKV <- subplot_info[,c("plot_plot_id", "soil_soil_text.1.",
                        "soil_soil_text.2.", "soil_soil_text.3.")]
# Remove NAs
subp_TexKV <- subp_TexKV[!is.na(subp_TexKV$soil_soil_text.1.),]

#add SOC data from data received from KU: for depth class 1
subp_TexKVC1 <- merge(subp_TexKV, soil_car[soil_car$soil_depth_cm==
                                    "0 to 15",][,c("plot_id",
                                                "organic_carbon_perc")],
                by.x="plot_plot_id",
                by.y="plot_id", all=T)

#add BD data from data received from KU: for depth class 1
```

```
subp_TexKVC2 <- merge(subp_TexKVC1, soil_car[soil_car$soil_depth_cm==
                                "15 to 30",][,c("plot_id",
                                        "organic_carbon_perc")],
                by.x="plot_plot_id",
                by.y="plot_id", all=T)
#add SOC data from data received from KU: for depth class 1
subp_TexKVC3 <- merge(subp_TexKVC2, soil_car[soil_car$soil_depth_cm==
                                "30 to 100",][,c("plot_id",
                                        "organic_carbon_perc")],
                by.x="plot_plot_id",
                by.y="plot_id", all=T)


names(subp_TexKVC3)[5:7] <- c("SOC_depth_class_1", "SOC_depth_class_2", "SOC_depth_class_3")
# remove 0 in soil_soil_text.1. column
subp_TexKVC3 <- subp_TexKVC3[(subp_TexKVC3$soil_soil_text.1.>0),]
# remove NA
subp_TexKVC3 <- subp_TexKVC3[!is.na(subp_TexKVC3$plot_plot_id),]
# Select cells field teams collected but we don't have bd data or field teams haven't
# about data collection but we have bd data
soc_prob <- subp_TexKVC3[is.na(subp_TexKVC3$soil_soil_text.1.)|
                    is.na(subp_TexKVC3$SOC_depth_class_1)|
```

```
                    (!is.na(subp_TexKVC3$SOC_depth_class_3)&
                      is.na(subp_TexKVC3$soil_soil_text.3.))|
                    (is.na(subp_TexKVC3$SOC_depth_class_3)&
                      !is.na(subp_TexKVC3$soil_soil_text.3.)),]


# prepare the outputs
if(dim(soc_prob)[1]==0){
  output9.8 <- data.frame(cp= "CP 9.8", cat="FT SOC sample collection and SOC data mismatching",
                          plot_id = NA, issue= "No SOC mismatch found", team=NA)
}else{
  soc_prob$issue <- ifelse(is.na(soc_prob$soil_soil_text.1.),
                           "FT havnt collected SOC sample from Depth class 1 or 2 but we have SOC data",
                           ifelse(is.na(soc_prob$SOC_depth_class_1),
                                  "FT collected SOC sample from Depth class 1 or 2 but we dont have SOC data",
                                  ifelse(is.na(soc_prob$SOC_depth_class_3),
                                         "FT collected SOC sample from Depth class 3 but we dont have SOC data",
                                         "FT havnt collected SOC sample from Depth class 3 but we have SOC data")))
  soc_prob$cat <- "FT SOC sample collection and SOC data mismatching"
  soc_prob$cp <- "CP 98"
  soc_prob <- merge(soc_prob, team_name, by.x="plot_plot_id", by.y = "plot_plot_id", all.x=T)
  names(soc_prob)[1] <- "plot_id"
```

```
  output9.8 <- soc_prob[,c("cp", "cat", "plot_id", "issue", "team")]}


# Export output
write.csv(output9.8, file = "./QAQC_outputs/9.8_FTs SOC sample collection and SOC data availability mismatching.csv",
          row.names = F)



#-------------------------------------------------------------------------------#
# CP  10 : Soil texture ----------------------------------------------------------
#-------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------#
# CP  10.1: NAs in soil texture --------------------------------------------------
#         [input: soil texture data;
#         Output: list of plots, subplot with NAs in cells for soil texture]
#-------------------------------------------------------------------------------#
# Prepare the database with key variables
tx <- soil_tex[,c("plot_id", "lf_id", "soil_depth_cm", "sand_perc","clay_perc",
                  "silt_perc","texture" )]
# find the NAs and prepare the outptut table
if(dim(tx[(is.na(tx$plot_id) | is.na(tx$soil_depth_cm) | is.na(tx$lf_id) |
```

```
        is.na(tx$sand_perc) | is.na(tx$clay_perc) | is.na(tx$silt_perc)|
        is.na(tx$texture)),])[1]==0){
  output10.1 <- data.frame(cp= "CP 10.1", cat="NA in texture data", plot_id = NA,
                           issue= "No NAs found", team=NA)
}else{
  tx <- tx[(is.na(tx$plot_id) | is.na(tx$soil_depth_cm) |
            is.na(tx$lf_id) | is.na(tx$sand_perc) |
            is.na(tx$clay_perc) | is.na(tx$silt_perc)|
            is.na(tx$texture)),]
  tx$cp <- "CP 10.1"
  tx$cat <- "NA in texture data"
  tx$issue <- paste(ifelse(is.na(tx$plot_id), "NA in plot_id",
                           ifelse(is.na(tx$soil_depth_cm), "NA in soil layer",
                                  ifelse(is.na(tx$lf_id), "NA in lf",
                                         ifelse(is.na(tx$sand_perc), "NA in sand perc",
                                                ifelse(is.na(tx$clay_perc), "NA in clay perc",
                                                       ifelse(is.na(tx$silt_perc), "NA in silt perc",
                                                              "NA in texture")))))))
  tx$team <- merge(tx, team_name, by.x="plot_id", by.y="plot_plot_id", all.x=T)
  names(tx)[1]<- "plot_id"
  output10.1 <- tx[,c("cp", "cat", "plot_id", "issue", "team")]}
```

```
# export the outputs
write.csv(output10.1, file = "./QAQC_outputs/10.1_NA in texture data.csv", row.names = F)


#-------------------------------------------------------------------------------#
# CP 10.2: Blank in texture data  ----------------------------------------------
# Input: soil texture data;
# Output: list of plots, with blank cells for soil texture]
#-------------------------------------------------------------------------------#
# Prepare the database with key variables
tx <- soil_tex[,c("plot_id", "lf_id", "soil_depth_cm", "sand_perc","clay_perc",
                  "silt_perc","texture" )]
# Identify the blank cells and put then in the output table
if(dim(tx[(tx$plot_id=="" | tx$soil_depth_cm=="" |
           tx$lf_id=="" | tx$sand_perc=="" |
           tx$sand_perc==""| tx$clay_perc==""|
           tx$silt_perc==""| tx$texture==""),])[1]==0){
  output10.2 <- data.frame(cp= "CP 10.2", cat="Blank cells in texture data", plot_id = NA,
                           issue= "No blank cells found", team=NA)
}else{
  tx <- tx[(tx$plot_id=="" | tx$soil_depth_cm=="" | tx$lf_id=="" | tx$sand_perc=="" |
```

```
          tx$sand_perc==""| tx$clay_perc==""| tx$silt_perc==""| tx$texture==""),]
 tx$cp <- "CP 10.2"
 tx$cat <- "Blank cells in texture data"
 tx$issue <- paste(ifelse(tx$plot_id=="", "Blank plot_no",
                    ifelse(tx$layer=="", "Blank layer",
                        ifelse(tx$lf_id=="", "Blank lf",
                            ifelse(tx$sand_perc=="", "Blank sand_perc",
                                ifelse(tx$clay_perc=="", "Blank clay_perc",
                                    ifelse(tx$silt_perc=="", "Blank silt_perc",
                                        "Blank in texture")))))))
 tx <- merge(tx, team_name, by.x="plot_id", by.y="plot_plot_id", all.x=T)
 names(tx)[1]<- "plot_id"
 output10.2 <- tx[,c("cp", "cat", "plot_id", "issue", "team")]}


# export the outputs
write.csv(output10.2, file = "./QAQC_outputs/10.2_Blanks in texture data.csv", row.names = F)


#-------------------------------------------------------------------------------#
# CP 10.3: Find the duplicate values in soil texture data  ----------------------------
#         [input: soil texture data;
#         output: list of plots with duplicate values.
```

```
#-----------------------------------------------------------------------------------------#
# Prepare the database with key variables
tx <- soil_tex[,c("plot_id", "lf_id", "soil_depth_cm", "sand_perc","clay_perc",
                  "silt_perc","texture" )]
tx$count <- 1

# CP 10.3.1 find duplicate upto texture
#-----------------------------------------------------------------------------------------#
tx_dp <-aggregate(count~plot_id+soil_depth_cm+lf_id+sand_perc+clay_perc+silt_perc+texture,
                  data = tx, sum)
tx_dp_1 <- tx_dp[tx_dp$count>1, ]

#add results to output
if(dim(tx_dp_1)[1]==0){
  output10.3.1 <- data.frame(cp= "CP 10.3.1", cat="Duplicate values in texture data",
                             plot_id = NA,issue= "No duplicates in texture data", team=NA)
}else{
  tx_dp <- tx_dp[-as.integer(rownames(tx_dp_1)),]
  tx_dp_1$cp <- "CP 10.3.1"
  tx_dp_1$cat <- "Duplicate values in texture"
  tx_dp_1$issue <- "values duplicated up to texture"
```

```
  tx_dp_1 <- merge(tx_dp_1, team_name, by.x="plot_id", by.y = "plot_plot_id", all.x=T)
  names(tx_dp_1)[1] <- "plot_id"
  output10.3.1 <- tx_dp_1[,c("cp", "cat", "plot_id", "issue", "team")]}



# CP 10.3.2 find duplicate upto silt_perc level
#-------------------------------------------------------------------------------------#
tx_dp <-aggregate(count~plot_id+soil_depth_cm+lf_id+sand_perc+clay_perc+silt_perc,
                  data = tx, sum)
tx_dp_2 <- tx_dp[tx_dp$count>1,]

#add results to output
if(dim(tx_dp_2)[1]==0){
  output10.3.2 <- data.frame(cp= "CP 10.3.2", cat="Duplicates in soil texture data",
                  plot_id = NA, issue= "No duplicates found up to silt_perc", team=NA)
}else{
  tx_dp <- tx_dp[-as.integer(rownames(tx_dp_2)),]
  tx_dp_2$cp <- "CP 10.3.2"
  tx_dp_2$cat <- "Duplicates in soil texture data"
  tx_dp_2$issue <- "values duplicated up to silt perc"
  tx_dp_2 <- merge(tx_dp_2, team_name, by.x="plot_id", by.y = "plot_plot_id", all.x=T)
```

```
  names(tx_dp_2)[1] <- "plot_id"
  output10.3.2 <- tx_dp_2[,c("cp", "cat","plot_id", "issue", "team")]}


# CP 10.3.3 find duplicate upto clay_perc
#-------------------------------------------------------------------------------#
tx_dp <-aggregate(count~plot_id+soil_depth_cm+lf_id+sand_perc+clay_perc, data = tx, sum)
tx_dp_3 <- tx_dp[tx_dp$count>1,]

#add results to output
if(dim(tx_dp_3)[1]==0){
  output10.3.3 <- data.frame(cp= "CP 3.3.3", cat="Duplicates in texture data", plot_id = NA,
                             issue= "No duplicates found up to clay_perc", team= NA)
}else{
  tx_dp <- tx_dp[-as.integer(rownames(tx_dp_3)),]
  tx_dp_3$cp <- "CP 10.3.3"
  tx_dp_3$cat <- "Duplicates in texture data"
  tx_dp_3$issue <- "values duplicated up to clay perc"
  tx_dp_3 <- merge(tx_dp_3, team_name, by.x="plot_id", by.y = "plot_plot_id", all.x=T)
  names(tx_dp_3)[1] <- "plot_id"
  output10.3.3 <- tx_dp_3[,c("cp", "cat", "plot_id", "issue", "team")]}
```

```
# CP 10.3.4 find duplicate upto sand_perc
#-------------------------------------------------------------------------------#
tx_dp <-aggregate(count~plot_id+soil_depth_cm+lf_id+sand_perc, data = tx, sum)
tx_dp_4 <- tx_dp[tx_dp$count>1,]

#add results to output
if(dim(tx_dp_4)[1]==0){
  output10.3.4 <- data.frame(cp= "CP 10.3.4", cat="Duplicates in texture data", plot_id = NA,
                             issue= "No duplicates found up to sand_perc", team=NA)
}else{
  tx_dp <- tx_dp[-as.integer(rownames(tx_dp_4)),]
  tx_dp_4$cp <- "CP 10.3.4"
  tx_dp_4$cat <- "Duplicates in texture data"
  tx_dp_4$issue <- "values duplicated upto sand perc"
  tx_dp_4 <- merge(tx_dp_4, team_name, by.x="plot_id", by.y = "plot_plot_id", all.x=T)
  names(tx_dp_4)[1] <- "plot_id"
  output10.3.4 <- tx_dp_4[,c("cp", "cat", "plot_id", "issue", "team")]}



#CHECK 10.3.5 find duplicate upto lf
```

```
#-----------------------------------------------------------------------------------#
tx_dp <-aggregate(count~plot_id+soil_depth_cm+lf_id, data = tx, sum)
tx_dp_5 <- tx_dp[tx_dp$count>1,]

#add results to output
if(dim(tx_dp_5)[1]==0){
  output10.3.5 <- data.frame(cp= "CP 10.3.5", cat="Duplicate values in texture",
                             plot_id = NA, issue= "No duplicates found up to lf", team=NA)
}else{
  tx_dp <- tx_dp[-as.integer(rownames(tx_dp_5)),]
  tx_dp_5$cp <- "CP 10.3.5"
  tx_dp_5$cat <- "Duplicate values in texture"
  tx_dp_5$issue <- "values duplicated up to lf"
  tx_dp_5 <- merge(tx_dp_5, team_name, by.x="plot_id", by.y = "plot_plot_id", all.x=T)
  names(tx_dp_5)[1] <- "plot_id"
  output10.3.5 <- tx_dp_5[,c("cp", "cat", "plot_id", "issue", "team")]}

# Create one table with all duplicates
output10.3 <- rbind(output10.3.1, output10.3.2, output10.3.3, output10.3.4, output10.3.5)

#export outputs
```

```
write.csv(output10.3, file = "./QAQC_outputs/10.3_Duplicates in texture data.csv", row.names = F)



#-------------------------------------------------------------------------------------#
# CP 10.4: Outliers in percentages of clay, silt and sandy particles ------------------
#         [Note: [i) if the sum of sand, silt and clay percentage
#         is 100 or not; ii) if there are any outliers/negative values.]
# Input: soil texture data;
# Output: List of plots and layers where the percentages of clay, silt and
#        sand particles of same sample is not 100]
#-------------------------------------------------------------------------------------#
# Prepare the database with key variables
tx <- soil_tex[,c("plot_id", "lf_id", "soil_depth_cm", "sand_perc","clay_perc",
                  "silt_perc","texture" )]
# Sum the sand, silt and clay percentage
tx$tex_sum_perc <- round(tx$sand_perc + tx$clay_perc + tx$silt_perc, 0)

#see the order of values in sand, silt and clay
tx[order(tx$tex_sum_perc, decreasing=FALSE),][1:10,]
tx[order(tx$silt_perc, decreasing=FALSE),][1:10,]
tx[order(tx$clay_perc, decreasing=FALSE),][1:10,]
```

```
#checking if the sum of sand, silt and clay is 100
tx[(tx$tex_sum_perc!=100),]

#checking if there is any negetive value in sand, silt and clay
tx[(tx$sand_perc <0 |tx$silt_perc <0 | tx$clay_perc <0),]



#add results to output
if(dim(tx[(tx$tex_sum_perc!=100) &
        (tx$sand_perc <0 |tx$silt_perc <0 | tx$clay_perc <0),])[1]==0){
  output10.4 <- data.frame(cp= "CP 10.6", cat="problems in texture values",
                           plot_id = NA, issue= "No errors found in soil texture values",
                           team=NA)
}else{
  tx_val_err <- tx[(tx$tex_sum_perc!=100) & (tx$sand_perc <0 | tx$silt_perc <0 |
                     tx$clay_perc <0),]
  tx_val_err$cp <- "CP 10.4"
  tx_val_err$cat <- "problems in texture values"
  tx_val_err$issue <- ifelse(tx$tex_sum_perc!=100, "sum of soil particles
                             is not 100", "tex values negative")
```

```
  tx_val_err <- merge(tx_lyr_err, team_name, by.x="plot_id", by.y = "plot_plot_id", all.x=T)
  names(tx_val_err)[1] <- "plot_id"
  output10.4 <- tx_val_err[,c("cp", "cat", "plot_id", "issue", "team")]}

#export outputs
write.csv(output10.4, file = "./QAQC_outputs/10.4_Strange texture values.csv", row.names = F)



#-------------------------------------------------------------------------------#
# CP 10.5: Mismatching lf_id in Soil Texture and LF database ---------------------------
# Input: Soil texture data; lf.csv;
# Output: number of plots with mismatching LF ID]
#-------------------------------------------------------------------------------#
# prepare a seperate database with the key variabes
txKV <- soil_tex[,c("plot_id", "lf_id", "soil_depth_cm")]
# prepare a seperate variable for plot and lf in "txKV"
txKV$plot_lf_tx <- paste(txKV$plot_id, txKV$lf_id, sep = "_")
# prepare a seperate variable for plot and lf in "lf" database
lf$plot_lf_lf <- paste(lf$plot_plot_id, lf$lf_id, sep="_")
# plot_lf in scKV database not present LF database
lf_mismatch <- txKV[!txKV$plot_lf_tx %in% lf$plot_lf_lf,]
```

```
# add the plot_id that are entered in LF database
lf$count <- 1
lf_mismatch <- merge(lf_mismatch, lf[,c("plot_plot_id", "count")], by.x="plot_id", by.y="plot_plot_id", all.x=T)

# prepare the outputs
if(dim(lf_mismatch)[1]==0){
  output10.5 <- data.frame(cp= "CP 10.5", cat="LF id mismatching between Soil Texture and LF database",
                           plot_id = NA, issue= "No Mismatch found", team=NA)
}else{
  lf_mismatch$issue <- ifelse(is.na(lf_mismatch$count), "plot not present in LF database",
                              "plot present in -Soil Texture- and -LF- database but LF ID mismatched")
  lf_mismatch$cat <- "LF id mismatching between Soil Texture and LF database"
  lf_mismatch$cp <- "CP 10.5"
  lf_mismatch <- merge(lf_mismatch, team_name, by.x="plot_id", by.y = "plot_plot_id", all.x=T)
  names(lf_mismatch)[1] <- "plot_id"
  output10.5 <- lf_mismatch[,c("cp", "cat", "plot_id", "issue", "team")]}

#export outputs
write.csv(output10.5, file = "./QAQC_outputs/10.5_Lf ID mismatching in Soil Texture and LF data.csv",
          row.names = F)
```

```
#-------------------------------------------------------------------------------#
# CP 11: Litter carbon data related checks ------------------------------------------
#-------------------------------------------------------------------------------#

#-------------------------------------------------------------------------------#
# CP 11.1: NAs in litter carbon -----------------------------------------------------
# Input: litter organic carbon data;
# Output: list of plots with NAs in cells
#-------------------------------------------------------------------------------#
# Select the key variables
lc <- lit_car[,c("plot_id", "lf_id", "organic_carbon_in_litter_perc")]

# identify the NAs and prepare the output file
if(dim(lc[(is.na(lc$plot_id) | is.na(lc$lf_id) |
          is.na(lc$organic_carbon_in_litter_perc)),])[1]==0){
  output11.1 <- data.frame(cp= "CP 11.1", cat="NA in litter carbon data",
                           plot_id = NA, issue= "No NAs in Litter carbon data", team=NA)
}else{
  lc <- lc[(is.na(lc$plot_id) | is.na(lc$lf_id) | is.na(lc$organic_carbon_in_litter_perc)),]
  lc$cp <- "CP 11.1"
  lc$cat <- "NA in litter carbon data"
```

```
  lc$issue <- paste(ifelse(is.na(lc$plot_id), "NA in plot_id",
                           ifelse(is.na(lc$Land_feature_no), "NA in lf",
                                  "NA in litter carbon data")))
  lc$team <- merge(lc, team_name, by.x="plot_id", by.y="plot_plot_id", all.x=T)
  names(lc)[1]<- "plot_id"
  output11.1 <- lc[,c("cp", "cat", "plot_id", "issue", "team")]}


#export outputs
write.csv(output11.1, file = "./QAQC_outputs/11.1_NA in litter carbon data.csv", row.names = F)


#-------------------------------------------------------------------------------#
# CP 11.2: Blank cells in litter carbon data  ---------------------------------------
# Input: litter carbon data;
# Output: list of plots, with blank cells
#-------------------------------------------------------------------------------#
# Select the key variables
lc <- lit_car[,c("plot_id", "lf_id", "organic_carbon_in_litter_perc")]


# identify the blank cells and prepare the output table
if(dim(lc[(lc$plot_id=="" | lc$lf_id=="" |
          lc$organic_carbon_in_litter_perc==""),])[1]==0){
```

```
  output11.2 <- data.frame(cp= "CP 11.2", cat="Blank cells litter carbon data",
                           plot_id = NA, issue= "No blank cells found", team=NA)
}else{
  lc <- lc[(lc$plot_id=="" | lc$lf_id=="" | lc$organic_carbon_in_litter_perc=="" ),]
  lc$cp <- "CP 11.2"
  lc$cat <- "Blank cells in litter carbon data"
  lc$issue <- paste(ifelse(lc$plot_id=="", "Blank plot_no",
                           ifelse(lc$Land_feature_no=="", "Blank lf",
                                  "Blank litter carbon data")))
  lc <- merge(lc, team_name, by.x="plot_id", by.y="plot_plot_id", all.x=T)
  names(lc)[1]<- "plot_id"
  output11.2 <- lc[,c("cp", "cat", "plot_id", "issue", "team")]}



#export outputs
write.csv(output11.2, file = "./QAQC_outputs/11.2_Blank in litter carbon data.csv", row.names = F)


#-------------------------------------------------------------------------------#
# CP 11.3: Find the duplicate values ----------------------------------------------
# Input: soil texture data;
# Output: list of plots with duplicate values.
```

```
#-------------------------------------------------------------------------------------#
# Select the key variables
lc <- lit_car[,c("plot_id", "lf_id", "organic_carbon_in_litter_perc")]
lc$count <- 1

# CP 11.3.1 find duplicate upto litter carbon
#-------------------------------------------------------------------------------------#
lc_dp <-aggregate(count~plot_id+lf_id+organic_carbon_in_litter_perc, data = lc, sum)
lc_dp_1 <- lc_dp[lc_dp$count>1, ]

#add results to output
if(dim(lc_dp_1)[1]==0){
  output11.3.1 <- data.frame(cp= "CP 11.3.1", cat="Duplicate values in litter carbon",
                plot_id = NA, issue= "No duplicates found up to organic carbon", team=NA)
}else{
  lc_dp <- lc_dp[-as.integer(rownames(lc_dp_1)),]
  lc_dp_1$cp <- "CP 11.3.1"
  lc_dp_1$cat <- "Duplicate values in litter carbon"
  lc_dp_1$issue <- "values duplicated up to organic carbon"
  lc_dp_1 <- merge(lc_dp_1, team_name, by.x="plot_id", by.y = "plot_plot_id", all.x=T)
  names(lc_dp_1)[1] <- "plot_id"
```

```
  output11.3.1 <- lc_dp_1[,c("cp", "cat", "plot_id", "issue", "team")]}



# CP 11.3.2 find duplicate upto land feature
#-------------------------------------------------------------------------------#
lc_dp <-aggregate(count~plot_id+lf_id, data = lc_dp, sum)
lc_dp_2 <- lc_dp[lc_dp$count>1,]

#add results to output
if(dim(lc_dp_2)[1]==0){
  output11.3.2 <- data.frame(cp= "CP 11.3.2", cat="Duplicate values in litter carbon",
                plot_id = NA, issue= "No duplicates found up to land feature", team=NA)
}else{
  lc_dp <- lc_dp[-as.integer(rownames(lc_dp_2)),]
  lc_dp_2$cp <- "CP 11.3.2"
  lc_dp_2$cat <- "Duplicate values in litter carbon"
  lc_dp_2$issue <- "values duplicated up to land feature"
  lc_dp_2 <- merge(lc_dp_2, team_name, by.x="plot_id", by.y = "plot_plot_id", all.x=T)
  names(lc_dp_2)[1] <- "plot_id"
  output11.3.2 <- lc_dp_2[,c("cp", "cat", "plot_id", "issue", "team")]}
```

```
#Put all the outputs together
output11.3 <- rbind(output11.3.1, output11.3.2)


#export outputs
write.csv(output11.3, file = "./QAQC_outputs/11.3_Duplicates in litter carbon data.csv",
          row.names = F)



#-------------------------------------------------------------------------------#
# CP 11.4: Outlier in the litter carbon (%) by LCC ------------------------------------
# Input: Litter carbon data;
# Output: graph (boxplot) presenting outliers in the litter carbon data
#-------------------------------------------------------------------------------#
# Select the key variables
lc <- lit_car[,c("plot_id", "lf_id", "organic_carbon_in_litter_perc")]
# add NLCL from LF data
lc <- merge(lc, lf[,c("plot_plot_id", "lf_id", "nlcl")], by.x = c("plot_id", "lf_id"),
            by.y = c("plot_plot_id", "lf_id"), all.x=T)

### A table of statistics (mean and sd) are done
f114 <- function(x) c(n = length(x), Mean = mean(x), SD = sd(x))
```

```
BasicSat <- aggregate(organic_carbon_in_litter_perc ~ nlcl, data=lc, FUN=f114)
BasicSat$n<-BasicSat$organic_carbon_in_litter_perc[,1]
BasicSat$Mean<-BasicSat$organic_carbon_in_litter_perc[,2]
BasicSat$SD<-BasicSat$organic_carbon_in_litter_perc[,3]

### Tree database and Statistical database are merged by sientific name
lcSt<-merge(lc,BasicSat[,c("nlcl","n", "Mean", "SD")], by="nlcl", all.x=T)
lcStF <- lcSt[lcSt$n>3,]

### The Z value is estimated and Possible Outliers are identified
# Z is estimated
lcStF$Z<-((lcStF$organic_carbon_in_litter_perc-lcStF$Mean)/lcStF$SD)
# Possible Outliers are identify
lcStF$PO<-ifelse(abs(lcStF$Z)>4,1,0)
# remove NAs
lcStF <- lcStF[!is.na(lcStF$plot_id),]

### Possible Outliers are summarized and saved
# A Database of possible OL is created
if(dim(lcStF[lcStF$PO==1,])[1]==0){
  output11.4 <- data.frame(cp= "CP 11.4", cat="Outliers in litter carbon", plot_id = NA,
```

```
                    issue= "No outliers in litter carbon", team=NA)
}else{
  lcStF$issue <- paste("Possible outlier in litter carbon (LCC:", lcStF$nlcl,
                       ", lc%:", lcStF$organic_carbon_in_litter_perc,
                       ", Mean:",round(lcStF$Mean,1), ", Z:",round(lcStF$Z,1),")",sep = "")
  lcStF$cp <- "CP 11.4"
  lcStF$cat <- "Outliers in Litter Carbon"
  lcStF <- merge(lcStF, team_name, by.x="plot_id",  by.y="plot_plot_id", all.x=T)
  names(lcStF)[1] <- "plot_id"
  output11.4 <- lcStF[lcStF$PO==1, c("cp", "cat", "plot_id", "issue", "team")]}

#export outputs
write.csv(output11.4, file = "./QAQC_outputs/11.4_Possible in litter carbon data.csv",
          row.names = F)



#-------------------------------------------------------------------------------#
#CP 11.5: Mismatching LF ID in Litter Carbon  and LF database -------------------------
# Input: SOC data; lf.csv;
# Output: number of plots with mismatching LF ID
#-------------------------------------------------------------------------------#
```

```
# Select the key variables
lcKV <- lit_car[,c("plot_id", "lf_id", "organic_carbon_in_litter_perc")]
# prepare a seperate variable for plot and lf in "lcKV"
lcKV$plot_lf_lc <- paste(lcKV$plot_id, lcKV$lf_id, sep = "_")
# prepare a seperate variable for plot and lf in "lf" database
lf$plot_lf_lf <- paste(lf$plot_plot_id, lf$lf_id, sep="_")
# plot_lf in scKV database not present LF database
lf_mismatch <- lcKV[!lcKV$plot_lf_lc %in% lf$plot_lf_lf,]
# add the plot_id that are entered in LF database
lf$count <- 1
lf_mismatch <- merge(lf_mismatch, lf[,c("plot_plot_id", "count")], by.x="plot_id",
                     by.y="plot_plot_id", all.x=T)

# prepare the outputs
if(dim(lf_mismatch)[1]==0){
  output11.5 <- data.frame(cp= "CP 11.5", cat="Mismatching lf_id in -Litter Carbon- and -LF- database",
                           plot_id = NA, issue= "No mismatch in lf_id", team=NA)
}else{
  lf_mismatch$issue <- ifelse(is.na(lf_mismatch$count), "plot not present in LF database",
                              "plot present in -Litter Carbon- and -LF- database but LF ID mismatched")
  lf_mismatch$cat <- "LF id mismatching between Litter Carbon and LF database"
```

```
  lf_mismatch$cp <- "CP 11.5"
  lf_mismatch <- merge(lf_mismatch, team_name, by.x="plot_id", by.y = "plot_plot_id", all.x=T)
  names(lf_mismatch)[1] <- "plot_id"
  output11.5 <- lf_mismatch[,c("cp", "cat", "plot_id", "issue", "team")]}

#export outputs
write.csv(output11.5,
        file = "./QAQC_outputs/11.5_Mismatching lf_id in litter carbon and LF data.csv",
        row.names = F)

#-------------------------------------------------------------------------------#
# CP 11.6: Outliers (extreme values) in litter carbon per hectare ----------------------
# Input: litter_dry_weight.csv, litter_carbon.csv
# Output: List of plots with outliers (extreme values) in litter carbon per hectare
#-------------------------------------------------------------------------------#
####### Preparation of data
#remove incorrect subplot names from data
lit_dry_wtF <- lit_dry_wt[lit_dry_wt$subplot_id!="1,2,&3",]
# select the key variables
lwKV <- lit_dry_wtF[,c("plot_id", "subplot_id", "lf_id", "litter_oven_dry_wt_g")]
lcKV <- lit_car[,c("plot_id", "lf_id", "organic_carbon_in_litter_perc")]
```

```
# merge litter carbon and lit_dry_wtF data by LF
lc <- merge(lwKV,lcKV, by.x=c("plot_id","lf_id"), by.y=c("plot_id", "lf_id"), all.x=T)
# remove the rows with NAs in organic carbon
lc <- lc[!is.na(lc$organic_carbon_in_litter_perc),]
# consider completely sampled subplots only
lc <- merge(lc, subplot_info[,c("plot_plot_id", "subp_id", "subp_details_subp_status_label")],
          by.x=c("plot_id", "subplot_id"), by.y=c("plot_plot_id", "subp_id"), all.x=T)
lc <- lc[lc$subp_details_subp_status_label=="Sampled completely",]
# compute litter carbon in ton
lc$lc_t <- (lc$litter_oven_dry_wt_g*(lc$organic_carbon_in_litter_perc/100))/1000000 #lc_t = litter carbon in ton
# add the nlcl with lc from lf
lcLf <- merge(lc, lf[,c("plot_plot_id", "lf_id", "nlcl")], by.x=c("plot_id","lf_id"),by.y=c("plot_plot_id","lf_id"), all.x=T)
# add the sampled area (1m*1m) per subplot
lcLf$lc_tHa <- lcLf$lc_t/0.0001

### Following portion may work after receiving all the data
# # prepare a data file with subplot sampled completely and sampled for soil
# subplot_infoT <- subplot_info[!is.na(subplot_info$soil_litter),]
# litdata <- subplot_infoT[subplot_infoT$subp_details_subp_status==1 &
#                          subplot_infoT$soil_litter==1,][,c("plot_plot_id", "subp_id",
#                                                           "subp_details_subp_status")]
```

```
### Identification of Outliers
# A table of statistics (mean and sd) is done
f116 <- function(x) c(n = length(x), Mean = mean(x), SD = sd(x))
BasicSat <- aggregate(lc_tHa ~ nlcl, data=lcLf[!is.na(ifelse(lcLf$lc_tHa==Inf,NA,
                                        lcLf$lc_tHa)),], FUN=f116)
BasicSat$n<-BasicSat$lc_tHa[,1]
BasicSat$Mean<-BasicSat$lc_tHa[,2]
BasicSat$SD<-BasicSat$lc_tHa[,3]

# Litter database and Statistical database are merged by nlcl
lcLfSt<-merge(lcLf, BasicSat[,c("nlcl", "n", "Mean", "SD")], by="nlcl", all.x=T)
lcLfStF <- lcLfSt[lcLfSt$n>3,]
# The Z value is estimated and Possible Outliers are identified
lcLfStF$Z<-((lcLfStF$lc_tHa-lcLfStF$Mean)/lcLfStF$SD)
lcLfStF$PO<-ifelse(lcLfStF$Z>4,1,0)
dim(lcLfStF)

# Identification of Outliers
lcLfStFf<-lcLfStF[!is.na(lcLfStF$PO) & lcLfStF$PO==1,]
```

```
### Possible Outliers are summarized and saved
# A Database of possible OL is created
if(dim(lcLfStFf)[1] == 0){
  output11.6 <- data.frame(cp="CP 11.6",cat="Outlier in Litter carbon per ha",plot_id=NA,
                           issue="No outlier in Litter carbon per ha", team = NA,
                           stringsAsFactors = F)
  } else {
    lcLfStFf$issue <- paste("Possible outlier in litter carbon t/ha (LCC:", lcLfStFf$nlcl,
                            ", C_tHa:", round(lcLfStFf$lc_tHa,1),
                            ", Mean:",round(lcLfStFf$Mean,1),
                            ", Z:",round(lcLfStFf$Z,1), ")", sep = "")
    lcLfStFf$cp <- "CP 11.6"
    lcLfStFf$cat <- "Outlier in litter carbon per hectare"
    lcLfStFf <- merge(lcLfStFf, team_name, by.x = "plot_id", by.y="plot_plot_id", all.x=T)
    names(lcLfStFf)[1] <- "plot_id"
    output11.6 <- lcLfStFf[,c("cp","cat","plot_id","issue", "team")]}

# export the outputs
write.csv(output11.6,
          file = "./QAQC_outputs/11.6_Possible outliners in Litter carbon per ha.csv",
          row.names = F)
```

```
#-------------------------------------------------------------------------------#
# CP 12: Litter dry wt related checks ------------------------------------------
#-------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------#
# CP 12.1: Identify the NAs in cells -------------------------------------------
# Input: litter dry weight data;
# Output: list of plots with NAs in cells
#-------------------------------------------------------------------------------#
# Select the key variables from Litter dry weight table
ldw <- lit_dry_wtF[,c("plot_id", "subplot_id", "lf_id", "litter_oven_dry_wt_g")]


# Identify the NAs and prepare the output table
if(dim(ldw[(is.na(ldw$plot_id) | is.na(ldw$subplot_id)| is.na(ldw$lf_id) |
          is.na(ldw$litter_oven_dry_wt_g)),])[1]==0){
  output12.1 <- data.frame(cp= "CP 12.1", cat="NA in litter dry weight data",
                           plot_id = NA, issue= "No NAs found", team=NA)
}else{
  ldw <- ldw[(is.na(ldw$plot_id) | is.na(ldw$subplot_id) | is.na(ldw$lf_id) |
```

```
            is.na(ldw$litter_oven_dry_wt_g)),]
  ldw$cp <- "CP 12.1"
  ldw$cat <- "NA in litter dry weight data"
  ldw$issue <- paste(ifelse(is.na(ldw$plot_id), "NA in plot_id",
                    ifelse(is.na(ldw$subplot_id), "NA in subp id",
                        ifelse(is.na(ldw$lf_id), "NA in lf",
                            "NA in litter carbon data"))))
  ldw$team <- merge(ldw, team_name, by.x="plot_id", by.y="plot_plot_id", all.x=T)
  names(ldw)[1]<- "plot_id"
  output12.1 <- ldw[,c("cp", "cat", "plot_id", "issue", "team")]}

# export the outputs
write.csv(output12.1,
        file = "./QAQC_outputs/12.1_NA in litter dry weight data.csv", row.names = F)



#-------------------------------------------------------------------------------#
# CP 12.2: Blanks in litter dry weight data ----------------------------------------
# Input: litter dry weight data;
# Output: list of plots, with blank cells
#-------------------------------------------------------------------------------#
```

```
# Select the key variables from Litter dry weight table
ldw <- lit_dry_wtF[,c("plot_id", "subplot_id", "lf_id", "litter_oven_dry_wt_g")]


# Idenfity the blank cells and prepare the output table
if(dim(ldw[(ldw$plot_id=="" | ldw$subplot_id=="" | ldw$lf_id=="" | ldw$litter_oven_dry_wt_g==""),])[1]==0){
  output12.2 <- data.frame(cp= "CP 12.2", cat="Blank cells litter dry weight data",
                           plot_id = NA, issue= "No blank cells found", team=NA)
}else{
  ldw <- ldw[(ldw$plot_id=="" | ldw$subplot_id=="" | ldw$lf_id=="" |
                ldw$litter_oven_dry_wt_g=="" ),]
  ldw$cp <- "CP 12.2"
  ldw$cat <- "Blank cells in litter dry wt data"
  ldw$issue <- paste(ifelse(ldw$plot_id=="", "Blank plot_no",
                            ifelse(ldw$subplot_id=="", "blank subp",
                                   ifelse(ldw$lf_id=="", "Blank lf",
                                          "Blank litter carbon data"))))
  ldw <- merge(ldw, team_name, by.x="plot_id", by.y="plot_plot_id", all.x=T)
  names(ldw)[1]<- "plot_id"
  output12.2 <- ldw[,c("cp", "cat", "plot_id", "issue", "team")]}


# export the outputs
```

```
write.csv(output12.2,
        file = "./QAQC_outputs/12.2_Blanks in litter dry weight data.csv", row.names = F)


#-------------------------------------------------------------------------------------#
# CP 12.3: Duplicate values in Litter Dry Weight data ---------------------------------
# Input: litter dry wt data;
# Output: list of plots with duplicate values.
#-------------------------------------------------------------------------------------#
# Select the key variables from Litter dry weight table
ldw <- lit_dry_wtF[,c("plot_id", "subplot_id", "lf_id", "litter_oven_dry_wt_g")]
ldw$count <- 1

# CP 12.3.1 find duplicate upto litter carbon
#-------------------------------------------------------------------------------------#
# Compute the number of duplicates by plot, subplot, lf and litter dry weight
ldw_dp <-aggregate(count~plot_id+subplot_id+lf_id+litter_oven_dry_wt_g, data = ldw, sum)
ldw_dp_1 <- ldw_dp[ldw_dp$count>1, ]

#add results to output
if(dim(ldw_dp_1)[1]==0){
  output12.3.1 <- data.frame(cp= "CP 12.3.1", cat="Duplicate values in litter dry wt",
```

```
                    plot_id = NA, issue= "No duplicates found up to litter dry wt",
                    team=NA)
}else{
  ldw_dp <- ldw_dp[-as.integer(rownames(ldw_dp_1)),]
  ldw_dp_1$cp <- "CP 12.3.1"
  ldw_dp_1$cat <- "Duplicate values in litter dry wt"
  ldw_dp_1$issue <- "values duplicated up to litter dry wt"
  ldw_dp_1 <- merge(ldw_dp_1, team_name, by.x="plot_id", by.y = "plot_plot_id", all.x=T)
  names(ldw_dp_1)[1] <- "plot_id"
  output12.3.1 <- ldw_dp_1[,c("cp", "cat", "plot_id", "issue", "team")]}



# CP 12.3.2 find duplicate upto land feature
#-------------------------------------------------------------------------------#
# Compute the number of duplicates by plot, subplot and lf
ldw_dp <-aggregate(count~plot_id+subplot_id+lf_id, data = ldw, FUN=sum)
ldw_dp_2 <- ldw[ldw$count>1,]

#add results to output
if(dim(ldw_dp_2)[1]==0){
  output12.3.2 <- data.frame(cp= "CP 5.3.2", cat="Duplicate values in litter dry wt",
```

```
                    plot_id = NA, issue= "No duplicates found up to land feature", team=99)
}else{
  ldw_dp <- ldw_dp[-as.integer(rownames(ldw_dp_2)),]
  ldw_dp_2$cp <- "CP 12.3.2"
  ldw_dp_2$cat <- "Duplicate values in litter dry wt"
  ldw_dp_2$issue <- "values duplicated up to lf"
  ldw_dp_2 <- merge(ldw_dp_2, team_name, by.x="plot_id", by.y = "plot_plot_id", all.x=T)
  names(ldw_dp_2)[1] <- "plot_id"
  output12.3.2 <- ldw_dp_2[,c("cp", "cat", "plot_id", "issue", "team")]}

# CP 12.3.3 find duplicate upto subplot
#-------------------------------------------------------------------------------#
ldw_dp <-aggregate(count~plot_id+subplot_id,
                   data = ldw_dp, FUN=sum)
ldw_dp_3 <- ldw_dp[ldw_dp$count>1,]

#add results to output
if(dim(ldw_dp_3)[1]==0){
  output12.3.3 <- data.frame(cp= "CP 12.3.3", cat="Duplicate values in litter dry wt",
                    plot_id = NA, issue= "No duplicates found up to subplot",
                    team=NA)
```

```
}else{
  ldw_dp <- ldw_dp[-as.integer(rownames(ldw_dp_3)),]
  ldw_dp_3$cp <- "CP 12.3.3"
  ldw_dp_3$cat <- "Duplicate values in litter dry wt"
  ldw_dp_3$issue <- "values duplicated up to subplot"
  ldw_dp_3 <- merge(ldw_dp_3, team_name, by.x="plot_id", by.y = "plot_plot_id", all.x=T)
  names(ldw_dp_3)[1] <- "plot_id"
  output12.3.3 <- ldw_dp_3[,c("cp", "cat", "plot_id", "issue", "team")]}

# Put all the duplicating outputs together
output12.3 <- rbind(output12.3.1, output12.3.2, output12.3.3)

# export the outputs
write.csv(output12.3,
          file = "./QAQC_outputs/12.3_Dulicates in litter dry weight data.csv", row.names = F)

#-------------------------------------------------------------------------------#
# CP 12.4: Consistency between canopy cover and leaf dry weight ----------------------
# Input: Litter dry weight data;
# Output: Box plot with outliers in litter dry weight data based on land class legends
#         and list of plots with strange values in litter weight
```

```
#-------------------------------------------------------------------------------#
# Select the key variables from Litter dry weight table
ldw <- lit_dry_wtF[,c("plot_id", "subplot_id", "lf_id", "litter_oven_dry_wt_g")]
# Aggregate the litter dry weight by plot
ldwF <- aggregate(data=ldw, litter_oven_dry_wt_g ~ plot_id+lf_id, sum)

#add nlcl
ldwLf <- merge(ldwF,lf[,c("plot_plot_id", "lf_id", "nlcl")], by.x=c("plot_id", "lf_id"),
          by.y=c("plot_plot_id", "lf_id"), all.x=TRUE)

#plot litter carbon data
ggplot(ldwLf, aes(y=litter_oven_dry_wt_g, x=nlcl))+
  geom_boxplot()+
  xlab("Land class legends")+ylab("leaf oven dry weigth (gm)")+
  ggtitle("Leaf dry weight (gm) in different land classes")


##### Add crown cover data by plot and lf
# Select key variables
lf_cc <- lf[,c("plot_plot_id", "lf_id", "lf_details_cover_min", "lf_details_cover_max"),]
# average the crown cover per plot and lf
```

```
lf_cc$cc_average <- (lf_cc$lf_details_cover_min + lf_cc$lf_details_cover_max)/2


#merge canopy cover
ldwLf <- merge(ldwLf, lf_cc[,c("plot_plot_id", "lf_id", "cc_average")],
               by.x= c("plot_id", "lf_id"), by.y = c("plot_plot_id","lf_id"),
               all.x = TRUE)
# remove NAs in cc_coverage
ldwLf <- ldwLf[!is.na(ldwLf$cc_average),]


#plot litter wt and canopy coverage
ggplot(ldwLf, aes(x=cc_average, y=litter_oven_dry_wt_g))+ geom_point()+
  xlab("Canopy coverage")+ylab("Oven dry weight of litter (g)")#+
#  facet_wrap(~nlcl)


#order the woven dry wt
ldwLf[order(-ldwLf$litter_oven_dry_wt_g, decreasing = TRUE),][1:10,]
ldwLf[order(-ldwLf$cc_average, decreasing = TRUE),][1:10,]


#select plots with cc_coverage is zero but litter dry wt is more than zero
#add results to output
if(dim(ldwLf[ldwLf$cc_average==0 & ldwLf$litter_oven_dry_wt_g>2,])[1]==0){
```

```
  output12.4 <- data.frame(cp = "CP 12.4", cat="litter dry wt and crown cov inconsistent",
                plot_id = NA, issue= "no inconsistency betwn litter_dry_wt and crown cov",
                team=NA)
}else{
  ldw_incons <- ldwLf[ldwLf$cc_average==0 & ldwLf$litter_oven_dry_wt_g>2,]
  ldw_incons$cp <- "CP 12.4"
  ldw_incons$cat <- "litter dry wt and crown cov inconsistent"
  ldw_incons$issue <- paste("litter drw wt ", ldw_incons$Oven_dry_litter_g,
                            "but crown coverage is ", ldw_incons$cc_average)
  ldw_incons <- merge(ldw_incons, team_name, by.x="plot_id", by.y = "plot_plot_id", all.x=T)
  names(ldw_incons)[1] <- "plot_id"
  output12.4 <- ldw_incons[,c("cp", "cat", "plot_id", "issue", "team")]}


# export the outputs
write.csv(output12.4,
          file = "./QAQC_outputs/12.4_Inconsistency btwn litter dry wt and canopy coverage.csv",
          row.names = F)


#-------------------------------------------------------------------------------------#
# CP 12.5: Mismatched lf_id in litter carbon and  litter dry weight data ----------
# Input: lit_car.csv and lit_dry_wt.csv, lf.csv
```

```
# Output: list of plots with mismatches with lf.csv by land feature id
#-------------------------------------------------------------------------------#
# Select Key variables from -Litter Carbon- database
lcKV <- lit_car[,c("plot_id", "lf_id")]
# Select Key variables from -Litter Dry Weight- database
lwKV <- lit_dry_wt[,c("plot_id", "lf_id")]
# select Key variables from -LF- database
lfKV <- lf[,c("plot_plot_id", "lf_id")]

###find mismatches in the lit_dry_wt2 and lit_car_work
#install.packages("sqldf")
require(sqldf)
### PLOT and LF present in -Litter Carbon- but absent in -Litter Weight- database
lcLf_NotIn_lwLf <- sqldf('SELECT * FROM lcKV EXCEPT SELECT * FROM lwKV')
# write the problem as issue
lcLf_NotIn_lwLf$issue <- paste("LF-", lcLf_NotIn_lwLf$lf_id,
                               "present in -Litter Carbon- but absent in -Litter Weight" )
### PLOT and LF present in -Litter Weight- but absent in -Litter Carbon- database
lwLf_NotIn_lcLf <- sqldf('SELECT * FROM lwKV EXCEPT SELECT * FROM lcKV')
# write the problem as issue
lwLf_NotIn_lcLf$issue <- paste("LF-", lwLf_NotIn_lcLf$Land_feature_no,
```

```
                    "present in -Litter Weight- but absent in -Litter Carbon" )


### PLOT and LF present in -Litter Weight- but absent in -LF- database
lwLf_NotIn_Lf <- sqldf('SELECT * FROM lwKV EXCEPT SELECT * FROM lfKV')
# Find the plot_id that present in LF database
lf$count <- 1
lwLf_NotIn_Lf <- merge(lwLf_NotIn_Lf, lf[,c("plot_plot_id", "count")], by.x="plot_id", by.y="plot_plot_id", all.x=T)
lwLf_NotIn_Lf$issue <- ifelse(is.na(lwLf_NotIn_Lf$count),
                          paste("LF-", lwLf_NotIn_lcLf$lf_id,
                              "present in -Litter Weight- but absent in -LF" ),
                          "plot present in -Litter Weight- and -LF- database but LF ID mismatched")


# Prepare the output table
output12.5 <- rbind(lcLf_NotIn_lwLf[,c("plot_id", "issue")], lwLf_NotIn_lcLf[,c("plot_id", "issue")],
                lwLf_NotIn_Lf[,c("plot_id", "issue")])
if(dim(output12.5)[1]==0){
  output12.5 <- data.frame(cp = "CP 12.5", cat= "lf_id mismatch between Litter carbon, Litter Weight and LF",
                       plot_id = NA, issue= "No mismatch found", team=NA)
}else{
  output12.5$cp <- "CP 12.5"
  output12.5$cat <- "lf_id mismatch between Litter carbon, Litter Weight and LF"
```

```
  output12.5 <- merge(output12.5, team_name, by.x="plot_id", by.y="plot_plot_id", all.x=T)
  names(output12.5)[1] <- "plot_id"
}

# export the outputs
write.csv(output12.5,
          file = "./QAQC_outputs/12.5_Mismatching lf_id in litter_dry_wt, lit carbon and lf.csv",
          row.names = F)

#-------------------------------------------------------------------------------#
# CP 13: Electric conductivity and pH related checks
#-------------------------------------------------------------------------------#

#-------------------------------------------------------------------------------#
# CP 13.1: NAs in electricity conductivity and pH data ---------------------------------
# Input: electric conductivity and pH data;
# Output: list of plots with NAs in cells
#-------------------------------------------------------------------------------#
# Select the key variables
sph <- salinity_ph[,c("plot_id", "soil_depth_cm", "lf_id", "pH", "EC_mS_per_cm", "location_zone")]
```

```
# Identify the NAs and prepare the output table
if(dim(sph[(is.na(sph$plot_id) | is.na(sph$depth)|
          is.na(sph$lf) | is.na(sph$pH) | is.na(sph$EC_mS_per_cm)),])[1]==0){
  output13.1 <- data.frame(cp= "CP 13.1", cat="NA in salinity and pH data",
                           plot_id = NA, issue= "No NAs found", team=NA)
}else{
  sph <- sph[(is.na(sph$plot_id) | is.na(sph$depth)|
              is.na(sph$lf) | is.na(sph$pH) | is.na(sph$EC_mS_per_cm)),]
  sph$cp <- "CP 13.1"
  sph$cat <- "NA in salinity and pH data"
  sph$issue <- paste(ifelse(is.na(sph$plot_id), "NA in plot_id",
                            ifelse(is.na(sph$depth), "NA in soil depth",
                                   ifelse(is.na(sph$lf), "NA in lf",
                                          ifelse(is.na(sph$pH), "NA in pH",
                                                 "NA in EC")))))
  sph$team <- merge(sph, team_name, by.x="plot_id", by.y="plot_plot_id", all.x=T)
  names(sph)[1]<- "plot_id"
  output13.1 <- sph[,c("cp", "cat", "plot_id", "issue", "team")]}

#export outputs
write.csv(output13.1, file = "./QAQC_outputs/13.1_NA in salinity and pH data.csv", row.names = F)
```

```
#-------------------------------------------------------------------------------#
# CP 13.2: Identify the blank cells ----------------------------------------------------
# Input: EC and pH data;
# Output: list of plots, with blank cells
#-------------------------------------------------------------------------------#
# Select the key variables
sph <- salinity_ph[,c("plot_id", "soil_depth_cm", "lf_id", "pH", "EC_mS_per_cm", "location_zone")]

# Identify the blanks and prepare the output table
if(dim(sph[(sph$plot_id=="" | sph$depth=="" | sph$lf=="" |
          sph$pH==""| sph$EC_mS_per_cm==""),])[1]==0){
  output13.2 <- data.frame(cp= "CP 6.2", cat="Blank cells in salinity and pH data",
                           plot_id = NA, issue= "No blank cells found", team=NA)
}else{
  sph <- sph[(sph$plot_id=="" | sph$depth=="" | sph$lf=="" |
              sph$pH==""| sph$EC_mS_per_cm==""),]
  sph$cp <- "CP 13.2"
  sph$cat <- "Blank cells in salinity and pH data"
  sph$issue <- paste(ifelse(sph$plot_id=="", "Blank plot_no",
                            ifelse(sph$depth=="", "blank depth",
```

```
                        ifelse(sph$lf=="", "blank lf",
                               ifelse(sph$pH=="", "Blank pH",
                                      "Blank electric conductivity")))))
  sph <- merge(sph, team_name, by.x="plot_id", by.y="plot_plot_id", all.x=T)
  names(sph)[1]<- "plot_id"
  output13.2 <- sph[,c("cp", "cat", "plot_id", "issue", "team")]}

#export outputs
write.csv(output13.2, file = "./QAQC_outputs/13.2_Blanks in salinity and pH data.csv", row.names = F)

#-------------------------------------------------------------------------------#
# CP 13.3: Duplicate values in salinity and pH data ----------------------------------
# Input: electric conductivity and pH data;
# Output: list of plots with duplicate values
#-------------------------------------------------------------------------------#
# Prepare the working data
# Select the key variables
sph <- salinity_ph[,c("plot_id", "soil_depth_cm", "lf_id", "pH", "EC_mS_per_cm", "location_zone")]
sph$count <- 1

# CP 13.3.1 find duplicate upto Electric conductivity and pH data
```

```
#-------------------------------------------------------------------------------------#
sph_dp <-aggregate(count~plot_id+soil_depth_cm+lf_id+pH+EC_mS_per_cm, data = sph, sum)
sph_dp_1 <- sph_dp[sph_dp$count>1, ]

#add results to output
if(dim(sph_dp_1)[1]==0){
  output13.3.1 <- data.frame(cp= "CP 13.3.1", cat="Duplicate values in EC and pH data",
                             plot_id = NA, issue= "No duplicates found up to EC", team=NA)
}else{
  sph_dp <- sph_dp[-as.integer(rownames(sph_dp_1)),]
  sph_dp_1$cp <- "CP 13.3.1"
  sph_dp_1$cat <- "Duplicate values in EC and pH data"
  sph_dp_1$issue <- "values duplicated up to EC"
  sph_dp_1 <- merge(sph_dp_1, team_name, by.x="plot_id", by.y = "plot_plot_id", all.x=T)
  names(sph_dp_1)[1] <- "plot_id"
  output13.3.1 <- sph_dp_1[,c("cp", "cat", "plot_id", "issue", "team")]}


# CP 13.3.2 find duplicate upto land feature
#-------------------------------------------------------------------------------------#
sph_dp <-aggregate(count~plot_id+soil_depth_cm+lf_id+pH, data = sph, FUN=sum)
```

```
sph_dp_2 <- sph_dp[sph_dp$count>1,]


#add results to output
if(dim(sph_dp_2)[1]==0){
  output13.3.2 <- data.frame(cp= "CP 13.3.2", cat="Duplicate values in EC and pH data",
                             plot_id = NA, issue= "No duplicates found up to pH", team=NA)
}else{
  sph_dp <- sph_dp[-as.integer(rownames(sph_dp_2)),]
  sph_dp_2$cp <- "CP 13.3.2"
  sph_dp_2$cat <- "Duplicate values in litter dry wt"
  sph_dp_2$issue <- "values duplicated up to lf"
  sph_dp_2 <- merge(sph_dp_2, team_name, by.x="plot_id", by.y="plot_plot_id", all.x=T)
  names(sph_dp_2)[1] <- "plot_id"
  output13.3.2 <- sph_dp_2[,c("cp", "cat", "plot_id", "issue", "team")]}



# CP 13.3.3 find duplicate upto subplot
#-------------------------------------------------------------------------------#
sph_dp <-aggregate(count~plot_id+soil_depth_cm+lf_id, data = sph, FUN=sum)
sph_dp_3 <- sph_dp[sph_dp$count>1,]
```

```
#add results to output
if(dim(sph_dp_3)[1]==0){
  output13.3.3 <- data.frame(cp= "CP 13.3.3", cat="Duplicate values in EC and pH data",
                             plot_id = NA, issue= "No duplicates found up to lf", team=NA)
}else{
  sph_dp <- sph_dp[-as.integer(rownames(sph_dp_3)),]
  sph_dp_3$cp <- "CP 13.3.3"
  sph_dp_3$cat <- "Duplicate values in EC and pH data"
  sph_dp_3$issue <- "values duplicated up to lf"
  sph_dp_3 <- merge(sph_dp_3, team_name, by.x="plot_id", by.y = "plot_plot_id", all.x=T)
  names(sph_dp_3)[1] <- "plot_id"
  output13.3.3 <- sph_dp_3[,c("cp", "cat", "plot_id", "issue", "team")]}

# Put all outputs together
output13.3 <- rbind(output13.3.1, output13.3.2, output13.3.3)

#export outputs
write.csv(output13.3, file = "./QAQC_outputs/13.3_Duplicates in salinity and pH.csv",
          row.names = F)

##### some visuals on salinity and pH per LCC
```

```
#plot EC and pH data by nlcl
sph_4plot <- sph[(sph$EC_mS_per_cm!="no sample " & sph$EC_mS_per_cm!="les than 10 g "),]
# add nlcl
sph_4plot <- merge(sph_4plot, lf[,c("plot_plot_id", "lf_id", "nlcl")],
                   by.x = c("plot_id", "lf_id"), by.y = c("plot_plot_id", "lf_id"),
                   all.x=T)
sph_4plot <- sph_4plot[!is.na(sph_4plot$nlcl),]
sph_4plot$EC_mS_per_cm=as.numeric(sph_4plot$EC_mS_per_cm)
sph_4plot$pH=as.numeric(sph_4plot$pH) #to convert character into numeric

#add nlcl
ggplot(sph_4plot, aes(x= nlcl, y=pH))+
  geom_boxplot()

ggplot(sph_4plot, aes(x=nlcl, y=EC_mS_per_cm))+
  geom_boxplot()

#plot EC Vs pH
ggplot(sph_4plot, aes(x=pH, y=EC_mS_per_cm))+
  geom_point()
```

```
#-------------------------------------------------------------------------------#
# CP 13.4: Missing sample or having sample less than 10g ------------------------------
# Input: electric conductivity and pH data;
# Output: list of plots missing sample or having sample less than 10g
#-------------------------------------------------------------------------------#
# Select the key variables
sph <- salinity_ph[,c("plot_id", "soil_depth_cm", "lf_id", "pH", "EC_mS_per_cm", "location_zone")]


#add results to output
if(dim(sph[(sph$EC_mS_per_cm=="no sample " |
          sph$EC_mS_per_cm=="les than 10 g "),])[1]==0){
  output13.4 <- data.frame(cp= "CP 13.4", cat="Problem with sample for analysing EC and pH data",
                           plot_id = NA, issue= "No problems found with sampmles", team=NA)
}else{
  sph_pr <- sph[(sph$EC_mS_per_cm=="no sample " |  sph$EC_mS_per_cm=="les than 10 g "),]
  sph_pr$cp <- "CP 13.4"
  sph_pr$cat <- "Problem with sample for analysing EC and pH data"
  sph_pr$issue <- paste(sph_pr$EC_mS_per_cm,"found")
  sph_pr <- merge(sph_pr, team_name, by.x="plot_id", by.y = "plot_plot_id", all.x=T)
  names(sph_pr)[1] <- "plot_id"
  output13.4 <- sph_pr[,c("cp", "cat", "plot_id", "issue", "team")]}
```

```
#export outputs
write.csv(output13.4, file = "./QAQC_outputs/13.4_NA in litter carbon data.csv", row.names = F)


#-------------------------------------------------------------------------------#
# CP 13.5: Mismatching lf_id in salinity_pH and LF database ----------------------
# Input: ec_and_ph.csv and lf.csv
# Output: list of plots having lf_id mismatching lf database]
#-------------------------------------------------------------------------------#
# Select Key variables from -Salinity_pH- database
epKV <- salinity_ph[,c("plot_id", "lf_id")]
# select Key variables from -LF- database
lfKV <- lf[,c("plot_plot_id", "lf_id")]

###find mismatches in the lit_dry_wt2 and lit_car_work
#install.packages("sqldf")
require(sqldf)
# PLOT and LF present in -salinity_pH- but absent in -LF- database
epLf_NotIn_Lf <- sqldf('SELECT * FROM epKV EXCEPT SELECT * FROM lfKV')
# Find the plot_id that present in LF database
lf$count <- 1
```

```
epLf_NotIn_Lf <- merge(epLf_NotIn_Lf, lf[,c("plot_plot_id", "count")], by.x="plot_id", by.y="plot_plot_id", all.x=T)


# Prepare the output table
if(dim(epLf_NotIn_Lf)[1]==0){
  output13.5 <- data.frame(cp = "CP 6.5", cat= "lf_id mismatch between -salinity_pH- and -LF",
                           plot_id = NA, issue= "No mismatch found", team=NA)
}else{
  epLf_NotIn_Lf$cp <- "CP 13.5"
  epLf_NotIn_Lf$cat <- "lf_id mismatch between -salinity_pH- and -LF"
  epLf_NotIn_Lf$issue <- ifelse(is.na(epLf_NotIn_Lf$count),
                                paste("Plot-",epLf_NotIn_Lf$plot_id, ", LF-",
                                      epLf_NotIn_Lf$lf_id,
                                      "present in -salinity_pH- but absent in -LF" ),
                                "plot present in -salinity_pH- and -LF- database but LF ID mismatched")
  epLf_NotIn_Lf <- merge(epLf_NotIn_Lf, team_name, by.x="plot_id", by.y="plot_plot_id", all.x=T)
  names(epLf_NotIn_Lf)[1] <- "plot_id"
  output13.5 <- epLf_NotIn_Lf[,c("cp", "cat", "plot_id", "issue", "team")]
}


#export outputs
write.csv(output13.5, file = "./QAQC_outputs/13.5_Mismatch in lf id of salinity and LF data.csv", row.names = F)
```

```
# ---------------------------------------- End ----------------------------------------#



### Export the outputs ------------------------------------------------------------------
# 1. Export outputs on plot, subplot and land feature related checks
output1 <- rbind(output1.1, output1.2, output1.3, output1.4, output1.5, output1.6,
                 output1.7, output1.8, output1.9, output1.10, output1.11, output1.12,
                 output1.13, output1.14, output1.15)
write.csv(output1, file="./QAQC_outputs/1_Plot, subp and LF related QC outputs.csv", row.names = F)

# 2. Export outputs on tree and sapling related checks
output2 <- rbind(output2.1, output2.2, output2.3, output2.4, output2.5,output2.6,
                 output2.7, output2.8, output2.9, output2.10, output2.11, output2.12,
                 output2.13, output2.14, output2.15, output2.16, output2.17)
write.csv(output2, file="./QAQC_outputs/2_Tree and sapling related QC outputs.csv", row.names = F)

# 3. Export outputs on Standing dead trees
```

```
output3 <- rbind(output3.1, output3.2)
write.csv(output3, file="./QAQC_outputs/3_Standing dead tree related QC outputs.csv", row.names = F)

# 4. Export outputs on stumps
output4 <- rbind(output4.1, output4.2)
write.csv(output4, file="./QAQC_outputs/4_Stump related QC outputs.csv", row.names = F)

# 5. Export outputs on Bamboo
output5 <- rbind(output5.1, output5.2)
write.csv(output5, file="./QAQC_outputs/5_Bamboo related QC outputs.csv", row.names = F)

# 6. Export outputs on time taken for surveying a plot
output6
# 7. Export outputs on estimated attributes
output7 <- rbind(output7.1, output7.2, output7.3, output7.4, output7.5, output7.6,
                 output7.7, output7.8, output7.9, output7.10, output7.11, output7.12,
                 output7.13, output7.14, output7.15)
write.csv(output7, file="./QAQC_outputs/7_Estimated attributes related QC outputs.csv", row.names = F)

# 8. Export outputs on soil bulk density
output8 <- rbind(output8.1, output8.2, output8.3, output8.4, output8.5, output8.6,
```

```
            output8.7)
write.csv(output8, file="./QAQC_outputs/8_Bulk density related QC outputs.csv", row.names = F)

# 9. Export outputs on soil organic carbon
output9 <- rbind(output9.1, output9.2, output9.3, output9.4, output9.5, output9.7)
write.csv(output9, file="./QAQC_outputs/9_Organic carbon related QC outputs.csv", row.names = F)

# 10. Export outputs on soil texture
output10 <- rbind(output10.1, output10.2, output10.3, output10.4, output10.5)
write.csv(output10, file="./QAQC_outputs/10_Soil texture related QC outputs.csv", row.names = F)

# 11. Export outputs on litter carbon
output11 <- rbind(output11.1, output11.2, output11.3, output11.4, output11.5,
             output11.6)
write.csv(output11, file="./QAQC_outputs/11_Litter carbon related QC outputs.csv", row.names = F)

# 12. Export outputs on litter dry weight
output12 <- rbind(output12.1, output12.2, output12.3, output12.4, output12.5)
write.csv(output12, file="./QAQC_outputs/12_Litter dry wt related QC outputs.csv", row.names = F)

# 13. Export outputs on salinity and electric conductivity
```

```
output13 <- rbind(output13.1, output13.2, output13.3, output13.4, output13.5)
write.csv(output13, file="./QAQC_outputs/13_Salinity and pH related QC outputs.csv", row.names = F)

# Export all output in one file
all_qc_output <- rbind(output1, output2, output3, output4, output5, output6, output7,
                       output8, output9, output10, output11)
write.csv(all_qc_output, file="./QAQC_outputs/All QC outputs.csv", row.names = F)
```