Scrinzi G, Floris A, Clementel F, Bernardini V, Chianucci F, Greco S, Michelini T, Penasa A, Puletti N, Rizzo M, Turco R, Corona P (2017). **Modelli di stima del volume e delle fitomasse del soprassuolo arboreo delle principali formazioni forestali della Calabria mediante dati LiDAR.**
Forest@ - Rivista di Selvicoltura ed Ecologia Forestale – doi: 10.3832/efor2399-014

Materiale Supplementare

Tab. S1 - Coefficienti moltiplicatori di trasformazione del volume a ettaro in fitomassa a ettaro, calcolati sulle composizioni medie ponderate dei gruppi modellistici AlForLab (Ps1: peso anidro fitomassa legnosa di fusto e rami, fino a 5 cm di diametro minimo; Ps2: peso anidro fitomassa epigea totale, compresa fascina e fogliame; Pf: peso fresco fitomassa legnosa avente peso anidro Ps1).

Dettaglio modelli	Tipo di formazione	Ps1	Ps2	Pf
elevato	Abetine di Abete bianco	0.439	0.585	0.974
	Pinete di Pino laricio	0.389	0.447	0.917
	Form. di conifere mediterranee	0.477	0.667	0.991
	Querceti sempreverdi	0.748	1.019	1.234
	Eucalitteti	0.607	0.738	0.985
	Faggete	0.614	0.772	1.142
	Castagneti	0.496	0.622	1.009
	Querceti caducifogli	0.630	0.786	1.112
	Form. ad altre latifoglie	0.472	0.620	0.965
medio	Form. Conifere montane	0.376	0.428	0.906
	Form. miste di conifere/latifoglie montane	0.623	0.783	1.151
	Form. di conifere mediterranee	0.467	0.657	0.983
	Form. Latifoglie non montane	0.444	0.615	0.939
	Form. Latifoglie montane	0.611	0.782	1.083
	Castagneti puri e misti a prevalenza di castagno	0.496	0.622	1.009
basso	Formazioni di conifere	0.394	0.485	0.919
	Formazioni di latifoglie	0.586	0.742	1.087
	Formazioni miste conifere/latifoglie	0.482	0.612	1.005

Fig. S1 - Copertura voli LiDAR della Regione Calabria (Piano Straordinario di Telerilevamento ambientale – MATTM 2014).

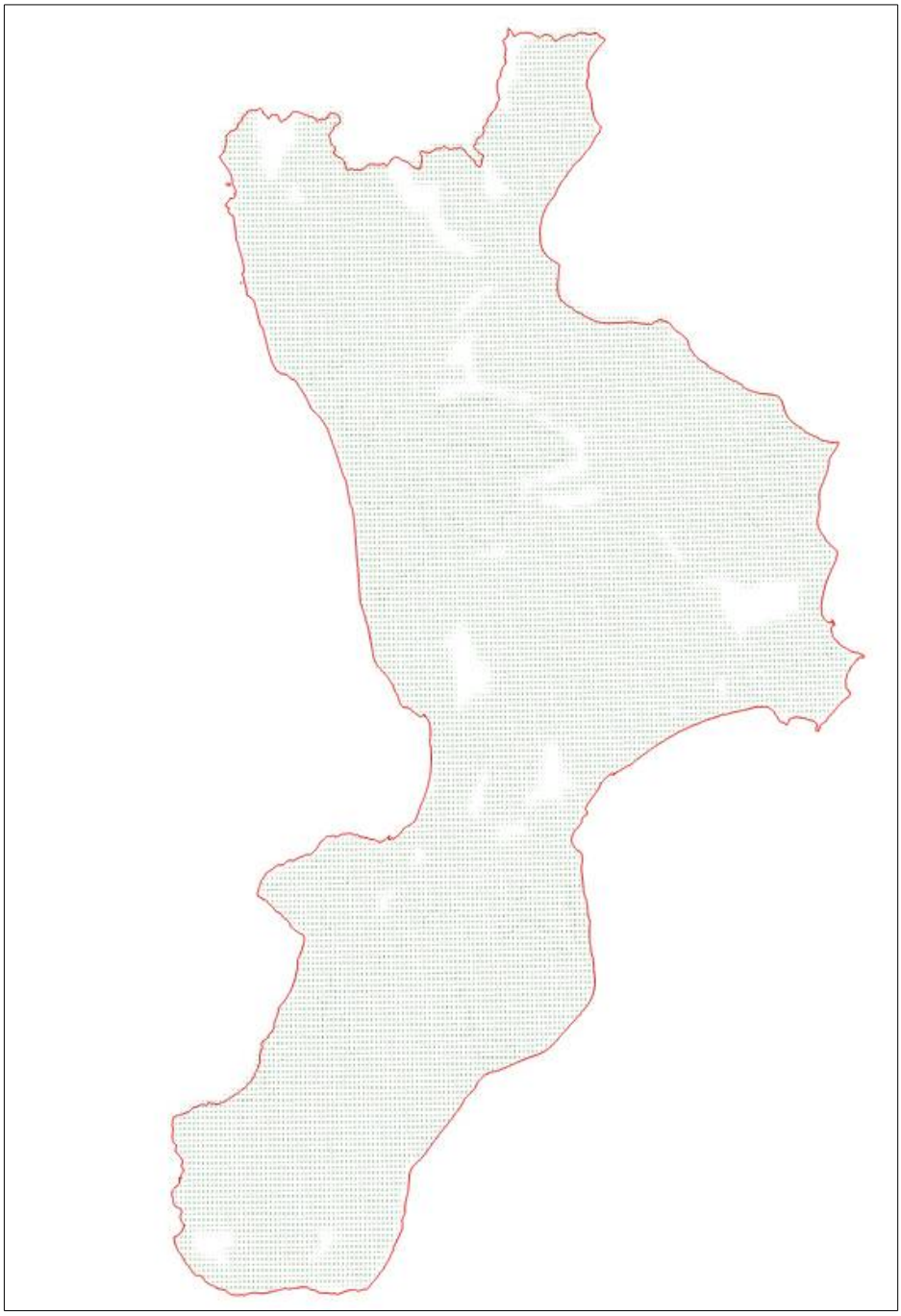


Fig. S2 - Ortofotocarta (a sinistra) e CHM LiDAR (a destra) del comune di Stilo (Serre calabresi).

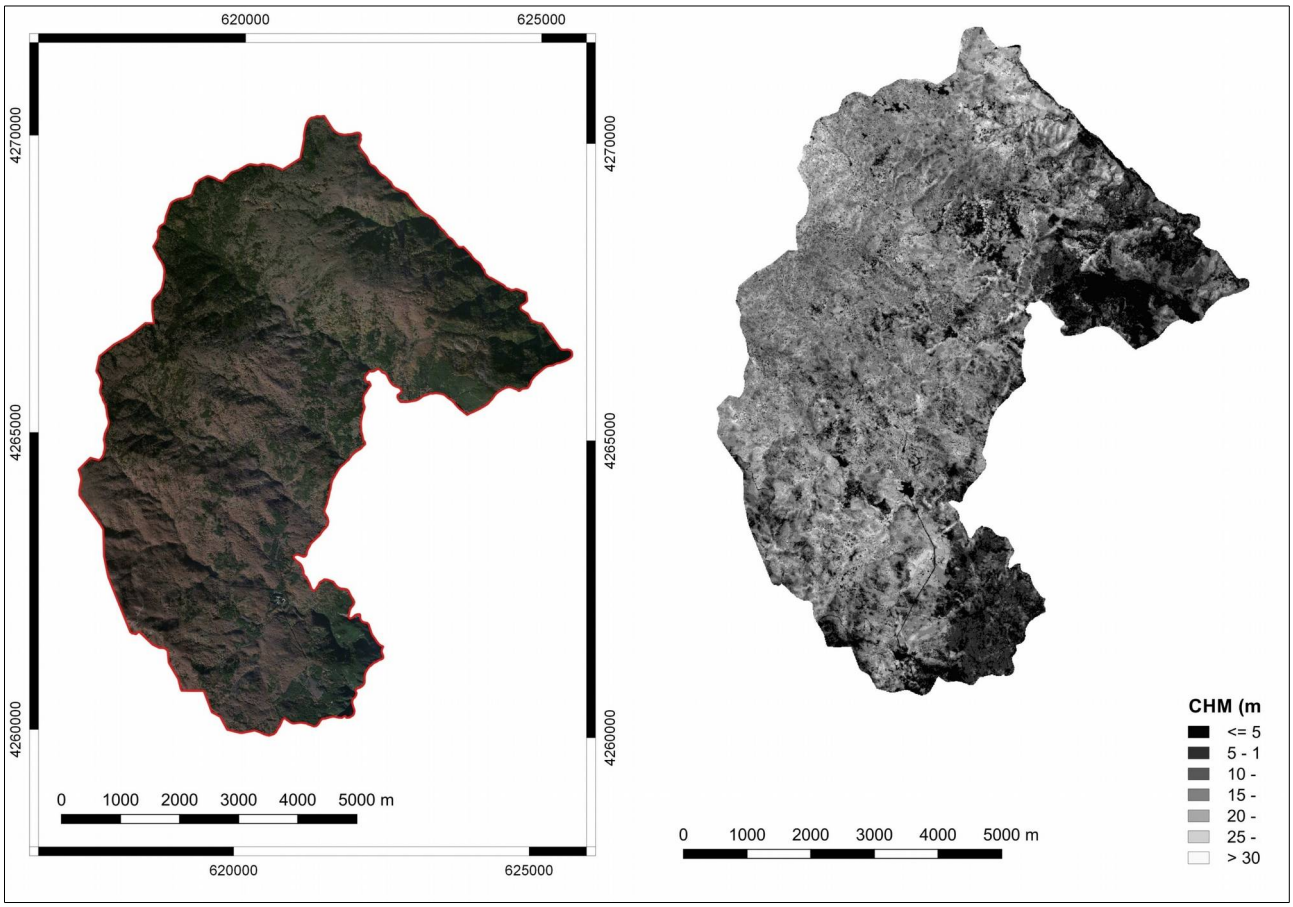
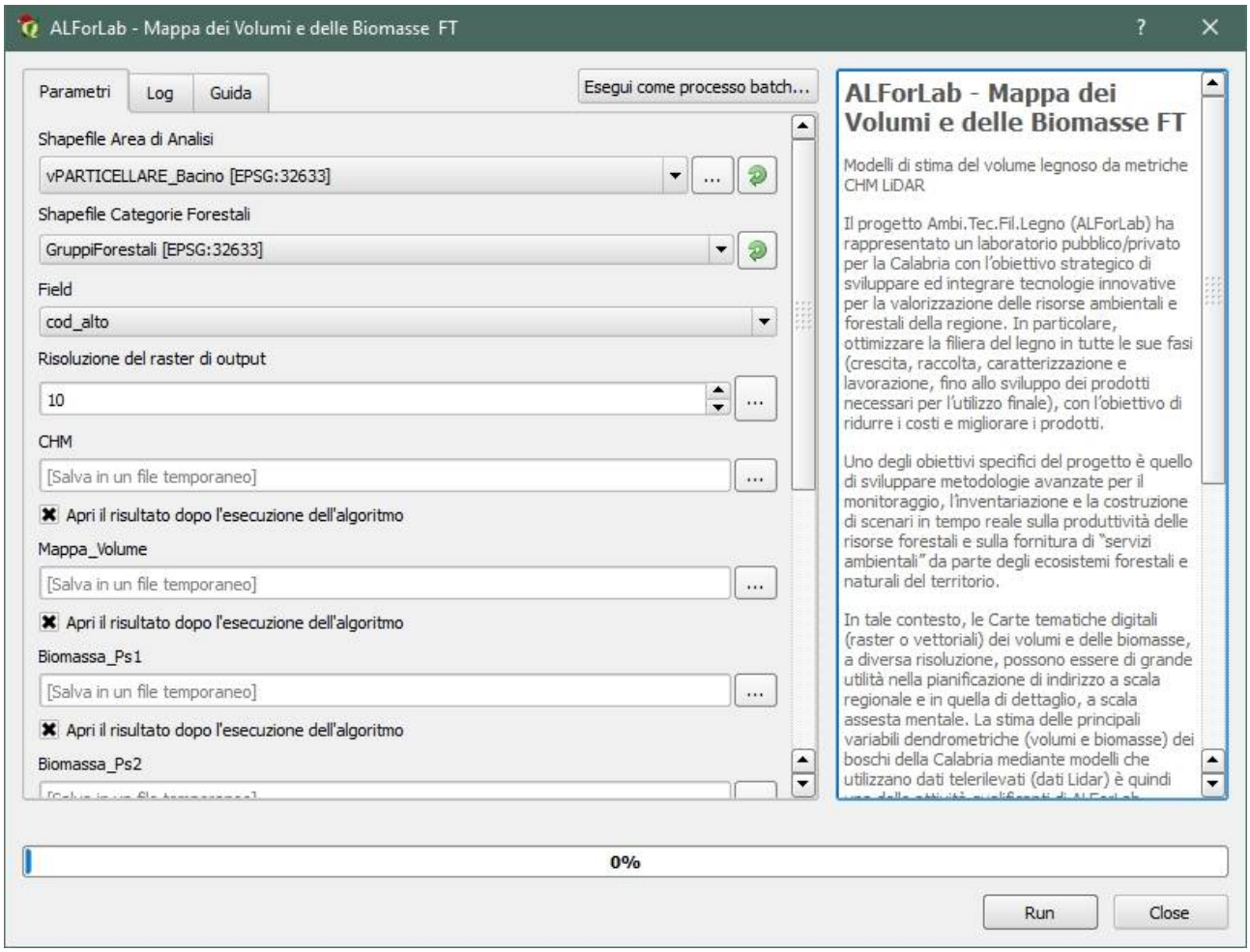


Fig. S3 - Interfaccia utente del modulo di QGIS (implementato come *script* R) per la stima del volume e delle fitomasse e la produzione dei relativi piani informativi digitali.



Scrinzi G, Floris A, Clementel F, Bernardini V, Chianucci F, Greco S, Michelini T, Penasa A, Puletti N, Rizzo M, Turco R, Corona P (2017). **Modelli di stima del volume e delle fitomasse del soprassuolo arboreo delle principali formazioni forestali della Calabria mediante dati LiDAR.**
Forest@ - Rivista di Selvicoltura ed Ecologia Forestale – doi: 10.3832/efor2399-014

Box S1 - *Script* di R per la produzione di piani tematici GIS dei volumi e delle fitomasse.

CHM.R

```
#*****************************************************************************
#
# # R version 3.3.0 (2016-05-03)
# # Platform: x86_64-w64-mingw32/x64 (64-bit)
# # Running under: Windows >= 8 x64 (build 9200)
# #
# # Autore: Tamara Michelini <tamara.michelini@yahoo.it>
# # Data: dicembre 2016
#
# ============================== **DESCRIZIONE** ==============================
# Lo script esegue il calcolo del CHM a partire da DTM e DSM su uno o più tiles.
#
# Tutti i dati di input vengono richiesti lanciando il programma (Source).
#
# INPUT:
#    - DTMFolder: cartella nella quale sono salvati i DTM
#    - DSMFolder: cartella nella quale sono salvati i DSM
#    - thr: soglia di altezza minima sotto la quale al CHM è dato valore di 0 m
#      (taglio per rimuovere dal CHM "rumore" ed elementi non arborei)
#    - thrMM: Soglia di altezza massima al di sopra della quale al CHM è dato
#      valore pari alla soglia (taglio per rimuovere valori eccessivamente elevati
#      prodotti da elementi non arborei)
#    - pathOut: cartella dove salvare i risultati
#
# NOTA: i primi 9 caratteri del nome del file DTM devono essere uguali all'omologo
#       DSM (es. D38421633_DTM, D38421633_DSM). Se non lo sono, lanciando lo script
#       appare un messaggio di errore.
#
# OUTPUT:
#    - file raster del CHM: salvato nella cartella scelta come output
#
# *****************************************************************************
#----------------------------------- INPUT -----------------------------------

# Select the folders where the all DTM and DSM files are saved
DTMFolder <- choose.dir(default = "",
                        caption = "Cartella nella quale sono salvati i DTM")
DSMFolder <- choose.dir(default = "",
                        caption = "Cartella nella quale sono salvati i DSM")

# The minimum height of the CHM
thr    <- readline(prompt="Soglia di altezza minima sotto la quale al CHM è dato valore di 0 m
(taglio per rimuovere dal CHM 'rumore' ed elementi non arborei): \n")
thrMM <- readline(prompt="Soglia di altezza massima al di sopra della quale al CHM è dato valore
pari alla soglia (taglio per rimuovere valori eccessivamente elevati prodotti da elementi non
arborei): \n")

# Output folder
pathOut <- choose.dir(caption = "Cartella dove salvare i risultati")

# ---------------------------------- LIBRERIA ---------------------------------

# Check if packages are installated and eventually installing them
require("raster")
if ("raster" %in% rownames(installed.packages()) == FALSE) {
  install.packages("raster")
  library("raster")
}

# ------------------------------- FUNZIONI ------------------------------------

# Rename the files into the two folder to avoid error during the sort
rename <- function(rasName) {
  file.rename(paste0(rasName),
```

```
              paste0(dirname(rasName),
                     "/",
                     substr(basename(rasName), 1, 9),
                     "_",
                     substr(basename(dirname(rasName)),1,3),
                     "_UTM33_1m.tif") )
}
#
# Creo una funzione che legge e sottrae i raster di DSM e DTM e che attribuisca
# valore 0 quando CHM è minore di 2 m e valore thrMM quando maggiore di thrMM
CHM <- function(DTMlist, DSMlist, CHMoutputFolder, thr) {
  grdDTM <- raster(DTMlist)
  grdDSM <- raster(DSMlist)
  CHM <- grdDSM - grdDTM
  CHM[CHM < thr] <- 0
  CHM[CHM >= thrMM] <- thrMM
  writeRaster(CHM,
              paste0(CHMoutputFolder,
                     "/",
                     substr(basename(DTMlist), 1, 9),
                     "CHM_min_max.tif"),
              format = "GTiff")
  }
#
#-------------------------------- CONTROLLO ----------------------------------
# For each tile checks if both DTM and DSM exist

# 1. Take the first part of the name of the DTM files and DSM files
DTMsName <- list.files(DTMFolder, full.names = TRUE)
DSMsName <- list.files(DSMFolder, full.names = TRUE)
#
# 2. Check if are equal. If they aren't return a Warning message with the name
# of the missing file/files
check <- function() {
  if (setequal(substr(basename(DTMsName), 1, 9),
               substr(basename(DSMsName), 1, 9)) == FALSE) {
    if (length(setdiff(substr(basename(DTMsName), 1, 9),
                       substr(basename(DSMsName), 1, 9))) > 0) {
      warning("The following ",
              length(setdiff(
                substr(basename(DTMsName), 1, 9),
                substr(basename(DSMsName), 1, 9))),
              " DSM tiles are missed: \n",
              setdiff(substr(basename(DTMsName), 1, 9),
                      substr(basename(DSMsName), 1, 9)),
              "\n",
              call. = FALSE)
    }
    else {warning("The following ",
                  length(setdiff(
                    substr(basename(DSMsName), 1, 9),
                    substr(basename(DTMsName), 1, 9))),
                  " DTM tiles are missed: \n",
                  setdiff(substr(basename(DSMsName), 1, 9),
                          substr(basename(DTMsName), 1, 9)),
                  "\n",
                  call. = FALSE)
    }
  }
}


#------------------------- PROCESSING AND SAVE OUTPUT --------------------------

# Eseguo il controllo
check()

# Rename the files and sort
rename(DTMsName)
rename(DSMsName)
```

```
#
filesDTM <-
  sort(list.files((DTMFolder), recursive = TRUE, full.names = TRUE))

filesDSM <-
  sort(list.files((DSMFolder), recursive = TRUE, full.names = TRUE))

# Compute the CHM with the CHM function
lapply(1:length(filesDTM), function(i)
  CHM(filesDTM[i], filesDSM[i], pathOut, as.numeric(thr)))

print("L'analisi è finita. Il risultato si trova nella cartella di output.")
```

MosaicRaster.R

```
#*******************************************************************************
#
# # R version 3.3.0 (2016-05-03)
# # Platform: x86_64-w64-mingw32/x64 (64-bit)
# # Running under: Windows >= 8 x64 (build 9200)
# #
# # Autore: Tamara Michelini <tamara.michelini@yahoo.it>
# # Data: dicembre 2016
#
#*******************************************************************************
# ============================== **DESCRIZIONE** ================================
#
# MosaicRaster.R
#
# Lo script unisce due o più raster in un singolo raster riempiendo eventuali "buchi"
# (no-data) generati durante il processo di unione.
# Il riempimento avviene assegnando il valore medio delle celle vicine all'interno
# di una matrice di dimensione n x n pixel e aventi valore diverso da "no-data".
# E' inoltre possibile cambiare la risoluzione del raster di output.
#
# Tutti i dati di input vengono richiesti lanciando il programma (Source).
#
# INPUT:
#   - rasFolder: cartella nella quale si trovano i DTM in formato raster
#   - pathOut: cartella nella quale salvare i risultati
#   - yn: scelta dell'utente se intende cambiare ('s') o no ('n') la risoluzione
#     del raster di output
#   - m: fattore di aggregazione per il cambio di risoluzione [opzionale]
#     (NOTA: si tratta di un fattore che viene applicato alla risoluzione originale
#     dei raster, quindi se la risoluzione iniziale è di 5 m, per ottenere una
#     risoluzione di 10 m occorre porre m pari a 2 (5*2=10))
#
# OUTPUT:
#   - file raster unito ("rasUnito.tif") salvato nella cartella di output
#
#   NOTA: la finestra mobile per il calcolo del valore medio di sostituzione è
#   posta pari a 5. Se necessario è possibile cambiarla (riga 88).
#
# La funzione "InstallPackages" installa una lista di librerie necessarie.
#
# *******************************************************************************
# ------------------------------------ INPUT ------------------------------------

# Select the folder where the rasters are saved
rasFolder <- choose.dir(default = "",
  caption = "Cartella nella quale si trovano i DTM in formato raster")

# Cartella di output
pathOut <- choose.dir(caption = "Cartella dove salvare i risultati")

# Richiesta per il cambio di risoluzione
if (yn=="s" | yn=="S"){
  m <-
```

```
    readline(prompt = "Fattore di aggregazione per i raster di output: ")
} else if (yn=="n" | yn=="N"){m = NULL}

# ---------------------------------- LIBRARIE ----------------------------------

InstallPackages <- function(package_names) {

  # This function install a list of packages on the character vector
  # package_names (es. package_names <- list("raster", "rgdal","rgeos"))

  for(package_name in package_names) {
    if(package_name %in% rownames(installed.packages()) == FALSE) {
      install.packages(package_name)
    }
    library(package_name,character.only=TRUE,quietly=TRUE,verbose=FALSE)
  }
}

packageNames <- list("raster", "rgdal")
InstallPackages(packageNames)


rasterOptions(maxmemory = 90000000)

# ---------------------------------- FUNZIONI ----------------------------------

# Read and merge rasters (with "hole" in the united raster)
readMergeRasters <- function(dir) {

  files <- list.files(dir, full.names = TRUE)
  tiles <- lapply(files, raster)
  tiles$fun <- mean
  do.call(mosaic, tiles)

}

# Function that fills the holes (Na) with the mean in the windows with n size
fillNa <- function(x, i = ceiling(n*n/2)) {

  if (is.na(x)[i] ) {return (mean(x, na.rm=TRUE))
  } else {return (x[i])}
}

# ---------------------------------- PROCESSING ----------------------------------

# Unisco i raster
union <- readMergeRasters(rasFolder)

# Sostituisco eventuali valori NaN
n <- 5
union.NaRemoved <- focal(union,
                         w = matrix(1,n,n),
                         fun = fillNa,
                         pad = TRUE,
                         na.rm = FALSE)

# Se richiesto, cambio la risoluzione
if (!is.null(m)){
  union <- aggregate(union.NaRemoved,
    fact=m,
    fun=mean,
    filename = file.path(pathOut,"rasUnito.tif")
    )
} else if (!is.null(m)) {
  writeRaster(union.NaRemoved,
    filename = file.path(pathOut,"rasUnion.tif"))
}
```

ReprojectRes.R

```
#*******************************************************************************
#
# # R version 3.3.0 (2016-05-03)
# # Platform: x86_64-w64-mingw32/x64 (64-bit)
# # Running under: Windows >= 8 x64 (build 9200)
# #
# # Autore: Tamara Michelini <tamara.michelini@yahoo.it>
# # Data: dicembre 2016
#
#*******************************************************************************
# ============================== **DESCRIZIONE** ================================
#
# ReprojectRes.R
#
# Lo script riproietta i raster contenuti in una cartella da un sistema di riferimento
# (geografico WGS-84 per la Regione Calabria) in un altro impostati dall'utente
# (impostare l'EPSG 32633 per la Regione Calabria). Dopo di ciò, il programma
# cambia la risoluzione dei raster riproiettati secondo un valore intero definito
# sempre dall'utente (impostare 1 m per la Regione Calabria).
# Il processo di ricampionatura avviene secondo la procedura scelta nell'ambito del
# progetto ALForLab. Nello specifico, sono troncati i decimali delle estensioni (extent)
# dei raster ed è forzata la risoluzione esatta all'intero superiore (1 metro per
# per la regione Calabria).
#
# Tutti i dati di input vengono richiesti lanciando il programma (Source).
#
# INPUT:
#   - rasFolder: cartella nella quale si trovano i DTM in formato raster
#   - CRS_from: codice EPSG dei raster di input (solo il numero, 4296 per Regione Calabria)
#   - CRS_to: codice EPSG dei raster di input (solo il numero, 32633 per Regione Calabria)
#   - pathOut: cartella nella quale salvare i risultati
#
# OUTPUT:
#   - file raster riproiettati e ricampionati salvati nella cartella di output
#
#   NOTA: durante l'elaborazione viene creata una cartella temporanea che viene
#   cancellata al termine.
#
# La funzione "InstallPackages" installa una lista di librerie necessarie.
#
# *******************************************************************************
# ------------------------------------ INPUT ------------------------------------

rasFolder <- choose.dir(default = "",
  caption = "Select the folder of rasters")

# Richiesta EPSG dei raster di input
CRS_from <-
      readline(prompt = "Codice EPSG dei raster di input (4296 per raster geografici regione
Calabria): ")

# Richiesta EPSG dei raster di output
CRS_to   <-
   readline(prompt = "Codice EPSG dei raster di output (32633 per raster geografici regione
Calabria): ")

# Cartella di output
pathOut <- choose.dir(caption = "Cartella dove salvare i risultati")

# ---------------------------------- LIBRARIE -----------------------------------

InstallPackages <- function(package_names) {

  # This function install a list of packages on the character vector
  # package_names (es. package_names <- list("raster", "rgdal","rgeos"))

  for(package_name in package_names) {
```

```
    if(package_name %in% rownames(installed.packages()) == FALSE) {
      install.packages(package_name)
    }
    library(package_name,character.only=TRUE,quietly=TRUE,verbose=FALSE)
  }
}

packageNames <- list("gdalUtils","raster",  "rgdal")
InstallPackages(packageNames)

# ---------------------------------- FUNZIONI ----------------------------------

# Riproizeioni da CRSfrom a CRSto
reproject <- function(filesRaster, CRSfrom, CRSto, folderName) {

  ras <- gdalwarp(filesRaster,
                  dstfile = paste0(folderName,"/",
                                   unlist(strsplit(
                                     basename(filesRaster), "[.]"))[1],
                                   "_ProjTmp.tif"),
                  s_srs =  CRSfrom,
                  t_srs = CRSto,
                  output_Raster = T,
                  overwrite = T,
                  verbose = F)

  return(ras@file@name)

}

# Risoluzione ad 1 m
projRes <- function(filesRaster, folderName, CRSfrom, CRSto, folderNameFinal) {

  rasFileProj  <- reproject(filesRaster, CRSfrom, CRSto, folderName)
  rasProj      <- readGDAL(rasFileProj)
  resolution   <- ceiling(res(raster(rasFileProj)))

  gdalwarp(rasFileProj,
           dstfile = paste0(
             folderNameFinal, "/",
             unlist(strsplit(basename(filesRaster), "[.]"))[1],
             "_Project.tif"),
           s_srs = CRSto,
           t_srs = CRSto,
           te = c(trunc(rasProj@bbox[1]),
                  trunc(rasProj@bbox[2]),
                  trunc(rasProj@bbox[3]),
                  trunc(rasProj@bbox[4])),
           tr = resolution,
           output_Raster = T,
           overwrite = T,
           verbose = F)

    }

# ---------------------------------- OUTPUT ------------------------------------

# Create the two output folders removing the final folders if already exsist
folderName      <- file.path(tempdir(), "Proiettati")
folderNameFinal <- file.path(pathOut, "Proiettati")

if (dir.exists(folderName) == T) {
  unlink(folderName, recursive = T)
} else if (dir.exists(folderNameFinal) == T) {
  unlink(folderNameFinal, recursive = T)
  }

dir.create(folderNameFinal)
dir.create(folderName)
```

```
# ----------------------------- RASTER PROCESSING ------------------------------

CRSfrom <- paste0("EPSG:", CRS_from)
CRSto   <- paste0("EPSG:", CRS_to)

# Read the raster in the input folder
filesRaster <- list.files(rasFolder,
                          recursive = F,
                          full.names = T)

# Applico le funzioni per la riproiezione e il cambio di risoluzione ad 1 m e salvo
lapply(1:length(filesRaster), function(i){

  projRes(filesRaster[i],
    folderName,
    CRSfrom,
    CRSto,
    folderNameFinal)

  }
)

# Remove the "temporary" folder
unlink(folderName, recursive = T)
```

Volume.R

```
#*********************************************************************************
#
# # R version 3.3.0 (2016-05-03)
# # Platform: x86_64-w64-mingw32/x64 (64-bit)
# # Running under: Windows >= 8 x64 (build 9200)
# #
# # Autore: Tamara Michelini <tamara.michelini@yahoo.it>
# # Data: dicembre 2016
#
# *********************************************************************************

setwd("~/TAMARA/Script/AlForLab_RShinyApp/data")

t <- 1  # risoluzione del CHM per calcolo Hmean

# *********************************************************************************
## ------------------------------- FUNCTIONS ----------------------------------
# Load my functions
source("./RFunctions/MyFunctions.R")
# *********************************************************************************
## ------------------------------- LIBRARIES ----------------------------------
## Check if packages are installated and eventually installing them
LoadInstallPackages(list("data.table", "raster", "rgdal","rgeos", "openxlsx"))
#
# *********************************************************************************
## ------------------------------- DESCRIPTION --------------------------------
#
# Lo script esegue il calcolo delle mappe raster dei volumi e delle biomasse
# (peso secco fusto e rami grossi, peso secco totale epigeo e peso fresco) medi
# all'ettaro per una definita area sulla base dei valori del CHM e della ripartizione
# delle formazioni forestali, se disponibile, all'interno di una definita area
# definita dall'utente. Se la suddivisione in formazioni forestali non è disponibile
# da parte dell'utente, è impiega una riclassificazione delle "aree boscate"
# del CORINE Land Cover (CLC), Cap. 3.2, al IV livello.
#
# NOTA: la relativa lentezza dello script è dovuta al rispetto di alcune richieste
# fatte:
#
# 1. mantenere il CHM diviso in tiles (non unito)
#    -> ciò implica di dover individuare i tiles che ricoprono l'area di interesse,
#       unirli e rimuovere i valori no-data generati
# 2. effettuare l'estrazione della variabile predittiva Hmean sul CHM con risoluzione
```

```
#    1 metro
# 3. effettuare l'estrazione della variabile predittiva Hmean sulla "maschera" delle
#    formazioni forestali per assicurare che l'altezza media da associare al pixel
#    derivi solo da pixel con uguale classificazione
#    -> generazione di più layer (una per ciascuna formazione forestale)
# 4. effettuare l'estrazione della variabile predittiva Hmean impiegando una finestra
#    mobile di scorrimento (non semplice media per aggregazione di pixel)
# 5. possibilità di caricamento di uno shapefile delle categorie forestali da
#    parte di un utente esterno
#    -> la rasterizzazione (necessaria) è lenta quando fatta in R.
#
# INPUT:
# - SHAPEFILE AREA DI INTERESSE "COMPARTIMENTATA"
# - SHAPEFILE CATEGORIE FORESTALI SECONDO CODIFICA INTERNA (opzionale)
# - LIVELLO DI DETTAGLIO ANALISI
#
# LO SCRIPT NECESSITA DI:
# - CHM tagliato a 2m e corretto
# - SHAPEFILE CLC 2012 RICLASSIFICATA SECONDO CODIFICA INTERNA PER ARFORLAB
#
# OUTPUT:
# - CHM
# - MAPPA RASTER DEI VOLUMI E DELLE BIOMASSE PER L'AREA DI INTERESSE
# - SHAPEFILE VOLUMI E BIOMASSE TOTALI E MEDIE ALL'ETTARO PER I COMPARTI
#   (SCALA COLORI)
#
# ****************************************************************************
## --------------------------------- INPUTS ----------------------------------
## 1. Take the shape of interest area with compartiments
AreaInt_file <-
  choose.files(caption = "SELECT THE SHAPEFILE OF THE INTEREST AREA",
               filters = ".shp")
#
## 2. Take the Forest Typolgies shapefile
FT_yn <-
  readline(prompt = "Do you want to load the shapefile of forest categories? (y/n): ")
#
if (FT_yn == "y"| FT_yn == "Y") {

  ForType_file <-
    choose.files(caption = "SELECT THE SHAPEFILE OF THE FOREST CATEGORY",
                 filters = ".shp")

} else if (FT_yn == "n" | FT_yn == "N") {

  ForType_file <-
    "./shapefile/CLC2012_RiclassBosco/CLC2012_RiclassBosco.shp"
}
#
## 3. Chose the level of datail
liv <-
   readline(prompt = "Chose the detail level of the forest typologies (1 = low, 2 = medium, 3 =
high): ")
#
## 4. Risoluzione dei raster di output (25 o 30?)
m <-
  readline(prompt = "Chose the resolution of the output rasters (suggested: 10, 25, 30 or 50): ")
m <- as.integer(m)
#
# ****************************************************************************
## ------------------------------ MEMORY LIMIT ---------------------------------
memory.limit(size = 100000)
rasterOptions(maxmemory = 90000000)
#
# ****************************************************************************
## ------------------------------ READ FILES -----------------------------------
#
# Read the shapefile of the interest area
shpAreaInt <- readOGR(
  dsn = dirname(AreaInt_file),
```

```
  layer = substr(basename(AreaInt_file), 1, nchar(basename(AreaInt_file)) - 4),
  stringsAsFactors = FALSE)
#
# Read the shapefile of the Forest Type
shpForTyp <- readOGR(dsn = dirname(ForType_file),
                     layer = substr(basename(ForType_file), 1, nchar(basename(ForType_file)) - 4),
                     stringsAsFactors = FALSE)
#
# Create an empty raster with the extention of the interest area (the
# shapefile of the area in this case)
rasEmpty <- brick(raster(
  ncols = round(extent(shpAreaInt)[2] - extent(shpAreaInt)[1])/1,
  nrows = round(extent(shpAreaInt)[3] - extent(shpAreaInt)[4])/1,
  round(extent(shpAreaInt), digit=2),
  resolution = 1,
  crs = crs(shpAreaInt),
  vals = NA),
  nl = 4)
origin(rasEmpty) <- 0
#
shpArea <- gUnionCascaded(shpAreaInt)
mask <- brick(rasterize(shpArea, rasEmpty))
origin(mask) <- 0
# ***************************************************************************
## ---------------------------- FIXED VARIABLES -------------------------------
# 1. Define, on the base of the detail level, the field name to read when the
# rasterization of the Forest Types will be perform.
fieldFT <- NULL

if (liv == 1) {
  fieldFT <- "cod_basso"
} else if (liv == 2) {
  fieldFT <- "cod_medio"
} else if (liv == 3) {
  fieldFT <- "cod_alto"
}
#
# 2. Read or rasterize the Forest Typolgies shapefile for the extention of the interest
# area based on the level of detail selected.

if (FT_yn == "y" | FT_yn == "Y") {

  shpForTyp <- crop(shpForTyp, shpArea)
  # shpForTyp <- crop(shpForTyp, round(extent(shpAreaInt), digit=2))
  rasForTyp_LivX <- brick(rasterize(shpForTyp,
                                    rasEmpty,
                                    field = fieldFT),
                          nl = 4)
  origin(rasForTyp_LivX) <- 0

} else if (FT_yn == "n" | FT_yn == "N") {

  if (liv == 1) {
    rasForTyp_LivX <- brick("./raster/rasCLC2012/rasCLC12_4liv_ZoneBoscate_dettBasso.tif")
  } else if (liv == 2) {
    rasForTyp_LivX <- brick("./raster/rasCLC2012/rasCLC12_4liv_ZoneBoscate_dettMedio.tif")
  } else if (liv == 3) {
    rasForTyp_LivX <- brick("./raster/rasCLC2012/rasCLC12_4liv_ZoneBoscate_dettAlto.tif")
  }

  rasForTyp_LivX <- crop(rasForTyp_LivX, rasEmpty)
  origin(rasForTyp_LivX) <- 0
  rasForTyp_LivX <- brick(rasForTyp_LivX * mask)
}
#
# 3. Folders where the CHM are stored
CHMFolder      <-
  # paste0(dataDir, "/LiDAR/CHM_UniPD10")
  #"C:/Users/tamara.michelini/Desktop/SanFili_NoOUT_CC"
  "./LiDAR/CHM"
```

```
  #"C:/Users/tamara.michelini/Desktop/CHM_UTM33_min2m_1m_Serre_clean_final"
# "C:/Users/tamara.michelini/Documents/TAMARA/DatiStudioAGIF/raster/CHM_UTM33_min2m_1m_clean"
# 4. Folders where the Quadro d'Unione is stored
QU_LiDARFolder <- "./shapefile/QuadriUnione/QU_LiDAR"
# 5. File contenente i coefficienti dei modelli
fileNameModVol <- "./modelli/CoeffModelliVolumi.xlsx"
# 6. Output folder
path_out       <- paste0(dirname(AreaInt_file), "/", "Risultati")
#
if (dir.exists(path_out) == TRUE){unlink(path_out, recursive = TRUE)}
# 7. Dimensione matrice per calcolo di Hmean
n <- 25
# ***************************************************************************
## -------------------------------- PROCESSING --------------------------------
## 1. CHM
# CHM <- CHMIntArea(shpAreaInt, CHMFolder, QU_LiDARFolder)
CHM <- raster("C:/Users/tamara.michelini/Desktop/CHM_ViadottiNAN.tif")

if (t!=res(CHM)){CHM <- aggregate(CHM, fact=t)}

# "Maschero" il CHM sull'area di interesse
CHM <- crop(CHM, rasEmpty)
CHM[is.na(mask)] <- NA
origin(CHM) <- 0
#
# Create a raster with more Leyers, one for each forest typologies, to use as a
# mask for the CHM
rasForTyp_LivX_all <- rasForTyp_LivX

values <- unique(values(rasForTyp_LivX)[, 1])[!is.na(unique(values(rasForTyp_LivX)[, 1]))]
for (i in 1:length(values)) {
  value  <- values[i]
  rasCat <- rasEmpty
  rasCat[rasForTyp_LivX[[1]]==value] <- value
  rasForTyp_LivX <- addLayer(rasForTyp_LivX, rasCat)
  }
#
rasForTyp_LivX <- dropLayer(rasForTyp_LivX, 1)
#
# Moltiplico per 0 e sommo 1 per creare la maschera al CHM
rasForTyp_LivX[!is.na(rasForTyp_LivX)] <- 1
ForTypCHM <- CHM * rasForTyp_LivX                                          # raster multylayers
del CHM per ciascun tipo forestare
#
## 2. HMEAN
Hmean_ForTyp <- rasEmpty
Hmean_ForTyp <-
  lapply(1:(dim(ForTypCHM)[3]),
         function(i)
           focal(ForTypCHM[[i]],
                 w = matrix(1, n, n),
                 fun = "mean",
                 na.rm = TRUE,
                 pad = TRUE)
  )

for (i in 1:dim(ForTypCHM)[3]){
  Hmean_ForTyp[[i]] <- mask(Hmean_ForTyp[[i]], ForTypCHM[[i]])
}
#
# Unisco i raster con la funzione cover che permette di sostituire valori Na nel
# primo raster con i valori del secondo, e così via per ulteriori raster.
# Applico la maschera dell'area di interesse
if (length(Hmean_ForTyp)==1){Hmean_ForTyp <- stack(Hmean_ForTyp)
} else if (length(Hmean_ForTyp)!=1){Hmean_ForTyp <- do.call(cover, Hmean_ForTyp)}
Hmean_ForTyp[is.na(mask)] <- NA
#
# ***************************************************************************
## -------------------------------- OUTCOMES ----------------------------------
#
```

```
# 1. MAPPA RASTER DEI VOLUMI E DELLE BIOMASSE
readCoeffDict <- readModelCoeff(fileNameModVol, as.integer(liv), rasForTyp_LivX_all)
origin(readCoeffDict$coeff) <- 0
#
# Volume per il livello X di dettaglio
VolLivX <- rasEmpty
VolLivX <-
         brick(readCoeffDict$coeff[[1]]    +    readCoeffDict$coeff[[2]]    *    Hmean_ForTyp    ^
readCoeffDict$coeff[[3]])
#
# Metto tutto in un unico raster (Volume e biomasse)
VolLivX <-
  addLayer(VolLivX,
           VolLivX * readCoeffDict$coeff[[4]],                                                    # Peso
secco fusto e rami grossi (Ps1): Ps1 <- VolLivX * r1
           VolLivX * readCoeffDict$coeff[[5]],                                                    # Peso
secco totale epigeo (Ps2): Ps2 <- VolLivX * r2
           VolLivX * readCoeffDict$coeff[[6]])
#
names(VolLivX) <- c("Vol_mean", "Ps1_mean", "Ps2_mean", "Pf_mean")
#
# Cambio la risoluzione dei raster di output
if (m == 1){
  VolBiom_LivX_m <- VolLivX
  CHMm <- CHM
} else if (m!=0){
  VolBiom_LivX_m <- aggregate(VolLivX,
                              fact = m,
                              fun = mean,
                              expand = TRUE,
                              na.rm = TRUE)
  CHMm <- aggregate(CHM,
                    fact = m,
                    fun = mean,
                    expand = TRUE,
                    na.rm = TRUE)

}
#
# # 2. SHAPEFILE DEI VOLUMI E DELLE BIOMASSE TOTALI E AD ETTARO PER FORMAZIONE FORESTALE
# # Calcolo l'area in ettari delle formazioni forestali e aggiungo il campo
# shpForTyp@data$Area_ha <-  round(gArea(shpForTyp, byid = TRUE)/10000, digit = 2)
# #
# # Calcolo i valori medi per compartimento di volume e biomasse e li aggioungo
# shpVBmean_FTha <- extract(VolBiom_LivX_m,
#                           shpForTyp,
#                           fun = mean,
#                           na.rm = TRUE,
#                           sp = TRUE)
# #
# shpVBmean_FTha@data$Vol_tot <-
#   shpVBmean_FTha@data$Area_ha * shpVBmean_FTha@data$Vol_mean
# shpVBmean_FTha@data$Ps1_tot <-
#   shpVBmean_FTha@data$Area_ha * shpVBmean_FTha@data$Ps1_mean
# shpVBmean_FTha@data$Ps2_tot <-
#   shpVBmean_FTha@data$Area_ha * shpVBmean_FTha@data$Ps2_mean
# shpVBmean_FTha@data$Pf_tot <-
#   shpVBmean_FTha@data$Area_ha * shpVBmean_FTha@data$Pf_mean
# #
# # Remove decimals from volume values
# shpVBmean_FTha@data[, (ncol(shpVBmean_FTha@data) - 7):(ncol(shpVBmean_FTha@data))] <-
#    round(shpVBmean_FTha@data[, (ncol(shpVBmean_FTha@data) - 7):(ncol(shpVBmean_FTha@data))], digit
= 0)
# shpVBmean_FTha@data[, (ncol(shpVBmean_FTha@data) - 7):(ncol(shpVBmean_FTha@data))] <-
#      sapply(shpVBmean_FTha@data[,  (ncol(shpVBmean_FTha@data)  -  7):(ncol(shpVBmean_FTha@data))],
as.integer)
# shpVBmean_FTha@data$Area_ha <- round(shpVBmean_FTha@data$Area_ha, digit = 2)
# #
# # 3. SHAPEFILE DEI VOLUMI E DELLE BIOMASSE TOTALI E AD ETTARO PER AREA
# # Calcolo l'area in ettari dei compartimenti e aggiungo il campo
```

```
# shpAreaInt@data$Area_ha <- round(gArea(shpAreaInt, byid = TRUE) / 10000, digit = 2)
# #
# # Calcolo i valori medi per compartimento di volume e biomasse e li aggioungo
# shpVBmean_ha <- extract(VolBiom_LivX_m,
#                         shpAreaInt,
#                         fun = mean,
#                         na.rm = TRUE,
#                         sp = TRUE)
# #
# shpVBmean_ha@data$Vol_tot <-
#   shpVBmean_ha@data$Area_ha * shpVBmean_ha@data$Vol_mean
# shpVBmean_ha@data$Ps1_tot <-
#   shpVBmean_ha@data$Area_ha * shpVBmean_ha@data$Ps1_mean
# shpVBmean_ha@data$Ps2_tot <-
#   shpVBmean_ha@data$Area_ha * shpVBmean_ha@data$Ps2_mean
# shpVBmean_ha@data$Pf_tot  <-
#   shpVBmean_ha@data$Area_ha * shpVBmean_ha@data$Pf_mean
# #
# # Remove decimals from volume values
# shpVBmean_ha@data[, (ncol(shpVBmean_ha@data) - 7):(ncol(shpVBmean_ha@data))] <-
#   round(shpVBmean_ha@data[, (ncol(shpVBmean_ha@data) - 7):(ncol(shpVBmean_ha@data))], digit = 0)
# shpVBmean_ha@data[, (ncol(shpVBmean_ha@data) - 7):(ncol(shpVBmean_ha@data))] <-
#   sapply(shpVBmean_ha@data[, (ncol(shpVBmean_ha@data) - 7):(ncol(shpVBmean_ha@data))], as.integer)
# #
# # 3. SHAPEFILE DEI VOLUMI E DELLE BIOMASSE TOTALI E AD ETTARO PER FORMAZIONE FORESTALE E PER AREA
# # (NB. Lo shapefile dell'area deve contenere un campo "id" e uno "Desc".
# #      Lo shape file viene prodotto solo se almeno uno dei due shape ha piu' di
# #      geometria.)
# shpInt <- disaggregate(intersect(shpAreaInt, shpForTyp))
#
# if (length(shpInt@polygons) > 1){
#
#   shpInt@data <- setnames(shpInt@data, fieldFT, "cod_FT")                          # Sostituisco il
campo della categoria forestale con "cod_FT" per generalizzare
#   shpInt@data <- subset(shpInt@data, select=c("id.1", "id.2", "Desc", "cod_FT"))                 #
Tengo solo ALCUNI CAMPI
#   # Calcolo l'area in ettari dei compartimenti e aggiungo il campo
#   shpInt@data$Area_ha <- round(gArea(shpInt, byid = TRUE) / 10000, digit = 2)
#   # Calcolo i valori medi per per categoria forestale e compartimento e li aggiungo
#   shpInt_VB <- extract(VolBiom_LivX_m,
#                        shpInt,
#                        fun = mean,
#                        na.rm = TRUE,
#                        sp = TRUE)
#   #
#   # Calcolo i valori totali per poligono di volume e biomasse
#   shpInt_VB@data$Vol_tot  <-
#     round((shpInt_VB@data$Area_ha * shpInt_VB@data$Vol_mean), digit = 0)
#   shpInt_VB@data$Vol_mean <-
#     round(shpInt_VB@data$Vol_mean, digit = 0)
#   shpInt_VB@data$Ps1_tot  <-
#     round((shpInt_VB@data$Area_ha * shpInt_VB@data$Ps1_mean), digit = 0)
#   shpInt_VB@data$Ps1_mean <-
#     round(shpInt_VB@data$Ps1_mean, digit = 0)
#   shpInt_VB@data$Ps2_tot  <-
#     round((shpInt_VB@data$Area_ha * shpInt_VB@data$Ps2_mean), digit = 0)
#   shpInt_VB@data$Ps2_mean <-
#     round(shpInt_VB@data$Ps2_mean, digit = 0)
#   shpInt_VB@data$Pf_tot   <-
#     round((shpInt_VB@data$Area_ha * shpInt_VB@data$Pf_mean), digit = 0)
#   shpInt_VB@data$Pf_mean  <-
#     round(shpInt_VB@data$Pf_mean, digit = 0)
#
#   shpInt_VB@data[, (ncol(shpInt_VB@data) - 7):(ncol(shpInt_VB@data))] <-
#     sapply(shpInt_VB@data[, (ncol(shpInt_VB@data) - 7):(ncol(shpInt_VB@data))], as.integer)
#   #
#   # Calcolo la percentuale di ciascuna formazione forestale all'interno della "sotto" area
#   # sumtp <- setDT(shpInt_VB@data)[, lapply(.SD, sum), by = .(id.1)]
#   sumtp <- setDT(shpInt_VB@data)[, lapply(.SD[, sapply(.SD, is.numeric), with = FALSE], sum), by =
.(id.1)]
```

```
#
#   for (i in 1:dim(shpInt_VB@data)[1]){
#     for (j in 1:dim(sumtp)[1]){
#       if (shpInt_VB@data$id.1[i] == sumtp$id.1[j]){
#         shpInt_VB@data$perc_A[i] <-
#           round((shpInt_VB@data$Area_ha [i]/sumtp$Area_ha[j]*100), digit = 2)
#         shpInt_VB@data$perc_V[i] <-
#           round((shpInt_VB@data$Vol_tot [i]/sumtp$Vol_tot[j]*100), digit = 2)
#       }
#     }
#   }
#   # Remove decimals from volume values and round to 2 decimals Area in hectars
#   shpInt_VB@data$Area_ha <- round(shpInt_VB@data$Area_ha, digit = 2)
#   val <- readCoeffDict$val[!is.na(readCoeffDict$val)]
#   key  <- readCoeffDict$key[!is.na(readCoeffDict$key)]
#
#   for (i in 1:length(shpInt_VB@data$cod_FT)){
#     for (j in 1: length(val)){
#       if (shpInt_VB@data$cod_FT[i] == val[j]){
#         shpInt_VB@data$FormaFor[i] <- key[j]
#       }
#     }
#   }
# } else {shpInt_VB <- as.null(shpInt)}
#
## ------------------------------ WRITE OUTPUTS -------------------------------
dir.create(path = path_out)
# Raster
lapply(1:dim(VolBiom_LivX_m)[3], function (i)
  writeRaster(VolBiom_LivX_m[[i]],
              filename = paste0(path_out,
                                "/",
                                names(VolBiom_LivX_m)[i],
                                "_ha.tif"),
              format = "GTiff",
              overwrite = TRUE))
writeRaster(CHMm,
            filename = paste0(path_out, "/CHM_", m, "m.tif"),
            format = "GTiff",
            overwrite = TRUE)
#
# # Shapefile Area
# shapefile(shpVBmean_ha,
#           filename = paste0(path_out, "/shp_VolBio.shp"),
#           overwrite = TRUE)
# #
# # Shapefile Formazioni forestali
# shapefile(shpVBmean_FTha,
#           filename = paste0(path_out, "/shp_VolBio_FT.shp"),
#           overwrite = TRUE)
# #
# # Shapefile Formazioni forestali per area
# if (!is.null(shpInt_VB)){
#   shapefile(shpInt_VB,
#             filename = paste0(path_out, "/shp_VolBio_Int.shp"),
#             overwrite = TRUE)
# } else {shpInt_VB <- subset(shpVBmean_ha, select="id")}
# #
# # Excel ----------------------------------------------------------------------
# # Titles and sheets
# sheets <- c("FormazioniForestali", "Compartimenti", "Comp_FormazForestali")
# title <- c(
#   "Risultati dei modelli di volume e di biomasse per formazioni forestali",
#   "Risultati dei modelli di volume e di biomasse per compartimenti",
#    "Risultati dei modelli di volume e di biomasse per compartimenti suddivisi per formazioni
forestali")
# #
# # Data
# FTData   <- shpVBmean_FTha@data
# CompData <- shpVBmean_ha@data
```

```
# IntData  <- shpInt_VB@data
#
# if (!is.null(shpInt_VB)) {
#   # Medio i valori di ciascuna formazione forestale all'interno della "sotto" area
#   IntData_mean <-
#     subset(IntData,
#            select = c(id.1, id.2, cod_FT, Vol_mean, Ps1_mean, Ps2_mean, Pf_mean)
#            )
#   IntData_mean <-
#     round(IntData_mean[, lapply(.SD, mean), by = .(id.1, cod_FT)], digits = 0)
#   IntData_sum <-subset(IntData,
#                        select = c(id.1, id.2, cod_FT, Area_ha, Vol_tot, Ps1_tot,
#                                   Ps2_tot, Pf_tot, perc_A, perc_V))
#   IntData_sum <- IntData_sum[, lapply(.SD, sum), by = .(id.1, cod_FT)]
#   IntData <-
#     data.table(IntData_mean, subset(IntData_sum, select = -c(id.1, id.2, cod_FT)))
#   #
#   for (i in 1:length(IntData$cod_FT)) {
#     for (j in 1:length(val)) {
#       if (IntData$cod_FT[i]==val[j]) {
#         IntData$Formazioni_Forestali[i] <- key[j]
#       } else {}
#     }
#   }
# }
# #
# # Styles
# titleStyle  <- createStyle(fontSize = 14,
#                            fontName = "Calibri",
#                            textDecoration = c("bold", "italic", "underline"))
# headerStyle <- createStyle(fontSize = 11,
#                            halign = "center",
#                            textDecoration = "italic")
# dataStyle  <- createStyle(fontSize = 10,
#                           halign = "center")
# #
# # Create a new workbook for outputs
# wb <- createWorkbook()
# #
# # Add data
# # Create an array to call the data in the loop
# if (!is.null(shpInt_VB)) {
#   data <- c("FTData", "CompData", "IntData")
# } else {data <- c("FTData", "CompData")}
# # Add sheets
# lapply(1:length(data), function(i) {
#   addWorksheet(wb, sheetName = sheets[i])
#   })
# # Add title
# lapply(1:length(data), function(i) {
#   writeData(wb,
#             sheets[i],
#             title[i],
#             startCol = 1,
#             startRow = 1)})
# #
# lapply(1:length(data), function(i)
#   writeData(wb,
#             sheets[i],
#             eval(parse(text=data[i])),
#             startCol = 1,
#             startRow = 3))
# #
# # Add styles to the column headers
# for (i in 1:length(data)){
#   addStyle(wb, sheets[i], titleStyle, rows=1, cols=1)
#   addStyle(wb, sheets[i], headerStyle, rows = 3, cols = 1:100, gridExpand = TRUE)
#   addStyle(wb, sheets[i], dataStyle, rows = 4:2000, cols = 1:100, gridExpand = TRUE)}
# #
# # Save the file Excel
```

```
# saveWorkbook(wb,
#              paste0(path_out, "/ModelVolume_Risultati.xlsx"),
#              overwrite = TRUE)
# #
# # remove(mask)
# # remove(idx)
# # remove(rasCat)
# # remove(rasForTyp_LivX)
# # remove(rasEmpty)
# # remove(VolLivX)
# # remove(rasForTyp_LivX_all)
```