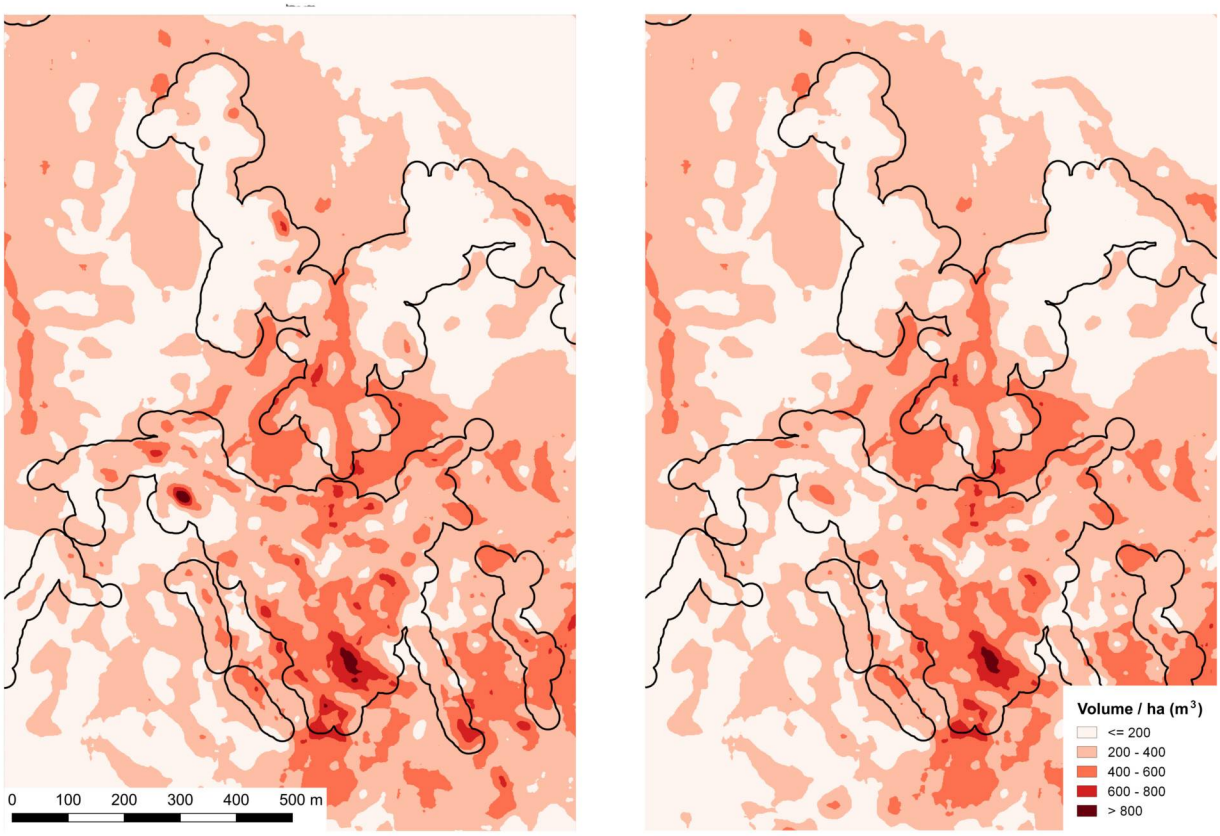
Floris A, Penasa A, Michelini T, Puletti N (2017). **Individuazione e correzione di *outlier* nei dati *raster* LiDAR provenienti dal Geoportale Nazionale e produzione di un CHM idoneo alla stima delle provvigioni legnose della Calabria.**
Forest@ - Rivista di Selvicoltura ed Ecologia Forestale – doi: 10.3832/efor2328-014

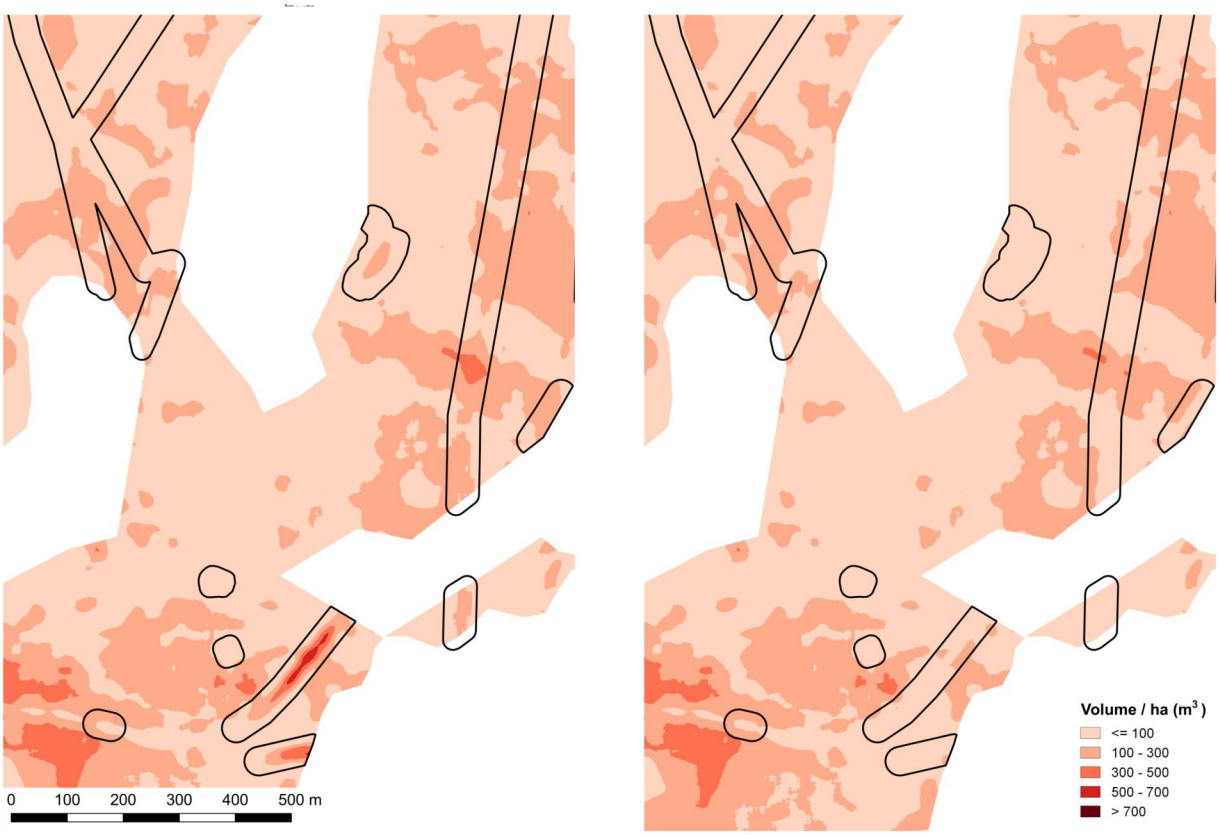
Materiale Supplementare

Fig. S1 - Estratto della mappa del volume stimato per il comune di Orsomarso: (A) impiegando il nDSM non corretto e (B) impiegando il CHM corretto.



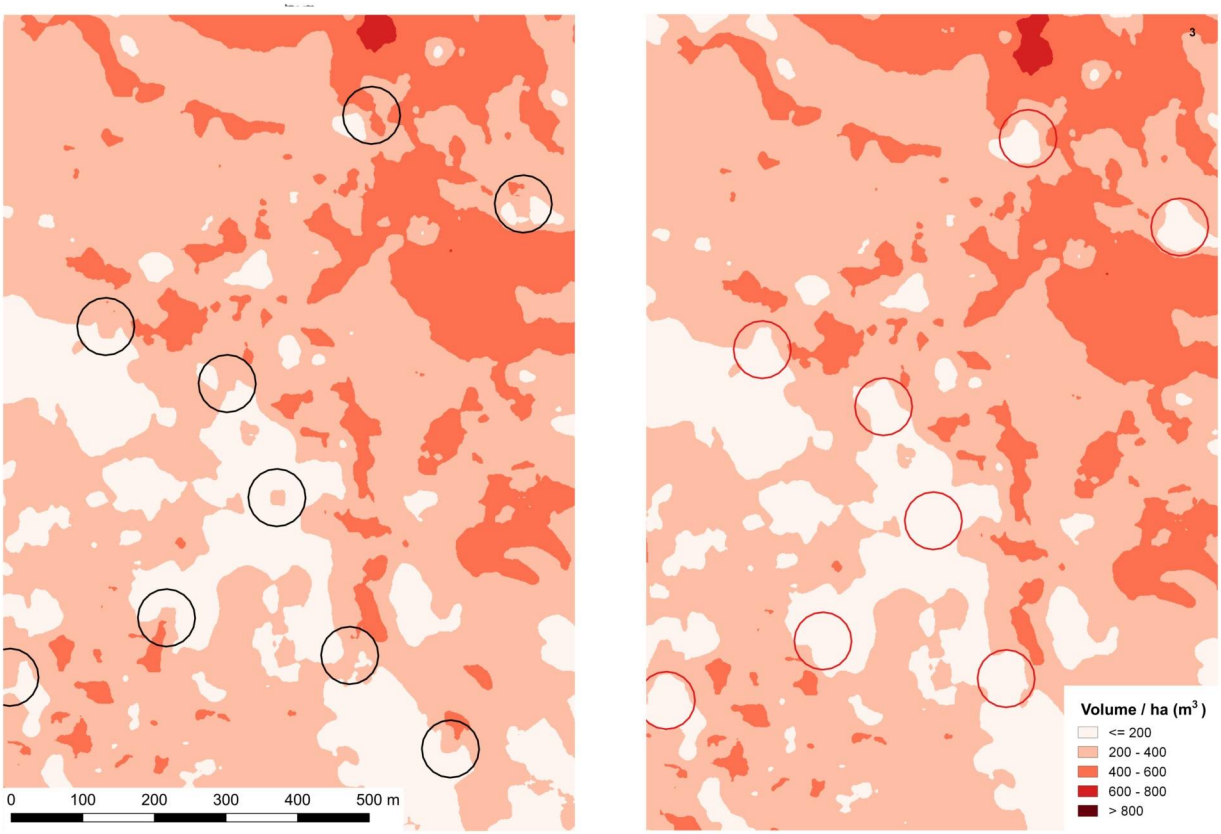
(A) (B)

Fig. S2 - Estratto della mappa del volume stimato per il comune di San Fili: (A) impiegando il nDSM non corretto e (B) impiegando il CHM corretto.



(A) (B)

Fig. S3 - Estratto della mappa del volume stimato per il comune di San Sostene: (A) impiegando il nDSM non corretto e (B) impiegando il CHM corretto.



(A) (B)

Box S1 - Esempi di procedure automatizzate in ambiente *open-source* R ("Affioramenti.R", "MosaicRaster.R", "ReprojectRes1m.R").

`Affioramenti.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** ================================
#
# Affioramenti.R
#
# Lo script, nelle aree dove la presenza di pareti sub-verticali (pendenza > 70°)
# da luogo a valori aberranti del nDSM rispetto al loro intorno, sostituisce questi
# valori assegnando la mediana calcolata sulle celle limitrofe, calcolata con
# matrice di esplorazione di 25 x 25 m.
#
# [Floris, A, Penasa, A, Michelini, T, Puletti, N (in stampa). Pre-trattamento
# dei dati raster LiDAR provenienti dal Geoportale Nazionale impiegati per la
# stima dei volumi e delle fitomasse forestali della Calabria. Forest@,
# collection AlForLab.]
#
# Tutti i dati di input vengono richiesti lanciando il programma (Source).
#
# INPUT:
#    - CHMFolder: cartella nella quale sono salvati i CHM
#    - DTMFolder: cartella nella quale sono salvati i DTM
#    - thrSlp: soglia di pendenza oltre la quale eseguire la sostituzione dei pixel
#
# OUTPUT:
#    - file raster del CHM: salvato nella cartella scelta come output
#
# *******************************************************************************
#------------------------------------ INPUT -------------------------------------
#
# CHM Folder
CHMFolder <- choose.dir(default = "",
                        caption = "Cartella nella quale sono salvati i CHM")
# DTM Folder
DTMFolder <- choose.dir(default = "",
                        caption = "Cartella nella quale sono salvati i DTM")

# The minimum height of the CHM
thrSlp   <- as.integer(readline(prompt="Soglia di pendenza oltre la quale eseguire la sostituzione
dei pixel in gradi: "))

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

# CHMmedianFile <- "C:/Users/tamara.michelini/Documents/TAMARA/Affioramenti/Intero/CHMmedian.tif"
# AffBuffFile   <- "C:/Users/tamara.michelini/Documents/TAMARA/Affioramenti/Intero/MaskSlope___.tif"

# ---------------------------------- 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 ----------------------------------
# Function that fills the holes (Na) with the mean in the windows with n size

# Function that fills the holes (Na) with the mean in the windows with n size
k <- 25
fillNa <- function(x, i = ceiling(k*k/2)) {

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


CHMcleanAff <- function(CHMfile, DTMfile, thrSlp){

  rasCHM <- raster(CHMfile)
  rasDTM <- raster(DTMfile)

  m <- 5
  rasDTMMean <- focal(rasDTM,
                      w = matrix(1, m, m),
                      fun = mean,
                      pad = TRUE,
                      na.rm = TRUE)
  n <- 25
  rasCHMMedian <- focal(rasCHM,
                        w = matrix(1, n, n),
                        fun = median,
                        pad = TRUE,
                        na.rm = TRUE)

  rasSLP <- terrain(rasDTMMean, opt='slope', unit='degrees')

  rasAff <- rasSLP >= 70

  n <- 21
  rasAffBuff <- focal(rasAff,
                      w = matrix(1, n, n),
                      fun = max,
                      pad = TRUE,
                      na.rm = TRUE)

  diff <- rasCHM - rasCHMMedian

  # Maschere
  Mask <- rasAffBuff == 1 & diff >= 10

  if(all(is.na(values(Mask))) == TRUE) {
    writeRaster(rasCHM,
                filename = paste0(file.path(pathOut, names(rasCHM)), ".tif"),
                overwrite = TRUE)

  } else if (all(is.na(values(Mask))) == FALSE) {

    CHMfinal <- rasCHM
    CHMmask  <- rasCHM

    CHMmask[Mask==1] <- NA

    n <- 25
    CHMmedAff <- focal(CHMmask,
                       w = matrix(1,n,n),
                       fun = fillNa,
                       pad=TRUE,
                       na.rm = FALSE)

    CHMfinal[Mask==1] <- CHMmedAff[Mask==1]
```

```
    # Salvataggio
    writeRaster(CHMfinal,
                filename = paste0(file.path(pathOut, names(rasCHM)), ".tif"),
                overwrite = TRUE)
  }
}

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

# Lista dei CHM da processare
CHMfiles <- list.files(CHMFolder, full.names = TRUE, pattern = "tif$")

# Lista dei DTM da processare
DTMfiles <- list.files(DTMFolder, full.names = TRUE, pattern = "tif$")

# Esecuzione della funzione di pulizia per tutti i tiles
lapply(1:length(CHMfiles), function(i){
  CHMcleanAff(CHMfiles[i], DTMfiles[i])})

#
#
# # CHM       <- raster(CHMFile)
# # CHMmedian <- raster(CHMmedianFile)
# # AffBuff   <- raster(AffBuffFile)
# # diff <- CHM - CHMmedian
#
# # Maschere
# Mask <- AffBuff$MaskSlope___==1 & diff>=10
# CHMfinal <- CHM
# CHMmask  <- CHM
# CHMmask[Mask==1] <- NA
#
# n <- 25
# CHMmedAff <- focal(CHMmask,
#                    w = matrix(1,n,n),
#                    fun = fillNa,
#                    pad=T,
#                    na.rm = FALSE)
#
# CHMfinal[Mask==1] <- CHMmedAff[Mask==1]
#
# # Salvataggio
# writeRaster(CHMfinal,
#             filename =
"C:/Users/tamara.michelini/Documents/TAMARA/Affioramenti/Intero/CHMfinal.tif",
#             overwrite = T)
```

MosaicRaster.R

```
rasterOptions(maxmemory = 90000000)
# ************************************************************************
## ------------------------------ LOAD FUNCTION -------------------------------
source("C:/Users/tamara.michelini/Documents/TAMARA/Script/RFunctions/LoadInstallPackages.R")
# ************************************************************************
## ---------------------------------- LIBRARIES -----------------------------------
LoadInstallPackages(list("raster", "rgdal"))

# ************************************************************************
## ------------------------------ DESCRIPTION -------------------------------

# This script combines a list of raster in a single raster filling any holes
# (NA values) that are created by the union. The filling is done by assigning
# the average value of the neighboring pixels in a window of size n x n.

# ************************************************************************
## ----------------------------------- INPUT ------------------------------------

# Select the folders where the rasters, which must be united, are saved

rasterFolder <- choose.dir(default = "",
                           caption = "Select the folder where raster are saved."
                           )
# FT_yn <-
#   readline(prompt = "Do you want to change raster resolution? (y/n): ")
# #
# if (FT_yn == "y"| FT_yn == "Y") {
#
#   m <-
#     readline(prompt = "Chose the resolution of the output rasters (suggested: 10, 25, 30 or 50):
")
#   m <- as.integer(m)
#
# } else if (FT_yn == "n" | FT_yn == "N") {
#
#   m <- NULL
# }

# ************************************************************************
## ------------------------------ FUNCTIONS -------------------------------

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

union <- readMergeRasters(rasterFolder)
n <- 5
union.NaRemoved <- focal(union,
                         w = matrix(1,n,n),
                         fun = fillNa,
                         pad = TRUE,
                         na.rm = FALSE,
                         filename = paste0(rasterFolder,"/Union.tif")
                         )
```

ReprojectRes1m.R

```
# ********************************************************************************
## ------------------------------ LOAD FUNCTION -------------------------------
source("C:/Users/tamara.michelini/Documents/TAMARA/Script/RFunctions/LoadInstallPackages.R")
# ********************************************************************************
## --------------------------------- **LIBRARIES ------------------------------
LoadInstallPackages(list("gdalUtils", "rgdal"))

# ********************************************************************************
## ------------------------------ DESCRIPTION -------------------------------

# This script performs the reprojection of a list of raster from an input
# reference system (in this case EPSG 4326) to another (in this case EPSG 32633)
# by assigning a specific resolution of the output raster (in this case 1 m).
# Following the previously developed method, the change of resolution to 1 m,
# necessary because reprojection produces lower values of the cells (about 0.978 m),
# is performed by deleting decimals in the extension of rasters and setting the
# resolution of 1 m.

# ********************************************************************************
## ------------------------------ FUNCTIONS -------------------------------

# Riproizeioni da CRSfrom a CRSto

reproject <- function(filesRaster, CRSfrom, CRSto, folderName) {

  ras <- gdalwarp(filesRaster,
                  dstfile = paste0(folderName, "/",
                                   unlist(strsplit(
                                     basename(filesRaster), "[.]"
                                   ))[1],
                                   "_UTM33.tif",
                                   sep = ""),
                  s_srs = CRSfrom,
                  t_srs = CRSto,
                  output_Raster = T,
                  overwrite = T,
                  verbose = F
  )
  return(ras@file@name)
}

# Risoluzione ad 1 m
projRes1m <- function(filesRaster, folderName, res, CRS_to, folderNameFinal) {

  rasUTM33  <- reproject(filesRaster, CRS_from, CRS_to, folderName)
  rasUTM33_ <- readGDAL(rasUTM33)

  gdalwarp(rasUTM33,
           dstfile = paste0(
             folderNameFinal,
             "/",
             unlist(strsplit(basename(filesRaster), "[.]"))[1],
             "_UTM33_1m.tif",
             sep = ""),
           s_srs = CRS_to,
           t_srs = CRS_to,
           te = c(trunc(rasUTM33_@bbox[1]),
                  trunc(rasUTM33_@bbox[2]),
                  trunc(rasUTM33_@bbox[3]),
                  trunc(rasUTM33_@bbox[4])),
           tr = c(res, res),
           output_Raster = TRUE,
           overwrite = TRUE,
           verbose = FALSE)
  }
```

```
# ************************************************************************
## ---------------------------------- INPUT -----------------------------------

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

CRS_from <- "+proj=longlat +datum=WGS84 +no_defs"
CRS_to   <- "+proj=utm +zone=33 +datum=WGS84"
resFinal <- 1

# ************************************************************************
## ---------------------------------- OUTPUT ----------------------------------
# Create the two output folders removing the final folders if already exsist

folderName      <- file.path(rasterFolder, "UTM33", fsep = "/")
folderNameFinal <- file.path(rasterFolder, "UTM33_1m", fsep = "/")

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

dir.create(folderNameFinal)
dir.create(folderName)
# ************************************************************************
## ---------------------------- RASTER PROCESSING -----------------------------

# Read the raster in the input folder
filesRaster <- list.files(rasterFolder,
                          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){

  projRes1m(
    filesRaster[i],
    folderName,
    resFinal,
    CRS_to,
    folderNameFinal)

  }
)

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