```
#########################################################################################################
## Analyze DEM, Land Cover, Road accessibility, Water + aux to identify zones suitable for camp
## remi.dannunzio@fao.org & rashed.jalal@fao.org
## 2018/02/05
#########################################################################################################
#########################################################################################################
options(stringsAsFactors=FALSE)

library(Hmisc)
library(sp)
library(rgdal)
library(raster)
library(plyr)
library(foreign)
library(rgeos)
library(glcm)

########################## Work directory
setwd("/media/dannunzio/OSDisk/Users/dannunzio/Documents/countries/bangladesh/rohingya_crisis/data/")

#########################################################################################################
#########################                                    GENERATE SEGMENTS
#########################################################################################################

#################### Load Landsat_8 mosaic from SEPAL
system("scp -P 443 dannunzio@sepal.io:/home/dannunzio/downloads/mosaic-2018-02-06-0917/mosaic* .")

#################### PERFORM SEGMENTATION USING THE OTB-SEG ALGORITHM
params <- c(3,   # radius of smoothing (pixels)
            16,  # radius of proximity (pixels)
            0.1, # radiance threshold
            50,  # iterations of algorithm
            10)  # segment minimum size (pixels)

system(sprintf("otbcli_MeanShiftSmoothing -in %s -fout %s -foutpos %s -spatialr %s -ranger %s -thres
               "mosaic-2018-02-06-0917.vrt",
               #paste0(seg_dir,"tile_1000.tif"),
               paste0("tmp_smooth_",paste0(params,collapse = "_"),".tif"),
               paste0("tmp_position_",paste0(params,collapse = "_"),".tif"),
               params[1],
               params[2],
               params[3],
               params[4]
))

system(sprintf("otbcli_LSMSSegmentation -in %s -inpos %s -out %s -spatialr %s -ranger %s -minsize 0
               paste0("tmp_smooth_",paste0(params,collapse = "_"),".tif"),
               paste0("tmp_position_",paste0(params,collapse = "_"),".tif"),
               paste0("tmp_seg_lsms_",paste0(params,collapse = "_"),".tif"),
               params[1],
               params[2],
               "."
))

system(sprintf("otbcli_LSMSSmallRegionsMerging -in %s -inseg %s -out %s -minsize %s -tilesizex 512 -
               paste0("tmp_smooth_",paste0(params,collapse = "_"),".tif"),
               paste0("tmp_seg_lsms_",paste0(params,collapse = "_"),".tif"),
               paste0("seg_lsms_",paste0(params,collapse = "_"),".tif"),
               params[5]
))

######################### REPROJECT IN BTM
system(sprintf("gdalwarp -t_srs \"%s\" -co COMPRESS=LZW %s %s",
               proj,
               paste0("seg_lsms_",paste0(params,collapse = "_"),".tif"),
               paste0("seg_lsms_btm",paste0(params,collapse = "_"),".tif")
))

######################### POLYGONIZE
```

```
system(sprintf("gdal_polygonize.py -f \"ESRI Shapefile\" %s %s",
               paste0("seg_lsms_btm",paste0(params,collapse = "_"),".tif"),
               paste0("seg_lsms_btm",paste0(params,collapse = "_"),".shp")
))

##############################################################################################################
##########################                                    LAND COVER MAP
##############################################################################################################

########################## READ LC MAP in vector format
zone_nme <- "lc_map_2015/_Coxs_Bazar.shp"
zone <- readOGR(zone_nme)
proj <- proj4string(zone)

df <- zone@data

head(zone)
attr_nme <- "CODE_R"

########################## Generate a numeric legend for the attribute
legend   <- levels(as.factor(zone@data[,attr_nme]))
legend   <- data.frame(cbind(legend,1:length(legend)))
names(legend) <- c(attr_nme,"zonal_code")
zone@data$sort_id <- row(zone@data)[,1]
zone@data <- arrange(merge(zone@data,legend,all.x=T),sort_id)[,c(attr_nme,"zonal_code")]

legend2 <- merge(legend,unique(df[,c("CODE_R","CLASS_R")]),all.x=T,by.x="CODE_R",by.y="CODE_R")
write.csv(legend2,"legend.csv",row.names = F)

########################## RASTERIZE THE MAP
writeOGR(zone,paste0("to_rasterize.shp"),"to_rasterize","ESRI Shapefile",overwrite_layer = T)

system(sprintf("gdal_rasterize -a %s -l %s -co COMPRESS=LZW -te %s %s %s %s -tr %s %s %s %s",
               "zonal_code",
               "to_rasterize",
               679608.323361,
               2272179.10381,
               759366.767431,
               2431316.18984,
               5.997068916830281,
               5.997068916830281,
               "to_rasterize.shp",
               "tmp_lcmap.tif"
               ))

system(sprintf("gdal_translate -ot byte -co COMPRESS=LZW %s %s",
               "tmp_lcmap.tif",
               "cb_lcmap.tif"
))

########################## IN GEO PROJECTION
########################## ALIGN WITH SEGMENTATION PRODUCT
system(sprintf("oft-clip.pl %s %s %s",
               paste0("segments/seg_lsms_",paste0(params,collapse = "_"),".tif"),
               "lc_map_2015/cb_lcmap.tif",
               "lc_map_2015/cb_lcmap_geo30.tif"))


########################## COMPUTE LAND COVER HISTOGRAM FOR EACH POLYGON
system(sprintf("oft-his -um %s -i %s -o %s -maxval 22",
               paste0("segments/seg_lsms_btm",paste0(params,collapse = "_"),".tif"),
               "lc_map_2015/cb_lcmap_btm30.tif",
               "his_lcmap.txt"))

df <- read.table("his_lcmap.txt")
names(df) <- c("poly_id","total",paste0("class",0:22))

########################## COMPUTE MAJORITY CLASS
df$mode_lcmap <- c(0:22)[max.col(df[,paste0("class",0:22)])]
```

```
table(df$mode_lcmap)

df1 <- merge(df,legend2,by.x="mode_lcmap",by.y="zonal_code",all.x=T)
df1[is.na(df1)] <- "nodata"
head(df1)

######################### MERGE MAJORITY CLASS INTO DBF
dbf <- read.dbf("segments/seg_lsms_3_16_0.1_50_10.dbf")

write.dbf(dbf,"segments/bckup_seg_lsms_3_16_0.1_50_10.dbf")
dbf$unique <- row(dbf)[,1]
head(dbf)

df2 <- merge(dbf,df1,by.x="DN",by.y="poly_id",all.x=T)
df3 <- arrange(df2,unique)
head(df3)

write.dbf(df3[,c("DN","total","CODE_R","CLASS_R")],"segments/seg_lsms_3_16_0.1_50_10.dbf")

######################### IN BTM PROJECTION
######################### ALIGN WITH SEGMENTATION PRODUCT
system(sprintf("oft-clip.pl %s %s %s",
               paste0("seg_lsms_btm",paste0(params,collapse = "_"),".tif"),
               "cb_lcmap.tif",
               "cb_lcmap_btm30.tif"))


######################### COMPUTE LAND COVER HISTOGRAM FOR EACH POLYGON
system(sprintf("oft-his -um %s -i %s -o %s -maxval 22",
               paste0("seg_lsms_btm",paste0(params,collapse = "_"),".tif"),
               "cb_lcmap_btm30.tif",
               "his_lcmap.txt"))

df <- read.table("his_lcmap.txt")
names(df) <- c("poly_id","total",paste0("class",0:22))

######################### COMPUTE MAJORITY CLASS
df$mode_lcmap <- c(0:22)[max.col(df[,paste0("class",0:22)])]
table(df$mode_lcmap)

df1 <- merge(df,legend2,by.x="mode_lcmap",by.y="zonal_code",all.x=T)
df1[is.na(df1)] <- "nodata"
head(df1)

######################### MERGE MAJORITY CLASS INTO DBF
dbf <- read.dbf("seg_lsms_btm3_16_0.1_50_10.dbf")

write.dbf(dbf,"bckup_seg_lsms_btm3_16_0.1_50_10.dbf")
dbf$unique <- row(dbf)[,1]
head(dbf)

df2 <- merge(dbf,df1,by.x="DN",by.y="poly_id",all.x=T)
df3 <- arrange(df2,unique)
head(df3)

write.dbf(df3[,c("DN","total","CODE_R","CLASS_R")],"seg_lsms_btm3_16_0.1_50_10.dbf")

##############################################################################################
#########################                                DISTANCE TO ROADS, RIVERS AND ELEPHANT HUMAN CON
##############################################################################################

#############################################################################
#######          ROADS : GENERATE NUMERICAL CODE, RASTERIZE AND COMPUTE DISTANCE MAP
dbf <- read.dbf("road_hec/osm_main_road.dbf")
write.dbf(dbf,"road_hec/osm_main_road_bckup.dbf")
names(dbf)
dbf$unique <- row(dbf)[,1]

legend   <- levels(as.factor(dbf[,"highway"]))
```

```
legend   <- data.frame(cbind(legend,1:length(legend)))
names(legend) <- c("highway","code")

dbf2 <- arrange(merge(dbf,legend),unique)
write.dbf(dbf2,"road_hec/osm_main_road.dbf")

system(sprintf("oft-rasterize_attr.py -v %s -i %s -o %s -a %s",
               "road_hec/osm_main_road.shp",
               paste0("seg_lsms_btm",paste0(params,collapse = "_"),".tif"),
               "road_hec/roads.tif",
               "code"
))

system(sprintf("gdal_proximity.py -co COMPRESS=LZW  -ot Int16 %s %s",
               "road_hec/roads.tif",
               "road_hec/dist_roads.tif"))

#############################################################################
#######          ELEPHANT HUMAN CONFLICT : RASTERIZE AND COMPUTE DISTANCE MAP
system(sprintf("oft-rasterize_attr.py -v %s -i %s -o %s -a %s",
               "road_hec/All_HEC_Line_BUTM2010.shp",
               paste0("seg_lsms_btm",paste0(params,collapse = "_"),".tif"),
               "road_hec/hec.tif",
               "TRACKNUM"
))

system(sprintf("gdal_proximity.py -co COMPRESS=LZW  -ot Int16 %s %s",
               "road_hec/hec.tif",
               "road_hec/dist_hec.tif"))

#############################################################################
#######          RIVERS : SELECT WATER FROM LC MAP
legend2

system(sprintf("gdal_calc.py -A %s --co=\"COMPRESS=LZW\" --outfile=%s --calc=\"%s\"",
               "cb_lcmap_geo30.tif",
               "road_hec/rivers.tif",
               "(A==19)+(A==18)+(A==14)+(A==12)"))

system(sprintf("gdal_proximity.py -co COMPRESS=LZW  -ot Int16 %s %s",
               "road_hec/rivers.tif",
               "road_hec/dist_riv.tif"))

#############################################################################################
#########################                                PROTECTED AREAS AND FLOODING ZONE
#############################################################################################
system(sprintf("oft-clip.pl %s %s %s",
               paste0("seg_lsms_btm",paste0(params,collapse = "_"),".tif"),
               "9_PA/PA_CXB.tif",
               "9_PA/pa_30m.tif"))

system(sprintf("oft-clip.pl %s %s %s",
               paste0("seg_lsms_btm",paste0(params,collapse = "_"),".tif"),
               "extent_90E_30N.tif",
               "9_PA/flood.tif"))

#############################################################################################
#########################                                DEM
#############################################################################################

#############################################################################
#######          Merge HGT tiles
system(sprintf("gdal_merge.py -v -o %s %s",
               paste0("tmp_dem.tif"),
               paste0("*.hgt")
))

#############################################################################
#######          Project in TM
```

```
system(sprintf("gdalwarp -t_srs \"%s\" -co COMPRESS=LZW %s %s",
               proj,
               paste0("tmp_dem.tif"),
               paste0("tmp_dem_utm.tif")
))

################################################################################
#######          Compute slope
system(sprintf("gdaldem slope -co COMPRESS=LZW %s %s",
               paste0("tmp_dem_utm.tif"),
               paste0("tmp_slope.tif")
))

################################################################################
#######          Compute aspect
system(sprintf("gdaldem aspect -co COMPRESS=LZW %s %s",
               paste0("tmp_dem_utm.tif"),
               paste0("tmp_aspect.tif")
))

dem_input <- paste0("DEM/srtm_elev_30m_aoi.tif")
slp_input <- paste0("DEM/srtm_slope_30m_aoi.tif")
asp_input <- paste0("DEM/srtm_aspect_30m_aoi.tif")

################################################################################
#######          Clip three DEM products
system(sprintf("gdal_translate -projwin %s %s %s %s -co COMPRESS=LZW  %s %s",
               679608.323361,
               2431316.18984,
               759366.767431,
               2272179.10381,
               paste0("tmp_dem_utm.tif"),
               dem_input
))

system(sprintf("gdal_translate -projwin %s %s %s %s -co COMPRESS=LZW  %s %s",
               679608.323361,
               2431316.18984,
               759366.767431,
               2272179.10381,
               paste0("tmp_slope.tif"),
               slp_input
))

system(sprintf("gdal_translate -projwin %s %s %s %s -co COMPRESS=LZW  %s %s",
               679608.323361,
               2431316.18984,
               759366.767431,
               2272179.10381,
               paste0("tmp_aspect.tif"),
               asp_input
))

################################################################################
#######          Compute texture for slope and DEM
glcm <- glcm(raster(dem_input),statistics= "dissimilarity",scale_factor=10,asinteger=TRUE)
writeRaster(glcm,"tmp_glcm_diss10_dem.tif")

system(sprintf("gdal_translate -ot byte -co COMPRESS=LZW %s %s",
               "tmp_glcm_diss10_dem.tif",
               "glcm_altitude.tif"
))

glcm <- glcm(raster(slp_input),statistics= "dissimilarity",scale_factor=10,asinteger=TRUE)
writeRaster(glcm,"tmp_glcm_diss10_slp.tif")

system(sprintf("gdal_translate -ot byte -co COMPRESS=LZW %s %s",
               "tmp_glcm_diss10_slp.tif",
               "glcm_slope.tif"
```

```
))

##########################################################################################################
#########################                                  ZONAL STATISTICS
##########################################################################################################

#################### DEM ALTITUDE TEXTURE
system(sprintf("oft-stat -i %s -o %s -um %s",
               "glcm_altitude.tif",
               "stats_glcm_altitude.txt",
               paste0("segments/seg_lsms_btm",paste0(params,collapse = "_"),".tif")
               ))

#################### DEM SLOPE TEXTURE
system(sprintf("oft-stat -i %s -o %s -um %s",
               "glcm_slope.tif",
               "stats_glcm_slope.txt",
               paste0("segments/seg_lsms_btm",paste0(params,collapse = "_"),".tif")
))

#################### DEM ALTITUDE
system(sprintf("oft-stat -i %s -o %s -um %s",
               dem_input,
               "stats_altitude.txt",
               paste0("segments/seg_lsms_btm",paste0(params,collapse = "_"),".tif")
))

#################### DEM SLOPE
system(sprintf("oft-stat -i %s -o %s -um %s",
               slp_input,
               "stats_slope.txt",
               paste0("segments/seg_lsms_btm",paste0(params,collapse = "_"),".tif")
))

#################### ROADS
system(sprintf("oft-stat -i %s -o %s -um %s",
               "road_hec/dist_roads.tif",
               "stats_roads.txt",
               paste0("segments/seg_lsms_btm",paste0(params,collapse = "_"),".tif")
))

#################### RIVER
system(sprintf("oft-stat -i %s -o %s -um %s",
               "road_hec/dist_riv.tif",
               "stats_riv.txt",
               paste0("segments/seg_lsms_",paste0(params,collapse = "_"),".tif")
))

#################### HEC
system(sprintf("oft-stat -i %s -o %s -um %s",
               "road_hec/dist_hec.tif",
               "stats_hec.txt",
               paste0("segments/seg_lsms_btm",paste0(params,collapse = "_"),".tif")
))

#################### PROTECTED AREAS
system(sprintf("oft-stat -i %s -o %s -um %s",
               "9_PA/pa_30m.tif",
               "stats_PA.txt",
               paste0("segments/seg_lsms_btm",paste0(params,collapse = "_"),".tif")
))

#################### FLOODING
system(sprintf("oft-stat -i %s -o %s -um %s",
               "9_PA/flood.tif",
               "stats_flood.txt",
               paste0("segments/seg_lsms_btm",paste0(params,collapse = "_"),".tif")
))
```

```
#########################################################################################################
#########################                                  MERGE INTO DBF
#########################################################################################################
dbf <- read.dbf("segments/seg_lsms_btm3_16_0.1_50_10.dbf")
write.dbf(dbf,"segments/bckup_seg_lsms_btm3_16_0.1_50_10.dbf")

names(dbf)
dbf$sort_code <- row(dbf)[,1]

##### glcm altitude
df <- read.table("stats_glcm_altitude.txt")[,c(1,3)]
names(df) <- c("DN","g_ele_mean")
hist(df)

out <- merge(dbf,df,by.x="DN",by.y="DN",all.x=T)

##### slope
df <- read.table("stats_slope.txt")[,c(1,3)]
names(df) <- c("DN","slope_mean")
hist(df)

out <- merge(out,df,by.x="DN",by.y="DN",all.x=T)

##### dist to road
df <- read.table("stats_roads.txt")[,c(1,3)]
names(df) <- c("DN","d_rd_mean")
hist(df)

out <- merge(out,df,by.x="DN",by.y="DN",all.x=T)

##### dist to river
df <- read.table("stats_riv.txt")[,c(1,3)]
names(df) <- c("DN","d_sw_mean")
hist(df)

out <- merge(out,df,by.x="DN",by.y="DN",all.x=T)

##### dist to HEC
df <- read.table("stats_hec.txt")[,c(1,3)]
names(df) <- c("DN","hec_mean")
hist(df)

out <- merge(out,df,by.x="DN",by.y="DN",all.x=T)

##### presence of flood
df <- read.table("stats_flood.txt")[,c(1,3)]
names(df) <- c("DN","we_fl_mean")
hist(df)

out <- merge(out,df,by.x="DN",by.y="DN",all.x=T)

##### presence of PA
df <- read.table("stats_PA.txt")[,c(1,3)]
names(df) <- c("DN","PA_mean")
hist(df)

out <- merge(out,df,by.x="DN",by.y="DN",all.x=T)

out <- arrange(out,sort_code)
head(dbf)
head(out)

write.dbf(out,"segments/seg_lsms_btm3_16_0.1_50_10.dbf")

#################### Suitability shapefile obtained from FAO-BGD
df <- read.dbf("Zonal/CXB_suit.dbf")

df$sc_slp <- 0
df$sc_tex <- 0
```

```
df$sc_hec <- 0
df$sc_rod <- 0
df$sc_riv <- 0
df$sc_wat <- 0
df$sc_par <- 0
df$sc_lcm <- 0

df[df$slope_mean == 0 ,]$sc_slp <- 3
df[df$slope_mean > 0  ,]$sc_slp <- 2
df[df$slope_mean > 5  ,]$sc_slp <- 1
df[df$slope_mean > 10 ,]$sc_slp <- 0

df[df$g_ele_mean == 0 ,]$sc_tex <- 3
df[df$g_ele_mean > 0  ,]$sc_tex <- 2
df[df$g_ele_mean > 10 ,]$sc_tex <- 1
df[df$g_ele_mean > 50 ,]$sc_tex <- 0

df[df$d_rd_mean <10000 ,]$sc_rod <- 1
df[df$d_rd_mean <5000  ,]$sc_rod <- 2
df[df$d_rd_mean <2000  ,]$sc_rod <- 3

df[df$d_sw_mean <10000 ,]$sc_riv <- 1
df[df$d_sw_mean <5000  ,]$sc_riv <- 2
df[df$d_sw_mean <1000  ,]$sc_riv <- 3

df[df$hec_mean > 100  ,]$sc_hec <- 1
df[df$hec_mean > 500  ,]$sc_hec <- 2
df[df$hec_mean > 1000 ,]$sc_hec <- 3

df[df$we_fl_mean == 0 ,]$sc_wat <- 3
df[df$we_fl_mean == 1 ,]$sc_wat <- 0

df[df$PA_mean == 0 ,]$sc_par <- 3
df[df$PA_mean == 1 ,]$sc_par <- 0

df[df$CODE_R %in% c("ShT","MF","BS","SC","H"),]$sc_lcm <- 3

legend2
df$score <- df$sc_hec * df$sc_par * df$sc_riv * df$sc_rod * df$sc_slp * df$sc_tex * df$sc_wat * df$s

df$suitability <- 0
df[df$score > 0,]$suitability <- 1
df[df$score > 500,]$suitability <- 2
df[df$score > 1000,]$suitability <- 3
df[df$score > 2000,]$suitability <- 4

table(df$CODE_R,df$suitability)

write.dbf(df,"Zonal/CXB_suit.dbf")
summary(df)

shp1 <- readOGR("Zonal/CXB_suit.shp")
shp2 <- readOGR("Zonal/selection_3areas.shp")
shp3 <- shp1[shp2,]

shp3$option <-  over(shp3,shp2)$area
table(shp3@data$option)

summary(shp3[shp3$option == 647,]@data[,c("ele_mean","slope_mean","d_rd_mean","d_sw_mean")])

df <- read.dbf("Zonal/CXB_suit_identified.dbf")
```