Setting up to Train Random Forest Model

Tina Deenik

18/05/2021

```
#library(raster)
#library(sf)
#library(dplyr)
```

Stack parameters

First, call in all the rasters you will be using for your parameters:

```
##Call in optical data
#NDVI <- raster("P:/AserLab/Projects/OK_Wetlands/NDVI/NDVI_UTM10_UTM11_clipped.tif")
#NDWI <- raster("P:/AserLab/Projects/OK_Wetlands/NDWI/NDWI_UTM10_UTM11_clipped.tif")
#B2 <- raster("P:/AserLab/Projects/OK_Wetlands/Multispectral bands/B2_UTM10_UTM11_clipped.tif")
#B3 <- raster("P:/AserLab/Projects/OK_Wetlands/Multispectral bands/B3_UTM10_UTM11_clipped.tif")
#B4 <- raster("P:/AserLab/Projects/OK_Wetlands/Multispectral bands/B4_UTM10_UTM11_clipped.tif")
#B8 <- raster("P:/AserLab/Projects/OK_Wetlands/Multispectral bands/B8_UTM10_UTM11_clipped.tif")

##Call in radar data
#npol <- raster("P:/AserLab/Projects/OK_Wetlands/Radar/NPOL_UTM10_UTM11_clipped.tif")

##Call in lidar data
#gm <- brick("P:/AserLab/Projects/OK_Wetlands/LiDAR_gridmets/AllFiles_gridmetrics_April26_10m.ti
f") #note, the grid metrics were already a raster stack so call in as brick.
#DEM <- raster("P:/AserLab/Projects/OK_Wetlands/Mega_Raster_10m/AllFIles_DEM_10m.tif")
#TWI <- raster("P:/AserLab/Projects/OK_Wetlands/TWI_NEW/AllFiles_TWI_10m.tif")
#TPI1 <- raster("P:/AserLab/Projects/OK_Wetlands/TPI_NEW/AllFiles_TPI10_1.tif")
#TPI10 <- raster("P:/AserLab/Projects/OK_Wetlands/TPI_NEW/AllFiles_TPI10_10m.tif")

#Call in any other data
#sdist <- raster("P:/AserLab/Projects/OK_Wetlands/Dist_to_Stream/streams.tif")

##check resolution (should be 10x10m), crs, and extent
#res(DEM)
#crs(NDVI)
#extent(sdist)
#...keep going for all of them

#Looks like some are in lat/long, some are UTM. Before reprojecting and setting extents, stack t
he ones that are same to save some work for yourself.

##Stack same ext/crs data:
#satellite <- stack(NDVI,NDWI,B2,B3,B4,B8,npol)
#lidar <- stack(DEM,TWI,TPI1,TPI10)
```

Now we want to reproject everything to be in the same coordinate reference system, in our case UTM. Since LiDAR is already in the format we want it, we can just match the resolution, extents and crs of everything else to that.

```
##we will use the "to" function to set what's not in UTM to be the same is lidar.
#r_proj = projectRaster(satellite, to = lidar, crs = crs(lidar))
#r_proj2 = projectRaster(sdist, to = lidar, crs = crs(lidar))

##or you could do it manually:
#r_proj = projectRaster(NDVI, res = 10, crs = +proj=utm +zone=11 +datum=WGS84 +units=m +no_defs,
method = 'ngb')

##now save r_proj to file using write raster
#satellite2 <- writeRaster(r_proj,filename="P:/AserLab/Projects/OK_Wetlands/Stacked_parameters/T
RASH/satellite_crs_extent.tif")
#sdist2 <- writeRaster(r_proj2,filename="P:/AserLab/Projects/OK_Wetlands/Stacked_parameters/TRAS
H/sdist_crs_extent.tif")

##grid metrics were a bit different - it's already a stack and they are in UTM just need to set
 crs label.
#crs(gm)  <- "+proj=utm +zone=11 +datum=WGS84"
#gridmet2 <- projectRaster(gm, to = lidar)

##save to file
#gridmetnew <- writeRaster(gridmet2,filename="P:/AserLab/Projects/OK_Wetlands/Stacked_parameter
s/TRASH/gridmet_crs_extent.tif")

#now stack all the files w the correct crs and extent
#bigstack <- stack(satellite2,lidar,sdist2,gridmetnew)

#now save the stack
#writeRaster(bigstack,filename="P:/AserLab/Projects/OK_Wetlands/Stacked_parameters/bigstack.ti
f")
```

When you plot the raster brick bigstack, be sure there are the correct number of parameters. The first time I did this, the grid metrics did not separate and I had 13 parameters instead of 19.

```
#bigstack <- brick("P:/AserLab/Projects/OK_Wetlands/Stacked_parameters/datastack_19param_April_2
7.tif")
#plot(bigstack)
```

The parameters are not labelled correctly. There are 19 parameters stacked in the following order:

satellite: NDVI[1],NDWI[2],B2[3],B3[4],B4[5],B8[6],npol[7] lidar: DEM[8],TWI[9],TPI1[10],TPI10[11] other: stream distance[12] grid mets: zmean[13], zmax[14], zsd[15], imax[16], imean[17], imin[18], isd[19]

I re-plotted some of the parameters to see if they matched the stacked list (i.e. by visual inspection). You can call in and plot individual parameters from the stack using plot(bigstack[[1]]). You can also call summary and see if the data matches. I'm sure there is a better way to do this.Keep in mind there might be some differences since the projection and extents changed a bit.

Create 100,000 random points from existing wetland database

```
##read in wetland database vector file
#wetlands <- read_sf("P:/AserLab/Projects/OK_Wetlands/Wetland_Database_Old/Okanagan_wetlands_cli
pped.shp")

##let's just select wetland class for now.
#wet_sel = wetlands %>% select(CLASS1)
#class(wet_sel)
#print(wet_sel)
#tail(wet_sel)

#now we want to clean up the class labels
#wet_sel = wet_sel %>% dplyr::mutate(CLASS1 = recode(CLASS1, March = 'Marsh'))
#wet_sel = wet_sel %>% dplyr::mutate(CLASS1 = recode(CLASS1, Flood_Low_Bench = "Flood_group"))
#wet_sel = wet_sel %>% dplyr::mutate(CLASS1 = recode(CLASS1, Flood_Mid_Bench = "Flood_group"))
#wet_sel = wet_sel %>% dplyr::mutate(CLASS1 = recode(CLASS1, "Flooded Land" = "Flood_group"))
#wet_sel = wet_sel %>% dplyr::mutate(CLASS1 = recode(CLASS1, Pond = "Shallow_water"))
#wet_sel = wet_sel %>% dplyr::mutate(CLASS1 = recode(CLASS1, "Shallow Open Water" = "Shallow_wat
er"))
#wet_sel = wet_sel %>% dplyr::mutate(CLASS1 = recode(CLASS1, "Shallow open water" = "Shallow_wat
er"))
#wet_sel = wet_sel %>% dplyr::mutate(CLASS1 = recode(CLASS1, "Shallow water" = "Shallow_water"))
#wet_sel = wet_sel %>% dplyr::mutate(CLASS1 = recode(CLASS1, Shallow_open_water = "Shallow_wate
r"))
#wet_sel = wet_sel %>% dplyr::mutate(CLASS1 = recode(CLASS1, Shallow_open_water = "Shallow_wate
r"))
#wet_sel = wet_sel %>% dplyr::mutate(CLASS1 = recode(CLASS1, 'Saline Meadow' = "Transition_clas
s"))
#wet_sel = wet_sel %>% dplyr::mutate(CLASS1 = recode(CLASS1, 'Alkaline Pond' = "Alkaline_pond"))

#types <- as.character(unique(unlist(wet_sel)))
#types

#wet_sel = wet_sel %>% dplyr::filter(!CLASS1 %in% c('NA', 'Lake', 'Golf Course Pond', 'Swamp,Mar
sh', 'Reservoir'))

##drop NA's
#wet_sel = wet_sel %>% dplyr::filter(!CLASS1 %in% NA)

#write to file so you don't need to do it again
#write.csv(wet_sel, "P:/AserLab/Projects/OK_Wetlands/Stacked_parameters/wet_sel_clean_allwetland
s_April_28_2021.csv")

#Now we have a list of wetland classes with point locations.
#load("P:/AserLab/Projects/OK_Wetlands/Wetland_Database_Old/wetland_data_CLASS_clean.Rda")
#head(wet_sel)
#unique(wet_sel$CLASS1)
# "Shallow_water"    "Marsh"            "Transition_class" "Flood_group"      "Swamp"
# "Alkaline_pond"
```

```
##set seed so the random points are the same
#set.seed = 123

#pts <- st_sample( #sample 100,000 random points from within the polygons
#  wet_sel,
#  size =100000,
#  type = "random",
#  exact = TRUE,
#  warn_if_not_integer = TRUE,
#  by_polygon = FALSE
#)

##convert to sf object
#newpts <- st_as_sf(pts)

##add sequential id for the points, probably not needed but can come in handy
#newpts$pID = 1:nrow(newpts)

##join spatial points to selected wetland data - remember this is just the wetland class in wet_
sel.
##This is now a spatial object.
#pts_wet = st_join(newpts, wet_sel, type = "intersect")

##remember to save and write point locations to a csv
#save(pts_wet, file = "P:/AserLab/Projects/OK_Wetlands/Wetland_Database_Old/100000_random_wetlan
d_points.Rda")
#write.csv(pts_wet, file ="P:/AserLab/Projects/OK_Wetlands/Wetland_Database_Old/100000_random_we
tland_points.csv")
```

Join parameters to 100,000 point locations

```
##Now bring in the stacked parameters again - I called this bigstack above.
#raster_stack <- brick("P:/AserLab/Projects/OK_Wetlands/Stacked_parameters/datastack_19param_Apr
il_27.tif")

##create a new df by extracting the data from our big raster stack at our randomly sampled wetla
nd points
#param_df <- extract(raster_stack, pts_wet, df = TRUE)

##back up pts_wet, just in case
#bu_pts_wet = pts_wet

#param_out = cbind(pts_wet, param_df)

##need to rename column names to match parameters - follow order noted above:
#param_out <- rename(param_out, Wetland_class = CLASS1)
#param_out <- rename(param_out, NDVI = datastack_19param_April_27.1)
#param_out <- rename(param_out, NDWI = datastack_19param_April_27.2)
#param_out <- rename(param_out, B2 = datastack_19param_April_27.3)
#param_out <- rename(param_out, B3 = datastack_19param_April_27.4)
#param_out <- rename(param_out, B4 = datastack_19param_April_27.5)
#param_out <- rename(param_out, B8 = datastack_19param_April_27.6)
#param_out <- rename(param_out, npol = datastack_19param_April_27.7)
#param_out <- rename(param_out, DEM = datastack_19param_April_27.8)
#param_out <- rename(param_out, TWI = datastack_19param_April_27.9)
#param_out <- rename(param_out, TPI1 = datastack_19param_April_27.10)
#param_out <- rename(param_out, TPI10 = datastack_19param_April_27.11)
#param_out <- rename(param_out, sdist = datastack_19param_April_27.12)
#param_out <- rename(param_out, zmean = datastack_19param_April_27.13)
#param_out <- rename(param_out, zmax = datastack_19param_April_27.14)
#param_out <- rename(param_out, zsd = datastack_19param_April_27.15)
#param_out <- rename(param_out, imax = datastack_19param_April_27.16)
#param_out <- rename(param_out, imean = datastack_19param_April_27.17)
#param_out <- rename(param_out, imin = datastack_19param_April_27.18)
#param_out <- rename(param_out, isd = datastack_19param_April_27.19)

##save
#write.csv(param_out, "P:/AserLab/Projects/OK_Wetlands/RF_Model/Training_input/pts_w_parameters_
column_names.csv")
```

Now param_out is my training table/database w wetland class, and all the 19 parameter values associated w each location (x100,000). We should check that the training database is representing each class well enough. We will do that once we are ready to dive into the RF model.