Script Draft: 'stems_grids_2020125'

LQ-SRM

22/02/2022

Contents

Action

This following includes R Markdown documentation of steps to derive stems/ha predictor from LiDAR point cloud for the TCC-EFI predictive ecosystem model. This pipeline was adopted from the original script shared by LQuan and reformatted to use inputs from the normalized, cleaned and classified point cloud produced in previous deliverable '13_lidR_PointCloud_Processing'.All package lists, markdown outputs, and virtual environment settings were stored in the github repository titled '14_lidR_ITDS_stemha_L'.

1 Import normalized LAS data

Since, the output from previous deliverable '13_lidR_Processing' that is needed here is in point-form, there are challenges with how to share the full collection remotely due to file size and internet connection here. Instead, I've saved all system files from that pipeline environment in a .RData file in the U:drive folder 'EFI Project - SRM 2021-2022' titled 'lidar-processing-pipeline.RData'. File paths to 'AOI' in the original script were replaced with the normalized LAScatolog 'las_ctg_ahbau_csf_sor_norm' and normalized tile 'las_tile_ahbau_csf_sor_norm. In effort to get a sample ready for you meeting tomorrow, I've tested this pipeline with the normalized tile chunk only and rendered in 2D using base plot functions. This tile chunk can also be processed, classified, and normalized quickly enough on your end with functions below which were copied and pasted from the previous output. Make sure to unnormalize if using the .RData files.

```
AOI<-"Meldrum"
AOIpath<-paste0("H:\\",AOI,"\\")
GRIDS<-paste0(AOIpath,"GRIDS\\")
TREES<-paste0(AOIpath,"TREES\\")
defaultCRS<-CRS("+init=EPSG:3153")
```

...replaced with:

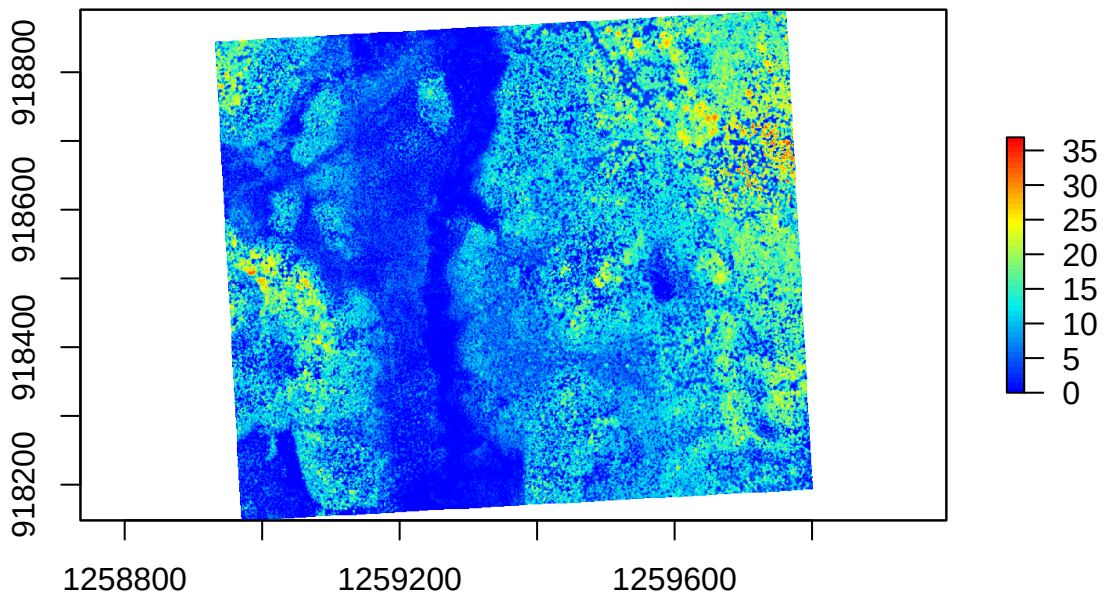
```
las_tile_ahbau = readLAS("./Data/Ahbau/Las_v12_ASPRS/093g030122ne.las", select = 'xyzr')
las_tile_ahbau = filter_duplicates(las_tile_ahbau)
las_tile_ahbau_csf = classify_ground(las_tile_ahbau, csf(sloop_smooth=TRUE, 0.5, 1))
las_tile_ahbau_csf_so = classify_noise(las_tile_ahbau_csf, sor(k=10, m=3))
```

```
## Metrics computation: [=--------------------------------------------------] 3% (1 threads)Metrics compu
```

```
las_tile_ahbau_csf_sor = filter_poi(las_tile_ahbau_csf_so, Classification != LASNOISE)
las_tile_ahbau_csf_sor_norm = normalize_height(las_tile_ahbau_csf_sor, knnidw())
```

```
## Inverse distance weighting: [=====----------------------------------------------] 10% (1 threads)Inve
```

```
# plot normalization using derived chm
las_tile_ahbau_chm = grid_canopy(las_tile_ahbau_csf_sor_norm, 1, dsmtin(8))
plot(las_tile_ahbau_chm, col = height.colors(50))
```



2 Individual Tree Detection

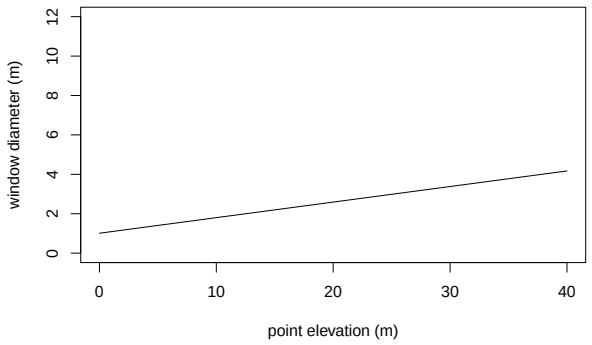
To avoid replicated trees across adjacent tiles, the catalog engine 'uniqueness' function was applied to the local maxima algorithm. This was fitted with your custom window function 'wf', which is also plotted below with your adopted code chunks. Super interesting stuff. For visualization, the detected tree top points were plotted as 'sp>sf' object over chm raster.

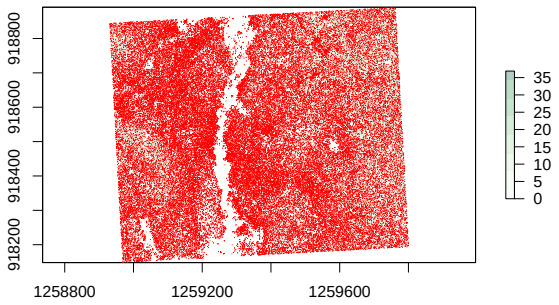
```
# window function (Quan, 2022)
wf<-function(x){ #so far these parameters are best will try 6m floor.
  a=0.179-0.1
  b=0.51+0.5 #Go big but not 2.49 big. maybe increase slope a hair.
  y<-a*x+b #y[x<15]=2.2
  return(y)}
heights <- seq(0,40,0.5)
window <- wf(heights)
plot(heights, window, type = "l",  ylim = c(0,12), xlab="point elevation (m)", ylab="window diameter (m)

# find trees
las_tile_ahbau_ttops <- find_trees(las_tile_ahbau_csf_sor_norm, lmf(wf), uniqueness = "bitmerge")
```

```
## Local maximum filter: [======----------------------------------------------] 12% (1 threads)Local maxim
```

```
proj4string(las_tile_ahbau_ttops) <- defaultCRS
las_tile_ahbau_ttops_sf = st_as_sf(las_tile_ahbau_ttops)
mypalette<-brewer.pal(8,"Greens")
plot(las_tile_ahbau_chm, col = mypalette, alpha=0.3)
plot(st_geometry(las_tile_ahbau_ttops_sf["treeID"]), add=TRUE, cex = 0.001, pch=19, col = 'red', alpha=0
```

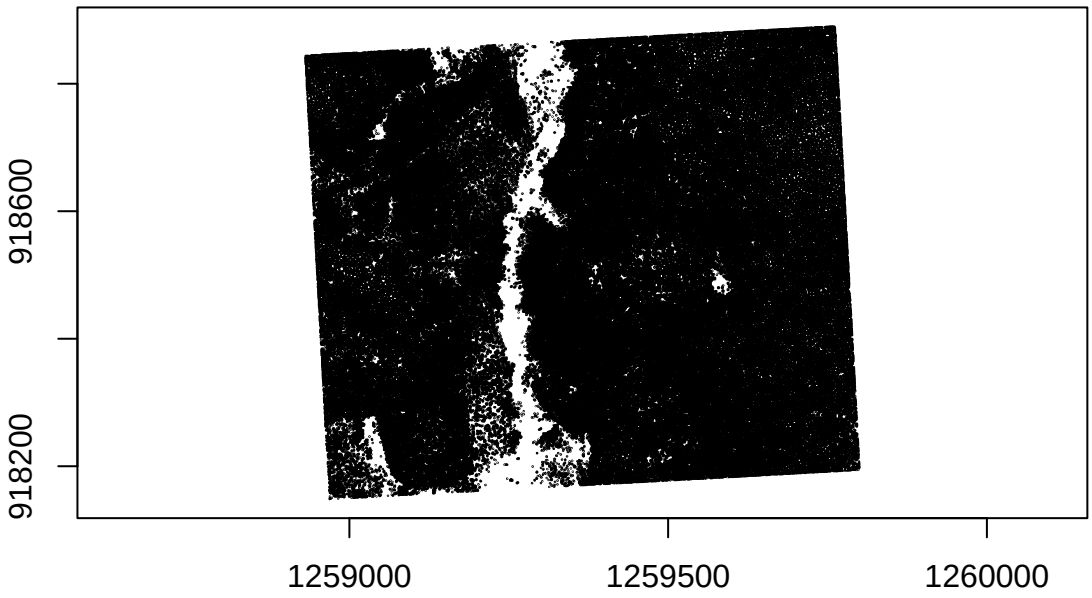




3 Individual Tree Segmentation (treeID-proofed)

Not 100% sure, but from partial reading of the lidR bookdown, it looks like Romaine is suggesting that because of tile bufferring needed in the engine processing, there is need to segment the crowns to make sure trees are not double counted by multiple tiles. He notes this is because the engine was designed for batch processing or even multi-cpu processing. See here his mention at top of section 14.12. Note that Im using deprecated function names here from lidR pkg version 3.2.3. Im not sure if new pkg version is out yet since the bookdown has been updated. Also, Im apprehensive to mess with my environment before finishing these outputs.

```
algo = dalponte2016(las_tile_ahbau_chm, las_tile_ahbau_ttops)
las_tile_ahbau_segmented = segment_trees(las_tile_ahbau_csf_sor_norm, algo)
las_tile_ahbau_segmented_crowns = delineate_crowns(las_tile_ahbau_segmented, type = 'convex')
sp::plot(las_tile_ahbau_segmented_crowns, cex=0.001, axes = TRUE, alpha=0.5)
```



4 Derive 'stemsha_L' Predictor From Segmented Tree Count

```
#sp::spTransform(las_tile_ahbau_segmented_crowns) = defaultCRS
las_tile_ahbau_segmented_crowns_sf = st_as_sf(las_tile_ahbau_segmented_crowns)
las_tile_ahbau_segmented_crowns_sf
```

```
## # A tibble: 73,158 x 5
##     treeID     XTOP    YTOP  ZTOP                                      geometry
##      <dbl>    <dbl>   <dbl> <dbl>                                  <POLYGON>
##  1 5.41e17 1259524. 918213. 13.6  ((1259527 918210.2, 1259527 918210.1, 1259526~
##  2 5.41e17 1259523. 918213. 11.7  ((1259526 918214, 1259526 918213.2, 1259526 9~
##  3 5.41e17 1259526. 918212. 10.0  ((1259527 918212.8, 1259527 918212.3, 1259526~
##  4 5.41e17 1259529. 918214   9.07 ((1259532 918212.2, 1259531 918211.2, 1259530~
##  5 5.41e17 1259530. 918215. 12.5  ((1259532 918214.4, 1259532 918214.3, 1259532~
##  6 5.41e17 1259523. 918207. 11.2  ((1259524 918206.1, 1259523 918206, 1259523 9~
##  7 5.41e17 1259519. 918212.  8.64 ((1259520 918211.3, 1259520 918211.2, 1259519~
##  8 5.41e17 1259525. 918210. 10.7  ((1259527 918209.1, 1259526 918208, 1259523 9~
##  9 5.41e17 1259521. 918211. 13.0  ((1259521 918208, 1259521 918208, 1259520 918~
## 10 5.41e17 1259518. 918215. 11.7  ((1259520 918214.7, 1259520 918213.7, 1259520~
## # ... with 73,148 more rows
```

```
psych::describe(las_tile_ahbau_segmented_crowns_sf$treeID)
```

```
## # A tibble: 1 x 13
##    vars     n    mean      sd  median trimmed     mad     min     max   range
##   <dbl> <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1     1 73158 5.41e17 9.94e13 5.41e17 5.41e17 1.25e14 5.41e17 5.41e17 3.73e14
## # ... with 3 more variables: skew <dbl>, kurtosis <dbl>, se <dbl>
```