Lab 8: TLS and MLS

Anthony Stewart & Rachel Deininger

2023-05-17

Objectives:

1. Exploring TLS and ALS data in Cloud Compare
2. Segmenting and comparing trees in ALS vs. TLS

Data and Software:

- R Studio
- CloudCompare

What you will turn in:

Lab 8 Quiz (https://canvas.uw.edu/courses/1633883/quizzes/1862065)

Introduction

This lab is all about **Terrestrial Lidar Scanning (TLS)**. Unlike aerial lidar or **Aerial Laser Scanning** which is acquired from airplanes or drones, TLS is acquired from scanners on the ground. TLS point clouds tend to be **much** denser and more detailed than ALS which we have been working with most of the time.

TLS vs. MLS

Terrestrial Laser Scanning (TLS), is a ground-based mechanism for collecting LiDAR scans. Unlike Airborne Laser Scanning (ALS), Which is mounted to an aircraft, TLS is collected from surface level scanners wither attached to tripods, backpacks, vehicles, or handheld. Mobile Laser Scanning (MLS), is a type of TLS collected using a hand-held device that an operator walks through an area. ALS has the benefit of being able to cover a large area very quickly, while MLS has the benefit of finer details and more precise individual tree measurements.

Two devices for TLS are the FARO Focus and Leica BLK 360, which require the user to set up a tripod at each position from which they want to take a scan. TLS scanners that are mounted on a tripod can only see what is directly in line of sight of the scanner, so they cannot create whole 3D images from a single scan. To create a complete point cloud that allows variables such as tree height and diameter at breast height (DBH) to be determined, scans must be taken from multiple positions.

FARO Laser Scanner

These scans can then be merged to create a more complete point cloud. In order to accurately combine scans in processing a recognizable object that can be tagged as the same location in each scan is needed. Often a checkered target or brightly colored orb mounted on conduit is used. For this to be effective, each orb much be positioned in a location where it can be seen my at least two scans, and each scan must be able to see three orbs, this setup can be seen in Figure 1.

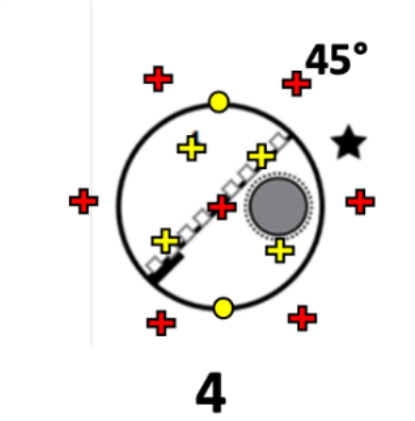
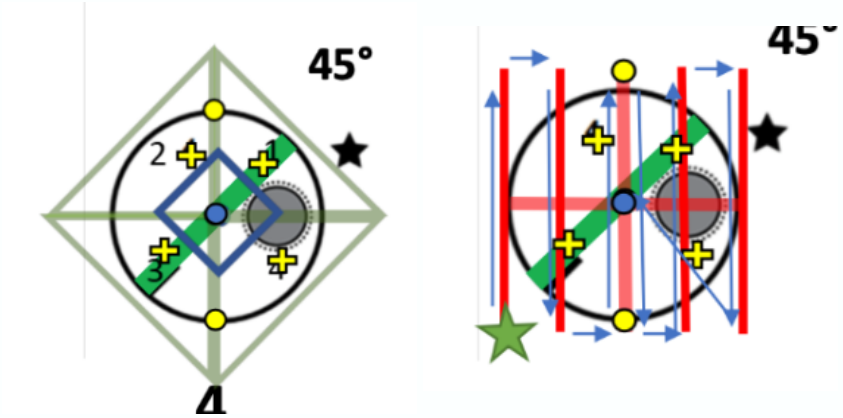


Figure 1

The Leica BLK2GO and Zeb Horizon are both mobile laser scanners (MLS). To operate the user walks a steady path throughout the subplot to capture a point cloud from many different angles. Two different path types are the triangle method (Figure 2) and the lawnmower (Figure 3). Unlike the FARO and non-mobile Leica, the MLS devices can create a single decreasing the amount of data processing required. MLS devices also have the advantage of taking less time to collect since there is no need to set up as many orbs or targets, although having a north and south marker is recommended.

Leica BLK2GO



Georeferencing

While ALS data is dependent on being accurately georeferenced, TLS data often doesn't need to be placed

accurately. TLS data most often is only related to the relative location of object within the scan rather than the geographic locations. An example is the costal cliff face erosion. All the point clouds are aligned to each other and it doesn't matter if they are located geographically.

However, if we are going to derive forest plot information from TLS to be scaled up to ALS or satellite data, we do need to know the precise locations of the scans. This initial georeferencing step I did for you.

To georeference the data. There are a few different ways that this can be done. Using survey grade GPS receivers marking the location of targets that can be identified in the lidar point cloud. Targets that are used to TLS scanning are often spheres or checkerboard patterns that can be seen in the point cloud.

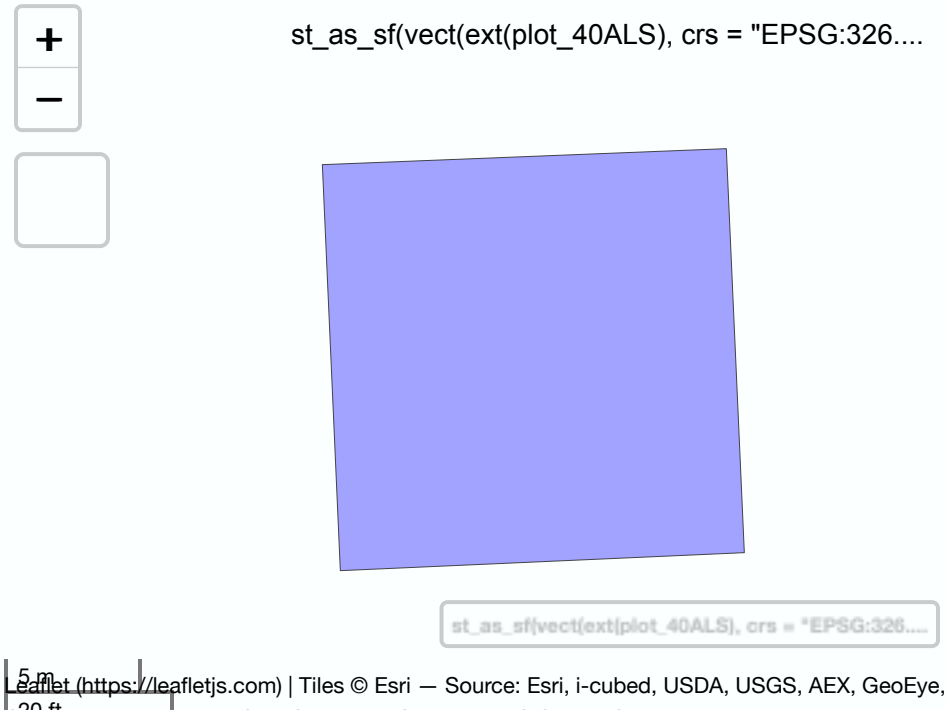
Another way that TLS scans can be georeferenced is to match the point cloud with an ALS cloud. Identifying features in both clouds and transforming the TLS to match. Ideally the features are things that don't move or grow (i.e. not trees) but trees can be used if there are not better options.

Part 1: Cloud Compare

For this lab there is a few point clouds to look at.

The first two are ALS plots from the NASA G-Liht Mission which you can check out here (https://gliht.gsfc.nasa.gov/). These were plots that we visited up near Anchorage Alaska.

- `plot_040_2.las`
- `plot_048_1.las`



The next two are MLS scans which were taken in these plots. They were specifically taken with the Leica BLK2GO and originally were super dense with up to 100million points and a density >100,000/m^2. So for this lab I have filtered and cropped them down to something more workable

- `Scan26_plot40_clipprj.las`
- `Scan43_plot48_clipfilprj.las`

The scan numbers refer to the number of total scans taken in the BLK2GO but they are referenced to the plot

number using cloud compare

For this part of the lab, compare the plot 40 ALS vs MLS and then the plot 48 ALS and MLS in cloud compare. If you remember from earlier cloud compare does not use conventional units. SO be sure to keep track of what you are comparing. It's good practice to keep the plot scans separate so you only compare 40MLS->40ALS and 48MLS->48ALS

QUESTION 1: Take a screenshot of one of the plot 40 and 48 TLS scans rendered in a cool way in Cloud Compare. What do you notice about the trees and ground?

QUESTION 2:Take screenshot with the plot 40 MLS and ALS combined. Discuss some of the similarities and differences between the TLS and ALS. Do you notice any errors or mis-alignments?

Part 2: R processing - deriving metrics between ALS and MLS

We've definitely had some issues with R for a lot of students and almost every one has been unique! So briefly here are a few things we can do to fix some issues:

- restart R and reinstall libraries using
  `install.packages("LIBRARY", lib = "FILE PATH TO YOUR LIBRARY")`
  - The file path to your library can be found using `.Library`
- If you get the 00LOCK file error try to copy the file path and delete that folder
- If that doesn't work then we may need to download and reinstall R which can be done from the CRAN website: https://cran.r-project.org/ (https://cran.r-project.org/)

For now:

As always set your working directory

```
setwd("YOUR OWN WORKING DIRECTORY")
```

Load in your libraries

```
library(lidR)
library(terra)
library(rgl)
```

We're going to compare the lidar point clouds from ALS to lidar point clouds from MLS

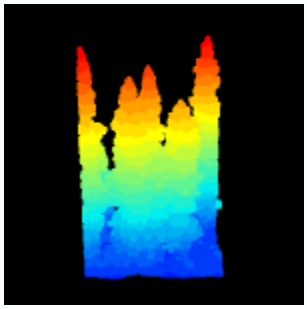
To do this, we'll need to 1. Normalize our lidar point clouds

```
2. Generate canopy height models

3. Segment trees

4. Explore voxel metrics
```

I'm going to show you how to do this with plot 40 ALS and MLS but then I want you to do the same with plot *48 MLS*

Let’s load the data

```
plot_40 <- readTLSLAS("Lab 8/LAB8_DATA_2023/Scan26_plot40_clipprj.las")
#plot_40@header

plot_40ALS <- readLAS("Lab 8/GLiHT/plot_040_2.las")
plot((plot_40ALS))
plot((plot_40))
```



Now let’s classify our ground points because MLS does not have any classification yet

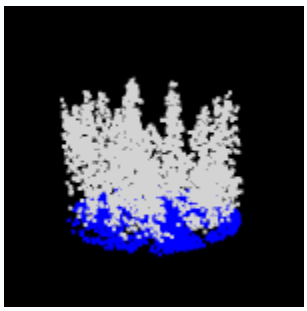
```
plot_40_ground <- classify_ground(plot_40, algorithm = pmf(ws = 0.5, th = 0.1))
```

```
## Morphological filter: 3% (1 threads)
Morphological filter: 4% (1 threads)
Morphological filter: 5% (1 threads)
Morphological filter: 6% (1 threads)
Morphological filter: 7% (1 threads)
Morphological filter: 8% (1 threads)
Morphological filter: 9% (1 threads)
Morphological filter: 10% (1 threads)
Morphological filter: 11% (1 threads)
Morphological filter: 12% (1 threads)
Morphological filter: 13% (1 threads)
Morphological filter: 14% (1 threads)
Morphological filter: 15% (1 threads)
```

```
plot(plot_40_ground, color = "Classification")
```

Our ALS data should already have ground classified

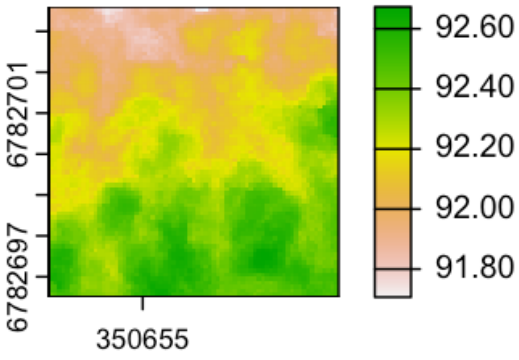
```
plot(plot_40ALS, color = "Classification")
```



We can now build a DTM for both of our point clouds. Because these are such small areas, I'm able to specify 0.1m as the resolution. Don't do this for large areas!

Here's the MLS

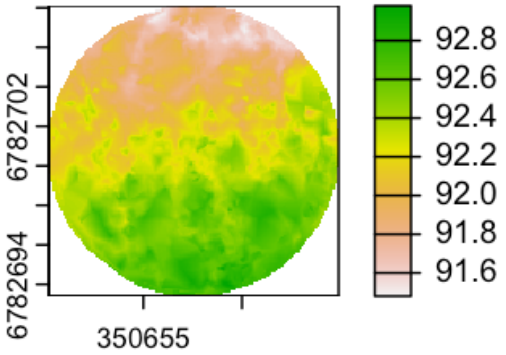
```
plot_40_dtm <- rasterize_terrain(plot_40_ground, res = 0.1, algorithm = knnidw())
plot(plot_40_dtm)
```



```
#plot_dtm3d(plot_40_dtm)
```

Here's the ALS

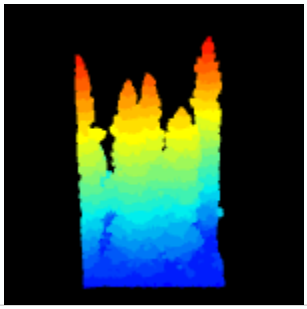
```
plot_40ALS_dtm <- rasterize_terrain(plot_40ALS, res = 0.1, algorithm = tin())
plot(plot_40ALS_dtm)
```



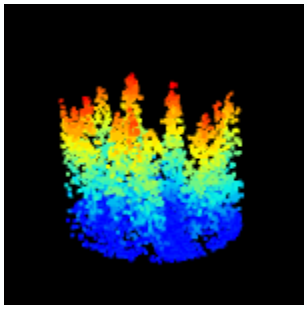

```
#plot_dtm3d(plot_40ALS_dtm)
```

Now we normalize our point cloud heights with the DTM raster that we created for each of the ALS and TLS point clouds

```
plot_40N <- normalize_height(plot_40_ground, dtm = plot_40_dtm, algorithm = knnidw())
plot(plot_40N)
```



```
plot_40ALSN <- normalize_height(plot_40ALS, dtm = plot_40ALS_dtm, algorithm = knnidw
())
plot(plot_40ALSN)
```



QUESTION 3: Why do we normalize the point cloud? Also why did we need to classify ground points before doing so?

Part 2a: Segmenting Trees

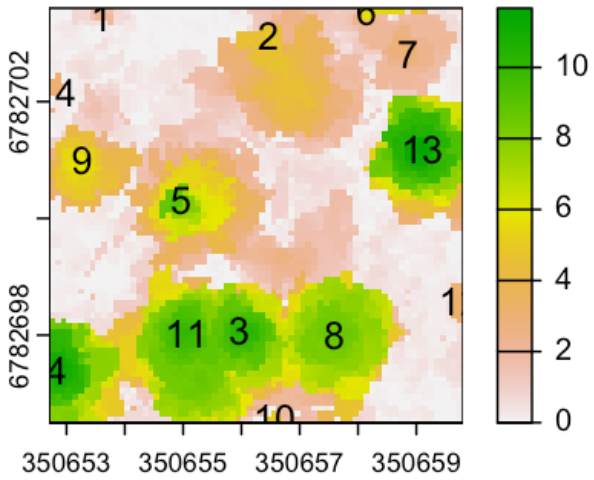
Let's start to segment our trees in just the MLS point cloud. Here this code is kind of jumbled together but it has these steps which should be familiar:

1. We rasterize the canopy to create a Canopy Height Model (CHM) using the `p2r` algorithm.
2. We then use the `locate_trees` function to locate the local maximums of tree heights in the normalized point cloud
3. we then have two algorithms for segmenting trees:
    - `dalponte2016`
    - `silva2016`
4. We then use these algorithms to segment the point clouds and plot them.

```
plot_40N_chm <- rasterize_canopy(las = plot_40N, res = 0.1, algorithm = p2r())
plot_40N_ttops <- locate_trees(plot_40N, lmf(ws = 1))
```

```
## Local maximum filter: [===-------------------------------------------------] 7% (1 t
hreads)
Local maximum filter: [====------------------------------------------------] 8% (1 thre
ads)
Local maximum filter: [====------------------------------------------------] 9% (1 thre
ads)
Local maximum filter: [=====-----------------------------------------------] 10% (1 thr
eads)
Local maximum filter: [=====-----------------------------------------------] 11% (1 thr
eads)
Local maximum filter: [======----------------------------------------------] 12% (1 thr
eads)
Local maximum filter: [======----------------------------------------------] 13% (1 thr
```

```
plot(plot_40N_chm)
terra::text(vect(plot_40N_ttops), "treeID")
```



```
eads)
#plot(sf::st_geometry(plot_40N_ttops),add = T)


algo1 <- dalponte2016(plot_40N_chm, plot_40N_ttops, th_tree = 0.1, th_seed = 0.1, max
_cr = 10, th_cr = 0.1)

algo2 <- silva2016(plot_40N_chm, plot_40N_ttops, max_cr_factor = 0.3, exclusion = 0.
1)
# plot_40N_trees1 <- segment_trees(plot_40N, algo1) # segment point cloud
# plot_40N_trees2 <- segment_trees(plot_40N, algo2) # segment point cloud

plot_40N_trees1 <- segment_trees(plot_40N, algo1) # segment point cloud
plot_40N_trees2 <- segment_trees(plot_40N, algo2) # segment point cloud
```

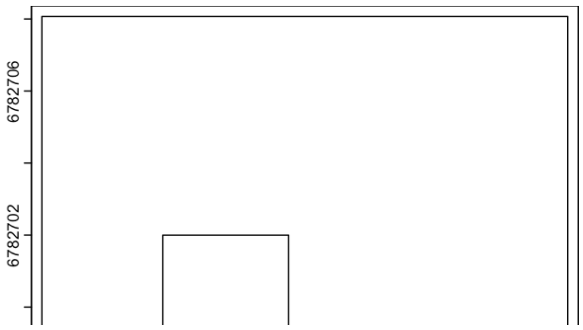
QUESTION 4: Look up the algorithms, what are 2 differences between the dalponte2016 and the silva2016 algorithms?

Part 2b: Singling out a tree for ALS and TLS comparisons

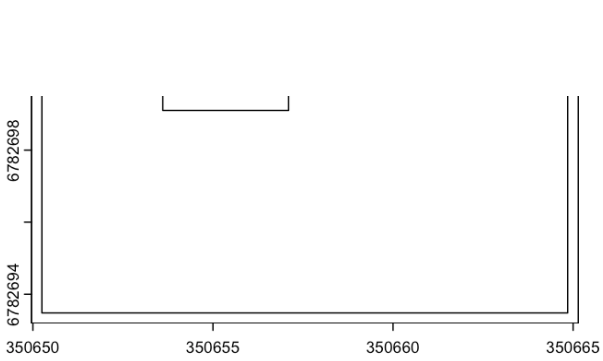
Here I'm using the `filter_poi` function to grab a specific tree. You can check out the number of trees by looking at the `plot_40N_trees2$treeID` values. Mostly I just picked number 5 that I thought looked good according to the plot above

Next I plotted the extent of the single tree onto the ALS extent to see if they overlapped. Once they did I converted the `tls_tree` extent into a vector with a projection that I can use to clip out the ALS point cloud

```
tls_tree <- filter_poi(plot_40N_trees2, treeID == 5)
plot(ext(plot_40ALSN), type = "polygons")
plot(ext(tls_tree), type = "polygons", add = T)
```



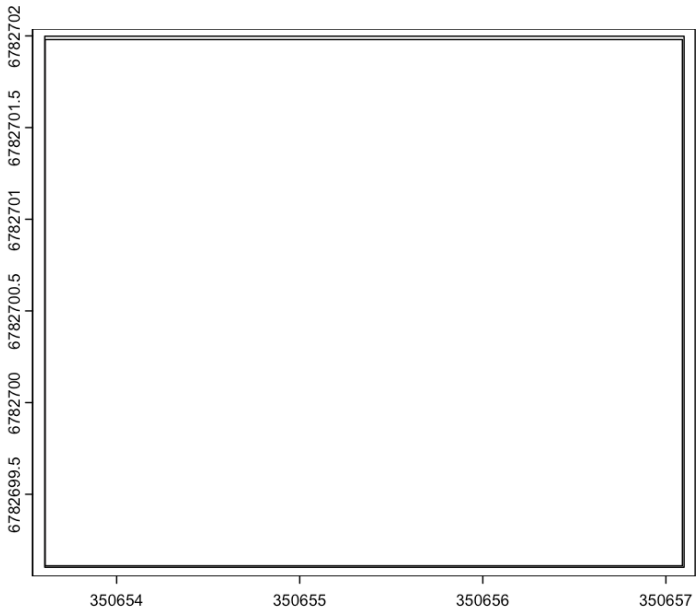
Local maximu 58% (1 thr




```
tls_tree_ext <- st_as_sf(vect(ext(tls_tree), crs = "EPSG:32606"))
```

Let's clip out the ALS tree and view our cutout of the ALS with the TLS to see if they align

```
plot_40ALS_treecrop <- clip_roi(plot_40ALSN, tls_tree_ext)
plot(ext(plot_40ALS_treecrop), type="polygons")
plot(ext(tls_tree), type = "polygons", add = T)
```



Now we can take that single tree area and repeat the steps in **Part 2a**

```
plot_40ALSN_chm <- rasterize_canopy(las = plot_40ALS_treecrop, res = 0.1, algorithm =
p2r())
plot_40ALSN_ttops <- locate_trees(plot_40ALS_treecrop, lmf(ws = 1))

algo2_ALS <- silva2016(plot_40N_chm, plot_40N_ttops, max_cr_factor = 0.3, 0.5)

ALS_tree <- segment_trees(plot_40ALS_treecrop, algo2_ALS) # segment point cloud
plot(ALS_tree)
plot(tls_tree)
```

QUESTION 5: Check out the differences between the two point clouds. Besides the lack of points, what is different structurally that you see between ALS and MLS?

Part 3: Voxelization

This part is going to be a very brief work flow to introduce the idea of voxels. The term voxel is a combination of Volume and pixel. They are essentially 3D pixels. They are defined by XYZ coordinates, an area value, and additional user defined values.

With TLS data, voxels are typically used in two distinctly different ways. The first way is to normalize the point density of a TLS scan. There are more points per unit area close to the scanner. A wall 3m from a TLS scanner could have a million points per m2 while a wall 100m from the scanner may only have 100s or dozens of points per m2 depending on the scan resolution set on the scanner. To normalize the point density is to decimate the point cloud so there are an equal amount of points per m2 on the wall 3m from the scanner and the wall 100m from the scanner.

The second way voxels are used is by imposing a voxel grid on an area and then counting the number of points within each voxel. This method typically is run on an already decimated point cloud. The output of this method is a table of “voxel metrics”.

Voxel metrics have been used extensively in quantifying how much “stuff” is in an area. Voxels are one of the best ways to estimate biomass and biomass change due to some disturbance event such as fire.

We can voxelize our ALS point clouds up to a certain level until the voxels lose point information because of the lack of points overall

Check out the 1m voxel resolution here

```
ALS_tree_vox1 <- voxel_metrics(ALS_tree, ~list(N = length(Z)), 1)
plot(ALS_tree_vox1, color="N", pal = heat.colors(10), size = 4, bg = "white", voxel =
TRUE)
```

```
## Warning in is.character(pal) && pal == "auto": 'length(x) = 10 > 1' in coercion
## to 'logical(1)'
```

Now compare to the 0.5m voxel resolution here

```
ALS_tree_vox2 <- voxel_metrics(ALS_tree, ~list(N = length(Z)), 0.5)
plot(ALS_tree_vox2, color="N", pal = heat.colors(10), size = 4, bg = "white", voxel =
TRUE)
```

```
## Warning in is.character(pal) && pal == "auto": 'length(x) = 10 > 1' in coercion
## to 'logical(1)'
```

Whereas with TLS we can go down to 0.1 meters and get a pretty good representation. Going further

becomes a computational burden

```
tls_tree_vox <- voxel_metrics(tls_tree, ~list(N = length(Z)), 0.1)
plot(tls_tree_vox, color="N", pal = heat.colors(10), size = 4, bg = "white", voxel =
TRUE)
```

```
## Warning in is.character(pal) && pal == "auto": 'length(x) = 10 > 1' in coercion
## to 'logical(1)'
```

To quantify the difference between the ALS and MLS voxel models we can sum up the number of voxels in each model

- MLS

```
tls_total <- sum(tls_tree_vox[which(tls_tree_vox$N >Z), 4])
tls_total
```

```
## [1] 63461
```

- ALS

```
ALS_total <- sum(ALS_tree_vox1[which(ALS_tree_vox1$N >Z), 4])
ALS_total
```

```
## [1] 251
```

QUESTION 6: Check out the differences between the two voxelized point clouds. What do you notice about the structure between the two. Also do you think the MLS voxelization is 100% accurate? Why or why not?

Now we can look at lidar crown metrics for the ALS compared to the TLS

```
metrics_tls <- crown_metrics(tls_tree, ~list(z_max = max(Z), z_mean = mean(Z))) # cal
culate tree metrics

as.data.frame(metrics_tls)
```

```
##   treeID    z_max   z_mean                       geometry
## 1      5 9.070053 2.394663 POINT Z (350655 6782700 9.0...
```

```
metrics_ALS <- crown_metrics(ALS_tree, ~list(z_max = max(Z), z_mean = mean(Z))) # cal
culate tree metrics

as.data.frame(metrics_ALS)
```

```
##   treeID z_max   z_mean                       geometry
## 1      5  8.61 3.748571 POINT Z (350654.8 6782701 8...
```

Part 4: Repeat Part 2 & 3 on your own with Plot 48 ALS and MLS data

For the next part of this lab I want you to repeat what I just went over with plot 40 but with plot 48 data with just the MLS point cloud, so you won't need to do the ALS cutouts. You can go about it in any way but produce the following:

1. **A plot of the plot 48 MLS point cloud**

2. **A plot of the normalized plot 48 MLS point cloud**

3. **A plot of the tree segmentation in the MLS point cloud and a justification of which algorithm and parameters used**

4. **A plot of a single tree from the MLS point cloud**

5. **A plot of a single tree voxelized in theMLS point cloud**

6. **The total number of voxels in the MLS**

7. **Finally: what was different about the forest vegetation of plot 48 vs. plot 40?**

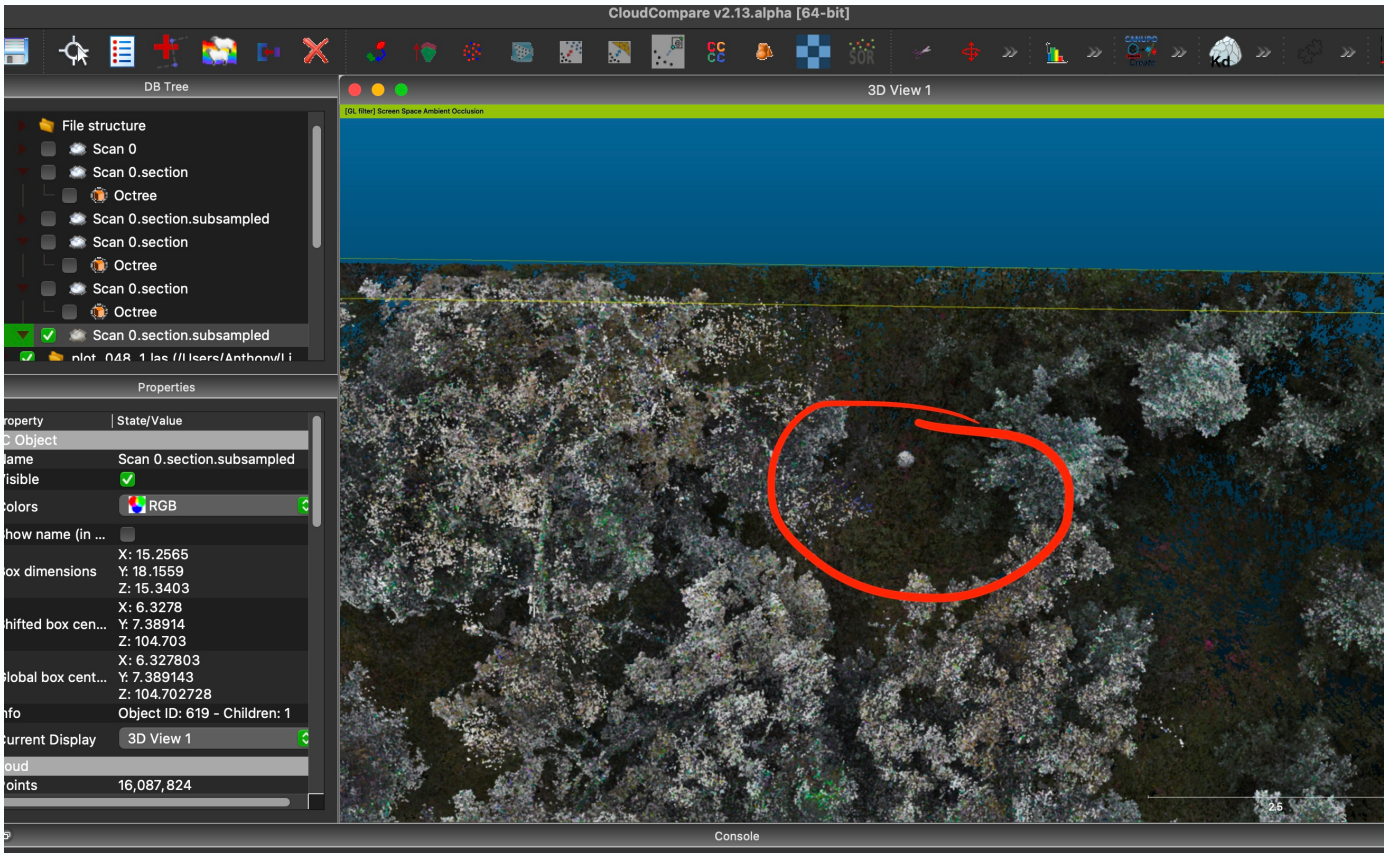
Part 5: Big clouds and small orbs in cloud compare

I've downloaded and filtered a bigger portion of the MLS point cloud used in this project. But we're not going to load it into R. Instead I want you to play around with it in Cloud Compare. Not totally playing around, but we do need to find a couple things in there.

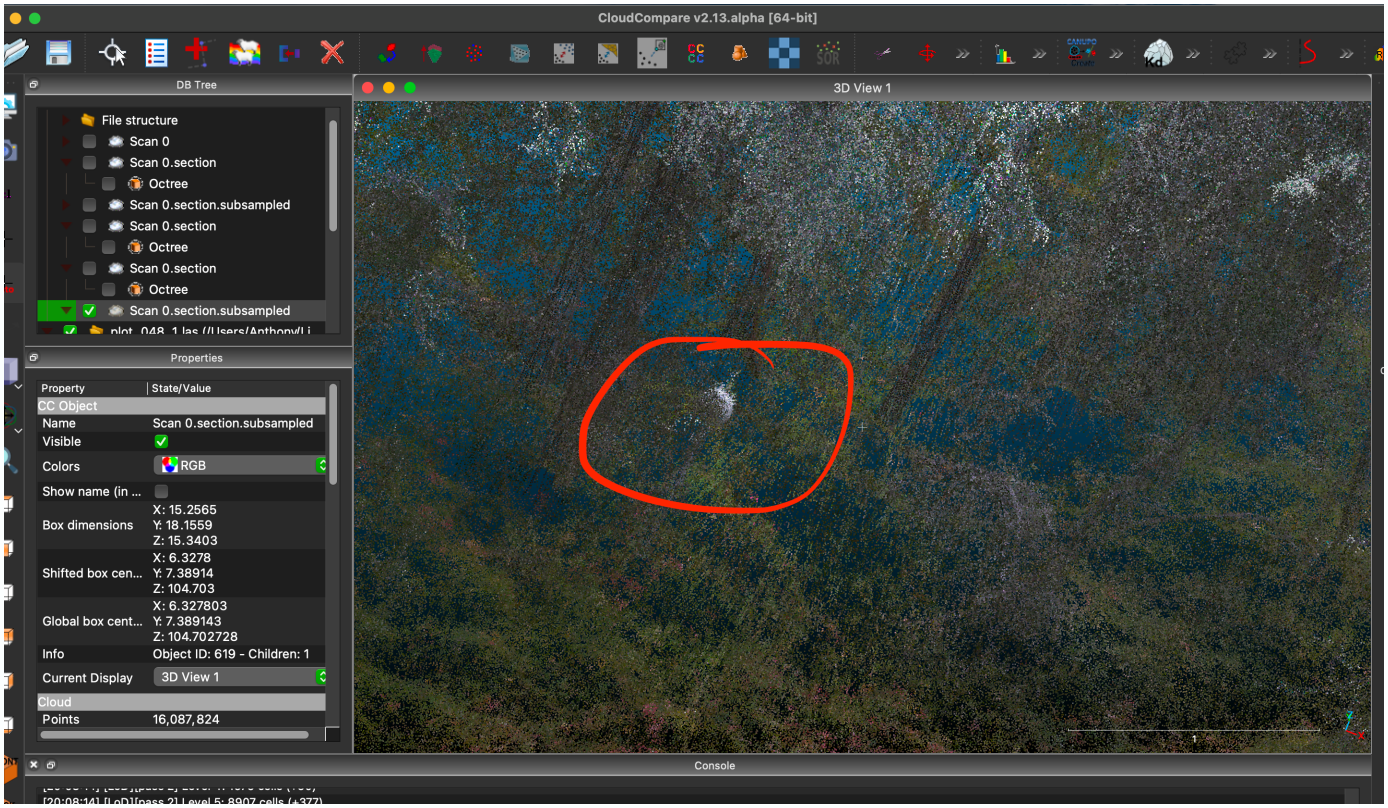
There are a couple orbs that are used for georeferencing point clouds which was talked about above. These have super precise geolocations from survey grade equipment and that helps pin down the location of the MLS point cloud since there is no projections or gps within them. Plus with this high of precision and quantity in laser pulses we need to accurately anchor our point clouds in order to get useful data.

The orbs are literally just white orbs

But finding them in the point cloud is tricky. The first one is fairly easy



But the second one will take a little more effort. *Hint: try using the Cross Section tool to reduce some vegetation.*



LAST SUBMISSION: Take screenshots of both orbs in the point cloud similar to the pics above

---

If you run into issues with R or cloud compare I can provide you either files or some alternatives. R has been a pain recently…
