Lidar & lidR Package Preliminary Exploratory Analysis Project

Lana Bibi

3/29/2022

Brief Overview

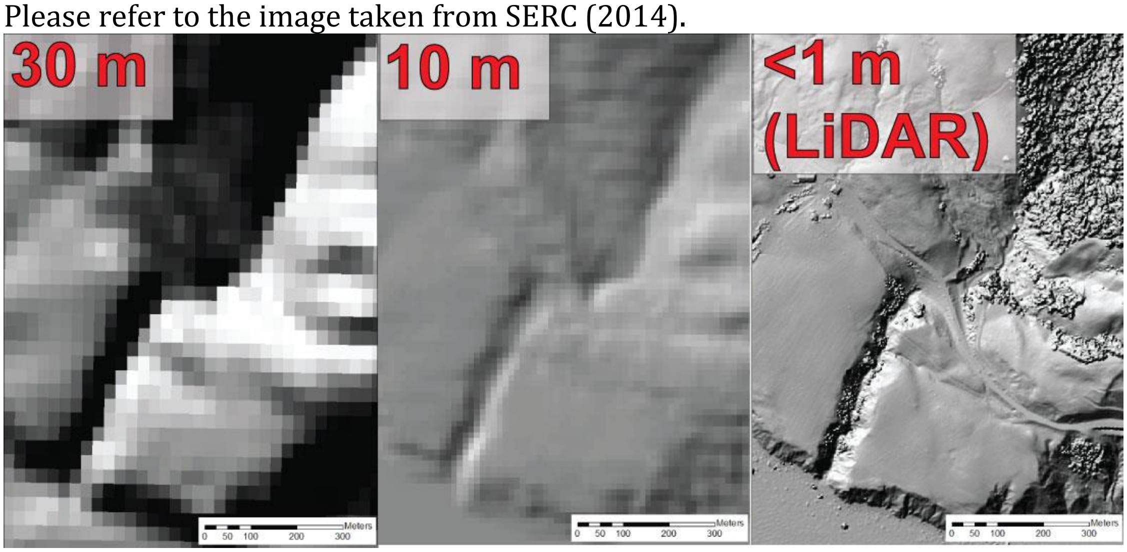
Hello! Welcome to the preliminary exploratory analysis of using Rstudio for LiDAR data.

For those who *don't* know. LiDAR data is known as light detection and ranging, it is a type of active remote sensing. Similar to other remotely sensed images, it can be collected via aerial, drone or satellite platforms.

In a nutshell, traditional maps you see are often passive remotely sensed data since it is easily accessible (i.e free, large-scale, variable locations and timescales, etc). However, LiDAR is becoming increasingly adopted as it collects information about earths surface across 3 axis (X, Y, Z), making it 3D.

FYI, passive remote sensing relies on sunlight and clear skies, active remote sensing relies on specific wavelengths (like microwaves) that can penetrate through clouds/ various weather conditions and isn't impacted by night

Please refer to the image taken from SERC (2014).



Benefits of LiDAR

Benefits of LiDAR that is increasing it's adoption is that (and not limited to);

1. That it is high resolution
2. Can collect data in various weather conditions

3. Provides critical information as it provides depth (especially in forestry applications)
4. Dense collection of data in a single collection

Issue regarding LiDAR is that it is not easily accessible due to it's unique advantages. Thus, open source data is minimal and processing techniques are currently limited (i.e often on paid software). Which is why my department wanted to explore it's potential processing via Rstudio.

Goals of the Project

My thesis and lab mates wanted to assess if the LiDAR data we collect from our in-lab made machines or open source files can be efficiently processed using free software to enhance our research. My thesis will be utilizing a ton of LiDAR data specifically in coastal ecosystems, which is why we wanted to assess the visualization and iteration capabilities of the LiDAR packages in Rstudio.

1. Assess if Rstudio can be used for LiDAR data, making it an optimal solution for accurate, efficient and free software
2. If the dedicated LiDAR packages can handle and possess all the qualitative and quantitative functions necessary for LiDAR analysis
3. The options for visualization, classification and filtration of LiDAR data features
4. If it is more time consuming than other softwares
5. If data and feature segregation can occur (/and if the steps to do so are more time consuming)
6. If open source LiDAR data is truly reliable

Rstudio Analysis

Packages for LiDAR data analysis are new developments and still with many bugs, missing functions, etc. The main package **lidR** was created in 2018 by Jean-Romain Roussel from Laval University. Till today he continuously works towards filling in the aforementioned gaps. The second specific package **rLIDAR** was created in 2021 by Silva, Crookston, Hudak, Vierling, Klauberg and Cardil from Moscow University.

A few tools and functions will be taken in this project to view the capabilities and limitations of using Rstudio for LiDAR data analysis. While trying to merge between the two.

To promote further push towards public data, data for this project **is obtained from the minimal opensource LiDAR resources**. Alberta is one of the few governments globally sharing their resources. Thus, data here is from Beaver Hills, Alberta.

to access their resources, kindly refer to this link: https://www.opendataareas.ca/#data

Workflow

Purpose & Authorship

Lana Bibi's (lana.bibi@ryerson.ca) project submission for ES8913

Packages used in this project

```
#Install Packages
#install.packages("lidR")
#install.packages("rLiDAR")
#install.packages("rgdal")
#install.packages("raster")
#install.packages("tmap")
#install.packages("tmaptools")
#install.packages("RStoolbox")
#install.packages("sf")
#install.packages("sp")
#install.packages("ggplot2")
#install.packages("rayshader")
#install.packages("mapview")
#install.packages("RMCC")
#install.packages("RCSF")

#Load packages
library("lidR")
library("rLiDAR")
library("rgdal")
library("raster")
library("tmap")
library("tmaptools")
library("RStoolbox")
library("sf")
library("sp")
library("ggplot2")
library("rayshader")
library("mapview")
library("RMCC")
library("RCSF")
```

Read in LiDAR Data and Wrangling

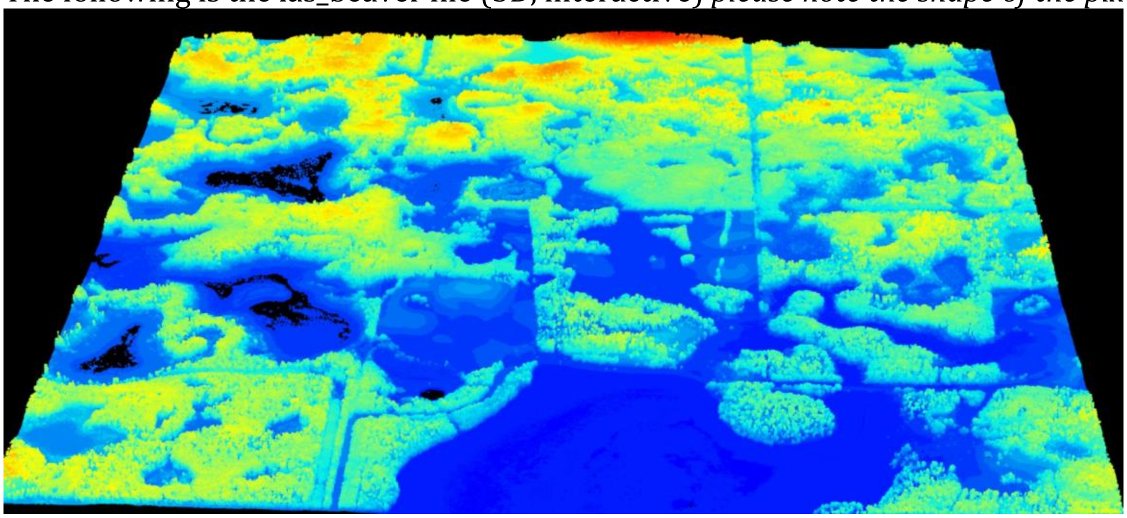
```
las_beaver <- readLAS(paste0(getwd(),"/point_cloud_las/pc_083H07SW38.las"))

#view data
summary(las_beaver)

# check coordinate system
epsg(las_beaver) # checks coordinate system via points, if undefined will
return zero
wkt(las_beaver)  # checks coordinate system based on string rather than
points, if undefined will be empty
```

```
# set coordinate system

st_crs(las_beaver) <-2955

las_beaver #check if it coordinate system is updated

# Determine if data is ready to be worked with, or if further wrangling
required

las_check(las_beaver)

# some functions require the variable format to be LAScatalog, not Las file.
So read in data in the two formats, for further processing.

# Read in file as LAScatalog (since there isn't a tool to convert it)

catalog_beaver <-
readLAScatalog(paste0(getwd(),"/point_cloud_las/pc_083H07SW38.las"))

st_crs(catalog_beaver) <- 2956 #set coordinate system

catalog_beaver #check
```

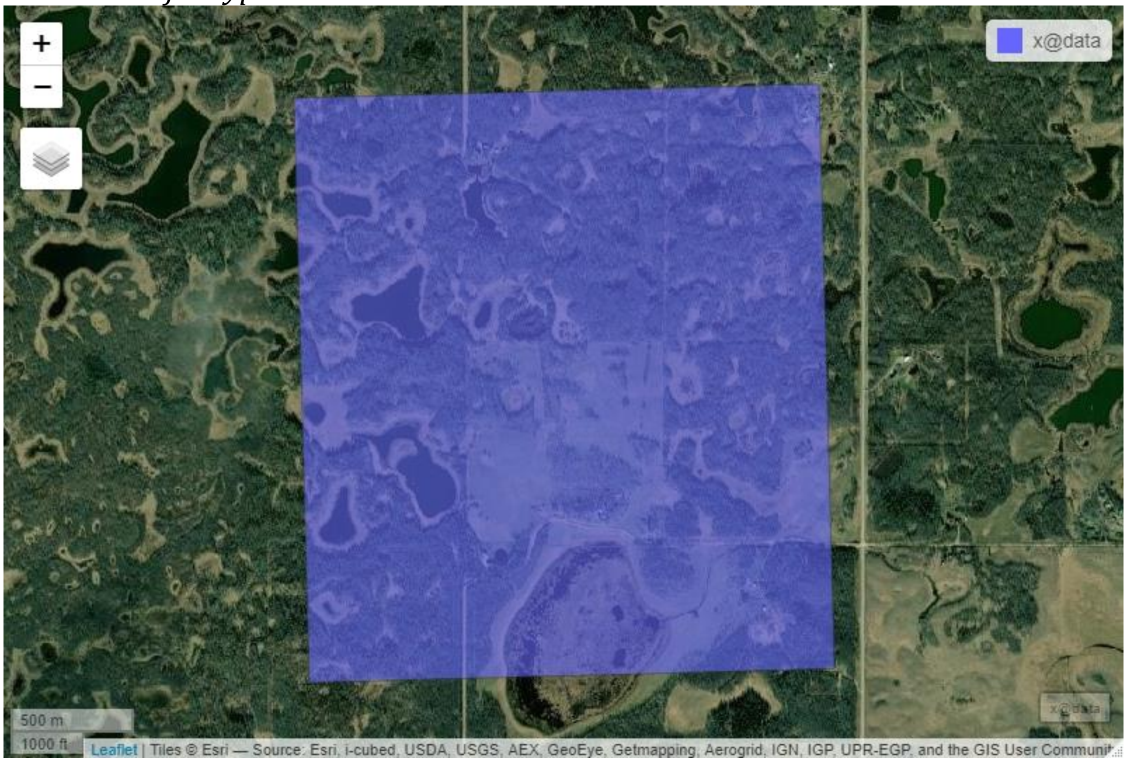
Simple Visualization

```
plot(las_beaver) #las class (default is 3D)
plot(catalog_beaver, mapview = TRUE, map.type = "Esri.WorldImagery")
#LAScatalog class(default 2D visualization)
```

The following is the las_beaver file (3D, interactive) *please note the shape of the pixels*

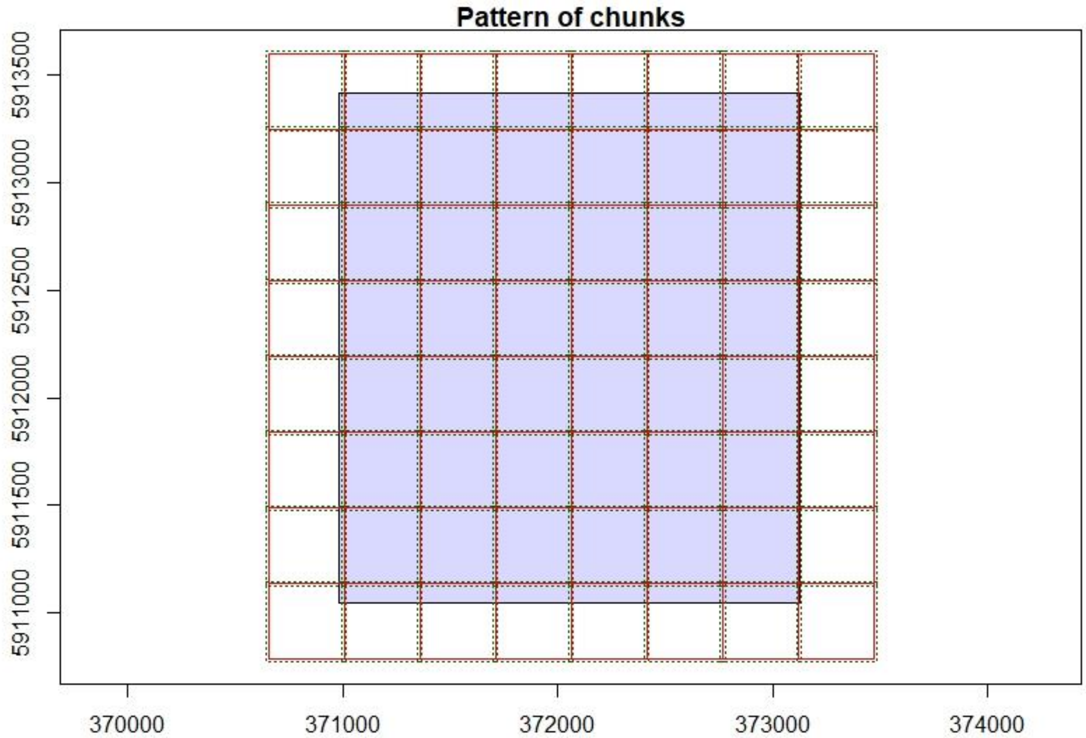


The following is the catalog_beaver file (2D and is overlayed on a map) *please note the variation in file types*

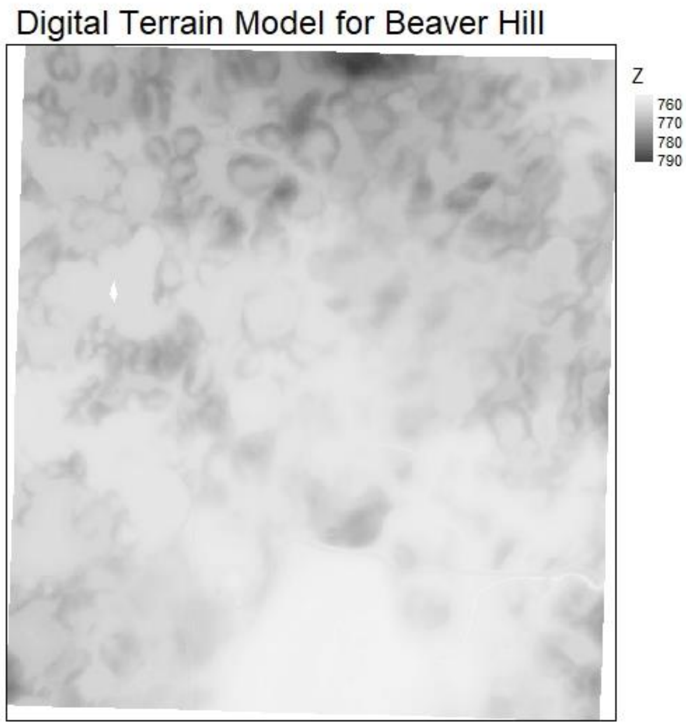


Digital Terrain Model (DTM)

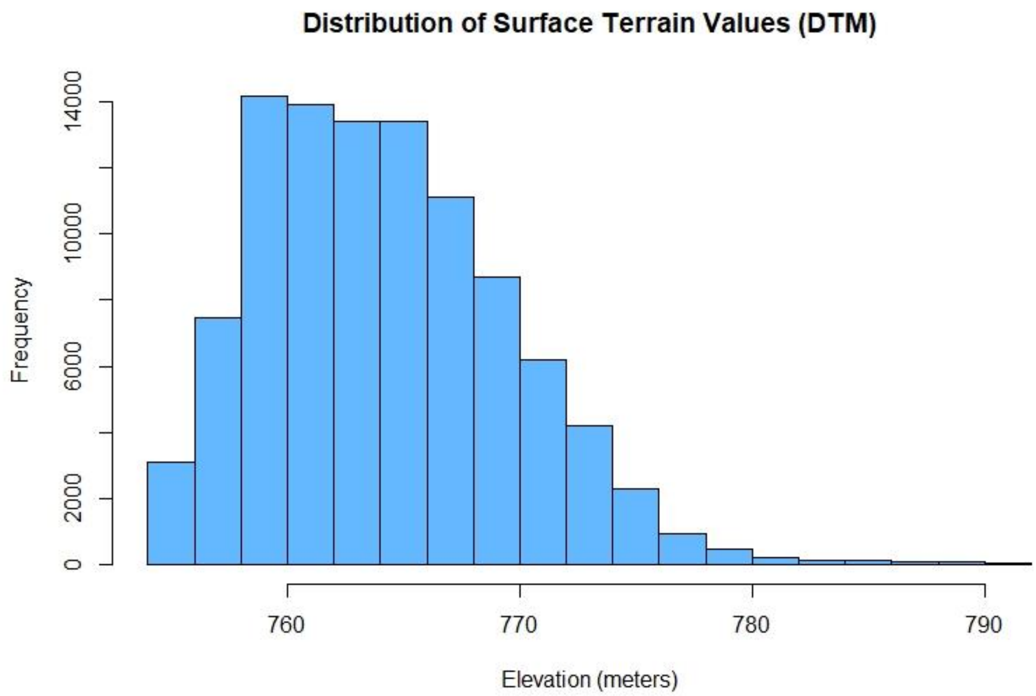
```
# Set Processing options

opt_chunk_size(catalog_beaver) <- 352   #352 x 352 meter tiles
plot(catalog_beaver, chunk_pattern = TRUE)

opt_chunk_buffer(catalog_beaver) <- 10 #overlap of 10 meter
plot(catalog_beaver, chunk_pattern = TRUE)

summary(catalog_beaver) #assess if processing extents set
```




```
# Create digital terrain model (DTM)

opt_output_files(catalog_beaver) <-
paste0(getwd(),"/las_out/dtm_{XLEFT}_{YBOTTOM}") #store as raster
DTM <- grid_terrain(catalog_beaver, res = 4, knnidw(k = 10, p = 2),
keep_lowest = FALSE, overwrite=TRUE) # cell-size set to 4x4 m, and k-nearest
neighbor inverse distance weighting with 10 neighbors and a power of 2

# Visualize DTM
tm_shape(DTM)+
  tm_raster(style= "cont", palette=get_brewer_pal("Greys", plot=FALSE))+
  tm_layout(main.title = "Digital Terrain Model for Beaver
Hill",legend.outside= TRUE)

writeRaster(DTM, paste0(getwd(),"/las_out/dtm.tif"))
```

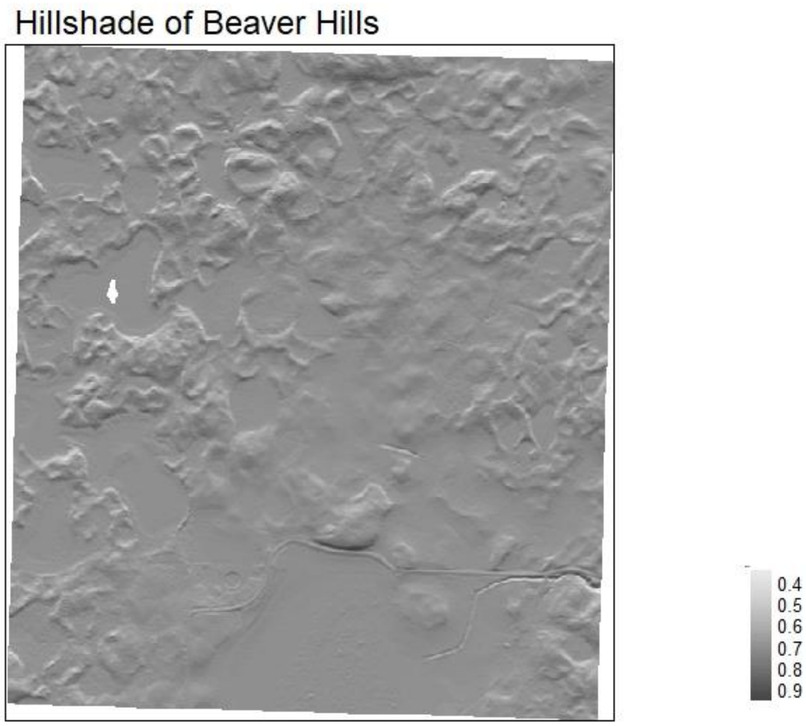



```r
#Histogram of Data Surface Elevation Distribution

DTM_distrubtion <- hist(DTM,
     main = "Distribution of Surface Terrain Values (DTM) Pixels",
     xlab = "Elevation (meters)", ylab = "Frequency",
     col = "steelblue1")
```

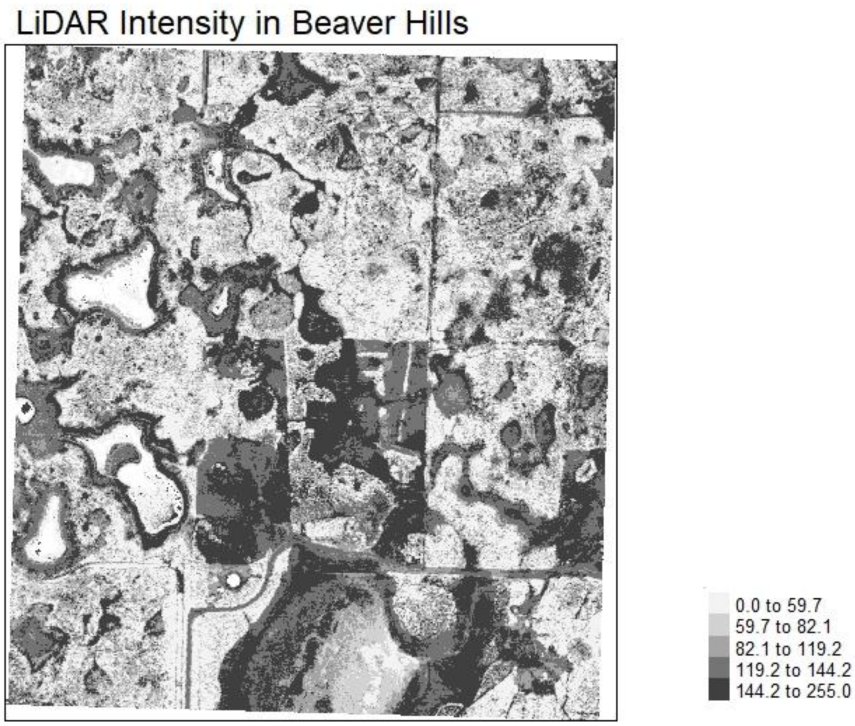


Hillshade & Intensity

```
# Create the components
slope <- terrain(DTM, opt='slope')
aspect <- terrain(DTM, opt='aspect')
hillshade <- hillShade(slope, aspect, angle=45, direction=180) #angle and
direction based on the dataset being located in Beaver Hills, Alberta

# Visualize hillshade
tm_shape(hillshade)+
  tm_raster(style= "cont", palette=get_brewer_pal("Greys", plot=FALSE))+
  tm_layout(main.title = "Hillshade of Beaver Hills", main.title.position =
"left",  legend.title.size=0.1 , legend.text.size = 0.8, legend.position=
c("center","bottom"), legend.outside = TRUE)
```




```r
# Set processing exents for intensity analysis
opt_output_files(catalog_beaver) <-
paste0(getwd(),"/lidar_points_out/intensity/int_{XLEFT}_{YBOTTOM}")
opt_filter(catalog_beaver) <- "-keep_first"
intensity <- grid_metrics(catalog_beaver, ~mean(Intensity), 5)

# Plot intensity
intensity[intensity<0]=0 # anything negative cannot be visualized and is
equates to zero reflectance regardless
tm_shape(intensity)+
  tm_raster(style= "quantile", n=5, palette=get_brewer_pal("Greys", n=5,
plot=FALSE))+
  tm_layout(main.title = "LiDAR Intensity in Beaver Hills",
main.title.position = "left",  legend.title.size=0.1 , legend.text.size =
0.8, legend.position= c("center","bottom"), legend.outside = TRUE)
```

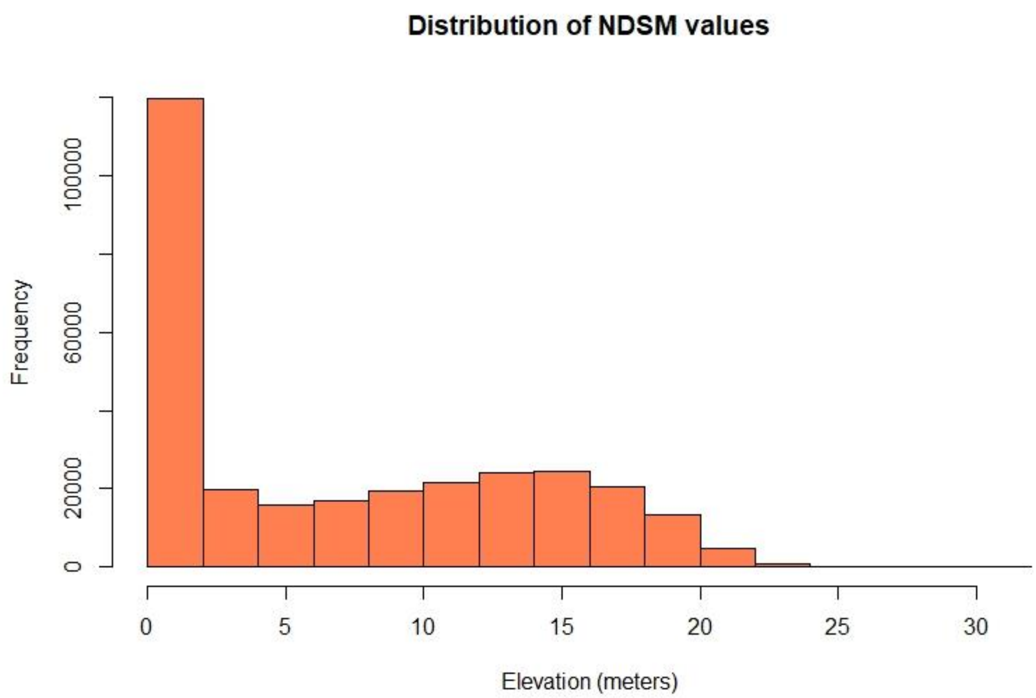


Normalized Digital Surface Model

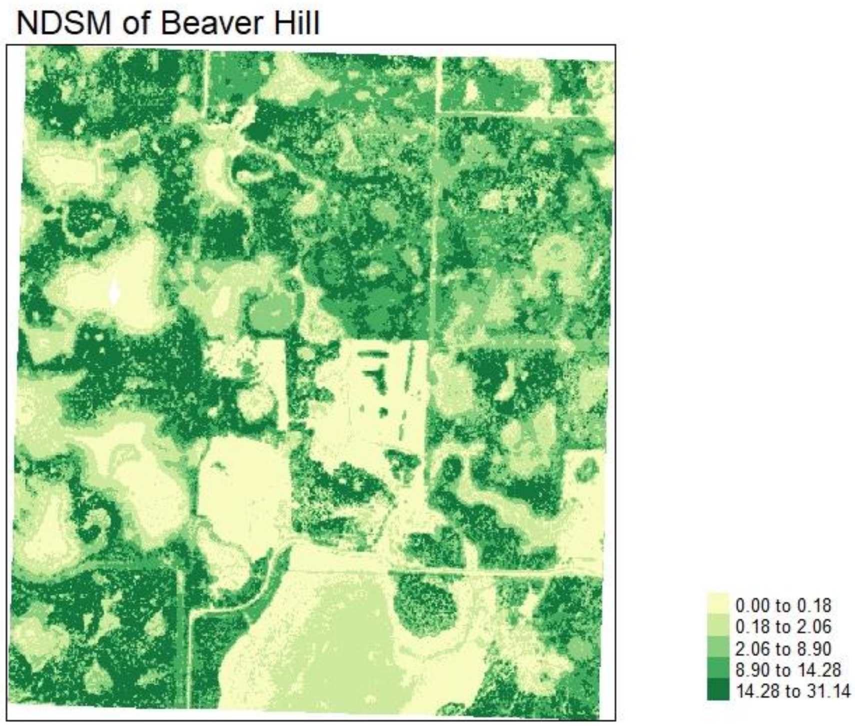
Normalized digital surface model is used as it can separate between above ground and ground features. Often a prerequiste for canopy height model.

```
# Identify ground features

opt_output_files(catalog_beaver) <-
paste0(getwd(),"/lidar_points_out/normalized/norm_{XLEFT}_{YBOTTOM}") #where
files will be saved
normalized_beaver <- normalize_height(catalog_beaver, DTM) # spatial
interpolation of features


# Identify above ground features (both built and natural) - Digital Surface
Model

opt_output_files(catalog_beaver) <-
paste0(getwd(),"/lidar_points_out/surface_model/dsm_{XLEFT}_{YBOTTOM}")
#where files will be saved
DSM <- grid_canopy(catalog_beaver, res = 4, pitfree(c(0,2,5,10,15), c(0, 1)))
#cell-size set to 4x4 m, and pitfree algorithim to smooth between features
writeRaster(DSM, paste0(getwd(),"/lidar_points_out/DSM.tif"), overwrite=
TRUE)

# Calculate a Normalized Digital Surface Model
```

```
NDSM <- DSM - DTM
NDSM[NDSM<0]=0
NDSM

# Histogram of  normalized digital surface model

NDSM_distrubtion <- hist(NDSM,
                         main = "Distribution of NDSM values",
                         xlab = "Elevation (meters)", ylab = "Frequency",
                         col = "coral")
```



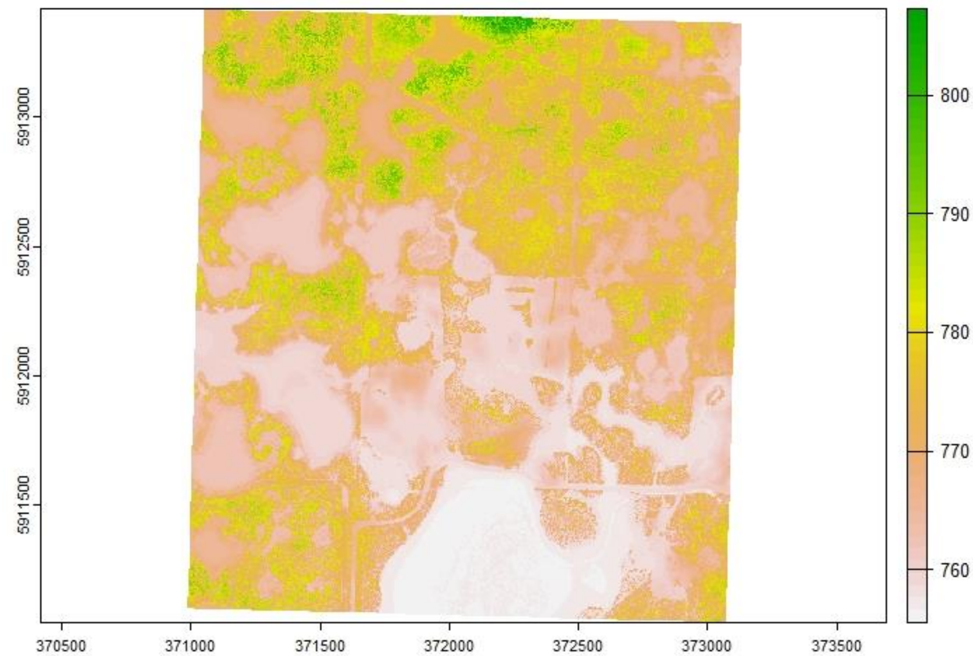

```
# Visualize the Normalized Digital Surface Model
tm_shape(NDSM)+
  tm_raster(style= "quantile", n=5, palette=get_brewer_pal("YlGn", n=5,
plot=FALSE))+ # number of classes is 5 and using a quantile

  tm_layout(main.title= "NDSM of Beaver Hill", main.title.position = "left",
legend.title.size=0.1 , legend.text.size = 0.8, legend.position=
c("center","bottom"), legend.outside = TRUE)
```



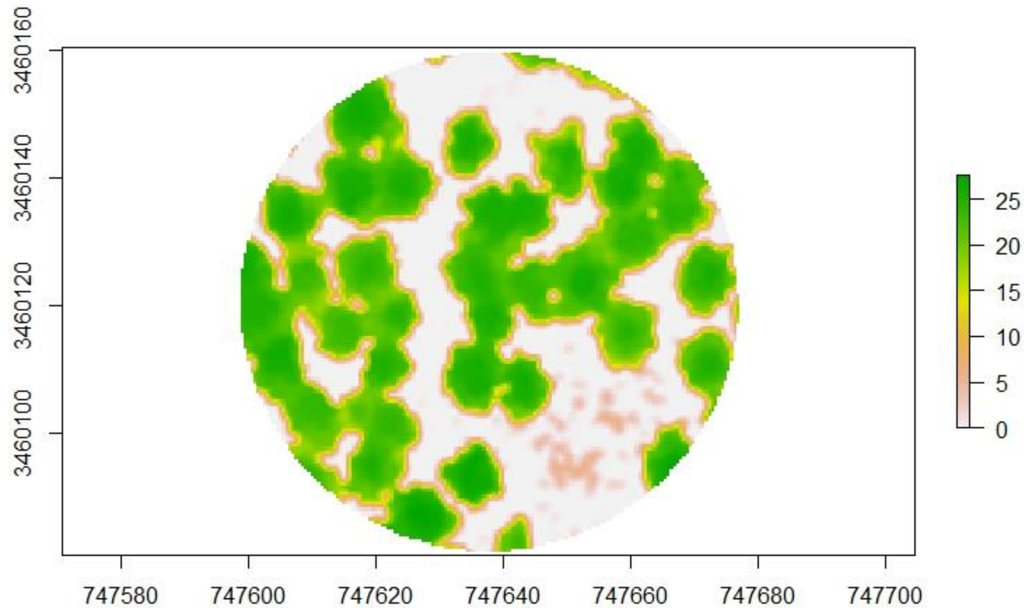
Canopy Height Model

Looks specifically at treetops. Thus, not only the above ground features, but the highest point in the scene.

```
# Khosravipour et al. pitfree algorithm

thr <- c(0,2,5,10,15)
edg <- c(0, 1.5)
chm <- rasterize_canopy(las_beaver, 1, pitfree(thr, edg))

plot(chm)
```

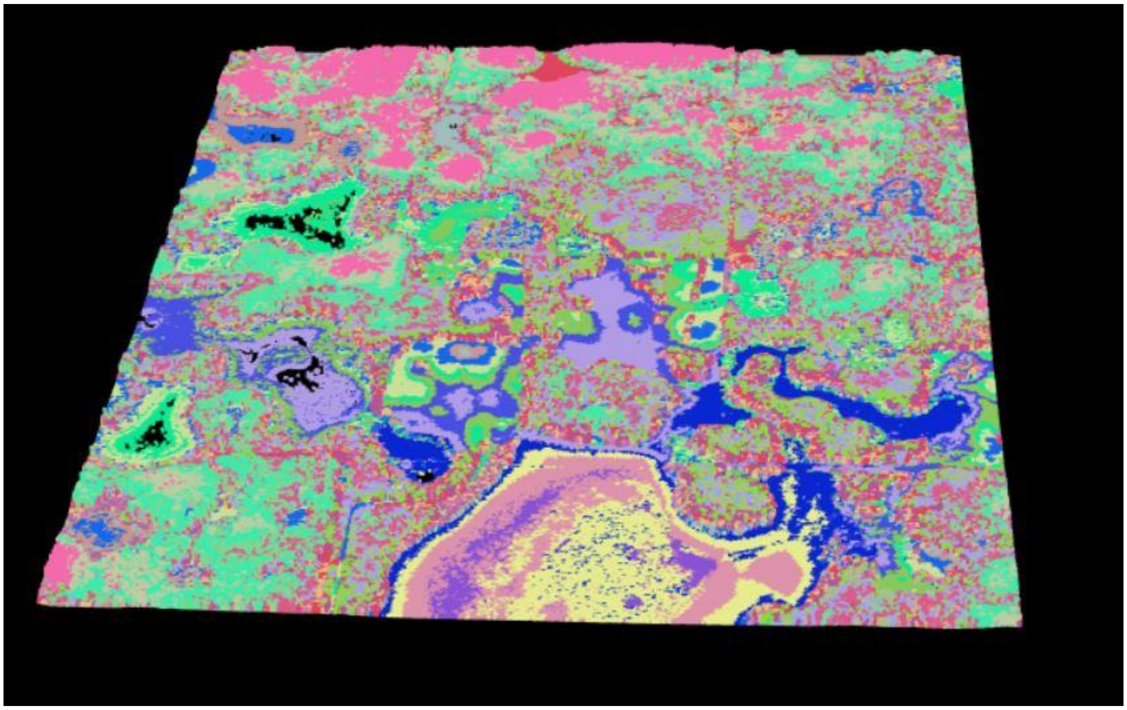



```r
# remove spurious effects of tree branches curtsey of Silva (2018)

data(chm) #changes format from spat raster to raster
chm_smooth <- CHMsmoothing(chm, "mean", 3, 0.67)

plot(chm_smooth)
```



This crashed the system but with the right server, when it ran, it gives you the individual classes/preliminary classification, which is highly beneficial. Should be carried out if possible.

```
# remove spurious effects of tree branches curtsey of Silva (2018)

data(chm) #changes format from spat raster to raster
chm_smooth <- CHMsmoothing(chm, "mean", 3, 0.67)

plot(chm_smooth)
```

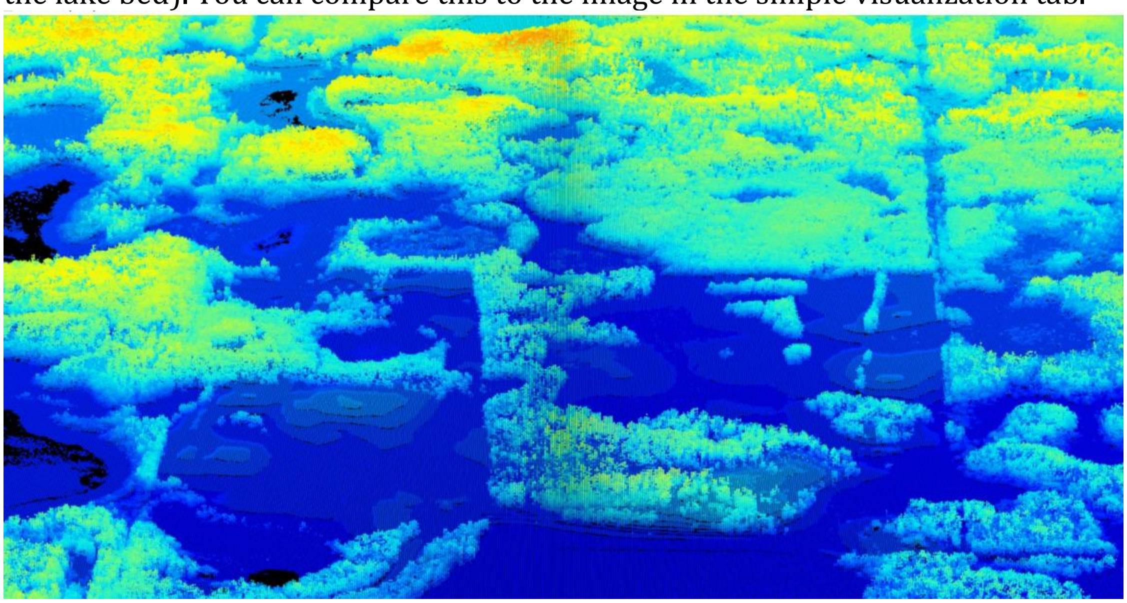


Voxelization

Voxelization of pixels.

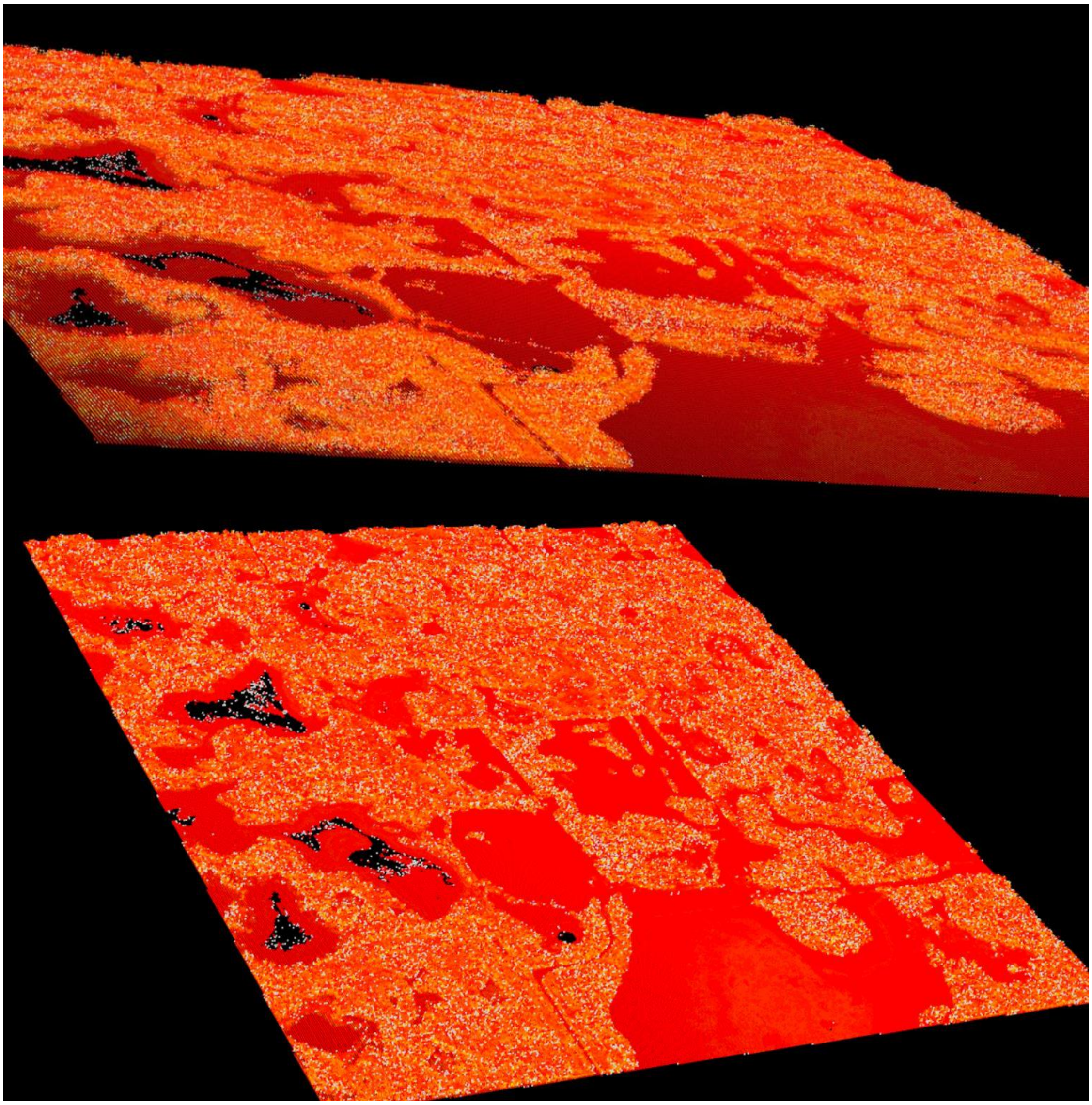
```
#3D voxel points
voxelize_p_beaver <- voxelize_points(las_beaver, 2)
plot(voxelize_p_beaver)
```

Notice that pixel shape has altered based on the feature, and the details provided (such as the lake bed). You can compare this to the image in the simple visualization tab.

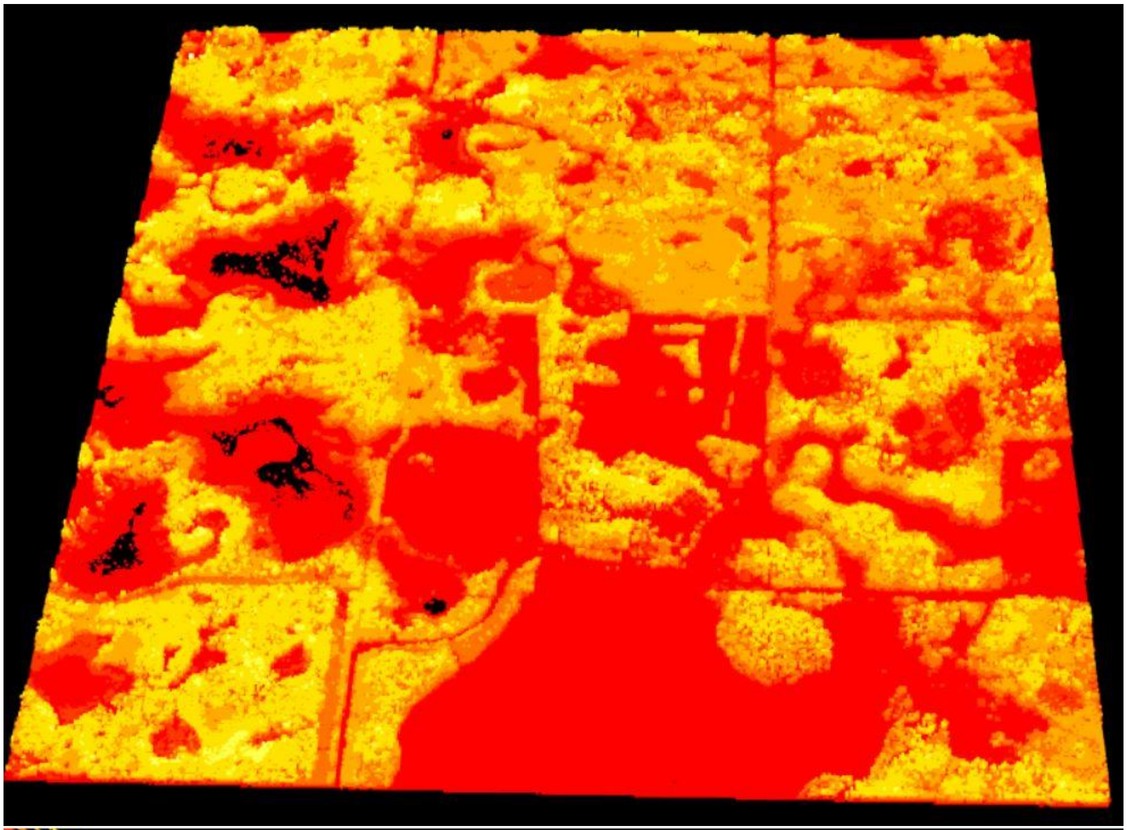


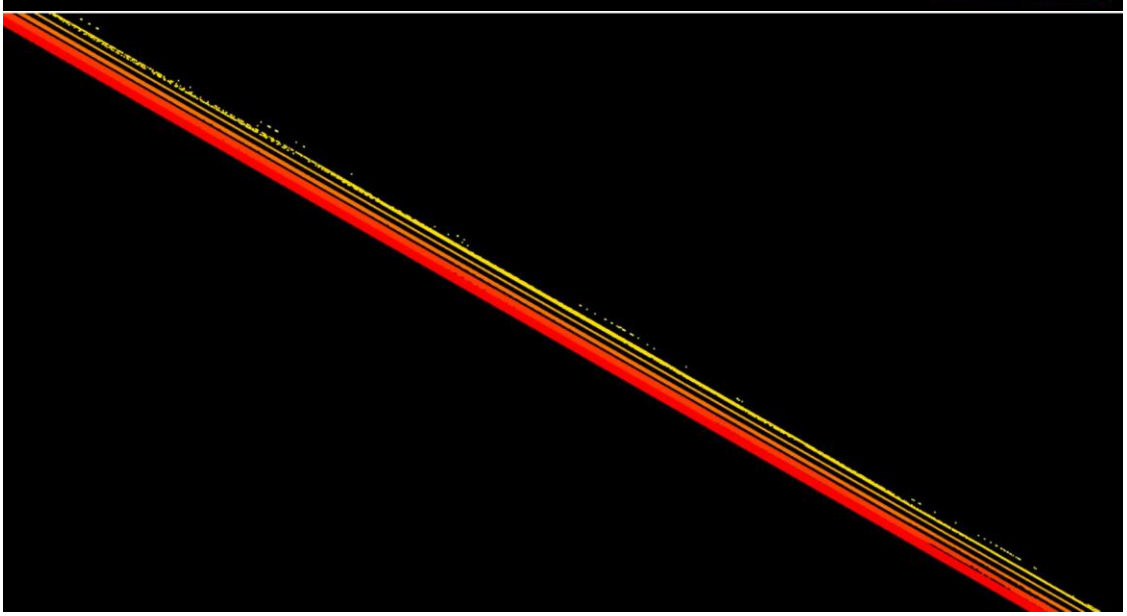
Voxelization of objects rather than pixels.

```
# 3D voxel of objects

las1_dtm <- grid_terrain(las_beaver, res = 2, knnidw(k = 10, p = 2),
keep_lowest = FALSE)
las1_n <- normalize_height(las_beaver, las1_dtm)
voxelize_beaver <- voxel_metrics(las1_n, ~sd(Z), res = 5)
voxelize_beaver

plot(voxelize_beaver, color = "V1", pal = heat.colors(60), mean)
#visualization based on intensity in pixels
```



```
plot(voxelize_beaver, color = "Z", pal = heat.colors(60), mean)
#visualization based on intensity in elevation
```



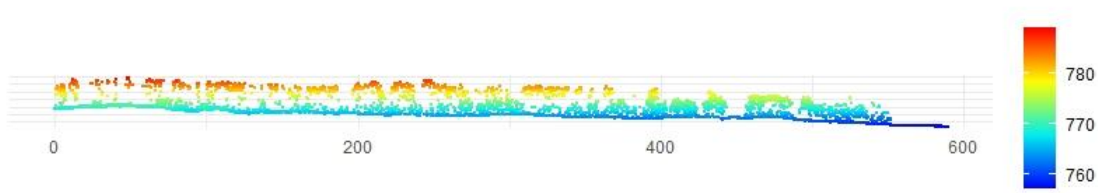


Transects

Horizontal transects shows variation along the X and Y axis.

```
#Create a horizontal transect
point1 <- c(371123, 5911101) #select first point
point2 <- c(371760, 5911101) # select second point
transect_h <- clip_transect(las_beaver, point1, point2, width = 5, xz = TRUE)
```

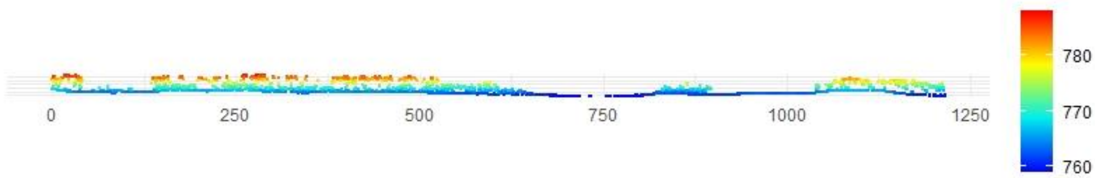
```
#make the transect

# Visualize horizontal transect
ggplot(transect_h@data, aes(X,Z, color = Z)) +
  geom_point(size = 0.5)  +
  coord_equal()  +
  theme_minimal() +
  theme (legend.title=element_blank(),
         axis.title.y=element_blank(),
         axis.title.x=element_blank(),
          axis.text.y=element_blank()
    ) +
  scale_color_gradientn(colours = height.colors(50))
```



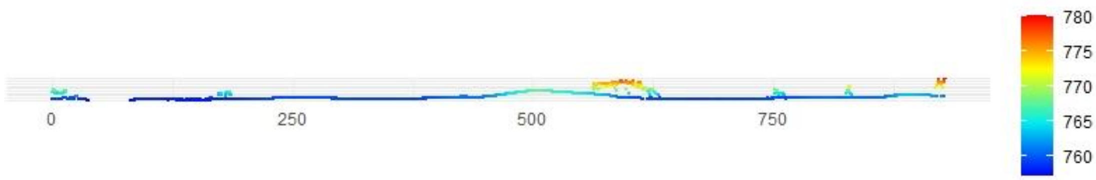
Vertical transects shows variation along, X, Y and Z axis.

```
# Create a vertical transect

point3 <- c(371270, 5911100) #select third point
point4<- c(371770, 5911477)  #select fourth point
point5<- c(371789, 5912200) #check out variation with this point if
interested
transect_v <- clip_transect(las_beaver, point3, point5, width = 2, xz = TRUE)

# Visualize vertical transect
ggplot(transect_v@data, aes(X,Z, color = Z)) +
  geom_point(size = 0.2)  +
```

```
  coord_equal()  +
  theme_minimal() +
 theme (legend.title=element_blank(),
        axis.title.y=element_blank(),
        axis.title.x=element_blank(),
         axis.text.y=element_blank()
    ) +
  scale_color_gradientn(colours = height.colors(100))
```

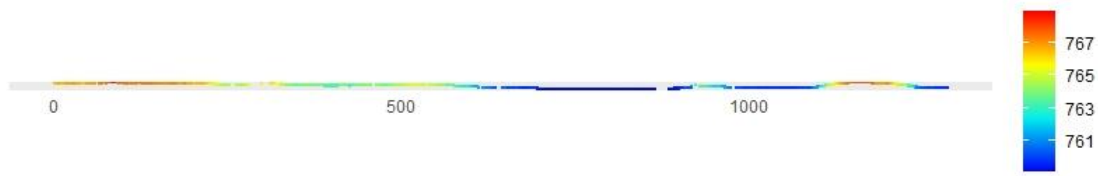


Another example;

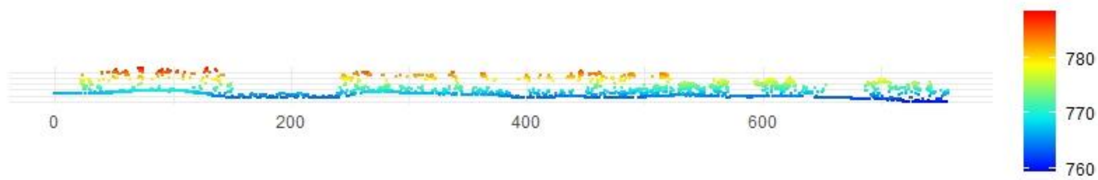


Ground vs all features Filtering via Transect

```
#Set points for transect
pointA <- c(371123, 5911100)
pointB<- c(371789, 5912200)


#Filter ground points

ground_points <- filter_ground(las_beaver)
ground_transect <- clip_transect(ground_points, pointA, pointB, width = 4, xz
= TRUE)

#Visualize ground points on transect
ggplot(ground_transect@data, aes(X,Z, color = Z)) +
  geom_point(size = 0.5) +
  coord_equal() +
  theme_minimal() +
  theme (legend.title=element_blank(),
         axis.title.y=element_blank(),
         axis.title.x=element_blank(),
         axis.text.y=element_blank()
  ) +
  scale_color_gradientn(colours = height.colors(50))
```



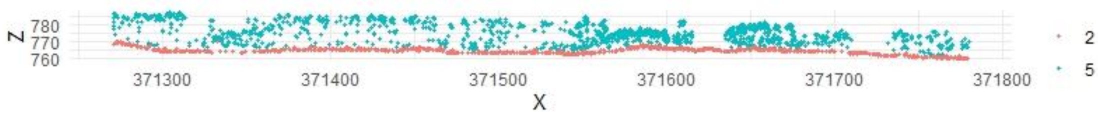

```r
# All features on transect
ggplot(transect@data, aes(X,Z, color = Z)) +
  geom_point(size = 0.5) +
  coord_equal() +
  theme_minimal() +
  theme (legend.title=element_blank(),
         axis.title.y=element_blank(),
         axis.title.x=element_blank(),
         axis.text.y=element_blank()
  ) +
  scale_color_gradientn(colours = height.colors(50))
```



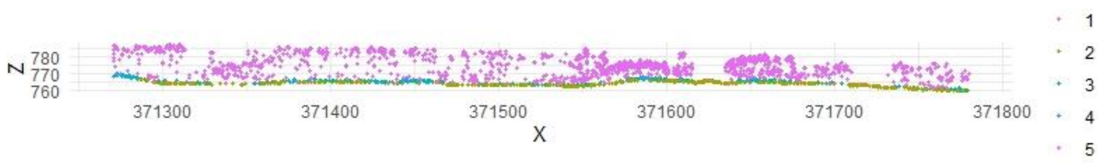
Comparison of Classification of Points via Transect

```
# Set function created by Jean-Romain Roussel (Laval University, Quebec, CA)
as it is the current only way to combine transect location and plotting

plot_crossection <- function(las_class,
                             p1 = c(min(las_class@data$X),
mean(las_class@data$Y)),
                             p2 = c(max(las_class@data$X),
mean(las_class@data$Y)),
                             width = 2, colour_by = NULL)
{
  colour_by <- enquo(colour_by)
  data_clip <- clip_transect(las_class, p1, p2, width)
  p <- ggplot(data_clip@data, aes(X,Z)) + geom_point(size = .7) +
coord_equal() + theme_minimal()

  if (!is.null(colour_by))
    p <- p + aes(color = !!colour_by) + labs(color = "")

  return(p)
}

# Using Progressive Morphological Filter (PMF) created by Zhang et al. (2003)

las_class_A <- classify_ground(las_beaver, algorithm = pmf(ws = 4, th = 2))
```

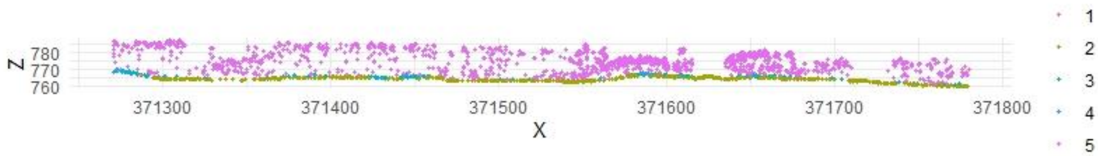
```
plot_crossection(las_class_A, point3 , point4, colour_by =
factor(Classification))
```




```
# Using Multiscale Curvature Classification (MCC) created by Evans & Hudak
(2017)

las_class_B <- classify_ground(las_beaver, algorithm = mcc(s = 4, t = 0.3))


plot_crossection(las_class_B, point3 , point4, colour_by =
factor(Classification))
```




```
# Using Cloth Simulation Filter (CSF) created by Zhang et al. (2016)

las_class_C <- classify_ground(las_beaver, algorithm = csf(TRUE, 0.5, 1,
time_step = 0.65))


plot_crossection(las_class_C, point3 , point4, colour_by =
factor(Classification))
```



After Thoughts

Limitations Encountered

1. Many bugs between **converting between frequent lidar data structures** i.e switching between LAS and LAScatalog doesn't exist unless it is read into it.
2. Classification algorithms in any package, do not clearly provide an accuracy assessment. If created as a function, will glitch or misread data sometimes, thus still not accurate.
3. Mainly visual/qualitative interpretation is required, rather than quantitative
4. Requires robust computing if using iteration (i.e for loops) on multiple scenes
5. Waveformlidar package provides increasingly beneficial tools, but not all cloud points provide waveform files to be read
6. Functions can glitch and deteriorate spontaneously
7. Package isn't updated, so some functions are removed or outdated
8. Interactive mapping doesn't allow for mapping or plotting features
9. clipping specific features is not as simple or easy as other softwares
10. Transect specific functions are limited, and thus quantitative analysis can be limited
11. Limited functions to fix orientation of point cloud or interpolate for data gaps (those created by flight path issues, platform glitch, etc)

Benefits of Using the software

1. Continuous improvement in packages

2. With the right server and computing power, efficient data analysis using iteration (in comparison to other software's that process one at a time)
3. Visualization is automatic and easier than other software (minimal steps, and several algorithms)
4. Handles 3D interactive mapping
5. Can extract histograms and pixel algorithms quickly
6. One of the few free and accessible software to handle LiDAR data
7. More technical methods for classification of pixels
8. Visualization is more intricate than other current free softwares (i.e can shape individual trees, categorize by pulse)
9. Geolocation transformation and hyper point cloud capacities
10. 2D and 3D analysis options for every function
11. Transect extraction, filtration processes, layer dissemination are time efficient and highly accurate

For more Information:

Please read the research paper. To access files and R processing, github repository can be access here: https://github.com/lanajbb/ES8913_lidR_project

References:

Gomez, C. (2020, December 30). Lidar data with the lidR package - 1. https://rstudio-pubs-static.s3.amazonaws.com/708624_40f54f88124b4b6eb214006a3a53a691.html

Luo, W., & Zhang, H. (2015). Visual analysis of large-scale LiDAR point clouds. Proceedings - 2015 IEEE International Conference on Big Data, IEEE Big Data 2015, 2487–2492. https://doi.org/10.1109/BIGDATA.2015.7364044

Roussel, J. R., Auty, D., Coops, N. C., Tompalski, P., Goodbody, T. R. H., Meador, A. S., Bourdon, J. F., de Boissieu, F., & Achim, A. (2020). lidR: An R package for analysis of Airborne Laser Scanning (ALS) data. Remote Sensing of Environment, 251. https://doi.org/10.1016/J.RSE.2020.112061

SERC. (2014, July 15). Figure 2. Satellite Imagery versus LiDAR. https://serc.carleton.edu/details/images/47158.html

Silva, C. A., Hudak, A. T., Vierling, L. A., Valbuena, R., Cardil, A., Mohan, M., Almeida, D. R. A., Broadbent, E. N., Almeyda Zambrano, A. M., Wilkinson, B., Sharma, A., Drake, J. B., Medley, P. B., Vogel, J. G., Prata, G. A., Atkins, J. W., Hamamura, C., Johnson, D. J., & Klauberg, C. (2022). treetop : A Shiny-based application and R package for extracting forest information from LiDAR data for ecologists and conservationists . Methods in Ecology and Evolution. https://doi.org/10.1111/2041-210X.13830

Silva, C. A., Klauberg, C., & Mohan, M. M. (2018). LiDAR Analysis in R and rLiDAR for Forestry Applications rGEDI: An R Package for NASA's Global Ecosystem Dynamics

Investigation (GEDI) Data Visualization and Processing. View project GEOAGRO-Geoprocessamento e Sensoriamento Remoto Aplicados a Sistemas Agroecológicos e Mapeamento da Cobertura Vegetal e Uso da Terra (MCVUT) View project. https://www.researchgate.net/publication/324437694

Wasser, L., & Jones, M. (2021, May 13). Data Activity: Visualize Elevation Change using LiDAR in R to Better Understand | NSF NEON | Open Data to Understand our Ecosystems. https://www.neonscience.org/resources/learning-hub/tutorials/da-viz-neon-lidar-co13flood-r

Zhou, T., & Popescu, S. (2019). remote sensing waveformlidar: An R Package for Waveform LiDAR Processing and Analysis. MDPI. https://doi.org/10.3390/rs11212552