ESRM433/SEFS533
Lab 6

Objectives:

- Rumple index and canopy height models
- Digital Aerial Photogrammetry (DAP)
- Linear regression analysis with rasters
- Creating forest characterization maps

Data and Software:

- LAB6Data.zip folder, downloadable from Canvas
- Data from lab 5
- Downloads from Washington DNR lidar Portal
- RStudio
- CloudCompare

What you will turn in:

- *Submission Template in PDF or DOCX file format submitted via Canvas*

Welcome to Lab 6 for ESRM433/SEFS533!

In this lab we will be comparing Digital Aerial Photogrammetry point clouds with ALS. We will be creating a rumple index with both sets of data, look for relationships between field data and our rumple index, and then extrapolating our metrics out across an area.

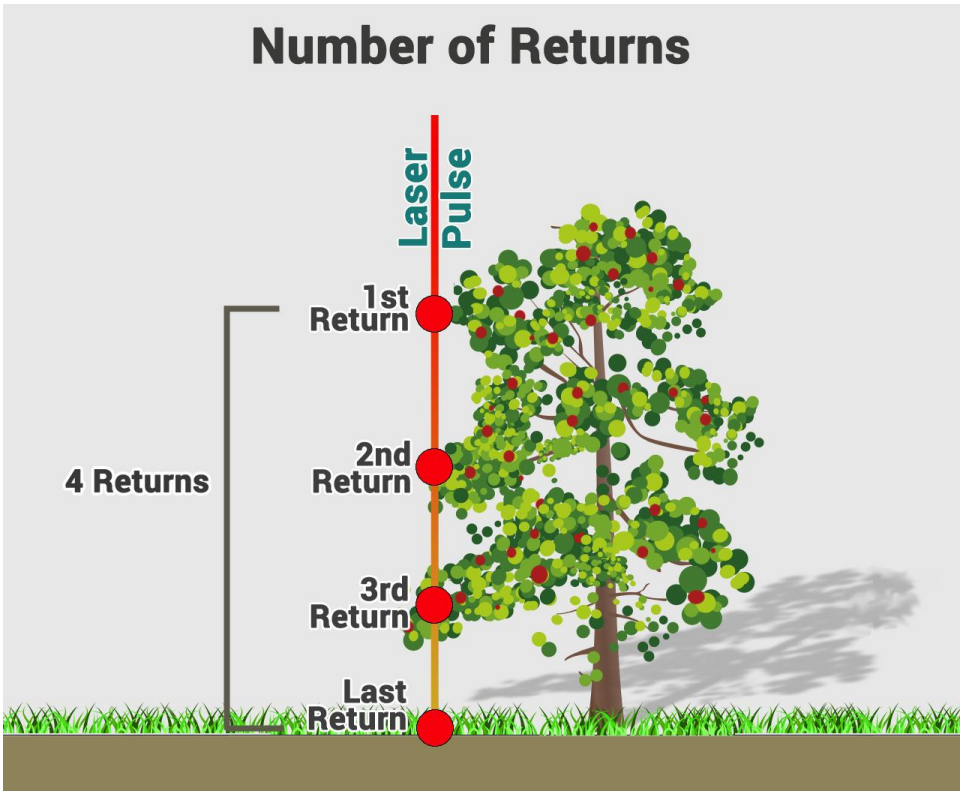
Lastly, you will be given a task to identify forests with specific characteristics in Pack Forest.

INTRO: Ecological meaning of lidar metrics

Mean, Max, 95th percentile, & 25th percentile are often used lidar metrics in ecology. We can make a few general statements about these metrics and how they relate to forest structure.

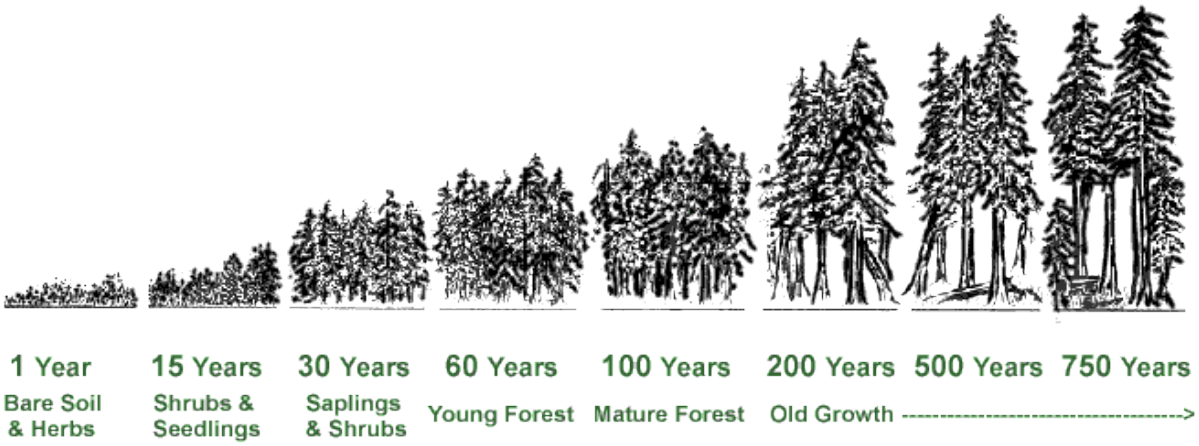
There are many reasons why these generalizations won't always be true, but hopefully this will aid in visualizing why we use these metrics.

The number and locations of returns depends on the height and density of forest canopy. Some pulses may have more or less than 4 returns. If there are many pulses that hit this tree with similar locations, the mean values would be between the 2nd and 3rd returns, the max would be the

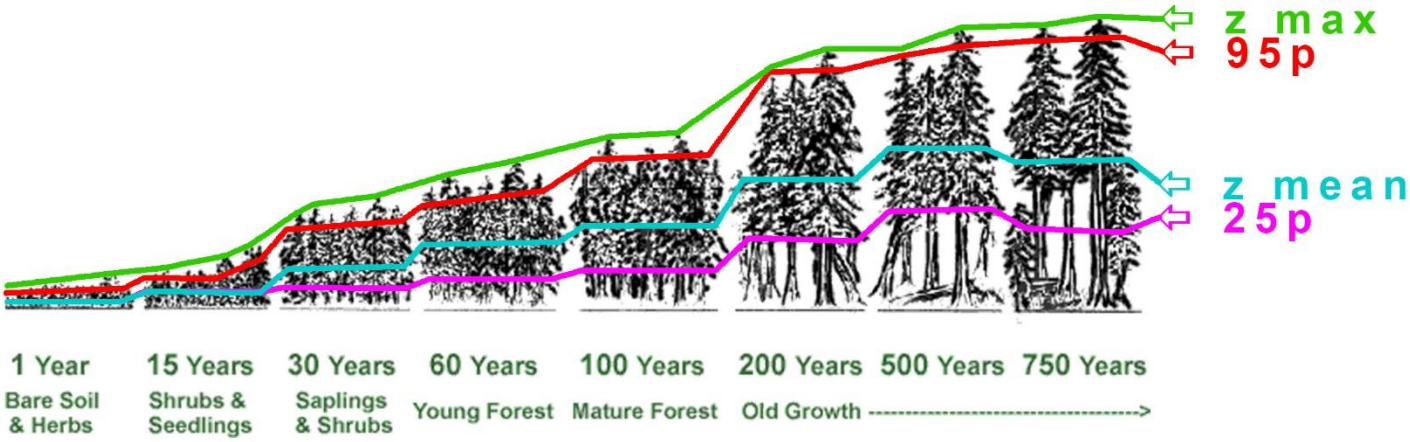


1st returns, the 95th percentile would be just below the 1st returns, and the 25th percentile would be around the location of the 3rd return.

To generalize about forest age and structure and percentiles, let's look at forest structure at different successional stages:



A conceptual diagram of where our lidar metrics would be located at the different successional stages:



There are direct measurements from lidar. Z max potentially is the tallest tree in an area, however, noise in the data might make z max unreliable so 95p may be reasonable to take as a direct measurement of tallest trees.
Other metrics like z mean and 25p aren't necessarily a direct measurement of forest structure but they can provide a lot of information about the general structure of a forest. 25p often correlates with the average height to crown base in a forest.
Other forest metrics like Trees Per Acre (TPA) are not directly measured by lidar but relationships can be found. In our example above, there would be many more TPA in the younger forests. Younger forests tend to have a lower 25p and zmean, therefore, 25p and zmean in a multiple linear regression may be able to model TPA while not being a direct measurement. A relationship in one forest type likely won't be the same in a different forest type. Some generalized metrics may be possible, but new models should be developed for different forests.

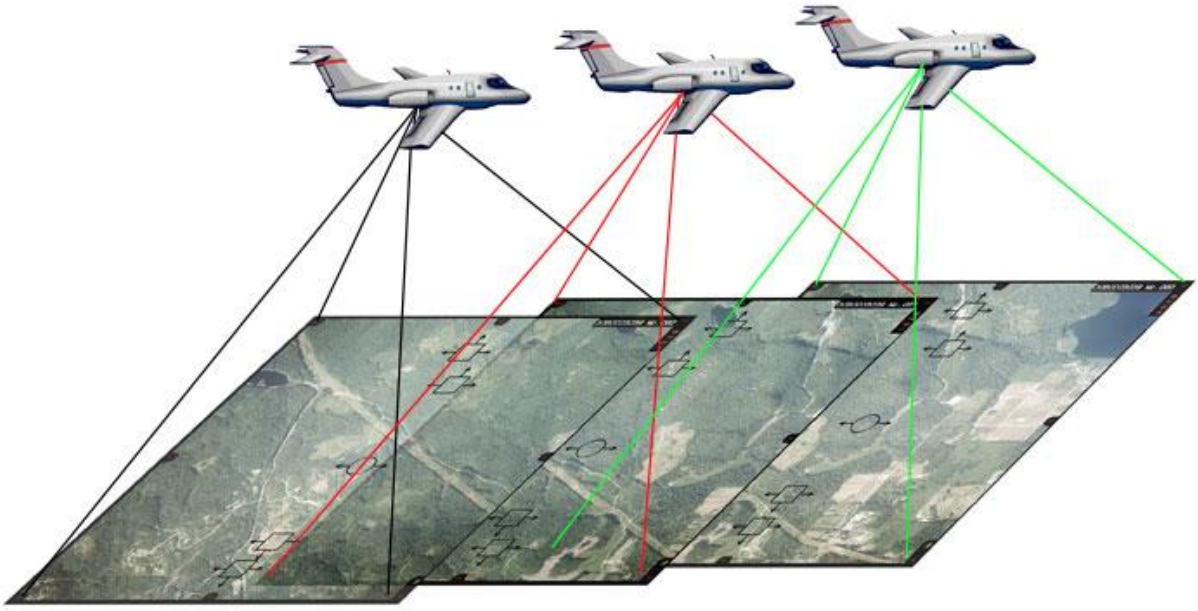
ESRM433/SEFS533

Lab 6

Digital Aerial Photogrammetry (DAP)

Unfortunately, DAP isn't readily available off the internet like lidar data is. Fortunately, you have friends in SEFS that can provide you with some data. To understand the basic principle of DAP take a moment to watch this video:

https://youtu.be/suo_aUTUpps



The video focuses on Googles creation of 3D buildings and topography but the basics apply to the data that we are working with today.

The DAP point cloud that we are working with was derived from National Agriculture Imagery Program (NAIP).

https://www.usgs.gov/centers/eros/science/usgs-eros-archive-aerial-photography-national-agriculture-imagery-program-naip

NAIP imagery is readily available for download, but the imagery that is made public is seamless orthographically corrected imagery. The images are basically merged into one orthomosaic where distortion from the oblique angle at the edges of images, and differences in topography are removed.

https://en.wikipedia.org/wiki/Orthophoto

To create a DAP point cloud, you need those raw images as those oblique angles and differences due to topography are what allow for structure to be modeled. DAP is often called Structure from Motion. As you can imagine, getting a point cloud from images can be very helpful in forest monitoring. DAP can be done with images from airplanes or from drones. Drones will generate much higher resolution images (<3cm/px) while NAIP imagery has significantly lower spatial resolution (1m to 0.5m). However, the nation-wide coverage with NAIP imagery every year or two makes it a fantastic resource for resampling forested areas.

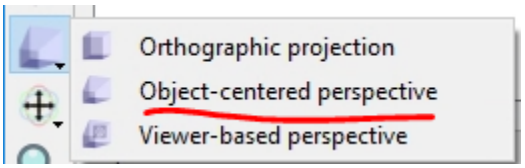
ESRM433/SEFS533
Lab 6
For more information about DAP and forests:

Goodbody, T.R.H., Coops, N.C. & White, J.C. Digital Aerial Photogrammetry for Updating Area-Based Forest Inventories: A Review of Opportunities, Challenges, and Future Directions. *Curr Forestry Rep* **5,** 55–75 (2019). https://doi.org/10.1007/s40725-019-00087-2

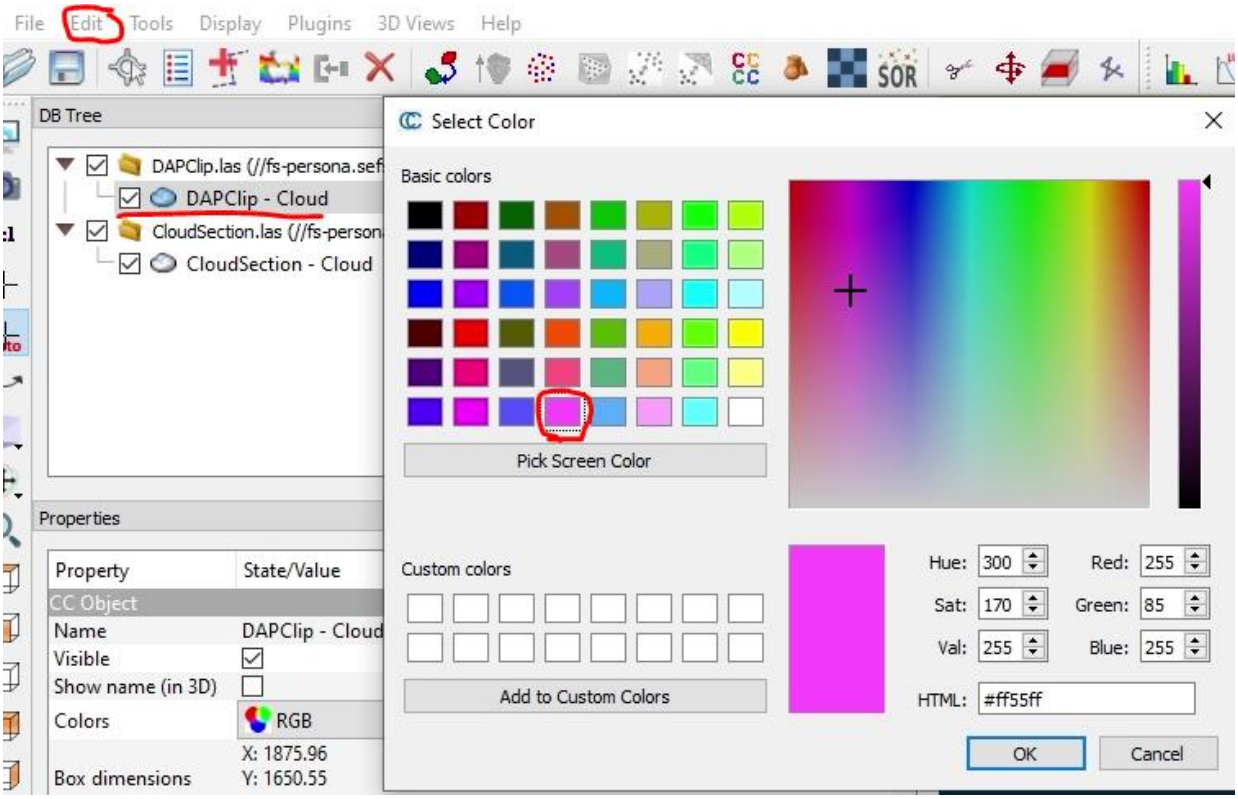
PART 1: Comparing DAP to ALS, Rumple

Let's compare the DAP point cloud to the ALS point cloud.
The ALS point could is LAB5/CloudSection.las
The DAP point cloud is LAB6/DAPClip.las

In CloudCompare, I suggest using Object-centered perspective, and having the EDL shader turned on. Also, adjust point size to larger when zoomed into a cloud.



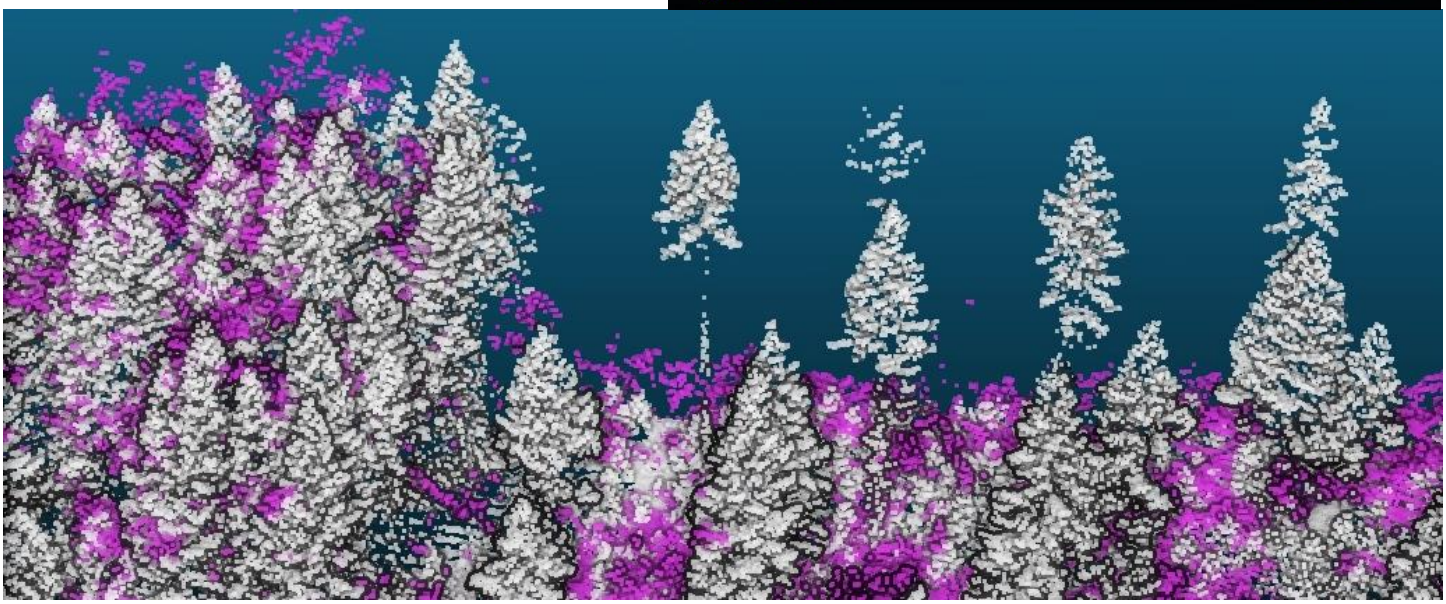
Drag both those files into the same instance of CloudCompare. To visually compare two point clouds, it is helpful to color them differently. Going to Edit > Colors > Set Unique



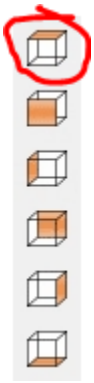
I am going to use pink for the DAP, and white for the ALS, but you can use whatever colors you want. The clip extent doesn't match exactly so points on the edge won't match, but internally the points should be very similar.

Take a moment and look around the cloud. Where do points align well, and where are points missing from either layer missing?

Look at the trees in the NW corner of the clip.



QUESTION 1: The white is lidar and the pink is DAP. The clustered trees on the left of the image above have some errant points above them, but generally are captured fairly well with both the lidar and DAP. The single trees that are standing on the right of the image seem to be missed entirely with the DAP point cloud. Why do you think this would occur?

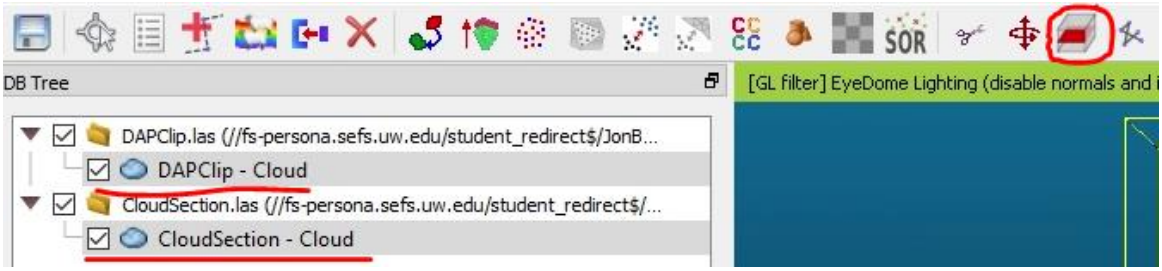


Let's take a slice of the point clouds to get a profile. Center your view to overhead using the top view button.

Highlight both the ALS and DAP point clouds, and select the Cross Section tool.

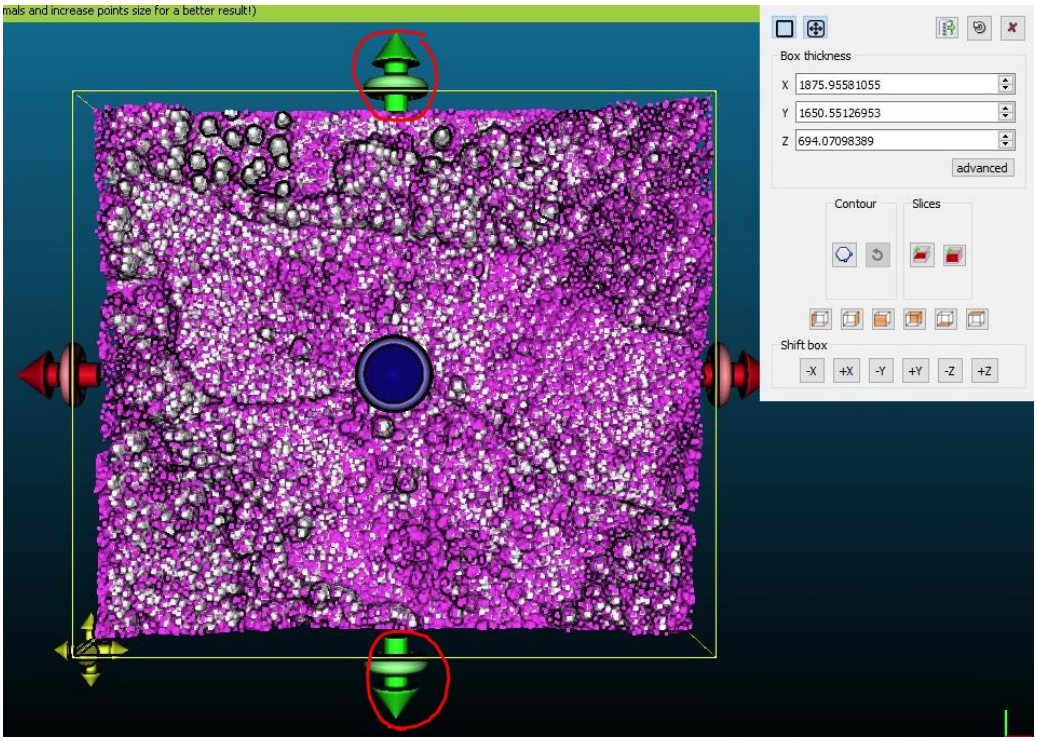
ESRM433/SEFS533

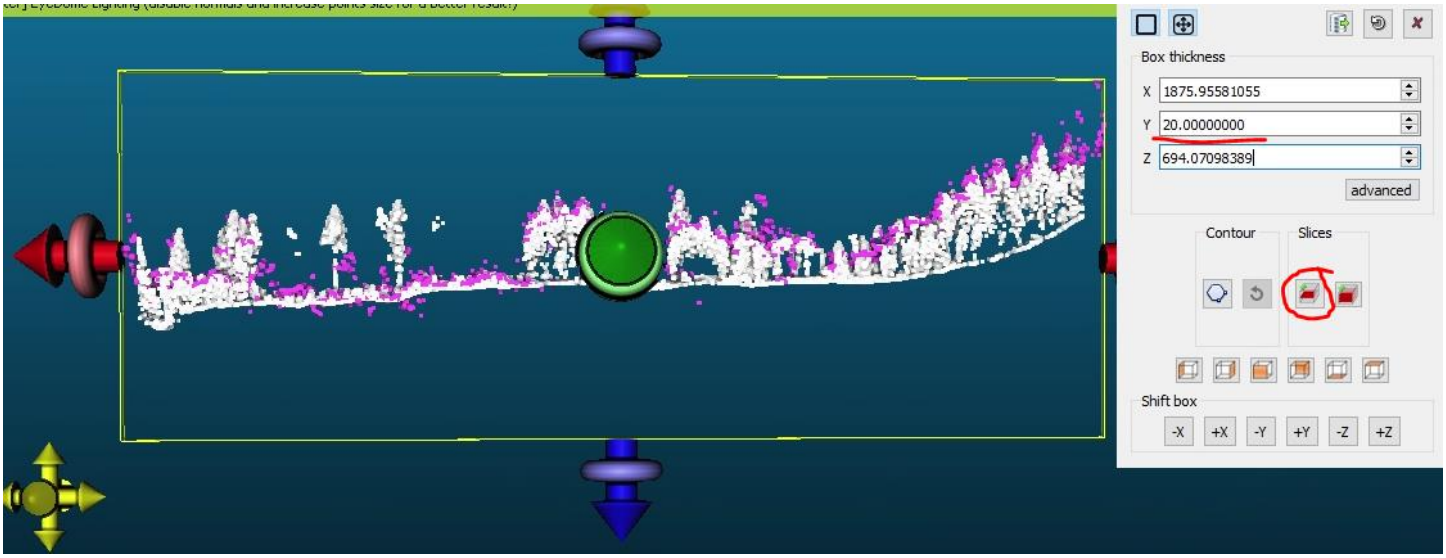
Lab 6



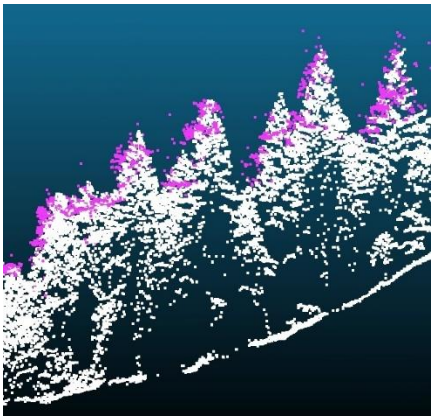
Use the green arrows to create a thin slice of the point clouds. Having a Y box thickness of around 10 to 20 should be good, and the location of your slice is totally up to you. When you are happy with your slice, use the “Export selection as new cloud” tool.

Close the Cross Section tool





QUESTION 2: Include a screen shot of your slice. Describe the differences between the ALS and DAP point clouds. Why does the DAP point cloud only model the tops of the trees and not penetrate all the way to the ground?

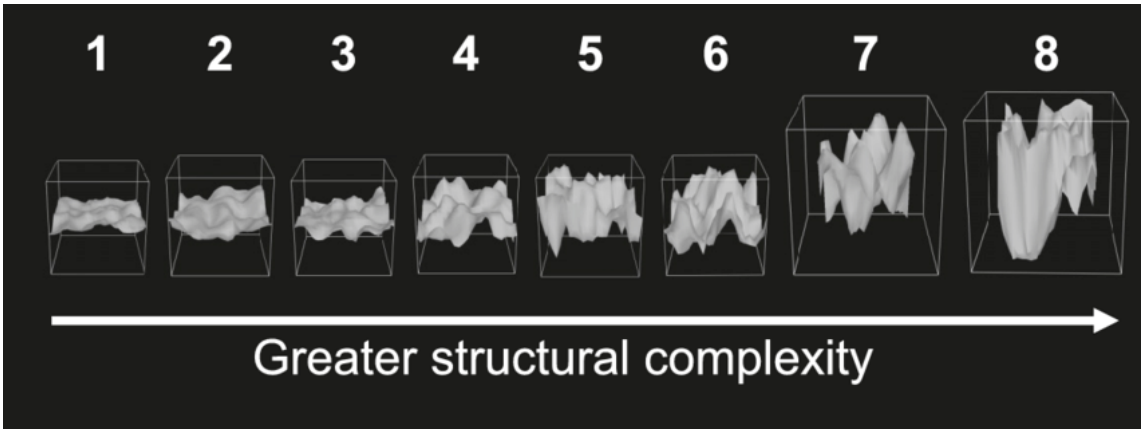


Back to RStudio and lidR!

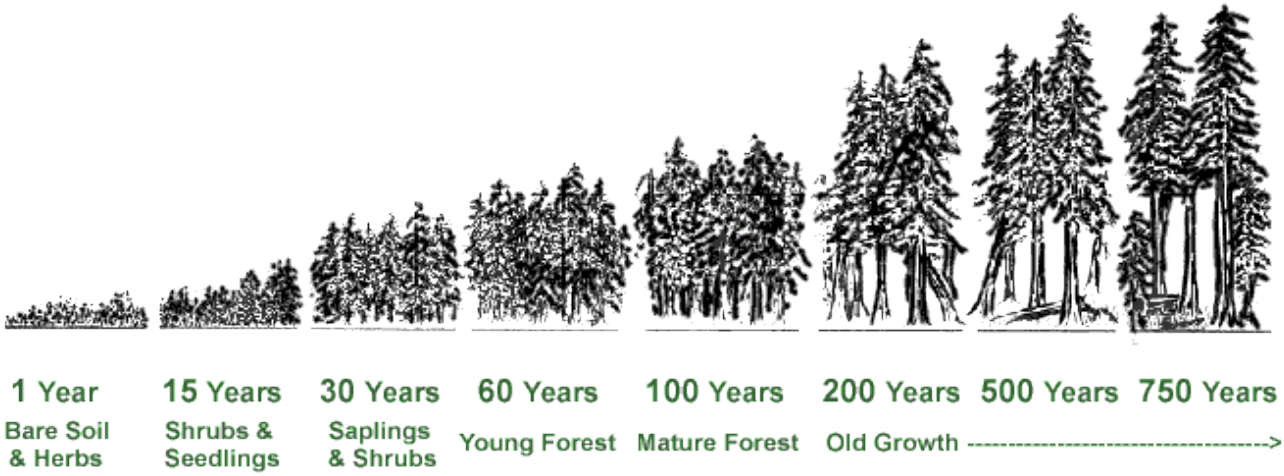
Rumple is the ratio of the surface area of a canopy height model to the area of that model projected to the ground. A higher rumple, the more complex the canopy surface. Rumple has been used to model basal area, QMD, and other forest metrics.

Jenness, J. S. (2004). Calculating landscape surface area from digital elevation models. Wildlife Society Bulletin, 32(3), 829–839

Kane, V. R., McGaughey, R. J., Bakker, J. D., Gersonde, R. F., Lutz, J. A., & Franklin, J. F. (2010). Comparisons between field-and LiDAR-based measures of stand structural complexity. *Canadian Journal of Forest Research*, *40*(4), 761-773.



Conceptually, think about an ancient stand of primordial forest. There will be some giant tall trees with gaps in the forest canopy. The gaps and tall trees will result in a higher rumple value. An even aged stand in a plantation will likely have a relatively “smooth” rumple as the canopy is more homogeneous. Keep in mind that disturbance events like fire, disease, logging also change forest structure.



QUESTION 3: Based on the illustration above, which age class would you expect to have the highest rumple, and the lowest rumple? Why? Note that this is a generalization and isn’t necessarily true in all cases.

For part 1, we will be using lab 5 data, and the new DAP clip.

Make sure you setwd and load the libraries.
There are a few additional packages that will need to be installed.
Also, the package ‘Terra’ is a new package that will be replacing the older ‘Raster’ package. LidR is already incorporating ‘Terra’ into its tools but it is still a little buggy. We will need to install the development version of terra.
Working with opensource software that is under active development can be buggy, but in my opinion, opensource is almost always preferable to proprietary software.

```
install.packages("future") #package for parallel processing
remove.packages('terra') #you may need to run this and restart R
install.packages('terra', repos='https://rspatial.r-universe.dev') #development version

library(lidR)
library(terra) #package that is replacing 'raster'
library(mapview) #package for viewing with maps. Has issues with terra
library(future) #run this if you get an error 'there is no future' ;)
help(package = terra)
```

ESRM433/SEFS533

Lab 6

Assuming that you have your CloudSection.las in a LAB5 folder and DAPClip.las in LAB6 along with VendorDTM_Clip.tif. For this exercise, you won't be creating a tin from the point cloud to normalize the cloud, but rather you will be using the vendor created DTM to help save some time.
We are using the new package terra for the DTM and it is still a little buggy. Load the ALS, DAP, and DTM into R, and check them out if you want.

```
setwd("//fs-persona.sefs.uw.edu/student_redirect$/jonbatch/Desktop/ESRM433/")

ALS <- readLAS("Lab5/CloudSection.las")
DAP <- readLAS("Lab6/DAPClip.las")
DTM <- rast("Lab6/VendorDTM_Clip.tif")
plot(ALS)
plot(DAP)S
plot(DTM)
```

We are going to normalize the points clouds:

```
?normalize_height
ALSN <- normalize_height(ALS,DTM) #using the vendor clip DTM instead of creating a tin()
DAPN <- normalize_height(DAP,DTM) #normalize the DAP point cloud using vendor DTM
writeLAS(ALSN, file="Lab6/ALSN.las") #writes out the normalized cloud
writeLAS(DAPN, file="Lab6/DAPN.las") #take them into CloudCompare if you want
```

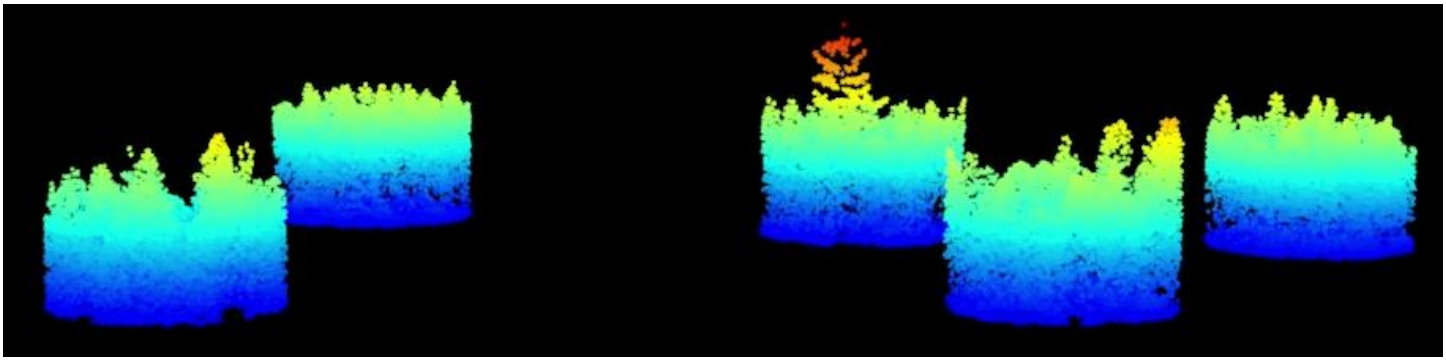
I have been getting an error "attempt to apply non-function" but it has been working when trying a second time. I am unsure about why this is.

Not required for the lab, but I would encourage you to take your normalized clouds into CloudCompare to literally compare the clouds. Because we normalized the ALS cloud, when we clip the data, we won't have to normalize each clip as we did in the previous lab.

Create the clips for the plots, stick them all into one file and then look at it:

```
#create 5 circle clips with the normalized ALS cloud
P1 <- clip_circle(ALSN,1636,-2636,60)
P2 <- clip_circle(ALSN,1430,-2230,60)
P3 <- clip_circle(ALSN,1216,-2425,60)
P4 <- clip_circle(ALSN,1279,-2725,60)
P5 <- clip_circle(ALSN,1139,-2174,60)
ALSplots <- rbind(P1,P2,P3,P4,P5) #combine the las clips
plot(ALSplots)
```

Should look familiar!



Let's go ahead and create a rumple index values for our plots, the first step for each plot is to create a canopy height model:

```
chm1 <- rasterize_canopy(P1, 1.6, p2r(3.28)) #creates the canopy surface model using p2R
chm1R <- rumple_index(chm1)                  #creates the rumple index value for the clip
chm2 <- rasterize_canopy(P2, 1.6, p2r(3.28))
chm2R <- rumple_index(chm2)
chm3 <- rasterize_canopy(P3, 1.6, p2r(3.28))
chm3R <- rumple_index(chm3)
chm4 <- rasterize_canopy(P4, 1.6, p2r(3.28))
chm4R <- rumple_index(chm4)
chm4 <- rasterize_canopy(P4, 1.6, p2r(3.28))
chm4R <- rumple_index(chm4)
chm5 <- rasterize_canopy(P5, 1.6, p2r(3.28))
chm5R <- rumple_index(chm5)
ALSRumple <- rbind(chm1R,chm2R,chm3R,chm4R,chm5R) #combines rumple values
plot (ALSRumple)
```

QUESTION 4: Using the ALS data, What plot had the highest rumple value? What plot had the lowest rumple value? Submit a screen shot of the las cloud for both of the plots. The function is: plot(P_) you just need to fill in the _.

Same steps with the DAP data:

```
#create 5 circle clips with the normalized DAP cloud
D1 <- clip_circle(DAPN,1636,-2636,60)
D2 <- clip_circle(DAPN,1430,-2230,60)
D3<- clip_circle(DAPN,1216,-2425,60)
D4 <- clip_circle(DAPN,1279,-2725,60)
D5 <- clip_circle(DAPN,1139,-2174,60)
DAP_PLOTS <- rbind(D1,D2,D3,D4,D5)
plot(DAP_PLOTS)
```

```
?rumple_index
chmD1 <- rasterize_canopy(D1, 1.6, p2r(3.28))
chmD1R <- rumple_index(chmD1)
chmD2 <- rasterize_canopy(D2, 1.6, p2r(3.28))
chmD2R <- rumple_index(chmD2)
chmD3 <- rasterize_canopy(D3, 1.6, p2r(3.28))
chmD3R <- rumple_index(chmD3)
chmD4 <- rasterize_canopy(D4, 1.6, p2r(3.28))
chmD4R <- rumple_index(chmD4)
chmD5 <- rasterize_canopy(D5, 1.6, p2r(3.28))
chmD5R <- rumple_index(chmD5)
DAPRumple <- rbind(chmD1R,chmD2R,chmD3R,chmD4R,chmD5R)
plot(DAPRumple)
```

QUESTION 5: Using the DAP data, What plot had the highest rumple value? What plot had the lowest rumple value? Are they the same as using the ALS data? If not, why do you think they were different? Submit a screen shot of the las cloud for both of the plots. The function is: plot(D_) you just need to fill in the _.

Let's see if the rumple value from the ALS and DAP correlate:

```
Y <- ALSRumple #Dependent Variable
X <- DAPRumple #Independent Variable
reg <- lm (Y ~ X)
summary(reg)
plot(X, Y, xlab="DAP", ylab="ALS", main="Rumple")
abline(reg, col="blue") #regression line
abline(0,1, col="red") #1 to 1 line
```

The blue line is the regression line, and the red is a 1 to 1. If a perfect match, the red and blue would be the same.

QUESTION 6: What is your p-value and R-squared for ALSRumple compared to DAPRumple? Is there a significant relationship?

We have a very small sample size of statistical inference is difficult, but DAP and ALS are comparable, especially if only using ALS first returns, or metrics based on CHMs.

PART 2: Comparing DAP to ALS, Cloud Metrics

So far we have only created cloud metrics for plots. A single value for an entire plot. It is possible to create a grid across a lidar tile and generate metrics for every grid cell. We are going to do that, with the intention of further comparing DAP to ALS.

To make the comparison easier, we want to make sure our two tiles are the exact same extent:

```
print(DAPN) #check out the extent and basic las information
print(ALSN) #Extent must match
```

The print command gets you the basic information about the las file. You should be familiar with this command. Unfortunately, the extents don't match exactly. An easy way to get the extents to match that will result in output rasters with aligned cells is to simply clip the las files, just like you did for the plots, but this time use a rectangle.

```
?clip_rectangle
#Clipping the ALS and DAP data
ALSNc <- clip_rectangle(ALSN, 700, -3300, 2400, -1700)
#for a direction comparison, the extent must match
DAPNc <- clip_rectangle(DAPN, 700, -3300, 2400, -1700)

print(DAPNc) #checking to see if the extents match now
print(ALSNc) #hopfully they do
```

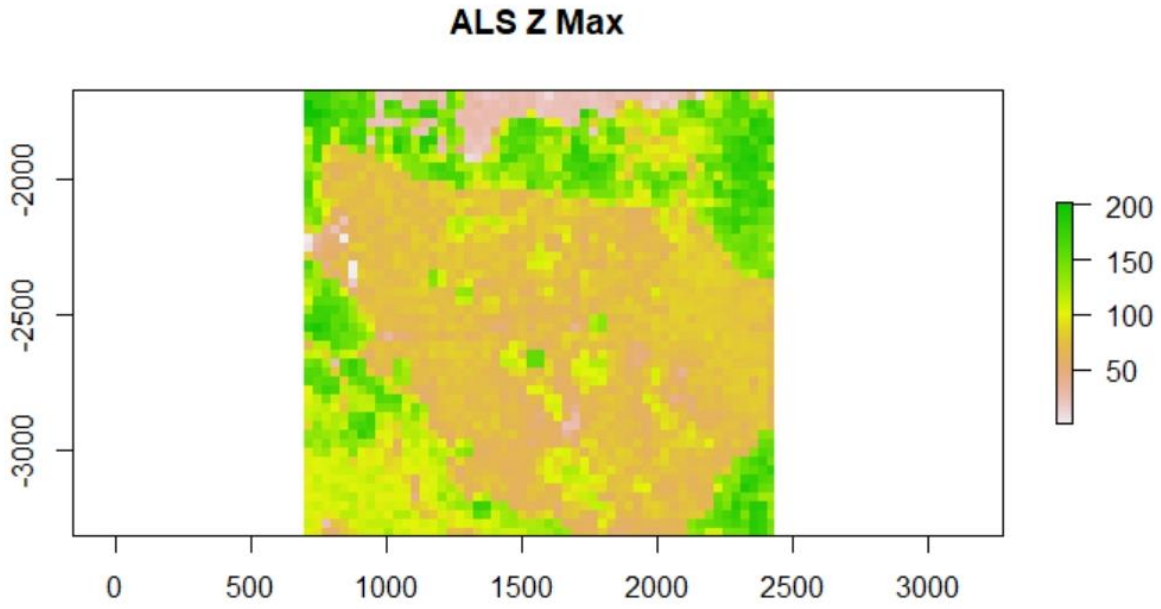
Once our extents match, we can use pixel_metrics to create our .stdmetrics for our Z values. If we ran cloud_metrics on DAPNc and ALSNc, we would get one value for all metrics for the entire extent. Exactly what we got when running cloud_metrics on our plots.

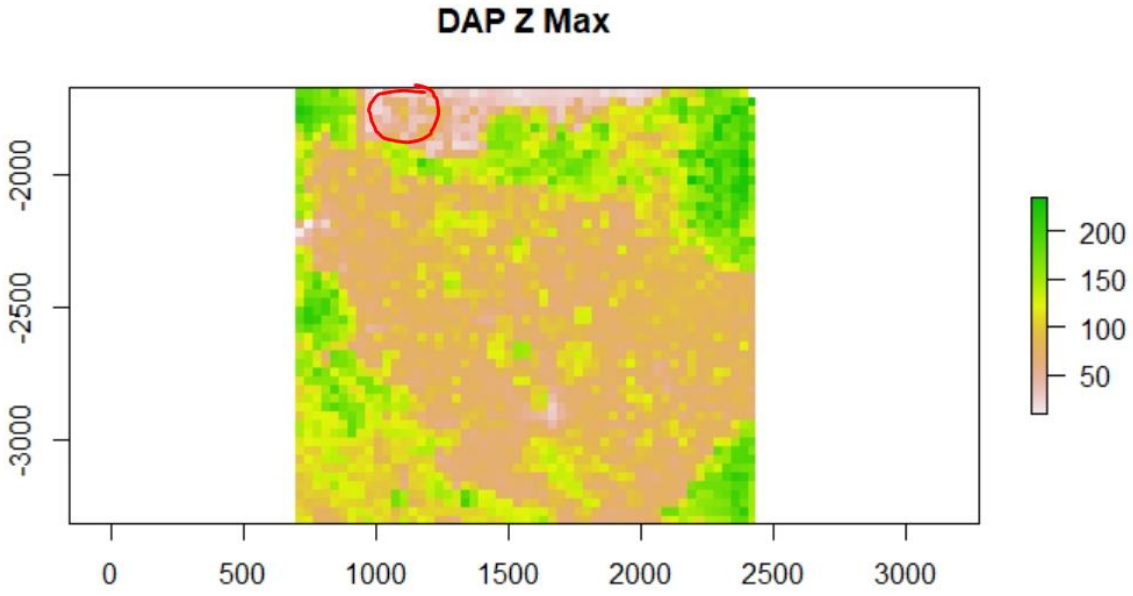
Pixel_metrics lets us input a grid size (res) and the output will be a raster with raster cells of our defined resolution. One raster for each metric and they get all pilled together in a raster brick in R.

```
?pixel_metrics
#creates standard elevation metrics for all Z values
ALSrasters <- pixel_metrics(ALSNc, ~stdmetrics_z(Z), res = 32.8)
#output is a raster brick, output won't include intensity metrics
DAPrasters <- pixel_metrics(DAPNc, ~stdmetrics_z(Z), res = 32.8)
plot(ALSrasters) #checking out all the rasters in the brick
plot(DAPrasters) #not all rasters will be shown
names(ALSrasters) #names of all rasters in brick
names(DAPrasters) #rasters are the cloud_metrics!
```

The lines should all look familiar to you. The additional "res = 32.8" tells R that we want output rasters with a resolution of 32.8 units. Our units in these clouds is feet, so the output will be 10m squares. The plot function will show you a partial list of the rasters in the raster bricks. Note that not all are showing. You can use "names" to get the names of all the rasters in the raster brick. The names should look familiar to you. They are the cloud_metrics! If you were processing a larger amount of lidar data, you can be more selective about what metrics you generate.
You can select which raster you want from the raster brick with "$"
Let's look at our output for zmax, and compare our two rasters with a scatter plot:

```
plot(ALSrasters$zmax, main="ALS Z Max") #cell size is 10m
plot(DAPrasters$zmax, main="DAP Z Max") #the maxium Z value within each cell
#scatter plot of matched cell values. Won't work if extents differ
plot(ALSrasters$zmax, DAPrasters$zmax)
```

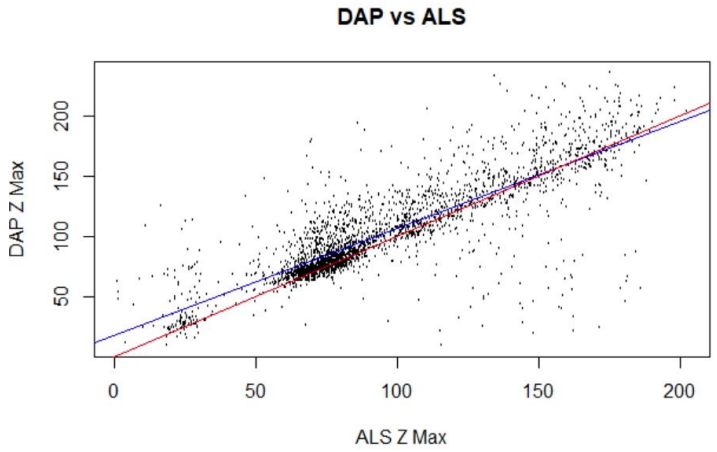




You can really see the omission of the tall, solo trees that we looked at in the beginning of the lab…

They look pretty close, but how close are they actually? We can preform a linear regression on the matched cells of the rasters:

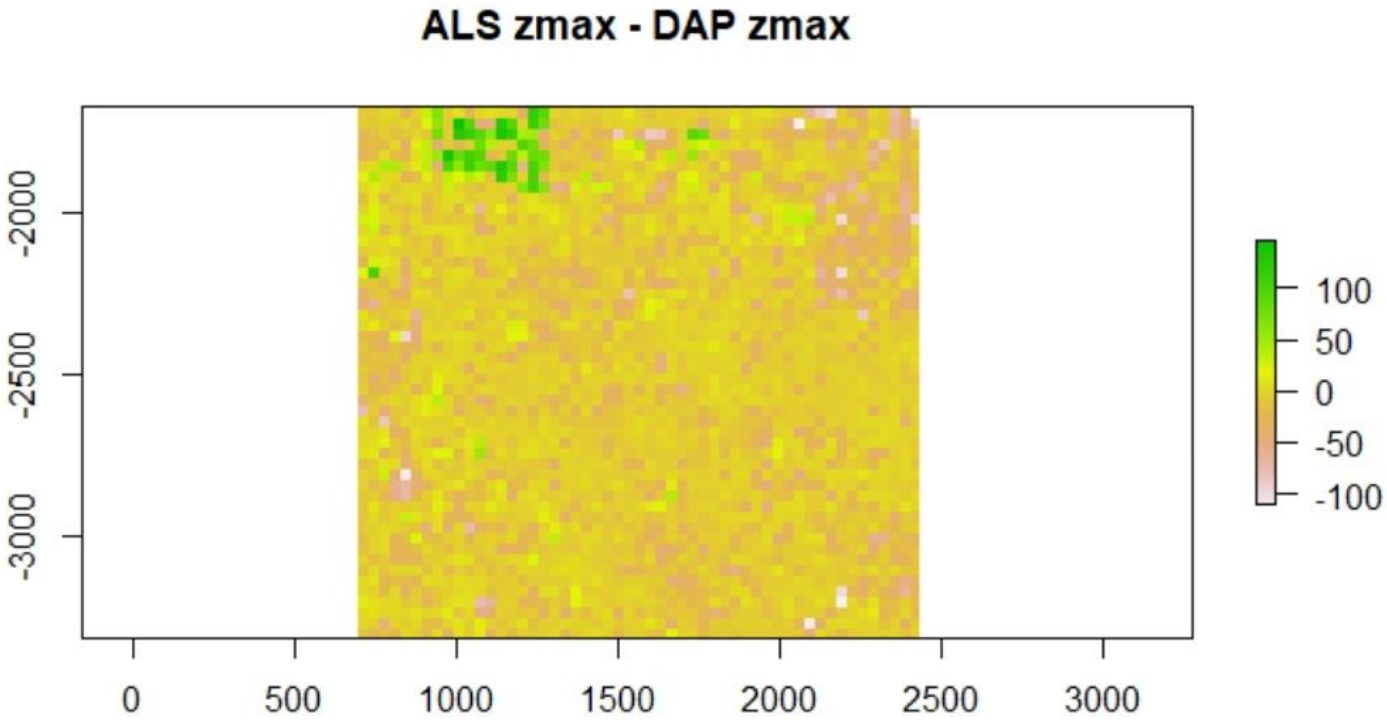
```
#we can do a basic linear regression to see if our cell values "match"
L1 <- ALSrasters$zmax #indentifies L1 as being the ALS zmax raster
L2 <- DAPrasters$zmax
s <- rast(list(L1, L2)) #creates a raster stack for the regression
#v <- data.frame(na.omit(value(s))) #turns rasters into a dataframe ### unnessesary now
names(s) <- c('L1', 'L2') #arranges the data for the regression
m <- lm(L2 ~ L1, data=s) #regression comparing matched raster cells
summary (m)
plot(L1, L2, xlab = 'ALS Z Max', ylab="DAP Z Max", main="DAP vs ALS")
abline(m, col="blue")
abline(0,1, col="red") #1 to 1 line
```



QUESTION 7: Pick a metric other than zmax from your raster outputs. All the options can be seen using names(ALSrasters). Include screenshots of the DAP and ALS versions of your metric rasters (like the ones above). Preform a linear regression on the rasters and report your findings. Include p-value, r-squared value, and a short discussion. Make sure to include a properly labeled figure of the scatter plot with the regression line and the 1 to 1 line added.

We can lastly create a raster of the difference between our ALS and DAP metric rasters. Check out the biggest difference where those solo standing tall trees were…

```
ZDiff <- (ALSrasters$zmax - DAPrasters$zmax)
plot(ZDiff, main="ALS zmax - DAP zmax")
```



QUESTION 8: Create a difference raster of a metric other than zmax. It can be the same metric you used for the regression, or a different one. Label and caption your figure.

Beyond the activity of comparing ALS and DAP, this process is how you would create metrics across larger areas, and not just single plots. One of the first processing steps when getting large acquisitions is to create metric rasters for the entire extent. This could be hundreds of gigabytes of data that would take a powerful computer days to process… so lets do it on Madrona!

PART 3: Metric Rasters from lascatalog

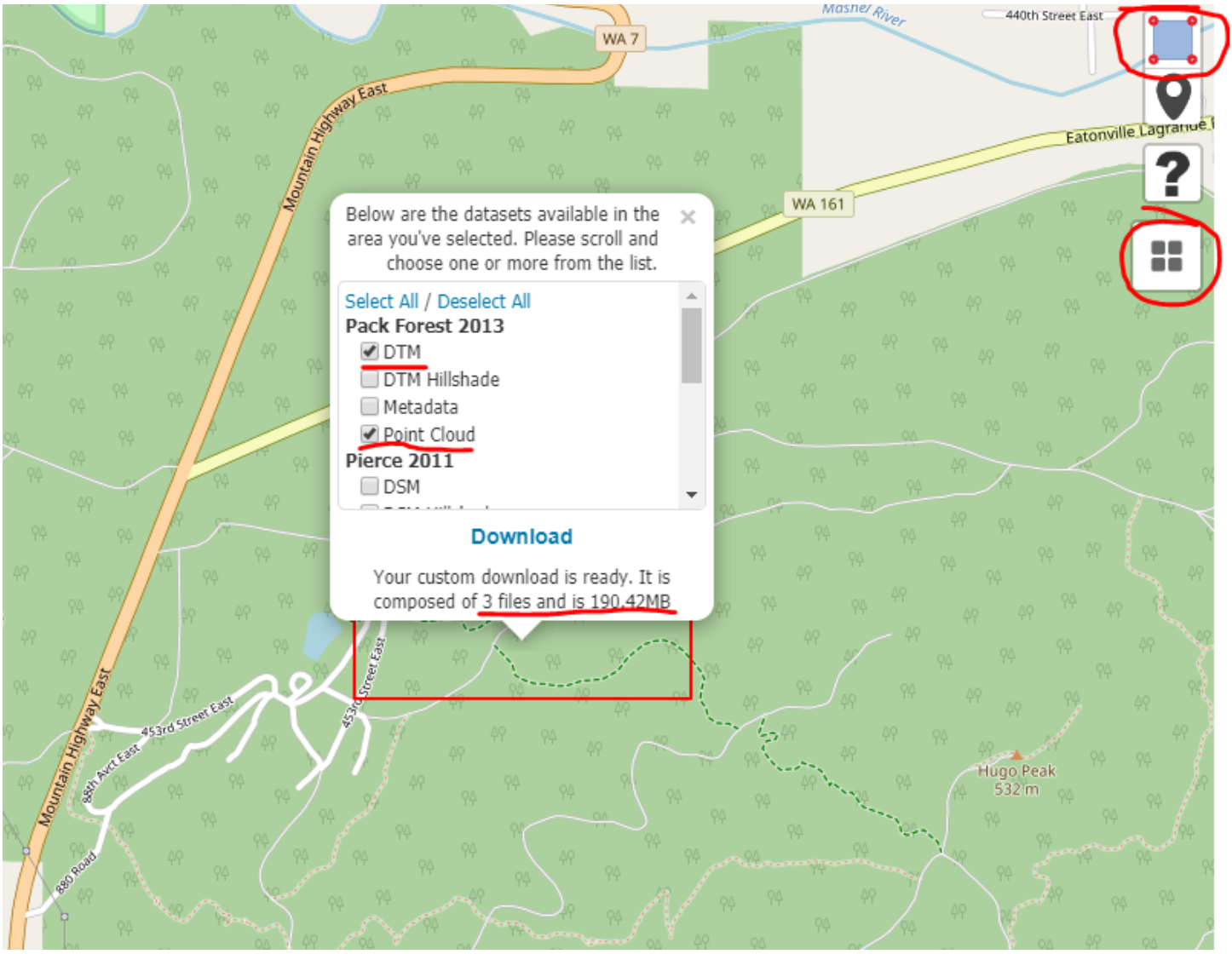
We are only going to process two tiles from Pack forest. The Pack forest tiles are relatively small. We are going to produce metric rasters from the tiles and then use linear regression to try and model the QMD for the area. Then finally determine which areas within the two tiles have both a canopy closure > 70% and a QMD > 15. Remember, these are believed to be habitat preferences for the Spotted Owl. As a note, we aren't doing full habitat modeling, just identifying forest that likely meets those two elements.

ESRM433/SEFS533

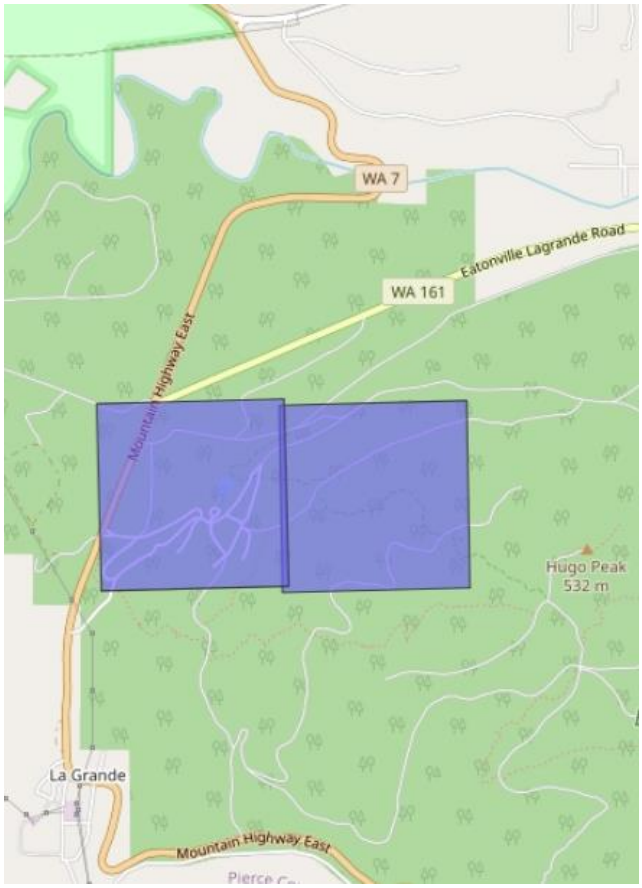
Lab 6

Go to Washington DNR lidar portal and find Pack forest.

I would suggest using the open street maps layer as Pack forest is clearly labeled. Use the area select tool to draw a rectangle around the approximate same area. You want the DTM and the Point cloud from Pack Forest 2013. “3 files and is 190.42MB”



ESRM433/SEFS533
Lab 6



This is the footprint of the actual tiles that you want.

Unzip the data and drop the dtm and laz folders directly into your Lab6 folder.
The DTM should be located at Decktop/ESRM433/Lab6/dtm/
The laz files should be located at Decktop/ESRM433/Lab6/laz/

Back to R

We are going to be using readLAScatalog. You used it before, but now we are actually going to use some of the power in the catalog tool. Import your data and fix the crs issues in lidR:

```
?readLAScatalog
#folder you put the 2 Pack forest laz files from DNR
ALS <- readLAScatalog("Lab6/laz/")
las_check(ALS)
#DTM you downloaded from DNR
DTM <- rast("LAB6/dtm/pack_forest_2013_dtm_1.tif")
plot(DTM)
#we know lidR doesn't like feet...
projection(ALS) #checking crs
st_crs(ALS)<-2927 #changing crs
projection(DTM)
crs(DTM)<-"epsg:2927" #different to define the CRS for a raster
```

Lets take a look at our DTM and laz files to make sure they are in the correct location now…

```
#this can take a bit. Make sure the DTM is located correctly
#mapview(DTM) #mapview doesn't currently work with terra
#check the location of pack forest tiles
plot(ALS,mapview=TRUE)
```

The output from plot(ALS,mapview=TRUE) should look familiar, it is the figure above where I show you exactly what two tiles we are interested in. We can't view the DTM in mapview as mapview has not been updated yet to work with the new raster package terra. Hopefully in the next couple of weeks they will work together.

We are now going to normalize both tiles with one set of commands, using the vendor supplied DTM. It is a little more complicated than what we did before. Make sure to check out the documentation in ?normalize that has to do with LAScatalog. We have to define chunk size and a location to write the normalized laz files as it processes them. Before we were only dealing with data that was small enough that it could be written out to R, and with only processing 2 tiles, R could handle it, but if you were to process all the tiles at once for Pack Forest, this is the same process you would use.

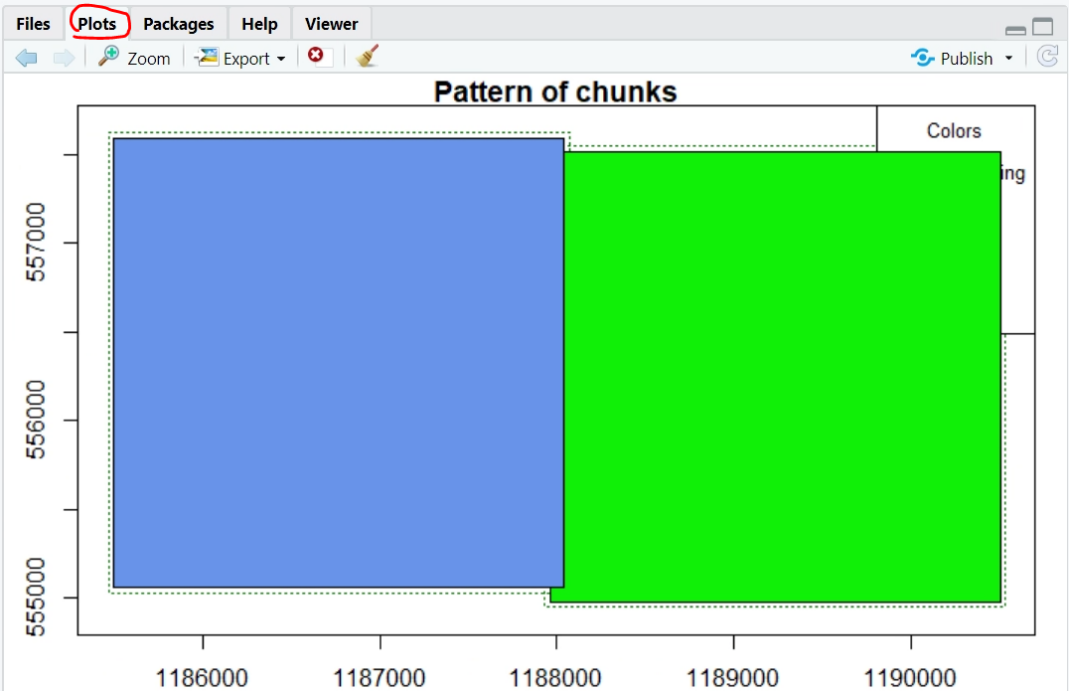
Note that there is a new folder it is pointing to (Lab6/lazN/…) R should just create this folder, but you may need to go in and manually create that folder in windows.

This is probably the longest running command we have done so far:

```
?`LAScatalog-class`
?normalize_height #extra steps are needed for working with a catalog
opt_chunk_size(ALS) <- 0 #the chunk size. 0 means the entire tile.
#location for output files. Files need to be written out of R due to size
opt_output_files(ALS) <- "Lab6/lazN/{ORIGINALFILENAME}_Norm"
#There are differnet naming conventions {ORIGINALFILENAME} keeps original file name
opt_laz_compression(ALS) <- TRUE #we want output as laz vs las
ALSN <- normalize_height(ALS, DTM) #Run normalize on the tiles using DTM you downloaded
#This will take a long time... Imagine if you did this for more than 2 tiles...
#in your "plots" window, it updates you on what tiles are being worked on.
plot(ALSN, mapview=TRUE)
```

Look in your plots tab and it will give you a status of the processing

If you get an error, try running the 'remove.packages ('terra') then restarting and reinstalling terra with the development version.

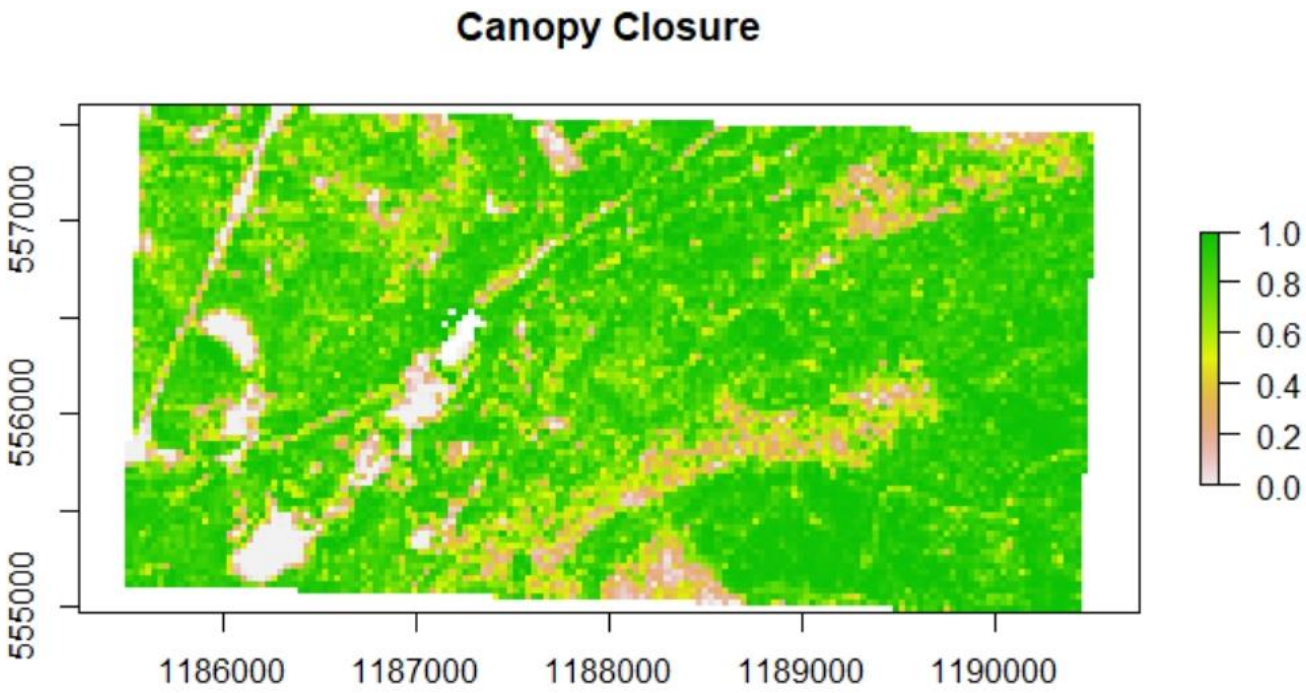


There are many more “opt_” parameters that can be set using las catalog. All the documentation is in the lascatalog help pages.
If you were dealing with really large tiles or a very weak computer, you can define chunk size to 500 or some other value. lidR will bite off chunks of that size to process. A chunk of 500 would be a 500 ft x 500 ft section and there would be created an output file for each chunk. You only need to write files to a directory if the data is really big (e.g. point clouds). Metric rasters don’t need to be written out to a external folder as R can handle the rasters (at least our two rasters). We can also filter out points we don’t want to include in our processing.

Next, calculate the canopy cover for our two tiles. The expression sum(Z>6.56)/sum(Z>=0) should be familiar to you. >= is “greater than or equal to”. We are filtering out points that have a z value below 0.

You now have a raster object named “PackCC” that gives the canopy closure for the two tiles at a resolution of 10m. How flippin cool is that? To judge the accuracy, we would need to do some ground truthing but if you use mapview, the cover percentages seem reasonable given what is on the ground

```
#need to have the package 'future' installed and loaded
opt_chunk_size(ALSN) <- 0 #the chunk size. 0 means the entire tile.
#If output_files = "" outputs are returned in R
opt_output_files(ALSN) <- ""
#run rlas:::lasfilterusage() for list of filters
opt_select(ALSN) <- "xyzci"
opt_filter(ALSN) <- " -drop_z_below 0"
PackCC <- pixel_metrics(ALSN, ~sum(Z>6.56)/sum(Z>=0), 32.8)
```



QUESTION 9: use plot() to create a figure of your canopy closure metric raster. Fully caption the figure with a very brief discussion as to how it was made. Think of it as a figure in a paper you are submitting. Don't want just snippets of R code, but rather a conceptual description of how it was made. Feel free to take the raster into ArcGIS to make a map there if you like.

We now can make metric rasters for all stdmetrics, or just a few that we are interested in. In the last lab, you estimated QMD from multiple linear regression. There was a significant relationship between QMD and the height of points in the 25th percentile + average point height. We can produce rasters for just those two metrics. In this case, we want to exclude all points below 6.56 feet so we include "-drop_z_below 6.56" as a filter. We can also make a custom function to get the 0.25 percentile and the mean z value. If you want, you can just run all the standard metrics with ~stdmetrics instead of ~f(z):

```
#If output_files = "" outputs are returned in R
opt_output_files(ALSN) <- ""
#run rlas:::lasfilterusage() for list of filters
opt_filter(ALSN) <- " -drop_z_below 6.56"
#makeing our own function
f = function(x) {list(zq25 = quantile(x,0.25), mean = mean(x))}
PackZ <- pixel_metrics(ALSN, ~f(Z), 32.8)
plot(PackZ$zq25)
plot(PackZ$mean)
```

Warnings like:

```
The original tiling pattern does not match with the resolution 32.8. Chunks were extended to avoid partial pixels.
```

Are fine. It just is letting you know that 32.8 didn't fit neatly into the tiles and needed to be adjusted.

ESRM433/SEFS533
Lab 6
PART 4: Regression with Rasters and "Habitat"

In part 3 I referred to the lab 5 regression analysis. We can use the coefficients from a linear regression to take cell values from input rasters, to create a new raster. Specifically, we can use the coefficients from the multiple linear regression to model QMD from zq25 and zmean.

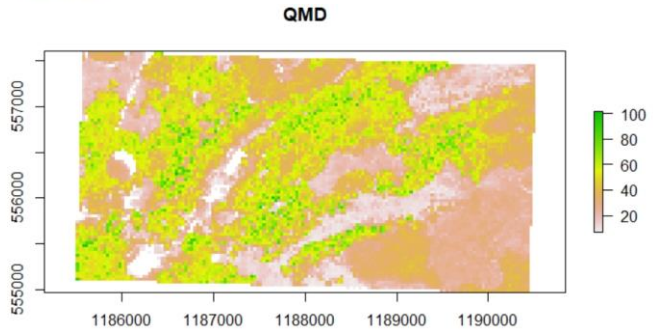
We will use the more extensive plot data and cloud metrics from FUSION that you used in the first part of lab 5:

```
field <-(read.csv("LAB5/plotdata.csv"))
lidar <- (read.csv("LAB5/cloudmetrics.csv"))
QMD <- field$QMD.GT5
mean <- lidar$Elev.mean
zq25 <- lidar$Elev.P25
Mreg <-lm(QMD ~ mean + zq25)
summary (Mreg)
```

We need the names to match our metric rasters so they need to be "mean" and "zq25", not Elev.mean and Elev.P25. Those are the naming convention from the lidar software FUSION. We have to run this regression so we have the object "Mreg" which contains the coefficients from the regression analysis.

We will model (predict) what the QMD value at each raster cell is based on the values in our zq25 and mean rasters using the predict function in the raster package.
Take a moment to read ?predict, then run:

```
?predict
#PackZ is our raster brick that contains rasters
#with names that match the coefficients in Mreg
QMDP <- predict(PackZ, Mreg) #new raster "QMDP"
plot (QMDP, main="QMD")
```



Again, we don't know how accurate our QMD model is without going out and ground truthing sections of the area. The regression we used was based on plot data from a different area so the relationships

may be wrong. I suspect we are over estimating QMD with our model. Who wants to head out to Pack forest to measure some trees?

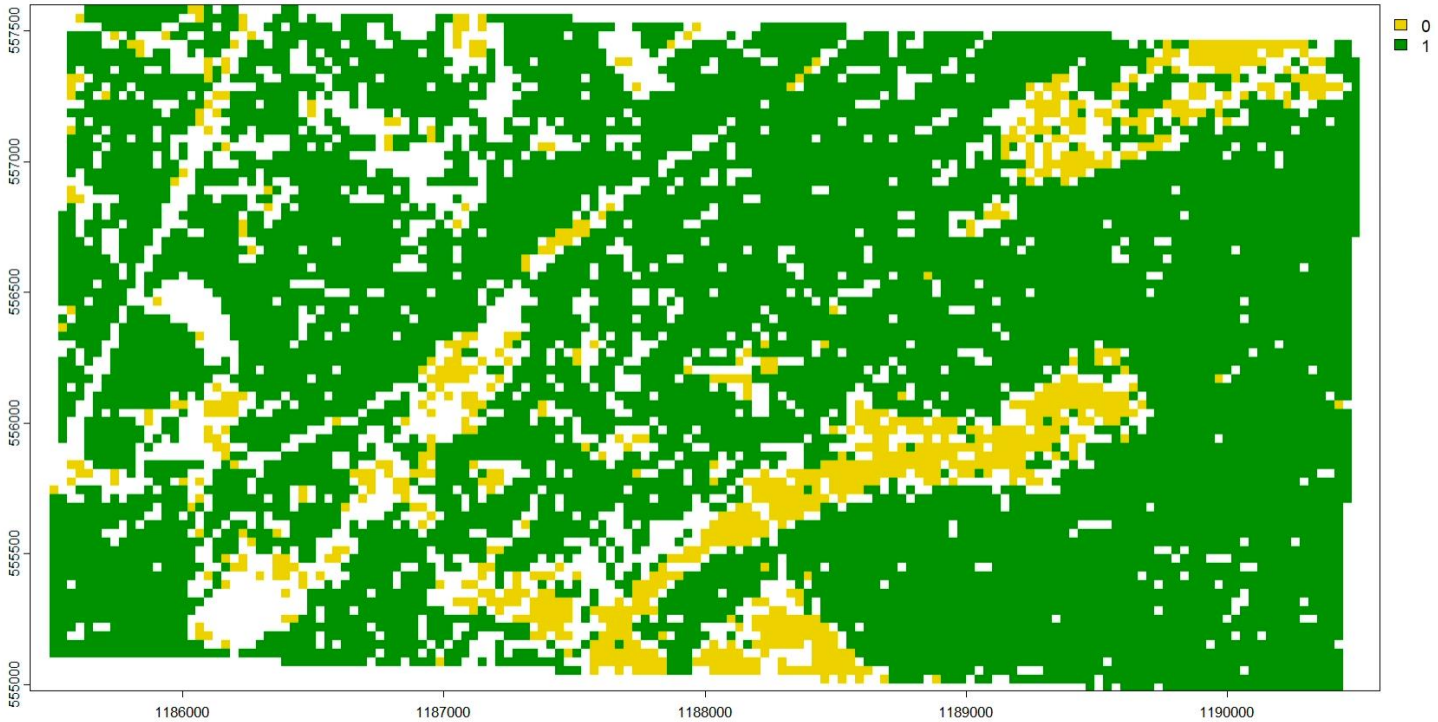
QUESTION 10: use plot() to create a figure of your QMD metric raster. Fully caption the figure with a very brief discussion as to how it was made. Think of it as a figure in a paper you are submitting. Don't want just snippets of R code, but rather a conceptual description of how it was made. Feel free to take the raster into ArcGIS to make a map there if you like.

LAST STEP!

OK, you were told that Spotted Owls like canopy closure > 70% and QMD > 15". You are now able to identify areas within our lidar data that meet those two criteria. We are going to use a very simple "yes/no" determination.

We can take our rasters QMDP and PackCC and create a new raster with values of either 0 or 1 to denote if that location has a QMD > 15 & Canopy Closure > 70%
Run:

```
Model<-rast(QMDP)
Model[QMDP > 15 & PackCC > 0.7] = 1
Model[QMDP < 15 & PackCC < 0.7] = 0
plot(Model)
```



According to our lidar data, a good portion of our area has QMD > 15 and Canopy Closure > 70%. Note that there are only three possible values in our raster; 1, 0, NA. The green bits are 1, the yellow bits are 0 and the "clear" bits are NA.

QUESTION 11: use plot() to create a figure of your model raster. Discuss resolution and why you made the raster. Fully caption the figure with a very brief discussion as to how it was made. Think of it as a figure in a paper you are submitting. Don't want just snippets of R code, but rather a conceptual description of how and why it was made (other than it was in an assignment). Feel free to take the raster into ArcGIS to make a map there if you like.

GRADUATE STUDENT

You downloaded two lidar tiles to test out readLAScatalog. Either download a third tile from Pack forest, or two tiles and a DTM from somewhere else, and re answer questions 9-11. For the captions you can just use a single sentence to label the metrics.

R Appendix

ESRM433/SEFS533

Lab 6

```
##################################
########## Lab 6 ################
##################################
install.packages("future") #package for parallel processing
remove.packages('terra') #you may need to run this and restart R
install.packages('terra', repos='https://rspatial.r-universe.dev') #development version

library(lidR)
library(terra) #package that is replacing 'raster'
library(mapview) #package for viewing with maps. Has issues with terra
library(future) #run this if you get an error 'there is no future' ;)
help(package = terra) #check out the help for the raster package

setwd("//fs-persona.sefs.uw.edu/student_redirect$/jonbatch/Desktop/ESRM433/")

ALS <- readLAS("Lab5/CloudSection.las")
DAP <- readLAS("Lab6/DAPClip.las")
DTM <- rast("Lab6/VendorDTM_Clip.tif")
plot(ALS)
plot(DAP)S
plot(DTM)

help(package="lidR") #just a reminder if you need help

?normalize_height
ALSN <- normalize_height(ALS,DTM) #using the vendor clip DTM instead of creating a tin()
DAPN <- normalize_height(DAP,DTM) #normalize the DAP point cloud using vendor DTM
writeLAS(ALSN, file="Lab6/ALSN.las") #writes out the normalized cloud
writeLAS(DAPN, file="Lab6/DAPN.las") #take them into CloudCompare if you want

#create 5 circle clips with the normalized ALS cloud
P1 <- clip_circle(ALSN,1636,-2636,60)
P2 <- clip_circle(ALSN,1430,-2230,60)
P3 <- clip_circle(ALSN,1216,-2425,60)
P4 <- clip_circle(ALSN,1279,-2725,60)
P5 <- clip_circle(ALSN,1139,-2174,60)
ALSplots <- rbind(P1,P2,P3,P4,P5) #combine the las clips
plot(ALSplots)

chm1 <- rasterize_canopy(P1, 1.6, p2r(3.28)) #creates the canopy surface model using p2R
chm1R <- rumple_index(chm1)                  #creates the rumple index value for the clip
chm2 <- rasterize_canopy(P2, 1.6, p2r(3.28))
chm2R <- rumple_index(chm2)
chm3 <- rasterize_canopy(P3, 1.6, p2r(3.28))
chm3R <- rumple_index(chm3)
chm4 <- rasterize_canopy(P4, 1.6, p2r(3.28))
chm4R <- rumple_index(chm4)
chm4 <- rasterize_canopy(P4, 1.6, p2r(3.28))
chm4R <- rumple_index(chm4)
chm5 <- rasterize_canopy(P5, 1.6, p2r(3.28))
chm5R <- rumple_index(chm5)
ALSRumple <- rbind(chm1R,chm2R,chm3R,chm4R,chm5R) #combines rumple values
plot (ALSRumple)

#create 5 circle clips with the normalized DAP cloud
D1 <- clip_circle(DAPN,1636,-2636,60)
D2 <- clip_circle(DAPN,1430,-2230,60)
D3 <- clip_circle(DAPN,1216,-2425,60)
D4 <- clip_circle(DAPN,1279,-2725,60)
D5 <- clip_circle(DAPN,1139,-2174,60)
DAP_PLOTS <- rbind(D1,D2,D3,D4,D5)
plot(DAP_PLOTS)

```

```
?rumple_index
chmD1 <- rasterize_canopy(D1, 1.6, p2r(3.28))
chmD1R <- rumple_index(chmD1)
chmD2 <- rasterize_canopy(D2, 1.6, p2r(3.28))
chmD2R <- rumple_index(chmD2)
chmD3 <- rasterize_canopy(D3, 1.6, p2r(3.28))
chmD3R <- rumple_index(chmD3)
chmD4 <- rasterize_canopy(D4, 1.6, p2r(3.28))
chmD4R <- rumple_index(chmD4)
chmD5 <- rasterize_canopy(D5, 1.6, p2r(3.28))
chmD5R <- rumple_index(chmD5)
DAPRumple <- rbind(chmD1R,chmD2R,chmD3R,chmD4R,chmD5R)
plot(DAPRumple)

Y <- ALSRumple #Dependent Variable
X <- DAPRumple #Independent Variable
reg <- lm (Y ~ X)
summary(reg)
plot(X, Y, xlab="DAP", ylab="ALS", main="Rumple")
abline(reg, col="blue") #regression line
abline(0,1, col="red") #1 to 1 line

###Across a region###

print(DAPN) #check out the extent and basic las information
print(ALSN) #Extent must match

?clip_rectangle
#Clipping the ALS and DAP data
ALSNc <- clip_rectangle(ALSN, 700, -3300, 2400, -1700)
#for a direction comparison, the extent must match
DAPNc <- clip_rectangle(DAPN, 700, -3300, 2400, -1700)

print(DAPNc) #checking to see if the extents match now
print(ALSNc) #hopfully they do

?pixel_metrics
#creates standard elevation metrics for all Z values
ALSrasters <- pixel_metrics(ALSNc, ~stdmetrics_z(Z), res = 32.8)
#output is a raster brick, output won't include intensity metrics
DAPrasters <- pixel_metrics(DAPNc, ~stdmetrics_z(Z), res = 32.8)
plot(ALSrasters) #checking out all the rasters in the brick
plot(DAPrasters) #not all rasters will be shown
names(ALSrasters) #names of all rasters in brick
names(DAPrasters) #rasters are the cloud_metrics!

plot(ALSrasters$zmax, main="ALS Z Max") #cell size is 10m
plot(DAPrasters$zmax, main="DAP Z Max") #the maxium Z value within each cell
#scatter plot of matched cell values. Won't work if extents differ
plot(ALSrasters$zmax, DAPrasters$zmax, xlab="DAP",ylab="ALS")

#we can do a basic linear regression to see if our cell values "match"
L1 <- ALSrasters$zmax #indentifies L1 as being the ALS zmax raster
L2 <- DAPrasters$zmax
s <- rast(list(L1, L2)) #creates a raster stack for the regression
#v <- data.frame(na.omit(value(s))) #turns rasters into a dataframe ### unnessesary now
names(s) <- c('L1', 'L2') #arranges the data for the regression
m <- lm(L2 ~ L1, data=s) #regression comparing matched raster cells
summary (m)
plot(L1, L2, xlab = 'ALS Z Max', ylab="DAP Z Max", main="DAP vs ALS")
abline(m, col="blue")
abline(0,1, col="red") #1 to 1 line
```

```
ZDiff <- (ALSrasters$zmax - DAPrasters$zmax)
plot(ZDiff, main="ALS zmax - DAP zmax")

#####Processing Tiles#####

?readLAScatalog
#folder you put the 2 Pack forest laz files from DNR
ALS <- readLAScatalog("Lab6/laz/")
las_check(ALS)
#DTM you downloaded from DNR
DTM <- rast("LAB6/dtm/pack_forest_2013_dtm_1.tif")
plot(DTM)
#we know lidR doesn't like feet...
projection(ALS) #checking crs
st_crs(ALS)<-2927 #changing crs
projection(DTM)
crs(DTM)<-"epsg:2927" #different to define the CRS for a raster

#this can take a bit. Make sure the DTM is located correctly
#mapview(DTM) #mapview doesn't currently work with terra
#check the location of pack forest tiles
plot(ALS,mapview=TRUE)

?`LAScatalog-class`
?normalize_height #extra steps are needed for working with a catalog
opt_chunk_size(ALS) <- 0 #the chunk size. 0 means the entire tile.
#location for output files. Files need to be written out of R due to size
opt_output_files(ALS) <- "Lab6/lazN/{ORIGINALFILENAME}_Norm"
#There are differnet naming conventions {ORIGINALFILENAME} keeps original file name
opt_laz_compression(ALS) <- TRUE #we want output as laz vs las
ALSN <- normalize_height(ALS, DTM) #Run normalize on the tiles using DTM you downloaded
#This will take a long time... Imagine if you did this for more than 2 tiles...
#in your "plots" window, it updates you on what tiles are being worked on.
plot(ALSN, mapview=TRUE)

#need to have the package 'future' installed and loaded
opt_chunk_size(ALSN) <- 0 #the chunk size. 0 means the entire tile.
#If output_files = "" outputs are returned in R
opt_output_files(ALSN) <- ""
#run rlas:::lasfilterusage() for list of filters
opt_select(ALSN) <- "xyzci"
opt_filter(ALSN) <- " -drop_z_below 0"
PackCC <- pixel_metrics(ALSN, ~sum(Z>6.56)/sum(Z>=0), 32.8)
plot(PackCC, main="Canopy Closure")
#mapview(PackCC, alpha.regions=0.2) #mapview is currently not working with terra rasters
?writeRaster
writeRaster(PackCC,filename =file.path("LAB6/PackCC.tif"), overwrite=TRUE)

#If output_files = "" outputs are returned in R
opt_output_files(ALSN) <- ""
#run rlas:::lasfilterusage() for list of filters
opt_filter(ALSN) <- " -drop_z_below 6.56"
#makeing our own function
f = function(x) {list(zq25 = quantile(x,0.25), mean = mean(x))}
PackZ <- pixel_metrics(ALSN, ~f(Z), 32.8)
plot(PackZ$zq25)
```

ESRM433/SEFS533

Lab 6

```
#####Predict with rasters and regression#####

field <-(read.csv("Lab5/plotdata.csv"))
lidar <- (read.csv("Lab5/cloudmetrics.csv"))
QMD <- field$QMD.GT5
mean <- lidar$Elev.mean
zq25 <- lidar$Elev.P25
Mreg <-lm(QMD ~ mean + zq25)
summary (Mreg)

?predict
#PackZ is our raster brick that contains rasters
#with names that match the coefficients in Mreg
QMDP <- predict(PackZ, Mreg) #new raster "QMDP"
plot (QMDP, main="QMD")
#mapview(QMDP, alpha.regions=0.2) #mapview is currently not working with Terra
writeRaster(QMDP,filename =file.path("LAB6/QMDP.tif"), overwrite=TRUE)

#####"Model Habitat" based on lidar#####

Model<-rast(QMDP)
Model[QMDP > 15 & PackCC > 0.7] = 1
Model[QMDP < 15 & PackCC < 0.7] = 0
plot(Model)
#mapview(Model, alpha.regions=0.2) #mapview is currently not working with Terra
writeRaster(Model,filename =file.path("LAB6/Model.tif"), overwrite=TRUE)
```