Introduction to LiDAR Data

Learning Objectives

After completing this tutorial, you will be able to:

- Use the extract() function to extract raster values using a vector extent or set of extents.

What you need

You will need a computer with internet access to complete this lesson and the data for week 4 of the course.

Download Week 3 Data (~250 MB){:data-proofer-ignore=" .btn }

```
# load libraries
library(raster)
library(rgdal)
library(ggplot2)
library(dplyr)

options(stringsAsFactors = FALSE)

# set working directory
# setwd("path-here/earth-analytics")
```

Import Canopy Height Model

First, we will import the NEON canopy height model. In the previous lessons / weeks we learned how to make this data product by subtracting the DEM from the DSM.

```
# import canopy height model (CHM).
SJER_chm <- raster("data/week4//california/SJER/2013/lidar/SJER_lidarCHM.tif")
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
SJER_chm
## class       : RasterLayer
## dimensions  : 5059, 4296, 21733464  (nrow, ncol, ncell)
## resolution  : 1, 1  (x, y)
## extent      : 254571, 258867, 4107303, 4112362  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=utm +zone=11 +datum=WGS84 +units=m +no_defs +ellps=WGS84 +towgs84=0,0,0
## data source : /Users/lewa8222/Documents/earth-analytics/data/week4/california/SJER/2013/lidar/SJER_li
## names       : SJER_lidarCHM
## values      : 0, 45.88  (min, max)

# plot the data
hist(SJER_chm,
     main="Histogram of Canopy Height\n NEON SJER Field Site",
     col="springgreen",
     xlab="Height (m)")
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## Warning in .hist1(x, maxpixels = maxpixels, main = main, plot = plot, ...):
## 0% of the raster cells were used. 100000 values used.
```

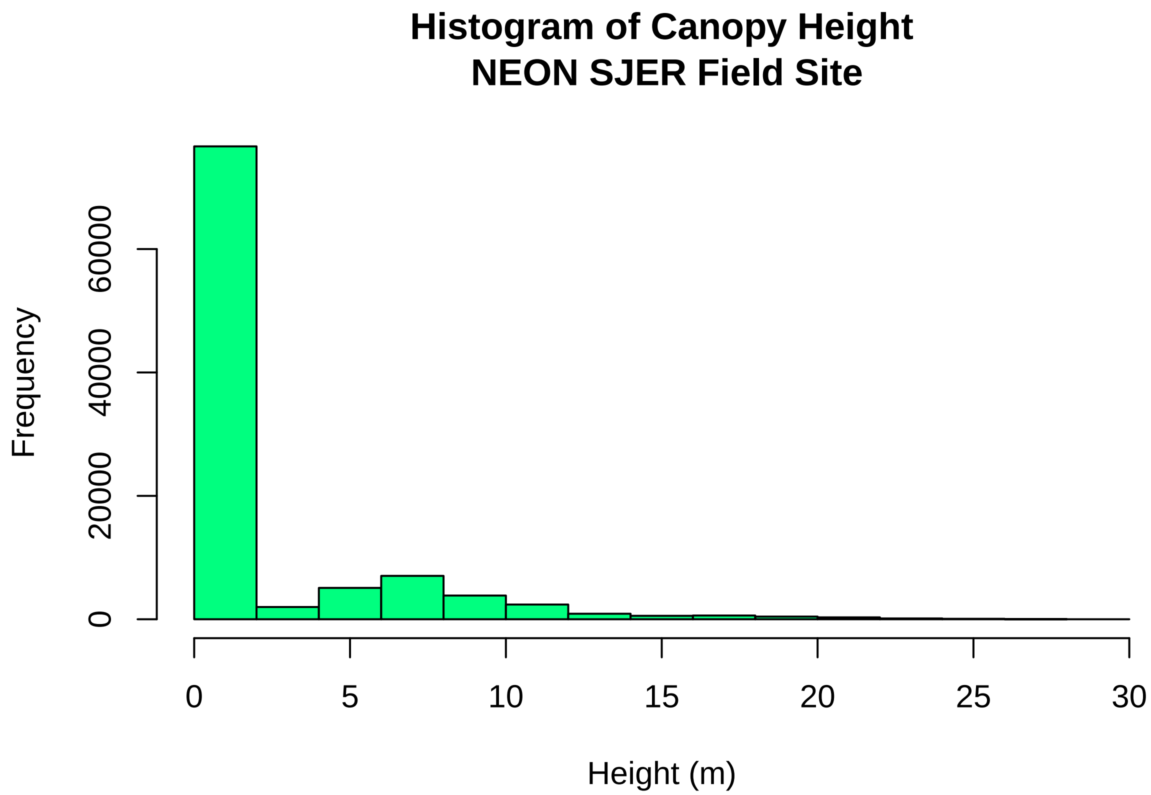

Figure 1: histogram of CHM values

```
# set values of 0 to NA as these are not trees
SJER_chm[SJER_chm==0] <- NA
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files

# plot the modified data
hist(SJER_chm,
     main="Histogram of Canopy Height\n NEON SJER Field Site, 0 values set to NA",
     col="springgreen",
     xlab="Height (m)")
```

Part 2. How does our CHM data compare to field measured tree heights?

We now have a canopy height model for our study area in California. However, how do the height values extracted from the CHM compare to our laboriously collected, field measured canopy height data? To figure this out, we will use *in situ* collected tree height data, measured within circular plots across our study area. We will compare the maximum measured tree height value to the maximum LiDAR derived height value for each circular plot using regression.

For this activity, we will use the a `csv` (comma separate value) file, located in `SJER/2013/insitu/veg_structure/D17_2013_SJ`

```
# import plot centroids
SJER_plots <- readOGR("data/week4/california/SJER/vector_data",
                      "SJER_plot_centroids")
## OGR data source with driver: ESRI Shapefile
```

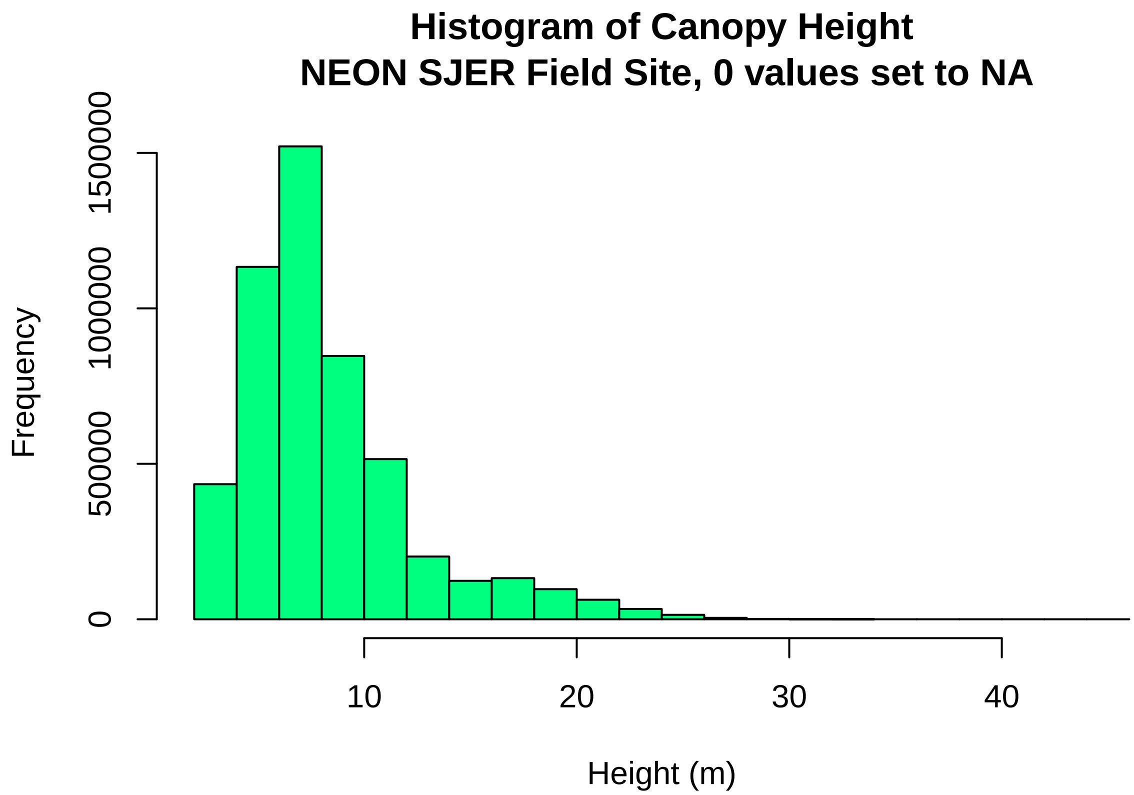

Figure 2: histogram of chm values

```
## Source: "data/week4/california/SJER/vector_data", layer: "SJER_plot_centroids"
## with 18 features
## It has 5 fields
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files


# Overlay the centroid points and the stem locations on the CHM plot
plot(SJER_chm,
     main="SJER  Plot Locations",
     col=gray.colors(100, start=.3, end=.9))
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files

# pch 0 = square
plot(SJER_plots,
     pch = 16,
     cex = 2,
     col = 2,
     add=TRUE)
```

Extract CMH data within 20 m radius of each plot centroid.

Next, we will create a boundary region (called a buffer) representing the spatial extent of each plot (where trees were measured). We will then extract all CHM pixels that fall within the plot boundary to use to

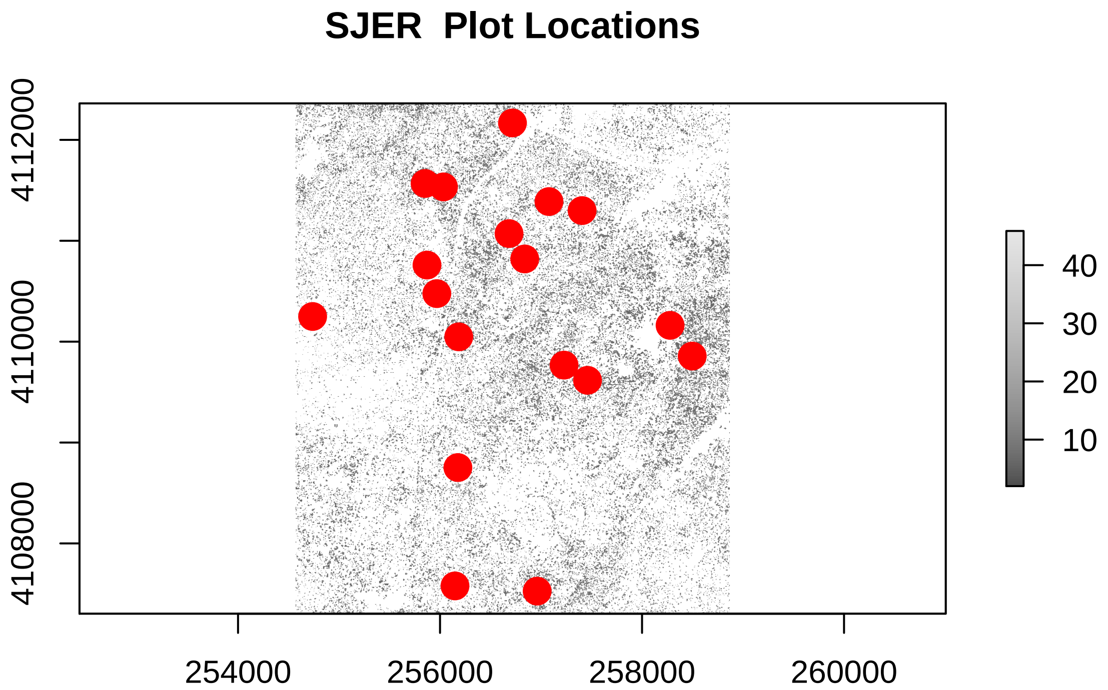

Figure 3: canopy height model / plot locations plot

estimate tree height for that plot.

There are a few ways to go about this task. If your plots are circular, then the extract tool will do the job!

```
<img src="~/Documents/Github/earthlab.github.io/images/course-materials/earth-analytics/week-4/buffer-c
<figcaption>The extract function in R allows you to specify a circular buffer
radius around an x,y point location. Values for all pixels in the specified
raster that fall within the circular buffer are extracted. In this case, we
will tell R to extract the maximum value of all pixels using the fun=max
command. Source: Colin Williams, NEON
</figcaption>
```

Extract Plot Data Using Circle: 20m Radius Plots

```
# Insitu sampling took place within 40m x 40m square plots, so we use a 20m radius.
# Note that below will return a dataframe containing the max height
# calculated from all pixels in the buffer for each plot
SJER_height <- extract(SJER_chm,
                    SJER_plots,
                    buffer = 20, # specify a 20 m radius
                    fun=mean, # extract the MEAN value from each plot
                    sp=TRUE, # create spatial object
                    stringsAsFactors=FALSE)
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
```

```
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
```

Explore The Data Distribution

If you want to explore the data distribution of pixel height values in each plot, you could remove the `fun` call to max and generate a list. `cent_ovrList <- extract(chm,centroid_sp,buffer = 20)`. It's good to look at the distribution of values we've extracted for each plot. Then you could generate a histogram for each plot `hist(cent_ovrList[[2]])`. If we wanted, we could loop through several plots and create histograms using a `for loop`.

```
# cent_ovrList <- extract(chm,centroid_sp,buffer = 20)
# create histograms for the first 5 plots of data
# for (i in 1:5) {
#  hist(cent_ovrList[[i]], main=(paste("plot",i)))
#  }
```

Variation 3: Derive Square Plot boundaries, then CHM values around a point

For how to extract square plots using a plot centroid value, check out the extracting square shapes activity .

`<img src="~/Documents/Github/earthlab.github.io/images/course-materials/earth-analytics/week-4/buffer-s`
`<figcaption>If you had square shaped plots, the code in the link above would`

extract pixel values within a square shaped buffer. Source: Colin Williams, NEON
</figcaption>

Extract descriptive stats from *In situ* Data

In our final step, we will extract summary height values from our field data. We will use the `dplyr` library to do this efficiently. We'll demonstrate both below

Extract stats from our spatial `data.frame` using the `DPLYR` package.

First let's see how many plots are in the centroid folder.

```
# import the centroid data and the vegetation structure data
SJER_insitu <- read.csv("data/week4/california/SJER/2013/insitu/veg_structure/D17_2013_SJER_vegStr.csv"
                        stringsAsFactors = FALSE)

# get list of unique plots
unique(SJER_plots$Plot_ID)
##  [1] "SJER1068" "SJER112"  "SJER116"  "SJER117"  "SJER120"  "SJER128"
##  [7] "SJER192"  "SJER272"  "SJER2796" "SJER3239" "SJER36"   "SJER361"
## [13] "SJER37"   "SJER4"    "SJER8"    "SJER824"  "SJER916"  "SJER952"
```

Extract Max Tree Height

Next, find the maximum MEASURED tree height value for each plot. This value represents the tallest tree in each plot. We will compare this value to the max lidar CHM value.

```
# find the max and averagestem height for each plot
insitu_stem_height <- SJER_insitu %>%
  group_by(plotid) %>%
  summarise(insitu_max = max(stemheight), insitu_avg = mean(stemheight))

head(insitu_stem_height)
## # A tibble: 6 × 3
##     plotid insitu_max insitu_avg
##      <chr>      <dbl>      <dbl>
## 1 SJER1068       19.3   3.866667
## 2  SJER112       23.9   8.221429
## 3  SJER116       16.0   8.218750
## 4  SJER117       11.0   6.512500
## 5  SJER120        8.8   7.600000
## 6  SJER128       18.2   5.211765

# let's create better, self documenting column headers
names(insitu_stem_height) <- c("plotid", "insituMaxHt")
head(insitu_stem_height)
## # A tibble: 6 × 3
##     plotid insituMaxHt       NA
##      <chr>       <dbl>    <dbl>
## 1 SJER1068        19.3 3.866667
## 2  SJER112        23.9 8.221429
```

```
## 3  SJER116        16.0 8.218750
## 4  SJER117        11.0 6.512500
## 5  SJER120         8.8 7.600000
## 6  SJER128        18.2 5.211765
```

Merge InSitu Data With Spatial data.frame

Once we have our summarized insitu data, we can `merge` it into the centroids `data.frame`. Merge requires two data.frames and the names of the columns containing the unique ID that we will merge the data on. In this case, we will merge the data on the plot_id column. Notice that it's spelled slightly differently in both data.frames so we'll need to tell R what it's called in each data.frame.

```
# merge the insitu data into the centroids data.frame
SJER_height <- merge(SJER_height,
                     insitu_stem_height,
                   by.x = 'Plot_ID',
                   by.y = 'plotid')

SJER_height@data
##     Plot_ID  Point northing  easting plot_type SJER_lidarCHM insituMaxHt
## 1  SJER1068 center  4111568 255852.4     trees     11.544348        19.3
## 2   SJER112 center  4111299 257407.0     trees     10.355685        23.9
## 3   SJER116 center  4110820 256838.8     grass      7.511956        16.0
## 4   SJER117 center  4108752 256176.9     trees      7.675347        11.0
## 5   SJER120 center  4110476 255968.4     grass      4.591176         8.8
## 6   SJER128 center  4111389 257078.9     trees      8.979005        18.2
## 7   SJER192 center  4111071 256683.4     grass      7.240118        13.7
## 8   SJER272 center  4112168 256717.5     trees      7.103862        12.4
## 9  SJER2796 center  4111534 256034.4      soil      6.405240         9.4
## 10 SJER3239 center  4109857 258497.1      soil      6.009128        17.9
## 11   SJER36 center  4110162 258277.8     trees      6.516288         9.2
## 12  SJER361 center  4107527 256961.8     grass     13.899027        11.8
## 13   SJER37 center  4107579 256148.2     trees      7.109851        11.5
## 14    SJER4 center  4109767 257228.3     trees      5.032620        10.8
## 15    SJER8 center  4110249 254738.6     trees      3.024286         5.2
## 16  SJER824 center  4110048 256185.6      soil      7.738203        26.5
## 17  SJER916 center  4109617 257460.5      soil     11.181955        18.4
## 18  SJER952 center  4110759 255871.2     grass      4.149286         7.7
##          NA
## 1  3.866667
## 2  8.221429
## 3  8.218750
## 4  6.512500
## 5  7.600000
## 6  5.211765
## 7  6.769565
## 8  6.819048
## 9  5.085714
## 10 3.920833
## 11 9.200000
## 12 2.451429
## 13 7.350000
## 14 5.910526
```

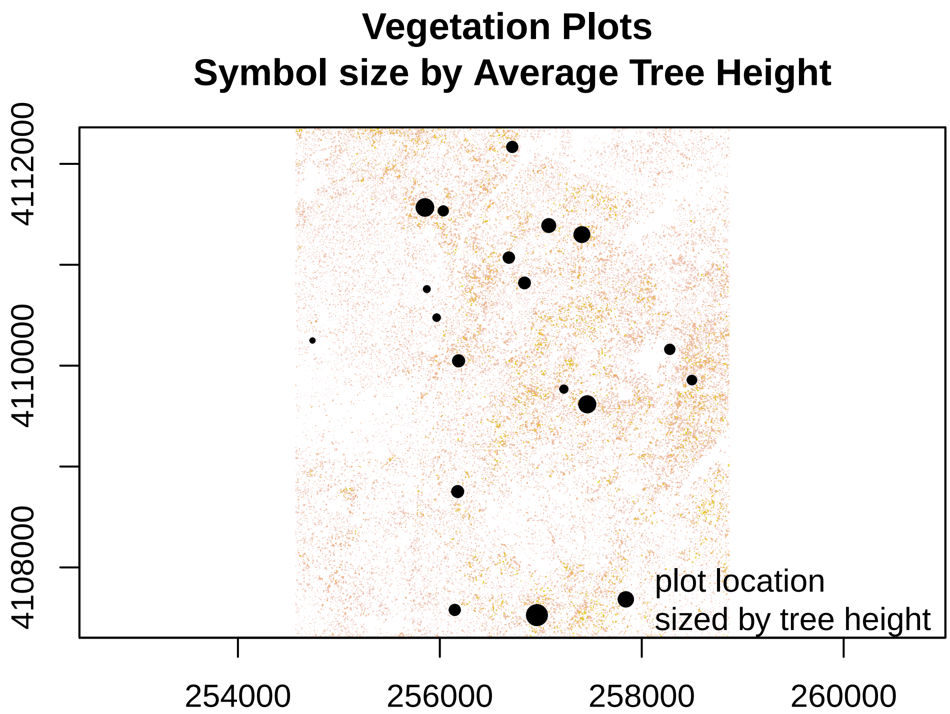

Figure 4: Plots sized by vegetation height

```
## 15 1.057143
## 16 5.357895
## 17 5.791667
## 18 1.558333
```

plot by height

```
# plot canopy height model
plot(SJER_chm,
     main="Vegetation Plots \nSymbol size by Average Tree Height",
     legend=F)
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files

# add plot location sized by tree height
plot(SJER_height,
     pch=19,
     cex=(SJER_height$SJER_lidarCHM)/10, # size symbols according to tree height attribute normalized b
     add=T)

legend('bottomright',
       legend="plot location \nsized by tree height",
       pch=19,
       bty='n')
```

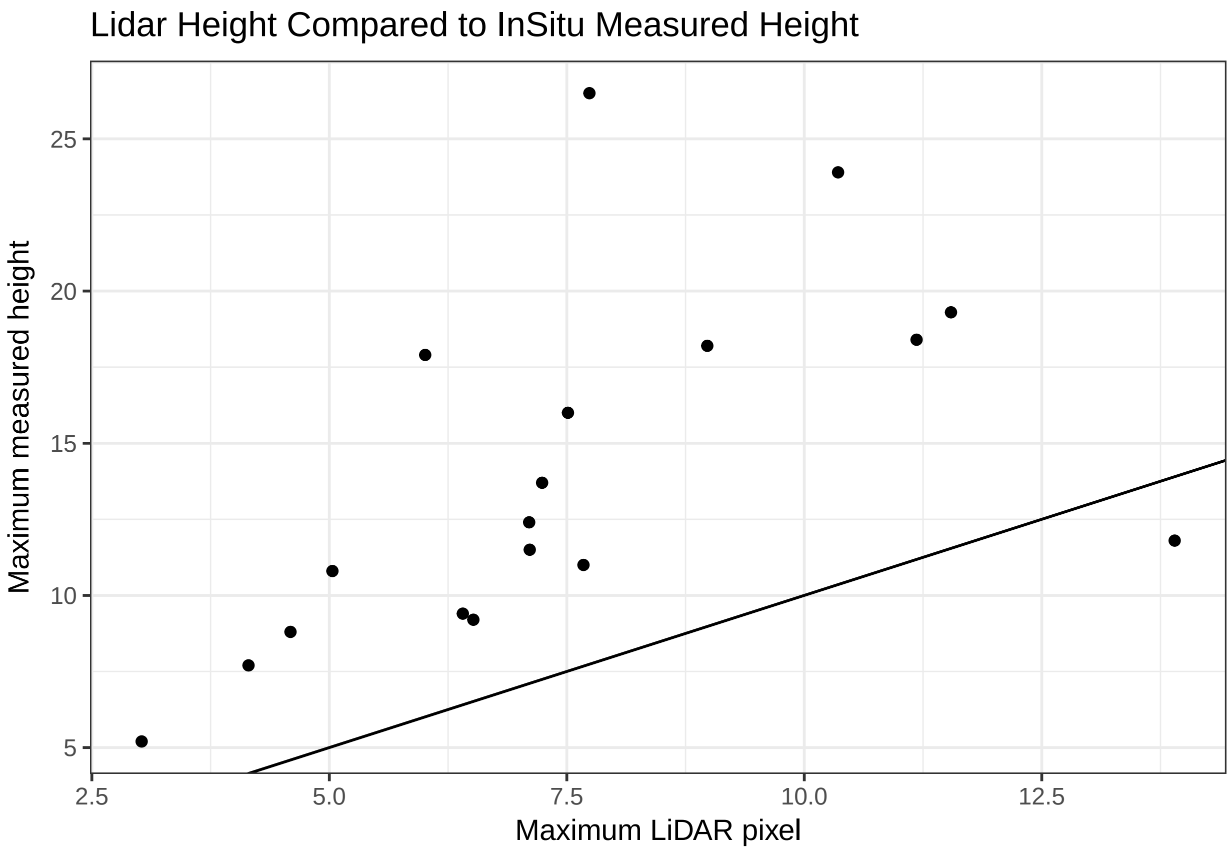

Figure 5: ggplot - measured vs lidar chm.

Plot Data (CHM vs Measured)

Let's create a plot that illustrates the relationship between in situ measured max canopy height values and lidar derived max canopy height values.

```
# create plot
ggplot(SJER_height@data, aes(x=SJER_lidarCHM, y = insituMaxHt)) +
  geom_point() +
  theme_bw() +
  ylab("Maximum measured height") +
  xlab("Maximum LiDAR pixel")+
  geom_abline(intercept = 0, slope=1) +
  ggtitle("Lidar Height Compared to InSitu Measured Height")
```

We can also add a regression fit to our plot. Explore the GGPLOT options and customize your plot.

```
# plot with regression fit
p <- ggplot(SJER_height@data, aes(x=SJER_lidarCHM, y = insituMaxHt)) +
  geom_point() +
  ylab("Maximum Measured Height") +
  xlab("Maximum LiDAR Height")+
  geom_abline(intercept = 0, slope=1)+
  geom_smooth(method=lm)
```

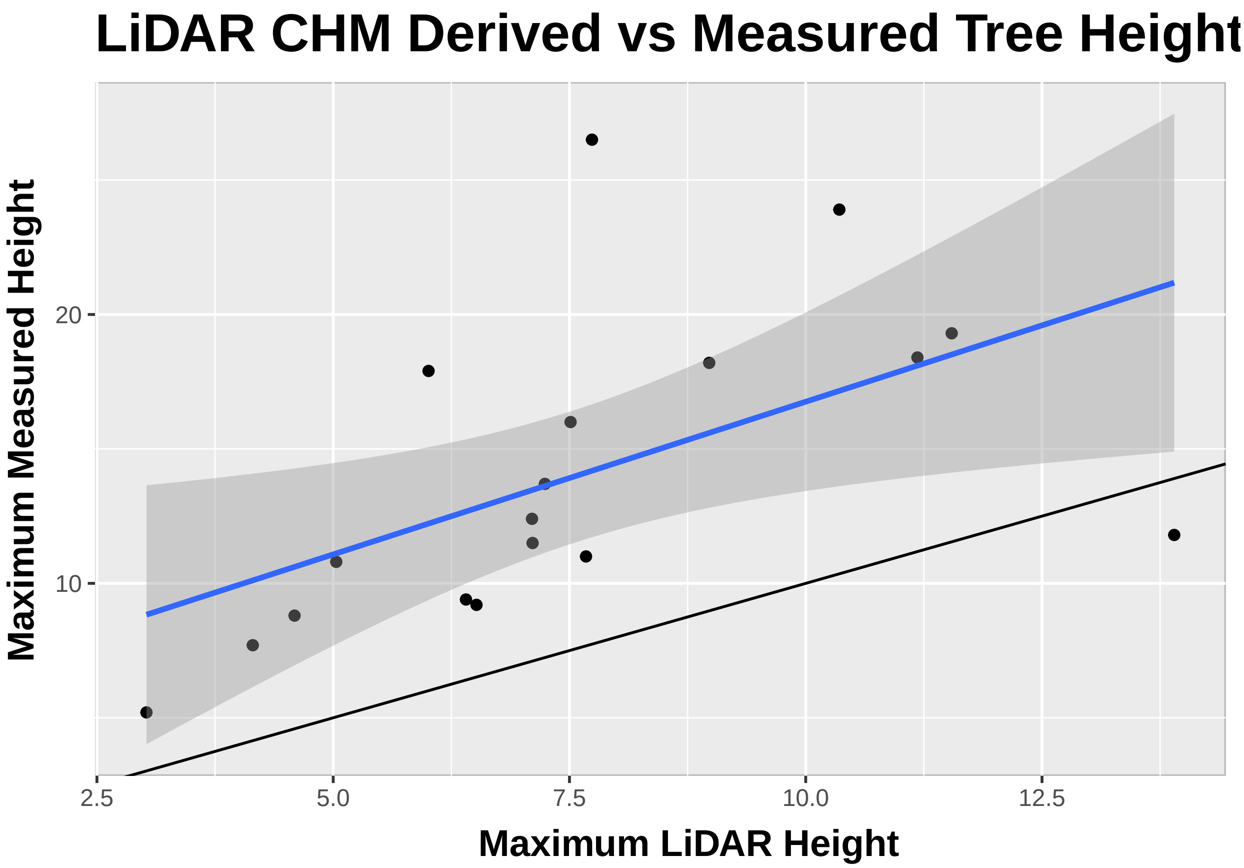

Figure 6: Scatterplot measured height compared to lidar chm.

```
p + theme(panel.background = element_rect(colour = "grey")) +
  ggtitle("LiDAR CHM Derived vs Measured Tree Height") +
  theme(plot.title=element_text(family="sans", face="bold", size=20, vjust=1.9)) +
  theme(axis.title.y = element_text(family="sans", face="bold", size=14, angle=90, hjust=0.54, vjust=1))
  theme(axis.title.x = element_text(family="sans", face="bold", size=14, angle=00, hjust=0.54, vjust=-.
```

View Differences

```
SJER_height@data$ht_diff <-  (SJER_height@data$SJER_lidarCHM - SJER_height@data$insituMaxHt)

# base plot example below
# barplot(SJER_height@data$ht_diff,
#         xlab = SJER_height@data$Plot_ID)


# create bar plot using ggplot()
ggplot(data=SJER_height@data, aes(x=Plot_ID, y=ht_diff, fill=Plot_ID)) +
    geom_bar(stat="identity")
```

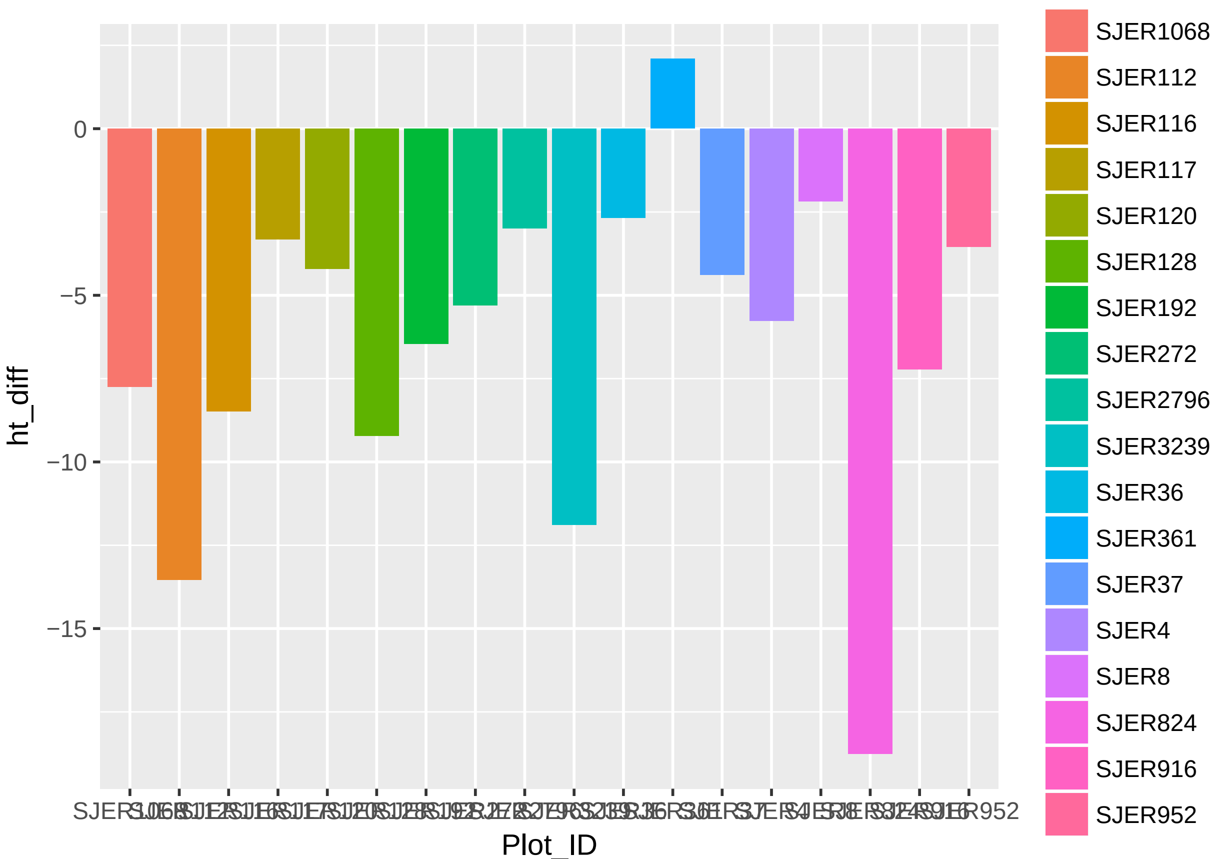

Figure 7: box plot showing differences between chm and measured heights.

QGIS Check

Here's a link to add imagery to QGIS. Add Imagery to QGIS

You have now successfully created a canopy height model using LiDAR data AND compared LiDAR derived vegetation height, within plots, to actual measured tree height data!

Challenge: LiDAR vs Insitu Comparison

Create a plot of LiDAR 95th percentile value vs *insitu* max height. Or LiDAR 95th percentile vs *insitu* 95th percentile. Add labels to your plot. Customize the colors, fonts and the look of your plot. If you are happy with the outcome, share your plot in the comments below!

Create Plot.ly Interactive Plot

Plot.ly is a free to use, online interactive data viz site. If you have the plot.ly library installed, you can quickly export a ggplot graphic into plot.ly! (NOTE: it also works for python matplotlib)!! To use plot.ly, you need to setup an account. Once you've setup an account, you can get your key from the plot.ly site (under Settings > API Keys) to make the code below work.

You must be signed into plot.ly online, from your current computer, at the time you use the `plotly_POST` command to upload you plot to your plot.ly account.

```
library(plotly)

# you must be signed into Plot.ly online on the same computer for this code to work.
# generate the plot
ggplotly(p,
        filename='NEON SJER CHM vs Insitu Tree Height')
```

Check out the results!

NEON Remote Sensing Data compared to NEON Terrestrial Measurements for the SJER Field Site