Lecture 21

Raster Analysis 1: Continuous Variables

GEOG 489

SPRING 2020

Raster Analysis: Geophysical Variable

Geophysical variables are variables we want to **map** using geospatial data. They are (hopefully) characteristics of the Earth's surface.

Nominal: variables representing discrete categories or classes, e.g. "Land cover class".

Continuous: variables representing ranges of values, e.g. "Percent grass cover".

Raster Models

Geophysical variable maps derived from geospatial data are created through the use of raster models, where:

geophysical variable = f(geospatial data)

If the function ***f*** is known, this function is applied to all pixels (or groups of pixels) in the image, thus producing an estimate of the geophysical variable for every location.

Raster models include continuous variable models and classification models

Continuous Variable Models

Continuous variable models to produce continuous geophysical variables can take many forms, the simplest being a linear relationship.

*percent cover of grass = 1.25 * NIR reflectance*

Thus, if a pixel has a NIR reflectance of 40%, this model estimates the percent cover of grass of that pixel as 1.25 * 40% = 50% covered in grass.

Continuous Variable Models

Continuous variable extraction: How much of (some geophysical variable of interest) is present in a pixel?

Methodological overview:

1. Collect field data and position of variable of interest.
2. Determine empirical relationship between geospatial data to field data.
   - Relationship determination can take an extremely wide range of methods, from regression to neural network to complex model formulation, etc…
3. Invert relationship on entire raster scene.
4. Determine accuracy of results.

Types of Continuous Variable Models

Linear regression (1 variable)

Multiple linear regression (multiple variables)

Generalized additive models (GAMs, can be non-linear)

Regression trees and randomForest (machine learning)

Classification Models

Classification models to produce nominal variables are **rulesets** used to split geospatial responses into different classes. A simple model might be:

Classes: vegetated and non-vegetated

Ruleset:

If NIR reflectance > 50%, then class = vegetated

If NIR reflectance <= 50%, then class = non-vegetated

Thus, if a pixel has a NIR reflectance of 40%, this pixel would be classified as "non-vegetated".

Case Study: Tree Height

Question: what is the tree height in the Tahoe high resolution imagery?

Methods:

1. Collect tree heights of sampling locations and GPS coordinates of those trees.
2. Perform a linear regression (or other empirical models) between the field measured tree height and spectral features derived from raster data at pixel locations where the field data was collected.
3. The regression gives a model (f) of tree height=f(raster data), so we can apply the model to an entire raster scene, and each pixel will be the estimated tree height.
4. Evaluate the model performance using independent testing samples

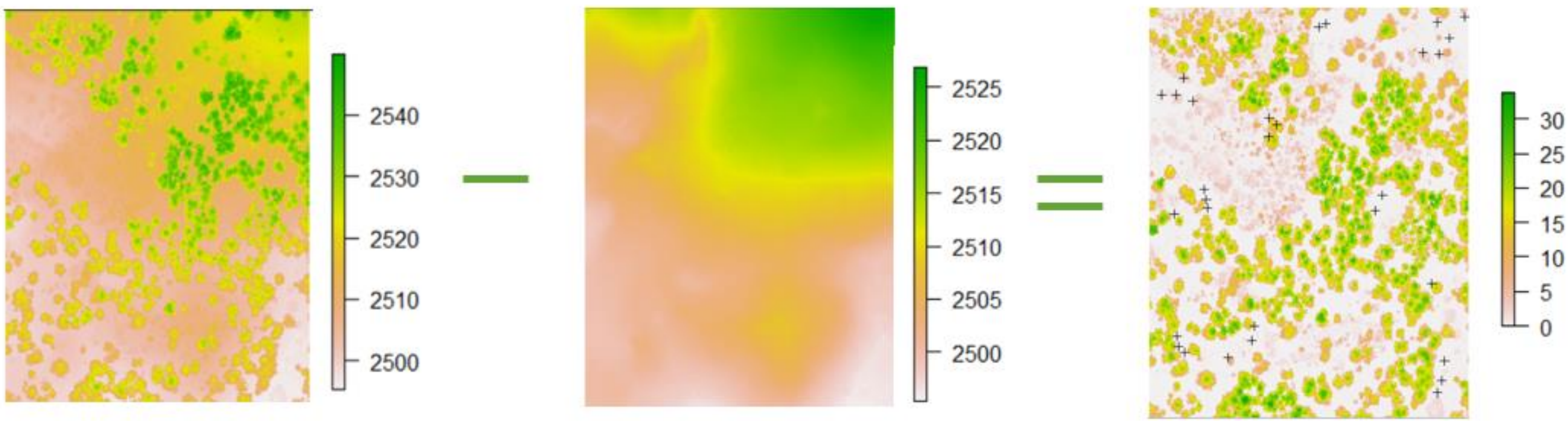
Case Study: Tree Height

Question: what is the tree height in the Tahoe high resolution imagery?

Methods:

1. Collect tree heights of sampling locations and GPS coordinates of those trees.

Response variable:



tahoe_lidar_highesthit.tif: **elevation of tree canopy**

tahoe_lidar_bareearth.tif: **elevation of ground surface**

Tree height

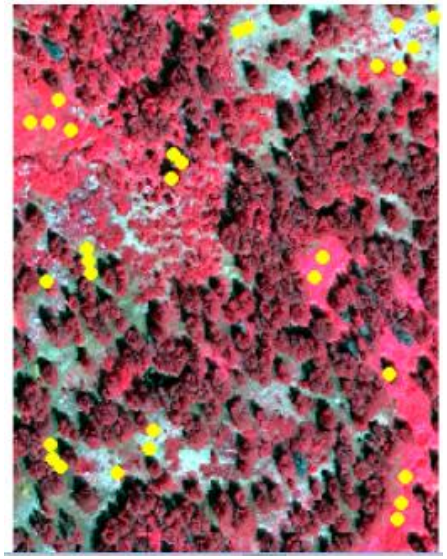
Case Study: Tree Height

Question: what is the tree height in the Tahoe high resolution imagery?

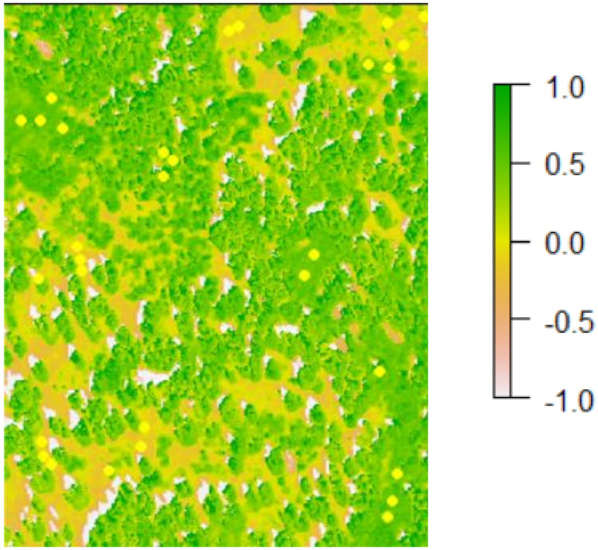
Methods:

1. Collect tree heights of sampling locations and GPS coordinates of those trees.

Independent variable (high resolution imagery):



tahoe_highrez.tif:
High resolution imagery (3 layers)



tahoe_ndvi:
NDVI (vegetation index) derived from high resolution imagery

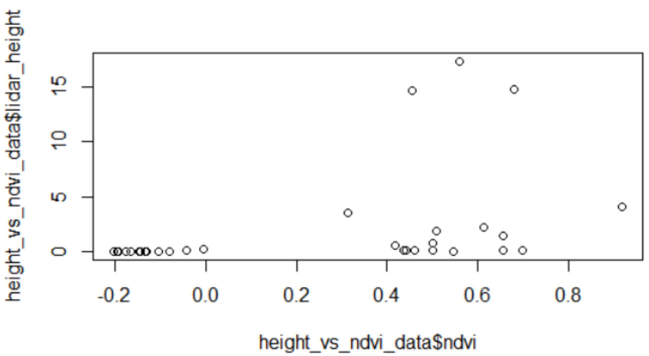
Case Study: Tree Height

Question: what is the tree height in the Tahoe high resolution imagery?

Methods:

2. Perform a linear regression (or other empirical models) between the field measured tree height and spectral features derived from raster data at pixel locations where the field data was collected.

```
> height_vs_ndvi_data
  ID.1       ndvi ID.2 lidar_height
1    1 0.55868545    1  17.32006836
2    2 0.31474104    2   3.60009766
3    3 0.68000000    3  14.77001953
4    4 0.45544554    4  14.66992188
5    5 0.91935484    5   4.07006836
```




```
# Extract the Lidar height and NDVI for 30 points
lidar_height_extract <- extract(lidar_height,
    tahoe_highrez_training_points,df=TRUE)

ndvi_extract <- extract(tahoe_ndvi,
    tahoe_highrez_training_points,df=TRUE)
```

Case Study: Tree Height

Question: what is the tree height in the Tahoe high resolution imagery?

Simple linear regression model

Linear regression: response variable ~ explanatory variable (tree height ~ ndvi)

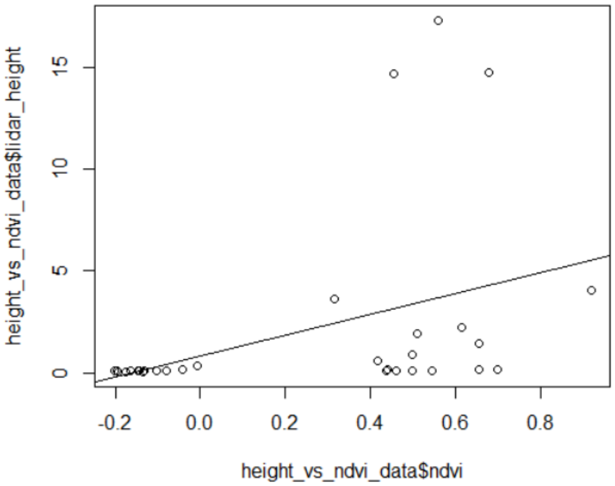
ndvi_height_lm <- lm(height_vs_ndvi_data$lidar_height ~ height_vs_ndvi_data$ndvi)

```
Call:
lm(formula = height_vs_ndvi_data$lidar_height ~ height_vs_ndvi_data$ndvi)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.2288 -2.6739 -0.3861  0.1059 13.6461 

Coefficients:
                         Estimate Std. Error t value Pr(>|t|)  
(Intercept)                0.8195     0.9867   0.831   0.4132  
height_vs_ndvi_data$ndvi   5.1093     2.2535   2.267   0.0313 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.395 on 28 degrees of freedom
Multiple R-squared:  0.1551,	Adjusted R-squared:  0.1249 
F-statistic:  5.14 on 1 and 28 DF,  p-value: 0.03129
```



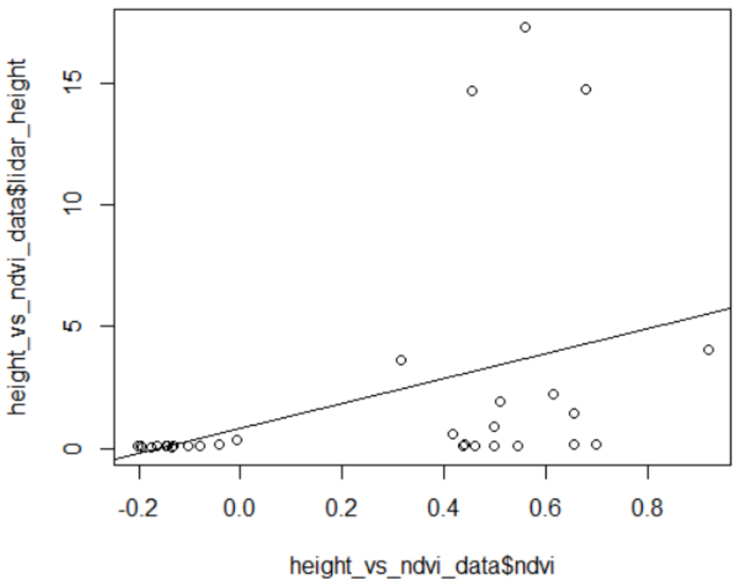
(tree height = 5.1093*ndvi + 0.8195)

Continuous Variable Models

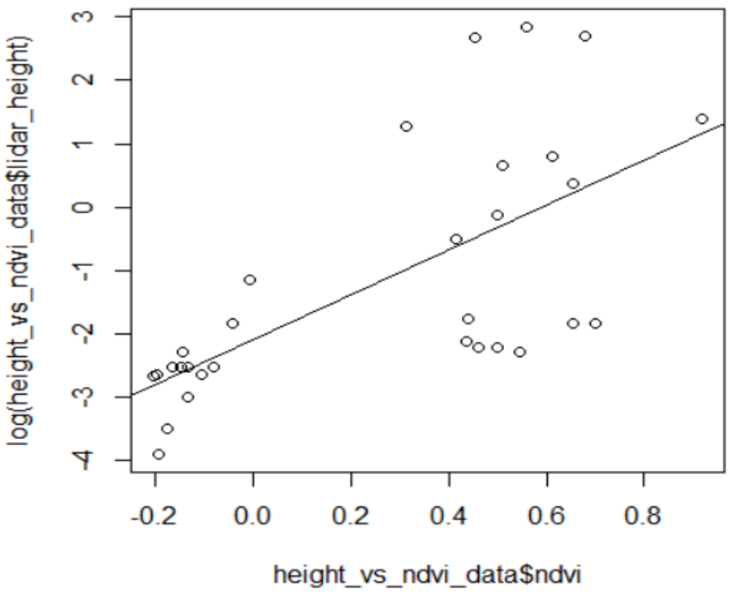
To improve the simple linear regression model (tree height ~ ndvi)

1) Transform variables in regression model (e.g. log transform the height data)

ndvi_height_lm_loght <- lm(log(lidar_height) ~ ndvi, data=height_vs_ndvi_data)



Simple linear regression model (R square = 0.16)



Log transform the height data (R square = 0.45)

Continuous Variable Models

To improve the simple linear regression model (tree height ~ ndvi)

2) Multiple regression (tree height ~ ndvi + b1 + b2 + b3)

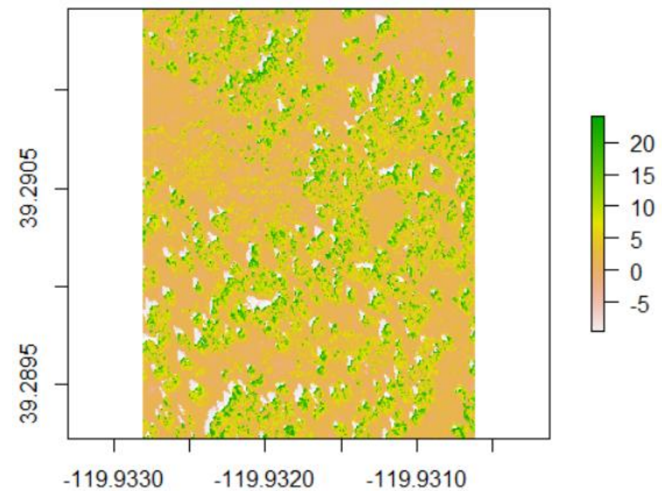
ndvi_height_lm_multi <- lm(lidar_height ~ ndvi + B1 + B2 + B3,
data=height_vs_ndvi_vs_allbands_data)

```
Call:
lm(formula = lidar_height ~ ndvi + B1 + B2 + B3, data = height_vs_ndvi_vs_allbands_data)

Residuals:
    Min      1Q  Median      3Q     Max
-9.6603 -1.1244 -0.6391  0.0880 10.7057

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.254104   3.774431   1.922   0.0661 .
ndvi        16.718849   6.337273   2.638   0.0141 *
B1          -0.077743   0.029425  -2.642   0.0140 *
B2          -0.001967   0.056091  -0.035   0.9723
B3           0.040778   0.058617   0.696   0.4931
---
Signif. codes:  0 ?**?0.001 ?*?0.01 ??0.05 ??0.1 ??1

Residual standard error: 3.79 on 25 degrees of freedom
Multiple R-squared:  0.4388,    Adjusted R-squared:  0.3491
F-statistic: 4.888 on 4 and 25 DF,  p-value: 0.004737
```



Estimated tree height

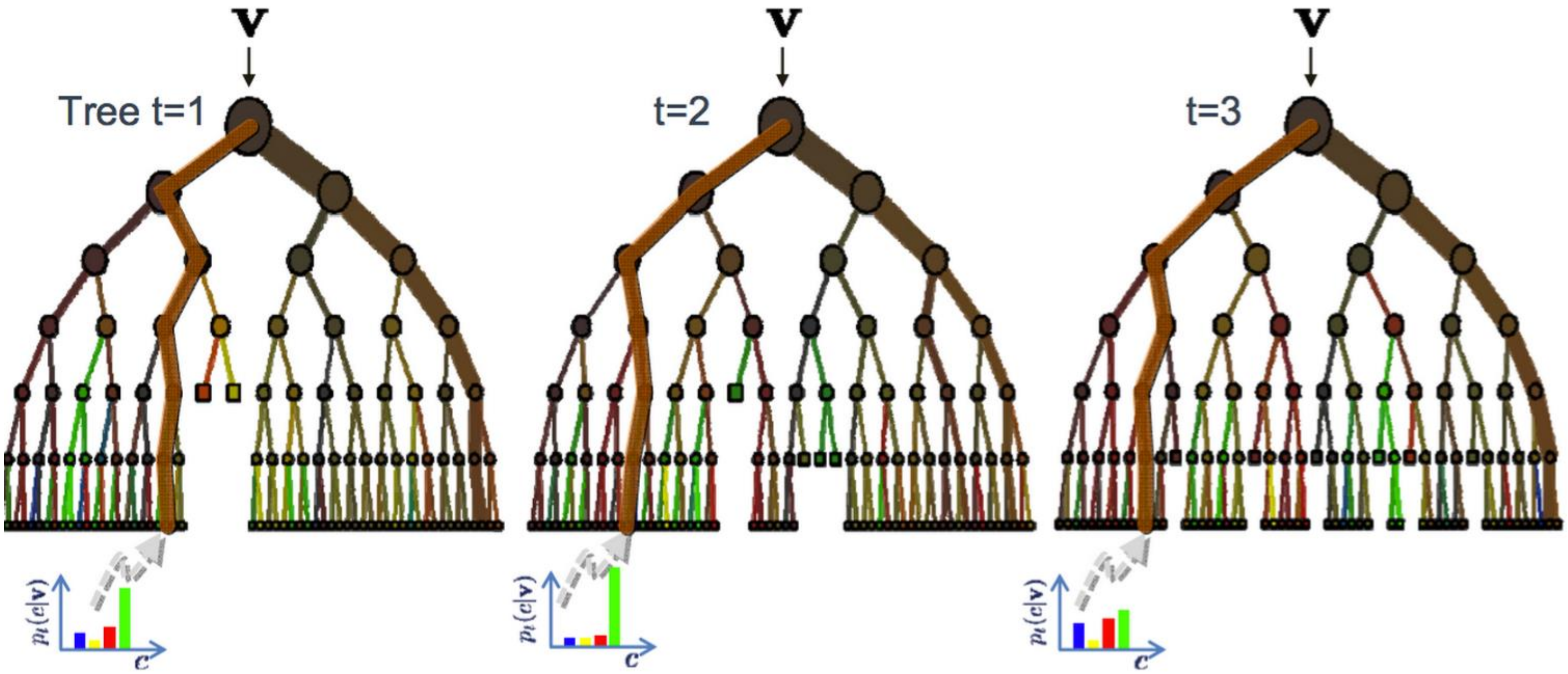
Multiple linear regression model (R square = 0.44)

Continuous Variable Models

To improve the simple linear regression model (tree height ~ ndvi)

3) Random Forest (non-linear relationship)

tahoe_height_rf <- randomForest(lidar_height ~ ndvi + B1 + B2 + B3,
data=height_vs_ndvi_vs_allbands_data)



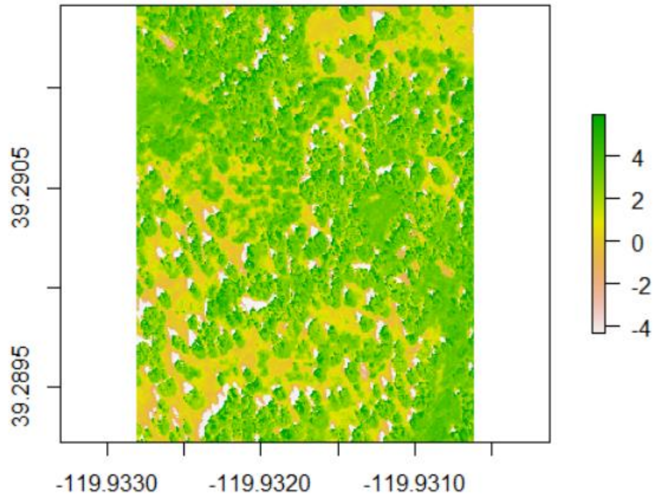
Case Study: Tree Height

Question: what is the tree height in the Tahoe high resolution imagery?

Methods:

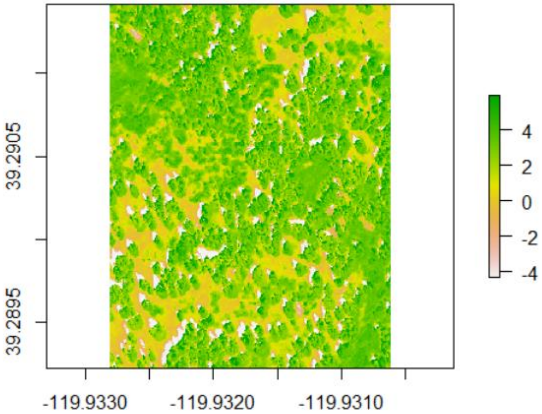
3. The regression gives a model (f) of (tree height = 5.1093*ndvi + 0.8195), so we can apply the model to an entire raster scene, and each pixel will be the estimated tree height.

tahoe_height_pred <- predict(tahoe_ndvi,ndvi_height_lm)

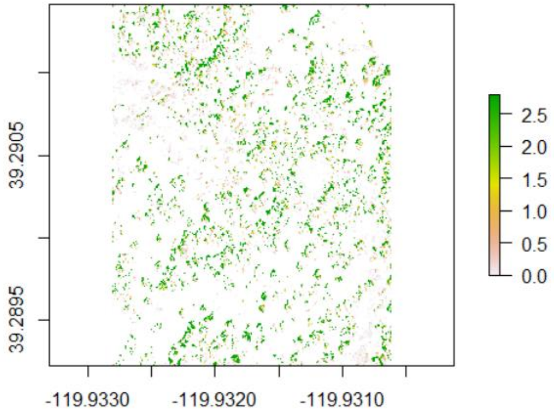


Simple regression: estimated tree height at every pixel location

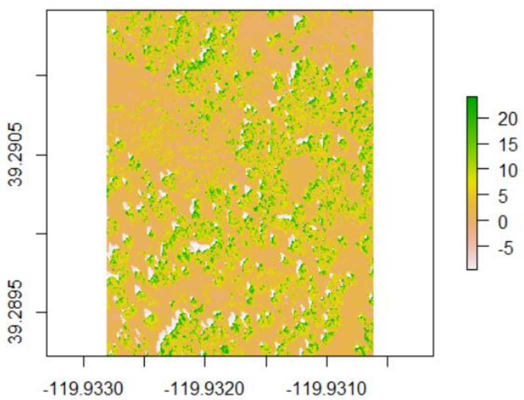
Case Study: Tree Height - Prediction



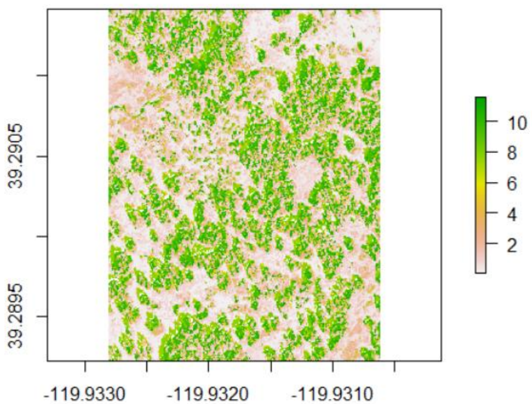
Simple regression



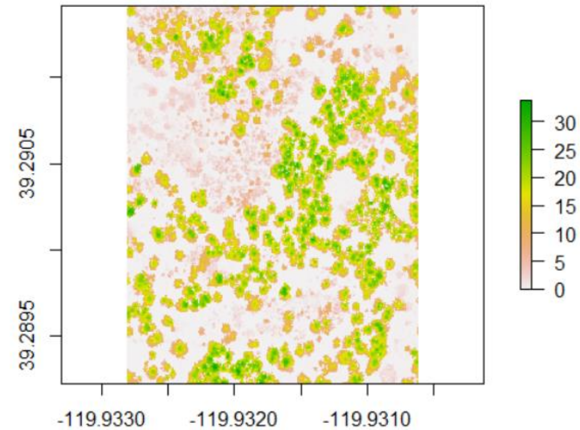
Transformed variable



Multiple regression



Random forest



LIDAR- derived tree height

Case Study: Tree Height

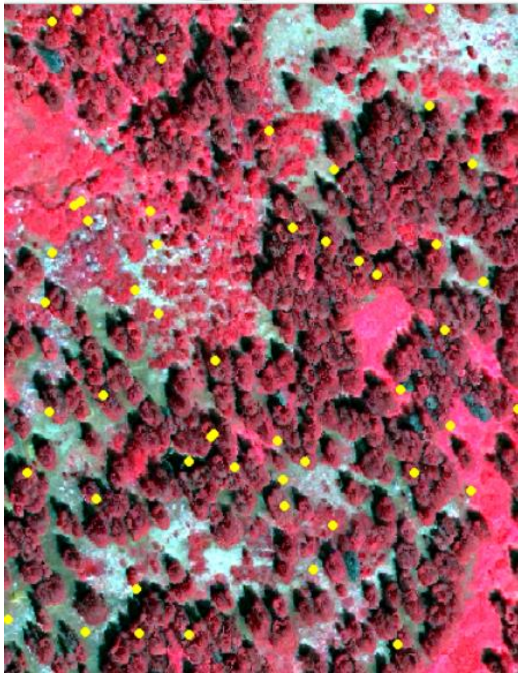
Question: what is the tree height in the Tahoe high resolution imagery?

Methods:

4. Evaluate the model performance using independent testing samples

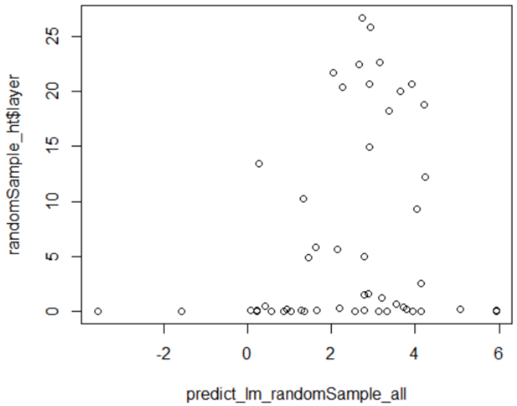
```
# Randomly choose 50 points to test:
randomSample <- sampleRandom(tahoe_multi,
                             size=50,sp=TRUE)

# Use samples to extract the height data
randomSample_ht <- extract(lidar_height,
                           randomSample,df=TRUE)
```

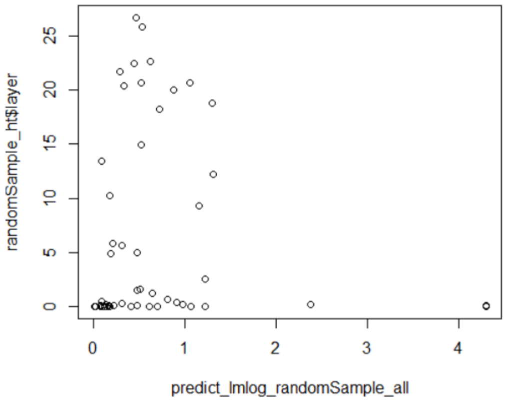


Case Study: Tree Height - Evaluation

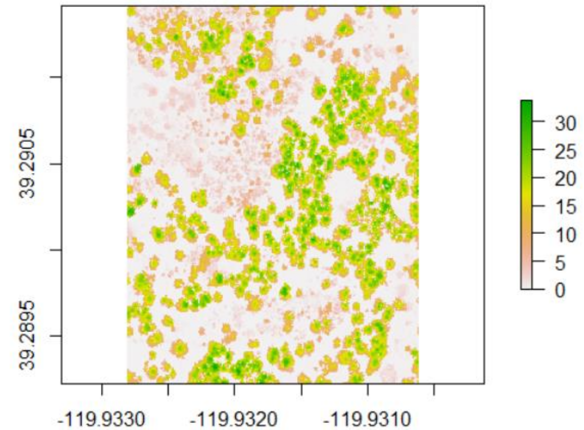
Predicted vs Estimated tree height



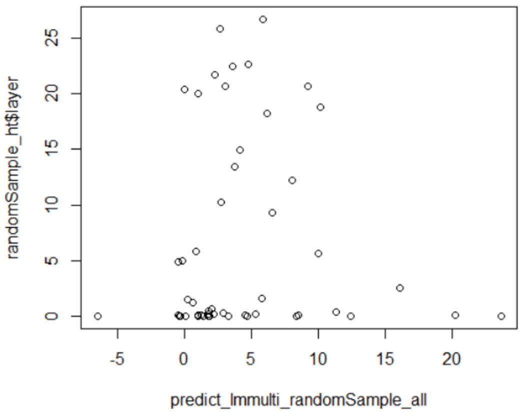
Simple regression (correlation is 0.33)



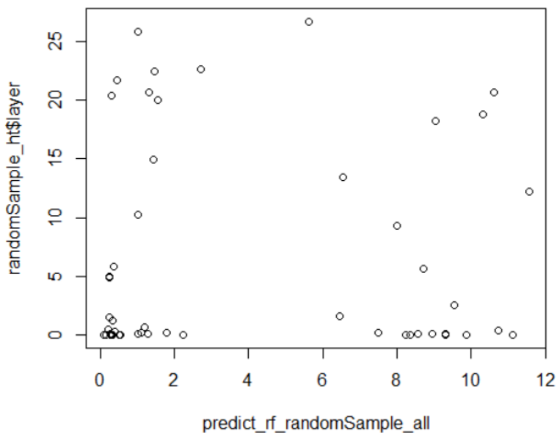
Transformed variable (correlation is 0.04)



LIDAR- derived tree height



Multiple regression (correlation is 0.15)



Random forest (correlation is 0.17)

Summary

R provides many ways to perform continuous variable extraction. The basic order of ops is typically:

1) Collect training and testing data

2) Extract pixel values at the location of the training data.

3) Perform transforms on the pixel values (e.g. calculate NDVI and other indices).

4) Try several models out with different input predictors.

5) Extract pixel values at the location of the testing data.

6) Apply models to test data to predict the variable.

7) Compare the predicted variable vs. the measured variable for the various models and predictor combinations.

8) Pick the best model and...

9) Apply (predict) the model to the raster.

Assignment 6 (Optional)

Your goal is to manipulate SpatialLines and SpatialPolygons and perform simple plotting using auckland dataset

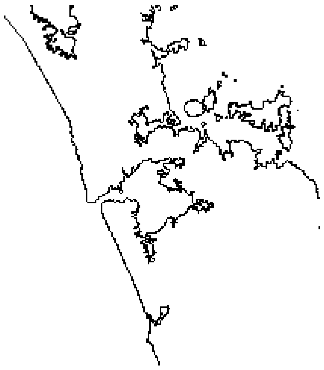
Requirements:

1) The function should be named "plotLinesOrPolygons" and have the following parameters:

lines_vector : an input SpatialLines object (assumption: one Line per Lines object)

whichPlot: a character value that can be "lines" or "polygons" or (for the extra credit) "linesToPolygons". The default should be "lines".

filename: an output pdf filename, should default to "lines.pdf"



Assignment 6 (Optional)

Your goal is to manipulate SpatialLines and SpatialPolygons and perform simple plotting using auckland dataset

Requirements:

2) If whichPlot=="lines", the function should subset out all Lines objects which CANNOT be polygons (1st coordinate does not match last coordinate).

3) If whichPlot=="polygons", the function should subset out all Lines objects which CAN be polygons (1st coordinate matches the last coordinate), and converts these to a SpatialPolygons object.

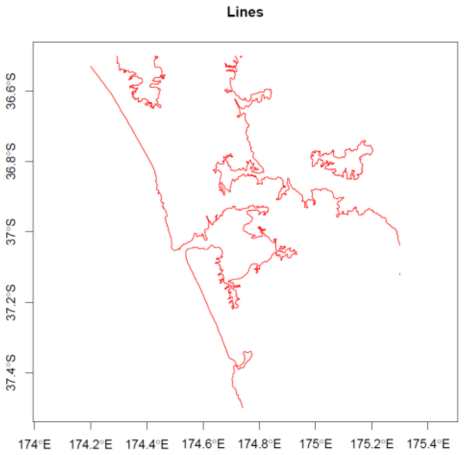
4) (Extra Credit) If whichplot=="linesToPolygons", the function should subset out all of the lines (requirement #3), and add a new node to each line that equals the first node, and then convert these lines to a SpatialPolygons object.

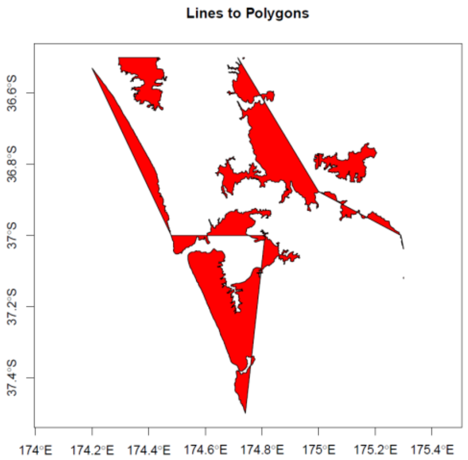
Assignment 6 (Optional)

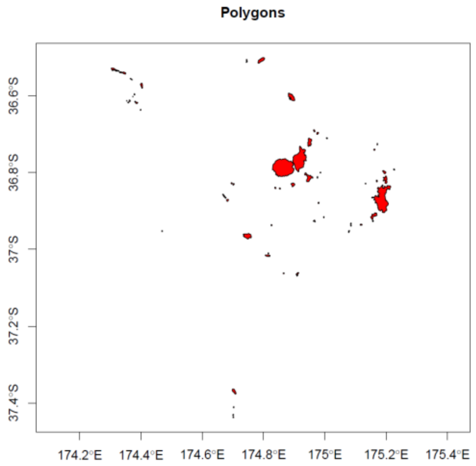
Your goal is to manipulate SpatialLines and SpatialPolygons and perform simple plotting using auckland dataset

Requirements:

5) Whatever subset the whichplot creates, it should be plotted using the default axes. SpatialLines should be plotted in red, and SpatialPolygons should be plotted with the outline in black and filled with red. These plots should be saved to the pdf filename.







Assignment 6 (Optional)

Your goal is to manipulate SpatialLines and SpatialPolygons and perform simple plotting using auckland dataset

Requirements:

6) The function should return the subset.

7) You may use any packages used in the class. In all likelihood, you should only need sp and rgdal, and maptools to run the example.

8) Comment your code in at least 3 places.

9) The code should be submitted tom Compass 2g as a single function with the filename: LastName-FirstName-geog489-s20-assignment-06.R

and should have at the top:
[Your name]
Assignment #6.

Assignment 6 is due the coming Thursday, April 23, 2020 at midnight.