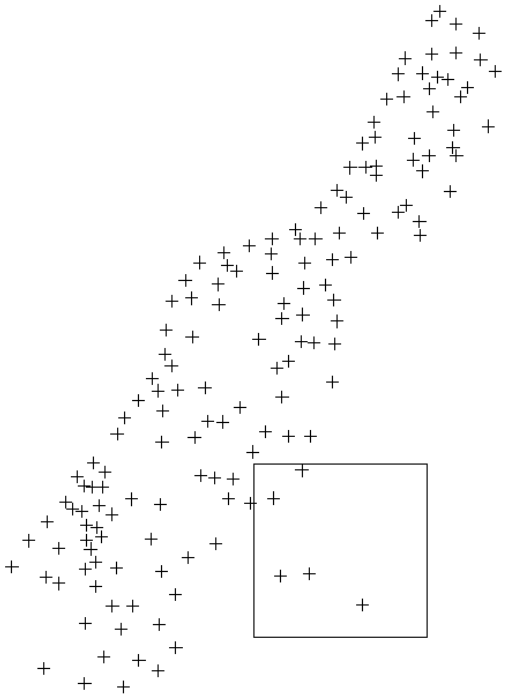
Quiz 8

Create a square polygon data frame with its upper left corner being (180000, 331000), its edge length being 1000, and its attribute being its area. Get the index of the meuse data points that are within the polygon.



Quiz 8 will be due by the end of April 15

Lecture 20

Vector/Raster Fusion

GEOG 489

SPRING 2020

Overlay and spatial query

Source layer: the layer that is used to determine the selection from the target layer based on some spatial relationship.

Target layer: the layer to extract data FROM

R function: over(x, y)

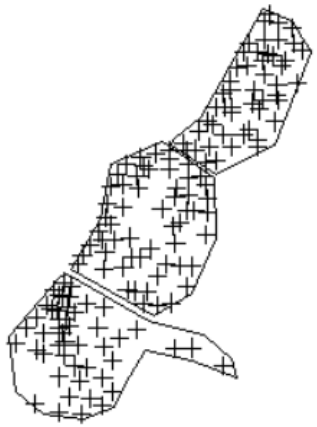
x: geometry (loctions) of the queries

y: layer from which the geometries or attributes are queried

Overlay and spatial query

over(meuse, srdf)

```
> meuse
        coordinates cadmium copper lead zinc   elev       dist   om ffreq soil lime landuse dist.m
1   (181072, 333611)    11.7     85  299 1022  7.909 0.00135803 13.6     1    1    1      Ah     50
2   (181025, 333558)     8.6     81  277 1141  6.983 0.01222430 14.0     1    1    1      Ah     30
3   (181165, 333537)     6.5     68  199  640  7.800 0.10302900 13.0     1    1    1      Ah    150
4   (181298, 333484)     2.6     81  116  257  7.655 0.19009400  8.0     1    2    0      Ga    270
5   (181307, 333330)     2.8     48  117  269  7.480 0.27709000  8.7     1    2    0      Ah    380
6   (181390, 333260)     3.0     61  137  281  7.791 0.36406700  7.8     1    2    0      Ga    470
7   (181165, 333370)     3.2     31  132  346  8.217 0.19009400  9.2     1    2    0      Ah    240
8   (181027, 333363)     2.8     29  150  406  8.490 0.09215160  9.5     1    1    0      Ab    120
9   (181060, 333231)     2.4     37  133  347  8.668 0.18461400 10.6     1    1    0      Ab    240
10  (181232, 333168)     1.6     24   80  183  9.049 0.30970200  6.3     1    2    0       W    420
> srdf
An object of class "SpatialPolygonsDataFrame"
Slot "data":
   X1 X2
r1  1  5
r2  2  4
r3  3  3
> over(meuse,srdf)
   X1 X2
1   1  5
2   1  5
3   1  5
4   1  5
5   1  5
```



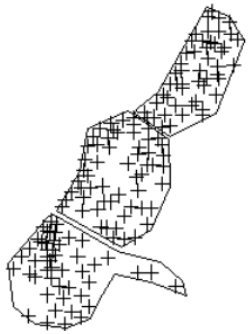
The output is, for each point, the data frame of the polygon it intersected.

Overlay and spatial query

over(srdf, meuse, fn = mean)

```
> meuse
        coordinates cadmium copper lead zinc   elev       dist   om ffreq soil lime landuse dist.m
1   (181072, 333611)    11.7     85  299 1022  7.909 0.00135803 13.6     1    1    1      Ah     50
2   (181025, 333558)     8.6     81  277 1141  6.983 0.01222430 14.0     1    1    1      Ah     30
3   (181165, 333537)     6.5     68  199  640  7.800 0.10302900 13.0     1    1    1      Ah    150
4   (181298, 333484)     2.6     81  116  257  7.655 0.19009400  8.0     1    2    0      Ga    270
5   (181307, 333330)     2.8     48  117  269  7.480 0.27709000  8.7     1    2    0      Ah    380
6   (181390, 333260)     3.0     61  137  281  7.791 0.36406700  7.8     1    2    0      Ga    470
7   (181165, 333370)     3.2     31  132  346  8.217 0.19009400  9.2     1    2    0      Ah    240
8   (181027, 333363)     2.8     29  150  406  8.490 0.09215160  9.5     1    1    0      Ab    120
9   (181060, 333231)     2.4     37  133  347  8.668 0.18461400 10.6     1    1    0      Ab    240
10  (181232, 333168)     1.6     24   80  183  9.049 0.30970200  6.3     1    2    0       W    420
```

```
> srdf
An object of class "SpatialPolygonsDataFrame"
Slot "data":
   X1 X2
r1  1  5
r2  2  4
r3  3  3
```



```
> over(srdf, meuse,fn = mean)
    cadmium   copper     lead     zinc     elev      dist       om ffreq soil lime landuse   dist.m
r1 4.036000 44.48000 147.2600 475.8800 8.225760 0.1791637       NA    NA   NA   NA      NA 237.4000
r2 2.910526 40.22807 145.4035 452.0702 8.485649 0.3207399 7.219298    NA   NA   NA      NA 361.2281
r3 2.820833 36.08333 169.1667 484.2500 7.722208 0.2075471 7.350000    NA   NA   NA      NA 261.2500
```

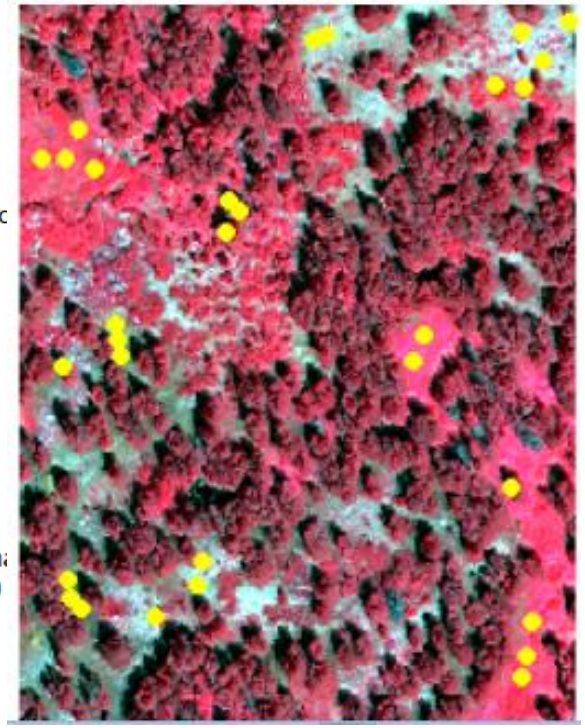
The output is, for each polygon, the fn (mean) value of the data frame of the points falling within it.

Vector and Raster Fusion

Extract raster information at vector-defined locations

Raster data: tahoe_highrez_brick.tif (3 layers)

```
> tahoe_highrez_brick
class       : RasterBrick
dimensions  : 400, 400, 160000, 3  (nrow, ncol, ncell, nlayers)
resolution  : 5.472863e-06, 5.472863e-06  (x, y)
extent      : -119.9328, -119.9306, 39.28922, 39.29141  (xmin, xmax, ymin, ymax)
coord. ref. : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
data source : Z:\Teaching\GISProgramming\geog489_lecture20_raster_vector_data\tahc
names       : tahoe_highrez.1, tahoe_highrez.2, tahoe_highrez.3
min values  :               0,              0,              0
max values  :             255,            255,            255
```



Vector data: tahoe_highrez_training_points.shp

```
> tahoe_highrez_training_points
class       : SpatialPointsDataFrame
features    : 30
extent      : -119.9327, -119.9306, 39.28936, 39.29136  (xmin, xmax, ymin, yma
coord. ref. : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
variables   : 2
names       : ID,        SPECIES
min values  :  1, Non-vegetation
max values  : 30,           Tree
```

Vector and Raster Fusion

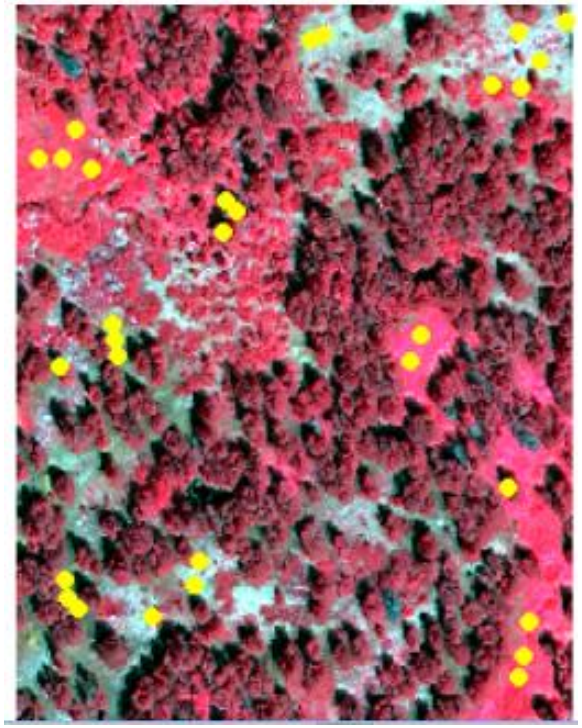
Extract raster information at vector-defined locations

To extract pixel data at specific points:

extract(x, y):

x: Raster object

y: Vector object



```
extracted_data <- extract(x=tahoe_highrez_brick,
    y=tahoe_highrez_training_points)
```

	tahoe_highrez.1	tahoe_highrez.2	tahoe_highrez.3
1	166	47	74
2	165	86	110
3	147	28	41
4	147	55	68
5	119	5	9

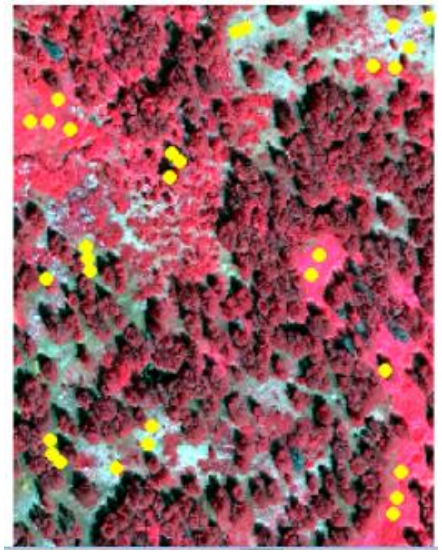
Vector and Raster Fusion

Extract raster information at vector-defined locations

To link this up with the coordinates of the spatial points

extracted_data_w_data <- cbind(extracted_data_w_cellnum, tahoe_highrez_training_points@coords))

ID	tahoe_highrez.1	tahoe_highrez.2	tahoe_highrez.3	coords.x1	coords.x2
1	166	47	74	-119.9319	39.29078
2	165	86	110	-119.9320	39.29082
3	147	28	41	-119.9320	39.29072
4	147	55	68	-119.9309	39.28994
5	119	5	9	-119.9323	39.28954



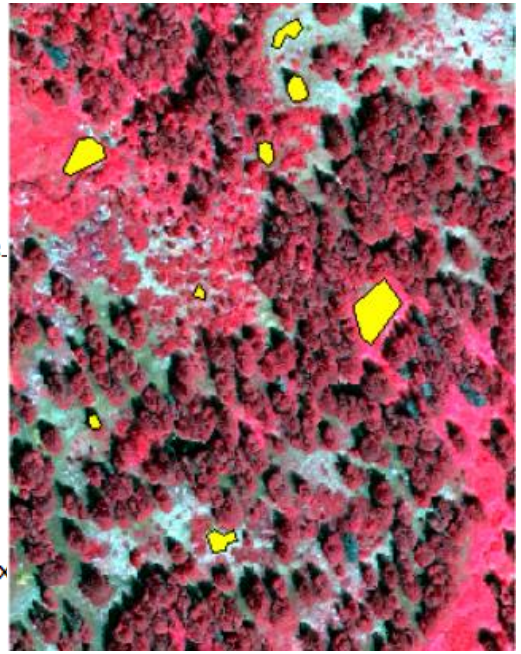
Vector and Raster Fusion

Extract raster information at vector-defined locations

A point will always return a single value. What about a polygon?

Raster data: tahoe_highrez_brick.tif (3 layers)

```
> tahoe_highrez_brick
class       : RasterBrick
dimensions  : 400, 400, 160000, 3  (nrow, ncol, ncell, nlayers)
resolution  : 5.472863e-06, 5.472863e-06  (x, y)
extent      : -119.9328, -119.9306, 39.28922, 39.29141  (xmin, xmax, ymin, ymax)
coord. ref. : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
data source : Z:\Teaching\GISProgramming\geog489_lecture20_raster_vector_data\tahoe_
names       : tahoe_highrez.1, tahoe_highrez.2, tahoe_highrez.3
min values  :               0,               0,               0
max values  :             255,             255,             255
```



Vector data: tahoe_highrez_training_polygons.shp

```
> tahoe_highrez_training_polygons
class       : SpatialPolygonsDataFrame
features    : 8
extent      : -119.9326, -119.9311, 39.28956, 39.29135  (xmin, xmax, ymin, ymax
coord. ref. : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
variables   : 2
names       : ID,         Class
min values  :  1, Non-vegetated
max values  :  8,          Tree
```

Vector and Raster Fusion

Extract raster information at vector-defined locations

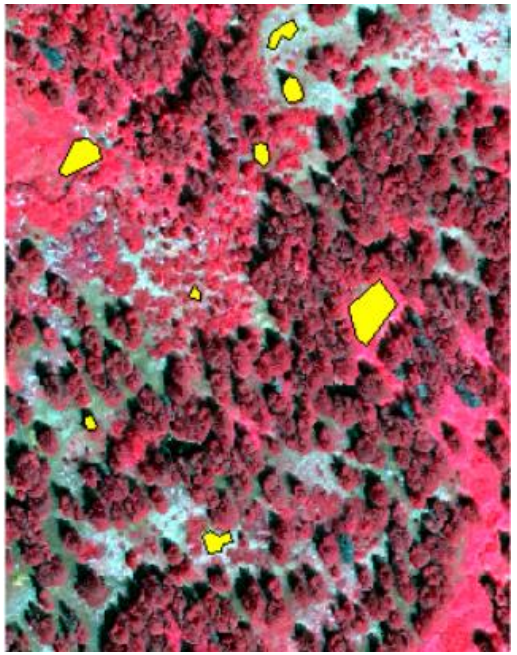
A point will always return a single value. What about a polygon?

```
extracted_from_poly <- extract(x=tahoe_highrez_brick,
                               y=tahoe_highrez_training_polygons)
```

Every pixel falling within a polygon is extracted

and stored in a list element.

```
> extracted_from_poly[[1]]
     tahoe_highrez.1 tahoe_highrez.2 tahoe_highrez.3
[1,]              31              11              16
[2,]              65              22              29
[3,]              65              22              29
[4,]              78              29              39
[5,]              86               7              20
```



Vector and Raster Fusion

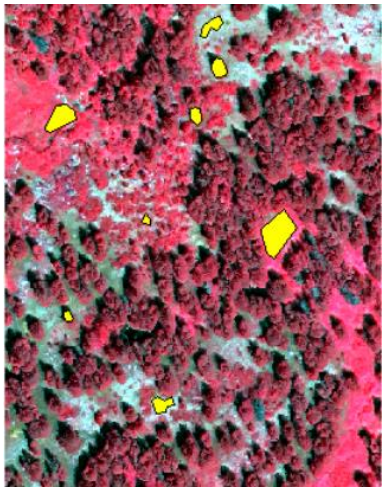
Extract raster information at vector-defined locations

What is the fraction of the pixel covered by a polygon?

extracted_from_poly <- extract(x=tahoe_highrez_brick, y=tahoe_highrez_training_polygons, weights = TRUE, normalizeWeights = FALSE)

Values of 1.0 mean the entire pixel was covered by polygon. Less than 1.0 mean the pixels were at the edge of the polygon, so they weren't entirely covered.

```
> extracted_from_poly[[1]]
     tahoe_highrez.1 tahoe_highrez.2 tahoe_highrez.3 weight
[1,]              16               2               6   0.25
[2,]              31              11              16   0.75
[3,]              75              46              47   0.45
[4,]              41               0               1   0.45
[5,]              65              22              29   1.00
```



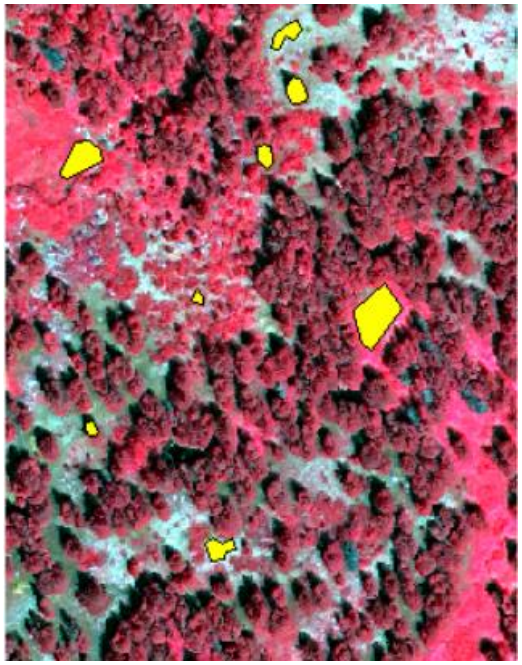
Vector and Raster Fusion

Extract raster information at vector-defined locations

A lot of times, we want the mean value per polygon, not the entire polygon.

```
extracted_from_poly <- extract(x=tahoe_highrez_brick,
                               y=tahoe_highrez_training_polygons,
                               fun=mean)
```

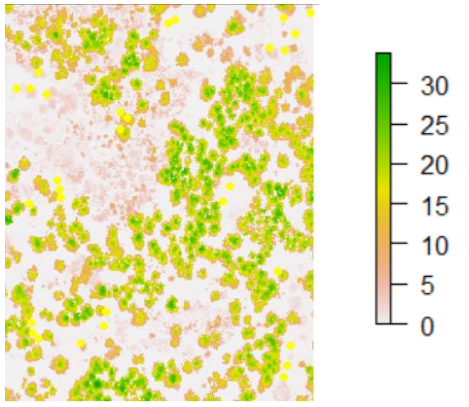
```
> extracted_from_poly
     tahoe_highrez.1 tahoe_highrez.2 tahoe_highrez.3
[1,]       147.26455        55.25397        71.51852
[2,]       139.96503        56.95105        66.35664
[3,]        88.05634        26.50704        33.94366
[4,]       254.99089        89.41686       158.08087
[5,]       212.00000        63.55932        91.06780
[6,]       252.62260        72.94030       111.49254
[7,]       179.44118       237.28922       233.55392
[8,]       194.05856       251.87387       252.10360
```



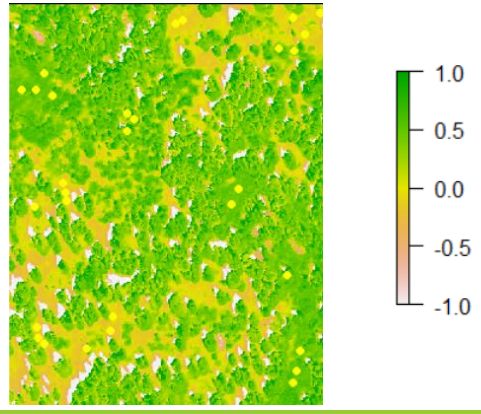
Raster Models

Let's perform a simple linear regression, but using extracted data. We are interested in seeing if Lidar derived vegetation height is linearly related to NDVI:

```
# Calculate Lidar height:
highest_hit_raster <- raster("tahoe_lidar_highesthit.tif")
bareearth_raster <- raster("tahoe_lidar_bareearth.tif")
lidar_height <- highest_hit_raster - bareearth_raster
```




```
# Calculate NDVI:
ndvi <- function(x) { (x[1] - x[2])/(x[1] + x[2])}
tahoe_ndvi <- calc(tahoe_highrez_brick,ndvi)
```

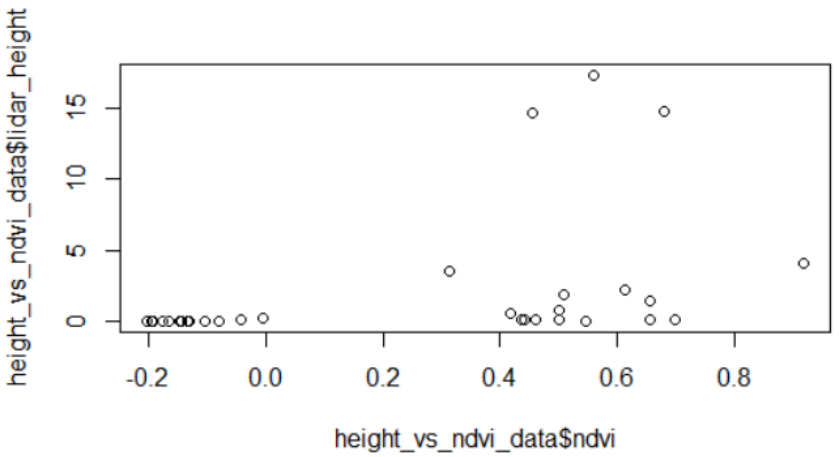


Raster Models

Let's perform a simple linear regression, but using extracted data. We are interested in seeing if Lidar derived vegetation height is linearly related to NDVI:

```
# 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)
```

```
> 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
```



Raster Models

Let's perform a simple linear regression, but using extracted data. We are interested in seeing if Lidar derived vegetation height is linearly related to NDVI:

Linear regression: response variable ~ explanatory variable (lidar_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
```

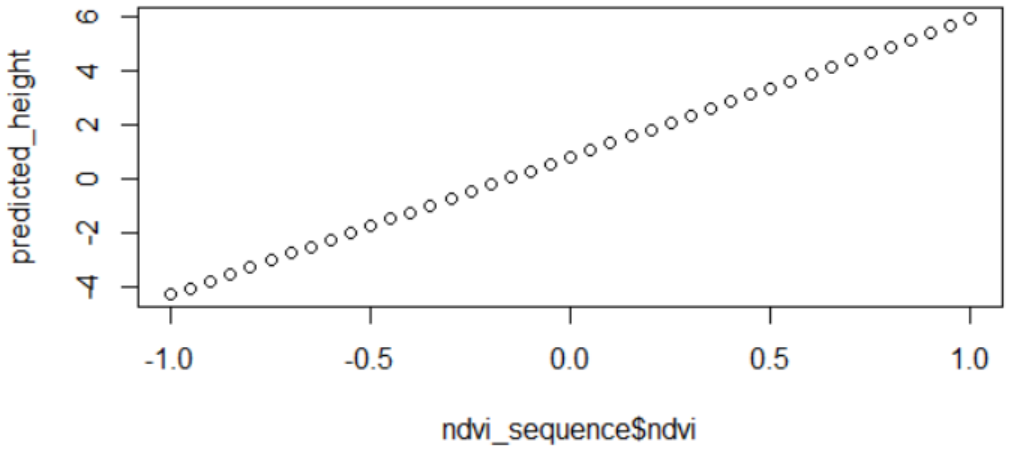
Raster Models

Use the linear regression model for PREDICTION

Let's create a sequence of NDVI values and predict the veg height from the linear regression model:

ndvi_sequence = data.frame(ndvi=seq(from=-1, to=1, by=0.05))

predicted_height <- predict(ndvi_height_lm,ndvi_sequence)



Raster Models

Use the linear regression model for PREDICTION

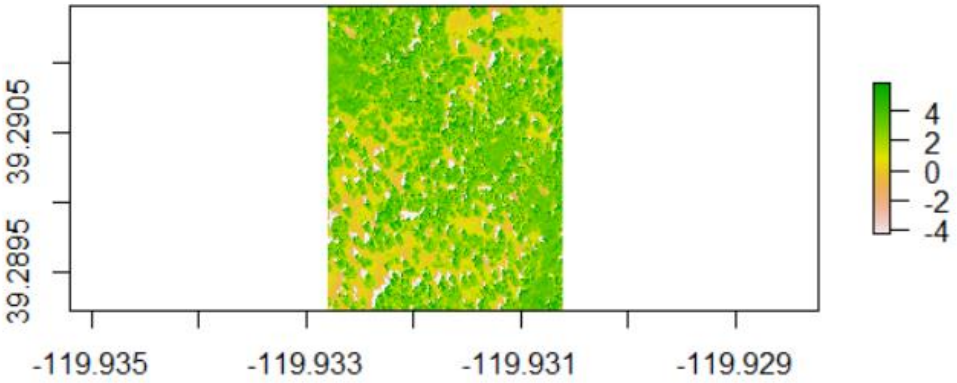
Let's apply this model to a raster image!

First off, we need to set the raster layer names to be the same as the variables:

names(tahoe_ndvi) <- “ndvi”

Next, we use predict in a similar way:

tahoe_height_pred <- predict(tahoe_ndvi,ndvi_height_lm)



Raster Models

#What you just learned is INCREDIBLY hard to do in ArcGIS, ENVI, Imagine, etc:

1) Extract data from a raster at specific points.

2) Create a model from the extracted data.

3) Apply the model to a 'test' dataset.

4) Apply the model to a raster.

This was TRIVIAL to do in R, and this is one of the chief reasons we use R with GIS!