Lecture 18

Visualizing Spatial Data and Spatial Import and Export

GEOG 489

SPRING 2020

Package Introduction

Please see the package introduction guideline PPT for more info

Description

Each student will be expected to introduce a package (or two) that is relevant to their research interests through discussion forum on Compass.

The objectives are:

- Learn how to find/download/install a new package and learn how to use it
- Teach your peers about existing R packages that may be useful in their research

Package Introduction

The package introduction PPT must include:

1. Brief introduction: what does the package do and why is it useful? (1-2 slides)

2. Author introduction: a short background (affiliation and other packages, etc.) on at least one of the package authors (1 slide)

3. Simple demonstration of package code: example input/output from the examples or custom coded examples (4-5 slides, 2.5 minutes)

Please submit your PPT on Compass by March 31, 2020

Visualize Spatial Data

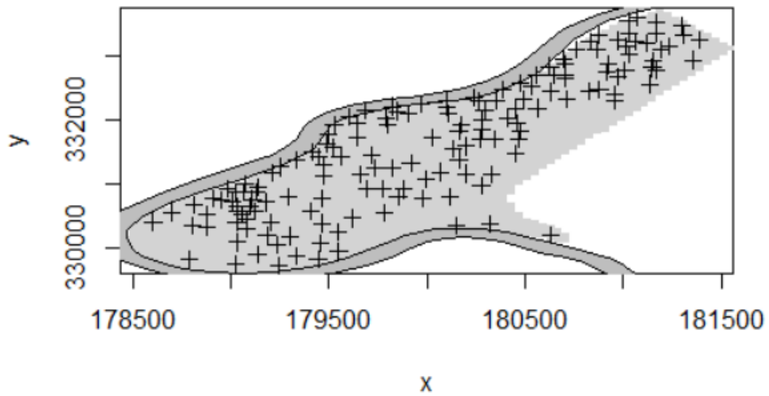
Plot raster/vector

Use the add=TRUE to add multiple layers to the plot:

image(meuse.raster,col="lightgrey")

plot(meuse.SpatialPolygons,col="grey",add=TRUE)

plot(meuse,add=TRUE))



Visualize Spatial Data

Plot attributes

make a grid of the soil types from meuse.grid:

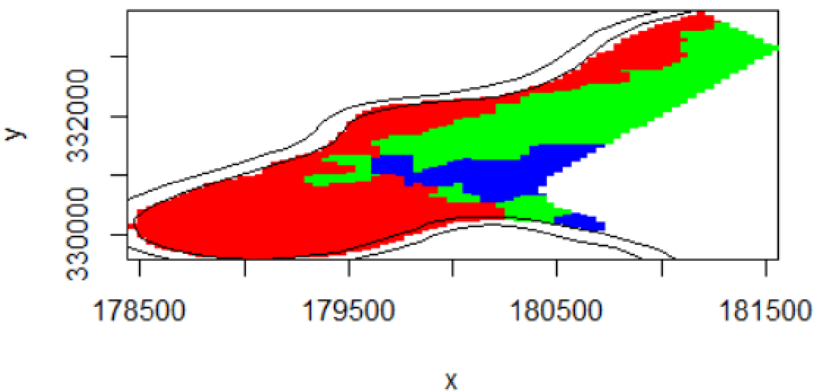
xyz <- data.frame(meuse.grid$x,meuse.grid$y,meuse.grid$soil)

meuse.raster <- rasterFromXYZ(xyz)

image(meuse.raster,col=rainbow(3),breaks=c(0,1,2,3))

plot(meuse.SpatialPolygons,add=TRUE)

```
> head(meuse.grid)
       x      y part.a part.b      dist soil ffreq
1 181180 333740      1      0 0.0000000    1     1
2 181140 333700      1      0 0.0000000    1     1
3 181180 333700      1      0 0.0122243    1     1
4 181220 333700      1      0 0.0434678    1     1
5 181100 333660      1      0 0.0000000    1     1
6 181140 333660      1      0 0.0122243    1     1
```

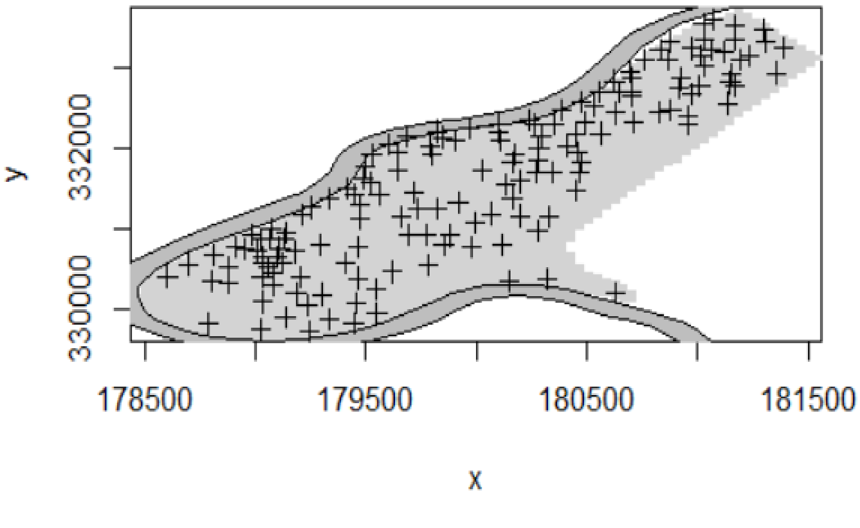


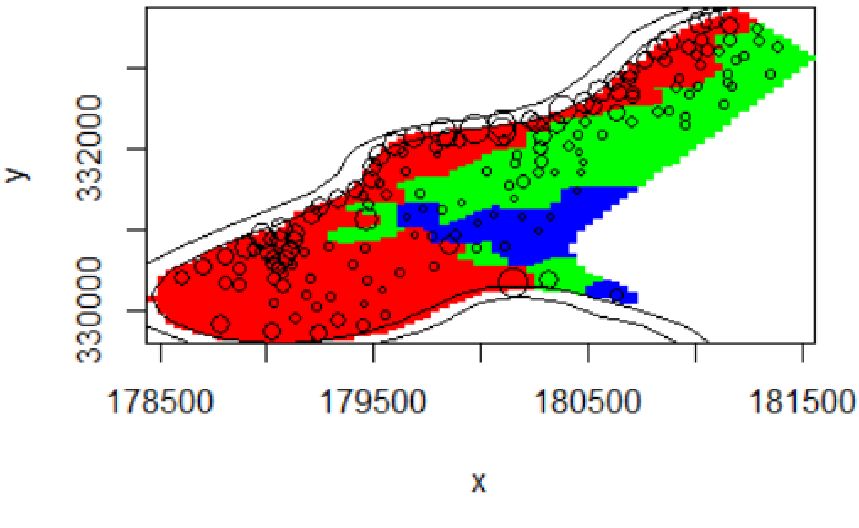
Visualize Spatial Data

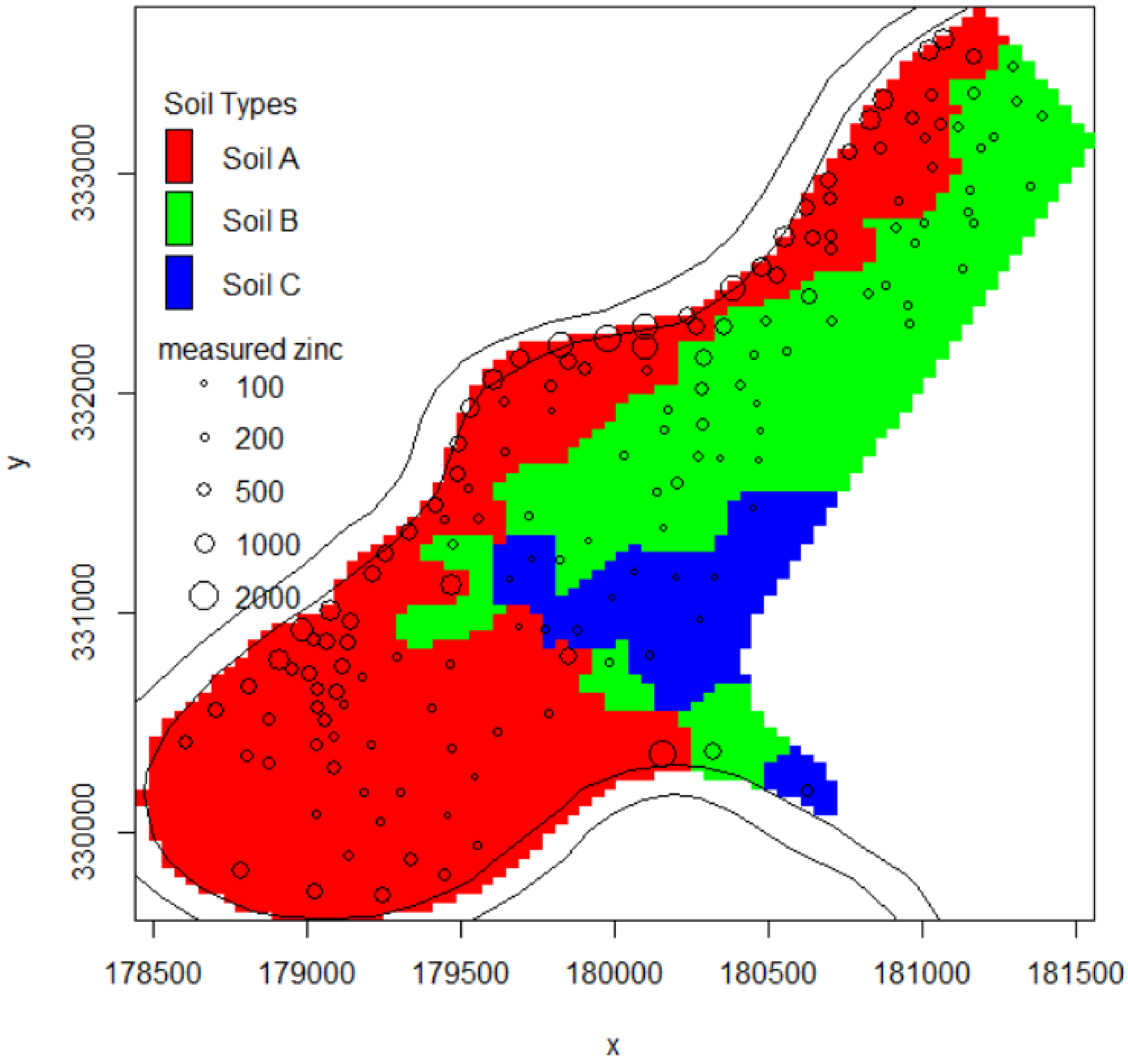
Plot raster/vector

Scale the points with the value of the zinc column:

plot(meuse, pch=1,cex=sqrt(meuse$zinc)/20,add=TRUE)







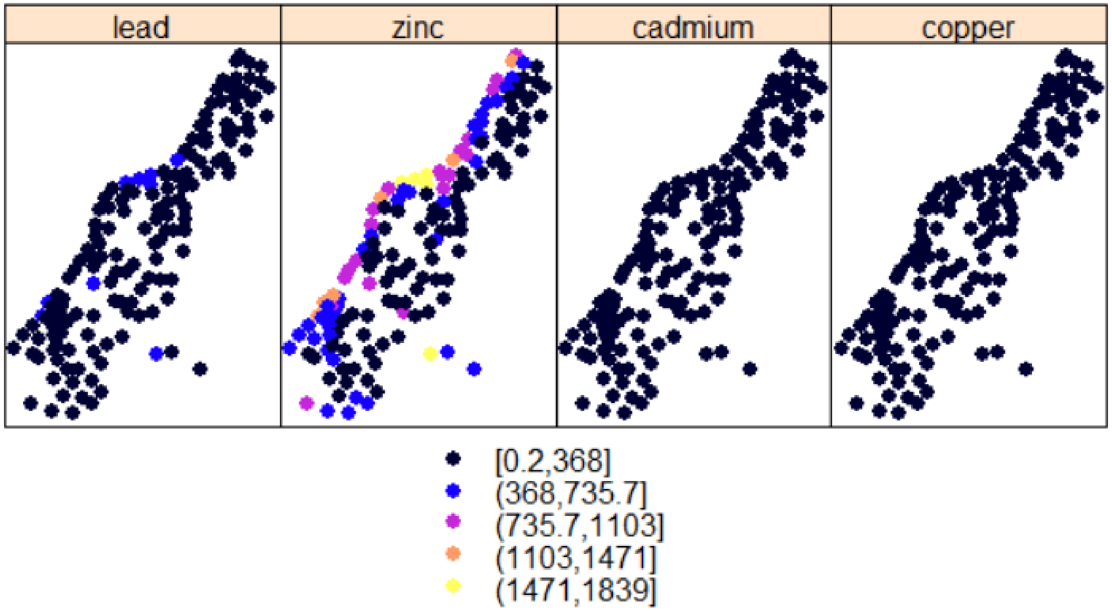
Soil Types
Soil A
Soil B
Soil C
measured zinc
100
200
500
1000
2000
y
333000
332000
331000
330000
x
178500
179000
179500
180000
180500
181000
181500

Visualize Spatial Data

Trellis/Lattice Plots with spplot

spplot uses lattice graphics rather than plot graphics. lattice is a more complicated plotting system, but can be extended more than plot can.

```
spplot(obj=meuse,zcol=c("lead","zinc","cadmium","copper"))
```



Visualize Spatial Data

Interact with plots

There are essentially two functions:

?locator # Returns the x,y positions of the clicked location

?identify # Plots and returns the labels of the items

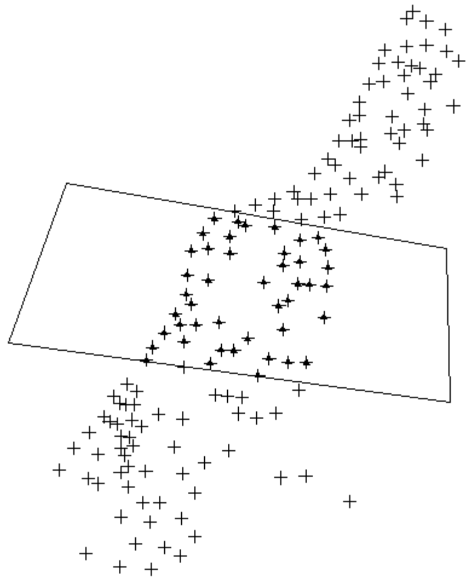
We can select points using a digitized polygon:

```
region <- locator(type="o")

plot(meuse[!is.na(over(meuse,sps)),],pch=24,

              cex=0.5,add=TRUE)

plot(sps,add=TRUE)
```



Visualize Spatial Data

Interact with plots

```
# To select a polygon:

nc <- readShapePoly(nc_shp,proj4string=prj)

pt <- locator(type = "p")

poly_selected<- !is.na(over(nc,SpatialPoints(cbind(pt$x,pt$y),proj4string=prj)))

plot(nc[poly_selected,], col = "blue", add = TRUE)
```

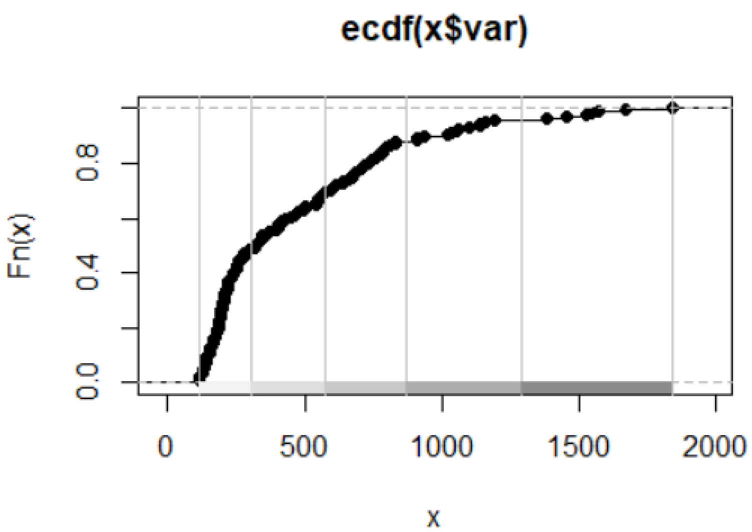
Visualize Spatial Data

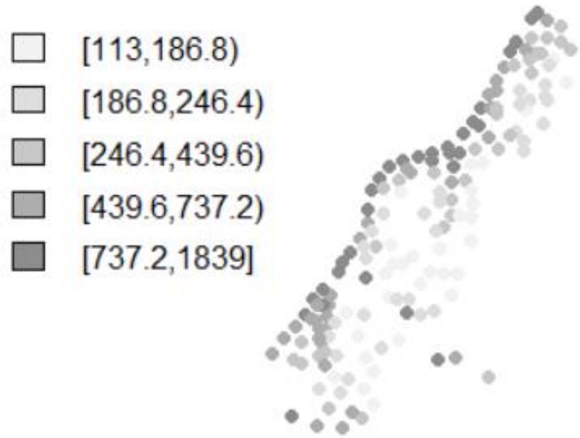
Color Palettes and Class Intervals

If we want to create intervals for continuous data for mapping/color choices:

fj5 <- classIntervals(meuse$zinc,n=5,style="fisher")

```
style: fisher
  one of 14,891,626 possible partitions of this variable into 5 classes
   [113,307.5)    [307.5,573)    [573,869.5) [869.5,1286.5)  [1286.5,1839]
            75             32             29             12              7
```





Spatial Data Import and Export

rgdal is a wrapper for the GDAL and PROJ.4 libraries.

PROJ.4: provides a library that standardizes defining projections and datums, and provides tools for converting between projections/datums (reprojection). (http://trac.osgeo.org/proj/)

GDAL/OGR (the Geospatial Data Abstraction Library) is an open source library that provides abstraction for basic raster/vector processing (http://gdal.org/). This includes:

- reading/writing raster and vector files
- querying and defining CRS
- getting raster/vector metadata
- re-projecting raster/vector data
- accessing subsets of the raster/vector data

Spatial Data Import and Export

Define and convert projection:

PROJ.4 CRS Specification: PROJ.4 uses a 'tag=value' representation of the parameters.

EPSG (European Petroleum Survey Group) is a now-defunct set of geodetic parameters that are distributed with PROJ.4. EPSG parameters are stored as a code that is linked to its set of parameters.

EPSG <- make_EPSG()

```
code       note                                                          prj4
3821    # TWD67                             +proj=longlat +ellps=aust_SA +no_defs
3824    # TWD97          +proj=longlat +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +no_defs
3889     # IGRS          +proj=longlat +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +no_defs
3906 # MGI 1901 +proj=longlat +ellps=bessel +towgs84=682,-203,480,0,0,0,0 +no_defs
```

Spatial Data Import and Export

Define and convert projection:

Define the European Datum 1950 (ED50), and then convert this to the WGS84 datum so we can use the coordinates with GPS.

EPSG[grep("^# ED50$",EPSG$note),]   # search EPSG code for Datum

```
    code   note                                                         prj4
159 4230 # ED50 +proj=longlat +ellps=intl +towgs84=-87,-98,-121,0,0,0,0 +no_defs
```

ED50 <- CRS("+init=epsg:4230")   # define the Datum using EPSG code

```
code       note                                                              prj4
3821     # TWD67                              +proj=longlat +ellps=aust_SA +no_defs
3824     # TWD97          +proj=longlat +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +no_defs
3889      # IGRS          +proj=longlat +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +no_defs
3906 # MGI 1901 +proj=longlat +ellps=bessel +towgs84=682,-203,480,0,0,0,0 +no_defs
```

IJ.ED50 <- SpatialPoints(cbind(x=IJ.east,y=IJ.north),ED50)

res <- spTransform(IJ.ED50,CRS("+proj=longlat +datum=WGS84"))   # convert

Spatial Data Import and Export

Vector File Format

We will use the OGR libraries that ship with GDAL, interfaced with the rgdal library. When a vector file is brought into R, OGR will (usually) figure out what format the file is in, load the correct driver, and import the data.

readOGR(): to import a vector into a Spatial* or Spatial*DataFrame,

The two basic input parameters are:

dsn: the dataset name

layer: the layer name to extract

For an ESRI Shapefile, which is actually a set of files stored in the same directory with extensions: a .shp, .dbf, and .shx . dsn will be the directory these files are in, and layer will be the shapefile name WITHOUT the extensions:

scot_LL <- readOGR(dsn=".",layer="scot")

Spatial Data Import and Export

Vector File Format

scot_LL <- readOGR(dsn=".",layer="scot")

```
> summary(scot_LL)
Object of class SpatialPolygonsDataFrame
Coordinates:
        min        max
x -8.621389 -0.7530556
y 54.627220 60.8444443
Is projected: NA
proj4string : [NA]
Data attributes:
           NAME          ID
```

We have another data file which contains data we want to link based on the IDs:

scot_dat <- read.table("scotland.dat",header=TRUE)

```
head(scot_dat)
District Observed Expected PcAFF Latitude Longitude
       1        9      1.4    16    57.29      5.50
       2       39      8.7    16    57.56      2.36
       3       11      3.0    10    58.44      3.90
       4        9      2.5    24    55.76      2.40
       5       15      4.3    10    57.71      5.09
       6        8      2.4    24    59.13      3.25
```

Spatial Data Import and Export

Vector File Format

Merge the column “ID” in scot_LL (SpatialPolygonsDataFrame) and column “District” in scot_dat (data frame)

scot_dat1 <- scot_dat[match(scot_LL$ID,scot_dat$District),]

scot_LLa <- spCbind(scot_LL,scot_dat1) # cbinding to a Spatial*DataFrames

```
> scot_LLa@data
          NAME ID District Observed Expected PcAFF Latitude Longitude
0   Sutherland 12       12        5      1.8    16    58.06      4.64
1        Nairn 13       13        3      1.1    10    57.47      3.98
2    Inverness 19       19        9      5.5     7    57.24      4.73
3  Banff-Buchan  2        2       39      8.7    16    57.56      2.36
4     Bedenoch 17       17        2      1.1    10    57.06      4.09
5   Kincardine 16       16        9      4.6    16    57.00      3.00
6        Angus 21       21       16     10.5     7    56.75      2.98
7       Dundee 50       50        6     19.6     1    56.45      3.20
8      NE Fife 15       15       17      7.8     7    56.30      3.10
```

Spatial Data Import and Export

Vector File Format

writeOGR(): to export a vector to shapefile (or other formats)

The two basic input parameters are:

obj: a Spatial* object

dsn: the dataset output name to use, can be a folder or filename.

layer: the layer name to write it to.

```
# We'll write to a KML file for use in Google Earth:
writeOGR(scot_LLa["ID"],dsn="scot_district.kml",layer="borders",
driver="KML")
```

```
# We can also write to a Shapefile:
drv <- "ESRI Shapefile"
writeOGR(scot_BNG,dsn=".",layer="scot_BNG",driver=drv)
```

Spatial Data Import and Export

Raster File Format (rgdal and raster)

Recall the differences between the three types of Raster* objects:

RasterLayer: a single layer (band) raster

RasterBrick: a multiband raster originating from a SINGLE file.

RasterStack: a multiband raster comprised of MULTIPLE files.

Spatial Data Import and Export

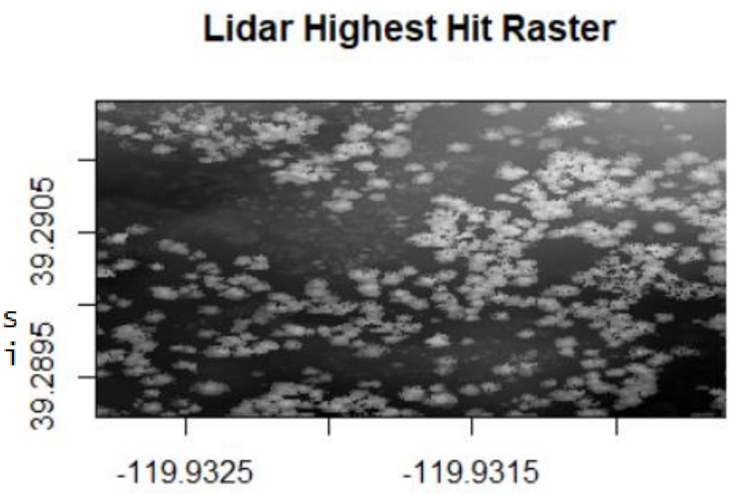
Raster File Format (rgdal and raster)

If we just want to import a single band file, we use raster() with the filename:

highest_hit_raster <- raster("tahoe_lidar_highesthit.tif")

highest_hit_raster

```
class       : RasterLayer
dimensions  : 400, 400, 160000  (nrow, ncol, ncell)
resolution  : 5.472863e-06, 5.472863e-06  (x, y)
extent      : -119.9328, -119.9306, 39.28922, 39.29141
coord. ref. : +proj=longlat +datum=WGS84 +no_defs +ellps
data source : Z:\Teaching\GISProgramming\Week10\tahoe_li
names       : tahoe_lidar_highesthit
```



Spatial Data Import and Export

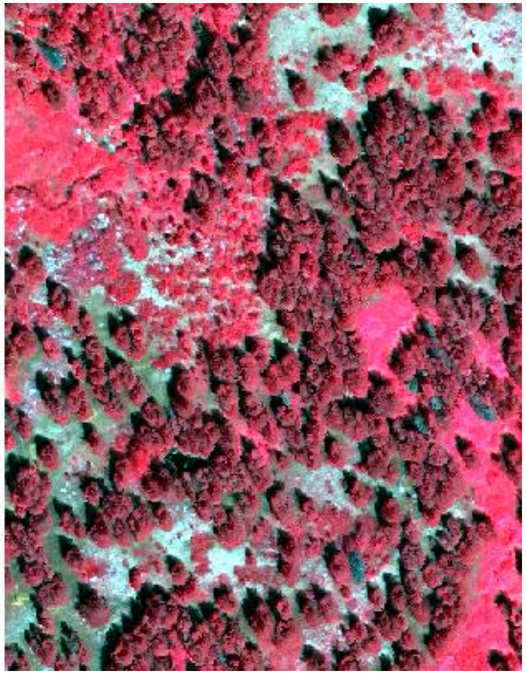
Raster File Format (rgdal and raster)

If we just want to import a multi-band file, we use brick() with the filename:

tahoe_highrez_brick <- brick("tahoe_highrez.tif")

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, ymi
coord. ref. : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=
data source : Z:\Teaching\GISProgramming\Week10\tahoe_highrez.tif
names       : tahoe_highrez.1, tahoe_highrez.2, tahoe_highrez.3
min values  :               0,               0,               0
max values  :             255,             255,             255
```



Spatial Data Import and Export

Raster File Format (rgdal and raster)

If we just want to fuse multiple files into a single raster *, we use stack()

band1_filename <- "tahoe_lidar_bareearth.tif"

band2_filename <- "tahoe_lidar_highesthit.tif"

tahoe_lidar <- stack(band1_filename,band2_filename)

```
> tahoe_lidar
class       : RasterStack
dimensions  : 400, 400, 160000, 2  (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
names       : tahoe_lidar_bareearth, tahoe_lidar_highesthit
```

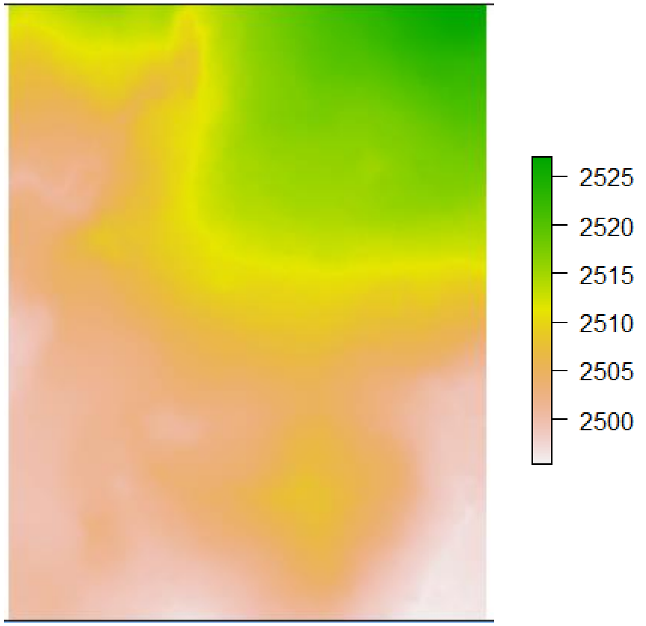
Spatial Data Import and Export

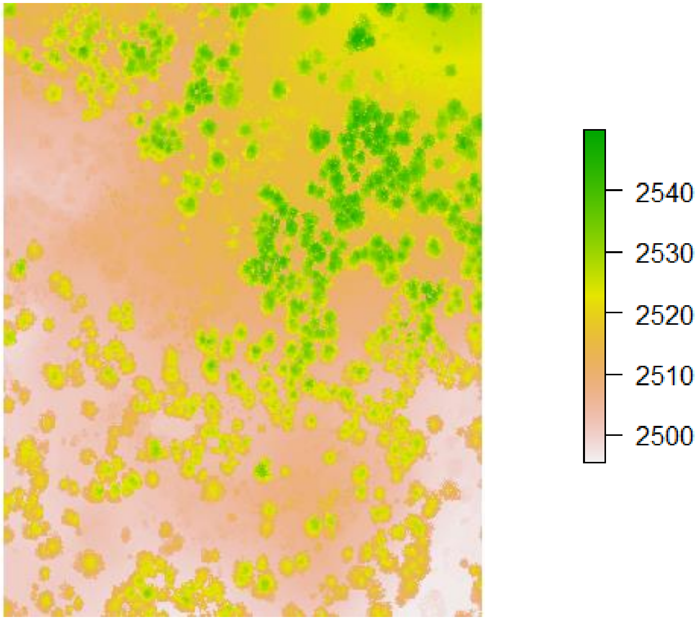
Raster File Format (rgdal and raster)

If we just want to fuse multiple files into a single raster *, we use stack()

plot(tahoe_lidar,y=1) # Plots layer = 1

plot(tahoe_lidar,y=2) # Plots layer = 2





Spatial Data Import and Export

Raster File Format (rgdal and raster)

To write a raster file (writeRaster)

 # To save to a GeoTIFF:

Mysavedraster <- writeRaster(x=tahoe_lidar,filename="tahoe_lidar", format="GTiff",overwrite=TRUE)

 # To save to a ENVI file:

writeRaster(x=tahoe_lidar,filename="tahoe_lidar_bsq",format="ENVI",

NAflag=-9999,bandorder="BSQ",overwrite=TRUE)