Week 2: Spatial Data

1. Overview of Worked Example

Author: Helene Wagner

This code builds on data and code from the 'GeNetIt' package by Jeff Evans and Melanie Murphy.

a) Goals

This worked example shows:

- How to import spatial coordinates and site attributes as spatially referenced data.

- How to plot raster data in R and overlay sampling locations.
- How to calculate patch-level and class-level (cover type) landscape metrics.
- How to extract landscape data at sampling locations and within a buffer around them.

Try modifying the code to import your own data!

b) Data set

This code uses landscape data and spatial coordinates from 30 locations where Colombia spotted frogs (*Rana luteiventris*) were sampled for the full data set analyzed by Funk et al. (2005) and Murphy et al. (2010). Please see the separate introduction to the data set.

- RALU_sites_all.csv: File with spatial coordinates and site attributes (preformatted for import, 31 rows x 19 columns).

We will extract values at sampling point locations and within a local neighborhood (buffer) from six raster layers, which are included with the 'GeNetIt' package (see Murphy et al. 2010 for definitions):

- cti: compound topographic index
- err27: elevation relief ratio
- ffp: frost-free period
- gsp: growing season precipitation
- hli: heat load index
- nlcd: national land cover data (categorical map)

c) Required R libraries

```r
require(sp)
require(raster)
require(GeNetIt)
require(tmaptools)
require(SDMTools) # for landscape metrics
```

d) List of tasks

- Import site data from .CSV file into a 'SpatialPointsDataFrame' object (package 'sp').
- Display raster maps (package 'raster') and overlay sampling locations. Extract raster values at sampling locations.

- Calculate patch-level and class-level landscape metrics (package 'SDMTools').
- Extract landscape metrics at sampling locations.

2. Import site data from .csv file

a) Import data into 'SpatialPointsDataFrame'

The .csv file with the data is part of the course R package. We can use the function 'system.file' to access it.

```
RALU.site <- read.csv(system.file("extdata", "RALU_site_all.csv",
                              package = "TestCoursePackage"), header=TRUE)
head(RALU.site)
```

```
##   coords.x1 coords.x2       SiteName        Drainage      Basin Substrate
## 1  688816.6   5003207   AirplaneLake ShipIslandCreek Sheepeater      Silt
## 2  688494.4   4999093 BachelorMeadow     WilsonCreek    Skyhigh      Silt
## 3  687938.4   5000223 BarkingFoxLake  WaterfallCreek    Terrace      Silt
## 4  689732.8   5002522   BirdbillLake      ClearCreek   Birdbill      Sand
## 5  690104.0   4999355        BobLake     WilsonCreek     Harbor      Silt
## 6  688742.5   4997481      CacheLake     WilsonCreek    Skyhigh      Silt
##                             NWI AREA_m2 PERI_m Depth_m  TDS FISH ACB
## 1                    Lacustrine 62582.2 1142.8   21.64  2.5    1   0
## 2 Riverine_Intermittent_Streambed   225.0   60.0    0.40  0.0    0   0
## 3                    Lacustrine 12000.0  435.0    5.00 13.8    1   0
## 4                    Lacustrine 12358.6  572.3    3.93  6.4    1   0
## 5                    Palustrine  4600.0  321.4    2.00 14.3    0   0
## 6                    Palustrine  2268.8  192.0    1.86 10.9    0   0
##     AUC AUCV  AUCC   AUF AWOOD  AUFV
## 1 0.411    0 0.411 0.063 0.063 0.464
## 2 0.000    0 0.000 1.000 0.000 0.000
## 3 0.300    0 0.300 0.700 0.000 0.000
## 4 0.283    0 0.283 0.717 0.000 0.000
## 5 0.000    0 0.000 0.500 0.000 0.500
## 6 0.000    0 0.000 0.556 0.093 0.352
```

The dataset has two columns with spatial coordinates and several attribute variables.

So far, R treats the spatial coordinates like any other quantitative variables. To let R know this is spatial information, we import it into a spatial object type, a 'SpatialPointsDataFrame' from the 'sp' package.

The conversion is done with the function 'coordinates', which takes a data frame and converts it to a spatial object of the same name. The code is not very intuitive.

Note: the tilde symbol '~' (here before the first coordinate) is often used in R formulas, we will see it again later. It roughly translates to 'is modeled as a function of'.

```
RALU.site.sp <- RALU.site
coordinates(RALU.site.sp) <- ~coords.x1+coords.x2
head(RALU.site.sp)
```

```
##         SiteName        Drainage      Basin Substrate
## 1   AirplaneLake ShipIslandCreek Sheepeater      Silt
## 2 BachelorMeadow     WilsonCreek    Skyhigh      Silt
## 3 BarkingFoxLake  WaterfallCreek    Terrace      Silt
## 4   BirdbillLake      ClearCreek   Birdbill      Sand
## 5        BobLake     WilsonCreek     Harbor      Silt
## 6      CacheLake     WilsonCreek    Skyhigh      Silt
```

```
##                                NWI AREA_m2 PERI_m Depth_m  TDS FISH ACB
## 1                        Lacustrine 62582.2 1142.8   21.64  2.5    1   0
## 2 Riverine_Intermittent_Streambed   225.0   60.0    0.40  0.0    0   0
## 3                        Lacustrine 12000.0  435.0    5.00 13.8    1   0
## 4                        Lacustrine 12358.6  572.3    3.93  6.4    1   0
## 5                        Palustrine  4600.0  321.4    2.00 14.3    0   0
## 6                        Palustrine  2268.8  192.0    1.86 10.9    0   0
##     AUC AUCV  AUCC   AUF AWOOD  AUFV
## 1 0.411    0 0.411 0.063 0.063 0.464
## 2 0.000    0 0.000 1.000 0.000 0.000
## 3 0.300    0 0.300 0.700 0.000 0.000
## 4 0.283    0 0.283 0.717 0.000 0.000
## 5 0.000    0 0.000 0.500 0.000 0.500
## 6 0.000    0 0.000 0.556 0.093 0.352
```

Now R knows these are spatial data and knows how to handle them. It does not treat the coordinates as variables anymore, hence the first column is now 'SiteName'.

b) Add spatial reference data

Before we can combine the sampling locations with other spatial datasets, such as raster data, we need to tell R where on earth these locations are (georeferencing). This is done by specifying the 'Coordinate Reference System' (CRS) or a 'proj4' string.

For more information on CRS, see: https://www.nceas.ucsb.edu/~frazier/RSpatialGuides/OverviewCoordinateReferenceSystems.pdf

We know that these coordinates are UTM zone 11 (Northern hemisphere) coordinates, hence we can use a helper function to find the correct 'proj4' string, using function 'get_proj4' from the 'tmaptools' package. (For the Southern hemisphere, you would add 's' after the zone: "utm11s"). Here we call the function and the package simultaneously (this is good practice, as it helps keep track of where the functions in your code come from).

```r
proj4string(RALU.site.sp) <- tmaptools::get_proj4("utm11")
```

If we had longitude and latitude coordinates, we would modify the command like this: proj4string(RALU.site.sp) <- tmaptools::get_proj4("longlat")

c) Access data in 'SpatialPointsDataFrame'

As an S4 object, RALU.site.sp has predefined slots. These can be accessed with the @ symbol:

- @data: the attribute data
- @coords: the spatial coordinates
- @coords.nrs: the column numbers of the input data from which the coordinates were taken (filled automatically)
- @bbox: bounding box, i.e., the minimum and maximum of x and y coordinates (filled automatically)
- @proj4string: the georeferencing information

```r
slotNames(RALU.site.sp)
```

```
## [1] "data"        "coords.nrs"  "coords"      "bbox"        "proj4string"
```

Here are the first few lines of the coordinates:

```r
head(RALU.site.sp@coords)
```

```
##   coords.x1 coords.x2
## 1  688816.6   5003207
## 2  688494.4   4999093
## 3  687938.4   5000223
## 4  689732.8   5002522
## 5  690104.0   4999355
## 6  688742.5   4997481
```

And the proj4 string:

```
RALU.site.sp@proj4string
```

```
## CRS arguments:
##  +proj=utm +zone=11 +ellps=WGS84 +datum=WGS84 +units=m +no_defs
## +towgs84=0,0,0
```

3. Display raster data and overlay sampling locations, extract data

a) Display raster data

The raster data for this project are already available in the package 'GeNetIt', under the name 'rasters', and we can load them with 'data(rasters)'. They are stored as a 'SpatialPixelsDataFrame', another S4 object type from the 'sp' package.

```
data(rasters)
class(rasters)
```

```
## [1] "SpatialPixelsDataFrame"
## attr(,"package")
## [1] "sp"
```

However, raster data are better analyzed with the package 'raster', which has an object type 'raster'. - Maybe it was a bit confusing now to name our data 'rasters'. So let's rename it first to 'RALU.rasters.sp', then convert to a 'stack' of 'raster' object type (i.e. a set of raster layers with the same geometry).

```
RALU.rasters.sp <- rasters
RALU.rasters.r <- stack(RALU.rasters.sp)
class(RALU.rasters.r)
```

```
## [1] "RasterStack"
## attr(,"package")
## [1] "raster"
```

Printing the name of the raster stack displays a summary. A few explanations:

- **dimensions**: number of rows (nrow), number of columns (ncol), number of cells (ncell), number of layers (nlayers). So we see there are 6 layers in the raster stack.
- **resolution**: cell size is 30 m both in x and y directions (typical for Landsat-derived remote sensing data)
- **coord.ref**: projected in UTM zone 11, though the 'datum' (NAD83) is different than what we used for the sampling locations.

```
RALU.rasters.r
```

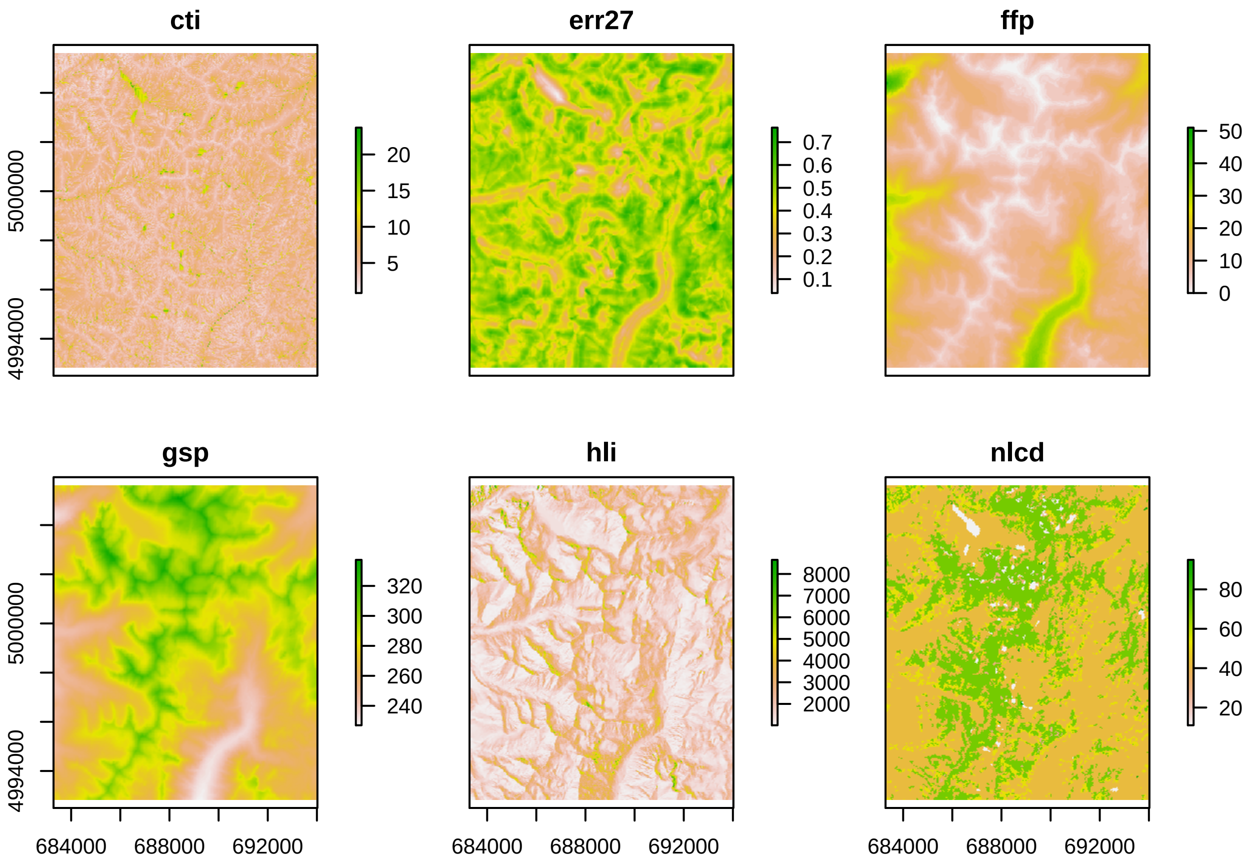
```
## class       : RasterStack
## dimensions  : 426, 358, 152508, 6  (nrow, ncol, ncell, nlayers)
## resolution  : 30, 30  (x, y)
## extent      : 683282.5, 694022.5, 4992833, 5005613  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=utm +zone=11 +datum=NAD83 +units=m +no_defs +ellps=GRS80 +towgs84=0,0,0
```

```
## names       :            cti,         err27,           ffp,           gsp,            hli,          nlcd
## min values  : 8.429851e-01, 3.906551e-02, 0.000000e+00, 2.270000e+02, 1.014000e+03, 1.100000e+01
## max values  :   23.7147598,    0.7637643,   51.0000000,  338.0696716, 9263.0000000,   95.0000000
```

Now we can use 'plot', which knows what to do with a raster stack.

Note: layer 'nlcd' is a categorical map of land cover types. See this week's bonus materials for how to better display a categorical map in R.

```
plot(RALU.rasters.r)
```



Some layers seem to show a similar pattern. It is easy to calculate the correlation between quantitative raster layers. Here, the last layer 'ncld', is in fact categorical (land cover type), and it's correlation here is meaningless.

```
layerStats(RALU.rasters.r, 'pearson', na.rm=T)
```

```
## $`pearson correlation coefficient`
##              cti       err27         ffp         gsp         hli
## cti    1.0000000 -0.25442672  0.12264734 -0.14029572 -0.30501483
## err27 -0.2544267  1.00000000 -0.23467075  0.21403415  0.07724426
## ffp    0.1226473 -0.23467075  1.00000000 -0.95144256 -0.07567975
## gsp   -0.1402957  0.21403415 -0.95144256  1.00000000  0.09520075
## hli   -0.3050148  0.07724426 -0.07567975  0.09520075  1.00000000
## nlcd  -0.1807878  0.12562961 -0.32975610  0.37653635  0.24655404
##             nlcd
## cti   -0.1807878
## err27  0.1256296
## ffp   -0.3297561
## gsp    0.3765363
```

```
## hli     0.2465540
## nlcd    1.0000000
##
## $mean
##          cti        err27          ffp          gsp          hli
##    5.3386441    0.4509513   11.2037444  277.2211529 1938.3644530
##         nlcd
##   50.8191308
```

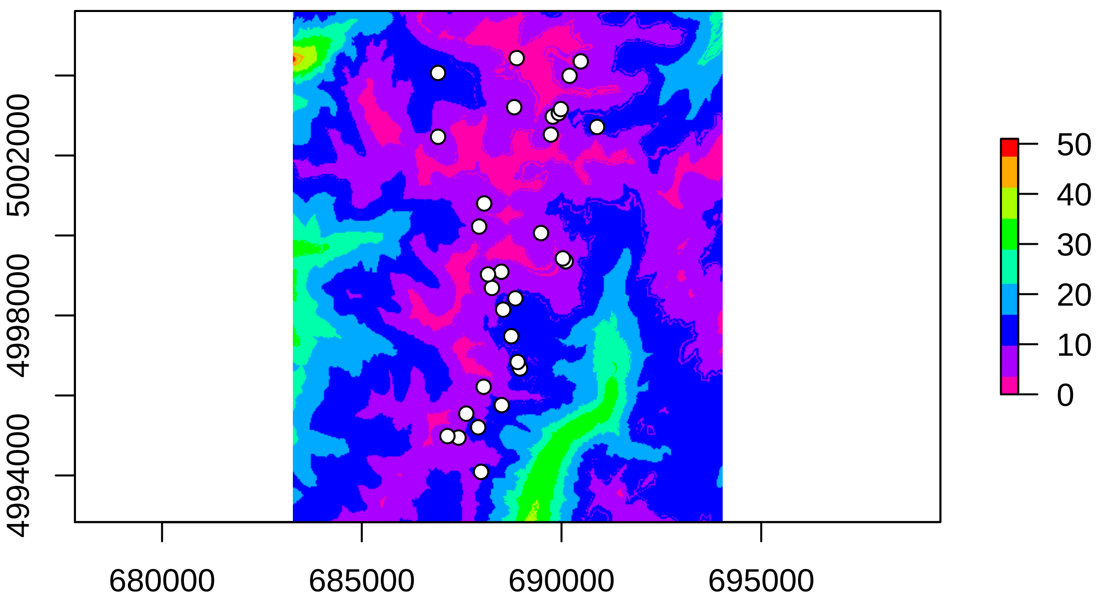
b) Change color ramp, add sampling locations

We can specify a color ramp by setting the ‘col’ argument. The default is ‘terrain.colors(255)’. Here we change it to ‘rainbow(9)’, a rainbow colorpalette with 9 color levels.

Note: To learn about options for the ‘plot’ function for ‘raster’ objects, access the help file by typing ‘?plot’ and select ‘Plot a Raster* object’.

We can add the sampling locations (if we plot only a single raster layer). Here we use ‘rev’ to reverse the color ramp for plotting raster layer ‘ffp’, and add the sites as white circles with black outlines.

```
plot(raster(RALU.rasters.r, layer="ffp"), col=rev(rainbow(9)))
points(RALU.site.sp, pch=21, col="black", bg="white")
```



Extract raster values at sampling locations

The following code adds six variables to the data slot of RALU.site.sp. Technically we combine the columns of the existing data frame ‘RALU.site.sp’ with the new columns in a new data frame with the same name.

R notices the difference in projection (CRS) between the sampling point data and the rasters and takes care of it, providing just a warning.

```
RALU.site.sp@data <- data.frame(RALU.site.sp@data, extract(RALU.rasters.r, RALU.site.sp))
```

```
## Warning in .local(x, y, ...): Transforming SpatialPoints to the CRS of the
## Raster
```

What land cover type is assigned to the most sampling units? Let's tabulate them.

Note: land cover types are coded by numbers. The most frequent type is '42'. Check here what the numbers mean: https://www.mrlc.gov/nlcd06_leg.php

```
table(RALU.site.sp@data$nlcd)
```

```
##
## 11 12 42 52 71 90
##  3  1 21  1  4  1
```

4. Calculate patch-level and class-level landscape metrics

a) Calculate class-level landscape metrics

Here we evaluate the spatial distribution of each cover type (class - this is not the same here as an object class). This is extremely fast in R. But first we'll extract the 'nlcd' raster layer in a separate raster 'NLCD' to simplify the code.

```
NLCD <- raster(RALU.rasters.r, layer="nlcd")
NLCD.class <- ClassStat(NLCD,cellsize=30)
```

For a list of all 37 metrics calculated, check the helpfile for 'ClassStat'. Background information is available on the Fragstats webpage: http://www.umass.edu/landeco/research/fragstats/documents/Metrics/Metrics%20TOC.htm

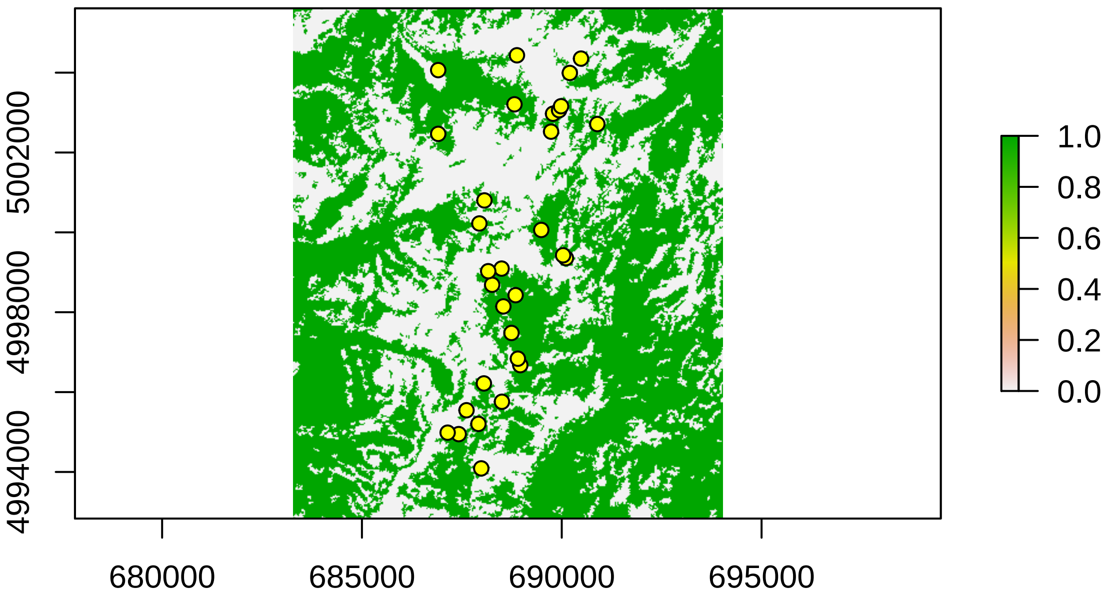
```
?ClassStat
```

b) Calculate patch-level landscape metrics for 'Evergreen Forest'

Calculating patch-level metrics is a little more involved, as we have to decide which cover type (class) to analyze, and then delinate patches for that cover type. Then we calculate statistics for each patch.

The first step is to reduce the land cover map 'nlcd' to a binary map showing evergreen forest vs. any other cover type. We can do this by using a logical test: 'RALU.rasters.r==42', which tests for each cell in NLCD whether it is equal to 42. This results in a binary map, which we can plot, and overlay the sampling locations.

```
Forest <- (NLCD==42)
plot(Forest)
points(RALU.site.sp, pch=21, bg="yellow", col="black")
```



We use the function 'ConnCompLabel' to delineate patches (with the 8-neighbor rule, other rules are not implemented). This creates a new raster 'Patches' where the value in each cell is the new patch ID if evergreen forest, or zero if not. Then we run 'PatchStat' on the new raster.

```
Patches <- ConnCompLabel(Forest)
NLCD.patch <- PatchStat(Patches,cellsize=30)
dim(NLCD.patch)
```

```
## [1] 223  12
```

This returns a list of 223 forest patches (rows) and 12 patch-level landscape metrics (columns). Let's look at the first few patches. Patches differ greatly in size!

Note: The first 'patch', with patchID = 0, contains all cells that are not evergreen forest!

```
head(NLCD.patch)
```

```
##   patchID n.cell n.core.cell n.edges.perimeter n.edges.internal     area
## 1       0  62447       34212             35760           214028 56202300
## 2       1      2           0                 6                2     1800
## 3       2  35332       24092             12898           128430 31798800
## 4       3     19           0                44               32    17100
## 5       4     39           5                46              110    35100
## 6       5      3           0                 8                4     2700
##   core.area perimeter perim.area.ratio shape.index frac.dim.index
## 1  30790800   1072800       0.01908819   35.760000       1.400937
## 2         0       180       0.10000000    1.000000       1.015714
## 3  21682800    386940       0.01216838   17.151596       1.329062
## 4         0      1320       0.07719298    2.444444       1.189944
## 5      4500      1380       0.03931624    1.769231       1.116677
## 6         0       240       0.08888889    1.000000       1.036411
##   core.area.index
## 1       0.5478566
## 2       0.0000000
## 3       0.6818748
## 4       0.0000000
## 5       0.1282051
```

```
## 6        0.0000000
```

For a list of the patch-level metrics calculated, check the helpfile.

```
?PatchStat
```

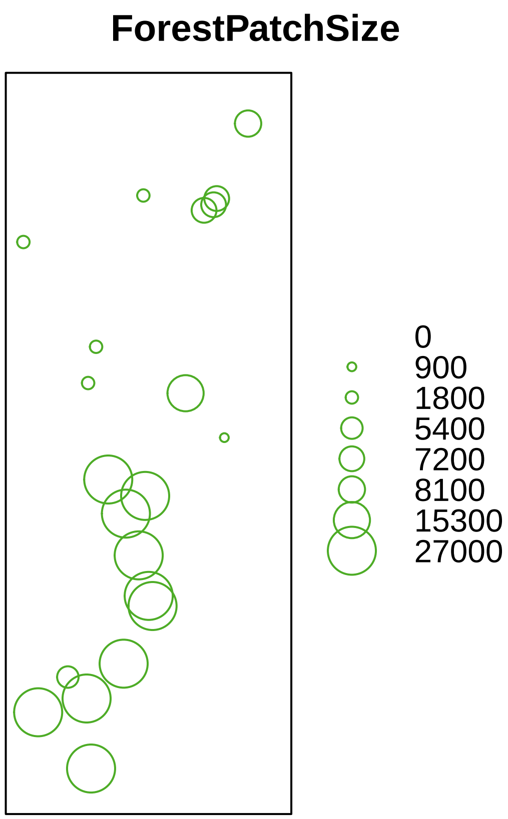
Let's add forest patch size to the RALU.site.sp data. First we need to get the patch ID at each sampling location, then its size.

```
a <- extract.data(RALU.site.sp@coords, Patches) # get patch IDs
a[a==0] <- NA           # this is all the non-forested areas
RALU.site.sp@data$ForestPatchSize <- NLCD.patch[a,"area"]
RALU.site.sp@data$ForestPatchSize[is.na(a)] <- 0
RALU.site.sp@data$ForestPatchSize
```

```
##  [1]  1800     0  1800     0   900 27000 27000 27000     0 27000 27000
## [12]  7200  7200     0     0 27000     0 27000  5400  1800     0 27000
## [23]  8100 27000     0     0  7200  1800     0 27000 15300
```

Plot a bubble map of forest patch size at each sampling location:

```
bubble(RALU.site.sp, "ForestPatchSize", fill=FALSE, key.entries=as.numeric(names(table(RALU.site.sp@data
```



Extract landscape metrics at sampling locations.

a) Calculate class-level metrics in buffer around sampling locations

First we define the buffer radius (in meters) and cell size:

```
Radius <- 500    # Define buffer radius
Cellsize <- 30   # Indicate cell size in meters
```

Then we create a loop through all sampling locations (all rows of the site data set), calculating class-level metrics for each one within its buffer (see video for further explanations).

```
RALU.site.class <- list()

for(i in 1:nrow(RALU.site.sp@data))
{
  # For each raster cell, calculate distance from sampling location
  dist <- distanceFromPoints(NLCD, RALU.site.sp@coords[i,])

  # Create logical raster where the cell with the smallest distance from
  # the sampling location) is 'TRUE' all others 'FALSE'
  site <- (dist== min(values(dist)))

  # Replace 'FALSE' by 'NA' as required for using function 'buffer'
  site[site==FALSE] <- NA

  # Identify cells within buffer around site centerpoint:
  # (this sets each cell within buffer to '1', all other cells to 'NA')
  site.buffer <- buffer(site, Radius)

  # Extract land cover values within buffer (NLCD values within buffer
  # are multiplied by 1, those outside by NA, thus setting them to 'NA')
  NLCD.buffer <- NLCD * site.buffer

  # Calculate class-level metrics within buffer (i.e., for all non-NA cells)
  RALU.site.class[[i]] <- ClassStat(NLCD.buffer,cellsize=30)
}
names(RALU.site.class) <- RALU.site.sp@data$SiteName

# Make sure all sites list all cover types, even if type is absent from buffer:
class.ID <- levels(as.factor(NLCD))[[1]]
RALU.site.class <- lapply(RALU.site.class, function(ls) merge(class.ID, ls, all=TRUE, by.x="ID", by.y="
```

```
RALU.site.class[[2]]
```

```
##   ID n.patches total.area prop.landscape patch.density total.edge
## 1 11         1      33300    0.042189282  1.266945e-06        900
## 2 12         1        900    0.001140251  1.266945e-06        120
## 3 31         1       6300    0.007981756  1.266945e-06        480
## 4 42         4     315000    0.399087799  5.067782e-06       6600
## 5 52         7      39600    0.050171038  8.868618e-06       3360
## 6 71         4     388800    0.492588369  5.067782e-06       6900
## 7 90        NA         NA             NA            NA         NA
## 8 95         1       5400    0.006841505  1.266945e-06        360
##   edge.density landscape.shape.index largest.patch.index mean.patch.area
## 1 0.0011402509              1.153846         0.042189282       33300.000
## 2 0.0001520334              1.000000         0.001140251         900.000
## 3 0.0006081338              1.333333         0.007981756        6300.000
## 4 0.0083618396              2.894737         0.376282782       78750.000
## 5 0.0042569365              4.000000         0.012542759        5657.143
## 6 0.0087419232              2.738095         0.460661345       97200.000
```

```
## 7           NA                  NA                 NA              NA
## 8 0.0004561003             1.200000        0.006841505        5400.000
##   sd.patch.area min.patch.area max.patch.area perimeter.area.frac.dim
## 1            NA          33300          33300              0.05405318
## 2            NA            900            900              0.26651795
## 3            NA           6300           6300              0.15237354
## 4    145559.988           1800         297000              0.04190457
## 5      2685.676            900           9900              0.16968895
## 6    177682.582            900         363600              0.03549367
## 7            NA             NA             NA                      NA
## 8            NA           5400           5400              0.13332222
##   mean.perim.area.ratio sd.perim.area.ratio min.perim.area.ratio
## 1            0.02702703                  NA           0.02702703
## 2            0.13333333                  NA           0.13333333
## 3            0.07619048                  NA           0.07619048
## 4            0.06282828          0.03377667           0.01797980
## 5            0.09193568          0.02036396           0.07619048
## 6            0.06375413          0.05030518           0.01501650
## 7                    NA                  NA                   NA
## 8            0.06666667                  NA           0.06666667
##   max.perim.area.ratio mean.shape.index sd.shape.index min.shape.index
## 1           0.02702703         1.153846             NA        1.153846
## 2           0.13333333         1.000000             NA        1.000000
## 3           0.07619048         1.333333             NA        1.333333
## 4           0.10000000         1.507601      0.6671214        1.000000
## 5           0.13333333         1.460544      0.2950167        1.000000
## 6           0.13333333         1.528092      0.5640899        1.000000
## 7                   NA               NA             NA              NA
## 8           0.06666667         1.200000             NA        1.200000
##   max.shape.index mean.frac.dim.index sd.frac.dim.index min.frac.dim.index
## 1        1.153846            1.040226                NA           1.040226
## 2        1.000000            1.000000                NA           1.000000
## 3        1.333333            1.094496                NA           1.094496
## 4        2.405405            1.077566        0.06566773           1.015714
## 5        1.857143            1.100678        0.04903234           1.000000
## 6        2.219512            1.070766        0.06508335           1.000000
## 7              NA                  NA                NA                 NA
## 8        1.200000            1.047179                NA           1.047179
##   max.frac.dim.index total.core.area prop.landscape.core
## 1           1.040226           10800          0.01368301
## 2           1.000000               0          0.00000000
## 3           1.094496               0          0.00000000
## 4           1.142196          166500          0.21094641
## 5           1.146268               0          0.00000000
## 6           1.127619          223200          0.28278221
## 7                 NA              NA                  NA
## 8           1.047179               0          0.00000000
##   mean.patch.core.area sd.patch.core.area min.patch.core.area
## 1                10800                 NA               10800
## 2                    0                 NA                   0
## 3                    0                 NA                   0
## 4                41625            83250.0                   0
## 5                    0                0.0                   0
## 6                55800           111000.8                   0
```

```
## 7                   NA                 NA                 NA
## 8                    0                 NA                  0
##   max.patch.core.area prop.like.adjacencies aggregation.index
## 1               10800             0.6629213          96.72131
## 2                   0             0.0000000           0.00000
## 3                   0             0.2727273          75.00000
## 4              166500             0.7283951          89.12387
## 5                   0             0.2222222          43.24324
## 6              222300             0.7650664          91.11922
## 7                  NA                    NA                NA
## 8                   0             0.3333333          85.71429
##   lanscape.division.index splitting.index effective.mesh.size
## 1               0.9982201    5.618181e+02        1.404903e+03
## 2               0.9999987    7.691290e+05        1.026226e+00
## 3               0.9999363    1.569651e+04        5.028506e+01
## 4               0.8581538    7.049891e+00        1.119592e+05
## 5               0.9995709    2.330694e+03        3.386545e+02
## 6               0.7873101    4.701680e+00        1.678762e+05
## 7                      NA              NA                  NA
## 8               0.9999532    2.136469e+04        3.694413e+01
##   patch.cohesion.index
## 1             8.073848
## 2                  NaN
## 3             6.010309
## 4             9.115865
## 5             6.183699
## 6             9.164208
## 7                   NA
## 8             5.717697
```

b) Extract landscape metric of choice for a single cover type (as vector)

Now we can extract any variable of interest for any cover type of interest. Here we'll extract the percentage of evergreen forest within a 500 m radius around each site. See tutorial for the use of lapply.

```r
# Extract one variable, 'prop.landscape', for one cover type 42 (Evergreen Forest)
# (this returns a vector with a single value for each site)
PercentForest500 <- unlist(lapply(RALU.site.class, function(ls) ls[ls$ID==42, "prop.landscape"]))
PercentForest500[is.na(PercentForest500)] <- 0
PercentForest500
```

```
##    AirplaneLake  BachelorMeadow  BarkingFoxLake    BirdbillLake
##       0.7981756       0.3990878       0.3751425       0.3055872
##         BobLake       CacheLake         DoeLake    EggWhiteLake
##       0.3797035       0.8392246       0.7137970       0.8825542
##     ElenasLake        FawnLake     FrogPondLake     GentianLake
##       0.1071836       0.7274800       0.9258837       0.3705815
##    GentianPonds      GoldenLake      GreggsLake    InandOutLake
##       0.3660205       0.2998860       0.3078677       0.6111745
##      MeadowLake       MooseLake    Mt.WilsonLake       NopezLake
##       0.6225770       0.5473204       0.3375143       0.7092360
##     ParagonLake ParagonWetland     PotholeLake     RamshornLake
##       0.4720639       0.3192702       0.2405929       0.5017104
##  ShipIslandLake     SkyhighLake StockingCapLake     Terrace1Lake
```

```
##       0.6168757       0.3215507       0.3067275       0.3147092
##      TobiasLake   WalkaboutLake     WelcomeLake
##       0.4310148       0.3272520       0.6989738
```

c) Extract landscape metric of choice for all cover types (as data frame)

To extract the landscape metric 'prop.landscape' for all cover types as a data.frame (one column per cover type), use this code.

We'll define column names combining 'Prop' for 'proportion of landscape', '500' to indicate the 500 m buffer radius, and the ID of each cover type.

```
tmp <- Reduce(rbind,lapply(RALU.site.class, function(ls) ls[, "prop.landscape"]))
dimnames(tmp) <- list(row.names=names(RALU.site.class),
                      col.names=paste("Prop.500", class.ID$ID, sep="."))
tmp[is.na(tmp)] <- 0
RALU.prop.landscape500 <- as.data.frame(tmp)
head(RALU.prop.landscape500)
```

```
##                 Prop.500.11 Prop.500.12 Prop.500.31 Prop.500.42 Prop.500.52
## AirplaneLake     0.08209806 0.000000000 0.000000000   0.7981756 0.006841505
## BachelorMeadow   0.04218928 0.001140251 0.007981756   0.3990878 0.050171038
## BarkingFoxLake   0.01710376 0.000000000 0.013683010   0.3751425 0.148232611
## BirdbillLake     0.00000000 0.020524515 0.000000000   0.3055872 0.036488027
## BobLake          0.00000000 0.000000000 0.000000000   0.3797035 0.118586089
## CacheLake        0.03876853 0.000000000 0.000000000   0.8392246 0.038768529
##                 Prop.500.71 Prop.500.90 Prop.500.95
## AirplaneLake     0.11288483 0.000000000 0.000000000
## BachelorMeadow   0.49258837 0.000000000 0.006841505
## BarkingFoxLake   0.44583808 0.000000000 0.000000000
## BirdbillLake     0.62257697 0.005701254 0.009122007
## BobLake          0.50171038 0.000000000 0.000000000
## CacheLake        0.08323831 0.000000000 0.000000000
```

d) Extract all landscape metrics for a single cover type (as data frame)

To extract all landscape metrics for a single cover type, we need to modify the code like this. Here we add the class ID '42' to all variable names to indicate that these are quantified for cover type '42' (evergreen forest)

```
tmp <- Reduce(rbind,lapply(RALU.site.class, function(ls) ls[ls$ID==42, ]))
dimnames(tmp) <- list(row.names=names(RALU.site.class),
                      col.names=paste("42",names(tmp), sep="."))
RALU.forest.500 <- as.data.frame(tmp)
head(RALU.forest.500)
```

```
##                 42.ID 42.n.patches 42.total.area 42.prop.landscape
## AirplaneLake       42            2        630000         0.7981756
## BachelorMeadow     42            4        315000         0.3990878
## BarkingFoxLake     42           10        296100         0.3751425
## BirdbillLake       42            4        241200         0.3055872
## BobLake            42            4        299700         0.3797035
## CacheLake          42            1        662400         0.8392246
##                 42.patch.density 42.total.edge 42.edge.density
## AirplaneLake        2.533891e-06          8580     0.010870391
## BachelorMeadow      5.067782e-06          6600     0.008361840
```

```
## BarkingFoxLake     1.266945e-05         10020     0.012694793
## BirdbillLake       5.067782e-06          7800     0.009882174
## BobLake            5.067782e-06          9780     0.012390726
## CacheLake          1.266945e-06          8580     0.010870391
##                 42.landscape.shape.index 42.largest.patch.index
## AirplaneLake                    2.698113              0.7833523
## BachelorMeadow                  2.894737              0.3762828
## BarkingFoxLake                  4.513514              0.2656784
## BirdbillLake                    3.939394              0.2633979
## BobLake                         4.405405              0.1254276
## CacheLake                       2.600000              0.8392246
##                 42.mean.patch.area 42.sd.patch.area 42.min.patch.area
## AirplaneLake                315000        428930.97             11700
## BachelorMeadow               78750        145559.99              1800
## BarkingFoxLake               29610         64220.25               900
## BirdbillLake                 60300         98502.39              6300
## BobLake                      74925         22013.23             48600
## CacheLake                   662400               NA            662400
##                 42.max.patch.area 42.perimeter.area.frac.dim
## AirplaneLake               618300                 0.02723807
## BachelorMeadow             297000                 0.04190457
## BarkingFoxLake             209700                 0.06767906
## BirdbillLake               207900                 0.06467639
## BobLake                     99000                 0.06526510
## CacheLake                  662400                 0.02590579
##                 42.mean.perim.area.ratio 42.sd.perim.area.ratio
## AirplaneLake                  0.03460979            0.030830521
## BachelorMeadow                0.06282828            0.033776666
## BarkingFoxLake                0.08269031            0.045796428
## BirdbillLake                  0.05868459            0.024104140
## BobLake                       0.03406556            0.008526118
## CacheLake                     0.01295290                     NA
##                 42.min.perim.area.ratio 42.max.perim.area.ratio
## AirplaneLake                 0.01280932              0.05641026
## BachelorMeadow               0.01797980              0.10000000
## BarkingFoxLake               0.02775393              0.13333333
## BirdbillLake                 0.02712843              0.07878788
## BobLake                      0.02666667              0.04567901
## CacheLake                    0.01295290              0.01295290
##                 42.mean.shape.index 42.sd.shape.index 42.min.shape.index
## AirplaneLake               1.932783         0.7888243           1.375000
## BachelorMeadow             1.507601         0.6671214           1.000000
## BarkingFoxLake             1.430339         0.6743979           1.000000
## BirdbillLake               1.972350         0.7390224           1.333333
## BobLake                    2.209921         0.3415475           1.777778
## CacheLake                  2.600000                NA           2.600000
##                 42.max.shape.index 42.mean.frac.dim.index
## AirplaneLake              2.490566               1.114334
## BachelorMeadow            2.405405               1.077566
## BarkingFoxLake            3.129032               1.058012
## BirdbillLake              3.032258               1.134129
## BobLake                   2.500000               1.144388
## CacheLake                 2.600000               1.144600
##                 42.sd.frac.dim.index 42.min.frac.dim.index
```

```
## AirplaneLake           0.03418863            1.090159
## BachelorMeadow         0.06566773            1.015714
## BarkingFoxLake         0.06535875            1.000000
## BirdbillLake           0.03984718            1.094496
## BobLake                0.02878016            1.111747
## CacheLake                      NA            1.144600
##                 42.max.frac.dim.index 42.total.core.area
## AirplaneLake                 1.138509             404100
## BachelorMeadow               1.142196             166500
## BarkingFoxLake               1.188689              78300
## BirdbillLake                 1.184395              73800
## BobLake                      1.171114              78300
## CacheLake                    1.144600             433800
##                 42.prop.landscape.core 42.mean.patch.core.area
## AirplaneLake                0.51197263                  202050
## BachelorMeadow              0.21094641                   41625
## BarkingFoxLake              0.09920182                    7830
## BirdbillLake                0.09350057                   18450
## BobLake                     0.09920182                   19575
## CacheLake                   0.54960091                  433800
##                 42.sd.patch.core.area 42.min.patch.core.area
## AirplaneLake                285741.85                      0
## BachelorMeadow               83250.00                      0
## BarkingFoxLake               21466.46                      0
## BirdbillLake                 36302.48                      0
## BobLake                      11766.16                   5400
## CacheLake                          NA                 433800
##                 42.max.patch.core.area 42.prop.like.adjacencies
## AirplaneLake                    404100                0.8146468
## BachelorMeadow                  166500                0.7283951
## BarkingFoxLake                   68400                0.5951515
## BirdbillLake                     72900                0.6096096
## BobLake                          33300                0.6067551
## CacheLake                       433800                0.8229102
##                 42.aggregation.index 42.lanscape.division.index
## AirplaneLake                93.31849                  0.3861394
## BachelorMeadow              89.12387                  0.8581538
## BarkingFoxLake              79.06602                  0.9263466
## BirdbillLake                80.71571                  0.9299311
## BobLake                     79.96820                  0.9616228
## CacheLake                   93.78970                  0.2957020
##                 42.splitting.index 42.effective.mesh.size
## AirplaneLake              1.629034              484520.18
## BachelorMeadow            7.049891              111959.18
## BarkingFoxLake           13.577098               58134.66
## BirdbillLake             14.271673               55305.36
## BobLake                  26.057154               30291.11
## CacheLake                 1.419854              555902.39
##                 42.patch.cohesion.index
## AirplaneLake                   9.289850
## BachelorMeadow                 9.115865
## BarkingFoxLake                 8.903398
## BirdbillLake                   8.978783
## BobLake                        8.634823
```

```
## CacheLake                        9.306166
```

e) Append to site data set

```
RALU.site.sp@data <- data.frame(RALU.site.sp@data, RALU.prop.landscape500,
                                RALU.forest.500)
```

Note: check this week's bonus material if you want to see how to use the new 'sf' library for spatial data, and how to export the site data to an shapefile that you can import into a GIS.