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.site: SpatialPointsDataFrame object with UTM coordinates (zone 11) in slot @coords and 17 site variables in slot @data for 31 sites. The data are included in the 'GeNetIt' package, for meta data type: ?ralu.site

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

- cti: Compound Topographic Index ("wetness")
- err27: Elevation Relief Ratio
- ffp: Frost Free Period
- gsp: Growing Season Precipitation
- hli: Heat Load Index
- nlcd: USGS Landcover (categorical map)

c) Required R libraries

```
require(sp)
require(raster)
require(GeNetIt)
require(tmaptools)
require(SDMTools) # for landscape metrics
getwd()           # check your working directory
```

```
## [1] "/Users/hhwagner1/Desktop/TestCoursePackage/inst/WE"
```

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 site data are already in a SpatialPointsDataFrame. To illustrate importing spatial data from Excel, we export the data as a csv file, import it again as a data frame, then convert it to a SpatialPointsDataFrame.

First we piece together the coordinates from the @coords slot and the attribute data from the @data slot into a single data frame.

```
data(ralu.site)
write.csv(data.frame(ralu.site@coords, ralu.site@data), file="ralu.site.csv", quote=FALSE, row.names=FA
Sites <- read.csv("ralu.site.csv", header=TRUE)
head(Sites)
```

```
##   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 17 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'.

```
Sites.sp <- Sites
coordinates(Sites.sp) <- ~coords.x1+coords.x2
head(Sites.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/OverviewCoordinateReferenceSystem pdf

We know that these coordinates are UTM zone 11 (Northern hemisphere) coordinates, hence we can use a helper function to find the correct 'proj4string', 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).

```
proj4string(Sites.sp) <- tmaptools::get_proj4("utm11")
```

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

c) Access data in 'SpatialPointsDataFrame'

As an S4 object, Sites.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

```
slotNames(Sites.sp)
```

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

Here are the first few lines of the coordinates:

```
head(Sites.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: Let's compare this to the proj4string of the original 'ralu.site' dataset

```
Sites.sp@proj4string
```

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

The default for 'get_proj4("utm11")' results in a slightly different proj4string than the 'ralu.site' dataset. The difference is in the 'datum' argument ('WGS84' vs. 'NAD83'):

```
ralu.site@proj4string
```

```
## CRS arguments:
##  +proj=utm +zone=11 +datum=NAD83 +units=m +no_defs +ellps=GRS80
## +towgs84=0,0,0
```

Let's go with the original information and copy it:

```
Sites.sp@proj4string <- ralu.site@proj4string
```

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'. Let's convert the data to a 'RasterStack' of 'RasterLayer' objects (i.e. a set of raster layers with the same spatial reference information).

```
RasterMaps <- stack(rasters)
class(RasterMaps)
```

```
## [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.

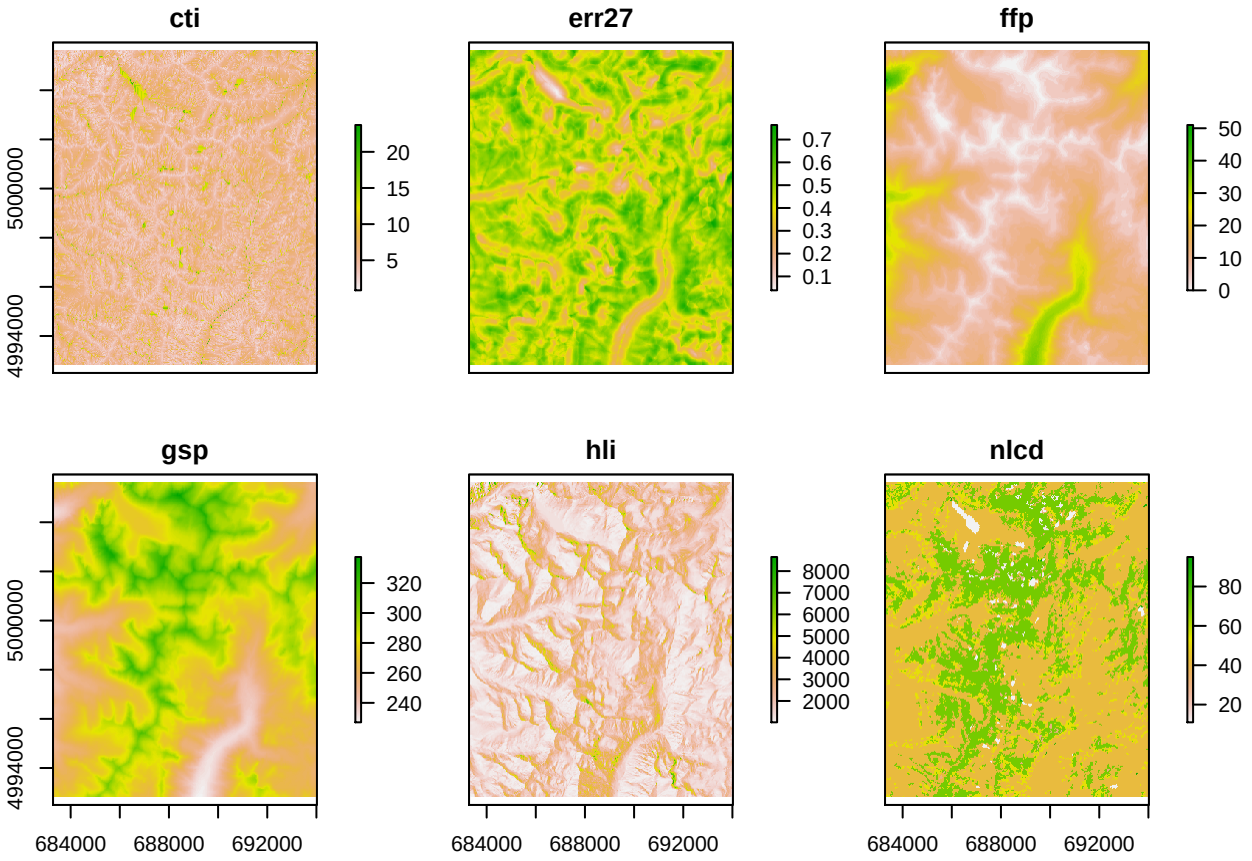
```
RasterMaps
```

```
## 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(RasterMaps)
```



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(RasterMaps, '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
```

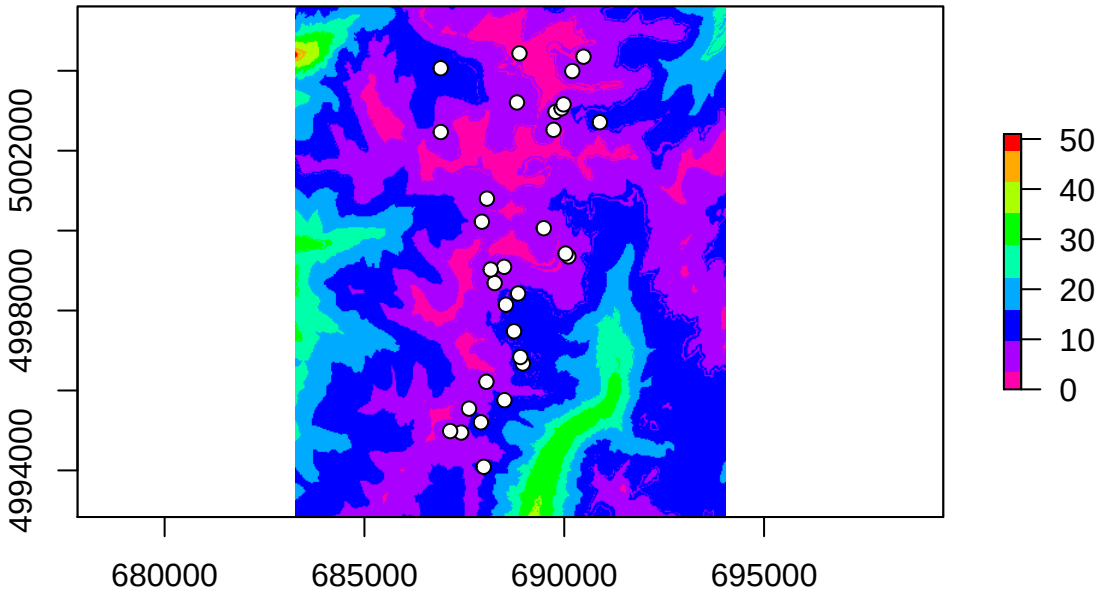
b) Change color ramp, add sampling locations

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(RasterMaps, layer="ffp"), col=rev(rainbow(9)))
points(Sites.sp, pch=21, col="black", bg="white")
```



Extract raster values at sampling locations

The following code adds six variables to the data slot of Sites.sp. Technically we combine the columns of the existing data frame 'Sites.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.

```
Sites.sp@data <- data.frame(Sites.sp@data, extract(RasterMaps, Sites.sp))
```

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

Note: land cover types are coded by numbers. A total of 21 sites are classified as '42'. Check here what the numbers mean: https://www.mrlc.gov/nlcd06_leg.php

```
table(Sites.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, using the function 'ClassStat' from the package 'SDMTools'. But first we'll extract the 'nlcd' raster layer in a separate raster 'NLCD' to simplify the code.

```
NLCD <- raster(RasterMaps, layer="nlcd")
NLCD.class <- SDMTools::ClassStat(NLCD,cellsize=30)
```

For a list of all 37 metrics calculated, check the helpfile for 'ClassStat'. Which metric would you use to quantify the percent forest cover in the landscape?

Background information is available on the Fragstats webpage: http://www.umass.edu/landeco/research/fragstats/documents/Metrics/Metrics%20TOC.htm

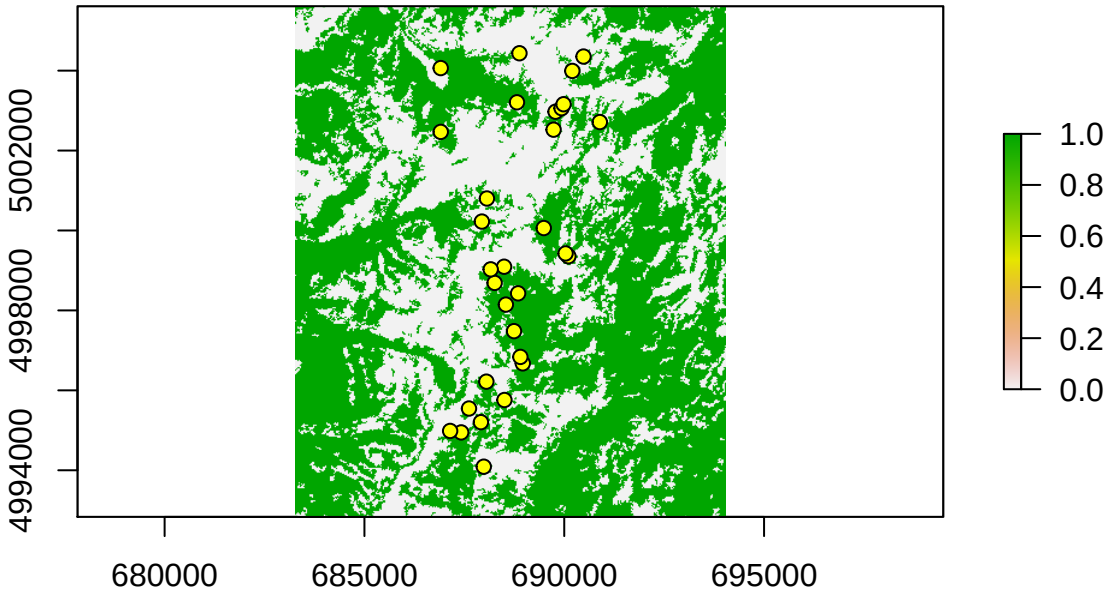
```
?ClassStat
```

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: 'RasterMaps==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(Sites.sp, pch=21, bg="yellow", col="black")
```



We use the function 'ConnCompLabel' (package 'SDMTools) to delineate patches with the 8-neighbor rule (other rules are not implemented at this time). 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 <- SDMTools::ConnCompLabel(Forest)
NLCD.patch <- SDMTools::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. Which metric would you use to quantify patch size?

```
?PatchStat
```

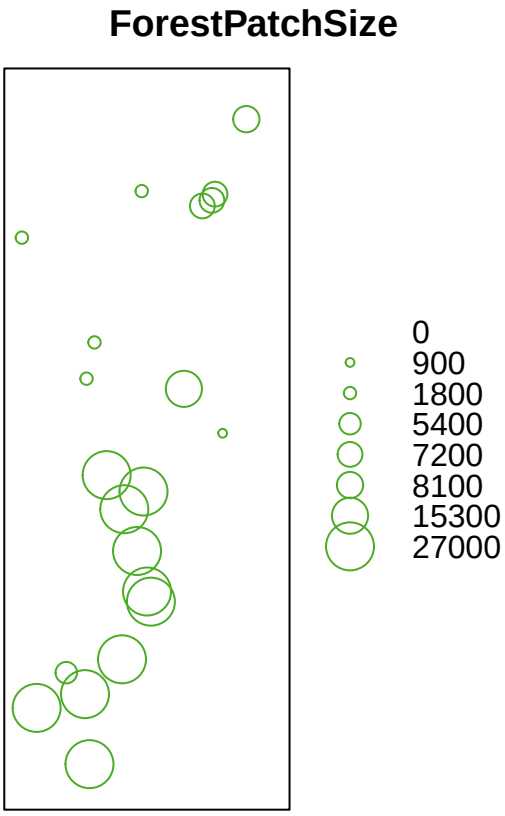
Let's add forest patch size to the 'Sites.sp' data. First we need to get the patch ID at each sampling location, then its size.

```
a <- extract.data(Sites.sp@coords, Patches)   # get patch IDs
a[a==0] <- NA                                 # these are the non-forested areas
Sites.sp@data$ForestPatchSize <- NLCD.patch[a,"area"]
Sites.sp@data$ForestPatchSize[is.na(a)] <- 0  # set patch size to zero for nonforested
Sites.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(Sites.sp, "ForestPatchSize", fill=FALSE, key.entries=as.numeric(names(table(Sites.sp@data$Forestl
```



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).

```
Sites.class <- list()
class.ID <- levels(ratify(NLCD))[[1]]

for(i in 1:nrow(Sites.sp@data))
{
  # Identify all cells that lie within buffer around site i:
  Buffer.cells <- extract(NLCD, Sites.sp[i,], cellnumbers=TRUE,
                          buffer=Radius)[[1]][,1]

  # Copy land cover map and delete all values outside of buffer:
  Buffer.nlcd <- NLCD
  values(Buffer.nlcd)[-Buffer.cells] <- NA

  # Calculate class-level metrics for cells within buffer:
  Result <- ClassStat(Buffer.nlcd,cellsize=Cellsize)
```

```
  # Merge Results table with 'class.ID' to ensure that all cover types
  # are listed for all sites, even if they are not present in buffer,
  # write results into ith element of list 'Sites.class':
  Sites.class[[i]] <- merge(class.ID, Result, all=TRUE, by.x="ID", by.y="class")
}
# Add labels for list elements
names(Sites.class) <- Sites.sp@data$SiteName
```

```
Sites.class[[2]]
```

```
##   ID n.patches total.area prop.landscape patch.density total.edge
## 1 11         1      32400    0.041426928  1.278609e-06        900
## 2 12         1        900    0.001150748  1.278609e-06        120
## 3 31         1       7200    0.009205984  1.278609e-06        480
## 4 42         5     311400    0.398158803  6.393044e-06       6540
## 5 52         7      38700    0.049482163  8.950262e-06       3300
## 6 71         3     386100    0.493670886  3.835827e-06       6540
## 7 90        NA         NA             NA            NA         NA
## 8 95         1       5400    0.006904488  1.278609e-06        360
##   edge.density landscape.shape.index largest.patch.index mean.patch.area
## 1 0.0011507480              1.250000         0.041426928       32400.000
## 2 0.0001534331              1.000000         0.001150748         900.000
## 3 0.0006137323              1.333333         0.009205984        7200.000
## 4 0.0083621020              2.868421         0.373993096       62280.000
## 5 0.0042194093              3.928571         0.012658228        5528.571
## 6 0.0083621020              2.595238         0.466052934      128700.000
## 7           NA                    NA                  NA             NA
## 8 0.0004602992              1.200000         0.006904488        5400.000
##   sd.patch.area min.patch.area max.patch.area perimeter.area.frac.dim
## 1            NA          32400          32400              0.05555466
## 2            NA            900            900              0.26651795
## 3            NA           7200           7200              0.13332677
## 4    128766.696            900         292500              0.04200356
## 5      2671.276            900           9900              0.17053431
## 6    204210.773           9900         364500              0.03387714
## 7            NA             NA             NA                      NA
## 8            NA           5400           5400              0.13332222
##   mean.perim.area.ratio sd.perim.area.ratio min.perim.area.ratio
## 1            0.02777778                  NA           0.02777778
## 2            0.13333333                  NA           0.13333333
## 3            0.06666667                  NA           0.06666667
## 4            0.07686154          0.04312596           0.01764103
## 5            0.09216244          0.02016730           0.07619048
## 6            0.03793797          0.02107661           0.01497942
## 7                    NA                  NA                   NA
## 8            0.06666667                  NA           0.06666667
##   max.perim.area.ratio mean.shape.index sd.shape.index min.shape.index
## 1           0.02777778         1.250000             NA        1.250000
## 2           0.13333333         1.000000             NA        1.000000
## 3           0.06666667         1.333333             NA        1.333333
## 4           0.13333333         1.389865      0.5883211        1.000000
## 5           0.13333333         1.470068      0.2912780        1.000000
## 6           0.05641026         1.531504      0.6246380        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        1.250000            1.042970                NA           1.042970
## 2        1.000000            1.000000                NA           1.000000
## 3        1.333333            1.078041                NA           1.078041
## 4        2.324324            1.061240        0.06540615           1.000000
## 5        1.857143            1.099043        0.04946169           1.000000
## 6        2.219512            1.076420        0.05906228           1.011699
## 7              NA                  NA                NA                 NA
## 8        1.200000            1.047179                NA           1.047179
##   max.frac.dim.index total.core.area prop.landscape.core
## 1           1.042970           10800          0.01380898
## 2           1.000000               0          0.00000000
## 3           1.078041               0          0.00000000
## 4           1.138133          164700          0.21058688
## 5           1.146268               0          0.00000000
## 6           1.127401          224100          0.28653625
## 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                32940           73656.08                    0
## 5                    0               0.00                    0
## 6                74700          128605.56                    0
## 7                   NA                 NA                   NA
## 8                    0                 NA                    0
##   max.patch.core.area prop.like.adjacencies aggregation.index
## 1               10800             0.6551724          96.61017
## 2                   0             0.0000000           0.00000
## 3                   0             0.3333333          80.00000
## 4              164700             0.7278402          89.14373
## 5                   0             0.2198582          43.05556
## 6              223200             0.7745605          91.78922
## 7                  NA                    NA                NA
## 8                   0             0.3333333          85.71429
##   lanscape.division.index splitting.index effective.mesh.size
## 1               0.9982838    5.826860e+02        1.342232e+03
## 2               0.9999987    7.551610e+05        1.035673e+00
## 3               0.9999152    1.179939e+04        6.628308e+01
## 4               0.8598656    7.136009e+00        1.095991e+05
## 5               0.9995802    2.382211e+03        3.283084e+02
## 6               0.7824106    4.595813e+00        1.701766e+05
## 7                     NA              NA                  NA
## 8               0.9999523    2.097669e+04        3.728423e+01
##   patch.cohesion.index
## 1             8.050644
## 2                 NaN
## 3             6.245174
## 4             9.110058
## 5             6.148054
## 6             9.164259
## 7                   NA
```

```
## 8             5.716779
```

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.

```
# Extract one variable, 'prop.landscape', for one cover type 42 (Evergreen Forest)
# (this returns a vector with a single value for each site)

PercentForest500 <- rep(NA, length(Sites.class))  # Create empty results vector
for(i in 1:length(Sites.class))
{
  # For site i, select row with cover type '42' and column 'prop.landscape':
  PercentForest500[i] <- Sites.class[[i]][class.ID$ID==42, "prop.landscape"]
}

# If there are any sites with no forest in buffer, set value to 0:
PercentForest500[is.na(PercentForest500)] <- 0

# Print results:
PercentForest500
```

```
##  [1] 0.7965517 0.3981588 0.3770115 0.3119266 0.3791523 0.8478261 0.7082380
##  [8] 0.8812785 0.1056257 0.7252874 0.9255441 0.3665521 0.3569794 0.3017143
## [15] 0.3024055 0.6242841 0.6258581 0.5518814 0.3360000 0.6994286 0.4817352
## [22] 0.3203215 0.2342857 0.5000000 0.6181193 0.3230241 0.3111366 0.3162100
## [29] 0.4388571 0.3222857 0.7124857
```

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.

```
# Create empty matrix for storing results:
Prop.landscape <- matrix(data=NA, nrow=length(Sites.class), ncol=length(class.ID$ID))

# Create row and column names:
dimnames(Prop.landscape) <- list(names(Sites.class),
                                 paste("Prop.500", class.ID$ID, sep="."))

# For each site i, extract "prop.landscape" for all cover types
# and write results into row i of Prop.landscape:
for(i in 1:length(Sites.class))
{
  Prop.landscape[i,] <- Sites.class[[i]][,"prop.landscape"]
}

# Set any missing values to 0:
Prop.landscape[is.na(Prop.landscape)] <- 0
```

```
# Convert matrix to data frame:
Prop.landscape <- as.data.frame(Prop.landscape)
head(Prop.landscape)
```

```
##                Prop.500.11 Prop.500.12 Prop.500.31 Prop.500.42 Prop.500.52
## AirplaneLake     0.08275862 0.000000000 0.000000000   0.7965517 0.005747126
## BachelorMeadow   0.04142693 0.001150748 0.009205984   0.3981588 0.049482163
## BarkingFoxLake   0.01724138 0.000000000 0.010344828   0.3770115 0.150574713
## BirdbillLake     0.00000000 0.020642202 0.000000000   0.3119266 0.036697248
## BobLake          0.00000000 0.000000000 0.000000000   0.3791523 0.117983963
## CacheLake        0.03890160 0.000000000 0.000000000   0.8478261 0.038901602
##                Prop.500.71 Prop.500.90 Prop.500.95
## AirplaneLake     0.11494253 0.000000000 0.000000000
## BachelorMeadow   0.49367089 0.000000000 0.006904488
## BarkingFoxLake   0.44482759 0.000000000 0.000000000
## BirdbillLake     0.61582569 0.005733945 0.009174312
## BobLake          0.50286369 0.000000000 0.000000000
## CacheLake        0.07437071 0.000000000 0.000000000
```

The percent cover of all cover types should add up to 100% (i.e., 1) for each site. We can check this with the function 'apply'. The argument 'MARGIN' specifies whether we want to apply the function FUN to each row (MARGIN=1) or each column (MARGIN=2).

Note: This function expects the object X to be a matrix or array - taking a row total only makes sense if all columns contain the same type of data, in the same units. It still does the calculation here even though we just converted 'Prop.landscape' to a data frame. Always double check whether what you ask R to calculate makes sense.

```
apply(X=Prop.landscape, MARGIN=1, FUN=sum)
```

```
##    AirplaneLake  BachelorMeadow  BarkingFoxLake    BirdbillLake
##               1               1               1               1
##         BobLake       CacheLake         DoeLake    EggWhiteLake
##               1               1               1               1
##      ElenasLake        FawnLake    FrogPondLake     GentianLake
##               1               1               1               1
##    GentianPonds      GoldenLake      GreggsLake    InandOutLake
##               1               1               1               1
##      MeadowLake       MooseLake   Mt.WilsonLake       NopezLake
##               1               1               1               1
##     ParagonLake  ParagonWetland     PotholeLake    RamshornLake
##               1               1               1               1
##  ShipIslandLake     SkyhighLake StockingCapLake    Terrace1Lake
##               1               1               1               1
##      TobiasLake   WalkaboutLake     WelcomeLake
##               1               1               1
```

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)

```
# Create empty matrix for storing results:
Forest.class <- matrix(data=NA, nrow=length(Sites.class),
                       ncol=ncol(Sites.class[[1]]))
```

```r
# Create row and column names:
dimnames(Forest.class) <- list(names(Sites.class),
                               paste("42",names(Sites.class[[1]]), sep="."))

# For each site i, extract all landscape metrics for cover type 42
# and write results into row i of Forest.class:
for(i in 1:length(Sites.class))
{
  Forest.class[i,] <- unlist(Sites.class[[i]][class.ID$ID==42,])
}

# Convert matrix to data frame:
Forest.class <- as.data.frame(Forest.class)
head(Forest.class)
```

```
##                42.ID 42.n.patches 42.total.area 42.prop.landscape
## AirplaneLake      42            2        623700         0.7965517
## BachelorMeadow    42            5        311400         0.3981588
## BarkingFoxLake    42           10        295200         0.3770115
## BirdbillLake      42            5        244800         0.3119266
## BobLake           42            4        297900         0.3791523
## CacheLake         42            1        666900         0.8478261
##                42.patch.density 42.total.edge 42.edge.density
## AirplaneLake       2.554278e-06          8460     0.010804598
## BachelorMeadow     6.393044e-06          6540     0.008362102
## BarkingFoxLake     1.277139e-05          9960     0.012720307
## BirdbillLake       6.371050e-06          7920     0.010091743
## BobLake            5.091002e-06          9780     0.012447499
## CacheLake          1.271294e-06          8280     0.010526316
##                42.landscape.shape.index 42.largest.patch.index
## AirplaneLake                   2.660377              0.7839080
## BachelorMeadow                 2.868421              0.3739931
## BarkingFoxLake                 4.486486              0.2689655
## BirdbillLake                   4.000000              0.2649083
## BobLake                        4.405405              0.1260023
## CacheLake                      2.509091              0.8478261
##                42.mean.patch.area 42.sd.patch.area 42.min.patch.area
## AirplaneLake               311850        427021.79              9900
## BachelorMeadow              62280        128766.70               900
## BarkingFoxLake              29520         64536.69               900
## BirdbillLake                48960         89124.09              1800
## BobLake                     74475         21741.72             48600
## CacheLake                  666900               NA            666900
##                42.max.patch.area 42.perimeter.area.frac.dim
## AirplaneLake              613800                 0.02712840
## BachelorMeadow            292500                 0.04200356
## BarkingFoxLake            210600                 0.06747889
## BirdbillLake              207900                 0.06470555
## BobLake                    99000                 0.06565945
## CacheLake                 666900                 0.02483130
##                42.mean.perim.area.ratio 42.sd.perim.area.ratio
## AirplaneLake                 0.03670577            0.033800119
## BachelorMeadow               0.07686154            0.043125955
```

```
## BarkingFoxLake               0.08278956            0.045749445
## BirdbillLake                 0.06684945            0.028844942
## BobLake                      0.03425420            0.008564533
## CacheLake                    0.01241565                     NA
##                 42.min.perim.area.ratio 42.max.perim.area.ratio
## AirplaneLake                 0.01280547              0.06060606
## BachelorMeadow               0.01764103              0.13333333
## BarkingFoxLake               0.02763533              0.13333333
## BirdbillLake                 0.02683983              0.10000000
## BobLake                      0.02666667              0.04567901
## CacheLake                    0.01241565              0.01241565
##                 42.mean.shape.index 42.sd.shape.index 42.min.shape.index
## AirplaneLake               1.950135         0.7376020           1.428571
## BachelorMeadow             1.389865         0.5883211           1.000000
## BarkingFoxLake             1.417839         0.6741217           1.000000
## BirdbillLake               1.711111         0.8225225           1.000000
## BobLake                    2.209921         0.3415475           1.777778
## CacheLake                  2.509091                NA           2.509091
##                 42.max.shape.index 42.mean.frac.dim.index
## AirplaneLake              2.471698               1.113613
## BachelorMeadow            2.324324               1.061240
## BarkingFoxLake            3.129032               1.057342
## BirdbillLake              3.000000               1.099142
## BobLake                   2.500000               1.144935
## CacheLake                 2.509091               1.138714
##                 42.sd.frac.dim.index 42.min.frac.dim.index
## AirplaneLake              0.03447658              1.089234
## BachelorMeadow            0.06540615              1.000000
## BarkingFoxLake            0.06552241              1.000000
## BirdbillLake              0.07100146              1.015714
## BobLake                   0.02934049              1.111747
## CacheLake                         NA              1.138714
##                 42.max.frac.dim.index 42.total.core.area
## AirplaneLake                 1.137991             402300
## BachelorMeadow               1.138133             164700
## BarkingFoxLake               1.188274              79200
## BirdbillLake                 1.182648              73800
## BobLake                      1.171114              78300
## CacheLake                    1.138714             444600
##                 42.prop.landscape.core 42.mean.patch.core.area
## AirplaneLake                0.51379310                  201150
## BachelorMeadow              0.21058688                   32940
## BarkingFoxLake              0.10114943                    7920
## BirdbillLake                0.09403670                   14760
## BobLake                     0.09965636                   19575
## CacheLake                   0.56521739                  444600
##                 42.sd.patch.core.area 42.min.patch.core.area
## AirplaneLake                284469.06                     0
## BachelorMeadow               73656.08                     0
## BarkingFoxLake               21748.66                     0
## BirdbillLake                 33004.36                     0
## BobLake                      11766.16                  5400
## CacheLake                         NA                444600
##                 42.max.patch.core.area 42.prop.like.adjacencies
```

```
## AirplaneLake                   402300                0.8153242
## BachelorMeadow                 164700                0.7278402
## BarkingFoxLake                  69300                0.5961071
## BirdbillLake                    73800                0.6094675
## BobLake                         33300                0.6048485
## CacheLake                      444600                0.8296296
##                42.aggregation.index 42.lanscape.division.index
## AirplaneLake               93.39835                  0.3853283
## BachelorMeadow             89.14373                  0.8598656
## BarkingFoxLake             79.15994                  0.9246453
## BirdbillLake               80.62622                  0.9289543
## BobLake                    79.84000                  0.9617637
## CacheLake                  94.18360                  0.2811909
##                42.splitting.index 42.effective.mesh.size
## AirplaneLake             1.626885              481287.93
## BachelorMeadow           7.136009              109599.08
## BarkingFoxLake          13.270566               59002.76
## BirdbillLake            14.075451               55756.65
## BobLake                 26.153152               30042.27
## CacheLake                1.391190              565415.22
##                42.patch.cohesion.index
## AirplaneLake                  9.287891
## BachelorMeadow                9.110058
## BarkingFoxLake                8.906521
## BirdbillLake                  8.973038
## BobLake                       8.630749
## CacheLake                     9.306812
```

e) Append to site data set

```
Sites.sp@data <- data.frame(Sites.sp@data, Prop.landscape,
                            Forest.class)
```

Done!

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.