Land Cover Analysis in R

Helene Wagner, University of Toronto

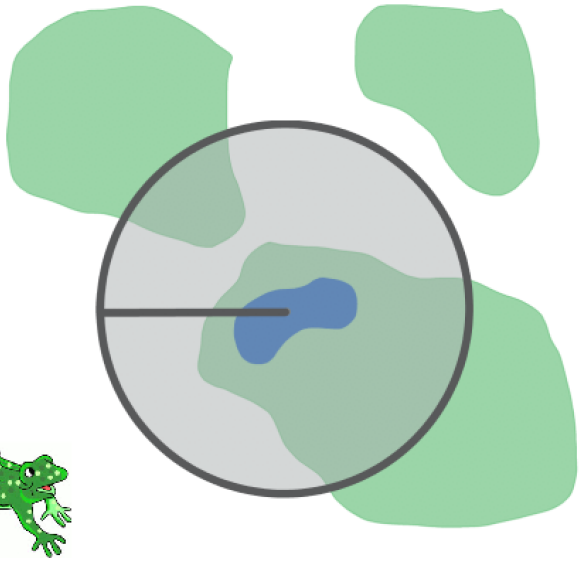
Task:

1. Find forest patch size at pond.
2. Determine percent forest cover within a 500 m radius.

R Goals:

1. Learn how spatial data are handled in R.
2. Programming with 'for' loops.

- What are land cover maps?
- Raster data manipulation (package 'raster')
- Site data as spatial features (package 'sp')
- Where on Earth is this? (define projection)
- Patch-level landscape metrics (package 'SDMTools')
- Class-level metrics within a buffer
- Looping through sites
- Final thoughts: GIS data in R



Tutorial: Learn to tweak code.
Worked Example: Model code that you can adapt to other data.
Bonus Material: New package 'sf'; how to plot land use map with predefined colors and labels.

What are Land Cover Maps?

Remote Sensing: Landsat Data

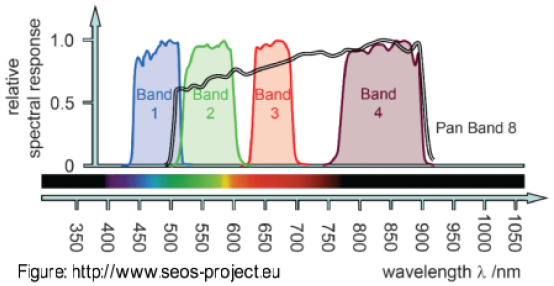
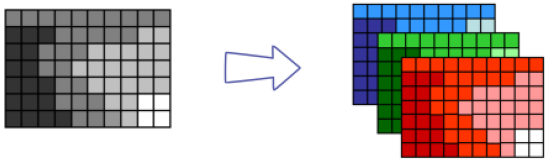


Figure: http://www.seos-project.eu

R Package 'raster'

Object class 'rasterLayer'

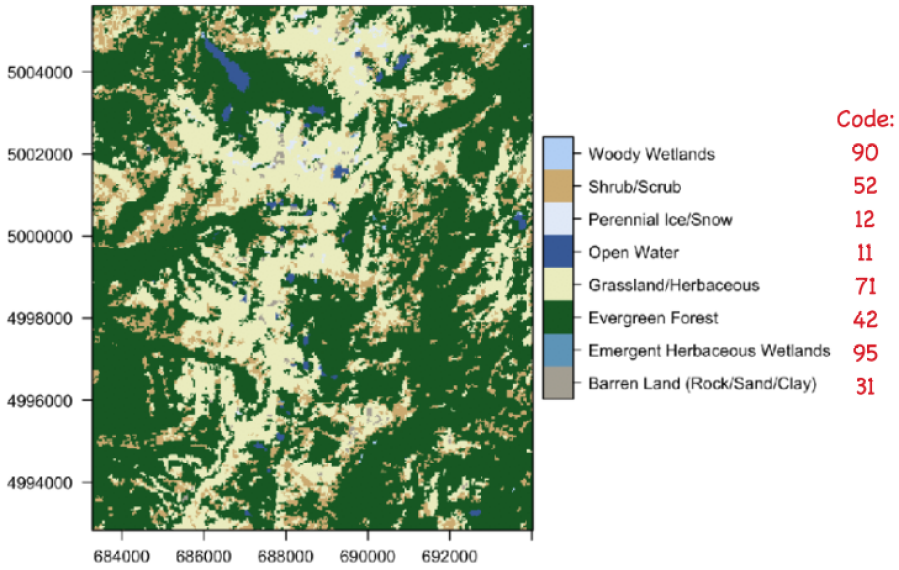
Raster stack



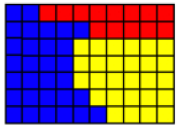
- Each raster contains one band (variable)
- Reflectance values 0 - 255
- RGB: Plot each band with different color
- Georeference, spatial extent, cell size (30 m)

Classification

National Land Cover Database (NLCD)



Categorical raster map



- Numbers indicate land cover type (factor levels)
- Raster attribute table (RAT) with labels and predefined colors
- For plotting, see Week 2 Bonus Materials

Raster Manipulation with Package 'raster'

```
raster <- raster::raster( raster )
```

Output object Package Function Input object

Friends don't let friends name their rasters 'raster'

Data type

- Can be 'numeric', 'integer' or 'logical'
- Spatially continuous variable (surface)

Can't handle 'character'. Numerical codes are interpreted as integers

RasterLayer object (S4) : NLCD

```
class       : RasterLayer
dimensions  : 426, 358, 152508  (nrow, ncol, ncell)
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 +towgs
84=0,0,0
data source : in memory
names       : nlcd
values      : 11, 95  (min, max)
```

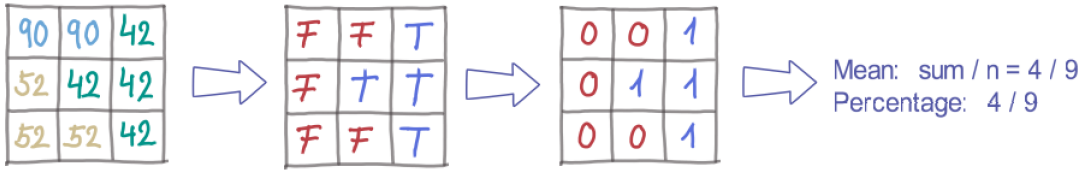
Raster algebra

```
NLCD * 5 - 17
cellStats(NLCD, stat='mean')
mean(values(NLCD))
table(values(NLCD))
```

Mean of cover type is nonsense!

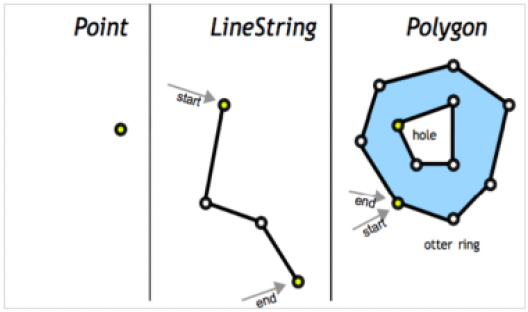
Raster logic

```
Forest <- ( NLCD == 42 )
PercentForest <- mean(values(Forest))
```



Site Data as Spatial Features in 'sp'

Spatial Features = Coordinates + Attribute table



http://geotools.org/

	coords.x1 <dbl>	coords.x2 <dbl>	SiteName <fctr>	Drainage <fctr>	Basin <fctr>	Substrate <fctr>
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

6 rows | 1-7 of 19 columns

Creating 'sp' objects

```
coordinates(myDataFrame) <-
  ~ xcoord + ycoord
```

- SpatialPointsDataFrame
- SpatialLinesDataFrame
- SpatialPolygonsDataFrame
- SpatialPixelsDataFrame
- SpatialGridDataFrame

Joining data tables

```
merge( x = Frogs, y = Sites, by.x = "Pop", by.y = "SiteName" )
```

Frogs

FrogID	Pop	LocusA
1	BobLake	1:1
2	BobLake	1:2
3	CacheLake	2:2
4	CacheLake	1:3

+

Sites

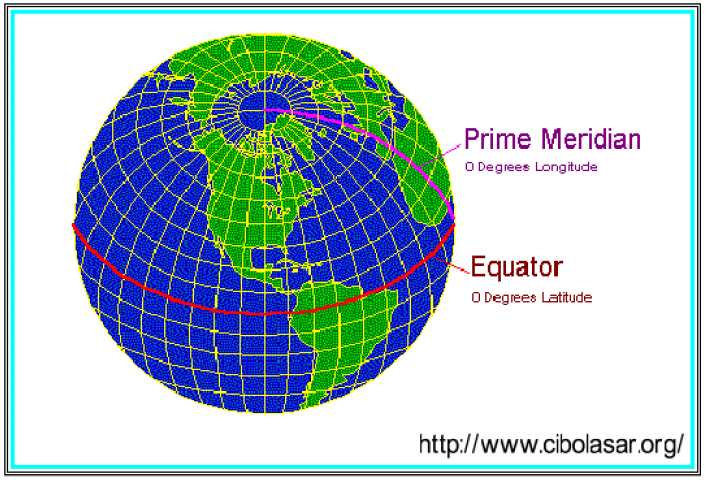
SiteName	Depth
AirplaneLake	21.64
BirdbillLake	3.93
BobLake	2.00
CacheLake	1.86

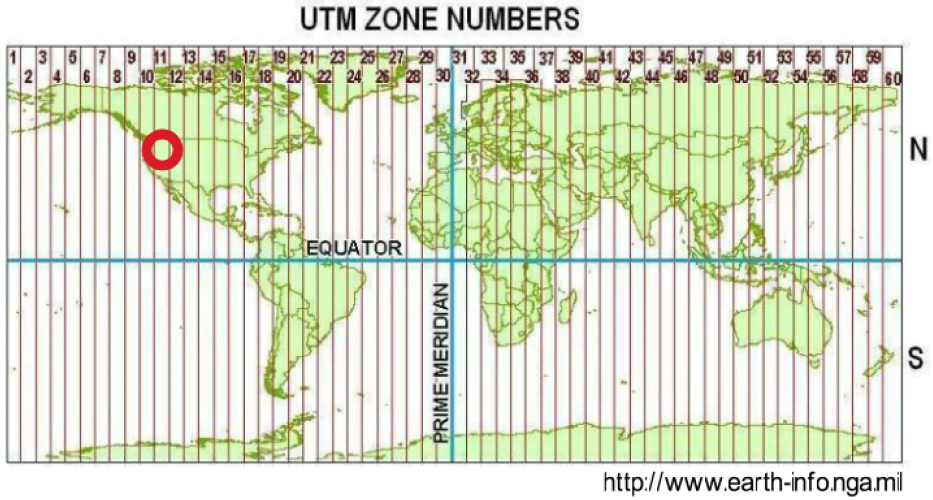
=

CombinedTable

FrogID	Pop	LocusA	Depth
1	BobLake	1:1	2.00
2	BobLake	1:2	2.00
3	CacheLake	2:2	1.86
4	CacheLake	1:3	1.86

Where on Earth? Define Projection





Geographic Coordinates (Lat-Long)

- GPS data often in longitude - latitude format
- Measured in degrees
- How far apart 1 degree is depends on latitude

```
tmaptools::get_proj4("longlat")
```

```
"+proj=longlat +ellps=WGS84
 +datum=WGS84 +no_defs +towgs84=0,0,0"
```

Universal Transverse Mercator (UTM)

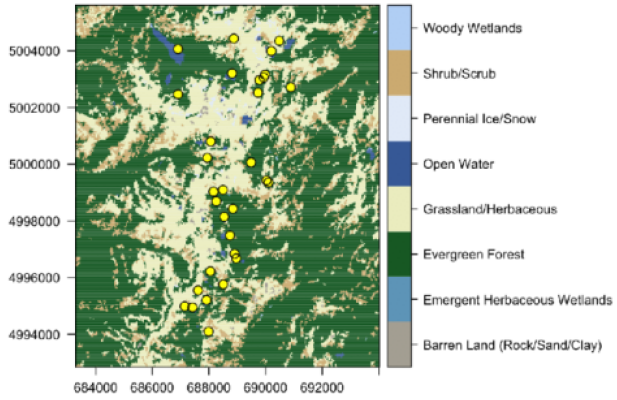
- Zone depends on longitude and hemisphere
- UTM coordinates in meters
- Distance easy to calculate within zone

```
tmaptools::get_proj4("utm11")
```

```
"+proj=utm +zone=11 +ellps=WGS84
 +datum=WGS84 +units=m +no_defs +towgs84=0,0,0"
```

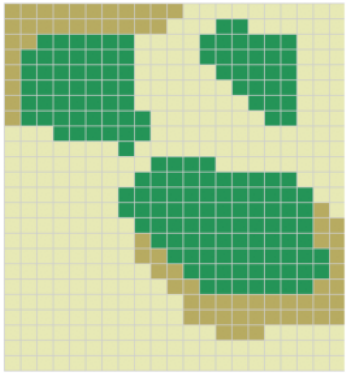
Patch-level Landscape Metrics

Overlay sites

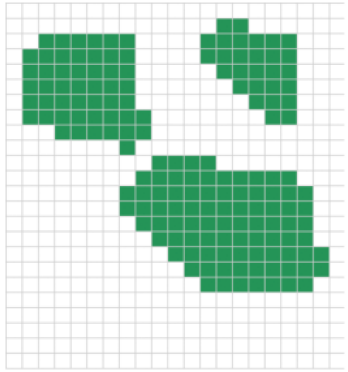


Task 1: Forest patch size?

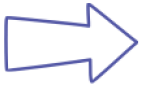
Land cover map



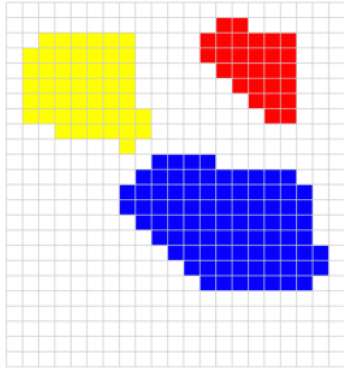
Forest map



Patch delineation



Map of forest patches



Patch-level metrics

Patch attribute table

patchID <int>	n.cell <int>	area <dbl>
0	62447	56202300
1	2	1800
2	35332	31798800
3	19	17100
4	39	35100
5	3	2700

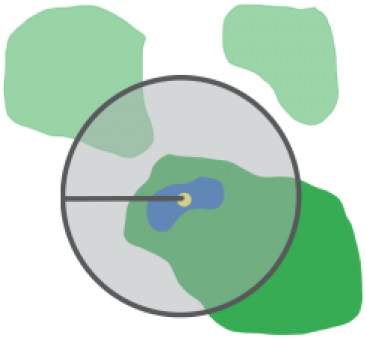
```
Forest <- (NLCD==42)
```

```
Patches <- ConnCompLabel(Forest)
```

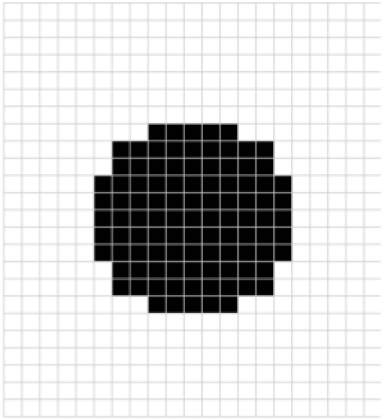
```
PatchStat(Patches, cellsize=30)
```

Class-level Metrics Within Buffer

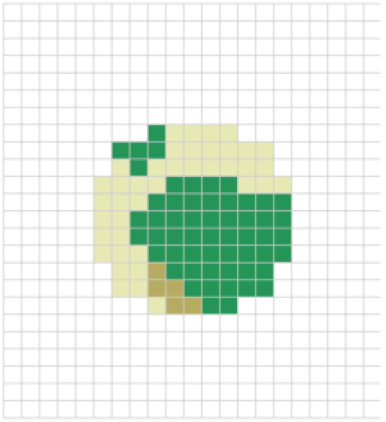
Task 2: Percent forest within 500 m?



Buffer around site



Land cover within buffer



Class attribute table

ID <dbl>	n.patches <int>	total.area <dbl>	prop.landscape <dbl>
11	1	33300	0.042189282
12	1	900	0.001140251
31	1	6300	0.007981756
42	4	315000	0.399087799
52	7	39600	0.050171038
71	4	388800	0.492588369
90	*NA*	*NA*	*NA*
95	1	5400	0.006841505

Looping Through Sites

```
Result <- list()
for(i in 1:n)
{
   Create buffer map for site i

   Create land cover map within buffer

   Calculate class-level metrics

   Write results into Result[[i]]
}
```

Empty list for results

Sets i to 1, 2, 3, ..., n=31

Writes results into i-th element of list

Final Thoughts: GIS Data in R

Exchange data as: shapefile geoTiff, ascii grid

Use object classes from R packages: 'sp', 'sf' 'raster'

See Bonus Material for 'sf' objects