02 - Scale

Cole LaCroix

2025-09-24

```r
#install packages if not already installed
if(!require(sf)) install.packages("sf")
```

```
## Loading required package: sf

## Linking to GEOS 3.11.0, GDAL 3.5.3, PROJ 9.1.0; sf_use_s2() is TRUE
```

```r
if(!require(fields)) install.packages("fields")
```

```
## Loading required package: fields

## Loading required package: spam

## Spam version 2.11-0 (2024-10-03) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.

##
## Attaching package: 'spam'

## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve

## Loading required package: viridisLite

##
## Try help(fields) to get started.
```

```r
if(!require(Matrix)) install.packages("Matrix")
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following object is masked from 'package:spam':
##
##     det
```

```r
if(!require(terra)) install.packages("terra")
```

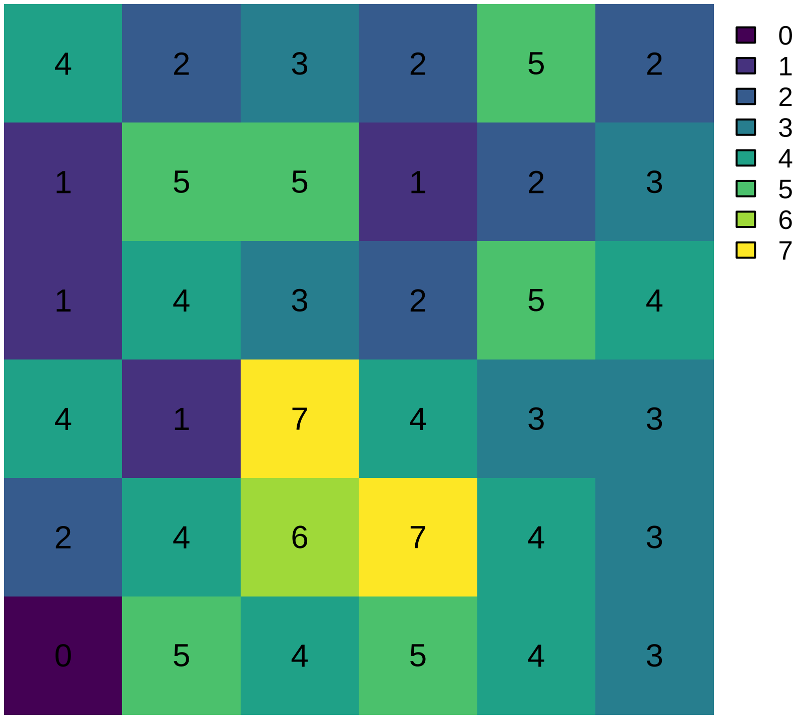
```
## Loading required package: terra

## terra 1.8.5

##
## Attaching package: 'terra'

## The following object is masked from 'package:fields':
##
##     describe
```

```r
if(!require(here)) install.packages("here")
```

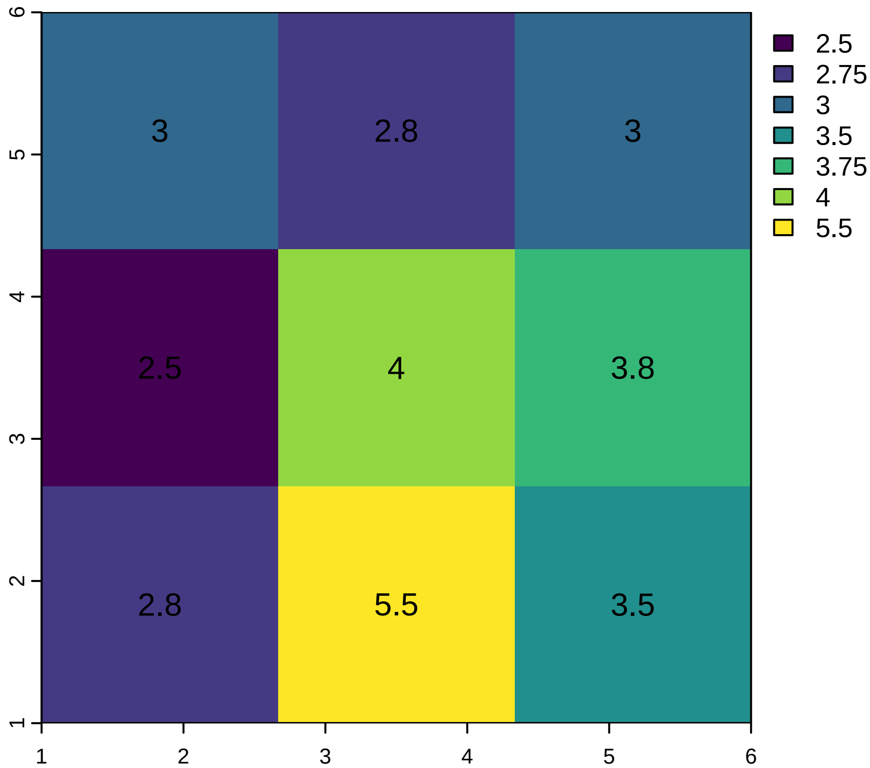
```
## Loading required package: here

## here() starts at /Users/GitHub Projects/landscape_ecology_wTongQiu
```

```r
#load required packages
library(sf)
library(fields)
library(Matrix)
library(terra)
library(here)
```

```r
# Ensure the random numbers are reproducible
set.seed(16)

# Create a 6x6 raster
toy <- rast(ncol = 6, nrow = 6, xmin = 1, xmax = 6, ymin = 1, ymax = 6)

# Fill the raster with random Poisson values (mean ~3)
toy[] <- rpois(ncell(toy), lambda=3)

# Visualize the raster and cell values
plot(toy, axes = FALSE, box = FALSE)
text(toy, digits = 2)
```




```
# Increase the grain (reduce resolution)
# Combine each 2x2 block of the original raster into one larger cell; Take the mean value of the four c
toy_mean <- aggregate(toy, fact = 2, fun = mean)

# Plot the mean-aggregated raster and display values
plot(toy_mean); text(toy_mean,digits=1)
```




```r
# Compare summary statistics
# Calculate mean and variance for the original raster
global(toy, mean);global(toy, var)
```

```
##           mean
## lyr.1 3.416667
```

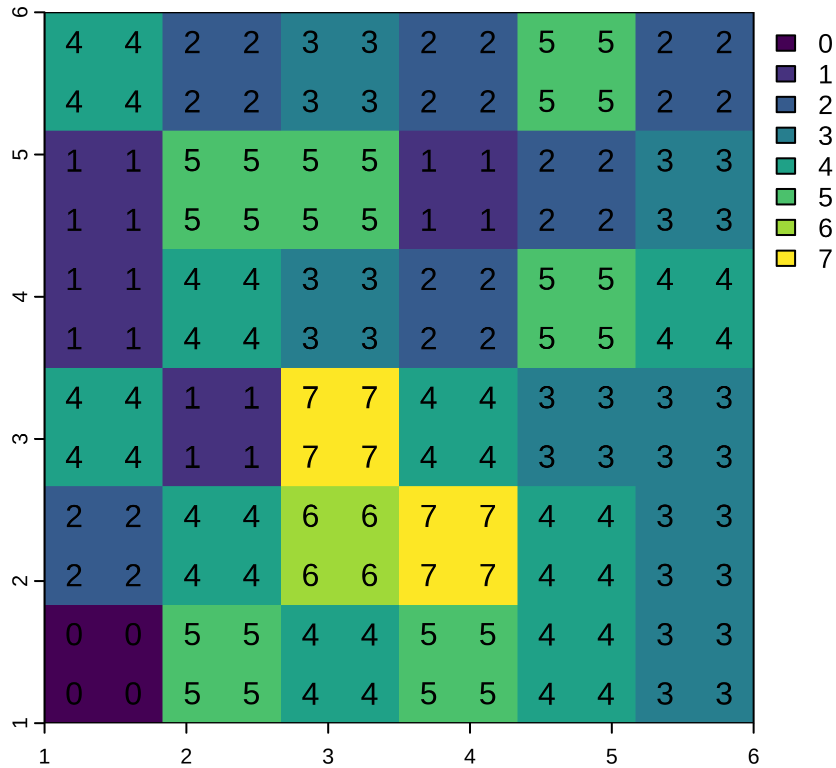
```
##          lyr.1
## lyr.1 2.821429
```

```r
# Calculate mean and variance for the mean-aggregated raster
global(toy_mean, mean); global(toy_mean, var)
```

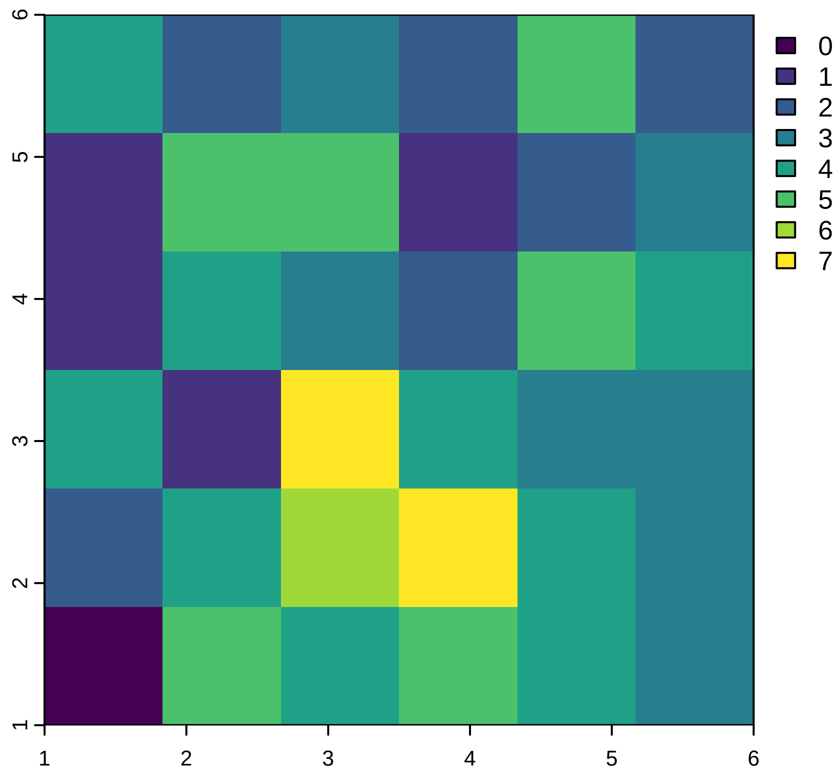
```
##           mean
## lyr.1 3.416667
```

```
##          lyr.1
## lyr.1 0.859375
```

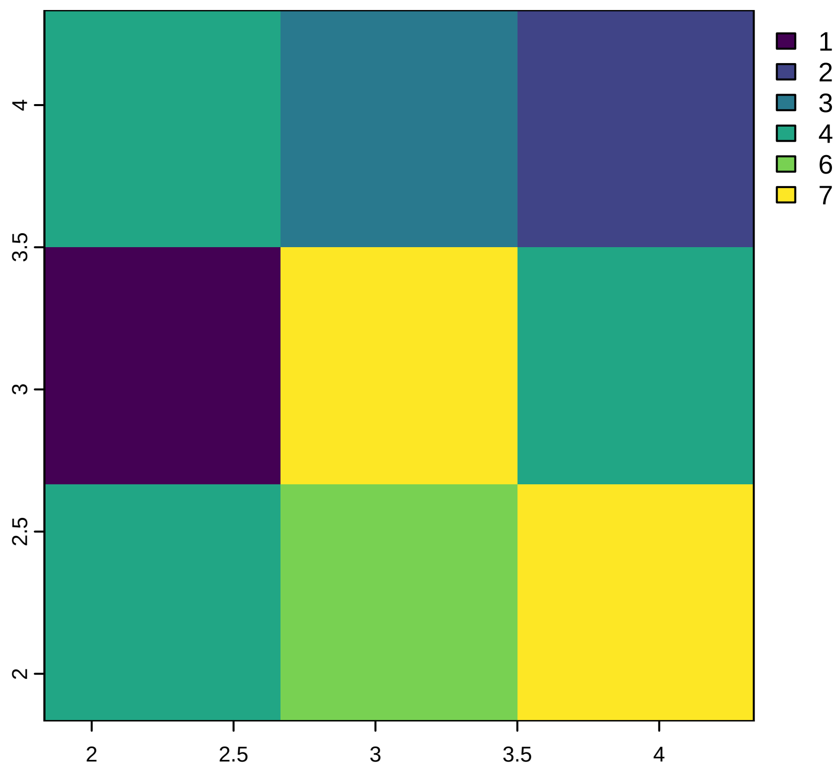
```r
# Decrease the grain (increase resolution)
# Split each cell into 2x2 smaller sub-cells (values simply replicated)
toy_dis2 <- disagg(toy, fact = 2)
plot(toy_dis2); text(toy_dis2,digits=1)
```



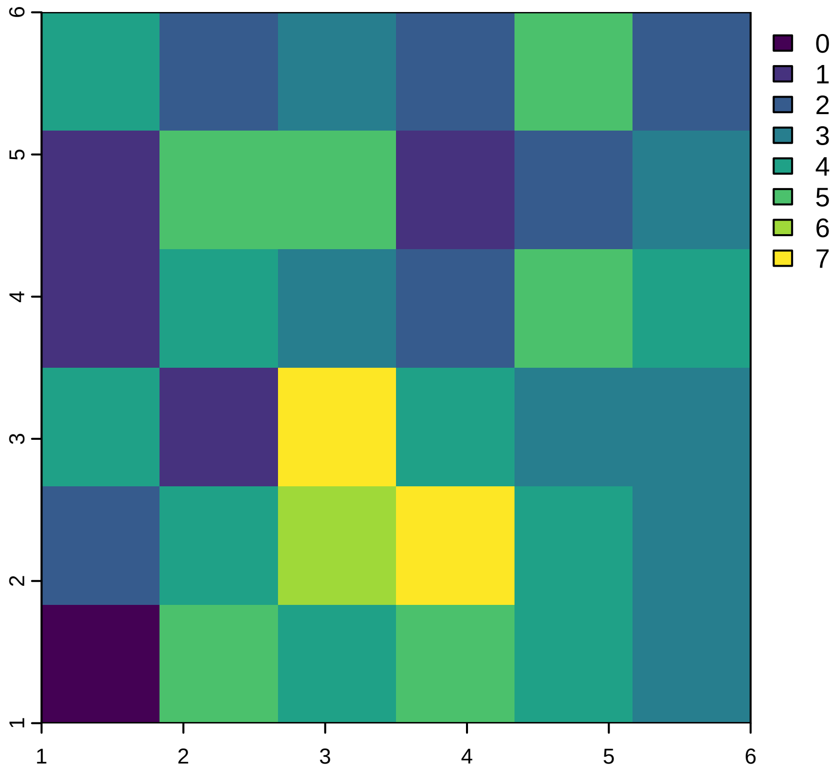

```r
# Crop to a smaller extent
e <- ext(2, 4, 2, 4) # Define a new extent (xmin=2, xmax=4, ymin=2, ymax=4)
toy_crop <- crop(toy, e) # Crop the raster to the smaller extent

plot(toy) # Plot the original raster
```



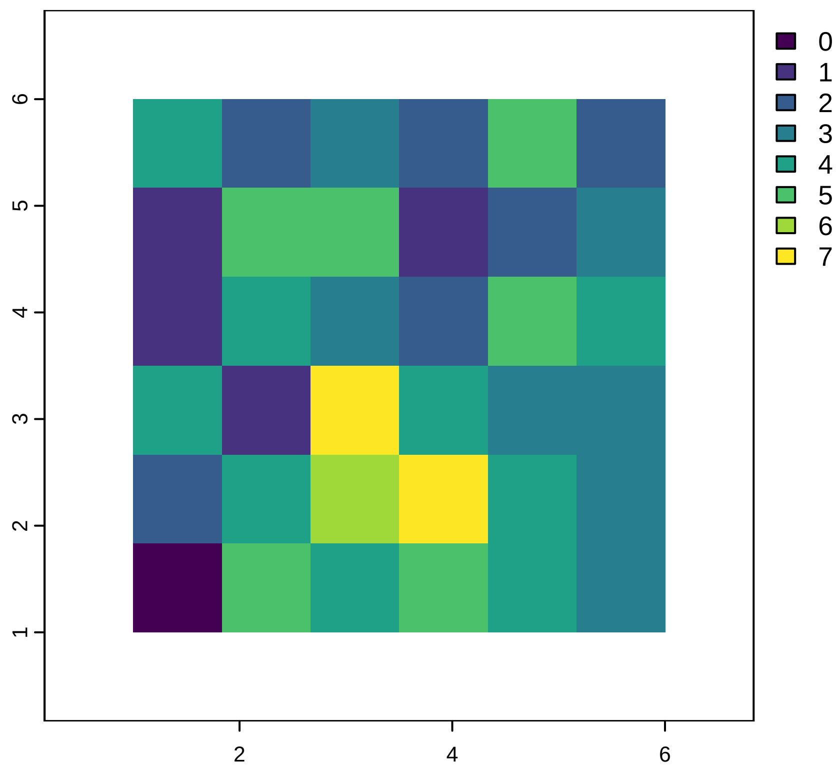

```r
plot(toy_crop) # Plot the cropped raster
```



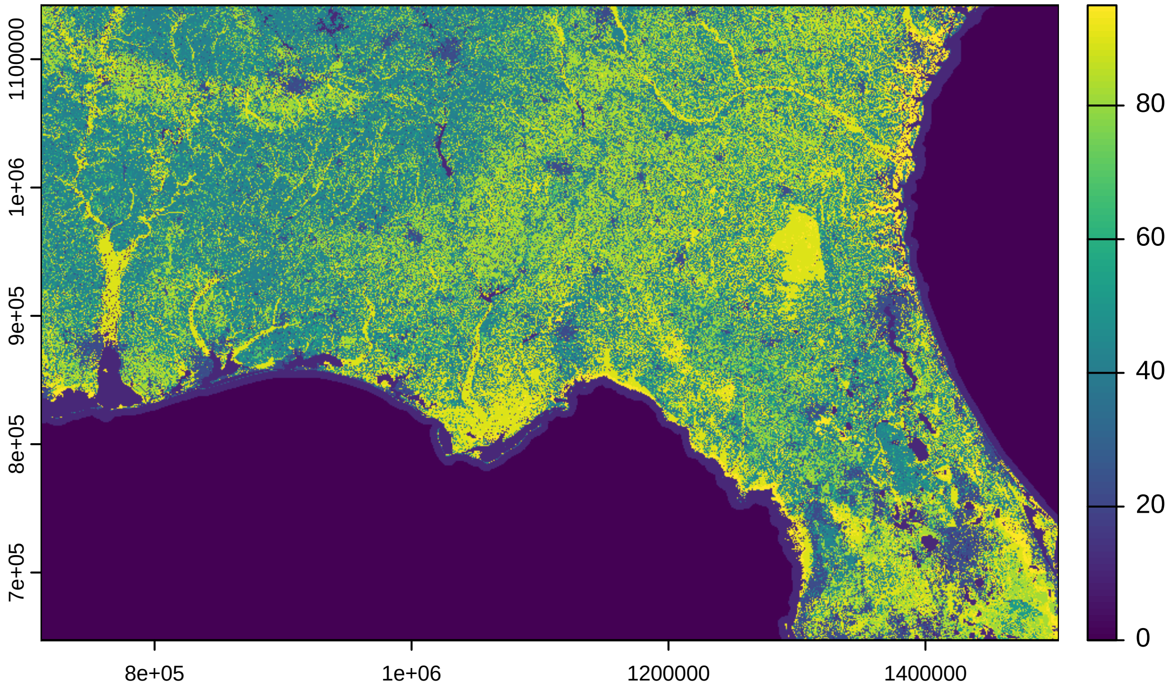

```r
# Extend to a larger extent
e <- ext(0, 7, 0, 7) # Define a new larger extent (xmin=0, xmax=7, ymin=0, ymax=7)
toy_big <- extend(toy,e) # Extend the raster to the larger extent (new cells filled with NA)

plot(toy)  # Plot the original raster
```




```r
plot(toy_big)  # Plot the extended raster
```




```
# Load the NLCD 2011 raster
nlcd <- rast("/Users/colelacroix/Documents/Lab1_data/nlcd2011SE")

# See the meaning of legend here https://www.mrlc.gov/data/legends/national-land-cover-database-2011-nl
plot(nlcd)
```




```r
# Check projection/CRS and basic properties: resolution, number of cells, and spatial extent
crs(nlcd); res(nlcd); ncell(nlcd); ext(nlcd)
```

```
## [1] "PROJCRS[\"unnamed\",\n    BASEGEOGCRS[\"NAD83\",\n        DATUM[\"North American Datum 1983\",\n
```

```
## [1] 30 30
```

```
## [1] 435629609
```

```
## SpatExtent : 711435, 1503825, 647235, 1142025 (xmin, xmax, ymin, ymax)
```

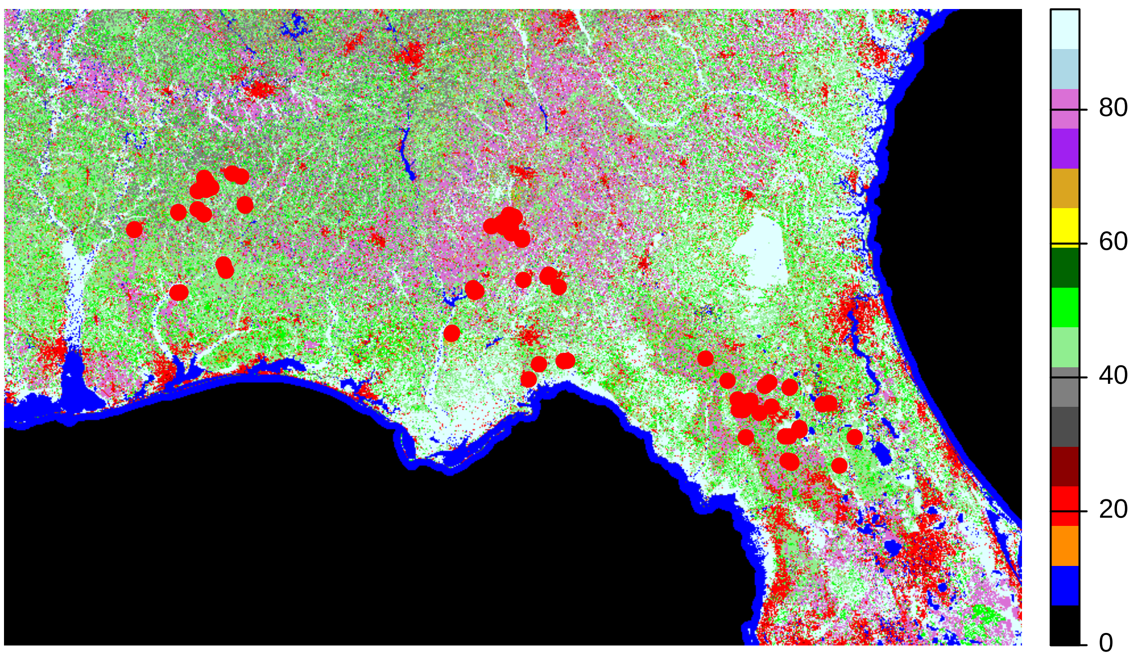
```r
# Load site shapefile
sites <- vect("/Users/colelacroix/Documents/Lab1_data/reptiledata")

# Assign the same CRS as the NLCD raster
crs(sites) <- crs(nlcd)

# Summarize attributes and previews the first two rows
summary(sites); head(sites, 2)
```

```
##      site             management          coords_x1         coords_x2
##  Length:85          Length:85          Min.   : 812599   Min.   : 786930
##  Class :character   Class :character   1st Qu.: 872612   1st Qu.: 838447
##  Mode  :character   Mode  :character   Median :1106233   Median : 933843
```

```
##                                         Mean   :1094814   Mean   : 918191
##                                         3rd Qu.:1288328   3rd Qu.: 982365
##                                         Max.   :1373597   Max.   :1014229
```

```
##   site                  management coords_x1 coords_x2
## 1  AL1                   Reference  846279.4  921444.9
## 2 AL10 Clear cut, residues removed  899063.5  989168.9
```

```r
# Plot with custom color scheme
my_col <- c("black","blue","darkorange","red","darkred","grey30","grey50", "lightgreen",
            "green", "darkgreen", "yellow", "goldenrod", "purple", "orchid","lightblue", "lightcyan")

plot(nlcd, col=my_col, axes=F, box=F)
plot(sites, col="red", add=T, pch=19)
```




```r
# Removes "Corn" management sites
# head(sites)
sites <- subset(sites, sites$management!="Corn")

# Crop raster to 10 km from sampling points: determine min/max coordinates for new extent
x.min <- min(sites$coords_x1) - 10000
x.max <- max(sites$coords_x1) + 10000
y.min <- min(sites$coords_x2) - 10000
y.max <- max(sites$coords_x2) + 10000
```

```r
# Defines a new extent based on site coordinates ±10 km
extent.new <- ext(x.min, x.max, y.min, y.max)

# Crops the raster to this smaller area of interest
nlcd <- crop(nlcd, extent.new)
```

```
## |---------|---------|---------|---------|=========================================
```

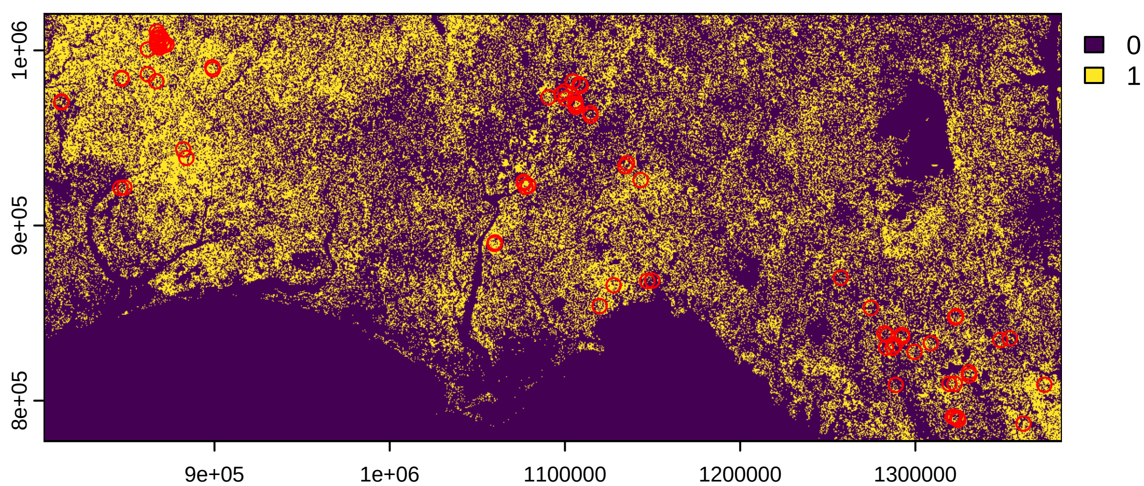
```r
# Create a binary forest layer using nlcd as template
forest <- nlcd

# Set to zero
values(forest) <- 0
```

```
## |---------|---------|---------|---------|=========================================
```

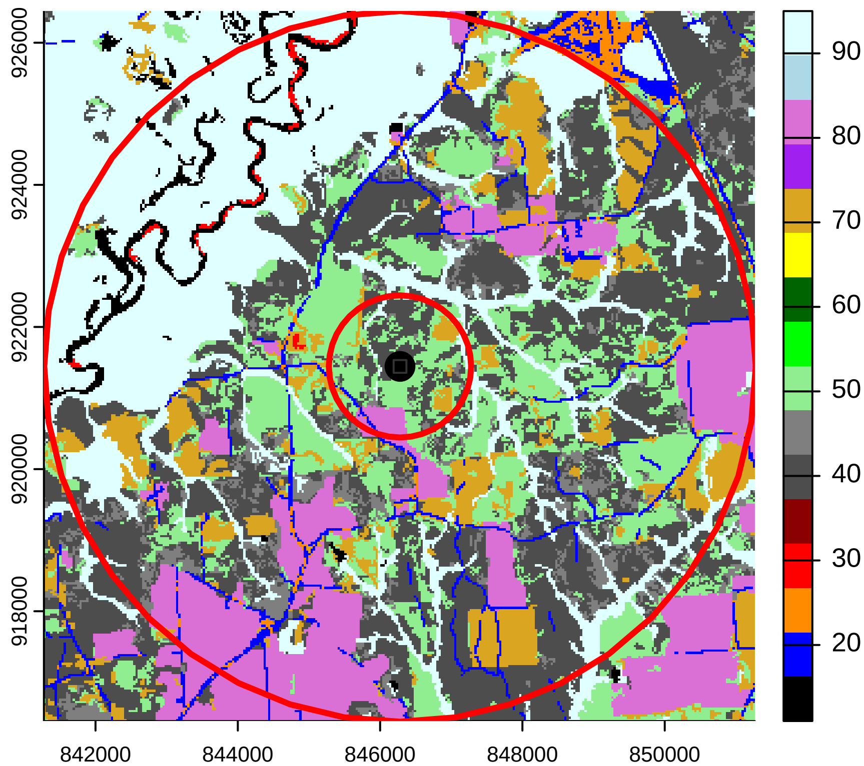
```r
# Reclassify
forest[nlcd==41 | nlcd==42 | nlcd==43] <- 1  # locations with deciduous + evergreen + mixed forest
```

```
## |---------|---------|---------|---------|=========================================
```

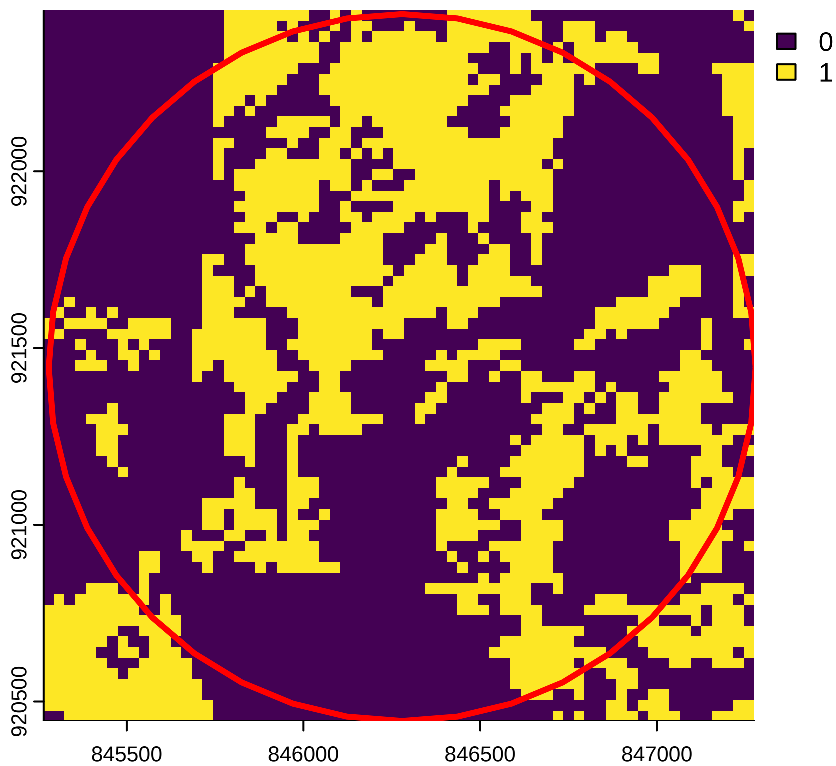
```r
# Plot
plot(forest)
plot(sites, pch=21, col="red", add=T)
```



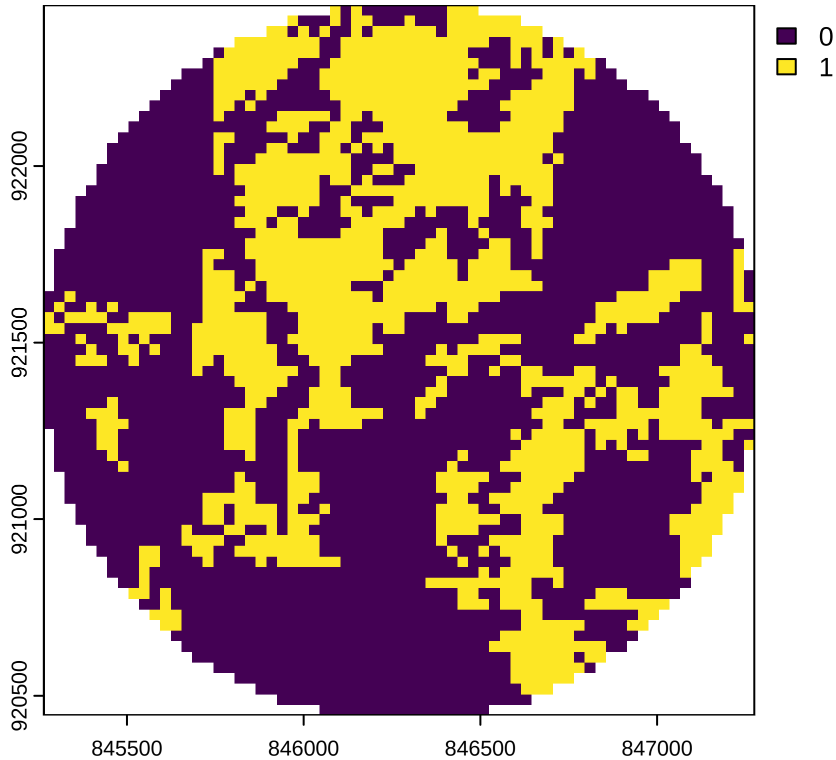
```r
buf1km <- 1000
buf5km <- 5000

# Try buffering first site
buffer.site1.1km <- buffer(sites[1,], width=buf1km)
buffer.site1.5km <- buffer(sites[1,], width=buf5km)

# Plot (remember to run the following codes at the same time otherwise will have errors)
zoom(nlcd, buffer.site1.5km, col=my_col, box=F)
plot(buffer.site1.1km, border="red", lwd = 3, add=T)
plot(buffer.site1.5km, border="red", lwd = 3, add=T)
points(sites[1,], pch=19, cex=2)
plot(sites[1,], col="grey20", bg="black", pch=22, cex=1, add=T)
```




```r
# View just forest within buffer
zoom(forest, buffer.site1.1km, box=F)
plot(buffer.site1.1km, border="red", lwd = 3,add=T)
```



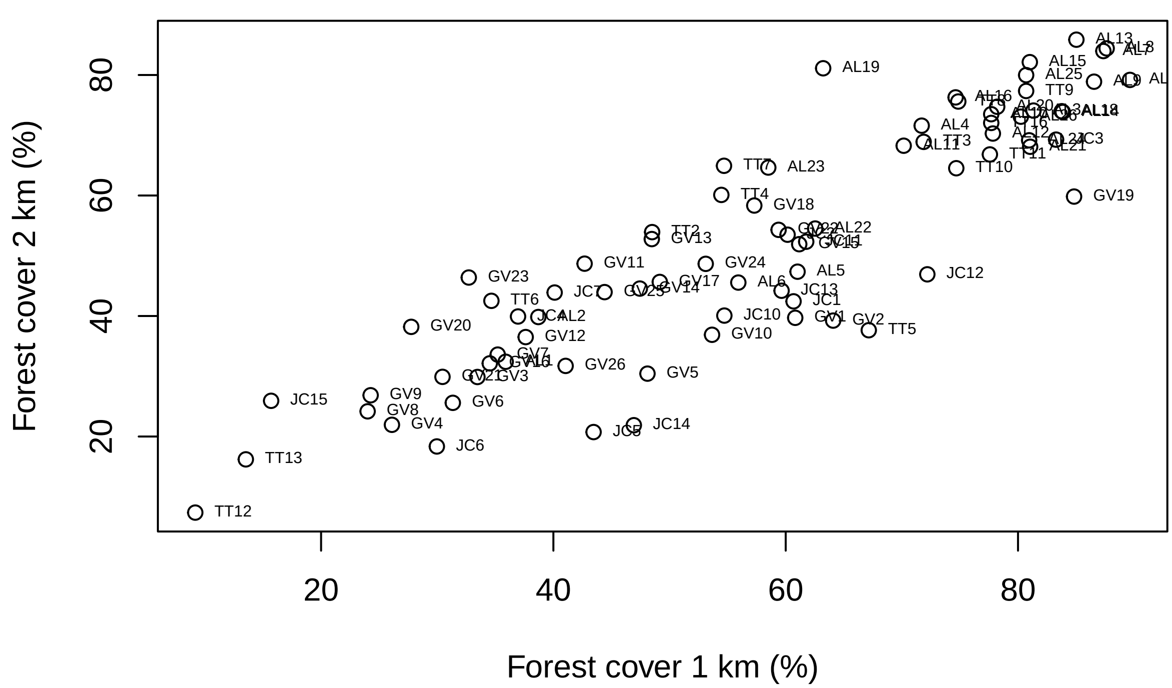

```r
# Calculate forest area within buffer
buffer.forest1.1km <- crop(forest, buffer.site1.1km)
buffer.forest1.1km <- mask(buffer.forest1.1km, buffer.site1.1km)

# Plot forest within buffer
plot(buffer.forest1.1km)
```




```
# Calculate percent forest cover
grainarea <- res(forest)[[1]]^2/10000 # pixel resolution ^2 and then convert the area into hectares
forestcover1km <- global(buffer.forest1.1km, 'sum', na.rm=TRUE)*grainarea
# forestcover1km

bufferarea <- (3.14159*buf1km^2)/10000 # pi*r^2
percentforest1km <- forestcover1km/bufferarea*100
# percentforest1km
```

```
# There are a total of 78 sites points
BufferCover <- function(coords, size, landcover, grain){

  bufferarea.i <- pi*size^2/10000

  buffer.i <- buffer(coords, width=size) # buffer
  crop.i <- crop(landcover, buffer.i) # crop with raster function
  crop.NA <- setValues(crop.i, NA) # empty raster for the rasterization
  buffer.r <- rasterize(buffer.i, crop.NA) # rasterize buffer
  land.buffer <- mask(x=crop.i, mask=buffer.r) # mask by putting NA outside the boundary

  coveramount<-as.numeric(global(land.buffer, 'sum', na.rm=TRUE)) * grain # calculate area
  percentcover<-100*(coveramount/bufferarea.i) # convert to %

  return(percentcover)
}
```

```r
# Create empty vector for storing output
f1km <- rep(NA, length = nrow(sites))
f2km <- rep(NA, length = nrow(sites))

# With for loop
for(i in 1:nrow(sites)) {
  f1km[i] <- BufferCover(coords=sites[i,],size=1000,landcover=forest,grain=grainarea)
  f2km[i] <- BufferCover(coords=sites[i,],size=2000,landcover=forest,grain=grainarea)
  #print(i)
}

# Make a data frame
forest.scale <- data.frame(
  site=sites$site,
  x=sites$coords_x1,
  y=sites$coords_x2,
  f1km=f1km, f2km=f2km)

# Plot
plot(f1km, f2km, xlab="Forest cover 1 km (%)", ylab="Forest cover 2 km (%)")
text(f1km, f2km, labels=sites$site, pos=4, cex=0.5, col="black")
```




```r
# Load reptile survey data
flsk <- read.csv("/Users/colelacroix/Documents/Lab1_data/reptiles_flsk.csv", header=T)
```

```r
# Merge reptile data with forest cover data by site ID
flsk <- merge(flsk, forest.scale, by="site", all=F)
print(flsk)
```

```
##    site pres         x         y      f1km      f2km
## 1   AL1    1  846279.4  921444.9 35.895806 32.407925
## 2  AL10    0  899063.5  989168.9 77.693077 73.496161
## 3  AL11    0  898755.1  990398.2 70.158682 68.260759
## 4  AL12    1  867689.7 1007135.9 77.836316 70.287598
## 5  AL13    1  868534.7 1001561.4 85.026937 85.857726
## 6  AL14    0  867288.2 1006417.8 83.852373 73.861422
## 7  AL15    1  871425.1 1004016.4 81.016232 82.133500
## 8  AL16    1  872611.8 1003293.9 74.627753 76.282168
## 9  AL17    1  869999.4 1005795.8 89.639247 79.182767
## 10 AL18    1  869257.0 1007598.6 83.766430 73.983175
## 11 AL19    1  868103.1 1009001.5 63.225893 81.116500
## 12  AL2    1  848545.2  921726.4 38.703299 39.834891
## 13 AL20    1  862003.2 1000559.3 78.208739 74.728020
## 14 AL21    0  867107.7 1010820.2 81.044880 68.088872
## 15 AL22    1  861935.7  986359.5 62.538343 54.516934
## 16 AL23    0  866875.7  982365.3 58.498991 64.636801
## 17 AL24    0  883971.8  938610.5 80.958936 69.163168
## 18 AL25    1  870981.5 1002773.4 80.701105 79.977746
## 19 AL26    1  882163.8  943493.1 80.242739 73.080767
## 20  AL3    0  846938.9  984261.9 81.331359 74.097767
## 21  AL4    0  847034.5  983809.7 71.705668 71.591077
## 22  AL5    0  812598.9  970310.4 61.020005 47.354962
## 23  AL6    0  812679.1  970837.7 55.920681 45.550145
## 24  AL7    1  867643.1 1002564.1 87.347416 83.974127
## 25  AL8    1  867614.2 1003607.1 87.633895 84.418169
## 26  AL9    1  867111.4 1004782.8 86.545275 78.889126
## 27  GV1    0 1323236.6  848200.8 60.819470 39.698813
## 28 GV10    0 1319390.1  809631.5 53.657498 36.884158
## 29 GV11    0 1322458.5  809631.8 42.685356 48.687089
## 30 GV12    0 1299536.0  827878.3 37.614679 36.504574
## 31 GV13    0 1308810.6  832869.1 48.472229 52.769413
## 32 GV14    0 1330762.0  814547.6 47.440905 44.554631
## 33 GV15    0 1330555.1  816132.4 61.163245 51.917138
## 34 GV16    0 1348466.4  834588.3 34.520707 32.142932
## 35 GV17    0 1353842.4  835575.0 49.159779 45.650412
## 36 GV18    0 1373596.6  809099.1 57.295780 58.334266
## 37 GV19    1 1361691.7  786930.5 84.826402 59.823956
## 38  GV2    0 1287060.6  830205.5 64.085329 39.240447
## 39 GV20    0 1324391.8  789096.5 27.759805 38.201961
## 40 GV21    0 1323975.5  790246.7 30.452707 29.901235
## 41 GV22    0 1321316.8  790877.2 59.387075 54.302075
## 42 GV23    0 1322367.0  790540.1 32.715890 46.388095
## 43 GV24    1 1322788.4  847495.5 53.113188 48.658441
## 44 GV25    0 1292529.2  836427.9 44.404229 43.967349
## 45 GV26    0 1292384.1  837582.1 41.052426 31.720376
## 46  GV3    0 1288327.9  831854.3 33.460735 29.879749
## 47  GV4    0 1288792.4  809018.7 26.098228 21.944284
## 48  GV5    0 1283301.0  830369.0 48.099807 30.438383
```

```
## 49  GV6    0 1257420.4  870211.9 31.340791 25.589728
## 50  GV7    0 1274614.2  853001.2 35.208257 33.596813
## 51  GV8    0 1282447.6  838446.6 24.006932 24.178819
## 52  GV9    0 1283806.6  837401.5 24.264763 26.850235
## 53  JC1    0 1099994.8  976568.6 60.676231 42.434687
## 54 JC10    0 1105881.7  971058.0 54.717469 40.071236
## 55 JC11    0 1106060.4  969561.8 61.764850 52.332533
## 56 JC12    0 1106175.7  968589.7 72.192682 46.918081
## 57 JC13    0 1106233.2  967689.9 59.644906 44.196532
## 58 JC14    0 1114532.3  964278.0 46.925243 21.879826
## 59 JC15    0 1114616.6  962744.1 15.699044 25.926340
## 60  JC2    0 1098065.6  975157.4 60.160568 53.492772
## 61  JC3    0 1090584.7  973251.4 83.279416 69.270597
## 62  JC4    0 1104486.0  982225.0 36.955778 39.920834
## 63  JC5    0 1108963.9  979827.9 43.458849 20.741072
## 64  JC6    0 1108428.6  980655.2 29.965693 18.348973
## 65  JC7    0 1100436.1  972504.3 40.107046 43.895729
## 66 TT10    0 1060079.6  890349.1 74.685049 64.529372
## 67 TT11    0 1119769.6  854023.3 77.578485 66.814041
## 68 TT12    0 1149792.4  868537.5  9.167325  7.376832
## 69 TT13    0 1146985.6  868351.8 13.521804 16.207544
## 70 TT16    0 1143193.1  925963.7 77.693077 72.042281
## 71  TT2    0 1076464.8  925051.8 48.500877 53.922490
## 72  TT3    1 1076574.2  924241.4 71.877555 68.898175
## 73  TT4    0 1079111.5  922758.9 54.459638 60.096111
## 74  TT5    0 1135394.3  935550.4 67.150654 37.643327
## 75  TT6    0 1134460.4  933842.9 34.663947 42.520630
## 76  TT7    0 1077529.3  922087.8 54.688822 64.930442
## 77  TT8    0 1127784.4  865897.2 74.856936 75.594619
## 78  TT9    0 1059748.6  889500.6 80.701105 77.349302
```

```r
# glms across scales
# Formula: response variable pres (presence/absence) is explained by predictor f1km or f2km.
# Tells R that the response variable is binary (0/1, yes/no, presence/absence).
pres.1km <- glm(pres ~ f1km, family = "binomial", data = flsk)
pres.2km <- glm(pres ~ f2km, family = "binomial", data = flsk)

# Summary information
summary(pres.1km); summary(pres.2km)
```

```
##
## Call:
## glm(formula = pres ~ f1km, family = "binomial", data = flsk)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.75912    1.45699  -3.953 7.73e-05 ***
## f1km         0.07189    0.02041   3.522 0.000428 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 86.608  on 77  degrees of freedom
```

```
## Residual deviance: 67.621  on 76  degrees of freedom
## AIC: 71.621
##
## Number of Fisher Scoring iterations: 5
```

```
##
## Call:
## glm(formula = pres ~ f2km, family = "binomial", data = flsk)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.16431    1.46539  -4.207 2.59e-05 ***
## f2km         0.08461    0.02198   3.850 0.000118 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 86.608  on 77  degrees of freedom
## Residual deviance: 62.724  on 76  degrees of freedom
## AIC: 66.724
##
## Number of Fisher Scoring iterations: 5
```

```r
# Likelihoods
logLik(pres.1km); logLik(pres.2km)
```

```
## 'log Lik.' -33.81059 (df=2)
```

```
## 'log Lik.' -31.3621 (df=2)
```

```r
# Coefficients
pres.1km.ci <- confint(pres.1km)
```

```
## Waiting for profiling to be done...
```

```r
pres.2km.ci <- confint(pres.2km)
```

```
## Waiting for profiling to be done...
```

```r
pres.1km.ci; pres.2km.ci
```

```
##                   2.5 %     97.5 %
## (Intercept) -9.06711914 -3.2690747
## f1km         0.03600982  0.1171198
```

```
##                   2.5 %     97.5 %
## (Intercept) -9.52656595 -3.6706095
## f2km         0.04614732  0.1338294
```

Exercise. Q1. What does this code create (please choose one option below)?

When we run: toy <- rast(ncol = 6, nrow = 6, xmin = 1, xmax = 6, ymin = 1, ymax = 6)

Answer - b: A raster with 6 rows × 6 columns, spanning coordinates from 1 to 6 in both x and y directions

Exercise. Q2. Estimates of forest cover can vary depending on the buffer size (scale) used. In your opinion, under what circumstances would a small-scale measure of forest cover better explain species distributions, and when would a larger-scale measure be more appropriate? Discuss your reasoning in relation to the concept of scale dependence in ecology.

Different buffer scales are appropriate in different circumstances. A very large buffer might be appropriate when evaluating the effect of a nationwide policy regarding general forest preservation. A smaller buffer might be appropriate when evaluating the effect of that same policy on a particular forest community type. A very small buffer might be appropriate when evaluating the extent of a species occupying a particular niche within a forest community type. Or, when evaluating the impact of microclimates on forest growth.

Exercise. Q3. Two regression models relating prey presence to forest cover within 1 km and 2 km buffers. Interpret the coefficients and significance of each model. Which model provides a better fit to the data, and why?

Both models show positive correlations between forest cover and reptile presence. The likelihood of reptile presence increases at both scales. However, the 2km scale has a steeper slope (1km = .072, 2km = .085) which indicates better responsiveness to the variable. Additionally, the 'log lik' of 1km is -33.81059 (df=2) and 2km 'log Lik.' is -31.3621 (df=2) which indicates that 2km is a better fit because the value is higher.