Landscape Composition Analysis in R

Danielle Clake - R Wizardry Course (Winter 2017)

12 April 2017

Contents

Background

Landscape ecologists and landscape geneticists often require an efficient way to quantify a landscape of interest. While mapping software such as ESRI's ArcGIS can be useful for viewing and manipulating spatial data and creating basic summaries, it does not allow for complex statistical analysis. The open source R software allows users both to do analysis of landscape data, and to use that data directly in further statistical analysis. This can be very useful, as it saves users from having to convert data between programs, and can allow a larger proportion of analyses to be automated.

This tutorial goes through an example of how to load and format spatial data, and gives one way to quantify the landscape composition within a certain radius (or set of radii) of given sample locations. It is meant for users with a baseline understanding of both the R program and spatial data. The tutorial begins with a brief overview on how to import and view spatial data in R, but for a more in-depth introduction users should refer to other sources such as the "Introduction to visualising spatial data in R" by Lovelace et al. (2017).

After the introduction to spatial data in R, the technique described in this tutorial begins by converting land cover data from "vector" format (a set of spatially referenced points, lines, or polygons that are each assigned one or more attributes) to "raster" format (a continuous grid of pixels or "cells" which each have an assigned value - think of a digital image as an example, where the assigned value would be a specific colour for each pixel). Both formats have benefits and drawbacks when used in spatial analysis. For large, continuous datasets such as landscapes, raster data is often faster and more efficient to analyze. Therefore this tutorial will focus on techniques that use raster data.

Once landscape data is in raster format, the tutorial gives an example of how to modify the data classes if required (in this case the land cover classification is simplified in order to be more ecologically relevant). Then, users will be given an example of how to create "buffers", or polygons that incorporate a set distance around a starting location or feature. After buffers are created with the desired distance around the locations of interest, the land cover within each buffer is extracted and summarized as an R "list", which essentially lists the value of each pixel falling within a given buffer. The tutorial then takes the user through steps to find the frequency of each value (or land cover class) within each buffer, before turning that into a proportion and organising it into a table that can be more easily queried.

The tutorial lastly provides code for a function which can summarize the land cover within many different buffer distances in one step. This is useful because often the most relevant spatial scale of analysis is not known *a priori*. Therefore, users can evaluate the landscape composition within several different distances at once.

The summary data created using this technique can be used in many different applications of landscape ecology and landscape genetics. The land cover composition surrounding each sampling location in a study can be correlated with a variety of different demographic characteristics of the individuals sampled at that location. For example, previous studies have compared landscape composition to nest density (Jha & Kremen 2013, Knight et al. 2009), foraging distance (Redhead et al. 2016) and lineage persistence (Carvell et al. 2017) in bumble bees.

While this tutorial uses land cover data, it should also be noted that the same techniques and functions used here can be applied to other raster datasets containing discrete categories, or modified for application to continuous datasets. For example, it could be used to determine the proportion of different elevation classes or soil regimes within the vicinity of points. It could also be scaled up or down in spatial extent depending on the research questions and organisms or mechanisms being studied.

Preparing Workspace

Load Packages

This tutorial will use a number of existing R packages. The code below will streamline the installation and loading of these required packages. If packages are already installed and/or loaded users should skip the code below. Prompts to load packages will also be provided in text before the first time each package is used.

```
x <- c("rgdal", "dplyr", "tmap", "raster", "rgeos", "tidyr", "ggplot2")
```

Uncomment the code below if you need to install packages:

```
#install.packages(x)
```

Uncomment the code below to load the required packages (or load in text):

```
#lapply(x, library, character.only = TRUE)
```

Update working directory

Prior to this tutorial, users should also set the default workspace to the folder where the tutorial data is stored. Update the file path in the code below accordingly.

```r
setwd("C:/User/Documents/University of Calgary/Courses/BIOL607_RWizardry/Final Project")
```

Part I - Displaying Map Data

The first part of this tutorial will use three spatial datasets:

1. The boundary of Banff National Park (located in Alberta, Canada) obtained from AltaLIS (http://www.altalis.com/)
2. Land cover data from the Alberta Biodiversity Monitoring Institute (ABMI - http://www.abmi.ca) clipped to the area of Banff National Park
3. Sampling locations spaced throughout the park.

Load data

We will be using the **rgal** package to load our spatial data in R. This package has a function *readOGR* which requires two inputs: dsn (the "data source name") and the name of the layer. Since we have already set our default workspace, we just need to specify which folder the data is in ("shapefiles"), and the name of our data ("BanffNP_UTM12" etc.). Note that for this function we do not need to include a file extension after the data name.

```r
library(rgdal)
BNP <- readOGR(dsn = "Data/BanffNP/shapefiles", layer = "BanffNP_UTM12")
LC <- readOGR(dsn = "Data/BanffNP/shapefiles", layer = "Landcover_ABMI_2010_UTM12")
SL <- readOGR(dsn = "Data/BanffNP/shapefiles", layer = "SamplingLocations_UTM12")
```

View and modify data

Spatial data in R has several different components, including:

- The visual points, lines, or polygons (hereafter referred to as "features")
- The data associated with each feature, organised in what is known as an "attribute table"

We can view the features using the "plot" tool in base R:

```r
#view the Banff National Park outline:
plot(BNP)

#Add the sampling locations:
plot(SL, add = TRUE) #(use code "add = TRUE" to add points to previous plot)
```

The line below can be uncommented to view land cover outlines as well, however it is a large file and will take several minutes to draw:

```
#plot(LC, add = TRUE)
```

The attribute table for each layer can be accessed by using the code "@data". The land cover layer has over 6000 features, so we will use the *head* function to view data for only the first 6 features.

```
head(LC@data)
```

```
##   OBJECTID LC_class MOD_TY Shape_Leng Shape_Le_1 Shape_Area
## 0        1       20   <NA>   434.5297   434.5297   11537.27
## 1        2       20   <NA>   535.2053   535.2053   19079.17
## 2        3       20   <NA>  3110.8275  3110.8275  514542.01
## 3        4       20   <NA>   875.0112   875.0112   52894.15
## 4        5       20   <NA>   827.7478   827.7478   35511.53
## 5        6       20   <NA>  1000.3386  1000.3386   60685.22
```

As we can see, the land cover data from ABMI comes with a simplified numeric "LC_class". These numbers aren't meaningful to us in their current form, but luckily we have access to a lookup table that relates each numeric code to a land cover description. In the code below we will first open this data, then use the *left_join* function in the **dplyr** package to attach the descriptions to our existing file.

```
library(dplyr) #load the dplyr package

#Load the look up table:
LC_LUT <- read.csv("Data/BanffNP/Shapefiles/ABMI_LUT.csv", stringsAsFactors = FALSE)
head(LC_LUT, 3) #view the first three lines of the look up table
```

```
##   Code        Name
## 1   20       Water
## 2   31    Snow/Ice
## 3   32 Rock/Rubble
```

```
#Attach descriptions, matching "LC_class" to "Code":
LC@data <- left_join(LC@data, LC_LUT, by = c("LC_class" = "Code"))
head(LC@data, 3) #check whether join worked
```

```
##   OBJECTID LC_class MOD_TY Shape_Leng Shape_Le_1 Shape_Area  Name
## 1        1       20   <NA>   434.5297   434.5297   11537.27 Water
## 2        2       20   <NA>   535.2053   535.2053   19079.17 Water
## 3        3       20   <NA>  3110.8275  3110.8275  514542.01 Water
```

Because our join has added the land cover description under a rather generic "Name" column, let's rename it:

```
colnames(LC@data)[colnames(LC@data) == "Name"] <- "LC_Desc"

#Check which land cover classes we have in our data:
unique(LC$LC_Desc) #We can use the "$" in place of "@data" here
```

```
##  [1] "Water"             "Snow/Ice"          "Rock/Rubble"
##  [4] "Exposed Land"      "Developed"         "Shrubland"
##  [7] "Grassland"         "Coniferous Forest" "Broadleaf Forest"
## [10] "Mixed Forest"
```

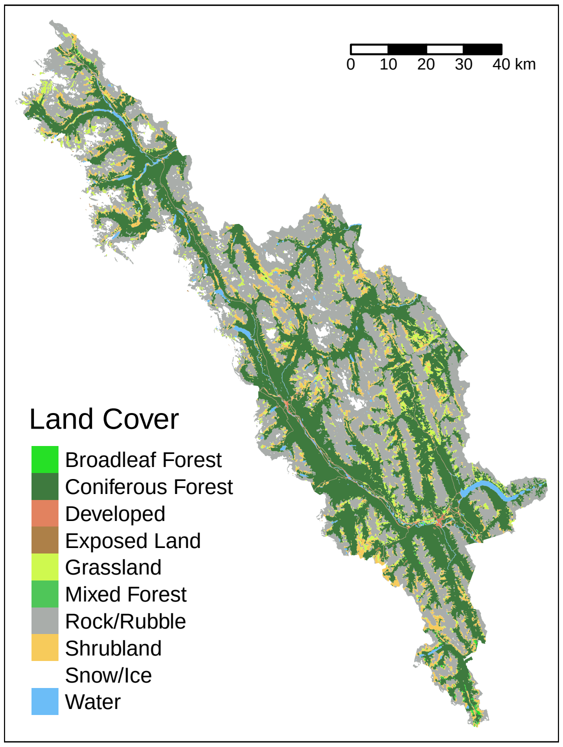
View map

We can use the **tmap** package to quickly visualize our land cover data.

```
library(tmap)
```

```
#Create palette for land cover types:
LC.palette <- c("#26e026", "#3e7a3e", "#e2825f", "#ad8048", "#cff94f",
                "#4fc659", "#a9adaa", "#f7cb5b", "#ffffff", "#6cbdf7")

#Create map of data:
tm_shape(LC) + #Specify shapefile to be mapped
  tm_fill("LC_Desc", palette = LC.palette,  title = "Land Cover") + #select fill data
  tm_scale_bar(position = c("right", "top")) #add a scale bar
```



Rasterize land cover data

So far we have been displaying our land cover data as *vector data* - a continuous mosaic of polygons. However, continuous data such as this is often better displayed and analyzed as a *raster* (see background for a description of raster and vector data). In the code below, we will use the **raster** package to create an empty "template" raster, then assign the land cover values from our polygons to the corresponding pixels in the raster.

```
library(raster)
```

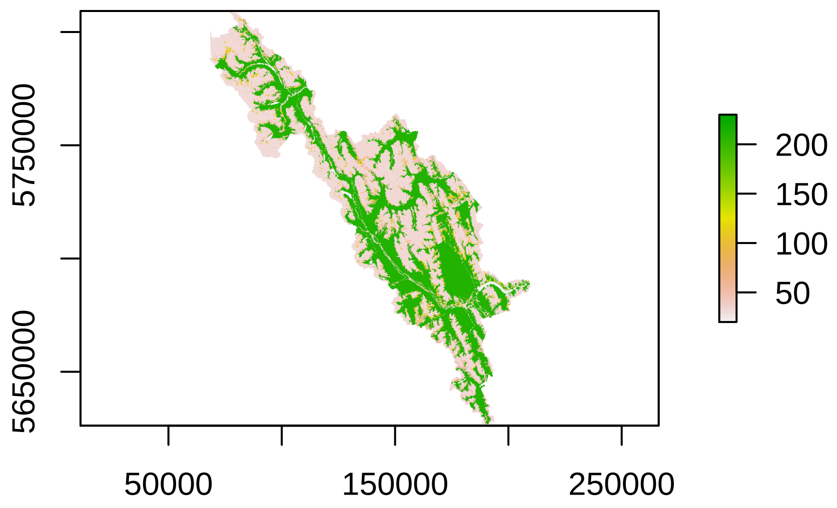
First, create a template raster with the spatial extent ("ext") and projection ("crs") set to be the same as our polygon landcover layer ("LC"). We are using a resolution, or pixel size, of 100 map units (in this case metres).

```
template.raster <- raster(ext = extent(LC), resolution = 100, crs = CRS(projection(LC)))
```

Next we essentially "stamp" our land cover polygons onto our template raster.

```
#Caution - this step will likely take several minutes to run.
LCrast <- rasterize(x = LC, y = template.raster, field = "LC_class", fun = 'last')

plot(LCrast) #lets see what our raster looks like!
```



Simplify land cover categories

Often the different land cover classes that have been created as relevant for humans are not relevant for the study species. For example, bumble bees may not experience rock/rubble any differently than exposed land. In order to simplify the data and any further analysis, we will simplify the land cover categories supplied in the ABMI data such that:

- 20 + 31 = Ice/Water (1)
- 32 + 33 = Exposed (3)

- 34 = Developed (2)
- 50 + 110 = Grass/Shrub (6)
- 120 = Agriculture (4)
- 210 + 220 + 230 = Forest (5)

(where the numbers in brackets represent the new numerical “class” that we will asign the data).

```
#First make a matrix to reclassify by:
m <- c(20, 1, 31, 1, 32, 3, 33, 3, 34, 2, 50, 6,
       110, 6, 120, 4, 210, 5, 220, 5, 230, 5)
lcrc <- matrix(m, ncol=2, byrow=TRUE)
lcrc
```

```
##       [,1] [,2]
##  [1,]   20    1
##  [2,]   31    1
##  [3,]   32    3
##  [4,]   33    3
##  [5,]   34    2
##  [6,]   50    6
##  [7,]  110    6
##  [8,]  120    4
##  [9,]  210    5
## [10,]  220    5
## [11,]  230    5
```

```
#in this matrix the first column represents the original class, and the second column
#represents the new class that we will assign to it.

#Then reclassify the raster:
LCrastRC <- reclassify(LCrast, lcrc)

#Make a new look up table for simplified land cover
LCS.Code <- 1:6
LCS.Desc <- c("Ice.Water", "Developed", "Exposed", "Agriculture",
              "Forest", "Grass.Shrub")
LCS_LUT <- as.data.frame(cbind(LCS.Code, LCS.Desc))
```

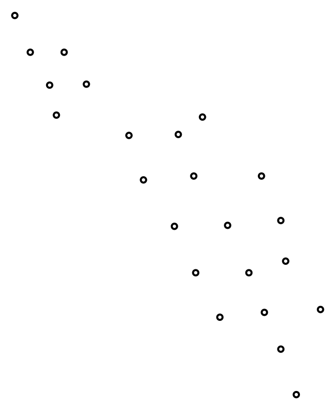
Part II - Spatial Analysis

Buffer sampling locations and extract values

First we need to create “buffers”, or circles with a set radius around our sampling locations. We will do this using the *gBuffer* tool in the **rgeos** package. We then can extract the landcover values within each buffer using the *extract* tool in the **raster** package.

```
library(rgeos)

#Create 1000 m buffers around sampling locations:
SL_buffer1km <- gBuffer(spgeom = SL, byid = TRUE, id = SL$id, width = 1000)
plot(SL_buffer1km) #look at the buffers created
```



```
#Extract number of pixels of each land cover class under each polygon:
LC1km <- raster::extract(x = LCrastRC, y = SL_buffer1km)
#(because there is also an "extract" function in the tidyr dataset, we use
#the "raster::extract" notation to specify that we want the raster function)

#Find frequency of each land class within each polygon:
LC1km.fq <- lapply(LC1km, table)

#Calculate proportion of land cover within each polygon:
LC1km.pr <- lapply(LC1km.fq, FUN = function(x){x/sum(x)})
```

Organize table for further analysis

We now have a "list" in R that gives us the proportion of each land cover within 1 km of each polygon, but in order to assess the results further we will need to change formats and clean the data.

First we create a function that will extract the data from each item in the list (corresponding to each polygon, or sample location):

```
l2d <- function(x, lcpoly){
  temp1 <- unlist(x[lcpoly])
  temp2 <- as.data.frame(temp1)
  names(temp2) <- paste("loc", lcpoly, sep="")
  LC.class <- rownames(temp2)
  final <- cbind(LC.class, temp2)
  print(final)
}
```

Next we will run the function for our first two sample locations:

```
LCp1 <- l2d(LC1km.pr, 1)
```

```
##   LC.class      loc1
## 1        1 0.6286645
## 3        3 0.3713355
```

```
LCp2 <- l2d(LC1km.pr, 2)
```

```
##   LC.class       loc2
## 1        1 0.16242038
## 3        3 0.67834395
## 5        5 0.07643312
## 6        6 0.08280255
```

Then merge the data frames created to create a starting data frame for the rest of the sample location data:

```
km1 <- merge.data.frame(LCp1, LCp2, by.x = 1, by.y = 1,
                        all.x = TRUE, all.y = TRUE, sort = TRUE)
```

Finally, we run a for-loop to populate the data frame with data for the rest of the sample locations:

```
for(i in 3:length(unique(SL$id))){
  temp <- l2d(LC1km.pr, i)
  km1 <- merge.data.frame(km1, temp, by.x = 1, by.y = 1,
                          all.x = TRUE, all.y = TRUE, sort = TRUE)
}
```

Let's look at our newly-populated data frame:

```
str(km1[,1:6]) #Summarize structure of first 6 columns
```

```
## 'data.frame':    5 obs. of  6 variables:
##  $ LC.class: Factor w/ 5 levels "1","3","5","6",..: 1 2 3 4 5
##  $ loc1    : num  0.629 0.371 NA NA NA
##  $ loc2    : num  0.1624 0.6783 0.0764 0.0828 NA
##  $ loc3    : num  0.174 NA 0.826 NA NA
##  $ loc4    : num  0.03236 0.00324 0.95469 0.00971 NA
##  $ loc5    : num  NA NA 0.9453 NA 0.0547
```

```
head(km1[,1:6], 5) #View first 5 lines of first 6 columns
```

```
##   LC.class      loc1       loc2      loc3        loc4       loc5
## 1        1 0.6286645 0.16242038 0.1741935 0.032362460         NA
## 2        3 0.3713355 0.67834395        NA 0.003236246         NA
## 3        5        NA 0.07643312 0.8258065 0.954692557 0.94533762
## 4        6        NA 0.08280255        NA 0.009708738         NA
## 5        2        NA         NA        NA          NA 0.05466238
```

As you can see, the data is listed so that each row corresponds to a unique land cover class. We want each row to correspond to a unique location, so we will transpose the data frame using the steps below:

```
LCp.km1 <- as.data.frame(t(km1[,-1])) #transpose data frame
colnames(LCp.km1) <- km1[,1] #replace column names in data frame
LCp.km1[is.na(LCp.km1)] <- 0 #replace 'NA's in data frame with 0
head(LCp.km1, 5) #look at the first 5 rows of transposed data frame
```

```
##               1           3          5           6          2
## loc1 0.62866450 0.371335505 0.00000000 0.000000000 0.00000000
## loc2 0.16242038 0.678343949 0.07643312 0.082802548 0.00000000
## loc3 0.17419355 0.000000000 0.82580645 0.000000000 0.00000000
## loc4 0.03236246 0.003236246 0.95469256 0.009708738 0.00000000
## loc5 0.00000000 0.000000000 0.94533762 0.000000000 0.05466238
```

One last step is to convert the “row names” containing our site information to a new column:

```
site <- rownames(LCp.km1) #Create new vector of row names
rownames(LCp.km1) <- NULL #Set row names in existing data frame to null
LCp.km1 <- cbind(site,LCp.km1) #add row name vector to data frame
```

All of the data is now in a data frame, but it is in “wide format”, where there are multiple observations in each row. In R, the preferred format is “long format”. We can convert to this “long format” using the *gather* tool in the **tidyr** package.

```
library(tidyr)
```

```
LCp.km1.l <- gather(LCp.km1, key = "LC_Code", value = "LC_Prop", -site)

#Now we can add the land cover names back in to the table:
LC.1km <- left_join(LCp.km1.l, LCS_LUT, by = c("LC_Code" = "LCS.Code"))
LC.1km$LC_Code <- as.factor(LC.1km$LC_Code)
head(LC.1km, 3) #View first 3 lines of updated table
```

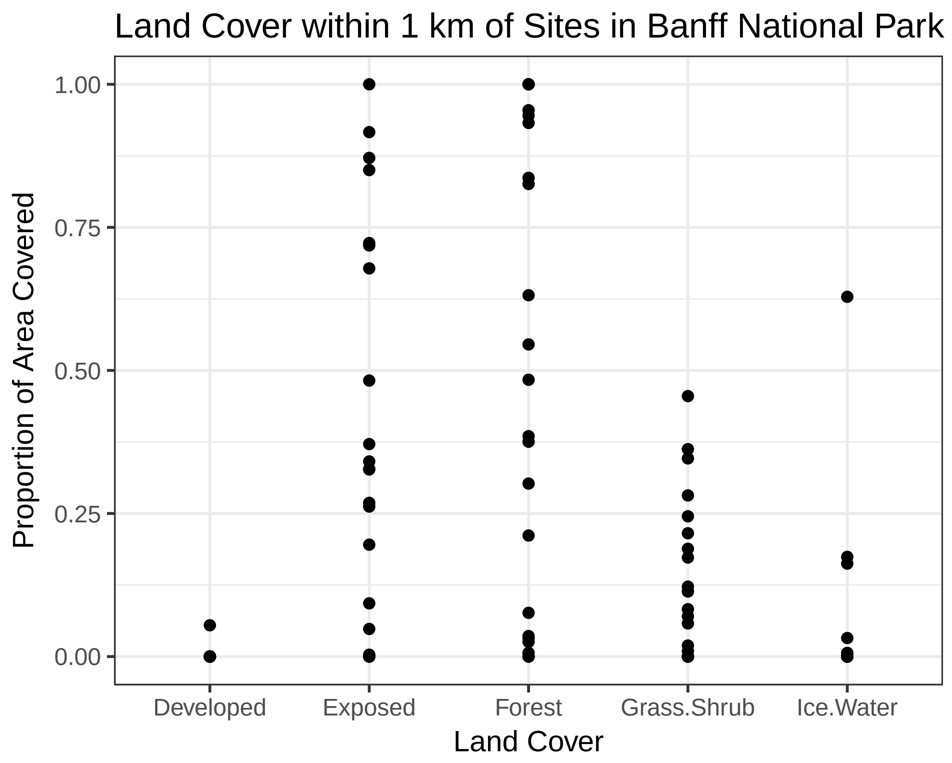
```
##   site LC_Code   LC_Prop  LCS.Desc
## 1 loc1       1 0.6286645 Ice.Water
## 2 loc2       1 0.1624204 Ice.Water
## 3 loc3       1 0.1741935 Ice.Water
```

Visualize results

Now that our data is properly formatted, we can use **ggplot** to look at the distribution of different land cover classes across all of our locations.

```
library(ggplot2)
```

```
ggplot(data = LC.1km, aes(x = LCS.Desc, y = LC_Prop)) + #set the x and y axes
  geom_point() + #add points to the graph
  labs(title = "Land Cover within 1 km of Sites in Banff National Park",
       x = "Land Cover", y = "Proportion of Area Covered") + #update the title/axes labels
  theme_bw() #change the look of the graph
```



We can see that the proportion of exposed land and forests within 1 km of each site is relatively well distributed between 0% and 100%. Grassland/Shrubland covers no more than 50% of the area within 1 km of any site, and developed land covers less than 10%.

Part III - Spatial Analysis Function

Now we have data about the landcover within 1 km of each site. However, as mentioned in the background, often we do not have enough information about the distance within which the landcover will actually be significant. The code below includes a function that follows a similar workflow to what was done in Part II above, but for multiple buffer distances. It will create an output table that includes the land cover within a defined set of distances and requires the following inputs:

1. "raster" - Land cover raster file
2. "LUT" - A lookup table that links the raster values to land cover descriptions. It should be formatted as a data frame with 2 columns - the first column listing the codes, the second listing the descriptions.
3. "samplocs" - Sample locations formatted as a shapefile with each location identified in a column named "id"
4. "buffdist" - Distances of interest

The function will then output a table giving the proportion of each type of land cover within each distance of interest.

Create function

First we will build the function in R using the code below:

```r
LCAnalysis <- function(raster, samplocs, buffdist, LUT){
  colnames(LUT) <- c("Code", "LC_Desc") #update column names in look-up table
  if(!"id" %in% names(samplocs@data)){
    print("Warning: Sample location naming error")
  } #check that the samploc data has required "id" field, if not print warning

  l2d <- function(x, lcpoly){
    temp.a <- unlist(x[lcpoly])
    temp.b <- as.data.frame(temp.a)
    names(temp.b) <- paste("SL", lcpoly, sep="")
    LC.class <- rownames(temp.b)
    cbind(LC.class, temp.b)
  } #function to convert a list of land cover proportions at each point to a data frame

  #Create empty data frame for for loop:
  LCdf <- data.frame(site = NA, LC_Code = NA, LC_Prop = NA, BDist = NA)

  for(i in 1:length(buffdist)){
    temp1 <- raster::extract(x = raster, y = samplocs, buffer = buffdist[i])
    temp2 <- lapply(temp1, table)
    temp3 <- lapply(temp2, FUN = function(x){x/sum(x)})

    SL1 <- l2d(temp3, 1)
    SL2 <- l2d(temp3, 2)
    temp4 <- merge.data.frame(SL1, SL2, by.x = 1, by.y = 1,
                              all.x = TRUE, all.y = TRUE, sort = TRUE)

    for(j in 3:length(unique(samplocs$id))){
      temp5 <- l2d(temp3, j)
      temp4 <- merge.data.frame(temp4, temp5, by.x = 1, by.y = 1,
                                all.x = TRUE, all.y = TRUE, sort = TRUE)
    } #fill in data frame for each sampling location and each buffer distance

    LC <- as.data.frame(t(temp4[,-1])) #transpose data frame
    colnames(LC) <- temp4[,1] #replace column names in data frame
    LC[is.na(LC)] <- 0 #replace 'NA's in data frame with 0

    site <- rownames(LC) #convert row names (Site ID) to a new vector
    rownames(LC) <- NULL #remove existing row names
    LC <- cbind(site,LC) #add Site ID as a new column to data frame

    #Convert from "wide" to "long" format:
    LC <- gather(LC, key = "LC_Code", value = "LC_Prop", -site)

    BDist <- rep(buffdist[i], length(LC$LC_Prop)) #Create new column for buffer distance
    LC <- cbind(LC, BDist) #add buffer distance column to data frame

    #Populate empty data frame with data for each new buffer distance:
    LCdf <- rbind(LCdf, LC)
  }
  LCdf <- LCdf[-1,] #remove "NA" row from data frame
  LCdf <- left_join(LCdf, LUT, by = c("LC_Code" = "Code")) #Add land cover descriptions
  LCdf$LC_Code <- as.factor(LCdf$LC_Code) #Convert land cover codes from character to factor
  LCdf
}
```

Apply function to data

Now that the function is built, it can be applied to our existing dataset. (*Note: you will likely get a warning saying that during the left join the factor will be coerced to a character - this can be ignored as it is taken care of later in the function*).

```
BNP.LC <- LCAnalysis(raster = LCrastRC, samplocs = SL,
                     buffdist = seq(from = 500, to = 3000, by = 500), LUT = LCS_LUT)
```

```
## Warning in left_join_impl(x, y, by$x, by$y, suffix$x, suffix$y): joining
## factor and character vector, coercing into character vector
```

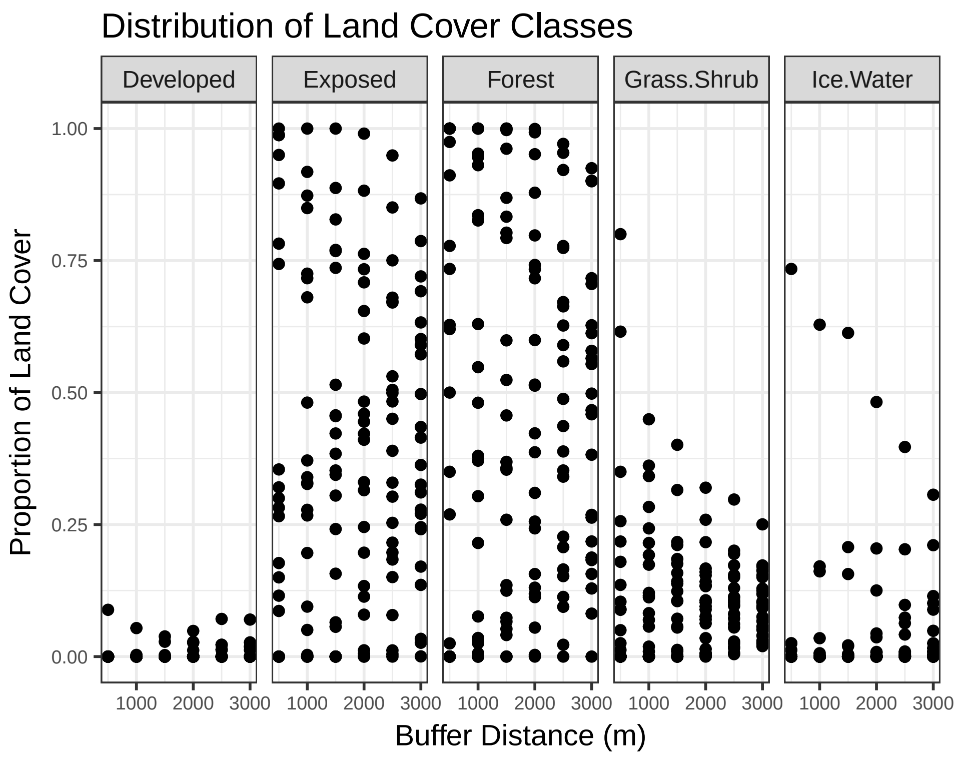
```
str(BNP.LC) #check structure of output table
```

```
## 'data.frame':    690 obs. of  5 variables:
##  $ site   : chr  "SL1" "SL2" "SL3" "SL4" ...
##  $ LC_Code: Factor w/ 5 levels "1","2","3","5",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ LC_Prop: num  0.7342 0.0123 0.0253 0 0 ...
##  $ BDist  : num  500 500 500 500 500 500 500 500 500 500 ...
##  $ LC_Desc: Factor w/ 6 levels "Agriculture",..: 6 6 6 6 6 6 6 6 6 6 ...
```

Graph results

The graph created with the code below will show how the proportion of each landcover changes across buffer distances.

```
ggplot(data = BNP.LC, aes(x = BDist, y = LC_Prop)) + #set data, and x and y axes
  geom_point() + #add points
  facet_grid(~LC_Desc) + #separate graphs so that there is one for each land cover
  labs(title = "Distribution of Land Cover Classes",
       x = "Buffer Distance (m)", y = "Proportion of Land Cover") +
  theme_bw() +
  theme(axis.text = element_text(size = 7))
```



From this graph we can see that there is a relatively uniform spread of different proportions of exposed land, forest, and developed areas across all buffer distances, but that only a couple sites have a high proportion of grassland/shrubland and ice/water, and only within the smaller buffer distances.

This kind of data output can help when assessing sites prior to field work, to ensure that there is a suitable distribution of landcover at all scales. It could also be used as input for further analysis into demographic correlations with land cover at various scales of influence.

Bibliography

References

Carvell, C. et al. (2017). Bumblebee family lineage survival is enhanced in high quality landscapes. *Nature* **543**: 547-549.

Jha, S. & Kremen, C. (2013). Resource diversity and landscape-level homogeneity drive native bee foraging. *Proceedings of the National Academy of Sciences* **110**: 555-558.

Knight, M. et al. (2009). Bumblebee nest density and the scale of available forage in arable landscapes. *Insect Conservation and Diversity* **2**: 116-124.

Lovelace, R. & Cheshire, J. (2014). Introduction to visualising spatial data in R. *National Centre for Research Methods Working Papers*, **14**(03). Retrieved April 2017 from https://github.com/Robinlovelace/Creating-maps-in-R

Redhead, J. et al. (2016). Effects of habitat composition and landscape structure on worker foraging distances of five bumble bee species. *Ecological Applications* **26**: 726-739.

Data Sources

Alberta Biodiversity Monitoring Institute. (2014). ABMI Wall-to-wall Land Cover Map 2010 Version 1.0. Retrieved March 2017 from http://www.abmi.ca/home/data-analytics/da-top/da-product-overview/GIS-Human-Footprint-Land-Cover-Data/Land-Cover.html

AltaLIS. (2016). 20K Base Features - Geoadministrative Areas - National Park Polygons. Retrieved July 2016 from http://www.altalis.com/products/base/20k_base_features.html

Appendices

Apendix I: Session Info

```
sessionInfo()
```

```
## R version 3.3.2 (2016-10-31)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 8.1 x64 (build 9600)
##
## locale:
## [1] LC_COLLATE=English_United Kingdom.1252
## [2] LC_CTYPE=English_United Kingdom.1252
## [3] LC_MONETARY=English_United Kingdom.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United Kingdom.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] ggplot2_2.2.1 tidyr_0.6.1   rgeos_0.3-22  raster_2.5-8  tmap_1.8-1
## [6] dplyr_0.5.0   rgdal_1.2-5   sp_1.2-4
##
## loaded via a namespace (and not attached):
##  [1] viridisLite_0.2.0   jsonlite_1.2        splines_3.3.2
##  [4] geojsonlint_0.2.0   foreach_1.4.3       R.utils_2.5.0
##  [7] gtools_3.5.0        shiny_1.0.0         assertthat_0.1
## [10] expm_0.999-1        stats4_3.3.2        latticeExtra_0.6-28
## [13] yaml_2.1.14         LearnBayes_2.15     backports_1.0.4
## [16] lattice_0.20-34     digest_0.6.11       RColorBrewer_1.1-2
## [19] colorspace_1.3-2    plyr_1.8.4          htmltools_0.3.5
## [22] httpuv_1.3.3        Matrix_1.2-7.1      R.oo_1.21.0
## [25] XML_3.98-1.6        rmapshaper_0.2.0    gmodels_2.16.2
## [28] xtable_1.8-2        webshot_0.4.0       scales_0.4.1
## [31] gdata_2.17.0        satellite_0.2.0     tibble_1.2
## [34] lazyeval_0.2.0      gdalUtils_2.0.1.7   mapview_1.2.0
## [37] magrittr_1.5        mime_0.5            deldir_0.1-12
## [40] evaluate_0.10       R.methodsS3_1.7.1   nlme_3.1-128
## [43] MASS_7.3-45         class_7.3-14        tools_3.3.2
## [46] geosphere_1.5-5     stringr_1.1.0       V8_1.4
## [49] munsell_0.4.3       e1071_1.6-8         classInt_0.1-23
## [52] grid_3.3.2          tmaptools_1.2       RCurl_1.95-4.8
## [55] dichromat_2.0-0     iterators_1.0.8     htmlwidgets_0.8
## [58] crosstalk_1.0.0     labeling_0.3        bitops_1.0-6
## [61] rmarkdown_1.3       boot_1.3-18         gtable_0.2.0
## [64] codetools_0.2-15    DBI_0.5-1           jsonvalidate_1.0.0
## [67] curl_2.3            reshape2_1.4.2      R6_2.2.0
## [70] knitr_1.15.1        rprojroot_1.1       spdep_0.6-11
## [73] KernSmooth_2.23-15  stringi_1.1.2       osmar_1.1-7
## [76] Rcpp_0.12.8         png_0.1-7           leaflet_1.1.0
## [79] coda_0.19-1
```