# Spatial R Exercise 7

Ben Davies

2022-05-12

# Contents

# Chapter 1

# Getting started with point patterns

Oftentimes, the geographic data we obtain in the anthropological sciences is expressed as point data. For example, points might be used to record the distribution of objects in an archaeological excavation, observations of primate activity in a study area, or instances of interpersonal violence among a population. In all instances, there is an underlying assumption that space and location matter.

In this exercise, we'll examine some methods for assessing patterning in point data.

## 1.1 Loading packages

For working with point data, we will introduce the `spatstat` package into our suite of tools.

```
require("sf")
```

```
## Loading required package: sf
```

```
## Linking to GEOS 3.9.1, GDAL 3.3.2, PROJ 7.2.1; sf_use_s2() is TRUE
```

```
require("spatstat")
```

```
## Loading required package: spatstat
```

```
## Loading required package: spatstat.data

## Loading required package: spatstat.geom

## spatstat.geom 2.4-0

## Loading required package: spatstat.random

## spatstat.random 2.2-0

## Loading required package: spatstat.core

## Loading required package: nlme

## Loading required package: rpart

## spatstat.core 2.4-2

## Loading required package: spatstat.linnet

## spatstat.linnet 2.3-2

##
## spatstat 2.3-4       (nickname: 'Watch this space')
## For an introduction to spatstat, type 'beginner'
```

```
require("ggplot2")
```

```
## Loading required package: ggplot2
```

```
require("leaflet")
```

```
## Loading required package: leaflet
```

The `spatstat` package was developed in 2002 by Adrian Baddeley principally for the analysis of 2D point patterns, and this package is still arguably the most widely used software for this application.

# Chapter 2

# Working with point data in spatstat

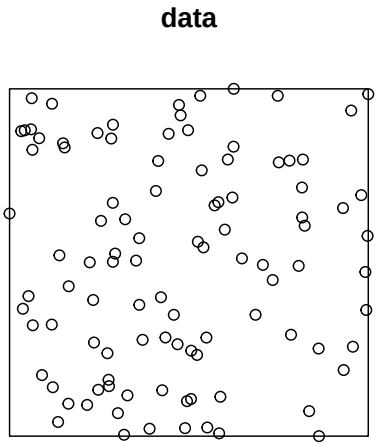
There are two principal objects required for analyzing point data in spatstat:

- a set of points expressed as coordinates in 2D space (a ppp object)
- a window of observation delineating where (an owin object)

Points can come from a number of sources. Here, we'll create some data and convert them into a ppp object.

```
x<-runif(100,0,1)
y<-runif(100,0,1)
xy<-data.frame(x,y)
w<-owin(c(min(x),max(x)),c(min(y),max(y)))
data<-as.ppp(xy,w)
```

```
plot(data)
```




## Converting `sf` data to `owin` and `ppp`

An important thing to keep in mind about point data analysis in `spatstat` is that it is concerned with distance relationships in two-dimensions. This means that any data that is not in a flat-plane coordinate space will not work. So for us to use `spatstat`, our data must either exist in a non-geographic space (as in our random points above), or must be in a projected coordinate space.

Here, we'll load in

```
ecuador<-read_sf("ecuador.shp")
ecuador<-st_transform(ecuador,32717)
ecWin<- as.owin(ecuador)
ecWin
```

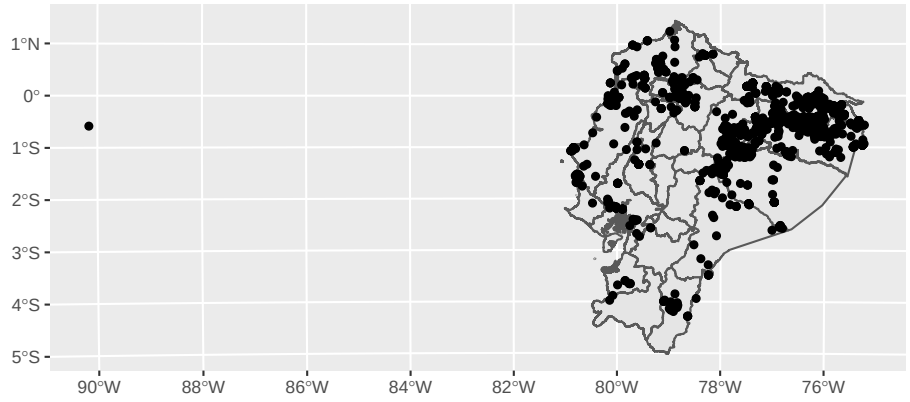
```
## window: polygonal boundary
## enclosing rectangle: [490690.4, 1147851.6] x [9445216, 10160820] units
```

When making point data

```
primates<-read_sf("SAPrimateObservations.csv")
primates<-subset(primates,countryCode=="EC")
primates<-st_as_sf(primates,coords=c("decimalLongitude","decimalLatitude"))
st_crs(primates)<-4326
primates<-st_transform(primates,32717)
```

OK, let's make sure that's in good shape:

```
g1<-ggplot() +
  geom_sf(data=ecuador) +
  geom_sf(data=primates)
print(g1)
```

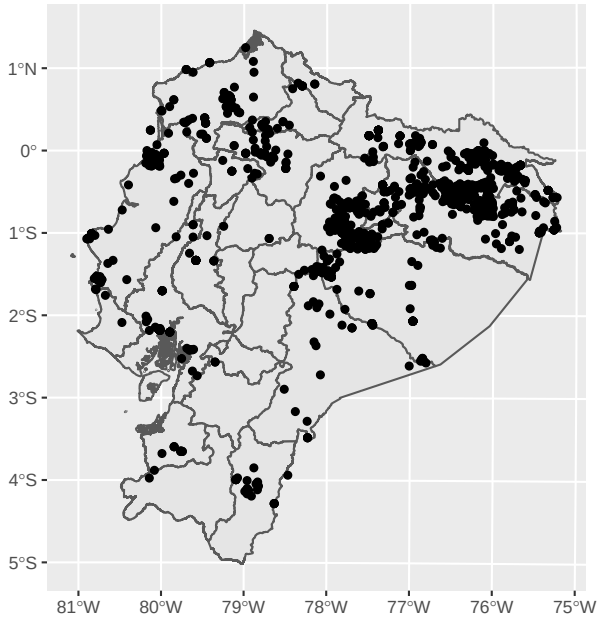



Oops! Apparently someone observed primates out in the Galapagos! We'll want to trim those points out before we convert.

```
primates<-st_crop(primates,ecuador)
```

```
## Warning: attribute variables are assumed to be spatially constant throughout all
## geometries
```

```
g1<-ggplot() +
  geom_sf(data=ecuador) +
  geom_sf(data=primates)
print(g1)
```




That's better. OK, to turn this `sf` into a `ppp` object, we'll use `as.ppp`.

```
primatesPoints<-as.ppp(primates)
```

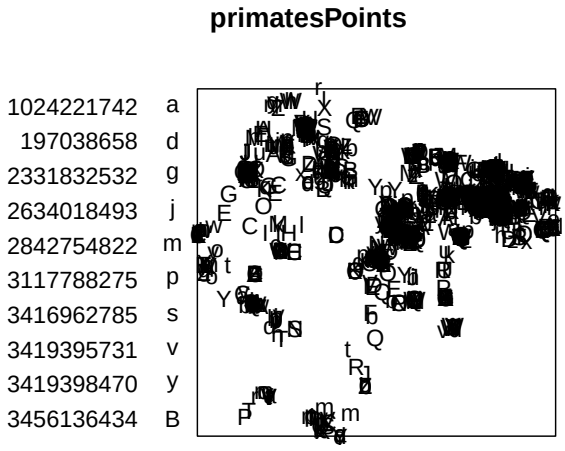
```
## Warning in as.ppp.sf(primates): only first attribute column is used for marks
```

Note that warning. "Marks" is point pattern analysis slang for attributes. Our primates data has quite a few of them. Traditionally, `spatstat` only allowed for a single column of marks, so when it converts other data to `ppp` it only keeps the first column.

Let's take a look at that pattern now.

```
plot(primatesPoints)
```

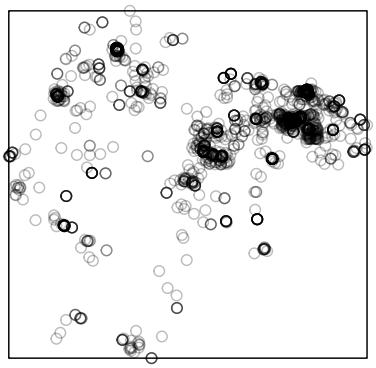
```
## Warning in default.charmap(ntypes, chars): Too many types to display every type
## as a different character
```




Yikes, what a mess. Here, `spatstat` is trying to plot all of the marks that came over with this data. This isn't really something we want to deal with at the moment, so we can get rid of them with `unmark`.

```
primatesPoints<-unmark(primatesPoints)
```

```
plot(primatesPoints)
```




Much better!

## 2.1 What is a owin object? What is a ppp object?

These objects are not something we've encountered before, so it's worth looking at their structure a bit. The `owin` carries a few pieces of information that relate to its geometry, such as whether it is a rectangle or polygon, its extent, and the vertices that form its boundary.

```
ecWin$type
```

```
## [1] "polygonal"
```

```
primatesPoints$n
```

```
## [1] 2637
```

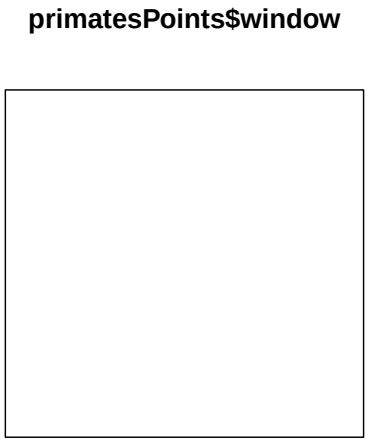
```
head(primatesPoints$x)
```

```
## [1] 1005545.2  943889.4  643055.4  643072.1 1005801.4 1005801.4
```

If a ppp object has marks, these would be stored as a vector or, in the case of more than one set of marks, a dataframe.

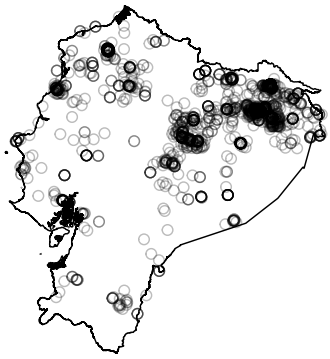
Finally, it also carries its own window object:

```
plot(primatesPoints$window)
```




Of course, here it is just the rectangular extent of the points. But much of the space this rectangle occupies would not really be considered the area of interest. This is what we need the ecWin object for: so we can assign a custom window to the point pattern.

```
primatesPoints$window<-ecWin
plot(primatesPoints)
```


primatesPoints

# Chapter 3

# Evaluating patterns

Now that we've got an understanding of `spatstat` objects, let's put them to use. Methods of spatial analysis developed for points usually concern two main aspects of their spatial patterning: density and distance.

## 3.1 Nearest neighbor

Spatial patterning in point data is often characterized by how our observed data are clustered or dispersed. Clustering might indicate some underlying process of attraction (e.g., resource abundance), while dispersed points may indicate A useful metric here is nearest neighbor distance: how far is each point from

```
head(nndist(primatesPoints))
```

```
## [1]   101.73072 16672.19195    60.50261    60.50261     0.00000     0.00000
```

The Clark-Evans test helps us to summarize these relationships over the entire dataset. Basically, this approach:

- Determines the average of distances between individual points in the observed data and their nearest neighbouring points. *Simulates distributions for the same number of points randomly distributed in the same space and calculates the same average nearest neighbour distance.* Divides the average nearest neighbour distance for the observed data by the mean of the simulated distributions.

For this metric, values over 1 tend to indicate patterns that are uniformly distributed, while values We can see what this looks like with our randomly distributed points:

```
x<-runif(100,0,1)
y<-runif(100,0,1)
xy<-data.frame(x,y)
w<-owin(c(min(x),max(x)),c(min(y),max(y)))
data<-as.ppp(xy,w)

mean(nndist(data))
```

```
## [1] 0.05059104
```

```
clarkevans(data)
```

```
##     naive  Donnelly       cdf
## 1.0283112 0.9845784 1.0299544
```

Here, we can see that the data are pretty close to one. This is what we'd expect from randomly distributed data. Now let's take a look at our primate data.

```
mean(nndist(primatesPoints))
```

```
## [1] 1171.82
```

```
clarkevans(primatesPoints)
```

```
##     naive       cdf
## 0.2415733 0.1920958
```

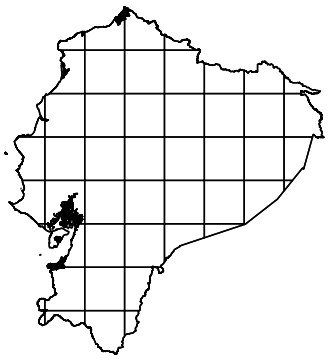
Our primate groups are, on average around a kilometer from their nearest neighbor. And, confirming what we could see from the, there is some notable clustering.

# Chapter 4

# Quadrat analysis

For quadrat analysis, we divide the up the study area into a regular grid and then count how many instances there are of a thing in that grid. In the case of Ecuador, we'll divide it into a 8x8 grid.

```
plot(quadrats(ecWin,8,8),main="")
```

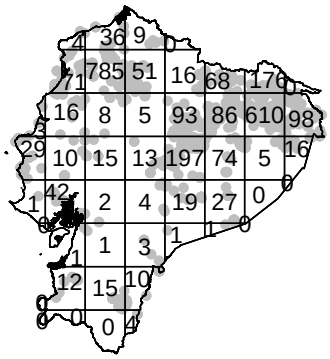


We can also create a primate dataset without marks. This is mostly just so they won't plot.

```
p<-unmark(primatesPoints)
```

Now we can use `quadratcounts`

```
quad <- quadratcount(p, nx= 8, ny=8)
#plot points
plot(p, pch=16, cols="grey", main="")
plot(quad, add=TRUE)
```




Just looking at this pattern, we can see that some quadrats have more than 100 observations, while others have none at all. This suggests intuitively that the distribution of is patterned, but we can estimate whether this is meaningful using `quadrat.test`.

```
quadrat.test(p, 8, 8)
```

```
## Warning: Some expected counts are small; chi^2 approximation may be inaccurate
```

```
##
##  Chi-squared test of CSR using quadrat counts
##
## data:  p
## X2 = 11974, df = 49, p-value < 2.2e-16
```

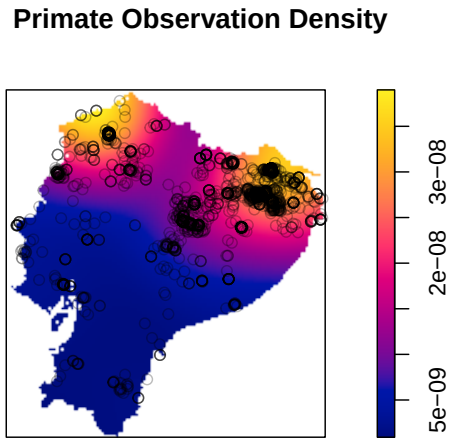
```
## alternative hypothesis: two.sided
##
## Quadrats: 50 tiles (irregular windows)
```

Note that the p-value is $<$ 2.2e-16. This means that the

## 4.1 Kernel Density Estimation

We can visualize broad scale patterning in our data through a kernel density estimate (KDE). This is similar to the above approaches in that it calculates the number of of instances

```
ecDensity <- density(p)
plot(ecDensity,main="Primate Observation Density")
plot(p,add=T)
```




These exercises are meant to introduce you to `spatstat` and start looking at spatial analysis in R.

## 4.2 Try it yourself

`spatstat` comes with some pre-made datasets to test things out with. These are already saved as `ppp` objects when you call them. Try doing the following:

- Look at nearest neighbor with the `redwoods` and `amacrine` datasets
- Create a kernel density estimate from the `cells` data
- Do a quadrat analysis, including the quadrat test, using the `nztrees` data

# Chapter 5

# Bringing it all together!

OK, now you've seen how we get started with doing some spatial analysis in `spatstat`. Like `sf` and `terra`, this package has its own quirks and conventions that take some getting used to. For this exercise, see if you can do the following:

*Load in the makiloa_boundary2.shp and primary_features.shp files as vector data* Convert the boundary to an owin object and the features to a point pattern *Explore the pattern using quadrats, kernel density, and nearest neighbor distance