 EEB 5301 February 2017

# Spatial Analyses in R

R commands are shown in **red**. After typing any of these lines in the RUI gui, you must hit the return key. Output produced in the R Console is shown in **blue**.

**Install the spatstat packages.** (Instructions for downloading and installing a package are given in "Matrices in PopBio.pdf".)

Note: If you're not using the latest version of R, you may have to update in order to install a package. This entails uninstalling R, downloading and installing the latest version, and then installing the package within R, rather than within RStudio.

The necessary commands from spatstat are described below. If you want to learn more, a helpful tutorial is available from this web site:

http://www.stats.uwo.ca/faculty/kulperger/S9934a/Papers/SpatStatIntro.pdf

A reference manual is also available, but you won't need it for this exercise. If you want it, go to this website:
http://cran.r-project.org/
click on "Packages" in the left column, then the initial letter of the package name, then scroll down to find the package you are interested in. Click on the link and look for a link for the Reference manual.

## Analysis of point patterns

**Load the spatstat package**

```
library(spatstat)
```

**Load the data for Japanese pines.** (This is one of 26 data sets included in the spatstat package.)

```
data(japanesepines)
```

Whenever you load a data set, it's a good idea to use the "summary" function to examine some of its properties.

```
summary(japanesepines)
```

```
Planar point pattern:  65 points
Average intensity 65 points per square unit
(one unit = 5.7 metres)

Coordinates are given to 2 decimal places
```

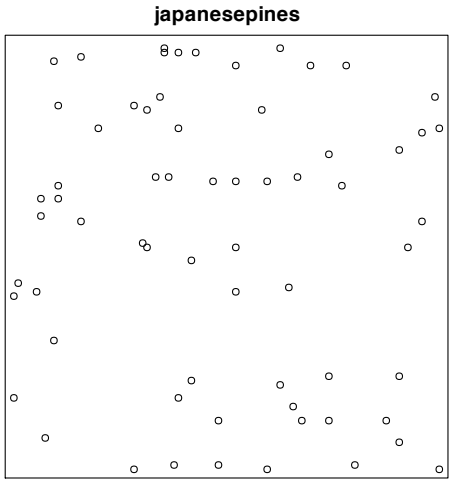
i.e. rounded to the nearest multiple of 0.01
units (one unit = 5.7 metres)

Window: rectangle = [0, 1] x [0, 1] units
Window area = 1 square unit
Unit of length: 5.7 metres

This data set contains the positions of 65 black pine saplings in a square region that is 5.7 m x 5.7 m in natural forest in Japan. Their coordinates have been divided by 5.7 to rescale the map into a 1 x 1 square.

**To produce a map of the points:**

```
plot(japanesepines)
```




**Calculate the Clark and Evans Index**

```
clarkevans(japanesepines)
```

```
  naive       Donnelly      cdf
1.064002   1.007507      1.056241
```

The "naive" value does not take into account that the nearest neighbors of some of the pines may be outside the mapped region. The average nearest neighbor distance is therefore likely to be an overestimate. The second value uses Donnelly's edge correction to produce a less biased value. For this example, the value of the index is close to 1, the value expected for random placement. (cdf: The clarkevans function also produces another estimate based on the cumulative distribution function method.)

**Test whether the observed Clark and Evans index is consistent with the null hypothesis of complete spatial randomness.**

The function is "clarkevans.test". The Donnelly edge correction can be used, but the calculations take a bit longer. (Wait for the prompt > to appear in the R Console. This may take >30 seconds for a data set with 100 points.)

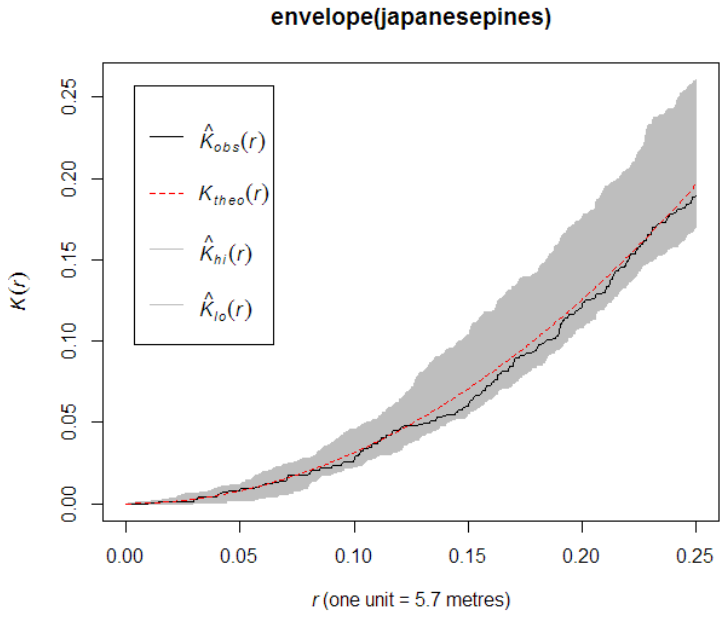
```
clarkevans.test(japanesepines, correction = "Donnelly")

Clark-Evans test
Donnelly correction
Monte Carlo test based on 999 simulations of CSR with fixed n

data:  japanesepines
R = 1.0075, p-value = 0.916
alternative hypothesis: two-sided
```

Because the *p* value is much greater than 0.05, the test provides no clear evidence against the null hypothesis of spatial randomness. Here, R is the observed value of the Clark and Evans index.

**Caclulate Ripley's K:**




```
RipleysK <- Kest(japanesepines)
plot(envelope(japanesepines))
```

The Kest command executes 99 random placements of the observed number of points (representing 65 pine trees) within a region with the same dimensions as the study region. This may take a minute or two (depending on the number of points). The resulting plot shows the confidence envelope for Ripley's K resulting from these simulations (the gray region bounded by "hi" and "lo"). Also shown is the plot expected for Ripley's K under the null hypothesis of complete spatial randomness ("theo") and the plot of Ripleys K for the observed positions of the pines ("obs").

Interpretation: The observed values of Ripley's K are close to the expected values and are entirely within the confidence envelopes. There is no clear evidence for departures from spatial randomness at any scale.

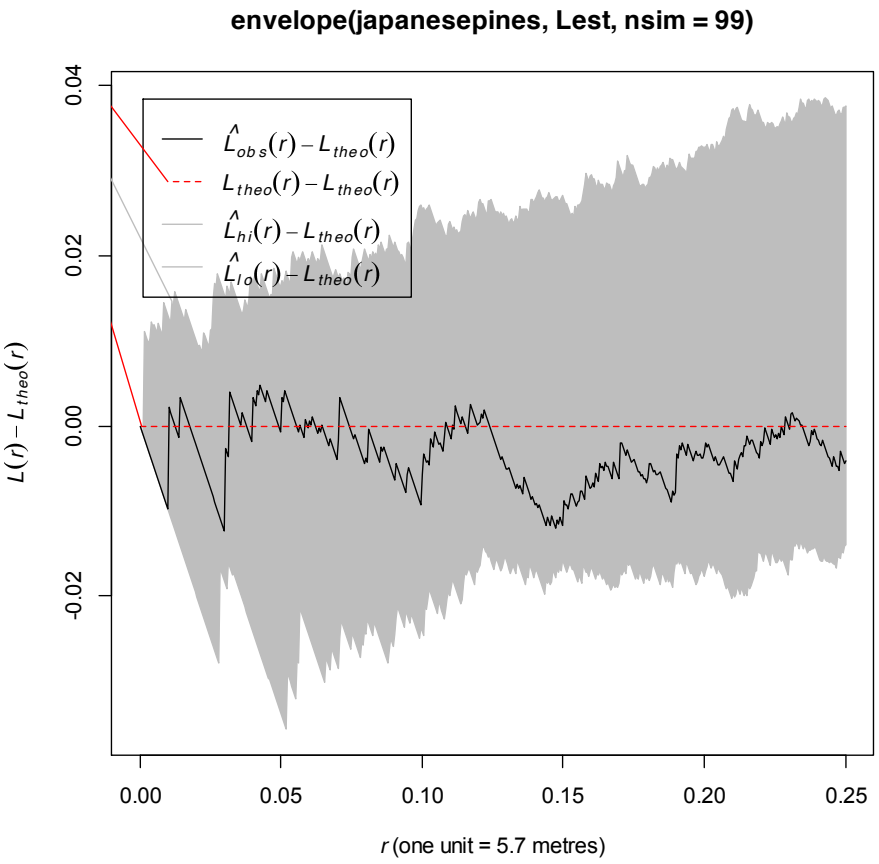
**To calculate Ripley's L**:

```
RipleysL <- Lest(japanesepines)
```

To plot Ripley's L-hat with a confidence envelope, and so that the x-axis corresponds to the expected values:

```
plot(envelope(japanesepines, Lest, nsim = 99), .-theo~r)
```

(If the legend covers the plot, try expanding the plots pane window in RStudio, then execute the plot command again.)

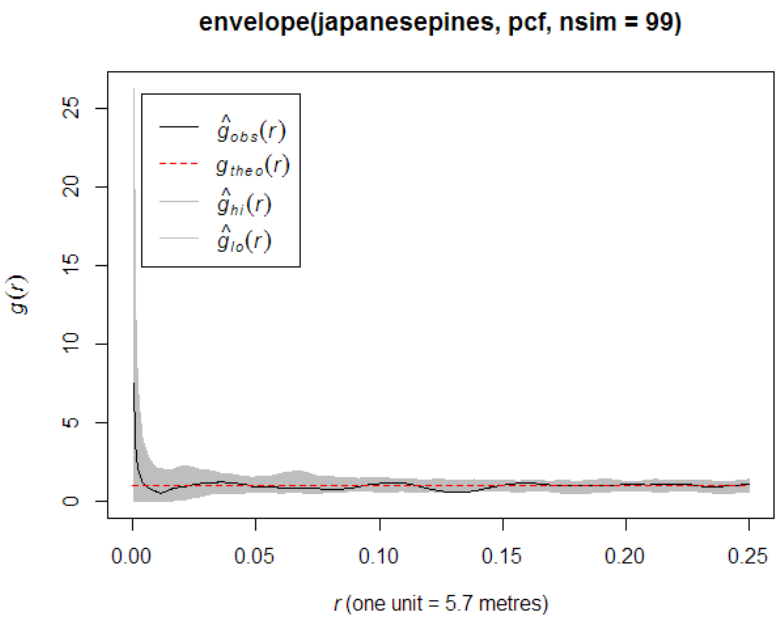



**The pair correlation function.**

These commands calculate and plot the pair correlation function:

```
p <- pcf(japanesepines)
plot(p)
```

The envelope function can be used again to draw confidence envelopes for the function based on multiple simulations of random placement. Here, the number of simulations is set to 99.




```
plot(envelope(japanesepines, pcf, nsim = 99))
```

You can see that the function calculated from the pine positions is highest at very short distances, but that is true for simulations of random placement as well. Again, there is no clear evidence for a non-random pattern.

**Quadrats and the Poisson distribution**

Following the methods discussed in Chapter 5 of Vandermeer and Goldberg, we can use the spatstat package to divide the forest plot into a grid of cells, to count the number of trees in each cell, and to compare the observed distribution of counts to the Poisson distribution. The methods are illustrated using a data set with a larger number of trees.

```
data(longleaf)
summary(longleaf)
```

The output shows that there are 584 trees in 200 m x 200 m region. The spatstat manual tells us that the trees are longleaf pines, *Pinus palustris*, mapped in southern Georgia and that the data set includes the dbh (diameter at breast height) for each tree.

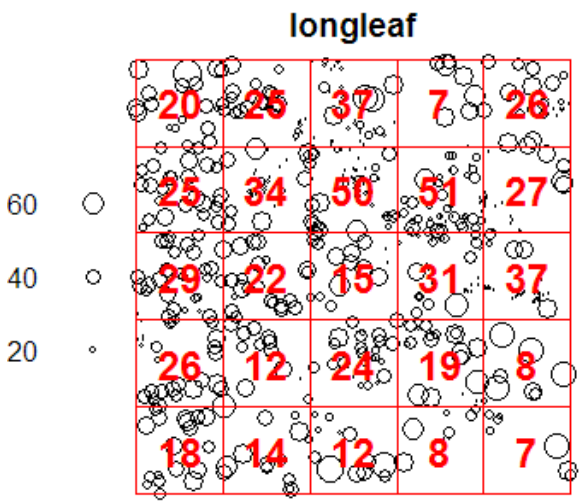
The following command places a grid over the study region and counts the number of trees in each cell. The default grid is 5 by 5. The results are stored in "QC".

```
QC <- quadratcount(longleaf)
```

Now the grid and counts can be plotted on top of the map.

```
plot(longleaf)
plot(QC, add=TRUE, cex = 1.5, col = "red", font = 2)
```

The plot command displays the sizes and positions of the trees, but only the positions are used for quadrat counts. In the second command, four options are included. The option "add=TRUE" causes the grid and counts to be plotted on top of the existing map, rather




than replacing it. The "cex = 1.5" option increases the size of the font, making it 1.5 times the default size. The 'col = "red"' option draws the grid and the counts in red, making them easier to see. The "font = 2" option specifies a bold font.

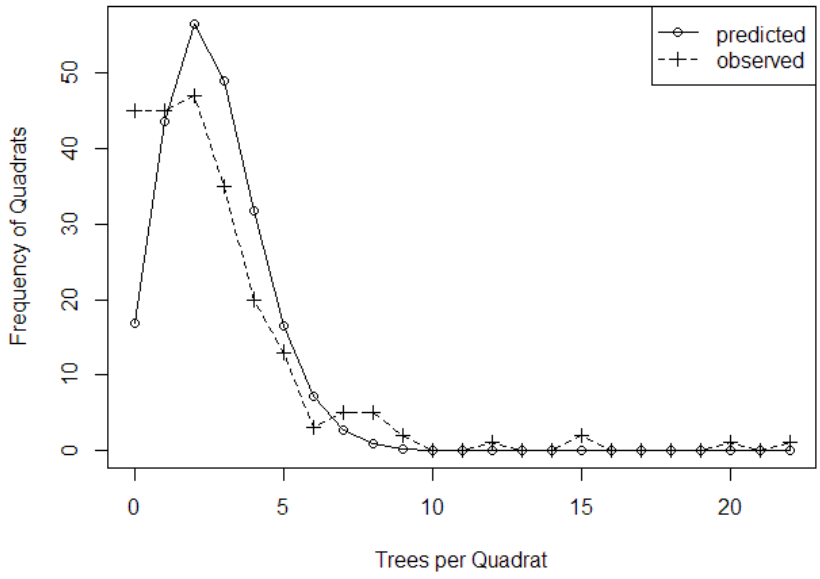
There are 20 trees in the upper left cell, 7 in the lower right cell, etc.

Instead of accepting the default 5 by 5 grid, different values can be selected by specifying the number of divisions along the *x* and the *y* axes. In this case, a 15 by 15 grid is produced.

This command places a grid of 15 x 15 cells (instead of the default grid of 5 x 5 cells).

```
QC <- quadratcount(longleaf, 15, 15)
```

With some more work, the quadrat counts can be turned into a figure a graph like that in Fig. 5.6 of the text.




The dashed line shows the observed number of quadrats with each number of trees, from 0 to 22. The solid line shows the expected frequencies for a spatially random population with the same density of trees. There are more quadrats with 0 trees and more quadrats with >10 trees than expected, indicating a clumped distribution.

The **Appendix** explains the code used to produce the graph (if you're interested).

Finally, the quadrat.test function tests whether the quadrat counts are consistent with the null hypothesis of random placement. A 15 by 15 grid is used; method = "M" indicates

Monte Carlo methods (the positions of the points are chosen randomly), and nsim = 1999 indicates that Monte Carlo placement is repeated 1999 times.

```
quadrat.test(longleaf, 15, 15, method="M", nsim=1999)
```

Here is the output:

```
Conditional Monte Carlo test of CSR using quadrat counts
Pearson X2 statistic

data:  longleaf
X2 = 766.7705, p-value = 0.001
alternative hypothesis: two.sided

Quadrats: 15 by 15 grid of tiles
```

The low *p* value indicates that the quadrat counts are very unlikely under the null hypothesis of CSR (complete spatial randomness).

For the problem set, you are asked to repeat the analyses above with a different data set, which is also provided with the spatstat package. The "lansing" data set contains the positions of more than 2000 trees of various species within a 924 ft x 924 plot in Lansing, Michigan. Your task is to analyze the spatial positions of white oaks in this plot. To select just the red oaks, they can be extracted from the entire data set as follows:

```
whiteoaks <- lansing[lansing$marks == "whiteoak"]
```

This "whiteoaks" object can now be used in the same way as "japanesepines" in the tutorial, except that you don't need to use the "data" command.

**Appendix** (Optional)

Here is an explanation of the script used to produce the last figure.

First, the quadratcount function from the spatstat package is used to count the number of trees in each cell of a 15 by 15 grid, using the longleaf data. The results are stored in "QC."

```
QC <- quadratcount(longleaf, 15, 15)
```

To scale the graph, the highest count (the maximum number of trees in any cell) is found using the max function and stored in "highest". This turns out to be 22.

```
highest <- max(QC)
```

The hist (histogram) function can be used to count the number of cells that have 0 trees, the number with 1 tree, the number with 2 trees, and so forth up to the maximum number of trees. The results are stored in an object called "h". However, setting up the hist function so that it makes the right counts is a bit tricky. By default, it will lump together quadrats with 0, 1, or 2 trees, but it's better to count these separately. The "breaks" option is used to split things up in the right way. The first count will be the number of quadrats with at least -1 tree and no more than 0, etc.

```
h <- hist(QC, breaks=-1:highest)
```

The object "h" has several parts to it. One of them, indicated by h$counts, holds the observed values. Just to make things clear, they are stored in a new object named "observed."

```
observed <- h$counts
```

The highest count is found. This turns out to be 47. (There are 47 quadrats containing two trees.)

```
MaxObs <- max(observed)
```

This makes a vector of *x* values, for graphing, from 0 to the highest number of trees

```
x <- seq(0,highest,1)
```

The function dpois is used to find the density of the Poisson distribution for each value of *x*. These values are multiplied by the square of 15, because there are 15 x 15 cells.

```
predicted <- 15^2*dpois(x,mean(QC))
```

The highest predicted frequency is found.

```
MaxPred <- max(predicted)
```

To scale the graph, we need the maximum observed or predicted frequency – whichever is greater.

```
MaxFreq <- max(MaxObs, MaxPred)
```

Finally, the plot is produced in several steps.

First a plot is made of the predicted frequencies versus *x*. The *x* and *y* axes are labeled. The limits of the y axis are specified to extend from 0 to the highest frequency that will be plotted, which has been stored in MaxFreq.

```
plot(x, predicted,
    xlab = "Trees per Quadrat", ylab = "Frequency of Quadrats",
    ylim = c(0,MaxFreq))
```

A line is added connecting the symbols for the predicted values.

```
lines(x, predicted, type = "l")
```

Points representing the observed frequencies are added. The “pch” option specifies the plot character. If this is set to 3, then plus symbols + are used, instead of the default open circles.

```
points(x, observed, pch = 3)
```

A line is added connecting the symbols for the observed values. The “lty” option specifies the line type. If this is set to 2, then a dashed line is drawn instead of the default solid line.

```
lines(x,observed, type = "l", lty = 2)
```

A legend is added. It is placed in the top right; the labels are “predicted” and “observed”; the print characters are open circles (pch = 21) and plus symbols (pch = 3), and the line types are solid (1) and dashed (2).

```
legend("topright",
      legend = c("predicted","observed"),
      pch = c(21,3),
      lty = c(1,2)
)
```