Lab 3: Spatial Point Pattern Analysis in R

Objectives

- Exploratory analysis of spatial point patterns
- Hypothesis testing of spatial point patterns

References: The lab instruction here is enough for completing the assignment. However, if you want to explore more about point pattern analysis, you are encourage to read more, such as the following references.

(1) Chapter 3, Bailey, Trevor C. and Anthony C. Gatrell. (1995). *Interactive Spatial Data Analysis*, Prentice Hall.
(2) Bivand, R.S., Pebesma, E.J., Gómez-Rubio, V. (2013). *Applied spatial data analysis with R*, Second Edition. Springer. Chapter 7.
(3) Adrian Baddeley, Ege Rubak, Rolf Turner, (2015). *Spatial Point Patterns: Methodology and Applications with R*, CRC Press.
(4) https://cran.r-project.org/web/packages/spatstat/

0. Install Packages

In R, many packages have similar functions. There are several libraries for point pattern analysis in R: spatstat, splancs, and so on. We are going to use **spatstat** library for this lab exercise. Install the package first.

```
> install.packages("spatstat",repos = "http://cran.r-project.org")
```

1. Load the R Packages

We will use the R package called `spatstat`, which is an R library for the statistical analysis of spatial data, mainly spatial point patterns.

```
> library(spatstat)        # load library
> ?spatstat                # help for the library
> rm(list=ls())       # clear workspace
```

2. Exploratory analysis of spatial point patterns

```
> data(japanesepines)                    # load Japanese pine data (Diggle, p.1)
> jp = japanesepines                     # save it with another name
> class(jp)
```

To explore jp data briefly, get summary and plot them.

```
> summary(jp)
> plot(jp,axes=T,main=" 65 Japanese black pine saplings") # plot
```

2.1 First-order effect

Kernel estimation is used to investigate the first-order effect. With different bandwidths, kernel estimation can generate different outputs. In `spatstat`, `density.ppp` function returns the intensity using the isotropic Gaussian kernel of standard deviation sigma with point masses at each of the data points.

```
> ?density.ppp          #refer to the help
> jp.Z.05 = density.ppp(jp, 0.05)

> par(mar=c(0,0,1,1)) # set plot margin
> tl.05 = expression(paste("Kernel Estimation of JP: ", sigma, " = 0.05"))
> plot(jp.Z.05, main = tl.05)
> plot(jp.Z.05, main="Kernel Estimation of JP: sigma = 0.05")
> points(jp,pch="+",col="6")

> jp.Z.1 = density.ppp(jp, 0.1)
> tl.1 = expression(paste("Kernel Estimation of JP: ", sigma, " = 0.1"))
> plot(jp.Z.1, main=tl.1)
> points(jp,pch="+",col="6")
```

Assignment I
o Include the plots of kernel estimation with $\sigma = 0.05$ and $\sigma = 0.1$.
o Describe first order effects in each plot of the two kernel estimations.

2.2 Second-order effect

Nearest neighbor distances including **event-to-nearest-event distances** and **point-to-nearest-event distances** are useful tools to explore the second order effect of spatial point patterns. We first use **event-to-nearest-event distances** to explore the second-order effect. The `Gest` function in `spatstat` returns 1 raw `Ghat`, 3 corrected `Ghat` (`rs, km, hazard`), and theoretical `Ghat` (`theo`). In this exercise, we will mostly use the raw estimate of `Ghat`.

```
> jp.ghat = Gest(jp)
> g.max = max(jp.ghat$r)
> plot(jp.ghat,cbind(rs,theo)~r, main="G Estimates",
      xlim=c(0,g.max), xlab="r", ylab="G(r)")
> jp.ghat
```

Similarly, we can use **point-to-nearest-event distances** to explore the second-order effect.

```
> jp.fhat = Fest(jp)
> f.max = max(jp.fhat$r)
> plot(jp.fhat, cbind(rs,theo)~r, main="F Estimates",
       xlim=c(0,f.max),xlab="r", ylab="F(r)")
```

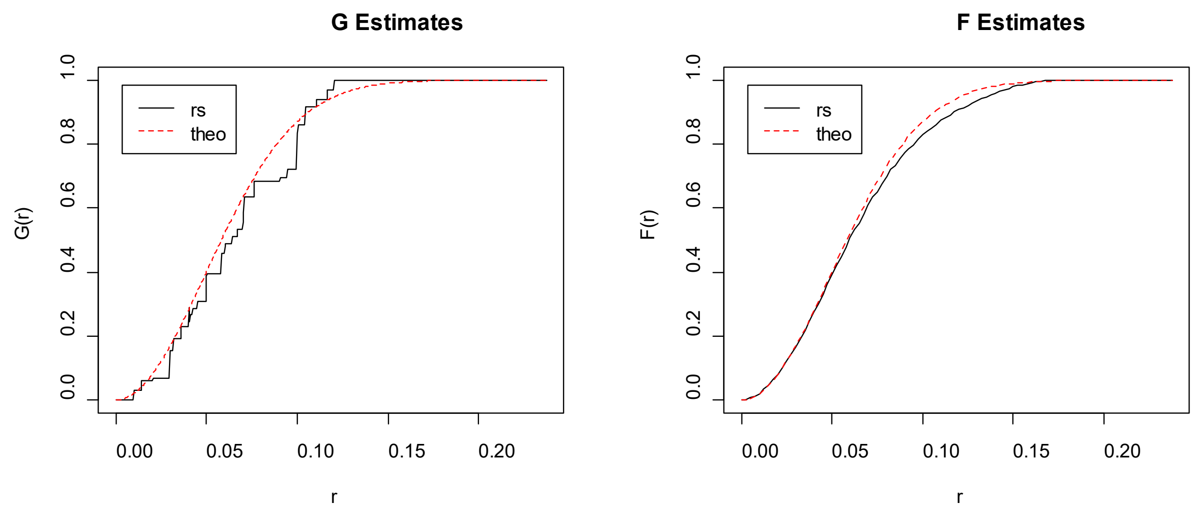

Figure 1. Ghat and Fhat for Japanese pine saplings data

K function (L function) is another tool to explore the second-order effect. Three different K function estimates can be generated from `Kest` in `spatstat`: `border`, `isotropic`(or `Ripley`), and `translate`. It also returns theoretical `Khat`values. To get information about the different options, please see the help file. For this exercise, we will use `border` corrected `Khat`.

```
> ?Kest
> jp.khat = Kest(jp)
> plot(jp.khat, cbind(border, theo)~r, main="K function for JP data")
```

# L plot

```
> plot(jp.khat, sqrt(border/pi)-r ~ r, ylab="L(r)",
        main="L function for JP",ylim=c(-0.025,0.025))
> abline(h=0,lty=2,col='red')
```

or

```
> plot(jp.khat, sqrt(cbind(border,theo)/pi)-r ~ r, ylab="L(r)",
       main="L function for JP",ylim=c(-0.025,0.025))
```

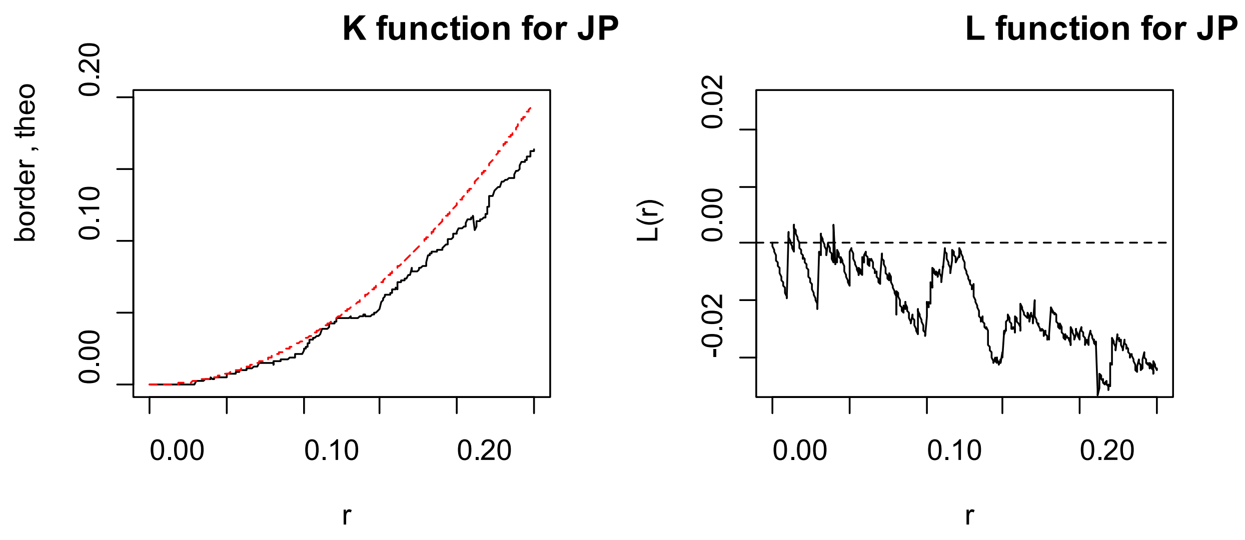

Figure 2. Khat and Lhat for Japanese pine saplings data

Assignment II
○ Create plots like Figure 1 and Figure 2 without specifying *xlim* argument (Plot function will use recommended range which is specified in the ppp object). Include plots in your report.
○ Interpret the plots (i.e. Ghat, Fhat, Khat, and Lhat) focusing on whether the point pattern is clustered, random, or regular and what leads you to the conclusion.

3. Hypothesis testing of spatial point patterns

3.1 Simulating CSR

Simulation is an important approach for analyzing spatial point processes. By simulating a point process, we are in effect "observing" one possible manifestation of the type of process we are interested in studying. Oftentimes, such observations are called "realizations" in Statistics.

Let us start by simulating some homogeneous spatial Poisson (CSR) point processes. "`spatstat`" provides `runifpoint` function to generate uniform random points.

```
> ?runifpoint
> N = 65                    # number of points to generate
> r1 = runifpoint(N)               #Generate N uniform random points
> par(mar=c(0,0,0,0)) # set plot margin
> plot(r1,pch="+",main="65 points under CSR")
```

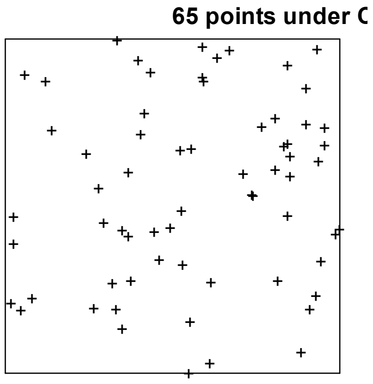

Figure 3. 65 random points

Repeat the generation of the N points several times and re-plot to see just how much variation there is in the different realizations from the same CSR process. Note, sometimes people refer to the process as a **homogeneous binomial spatial point process** if it is conditioned on a fixed N.

3.2 Testing CSR using nearest neighbor distances and Monte Carlo simulation

Upper and lower bound can be achieved from Monte Carlo simulation. The following function creates upper bound and lower bound.

```
> r1.ghat=Gest(r1)
> r1.ghat$rs                # Border corrected estimate of G(r): Ĝ values at every r
```

# The following two plots are similar

```
> par(mfrow=c(1,2))
> plot(r1.ghat)
> plot(r1.ghat$r, r1.ghat$rs,type="l",xlim=c(0,0.1))
```

#One more random points

```
> r2 = runifpoint(65)

> r2.ghat=Gest(r2)
> plot(r2.ghat)
> plot(r2.ghat$r, r2.ghat$rs,type="l",xlim=c(0,0.1))
> par(mfrow=c(1,1))
```

Monte Carlo simulation: Repeat the generation of random points 100 times to obtain the distribution of `Ghat` values under CSR.

```
> hold = matrix(0, nrow=100, ncol=length(r1.ghat$r))        #To save results
> dim(hold)

for (i in 1:100) {
    rp =runifpoint(65)
    rp.ghat = Gest(rp, r1.ghat$r)
    rp.ghat.rs = rp.ghat$rs
    hold[i,] = rp.ghat.rs
}
```

```
> r1.ghat$r[100]                #the 100th distance point
> summary(hold[,100])           #summary of Ghat at the 100th distance point
> max(hold[,100])                 #upper bound of 100 simulations at the 100th point
> min(hold[,100])                 #lower bound of 100 simulations at the 100th point
> apply(hold,2,max)[100]
```

# get the upper bound and lower bound at every point

```
> ubnd = apply(hold,2,max)
> lbnd = apply(hold,2,min)
```

# plot the results with JP data (Figure 5)

```
> plot(jp.ghat,rs~r,xlim=c(0,max(r1.ghat$r)))
> par(mar=c(4,4.5,1.5,.5))
> plot(jp.ghat,rs~r,xlim=c(0,max(r1.ghat$r)))
> lines(r1.ghat$r,ubnd,lty=2,col=2)
> lines(r1.ghat$r,lbnd,lty=2,col=2)
> dev.copy(jpeg,"Gplot.jpg",width=5,height=5,units="in",res=300)
> dev.off()
```

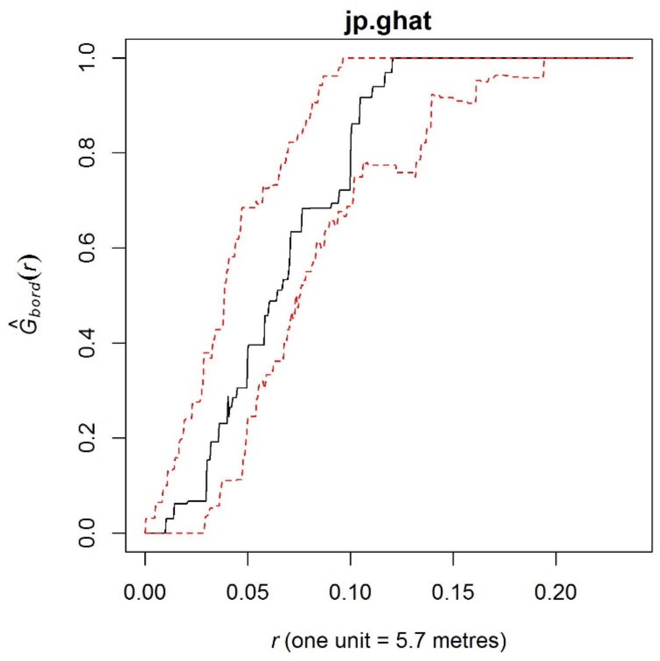

Figure 4. Ghat and corresponding bounds from simulation

Let's make a function for getting the upper bound and lower bound through simulation

```
ghat.env = function(n, s, r, win=owin(c(0,1),c(0,1)))
{
   hold = matrix(0, s, length(r))
   for(i in 1:s)
   {
      hold[i,] = Gest(runifpoint(n, win=win), r=r)$rs
```

```
    }
    mn = apply(hold, 2, mean)
    Up = apply(hold, 2, max)
    Down = apply(hold, 2, min)
    return(data.frame(mn, Up, Down))
}
```

```
> jp.ghat = Gest(jp)
> jp.win = window(jp)
> plot(jp.ghat,rs~r, main="G estimates")
> jp.genv = ghat.env(n=jp$n, s=100, r=jp.ghat$r, win=jp.win)
```

# upper and lower envelopes

```
> lines(jp.ghat$r, jp.genv$Up, lty=5, col=2)
> lines(jp.ghat$r, jp.genv$Down, lty=5, col=3)
```

```
> # alternatively, you can use the envelope function:
> plot(envelope(jp,Gest))
```

Similarly, the upper and lower bound for F function can be achieved from Monte Carlo simulation. The function below creates upper bound and lower bound for F function.

```
fhat.env = function(n, s, r, win=owin(c(0,1),c(0,1)))
{
    hold = matrix(0, s, length(r))
    for(i in 1:s)
    {
        hold[i,] = Fest(runifpoint(n, win=win), r=r)$rs
    }
    mn = apply(hold, 2, mean)
    Up = apply(hold, 2, max)
    Down = apply(hold, 2, min)
    return(data.frame(mn, Up, Down))
}
```

```
> jp.fhat = Fest(jp)
> jp.win = window(jp)
> jp.fenv = fhat.env(n=jp$n, s=100, r=jp.fhat$r, jp.win)
> plot(jp.fhat,rs~r, main="F estimates")
```

# upper and lower envelopes

```
> lines(jp.fhat$r, jp.fenv$Up, lty=5, col=2)
> lines(jp.fhat$r, jp.fenv$Down, lty=5, col=3)
```

Assignment III
o Create a plot for Ghat and Fhat with simulating bounds for Japanese pine sapling and provide it in your report o Based on the Ghat and Fhat with the simulation results, interpret the point pattern of Japanese pine sapling

3.3 Testing CSR using K function and Monte Carlo simulation

Similarly, the bounds for K function can be created from simulation.

Assignment IV

- o Create a function to get upper bound and lower bound for K function as above

  [Hint] Kest(runifpoint(n, win=win), r=r)$border

- o Create a plot for K hat with simulating bounds for Japanese pine sapling and provide it in your report

- o Based on the Khat with simulation results, interpret the point pattern of Japanese pine sapling

4 Conduct point pattern analysis with another dataset

With knowledge you have got from this lab exercise, analyze spatial point pattern of following datasets.

- Redwood dataset

```
> data(redwood)              #load the dataset
> ?redwood                   #get information for the dataset
> summary(redwood)
```

- Regular points

```
> regular = rsyst(nx=10)   #generate regular pattern
> summary(regular)
```

Assignment V

- o For each of the above dataset (i.e. redwood and regular), conduct point pattern analysis including kernel estimation, nearest neighbor distance, point-to-nearest event distance, K function, and test for CSR with simulations (for nearest neighbor distance, point-to-nearest event distance, K function)

- o Report the results of the point pattern analyses including all test results and plots.

- o Based on the results, make conclusions about point patterns of the two dataset (aspects of first and second order) and explain reasons for the conclusion.

-End-