# Hands-on Exercise 5: Analysing Marked Point Patterns

In this hands-on exercise, you will gain hands-on experience on using appropriate R functions to analyse marks spatial point events. The case study aims to discover the spatial point processes of childecare centres by operators in Singapore.

AUTHOR

**Dr. Kam Tin Seong, Associate Professor of Information Systems (Practice)**

AFFILIATION

School of Computing and Information Systems, Singapore Management University

PUBLISHED

Sept. 24, 2021

## Contents

**Overview**
- The research questions
- The data

**Installing and Loading the R packages**

**Importing the Geospatial Data**

**Mapping the geospatial layers**

**Spatial Data Wrangling**
- Converting the SpatialPointsDataFrame into ppp format
- Avoiding duplicated spatial point event by using jittering method
- Creating ***owin***
- Extracting study area
- Converting the spatial point data frame into generic sp format
- Creating ***owin*** object
- Combining childcare points and the study area
- Plotting childcare points and the study area

**Analysing Marked Point Patterns**
- First-order Spatial Point Patterns Analysis
- Second-order Multi-tpye Point Patterns Analysis: Cross K-Function
- Performing CSR testing on the Cross K-Function
- Second-order Multi-tpye Point Patterns Analysis: Cross L-Function
- Performing CSR testing on the Cross L-Function

## Overview

As discussed in class, a point pattern dataset contains a complete enumeration of events (i.e., objects of interest) occurring in a defined study region. These events could represent anything with a measurable location including traffic accidents, crime occurrences, social service location, business establishment locations, etc. In addition to locational information, each event could have an associated continuous (e.g., number of students, volume of sales) or categorical measurement (e.g., type of schools, operators of the fast food chains). The measurements are called **marks** and the events with marks are called a **marked point pattern**.

Marked point patterns have first-order properties, which are related to the intensity (i.e., density) of events and associated marks across the study region, and second-order properties, which are related to the spatial dependence (i.e., spatial arrangement) of the events and associated marks across the study area.

### The research questions

The specific question we would like to answer is:

- are the locations of childcare centre by different business groups (i.e. NT, PT, RC, ST) spatial independent?
- If the answer is NO, are there any phenomena of attraction or repulsion?

### The data

To provide answer to the questions above, two data sets will be used. They are:

- Childcare centre: The original data is in KML format. It has been converted into ESRI shapefile format.
- URA Master Plan Subzone 2014: It is in ESRI shapefile format.

Both data sets were downloaded from Data.gov.

## Installing and Loading the R packages

For the purpose of this study, five R packages will be used. They are:

- rgdal for importing geospatial data in GIS file format such as shapefile into R and save them as Spatial*DataFrame,
- maptools for converting Spatial* object into ppp object,
- raster for handling raster data in R,

- spatstat for performing Spatial Point Patterns Analysis such as kcross, Lcross, etc., and
- tmap for producing cartographic quality thematic maps.

```
packages = c ( 'rgdal' , 'maptools', 'raster' ,'spatstat', 'tmap'
)
for ( p in packages ) {
if ( ! require ( p , character.only = T
) ) {
install.packages( p )
}
library ( p ,character.only = T )
}
```

## Importing the Geospatial Data

The code chunk below uses **readOGR()** of **rgdal** package toimport both geospatial data files (i.e. shapefile) into R.

```
childcare <- readOGR ( dsn = "data/geospatial", layer= "CHILDCARE"
)
```

```
OGR data source with driver: ESRI Shapefile
Source: "D:\tskam\IS415\Hands-on_Ex\Hands-on_Ex05\data\geospatial", layer: "CHILDCARE"
with 1885 features
It has 1 fields
```

```
mpsz = readOGR ( dsn = "data/geospatial", layer=
"MP14_SUBZONE_WEB_PL")
```

```
OGR data source with driver: ESRI Shapefile
Source: "D:\tskam\IS415\Hands-on_Ex\Hands-on_Ex05\data\geospatial", layer: "MP14_SUBZONE_WEB_PL"
with 323 features
It has 15 fields
```

Since, *readOGR()* of **rgdal** package is used, the output R objectswill be in SpatialPointsDataframe and SpatialPolygonsDataframe classes respectively.

Next *str()* of Base R will be used to check the data type of `childcare` SpatialPointsDataFrame. This is necessary because the **marked** field must be in **factor** data type if its values are categorical.

```
str ( childcare)
```

```
Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
  ..@ data       :'data.frame': 1885 obs. of  1 variable:
  .. ..$ Type: chr [1:1885] "PT" "PT" "ST" "ST" ...
  ..@ coords.nrs : num(0)
  ..@ coords     : num [1:1885, 1:2] 11227 11783 11894 11961 12128 ...
```

```
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:2] "coords.x1" "coords.x2"
..@ bbox       : num [1:2, 1:2] 11227 25524 44936 49308
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : chr [1:2] "coords.x1" "coords.x2"
.. .. ..$ : chr [1:2] "min" "max"
..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
.. .. ..@ projargs: chr "+proj=tmerc +lat_0=1.36666666666667 +lon_0=103.833333333333 +k=1 +x_0=2800
.. .. ..$ comment: chr "PROJCRS[\"SVY21 / Singapore TM\",\n    BASEGEOGCRS[\"SVY21\",\n        DATU
```

The output above shows that `Type` field is in character data type and not in factor data type as required by spatstat package. Hence, the code chunk below will be used to convert `Type` field to factor data type.

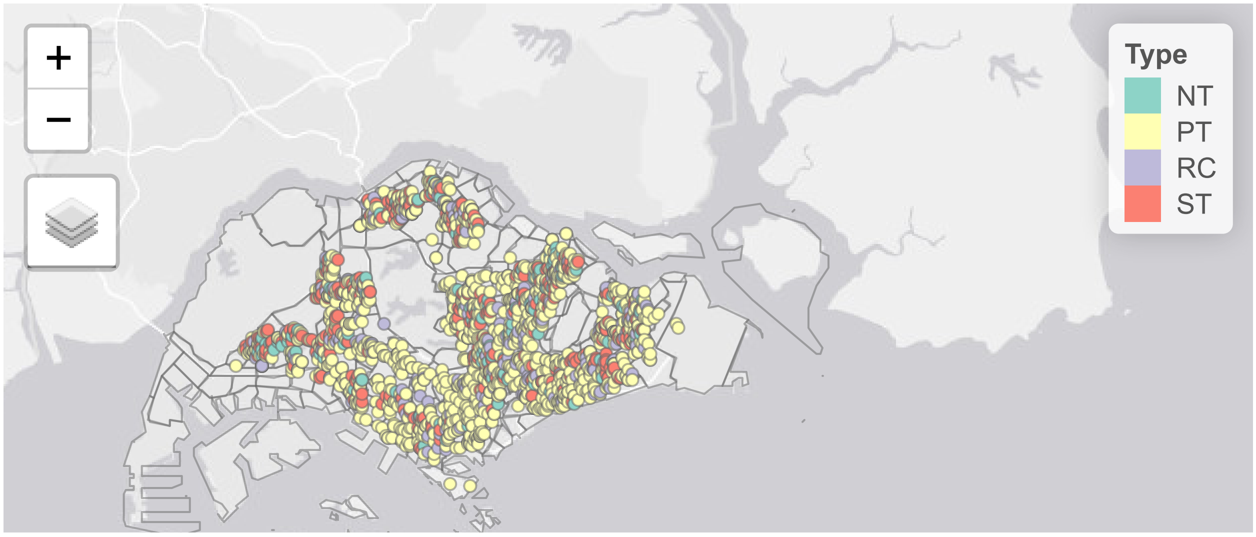
```
childcare@        data      $         Type       <-         as.factor(         childcare@
data      $         Type       )
```

> *DIY: Using the skill you learned from previous step, check to ensure that `Type` field is in **factor** data type now.*

## Mapping the geospatial layers

Next, let us take a quick look at the distribution of the geospatial data. In the code chunk below, mapping functions of **tmap** package is used. `tmap_mode("view")` is used to plot an interactive map by using leaflet api.

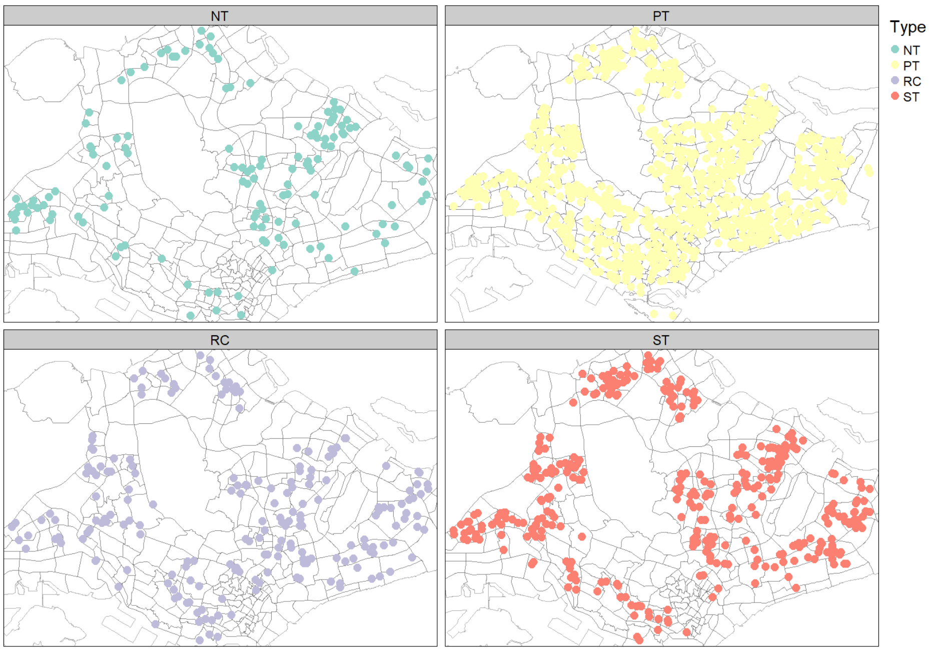
```
tmap_mode(         "view"    )
tm_shape (         mpsz      )          +
  tm_borders(         alpha =          0.5       )          +
  tmap_options(         check.and.fix =          TRUE      )          +
tm_shape (         childcare)          +
  tm_dots   (         col =          'Type'    , size =          0.02      )
```



```
tmap_mode(        "plot"    )
```

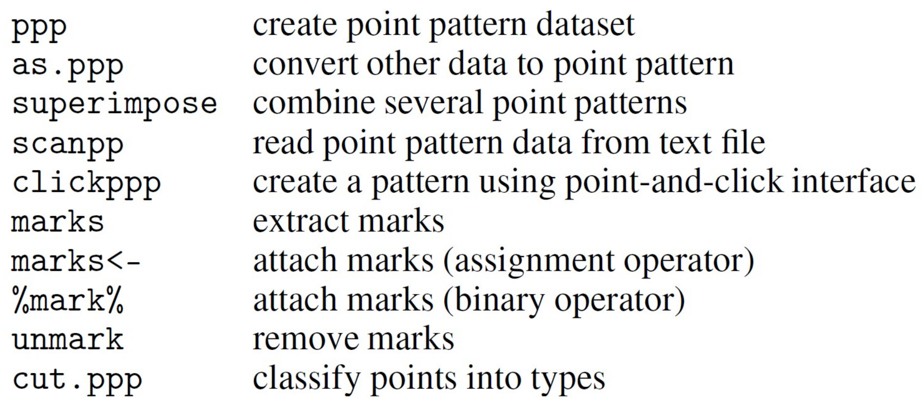
Alternatively, we can use the code chunk below to create four small point maps by using *tm_facets()* of **tmap** pckage.

```
tm_shape (        mpsz      )         +
  tm_borders(        alpha =         0.5       )         +
tm_shape (        childcare)         +
  tm_dots   (        col =          'Type'    ,
          size =         0.5       )         +
tm_facets(        by=          "Type"    )
```



## Spatial Data Wrangling

Table below shows **spatstat** functions for wrangling geospatial data. It is advisable for students to familiarise yourself with each of them before you continue.



| | |
|---|---|
| `ppp` | create point pattern dataset |
| `as.ppp` | convert other data to point pattern |
| `superimpose` | combine several point patterns |
| `scanpp` | read point pattern data from text file |
| `clickppp` | create a pattern using point-and-click interface |
| `marks` | extract marks |
| `marks<-` | attach marks (assignment operator) |
| `%mark%` | attach marks (binary operator) |
| `unmark` | remove marks |
| `cut.ppp` | classify points into types |

## Converting the SpatialPointsDataFrame into ppp format

The code chunk below uses *as.(x, "ppp")* or *as.ppp(x)* of **maptools** package to convert an object *x* of class SpatialPointsDataFrame to a spatial point pattern in **spatstat**. In this conversion, the additional field in *x* data frame will become the marks of the point pattern z.

```
childcare_ppp <- as(childcare, "ppp")
plot(childcare_ppp)
```

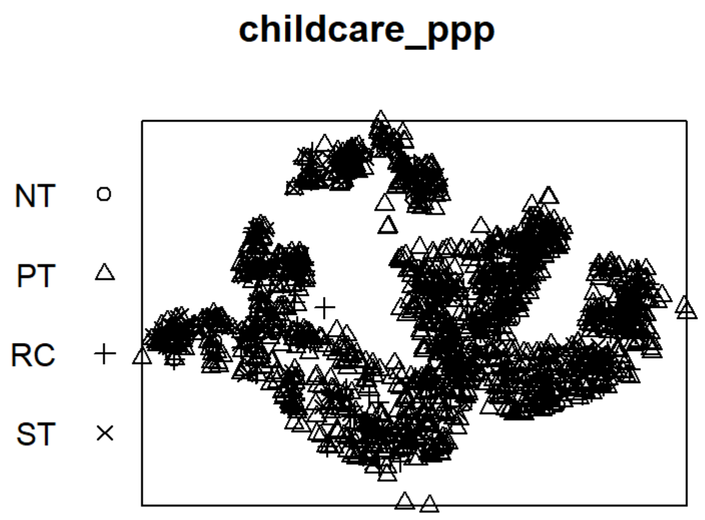


Figure above reveals that there are four sub-types in the marks list. They are: NT, PT, RC and ST.

To examine the summary statistics of this spatial object, summary() of Base R will be used as shown in the code chunk below.

```
summary(childcare_ppp)
```

```
Marked planar point pattern:  1885 points
Average intensity 2.351049e-06 points per square unit

*Pattern contains duplicated points*

Coordinates are given to 3 decimal places
i.e. rounded to the nearest multiple of 0.001 units

Multitype:
   frequency proportion    intensity
NT       138 0.07320955 1.721193e-07
PT      1183 0.62758620 1.475486e-06
RC       200 0.10610080 2.494482e-07
ST       364 0.19310340 4.539957e-07

Window: rectangle = [11226.55, 44936.07] x [25523.51, 49308.17] units
                    (33710 x 23780 units)
Window area = 801770000 square units
```

The report above reveals that PT is the largest childcare operator in Singapore with a market share of 63%. This is followed by ST, RC and NT.

It is also important to node that the spatial point object contains duplicated points. The quality of our analysis will be compromised if we failed to resolve this data issue.

## Avoiding duplicated spatial point event by using jittering method

The code chunk below resolves the duplicated spatial point events issue by using the jittering approach.

```
childcare_ppp_jit <- rjitter(childcare_ppp, retry=TRUE, nsim
=1, drop=TRUE)
```

Let us check the output to ensure that there is no more duplicated spatial point events in the data.

```
any(duplicated(childcare_ppp_jit))
```

```
[1] FALSE
```

The output shows that the duplicated points issue has been resolved.

## Creating *owin*

When analysing spatial point patterns, it is a good practice to confine the analysis within a geographical area like Singapore boundary. In **spatstat**, an object called ***owin*** is specially designed to represent this polygonal region.

Before we going ahead to create owin object, however, it is important to understand the geography of the study area. Figure below reveals that the distribution of settlements in our country are constrained by natural such as central water catchment and western reserved area and strategic location such as airports.

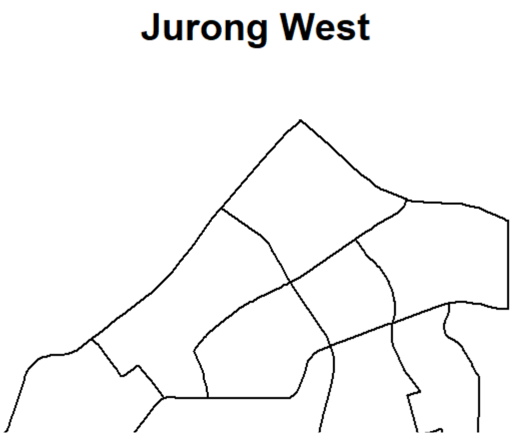
Johor Bahru
Pasir Gudang
Iskandar Puteri
Singapore
Southwest
Southeast
Jurong Island
Leaflet | © OpenStreetMap contributors

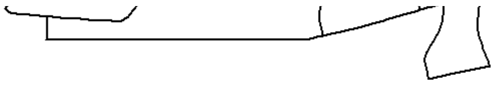
In view of this, it is wiser for us to narrow down the study area by more appropriate geographical area such as by planning area.

## Extracting study area

For the purpose of this study, we will focus of Jurong West planning area. The code chunk below will be used to extract the target planning areas.

```
jw = mpsz[mpsz@data$PLN_AREA_N == "JURONG WEST",]
plot(jw, main = "Jurong West")
```





## Converting the spatial point data frame into generic sp format

Next, we will convert these SpatialPolygonsDataFrame layers into generic spatialpolygons layers by using *as.SpatialPolygons.tess(x)* of **maptools** package.

```
jw_sp     =         as       (        jw       , "SpatialPolygons")
str      (         jw_sp     )
```

```
Formal class 'SpatialPolygons' [package "sp"] with 4 slots
  ..@ polygons   :List of 9
  .. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
  .. .. .. ..@ Polygons :List of 1
  .. .. .. .. ..$ :Formal class 'Polygon' [package "sp"] with 5 slots
  .. .. .. .. .. .. ..@ labpt  : num [1:2] 12930 34858
  .. .. .. .. .. .. ..@ area   : num 2098176
  .. .. .. .. .. .. ..@ hole   : logi FALSE
  .. .. .. .. .. .. ..@ ringDir: int 1
  .. .. .. .. .. .. ..@ coords : num [1:105, 1:2] 14212 14156 14122 14085 14067 ...
  .. .. .. ..@ plotOrder: int 1
  .. .. .. ..@ labpt    : num [1:2] 12930 34858
  .. .. .. ..@ ID       : chr "142"
  .. .. .. ..@ area     : num 2098176
  .. .. .. ..$ comment: chr "0"
  .. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
  .. .. .. ..@ Polygons :List of 1
  .. .. .. .. ..$ :Formal class 'Polygon' [package "sp"] with 5 slots
  .. .. .. .. .. .. ..@ labpt  : num [1:2] 11290 35200
  .. .. .. .. .. .. ..@ area   : num 1524551
  .. .. .. .. .. .. ..@ hole   : logi FALSE
  .. .. .. .. .. .. ..@ ringDir: int 1
  .. .. .. .. .. .. ..@ coords : num [1:93, 1:2] 11769 11774 11901 11925 11934 ...
  .. .. .. ..@ plotOrder: int 1
  .. .. .. ..@ labpt    : num [1:2] 11290 35200
  .. .. .. ..@ ID       : chr "143"
  .. .. .. ..@ area     : num 1524551
  .. .. .. ..$ comment: chr "0"
  .. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
  .. .. .. ..@ Polygons :List of 1
  .. .. .. .. ..$ :Formal class 'Polygon' [package "sp"] with 5 slots
  .. .. .. .. .. .. ..@ labpt  : num [1:2] 15522 35189
  .. .. .. .. .. .. ..@ area   : num 1484296
  .. .. .. .. .. .. ..@ hole   : logi FALSE
  .. .. .. .. .. .. ..@ ringDir: int 1
  .. .. .. .. .. .. ..@ coords : num [1:95, 1:2] 15941 15935 15924 15922 15921 ...
  .. .. .. ..@ plotOrder: int 1
  .. .. .. ..@ labpt    : num [1:2] 15522 35189
  .. .. .. ..@ ID       : chr "145"
  .. .. .. ..@ area     : num 1484296
  .. .. .. ..$ comment: chr "0"
  .. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
  .. .. .. ..@ Polygons :List of 1
  .. .. .. .. ..$ :Formal class 'Polygon' [package "sp"] with 5 slots
  .. .. .. .. .. .. ..@ labpt  : num [1:2] 13397 35812
  .. .. .. .. .. .. ..@ area   : num 1404537
  .. .. .. .. .. .. ..@ hole   : logi FALSE
  .. .. .. .. .. .. ..@ ringDir: int 1
  .. .. .. .. .. .. ..@ coords : num [1:65, 1:2] 13673 13657 12785 12776 12776 ...
  .. .. .. ..@ plotOrder: int 1
  .. .. .. ..@ labpt    : num [1:2] 13397 35812
  .. .. .. ..@ ID       : chr "151"
  .. .. .. ..@ area     : num 1404537
  .. .. .. ..$ comment: chr "0"
  .. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
  .. .. .. ..@ Polygons :List of 1
  .. .. .. .. ..$ :Formal class 'Polygon' [package "sp"] with 5 slots
  .. .. .. .. .. .. ..@ labpt  : num [1:2] 14632 35241
  .. .. .. .. .. .. ..@ area   : num 1287950
  .. .. .. .. .. .. ..@ hole   : logi FALSE
  .. .. .. .. .. .. ..@ ringDir: int 1
  .. .. .. .. .. .. ..@ coords : num [1:57, 1:2] 14106 14097 14085 14070 14065 ...
  .. .. .. ..@ plotOrder: int 1
  .. .. .. ..@ labpt    : num [1:2] 14632 35241
  .. .. .. ..@ ID       : chr "157"
  .. .. .. ..@ area     : num 1287950
  .. .. .. ..$ comment: chr "0"
  .. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
  .. .. .. ..@ Polygons :List of 1
  .. .. .. .. ..$ :Formal class 'Polygon' [package "sp"] with 5 slots
  .. .. .. .. .. .. ..@ labpt  : num [1:2] 14404 36445
  .. .. .. .. .. .. ..@ area   : num 906317
  .. .. .. .. .. .. ..@ hole   : logi FALSE
  .. .. .. .. .. .. ..@ ringDir: int 1
  .. .. .. .. .. .. ..@ coords : num [1:41, 1:2] 14220 14209 14157 14090 13766 ...
  .. .. .. ..@ plotOrder: int 1
  .. .. .. ..@ labpt    : num [1:2] 14404 36445
  .. .. .. ..@ ID       : chr "171"
  .. .. .. ..@ area     : num 906317

  .. .. .. ..$ comment: chr "0"
  .. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
  .. .. .. ..@ Polygons :List of 1
  .. .. .. .. ..$ :Formal class 'Polygon' [package "sp"] with 5 slots
  .. .. .. .. .. .. ..@ labpt  : num [1:2] 15508 36890
  .. .. .. .. .. .. ..@ area   : num 1793464
  .. .. .. .. .. .. ..@ hole   : logi FALSE
  .. .. .. .. .. .. ..@ ringDir: int 1
  .. .. .. .. .. .. ..@ coords : num [1:173, 1:2] 16296 16297 16297 16297 16297 ...
  .. .. .. ..@ plotOrder: int 1
  .. .. .. ..@ labpt    : num [1:2] 15508 36890
  .. .. .. ..@ ID       : chr "176"
  .. .. .. ..@ area     : num 1793464
  .. .. .. ..$ comment: chr "0"
  .. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
  .. .. .. ..@ Polygons :List of 1
  .. .. .. .. ..$ :Formal class 'Polygon' [package "sp"] with 5 slots
  .. .. .. .. .. .. ..@ labpt  : num [1:2] 13954 37533
  .. .. .. .. .. .. ..@ area   : num 1974943
  .. .. .. .. .. .. ..@ hole   : logi FALSE
  .. .. .. .. .. .. ..@ ringDir: int 1
  .. .. .. .. .. .. ..@ coords : num [1:44, 1:2] 13849 13735 13710 13616 13596 ...
  .. .. .. ..@ plotOrder: int 1
  .. .. .. ..@ labpt    : num [1:2] 13954 37533
  .. .. .. ..@ ID       : chr "186"
  .. .. .. ..@ area     : num 1974943
  .. .. .. ..$ comment: chr "0"
  .. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
  .. .. .. ..@ Polygons :List of 1
  .. .. .. .. ..$ :Formal class 'Polygon' [package "sp"] with 5 slots
  .. .. .. .. .. .. ..@ labpt  : num [1:2] 12593 36300
  .. .. .. .. .. .. ..@ area   : num 2206305
  .. .. .. .. .. .. ..@ hole   : logi FALSE
  .. .. .. .. .. .. ..@ ringDir: int 1
  .. .. .. .. .. .. ..@ coords : num [1:106, 1:2] 12671 12711 12723 12726 12732 ...
  .. .. .. ..@ plotOrder: int 1
  .. .. .. ..@ labpt    : num [1:2] 12593 36300
  .. .. .. ..@ ID       : chr "192"
  .. .. .. ..@ area     : num 2206305
  .. .. .. ..$ comment: chr "0"
  ..@ plotOrder  : int [1:9] 9 1 8 7 2 3 4 5 6
  ..@ bbox       : num [1:2, 1:2] 10373 33982 16297 38489
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2] "x" "y"
  .. .. ..$ : chr [1:2] "min" "max"
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
  .. .. ..@ projargs: chr "+proj=tmerc +lat_0=1.36666666666667 +lon_0=103.833333333333 +k=1 +x_0=2800
  .. .. ..$ comment: chr "PROJCRS[\"SVY21\",\n    BASEGEOGCRS[\"SVY21[WGS84]\",\n        DATUM[\"Worl
```

> *Best Practice: It is always recommended to review the structure of the output object by using either the UI of RStudio or str() function.*

## Creating *owin* object

Now, we will convert these SpatialPolygons objects into owin objects that is required by **spatstat**.

```
jw_owin = as(jw_sp, "owin")
str(jw_owin)
```

```
List of 5
 $ type  : chr "polygonal"
 $ xrange: num [1:2] 10373 16297
 $ yrange: num [1:2] 33982 38489
 $ bdry  :List of 9
  ..$ :List of 2
  .. ..$ x: num [1:104] 14212 14240 14250 14250 14243 ...
  .. ..$ y: num [1:104] 35397 35520 35596 35689 35740 ...
  ..$ :List of 2
  .. ..$ x: num [1:92] 11769 11764 11760 11756 11688 ...
  .. ..$ y: num [1:92] 35484 35486 35489 35493 35582 ...
  ..$ :List of 2
  .. ..$ x: num [1:94] 15941 15944 15944 15945 15950 ...
  .. ..$ y: num [1:94] 34916 34979 34995 35012 35161 ...
  ..$ :List of 2
  .. ..$ x: num [1:64] 13673 13689 13704 13733 13747 ...
  .. ..$ y: num [1:64] 35226 35229 35233 35245 35252 ...
  ..$ :List of 2
  .. ..$ x: num [1:56] 14106 14195 14305 14329 14353 ...
  .. ..$ y: num [1:56] 34492 34512 34537 34542 34548 ...
  ..$ :List of 2
  .. ..$ x: num [1:40] 14220 14444 14878 14938 14945 ...
  .. ..$ y: num [1:40] 35850 35933 36095 36113 36145 ...
  ..$ :List of 2
  .. ..$ x: num [1:172] 16296 16296 16296 16296 16296 ...
  .. ..$ y: num [1:172] 36694 36695 36696 36697 36698 ...
  ..$ :List of 2
  .. ..$ x: num [1:43] 13849 14496 14639 14684 14723 ...
  .. ..$ y: num [1:43] 36652 37092 37190 37216 37242 ...
  ..$ :List of 2
  .. ..$ x: num [1:105] 12671 12648 12605 12637 12678 ...
  .. ..$ y: num [1:105] 35650 35702 35777 35827 35882 ...
 $ units :List of 3
  ..$ singular  : chr "unit"
  ..$ plural    : chr "units"
  ..$ multiplier: num 1
  ..- attr(*, "class")= chr "unitname"
 - attr(*, "class")= chr "owin"
```

## Combining childcare points and the study area

By using the code chunk below, we are able to extract childcare that is within the specific region to do our analysis later on.

```
childcare_jw_ppp = childcare_ppp_jit[jw_owin]
```

Next, *summary()* is used to reveal the data object as shown in the code chunk below.

```
summary(childcare_jw_ppp)
```

```
Marked planar point pattern:  114 points
Average intensity 7.765382e-06 points per square unit

Coordinates are given to 3 decimal places
i.e. rounded to the nearest multiple of 0.001 units

Multitype:
   frequency proportion    intensity
NT        16  0.1403509 1.089878e-06
PT        58  0.5087719 3.950808e-06
RC        12  0.1052632 8.174086e-07
ST        28  0.2456140 1.907287e-06

Window: polygonal boundary
9 separate polygons (no holes)
           vertices    area relative.area
polygon 1       104 2098180        0.1430
polygon 2        92 1524550        0.1040
polygon 3        94 1484300        0.1010
polygon 4        64 1404540        0.0957
polygon 5        56 1287950        0.0877
polygon 6        40  906317        0.0617
polygon 7       172 1793460        0.1220
polygon 8        43 1974940        0.1350
polygon 9       105 2206310        0.1500
enclosing rectangle: [10373.179, 16297.184] x [33981.5, 38488.61]
units
                     (5924 x 4507 units)
Window area = 14680500 square units
Fraction of frame area: 0.55
```

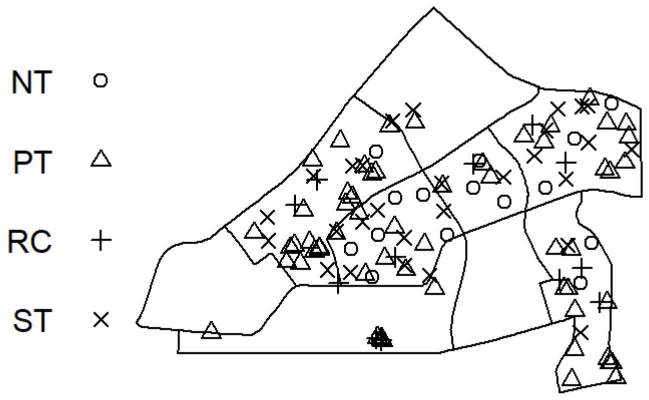
> *DIY: By referring to previous discussion, describe the content of the output.*

## Plotting childcare points and the study area

Lastly, we will plot the combined childcare point and the study area to ensure that the spatial point events are indeed contained within the study area.

```
plot(childcare_jw_ppp)
```

**childcare_jw_ppp**

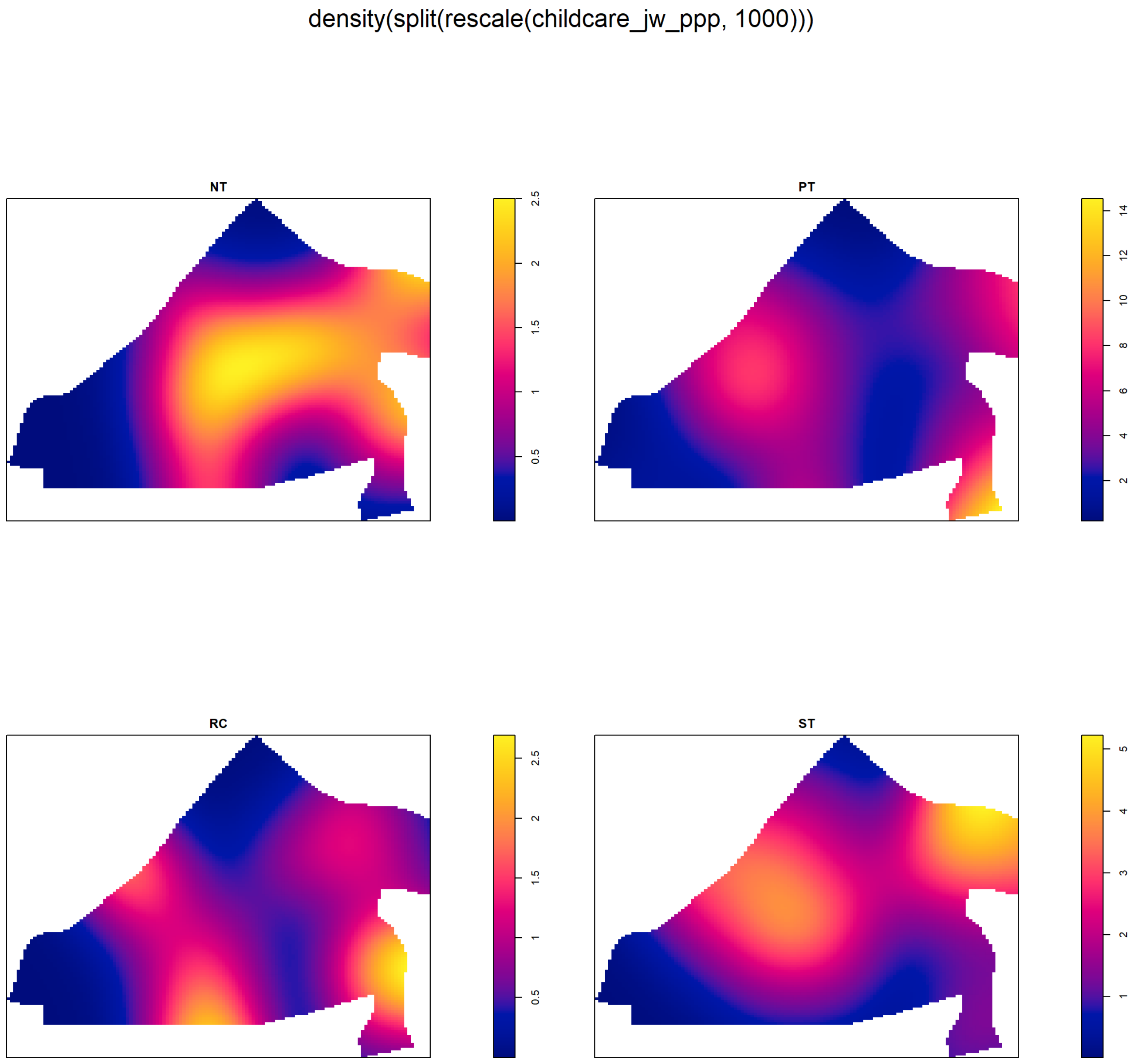


# Analysing Marked Point Patterns

## First-order Spatial Point Patterns Analysis

In the code chunk below, *density()* of **spatstat** package is used to compute the kernel density objects. Then, **plot()** is used to plot the output kernel density objects derived. Instead of writing them in two seperate lines, the code chunk below shows how they can be combined into one single line code chunk. However, for clarity purpose, it is nothing wrong if you prefer to keep them as two seperate lines of code.

```
plot      (         density   (         split     (         rescale   (         childcare_jw_ppp,
1000      )         )         )         )
```



> *Question: Can you recall what is the purpose of rescale() and why it is used in our case?*

> *DIY: What observations can you draw from the figure above?*

Next, *intensity()* of spatstat package is used to reveal the density of childcare centres by operators as shown the code chunk below.

```
intensity(        rescale  (        childcare_jw_ppp, 1000     )         )
```

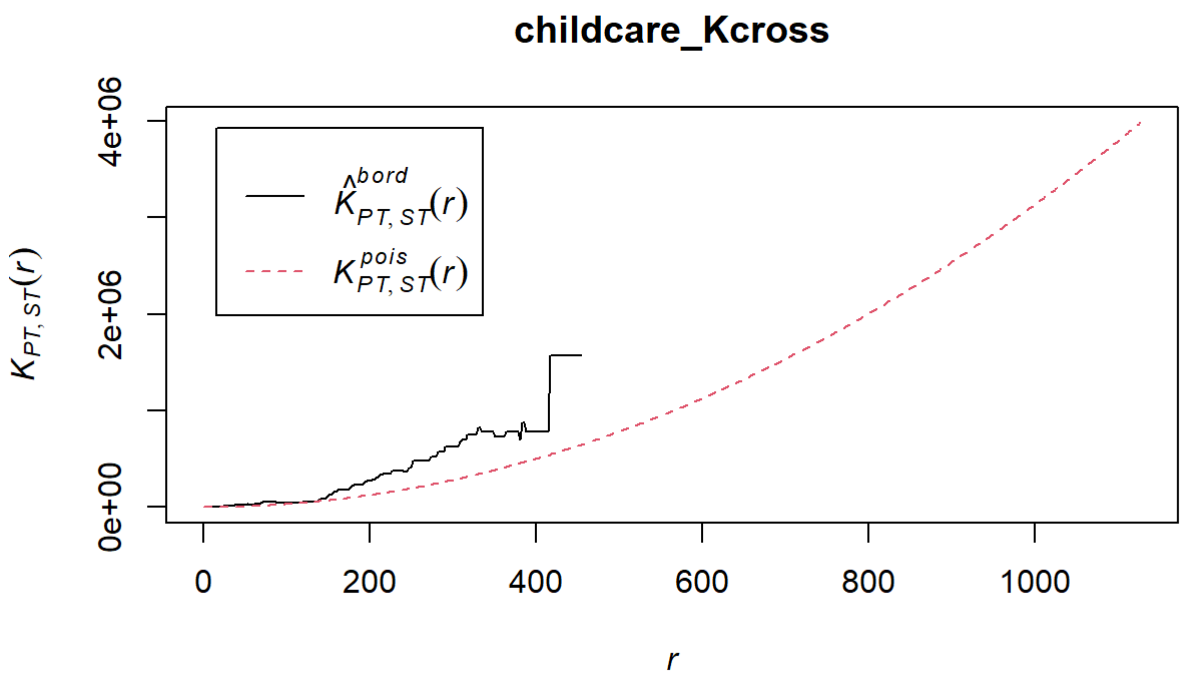
```
      NT        PT        RC        ST 
1.0898782 3.9508083 0.8174086 1.9072868 
```

The output reveals that childcare centres operate by PT has the highest density of 3.95 units per km square. This is followed by 1.91 units per km square, 1.09 unit per km square and 0.82 unit per km square for ST, NT and RC respectively.

## Second-order Multi-tpye Point Patterns Analysis: Cross K-Function

Now, we will analyse the relationship of PT and ST by using *Kcross()* of **spatstat** package.

```
childcare_Kcross <-        Kcross    (        childcare_jw_ppp,
                           i=        "PT"     , j=        "ST"      ,
                           correction=        'border' )
plot     (        childcare_Kcross)
```



The plot above reveals that there is a sign that the marked spatial point events are not independent spatially. However, a hypothesis test is required to confirm the observation statistically.

## Performing CSR testing on the Cross K-Function

The hypothesis and test are as follows:

Ho = The distribution of ST childcare centres and NT chilcare centres are spatially independent.

H1= The distribution of ST childcare centres and NT chilcare centres are NOT at spatially independent.

The null hypothesis will be rejected if p-value is smaller than alpha value of 0.001 (i.e. at 99.9% confident interval).

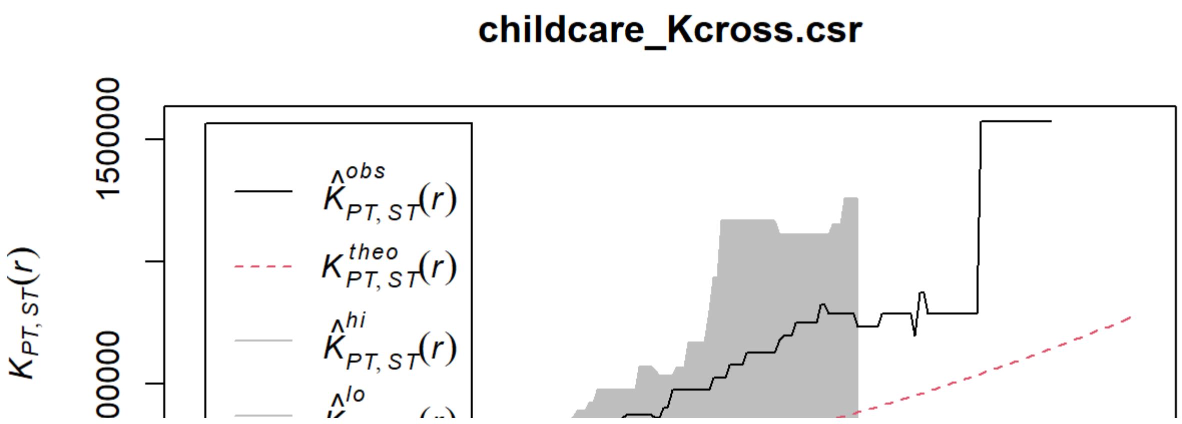
In order to perform the CSR test, the *envelope()* of **spatstat** package will be used.

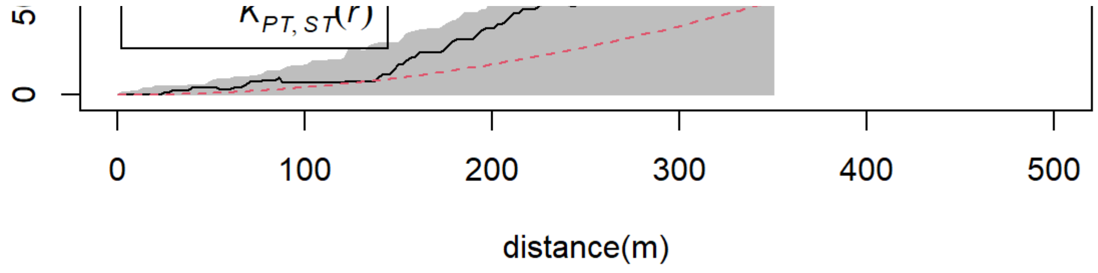
```
childcare_Kcross.csr <- envelope(childcare_jw_ppp, Kcross, i="PT", j="ST", correction='border', nsim=999)
```

```
Generating 999 simulations of CSR  ...
1, 2, 3, ......10.........20.........30.........40.........50.........60
.........70.........80.........90.........100.........110.........120
.........130.........140.........150.........160.........170.........180
.........190.........200.........210.........220.........230.........240
.........250.........260.........270.........280.........290.........300
.........310.........320.........330.........340.........350.........360
.........370.........380.........390.........400.........410.........420
.........430.........440.........450.........460.........470.........480
.........490.........500.........510.........520.........530.........540
.........550.........560.........570.........580.........590.........600
.........610.........620.........630.........640.........650.........660
.........670.........680.........690.........700.........710.........720
.........730.........740.........750.........760.........770.........780
.........790.........800.........810.........820.........830.........840
.........850.........860.........870.........880.........890.........900
.........910.........920.........930.........940.........950.........960
.........970.........980.........990........ 999.

Done.
```

```
plot(childcare_Kcross.csr, xlab="distance(m)", xlim=c(0,500))
```





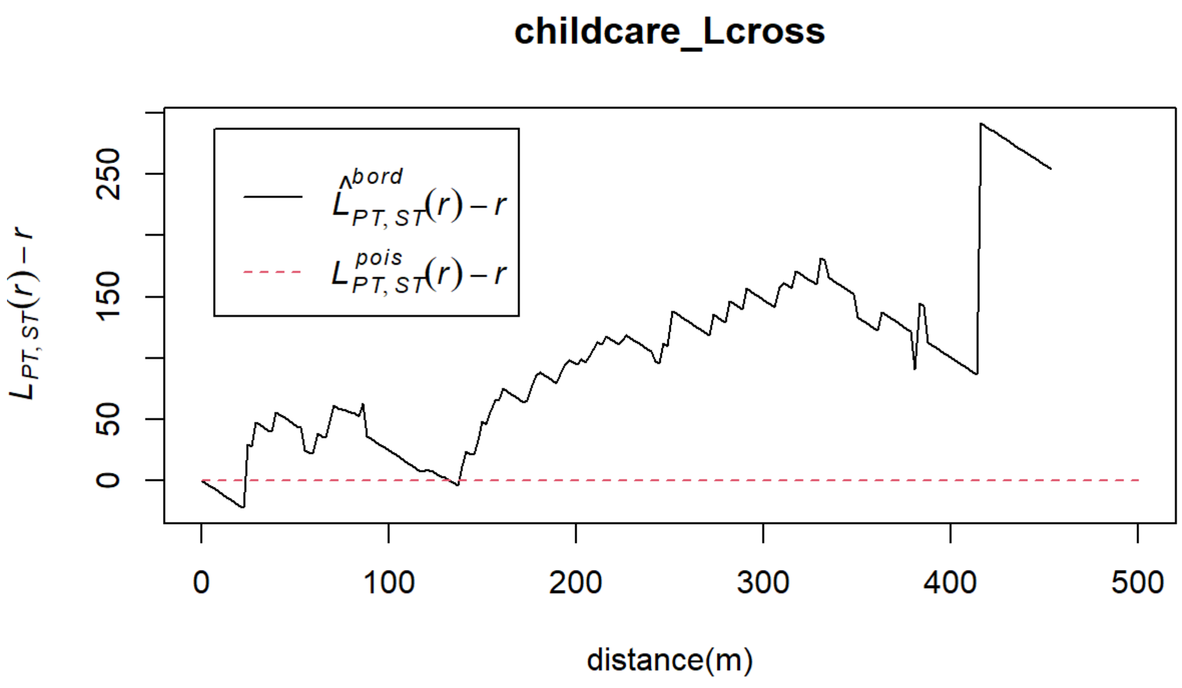
> *Question: Why nsim=999 is used?*

The plot above reveals that the are signs that the distribution of childcare centres operate by NT and ST are not independent spatially. Unfortunately, we failed to reject the null hypothesis because the empirical k-cross line is within the envelop of the 99.9% confident interval.

## Second-order Multi-tpye Point Patterns Analysis: Cross L-Function

In the code chunk below, *Lcross()* of **spatstat** package is used to compute Cross L-function.

```
childcare_Lcross <-         Lcross    (         childcare_jw_ppp, i=         "PT"     , j=
"ST"      , correction=         'border' )
plot      (         childcare_Lcross, .         -         r         ~         r         ,
     xlab =         "distance(m)",
     xlim=         c         (         0         , 500      )         )
```



> *DIY: With reference to discussion in the earlier section, what observation(s) can you draw from the plot above?*

## Performing CSR testing on the Cross L-Function

> *DIY: With reference to the example given in previous section, define the hypothesis null, hypothesis alternative and rejection criterion.*

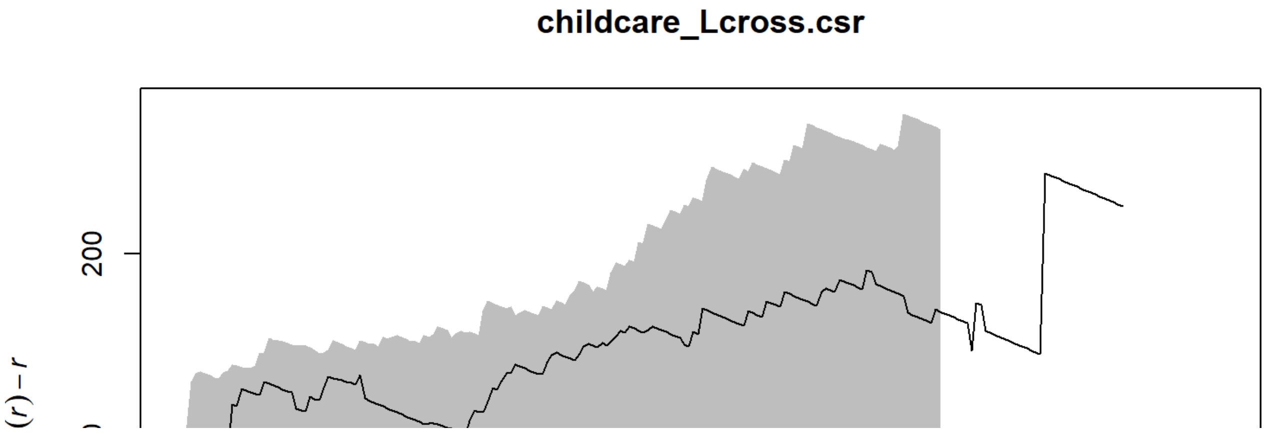
Similar to Cross-K-Function, we can perform the CSR test by using *envelope()* of **spatstat** package will be used.

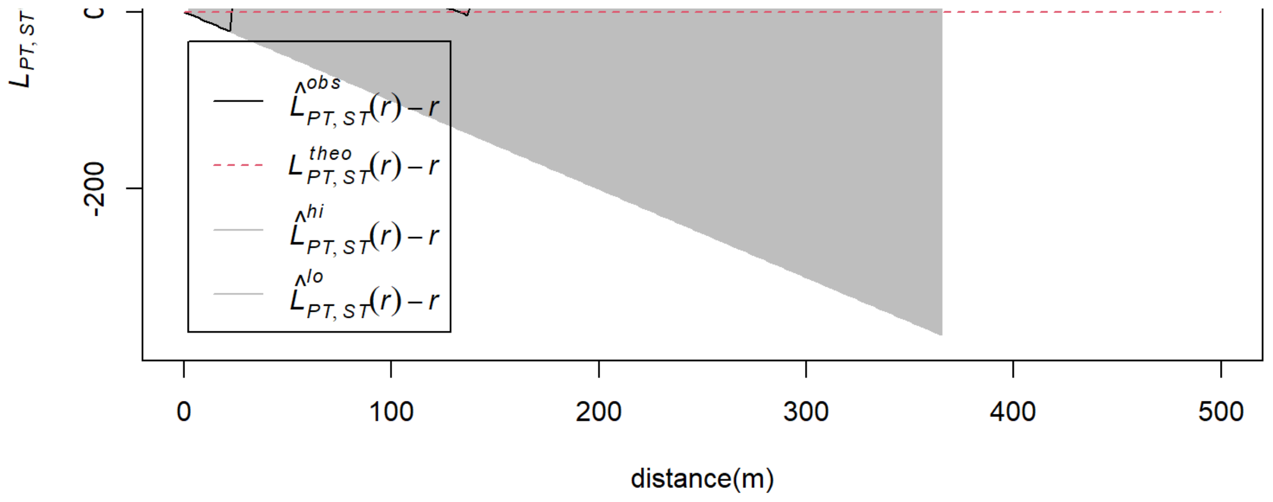
```
childcare_Lcross.csr <- envelope(childcare_jw_ppp, Lcross, i="PT", j="ST", correction='border', nsim=999)
```

```
Generating 999 simulations of CSR ...
1, 2, 3, ......10.........20.........30.........40.........50.........60
.........70.........80.........90.........100.........110.........120
.........130.........140.........150.........160.........170.........180
.........190.........200.........210.........220.........230.........240
.........250.........260.........270.........280.........290.........300
.........310.........320.........330.........340.........350.........360
.........370.........380.........390.........400.........410.........420
.........430.........440.........450.........460.........470.........480
.........490.........500.........510.........520.........530.........540
.........550.........560.........570.........580.........590.........600
.........610.........620.........630.........640.........650.........660
.........670.........680.........690.........700.........710.........720
.........730.........740.........750.........760.........770.........780
.........790.........800.........810.........820.........830.........840
.........850.........860.........870.........880.........890.........900
.........910.........920.........930.........940.........950.........960
.........970.........980.........990........ 999.

Done.
```

```
plot(childcare_Lcross.csr, . - r ~ r, xlab="distance(m)", xlim=c(0,500))
```





> *DIY: Intepret the analysis result and draw conclusion with reference to the statistical testing result.*