CHALMERS

GÖTEBORGS UNIVERSITET

Parameterskattning av spatiala klusterprocesser med en inblick i nervdata

Parameter estimation of spatial cluster processes with a view towards nerve data

Examensarbete för kandidatexamen i matematik vid Göteborgs universitet
Kandidatarbete inom civilingenjörsutbildningen vid Chalmers

Hanna Ekelund
Elsa Helle
Lirim Kadriu
Casper Opperud
Hanna Skytt
Samuel Thorén

Institutionen för Matematiska vetenskaper
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2020

Parameterskattning av spatiala klusterprocesser med en inblick i nervdata

Examensarbete för kandidatexamen i matematik inom Matematikprogrammet vid Göteborgs universitet

Elsa Helle

Kandidatarbete i matematik inom civilingenjörsprogrammet Teknisk matematik vid Chalmers

Hanna Ekelund Lirim Kadriu Casper Opperud Hanna Skytt Samuel Thorén

Handledare: Aila Särkkä

Institutionen för Matematiska vetenskaper
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2020

Förord

Vi vill främst tacka vår handledare Aila Särkkä vars handledning och insikter har varit ovärderliga. Vi vill även tacka Esmée Berger, Christoffer Ekgrim, Julia Jansson och David Larsson för gott samarbete när det gäller feedback på varandras rapporter. Vi vill dessutom tacka varandra för gott samarbete.

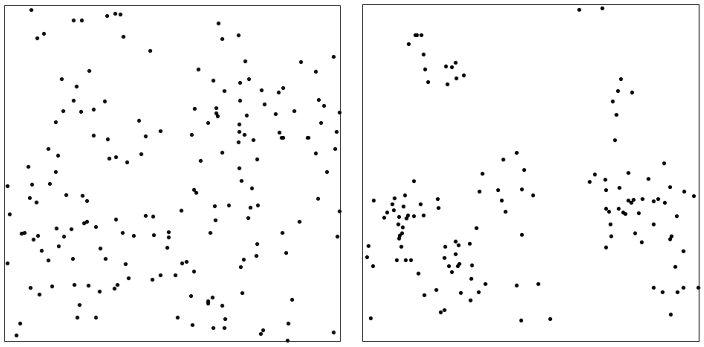
Gruppen har löpande fört loggbok där allas bidrag till projektet loggats och där alla problem som vi stött på under arbetets gång har dokumenterats. Nedan i tabellen specificeras vilka medlemmar som tillskrivits som huvudförfattare till de olika avsnitten av projektet.

Förutom det som syns som resultat i rapporten har det även spenderats många timmar åt litteraturstudier, kodning i R, illustrerande av figurer, samt korrekturläsning och feedback till varandras text. Alla i gruppen har ägnat sig åt litteraturstudier för att få en förståelse för projektets teoretiska bakgrund. Samuel läste på lite extra om Ripleys funktioner samt Monte Carlo-simuleringnar, Elsa läste om parkorrelationsfunktionen och Hanna S läste om Minimum Contrast-metoden. Största delen av simuleringsstudierna samt övrig kod i R har skapats av Lirim och Casper med undantag för violindiagrammen i appendix D.3 som skapats av Samuel. Det är också de som genererat alla figurer i rapporten med undantag för figur 4, figur 5, 6 samt figur 20 som illustrerats av Elsa. Lirim har även deltagit aktivt i avsnitten teori och analys av nervdata samt engagerat sig i den generella utformningen av rapporten. Under de sista dagarna lästes rapporten genom noggrant av Hanna S och Elsa där språket var i fokus. Utöver arbete kring rapporten så har Hanna S och Hanna E även varit på flera föreläsningar om hur man skriver vetenskaplig text, de har använt sina kunskaper från dessa och delat med sig till övriga gruppmedlemmar.

Avsnitt	**Skrivet av**	**Övrig information**
Populärvetenskaplig presentation	Hanna E	Diskuterat med Elsa, Figur: Lirim
Sammanfattning och abstract	Lirim	
1 Inledning	Elsa	
1.1	Hanna S	
1.2	Casper	
1.3	Samuel	
2 Teori	Lirim	
2.1	Lirim, Samuel	
2.1.1	Lirim	
2.1.2	Elsa, Lirim	Figur: Elsa
2.2	Samuel	
2.2.1	Samuel, Elsa	Figur: Elsa
2.2.2	Hanna S	
2.3	Elsa	
2.3.1	Elsa, Samuel	Figur: Elsa
2.3.2	Hanna E	
2.4	Casper	
2.5	Hanna S	
2.6	Samuel, Lirim	
2.7	Lirim, Casper	
2.7.1	Lirim	
2.7.2	Casper	
3 Simuleringsstudie	Lirim	
3.1	Lirim	
3.2	Lirim	
4 Analys av nervdata	Lirim	
4.1	Lirim	
4.2	Lirim, Elsa	
4.3	Lirim, Elsa	
4.4	Lirim, Elsa	
4.4.1	Elsa	Figur och figurtext: Lirim
4.4.2	Lirim	
4.5	Elsa	Figur och figurtext: Lirim
5 Sammanfattning	Samuel, Casper, Lirim	Figur: Elsa
A Härledningar		
A.1	Hanna S	
A.2	Samuel	
B Simulering		
B.1	Lirim, Elsa	
B.2	Lirim	
C Tabeller och resultat	Lirim	
D Figurer		
D.1	Lirim, Samuel	Text: Samuel
D.2	Lirim, Samuel	Text: Samuel
D.3	Samuel	
E Kod		
E.1	Lirim	
E.2	Lirim	
E.3	Lirim	
E.4	Casper	
E.5	Lirim	
E.6	Lirim	
E.7	Casper	
E.8	Samuel	
E.9	Lirim	

Populärvetenskaplig presentation

Spatiala punktprocesser beskriver hur punkter i ett punktmönster förhåller sig till varandra. För att kunna beskriva ett punktmönster matematiskt används modeller för punktprocesser. En modell använder sig av två olika sorters punkter, föräldra- och dotterpunkter. Punktprocessmodellen placerar först ut föräldrapunkter i rummet och sedan ett antal dotterpunkter runt varje föräldrapunkt. Både antalet föräldrapunkter och antalet dotterpunkter per föräldrapunkt beskrivs med funktioner som beskriver en fördelning, även dotterpunkternas position förhållande till föräldrarna beskrivs med dessa funktioner. Detta gör det möjligt att beskriva många olika punktmönster med en och samma modell. Av det följer att olika punktmönster kan se olika ut och därmed har även vissa mönster fått namn som beskriver dem. I figur 1(a) ser vi ett punktmönster som är helt slumpmässigt, det vill säga att dotterpunkterna är helt oberoende och likformigt fördelade i hela planet. Det är värt att notera att det endast är dotterpunkter som visas i figuren. Ett annat punktmönster är klustrat som vi kan se i figur 1(b). Punkterna i ett klustrat punktmönster är samlade i grupper. I vissa fall kan det vara svårt att avgöra om ett punktmönster är klustrat eller slumpmässigt med blotta ögat, men med statistik visas hur klustrat ett punktmönster är.



(a) *Slumpmässigt punktmönster.* (b) *Klustrat punktmönster.*

Figur 1: *Olika punktmönster på ett begränsat område i planet* ($\mathbb{R}^2$*).*

Det är intressant att undersöka punktmönster då de kan modellera många fenomen i vår vardag. Till exempel går det att beskriva stjärnornas position i himlen eller trädens position i en skog. I detta arbete har vi använt punktprocesser för att beskriva positionen för nervändar i överhuden för friska individer och individer med sjukdomen diabetesneuropati.

Diabetespatienter riskerar att utveckla en rad olika komplikationer som bland annat diabetesneuropati, vilket först ofta visar sig i fötter och lår i form av domningar och smärtor, men kan sprida sig till andra delar av kroppen och leda till nedsatt känsel. I nuläget upptäcks neuropati sent i förloppet, när symptom visat sig ett tag. Den sena upptäckten beror på att neuropati utvecklas långsamt vilket leder till att patienterna är sjuka långt innan de upptäcker symptomen. För att kunna behandla neuropati krävs att patienten diagnoseras så tidigt som möjligt.

En skillnad som observerats i tidigare studier är att patienter med diabetesneuropati har färre nervtrådar i överhuden, samt att nerverna ej försvinner slumpmässigt. Nerverna försvinner så att resterande nerver formar kluster. Nervmönstrerna kan modelleras som en spatial punktprocess, där nerverna i överhuden representeras av dotterpunkter och deras rötter som föräldrapunkter. Detta punktmönster tenderar att vara klustrat för alla individer, oberoende av neuropati, men mönstren hos patienter med neuropati är mer klustrade än hos friska individer. Med hjälp av den så kallade Ripleys K-funktion och parkorrelationsfunktionen är det möjligt att mäta hur klustrat ett punktmönster är.

Dessa funktioner kan användas för att anpassa en klustrad punktprocessmodell på given data. Utifrån dessa kunde vi kvantifiera att punktmönstrena för både friska och sjuka individer är klustrade. Dessutom såg vi att patienterna med diabetesneuropati hade mer klustrade punktmönster är de friska. Detta betyder att det är möjligt att använda denna metod för att diagnostisera diabetesneuropati, men metoden behöver testas på en större mängd data.

Sammanfattning

Ny teknik har lett till möjligheten att ta fram detaljerade bilder på nerverna i ytterhuden med hjälp av mikroskopi, vilket i sin tur avslöjat olika detaljer kring fördelningen av nervfibrer. Hos patienter med olika former av neuropati har det kunnat uttydas en större grad av klustring för nervtrådsändarna, jämfört med hos friska individer. Exempelvis är det intressant att se hur ändpunkterna av nervfibrernas spatiala mönster förändras i takt med att diabetesneuropatin avancerar. Eftersom mikroskopibilderna starkt tyder på att klustringen blir starkare ju mer utvecklad sjukdomen är, blir det viktigt att undersöka huruvida det är möjligt att i ett tidigare stadie diagnostisera patienter baserat på statistisk analys av ändpunktmönstren.

Centralt i detta arbete var att modellera spridningen av nervfibrers ändpunkter med hjälp av så kallade thomasprocesser, som är modeller för klustrade punktmönster, och därefter se hur väl parametrarna kan skattas för modellen. Populära skattningsmetoder som *maximum likelihood-metoden* fungerar väldigt bra vid arbete med analytiska uttryck av likelihoodfunktionen. Vid spatiala punktprocessmodeller och klusterprocesser är det dock väldigt svårt eller rent av omöjligt att härleda ett analytiskt uttryck. Därför användes skattningsmetoden *minimum contrast-metoden*, som är en minsta kvadrat-metod som bygger på sammanfattande statistikor.

I detta arbete undersökte vi hur valet av den sammanfattande statistikan i minimum contrast-metoden påverkar parameterskattningarna av thomasprocessen. Vi valde att jämföra hur två statistikor, parkorrelationsfunktionen och Ripleys K-funktion, påverkar skattningarna för parametrarna i thomasprocessen. Från simuleringsstudien tog vi fram parametervärden för vilka minimum contrast-metoden kombinerat med en av de två sammanfattande statistikorna ger tillförlitliga skattningar. Vidare anpassades modellen efter den givna nervdatan som tagits fram genom biopsi av huden och vi undersökte möjligheten av att modellera punktmönstrens nervtrådsändar genom thomasprocessen.

Från simuleringsstudien framgick det, om två av parametrarna fixeras och den sista varieras, att minimum contrast-algoritmen endast går att lita på för vissa värden av den varierande parametern. Dessa intervall skiljde sig åt beroende på om parkorrelationsfunktionen eller Ripleys K-funktion användes. Intervallet då Ripleys K-funktion användes var större än motsvarande för parkorrelationsfunktionen.

Det kan konstateras från analysen av nervdatan att punktmönstret av nervfibrernas ändpunkter är klustrade för både friska och sjuka patienter och att thomasprocessen fungerar bra som modell för detta. De sjuka patienterna verkar ha färre basnerver samt mer täta och separerade kluster än de friska. Eftersom den givna datan är ett väldigt litet stickprov så är det svårt att ta fram intervall för parametrarna i thomasprocessen som kan hjälpa oss avgöra ifall ett givet punktmönster kommer från en frisk eller en sjuk patient.

Nyckelord: diabetesneuropati, minimum contrast-metoden, Monte Carlo-simulering, parkorrelationsfunktionen, Ripleys K-funktion, spatiala punktmönster, thomasprocessen.

Abstract

Advances in image technology have given medical scientists the ability to take microscopic images of nerve fibers in the epidermis, which is the outermost part of the skin. This has subsequently unveiled some interesting characteristics of the spatial distribution of the nerve fibers. Patients suffering from different forms of neuropathy have exhibited a greater level of clustering of the nerve fiber end points as opposed to healthy individuals. Of particular interest is to study how the spatial pattern of the nerve fiber end points changes based on how advanced the diabetic neuropathy, in our case diabetic neuropathy, is. This leads to the question whether we can use spatial statistical analysis of the end point patterns in order to diagnose patients earlier.

Of major importance in this project was to model the distribution of nerve fibers using the so-called Thomas process, which is a model for clustered point patterns. This is used to evaluate how well we can estimate the parameters of the model. The popular parameter estimation method *maximum likelihood method* works well when we have analytical expressions of the likelihood function in closed form available. However, regarding spatial point processes and clustered processes, analytical expressions are very rarely available. This means that we had to resort to other means of estimation such as the *minimum contrast method*, which is a least square estimation technique that uses different summary statistics as input.

In this project we investigated how the choice of summary statistic in the minimum contrast method affects the estimations of the parameters of the Thomas process. We chose to compare the two summary statistics, the pair correlation function and Ripley's K-function. From the simulation work we found a range of parameter values that can be estimated reliably by using the minimum contrast method combined with one of the two summary statistics. Then, the model was fitted to the given data that has been collected by skin biopsies and we investigated whether it is possible to model the nerve end points data using the Thomas process.

The pair correlation function and Ripley's K-function influence the accuracy of the estimations of the parameters in the Thomas process. The simulation study is essential in this project since it was used in order to conclude proper intervals of parameters which gives reasonably accurate estimates, after which the model is fitted to the data that has been generated from skin biopsies.

Keeping two parameters fixed and only varying the third, the results of the simulation study indicated that we could only obtain reliable estimates of the parameters when the varied parameter was kept within a certain interval. This interval differed depending on whether we used the pair correlation function or Ripley's K-function as summary statistic. The interval obtained using the K-function was larger than the corresponding interval for the pair correlation function.

The analysis of the nerve end points data indicated that both point patterns for sick and healthy individuals alike were clustered and the Thomas process was indeed a good model for the data. However the sick patients had fewer base nerves and tighter and more separated clusters. Since the data is a very small sample, we could not calculate specific values of the parameters for which we could ascribe any point pattern to a healthy or sick individual.

Keywords: diabetic neuropathy, minimum contrast method, Monte Carlo simulation, pair correlation function, Ripley's K-function, spatial point processes, Thomas process.

Förkortningslista

Förkortning	Betydelse
CSR	Complete spacial randomness
MCM	Minimum contrast-metoden
MCS	Monte Carlo-simulering
PCF	Parkorrelationsfunktionen

Notationslista

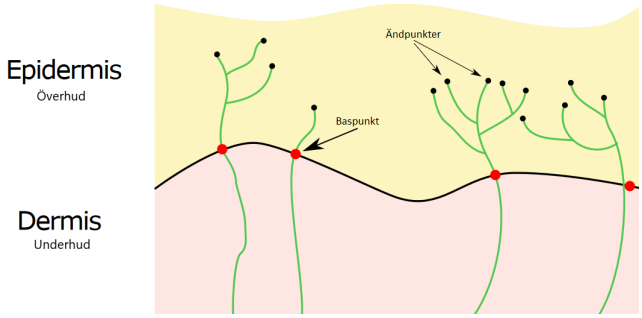
Notation	Förklaring
$\mathcal{T}(\kappa, \mu, \sigma)$	Thomasprocessen med parametrar κ, μ, σ.
κ	Intensitet av föräldrapunkter för thomasprocessen.
μ	Genomsnittliga antalet dotterpunkter per kluster.
σ	Standardavvikelsen för avståndet av en punkt från sin föräldrapunkt.
λ	Förväntade antalet totala dotterpunkter i ett givet punktmönster.
$\mathbb{V}[X]$	Varians av slumpvariabeln X.
$\mathbb{E}[X]$	Väntevärde av slumpvariabeln X.
$S(t)$	Beteckningen för en godtycklig sammanfattande statistika.
θ	Godtycklig parametervektor, till exempel $\theta = (\kappa, \mu, \sigma)$.
$D(\theta)$	Funktionen som MCM bygger på. Minimerar felet mellan teoretisk och skattad parameter.
$K(r)$	Ripleys K-funktion, beror av radien r.
$L(r)$	Ripleys L-funktion, beror av radien r.
$g(r)$	Parkorrelationsfunktionen, beror av radien r.
$K_{\text{poi}}(r)$	K-funktionen motsvarande den för CSR-punktmönster.
$L_{\text{poi}}(r)$	L-funktionen motsvarande den för CSR-punktmönster.
$g_{\text{poi}}(r)$	PCF motsvarande den för CSR-punktmönster.
$K_{\text{iso}}(r)$	K-funktionen skattad från data.
$L_{\text{iso}}(r)$	L-funktionen skattad från data.
$g_{\text{iso}}(r)$	PCF skattad från data.
$I_{K,\sigma}$	Intervall över σ som ger tillförlitliga parametrar, skattat av Ripleys K-funktion.
$I_{P,\sigma}$	Intervall över σ som ger tillförlitliga parametrar, skattat av PCF.

Innehåll

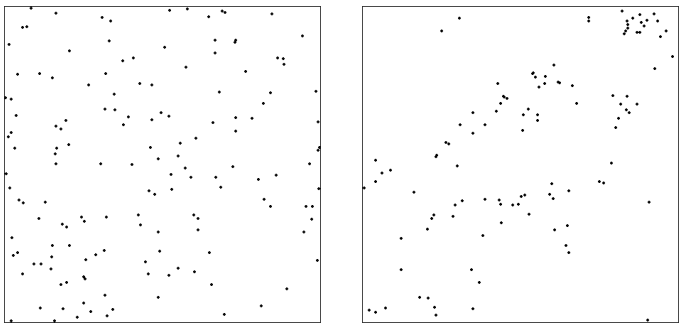
1 Inledning

En komplikation för diabetespatienter är att de även kan utveckla nervsjukdomen diabetesneuropati. Sjukdomen bryter ofta ut i patientens fötter och kan ge upphov till smärtor, nedsatt känsel och i de värsta fall infekterade fotsår som kan leda till amputation. När sjukdomen väl har uppstått kan inte skadorna i nerverna gå tillbaka, det är därför av stor vikt att sjukdomen upptäcks i tid så att det finns en chans att förebygga skador i bästa mån [1].

Tidigare forskning har visat att personer med diabetesneuropati har färre nervändar i överhuden. Dessutom verkar inte nerverna förtvina slumpmässigt, utan de försvinner så att resterande nervändar tenderar att bilda ett mer klustrat punktmönster än hos friska personer [16]. Ett punktmönster är data i form av punkter som är spridda över ett område och är ett resultat av en spatial punktprocess. Genom att analysera spridningen av punkterna är det möjligt att avgöra om punktmönstret är helt slumpmässigt, klustrat eller reguljärt. Vi behandlar så kallade klusterprocesser på grund av att spridningen av nervändar tenderar att vara mer klustrad än slumpmässig.



Figur 2: *Nervändarna i överhuden sett från sidan i ett tvärsnitt av huden. Sett uppifrån fås en bild av hur spridningen ser ut. Bild från Claes Andersson, 2017.*



(a) *Slumpmässigt punktmönster.* (b) *Klustrat punktmönster.*

Figur 3: *Punkterna representerar nervändarna i ytterhuden, sett uppifrån.*

För att skatta parametrar av modeller till data används traditionellt maximum likelihood-metoden, men för klusterprocesser är det inte möjligt att skriva ner likelihood-funktionen analytiskt och därför kommer vi istället använda oss av den så kallade minimum contrast-metoden. Det är en typ av minsta kvadrat-metod som har som avsikt att mäta avvikelsen mellan modell och data. Den gör detta genom att minimera skillnaden mellan en godtyckligt sammanfattad statistika som uppskattas från nervmönsterna och dess simulerade motsvarighet givet modellen [15]. Exempel på sammanfattande statistikor är parkorrelationsfunktionen och Ripleys K-funktion, som uppskattar antal punkter inom en given radie till en godtycklig punkt av processen. Genom att studera nervdata i form av punktmönster, från såväl friska som diabetessjuka patienter som är i ett tidigt stadie av neuropati, är förhoppningen att kunna hitta en modell som beskriver dessa typer av klustrade punktmönster. Detta för att kunna diagnostisera diabetesnuropati innan sjukdomsförloppet har utvecklas avsevärt.

1.1 Syfte

Syftet med detta projekt är att med hjälp av simuleringar undersöka hur parameterskattning för punktprocessmodeller påverkas av olika statistikor när minimum contrast-metoden används som skattningsmetod. Målen med projektet är att få en djupare förståelse för teorin om spatiala punktprocesser, undersöka klustrade punktmönster med hjälp av paketet `spatstat` i programspråket R samt kunna anpassa den modell som undersöks på given data. Projektet strävar efter att vara en del av huvudmålet, vilket är att kunna ligga ett steg före i processen av att upptäcka tidig diabetesneuropati genom att undersöka patienters nervändsfördelning.

1.2 Problem

Projektet har för avsikt att i första hand utföra en simuleringsstudie på genererad data, detta genom att göra ett lämpligt val av modell och statistika i minimum contrast-metoden. I detta arbete kommer vi att använda oss av thomasprocessen som är en modell som lämpar sig väl när det rör sig om mönster som uppvisar högre grad av klustring jämfört med ett mönster som är helt slumpmässigt genererat. De sammanfattande statistikorna som ligger i fokus för denna studie är Ripleys K-funktion och parkorrelationsfunktionen.

Målet med simuleringsstudien är att undersöka om valet av statistika påverkar parameterskattningarna på den genererade datan. Utifrån detta kommer vi få en bild av vilken av statistikorna som ger bäst resultat för den genererade datan, som sedan kommer att användas när vi anpassar thomasprocessen på nervdatan. Förutsättningarna säger att både individer med och utan diabetesneuropati kommer att följa ett klustrat punktmönster. Däremot förväntas det att insjuknade individer kommer att ha ett mer klustrat punktmönster än de som inte är drabbade av diabetesneuropati.

1.3 Avgränsningar

I detta projekt har vi valt att avgränsa valet till två olika statistikor. De metoder som kommer att undersökas är Ripleys K-funktion och parkorrelationsfunktionen. Anledningen till valet av dessa statistikor är att de ofta används på spatiala punktprocesser samt att båda de teoretiska funktionerna för thomasprocessen är kända. Dessutom är tidskravet för projektet ytterligare en bidragande faktor till varför vi har valt att avgränsa projektet till dessa två statistikor.

Projektet kommer att använda sig av data från totalt 14 personer, vilket är ett ganska litet urval. Detta kan komma att påverka vilka slutsatser som kan dras och det är inte säkert om det går att generalisera resultatet i samma utsträckning som om stickproven varit fler.

2 Teori

I detta avsnitt presenteras den matematiska teorin som ligger till grund för de metoder och statistiska modeller som tillämpas. Thomasprocessen är en klusterprocess som bygger på ett antal fördelningar och processer som vi definierar nedan. Vi skattar parametrarna med hjälp av minimum contrast-metoden (MCM) där vi kommer att använda oss av Ripleys K-funktion samt parkorrelationsfunktionen (PCF). En beskrivning av dessa samt redogörelse för hur de uppskattas hittas i detta avsnitt. Vi visualiserar sedan resultaten med hjälp av Monte Carlo-simuleringar (MCS).

2.1 Punktprocesser och spatial statistik

Inom punktmönsteranalys studeras utfallen av punkter relativt med varandra samt deras position inom given yta. Eftersom vi i detta projekt enbart kommer att behandla två dimensioner så pratar vi om yta, men generellt sett är det möjligt att arbeta i rummet $\mathbb{R}^n$, $n = 1, 2, ...$ Ett vanligt exempel är att analysera hur en viss trädart växer inom ett slutet geografiskt område. Då markeras positionerna av träden på en karta och använder sig av modeller för att beskriva den spatiala fördelningen av träden. Om positionerna av träden är samlade i flera olika grupper, så kallade kluster, är trädens placering klustrad. I det fallet kan *klusterprocesser* användas, som är en viss typ

av punktprocesser, som modell för trädens positioner. Detta görs ofta genom att använda sig av så kallade föräldra- och dotterpunkter. I en klusterprocess genereras en del punkter som föräldrapunkter med ett antal dotterpunkter fördelade runt dem enligt en fördelning. Klusterprocesser består bara av dotterpunkterna och ofta finns ingen information om föräldrapunkterna. I exemplet med träden skulle föräldrapunkterna vara de ursprungliga träden och dotterpunkterna vara de träden som skapas från frön från föräldrapunkterna [9].

Spatiala punktmönster kan analyseras genom deskriptiv och inferentiell statistik. I deskriptiv statistik används icke-parametriska metoder för att belysa strukturen av det spatiala mönstret. I inferentiell statistik anpassas en parametrisk modell för datan och skattar dess parametrar från data. I detta projekt kommer vi att använda oss av deskriptiv statistik i form av sammanfattande statistikor och inferentiell statistik i form av anpassning av en klusterprocess.

2.1.1 Statistiska fördelningar och Poissonprocessen

Den process som vi har valt att använda för simuleringar är thomasprocessen. För att närmare kunna beskriva den måste vi presentera lite förkunskaper. Thomasprocessen använder sig av fördelningsfunktionerna normalfördelning och poissonfördelning.

Normalfördelning är den vanligaste fördelningen som uppkommer naturligt i väldigt många sammanhang. Det är en kontinuerlig sannolikhetsfördelning för en reell stokastisk variabel.

Definition 1 (Normalfördelning)**.** *För en kontinuerlig stokastisk variabel X ges dess täthetsfunktion av*

$$f_X(x : \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left[\frac{x-\mu}{\sigma}\right]^2\right), \quad x \in \mathbb{R}. \tag{1}$$

Där parametrarna är väntevärdet $\mathbb{E}[X] = \mu \in \mathbb{R}$ och variansen $\mathbb{V}[X] = \sigma^2 > 0$. För att beskriva att en variabel X är normalfördelad med väntevärde μ och varians σ^2 skriver vi $X \sim \mathcal{N}(\mu, \sigma^2)$.

Utfallen i en normalfördelning är symmetriskt fördelade kring medelvärdet och ju längre ifrån medelvärdet desto färre utfall förekommer.

Poissonfördelningen är den vanligaste diskreta fördelningen. Denna fördelning används i poissonprocessen för att modellera antalet gånger en händelse har inträffat inom ett givet tidsintervall eller ett intervall i ett godtyckligt rum [13], [11], [10]. Vi definierar fördelningen enligt:

Definition 2 (Poissonfördelning)**.** *En diskret stokastisk variabel X sägs vara poissonfördelad med parameter $\lambda > 0$, om för varje $x = 0, 1, 2, ...$, ges täthetsfunktionen av*

$$f_X(x : \lambda) = \mathbb{P}[X = x] = \frac{\lambda^x e^{-\lambda}}{x!}. \tag{2}$$

Väntevärdet och variansen av X är lika med λ, $\mathbb{E}[X] = \mathbb{V}[X] = \lambda$. För en stokastisk variabel som är poissonfördelad med parameter λ skriver vi $X \sim \mathcal{P}(\lambda)$.

Som nämnts tidigare används denna fördelning för att modellera händelser i tid och endimensionellt rum i poissonprocessen. Det är dock möjligt att generellt arebta i $\mathbb{R}^d$, $d \geq 1$. När $d > 1$ brukar man prata om den *spatiala poissonprocessen.*

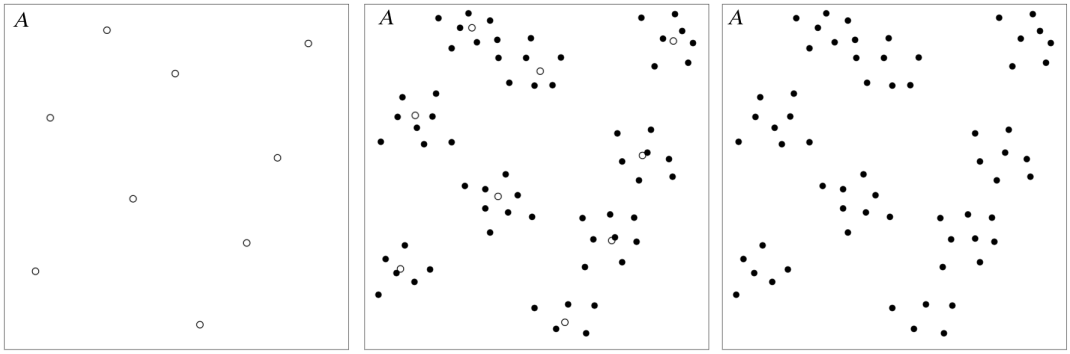
Definition 3 (Spatial poissonprocess)**.** *Låt $A \subset \mathbb{R}^2$ och låt N_A vara antalet punkter i A. Beteckna ytan av A som $|A|$. För en poissonprocess med parameter λ, även kallat intensiteten, gäller att*

1. *N_A har, för varje sluten mängd A, en poissonfördelning med parameter $\lambda|A|$,*
2. *om två mängder A och B är disjunkta, $A \cap B = \emptyset$, så är N_A och N_B oberoende stokastiska variabler.*

Den likformiga fördelningen uppkommer naturligt i denna process eftersom punkterna får en likformig fördelning över given yta. Av denna anledning refereras denna process även som *Complete Spatial Randomness (CSR).* Vid simulering av denna process så genereras x- och y-koordinaterna likformigt och oberoende av varandra och antalet punkter genereras enligt poissonfördelningen.

2.1.2 Thomasprocessen

Thomasprocessen är en klusterprocess som vi kommer att använda oss av i simuleringsstudien och i anpassning till data. Den bygger på alla ovanstående definitioner. Först genereras ett antal föräldrapunkter enligt den spatiala poissonprocessen som beskrivs ovan, med intensitet κ, $\sim \mathcal{P}(\kappa)$. Kring dessa föräldrapunkter genereras, för varje förälder, dotterpunkter som i antal är poissonfördelade med μ, $\sim \mathcal{P}(\mu)$. Dotterpunkternas avstånd från sina respektive föräldrar är normalfördelade med väntevärde 0 och varians σ^2, $\sim \mathcal{N}(0, \sigma^2)$, längs x- och y-axeln med föräldrapunkten i origo. Slutligen tas föräldrapunkterna bort så att endast dotterpunkterna kvarstår. Vi har valt att använda oss av thomasprocessen då den motsvarar nervdatan genom att låta basnerverna, se figur 3(b), betecknas av föräldrapunkter som inte observeras och deras förgreningar (ändpunkter) av dotterpunkter. Figur 4 visar visuellt hur denna process går till för att modellera ett punktmönster. För att uppskatta parametrarna av thomasprocessen krävs mer bakgrundsteori som presenteras i följande kapitel.



Figur 4: *Thomasprocessen i tre steg i område A. Modellering av föräldrapunkter (vänster), föräldrapunkter med tillhörande dotterpunkter (mitten) och bara dotterpunkter (höger).*

En statistika är ett tal som beskriver stickprov, som medelvärde och varians, men det kan även beskriva karaktärsdrag hos ett stickprov som är betydligt mer komplicerade. Följande kapitel presenterar de statistikor vi undersöker och kommer att använda oss av för att beskriva punktmönster och skatta parametrarna av thomasprocessen.

2.2 Ripleys K-funktion

Ripleys K-funktion är en fördelningsfunktion som används för att beskriva beteendet hos spatiala punktprocesser. Den beskriver, i det tvådimensionella fallet, antal punkter inom en radie, r, från en godtycklig punkt i punktprocess. Funktionen är definierad enligt:

$$K(r) = \lambda^{-1}\mathbb{E}[N_0(r)], \tag{3}$$

där $N_0(r)$ är antalet punkter hos processen inom ett avstånd r till en godtycklig punkt av processen och intensiteten λ är det genomsnittliga antalet punkter per enhetsarea [6].

Ripleys K-funktion ger information om hur klustrat ett spatialt punktmönster är. För ett punktmönster som uppfyller CSR, gäller det att $K(r) = \pi r^2, \forall r > 0$. Om värdet för K-funktionen är större än πr^2 betyder det att punktmönstret är mer klustrat, och om det har ett lägre värde är punktmönstret mer utspritt än CSR [7].

En variant av K-funktionen är Ripleys L-funktion, vilket ger en fördelningsfunktion med stabiliserad varians. Ripleys L-funktion definieras enligt

$$L(r) = \left(\frac{K(r)}{\pi}\right)^{1/2}. \tag{4}$$

Precis som med K-funktionen går det att jämföra L-funktionen mot CSR. För ett punktmönster som uppfyller CSR är $L(r) = r, \forall r > 0$. Det är vanligt att rita upp en graf för $L(r) - r$. Om

$L(r) - r > 0$ uppvisar punktmönstret en högre grad av klustring och om $L(r) - r < 0$ så rör det sig om ett regelbundet mönster. Ju närmare värdet 0 desto bättre följer punktmönstret CSR.

2.2.1 Uppskattning av K-funktionen

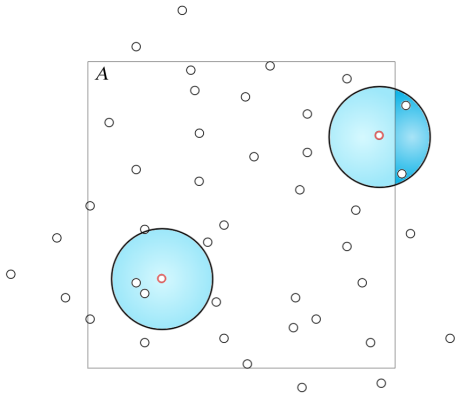
För att det ska vara möjligt att använda K-funktionen krävs det att vi definierar en lämplig uppskattning av funktionen. För att uppskattningen ska vara väntevärdesriktig krävs det även att vi introducerar en viktfunktion w_{ij}. Den är nödvändig då punkter som ligger nära områdets kant annars riskerar att bidra till att uppskattningen blir för låg, på grund av att inte hela området med radie r kring dessa punkter ligger i området A [7], [8]. Detta fenomen kallas *kanteffekt* och illustreras i figur 5. Då kan K-funktionen skattas av

$$\hat{K}(r) = \hat{\lambda}^{-1} \sum_{i=1}^{n} \sum_{j \neq i} \frac{w_{ij}^{-1} \mathbf{I}(r_{ij} \leq r)}{n}, \tag{5}$$

där

$$\hat{\lambda} = \frac{n-1}{|A|} \tag{6}$$

är en uppskattning av processens intensitet och n är antal punkter på området A. I ekvation (5) betecknar r_{ij} avståndet mellan punkt i och j. $\mathbf{I}$ är indikatorfunktionen, alltså är den lika med noll om $r_{ij} > r$ och lika med ett om $r_{ij} \leq r$ [6].



Figur 5: *Bilden ovan illustrerar fenomenet kanteffekt och hur det bidrar till felaktig uppskattning av K-funktionen.*

Viktfunktionen, w_{ij}, används i syfte att minska kanteffekten. Vi kommer använda Ripleys viktfunktion som definieras genom att låta $w(x_i, r)$ beteckna andelen av omkretsen av cirkeln med centrum i x_i och radie r som ligger i området A. Definiera sedan $w_{ij} = w(w_i, r_{ij})$. Resultatet av detta blir att de punkter som ligger nära kanten av området vi undersöker kommer att viktas högre i summan av uppskattningen i ekvation (5).

2.2.2 Teoretiska K-funktionen för thomasprocessen

I sektion 2.2 definierades Ripleys K-funktion enligt ekvation (3). Denna funktion kan explicit härledas för vissa processer som exempelvis thomasprocessen [12]. Den teoretiska K-funktionen för thomasprocessen med parametrar $\theta = (\kappa, \mu, \sigma)$ blir

$$K(r) = \pi r^2 + \kappa^{-1} \left(1 - \exp\left(-\frac{r^2}{4\sigma^2}\right)\right). \tag{7}$$

Härledningen hittas i appendix A.1.

Det kan noteras från ekvation (7) att K-funktionen inte beror på μ. Därmed kan två olika thomasprocesser där det enda som skiljer är parametern μ fortfarande ha samma K-funktion.

2.3 Parkorrelationsfunktionen

Parkorrelationsfunktionen, förkortad PCF, för ett punktmönster beskriver hur densiteten av resterande utfall varierar i förhållande till avståndet från ett godtyckligt utfall i systemet. På så sätt beskriver PCF derivatan av Ripleys K-funktion. Funktionen definieras med hjälp av funktionen $\rho(\xi)\mathrm{d}\xi$, vilket är sannolikheten att vi kan hitta ett utfall i den oändligt lilla omgivningen med centrum i ξ, och funktionen $\rho^{(2)}(\xi,\eta)\mathrm{d}\xi\mathrm{d}\eta$ som är sannolikheten att hitta utfall i båda oändligt små omgivningar med centrum i ξ och η [14]. PCF ges sedan av

$$g(\xi,\eta) = \frac{\rho^{(2)}(\xi,\eta)}{\rho(\xi)\rho(\eta)}. \tag{8}$$

Vi kan anta att vårt system är stationärt och isotropiskt, det vill säga att vi antar att det inte spelar någon roll var i systemet vi befinner oss eller hur punkterna ξ och η ligger i förhållande till varandra förutom avståndet, $r = |\xi - \eta|$, mellan dem. I det fallet blir g invariant under omvandling så att $g(\xi,\eta) = g(r)$.

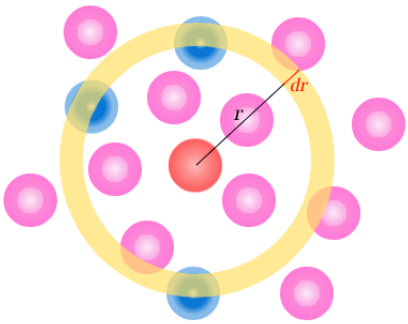
För CSR är $\rho^{(2)}(\xi,\eta) = \rho(\xi)\rho(\eta)$ och därför är $g(r) = 1$. Om $g(r) > 1$ betyder det att det är mer sannolikt att hitta två punkter på avståndet r jämfört med CSR, det vill säga en homogen poissonprocess med samma intensitet. Vilket betyder att punktprocessen är klustrat, medan om $g(r) < 1$ betyder det att systemet är mer regelbundet än CSR. Notera att

$$\lim_{r\to\infty} g(r) = 1$$

oavsett om systemet är helt slumpmässigt, klustrat eller regelbundet.

2.3.1 Uppskattning av parkorrelationsfunktionen

PCF uppskattas genom en algoritm som beräknar antal punkter som ligger inom ett avstånd från r och $r+dr$ från en referenspunkt, vilket illustreras i Figur 6. Antag ett punktmönster med N punkter



Figur 6: *Bilden ovan illustrerar hur PCF uppskattas med hjälp av att införa avståndet dr och på så sätt räkna antal punkter som ligger på ett visst avstånd, r, från en godtycklig punkt.*

och area A, då är medeldensiteten $\rho = N/A$. Algoritmen för uppskattningen av $g(r)$ innefattar upprepade beräkningar för de värden av r som är av intresse. För varje r beräknas medelantalet punkter som ligger inom området mellan r och $r + dr$ från en godtycklig referenspunkt i systemet. Därefter divideras det antalet med ytarean av området, ungefär $2\pi r\, dr$. Slutligen divideras talet även med ρ, för att säkerställa att $g(r) = 1$ för punktmönster som inte har någon struktur. Detta beskrivs enligt

$$g(r) = \frac{N(r+dr)}{\Omega(r+dr)}\frac{1}{\rho}, \tag{9}$$

där $N(r+dr)$ är antalet punkter som ligger innanför intervallet $r+dr$ och $\Omega(r+dr)$ är arean av det området som undersöks [5].

2.3.2 Teoretiska parkorrelationsfunktionen för thomasprocessen

Tidigare i kapitlet definierades PCF enligt ekvation (8) och thomasprocessen i sektion 2.1.2. Den teoretiska PCF för thomasprocessen med parametrar $\theta = (\kappa, \mu, \sigma)$ är

$$g(r) = 1 + \frac{1}{4\pi\kappa\sigma^2} \exp\left(-\frac{r^2}{4\sigma^2}\right). \tag{10}$$

En härledning för detta finns i appendix A.2. Som tidigare noterat så gäller att

$$\lim_{r\to\infty} g(r) = 1, \tag{11}$$

vilket följer direkt från ekvation (10). Notera även att PCF ej beror på μ.

2.4 Viktat medelvärde från data

Anta att vi har data som består av m punktmönster. Om de underliggande punktprocesserna antas ha samma K-funktion, kan vi uppskatta den gemensamma K-funktionen genom att ta ett medelvärde av K-funktionerna som skattats från de m punktmönstrena. Det kan vara fördelaktigt att ta hänsyn till hur många punkter varje punktmönster har, för att få en bättre bild av hur processen beter sig. Vi kan då beräkna ett viktat medelvärde av K-funktionen, enligt ekvation (12). Här är n_i antalet punkter för punktmönstret i, och K_i är K-funktionen till process $i = 1, ..., m$.

$$\hat{K}(r) = \sum_{i=1}^{m} n_i \hat{K}_i(r) / \sum_{i=1}^{m} n_i. \tag{12}$$

Vi behöver inte anta att intensiteten till de underliggande processerna är samma på grund av att K-funktionen inte påverkas punkter tas bort slumpmässigt. Vi kommer använda oss av detta för att kunna jämföra de två grupperna i nervdatan, genom att använda funktionen `pool` i R. Framöver kommer vi därför referera till de gemensamma K-funktionerna som *poolade* K-funktioner.

2.5 Minimum Contrast-metoden

Maximum likelihood-metoden (MLM) är en parameterskattningsmetod som ofta används tack vare dess bra teoretiska egenskaper för att beskriva data under en viss sannolikhetsfördelning. Då maximeras likelihood-funktionen, $L(\theta)$, som fås från täthetsfunktionen eller sannolikhetsfunktionen med avseende på parametrarna θ. Men för klusterprocesser så är det sällan möjligt att skriva ner likelihood-funktionen analytiskt, varför det istället är möjligt att använda minimum contrast-metoden, förkortad MCM.

MCM är en minsta kvadrat-metod som skattar parametrar som minimerar avvikelsen mellan det teoretiska värdet (eller simuleringsbaserade värdet), $S(t;\theta)$, och det uppskattade värdet från datan, $\hat{S}(t)$, för den givna sammanfattande statistikan. Parametrarna uppskattas genom att minimera integralen

$$D(\theta) = \int_{\nu}^{t_0} w(t) \left(\hat{S}(t)^c - S(t;\theta)^c\right)^p dt, \tag{13}$$

med avseende på θ. Integralen beror på flera parametrar; ν, t_0, $w(t)$, c och p. R-paketet `spatstat` har en metod för MCM, `mincontrast`, som används för simuleringsstudien, där $c = 0.25$ och $p = 2$ är de förvalda värdena [3], och om inte annat specificeras är $t_0 = 0.25$, $\nu = 0$. Detta är lämpligt eftersom både vid Ripleys K-funktion och PCF så rör det sig om radiella avstånd. Parametervärdet $p = 2$ ges av likheten med minsta kvadrat-metoden. Enligt Diggle så är $w(t) = 1$ ett bra val till klustrade punktmönster [6]. Diggle rekommenderar även $t_0 = 0.25 \min(a, b)$ där a och b är sidlängderna för området. I ett tidigare arbete undersöktes hur valet av integrationsgränserna påverkade parameterskattningarna [12]. Då det redan har gjorts en gång, undersöks det inte igen, utan de förvalda värdena för t_0 och ν används.

För vissa statistikor såsom Ripleys K-funktion och PCF är det möjligt att explicit definiera det teoretiska värdet. Exempelvis sätter vi $S(t)$ i MCM som Ripleys K-funktion med de valda parametrarna och erhåller följande integral:

$$D(\theta) = \int_0^{0.25} \left(\hat{K}(t)^{0.25} - K(t;\theta)^{0.25} \right)^2 dt. \tag{14}$$

Analogt görs för PCF.

2.6 Monte Carlo-simulering

Monte Carlo-metoden är en simuleringsmetod som används för att göra numeriska uppskattningar då andra metoder blir för komplexa. Den grundas på ett stort antal simuleringar, så kallade Monte Carlo-simuleringar (MCS), som sker genom att slumpmässigt generera en stor mängd variabler från en sannolikhetsfördelning över ett område. Trots att varje enskild simulering är slumpmässig så är idén att genomsnittet ska representera den verkliga fördelningen.

Monte-Carlo metoden är möjlig att använda för att uppskatta integralen

$$I = \int_\Omega f(x)\hat{\mu}(dx), \tag{15}$$

med summan

$$\hat{I} = \frac{1}{m}\sum_{i=1}^{m} f(x_i). \tag{16}$$

Parametern m är antalet genererade slumpvariabler, x_i, med fördelningen $\hat{\mu}$. Området Ω är det som variablerna slumpas över och f är en godtycklig funktion som är intressant i sammanhanget. Med hjälp av MCS är det möjligt att bland annat göra hypotesprövningar och uppskatta konfidensintervall [4].

I vårt fall kommer vi småningom göra ett p-test för anpassningen av thomasprocessen. Det enklaste sättet att ta fram ett p-värde är att beräkna skattningen av den sammanfattande statistikan, $\hat{S}(t)$, för observerat ändpunktmönster och därefter skattningarna $\hat{S}_i(t),\ i = 1, ..., m$ från m antal simuleringar av den anpassade thomasprocessen med parametrar $\hat{\theta} = (\hat{\kappa}, \hat{\mu}, \hat{\sigma})$. Dessa parametrar erhålls genom att använda $\hat{S}(t)$ och MCM, se ekvation (13). Dessa $\hat{S}_i(t)$ jämförs mot det analytiska uttrycken för $S_i(t : \hat{\kappa}, \hat{\mu})$ som i vårt fall antingen kommer vara det analytiska uttrycket för Ripleys K-funktion eller PCF för thomasprocessen vilka ges i avsnitt 2.2.2 och 2.3.2. Notera att de analytiska uttrycken $S_i(t : \hat{\kappa}, \hat{\mu})$ innehåller de *olika* skattningarna av κ, μ som erhålls av ekvation (13), därav indexeras dem med i nedan i ekvation (17). Statistikan skattas för alla $t \in [0, \min(a, b)]$. Vi inför en ny teststatistika G_i som vi definierar som summan av den kvadratiska skillnaden mellan skattningarna $\hat{S}_i(t)$ och den teoretiska statistikan $S_i(t)$, alltså

$$G_i = \sum_{t\in[0,\min(a,b)]} \left[\hat{S}_i(t) - S_i(t : \hat{\kappa}_i, \hat{\mu}_i) \right]^2. \tag{17}$$

Notera att dessa G_i enbart kommer från simuleringarna, men vi behöver även en referens av G-statistikan att jämföra mot. Denna referens är G_{obs} som erhålls genom att sätta in $\hat{S}(t)$ och $S(t : \hat{\kappa}, \hat{\mu})$ i summanden i ekvation (17) enligt

$$G_{\text{obs}} = \sum_{t\in[0,\min(a,b)]} [\hat{S}(t) - S(t : \hat{\kappa}, \hat{\mu})]. \tag{18}$$

För att beräkna p-värdet för varje patient räknar vi hur många simulerade statistikor G_i är större än eller lika med G_{obs} och beräknar andelen enligt

$$p = \frac{1}{m}\sum_{i=1}^{m} \mathbf{I}(G_i \geq G_{\text{obs}}). \tag{19}$$

2.7 Modellanpassning

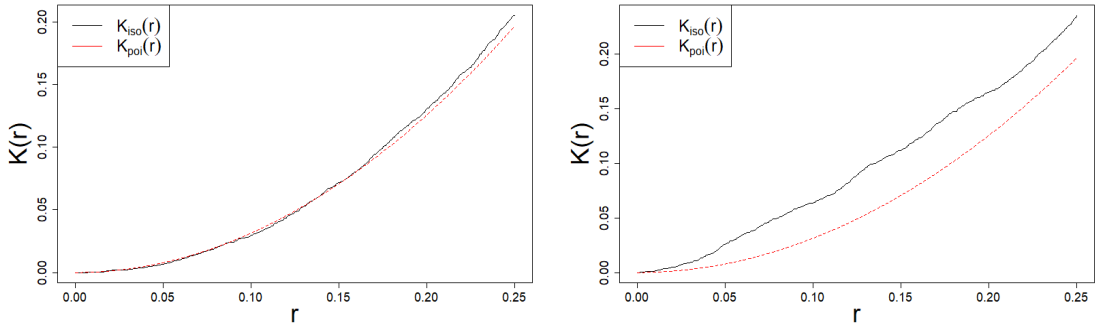
Som nämnts tidigare kan det vara komplicerat, om inte omöjligt, att ta fram slutna analytiska uttryck. Som följd av detta tillgripes ofta simulering av nollhypotesen som ett sätt att skapa en

referensfördelning. Analysen av punktmönster börjar ofta genom att testa nollhypotesen $\mathcal{H}_0$, då punktspridningen representeras av en CSR-modell. I detta avsnitt visar vi hur statistikorna används för att skapa överskådliga grafer, som enkelt kan läsas av för att få en uppskattning av de olika egenskaperna hos punktmönster.

2.7.1 Ripleys K-funktion

Vid första anblick är det lockande att bara visuellt titta på ett punktmönster för att avgöra hur klustrat mönstret är. Det är dock svårt att bedöma egenskaper hos punktmönster genom att endast observera det. Därför använder vi oss av `Kest` i `spatstat` för att skatta K-funktionen för givna punktmönster. Vi jämför K-funktionen K_{iso} som är skattad från data med K-funktionen K_{poi} som är skattad från CSR-mönster med samma intensitet som datan. Observera att K_{iso} motsvarar $K(r)$ i integranden i ekvation (14).

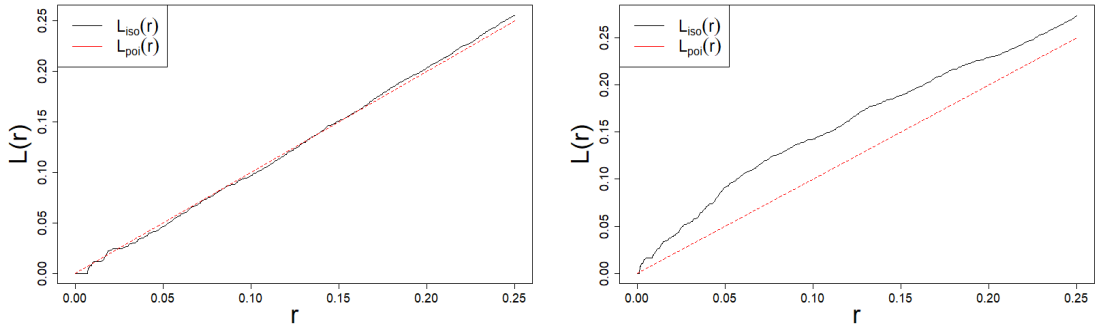
För de r som ger $K_{\text{iso}}(r) > K_{\text{poi}}(r)$ kan vi förvänta oss fler punkter på det avståndet än vid CSR med samma intensitet, dvs. det finns högre tendens för ett klustrat mönster. För de r som medför $K_{\text{iso}}(r) < K_{\text{poi}}(r)$ förväntas ett mer regelbundet punktmönster. Figur 7 visar hur $K_{\text{iso}}(r)$ och $K_{\text{poi}}(r)$ förhåller sig till varandra beroende på om vi tittar på ett ett helt slumpmässigt mönster så som figur 3(a) eller ett mer klustrat mönster, som figur 3(b).



(a) *Slumpmässigt mönster.* (b) *Klustrat mönster.*

Figur 7: *Uppskattade K-funktionen $K_{iso}(r)$ jämfört med den teoretiska K-funktionen under CSR $K_{poi}(r)$.*

För att få en tydligare bild av situationen för små r använder vi istället L-funktionen. Detta beräknas i R via funktionen `Lest` i `spatstat`. Se resultaten av detta i figur 8.

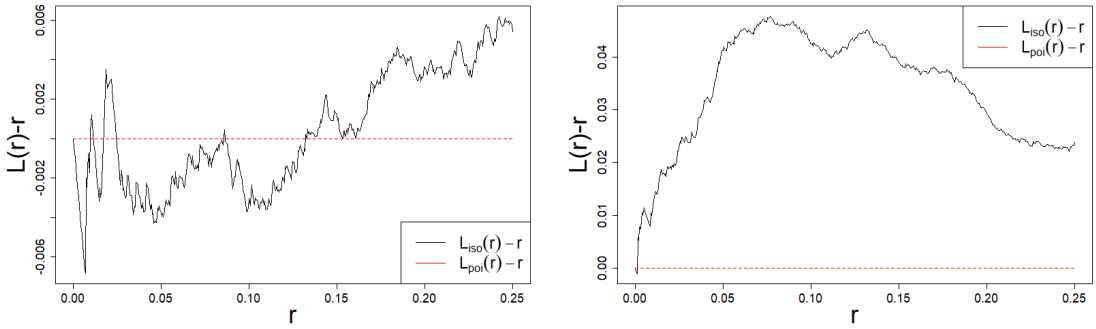


(a) *Slumpmässigt punktmönster.* (b) *Klustrat punktmönster.*

Figur 8: *Uppskattade L-funktionen $L_{iso}(r)$ jämfört med den teoretiska L-funktionen under CSR $L_{poi}(r)$.*

För att få en ännu tydligare bild gör vi ytterligare en modifikation av K-funktionen och ritar $L(r)-r$ istället. Då är $L_{\text{poi}}(r)-r$ alltid lika med 0. Figur 9 nedan visar resultaten. Notera skillnaden

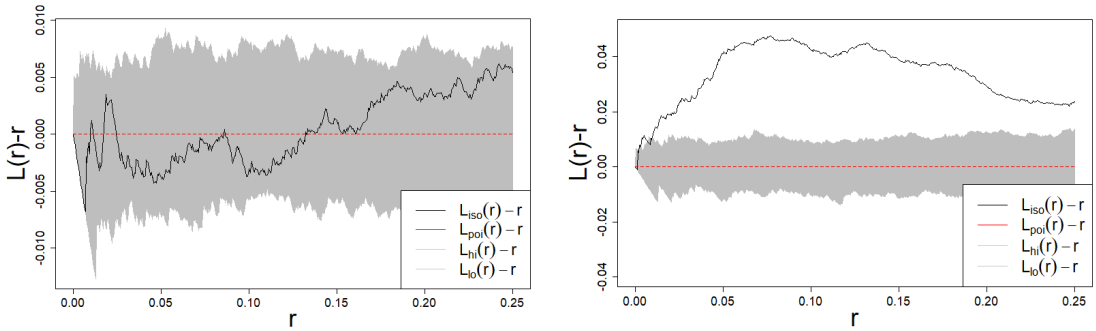
mellan dessa funktioner beroende på om vi använder ett klustrat mönster eller ett CSR-mönster. För det slumpmässiga mönstret är skillnaden mellan $L_{\text{poi}}(r)$ och $L_{\text{iso}}(r)$ betydligt mindre längs hela intervallet för r medan denna skillnad ökar väldigt snabbt för det klustrade mönstret.



(a) *Slumpmässigt punktmönster.* (b) *Klustrat punktmönster.*

Figur 9: *Uppskattad $L_{iso}(r) - r$ funktion jämfört med den teoretiska $L_{poi}(r) - r$ funktion under CSR.*

Problemet med att enbart jämföra $L_{\text{iso}}(r)$ med $L_{\text{poi}}(r)$ är att det inte ger oss någon information om variationen. Genom att simulera ett stort antal CSR-mönster och använda `envelope` i R är det möjligt att konstruera ett band baserat på de K-funktionerna uppskattade från punktmönstrena. Bandet är räknat så att för varje r lämnas 2.5% av de lägsta och 2.5% av de högsta värdena utanför bandet. Figur 10(a) visar att $L(r) - r$ det slumpmässiga mönstret ligger inom det gråa bandet vilket indikerar att det inte finns något som talar emot att detta faktiskt är ett CSR-mönster.



(a) *Slumpmässigt punktmönster.* (b) *Klustrat punktmönster.*

Figur 10: *95%-band på respektive $L(r) - r$ grafer.*

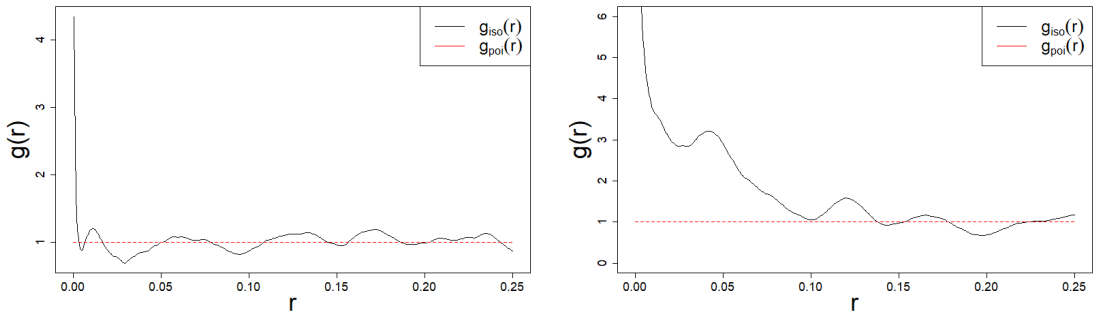
I figur 10(b) avviker $L_{\text{iso}}(r) - r$ mycket ifrån $L_{\text{poi}}(r) - r$ vilket medför att den hamnar utanför det grå CSR-bandet. Det är därmed inte möjligt att med stor säkerhet säga att det rör sig om en CSR-process och vi måste förkasta nollhypotesen vilket betyder att punktmönstret som används i figur 10(b) verkar klart vara mer klustrat än CSR.

2.7.2 Parkorrelationsfunktionen

Som nämns i 2.3 så beskriver PCF densiteten av förväntat utfall för ett kluster i ett punktmönster. Vi kommer använda oss av 3(a) och 3(b) som referensmönster. Även om det går att avgöra visuellt vilken typ av punktmönster som de följer, är detta inte alltid fallet.

Vi använder oss av `PCF` i `spatstat` för att uppskatta $g(r)$, $r \in [0, 0.25]$, för samma punktmönster som i figurerna ovan. Vi jämför sedan $g_{\text{iso}}(r)$ med $g_{\text{poi}}(r)$. Här är $g_{\text{iso}}(r)$ PCF som uppskattas från datapunkterna med Ripleys viktfunktion, se 2.2.1, och $g_{\text{poi}}(r)$ betecknar PCF för CSR. För de r där $g_{\text{iso}}(r) > g_{\text{poi}}(r)$ så observeras det flera par av punkter vid detta avstånd jämfört med CSR, alltså

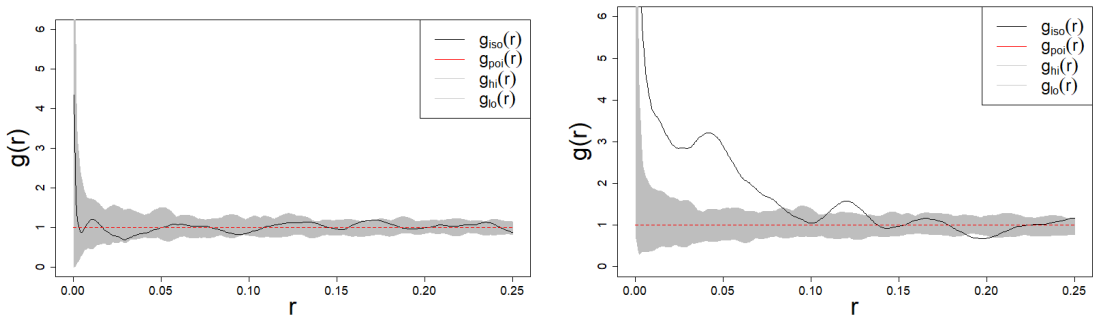
är mönstret mer klustrat. När $g_{\text{iso}}(r) < g_{\text{poi}}(r)$ är mönstret mer regelbundet än CSR. Figur 11 visar hur $g_{\text{iso}}(r)$ och $g_{\text{poi}}(r)$ ser ut för ett helt slumpmässigt mönster och ett klustrat mönster. För CSR-mönstret följer det observerade PCF, $g_{\text{iso}}(r)$, $g_{\text{poi}}(r)$. För det klustrade mönstret observeras att flera punkter ligger i närheten av föräldrapunkterna för intervallet $r \in [0.00, 0.05]$. Sedan avtar antal punkter i förhållandet till föräldrapunkten och börjar följa $g_{\text{poi}}(r)$. Detta tyder på att enbart det slumpmässiga mönstret följer CSR, som det bör göra.



(a) *Slumpmässigt mönster.* (b) *Klustrat mönster.*

Figur 11: *Uppskattad g-funktion $g_{iso}(r)$ jämfört med den teoretiska g-funktionen under CSR $g_{poi}(r)$.*

Vi använder oss nu av MCS och funktionen `envelope` i R. Vi simulerar 99 stycken CSR punktmönster, där PCF uppskattas. Detta visualiseras i figur 12(a) och 12(b). För det slumpmässiga mönstret kan vi konstatera att det verkar ha genererats från CSR, så nollhypotsen kan inte förkastas. Detta representeras med att de observerade värdena $g_{\text{iso}}(r)$ ligger mellan $g_{\text{hi}}(r)$ och $g_{\text{lo}}(r)$. För det klustrade mönstret ligger $g_{\text{iso}}(r)$ utanför bandet, så vi kan förkasta nollhypotesen och dra slutsatsen att det inte finns skäl till att anta att det rör sig om en CSR-process.



(a) *Slumpmässigt punktmönster.* (b) *Klustrat punktmönster.*

Figur 12: *95%-band på respektive $g(r)$-grafer.*

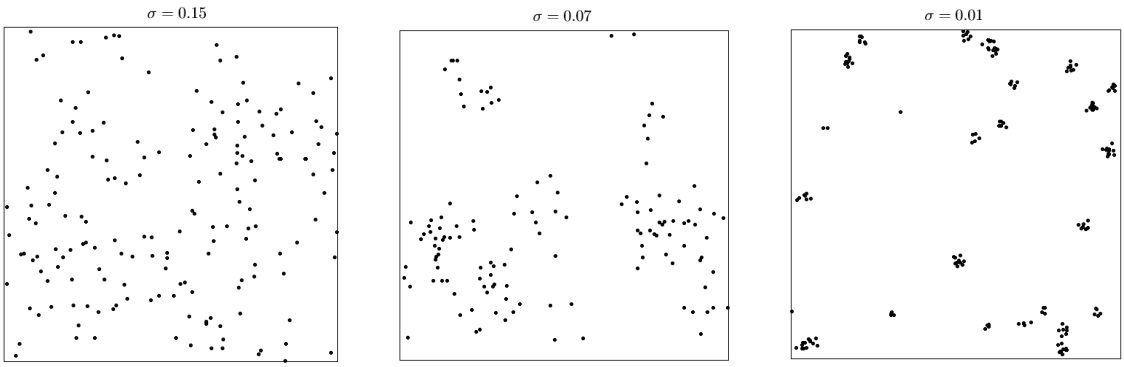
3 Simuleringsstudie

I simuleringsstudien undersöker vi hur de olika statistikorna påverkar skattningen av parametrarna i thomasprocessen. Vi simulerar realisationer av thomasprocessen med olika parametrar som vi har valt utifrån observationer från förundersökningen av nervdata, se appendix B.2. Eftersom thomasprocessen karaktäriseras av parametrarna κ, μ och σ, betecknas processen med $\mathcal{T}(\kappa, \mu, \sigma)$. Parametern κ motsvarar intensiteten av föräldrapunkter, μ är intensiteten av dotterpunkterna och σ är standardavvikelsen av avståndet av en punkt (längs varje koordinataxel) från sitt klustercentrum. Den totala intensiteten för hela processen är $\lambda = \kappa\mu$ som är antalet förväntade punkter.

3.1 Skattning av σ, κ och μ

Vi vill få en inblick i hur väl vi kan skatta parametrarna med olika parametervärden och de sammanfattande statistikor som vi arbetar med. Vi har valt att simulera med fasta värden för κ och μ, nämligen $\kappa = 22.9$ och $\mu = 4$, och låta σ variera. Värden på κ och μ har valts så att de motsvarar värdena i nervdatan. Hur vi kommit fram till dessa värden förklaras i appendix B.2 där vi gör en förundersökning av nervdatan. Alla simuleringar är gjorda i enhetskvadraten $[0, 1] \times [0, 1]$.

Vi börjar med att titta på hur punktmönstrena för tre olika värden på σ, se figur 13.



(a) *Punktprocess nära CSR.* (b) *Uppenbar klustring.* (c) *Extrem klustring.*

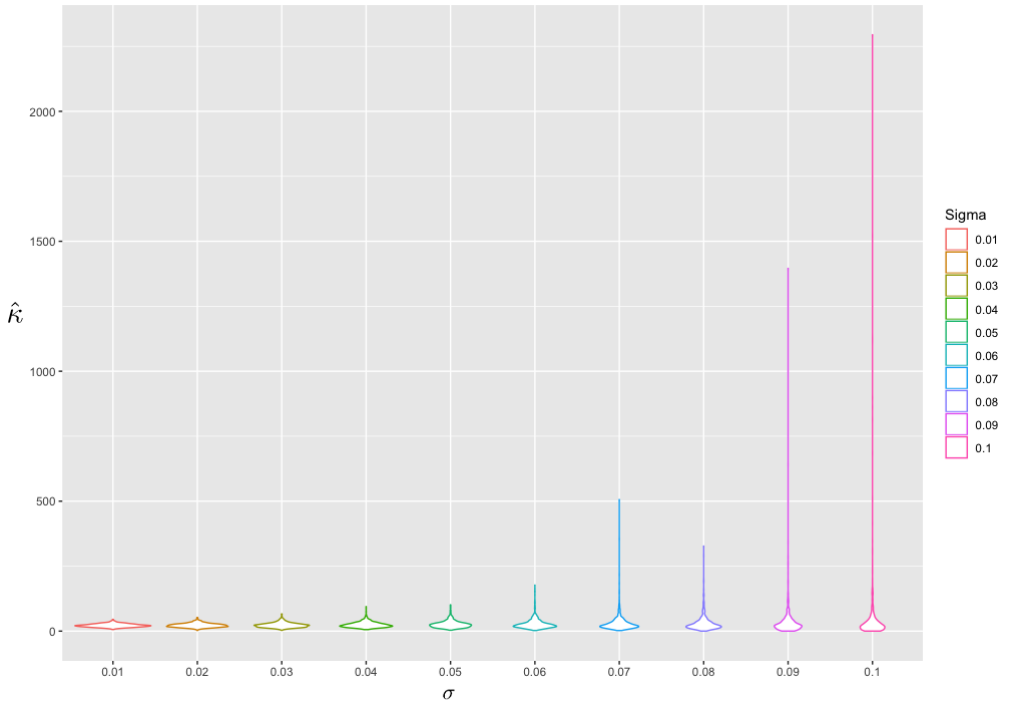
Figur 13: *Punktmönster med varierande σ. När σ är relativt stort ser punktmönstret ut att vara slumpmässigt och minskande σ ökar klustringen.*

Skulle vi välja för stora värden på standardavvikelsen så närmar sig thomasprocessen CSR som är ointressant, se figur 13(a). Ett för litet värde på standardavvikelsen ger å andra sidan ett väldigt tätt klustrat mönster som kan ses i figur 13(c). Vi bör hålla oss till $\sigma > 0.01$ för att inte klustringen ska bli så extrem och icke-representativ av nervdatan. För att undersöka ett lite bredare spektrum av värden har vi valt att undersöka alla $\sigma \in [0.01, 0.15]$.

De funktioner i R som användes var `rThomas, thomas.estK` och `thomas.estpcf`. Den funktion som genererar slumpmässiga realisationer av thomasprocessen är `rThomas`, som genererar algoritmen för föräldrapunkter och dotterpunkter. Funktionerna `thomas.estK` och `thomas.estpcf` skattar parametrarna κ och μ givet ett thomas-punktmönster genom att använda Ripleys K-funktion respektive PCF i MCM. Se detaljerna för hur skattningarna beräknas i avsnitt 2.2.1 och 2.3.1. Enligt ekvationerna (7) och (8) är statistikorna obereoende av μ, och därför så kommer inte funktionerna att skatta denna parameter. Detta gör vi istället genom att från punktmönstret ta fram totala antalet punkter och dividera dessa med $\hat{\kappa}$, ty vi vet att $\kappa \cdot \mu = \lambda$ är förväntade antalet punkter.

För att gå vidare med frågeställningen måste vi se hur väl vi kan skatta parametrarna från den simulerade thomasprocessen med varierande $\sigma \in [0.01, 0.15]$. Detta görs enklast genom att partitionera intervallet $[0.01, 0.15]$ i några mindre delintervall med likformig steglängd $h = \frac{0.15-0.01}{14} = 0.01$ och loopa igenom med `rThomas` och `thomas.estK` respektive `thomas.estpcf` för dessa värden. För varje värde på σ kördes 1000 simuleringar av thomasprocessen och lika många skattningar av varje parameter. Dessa parametrar läggs in i 3 olika 1×1000-vektorer och medelvärdet av alla värden i respektive vektor beräknades för att slutligen ta fram en rimlig skattning. Detta gjordes en gång med Ripleys K-funktion och en gång med PCF, tabell C.7 och C.8 i appendix visar resultaten.

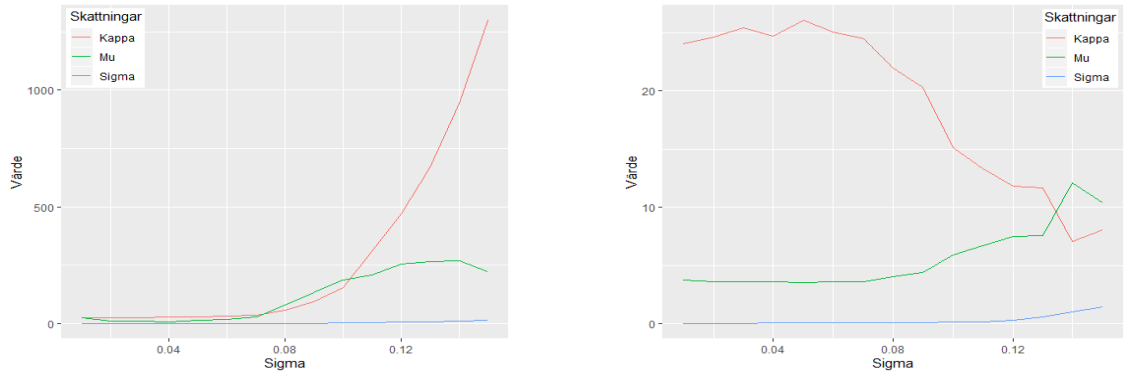
Tabellerna visar en stor varians hos skattningarna och att generella trender saknas. Vi kan därför inte säga så mycket om intervallet I_σ där skattningsmetoden fungerar bra. För att förstå vad som har gått snett tittar vi på de empiriska fördelningarna av parameterskattningarna. Vi kan visualisera dessa avvikande värden genom att använda så kallade violindiagram, se figur 14 och appendix D.3. Observera att y-axeln är logaritmerad för de violindiagrammen i appendix. Vi ser att för både PCF och Ripleys K-funktion har alla empiriska fördelningar för κ och μ en stor spridning efter $\sigma \approx 0.05$. Detta leder självklart till att även σ får en stor spridning eftersom σ direkt beror av κ och μ. Detta leder till att beräkningar av medelvärde påverkas väldigt mycket



Figur 14: *Ett violindiagram av empiriska fördelningar av $\hat{\kappa}$ då K-funktionen används. När σ är lågt koncentreras alla tusen utfallen från simuleringen kring ett värde, i detta fall kring 23. I takt med att σ ökar uppkommer fler avvikande värden upp till två tusen. "Kroppen" på violinen innehåller de flesta utfall och de långa strecken är avvikande värden.*

och därmed ger missvisande resultat.

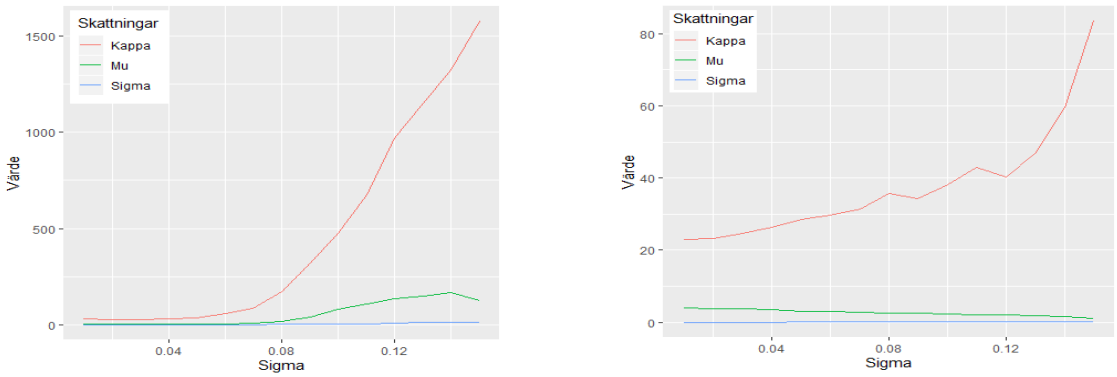
Eftersom de flesta värden för μ och κ hamnar i intervallen $[0, 10]$ respektive $[0, 40]$ så måste vi ändra på hur vi får gemensamma skattningen från simuleringarna för att eliminera påverkan från avvikelserna. Det enklaste sättet är att använda medianvärdet istället för medelvärdet. Figurerna 15 och 16 visar hur värden på skattningarna beror på σ. För både K-funktionen och PCF erhålls betydligt bättre skattningar baserat på simuleringarna när median används. Notera skalan på y-axlarna för medelvärde kontra de för median.



(a) *K-funktion och medelvärde.*

(b) *K-funktion och median.*

Figur 15: *Skattningar av de tre parametrar vid användning av Ripleys K-funktion och medelvärde (vänster) respektive median (höger) av de 1000 skattningarna. Notera att skalan på y-axeln i den vänstra figuren skiljer sig åt från skalan i den högra.*



(a) *PCF och medelvärde.* (b) *PCF och median.*

Figur 16: *Skattningar av de tre parametrar vid användning av PCF och medelvärde (vänster) respektive median (höger) av de 1000 skattningarna. Notera att skalan på y-axeln i den vänstra figuren skiljer sig åt från skalan i den högra.*

Från tabell C.7 och C.8 i appendix och figurerna 15(b) och 16(b) ser vi att *alla* skattningarna $\hat{\kappa}, \hat{\mu}, \hat{\sigma}$, för K-funktionen då median används, är väldigt stabila kring det verkliga värdet fram till och med $\sigma = 0.09$. När PCF används försämras värdena för $\hat{\kappa}$ betydlig fortare, redan vid $\sigma = 0.03$.

3.2 Sammanfattning av resultaten simuleringsstudien

Som slutsats för simuleringsstudien kan vi säga att ett bra intervall för σ då vi använder oss av K-funktionen är $I_{\sigma,K} = [0.02, 0.09]$, ty vi tidigare infört restriktionen att $\sigma > 0.01$ för att begränsa klustringen. Då vi använder PCF fås det tillförlitliga intervallet till $I_{\sigma,P} = [0.02, 0.03]$. Spontant ser det ut som att K-funktionen fungerar bättre eftersom det erhålls högre stabilitet på skattningarna över ett större intervall för σ. Notera att dessa intervall baseras på att simuleringarna av thomasprocessen genererats av $(\kappa, \mu) = (22.9, 4)$. Hade dessa värden varit annorlunda hade även intervallen $I_{\sigma,K}$ och $I_{\sigma,P}$ skiljt sig från erhållna resultat. Detta säger oss att MCM-algoritmen i R konvergerar endast för σ som leder till att punktmönstret är klustrat. Ju längre ifrån CSR, desto bättre skattningar.

4 Analys av nervdata

Datan är tillgänglig som en rds-fil, `endpoints.rds`, visualisering av datan finns i appendix D.1. Nervdatan och övriga punktmönster lagras som ett `ppp`-objekt i R och står för *planar point pattern*. Struktureringen av ett sådant objekt innehåller alla x- och y-koordinater för varje dotterpunkt, antal punkter n i varje mönster, samt fönsterstorleken i vilken dessa punkter ska genereras. Dessa punktmönster har samlats genom att undersöka prover av hudbiopsin, vilket är små fragment av hud som skurits ut från patienter. Datan består av två grupper, en grupp med 8 friska patienter, kallad grupp 1, samt grupp 2 som innehåller data från 6 patienter som lider av diabetesneuropati. Vi kommer i detta kapitel börja med att analysera den givna nervdatan för att se hur mycket punktmönstren skiljer sig från en homogen, slumpmässig poissonprocess, samt i vilken riktning; är de mer klustrade än CSR eller mindre?

4.1 Förundersökning av nervdata

Vi vill undersöka hur punktmönstrerna för de två grupperna av data ser ut i jämförelse med CSR och vill därför plotta liknande figurer med `envelopes` som vi gjort i tidigare avsnitt. För detta måste vi veta intensiteten av ändpunktsprocessen i de två grupperna.

Intensiteten av processen ges av $\lambda = \kappa\mu$, vilket ger det förväntade antalet punkter per enhetsyta. Vi använder enhetskvadrat i simuleringarna och därför kan λ uppskattas med antal punkter i

mönstret delat med fönstrets storlek. Vi sammanställer medelantalet punkter i alla grupper. Antalet punkter för patient j i grupp i betecknas av n_{ij}, där $i \in \{1, 2\}$ och $j \in \{1, 2, 3, 4, 5, 6, 7, 8\}$.

Grupp 1		
Patient 1	n_{11}	111
Patient 2	n_{12}	144
Patient 3	n_{13}	154
Patient 4	n_{14}	143
Patient 5	n_{15}	113
Patient 6	n_{16}	175
Patient 7	n_{17}	157
Patient 8	n_{18}	72
Medelantal	$\bar{n}_{1.}$	142.4

Tabell 1: *Antal punkter i grupp 1, rödmarkerade exkluderade i beräkningen av medelantal.*

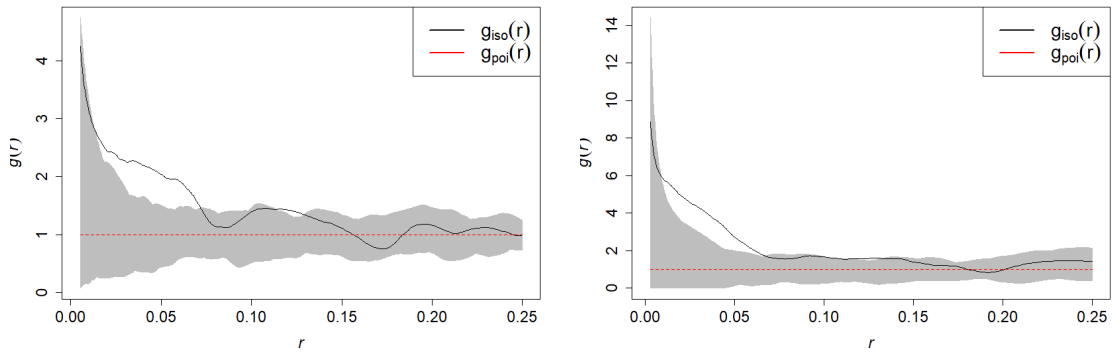
Grupp 2		
Patient 1	n_{21}	153
Patient 2	n_{22}	113
Patient 3	n_{23}	83
Patient 4	n_{24}	94
Patient 5	n_{25}	30
Patient 6	n_{26}	101
Medelantal	$\bar{n}_{2.}$	108.8

Tabell 2: *Antal punkter i grupp 2, rödmarkerade exkluderade i beräkningen av medelantal.*

För att inte få missvisande medelvärden behöver vi ta bort avvikelser. Eftersom vi har ett väldigt litet dataset med endast 14 patienter så får vi vara försiktiga med att eliminera allt för många avvikande värden. Vi nöjer oss med att ta bort det mest extrema fallen i respektive grupp, patient 8 i grupp 1 och patient 5 i grupp 2. Vi har därmed inte tagit hänsyn till dessa avvikelser i beräkningarna av medelvärdena $\bar{n}_{1.}$ och $\bar{n}_{2.}$. Nervdatan är skalad, se appendix B.1, och de skalade fönsterdimensionerna är $[0, 1.31] \times [0, 1]$. Detta betyder att den uppskattade intensiteten för de två grupperna är $\lambda_1 = \frac{\bar{n}_{1.}}{1.31}$ respektive $\lambda_2 = \frac{\bar{n}_{2.}}{1.31}$.

4.2 Visualisering av nervdata

För att se hur varje enskilt punktmönster ser ut hänvisar vi till appendix D.1. Notera att datan där är skalad, enligt beskrivning i appendix B.1. Det kan ges en bättre uppfattning om hur klus-



(a) *Grupp 1.* (b) *Grupp 2.*

Figur 17: *Envelopes för PCF för grupp 1 (vänster) och grupp 2 (höger). Notera att skalan på y-axeln skiljer sig åt avsevärt mellan grupp 1 och grupp 2.*

tertrenderna ser ut för de olika grupperna genom att generera och studera figurer som visar den uppskattade statistikan, antingen Ripleys K-funktion eller PCF, för ett punktmönster tillsammans med `envelopes` för CSR, precis som vi redogjort för i avsnitt 2.7. Målet är att kunna jämföra de två grupperna och få en tydligare bild av hur de förhåller sig till CSR. Vi visar figurer för de genomsnittliga poolade uppskattningarna av PCF för den samlade datan i grupp 1 respektive grupp 2, se figur 17 och D.2 i appendix.

Kom ihåg att vad PCF visar är antalet par av punkter som befinner sig på radie r från varandra, i förhållande till vad som förväntas för CSR. Det vi kan se i figurerena är att det verkar som nervmönsterna för båda grupperna är generellt mer klustrade än CSR. Framförallt kan vi se att för grupp 1 finns det en klustertrend i intervallet $r \in [0.01, 0.07]$ från en godtycklig punkt, medan för grupp 2 verkar klustertrenden vara något snävare inom intervall $r \in [0.00, 0.05]$. Dessutom är avvikelsen från CSR större i grupp 2 än grupp 1. Tolkningen av detta blir att vi kan förvänta oss ungefär lika stora kluster, det vill säga antal dotterpunkter, för de två grupperna, men att grupp 2 genererar något tätare och mer separerade kluster än grupp 1.

4.3 Anpassning av thomasprocessen till nervdata

Förundersökningen visar att nervpunktmönstrerna är klustrade och därför kan thomasprocessen vara en bra modell för ändpunkterna. Vi skattar parametrarna av thomasprocessen genom att använda både Ripleys K-funktion och PCF för varje patient i varje grupp, resultatet är presenterat i tabellerna 3 och 4. Notera återigen att nervdatan är skalad, vilket betyder att dessa skattningar för σ och κ inte är detsamma som för datans originalstorlek, utan betydligt mindre. Se appendix B.1 för mer information om skalningen. Notera även att värdena i tabellen nedan är utifrån enhetsfönstret $[0, 1] \times [0, 1]$. Vid första anblick av resultaten lägger vi märke till att den största

Grupp 1						
	Ripleys			PCF		
Patient	$\hat{\mu}$	$\hat{\kappa}$	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\kappa}$	$\hat{\sigma}$
1	3.65061	30.40585	0.03974	2.72907	40.67326	0.02851
2	1.59493	90.28592	0.02061	1.71094	84.16438	0.01831
3	3.48571	44.18040	0.03328	2.49982	61.60454	0.02325
4	2.27386	62.88867	0.02209	1.79656	79.59639	0.01660
5	6.61844	17.07352	0.04996	6.58899	17.14983	0.04500
6	8.53601	20.50139	0.05050	5.15993	33.91520	0.03015
7	4.70917	33.33922	0.03523	2.54671	61.64821	0.01868
8	3.20799	22.44392	0.03298	3.17679	22.66436	0.03060
Medel	4.25959	40.13986	0.03555	3.27612	50.17702	0.02639

Tabell 3: *Parameterskattningar för grupp 1.*

Grupp 2						
	Ripleys			PCF		
Patient	$\hat{\mu}$	$\hat{\kappa}$	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\kappa}$	$\hat{\sigma}$
1	6.81699	15.26133	0.03895	6.75069	21.73420	0.02686
2	5.03477	63.21135	0.01496	4.98580	76.92758	0.01166
3	3.69811	20.46026	0.02143	3.66214	19.22366	0.02020
4	4.18822	20.44933	0.02899	4.14748	20.14053	0.02675
5	1.33667	18.83654	0.02727	1.32366	15.97330	0.02628
6	4.50011	12.41746	0.03316	4.45634	17.23940	0.02292
Medel	4.26248	22.44392	0.03298	4.22102	22.66436	0.03060

Tabell 4: *Parameterskattningar för grupp 2.*

skillnaden mellan grupperna verkar vara gällande parametern κ. Minns att κ är intensiteten av föräldrapunktprocessen, vilket i vår data motsvarar antal basnerver i enhetsfönstret. Vi kan se att antalet basnerver verkar var större i grupp 1, friska patienter, än i grupp 2, sjuka patienter, vilket troligen är grund av att för patienter drabbade av diabetesneuropati så dör en del av nerverna. Hela

nervträdet, dvs. föräldrapunkter och dotterpunkter, försvinner. Vi ser att parameterskattningarna för μ, dvs. det förväntade antalet dotterpunkter per kluster, inte verkar skilja sig anmärkningsvärt mellan de olika skattnigsmetoderna eller grupperna i datan. Detta är förenligt med den analys som gjordes i avsnittet ovan med stöd av figur 17.

Slutligen, om vi vänder blickarna mot paramterskattningarna av parametern σ så verkar det även här inte vara någon avsevärd stor skillnad mellan grupperna. Dessutom fastställdes i avsnitt 3.1 att det pålitliga intervallet gällande parameterskattningar för PCF var $I_{\sigma,P} = [0.02, 0.03]$ och då våra skattningar ligger precis på den övre gränsen kanske vi inte ska förlita oss helt på dem utan fokusera mer på skattningarna gjorda med Ripleys K-funktion. Där skiljer sig skattningarna av σ mellan grupp 1 och 2 minimalt. Skulle den skillnaden vara signifikant så stöder resultatet hypotesen om att grupp 2 har tätare kluster än grupp 1.

4.4 Utvärdering av thomasprocessen som modell för nervdata

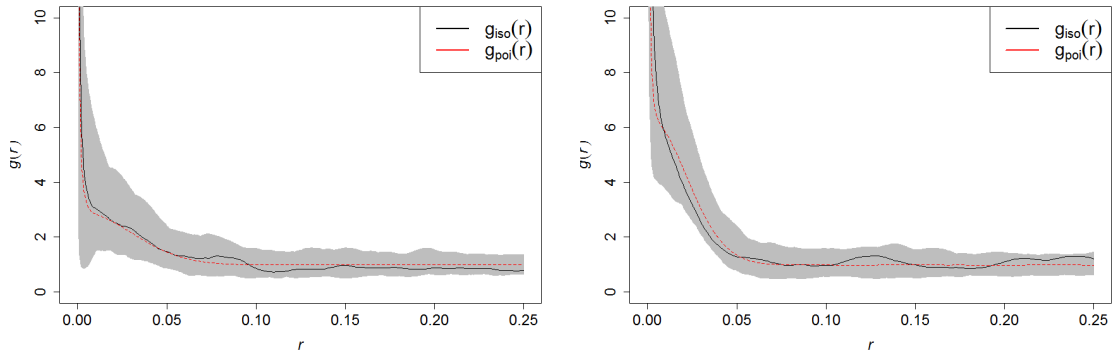
I detta avsnitt ska vi undersöka om thomasprocessen är en bra modell för vår data. Vi har valt att göra detta på två sätt. Den första metoden bygger på att göra en grafisk jämförelse genom upprepade simuleringar av thomasprocessen där vi använder de skattade parametrarna för nervdatan enligt tabellerna 3 och 4.

Den andra metoden bygger på konstruktion av Monte Carlo test baserat på PCF [4]. Denna metod är bra för att kunna kvantifiera *hur* bra thomasprocessen anpassar punktmönstrena i nervdatan. Vi kommer i detta test att basera teststatistikan på PCF och bifoga de för Ripleys K-funktion i appendix. Metoden är i grund och botten ett hypotestest där vi beräknar p-värdet för att avgöra om nollhypotesen, som i detta fall är att ändpunktmönstret kommer från thomasprocessen med skattade parametervärden, bör förkastas.

4.4.1 Grafisk utvärdering med hjälp av envelopes

Utvärderingen görs genom att utföra 9999 Monte Carlo simuleringar och uppskatta en teststatistika för varje patients skattade parametrar, enligt tabeller 3 och 4. Sedan skapar vi `envelopes` som representerar ett intervall för vart vi förväntar oss att kurvan ska ligga för ett punktmönster med nervdatans skattade parametrar. Till sist skapar vi en figur som visar dessa `envelopes` tillsammans med PCF uppskattad för patienten i fråga och undersöker hur de placerar sig i förhållande till varandra. Om kurvan faller inom området som illustrerar acceptansintervallet av vad vi förväntar oss kan vi anta att thomasprocessen är en god modell för datan. Vi undersöker dessa grafer för varje patient i de två grupperna.

För att undvika ett cirkulärt resonemang väljer vi att använda skattningarna som gjordes med hjälp av Ripleys K-funktion och sedan använda PCF som teststatistika för att utvärdera modellen.



(a) *Kurvor motsvarande grupp 1.*

(b) *Kurvor motsvarande grupp 2.*

Figur 18: *De skattade PCF för två patienter (en frisk till vänster och en sjuk till höger) från nervdatan tillsammans med envelopes för motsvarande simulerad data, samt en röd linje som representerar den genomsnittliga kurvan för thomasprocessen med motsvarande parametrar.*

Vi valde ut ovanstående figurer från ovannämnda process för att illustrera trenden vi såg bland resultaten. Övriga figurer finns att hitta i Appendix D.2. Genom att observera figurerna ovan verkar det som vår data har liknande mönster som den data som är resultat från en thomasprocess, eftersom kurvan faller inom det förväntade området samt ligger nära den teoretiska röda linjen som ovan betecknas med $g_{poi}(r)$. Detta stämde för alla patienter och vi kan säga att thomasprocessen är en bra modell för denna nervdata.

4.4.2 Utvärdering via Monte Carlo-test

Vi formulerar nollhypotesen enligt

$$\mathcal{H}_0 = \text{Punktmönstren från nervdatan följer en thomasprocess.} \tag{20}$$

Genom att använda oss av ekvationerna (17), (18) i avsnitt 2.6 kan vi bara sätta in PCF istället för $S(t)$. Vi får då

$$G_i = \sum_{r\in[0,\min(a,b)]} \left[\hat{g}_i(r) - g_i(r : \hat{\kappa}_i, \hat{\mu}_i)\right]^2 \tag{21}$$

och efter insättning av det teoretiska uttrycket för $g(r)$ erhålles

$$G_i = \sum_{r\in[0,\min(a,b)]} \left[\hat{g}_i(r) - 1 - \frac{r^2}{4\pi\hat{\kappa}_i\hat{\sigma}_i^2}\exp\left(-\frac{r^2}{4\hat{\sigma}_i^2}\right)\right]^2. \tag{22}$$

För G_{obs} har vi

$$G_{\text{obs}} = \sum_{r\in[0,\min(a,b)]} \left[\hat{g}(r) - g(r : \hat{\kappa}, \hat{\mu})\right]. \tag{23}$$

Slutligen beräknar vi p-värdet genom ekvation (19) där vi förkastar $\mathcal{H}_0$ utifrån signifikansnivån $\alpha = 0.05$. Resultaten presenteras nedan i tabell 5 och 6. Det är tydligt att vi i samtliga fall inte kan förkasta $\mathcal{H}_0$.

Grupp 1	p-värde
Patient 1	0.41
Patient 2	0.57
Patient 3	0.50
Patient 4	0.50
Patient 5	0.77
Patient 6	0.49
Patient 7	0.63
Patient 8	0.61

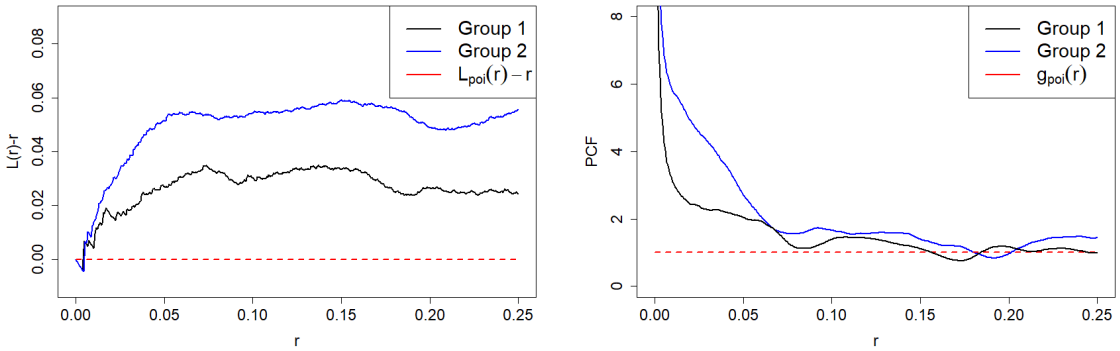
Tabell 5: *p-värden för grupp 1.*

Grupp 2	p-värde
Patient 1	0.87
Patient 2	0.64
Patient 3	0.45
Patient 4	0.69
Patient 5	0.50
Patient 6	0.63

Tabell 6: *p-värden för grupp 2.*

4.5 Sammanfattning av resultaten från analysen av nervdata

Figurerna nedan jämför de poolade teststatistikorna mot varandra för de två grupperna av patienter.



(a) *Ripleys K-funktion.* (b) *PCF.*

Figur 19: *Jämförelse av de två sammanfattande statistikorna, Ripleys K-funktion och PCF, för de två grupperna.*

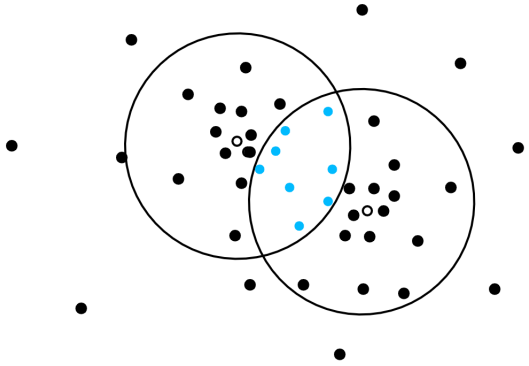
Genom att observera dessa kurvor kan vi även sammanfatta det tidigare delar av detta avsnitt konstaterat. Båda grupperna är klart klustrade och det är inte ett klart felaktigt antagande att säga att de kommer från en thomasprocess. Vi kan även konstatera att grupp 2 verkar har betydligt färre basnerver, föräldrapunkter, än grupp 1 samt mer tätare och separerade kluster. Det är dock svårt att få fram några intervall som skulle kunna användas för att avgöra om ett nervmönster kommer från en frisk eller sjuk patient eftersom vi har blivit försedda med en väldigt liten datamängd.

5 Sammanfattning

Som resultatet av simuleringsstudien visar i sektion 3.2 ger Ripleys K-funktion och PCF olika intervall för vilka värden på σ som är acceptabla. Vid simulering med paremetervärdena $(\kappa, \mu) = (22.9, 4)$ så bör σ ligga i $I_{K,\sigma}$ eller $I_{P,\sigma}$ för att MCM-algoritmerna `thomas.estK` och `thomas.estpcf` ska konvergera. För Ripleys K-funktion är intervallet $I_{\sigma,K} = [0.02, 0.09]$ och för PCF är intervallet $I_{\sigma,P} = [0.02, 0.03]$, vid simulering i enhetskvadraten. Att Ripleys K-funktion har ett längre intervall för acceptabla värden för σ betyder att mer stabila och säkra uppskattningar kan göras med Ripleys K-funktion och är därför att föredra framför PCF.

När vi varierar σ så varieras det förväntade avståndet mellan föräldrapunkt och berörda dotterpunkter. När små värden på σ används, givet fixerade κ, μ, kommer varje kluster att vara tätare och hela punktmönstret kommer att se mer klumpat ut. Anledningen till att algoritmerna börjar konvergera dåligt och ge sämre skattningar när vi ökar σ över ett visst värde är att dotterpunkterna till varje kluster börjar överlappa. I ett idealt fall där varje kluster är en cirkel där radien är avståndet från föräldrapunkten till den yttersta dotterpunkten så betyder detta att dessa cirklar överlappar när σ blir stort. Då fås en yta där två cirklar överlappar, en så kallad *vesica piscis*, se figur 20. När dotterpunkterna befinner sig i denna yta så har algoritmen svårt att veta till vilken förälderpunkt given dotterpunkt tillhör, vilket ger upphov till felskattningar. Dessutom kan algoritmerna heller inte identifiera vilka av punkterna som är dotter- respektive föräldrapunkter och då ger det fel i skattningar för κ. När man får fel skattning på κ skattas även μ fel eftersom $\hat{\mu} = \lambda/\hat{\kappa}$.

Figur 14 i sektion 3, samt figurerna i appendix D.3 visar att simuleringarna ger en hel del avvikande värden. Vid skattning av parametrarna genom att använda ett medelvärde jämfört med ett medianvärde ger detta stora skillnader, vilket kan ses i appendix C. Detta är ett välkänt fenomen inom statistiken och generellt blir skattningarna mycket bättre när ett medianvärde används istället



Figur 20: *Överlappande kluster som delar på dotterpunkter. De ifyllda punkterna är dotterpunkter och ringarna är föräldrapunkter. Notera att det inte är möjligt att veta vilken föräldrapunkt de blå dotterpunkterna tillhör.*

för ett medelvärde, just på grund av att det finns avvikande värden med extremt höga värden. Detta är något som är viktigt att ha i åtanke när man till exempel vill hitta de gemensamma skattningarna i grupp 1 respektive grupp 2, vid användning av medelvärdet eller medianvärdet kan resultatet skilja sig avsevärt.

Som nämns i 4.4 är thomasprocessen ett bra sätt att representera nervdatan. Detta trots att vi bara haft data på antal dotterpunkter, utan någon information angående föräldrapunkterna.

Resultatet i sektion 4.5 visar att grupp 2, som var patienter drabbade av diabetesneuropati, generellt hade tätare kluster av nervtrådar än de patienter från grupp 1 samt färre baspunkter vilket minskar antalet totala nervändpunkter. Detta stämmer väl överens med den tidigare forskning som gjorts på diabetesneuropati och som nämns i sektion 1. Utifrån de resultaten vi kommit fram till i denna studie är det dock väldigt svårt att kunna dra några slutsatser huruvida en enskild individ lider av diabetesneuropati eller inte, något som också diskuteras i sektion 4.5. Detta beror främst på att den givna datamängden är väldigt liten, och det är därför svårt att göra en noggrann dataanalys, eftersom det i nuläget är svårt att kvantifiera värden på κ, μ och σ för att kunna åtskilja individer med och utan sjukdomen diabetesneuropati.

I simuleringsstudien valde vi värdena för intensiteten av föräldrapunkter κ och det genomsnittliga antalet dotterpunter per kluster μ så att de liknar värdena i nervdatan så att det är möjligt att använda resultaten av simuleringsstudien för tolkning av resultaten baserat på nervdatan. Det skulle vara intressant att undersöka hur skattningarna beter sig om andra värden hade valts för κ och μ. Om vi skulle fortsätta detta arbete skulle vi utföra en större simuleringsstudie med flera värden för de tre parametrarna, och eventuellt kunnat ta fram exakt hur mycket σ får skilja sig i förhållande till de andra parametrarna. Det skulle också ha varit intressant att använda de verkliga antalen föräldrapunkter och se om detta hade ändrat skattningarna av de två andra parametrarna i nervdatan.

Referenser

[1] Ulf Adamsson. *Diabetesneuropati*. 2013. URL: https://www.netdoktorpro.se/diabetes/medicinska-oversikter/diabetesneuropati/. Accessed: 2020-02-08.

[2] Claes Andersson, Peter Guttorp och Aila Särkkä. "Discovering early diabetic neuropathy from epidermal nerve fiber patterns". I: (2016). DOI: 10.1002/sim.7009.

[3] Adrian Baddeley. *mincontrast*. URL: https://www.rdocumentation.org/packages/spatstat/versions/1.61-0/topics/mincontrast. Accessed: 2020-02-24.

[4] Julian Besag och Peter J. Diggle. "Simple Monte Carlo Tests for Spatial Pattern". I: *Journal of the Royal Statistical Society* 26.3 (1977), s. 327–333. DOI: 10.1080/01621459.1975.10480272.

[5] Markus J. Buehler. *Property calculation II*. 2011. URL: https://ocw.mit.edu/courses/materials-science-and-engineering/3-021j-introduction-to-modeling-and-simulation-spring-2012/part-i-lectures-readings/MIT3_021JS12_P1_L4.pdf.

[6] Peter J. Diggle. *Statistical Analysis of Spatial and Spatio-Temporal Point Patterns*. 2014, s. 92.

[7] Philip M. Dixon. "Ripley's K-function". I: *Encyclopedia of Environmetrics* 3.3 (2002), s. 1796–1803. DOI: https://doi.org/10.1002/9780470057339.var046.

[8] Giuseppe Espa, Diego Giuliani och Giuseppe Arbia. "Weighting Ripley's K-function to account for the firm dimension in the analysis of spatial concentration". I: (2010).

[9] Tim Johnson. *Introduction to Spatial Point Processes*. 2010. URL: https://warwick.ac.uk/fac/sci/statistics/staff/academic-research/nichols/research/spatbayes/Johnson_SpatialPointProc.pdf.

[10] Tim Johnson. *Introduction to Spatial Point Processes*. 2010. URL: https://warwick.ac.uk/fac/sci/statistics/staff/academic-research/nichols/research/spatbayes/Johnson_SpatialPointProc.pdf. Accessed: 2020-03-24.

[11] Dale W. Jorgenson. "Multiple Regression Analysis of a Poisson Process". I: *Journal of the American Statistical Association* 56.294 (1961), s. 235–245. DOI: https://doi.org/10.1080/01621459.1961.10482106.

[12] Christian Källgren, Hampus Lane och Simon Eriksson. "Modellering av nervmönster med spatiala punkt- processer". I: (2017).

[13] Günter Last och Mathew Penrose. *Lectures on the Poisson Process*. 2018.

[14] Jesper Moller och Rasmus Plenge Waagepetersen. *Statistical Inference and Simulation for Spatial Point Processes*. 2003.

[15] *MVEX01-20-13 Punktprocesser och nervtrådar*. URL: https://www.chalmers.se/sv/institutioner/math/utbildning/grundutbildning-chalmers/examensarbeten-och-kandidatprojekt/Kandidatprojekt_2020/Sidor/MVEX01-20-13.aspx. Accessed: 2020-02-08.

[16] Lance Waller, Aila Särkkä, Viktor Olsbo, Mari Myllymäki, Ioanna Panoutsopoulou, William Kennedy och Gwen Wendelschafer-Crabb. "Second-order spatial analysis of epidermal nerve fibers". I: *Statistics in medicine* 30 (okt. 2011), s. 2827–41. DOI: 10.1002/sim.4315.

A Härledning av statistikor för thomasprocessen

A.1 Ripleys K-funktion

Här presenteras en härledning för den teoretiska K-funktionen för thomasprocessen [12]. Ripleys K-funktion definieras enligt följande ekvation,

$$K(r) = \lambda^{-1}\mathbb{E}[N_0(r)], \tag{24}$$

där λ är antalet punkter per enhetsarea, intensiteten, och $N_0(r)$ är antalet punkter hos processen inom ett avstånd r till en godtycklig punkt av processen. Antag att vi har en punkt, x, som är placerad i origo och tillhör ett kluster som vi benämner c. Väntevärdet för området med centrum i x och radie r beror på antalet förväntade punkter från kluster c, $\mathbb{E}[N_{0,c}(r)]$, och antalet förväntade punkter från andra kluster, $\mathbb{E}[N_{0,-c}(r)]$. Därmed är det möjligt att skriva väntevärdet som en summa av två väntevärden,

$$\mathbb{E}[N_0(r)] = \mathbb{E}[N_{0,c}(r)] + \mathbb{E}[N_{0,-c}(r)]. \tag{25}$$

Den andra termen beror inte på något specifikt kluster utan beror bara på arean, πr^2, och intensiteten, $\lambda = \kappa\mu$. Den andra termen i (25) kan därmed skrivas

$$\mathbb{E}[N_{0,-c}(r)] = \pi r^2 \kappa\mu.$$

För den första termen i (25) använder vi lagen om sammanlagd förväntan

$$\mathbb{E}[N_{0,c}(r)] = \sum_{k=2}^{\infty} \mathbb{E}[N_{0,c}(r)|K_c = k]P(K_c = k). \tag{26}$$

Vi summerar väntevärdet av $N_{0,c}(r)$ givet att klustret är av storlek k multiplicerad med sannolikheten för att klustret är av storlek k. Anledningen till att k börjar från 2 är att ett kluster med 0 punkter är inte rimligt, och när $k = 1$ så är väntevärdet lika med 0, då vi bara räknar ytterligare punkter. $N_{0,c}(r)|K_c = k$ är därmed antalet ytterligare punkter som tillhör kluster c inom en radie r givet att det finns totalt k punkter i klustret. Sannolikheten för att dessa $k-1$ ytterligare punkter befinner sig inom den radien är givet av en fördelningsfunktion, $F(r)$, för avståndet mellan två punkter som befinner sig i samma kluster. Eftersom att varje punkt är oberoende av varandra så är $N_{0,c}(r)|K_c = k$ binomialfördelad med $k-1$ utfall med sannolikheten $F(r)$ för varje utfall. Väntevärdet av en variabel med binomialfördelning är antalet utfall multiplicerat med sannolikheten för varje utfall,

$$\mathbb{E}[N_{0,c}(r)|K_c = k] = (k-1)F(r). \tag{27}$$

Den andra faktorn i (26), $P(K_c = k)$, är samma som att multiplicera k med sannolikheten att ett kluster är av storlek k, $P(D_c = k)$. Men en sannolikhetsfördelning kräver att $\sum_{k=1}^{\infty} P(K_c = k) = 1$. Därför delar vi $P(K_c = k)$ för $k = 1, 2, ...$ med $\sum_{l=1}^{\infty} lP(D_c = l)$, vilket ger

$$\sum_{k=1}^{\infty} P(K_c = k) = \frac{\sum_{k=1}^{\infty} kP(D_c = k)}{\sum_{l=1}^{\infty} lP(D_c = l)}.$$

Men då $\sum_{l=1}^{\infty} lP(D_c = l) = \mathbb{E}[D_c] = \mu$ är det möjligt att förenkla uttrycket på följande sätt

$$P(K_c = k) = \frac{kP(D_c = k)}{\mu} \tag{28}$$

Givet (27), (28) och faktumet att $D_c \sim \mathcal{P}(\mu)$ kan (26) utvecklas,

$$\begin{aligned}
\mathbb{E}[N_{0,c}(r)] &= \sum_{k=2}^{\infty} \mathbb{E}[N_{0,c}(r)|K_c = k]P(K_c = k) \\
&= \sum_{k=2}^{\infty} (k-1)F(r)\frac{kP(D_c = k)}{\mu} \\
&= \frac{F(r)}{\mu} \sum_{k=2}^{\infty} (k-1)kP(D_c = k) \\
&= \frac{F(r)}{\mu} \mathbb{E}[D_c(D_c - 1)] \\
&= \frac{F(r)}{\mu} (\mu^2 + \mu - \mu) \\
&= \mu F(r).
\end{aligned} \tag{29}$$

För att utveckla $F(r)$, avståndet mellan två oberoende och normalfördelade variabler, krävs ytterligare en härledning. Anta att $X = (x_1, x_2)$ och $Y = (y_1, y_2)$ är två punkter i $\mathbb{R}^2$, där x_1, x_2, y_1 och y_2 är oberoende och $\mathcal{N}(a, \omega^2)$, som ger

$$\begin{aligned}
F(r) &= P(||X - Y|| \leq r) \\
&= P(||X - Y||^2 \leq r^2) \\
&= P((x_1 - y_1)^2 + (x_2 - y_2)^2 \leq r^2).
\end{aligned} \tag{30}$$

Då $x_1 - y_1$ och $x_2 - y_2$ båda har väntevärdet 0 och variansen $2\omega^2$, kan vi multiplicera dem med inversen av variansen, som då ger en ny fördelning, $\mathcal{N}(0, 1)$ innanför kvadraterna som beskrivs av (30).

$$P((x_1 - y_1)^2 + (x_2 - y_2)^2 \leq r^2) = P\left(\left(\frac{x_1 - y_1}{\sqrt{2}\omega}\right)^2 + \left(\frac{x_2 - y_2}{\sqrt{2}\omega}\right)^2 \leq \frac{r^2}{2\omega^2}\right). \tag{31}$$

Summan av kvadraterna i högerledet av ekvation (31) visar en χ^2-fördelad variabel med två frihetsgrader,

$$\left(\frac{x_1 - y_1}{\sqrt{2}\omega}\right)^2 + \left(\frac{x_2 - y_2}{\sqrt{2}\omega}\right)^2 = Q \sim \chi_2^2.$$

Fördelningsfunktionen för en χ^2-fördelad variabel med två frihetsgrader är följande,

$$P\left(Q \leq \frac{r^2}{2\omega^2}\right) = 1 - \exp\left(\frac{-r^2}{4\omega^2}\right). \tag{32}$$

Sista steget är att koppla samman $\mathbb{E}[N_{0,-c}(r)] = \pi r^2 \kappa\mu$, (29), (32), med $\omega = \sigma$,

$$\begin{aligned}
K(r) &= \frac{1}{\kappa\mu}(\mathbb{E}[N_{0,c}(r)] + \mathbb{E}[N_{0,-c}(r)]) \\
&= \pi r^2 + \frac{1}{\kappa} F(r) \\
&= \pi r^2 + \frac{1}{\kappa}\left(1 - \exp\left(-\frac{r^2}{4\sigma^2}\right)\right).
\end{aligned} \tag{33}$$

A.2 Parkorrelationsfunktionen

Här presenteras än härledning av den teoretiska PCF för thomasprocessen. PCF definieras enligt

$$g(r) = \frac{K'(r)}{2\pi r}, \tag{34}$$

där

$$K(r) = \pi r^2 + \frac{1}{\kappa}(1 - \exp\left(\frac{-r^2}{4\sigma^2}\right)). \tag{35}$$

Detta leder till att

$$K'(r) = 2\pi r + \frac{2r}{4\kappa\sigma^2}\exp\left(\frac{-r^2}{4\sigma^2}\right), \tag{36}$$

vilket ger oss den teoretiska funktionen för PCF i en thomasprocess enligt:

$$g(r) = \frac{K'(r)}{2\pi r} = \frac{2\pi r + \frac{2r}{4\kappa\sigma^2}\exp\left(\frac{-r^2}{4\sigma^2}\right)}{2\pi r} = 1 + \frac{1}{4\pi\kappa\sigma^2}\exp\left(\frac{-r^2}{4\sigma^2}\right). \tag{37}$$

B Simulering

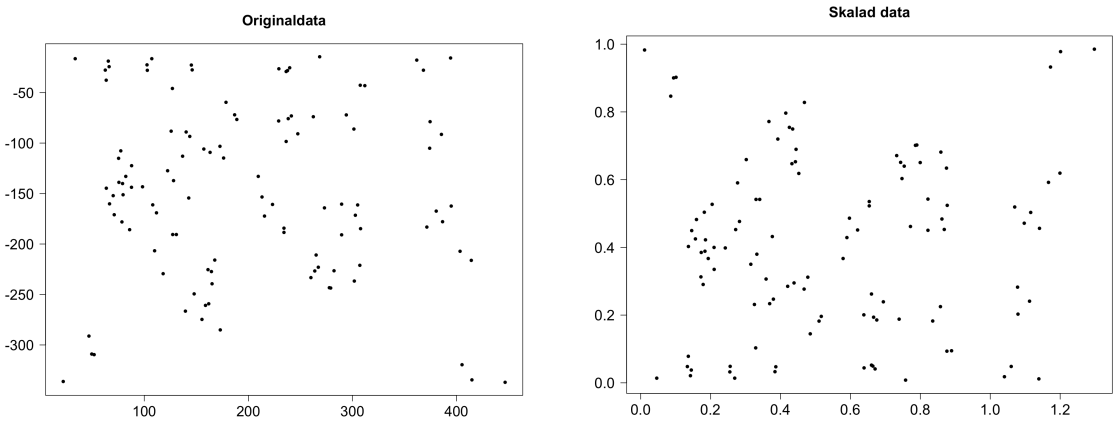
B.1 Ändring av data och skalning av simuleringsfönster

I genereringen av en realisation av en thomasprocess kräver algoritmen dimensionerna av det fönster i vilken punkterna genereras. Som standard tar funktionen `rThomas` in parametern `c(0,1), c(0,1)`, vilket betyder att punkterna genereras inom enhetskvadraten $[0,1] \times [0,1]$. För att på ett lämpligt sätt kunna simulera ny data som liknar den givna nervdatan vill vi ha dem i ungefär samma storlek. Den givna datan har flera brister som vi måste åtgärda för att kunna uppnå detta. Först och främst måste vi se till att alla koordinater ligger i den första kvadranten, dvs. endast positiva koordinater. Därför skalar vi de nervmönster med negativa koordinater med -1, vilket resulterar i att koordinaterna speglas i motsvarande axel. Detta kommer inte påverka resultatet av våra skattningar. Sedan vill vi se till att alla fönster börjar i origo och sträcker sig till samma x- respektive y-gräns. Vi väljer dessa gränser och skjuter koordinaterna i x- och y-led så att alla punktmönster får plats i samma fönster, nämligen $[0, 433] \times [0, 330]$. Slutligen skalar vi fönstret och koordinaterna med $\frac{1}{330}$ längs båda axlarna. Resultatet blir då att nervdatan nu ligger i fönstret $[0, 1.31] \times [0, 1]$. Detta är så nära den enhetskvadraten vi kan komma, eftersom att skala om nervdatan till en kvadrat påverkar skattningen av σ på ett sätt som vi vill undvika.
Vi kommer hädanefter låta $\tilde{\lambda}$ beteckna det totala antalet dotterpunkter i fönstret $[0, 1.31] \times [0, 1]$ och därmed, eftersom $\tilde{\lambda} = \tilde{\kappa}\mu$, kommer vi låta $\tilde{\kappa}$ beteckna antalet föräldrapunkter i fönstret $[0, 1.31] \times [0, 1]$. För $\tilde{\mu} = \mu$ blir det ingen skillnad då det inte är relaterat till storleken av fönstret. Detta betyder att när vi ska mata in värden för κ och σ i `rThomas` tvingas vi dividera de parametrar vi uppskattat utifrån datan med 1.31. Dessa skalade parametrar kommer refereras till som κ och σ.

För att sammanfatta detta stycke matematiskt låter vi den generiska rektangeln i vilken punkterna genereras vara Ω och dess area $|\Omega|$. Då gäller

$$\kappa = \frac{\tilde{\kappa}}{|\Omega|}, \quad \Omega = [0, A] \times [0, B], \quad |\Omega| = AB, \quad A, B \in \mathbb{R}, \tag{38}$$

Nedan i Figur B.21 kan vi se bilder på nervdatan före och efter skalning.



(a) Innan skalning.

(b) Efter skalning.

Figur B.21: Data från person 1, grupp 1, innan och efter skalning.

Avslutningsvis måste noteras att när vi ska utföra de slutgiltiga analyserna av parametrarna så kommer vi vilja skala om de tillbaka till det ursprungliga dimensionen av datan. Exempelvis kommer intensiteten av föräldrapunkter i det stora fönstret $[0, 433] \times [0, 330]$ vara $\kappa^* = \frac{\kappa}{433 \times 330}$, där κ är totala antalet föräldrapunkter i nervmönstret.

B.2 Framtagning av lämpliga parametrar för simuleringsstudien

I simuleringsstudien användes det totala medelvärdet för båda grupper, det vill säga

$$\bar{n} = \frac{\bar{n}_{1.} + \bar{n}_{2.}}{2} \approx 125.6. \tag{39}$$

Detta är också det förväntade antalet dotterpunkter $\tilde{\lambda}$ för thomasprocessen i det givna fönstret. Eftersom vi simulerar i fönstret $[0,1]\times[0,1]$ vill vi använda oss av λ istället. Detta kan åstadkommas genom att först bestämma $\tilde{\kappa}$. För att bättre förstå hur fönsterstorleken påverkar skattningarna, se appendix B.1.

Eftersom $\tilde{\lambda} = \tilde{\kappa}\mu$ gäller det att hitta lämpliga $\tilde{\kappa}$ och μ sådana att deras produkt kommer nära $\tilde{\lambda} = 125.6$. Parametern $\tilde{\kappa}$ representerar antalet föräldrapunkter och μ representerar antalet dotterpunkter till varje föräldrapunkt. Varje föräldrapunkt är i själva verket en baspunkt utifrån vilken nervtrådar förgrenar sig från, alltså måste det i de flesta fall gälla att antal föräldrapunkter är färre än det totala antalet dotterpunkter. Ett rimligt värde på μ ligger mellan 2 och 5 [2]. Genom att välja ett $\mu \in [2,\ 5]$ så kan vi ta fram ett rimligt $\tilde{\kappa}$ utifrån $\tilde{\lambda} = \tilde{\kappa}\mu$. Vi har valt $\mu = 4$ och får därmed $\tilde{\kappa} = 30 \implies \kappa = \frac{30}{1.31} \approx 22.9$, vilket slutligen ger $\lambda = 22.9 \cdot 4 \approx 91.6$. Notera att parametrarna $\lambda = 91.6,\ \kappa = 22.9$ och $\mu = 4$ är de parametrar som nervdatan genomsnittligt hade givit i thomasprocessen om det istället varit ett $[0,1]\times[0,1]$ fönster och det är dessa vi matade in algoritmen `rThomas` som in-parametrar under simuleringsstudien.

C Tabeller och resultat

Resultat av skattningar från simuleringsstudien i tabellform.

Verkliga värden	$\kappa = 22.9$	$\mu = 4$	$\sigma = 0.01 \ : \ 0.15$
Skattningar	$\hat{\kappa}$	$\hat{\mu}$	$\hat{\sigma}$
$\sigma = 0.01$	24.15359	24.417317	0.03083419
$\sigma = 0.02$	25.45370	9.843589	0.02875202
$\sigma = 0.03$	26.15853	11.495781	0.04218288
$\sigma = 0.04$	26.98548	7.130926	0.04560641
$\sigma = 0.05$	29.92780	12.640477	0.06679906
$\sigma = 0.06$	31.18656	17.959941	0.09168543
$\sigma = 0.07$	34.23076	29.757148	0.16791205
$\sigma = 0.08$	56.22653	77.464241	0.49188860
$\sigma = 0.09$	91.61842	131.678098	0.99224079
$\sigma = 0.10$	155.52405	187.217000	2.03331761
$\sigma = 0.11$	308.54949	206.823284	3.84776921
$\sigma = 0.12$	471.68262	254.867052	5.92067473
$\sigma = 0.13$	666.46080	266.121395	8.18679975
$\sigma = 0.14$	944.84811	267.339053	10.97474268
$\sigma = 0.15$	1298.63805	222.187319	15.07363424

Tabell C.7: *Parameterskattningar med K-funktionen då medelvärde används.*

Verkliga värden	$\kappa = 22.9$	$\mu = 4$	$\sigma = 0.01 \ : \ 0.15$
Skattningar	$\hat{\kappa}$	$\hat{\mu}$	$\hat{\sigma}$
$\sigma = 0.01$	25.62470	5.681309	0.08201874
$\sigma = 0.02$	24.38073	6.661124	0.03834710
$\sigma = 0.03$	25.96076	5.521146	0.03024794
$\sigma = 0.04$	28.54149	3.887598	0.03760578
$\sigma = 0.05$	36.60449	4.273384	0.10420623
$\sigma = 0.06$	49.78715	4.388362	0.11217682
$\sigma = 0.07$	102.24685	11.059789	0.37786868
$\sigma = 0.08$	214.59063	20.681136	0.78503836
$\sigma = 0.09$	295.34809	35.965411	1.08958946
$\sigma = 0.10$	457.87318	71.349599	2.50063923
$\sigma = 0.11$	540.95338	109.523013	3.87994809
$\sigma = 0.12$	935.75361	105.090919	7.60277767
$\sigma = 0.13$	1054.10587	131.930662	8.17112979
$\sigma = 0.14$	1410.07767	144.104112	11.82033020
$\sigma = 0.15$	1558.85421	147.439841	13.94213757

Tabell C.8: *Parameterskattningar med PCF då medelvärde används.*

Verkliga värden	$\kappa = 22.9$	$\mu = 4$	$\sigma = 0.01 \ : \ 0.15$
Skattningar	$\hat{\kappa}$	$\hat{\mu}$	$\hat{\sigma}$
$\sigma = 0.01$	24.026030	3.760774	0.009809099
$\sigma = 0.02$	24.619311	3.638875	0.019179392
$\sigma = 0.03$	25.414828	3.587787	0.028593227
$\sigma = 0.04$	24.690064	3.567788	0.039147788
$\sigma = 0.05$	26.064837	3.513402	0.048603808
$\sigma = 0.06$	25.082254	3.611746	0.060528256
$\sigma = 0.07$	24.479339	3.621957	0.070765998
$\sigma = 0.08$	22.003507	4.047569	0.083422713
$\sigma = 0.09$	20.338054	4.404147	0.098905060
$\sigma = 0.10$	15.130373	5.892428	0.129802895
$\sigma = 0.11$	13.353137	6.674042	0.175497927
$\sigma = 0.12$	11.785156	7.465746	0.284439483
$\sigma = 0.13$	11.636988	7.545923	0.581603998
$\sigma = 0.14$	7.048926	12.079488	0.981315831
$\sigma = 0.15$	8.092667	10.399924	1.410125572

Tabell C.9: *Parameterskattningar med K-funktionen då median används.*

Verkliga värden	$\kappa = 22.9$	$\mu = 4$	$\sigma = 0.01 \ : \ 0.15$
Skattningar	$\hat{\kappa}$	$\hat{\mu}$	$\hat{\sigma}$
$\sigma = 0.01$	22.91234	3.945210	0.009904673
$\sigma = 0.02$	23.23429	3.830248	0.018518627
$\sigma = 0.03$	24.59410	3.733664	0.027270609
$\sigma = 0.04$	26.42808	3.421231	0.035787127
$\sigma = 0.05$	28.58298	3.114991	0.042898650
$\sigma = 0.06$	29.63581	3.055611	0.050125561
$\sigma = 0.07$	31.29329	2.865039	0.057859330
$\sigma = 0.08$	35.63664	2.507547	0.063570632
$\sigma = 0.09$	34.40629	2.667989	0.073551943
$\sigma = 0.10$	38.09664	2.321308	0.077499501
$\sigma = 0.11$	42.82919	2.128642	0.082632651
$\sigma = 0.12$	40.33463	2.124513	0.092541965
$\sigma = 0.13$	46.68532	1.835994	0.102312497
$\sigma = 0.14$	59.41088	1.525845	0.103795704
$\sigma = 0.15$	83.45480	1.105411	0.115755801

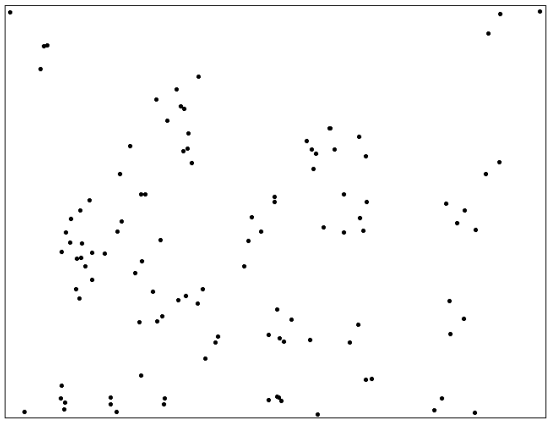
Tabell C.10: *Parameterskattningar med PCF då median används.*

D Figurer

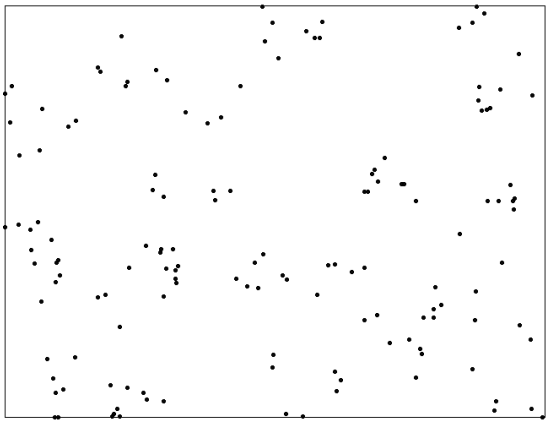
D.1 Punktmönster från nervdatan

I sektion D.1.1 och D.1.2 presenteras en visualisering av nervdatan från `endpoints.rds` för gruppen utan diabetesneuropati (grupp 1) respektive gruppen med diabetesneuropati (grupp 2).

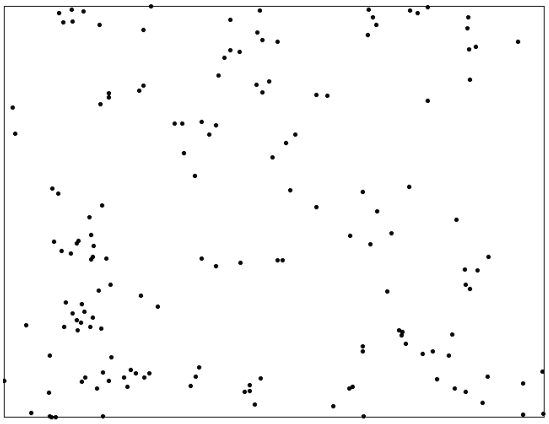
D.1.1 Grupp 1



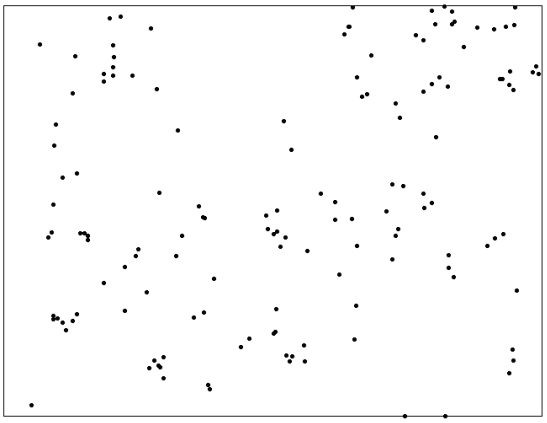
(a) *Punktmönster för Grupp 1 - Patient 1.*



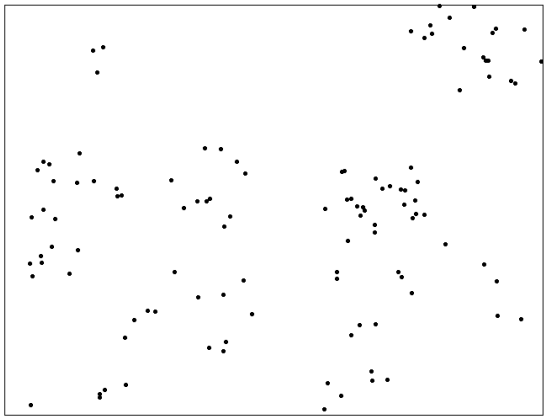
(b) *Punktmönster för Grupp 1 - Patient 2.*



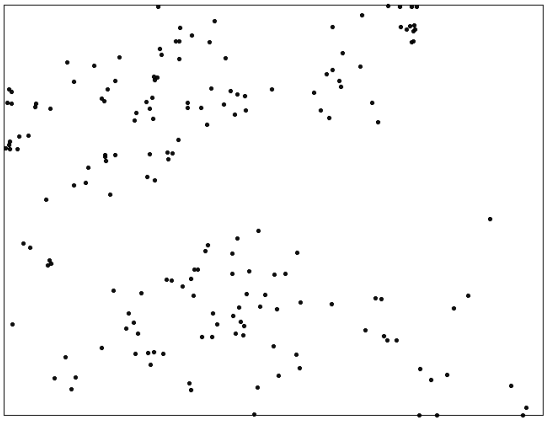
(c) *Punktmönster för Grupp 1 - Patient 3.*



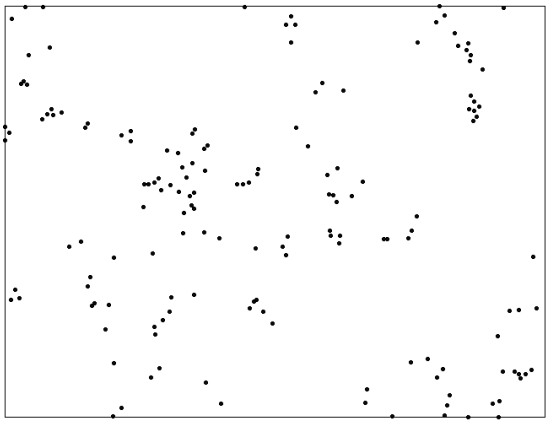
(d) *Punktmönster för Grupp 1 - Patient 4.*



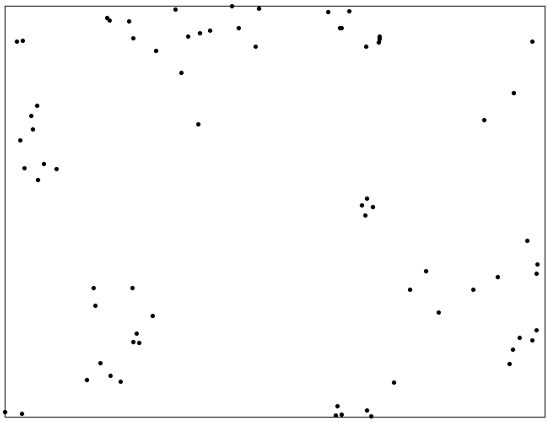
(e) *Punktmönster för Grupp 1 - Patient 5.*



(f) *Punktmönster för Grupp 1 - Patient 6.*

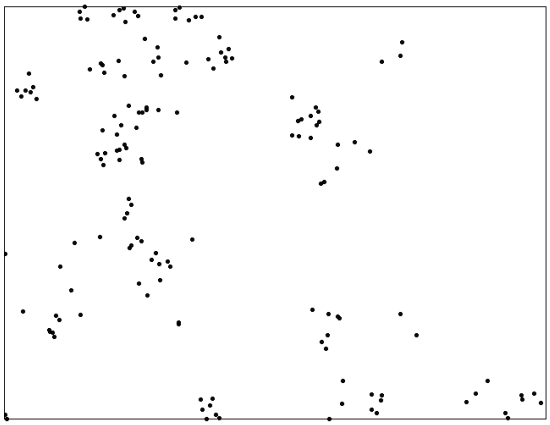


(g) *Punktmönster för Grupp 1 - Patient 7.*

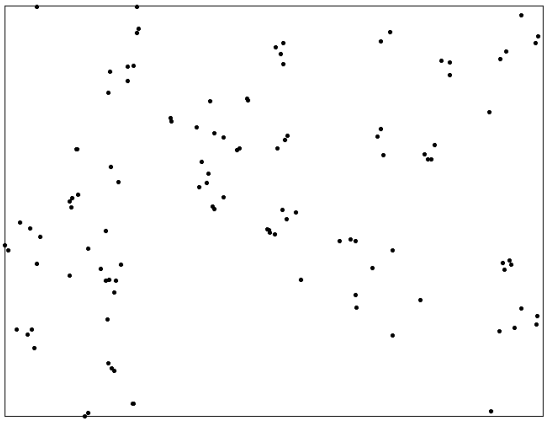


(h) *Punktmönster för Grupp 1 - Patient 8.*

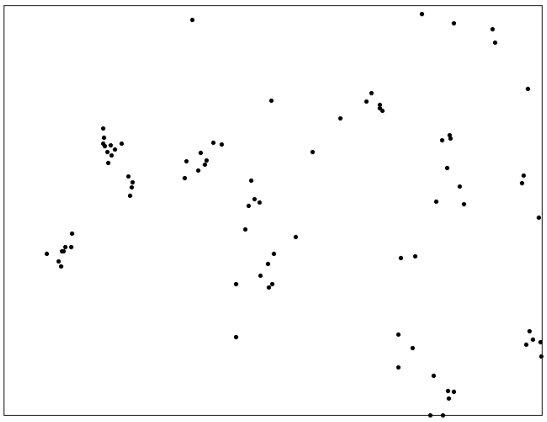
D.1.2 Grupp 2



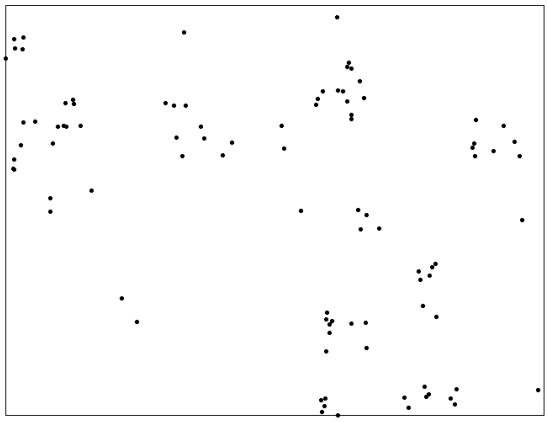
(i) *Punktmönster för Grupp 2 - Patient 1.*



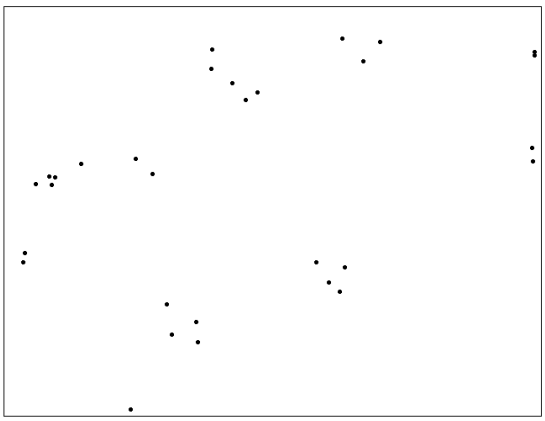
(j) *Punktmönster för Grupp 2 - Patient 2.*



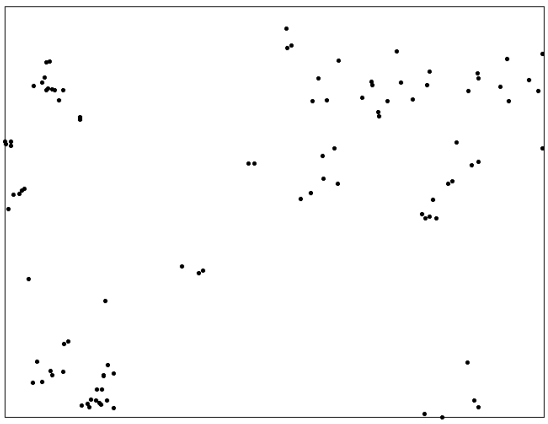
(k) *Punktmönster för Grupp 2 - Patient 3.*



(l) *Punktmönster för Grupp 2 - Patient 4.*



(m) *Punktmönster för Grupp 2 - Patient 5.*

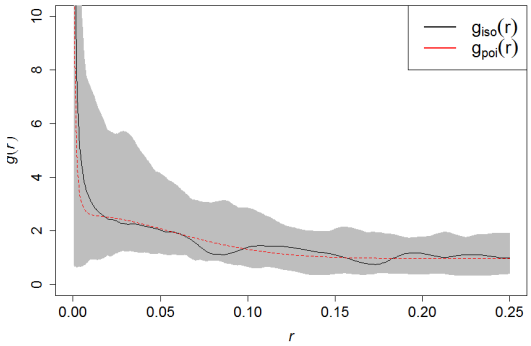


(n) *Punktmönster för Grupp 2 - Patient 6.*

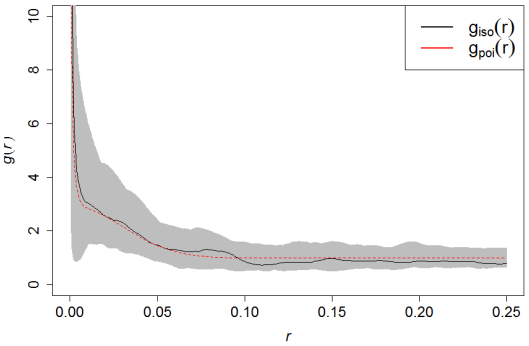
D.2 Grafisk undersökning för anpassningen av thomasprocessen

I sektion D.2.1 och D.2.2 presenteras resultatet över den skattade PCF för patienterna i grupp 1 repsektive grupp 2, tillsammans med `envelopes` från den simulerade datan och en röd linje som visar den genomsnittliga kurvan för thomasprocessen med motsvarande parametrar.

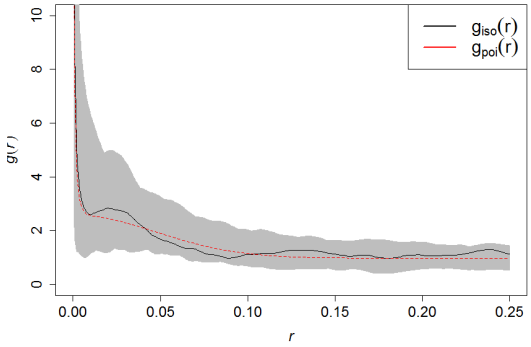
D.2.1 Grupp 1



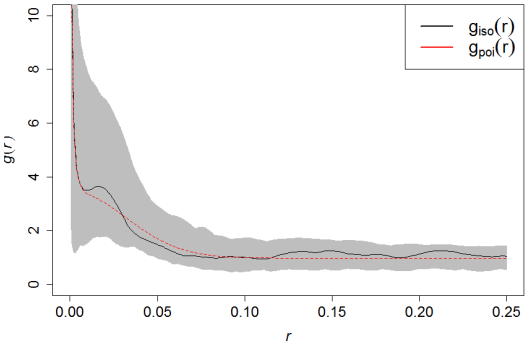
(a) *Grupp 1 - Patient 1.*



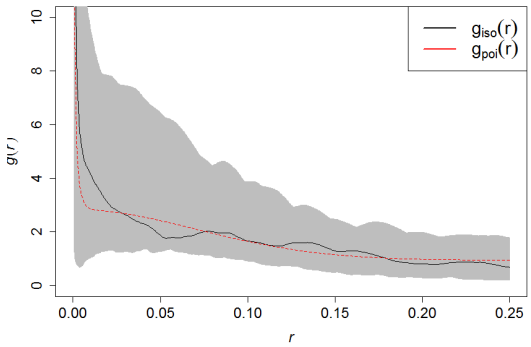
(b) *Grupp 1 - Patient 2.*



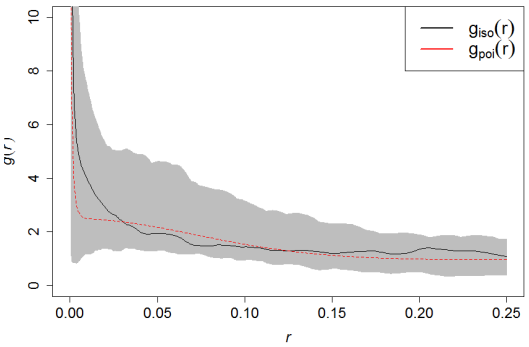
(c) *Grupp 1 - Patient 3.*



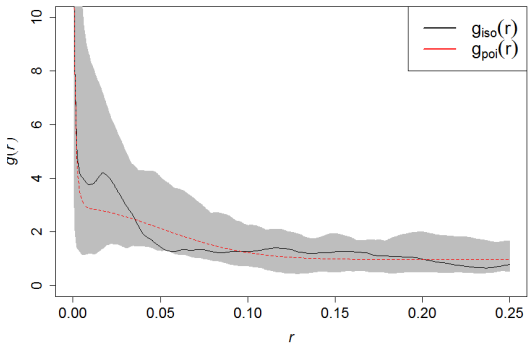
(d) *Grupp 1 - Patient 4.*



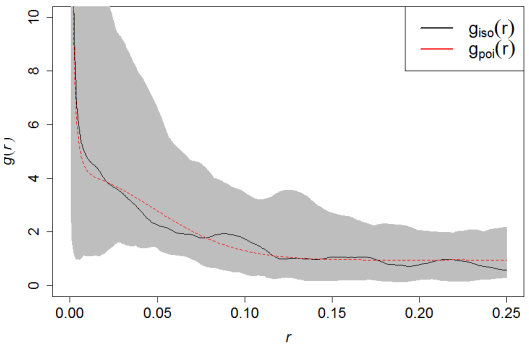
(e) *Grupp 1 - Patient 5.*



(f) *Grupp 1 - Patient 6.*

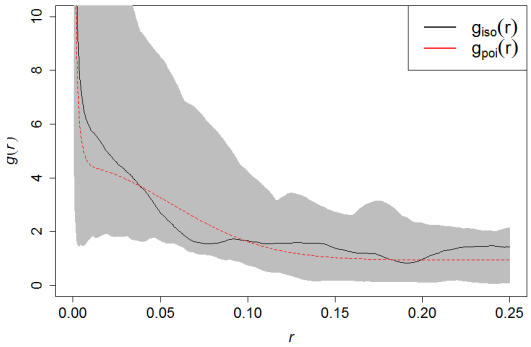


(g) *Grupp 1 - Patient 7.*

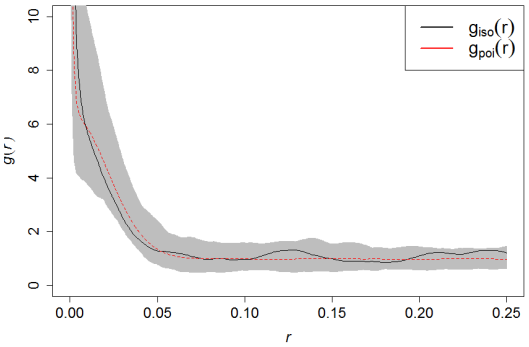


(h) *Grupp 1 - Patient 8.*

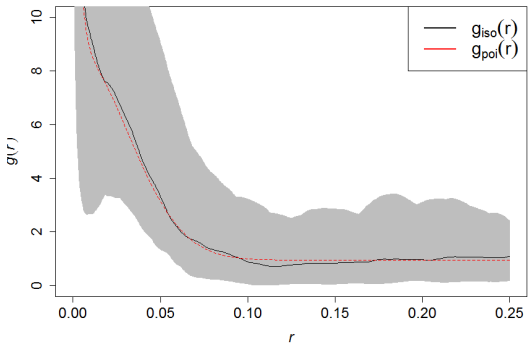
D.2.2 Grupp 2



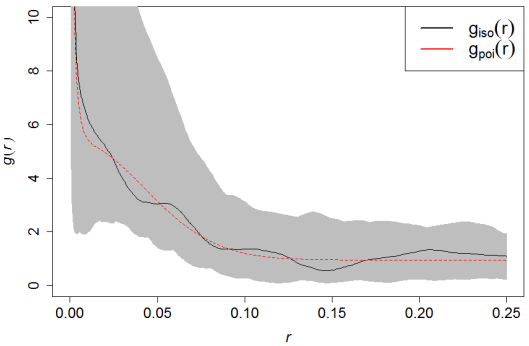
(i) *Grupp 2 - Patient 1.*



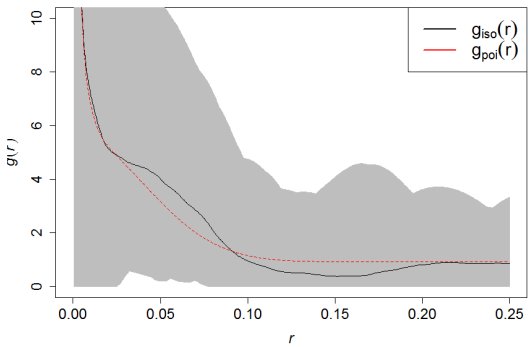
(j) *Grupp 2 - Patient 2.*



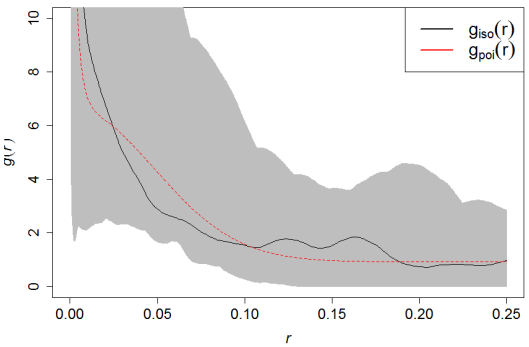
(k) *Grupp 2 - Patient 3.*



(l) *Grupp 2 - Patient 4.*



(m) *Grupp 2 - Patient 5.*

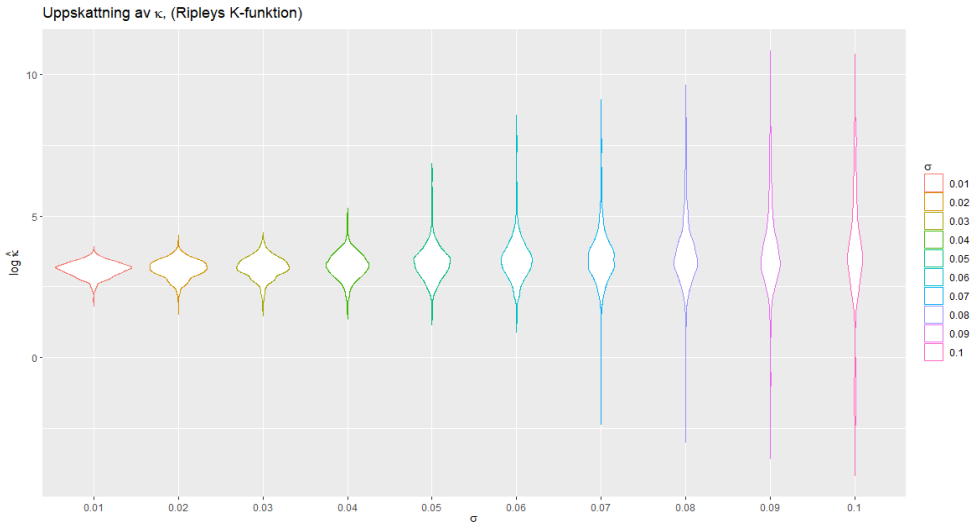


(n) *Grupp 2 - Patient 6.*

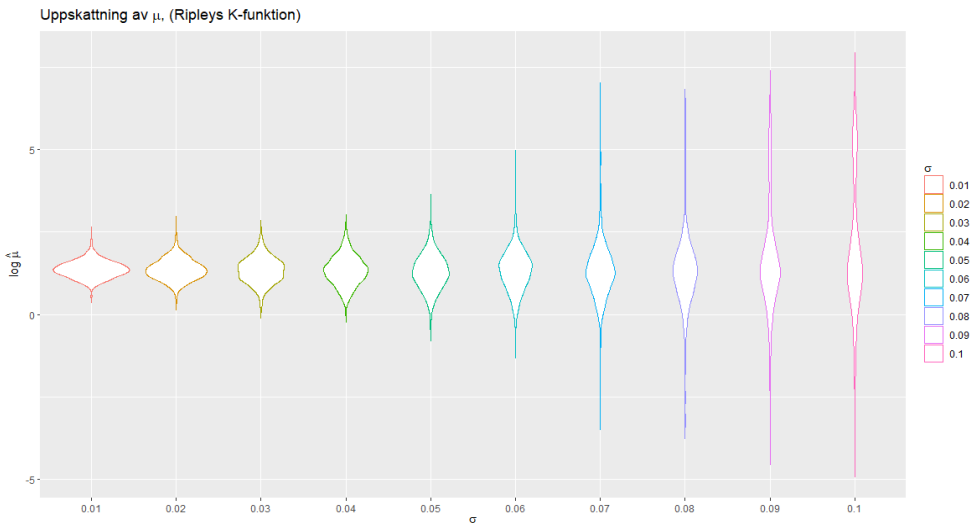
D.3 Violindiagram

I sektion D.3.1 och D.3.2 presenteras de skattade parametrarna $\hat{\kappa}$, $\hat{\mu}$ och $\hat{\sigma}$ från Ripleys K-funktion respektive PCF för att visa spridningen på skattningarna. Notera att y-axlarna är logaritmerade.

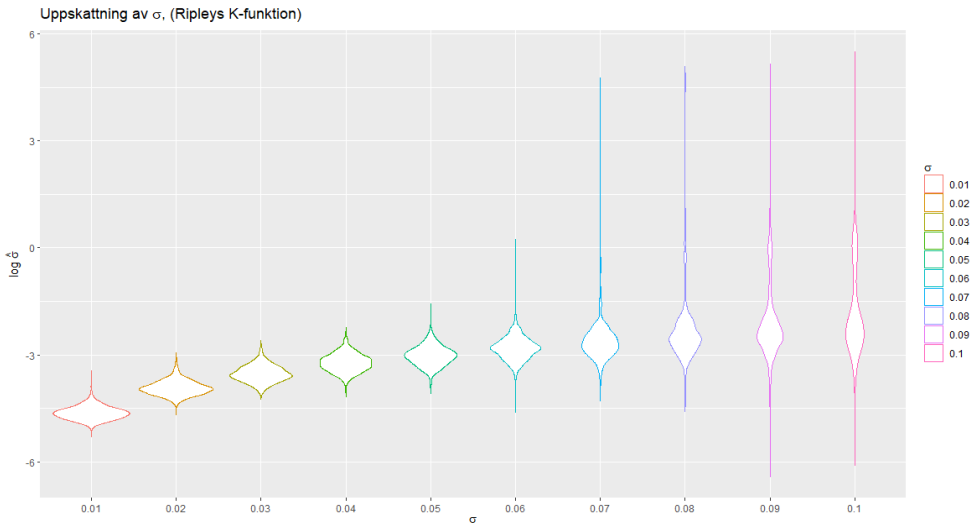
D.3.1 Ripleys K-funktion



Figur D.22: *Violindiagram över uppskattade värden för κ för olika värden på σ med Ripleys K-funktion.*

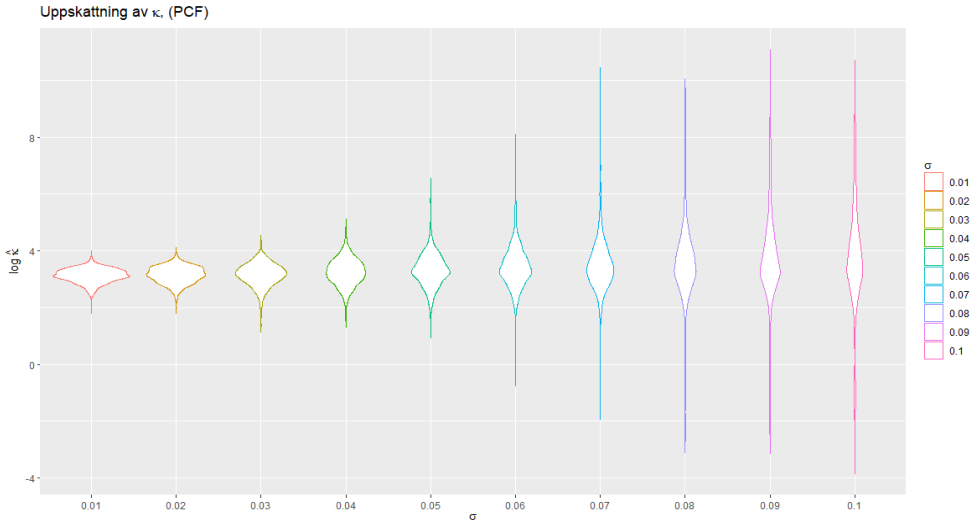


Figur D.23: *Violindiagram över uppskattade värden för μ för olika värden på σ med Ripleys K-funktion.*

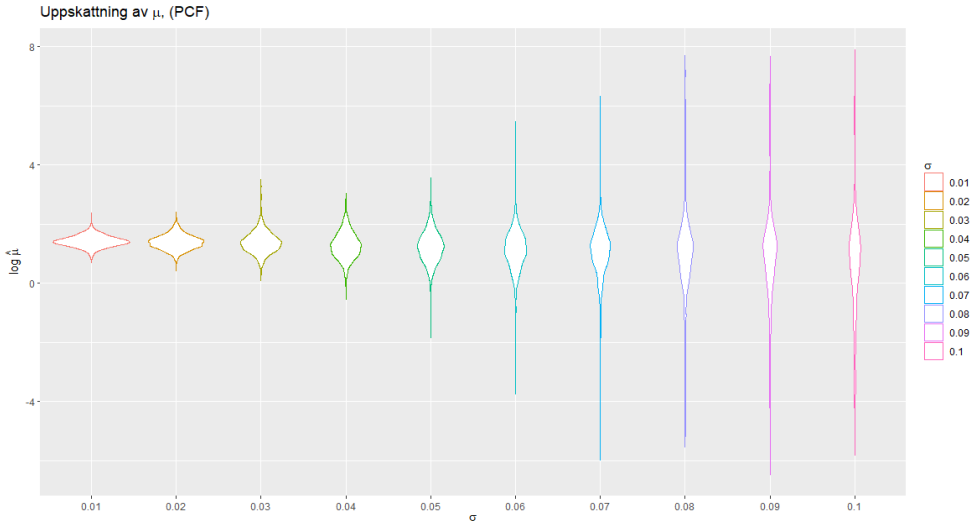


Figur D.24: *Violindiagram över uppskattade värden för σ för olika värden på σ med Ripleys K-funktion.*

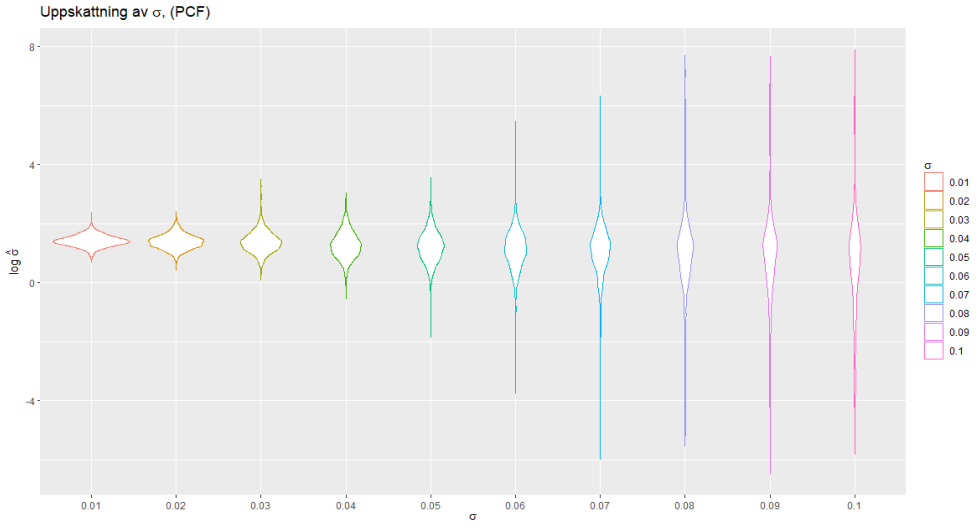
D.3.2 Parkorrelationsfunktionen



Figur D.25: *Violindiagram över uppskattade värden för κ för olika värden på σ med PCF.*



Figur D.26: *Violindiagram över uppskattade värden för μ för olika värden på σ med PCF.*



Figur D.27: *Violindiagram över uppskattade värden för σ för olika värden på σ med PCF.*

E Kod

E.1 Kod för histogram för skattningar då medelvärde används

```
library("latex2exp")
library("spatstat")

kappa = 22.9
mu = 4

Hist_mean = function(kappa, mu) {

  timesteps = 100
  kappaVector = vector("numeric", timesteps)
  sigmaVector = vector("numeric", timesteps)
  muVector = vector("numeric", timesteps)
  numberOfPoints = vector("numeric", timesteps)
  estimates = matrix(0, nrow = 15, ncol = 4)
  sigmas = seq(from = 0.03, to = 0.07, by = 0.04)

  for (ind in seq_along(sigmas)){

    for (i in 1:timesteps) {

      thomasSim = rThomas(kappa = kappa, scale = sigmas[ind], mu = mu, win = owin(c
    (0,1),c(0,1)))

      numberOfPoints[i] = thomasSim[["n"]]

      result = thomas.estK(thomasSim)

      kappaVector[i] = result$modelpar[[1]]
      sigmaVector[i] = result$modelpar[[2]]
      muVector[i] = numberOfPoints[i]/kappaVector[i]

    }

    hist(kappaVector,
         main = TeX('$\\sigma$ = 0.07$'),
         xlab = TeX('$\\kappa$'),
         ylab = "Frekvens",
         border = "blue",
         col = "grey",
         las = 1,
         breaks = 50)

    hist(muVector,
         main = TeX('$\\sigma = 0.07$'),
         xlab = TeX('$\\mu$'),
         ylab = "Frekvens",
         border = "blue",
         col = "grey",
         las = 1,
         breaks = 50)

  }
}

Hist_mean(22.9,4)
```

E.2 Skillnaden i avvikande datapunkter, median kontra medelvärde

```
library("latex2exp")
library("spatstat")

kappa = 22.9
mu = 4

ripleysMean = function(kappa, mu) {

  timesteps = 100
  kappaVector = vector("numeric", timesteps)
  sigmaVector = vector("numeric", timesteps)
  muVector = vector("numeric", timesteps)
  numberOfPoints = vector("numeric", timesteps)
  estimates = matrix(0, nrow = 15, ncol = 4)
  sigmas = seq(from = 0.03, to = 0.07, by = 0.04)

  for (ind in seq_along(sigmas)){

    for (i in 1:timesteps) {

      thomasSim = rThomas(kappa = kappa, scale = sigmas[ind], mu = mu, win = owin(c
    (0,1),c(0,1)))

      numberOfPoints[i] = thomasSim[["n"]]

      result = thomas.estK(thomasSim)

      kappaVector[i] = result$modelpar[[1]]
      sigmaVector[i] = result$modelpar[[2]]
      muVector[i] = numberOfPoints[i]/kappaVector[i]

    }

    hist(kappaVector,
         main = TeX('$\\sigma$ = 0.07$'),
         xlab = TeX('$\\kappa$'),
         ylab = "Frekvens",
         border = "blue",
         col = "grey",
         las = 1,
         breaks = 50)

    hist(muVector,
         main = TeX('$\\sigma = 0.07$'),
         xlab = TeX('$\\mu$'),
         ylab = "Frekvens",
         border = "blue",
         col = "grey",
         las = 1,
         breaks = 50)

  }
  print(mean(kappaVector))
  print(median(kappaVector))
  print(mean(muVector))
  print(median(muVector))
}

ripleysMean(22.9,4)
```

```
PCFMean = function(kappa, mu) {

  timesteps = 100
  kappaVector = vector("numeric", timesteps)
  sigmaVector = vector("numeric", timesteps)
  muVector = vector("numeric", timesteps)
  numberOfPoints = vector("numeric", timesteps)
  estimates = matrix(0, nrow = 15, ncol = 4)
  sigmas <- seq(from = 0.01, to = 0.15, by = 0.01)

  for (ind in seq_along(sigmas)){

    for (i in 1:timesteps) {

      thomasSim = rThomas(kappa = kappa, scale = sigmas[ind], mu = mu, win = owin(c
    (0,1),c(0,1)))

      numberOfPoints[i] = thomasSim[["n"]]

      result = thomas.estpcf(thomasSim)

      kappaVector[i] = result$modelpar[[1]]
      sigmaVector[i] = result$modelpar[[2]]
      muVector[i] = numberOfPoints[i]/kappaVector[i]

    }

    meanSigma = mean(sigmaVector)
    meanKappa = mean(kappaVector)
    meanMu = mean(muVector)

    estimates[ind,4] = meanSigma
    estimates[ind,3] = meanMu
    estimates[ind,2] = meanKappa
    estimates[ind,1] = sigmas[ind]

  }

  return(estimates)

}

PCFMean(22.9,4)
```

E.3 Kod för parameterskattningar då medelvärde används

```
############################## Run these first

require("spatstat")
require(ggplot2)
require(reshape2)

############################## Estimations for Ripleys with mean

rm(list = ls())
kappa = 22.9
mu = 4
timesteps = 100
kappaVector = vector("numeric", timesteps)
sigmaVector = vector("numeric", timesteps)
muVector = vector("numeric", timesteps)
numberOfPoints = vector("numeric", timesteps)
estimates = matrix(0, nrow = 15, ncol = 4)
sigmas = seq(from = 0.01, to = 0.15, by = 0.01)

for (ind in seq_along(sigmas)){

  for (i in 1:timesteps) {

    thomasSim = rThomas(kappa = kappa, scale = sigmas[ind], mu = mu, win = owin(c
    (0,1),c(0,1)))

    numberOfPoints[i] = thomasSim[["n"]]

    result = thomas.estK(thomasSim)

    kappaVector[i] = result$modelpar[[1]]
    sigmaVector[i] = result$modelpar[[2]]
    muVector[i] = numberOfPoints[i]/kappaVector[i]

  }

  meanSigma = mean(sigmaVector)
  meanKappa = mean(kappaVector)
  meanMu = mean(muVector)

  estimates[ind,4] = meanSigma
  estimates[ind,3] = meanMu
  estimates[ind,2] = meanKappa
  estimates[ind,1] = sigmas[ind]

}

print(estimates)
df = as.data.frame(estimates)
col_names = paste(c("Varying_sigma", "Kappa", "Mu", "Sigma"), sep = "")
names(df) = col_names
df_mod = melt(df ,  id.vars = 'Varying_sigma', variable.name = 'Skattningar')
ggplot(df_mod, aes(Varying_sigma, value)) + geom_line(aes(colour = Skattningar)) +
    ggtitle("Skattningar f r varierande sigma (Ripleys/Medelvrde)") + xlab("Sigma"
    ) + ylab("V rde ")

############################## Estimations for PCF with mean

rm(list = ls())
kappa = 22.9
mu = 4
timesteps = 1000
kappaVector = vector("numeric", timesteps)
sigmaVector = vector("numeric", timesteps)
muVector = vector("numeric", timesteps)
numberOfPoints = vector("numeric", timesteps)
estimates = matrix(0, nrow = 15, ncol = 4)
sigmas = seq(from = 0.01, to = 0.15, by = 0.01)
```

```
66
67 for (ind in seq_along(sigmas)){
68
69   for (i in 1:timesteps) {
70
71     thomasSim = rThomas(kappa = kappa, scale = sigmas[ind], mu = mu, win = owin(c
       (0,1),c(0,1)))
72
73     numberOfPoints[i] = thomasSim[["n"]]
74
75     result = thomas.estpcf(thomasSim)
76
77     kappaVector[i] = result$modelpar
78     sigmaVector[i] = result$modelpar[[2]]
79     muVector[i] = numberOfPoints[i]/kappaVector[i]
80
81   }
82
83   meanSigma = mean(sigmaVector)
84   meanKappa = mean(kappaVector)
85   meanMu = mean(muVector)
86
87   estimates[ind,4] = meanSigma
88   estimates[ind,3] = meanMu
89   estimates[ind,2] = meanKappa
90   estimates[ind,1] = sigmas[ind]
91
92 }
93
94 print(estimates)
95 df = as.data.frame(estimates)
96 col_names = paste(c("Varying_sigma", "Kappa", "Mu", "Sigma"), sep = "")
97 names(df) = col_names
98 df_mod = melt(df ,  id.vars = 'Varying_sigma', variable.name = 'Skattningar')
99 ggplot(df_mod, aes(Varying_sigma, value)) + geom_line(aes(colour = Skattningar)) +
       ggtitle("Skattningar f r varierande sigma (PCF/Medelvrde)") + xlab("Sigma") +
       ylab(" V rde ")
```

E.4 Kod för visualisering av statistikorna

```
library("spatstat")

# Generate CSR point pattern
PoissonCSR <- rThomas(mu = 4, kappa = 25, scale = 0.1, win = owin(c(0,1), c(0,1)))
png(filename = "pointpattern_CSR.png", bg = "white")
par(mar=c(0,0,0,0))
plot(PoissonCSR, pch = 20, main = "")
dev.off()

# Kest for CSR
CSR_Kest <- Kest(PoissonCSR, correction = "Ripley")
png(filename = "Kest_CSR.png", width = 800, bg = "white")
par(mar=c(5,6,1,1)+.1)
plot(CSR_Kest, main = "", xlab = "r", ylab = "K(r)", cex.lab = 3, cex.axis = 1.5,
    legend = NULL)
legend("topleft", c(expression(K[iso](r)),expression(K[poi](r))), col = c("black","
    red"), cex = 2, lty = 7, pt.cex = 2)
dev.off()

# Lest for CSR
CSR_Lest <- Lest(PoissonCSR, correction = "Ripley")
png(filename = "Lest_CSR.png", width = 800, bg = "white")
par(mar=c(5,6,1,1)+.1)
plot(CSR_Lest, main = "", xlab = "r", ylab = "L(r)", cex.lab = 3, cex.axis = 1.5,
    legend = NULL)
legend("topleft", c(expression(L[iso](r)),expression(L[poi](r))), col = c("black","
    red"), cex = 2, lty = 7, pt.cex = 2)
dev.off()

# L(r)-r for CSR
png(filename = "L(r)-r_CSR.png", width = 800, bg = "white")
par(mar = c(5,6,1,1)+.1)
plot(CSR_Lest,.-r~r, main = "",xlab = "r", ylab = "L(r)-r", cex.lab = 3, cex.axis =
     1.5, legend = NULL)
legend("bottomright", c(expression(L[iso](r)-r),expression(L[poi](r)-r)), col = c("
    black","red"), cex = 2, lty = 7, pt.cex = 2)
dev.off()

# Envelope for Lest CSR
Env_CSR_Lest <- envelope(PoissonCSR, fun = Lest, global = FALSE, correction = "
    Ripley")
png(filename = "Env_CSR_Lest.png", width = 800,height = 500, bg = "white")
par(mar=c(5,6,2,1)+.1)
plot(Env_CSR_Lest, .-r ~ r,main = "", xlab = "r", ylab = "L(r)-r", cex.lab = 3, cex
    .axis = 1.5, legend = c(""))
legend("bottomright", c(expression(L[iso](r)-r),expression(L[poi](r)-r),expression(
    L[hi](r)-r),expression(L[lo](r)-r)), col = c("black","red","grey80","grey80"),
    cex = 2, lty = 7, pt.cex = 2)
dev.off()


# PCF for CSR
CSR_PCF <- pcf(PoissonCSR, correction = "Ripley")
png(filename = "PCF_CSR.png", width = 800, bg = "white")
par(mar=c(5,6,1,1)+.1)
plot(CSR_PCF, main = "", xlab = "r", ylab = "g(r)", cex.lab = 3, cex.axis = 1.5,
    legend = NULL)
legend("topright", c(expression(g[iso](r)),expression(g[poi](r))), col = c("black",
    "red"), cex = 2, lty = 7, pt.cex = 2)
dev.off()

# Envelope for PCF CSR
Env_CSR_PCF <- envelope(PoissonCSR, fun = pcf, global = FALSE, correction = "Ripley
    ")
png(filename = "env_PCF_CSR.png", width = 800, bg = "white")
par(mar=c(5,6,2,1)+.1)
plot(Env_CSR_PCF, main = "",legend = c(""), xlab = "r", ylab = "g(r)", cex.lab = 3,
     cex.axis = 1.5, ylim = c(0,6))
```

```r
legend("topright", c(expression(g[iso](r)),expression(g[poi](r)),expression(g[hi](r
    )),expression(g[lo](r))), col = c("black","red","grey80","grey80"), cex = 2,
    lty = 7, pt.cex = 2)
dev.off()



# Generara klustrad punktprocess
PoissonClustered <- rThomas(mu = 4, kappa = 25, scale = 0.03, win = owin(c(0,1), c
    (0,1)))
png(filename = "pointpattern_clustered.png", bg = "white")
par(mar=c(0,0,0,0))
plot(PoissonClustered, pch = 20, main = "")
dev.off()

# Kest for Clustered
Clustered_Kest <- Kest(PoissonClustered, correction = "Ripley")
png(filename = "Kest_Clustered.png", width = 800, bg = "white")
par(mar=c(5,6,1,1)+.1)
plot(Clustered_Kest, main = "", xlab = "r", ylab = "K(r)", cex.lab = 3, cex.axis =
    1.5, legend = NULL)
legend("topleft", c(expression(K[iso](r)),expression(K[poi](r))), col = c("black","
    red"), cex = 2, lty = 7, pt.cex = 2)
dev.off()

# Lest for Clustered
Clustered_Lest <- Lest(PoissonClustered, correction = "Ripley")
png(filename = "Lest_Clustered.png", width = 800, bg = "white")
par(mar=c(5,6,1,1)+.1)
plot(Clustered_Lest, main = "", xlab = "r", ylab = "L(r)", cex.lab = 3, cex.axis =
    1.5, legend = NULL)
legend("topleft", c(expression(L[iso](r)),expression(L[poi](r))), col = c("black","
    red"), cex = 2, lty = 7, pt.cex = 2)
dev.off()

# L(r)-r for Clustered
png(filename = "L(r)-r_Clustered.png", width = 800, bg = "white")
par(mar = c(5,6,1,1)+.1)
plot(Clustered_Lest, .-r ~ r, main = "",xlab = "r", ylab = "L(r)-r", cex.lab = 3,
    cex.axis = 1.5, legend = c(""))
legend("topright", c(expression(L[iso](r)-r),expression(L[poi](r)-r)), col = c("
    black","red"), cex = 2, lty = 7, pt.cex = 2)
dev.off()

# Envelope for Lest Clustered
Env_Clustered_Lest <- envelope(PoissonClustered, fun = Lest, global = FALSE,
    correction = "Ripley")
png(filename = "Env_Clustered_Lest.png", width = 800, bg = "white")
par(mar=c(5,6,2,1)+.1)
plot(Env_Clustered_Lest, .-r ~ r,main = "", xlab = "r", ylab = "L(r)-r", cex.lab =
    3, cex.axis = 1.5, legend = c("","","",""), ylim = c(-0.04,0.05))
legend("bottomright", c(expression(L[iso](r)-r),expression(L[poi](r)-r),expression(
    L[hi](r)-r),expression(L[lo](r)-r)), col = c("black","red", "grey80","grey80"),
     cex = 2, lty = 7, pt.cex = 2)
dev.off()


# PCF for CSR
Clustered_PCF <- pcf(PoissonClustered, correction = "Ripley")
png(filename = "PCF_Clustered.png", width = 800, bg = "white")
par(mar=c(5,6,1,1)+.1)
plot(Clustered_PCF, main = "", xlab = "r", ylab = "g(r)",ylim = c(0,6), cex.lab =
    3, cex.axis = 1.5, legend = c(""))
legend("topright", c(expression(g[iso](r)),expression(g[poi](r))), col = c("black",
    "red"), cex = 2, lty = 7, pt.cex = 2)
dev.off()

# Envelope for PCF CSR
Env_Clustered_PCF <- envelope(PoissonClustered, fun = pcf, global = FALSE,
    correction = "Ripley")
```

```
109 png(filename = "env_PCF_Clustered.png", width = 800,height = 500, bg = "white")
110 par(mar=c(5,6,2,1)+.1)
111 plot(Env_Clustered_PCF, main = "", xlab = "r", ylab = "g(r)", cex.lab = 3, cex.axis
        = 1.5, ylim = c(0,6), legend = NULL)
112 legend("topright", c(expression(g[iso](r)),expression(g[poi](r)),expression(g[hi](r
    )),expression(g[lo](r))), col = c("black","red","grey80","grey80"), cex = 2,
    lty = 7, pt.cex = 2)
113 dev.off()
```

E.5 Kod för parameterskattningar då median används

```
############################ Run these first

require("spatstat")
require(ggplot2)
require(reshape2)

############################ Estimations for Ripleys with median

rm(list = ls())
kappa = 22.9
mu = 4
timesteps = 1000
kappaVector = vector("numeric", timesteps)
sigmaVector = vector("numeric", timesteps)
muVector = vector("numeric", timesteps)
numberOfPoints = vector("numeric", timesteps)
estimates = matrix(0, nrow = 15, ncol = 4)
sigmas = seq(from = 0.01, to = 0.15, by = 0.01)

for (ind in seq_along(sigmas)){

  for (i in 1:timesteps) {

    thomasSim = rThomas(kappa = kappa, scale = sigmas[ind], mu = mu, win = owin(c
    (0,1),c(0,1)))

    numberOfPoints[i] = thomasSim[["n"]]

    result = thomas.estK(thomasSim)

    kappaVector[i] = result$modelpar[[1]]
    sigmaVector[i] = result$modelpar[[2]]
    muVector[i] = numberOfPoints[i]/kappaVector[i]

  }

  medianSigma = median(sigmaVector)
  medianKappa = median(kappaVector)
  medianMu = median(muVector)

  estimates[ind,4] = medianSigma
  estimates[ind,3] = medianMu
  estimates[ind,2] = medianKappa
  estimates[ind,1] = sigmas[ind]

}

print(estimates)
df = as.data.frame(estimates)
col_names = paste(c("Varying_sigma", "Kappa", "Mu", "Sigma"), sep = "")
names(df) = col_names
df_mod = melt(df ,  id.vars = 'Varying_sigma', variable.name = 'Skattningar')
ggplot(df_mod, aes(Varying_sigma, value)) + geom_line(aes(colour = Skattningar)) +
    ggtitle("Skattningar f r varierande sigma (Ripleys/Median)") + xlab("Sigma") +
    ylab(" V rde ")

############################ Estimations for PCF with median

rm(list = ls())
kappa = 22.9
mu = 4
timesteps = 1000
kappaVector = vector("numeric", timesteps)
sigmaVector = vector("numeric", timesteps)
muVector = vector("numeric", timesteps)
numberOfPoints = vector("numeric", timesteps)
estimates = matrix(0, nrow = 15, ncol = 4)
sigmas = seq(from = 0.01, to = 0.15, by = 0.01)
```

```
66
67 for (ind in seq_along(sigmas)){
68
69   for (i in 1:timesteps) {
70
71     thomasSim = rThomas(kappa = kappa, scale = sigmas[ind], mu = mu, win = owin(c
       (0,1),c(0,1)))
72
73     numberOfPoints[i] = thomasSim[["n"]]
74
75     result = thomas.estpcf(thomasSim)
76
77     kappaVector[i] = result$modelpar[[1]]
78     sigmaVector[i] = result$modelpar[[2]]
79     muVector[i] = numberOfPoints[i]/kappaVector[i]
80
81   }
82
83   medianSigma = median(sigmaVector)
84   medianKappa = median(kappaVector)
85   medianMu = median(muVector)
86
87   estimates[ind,4] = medianSigma
88   estimates[ind,3] = medianMu
89   estimates[ind,2] = medianKappa
90   estimates[ind,1] = sigmas[ind]
91
92 }
93
94 print(estimates)
95 df = as.data.frame(estimates)
96 col_names = paste(c("Varying_sigma", "Kappa", "Mu", "Sigma"), sep = "")
97 names(df) = col_names
98 df_mod = melt(df ,  id.vars = 'Varying_sigma', variable.name = 'Skattningar')
99 ggplot(df_mod, aes(Varying_sigma, value)) + geom_line(aes(colour = Skattningar)) +
       ggtitle("Skattningar f r varierande sigma (PCF/Median)") + xlab("Sigma") + ylab
       (" V rde ")
```

E.6 Kod för K- och L-figurerna och deras envelopes

```
library(spatstat)
library(raster)
install.packages("maptools")
install.packages("Rtools")
library(maptools)

nclust = function(x0, y0, radius, n) {
  return(runifdisc(n, radius, centre=c(x0, y0)))
}

poisson1 = rPoissonCluster(40, 0.2, nclust, radius = 0.2, n = 5, win = owin(c
    (0,1.31),c(0,1)), lmax = NULL, nsim = 1, drop = TRUE, saveparents=TRUE)
poisson2 = rPoissonCluster(10, 0.2, nclust, radius = 0.2, n = 5, win = owin(c
    (0,1.31),c(0,1)), lmax = NULL, nsim = 1, drop = TRUE, saveparents=TRUE)

K_CSR1 = Kest(poisson1, correction = "Ripley")
plot(K_CSR1, font.axis = 6, ylab = "K", main = "K-funktion av CSR (N = 400)", font.
    main = 6)

K_CSR2 = Kest(poisson2, correction = "Ripley")
plot(K_CSR2, font.axis = 6, ylab = "K", main = "K-funktion av CSR (N = 100)", font.
    main = 6)

L_CSR1 = Lest(poisson1, correction = "Ripley")
plot(L_CSR1, font.axis = 6, ylab = "L", main = "L-funktion av CSR (N = 400)", font.
    main = 6)

L_CSR2 = Lest(poisson2, correction = "Ripley")
plot(L_CSR2, font.axis = 6, ylab = "L", main = "L-funktion av CSR (N = 100)", font.
    main = 6)

plot(L_CSR1, .-r ~ r, ylab = "L", main = "Modifierad L-funktion av CSR (N = 400)",
    font.main = 6)
plot(L_CSR2, .-r ~ r, ylab = "L", main = "Modifierad L-funktion av CSR (N = 100)",
    font.main = 6)

n = 10000
p  = 0.05 # Desired p significance level to display
EL = envelope(poisson2[owin(c(0,1.31),c(0,1))], Lest, nsim = n, rank=(p * (n + 1)))
OP = par(mar = c(5,5,4,4))
plot(EL, . - r ~ r, ylab = "L", xlab = "r", main=paste("p = ", 1-p), font.main = 6,
     font.axis = 6)
par(OP)




thomasSim = rThomas(50, scale = 0.01, mu = 10, win = owin(c(0,1.31),c(0,1)))
L_thomas = Lest(thomasSim, correction = "Ripley")
plot(L_thomas)
plot(thomasSim)


n = 10000
p  = 0.05 # Desired p significance level to display
ELT = envelope(thomasSim[owin(c(0,1.31),c(0,1))], Lest, nsim = n, rank=(p * (n + 1)
    ))
OP = par(mar = c(5,5,4,4))
plot(ELT, . - r ~ r, ylab = "L", xlab = "r", main=paste("p = ", 1-p), font.main =
    6, font.axis = 6)
par(OP)


n = 9999
Lm = matrix(nrow = n, ncol = length(L_CSR1$r))        # Define an empty matrix
for(i in 1:n){
  Pmc = runifpoint(poisson1$n, win = owin(c(0,1.31),c(0,1)))
  Lmc = Lest(Pmc, nsim = n, border = "Ripley")      # Compute L
```

```
  Lm[i,] = Lmc$iso - Lmc$r                              # Set Ltheo to 0 by rotating
    plot
}


# Create a density raster


# Create a ppp object from Lm
Lmp <- ppp(rep(L_CSR1$r, each = n), as.vector(Lm), window = owin( c(min(L_CSR1$r),
    max(L_CSR1$r)), c(min(Lm), max(Lm))))

# Define an empty raster (grid). The extent is defined by the distance bands r and
    the
# minimum and maximum L-values.
r <- raster(nrows = 150, ncols = length(L_CSR1$r), xmn = min(L_CSR1$r), xmx = max(L
    _CSR1$r), ymn = min(Lm), ymx = max(Lm))

# Rasterize the ppp object. This creates a raster 'density' object where each cell
    (pixel) is assigned
# the number of points.
poisson1.r <- rasterize(as.SpatialPoints.ppp(Lmp), r, field = 1, fun = sum)

# Next, convert the raster object to a matrix object. This step is needed if we
    want to normalize
# the cell values at their respective band distances.
poisson1.rm <- as.matrix(poisson1.r)

# Convert column values to normalized values
N <- apply(poisson1.rm,2, function(x) {Fn = ecdf(x); Fn(x)})

# Convert matrix back to a raster
N.r <- raster(N, xmn = min(L$r), xmx = max(L$r), ymn = min(Lm), ymx = max(Lm))

# Use ggplot to plot the results
library(reshape)
library(ggplot2)
r.df <- as.data.frame(cbind(xyFromCell(N.r,1:ncell(N.r)),getValues(N.r)))
r.df <- na.exclude(r.df)
plotData <- melt(r.df, id=c("x","y"))
cols <- c('#AA0000', '#F03B20', '#FD8D3C', '#FECC5C', '#FFFFCC')
brks <- c(0.05, .1, 0.5, 0.9, 0.95)
ggplot(aes(x = x, y = y), data = plotData) +
  geom_tile(aes(fill = value)) +
  scale_fill_gradientn(colours = cols, breaks=brks) +
  theme(legend.position = "right") + xlab("Distance(m)") + ylab(expression(hat("L")
    )) +
  geom_line(aes(r,iso),data=L,size = 1.5,col="#707070")
```

E.7 Kod för envelopes på nervdatan

```
library(spatstat) # Load in spatstat

# In this version the window is fitted after the nervedata, so the window is c
    (1.31,1) instead of c(1,1)
endPointsData <- readRDS("C:/Users/caspe/Dropbox/Kurser/Kandidatarbete_myFolder/
    code/endPointsCPY_v3.rds") # Casper Laptop


Group1Size <- length(endPointsData[["group1"]]) # Size of group1, healthy subjects
Group2Size <- length(endPointsData[["group2"]]) # Size of group2, non-healthy
    subjects



# Envelope with Ripleys L function on group 1
EnvelopeLestGroup1 <- parallel::mclapply(endPointsData[["group1"]], spatstat::
    envelope, savefuns = TRUE, nsim = 99,
                                         fun = Lest, global = FALSE)
envPooledLestGroup1 <- do.call(pool, EnvelopeLestGroup1)
plot(envPooledLestGroup1, . -r ~ r, main = "Envelope on Group 1 with Ripley's L
    function")



# Envelope with Ripleys L function on group 2
EnvelopeLestGroup2 <- parallel::mclapply(endPointsData[["group2"]], spatstat::
    envelope, savefuns = TRUE, nsim = 99,
                                         fun = Lest, global = FALSE)
envPooledLestGroup2 <- do.call(pool, EnvelopeLestGroup2)
plot(envPooledLestGroup2, . -r ~ r, main = "Envelope on Group 2 with Ripley's L
    function")

# Envelope Ripleys L Both Groups
endPointsDataBoth <- c(endPointsData[["group1"]],endPointsData[["group2"]])

EnvelopeLestBothGroups <- parallel::mclapply(endPointsDataBoth, spatstat::envelope,
     savefuns = TRUE, nsim = 99,
                                             fun = Lest, global = FALSE)
envPooledLestBothGroups <- do.call(pool, EnvelopeLestBothGroups)




# Plot Lest for both group 1 and group 2 in same plot

plot(envPooledLestGroup1[["r"]],envPooledLestGroup1[["obs"]]-envPooledLestGroup1[["
    r"]],type = "l", col = "black", xlab = "r", ylab = "L(r)-r",
     main = "Ripleys L(r)-r function for Group 1 and Group 2", ylim = c(-0.01,
    0.09))
lines(envPooledLestGroup2[["r"]],envPooledLestGroup2[["obs"]]-envPooledLestGroup2[[
    "r"]], col = "blue", lty = 1)
lines(envPooledLestGroup1[["r"]],envPooledLestGroup1[["theo"]]-envPooledLestGroup1
    [["r"]], col = "red", lty = 2)
lines(envPooledLestGroup1[["r"]],envPooledLestGroup1[["obs"]]-envPooledLestGroup1[[
    "r"]], col = "black", lty = 1)
legend(x = 0.2, y = 0.085, legend = c("Group 1", "Group 2","Theoretical"), col = c(
    "black","blue","red"), lty = 1)









# Envelope with Pair-Correlation Function on group 1
```

```r
EnvelopePCFGroup1 <- parallel::mclapply(endPointsData[["group1"]], spatstat::envelope, savefuns = TRUE, nsim = 99,
                                        fun = pcf, global = FALSE)
envPooledPCFGroup1 <- do.call(pool, EnvelopePCFGroup1)
plot(envPooledPCFGroup1, xlim = c(0.005,0.25), main = "Envelope on group 1 with Pair-Correlation Function")



# Envelope with Pair-Correlation Function on group 2
EnvelopePCFGroup2 <- parallel::mclapply(endPointsData[["group2"]], spatstat::envelope, savefuns = TRUE, nsim = 99,
                                        fun = pcf, global = FALSE)
envPooledPCFGroup2 <- do.call(pool, EnvelopePCFGroup2)
plot(envPooledPCFGroup2, xlim = c(0.0025,0.25), main = "Envelope on group 2 with Pair-Correlation Function")


# Create envelopes for both groups

endPointsDataBoth <- c(endPointsData[["group1"]],endPointsData[["group2"]])

EnvelopePCFBothGroups <- parallel::mclapply(endPointsDataBoth, spatstat::envelope, savefuns = TRUE, nsim = 99,
                                            fun = pcf, global = FALSE)
envPooledPCFBothGroups <- do.call(pool, EnvelopePCFBothGroups)


plot(envPooledPCFGroup1[["r"]],envPooledPCFGroup1[["obs"]],type = "l", col = "green", xlab = "r", ylab = "Pair-correlation function",
     main = "Pair-correlation function for Group 1 and Group 2", xlim = c(0,0.25), ylim = c(0,8))
lines(envPooledPCFBothGroups[["r"]], envPooledPCFBothGroups[["hi"]] ,col = "grey80", lty = 1)
lines(envPooledPCFBothGroups[["r"]], envPooledPCFBothGroups[["lo"]] ,col = "grey80", lty = 1)
polygon(c(envPooledPCFBothGroups[["r"]], rev(envPooledPCFBothGroups[["r"]])),
        c(envPooledPCFBothGroups[["hi"]], rev(envPooledPCFBothGroups[["lo"]])), col = "grey80", border = NA)
lines(envPooledPCFGroup2[["r"]],envPooledPCFGroup2[["obs"]], col = "blue", lty = 1)
lines(envPooledPCFGroup2[["r"]], envPooledPCFGroup2[["theo"]], col = "red", lty = 1)
lines(envPooledPCFGroup1[["r"]],envPooledPCFGroup1[["obs"]], col = "black", lty = 1)

legend(x = 0.18, y = 8, legend = c("Group 1", "Group 2", "Theoretical", "Envelope"), col = c("black","blue","red", "grey80"), lty = 1)
```

E.8 Kod för violindiagram

```
# packages
install.packages("latex2exp")
library("latex2exp")

install.packages("spatstat")
library("spatstat")

install.packages("vioplot")
library("vioplot")

install.packages("ggplot2")
library("ggplot2")


# create matrixes and vectors
kappaVektor = vector("numeric", tidsSteg)
sigmaVektor = vector("numeric", tidsSteg)
muVektor = vector("numeric", tidsSteg)
antalPunkter = vector("numeric", tidsSteg)

Skattningar = matrix(0, nrow = 10, ncol = 4)

kappaMatrix = matrix(nrow = 10, ncol = tidsSteg)
muMatrix = matrix(nrow = 10, ncol = tidsSteg)
sigmaMatrix = matrix(nrow = 10, ncol = tidsSteg)


# constants
tidsSteg = 1000
kappa = 30/1.31
mu = 4.2


for (j in 1:10){

  for (i in 1:tidsSteg) {

    thomasSim = rThomas(kappa = kappa, scale = j/100, mu = mu, win = owin(c(0,1),c
    (0,1)))

    antalPunkter[i] = thomasSim[["n"]]

    # change to thomas.estK or thomas.estpcf
    resultat = thomas.estpcf(thomasSim, startpar = c(kappa = kappa ,scale = j/100))

    kappaVektor[i] = resultat$modelpar[[1]]
    sigmaVektor[i] = resultat$modelpar[[2]]
    muVektor[i] = antalPunkter[i]/kappaVektor[i]


  }
  medelSigma = mean(sigmaVektor)
  medelKappa = mean(kappaVektor)
  medelMu = mean(muVektor)

  kappaMatrix[j,] = sort(kappaVektor)
  muMatrix[j,] = sort(muVektor)
  sigmaMatrix[j,] = sort(sigmaVektor)

  Skattningar[j, 4] = medelSigma
  Skattningar[j, 3] = medelMu
  Skattningar[j, 2] = medelKappa
  Skattningar[j, 1] = j/100

  cat("\n Sigma: ", j/100)
  cat("\n Kappa_est: ", medelKappa)
  cat(" \n Mu_est: ", medelMu)
  cat("\n Sigma_est: ", medelSigma)
```

```
  cat("\n ************")

}



# make data frames
dfKappa= data.frame(name=c(rep("0.01",tidsSteg),rep("0.02",tidsSteg),rep("0.03",tidsSteg),rep("0.04",tidsSteg),rep("0.05",tidsSteg),rep("0.06",tidsSteg),rep("0.07",tidsSteg),rep("0.08",tidsSteg),rep("0.09",tidsSteg),rep("0.1",tidsSteg)),
                    est = c(kappaMatrix[1,],kappaMatrix[2,],kappaMatrix[3,],kappaMatrix[4,],kappaMatrix[5,],kappaMatrix[6,],kappaMatrix[7,],kappaMatrix[8,],kappaMatrix[9,],kappaMatrix[10,]))
dfMu = data.frame(name=c(rep("0.01",tidsSteg),rep("0.02",tidsSteg),rep("0.03",tidsSteg),rep("0.04",tidsSteg),rep("0.05",tidsSteg),rep("0.06",tidsSteg),rep("0.07",tidsSteg),rep("0.08",tidsSteg),rep("0.09",tidsSteg),rep("0.1",tidsSteg)),
                  est = c(muMatrix[1,],muMatrix[2,],muMatrix[3,],muMatrix[4,],muMatrix[5,],muMatrix[6,],muMatrix[7,],muMatrix[8,],muMatrix[9,],muMatrix[10,]))
dfSigma = data.frame(name=c(rep("0.01",tidsSteg),rep("0.02",tidsSteg),rep("0.03",tidsSteg),rep("0.04",tidsSteg),rep("0.05",tidsSteg),rep("0.06",tidsSteg),rep("0.07",tidsSteg),rep("0.08",tidsSteg),rep("0.09",tidsSteg),rep("0.1",tidsSteg)),
                     est = c(sigmaMatrix[1,],sigmaMatrix[2,],sigmaMatrix[3,],sigmaMatrix[4,],sigmaMatrix[5,],sigmaMatrix[6,],sigmaMatrix[7,],sigmaMatrix[8,],sigmaMatrix[9,],sigmaMatrix[10,]))



# violin plots
kappaPlot = ggplot(dfKappa, aes(x = name, y = log(est), color = name)) + geom_violin(trim = FALSE)
muPlot = ggplot(dfMu, aes(x = name, y = log(est), color = name)) + geom_violin(trim = FALSE)
sigmaPlot = ggplot(dfSigma, aes(x = name, y = log(est), color = name)) + geom_violin(trim = FALSE)

# print the plots
kappaPlot + ggtitle(TeX("Estimates of $\\kappa$, (PCF)")) + xlab("True sigma") + ylab("Estimation") + labs(color = "sigma")
muPlot + ggtitle(TeX("Estimates of $\\mu$, (PCF)")) + xlab("True sigma") + ylab("Estimation") + labs(color = "sigma")
sigmaPlot + ggtitle(TeX("Estimates of $\\sigma$, (PCF)")) + xlab("True sigma") + ylab("Estimation") + labs(color = "sigma")
```

E.9 Kod för skattning/figurer av nervdata

```
endPointsData = readRDS("C:/Users/lirim/Dropbox/Chalmers/Kandidatarbete MVEX01
    -20-13/Data/endpointsCPY_v3.rds")
require(spatstat)
# Skattning av parametrar f r
antalPunkter = vector("numeric", 8)

kappa = vector("numeric", 8)
mu = vector("numeric", 8)
sigma = vector("numeric", 8)

antalPunkter2 = vector("numeric", 6)
kappa2 = vector("numeric", 6)
mu2 = vector("numeric", 6)
sigma2 = vector("numeric", 6)

## Estimation using Ripley's K-function / or pcf

for (i in 1:8){

  X = endPointsData[["group1"]][[i]][["x"]]
  Y = endPointsData[["group1"]][[i]][["y"]]
  # plot(X, Y, xlab = "x-coordinate", ylab = "y-coordinate",las = 1,
  #      font.axis = 6, font.lab = 6, font.main = 6, pch = 20, main = "Group 1 -
    Subject 1")



  pointPattern = endPointsData[["group1"]][[i]]
  antalPunkter[i] = pointPattern[["n"]]

  resultat = thomas.estpcf(pointPattern)

  kappa[i] = resultat$modelpar[[1]]
  sigma[i] = resultat$modelpar[[2]]
  mu[i] = antalPunkter[i]/kappa[i]

  plot(pointPattern, pch = 20, main = paste(c("Grupp 1 - patient"),i))

}


for (i in 1:6){

  X = endPointsData[["group2"]][[i]][["x"]]
  Y = endPointsData[["group2"]][[i]][["y"]]
  # plot(X, Y, xlab = "x-coordinate", ylab = "y-coordinate",las = 1,
  #      font.axis = 6, font.lab = 6, font.main = 6, pch = 20, main = "Group 1 -
    Subject 1")



  pointPattern2 = endPointsData[["group2"]][[i]]
  antalPunkter2[i] = pointPattern2[["n"]]

  resultat = thomas.estpcf(pointPattern)

  kappa2[i] = resultat$modelpar[[1]]
  sigma2[i] = resultat$modelpar[[2]]
  mu2[i] = antalPunkter2[i]/kappa2[i]

  plot(pointPattern, pch = 20, main = paste(c("Grupp 2 - patient"),i))

}
```