SISMID 2025 Spatial Statistics Waller Point Process 3: K functions Archaeology Dental Example

Lance A. Waller

7/15/2025

Point Processes: K function example

This code K function figures for early medieval gravesites.

Grave dental data from Richard Wright in Australia Emeritus Professor, School of Archaeology, University of Sydney NSW 2006, Australia

This code regenerates K function figures from Waller and Gotway (2004, Applied Spatial Statistics for Public Health Data)

** Note, this takes a few minutes to run. . . in several places I have the number of simulations (nsim) set to 500. . . if you reduce this number, it will run faster but your estimates will be noisier.**

First, set the library (splancs), set the working directory (set to your own), reading in the data, and read in the polygon boundary file. (see the Intensities Rmd to see how to create the polygon boundary from splancs).

```
######################
# Open libraries
######################
library(here)
```

```
## here() starts at /Users/lwaller/Library/CloudStorage/OneDrive-Emory/meetings/SISMID.2025/lectures/Le
```

```
library(splancs)
```

```
## Loading required package: sp
```

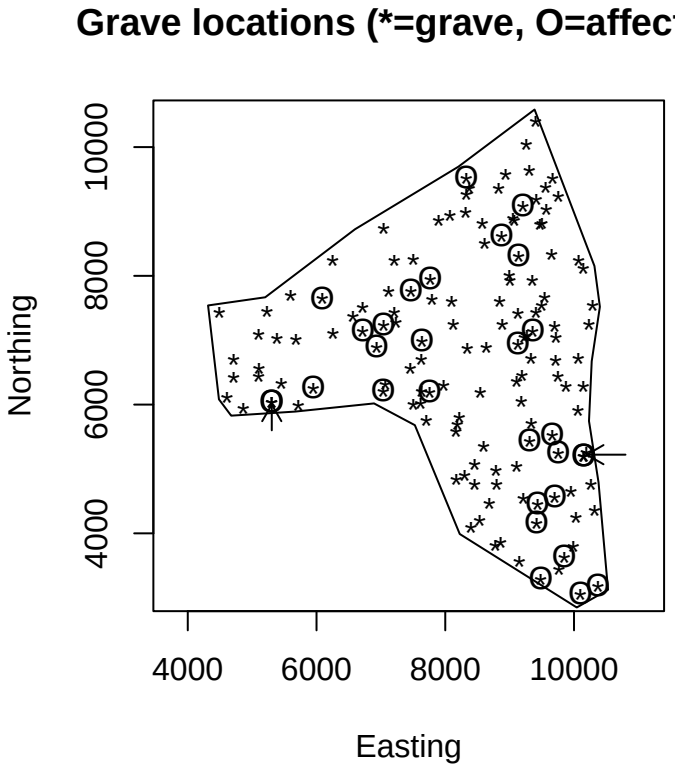
```
##
## Spatial Point Pattern Analysis Code in S-Plus
##
##  Version 2 - Spatial and Space-Time analysis
```

```
######################
# Set path to data sets, etc.
######################

# Set my working directory (Lance's here for example)
#setwd("~/OneDrive - Emory University/meetings/SISMID.2021/SISMID.2021.Waller.Rcode")
#***HERE***

path = "/Users/lwaller/Library/CloudStorage/OneDrive-Emory/meetings/SISMID.2024/SISMID_2024_spatial_sta
```

```
# Read in data with south region removed
#dental <- scan(here("data", "dental.reduced.dat"),list(lab=0,aff=0,x=0,y=0))
dental <- scan(paste(path,"data/dental.reduced.dat",sep=""),list(lab=0,aff=0,x=0,y=0))

# read in previously defined polygon boundary
#mypoly <- read.table(here("data", "mypoly.dat"),col.names = c("x","y"))
mypoly <- read.table(paste(path,"data/mypoly.dat",sep=""),col.names = c("x","y"))
```

Now adjust the data, make a plot of the locations (adjusted to be in a square) and add arrows to indicate the two pairs of affected graves (two graves very close together, both of which contain remains with reduced or missing wisdom teeth).

```
####################################
# K function analysis comparing to CSR within a polygon.
# Using splancs functions
####################################

dental.p <- as.points(dental$x,dental$y)
dentcas.p <- as.points(dental$x[dental$aff==1],dental$y[dental$aff==1])
dentcon.p <- as.points(dental$x[dental$aff==0],dental$y[dental$aff==0])

#####
# Plot points to see data set
#####

par(pty="s")

# dental$y has a bigger range so this tries to add half of the
# extra to each side

extra <- ( diff(range(dental$y))-diff(range(dental$x)) )/2

plot(dental.p,pch="*",xlim=c(min(dental$x)-extra,max(dental$x)+extra),
     ylim=range(dental$y),
     xlab="Easting",ylab="Northing")
title("Grave locations (*=grave, O=affected)")

points(dentcas.p,pch="O")

## Interactively define polygon
#mypoly <- getpoly()
##save this polygon
#write.table(mypoly,paste(path,"mypoly.dat",sep=""))

polygon(mypoly)
# Add arrows to double locations
arrows(5305,6000,5305,5600,code=1,length=0.1)
arrows(10200,5222,10800,5222,code=1,length=0.1)
```



Next, we'll set things up to compare the gravesite locations to complete spatial randomness (CSR) within a bounding rectangle... not the right thing to do since the graves occur only in the irregular polygon, but a point of comparison for proper analysis.

To do this, + We calculate the K (L) function from the data for a fixed set of distances.

- We generate complete spatial randomness (here, in the rectangle) for each simulation, calculate the K (L) function *for the same distances*.
- We find the quantiles for K(h) (L(h)) *at each distance*.
- We plot the K (L) function for the data, add lines corresponding to the 95th and 5th percentile (the "envelopes"). These represent reasonable variability around the null value of L(h) = 0 for all distances h.
- When the L function is outside the high quantile we have evidence of clustering, when L is below the low quantile, we have evidence of regularity (for those distances).

```
###################################
# CSR, edge-corrected within bounding rectangle
###################################

# define bounding rectangle
dent.poly <- cbind(c(min(dental.p[,1]),max(dental.p[,1]),max(dental.p[,1]),min(dental.p[,1])),
                   c(min(dental.p[,2]),min(dental.p[,2]),max(dental.p[,2]),max(dental.p[,2])))

dists <- 1:300
dists <- dists*20

nsim <- 500
```

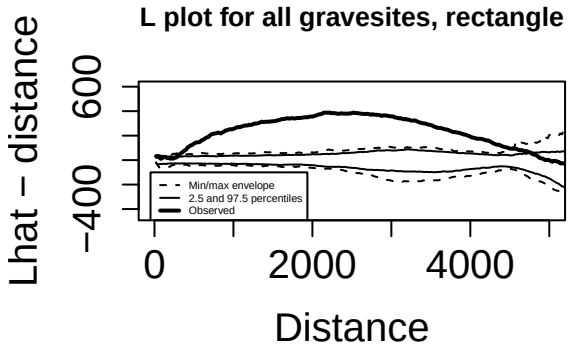
```
Kmat <- matrix(0,nsim,length(dists))
Kmat.cas <- matrix(0,nsim,length(dists))
Kmat.con <- matrix(0,nsim,length(dists))

Khat.dent <- khat(dental.p,dent.poly,dists)
Lhat.dent <- sqrt(Khat.dent/pi)
Khat.cas <- khat(dentcas.p,dent.poly,dists)
Lhat.cas <- sqrt(Khat.cas/pi)
Khat.con <- khat(dentcon.p,dent.poly,dists)
Lhat.con <- sqrt(Khat.con/pi)

maxLplot <- max(Lhat.dent - dists)
minLplot <- min(Lhat.dent - dists)
maxLplot.cas <- max(Lhat.cas - dists)
minLplot.cas <- min(Lhat.cas - dists)
maxLplot.con <- max(Lhat.con - dists)
minLplot.con <- min(Lhat.con - dists)

for (sim in 1:nsim) {
  simlocs <- csr(as.points(mypoly),length(dental.p[,1]))
  Kmat[sim,] <- khat(simlocs,as.points(mypoly),dists)
  simlocs.cas <- csr(as.points(mypoly),length(dentcas.p[,1]))
  Kmat.cas[sim,] <- khat(simlocs.cas,as.points(mypoly),dists)
  simlocs.con <- csr(as.points(mypoly),length(dentcon.p[,1]))
  Kmat.con[sim,] <- khat(simlocs.con,as.points(mypoly),dists)
  }
Kenv.up <- apply(Kmat,2,max)
Kenv.lo <- apply(Kmat,2,min)
Kenv.cas.up <- apply(Kmat.cas,2,max)
Kenv.cas.lo <- apply(Kmat.cas,2,min)
Kenv.con.up <- apply(Kmat.con,2,max)
Kenv.con.lo <- apply(Kmat.con,2,min)

Kenv.975 <- rep(0,length(dists))
Kenv.025 <- rep(0,length(dists))
Kenv.cas.975 <- rep(0,length(dists))
Kenv.cas.025 <- rep(0,length(dists))
Kenv.con.975 <- rep(0,length(dists))
Kenv.con.025 <- rep(0,length(dists))
for (i in 1:length(dists)){
   Kenv.975[i] <- quantile(Kmat[,i],prob=0.975)
   Kenv.025[i] <- quantile(Kmat[,i],prob=0.025)
   Kenv.cas.975[i] <- quantile(Kmat.cas[,i],prob=0.975)
   Kenv.cas.025[i] <- quantile(Kmat.cas[,i],prob=0.025)
   Kenv.con.975[i] <- quantile(Kmat.con[,i],prob=0.975)
   Kenv.con.025[i] <- quantile(Kmat.con[,i],prob=0.025)
}

Lenv.dentup <- sqrt(Kenv.up/pi)
Lenv.dentlo <- sqrt(Kenv.lo/pi)
Lenv.dent975 <- sqrt(Kenv.975/pi)
Lenv.dent025 <- sqrt(Kenv.025/pi)
Lenv.cas.up <- sqrt(Kenv.cas.up/pi)
```

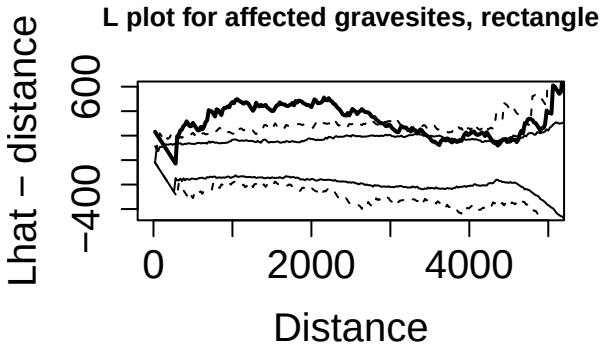
```
Lenv.cas.lo <- sqrt(Kenv.cas.lo/pi)
Lenv.cas.975 <- sqrt(Kenv.cas.975/pi)
Lenv.cas.025 <- sqrt(Kenv.cas.025/pi)
Lenv.con.up <- sqrt(Kenv.con.up/pi)
Lenv.con.lo <- sqrt(Kenv.con.lo/pi)
Lenv.con.975 <- sqrt(Kenv.con.975/pi)
Lenv.con.025 <- sqrt(Kenv.con.025/pi)
```

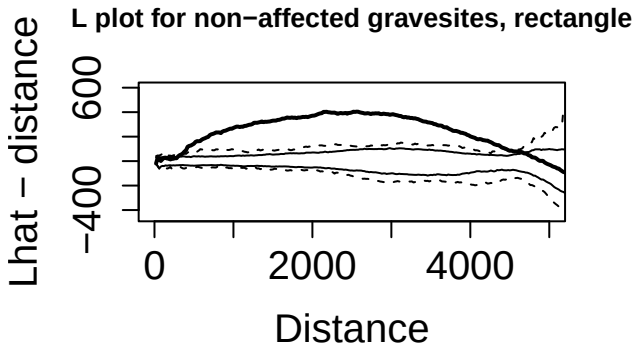
Now plot the L functions (and envelopes) for all gravesites, the affected gravesites, and the non-affected gravesites.

```
#postscript(paste(path,"dentalK.ps",sep=""),horizontal=FALSE,paper="letter")
par(mfrow=c(2,2),pty="m")
plot(dists,Lhat.dent-dists,xlab="Distance",ylab="Lhat - distance",type="l",
     ylim=c(#min(c(minLplot,(Lenv.dentlo - dists))),
#            max(c(maxLplot,(Lenv.dentup - dists)))
            -450,600),
     xlim=c(0,5000),cex.axis=1.5,cex.lab=1.5 )
lines(dists,Lhat.dent-dists,lwd=2)
lines(dists,Lenv.dentup - dists,lty=2)
lines(dists,Lenv.dentlo - dists,lty=2)
lines(dists,Lenv.dent975 - dists,lty=1)
lines(dists,Lenv.dent025 - dists,lty=1)
title("L plot for all gravesites, rectangle",cex.main=1.0)

legend(-50,-100,legend=c("Min/max envelope","2.5 and 97.5 percentiles",
       "Observed"),
       lty=c(2,1,1),lwd=c(1,1,2),cex=0.4)

plot(dists,Lhat.cas-dists,xlab="Distance",ylab="Lhat - distance",type="l",
# use same limits for all plots...
     ylim=c(#min(c(minLplot,(Lenv.dentlo - dists))),
#            max(c(maxLplot,(Lenv.dentup - dists)))
            -450,600),
     xlim=c(0,5000),cex.axis=1.5,cex.lab=1.5 )
lines(dists,Lhat.cas-dists,lwd=2)
lines(dists,Lenv.cas.up - dists,lty=2)
lines(dists,Lenv.cas.lo - dists,lty=2)
lines(dists,Lenv.cas.975 - dists,lty=1)
lines(dists,Lenv.cas.025 - dists,lty=1)
title("L plot for affected gravesites, rectangle",cex.main=1.0)

#legend(0,600,legend=c("Min/max envelope","2.5 and 97.5 percentiles",
#       "Observed"),
#       lty=c(2,1,1),lwd=c(1,1,2))

plot(dists,Lhat.con-dists,xlab="Distance",ylab="Lhat - distance",type="l",
# use same limits for all plots...
     ylim=c(#min(c(minLplot,(Lenv.dentlo - dists))),
            #max(c(maxLplot,(Lenv.dentup[dists<=5000] - dists[dists<=5000])))
            -450,600 ),
```

```
     xlim=c(0,5000),cex.axis=1.5,cex.lab=1.5 )
lines(dists,Lhat.con-dists,lwd=2)
lines(dists,Lenv.con.up - dists,lty=2)
lines(dists,Lenv.con.lo - dists,lty=2)
lines(dists,Lenv.con.975 - dists,lty=1)
lines(dists,Lenv.con.025 - dists,lty=1)
title("L plot for non-affected gravesites, rectangle",cex.main=1.0)

#legend(0,600,legend=c("Min/max envelope","2.5 and 97.5 percentiles",
#       "Observed"),
#       lty=c(2,1,1),lwd=c(1,1,2))
#dev.off()
```







These look like clustering at almost all distances, but these are checking for clustering within a rectangle around the entire area, but the gravesites are in the "7" shaped area in the polygon.

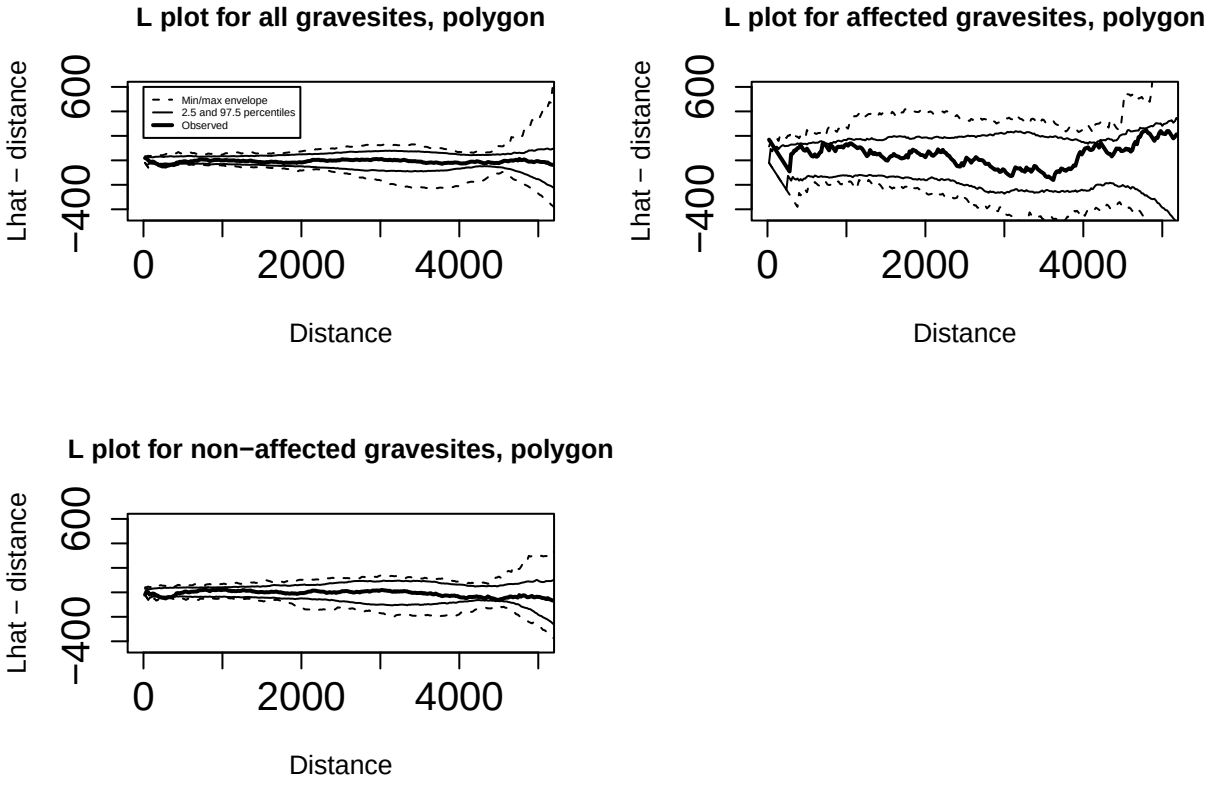
Let's add the polygon boundary and run with Ripley's edge correction to see if there is clustering *within the polygon*, rather than clustering *within the rectangle*.

```
###################################
# CSR, edge-corrected within polygon
###################################

dists <- 1:300
dists <- dists*20

nsim <- 500
```

```
Kmat <- matrix(0,nsim,length(dists))
Kmat.cas <- matrix(0,nsim,length(dists))
Kmat.con <- matrix(0,nsim,length(dists))

# Need mypoly to be a matrix)
mypoly.mat = cbind(mypoly$x,mypoly$y)

Khat.dent <- khat(dental.p,mypoly.mat,dists)
Lhat.dent <- sqrt(Khat.dent/pi)
Khat.cas <- khat(dentcas.p,mypoly.mat,dists)
Lhat.cas <- sqrt(Khat.cas/pi)
Khat.con <- khat(dentcon.p,mypoly.mat,dists)
Lhat.con <- sqrt(Khat.con/pi)

maxLplot <- max(Lhat.dent - dists)
minLplot <- min(Lhat.dent - dists)
maxLplot.cas <- max(Lhat.cas - dists)
minLplot.cas <- min(Lhat.cas - dists)
maxLplot.con <- max(Lhat.con - dists)
minLplot.con <- min(Lhat.con - dists)

for (sim in 1:nsim) {
  simlocs <- csr(as.points(mypoly),length(dental.p[,1]))
  Kmat[sim,] <- khat(simlocs,as.points(mypoly),dists)
  simlocs.cas <- csr(as.points(mypoly),length(dentcas.p[,1]))
  Kmat.cas[sim,] <- khat(simlocs.cas,as.points(mypoly),dists)
  simlocs.con <- csr(as.points(mypoly),length(dentcon.p[,1]))
  Kmat.con[sim,] <- khat(simlocs.con,as.points(mypoly),dists)
  }
Kenv.up <- apply(Kmat,2,max)
Kenv.lo <- apply(Kmat,2,min)
Kenv.cas.up <- apply(Kmat.cas,2,max)
Kenv.cas.lo <- apply(Kmat.cas,2,min)
Kenv.con.up <- apply(Kmat.con,2,max)
Kenv.con.lo <- apply(Kmat.con,2,min)

Kenv.975 <- rep(0,length(dists))
Kenv.025 <- rep(0,length(dists))
Kenv.cas.975 <- rep(0,length(dists))
Kenv.cas.025 <- rep(0,length(dists))
Kenv.con.975 <- rep(0,length(dists))
Kenv.con.025 <- rep(0,length(dists))
for (i in 1:length(dists)){
   Kenv.975[i] <- quantile(Kmat[,i],prob=0.975)
   Kenv.025[i] <- quantile(Kmat[,i],prob=0.025)
   Kenv.cas.975[i] <- quantile(Kmat.cas[,i],prob=0.975)
   Kenv.cas.025[i] <- quantile(Kmat.cas[,i],prob=0.025)
   Kenv.con.975[i] <- quantile(Kmat.con[,i],prob=0.975)
   Kenv.con.025[i] <- quantile(Kmat.con[,i],prob=0.025)
}

Lenv.dentup <- sqrt(Kenv.up/pi)
Lenv.dentlo <- sqrt(Kenv.lo/pi)
```

```
Lenv.dent975 <- sqrt(Kenv.975/pi)
Lenv.dent025 <- sqrt(Kenv.025/pi)
Lenv.cas.up <- sqrt(Kenv.cas.up/pi)
Lenv.cas.lo <- sqrt(Kenv.cas.lo/pi)
Lenv.cas.975 <- sqrt(Kenv.cas.975/pi)
Lenv.cas.025 <- sqrt(Kenv.cas.025/pi)
Lenv.con.up <- sqrt(Kenv.con.up/pi)
Lenv.con.lo <- sqrt(Kenv.con.lo/pi)
Lenv.con.975 <- sqrt(Kenv.con.975/pi)
Lenv.con.025 <- sqrt(Kenv.con.025/pi)

#postscript(paste(path,"dentalK.poly.ps",sep=""),horizontal=F,paper="letter")
par(mfrow=c(2,2),pty="m")
plot(dists,Lhat.dent-dists,xlab="Distance",ylab="Lhat - distance",type="l",
     ylim=c(#min(c(minLplot,(Lenv.dentlo - dists))),
#            max(c(maxLplot,(Lenv.dentup - dists)))
            -450,600),
     xlim=c(0,5000),cex.axis=1.5,cex.lab=1.0 )
lines(dists,Lhat.dent-dists,lwd=2)
lines(dists,Lenv.dentup - dists,lty=2)
lines(dists,Lenv.dentlo - dists,lty=2)
lines(dists,Lenv.dent975 - dists,lty=1)
lines(dists,Lenv.dent025 - dists,lty=1)
title("L plot for all gravesites, polygon",cex.main=1.0)

legend(0,600,legend=c("Min/max envelope","2.5 and 97.5 percentiles",
       "Observed"),
       lty=c(2,1,1),lwd=c(1,1,2),cex=0.4)

plot(dists,Lhat.cas-dists,xlab="Distance",ylab="Lhat - distance",type="l",
# use same limits for all plots...
     ylim=c(#min(c(minLplot,(Lenv.dentlo - dists))),
#            max(c(maxLplot,(Lenv.dentup - dists)))
            -450,600),
     xlim=c(0,5000),cex.axis=1.5,cex.lab=1.0 )
lines(dists,Lhat.cas-dists,lwd=2)
lines(dists,Lenv.cas.up - dists,lty=2)
lines(dists,Lenv.cas.lo - dists,lty=2)
lines(dists,Lenv.cas.975 - dists,lty=1)
lines(dists,Lenv.cas.025 - dists,lty=1)
title("L plot for affected gravesites, polygon",cex.main=1.0)

#legend(0,600,legend=c("Min/max envelope","2.5 and 97.5 percentiles",
#       "Observed"),
#       lty=c(2,1,1),lwd=c(1,1,2))

plot(dists,Lhat.con-dists,xlab="Distance",ylab="Lhat - distance",type="l",
# use same limits for all plots...
     ylim=c(#min(c(minLplot,(Lenv.dentlo - dists))),
            #max(c(maxLplot,(Lenv.dentup[dists<=5000] - dists[dists<=5000])))
            -450,600 ),
     xlim=c(0,5000),cex.axis=1.5,cex.lab=1.0 )
lines(dists,Lhat.con-dists,lwd=2)
```

```
lines(dists,Lenv.con.up - dists,lty=2)
lines(dists,Lenv.con.lo - dists,lty=2)
lines(dists,Lenv.con.975 - dists,lty=1)
lines(dists,Lenv.con.025 - dists,lty=1)
title("L plot for non-affected gravesites, polygon",cex.main=1.0)

#legend(0,600,legend=c("Min/max envelope","2.5 and 97.5 percentiles",
#       "Observed"),
#       lty=c(2,1,1),lwd=c(1,1,2))
#dev.off()
```



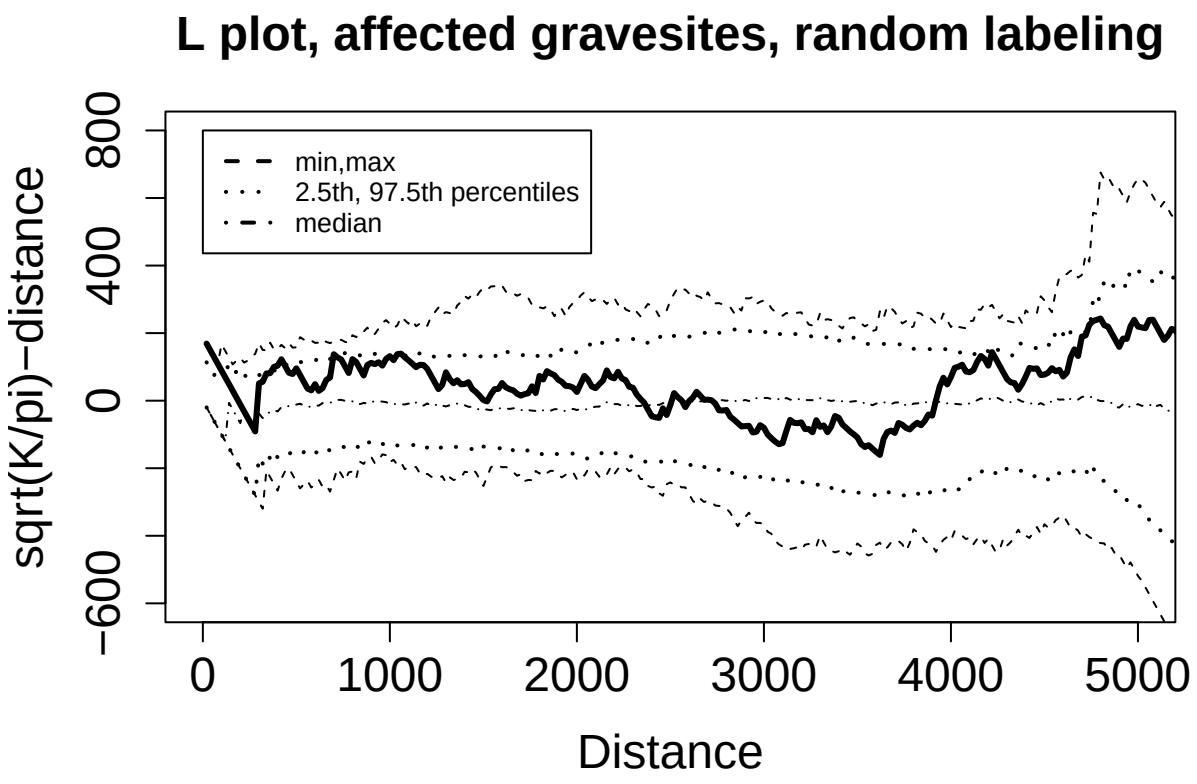
Now for bonus code (not covered in the presentation).

In addition to simulating CSR as the null hypothesis, we can also consider a "random labeling" null hypothesis where we keep the same locations and simply select 30 of them to be cases each time (we randomize the affected/non-affected labels rather than randomizing locations). The two null hypotheses are close but not identical. (see discussion in Waller and Gotway 2004).

```
###################
# Code past this point for two additional examples from Waller and Gotway (2004)
#  K function inference under the random labeling null hypothesis
#  Monte Carlo inference for the difference between K functions for cases and controls.
#  ***NOTE***:  Code not fully up-to-date, generates some warnings but runs. (July 2020)
###################
# K function analysis under "random labeling hypothesis".
# Conditional on all locations, randomly assign 31 as cases, then
# find K function for the point process defined by those 31 points.
```

```
###################
#postscript("dentalKrl.ps")

#par(mfrow=c(1,1))

#plot(dists,Lhat.cas-dists,xlab="Distance",ylab="Lhat - distance",type="l",
#     ylim=c(min(c(minLplot,(Lenv.cas.lo - dists))),
#            max(c(maxLplot,(Lenv.cas.up - dists))) ) )
#lines(dists,Lhat.cas-dists,lwd=2)
#lines(dists,Lenv.cas.up - dists,lty=2)
#lines(dists,Lenv.cas.lo - dists,lty=2)

ind <- 1:length(dental$x)

sim <- 499 # number of simulations

Lmat <- matrix(0,sim,length(dists)) # matrix to store L-hat values
quantmat <- matrix(0,5,length(dists)) # matrix to store quantiles
teststat <- rep(0,sim) # vector to store test statistic values
hmax <- 2000  # maximum distance for Monte Carlo test

for (i in 1:sim) {
  randind <- sample(ind, length(dental$x[dental$aff==1]))

  dentrandx <- dental$x[randind]
  dentrandy <- dental$y[randind]
  dentrand.p <- as.points(dentrandx,dentrandy)

  Khat.rand <- khat(dentrand.p,as.points(mypoly),dists)
  Lhat.rand <- sqrt(Khat.rand/pi)
  Lmat[i,] <- Lhat.rand-dists

#  lines(dists,Lhat.rand-dists,col="grey70",lwd=1)

  teststat[i] <- max( abs(Lhat.rand[dists<hmax]-dists[dists<hmax]) )
}
#lines(dists,Lhat.cas-dists,col="black",lwd=2) #col="red")
teststatobs <- max( abs(Lhat.cas[dists<hmax]-dists[dists<hmax]) )

#title("Gravesite data, K functions under random labeling")

####################################
# Calc quantiles

for (j in 1:length(dists)) {
  quantmat[,j] <- quantile(Lmat[,j],
      c(0.0,0.025,0.5,0.975,1.0))
}

####################################
# Make plot

#postscript(paste(path,"dentalK.rl.ps",sep=""),horizontal=T,paper="letter")
```

```r
par(mfrow=c(1,1))

plot(dists,Lhat.cas-dists,xlab="Distance",ylab="sqrt(K/pi)-distance",
    xlim=c(0,5000),ylim=c(-600,800),type="l",cex.axis=1.5,cex.lab=1.5)
lines(dists,Lhat.cas-dists,col="black",lwd=3)

# min/max envelope
lines(dists,quantmat[1,],lty=2)#,col="blue")
lines(dists,quantmat[5,],lty=2)#,col="blue")

# 95% envelope
lines(dists,quantmat[2,],lty=3,lwd=2)#,col="orange")
lines(dists,quantmat[4,],lty=3,lwd=2)#,col="orange")

# median
lines(dists,quantmat[3,],lty=4)#,col="yellow")

legend(0,800,legend=c("min,max",
              "2.5th, 97.5th percentiles",
              "median"),
              col=1,
                  lty=c(2,3,4),lwd=c(2,2,2),cex=0.8)

title("L plot, affected gravesites, random labeling",cex.main=1.5)
```




```r
#dev.off()
```

To compare the K functions from cases and control, you might be interested in the *difference in K functions*

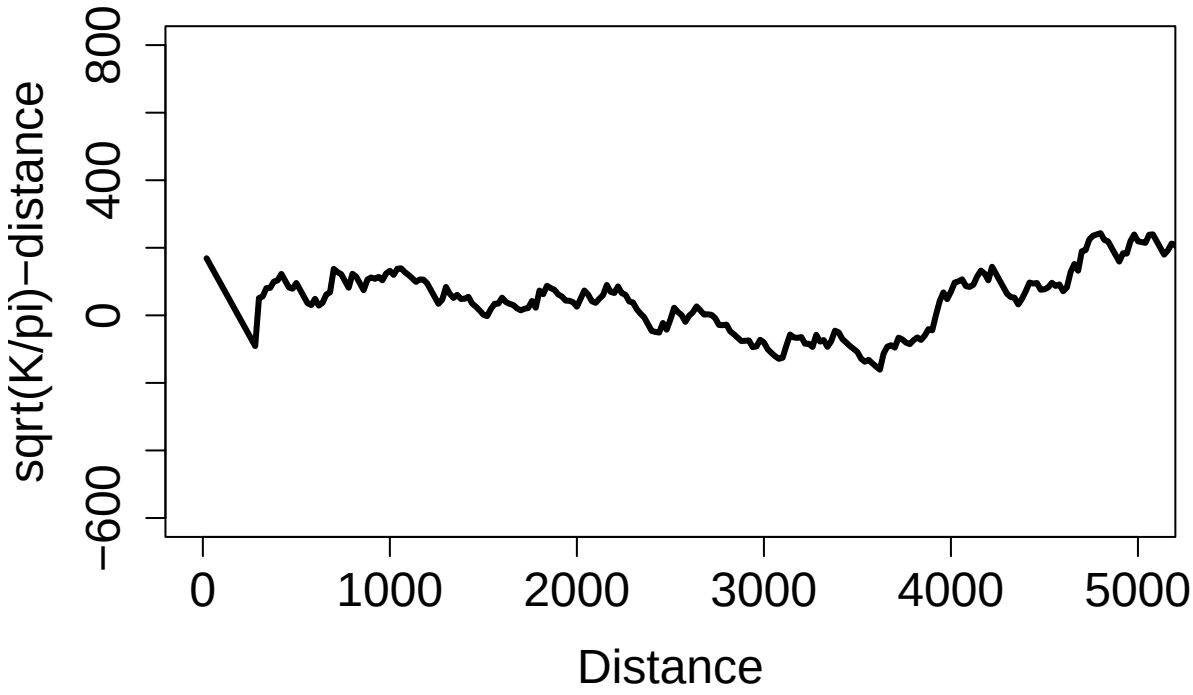
where

$K_{\text{diff}}(h) = K_1(h) - K_2(h)$.

Then you can think about whether process 1 (cases) is more clustered or more regular than process 2 (controls). (What would that look like?)

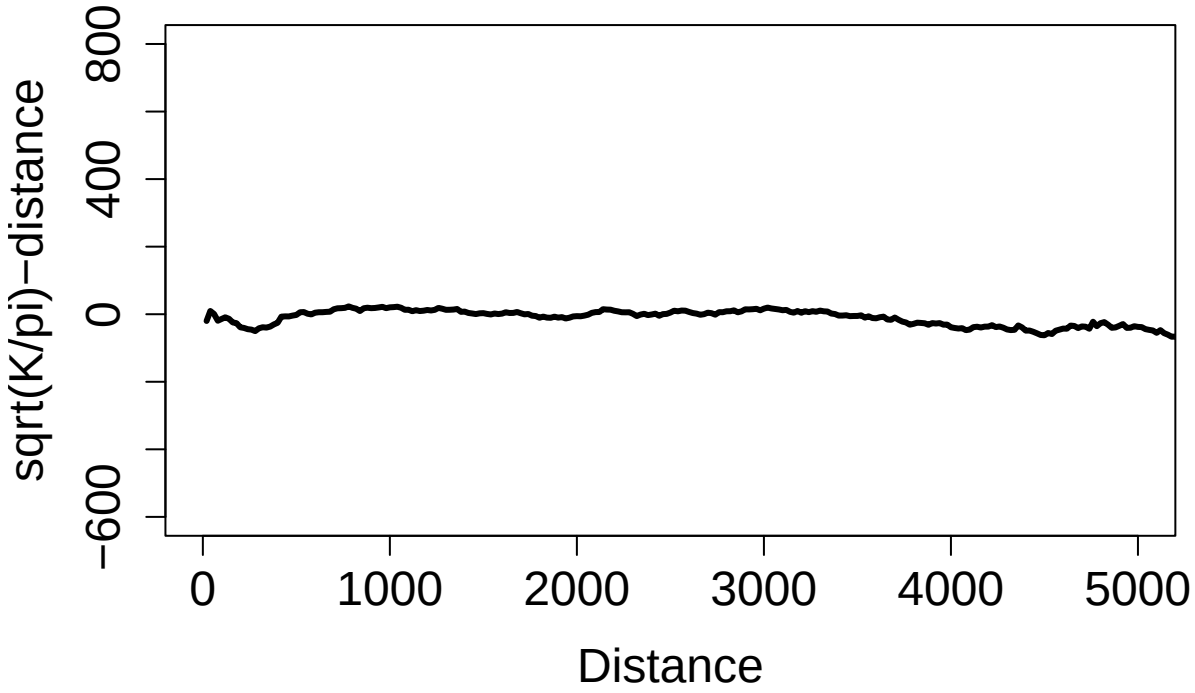
Note, I'm using 500 simulations (nsim=500)... if you make this smaller, it will run faster.

```
##############################################
# Difference in K functions
##############################################

dists <- 1:300
dists <- dists*20

nsim <- 500
Kmat <- matrix(0,nsim,length(dists))
Kmat.cas <- matrix(0,nsim,length(dists))
Kmat.con <- matrix(0,nsim,length(dists))

Khat.cas <- khat(dentcas.p,as.points(mypoly),dists)
Lhat.cas <- sqrt(Khat.cas/pi)
Khat.con <- khat(dentcon.p,as.points(mypoly),dists)
Lhat.con <- sqrt(Khat.con/pi)

par(mfrow=c(1,1),pty="m")
plot(dists,Lhat.cas-dists,xlab="Distance",ylab="sqrt(K/pi)-distance",
    xlim=c(0,5000),ylim=c(-600,800),type="l",cex.axis=1.5,cex.lab=1.5)
lines(dists,Lhat.cas-dists,col="black",lwd=3)
```

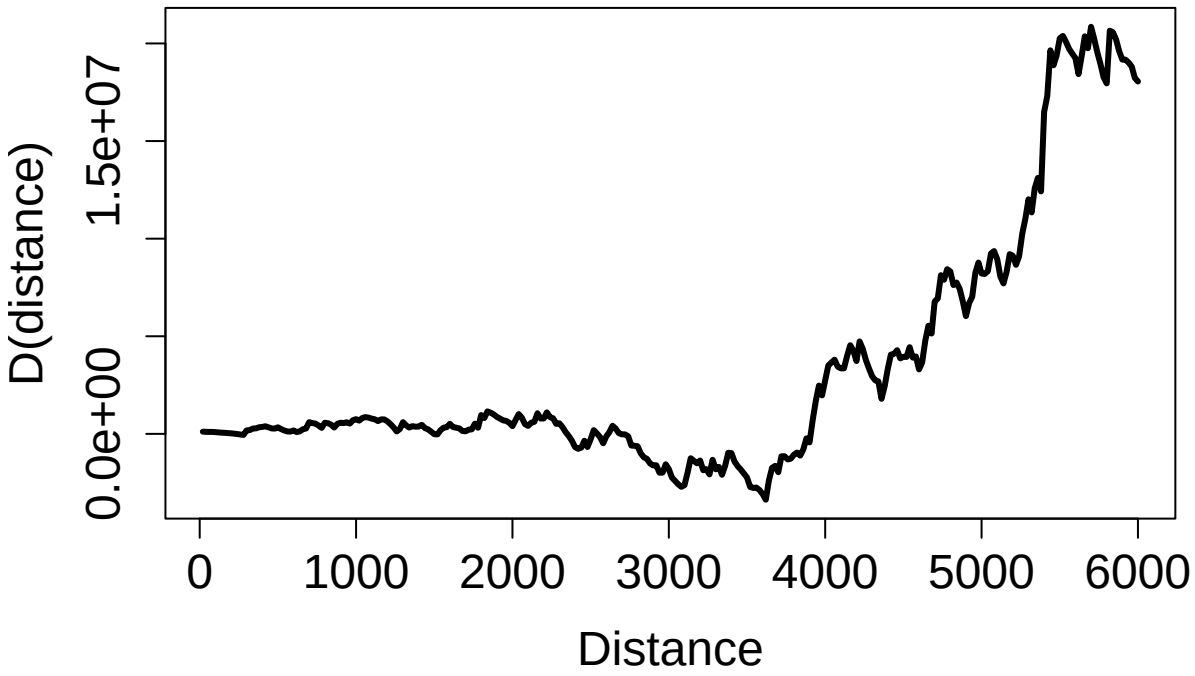



```
plot(dists,Lhat.con-dists,xlab="Distance",ylab="sqrt(K/pi)-distance",
    xlim=c(0,5000),ylim=c(-600,800),type="l",cex.axis=1.5,cex.lab=1.5)
```

```
lines(dists,Lhat.con-dists,col="black",lwd=3)
```

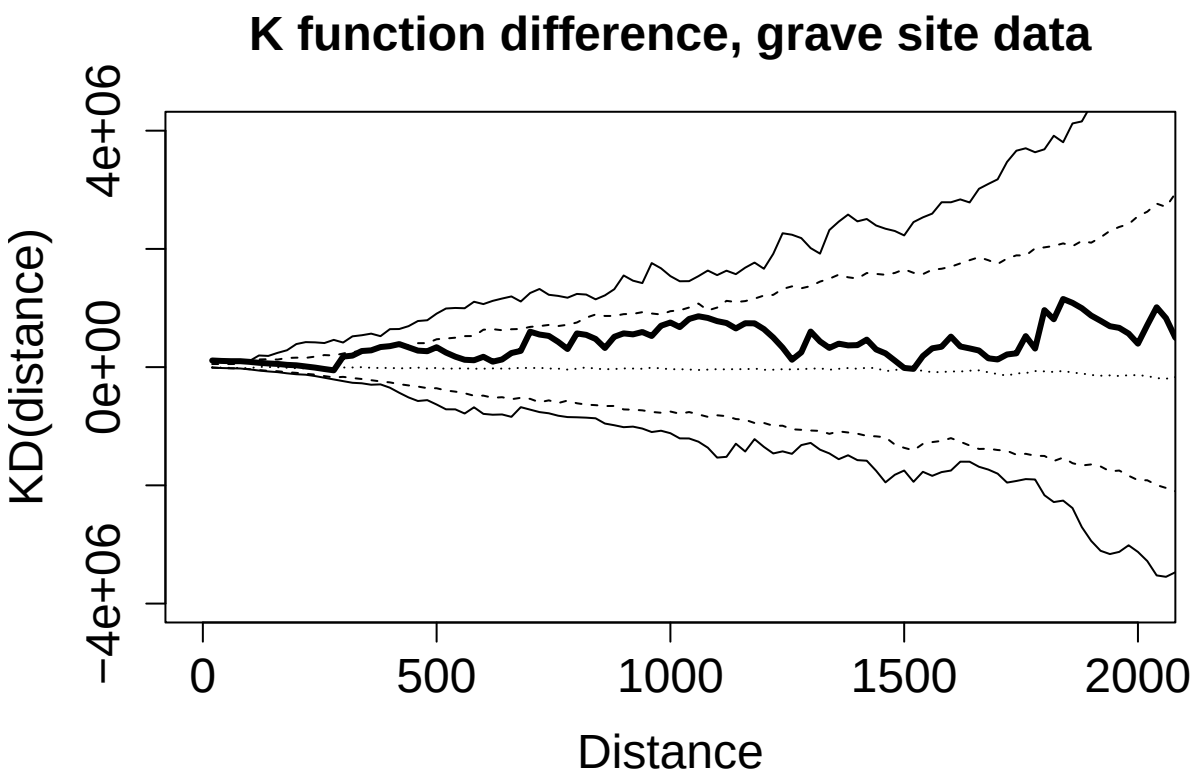



```
D <- Khat.cas-Khat.con

par(mfrow=c(1,1),pty="m")
plot(dists,D,xlab="Distance",ylab="D(distance)",
     type="l",cex.axis=1.5,cex.lab=1.5)
lines(dists,D,col="black",lwd=3)
```




```
sim <- 499 # number of simulations
```

```
Dmat <- matrix(0,sim,length(dists)) # matrix to store difference values
D.quantmat <- matrix(0,5,length(dists)) # matrix to store quantiles

for (i in 1:sim) {
  randind <- sample(ind, length(dental$x[dental$aff==1]))

  dent.cas.randx <- dental$x[randind]
  dent.cas.randy <- dental$y[randind]
  dent.con.randx <- dental$x[-randind]
  dent.con.randy <- dental$y[-randind]
  dent.cas.rand.p <- as.points(dent.cas.randx,dent.cas.randy)
  dent.con.rand.p <- as.points(dent.con.randx,dent.con.randy)

  Khat.cas.rand <- khat(dent.cas.rand.p,as.points(mypoly),dists)
  Khat.con.rand <- khat(dent.con.rand.p,as.points(mypoly),dists)
  Dmat[i,] <- Khat.cas.rand-Khat.con.rand
}

####################################
# Calc quantiles

for (j in 1:length(dists)) {
  D.quantmat[,j] <- quantile(Dmat[,j],
      c(0.0,0.025,0.5,0.975,1.0))
}

# add median, and 95 envelope

#postscript(paste(path,"dental.diffK.ps",sep=""),horizontal=T,paper="letter")
par(mfrow=c(1,1),pty="m")
plot(dists,D,xlab="Distance",ylab="KD(distance)",
     xlim=c(0,2000),ylim=c(-4000000,4000000),
     type="l",cex.axis=1.5,cex.lab=1.5)
lines(dists,D,col="black",lwd=3)
title("K function difference, grave site data",cex.main=1.5)

lines(dists,D.quantmat[1,])
lines(dists,D.quantmat[5,])

lines(dists,D.quantmat[2,],lty=2)
lines(dists,D.quantmat[4,],lty=2)

lines(dists,D.quantmat[3,],lty=3)
```


K function difference, grave site data
KD(distance)
4e+06
0e+00
−4e+06
0
500
1000
1500
2000
Distance

```
#dev.off()
```