1. *On the last homework there was some confusion about two problems. I took o some points for one of those but am now giving you a chance to get them back.*

   (a) *It was pointed out in class that, conditional on n events, event locations are uniformly distributed for a homogeneous Poisson process. Show this result for a 1 d process. Hint: Consider a one-dimensional process on a transect of length L, (0,L]. Given that one event has occurred on the interval (0,L] what is the probability that it occurred in the subinterval (0, s] for s <L? Show that this is the cdf of a Unif (0,L) distribution.*

   (b) *Suppose we have a realization of a spatial point process consisting of N event locations s1, s2, , sN . Let Hi denote the distance between the ith event and the nearest neigh- boring event. The cumulative distribution function of H (the nearest event-event dis- tance) is the G function. (This problem will be continued on the next homework assignment). Derive the G function if the point process is CSR; i.e. what is G(h) = P (H ≤ h).*

2. *We looked at one simple method of using nearest neighbor distances to assess a null hypothesis of CSR. The method was based on using Monte Carlo tests to evaluate the deviation of the mean distance from that expected under CSR. We will look at another possible approach in this problem, one that theoretically would allow us to use a test based on normal theory. A question on Homework 2 asked you to find the probability density function of H, the distance between an event and the nearest neighboring event.*

   We will be working with a homogeneous Poisson process with intensity $\lambda = 30$.

   (a) *What are the mean and variance of $\bar{H} = \frac{1}{30}\Sigma H_i$ when $\lambda = 30$, ie.e both the sample size and the intensity equal 30.*

   $\beta = 2$; $\theta = (\lambda\pi)^{\frac{-1}{2}}$; $\lambda = n = 30$

   $E[\bar{H}] = \frac{1}{n}E[\Sigma\bar{H}] = E[H] = \theta * \Gamma[1 + \frac{1}{\beta}]$

   $\rightarrow E[\bar{H}] = (30\pi)^{-\frac{1}{2}}\Gamma(\frac{3}{2}) = (30\pi)^{-\frac{1}{2}}\frac{1}{2}\Gamma(\frac{1}{2}) = (30^2\pi)^{-1}\frac{1}{2}(\pi)^{\frac{1}{2}} = \frac{1}{2*(30)^2} \approx 0.0913$

   $Var[\bar{H}] = \frac{1}{n}Var[H] = \frac{\theta^2[\Gamma(1+\frac{2}{\beta})-(\Gamma(1+\frac{1}{\beta}))^2]}{n} = 30 * [\Gamma(2) - (\frac{1}{2}\Gamma(\frac{1}{2})^2] = \frac{1}{30^2\pi}[1 - \frac{1}{4}\pi] \approx 10^{-4}$

   (b) *What is the approximate sampling distribution of $Z = \frac{\bar{H}-E[\bar{H}]}{(Var[\bar{H}])^{\frac{1}{2}}}$ under CSR and how do you know this?*

   Note that $\beta = 2 \geq 1 \rightarrow$ the mgf exists. By the CLT, $\frac{\bar{H}-E[\bar{H}]}{(Var[\bar{H}])^{\frac{1}{2}}} \rightarrow N(0, 1)$.

   (c) *Simulate 1000 realizations of complete spatial randomness in the unit square with 30 events in each realization. For each realization*

      i. *Calculate the distance between each event and its nearest-neighboring event (Hi for the ith event in the realization)*

```
csr_sim30 <- array(0,c(30,2,1000))
csr_nndist <- data.frame(matrix(0,nrow = 30, ncol=1000))
set.seed(123)
for(i in 1:1000){
  re <- runifpoint(30, win=owin(c(0,1),c(0,1)))
  csr_sim30[,,i] <- c(re$x,re$y)
}
```

```
csr_nndist <- apply(csr_sim30,3,nndist)
```

      ii. *Calculate and store the mean distance.*

```
nndist_mean <- c(colMeans(csr_nndist))
```

      iii. *Calculate and store the values of Z using the mean and variance from part (a) above.*

```
Z <- (nndist_mean - e_hbar)/sd_hbar
```

(d) *Compute the mean and standard deviation of the 1000 simulated H values and compare them to what would be expected under CSR. Are they higher or lower than expected? What could explain this result?*

The theoretical values are slightly smaller. That may happen because assuming the nearest neighbor distances are within a circle will give slightly smaller nearest neighbor distances on average than looking at points in a square because the circle fits inside the square, leaving the corners of the square unattainable under the theoretical calculation. The correction adjusts to use the area and perimeter of a unitized square, which is what was used to simulate points.

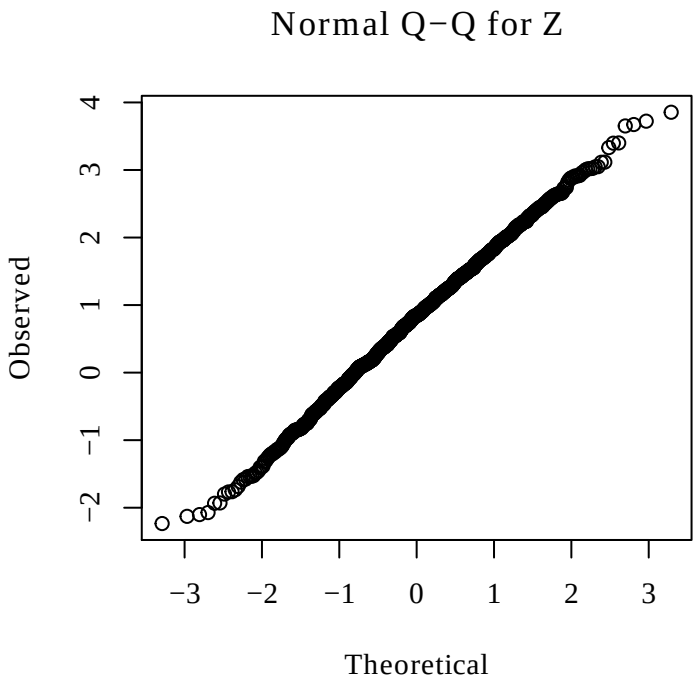
```
emp_mean <- mean(nndist_mean)
emp_sd <- sqrt(var(nndist_mean))

all_hbar <- data.frame(matrix(c(emp_mean, e_hbar, emp_sd, sd_hbar), nrow = 2, ncol = 2, byrow =
rownames(all_hbar) <- c("Mean(Hbar)", "SD(Hbar)")
colnames(all_hbar) <- c("Simulated", "Theoretical")

print(xtable(all_hbar, align = "||l|l|l||",digits=c(4,4,4)))
```

	Simulated	Theoretical
Mean(Hbar)	0.0994	0.0913
SD(Hbar)	0.0104	0.0100

(e) *Produce a qqplot of the z scores. Comment.*

The theoretical quantiles and the observed quantiles fall on a straight line with slight deviations at the tails. This suggests normality is reasonable with possibly some edge effects.

```
qqnorm(Z, xlab = "Theoretical", ylab = "Observed", main = "Normal Q-Q for Z")
```




```
#abline(0,1,col=2)
```

(f) *Use the following formulas for the expected value and variance of* $\bar{H}$*:*

where A *is the area and* P *is the perimeter of the spatial domain (the unit square). Compare the mean and standard deviation from these formulas to those you computed from the simulations above. Does this modification seem to help?*

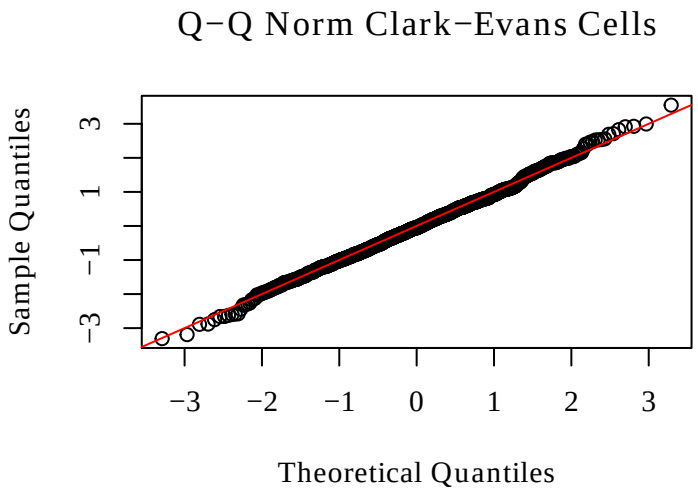
The modification seems to correct for discrepancies.

```
A <- 1
P <- 4
n <- nrow(csr_sim30)

e_hbar1 <- 0.5*sqrt(A/n) + 0.051*(P/n) + 0.041*P/n^(3/2)
var_hbar1 <- 0.0703*A/n^2 + 0.037*P*sqrt(A/n^5)

all_plus <- data.frame(cbind(all_hbar,c(e_hbar1,sqrt(var_hbar1))))
rownames(all_plus) <- rownames(all_hbar)
colnames(all_plus) <- c("Simulated", "Theoretical", "Corrected")

print(xtable(all_plus, align = "||l|l|l|l||", digits = c(5,5,5,5)))
```

	Simulated	Theoretical	Corrected
Mean(Hbar)	0.09939	0.09130	0.09909
SD(Hbar)	0.01040	0.01000	0.01040

(g) *The above procedure is called the Clark-Evans test. Use it to test the null hypothesis of CSR for the cells and redwood data sets. Interpret the results of each test. Also, compute approximate large sample 95% confidence intervals for the mean distance and interpret.*

```
data("cells")
csr_cells <- array(0,c(cells$n, 2, 1000))
nndist_cells <- data.frame(matrix(0,nrow = cells$n, ncol=1000))

set.seed(123)
for(i in 1:1000){
  re <- runifpoint(42, win=owin(c(0,1),c(0,1)))
  csr_cells[,,i] <- c(re$x,re$y)
}


nndist_cells <- apply(csr_cells,3,nndist)
mean_cells <- (colMeans(nndist_cells))

# the cell data were rescaled to the unit square
A <- 1
P <- 4
n.cells <- nrow(csr_cells)

e_hbar.cells <- 0.5*sqrt(A/n.cells) + 0.051*(P/n.cells) + 0.041*P/n.cells^(3/2)
var_hbar.cells <- 0.0703*A/n.cells^2 + 0.037*P*sqrt(A/n.cells^5)
sd_hbar.cells <- sqrt(var_hbar.cells)
Z_cells <- (mean_cells - e_hbar.cells)/sd_hbar.cells

qqnorm(Z_cells, main = "Q-Q Norm Clark-Evans Cells")
abline(a=0,b=1,col=2)
```




```
obs_cells <- nndist(cells)
obs_mean.cells <- mean(obs_cells)
obs_sd.cells <- sqrt(var(obs_cells))

all_cells <- data.frame(cbind(c(obs_mean.cells,obs_sd.cells),c(mean(mean_cells), sqrt(var(mean_
rownames(all_cells) <- rownames(all_hbar)
colnames(all_cells) <- c("Observed", "Simulated", "Corrected")

print(xtable(all_cells, align = "||l|l|l|l||", digits = c(5,5,5,5), caption = "Cells"))
```

	Observed	Simulated	Corrected
Mean(Hbar)	0.12897	0.08240	0.08261
SD(Hbar)	0.01765	0.00736	0.00727

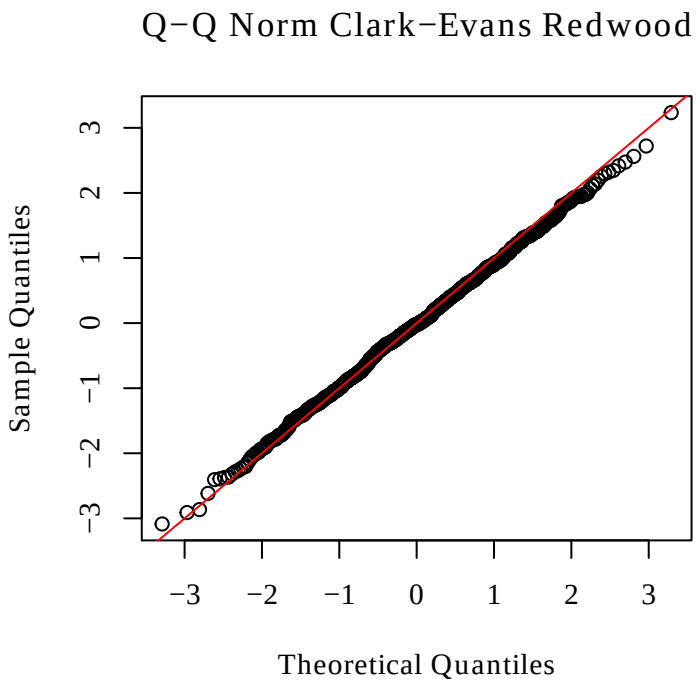
Table 1: Cells

```
lb.cells <- e_hbar.cells - 1.96*sqrt(var_hbar.cells)
ub.cells <- e_hbar.cells + 1.96*sqrt(var_hbar.cells)
```

I am 95% confident the true mean distance for the cells data is between 0.0684 and 0.0969. The observed mean distance for the cells data was 0.129 yielding strong evidence of spatial clustering with the cells data at the 95% confidence level.

```
data("redwood")
# redwood was also rescaled to the unit square

red <- redwood
csr_red <- array(0,c(red$n, 2, 1000))
nndist_red <- data.frame(matrix(0,nrow = red$n, ncol=1000))

set.seed(123)
for(i in 1:1000){
  re <- runifpoint(nrow(csr_red), win=owin(c(0,1),c(0,1)))
  csr_red[,,i] <- c(re$x,re$y)
}
```

```
nndist_red <- apply(csr_red,3,nndist)
mean_red <- (colMeans(nndist_red))

# the cell data were rescaled to the unit square
A <- 1
P <- 4
n.red <- nrow(csr_red)

e_hbar.red <- 0.5*sqrt(A/n.red) + 0.051*(P/n.red) + 0.041*P/n.red^(3/2)
var_hbar.red <- 0.0703*A/n.red^2 + 0.037*P*sqrt(A/n.red^5)
sd_hbar.red <- sqrt(var_hbar.red)
Z_red <- (mean_red - e_hbar.red)/sd_hbar.red

qqnorm(Z_red, main = "Q-Q Norm Clark-Evans Redwood")
abline(a=0,b=1,col=2)
```




```
# need to calculate the observed statistic
obs_red <- nndist(red)
obs_mean <- mean(obs_red)
obs_sd <- sqrt(var(obs_red))

all_red <- data.frame(cbind(c(obs_mean,obs_sd),c(mean(mean_red), sqrt(var(mean_red))),
                            c(e_hbar.red,sqrt(var_hbar.red))))
rownames(all_red) <- rownames(all_cells)
colnames(all_red) <- c("Observed", "Simulated", "Corrected")

print(xtable(all_red, align = "||l|l|l|l||", digits = c(5,5,5,5), caption = "Redwood"))
```

```
lb.red <- e_hbar.red - 1.96*sqrt(var_hbar.red)
ub.red <- e_hbar.red + 1.96*sqrt(var_hbar.red)
```

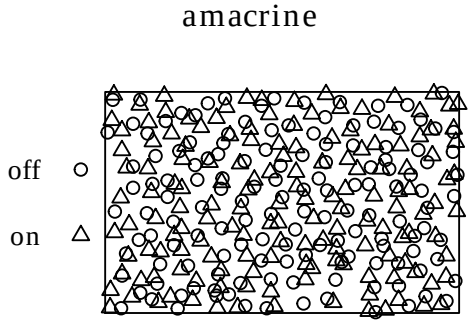
	Observed	Simulated	Corrected
Mean(Hbar)	0.03928	0.06702	0.06713
SD(Hbar)	0.02590	0.00455	0.00481

Table 2: Redwood

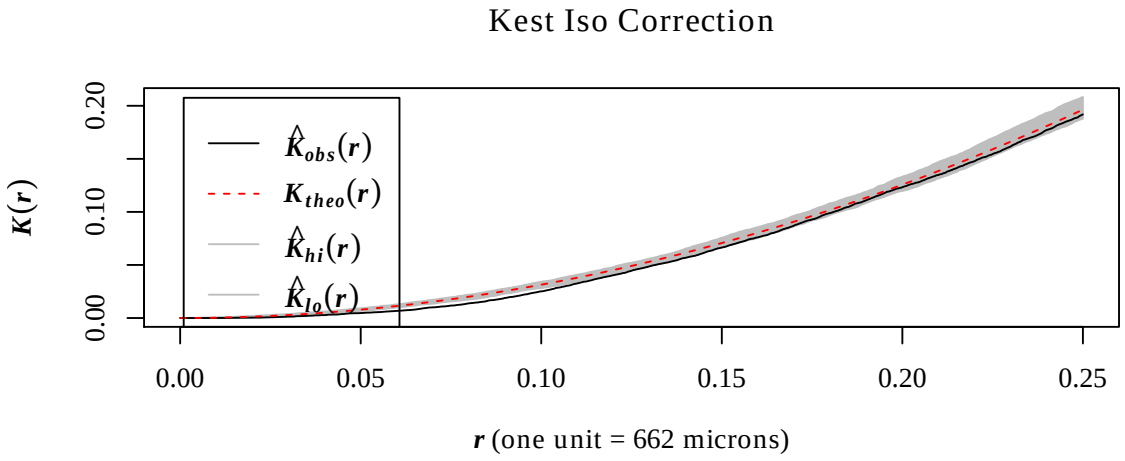
I am 95% confident the true mean distance for the redwood data is between 0.0577 and 0.0766. The observed mean distance for the cells data was 0.0393 yielding strong evidence of regularity in the redwood data at the 95% confidence level.

3. *I am sending you a copy of a paper by Peter Diggle on the use of K and cross K functions in the analysis of spatial point patterns. The data he is referring to are in the amacrine data set in the spatstat library in R. Read the paper and reproduce the analysis. The data are in spatstat (use the command data(amacrine). You do not have to carry out the significance tests he refers to but I would like for you to take the same approach I took on the analysis of the betacells data set we discussed in class. Write up a summary of your analysis. Pay attention to the distinction between the independence and random labelling hypotheses.*

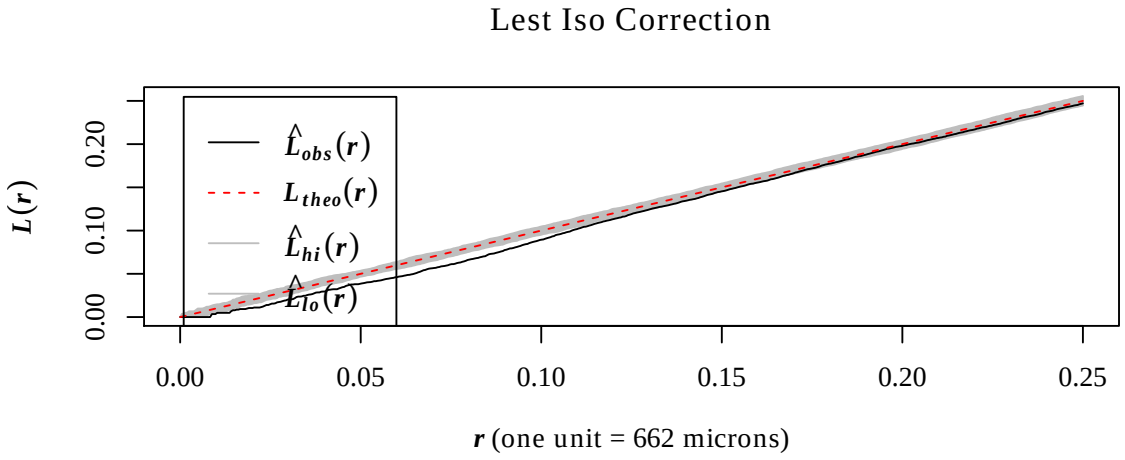
```
data("amacrine")
plot(amacrine, quiet = TRUE)
```



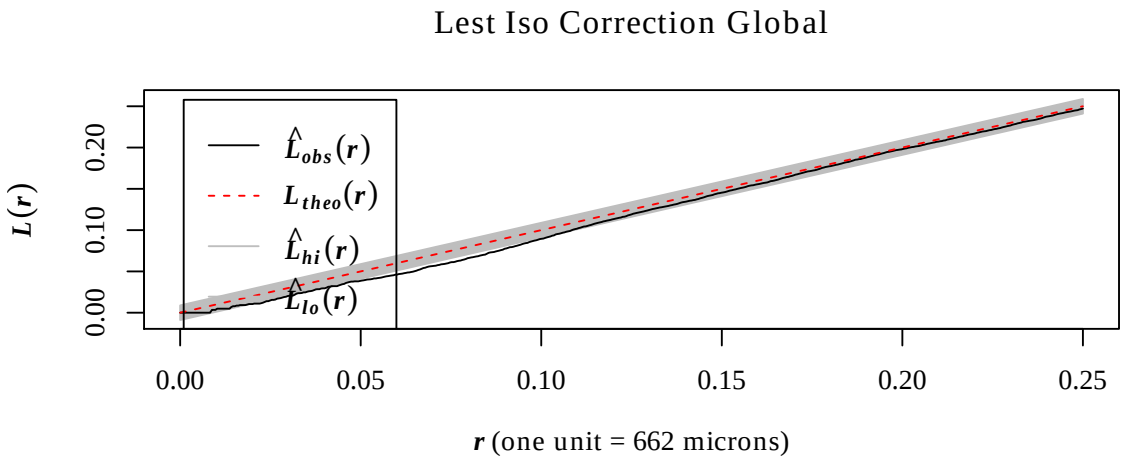

```
#use K it suggests regularity
plot(envelope(amacrine, fun = "Kest", correction = "iso", verbose = FALSE), main = "Kest Iso Correction")
```



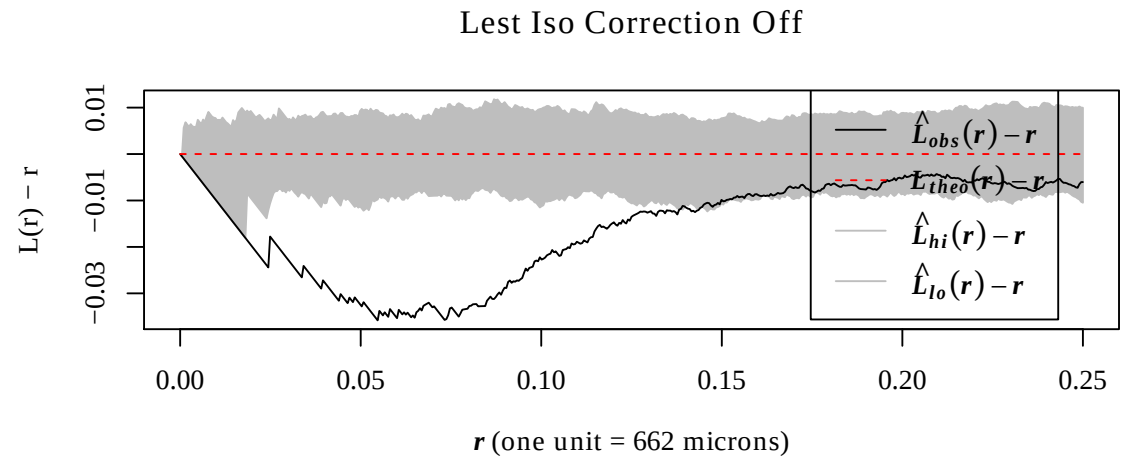

```
plot(envelope(amacrine, fun = "Lest", correction = "iso", verbose = FALSE), main = "Lest Iso Correction")
```



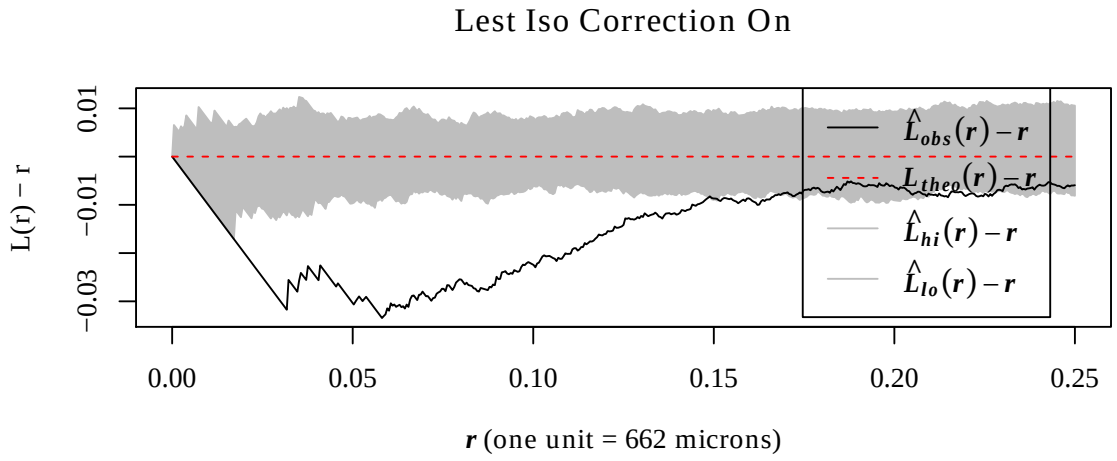

```
plot(envelope(amacrine, fun = "Lest", global = TRUE, correction = "iso", verbose = FALSE), main = "Lest Iso Correction Glo
```



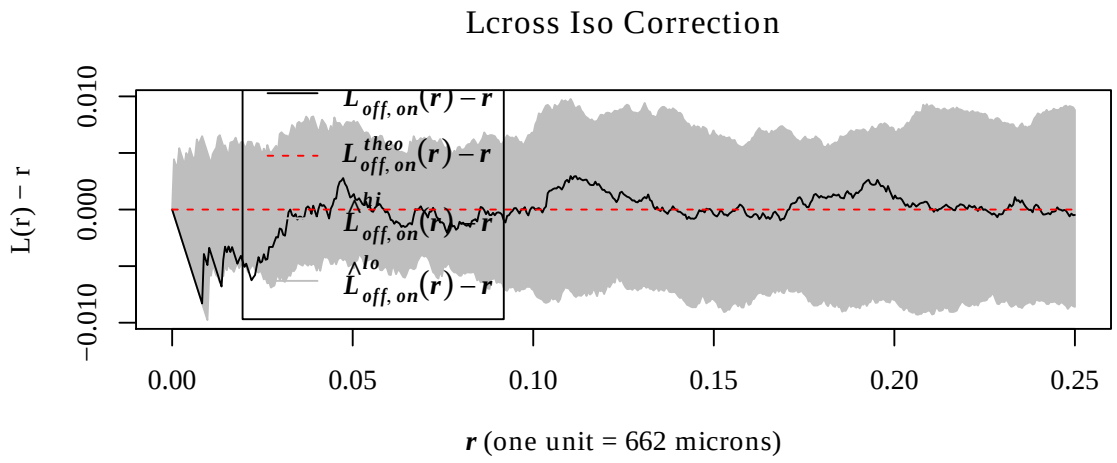

```
off <- split(amacrine)$off
on <- split(amacrine)$on

plot(envelope(off, fun = "Lest", correction = "iso", verbose = FALSE), .-r~r, ylab = "L(r) - r", main = "Lest Iso Correcti
```




```
plot(envelope(on, fun = "Lest", correction = "iso", verbose = FALSE), .-r~r, ylab = "L(r) - r", main = "Lest Iso Correctio
```




```
plot(envelope(amacrine, fun = "Lcross", correction = "iso", verbose = FALSE), .-r~r, ylab = "L(r) - r", main = "Lcross Iso
```




```
require(splancs)
amacrine.poly <- list(x=c(off$x, on$x), y=c(off$y, on$y))

# from the paper, make max dist = 0.25*min(side(rectangle(A)))
# shorter side = height = 1
# 0.25*1 = 0.25
```

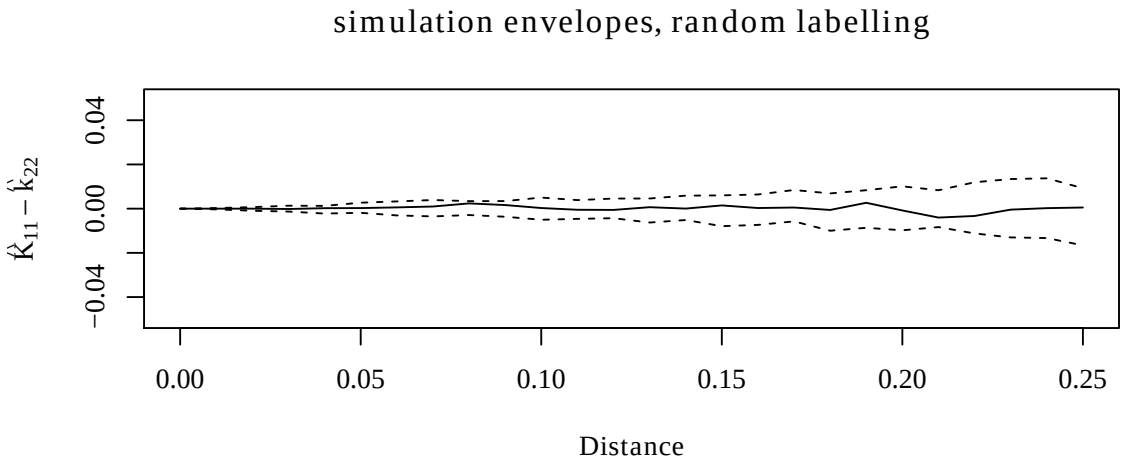
Let 11 = on and 22 = off.

H_0: $K_{11} = K_{22}$; random labelling

H_A: $K_{11} \neq K_{22}$; no random labelling

There is no evidence against random labelling as the difference in estimated K functions between the on and off cells lies within the simulation envelopes.

```
# random labelling
# ho: randomly labelling
# no evidence against random labelling
h <- c(seq(0,0.25,by=0.01))
khat.on <- khat(as.points(on), bboxx(bbox(as.points(amacrine.poly))), s = h)
khat.off <- khat(as.points(off), bboxx(bbox(as.points(amacrine.poly))), s = h)

kdiff <- khat.on-khat.off
plot(h, kdiff, xlab="Distance", ylab=expression(hat(K)[11] - hat(k)[22]),
     type="l", main="simulation envelopes, random labelling", ylim=c(-.05,.05))

env.lab <- Kenv.label(as.points(on),as.points(off),
                      bboxx(bbox(as.points(amacrine.poly))), nsim = 99,s=h, quiet = TRUE)

lines(h, env.lab$upper, lty=2)
lines(h, env.lab$lower, lty=2)
```



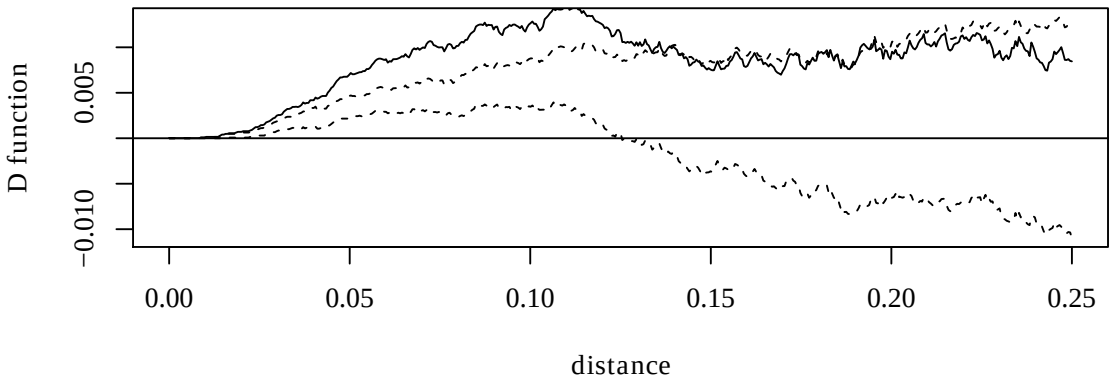
H_0: $K_{11} = K_{12}$; random labelling

H_A: $K_{11} \neq K_{12}$; no random labelling

There is evidence against random labelling as the difference line lies outside the simulation envelopes.

```
## D function calculation - cross K and K.on
# First calculate the observed value of D
# I had to modify the code a bit because betacells
# has two types of marks and Kcross had trouble with that.
# I only needed to work with
# the binary type (on,off) marks.
amacrine.new<-amacrine
amacrine.new$marks<-amacrine$marks
amacrine.newoff<-split(amacrine,f=marks(amacrine.new))$off
amacrine.newon<-split(amacrine,f=marks(amacrine.new))$on
## Set up
mark.vec<-marks(amacrine.new)
sim.mat<-matrix(0,nrow=513,ncol=100)
Dfun<-Kcross(amacrine.new)$iso - Kest(amacrine.newon)$iso
sim.mat[,1]<-Dfun
## Loop
```

```
for(i in 2:100){
   indx<-sample(1:294,rep=F)
   amacrine.new$marks<-amacrine.new$marks[indx]
   sim.mat[,i]<-Kcross(amacrine.new)$iso -
               Kest(amacrine.newon)$iso
}
Dfun.min<-apply(sim.mat[,-1],1,min)
Dfun.max<-apply(sim.mat[,-1],1,max)
hdist<-Kest(amacrine)$r
minD <- min(Dfun.min)
maxD <- max(Dfun.max)
maxHdist <- max(hdist)
plot(c(0,maxHdist),c(minD,maxD),type="n",xlab="distance",ylab="D function")
abline(h=0)
lines(hdist,Dfun.min,lty=2)
lines(hdist,Dfun.max,lty=2)
lines(hdist,Dfun)
```

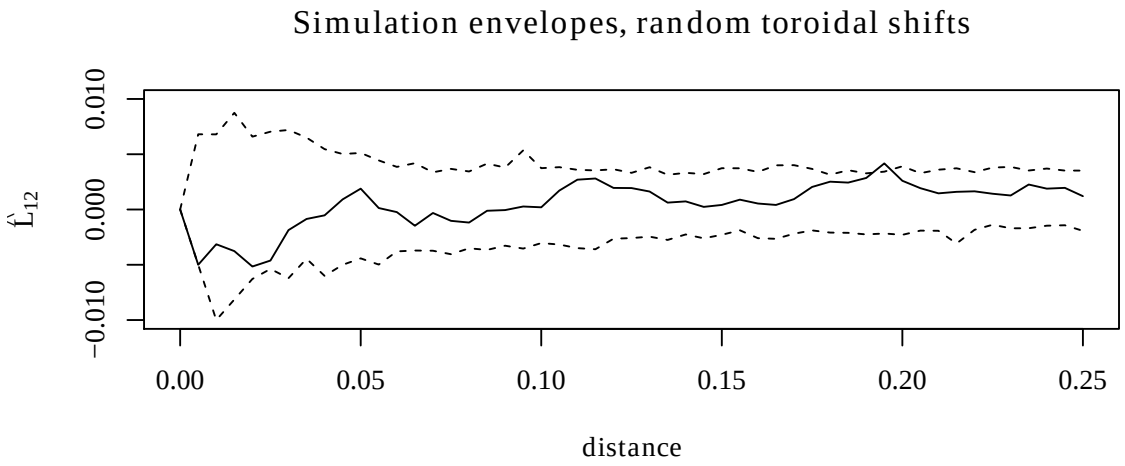


H_0 : independence

H_A: no independence

There is no evidence against independence as the line lies within the envelope.

```
amacrine.poly <- list(x=c(amacrine.newon$x, amacrine.newoff$x),
y=c(amacrine.newon$y,amacrine.newoff$y))
# specify the range of radii for the plot
h<-seq(0,0.25,.005)
# Produce a modified cross L function
Lcross.plot<-sqrt(k12hat(as.points(amacrine.newon),as.points(amacrine.newoff),
            bboxx(bbox(as.points(amacrine.poly))),h)/pi) - h
plot(h,Lcross.plot, xlab="distance",
     ylab=expression(hat(L)[12]), ylim = c(-0.01,0.01),type="l",
     main="Simulation envelopes, random toroidal shifts")
# Get the bounds on the simulation envelope.
Lcross.env<- Kenv.tor(as.points(amacrine.newon), as.points(amacrine.newoff),
     bboxx(bbox(as.points(amacrine.poly))), nsim=499, s=h, quiet = TRUE)
lines(h, sqrt(Lcross.env$upper/pi)-h, lty=2)
lines(h, sqrt(Lcross.env$lower/pi)-h, lty=2)
```



Overall, CSR is violated as we cannot assume random labelling.