Solutions to Exercise Sheet 1

1. Let X be random variable for the number of calls the centre receives on any one day. From the question,we know $X \sim \text{Poisson}(1523)$. We are therefore interested in computing $P(X > 1600) = 1 - P(X \leq 1600)$. Let $F_X(x) \equiv P(X \leq x)$, be the cumulative distribution function for X. The problem equates to computing $1 - F_X(1600)$.

   Looking at the documentation for `ppois` (type `?ppois`), we can perform this computation with

   `1-ppois(1600,1523)`.

   This gives $P(X > 1600) \approx 0.02421$

2. Here is some example code for simulating a homogeneous temporal Poisson process. I have written mine as a function, however, you could easily just write it as a script.

```
# This function simulates events from a homogeneous Poisson process
# with intensity lambda on the interval (0,T]
homogeneouspp = function(lambda,T)
{
n = rpois(1,lambda*T)
E = runif(n,min=0,max=T)
E = sort(E)
}
```

   We can then commit this function to memory with the command

```
source('homogeneouspp.R')
```

   To simulate the Poisson process stated in the question, we simply call

```
N = homogeneouspp(10,10)
```

   To store 100 realisations of this process, we could perform a `for` loop and store in a list. E.g.

```
l = list()
for (ii in seq(1,100,1))
{
N = homogeneouspp(10,10)
l[[ii]] = N
}
```

   (a) To verify the sample mean of events in the interval $(3, 5]$ is consistent with the expected number of events we first compute the sample mean.

```
# Initialise the sum of events in (3,5]
S = 0

# loop through every realisation and sum up the number of events in (3,5]
for (ii in seq(1,100,1))
```

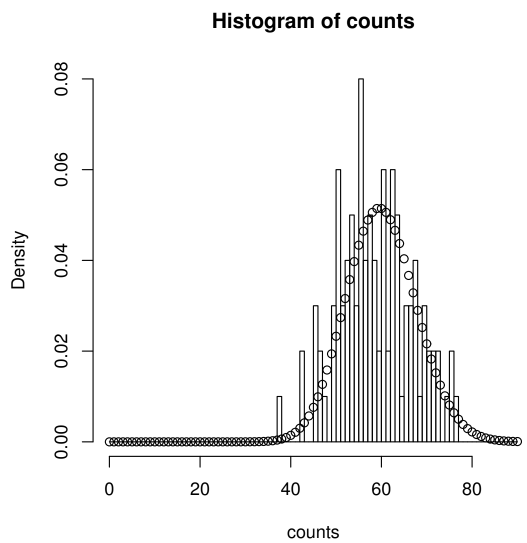
```
{
# Extract events that exist between 3 and 5
E = l[[ii]]
Eshort = E[ (E > 3) & (E <= 5) ]

# add to the sum
S = S + length(Eshort)
}

mean_events = S/100
```

When I run this, I get 19.7. This is consistent with $E\{N((3,5])\} = 2 \times 10 = 20$.

(b) To extract the counts for the histogram I run

```
# initialise my count vector
counts = rep(0,1,100)

# loop though every realisation and extract the counts in (2,8]
for (ii in seq(1,100,1))
{
E = l[[ii]]
Eshort = E[ (E > 2) & (E <= 8) ]

# store in counts
counts[ii] = length(Eshort)
}

# plot histogram
hist(counts,freq=F,breaks = seq(0,90,1))

#plot poisson(60) pdf over the top
points(seq(0,90,1),dpois(seq(0,90,1),60))
```

I get the following plot



This certainly seems consistent. Discrepancies are due to random sampling. As the number of realisations increases (currently 100), a closer match will be achieved.

(c) Store the two sets of counts in separate vectors

```
# initialise my count vectors
counts1 = rep(0,1,100)
counts2 = rep(0,1,100)

# loop though every realisation and extract the counts in (2,8]
for (ii in seq(1,100,1))
{
E = l[[ii]]
Eshort1 = E[ (E > 2) & (E <= 5) ]

# store in counts1
counts1[ii] = length(Eshort1)

Eshort2 = E[ (E > 6) & (E <= 8) ]

# store in counts
counts2[ii] = length(Eshort2)
}

# compute correlation
rho = cor(counts1,counts2)
```

I get $\rho = 0.10636$. This certainly seems small, but we would need to do a formal test on to see if ρ is statistically significant or is consistent with the correlation being zero. We'll talk more about hypothesis testing on Wednesday.

3. We will use the rpoispp command to simulate homogeneous Poisson data. We will store 100 realisations of the process in a list.

```
l = list()
for (ii in seq(1,100,1))
{
N = rpoispp(0.5,win = c(0,10,0,10))
l[[ii]] = N
}
```

We will then sum the number of events in each pattern. This is stored as the field n.

```
# Initialise the sum of events in (3,5]
S = 0

# loop through every realisation and sum up the number of events in (3,5]
for (ii in seq(1,100,1))
{
# Extract number of events and add to sum

S = S + l[[ii]]$n
}

mean_events = S/100
```

When I run this, I get the mean number of events to be 50.34, this is consistent with the expected number of events which equals 0.5*10*10 = 50.

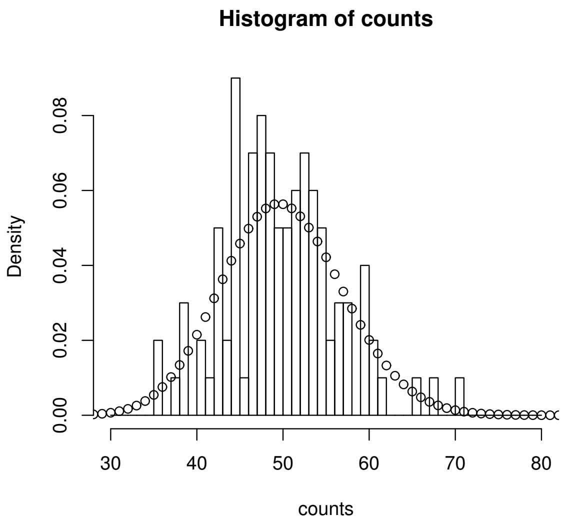
4. Here is my code for simulating this

```
# simulate homogeneous Poisson process with intensity max{lambda(t)}
N = homogeneouspp(100,100)

# initialise vector of probabilities Bernoulli indicators
B = rep(0,length(N))

# For each event, keep or discard by performing a bernoulli random trial

for (ii in seq(1,length(N),1))
{
# p = lambda(t)/max{lambda(t)}
p = exp(-((N[[ii]]-50)^2)/50)
B[ii] = rbinom(1,1,p)
}

#Turn these into logicals and extract the events that need keeping
B = as.logical(B)
N = N[B]
```

5. To plot a histogram of the event counts, I run

```
# initialise my count vector
counts = rep(0,1,100)

# loop though every realisation and extract the counts
for (ii in seq(1,100,1))
{
counts[ii] = l[[ii]]$n
}

# plot histogram
hist(counts,freq=F,breaks = seq(30,80,1))

#plot poisson(50) pdf over the top
points(seq(0,90,1),dpois(seq(0,90,1),50))
```

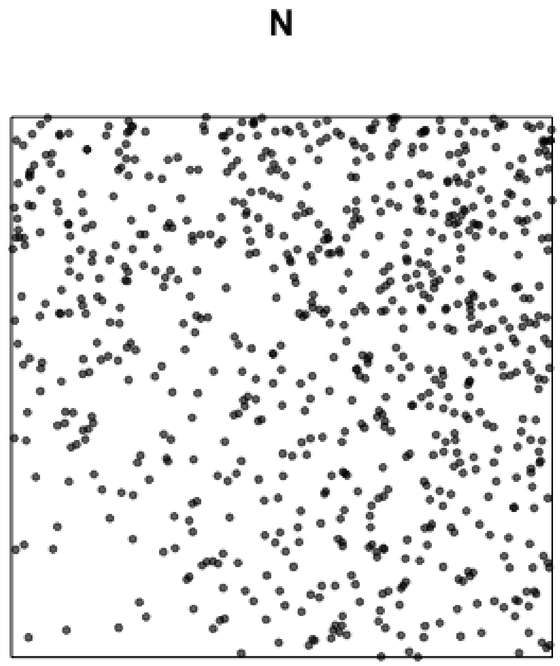
I get the following plot


Histogram of counts
Density
0.00
0.02
0.04
0.06
0.08
30
40
50
60
70
80
counts

6. This can be achieved with the following code

```
N = rpoispp(function(x,y) {sqrt(x^2 + y^2)},win = c(0,10,0,10))
plot(N,type="p",pch=19,cex=0.5)
```

and I obtain the following plot:



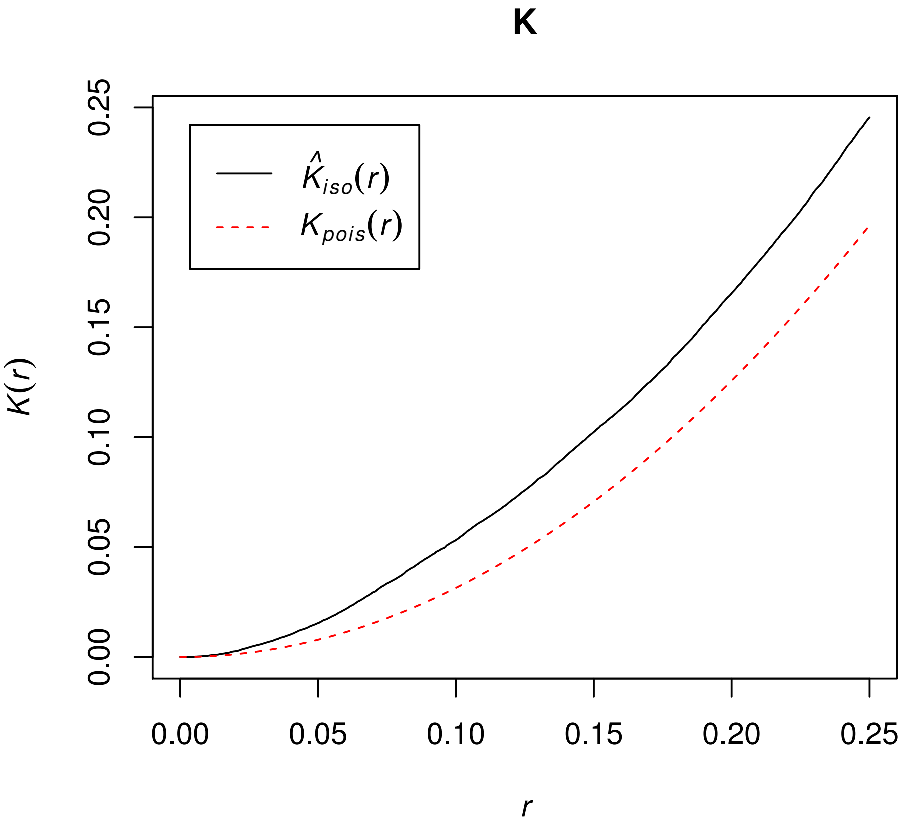
7. The important step here is to load the data, and then convert it to `ppp` class for point patterns in `SpatStat`. For example

```
N = read.table("data1.csv",sep=",")
N = as.ppp(N,c(0,1,0,1))
```

We can then compute the K function and plot it as

```
K = Kest(N,correction = "Ripley")
plot(K)
```

For example, for `data1.csv`, I obtain



To plot the $L(r) - r$ function, use `plot(K,sqrt(./pi)-r ~ r)`.