Paper Replication

Make simulated spp data

I'm trying to figure out the spatstat package here.

```
library(spatstat)
```

```
##
## spatstat 1.42-2       (nickname: 'Barking at Balloons')
## For an introduction to spatstat, type 'beginner'
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:spatstat':
##
##     area
```

```
set.seed(1234)

#set the desired window.
my_window <- owin(xrange = c(-1, 12), yrange = c(-1, 12))

#Run simulation with homogenous poisson process with intensity lambda = 0.5
simulation_a <- rpoispp(0.5, win = my_window)

#Run simulation with homogenous poisson process with intensity lambda = 4
simulation_b <- rpoispp(4, win = my_window )
```

The next two simulations are slight more complicated and will require more code. The first is given by $\lambda(x,y) = 100\times$ the $N_2(\binom{5}{5}\begin{pmatrix} 3 & 0.5\sqrt{6} \\ 0.5\sqrt{6} & 2 \end{pmatrix})$. First I make a function, mvdnorm, that computes the density for a bivariate gaussian distribution. Then I make the simulation.

```
mvdnorm <- function(x, y, mu, sigma){
  p <- c(x, y)
  (2 * pi)^{-1} * det(sigma)^{-1 / 2} * exp(-1 / 2 * (t(p-mu) %*% solve(sigma) %*% (p-mu)))
}

mvdnorm_for_rpoispp <- function(x, y, mu, sigma){
  d <- rep(NA, length(x))
  for(i in 1:length(d)){
    d[i] <- mvdnorm(x[i], y[i], mu, sigma)
  }
  return(d)
}
my_mu <- c(5, 5)
```

```
my_sigma <- matrix(c(3, 0.5*sqrt(6), 0.5*sqrt(6), 2), nrow = 2, byrow = T)
lambda_c <- function(x, y) {100 * mvdnorm_for_rpoispp(x, y, my_mu, my_sigma)}

simulation_c <-  rpoispp(lambda = lambda_c, win = my_window)
```

The final function is just a drop off function. It is inhomogenous with $\lambda(x, y) = 0.2$ for $x < 6$ and $\lambda(x, y) = 4$, otherwise.

```
lambda_d <- function(x, y){
  l <- rep(4, length(x))
  for(i in 1:length(l))
    if(x[i]<6) l[i] = 0.2
  l
}

simulation_d <- rpoispp(lambda_d, win = my_window)
```

Next construct a search function as presented in the paper

We will make this whole thing workable by having a clever matrix with all the relevant information.

First, Create the lattice

```
my_lattice <- cbind(c(rep(1, 10), rep(2, 10), rep(3, 10), rep(4, 10), rep(5, 10),rep(6, 10), rep(7, 10)
```

Next lets make a nearest-neighbor function. It will use euclidean distance to find a nearest neighbor. It works returns both the point and the distance.

```
d2 <- function(a, b) (a[1]-b[1])^2 + (a[2]- b[2])^2
nearest_point <- function(x, y, d){
  my_min <- c(NA, NA, Inf)
  for(i in 1:nrow(d)){
    current_distance <- d2(c(d[i,1], d[i,2]), c(x, y))
    current_min <- my_min[3]
    if(current_distance < current_min){
      my_min <- c(d[i,1], d[i,2], current_distance)
    }
  }
  my_min
}
```

Next let's break up the lattice into validation sets.

```
val_sets <- function(sets_number, total){
  sample(1:total, size = total, replace = FALSE) %/% sets_number + 1
}

cbind(val_sets(10, nrow(my_lattice)), my_lattice)
```

```
##        [,1] [,2] [,3]
##   [1,]    3    1    1
##   [2,]    9    1    2
##   [3,]    3    1    3
##   [4,]    6    1    4
##   [5,]    7    1    5
##   [6,]    3    1    6
##   [7,]    9    1    7
##   [8,]    4    1    8
##   [9,]    7    1    9
##  [10,]    1    1   10
##  [11,]    9    2    1
##  [12,]    4    2    2
##  [13,]    7    2    3
##  [14,]    9    2    4
##  [15,]    3    2    5
##  [16,]    4    2    6
##  [17,]    5    2    7
##  [18,]    2    2    8
##  [19,]    7    2    9
##  [20,]    7    2   10
##  [21,]    1    3    1
##  [22,]    6    3    2
##  [23,]   10    3    3
##  [24,]    8    3    4
##  [25,]    5    3    5
##  [26,]   10    3    6
##  [27,]    8    3    7
##  [28,]    3    3    8
##  [29,]    9    3    9
##  [30,]    6    3   10
##  [31,]    6    4    1
##  [32,]    2    4    2
##  [33,]    5    4    3
##  [34,]   10    4    4
##  [35,]    7    4    5
##  [36,]    8    4    6
##  [37,]    8    4    7
##  [38,]    2    4    8
##  [39,]    2    4    9
##  [40,]    4    4   10
##  [41,]    9    5    1
##  [42,]    7    5    2
##  [43,]    8    5    3
##  [44,]    4    5    4
##  [45,]    1    5    5
##  [46,]    9    5    6
##  [47,]    6    5    7
##  [48,]    2    5    8
##  [49,]    5    5    9
##  [50,]    1    5   10
##  [51,]    4    6    1
##  [52,]    2    6    2
##  [53,]   10    6    3
##  [54,]    2    6    4
##  [55,]   11    6    5
##  [56,]    2    6    6
##  [57,]    2    6    7
##  [58,]    5    6    8
##  [59,]    5    6    9
##  [60,]    7    6   10
##  [61,]    1    7    1
##  [62,]   10    7    2
##  [63,]    1    7    3
##  [64,]    1    7    4
##  [65,]    5    7    5
##  [66,]    3    7    6
##  [67,]   10    7    7
##  [68,]    3    7    8
##  [69,]    9    7    9
##  [70,]   10    7   10
##  [71,]    8    8    1
##  [72,]    6    8    2
##  [73,]    1    8    3
##  [74,]    6    8    4
##  [75,]   10    8    5
##  [76,]    5    8    6
##  [77,]    8    8    7
##  [78,]    4    8    8
##  [79,]    4    8    9
##  [80,]    6    8   10
##  [81,]    7    9    1
##  [82,]    8    9    2
##  [83,]    8    9    3
##  [84,]    9    9    4
##  [85,]   10    9    5
##  [86,]    4    9    6
##  [87,]    3    9    7
##  [88,]    3    9    8
##  [89,]    7    9    9
##  [90,]    5    9   10
##  [91,]    5   10    1
##  [92,]    1   10    2
##  [93,]    6   10    3
##  [94,]    3   10    4
##  [95,]    4   10    5
##  [96,]   10   10    6
##  [97,]    8   10    7
##  [98,]    2   10    8
##  [99,]    9   10    9
## [100,]    6   10   10
```

I'm moving in a weird order because I'm struggling. SO I NEED TO FIX THIS LATER