ST5226 Assignment 1

Yeo Ming Jie, Jonathan (A0164616A)

Question 1

```
# Load data
abc <- readRDS("abc.rds") # SpatialPolygonsDataFrame Object
class(abc)
```

```
## [1] "SpatialPolygonsDataFrame"
## attr(,"package")
## [1] "sp"
```

```
str(abc@data[c("BIR74","SID74")])
```

```
## 'data.frame':    100 obs. of  2 variables:
##  $ BIR74: num  4672 1333 487 1570 1091 ...
##  $ SID74: num  13 0 0 NA 1 0 7 6 8 5 ...
```

```
# BIR74 - number of births // SID74 - number of birth defects
# Both over period of 4 years
```

(a) We observe that `abc` is a `SpatialPolygonsDataFrame` object.

```
slot(abc,"proj4string") # Obtaining CRS: NAD27
```

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

```
## Coordinate Reference System:
## Deprecated Proj.4 representation: +proj=longlat +datum=NAD27 +no_defs
## WKT2 2019 representation:
## GEOGCRS["unknown",
##     DATUM["North American Datum 1927",
##         ELLIPSOID["Clarke 1866",6378206.4,294.978698213898,
##             LENGTHUNIT["metre",1]],
##         ID["EPSG",6267]],
##     PRIMEM["Greenwich",0,
##         ANGLEUNIT["degree",0.0174532925199433],
##         ID["EPSG",8901]],
##     CS[ellipsoidal,2],
##         AXIS["longitude",east,
##             ORDER[1],
##             ANGLEUNIT["degree",0.0174532925199433,
##                 ID["EPSG",9122]]],
##         AXIS["latitude",north,
##             ORDER[2],
##             ANGLEUNIT["degree",0.0174532925199433,
##                 ID["EPSG",9122]]]]
```

```
# Transform to WGS84
abc2 <- spTransform(abc, CRS("+proj=longlat +datum=WGS84"))
```

(b) Checking the `proj4string` argument, CRS of `abc`: `NAD27`.

(c) We use the `spTransform` function to transform the coordinate system to `WGS84`.

(d) The rate of birth defects per $10,000$ births and;

(e) that in a typical year are computed in the following manner:

$$\text{Rate of SID74 per 10,000} = \frac{SID74}{BIR74} \times 10000 \qquad \text{(over 4 year period)}$$

$$\text{Rate of SID74 per 10,000} = \frac{SID74}{BIR74} \times 10000 \times \frac{1}{4} \qquad \text{(in a typical year)}$$

```
# Compute rate of BIR74 (birth defects) per 10,000
abc$r = abc$SID74/abc$BIR74 /10 * 1e+5 # over 4 year period
abc$r_annum = abc$SID74/abc$BIR74 /10 * 1e+5 / 4 # per year
```

(f) It is noted that the entry for `SID74` is `NA` for the county of Anson, which will be set to zero. The previously computed rates are also set to zero for Anson.

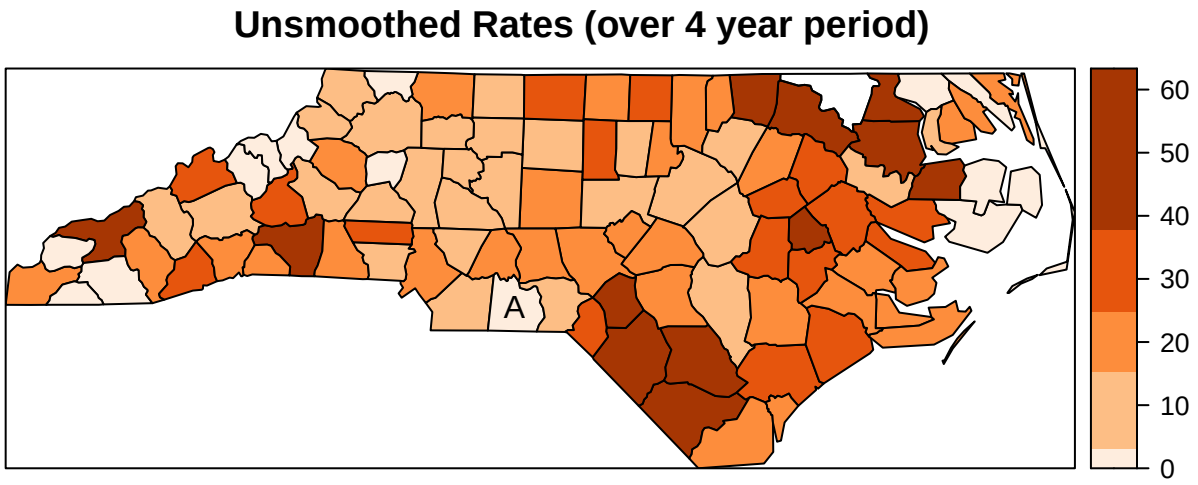
```
na_index = which(is.na(abc$SID74)) # obtain index of NA entry
county_name = as.character(abc$NAME[na_index]); county_name
```

```
## [1] "Anson"
```

```
abc$SID74[na_index] <- 0 # set to zero

# Also set the assigned rates to 0
abc$r[na_index] <- 0; abc$r_annum[na_index] <- 0
```

(g) Next, we plot the following chloropleth map of raw rates per 10,000 births over the 4 year period. The county of Anson is indicated with the letter A.

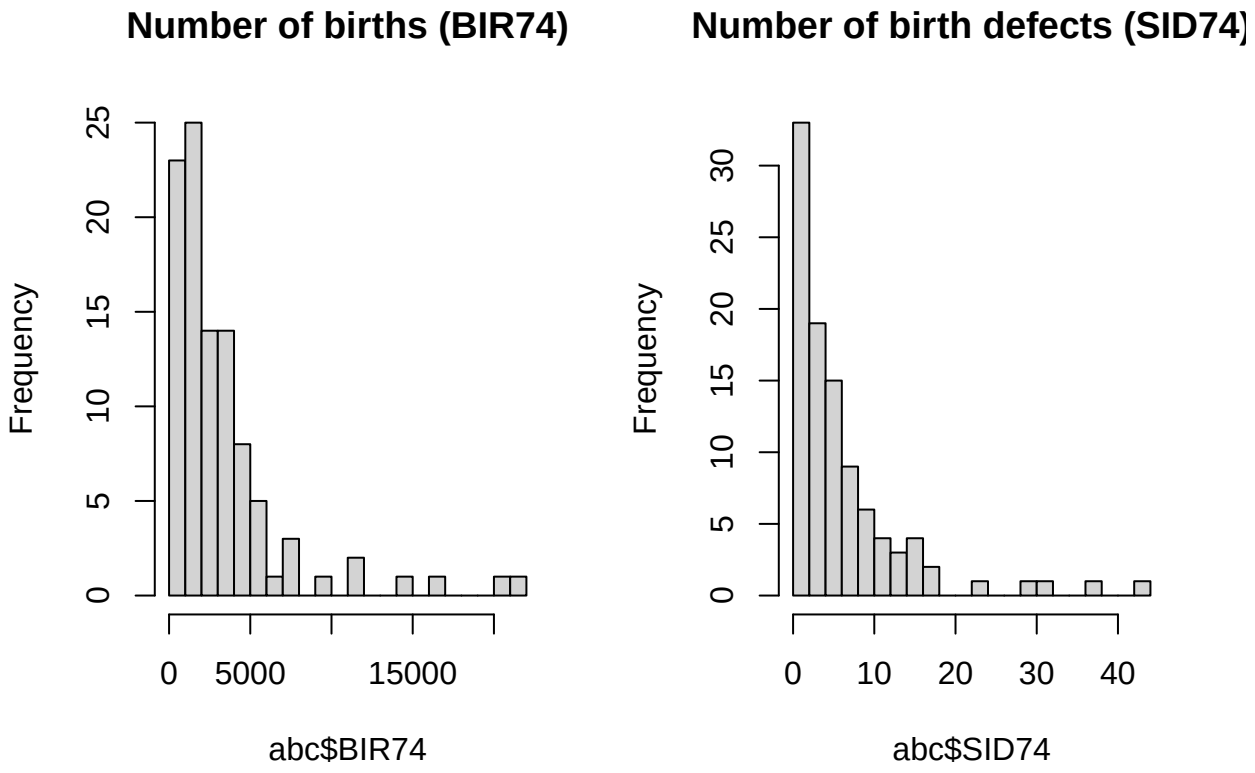
```
# Chloropleth map plot
library(RColorBrewer); library(classInt); library(sp)

con_names = abbreviate(abc$NAME)

pal <- brewer.pal(n=5, "Oranges")
spplot(abc, "r", col.regions=pal, main="Unsmoothed Rates (over 4 year period)",
       at=classIntervals(abc$r, n=5, "fisher")$brks, # Using Natural (Fisher) Breaks
       sp.layout=list("sp.text", coordinates(abc)[4,],"A"))
```




```
# Checking:
#spplot(abc, "r", col.regions=pal, main="Unsmoothed Rates (over 4 year period)",
#       at=classIntervals(abc$r, n=5, "fisher")$brks,
#       sp.layout=list("sp.text", coordinates(abc),con_names)) # Overlay all County names on plot
```

(h) We conduct some exploratory data analysis by plotting the histograms and studying counties with zero, high and low raw rates of birth defects. We observe that there are counties, such as Greene and Washington, with high rates, but a low count in both birth defects and birth counts (i.e. population size), which is an indication of the "small number problem".

```
par(mfrow=c(1,2))
# Histogram Plots
hist(abc$BIR74, breaks = 20, main = "Number of births (BIR74)") # no. of births
hist(abc$SID74, breaks = 20, main = "Number of birth defects (SID74)") # no. birth defects
```



```
# Illustration of "Small Number Problem"
df <- abc@data[,c("NAME","BIR74","SID74","r")]
head(df[order(df$r),]) # zero birth defects
```

```
##            NAME BIR74 SID74 r
## 37003 Alexander  1333     0 0
## 37005 Alleghany   487     0 0
## 37007     Anson  1570     0 0
## 37011     Avery   781     0 0
## 37029    Camden   286     0 0
## 37043      Clay   284     0 0
```

```
df[order(df$r),][15:20,] # low rates
```

```
##           NAME BIR74 SID74        r
## 37169   Stokes  1612     1 6.203474
## 37159    Rowan  4606     3 6.513244
## 37025 Cabarrus  4099     3 7.318858
## 37189  Watauga  1323     1 7.558579
## 37197   Yadkin  1269     1 7.880221
## 37059    Davie  1207     1 8.285004
```

```
tail(df[order(df$r),]) # high rates
```

```
##              NAME BIR74 SID74        r
## 37079      Greene   870     4 45.97701
## 37093        Hoke  1494     7 46.85408
## 37091    Hertford  1452     7 48.20937
## 37083     Halifax  3608    18 49.88914
## 37187  Washington   990     5 50.50505
## 37131 Northampton  1421     9 63.33568
```

(i) To deal with the small number problem, we could consider the grouping of counties with roughly equal population sizes for a fairer means of comparison in rates between different groups. (i.e. the denominator of birth counts should be roughly equal). However, this results in the loss of spatial/geographical information that we may wish to convey when regions are aggregated.

(j) Next, we consider the approach of using smoothed rates, with the use of the following ratio smoother, given by

$$\hat{\xi}_i = \frac{\sum_{j=1}^{100} w_{ij} Y_j}{\sum_{j=1}^{100} w_{ij} n_j}$$

where Y_j is the number of defect cases and n_j the number of births in county j. The smoothing weights are given by

$$w_{ij} = \exp(-\frac{d_{ij}^2}{1000})$$

Suppose $Y_j \sim Pois(\xi n_j)$ for all j, then note that

$$Var(\frac{Y_i}{n_i}) = \frac{1}{n_i^2} Var(Y_i) = \frac{1}{n_i^2} \times \xi n_i = \frac{\xi}{n_i}$$

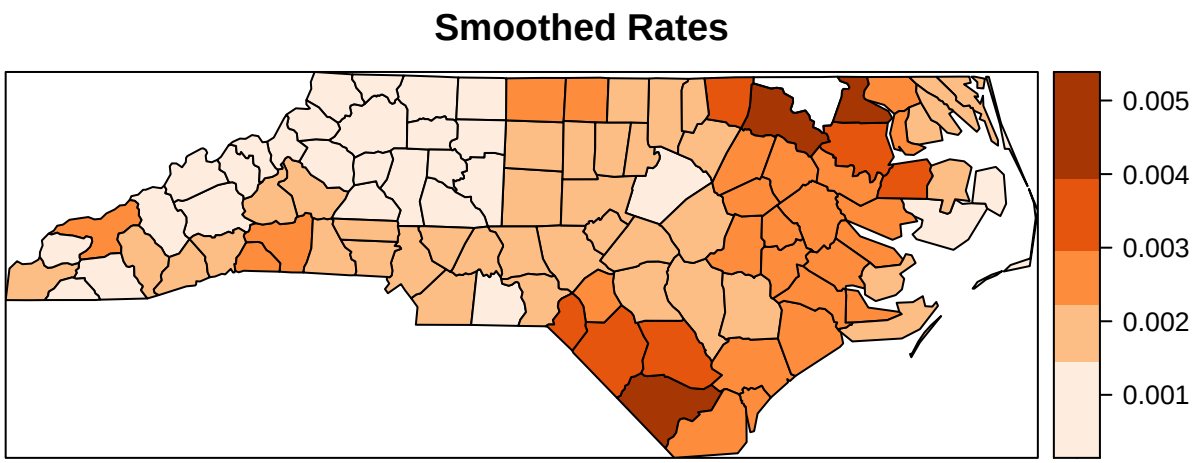
and hence,

$$
\begin{aligned}
Var(\hat{\xi}_i) &= Var(\frac{\sum w_{ij} Y_j}{\sum w_{ij} n_j}) \\
&= \frac{\sum w_{ij}^2 Var(Y_j)}{(\sum w_{ij} n_j)^2} \\
&= \frac{\sum w_{ij}^2 n_j \xi}{(\sum w_{ij} n_j)^2} \\
&\leq \xi \frac{\sum w_{ij} n_j}{(\sum w_{ij} n_j)^2} \qquad \text{since } w_{ij}^2 \leq w_{ij} \\
&= \xi \frac{1}{\sum w_{ij} n_j} \leq \frac{\xi}{n_i}
\end{aligned}
$$

since $0 \leq w_{ij} \leq 1$.

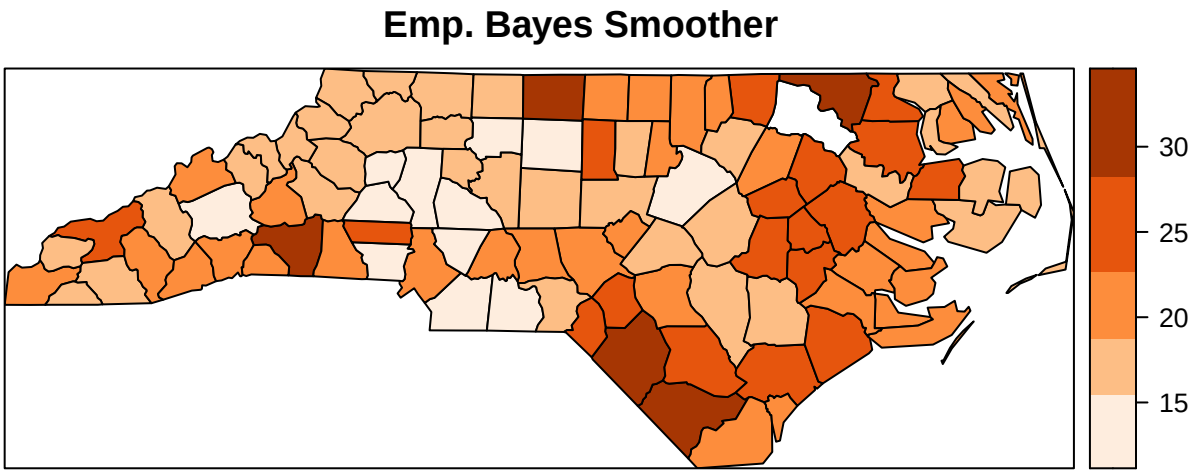
(k) Mapping the smoothed weights using the ratio smoother above, we obtain the following result:

```
# Mapping Smoothed Rates
d_ij = spDists(coordinates(abc), longlat = TRUE) # pairwise distances (100 x 100 matrix)
w_ij = exp(-(d_ij^2)/1000) # weights

# Ratio Smoother
Y_j = abc$SID74; n_j = abc$BIR74
xi_hat = colSums(w_ij*Y_j)/colSums(w_ij*n_j); abc$xi_hat = xi_hat

# Generating plot using smoothed rates
spplot(abc, "xi_hat", col.regions=pal, main="Smoothed Rates",
       at=classIntervals(abc$xi_hat, n=5, "fisher")$brks) # Using Natural Breaks
```



(l) We consider the alternative use of the method of Empirical Bayes for mapping of our smoothed rates:

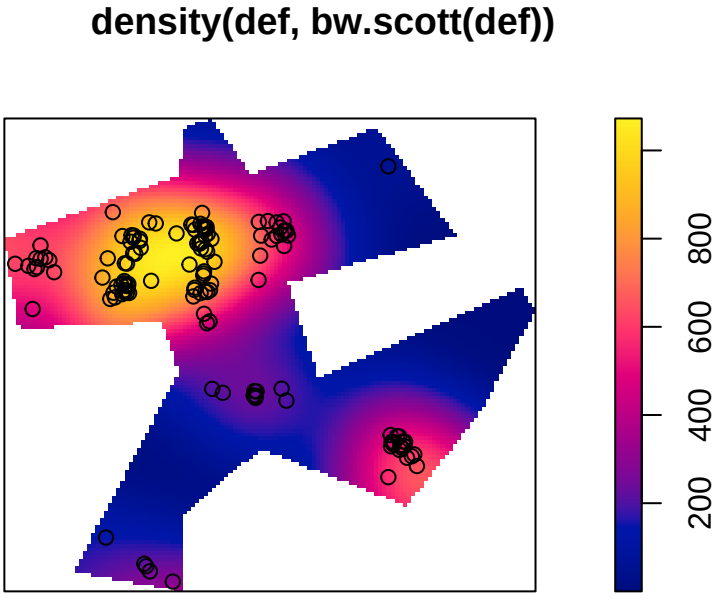
```
library(spdep)
# Using Method of Empirical Bayes
abc$eb <- EBest(abc$SID74,abc$BIR74)$estmm /10 * 1e+5 # Extract empirical Bayes estimates, then scale
spplot(abc,"eb",col.regions=pal, main="Emp. Bayes Smoother",
       at=classIntervals(abc$eb,n=5,"fisher")$brks)
```



(m) The range of the smoothed rates appears to be smaller for the ratio smoother as compared to that from the empirical Bayes estimator, possibly suggesting that the likelihood-based ratio smoother is more accurate. This makes sense as Bayes smoothing does not take into account geographical information. Typical Bayesian smoothing also requires the specification of the prior distribution model for ξ_i.

Question 2

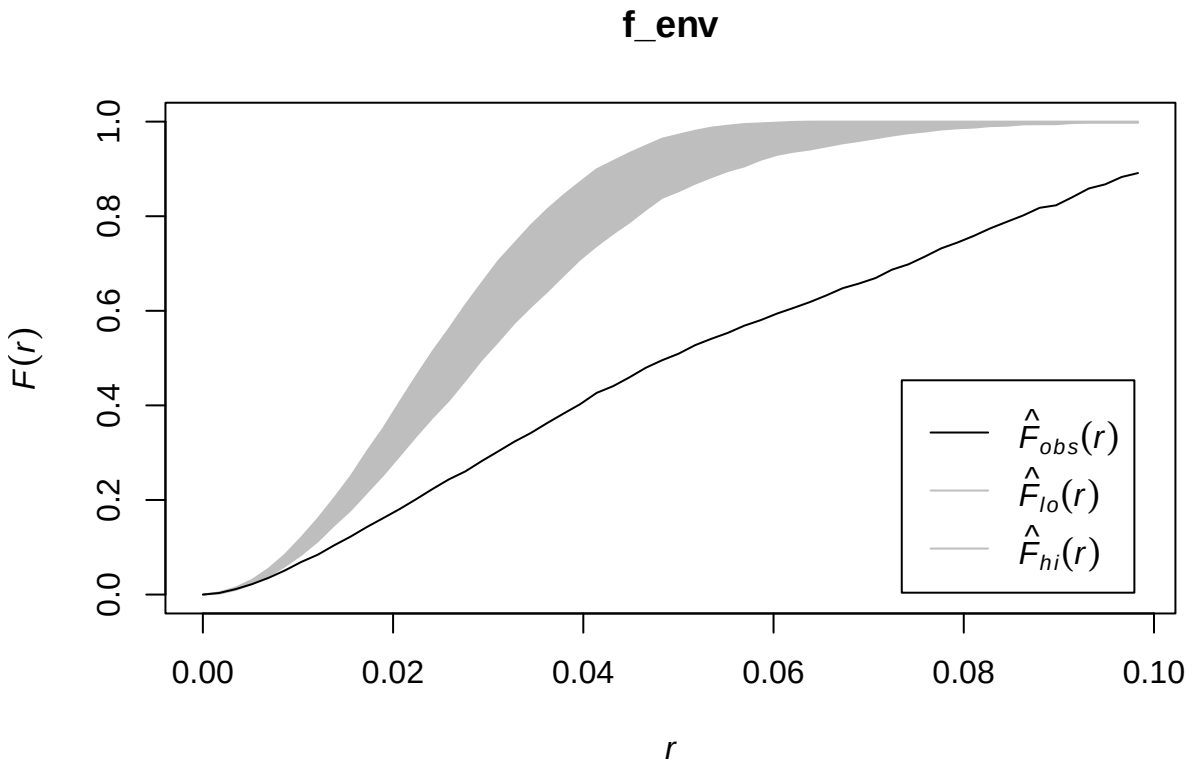
```
library(spatstat)
def <- readRDS("def.rds") # ppp object - locations of individuals with health condition
dist <- readRDS("dist.rds") # covariates d1 and d2 (distances to sources of pollution)

# Exploratory Data Analysis
plot(density(def,bw.scott(def)))
plot(def,add=TRUE, pcf = "+")
```



(a) We want a 95% pointwise confidence interval for the envelope of F with `nsim = 199`. Hence, we set `nrank = 5`.

```
# F function plot of point pattern against radius r
# 95% pointwise confidence interval <=> nsim = 199, nrank = 5
```

```
set.seed(11)
f_env <- envelope(def, Fest, nrank = 5, nsim = 199, verbose = FALSE)
plot(f_env, obs~r) # F_obs lower than envelope generated by CSR, indication of clustering
```



(b) Under the null hypothesis, Complete Spatial Randomness (CRS) is assumed at $r = 0.06$. Since $\hat{F}_{obs}(0.06)$ lies below the envelope, we reject the null hypothesis of CSR; the plot provides an indication of clustering of the point pattern.

(c) The G function is the nearest neighbour distance distribution function (between events) while F is the empty space distribution function (for an arbitrary point to the nearest event).

We assess if a point pattern is CSR by comparing the plot of empirical function $\hat{G}_{obs}(r)$ against the theoretical expectation of G under CSR. Intuitively, if points are close together, then the proportion of points with distance being less than a given threshold is large, and $\hat{G}_{obs}(r)$ will be larger than what we expect under CSR (or the theoretical expectation of G), and thus lie above our envelope of G providing an indication of clustering. Conversely, if $\hat{G}_{obs}(r)$ is larger than the expected value under CSR, it provides an indication of a regular point pattern.

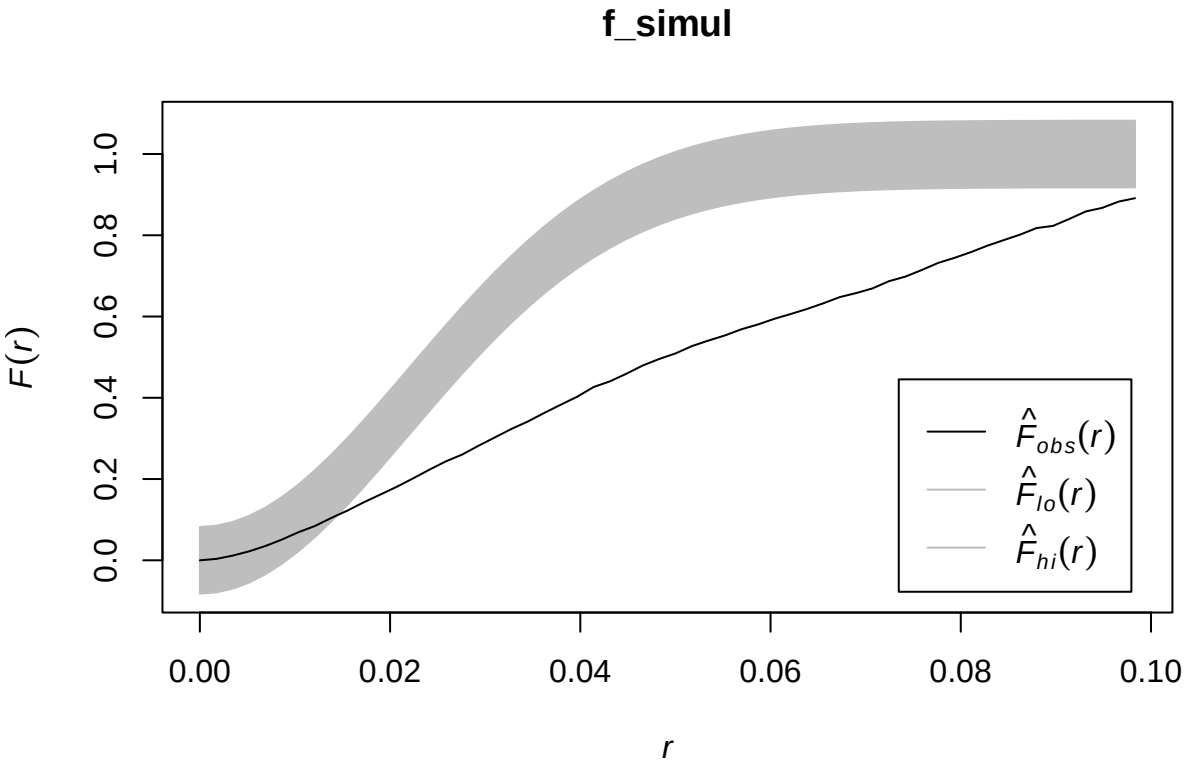
For the F function measuring distance of an arbitrary point to the nearest event, intuitively, if the distance of a random point to the nearest event is large, it provides an indication of large empty spaces, and thus events would be clustered together. With respect to the empirical function, $\hat{F}_{obs}(r)$ which is the proportion of points with distance to the nearest event being less than a specified threshold would be smaller than what we expect under CSR, and thus lie below our envelope of F.

(d) Monte Carlo are appropriate in this setting when some expected cell counts are small, hence using a χ^2 approximation instead may be inaccurate - See plot in part (i).

(e) Suppose instead we wish to find envelopes corresponding to simultaneous confidence bands:

```
# Simultaneous Confidence Bands (set global = TRUE in envelope)
# Monte Carlo test rejects the null hypothesis if graph of the observed function
## lies outside the envelope at any value of r.
```

```
set.seed(11)
f_simul <- envelope(def, Fest, nrank = 10, nsim = 199, global = TRUE ,verbose = FALSE)
# Test has exact significance level, alpha = nrank/(1 + nsim).
## For alpha = 0.05, use nrank = 10.

plot(f_simul, obs~r) # reject null of CSR
```



Simultaneous confidence bands/critical envelopes are such that if $\hat{F}_{obs}$ lies outside the envelope for any value of r, then the test would reject the null hypothesis. The Monte Carlo tests are executed in the following manner:

- For each estimate $\hat{F}_{obs}(r)$, the maximum deviation (or supremum/least upper bound) from the theoretical value of F under the null of CSR is computed. i.e.

$$T_{obs} = \sup_r |\hat{F}_{obs}(r) - F(r)|$$

- This is done for each of the `nsim` simulated datasets. The `nrank`-ed largest value of T_obs is obtained, call this D_{crit}. Then, the upper and lower limits are given by expected value under CSR $\pm D_{crit}$. Test has exact significance level, `alpha = nrank/(1 + nsim)`.

(f) Next, we fit a log-linear model of the intensity using covariates `x`, `y`, `d1` and `d2`.

```
# Fit log-linear model of intensity with following covariates
# x, y, d1 and d2
d1 <- dist$d1; d2 <- dist$d2
model <- ppm(def, ~x+y+d1+d2); model
```

```
## Nonstationary Poisson process
##
## Log intensity:  ~x + y + d1 + d2
##
```

```
## Fitted trend coefficients:
## (Intercept)           x           y          d1          d2
##   8.0315422   3.1992971  -2.2244723  -0.1673729  -7.8216128
##
##               Estimate      S.E.     CI95.lo    CI95.hi Ztest       Zval
## (Intercept)  8.0315422 0.5627183   6.9286346  9.1344497   *** 14.2727584
## x            3.1992971 1.4793325   0.2998588  6.0987355     *  2.1626627
## y           -2.2244723 0.8059319  -3.8040699 -0.6448747    ** -2.7601242
## d1          -0.1673729 1.5135785  -3.1339323  2.7991866       -0.1105809
## d2          -7.8216128 2.1225468 -11.9817280 -3.6614976   *** -3.6850132
```

The following fitted model is obtained:

$$\begin{aligned}\log\lambda(u) &= \beta_0 + \beta_1 x + \beta_2 y + \beta_3 d_1(u) + \beta_4 d_2(u)\\ &= 8.03 + 3.20x - 2.22y - 0.167 d_1(u) - 7.82 d_2(u)\end{aligned}$$

(g) We perform an ANOVA test for model selection at significance level $\alpha = 0.05$.

```
# Model selection with ANOVA (nested models)
fitnull <- ppm(def, ~x+y)
anova(fitnull,model,test = "Chi") # p-val < 0.05
```

```
## Analysis of Deviance Table
##
## Model 1: ~x + y   Poisson
## Model 2: ~x + y + d1 + d2     Poisson
##   Npar Df Deviance  Pr(>Chi)
## 1    3
## 2    5  2    43.57 3.458e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
AIC(model) # smaller AIC (preferred model)
```

```
## [1] -1301.462
```

```
AIC(fitnull)
```

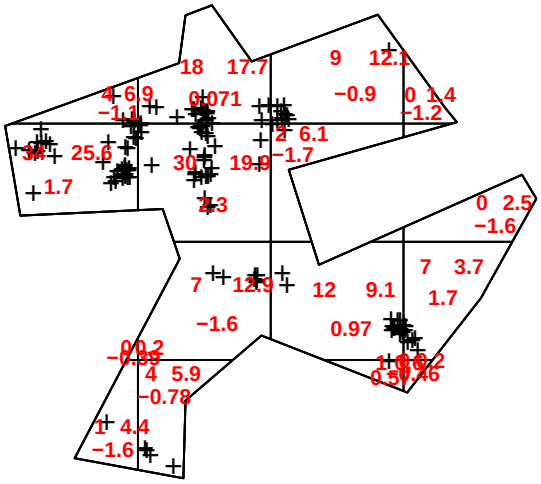
```
## [1] -1261.892
```

The p-value is smaller than $\alpha = 0.05$, indicating a rejection of the linear model in favor of the alternative log-linear model with covariates. The smaller AIC favours the model with covariates.

(h) It is **not true** that if two models A and B are such that the covariates of B are a subset of A, model A will have an AIC smaller than that of B, as the drop in deviance may be less than the penalty of 2 for each additional covariate parameter added to the model.

(i) Monte Carlo Quadrat Test for log-linear model with covariates `x` and `y`

```
# Monte Carlo Quadrat Test to assess Goodness of Fit
set.seed(11)
Q1 <- quadrat.test(fitnull, nx = 4, ny = 4, method = "MonteCarlo")
plot(def, main = "def", pch = "+")
plot(Q1, add = TRUE, col = "red", textargs=list(cex=0.7, font=2))
```



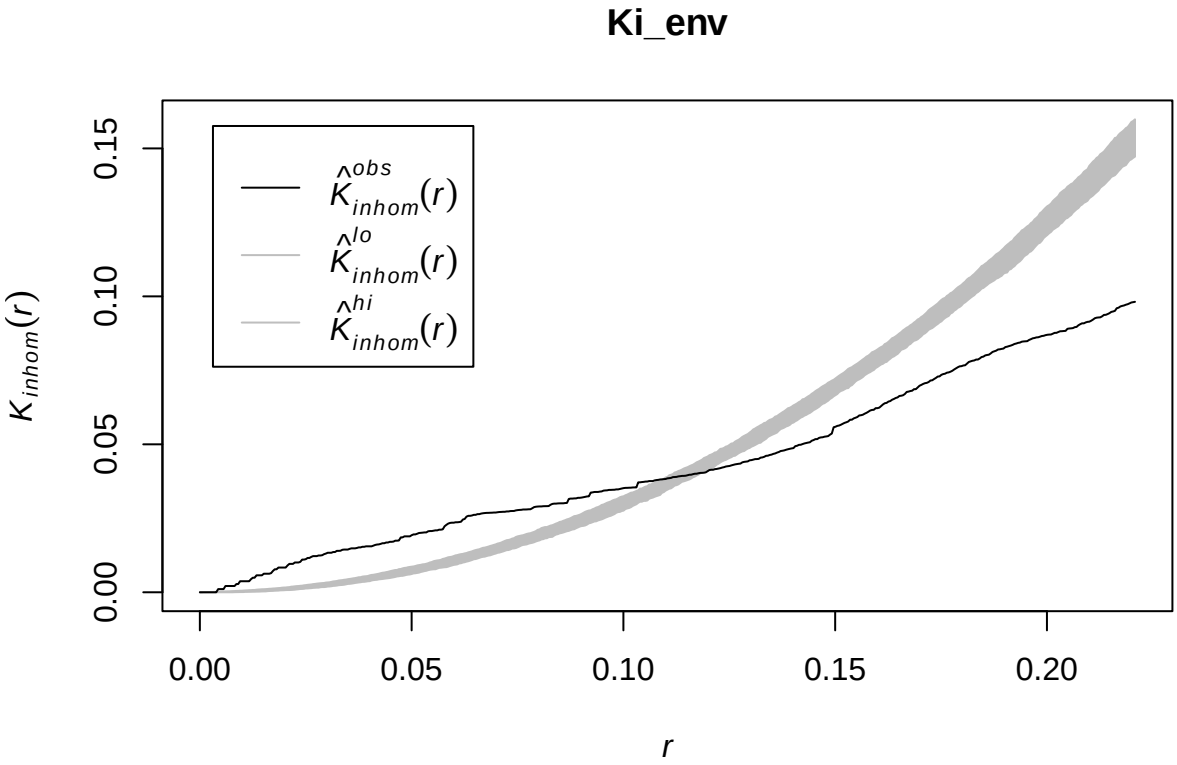

```
Q1 # p-value = 0.087 > 0.05 (do not reject the null of IPP)
```

```
##
##  Conditional Monte Carlo test of fitted Poisson model 'fitnull' using
##  quadrat counts
##  Test statistic: Pearson X2 statistic
##
## data:  data from fitnull
## X2 = 27.229, = NA, p-value = 0.087
## alternative hypothesis: two.sided
##
## Quadrats: 16 tiles (irregular windows)
```

Since p-value $= 0.087 > 0.05$, we do not reject the null of IPP.

(j) Plot of K_{inhom}:

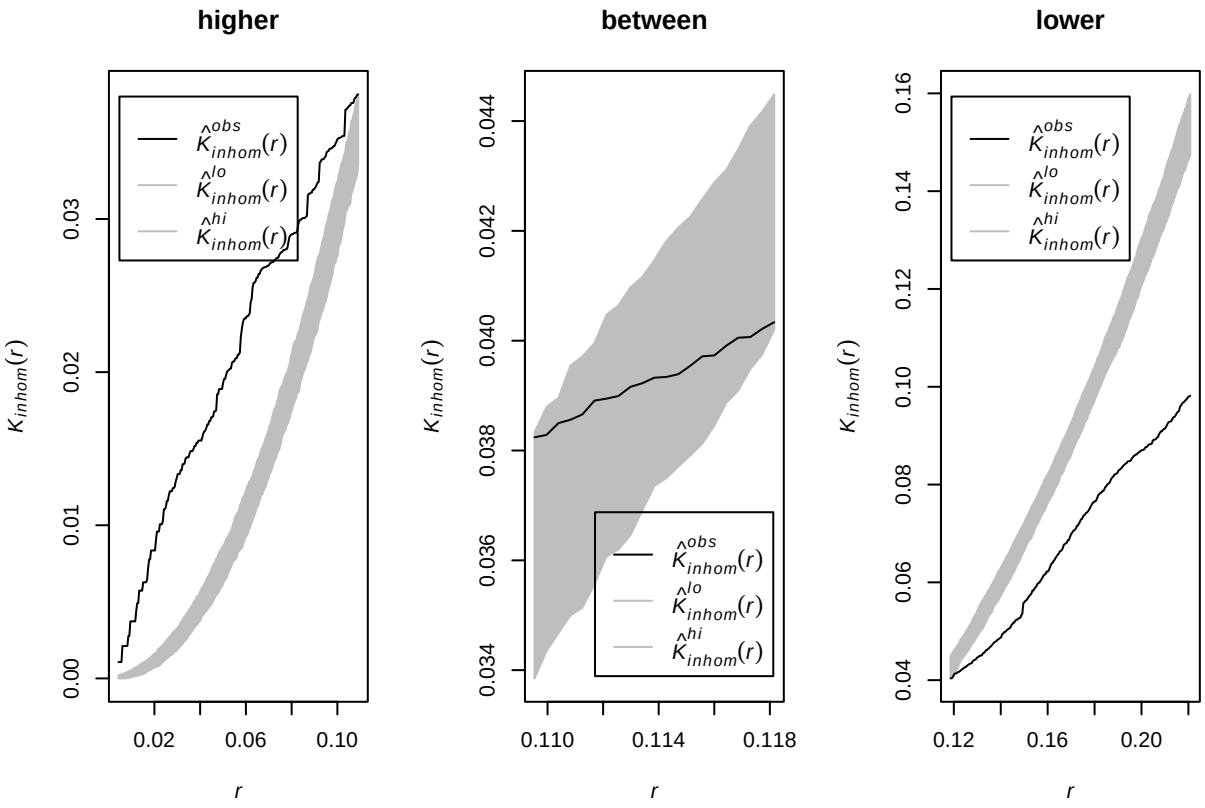
```
# Inhomogeneous K envelope
## 95% pointwise confidence intervals, nsim = 199
set.seed(11)
Ki_env <- envelope(def, Kinhom, nrank = 5, nsim = 199, verbose= FALSE)
plot(Ki_env, obs~r)
```



The Inhomogeneous K function assumes IPP (non-constant intensity but no interactions) under the null. From the plot, we observe that $\hat{K}_{obs}$ lies above the envelope for r approximately less than 0.10, indicating a rejection of IPP and existence of attraction, but $\hat{K}_{obs}$ will also lie below the envelope for r approximately greater than 0.12, indicating that we observe fewer than expected events nearby.

We do not reject the null of IPP only for r approximately between 0.110 and 0.118.

```
par(mfrow=c(1,3))
higher <- Ki_env[Ki_env$obs > Ki_env$hi]; plot(higher, obs~r)
between <- Ki_env[(Ki_env$obs < Ki_env$hi) & (Ki_env$obs > Ki_env$lo)]; plot(between, obs~r)
lower <- Ki_env[Ki_env$obs < Ki_env$lo]; plot(lower, obs~r)
```



Question 3

Consider the log-linear model of the intensity function

$$\log \lambda(u) = \beta_0 + \sum_{j=1}^{p} \beta_j z_j(u),$$

where $z_1(u), \ldots, z_p(u)$ are the covariates at location u and $\beta_0, \ldots, \beta_p$ are the unknown parameters. Given a point pattern $u_1, \ldots, u_n$ the maximum likelihood estimates $\widehat{\beta}_j$ maximizes

$$L(\beta) = \sum_{i=1}^{n} \log \lambda\left(u_i\right) - \int_A \lambda(u) du.$$

(a) The number of events, N in A follow the Poisson distribution with mean $C_\lambda = \int_A \lambda(u) du$, whereby

$$P(N = n) = e^{-C_\lambda} \frac{(C_\lambda)^n}{n!}$$

(b) The location of each event has density $f(u)$ proportional to the intensity $\lambda(u)$, or exactly,

$$f(u) = \frac{1}{C_\lambda} \lambda(u) \quad \text{since} \quad \int_A f(u) du = \frac{1}{C_\lambda} \int_A \lambda(u) du = 1$$

(c) Consider the likelihood function

$$P(N=n)\prod_{i=1}^{n} f(u_i) = \left\{e^{-C_\lambda}\frac{(C_\lambda)^n}{n!}\right\}\prod_{i=1}^{n}\frac{\lambda(u_i)}{C_\lambda}$$
$$= \left\{e^{-C_\lambda}\frac{1}{n!}\right\}\prod_{i=1}^{n}\lambda(u_i) \qquad \text{since } (C_\lambda)^n \text{ cancels out}$$

Maximizing the likelihood is equivalent to maximizing the log-likelihood function (since log is monotone), hence, taking log we have

$$\{-C_\lambda - \log n!\} + \sum_{i=1}^{n}\log\lambda(u_i)$$

where we can drop the factorial term to obtain the expression $L(\beta)$. i.e. finding β maximizing the likelihood is the same as finding β maximizing $L(\beta)$.

Question 4

(a) We observe that the `bei` object belongs to the `ppp` class, or class of point patterns. From studying the structure of its data, we note that it specifies the coordinates `x` and `y`, the number of points `n` and a window for which the point pattern was observed.

```
library(spatstat); library(sp)

class(bei); str(bei) # bei belongs to class of point patterns `ppp`
```

```
## [1] "ppp"
```

```
## List of 5
##  $ window    :List of 4
##   ..$ type  : chr "rectangle"
##   ..$ xrange: num [1:2] 0 1000
##   ..$ yrange: num [1:2] 0 500
##   ..$ units :List of 3
##   .. ..$ singular  : chr "metre"
##   .. ..$ plural    : chr "metres"
##   .. ..$ multiplier: num 1
##   .. ..- attr(*, "class")= chr "unitname"
##   ..- attr(*, "class")= chr "owin"
##  $ n         : int 3604
##  $ x         : num [1:3604] 11.7 998.9 980.1 986.5 944.1 ...
##  $ y         : num [1:3604] 151 430 434 426 415 ...
##  $ markformat: chr "none"
##  - attr(*, "class")= chr "ppp"
```

```
# location of 3604 trees, with the x and y coordinates of the data points
```

For the `bei.extra` object, a call on the object shows that it is a list of two pixel images, `elev` (elevation) and `grad` (gradient), belonging to the image class `im`. Each image object specifies the dimensions of the pixel array, the coordinate values of the pixels and the pixel values.

```
bei.extra # list of 2 pixel images, elev and grad, both of class `im`
```

```
## List of pixel images
##
## elev:
## real-valued pixel image
```

```
## 101 x 201 pixel array (ny, nx)
## enclosing rectangle: [-2.5, 1002.5] x [-2.5, 502.5] metres
##
## grad:
## real-valued pixel image
## 101 x 201 pixel array (ny, nx)
## enclosing rectangle: [-2.5, 1002.5] x [-2.5, 502.5] metres
```

```
class(bei.extra)
```

```
## [1] "imlist"  "solist"  "anylist" "listof"  "list"
```

```
str(bei.extra, max.level = 5)
```

```
## List of 2
##  $ elev:List of 10
##   ..$ v     : num [1:101, 1:201] 121 121 121 121 121 ...
##   ..$ dim   : int [1:2] 101 201
##   ..$ xrange: num [1:2] -2.5 1002.5
##   ..$ yrange: num [1:2] -2.5 502.5
##   ..$ xstep : num 5
##   ..$ ystep : num 5
##   ..$ xcol  : num [1:201] 0 5 10 15 20 25 30 35 40 45 ...
##   ..$ yrow  : num [1:101] 0 5 10 15 20 25 30 35 40 45 ...
##   ..$ type  : chr "real"
##   ..$ units :List of 3
##   .. ..$ singular  : chr "metre"
##   .. ..$ plural    : chr "metres"
##   .. ..$ multiplier: num 1
##   .. ..- attr(*, "class")= chr "unitname"
##   ..- attr(*, "class")= chr "im"
##  $ grad:List of 10
##   ..$ v     : num [1:101, 1:201] 0.252 0.202 0.162 0.17 0.207 ...
##   ..$ dim   : int [1:2] 101 201
##   ..$ xrange: num [1:2] -2.5 1002.5
##   ..$ yrange: num [1:2] -2.5 502.5
##   ..$ xstep : num 5
##   ..$ ystep : num 5
##   ..$ xcol  : num [1:201] 0 5 10 15 20 25 30 35 40 45 ...
##   ..$ yrow  : num [1:101] 0 5 10 15 20 25 30 35 40 45 ...
##   ..$ type  : chr "real"
##   ..$ units :List of 3
##   .. ..$ singular  : chr "metre"
##   .. ..$ plural    : chr "metres"
##   .. ..$ multiplier: num 1
##   .. ..- attr(*, "class")= chr "unitname"
##   ..- attr(*, "class")= chr "im"
##  - attr(*, "class")= chr [1:5] "imlist" "solist" "anylist" "listof" ...
```

(b) Converting `bei` to `SpatialPoints` and `grad` to `SpatialGridDataFrame`

```
# Convert bei to SpatialPoints
bei_sp <- as(bei, "SpatialPoints"); summary(bei_sp)
```

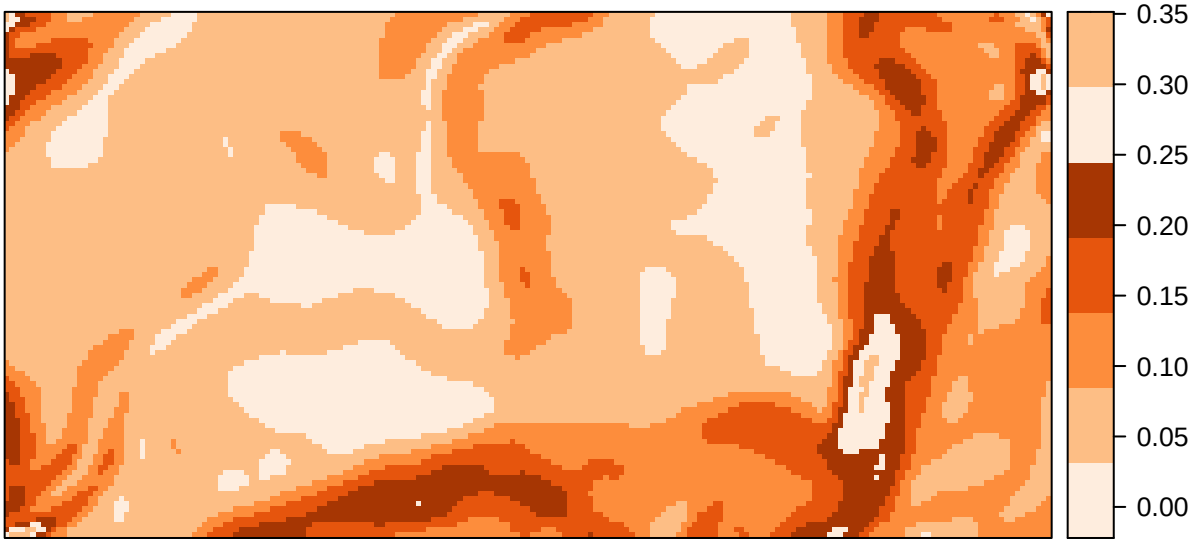
```
## Object of class SpatialPoints
## Coordinates:
##      min  max
```

```
## [1,]   0 1000
## [2,]   0  500
## Is projected: NA
## proj4string : [NA]
## Number of points: 3604
```

```
# Convert gradient (slope) in bei.extra to SpatialGridDataFrame
slope <- bei.extra$grad
slope_sgdf <- as(slope, "SpatialGridDataFrame"); str(slope_sgdf)
```

```
## Formal class 'SpatialGridDataFrame' [package "sp"] with 4 slots
##   ..@ data       :'data.frame':  20301 obs. of  1 variable:
##   .. ..$ v: num [1:20301] 0.311 0.252 0.212 0.194 0.176 ...
##   ..@ grid       :Formal class 'GridTopology' [package "sp"] with 3 slots
##   .. .. ..@ cellcentre.offset: num [1:2] 0 0
##   .. .. ..@ cellsize         : num [1:2] 5 5
##   .. .. ..@ cells.dim        : int [1:2] 201 101
##   ..@ bbox       : num [1:2, 1:2] -2.5 -2.5 1002.5 502.5
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : NULL
##   .. .. ..$ : chr [1:2] "min" "max"
##   ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
##   .. .. ..@ projargs: chr NA
```

```
library(RColorBrewer)
pal <- brewer.pal(n=5, "Oranges")
# Visualize SpatialGridsDataFrame object
spplot(slope_sgdf, cuts = 6, col.regions = pal)
```



(c) Lastly, we create a `SpatialPointsDataFrame` using the tree locations in `bei` as the point pattern, and the slope at the tree locations as the covariates.

```
# Creating SpatialPointsDataFrame with bei
# Overlay points onto grid, obtain number of points in each cell of the grid
idx = over(bei_sp,slope_sgdf)
bei_spdf <- SpatialPointsDataFrame(bei_sp, idx)
head(bei_spdf); str(bei_spdf)
```

```
## class       : SpatialPointsDataFrame
```

```
## features    : 6
## extent      : 11.7, 998.9, 151.1, 433.5  (xmin, xmax, ymin, ymax)
## crs         : NA
## variables   : 1
## names       :         v
## min values  : 0.0852658
## max values  : 0.2846527
```

```
## Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
##   ..@ data       :'data.frame':  3604 obs. of  1 variable:
##   .. ..$ v: num [1:3604] 0.1162 0.2847 0.2228 0.2417 0.0853 ...
##   ..@ coords.nrs : num(0)
##   ..@ coords     : num [1:3604, 1:2] 11.7 998.9 980.1 986.5 944.1 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : NULL
##   .. .. ..$ : chr [1:2] "mx" "my"
##   ..@ bbox       : num [1:2, 1:2] 0 0 1000 500
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : NULL
##   .. .. ..$ : chr [1:2] "min" "max"
##   ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
##   .. .. ..@ projargs: chr NA
```