GET: Hotspot detection on a linear network

Mari Myllymäki
Natural Resources Institute Finland (Luke)

Tomáš Mrkvička
University of South Bohemia

Mikko Kuronen
Natural Resources Institute Finland (Luke)

Abstract

This vignette shows how the methodology proposed by Mrkvička, Kraft, Blažek, and Myllymäki (2023) for detecting hotspots on a linear network can be performed using the R package **GET** (Myllymäki and Mrkvička 2024).

Keywords: false discovery rate, hotspot, linear network, Monte Carlo test, road accidents, R, spatial point pattern.

Loading required packages and setting a **ggplot2** theme for images.

```
library("GET")
library("spatstat")
library("spatstat.linnet")
library("spatstat.Knet")
library("ggplot2")
library("parallel")
theme_set(theme_bw(base_size = 9))
```

Defining functions for the later use below:

```
# Fitting the inhomogeneous Poisson process
# @param PP Input, a point pattern.
# @param formul An R formula to estimate the first order model.
#  This formula can contain objects of full size.
# @param data Data from where the formula takes objects.
#  Must be acceptable by the function lppm of spatstat.linnet.
lpoisppm <- function(PP, formul, data) {
  M <- lppm(formul, data = data)
  EIP <- predict(M)
  maxr <- sqrt(area(PP))/10
  r <- seq(0, maxr, length.out=100) # Up to 5 km
```

```
  PP_K <- Knetinhom(PP, lambda = update(M, PP), r=r)
  list(EIP=EIP, firstordermodel=M, secondorder=PP_K)
}
# Simulating the Matern cluster process on a linear network
# @param Centers The (x,y)-coordinates of parent points
# @param R The R parameter of the Matern cluster process
# @param alpha The alpha parameter of the Matern cluster process
# @param LL The linear network on which the point pattern should be simulated.
# @check_vol Logical. TRUE for checking if the ball producing the cluster
#  has any intersection with linear network.
rMatClustlpp <- function(Centers, R, alpha, LL, check_vol=FALSE) {
  e <- as.matrix(LL$lines$ends)
  e2 <- list()
  for(i in 1:nrow(e)) {
    e2[[i]] <- sf::st_linestring(matrix(e[i,], 2,2, byrow=TRUE))
  }
  LN <- sf::st_sfc(e2)
  X <- array(0,0)
  Y <- array(0,0)
  cs <- sf::st_sfc(apply(cbind(Centers$data$x, Centers$data$y), 1,
                         sf::st_point, simplify=FALSE))
  bufs <- sf::st_buffer(cs, R, nQuadSegs = 8)
  LN1s <- sf::st_intersection(bufs, sf::st_union(LN))
  stopifnot(length(LN1s)==length(Centers$data$x))
  for(p in 1:length(Centers$data$x)) {
    LN1 <- LN1s[p]
    LN1 <- unlist(LN1) # LN1 contains line segments with exactly 4 numbers each
    LL1 <- psp(LN1[1:4 == 1], LN1[1:4 == 3],
               LN1[1:4 == 2], LN1[1:4 == 4],
               window=Window(LL), check=FALSE)
    vol <- sum(lengths_psp(LL1))
    if(check_vol) {
      BBCOutD_ss <- disc(radius=R, centre=c(Centers$data$x[p],
                                            Centers$data$y[p]),
                         npoly = 32)
      vol2 <- volume(LL[BBCOutD_ss])
      if(abs(vol-vol2) > 1e-4) stop("vol")
    }
    Xp <- rpoisppOnLines(alpha/vol, L=LL1)
    X <- append(X, as.numeric(Xp$x))
    Y <- append(Y, as.numeric(Xp$y))
  }
  lpp(cbind(X,Y), LL)
}
# Fitting the Matern cluster pp - parallel version
lmcppm_par <- function(PP, formul, valpha, vR, data,
                       nsim = 10, ncores=NULL) {
```

```
lmcppm_par_inner <- function(fakeArg, valpha, EIP, PP, vR, M, r) {
  # Centers from a Poisson process
  Centers <- rpoislpp(EIP/valpha, L=PP[['domain']])
  XX <- rMatClustlpp(Centers, vR, valpha, PP[['domain']])
  return(Knetinhom(XX, lambda = update(M, XX), r=r)$est)
}

if(!is.vector(valpha)) { stop("valpha is not a vector") }
if(!is.vector(vR)) { stop("vR is not a vector") }
M <- lppm(formul, data = data)
EIP <- predict(M)
maxr <- sqrt(area(PPfull))/10
r <- seq(0, maxr, length.out=100) # Up to 5 km
PP_K <- Knetinhom(PP, lambda = update(M, PP), r=r)
Contrast <- array(0, c(length(valpha), length(vR)))
Karray <- array(0, c(length(valpha), length(vR),length(r)))

fakeArgs <- rep(1, times=nsim)

# if the number of cores has not been set by ncores function parameter,
# try to detect it on the host
if ( is.null(ncores) ) {
  # detect number of cores
  ncores <- detectCores()
  ncores <- max(1L, detectCores()-1, na.rm = TRUE)
}

# create cluster with ncores nodes
cl <- makeCluster(ncores, type = "SOCK")
clusterEvalQ(cl, library("spatstat"))
clusterEvalQ(cl, library("spatstat.Knet"))
clusterExport(cl=cl, c("rMatClustlpp", "PPfull"))

for(i in 1:length(valpha)) {
  for(j in 1:length(vR)) {
    # Compute the average K of 10 simulation from the model
    KMC_sim_results <-clusterApplyLB(cl, fakeArgs, lmcppm_par_inner,
                                     valpha[i], EIP, PP, vR, M, r)

    # summarize result of all simulations into one array
    KMC <- array(0,length(r))
    for(s in 1:length(KMC_sim_results)) {
      KMC <- KMC + KMC_sim_results[[s]]
    }

    # Compute the difference between estimated and average K
    Contrast[i,j] <- sqrt(sum((PP_K$est-KMC/nsim)^2))
```

```
      Karray[i,j,] <- KMC/nsim
    }
  }

  # stop the cluster
  stopCluster(cl)

  # Finding the minimum value of the contrast
  id <- which(Contrast == min(Contrast), arr.ind = TRUE)
  alpha <- valpha[id[,1]]
  R <- vR[id[,2]]
  EIP <- EIP/alpha

  list(EIP=EIP, alpha=alpha, R=R, firstordermodel=M,
       secondorder=PP_K, MCsecondorder=Karray[id[,1], id[,2],])
}
# Hot spots detection from Poisson or Matern Cluster point pattern
# - parallel version
# @param PP The point pattern living in a network.
# @param formul A formula for the intensity
# @param clusterparam NULL if the model to be used is the Poisson point process.
# A named list with components 'alpha' and 'R' for the Matern cluster process.
# @param sigma To be passed to density.lpp.
# @param nsim Number of simulations to be performed.
# @param ncores Number of cores used for computations.
hotspots_par <- function(PP, formul, clusterparam, data,
                         sigma=250, nsim = 10000,
                         ncores=NULL) {
  if(is.null(clusterparam)) {
    model <- "Pois"
    hotspots_par_inner <- function(fakeArg, clusterparam, EIP, PP, sigma) {
      simss <- rpoislpp(EIP, L=PP[['domain']])
      return(density.lpp(simss, sigma = sigma, distance="euclidean"))
    }
  }
  else {
    model <- "MC"
    if(!("alpha" %in% names(clusterparam) & "R" %in% names(clusterparam)))
      stop("clusterparam should be a named list with components ",
      "R and alpha (or NULL for Poisson process).")
    hotspots_par_inner <- function(fakeArg, clusterparam, EIP, PP, sigma) {
      Centers <- rpoislpp(EIP, L=PP[['domain']])
      simss <- rMatClustlpp(Centers, clusterparam[['R']],
                            clusterparam[['alpha']], PP[['domain']])
      return(density.lpp(simss, sigma = sigma, distance="euclidean"))
    }
  }
```

```
  M <- lppm(formul, data = data)
  if(model == "Pois") EIP <- predict(M)
  else EIP <- predict(M)/clusterparam[['alpha']]
  densi <- density.lpp(PP, sigma = sigma, distance="euclidean")
  sims.densi <- vector(mode = "list", length = nsim)
  fakeArgs <- rep(1, times=nsim)

  # if the number of cores has not been set by ncores function parameter,
  # try to detect it on the host
  if(is.null(ncores)) {
    # detect number of cores
    ncores <- detectCores()
    ncores <- max(1L, detectCores()-1, na.rm = TRUE)
  }

  # create cluster with ncores nodes
  cl <- makeCluster(ncores, type = "SOCK")
  clusterEvalQ(cl, library("spatstat"))
  clusterExport(cl=cl,  c("rMatClustlpp"))

  sims.densi <-clusterApplyLB(cl, fakeArgs, hotspots_par_inner,
                              clusterparam, EIP, PP, sigma)

  # stop the cluster
  stopCluster(cl)

  yx <- expand.grid(densi$yrow, densi$xcol)

  noNA_id <- which(!is.na(densi$v))
  noNA_idsim <- which(!is.na(sims.densi[[1]]$v))
  noNA_id <- intersect(noNA_id, noNA_idsim)
  #max(densi[noNA_id]); summary(sapply(sims.densi, FUN=max))

  cset <- create_curve_set(list(
    r=data.frame(x=yx[,2], y=yx[,1],
                 width=densi$xstep, height=densi$ystep)[noNA_id,],
    obs=as.vector(densi$v)[noNA_id],
    sim_m=sapply(sims.densi, FUN=function(x){ as.vector(x$v)[noNA_id] },
                 simplify=TRUE)))
  res <- fdr_envelope(cset, alternative = "greater")
  res
}
```

1. Data

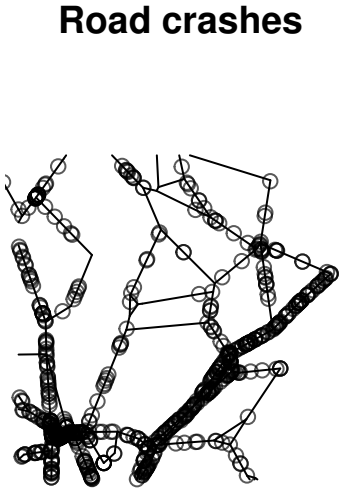
Mrkvička *et al.* (2023) worked with the database of road crashes reported to the Police in the Czech Republic from 1 January 2016 to 31 December 2020. Here we show the methodology for a subpattern of this full data set. The **GET** package provides a data object `roadcrash` that has 7700 road crashes lying on a linear network with 269 vertices and 354 lines. Because the computations of inhomogeneous K-function and density are rather computational, for illustration of the methodology below, we use a subset of the `roadcrash` data.

Load the road crash data from **GET**:

```
data("roadcrash")
win <- owin(xrange = roadcrash$xrange,
            yrange = roadcrash$yrange)
X <- ppp(x = roadcrash$x, y = roadcrash$y, window = win)
Vertices.pp <- ppp(x = roadcrash$Vertices.x,
                   y = roadcrash$Vertices.y,
                   window=win)
L <- linnet(vertices=Vertices.pp,
            edges = roadcrash$Edges)
PPfull <- lpp(X, L)
roadcrash$Traffic <- im(roadcrash$Traffic,
         xrange = roadcrash$xrange,
         yrange = roadcrash$yrange)
roadcrash$ForestDensity <- im(roadcrash$ForestDensity,
         xrange = roadcrash$xrange,
         yrange = roadcrash$yrange)
roadcrash$BuildingDensity <- im(roadcrash$BuildingDensity,
         xrange = roadcrash$xrange,
         yrange = roadcrash$yrange)
```

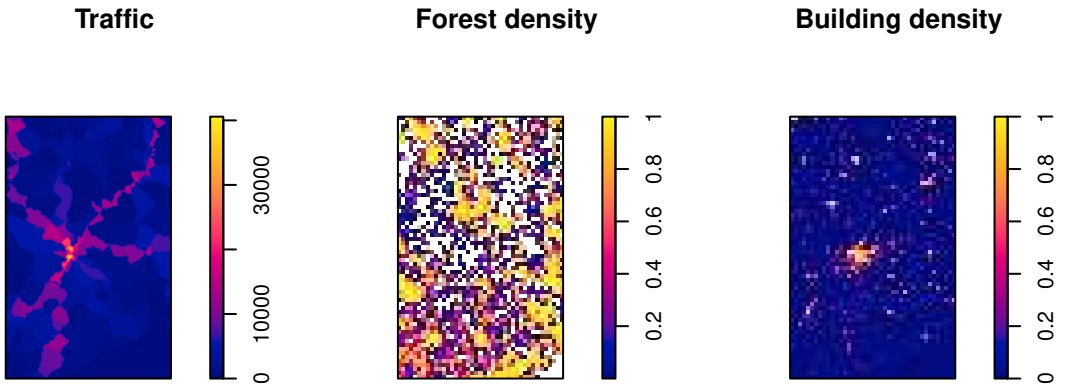
Define a subpattern:

```
subwin <- owin(c(-760000, -740000), c(-1160000, -1140000))
PP <- PPfull[, subwin]
plot(PP, main="Road crashes")
```

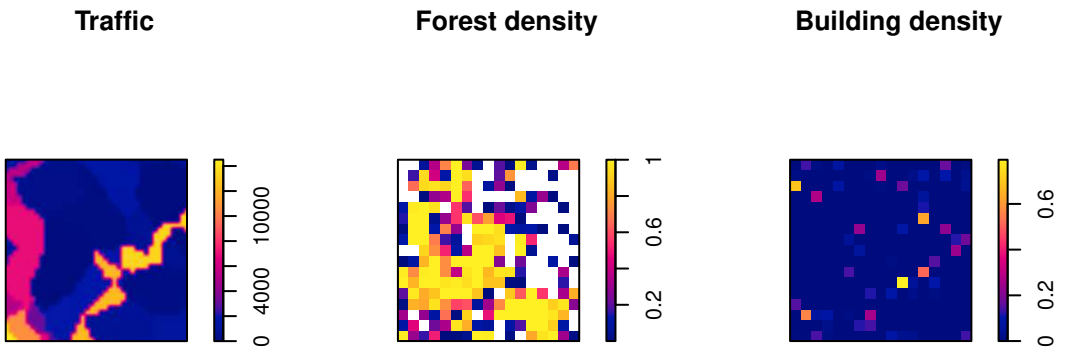


Mrkvička *et al.* (2023) had a total of 9 spatially defined covariates. In our example here and available in `roadcrash` in **GET** are three covariates, namely average traffic volume (number of vehicles per 24 hours), forest density and building density in the cell.

```
par(mfrow=c(1,3))
plot(roadcrash$Traffic, main="Traffic")
plot(roadcrash$ForestDensity, main="Forest density")
plot(roadcrash$BuildingDensity, main="Building density")
```



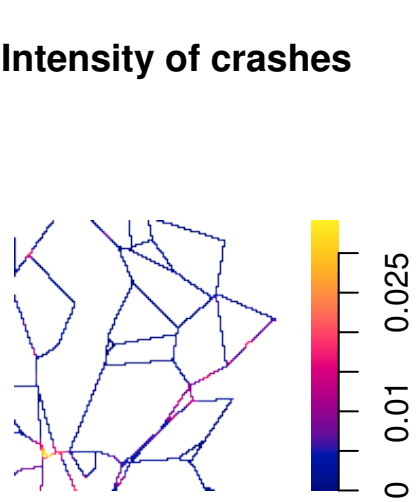
```
par(mfrow=c(1,3))
plot(roadcrash$Traffic[subwin], main="Traffic")
plot(roadcrash$ForestDensity[subwin], main="Forest density")
plot(roadcrash$BuildingDensity[subwin], main="Building density")
```



2. Non-parametric intensity estimate

A non-parametric density estimate of the point pattern on a linear network can be obtained using the function `density.lpp()` of the **spatstat** package.

```
densi <- density.lpp(PP, sigma = 250, distance="euclidean")
plot(densi, main="Intensity of crashes")
```



3. Fitting the inhomogeneous Poisson process

The simplest point process model for road crashes is the (inhomogeneous) Poisson process with intensity

$$\rho_\beta(u) = \kappa \exp(z(u)\beta^T), \ u \in L, \tag{1}$$

where L is a linear network, $z = (z_1, \ldots, z_k)$ is a vector of covariates and $\beta = (\beta_1, \ldots, \beta_k)$ is a regression parameter. This process can be fitted using the **spatstat** package. We fit the model using the full `roadcrash` data.

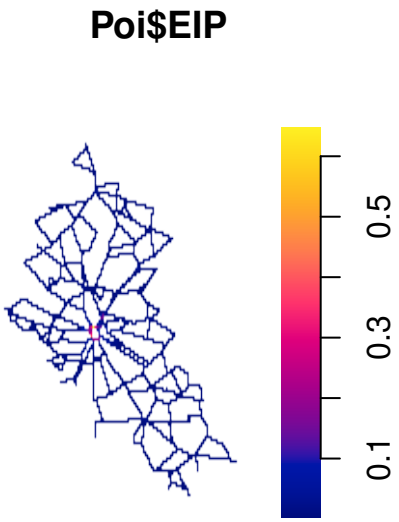
The function lpoisppm (see above) both fits the model for the intensity as well as provides predicted point process intensity and the inhomogeneous K-function estimated from a given point pattern (here the sub-point pattern PP) using the estimated intensity surface.

```
myformula <- PPfull ~ Traffic + ForestDensity + BuildingDensity
system.time(
  Poi <- lpoisppm(PP, myformula, roadcrash)
)
```

```
  user  system elapsed
  0.15    0.01    0.17
```

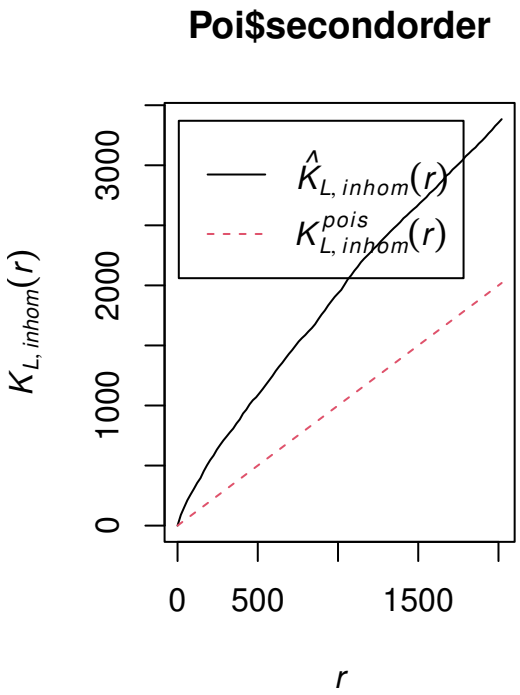
The predicted point process intensity can then be plotted:

```
plot(Poi$EIP)
```



The inhomogeneous K-function with theoretical Poisson K function can be plotted, too:

```
plot(Poi$secondorder)
```



Here the inhomogeneous K-function estimated from the data lies above the theoretical line for the Poisson process and suggests clustering of points.

4. Fitting the Matern cluster process on a linear network

Mrkvička *et al.* (2023) considered instead of the Poisson process the Matern cluster point process with inhomogeneous cluster centers. This process is more suitable for clustered data. It can be estimated in two steps according to its construction following Mrkvička, Muška, and Kubečka (2014). In first step, the first order intensity function is estimated through Poisson likelihood. This was done above, i.e., the object `EIP` contains the estimated intensity. In second step, the second order interaction parameters α (mean number of points in a cluster) and R (cluster radius) are estimated through minimum contrast method. Unfortunately, working with cluster processes on linear networks is rather consuming and therefore they are currently not covered by the **spatstat** package. Thus, we have used the inhomogeneous K-function and the minimum contrast and grid search methods to find the optimal parameters. We implemented functions for simulating the Matern cluster process on a linear network `LL` with pre-specified centers (function `rMatClustlpp`) and for fitting the Matern cluster process on a point pattern on a linear network (function `lmcppm_par`).

In the procedure, we estimate the parameters R and α of the Matern cluster process using the inhomogeneous K-function (with the estimated Poisson process intensity). We consider a range of possible values of the parameters R and α. For each value of R and α, we compute the difference of the observed K-function from the "theoretical" K-function of the model, computed from the average of `nsim` (here 5) simulation from the model. (Note that this

computation takes some time.) Simulations are used, because the theoretical K-function is not known for the Matern cluster process on the linear network. We then find out which of these possible values of parameters R and α lead to the smallest difference between the observed and "theoretical" K-functions.

Remark here that the first order structure is estimated below from the full pattern `PPfull`, whereas the second order structure is estimated from the pattern `PP` observed in the subwindow. The second order structure has a limited range; therefore, estimating it only from the subpattern is useful for time saving.

```
set.seed(2023)
```

```
valpha <- seq(5, 30, by=5)
vR <- seq(250, 2500, by=500)
myformula <- PPfull ~ Traffic + ForestDensity + BuildingDensity
system.time( # Took about 1,2 minutes with 4 cores on a laptop
  MatCl <- lmcppm_par(PP, myformula, valpha, vR, roadcrash, ncores = 4)
)
```

```
   user  system elapsed
  10.56    7.73   69.34
```

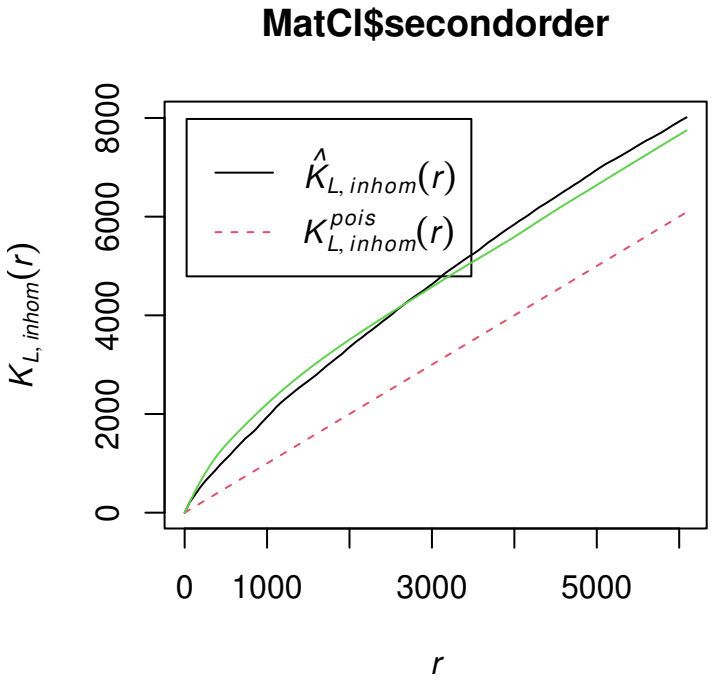
These results can be viewed:

```
# The observed K, and theoretical Poisson line
plot(MatCl$secondorder)
# The Matern Cluster process K from nsim (here 5) simulations
#  with chosen values of alpha and R
lines(x=MatCl$secondorder$r, y=MatCl$MCsecondorder, col=3)
# Chosen values
MatCl$alpha
```

```
[1] 25
```

```
MatCl$R
```

```
[1] 1250
```



5. False discovery rate envelopes

To find the hotspots of road crashes that are not explained by the covariates, we first generate `nsim` simulations from the fitted Matern cluster process and estimate the intensity for each of the simulated patterns. We note that we estimate the intensity here similarly as above for the observed pattern. This computation takes a bit of time (using 7 cores on a laptop this took 18 minutes).

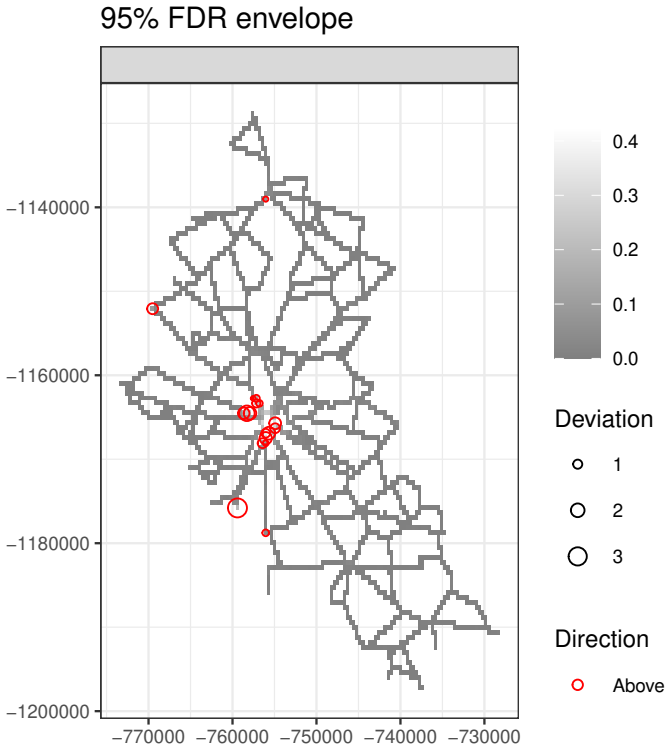
```
nsim <- 10000
system.time(
  res <- hotspots_par(PPfull, myformula,
                      clusterparam=list(alpha=MatCl$alpha, R=MatCl$R),
                      data = roadcrash, sigma=250, nsim=nsim, ncores=4)
)
```

```
   user  system elapsed
352.59  260.26 1517.58
```

```
#save(res, file="roadcrash_res.Rdata")
load(file="roadcrash_res.Rdata")
```

Then the FDR envelope (Mrkvička and Myllymäki 2023) can be computed using the function `fdr_envelope()` of the **GET** package. We set the alternative to `"greater"`, because we are only interested in locations where the intensity is higher than expected.

```
plot(res) + scale_radius(range = 0.5 * c(1, 6))
```



The size of the cluster is indicated by the circles. The circle radius is proportional to the size of the deviation of the observed intensity from the upper bound of the FDR envelope divided by the difference of the upper FDR envelope and the centre of the envelope. Thus, the size is a measure of relative exceedance.

References

Mrkvička T, Kraft S, Blažek V, Myllymäki M (2023). "Hotspot Detection on a Linear Network in the Presence of Covariates: A Case Study on Road Crash Data." doi:http://dx.doi.org/10.2139/ssrn.4627591.

Mrkvička T, Muška M, Kubečka J (2014). "Two Step Estimation for Neyman-Scott Point Process with Inhomogeneous Cluster Centers." *Statistics and Computing*, **24**(1), 91–100. doi:10.1007/s11222-012-9355-3.

Mrkvička T, Myllymäki M (2023). "False Discovery Rate Envelopes." *Statistics and Computing*, **33**, 109. doi:10.1007/s11222-023-10275-7.

Myllymäki M, Mrkvička T (2024). "**GET**: Global Envelopes in R." *Journal of Statistical Software*, **111**(3), 1–40. doi:10.18637/jss.v111.i03.

Affiliation:

Mari Myllymäki
Natural Resources Institute Finland (Luke)
Latokartanonkaari 9
FI-00790 Helsinki, Finland
E-mail: mari.myllymaki@luke.fi
URL: https://www.luke.fi/en/experts/mari-myllymaki/
and
Tomáš Mrkvička
Faculty of Agriculture and Technology
University of South Bohemia,
Studentská 1668
37005 České Budějovice, Czech Republic
E-mail: mrkvicka.toma@gmail.com
URL: http://home.ef.jcu.cz/~mrkvicka/