Ville Väänänen / 63527M
ville.vaananen@aalto.fi

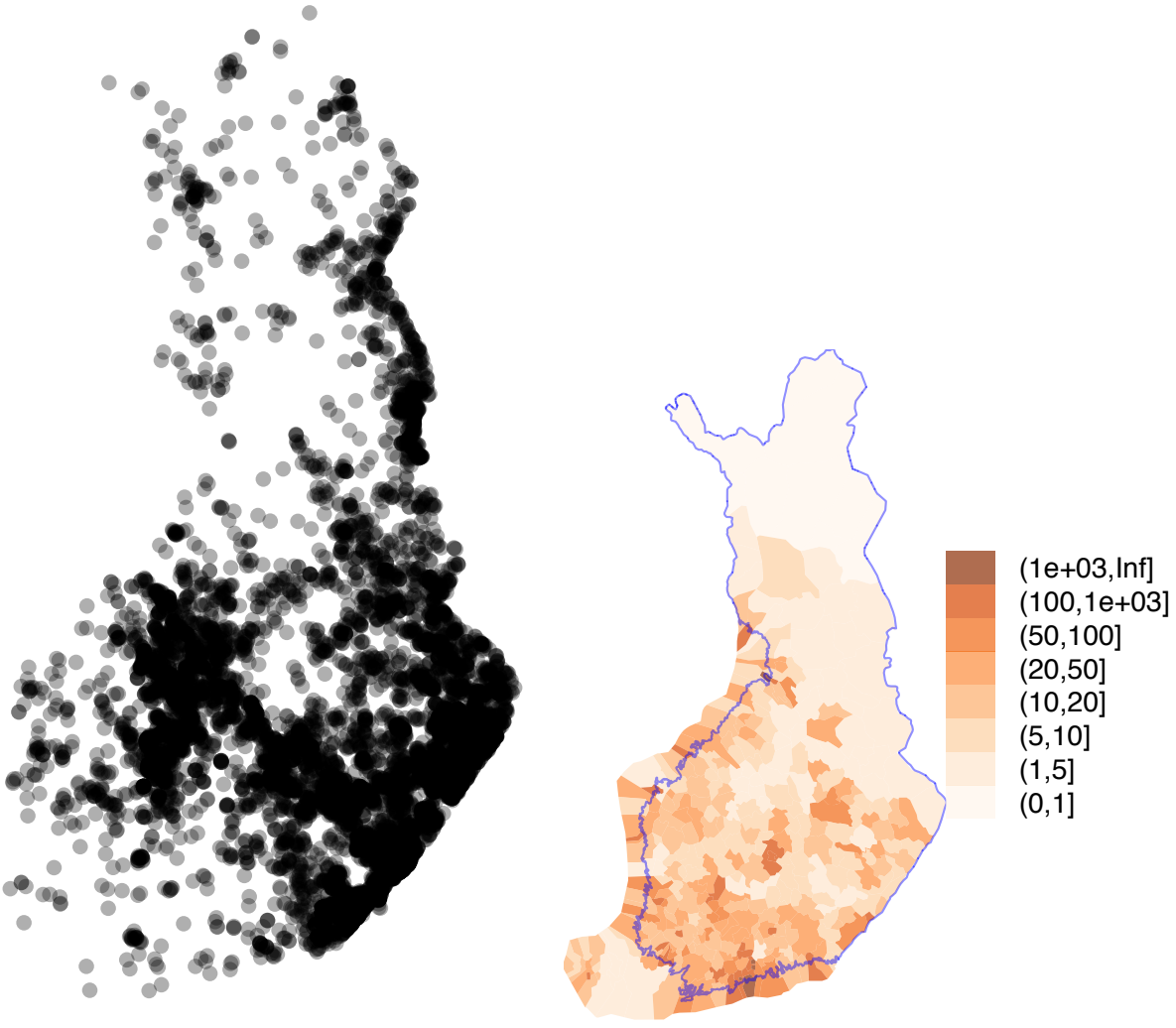
Bear observations in Finland in 2010

S-114.4202 Special Course in Computational Engineering II

March 30, 2012

1 Data description

The bear observations in the year 2010 are plotted in figure 1a and the population density of the municipalities (the number of people in the municipality per its land area) is plotted in figure 1b. As can be seen, there is some clustering which has to be associated with increased probability of reporting due population density. Thus an interesting problem is, how should we take into account the probability for an animal to get observed in a given area? Is it directly proportional to population density? Is there some other factors associated such as the amount of hunters in the area?



(a) Bear observations in 2010

(b) Population density

Figure 1: Bear observations in 2010 and the population density of Finland

2 Methods

Poisson process

Poisson processes are considered the *null* model in spatial point process statistics, since they exhibit *complete spatial randomness* (CSR). This means, that given the intensity field $\lambda(\mathbf{s})$, the points of the process in two disjoint areas are indepedent.

Throughout the report, we will denote vectors and vector valued functions with bold lower-case symbols and matrices with bold upper case symbols. We will denote a *spatial point process*, defined in some bounded region $\Omega \subset \mathbb{R}^2$, with $\mathcal{X}$. A spatial point process can be considered a *random finite subset* of, in this case, Ω [2]. It is usually observed in some bounded observation window $W \subset \Omega$. A realisation of a point process is called a *point pattern*

$$\mathcal{X}_W = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}, \tag{1}$$

where the points $\mathbf{x}_k \in W$. The distribution of $\mathcal{X}$ can be defined by specifying the distribution of the number of points $N(\Omega)$ (which we will also denote $N(\mathcal{X})$) and then the joint distribution of the n points given $N(\Omega) = n$. Equivalently one can specify the distribution of $N(D)$, that is is the number of points of $\mathcal{X}$ in any subset $D \subset \Omega$.

For the Poisson process we have

$$P\big(N(D) = k \mid \lambda\big) = \frac{\Lambda(D)^k}{k!} e^{-\Lambda(D)}, \tag{2}$$

where $\Lambda(D) = \displaystyle\int_D \lambda(\mathbf{s}) \ \mathrm{d}\mathbf{s}$. Then, given the number of points, the points are i.i.d so that the density is proportional to the intensity function. The joint distribution of the number of points and the points, i.e. the likelihood, is then given by

$$\begin{aligned} P(\mathcal{X} \mid \lambda) &= P\big(N(\mathcal{X}) = n \mid \lambda\big) \prod_{i=1}^{n} \frac{\lambda(\mathbf{x}_i)}{\Lambda(\Omega)} \\ &= \frac{\Lambda(\Omega)^n}{n!} e^{-\Lambda(\Omega)} \prod_{i=1}^{n} \frac{\lambda(\mathbf{x}_i)}{\Lambda(\Omega)} \\ &\propto \exp\left(-\int_\Omega \lambda(\mathbf{s}) \ \mathrm{d}\mathbf{s}\right) \prod_{i=1}^{n} \lambda(\mathbf{x}_i) \end{aligned} \tag{3}$$

Cox processes

Cox processes are a flexible family of point processes described by a *random* intensity field. Thus they are also called doubly stochastic. Given the random

intensity field, a Cox process is a Poisson process. In a log-Gaussian Cox process, the random latent intensity field has the form

$$\lambda(\mathbf{s}) = e^{Z(\mathbf{s})}, \ \mathbf{s} \in \mathbb{R}^2 \tag{4}$$

where Z is a *Gaussian random field* (also called a Gaussian process, $\mathcal{GP}$) defined by a mean function $\mu(\mathbf{s})$ and a covariance function $c(\mathbf{s}, \mathbf{s}')$ [2]. In this report a linear model will be used for the mean function, i.e

$$\mu(\mathbf{s}) = \boldsymbol{\beta}^T \mathbf{b}(\mathbf{s}), \tag{5}$$

where typically $b_1(\mathbf{s}) = 1$, so that β_1 is the intercept. With this definition, possible covariate data can be included directly in $\mathbf{z}$.

Latent Gaussian models

Our aim is to be able to use a computationally efficient Bayesian procedure for inference in latent Gaussian models called INLA (or integrated nested Laplace approximations) [3]. Latent Gaussian models are a large set of flexible models, where given a latent (unobserved) Gaussian random field the observations $\mathbf{y}$ are independent and identically distributed with an exponential family distribution [3]. More precisely we have

$$Z(\mathbf{s})|\boldsymbol{\theta} \sim \mathcal{GP}(\mu(\mathbf{s}, \boldsymbol{\theta}), Q(\mathbf{s}, \mathbf{s}', \boldsymbol{\theta})) \tag{6}$$

$$\mathbf{y}|Z(\mathbf{s}), \boldsymbol{\theta} \sim \prod_{k=1}^{n} p\big(y_k \mid Z(\mathbf{s}), \boldsymbol{\theta}\big) \tag{7}$$

In general, it is the posterior distribution of $Z(\mathbf{s})$ and also of the parameters θ that we are interested in. We shall not venture deeper into how these are approximated in the INLA framework, but it suffices to be able to evaluate the likelihood, given a realisation of the field and the parameters.

Likelihood approximation

LGCP's can be thought of as latent Gaussian models and their likelihood is of the form (3), with the random intensity field (4). Commonly one places a regular grid over the observation window, and counts the number of points y_i in the cell C_i with area $|C_i|$. Then if we assume that the field is constant in a cell, we have

$$y_i \sim Poisson(|C_i|\, e^{Z(c_i)}), \tag{8}$$

and

$$P(\mathbf{y} \mid \mathbf{z}) \propto \exp\left(-\sum_{i=1}^{m} |C_i|\, e^{z_i}\right) \prod_{i=1}^{m} \left(|C_i|\, e^{z_i}\right)^{y_i}, \tag{9}$$

where c_i is chosen to be some representative point of the cell, where the field is evaluated. As the field is evaluated in a discrete set of points c_i, the vector $\mathbf{z} = \big[Z(c_1), \ldots, Z(c_m)\big]$ is jointly Gaussian and the problem is reduced into a rather standard one [4].

From the point of view of the exact likelihood (3), (9) has been obtained by approximating the integral in (3) with a Riemann sum over the regular grid and binning the data to the locations c_i. Clearly this approximation turned the instractable likelihood (3) into a product of independent Poisson densities. However, as argued in [4], it is far from optimal to use the grid both for approximating the intensity field *and* the points. A nicer alternative would be to form a finite dimensional *continous* approximation to the field, that could be evaluated at the exact point locations and integrated on a *mesh* that is only used for that purpose.

This can be achieved by forming a piecewise linear basis function approximation to $Z(\mathbf{s})$

$$\hat{Z}(\mathbf{s}) = \sum_{j=1}^{m} z_j \phi_j(\mathbf{s}) \tag{10}$$

$$= \mathbf{z}^T \boldsymbol{\phi}(\mathbf{s}) \tag{11}$$

The details of this approximation are laid out in [1] and [5] and only the results will be stated here. First of all, computational efficiency leads us to only consider *Markovian* Gaussian random fields, and it turns out that a subset of the Gaussian random fields with the *Matérn* covariance functions has the spatial Markov property. The Matérn covariance functions are functions of only the distance $r = \|\mathbf{s} - \mathbf{s}'\|$ between two points

$$c_M(r) = \frac{\sigma^2}{2^{\nu-1}\Gamma\nu}(\kappa\, r)^\nu K_\nu(\kappa\, r), \tag{12}$$

where $K_\nu(\cdot)$ is the modified Bessel function of the second kind, $\nu > 0$ is the smoothing parameter, $\kappa > 0$ is the range parameter and σ^2 is the variance. The Matérn fields have the Markov property when $\alpha = \nu + \dfrac{d}{2}$ is an integer. In our case we have $d = 2$ and we will choose $\nu = 3$, so that $\alpha = 2$. The piecewise linear basis functions are defined on a triangulated mesh of the observation window, so that the value of $\phi_i(\mathbf{s})$ is 1 on vertex i and goes linearly to zero on the neighboring vertices. The mesh is illustrated in figure 2. Second, the random vector $\mathbf{z}$ has a joint Gaussian distribution with a *sparse* precision matrix and so $\mathbf{z}$ is an example of a Gaussian Markov random field (GMRF). This means that evaluating the field is extremely fast.

The integral in (3) is approximated with a sum using a simple deterministic numerical integration scheme with weights $\tilde{\alpha}_i$ and sigma points $\tilde{\mathbf{s}}_i$, so that the approximation to the integral of some function f over the domain Ω is given by

$$\int_\Omega f(\mathbf{s})\ \mathrm{d}\mathbf{s} \approx \sum_{i=1}^{p} \tilde{\alpha}_i\ f(\tilde{\mathbf{s}}_i) \tag{13}$$

Examples of such numerical integration rules are the well known midpoint rule, trapezoid rule and and the Simpson's rule (f is interpolated with a polynomial of order 0, 1 or 2 respectfully).

The main difference to the grid approximation (9) is that the sigma points and the observations corresponding to the point pattern points now differ. The likelihood approximation can however be written as a product of independent Poisson likelihoods also in this case just by considering what constitutes the set of observations $\mathbf{y}$. Let

$$\mathbf{y} = [\mathbf{o}_{p\times 1}^T, \mathbf{1}_{n\times 1}^T]^T \tag{14}$$

$$\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_p, \mathbf{o}_{n\times 1}^T]^T \tag{15}$$

$$\mathbf{z} = [\hat{Z}(\tilde{\mathbf{s}}_1), \ldots, \hat{Z}(\tilde{\mathbf{s}}_n), \hat{Z}(\mathbf{x}_1), \ldots, \hat{Z}(\mathbf{x}_n)]^T, \tag{16}$$

then

$$P(\mathbf{y} \mid \mathbf{z}) \propto \exp\left(-\sum_{i=1}^{n+p} \tilde{\alpha}_i e^{z_i}\right) \prod_{i=1}^{n+p} (\tilde{\alpha}_i e^{z_i})^{y_i} \tag{17}$$

Putting all this together, the approximation to the conditional log-likelihood is given as

$$\log p(\mathbf{y} \mid Z) \approx C - \sum_{i=1}^{p} w_i \exp\left(\mathbf{z}^T \boldsymbol{\phi}(\hat{\mathbf{s}}_i)\right) + \sum_{k=1}^{n} \mathbf{z}^T \boldsymbol{\phi}(\mathbf{s}_k) \tag{18}$$

$$= C - \mathbf{w}^T \exp\left(\mathbf{A}_1 \mathbf{z}\right) + \mathbf{1}^T \mathbf{A}_2 \mathbf{z}, \tag{19}$$

where $[\mathbf{A}_1]_{ij} = \phi_j(\hat{\mathbf{s}}_i)$, $[\mathbf{A}_2]_{kj} = \phi_j(\mathbf{s}_k)$ and C is an unimportant constant.

3 Results

4 Conclusion

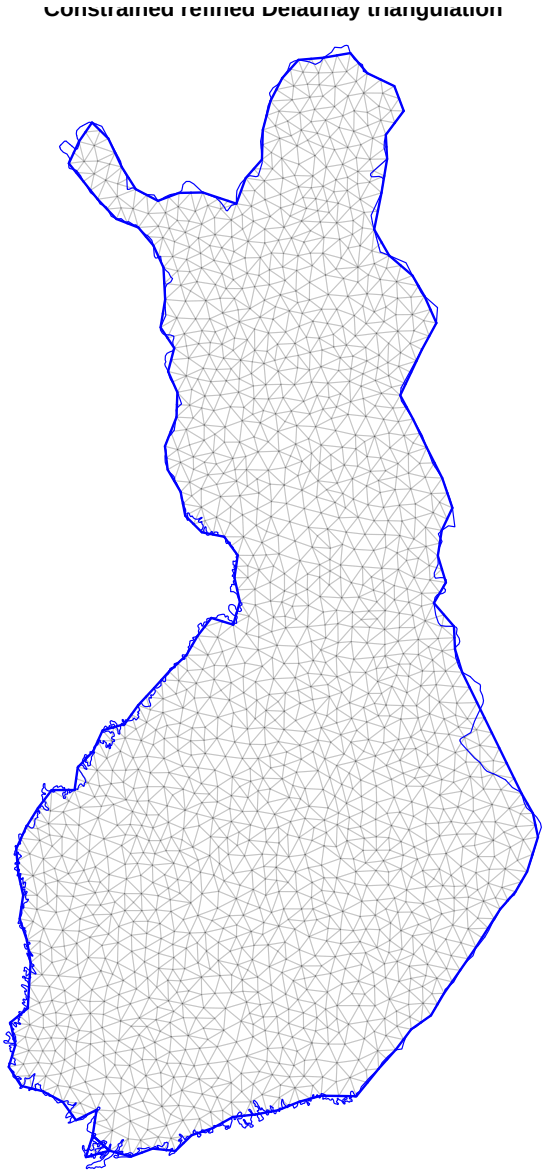


Figure 2: The mesh over the observation window, which is the land area of Finland.

References

[1] Finn Lindgren. "An explicit link between Gaussian fields and Gaussian Markov random fields: The SPDE approach". In: *Science And Technology* (2010). URL: http://inla.googlecode.com/hg-history/default/r-inla.org/papers/spde-jrssb-revised.pdf.

[2] Jesper Møller and Rasmus P. Waagepetersen. "Modern Statistics for Spatial Point Processes". In: *Scandinavian Journal of Statistics* 34.June 2006 (Sept. 2007), 070927154002001–??? ISSN: 0303-6898. DOI: 10.1111/j.1467-9469.2007.00569.x. URL: http://doi.wiley.com/10.1111/j.1467-9469.2007.00569.x.

[3] Håvard Rue, Sara Martino, and Nicolas Chopin. "Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 71.2 (Apr. 2009), pp. 319–392. ISSN: 13697412. DOI: 10.1111/j.1467-9868.2008.00700.x. URL: http://doi.wiley.com/10.1111/j.1467-9868.2008.00700.x.

[4] Daniel Simpson, Janine Illian, and Finn Lindgren. "Going off grid: Computationally efficient inference for log-Gaussian Cox processes". In: *Science And Technology* (2011). URL: http://adsabs.harvard.edu/abs/2011arXiv1111.0641S.

[5] Daniel Simpson and Finn Lindgren. "Think continuous: Markovian Gaussian models in spatial statistics". In: *Arxiv preprint arXiv:1110.6796* (2011). URL: http://arxiv.org/abs/1110.6796.

1 R-code

```
exportFigs <- 1
displayFigs <- 0

plots.obs <- TRUE
plots.mesh <- TRUE
plots.results <- FALSE

run.mesh <- TRUE
run.INLA <- FALSE

source("util.R",local=TRUE)
if(1 | !exists("kunnat")) {
  source("loadData.R",local=TRUE)
  source("dist.R",local=TRUE)
  ocoords <- coords
}
bb <- map@bbox
#corners <- matrix(c(bb[1,1],bb[2,1], bb[1,2],bb[2,1], bb[1,2],bb[2,2], bb
    [1,1],bb[2,2] ),4,2,byrow=TRUE)
#ocoords <- corners[dim(corners)[1]:1,]
#coords <- ocoords
lx <- max(ocoords[,1])-min(ocoords[,1])
ly <- max(ocoords[,2])-min(ocoords[,2])

offX <- mean(ocoords[,1])
offY <- mean(ocoords[,2])

# coords[,1] <- (ocoords[,1]-offX)/ly
# coords[,2] <- (ocoords[,2]-offY)/ly

if(run.mesh) {

  N <- nrow(coords)
  loc.bnd <- coords[N:1,]
  segm.bnd <- inla.mesh.segment(as.matrix(loc.bnd))

  maxedge <- ly/50
  minedge <- maxedge
  mesh <- inla.mesh.create(
                    ## Data locations:
                   bound=segm.bnd,
                    ## Where to put the mesh/graph files:
                    ## Set to >=0 for visual (not on Windows):
                    plot.delay=NULL,
                    keep=FALSE,
                    cutoff=minedge,
                    refine=(list(min.angle=26,
                                 max.edge.data=maxedge,
                                 max.edge.extra=2*maxedge))
          )
```

```r
  spde <- inla.spde.create(mesh, model="matern")
}

#########################
#### OBSERVATIONS ########
#########################

loc <- as.matrix(obs.y2010)
loc <- loc[!duplicated(loc[,c("x","y")]),]
loc <- loc[sample.int(min(2000,nrow(loc))),]

##############################
#### COVARIATES ##############
##############################
distmat <- crossdist(mesh$loc[,1],mesh$loc[,1],cov.pdens[,"x"],cov.pdens[,"y"])
cov.pdens <- cov.pdens[apply(distmat,1,which.min),"pdens"]

#loc <- loc[loc[,1] < 0.17,]

# kappa =0.5
# tau =0.5

############################
#### RESCALE ###############
############################

# loc[,1] <- (loc[,1]-offX)/ly
# loc[,2] <- (loc[,2]-offY)/ly

# ##image.plot(inla.mesh.project(proj, variances^0.5), col=cp(256))

# sample = inla.spde.query(spde, sample=list(tau=tau, kappa2=kappa^2))$sample

# max_points = 10000
# intercept =log(max_points) - max(sample)
# source("Daniel/simulate_lgcp.R",local=TRUE)
# loc <- simulate_lgcp(mesh,intercept+sample)

# print(dim(loc)[1])
# # discard points that fall outside the window
loc <- pip(loc,coords,bound=FALSE)

plot(mesh)
points(loc,col="#ff000080",pch=16)
nV <- mesh$n
nData <- dim(loc)[1]
print(nData)
print(nV)
# print(dim(loc)[1])

if(run.INLA) {
  ## Create the SPDE/GMRF model, (kappa^2-Delta)(tau x) = W:
  #Convenient definitions - number of vertices and number of observations

  #See paper - we need two A matrices, one for the location of the points and
```

```
      one
  #for the integration points (here taken to be the identity matrix)
  LocationMatrix = inla.mesh.project(mesh, loc)$A
  IntegrationMatrix = sparseMatrix(i=1:nV,j=1:nV,x=rep(1,nV)) #simple
      integration scheeme
  ObservationMatrix=rBind(IntegrationMatrix,LocationMatrix) #Glue matrices
      together

  #The integration weights (alpha in the paper)
  IntegrationWeights = diag(spde$internal$c0)
  E_point_process =c(IntegrationWeights,rep(0,nData)) #Glue everything together

  fake_data <- c(rep(0,nV),rep(1,nData)) #Put a zero where there aren't
      observations and a 1 where there is a point

  formula <- y ~ 1 + pdens + f(idx, model=spde) #Basic latent model - feel free
       to add covariates etc

  data <- list(y=fake_data,idx = c(1:nV),pdens=cov.pdens) #put hte data in

  #The INLA call.  Likelihood is Poisson with Observation Matrix and
      appropriate value fo E.
  #result = inla(formula, data=data, family="poisson",
  # control.predictor=list(A=ObservationMatrix),E=E_point_process,verbose=TRUE)

  sigma2.approx = 0.5^2
  range.approx = 0.4
  kappa = sqrt(8)/range.approx
  tau = 1/(4*pi*kappa^2*sigma2.approx)^0.5

  #init.mode = c(-5.5, 3) # works
  #init.mode = c(-8, 5.5) # works
  #init.mode = c(-4.905784,6.317846)
  #init.mode = c(log(tau), log(kappa^2))
  #init.mode = c(-1.304829,2.717723)

  #control.mode = list(theta = init.mode, restart=TRUE)

#The INLA call.  Likelihood is Poisson with Observation Matrix and appropriate
    value fo E.
result =
    inla(formula, data=data, family="poisson",
         control.predictor=list(A=ObservationMatrix,compute=TRUE),
         E=E_point_process,
         #control.mode = list(theta = init.mode, restart=TRUE),
         verbose=TRUE,
          ## We don't need the marginals:
          control.compute = list(return.marginals=FALSE),
          #control.mode = control.mode,
          ## We don't need to overoptimise:
          control.inla=list(tolerance=1e-4)
         )

}
```

```
############################
### PLOTS ##################
############################

if(displayFigs | exportFigs) {

  ms <-summary(mesh)

  wh <- (ms$xlim[2]-ms$xlim[1])/(ms$ylim[2]-ms$ylim[1])

if(plots.results) {
  ########################
  #Some fancy plotting code stolen from Finn
  ############################
  ## Prepare for plotting:

  ## Calculate mapping between triangulation vertices and grid points:

  proj = inla.mesh.projector(mesh, dims=c(200,200))

  ## Construct greyscale palette function:
  my.grey.palette = function (n,...) { return (grey.colors(n,0.05,0.95,...))}
  ## Use it:
  my.palette = my.grey.palette

    cp = colorRampPalette(c("darkblue", "blue", "cyan", "yellow", "red", "
        darkred"))
    my.palette = cp

  ## Construct map data appropriate for easy plotting:
  #mm = calc.map(map)

  levelplotmap = function(..., mm) {
      panel.levelplot(...)
      panel.lines(mm$x, mm$y, col="red")
  }

  ####################
  ## Plot results:

  ## Map resulting posterior mean field to a grid:
  plotdata = inla.mesh.project(proj, result$summary.random$idx[,"mean"])
  ## Plot PM contours:
  if(exportFigs) {
    pdf(file="report/postint_b2010.pdf",width=5,height=5/wh)
    par(mar=mar_tight)
  } else {

    dev.new()
  }
  bbb = (levelplot(row.values=proj$x, column.values=proj$y, x=plotdata,
                   mm=NULL, panel=levelplotmap,
                   col.regions=my.palette,
                   xlim=range(proj$x), ylim=range(proj$y), aspect="iso",
                   contour=TRUE, cuts=11, labels=FALSE, pretty=TRUE,
```

```
                  xlab="Easting",ylab="Northing"))
  print(bbb)
  if(exportFigs) {
    dev.off()
  }

  # pp<-myplot(plotdata,
  #   file="report/postint_b2010.pdf",
  #   plotfn=levelplot,
  #   width=5,
  #   height=5/wh,
  #   afterfn=function(p,k) {
  #     print(p)
  #   },
  #   mar=mar_tight,
  #   main="",
  #   col="#00000020",
  #   row.values=proj$x,
  #   column.values=proj$y,
  #     mm=NULL,
  #     panel=levelplotmap,
  #     col.regions=my.palette,
  #     xlim=range(proj$x), ylim=range(proj$y), aspect="iso",
  #     contour=TRUE, cuts=11, labels=FALSE, pretty=TRUE,
  #     lab="Easting",ylab="Northing"
  # )

  #Plot the interecept and the hyperparameter posteriors
  #plot(result, plot.random.effects=FALSE)

  #Plot the posterior for the RANGE of the random field
  range=inla.tmarginal(function(x) sqrt(8)/exp(x/2),(result$marginals.hyperpar$
      "K.0 for idx-basisK" ))
  attr(range,"inla.tag")="Range"
  #plot(range)

}

if(plots.mesh) {

  ms <-summary(mesh)

  wh <- (ms$xlim[2]-ms$xlim[1])/(ms$ylim[2]-ms$ylim[1])
  myplot(mesh,
    file="report/mesh.pdf",
    width=5,
    height=5/wh,
    afterfn=function(p,k) {
      lines(map,col="#0000ffff")
      print(p)
    },
    mar=mar_tight,
    main="",
    col="#00000020"
  )
```

```
}

if(plots.obs) {

  cut.val <- 0 ### Just to force it.
  theme.novpadding <-
      list(layout.heights =
           list(top.padding = cut.val,
                main.key.padding = cut.val,
                key.axis.padding = cut.val,
                axis.xlab.padding = cut.val,
                xlab.key.padding = cut.val,
                key.sub.padding = cut.val,
                bottom.padding = cut.val),
           layout.widths =
           list(left.padding = cut.val,
                key.ylab.padding = cut.val,
                ylab.axis.padding = cut.val,
                axis.key.padding = cut.val,
                right.padding = cut.val))

  oranges <- sapply(brewer.pal(8,"Oranges"),function(c) {addAlpha(c,0.7)})

  rajj <- list("sp.lines", map, col = "#0000ff70")
  pt <- list(
      "b2010"=list(
        "sp.points",
        obs.y2010,
        pch = 16,
        col = "#00000088",
        cex=0.5),
      "b2009"=list("sp.points",
        obs.y2009[obs.y2009$laji=="karhu",c("x","y")],
        pch = 16,
        col = "#00000088",
        cex=0.5),
      "l2009"=list("sp.points",
        obs.y2009[obs.y2009$laji=="ilves",c("x","y")],
        pch = 16,
        col = "#00000088",
        cex=0.5),
      "w2009"=list("sp.points",
          obs.y2009[obs.y2009$laji=="susi",c("x","y")],
          pch = 16,
          col = "#00000088",
          cex=0.5)
      )

  # plot the points

pnel <- function() {
    #panel.levelplot(...)
    panel.lines(map,col="#0000ffff")
}
```

```
  myplot(y~x,
    obs.y2010,
    file="report/b2010.pdf",
    plotfn=xyplot,
    #panel=panel.lines(map,col="#0000ffff"),
    width=5,
    #height=5/wh,
    afterfn=function(p,k) {
          trellis.par.set(axis.line=list(col=NA))
          print(p)
    },
    mar=mar_tight,
    main="",
    col="#00000050",
    pch=16,
    aspect="iso",
    settings=theme.novpadding
  )

  # plot the population density
  pl_b10 <- myplot(kunnat,
      sp.layout=list(rajj),
        plotfn=spplot,
        file="report/pdens.pdf",
        width=3,
        height=5.5,
        afterfn=function(p,k) {
          trellis.par.set(axis.line=list(col=NA))
          print(p)
        },
        par.settings=theme.novpadding,
        mar=mar_tight,
        main="",
        zcol="pdens_binned",
        col.regions=oranges,
        col="transparent",
        colorkey=TRUE,
        pretty=TRUE
      )
  }
}
```

```
kunnat <- readShapeSpatial("map45/HallintoAlue45Milj_ETRS.shp")
#kraj <- readShapeSpatial("map45/HallintoalueRaja45Milj_ETRS.shp",verbose=TRUE)
#kraj <- kraj[kraj@data$Kohdeluokk==82100|kraj@data$Kohdeluokk==84111,]
city <- readShapeSpatial("map/cityp.shp")
# turn the factor to numeric vector
kunnat@data$Kunta <- as.numeric(as.vector(kunnat@data$Kunta))
hcont <- readShapeSpatial("map45/hcont_l.shp")
map <- hcont[row.names(hcont@data)=="761"|hcont@data$Z==9999,]
#forest <- readShapeSpatial("map/forest.shp")
population <- read.csv(
  "asukastiheys.csv",
```

```
  sep=",",
  header=FALSE,
  skip=0,
  col.names=c("Kunta",
        "land_area",
        "water_area",
        "water_area2",
        "total_area",
        "population",
        "pdens_area",
        "pdens_land_area"))

pdens <- sapply(kunnat@data$Kunta, function(k) {
  if(sum(population$Kunta==k) > 0) {
    tmp <- population[population$Kunta==k,c("pdens_land_area")]
    return(tmp[1])
  }
  print(k)
  return(NA)
})

kunnat_cnt <- coordinates(kunnat)

cov.pdens <- as.data.frame(cbind(
        kunta=kunnat@data$Kunta,
        x=kunnat_cnt[,1],
        y=kunnat_cnt[,2],
        pdens=pdens))

pdensbins <- c(0,1,5,10,20,50,100,1000,Inf)
kunnat@data <- cbind(kunnat@data,pdens=pdens,pdens_binned=cut(pdens,pdensbins))

obs.y2010 <-read.csv("BearObs2010XY_ETRS.csv",
  header=FALSE,
  sep=" ",
  skip=0,
  col.names=c("x","y"))
obs.y2009 <- read.csv(
  "Obs2009XYS_ETRS.csv",
  sep=",",
  header=FALSE,
  skip=0,
  col.names=c("x","y","laji"))
```

```
sp1 <- sapply(coordinates(map), function(x) do.call("rbind", x))
N <- length(sp1)
#sp1 <- sp1[c(1:14,17:N)]
N <- length(sp1)
lps <- matrix(nrow=N,ncol=2)
fps <- matrix(nrow=N,ncol=2)
NNN <- 0
for(i in 1:N) {
  line <- sp1[[i]]
  NN <- nrow(line)
```

```
  NNN <- NNN+NN
  lps[i,] <- line[NN,]
  fps[i,] <- line[1,]

  # for(j in 1:N) {
  #   mi <- colMeans(sp1[[i]])
  #   mj <- colMeans(sp1[[j]])
  #   dist[i,j] = sqrt((mi[1]-mj[1])^2+(mi[1]-mj[1])^2)
  # }
}
#d <- as.matrix(dist(mns))
#d[d==0]=NA
c <- vector("integer",N)
c[1] <- 1
coords <- sp1[[c[1]]][nrow(sp1[[c[1]]]):1,]

for(i in 1:(N-1)) {
  dmin <- 99999999
  jmin <- 0
  fp <- fps[c[i],]
  for(j in 1:N) {
    lp <- lps[j,]
    d <- dist(rbind(fp,lp))[1]

    if(!is.element(j,c) & d < dmin) {
      jmin <- j
      dmin <- d
    }
  }
  if(jmin == 16)
    break
  # print(c[i])
  # print(jmin)
  # print(dmin)
  c[i+1]<-jmin
  nline <- sp1[[jmin]]
  coords <- rbind(coords,nline[nrow(nline):1,])
  #coords <- rbind(coords,nline)
}
coords <- rbind(coords,getAllCoords(map,c(16,3,4,5)))

# coords <- matrix(nrow=0,ncol=2)
# #c <- c(1,unique(c))
# for(i in 1:3) {
#   line <- sp1[c(i)]
#   NN <- nrow(line)
#   coords <- rbind(coords,line[NN:1,])
# }

# coords2 <- matrix(nrow=0,ncol=2)
# for(i in 1:N) {
#   coords2 <- rbind(coords2,fps[i,])
#   coords2 <- rbind(coords2,lps[c(i),])
# }
```