# Intro to spatial models

Anders Nielsen & Ben Bolker

an@aqua.dtu.dk

# Spatial models

- Data often has spatial resolution (e.g. think about species distribution)
- Correlation could be a function of the distance between observation points
- Correlation between observations are described via (often huge) multivariate normals
- (R)TMB was initially built for spatial applications
- Some models can be described surprisingly efficiently by their sparse inverse covariance

# Three main flavors of spatial data

- Geostatistical: Values are observed along with location coordinates
  - Number of clams collected in an individual survey tow, with tow location recorded
- Point Process: The location coordinate is itself the value of interest
  - GPS coordinate pings from a tagged animal
- Areal: Values are observed as the total for an area
  - Total number of scallops harvested within each zone in the Bay of Fundy

- We will focus on geostatistical modelling

# Correlation matrices and functions

- The multivariate normal is used for many things (correlated observations, correlated processes, ...). We write:

$$X \sim \mathcal{N}\left(0,\ \Sigma\right)$$

and then

$$\text{Cor}\left[X_i,\ X_j\right] = \rho_{i,j} = \frac{\Sigma_{ij}}{\sqrt{\Sigma_{ii}\Sigma_{jj}}}$$

- We can do the same in spatial modelling: assume similar locations have similar values
  - Correlation between values at any two locations $\mathbf{s}_1$ and $\mathbf{s}_2$ is a function of their distance

$$\text{Cor}\left[X_{\mathbf{s}_1},\ X_{\mathbf{s}_2}\right] = \mathcal{C}\left(d_{ij}\right) \quad \text{where} \quad d_{ij} = ||\mathbf{s}_1 - \mathbf{s}_2||$$

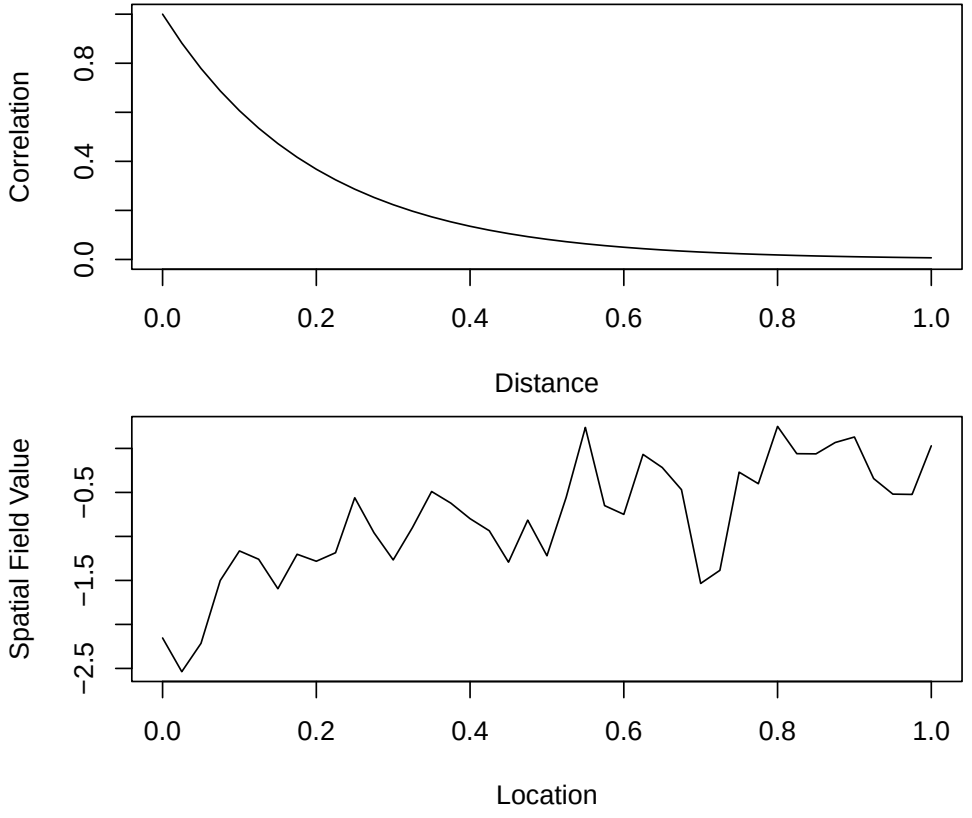
  - Variance of each $X_{\mathbf{s}_1}$ assumed to be constant

$$\begin{bmatrix} X_{\mathbf{s}_1} \\ X_{\mathbf{s}_2} \\ X_{\mathbf{s}_3} \end{bmatrix} \sim \mathcal{N}\left( \begin{bmatrix} \mu \\ \mu \\ \mu \end{bmatrix},\ \sigma^2 \begin{bmatrix} 1 & \mathcal{C}\left(d_{12}\right) & \mathcal{C}\left(d_{13}\right) \\ \mathcal{C}\left(d_{12}\right) & 1 & \mathcal{C}\left(d_{23}\right) \\ \mathcal{C}\left(d_{13}\right) & \mathcal{C}\left(d_{23}\right) & 1 \end{bmatrix} \right)$$

# Two common correlation functions

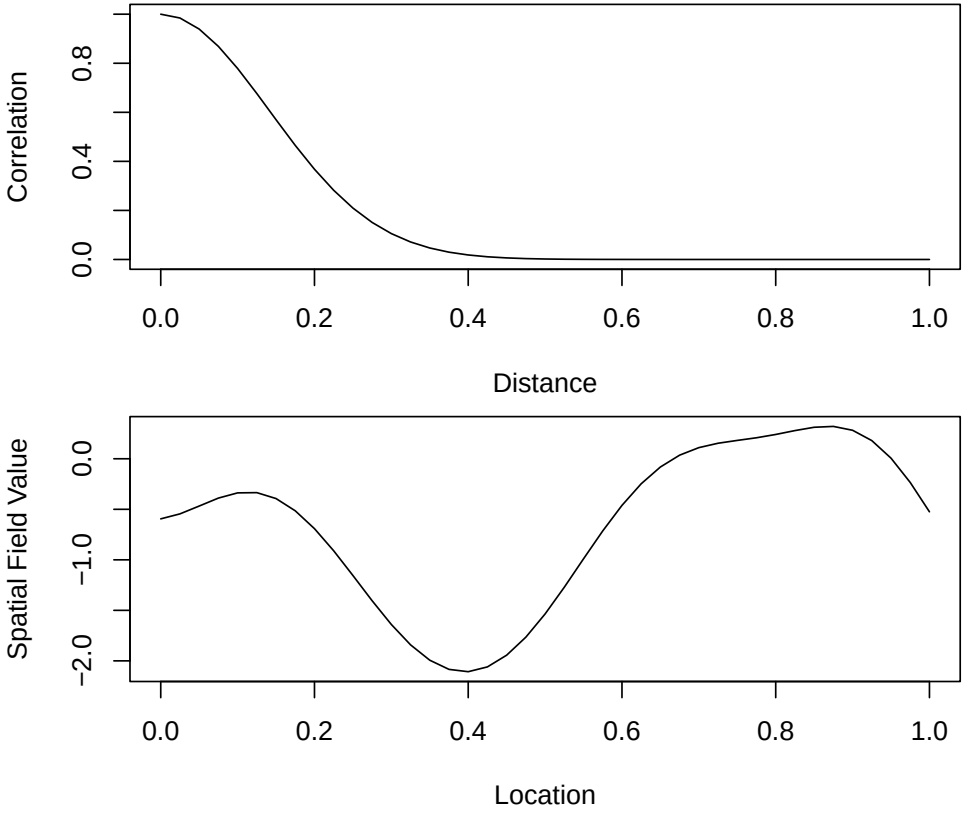
- Exponential
- Noisy looking fields (depends on scale)

$$\mathcal{C}(d) = e^{-d/\rho}$$




- Gaussian
- Very smooth looking fields (depends on scale)

$$\mathcal{C}(d) = e^{-(d/\rho)^2}$$



# Specifying the covariance directly

- Consider a situation where we collect count observations on a grid (does not have to be a grid)
- We expect that our observations are spatially correlated (highly correlated when close, and less correlated when far apart).
- The model is:

$$Y_i \sim \text{Pois}(e^{\mu+\omega_i}) \quad , \quad \text{where}$$
$$\omega \sim \mathcal{N}(0, \Sigma)$$

- The correlation is in the Gaussian part only, so it only depends on the structure of $\Sigma_{i,i'} = \sigma^2 \mathcal{C}(d_{i,i'})$
- Let's start with the exponential correlation structure:

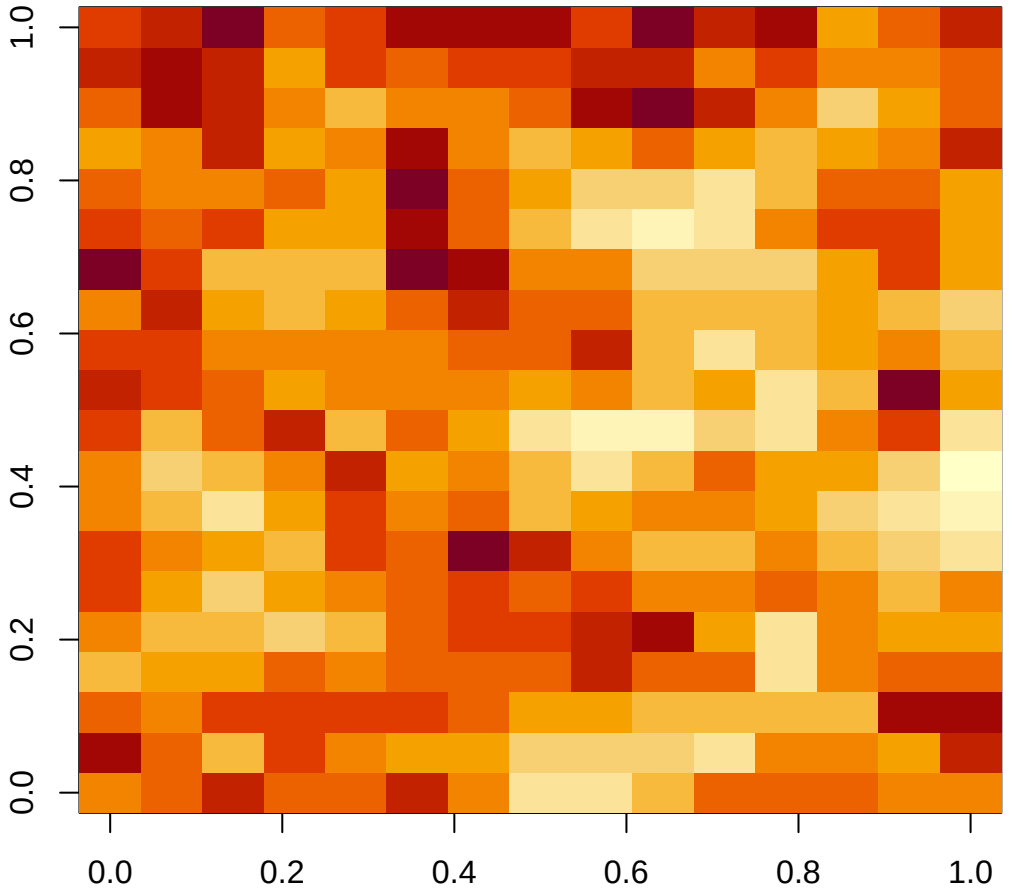
$$\mathcal{C}(d_{i,i'}) = \text{cor}(\omega_i, \omega_{i'}) = \exp\left(\frac{-d_{i,i'}}{\varphi}\right) \quad , \quad \text{where } d_{i,i'} \text{ is distance and } \varphi > 0$$

- and simulating it

# Simulated exponential correlation structure

```
set.seed(123)
#Setup Grid locations
x <- 1:15
y <- 1:20
xy <- expand.grid(x,y)
n <- nrow(xy)

#Euclidian distance matrix
d <- as.matrix(dist(xy))

#Simulate Omega values using exponential function
phi <- 2.5
sigma <- 1.5
Sigma <- exp(-d/phi)*sigma^2

omega <- mvtnorm::rmvnorm(1, rep(0,n), Sigma)

mu <- 2
y.sim <- rpois(n,exp(mu+omega))
sim1 <- list(Y=y.sim, D=d)

save(sim1, file="sim1.RData")
```

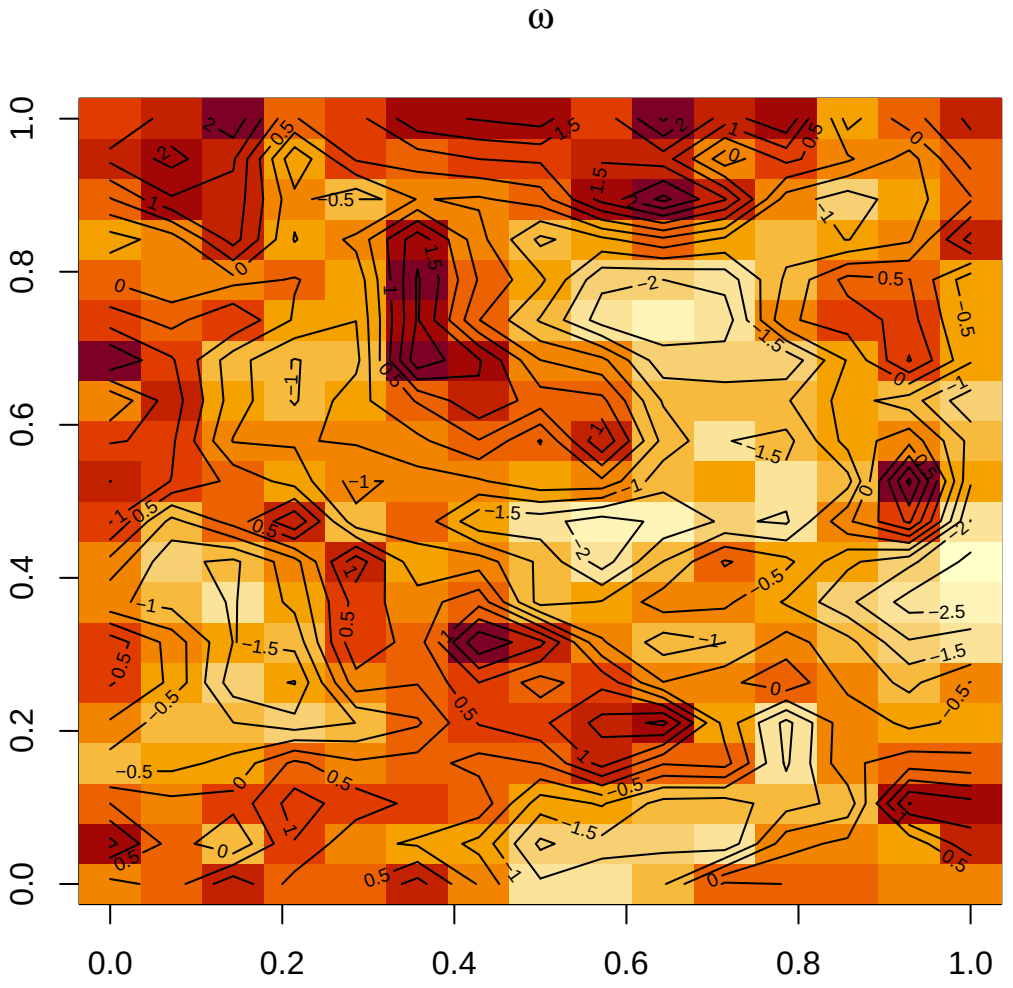
sim1.R



# Estimated exponential correlation structure

```
load("sim1.RData") # gets sim1 with "Y" vector and "D" matrix
library(RTMB)

f<-function(par){
  getAll(par,sim1)
  phi <- exp(logPhi)
  sigma <- exp(logSigma)
  S <- exp(-D/phi)*sigma^2
  nll <- -dmvnorm(omega, mu=0, Sigma=S, log=TRUE)
  nll <- nll - sum(dpois(Y,exp(mu+omega),log=TRUE))
  nll
}
par<-list()
par$mu <- 0
par$logPhi <- 0
par$logSigma <- 0
par$omega <- numeric(nrow(sim1$D))

obj <- MakeADFun(f, par, random="omega")
fit <- nlminb(obj$par, obj$fn, obj$gr)
sdr <- sdreport(obj)
est1<-list(fit=fit, sdr=sdr)
save(est1, file="est1.RData")
```

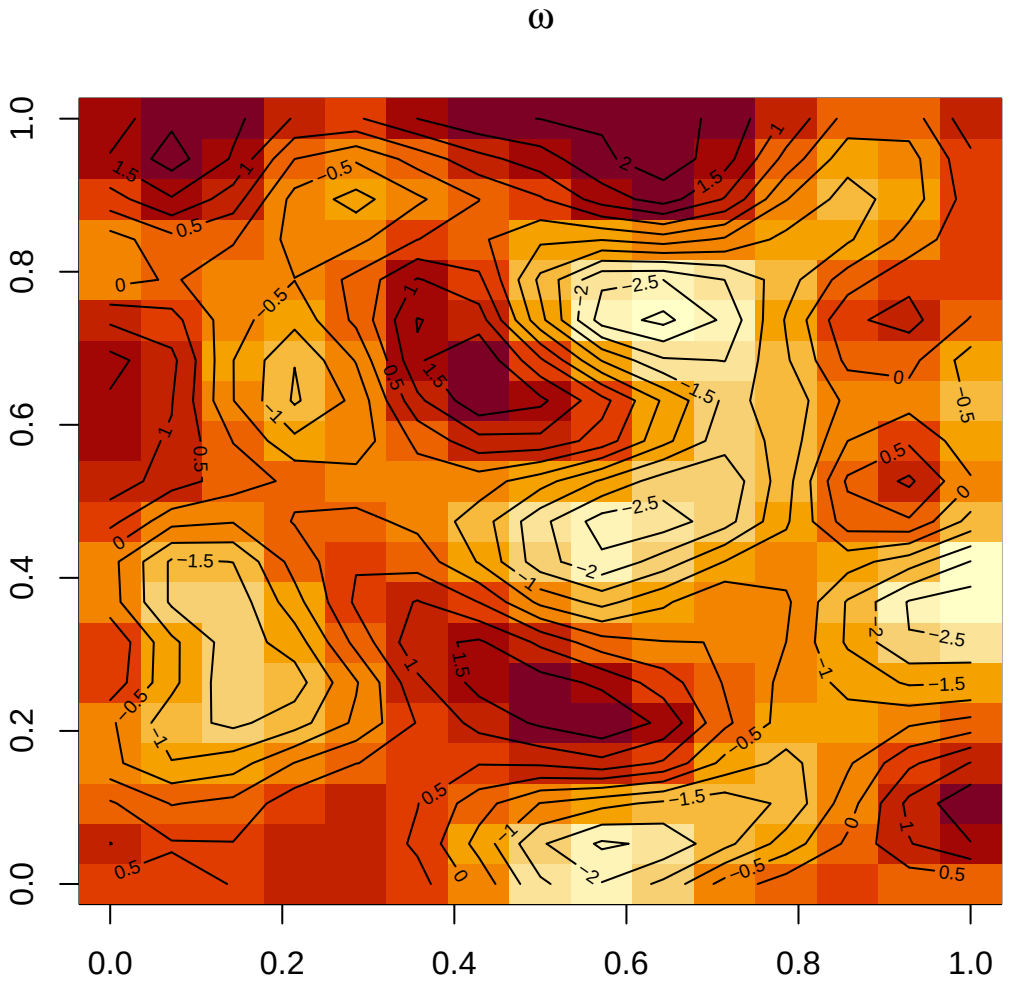
est1.R



# Result from a similar example with Gaussian correlation

```
load("sim1b.RData") # gets sim1 with "Y" vector and "D" matrix
library(RTMB)

f<-function(par){
  getAll(par,sim1)
  phi <- exp(logPhi)
  sigma <- exp(logSigma)
  S <- exp(-(D/phi)^2)*sigma^2
  nll <- -dmvnorm(omega, mu=0, Sigma=S, log=TRUE)
  nll <- nll - sum(dpois(Y,exp(mu+omega),log=TRUE))
  nll
}
par<-list()
par$mu <- 0
par$logPhi <- 0
par$logSigma <- 0
par$omega <- numeric(nrow(sim1$D))

obj <- MakeADFun(f, par, random="omega")
fit <- nlminb(obj$par, obj$fn, obj$gr)
sdr <- sdreport(obj)
est1<-list(fit=fit, sdr=sdr)
save(est1, file="est1b.RData")
```

est1b.R



# Example: Mite data with Gaussian correlation (thanks Andrea!)

- The `mite` data set from the `vegan` R-package is used for this example. To get the data type:

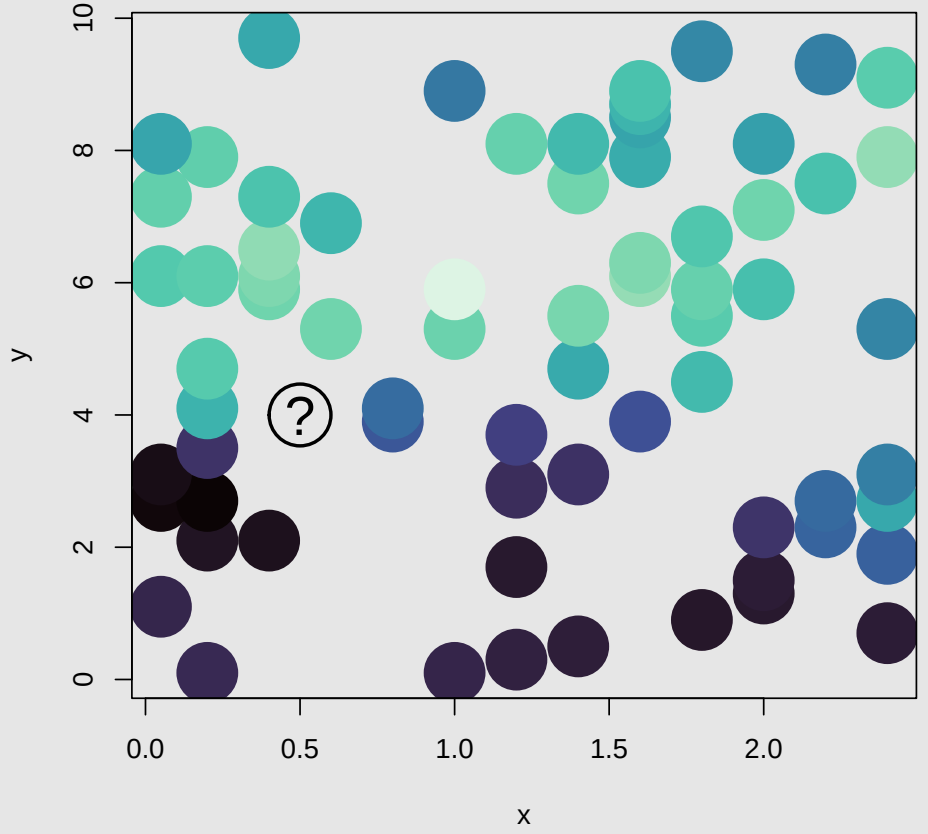
```
install.packages("vegan")
library(vegan)
data("mite") #Counts for each mite species we use column 31
data("mite.env") #Environmental covariate data
data("mite.xy") #Locations
```

- We want to setup a model where the log of the mean of the Poisson count $Y$ for species number 31 is a function of a spatial random field and the environmental covariate `SubsDens`. The model is:

$$Y_i \sim \text{Pois}(\lambda_i) \quad , \quad \text{where}$$
$$\log(\lambda_i) = \mu + \alpha \texttt{SubsDens}_i + \omega_i \quad , \quad \text{where}$$
$$\omega \sim \mathcal{N}(0, \Sigma) \quad , \quad \text{where} \quad \Sigma_{i,i'} = \sigma^2 \exp(-(d_{i.i'}/\phi)^2)$$

- An implementation is on the next slide.
- How could we predict a value of $\omega$ at position $(0.5, 4)$?

```r
library(RTMB)
library(vegan)
data(mite)
data(mite.env)
data(mite.xy)
xy<-mite.xy

dat <- list(Y=mite[,31], X=mite.env$SubsDens, D=as.matrix(dist(xy)))
param <- list(mu=0, alpha=0, logPhi=0, logSigma=0, omega=numeric(nrow(mite)))

f <- function(par){
  getAll(par,dat)
  phi <- exp(logPhi)
  sigma <- exp(logSigma)
  S <- sigma^2*exp(-(D/phi)^2)
  nll <- -dmvnorm(omega,mu=0,Sigma=S,log=TRUE)
  lambda <- exp(mu+alpha*X+omega)
  nll <- nll -sum(dpois(Y, lambda, log=TRUE))
  nll
}

obj <- MakeADFun(f, param, random = "omega", silent=TRUE)

fit <- nlminb(obj$par, obj$fn, obj$gr)

sdr <- sdreport(obj)
pl <- as.list(sdr, "Est")
plsd <- as.list(sdr, "Std")

cc <- viridis::mako(500)
lamCol<-cc[as.numeric(cut(pl$omega,breaks = 501))]
plot(xy, col=lamCol, pch=16, cex=5)
```




files/mite.R

# Multivariate normal

A couple of functions to help define multivariate normal densities are:

`dmvnorm` Multivariate normal density specified via mean vector and covariance matrix

`dgmrf` Multivariate normal density specified via mean vector and sparse inverse covariance

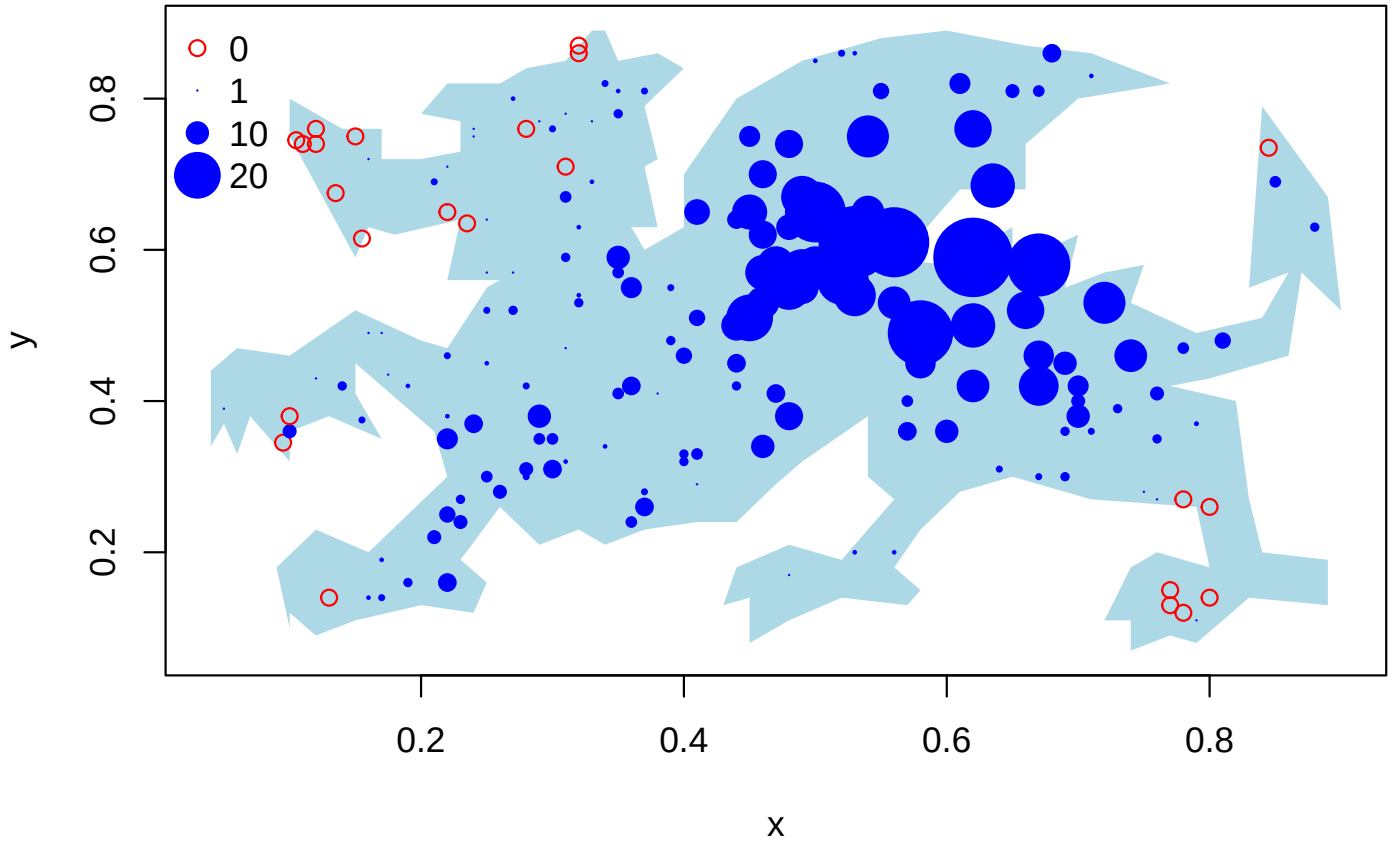
`dautoreg` Multivariate normal density with AR($k$) covariance structure specified via mean vector and $\phi$ vector. The order ($k$) is determined by the length of the $\phi$ vector

`dseparable` Multivariate normal density defined as separable extensions of 2 or more already defined densities

In addition to these there is a function `unstructured(k)` to help setup unstructured covariances to use with `dmvnorm` and further the functions `dmvnorm`, `dgmrf`, and `dautoreg` have an argument `scale` (which can be a single element or a vector) to scale the standard deviation.

# Example: Mapping of plastic pollution

- Plastic particles are counted in 1l water samples from a lake at an undisclosed location
- Wish to map the intensity of this polution




- Observations are in the file `plastic.dat` and the area is define by the polygon in `slogo.RData`

# Simple GMRF spatial (CAR model)

- A simple spatial model can be defined as: $Y_i \sim \mathcal{P}(\lambda_{\texttt{cell}_i})$, where $\log(\lambda) \sim \mathcal{N}(\mu, Q^{-1})$

```
1  jnll <- function(par){
2    getAll(par, dat)
3    delta <- exp(logDelta)
4    sigma <- exp(logSigma)
5    Q <- Q0 + delta*I
6    ret <- -dgmrf(logLambda-mu, Q=Q, log=TRUE, scale=sigma)
7    ret <- ret-sum(dpois(count[inside], exp(logLambda[cellidx[inside]]), log=TRUE))
8    ret
9  }
```

files/sp.R

- Inverse covariance here is $Q = Q_0 + \delta I$, where:

$$Q_0(p, \tilde{p}) = \begin{cases} |\text{neighbors}(p)| & , \quad p = \tilde{p} \\ -1 & , \quad p \sim \tilde{p} \\ 0 & , \quad \text{otherwise} \end{cases}$$

- Look at file `sp.R` for more details. Change resolution to get a sense of speed.

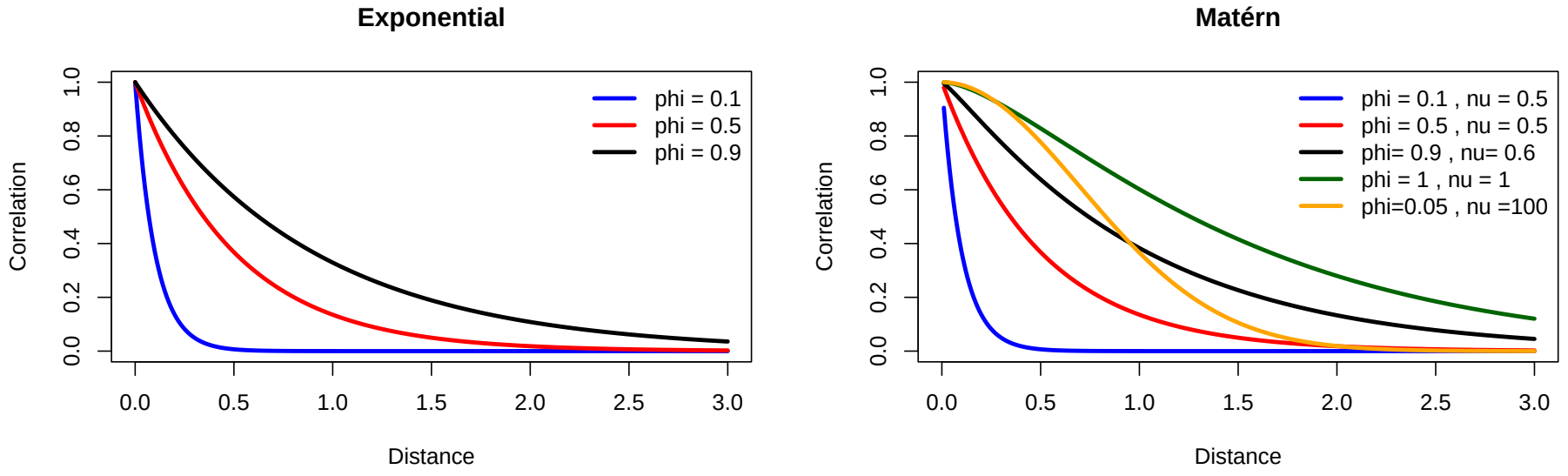
# Matérn correlation

- The Matérn correlation function is another popular option:

$$\text{cor}(\omega_i, \omega_{i'}) = \frac{1}{2^{\nu-1}\Gamma(\nu)} \left(\frac{d_{i,i'}}{\varphi}\right)^{\nu} K_\nu \left(\frac{d_{i,i'}}{\varphi}\right) \quad , \quad \text{where } d_{i,i'} \text{ is distance and } \varphi > 0$$

  where $\Gamma$ is the gamma function and $K_\nu$ is the modified Bessel function of the second kind.

- The Matérn correlation is equal to the exponential when $\nu = 0.5$. Try fixing $\nu$ to 0.5 to verify.



# Setting up the corresponding inverse covariance Q via mesh

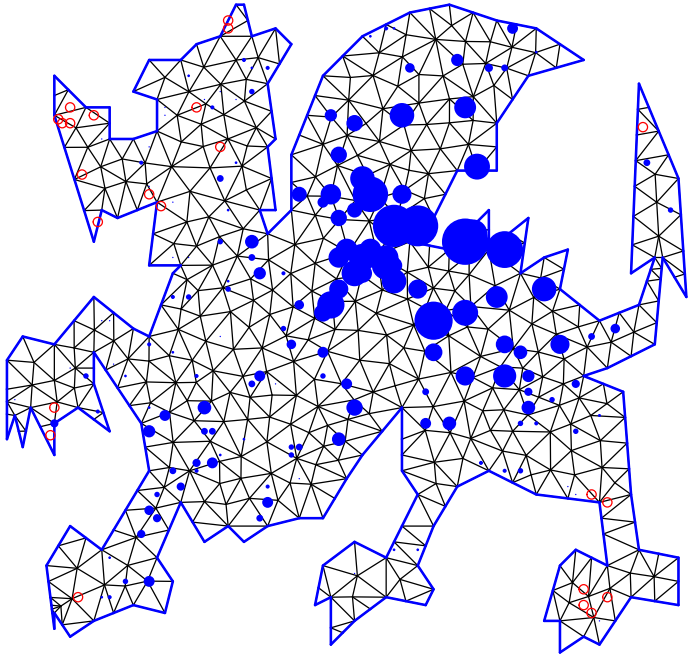
- Three sparse matrices $M_0$, $M_1$, and $M_2$ are defined via the neighborhood structure (mesh)
- These matrices are used to construct $Q = \Sigma^{-1}$ (solution to SPDE, see Lindgren et al 2011)

$$Q = \tau^2(\kappa^4 M_0 + 2\kappa^2 M_1 + M_2)$$

- The random effects (defined at each mesh intersection) will approximately have the Matérn covariance structure with $\nu = 1$
- We will use same example as before

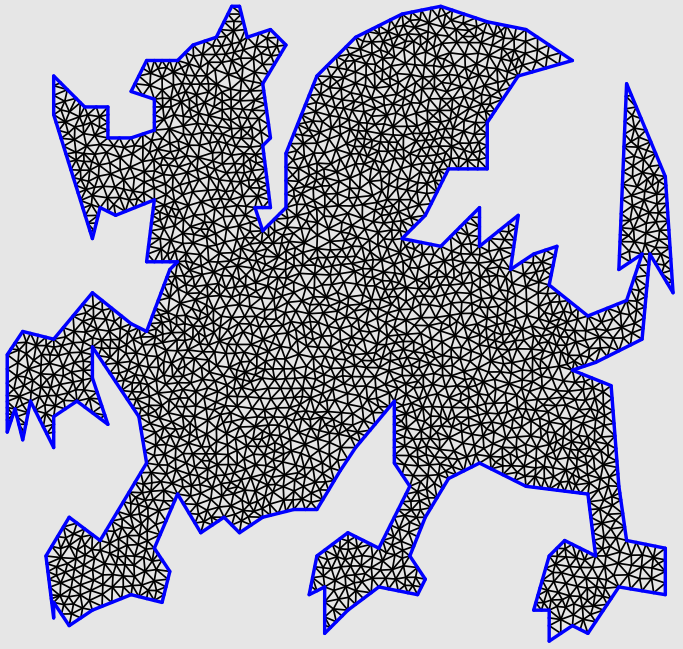
# Steps in sparse spatial modelling using Matérn

1. Create helper objects
   - mesh: random effect locations
   - spde object: precision matrix constructors
   - interpolators: sparse matrices to interpolate spatial field to data locations and prediction locations (if needed)
2. Compute likelihood
   - Compute precision matrix from helper objects
   - Evaluate built-in dgmrf density
   - Interpolate spatial field for observations and predictions



# Setup mesh

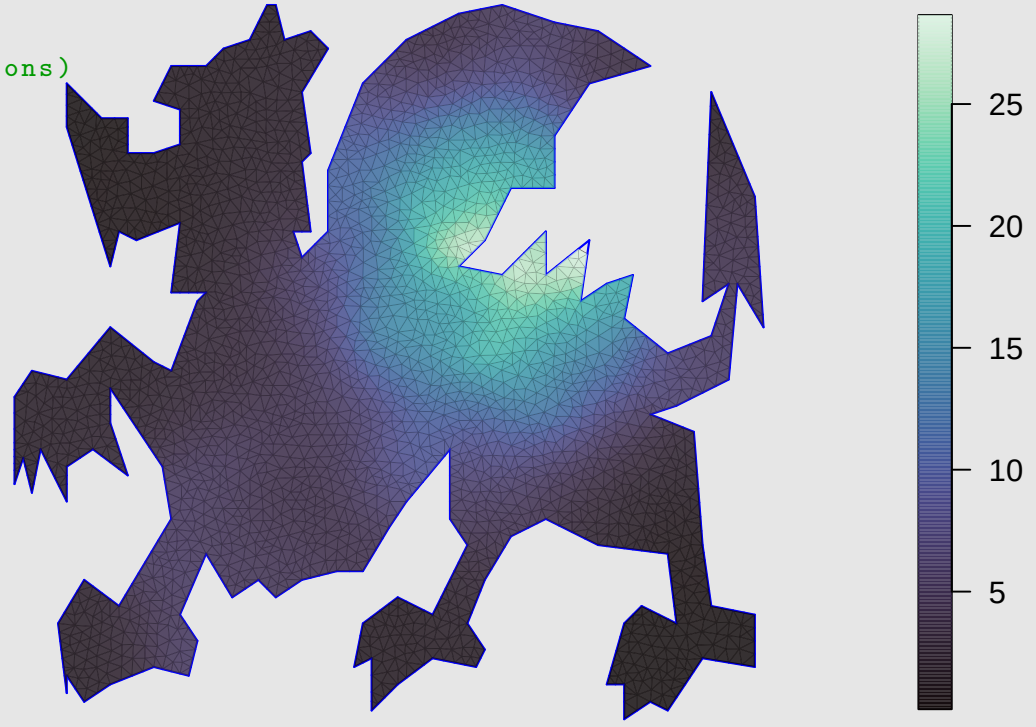
```
1  library(RTMB)
2  library(Matrix)
3  library(fmesher)
4
5  load("slogo.RData")
6  pol$x<-c(pol$x,pol$x[1]) # following methods are fuzzy about having a closed polygon
7  pol$y<-c(pol$y,pol$y[1])
8
9  # Observations
10 dat <- as.list(read.table("plastic.dat", header=TRUE))
11
12 # setup mesh
13 mesh <- fm_mesh_2d(boundary = sf::st_polygon(list(cbind(pol$x, pol$y))), max.edge=c(.02))
14 spde <- fm_fem(mesh)
15 plot(mesh, edge.col="gray")
16 points(dat$x, dat$y, cex=dat$count/max(dat$count)*5)
17 projObs <- fm_basis(mesh, loc = cbind(dat$x, dat$y))
18
19 # add objects to data set
20 dat$mesh <- mesh
21 dat$spde <- spde
22 dat$projObs <- projObs
```



files/spmesh.R

# The likelihood part

```
1  # define parameters
2  par <- list()
3  par$mu <- 0
4  par$logTau <- 0
5  par$logKappa <- 0
6  par$logLambda <- numeric(dat$mesh$n)
7
8  # setup likelihood (here gaussian field for log-intensity of pois observations)
9  jnll <- function(par){
10   getAll(par, dat)
11   tau <- exp(logTau)
12   kappa <- exp (logKappa)
13   Q <- tau*(kappa^4*spde$c0+2*kappa^2*spde$g1+spde$g2)
14   ret <- -dgmrf(logLambda-mu, Q=Q, log = TRUE)
15   logPredObs <- projObs%*%logLambda
16   ret <- ret -sum(dpois(count, exp(logPredObs), log=TRUE))
17   sdField <- sqrt(1/(4*pi*tau^2*kappa^2))
18   range <- sqrt(8)/kappa
19   ADREPORT(sdField)
20   ADREPORT(range)
21   ret
22 }
23
24 obj <- MakeADFun(jnll, par, random="logLambda", silent=FALSE)
25 fit <- nlminb(obj$par, obj$fn, obj$gr)
```




files/spmesh.R

# Comments

- In spatial applications on the globe consider map projection (e.g. UTM)
- Space-time models can be efficiently implemented if separable. E.g. as an AR(1) process in the time dimension and as a GMRF in the space dimension.
- See file `spmeshtime.R` for example. The cental part is:

```
1   d1 <- function(x)dgmrf(x, Q=Q, log=TRUE)
2   d2 <- function(x)dautoreg(x, phi=phi, log=TRUE)
3   ret <- -dseparable(d1,d2)(logLambda-mu)
```

files/spmeshtime.R

- Efficiency is obtained by using sparse matrices, so make sure matrix libraries are properly set up.
- Coding these models is not too bad, but important to understand what is going on.
- You now have all the pieces needed to build advanced spatio-temporal models.