Some new items in the space-time statistical toolbox that may be useful in fisheries

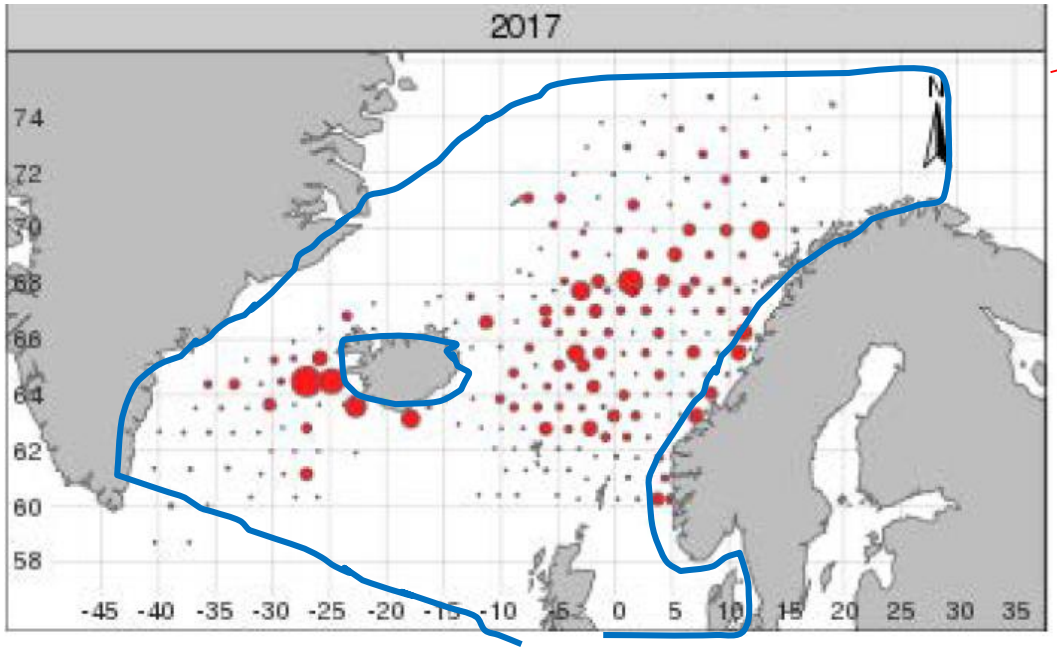
CAPAM workshop, February 26, 2018

Hans J. Skaug
University of Bergen

Outline

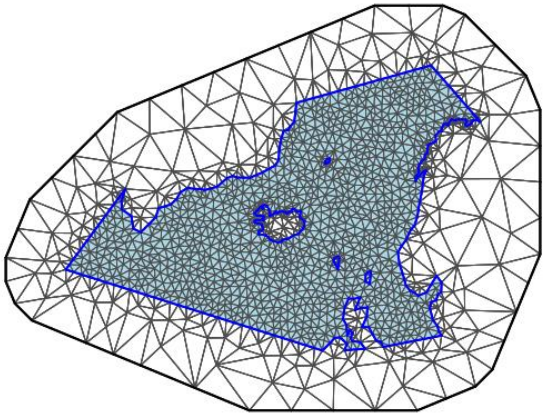
- TMB (Template Model Builder)
- Using TMB with SPDE/INLA spatial models
  - Barriers models (e.g. islands in the ocean)
- Using TMB with Soap smoothers from `mgcv`
  - “Induced” spatial covariance
- Application to North East Atlantic mackerel

Motivation: North East Atlantic mackerel



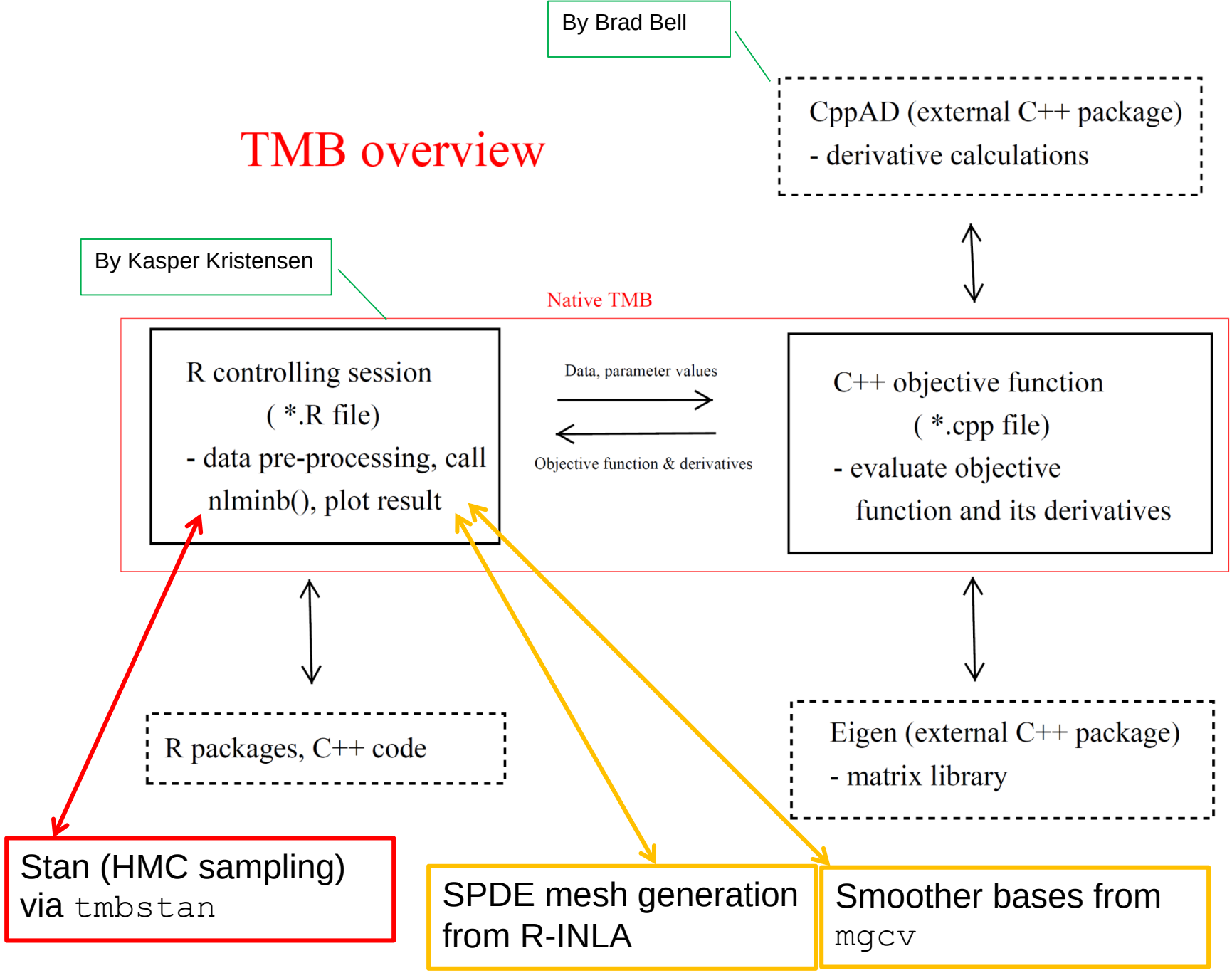
Unpublished data by
Nikos Nikolioudakis
IMR, Bergen

Triangulation created in
R-INLA

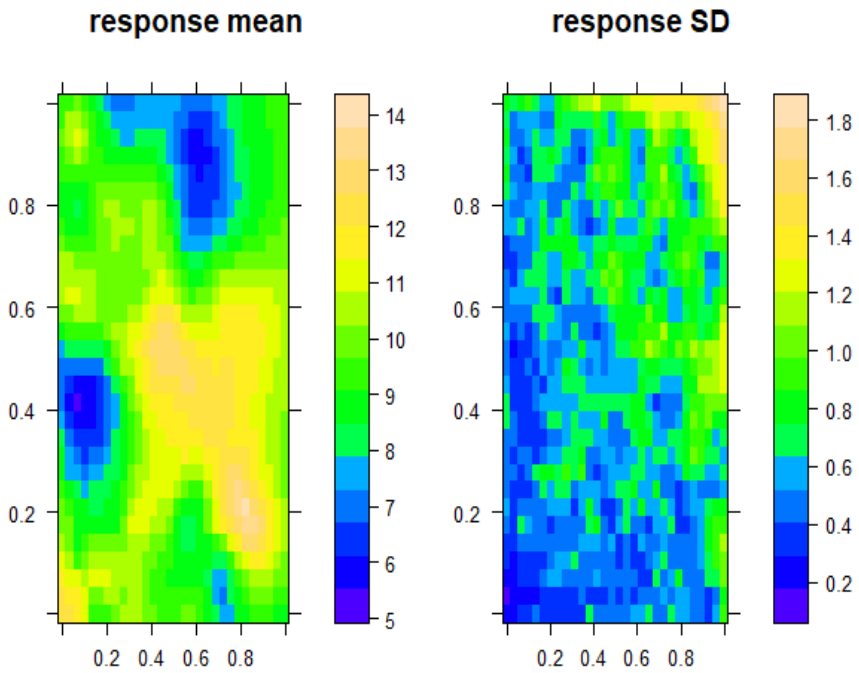


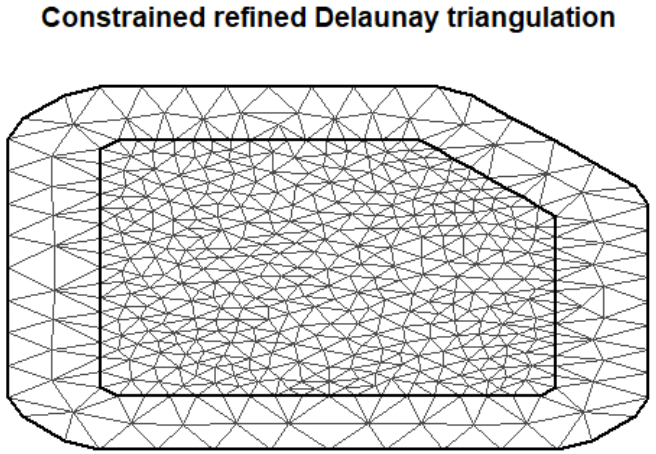
What is TMB?

- «Template Model Builder»
  - R-package on CRAN
- Makes linking R to C++ easy
  - Like `Rcpp`, but with additional functionality
- Key features?
  - *Automatic Differentiation*
  - *Automatic Laplace approximation*
- TMB developer: Kasper Kristensen, DTU


By Brad Bell
TMB overview
CppAD (external C++ package)
- derivative calculations
By Kasper Kristensen
Native TMB
R controlling session
(*.R file)
- data pre-processing, call
nlminb(), plot result
Data, parameter values
Objective function & derivatives
C++ objective function
(*.cpp file)
- evaluate objective
function and its derivatives
R packages, C++ code
Eigen (external C++ package)
- matrix library
Stan (HMC sampling)
via tmbstan
SPDE mesh generation
from R-INLA
Smoother bases from
mgcv

Using SPDE/INLA meshes in TMB

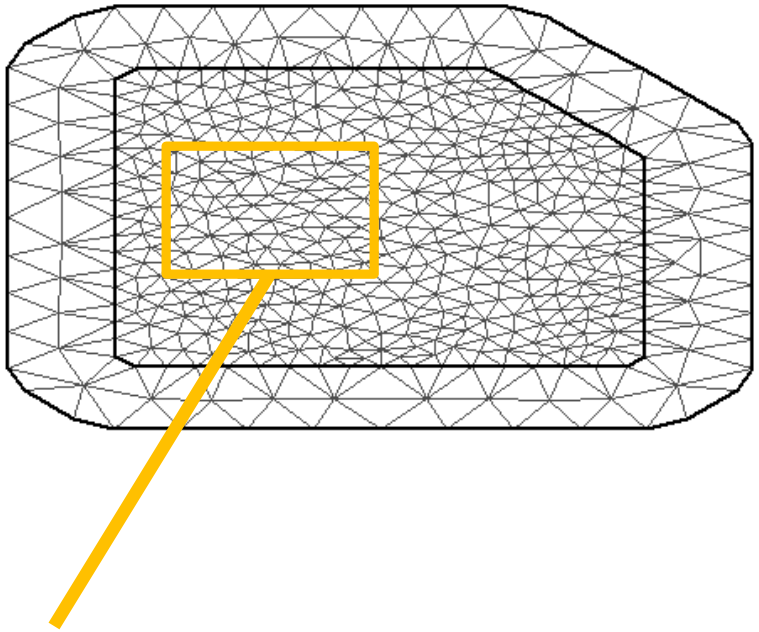


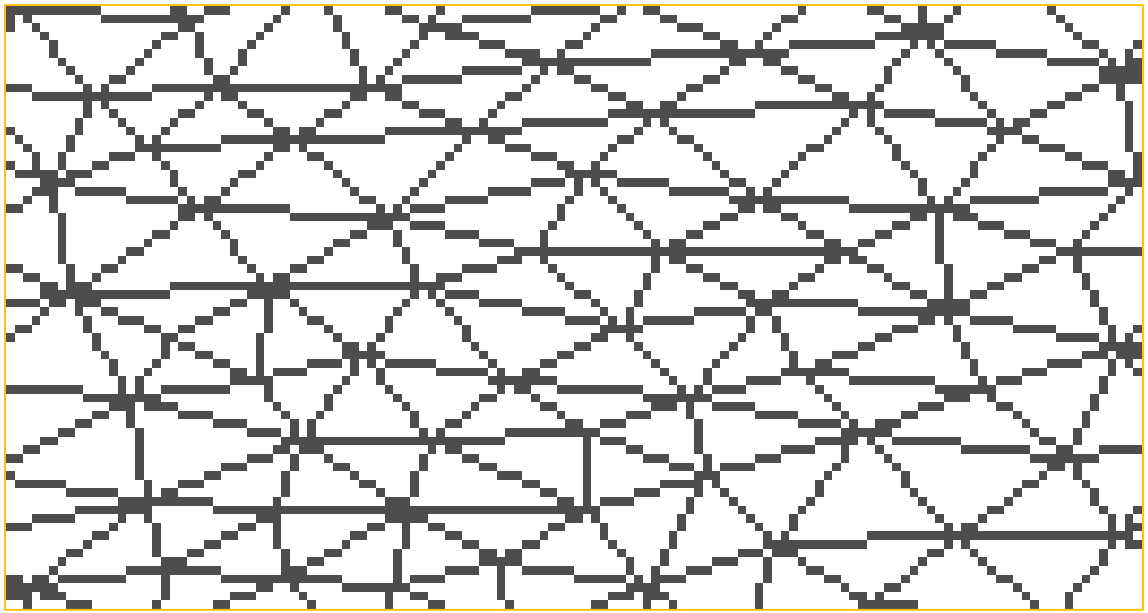


Basis for MLE

Integration in R^n by Laplace approximaiton

The A-matrix





RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

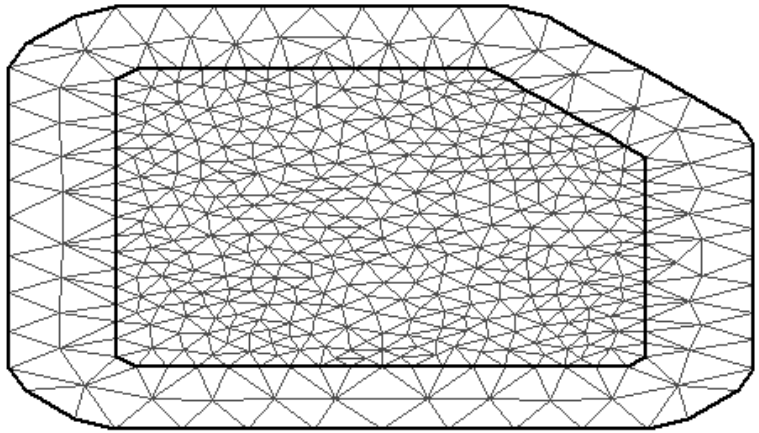
Go to file/function | Addins

SPDEtoy.R*

Source on Save | Run

```
1  library(TMB)
2  compile("SPDEtoy.cpp")
3  dyn.load(dynlib("SPDEtoy"))
4
5  library(INLA)
6  data(SPDEtoy)
7  head(SPDEtoy)
8  pl.dom <- cbind(c(0,1,1,0.7,0), c(0,0,0.7,1,1))
9  mesh5 <- inla.mesh.2d(loc.domain = pl.dom, max.e=c(0.092, 0.2))
10 #plot(mesh5)
11 spde5 <- inla.spde2.matern(mesh=mesh5, alpha=2)
12 coords <- as.matrix(SPDEtoy[,1:2])
13 A5 <- inla.spde.make.A(mesh5, loc=coords)
14 n = ncol(A5)
15
16 # For prediction of the response
17 coords_pred = expand.grid(x1=seq(0,1,l=30),x2=seq(0,1,l=30))   # grid
18 A_pred <- inla.spde.make.A(mesh5, loc=as.matrix(coords_pred))  # projection matrix
19
20 # Data and parameters for TMB
21 data <- list(y=SPDEtoy$y,A=A5,A_pred=A_pred)
22 data$spde <- spde5$param.inla[c("M0","M1","M2")]  # Encapsula
23 parameters = list(
24              beta0=mean(SPDEtoy$y),
25              log_sigma_e=0,
26              log_tau=-2.0,
27              log_kappa=2.5,
28              x=rep(0.0,n))
29
30 obj <- MakeADFun(data,parameters,random="x",DLL="SPDEtoy")
31 opt <- nlminb(obj$par,obj$fn,obj$gr)
32
```

Constrained refined Delaunay triangulation



Interface to R-INLA

RStudio

SPDEtoy.R*

```r
library(TMB)
compile("SPDEtoy.cpp")
dyn.load(dynlib("SPDEtoy"))

library(INLA)
data(SPDEtoy)
head(SPDEtoy)
pl.dom <- cbind(c(0,1,1,0.7,0), c(0,0,0.7,1,1))
mesh5 <- inla.mesh.2d(loc.domain = pl.dom, max.e=c(0.092
#plot(mesh5)
spde5 <- inla.spde2.matern(mesh=mesh5, alpha=2)
coords <- as.matrix(SPDEtoy[,1:2])
A5 <- inla.spde.make.A(mesh5, loc=coords)
n = ncol(A5)

# For prediction of the response
coords_pred = expand.grid(x1=seq(0,1,l=30),x2=seq(0,1,l=
A_pred <- inla.spde.make.A(mesh5, loc=as.matrix(coords_p

# Data and parameters for TMB
data <- list(y=SPDEtoy$y,A=A5,A_pred=A_pred)
data$spde <- spde5$param.inla[c("M0","M1","M2")]  # Enca
parameters = list(
              beta0=mean(SPDEtoy$y),
              log_sigma_e=0,
              log_tau=-2.0,
              log_kappa=2.5,
              x=rep(0.0,n))

obj <- MakeADFun(data,parameters,random="x",DLL="SPDEtoy
opt <- nlminb(obj$par,obj$fn,obj$gr)
```

SPDEtoy.cpp

```cpp
// Illustration SPDE/INLA approach to sp

#include <TMB.hpp>

template<class Type>
Type objective_function<Type>::operator(
{
  using namespace R_inla;
  using namespace density;
  using namespace Eigen;

  DATA_VECTOR(y);
  DATA_SPARSE_MATRIX(A);
  DATA_SPARSE_MATRIX(A_pred);
  DATA_STRUCT(spde,spde_t);

  PARAMETER(beta0);
  PARAMETER(log_sigma_e);
  PARAMETER(log_tau);
  PARAMETER(log_kappa);
  PARAMETER_VECTOR(x);

  Type sigma_e = exp(log_sigma_e);
  Type tau = exp(log_tau);
  Type kappa = exp(log_kappa);

  Type nll = 0.0;

  SparseMatrix<Type> Q = Q_spde(spde,kappa);

  nll = GMRF(Q)(x);                                  // Negative log likeli

  vector<Type> mu = beta0+(A*x)/tau;
  nll -= dnorm(y,mu,sigma_e,true).sum();

  // Report section
  double nu = 1.0;                 // nu = alpha-d/2 = 2-1 by eqn (2) in Lin
  Type rho = sqrt(8*nu)/kappa;  // Distance at which correlation has dr
  ADREPORT(rho);

  vector<Type> y_pred = beta0 + (A_pred*x)/tau;
  ADREPORT(y_pred);
```

Multivariate normal density

R_inla.hpp

```cpp
};

/** Precision matrix eqn (10) in Lindgren et al. (2011
template<class Type>
  SparseMatrix<Type> Q_spde(spde_t<Type> spde, Type kappa){
  Type kappa_pow2 = kappa*kappa;
  Type kappa_pow4 = kappa_pow2*kappa_pow2;

  return kappa_pow4*spde.M0 + Type(2.0)*kappa_pow2*spde.M1 + spde.M2;     // M0=G0, M1=G1, M2=G2
}
```

Eqn (22) in Lindgren et al 2011

RStudio

Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins

SPDEtoy.R*

Source on Save

```r
library(TMB)
compile("SPDEtoy.cpp")
dyn.load(dynlib("SPDEtoy"))

library(INLA)
data(SPDEtoy)
head(SPDEtoy)
pl.dom <- cbind(c(0,1,1,0.7,0), c(0,0,0.7,1,1))
mesh5 <- inla.mesh.2d(loc.domain = pl.dom, max.e=c(0.092, 0
#plot(mesh5)
spde5 <- inla.spde2.matern(mesh=mesh5, alpha=2)
coords <- as.matrix(SPDEtoy[,1:2])
A5 <- inla.spde.make.A(mesh5, loc=coords)
n = ncol(A5)

# For prediction of the response
coords_pred = expand.grid(x1=seq(0,1,l=30),x2=seq(0,1,l=30)
A_pred <- inla.spde.make.A(mesh5, loc=as.matrix(coords_pred

# Data and parameters for TMB
data <- list(y=SPDEtoy$y,A=A5,A_pred=A_pred)
data$spde <- spde5$param.inla[c("M0","M1","M2")]  # Encapsu
parameters = list(
              beta0=mean(SPDEtoy$y),
              log_sigma_e=0,
              log_tau=-2.0,
              log_kappa=2.5,
              x=rep(0.0,n))

obj <- MakeADFun(data,parameters,random="x",DLL="SPDEtoy")
opt <- nlminb(obj$par,obj$fn,obj$gr)
```

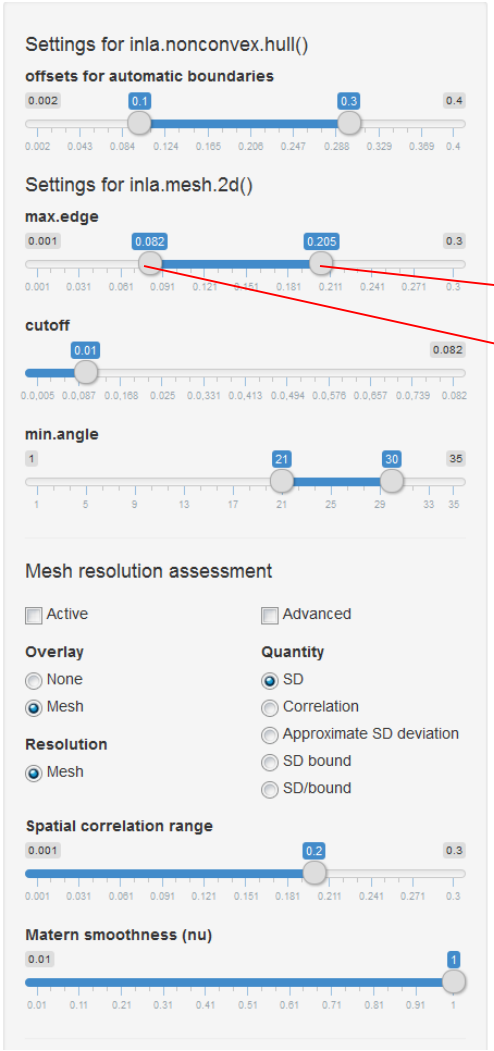
SPDEtoy.cpp

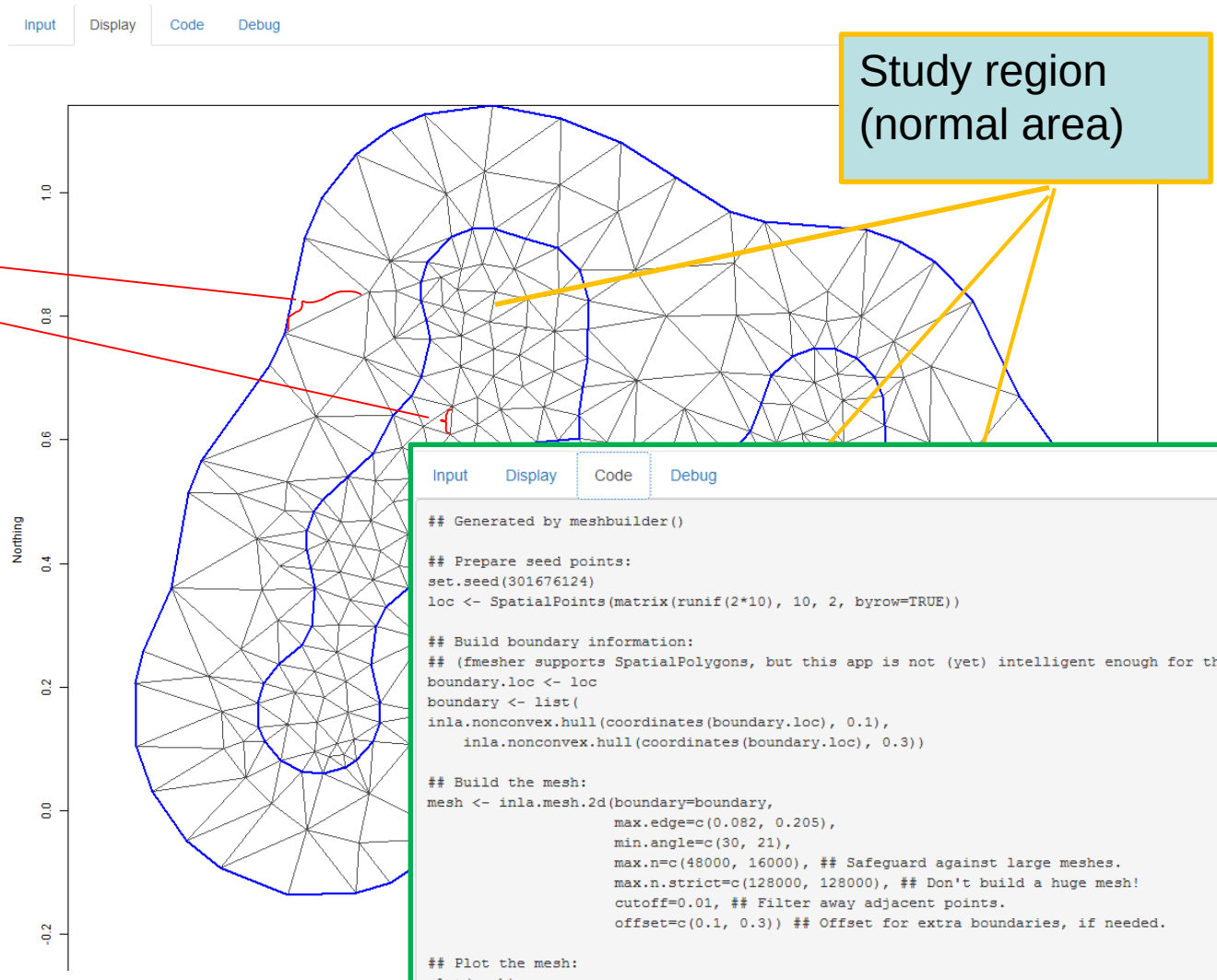
Source on Save

```cpp
// Illustration SPDE/INLA approach to s

#include <TMB.hpp>

template<class Type>
Type objective_function<Type>::operator
{
  using namespace R_inla;
  using namespace density;
  using namespace Eigen;

  DATA_VECTOR(y);
  DATA_SPARSE_MATRIX(A);
  DATA_SPARSE_MATRIX(A_pred);
  DATA_STRUCT(spde,spde_t);

  PARAMETER(beta0);
  PARAMETER(log_sigma_e);
  PARAMETER(log_tau);
  PARAMETER(log_kappa);
  PARAMETER_VECTOR(x);

  Type sigma_e = exp(log_sigma_e);
  Type tau = exp(log_tau);
  Type kappa = exp(log_kappa);

  Type nll = 0.0;

  SparseMatrix<Type> Q = Q_spde(spde,kappa);

  nll = GMRF(Q)(x);                                  // Negative log likeli

  vector<Type> mu = beta0+(A*x)/tau;
  nll -= dnorm(y,mu,sigma_e,true).sum();

  // Report section
  double nu = 1.0;              // nu = alpha-d/2 = 2-1 by eqn (2) in Lin
  Type rho = sqrt(8*nu)/kappa;  // Distance at which correlation has dr
  ADREPORT(rho);

  vector<Type> y_pred = beta0 + (A_pred*x)/tau;
  ADREPORT(y_pred);
```

Interface to R-INLA

INLA: Shiny app for mesh generation





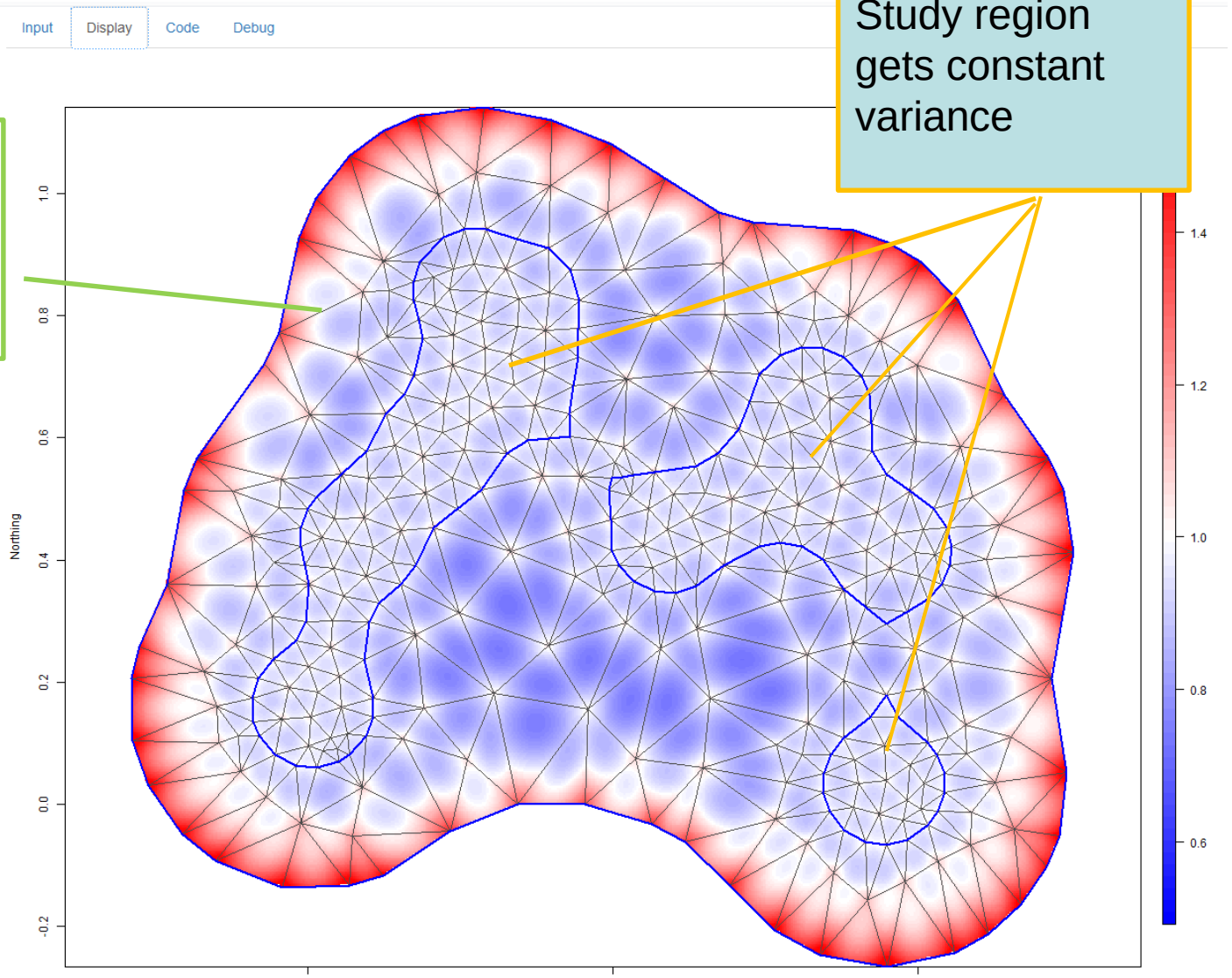
Input Display Code Debug

```
## Generated by meshbuilder()

## Prepare seed points:
set.seed(301676124)
loc <- SpatialPoints(matrix(runif(2*10), 10, 2, byrow=TRUE))

## Build boundary information:
## (fmesher supports SpatialPolygons, but this app is not (yet) intelligent enough for that.)
boundary.loc <- loc
boundary <- list(
inla.nonconvex.hull(coordinates(boundary.loc), 0.1),
    inla.nonconvex.hull(coordinates(boundary.loc), 0.3))

## Build the mesh:
mesh <- inla.mesh.2d(boundary=boundary,
                     max.edge=c(0.082, 0.205),
                     min.angle=c(30, 21),
                     max.n=c(48000, 16000), ## Safeguard against large meshes.
                     max.n.strict=c(128000, 128000), ## Don't build a huge mesh!
                     cutoff=0.01, ## Filter away adjacent points.
                     offset=c(0.1, 0.3)) ## Offset for extra boundaries, if needed.

## Plot the mesh:
plot(mesh)
```

Marginal variance (SD in figure)

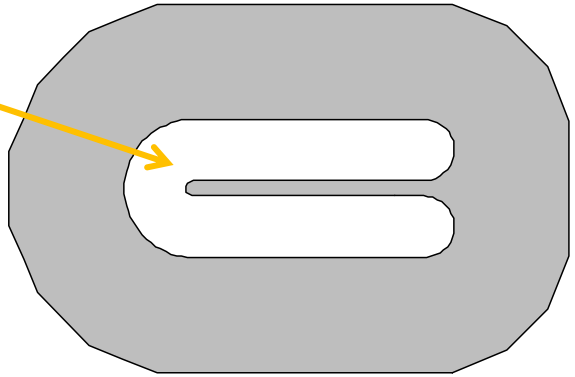
Boundary region gets **non-constant** variance

Study region gets constant variance

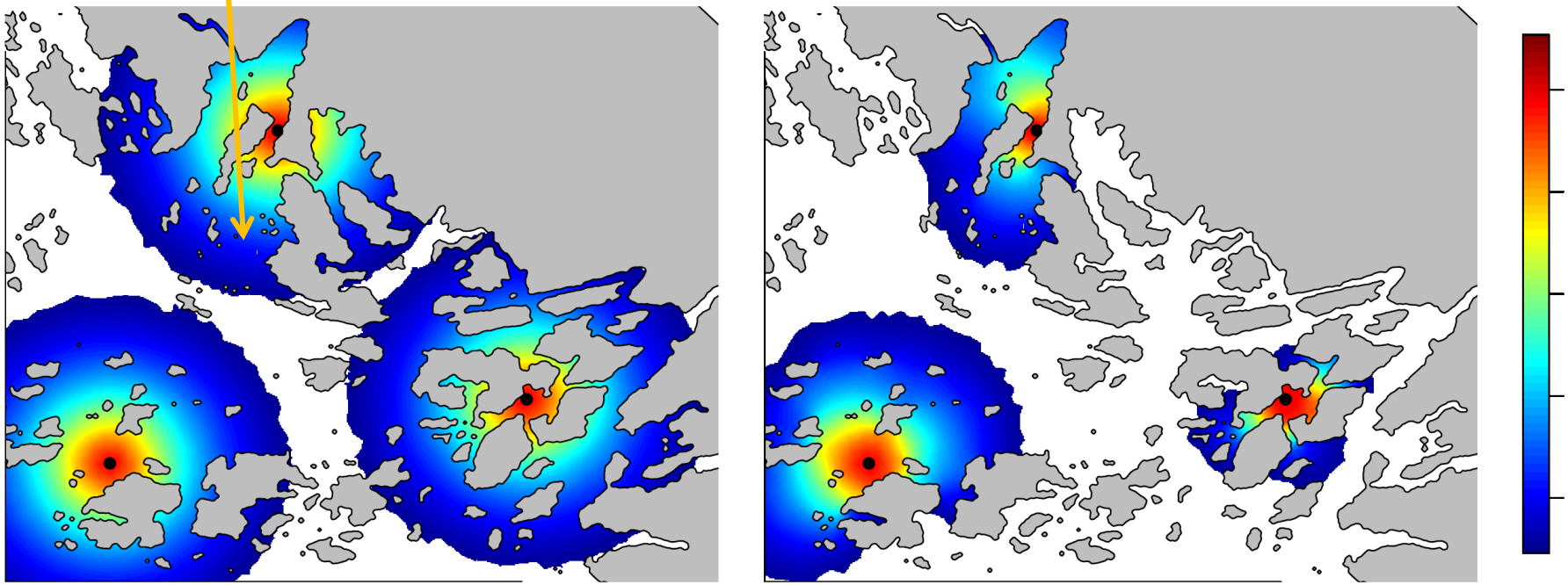


Barrier models

Horse shoe example: normal area and barrier (grey)

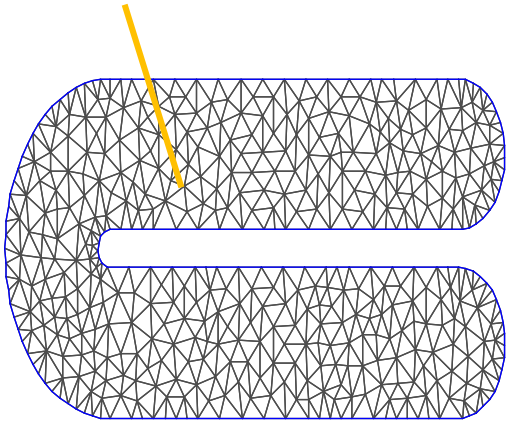


Fisheries data from Bakka et al (arxiv.org)

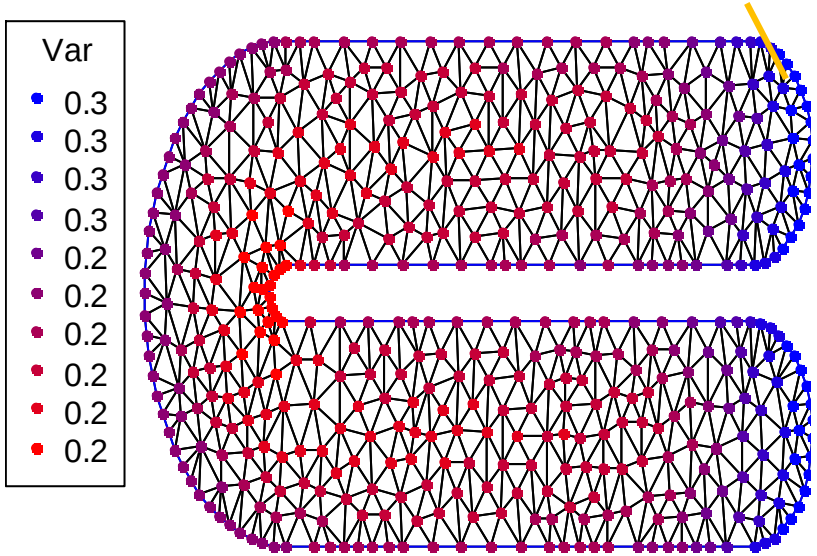


SPDE: boundary approach

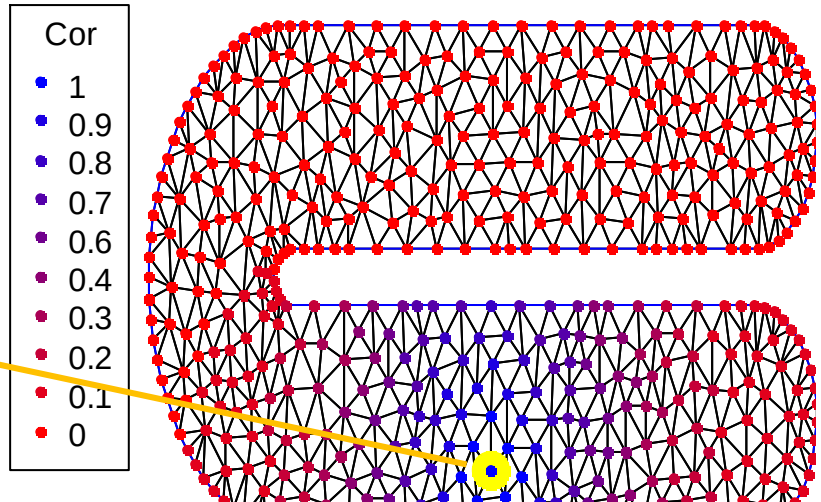
Study region with no boundary



Inflated variance

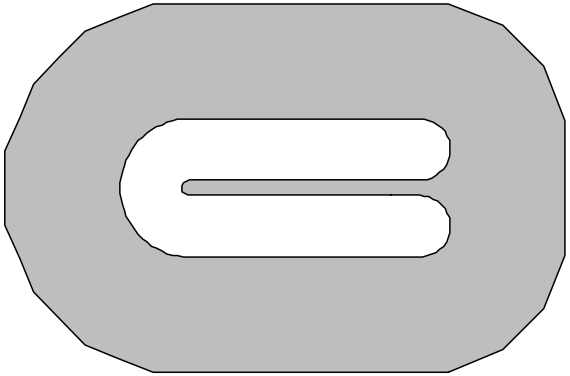


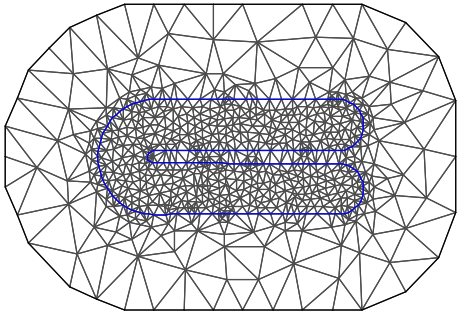
Correlation relative to this position



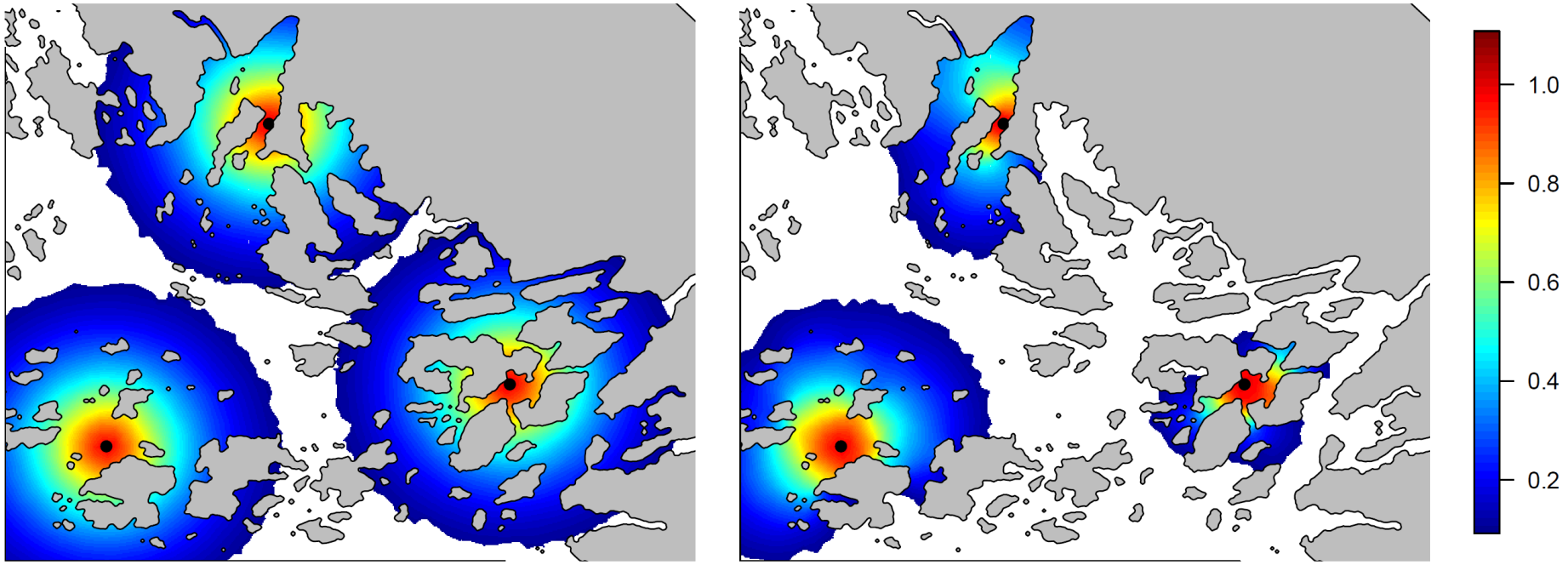
SPDE Barrier model by Bakka et al

The barrier region (in grey)





Results from Bakka model

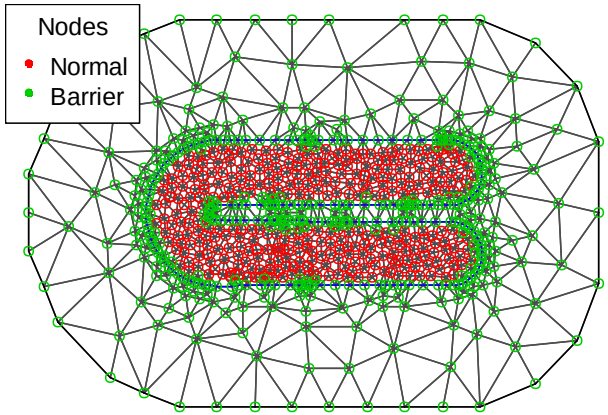


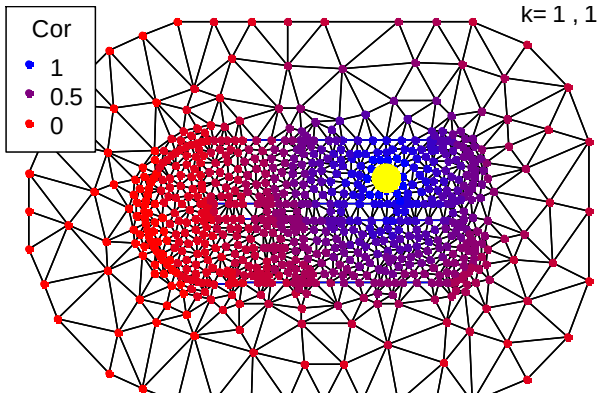
Direct implementation of (22) in Lindgren

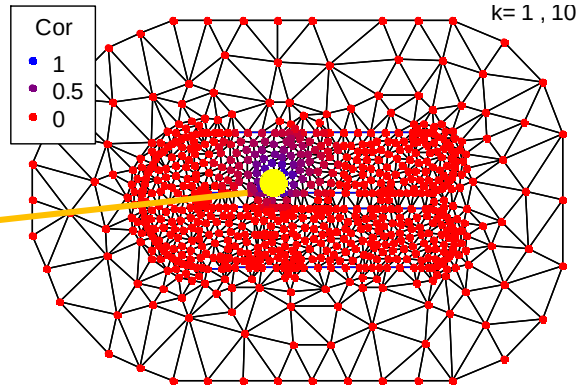
M0 M1 M2

$$Q_2(\kappa^2(\cdot), \tau(\cdot)) = \boldsymbol{\tau}\left(\boldsymbol{\kappa}^2\boldsymbol{C}\boldsymbol{\kappa}^2 + \boldsymbol{\kappa}^2\boldsymbol{G} + \boldsymbol{G}\boldsymbol{\kappa}^2 + \boldsymbol{G}\boldsymbol{C}^{-1}\boldsymbol{G}\right)\boldsymbol{\tau}$$

Constrained refined Delaunay triangulation

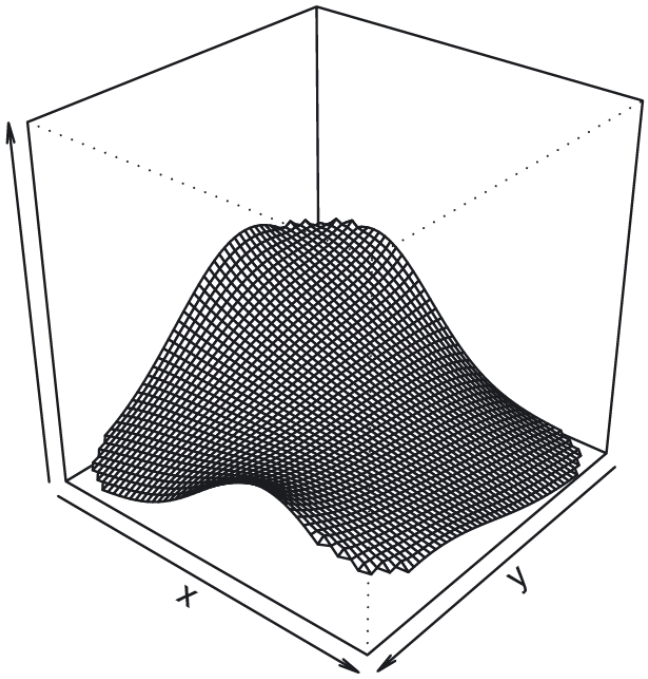






When I increase «kappa-barrier» the correlation range shrinks in normal region

Soap film smoother Wood et al (2008): implied varince / covariance



Soap film smoother in TMB

```r
knots <- expand.grid(v=seq(-.5,3,by=1),w=c(-.6,-.3,.3,.6))

b <- gam(y~s(v,w,k=30,bs="so",xt=list(bnd=fsb)),knots=knots, method="REML")
data_tmb = list(y=y,                          # Response
                S1=b$smooth[[1]]$S[[1]],      # Penlty matrix for boundary of horse shoe
                S2=b$smooth[[1]]$S[[2]],      # Penlty matrix for internal of horse shoe
                X=model.matrix(b)[, -1])      # X*beta is the smoother

par = list(mu=0,
           beta=rep(0, length(coef(b))-1),
           log_lambda=log(b$sp),#c(0, 0),
           log_sigma=0)

compile("mgcv_soap.cpp")
dyn.load(dynlib("mgcv_soap"))
model <- MakeADFun(data=data_tmb,parameters=par,ra
opt <- nlminb(model$par, model$fn, model$gr,lower=
rep = sdreport(model)

# Sets up the prediction grid
mm<-100;nn<-50#mm<-300;nn<-150
xm <- seq(-1,4,length=mm);yn<-seq(-1,1,length=nn)
xx <- rep(xm,nn);yy<-rep(yn,rep(mm,nn))
Lp <- predict(b, newdata=data.frame(v=xx, w=yy), t
pred_tmb <- Lp%*%summary(rep)[, 1][-(2:4)]
```

Setting up the smoothing using `mgcv`

```r
template<class Type>
Type objective_function<Type>::operator() ()
{

  DATA_VECTOR(y);
  DATA_MATRIX(S1);
  DATA_MATRIX(S2);
  DATA_MATRIX(X);

  PARAMETER(mu);
  PARAMETER_VECTOR(beta);
  PARAMETER_VECTOR(log_lambda);
  PARAMETER(log_sigma);

  vector<Type> lambda = exp(log_lambda);
  Type sigma = exp(log_sigma);

  Type nll=0;

  matrix<Type> S = lambda(0)*S1+lambda(1)*S2;

  using namespace atomic;

  nll -= 0.5*logdet(S) - 0.5*(beta* vector<Type>(S*beta)).sum()

  vector<Type> eta = mu + X*beta;
  nll -= dnorm(y,eta,sigma,true).sum();

  return nll;
```

Soap film smoother in TMB

```r
knots <- expand.grid(v=seq(-.5,3,by=1),w=c(-.6,-.3,.3,.6))

b <- gam(y~s(v,w,k=30,bs="so",xt=list(bnd=fsb)),knots=knots, method="REML")
data_tmb = list(y=y,                          # Response
                S1=b$smooth[[1]]$S[[1]],      # Penlty matrix for boundary of horse shoe
                S2=b$smooth[[1]]$S[[2]],      # Penlty matrix for internal of horse shoe
                X=model.matrix(b)[, -1])      # X*beta is the smoother

par = list(mu=0,
           beta=rep(0, length(coef(b))-1),
           log_lambda=log(b$sp),#c(0, 0),
           log_sigma=0)

compile("mgcv_soap.cpp")
dyn.load(dynlib("mgcv_soap"))
model <- MakeADFun(data=data_tmb,parameters=par,ra
opt <- nlminb(model$par, model$fn, model$gr,lower=
rep = sdreport(model)

# Sets up the prediction grid
mm<-100;nn<-50#mm<-300;nn<-150
xm <- seq(-1,4,length=mm);yn<-seq(-1,1,length=nn)
xx <- rep(xm,nn);yy<-rep(yn,rep(mm,nn))
Lp <- predict(b, newdata=data.frame(v=xx, w=yy), t
pred_tmb <- Lp%*%summary(rep)[, 1][-(2:4)]
```

```r
template<class Type>
Type objective_function<Type>::operator() ()
{

  DATA_VECTOR(y);
  DATA_MATRIX(S1);
  DATA_MATRIX(S2);
  DATA_MATRIX(X);

  PARAMETER(mu);
  PARAMETER_VECTOR(beta);
  PARAMETER_VECTOR(log_lambda);
  PARAMETER(log_sigma);

  vector<Type> lambda = exp(log_lambda);
  Type sigma = exp(log_sigma);

  Type nll=0;

  matrix<Type> S = lambda(0)*S1+lambda(1)*S2;

  using namespace atomic;

  nll -= 0.5*logdet(S) - 0.5*(beta* vector<Type>(S*beta)).sum()

  vector<Type> eta = mu + X*beta;
  nll -= dnorm(y,eta,sigma,true).sum();

  return nll;
```

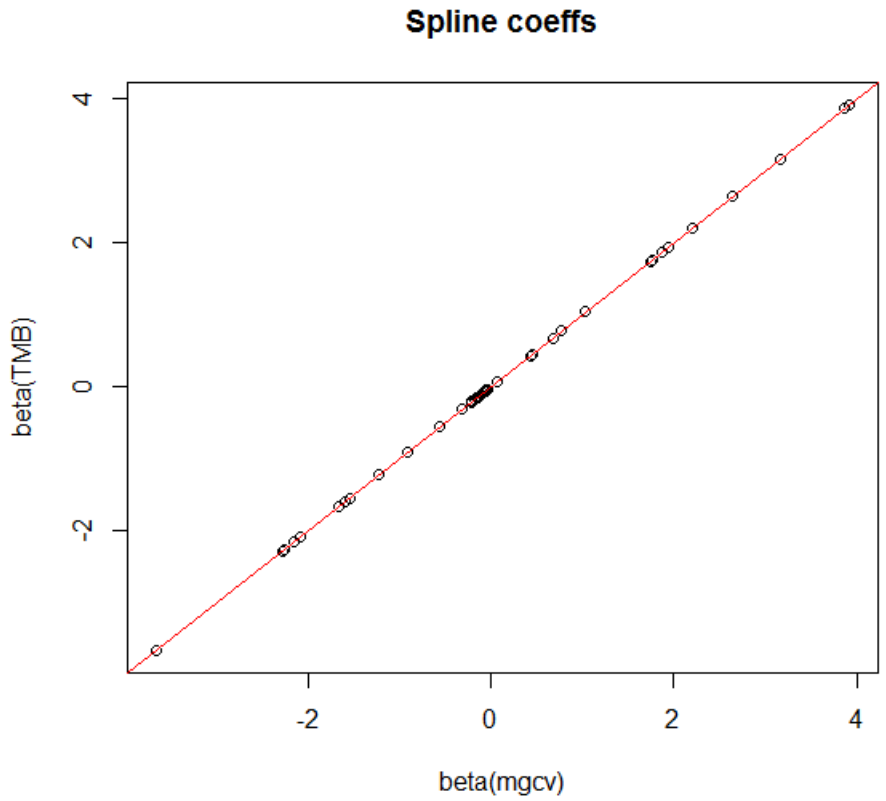
Penalty matrices

Soap film smoother in TMB

```r
knots <- expand.grid(v=seq(-.5,3,by=1),w=c(-.6,-.3,.3,.6))

b <- gam(y~s(v,w,k=30,bs="so",xt=list(bnd=fsb)),knots=knots, method="REML")
data_tmb = list(y=y,                          # Response
                S1=b$smooth[[1]]$S[[1]],  # Penlty matrix for boundary of horse shoe
                S2=b$smooth[[1]]$S[[2]],  # Penlty matrix for internal of horse shoe
                X=model.matrix(b)[, -1])  # X*beta is the smoother

par = list(mu=0,
           beta=rep(0, length(coef(b))-1),
           log_lambda=log(b$sp),#c(0, 0),
           log_sigma=0)

compile("mgcv_soap.cpp")
dyn.load(dynlib("mgcv_soap"))
model <- MakeADFun(data=data_tmb,parameters=par,ra
opt <- nlminb(model$par, model$fn, model$gr,lower=
rep = sdreport(model)

# Sets up the prediction grid
mm<-100;nn<-50#mm<-300;nn<-150
xm <- seq(-1,4,length=mm);yn<-seq(-1,1,length=nn)
xx <- rep(xm,nn);yy<-rep(yn,rep(mm,nn))
Lp <- predict(b, newdata=data.frame(v=xx, w=yy), t
pred_tmb <- Lp%*%summary(rep)[, 1][-(2:4)]
```

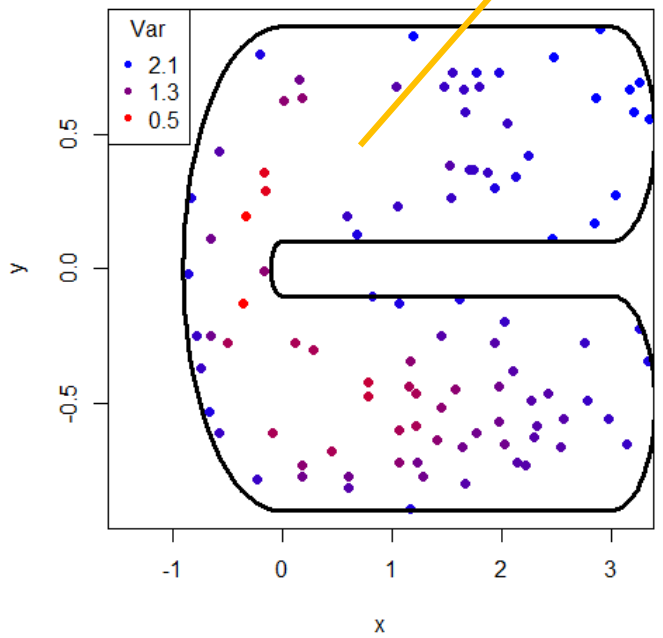
```r
template<class Type>
Type objective_function<Type>::operator() ()
{

  DATA_VECTOR(y);
  DATA_MATRIX(S1);
  DATA_MATRIX(S2);
  DATA_MATRIX(X);

  PARAMETER(mu);
  PARAMETER_VECTOR(beta);
  PARAMETER_VECTOR(log_lambda);
  PARAMETER(log_sigma);

  vector<Type> lambda = exp(log_lambda);
  Type sigma = exp(log_sigma);

  Type nll=0;

  matrix<Type> S = lambda(0)*S1+lambda(1)*S2;

  using namespace atomic;

  nll -= 0.5*logdet(S) - 0.5*(beta* vector<Type>(S*beta)).sum()

  vector<Type> eta = mu + X*beta;
  nll -= dnorm(y,eta,sigma,true).sum();

  return nll;
```

TMB gives exactly the same estimates of the spline coefficient



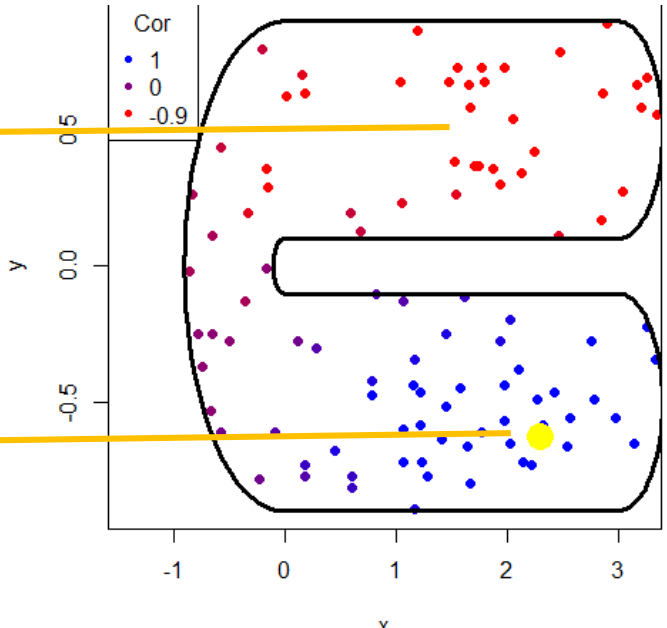
Soap film smoother: implied varince / covariance

Non-constant variance



Negative correlation

Corelation relative to this position



Summary and conclusion

- It is “easy” to import spatial smoothers into TMB
  - INLA: great mesh generation tools
  - `mgcv`: a large variety of spline smoothers
- Barrier model:
  - Direct implementation based on (22) in Lindgreen does not work properly.

Summary and conclusion

- The soap smoother “implies” a spatial covariance matrix
  - Natural “metric” for comparison with covariance based methods
  - Negative long range spatial correlation not yet understood (have I done something wrong???)
- Norwegian mackerel data:
  - Zero-inflated model with spatial components in both p(empty trawl) and in p(y | y>0)
  - Spatio-temporal model

Some references

Papers

Kristensen et al (2016) TMB: Automatic differentiation and Laplace approximation. Journal of Statistical Software. **70** (5)

Bakka, Haakon et al (2016) Accounting for physical barriers in species distribution modeling with non-stationary spatial random effects. arxiv.org

Wood S., Bravington M., Hedley S. (2008) Soap film smoothing, JRSS-B

Main TMB reference