The Quick Start Guide to Spatial Modelling

Anders Nielsen, Ethan Lawler, & Sean Anderson

an@aqua.dtu.dk

Reasons to use spatial modelling

- Spatial modelling is relatively easy these days
- Residuals from a previous model show spatial pattern
- Simple data visualization method for complicated data
- Good results even without environmental covariates
- Predictions can be made for locations where you have no data

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

- Some small points...
  - If areas are very small relative to study area, areal data can be treated as geostatistical data
  - If geostatistical data are aggregated by zone, they become areal data
  - The conceptual difference between geostatistical data and point process data can be unclear

- We will focus on geostatistical modelling

Correlation matrices and functions

- Reminder: In the fishing mortality example, we assumed similar age groups had similar fishing mortality
  - "Similar fishing mortality" meant fishing mortality at ages 1, 2, 3, ... followed a multivariate normal distribution

$$\log F_{y,a} \sim \log F_{y-1,a} + \psi_{y,a}, \quad \text{where} \quad \psi_{y,a} \sim \mathrm{N}\left(0,\ \Sigma\right)$$

$$\mathrm{Cor}\left[\psi_{y,a_i},\ \psi_{y,a_j}\right] = \rho_{i,j} = \frac{\Sigma_{ij}}{\sqrt{\Sigma_{ii}\Sigma_{jj}}}$$

- We will 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

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

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

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

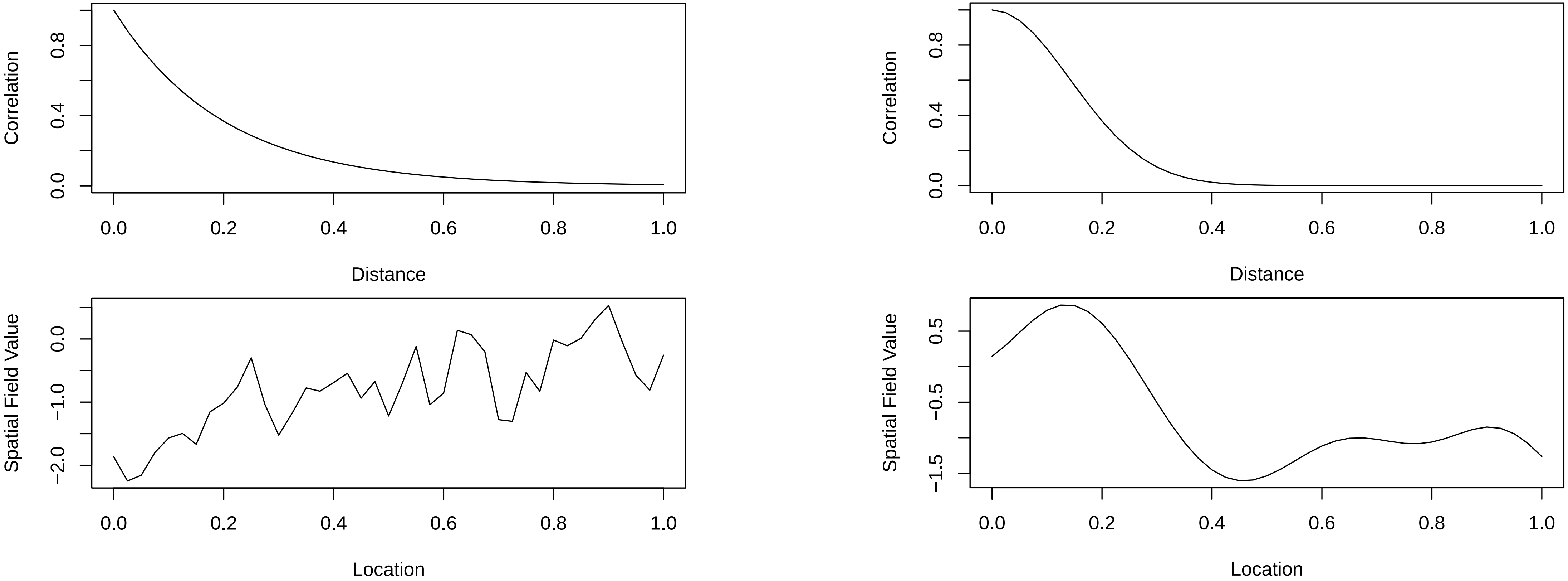
Two common correlation functions

- Exponential
- Noisy looking fields

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

- Gaussian
- Very smooth looking fields

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



Exercise: Direct Spatial Modelling

Implement the following spatial model to analyse the data in spatial_data.RData:

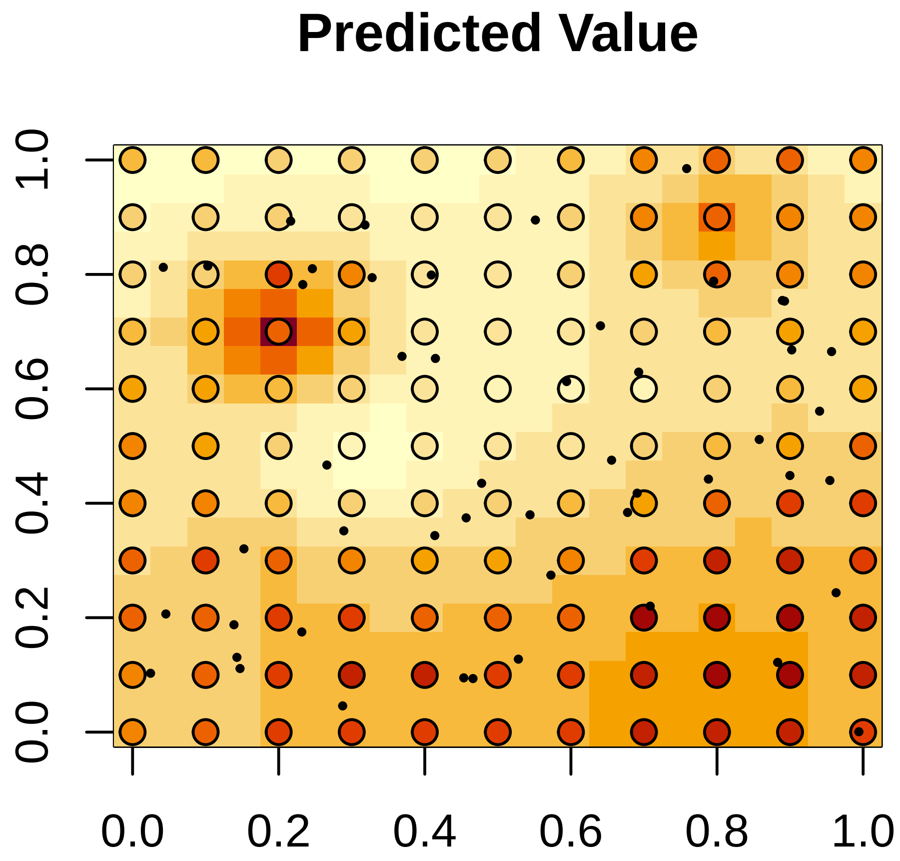
```
      value       lon        lat
1 10.726830 0.2875775 0.04583117
2  9.992624 0.7883051 0.44220007
3  8.816444 0.4089769 0.79892485
4 11.872133 0.8830174 0.12189926
```

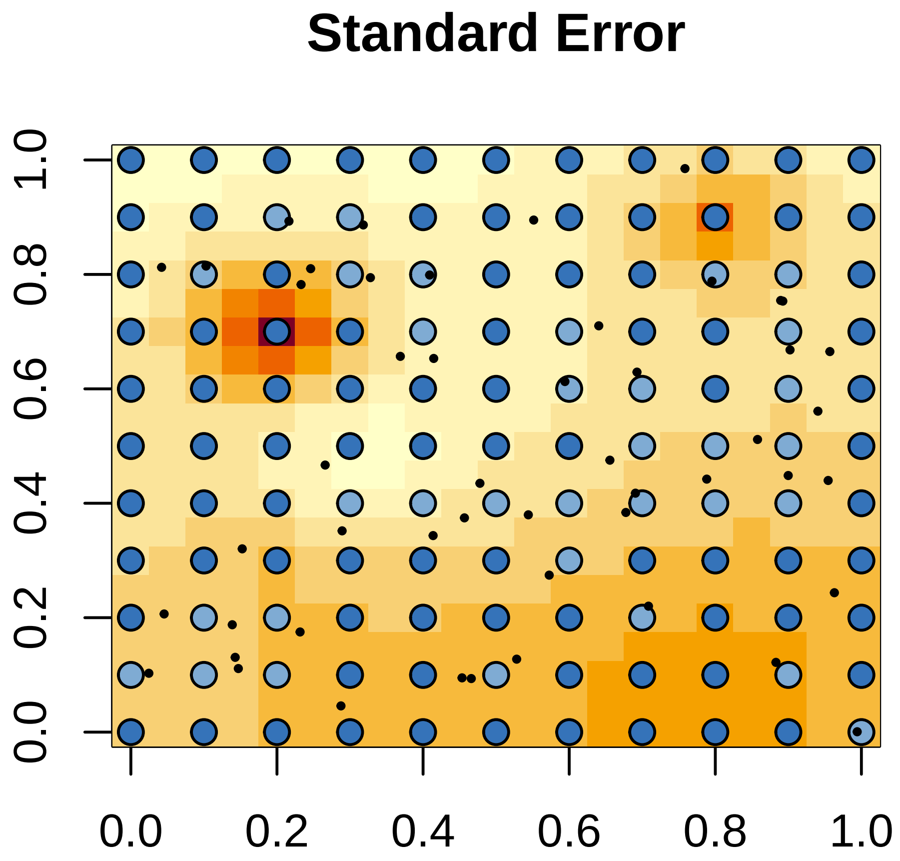
$$Y_i = X_{\mathbf{s}_i} + \epsilon_i, \qquad \epsilon_i \sim \mathsf{N}\left(0, \sigma^2\right)$$

$$\begin{bmatrix} X_{\mathbf{s}_1} \\ X_{\mathbf{s}_2} \\ \vdots \\ X_{\mathbf{s}_n} \end{bmatrix} \sim \mathsf{N}\left(\mu, \Sigma\right), \qquad \Sigma_{ij} = \sigma_x^2 \mathcal{C}\left(d_{ij}\right)$$

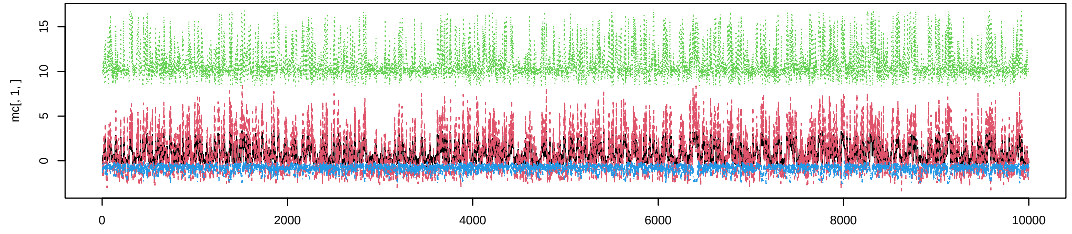
1. Use the exponential correlation function - ρ must be positive; $d_{ij} = \sqrt{\left(\mathbf{s}_{i,1} - \mathbf{s}_{j,1}\right)^2 + \left(\mathbf{s}_{i,2} - \mathbf{s}_{j,2}\right)^2}$
2. We've implemented very similar models in mvnorm/ar1.R with code = 2
3. What do the 95% confidence intervals for σ_x^2 and ρ look like?
4. Test what happens if you replace $\mathbf{s}_2$ with $\mathbf{s}_1$, so that Y_1 and Y_2 are observed at the same location. How might you prevent the error that occurs?
5. How can you predict the spatial field X at the prediction_locations?

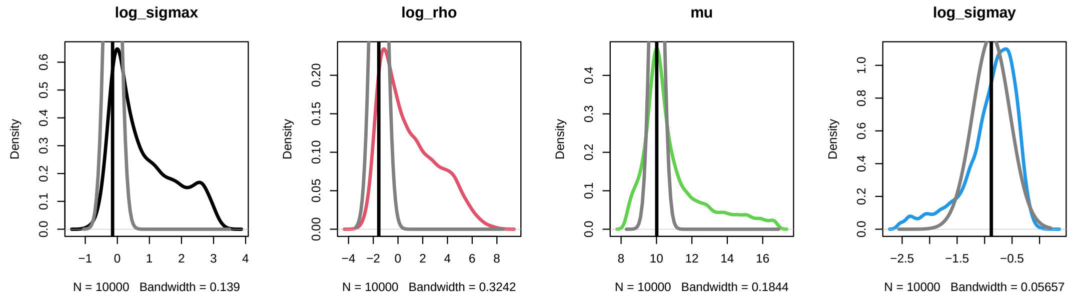
Predictions from spatial model





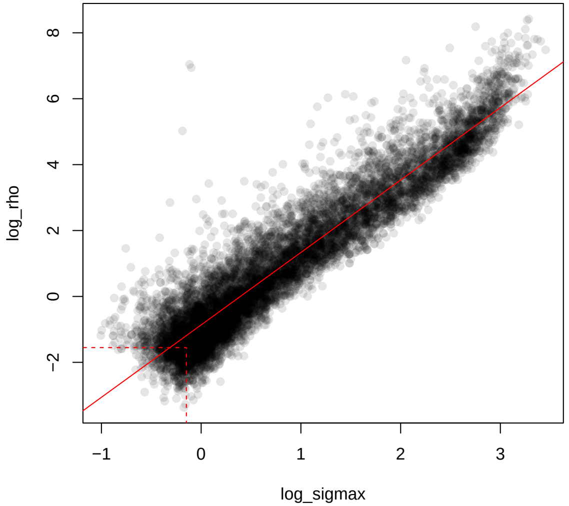
Exploring the likelihood with MCMC and tmbstan - the UGLY



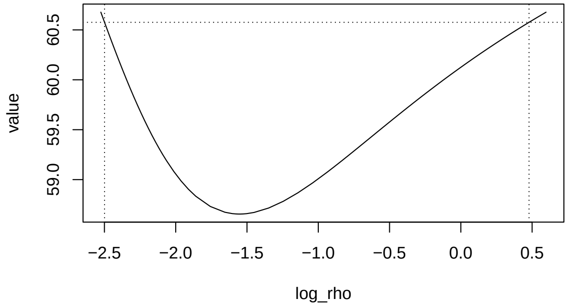


- Also take a look at spatial_uncertainty.R
- Problems can't be easily diagnosed with normal tools

Scatterplot from MCMC



Profile likelihood for $\log(\rho)$ from tmbprofile



Some words of caution

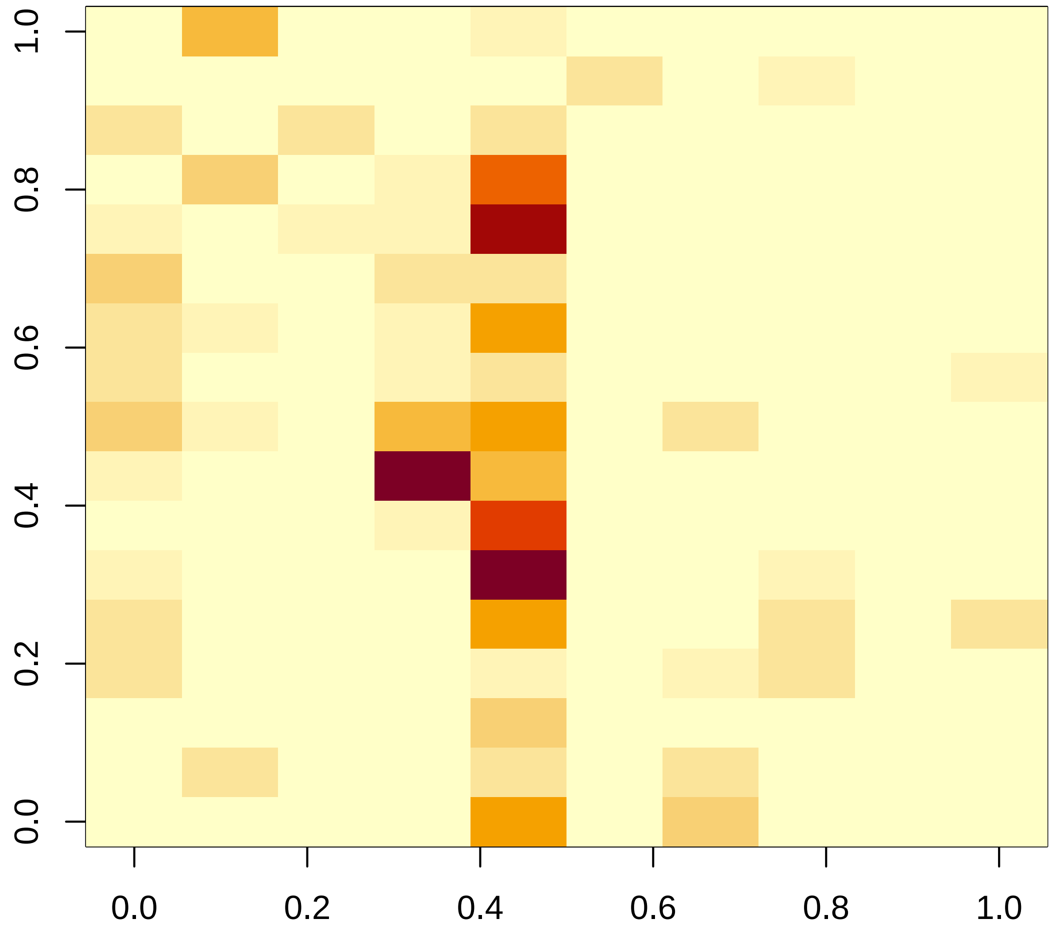
- Don't try to interpret σ_x^2 and ρ
  - Estimators of σ_x^2 and ρ aren't well behaved (see above)
  - Log-likelihood has a "ridge" along which it is approximately constant
  - Predictions of random effects are more or less equivalent if σ_x^2 and ρ move along the ridge
  - Estimation can be improved by fixing ρ or penalizing the likelihood, but then you're just choosing which values to use and they shouldn't be interpreted as true values
- The Gaussian correlation function can give NaN likelihoods if locations are close together
  - Solution: Remove a small bit of correlation from the off-diagonal elements

$$\tilde{\mathsf{COR}} := \mathsf{COR} - 0.999\,(\mathsf{COR} - I)$$

- Direct spatial modelling becomes very slow even with moderately many locations
  - Solution: use sparse approximations

Raster data

- Raster data can be treated as geostatistical data if the cell size is small compared to the whole area
- Data are collected along a regular grid, so the locations are very structured
- The structure lets us use dautoreg and dseparable to evaluate the likelihood



A special case: raster data with dseparable

```
library(RTMB)
load("matrix_ar1.RData")

nll<- function(pars) {
  getAll(pars)
  rho<- 2 * plogis(working_rho) - 1
  sigma<- exp(log_sigma)

  ar1_within_rows<- function(x) dautoreg(
    x,
    phi = rho[1],
    log = TRUE,
    scale = sigma[1]
  )
  ar1_within_cols<- function(x) dautoreg(
    x,
    phi = rho[2],
    log = TRUE,
    scale = sigma[2]
  )
  dmatrix_ar1<- dseparable(ar1_rows, ar1_cols)
  x %~% dmatrix_ar1()

  # Can also scale x directly by doing...
  # ar1_rows<- function(x) dautoreg(x, phi = rho[1], log = TRUE)
  # ar1_cols<- function(x) dautoreg(x, phi = rho[2], log = TRUE)
  # dmatrix_ar1<- dseparable(ar1_along_rows, ar1_along_columns)
  # x %~% dmatrix_ar1(scale = sigma[1] * sigma[2])
```

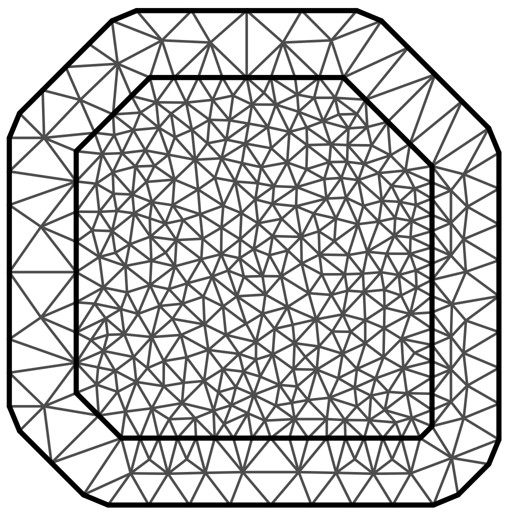
files/separable.R

```
  # You can input a scaling matrix...
  # scale_matrix<- matrix(sigma[1] * sigma[2], nrow(x), ncol(x))
  # x %~% dmatrix_ar1(scale = scale_matrix)

  # Can write the negative likelihood directly with...
  # nll<- -dseparable(ar1_along_rows, ar1_along_columns)(x)

  y %~% dpois(exp(mu + x))
}
pars<- list(
  mu = 0,
  working_rho = qlogis(0.5 * c(0.1, 0.9) + 0.5),
  log_sigma = log(c(0.3, 5)),
  x = matrix(
    0,
    nrow(y),
    ncol(y)
  )
)
obj<- MakeADFun(nll, pars, random = "x")
opt<- nlminb(obj$par, obj$fn, obj$gr)
```

files/separable.R

Steps in sparse spatial modelling using dgmrf

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
2. Compute likelihood
    - Compute precision matrix from helper objects
    - Evaluate built-in dgmrf density
    - Interpolate spatial field for observations and predictions



Create helper objects

```
load("spatial_data.RData")

mesh <- fmesher::fm_mesh_2d(
  loc = as.matrix(data[, c("lon", "lat")]),
  min.angle = 24,
  max.edge = c(0.1, 0.3),
  cutoff = 0.05
)
spde <- fmesher::fm_fem(mesh)

interpolator_data <- fmesher::fm_basis(
  mesh,
  loc = as.matrix(data[, c("lon", "lat")])
)

interpolator_prediction <- fmesher::fm_basis(
  mesh,
  loc = as.matrix(prediction_locations)
)

pdf("mesh.pdf", width = 4, height = 4)
plot(mesh)
dev.off()

save(mesh, spde, interpolator_data, interpolator_prediction, file = "spatial_helpers.RData")
```

files/spatial_helpers.R

Compute likelihood

```r
library(RTMB)
load("spatial_data.RData") # gets data, prediction_locations,
     raster
load("spatial_helpers.RData") # gets mesh, spde, interpolators

mesh_info<- list(
  mesh = mesh,
  spde = spde,
  interpolator_data = interpolator_data,
  interpolator_prediction = interpolator_prediction
)

nll <- function(par) {
  getAll(par, data, mesh_info)
  tau <- exp(log_tau)
  kappa <- exp(log_kappa)
  # Computation of precision matrix Q here is a standard line
  # of code that you can copy/paste into your own code
  Q <- tau * (kappa^4 * spde$c0 + 2 * kappa^2 * spde$g1 + spde$g2)
  x %~% dgmrf(0, Q)

  data_predictions <- mu + interpolator_data %*% x
  obs_sd <- exp(log_obs_sd)
  value %~% dnorm(data_predictions, obs_sd)

  predictions <- mu + interpolator_prediction %*% x
  ADREPORT(predictions)
}
```

files/spatial_gmrf.R

```r
par <- list(
  log_tau = 0,
  log_kappa = 0,
  x = numeric(mesh$n),
  mu = 0,
  log_obs_sd = 0
)
obj <- MakeADFun(nll, par, random = "x")
fit <- nlminb(obj$par, obj$fn, obj$gr)
sdrep <- sdreport(obj)
pl <- as.list(sdrep, "Estimate", report = TRUE)
plsd <- as.list(sdrep, "Std. Error", report = TRUE)

est_field <- matrix(pl$predictions, nrow = nrow(raster))
sd_field <- matrix(plsd$predictions, nrow = nrow(raster))

par(mfrow = c(3, 1))
image(raster)
image(est_field)
image(sd_field)
```

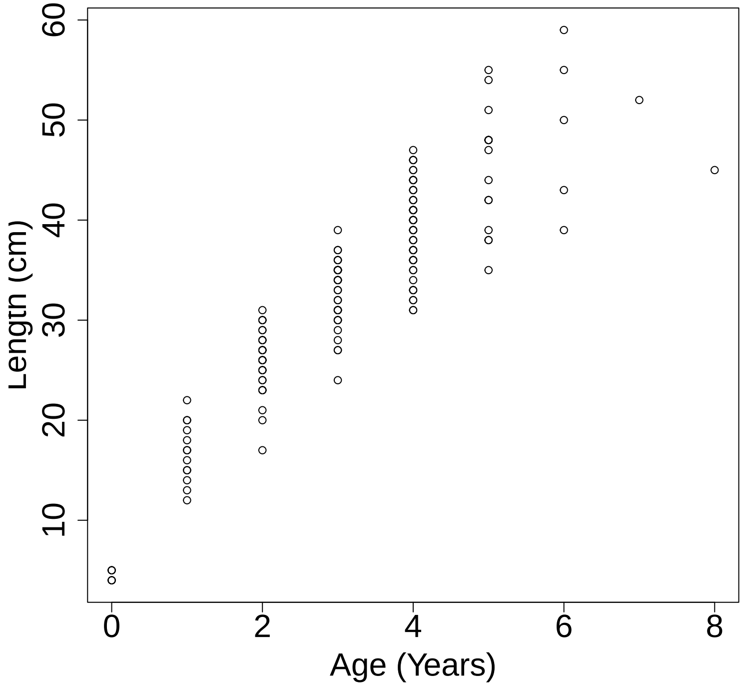
files/spatial_gmrf.R

Investigating spatial von Bertalanffy growth curve

$$\log L = \log L_\infty + \log\left(1 - e^{-r(a-t_0)}\right) + \epsilon$$

- L: Length of individual
- a: Age of individual
- Young cod morphometrics from 2008 4VSW survey

	length	age	lat	lon
1	12	1	44.0515	-59.95950
2	24	2	44.2855	-59.04325
3	32	3	44.2855	-59.04325
4	22	1	44.2855	-59.04325
5	47	5	44.2855	-59.04325
6	50	6	44.2855	-59.04325



https://open.canada.ca/data/en/dataset/a851ce30-e216-4d7d-a29c-05631eef140e

Non-spatial von Bertalanffy code

```
library(RTMB)
load("age_length_data.RData")
cod$log_length<- log(cod$length)

par<- list(
  log_L_inf = log(max(cod$length)),
  log_rate = 0,
  nlog_t0 = 0,
  log_obs_sd = 0
)
nll<- function(par) {
  getAll(par, cod)
  L_inf<- exp(log_L_inf)
  rate<- exp(log_rate)
  t0<- -exp(nlog_t0)
  pred<- log(L_inf) + log(1 - exp(-rate * (age - t0)))
  ADREPORT(pred)

  obs_sd<- exp(log_obs_sd)
  ADREPORT(obs_sd)
  log_length<- OBS(log_length)
  log_length %~% dnorm(
    pred,
    obs_sd
  )
}
obj<- MakeADFun(nll, par)
non_spatial_fit<- nlminb(obj$par, obj$fn, obj$gr)
```

files/age_length.R

Exercise

- Modify the code above to allow growth rate to be a spatial random effect
- Plot maps of the lower 95% CI, mean, and upper 95% CI of the predicted growth rate
- Use a likelihood ratio test to determine if we can get away with a non-spatial model
- Hints:
  - Growth rate still needs to be positive at every location
  - Helper objects can be found in age_length_helpers.RData
  - If you want to make your own mesh, try using...
    - min.angle = 24
    - max.edge = c(0.5, 1.0)
    - cutoff = 0.2
- Bonus:
  - Some prediction locations might be outside the mesh...
    - How could you identify these?
    - What should you do about them?