Random effects

Olav Nikolai Risdal Breivik

Thanks to Anders Nielsen for borrowing course material

Norsk Regnesentral
ANVENDT DATAFORSKNING

Random effect model

For *fixed effect models* we have

- Random variables we observe (the response)
- Model parameters we want to estimate

For *random effect models* we have

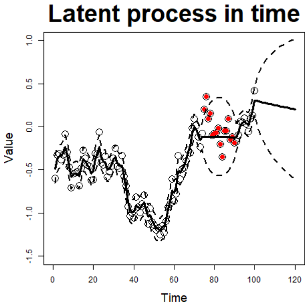
- Random variables we observe (the response)
- Random variables we do **NOT** observe
- Model parameters we want to estimate

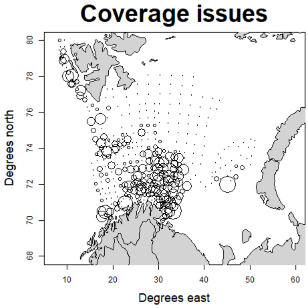
This model class is very usefull and goes under many names:

- state space models
- latent variable models
- hidden Markov models
- random effect models
- latent hierarchical models
- mixed models
- frailty models

Examples of use

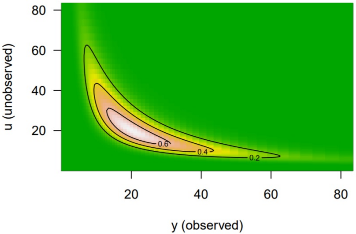
- Time series models via state space models
  - Correlation via hidden processes
- Spatial statistics
  - Correlation introduces with hidden field
- Something unobserved gives extra variation
  - Overdispersion in Poisson via negative binomial
- Repeated experiment
  - Correlation within subject





Latent random effects

- Assume we have:
  Observations: y
  NOT observed random effects: u
  Parameters (θ) in the model: $(y, u) \sim D(\theta)$



- How do we estimate our parameters when some of our random variables are not observed?

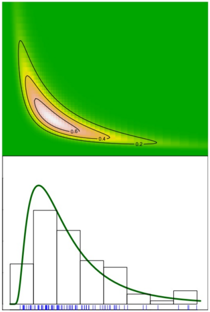
Latent random effects

1. Joint model (banana) is determined by model parameters (θ)
2. Marginal model is calculated from joint by integration
3. Marginal is matched to data (what we observe)

The marginal likelihood is:

$$L_M(\theta, y) = \int L(\theta, u, y) du.$$

Match your data with the marginal likelihood

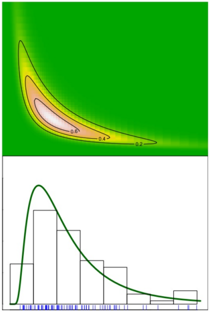


Latent random effects

- The marginal likelihood is:

$$L_M(\theta, y) = \int_{\mathbb{R}^q} L(\theta, u, y) du$$

  - θ is model parameters
  - u is the NOT observed random values
  - y is the observed random values (the observations)



- How to calculate the integral?
  - Numerical integration not practical (need a loot of integration points)
  - Seldom an analytical solution
  - Solution: Approximate with use of Taylor-approximation

Latent random effects

We need to approximate the difficult integral

$$L_M(\theta, y) = \int_{\mathbb{R}^q} L(\theta, u, y) du.$$

Solution:

- Let $\ell(\theta, u, y) = \log L(\theta, u, y)$. Note that

  $$\ell(\theta, u, y) \approx \ell(\theta, \hat{u}_\theta, y) - \frac{1}{2}(u - \hat{u}_\theta)^t(-\ell''_{uu}|_{u=\hat{u}_\theta})(u - \hat{u}_\theta)$$

  is a a 2.order Taylor approximation around

  $$\hat{u}_\theta = \underset{u}{\operatorname{argmax}}\, \ell(\theta, u, y)$$

  for a given θ.

Thereby is:

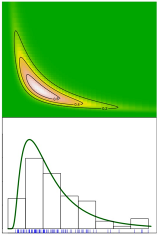
$$
\begin{aligned}
L_M(\theta, y) &= \int_{\mathbb{R}^q} L(\theta, u, y) du \\
&= \int_{\mathbb{R}^q} \exp\Big(\ell(\theta, u, y)\Big) du \\
&\approx \int_{\mathbb{R}^q} \exp\Big(\ell(\theta, \hat{u}_\theta, y) - \frac{1}{2}(u - \hat{u}_\theta)^t(-\ell''_{uu}|_{u=\hat{u}_\theta})(u - \hat{u}_\theta)\Big) du \\
&= L(\theta, \hat{u}_\theta, y) \int_{\mathbb{R}^q} \exp\Big(- \frac{1}{2}(u - \hat{u}_\theta)^t(-\ell''_{uu}|_{u=\hat{u}_\theta})(u - \hat{u}_\theta)\Big) du \\
&= L(\theta, \hat{u}_\theta, y) \sqrt{\frac{(2\pi)^q}{|-\ell''_{uu}|_{u=\hat{u}_\theta}|}}
\end{aligned}
$$

The last step is obtained by observing that the integrand has a Gaussian shape.

Taking the logarithm gives:

$$\ell_M(\theta, y) \approx \ell(\theta, \hat{u}_\theta, y) - \frac{1}{2}\log(|-\ell''_{uu}|_{u=\hat{u}_\theta}|) + \frac{q}{2}\log(2\pi)$$

- This is the Laplace approximation
- Mini exercises:
  - Where do we utilize automatic differentiation here?
  - Why care about conditional independence?



Syntax

- Very simple syntax:

```
obj <- MakeADFun(nll, par, random = c("parName1","parName2"))
```

- The parameters *parName1* and *parName2* are now integrated over with the Laplace approximation
- Just as before:

```
obj$fn()#Marginal likelihood
obj$gn()#Gradient of marginal likelihood
```

Gradient based search

Fixed effect model:

```
opt  = nlminb(obj$par,obj$fn,obj$gr, control = list(trace = 1))
outer mgc:  103.5774
0:     398.42982:  0.00000  0.00000
outer mgc:  112.5499
1:     281.66042: -0.465673 0.884957
outer mgc:  68.03514
2:     205.84484: -0.622067  2.87883
outer mgc:  45.85355
3:     202.84258: -0.821876  2.87009
outer mgc:  1.537242
4:     200.74572: -0.753957  2.68198
outer mgc:  0.6074853
5:     200.68054: -0.753045  2.55830
outer mgc:  0.07071213
6:     200.67113: -0.752270  2.59332
outer mgc:  0.002883933
7:     200.67099: -0.752283  2.58967
outer mgc:  8.665257e-05
8:     200.67099: -0.752282  2.58952
outer mgc:  5.93635e-06
9:     200.67099: -0.752282  2.58952
```

- `mgc` is maximum gradient component

Gradient based search

Random effect model:

```
> opt = nlminb(obj$par, obj$fn, obj$gr, control = list(trace = 1))
...
1:     867.24500: 0.132409 0.991195
iter: 1  value: 848.5753 mgc: 7.724378 ustep: 1
iter: 2  value: 847.7413 mgc: 0.5669531 ustep: 1
iter: 3  value: 847.741 mgc: 0.0157684 ustep: 1
iter: 4  value: 847.741 mgc: 1.253356e-05 ustep: 1
iter: 5  mgc: 8.010481e-12
iter: 1  mgc: 8.010481e-12
outer mgc:  100.4661
2:     847.59842: -0.726562  1.50322
iter: 1  value: 865.6699 mgc: 1.774005 ustep: 1
iter: 2  value: 865.4707 mgc: 0.1210295 ustep: 1
iter: 3  value: 865.4705 mgc: 0.002688139 ustep: 1
iter: 4  value: 865.4705 mgc: 4.835442e-06 ustep: 1
iter: 5  mgc: 1.364867e-11
iter: 1  mgc: 1.364867e-11
outer mgc:  25.57054
3:     816.46720: -0.521138  2.48189
```

- `mgc` stands for maximum gradient component
- Inner optimization:
  - Find maximum a posteriori estimates of latent effects
  - Marginalize over the latent effects using the Laplace approximation

Markov property

- We will use Gaussian Markov random fields
- Remember this important result:

  Let **Q** be the precision matrix of a Gaussian random field γ, then

  $$Q_{i,j} = 0 \Leftrightarrow \gamma_i \text{ and } \gamma_j \text{ are conditionally independent.}$$

- A Gaussian Markov random field has a sparse **Q**.
- Using sparse structures is essential for fast inference
- RTMB automatically detects and use sparse structures
  - This makes RTMB much faster than ADMB

Exercise

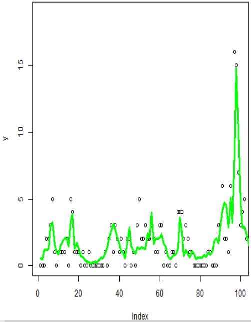
Assume we observe $y_1, ..., y_n$ where $y_i \sim Pois(\mu_i)$ and

$$\log \mu_i = \gamma_i$$
$$\gamma \sim N(0, \Sigma_{AR1}).$$

Here, γ is a stationary mean zero AR(1) process.

- Exercise 3a: Implement and estimate the model with RTMB
- Data and code to get you started is provided in `ar1Latent.R`
- Exercise 3b: Find a 95% C.I. for $\sum_{i=1}^{n} \mu_i$

.



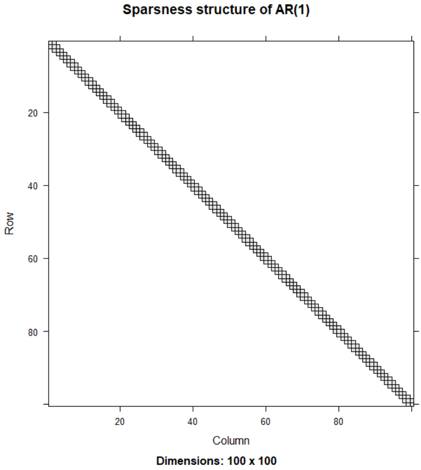
AR1: A stationary mean zero AR1 process has covariance matrix

$$\Sigma_{\text{AR1}} = \frac{\sigma^2}{1-\rho^2} \begin{pmatrix} 1 & \rho & \rho^2 & \cdots & \rho^{n-1} \\ \rho & 1 & \rho & \cdots & \rho^{n-2} \\ \rho^2 & \rho & 1 & \cdots & \rho^{n-2} \\ \vdots & \vdots & \vdots & & \vdots \\ \rho^{n-1} & \rho^{n-2} & \rho^{n-3} & \cdots & 1 \end{pmatrix}$$

- Coefficient of correlation is $cor(\gamma_i, \gamma_j) = \rho^{|j-i|}$
- The precision matrix is:

$$\Sigma_{\text{AR1}}^{-1} = \frac{1}{\sigma^2} \begin{pmatrix} 1 & -\rho & 0 & 0 & \cdots & 0 & 0 & 0 \\ -\rho & 1+\rho^2 & -\rho & 0 & \cdots & 0 & 0 & 0 \\ 0 & -\rho & 1+\rho^2 & -\rho & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & -\rho & 1+\rho^2 & -\rho \\ 0 & 0 & 0 & 0 & \cdots & 0 & -\rho & 1 \end{pmatrix}$$

Obtained Sparsity structure in γ in Exercise 3



- Marginal log-likelihood:

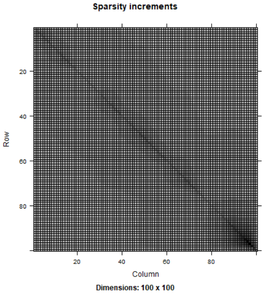
$$\ell_M(\theta, y) \approx \ell(\theta, \hat{u}_\theta, y) - \frac{1}{2}\log(|-\ell''_{uu}|_{u=\hat{u}_\theta}|) + \frac{q}{2}\log(2\pi)$$

Sparsity is important

- Could include the AR1 increments as the random variable
- The following code will give the same results but be slow!

```
par = list(logsd = 0,logitRho = 0,
     epsilon = rep(0,n))

nllBad = function(par){
     getAll(par,data)
     sd = exp(logsd)
     rho = 2/(1 + exp(-logitRho))-1
     gamma = rep(0,length(epsilon))
     nll = dnorm(epsilon[1],0,sqrt(sd*sd/(1-rho^2)),TRUE);
     gamma[1] = epsilon[1]
     for(i in 2:length(gamma)){
          gamma[i] = rho*gamma[i-1] + epsilon[i]
          nll = nll -dnorm(epsilon[i],0,sd,TRUE);
     }
     nll = nll -sum(dpois(y,exp(gamma),TRUE))
     return(nll)
}
objBad = MakeADFun(nllBad,par,random = "epsilon")
```



- The increments do not have the Markov property
- The Laplace approximation is now time-consuming!

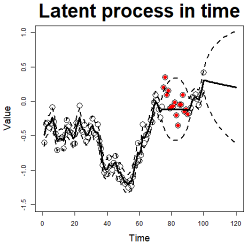
Summary

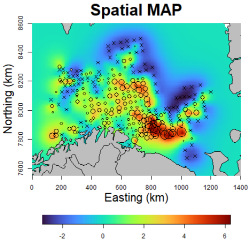
- A random effect is a random variable we do not observe
- We assign statistical structures to the random effects
- Marginalize over the latent variables
    - Laplace approximation

```
obj = MakeADFun(nll, par, random = c("nameOfLatentVariable1", ...))
```

- Optimize the marginal log-likelihood

```
opt = nlminb(obj$par,obj$fn,obj$gr)
```





Syntax for Markov models

- Let the random effect $\gamma \sim N(0, \mathbf{Q}^{-1})$
  - Syntax in RTMB:

```
nll = nll - dgmrf(gamma, 0, Q,TRUE)
```

- Let the random effect $\gamma \sim N(0, \mathbf{Q}_{\text{AR1}}^{-1})$
  - Syntax in RTMB:

```
nll = nll - dautoreg(gamma, 0, phi,log =TRUE, scale = sd, log=TRUE)
```

- Let the random effect $\gamma \sim N(0, \mathbf{Q}^{-1} \otimes \mathbf{Q}_{\text{AR1}}^{-1})$
  - Syntax in RTMB:

```
f1 = function(x) dgmrf(gamma, 0, Q,TRUE)
f2 = function(x) dautoreg(x, phi=phi, log=TRUE)
nll = nll - dseparable(f1, f2)(gamma)
```

  - Much used in spatio-temporal statistics:

$$cov(\gamma(s_1, t_1), \gamma(s_2, t_2)) = C^{(s)}(s_1, s_2) \cdot C^{(t)}(t_1, t_2)$$

Matern covariance structure with smoothness parameter equal 1

- Much used in spatial statistics
- Covariance is given by:

$$\text{Cov}\left(\gamma(s_1), \gamma(s_2)\right) = \sigma^2 \kappa ||s_1 - s_2|| \text{K}(\kappa ||s_1 - s_2||).$$

- σ is the marginal variance
- κ is a spatial scale parameter
  - This variable can be increased on land to include spatial barriers
- Can be represented with a sparse precision matrix!
  - Use functionality from INLA/fmesher to extract Q and call

```
dgmrf(gamma, 0, Q,log = TRUE) #Liklihood contribution of latent effects
```

Spatial example

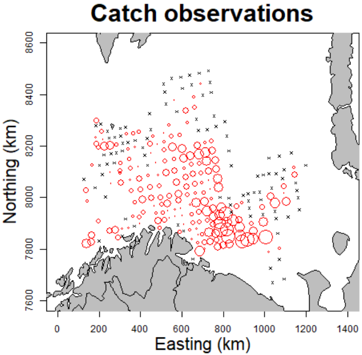
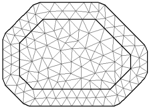


Figure: Area of bubbles is proportional to catch of haddock, crosses are zero-observations.

Spatial modeling with the SPDE-procedure

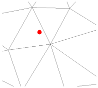
- Set up mesh

```
mesh = fmesher::fm_mesh_2d(locUTM,
          max.edge = c(20, 50)) #inner and outer area
```



- Set up projection matrix

```
A = fmesher::fm_basis(mesh,loc = locUTM)
```



- Set up SPDE-matrices

```
spde = fmesher::fm_fem(mesh)
spdeMatrices = list(c0 = spde$c0, g1 = spde$g1,g2 = spde$g2)
```

- Calculate precision matrix

```
#Precision matrix Q, see Lindgren and Rue 2015 JSS p4
Q <- tau^2*(kappa^4*spdeMatrices$c0 + 2*kappa^2*
      spdeMatrices$g1 + spdeMatrices$g2)
```

$$Q = \frac{1}{\tau^2}\left(\kappa^4 C_0 + 2\kappa^2 G_1 + G2\right)$$

Spatial modeling

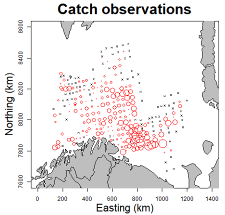
The Institute of Marine Research conducts surveys to calculate fish stock. In this exercise, we use the catch of haddock from the Barents Sea winter survey.

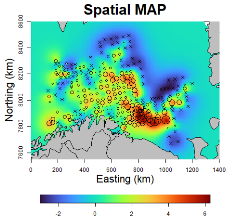
Assume catch $C_i \sim Pois(\mu_i d_i)$, where

$$\log \mu(s_i) = \beta_0 + \boldsymbol{\delta}(s_i).$$

Here, $\boldsymbol{\delta} \sim N(0, \Sigma_{\text{Matern}})$ and d_i is distance trawled

- Exercise 4 a) Estimate the model
    - See `spde.R` to get you started
- Exercise 4 b) Estimate $\sum_{s \in \Omega} \mu(s)$
- Ω is a dense grid covering the survey area given in `predPoints.rds`





Simulate from the model

- Inform what are the observations

```
nll = function(par){
      ...
      y = OBS(y)
      ...
}
```

- Simulate from the model given hyperparameters

```
obj$simulate()
```