Normal distribution and random effects

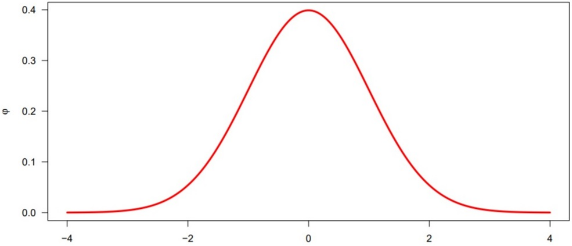
Anders Nielsen and Olav Nikolai Breivik

The Normal Distribution

- A continous probability distribution on $(-\infty, \infty)$
- Probability density function:

$$\phi(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

- Mean is μ and standard deviation is σ
- The interval $(\mu - 2\sigma, \mu + 2\sigma)$ contains 95%



Complete program using normal likelihood

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

data = list()
data$Y = rnorm(1000,2,0.3)

par = list()
par$mu = 0
par$logSigma = 0

obj = MakeADFun(data,par,DLL = "norm")
opt = nlminb(obj$par,obj$fn,obj$gr)
rep = sdreport(obj)
```

```cpp
#include <TMB.hpp>
template<class Type>
Type objective_function<Type>::operator() ()
{
  DATA_VECTOR(Y);
  PARAMETER(mu);
  PARAMETER(logSigma);

  Type sigma = exp(logSigma);
  Type nll = -sum(dnorm(Y,mu,sigma,true));
  return nll;
}
```

- We have now implemented:

$$\text{nll} = -\sum_{i=1}^{1000} \log \phi(y_i|\mu,\sigma),$$

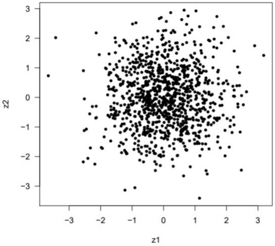
and estimated μ and σ.

Two normal random variables

- Assume we have two independent Gaussian random variables:

$$Z_1 \sim N(0,1) \text{ and } Z_2 \sim N(0,1)$$

- Realisations of (Z_1, Z_2):



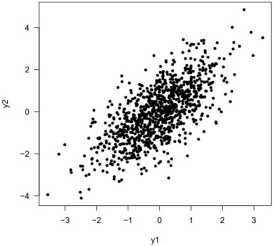
The marginal distribution on each axis is a $N(0,1)$.

Two normal random variables

- Assume now that

$$\mathbf{Y} = \begin{pmatrix} Z_1 \\ Z_1 + Z_2 \end{pmatrix}$$

- Realisations of $\mathbf{Y}$:



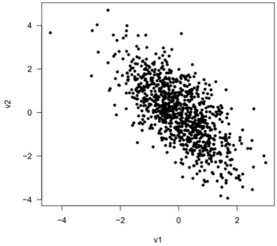
The marginal distributions are $N(0,1)$ and $N(0,2)$.

Two normal random variables

- Assume now that

$$\mathbf{Y} = \begin{pmatrix} Z_1 \\ Z_2 - Z_1 \end{pmatrix}$$

- Realisations of $\mathbf{Y}$:



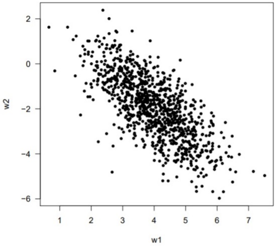
The marginal distributions are $N(0, 1)$ and $N(0, 2)$.

Two normal random variables

- Assume now that

$$\mathbf{Y} = \begin{pmatrix} Z_1 + 4 \\ Z_2 - Z_1 - 2 \end{pmatrix}$$

- Realisations of $\mathbf{Y}$:



The marginal distributions are $N(4, 1)$ and $N(-2, 2)$.

Two normal random variables

- Note that all these cases can be written as:

$$\mathbf{Y} = \mathbf{A}\mathbf{Z} + \mathbf{b},$$

  where $\mathbf{A}$ is a matrix and $\mathbf{Z} = \begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix}$

- E.g. the last example:

$$\mathbf{Y} = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \mathbf{Z} + \begin{pmatrix} 4 \\ -2 \end{pmatrix}.$$

Multivariate normal distribution

- We say that a k-dim random variable X follows a multivariate normal distribution, $X \sim N(\mu, \Sigma)$, if there exists a random l-dim random variable Z such that $X \sim AZ + b$.
- We have that $\Sigma = AA^t$ and $\mu = b$.
- The density of X is:

$$L(x) = \frac{1}{(2\pi)^{k/2}\sqrt{|\Sigma|}} \exp\Big(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\Big)$$

- We write $X \sim N(\mu, \Sigma)$

Covariance and correlation

- The covariance between two random variables is defined as:

$$cov(X, Y) = E[(X - \mu_x)(Y - \mu_y)]$$

- When $X \sim N(\mu, \Sigma)$, then:

$$\Sigma_{ij} = cov(X_i, X_j)$$

- We have that the covariance between a variable ant itself is the variance:

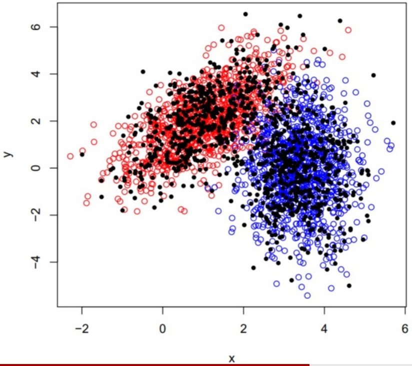
$$\Sigma_{ii} = cov(X_i, X_i) = var(X_i)$$

- The correlation coefficient is deined as:

$$\rho_{ij} = \frac{\Sigma_{ij}}{\sqrt{\Sigma_{ii}\Sigma_{jj}}}$$

Exercise

Assume we have 1000 points from two groups ("red" and "blue"). We are then given 1000 additional points with unknown class. Each of the groups are well described by a 2 dimensional normal distribution. Code to estimate these two normal distributions is provided in `twoNormal.R` and `twoNormal.cpp`. Your task is to assign the most likely class to each of the "black" points.



$$\mathbf{x}_{red} \sim N\left(\begin{pmatrix}\mu_{1,r}\\ \mu_{2,r}\end{pmatrix}, \begin{pmatrix}\sigma_{1,r}^2 & \rho_r\sigma_{1,r}\sigma_{2,r}\\ \rho_r\sigma_{1,r}\sigma_{2,r} & \sigma_{2,r}^2\end{pmatrix}\right)$$

$$\mathbf{x}_{blue} \sim N\left(\begin{pmatrix}\mu_{1,b}\\ \mu_{2,b}\end{pmatrix}, \begin{pmatrix}\sigma_{1,b}^2 & \rho_b\sigma_{1,b}\sigma_{2,b}\\ \rho_b\sigma_{1,b}\sigma_{2,b} & \sigma_{2,b}^2\end{pmatrix}\right)$$

Asymptomatically distribution of MLE

- Asymptotically (as we gather more data) the distribution of our estimator will become

$$\hat{\theta} \sim N(\theta_{\text{true}}, H(\theta_{\text{true}})^{-1})$$

where H is the hessian matrix:

$$H(\theta) = \begin{pmatrix} \frac{\partial^2 f(\theta)}{\partial^2 \theta_1} & \cdots & \frac{\partial^2 f(\theta)}{\partial \theta_1 \partial \theta_k} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f(\theta)}{\partial \theta_k \partial \theta_1} & \cdots & \frac{\partial^2 f(\theta)}{\partial^2 \theta_k} \end{pmatrix}$$

- We can use this result to construct parameter confidence intervals

Often interested in a function of the parameters

- Say we are interested in an estimate of a differentiable function $g(\theta)$
- MLE is $g(\hat{\theta})$
- Uncertainty can be obtained with the delta-method:

$$g(\theta) \overset{\text{approx}}{\sim} N(g(\hat{\theta}), \nabla g(\hat{\theta})^T H(\hat{\theta})^{-1} \nabla g(\hat{\theta}))$$

- Mini Exercise: Assume we have estimated $\theta = (\log F_2, \log F_3, \log F_4)$ to (-1.13, -0.75, -0.94) with an estimated covariance matrix:

$$H(\hat{\theta})^{-1} = \begin{pmatrix} 0.0222 & 0.0135 & 0.0114 \\ 0.0135 & 0.0169 & 0.0137 \\ 0.0114 & 0.0137 & 0.0191 \end{pmatrix}$$

Set up a 95% confidence interval for $\log \bar{F}_{2-4}$

Often interested in a function of the parameters

- Say we are interested in an estimate of a differentiable function $g(\theta)$
- MLE is $g(\hat{\theta})$
- Uncertainty can be obtained with the delta-method:

$$g(\theta) \overset{\text{approx}}{\sim} N(g(\hat{\theta}), \nabla g(\hat{\theta})^T H(\hat{\theta})^{-1} \nabla g(\hat{\theta}))$$

- Mini Exercise: Assume we have estimated $\theta = (\log F_2, \log F_3, \log F_4)$ to (-1.13, -0.75, -0.94) with an estimated covariance matrix:

$$H(\hat{\theta})^{-1} = \begin{pmatrix} 0.0222 & 0.0135 & 0.0114 \\ 0.0135 & 0.0169 & 0.0137 \\ 0.0114 & 0.0137 & 0.0191 \end{pmatrix}$$

  Set up a 95% confidence interval for $\log \bar{F}_{2-4}$

- Solution: $g(\hat{\theta}) = \log(\frac{1}{3}(e^{\widehat{\log F_2}} + e^{\widehat{\log F_3}} + e^{\widehat{\log F_4}})) \approx -0.928$
- $\nabla g(\hat{\theta}) = \frac{1}{e^{\widehat{\log F_2}} + e^{\widehat{\log F_3}} + e^{\widehat{\log F_4}}}(e^{\widehat{\log F_2}}, e^{\widehat{\log F_3}}, e^{\widehat{\log F_4}})$
- $g(\hat{\theta}) \pm 2 * \sqrt{\nabla g(\hat{\theta})^T H(\hat{\theta})^{-1} \nabla g(\hat{\theta})} \approx -0.928 \pm 2 * 0.122$

Assume $\mathbf{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix}, \boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix}, \boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{pmatrix}$$

Then $\mathbf{x}_1 | \mathbf{x}_2 \sim N(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}})$ where

$$\begin{aligned} \tilde{\boldsymbol{\mu}} &= \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12} \boldsymbol{\Sigma}_{22}^{-1} (\mathbf{x}_2 - \boldsymbol{\mu}_2) \\ \tilde{\boldsymbol{\Sigma}} &= \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12} \boldsymbol{\Sigma}_{22}^{-1} \boldsymbol{\Sigma}_{11}. \end{aligned}$$

Let $\boldsymbol{Q} = \boldsymbol{\Sigma}^{-1}$. We have that

$$\begin{aligned} \mathrm{Var}[x_i | \mathbf{x}_{-i}] &= Q_{ii}^{-1} \\ \mathrm{E}[x_i | \mathbf{x}_{-i}] &= \mu_i - Q_{ii}^{-1} \sum_{i \neq j} Q_{ij} (x_j - \mu_j). \end{aligned}$$

Note that $Q_{ij} = 0$ is equivalent to that x_i and x_j are conditional independent.

Implement an AR(1) process

- Assume we have $n = 100$ observations $x_1, ..., x_n$ from a mean zero AR(1) process:

$$x_{i+1} = \phi x_i + \epsilon_i, \text{ where } \epsilon_i \sim N(0, \sigma^2)$$

- If the process is in equilibrium when we start observing it, then $x_1 \sim N(0, \sigma^2/(1-\phi^2))$
- Exercise a) Implement the model with use of the univariate normal distribution (`dnorm`)
- Exercise b) Implement the model with use of multivariate normal function (`AR1_t`)
  - How sparse is the precision matrix to an AR(1) process?
- Verify that you get similar results.

Latent random effects

- Do not observe directly
- Observe indirectly trough the model
    - E.g. fishing mortality in SAM

Simple example: Assume we observe $Y_1, ..., Y_n$ where $Y_i \sim Pois(\mu_i)$ and

$$\mu_i = \gamma_i$$
$$\boldsymbol{\gamma} \sim N(\mathbf{0}, \boldsymbol{\Sigma}).$$

Here γ is a latent random effect.

Latent random effects

- Do not observe directly
- Observe indirectly trough the model
  - E.g. fishing mortality in SAM

Simple example: Assume we observe $Y_1, ..., Y_n$ where $Y_i \sim Pois(\mu_i)$ and

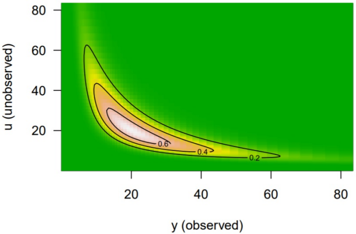
$$\mu_i = \gamma_i$$
$$\boldsymbol{\gamma} \sim N(\mathbf{0}, \boldsymbol{\Sigma}).$$

Here γ is a latent random effect.

- If γ_i is Gamma distributed, Y_i is negative binomial distributed.

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

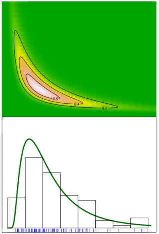
We are typically not interested in the likelihood for a specific u (it is random)

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

The marginal distribution is:

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

Observations provide information trough the marginal distribution



Latent random effects

- The marginal likelihood is:

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

  - θ is model parameters
  - y is the observed random values (the observations)
  - u is the NOT observed random values
- How to calculate the integral?
  - Numerical integration not practical (need a loot of integration points)
  - Seldom an analytical solution, e.g. negative binomial likelihood if...
  - Solution: Approximate with a 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:

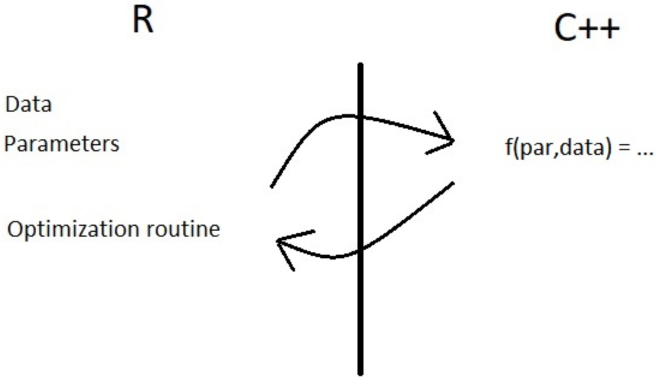
$$\begin{aligned}
L_M(\theta, y) &= \int_{\mathbb{R}^q} L(\theta, u, y) du \\
&= \int_{\mathbb{R}^q} \exp\Big(\ell(\theta, u, y)\Big) du \\
&\approx \int_{\mathbb{R}^q} \exp\Big(\ell(\theta, \hat{u}_\theta, y) - \frac{1}{2}(u - \hat{u}_\theta)^t (\ell''_{uu}|_{u=\hat{u}_\theta})(u - \hat{u}_\theta)\Big)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

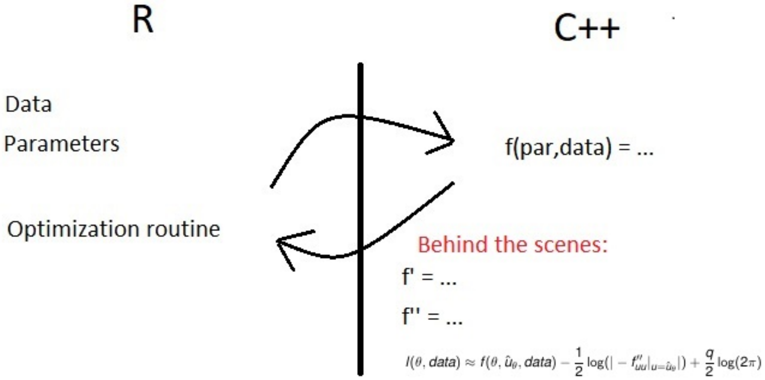
The last step is obtained by observing that the integrand has a Gaussian shape.

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
- Why is automatic differentiation and latent conditional independence structure important to utilize?


R
C++
Data
Parameters
f(par,data) = ...
Optimization routine


R
C++
Data
Parameters
f(par,data) = ...
Optimization routine
Behind the scenes:
f' = ...
f'' = ...
$l(\theta, data) \approx f(\theta, \hat{u}_\theta, data) - \frac{1}{2}\log(|-f''_{uu}|_{u=\hat{u}_\theta}|) + \frac{q}{2}\log(2\pi)$

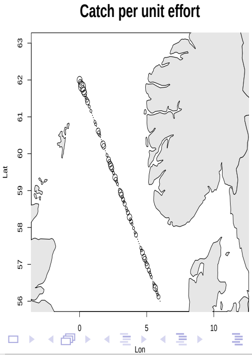
Exercise

Assume we observe $Y_1, ..., Y_n$ where $Y_i \sim Pois(\mu_i)$ and

$$\log \mu_i = \gamma_i$$
$$\boldsymbol{\gamma} \sim N(\mathbf{0}, \boldsymbol{\Sigma}),$$

and γ is an AR(1)-process.

- Code is provided in `ar1Latent.R` and `ar1Latent.cpp`
- Data provided in `ar1Latent.RData`
- Exercise a) Implement the mode
- Exercise b) Inspect the sparseness structure of γ
  - Why is the sparseness important?



Solution, R- and C-side

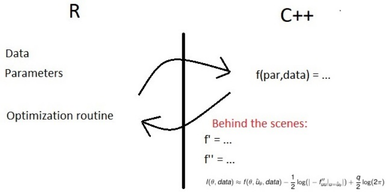
```r
library(TMB)
compile("ar1Latent.cpp")
dyn.load("ar1Latent")

load("cpue.RData")
data = list(y = y)
par = list(logSigma = -2,
           phiTrans = 1,
           gamma = rep(0,length(y)))

obj = MakeADFun(data,par,random = "gamma", DLL = "ar1Latent")
opt = nlminb(obj$par,obj$fn,obj$gr,control = list(trace = 1))
rep = sdreport(obj,getJointPrecision = TRUE)
```

```cpp
#include <TMB.hpp>
template<class Type>
Type objective_function<Type>::operator() ()
{
  using namespace density;
  DATA_VECTOR(y);
  PARAMETER(logSigma);
  PARAMETER(phiTrans);
  PARAMETER_VECTOR(gamma);

  Type phi =  Type(2)/(1 + exp(-2*phiTrans)) -Type(1);
  Type sd = exp(logSigma);

  Type nll=0;
  nll+=SCALE(AR1(phi),sqrt(sd*sd/(1-phi*phi)))(gamma);

  for(int i=1;i<y.size();i++){
    nll += -dpois(y(i),exp(gamma(i)),true);
  }
  return nll;
}
```

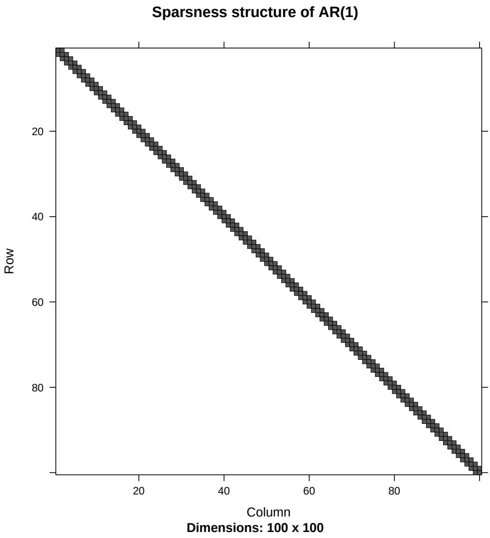
- Note that γ is now included as `random`

Exercise solution

```
outer mgc:  2.125321
  4:     158.72116: -0.405600  1.00234
iter: 1  value: 167.6788 mgc: 0.9840473 ustep: 1
iter: 2  value: 167.6786 mgc: 0.01624102 ustep: 1
iter: 3  value: 167.6786 mgc: 1.878016e-05 ustep: 1
iter: 4  mgc: 3.701173e-11
iter: 1  value: 153.1634 mgc: 0.05411768 ustep: 1
iter: 2  value: 153.1634 mgc: 0.0001761833 ustep: 1
iter: 3  mgc: 1.925322e-09
iter: 1  value: 153.1634 mgc: 0.05411768 ustep: 1
iter: 2  value: 153.1634 mgc: 0.0001761833 ustep: 1
iter: 3  mgc: 1.925322e-09
outer mgc:  0.4393079
  5:     158.64804: -0.348985  1.00601
iter: 1  value: 150.4947 mgc: 0.2225958 ustep: 1
iter: 2  value: 150.4947 mgc: 0.0004934857 ustep: 1
iter: 3  mgc: 9.436551e-09
iter: 1  value: 152.9936 mgc: 0.006990456 ustep: 1
iter: 2  value: 152.9936 mgc: 6.445189e-07 ustep: 1
iter: 3  mgc: 1.110223e-14
iter: 1  value: 152.9936 mgc: 0.006990456 ustep: 1
iter: 2  value: 152.9936 mgc: 6.445189e-07 ustep: 1
iter: 3  mgc: 1.110223e-14
outer mgc:  0.01172901
  6:     158.64298: -0.332983  1.02619
iter: 1  value: 155.2799 mgc: 0.1856167 ustep: 1
iter: 2  value: 155.2799 mgc: 0.0004893153 ustep: 1
iter: 3  mgc: 6.493412e-09
iter: 1  value: 153.249 mgc: 0.01630964 ustep: 1
iter: 2  value: 153.249 mgc: 4.901816e-06 ustep: 1
iter: 3  mgc: 5.8753e-13
iter: 1  value: 153.0547 mgc: 0.001403454 ustep: 1
iter: 2  value: 153.0547 mgc: 5.976854e-08 ustep: 1
iter: 3  mgc: 1.998401e-15
iter: 1  value: 153.0547 mgc: 0.001403454 ustep: 1
iter: 2  value: 153.0547 mgc: 5.976854e-08 ustep: 1
iter: 3  mgc: 1.998401e-15
outer mgc:  0.004273235
  7:     158.64297: -0.332503  1.02580
```



Exercise solution



Motivation

Structures in many dimensions can be included in $\boldsymbol{\Sigma}$

- It is then essential to represent $\boldsymbol{\Sigma}$ with a sparse precision matrix.