Project2

Gaussian random field with application of INLA

Yaolin Ge, Florian Beiser

Contents

Part I Multivariate normal distribution

Let $\boldsymbol{x} = (x_1, \ldots, x_n), n = 100$ be multivariate normal distributed with $E(x_i) = 0, Var(x_i) = 1$, and $Corr(x_i, x_j) = e^{-0.1|i-j|}$

a) Compute and image the covariance matrix $\boldsymbol{\Sigma}$ of $\boldsymbol{x}$

b) Find the lower Cholesky factor $\boldsymbol{L}$, such that $\boldsymbol{L}\boldsymbol{L}^T = \boldsymbol{\Sigma}$, of this covariance matrix, and image.

c) Sample $\boldsymbol{x} = \boldsymbol{L}\boldsymbol{z}$, where $\boldsymbol{z}$ is a length n random vector of independent standard normal variables. Plot the sample.

d) Find the precision matrix $\boldsymbol{Q}$ of the covariance matrix, and compute the lower Cholesky factor $\boldsymbol{L}_Q$, such that $\boldsymbol{L}_Q\boldsymbol{L}_Q^T = \boldsymbol{Q}$, of this matrix. Image these matrices and compare them to the images obtained in a) and b)

e) Sample $\boldsymbol{x}$ by solving $\boldsymbol{L}_Q^T\boldsymbol{x} = \boldsymbol{z}$, where $\boldsymbol{z}$ is a length n random vector of independen standard normal variables. Plot the sample.

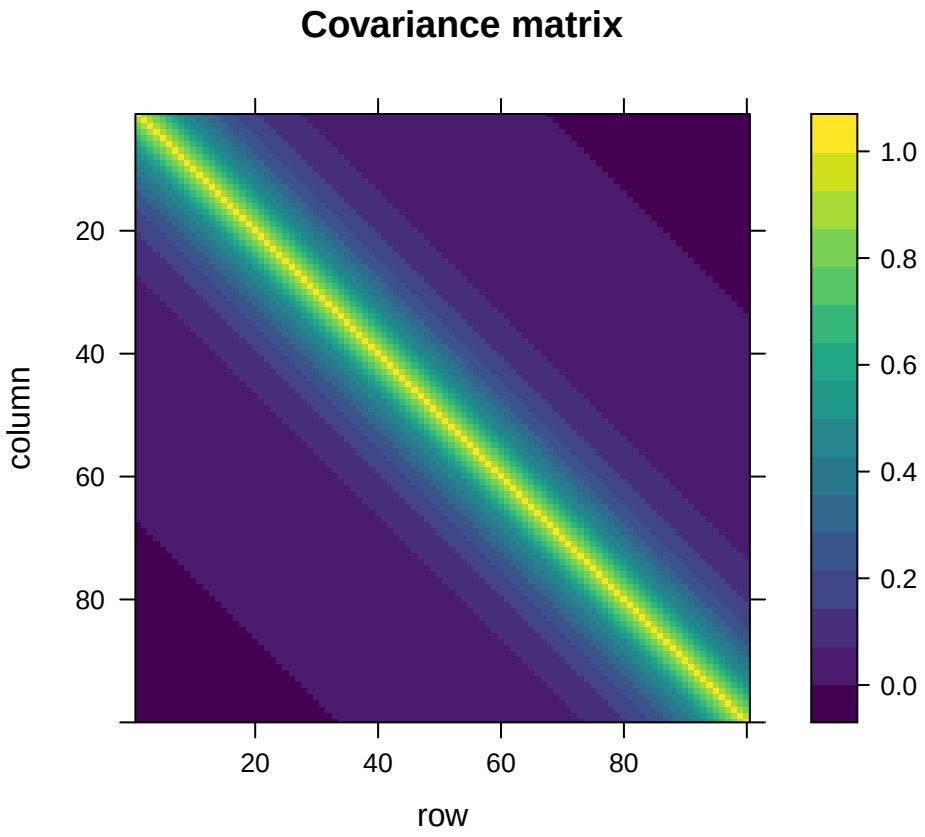
f) Permute the ordering of variables in $\boldsymbol{x}$, and redo the exercises.

Solution to Part I

a)

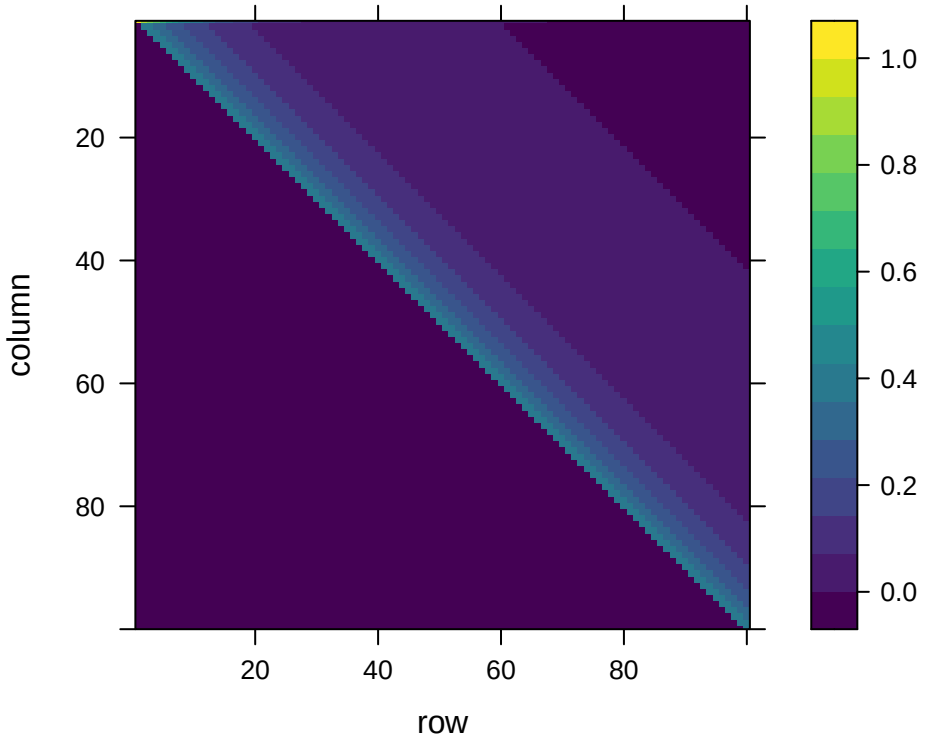
Given that $\mathbf{\Sigma} = e^{-0.1|i-j|}$. The covariance matrix can be expressed as follows:

$$\Sigma = \begin{pmatrix} 1 & e^{-0.1h_{12}} & \dots & e^{-0.1h_{1n}} \\ e^{-0.1h_{21}} & 1 & \dots & e^{-0.1h_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ e^{-0.1h_{n1}} & e^{-0.1h_{n2}} & \dots & 1 \end{pmatrix}$$



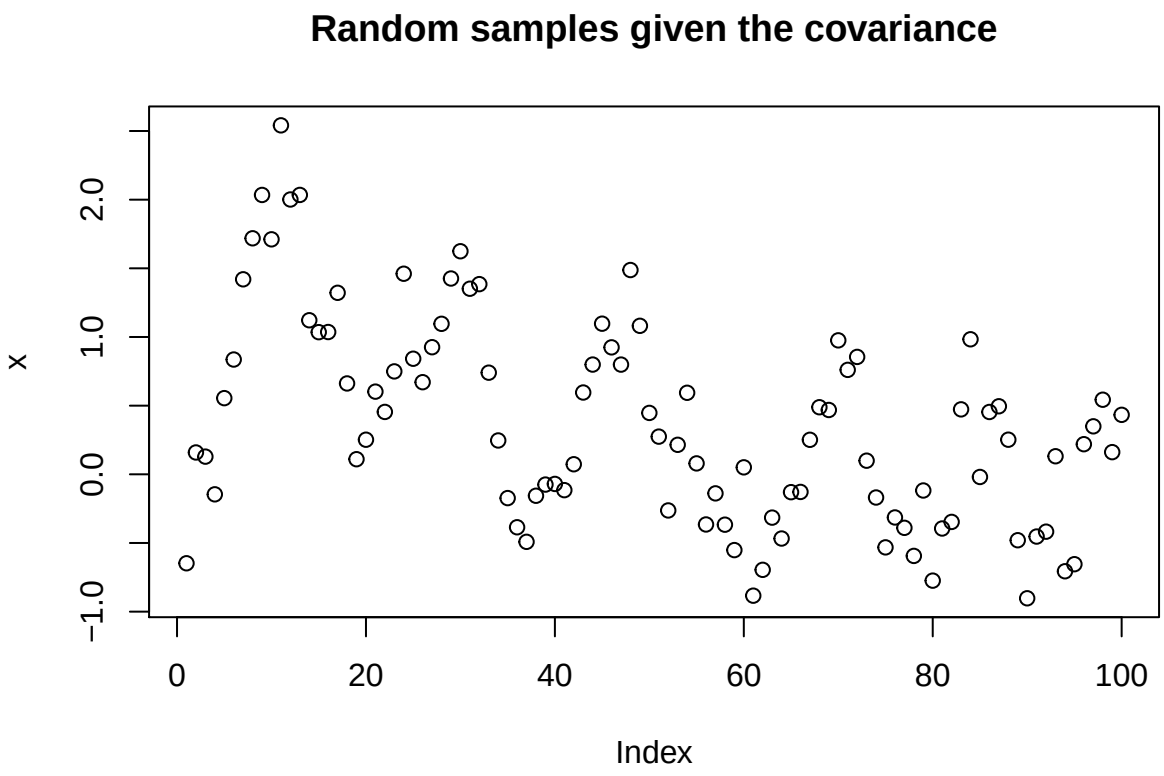
b)

According to the cholesky decomposition rule, $\boldsymbol{L}$ is the lower triangular matrix for $\mathbf{\Sigma}$, it can be easily computed from R using `L = chol(Sigma)`. It is then plotted as below.



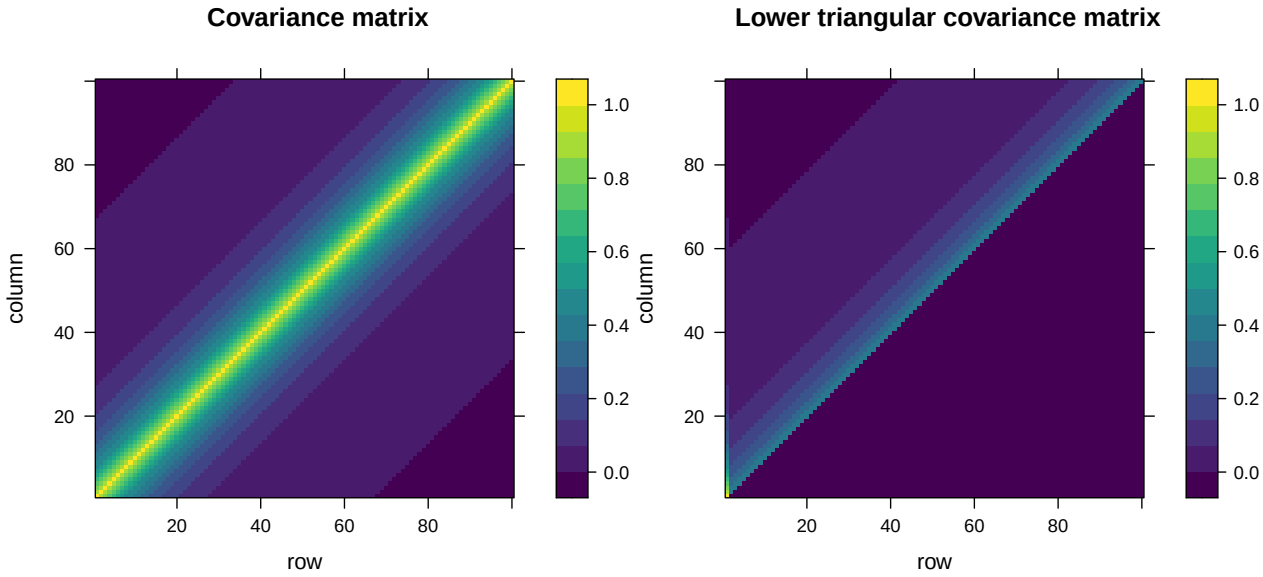
c)

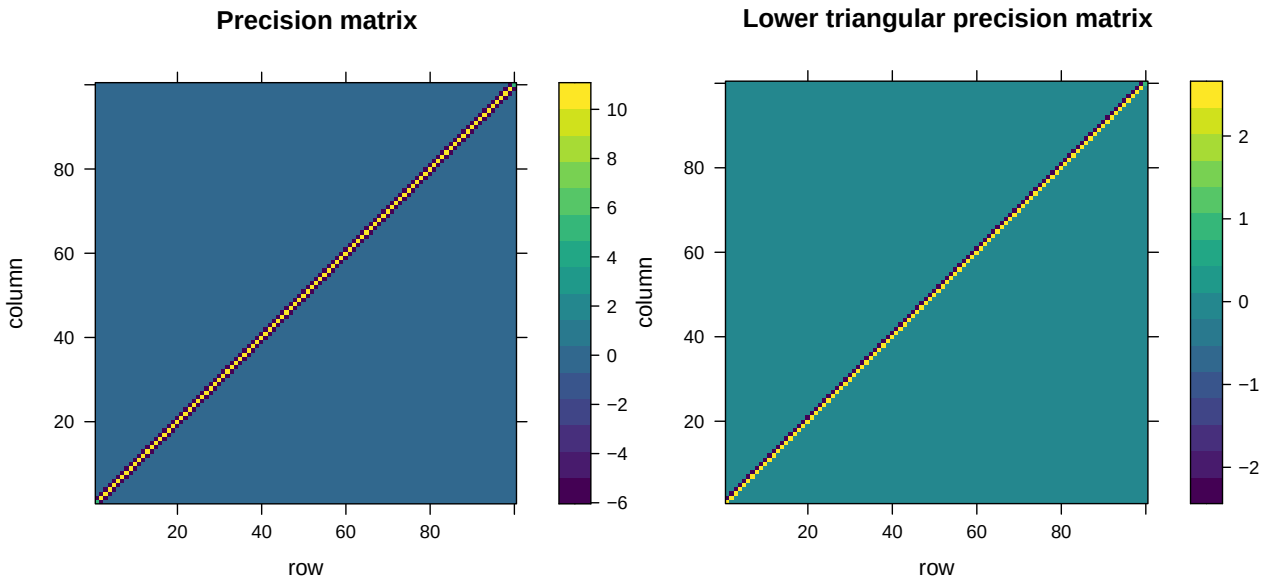
Sample using $\boldsymbol{x} = \boldsymbol{L}\boldsymbol{z}$ transforms the zero-mean, standard normal random variales to the random variables with the desired covariance matrix.



d)

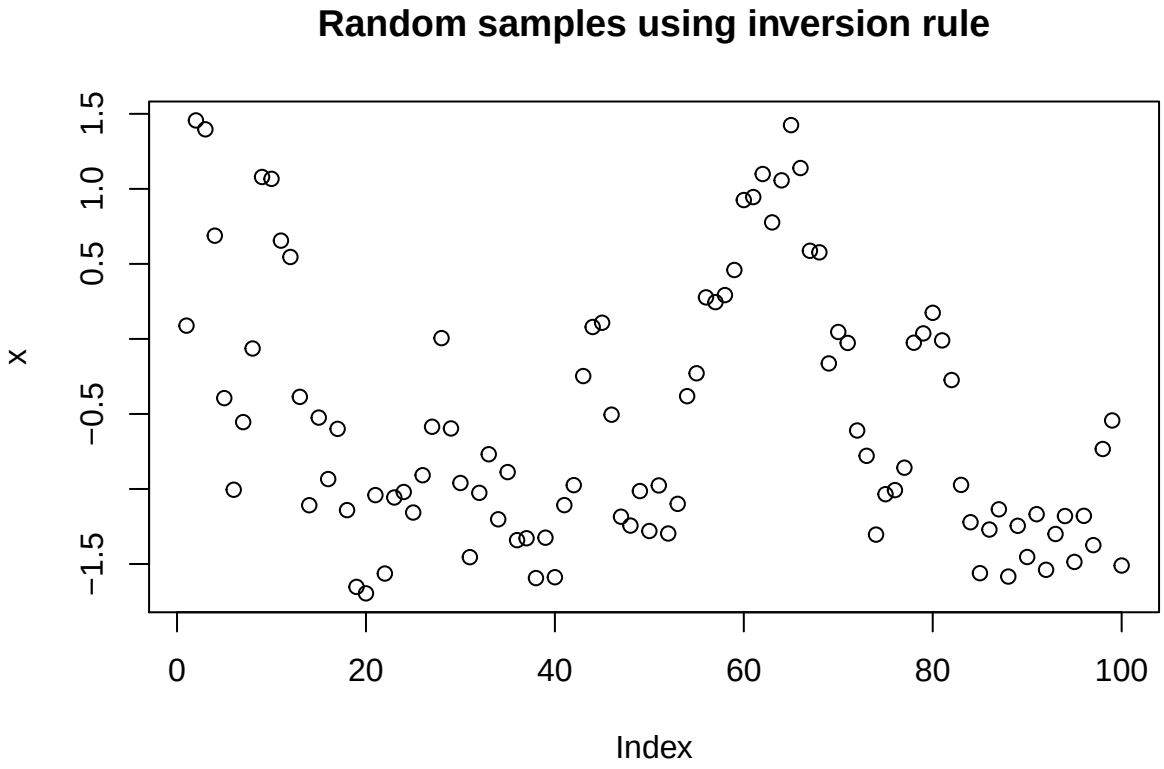
The precision matrix $\boldsymbol{Q}$ is the inverse of the covariance matrix $\boldsymbol{\Sigma}$, it is computed using `Q = solve(Sigma)` in R. The three matrices are thereby depicted as follows. Since the covariance matrix is not singular, given that the it belongs to the Matern family, thus it is analytically guaranteed to have positive definite property. Therefore, both precision matrix and the lower triangular precision matrix exist.





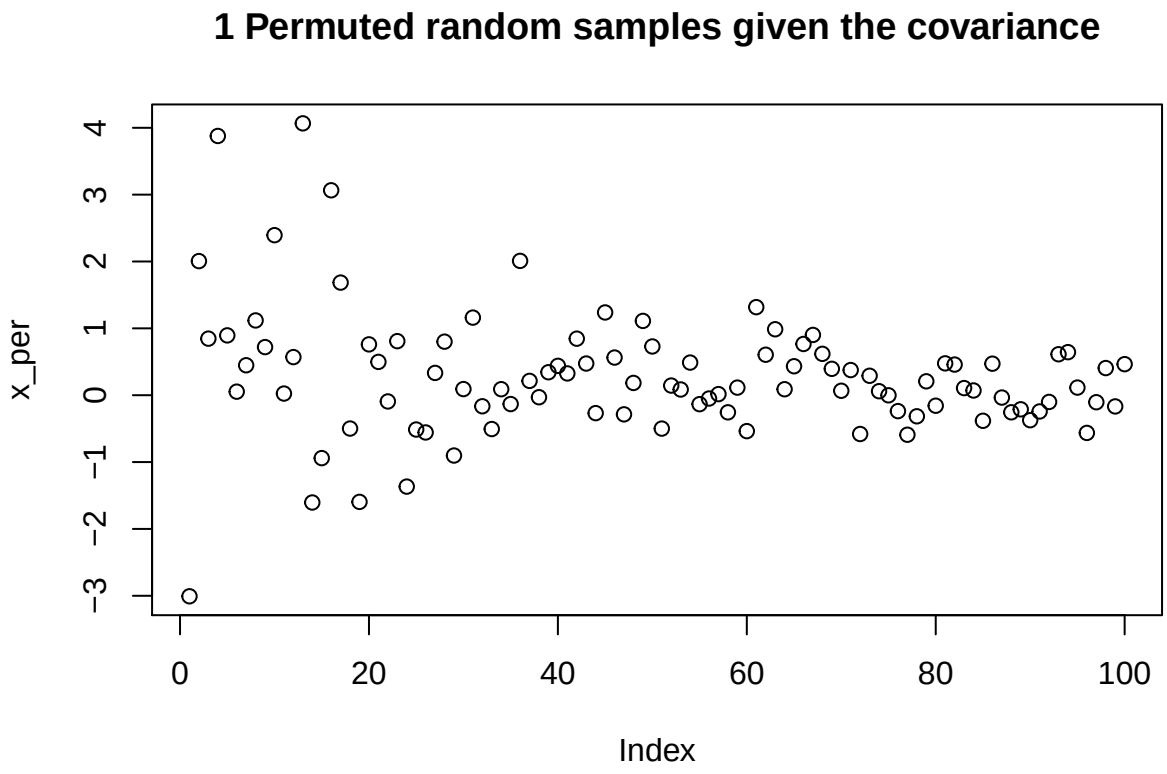
e)

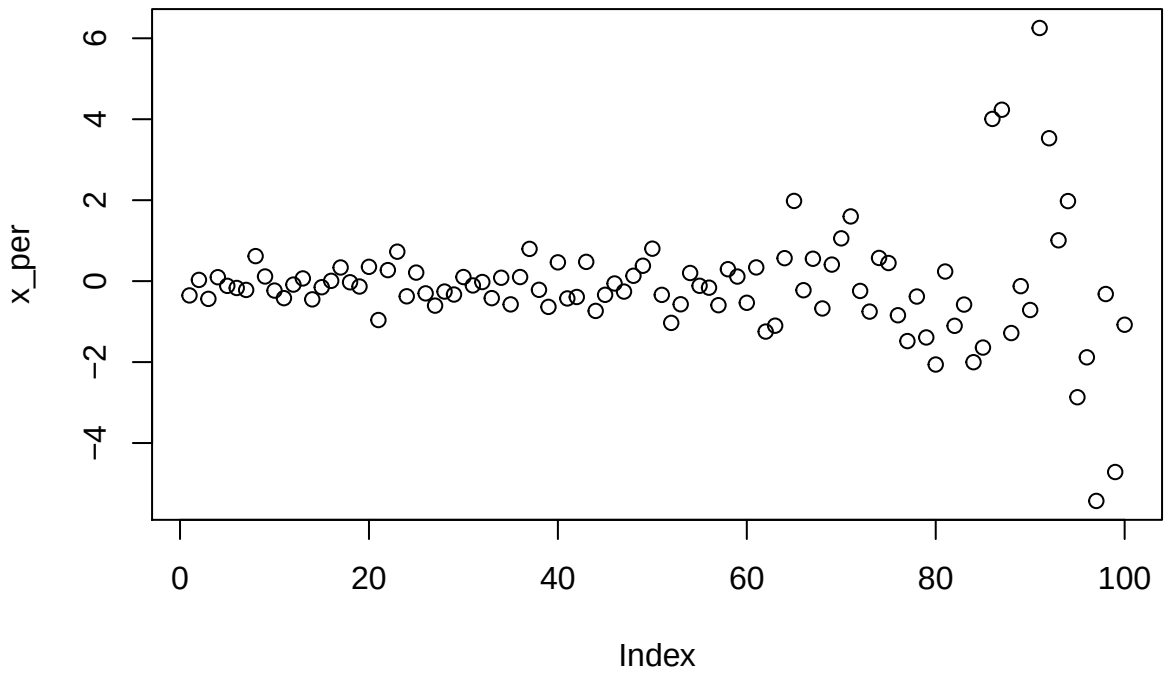
Similarly, the expected random samples can be generated using the inversion of the above formula, thus $\boldsymbol{L}_Q^T \boldsymbol{x} = \boldsymbol{z}$

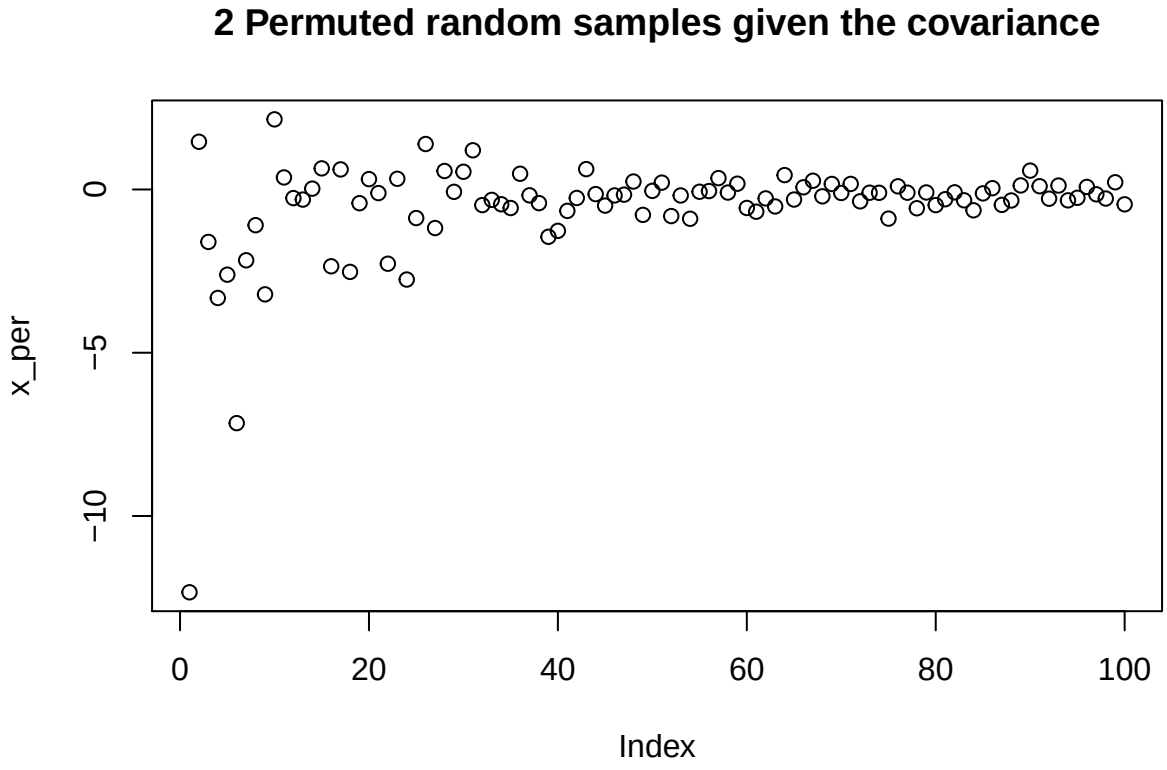


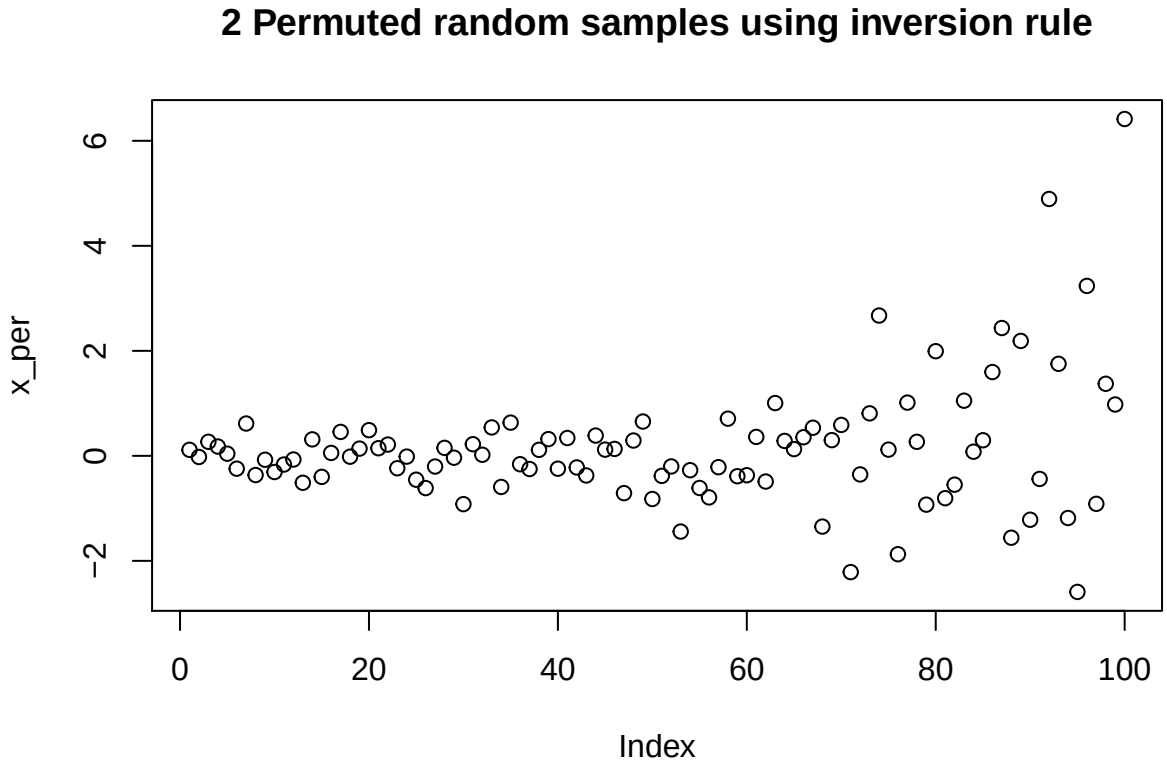
f)

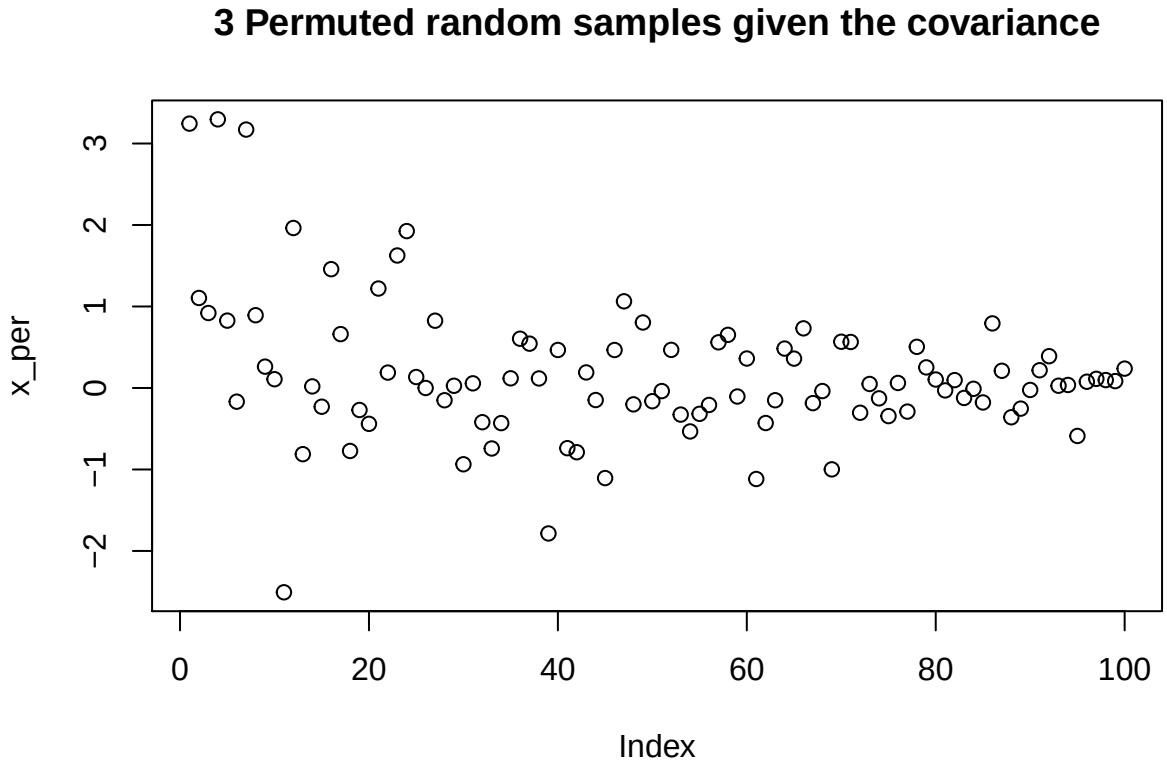
Permute $\boldsymbol{x}$ to make randomise the ordering of the grid, the asscociated covariance matrix can be thereby modified in a sparse way.

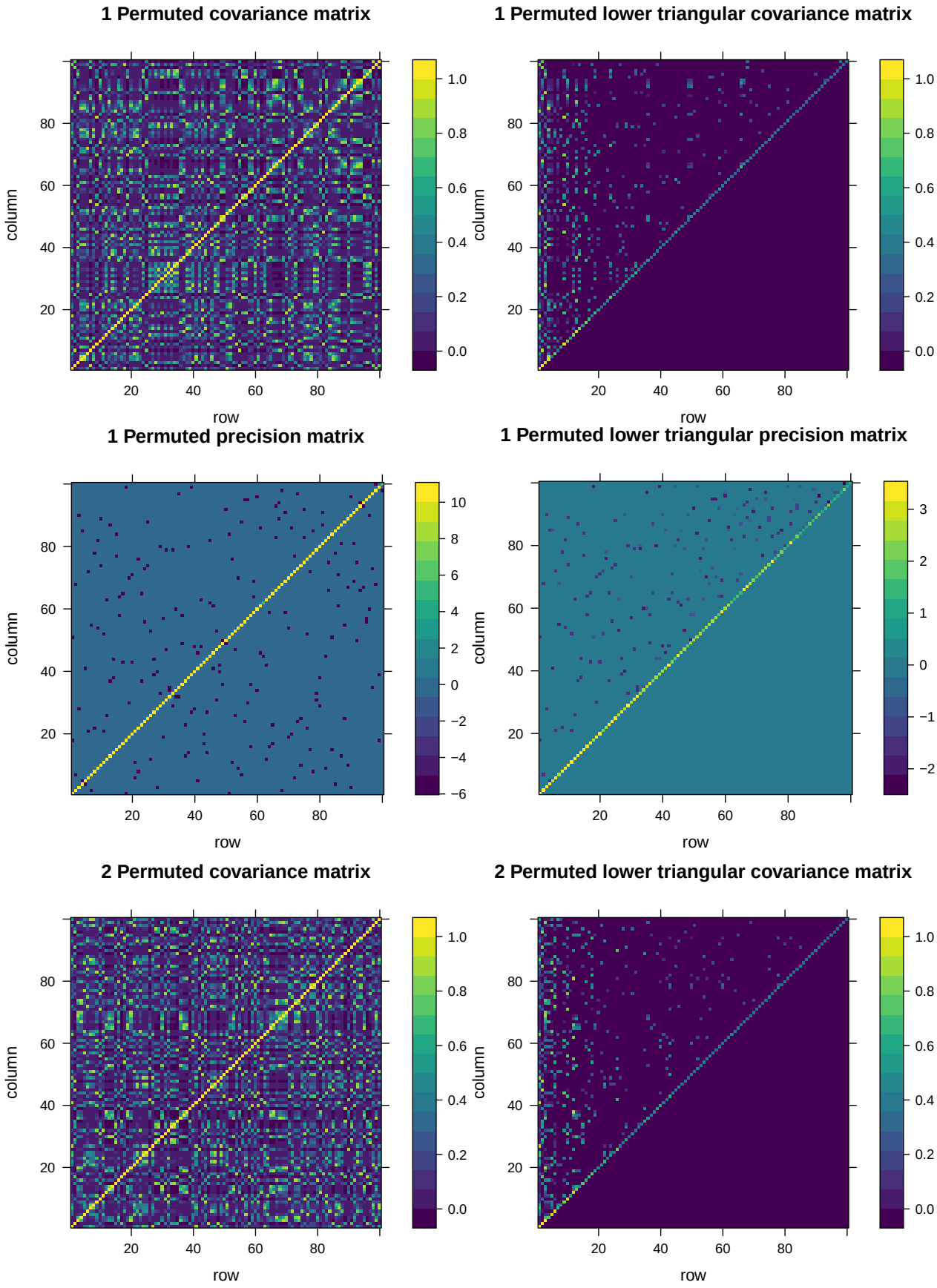

1 Permuted random samples given the covariance
x_per
−3
−2
−1
0
1
2
3
4
0
20
40
60
80
100
Index

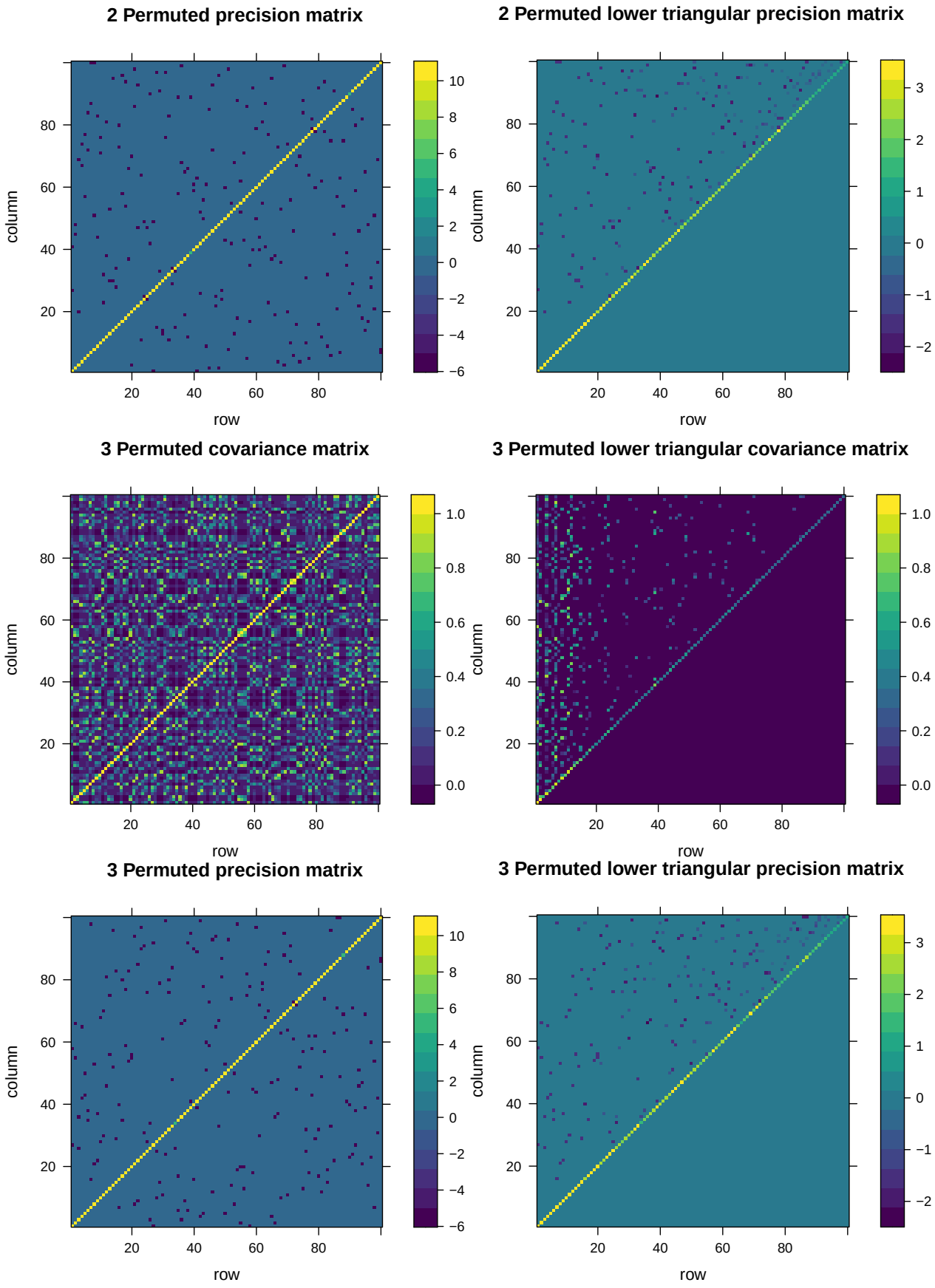

1 Permuted random samples using inversion rule
x_per
−4
−2
0
2
4
6
0
20
40
60
80
100
Index


2 Permuted random samples given the covariance
x_per
0
−5
−10
0
20
40
60
80
100
Index


2 Permuted random samples using inversion rule
x_per
6
4
2
0
−2
0
20
40
60
80
100
Index


3 Permuted random samples given the covariance
x_per
Index
3 Permuted random samples using inversion rule
x_per
Index


1 Permuted covariance matrix
1 Permuted lower triangular covariance matrix
1 Permuted precision matrix
1 Permuted lower triangular precision matrix
2 Permuted covariance matrix
2 Permuted lower triangular covariance matrix
column
row


2 Permuted precision matrix
2 Permuted lower triangular precision matrix
3 Permuted covariance matrix
3 Permuted lower triangular covariance matrix
3 Permuted precision matrix
3 Permuted lower triangular precision matrix
column
row
20
40
60
80
10
8
6
4
2
0
−2
−4
−6
3
1
−1
1.0
0.8
0.6
0.4
0.2
0.0

Part II Gaussian random fields and Kriging

The purpose of this computer exercise is to give an introduction to parameter estima- tion and kriging for Gaussian random field models for spatial data.

We assume the following observation model on the unit square:

$$y(\boldsymbol{s}_j) = x(\boldsymbol{s}_j) + \epsilon_j, \quad j = 1, \ldots, N,$$

where $\epsilon_j \;\; N(0, \tau^2)$ are independent measurement noise terms. Further, consider a Matérn covariance function for the Gaussian random field $\boldsymbol{x(s)}$:

$$Cov(x(\boldsymbol{s}_i), x(\boldsymbol{s}_j)) = \Sigma_{i,j} = \sigma^2(1 + \phi h)\exp(-\phi h),$$

where h denotes the Euclidean distance between the two sites $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$.

We assume the mean increases with east and north coordinates as follows: $\mu_j = \alpha((s_{j1} - 0.5) + (s_{j2} - 0.5))$, for site $\boldsymbol{s}_j = (s_{j1}, s_{j2})$ on the unit square.

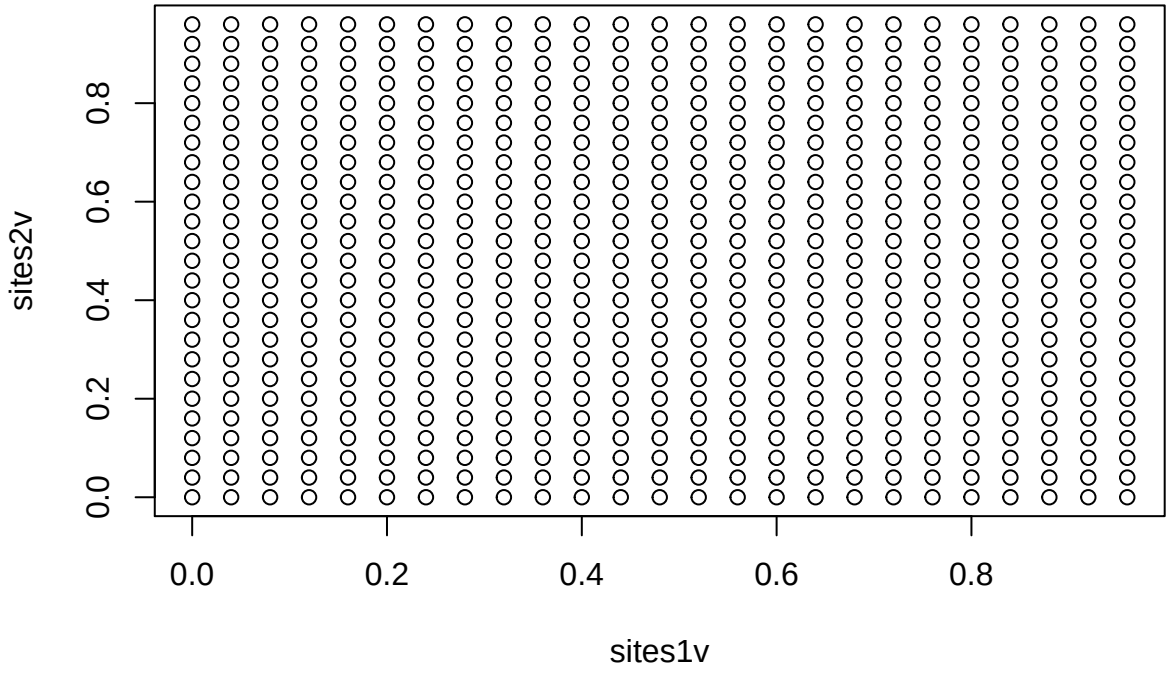
2.1 Simulation

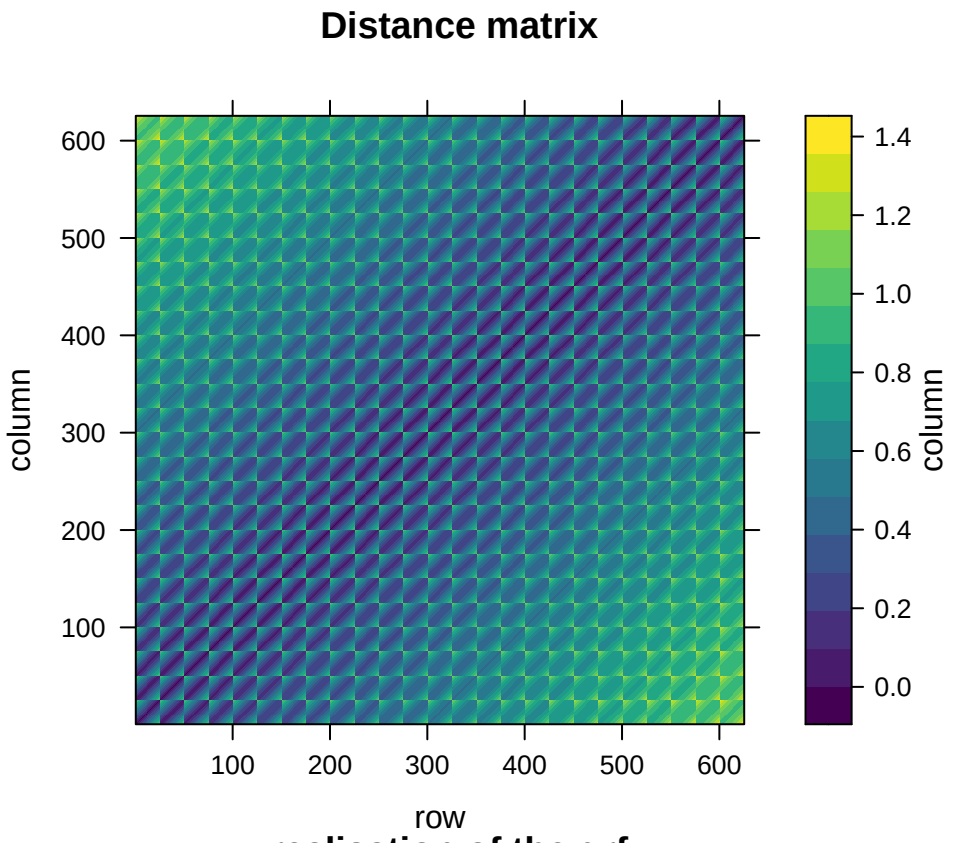
Simulate $N = 200$ random sites in the unit square and plot them. Form the covariance matrix using $\sigma = 1, \phi = 10, \tau = 0.05$. Take its Cholesky decomposition and simulate dependent zero-mean Gaussian data variables, then add the mean using $\alpha = 1$. Plot your observations.

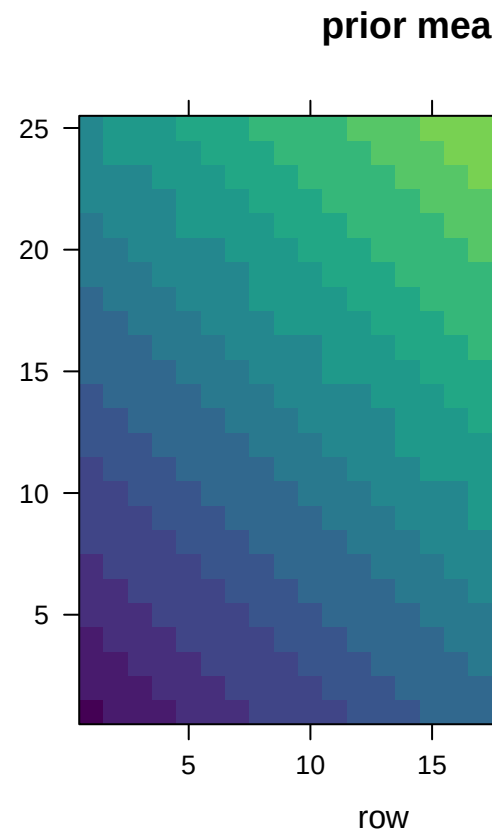
The true mean of the field is expressed as

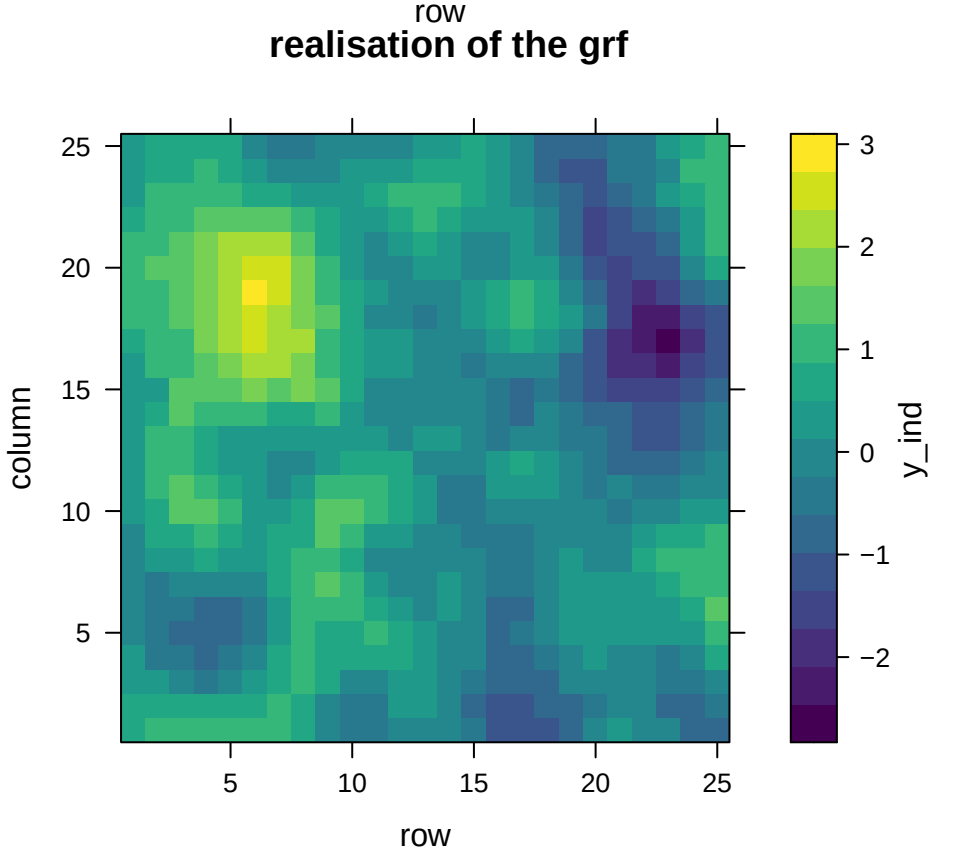
$$\mu_i = \alpha((s_{i1} - 0.5) + (s_{i2} - 0.5)) = -\alpha + \alpha s_{i1} + \alpha s_{i2} = \underbrace{\begin{bmatrix} 1 & s_{i1} & s_{i2} \end{bmatrix}}_{\boldsymbol{h}^T(\boldsymbol{s}_i)} \underbrace{\begin{bmatrix} -\alpha \\ \alpha \\ \alpha \end{bmatrix}}_{\boldsymbol{\beta}}$$

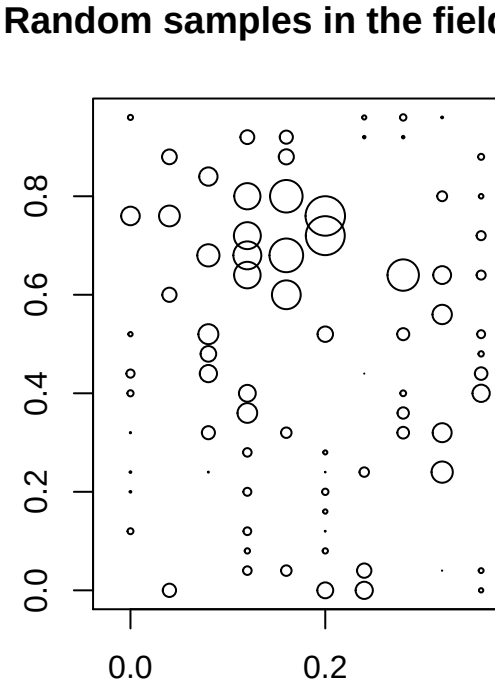
where s_{i1}, s_{i2} are the location from east and north direction in the grid.




Distance matrix
column
row
600
500
400
300
200
100
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0


prior mea
column
row
25
20
15
10
5


realisation of the grf
column
row
y_ind
25
20
15
10
5
3
2
1
0
−1
−2


Random samples in the field
0.0
0.2
0.4
0.6
0.8

2.2 Paramter estimation

We will now use the simulated data to estimate the model parameters $\alpha, \sigma^2, \tau^2, \phi$ using maximum likelihood estimation. Iterate between the update for the mean parameter, and updating the covariance parameters. Monitor the likelihood function at each step of the algorithm to check convergence.

The mean of the field is modelled by $p(\boldsymbol{x})$ and the imperfect information $\boldsymbol{y} = (y_1, \ldots, y_m)$ conditional on $\boldsymbol{x}$ can be modelled by $p(\boldsymbol{y}|\boldsymbol{x})$, which can be expressed as follows:

$$p(\boldsymbol{x}) = N(\boldsymbol{H}\beta, \boldsymbol{\Sigma}), \quad p(\boldsymbol{y}|\boldsymbol{x}) = N(\boldsymbol{F}\boldsymbol{x}, \boldsymbol{T})$$

Therefore, the marginal likelihood of the data is

$$p(\boldsymbol{y}) = N(\boldsymbol{G}\boldsymbol{\beta}, \boldsymbol{C}), \quad \boldsymbol{G} = \boldsymbol{F}\boldsymbol{H}, \quad \boldsymbol{C} = \boldsymbol{F}\boldsymbol{\Sigma}\boldsymbol{F}^{\boldsymbol{T}} + \boldsymbol{T}$$

The log-likelihood as a function of β and unknown fixed nuisance parameters $\boldsymbol{\theta}$ in the prior covariance matrix $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}(\boldsymbol{\theta})$, and/or the likelihood noise matrix $\boldsymbol{T} = \boldsymbol{T}(\boldsymbol{\theta})$ becomes

$$l(\boldsymbol{\theta}, \boldsymbol{\beta}) = -\frac{m}{2}\log(2\pi) - \frac{1}{2}\log|\boldsymbol{C}| - \frac{1}{2}(\boldsymbol{y} - \boldsymbol{G}\boldsymbol{\beta})^T\boldsymbol{C}^{-1}(\boldsymbol{y} - \boldsymbol{G}\boldsymbol{\beta})$$

The MLEs of $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ are obtained by

$$(\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\theta}}) = \arg\max_{\boldsymbol{\beta}, \boldsymbol{\theta}}\{l(\boldsymbol{\beta}, \boldsymbol{\theta})\}$$

For fixed $\boldsymbol{\theta}$, the MLEs of $\boldsymbol{\beta}$ can be determined analytically.

$$\frac{dl}{d\boldsymbol{\beta}} = \boldsymbol{G}^T\boldsymbol{C}^{-1}\boldsymbol{y} - \boldsymbol{G}^T\boldsymbol{C}^{-1}\boldsymbol{G}\boldsymbol{\beta} = \boldsymbol{0}, \quad \hat{\boldsymbol{\beta}} = (\boldsymbol{G}^T\boldsymbol{C}^{-1}\boldsymbol{G})^{-1}\boldsymbol{G}^T\boldsymbol{C}^{-1}\boldsymbol{y}$$

Whereas for fixed $\boldsymbol{\beta}$, the MLE of nuisance parameters $\boldsymbol{\theta}$ can be obtained by numerical maximization. Let $\boldsymbol{z} = \boldsymbol{y} - \boldsymbol{G}\boldsymbol{\beta}$, and $\boldsymbol{Q} = \boldsymbol{C}^{-1}$. For each component of $\boldsymbol{\theta}_r$, $r = 1, \ldots, d$, in this case, θ has 3 components (σ^2, η, τ^2). The score of the log-likelihood becomes

$$\frac{dl}{d\theta_r} = -\frac{1}{2}\text{trace}(\boldsymbol{Q}\frac{d\boldsymbol{C}}{d\theta_r}) + \frac{1}{2}\boldsymbol{z}^T\boldsymbol{Q}\frac{d\boldsymbol{C}}{d\theta_r}\boldsymbol{Q}\boldsymbol{z}$$

The above mentioned score can be solved iteratively using Fisher scoring algorithm. To achieve the numerical stability, the expected Hessian is applied, which is

$$E(\frac{d^2l}{d\theta_r d\theta_{\bar{r}}}) = -\frac{1}{2}\text{trace}(\boldsymbol{Q}\frac{d\boldsymbol{C}}{d\theta_{\bar{r}}}\boldsymbol{Q}\frac{d\boldsymbol{C}}{d\theta_r})$$

The pseudo code for the Fisher scoring algorithm can then be expressed as follows:

Data: initial β_0, θ_0
Result: Converged $\hat{\beta}$, $\hat{\theta}$
while *not converged* **do**
  $\boldsymbol{C} = \boldsymbol{C}(\boldsymbol{\theta}^b)$;
  $\boldsymbol{\beta}^{b+1} = [\boldsymbol{G}^T\boldsymbol{C}^{-1}\boldsymbol{G}]^{-1}\boldsymbol{G}^T\boldsymbol{C}^{-1}\boldsymbol{y}$;
  $\boldsymbol{z} = \boldsymbol{y} - \boldsymbol{G}\boldsymbol{\beta}^{b+1}$;
  $\boldsymbol{Q} = \boldsymbol{C}^{-1}$;
  $\boldsymbol{C}_r^* = \frac{d\boldsymbol{C}(\boldsymbol{\theta}^b)}{d\theta_r}, \quad r = 1, \ldots, d$;
  $u_r = \frac{dl}{d\theta_r} = -\frac{1}{2}\text{trace}(\boldsymbol{Q}\boldsymbol{C}_r^*) + \frac{1}{2}\boldsymbol{z}^T\boldsymbol{Q}\boldsymbol{C}_r^*\boldsymbol{Q}\boldsymbol{z}$;
  $\boldsymbol{V}_{r\bar{r}} = E(\frac{d^2l}{d\theta_r d\theta_{\bar{r}}}) = -\frac{1}{2}\text{trace}(\boldsymbol{Q}\boldsymbol{C}_r^*\boldsymbol{Q}\boldsymbol{C}_{\bar{r}}^*)$;
  $\boldsymbol{\theta}^{b+1} = \boldsymbol{\theta}^b + \boldsymbol{V}^{-1}\boldsymbol{u}$;
  $b = b + 1$
end

```
## [1] "3.06394765464067  , iter no is  0"
## [1] "1.64091463270963  , iter no is  1"
## [1] "1.60801653018526  , iter no is  2"
## [1] "1.56270915985108  , iter no is  3"
## [1] "1.54272888106321  , iter no is  4"
## [1] "1.51788676453651  , iter no is  5"
## [1] "1.50958874939674  , iter no is  6"
## [1] "1.50342923560875  , iter no is  7"
## [1] "1.50560154872536  , iter no is  8"
## [1] "1.51756433934151  , iter no is  9"
## [1] "1.52747452409346  , iter no is  10"
## [1] "1.5565157596673  , iter no is  11"
## [1] "1.56907980430871  , iter no is  12"
## [1] "1.61240433904859  , iter no is  13"
## [1] "1.62094197361999  , iter no is  14"
## [1] "1.67359982894317  , iter no is  15"
## [1] "1.67232531588532  , iter no is  16"
## [1] "1.72832607614144  , iter no is  17"
## [1] "1.71474360515394  , iter no is  18"
## [1] "1.76913845290875  , iter no is  19"
## [1] "1.74425790999802  , iter no is  20"
## [1] "1.79449432635815  , iter no is  21"
## [1] "1.76118094286876  , iter no is  22"
## [1] "1.80688889019921  , iter no is  23"
## [1] "1.76815908524171  , iter no is  24"
## [1] "1.81012189806186  , iter no is  25"
## [1] "1.76832251187303  , iter no is  26"
## [1] "1.80760648632958  , iter no is  27"
## [1] "1.76429920865291  , iter no is  28"
## [1] "1.80183164106068  , iter no is  29"
## [1] "1.75796140330358  , iter no is  30"
## [1] "1.79441993692024  , iter no is  31"
## [1] "1.75052058211967  , iter no is  32"
## [1] "1.78635716085785  , iter no is  33"
## [1] "1.74271140172132  , iter no is  34"
## [1] "1.77821116329649  , iter no is  35"
## [1] "1.73495628178073  , iter no is  36"
## [1] "1.77029234668556  , iter no is  37"
## [1] "1.72748457834246  , iter no is  38"
## [1] "1.76275882415606  , iter no is  39"
## [1] "1.72041076562503  , iter no is  40"
## [1] "1.75568129853456  , iter no is  41"
## [1] "1.71378297780637  , iter no is  42"
## [1] "1.7490816578311  , iter no is  43"
## [1] "1.7076121283723  , iter no is  44"
## [1] "1.74295537849794  , iter no is  45"
## [1] "1.70188896725894  , iter no is  46"
## [1] "1.73728428996301  , iter no is  47"
## [1] "1.6965938887692  , iter no is  48"
## [1] "1.73204372951571  , iter no is  49"
## [1] "1.69170247739839  , iter no is  50"
## [1] "1.72720648486217  , iter no is  51"
## [1] "1.68718858748981  , iter no is  52"
## [1] "1.72274492046219  , iter no is  53"
```

```
## [1] "1.68302601387362  , iter no is  54"
## [1] "1.71863208835135  , iter no is  55"
## [1] "1.67918936642378  , iter no is  56"
## [1] "1.71484227731307  , iter no is  57"
## [1] "1.67565449979793  , iter no is  58"
## [1] "1.71135125518107  , iter no is  59"
## [1] "1.67239869773658  , iter no is  60"
## [1] "1.7081363460662  , iter no is  61"
## [1] "1.66940072413446  , iter no is  62"
## [1] "1.70517642077192  , iter no is  63"
## [1] "1.66664080352513  , iter no is  64"
## [1] "1.70245184322263  , iter no is  65"
## [1] "1.66410056565195  , iter no is  66"
## [1] "1.69994439611452  , iter no is  67"
## [1] "1.66176297313923  , iter no is  68"
## [1] "1.69763719821609  , iter no is  69"
## [1] "1.65961224255939  , iter no is  70"
## [1] "1.69551461986368  , iter no is  71"
## [1] "1.65763376438925  , iter no is  72"
## [1] "1.69356220001364  , iter no is  73"
## [1] "1.65581402470162  , iter no is  74"
## [1] "1.69176656651182  , iter no is  75"
## [1] "1.65414053001356  , iter no is  76"
## [1] "1.69011536032498  , iter no is  77"
## [1] "1.65260173593277  , iter no is  78"
## [1] "1.68859716401274  , iter no is  79"
## [1] "1.65118697983806  , iter no is  80"
## [1] "1.68720143447249  , iter no is  81"
## [1] "1.64988641761311  , iter no is  82"
## [1] "1.68591843986136  , iter no is  83"
## [1] "1.64869096434037  , iter no is  84"
## [1] "1.68473920054452  , iter no is  85"
## [1] "1.64759223880869  , iter no is  86"
## [1] "1.68365543389366  , iter no is  87"
## [1] "1.64658251166552  , iter no is  88"
## [1] "1.68265950274227  , iter no is  89"
## [1] "1.64565465702962  , iter no is  90"
## [1] "1.68174436730837  , iter no is  91"
## [1] "1.6448021073831  , iter no is  92"
## [1] "1.68090354039747  , iter no is  93"
## [1] "1.64401881156066  , iter no is  94"
## [1] "1.6801310456987  , iter no is  95"
## [1] "1.64329919565938  , iter no is  96"
## [1] "1.67942137899569  , iter no is  97"
## [1] "1.64263812669862  , iter no is  98"
## [1] "1.67876947212088  , iter no is  99"
## [1] "1.64203087886297  , iter no is  100"
## [1] "1.67817065948396  , iter no is  101"
## [1] "1.64147310217042  , iter no is  102"
## [1] "1.67762064701766  , iter no is  103"
## [1] "1.64096079341409  , iter no is  104"
## [1] "1.67711548338452  , iter no is  105"
## [1] "1.64049026922994  , iter no is  106"
## [1] "1.67665153330096  , iter no is  107"
```

```
## [1] "1.64005814115275  , iter no is  108"
## [1] "1.67622545283904  , iter no is  109"
## [1] "1.63966129252853  , iter no is  110"
## [1] "1.67583416657063  , iter no is  111"
## [1] "1.63929685715731  , iter no is  112"
## [1] "1.67547484643431  , iter no is  113"
## [1] "1.63896219954789  , iter no is  114"
## [1] "1.6751448921993  , iter no is  115"
## [1] "1.63865489667268  , iter no is  116"
## [1] "1.67484191342051  , iter no is  117"
## [1] "1.63837272111698  , iter no is  118"
## [1] "1.67456371277511  , iter no is  119"
## [1] "1.63811362552219  , iter no is  120"
## [1] "1.67430827068447  , iter no is  121"
## [1] "1.63787572823293  , iter no is  122"
## [1] "1.674073731126  , iter no is  123"
## [1] "1.6376573000549  , iter no is  124"
## [1] "1.67385838854833  , iter no is  125"
## [1] "1.6374567520477  , iter no is  126"
## [1] "1.6736606758107  , iter no is  127"
## [1] "1.63727262427183  , iter no is  128"
## [1] "1.67347915306507  , iter no is  129"
## [1] "1.63710357541921  , iter no is  130"
## [1] "1.67331249751692  , iter no is  131"
## [1] "1.63694837326319  , iter no is  132"
## [1] "1.67315949399158  , iter no is  133"
## [1] "1.63680588586144  , iter no is  134"
## [1] "1.67301902625044  , iter no is  135"
## [1] "1.636675073459  , iter no is  136"
## [1] "1.67289006899486  , iter no is  137"
## [1] "1.63655498103309  , iter no is  138"
## [1] "1.6727716805083  , iter no is  139"
## [1] "1.63644473143411  , iter no is  140"
## [1] "1.6726629958864  , iter no is  141"
## [1] "1.63634351907501  , iter no is  142"
## [1] "1.67256322080713  , iter no is  143"
## [1] "1.63625060412467  , iter no is  144"
## [1] "1.67247162580186  , iter no is  145"
## [1] "1.63616530716815  , iter no is  146"
## [1] "1.67238754098671  , iter no is  147"
## [1] "1.63608700429692  , iter no is  148"
## [1] "1.67231035121709  , iter no is  149"
## [1] "1.6360151225932  , iter no is  150"
## [1] "1.67223949163246  , iter no is  151"
## [1] "1.63594913597935  , iter no is  152"
## [1] "1.67217444356331  , iter no is  153"
## [1] "1.6358885614029  , iter no is  154"
## [1] "1.67211473076638  , iter no is  155"
## [1] "1.635832955329  , iter no is  156"
## [1] "1.67205991596435  , iter no is  157"
## [1] "1.63578191051802  , iter no is  158"
## [1] "1.67200959766941  , iter no is  159"
## [1] "1.6357350530639  , iter no is  160"
## [1] "1.67196340725953  , iter no is  161"
```

```
## [1] "1.63569203967222  , iter no is  162"
## [1] "1.67192100629663  , iter no is  163"
## [1] "1.635652555161  , iter no is  164"
## [1] "1.67188208405997  , iter no is  165"
## [1] "1.63561631016311  , iter no is  166"
## [1] "1.67184635528052  , iter no is  167"
## [1] "1.63558303901625  , iter no is  168"
## [1] "1.67181355806142  , iter no is  169"
## [1] "1.63555249782465  , iter no is  170"
## [1] "1.67178345196493  , iter no is  171"
## [1] "1.63552446267918  , iter no is  172"
## [1] "1.67175581625865  , iter no is  173"
## [1] "1.63549872802148  , iter no is  174"
## [1] "1.6717304483017  , iter no is  175"
## [1] "1.63547510514256  , iter no is  176"
## [1] "1.67170716206457  , iter no is  177"
## [1] "1.63545342080312  , iter no is  178"
## [1] "1.67168578676881  , iter no is  179"
## [1] "1.63543351596767  , iter no is  180"
## [1] "1.67166616563846  , iter no is  181"
## [1] "1.63541524464021  , iter no is  182"
## [1] "1.67164815475324  , iter no is  183"
## [1] "1.63539847279734  , iter no is  184"
## [1] "1.67163162199492  , iter no is  185"
## [1] "1.63538307740636  , iter no is  186"
## [1] "1.67161644608088  , iter no is  187"
## [1] "1.63536894552553  , iter no is  188"
## [1] "1.67160251567602  , iter no is  189"
## [1] "1.63535597347679  , iter no is  190"
## [1] "1.6715897285773  , iter no is  191"
## [1] "1.63534406608617  , iter no is  192"
## [1] "1.67157799096567  , iter no is  193"
## [1] "1.63533313598722  , iter no is  194"
## [1] "1.67156721671821  , iter no is  195"
## [1] "1.63532310298071  , iter no is  196"
## [1] "1.67155732677728  , iter no is  197"
## [1] "1.63531389344704  , iter no is  198"
## [1] "1.67154824857187  , iter no is  199"
## [1] "1.63530543980634  , iter no is  200"
## [1] "1.67153991548455  , iter no is  201"
## [1] "1.63529768002386  , iter no is  202"
## [1] "1.67153226636431  , iter no is  203"
## [1] "1.6352905571549  , iter no is  204"
## [1] "1.67152524507756  , iter no is  205"
## [1] "1.63528401892734  , iter no is  206"
## [1] "1.67151880009667  , iter no is  207"
## [1] "1.63527801735797  , iter no is  208"
## [1] "1.67151288412248  , iter no is  209"
## [1] "1.63527250840128  , iter no is  210"
## [1] "1.67150745373693  , iter no is  211"
## [1] "1.63526745162625  , iter no is  212"
## [1] "1.67150246908534  , iter no is  213"
## [1] "1.63526280991964  , iter no is  214"
## [1] "1.67149789358341  , iter no is  215"
```

```
## [1] "1.63525854921396  , iter no is  216"
## [1] "1.67149369364936  , iter no is  217"
## [1] "1.63525463823769  , iter no is  218"
## [1] "1.67148983845734  , iter no is  219"
## [1] "1.63525104828565  , iter no is  220"
## [1] "1.67148629971138  , iter no is  221"
## [1] "1.63524775300846  , iter no is  222"
## [1] "1.67148305143755  , iter no is  223"
## [1] "1.63524472821905  , iter no is  224"
## [1] "1.67148006979404  , iter no is  225"
## [1] "1.63524195171601  , iter no is  226"
## [1] "1.67147733289553  , iter no is  227"
## [1] "1.6352394031195  , iter no is  228"
## [1] "1.67147482065327  , iter no is  229"
## [1] "1.63523706372291  , iter no is  230"
## [1] "1.67147251462697  , iter no is  231"
## [1] "1.63523491635458  , iter no is  232"
## [1] "1.67147039789003  , iter no is  233"
## [1] "1.63523294525236  , iter no is  234"
## [1] "1.67146845490528  , iter no is  235"
## [1] "1.63523113594809  , iter no is  236"
## [1] "1.67146667141054  , iter no is  237"
## [1] "1.63522947516081  , iter no is  238"
## [1] "1.67146503431424  , iter no is  239"
## [1] "1.63522795069958  , iter no is  240"
## [1] "1.67146353159942  , iter no is  241"
## [1] "1.63522655137427  , iter no is  242"
## [1] "1.67146215223576  , iter no is  243"
## [1] "1.63522526691337  , iter no is  244"
## [1] "1.67146088609791  , iter no is  245"
## [1] "1.63522408788829  , iter no is  246"
## [1] "1.67145972389191  , iter no is  247"
## [1] "1.63522300564426  , iter no is  248"
## [1] "1.67145865708666  , iter no is  249"
## [1] "1.63522201223747  , iter no is  250"
## [1] "1.67145767785106  , iter no is  251"
## [1] "1.63522110037543  , iter no is  252"
## [1] "1.67145677899729  , iter no is  253"
## [1] "1.63522026336462  , iter no is  254"
## [1] "1.67145595392664  , iter no is  255"
## [1] "1.6352194950607  , iter no is  256"
## [1] "1.67145519658304  , iter no is  257"
## [1] "1.63521878982374  , iter no is  258"
## [1] "1.67145450140665  , iter no is  259"
## [1] "1.63521814247704  , iter no is  260"
## [1] "1.67145386329478  , iter no is  261"
## [1] "1.63521754826884  , iter no is  262"
## [1] "1.67145327756341  , iter no is  263"
## [1] "1.63521700283698  , iter no is  264"
## [1] "1.67145273991245  , iter no is  265"
## [1] "1.63521650217766  , iter no is  266"
## [1] "1.67145224639541  , iter no is  267"
## [1] "1.63521604261581  , iter no is  268"
## [1] "1.67145179338956  , iter no is  269"
```

```
## [1] "1.63521562077774  , iter no is  270"
## [1] "1.67145137756939  , iter no is  271"
## [1] "1.63521523356699  , iter no is  272"
## [1] "1.67145099588244  , iter no is  273"
## [1] "1.63521487814107  , iter no is  274"
## [1] "1.67145064552706  , iter no is  275"
## [1] "1.63521455189088  , iter no is  276"
## [1] "1.67145032393101  , iter no is  277"
## [1] "1.63521425242146  , iter no is  278"
## [1] "1.67145002873375  , iter no is  279"
## [1] "1.63521397753449  , iter no is  280"
## [1] "1.67144975776839  , iter no is  281"
## [1] "1.63521372521213  , iter no is  282"
## [1] "1.67144950904558  , iter no is  283"
## [1] "1.63521349360205  , iter no is  284"
## [1] "1.67144928073963  , iter no is  285"
## [1] "1.63521328100415  , iter no is  286"
## [1] "1.6714490711746  , iter no is  287"
## [1] "1.63521308585764  , iter no is  288"
## [1] "1.67144887881203  , iter no is  289"
## [1] "1.63521290673006  , iter no is  290"
## [1] "1.67144870223979  , iter no is  291"
## [1] "1.63521274230648  , iter no is  292"
## [1] "1.67144854016202  , iter no is  293"
## [1] "1.63521259137996  , iter no is  294"
## [1] "1.67144839138849  , iter no is  295"
## [1] "1.63521245284252  , iter no is  296"
## [1] "1.6714482548275  , iter no is  297"
## [1] "1.63521232567725  , iter no is  298"
## [1] "1.67144812947643  , iter no is  299"
## [1] "1.63521220895064  , iter no is  300"
## [1] "1.67144801441477  , iter no is  301"
## [1] "1.63521210180551  , iter no is  302"
## [1] "1.67144790879838  , iter no is  303"
## [1] "1.63521200345589  , iter no is  304"
## [1] "1.67144781185178  , iter no is  305"
## [1] "1.63521191317941  , iter no is  306"
## [1] "1.67144772286314  , iter no is  307"
## [1] "1.63521183031329  , iter no is  308"
## [1] "1.67144764117931  , iter no is  309"
## [1] "1.63521175424959  , iter no is  310"
## [1] "1.67144756620061  , iter no is  311"
## [1] "1.63521168442953  , iter no is  312"
## [1] "1.67144749737661  , iter no is  313"
## [1] "1.63521162034082  , iter no is  314"
## [1] "1.67144743420214  , iter no is  315"
## [1] "1.63521156151297  , iter no is  316"
## [1] "1.67144737621346  , iter no is  317"
## [1] "1.63521150751416  , iter no is  318"
## [1] "1.67144732298514  , iter no is  319"
## [1] "1.63521145794802  , iter no is  320"
## [1] "1.67144727412626  , iter no is  321"
## [1] "1.63521141245068  , iter no is  322"
## [1] "1.67144722927779  , iter no is  323"
```

```
## [1] "1.63521137068791  , iter no is  324"
## [1] "1.67144718811086  , iter no is  325"
## [1] "1.63521133235337  , iter no is  326"
## [1] "1.67144715032305  , iter no is  327"
## [1] "1.63521129716545  , iter no is  328"
## [1] "1.6714471156372  , iter no is  329"
## [1] "1.63521126486603  , iter no is  330"
## [1] "1.67144708379871  , iter no is  331"
## [1] "1.63521123521814  , iter no is  332"
## [1] "1.67144705457364  , iter no is  333"
## [1] "1.63521120800388  , iter no is  334"
## [1] "1.67144702774769  , iter no is  335"
## [1] "1.63521118302361  , iter no is  336"
## [1] "1.6714470031237  , iter no is  337"
## [1] "1.63521116009377  , iter no is  338"
## [1] "1.67144698052111  , iter no is  339"
## [1] "1.63521113904632  , iter no is  340"
## [1] "1.67144695977397  , iter no is  341"
## [1] "1.63521111972666  , iter no is  342"
## [1] "1.67144694072982  , iter no is  343"
## [1] "1.63521110199266  , iter no is  344"
## [1] "1.67144692324882  , iter no is  345"
## [1] "1.63521108571452  , iter no is  346"
## [1] "1.67144690720279  , iter no is  347"
## [1] "1.6352110707725  , iter no is  348"
## [1] "1.67144689247409  , iter no is  349"
## [1] "1.63521105705721  , iter no is  350"
## [1] "1.67144687895436  , iter no is  351"
## [1] "1.63521104446759  , iter no is  352"
## [1] "1.67144686654443  , iter no is  353"
## [1] "1.63521103291149  , iter no is  354"
## [1] "1.67144685515306  , iter no is  355"
## [1] "1.63521102230395  , iter no is  356"
## [1] "1.67144684469703  , iter no is  357"
## [1] "1.63521101256726  , iter no is  358"
## [1] "1.67144683509919  , iter no is  359"
## [1] "1.63521100362979  , iter no is  360"
## [1] "1.67144682628918  , iter no is  361"
## [1] "1.63521099542593  , iter no is  362"
## [1] "1.67144681820238  , iter no is  363"
## [1] "1.63521098789549  , iter no is  364"
## [1] "1.67144681077937  , iter no is  365"
## [1] "1.63521098098328  , iter no is  366"
## [1] "1.67144680396568  , iter no is  367"
## [1] "1.63521097463833  , iter no is  368"
## [1] "1.67144679771137  , iter no is  369"
## [1] "1.63521096881438  , iter no is  370"
## [1] "1.67144679197024  , iter no is  371"
## [1] "1.63521096346825  , iter no is  372"
## [1] "1.67144678670054  , iter no is  373"
## [1] "1.63521095856105  , iter no is  374"
## [1] "1.67144678186343  , iter no is  375"
## [1] "1.63521095405677  , iter no is  376"
## [1] "1.67144677742343  , iter no is  377"
```

```
## [1] "1.63521094992227  , iter no is  378"
## [1] "1.67144677334797  , iter no is  379"
## [1] "1.63521094612721  , iter no is  380"
## [1] "1.67144676960677  , iter no is  381"
## [1] "1.63521094264345  , iter no is  382"
## [1] "1.67144676617291  , iter no is  383"
## [1] "1.63521093944578  , iter no is  384"
## [1] "1.67144676302102  , iter no is  385"
## [1] "1.63521093651066  , iter no is  386"
## [1] "1.67144676012752  , iter no is  387"
## [1] "1.63521093381639  , iter no is  388"
## [1] "1.67144675747176  , iter no is  389"
## [1] "1.63521093134334  , iter no is  390"
## [1] "1.671446755034  , iter no is  391"
## [1] "1.63521092907322  , iter no is  392"
## [1] "1.67144675279623  , iter no is  393"
## [1] "1.63521092698946  , iter no is  394"
## [1] "1.67144675074223  , iter no is  395"
## [1] "1.63521092507674  , iter no is  396"
## [1] "1.67144674885684  , iter no is  397"
## [1] "1.63521092332105  , iter no is  398"
## [1] "1.67144674712605  , iter no is  399"
## [1] "1.63521092170944  , iter no is  400"
## [1] "1.67144674553755  , iter no is  401"
## [1] "1.63521092023018  , iter no is  402"
## [1] "1.67144674407929  , iter no is  403"
## [1] "1.63521091887225  , iter no is  404"
## [1] "1.67144674274088  , iter no is  405"
## [1] "1.63521091762599  , iter no is  406"
## [1] "1.67144674151229  , iter no is  407"
## [1] "1.63521091648187  , iter no is  408"
## [1] "1.67144674038452  , iter no is  409"
## [1] "1.63521091543176  , iter no is  410"
## [1] "1.67144673934935  , iter no is  411"
## [1] "1.63521091446778  , iter no is  412"
## [1] "1.67144673839921  , iter no is  413"
## [1] "1.63521091358303  , iter no is  414"
## [1] "1.6714467375268  , iter no is  415"
## [1] "1.63521091277063  , iter no is  416"
## [1] "1.67144673672615  , iter no is  417"
## [1] "1.63521091202505  , iter no is  418"
## [1] "1.67144673599138  , iter no is  419"
## [1] "1.63521091134088  , iter no is  420"
## [1] "1.6714467353169  , iter no is  421"
## [1] "1.63521091071273  , iter no is  422"
## [1] "1.67144673469762  , iter no is  423"
## [1] "1.63521091013613  , iter no is  424"
## [1] "1.67144673412944  , iter no is  425"
## [1] "1.63521090960694  , iter no is  426"
## [1] "1.67144673360759  , iter no is  427"
## [1] "1.6352109091211  , iter no is  428"
## [1] "1.67144673312877  , iter no is  429"
## [1] "1.63521090867516  , iter no is  430"
## [1] "1.67144673268912  , iter no is  431"
```

```
## [1] "1.63521090826581  , iter no is  432"
## [1] "1.67144673228568  , iter no is  433"
## [1] "1.63521090789017  , iter no is  434"
## [1] "1.67144673191536  , iter no is  435"
## [1] "1.63521090754525  , iter no is  436"
## [1] "1.67144673157544  , iter no is  437"
## [1] "1.63521090722875  , iter no is  438"
## [1] "1.67144673126332  , iter no is  439"
## [1] "1.63521090693805  , iter no is  440"
## [1] "1.67144673097679  , iter no is  441"
## [1] "1.63521090667128  , iter no is  442"
## [1] "1.67144673071395  , iter no is  443"
## [1] "1.63521090642652  , iter no is  444"
## [1] "1.67144673047265  , iter no is  445"
## [1] "1.63521090620188  , iter no is  446"
## [1] "1.67144673025121  , iter no is  447"
## [1] "1.63521090599557  , iter no is  448"
## [1] "1.67144673004766  , iter no is  449"
## [1] "1.63521090580615  , iter no is  450"
## [1] "1.67144672986097  , iter no is  451"
## [1] "1.63521090563219  , iter no is  452"
## [1] "1.67144672968972  , iter no is  453"
## [1] "1.63521090547274  , iter no is  454"
## [1] "1.67144672953245  , iter no is  455"
## [1] "1.63521090532629  , iter no is  456"
## [1] "1.67144672938817  , iter no is  457"
## [1] "1.63521090519193  , iter no is  458"
## [1] "1.67144672925563  , iter no is  459"
## [1] "1.63521090506856  , iter no is  460"
## [1] "1.67144672913396  , iter no is  461"
## [1] "1.63521090495532  , iter no is  462"
## [1] "1.67144672902242  , iter no is  463"
## [1] "1.63521090485139  , iter no is  464"
## [1] "1.67144672891983  , iter no is  465"
## [1] "1.63521090475583  , iter no is  466"
## [1] "1.67144672882579  , iter no is  467"
## [1] "1.63521090466822  , iter no is  468"
## [1] "1.67144672873947  , iter no is  469"
## [1] "1.63521090458787  , iter no is  470"
## [1] "1.67144672866015  , iter no is  471"
## [1] "1.63521090451404  , iter no is  472"
## [1] "1.67144672858732  , iter no is  473"
## [1] "1.63521090444615  , iter no is  474"
## [1] "1.67144672852057  , iter no is  475"
## [1] "1.63521090438413  , iter no is  476"
## [1] "1.67144672845929  , iter no is  477"
## [1] "1.63521090432698  , iter no is  478"
## [1] "1.67144672840315  , iter no is  479"
## [1] "1.63521090427472  , iter no is  480"
## [1] "1.67144672835159  , iter no is  481"
## [1] "1.6352109042267  , iter no is  482"
## [1] "1.67144672830412  , iter no is  483"
## [1] "1.6352109041825  , iter no is  484"
## [1] "1.67144672826056  , iter no is  485"
```

```
## [1] "1.63521090414191  , iter no is  486"
## [1] "1.67144672822068  , iter no is  487"
## [1] "1.63521090410469  , iter no is  488"
## [1] "1.67144672818384  , iter no is  489"
## [1] "1.63521090407057  , iter no is  490"
## [1] "1.67144672815017  , iter no is  491"
## [1] "1.63521090403925  , iter no is  492"
## [1] "1.6714467281193  , iter no is  493"
## [1] "1.63521090401041  , iter no is  494"
## [1] "1.67144672809104  , iter no is  495"
## [1] "1.63521090398411  , iter no is  496"
## [1] "1.67144672806501  , iter no is  497"
## [1] "1.63521090395978  , iter no is  498"
## [1] "1.67144672804107  , iter no is  499"
## [1] "1.63521090393747  , iter no is  500"
## [1] "1.67144672801899  , iter no is  501"
## [1] "1.63521090391708  , iter no is  502"
## [1] "1.67144672799901  , iter no is  503"
## [1] "1.63521090389836  , iter no is  504"
## [1] "1.67144672798063  , iter no is  505"
## [1] "1.63521090388131  , iter no is  506"
## [1] "1.67144672796378  , iter no is  507"
## [1] "1.63521090386555  , iter no is  508"
## [1] "1.67144672794819  , iter no is  509"
## [1] "1.63521090385101  , iter no is  510"
## [1] "1.67144672793384  , iter no is  511"
## [1] "1.6352109038377  , iter no is  512"
## [1] "1.67144672792068  , iter no is  513"
## [1] "1.63521090382543  , iter no is  514"
## [1] "1.67144672790861  , iter no is  515"
## [1] "1.63521090381416  , iter no is  516"
## [1] "1.67144672789759  , iter no is  517"
## [1] "1.635210903804  , iter no is  518"
## [1] "1.67144672788742  , iter no is  519"
## [1] "1.63521090379442  , iter no is  520"
## [1] "1.67144672787808  , iter no is  521"
## [1] "1.6352109037856  , iter no is  522"
## [1] "1.67144672786939  , iter no is  523"
## [1] "1.63521090377766  , iter no is  524"
## [1] "1.67144672786154  , iter no is  525"
## [1] "1.63521090377041  , iter no is  526"
## [1] "1.67144672785446  , iter no is  527"
## [1] "1.63521090376374  , iter no is  528"
## [1] "1.67144672784783  , iter no is  529"
## [1] "1.63521090375769  , iter no is  530"
## [1] "1.67144672784192  , iter no is  531"
## [1] "1.6352109037521  , iter no is  532"
## [1] "1.67144672783629  , iter no is  533"
## [1] "1.63521090374681  , iter no is  534"
## [1] "1.67144672783122  , iter no is  535"
## [1] "1.63521090374215  , iter no is  536"
## [1] "1.67144672782653  , iter no is  537"
## [1] "1.63521090373775  , iter no is  538"
## [1] "1.67144672782224  , iter no is  539"
```

```
## [1] "1.63521090373373  , iter no is  540"
## [1] "1.67144672781815  , iter no is  541"
## [1] "1.63521090372999  , iter no is  542"
## [1] "1.67144672781465  , iter no is  543"
## [1] "1.63521090372662  , iter no is  544"
## [1] "1.67144672781127  , iter no is  545"
## [1] "1.63521090372352  , iter no is  546"
## [1] "1.67144672780813  , iter no is  547"
## [1] "1.63521090372063  , iter no is  548"
## [1] "1.67144672780534  , iter no is  549"
## [1] "1.63521090371808  , iter no is  550"
## [1] "1.67144672780277  , iter no is  551"
## [1] "1.6352109037156  , iter no is  552"
## [1] "1.67144672780049  , iter no is  553"
## [1] "1.63521090371354  , iter no is  554"
## [1] "1.67144672779819  , iter no is  555"
## [1] "1.63521090371148  , iter no is  556"
## [1] "1.67144672779618  , iter no is  557"
## [1] "1.63521090370944  , iter no is  558"
## [1] "1.67144672779441  , iter no is  559"
## [1] "1.63521090370781  , iter no is  560"
## [1] "1.6714467277927  , iter no is  561"
## [1] "1.63521090370625  , iter no is  562"
## [1] "1.67144672779114  , iter no is  563"
## [1] "1.63521090370475  , iter no is  564"
## [1] "1.6714467277897  , iter no is  565"
## [1] "1.63521090370346  , iter no is  566"
## [1] "1.67144672778848  , iter no is  567"
## [1] "1.63521090370233  , iter no is  568"
## [1] "1.67144672778735  , iter no is  569"
## [1] "1.63521090370121  , iter no is  570"
## [1] "1.67144672778635  , iter no is  571"
## [1] "1.63521090370029  , iter no is  572"
## [1] "1.67144672778546  , iter no is  573"
## [1] "1.63521090369943  , iter no is  574"
## [1] "1.67144672778438  , iter no is  575"
## [1] "1.63521090369852  , iter no is  576"
## [1] "1.67144672778347  , iter no is  577"
## [1] "1.63521090369759  , iter no is  578"
## [1] "1.67144672778259  , iter no is  579"
## [1] "1.63521090369689  , iter no is  580"
## [1] "1.67144672778195  , iter no is  581"
## [1] "1.63521090369626  , iter no is  582"
## [1] "1.67144672778151  , iter no is  583"
## [1] "1.63521090369578  , iter no is  584"
## [1] "1.6714467277807  , iter no is  585"
## [1] "1.63521090369504  , iter no is  586"
## [1] "1.67144672778009  , iter no is  587"
## [1] "1.63521090369459  , iter no is  588"
## [1] "1.67144672777969  , iter no is  589"
## [1] "1.63521090369418  , iter no is  590"
## [1] "1.67144672777921  , iter no is  591"
## [1] "1.63521090369374  , iter no is  592"
## [1] "1.67144672777871  , iter no is  593"
```

```
## [1] "1.63521090369333  , iter no is  594"
## [1] "1.6714467277784  , iter no is  595"
## [1] "1.63521090369287  , iter no is  596"
## [1] "1.67144672777788  , iter no is  597"
## [1] "1.63521090369245  , iter no is  598"
## [1] "1.67144672777742  , iter no is  599"
## [1] "1.63521090369204  , iter no is  600"
## [1] "1.67144672777709  , iter no is  601"
## [1] "1.63521090369174  , iter no is  602"
## [1] "1.67144672777686  , iter no is  603"
## [1] "1.63521090369152  , iter no is  604"
## [1] "1.67144672777668  , iter no is  605"
## [1] "1.63521090369144  , iter no is  606"
## [1] "1.67144672777651  , iter no is  607"
## [1] "1.6352109036913  , iter no is  608"
## [1] "1.67144672777633  , iter no is  609"
## [1] "1.635210903691  , iter no is  610"
## [1] "1.67144672777612  , iter no is  611"
## [1] "1.635210903691  , iter no is  612"
## [1] "1.6714467277761  , iter no is  613"
## [1] "1.63521090369072  , iter no is  614"
## [1] "1.67144672777569  , iter no is  615"
## [1] "1.63521090369049  , iter no is  616"
## [1] "1.67144672777571  , iter no is  617"
## [1] "1.63521090369041  , iter no is  618"
## [1] "1.67144672777559  , iter no is  619"
## [1] "1.63521090369031  , iter no is  620"
## [1] "1.67144672777548  , iter no is  621"
## [1] "1.63521090369017  , iter no is  622"
## [1] "1.67144672777541  , iter no is  623"
## [1] "1.63521090369008  , iter no is  624"
## [1] "1.67144672777516  , iter no is  625"
## [1] "1.63521090368991  , iter no is  626"
## [1] "1.6714467277751  , iter no is  627"
## [1] "1.63521090368996  , iter no is  628"
## [1] "1.67144672777514  , iter no is  629"
## [1] "1.63521090368993  , iter no is  630"
## [1] "1.67144672777503  , iter no is  631"
## [1] "1.63521090368969  , iter no is  632"
## [1] "1.67144672777486  , iter no is  633"
## [1] "1.63521090368948  , iter no is  634"
## [1] "1.67144672777463  , iter no is  635"
## [1] "1.63521090368951  , iter no is  636"
## [1] "1.67144672777464  , iter no is  637"
## [1] "1.63521090368945  , iter no is  638"
## [1] "1.67144672777467  , iter no is  639"
## [1] "1.63521090368942  , iter no is  640"
## [1] "1.67144672777467  , iter no is  641"
## [1] "1.63521090368953  , iter no is  642"
## [1] "1.67144672777474  , iter no is  643"
## [1] "1.63521090368951  , iter no is  644"
## [1] "1.6714467277746  , iter no is  645"
## [1] "1.63521090368944  , iter no is  646"
## [1] "1.67144672777457  , iter no is  647"
```

```
## [1] "1.63521090368934  , iter no is  648"
## [1] "1.67144672777448  , iter no is  649"
## [1] "1.63521090368929  , iter no is  650"
## [1] "1.67144672777441  , iter no is  651"
## [1] "1.63521090368911  , iter no is  652"
## [1] "1.67144672777426  , iter no is  653"
## [1] "1.63521090368904  , iter no is  654"
## [1] "1.67144672777427  , iter no is  655"
## [1] "1.635210903689  , iter no is  656"
## [1] "1.67144672777418  , iter no is  657"
## [1] "1.63521090368905  , iter no is  658"
## [1] "1.67144672777414  , iter no is  659"
## [1] "1.63521090368896  , iter no is  660"
## [1] "1.67144672777425  , iter no is  661"
## [1] "1.6352109036891  , iter no is  662"
## [1] "1.6714467277742  , iter no is  663"
## [1] "1.63521090368904  , iter no is  664"
## [1] "1.6714467277742  , iter no is  665"
## [1] "1.63521090368909  , iter no is  666"
## [1] "1.67144672777427  , iter no is  667"
## [1] "1.63521090368901  , iter no is  668"
## [1] "1.6714467277743  , iter no is  669"
## [1] "1.63521090368901  , iter no is  670"
## [1] "1.67144672777416  , iter no is  671"
## [1] "1.63521090368904  , iter no is  672"
## [1] "1.67144672777426  , iter no is  673"
## [1] "1.63521090368913  , iter no is  674"
## [1] "1.67144672777409  , iter no is  675"
## [1] "1.63521090368899  , iter no is  676"
## [1] "1.67144672777418  , iter no is  677"
## [1] "1.63521090368901  , iter no is  678"
## [1] "1.67144672777416  , iter no is  679"
## [1] "1.63521090368897  , iter no is  680"
## [1] "1.67144672777407  , iter no is  681"
## [1] "1.63521090368891  , iter no is  682"
## [1] "1.67144672777408  , iter no is  683"
## [1] "1.63521090368895  , iter no is  684"
## [1] "1.67144672777409  , iter no is  685"
## [1] "1.63521090368889  , iter no is  686"
## [1] "1.67144672777387  , iter no is  687"
## [1] "1.63521090368864  , iter no is  688"
## [1] "1.67144672777379  , iter no is  689"
## [1] "1.63521090368865  , iter no is  690"
## [1] "1.67144672777387  , iter no is  691"
## [1] "1.63521090368878  , iter no is  692"
## [1] "1.67144672777387  , iter no is  693"
## [1] "1.63521090368857  , iter no is  694"
## [1] "1.67144672777373  , iter no is  695"
## [1] "1.63521090368859  , iter no is  696"
## [1] "1.67144672777393  , iter no is  697"
## [1] "1.63521090368876  , iter no is  698"
## [1] "1.671446727774  , iter no is  699"
## [1] "1.63521090368883  , iter no is  700"
## [1] "1.67144672777389  , iter no is  701"
```

```
## [1] "1.63521090368871  , iter no is  702"
## [1] "1.67144672777387  , iter no is  703"
## [1] "1.63521090368871  , iter no is  704"
## [1] "1.67144672777384  , iter no is  705"
## [1] "1.63521090368875  , iter no is  706"
## [1] "1.67144672777392  , iter no is  707"
## [1] "1.63521090368872  , iter no is  708"
## [1] "1.67144672777402  , iter no is  709"
## [1] "1.63521090368886  , iter no is  710"
## [1] "1.67144672777398  , iter no is  711"
## [1] "1.63521090368885  , iter no is  712"
## [1] "1.67144672777398  , iter no is  713"
## [1] "1.63521090368882  , iter no is  714"
## [1] "1.67144672777397  , iter no is  715"
## [1] "1.6352109036888  , iter no is  716"
## [1] "1.67144672777396  , iter no is  717"
## [1] "1.63521090368879  , iter no is  718"
## [1] "1.67144672777397  , iter no is  719"
## [1] "1.63521090368879  , iter no is  720"
## [1] "1.67144672777389  , iter no is  721"
## [1] "1.63521090368868  , iter no is  722"
## [1] "1.67144672777388  , iter no is  723"
## [1] "1.63521090368871  , iter no is  724"
## [1] "1.67144672777393  , iter no is  725"
## [1] "1.63521090368877  , iter no is  726"
## [1] "1.67144672777388  , iter no is  727"
## [1] "1.63521090368877  , iter no is  728"
## [1] "1.67144672777391  , iter no is  729"
## [1] "1.63521090368873  , iter no is  730"
## [1] "1.67144672777387  , iter no is  731"
## [1] "1.63521090368868  , iter no is  732"
## [1] "1.67144672777385  , iter no is  733"
## [1] "1.63521090368873  , iter no is  734"
## [1] "1.67144672777395  , iter no is  735"
## [1] "1.63521090368887  , iter no is  736"
## [1] "1.67144672777409  , iter no is  737"
## [1] "1.63521090368894  , iter no is  738"
## [1] "1.67144672777398  , iter no is  739"
## [1] "1.63521090368884  , iter no is  740"
## [1] "1.67144672777403  , iter no is  741"
## [1] "1.63521090368893  , iter no is  742"
## [1] "1.6714467277741  , iter no is  743"
## [1] "1.63521090368896  , iter no is  744"
## [1] "1.67144672777412  , iter no is  745"
## [1] "1.63521090368888  , iter no is  746"
## [1] "1.67144672777419  , iter no is  747"
## [1] "1.63521090368894  , iter no is  748"
## [1] "1.67144672777413  , iter no is  749"
## [1] "1.63521090368892  , iter no is  750"
## [1] "1.67144672777409  , iter no is  751"
## [1] "1.63521090368894  , iter no is  752"
## [1] "1.67144672777409  , iter no is  753"
## [1] "1.635210903689  , iter no is  754"
## [1] "1.67144672777406  , iter no is  755"
```

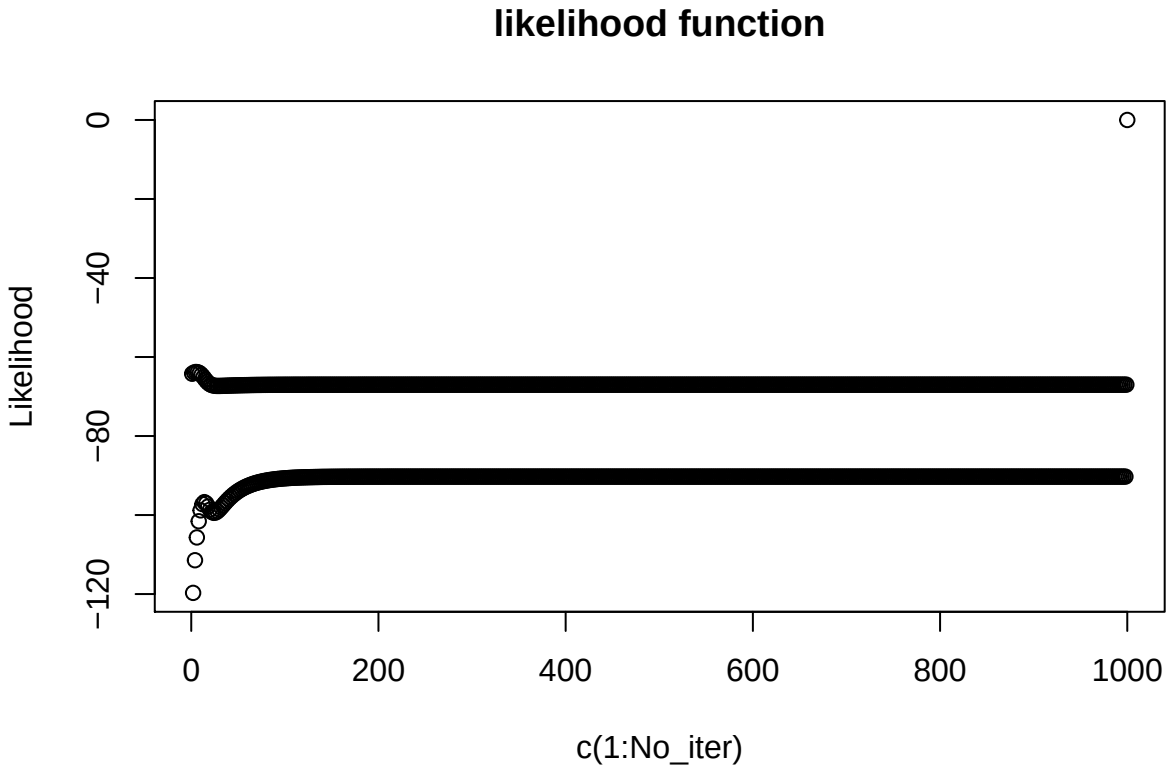
```
## [1] "1.63521090368893  , iter no is  756"
## [1] "1.67144672777408  , iter no is  757"
## [1] "1.63521090368895  , iter no is  758"
## [1] "1.67144672777412  , iter no is  759"
## [1] "1.63521090368886  , iter no is  760"
## [1] "1.67144672777409  , iter no is  761"
## [1] "1.63521090368891  , iter no is  762"
## [1] "1.67144672777416  , iter no is  763"
## [1] "1.63521090368896  , iter no is  764"
## [1] "1.67144672777405  , iter no is  765"
## [1] "1.63521090368879  , iter no is  766"
## [1] "1.6714467277739  , iter no is  767"
## [1] "1.63521090368888  , iter no is  768"
## [1] "1.67144672777388  , iter no is  769"
## [1] "1.63521090368871  , iter no is  770"
## [1] "1.67144672777401  , iter no is  771"
## [1] "1.63521090368879  , iter no is  772"
## [1] "1.67144672777402  , iter no is  773"
## [1] "1.63521090368879  , iter no is  774"
## [1] "1.67144672777383  , iter no is  775"
## [1] "1.63521090368872  , iter no is  776"
## [1] "1.6714467277738  , iter no is  777"
## [1] "1.63521090368859  , iter no is  778"
## [1] "1.67144672777383  , iter no is  779"
## [1] "1.63521090368871  , iter no is  780"
## [1] "1.67144672777404  , iter no is  781"
## [1] "1.63521090368881  , iter no is  782"
## [1] "1.67144672777406  , iter no is  783"
## [1] "1.63521090368895  , iter no is  784"
## [1] "1.67144672777391  , iter no is  785"
## [1] "1.63521090368872  , iter no is  786"
## [1] "1.67144672777402  , iter no is  787"
## [1] "1.63521090368881  , iter no is  788"
## [1] "1.67144672777402  , iter no is  789"
## [1] "1.63521090368883  , iter no is  790"
## [1] "1.6714467277739  , iter no is  791"
## [1] "1.63521090368874  , iter no is  792"
## [1] "1.67144672777401  , iter no is  793"
## [1] "1.63521090368885  , iter no is  794"
## [1] "1.67144672777393  , iter no is  795"
## [1] "1.6352109036888  , iter no is  796"
## [1] "1.67144672777396  , iter no is  797"
## [1] "1.63521090368885  , iter no is  798"
## [1] "1.67144672777398  , iter no is  799"
## [1] "1.63521090368891  , iter no is  800"
## [1] "1.67144672777403  , iter no is  801"
## [1] "1.63521090368894  , iter no is  802"
## [1] "1.6714467277741  , iter no is  803"
## [1] "1.6352109036889  , iter no is  804"
## [1] "1.67144672777402  , iter no is  805"
## [1] "1.63521090368892  , iter no is  806"
## [1] "1.67144672777411  , iter no is  807"
## [1] "1.63521090368882  , iter no is  808"
## [1] "1.67144672777397  , iter no is  809"
```

```
## [1] "1.63521090368881  , iter no is  810"
## [1] "1.67144672777393  , iter no is  811"
## [1] "1.6352109036887  , iter no is  812"
## [1] "1.67144672777392  , iter no is  813"
## [1] "1.63521090368874  , iter no is  814"
## [1] "1.67144672777393  , iter no is  815"
## [1] "1.63521090368876  , iter no is  816"
## [1] "1.67144672777392  , iter no is  817"
## [1] "1.63521090368884  , iter no is  818"
## [1] "1.67144672777386  , iter no is  819"
## [1] "1.6352109036887  , iter no is  820"
## [1] "1.67144672777386  , iter no is  821"
## [1] "1.63521090368872  , iter no is  822"
## [1] "1.671446727774  , iter no is  823"
## [1] "1.63521090368884  , iter no is  824"
## [1] "1.67144672777395  , iter no is  825"
## [1] "1.63521090368885  , iter no is  826"
## [1] "1.67144672777397  , iter no is  827"
## [1] "1.6352109036888  , iter no is  828"
## [1] "1.67144672777395  , iter no is  829"
## [1] "1.63521090368882  , iter no is  830"
## [1] "1.67144672777394  , iter no is  831"
## [1] "1.63521090368881  , iter no is  832"
## [1] "1.6714467277739  , iter no is  833"
## [1] "1.63521090368876  , iter no is  834"
## [1] "1.67144672777382  , iter no is  835"
## [1] "1.6352109036887  , iter no is  836"
## [1] "1.67144672777388  , iter no is  837"
## [1] "1.63521090368865  , iter no is  838"
## [1] "1.67144672777375  , iter no is  839"
## [1] "1.63521090368873  , iter no is  840"
## [1] "1.671446727774  , iter no is  841"
## [1] "1.63521090368873  , iter no is  842"
## [1] "1.67144672777392  , iter no is  843"
## [1] "1.63521090368875  , iter no is  844"
## [1] "1.67144672777402  , iter no is  845"
## [1] "1.63521090368884  , iter no is  846"
## [1] "1.67144672777398  , iter no is  847"
## [1] "1.63521090368886  , iter no is  848"
## [1] "1.67144672777391  , iter no is  849"
## [1] "1.63521090368879  , iter no is  850"
## [1] "1.67144672777391  , iter no is  851"
## [1] "1.63521090368878  , iter no is  852"
## [1] "1.67144672777402  , iter no is  853"
## [1] "1.63521090368894  , iter no is  854"
## [1] "1.67144672777406  , iter no is  855"
## [1] "1.63521090368883  , iter no is  856"
## [1] "1.67144672777392  , iter no is  857"
## [1] "1.63521090368873  , iter no is  858"
## [1] "1.671446727774  , iter no is  859"
## [1] "1.63521090368883  , iter no is  860"
## [1] "1.67144672777398  , iter no is  861"
## [1] "1.6352109036888  , iter no is  862"
## [1] "1.67144672777398  , iter no is  863"
```

```
## [1] "1.6352109036888  , iter no is  864"
## [1] "1.67144672777398  , iter no is  865"
## [1] "1.6352109036888  , iter no is  866"
## [1] "1.67144672777405  , iter no is  867"
## [1] "1.6352109036888  , iter no is  868"
## [1] "1.67144672777395  , iter no is  869"
## [1] "1.6352109036888  , iter no is  870"
## [1] "1.67144672777403  , iter no is  871"
## [1] "1.63521090368876  , iter no is  872"
## [1] "1.67144672777385  , iter no is  873"
## [1] "1.6352109036887  , iter no is  874"
## [1] "1.671446727774  , iter no is  875"
## [1] "1.63521090368891  , iter no is  876"
## [1] "1.67144672777394  , iter no is  877"
## [1] "1.63521090368884  , iter no is  878"
## [1] "1.67144672777412  , iter no is  879"
## [1] "1.63521090368893  , iter no is  880"
## [1] "1.67144672777412  , iter no is  881"
## [1] "1.63521090368889  , iter no is  882"
## [1] "1.67144672777388  , iter no is  883"
## [1] "1.63521090368871  , iter no is  884"
## [1] "1.6714467277739  , iter no is  885"
## [1] "1.63521090368879  , iter no is  886"
## [1] "1.67144672777399  , iter no is  887"
## [1] "1.63521090368887  , iter no is  888"
## [1] "1.67144672777401  , iter no is  889"
## [1] "1.63521090368877  , iter no is  890"
## [1] "1.67144672777377  , iter no is  891"
## [1] "1.63521090368881  , iter no is  892"
## [1] "1.67144672777392  , iter no is  893"
## [1] "1.63521090368873  , iter no is  894"
## [1] "1.67144672777397  , iter no is  895"
## [1] "1.63521090368899  , iter no is  896"
## [1] "1.67144672777416  , iter no is  897"
## [1] "1.63521090368898  , iter no is  898"
## [1] "1.6714467277741  , iter no is  899"
## [1] "1.63521090368886  , iter no is  900"
## [1] "1.67144672777389  , iter no is  901"
## [1] "1.63521090368874  , iter no is  902"
## [1] "1.67144672777387  , iter no is  903"
## [1] "1.63521090368879  , iter no is  904"
## [1] "1.671446727774  , iter no is  905"
## [1] "1.63521090368887  , iter no is  906"
## [1] "1.6714467277741  , iter no is  907"
## [1] "1.63521090368896  , iter no is  908"
## [1] "1.67144672777404  , iter no is  909"
## [1] "1.63521090368891  , iter no is  910"
## [1] "1.67144672777407  , iter no is  911"
## [1] "1.63521090368893  , iter no is  912"
## [1] "1.67144672777406  , iter no is  913"
## [1] "1.63521090368892  , iter no is  914"
## [1] "1.6714467277741  , iter no is  915"
## [1] "1.63521090368887  , iter no is  916"
## [1] "1.67144672777391  , iter no is  917"
```

```
## [1] "1.63521090368872  , iter no is  918"
## [1] "1.67144672777395  , iter no is  919"
## [1] "1.6352109036888  , iter no is  920"
## [1] "1.67144672777406  , iter no is  921"
## [1] "1.63521090368893  , iter no is  922"
## [1] "1.67144672777415  , iter no is  923"
## [1] "1.63521090368893  , iter no is  924"
## [1] "1.67144672777414  , iter no is  925"
## [1] "1.63521090368891  , iter no is  926"
## [1] "1.67144672777399  , iter no is  927"
## [1] "1.63521090368887  , iter no is  928"
## [1] "1.67144672777392  , iter no is  929"
## [1] "1.63521090368874  , iter no is  930"
## [1] "1.67144672777405  , iter no is  931"
## [1] "1.63521090368885  , iter no is  932"
## [1] "1.67144672777407  , iter no is  933"
## [1] "1.63521090368893  , iter no is  934"
## [1] "1.67144672777411  , iter no is  935"
## [1] "1.63521090368894  , iter no is  936"
## [1] "1.67144672777416  , iter no is  937"
## [1] "1.63521090368892  , iter no is  938"
## [1] "1.67144672777405  , iter no is  939"
## [1] "1.63521090368894  , iter no is  940"
## [1] "1.67144672777413  , iter no is  941"
## [1] "1.63521090368892  , iter no is  942"
## [1] "1.67144672777404  , iter no is  943"
## [1] "1.63521090368888  , iter no is  944"
## [1] "1.67144672777391  , iter no is  945"
## [1] "1.63521090368874  , iter no is  946"
## [1] "1.67144672777408  , iter no is  947"
## [1] "1.63521090368896  , iter no is  948"
## [1] "1.67144672777405  , iter no is  949"
## [1] "1.63521090368896  , iter no is  950"
## [1] "1.67144672777415  , iter no is  951"
## [1] "1.63521090368896  , iter no is  952"
## [1] "1.67144672777403  , iter no is  953"
## [1] "1.63521090368886  , iter no is  954"
## [1] "1.67144672777397  , iter no is  955"
## [1] "1.63521090368887  , iter no is  956"
## [1] "1.67144672777404  , iter no is  957"
## [1] "1.63521090368882  , iter no is  958"
## [1] "1.67144672777403  , iter no is  959"
## [1] "1.63521090368879  , iter no is  960"
## [1] "1.67144672777407  , iter no is  961"
## [1] "1.63521090368879  , iter no is  962"
## [1] "1.67144672777385  , iter no is  963"
## [1] "1.63521090368863  , iter no is  964"
## [1] "1.67144672777386  , iter no is  965"
## [1] "1.63521090368867  , iter no is  966"
## [1] "1.67144672777393  , iter no is  967"
## [1] "1.63521090368876  , iter no is  968"
## [1] "1.67144672777396  , iter no is  969"
## [1] "1.63521090368882  , iter no is  970"
## [1] "1.67144672777399  , iter no is  971"
```

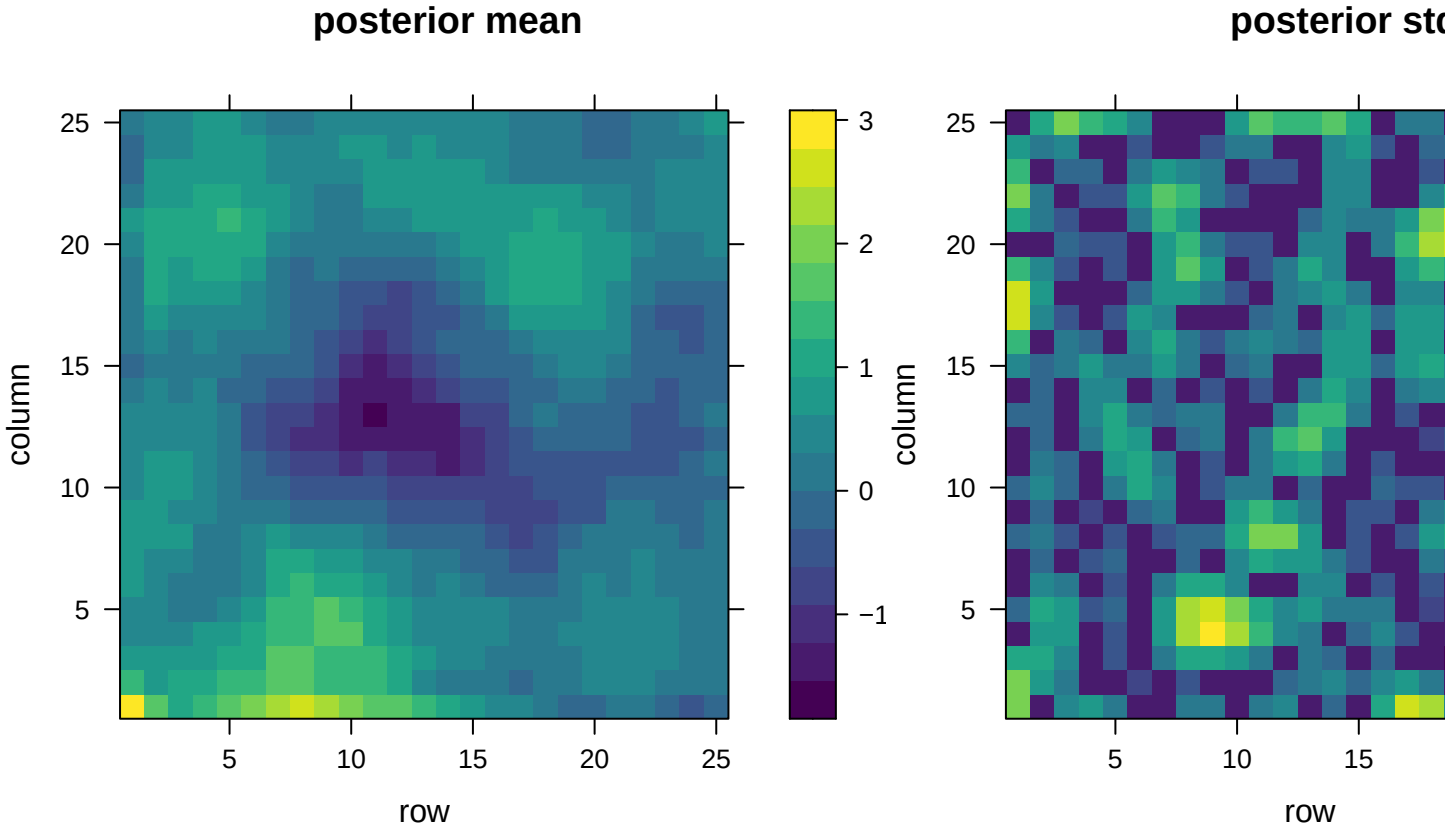
```
## [1] "1.63521090368878  , iter no is  972"
## [1] "1.67144672777401  , iter no is  973"
## [1] "1.63521090368888  , iter no is  974"
## [1] "1.671446727774  , iter no is  975"
## [1] "1.63521090368886  , iter no is  976"
## [1] "1.67144672777405  , iter no is  977"
## [1] "1.63521090368892  , iter no is  978"
## [1] "1.67144672777399  , iter no is  979"
## [1] "1.63521090368882  , iter no is  980"
## [1] "1.67144672777388  , iter no is  981"
## [1] "1.63521090368884  , iter no is  982"
## [1] "1.67144672777397  , iter no is  983"
## [1] "1.6352109036888  , iter no is  984"
## [1] "1.67144672777395  , iter no is  985"
## [1] "1.63521090368879  , iter no is  986"
## [1] "1.67144672777397  , iter no is  987"
## [1] "1.63521090368879  , iter no is  988"
## [1] "1.67144672777387  , iter no is  989"
## [1] "1.63521090368874  , iter no is  990"
## [1] "1.67144672777381  , iter no is  991"
## [1] "1.63521090368863  , iter no is  992"
## [1] "1.67144672777386  , iter no is  993"
## [1] "1.63521090368879  , iter no is  994"
## [1] "1.67144672777402  , iter no is  995"
## [1] "1.63521090368889  , iter no is  996"
## [1] "1.67144672777407  , iter no is  997"
## [1] "1.63521090368878  , iter no is  998"
## [1] "1.67144672777402  , iter no is  999"
```




```
## [1] "\nEstimated sigma is  0.689 ; True sigma is  1 \nEstimated phi is  9.53 ; True phi is  10 \nEsti
```

2.3 Kriging

We will now use the estimated model parameters to perform kriging prediction. Predict variables $x(s)$, where predictions sites lie on a regular grid of size 25x25 for the unit square. Visualize the Kriging surface and the prediction standard error. Compare with the true field.




```
## [1] "The prediction error is  0.929658012246742"
```

Part III Integrated nested Laplace Approximations (INLA)

In the last part of this exercise, we explore the `R-INLA` package along two examples.

3.1 Simple Linear Regression

First, we analyse the ski jumping data set using a linear regression model, which can be phrased as Latent Gaussian model suitable for INLA. Therefore, we start with loading the INLA package and exploring the dataset.

The ski jumping data set contains 26 observations of measured lengths in ski jumping competitions (in meters) between the years 1961 and 2011.

In Figure 1 we depict the 26 observations given their year. We observe a clear (almost linear) trend in the measured jumping lengths to increase with the years.

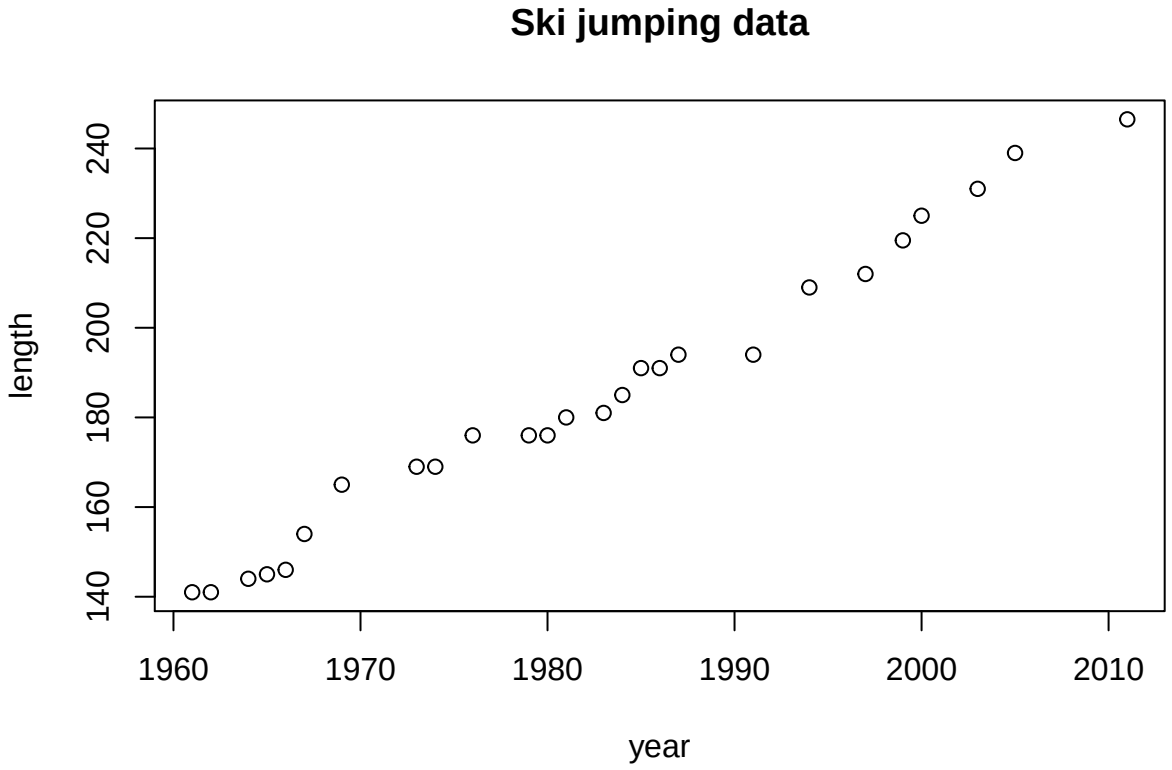


Figure 1: Visualisation of the ski jumping data set

Suitable for this model assumption on the data, we use linear regression approach for the statistical modelling of this data, where the years x_i are the covariates and the lengths y_i are the responses for $i = 1, \ldots, 26$:

$$\mathbb{E}[y_i] = \mu + \beta x_i, \quad \mathbb{V}\partial\backslash[y_i] = \tau^{-1}.$$

This can be posed as a latent Gaussian model suited for the INLA framework.

1. The response depends on the linear predictors η as $y|x, \theta = \Pi\pi(y_i|\eta_i, \tau)$ with Gaussian likelihood $\pi(y_i|\eta_i, \tau) \sim \mathcal{N}(\eta_i, \tau^{-2})$
2. The parameters μ and β of the linear predictor $eta_i = \mu + x_i\beta$ are independent Gaussian with a fixed huge variance and mean zero. Note that no additional hyperparameter is introduced here.
3. The model's hyperparameter τ is only one-dimensional and is equipped with a Gaussian prior by default with out specification.

The distributions which are not specified in detail here use default settings in the `R-INLA` package, which are naturally compatible with the LGM construction.

```
##
## Call:
##    "inla(formula = Length ~ Year, data = skiData, control.predictor =
##    list(compute = TRUE))"
## Time used:
##     Pre = 3.65, Running = 4.29, Post = 0.236, Total = 8.18
## Fixed effects:
##                   mean      sd 0.025quant  0.5quant 0.975quant      mode kld
## (Intercept) -4029.646 106.615  -4240.649 -4029.650  -3818.934 -4029.646   0
## Year            2.126   0.054      2.019     2.126      2.232     2.126   0
##
```

```
## Model hyperparameters:
##                                           mean   sd 0.025quant 0.5quant
## Precision for the Gaussian observations 0.072 0.02      0.038     0.07
##                                         0.975quant  mode
## Precision for the Gaussian observations      0.116 0.066
##
## Expected number of effective parameters(stdev): 2.00(0.00)
## Number of equivalent replicates : 13.00
##
## Marginal log-Likelihood:  -89.58
## Posterior marginals for the linear predictor and
##  the fitted values are computed
```

The INLA run generates posterior estimates for the fixed effects μ and β, which will be investigated blow. In the summary, we read the precision of the distribution for the hyperparameter τ which is rather small, meaning that the variance will be rather big. However, in a ski jumping competition we can expect a variance of several meters such that this is reasonable.

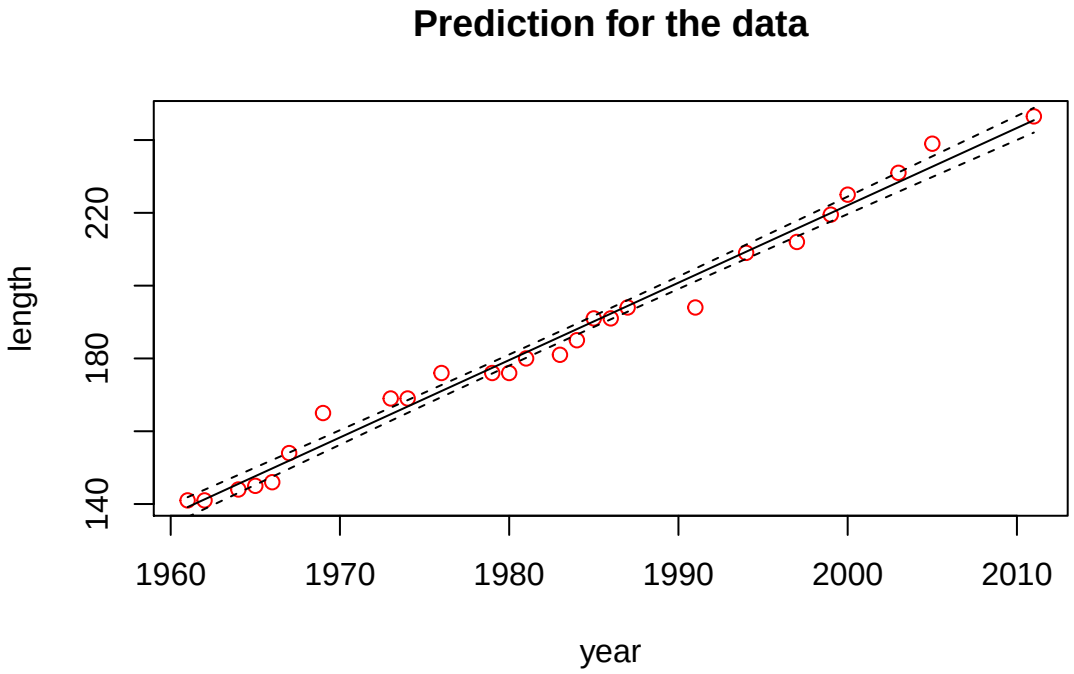


Figure 2: Linear regression with INLA

In Figure 2 we see that the linear trend is really well captured by the model (black line), whereas the 95% credibility interval (dashed line) does not cover all data points.

```
## [1] "Estimates for sigma:"
```

```
## Mean            3.83993
## Stdev           0.552976
## Quantile  0.025 2.93482
## Quantile  0.25  3.44617
## Quantile  0.5   3.77761
## Quantile  0.75  4.16357
## Quantile  0.975 5.09958
```

In Figure 3, we depict the marginal posterior distributions for all variables of interest. In particular, the

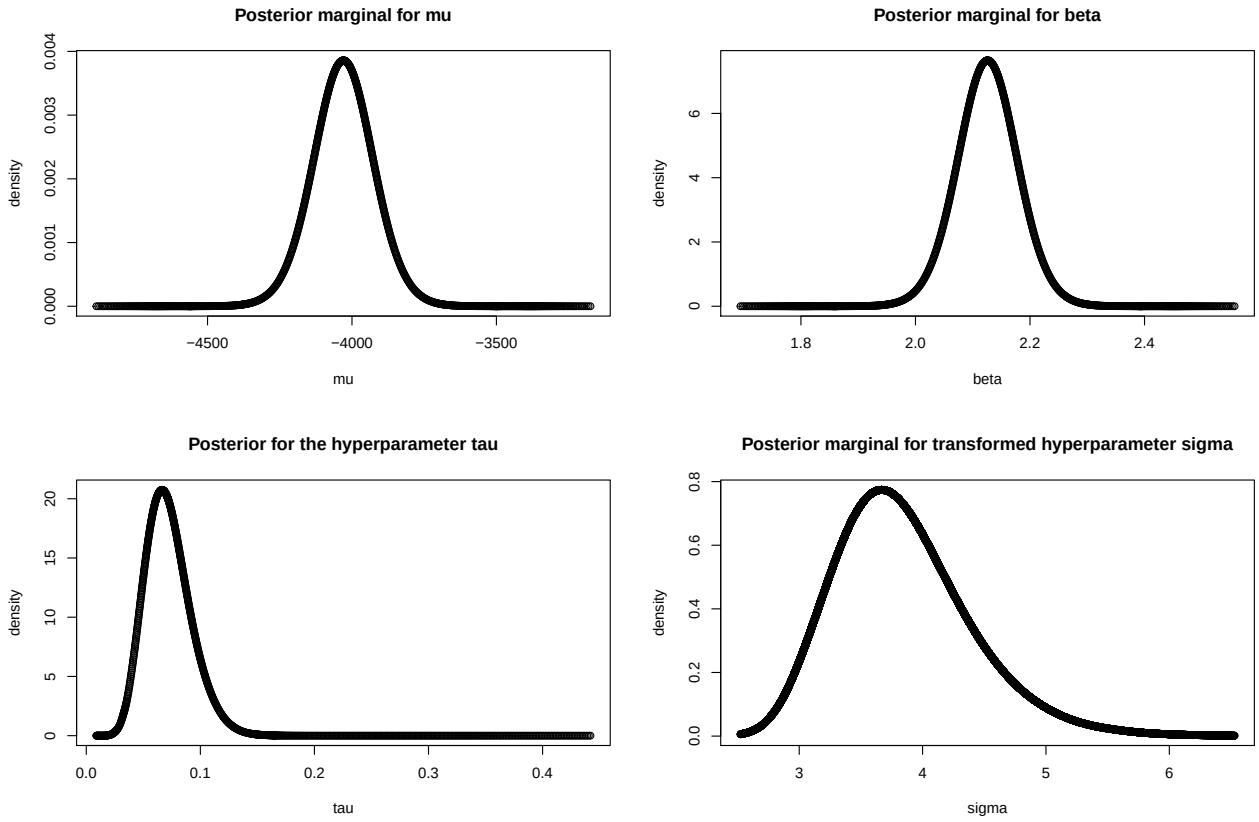


Figure 3: Posterior marginals for selected effects

transformed marginal analysis confirms our interpretation that the variance in the magnitude of several meters (roughly around 4 meters), but again this is very reasonable for this application.

3.2 GLMM with random effects

Last, we use INLA to analyse the "Seeds" data set. This data concerns the portion of seeds that germinated on a sample set of 21 plates. The plates are equipped with one of two types of seeds and one of two types of extracts. The characteristics of plate i are described by the covariates $x_{1,i}$ which is the type of seed and $x_{2,i}$ which is the type of root extract - these covariates are either 0 or 1 for the different possibilies. Then the number of germinated seeds r_i on plate i is counted in contrast to the number of total seeds n_i on that plate. Having p_i as the probability of germination on plate i, a binomial model for this example is

$$r_i \sim Binomial(p_i, n_i)$$
$$\text{logit}(p_i) = a_0 + a_1 x_{1,i} + a_2 x_{2,i} + \varepsilon_i$$

where ε_i is some iid noise. As above non-specified prior- and hyperparameter-distributions use the default of `R-INLA`. This model is then again implemented in `R-INLA`.

```
## [1] "The x1 covariate:"

##  [1] 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1

## [1] "The x2 covariate:"

##  [1] 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1

##
## Call:
##    c("inla(formula = formula, family = \"binomial\", data = data, Ntrials
##    = n, ", " control.predictor = list(compute = TRUE), control.family =
##    list(link = \"logit\"))" )
## Time used:
##     Pre = 4.56, Running = 0.484, Post = 0.217, Total = 5.26
## Fixed effects:
##               mean    sd 0.025quant 0.5quant 0.975quant   mode kld
## (Intercept) -0.429 0.115     -0.656   -0.429     -0.204 -0.428   0
## x1          -0.272 0.156     -0.580   -0.272      0.033 -0.271   0
## x2           1.066 0.146      0.782    1.066      1.353  1.065   0
##
## Random effects:
##   Name     Model
##     plate IID model
##
## Model hyperparameters:
##                           mean       sd 0.025quant 0.5quant 0.975quant mode
## Precision for plate 18930.69 20071.40      30.63 12701.76   73765.97 7.83
##
## Expected number of effective parameters(stdev): 3.25(1.16)
## Number of equivalent replicates : 6.46
##
## Marginal log-Likelihood:  -73.75
## Posterior marginals for the linear predictor and
##  the fitted values are computed
```

Remark the structured pattern in the covariates. Furthermore, here the precision is rather higher yielding a small variance in the estimate.

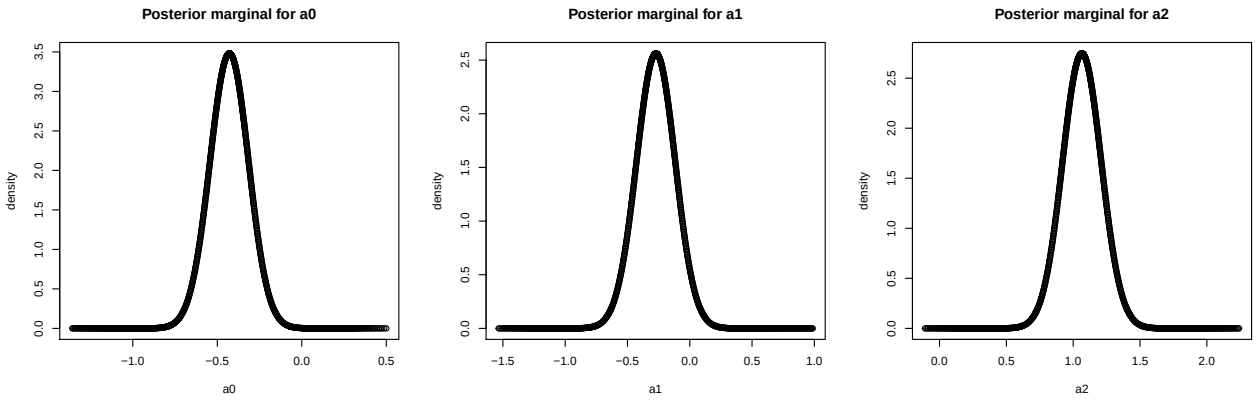


Figure 4: Posterior marginals for selected effects

In Figure 4, we see rather confident estimates for the fixed effects of the model. In particular, we note that the effect a_2 will dominate the model since we only have 0 and 1 values for the covariates. The effect a_0 is common for all plates and the absolute value of a_1 for the seed type is a way smaller than the one for the root extract.

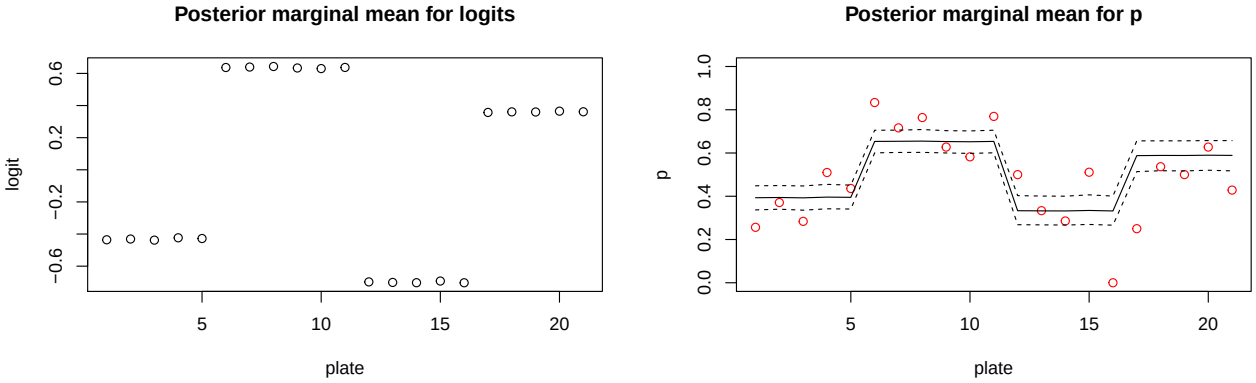


Figure 5: GLMM with INLA

In Figure 5, we recognize a clear pattern with high values when the root extract with value 1 is chosen. The choice of the seed only has minor influence on the prediction. Moreover, the variance of this model cannot cover all data points, which is a common issue in binomial models with few data though.