Advanced Spatial Modeling with Stochastic Partial Differential Equations Using R and INLA

Chapter 7 Space-time models

分離可能な時空間モデルを適合させる方法について説明

分離可能(離散化できる)な時空間モデル

・空間領域のSPDEモデル + 次数1の自己回帰モデル(AR1）。

・空間的および時間的ランダム効果の精度行列間のクロネッカー積によって定義される。

時空モデルを実装する2つの方法(7.1と7.2)

・1:離散時間、2:連続時間の例を紹介。

・連続時間のデータは、時間ノットを定義する必要がある。

・どちらの方法でも、時間の経過とともに測定位置が同じである必要はない。

計算を高速化するためのモデルの構造化方法(7.3と7.4)

・7章は基本的なコード例に焦点を当て、8章ではいくつかの高度な例を示す。

7.1 離散時間のモデル

このセクションでは、Cameletti（2013）らのように、時空間分離可能モデルを適合させる方法を示す。さらに、カテゴリカル共変量を含める方法を示す。

7.1.1 シミュレーションデータの作成

7.1.2 データスタックの準備

7.1.3 モデルへのフィッティングと結果

7.1.4 ランダムフィールドの事後分布

7.1.5 モデルの検証

7.1.1 シミュレーションデータの作成

①空間モデルを定義する

セクション2.6で作成したパラナ州の境界に低解像度メッシュを使用。

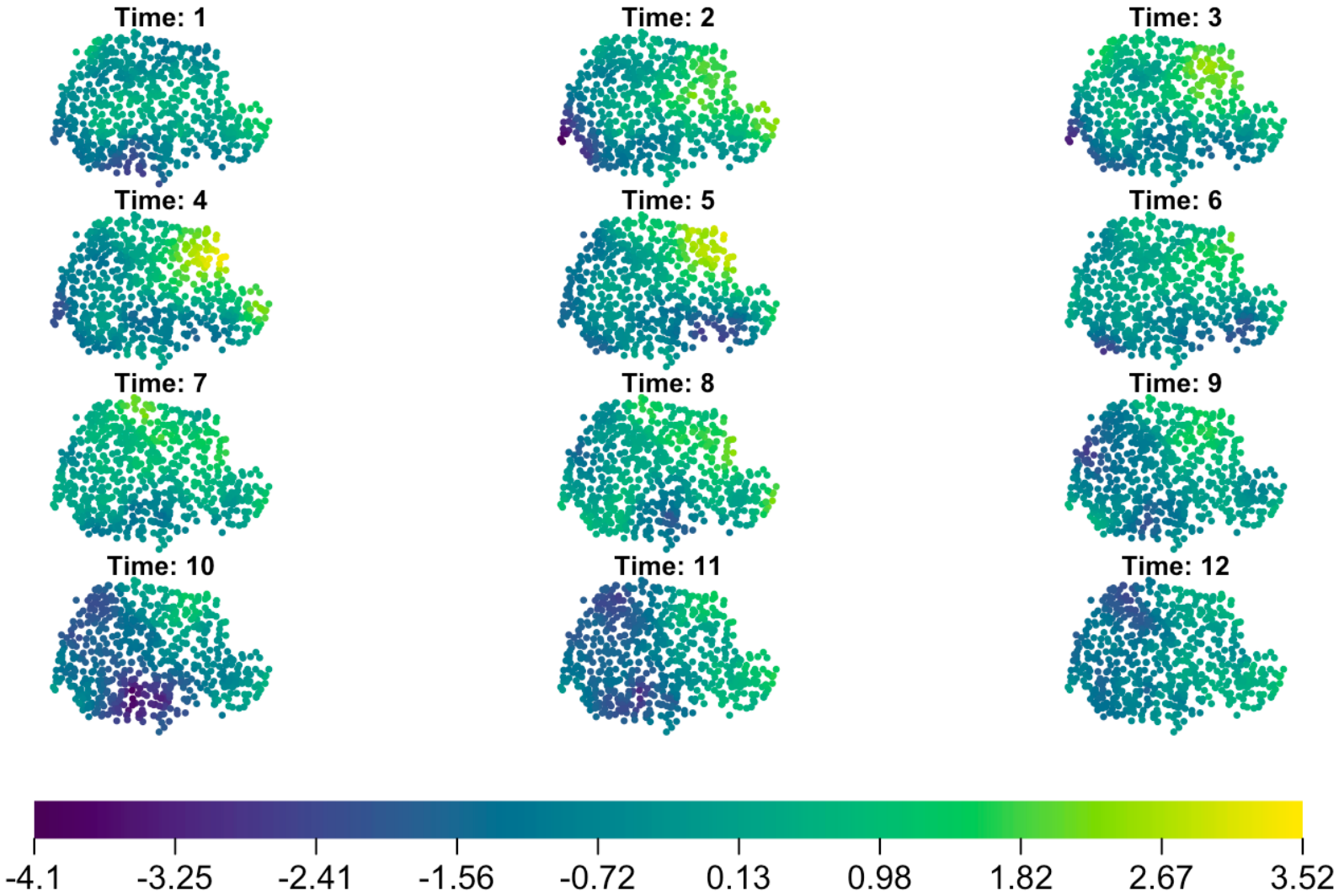
②時間的な相関を考慮した環境場を作成する

独立した環境場に相関を与える。

```
x <- x.k
for (j in 2:k)    {
      x[, j] <- rho * x[, j - 1] + sqrt(1 - rho^2) * x.k[, j]
}
```

x.k: 12の独立した環境場(616 X 12の行列)
rho=0.7 (自己相関係数)
sqrt{1-¥rho^2}: 時間軸における定常過程

7.1.1 シミュレーションデータの作成



7.1.1 シミュレーションデータの作成

③シミュレーションデータの作成

3つのレベル（A、B、Cのラベルが付いている）を持つカテゴリカル共変量を出力。

```
n <- nrow(coords)　set.seed(2)
ccov <- factor(sample(LETTERS[1:3], n * k, replace = TRUE))
```

回帰係数と回帰パラメーターは次のとおり。

```
beta <- -1:1
```

応答変数は、カテゴリカル共変量（固定効果）、時空間ランダム効果、誤差から算出される。

```
sd.y <- 0.1
y <- beta[unclass(ccov)] + x + rnorm(n * k, 0, sd.y)
```

さまざまな時間・場所のデータを使用できることを示すために、一部の観測を削除する。

```
isel <- sample(1:(n * k), n * k / 2)
```

7.1.1 シミュレーションデータの作成

④データフレームの作成

これまでのデータを、データフレームとしてまとめる。

```
dat <- data.frame(y = as.vector(y), w = ccov,
time = rep(1:k, each = n),
xcoo = rep(coords[, 1], k),
ycoo = rep(coords[, 2], k))[isel, ]
```

実際のアプリケーションでは、**異なる時間で観測位置が完全にずれている場合がある。**
この例で提供するコードは、そのような状況で機能する。

7.1.2 データスタックの準備

①パラメータの事前分布を設定

Fuglstad et al（2018）で導出されたPC-priorsを事前分布として使用。
（パラメータの範囲と限界標準偏差を設定。）

```
spde <- inla.spde2.pcmatern(mesh = prmesh1,
prior.range = c(0.5, 0.01), # P(range < 0.05) = 0.01 SPDE
prior.sigma = c(1, 0.01)) # P(sigma > 1) = 0.01 SPDE
h.spec <- list(theta = list(prior = 'pccor1', param = c(0, 0.9))) #P(cor>0)=0.9 Rho
prec.prior <- list(prior = 'pc.prec', param = c(1, 0.01)) # P(sigma > 1) = 0.01 SD
```

②インデックスセットの作成

SPDEモデルのメッシュポイントの数とグループの数(k=12)を考慮して作成される。

```
iset <- inla.spde.make.index('i', n.spde = spde$n.spde,  n.group = k)
```

7.1.2 データスタックの準備

③プロジェクターマトリックス（A）の作成

時間インデックスをグループ引数に渡す必要がある。

```
A <- inla.spde.make.A(mesh = prmesh1,
                      loc = cbind(dat$xcoo, dat$ycoo), group = dat$time)
```

④データスタックの作成

```
sdat <- inla.stack(data = list(y = dat$y),
                   A = list(A, 1),
                   effects = list(iset, w = dat$w),
                   tag = 'stdata')
```

7.1.3 モデルへのフィッティングと結果

カテゴリカル共変量を処理するには、control.fixed引数リストでexpand.factor.strategy = 'inla'を使用して、直感的な結果を得る必要がある。

Model formula

```
formulae <- y ~ 0 + w + f(i, model = spde, group = i.group,
                          control.group = list(model = 'ar1', hyper = h.spec))
```

Model fitting

```
res <- inla(formulae,  data = inla.stack.data(sdat),
            control.predictor = list(compute = TRUE,
              A = inla.stack.A(sdat)),
            control.family = list(hyper = list(theta = prec.prior)),
            control.fixed = list(expand.factor.strategy = 'inla'))
```

7.1.3 モデルへのフィッティングと結果

3つの切片の要約と、各共変量の平均値は次のとおり。

データ

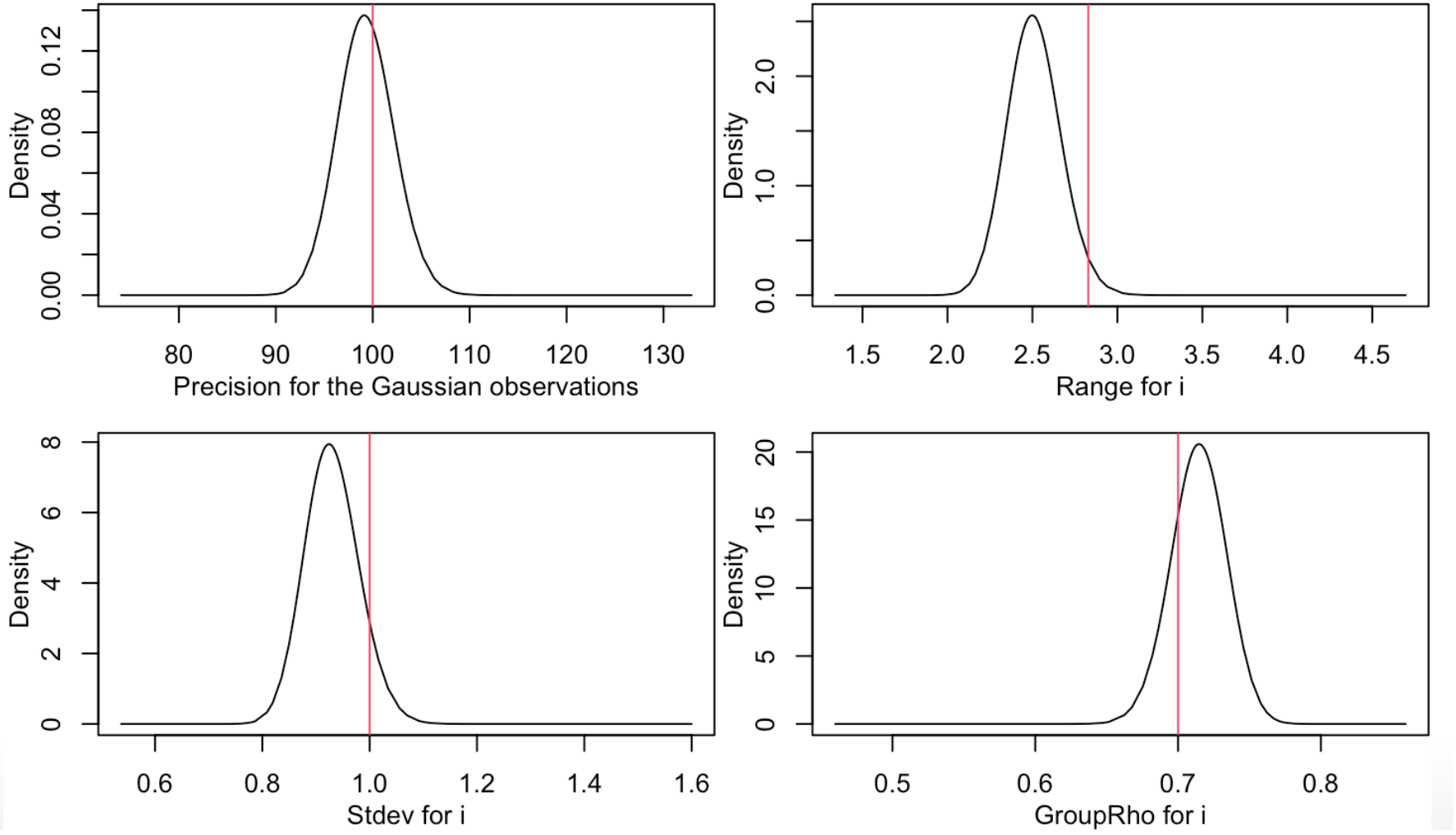
```
##           A           B           C
## -1.1254313 -0.1993376  0.8535994
```

推定値

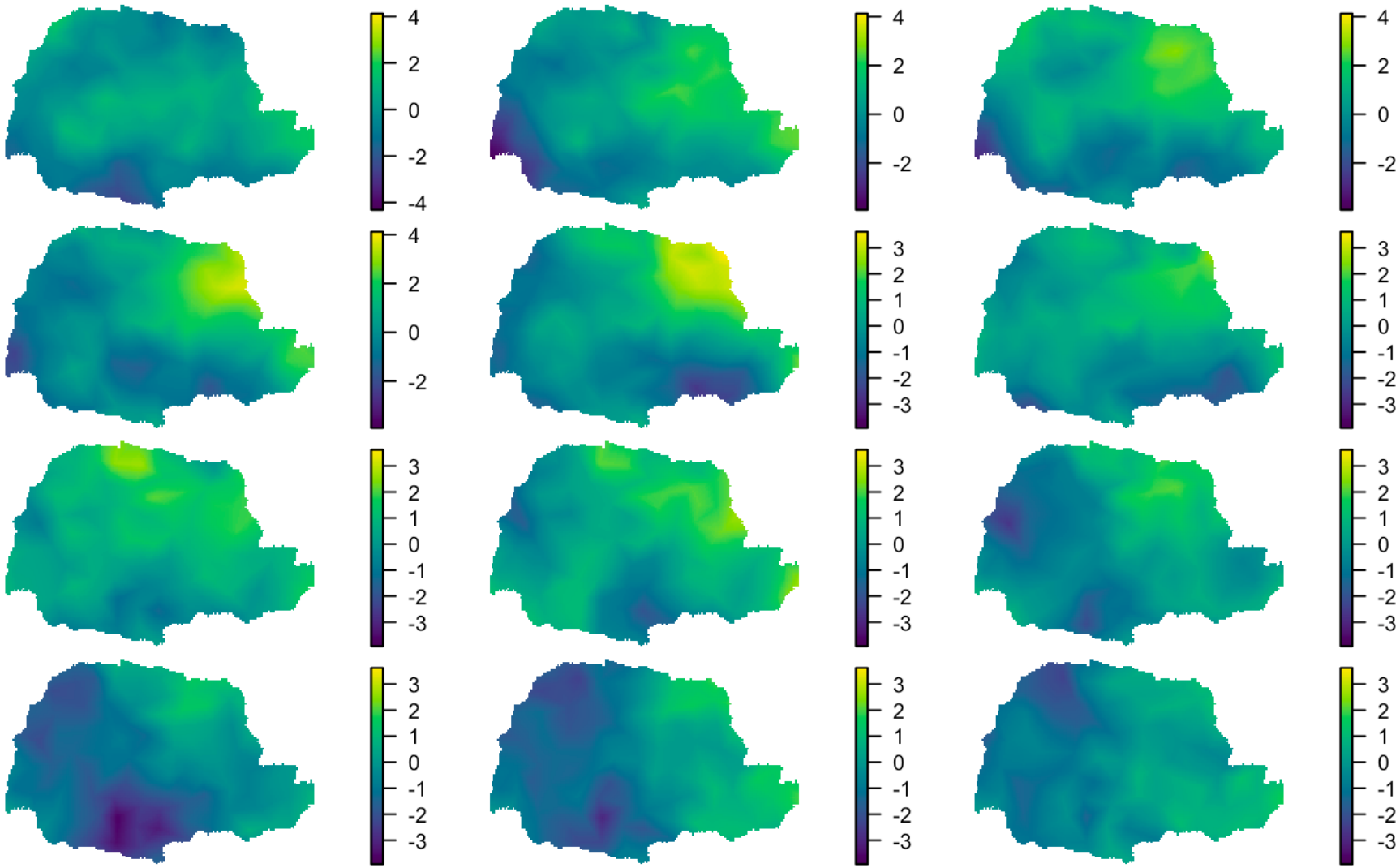
```
##          Obs.       mean         sd 0.025quant   0.5quant 0.975quant
## A -1.1353911 -1.1726136 0.3182682 -1.8021666 -1.1723100 -0.5456799 -
## B -0.2051161 -0.1735047 0.3182694 -0.8030542 -0.1732031  0.4534366 -
## C  0.8437480  0.8252023 0.3182716  0.1956347  0.8255088  1.4521350
```

7.1.3 モデルへのフィッティングと結果

パラメーターの事後周辺分布と時間相関係数の周辺分布を図示する。。



7.1.4 ランダムフィールドの事後分布



7.1.5 モデルの検証

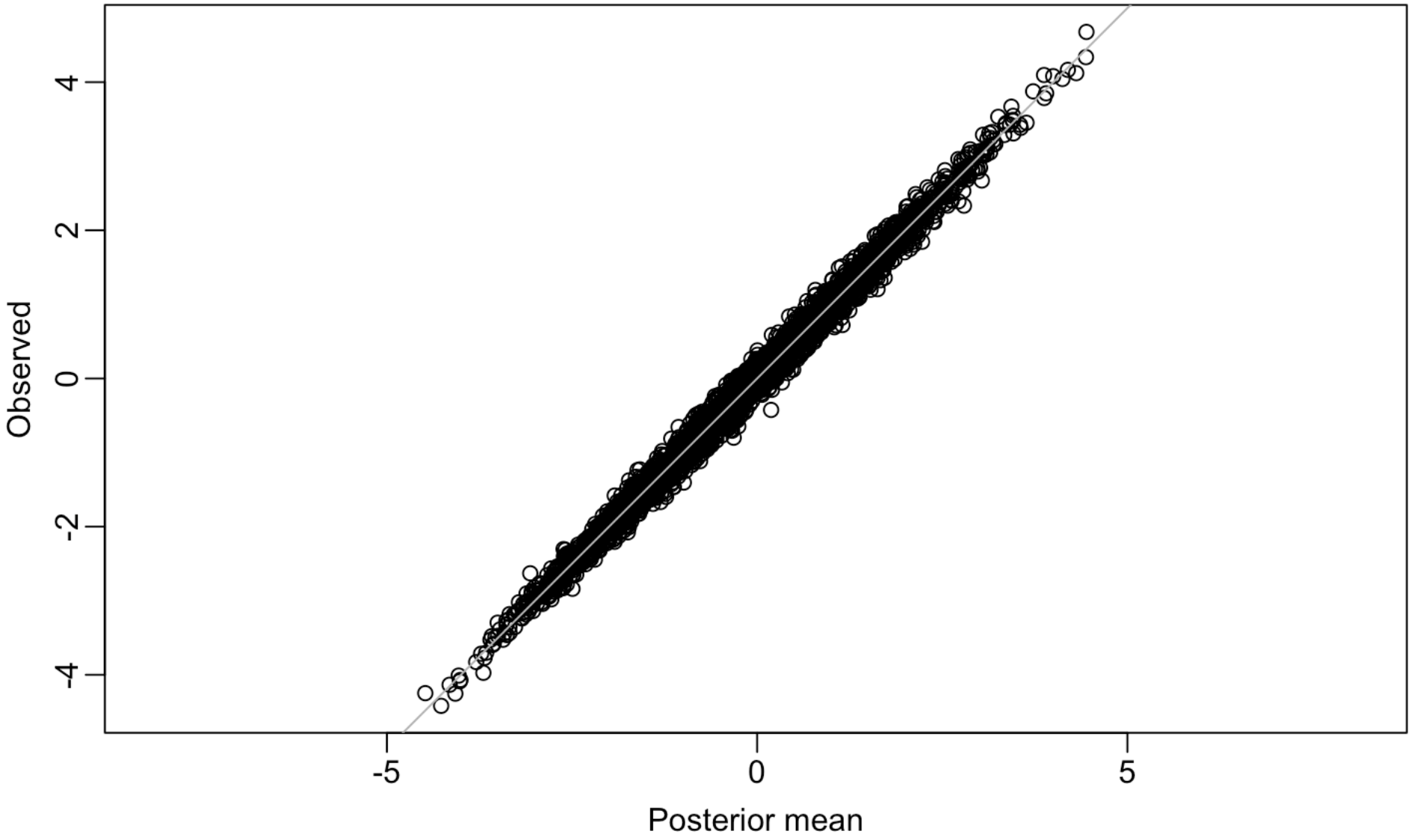
シミュレーションデータの他の部分は、検証（クロスバリデーション）に使用できる。

検証データの事後分布を計算するには、別のデータスタックが必要。

```
vdat <- data.frame(r = as.vector(y), w = ccov,
  t = rep(1:k, each = n), x = rep(coords[, 1], k),
  y = rep(coords[, 2], k))
vdat <- vdat[-isel, ]

Aval <- inla.spde.make.A(prmesh1,
  loc = cbind(vdat$x, vdat$y), group = vdat$t)
stval <- inla.stack(
  data = list(y = NA), # NA: no data, only enable
predictions
  A = list(Aval, 1),
  effects = list(iset, w = vdat$w),
  tag = 'stval')
```

```
stfull <- inla.stack(sdat, stval)
vres <- inla(formulae,  data = inla.stack.data(stfull),
  control.predictor = list(compute = TRUE,
    A = inla.stack.A(stfull)),
  control.family = list(hyper = list(theta = prec.prior)),
  control.fixed = list(expand.factor.strategy = 'inla'),
  control.mode = list(theta = res$mode$theta,
restart = FALSE))
```

7.1.5 モデルの検証



7.2 連続時間のドメイン

観測値が離散的にサンプリングされたという仮定を排除する。
(漁業や時空ポイントプロセスなどのデータ)
→時間軸に対しても空間と同様のアプローチでデータを作成。

AR1で時間軸の相関を近似する。

7.2.1 データシミュレーション

7.2.2 データスタックの準備

7.2.3 モデルへのフィッティングと結果

7.2.1 データシミュレーション

まず、連続的なデータから空間位置とサンプル時間点を設定する。

```
loc <- unique(as.matrix(PRprec[, 1:2]))
n <- nrow(loc)
time <- sort(runif(n, 0, 1))
```

次に、時空分離可能な共分散関数を定義する。

空間：Matérn covariance
時間：指数関数的に減衰する共分散関数

```
local.stcov <- function(coords, time, kappa.s, kappa.t,variance = 1, nu = 1) {
                        s <- as.matrix(dist(coords))
                        t <- as.matrix(dist(time))
                        scorr <- exp((1 - nu) * log(2) + nu * log(s * kappa.s) -
                        lgamma(nu)) * besselK(s * kappa.s, nu)
                        diag(scorr) <- 1
                        return(variance * scorr * exp(-t * kappa.t))
}
```

7.2.1 データシミュレーション

関数local.stcov()を使用して、シミュレーションされた時空間点での
共分散関数を計算し、サンプリングする。

```
Kappa.s <- 1
Kappa.t <- 5
S2 <- 1 / 2
xx <- crossprod(chol(local.stcov(loc, time, kappa.s, kappa.t, s2)),   rnorm(n))

beta0 <- -3
tau.error <- 3

y <- beta0 + xx + rnorm(n, 0, sqrt(1 / tau.error))
```

7.2.2 データスタックの準備

連続的な時空間モデルに適合させるために、
時間ノットと時間メッシュを定義する必要がある。
ここでは、10ノットの1次元メッシュを定義する。

```
k <- 10
mesh.t <- inla.mesh.1d(seq(0 + 0.5 / k, 1 - 0.5 / k, length = k))
```

時間メッシュのノットは以下のようになる。

```
mesh.t$loc
```

```
##  [1] 0.05 0.15 0.25 0.35 0.45 0.55 0.65 0.75 0.85 0.95
```

7.2.2 データスタックの準備

セクション2.6で作成したパラナ州の境界線には、引き続き低解像度メッシュを使用。
時空間モデルのインデックスセットは、次のように定義できる。

```
iset <- inla.spde.make.index('i', n.spde = spde$n.spde, n.group = k)
```

射影行列は、空間的および時間的射影の両方を考慮している。
したがって、空間メッシュと空間位置、時間点と時間メッシュが必要。
これらは、次のように関数inla.spde.make.Aに渡される。

```
A <- inla.spde.make.A(mesh = prmesh1, loc = loc, group = time,
                      group.mesh = mesh.t)
```

7.2.2 データスタックの準備

データスタックの説明変数は、2つの要素を持つリスト。
(①時空間効果と②カテゴリカル共変量のインデックスセット。)
スタックデータは次のように定義される。

```
sdat <- inla.stack(data = list(y = y),
                   A = list(A,1),
                   effects = list(iset, list(b0 = rep(1, n))),
                   tag = "stdata")
```

7.2.3 モデルへのフィッティングと結果

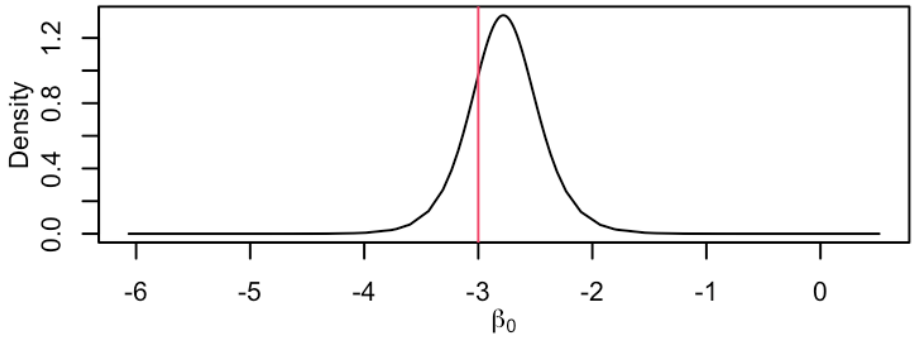
時間ノット間の相関は、kappaを逆範囲パラメーターとした指数相関関数を使用。

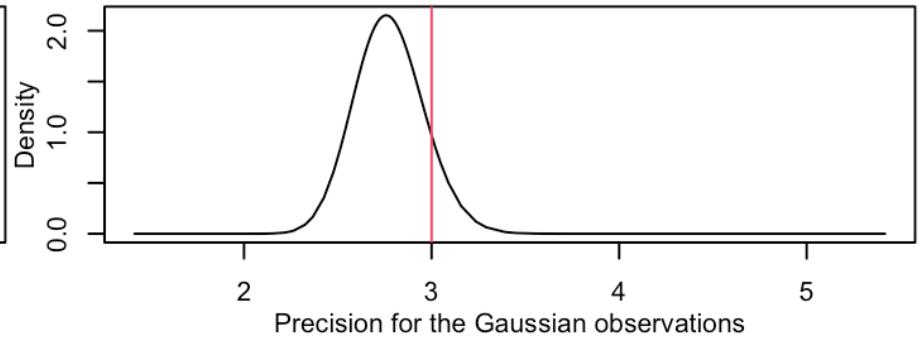
exp(-kappa.t * diff(mesh.t$loc[1:2]))=0.606

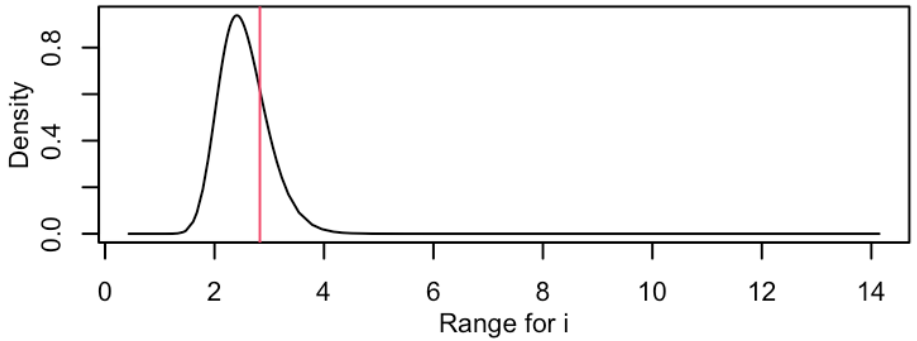
最後に、時間ノットに対するAR（1）モデルを使用して近似する。

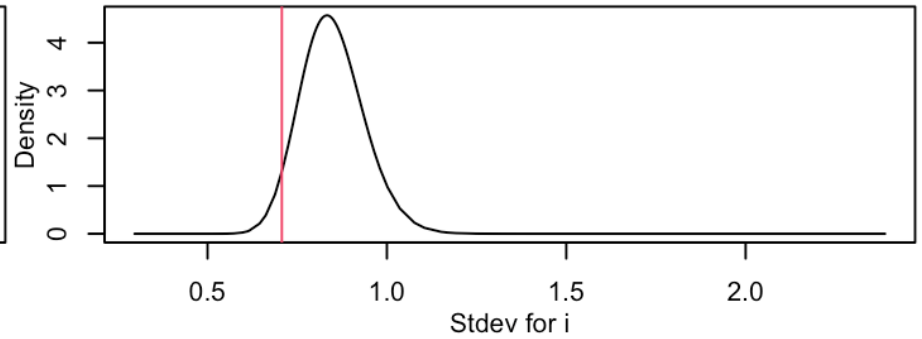
```
formulae <- y ~ 0 + b0 + f(i, model = spde, group = i.group,
                                    control.group = list(model = 'ar1', hyper = h.spec))

res <- inla(formulae, data = inla.stack.data(sdat),
            control.family = list(hyper = list(theta = prec.prior)),
            control.predictor = list(compute = TRUE,
            A = inla.stack.A(sdat)))
```

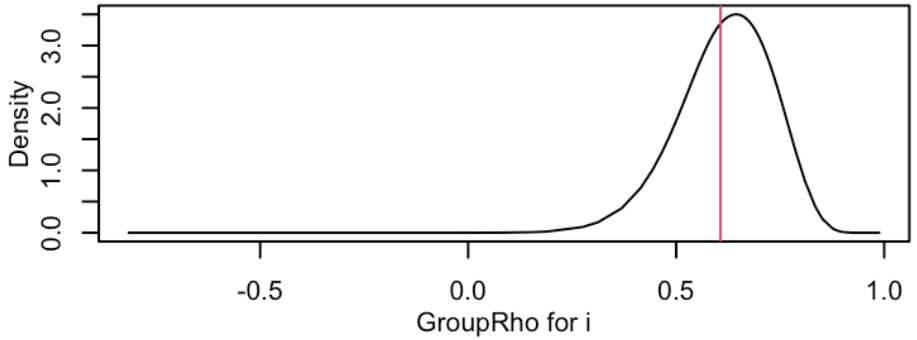
7.2.3 モデルへのフィッティングと結果











7.3 時空間モデルの解像度を下げる

大きなデータセットを扱う場合、モデルのフィッティングは困難な場合がある。
このセクションでは、モデルのフィッティングを高速化するために、
時空間ランダム効果の表現の解像度を下げる手法を示す。

まず、パラナ州の降雨データを使用して、空間メッシュとSPDEモデルを構築する。

```
data(PRprec)
bound <- inla.nonconvex.hull(as.matrix(PRprec[, 1:2]), 0.2, 0.2, resol = 50)
mesh.s <- inla.mesh.2d(bound = bound, max.edge = c(1,2),
offset = c(1e-5, 0.7), cutoff = 0.5)
spde.s <- inla.spde2.matern(mesh.s)
```

7.3.1 データの一時的な集計

分析するデータ：365日間、616の観測地点の降水量。

目的：雨の降る確率を分析すること。

集計①：降雨量の連続データセットを雨の発生に変換する。
（降雨量が0.1より多かったかどうか。）

集計②：データセットのサイズを縮小するために、5日間ごとに再集計する。

- 集約されたデータセットは二項式によってモデル化される(5回試行の二項分布)。
- 欠損値のため、5日未満の観測値を持つブロックが多数あり、
  これらは5回未満の試行で2項式を与える。

```
##    Longitude Latitude Altitude d0101 d0102 d0103 d0104
## 3   -50.7711 -22.9597      344     0     1     0   0.0
## 4   -50.6497 -22.9500      904     0     0     0   3.3
```

7.3.1 データの一時的な集計

再集計されたデータフレームは以下の通り。

降水回数

観測日数（最大５日）

```
k <- ncol(y5)
n <- nrow(PRprec)
df = data.frame(y = as.vector(y5), ntrials = as.vector(n5),
                locx = rep(PRprec[, 1], k),
                locy = rep(PRprec[, 2], k),
                time = rep(1:k, each = n),
                station.id = rep(1:n, k))
```

集計データには73のタイムポイントがある(73x5=365)。
しかし、3563のn5ビンでデータが記録されていない。

欠損値は、削除せずNAをyに振り分ける。nには5日を割り当てる。

```
y5[n5 == 0] <- NA
n5[n5 == 0] <- 5
```

7.3.2 時間の解像度を下げる

時間軸上にいくつかのノットを配置してモデルを定義する
Lindgren and Rue(2015), Blangiardo and Cameletti(2015) 。

・メッシュを使用した空間の場合と同様に、時間ノットから投影が定義される。

・ノットは、全体で73のタイムポイントを持つ時間的に集計されたデータの
6つのタイムポイントごとに配置される。

・最終的に、12ノットだけになる。

```
k <- 73
bt <- 6
gtime <- seq(1 + bt, k, length = round(k / bt)) - bt / 2
mesh.t <- inla.mesh.1d(gtime, degree = 1)
```

最終的に、モデルの次元は1152になる。

7.3.2 時間の解像度を下げる

射影行列が計算されるとき、分析されるデータのスケールで時間メッシュとグループインデックスを考慮する必要がある。

```
Ast <- inla.spde.make.A(mesh = mesh.s,
                        loc = cbind(df$locx, df$locy), group.mesh = mesh.t,
                        group = df$time)
```

インデックスセットとデータスタックは通常どおり作成できる。

```
idx.st <- inla.spde.make.index('i', n.spde = spde.s$n.spde, n.group = mesh.t$n)
stk <- inla.stack(data = list(y = df$y, ntrials = df$ntrials),
                  A = list(Ast, 1),
                  effects = list(idx.st, data.frame(mu0 = 1,
                  altitude = rep(PRprec$Altitude / 1000, k))))
```

7.3.2 時間の解像度を下げる

回帰式は、分離可能な時空間モデルと同様に設定できる。

```
form <- y ~ 0 + mu0 + altitude + f(i, model = spde.s,
                                   group = i.group, control.group = list(model = 'ar1',
                                   hyper=list(theta=list(prior='pc.cor1', param=c(0.7, 0.7)))))
```

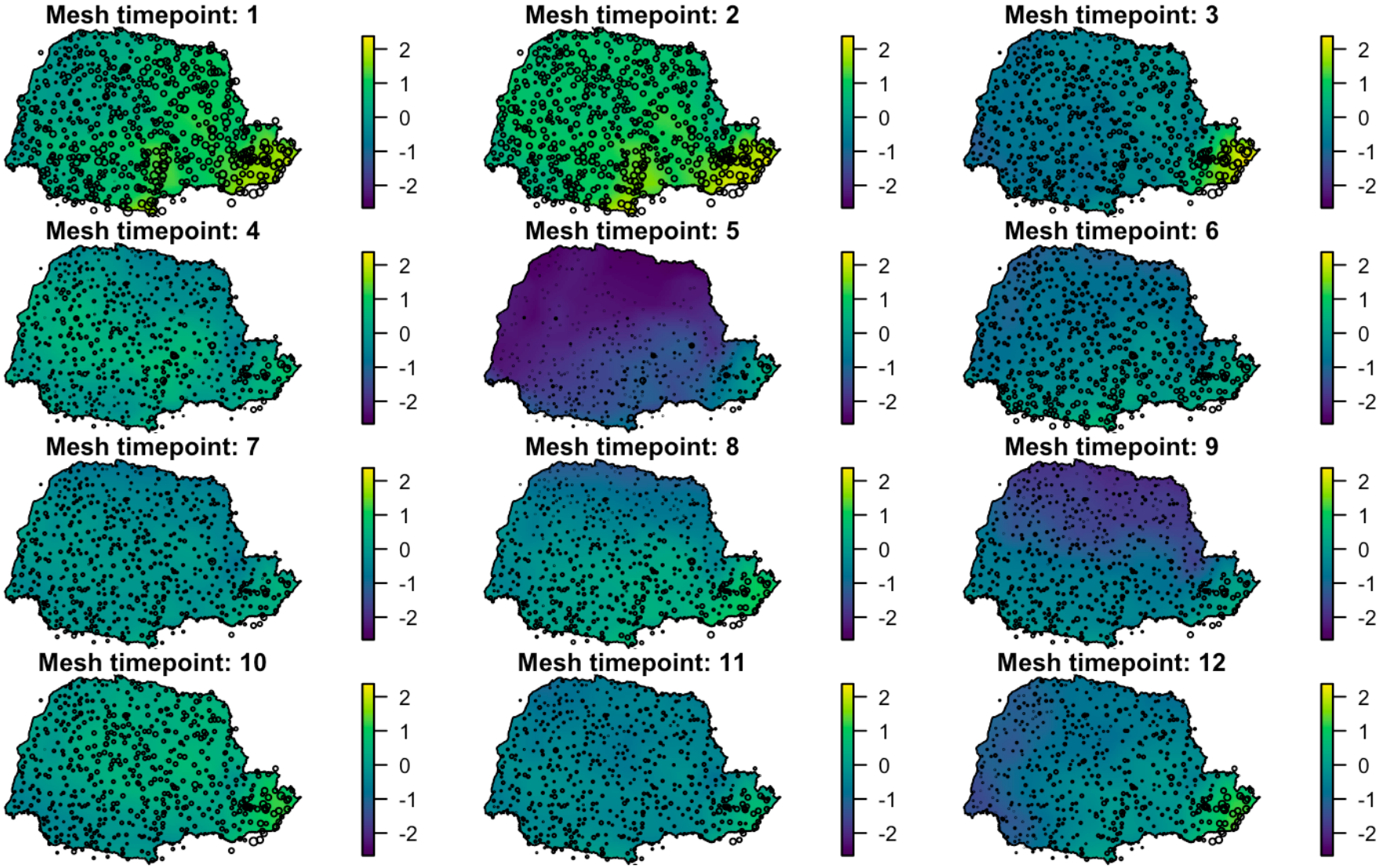
計算時間を短縮するために、いくつかのオプションを設定する。

ハイパーパラメータ（int.strategy = 'eb'）に対する適応近似（strategy = 'adaptive'）およびEmpirical Bayesを使用。

初期値initからオプティマイザーを開始する。

```
init = c(-0.5, -0.9, 2.6) # Initial values of hyperparameters
result <- inla(form, 'binomial', data = inla.stack.data(stk),
               Ntrials = inla.stack.data(stk)$ntrials,
               control.predictor = list(A = inla.stack.A(stk), link = 1),
               control.mode = list(theta = init, restart=TRUE),
               control.inla = list(strategy = 'adaptive', int.strategy = 'eb'))
```

7.3.2 時間の解像度を下げる



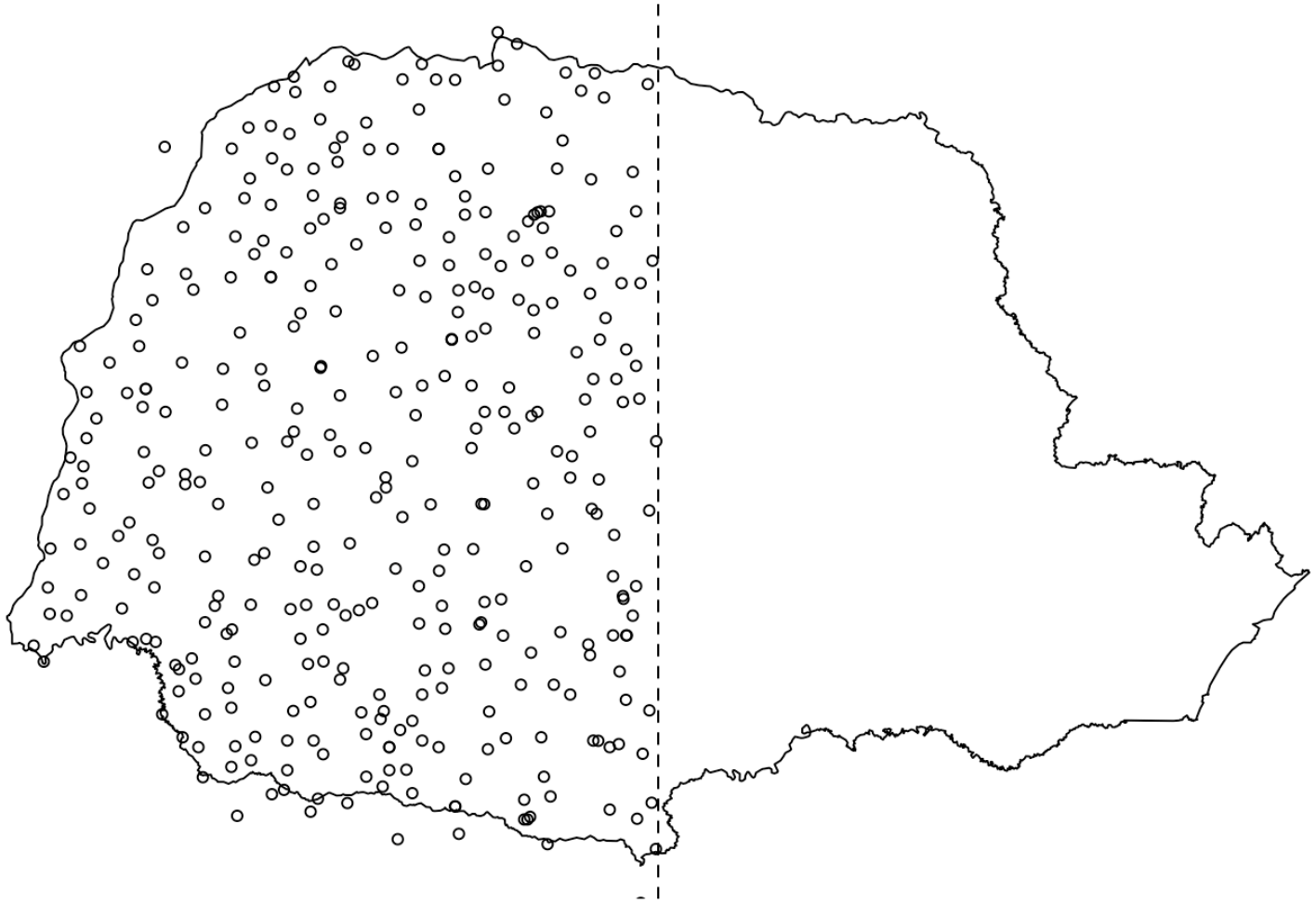
7.4 条件付きシミュレーション：2つのメッシュの結合

広域にわたる空間的または時空間的な現象のモデリングと予測をする場合、
多くの状況で、観測は限られた部分でのみ利用可能。

- ✓ このセクションでは、計算効率の良い方法でこれに対処する方法について説明する。
- ✓ 説明を簡略化するために空間データに焦点を当てているが、同じ原理を時空間の場合に簡単に拡張できる。
- ✓ パラナ状態の境界領域を使用し、パラナの左半分の部分からのデータのみがあると仮定。しかし、状態全体に対して空間予測が必要。

7.4.1 モチベーション

利用可能なデータの周りのメッシュを使用してモデルをフィッティングし、関心領域全体のメッシュを使用して予測することを検討する。



7.4.1 モチベーション

- ✓ データが観測される場所(mesh1)と予測場所(mesh2)の二つのメッシュを使用してモデルを近似する。
- ✓ 具体的には、観測値がある周りのみのmesh1を使用して推定し、条件付きシミュレーションを使用して、mesh2のノードで予測する。
- ✓ mesh2を直接使用してモデルをフィッティングする場合と比較すると、計算が大幅に高速化される。
- ✓ 調整後、データの場所での予測は、mesh1を使用した近似から得られた値とまったく同じになる。

7.4.1 モチベーション

- ✓ データが観測される場所(mesh1)と予測場所(mesh2)の二つのメッシュを使用してモデルを近似する。
- ✓ 具体的には、観測値がある周りのみのmesh1を使用して推定し、条件付きシミュレーションを使用して、mesh2のノードで予測する。
- ✓ mesh2を直接使用してモデルをフィッティングする場合と比較すると、計算が大幅に高速化される。
- ✓ 調整後、データの場所での予測は、mesh1を使用した近似から得られた値とまったく同じになる。

7.4.2 パラナ州の例

統計モデル

それぞれの座標 i に対し、以下の観測値 y_i が以下の分布によって得られる。

$$y_i \sim N(\eta_i, \sigma_\epsilon)$$

ここで、σ_ϵはノイズで、η_iは以下のような線形予測子である。

$$\eta_i = \beta_0 + u_i$$

ここで、β_0は切片であり、u_iはGFである。

7.4.2 パラナ州の例: シミュレーションデータ

①パラナ州の西半分からのデータのみを考慮してmesh1を構築する。

mesh1 <- inla.mesh.2d(loc = PRprec[sel.loc, 1:2], max.edge = 1,　cutoff = 0.1, offset = 1.2)

②パラナ州の境界線を使用して、メッシュ2の高解像度内部を定義する。

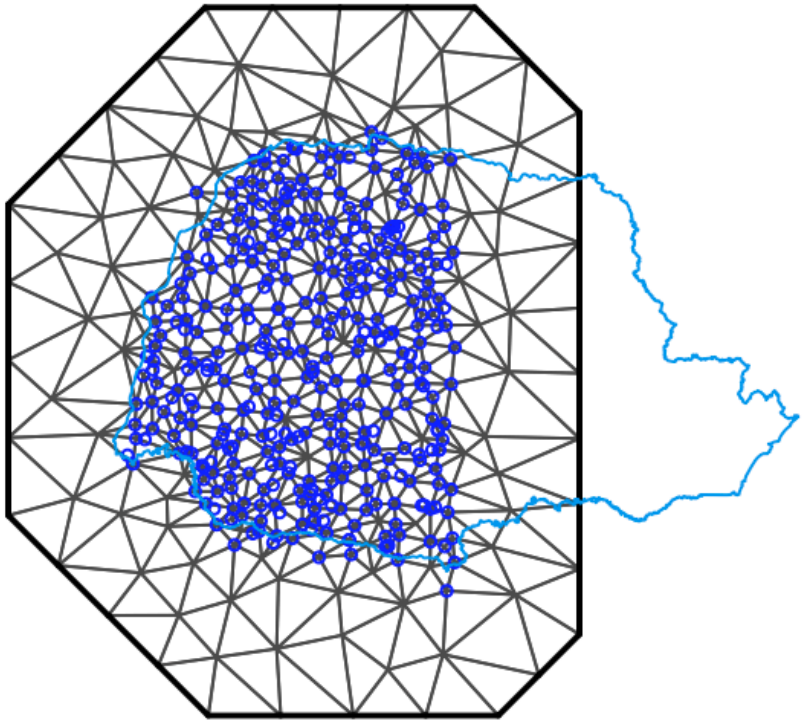
③これらの両方の制限を実装するために、最初にパラナ州の境界を考慮し、
　補助メッシュmesh2aを作成する。

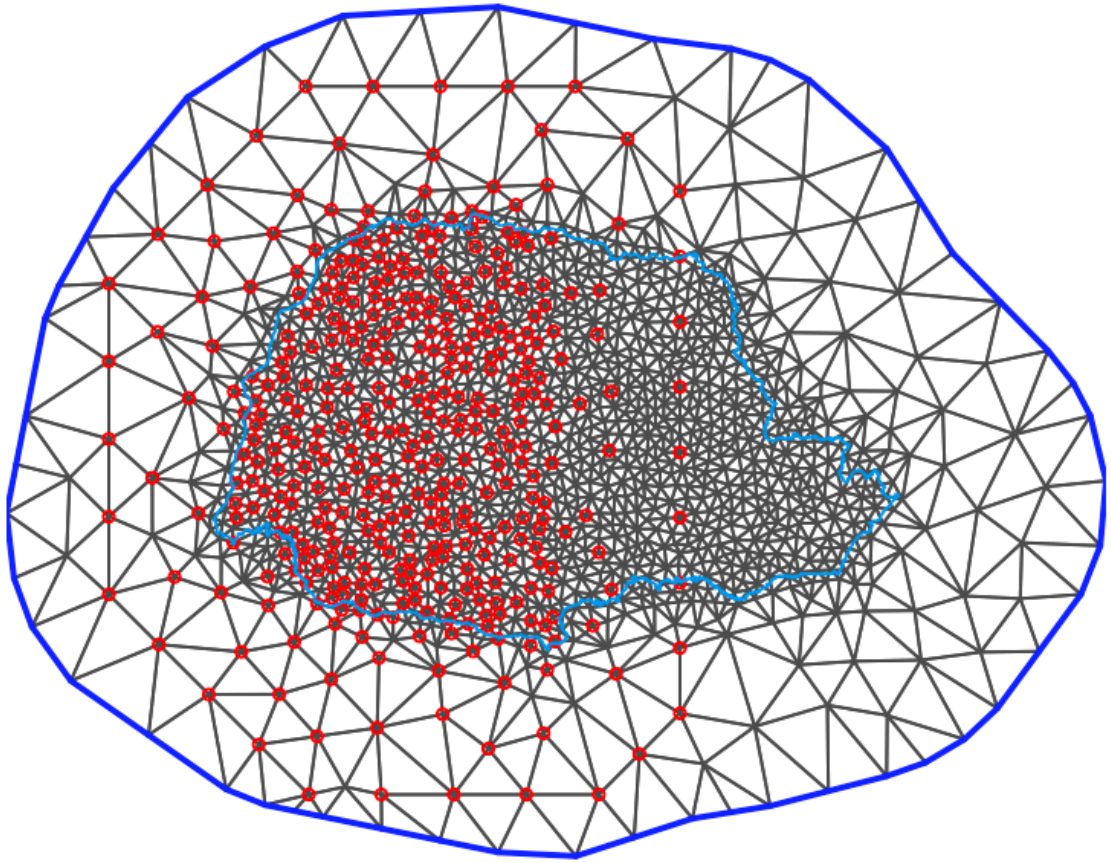
④mesh1と補助メッシュの位置を使用してmesh2を作成する。

```
ibound <- inla.nonconvex.hull(PRborder, 0.05, 2, resol = 250)
mesh2a <- inla.mesh.2d(mesh1$loc, boundary = ibound,
max.edge = 0.2, cutoff = 0.1)
bound <- inla.nonconvex.hull(PRborder, 2)
mesh2 <- inla.mesh.2d(loc = rbind(mesh1$loc, mesh2a$loc),
boundary = bound, max.edge = 1, cutoff = 0.1)
```

7.4.2 パラナ州の例: シミュレーションデータ

結果として、mesh1が379のノード、mesh2が1477のノードが得られる。





7.4.2 パラナ州の例: シミュレーションデータ

データをシミュレートするには、範囲と標準偏差を固定してから、
両方のメッシュのSPDEモデルを次のように定義する必要がある。

```
range <- 3
std.u <- 1

spde1 = inla.spde2.pcmatern(mesh1, prior.range = c(1, 0.1), prior.sigma = c(1, 0.1))
spde2 = inla.spde2.pcmatern(mesh2, prior.range = c(1, 0.1), prior.sigma = c(1, 0.1))
```

両SPDEモデルの精度行列は以下の通り得られる。

```
Q1 = inla.spde2.precision(spde1, theta = c(log(range), log(std.u)))
Q2 = inla.spde2.precision(spde2, theta = c(log(range), log(std.u)))
```

mesh2のノードでの確率場のシミュレーションは次のように実行できる。

```
u <- as.vector(inla.qsample(n = 1, Q = Q2, seed = 1))
```

7.4.2 パラナ州の例: シミュレーションデータ

メッシュノードを観測されたデータポイントに投影し、iidノイズを追加することで、データシミュレーションを完了する。 また、モデルのフィッティングに使用されるmesh1の射影行列を作成する。

```
A1 <- inla.spde.make.A(mesh1, loc = as.matrix(PRprec[sel.loc, 1:2]))
A2 <- inla.spde.make.A(mesh2, loc = as.matrix(PRprec[sel.loc, 1:2]))
```

観測場所で空間フィールドとiidガウスノイズをサンプリングする。

```
std.epsilon = 0.1
y <- drop(A2 %*% u) + rnorm(nrow(A2), sd = std.epsilon)
```

7.4.3 モデルフィッティング

スタックデータには、mesh1で定義された切片SPDEモデルが含まれる。

```
stk <- inla.stack(data = list(resp = y),
                  A = list(A1, 1),
                  effects = list(i = 1:spde1$n.spde,m = rep(1, length(y))),
                  tag = 'est')
```

フィットさせるモデルは以下の通り。

```
res <- inla(resp ~ 0 + m + f(i, model = spde1),
            data = inla.stack.data(stk),
            control.compute = list(config = TRUE),
            control.predictor = list(A = inla.stack.A(stk)))
```

7.4.3 モデルフィッティング

mesh1のみを使用して推定されたパラメータ

```
##                True      mean           sd 0.025quant 0.975quant
## Std epsilon     0.1 0.1121028 0.007032714 0.09898217  0.1265848
## Range field     3.0 2.7820764 0.732893336 1.73810739  4.5757236
## Std field       1.0 0.7637890 0.176023779 0.50545856  1.1889200
```

7.4.4 予測結果の取得

実際の予測に進む前に、近似モデルを使用して事後分布からサンプリングする必要がある。 次のハイパーパラメータの内部パラメータ化を考慮して、事後分布から100個のサンプルを抽出する。

```
nn <- 100
s <- inla.posterior.sample(n = nn, res, intern = TRUE,
                           seed = 1, add.names = FALSE)
```

以下の方法で、空間の変量効果の指標を見つけることができる。

```
contents <- res$misc$configs$contents
effect <- "i"
id.effect <- which(contents$tag == effect)
ind.effect <- contents$start[id.effect] - 1 +   (1:contents$length[id.effect])
```

7.4.4 予測結果の取得

事後分布からの各サンプルについて、mesh2のノードでの潜在フィールドuの予測を生成する。これは、モデルのフィッティングから生成された事後サンプルからの潜在フィールドの値に等しいmesh1のノードでの予測に制約される。

$$\mathbf{Ax} = \mathbf{b}$$

$$\mathbf{x} \sim N(\mu, \mathbf{Q}^{-1})$$

$$\mathbf{x}^* = \mathbf{x} - \mathbf{Q}^{-1}\mathbf{A}^T(\mathbf{A}\mathbf{Q}^{-1}\mathbf{A}^T)(\mathbf{Ax} - \mathbf{b})$$

上記数式は、精度行列のコレスキー分解を計算している。
行列Qのスパース性を考慮することにより多くの場所で高速な予測値を取得できる。
もう1つの可能性は、mesh1の代わりにmesh2を使用してモデルを直接フィットさせることである。
ただし、これにはさらに多くの計算時間が必要になり、結果はここに示す手順と同様になる。

7.4.4 予測結果の取得