Analysing data with spatial dependence

Peter Solymos and Subhash Lele

July 16, 2016 — Madison, WI — NACCB Congress

Kriging example

Exponential decay is used ($e^{-\lambda D}$), but it can be modified to half-Normal ($e^{-(\lambda D)^2}$).

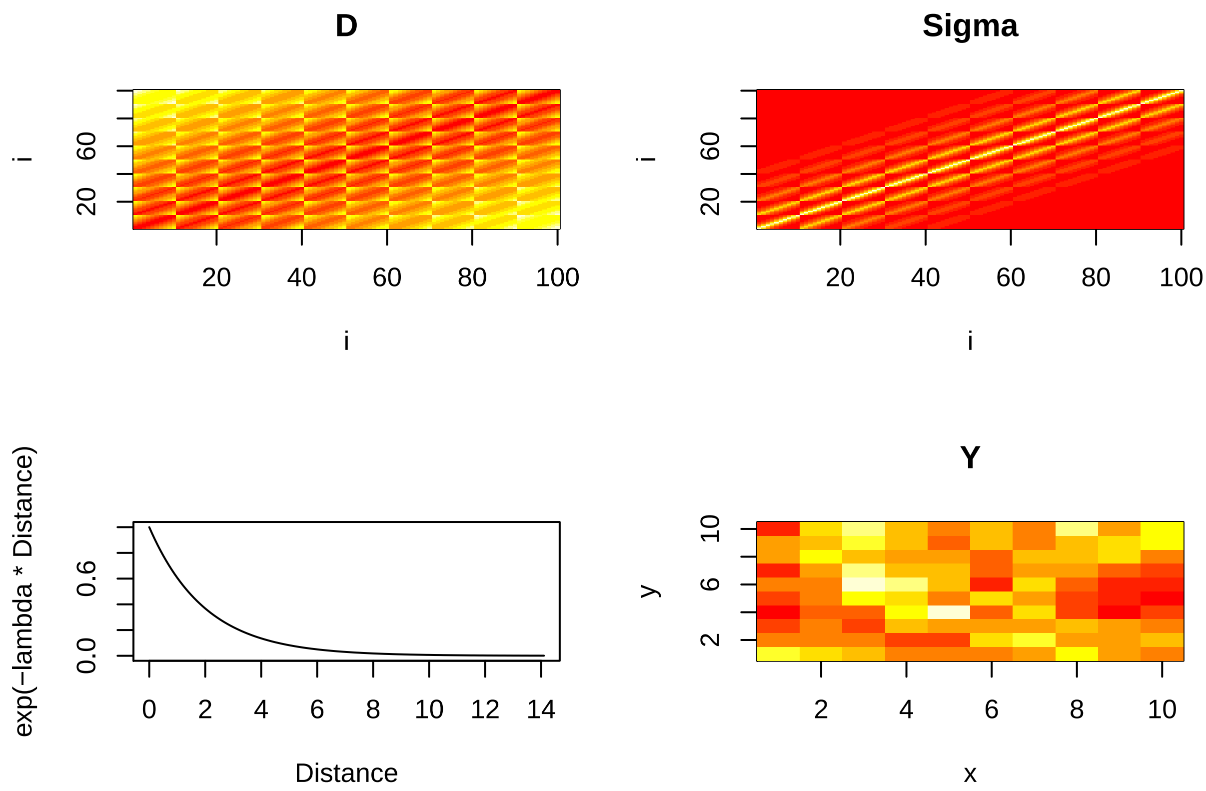
```
set.seed(2345)
library(MASS)
mu <- 5
sigma_sq <- 1
lambda <- 0.5 # try different values: 0, 0.1, 0.5, 1

## set up an m x m square lattice
m <- 10
xy <- expand.grid(x=seq_len(m), y=seq_len(m))
n <- nrow(xy)
D <- as.matrix(dist(xy))

Sigma <- sigma_sq * exp(-lambda*D)

Y <- mvrnorm(1, rep(mu, n), Sigma)

op <- par(mfrow = c(2, 2))
image(seq_len(n), seq_len(n), D, main = "D", ylab="i", xlab="i")
image(seq_len(n), seq_len(n), Sigma, main="Sigma", ylab="i", xlab="i")
Distance <- seq(0, m * sqrt(2), by = 0.1)
plot(Distance, exp(-lambda*Distance), type = "l")
image(seq_len(m), seq_len(m), matrix(Y, m, m), main = "Y", ylab="y", xlab="x")
```




```
par(op)
```

```
library(dclone)
```

```
## Loading required package: coda

## Loading required package: parallel

## Loading required package: Matrix

## dclone 2.1-1      2016-01-11
```

```
model <- custommodel("model {
    for (i in 1:n) {
        for (j in 1:n) {
            Sigma[i,j] <- sigma_sq * exp(-lambda*D[i,j])
        }
        mu_vec[i] <- mu
    }
    Y[1:n] ~ dmnorm(mu_vec, invSigma)
    invSigma <- inverse(Sigma)
    log_sigma ~ dnorm(0, 0.001)
    sigma_sq <- exp(log_sigma)^2
    mu ~ dnorm(0, 0.1)
    lambda ~ dgamma(1, 0.1)
```

```
}")
dat <- list(Y = Y, n = n, D = D)
fit <- jags.fit(data = dat, params = c("mu", "sigma_sq","lambda"),
    model = model, n.iter = 1000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1
##    Unobserved stochastic nodes: 3
##    Total graph size: 10172
##
## Initializing model
```

```
summary(fit)
```

```
##
## Iterations = 2001:3000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 1000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##            Mean     SD Naive SE Time-series SE
## lambda   0.7841 0.2822 0.005152        0.02079
## mu       5.3841 0.3307 0.006037        0.01473
## sigma_sq 0.7481 0.5157 0.009415        0.05708
##
## 2. Quantiles for each variable:
##
##            2.5%    25%    50%    75% 97.5%
## lambda   0.2087 0.6018 0.7807 0.9556 1.361
## mu       4.6933 5.2591 5.4146 5.5545 5.882
## sigma_sq 0.4295 0.5481 0.6349 0.7600 1.925
```

Inverse Wishart prior, σ^2 and λ is hard to recover:

```
library(dclone)
model <- custommodel("model {
    for (i in 1:n) {
        mu_vec[i] <- mu
    }
    Y[1:n] ~ dmnorm(mu_vec, invSigma)
    invSigma[1:n,1:n] ~ dwish(R[1:n,1:n], n)
    mu ~ dnorm(0, 0.1)
}")
dat <- list(Y = Y, n = n, R = diag(1, n, n))
fit <- jags.fit(data = dat, params = c("mu"),
    model = model, n.iter = 1000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1
##    Unobserved stochastic nodes: 2
##    Total graph size: 10008
##
## Initializing model
```

```
summary(fit)
```

```
##
## Iterations = 1001:2000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 1000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean             SD       Naive SE Time-series SE
##       5.494872       0.081576       0.001489       0.016164
##
## 2. Quantiles for each variable:
##
##  2.5%   25%   50%   75% 97.5%
## 5.347 5.441 5.492 5.545 5.671
```

Correlation (ρ) based parametrization

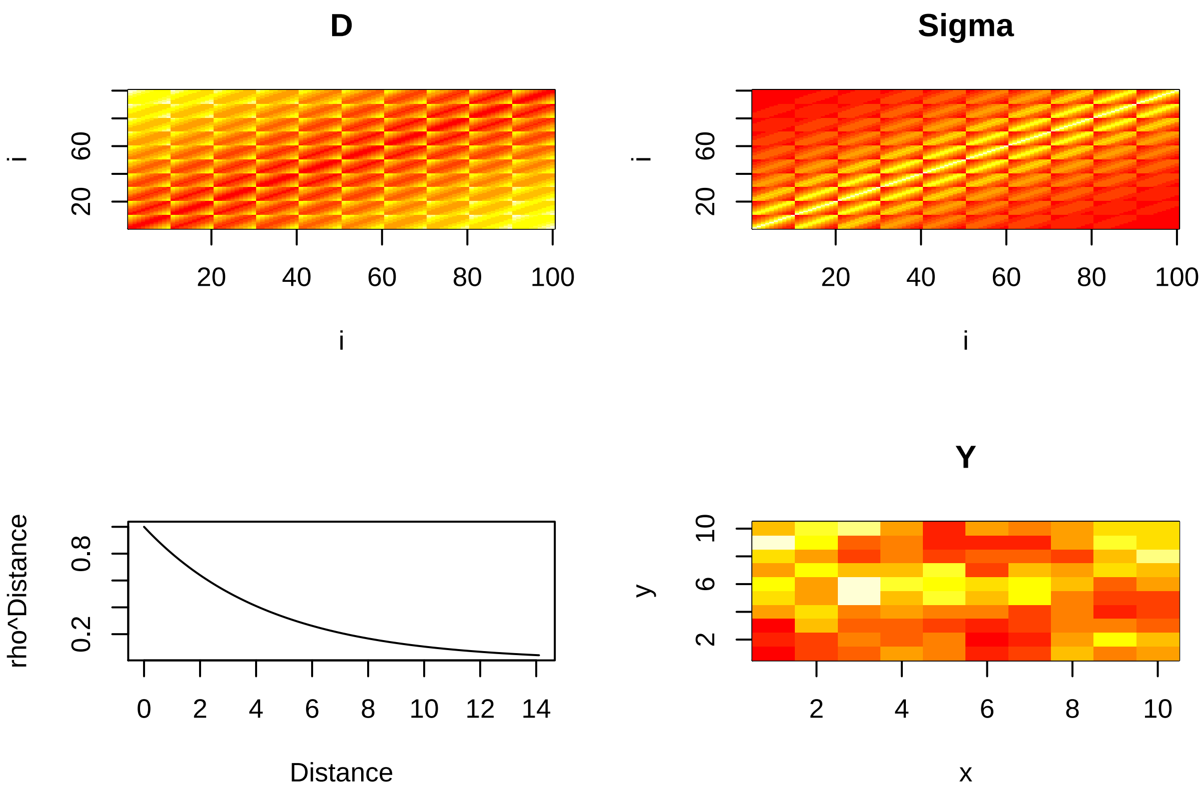
```
set.seed(2345)
library(MASS)
mu <- 5
sigma_sq <- 1
rho <- 0.8

## set up an m x m square lattice
m <- 10
xy <- expand.grid(x=seq_len(m), y=seq_len(m))
n <- nrow(xy)
D <- as.matrix(dist(xy))

Sigma <- sigma_sq * rho^D

Y <- mvrnorm(1, rep(mu, n), Sigma)

op <- par(mfrow = c(2, 2))
image(seq_len(n), seq_len(n), D, main = "D", ylab="i", xlab="i")
image(seq_len(n), seq_len(n), Sigma, main="Sigma", ylab="i", xlab="i")
Distance <- seq(0, m * sqrt(2), by = 0.1)
plot(Distance, rho^Distance, type = "l")
image(seq_len(m), seq_len(m), matrix(Y, m, m), main = "Y", ylab="y", xlab="x")
```




```
par(op)
```

```
library(dclone)
model <- custommodel("model {
    for (i in 1:n) {
        for (j in 1:n) {
            Sigma[i,j] <- sigma_sq * rho^D[i,j]
        }
        mu_vec[i] <- mu
    }
    Y[1:n] ~ dmnorm(mu_vec, invSigma)
    invSigma <- inverse(Sigma)
    log_sigma ~ dnorm(0, 0.001)
    sigma_sq <- exp(log_sigma)^2
    mu ~ dnorm(0, 0.1)
    rho ~ dunif(0, 0.999)
}")
dat <- list(Y = Y, n = n, D = D)
fit <- jags.fit(data = dat, params = c("mu", "sigma_sq","rho"),
    model = model, n.iter = 1000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1
##    Unobserved stochastic nodes: 3
```

```
##    Total graph size: 10120
##
## Initializing model
```

```r
summary(fit)
```

```
##
## Iterations = 2001:3000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 1000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##            Mean     SD Naive SE Time-series SE
## mu       5.6876 0.3273 0.005975        0.01005
## rho      0.5937 0.1250 0.002281        0.01207
## sigma_sq 0.4975 0.3927 0.007170        0.04306
##
## 2. Quantiles for each variable:
##
##            2.5%    25%    50%    75%  97.5%
## mu       5.0171 5.5635 5.7084 5.8473 6.2300
## rho      0.3898 0.5023 0.5823 0.6668 0.8926
## sigma_sq 0.2449 0.3257 0.3928 0.5092 1.6357
```