DSA 8020 R Session 13: Spatial Interpolation I

Whitney

Contents

Toy Examples

Create spatial locations

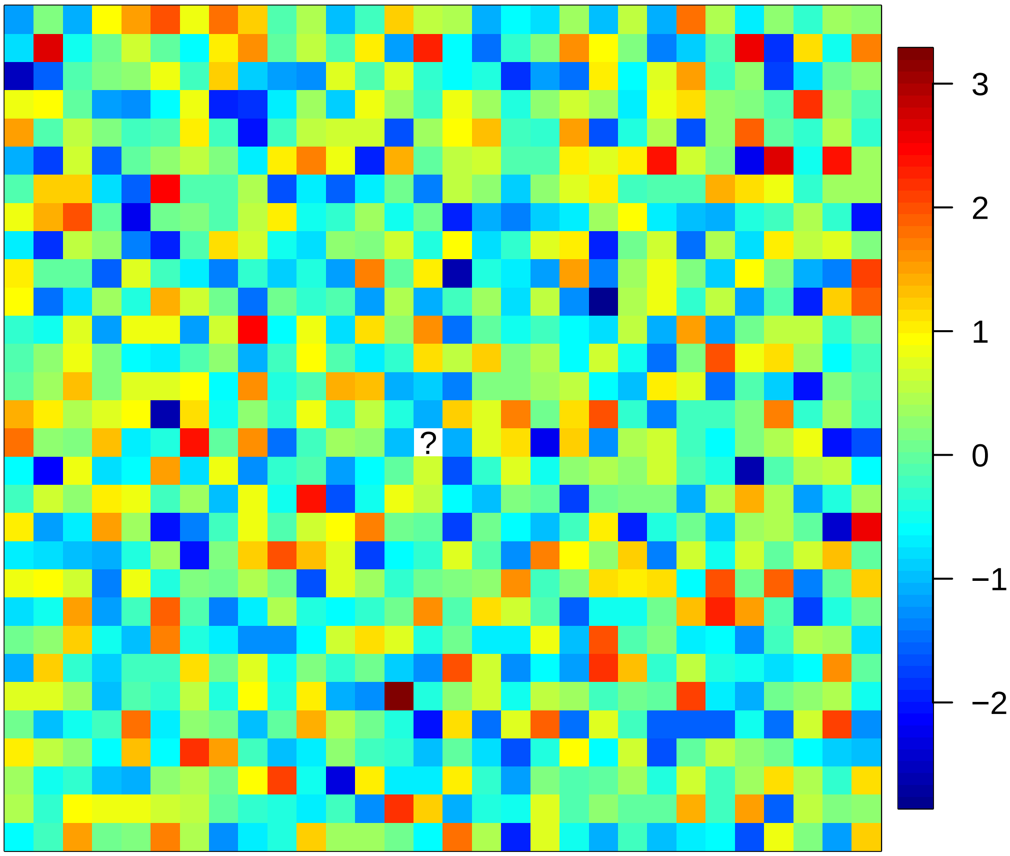
```
N = 30
xg <- yg <- seq(0, 1, length = N)
locs <- expand.grid(xg, yg)
```

Case 1: No spatial pattern

```
par(mar = c(2, 2, 1, 0.6), mgp = c(2.4, 1, 0), las = 1)
set.seed(123)
y1 <- array(rnorm(n = N^2), dim = c(N, N))
y1[N / 2, N / 2] <- NA
which(is.na(y1) == 1)
```

```
## [1] 435
```

```r
library(fields)
image.plot(xg, yg, y1, xlab = "", ylab = "", xaxt = "n", yaxt = "n")
text(xg[N / 2], yg[N / 2], "?")
```

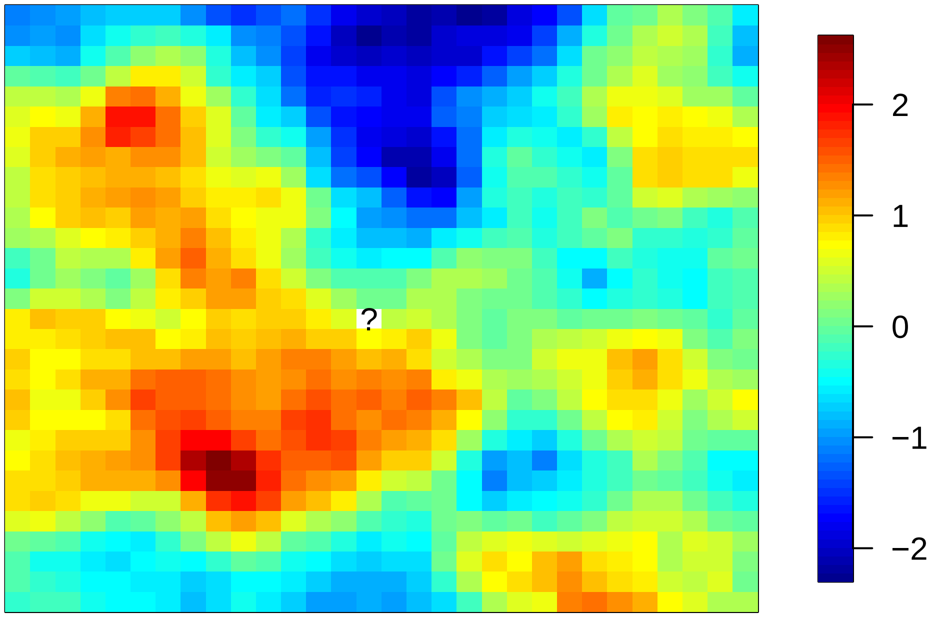


Case 2: A smooth spatial image

```r
library(MASS)
cov.Matern <- function(h, pars){
  Matern(h, phi = pars[1], range = pars[2], smoothness = pars[3])
}
dist <- rdist(locs)
Sigma_Matern <- cov.Matern(dist, c(1, 0.1, 1.5))
set.seed(123)
y2 <- array(mvrnorm(n = 1, rep(0, N^2), Sigma_Matern), dim = c(N, N))
y2[N / 2,  N / 2] <- NA
which(is.na(y2) == 1)
```

```
## [1] 435
```

```r
image.plot(xg, yg, y2, xlab = "", ylab = "", xaxt = "n", yaxt = "n")
text(xg[N / 2], yg[N / 2], "?")
```



Using Variogram Cloud and (Binned) Variogram to Examine the Spatial Dependence

```
gamma1 <- array(dim = c(N^2, N^2))
system.time(for (i in 1:N^2){
  for (j in 1:N^2){
    gamma1[i, j] <- (c(y1)[i] - c(y1)[j])^2
  }
})
```

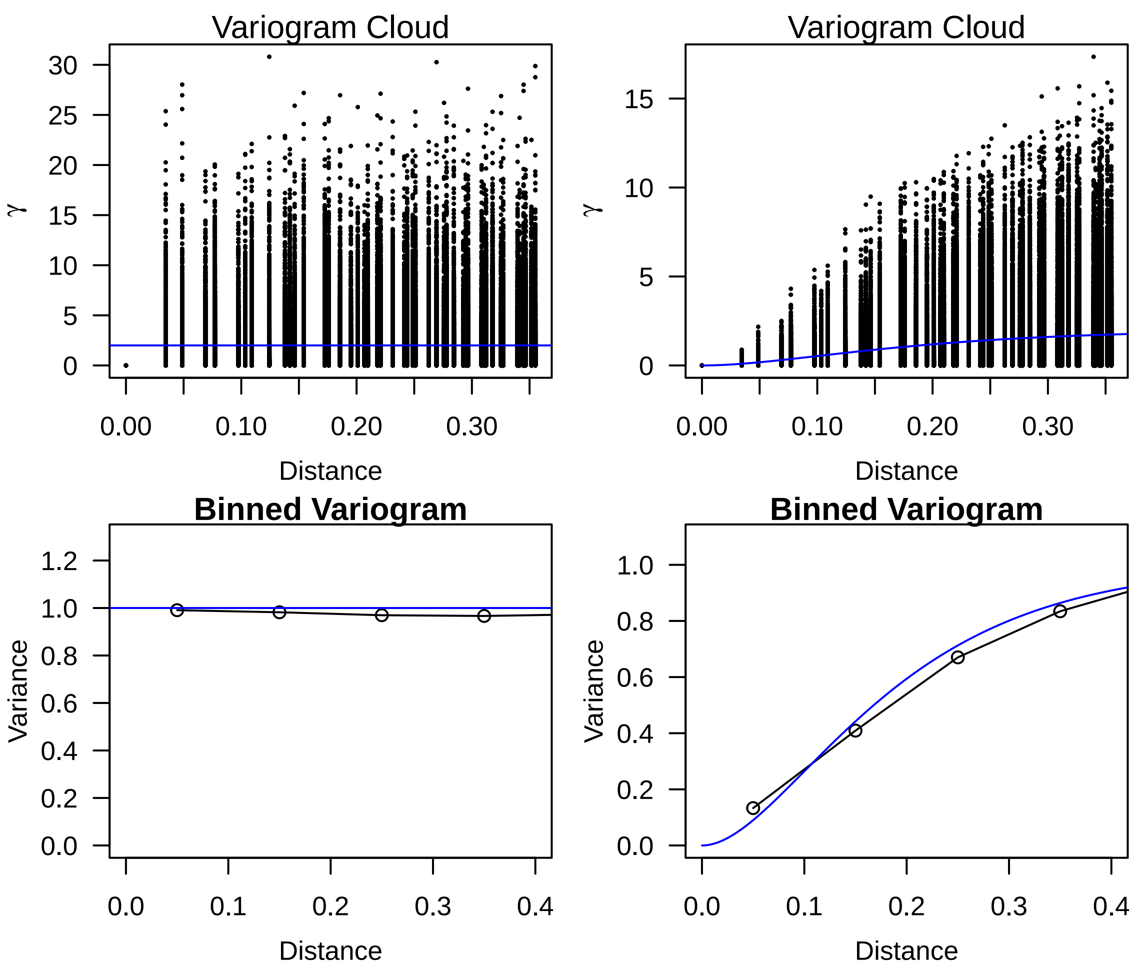
```
##    user  system elapsed
##   3.335   1.001   4.525
```

```
system.time(gamma1 <- outer(c(y1), c(y1), FUN = "-")^2)
```

```
##    user  system elapsed
##   0.003   0.003   0.007
```

```
gamma2 <- outer(c(y2), c(y2), FUN = "-")^2

dist_cut <- 0.36
good <- which(dist <= dist_cut)

par(mfrow = c(2, 2), mar = c(3.6, 3.6, 1, 0.6),
    mgp = c(2.4, 1, 0), las = 1)
plot(c(dist)[good], c(gamma1)[good],
     cex = 0.2, xlab = "Distance", ylab = expression(gamma))
abline(h = 2 * 1, col = "blue")
mtext("Variogram Cloud")
```

```r
plot(c(dist)[good], c(gamma2)[good], cex = 0.2, xlab = "Distance",
     ylab = expression(gamma))
dgrid <- seq(0, sqrt(2), 0.001)
lines(dgrid, 2 * (1 - cov.Matern(dgrid, c(1, 0.1, 1.5))),
      col = "blue")
mtext("Variogram Cloud")

vargram1 <- vgram(locs, c(y1), N = 30)
plot(vargram1, ylim = c(0, 1.3), xlim = c(0, 0.4), main = "Binned Variogram")
abline(h = 1, col = "blue")

vargram2 <- vgram(locs, c(y2), N = 30)
plot(vargram2, ylim = c(0, 1.1), xlim = c(0, 0.4), main = "Binned Variogram")
lines(dgrid, 1 - cov.Matern(dgrid, c(1, 0.1, 1.5)), col = "blue")
```

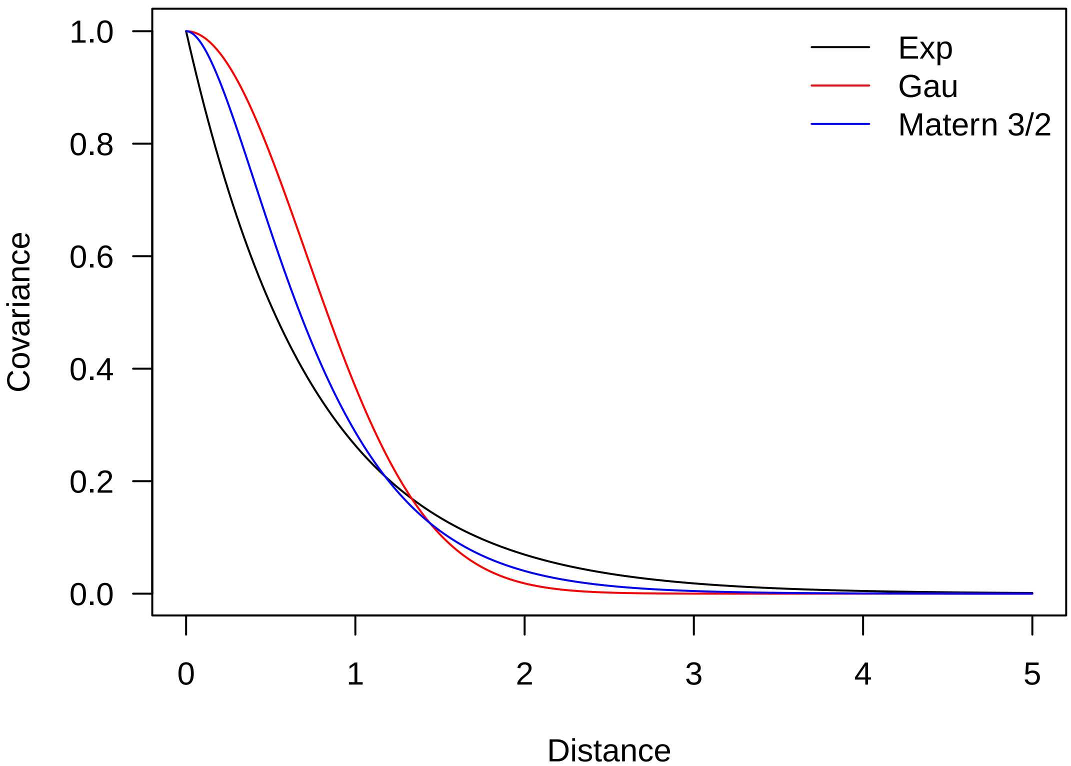


Gaussian Processes: Covariance functions and their realzations

Simulate 1D realzations

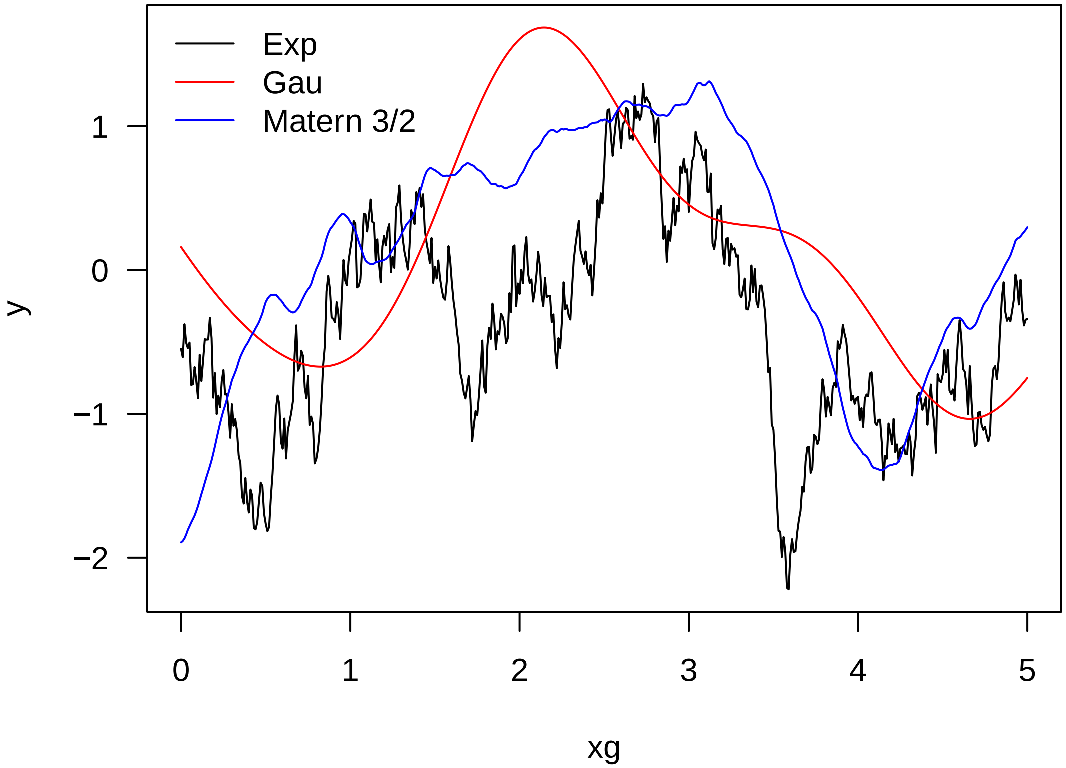
```r
# Commonly used covariance functions
cov.exp <- function(h, pars) pars[1] * exp(-h / pars[2])
```

```r
cov.doubleExp <- function(h, pars) pars[1] * exp(-(h / pars[2])^2)

xg <- seq(0, 5, 0.01)
c_exp <- cov.exp(xg, c(1, 0.75))
c_doubleExp <- cov.doubleExp(xg, c(1, 1))
c_Matern <- cov.Matern(xg, c(1, 0.4, 1.5))

plot(xg, c_exp, type = "l", ylab = "Covariance", xlab = "Distance", las = 1)
lines(xg, c_doubleExp, col = "red")
lines(xg, c_Matern, col = "blue")
legend("topright", legend = c("Exp", "Gau", "Matern 3/2"),
       col = c("black", "red", "blue"), lty = 1, bty = "n")
```




```r
Sigma_exp <- cov.exp(rdist(xg), c(1, 0.75))
Sigma_doubleExp <- cov.doubleExp(rdist(xg), c(1, 1))
Sigma_Matern <- cov.Matern(rdist(xg), c(1, 0.4, 1.5))
library(MASS)
set.seed(123)
sim_exp_1d <- mvrnorm(n = 1, rep(0, 501), Sigma_exp)
set.seed(123)
sim_doubleExp_1d <- mvrnorm(n = 1, rep(0, 501), Sigma_doubleExp)
set.seed(123)
sim_Matern_1d <- mvrnorm(n = 1, rep(0, 501), Sigma_Matern)

plot(xg, sim_exp_1d, type = "l", ylim = range(sim_exp_1d, sim_doubleExp_1d,
                                              sim_Matern_1d),
     ylab = "y", las = 1)
lines(xg, sim_doubleExp_1d, col = "red")
```

```
lines(xg, sim_Matern_1d, col = "blue")
legend("topleft", legend = c("Exp", "Gau", "Matern 3/2"),
       col = c("black", "red", "blue"), lty = 1, bty = "n")
```

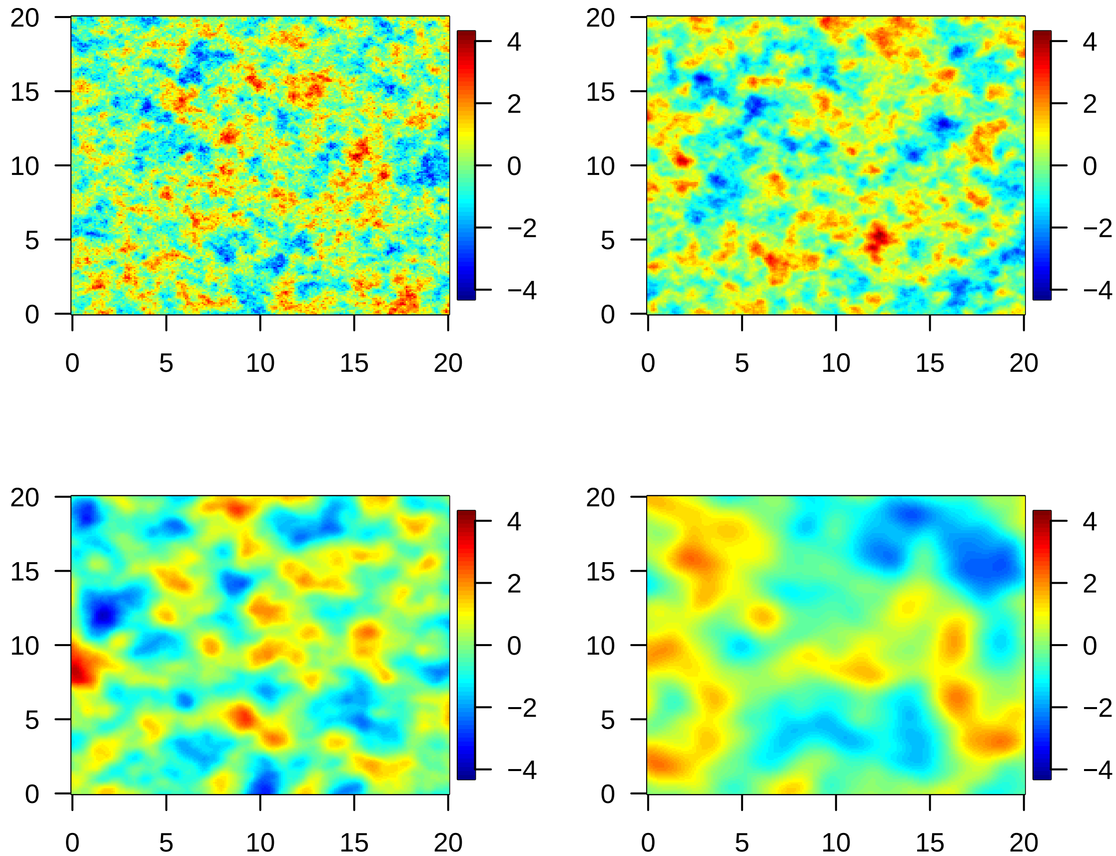


Simulate 2D realzations

```
grid <- list(x = seq(0, 20, len = 200), y = seq(0, 20, len = 200))
nu <- c(0.5, 1, 2.5, 5)
obj <- list()
for (i in 1:4) obj[[i]] <- matern.image.cov(grid = grid, theta = 0.5,
                                            smoothness = nu[i], setup = TRUE)
set.seed(2021)
sim <- lapply(obj, sim.rf)
set.panel(2, 2)
```

```
## plot window will lay out plots in a 2 by 2 matrix
```

```
par(mar = c(2.6, 3.1, 3.1, 0.6), las = 1)
for (i in 1:4){
  image.plot(grid$x, grid$y, sim[[i]], zlim = c(-4.25, 4.25), xlab = "", ylab = "")
}
```



Spatial interpolation

Here we assume $m(s) = 0 \quad \forall s \in \mathcal{S}$

Predicting one location

$$\hat{y}_0 = k^T \Sigma^{-1} \boldsymbol{y}$$
$$\text{Var}(\hat{y}_0) = \sigma^2 - k^T \Sigma^{-1} k$$

```
Sigma_Matern <- cov.Matern(dist, c(1, 0.1, 1.5))
set.seed(123)
y2 <- array(mvrnorm(n = 1, rep(0, N^2), Sigma_Matern), dim = c(N, N))
y2[N / 2,  N / 2] <- NA
# k vector
k <- Sigma_Matern[435, -435]
# Sigma matrix
Sigma <- Sigma_Matern[-435, -435]
# y vector
y <- y2[-435]
## prediction
system.time(y0_hat <- t(k) %*% solve(Sigma) %*% y)
```

```
##    user  system elapsed
##   0.314   0.001   0.318
```

```r
system.time(y0_hat_faster <-  t(k) %*% solve(Sigma, y))
```

```
##    user  system elapsed
##   0.099   0.000   0.099
```

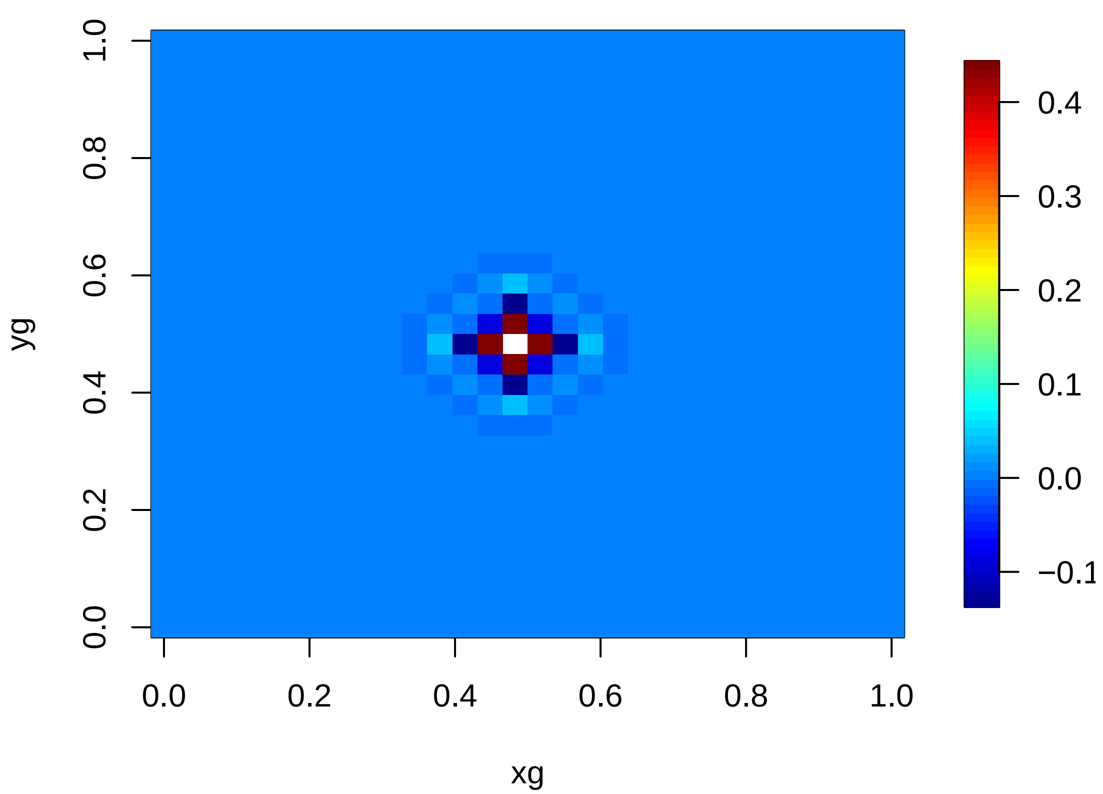
```r
## prediction uncertainty
system.time(var_y0_hat <- Sigma_Matern[435, 435] - t(k) %*% solve(Sigma) %*% k)
```

```
##    user  system elapsed
##   0.311   0.000   0.312
```

```r
system.time(Sigma_Matern[435, 435] - t(k) %*% solve(Sigma, k))
```

```
##    user  system elapsed
##     0.1     0.0     0.1
```

```r
w <- t(k) %*% solve(Sigma)
weight_map <- array(c(w[1, 1:434], NA, w[1, 435:899]),
                    dim = c(30, 30))
xg <- yg <- seq(0, 1, length = N)
image.plot(xg, yg, weight_map)
```



Predicting multiple locations

```r
N = 30
xg <- yg <- seq(0, 1, length = N)
locs <- expand.grid(xg, yg); dist <- rdist(locs)
Sigma_Matern <- cov.Matern(dist, c(1, 0.1, 1.5))
set.seed(123)
y2 <- array(mvrnorm(n = 1, rep(0, N^2), Sigma_Matern), dim = c(N, N))
y2_vec <- c(y2)
set.seed(123)
rm <- sample(1:(N^2), 0.5 * N^2)
y2_vec[rm] <- NA
y2_rm <- array(y2_vec, dim = c(N, N))

# k matrix
k <- Sigma_Matern[-rm, rm]
# Sigma matrix
Sigma <- Sigma_Matern[-rm, -rm]
# y vector
y <- y2_rm[-rm]
## prediction
system.time(y0_hat <- t(k) %*% solve(Sigma) %*% y)
```

```
##    user  system elapsed
##   0.079   0.000   0.079
```

```r
system.time(y0_hat_faster <-  t(k) %*% solve(Sigma, y))
```

```
##    user  system elapsed
##   0.014   0.000   0.015
```

```r
## prediction uncertainty
system.time(var_y0_hat <- Sigma_Matern[rm, rm] - t(k) %*% solve(Sigma) %*% k)
```

```
##    user  system elapsed
##   0.115   0.000   0.114
```

```r
system.time(Sigma_Matern[rm, rm] - t(k) %*% solve(Sigma, k))
```

```
##    user  system elapsed
##   0.092   0.000   0.092
```

```r
par(mfrow = c(1, 3), mar = c(0, 1.6, 1.2, 0),
    mgp = c(2.4, 1, 0), las = 1)
image.plot(xg, yg, y2_rm, xlab = "", ylab = "", xaxt = "n", yaxt = "n",
           horizontal = T, legend.width = 0.8, legend.line= 1)
mtext("Observed")

y2_pred <- y2_rm
y2_pred[rm] <- y0_hat[, 1]
image.plot(xg, yg, y2_pred, xlab = "", ylab = "", xaxt = "n", yaxt = "n",
           horizontal = T, legend.width = 0.8, legend.line= 1)
```

```r
mtext("Predicted")

con_sd <- sqrt(diag(var_y0_hat))
condSD <- array(0, dim = c(N, N))
temp <- c(condSD); temp[rm] <- con_sd
condSD <- array(temp, dim = c(N, N))

image.plot(xg, yg, matrix(condSD, N, N), xlab = "", ylab = "", xaxt = "n", yaxt = "n",
           horizontal = T, legend.width = 0.8, legend.line = 1)
mtext("Predicted Sd")
```