Processo gaussiano espaço temporal

Wesley Satelis

2025-06-02

Processo Gaussiano

```
require(fields)
set.seed(3)

# x e y criam vetores com 30 pontos igualmente espaçados entre 0 e 1.
x <- seq(0, 1, l=30)
y <- seq(0, 1, l=30)
tim <- 0:5

# cria a malha 2D de pontos no espaço [0,1]^2, totalizando 30×30=900 pontos.
# xy será uma matriz de coordenadas com 900 linhas e 2 colunas: cada linha é um ponto no plano.
xy <- expand.grid(x=x, y=y)

distancias <- as.matrix(dist(xy)) # distancias euclidianas

# Elevamos ao quadrado e dividimos por 0.1 → isso define o decaimento da correlação espacial.
# exp(-d^2 / .1) define uma função de covariância gaussiana (isotrópica).
G <- exp(-(distancias^2)/.1)
```

$$C(h) = \exp(\frac{-h^2}{\phi})$$

onde h é a distância entre dois pontos e $\phi = 0.1$ define o "alcance" da correlação.

```
temp <- eigen(G, symmetric = TRUE)
temp$values[temp$values < 0] <- 0 # zera autovalores negativos
```

eigen(G) faz a decomposição espectral da matriz de covariância G:

$$G = V\Lambda V^T$$

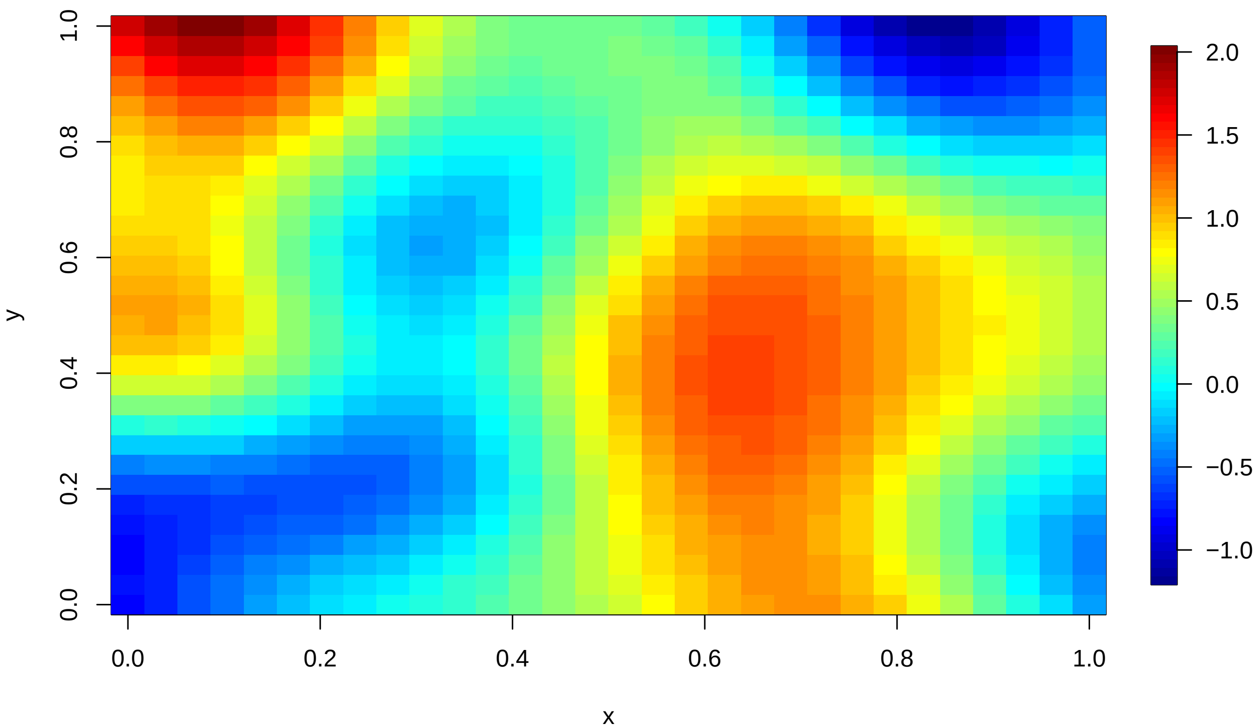
onde V são os autovetores e Λ os autovalores.

```
normal <- rnorm(nrow(xy)) # vetor normal com média 0, variância 1
DP <- diag(sqrt(temp$values)) # matriz diagonal com quadrado dos autovalores
Z <- temp$vectors %*% crossprod(DP, normal) # cria vetor N(0, G)
```

Z simula uma realização de vetor gaussiano multivariado com covariância G.

$Z = V\sqrt{\Lambda}z$ com $z \sim N(0, 1)$.

```
fields::image.plot(x, y, matrix(Z, nrow = length(x)))
```



Incluindo covariáveis espaciais

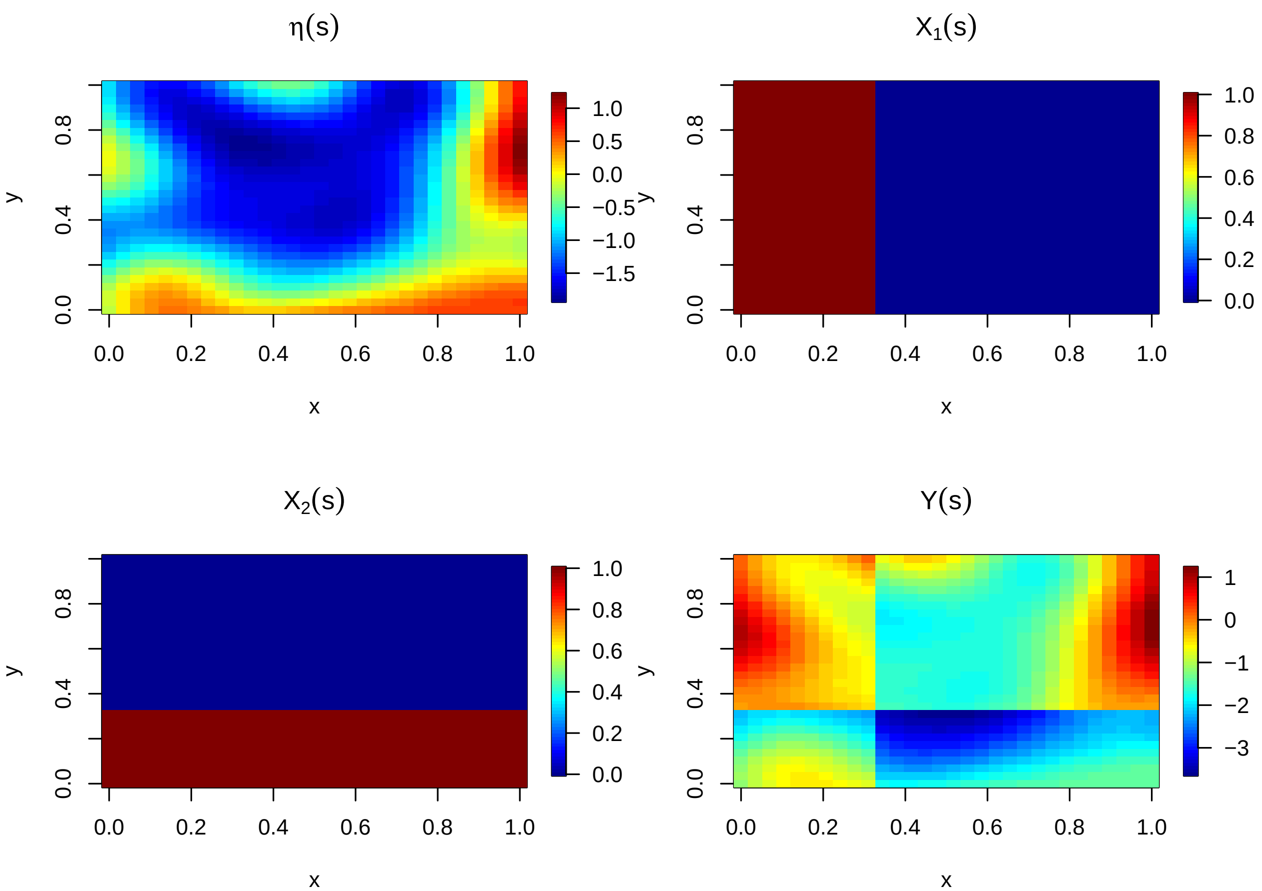
$$Y(s) = \eta(s) + \beta_1 X_1(s) + \beta_2 X_2(s) + \epsilon(s)$$

onde

- $\eta(s) \sim GP(0, C)$: campo espacial gaussiano
- $X_1, X2$ são covariáveis definidas no espaço
- β_1, β_2 são coeficientes fixos
- $\epsilon \sim N(0, \sigma^2)$ ruido

```
# Grade espacial
x <- seq(0, 1, l = 30)
y <- seq(0, 1, l = 30)
xy <- expand.grid(x = x, y = y)
n <- nrow(xy)

# Matriz de covariância espacial (gaussiana)
D <- as.matrix(dist(xy))
G <- exp(-(D^2) / 0.1)

# Decomposição espectral
temp <- eigen(G, symmetric = TRUE)
temp$values[temp$values < 0] <- 0

# Campo gaussiano
Z <- temp$vectors %*% crossprod(diag(sqrt(temp$values)), rnorm(n))
eta <- as.numeric(Z)

# ------- Covariáveis espaciais -------
# Covariável 1: um "bloco" com valor 1 no canto superior esquerdo
cov1 <- matrix(0, nrow = length(x), ncol = length(y))
```

```r
cov1[1:10, ] <- 1
X1 <- as.numeric(cov1)

# Covariável 2: faixa vertical à esquerda
cov2 <- matrix(0, nrow = length(x), ncol = length(y))
cov2[, 1:10] <- 1
X2 <- as.numeric(cov2)

# Coeficientes
beta1 <- 1.0
beta2 <- -2.0

# Resposta com covariáveis
Y <- eta + beta1 * X1 + beta2 * X2

# --- Visualizações --- #
par(mfrow = c(2, 2))

fields::image.plot(x, y, matrix(eta, nrow = length(x)), main = expression(eta(s)))
fields::image.plot(x, y, matrix(X1, nrow = length(x)), main = expression(X[1](s)))
fields::image.plot(x, y, matrix(X2, nrow = length(x)), main = expression(X[2](s)))
fields::image.plot(x, y, matrix(Y, nrow = length(x)), main = expression(Y(s)))
```



- Simula um campo gaussiano eta(s).
- Define duas covariáveis espaciais binárias:
  - X1: ativa em um bloco no canto superior esquerdo.

  - X2: ativa em uma faixa vertical à esquerda.
- Define coeficientes arbitrários $\beta_1 = 1$, $\beta_2 = -2$.
- Soma tudo para gerar uma variável resposta $Y(s)$.

$s \epsilon S$ em R^2 — um ponto em duas dimensões (neste caso, no retângulo $[0,1]^2$).

Efeito das covariáveis em Y(s) nas regiões afetadas:

- Onde X1 = 1, Y é aumentado.
- Onde X2 = 1, Y é reduzido.
- Onde ambas = 1, os efeitos se somam.

Com dependência temporal

```
# Grade espacial
x <- seq(0, 1, length = 30)
y <- seq(0, 1, length = 30)
xy <- expand.grid(x = x, y = y)
n <- nrow(xy)

# Matriz de covariância espacial
D <- as.matrix(dist(xy))
G <- exp(-(D^2) / 0.1)
eig <- eigen(G, symmetric = TRUE)
eig$values[eig$values < 0] <- 0

# Covariáveis espaciais
cov1 <- matrix(0, nrow = length(x), ncol = length(y))
cov1[1:10, ] <- 1
X1 <- as.numeric(cov1)

cov2 <- matrix(0, nrow = length(x), ncol = length(y))
cov2[, 1:10] <- 1
X2 <- as.numeric(cov2)

# betas e inicializações
beta1 <- 0.5
beta2 <- -0.5

eta_list <- list()
Y_list <- list()

tmax <- 3 # Número de tempos
phi <- 0.5  # parâmetro de dependência temporal

# t = 0
omega0 <- eig$vectors %*% crossprod(diag(sqrt(eig$values)), rnorm(n))
eta_list[[1]] <- as.numeric(omega0)
Y_list[[1]] <- eta_list[[1]] + beta1 * X1 + beta2 * X2

# t > 0
for(t in 2:(tmax+1)) {
    omega_t <- eig$vectors %*% crossprod(diag(sqrt(eig$values)), rnorm(n))
    eta_t <- phi * eta_list[[t-1]] + omega_t
    eta_list[[t]] <- as.numeric(eta_t)
```

```
    Y_list[[t]] <- eta_t + beta1 * X1 + beta2 * X2
}

# --- Visualização --- #
par(mfrow = c(3, 2))
for (t in 1:(tmax+1)) {
    fields::image.plot(x, y, matrix(Y_list[[t]], nrow = length(x)),
                       main = paste0("Y(s, t=", t-1, ")"))
}
```