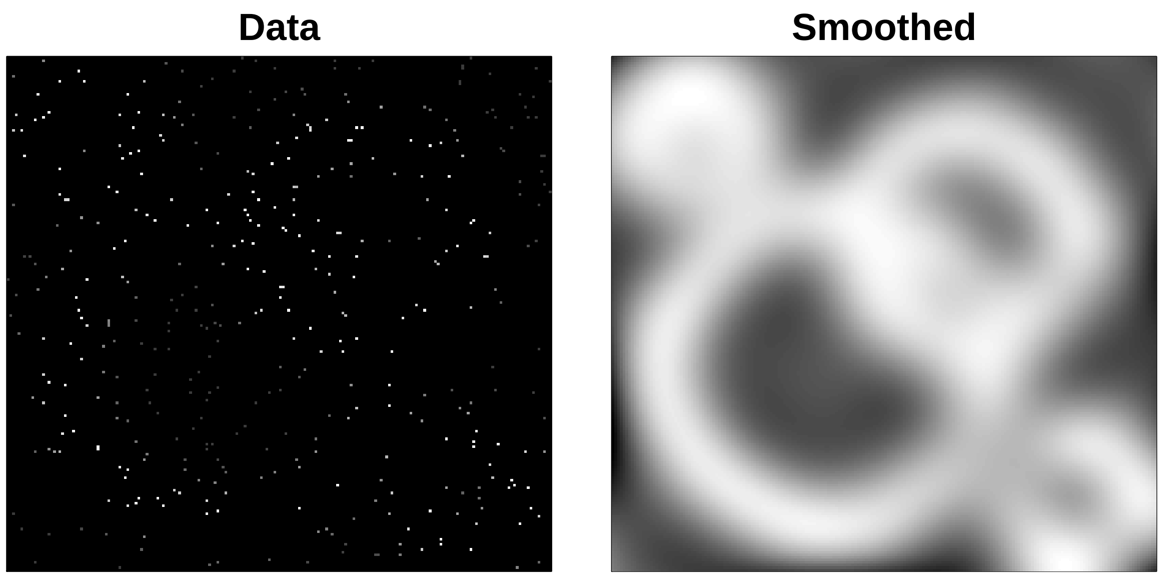
Smoothing scattered data with 2D P-splines (Simulated data)



Smoothing on a 200 by 200 grid with only 200 non-missing data points. Tensor product basis with 20 times 20 segments, second order penalties and both λs set to 1. R code in `f-scatring.R`

```
# Smoothing scattered data with 2D P-splines (Simulated data)
# A graph in the book 'Practical Smoothing. The Joys of P-splines'
# Paul Eilers and Brian Marx, 2019

library(ggplot2)
library(JOPS)
library(fields)

# Simulate the rings
nx = 200
ny = 200
x = seq(-1, 1, length = nx)
y = seq(-1, 1, length = ny)
ex = rep(1, nx)
ey = rep(1, ny)
X = outer(x, ey)
Y = outer(ex, y)
R1 = sqrt((X - 0.3)^2 + (Y - 0.3)^2)
R2 = sqrt((X + 0.2)^2 + (Y + 0.2)^2)
R3 = sqrt((X - 0.7)^2 + (Y + 0.7)^2)
R4 = sqrt((X + 0.7)^2 + (Y - 0.7)^2)
Z1 = exp(-50 * (R1 - 0.4)^2)
Z2 = exp(-50 * (R2 - 0.6)^2)
Z3 = exp(-50 * (R3 - 0.2)^2)
Z4 = exp(-50 * (R4 - 0.2)^2)
Z = pmax(pmax(pmax(Z1, Z2), Z3), Z4) + 0.3

# Prepare bases
Bx = bbase(x, nseg = 20)
By = bbase(y, nseg = 20)
nbx = ncol(Bx)
nby = ncol(By)

# Prpare the penalty matrices
Dx = diff(diag(nbx), diff = 3)
```

```
Dy = diff(diag(nby), diff = 3)
lambdax = lambday = 0.1
Px = lambdax * t(Dx) %*% Dx
Py = lambday * t(Dy) %*% Dy
P = kronecker(Py, diag(nbx)) + kronecker(diag(nby), Px)

# Do the smoothing, using the array algorithm
W = matrix(runif(nx * ny) < 0.01, nx, ny)

Tx = rowtens(Bx)
Ty = rowtens(By)
Q = t(Tx) %*% W %*% Ty
dim(Q) = c(nbx, nbx, nby, nby)
Q = aperm(Q, c(1, 3, 2, 4))
dim(Q) = c(nbx * nby, nbx * nby)
r = t(Bx) %*% (Z * W) %*% By
dim(r) = c(nbx * nby, 1)
A = solve(Q + P, r)
dim(A) = c(nbx, nby)
Zhat = Bx %*% A %*% t(By)

# Make and save plots
cols = gray(seq(0, 1, by = 0.01))
par(mfrow = c(1, 2), mar = c(1, 1, 2, 1))

image(x, y, Z * W, col = cols, xlab = "", ylab = "", xaxt = "n", yaxt = "n")
title("Data", cex.main = 1.5)
image(x, y, Zhat, col = cols, xlab = "", ylab = "", xaxt = "n", yaxt = "n")
title("Smoothed", cex.main = 1.5)
```