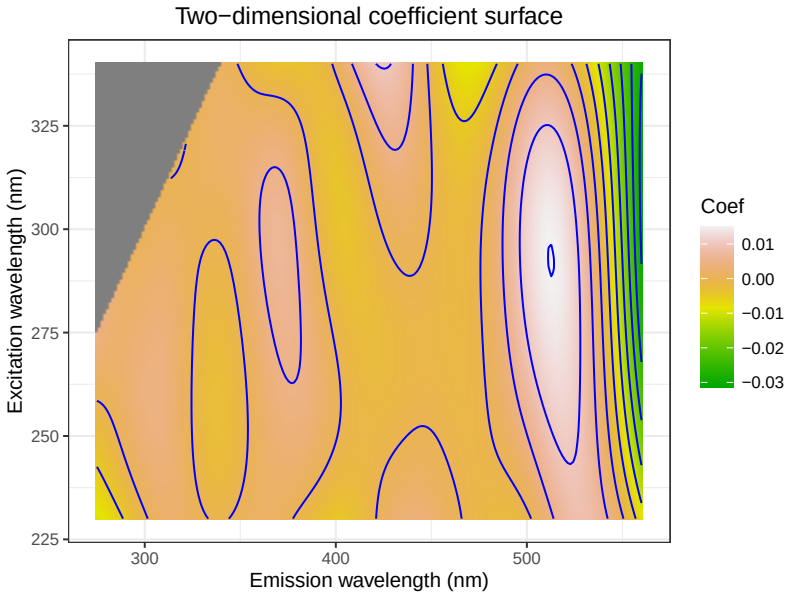
Image of optimal two-dimensional coefficient surface (Sugar data)



Optimal two-dimensional coefficient image surface using LOOCV, for response ash. R code in `f-sugar-coef-image.R`

```
# Image of optimal two-dimensional coefficient surface (Sugar data)
# A graph in the book 'Practical Smoothing. The Joys of P-splines'
# Paul Eilers and Brian Marx, 2019


library(fields)
library(ucminf)
library(data.table)
library(reshape2)
library(ggplot2)
library(JOPS)

# Get the data
x0 <- Sugar$X
x0 <- x0 - apply(x0, 1, mean) # center Signal
y <- as.vector(Sugar$y[, 3]) # Response is Ash

# Inputs for two-dimensional signal regression
nseg <- c(7, 37)
pord <- c(3, 3)
min_ <- c(230, 275)
max_ <- c(340, 560)
M1_index <- c(340, 325, 305, 290, 255, 240, 230)
M2_index <- seq(from = 275, to = 560, by = 0.5)
p1 <- length(M1_index)
p2 <- length(M2_index)
max_iter <- 100 # Number of iterations
int <- T # intercept in model

# Wrapper function to facilitate searching for optimal tuning
ucminfwrapper <- function(vec) {
  Pars_ <- rbind(
    c(min_[1], max_[1], nseg[1], 3, 10^vec[1], pord[1]),
    c(min_[2], max_[2], nseg[2], 3, 10^vec[2], pord[2])
  )
  fit <- ps2DSignal(y, x0, p1, p2, "unfolded",
```

```
                    M1_index, M2_index, Pars_, int = int)
  out <- fit$cv # LOOCV is being minimized
  out
}

# Search for best (log) lambdas
op = ucminf(c(5, 5), ucminfwrapper)
opt_lam = c(10^op$par)

# Fitting optimal model based on search
Pars <- rbind(
  c(min_[1], max_[1], nseg[1], 3, opt_lam[1], pord[1]),
  c(min_[2], max_[2], nseg[2], 3, opt_lam[2], pord[2])
)
fit <- fit_opt <- ps2DSignal(y, x0, p1, p2, "unfolded", M1_index, M2_index,
  Pars, int = int, ridge_adj = 1e-06)

# Printing of some outputs, e.g. lambdas, cv, test error
nobs <- length(y)
h <- fit_opt$h
opt_lam
test_err <- fit_opt$cv
test_err
MSE <- (sum((y - fit_opt$mu)^2) / (nobs - sum(h)))
Rstudent_train <- (c(y) - fit_opt$mu) / sqrt(MSE * (1 - h))

Resol = 200
# High resolution Grid Bases
M1grid <- seq(from = min(M1_index), to = max(M1_index), length = Resol)
M2grid <- seq(from = min(M2_index), to = max(M2_index), length = Resol)
p1 <- length(M1grid)
p2 <- length(M2grid)
oM1g <- outer(rep(1, p2), M1grid)
B1grid <- bbase(as.vector(oM1g), Pars[1, 1], Pars[1, 2], Pars[1, 3], Pars[1, 4])
oM2g <- outer(M2grid, rep(1, p1))
B2grid <- bbase(as.vector(oM2g), Pars[2, 1], Pars[2, 2], Pars[2, 3], Pars[2, 4])
n1g <- ncol(B1grid)
n2g <- ncol(B2grid)

# Compute tensor products for estimated alpha surface
B1_g <- kronecker(B1grid, t(rep(1, n2g)))
B2_g <- kronecker(t(rep(1, n1g)), B2grid)
Bgrid <- B1_g * B2_g
ncolB <- ncol(Bgrid)

# Creating mask of 1s and NAs for constraint
mask_ <- 1 * outer(M2grid, M1grid, ">")
mask_[mask_ == 0] <- NA

# Turn coeff matrix into a "long" data frame
A_hatm <- matrix(Bgrid %*% fit$pcoef[-1], Resol, Resol, byrow = T)
Mu <- t(A_hatm) * mask_
rownames(Mu) <- M2grid
colnames(Mu) <- M1grid
#browser()
dens <- melt((Mu))

names(dens) <- c("x", "y", "Coef")

# Plot coefficient image with contours
sl <- T
ccol <- "blue"
plt <- ggplot(dens, aes(x, y, fill = Coef)) +
  geom_raster(show.legend = sl) +
  scale_fill_gradientn(colours = terrain.colors(100)) +
  xlab("Emission wavelength (nm)") + ylab("Excitation wavelength (nm)") +
  ggtitle("Two-dimensional coefficient surface") +
```

```
  theme(
    plot.title = element_text(size=16),
    axis.title.x = element_text(size=16),
    axis.title.y = element_text(size=16)) +
  geom_contour(aes(z = Coef), color = ccol, show.legend = T) +
  JOPS_theme()

# Plot and save pdf
print(plt)
```