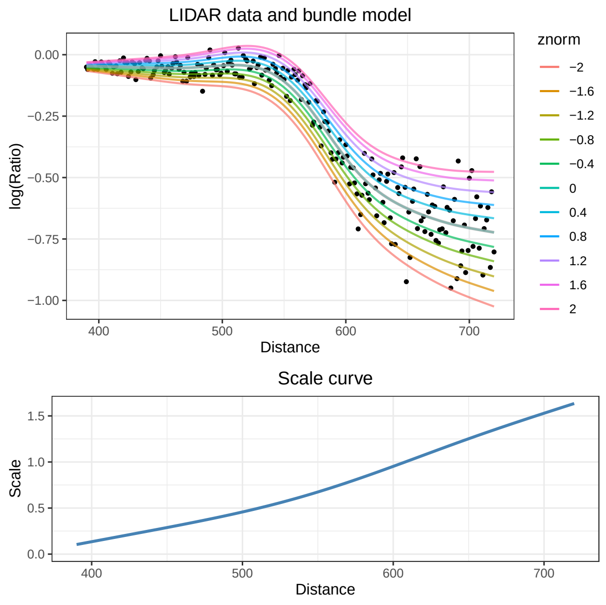
Bundle model (LIDAR data)



An expectile bundle for LIDAR data. Top panel: the bundle, where the mean curve $t(x)$ is shown in grey. Bottom panel: the scale curve $s(x)$. `R` code in `f-lidar-bundle.R`

```
# Bundle model (LIDAR data)
# A graph in the book 'Practical Smoothing. The Joys of P-splines'
# Paul Eilers and Brian Marx, 2019

library(SemiPar)
library(JOPS)
library(ggplot2)
library(gridExtra)

# Get the data
data(lidar)
m <- length(lidar$range)
sel <- seq(1, m, by = 1)
xx <- lidar$range[sel]
yy <- lidar$logratio[sel]
###selection of data from LIDAR data set

# Set asymmetry levels
zz = seq(-2, 2, by = 0.5)
sf = sqrt(2 * pi)
mu = 0
Fz = pnorm(zz )
Gz = -exp(-zz ^ 2 / 2) / sf
gf = Gz - zz * Fz
p = gf / ( 2 * gf + (zz - mu))
np <- length(p)

# Set P-spline parameters
x0 <- 350
x1 <- 750
ndx <- 20
bdeg <- 3
pdeg <- 2

# Compute bases
```

```
B <- bbase(xx, x0, x1, ndx, bdeg)
ng = 100
xg <- seq(from = min(xx), to = max(xx), length = ng)
Bg <- bbase(xg, x0, x1, ndx, bdeg)
n <- ncol(B)

# Subtract trend
lambda_t = 10
DD <- diff(diag(n), diff = 2)
P <- lambda_t * t(DD) %*% DD
a <- solve(t(B) %*% B + P, t(B) %*% yy)
trend <- B %*% a
r <- yy - trend

# Set asymmetry levels
zz = seq(-2, 2, by = 0.4)
sf = sqrt(2 * pi)
mu = 0
Fz = pnorm(zz )
Gz = -exp(-zz ^ 2 / 2) / sf
gf = Gz - zz * Fz
p = gf / ( 2 * gf + (zz - mu))

# Fit model
b <- rep(1,n)
np <- length(p)
cc <- p
lambda_a = 10
for (i in 1:20){
  b <- fitampl(r, B, b, p, cc, pdeg, lambda_a)
  b <- b / sqrt(mean(b^2))
  cnew <- fitasy(r, B, b, p, cc)
  dc = max(abs(cc - cnew))
  cc = cnew
  print(dc)
  if (dc < 1e-6) break
}
if (mean(b) < 0) {
  b = -b
  cc = -cc
}
ampl = B %*% b

# Compute bundle on grid
ampl_g <- Bg %*% b
trend_g <- Bg %*% a
Z <- ampl_g %*% cc + trend_g %*% rep(1, np)


# Standardized residuals
rst = r / ampl
umax = 1.5 * max(abs(rst))
u <- seq(-umax, umax, length=100)
nu <- length(u)

# Prepare penalty
D <- diff(diag(nu), diff = 2)
D2 <- diff(diag(nu), diff = 2)
lambda2 <- 1
P2 <- lambda2 * t(D2) %*% D2

#model matrix A
A <- matrix(0, np + 1, nu)
A[np + 1,] <- 1
for(k in 1:np){
  a1 <- (1 - p[k]) * (u - cc[k]) * (u <= cc[k])
  a2 <- p[k] * (u - cc[k]) * (u > cc[k])
  A[k,] <- a1 + a2
}

# Linear start for density
v <- solve(t(A) %*% A + P2, t(A) %*% c(rep(0,np), 1))

# Exponential model for density
q <- c(rep(0, np), 1)
lambda2 <- 10
z <- log(v - min(v) + 0.02 * max(v))
for (it2 in 1:20) {
  P2 <- lambda2 * t(D2) %*% D2
  for (it in 1:20) {
    g <- exp(z)
    r <- q - A %*% g
    B <- A * outer(rep(1, np + 1), as.vector(g))
    Q <- t(B) %*% B
    znew <- solve(Q + P2, t(B) %*%  r + Q %*% z)
    dz <- max(abs(z - znew))
    z <- znew
  #  cat(dz, '\n')
    if (dz < 1e-6) break
  }
  G = solve(Q + P2, Q)
  ed = sum(diag(G))
  v1 = sum((D2 %*% z) ^ 2) / ed
  v2 = sum(r ^ 2) / (length(r) - ed - 3)
  lanew = v2 / v1
  cat(it2, lambda2, lanew, '\n')
  dla = abs(lambda2 - lanew)
  if (dla < 1e-4 * lambda2) break
  lambda2 = lanew
}

# Build the graph with data and fit
k50 = which(p == 0.5)
Data = data.frame(x = xx, y = yy, r = rst)
Zdata = data.frame(x = rep(xg, np), y = as.vector(Z), znorm = as.factor(rep(zz, each = ng)))
Data_t = data.frame(x = xg, y = trend_g)
plt1 = ggplot(Data, aes(x = x, y = y)) +
  geom_point(size = 1) +
  geom_line(data = Zdata, aes(x = x, y = y, group = znorm, color = znorm, alpha = I(0.7)), size = 0.7, lty = 1
  geom_line(data = Data_t, color = 'darkgrey', size = 1, lty = 1, alpha = 0.7) +
  xlab("Distance") + ylab("log(Ratio)") +
  ggtitle("LIDAR data and bundle model") +
  JOPS_theme()

# Build the graph for the scale function
Data_a = data.frame(x = xg, y = ampl_g)
plt2 = ggplot(Data_a) +
  geom_line(aes(x = x, y = y), size = 1, color = 'steelblue') +
  xlab("Distance") + ylab("Scale") +
  ggtitle("Scale curve") +
  ylim(0, max(Data_a$y)) +
  JOPS_theme()

grid.arrange(plt1, plt2, nrow = 2, ncol = 1, heights = c(3, 2))

# Build graph for standardized residuals
bw = 0.05
du = u[2] - u[1]
Data_dens = data.frame(x = u, y =  bw * m * g / du)
plt3 = ggplot(Data, aes(x = r)) +
  geom_histogram(stat = "bin", fill = "wheat3", color = 'brown', binwidth = bw) +
  geom_line(data = Data_dens, aes(x = x, y = y, color = I('steelblue')), size = 1) +
  ggtitle('Standardized residuals') +
  xlab('Value') + ylab('Count') +
  JOPS_theme()

# Build graph for EE plot
Data_ee = data.frame(x = zz, y =  cc)
plt4 = ggplot(Data_ee, aes(x = x, y = y)) +
  geom_point() +
  geom_line(color = I('steelblue'), size = 1) +
  JOPS_theme()+
  ggtitle('EE plot') +
  xlab('Normal variate') + ylab('Asymmetry variate')

grid.arrange(plt3, plt4, nrow = 1, ncol = 2)
```