# Quantile Regression via the ELF loss in R

Matteo Fasiolo

## Quantile regression models

Quantile regression aims at modelling the $\tau$-th quantile (where $\tau \in (0, 1)$) of the response, $y$, conditionally on a $p$-dimensional vector of covariates, $x$. More precisely, if $F(y|x)$ is the conditional c.d.f. of y, then the $\tau$-th conditional quantile is

$$\mu = \inf\{y : F(y|x) \geq \tau\}.$$

The $\tau$-th conditional quantile can also be defined as the minimiser of the expected loss

$$L(\mu|x) = \mathbb{E}\big\{ \rho_\tau(y - \mu)|x\big\} = \int \rho_\tau(y - \mu) dF(y|x), \tag{1}$$

w.r.t. $\mu = \mu(x)$, where

$$\rho_\tau(z) = (\tau - 1)z1(z < 0) + \tau z1(z \geq 0), \tag{2}$$

is the so-called pinball loss and 1() is the indicator function. As in standard regression problems, we have $\mu = x^T\beta$, so we want to estimate the regression parameters by minimizing the pinball loss w.r.t. $\beta$. More precisely, given the response variables $y_1, \ldots, y_n$ and the correspoding covariates vectors $x_1, \ldots, x_n$, we have

$$\hat{\beta} = \text{argmin}_\beta \sum_i \rho_\tau(y_i - x_i^T\beta).$$

It turns out that $\exp(-\rho_\tau(y_i - x_i^T\beta))$ is (up to a normalizing constant) the p.d.f. of an asymmetric Laplace distribution (ALD). Hence, minimizing the pinball loss is equivalent to maximizing the likelihood of a (misspecified) ALD model. So we are back in the misspecified frequentist framework.

## Linear quantile regression using optim()

Note that the pinball loss is not smooth, hence if we want to use standard optimization methods we need to smooth it first. We do it by using the ELF loss:

$$\tilde{\rho}(y - \mu) = (\tau - 1)\frac{y - \mu}{\sigma} + \psi \log(1 + e^{\frac{y-\mu}{\psi\sigma}}), \tag{3}$$

where $\psi > 0$ and the pinball loss is recovered as $\psi \to 0$. In the following we assume that $\sigma = 1$, so we don't need to worry about it. Let $\text{Beta}(\cdot, \cdot)$ be the beta function. While $\exp(-\rho_\tau(y_i - \mu_i))$ is (up to a normalizing constant) the p.d.f. of an ALD distribution $\exp(-\tilde{\rho}_\tau(y_i - \mu_i))$ is proportional to the following p.d.f

$$\tilde{p}_F(y - \mu) = \frac{e^{(1-\tau)\frac{y-\mu}{\sigma}}(1 + e^{\frac{y-\mu}{\psi\sigma}})^{-\psi}}{\psi\sigma\text{Beta}\big\{\psi(1 - \tau), \psi\tau\big\}}. \tag{4}$$

We do not need to worry about the details of this, but see:

- Fasiolo, Matteo, Simon N. Wood, Margaux Zaffran, Raphaël Nedellec, and Yannig Goude. "Fast calibrated additive quantile regression." Journal of the American Statistical Association 116, no. 535 (2021): 1402-1412.

if you are interested. We will call this the ELF log-likelihood.

To fit the model we need functions that evaluate the negative log-likelihood and its derivatives. Here they are:

```
library(qgam)

# Computes ELF density and its derivatives w.r.t mu
dlf <- function(x, tau, mu, psi, log = FALSE, deriv = 0)
{
  sig <- 1
  y <- (x - mu) / sig

  out <- (1-tau) * y - psi * log1pexp( y / psi ) - log( sig * psi * beta(psi*(1-tau), tau*psi) )

  if( !log ) out <- exp(out)

  if( deriv > 0 )
  {
    out <- list("d" = out)

    dl <- dlogis(x, mu, psi*sig)
    pl <- plogis(x, mu, psi*sig)

    out$D <- sum((pl - (1-tau)) / sig)

    if(deriv > 1)
    {
      out$D2 <- sum( - dl / sig )
    }
  }
  return( out )
}

# Computes negative log-likelihood as a function of beta
negllkFun <- function(par, tau, psi, dat)
{

  mu <- X %*% par

  out <- - sum( dlf(x = dat, tau = tau, mu = mu, psi = psi, log = TRUE)  )

  return(out)
}
```

The following function computes derivative of the log-likelihood w.r.t. $\beta$:

```
negGrad <- function(par, tau, psi, dat)
{
  mu <- X %*% par

  tmp <- lapply(1:length(mu),
                function(ii){
```

```
                  a <- - dlf(x = dat[ii], tau = tau, mu = mu[ii],
                             psi = psi, log = TRUE, deriv = 1)$D
                  return( a[1] * X[ii, ] )
                }
  )

  out <- Reduce("+", tmp)

  return(out)
}
```

In practice, it implements the formula (this is just the chain rule):

$$\frac{\partial \log p(y_{1:N})}{\partial \beta_k} = \sum_{i=1}^{N} \frac{\partial \mu}{\partial \beta_k} \frac{\partial \log p(y_i)}{\partial \mu} = \sum_{i=1}^{N} X_{ik} \frac{\partial \log p(y_i)}{\partial \mu},$$

for $k = 1, \ldots, p$, where $p$ is number of elements of $\beta$.

To check whether this works, we consider the model:

$$y \sim \beta_1 + \beta_2 x + \beta_3 x^2 + z, \qquad z \sim N\{0, \sigma^2\}.$$

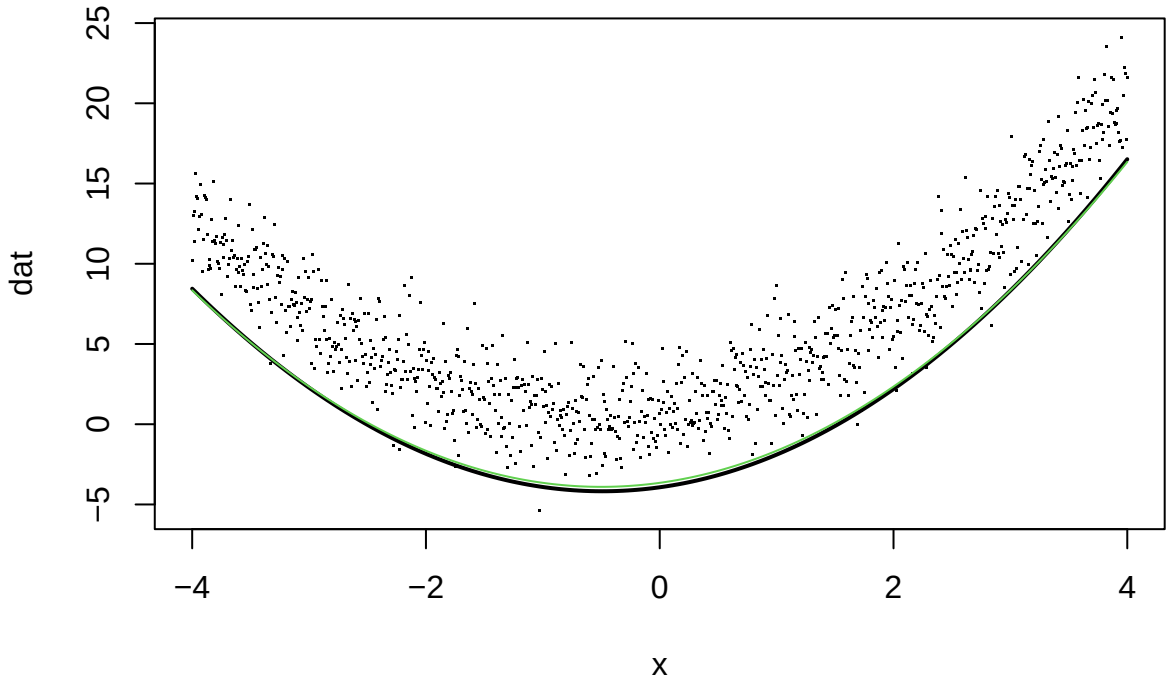
We simulate data from the model above:

```
set.seed(124)
n <- 1000
x <- seq(-4, 4, length.out = n)
X <- cbind(1, x, x^2)
beta <- c(1, 1, 1)
f <- X %*% beta
sigma =  2
dat <- f + rnorm(n, 0, sigma)
```

and we estimate the quantile $1 - \tau$

```
iBeta <- beta # Initial values
tau <- 0.01
psi <- 0.1 # Arbitrary, bias increases if you set it to a higher value (try it!)
fitB <- optim(par = iBeta, negllkFun, gr = negGrad,
              method = "BFGS", tau = tau, psi = psi, dat = dat)
```

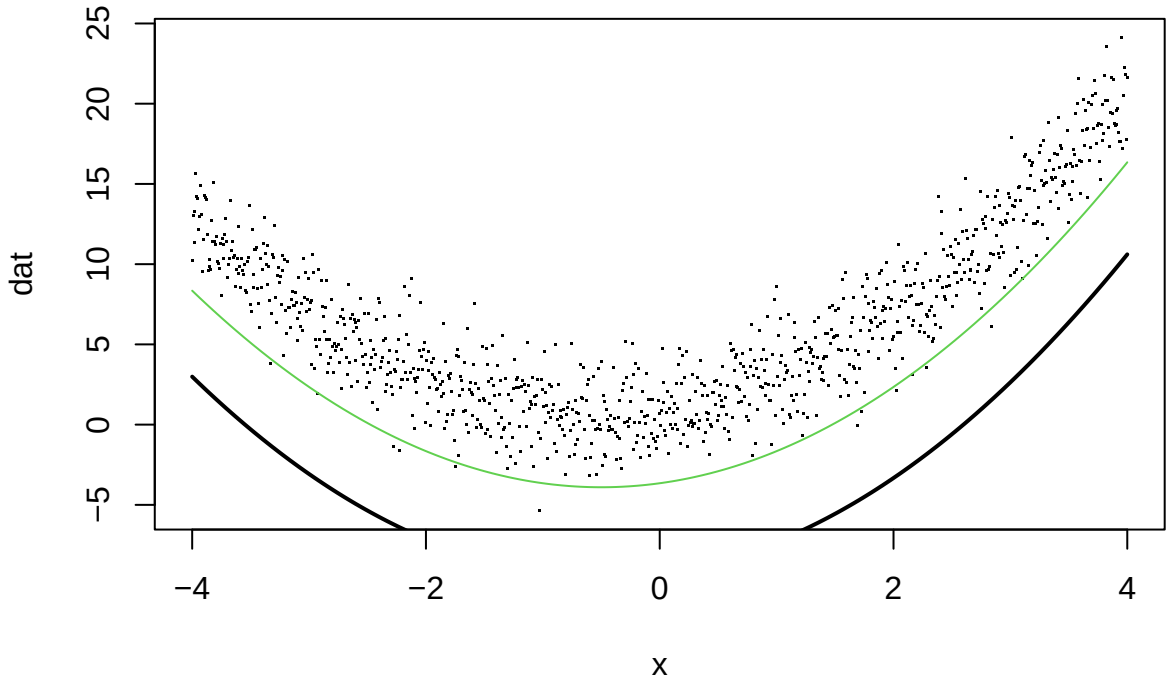
```
plot(x, dat, pch = '.')
lines(x, X%*%fitB$par, type = 'l', col = 1, lwd = 2)
lines(x, f + qnorm(tau, 0, sigma), col = 3) # Truth
```




Note that increasing $\psi$ leads to more bias:

```
psi <- 2
fitB <- optim(par = iBeta, negllkFun, gr = negGrad,
              method = "BFGS", tau = tau, psi = psi, dat = dat)

plot(x, dat, pch = '.')
lines(x, X%*%fitB$par, type = 'l', col = 1, lwd = 2)
lines(x, f + qnorm(tau, 0, sigma), col = 3) # Truth
```




while decreasing it too much lead to numerical problems (because the ELF loss becomes non-smooth). Hence,

$\psi$ should be reasonably small, but not too small.