```
# CODE FOR FITTING M-QUANTILE REGRESSION

library(MASS)

QRLM <- function (x, y, case.weights = rep(1, nrow(x)), var.weights =
rep(1, nrow(x)), ..., w = rep(1, nrow(x)), init = "ls", psi = psi.huber,
scale.est = c("MAD", "Huber", "proposal 2"), k2 = 1.345, method = c("M",
"MM"), maxit = 20, acc = 1e-04, test.vec = "resid", q = 0.5)
{
irls.delta <- function(old, new) sqrt(sum((old - new)^2)/max(1e-20,
sum(old^2)))
irls.rrxwr <- function(x, w, r) {
w <- sqrt(w)
max(abs((matrix(r*w,1,length(r)) %*% x)/sqrt(matrix(w,1,length(r)) %*%
(x^2))))/sqrt(sum(w*r^2))
}
method <- match.arg(method)
nmx <- deparse(substitute(x))
if (is.null(dim(x))) {
x <- as.matrix(x)
colnames(x) <- nmx
}
else x <- as.matrix(x)
if (is.null(colnames(x)))
colnames(x) <- paste("X", seq(ncol(x)), sep = "")
if (qr(x)$rank < ncol(x))
stop("x is singular: singular fits are not implemented in rlm")
if (!(any(test.vec == c("resid", "coef", "w", "NULL")) ||
is.null(test.vec)))
stop("invalid testvec")
if (length(var.weights) != nrow(x))
stop("Length of var.weights must equal number of observations")
if (any(var.weights < 0))
stop("Negative var.weights value")
if (length(case.weights) != nrow(x))
stop("Length of case.weights must equal number of observations")
w <- (w * case.weights)/var.weights
if (method == "M") {
scale.est <- match.arg(scale.est)
if (!is.function(psi))
psi <- get(psi, mode = "function")
arguments <- list(...)
if (length(arguments)) {
pm <- pmatch(names(arguments), names(formals(psi)), nomatch = 0)
if (any(pm == 0))
warning(paste("some of ... do not match"))
pm <- names(arguments)[pm > 0]
formals(psi)[pm] <- unlist(arguments[pm])
}
```

```
if (is.character(init)) {
if (init == "ls")
temp <- lm.wfit(x, y, w, method = "qr")
else if (init == "lts")
temp <- lqs.default(x, y, intercept = FALSE, nsamp = 200)
else stop("init method is unknown")
coef <- temp$coef
resid <- temp$resid
}
else {
if (is.list(init))
coef <- init$coef
else coef <- init
resid <- y - x %*% coef
}
}
else if (method == "MM") {
scale.est <- "MM"
temp <- lqs.default(x, y, intercept = FALSE, method = "S", k0 = 1.548)
coef <- temp$coef
resid <- temp$resid
psi <- psi.bisquare
if (length(arguments <- list(...)))
if (match("c", names(arguments), nomatch = FALSE)) {
c0 <- arguments$c
if (c0 > 1.548) {
psi$c <- c0
}
else warning("c must be at least 1.548 and has been ignored")
}
scale <- temp$scale
}
else stop("method is unknown")
done <- FALSE
conv <- NULL
n1 <- nrow(x) - ncol(x)
if (scale.est != "MM")
scale <- mad(resid/sqrt(var.weights), 0)
theta <- 2 * pnorm(k2) - 1
gamma <- theta + k2^2 * (1 - theta) - 2 * k2 * dnorm(k2)
qest <- matrix(0, nrow = ncol(x), ncol = length(q))
qwt <- matrix(0, nrow = nrow(x), ncol = length(q))
qfit <- matrix(0, nrow = nrow(x), ncol = length(q))
qres <- matrix(0, nrow = nrow(x), ncol = length(q))
for(i in 1:length(q)) {
for (iiter in 1:maxit) {
if (!is.null(test.vec))
testpv <- get(test.vec)
if (scale.est != "MM") {
if (scale.est == "MAD")
```

```
scale <- median(abs(resid/sqrt(var.weights)))/0.6745
else scale <- sqrt(sum(pmin(resid^2/var.weights,(k2*scale)^2))/(n1*gamma))
if (scale == 0) {
done <- TRUE
break
}
}
w <- psi(resid/(scale * sqrt(var.weights))) * case.weights
ww <- 2 * (1 - q[i]) * w
ww[resid > 0] <- 2 * q[i] * w[resid > 0]
w <- ww
temp <- lm.wfit(x, y, w, method = "qr")
coef <- temp$coef
resid <- temp$residuals
if (!is.null(test.vec))
convi <- irls.delta(testpv, get(test.vec))
else convi <- irls.rrxwr(x, wmod, resid)
conv <- c(conv, convi)
done <- (convi <= acc)
if (done)
break
}
if (!done)
warning(paste("rlm failed to converge in", maxit, "steps at q = ", q[i]))
qest[, i] <- coef
qwt[, i] <- w
qfit[, i] <- temp$fitted.values
qres[,i] <- resid
}
list(fitted.values = qfit, residuals = qres, q.values = q, q.weights = qwt,
coefficients = qest)
}

#APPLICATION TO ENGELíS DATA OF ALTERNATIVE APPROACHES TO #QUANTILE
REGRESSION


library(quantreg)

data(engel)

ccc=1.345

cons=rep(1,235)
x=matrix(cbind(cons,engel$income),nrow=235,ncol=2)
y= engel$foodexp

MQmodel<-QRLM(x,y,maxit=100,q=0.5,k=ccc) #fit the QRLM model with x design
#matrix and y the response variable
s<-median(abs(MQmodel$res))/0.6745 #compute the MAD
```

```
p=2 #covariates
quant=0.5 #quantile
n=235 #sample size
resid=MQmodel$res/s #standardized residuals
cc=ccc #tunue constant
Epsi2=sum((MQmodel$q.weights*resid)^2)/(n-p) # estimation of the #expexted
value of psi^2
Epsi=(sum(2*(quant*(0<=resid & resid<=cc)+(1-quant)*(-cc<=resid &
resid<0)))/n) #compute the expected value of the derivate of psi
U=diag(2*c(quant*(0<=resid & resid<=cc)+(1-quant)*(-cc<=resid &
resid<0)),n,n)
W=diag(c((MQmodel$q.weights^2)*(resid^2)),n,n)
xt=t(x) #x transpose
lambda=1+(p/n)*((1-Epsi)/Epsi)
#cov
M1=((s^2)*lambda*n/(n-p))*solve(xt%*%U%*%x)%*%(xt%*%W%*%x)%*%solve(xt%*%U%*
%x) #matrix var-cov

M2=(s^2)*(lambda*Epsi2/Epsi^2)*solve(xt%*%x) #matrix var-cov


#Least Squares Fit

ls=summary(lm(engel$foodexp~engel$income))

lsb0=ls$coefficients[1]
lsb1=ls$coefficients[2]
vlsb0=(ls)$coefficients[3]
vlsb1=(ls)$coefficients[4]

# Quantile Fit
f=summary(rq(engel$foodexp~engel$income, tau=0.5) ,se= "iid")

# Robust, M-type Fit
rob=rlm(engel$foodexp~engel$income)

br0=rob$coefficients[1]
br1=rob$coefficients[2]


vrobb0=summary(rob)$coefficients[3]
vrobb1=summary(rob)$coefficients[4]



# M-quantile Fit

mqb0=MQmodel$coefficients[1,1]
mqb1= MQmodel$coefficients[2,1]
```

```
vb0=sqrt(M2[1,1])
vb1=sqrt(M2[2,2])


qb0=f$coefficients[1]
qb1=f$coefficients[2]


vqb0=f$coefficients[3]
vqb1=f$coefficients[4]


# Tables of Results

results.coefficients= matrix(rbind(qb0,qb1,mqb0,mqb1,
br0,br1,lsb0,lsb1) ,nrow=2,ncol=4,dimnames= list(c("B0", "B1"),c("Quantile
Model",
"M-quantile Model", "M-regression (Huber)", "Least Squares")))



results.sterrors= matrix(rbind(vqb0,vqb1,vb0,vb1, vrobb0,vrobb1,
vlsb0,vlsb1),nrow=2,ncol=4,
dimnames= list(c("B0", "B1"), c("Quantile Model", "M-quantile Model", "M-
regression (Huber)", "Least Squares")))
```