Main updates to rdrobust package, June 2022

1. Definition of dups/dupsid for vce=”nn” (rdrobust/rdbwselect)

OLD CODE:

for (i in 1:eN\_l) edups\_l[i]=sum(eX\_l==eX\_l[i])

for (i in 1:eN\_r) edups\_r[i]=sum(eX\_r==eX\_r[i])

i=1

while (i<=eN\_l) {

edupsid\_l[i:(i+edups\_l[i]-1)] = 1:edups\_l[i]

i = i+edups\_l[i]

}

i=1

while (i<=eN\_r) {

edupsid\_r[i:(i+edups\_r[i]-1)]=1:edups\_r[i]

i=i+edups\_r[i]

}

NEW CODE:

nn\_l = rep(1,N\_l)

nn\_r = rep(1,N\_r)

dups\_l = ave(nn\_l, X\_l, FUN = sum)

dups\_r = ave(nn\_r, X\_r, FUN = sum)

dupsid\_l = ave(nn\_l, X\_l, FUN = cumsum)

dupsid\_r = ave(nn\_r, X\_r, FUN = cumsum)

1. vce=”hc2”, ”hc3” (rdrobust/rdrobust\_bw function)

OLD CODE

hii\_l=matrix(NA,eN\_l,1); hii\_r=matrix(NA,eN\_r,1)

for (i in 1:eN\_l) hii\_l[i] = R\_p\_l[i,]%\*%invG\_p\_l%\*%(R\_p\_l\*W\_h\_l)[i,]

for (i in 1:eN\_r) hii\_r[i] = R\_p\_r[i,]%\*%invG\_p\_r%\*%(R\_p\_r\*W\_h\_r)[i,]

NEW CODE

hii\_l = rowSums((R\_p\_l%\*%invG\_p\_l)\*(R\_p\_l\*W\_h\_l))

hii\_r = rowSums((R\_p\_r%\*%invG\_p\_r)\*(R\_p\_r\*W\_h\_r))

1. Rdplot

OLD CODE

bin\_x\_l = rep(0,length(x\_l)); bin\_x\_r = rep(0,length(x\_r))

for (k in 1:(J\_star\_l-1)) bin\_x\_l[x\_l>=jumps\_l[k] & x\_l<jumps\_l[k+1]] = -J\_star\_l+k-1

bin\_x\_l[x\_l>=jumps\_l[(J\_star\_l)]] = -1

for (k in 1:(J\_star\_r-1)) bin\_x\_r[x\_r>=jumps\_r[k] & x\_r<jumps\_r[k+1]] = k

bin\_x\_r[x\_r>=jumps\_r[(J\_star\_r)]] = J\_star\_r

rdplot\_mean\_bin\_l=rdplot\_mean\_x\_l=rdplot\_mean\_y\_l=rep(0,J\_star\_l)

rdplot\_mean\_bin\_r=rdplot\_mean\_x\_r=rdplot\_mean\_y\_r=rep(0,J\_star\_r)

for (k in 1:(J\_star\_l)) {

rdplot\_mean\_bin\_l[k] = mean(c(jumps\_l[k],jumps\_l[k+1]))

rdplot\_mean\_x\_l[k] = mean(x\_l[bin\_x\_l==-k])

rdplot\_mean\_y\_l[k] = mean(y\_l[bin\_x\_l==-k])

if (!is.null(covs) & covs\_eval=="mean") rdplot\_mean\_y\_l[k] = mean(yhatZ\_l[bin\_x\_l==-k])

}

rdplot\_mean\_y\_l = rev(rdplot\_mean\_y\_l)

rdplot\_mean\_x\_l = rev(rdplot\_mean\_x\_l)

for (k in 1:(J\_star\_r)) {

rdplot\_mean\_bin\_r[k] = mean(c(jumps\_r[k],jumps\_r[k+1]))

rdplot\_mean\_x\_r[k] = mean(x\_r[bin\_x\_r==k])

rdplot\_mean\_y\_r[k] = mean(y\_r[bin\_x\_r==k])

if (!is.null(covs) & covs\_eval=="mean") rdplot\_mean\_y\_r[k] = mean(yhatZ\_r[bin\_x\_r==k])

}

rdplot\_mean\_bin\_l[J\_star\_l]=mean(c(jumps\_l[J\_star\_l],c))

rdplot\_mean\_bin\_r[J\_star\_r]=mean(c(jumps\_r[J\_star\_r],x\_max))

rdplot\_sd\_y\_l=rdplot\_N\_l=rdplot\_sd\_y\_r=rdplot\_N\_r=0

for (j in 1:(J\_star\_l)) {

rdplot\_sd\_y\_l[j] = sd(y\_l[bin\_x\_l==-j])

rdplot\_N\_l[j] = length(y\_l[bin\_x\_l==-j])

}

for (j in 1:(J\_star\_r)) {

rdplot\_sd\_y\_r[j] = sd(y\_r[bin\_x\_r==j])

rdplot\_N\_r[j] = length(y\_r[bin\_x\_r==j])

}

NEW CODE:

bin\_x\_l = findInterval(x\_l, jumps\_l, rightmost.closed=TRUE) - J\_star\_l-1

bin\_x\_r = findInterval(x\_r, jumps\_r, rightmost.closed=TRUE)

rdplot\_l = aggregate(cbind(y\_l,x\_l), list(bin\_x\_l), FUN=mean)

rdplot\_r = aggregate(cbind(y\_r,x\_r), list(bin\_x\_r), FUN=mean)

rdplot\_bin\_l = rdplot\_l[,1]; rdplot\_mean\_y\_l = rdplot\_l[,2]; rdplot\_mean\_x\_l = rdplot\_l[,3]

rdplot\_bin\_r = rdplot\_r[,1]; rdplot\_mean\_y\_r = rdplot\_r[,2]; rdplot\_mean\_x\_r = rdplot\_r[,3]

t\_ind\_l = 1:J\_star\_l

t\_ind\_r = 1:J\_star\_r

rdplot\_mean\_bin\_l = rowMeans(cbind(matrix(jumps\_l[t\_ind\_l]),matrix(jumps\_l[(t\_ind\_l+1)])))

rdplot\_mean\_bin\_r = rowMeans(cbind(matrix(jumps\_r[t\_ind\_r]),matrix(jumps\_r[(t\_ind\_r+1)])))

rdplot\_mean\_bin\_l = rdplot\_mean\_bin\_l[rev(-rdplot\_bin\_l)]

rdplot\_mean\_bin\_r = rdplot\_mean\_bin\_r[rdplot\_bin\_r]

bin\_x = c(bin\_x\_l,bin\_x\_r)

rdplot\_mean\_bin = c(rdplot\_mean\_bin\_l, rdplot\_mean\_bin\_r)

rdplot\_mean\_x = c(rdplot\_mean\_x\_l, rdplot\_mean\_x\_r)

rdplot\_mean\_y = c(rdplot\_mean\_y\_l, rdplot\_mean\_y\_r)

rdplot\_N\_l = aggregate(y\_l, list(-bin\_x\_l), FUN=length)[,2]

rdplot\_N\_r = aggregate(y\_r, list(bin\_x\_r), FUN=length)[,2]

rdplot\_sd\_y\_l = aggregate(y\_l, list(-bin\_x\_l), FUN=sd)[,2]

rdplot\_sd\_y\_r = aggregate(y\_r, list(bin\_x\_r), FUN=sd)[,2]

1. rdrobust\_vce function

OLD CODE

rdrobust\_vce = function(d, s, RX, res, C) {

k = ncol(as.matrix(RX))

M = matrix(0,k,k)

n = length(C)

if (is.null(C)) {

w = 1

if (d==0){

M = crossprod(c(res)\*RX)

}

else {

for (i in 1:(1+d)) {

SS = res[,i]\*res

for (j in 1:(1+d)) {

M = M + crossprod(RX\*(s[i]\*s[j])\*SS[,j],RX)

}

}

}

}

else {

clusters = unique(C)

g = length(clusters)

w=((n-1)/(n-k))\*(g/(g-1))

if (d==0){

for (i in 1:g) {

ind=C==clusters[i]

Xi = RX[ind,,drop=FALSE]

ri = res[ind,,drop=FALSE]

M = M + crossprod(t(crossprod(Xi,ri)),t(crossprod(Xi,ri)))

}

}

else {

for (i in 1:g) {

ind=C==clusters[i]

Xi = RX[ind,,drop=FALSE]

ri = res[ind,,drop=FALSE]

for (l in 1:(1+d)) {

for (j in 1:(1+d)) {

M = M + crossprod(t(crossprod(Xi,s[l]\*ri[,l])),t(crossprod(Xi,s[j]\*ri[,j])))

}

}

}

}

}

return(w\*M)

}

NEW CODE

rdrobust\_vce = function(d, s, RX, res, C) {

k = ncol(as.matrix(RX))

M = matrix(0,k,k)

n = length(C)

if (is.null(C)) {

w = 1

if (d==0){

M = crossprod(c(res)\*RX)

}

else {

for (i in 1:(1+d)) {

SS = res[,i]\*res

for (j in 1:(1+d)) {

M = M + crossprod(RX\*(s[i]\*s[j])\*SS[,j],RX)

}

}

}

}

else {

clusters = unique(C)

g = length(clusters)

w=((n-1)/(n-k))\*(g/(g-1))

if (d==0){

for (i in 1:g) {

ind=C==clusters[i]

Xi = RX[ind,,drop=FALSE]

ri = res[ind,,drop=FALSE]

Xr = t(crossprod(Xi,ri))

M = M + crossprod(Xr,Xr)

}

}

else {

for (i in 1:g) {

ind=C==clusters[i]

Xi = RX[ind,,drop=FALSE]

ri = res[ind,,drop=FALSE]

MHolder = matrix(0,1+d,k)

for (l in 1:(1+d)) {

MHolder[l,] = t(crossprod(Xi,s[l]\*ri[,l]))

}

summedvalues = t(colSums(MHolder))

M = M + crossprod(summedvalues,summedvalues)

}

}

}

return(w\*M)

}