function (Y = NULL, Tr, X, Z = X, V = rep(1, length(Y)), estimand = "ATT",

M = 1, BiasAdjust = FALSE, exact = NULL, caliper = NULL,

replace = TRUE, ties = TRUE, CommonSupport = FALSE, Weight = 1,

Weight.matrix = NULL, weights = NULL, Var.calc = 0, sample = FALSE,

restrict = NULL, match.out = NULL, distance.tolerance = 1e-05,

tolerance = sqrt(.Machine$double.eps), version = "standard")

{

BiasAdj <- as.double(BiasAdjust)

sample <- as.double(sample)

if ((BiasAdj != 0) & (BiasAdj != 1)) {

warning("User set 'BiasAdjust' to a non-logical value. Resetting to the default which is FALSE.")

BiasAdj <- 0

}

if (is.null(Y)) {

Y = rep(0, length(Tr))

version <- "fast"

if (BiasAdj) {

warning("'BiasAdjust' set to FALSE because Y is NULL")

BiasAdj <- FALSE

}

}

Y <- as.double(Y)

Tr <- as.double(Tr)

X <- as.matrix(X)

Z <- as.matrix(Z)

V <- as.matrix(V)

orig.nobs <- length(Y)

nobs <- orig.nobs

xvars <- ncol(X)

orig.tr.nobs <- length(Tr)

if (orig.tr.nobs != orig.nobs) {

stop("length(Y) != length(Tr)")

}

if (orig.tr.nobs != nrow(X)) {

stop("length(Tr) != nrow(X)")

}

if (orig.nobs != nrow(X)) {

stop("length(Y) != nrow(X)")

}

if (orig.nobs != nrow(V)) {

stop("length(Y) != nrow(V)")

}

if (orig.nobs != nrow(Z)) {

stop("length(Y) != nrow(Z)")

}

if (is.null(weights)) {

weights <- rep(1, length(Y))

weights.flag <- FALSE

}

else {

weights.flag <- TRUE

weights <- as.double(weights)

if (orig.tr.nobs != length(weights)) {

stop("length(Tr) != length(weights)")

}

}

isna <- sum(is.na(Y)) + sum(is.na(Tr)) + sum(is.na(X)) +

sum(is.na(weights)) + sum(is.na(Z)) + sum(is.na(V))

if (isna != 0) {

stop("Match(): input includes NAs")

return(invisible(NULL))

}

if (sum(Tr != 1 & Tr != 0) > 0) {

stop("Treatment indicator ('Tr') must be a logical variable---i.e., TRUE (1) or FALSE (0)")

}

if (var(Tr) == 0) {

stop("Treatment indicator ('Tr') must contain both treatment and control observations")

}

if (distance.tolerance < 0) {

warning("User set 'distance.tolerance' to less than 0. Resetting to the default which is 0.00001.")

distance.tolerance <- 1e-05

}

if (CommonSupport != 1 & CommonSupport != 0) {

stop("'CommonSupport' must be a logical variable---i.e., TRUE (1) or FALSE (0)")

}

if (CommonSupport == TRUE) {

tr.min <- min(X[Tr == 1, 1])

tr.max <- max(X[Tr == 1, 1])

co.min <- min(X[Tr == 0, 1])

co.max <- max(X[Tr == 0, 1])

if (tr.min >= co.min) {

indx1 <- X[, 1] < (tr.min - distance.tolerance)

}

else {

indx1 <- X[, 1] < (co.min - distance.tolerance)

}

if (co.max <= tr.max) {

indx2 <- X[, 1] > (co.max + distance.tolerance)

}

else {

indx2 <- X[, 1] > (tr.max + distance.tolerance)

}

indx3 <- indx1 == 0 & indx2 == 0

Y <- as.double(Y[indx3])

Tr <- as.double(Tr[indx3])

X <- as.matrix(X[indx3, ])

Z <- as.matrix(Z[indx3, ])

V <- as.matrix(V[indx3, ])

weights <- as.double(weights[indx3])

orig.nobs <- length(Y)

nobs <- orig.nobs

}

if (tolerance < 0) {

warning("User set 'tolerance' to less than 0. Resetting to the default which is 0.00001.")

tolerance <- 1e-05

}

if (M < 1) {

warning("User set 'M' to less than 1. Resetting to the default which is 1.")

M <- 1

}

if (M != round(M)) {

warning("User set 'M' to an illegal value. Resetting to the default which is 1.")

M <- 1

}

if (Var.calc < 0) {

warning("User set 'Var.calc' to less than 0. Resetting to the default which is 0.")

Var.calc <- 0

}

if ((sample != 0) & (sample != 1)) {

warning("User set 'sample' to a non-logical value. Resetting to the default which is FALSE.")

sample <- 0

}

if (Weight != 1 & Weight != 2 & Weight != 3) {

warning("User set 'Weight' to an illegal value. Resetting to the default which is 1.")

Weight <- 1

}

if (version != "fast" & version != "standard" & version !=

"legacy" & version != "Matchby" & version != "MatchbyAI") {

warning("User set 'version' to an illegal value. Resetting to the default which is 'standard'.")

version <- "standard"

}

if (version == "Matchby") {

version <- "fast"

Matchby.call <- TRUE

MatchbyAI <- FALSE

}

else if (version == "MatchbyAI") {

version <- "standard"

Matchby.call <- TRUE

MatchbyAI <- TRUE

}

else {

Matchby.call <- FALSE

MatchbyAI <- FALSE

}

if (Var.calc != 0 & version == "fast") {

warning("Var.calc cannot be estimate when version=='fast'")

Var.calc = 0

}

if (BiasAdj != FALSE & version == "fast") {

warning("Bias Adjustment cannot be estimated when version=='fast'")

BiasAdj = 0

}

if (replace != FALSE & replace != TRUE) {

warning("'replace' must be TRUE or FALSE. Setting to TRUE")

replace <- TRUE

}

if (replace == FALSE) {

ties <- FALSE

version = "fast"

if (version == "legacy")

warning("'version' is set to 'fast' because replace==FALSE")

}

if (ties != FALSE & ties != TRUE) {

warning("'ties' must be TRUE or FALSE. Setting to TRUE")

ties <- TRUE

}

if (ties == FALSE) {

version = "fast"

if (version == "legacy")

warning("'version' is set to 'fast' because ties==FALSE")

if (BiasAdjust == TRUE) {

warning("Bias Adjustment can only be estimated when ties==TRUE and replace=TRUE. Setting BiasAdjust=FALSE")

BiasAdjust <- FALSE

BiasAdj <- 0

}

}

if (!is.null(match.out) & class(match.out) != "Match") {

warning("match.out object not of class 'Match'")

return(invisible(NULL))

}

ccc <- tolerance

cdd <- distance.tolerance

orig.treated.nobs <- sum(Tr == 1)

orig.control.nobs <- sum(Tr == 0)

orig.wnobs <- sum(weights)

orig.weighted.treated.nobs <- sum(weights[Tr == 1])

orig.weighted.control.nobs <- sum(weights[Tr == 0])

weights.orig <- as.matrix(weights)

zvars <- ncol(Z)

estimand.orig <- estimand

if (estimand == "ATT") {

estimand <- 0

if (BiasAdj == 1 & orig.treated.nobs < zvars) {

warning("Fewer treated obs than variables in 'Z': BiasAdjust set to FALSE")

BiasAdj = 0

}

}

else if (estimand == "ATE") {

estimand <- 1

if (BiasAdj == 1 & orig.nobs < zvars) {

warning("Fewer obs than variables in 'Z': BiasAdjust set to FALSE")

BiasAdj = 0

}

}

else if (estimand == "ATC") {

estimand <- 2

if (BiasAdj == 1 & orig.control.nobs < zvars) {

warning("Fewer control obs than variables in 'Z': BiasAdjust set to FALSE")

BiasAdj = 0

}

}

else {

estimand <- 0

warning("User set 'estimand' to an illegal value. Resetting to the default which is 'ATT'")

}

if (!is.null(Weight.matrix)) {

if (class(Weight.matrix) == "GenMatch") {

Weight.matrix = Weight.matrix$Weight.matrix

}

if (Weight == 2) {

warning("User supplied 'Weight.matrix' is being used even though 'Weight' is not set equal to 3")

}

Weight <- 3

}

else {

Weight.matrix <- dim(X)[2]

}

if (Var.calc > orig.weighted.treated.nobs) {

warning("'Var.calc' > the number of treated obs: 'Var.calc' reset to ",

orig.weighted.treated.nobs, immediate. = Matchby.call)

Var.calc <- orig.weighted.treated.nobs

}

if (Var.calc > orig.weighted.control.nobs) {

warning("'Var.calc' > the number of control obs: 'Var.calc' reset to ",

orig.weighted.control.nobs, immediate. = Matchby.call)

Var.calc <- orig.weighted.control.nobs

}

if (orig.nobs > 20000 & version != "fast" & !Matchby.call) {

warning("The version='fast' option is recommended for large datasets if speed is desired. For additional speed, you may also consider using the ties=FALSE option.",

immediate. = TRUE)

}

if (!is.null(restrict)) {

if (!is.matrix(restrict))

stop("'restrict' must be a matrix of restricted observations rows and three columns: c(i,j restriction)")

if (ncol(restrict) != 3)

stop("'restrict' must be a matrix of restricted observations rows and three columns: c(i,j restriction)")

}

if (!is.null(exact)) {

exact = as.vector(exact)

nexacts = length(exact)

if ((nexacts > 1) & (nexacts != xvars)) {

warning("length of exact != ncol(X). Ignoring exact option")

exact <- NULL

}

else if (nexacts == 1 & (xvars > 1)) {

exact <- rep(exact, xvars)

}

}

if (!is.null(caliper)) {

caliper = as.vector(caliper)

ncalipers = length(caliper)

if ((ncalipers > 1) & (ncalipers != xvars)) {

warning("length of caliper != ncol(X). Ignoring caliper option")

caliper <- NULL

}

else if (ncalipers == 1 & (xvars > 1)) {

caliper <- rep(caliper, xvars)

}

}

if (!is.null(caliper)) {

ecaliper <- vector(mode = "numeric", length = xvars)

sweights <- sum(weights.orig)

for (i in 1:xvars) {

meanX <- sum(X[, i] \* weights.orig)/sweights

sdX <- sqrt(sum((X[, i] - meanX)^2)/sweights)

ecaliper[i] <- caliper[i] \* sdX

}

}

else {

ecaliper <- NULL

}

if (!is.null(exact)) {

if (is.null(caliper)) {

max.diff <- abs(max(X) - min(X) + tolerance \* 100)

ecaliper <- matrix(max.diff, nrow = xvars, ncol = 1)

}

for (i in 1:xvars) {

if (exact[i])

ecaliper[i] <- tolerance

}

}

if (replace == FALSE) {

orig.weighted.control.nobs <- sum(weights[Tr != 1])

if (estimand == 0) {

if (orig.weighted.treated.nobs > orig.weighted.control.nobs) {

warning("replace==FALSE, but there are more (weighted) treated obs than control obs. Some treated obs will not be matched. You may want to estimate ATC instead.")

}

}

else if (estimand == 1) {

if (orig.weighted.treated.nobs > orig.weighted.control.nobs) {

warning("replace==FALSE, but there are more (weighted) treated obs than control obs. Some treated obs will not be matched. You may want to estimate ATC instead.")

}

if (orig.weighted.treated.nobs < orig.weighted.control.nobs) {

warning("replace==FALSE, but there are more (weighted) control obs than treated obs. Some control obs will not be matched. You may want to estimate ATT instead.")

}

}

else {

if (orig.weighted.treated.nobs < orig.weighted.control.nobs) {

warning("replace==FALSE, but there are more (weighted) control obs than treated obs. Some obs will be dropped. You may want to estimate ATC instead")

}

}

if (is.null(restrict)) {

restrict <- t(as.matrix(c(0, 0, 0)))

}

if (version != "fast" & version != "standard") {

warning("reverting to 'standard' version because replace=FALSE")

version = "standard"

}

}

if (version == "fast" | version == "standard") {

if (!is.null(match.out)) {

ret <- RmatchLoop(Y = Y, Tr = Tr, X = X, Z = Z, V = V,

All = estimand, M = M, BiasAdj = BiasAdj, Weight = Weight,

Weight.matrix = Weight.matrix, Var.calc = Var.calc,

weight = weights, SAMPLE = sample, ccc = ccc,

cdd = cdd, ecaliper = ecaliper, exact = exact,

caliper = caliper, restrict = restrict, MatchLoopC.indx = match.out$MatchLoopC,

weights.flag = weights.flag, replace = replace,

ties = ties, version = version, MatchbyAI = MatchbyAI)

}

else {

ret <- RmatchLoop(Y = Y, Tr = Tr, X = X, Z = Z, V = V,

All = estimand, M = M, BiasAdj = BiasAdj, Weight = Weight,

Weight.matrix = Weight.matrix, Var.calc = Var.calc,

weight = weights, SAMPLE = sample, ccc = ccc,

cdd = cdd, ecaliper = ecaliper, exact = exact,

caliper = caliper, restrict = restrict, weights.flag = weights.flag,

replace = replace, ties = ties, version = version,

MatchbyAI = MatchbyAI)

}

}

else {

ret <- Rmatch(Y = Y, Tr = Tr, X = X, Z = Z, V = V, All = estimand,

M = M, BiasAdj = BiasAdj, Weight = Weight, Weight.matrix = Weight.matrix,

Var.calc = Var.calc, weight = weights, SAMPLE = sample,

ccc = ccc, cdd = cdd, ecaliper = ecaliper, restrict = restrict)

}

if (is.null(ret$est)) {

if (!Matchby.call) {

if (ret$valid < 1) {

if (ret$sum.caliper.drops > 0) {

warning("'Match' object contains no valid matches (probably because of the caliper or the exact option).")

}

else {

warning("'Match' object contains no valid matches")

}

}

else {

if (ret$sum.caliper.drops > 0) {

warning("'Match' object contains only 1 valid match (probably because of the caliper or the exact option).")

}

else {

warning("'Match' object contains only one valid match")

}

}

}

z <- NA

class(z) <- "Match"

return(z)

}

indx <- cbind(ret$art.data[, 1], ret$art.data[, 2], ret$W)

index.treated <- indx[, 1]

index.control <- indx[, 2]

weights <- indx[, 3]

sum.caliper.drops <- ret$sum.caliper.drops

indx <- as.matrix(cbind(index.treated, index.control))

if (estimand == 0) {

index.treated <- indx[, 1]

index.control <- indx[, 2]

}

else if (estimand == 1) {

tmp.index.treated <- indx[, 1]

tmp.index.control <- indx[, 2]

tl <- length(tmp.index.treated)

index.treated <- vector(length = tl, mode = "numeric")

index.control <- vector(length = tl, mode = "numeric")

trt <- Tr[tmp.index.treated] == 1

for (i in 1:tl) {

if (trt[i]) {

index.treated[i] <- tmp.index.treated[i]

index.control[i] <- tmp.index.control[i]

}

else {

index.treated[i] <- tmp.index.control[i]

index.control[i] <- tmp.index.treated[i]

}

}

}

else if (estimand == 2) {

index.treated <- indx[, 2]

index.control <- indx[, 1]

}

mdata <- list()

mdata$Y <- c(Y[index.treated], Y[index.control])

mdata$Tr <- c(Tr[index.treated], Tr[index.control])

mdata$X <- rbind(X[index.treated, ], X[index.control, ])

mdata$orig.weighted.treated.nobs <- orig.weighted.treated.nobs

mest <- sum((Y[index.treated] - Y[index.control]) \* weights)/sum(weights)

v1 <- Y[index.treated] - Y[index.control]

varest <- sum(((v1 - mest)^2) \* weights)/(sum(weights) \*

sum(weights))

se.standard <- sqrt(varest)

wnobs <- sum(weights)

if (estimand == 0) {

actual.drops <- orig.weighted.treated.nobs - wnobs

}

else if (estimand == 1) {

actual.drops <- orig.wnobs - wnobs

}

else {

actual.drops <- (orig.wnobs - orig.weighted.treated.nobs) -

wnobs

}

index.dropped <- NULL

if (sum.caliper.drops > 0) {

if (estimand.orig == "ATT") {

matched.index <- which(Tr == 1)

matched <- !(matched.index %in% index.treated)

}

else if (estimand.orig == "ATC") {

matched.index <- which(Tr == 0)

matched <- !(matched.index %in% index.control)

}

else if (estimand.orig == "ATE") {

matched.index <- 1:length(Tr)

matched <- !(matched.index %in% c(index.treated,

index.control))

}

index.dropped <- matched.index[matched]

}

z <- list(est = ret$est, se = ret$se, est.noadj = mest, se.standard = se.standard,

se.cond = ret$se.cond, mdata = mdata, index.treated = index.treated,

index.control = index.control, index.dropped = index.dropped,

weights = weights, orig.nobs = orig.nobs, orig.wnobs = orig.wnobs,

orig.treated.nobs = orig.treated.nobs, nobs = nobs, wnobs = wnobs,

caliper = caliper, ecaliper = ecaliper, exact = exact,

ndrops = actual.drops, ndrops.matches = sum.caliper.drops,

MatchLoopC = ret$MatchLoopC, version = version, estimand = estimand.orig)

if (MatchbyAI) {

z$YCAUS <- ret$YCAUS

z$ZCAUS <- ret$ZCAUS

z$Kcount <- ret$Kcount

z$KKcount <- ret$KKcount

z$Sigs <- ret$Sigs

}

class(z) <- "Match"

return(z)

}