jackknife\_functions.R

zsmith

Thu Nov 10 08:58:41 2016

#==============================================================================  
#  
# Jackknife Functions  
#  
#==============================================================================  
#==============================================================================

Delete-d Jackknife

@param metrics.df = a data frame containing the raw metric values. Only the metrics selected for the final index are included. @param runs = number of jackknife resampling events. @param keep = the proportion of the reference population that should be kept. @param bioregion = the bioregion of interest @param m.c = a data frame containing all of the metrics and thier associated metric category. @param Fam = If TRUE, all family metrics are grouped and scored together prior to the final scoring procedure. If FALSE, family metrics are grouped with composition metrics. @param Master = a data frame containing taxonomic traits and attributes. @param zero.null = If TRUE, all zeros are converted to NA for all metrics with > 5% of Reference values equal to zero. If FALSE, metric values are unaltered. @param redund.df = IF TRUE, metrics are evaluated for redundancy. If FALSE, a redundancy analysis is not preformed. @param method = specify the method being preformed. FIGURE OUT A BETTER WAY!!!! @return Recomputes the final IBI threshold after removing a specified percentage of the data. @export

jack.calc <- function(metrics.df, runs, keep, bioregion, m.c, Fam = FALSE, Master = master,  
 zero.null = TRUE, metric\_types = TRUE, redund.df, method = ibi.method){  
   
 ref.df <- metrics.df[metrics.df$CATEGORY %in% "REF", ]  
 not\_ref.df <- metrics.df[!metrics.df$CATEGORY %in% "REF", ]  
 deg.df <- metrics.df[metrics.df$CATEGORY %in% "SEV", ]  
 ref\_deg <- rbind(ref.df, deg.df)  
   
 choose.int <- function(x, n, k) {  
 if(n <= k) return(rep(TRUE, k))  
 u <- choose(n-1, k-1)  
 pick <- x < u  
 if (pick) y <- choose.int(x, n-1, k-1) else y <- choose.int(x-u, n-1, k)  
 return(c(pick, y))  
 }  
   
 #REFERENCE subsample  
 n <- nrow(ref.df)  
 k <- floor(nrow(ref.df) \* keep)  
 #If n is greater than 1000 than R cannot seem to handle the number of possible outcomes  
 # Seting n = 1000 argueably provides an unbiased random sample  
 # k is reduced relative to the size of "keep"  
 if (n > 1000){  
 n <- 1000  
 k <- 1000 \* keep  
 }  
 bf <- gmp::chooseZ(n, k)  
 rand.samp <- gmp::urand.bigz(nb = runs, size = (log(bf) / log(2)), seed = 0)  
 ref.sub.samp <- data.frame(sapply(as.numeric(as.character(rand.samp)), choose.int, n = n, k = k))  
 #DEGRADED subsample  
 n <- nrow(deg.df)  
 k <- floor(nrow(deg.df) \* keep)  
 #If n is greater than 1000 than R cannot seem to handle the number of possible outcomes  
 # Seting n = 1000 argueably provides an unbiased random sample  
 # k is reduced relative to the size of "keep"  
 if (n > 1000){  
 n <- 1000  
 k <- 1000 \* keep  
 }  
 bf <- gmp::chooseZ(n, k)  
 rand.samp <- gmp::urand.bigz(nb = runs, size = (log(bf) / log(2)), seed = 0)  
 deg.sub.samp <- data.frame(sapply(as.numeric(as.character(rand.samp)), choose.int, n = n, k = k))  
   
 sub.samp <- rbind(ref.sub.samp, deg.sub.samp)  
   
 #x <- sub.samp [, 1]  
 jack.sim <- apply(sub.samp , 2, function(x){  
 all\_sites.df <- ref\_deg[unlist(x), ]  
 #all\_sites.df <- rbind(ref, not\_ref.df)  
   
 ms.test <- metrics\_summary2(all\_sites.df, bioregion)  
 os.test <- old\_scoring3(all\_sites.df, bioregion, zero\_null = zero.null)  
 #bd.test <- best.distribution2(all\_sites.df, ms.test, os.test, m.c)  
 #solid.m <- solid\_metrics(new.df, os2, ms, bd2[1, "THRESH"], m.c)  
 if(metric\_types == TRUE){  
 final.metrics.vec <- unlist(list("EVENT\_ID", "CATEGORY", "STATION\_ID",  
 "SAMPLE\_NUMBER", "AGENCY\_CODE", "DATE",  
 as.character(ms.test$METRICS)))  
   
 #tt <- ms[ms$METRICS %in% final.metrics.vec, ]  
 scored.test <- scoring2(metrics.df = all\_sites.df [, names(all\_sites.df ) %in% final.metrics.vec],  
 scores = os.test[, names(os.test) %in% final.metrics.vec],  
 ms = ms.test,  
 m.c, Family = Fam, master,   
 redund = redund.df,  
 method)  
   
 final.df <- bde(scored.test, bioregion)  
 }else{  
 if(ncol(os.test) > 7){  
 os.test[, 7:ncol(os.test)] <- apply(os.test[, 7:ncol(os.test)], 2, function(x) as.numeric(as.character(x)))  
 os.test$FINAL\_SCORE <- apply(os.test[, 7:ncol(os.test)], 1, mean, na.rm = TRUE)  
 }else{  
 os.test[, 7] <- as.numeric(as.character(os.test[, 7]))  
 os.test$FINAL\_SCORE <- os.test[, 7]  
 }  
  
 final.df <- bde(os.test, bioregion)  
 }  
   
 return(final.df[1, ])  
 } )  
   
 return(jack.sim)  
}  
  
#==============================================================================

Delete-d Jackknife Validation

@param metrics.df = a data frame containing the raw metric values. Only the metrics selected for the final index are included. @param runs = number of jackknife resampling events. @param keep = the proportion of the reference population that should be kept. @param bioregion = the bioregion of interest @param m.c = a data frame containing all of the metrics and thier associated metric category. @param Fam = If TRUE, all family metrics are grouped and scored together prior to the final scoring procedure. If FALSE, family metrics are grouped with composition metrics. @param Master = a data frame containing taxonomic traits and attributes. @param zero.null = If TRUE, all zeros are converted to NA for all metrics with > 5% of Reference values equal to zero. If FALSE, metric values are unaltered. @param redund.df = IF TRUE, metrics are evaluated for redundancy. If FALSE, a redundancy analysis is not preformed. @param method = specify the method being preformed. FIGURE OUT A BETTER WAY!!!! @return Recomputes the final IBI threshold after removing a specified percentage of the data. @export

jack.val <- function(metrics.df, runs, keep, bioregion, m.c, Fam = FALSE, Master = master,  
 zero.null = TRUE, metric\_types = TRUE, redund.df, method = ibi.method){  
   
 ref.df <- metrics.df[metrics.df$CATEGORY %in% "REF", ]  
 not\_ref.df <- metrics.df[!metrics.df$CATEGORY %in% "REF", ]  
 deg.df <- metrics.df[metrics.df$CATEGORY %in% "SEV", ]  
 ref\_deg <- rbind(ref.df, deg.df)  
   
   
 choose.int <- function(x, n, k) {  
 if(n <= k) return(rep(TRUE, k))  
 u <- choose(n-1, k-1)  
 pick <- x < u  
 if (pick) y <- choose.int(x, n-1, k-1) else y <- choose.int(x-u, n-1, k)  
 return(c(pick, y))  
 }  
   
 #REFERENCE subsample  
 n <- nrow(ref.df)  
 k <- floor(nrow(ref.df) \* keep)  
 if (n > 1000){  
 n <- 1000  
 k <- 1000 \* keep  
 }  
 bf <- gmp::chooseZ(n, k)  
 rand.samp <- gmp::urand.bigz(nb = runs, size = (log(bf) / log(2)), seed = 0)  
 ref.sub.samp <- data.frame(sapply(as.numeric(as.character(rand.samp)), choose.int, n = n, k = k))  
 #DEGRADED subsample  
 n <- nrow(deg.df)  
 k <- floor(nrow(deg.df) \* keep)  
 if (n > 1000){  
 n <- 1000  
 k <- 1000 \* keep  
 }  
 bf <- gmp::chooseZ(n, k)  
   
 rand.samp <- gmp::urand.bigz(nb = runs, size = (log(bf) / log(2)), seed = 0)  
 deg.sub.samp <- data.frame(sapply(as.numeric(as.character(rand.samp)), choose.int, n = n, k = k))  
   
 train.samp <- rbind(ref.sub.samp, deg.sub.samp)  
   
   
   
   
 #x <- train.samp [, 68]  
 #train.samp2 <- train.samp [,66:68 ]  
 jack.sim <- apply(train.samp , 2, function(x){  
   
 train\_sites.df <- ref\_deg[unlist(x), ]  
   
   
 #all\_sites.df <- rbind(ref, not\_ref.df)  
   
 ms.train <- metrics\_summary2(train\_sites.df, bioregion)  
 os.train <- old\_scoring3(train\_sites.df, bioregion, zero\_null = zero.null)  
 #bd.test <- best.distribution2(all\_sites.df, ms.test, os.test, m.c)  
 #solid.m <- solid\_metrics(new.df, os2, ms, bd2[1, "THRESH"], m.c)  
 if(metric\_types == TRUE){  
 final.metrics.vec <- unlist(list("EVENT\_ID", "CATEGORY", "STATION\_ID",  
 "SAMPLE\_NUMBER", "AGENCY\_CODE", "DATE",  
 as.character(ms.train$METRICS)))  
 scored.train <- scoring2(metrics.df = train\_sites.df[, names(train\_sites.df) %in% final.metrics.vec],  
 scores = os.train[, names(os.train) %in% final.metrics.vec],  
 ms = ms.train, m.c,  
 Family = Fam,  
 master = Master,  
 redund = redund.df,  
 method)  
   
 bde.train <- bde(scored.train, bioregion)  
 }else{  
 if(ncol(os.train) > 7){  
 os.train[, 7:ncol(os.train)] <- apply(os.train[, 7:ncol(os.train)], 2, function(x) as.numeric(as.character(x)))  
 os.train$FINAL\_SCORE <- apply(os.train[, 7:ncol(os.train)], 1, mean, na.rm = TRUE)  
 }else{  
 os.train[, 7] <- as.numeric(as.character(os.train[, 7]))  
 os.train$FINAL\_SCORE <- os.train[, 7]  
 }  
 bde.train <- bde(os.train, bioregion)  
 }  
   
   
 val.samp <- ifelse(x == TRUE, FALSE, TRUE)  
 val\_sites.df <- ref\_deg[unlist(val.samp), ]  
 os.val <- old\_scoring3(val\_sites.df, bioregion, zero\_null = zero.null, metric.summary = ms.train)  
 #tt <- ms[ms$METRICS %in% final.metrics.vec, ]  
 if(metric\_types == TRUE){  
 scored.val <- scoring2(metrics.df =val\_sites.df[, names(val\_sites.df) %in% final.metrics.vec],  
 scores = os.val[, names(os.val) %in% final.metrics.vec],  
 ms = ms.train,  
 m.c,  
 Family = Fam,  
 master = Master,  
 redund = redund.df,   
 method)  
 }else{  
 if(ncol(os.val) > 7){  
 os.val[, 7:ncol(os.val)] <- apply(os.val[, 7:ncol(os.val)], 2, function(x) as.numeric(as.character(x)))  
 os.val$FINAL\_SCORE <- apply(os.val[, 7:ncol(os.val)], 1, mean, na.rm = TRUE)  
 }else{  
 os.val[, 7] <- as.numeric(as.character(os.val[, 7]))  
 os.val$FINAL\_SCORE <- os.val[, 7]  
 }  
  
 scored.val <- os.val  
 }  
   
 scored.val.ref <- scored.val[scored.val$CATEGORY %in% "REF", ]  
 ref.vec <- ifelse(scored.val.ref$FINAL\_SCORE \* 100 >= bde.train$THRESHOLD[1], 1, 0)   
 pct.ref <- (sum(ref.vec) / length(ref.vec)) \* 100  
 scored.val.deg <- scored.val[scored.val$CATEGORY %in% "SEV", ]  
 deg.vec <- ifelse(scored.val.deg$FINAL\_SCORE \* 100 < bde.train$THRESHOLD[1], 1, 0)  
 pct.deg <- (sum(deg.vec) / length(deg.vec)) \* 100  
   
 final.value <- (pct.ref + pct.deg) / 2  
 return(final.value)  
 } )  
   
 jack <- data.frame(BIOREGION = bioregion)  
   
 jack$MEAN\_SIM\_VALUE <- mean(jack.sim)  
   
 n.ref <- nrow(ref.df)  
 n.deg <- nrow(deg.df)  
 n <- n.ref + n.deg  
 d.ref <- round(n.ref \* 0.25, 0)  
 d.deg <- round(n.deg \* 0.25, 0)  
 d <- d.ref + d.deg  
 N <- length(jack.sim)  
   
 resid.jack <- (jack.sim) - jack$MEAN\_SIM\_VALUE  
 jack$MSE <- (sum(resid.jack ^ 2)) / length(jack.sim)  
 jack$RMSE <- sqrt(jack$MSE)  
 jack$BIAS <- sum(jack$MEAN\_SIM\_VALUE - jack.sim) / length(jack.sim)  
 jack$'95%.CI.ERROR' <- qnorm(0.975) \* jack$RMSE / sqrt(N)  
 jack$LOW.CI <- jack$MEAN\_SIM\_VALUE - jack$'95%.CI.ERROR'  
 jack$UP.CI <- jack$MEAN\_SIM\_VALUE + jack$'95%.CI.ERROR'  
   
 #sd(resid.jack) + jack$BIAS  
 #jack$SE + jack$BIAS  
 #jack$SE <- sqrt(((n - d) / (d \* length(jack.sim))) \* (sum(resid.jack ^ 2)))  
 #jack$SE2 <- (sum(resid.jack ^ 2)) / sqrt(length(jack.sim))  
 #jack$SD <- jack$SE2 \* sqrt(length(jack.sim))  
 #jack$SD <- sd(resid.jack)  
 #jack$SE <- sqrt((sum(resid.jack ^ 2)) / length(jack.sim))  
 #jack$SE <- jack$SD / sqrt(length(jack.sim))  
 #jack$SEM <- sd(resid.jack) / sqrt(length(resid.jack))  
 #jack$MSE2 <- (sum(resid.jack ^ 2)) / n  
 #jack$RMSE2 <- sqrt(jack$MSE2)  
   
 return(jack)  
}