# like many R scripts, this was written to be run in pieces. Sections are numbered, e.g. (1), to make it easier to see which lines of code might be run together. Everything starting with "#" is a comment.
# If the data set is large (hundreds of cultivars), (4) may take a long time (e.g. a day); in this case select a subset to make sure everything works first, e.g. 10 trials, 100 cultivars
# R code written by Matt Kramer: email matt.kramer@ars.usda.gov last edit: 2012-02-15

# (1)
data1 <- read.csv("DM.csv") # note the format for the data: the first column holds accession names

# most data fields are blank since this accession was tested only twice
# the first record gets read in as a header (i.e. column names)
# in the first line, field1 = title for accession names, other fields are trial names

## AccessionName,Trial1,Trial2,Trial3, # ... not showing fields 4+
## Acc1,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,4.08,,,,,,,,,,4,,,,,,,,,,,,,,,,
## Acc2,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,3.58,,,,,,,,,,4,,,,,,,,,,,,,,,,
## Acc3,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,3.67,,,,,,,,,,4,,,,,,,,,,,,,,,,
## Acc4,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,4,,,,,,,,,,,,,,,,
## Acc5,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,0,,,,,,,,,0

# how many trials per accession?
for (i in 1:nrow(data1)) {
  cat (sprintf("%-40s %5d
", data1[i,1], sum(!is.na(data1[i,]))))
}
data2 <- data1   # keep a copy of data1, data2 will have some rows and columns eliminated

#------------------------------------------------------------------#
# try to eliminate problem rows and columns
# (eliminate accessions/trials that might be problematic)
# iterate this part of the code a few times -----------------------
# (2)

br <- numeric (0)
mna <- 3   # minimum number of trials to keep an accession in the data set

for (i in 1:dim(data2)[1]) {
  # cat (i, "\n")
  tl <- !is.na(data2[i,])
  gd <- sum (tl)
  flag1 <- TRUE
  if (sum(tl) < mna) {
    br <- c (br, i)
    flag1 <- FALSE  # no need to continue, too few trials for this accession
  }
}

# code below gets rid of accessions with few trials and is the maximum or minimum in a trial
if ((flag1 == TRUE) && (sum(tl) < (mna + 1))) { # first column holds names so need to adjust for that
  j <- (1:dim(data2)[2])[tl]
  if (length (j) > 1) {
    j <- j[2]
    tl.g <- !is.na(data2 [,j])
  }
if ((data2 [i, j] == max (data2 [tl.g,j])) &
    (sum (match (data2 [tl.g, j], data2 [i,j], nomatch=0)) == 1)) {
    # is also unique
    br <- c (br, i)
    cat (sprintf (" ------> Accession %3d (%s) has only one rep. and is the maximum
    \n", i, data2 [i, 1]))
} else
    if ((data2 [i, j] == min (data2 [tl.g,j])) &
        (sum (match (data2 [tl.g, j], data2 [i,j], nomatch=0)) == 1)) {
        br <- c (br, i)
        cat (sprintf (" ------> Accession %3d (%s) has only one rep. and is the minimum
        \n", i, data2 [i, 1]))
    } else {
        # no data at all, drop Accession
        br <- c (br, i)
        cat (sprintf (" ------> Accession %3d (%s) has no reps.\n", i, data2 [i, 1]))
    }
} # matches if (sum(tl) < (mna + 1)) {
# replace data2 with a version without the bad accessions (i.e. eliminating some rows)
if (length (br) > 0) {
    acc <- 1:dim(data2)[1]
    acc <- acc[!(acc %in% br)]
    data2 <- data2[acc, ]
} # get rid of bad trials
uvt <- 3  # must have at least this number of unique values
bc <- numeric (0)
for (i in 2:dim(data2)[2]) {
    tl <- !is.na(data2[,i])
    gd <- sum (tl)
    # cat (sprintf ("%3d  %-25s %3d\n", i, dimnames(data2)[[2]][i], gd))
    if (sum(tl) < 3) { # get rid of trials with less than 3 accessions
        bc <- c(bc, i)
        cat (sprintf (" ------> Trial %3d (%s) has less than 3 reps.\n", i, dimnames(data2)[[2]][i]))
    } else {
        lun <- length (unique (data2[tl,i]))
        if (lun < uvt) { # less than uvt unique values, get rid of binary columns
            bc <- c(bc, i)
            cat (sprintf (" ------> Trial %3d (%s) has less than %1d unique values\n", i, dimnames(data2)[[2]][i], uvt))
        }
    }
} # replace data2 with a version that eliminates bad trials
if (length (bc) > 0) {
    blocks <- 1:dim(data2)[2]
    blocks <- blocks[!(blocks %in% bc)]
    data2 <- data2[, blocks]
dataorig2 <- data2 [,2:ncol(data2)]  # after going through the algorithm that removed "bad" accessions
# get rid of first column

# correlation between data columns (trials)
# create graphic
postscript ("pairs2.ps", horizontal=FALSE)
par(mfrow = c(4,3))
for (i in 1:(ncol(dataorig2) - 1)) {
  tl <- !is.na(dataorig2[,i]) & !is.na(dataorig2[,ncol(dataorig2)])
  td1 <- dataorig2 [tl,i]
  td2 <- dataorig2 [tl,ncol(dataorig2)]
  if (sum(tl) > 1) {
    cor1 <- sprintf("%6.3f", cor(td1,td2))
    plot (td1, td2, main = paste (i, "vs", ncol(dataorig2), " corr. =", cor1, " pairs =", sum(tl)))
  }
}
dev.off()
# produce table
sink("corr2.txt")
cat ("trial1 trial2 numpairs correl.
")
for (i in 1:(ncol(dataorig2) - 1)) {
  for (j in (i + 1):ncol(dataorig2)) {
    tl <- !is.na(dataorig2[,] i) & !is.na(dataorig2[,] j)
    td1 <- dataorig2 [tl,i]
    td2 <- dataorig2 [tl,j]
    if (sum(tl) > 1) {
      cor1 <- sprintf("%6.3f", cor(td1,td2))
      cat (sprintf("%8d %8d %8d %6s\n", i, j, sum(tl), cor1))
    } else {
      cor1 <- "NA"
      cat (sprintf("%8d %8d %8d %6s\n", i, j, sum(tl), cor1))
    }
  }
}sink()
# how many were positive corr, how many negative?
corr2 <- read.table("corr2.txt", header=TRUE)
frac0 <- 0
for (i in 1:ncol(dataorig2)) {
  tl <- (corr2$trial1 == i) | (corr2$trial2 == i)
  cat (sprintf("%3d %3d %3d\n", i, sum(corr2$correl[tl] > 0, na.rm=TRUE),
           sum(corr2$correl[tl] < 0, na.rm=TRUE)))
  if (sum(corr2$correl[tl] > 0, na.rm=TRUE)) {
    frac1 <- sum(corr2$correl[tl] < 0, na.rm=TRUE) / sum(corr2$correl[tl] > 0, na.rm=TRUE)
    if (frac1 > frac0) {
      frac0 <- frac1
      highcol <- i
    }
  } else {
    cat (" Number of positive correlations is zero\n")
High fractions = frac0, High column = highcol

# based on this, additional columns (trials) may need to be removed, e.g.
# to remove col 49
## dataorig2 <- dataorig2[,c(1:48,50:ncol(dataorig2))]
# after this, check that there are enough accessions in each column
# (4)

#-------------------------------iterations-----------------------------------#
#-------- this is where the trials get merged and cultivars are given a score -------#
# the starting data matrix is dataorig2
ntotiter <- 30  # number of total iterations (i.e. how many sets of scores will be
calculated)
res1 <- matrix (NA, nrow(dataorig2), ntotiter)       # will hold scores
mv1vec <- numeric(ntotiter)    # recording of how close the columns are to each other in
each overall iteration
cnt2vec <- numeric(ntotiter)   # recording of how many loops it took to get them this
close
oitl <- 2000   # outer iteration limit in while {}
iiitl <- 1000   # inner iteration limit (to find two matching columns)

for (k in 1:ntotiter) {  # total iterations
  samp1 <- 1000           # how many samples will be carried through the calculations
datao <- dataorig2      # datao = data to work with, will become old data
datan <- matrix (NA, dim(datao)[[1]], samp1)  # datan = new data matrix after combining
trials or projected values
mv1 <- 10   # start with a large value to get the loop started
cnt2 <- 0

# loop, continue until the columns in datao are similar enough from all the merging
while ((mv1 > 0.025) & (cnt2 < oitl)) {   # what is the best value for mv1?  Should be
related to sample size.
  cnt2 <- cnt2 + 1
  for (j in 1:samp1) {  # for j times, grab two random columns from datao
    es <- 0
    cnt1 <- 0
    while ((es == 0) && (cnt1 < iiitl)) {
      col2 <- sample (1:dim(datao)[[2]], 2)         # columns that will be used
      # cat ("cnt1 =", cnt1, "  col2 =", col2, "\n")
      tl <- !is.na(datao[,col2[1]]) & !is.na(datao[,col2[2]])
      td1 <- datao [tl,col2[1]]
      td2 <- datao [tl,col2[2]]
      # if (sum (tl) >= 2) {
      if (sum (tl) >= 3) {   # now need at least 3 in common and want each column to
        have different numbers
        if ((var(td1) > 1e-12) & (var(td2) > 1e-12)) {   # are these small enough
          numbers?  Don't want var = 0.
          es <- 1
        } else {
          cnt1 <- cnt1 + 1
        }
      } else {
        cnt1 <- cnt1 + 1
      }
    }
  }
}


if (es == 0) { stop("Could not find two columns to use in", iitl, "tries
") }
d1 <- datao [,col2[1]]
d2 <- datao [,col2[2]]
pr1 <- prcomp(cbind(td1,td2), retx = TRUE, center = TRUE, scale. = TRUE)
lm1 <- lm (pr1$x[,1] ~ td1)
if (lm1$coefficients[2] < 0) {pr1$x[,1] <- -1 * pr1$x[,1]}  
datan[tl,j] <- pr1$x[,1]      # the two columns merged
tl1 <- !tl & !is.na(d1)
if (sum(tl1) > 0) {  # put in additional d1 values that were not common to both columns
    idx1 <- (1:length(tl1))[tl1]
lm1 <- lm (pr1$x[,1] ~ td1)
}
tl1 <- !tl & !is.na(d2)
if (sum(tl1) > 0) {  # put in additional d2 values that were not common to both columns
    idx1 <- (1:length(tl1))[tl1]
lm1 <- lm (pr1$x[,1] ~ td1)
}
# for j --- finished with the two columns
# replace values in datao with values in datan, then can start process over
if (dim(datao)[[2]] == dim(datan)[[2]]) {
    gc1 <- !is.na(datan)
datao[gc1] <- datan[gc1]
} else {
    datao <- datan
}
var1 <- apply(datao,1,var, na.rm = TRUE)
mv1 <- mean (var1, na.rm = TRUE)
## cat (sprintf ("cnt2 = %5d, k = %5d, mv1 = %12.6f\n",  #
##             cnt2, k, mv1))                          # error checking
## flush.console()
}  # matches while ((mv1 > 0.025) & (cnt2 < oitl)) {  # this will loop until columns in datan are very similar
# criteria satisfied, columns in datao=datan are very similar and can take their averages
# for the scores
# (unless there were too many iterations [if (cnt2 == oitl)], in which case need to redo the loop)
res1 [,k] <- apply(datao,1,mean, na.rm = TRUE)
mv1vec[k] <- mv1
cnt2vec[k] <- cnt2
cat (sprintf ("Finished iteration %2d of %2d total iterations, ", k, ntotiter))
cat (sprintf ("number of iterations = %4d, criterion = %8.4f \n",
            cnt2, mv1))
if (cnt2 == oitl) {cat (sprintf ("\n--> hit upper iteration limit --\n\n"))}
if (cnt2 == oitl) {cnt2 <- cnt2 - 1}  # do this iteration again --- better to start over than keep going
}  # matches "for k ...") this is the overall loop
cat ("\nAll Finished\n\n")
# (5)
# check to make sure all columns of res1 are ordered in the same direction (they usually
# aren't)
# (recall that the sign from a PCA is arbitrary)
# create a file of plots of each of the iteration results against the last one
[ncol(res1)]
postscript("plot1.ps", horizontal=FALSE)
par(mfrow=c(4,3))
for (i in 1:(ncol(res1) - 1)) {
  plot (res1[,i], res1[,ncol(res1)], ylab=paste("iteration", ncol(res1)),
    xlab=paste("iteration", i), bty="l")
  abline(0,1, col="blue")
}
dev.off()
# reverse the sign on scores which don't the sign of the last set of scores
# (look at plot1.ps, you may want to choose a different column)
for (i in 1:29) {
  if (cor(res1[,i], res1[,30]) < 0) { res1[,i] <- -1 * res1[,i] }
}
res1.20120202 <- res1  # change this (20120202) to today's date (here and elsewhere),
# this keeps a copy available in the workspace
# calculate ranks for the accessions for each column of rows
rank1 <- matrix(NA, nrow(res1), ncol(res1))
for (k in 1:dim(res1)[[2]]) {
  rank1 [,k] <- rank (res1 [,k], na.last="keep")
}
# averages and standard errors on scores
mn1 <- apply(res1,1,mean, na.rm = TRUE)
mn1.s <- mn1     # keep an extra copy of the mean scores
se1 <- sqrt (apply(res1,1,var, na.rm = TRUE))  # SE of estimate
# create a file of histograms of ranks and scores
numacc <- dim(rank1)[[1]]
mn2 <- numeric (numacc)
sd1 <- numeric (numacc)
postscript ("plot20120222.ps", horizontal=FALSE)
par (mfrow=c(3,2))
xrange <- c(min(res1,na.rm=TRUE), max(res1,na.rm=TRUE))
estrange <- c(min(res1), max(res1))
rankrange <- c(min(rank1), max(rank1))
for (i in 1:numacc) {
  mn2 [i] <- mean (rank1 [i,,], na.rm = TRUE)
  sd1 [i] <- sqrt (var (rank1 [i,,], na.rm = TRUE))
  main1 <- data2 [i,1]                  #  <- check this, make sure the right names are
      # used!
  hist (rank1 [i,,], xlab="rank", main=main1, xlim=rankrange)
  hist (res1 [i,,], xlab="score", main=main1, xlim=estrange)
}
dev.off()

temp <- sort (mn1, index.return=TRUE)
oan <- data2 [temp$ix,1]         # holds the names of the accessions used for ranking
mn1 <- mn1[temp$ix]
se1 <- se1[temp$ix]
mn2 <- mn2 [temp$ix]
sd1 <- sd1 [temp$ix]
for (i in 1:length(oan)) {
  if (i == 1) {
    cat(sprintf("%-40s %10s  %15s   %10s  %15s 
", "Accession", "estimate", "SE
(estimate)", "rank", "SD (rank)"))
  }
  cat(sprintf("%-40s %10.6f  %15.6f   %10.6f  %15.6f 
", oan[i], mn1[i], se1[i], mn2[i],
  sd1[i]))
}

#--------------------------------------------------------------------#
# (6)
# check correlations with trials not used

cn1 <- colnames (data2)               # columns used in data2
idx1 <- !(colnames(data1) %in% cn1)
data1b <- data1[,idx1]

rn1 <- match(oan, data1[,1], nomatch=0)
data1b <- data1b[rn1,]

# check correlation matrix again for data1
sink("corr2b.txt")
cat ("trial1 numpairs correl.\n")
for (i in 1:(ncol(data1b))) {
  tl <- !is.na(data1b[,i])
  td1 <- data1b [tl,i]
  td2 <- mn1[tl]
  if (sum(tl) > 1) {
    cor1 <- sprintf ("%6.3f", cor(td1,td2))
  } else {
    cor1 <- "NA"
  }
  cat (sprintf ("%8d %8d %6s\n", i, sum(tl), cor1))
}
sink()

data1b <- data1[,idx1]
for (i in 1:ncol(data1b)) {
  cat (sprintf ("%-20s %8d
", colnames(data1b)[i], sum(!is.na(data1b[,i]))))
}

#---------- put back in the missing trials and accessions -------------------#
# (7)
# accessions
# which ones are they??
data3 <- data1[,match(cn1, colnames(data1), nomatch=0)]  # all accessions
res3 <- data.frame(data2[,1], mn1.s)
idx1 <- match (res3[,1], data3[,1], nomatch=0)          # match the names
data3b <- data3[idx1,]                                 # just the rows that match accessions in res3
# now we need to go through the columns in data3, get subsets that have accessions in common with res3
# and do the prediction
sink ("Predicted20120202a.txt")
for (i in 2:ncol(data3b)) {
  idx2 <- !is.na(data3b[,i])
  # cat (sum(idx2))     # how many are there?
  lm1 <- lm (res3[idx2,2] ~ data3b[idx2,i])
  pred1 <- lm1$coefficients[1] + lm1$coefficients[2] * data3[,i]
  idx3 <- !is.na(pred1)
  idx4 <- match(data3[idx3,1], res3[,1], nomatch=NA)
  cat(colnames(data3)[i], "has", sum(idx2), "accessions available \n")
  cat(sprintf("%-50s   %8s  %8s\n", "Accession","predicted","composite"))
  cat(sprintf("%-50s   %8.4f  %8.4f\n", data3[idx3,1], pred1[idx3], res3[idx4,2]))
  cat (sprintf ("Standard Deviation of the Differences: %8.4f\n", 
                sqrt(var(pred1[idx3] - res3[idx4,2], na.rm=TRUE))))
  cat ("\n")
}
sink()

# getting predictions for accessions from trials not used in modeling
idx1 <- is.na(!match(colnames(data1), cn1))
idx1[1] <- TRUE     # include column with names
data4 <- data1[,idx1]  # data4 now has only the columns that were not used for the composite scores
rn1 <- match(oan, data1[,1], nomatch=0)
data4b <- data4[rn1,]
sink ("Predicted20120202b.txt")
for (i in 2:ncol(data4)) {
  idx2 <- !is.na(data4b[,i])
  # cat (sum(idx2))     # how many are there?
  if (sum(idx2) > 1) {
    lm1 <- lm (res3[idx2,2] ~ data4b[idx2,i])
    pred1 <- lm1$coefficients[1] + lm1$coefficients[2] * data4[,i]
    idx3 <- !is.na(pred1)
    idx4 <- match(data4[idx3,1], res3[,1], nomatch=NA)
    cat(colnames(data4)[i], "has", sum(idx2), "accessions available \n")
    cat(sprintf("%-50s   %8s  %8s\n", "Accession","predicted","composite"))
    cat(sprintf("%-50s   %8.4f  %8.4f\n", data4[idx3,1], pred1[idx3], res3[idx4,2]))
    cat (sprintf ("Standard Deviation of the Differences: %8.4f\n", 
                    sqrt(var(pred1[idx3] - res3[idx4,2], na.rm=TRUE))))
    cat ("\n")
  }
}
sink()