Supporting information to accompany: A simplified algorithm for dealing with inconsistencies using the Analytic Hierarchy Process

R code and associated output used in the study

Contents

example-script.R

2022-10-13

```
### A simplified algorithm for dealing with inconsistency
### using the Analytic Hierarchy Process
###
### Example script for three comparison matrix (A, B, C)

## structure
## 1  AB AC
## BA 1  BC
## CA CB 1

# set the initial values of the comparison matrix
AHP_AB <- 4
AHP_AC <- 6
AHP_BC <- 5

AHP_BA <- 1/AHP_AB
AHP_CA <- 1/AHP_AC
AHP_CB <- 1/AHP_BC

#estimate GMM weights

GMM_A <- (1*AHP_AB*AHP_AC)^(1/3)
GMM_B <- (AHP_BA*1*AHP_BC)^(1/3)
GMM_C <- (AHP_CA*AHP_CB*1)^(1/3)

A_GMM <- GMM_A/(GMM_A+GMM_B+GMM_C)
B_GMM <- GMM_B/(GMM_A+GMM_B+GMM_C)
C_GMM <- GMM_C/(GMM_A+GMM_B+GMM_C)


### estimate initial GCI
n <- 3
GCI <- 2/((n-1)*(n-2))*(((log(AHP_AB*GMM_B/GMM_A))^2)+
                          ((log(AHP_AC*GMM_C/GMM_A))^2)+
                          ((log(AHP_BC*GMM_C/GMM_B))^2))

initial <- cbind(A_GMM,B_GMM,C_GMM, GCI)
GCI

## [1] 0.4831835

#####################################################
# adjust scores to reduce inconsistency  #####
#####################################################


  NEW_AB <- ifelse(AHP_AC*AHP_CB>9,9,
                   ifelse(AHP_AC*AHP_CB<1/9,1/9,AHP_AC*AHP_CB))
  NEW_BA <- 1/NEW_AB
  NEW_AC <- ifelse(AHP_AB*AHP_BC>9,9,
                   ifelse(AHP_AB*AHP_BC<1/9,1/9,AHP_AB*AHP_BC))
```

```
  NEW_CA <- 1/NEW_AC
  NEW_BC <- ifelse(AHP_BA*AHP_AC>9,9,
                   ifelse(AHP_BA*AHP_AC<1/9,1/9,AHP_BA*AHP_AC))
  NEW_CB <- 1/NEW_BC


  ## look for greatest relative differences

  D1 <- ((AHP_AB-NEW_AB)^2)/AHP_AB
  D2 <- ((AHP_AC-NEW_AC)^2)/AHP_AC
  D3 <- ((AHP_BC-NEW_BC)^2)/AHP_BC


# find maximum value
  temp <- as.data.frame(cbind(GCI,D1,D2,D3))
  max <- apply(temp[,2:4],1, FUN = function(x) {max(x[x > 0])})


#replace the cell with the maximum difference with the esttimated value
  AHP_AB <- ifelse(temp$GCI>0.315,ifelse(D1==max, NEW_AB,AHP_AB),AHP_AB)
  AHP_BA <- 1/AHP_AB
  AHP_AC <- ifelse(temp$GCI>0.315,ifelse(D2==max, NEW_AC,AHP_AC),AHP_AC)
  AHP_CA <- 1/AHP_AC
  AHP_BC <- ifelse(temp$GCI>0.315,ifelse(D3==max, NEW_BC,AHP_BC),AHP_BC)
  AHP_CB <- 1/AHP_BC

#re-estimate GMM weights

  GMM_A <- (1*AHP_AB*AHP_AC)^(1/3)
  GMM_B <- (1*AHP_BA*AHP_BC)^(1/3)
  GMM_C <- (1*AHP_CA*AHP_CB)^(1/3)

  A_GMM <- GMM_A/(GMM_A+GMM_B+GMM_C)
  B_GMM <- GMM_B/(GMM_A+GMM_B+GMM_C)
  C_GMM <- GMM_C/(GMM_A+GMM_B+GMM_C)


#re-estimate GCI

  n <- 3
  GCI <- 2/((n-1)*(n-2))*(((log(AHP_AB*GMM_B/GMM_A))^2)+
                            ((log(AHP_AC*GMM_C/GMM_A))^2)+
                            ((log(AHP_BC*GMM_C/GMM_B))^2))


  revised <-cbind(A_GMM,B_GMM,C_GMM, GCI)

# compare the outputs
  initial

##          A_GMM     B_GMM      C_GMM       GCI
## [1,] 0.6733904 0.2514779 0.07513163 0.4831835

  revised
```

```
##           A_GMM     B_GMM     C_GMM GCI
## [1,] 0.7058824 0.1764706 0.1176471   0
```

example-script-triad.R

2022-10-13

```
### A simplified algorithm for dealing with inconsistency
### using the Analytic Hierarchy Process
###
### Example script for three comparison matrix (A, B, C)

## structure
## 1  AB AC
## BA 1  BC
## CA CB 1

# set the initial values of the comparison matrix
AHP_AB <- 4
AHP_AC <- 0.2
AHP_BC <- 5

AHP_BA <- 1/AHP_AB
AHP_CA <- 1/AHP_AC
AHP_CB <- 1/AHP_BC

#estimate GMM weights

GMM_A <- (1*AHP_AB*AHP_AC)^(1/3)
GMM_B <- (AHP_BA*1*AHP_BC)^(1/3)
GMM_C <- (AHP_CA*AHP_CB*1)^(1/3)

A_GMM <- GMM_A/(GMM_A+GMM_B+GMM_C)
B_GMM <- GMM_B/(GMM_A+GMM_B+GMM_C)
C_GMM <- GMM_C/(GMM_A+GMM_B+GMM_C)


### estimate initial GCI
n <- 3
GCI <- 2/((n-1)*(n-2))*(((log(AHP_AB*GMM_B/GMM_A))^2)+
                          ((log(AHP_AC*GMM_C/GMM_A))^2)+
                          ((log(AHP_BC*GMM_C/GMM_B))^2))

initial <- cbind(A_GMM,B_GMM,C_GMM, GCI)
GCI

## [1] 7.069197

#####################################################
# adjust scores to reduce inconsistency  #####
#####################################################


  NEW_AB <- ifelse(AHP_AC*AHP_CB>9,9,
                   ifelse(AHP_AC*AHP_CB<1/9,1/9,AHP_AC*AHP_CB))
  NEW_BA <- 1/NEW_AB
  NEW_AC <- ifelse(AHP_AB*AHP_BC>9,9,
                   ifelse(AHP_AB*AHP_BC<1/9,1/9,AHP_AB*AHP_BC))
```

```
  NEW_CA <- 1/NEW_AC
  NEW_BC <- ifelse(AHP_BA*AHP_AC>9,9,
                   ifelse(AHP_BA*AHP_AC<1/9,1/9,AHP_BA*AHP_AC))
  NEW_CB <- 1/NEW_BC

  cbind(NEW_AB,NEW_AC,NEW_BC)
```

```
##         NEW_AB NEW_AC    NEW_BC
## [1,] 0.1111111      9 0.1111111
```

```
  ## look for greatest relative differences

  D1 <- ((AHP_AB-NEW_AB)^2)/AHP_AB
  D2 <- ((AHP_AC-NEW_AC)^2)/AHP_AC
  D3 <- ((AHP_BC-NEW_BC)^2)/AHP_BC


# find maximum value
  temp <- as.data.frame(cbind(GCI,D1,D2,D3))
  max <- apply(temp[,2:4],1, FUN = function(x) {max(x[x > 0])})


#replace the cell with the maximum difference with the esttimated value
  AHP_AB <- ifelse(temp$GCI>0.315,ifelse(D1==max, NEW_AB,AHP_AB),AHP_AB)
  AHP_BA <- 1/AHP_AB
  AHP_AC <- ifelse(temp$GCI>0.315,ifelse(D2==max, NEW_AC,AHP_AC),AHP_AC)
  AHP_CA <- 1/AHP_AC
  AHP_BC <- ifelse(temp$GCI>0.315,ifelse(D3==max, NEW_BC,AHP_BC),AHP_BC)
  AHP_CB <- 1/AHP_BC

  cbind(AHP_AB,AHP_AC,AHP_BC)
```

```
##      AHP_AB AHP_AC AHP_BC
## [1,]      4      9      5
```

```
#re-estimate GMM weights

  GMM_A <- (1*AHP_AB*AHP_AC)^(1/3)
  GMM_B <- (1*AHP_BA*AHP_BC)^(1/3)
  GMM_C <- (1*AHP_CA*AHP_CB)^(1/3)

  A_GMM <- GMM_A/(GMM_A+GMM_B+GMM_C)
  B_GMM <- GMM_B/(GMM_A+GMM_B+GMM_C)
  C_GMM <- GMM_C/(GMM_A+GMM_B+GMM_C)


#re-estimate GCI

  n <- 3
  GCI <- 2/((n-1)*(n-2))*(((log(AHP_AB*GMM_B/GMM_A))^2)+
                            ((log(AHP_AC*GMM_C/GMM_A))^2)+
                            ((log(AHP_BC*GMM_C/GMM_B))^2))


  revised <-cbind(A_GMM,B_GMM,C_GMM, GCI)
```

```
# compare the outputs
  initial

##          A_GMM     B_GMM     C_GMM      GCI
## [1,] 0.3088694 0.3584112 0.3327195 7.069197

  revised

##          A_GMM     B_GMM      C_GMM       GCI
## [1,] 0.7085242 0.2311482 0.06032764 0.2125382
```

example-script-extreme-triad.R

2022-10-13

```
### A simplified algorithm for dealing with inconsistency
### using the Analytic Hierarchy Process
###
### Example script for three comparison matrix (A, B, C)

## structure
## 1  AB AC
## BA 1  BC
## CA CB 1

# set the initial values of the comparison matrix
AHP_AB <- 9
AHP_AC <- 9
AHP_BC <- 9

AHP_BA <- 1/AHP_AB
AHP_CA <- 1/AHP_AC
AHP_CB <- 1/AHP_BC

#estimate GMM weights

GMM_A <- (1*AHP_AB*AHP_AC)^(1/3)
GMM_B <- (AHP_BA*1*AHP_BC)^(1/3)
GMM_C <- (AHP_CA*AHP_CB*1)^(1/3)

A_GMM <- GMM_A/(GMM_A+GMM_B+GMM_C)
B_GMM <- GMM_B/(GMM_A+GMM_B+GMM_C)
C_GMM <- GMM_C/(GMM_A+GMM_B+GMM_C)


### estimate initial GCI
n <- 3
GCI <- 2/((n-1)*(n-2))*(((log(AHP_AB*GMM_B/GMM_A))^2)+
                          ((log(AHP_AC*GMM_C/GMM_A))^2)+
                          ((log(AHP_BC*GMM_C/GMM_B))^2))

initial <- cbind(A_GMM,B_GMM,C_GMM, GCI)
GCI

## [1] 1.609265

#####################################################
# adjust scores to reduce inconsistency  #####
#####################################################


  NEW_AB <- ifelse(AHP_AC*AHP_CB>9,9,
                   ifelse(AHP_AC*AHP_CB<1/9,1/9,AHP_AC*AHP_CB))
  NEW_BA <- 1/NEW_AB
  NEW_AC <- ifelse(AHP_AB*AHP_BC>9,9,
                   ifelse(AHP_AB*AHP_BC<1/9,1/9,AHP_AB*AHP_BC))
```

```
  NEW_CA <- 1/NEW_AC
  NEW_BC <- ifelse(AHP_BA*AHP_AC>9,9,
                   ifelse(AHP_BA*AHP_AC<1/9,1/9,AHP_BA*AHP_AC))
  NEW_CB <- 1/NEW_BC


  ## look for greatest relative differences

  D1 <- ((AHP_AB-NEW_AB)^2)/AHP_AB
  D2 <- ((AHP_AC-NEW_AC)^2)/AHP_AC
  D3 <- ((AHP_BC-NEW_BC)^2)/AHP_BC


# find maximum value
  temp <- as.data.frame(cbind(GCI,D1,D2,D3))
  max <- apply(temp[,2:4],1, FUN = function(x) {max(x[x > 0])})


#replace the cell with the maximum difference with the esttimated value
  AHP_AB <- ifelse(temp$GCI>0.315,ifelse(D1==max, NEW_AB,AHP_AB),AHP_AB)
  AHP_BA <- 1/AHP_AB
  AHP_AC <- ifelse(temp$GCI>0.315,ifelse(D2==max, NEW_AC,AHP_AC),AHP_AC)
  AHP_CA <- 1/AHP_AC
  AHP_BC <- ifelse(temp$GCI>0.315,ifelse(D3==max, NEW_BC,AHP_BC),AHP_BC)
  AHP_CB <- 1/AHP_BC

#re-estimate GMM weights

  GMM_A <- (1*AHP_AB*AHP_AC)^(1/3)
  GMM_B <- (1*AHP_BA*AHP_BC)^(1/3)
  GMM_C <- (1*AHP_CA*AHP_CB)^(1/3)

  A_GMM <- GMM_A/(GMM_A+GMM_B+GMM_C)
  B_GMM <- GMM_B/(GMM_A+GMM_B+GMM_C)
  C_GMM <- GMM_C/(GMM_A+GMM_B+GMM_C)


#re-estimate GCI

  n <- 3
  GCI <- 2/((n-1)*(n-2))*(((log(AHP_AB*GMM_B/GMM_A))^2)+
                            ((log(AHP_AC*GMM_C/GMM_A))^2)+
                            ((log(AHP_BC*GMM_C/GMM_B))^2))


  revised <-cbind(A_GMM,B_GMM,C_GMM, GCI)

# compare the outputs
  initial

##          A_GMM     B_GMM      C_GMM      GCI
## [1,] 0.7784906 0.1799251 0.04158436 1.609265

  revised
```

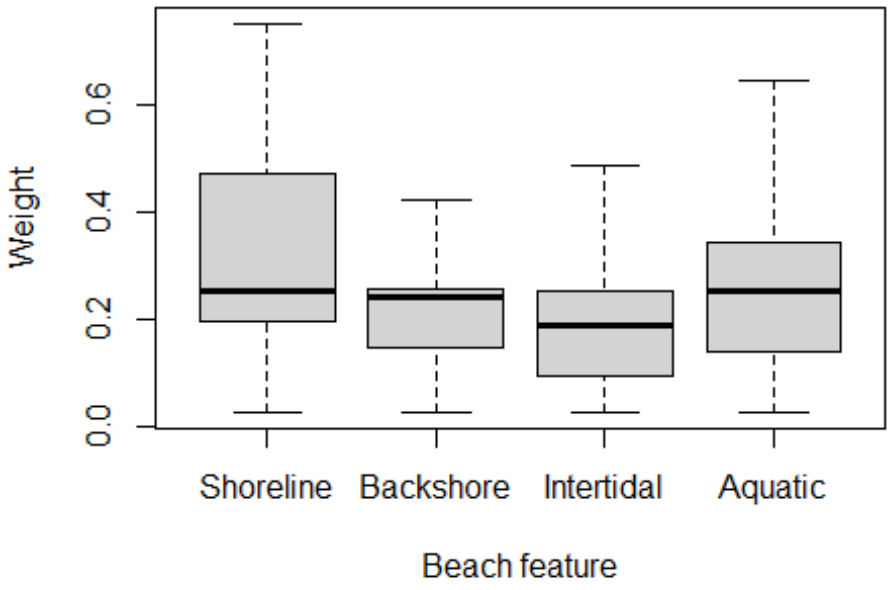
```
##           A_GMM     B_GMM     C_GMM      GCI
## [1,] 0.5841564 0.2808331 0.1350105 1.609265
```

coastal-example.R

2022-10-13

```
###
### A simplified algorithm for dealing with inconsistencies using AHP
###

#### Coastal example  #####


## 1  AB AC AD
## BA 1  BC BD
## CA CB 1  CD
## DA DB DC 1


#### AHP scores
data <- read.csv("coastal.csv", header=TRUE)

head(data)

##   X AHP_AB    AHP_AC    AHP_AD AHP_BC AHP_BD AHP_CD
## 1 1      1 1.0000000 1.0000000    1.0   1.00   1.00
## 2 2      1 0.2000000 0.3333333    0.2   0.25   5.00
## 3 3      1 1.0000000 0.3333333    1.0   0.25   1.00
## 4 4      1 1.0000000 1.0000000    1.0   1.00   1.00
## 5 5      1 1.0000000 1.0000000    1.0   1.00   1.00
## 6 6      1 0.3333333 0.3333333    1.0   0.25   0.25

AHP_AB <- data$AHP_AB
AHP_BA <- 1/AHP_AB
AHP_AC <- data$AHP_AC
AHP_CA <- 1/AHP_AC
AHP_AD <- data$AHP_AD
AHP_DA <- 1/AHP_AD
AHP_BC <- data$AHP_BC
AHP_CB <- 1/AHP_BC
AHP_BD <- data$AHP_BD
AHP_DB <- 1/AHP_BD
AHP_CD <- data$AHP_CD
AHP_DC <- 1/AHP_CD

## calculate the weights based on the original 1-9 scores
GMM_A <- (1*AHP_AB*AHP_AC*AHP_AD)^(1/4)
GMM_B <- (AHP_BA*1*AHP_BC*AHP_BD)^(1/4)
GMM_C <- (AHP_CA*AHP_CB*1*AHP_CD)^(1/4)
GMM_D <- (AHP_DA*AHP_DB*AHP_DC*1)^(1/4)

sumval <- (GMM_A+GMM_B+GMM_C+GMM_D)

A_GMM <- GMM_A/sumval
B_GMM <- GMM_B/sumval
C_GMM <- GMM_C/sumval
D_GMM <- GMM_D/sumval
```

```
## calculate the geometric consistency index
n <- 4
GCI <- (2/((n-1)*(n-2)))*(((log(AHP_AB*GMM_B/GMM_A))^2)+
                          ((log(AHP_AC*GMM_C/GMM_A))^2)+
                          ((log(AHP_AD*GMM_D/GMM_A))^2)+
                          ((log(AHP_BC*GMM_C/GMM_B))^2)+
                          ((log(AHP_BD*GMM_D/GMM_B))^2)+
                          ((log(AHP_CD*GMM_D/GMM_C))^2))



boxplot(cbind(A_GMM,B_GMM,C_GMM,D_GMM),names=c("Shoreline", "Backshore","Intertidal",
"Aquatic"),ylab="Weight",xlab="Beach feature", outline=FALSE)
```

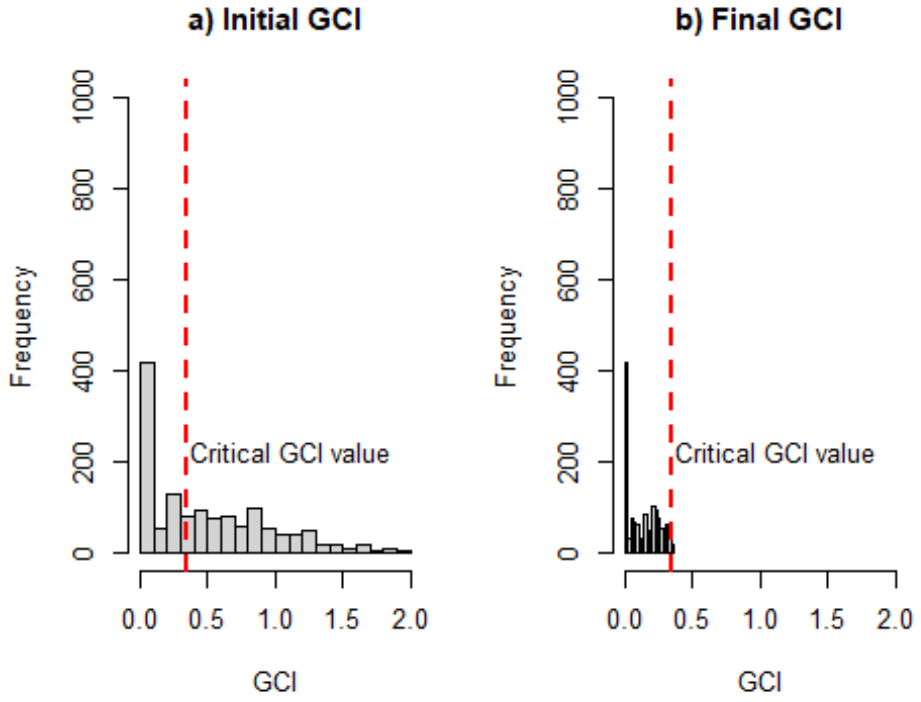



```
original <- as.data.frame(cbind(A_GMM,B_GMM,C_GMM,D_GMM, GCI))


#####################################################
# adjusting scores to reduce inconsistency  #####
#####################################################

count <- 0
improve1 <- NULL
improve2 <- NULL

## Loop twenty times  ###
for(i in 1:20) {
```

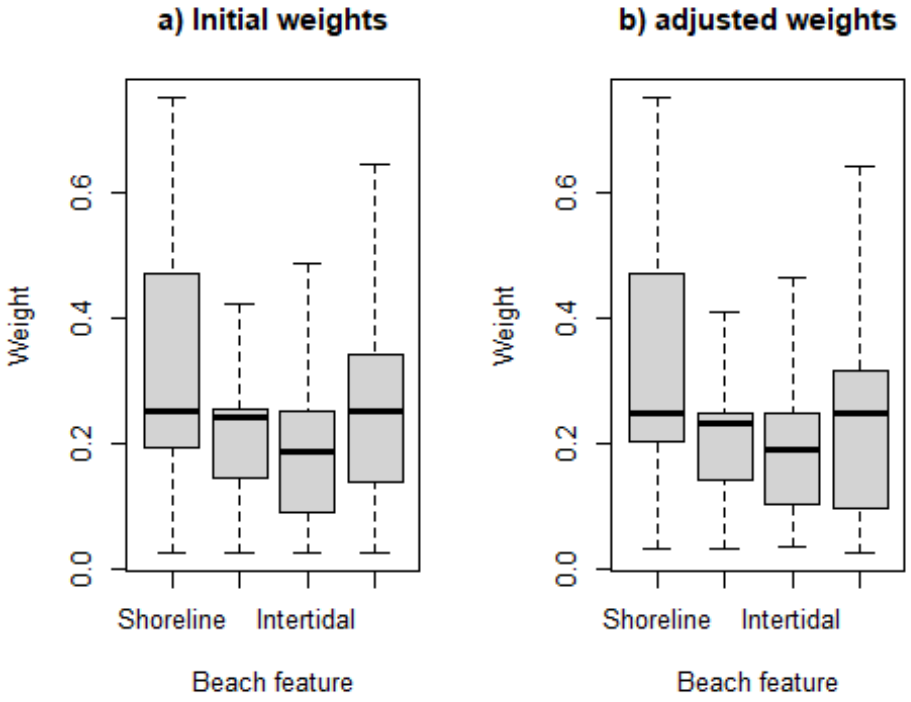
```
  ## 1  AB AC AD
  ## BA 1  BC BD
  ## CA CB 1  CD
  ## DA DB DC 1


#  calculate the expected values as a
#  geometric mean of the potential combinations

  NEW_AB1 <- ifelse(AHP_AC*AHP_CB>9,9,
                    ifelse(AHP_AC*AHP_CB<1/9,1/9,AHP_AC*AHP_CB))
  NEW_AB2 <- ifelse(AHP_AD*AHP_DB>9,9,
                     ifelse(AHP_AD*AHP_DB<1/9,1/9,AHP_AD*AHP_DB))
  NEW_AB3 <- ifelse(AHP_AC*AHP_CD*AHP_DB>9,9,
                     ifelse(AHP_AC*AHP_CD*AHP_DB<1/9,1/9,AHP_AC*AHP_CD*AHP_DB))
  NEW_AB <- (NEW_AB1*NEW_AB2*NEW_AB3)^(1/3)

  NEW_BA <- 1/NEW_AB


  NEW_AC1 <- ifelse(AHP_AB*AHP_BC>9,9,
                    ifelse(AHP_AB*AHP_BC<1/9,1/9,AHP_AB*AHP_BC))
  NEW_AC2 <- ifelse(AHP_AD*AHP_DC>9,9,
                     ifelse(AHP_AD*AHP_DC<1/9,1/9,AHP_AD*AHP_DC))
  NEW_AC3 <- ifelse(AHP_AB*AHP_BD*AHP_DC>9,9,
                     ifelse(AHP_AB*AHP_BD*AHP_DC<1/9,1/9,AHP_AB*AHP_BD*AHP_DC))
  NEW_AC <- (NEW_AC1*NEW_AC2*NEW_AC3)^(1/3)

  NEW_CA <- 1/NEW_AC


  NEW_AD1 <- ifelse(AHP_AB*AHP_BD>9,9,
                    ifelse(AHP_AB*AHP_BD<1/9,1/9,AHP_AB*AHP_BD))
  NEW_AD2 <- ifelse(AHP_AC*AHP_CD>9,9,
                     ifelse(AHP_AC*AHP_CD<1/9,1/9,AHP_AC*AHP_CD))
  NEW_AD3 <- ifelse(AHP_AB*AHP_BC*AHP_CD>9,9,
                     ifelse(AHP_AB*AHP_BC*AHP_CD<1/9,1/9,AHP_AB*AHP_BC*AHP_CD))
  NEW_AD <- (NEW_AD1*NEW_AD2*NEW_AD3)^(1/3)

    NEW_DA <- 1/NEW_AD


  NEW_BC1 <- ifelse(AHP_BA*AHP_AC>9,9,
                    ifelse(AHP_BA*AHP_AC<1/9,1/9,AHP_BA*AHP_AC))
  NEW_BC2 <- ifelse(AHP_BD*AHP_DC>9,9,
                     ifelse(AHP_BD*AHP_DC<1/9,1/9,AHP_BD*AHP_DC))
  NEW_BC3 <- ifelse(AHP_BA*AHP_AD*AHP_DC>9,9,
                     ifelse(AHP_BA*AHP_AD*AHP_DC<1/9,1/9,AHP_BA*AHP_AD*AHP_DC))
  NEW_BC <- (NEW_BC1*NEW_BC2*NEW_BC3)^(1/3)

  NEW_CB <- 1/NEW_BC
```

```
  NEW_BD1 <- ifelse(AHP_BC*AHP_CD>9,9,
                    ifelse(AHP_BC*AHP_CD<1/9,1/9,AHP_BC*AHP_CD))
  NEW_BD2 <- ifelse(AHP_BA*AHP_AD>9,9,
                     ifelse(AHP_BA*AHP_AD<1/9,1/9,AHP_BA*AHP_AD))
  NEW_BD3 <- ifelse(AHP_BA*AHP_AC*AHP_CD>9,9,
                     ifelse(AHP_BA*AHP_AC*AHP_CD<1/9,1/9,AHP_BA*AHP_AC*AHP_CD))
  NEW_BD <- (NEW_BD1*NEW_BD2*NEW_BD3)^(1/3)

  NEW_DB <- 1/NEW_BD


  NEW_CD1 <- ifelse(AHP_CB*AHP_BD>9,9,
                    ifelse(AHP_CB*AHP_BD<1/9,1/9,AHP_CB*AHP_BD))
  NEW_CD2 <- ifelse(AHP_CA*AHP_AD>9,9,
                    ifelse(AHP_AB*AHP_AD<1/9,1/9,AHP_AB*AHP_AD))
  NEW_CD3 <- ifelse(AHP_CA*AHP_AB*AHP_BD>9,9,
                    ifelse(AHP_CA*AHP_AB*AHP_BD<1/9,1/9,AHP_CA*AHP_AB*AHP_BD))
  NEW_CD <- (NEW_CD1*NEW_CD2*NEW_CD3)^(1/3)

  NEW_DC <- 1/NEW_CD


  D1 <- ((AHP_AB-NEW_AB)^2)/AHP_AB
  D2 <- ((AHP_AC-NEW_AC)^2)/AHP_AC
  D3 <- ((AHP_BC-NEW_BC)^2)/AHP_BC
  D4 <- ((AHP_AD-NEW_AD)^2)/AHP_AD
  D5 <- ((AHP_BD-NEW_BD)^2)/AHP_BD
  D6 <- ((AHP_CD-NEW_CD)^2)/AHP_CD


# find maximum value
  temp <- as.data.frame(cbind(GCI,D1,D2,D3,D4,D5,D6))
  max <- apply(temp[,2:7],1, FUN = function(x) {max(x)})
  inconsistency <- sum(ifelse(GCI>0.353,1,0))/length(GCI)

  AHP_AB <- ifelse(GCI>0.353,ifelse(D1==max, NEW_AB,AHP_AB),AHP_AB)
  AHP_BA <- 1/AHP_AB
  AHP_AC <- ifelse(GCI>0.353,ifelse(D2==max, NEW_AC,AHP_AC),AHP_AC)
  AHP_CA <- 1/AHP_AC
  AHP_BC <- ifelse(GCI>0.353,ifelse(D3==max, NEW_BC,AHP_BC),AHP_BC)
  AHP_CB <- 1/AHP_BC
  AHP_AD <- ifelse(GCI>0.353,ifelse(D4==max, NEW_AD,AHP_AD),AHP_AD)
  AHP_DA <- 1/AHP_AD
  AHP_BD <- ifelse(GCI>0.353,ifelse(D5==max, NEW_BD,AHP_BD),AHP_BD)
  AHP_DB <- 1/AHP_BD
  AHP_CD <- ifelse(GCI>0.353,ifelse(D6==max, NEW_CD,AHP_CD),AHP_CD)
  AHP_DC <- 1/AHP_CD


  GMM_A <- (1*AHP_AB*AHP_AC*AHP_AD)^(1/4)
  GMM_B <- (AHP_BA*1*AHP_BC*AHP_BD)^(1/4)
  GMM_C <- (AHP_CA*AHP_CB*1*AHP_CD)^(1/4)
  GMM_D <- (AHP_DA*AHP_DB*AHP_DC*1)^(1/4)
```

```
  sumval <- (GMM_A+GMM_B+GMM_C+GMM_D)

  A_GMM <- GMM_A/sumval
  B_GMM <- GMM_B/sumval
  C_GMM <- GMM_C/sumval
  D_GMM <- GMM_D/sumval

  checkwt <- (A_GMM+B_GMM+C_GMM+D_GMM)
  checkwt

  # recalcualte the GCI
  n <- 4
  GCI <- (2/((n-1)*(n-2)))*(((log(AHP_AB*GMM_B/GMM_A))^2)+
                              ((log(AHP_AC*GMM_C/GMM_A))^2)+
                              ((log(AHP_AD*GMM_D/GMM_A))^2)+
                              ((log(AHP_BC*GMM_C/GMM_B))^2)+
                              ((log(AHP_BD*GMM_D/GMM_B))^2)+
                              ((log(AHP_CD*GMM_D/GMM_C))^2))




  adjusted <- as.data.frame(cbind(A_GMM,B_GMM,C_GMM,D_GMM, GCI))


   count <- count+1
   improve1 <- cbind(count,inconsistency)
   improve2 <- rbind(improve2,improve1)
  }

# drop the observations with GCI>2 just for the histograms (and see what % is exclude
d)
# this is just for the purposes of graphing the result

  original1 <- subset(original,GCI<2)
  (dim(original)-dim(original1))/dim(original) # check the proportion of obs removed

## [1] 0.04314003 0.00000000

  adjusted1 <- subset(adjusted,GCI<2)
  (dim(adjusted)-dim(adjusted1))/dim(adjusted) # check the proportion of obs removed

## [1] 0 0

# produce the two histograms

  par(mfrow=c(1,2),cex=0.8)
  hist(original1$GCI,xlim=c(0,2),breaks=20,main="a) Initial GCI", xlab="GCI",ylim=c(0
,1000))
  abline(v=0.353,col="red", lty=2, lwd=2)
  text(0.38,200,"Critical GCI value",adj=c(0,0))
  hist(adjusted1$GCI,xlim=c(0,2),breaks=20,main="b) Final GCI", xlab="GCI",ylim=c(0,1
000))
  abline(v=0.353,col="red", lty=2, lwd=2)
  text(0.38,200,"Critical GCI value",adj=c(0,0))
```

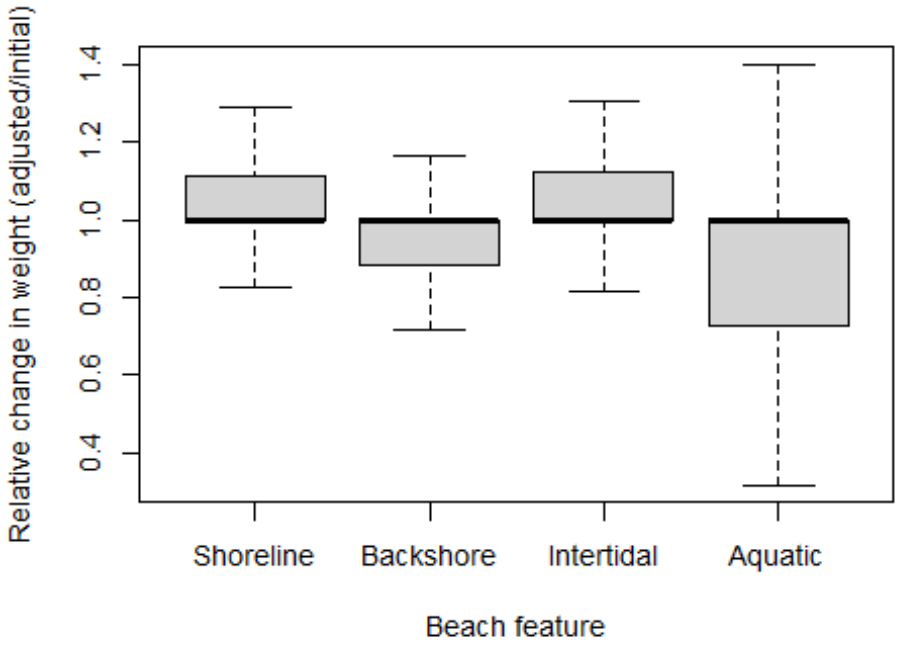



```
## do direct comparisons by asset

par(mfrow=c(1,2),cex=0.8)
boxplot(cbind(original$A_GMM,original$B_GMM,original$C_GMM,original$D_GMM),names=c("S
horeline", "Backshore","Intertidal","Aquatic"),ylab="Weight",xlab="Beach feature",out
line=FALSE, main="a) Initial weights")
boxplot(cbind(adjusted1$A_GMM,adjusted1$B_GMM,adjusted1$C_GMM,adjusted1$D_GMM),names=
c("Shoreline", "Backshore","Intertidal","Aquatic"),ylab="Weight",xlab="Beach feature"
,outline=FALSE,main="b) adjusted weights")
```

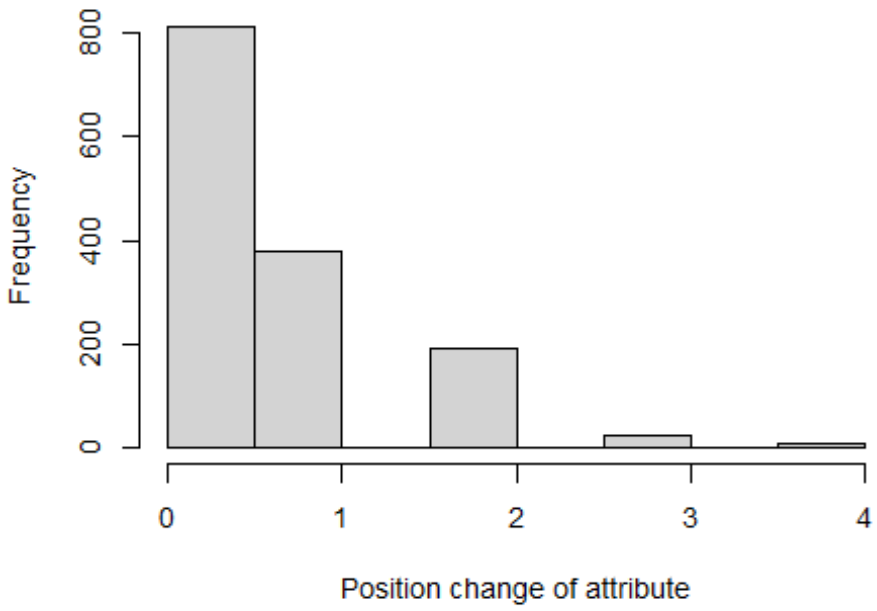



```
# summary statistics on original and adjusted values
summary(original)

##      A_GMM            B_GMM            C_GMM            D_GMM
##  Min.   :0.0250   Min.   :0.0250   Min.   :0.0250   Min.   :0.0250
##  1st Qu.:0.1942   1st Qu.:0.1439   1st Qu.:0.0918   1st Qu.:0.1382
##  Median :0.2500   Median :0.2405   Median :0.1864   Median :0.2500
##  Mean   :0.3166   Mean   :0.2318   Mean   :0.1855   Mean   :0.2661
##  3rd Qu.:0.4701   3rd Qu.:0.2555   3rd Qu.:0.2500   3rd Qu.:0.3406
##  Max.   :0.7500   Max.   :0.6997   Max.   :0.6789   Max.   :0.7000
##       GCI
##  Min.   :0.00000
##  1st Qu.:0.01379
##  Median :0.40815
##  Mean   :0.60216
##  3rd Qu.:0.87173
##  Max.   :5.58507

summary(adjusted)

##      A_GMM            B_GMM            C_GMM            D_GMM
##  Min.   :0.03458   Min.   :0.03422   Min.   :0.03695   Min.   :0.02651
##  1st Qu.:0.20268   1st Qu.:0.14202   1st Qu.:0.10579   1st Qu.:0.09893
##  Median :0.25000   Median :0.23356   Median :0.19288   Median :0.25000
##  Mean   :0.32991   Mean   :0.23038   Mean   :0.20050   Mean   :0.23921
##  3rd Qu.:0.47222   3rd Qu.:0.25000   3rd Qu.:0.25000   3rd Qu.:0.31631
##  Max.   :0.75000   Max.   :0.70888   Max.   :0.66374   Max.   :0.74468
##       GCI
##  Min.   :0.0000
##  1st Qu.:0.0000
```

```
##  Median :0.1386
##  Mean   :0.1336
##  3rd Qu.:0.2293
##  Max.   :0.3506

## look at correlation between the original and adjusted values
cor(as.data.frame(original),as.data.frame(adjusted))

##            A_GMM        B_GMM        C_GMM      D_GMM          GCI
## A_GMM  0.8613313 -0.355939410 -0.094186617 -0.6487119  0.087155711
## B_GMM -0.1909911  0.814367312 -0.118102172 -0.3315410 -0.008524216
## C_GMM -0.5324186 -0.007270329  0.655876804  0.1634843 -0.187054883
## D_GMM -0.5103647 -0.180890386 -0.241910174  0.8952777  0.029623782
## GCI    0.1400396 -0.023765230 -0.002722852 -0.1407087  0.344543536

## count the number of observations that were consistent before and after adjustment
original2 <- subset(original,GCI<0.353)
1-(dim(original)-dim(original2))/dim(original) # check the proportion of obs removed

## [1] 0.4575672 1.0000000

adjusted2 <- subset(adjusted,GCI<0.353)
1-(dim(adjusted)-dim(adjusted2))/dim(adjusted) # check the proportion of obs removed

## [1] 1 1

## distribution of changes in the weights
summary(adjusted/original)

##      A_GMM            B_GMM            C_GMM            D_GMM
##  Min.   :0.3328   Min.   :0.2570   Min.   : 0.1837   Min.   :0.0994
##  1st Qu.:1.0000   1st Qu.:0.8842   1st Qu.: 1.0000   1st Qu.:0.7261
##  Median :1.0000   Median :1.0000   Median : 1.0000   Median :1.0000
##  Mean   :1.1211   Mean   :1.0427   Mean   : 1.3736   Mean   :0.8987
##  3rd Qu.:1.1149   3rd Qu.:1.0000   3rd Qu.: 1.1207   3rd Qu.:1.0000
##  Max.   :5.9229   Max.   :3.6209   Max.   :11.4817   Max.   :3.1439
##
##       GCI
##  Min.   :0.0000
##  1st Qu.:0.1123
##  Median :0.2856
##  Mean   :0.4459
##  3rd Qu.:1.0000
##  Max.   :1.0000
##  NA's   :332

par(mfrow=c(1,1), cex=0.9)
boxplot(adjusted[,1:4]/original[,1:4], outline=FALSE,names=c("Shoreline", "Backshore"
,"Intertidal","Aquatic"),ylab="Relative change in weight (adjusted/initial)",xlab="Be
ach feature")
```

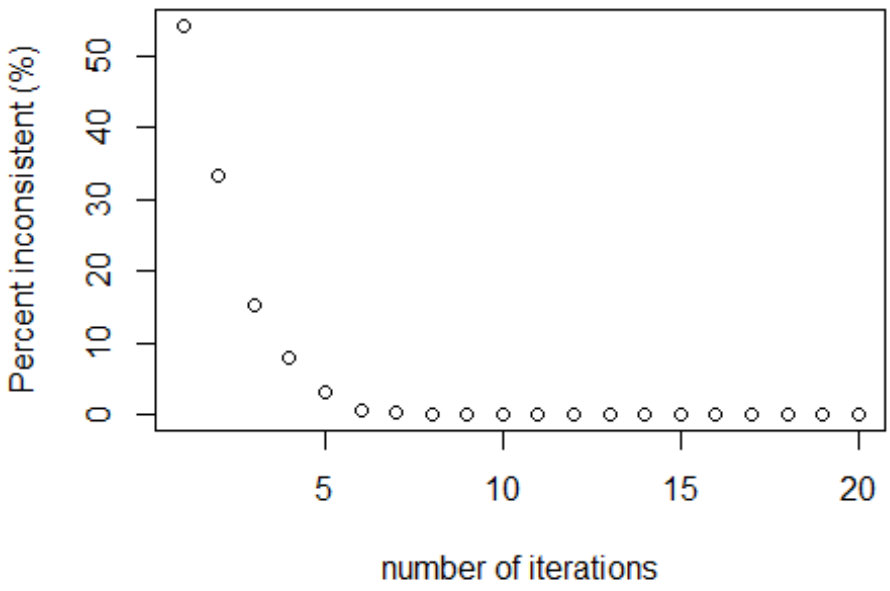



```
### impact on changes in rankings
rank_original <- data.frame(original[,1:4], t(apply(-original[,1:4], 1, rank, ties.me
thod='min')))
rank_adjusted <- data.frame(adjusted[,1:4], t(apply(-adjusted[,1:4], 1, rank, ties.me
thod='min')))

head(rank_original)

##        A_GMM     B_GMM     C_GMM     D_GMM A_GMM.1 B_GMM.1 C_GMM.1 D_GMM.1
## 1 0.25000000 0.2500000 0.2500000 0.2500000       1       1       1       1
## 2 0.09123701 0.0849056 0.6003733 0.2234841       3       4       1       2
## 3 0.17555660 0.1633738 0.2310455 0.4300241       3       4       2       1
## 4 0.25000000 0.2500000 0.2500000 0.2500000       1       1       1       1
## 5 0.25000000 0.2500000 0.2500000 0.2500000       1       1       1       1
## 6 0.11910981 0.1458791 0.1919877 0.5430233       4       3       2       1

head(rank_adjusted)

##        A_GMM     B_GMM     C_GMM     D_GMM A_GMM.1 B_GMM.1 C_GMM.1 D_GMM.1
## 1 0.25000000 0.2500000 0.2500000 0.2500000       1       1       1       1
## 2 0.09123701 0.0849056 0.6003733 0.2234841       3       4       1       2
## 3 0.17555660 0.1633738 0.2310455 0.4300241       3       4       2       1
## 4 0.25000000 0.2500000 0.2500000 0.2500000       1       1       1       1
## 5 0.25000000 0.2500000 0.2500000 0.2500000       1       1       1       1
## 6 0.11910981 0.1458791 0.1919877 0.5430233       4       3       2       1

rank_change <- abs(rank_original[,5:8]-rank_adjusted[,5:8])
rank_change_sum <- round(apply(rank_change,1,sum)/2,0)
par(mfrow=c(1,1), cex=0.9)
hist(rank_change_sum,main="",xlab="Position change of attribute")
```




```
table(rank_change_sum)/length(rank_change_sum)  # proportion in each change category

## rank_change_sum
##           0           1           2           3           4
## 0.575671853 0.267326733 0.135785007 0.016265912 0.004950495

### number of potential circular triads  ####

# number of times the same score was given to all comparisons
ECT2 <- ifelse(data[,2]==data[,3]&data[,3]==data[,4]&data[,4]==data[,5]&data[,5]==dat
a[,6]&data[,6]==data[,7],data[,2],0)
table(ECT2)/length(ECT2)

## ECT2
##                 0 0.111111111111111 0.142857142857143 0.166666666666667
##      0.7567185290      0.0035360679      0.0007072136      0.0007072136
##               0.2              0.25 0.333333333333333               0.5
##      0.0014144272      0.0007072136      0.0021216407      0.0021216407
##                 1                 2                 3                 4
##      0.2065063649      0.0007072136      0.0056577086      0.0021216407
##                 5                 6                 7                 9
##      0.0070721358      0.0014144272      0.0063649222      0.0021216407

# number of times a "positive" score was given to all comparisons
ECT3 <- ifelse(data[,2]>1&data[,3]>1&data[,4]>1&data[,5]>1&data[,6]>1, 1, 0)
table(ECT3)/length(ECT3)

## ECT3
##         0         1
## 0.8670438 0.1329562
```

```r
# number of times a "negative" score was given to all comparisons
ECT4 <- ifelse(data[,2]<1&data[,3]<1&data[,4]<1&data[,5]<1&data[,6]<1, 1, 0)
table(ECT4)/length(ECT4)

## ECT4
##          0          1
## 0.92644979 0.07355021

# number of times the expected value changed from positive to negative (or vice versa
)
x <- ifelse(log(AHP_AB)>0,ifelse((log(NEW_AB)+log(AHP_AB))/2<log(AHP_AB),1,0),ifelse(
(log(NEW_AB)+log(AHP_AB))/2>log(AHP_AB),1,0))
table(x)/length(x)

## x
##        0        1
## 0.523338 0.476662

y <- ifelse(log(AHP_AC)>0,ifelse((log(NEW_AC)+log(AHP_AC))/2<log(AHP_AC),1,0),ifelse(
(log(NEW_AC)+log(AHP_AC))/2>log(AHP_AC),1,0))
table(y)/length(y)

## y
##         0         1
## 0.5693069 0.4306931

z <- ifelse(log(AHP_AD)>0,ifelse((log(NEW_AD)+log(AHP_AD))/2<log(AHP_AD),1,0),ifelse(
(log(NEW_AD)+log(AHP_AD))/2>log(AHP_AD),1,0))
table(z)/length(z)

## z
##         0         1
## 0.7029703 0.2970297

par(mfrow=c(1,1))
plot(inconsistency*100~count,data=improve2,ylab="Percent inconsistent (%)",xlab="numb
er of iterations")
```




```
improve2[1:20,]

##       count inconsistency
##  [1,]     1   0.542432815
##  [2,]     2   0.331683168
##  [3,]     3   0.152050919
##  [4,]     4   0.079915134
##  [5,]     5   0.030410184
##  [6,]     6   0.004243281
##  [7,]     7   0.001414427
##  [8,]     8   0.000000000
##  [9,]     9   0.000000000
## [10,]    10   0.000000000
## [11,]    11   0.000000000
## [12,]    12   0.000000000
## [13,]    13   0.000000000
## [14,]    14   0.000000000
## [15,]    15   0.000000000
## [16,]    16   0.000000000
## [17,]    17   0.000000000
## [18,]    18   0.000000000
## [19,]    19   0.000000000
## [20,]    20   0.000000000

### check for cyclic triads
D12 <- ifelse(log(NEW_AB)*log(AHP_AB)>0,0,1)
D22 <- ifelse(log(NEW_AC)*log(AHP_AC)>0,0,1)
D32 <- ifelse(log(NEW_BC)*log(AHP_BC)>0,0,1)
D42 <- ifelse(log(NEW_AD)*log(AHP_AD)>0,0,1)
D52 <- ifelse(log(NEW_BD)*log(AHP_BD)>0,0,1)
```

```
D62 <- ifelse(log(NEW_CD)*log(AHP_CD)>0,0,1)
summary(cbind(D12,D22,D32,D42,D52,D62))
```

```
##       D12              D22              D32              D42        
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000  
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000  
##  Median :0.0000   Median :0.0000   Median :0.0000   Median :0.0000  
##  Mean   :0.3861   Mean   :0.3649   Mean   :0.4279   Mean   :0.3663  
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000  
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000  
##       D52              D62        
##  Min.   :0.000   Min.   :0.0000  
##  1st Qu.:0.000   1st Qu.:0.0000  
##  Median :0.000   Median :0.0000  
##  Mean   :0.401   Mean   :0.4533  
##  3rd Qu.:1.000   3rd Qu.:1.0000  
##  Max.   :1.000   Max.   :1.0000
```