Code and statistical analysis in R

```
#Xie, G., Guo, Y., Tong, S., and Ma, L. (2014). Calculate excess mortality during heatwaves
#using Hilbert-Huang transform algorithm. BMC medical research methodology, 14, 35.
```

```
####################################################################
##############
# 12/12/2012
# The following R programs are prepared by Gang Xie  (John Xie) at
# Queensland University of Technology, Brisbane, Australia,
# for implementation of Hilbert-Huang Transform (HHT) algorithm.
#
# These R programs should only be used free of charge for the non-profit purposes
# such as  academic study and research.
# Disclaimer: No liability is accepted by the author
# (or any affiliated organisation) for any loss or damage
# that may be caused by using this software.
####################################################################
##############
#
#
# These are a group of special R functions for time series data analysis
# using HHT algorithm originally introduced by Norden E. Huang, et. al.
# A large part of the R code contained in this file are the realisation of
# the Matlab functions written  by Zhaohua Wu and these Matlab programs are free available
# from the website http://rcada.ncu.edu.tw/.
# Details are as follows.
# EEMDR: is essentially a R realisation of Matlab program 'eemd'; the major difference between
#        EEMDR and eemd is in its subroutine function 'extrema'.  In EEMDR, the function
'extrema'
#        is a copy of the 'extrema' function from the R package 'EMD' authored by
#        Donghoh Kim and Hee-Seok Oh.  Depending on  specification of the input
arguments,
#        users can decompose the input time series by EMD procedure or EEMD procedure.
#        EEMDR gives both the numeric output and graphic output of the generated IMFs.
# EEMDR1: is a simplified version of EEMDR function.  EEMDR1 acts as a subroutine function
#        to be called by function 'postEEMD'.  EEMDR1 generates the post processed IMFs
#        from EEMD procedure.  Details of post processing treatment on EEMD can be
found
#        in reference [4].
# postEEMD: a function which generates the post processed IMFs.  There is no counterpart
#        Matlab function.
# ifndqR: This is a function to calculate instantaneous frequency based on HHT algorithm,i.e.
#        a translation of the Matlab code function 'ifndq' written by Zhaohua Wu
(zwu@fsu.edu).
# testimf: This is a significance test function based on the average energy of IMFs to
distinguish
```

```
#             those    trend IMFs from the "random" (i.e. non-trend) IMFs.    This function is of our
#             original work by implementing the theoretic results given in [6].    Wu also
#                  provided a significance test function, 'significance' (need to call a subroutine
function
#               'dist_value'),    in Matlab code.    Based on our experience, we observed that our
approach
#             can achieve the same test result but our test function is simpler in programming.
#                 More importantly, the theoretic results given in [6] imply that our approach is
applicable
#             in    more general case than identifying white noise non-trend IMFs.
#
#-------------------------------------------------------------------------------
# This is a utitlity program (i.e. subroutine function)    being called by 'EEMDR' or 'EEMDR1'.
# This function, extrema, returns the positions of extrema (minima or maxima points) and
# the number of extrema of a time series sample data.
# This program is copied from R package 'EMD' authored by Donghoh Kim and Hee-Seok Oh
# (published on 29/10/2012)..
#
# INPUT:
#             y:    An input time series from which the positions of extrema will be determined
#
# OUPUT:
#             minindex:    positions or index of the minima points.
#             maxindex:    positions or index of the maxima points.
#             nextreme:    number of extrema.
#


 extrema <- function (y, ndata = length(y), ndatam1 = ndata - 1)
{
      minindex <- maxindex <- NULL
      nextreme <- 0
      cross <- NULL
      ncross <- 0
      z1 <- sign(diff(y))
      index1 <- seq(1, ndatam1)[z1 != 0]
      z1 <- z1[z1 != 0]
      if (!(is.null(index1) || all(z1 == 1) || all(z1 == -1))) {
            index1 <- index1[c(z1[-length(z1)] != z1[-1], FALSE)] +
                  1
            z1 <- z1[c(z1[-length(z1)] != z1[-1], FALSE)]
            nextreme <- length(index1)
            if (nextreme >= 2)
                  for (i in 1:(nextreme - 1)) {
                        tmpindex <- index1[i]:(index1[i + 1] - 1)
                        if (z1[i] > 0) {
                           tmpindex <- tmpindex[y[index1[i]] == y[tmpindex]]
                           maxindex <- rbind(maxindex, c(min(tmpindex),
                              max(tmpindex)))
                        }
                        else {
                           tmpindex <- tmpindex[y[index1[i]] == y[tmpindex]]
                           minindex <- rbind(minindex, c(min(tmpindex),
                    max(tmpindex)))
                }
            }
        tmpindex <- index1[nextreme]:ndatam1
        if (z1[nextreme] > 0) {
            tmpindex <- tmpindex[y[index1[nextreme]] == y[tmpindex]]
            maxindex <- rbind(maxindex, c(min(tmpindex), max(tmpindex)))
        }
        else {
            tmpindex <- tmpindex[y[index1[nextreme]] == y[tmpindex]]
            minindex <- rbind(minindex, c(min(tmpindex), max(tmpindex)))
        }
        if (!(all(sign(y) >= 0) || all(sign(y) <= 0) || all(sign(y) ==
            0))) {
            index1 <- c(1, index1)
            for (i in 1:nextreme) {
                if (y[index1[i]] == 0) {
                  tmp <- c(index1[i]:index1[i + 1])[y[index1[i]:index1[i +
                    1]] == 0]
                  cross <- rbind(cross, c(min(tmp), max(tmp)))
                }
                else if (y[index1[i]] * y[index1[i + 1]] < 0) {
                  tmp <- min(c(index1[i]:index1[i + 1])[y[index1[i]] *
                    y[index1[i]:index1[i + 1]] <= 0])
                  if (y[tmp] == 0) {
                    tmp <- c(tmp:index1[i + 1])[y[tmp:index1[i +
                      1]] == 0]
                    cross <- rbind(cross, c(min(tmp), max(tmp)))
                  }
                  else cross <- rbind(cross, c(tmp - 1, tmp))
                }
            }
            if (any(y[index1[nextreme + 1]] * y[index1[nextreme +
                1]:ndata] <= 0)) {
                tmp <- min(c(index1[nextreme + 1]:ndata)[y[index1[nextreme +
                  1]] * y[index1[nextreme + 1]:ndata] <= 0])
                if (y[tmp] == 0) {
                  tmp <- c(tmp:ndata)[y[tmp:ndata] == 0]
                  cross <- rbind(cross, c(min(tmp), max(tmp)))
                }
                else cross <- rbind(cross, c(tmp - 1, tmp))
            }
            ncross <- nrow(cross)
        }
    }
    list(minindex = minindex, maxindex = maxindex, nextreme = nextreme,
        cross = cross, ncross = ncross)
}

# end of the function 'extrema'.

#-------------------------------------------------------------------------------
# This is an EMD/EEMD program.    The R code is prepared according to Matlab code
# in http://rcada.ncu.edu.tw/eemd.m.    There are changes in this R code program.
# (1) end effects of EMD are treated according to the PPT document 'End Effects of EMD
# An unsolved, and perhaps, unsolvable problem' presented in    http://rcada.ncu.edu.tw
# website; (2) this R code EMD/EEMD program needs to call a function named 'extrema'
# which returns the minimum and maximum information of an input time sereies.    The
# 'extrema' R function is given in the R package 'EMD'; (3) the spline function used in
# this R EMD/EEMD program is 'splinefun' (R 'base' package) with which the user may specify
# different spline method.    As research shows and with our limited experience, we
recommend
# either the 'fmm' method (default method here) or the 'natural' method (details may
# see the decription to 'splinefun' function in R help manual. (4) The stopping rule applied
# in this R EMD/EEMD program is fixing the number of iterations (of the sifting process) to be
10,
# given the optimal objective is to make sure that all the generated intrinsic mode functions
# (imf) meet the definition criteria (i.e. (a) The number of local extrema of the time
# series and the number of its zero-crossings must either be equal or differ by at most one.
# (b) At any time, the mean value of the upper envelope determined by the local maxima and
the
# lower envelope determined by the local minima is zero).
# Justification of using this fixed number of iterations stopping rule can be found in reference
[5].

# INPUT:
# inputd: input data, a time series
# Nstd: ratio of the standard deviation of the added noise and that of inputd
# NE: Ensemble number for the EEMDR.
# method: cubic spline method chosen for construction of envelopes
# graphoutput: a logic variable, default vaule 1 gives an on-screen graphic output of the
decomposed IMFs;
#                     other values will    export a document named "emdIMFs.pdf".
# viewdiff: a logic variable, default value 0 will NOT output the information of differences of
number of
#                 extrema and the number of zero crossing.    Otherwise, the information will show
on screen.
#
# Note: When Nst is set to zero and NE is set to 1, the program degenerateds to a EMD
program
#
# OUTPUT:
# (1) the numeric output is a n by (NM+1) dimension data frame, where n = length(input time
series)
# and NM = number of imfs.    The first column of the data frame contains the input time series.
# Column 2 to column (NM + 1) are the imfs.    Note that the last imf series is the residual
series.
# Therefore, the values in column 1 can be recovered by summing up    the corresponding
elements in
# column 2 to column (NM + 1).
# (2)    a graphic output which produce the corresponding time series plots of the numeric
output.
#

#-----------------------------------------------
```

```
   EEMDR <- function(inputd, Nstd=0, NE=1, method="fmm",
                        graphoutput=1, viewdiff = 0) {

#

 if(Nstd > 0.3) cat("Warning: Nstd value is recommended to be between 0.05 to 0.2. /n")
 if(NE > 200) cat("Warning: it is recommended for NE to be between 50 to 200. /n")

 if(Nstd == 0 && NE == 1) cat("This program is performing an EMD operation.    \n\n")
      else cat("This program is performing an EEMD operation.    \n\n")

  YY = inputd
  nn =    length(YY); tyr = 1:nn
  dd = 1:nn
  Ystd = sd(YY); YY = YY/Ystd

 TNM = trunc(log(nn)/log(2)) - 1
 TNM2 = TNM + 2

#---------

 Amode = allmode = matrix(rep(0,nn*TNM2), nrow=nn,ncol=TNM2)

#--------

 for(ni in 1:NE)    {

  X1 = YY + rnorm(nn,0,Nstd)

 Amode[,1] = YY

 yorigin = X1
 yend = yorigin

 nmode = 1
 k = 1

#--------

 while (nmode <= TNM)    {

           ystart = yend

 y.extre = extrema(ystart)

 xminp = tyr[y.extre$mini[,1]];    xmaxp = tyr[y.extre$maxi[,1]]

 ncros = y.extre$ncross
 ndiff = length(xminp) + length(xmaxp) - ncros
```

```
#-------
# Apply the fixed number of iterations (iteration    =10) stopage rule.
# Details see "Ensemble empirical mode decomposition:
##     a noise-assisted data analysis method" (2009), appendix A1.
#
# Reference: page 4, "A review on Hilbert-Huang Transform: method and its applications to
# Geophysical studies", by Norden E. Huang and Zhaohua Wu, (2008)
#

                        iter = 1

           while ( iter < 10)


#

                  {
   if ((length(xminp)+length(xmaxp)) < 2)
          {    k = k+1;    break}

  xminp = sort(xminp)
  xmaxp = sort(xmaxp)

  ymin = ystart[xminp];    ymax = ystart[xmaxp]

# end points treatment
   ny1 <- length(ymin); ny2 <- length(ymax)
   ny <- min(ny1, ny2)
  if(ny >= 2)    {
     mxy = (ymin[2] - ymin[1])/(xminp[2] - xminp[1])
     cxy = ymin[1]- mxy*xminp[1]
     ymin1 = mxy*tyr[1] + cxy
     ymin1 = ifelse(ystart[1] <= ymin1, ystart[1], (ymin1+ystart[1])/2)

     mxy = (ymin[(ny1-1)] - ymin[ny1])/(xminp[(ny1-1)] - xminp[ny1])
     cxy = ymin[ny1]- mxy*xminp[ny1]
     yminn = mxy*tyr[nn] + cxy
     yminn = ifelse(ystart[nn] <= yminn, ystart[nn], (yminn+ystart[nn])/2)

     mxy = (ymax[2] - ymax[1])/(xmaxp[2] - xmaxp[1])
     cxy = ymax[1]- mxy*xmaxp[1]
     ymax1 = mxy*tyr[1] + cxy
     ymax1 = ifelse(ystart[1] >= ymax1, ystart[1], (ymax1+ystart[1])/2)

     mxy = (ymax[(ny2-1)] - ymax[ny2])/(xmaxp[(ny2-1)] - xmaxp[ny2])
     cxy = ymax[ny2]- mxy*xmaxp[ny2]
     ymaxn = mxy*tyr[nn] + cxy
     ymaxn = ifelse(ystart[nn] >= ymaxn, ystart[nn], (ymaxn+ystart[nn])/2)

     dmin.f <- splinefun(c(tyr[1],xminp,tyr[nn]),c(ymin1,ymin,yminn), method=method)
     dmax.f <- splinefun(c(tyr[1],xmaxp,tyr[nn]), c(ymax1,ymax,ymaxn), method=method)
                                      }    # end of 'if(ny >= 2) '
#---------------
   else{
     dmin.f <- splinefun(xminp,ymin, method=method)

     dmax.f <- splinefun(xmaxp, ymax, method=method)}

#
   ylower = dmin.f(tyr)


#

   yupper = dmax.f(tyr)


         meanUL = (ylower + yupper) / 2
         ystart = ystart - meanUL

  y.extre = extrema(ystart)

  xminp = tyr[y.extre$mini[,1]];    xmaxp = tyr[y.extre$maxi[,1]]

  ncros = y.extre$ncross
  ndiff = length(xminp) + length(xmaxp) - ncros

                   iter = iter + 1


#

                                }      # end of 'while ( )'

               yend = yend - ystart

                        nmode = nmode + 1

            Amode[,nmode] = ystart
            Amode[,(nmode+1)] = yend

#----------------
  k = k+1
  stp = sum(yend - 0)
   if(stp == 0)    break
#--------------

                    }      # end of 'while (nmode <= TNM)'

          allmode = allmode + Amode
                             } # end of 'for(ni in 1:NE)'

                      allmode = allmode / NE
                      allmode = allmode * Ystd
  if(k < TNM2) {
          for(i in (k+1):TNM2)
             {allmode[,k] = allmode[,k] + allmode[,i]
                     allmode[,i] = 0 } }

#
 if(graphoutput != 1) pdf(file="emdIMFs.pdf")

 op <- par(mfrow=c(k,1),mar=c(0.2,0.5,1,1),mgp=c(0.5,0.2,0))
   a = "imf"; qp = c(0,0.2,0.4,0.6,0.8,1); atw = as.numeric(quantile(1:nn, qp))
    for(i in 1:k)    {
   if(i ==1) {
 plot(allmode[,1], xlab=" ", ylab=" ",type="l",yaxt ="n", xaxt="n" ,
      main = "Raw time series data", cex.main=0.9,lwd=2)
  axis(1, at=atw, labels= rep("", 6), tck = -0.1, tcl =0.2)
  axis(2, at=0, labels= "", tck = -0.1, tcl =0.3)
  axis(3, at=atw, labels= rep("", 6), tck = -0.1, tcl =0.2)
  axis(4, at=0, labels= "", tck = -0.1, tcl =0.3)
                          }

   if( i >1 && i < k)    {
 plot(allmode[,i], xlab=" ", ylab=" ",type="l",yaxt ="n", xaxt="n" )
  axis(1, at=atw, labels= rep("", 6), tck = -0.1, tcl =0.2)
  axis(2, at=0, labels= "", tck = -0.1, tcl =0.3)
  axis(3, at=atw, labels= rep("", 6), tck = -0.1, tcl =0.2)
  axis(4, at=0, labels= "", tck = -0.1, tcl =0.3)
  title(paste(a,(i-1),sep=" "), cex.main=0.9)    }

    if(i == k) {
 plot(allmode[,k], xlab=" ", ylab=" ",type="l",yaxt ="n", xaxt="n",
  main= "Residual series", cex.main=0.9 )
  axis(1, at=atw, labels= rep("", 6), tck = -0.1, tcl =0.2)
  axis(2, at=0, labels= "", tck = -0.1, tcl =0.3)
  axis(3, at=atw, labels= rep("", 6), tck = -0.1, tcl =0.2)
  axis(4, at=0, labels= "", tck = -0.1, tcl =0.3)
                       }

  }    # end of 'for(i in 1:k)'

 par(op)

 if(graphoutput != 1) dev.off()

#
#-------
#
          modes = allmode[,2:k]
       extremaDiff <- NULL
    for(i in 1:(k-1)) {
                midext = extrema(modes[,i])
  mminx = tyr[midext$mini[,1]];    mmaxx = tyr[midext$maxi[,1]]

 ncos = midext$ncross
 ndif = length(mminx) + length(mmaxx) - ncos
```

```
          extremaDiff = c(extremaDiff,ndif) }

   if(viewdiff != 0) {
      cat("Difference between # of extrema and ncross: \n")
      for(j in 1:(k-1)) cat(extremaDiff[j], "\n") }
#
#---------

              invisible(list(allmode = allmode))

                                          }    # end of the function 'EEMDR'

# Examples:
# yt= gsta.df[,2]
# imf.df = EEMDR(yt)
# imf.df = EEMDR(yt,0.1,100)

#----------------------------------------------------------
#
# a simplified version of EEMDR for the purpose of calculating the posterior EEMD
# IMFs.
#

   EEMDR1 <- function(inputd,    method="fmm", viewdiff = 0) {

#



   YY = inputd
   nn =    length(YY); tyr = 1:nn
   dd = 1:nn
   Ystd = sd(YY); YY = YY/Ystd

  TNM = trunc(log(nn)/log(2)) - 1
  TNM2 = TNM + 2

#----------

  Amode = matrix(rep(0,nn*TNM2), nrow=nn,ncol=TNM2)

#--------


  Amode[,1] = YY

  yorigin = YY
  yend = yorigin

  nmode = 1
  k = 1

#--------
```

```
  while (nmode <= TNM)    {

                    ystart = yend

  y.extre = extrema(ystart)

  xminp = tyr[y.extre$mini[,1]];    xmaxp = tyr[y.extre$maxi[,1]]

  ncros = y.extre$ncross
  ndiff = length(xminp) + length(xmaxp) - ncros


#-------
## Apply the fixed number of iterations (iteration    =10) stoppage rule.
# Details see "Ensemble empirical mode decomposition:
##      a noise-assisted data analysis method" (2009), appendix A1.

                        iter = 1

          while ( iter < 10)


                {
    if ((length(xminp)+length(xmaxp)) < 2)
          {    k = k+1;    break}

  xminp = sort(xminp)
  xmaxp = sort(xmaxp)

  ymin = ystart[xminp];    ymax = ystart[xmaxp]

# end points treatment
    ny1 <- length(ymin); ny2 <- length(ymax)
    ny <- min(ny1, ny2)
  if(ny >= 2)    {
      mxy = (ymin[2] - ymin[1])/(xminp[2] - xminp[1])
      cxy = ymin[1]- mxy*xminp[1]
      ymin1 = mxy*tyr[1] + cxy
      ymin1 = ifelse(ystart[1] <= ymin1, ystart[1], (ymin1+ystart[1])/2)

      mxy = (ymin[(ny1-1)] - ymin[ny1])/(xminp[(ny1-1)] - xminp[ny1])
      cxy = ymin[ny1]- mxy*xminp[ny1]
      yminn = mxy*tyr[nn] + cxy
      yminn = ifelse(ystart[nn] <= yminn, ystart[nn], (yminn+ystart[nn])/2)

      mxy = (ymax[2] - ymax[1])/(xmaxp[2] - xmaxp[1])
      cxy = ymax[1]- mxy*xmaxp[1]
      ymax1 = mxy*tyr[1] + cxy
      ymax1 = ifelse(ystart[1] >= ymax1, ystart[1], (ymax1+ystart[1])/2)

      mxy = (ymax[(ny2-1)] - ymax[ny2])/(xmaxp[(ny2-1)] - xmaxp[ny2])
      cxy = ymax[ny2]- mxy*xmaxp[ny2]
      ymaxn = mxy*tyr[nn] + cxy
      ymaxn = ifelse(ystart[nn] >= ymaxn, ystart[nn], (ymaxn+ystart[nn])/2)

      dmin.f <- splinefun(c(tyr[1],xminp,tyr[nn]),c(ymin1,ymin,yminn), method=method)
      dmax.f <- splinefun(c(tyr[1],xmaxp,tyr[nn]), c(ymax1,ymax,ymaxn), method=method)
                                    }    # end of 'if(ny >= 2) '

#--------------
    else{
      dmin.f <- splinefun(xminp,ymin, method=method)

      dmax.f <- splinefun(xmaxp, ymax, method=method)}

#

    ylower = dmin.f(tyr)


#

    yupper = dmax.f(tyr)


          meanUL = (ylower + yupper) / 2
          ystart = ystart - meanUL

  y.extre = extrema(ystart)

  xminp = tyr[y.extre$mini[,1]];    xmaxp = tyr[y.extre$maxi[,1]]

  ncros = y.extre$ncross
  ndiff = length(xminp) + length(xmaxp) - ncros

                 iter = iter + 1
#

                              }      # end of 'while ( )'

                yend = yend - ystart

                      nmode = nmode + 1

              Amode[,nmode] = ystart
              Amode[,(nmode+1)] = yend

#---------------
  k = k+1
  stp = sum(yend - 0)
    if(stp == 0)    break
#-------------

                    }      # end of 'while (nmode <= TNM)'
```

```
 if(k < TNM2) {
              for(i in (k+1):TNM2)
                  {Amode[,k] = Amode[,k] + Amode[,i]
                            Amode[,i] = 0 } }

              Amode = Amode * Ystd
#-------
#
              modes = Amode[,2:k]
          extremaDiff <- NULL
     for(i in 1:(k-1)) {
                        midext = extrema(modes[,i])
  mminx = tyr[midext$mini[,1]];    mmaxx = tyr[midext$maxi[,1]]

 ncos = midext$ncross
 ndif = length(mminx) + length(mmaxx) - ncos
        extremaDiff = c(extremaDiff,ndif) }

 if(viewdiff !=0) {
     cat("Difference between # of extrema and ncross: \n")
     for(j in 1:(k-1)) cat(extremaDiff[j], "\n") }
#
#---------


             return(Amode)

                                        }    # end of the function 'EEMDR1'

#--------------------------------------------------------
#
# a posterior EEMD function to make sure EEMD results are true IMFs.
#
# For Ensemble Empirical Mode Decomposition (EEMD),
#    details see "Ensemble empirical mode decomposition:
##      a noise-assisted data analysis method" (2009), by Zhaohua Wu and Norden E. Huang.


  postEEMD <- function(inputd, Nstd=0, NE=1, graphoutput =1, viewdiff=0)
                                    {

   out.df <- EEMDR(inputd=inputd, Nstd=Nstd, NE = NE)

   nc = dim(out.df$allmode)[2]    # find the number of columns

   nx = length(out.df$allmode[,1])

     postmode = matrix(rep(0,nc*nx),ncol=nc)

   postmode[,1] = out.df$allmode[,1]
 if(nc < 4) stop("The number of columns needs to be greater than 3. /n")

    for(i in 2:(nc-2)) {
            if(i ==2)    midc = out.df$allmode[,i] + out.df$allmode[,(i+1)]
            midmode = EEMDR1(midc)
            postmode[,i] = midmode[,2]
            residc = rep(0,nx); nd = dim(midmode)[2]
        for(j in 3:nd) residc = residc + midmode[,j]
              midc = residc + out.df$allmode[,(i+2)]

            if(i == (nc-2)) {
                    midmode = EEMDR1(midc, viewdiff=viewdiff)
            postmode[,(i+1)] = midmode[,2]
            residc = rep(0,nx); nd = dim(midmode)[2]
        for(j in 3:nd) residc = residc + midmode[,j]
                  postmode[,(i+2)] = residc
                                    }      # end of 'if(i == (nc-2))'

                                            }    # end of 'for(i in 2:(nc-2))'

if(graphoutput != 1) pdf(file="postIMFs.pdf")

              k = nc
op <- par(mfrow=c(k,1),mar=c(0.2,0.5,1,1),mgp=c(0.5,0.2,0))
    a = "post-imf"; qp = c(0,0.2,0.4,0.6,0.8,1); atw = as.numeric(quantile(1:nx, qp))
      for(i in 1:k)    {
    if(i ==1) {
plot(postmode[,1], xlab=" ", ylab=" ",type="l",yaxt ="n", xaxt="n" ,
          main = "Raw time series data", cex.main=0.9, lwd=2)
  axis(1, at=atw, labels= rep("", 6), tck = -0.1, tcl =0.2)
  axis(2, at=0, labels= "", tck = -0.1, tcl =0.3)
  axis(3, at=atw, labels= rep("", 6), tck = -0.1, tcl =0.2)
  axis(4, at=0, labels= "", tck = -0.1, tcl =0.3)
                                        }

    if( i >1 && i < k)    {
plot(postmode[,i], xlab=" ", ylab=" ",type="l",yaxt ="n", xaxt="n" )
  axis(1, at=atw, labels= rep("", 6), tck = -0.1, tcl =0.2)
  axis(2, at=0, labels= "", tck = -0.1, tcl =0.3)
  axis(3, at=atw, labels= rep("", 6), tck = -0.1, tcl =0.2)
  axis(4, at=0, labels= "", tck = -0.1, tcl =0.3)
  title(paste(a,(i-1),sep=" "), cex.main=0.9)    }

      if(i == k) {
plot(postmode[,k], xlab=" ", ylab=" ",type="l",yaxt ="n", xaxt="n",
  main= "Residual series", cex.main=0.9 )
  axis(1, at=atw, labels= rep("", 6), tck = -0.1, tcl =0.2)
  axis(2, at=0, labels= "", tck = -0.1, tcl =0.3)
  axis(3, at=atw, labels= rep("", 6), tck = -0.1, tcl =0.2)
  axis(4, at=0, labels= "", tck = -0.1, tcl =0.3)
                                  }

                                    }    # end of 'for(i in 1:k)'
```

```
     par(op)

if(graphoutput != 1) dev.off()

              invisible(list(postmode = postmode))

                                             }    # end of the function 'postEEMD'

#
# Example:
# postEEMD(yt,0.1,50)



#-----------------------------------------------------------------------
#
#
#
# This is a function to calculate instantaneous frequency based on HHT algorithm
# according to Matlab code of function 'ifndq' written by Zhaohua Wu    (zwu@fsu.edu).
#
# INPUT: vimf: an IMF;
#              dt:      time interval of the inputted data.    e.g. if vimf is a monthly data,
#                          dt = 1 implies the output omega will be in cycle per month; if, instead,
#                          dt = 1/12 implies the output omega will be measured in cycle per year, etc.
# OUTPUT: omega: instantaneous frequency, which is 2*pi/T, where T is hte period of an
ascillation.
#

        ifndqR <- function(vimf, dt, method="fmm") {

                Nnormal    = 5
                rangetop = 0.9

                vlength = length(vimf)
                vlength1 = vlength - 1

        abs_vimf = numeric(vlength)
  for(i in 1:vlength)      abs_vimf[i] = ifelse(vimf[i] >= 0 , vimf[i], -vimf[i])

#---------
# Apply the normalized Hibert Transform procedure.
# Reference: page 6, "A review on Hilbert-Huang Transform: method and its applications to
# Geophysical studies", by Norden E. Huang and Zhaohua Wu, (2008)



       tyr = 1:vlength
       for (j in 1:Nnormal) {

                y.extre = extrema(abs_vimf)
```

```
            xmaxp = tyr[y.extre$maxi[,1]]
            xmaxp = sort(xmaxp)
            ymax = abs_vimf[xmaxp]
            dmax.f <- splinefun(xmaxp,ymax,method=method)
            yupper = dmax.f(tyr)
                  abs_vimf = abs_vimf / yupper
                                                }    # end of 'for (j in 1:Nnormal)'
#

 nvimf = numeric(vlength)
 for(i in 1:vlength)      nvimf[i] = ifelse(vimf[i] >= 0 , abs_vimf[i], -abs_vimf[i])


#--------
 omgcos = rep(-9999, vlength)
 rangebot = -rangetop
    for(i in 2:vlength1)    {
         if(nvimf[i] > rangebot && nvimf[i] < rangetop)
      { omgcos[i] = abs(nvimf[i+1] - nvimf[i-1]) * 0.5/
                  sqrt(1- nvimf[i]*nvimf[i]) }
                                          }

 ddd = temp = NULL
 for(i in 1:vlength) {
                  if(omgcos[i] > -1000)
                     { ddd = c(ddd, i)
                       temp = c(temp, omgcos[i]) }
                                           }

  temp.f <- splinefun(ddd,temp,method=method)
        omega = temp.f(tyr)

     pi2 = 2*pi
  omega = omega/dt/pi2
     invisible(list(omega = omega))


                                        }    # end of function 'ifndqR'


#


#-----------------------------------------------------
#
# An alternative hypothesis testing algorithm to distringuish the trend components from
# the "white noise" / random components.
# This significance test function is different from the Matlab function 'significance'
# written by Zhaohua Wu.    All the formulas are derived from the research results presented
# in reference [6]. This is our original research work.

# This is a function based on the average energy of IMFs to distinguish those
# trend IMFs from the "random" IMFs.
# inputimfs: a matrix which contains all IMFs generated from the application of the function EEMDR
# or function postEEMD.
# This function returns a significance test plot with approximate 95% and 99% confidence band lines.
#

  testimf <- function(inputimfs, nfit = 3, wnoise=1) {

                sizeM = dim(inputimfs)
              nx = sizeM[1]; nc = sizeM[2]
                mse1 = sum(inputimfs[,1]^2)/nx
                ek = NULL
              for(i in 1:nc) ek <- c(ek, mse1*2.01^(-i)/0.719)
#                 ek = c(mse1, ek[-1])
              logek = log(ek)/log(2)

            up95 = logek + 2^(0.474*(1:nc)-2.449)
            up99 = logek + 2^(0.460*(1:nc)-1.919)

     sampek = NULL
       for(i in 1:nc) sampek <- c(sampek, sum(inputimfs[,i]^2)/nx)
       logeek = log(sampek)/log(2)

       yy =c(logek, logeek)
     plot(1:nc, logek, ylim=c(min(yy), max(yy)*1.1),xlab="IMF index",
                  ylab= expression(log[2](energy)))
     lines(1:nc,logek)

     points(1:nc, logeek, pch=8, cex=1.2)
     lines(1:nc,logeek)

     if(wnoise == 1) {
     lines(1:nc, up95, lty=3)
     lines(1:nc, up99, lty=2) }
     else{
     yc = logeek[1:nfit]; xc = 1:nfit
     lmc = lm(yc~xc)
     abline(lmc)
     logek1 = as.numeric(lmc$coef[2])*(1:nc) + as.numeric(lmc$coef[1])
       up95a = logek1 + 2^(0.474*(1:nc)-2.449)
       up99a = logek1 + 2^(0.460*(1:nc)-1.919)
       lines(1:nc, up95a, lty=3)
       lines(1:nc, up99a, lty=2)    }

                                         }    # end of the function


# Examples:
# imf.dd = EEMDR(dailyD)
#    modes = imf.dd$allmode[,2:10]
# testimf(modes)
#-----------------------------------------------------------------------------------------------
```

```
######################################################################
########################################

# major references:
# [1] Norden E. Huang, and et al., The empirical mode decomposition method and the Hilbert spectrum for
#         non-stationary time series analysis, Proc. Roy. Soc. London 454A (1998) 903 - 995.
# [2] Zhaohua Wu, and et al., On the trend, detrending, and variability of nonlinear and nonstationary
#           time series, Proceedings of the National Academy of Sciences of the USA (PANS), September 18, 2007,
#         Vol. 104, No. 38, 14889-14894.
# [3] Norden E. Huang and Zhaohua Wu, A review on Hilbert-Huang transform: method and its applications
#                     to geophysical studies, Reviews of Geophysics, 46, RG2006, doi:10.1029/2007RG000228, 2008.
# [4] Zhaohua Wu and Norden E. Huang, Ensemble Empirical Mode Decomposition: A Noise-Assisted Data Analysis
#         Method, Advances in Adaptive Data Analysis, Vol. 1, No. 1, (2009) 1-41.
# [5] Gang Wang, Xian-Yao Chen, Fang-li Qiao, Zhaohua Wu, and Norden E. Huang,   On Intrinsic Mode Function,
#        Advances in Adaptive Data Analysis, Vol. 2, No. 3, (2010) 277-293.
# [6] Patrick Flandrin and Gabriel Rilling and Paulo Gongcalves , EMD equivalent filter banks, from interpretation
#         to applications, in book Hilbert-Huang Transform and Its Applications edited by N.E. Huang and S. Shen(2005).
#
######################################################################
#########################################
```

```
source.with.encoding('D:/HHT-R-code-XIE-Dec2012.txt', encoding='UTF-8')
```

if D:/ is the location of the source file HHT-R-code-XIE-Dec2012.txt

```
#Kim, D. and Oh, H. S. (2009). EMD: A package for empirical mode decomposition
#and Hilbert spectrum. The R Journal, 1, 40-46.
#Kim, D. and Oh, H. S. (2014). EMD: Empirical mode decomposition and Hilbert
#spectral analysis. R package version 1.5.7.
```

```
library("emd")

postexper1Ozsquaemdtryimfwave <- postEEMD(c(exper1Ozsqua[,"OzSquare"]), c(exper1Ozsqua[,"msec"]))

postexper2Ozsquaemdtryimfwave <- postEEMD(c(exper2Ozsqua[,"OzSquare"]),
```

c(exper2Ozsqua[,"msec"]))

postexper1Ozdiamoemdtryimfwave <- postEEMD(c(exper1OZdiamo[,"OzDiamond"]), c(exper1OZdiamo[,"msec"]))

postexper2Ozdiamoemdtryimfwave <- postEEMD(c(exper2OZdiamo[,"OzDiamond"]), c(exper2OZdiamo[,"msec"]))

postexper1OzMaskemdtryimfwave <- postEEMD(c(exper1OzMask[,"OzMask"]), c(exper1OzMask[,"msec"]))

postexper2OzMaskemdtryimfwave <- postEEMD(c(exper2OzMask[,"OzMask"]), c(exper2OzMask[,"msec"]))

postexper1OzPseudomaskemdtryimfwave <- postEEMD(c(exper1OzPseudomask[,"OzPseudomask"]), c(exper1OzPseudomask[,"msec"]))

postexper2OzPseudomaskemdtryimfwave <- postEEMD(c(exper2OzPseudomask[,"OzPseudomask"]), c(exper2OzPseudomask[,"msec"]))

postcombinexp1OzSquareMaskEEMD <- postEEMD(c(combinexp1Oz[,"OzSquareMask"]), c(combinexp1Oz[,"msec"]))

postcombinexp2OzSquareMaskEEMD <- postEEMD(c(combinexp2Oz[,"OzSquareMask"]), c(combinexp2Oz[,"msec"]))

postcombinexp1OzSquarePseudoEEMD <- postEEMD(c(combinexp1Oz[,"OzSquarePseudo"]), c(combinexp1Oz[,"msec"]))

postcombinexp2OzSquarePseudoEEMD <- postEEMD(c(combinexp2Oz[,"OzSquarePseudo"]), c(combinexp2Oz[,"msec"]))

postcombinexp1OzDiamondMaskEEMD <- postEEMD(c(combinexp1Oz[,"OzDiamondMask"]), c(combinexp1Oz[,"msec"]))

postcombinexp2OzDiamondMaskEEMD <- postEEMD(c(combinexp2Oz[,"OzDiamondMask"]), c(combinexp2Oz[,"msec"]))

postcombinexp1OzDiamondPseudoEEMD <- postEEMD(c(combinexp1Oz[,"OzDiamondPseudo"]), c(combinexp1Oz[,"msec"]))

postcombinexp2OzDiamondPseudoEEMD <- postEEMD(c(combinexp2Oz[,"OzDiamondPseudo"]), c(combinexp2Oz[,"msec"]))

source.with.encoding('D:/HHT-R-code-XIE-Dec2012.txt', encoding='UTF-8')

if D:/ is the location of the source file HHT-R-code-XIE-Dec2012.txt

postexper1T5squaemdtryimfwave <- postEEMD(c(exper1T5squa[,"T5Square"]),

```
c(exper1T5squa[,"msec"]))

postexper2T5squaemdtryimfwave <- postEEMD(c(exper2T5squa[,"T5Square"]), c(exper2T5squa[,"msec"]))

postexper1T5diamoemdtryimfwave <- postEEMD(c(exper1T5diamo[,"T5Diamond"]), c(exper1T5diamo[,"msec"]))

postexper2T5diamoemdtryimfwave <- postEEMD(c(exper2T5diamo[,"T5Diamond"]), c(exper2T5diamo[,"msec"]))

postexper1T5Maskemdtryimfwave <- postEEMD(c(exper1T5Mask[,"T5Mask"]), c(exper1T5Mask[,"msec"]))

postexper2T5Maskemdtryimfwave <- postEEMD(c(exper2T5Mask[,"T5Mask"]), c(exper2T5Mask[,"msec"]))

postexper1T5Pseudomaskemdtryimfwave <- postEEMD(c(exper1T5Pseudomask[,"T5Pseudomask"]), c(exper1T5Pseudomask[,"msec"]))

postexper2T5Pseudomaskemdtryimfwave <- postEEMD(c(exper2T5Pseudomask[,"T5Pseudomask"]), c(exper2T5Pseudomask[,"msec"]))

postcombinexp1T5SquareMaskEEMD <- postEEMD(c(combinexp1T5[,"T5SquareMask"]), c(combinexp1T5[,"msec"]))

postcombinexp2T5SquareMaskEEMD <- postEEMD(c(combinexp2T5[,"T5SquareMask"]), c(combinexp2T5[,"msec"]))

postcombinexp1T5SquarePseudoEEMD <- postEEMD(c(combinexp1T5[,"T5SquarePseudo"]), c(combinexp1T5[,"msec"]))

postcombinexp2T5SquarePseudoEEMD <- postEEMD(c(combinexp2T5[,"T5SquarePseudo"]), c(combinexp2T5[,"msec"]))

postcombinexp1T5DiamondMaskEEMD <- postEEMD(c(combinexp1T5[,"T5DiamondMask"]), c(combinexp1T5[,"msec"]))

postcombinexp2T5DiamondMaskEEMD <- postEEMD(c(combinexp2T5[,"T5DiamondMask"]), c(combinexp2T5[,"msec"]))

postcombinexp1T5DiamondPseudoEEMD <- postEEMD(c(combinexp1T5[,"T5DiamondPseudo"]), c(combinexp1T5[,"msec"]))

postcombinexp2T5DiamondPseudoEEMD <- postEEMD(c(combinexp2T5[,"T5DiamondPseudo"]), c(combinexp2T5[,"msec"]))


#Bowman, D. (2014). The Hilbert-Huang Transform: Tools and Methods. R package 'hht'. R
#package version 2.1.0.
```

```
library("hht")

hhtpostEEMDexp1Ozsqua <- hilbertspec(postexper1Ozsquaemdtryimfwave$postmode[,2:8])

hhtpostEEMDexp2Ozsqua <- hilbertspec(postexper2Ozsquaemdtryimfwave$postmode[,2:8])

hhtpostEEMDexp1Ozdiamo <- hilbertspec(postexper1Ozdiamoemdtryimfwave$postmode[,2:8])

hhtpostEEMDexp2Ozdiamo <- hilbertspec(postexper2Ozdiamoemdtryimfwave$postmode[,2:8])

hhtpostEEMDexp1OzMask <- hilbertspec(postexper1OzMaskemdtryimfwave$postmode[,2:8])

hhtpostEEMDexp2OzMask <- hilbertspec(postexper2OzMaskemdtryimfwave$postmode[,2:8])

hhtpostEEMDexp1OzPseudomask                                                    <-
hilbertspec(postexper1OzPseudomaskemdtryimfwave$postmode[,2:8])

hhtpostEEMDexp2OzPseudomask                                                    <-
hilbertspec(postexper2OzPseudomaskemdtryimfwave$postmode[,2:8])

hhtpostEEMDcombinexp1OzSquareMask                                              <-
hilbertspec(postcombinexp1OzSquareMaskEEMD$postmode[,2:8])

hhtpostEEMDcombinexp2OzSquareMask                                              <-
hilbertspec(postcombinexp2OzSquareMaskEEMD$postmode[,2:8])

hhtpostEEMDcombinexp1OzSquarePseudo                                            <-
hilbertspec(postcombinexp1OzSquarePseudoEEMD$postmode[,2:8])

hhtpostEEMDcombinexp2OzSquarePseudo                                            <-
hilbertspec(postcombinexp2OzSquarePseudoEEMD$postmode[,2:8])

hhtpostEEMDcombinexp1OzDiamondMask                                             <-
hilbertspec(postcombinexp1OzDiamondMaskEEMD$postmode[,2:8])

hhtpostEEMDcombinexp2OzDiamondMask                                             <-
hilbertspec(postcombinexp2OzDiamondMaskEEMD$postmode[,2:8])

hhtpostEEMDcombinexp1OzDiamondPseudo                                           <-
hilbertspec(postcombinexp1OzDiamondPseudoEEMD$postmode[,2:8])

hhtpostEEMDcombinexp2OzDiamondPseudo                                           <-
hilbertspec(postcombinexp2OzDiamondPseudoEEMD$postmode[,2:8])



hhtpostEEMDexp1T5squa <- hilbertspec(postexper1T5squaemdtryimfwave$postmode[,2:8])

hhtpostEEMDexp2T5squa <- hilbertspec(postexper2T5squaemdtryimfwave$postmode[,2:8])
```

```
hhtpostEEMDexp1T5diamo <- hilbertspec(postexper1T5diamoemdtryimfwave$postmode[,2:8])

hhtpostEEMDexp2T5diamo <- hilbertspec(postexper2T5diamoemdtryimfwave$postmode[,2:8])

hhtpostEEMDexp1T5Mask <- hilbertspec(postexper1T5Maskemdtryimfwave$postmode[,2:8])

hhtpostEEMDexp2T5Mask <- hilbertspec(postexper2T5Maskemdtryimfwave$postmode[,2:8])

hhtpostEEMDexp1T5Pseudomask                                                    <-
hilbertspec(postexper1T5Pseudomaskemdtryimfwave$postmode[,2:8])

hhtpostEEMDexp2T5Pseudomask                                                    <-
hilbertspec(postexper2T5Pseudomaskemdtryimfwave$postmode[,2:8])

hhtpostEEMDcombinexp1T5SquareMask                                              <-
hilbertspec(postcombinexp1T5SquareMaskEEMD$postmode[,2:8])

hhtpostEEMDcombinexp2T5SquareMask                                              <-
hilbertspec(postcombinexp2T5SquareMaskEEMD$postmode[,2:8])

hhtpostEEMDcombinexp1T5SquarePseudo                                            <-
hilbertspec(postcombinexp1T5SquarePseudoEEMD$postmode[,2:8])

hhtpostEEMDcombinexp2T5SquarePseudo                                            <-
hilbertspec(postcombinexp2T5SquarePseudoEEMD$postmode[,2:8])

hhtpostEEMDcombinexp1T5DiamondMask                                             <-
hilbertspec(postcombinexp1T5DiamondMaskEEMD$postmode[,2:8])

hhtpostEEMDcombinexp2T5DiamondMask                                             <-
hilbertspec(postcombinexp2T5DiamondMaskEEMD$postmode[,2:8])

hhtpostEEMDcombinexp1T5DiamondPseudo                                           <-
hilbertspec(postcombinexp1T5DiamondPseudoEEMD$postmode[,2:8])

hhtpostEEMDcombinexp2T5DiamondPseudo                                           <-
hilbertspec(postcombinexp2T5DiamondPseudoEEMD$postmode[,2:8])


#Bergsma, Tim
#https://cran.r-project.org/web/packages/csv/index.html

library ("csv")



write.csv(hhtpostEEMDcombinexp1OzSquareMask$amplitude[,],"hhtpostEEMDcombinexp1Oz
SquareMaskamplitude.csv")
```

```
write.csv(hhtpostEEMDcombinexp2OzSquareMask$amplitude[,],"hhtpostEEMDcombinexp2Oz
SquareMaskamplitude.csv")


write.csv(hhtpostEEMDcombinexp1OzSquareMask$instantfreq[,],"hhtpostEEMDcombinexp1Oz
SquareMaskinstantfreq.csv")

write.csv(hhtpostEEMDcombinexp2OzSquareMask$instantfreq[,],"hhtpostEEMDcombinexp2Oz
SquareMaskinstantfreq.csv")


write.csv(hhtpostEEMDcombinexp1OzSquareMask$energy,"hhtpostEEMDcombinexp1OzSquar
eMaskenergy.csv")


write.csv(hhtpostEEMDcombinexp2OzSquareMask$energy,"hhtpostEEMDcombinexp2OzSquar
eMaskenergy.csv")


write.csv(hhtpostEEMDcombinexp1OzSquarePseudo$amplitude[,],"hhtpostEEMDcombinexp1O
zSquarePseudoamplitude.csv")


write.csv(hhtpostEEMDcombinexp2OzSquarePseudo$amplitude[,],"hhtpostEEMDcombinexp2O
zSquarePseudoamplitude.csv")


write.csv(hhtpostEEMDcombinexp1OzSquarePseudo$instantfreq[,],"hhtpostEEMDcombinexp1
OzSquarePseudoinstantfreq.csv")

write.csv(hhtpostEEMDcombinexp2OzSquarePseudo$instantfreq[,],"hhtpostEEMDcombinexp2
OzSquarePseudoinstantfreq.csv")


write.csv(hhtpostEEMDcombinexp1OzSquarePseudo$energy,"hhtpostEEMDcombinexp1OzSqu
arePseudoenergy.csv")


write.csv(hhtpostEEMDcombinexp2OzSquarePseudo$energy,"hhtpostEEMDcombinexp2OzSqu
arePseudoenergy.csv")


write.csv(hhtpostEEMDcombinexp1OzDiamondMask$amplitude[,],"hhtpostEEMDcombinexp1O
zDiamondMaskamplitude.csv")


write.csv(hhtpostEEMDcombinexp2OzDiamondMask$amplitude[,],"hhtpostEEMDcombinexp2O
zDiamondMaskamplitude.csv")


write.csv(hhtpostEEMDcombinexp1OzDiamondMask$instantfreq[,],"hhtpostEEMDcombinexp1
OzDiamondMaskinstantfreq.csv")

write.csv(hhtpostEEMDcombinexp2OzDiamondMask$instantfreq[,],"hhtpostEEMDcombinexp2
OzDiamondMaskinstantfreq.csv")


write.csv(hhtpostEEMDcombinexp1OzDiamondMask$energy,"hhtpostEEMDcombinexp1OzDia
mondMaskenergy.csv")


write.csv(hhtpostEEMDcombinexp2OzDiamondMask$energy,"hhtpostEEMDcombinexp2OzDia
mondMaskenergy.csv")




write.csv(hhtpostEEMDcombinexp1OzDiamondPseudo$amplitude[,],"hhtpostEEMDcombinexp
1OzDiamondPseudoamplitude.csv")


write.csv(hhtpostEEMDcombinexp2OzDiamondPseudo$amplitude[,],"hhtpostEEMDcombinexp
2OzDiamondPseudoamplitude.csv")


write.csv(hhtpostEEMDcombinexp1OzDiamondPseudo$instantfreq[,],"hhtpostEEMDcombinexp
1OzDiamondPseudoinstantfreq.csv")

write.csv(hhtpostEEMDcombinexp2OzDiamondPseudo$instantfreq[,],"hhtpostEEMDcombinexp
2OzDiamondPseudoinstantfreq.csv")


write.csv(hhtpostEEMDcombinexp1OzDiamondPseudo$energy,"hhtpostEEMDcombinexp1OzD
iamondPseudoenergy.csv")


write.csv(hhtpostEEMDcombinexp2OzDiamondPseudo$energy,"hhtpostEEMDcombinexp2OzD
iamondPseudoenergy.csv")




write.csv(hhtpostEEMDcombinexp1T5SquareMask$amplitude[,],"hhtpostEEMDcombinexp1T5
SquareMaskamplitude.csv")
```

```
write.csv(hhtpostEEMDcombinexp2T5SquareMask$amplitude[,],"hhtpostEEMDcombinexp2T5
SquareMaskamplitude.csv")


write.csv(hhtpostEEMDcombinexp1T5SquareMask$instantfreq[,],"hhtpostEEMDcombinexp1T5
SquareMaskinstantfreq.csv")

write.csv(hhtpostEEMDcombinexp2T5SquareMask$instantfreq[,],"hhtpostEEMDcombinexp2T5
SquareMaskinstantfreq.csv")


write.csv(hhtpostEEMDcombinexp1T5SquareMask$energy,"hhtpostEEMDcombinexp1T5Squar
eMaskenergy.csv")


write.csv(hhtpostEEMDcombinexp2T5SquareMask$energy,"hhtpostEEMDcombinexp2T5Squar
eMaskenergy.csv")


write.csv(hhtpostEEMDcombinexp1T5SquarePseudo$amplitude[,],"hhtpostEEMDcombinexp1T
5SquarePseudoamplitude.csv")


write.csv(hhtpostEEMDcombinexp2T5SquarePseudo$amplitude[,],"hhtpostEEMDcombinexp2T
5SquarePseudoamplitude.csv")


write.csv(hhtpostEEMDcombinexp1T5SquarePseudo$instantfreq[,],"hhtpostEEMDcombinexp1
T5SquarePseudoinstantfreq.csv")

write.csv(hhtpostEEMDcombinexp2T5SquarePseudo$instantfreq[,],"hhtpostEEMDcombinexp2
T5SquarePseudoinstantfreq.csv")


write.csv(hhtpostEEMDcombinexp1T5SquarePseudo$energy,"hhtpostEEMDcombinexp1T5Squ
arePseudoenergy.csv")


write.csv(hhtpostEEMDcombinexp2T5SquarePseudo$energy,"hhtpostEEMDcombinexp2T5Squ
arePseudoenergy.csv")


write.csv(hhtpostEEMDcombinexp1T5DiamondMask$amplitude[,],"hhtpostEEMDcombinexp1T
5DiamondMaskamplitude.csv")
```

```
write.csv(hhtpostEEMDcombinexp2T5DiamondMask$amplitude[,],"hhtpostEEMDcombinexp2T
5DiamondMaskamplitude.csv")


write.csv(hhtpostEEMDcombinexp1T5DiamondMask$instantfreq[,],"hhtpostEEMDcombinexp1
T5DiamondMaskinstantfreq.csv")

write.csv(hhtpostEEMDcombinexp2T5DiamondMask$instantfreq[,],"hhtpostEEMDcombinexp2
T5DiamondMaskinstantfreq.csv")


write.csv(hhtpostEEMDcombinexp1T5DiamondMask$energy,"hhtpostEEMDcombinexp1T5Dia
mondMaskenergy.csv")


write.csv(hhtpostEEMDcombinexp2T5DiamondMask$energy,"hhtpostEEMDcombinexp2T5Dia
mondMaskenergy.csv")




write.csv(hhtpostEEMDcombinexp1T5DiamondPseudo$amplitude[,],"hhtpostEEMDcombinexp
1T5DiamondPseudoamplitude.csv")


write.csv(hhtpostEEMDcombinexp2T5DiamondPseudo$amplitude[,],"hhtpostEEMDcombinexp
2T5DiamondPseudoamplitude.csv")


write.csv(hhtpostEEMDcombinexp1T5DiamondPseudo$instantfreq[,],"hhtpostEEMDcombinexp
1T5DiamondPseudoinstantfreq.csv")

write.csv(hhtpostEEMDcombinexp2T5DiamondPseudo$instantfreq[,],"hhtpostEEMDcombinexp
2T5DiamondPseudoinstantfreq.csv")


write.csv(hhtpostEEMDcombinexp1T5DiamondPseudo$energy,"hhtpostEEMDcombinexp1T5Di
amondPseudoenergy.csv")


write.csv(hhtpostEEMDcombinexp2T5DiamondPseudo$energy,"hhtpostEEMDcombinexp2T5Di
amondPseudoenergy.csv")
```

```
write.csv(hhtpostEEMDexp1Ozdiamo$amplitude[,],"hhtpostEEMDexp1Ozdiamoamplitude.csv")

write.csv(hhtpostEEMDexp2Ozdiamo$amplitude[,],"hhtpostEEMDexp2Ozdiamoamplitude.csv")


write.csv(hhtpostEEMDexp1Ozdiamo$instantfreq[,],"hhtpostEEMDexp1Ozdiamoinstantfreq.csv
")

write.csv(hhtpostEEMDexp2Ozdiamo$instantfreq[,],"hhtpostEEMDexp2Ozdiamoinstantfreq.csv
")


write.csv(hhtpostEEMDexp1Ozdiamo$energy,"hhtpostEEMDexp1Ozdiamoenergy.csv")


write.csv(hhtpostEEMDexp2Ozdiamo$energy,"hhtpostEEMDexp2Ozdiamoenergy.csv")




write.csv(hhtpostEEMDexp1OzMask$amplitude[,],"hhtpostEEMDexp1OzMaskamplitude.csv")

write.csv(hhtpostEEMDexp2OzMask$amplitude[,],"hhtpostEEMDexp2OzMaskamplitude.csv")


write.csv(hhtpostEEMDexp1OzMask$instantfreq[,],"hhtpostEEMDexp1OzMaskinstantfreq.csv")

write.csv(hhtpostEEMDexp2OzMask$instantfreq[,],"hhtpostEEMDexp2OzMaskinstantfreq.csv")


write.csv(hhtpostEEMDexp1OzMask$energy,"hhtpostEEMDexp1OzMaskenergy.csv")


write.csv(hhtpostEEMDexp2OzMask$energy,"hhtpostEEMDexp2OzMaskenergy.csv")




write.csv(hhtpostEEMDexp1OzPseudomask$amplitude[,],"hhtpostEEMDexp1OzPseudomaska
mplitude.csv")

write.csv(hhtpostEEMDexp2OzPseudomask$amplitude[,],"hhtpostEEMDexp2OzPseudomaska
mplitude.csv")


write.csv(hhtpostEEMDexp1OzPseudomask$instantfreq[,],"hhtpostEEMDexp1OzPseudomaski
nstantfreq.csv")

write.csv(hhtpostEEMDexp2OzPseudomask$instantfreq[,],"hhtpostEEMDexp2OzPseudomaski
nstantfreq.csv")


write.csv(hhtpostEEMDexp1OzPseudomask$energy,"hhtpostEEMDexp1OzPseudomaskenergy.
csv")

write.csv(hhtpostEEMDexp2OzPseudomask$energy,"hhtpostEEMDexp2OzPseudomaskenergy.
csv")




write.csv(hhtpostEEMDexp1Ozsqua$amplitude[,],"hhtpostEEMDexp1Ozsquaamplitude.csv")

write.csv(hhtpostEEMDexp2Ozsqua$amplitude[,],"hhtpostEEMDexp2Ozsquaamplitude.csv")


write.csv(hhtpostEEMDexp1Ozsqua$instantfreq[,],"hhtpostEEMDexp1Ozsquainstantfreq.csv")

write.csv(hhtpostEEMDexp2Ozsqua$instantfreq[,],"hhtpostEEMDexp2Ozsquainstantfreq.csv")


write.csv(hhtpostEEMDexp1Ozsqua$energy,"hhtpostEEMDexp1Ozsquaenergy.csv")

write.csv(hhtpostEEMDexp2Ozsqua$energy,"hhtpostEEMDexp2Ozsquaenergy.csv")




write.csv(hhtpostEEMDexp1T5diamo$amplitude[,],"hhtpostEEMDexp1T5diamoamplitude.csv")

write.csv(hhtpostEEMDexp2T5diamo$amplitude[,],"hhtpostEEMDexp2T5diamoamplitude.csv")


write.csv(hhtpostEEMDexp1T5diamo$instantfreq[,],"hhtpostEEMDexp1T5diamoinstantfreq.csv
")

write.csv(hhtpostEEMDexp2T5diamo$instantfreq[,],"hhtpostEEMDexp2T5diamoinstantfreq.csv
")


write.csv(hhtpostEEMDexp1T5diamo$energy,"hhtpostEEMDexp1T5diamoenergy.csv")
write.csv(hhtpostEEMDexp2T5diamo$energy,"hhtpostEEMDexp2T5diamoenergy.csv")




write.csv(hhtpostEEMDexp1T5Mask$amplitude[,],"hhtpostEEMDexp1T5Maskamplitude.csv")

write.csv(hhtpostEEMDexp2T5Mask$amplitude[,],"hhtpostEEMDexp2T5Maskamplitude.csv")


write.csv(hhtpostEEMDexp1T5Mask$instantfreq[,],"hhtpostEEMDexp1T5Maskinstantfreq.csv")

write.csv(hhtpostEEMDexp2T5Mask$instantfreq[,],"hhtpostEEMDexp2T5Maskinstantfreq.csv")

write.csv(hhtpostEEMDexp1T5Mask$energy,"hhtpostEEMDexp1T5Maskenergy.csv")

write.csv(hhtpostEEMDexp2T5Mask$energy,"hhtpostEEMDexp2T5Maskenergy.csv")




write.csv(hhtpostEEMDexp1T5Pseudomask$amplitude[,],"hhtpostEEMDexp1T5Pseudomaska
mplitude.csv")

write.csv(hhtpostEEMDexp2T5Pseudomask$amplitude[,],"hhtpostEEMDexp2T5Pseudomaska
mplitude.csv")


write.csv(hhtpostEEMDexp1T5Pseudomask$instantfreq[,],"hhtpostEEMDexp1T5Pseudomaski
nstantfreq.csv")

write.csv(hhtpostEEMDexp2T5Pseudomask$instantfreq[,],"hhtpostEEMDexp2T5Pseudomaski
nstantfreq.csv")


write.csv(hhtpostEEMDexp1T5Pseudomask$energy,"hhtpostEEMDexp1T5Pseudomaskenergy.
csv")
```

```
write.csv(hhtpostEEMDexp2T5Pseudomask$energy,"hhtpostEEMDexp2T5Pseudomaskenergy.
csv")
```

```
write.csv(hhtpostEEMDexp1T5squa$amplitude[,],"hhtpostEEMDexp1T5squaamplitude.csv")

write.csv(hhtpostEEMDexp2T5squa$amplitude[,],"hhtpostEEMDexp2T5squaamplitude.csv")


write.csv(hhtpostEEMDexp1T5squa$instantfreq[,],"hhtpostEEMDexp1T5squainstantfreq.csv")

write.csv(hhtpostEEMDexp2T5squa$instantfreq[,],"hhtpostEEMDexp2T5squainstantfreq.csv")


write.csv(hhtpostEEMDexp1T5squa$energy,"hhtpostEEMDexp1T5squaenergy.csv")


write.csv(hhtpostEEMDexp2T5squa$energy,"hhtpostEEMDexp2T5squaenergy.csv")
```

```
AmpostEEMDOz<-    spss.get("E:/Geral/VITOR    PEREIRA/hhtpostEEMD/AmpostEEMDOz.sav",
lowernames=FALSE, datevars = NULL,
use.value.labels = TRUE, to.data.frame = TRUE,
max.value.labels = Inf, force.single=TRUE,
allow=NULL, charfactor=FALSE)


AmpostEEMDT5<-    spss.get("E:/Geral/VITOR    PEREIRA/hhtpostEEMD/AmpostEEMDT5.sav",
lowernames=FALSE, datevars = NULL,
use.value.labels = TRUE, to.data.frame = TRUE,
max.value.labels = Inf, force.single=TRUE,
allow=NULL, charfactor=FALSE)
```

```
HzpostEEMDOz<-     spss.get("G:/Geral/VITOR     PEREIRA/hhtpostEEMD/HzpostEEMDOz.sav",
lowernames=FALSE, datevars = NULL,
use.value.labels = TRUE, to.data.frame = TRUE,
max.value.labels = Inf, force.single=TRUE,
allow=NULL, charfactor=FALSE)


HzpostEEMDT5<-     spss.get("G:/Geral/VITOR     PEREIRA/hhtpostEEMD/HzpostEEMDT5.sav",
lowernames=FALSE, datevars = NULL,
use.value.labels = TRUE, to.data.frame = TRUE,
max.value.labels = Inf, force.single=TRUE,
allow=NULL, charfactor=FALSE)



#beckmw (2013, February 5). Collinearity and stepwise VIF selection [Blog post].
#Retrieved from #https://beckmw.wordpress.com/2013/02/05/collinearity-and-stepwi
#se-vif-selectio/ and revisions from https://gist.github.com/fawda123/4717702



require(MASS)
require(clusterGeneration)


vif_func<-function(in_frame,thresh=10,trace=T,...){

  require(fmsb)

  if(class(in_frame) != 'data.frame') in_frame<-data.frame(in_frame)

  #get initial vif value for all comparisons of variables
  vif_init<-NULL
  var_names <- names(in_frame)
  for(val in var_names){
      regressors <- var_names[-which(var_names == val)]
      form <- paste(regressors, collapse = '+')
      form_in <- formula(paste(val, '~', form))
      vif_init<-rbind(vif_init, c(val, VIF(lm(form_in, data = in_frame, ...))))
      }
  vif_max<-max(as.numeric(vif_init[,2]))

  if(vif_max < thresh){
    if(trace==T){ #print output of each iteration
        prmatrix(vif_init,collab=c('var','vif'),rowlab=rep('',nrow(vif_init)),quote=F)
        cat('\n')
        cat(paste('All variables have VIF < ', thresh,', max VIF ',round(vif_max,2), sep=''),'\n\n')
        }
    return(var_names)
    }
  else{

    in_dat<-in_frame

    #backwards selection of explanatory variables, stops when all VIF values are below
'thresh'
    while(vif_max >= thresh){

      vif_vals<-NULL
      var_names <- names(in_dat)

      for(val in var_names){
        regressors <- var_names[-which(var_names == val)]
        form <- paste(regressors, collapse = '+')
        form_in <- formula(paste(val, '~', form))
        vif_add<-VIF(lm(form_in, data = in_dat, ...))
        vif_vals<-rbind(vif_vals,c(val,vif_add))
        }
      max_row<-which(vif_vals[,2] == max(as.numeric(vif_vals[,2])))[1]

      vif_max<-as.numeric(vif_vals[max_row,2])

      if(vif_max<thresh) break

      if(trace==T){ #print output of each iteration
        prmatrix(vif_vals,collab=c('var','vif'),rowlab=rep('',nrow(vif_vals)),quote=F)
        cat('\n')
        cat('removed: ',vif_vals[max_row,1],vif_max,'\n\n')
        flush.console()
        }

      in_dat<-in_dat[,!names(in_dat) %in% vif_vals[max_row,1]]

      }

    return(names(in_dat))

    }

  }
```

```
source.with.encoding('D:/vif_func.txt', encoding='UTF-8')
```

if D:/ is the location of the source file vif_func.txt

```
vif_func(AmpostEEMDOz)
vifAmpostEEMDOz
```

```
vif_func(AmpostEEMDT5)
vifAmpostEEMDT5


vif_func(HzpostEEMDOz)
vifHzpostEEMDOz



vif_func(HzpostEEMDT5)
vifHzpostEEMDT5
```

```
source.with.encoding('D:/vif_func.txt', encoding='UTF-8')
```

if D:/ is the location of the source file vif_func.txt

```
vif_func(colli_AmpostEEMDOz)
```

All variables have VIF < 10, max VIF 8.83

```
vif_func(colli_AmpostEEMDT5)
```

All variables have VIF < 10, max VIF 9.06

```
vif_func(colli_HzpostEEMDOz)
```

All variables have VIF < 10, max VIF 7.02

```
vif_func(colli_HzpostEEMDT5)
```

All variables have VIF < 10, max VIF 8.47

```
pls_AmpostEEMDOz<-    spspss.get("F:/Geral/VITOR    PEREIRA/PLS/colli_AmpostEEMDOz.sav",
lowernames=FALSE, datevars = NULL,
use.value.labels = TRUE, to.data.frame = TRUE,
max.value.labels = Inf, force.single=TRUE,
allow=NULL, charfactor=FALSE)



pls_AmpostEEMDT5<-     spss.get("F:/Geral/VITOR     PEREIRA/PLS/colli_AmpostEEMDT5.sav",
lowernames=FALSE, datevars = NULL,
use.value.labels = TRUE, to.data.frame = TRUE,
max.value.labels = Inf, force.single=TRUE,
allow=NULL, charfactor=FALSE)




pls_HzpostEEMDOz<-      spss.get("F:/Geral/VITOR      PEREIRA/PLS/colli_HzpostEEMDOz.sav",
lowernames=FALSE, datevars = NULL,
use.value.labels = TRUE, to.data.frame = TRUE,
max.value.labels = Inf, force.single=TRUE,
allow=NULL, charfactor=FALSE)



pls_HzpostEEMDT5<-      spss.get("F:/Geral/VITOR      PEREIRA/PLS/colli_HzpostEEMDT5.sav",
lowernames=FALSE, datevars = NULL,
use.value.labels = TRUE, to.data.frame = TRUE,
max.value.labels = Inf, force.single=TRUE,
allow=NULL, charfactor=FALSE)











#Mevik, Bjørn-Helge and Wehrens, Ron. (2007). The pls Package: Principal
#Component and Partial Least Squares Regression in R. Journal of Statistical
#Software, 18(2), 1-23. R package version 2.5-0.


library("pls")

set.seed(59)
plsFitAmpostEEMDOz<-plsr(Exper~.,data=pls_AmpostEEMDOz,validation="CV")
```

```
summary(plsFitAmpostEEMDOz)
predictionsAmpostEEMDOz <- predict(plsFitAmpostEEMDOz, PLSR_Test_Derivação_Oz,
ncomp=23)
validationplot(plsFitAmpostEEMDOz, val.type="RMSEP")
pls.RMSEPAmpostEEMDOz<-RMSEP(plsFitAmpostEEMDOz, estimate="CV")
plot(pls.RMSEPAmpostEEMDOz, main="RMSEP PLS Exper", xlab="components")
minRMSEP<-which.min(pls.RMSEPAmpostEEMDOz$val)
points(minRMSEP,min(pls.RMSEPAmpostEEMDOz$val), pch=1, col="red", cex=1.5)



set.seed(59)
plsFitAmpostEEMDT5<-plsr(Exper~.,data=pls_AmpostEEMDT5,validation="CV")
summary(plsFitAmpostEEMDT5)
predictionsAmpostEEMDT5 <- predict(plsFitAmpostEEMDT5, PLSR_Test_Derivação_T5,
ncomp=19)
validationplot(plsFitAmpostEEMDT5, val.type="RMSEP")
pls.RMSEPAmpostEEMDT5<-RMSEP(plsFitAmpostEEMDT5, estimate="CV")
plot(pls.RMSEPAmpostEEMDT5, main="RMSEP PLS Exper", xlab="components")
minRMSEP<-which.min(pls.RMSEPAmpostEEMDT5$val)
points(minRMSEP,min(pls.RMSEPAmpostEEMDT5$val), pch=1, col="red", cex=1.5)




library("pls")

set.seed(59)
plsFitHzpostEEMDOz<-plsr(Exper~.,data=pls_HzpostEEMDOz,validation="CV")
summary(plsFitHzpostEEMDOz)
predictionsHzpostEEMDOz <- predict(plsFitHzpostEEMDOz, PLSR_Test_Derivação_Oz,
ncomp=40)
validationplot(plsFitHzpostEEMDOz, val.type="RMSEP")
pls.RMSEPHzpostEEMDOz<-RMSEP(plsFitHzpostEEMDOz, estimate="CV")
plot(pls.RMSEPHzpostEEMDOz, main="RMSEP PLS Exper", xlab="components")
minRMSEP<-which.min(pls.RMSEPHzpostEEMDOz$val)
points(minRMSEP,min(pls.RMSEPHzpostEEMDOz$val), pch=1, col="red", cex=1.5)



set.seed(59)
plsFitHzpostEEMDT5<-plsr(Exper~.,data=pls_HzpostEEMDT5,validation="CV")
summary(plsFitHzpostEEMDT5)
predictionsHzpostEEMDT5 <- predict(plsFitHzpostEEMDT5, PLSR_Test_Derivação_T5,
ncomp=43)
validationplot(plsFitHzpostEEMDT5, val.type="RMSEP")
pls.RMSEPHzpostEEMDT5<-RMSEP(plsFitHzpostEEMDT5, estimate="CV")
plot(pls.RMSEPHzpostEEMDT5, main="RMSEP PLS Exper", xlab="components")
minRMSEP<-which.min(pls.RMSEPHzpostEEMDT5$val)
points(minRMSEP,min(pls.RMSEPHzpostEEMDT5$val), pch=1, col="red", cex=1.5)
```

```
#Del Re, A. C. (2013). Compute Effect Sizes. R package ‘compute.es’. R package
#version 0.2-2.


library("compute.es")


postIMF 3 combined Diamond Pseudo Oz instantaneous amplitude
mes(250.07726400000004,141.84749921304348,221.23649381824532,116.99762855770675,
230,230)

postIMF 2 combined Diamond Pseudo T5 instantaneous amplitude
mes(231.50092008702,148.87988807983,193.047258710684,116.233060388419,230,230)

postIMF 1 Mask Oz instantaneous frequency
mes(0.2922027197375933,0.22709554991124395,0.1440134078351405,0.150697550171260
8,230,230)

postIMF 5 Mask T5 instantaneous frequency
mes(0.017258188931635346,          0.012934933405625387,          0.005633557981819243,
0.004605120575712734,230,230)




#Afanador, Nelson Lee, Tran, Thanh and Blanchet, Lionel (2016). Package
#‘mvdalab’. R package version 1.0.


library("mvdalab")

modAmpostEEMDOz <- plsFit(Exper~., scale = TRUE, data = pls_AmpostEEMDOz,
                      ncomp = 23, validation = "oob", boots = 1000)

smc((modAmpostEEMDOz),corrected = T)
corrected=T
whether there should be a correction of 1st order auto-correlation in the residuals

plot(smc((modAmpostEEMDOz),corrected = T))




modAmpostEEMDT5 <- plsFit(Exper~., scale = TRUE, data = pls_AmpostEEMDT5,
                      ncomp = 19, validation = "oob", boots = 1000)

smc((modAmpostEEMDT5),corrected = T)
corrected=T
whether there should be a correction of 1st order auto-correlation in the residuals
```

```
plot(smc((modAmpostEEMDT5),corrected = T))
```

```
library("mvdalab")

modHzpostEEMDOz <- plsFit(Exper~., scale = TRUE, data = pls_HzpostEEMDOz,
                    ncomp = 23, validation = "oob", boots = 1000)

smc((modHzpostEEMDOz),corrected = T)
corrected=T
whether there should be a correction of 1st order auto-correlation in the residuals

plot(smc((modHzpostEEMDOz),corrected = T))
```

```
modHzpostEEMDT5 <- plsFit(Exper~., scale = TRUE, data = pls_HzpostEEMDT5,
                    ncomp = 19, validation = "oob", boots = 1000)

smc((modHzpostEEMDT5),corrected = T)
corrected=T
whether there should be a correction of 1st order auto-correlation in the residuals

plot(smc((modHzpostEEMDT5),corrected = T))
```

```
#Lakens, D. (2013). Calculating and reporting effect sizes to facilitate cumulative
#science: A practical primer for t-tests and ANOVAs. Frontiers in Psychology,
#4:863. This is version 3.2. For updates, check:
#http://opensciencegramework.org/project/ixGcd
```

```
#modAmpostEEMDOz significant multivariate correlation statistic (smc)
```

ηG² for (A; between) X (P; within)			
Main Between	Main Within	Interaction	
SSA	SSP	SSPA	
11399094.094	48269330.112	49600420.754	
SSs/A	SSPs/A		
124030553.706	150787769.532		
MSA	MSP	MSPA	
11399094.094	40335816.757	41448130.271	

MSs/A	MSPs/A		
270809.069	275118.771		
dfA	dfP	dfPA	
1	1.197	1.197	
F-ratio	F-ratio	F-ratio	
42.0927	146.612	150.655	
ηG^2	ηG^2	ηG^2	
0.03983	0.14940	0.15289	
ηp^2	ηp^2	ηp^2	
0.08417	0.24249	0.24752	
η^2	η^2	η^2	
0.08417	0.19412	0.19947	

#modAmpostEEMDT5 significant multivariate correlation statistic (smc)

ηG^2 for (A; between) X (P; within)			
Main Between	Main Within	Interaction	
SSA	SSP	SSPA	
735365.753	22210617.419	9050816.246	
SSs/A	SSPs/A		
18344418.588	2741647.728		
MSA	MSP	MSPA	
735365.753	22210617.419	9050816.246	
MSs/A	MSPs/A		
40053.316	5986.130		
dfA	dfP	dfPA	
1	1.000	1.000	
F-ratio	F-ratio	F-ratio	
18.3597	3710.35	1511.96	
ηG^2	ηG^2	ηG^2	
0.03370	0.51299	0.30032	
ηp^2	ηp^2	ηp^2	
0.03854	0.89012	0.76751	
η^2	η^2	η^2	
0.03854	0.65319	0.26618	

#modHzpostEEMDOz significant multivariate correlation statistic (smc)

ηG^2 for (A; between) X (P; within)			
Main Between	Main Within	Interaction	
SSA	SSP	SSPA	
.007	.022	.015	
SSs/A	SSPs/A		
.112	.135		
MSA	MSP	MSPA	
.007	.007	.005	
MSs/A	MSPs/A		
.000	.000		
dfA	dfP	dfPA	
1	3.111	3.111	
F-ratio	F-ratio	F-ratio	
30.046	73.5862	49.7628	
ηG^2	ηG^2	ηG^2	
0.02884	0.08082	0.05612	
ηp^2	ηp^2	ηp^2	
0.06156	0.13843	0.09800	
η^2	η^2	η^2	
0.06156	0.12658	0.08560	

#modHzpostEEMDT5 significant multivariate correlation statistic (smc)

ηG^2 for (A; between) X (P; within)			
Main Between	Main Within	Interaction	
SSA	SSP	SSPA	
.048	.543	.052	
SSs/A	SSPs/A		
.158	.563		
MSA	MSP	MSPA	
.048	.218	.021	
MSs/A	MSPs/A		
.000	.000		
dfA	dfP	dfPA	
1	2.493	2.493	
F-ratio	F-ratio	F-ratio	
138.338	441.582	42.0154	
ηG^2	ηG^2	ηG^2	
0.06223	0.42934	0.06680	
ηp^2	ηp^2	ηp^2	
0.23198	0.49087	0.08403	
η^2	η^2	η^2	
0.23198	0.46897	0.04462	