```
#EAR Calculator
#Author
#Joe Swintek 3-3-2017


################################################################################################
######
################################################################################################
######
##                                          Calculation
Functions                                        ##
################################################################################################
######
################################################################################################
######
#This file contains all the functions used in calculating EAR values
#This is the back end to the that handles calculations



################################################################################################
######
#
Global                                                          #
################################################################################################
######
#Global values used in the calculation algorithms


#Warnings.EAR
    #List data structure [[1]] contains it's length will each element is a warning to the user
#ChemMasterList.EAR
    #Data Set containing assays for all the chemical
#ChemMasterListFlat.EAR
    #Data Set containing assays for all the chemical flattened into a matrix of chemical by
    assay
#ChemInfoList.EAR
    #A data set containing the Names, CAS numbers, molecular weights, and Formulas of the
    chemicals

#PlottingCat.EAR
    #Global Vector that stores the values for the plotting categories
        #4 Categories 1: dot, 2: blue, 3:green, 4:red (
        #3 numbers indicating the upper bound of each category

################################################################################################
######
#                                       Function
List                                                    #
################################################################################################
######

#RunEARAll
```

```
    #ConvertChemData
        #FilterChemData
            #Convert2ug
        #Find Detects
    #CalculatedAllEAR
        #CalculateEAR3D
            #GetAC50All
            #GetAC50ListAll
            #CalculateEARAll
        #GetMW
        #Convert2uM


################################################################################################
######
#                                  Function
Information                                                  #
################################################################################################
######
#RunEARAll
    #Main Function call for EAR calculation
    #Command line only, not called from GUI
    #Note '.' notion designates the use of a global

    #Inputs
        #File
            #File containing chemical survey data, the format has to be specific

    #OutPut
        #Creates Global Warnings.EAR
            #Contains all the warning messages
            #Contains $N
                #The number of warning messages
        #OutputEARAll
            #List containing:
                #CleanedData
                    #Filtered chemistry data with all of the detect limit rules applied
                #ResultsEAR
        #ResultsEAR
            #List containing:
                #EARCube
                    #A rank 3 tensor containing EAR Values for in a Site by Chemical by Assay
                #ChemicalNames
                    #A number of chemicals by 2 character vector that contain chemical names [
                    ,1] and CAS numbers [ ,2]
                #SiteData
                    #A data frame containing:
                        #USGS STATION NAME
                            #The Full Name of the Station
                        #FIELD NAME
                            #An abbreviated name of the station
                        #DATE COLLECTED
                            #The date of collection
```

```
                    #TIME COLLECTE
                        #The time the data is collected
                    #SAMPLE RECORD NUMBER
                        #A number unique to the collected sample
                    #Units
            #MW
                #A character vector that contains the molecular weights of each chemical as
                its entries.
                #The names of the vector are the chemicals themselves
            #AC50s
                #A number of assays by number of chemicals matrix
                #The row names are the names of the assays
            #UnderLimit
                #A Site by Chemical by Assay tensot containing a 1 if the chemical is under
                the detection limit and 0 otherwise


    #Called By
        #NOTHING
    #Calls Directly \ Indirectly
        #ConvertChemData
            #FilterChemData
                #Convert2ug
            #Find Detects
        #CalculateEAR
            #CalculateEAR3D
                #CalculateEAR3D
                #GetAC50All
                #GetAC50ListAll
                #CalculateEARAll
            #GetMW
            #Convert2uM


################################################################################################
#############################
#ConvertChemData
    #Converts formatted Chemical data into EAR comparable Data

    #Inputs
        #Data
            #A Read in MED EAR formatted exposure data

    #OutPut
        #AllCleanData
            #List containing:
                #SiteIdData
                    #Data set containing site information
                #ChemData
                    #Data set containing chemical information (concentrations)
                #UNITS
                    #Units of chemical concentrations
                #ChemNames
```

```
                    #names of chemical
                #CasRN
                    #Cas numbers for each chemical
                #UnderLimit
                    #A matrix containing a 1 if the sit/chemical combination is under the
                    detection limit

    #Called By
        #RunEARAll
        #StartGUI.EAR (RunButton.EAR button in GUI)
    #Calls Directly \ Indirectly
        #FilterChemData
            #Convert2ug
        #Find Detects

##############################################################################################
############################
#FilterChemData
    #Takes a column from the the ChemData data set within the ConvertChemData function and
    filters it (see function for details)

    #Inputs
        #ChemColmn
            #A columns of chemistry data from ConvertChemData

    #OutPut
        #c(ChemColmn[1],MeasuredValues)
            #Column vector containing name of chemical then filtered and converted exposure
            data for each site

    #Called By
        #ConvertChemData
    #Calls Directly \ Indirectly
        #Convert2ug

##############################################################################################
############################
#CalculatedAllEAR
    #Calculates EAR on all possible Assay

    #Inputs
        #CleanData
            #Data cleaned by ConvertChemData
        #BOOL
            #Bool to use the Cytotoxic values

    #OutPut
        #ResultsEAR
            #List containing:
                #EARCube
                    #A rank 3 tensor containing EAR Values for in a Site by Chemical by Assay
                #ChemicalNames
                    #A number of chemicals by 2 character vector that contain chemical names [
```

```
                ,1] and CAS numbers [ ,2]
            #SiteData
                #A data frame containing:
                    #USGS STATION NAME
                        #The Full Name of the Station
                    #FIELD NAME
                        #An abbreviated name of the station
                    #DATE COLLECTED
                        #The date of collection
                    #TIME COLLECTE
                        #The time the data is collected
                    #SAMPLE RECORD NUMBER
                        #A number unique to the collected sample
                    #Units
            #MW
                #A character vector that contains the molecular weights of each chemical as
                its entries.
                #The names of the vector are the chemicals themselves
            #AC50s
                #A number of assays by number of chemicals matrix
                #The row names are the names of the assays
            #UnderLimit
                #A Site by Chemical by Assay tensot containing a 1 if the chemical is under
                the detection limit and 0 otherwise

    #Called By
        #RunEARAll
        #StartGUI.EAR (RunButton.EAR button in GUI)
    #Calls Directly \ Indirectly
        #CalculateEAR3D
            #GetAC50All
            #GetAC50ListAll
            #CalculateEARAll
        #GetMW
        #Convert2uM

##########################################################################################
#########################
#CalculateEAR3D
    #Calculates EAR values and places them in a site by chemical by assay rank 3 tensor

    #Inputs
        #ChemDataMW
            #A character vector that contains the molecular weights of each chemical as its
            entries.
            #The names of the vector are the chemicals themselves
        #NameCas
            #A number of chemicals by 2 character vector that contain chemical names [ ,1] and
            CAS numbers [ ,2]
        #ID
            #A string contain 'CAS' or 'name'
            #Indicates which part of NameCAS is used
```

```
    #OutPut
        #EARInformation
            #A list containing two values
                #EARCube
                    #A rank 3 tensor containing EAR Values for in a Site by Chemical by Assay
                #AC50Mat
                    #A number of assays by number of chemicals matrix
                    #The row names are the names of the assays

    #Called By
        #CalculateEAR3D

    #Calls Directly \ Indirectly
        #GetAC50All
        #GetAC50ListAll
        #CalculateEARAll

################################################################################################
############################
#GetAC50All
    #gets the AC 50s for all assays for all chemicals in the data set

    #Inputs
        #Name
            #A value with chemical name and CAS name
        #NAssay
            #The total number of Assays
        #ChemMasterListFlat.EAR
            #Global, Contains all the Assays for every chemical

    #OutPut
        #AC50s
            #Vector of AC50 (NA is capacitive or missing) for every possible assay pertaining
            to the chemical in Name

    #Called By
        #CalculateEAR3D

    #Calls Directly \ Indirectly
        #NOTHING
################################################################################################
############################
#GetAC50ListAll
    #Arranges the AC 50 to be a NSite by NChem matrix as a list

    #Inputs
        #Assay
            #The assay being used
        #AC50Mat
            #output from GetAC50All
                #Matrix of AC50 (NA is capacitive or missing) for every possible assay
                pertaining to the chemical in Name
        #AssayList
```

```
            #A vector containing all possible assays
        #SiteN
            #Number of Sites

    #OutPut
        #AC50Mat
            ##A number of assays by number of chemicals matrix
            #The row names are the names of the assays

    #Called By
        #CalculateEAR3D

    #Calls Directly \ Indirectly
        #NOTHING

##############################################################################################
############################
#CalculateEARAll
    #Calculates EAR for all Assays

    #Inputs
        #AC50Mat
            #Site by Chemical matrix of AC50's for 1 assay
        #ChemDataMW
            #Chemistry data converted to uMol

    #OutPut
        #Site by Chemical matrix of EARValues for 1 assay

    #Called By
        #CalculateEAR3D

    #Calls Directly \ Indirectly
        #NOTHING

##############################################################################################
############################
#FindDetects
    #Identifies the values in the data set that under the detection limit
    #Meant to be called using the apply function

    #Inputs
        #ChemColmn
            #A column corresponding the the chemical entries for all sites from the original
            data file

    #OutPut
        #A vector of 1's and 0's
        #1 corresponds to an observation under the detection limit
        #0 corresponds to an observed value

    #Called By
        #ConvertChemData
```

```
    #Calls Directly \ Indirectly
        #NOTHING

##############################################################################################################
#############################
#GetMW
    #gets the molecular weight of each chemical

    #Inputs
        #Chemical
            #The name or CAS number of a chemical
        #ID
            #Tells the function whether chemical is a Name or CAS
        #Uses Global ChemInfoList.EAR

    #OutPut
        #Chem1Info$Mol
            #The molecular weight of a chemical

    #Called By
        #CalculateEAR
    #Calls Directly \ Indirectly
        #NOTHING

##############################################################################################################
#############################
#Convert2uM
    #converts numbers in Vec units to uM

    #Inputs
        #Vec
            #Vector of concentrations to convert to uM, assumes Vec[1] is the MW
        #Units
            #Units concentrations are in can be 'ug/L', 'ng/L' or 'MW'

    #OutPut
        #c(MW,Out)
            #Vector [1] is the MW the rest are uM of the chemicals

    #Called By
        #CalculateEAR
    #Calls Directly \ Indirectly
        #NOTHING

##############################################################################################################
#############################
#AboveThreshold  (1-6-2016)


    #Inputs #Threshold
            #Value to compare results against
        #ResultsEAR
```

```
            #Results from CalculatedAllEAR

    #OutPut
        #OutData
            #Data containing site, chemical, and assay information for EARs above the threshold

    #Called By
        #PlotViewerGUI.EAR      [ExportTheshButton]
    #Calls Directly \ Indirectly
        #NOTHING


################################################################################################
######
#
Functions                                                    #
################################################################################################
######
RunEARAll<-function(File){
#Main Function call for EAR calculation
    #File is the name of the file with the survey data
    # '.' notion designates the use of a global

#Start Box of warning Messages
    Warnings.EAR<<-list()
    Warnings.EAR$N<<-1

#Read File
    Data<-read.csv(File,stringsAsFactors=FALSE,header=FALSE)
#Clean the Data
    CleanedData<-ConvertChemData(Data)
#Calculate EAR
    ResultsEARAll<-CalculatedAllEAR(CleanedData,FALSE)


return(ResultsEARAll)
}


################################################################################################
###########
ConvertChemData<-function(Data){
#Converts formatted Chemical data into EAR comparable Data
#Data is a comparable formate R Data structure which assumes:
    #First row contains a chemical name
    #Second row contains a CAS number with ? or - between numbers
    #Third row contain concentrations units or are header observation identifiers
    #Columns that have observation identifier in the third row are identified by '' (an empty
    string) in the first rows
    #'Sample record number'  is the unique identifier for a site
        #Removes all rows without a record number
```

```
#Attain identifiers
    IdColmn=which(Data[1,]=='')
#Compensate for capitalization differences
    Data[3,IdColmn]<-toupper(Data[3,IdColmn])
#Filter out non-sites
    IdUnique<-which(Data[3,IdColmn]=='SAMPLE RECORD NUMBER')
    NotASite<-which(Data[ ,IdUnique]=='')
    NotASite<-NotASite[-which(NotASite<=2)]
    if (length(NotASite)>0){
        Data<-Data[-NotASite, ]
    }
#Attain a separate data sets of site Ids and chem information
    SiteIdData<-Data[ ,IdColmn]
    ChemData<-Data[ ,-IdColmn]
#Clean up site ID data
    SiteIdDataHeader<-SiteIdData[3, ]
    SiteIdData<-SiteIdData[-{1:3}, ]
    colnames(SiteIdData)<-SiteIdDataHeader
#Fix the string containing CAS numbers
    CAS<-gsub('[?]','-',ChemData[2, ])
#Filter the ChemData
    options(warn=-1) #Turns off warns as there will be some
        UnderLimit<-apply(ChemData,2,FindDetects)
        UnderLimit<-matrix(apply(UnderLimit[2:dim(UnderLimit)[1], ],2,as.numeric),ncol=dim(
        UnderLimit)[2])
        ChemDataUpdate<-apply(ChemData,2,FilterChemData)

    options(warn=0)
    ChemNames<-ChemDataUpdate[1, ]
#Merge and Clean up data set
    ChemDataFrame<-as.data.frame(apply(ChemDataUpdate[-1, ],2,as.numeric))
    colnames(ChemDataFrame)<-ChemNames
#Add units and complete the data set
    UNITS<-'ug/L'
    SiteIdData<-cbind(SiteIdData,UNITS)
    SiteIdData$UNITS<-as.character(SiteIdData$UNITS)
    FilteredData<-cbind(SiteIdData,ChemDataFrame)
#Get data structure
    AllCleanData<-list('SiteData'=SiteIdData,'ChemData'=ChemDataFrame,'UNITS'=UNITS,'ChemNames'=
    ChemNames,'CasRN'= CAS,'UnderLimit'=UnderLimit)
return(AllCleanData)
}

##############################################################################################
############
CalculatedAllEAR<-function(CleanedData,BOOL){
#Calculates EAR on all possible Assay

#Apply the Bool
    if (BOOL==TRUE){
        ChemMasterListFlat.EAR<-ChemMasterListFlatCytotox.EAR  #3-21-2016
    }
```

```
#Step 1 Extract Information from the Data Structure
    NameCas<-cbind(CleanedData$ChemNames,CleanedData$CasRN)
    ChemData<-CleanedData$ChemData
    Units<-CleanedData$UNITS


#Step 2 get MW
    MW<-apply(NameCas,1,GetMW,'CAS')
    ChemDataMW<-rbind(MW,ChemData)

#Step 3 Covert to uM
    ChemDataMW<-apply(ChemDataMW,2,Convert2uM,Units)

#Step 4 Covert to EAR
    EARInformation<-CalculateEAR3D(ChemDataMW,NameCas,'CAS',BOOL)

    ResultsEAR<-list('EARCube'=EARInformation$EARCube,'ChemicalNames'=NameCas,'SiteData'=
    CleanedData$SiteData,
                     'MW'=ChemDataMW[1, ],'AC50s'=EARInformation$AC50s,'UnderLimit'=CleanedData$
                     UnderLimit)



return(ResultsEAR)
}

##########################################################################################
############
CalculateEAR3D<-function(ChemDataMW,NameCas,ID,BOOL){
#Calculate EAR on all Assays and for a Data Set
#ChemDataMW is pre-calculated data as MW
#Name CAS is link with chemical names and CAS in the same order of column Names of ChemDataMW
#ID is what is used as an identifier

#Apply Crytox
    if (BOOL==TRUE){
        ChemMasterListFlat.EAR<-ChemMasterListFlatCytotox.EAR  #3-21-2016
    }
#Get Matrix of AC 50 for all used chemicals for all Assays
    NAssay<-dim(ChemMasterListFlat.EAR)[1]
    if (ID=='name'){
        Names<-NameCas[ ,1]
    }
    if (ID=='CAS'){
        Names<-NameCas[ ,2]
    }
    AssayList<-rownames(ChemMasterListFlat.EAR)
    NSite<-dim(ChemDataMW)[1]-1
    AC50Mat<-sapply(Names,GetAC50All,NAssay,BOOL)
    AC50List<-lapply(AssayList,GetAC50ListAll,AC50Mat,AssayList,NSite)

#Convert to a list for  NSite,NChemical with NAssay elements
    EARList<-lapply(AC50List,CalculateEARAll,ChemDataMW)
```

```
#Convert to a cube of data
    EARCube<-array(unlist(EARList), dim = c(nrow(EARList[[1]]), ncol(EARList[[1]]), length(
    EARList)))
    EARInformation<-list('EARCube'=EARCube,'AC50s'=AC50Mat)
return(EARInformation)
}

##########################################################################################
########
GetMW<-function(Chemical,ID){
#Gets the MW of a Chemical
#Uses Global ChemInfoList.EAR
    NameOut<-paste(Chemical[1],Chemical[2])

    if (ID=='Name'){#Name may be subject to white space thus not recommended
        Chem1Info<-ChemInfoList.EAR[which(ChemInfoList.EAR$Name==Chemical[1]), ]
    }
    if (ID=='CAS'){
        Chem1Info<-ChemInfoList.EAR[which(ChemInfoList.EAR$CAS==Chemical[2]), ]
    }
#Check for of Chemical
    if (min(dim(Chem1Info))==0){
        message(paste('Warning! Information for ',NameOut,' is not in the database',sep=''))
        Warnings.EAR[[Warnings.EAR$N+1]]<<-paste('Warning! Information for ',NameOut,' is not
        in the database',sep='')
        Warnings.EAR$N<<-Warnings.EAR$N+1

        return(NA)
    }
#Check for Molecule weight for the chemical
    if (length(Chem1Info$Mol)==0){
        message(paste('Warning! Molecule weight for ',NameOut,' is not in the database',sep=''))
        Warnings.EAR[[Warnings.EAR$N+1]]<<-paste('Warning! Molecule weight for ',NameOut,' is
        not in the database',sep='')
        Warnings.EAR$N<<-Warnings.EAR$N+1
        return(NA)
    }

#Warning Messages
return(Chem1Info$Mol)
}

##########################################################################################
###########
FindDetects<-function(ChemColmn){
#Takes a column from the the ChemData data set within and find the values of non-detect
    #Entries that contain < anywhere within them are below the detection limit
    #1 means it's under the detection limit 0 means it's not

#Find values below detection limit
    SiteValues<-ChemColmn[-{1:3}]
    NonDetecs=which(SiteValues != gsub('<','',SiteValues))
```

```
    SiteValues[1:length(SiteValues)]<-0
    SiteValues[NonDetecs]<-1
return(c(ChemColmn[1],SiteValues))

}

##############################################################################################
############
GetAC50All<-function(Name,NAssay,BOOL){
#gets the AC 50s for all assays for all chemicals in the data set
#Uses Global ChemMasterListFlat.EAR
#NameCasIn is a value with chemical name and CAS name
#NAssay is the total number of Assays

#Apply Crytox
    if (BOOL==TRUE){
        ChemMasterListFlat.EAR<-ChemMasterListFlatCytotox.EAR  #3-21-2016
    }

#Return NA if no name is supplied
    if (Name == ''){
        return(rep(NA,NAssay))
    }
    ColNumber<-which(colnames(ChemMasterListFlat.EAR)==Name)
#Return NA if chemical is not in Assay list
    #6-20-2016 change
    if (length(ColNumber)==0){
        AC50s<-ChemMasterListFlat.EAR[ ,1]
        AC50s[1:length(AC50s)]<-NA
        return(AC50s)
    }
    AC50s<-ChemMasterListFlat.EAR[ ,ColNumber]
#Return NA if chemical is not Active
    Inactive<-which(AC50s==1000)
    if (length(Inactive)>0){
        AC50s[Inactive]<-NA
    }
return(AC50s)
}

##############################################################################################
############
GetAC50ListAll<-function(Assay,AC50Mat,AssayList,SiteN){
#Arranges the AC 50 to be a NSite by NChem matrix as a list
#Assay is the assay being used
#AC50Mat is output from GetAC50All
#AssayList List of every possible assay
#SiteN is the number of Sites
    AssayNum<-which(AssayList==Assay)
    AC50<-AC50Mat[AssayNum, ]
    AC50Rep<-rep(AC50,SiteN)
    AC50Mat<-matrix(AC50Rep,nrow=SiteN,byrow = TRUE)
return(AC50Mat)
```

```
}

##########################################################################################################
########
Convert2uM<-function(Vec,units){
#converts numbers in Vec units to uM
#Only works if called from CalculateEAR
    MW<-Vec[1] #Assumes the first element of the vector is the MW
    Vec<-Vec[-1]
    CorrectUnits<-FALSE
    if (units=='ug/L'){
        Out<-Vec/MW  #changed 9/2/2015
        CorrectUnits<-TRUE
    }
    if (units=='ng/L'){
        Out<-Vec*1000
        CorrectUnits<-TRUE
    }
    if (units=='MW'){  #Double Check this
        Out<-Vec
        CorrectUnits<-TRUE
    }

    if (CorrectUnits==FALSE){
        message('Warning units are not known by the conversion function')
        message('Conversion function can only handle ug/L and ng/L')
    }
return(c(MW,Out))
}


##########################################################################################################
###########
Convert2ug<-function(Vec,units){
#converts numbers in Vec units to ug/L
#only handles ng/L as of right now
    if (units=='ug/L'){
        return(Vec*1)
    }
    if (units=='ng/L'){
        return(Vec*0.001)
    }

    message('Warning units are not known by the conversion function')
    message('Conversion function can only handle ug/L and ng/L')
return(Vec)
}
##########################################################################################################
###########
FilterChemData<-function(ChemColmn){
#Takes a column from the the ChemData data set within the ConvertChemData function and filters
it based on
    #If there is a value reported replace the value reported with 1/4 the lowest value reported
```

```
    #If there is not a value reported replace the value reported with 1/10 the detection limit
    #Entries that contain < anywhere within them are below the detection limit
    #Uses All other symbols as untested chemicals

#Removes all alpha characters
    SiteValues<-ChemColmn[-{1:3}]
    SiteValues<-gsub('[A-Z,a-z]','',SiteValues)
    SiteValues<-gsub(' ','',SiteValues)
#Check for reported Values
    MeasuredValues<-as.numeric(SiteValues)
    ValueReported<-(sum(!is.na(MeasuredValues)>0))   #bool
#Find values below detection limit
    NonDetecs=which(SiteValues != gsub('<','',SiteValues))

    #End Function if there is no Non-Detects
    if (length(NonDetecs)==0){
        MeasuredValues<-Convert2ug(MeasuredValues,ChemColmn[3]) #Convert to ug/L
        return(c(ChemColmn[1],MeasuredValues))
    }

#Detection limit rules
    if (ValueReported){   #Found a measured value for that chemical
        MinValue<-min(MeasuredValues,na.rm=TRUE)/4;
    }
    if (!ValueReported){    #Did not find measured value for that chemical 1/10 detection limit
        MinValue<-as.numeric(gsub('<','',SiteValues))[NonDetecs]/10;
    }
#Apply Detection limit rules
    MeasuredValues[NonDetecs]<-MinValue
#Convert to ug/L
    MeasuredValues<-Convert2ug(MeasuredValues,ChemColmn[3])
return(c(ChemColmn[1],MeasuredValues))
}

##########################################################################################
###########
CalculateEARAll<-function(AC50Mat,ChemDataMW){
#Calculates EAR for all Assays
#AC50Mat is a matrix of AC50's for 1 assay
#ChemDataMW is the chemistry data converted to uMol
    return(ChemDataMW[-1, ]/AC50Mat)
}

##########################################################################################
###########
AboveThreshold<-function(Threshold,ResultsEAR){
#Finds all site-chemical-assay combination above the Threshold
    #Threshold
        #Value to compare results against
    #ResultsEAR
        #Results from CalculatedAllEAR

#Find the values above the Threshold
```

```
    if (is.na(Threshold) == TRUE){
        Threshold<-0
    }
    UpThresh<-which(ResultsEAR$EARCube >=Threshold)
    if (length(UpThresh)<2){
        message('Error not enough values to be exported')
        return(NULL)
    }


    Values<-ResultsEAR$EARCube[UpThresh]

#Find the sites, chemical, and assay associated with that value
    Size<-dim(ResultsEAR$EARCube)

#Assay
    AssayLoc<-ceiling(UpThresh/(Size[1]*Size[2]))
#Prep for site and chemical
    BlockN <-(UpThresh) %% (Size[1]*Size[2])
    if (length(which(BlockN==0))>0){
        BlockN[which(BlockN==0)]<-Size[1]*Size[2]
    }
#Site
    ChemicalLoc<-ceiling(BlockN/(Size[1]))
#Chemical
    SiteLoc<-BlockN %% (Size[1])
    if (length(which(SiteLoc==0))>0){
        SiteLoc[which(SiteLoc==0)]<-Size[1]
    }
    OutSite<-ResultsEAR$SiteData[SiteLoc, ]
    OutChemical<-ResultsEAR$ChemicalNames[ChemicalLoc, ]
    OutAssay<-rownames(ResultsEAR$AC50s)[AssayLoc]
#prep data set
    OutData<-cbind(OutSite,OutChemical,OutAssay)
    OutData<-cbind(OutData,Values)
    colnames(OutData)[c(7,8,9,10)]<-c('Chemical','CASN','Assay','EAR')
    OutData<-OutData[ ,-6]


return(OutData)
}

##############################################################################################
######
##############################################################################################
######
##                                    Information Loading
Functions                                  ##
##############################################################################################
######
##############################################################################################
######
#This file contains all the functions used in loading information files in the EAR Calculator
```

```
#This is the back end to the that handles calculations



################################################################################################
######
#                                          Change
Log                                                    #
################################################################################################
######

#################################################0.75#############################################
######
#Added the global PlottingCat.EAR
    #This vector contains the plotting categories
    #linked all references to the plotting categories to this vector


################################################################################################
######
#
Global                                                  #
################################################################################################
######
#Global values used in the calculation algorithms


#ChemMasterList.EAR
    #Data Set containing assays for all the chemical
#ChemMasterListFlat.EAR
    #Data Set containing assays for all the chemical flattened into a matrix of chemical by
    assay
#ChemInfoList.EAR
    #A data set containing the Names, CAS numbers, molecular weights, and Formulas of the
    chemicals


################################################################################################
######
#                                       Function
List                                                    #
################################################################################################
######

#Melt.EAR
    #GetColumn.Melt.EAR
        #GetRow.GetColumn.Melt.EAR

################################################################################################
######
#                                  Function
Information                                           #
################################################################################################
```

```
######
#Melt.EAR
    #Melt function that converts tall data to wide data

    #Inputs
        #Data
            #The data frame to be melted
        #VarC
            #the name of the variable that will become the new column variable
        #VarR
            #the name of the variable that will become the new row variable
        #VarValue
            #The name of the variable that will become the new value in each cell

    #Outputs
        #FlatData
            #The data set flatten out

    #Called By
        #NOTHING
    #Calls Directly \ Indirectly
        #GetColumn.Melt.EAR
            #GetRow.GetColumn.Melt.EAR

#############################################################################################
######
#GetColumn.Melt.EAR
    #gathers information of the new columns

    #Inputs
        #ColumnValue
            #Value of the variable in the new column the is being attained
        #VarC
            #the name of the variable that will become the new column variable
        #RowValues
            #Value of the variable in the new row the is being attained
        #VarR
            #the name of the variable that will become the new row variable
        #VarValue
            #The name of the variable that will become the new value in each cell
        #Data
            #The data frame to be melted

    #Outputs
        #ColData
            #A column of flat data

    #Called By
        #Melt.EAR
    #Calls Directly \ Indirectly
        #GetRow.GetColumn.Melt.EAR

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

```
#####
#GetRow.GetColumn.Melt.EAR
    #Gathers VarValue from VarR

    #Inputs
        #RowValues
            #Value of the variable in the new row the is being attained
        #VarR
            #the name of the variable that will become the new row variable
        #VarValue
            #The name of the variable that will become the new value in each cell
        #DataSub
            #The subset of the data frame to be melted

    #Outputs
        #Value
            #Value of the VarValue corresponding to the row and columns being searched on

    #Called By
        #NOTHING
    #Calls Directly \ Indirectly
        #GetColumn.Melt.EAR

#############################################################################################
#####
#DefaultAssayGroups
    #Function sets up default groups for assays

    #Inputs
        #NOTHING
        #Global ChemMasterList.EAR
        #Global AssayGroup.EAR

    #Outputs
        #Modifies Global AssayGroup.EAR

    #Called By
        #UpdateEARCalculator
    #Calls Directly \ Indirectly
        #NOTHING
#############################################################################################
#####
#LoadAssayInformation
    #used load and sanitize an Assay file to be merged into

    #Inputs
        #File
            #Name of file to be uploaded
    #Outputs
        #OutData
            #Sanitized Assay information

    #Called By
```

```
        #UpdateAssayGUI.EAR

    #Calls Directly \ Indirectly
        #ApplySelection


################################################################################################
#####
#MergeAssayInformation
    #Merges a new data file with ChemMasterList.EAR

    #Inputs
        #Data
            #Sanitized assay data to be merged
        #Uses global ChemMasterList.EAR

    #Outputs
        #ChemMasterList
            #Non-global version of ChemMasterList.EAR

    #Called By
        #UpdateAssayGUI.EAR

    #Calls Directly \ Indirectly
        #ApplySelection

################################################################################################
#####
#ApplyCytox
    #Apply Cytotoxic information to the AC50s

    #Inputs
        #Data
            #the new cytotoxic data set

    #Outputs
        #NOTHING
        #Modifies global variables of:
            #Cytox.EAR
            #ChemMasterListFlatCytotox.EAR

    #Called By
        #UpdateEARCalculator
    #Calls Directly \ Indirectly
        #NOTHING

################################################################################################
#####
#ApplySelection
    #used to apply the selection criteria across multiple assays

    #Inputs
        #UID
```

```
            #The unique assay-chemical combination being filtered
        #Data
            #Data set containing all assay-chemical combinations

    #Outputs
        #SubData[UseRow, ]
            #the selected row

    #Called By
        #LoadAssayInformation
        #MergeAssayInformation
    #Calls Directly \ Indirectly
        #NOTHING

#################################################################################################
#####
#FilterDataFlags
    #Filters out specific flag_ids

    #Inputs
        #Data
            #MasterChemList formatted data
        #Remove
            #numeric vector of flags to remove

    #Outputs
        #Data
            #Assay data with

    #Called By
        #
    #Calls Directly \ Indirectly
        #NOTHING




#################################################################################################
######
#
Functions                                                    #
#################################################################################################
######

Melt.EAR<-function(Data,VarC,VarR,VarValue){
#Melt function that converts tall data to wide data
#Data is original data
#VarC is the new column value
#VarR is the new row value
#VarValue is the value of the new cells
    ColumnValues<-unique(Data[[VarC]])
    RowValues<-unique(Data[[VarR]])
    FlatData<-sapply(ColumnValues,GetColumn.Melt.EAR,VarC,RowValues,VarR,VarValue,Data)
    FlatData<-t(FlatData)
```

```
}
##############################################################################################################
######




GetColumn.Melt.EAR<-function(ColumnValue,VarC,RowValues,VarR,VarValue,Data){
#gathers information of the new columns
#Only called by Melt.EAR
    DataSub<-Data[which(Data[[VarC]]==ColumnValue), ]
    ColData<-sapply(RowValues,GetRow.GetColumn.Melt.EAR,VarR,VarValue,DataSub)
    print(ColumnValue)
return(ColData)
}
##############################################################################################################
######

GetRow.GetColumn.Melt.EAR<-function(RowValue,VarR,VarValue,DataSub){
#Gathers VarValue from VarR
#Only called by GetColumn.Melt.EAR
    Value<-DataSub[which(DataSub[[VarR]]==RowValue),VarValue]
    if (length(Value)==0){
        return(NA)
    }
    if (length(Value)>1){
        Test1<<-DataSub
        Test2<<-RowValue
        Test3<<-VarValue
    }

return(Value)
}
##############################################################################################################
######

DefaultAssayGroups<-function(){
#Function sets up default groups for assays
#Ran when assays are updated
#Uses global ChemMasterList.EAR
#Modifies global AssayGroup.EAR

#Find where the default groups are
    #AssayGroup.EAR<-NULL #needed on first run


    AllAssays<-unique(ChemMasterList.EAR$Assay.Endpoint)
    Assaykey<-sapply(strsplit(AllAssays,'_'),function(X){return (X[1])} )
    AssayGroup.EAR$defaultgroups<-unique(Assaykey)
    #AssayGroup.EAR$Groups<- AssayGroup.EAR$defaultgroups #needed on first run

#Replace into the default groups
```

```
    for (e in AssayGroup.EAR$defaultgroups){
        AssayGroup.EAR[[e]]<-AllAssays[Assaykey==e]

    }
#Alter Global AssayGroup.EAR
    AssayGroup.EAR<<-AssayGroup.EAR
return(NULL)
}

################################################################################################
######
ApplySelection<-function(UID,Data){
#This function selects one value for the AC50 of an assay-chemical combination
#Data is the data set to be collapsed
#UID is the unique assay chemical combination

    SubLoc<-which(Data[['UID']]==UID)
    SubData<-Data[SubLoc, ]
    print(UID)
#Use the most conservative
    UseRow<-which(SubData[['AC.50']]==min(SubData[['AC.50']]))[1]

return(SubData[UseRow, ])
}

################################################################################################
######
LoadAssayInformation<-function(File){
#used load and sanitize an Assay file to be merged into
#File is the name of the file to be imported
#Added id flags 8-1-2016


#Read Data
    Data<-read.csv(File,stringsAsFactors=FALSE,header=TRUE)
#Apply gsid as the representative sample
    Data<-Data[Data$gsid_rep==1, ]
#filter out information
    Data1<-Data[ ,c('casn','chnm','chid','aenm','logc_min','modl_ga','modl_la','hitc','flag_ids'
    )] #Added
#Gather AC50s
    #AC50 is the model gain AC 50
    #unless a gain DNE then it's the loss
    #If hitc == 0 it's na
    AC50<-10^Data1$modl_ga

    IDga<-which(is.na(Data1[ ,'modl_ga'])==TRUE)
    if (length(IDga)>0){
        AC50[IDga]<-10^Data1[IDga,'modl_ga']
    }
    IDhit<-which(Data1[ ,'hitc']==0)
    if (length(IDhit)>0){
        AC50[IDhit]<-NA
```

```
    }

    R1000<-which(is.na(AC50)==TRUE)
    if(length(R1000)>0){
        AC50[R1000]<-1000
    }

#Store AC 50 and minimum tested values
    Data1$AC.50<-AC50
    Data1$MinTest<-10^Data1$logc_min

#Test to see if AC is less then the smallest tested amount
    Data1$UnderTest<-{AC50<{10^Data1$logc_min}}
    Data2<-Data1[ ,c('casn','chnm','chid','aenm','AC.50','MinTest','UnderTest','flag_ids')]
#Convert columns names back to old format
    colnames(Data2)<-c('CASRN','Chemical.Name','chid','Assay.Endpoint','AC.50',
    'MinTestedConcentration','UnderTest','flag_ids')
    Data2[['Assay.Endpoint']]<-gsub(',','_',Data2[['Assay.Endpoint']])
    Data2[['Chemical.Name']]<-gsub(',','_',Data2[['Chemical.Name']])

#Remove tests that do not have identification of the chemicals used
    Remove<-which(is.na(Data2[['Chemical.Name']])==TRUE)
    if(length(Remove)>0){
        Data2<-Data2[-Remove, ]
    }

return(Data2)
}

##########################################################################################################
######
LoadAssayInformation2<-function(File){
#used load and sanitize an Assay file to be merged into
#File is the name of the file to be imported
#Added id flags 8-1-2016
#Added modl_cc 11-17-2016

#Read Data
    Data<-read.csv(File,stringsAsFactors=FALSE,header=TRUE)
#Apply gsid as the representative sample
    Data<-Data[Data$gsid_rep==1, ]
#filter out information
    Data1<-Data[ ,c('casn','chnm','chid','aenm','logc_min','modl_ga','modl_la','hitc','flag_ids'
    ,'modl_acc')] #Added
#Gather AC50s
    #AC50 is the model gain AC 50
    #Unless a gain DNE then it's the loss
    #If hitc == 0 it's na
    #AC50<-10^Data1$modl_ga
    AC50<-10^Data1$modl_acc
    IDga<-which(is.na(Data1[ ,'modl_acc'])==TRUE)
    if (length(IDga)>0){
        AC50[IDga]<-10^Data1[IDga,'modl_acc']
```

```
    }
    IDhit<-which(Data1[ ,'hitc']==0)
    if (length(IDhit)>0){
        AC50[IDhit]<-NA
    }

    R1000<-which(is.na(AC50)==TRUE)
    if(length(R1000)>0){
        AC50[R1000]<-1000
    }

#Store AC 50 and minimum tested values
    Data1$AC.50<-AC50
    Data1$MinTest<-10^Data1$logc_min

#Test to see if AC is less then the smallest tested amount
    Data1$UnderTest<-{AC50<{10^Data1$logc_min}}
    Data2<-Data1[ ,c('casn','chnm','chid','aenm','AC.50','MinTest','UnderTest','flag_ids')]
#Convert columns names back to old format
    colnames(Data2)<-c('CASRN','Chemical.Name','chid','Assay.Endpoint','AC.50',
    'MinTestedConcentration','UnderTest','flag_ids')
    Data2[['Assay.Endpoint']]<-gsub(',','_',Data2[['Assay.Endpoint']])
    Data2[['Chemical.Name']]<-gsub(',','_',Data2[['Chemical.Name']])

#Remove tests that do not have identification of the chemicals used
    Remove<-which(is.na(Data2[['Chemical.Name']])==TRUE)
    if(length(Remove)>0){
        Data2<-Data2[-Remove, ]
    }

return(Data2)
}

##########################################################################################
######
MergeAssayInformation<-function(Data){
#Merges a new data file with ChemMasterList.EAR
    #It will merge new information in, but it actually replaces the old information
    #with new information in the case of a conflict
#Data, sanitized assay data to be merged
#Uses global ChemMasterList.EAR

#Use a temporay master list
    ChemMasterList<-ChemMasterList.EAR
#Find the conflicts
    ChemMasterListUID<-paste(ChemMasterList[['Assay.Endpoint']],ChemMasterList[['CASRN']])
    DataUID<-paste(Data[['Assay.Endpoint']],Data[['CASRN']])
    Conflics<-which(is.element(ChemMasterListUID,DataUID))

#Remove conflics
    if (length(Conflics)>0){
        ChemMasterList<-ChemMasterList[-Conflics, ]
    }
```

```
#Merge
    ChemMasterList<-rbind(ChemMasterList,Data)

return(ChemMasterList)
}

##########################################################################################
######
ApplyCytox<-function(Data){
    CytoxInfo.EAR<<-Data
    Limit<-CytoxInfo.EAR$lower_bnd_um
    CAS<-sapply(CytoxInfo.EAR$chid,function(X){  ChemInfoList.EAR$CASNum[ChemInfoList.EAR$chid==
    X]})
    Chemicals<-colnames(ChemMasterListFlat.EAR)

#Apply the Cytotoxic information
    ChemMasterListFlatCytotox.EAR<<-sapply(Chemicals,function(X,Limit,CAS){
            LimVal<-Limit[which(CAS==X)]
        #if the value d.n.e set keep all hits
            if (length(LimVal)==0){
                LimVal<-999
            }
        #Test to see if X is element of  ChemMasterListFlatCytotox.EAR
            Out<-ChemMasterListFlat.EAR[ ,X]*{ChemMasterListFlat.EAR[ ,X] < LimVal}
            if (length(which(Out==0))>0){
                Out[which(Out==0)]<-1000
            }
        return(Out)
        },Limit,CAS)
return(TRUE)
}


##########################################################################################
######
ApplySelection.EAR<-function(UID,Data){
#This function selects one value for the AC50 of an assay-chemical combination
#Data is the data set to be collapsed
#UID is the unique assay chemical combination

    SubLoc<-which(Data[['UID']]==UID)
    SubData<-Data[SubLoc, ]
    print(UID)
#Use the most conservative
    UseRow<-which(SubData[['AC.50']]==min(SubData[['AC.50']]))[1]

return(SubData[UseRow, ])
}


##########################################################################################
######
FilterDataFlags<-function(Data,Remove){
```

```
#Filters out specific flag_ids

    Flags<-Data[['flag_ids']]
    Flags<-strsplit(Flags,split='[|]')

    Keep<-unlist(lapply(Flags,function(X){
        return(sum(is.element(as.numeric(X),Remove))==0)
    }))

    Data<-Data[Keep, ]
return(Data)
}

##############################################################################################
######
##############################################################################################
######
##                                                GUI
Functions                                             ##
##############################################################################################
######
##############################################################################################
######
#This file contains all the functions used in the front end user Interface
#This is the back end to the that handles plotting




##############################################################################################
######
#
Global                                                        #
##############################################################################################
######
#Global values

    #Warnings.EAR
        #List data structure [[1]] contains it's length will each element is a warning to the
        user
    #ChemMasterList.EAR
        #Data Set containing assays for all the chemical
    #ChemMasterListFlat.EAR
        #Data Set containing assays for all the chemical flattened into a matrix of chemical by
        assay \
    #ChemMasterListFlatCytotox.EAR
        #Data Set containing assays for all the chemical flattened into a matrix of chemical by
        assay
            #This is adjusted to in a way s.t no EC50s can be Cytotoxic
    #CytoxInfo.EAR
        #Data Structure that store the Cytotoxic information for every chemical
    #ChemInfoList.EAR
        #A data set containing the Names, CAS numbers, molecular weights, and Formulas of the
```

```
        chemicals
    #AssayGroup.EAR
        #Hash table containing Assay groups where group name is a key
    #FunctionList.EAR
        #This contains the names of all functions and pre-set globals in the EAR Calculator
    #SitesList.EAR
        #A  vector of stings containing the sites to plot
    #ChemicalsList.EAR
        #A  vector of stings containing the chemicals to plot
    #AssayList.EAR
        #A  vector of stings containing the Assay to plot
    #SortList.EAR
        #A  vector of stings containing the possible sort criteria
    #MainData.EAR
        #The chemical survey data loaded into
    #CanRun.EAR
        #Bool indicates a whether or not a plot can be made
    #Results.EAR
        #S3 Data structure contain the results information from RunEARAll
    #Replot.EAR
        #Bool indicate that a plot has already been created

#Global GUI containers and objects
    #StartGUI.EAR
        #MainWindow.EAR
            #Main window for data entry form on the EAR Calculator
        #Maingroup.EAR
            #Largest group containing
        #ButtonBox.EAR
            #Box containing the all the buttons for StartGUI.EAR
        #DataBox.EAR
            #The frame containing the table the data is stored in
        #DataGrid.EAR
            #The table object the data is contained within
        #RunButton.EAR
            #Button to run EAR Calculations
        #ShowData.EAR
            #Data displayed in the windaow
        #OpenReulstsButton.EAR
            #Calls the window containing the plotting options

    #PlotViewerGUI.EAR
        #Replot.EAR
            #Bool indicating a plot has been created
        #SelectSiteLabel.EAR
            #Displays the selected site
        #SelectChemicalLabel.EAR
            #Displays the selected chemical
        #SelectAssayLabel.EAR
            #Displays the selected assay
        #GroupSelectCBox.EAR
            #Combo box for selecting how to collapse one of the dimensions of EARCube
        #XMinSelectionCbox.EAR
```

```
            #The minimum index to plot on the x-axis
        #XMaxSelectionCbox.EAR
            #The maximum index to plot on the x-axis
        #YMinSelectionCbox.EAR
            #The minimum index to plot on the y-axis
        #YMaxSelectionCbox.EAR
            #The maximum index to plot on the y-axis
        #SCritSelectionCbox.EAR
            #Combo box for selecting the sort criteria
        #ThresholdSelectionCbox.EAR
            #Combo box for selecting the minimum value a row and column must contain to show up
            the plot
        #SwitchAxisCbox.EAR
            #Combo box containing a Bool indicating if axes should be switch
        #DetectionLimitCbox.EAR
            #Combo box containing the selection for how to handle values under the detection
            limit

    #Temps
        #TempData.EAR
            #Temporary Assay information
        #Flags.EAR
            #ID flags to be removed


################################################################################################
######
#                                        Function
List                                                    #
################################################################################################
######

#Install.EAR

#Run.EAR
    #SetFunctionList
    #Main.EAR
        #StartGUI.EAR               [StartButton]
            #open_cb.EAR            [LoadButton]
            #CheckData.EAR          [RunButton.EAR]
            #ConvertChemData        [RunButton.EAR]
                #ConvertChemData
                    #FilterChemData
                #Convert2ug
            #Find Detects
            #GetShowData.EAR        [RunButton.EAR]
            #CalculatedAllEAR       [RunButton.EAR]
                #CalculateEAR3D
                    #GetAC50All
                    #GetAC50ListAll
                    #CalculateEARAll
                #GetMW
                #Convert2uM
```

```
            #PlotViewerGUI.EAR       [OpenReulstsButton.EAR]
                #GetSitesWindowPop          [SelectSiteButton]
                #GetChemicalsWindowPop      [SelectChemicalButton]
                #GetAssayWindowPop          [SelectAssayButton]
                #CheckPlot                  [UpdateGraphButton] [SaveGraphButton]
                [SaveEARButton]
                #MainPlotEARAll             [UpdateGraphButton] [SaveGraphButton]
                #GetFileName                [SaveGraphButton]
                #ApplyUnderLimit            [SaveEARButton]
                #SumOverAllAssays           [SaveEARButton]
                #FilterEARDataAll           [SaveEARButton]
                #SortDataAll                [SaveEARButton]
                #PrepEARSave                [SaveEARButton]
                #save_cb                    [SaveEARButton]
            #ReadMeFile.EAR              [ReadMeButton]
                #Reference.EAR
                #ShowChangeLog.EAR
            #UpdateAssayGUI.EAR          [UpdateAssayButton]
                #UpdateEARCalculator
                    #Melt.EAR
                        #GetColumn.Melt.EAR
                            #GetRow.GetColumn.Melt.EAR
                    #SaveWorkSpace
                    #Which.Apply
            #UpdateChemicalGUI.EAR       [UpdateChemicalButton]
                #UpdateEARCalculator
                    #Melt.EAR
                        #GetColumn.Melt.EAR
                            #GetRow.GetColumn.Melt.EAR
                    #SaveWorkSpace
                    #Which.Apply
            #UpdateCytox.EAR            [UpdateCytoxButton]
                    #open_cb
                    #SaveEARInformation
                        #UpdateEARCalculator
                            #SaveWorkSpace
                            #ApplyCytox




##########################################################################################################
######
#                                  Function
Information                                                  #
##########################################################################################################
######
#Install.EAR
    #Function to install the packages needed to run the EAR calculator

    #Inputs
        #Folder
            #Folder to install the packages to; defaults to current directory
```

```
    #Output
        #NULL

    #Called By
        #NOTHING
    #Calls Directly \ Indirectly
        #N0THING

################################################################################################
######
#Run.EAR
    #Function used to load all the needed packages and call the main window for the EAR
    Calculator

    #Inputs
        #Folder
            #Folder to install the packages to; defaults to current directory

    #Output

    #Called By
        #NOTHING
    #Calls Directly \ Indirectly
        #SetFunctionList
        #Main.EAR
            #StartGUI.EAR               [StartButton]
                #open_cb.EAR            [LoadButton]
                #CheckData.EAR          [RunButton.EAR]
                #ConvertChemData        [RunButton.EAR]
                    #ConvertChemData
                        #FilterChemData
                    #Convert2ug
                #Find Detects
                #GetShowData.EAR        [RunButton.EAR]
                #CalculatedAllEAR       [RunButton.EAR]
                    #CalculateEAR3D
                        #GetAC50All
                        #GetAC50ListAll
                        #CalculateEARAll
                    #GetMW
                    #Convert2uM
                #PlotViewerGUI.EAR      [OpenReulstsButton.EAR]
                    #GetSitesWindowPop          [SelectSiteButton]
                    #GetChemicalsWindowPop      [SelectChemicalButton]
                    #GetAssayWindowPop          [SelectAssayButton]
                    #CheckPlot                  [UpdateGraphButton] [SaveGraphButton]
                    [SaveEARButton]
                    #MainPlotEARAll             [UpdateGraphButton] [SaveGraphButton]
                    #GetFileName                [SaveGraphButton]
                    #ApplyUnderLimit            [SaveEARButton]
                    #SumOverAllAssays           [SaveEARButton]
                    #FilterEARDataAll           [SaveEARButton]
```

```
                #SortDataAll                [SaveEARButton]
                #PrepEARSave                [SaveEARButton]
                #save_cb                    [SaveEARButton]
            #ReadMeFile.EAR             [ReadMeButton]
                #Reference.EAR
                #ShowChangeLog.EAR
            #UpdateAssayGUI.EAR         [UpdateAssayButton]
                #UpdateEARCalculator
                    #Melt.EAR
                        #GetColumn.Melt.EAR
                            #GetRow.GetColumn.Melt.EAR
                    #SaveWorkSpace
                    #Which.Apply
                #UpdateUpdateAssayGUI.EAR
                #FilterDataFlags
            #UpdateChemicalGUI.EAR      [UpdateChemicalButton]
                #UpdateEARCalculator
                    #Melt.EAR
                        #GetColumn.Melt.EAR
                            #GetRow.GetColumn.Melt.EAR
                    #SaveWorkSpace
                    #Which.Apply
            #UpdateCytox.EAR            [UpdateCytoxButton]
                    #open_cb
                    #SaveEARInformation
                        #UpdateEARCalculator
                            #SaveWorkSpace
                            #ApplyCytox


#############################################################################################
######
#SetFunctionList
    #Sets the list of functions and globals for the EAR Calculator

    #Inputs
        #NOTHING

    #Output
        #Function List

    #Called By
        #Run.EAR
    #Calls Directly \ Indirectly
        #NOTHING

#############################################################################################
######
#Main.EAR
    #Function call for opening the Main Window for the EAR Calculator

    #Inputs
        #NOTHING
```

```
    #Output
        #NULL

    #Called By
        #Run.EAR
    #Calls Directly \ Indirectly
        #StartGUI.EAR               [StartButton]
            #open_cb.EAR            [LoadButton]
            #CheckData.EAR          [RunButton.EAR]
            #ConvertChemData        [RunButton.EAR]
                #ConvertChemData
                    #FilterChemData
                #Convert2ug
            #Find Detects
            #GetShowData.EAR        [RunButton.EAR]
            #CalculatedAllEAR       [RunButton.EAR]
                #CalculateEAR3D
                    #GetAC50All
                    #GetAC50ListAll
                    #CalculateEARAll
                #GetMW
                #Convert2uM
            #PlotViewerGUI.EAR      [OpenReulstsButton.EAR]
                #GetSitesWindowPop          [SelectSiteButton]
                #GetChemicalsWindowPop      [SelectChemicalButton]
                #GetAssayWindowPop          [SelectAssayButton]
                #CheckPlot                  [UpdateGraphButton] [SaveGraphButton]
                [SaveEARButton]
                #MainPlotEARAll             [UpdateGraphButton] [SaveGraphButton]
                #GetFileName                [SaveGraphButton]
                #ApplyUnderLimit            [SaveEARButton]
                #SumOverAllAssays           [SaveEARButton]
                #FilterEARDataAll           [SaveEARButton]
                #SortDataAll                [SaveEARButton]
                #PrepEARSave                [SaveEARButton]
                #save_cb                    [SaveEARButton]


################################################################################################
#####
#ReadMeFile.EAR
    #Shows authers and refrences

    #Inputs
        #NOTHING

    #Output
        #OTHING

    #Called By
        #Run.EAR
    #Calls Directly \ Indirectly
```

```
        #Reference.EAR
        #ShowChangeLog.EAR

##########################################################################################################
#####
#Reference.EAR
    #Shows the refrences

    #Inputs
        #NOTHING

    #Output
        #OTHING

    #Called By
        #ReadMeFile.EAR
    #Calls Directly \ Indirectly
        #NOTHING

##########################################################################################################
#####
#ShowChangeLog.EAR
    #Shows all the changes to the program

    #Inputs
        #NOTHING

    #Output
        #OTHING

    #Called By
        #ReadMeFile.EAR
    #Calls Directly \ Indirectly
        #NOTHING

##########################################################################################################
######
#UpdateAssayGUI.EAR
    #Calls the entry form to load an updated assay file

    #Inputs
        #NOTHING
    #Outputs
        #NOTHING
        #Will erase everything not in FunctionList.EAR
        #Will save the current R Work Space to disk

    #Called By
        #Main.EAR
    #Calls Directly \ Indirectly
        #open_cb
        #LoadAssayInformation
            #ApplySelection
```

```
        #MergeAssayInformation
            #ApplySelection
        #SaveEARInformation
            #UpdateEARCalculator
                #Melt.EAR
                    #GetColumn.Melt.EAR
                        #GetRow.GetColumn.Melt.EAR
                #SaveWorkSpace
                #Which.Apply
        #UpdateUpdateAssayGUI.EAR       [RemovedFlagsButton]
        #FilterDataFlags                [RemovedFlagsButton]


################################################################################################################
######
#open_cb
    #Loads a CSV file

    #Inputs
        #widget
            #Temporary widget
        #window
            #Temporary window

    #Output
        #df
            #The loaded CSV as a data frame

    #Called By
        #UpdateAssayGUI.EAR
        #UpdateChemicalGUI.EAR
    #Calls Directly \ Indirectly
        #NOTHING

################################################################################################################
######
#SaveEARInformation
    #Saves the EAR Calculator with updated information

    #Inputs
        #OldData
            #The old data frame, either Assay information or chemical information
        #NewData
            #The new data frame, either Assay information or chemical information
        #Name
            #Name of the information type being updated either 'Chemical' or 'Assay'


    #Outputs
        #NOTHING
        #Will erases everything not in FunctionList.EAR
        #Will save the current R Work Space to disk
```

```
    #Called By
        #UpdateAssayGUI.EAR
        #UpdateChemicalGUI.EAR
        #UpdateAssayGroup.EAR
        #UpdateCytox.EAR
    #Calls Directly \ Indirectly
        #UpdateEARCalculator
            #Melt.EAR
                #GetColumn.Melt.EAR
                    #GetRow.GetColumn.Melt.EAR
            #SaveWorkSpace
            #Which.Apply

#################################################################################################
######
#UpdateEARCalculator
    #Updates the EAR Calculator File
    #This writes a new R Data base file at a location

    #Inputs
        #NewData
            #The new data to be updated
        #Name
            #Name of the information type being updated either 'Chemical', 'Assay', or
            'AssayGroup'

    #Output
        #NOTHING
        #Will erase everything not in FunctionList.EAR
        #Will save the current R Work Space to disk


    #Called By
        #SaveEARInformation
    #Calls Directly \ Indirectly
        #DefaultAssayGroups
        #Melt.EAR
            #GetColumn.Melt.EAR
                #GetRow.GetColumn.Melt.EAR
        #SaveWorkSpace
        #Which.Apply
        #ApplyCytox

#################################################################################################
######
#SaveWorkSpace
    #Saves the current R workspace

    #Inputs
        #widget
            #Temporary widget
        #window
```

```
            #Temporary window

    #Outputs
        #NOTHING
        #Writes an R data base file to disk

    #Called By
        #UpdateEARCalculator
    #Calls Directly \ Indirectly
        #NOTHING

################################################################################################
######
#UpdateChemicalGUI.EAR
    #Updates the EAR Calculator with new Information about Chemicals

    #Inputs
        #NOTHING
    #Outputs
        #NOTHING
        #Will erase everything not in FunctionList.EAR
        #Will save the current R Work Space to disk

        #Called By
        #Main.EAR
    #Calls Directly \ Indirectly
        #open_cb
        #SaveEARInformation
            #UpdateEARCalculator
                #Melt.EAR
                    #GetColumn.Melt.EAR
                        #GetRow.GetColumn.Melt.EAR
                #SaveWorkSpace
                #Which.Apply

################################################################################################
######
#UpdateAssayGroup.EAR
    #Updates the EAR Calculator with new Assay Groups

    #Inputs
        #NOTHING
    #Outputs
        #NOTHING
        #Will erase everything not in FunctionList.EAR
        #Will save the current R Work Space to disk

        #Called By
        #Main.EAR
    #Calls Directly \ Indirectly
        #open_cb
        #SaveEARInformation
            #UpdateEARCalculator
```

```
                #Melt.EAR
                    #GetColumn.Melt.EAR
                        #GetRow.GetColumn.Melt.EAR
                #SaveWorkSpace
                #Which.Apply

################################################################################################################
######
#UpdateCytox.EAR
    #Front end to update the Cytotoxic information

    #Inputs
        #NOTHING
    #Outputs
        #NOTHING
        #Will erase everything not in FunctionList.EAR
        #Will save the current R Work Space to disk
        #Modifies global variables of:
            #Cytox.EAR
            #ChemMasterListFlatCytotox.EAR

    #Calls Directly \ Indirectly
        #open_cb
        #SaveEARInformation
            #UpdateEARCalculator
                #SaveWorkSpace
                #ApplyCytox



################################################################################################################
######
#StartGUI.EAR
    #Function call for data entry form

    #Inputs
        #NOTHING

    #Objects
        #MainWindow.EAR
            #Main window for data entry form on the EAR Calculator
        #Maingroup.EAR
            #<MainWindow.EAR>
            #Largest group containing
        #ButtonBox.EAR
            #<Maingroup.EAR >
            #Box containing the all the buttons for StartGUI.EAR
        #LoadButton
            #<ButtonBox.EAR>
            #Buttion calls the functions used to load the data
        #DataBox.EAR
            #<Maingroup.EAR >
            #The frame containing the table the data is stored in
```

```
        #DataGrid.EAR
            #<DataGrid.EAR>
            #The table object the data is contained within
        #RunButton.EAR
            #<ButtonBox.EAR>
            #Button to run EAR Calculations
        #ShowData.EAR
            #<ButtonBox.EAR>
            #Data displayed in the window
        #ExportDataButton.EAR
            #<ButtonBox.EAR>
            #Exports the cleaned data set
        #ExportUnderTestButton.EAR
            #<ButtonBox.EAR>
            #Exports the assays that have an AC50 under the lowest tested amount
        #OpenReulstsButton.EAR
            #<ButtonBox.EAR>
            #Calls the window containing the plotting options


    #OutPut
        #NULL

    #Called By
        #Run.EAR
    #Calls Directly \ Indirectly
        #open_cb.EAR             [LoadButton]
        #CheckData.EAR           [RunButton.EAR]
        #ConvertChemData         [RunButton.EAR]
            #ConvertChemData
                #FilterChemData
            #Convert2ug
        #Find Detects
        #GetShowData.EAR         [RunButton.EAR]
        #CalculatedAllEAR        [RunButton.EAR]
            #CalculateEAR3D
                #GetAC50All
                #GetAC50ListAll
                #CalculateEARAll
            #GetMW
            #Convert2uM
        #ConvertChemData         [ExportDataButton.EAR]
        #PlotViewerGUI.EAR       [OpenReulstsButton.EAR]

##############################################################################################
######
#open_cb.EAR
    #Opens an EAR formated CSV file

    #widget
        #Temporary widget
    #window
        #Temporary window
```

```
    #Objects
        #dialog
            #dialogue box used to select a file

    #Output
        #df
            #a data frame

    #Called By
        #StartGUI.EAR [LoadButton]
    #Calls Directly \ Indirectly
        #NOTHING

#################################################################################################
######
#CheckData.EAR
    #Function used to check for problems in the data
    #Function is a place holder to be alterted when needed as problems in input may arive

    #Inputs
        #Data
            #Data frame to be checked

    #Output
        #TRUE or FALSE

    #Called By
        #StartGUI.EAR [RunButton.EAR]
    #Calls Directly \ Indirectly
        #NOTHING



#################################################################################################
######
#GetShowData.EAR
    #Function that combines the data set for display

    #Inputs
        #SiteData
            #Site data from results output CalculatedAllEAR
        #ChemData
            #Site data from results output CalculatedAllEAR
        #CAS
            #CAS data  from results output CalculatedAllEAR

    #Output
        #A data frame meant to be displayed

    #Called By
        #StartGUI.EAR [RunButton.EAR]
    #Calls Directly \ Indirectly
```

```
        #NOTHING

#########################################################################################################
######
#PlotViewerGUI.EAR
    #Function call for bringing up the plot selection window


    #Inputs
        #NOTHING
        #Uses Global Results.EAR

    #Objects
        #PlotWindow
            #NOTHING
            #Main window container
        #Maingroup
            #<PlotWindow>
            #group containing all other containers
        #ControlBox
            #<Maingroup>
            #Frame containing all the widjets relating to graph controls
        #SelectSiteButton
            #<ControlBox>
            #Button to call the site selection window
        #SelectSiteFrame
            #<ControlBox>
            #Frame containing what sites are selected
        #SelectSiteLabel.EAR
            #<SelectSiteFrame>
            #Label containing the selected sites
        #SelectChemicalButton
            #<ControlBox>
            #Button to call the chemical selection window
        #SelectChemicalFrame
            #<ControlBox>
            #Frame containing what Chemical are selected
        #SelectChemicalLabel.EAR
            #<SelectChemicalFrame>
            #Label containing the selected chemical
        #SelectAssayButton
            #<ControlBox>
            #Button to call the assay selection window
        #SelectAssayFrame
            #<ControlBox>
            #Frame containing what assays are selected
        #SelectAssayLabel.EAR
            #<SelectAssayFrame>
            #Label containing the selected assay
        #GroupSelectFrame
            #<ControlBox>
            #Frame containing the combo box that is used to
        #GroupSelectCBox.EAR
```

```
    #<GroupSelectFrame>
    #Combo box containing the values to collapse one of the dimensions of the EARCube
#XRangeFrame
    #<ControlBox>
    #Box containing the min and max values for the x-axis
#XMinSelectionCbox.EAR
    #<XRangeFrame>
    #The minimum index to plot on the x-axis
#XMaxSelectionCbox.EAR
    #<XRangeFrame>
    #The maximum index to plot on the x-axis
#YRangeFrame
    #<ControlBox>
    #Box containing the min and max values for the y-axis
#YMinSelectionCbox.EAR
    #<YRangeFrame>
    #The minimum index to plot on the y-axis
#YMaxSelectionCbox.EAR
    #<YRangeFrame>
    #The maximum index to plot on the y-axis
#SCritFrame
    #<ControlBox>
    #Box containing the combo box for the sort criteria
#SCritSelectionCbox.EAR
    #<SCritFrame>
    #Combo box for selecting the sort criteria
#ThresholdFrame
    #<ControlBox>
    #box containing the combo box for threshold selection
#ThresholdSelectionCbox.EAR
    #<ThresholdFrame>
    #combo box containing the threshold criteria
#SwitchAxisFrame
    #<ControlBox>
    #Box containing the combo box for switching the axes
#SwitchAxisCbox.EAR
    #<ThresholdFrame>
    #Combo box containing bool which indicate the what is on each axis
#DetectionLimitFrame
    #<ControlBox>
    #Box containing the the detection limit combo box
#DetectionLimitCbox.EAR
    #<DetectionLimitCbox.EAR>
    #Combo box containing the values that dictate how measurements under the detection
    limit are handled.
#UpdateGraphButton
    #<ControlBox>
    #Button that updates the graph
#SaveGraphButton
    #<ControlBox>
    #Button that saves the graph to a PDF
#SaveEARButton
    #<ControlBox>
```

```
            #Button that saves the EAR Values to a csv file
        #ExportTheshButton
            #<ControlBox>
            #Button that saves information about EAR Values above the threshold to a csv file
        #GraphBox
            #<Maingroup>
            #Box containing the graph object
        #Graphs
            #<GraphBox>
            #object that is used as the plotting window



        #Called By
            #StartGUI.EAR               [OpenReulstsButton.EAR]
        #Calls Directly \ Indirectly
            #GetSitesWindowPop          [SelectSiteButton]
            #GetChemicalsWindowPop      [SelectChemicalButton]
            #GetAssayTypeWindowPop      [SelectAssayButton]
                #GetAssayWindowPop      [AssayButton]
                #GetAssayGroupWindowPop [AssayGroupButton]

            #CheckPlot                  [UpdateGraphButton] [SaveGraphButton] [SaveEARButton]
            #MainPlotEARAll             [UpdateGraphButton] [SaveGraphButton]
            #GetFileName                [SaveGraphButton]
            #ApplyUnderLimit            [SaveEARButton]
            #SumOverAllAssays           [SaveEARButton]
            #FilterEARDataAll           [SaveEARButton]
            #SortDataAll                [SaveEARButton]
            #PrepEARSave                [SaveEARButton]
            #save_cb                    [SaveEARButton]

################################################################################################
######
#GetSitesWindowPop
    #Function that pops up a window to select the sites to be graphed

    #Inputs
        #ResultsEAR
        #Results from CalculatedAllEAR

    #Output
        #NOTHING
        #Alters Global:
            #SitesList.EAR
            #SelectSiteLabel.EAR

    #Called By
        #PlotViewerGUI.EAR              [SelectSiteButton]
    #Calls Directly \ Indirectly
        #NOTHING
```

```
#################################################################################################################
######
#GetChemicalsWindowPop
    #Function that pops up a window to select the chemicals to be graphed

    #Inputs
        #ResultsEAR
            #Results from CalculatedAllEAR

    #Output
        #NOTHING
        #Alters Global:
            #ChemicalsList.EAR
            #SelectChemicalLabel.EAR

    #Called By
        #PlotViewerGUI.EAR              [SelectChemicalButton]
    #Calls Directly \ Indirectly
        #NOTHING

#################################################################################################################
######
#GetAssayTypeWindowPop
    #Function that pops up a window to select the how the assays are selected


    #Inputs
        #ResultsEAR
            #Results from CalculatedAllEAR

    #Output
        #NOTHING

    #Called By
        #PlotViewerGUI.EAR              [SelectAssayButton]
    #Calls Directly \ Indirectly
        #GetAssayWindowPop              [AssayButton]
        #GetAssayGroupWindowPop         [AssayGroupdButton]


#########################################################################
#GetAssayWindowPop
    #Function that pops up a window to select the assays to be graphed

    #Inputs
        #ResultsEAR
        #Results from CalculatedAllEAR

    #Output
        #NOTHING
        #Alters Global:
            #AssayList.EAR
            #SelectAssayLabel.EAR
```

```
    #Called By
        #GetAssayWindowPop              [AssayButton]
    #Calls Directly \ Indirectly
        #NOTHING

#############################################################################
#GetAssayGroupWindowPop
    #Used to define what assays to by groups

    #Inputs
        #NOTHING
        #Uses global AssayGroup.EAR

    #Output
        #NOTHING
        #Alters Global:
            #AssayList.EAR
            #SelectAssayLabel.EAR

    #Called By
        #GetAssayWindowPop              [AssayGroupButton]
    #Calls Directly \ Indirectly
        #NOTHING


#############################################################################################################
######
#CheckPlot
    #Function that checks the input parameters for a plot and warn the user if conditions are
    not met

    #Inputs
        #XMin
            #Minimum plotting index for the x-axis
        #XMax
            #Maximum plotting index for the x-axis
        #YMin
            #Minimum plotting index for the y-axis
        #YMax
            #Maximum plotting index for the y-axis
        #SCrit
            #Sorting criteria
        #MCrit
            #Multiple selection criteria

    #Output
        #Check
            #Bool indicating that a plot can be made


    #Called By
        #PlotViewerGUI.EAR              [UpdateGraphButton] [SaveGraphButton] [SaveEARButton]
```

```
    #Calls Directly \ Indirectly
        #NOTHING

##########################################################################################################
######
#GetFileName
    #Window to save the output

    #Inputs
        #widget
            #Temporary widget
        #window
            #Temporary window
    #Output
        #FileOut
            #Name of a file

    #Called By
        #PlotViewerGUI.EAR              [SaveGraphButton]
    #Calls Directly \ Indirectly
        #NOTHING

##########################################################################################################
######
#save_cb
    #Window to save the output

    #Inputs
        #widget
            #Temporary widget
        #window
            #Temporary window
        #OutData
            #Data to be saved

    #Output
        #NOTHING

    #Called By
        #PlotViewerGUI.EAR              [SaveEARButton]
    #Calls Directly \ Indirectly
        #save_file

##########################################################################################################
######
#save_file
    #saves a file as a csv

    #Inputs
        #File
            #File name and location

        #OutData
```

```
            #Data to be saves

    #Output
        #NOTHING
        #Writes a csv to disk

    #Called By
        #PlotViewerGUI.EAR              [SaveEARButton]
    #Calls Directly \ Indirectly
        #NOTHING

#########################################################################################################
######
#pop_message
    #Function for displaying messages to the user

    #Inputs
        #message
            #A message to be displayed to the user

    #Outputs
        #NOTHING


    #Called By
        #Most GUI functions
    #Calls Directly \ Indirectly
        #NOTHING

#########################################################################################################
######
#PrepEARSave
    #Prepares  the EAR value so they can be saved to an csv

    #Inputs
        #Data
            #Filterd results from the EAR Calculator

        #SwitchAxis
            #Bool indicating that the axes are switched
    #Outputs
        #DataOut
            #EAR values identical to what is being displayed in the adjacent plot

    #Called By
        #PlotViewerGUI.EAR          [SaveEARButton]
    #Calls Directly \ Indirectly
        #NOTHING


#########################################################################################################
######
#DefineCategory
```

```
    #Calls the window for entering the plotting categories

    #Inputs
        #NOTHING

    #Outputs
        #Changes
            #PlottingCat.EAR
            #BlueLable.EAR
            #GreenLable.EAR
            #RedLable.EAR
    #Called By
        #PlotViewerGUI.EAR          [DefineCatButton]
    #Calls Directly \ Indirectly
        #NOTHING

#################################################################################################
######
#SelectFlags
    #Selects the Flag ID's to remove from the data set

    #Inputs
        #NOTHING

    #Outputs
        #Changes
            #Flags.EAR

    #Called By
        #UpdateUpdateAssayGUI.EAR       [RemovedFlagsButton]
    #Calls Directly \ Indirectly
        #NOTHING









#################################################################################################
######
#
Functions                                                   #
#################################################################################################
######

Install.EAR<-function(Folder=NULL){
#used to install the packages needed for the EAR Calculator

#Set private library is need be
if (is.null(Folder)==FALSE){
    dir.create(Folder, showWarnings = FALSE)
    R_LIBS_USER=Folder
    .libPaths(R_LIBS_USER)
```

```
}
#install packages
    install.packages('RGtk2',dependencies ="Depends" )
    install.packages('R2HTML',dependencies ="Depends")
    install.packages('gWidgetsRGtk2',dependencies ="Depends")
    #install.packages('webchem',dependencies ="Depends")
    install.packages('cairoDevice',dependencies ="Depends")
    install.packages('abind',dependencies ="Depends")

#prompt the installation of GTK+
    library(RGtk2)
}
##########################################################################################################
#############

Run.EAR<-function(Folder=NULL){
#Function used to run the EAR Calculator
#Use private library is need be
#last updated 12-24-2015 JS

if (is.null(Folder)==FALSE){
    dir.create(Folder, showWarnings = FALSE)
    R_LIBS_USER=Folder
    .libPaths(R_LIBS_USER)
}
    require(RGtk2)
    require(gWidgetsRGtk2)
    require(R2HTML)
    #library(webchem)
    require(cairoDevice)
    require(abind)
#Define global PlottingCat.EAR
    PlottingCat.EAR<<-c(0.1,1,10)
#Set function list
    FunctionList.EAR<<-SetFunctionList()

Main.EAR()
}
##########################################################################################################
#############
SetFunctionList<-function(){
#Sets the list of functions and globals for the EAR Calculator
#Update if functions are added
FunctionList<-c("ApplyCharLimit","ApplyUnderLimit","CalcChemWieght","CalculatedAllEAR",
"CalculateEAR3D","CalculateEARAll","CheckData.EAR",
"CheckPlot","ChemInfoList.EAR","ChemMasterList.EAR","ChemMasterListFlat.EAR","CollapseData",
"Convert2ug","Convert2uM","ConvertChemData",
"FilterChemData","FilterEARDataAll","FindDetects","GetAC50All","GetAC50ListAll",
"GetAssayWindowPop","GetChemicalsWindowPop","Getcol",
"GetColumn.Melt.EAR","GetFileName","GetIndexVec","GetLocation","GetMW","GetpchGUI","GetpchPDF",
"GetRow.GetColumn.Melt.EAR","GetShowData.EAR",
"GetSitesWindowPop","GetValueOverCube","InfoFile","Install.EAR","Main.EAR","MainPlotEARAll",
"MasterFile","Melt.EAR","open_cb","open_cb.EAR",
```

```
"PlotEARAll","PlotViewerGUI.EAR","pop_message","Run.EAR","RunEARAll","save_cb","save_file",
"SaveOldInformation","SavePlotEARAll",
"SaveWorkSpace","SetFunctionList","SortDataAll","StartGUI.EAR","SumOverAllAssays","UnderToWhite"
,"UpdateAssayGUI.EAR","UpdateChemicalGUI.EAR",
"UpdateEARCalculator","Which.Apply","WhichOverMinMax",'FunctionList.EAR','ReadMeFile.EAR',
'ShowChangeLog','Reference.EAR','PrepEARSave',
'AssayGroup.EAR','UpdateAssayGroup.EAR','AboveThreshold','GetAssayTypeWindowPop',
'GetAssayGroupWindowPop','DefaultAssayGroups','ApplySelection',
'LoadAssayInformation','MergeAssayInformation','DefineCategory','ChemMasterListFlatCytotox.EAR',
'UpdateCytox.EAR','CytoxInfo.EAR','ApplyCytox',
'Warnings.EAR','SelectFlags','FilterDataFlags','ShowChangeLog.EAR','Flags.EAR'
)


return(FunctionList)
}



#############################################################################################
#############
Main.EAR<-function(){
#main GUI call for EAR Calculator
#Globals (.EAR designates a global)
    SitesList.EAR<<-'All'
    ChemicalsList.EAR<<-'All'
    AssayList.EAR<<-'All'
    SortList.EAR<<-c('Color','Mean','Sum','Median','Active')
    MainData.EAR<<-NULL
    CanRun.EAR<<-FALSE
    Results.EAR<<-NULL
    ShowData.EAR<<-NULL
    Warnings.EAR<<-list()
    Warnings.EAR$N<<-1
#This contains all the function and globals that are in the EAR Calculator program
#This list needs to be updated if more function are added later


#Introductory Window
#Changed on 1-4-2016 to allow for more options
    IntroWindow<-gwindow(horizontal = FALSE,"EAR Calculator ", visible=FALSE,expand=TRUE,fill=
    TRUE)
        size(IntroWindow)<-c(800,700)
        IntroTextframe<-gframe(horizontal = FALSE, container=IntroWindow,where="center",expand=
        TRUE,fill=TRUE)
            IntroTextMessage1<-glabel('Welcome to',container=IntroTextframe,where="center")
            #Title1
                font(IntroTextMessage1)<-list(size=34,color='Black')
            IntroTextMessage2<-glabel('The EAR Calculator',container=IntroTextframe,where=
            "center") #Title1
                font(IntroTextMessage2)<-list(size=34,color='Black')
            IntroTextMessage3<-glabel('March 23, 2016',container=IntroTextframe,where="center")
            #Title2  #Update This
```

```
            font(IntroTextMessage3)<-list(size=16,color='Black')
        IntroTextMessage4<-glabel('Beta V0.76 -Developmental-',container=IntroTextframe,
        where="center") #Title2
            font(IntroTextMessage4)<-list(size=16,color='Black')
        BlankSpace<-gframe(container=IntroTextframe,where="center") #Blank space for formats
            size(BlankSpace)<-c(1,75)
        Group1<-ggroup(container=IntroTextframe,horizontal = FALSE,expand=TRUE,fill=TRUE)
            ReadMeButton<-gbutton('Read Me', container=Group1,handler= function(h,...){
                ReadMeFile.EAR()
                    })
                size(ReadMeButton)<-c(80,60)
                font(ReadMeButton)<-list(size=24,color='Black')
            StartButton<-gbutton('EAR Calculator', container=Group1,handler= function(h,...){
                StartGUI.EAR()
                    })
                size(StartButton)<-c(80,60)
                font(StartButton)<-list(size=24,color='Black')



        UpdateAssayButton<-gbutton('Update Assay', container=Group1,handler= function(h,...){
            UpdateAssayGUI.EAR()
        })
            size(UpdateAssayButton)<-c(80,60)
            font(UpdateAssayButton)<-list(size=24,color='Black')
        UpdateChemicalButton<-gbutton('Update Chemical', container=Group1,handler= function(
        h,...){
            UpdateChemicalGUI.EAR()
        })
            size(UpdateChemicalButton)<-c(80,60)
            font(UpdateChemicalButton)<-list(size=24,color='Black')

        UpdateAssayGroupButton<-gbutton('Update Assay Group', container=Group1,handler=
        function(h,...){
            UpdateAssayGroup.EAR()
        })
            size(UpdateAssayGroupButton)<-c(80,60)
            font(UpdateAssayGroupButton)<-list(size=24,color='Black')
        UpdateCytoxButton<-gbutton('Update Cytotoxic Value', container=Group1,handler=
        function(h,...){
            UpdateCytox.EAR()
        })
            size(UpdateCytoxButton)<-c(80,60)
            font(UpdateCytoxButton)<-list(size=24,color='Black')



    visible(IntroWindow)<-TRUE

}


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

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

ReadMeFile.EAR<-function(){
#This contains the current notes and changes to the program.
    ReadMeWindow<-gwindow(horizontal = FALSE,"EAR Calculator Readme File", visible=FALSE)
        ReadMeGroup<-ggroup(horizontal = FALSE,container=ReadMeWindow)
            #size(IntroWindow)<-c(784,536)
            AuthorMessage<-'
Written and Tested By Joe Swintek
Additional Testing By  Anthony Schroeder, Brett Blackwell, Shibin Li'
            #Update This
            #ChangeMessage<-'', container=ReadMeGroup,where='center')  #Update This
                #font(WelcomeLab)<-list(size=24)
            AuthorLab<-glabel('Authors', container=ReadMeGroup,where='center')
                font(AuthorLab)<-list(size=16)

            IntroTextframe<-gtext(AuthorMessage,horizontal = FALSE, container=ReadMeGroup,expand
            =TRUE,fill=TRUE)
            ChangeLogButton<-gbutton('changes', container=ReadMeGroup,handler= function(h,...){
                        ShowChangeLog.EAR()
                    })
            ReferenceButton<-gbutton('References', container=ReadMeGroup,handler= function(h,...
            ){
                        Reference.EAR()
                    })


    visible(ReadMeWindow)<-TRUE
}
#########################################################################################################
#############


Reference.EAR<-function(){
    #This contain the list of References for the RSCABS Modular

    MessageA1<-'-------------------------------------------------'
    MessageA2<-'-           Package Reference                   -'
    MessageA3<-'-------------------------------------------------'


    MessageA4<-"GTK+
     http://www.gtk.org/ maintained by the GNOME foundation"

    MessageA5<-paste("RGtk2 Package \n",citation('RGtk2')$textVersion,sep='')

    MessageA6<-paste("gWidgetsRGtk2 Package \n", citation('gWidgetsRGtk2')$textVersion,sep='')

    MessageA7<-paste("cairoDevice Package \n", citation('cairoDevice')$textVersion ,sep='')

    MessageA8<-paste("abind Package \n", citation('abind')$textVersion,sep='')
```

```
    ReferenceMessageA<-paste(MessageA1,'\n\n',MessageA2,'\n\n',MessageA3,'\n\n',
        MessageA4,'\n\n',MessageA5,'\n\n',MessageA6,'\n\n',MessageA7,'\n\n',MessageA8,'\n\n',
        MessageA8,'\n\n')

    MessageB1<-'-------------------------------------------------'
    MessageB2<-'-       Assay and Chemical Information          -'
    MessageB3<-'-------------------------------------------------'

    MessageB4<-"Chemical and Assay Information By:
    ToxCast: http://www.epa.gov/ncct/toxcast/data.html
    "

    MessageB5<-"With additional Chemical Information by:
    PubChem: https://pubchem.ncbi.nlm.nih.gov/
    "
    ReferenceMessageB<-paste(MessageB1,'\n\n',MessageB2,'\n\n',MessageB3,'\n\n',
        MessageB4,'\n\n',MessageB5,'\n\n')


    ReferenceMessage<-paste(ReferenceMessageA,ReferenceMessageB)

    ReferenceWindow<-gwindow(horizontal = FALSE,"EAR Calculator References", visible=FALSE)
        ReferenceGroup<-ggroup(horizontal = FALSE,container=ReferenceWindow)
            ReferenceLabel<-gtext( ReferenceMessage,horizontal = FALSE,container=ReferenceGroup,
            expand=TRUE,fill=TRUE)

    size(ReferenceWindow)<-c(600,420)
    visible(ReferenceWindow)<-TRUE
}
##########################################################################################
#############

ShowChangeLog.EAR<-function(){
    Changes<-"---------------------------------V0.78--------------------------------------
Cytotoxic values
Flags.EAR Stores the flag ID that have been removed
Added the ablity to filter on Flag ID
Added sum to multiple selection critera
Added Cytotoxic filtering
Added new assay information loading function
Added assay grouping function

---------------------------------V0.70--------------------------------------
Started Change Log"


    ChangeLogWindow<-gwindow(horizontal = FALSE,"EAR Calculator Change Log", visible=FALSE)
        ChangeLogGroup<-ggroup(horizontal = FALSE,container=ChangeLogWindow)

            ChangeLogLabel<-gtext(Changes,horizontal = FALSE,container=ChangeLogGroup,expand=
            TRUE,fill=TRUE)
```

```
    size(ChangeLogWindow)<-c(450,280)
    visible(ChangeLogWindow)<-TRUE

}
##########################################################################################################
#############



##########################################################################################################
#############
UpdateAssayGUI.EAR<-function(){
#used to update the Assay information
#Altered to change the new assay format 1-8-2016
pop_message('The Assay File is Large, and is truncated for display')
Sys.sleep(0.5)
TempData.EAR<<-NULL

    AssayWindow<-gwindow("Assay Information", visible=FALSE)
        size(AssayWindow)<-c(800,600)
            MainGroup <- ggroup(horizontal = TRUE,container=AssayWindow)
                ButtonBox<-gframe(horizontal = FALSE, container=MainGroup)
                    LoadButton<-gbutton("Load Data",container=ButtonBox,handler= function(h,...
                    ){ #load data button
                        temp<-gtkWindow(show=FALSE)
                        pop_message('The Assay File is Large, and is truncated for display.\n
                        This will take a moment to complete')
                        Sys.sleep(0.5)
                        open_cb(widget, temp) #Used to just get file name  #FileName as global
                        pop_message('This may take a while')
                        #Load sanitize and merge the new assay information #JS 1-8-2016
                            Flags.EAR<<-NULL #Set this to null
                            Data<-LoadAssayInformation(FileName)
                            TempData.EAR<<-MergeAssayInformation(Data)
                        #Freeze GUI while the data set is loading
                            while (is.null(TempData.EAR)==TRUE){
                            }
                        delete(DataBox,DataGrid)
                        Disp<-min(dim(TempData.EAR)[1],100)
                        DataGrid<<-gtable(TempData.EAR[1:Disp, ])
                        Sys.sleep(0.1)
                        add(DataBox,DataGrid,  expand=TRUE,fill=TRUE)
                        add(ButtonBox,SaveButton)
                        SaveAs.EAR<<-'Assay'

                    })
                #Use to remove flagged ID's in the Assays
                RemovedFlagsButton<-gbutton("Remove Flagged IDs",container=ButtonBox,handler=
                function(h,...){
                    Flags.EAR<<-NULL
                    SelectFlags()
                    while(is.null(Flags.EAR)){
                        Sys.sleep(0.1)
```

```
                    }
                    if (Flags.EAR==1982){
                        Flags.EAR<<-NULL
                    }
                    if (is.null(Flags.EAR)==FALSE){
                        TempData.EAR<<-NULL
                        TempData.EAR<<-FilterDataFlags(ChemMasterList.EAR,Flags.EAR)
                            while(is.null(TempData.EAR)){
                                Sys.sleep(0.1)
                            }

                        delete(DataBox,DataGrid)
                        Disp<-min(dim(TempData.EAR)[1],100)
                        DataGrid<<-gtable(TempData.EAR[1:Disp, ])
                        Sys.sleep(0.1)
                        add(DataBox,DataGrid,  expand=TRUE,fill=TRUE)
                        add(ButtonBox,SaveButton)
                        SaveAs.EAR<<-'Flagged'
                    }

                })

            DataBox<<-gframe('Current Assay Information',horizontal = FALSE,expand=TRUE,
            container=MainGroup)  #This box contains the displayed data
                Disp<-min(dim(ChemMasterList.EAR)[1],100)
                DataGrid<<-gtable(ChemMasterList.EAR[1:Disp, ])
                add(DataBox,DataGrid,  expand=TRUE,fill=TRUE)
            SaveButton<<-gbutton("Update EAR Calculator",handler= function(h,...){ #load data
            button
                    SaveOldInformation(ChemMasterList.EAR,TempData.EAR,SaveAs.EAR)


            })
    visible(AssayWindow)<-TRUE
}

##########################################################################################
#############
UpdateAssayGroup.EAR<-function(){
#used to update the chemical information
#uses Global AssayGroup.EAR

#Convert the Hash Table to an Data set
    GroupNames<-sapply(names(AssayGroup.EAR)[-c(1,2)],function(X){rep(X,length(AssayGroup.EAR[[X
    ]]))})
    Assays<-sapply(names(AssayGroup.EAR)[-c(1,2)],function(X){AssayGroup.EAR[[X]]})

    AssayGroupData<-data.frame(as.vector(unlist(Assays)),as.vector(unlist(GroupNames)),
    stringsAsFactors=FALSE)
    colnames(AssayGroupData)<-c("Assay","Group")

TempData.EAR<<-NULL
AssayGroupWindow<-gwindow("Assay Groups", visible=FALSE)
```

```
    size(AssayGroupWindow)<-c(800,600)
        MainGroup <- ggroup(horizontal = TRUE,container=AssayGroupWindow)
            ButtonBox<-gframe(horizontal = FALSE, container=MainGroup)
                LoadButton<-gbutton("Load Data",container=ButtonBox,handler= function(h,...){
                #load data button
                    temp<-gtkWindow(show=FALSE)
                    TempData.EAR<<-open_cb(widget, temp)
                    delete(DataBox,DataGrid)
                    DataGrid<<-gtable(TempData.EAR)
                    Sys.sleep(0.1)
                    add(DataBox,DataGrid,  expand=TRUE,fill=TRUE)
                    add(ButtonBox,SaveButton)
                })

        DataBox<<-gframe('Current Assay Groups',horizontal = FALSE,expand=TRUE, container=
        MainGroup)  #This box contains the displayed data
                    DataGrid<<-gtable(AssayGroupData)
                    add(DataBox,DataGrid,  expand=TRUE,fill=TRUE)
                SaveButton<<-gbutton("Update EAR Calculator",handler= function(h,...){ #load
                data button
                        SaveOldInformation(AssayGroup.EAR,TempData.EAR,'AssayGroup')
                })




    visible(AssayGroupWindow)<-TRUE

}


##############################################################################################
#############
 open_cb <- function(widget, window) {
    #Creates a window to open a file and loads a CVS into R
    #This function is called from all Mods
    dialog <- gtkFileChooserDialog("Choose a CSV file", window, "open",
    "gtk-cancel", GtkResponseType["cancel"], "gtk-open",
    GtkResponseType["accept"])
    if (dialog$run() == GtkResponseType["accept"]) {
        FileName<<-dialog$getFilename()
        if (strsplit(FileName,'.csv') != FileName){
            df <- read.csv(FileName,stringsAsFactors=FALSE,header=TRUE)
        }
        if (strsplit(FileName,'.csv') == FileName){
            df <-  data.frame(variables=character(0), stringsAsFactors=FALSE)
            pop_message('Only comma separated value (.csv) files can be loaded')
        }
    }
    dialog$destroy()
    df
 }
##############################################################################################
```

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

SaveOldInformation<-function(OldData,NewData,Name){
#Prompt user to save the old information
#OldData is the old information data set
#NewData is the Data set that will replace it
#Name is the name of what is being updated either Chemical or Assay
#This function will pop up and display a message to the user
    PopWindow<-gwindow("",width = 240, height= 157,visible=FALSE)
        Textframe<-ggroup(horizontal = FALSE, container=PopWindow,expand=TRUE,fill=TRUE)
            TextMessage<-gtext('Would you like to save the old information as a file?',container
            =Textframe,expand=TRUE,fill=TRUE);
        Yesbutton<-gbutton("Yes",container=Textframe,handler= function(h,...){
                temp<-gtkWindow(show=FALSE)
                save_cb(widget, temp,OldData)
                dispose(PopWindow)
                Sys.sleep(0.25)
                UpdateEARCalculator(NewData,Name)
        })
        Nobutton<-gbutton("No",container=Textframe,handler= function(h,...){
                dispose(PopWindow)
                Sys.sleep(0.25)
                UpdateEARCalculator(NewData,Name)
        })


    visible(PopWindow)<-TRUE
}
##############################################################################################
##############

UpdateEARCalculator<-function(NewData,Name){
#Updates the EAR Calculator File
#NewData is the Data set that will replace it
#Name is the name of what is being updated Chemical or Assay

#Set the New Data as the new information data
    if (Name=='Chemical'){
        names(NewData)<-c('chid',  "DSSTox_Substance_Id", "DSSTox_Structure_Id",
        "DSSTox_QC.Level",'Names', "CASNum","Substance_Type",
            "Substance_Note" , "Structure_SMILES","Structure_InChI","Structure_InChIKey",
            "Formula","Mol")
        ChemInfoList.EAR<<-NewData
    }
    if (Name=='Assay'){
        ChemMasterList.EAR<<-NewData
        pop_message('Flattening the assay file.  This may take a minute')
        ChemMasterList.Flat.EAR<<-NULL
        try(ChemMasterListFlat.EAR<<-Melt.EAR(ChemMasterList.EAR,'Assay.Endpoint','CASRN',
        'AC.50'))

        #Freeze gui while the data set is flattening
            while (is.null(ChemMasterListFlat.EAR)==TRUE){
```

```
            }
        #Update the default assay groups
            DefaultAssayGroups()
    }
    #Added 1-4-2016 to allow for updating
    if (Name=='AssayGroup'){
        #Parse out new groups in the New Data
            NewGroup<-unique(NewData$Group)
            RM<-which( is.element(NewGroup,AssayGroup.EAR$defaultgroups))
            if (length(RM)>0){
                NewGroup<-NewGroup[-RM]
            }
        #Add them to the data set
            for (e in NewGroup){
                AssayGroup.EAR[[e]]<<-NewData[NewData$Group==e, ]
            }
        #Update the group list in AssayGroup.EAR
        RM<-which( is.element(NewGroup,AssayGroup.EAR$Groups))
        if (length(RM)>0){
            NewGroup<-NewGroup[-RM]
        }
        AssayGroup.EAR$Groups<<-c(AssayGroup.EAR$Groups,NewGroup)
    }
    if (Name=='Cytox'){
        CanGo<-NULL
        CanGo<-ApplyCytox(NewData)
        #Pause GUI
            while (is.null(CanGo)==TRUE){

            }
    }

    if (Name=='Flagged'){
        #Does not update ChemMasterlist
        pop_message('Flattening the assay file.  This may take a minute')
        ChemMasterList.Flat.EAR<<-NULL
        try(ChemMasterListFlat.EAR<<-Melt.EAR(NewData,'Assay.Endpoint','CASRN','AC.50'))

        #Freeze gui while the data set is flattening
            while (is.null(ChemMasterListFlat.EAR)==TRUE){

            }
        #Update the default assay groups
            DefaultAssayGroups()
    }


#This program removes everything from the top level workspace the is not in FunctionList.EAR
    RemoveList<-ls(envir = .GlobalEnv)
    RemoveNumber<-unlist(sapply(FunctionList.EAR,Which.Apply ,RemoveList))
    if (length(RemoveNumber)>0){
        RemoveList<-RemoveList[-RemoveNumber]
```

```
    }
    rm(list=RemoveList,pos= .GlobalEnv,envir = .GlobalEnv)

    temp<-gtkWindow(show=FALSE)
    SaveWorkSpace(widget, temp);

return()
}

##############################################################################################################
#############


 SaveWorkSpace<-function(widget, window){
 #Window to save the output
    dialog <- gtkFileChooserDialog("Enter a name for the file", window,
    "save", "gtk-cancel", GtkResponseType["cancel"], "gtk-save",
    GtkResponseType["accept"])
    if (dialog$run() == GtkResponseType["accept"]){
        File<-paste(dialog$getFilename(), '.RData',sep='');
        save.image(File)
        dialog$destroy()
    }
    dialog$destroy()
 }

##############################################################################################################
#############
UpdateChemicalGUI.EAR<-function(){
#used to update the chemical information
#uses Global ChemInfoList.EAR
TempData.EAR<<-NULL
    ChemicialWindow<-gwindow("Chemical Information", visible=FALSE)
        size(ChemicialWindow)<-c(800,600)
            MainGroup <- ggroup(horizontal = TRUE,container=ChemicialWindow)
                ButtonBox<-gframe(horizontal = FALSE, container=MainGroup)
                    LoadButton<-gbutton("Load Data",container=ButtonBox,handler= function(h,...
                    ){ #load data button
                        temp<-gtkWindow(show=FALSE)
                        TempData.EAR<<-open_cb(widget, temp)
                        delete(DataBox,DataGrid)
                        DataGrid<<-gtable(TempData.EAR)
                        Sys.sleep(0.1)
                        add(DataBox,DataGrid,  expand=TRUE,fill=TRUE)

                        add(ButtonBox,SaveButton)
                    })

                DataBox<<-gframe('Current Chemical Information',horizontal = FALSE,expand=TRUE,
                container=MainGroup)  #This box contains the displayed data
                    DataGrid<<-gtable(ChemInfoList.EAR)
                    add(DataBox,DataGrid,  expand=TRUE,fill=TRUE)
                SaveButton<<-gbutton("Update EAR Calculator",handler= function(h,...){ #load
```

```
                data button
                        SaveOldInformation(ChemInfoList.EAR,TempData.EAR,'Chemical')


                    })


    visible(ChemicialWindow)<-TRUE
}

##############################################################################################
#############
UpdateCytox.EAR<-function(){
#used to update the chemical information
#uses Global ChemInfoList.EAR
TempData.EAR<<-NULL
    CytoxWindow<-gwindow("Cytotoxic Information", visible=FALSE)
        size(CytoxWindow)<-c(800,600)
            MainGroup <- ggroup(horizontal = TRUE,container=CytoxWindow)
                ButtonBox<-gframe(horizontal = FALSE, container=MainGroup)
                    LoadButton<-gbutton("Load Data",container=ButtonBox,handler= function(h,...
                    ){ #load data button
                        temp<-gtkWindow(show=FALSE)
                        TempData.EAR<<-open_cb(widget, temp)
                        delete(DataBox,DataGrid)
                        Disp<-min(dim(TempData.EAR)[1],100)
                        DataGrid<<-gtable(TempData.EAR[1:Disp, ])
                        Sys.sleep(0.1)
                        add(DataBox,DataGrid,  expand=TRUE,fill=TRUE)
                        add(ButtonBox,SaveButton)
                    })

                DataBox<<-gframe('Current Cytotoxic Information',horizontal = FALSE,expand=TRUE
                , container=MainGroup)  #This box contains the displayed data
                    Disp<-min(dim(CytoxInfo.EAR)[1],100)
                    DataGrid<<-gtable(CytoxInfo.EAR[1:Disp, ])
                    add(DataBox,DataGrid,  expand=TRUE,fill=TRUE)
                SaveButton<<-gbutton("Update EAR Calculator",handler= function(h,...){ #load
                data button
                        SaveOldInformation(CytoxInfo.EAR,TempData.EAR,'Cytox')
                    })
visible(CytoxWindow)<-TRUE
}


##############################################################################################
#############
StartGUI.EAR<-function(){
#GUI call for the form for the EAR calculator
MainWindow.EAR<<-gwindow("Calculate EAR", visible=FALSE)
        size(MainWindow.EAR)<-c(800,600)
            Maingroup.EAR <<- ggroup(horizontal = TRUE)
            add(MainWindow.EAR,Maingroup.EAR, label="Main")
```

```
ButtonBox.EAR<<-gframe(horizontal = FALSE, container=Maingroup.EAR)
    LoadButton<-gbutton("Load Data",container=ButtonBox.EAR,handler= function(h
    ,...){ #load data button
        temp<-gtkWindow(show=FALSE)
        MainData.EAR<<-open_cb.EAR(widget, temp)
        delete(DataBox.EAR,DataGrid.EAR)
        delete(Maingroup.EAR,DataBox.EAR)
        DataBox.EAR<<-gframe(horizontal = FALSE,expand=TRUE)
            add(Maingroup.EAR,DataBox.EAR,  expand=TRUE)
            DataGrid.EAR<<-gtable(MainData.EAR)
            add(DataBox.EAR,DataGrid.EAR,expand=TRUE)
        add(ButtonBox.EAR,RunButton.EAR)
        add(ButtonBox.EAR,ExportDataButton.EAR)
    })
    RunButton.EAR<<-gbutton("Calculate EAR",handler= function(h,...){ #Button
    to run calculator
        #Check the Data
         CanRun.EAR<<-CheckData.EAR(MainData.EAR)
        if (CanRun.EAR==FALSE){
            pop_message("")
        }
        if (CanRun.EAR==TRUE){
            CleanedData<-ConvertChemData(MainData.EAR)
            pop_message('The Data has been filtered and converted to
            micromolars')
            ShowData.EAR<<-GetShowData.EAR(CleanedData$SiteData,CleanedData$
            ChemData,CleanedData$CasRN)


            delete(DataBox.EAR,DataGrid.EAR)
            Results.EAR<<-CalculatedAllEAR(CleanedData,FALSE)
            ResultsCytox.EAR<<-CalculatedAllEAR(CleanedData,TRUE)

            Sys.sleep(0.1) #need to pause
            DataGrid.EAR<<-gtable(ShowData.EAR)
            add(DataBox.EAR,DataGrid.EAR,expand=TRUE)
            add(ButtonBox.EAR,OpenReulstsButton.EAR)
            add(ButtonBox.EAR,ExportUnderTestButton.EAR)

        }
    })
    #added 12-24-2015; Exports the cleaned data set
    ExportDataButton.EAR<<-gbutton("Export Data",handler= function(h,...){
    #Button to run calculator
        #Calculate cleaned data
            CleanedData<-ConvertChemData(MainData.EAR)
            OutData<-cbind(CleanedData$SiteData,CleanedData$ChemData)
        #Save Data
            stemp<-gtkWindow(show=FALSE)
            save_cb(widget, stemp,OutData)
    })
    #added 1-6-2016; Exports the values lower then the lowest test concentration
    ExportUnderTestButton.EAR<<-gbutton("Under Lowest Tested",handler= function(
```

```
                    h,...){ #Button to run calculator
                        #Identify Data
                            UnderTest<-ChemMasterList.EAR[which(ChemMasterList.EAR$UnderTest==
                            TRUE), ]
                        #Save Data
                            stemp<-gtkWindow(show=FALSE)
                            save_cb(widget, stemp,UnderTest)
                    })


                    OpenReulstsButton.EAR<<-gbutton("View EAR",handler= function(h,...){
                    #Button to run calculator
                        PlotViewerGUI.EAR(Results.EAR)
                    })
                DataBox.EAR<<-gframe(horizontal = FALSE, label='Histopath Main',expand=TRUE)
                #This box contains the displayed data
                    add(Maingroup.EAR,DataBox.EAR,  expand=TRUE)
                    blankDF = data.frame(variables=character(0), stringsAsFactors=FALSE)
                    DataGrid.EAR<<-gtable(blankDF,  expand=TRUE )
                    add(DataBox.EAR,DataGrid.EAR)

    visible(MainWindow.EAR)<-TRUE
}
##########################################################################################################
#############

 open_cb.EAR <- function(widget, window) {
    #Creates a window to open a file and loads a CVS into R
    #This function is called from all Mods
    dialog <- gtkFileChooserDialog("Choose a CSV file", window, "open",
    "gtk-cancel", GtkResponseType["cancel"], "gtk-open",
    GtkResponseType["accept"])
    if (dialog$run() == GtkResponseType["accept"]) {
        FileName<<-dialog$getFilename()
        if (strsplit(FileName,'.csv') != FileName){
            df <- read.csv(FileName,header=FALSE,stringsAsFactors=FALSE)
        }
        if (strsplit(FileName,'.csv') == FileName){
            df <-  data.frame(variables=character(0), stringsAsFactors=FALSE)
            pop_message('Only comma separated value (.csv) files can be loaded')
        }
    }
    dialog$destroy()
    df
 }
##########################################################################################################
#############

 CheckData.EAR<-function(Data){
 #Function check the data for compatibility
 return(TRUE)
 }
##########################################################################################################
```

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

GetShowData.EAR<-function(SiteData,ChemData,CAS){
#Function that combines the Data set for display
#Site data is Site Data from a a .EAR data set
#Chem data is Chemistry Data from a a .EAR data set
#CAS data are CAS numbers to go with the chemicals
    options(warn=-1) #Turn off known warning
        #Convert to micro-molars
            NameCas<-cbind(CAS,CAS)
            MW<-apply(NameCas,1,GetMW,'CAS')
            ChemDataMW<-rbind(MW,ChemData)
            ChemDataMW<-apply(ChemDataMW,2,Convert2uM,"ug/L")

        #Square off Site Data
        SiteData<-rbind('CAS','MW',SiteData)
        NAS1<-which(is.na(SiteData[1, ])==TRUE)
        NAS2<-which(is.na(SiteData[2, ])==TRUE)
        if (length(NAS1)>0){
            SiteData[1,NAS1]='CAS'
        }
        if (length(NAS2)>0){
            SiteData[2,NAS2]='MW'
        }
        ChemData<-rbind(CAS,ChemDataMW)
    options(warn=-1)
return(cbind(SiteData,ChemData))
}
#########################################################################################################
#############

PlotViewerGUI.EAR<-function(ResultsEAR){
#This is a GUI function that is used to View EAR Results


#Parameters
    MRange=max(dim(ResultsEAR$EARCube))
    Replot.EAR<<-FALSE

 #Main Window
    PlotWindow<-gwindow("Calculated EAR", visible=FALSE)
        size(PlotWindow)<-c(1000,640) #Length, Height

        Maingroup <- ggroup(horizontal = TRUE,container=PlotWindow,expand=TRUE)
            #Frame containing the controls for Graph
                ControlBox<-gframe(horizontal = FALSE,container=Maingroup)
                SelectSiteButton<-gbutton('Select Site',fill=TRUE,container=ControlBox,handler=
                function(h,...){
                    GetSitesWindowPop(ResultsEAR)
                })
                    SelectSiteFrame<-gframe('Selected Site',horizontal = FALSE,container=
                    ControlBox)
                        SelectSiteLabel.EAR<<-glabel('All',horizontal = FALSE,container=
```

```
        SelectSiteFrame)
        if (length(SitesList.EAR)>1){
            svalue(SelectSiteLabel.EAR)<<-'Small List'
        }else{
            svalue(SelectSiteLabel.EAR)<<-SitesList.EAR
        }
SelectChemicalButton<-gbutton('Select Chemicals',fill=TRUE,container=ControlBox,
handler= function(h,...){
    GetChemicalsWindowPop(ResultsEAR)
})
    SelectChemicalFrame<-gframe('Selected Chemical',horizontal = FALSE,container
    =ControlBox)
        SelectChemicalLabel.EAR<<-glabel('All',horizontal = FALSE,container=
        SelectChemicalFrame)
            if (length(ChemicalsList.EAR)>1){
                svalue(SelectChemicalLabel.EAR)<<-'Small List'
            }else{
                svalue(SelectChemicalLabel.EAR)<<-ChemicalsList.EAR
            }
SelectAssayButton<-gbutton('Select Assays',fill=TRUE,container=ControlBox,
handler= function(h,...){
    GetAssayTypeWindowPop(ResultsEAR)
})
    SelectAssayFrame<-gframe('Selected Assay',horizontal = FALSE,container=
    ControlBox)
        SelectAssayLabel.EAR<<-glabel('All',horizontal = FALSE,container=
        SelectAssayFrame)
            if (length(AssayList.EAR)>1){
                svalue(SelectAssayLabel.EAR)<<-'Small List'
            }else{
                svalue(SelectAssayLabel.EAR)<<-AssayList.EAR
            }
GroupSelectFrame<-gframe('Criteria For Multiple Selection',horizontal = FALSE,
container=ControlBox)
    GroupSelectCBox.EAR<<-gcombobox(c('Ignore','Site - Largest','Site -
    Smallest','Site - Sum',
                                      'Chemical - Largest','Chemical -
                                      Smallest','Chemical - Sum',
                                      'Assay - Largest','Assay - Smallest'
                                      ,'Assay - Sum'),
        selected = 1,editable = FALSE,container=GroupSelectFrame)

XRangeFrame<-gframe('First and Last Element of X-Axis',container=ControlBox)
    XMinSelectionCbox.EAR<<-gcombobox(1:MRange,selected = 1,container=
    XRangeFrame,editable = TRUE)
        size(XMinSelectionCbox.EAR)<-c(100,20)
    XMaxSelectionCbox.EAR<<-gcombobox(2:MRange,selected = 99,container=
    XRangeFrame,editable = TRUE)
        size(XMaxSelectionCbox.EAR)<-c(100,20)
YRangeFrame<-gframe('First and Last Element of Y-Axis',container=ControlBox)
    YMinSelectionCbox.EAR<<-gcombobox(1:MRange,selected = 1,container=
    YRangeFrame,editable = TRUE)
        size(YMinSelectionCbox.EAR)<-c(100,20)
```

```
    YMaxSelectionCbox.EAR<<-gcombobox(2:MRange,selected = 59,container=
    YRangeFrame,editable = TRUE)
        size(YMaxSelectionCbox.EAR)<-c(100,20)
SCritFrame<-gframe('Sorting Criteria',container=ControlBox)
    SCritSelectionCbox.EAR<<-gcombobox(SortList.EAR,selected = 1,container=
    SCritFrame,editable = FALSE,expand=TRUE)
ThresholdFrame<-gframe('Plotting Threshold',container=ControlBox)
    #Changed 12-24-2015 to account for changes in how categories are defined
    ThresholdSelectionCbox.EAR<<-gcombobox(c(NA,0.01,0.1,0.3,0.5,1,2,5,10,100),
    selected = 1,container=ThresholdFrame,editable = TRUE,expand=TRUE)
SwitchAxisFrame<-gframe('Switch Axes?',container=ControlBox)
    SwitchAxisCbox.EAR<<-gcombobox(c(FALSE,TRUE),selected = 1,container=
    SwitchAxisFrame,editable = FALSE,expand=TRUE)
DetectionLimitFrame<-gframe('Under Detection Limit',container=ControlBox)
    DetectionLimitCbox.EAR<<-gcombobox(c('Nothing','Circle','Zero Out'),selected
     = 1,container=DetectionLimitFrame,editable = FALSE,expand=TRUE)
CytoxFrame<-gframe('Compensate for Cytotoxic Values',container=ControlBox)
    CytoxCbox.EAR<<-gcombobox(c(TRUE,FALSE),selected = 1,container=CytoxFrame,
    editable = FALSE,expand=TRUE)


#Added
DefineCatFrame<-gframe('Define Categories',horizontal = FALSE,container=
ControlBox)
    DefineCatButton<-gbutton('Select Categories',expand=TRUE,container=
    DefineCatFrame,handler= function(h,...){
        DefineCategory()})
    ColorGroup<-ggroup(horizontal = TRUE,expand=TRUE,container=DefineCatFrame)
        BlueLabFrame<-gframe('Blue',container=ColorGroup)
            BlueLable.EAR<<-glabel('0.1',container=BlueLabFrame)
        GreenLabFrame<-gframe('Green',container=ColorGroup)
            GreenLable.EAR<<-glabel('1',container=GreenLabFrame)
        RedLabFrame<-gframe('Red',container=ColorGroup)
            RedLable.EAR<<-glabel('10',container=RedLabFrame)


UpdateGraphButton<-gbutton('Plot Selection',expand=TRUE,container=ControlBox,
handler= function(h,...){
    #Apply Cytox
    if (svalue(CytoxCbox.EAR)==TRUE){
        Results.EAR<-ResultsCytox.EAR
    }

    #plotting Ranges
        XMin<-as.numeric(svalue(XMinSelectionCbox.EAR))
        XMax<-as.numeric(svalue(XMaxSelectionCbox.EAR))
        YMin<-as.numeric(svalue(YMinSelectionCbox.EAR))
        YMax<-as.numeric(svalue(YMaxSelectionCbox.EAR))
    #Sorting criteria
        SCrit<-svalue(SCritSelectionCbox.EAR)
    #Check if plot can be made
        MCrit<-svalue(GroupSelectCBox.EAR)
        MakePlot<-CheckPlot(XMin,XMax,YMin,YMax,SCrit,MCrit)
```

```
            SwitchAxis<-!as.logical(svalue(SwitchAxisCbox.EAR))
            if (MakePlot==TRUE){
                Threshold=as.numeric(svalue(ThresholdSelectionCbox.EAR))
                NonDetect<-svalue(DetectionLimitCbox.EAR)

                Sys.sleep(0.1)
                MainPlotEARAll(ResultsEAR,NULL,SitesList.EAR,ChemicalsList.EAR,
                AssayList.EAR,SwitchAxis,
                    SCrit,Replot.EAR,Threshold,XMin,XMax,YMin,YMax,'GUI',NonDetect,
                    MCrit)
                Replot.EAR<<-TRUE
            }
        })
        SaveGraphButton<-gbutton('Save Plot',expand=TRUE,container=ControlBox,handler=
        function(h,...){
            #Apply Cytox
            if (svalue(CytoxCbox.EAR)==TRUE){
                Results.EAR<-ResultsCytox.EAR
            }
            #plotting Ranges
                XMin<-as.numeric(svalue(XMinSelectionCbox.EAR))
                XMax<-as.numeric(svalue(XMaxSelectionCbox.EAR))
                YMin<-as.numeric(svalue(YMinSelectionCbox.EAR))
                YMax<-as.numeric(svalue(YMaxSelectionCbox.EAR))
            #Sorting criteria
                SCrit<-svalue(SCritSelectionCbox.EAR)
            #Check if plot can be made
                MCrit<-svalue(GroupSelectCBox.EAR)
                MakePlot<-CheckPlot(XMin,XMax,YMin,YMax,SCrit,MCrit)

                if (MakePlot==TRUE){
                    Threshold=as.numeric(svalue(ThresholdSelectionCbox.EAR))
                    SwitchAxis<-!as.logical(svalue(SwitchAxisCbox.EAR))
                    NonDetect<-svalue(DetectionLimitCbox.EAR)

                    Sys.sleep(0.1)
                    stemp<-gtkWindow(show=FALSE)
                    File.Out<-GetFileName(widget, stemp)
                    if (is.null(File.Out)==FALSE){
                        MainPlotEARAll(ResultsEAR,File.Out,SitesList.EAR,
                        ChemicalsList.EAR,AssayList.EAR,SwitchAxis,
                            SCrit,'FALSE',Threshold,XMin,XMax,YMin,YMax,'PDF',NonDetect,
                            MCrit)
                    }
                }

        })
        SaveEARButton<-gbutton('Save EARs',expand=TRUE,container=ControlBox,handler=
        function(h,...){
            #Apply Cytox
            if (svalue(CytoxCbox.EAR)==TRUE){
                Results.EAR<-ResultsCytox.EAR
            }
```

```
        #Check data
            XMin<-as.numeric(svalue(XMinSelectionCbox.EAR))
            XMax<-as.numeric(svalue(XMaxSelectionCbox.EAR))
            YMin<-as.numeric(svalue(YMinSelectionCbox.EAR))
            YMax<-as.numeric(svalue(YMaxSelectionCbox.EAR))
            SCrit<-svalue(SCritSelectionCbox.EAR)
            MCrit<-svalue(GroupSelectCBox.EAR)
            MakePlot<-CheckPlot(XMin,XMax,YMin,YMax,SCrit,MCrit)

            if (MakePlot==TRUE){
            #Filter Data
                Threshold=as.numeric(svalue(ThresholdSelectionCbox.EAR))
                NonDetect<-svalue(DetectionLimitCbox.EAR)
            #Apply the under limit
                BoolUnderLimit<-{NonDetect == 'Zero Out'}
                ResultsEAR<-ApplyUnderLimit(Results.EAR,BoolUnderLimit)


            #Add the sum of EARs of chemicals over all Assays
                if ({ChemicalsList.EAR=='All' || length(which(ChemicalsList.EAR==
                'Sum'))>0}){
                    ResultsEAR<-SumOverAllAssays(ResultsEAR)
                }
            #Filter the data
                FilteredData<-FilterEARDataAll(ResultsEAR,SitesList.EAR,
                ChemicalsList.EAR,AssayList.EAR,Threshold,NonDetect,MCrit)
            #Sort on EAR
                SortedData<-SortDataAll(FilteredData,SCrit=SCrit)
            #Prep the data
                SwitchAxis<-!as.logical(svalue(SwitchAxisCbox.EAR))
                PrepData<-PrepEARSave(SortedData,SwitchAxis)

            #Save the Data
                stemp<-gtkWindow(show=FALSE)
                save_cb(widget, stemp,PrepData)
                Sys.sleep(0.1) #need to pause
                pop_message('Data Saved')
            }
    })
    #Added 1-6-2016
    ExportTheshButton<-gbutton('Export Above Threshold',expand=TRUE,container=
    ControlBox,handler= function(h,...){
        #Apply Cytox
        if (svalue(CytoxCbox.EAR)==TRUE){
            Results.EAR<-ResultsCytox.EAR
        }

        #Gather information from GUI
            Threshold=as.numeric(svalue(ThresholdSelectionCbox.EAR))
            NonDetect<-svalue(DetectionLimitCbox.EAR)
            BoolUnderLimit<-{NonDetect == 'Zero Out'}
        #Apply under limit
```

```
                        ResultsEAR<-ApplyUnderLimit(Results.EAR,BoolUnderLimit)
                        ResultsEAR<-SumOverAllAssays(ResultsEAR)
                    #Find the values above the threshold
                        Outdata<-AboveThreshold(Threshold,ResultsEAR)
                    #Write the results to disk
                        stemp<-gtkWindow(show=FALSE)
                        save_cb(widget, stemp,Outdata)
                        Sys.sleep(0.1) #need to pause
                        pop_message('Data Saved')
                    })




                    #size(UpdateGraphButton)<-c(200,40)
            #Frame Containing the Graph
                GraphBox<-gframe(horizontal = FALSE,container=Maingroup,fill=TRUE,expand=TRUE)
                    Graphs<-ggraphics(container=GraphBox,fill=TRUE,expand=TRUE)

    visible(PlotWindow)<-TRUE
}


##############################################################################################
#############
GetSitesWindowPop<-function(ResultsEAR){
#Creates a pop up window were the user can select sites to be ploted
    Varaibles<-rbind('All',ResultsEAR$SiteData[ ,1:5])

    SitesSelectWindow<-gwindow("Please select Site(s)", visible=FALSE)
        group <- ggroup(horizontal = FALSE, container=SitesSelectWindow,spacing = 20)
        Subsetselect<- gtable(Varaibles,container=group, expand=TRUE, multiple = TRUE)
        SelectButton <- gbutton("Select",container=group,handler= function(h,...)
        {SiteList<-svalue(Subsetselect,index=TRUE);
        #Filter Site List for results
            if (length(which(SiteList==1))>0){ # if all is in the site list just use all
                    SitesList.EAR<<-'All'
            }else{
                SitesList.EAR<<-ResultsEAR$SiteData[['SAMPLE RECORD NUMBER']][SiteList-1]

            }
            #Update GUI
            if(length(SitesList.EAR)>1){
                svalue(SelectSiteLabel.EAR)<<-'Small List'
            }else{
                svalue(SelectSiteLabel.EAR)<<-SitesList.EAR
            }
            dispose(SitesSelectWindow)})
    visible(SitesSelectWindow)<-TRUE
}
##############################################################################################
#############
```

```
GetChemicalsWindowPop<-function(ResultsEAR){
#Creates a pop up window were the user can select Chemicals to be plotted

    colnames(ResultsEAR$ChemicalNames)<-c('Chemical','CAS')
    Varaibles<-ResultsEAR$ChemicalNames
    Varaibles<-rbind('All','Sum',Varaibles)

    SitesSelectWindow<-gwindow("Please select Chemical(s)", visible=FALSE)
        group <- ggroup(horizontal = FALSE, container=SitesSelectWindow,spacing = 20)

        Subsetselect<- gtable(Varaibles,container=group, expand=TRUE, multiple = TRUE)
        SelectButton <- gbutton("Select",container=group,handler= function(h,...){
            ChemicalsList<-svalue(Subsetselect,index=TRUE);
            ChemicalsList.EAR<<-Varaibles[ChemicalsList,2]
            #Filter Site List for results
                if (length(which(ChemicalsList.EAR=='All'))>0){ # if all is in the site list
                just use all
                        ChemicalsList.EAR<<-'All'
                }
                #Update GUI
                if(length(ChemicalsList.EAR)>1){
                    svalue(SelectChemicalLabel.EAR)<<-'Small List'
                }else{
                    svalue(SelectChemicalLabel.EAR)<<-ChemicalsList.EAR
                }
                dispose(SitesSelectWindow)
            })
    visible(SitesSelectWindow)<-TRUE
}
##########################################################################################
#############

GetAssayTypeWindowPop<-function(ResultsEAR){
#Creates a pop up window were the user can select how to select the assays

    AssayTypeSelectWindow<-gwindow("Please select one",,width = 240, height= 157, visible=FALSE)
        group <- ggroup(horizontal = FALSE, container=AssayTypeSelectWindow,spacing = 5,fill=
        TRUE,extend=TRUE)

        AssayButton <- gbutton("Select Individual Assays",container=group,fill=TRUE,extend=TRUE,
        handler= function(h,...){
                GetAssayWindowPop(ResultsEAR)
                dispose(AssayTypeSelectWindow)
            })
            font(AssayButton)<-list(size=24,color='black')
        AssayGroupButton <- gbutton("Select Assays by Group",container=group,fill=TRUE,extend=
        TRUE,handler= function(h,...){
                GetAssayGroupWindowPop()
                dispose(AssayTypeSelectWindow)
            })
            font(AssayGroupButton)<-list(size=24,color='black')
visible(AssayTypeSelectWindow)<-TRUE
}
```

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


GetAssayWindowPop<-function(ResultsEAR){
#Creates a pop up window were the user can select Chemicals to be plotted
    Varaibles<-rownames(ResultsEAR$AC50s)
    Varaibles<-c('All',Varaibles)

    SitesSelectWindow<-gwindow("Please select Assay(s)", visible=FALSE)
        group <- ggroup(horizontal = FALSE, container=SitesSelectWindow,spacing = 20)

        Subsetselect<- gtable(Varaibles,container=group, expand=TRUE, multiple = TRUE)
        SelectButton <- gbutton("Select",container=group,handler= function(h,...){
            AssayList<-svalue(Subsetselect,index=TRUE);
            AssayList.EAR<<-Varaibles[AssayList]
            #Filter Site List for results
                if (length(which(AssayList.EAR=='All'))>0){ # if all is in the site list just
                use all
                        AssayList.EAR<<-'All'
                }
                #Update GUI
                if(length(AssayList.EAR)>1){
                    svalue(SelectAssayLabel.EAR)<<-'Small List'
                }else{
                    svalue(SelectAssayLabel.EAR)<<-AssayList.EAR
                }
                dispose(SitesSelectWindow)
            })
    visible(SitesSelectWindow)<-TRUE
}
##############################################################################################################
#############

GetAssayGroupWindowPop<-function(){
#Creates a pop up window were the user can select assays to be plotted *by group*
#Uses global AssayGroup.EAR

#Get selections
    Varaibles<-AssayGroup.EAR$Groups

    AssaySelectWindow<-gwindow("Please select Assay(s)", visible=FALSE)
        group <- ggroup(horizontal = FALSE, container=AssaySelectWindow,spacing = 20)

        Subsetselect <- gtable(Varaibles,container=group, expand=TRUE, multiple = TRUE)
        SelectButton <- gbutton("Select",container=group,handler= function(h,...){
            GroupList <-svalue(Subsetselect,index=TRUE);

            #Apply user selected list
                OutList<-as.vector(unlist(sapply(GroupList,function(X){return(AssayGroup.EAR[[
                AssayGroup.EAR$Groups[X]]])})))
                AssayList.EAR<<-OutList
            #Update GUI
```

```
                if(length(GroupList)>1){
                    svalue(SelectAssayLabel.EAR)<<-'Small List'
                }else{
                    svalue(SelectAssayLabel.EAR)<<-AssayGroup.EAR$Groups[GroupList]
                }
                dispose(AssaySelectWindow)
            })


visible(AssaySelectWindow)<-TRUE
}

##########################################################################################################
#############

CheckPlot<-function(XMin,XMax,YMin,YMax,SCrit,MCrit){
#Function that checks the input parameters for a plot and warn the user if conditions are not
met
    Message<-NULL #Message for user
    Check<-TRUE #Bool, used as output

    MSite<-FALSE
    MChemical<-FALSE
    MAssay<-FALSE

    if (SitesList.EAR=='All' || length(SitesList.EAR)>1){
        MSite<-TRUE
        if (MCrit=='Site - Largest' || MCrit=='Site - Smallest' || MCrit=='Site - Sum'){
            MSite<-FALSE
        }
    }

    if (ChemicalsList.EAR=='All' || length(ChemicalsList.EAR)>1){
        MChemical<-TRUE
        if (MCrit=='Chemical - Largest' || MCrit=='Chemical - Smallest' || MCrit=='Chemical -
        Sum'){
            MChemical<-FALSE
        }
    }
    if (AssayList.EAR=='All' || length(AssayList.EAR)>1){
        MAssay<-TRUE
        if (MCrit=='Assay - Largest' || MCrit=='Assay - Smallest' || MCrit=='Assay - Sum'){
            MAssay<-FALSE
        }
    }

    if (MAssay==TRUE && MChemical==TRUE && MSite==TRUE){
        Check<-FALSE
    }

    if (Check ==FALSE){
        Message<-paste(Message,'Only one Site, or one Chemical or one Assay must be selected\n')
    }
```

```
if ((XMax-XMin)>99){
#   Check<-FALSE
#   Message<-paste(Message,'The X-Axis can only have up to 100 different values please
lessen your range\n')
    Message<-paste(Message,'Warning! The X-Axis can only handle up to 100 different values
    please lessen your range\n')
}
if ((XMax-XMin)<1){
    Check<-FALSE
    Message<-paste(Message,'The maximum value for X must be more than the minimum\n')
}
if ((YMax-YMin)>60){
#   Check<-FALSE
#   Message<-paste(Message,'The Y-Axis can only have up to 60 different values please
lessen your range\n')
    Message<-paste(Message,'Warning! The Y-Axis can only handle up to 60 different values
    please lessen your range\n')
}
if ((YMax-YMin)<1){
    Check<-FALSE
    Message<-paste(Message,'The maximum value for Y must be more than the minimum\n')
}
if (XMin<0){
    Check<-FALSE
    Message<-paste(Message,'The minimum value for X must be more than 0\n')
}
if (XMax<0){
    Check<-FALSE
    Message<-paste(Message,'The maximum value for X must be more than 0\n')
}
if (YMin<0){
    Check<-FALSE
    Message<-paste(Message,'The minimum value for Y must be more than 0\n')
}
if (YMax<0){
    Check<-FALSE
    Message<-paste(Message,'The maximum value for Y must be more than 0\n')
}
if (floor(XMin) !=XMin){
    Check<-FALSE
    Message<-paste(Message,'The minimum value for X must be an integer \n')
}
if (floor(XMax) !=XMax){
    Check<-FALSE
    Message<-paste(Message,'The maximum value for X must be an integer \n')
}
if (floor(YMin) !=YMin){
    Check<-FALSE
    Message<-paste(Message,'The minimum value for Y must be an integer \n')
}
if (floor(YMax) !=YMax){
    Check<-FALSE
    Message<-paste(Message,'The maximum value for Y must be an integer \n')
```

```
    }
    if (length(which(SortList.EAR==SCrit)) == 0){
        Check<-FALSE
        Message<-paste(Message,'Please select a sorting criteria from the list \n')
    }
    if (is.null(Message)==FALSE){
        pop_message(Message)
    }
return(Check)
}
##############################################################################################
#############


GetFileName<- function(widget, window) {
    #Window to save the output
    FileOut<-NULL
    dialog <- gtkFileChooserDialog("Enter a name for the file", window,
    "save", "gtk-cancel", GtkResponseType["cancel"], "gtk-save",
    GtkResponseType["accept"])
    if (dialog$run() == GtkResponseType["accept"]){
        FileOut=dialog$getFilename()
    }
    dialog$destroy()
return(FileOut)
 }
##############################################################################################
#############

 save_cb <- function(widget, window,OutData) {
    #Window to save the output
    dialog <- gtkFileChooserDialog("Enter a name for the file", window,
    "save", "gtk-cancel", GtkResponseType["cancel"], "gtk-save",
    GtkResponseType["accept"])
    if (dialog$run() == GtkResponseType["accept"])
        save_file(dialog$getFilename(),OutData)
    dialog$destroy()
 }

 ##############################################################################################
 #############

save_file<-function(File,OutData){
#Saves the output
    write.table(OutData,paste(File,'.csv',sep=''),row.names=FALSE,sep=',')
}
##############################################################################################
#############

 pop_message<-function(Message){
    #This function will pop up and display a message to the user
    PopWindow<-gwindow("Alert!",width = 240, height= 157,visible=FALSE)
        Textframe<-ggroup(horizontal = FALSE, container=PopWindow,expand=TRUE,fill=TRUE)
```

```
            TextMessage<-gtext(Message,container=Textframe,expand=TRUE,fill=TRUE);
        Okbutton<-gbutton("OK",container=Textframe,handler= function(h,...){
                dispose(PopWindow)
        })
    visible(PopWindow)<-TRUE
 }
##############################################################################################
#############

PrepEARSave<-function(Data,SwitchAxis){
#Preparea EAR Data to be saved to a CSV
#Data is an EAR calculated File, already filtered and sorted
    YAxis<-Data$ColName
    if (YAxis=='Chemical'){
        YLabels<-Data$ChemicalNames[ ,1]
        DimGraphY<-length(YLabels)
    }
    if (YAxis=='Assay'){
        YLabels<-rownames(Data$AC50s)
        if (Data$TitleName=='Chemical'){
            YLabels<-names(Data$AC50s)
        }
        DimGraphY<-length(YLabels)

    }
    if (YAxis=='Site'){
        SiteData<-Data$SiteData
        FieldNames<-Data[['FIELD NAME']]
        Dates<-Data[['DATE COLLECTED']]
        Times<-Data[['TIME COLLECTED']]


        DimGraphY<-dim(SiteData)[1]
    }
#Get X-axis (Column of EARCube)
    XAxis<-Data$RowName
    if (XAxis=='Chemical'){
        XLabels<-Data$ChemicalNames[ ,1]
        DimGraphX<-length(XLabels)
    }
    if (XAxis=='Assay'){
        XLabels<-rownames(Data$AC50s)
        DimGraphX<-length(XLabels)
    }
    if (XAxis=='Site'){
        SiteData<-Data$SiteData
        FieldNames<-SiteData[['FIELD NAME']]
        Dates<-SiteData[['DATE COLLECTED']]
        Times<-SiteData[['TIME COLLECTED']]
        XLabels<-paste(FieldNames,Dates,Times)
        DimGraphX<-dim(SiteData)[1]
    }
    #Switch Axes if need be
```

```
    if (SwitchAxis==TRUE){
        TAxis<-XAxis
        TLabels<-XLabels
        DimGraphT<-DimGraphX
        XAxis<-YAxis
        XLabels<-YLabels
        DimGraphX<-DimGraphY
        YAxis<-TAxis
        YLabels<-TLabels
        DimGraphY<-DimGraphT
        Data$EARCube<-t(Data$EARCube)
    }
#Prep data to be displaced
    DataOut<-t(Data$EARCube)
    colnames(DataOut)<-XLabels
    DataOut<-cbind(YLabels,DataOut)
    colnames(DataOut)[1]<-YAxis
return(DataOut)
}
################################################################################################
#############

DefineCategory<-function(){
#Calls a window where users can set the plotting categories
#Modifies global PlottingCat.EAR
#Modifies global BlueLable.EAR, GreenLable.EAR, RedLable.EAR

DefineCategoryWindow<-gwindow("Plotting Categories", visible=FALSE)
        Group <- ggroup(horizontal = FALSE, container=DefineCategoryWindow,spacing = 20)
            #Select plotting categories
            DCBlueFrame<-gframe('Blue Category',container=Group,expand=FALSE,fill=FALSE)
                DCBlueCbox<-gcombobox(c(0.01,0.1,0.5,1,2,5,10,20),selected = 2,container=
                DCBlueFrame,
                    editable = TRUE,expand=TRUE,fill=TRUE)
            DCGreenFrame<-gframe('Green Category',container=Group,expand=FALSE,fill=FALSE)
                DCGreenCbox<-gcombobox(c(0.01,0.1,0.5,1,2,5,10,20),selected = 4,container=
                DCGreenFrame,
                    editable = TRUE,expand=TRUE,fill=TRUE)
            DCRedFrame<-gframe('Red Category',container=Group,expand=FALSE,fill=FALSE)
                DCRedCbox<-gcombobox(c(0.01,0.1,0.5,1,2,5,10,20),selected = 7,container=
                DCRedFrame,
                    editable = TRUE,expand=TRUE,fill=TRUE)
            #Confirm Selection
                ConfirmButton <- gbutton("Select",container=Group,expand=FALSE,fill=TRUE,handler
                = function(h,...){
                    PlottingCat.EAR[1]<<-as.numeric(svalue(DCBlueCbox))
                    PlottingCat.EAR[2]<<-as.numeric(svalue(DCGreenCbox))
                    PlottingCat.EAR[3]<<-as.numeric(svalue(DCRedCbox))
                    svalue(BlueLable.EAR)<<-as.numeric(svalue(DCBlueCbox))
                    svalue(GreenLable.EAR)<<-as.numeric(svalue(DCGreenCbox))
                    svalue(RedLable.EAR)<<-as.numeric(svalue(DCRedCbox))
                    #Kill the window
                        dispose(DefineCategoryWindow)
```

```
                })


visible(DefineCategoryWindow)<-TRUE
}

SelectFlags<-function(){
#selects flags ID (By number)
    Window<-gwindow("Flag IDs to remove.", visible=FALSE)
        size(Window)<-c(100,100)
        group <- ggroup(horizontal = FALSE, container=Window,spacing = -10)
            Frame<-gframe(container=group,expand=TRUE,fill=TRUE)
                Message<-glabel('Please type in each flag ID to remove separated by a ","',
                container=Frame,editable = FALSE,expand=TRUE,fill=TRUE)
            Input<-gedit('7,15,17',container=group,editable = TRUE,expand=TRUE,fill=TRUE)


        SelectButton <- gbutton("Select",container=group,handler= function(h,...){
            Flags<-svalue(Input)
            Flags<-as.numeric(strsplit(Flags,',')[[1]])
            MessageOut<-paste0('The flag ID of: \n',Flags[1])
            if (length(Flags)>1){
                for (i in 2:length(Flags)){
                    MessageOut<-paste0(MessageOut,'\n',Flags[i])
                }
            }
                MessageOut<-paste0(MessageOut,'\n','will be removed!\n This may take a minute!')
                pop_message(MessageOut)
                Flags.EAR<<-Flags
                dispose(Window)
            })
    visible(Window)<-TRUE

    addHandlerUnrealize(Window, handler = function(h,...) {
            val <- gconfirm("Forgo removing flagged data", parent=h$obj)
            if(as.logical(val)){
                Flags.EAR<<-1982     #is the null value for flags
                return(FALSE)        # destroy
            }else{
                return(TRUE)        # don't destroy
            }
          })
}


##############################################################################################
######
##############################################################################################
######
##                                          Plotting
Functions                                        ##
##############################################################################################
######
```

```
##############################################################################################################
#This file contains all the functions used in plotting EAR values
#This is the back end to the that handles plotting


##############################################################################################################
#
Global                                                    #
##############################################################################################################
#Global values used in the calculation algorithms


#Warnings.EAR
    #List data structure [[1]] contains it's length will each element is a warning to the user
#ChemMasterList.EAR
    #Data Set containing assays for all the chemical
#ChemMasterListFlat.EAR
    #Data Set containing assays for all the chemical flattened into a matrix of chemical by
    assay
#ChemInfoList.EAR
    #A data set containing the Names, CAS numbers, molecular weights, and Formulas of the
    chemicals
#PlottingCat.EAR
    #Global Vector that stores the values for the plotting categories
        #4 Categories 1: dot, 2: blue, 3:green, 4:red (
        #3 numbers indicating the upper bound of each category

##############################################################################################################
#                                        Function
List                                                  #
##############################################################################################################

#MainPlotEARAll
    #ApplyUnderLimit
        #SumOverAllAssays
        #FilterEARDataAll
            #Which.Apply
            #CollapseData
                #WhichOverMinMax
                #GetValueOverCube
                    #GetIndexVec
                #GetIndexVec
        #SortDataAll
            #CalcChemWieght
        #PlotEARAll
            #ApplyCharLimit
            #UnderToWhite
            #GetpchPDF
```

```
            #GetpchGUI
            #Getcol
            #GetLocation
        #SavePlotEARAll
            #PlotEARAll
                #ApplyCharLimit
                #UnderToWhite
                #GetpchPDF
                #GetpchGUI
                #Getcol
                #GetLocation



################################################################################################
######
#                                    Function
Information                                                  #
################################################################################################
######
#MainPlotEARAll
    #Main Function call for plotting EAR Values

    #Inputs
        #ResultsEAR
            #Results from RunEARAll, see RunEARAll for details
        #File
            #Name of the PDF that the plot creates
            #if a file is not supplied a PDF will not be made
        #Sites
            #A character vector containing a list of sites to be plotted
            #Can be 'All' for all sites
        #Chemicals
            #A character vector containing a list of chemicals to be plotted
            #Can be 'All' for all Chemicals
        #Assay
            #A character vector containing a list of assays to be plotted
            #Can be 'All' for all Chemicals
        #SwitchAxis
            #Bool used to switch X and Y axes
        #SCrit
            #The criteria for sorting values on the graph
            #can be 'Mean', 'Sum', 'Median', 'WMedain','Active','Red', or 'Color'
        #Replot
            #Bool indicating if this is a plot made on an already created window or not
            #If a plotting window already exists and Replot is set to FALSE then an error will
            occur
        #Threshold
            #Minimum EAR value that a row or column must contain to appear on the graph
        #XMin,XMax,Ymin,Ymax
            #The minimum and maximum row and column numbers that are plotted
        #PchTy
            #Selects the function that handles the plotting symbols used
```

```
            #can be either 'GUI' or 'PDF'
        #NonDetect
            #Argument to handle chemicals that are under the detection limit
            #can be 'Nothing','Circle',or 'Zero Out'
        #MCrit
            #Selection criteria used when the size of Sites, Chemical and Assay are all larger
            then 1
            #Can be 'Ignore',
            #'Site - Smallest', 'Site - Largest','Site - Sum'
            #'Chemical - Smallest', 'Chemical - Largest', 'Chemical - Sum
            #'Assay - Smallest', 'Assay - Largest', 'Assay - Sum


    #OutPut
        #A graph to a plotting window
        #A PDF to a file location

    #Called By
        #NOTHING
        #PlotViewerGUI.EAR (GUI) [UpdateGraphButton]
        #PlotViewerGUI.EAR (GUI) [SaveGraphButton]
    #Calls Directly \ Indirectly
        #ApplyUnderLimit
        #SumOverAllAssays
        #FilterEARDataAll
            #Which.Apply
            #CollapseData
                #WhichOverMinMax
                #GetValueOverCube
                    #GetIndexVec
                #GetIndexVec
        #SortDataAll
            #CalcChemWieght
        #PlotEARAll
            #GetpchPDF
            #GetpchGUI
            #Getcol
        #SavePlotEARAll
            #PlotEARAll
                #GetpchPDF
                #GetpchGUI
                #Getcol
################################################################################################
############################
#ApplyUnderLimit
    #Fully build the under limit cube and applies it if selected

    #Inputs
        #ResultsEAR
            #Results from RunEARAll, see RunEARAll for details
        #BoolUnderLimit
            #Bool to select the use of the UnderLimit
```

```
    #OutPut
        #ResultsEAR
            #Results from RunEARAll, see RunEARAll for details
            #Addition UnderLimit is now a cube
            #UnderLimit may have been applyed to EARCube


    #Called By
        #MainPlotEARAll
        #PlotViewerGUI.EAR (GUI) [SaveEARButton]

    #Calls Directly \ Indirectly
        #NOTHING

##############################################################################################
#########################
#SumOverAllAssays
    #Adds a new chemical entry to that is the calculated sum EARs over for an assay over all
    chemicals
    #Is only called if Chemicals [in MainPlotEARAll] is set to 'All'

    #Inputs
        #ResultsEAR
            #Results from RunEARAll, see RunEARAll for details

    #OutPut
        #ResultsEAR
        #Results from RunEARAll, see RunEARAll for details
        #Addition UnderLimit is changed by the addition of 'Sum'
        #EARCube is changed by the addition of 'Sum'
        #ChemicalNames is changed by the addition of 'Sum'
        #MW is changed by the addition of 'Sum'
        #AC50s is changed by the addition of 'Sum'



    #Called By
        #MainPlotEARAll
        #PlotViewerGUI.EAR (GUI) [SaveEARButton]

    #Calls Directly \ Indirectly
        #NOTHING

##############################################################################################
#########################
#FilterEARDataAll
    #Filters the EAR results based on subset selection


    #Inputs
        #ResultsEAR
            #Results from RunEARAll, see RunEARAll for details
        #Sites
```

```
        #A character vector containing a list of sites to be plotted
        #Can be 'All' for all sites
    #Chemicals
        #A character vector containing a list of chemicals to be plotted
        #Can be 'All' for all Chemicals
    #Assay
        #A character vector containing a list of assays to be plotted
        #Can be 'All' for all Chemicals
    #NonDetect
        #Argument to handle chemicals that are under the detection limit
        #can be 'Nothing','Circle',or 'Zero Out'
    #MCrit
        #Selection criteria used when the size of Sites, Chemical and Assay are all larger
        then 1
        #Can be 'Ignore',
        #'Site - Smallest', 'Site - Largest','Site - Sum'
        #'Chemical - Smallest', 'Chemical - Largest', 'Chemical - Sum
        #'Assay - Smallest', 'Assay - Largest', 'Assay - Sum
#OutPut
    #FilteredData
        #ResultsEAR Filtered by selections
        #Contains
            #EARCube
                #Now 2 dimensional Array of EAR Values
            #ChemicalNames
                #A number of chemicals by 2 character vector that contain chemical names [
                ,1] and CAS numbers [ ,2]
            #SiteData
                #A data frame containing:
                    #USGS STATION NAME
                        #The Full Name of the Station
                    #FIELD NAME
                        #An abbreviated name of the station
                    #DATE COLLECTED
                        #The date of collection
                    #TIME COLLECTE
                        #The time the data is collected
                    #SAMPLE RECORD NUMBER
                        #A number unique to the collected sample
                    #Units
            #MW
                #A character vector that contains the molecular weights of each chemical as
                its entries.
                #The names of the vector are the chemicals themselves
            #AC50s
                #A number of assays by number of chemicals matrix
                #The row names are the names of the assays
            #UnderLimit
                #Same size arraay as EARCube with values of 1 indicating that an
                observation is under the detection limit
            #RemoveListRow:
                #List of enrties removed from the rows of EARCube due to Theshhold values
            #RemoveListCol:
```

```
                    #List of enrties removed from the columns of EARCube due to Theshhold values
                #RowName
                    #Row axis label
                #ColName
                    #Col axis label
                #TitleName
                    #Dimention for title of graph

    #Called By
        #MainPlotEARAll
        #PlotViewerGUI.EAR (GUI) [SaveEARButton]

    #Calls Directly \ Indirectly
        #Which.Apply
        #CollapseData
################################################################################################
############################
#CollapseData
    #Collapses part of the EAR Cube into the smallest of largest values of one dimension
    #This function is designed to be called from FilterEARDataAll and can not run independent
    of it

    #Inputs
        #FilteredData
            #Data that is currently being filtered by FilterEARDataAll
        #MCrit
            #Selection criteria used when the size of Sites, Chemical and Assay are all larger
            then 1
            #Can be 'Ignore',
            #'Site - Smallest', 'Site - Largest','Site - Sum'
            #'Chemical - Smallest', 'Chemical - Largest', 'Chemical - Sum
            #'Assay - Smallest', 'Assay - Largest', 'Assay - Sum
        #NonDetect
            #Argument to handle chemicals that are under the detection limit
            #can be 'Nothing','Circle',or 'Zero Out'

    #OutPut
        #FilteredData
         #FilterdData that has had UnderLimit and EARCube collapsed across one dimension

    #Called By
        #FilterEARDataAll
    #Calls Directly \ Indirectly
        #WhichOverMinMax
        #GetValueOverCube
            #GetIndexVec
        #GetIndexVec

################################################################################################
############################
#GetValueOverCube
    #Gets the value of UnderCube at location [i,e,Val]
```

```
    #Inputs
        #Val
            #Value of the unknown dimension
        #i
            #The index of the first dimension
        #e
            #the index of the second dimension
        #UnderCube
            #The UnderCube from FilterdData
        #Dim
            #The dimensions UnderCube is searched on

    #Output
        #UnderCube[Index[1],Index[2],Index[3]]
            #The value at the correct location

    #Called By
        #CollapseData
    #Calls Directly
        #GetIndexVec

##############################################################################################
############################
#WhichOverMinMax
    #Which function for Min, max, mean, any find function that can take na.rm as an argument

    #Inputs
        #Data
            #A rank 3 tensor to run a which (fun) on
        #fun
            #The function which is being used on
            #Created for either min or max

    #Output
        #Out
            #A vector of indices

    #Called By
        #CollapseData
    #Calls Directly
        #NOTHING

##############################################################################################
############################
#GetIndexVec
    #Gets an index vector based on Dim
    #This will rearrange the index vector to follow the dimensions of the Cube

    #Inputs
        #VecIn
            #The index vector that is to be rearranged
        #Dim
            #The dimension filtering is on
```

```
    #Output
        #VecOut
            #A vector of indexes in the correct order

    #Called By
        #CollapseData
        #GetValueOverCube
    #Calls Directly \ Indirectly
        #NOTHING

#############################################################################################
#############################
#SortDataAll
#Sorts Data based on a sort criteria

    #Inputs
        #FilterdData
            #Data filtered by FilterEARDataAll
        #SCrit
            #Sort criteria
            #can be 'Mean', 'Sum', 'Median', 'WMedain','Active','Red', or 'Color'

    #OutPut
        #SortData
            #FilterdData sorted by the sort criteria

    #Called By
        #MainPlotEARAll
    #Calls Directly \ Indirectly
        #CalcChemWieght

#############################################################################################
#############################
#PlotEARAll
    #Plots EAR results it uses something similar to a Cleveland dot plot

    #Inputs
        #Uses PlottingCat.EAR
        #Data
            #Data sorted and filtered by SortDataAll and FilterEARDataAll
        #AddLegend
            #Bool to include the legend
            #Only used when saving the plot
        #SwitchAxis
            #Bool used to switch the X and Y axes
        #Replot
            #Bool indicating if the plotting window has already been established
        #XMin, XMax, YMin, YMax
            #Used to indicate the location of the first and last elements in the X and Y axis
        #PchTy
            #indicate the function that determines the plotting symbols
```

```
            #Can be 'GUI' of 'PDF'
        #NonDetect
            #Determines the effect of locations under the detection limit
            #Can be 'Nothing','Circle', or 'Zero Out'



    #OutPut
        #Either an update to the plotting window or PDF

    #Called By
        #MainPlotEARAll
        #SavePlotEARAll

    #Calls Directly/Indirectly
        #ApplyCharLimit
        #UnderToWhite
        #GetpchPDF
        #GetpchGUI
        #Getcol
        #GetLocation

################################################################################################################
#############################
#GetpchPDF
    #Gets the plotting symbol used in the pdf

    #Inputs
        #Uses global PlottingCat.EAR
        #Num
            #A number to determine what plotting symbol is used

    #Output
        #A number or string that determines a plotting symbol

    #Called By
        #PlotEARAll

    #Calls Directly/Indirectly
        #NOTHING


################################################################################################################
#############################
#GetpchGUI
    #Gets the plotting symbol used in the GUI

    #Inputs
        #Uses global PlottingCat.EAR
        #Num
            #A number to determine what plotting symbol is used

    #Output
```

```
        #A number or string that determines a plotting symbol

    #Called By
        #PlotEARAll

    #Calls Directly/Indirectly
        #NOTHING

##############################################################################################################
############################
#Getcol
    #Function used to determin the plotting color

    #Inputs
        #Uses global PlottingCat.EAR
        #Num
            #A number to determine what color is used

    #Output
        #A  string that determines a plotting color

    #Called By
        #PlotEARAll

    #Calls Directly/Indirectly
        #NOTHING

##############################################################################################################
############################
#SavePlotEARAll
    #Function called to save a plot to a PDF

    #Inputs
        #Data
            #Data sorted and filtered by SortDataAll and FilterEARDataAll
        #SwitchAxis
            #Bool used to switch the X and Y axes
        #XMin, XMax, YMin, YMax
            #Used to indicate the location of the first and last elements in the X and Y axis
        #NonDetect
            #Determines the effect of locations under the detection limit
            #Can be 'Nothing','Circle', or 'Zero Out'

    #Called By
        #MainPlotEARAll
    #Calls Directly/Indirectly
        #PlotEARAll
            #ApplyCharLimit
            #UnderToWhite
            #GetpchPDF
            #GetpchGUI
            #Getcol
            #GetLocation
```

```
################################################################################################################
############################
#CalcChemWieght
    #Calculates the sorting wieght for an EAR value that is to be plotted
    #Ment to be called using the apply function

    #Inputs
        #Uses PlottingCat.EAR
        #Vec
            #A vector of numbers
        #SCrit
            #Sort criteria
            #can be 'Mean', 'Sum', 'Median', 'WMedain','Active','Red', or 'Color'

    #Output
        #Wieght
        #A vector of plotting weights that can be sorted on

    #Called By
        #SortDataAll
    #Calls Directly/Indirectly
        #NOTHING

################################################################################################################
############################
#ApplyCharLimit
    #Apply a Character limit, removes right white space, pads left white space, can change
    chemical name to formula

    #Inputs
        #String
            #Sting to be filtered
        #Limit
            #Max and Min size of the output string

        #IsChemical
            #Bool indicating whether the sting is a chemical or not

        #Uses global ChemInfoList.EAR

    #OutPut
        #NewString
            #Sting containing of 'Limit' number of characters

    #Called By
        #PlotEAR
    #Calls Directly \ Indirectly
        #NOTHING

################################################################################################################
############################
#UnderToWhite
```

```
    #Changes underscores in a string to while space

    #Inputs
        #Str
            #A string to be converted

    #OutPut
        #Str
            #A string with all underscores in the string converted to white space

    #Called By
        #PlotEAR
    #Calls Directly \ Indirectly
        #NOTHING

#############################################################################################
############################
#GetLocation
    #Attains the locations within EAR Cube that are between ValueMin and ValueMax

    #Inputs
        #EARCube
            #EAR calues that are prepaired to be plotted

        #ValueMin
            #Min Value of an entry of EARCube to be plotted

        #ValueMax
            #Max Value of an entry of EARCube to be plotted

        #UseNA
            #Bool, when TRUE it finds the locations of NA within EARCube

        #OutPut
            #Points
                #A 2-column matrix containing the X and Y cordanites of the points

    #Called By
        #PlotEAR
    #Calls Directly \ Indirectly
        #NOTHING

#############################################################################################
############################
#Which.Apply
    #Which to be called though an apply function

    #Inputs
        #Value
            #The value being seached on
        #Vec
            #The vector being searched
    #Output
```

```
        #N
            #The index number of value in vector

    #Called By
        #FilterEARDataAll
    #Calls Directly \ Indirectly
        #NOTHING

#################################################################################################
######
#
Functions                                                    #
#################################################################################################
######
MainPlotEARAll<-function(ResultsEAR,File=NULL,Sites='All',Chemicals='All',Assay='All',SwitchAxis
=FALSE,SCrit='Color',
        Replot=FALSE,Threshold=NA,XMin=1,XMax=100,YMin=1,YMax=60,PchTy='GUI',NonDetect='Nothing'
        ,MCrit='Ignore'){
#Main function call to plot EAR results
#ResultsEAR Results from RunEAR
#File, if supplied it will same the graph to a named file
#Sites are the subset of sites used in plotting
#Chemicals are the chemicals used for graphing
#SwitchAxis Flip X and Y axis

    Check<-TRUE
#Check inputs
    if (Sites=='All' && Chemicals=='All' && Assay=='All'){
        Check<-FALSE
        if (MCrit !='Ignore'){
            Check<-TRUE
        }
    }

    if (length(Sites)>1 && length(Chemicals)>1 && length(Assay)>1){
        Check<-FALSE
        if (MCrit !='Ignore'){
            Check<-TRUE
        }
    }
    if (Check ==FALSE){
        message('Only one Site, or one Chemical or one Assay must be selected')
        return()
    }

#Condition to apply UnderLimit  TRUE-> it is applied FALSE-> it is skipped
    BoolUnderLimit<-{NonDetect == 'Zero Out'}
    ResultsEAR<-ApplyUnderLimit(ResultsEAR,BoolUnderLimit)

#Add the sum of EARs of chemicals over all Assays
    if ({Chemicals=='All' || length(which(Chemicals=='Sum'))>0}){
        ResultsEAR<-SumOverAllAssays(ResultsEAR)
    }
```

```
#Filter the Data
    FilteredData<-FilterEARDataAll(ResultsEAR,Sites,Chemicals,Assay,Threshold,NonDetect,MCrit)


#Sort on EAR
    SortedData<-SortDataAll(FilteredData,SCrit=SCrit)


#Plot The Data
    if  (is.null(File)==TRUE){
        PlotEARAll(SortedData,AddLegend=FALSE,SwitchAxis,Replot,XMin,XMax,YMin,YMax,PchTy,
        NonDetect)
    }
#Save the plot
    if  (is.null(File)==FALSE){
        File=paste(File,'.pdf')
        SavePlotEARAll(SortedData,File,SwitchAxis,XMin,XMax,YMin,YMax,PchTy,NonDetect)
    }
return()
}
##############################################################################################
############################

ApplyUnderLimit<-function(ResultsEAR,BoolUnderLimit=FALSE){
#Applies the under limit when the zero out selection is used
#BoolUnderLimit is to select to apply the under limit

#Get under limit to be a multiplier
    UnderLimit<-apply(!ResultsEAR$UnderLimit,2,as.numeric)

#Cube the under limit data
    Dim<-dim(ResultsEAR$EARCube)
    UnderCube<-array(rep(UnderLimit,Dim[3]),dim=Dim)
#Apply the under cube
    if ( BoolUnderLimit==TRUE ){
        ResultsEAR$EARCube<-ResultsEAR$EARCube*UnderCube
    }
    ResultsEAR$UnderLimit<-(UnderCube-1)*-1

return(ResultsEAR)
}
##############################################################################################
############################

SumOverAllAssays<-function(ResultsEARAll){
#Function Adds a new chemical entry to that is the calculated sum EARs over for an assay over
all chemicals

#Update identifier structures with 'Sum'
    #The ChemicalNames data structure
        ResultsEARAll$ChemicalNames<-rbind(ResultsEARAll$ChemicalNames,'Sum')
    #The MW data structure
```

```
        ResultsEARAll$MW<-c(ResultsEARAll$MW,NA)
        names(ResultsEARAll$MW)[length(ResultsEARAll$MW)]<-'Sum'
    #The AC data structure
        ResultsEARAll$AC50s<-cbind(ResultsEARAll$AC50s,NA)


#Update EARCube with Sum
    AssaySum<-apply(ResultsEARAll$EARCube, c(1,3),sum,na.rm=TRUE)
    ResultsEARAll$EARCube<-abind(ResultsEARAll$EARCube,AssaySum,along = 2)
#UnderLimit
    ResultsEARAll$UnderLimit<-abind(ResultsEARAll$UnderLimit,AssaySum*0,along = 2)

return(ResultsEARAll)
}
##############################################################################################
############################

FilterEARDataAll<-function(ResultsEARAll,Sites,Chemicals,Assay,Threshold=NA,NonDetect,MCrit){
#Filters the EAR results
    #Removes site, chemicals, assays not selected
    #Removes Rows and columns that contain now information
#ResultsEARAll Results from CalculatedAllEAR
#Sites array of sites selected can be 'All'
    #Must be a sample record number
#Chemicals array of chemicals selected can be 'All'
#Assay array of assays selected can be 'All'
    FilteredData<-ResultsEARAll;
    options(warn=-1) #Turn off warnings as they are know
#Filter out non-selected entries
    if (Sites !='All'){
        KeepSites<-sapply(Sites,Which.Apply,ResultsEARAll$SiteData[['SAMPLE RECORD NUMBER']])
        KeepSites<-sort(as.vector(unlist(KeepSites)))
        #Check to see if there is a matching site in the data set
            if (length(KeepSites)==0){
                message('Please Select a Site in the Data Set')
                return(NULL)
            }
        #Remove Non-selected Sites
            FilteredData$EARCube<-FilteredData$EARCube[KeepSites, , ]
            FilteredData$SiteData<-FilteredData$SiteData[KeepSites, ]

        #Prep UnderLimit
            FilteredData$UnderLimit<-FilteredData$UnderLimit[KeepSites, , ]

    }
    if (Chemicals !='All'){
        KeepChemicals<-sapply(Chemicals,Which.Apply,ResultsEARAll$ChemicalNames[ ,2])
        KeepChemicals<-sort(as.vector(unlist(KeepChemicals)))
        #Check to see if there is a matching Chemical in the data set
            if (length(KeepChemicals)==0){
                message('Please Select a Chemicals in the Data Set')
                return(NULL)
            }
```

```
        #Remove Non-selected Sites
            if (length(dim(FilteredData$EARCube))==2){
                FilteredData$EARCube<-FilteredData$EARCube[KeepChemicals, ]

                #UnderLimit
                    FilteredData$UnderLimit<-FilteredData$UnderLimit[KeepChemicals , ]
            }
            if (length(dim(FilteredData$EARCube))==3){
                FilteredData$EARCube<-FilteredData$EARCube[ ,KeepChemicals, ]
                #UnderLimit
                    FilteredData$UnderLimit<-FilteredData$UnderLimit[ ,KeepChemicals, ]

            }

            FilteredData$ChemicalNames<-FilteredData$ChemicalNames[KeepChemicals, ]
            FilteredData$AC50s<-FilteredData$AC50s[ ,KeepChemicals]
            FilteredData$MW<-FilteredData$MW[KeepChemicals]

    }
    if (Assay !='All'){
        #Correct for the change in the data structure when on one chemical is present
        if (is.null(rownames(FilteredData$AC50s))==FALSE){
            KeepAssay<-sapply(Assay,Which.Apply,rownames(FilteredData$AC50s))
            KeepAssay<-sort(as.vector(unlist(KeepAssay)))
            FilteredData$AC50s<-FilteredData$AC50s[KeepAssay, ]
        }else{
            KeepAssay<-sapply(Assay,Which.Apply, names(FilteredData$AC50s))
            KeepAssay<-sort(as.vector(unlist(KeepAssay)))
            FilteredData$AC50s<-FilteredData$AC50s[KeepAssay]
        }


        #Check to see if there is a matching Chemical in the data set
            if (length(KeepAssay)==0){
                message('Please Select a Assay in the Data Set')
                return(NULL)
            }
        #Remove Non-selected Sites
            if (length(dim(FilteredData$EARCube))==2){
                FilteredData$EARCube<-FilteredData$EARCube[  ,KeepAssay]
                #UnderLimit
                    FilteredData$UnderLimit<-FilteredData$UnderLimit[ ,KeepAssay]
            }
            if (length(dim(FilteredData$EARCube))==3){
                FilteredData$EARCube<-FilteredData$EARCube[ , ,KeepAssay]
                #UnderLimit
                    FilteredData$UnderLimit<-FilteredData$UnderLimit[ , ,KeepAssay]
            }
    }

    #Find Min/Max if applicable
        if (MCrit != 'Ignore'){
            FilteredData<-CollapseData(FilteredData,MCrit,NonDetect)
```

```
        }


    #Figure out plotting dimensions
        if (is.null(dim(FilteredData$AC50s))==TRUE){
            TitleD<-'Assay' #Site,Chemical    #priority 3
            if (MCrit !='Assay - Largest' && MCrit !='Assay - Smallest' && MCrit !='Assay - Sum'
            ){
                TitleAdd<-Assay
            }else{
                TitleAdd<-strsplit(MCrit,' - ')[[1]][2]
            }
            FilteredData[['TitleAdd']]<-TitleAdd
            RowName<-'Site'
            ColName<-'Chemical'
        }

        if (length(FilteredData$MW)==1){
            TitleD<-'Chemical' #Site, Assay    #priority 2
            RowName<-'Site'
            ColName<-'Assay'
        }

        if (dim(FilteredData$SiteData)[1]==1){
            TitleD<-'Site'   #Chemical, Assay  #priority 1
            RowName<-'Chemical'
            ColName<-'Assay'
        }


#EARCube Starts as Site,Chemical,Assay
        #ends as Site,Chemical
        #or  Site, Assay
        #or  Chemical, Assay
#FilteredData$EARCube should be a matrix now
#Next Step is to remove all row and columns that contain nothing but NA
    #Get rows and columns to be removed
        RemoveCols<-which(colSums(is.finite(FilteredData$EARCube))==0)
        RemoveRows<-which(colSums(t(is.finite(FilteredData$EARCube)))==0)
#Filter Out Based on Threshold
    if (is.finite(Threshold)==TRUE){
        RemoveCols<-c(RemoveCols,which(colSums(FilteredData$EARCube>Threshold,na.rm=TRUE)==0))
            RemoveCols<-unique(RemoveCols)
        RemoveRows<-c(RemoveRows,which(colSums(t(FilteredData$EARCube)>Threshold,na.rm=TRUE)==0))
            RemoveRows<-unique(RemoveRows)
    }

    #Remove rows and columns and #Update Stored Information
        if (length(RemoveCols)>0){
            FilteredData$EARCube<-FilteredData$EARCube[ ,-RemoveCols];
            #Apply Removal to UnderLimit
                FilteredData$UnderLimit<-FilteredData$UnderLimit[ ,-RemoveCols]
            if (ColName=='Chemical'){
```

```
            RemoveListCol<-names(FilteredData$MW[RemoveCols])    #Chemical

            FilteredData$ChemicalNames<-FilteredData$ChemicalNames[-RemoveCols, ]
            if (is.null(dim(FilteredData$AC50s))==TRUE){
                FilteredData$AC50s<-FilteredData$AC50s[-RemoveCols]
            }else{
                FilteredData$AC50s<-FilteredData$AC50s[ ,-RemoveCols]
            }
            FilteredData$MW<-FilteredData$MW[-RemoveCols]



        }
        if (ColName=='Assay'){
            RemoveListCol<-names(FilteredData$AC50s[RemoveCols])#Assay
            if (is.null(dim(FilteredData$AC50s))==TRUE){
                FilteredData$AC50s<-FilteredData$AC50s[-RemoveCols];
            }else{
                FilteredData$AC50s<-FilteredData$AC50s[-RemoveCols, ];
            }

        }
        FilteredData[['RemoveListCol']]<-RemoveListCol

    }
    if (length(RemoveRows)>0){
        FilteredData$EARCube<-FilteredData$EARCube[-RemoveRows, ];
        #Apply Removal to UnderLimit
            FilteredData$UnderLimit<-FilteredData$UnderLimit[-RemoveRows, ]

        if (RowName=='Site'){ #Chemical, Assay
            RemoveListRow<-FilteredData$SiteData[-RemoveRows,'USGS STATION NAME']  #Site

            FilteredData$SiteData<-FilteredData$SiteData[-RemoveRows, ]
        }
        if (RowName=='Chemical'){
            RemoveListRow<-names(FilteredData$MW[RemoveRows]) #Chemical

            FilteredData$ChemicalNames<-FilteredData$ChemicalNames[-RemoveRows, ]
            FilteredData$AC50s<-FilteredData$AC50s[ ,-RemoveRows]
            FilteredData$MW<-FilteredData$MW[-RemoveRows]
        }
        FilteredData[['RemoveListRow']]<-RemoveListRow
    }


FilteredData[['RowName']]<-RowName
FilteredData[['ColName']]<-ColName
FilteredData[['TitleName']]<-TitleD

if(length(dim(FilteredData$EARCube))<2){
    pop_message('Your Selection criteria has caused the data to only have 1 dimension.\n
```

```
        Because of this the EAR calculator can not produce a plot.')
    }

return(FilteredData)
}
##############################################################################################
############################

CollapseData<-function(FilteredData,MCrit,NonDetect){
#Collapses the data set to take the largest or smallest value of a dimension
#This function triggers if criteria for multiple selection is not set to ignore
#EarCube must still have three dimensions for this to work
#MCrit is the Criteria for multiple selections




#Check dimensions of EAR Cube
    if(length(dim(FilteredData$EARCube))<3){
        return(FilteredData)
    }

#Remove "Sum" from chemicals if applicable
    if (length(which(FilteredData$Chemical[ ,1]=='Sum')) > 0){
        #Last value of chemical dimension is sum
        FilteredData$EARCube<-FilteredData$EARCube[ ,-dim(FilteredData$EARCube)[2], ]

    }


#Assay
    if (MCrit=='Assay - Largest'){
        Dim<-c(1,2)
        fun<-max
        FilteredData$AC50s<-'Largest'
    }
    if (MCrit=='Assay - Smallest'){
        Dim<-c(1,2)
        fun<-min
        FilteredData$AC50s<-'Smallest'
    }
    if (MCrit=='Assay - Sum'){
        Dim<-c(1,2)
        fun<-sum
        FilteredData$AC50s<-'Sum'
    }
#Site
    if (MCrit=='Site - Largest'){
        Dim<-c(2,3)
        fun<-max
        FilteredData$SiteData[1,1:2]<-'Largest'
        FilteredData$SiteData[1,3:4]<-''
        FilteredData$SiteData[1,5:6]<-'Largest'
```

```
        FilteredData$SiteData<-FilteredData$SiteData[1, ]
    }
    if (MCrit=='Site - Smallest'){
        Dim<-c(2,3)
        fun<-min
        FilteredData$SiteData[1,1:2]<-'Smallest'
        FilteredData$SiteData[1,3:4]<-''
        FilteredData$SiteData[1,5:6]<-'Smallest'
        FilteredData$SiteData<-FilteredData$SiteData[1, ]
    }
    if (MCrit=='Site - Sum'){
        Dim<-c(2,3)
        fun<-sum
        FilteredData$SiteData[1,1:2]<-'Sum'
        FilteredData$SiteData[1,3:4]<-''
        FilteredData$SiteData[1,5:6]<-'Sum'
        FilteredData$SiteData<-FilteredData$SiteData[1, ]
    }



#Chemical
    if (MCrit=='Chemical - Largest'){
        Dim<-c(1,3)
        fun<-max
        #FilteredData$ChemicalNames<-c('Largest','Largest')
        #FilteredData$MW<-c('Largest','Largest')
        #names(FilteredData$MW)<-c('Largest','Largest')
        FilteredData$ChemicalNames<-c('Largest')
        FilteredData$MW<-c('Largest')
        names(FilteredData$MW)<-c('Largest')
    }
    if (MCrit=='Chemical - Smallest'){
        Dim<-c(1,3)
        fun<-min
        #FilteredData$ChemicalNames<-c('Smallest','Smallest')
        #FilteredData$MW<-c('Smallest','Smallest')
        #names(FilteredData$MW)<-c('Smallest','Smallest')
        FilteredData$ChemicalNames<-c('Smallest')
        FilteredData$MW<-c('Smallest')
        names(FilteredData$MW)<-c('Smallest')
    }
    if (MCrit=='Chemical - Sum'){
        Dim<-c(1,3)
        fun<-sum
        #FilteredData$ChemicalNames<-c('Smallest','Smallest')
        #FilteredData$MW<-c('Smallest','Smallest')
        #names(FilteredData$MW)<-c('Smallest','Smallest')
        FilteredData$ChemicalNames<-c('Sum')
        FilteredData$MW<-c('Sum')
        names(FilteredData$MW)<-c('Sum')
    }
```

```
#get min or max
    Square<-apply(FilteredData$EARCube,Dim,fun,na.rm=TRUE)
    if (length(which(!is.finite(Square)))>0){
        Square[which(!is.finite(Square))]<-NA
    }



    #Get the Values of UnderLimit
        #Double for loop :(
        #Vectorize later if time allows
        Size<-dim(Square)
        UnderSquare<-mat.or.vec(Size[1],Size[2])  #Sets this to all zeros if UnderSquare is not
        used
    #Only change UnderSquare if it is used for a Circle (This is done for speed)
        if (NonDetect=='Circle'){
        #Find corresponding Under Limit
            UnderCube<-FilteredData$UnderLimit
            UnderSquareLoc<-apply(FilteredData$EARCube,Dim,WhichOverMinMax,fun)
            for (i in 1:Size[1]){
                for (j in 1:Size[2]){
                    UnderLoc<-UnderSquareLoc[i,j][[1]]
                        if (length(UnderLoc)>1){
                            Values<-sapply(UnderLoc,GetValueOverCube,i,j,UnderCube,Dim)
                            UnderLoc<-UnderLoc[max(which(Values==min(Values)))]
                        }
                        if (is.na(UnderLoc)==FALSE){
                            UseLoc<-GetIndexVec(c(i,j,UnderLoc),Dim)
                            UnderSquare[i,j]<-UnderCube[UseLoc[1],UseLoc[2],UseLoc[3]]
                        }
                }
            }
        }
#Set into Data set
    FilteredData$EARCube<-Square
    FilteredData$UnderLimit<-UnderSquare

return(FilteredData)
}
##############################################################################################
############################

WhichOverMinMax<-function(Data,fun){
#Which function for Min, max, mean, any find function that can take na.rm as an argument
    Out<-which(Data==fun(Data,na.rm=TRUE))
    if (length(Out)==0){
        return(NA)
    }
return(Out)
}
##############################################################################################
############################
```

```
GetValueOverCube<-function(Val,i,e,UnderCube,Dim){
#Gets the value of UnderCube at location [i,e,Val]
#Modified by GetIndexVec
    Index<-GetIndexVec(c(i,e,Val),Dim)
    return(UnderCube[Index[1],Index[2],Index[3]])
}
##############################################################################################
############################

GetIndexVec<-function(VecIn,Dim){
#Gets an index vector based on Dim
#Used in CollapseData to adjust for the changing dimensions of the 3D vector
#Vec in contains the values for Dim[1],Dim[2], then the max/min Dim

#Initiate Vec Out
    VecOut<-rep(0,3)
#Resort
    VecOut[Dim[1]]<-VecIn[1]
    VecOut[Dim[2]]<-VecIn[2]
    VecOut[which(VecOut==0)]<-VecIn[3]
return(VecOut)
}
##############################################################################################
############################

SortDataAll<-function(FilteredData,SCrit='Mean'){
#Sorts the data to be plotted
#Only Called by MainPlotEARAll
#SCrit is the criteria to sort on
#FilteredData is data filtered by the previous step
#Get order
    Data<-FilteredData$EARCube
    WieghtVec1<-apply(Data,1,CalcChemWieght,SCrit) #Rows
    WieghtVec2<-apply(Data,2,CalcChemWieght,SCrit) #Columns
#Apply order to plotted data
    Data<-Data[ ,order(WieghtVec2, decreasing=TRUE)] #Columns
    Data<-Data[order(WieghtVec1, decreasing=TRUE), ] #Rows
#Apply order to Under Limit
    FilteredData$UnderLimit<-FilteredData$UnderLimit[ ,order(WieghtVec2, decreasing=TRUE)]
    FilteredData$UnderLimit<-FilteredData$UnderLimit[order(WieghtVec1, decreasing=TRUE), ]

    FilteredData$EARCube<-Data
#Update Identifiers with With Order
    if (FilteredData$RowName=='Site'){
        FilteredData$SiteData<-FilteredData$SiteData[order(WieghtVec1, decreasing=TRUE), ]
    }
    if (FilteredData$RowName=='Chemical'){
        FilteredData$ChemicalNames<-FilteredData$ChemicalNames[order(WieghtVec1, decreasing=TRUE
        ), ]
        FilteredData$AC50s<-FilteredData$AC50s[ ,order(WieghtVec1, decreasing=TRUE)]
        FilteredData$MW<-FilteredData$MW[order(WieghtVec1, decreasing=TRUE)]

    }
```

```
    if (FilteredData$ColName=='Chemical'){
        FilteredData$ChemicalNames<-FilteredData$ChemicalNames[order(WieghtVec2, decreasing=TRUE
        ), ]
        if (is.null(dim(FilteredData$AC50s))==TRUE){
            FilteredData$AC50s<-FilteredData$AC50s[order(WieghtVec2, decreasing=TRUE)]
        }else{
            FilteredData$AC50s<-FilteredData$AC50s[ ,order(WieghtVec2, decreasing=TRUE)]
        }
        FilteredData$MW<-FilteredData$MW[order(WieghtVec2, decreasing=TRUE)]
    }

    if (FilteredData$ColName=='Assay'){
        if (is.null(dim(FilteredData$AC50s))==TRUE){
            FilteredData$AC50s<-FilteredData$AC50s[order(WieghtVec2, decreasing=TRUE)]
        }else{
            FilteredData$AC50s<-FilteredData$AC50s[order(WieghtVec2, decreasing=TRUE), ]
        }
    }

return(FilteredData)
}
##############################################################################################
############################

PlotEARAll<-function(Data,AddLegend=FALSE,SwitchAxis=FALSE,Replot=FALSE,XMin=NULL,XMax=NULL,YMin
=NULL,YMax=NULL,PchTy='GUI',NonDetect='Nothing'){
#This function plots EAR results it uses something similar to a Cleveland dot plot

#Get Y-axis (Column of EARCube)
    YAxis<-Data$ColName
    if (YAxis=='Chemical'){
        YLabels<-Data$ChemicalNames[ ,1]
        YLabels<-unlist(lapply(YLabels,ApplyCharLimit,32,TRUE))
        DimGraphY<-length(YLabels)
    }
    if (YAxis=='Assay'){
        YLabels<-rownames(Data$AC50s)
        if (is.null(YLabels)==TRUE){
            YLabels<-names(Data$AC50s)
        }
        YLabels<-unlist(lapply(YLabels,ApplyCharLimit,32,FALSE))
        DimGraphY<-length(YLabels)

    }
    if (YAxis=='Site'){
        SiteData<-Data$SiteData
        FieldNames<-Data[['FIELD NAME']]
            FieldNames<-unlist(lapply(FieldNames,ApplyCharLimit,14,FALSE))
        Dates<-Data[['DATE COLLECTED']]
            Dates<-unlist(lapply(Dates,ApplyCharLimit,10,FALSE))
        Times<-Data[['TIME COLLECTED']]
            Times<-unlist(lapply(Times,ApplyCharLimit,5,FALSE))
```

```
        DimGraphY<-dim(SiteData)[1]
    }
#Get X-axis (Column of EARCube)
    XAxis<-Data$RowName
    if (XAxis=='Chemical'){
        XLabels<-Data$ChemicalNames[ ,1]
        XLabels<-unlist(lapply(XLabels,ApplyCharLimit,35,TRUE))
        DimGraphX<-length(XLabels)
    }
    if (XAxis=='Assay'){
        XLabels<-rownames(Data$AC50s)
        XLabels<-unlist(lapply(XLabels,ApplyCharLimit,35,FALSE))
        DimGraphX<-length(XLabels)
    }
    if (XAxis=='Site'){
        SiteData<-Data$SiteData
        FieldNames<-SiteData[['FIELD NAME']]
            FieldNames<-unlist(lapply(FieldNames,ApplyCharLimit,14,FALSE))
        Dates<-SiteData[['DATE COLLECTED']]
            Dates<-unlist(lapply(Dates,ApplyCharLimit,10,FALSE))
        Times<-SiteData[['TIME COLLECTED']]
            Times<-unlist(lapply(Times,ApplyCharLimit,5,FALSE))
        XLabels<-paste(FieldNames,Dates,Times)
        DimGraphX<-dim(SiteData)[1]
    }
    #Switch Axes if need be
    if (SwitchAxis==TRUE){
        TAxis<-XAxis
        TLabels<-XLabels
        DimGraphT<-DimGraphX
        XAxis<-YAxis
        XLabels<-YLabels
        DimGraphX<-DimGraphY
        YAxis<-TAxis
        YLabels<-TLabels
        DimGraphY<-DimGraphT
        Data$EARCube<-t(Data$EARCube)
        Data$UnderLimit<-t(Data$UnderLimit)
    }

    XLabels<-unlist(lapply(XLabels,UnderToWhite))
    YLabels<-unlist(lapply(YLabels,UnderToWhite))

    #Build Title
    Title<-'EAR Results for '
    TitleAdd<-switch(Data$TitleName,
        "Chemical"=paste("the Chemical: ",names(Data$MW)[1],sep=''),
        "Assay"=paste("the Assay: ",Data$TitleAdd,sep=''),
        "Site"=paste("the Site:",  Data$SiteData[2],Data$SiteData[3],Data$SiteData[4])
    )
    Title<-paste(Title,TitleAdd,sep=' ')
    EARCube<-Data$EARCube
```

```
#Apply Dimension Restrictions
#Test Restrictions
    if (is.null(XMin)==FALSE && is.null(XMax)==FALSE && is.null(XMin)==FALSE && is.null(XMax)==
    FALSE){
        if (XMin>DimGraphX){
            XMin=DimGraphX;
        }
        if (YMin>DimGraphY){
            YMin=DimGraphY;
        }
        if (XMax>DimGraphX){
            XMax=DimGraphX;
        }
        if (YMax>DimGraphY){
            YMax=DimGraphY;
        }
        EARCube<-EARCube[XMin:XMax,YMin:YMax]
        DimGraphX=XMax-XMin+1
        DimGraphY=YMax-YMin+1
        XLabels<-XLabels[XMin:XMax]
        YLabels<-YLabels[YMin:YMax]
        UnderLimit<-Data$UnderLimit[XMin:XMax,YMin:YMax]
    }



#Set Text and pt Size
    cexY=0.50  #(0.3 to 0.5)
        cexY=max(0.3, 0.5-0.002*DimGraphY)
    cexX=0.50  #(0.3 to 0.5)
        cexX=max(0.3, 0.5-0.002*DimGraphX)
    CexPt=1    #(0.6 to 1.0)
        CexPt=max(0.6, 0.1-0.004*CexPt)

#Make the plot
    if (Replot==FALSE){
        par('mai'=par('mai')*c(2,2,1,0.75))
    }
    plot(c(1,DimGraphX),c(1, DimGraphY), type="n", axes=FALSE, frame.plot=TRUE, xlab="", ylab=""
    ,main=Title)
    axis(2, 1:DimGraphY,YLabels,labels = FALSE,pos=par("usr")[1],tck=-.01)
    text(y = 1:DimGraphY, par("usr")[1], labels = YLabels, srt = 0, pos = 2,offset=0.3 , xpd =
    TRUE,cex=cexY)

    axis(1, 1:DimGraphX,XLabels,labels = FALSE,pos=par("usr")[3],tck=-.01)
    text(x = 1:DimGraphX, par("usr")[3]-0.02*DimGraphY, labels = XLabels, srt = 90, pos = 2,
    offset=0.01 , xpd = TRUE,cex=cexX)

#Set the plotting symbols
    Getpch<-switch(PchTy,
        PDF=GetpchPDF,
        GUI=GetpchGUI
    )
```

```
#Adjust the UnderLimit for NAs
    if (length(which(is.na(EARCube)==TRUE))>0){
        EARCubeNA<-EARCube
        EARCubeNA[which(is.na(EARCube)==TRUE)]<-0
        UnderLimit<-UnderLimit*(EARCubeNA!=0)
    }

#Apply the UnderLimit(Uses Ranges)
#Changed for PlottingCat.EAR
    PointsNA<-GetLocation(EARCube,0,0,TRUE)
    Points1<-GetLocation(EARCube,0,PlottingCat.EAR[1],FALSE)
    Points2<-GetLocation(EARCube,PlottingCat.EAR[1],PlottingCat.EAR[2],FALSE)
    Points3<-GetLocation(EARCube,PlottingCat.EAR[2],PlottingCat.EAR[3],FALSE)
    Points4<-GetLocation(EARCube,PlottingCat.EAR[3],10^10,FALSE)

#Plot each category (Uses Ranges)
#Changed to reflect changes in the EAR categories
    points(PointsNA,pch=Getpch(NA),col=Getcol(NA),cex=CexPt)
    points(Points1,pch=Getpch(0),col=Getcol(0),cex=CexPt)
    points(Points2,pch=Getpch(PlottingCat.EAR[1]),col=Getcol(PlottingCat.EAR[1]),cex=CexPt)
    points(Points3,pch=Getpch(PlottingCat.EAR[2]),col=Getcol(PlottingCat.EAR[2]),cex=CexPt)
    points(Points4,pch=Getpch(PlottingCat.EAR[3]),col=Getcol(PlottingCat.EAR[3]),cex=CexPt)

#Add Circle to plots if need be
    if (NonDetect=='Circle'){
        Points5<-GetLocation(UnderLimit,0.9,1.1,FALSE)
        points(Points5,pch=1,col='black',cex=CexPt*1.2)
    }

#Add legend, only used when saving as a PDF
#Changed for PlottingCat.EAR
    if (AddLegend==TRUE){
        par(xpd=TRUE)
        if (NonDetect!='Circle'){
            legend(x=par("usr")[1],y=par("usr")[3], cex = 0.75,
                c('No Calculation',
                paste0('EAR < ',PlottingCat.EAR[1]),
                paste0(PlottingCat.EAR[1],' <= EAR < ', PlottingCat.EAR[2]),
                paste0(PlottingCat.EAR[2], ' <= EAR < ', PlottingCat.EAR[3]),
                paste0(' EAR >= ',PlottingCat.EAR[3])),
                col=c('black','black','blue','green','red'), pch=c(4,20,16,16,16),xjust=1,yjust=
                1)
        }
        if (NonDetect=='Circle'){
            legend(x=par("usr")[1],y=par("usr")[3], cex = 0.75,
                c('No Calculation',
                paste0('EAR < ',PlottingCat.EAR[1]),
                paste0(PlottingCat.EAR[1],' <= EAR < ', PlottingCat.EAR[2]),
                paste0(PlottingCat.EAR[2], ' <= EAR < ', PlottingCat.EAR[3]),
                paste0(' EAR >= ',PlottingCat.EAR[3]),
                '<Detection Limit'),
                col=c('black','black','blue','green','red','black'), pch=c(4,20,16,16,16,1),
```

```
                xjust=1,yjust=1)
        }
    }

}
#############################################################################################################
############################

GetpchPDF<-function(Num){
#gets the plotting symbol for PlotEAR
#Now uses global PlottingCat.EAR
    if (is.na(Num)==TRUE){
        return('x')
    }
    if (Num < PlottingCat.EAR[1]){
        return('.')
    }
    if (Num >= PlottingCat.EAR[1]){
        return(16) #R for filled in circle
    }
message('Error Unknown Value For EAR')
return('x')
}
#############################################################################################################
############################

GetpchGUI<-function(Num){
#gets the plotting symbol for PlotEAR
#Now uses global PlottingCat.EAR
#last updated 12-24-2015 JS
    if (is.na(Num)==TRUE){
        return('x')
    }
    if (Num < PlottingCat.EAR[1]){
        return('.')
    }
    if (Num >= PlottingCat.EAR[3]){
        return(13) #R for filled in circle
    }
    if (Num >= PlottingCat.EAR[2]){
        return(19) #R for filled in circle
    }

    if (Num >= PlottingCat.EAR[1]){
        return(20) #R for filled in circle
    }

message('Error Unknown Value For EAR')
return('x')
}
#############################################################################################################
############################
```

```
Getcol<-function(Num){
#gets the plotting color for PlotEAR
#Now uses global PlottingCat.EAR
#last updated 12-24-2015 JS
    if (is.na(Num)==TRUE){
        return('black')
    }
    if (Num < PlottingCat.EAR[1]){
        return('black')
    }
    if (Num >= PlottingCat.EAR[1] && Num < PlottingCat.EAR[2]){
        return('blue')
    }
    if (Num >= PlottingCat.EAR[2] && Num < PlottingCat.EAR[3]){
        return('green')
    }
    if (Num >= PlottingCat.EAR[3]){
        return('red')
    }
message('Error Unknown Value For EAR')
return('black')
}
################################################################################################################
################################

SavePlotEARAll<-function(SortedData,File,SwitchAxis,XMin,XMax,YMin,YMax,PchTy,NonDetect){
#Saves the plot of EAR Results
pdf(File)
    PlotEARAll(SortedData,TRUE,SwitchAxis,FALSE,XMin,XMax,YMin,YMax,'PDF',NonDetect)

dev.off()
}
################################################################################################################
################################

CalcChemWieght<-function(Vec,SCrit){
#Calculates a rank-able metric for a vector vector
#Used to determine A Chemicals average impact from an apply function
#uses global PlottingCat.EAR
Wieght<-switch(SCrit,
    Mean   = mean(Vec,na.rm = TRUE),
    Sum    = sum(Vec,na.rm = TRUE),
    Median =median(Vec,na.rm = TRUE),
    WMedain=median(Vec,na.rm = TRUE)*sum(is.na(Vec)==FALSE),
    Active=sum(is.finite(Vec)),
    Red=Sum((Vec >= PlottingCat.EAR[3]),na.rm = TRUE),
    Color=1000000*sum((Vec >= PlottingCat.EAR[3]),na.rm = TRUE)+10000*sum((Vec >=
    PlottingCat.EAR[2]),na.rm = TRUE)+100*sum((Vec >= PlottingCat.EAR[1]),na.rm = TRUE)+sum((Vec
     >= 0),na.rm = TRUE)
)
#Color Changed 12-24-205 JS

return(Wieght)
```

```
}

#################################################################################################
#######

ApplyCharLimit<-function(String,Limit,IsChemical){
#Apply a Character limit, removes right white space, pads left white space
    #IsChemical is a bool for chemical names
    #Uses Global ChemInfoList.EAR

#Filter right white space
    Split<-strsplit(String,split='')[[1]]
    WhiteNum<-which(Split==' ')
    CharNum<-which(Split!=' ')
    Remove<-WhiteNum[which(WhiteNum > max(CharNum))]
    if (length(Remove)>0){
        Split<-Split[-Remove]
    }
#Truncate String larger then the limit
    if (length(Split) > Limit){
        NewString<-paste(Split[1:Limit],collapse="")

        #Replace Chemical name with formula is a chemical is above the limit
        if (IsChemical==TRUE){
            ChemName<-paste(Split,collapse="")
            ChemName<-as.character(ChemInfoList.EAR$Formula[which(ChemInfoList.EAR$Names==
            ChemName)])
            if (length(ChemName)>0){
                Split<-strsplit(ChemName,split='')[[1]]
            }
        }
    }
#Left pad white to ensure a constant limit
    if (length(Split) < Limit){
        WhiteSpace<-paste(rep(' ',Limit-length(Split)),collapse="")
        NewString<-paste(Split,collapse="")
        NewString<-paste(WhiteSpace,NewString,sep='')
    }
#If at limit do nothing
    if(length(Split) == Limit){
        NewString<-paste(Split,collapse="")
    }
return(NewString)
}
#################################################################################################
#######

UnderToWhite<-function(Str){
#Changes underscores in a string to while space
    SpStr<-strsplit(Str,'')[[1]]
    Under<-which(SpStr=='_')
    if (length(Under)>0){
        SpStr[Under]<-' '
```

```
    }
    Str<-paste(SpStr,collapse = "")
return(Str)
}
#############################################################################################
#######

GetLocation<-function(EARCube,ValueMin,ValueMax,UseNA){
#Attains the locations within EAR Cube that are between
    NRows<-dim(EARCube)[1]
    if (UseNA==TRUE){
        Locs<-which(is.na(EARCube)==TRUE)
    }else{
        Locs<-which(EARCube >= ValueMin & EARCube < ValueMax)
    }
    X<-Locs %% NRows
    if (length(X[X==0])>0){
        X[X==0]<-NRows
    }
    Y<-ceiling(Locs/NRows)

    Points<-cbind(X,Y)
return(Points)
}
#############################################################################################
#######

Which.Apply<-function(Value,Vec){
#Which to be called though an apply function
    N<-which(Vec==Value)
    if (length(N)>0){
        return(N)
    }
#Return Null is nothing is found
return(NULL)
}
#############################################################################################
#######
```