```
## R code for building PLSR models using 100 different wavelength ranges
## 'Spectral wavelength range influences the performance of chemometric models estimating various foliar
functional traits'
## Authors: Minjee Park, Lorenzo Cotrozzi, Geoffrey M Williams, Matthew D Ginzel, Michael V Mickelbart,
Douglass F Jacobs, John J Couture


  #install.packages("reshape")
  #install.packages("pls")
  library(reshape)
  library(pls)

  ############################################## FUNCTIONS

  fxp<- function(saveDir, inVar, y, pressDFres){
    png(paste(saveDir,inVar,"_",y,"_findComponents.png",sep=""), type = "cairo",
width=6,height=5,units="in",res=200) ;
    boxplot(pressDFres$value~pressDFres$variable,xlab="n Components",ylab="PRESS",main=inVar);
    dev.off()
  }

  VIPjh=function(object, j, h)
        {
        if (object$method != "oscorespls") stop("Only implemented for orthogonal scores algorithm. Refit with
'method = \"oscorespls\"')
        if (nrow(object$Yloadings) > 1) stop("Only implemented for single-response models")
        b=c(object$Yloadings)[1:h]
        T=object$scores[,1:h, drop = FALSE]
        SS=b^2 * colSums(T^2)
        W=object$loading.weights[,1:h, drop = FALSE]
        Wnorm2=colSums(W^2)
        VIP=sqrt(nrow(W) * sum(SS * W[j,]^2 / Wnorm2) / sum(SS))
        return(VIP)
        }

  ############################################## DEFAULTS
  inVar= "Sugars" # Foliar trait, e.g., Sugars
  baseDir="D:/Purdue_Research/"
  saveDir=paste(baseDir,"PLSR_Outputs_",inVar,"/", sep="")

  # Input dataset
  allData=read.csv(paste(baseDir,"2018+2019_ChemicalData_",inVar,"_CalVal.csv", sep="")) # Reference foliar
trait+ spectral data
  WR<-read.csv(paste(baseDir,"Wavelength_ranges_PLSR.csv", sep="")) # One hundred wavelength ranges

  lowest_NComps <- data.frame(Loop = integer(), Lowest_NComps = integer()) #Saving the number of components
based on the lowest PRESS
  Stats_combined=matrix(data=NA,nrow=nrow(WR),ncol=6) #Stat outcomes across PLSR models using different
wavelength ranges

  for (y in 1:nrow(WR)){
  #...Select spectral data
  s=as.numeric(WR[y,2]) # starting wavelength
  f=as.numeric(WR[y,3]) # ending wavelength


  if((s<1912)&(f>1850)){  #removing from 1891 to 1911 nm because of noise associated with the SWIR 1 and SWIR
2 detector overlap region.
    inBands=c(paste("X",c(seq(s,1890,1),seq(1912,f,1)),sep=""))
  } else {
    inBands=c(paste("X",seq(s,f,m),sep=""))
  }

  iForm=paste(inBands,collapse="+")
  iForm=as.formula(paste(inVar,"~",iForm))
  iForm

  ############################################## FIND COMPONENTS

  ## RANDOMIZE TO GET OPTIMAL NCOMP
  ## USE nCut OF DATA TO BUILD MODELS
  ## USE nsims CVs

  nCut=floor(.8*nrow(allData))
  nComps=30
  nsims=100
  outMat=matrix(data=NA,nrow=nsims,ncol=nComps)
  for (nsim in seq(nsims))
        {
        print(nsim)
        flush.console()
        allData$RAND=order(runif(nrow(allData)))
        subData=allData[allData["RAND"]<nCut,]
        resNCOMP=plsr(iForm,data=subData,ncomp=nComps,validation="LOO",method="oscorespls")
        resPRESS=as.vector(resNCOMP$validation$PRESS) # the predicted residual sum of square
        outMat[nsim,seq(resNCOMP$validation$ncomp)]=resPRESS
```

```
        }
  pressDF=as.data.frame(outMat)
  names(pressDF)=as.character(seq(nComps))
  pressDFres=melt(pressDF)
  fxp(saveDir, inVar, y, pressDFres)

  #NComps which has the lowest average of PRESS

  avg_press <- colMeans(pressDF)
  r <- which.min(avg_press)
  lowest_NComps <- rbind(lowest_NComps, data.frame(Loop = y, Lowest_NComps = r))

  }

  write.csv(lowest_NComps,paste(saveDir,inVar,"_NComponents.csv",sep=""), row.names = FALSE)


######################################################## FIT MODELS

Dataset<-read.csv(paste(baseDir,"Wavelength_ranges_PLSR_",inVar,".csv", sep="")) #This dataset includes the
starting and ending wavelengths [WR], along with the number of components determined by the lowest PRESS
value [Lowest].

for (y in 1:nrow(Dataset)){

  s=as.numeric(Dataset[y,2]) #the starting (s) # the ending wavelengths (f)
  f=as.numeric(Dataset[y,3])
  com=as.numeric(Dataset[y,4]) #the number of components determined by the lowest PRESS value (com).

  if((s<1912)&(f>1850)){
    inBands=c(paste("X",c(seq(s,1890,1),seq(1912,f,1)),sep=""))
  } else {
    inBands=c(paste("X",seq(s,f,1),sep=""))
  }

  iForm=paste(inBands,collapse="+")
  iForm=as.formula(paste(inVar,"~",iForm))
  iForm

nsims=100 #For the final model using an optimal wavelength range, nsims=500
nComps=com
nCut=floor(.8*nrow(allData))
coefMat=matrix(data=NA,nrow=nsims,ncol=length(inBands)+1)
coefStd=matrix(data=NA,nrow=nsims,ncol=length(inBands)+1)
vipMat=matrix(data=NA,ncol=nsims,nrow=length(inBands))

statMat=matrix(data=NA,nrow=nsims,ncol=6)
for (nsim in seq(nsims))
        {
        print(nsim)
        flush.console()
        allData$RAND=order(runif(nrow(allData)))
        subData=allData[allData["RAND"]<nCut,]
        tstData=allData[allData["RAND"]>=nCut,]
        nSub=nrow(subData)

        resX=plsr(iForm,data=subData,ncomp=nComps,method="oscorespls")
        resS=plsr(iForm,data=subData,ncomp=nComps,method="oscorespls",scale=T)

        # GET COEFFICIENTS (raw and standardized)
        coefs=as.vector(coef(resX,ncomp=nComps,intercept=T))
        zcoef=as.vector(coef(resS,ncomp=nComps,intercept=T))
        coefMat[nsim,]=coefs
        coefStd[nsim,]=zcoef

        # GET VIP
        vip=c()
        for (j in seq(length(inBands)))
                {
                vip=c(vip,VIPjh(resS,j,nComps))
                }
        vipMat[,nsim]=vip

        # GET FITS
        fitX=as.vector(unlist(resX$fitted.values[,1,nComps]))
        preX=as.vector(unlist(predict(resX,ncomp=nComps,newdata=tstData)))

        fitBias=mean(subData[,inVar]-fitX)
        valBias=mean(tstData[,inVar]-preX)

        fitR2=summary(lm(subData[,inVar]~fitX))$adj.r.squared
        valR2=summary(lm(tstData[,inVar]~preX))$adj.r.squared

        fitRMSE=sqrt(mean((subData[,inVar]-fitX)^2))
        valRMSE=sqrt(mean((tstData[,inVar]-preX)^2))

        outVec=c(fitR2,fitRMSE,fitBias,valR2,valRMSE,valBias)
```

```
        statMat[nsim,]=outVec

        }

######################################################## GET PREDICTIONS

statMat=as.data.frame(statMat)
names(statMat)=c("fitR2","fitRMSE","fitBias","valR2","valRMSE","valBias")

specMat=allData[,inBands]
specMat=cbind(rep(1,nrow(specMat)),specMat)
specMat=as.matrix(specMat)
dim(specMat)

predMat=specMat%*%t(coefMat)
predMean=apply(predMat,FUN=mean,MAR=1)
predStdv=apply(predMat,FUN=sd,MAR=1)

modSum=summary(lm(allData[,inVar]~predMean))
modR2=modSum$adj.r.squared
modRMSE=sqrt(mean((allData[,inVar]-predMean)^2))
modBias=mean(allData[,inVar]-predMean)

# WRITE OUT STATISTICS
write.csv(statMat,paste(saveDir,inVar,"_",y,"_fitModelStats.csv",sep=""),row.names=FALSE)
allData[,paste("predMean_",inVar,sep="")]=predMean
allData[,paste("predStdv_",inVar,sep="")]=predStdv
write.csv(allData,paste(saveDir,inVar,"_",y,"_fitModelData.csv",sep=""),row.names=FALSE)

png(paste(saveDir,inVar,"_",y,"Fit_fitModels.png",sep=""),width=5,height=5,units="in",res=200)
plot(allData[,inVar],predMean,pch=16,cex=1,xlab="observed",ylab="predicted",main=inVar,ylim=c(min(predMean-
predStdv),max(predMean+predStdv)))
abline(0,1,col="red",lwd=2,lty=2)
abline(lm(predMean~allData[,inVar]),col="black",lwd=2)
arrows(allData[,inVar],predMean,allData[,inVar],predMean+predStdv,angle=90,length=0.05)
arrows(allData[,inVar],predMean,allData[,inVar],predMean-predStdv,angle=90,length=0.05)
points(allData[,inVar],predMean,pch=16,cex=1)
dev.off()

######################################################## GET VIPs, COEFFICIENTS

# WRITE OUT VIPs
vipAggr=as.data.frame(t(apply(vipMat,MAR=1,FUN=quantile,probs=c(0.05,0.5,0.95))))
vipMean=apply(vipMat,MAR=1,FUN=mean)
vipStdv=apply(vipMat,MAR=1,FUN=sd)
vipAggr$mean=vipMean
vipAggr$stdv=vipStdv
vipAggr$BandName=inBands
write.table(vipAggr,paste(saveDir,inVar,"_",y,"_output_VIP.csv",sep=""),sep=",",col.names=T,row.names=F)

# WRITE OUT COEFFICIENTS RAW
coefAggr=as.data.frame(t(apply(coefMat,MAR=2,FUN=quantile,probs=c(0.05,0.5,0.95))))
dim(coefAggr)
coefMean=apply(coefMat,MAR=2,FUN=mean)
coefStdv=apply(coefMat,MAR=2,FUN=sd)
coefAggr$mean=coefMean
coefAggr$stdv=coefStdv
coefAggr$BandName=c("Intercept",inBands)
write.table(coefAggr,paste(saveDir,inVar,"_",y,"_output_COEF_RAW.csv",sep=""),sep=",",col.names=T,row.names=F)
write.table(coefMat,paste(saveDir,inVar,"_",y,"_output_COEFmat.csv",sep=""),sep=",",col.names=T,row.names=F)

# WRITE OUT COEFFICIENTS STD
coefAggrS=as.data.frame(t(apply(coefStd,MAR=2,FUN=quantile,probs=c(0.05,0.5,0.95))))
dim(coefAggrS)
coefMeanS=apply(coefStd,MAR=2,FUN=mean)
coefStdvS=apply(coefStd,MAR=2,FUN=sd)
coefAggrS$mean=coefMeanS
coefAggrS$stdv=coefStdvS
coefAggrS$BandName=c("Intercept",inBands)
write.table(coefAggrS,paste(saveDir,inVar,"_",y,"_output_COEF_STD.csv",sep=""),sep=",",col.names=T,row.names=F)

Stats_combined[y,] <- sapply(statMat,mean)
}

Stats_combined<-cbind(Dataset,Stats_combined)
write.csv(Stats_combined, paste(saveDir,inVar,"_Stats_combined.csv",sep=""))
```