**Supplemental 4:**

When building PLSR models, the selection of components to use can be somewhat subjective. The functions to assist in this as implemented in the *pls* package were used for the CV00 cross validation methods, but a more visual approach was used in selecting 15 components for the held out validation sets as presented for the elite breeding material experiments (Fig. S1). Although the algorithm suggests 10 components, 15 appeared more optimal in many cases. In general, 10-15 components were optimal.

C:\Users\holly.lane\Project\Figures\FINALS\WCPLS RMSEP CURVE COMP SELECT.tiffC:\Users\holly.lane\Project\Figures\FINALS\WCPLS RMSEP CURVE.tiffC:\Users\holly.lane\Project\Figures\FINALS\2011RMSEPCURVE.tiffC:\Users\holly.lane\Project\Figures\FINALS\USDARMSEPCURVE.tiff

***R CODE FOR BUILDING PLSR AND LM MODELS***

***Building PLSR Models 1, 3-5, 7, 8***

## To convert yield from BU/AC to TONNES/HA, multiply yield by (0.06271)

setwd("C:/Users/holly.lane/Project/Manuscript Files") ## set this for your system

library(reshape); library(reshape2);library(ggplot2); library(dplyr)

library(prospectr);library(RcppArmadillo);library(Rcpp); library(foreach) ; library(iterators)

library(resemble);require(prospectr);library(ggfortify);library(cluster); library(pls)

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

# First, we clean the data and transform it with an SG derivative

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

wholecorn<- read.csv("Supplemental 1.csv", header=TRUE, check.names = FALSE)

#check.names prevents R from placing an X in front of the wavelength column names

dim(wholecorn)

colnames(wholecorn[,1:83])

##forcing R to accept numeric column names makes the column names in valid

##extract spectra columns before fixing this

wholecornspc <- wholecorn[,83:3194]

wholespc <- data.matrix(wholecornspc)

##now fix column names

valid\_column\_names <- make.names(names=names(wholecorn), unique=TRUE, allow\_ = TRUE)

names(wholecorn) <- valid\_column\_names

#select informational columns of interest, then attach matrix of spectra as variable

wholecornfull<- select(wholecorn,c(5,6,7,8,9,21,22,23,25,32,43,77,78))

wholecornfull$year<-as.factor(wholecornfull$year)

wholecornfull$spc<-wholecornspc

###in example dataset, NIRsoil, spectra is variable spc as a matrix within "dataframe"

wholecornfull$raw<-wholespc

#sum(wholecornfull$yield.buac<10) #47 removed

wholecornya<-wholecornfull[!wholecornfull$yield.buac<10,]

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

whole.sg <- savitzkyGolay(wholecornya$spc, p = 3, w = 37, m = 1)

whole.sg2 <- savitzkyGolay(wholecornya$spc, p = 3, w = 37, m = 2) ## adjust w

wholecornya$sg1<-whole.sg

wholecornya$sg2<-whole.sg2

wc<-wholecornya

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

#create average spectra to compare to as baseline

avewc <- wc [c(10, 50, 100, 222, 500, 300, 1300, 2000),]

avewc1 <- wc[1,]

avewc1$sg1<-t(as.matrix(colMeans(avewc$sg1)))

aveSpec <- avewc1$sg1

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

aveSpec <-matrix(aveSpec, nrow= nrow(wc$sg1), ncol=ncol(aveSpec), byrow=TRUE)

wc$diff<- wc$sg1 - aveSpec

wc$sumdiff<-rowSums (abs(wc$diff))

hist(wc$sumdiff, breaks=200) #see the distribution of differences

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

sum(wc$sumdiff>.5)

sum(wc$sumdiff>.4)

wholecornCLEAN<-wc[!wc$sumdiff>.4,]# Final threshold selected to remove plots with poor scans

wc<-wholecornCLEAN

hist(wc$sumdiff, breaks=200)

rownames(wc) <- seq(length=nrow(wc))

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

#we removed two extra odd points, identified here

wc.pls <- plsr(yield.buac~sg1, data= wc, ncomp=15, validation="CV")

summary(wc.pls)

plot(wc.pls, ncomp = 15, asp = 1, line = TRUE)

plot(RMSEP(wc.pls), legendpos = "topright")

plot(wc.pls, plottype = "scores", comps = 1:10)

plot(x= wc.pls$scores[,5], y= wc.pls$scores[,9])

#identify(x= wc.pls$scores[,5], y= wc.pls$scores[,9], plot=TRUE)

wc2<- wc[-c(1778, 2032),]

wc<-wc2

rownames(wc) <- seq(length=nrow(wc))

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

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

hist(wc$yield.buac)

wc11 <- wc[(wc$year=="2011"),]

wc12 <-wc[(wc$year=="2012"),]

rownames(wc11) <- seq(length=nrow(wc11))

rownames(wc12) <- seq(length=nrow(wc12))

## below are histograms to compare the difference between years in yield distribution

hist(wc11$yield.buac,

main="Distribution of 2011 Yield",

xlab = expression(paste(" Yield ", "(bu ac"^"-1" ,")" )),

xlim=c(0,250),

ylim=c(0,175),

cex.axis=1.25,

col="#003A66", #poster color

#col = "darkred", #presentation color

las=1,

breaks=20)

range(wc11$yield.buac)

hist(wc12$yield.buac,

main="Distribution of 2012 Yield",

xlab = expression(paste(" Yield ", "(bu ac"^"-1" ,")" )),

cex.axis=1.25,

xlim=c(0,250),

ylim=c(0,175),

col="#003A66", #poster color

#col = "darkred", #presentation color

las=1,

breaks=20)

range(wc12$yield.buac)

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

###

## Importing, cleaning, and filtering 2011 experiments

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

corn2011<- read.csv("Supplemental 2.csv", header=TRUE, check.names = FALSE)

#check.names prevents R from placing an X in front of the wavelength column names

dim(corn2011)

##forcing R to accept numeric column names makes the column names in valid

##extract spectra columns before fixing this

colnames(corn2011[,1:80])

corn2011spc <- corn2011[,30:3141]

spc2011 <- data.matrix(corn2011spc)

##now fix column names

valid\_column\_names <- make.names(names=names(corn2011), unique=TRUE, allow\_ = TRUE)

names(corn2011) <- valid\_column\_names

#select informational columns of interest, then attach matrix of spectra as variable

corn2011full<- select(corn2011,c(1,2,3,4,9:12,20))

corn2011full$year<-as.factor(corn2011full$year)

corn2011full$spc<-corn2011spc

###in example dataset, NIRsoil, spectra is variable spc as a matrix within "dataframe"

corn2011full$raw<-spc2011

#sum(corn2011full$yield.buac<10) #47 removed

corn2011ya<-corn2011full[!corn2011full$yield.buac<10,]

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

whole.sg <- savitzkyGolay(corn2011ya$spc, p = 3, w = 37, m = 1)

whole.sg2 <- savitzkyGolay(corn2011ya$spc, p = 3, w = 37, m = 2) ## adjust w

corn2011ya$sg1<-whole.sg

corn2011ya$sg2<-whole.sg2

MORE2011<-corn2011ya

#MORE2011<-na.omit(MORE2011) #NAs are present in the dataset for different variables

MORE2011<-na.omit(MORE2011, cols= "yield.buac")

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

aveMORE2011 <- MORE2011

aveMORE20111 <- MORE2011[1,]

aveMORE20111$sg1<-t(as.matrix(colMeans(aveMORE2011$sg1)))

aveSpec <- aveMORE20111$sg1

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

aveSpec <-matrix(aveSpec, nrow= nrow(MORE2011$sg1), ncol=ncol(aveSpec), byrow=TRUE)

MORE2011$diff<- MORE2011$sg1 - aveSpec

MORE2011$sumdiff<-rowSums (abs(MORE2011$diff))

hist(MORE2011$sumdiff, breaks=200)

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

sum(MORE2011$sumdiff>.5)

which(MORE2011$sumdiff>.5)

sum(MORE2011$sumdiff>.35)

which(MORE2011$sumdiff>.4)

MORE2011CLEAN<-MORE2011[!MORE2011$sumdiff>.35,] ##Final threshold selected

MORE2011<-MORE2011CLEAN

#hist(MORE2011$sumdiff, breaks=200)

rownames(MORE2011) <- seq(length=nrow(MORE2011))

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

## again, two odd points identified and removed

pls.11<- plsr(yield.buac~sg1, data= MORE2011, ncomp=15, validation="CV")

summary(pls.11)

plot(pls.11, ncomp = 15, asp = 1, line = TRUE)

plot(RMSEP(pls.11), legendpos = "topright")

plot(pls.11, plottype = "scores", comps = 1:10)

plot(x= pls.11$scores[,7], y= pls.11$scores[,1])

#identify(x= pls.11$scores[,7], y= pls.11$scores[,1], plot=TRUE)

MORE20112<- MORE2011[-598,]

MORE2011<-MORE20112

rownames(MORE2011) <- seq(length=nrow(MORE2011))

MORE20112<- MORE2011[-157,]

MORE2011<-MORE20112

rownames(MORE2011) <- seq(length=nrow(MORE2011))

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

#

## Now we can start building testing and training sets for PLSR models

#

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

# Need to select a representative validation set to test our PLS on, using DUPLEX method

wc.duplex<-duplex(X=wc$sg1,k=120,pc=10) #takes awhile to run

plot(wc.duplex$pc)

points(wc.duplex$pc[wc.duplex$model, 1],wc.duplex$pc[wc.duplex$model, 2], col = "red", pch = 19)

points(wc.duplex$pc[wc.duplex$test, 1], wc.duplex$pc[wc.duplex$test, 2], col = "blue", pch = 19)

legend("topright", legend = c("calibration", "validation"), pch = c(19, 19),

col = c("red", "blue"))

##result is list with row index for calibration(model)/and validation(test) samples

testrows<-wc.duplex$test

testrows<-sort.int(testrows)

wcTest <- wc[(testrows),]

wcTrain <-wc[-testrows,]

levels(droplevels(wc$pedigree))

# 344 represented -- NOTE: Some may be missing from yield adjustments

levels(droplevels(wcTest$pedigree))

# 91 represented, need to eliminate all from training set

levels(droplevels(wcTrain$pedigree))

#343 represented

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

wcTest.pedigree<-wc[wc$pedigree %in% wcTest$pedigree, ]

levels(droplevels(wcTest.pedigree$pedigree))

dim(wcTest)

dim(wcTest.pedigree)

wcTest.pedigree[1:10,1:5]

wcTrain.pedigree<-wc[!wc$pedigree %in% wcTest$pedigree, ]

dim(wcTrain)

dim(wcTrain.pedigree)

wcTrain.pedigree[1:10,1:5]

levels(droplevels(wcTrain.pedigree$pedigree))

#253 pedigrees in training set

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

wcTest<-wcTest.pedigree

wcTrain<-wcTrain.pedigree

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

whole.pls.ya <- plsr(yield.buac~sg1, data= wcTrain, ncomp=50, validation="CV")

#now we have built PLSR 1

summary(whole.pls.ya)

plot(whole.pls.ya, ncomp = 25, asp = 1, line = TRUE)

####

#modelresults<-sort.default(whole.pls.ya$validation$pred[,,15])

##

RMSEP(whole.pls.ya, newdata=wcTest)

plot(RMSEP(whole.pls.ya), legendpos = "topright")

ncomp.onesigma <- selectNcomp(whole.pls.ya, method = "onesigma", plot = TRUE)

ncomp.permut <- selectNcomp(whole.pls.ya, method = "randomization", plot = TRUE)

plot(RMSEP(whole.pls.ya, newdata=wcTest), legendpos = "topright")

plot(R2(whole.pls.ya, newdata=wcTest))

ncomp.onesigma <- selectNcomp(whole.pls.ya, method = "onesigma", plot = TRUE,

main = "Component Selection")

##### PLOT

rmsep<-RMSEP(whole.pls.ya, newdata=wcTest, ncomp=10)

R2(whole.pls.ya, newdata=wcTest, ncomp=10)

regline<- lm((predict(whole.pls.ya, ncomp=10, newdata=wcTest))~ wcTest$yield.buac)

predplot(whole.pls.ya, ncomp = 10, newdata = wcTest, line = TRUE,

cex.axis = 1.5, cex.lab = 1.5,cex.main=2,

pch = 16, main = "Global Prediction Ability (NIRS)",

xlab = expression(paste("True Yield ", "(bu ac"^"-1" ,")" )),

ylab = expression(paste("Predicted Yield ", "(bu ac"^"-1" ,")" )),

las=1)+

abline(a=0,b=1,col="firebrick3",lwd=2)

abline(regline, lwd=2)

legend(x=140, y=60, legend= paste("R2 =", format(summary(regline)$r.squared, digits=2)), bty="n")

legend(x=140, y=50, legend= noquote(paste("RMSEP=", (round(rmsep$val[1,1,2], digits=2)), "bu/ac")), bty="n")

legend(x=175, y=20, legend = c("regression", "1:1"),pch= "\_",

col = c("black", "firebrick3"), pt.cex = 2,bty="n" )

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

## Below are within year and between year models

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

## 2011 ###

wc11.duplex<-duplex(X=wc11$sg1,k=70,pc=10)

plot(wc11.duplex$pc)

points(wc11.duplex$pc[wc11.duplex$model, 1],wc11.duplex$pc[wc11.duplex$model, 2], col = "red", pch = 19)

points(wc11.duplex$pc[wc11.duplex$test, 1], wc11.duplex$pc[wc11.duplex$test, 2], col = "blue", pch = 19)

legend("topright", legend = c("calibration", "validation"), pch = c(19, 19),

col = c("red", "blue"))

testrows11<-wc11.duplex$test

testrows11<-sort.int(testrows11)

wc11Test <- wc11[(testrows11),]

wc11Train <-wc11[-testrows11,]

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

wc11Test.pedigree<-wc11[wc11$pedigree %in% wc11Test$pedigree, ]

levels(droplevels(wc11Test.pedigree$pedigree))

wc11Train.pedigree<-wc11[!wc11$pedigree %in% wc11Test$pedigree, ]

levels(droplevels(wc11Train.pedigree$pedigree))

#213 pedigrees in training set

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

wc11Test<-wc11Test.pedigree

wc11Train<-wc11Train.pedigree

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

wc11.pls <- plsr(yield.buac~sg1, data= wc11Train, ncomp=15, validation="CV")

#PLSR 3

summary(wc11.pls)

plot(wc11.pls, ncomp = 15, asp = 1, line = TRUE)

RMSEP(wc11.pls, newdata=wc11Test)

plot(RMSEP(wc11.pls), legendpos = "topright")

plot(RMSEP(wc11.pls, newdata=wc11Test), legendpos = "topright")

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

#plot

regline<- lm((predict(wc11.pls, ncomp=12, newdata=wc11Test))~ wc11Test$yield.buac)

rmsep<-RMSEP(wc11.pls, newdata=wc11Test, ncomp=12)

predplot(wc11.pls, ncomp = 12, newdata = wc11Test, line = TRUE,

pch = 16, main = "Prediction Ability on 2011 Data Alone",

xlab = expression(paste("True Yield ", "(bu ac"^"-1" ,")" )),

ylab = expression(paste("Predicted Yield ", "(bu ac"^"-1" ,")" )),

las=1, cex.axis=1.5)+

abline(a=0,b=1,col="firebrick3",lwd=2)

abline(regline, lwd=2)

legend(x=100, y=45, legend= paste("R2 =", format(summary(regline)$r.squared, digits=4)), bty="n")

legend(x=100, y=35, legend= noquote(paste("RMSEP=", (round(rmsep$val[1,1,2], digits=2)), "bu/ac")), bty="n")

legend(x=100, y=20, legend = c("regression", "m=1"),pch= "\_",

col = c("black", "firebrick3"), pt.cex = 2,bty="n" )

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

## 2012 ###

wc12.duplex<-duplex(X=wc12$sg1,k=100,pc=10)

plot(wc12.duplex$pc)

points(wc12.duplex$pc[wc12.duplex$model, 1],wc12.duplex$pc[wc12.duplex$model, 2], col = "red", pch = 19)

points(wc12.duplex$pc[wc12.duplex$test, 1], wc12.duplex$pc[wc12.duplex$test, 2], col = "blue", pch = 19)

legend("topright", legend = c("calibration", "validation"), pch = c(19, 19),

col = c("red", "blue"))

testrows12<-wc12.duplex$test

testrows12<-sort.int(testrows12)

wc12Test <- wc12[(testrows12),]

wc12Train <-wc12[-testrows12,]

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

wc12Test.pedigree<-wc12[wc12$pedigree %in% wc12Test$pedigree, ]

levels(droplevels(wc12Test.pedigree$pedigree))

wc12Train.pedigree<-wc12[!wc12$pedigree %in% wc12Test$pedigree, ]

levels(droplevels(wc12Train.pedigree$pedigree))

#258 pedigrees in training set

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

wc12Test<-wc12Test.pedigree

wc12Train<-wc12Train.pedigree

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

wc12.pls <- plsr(yield.buac~sg1, data= wc12Train, ncomp=15, validation="CV")

# PLSR4

summary(wc12.pls)

plot(wc12.pls, ncomp = 15, asp = 1, line = TRUE)

RMSEP(wc12.pls, newdata=wc12Test)

plot(RMSEP(wc12.pls), legendpos = "topright")

plot(RMSEP(wc12.pls, newdata=wc12Test), legendpos = "topright")

######### PLOT

rmsep<-RMSEP(wc12.pls, newdata=wc12Test, ncomp=15)

regline<- lm((predict(wc12.pls, ncomp=15, newdata=wc12Test))~ wc12Test$yield.buac)

predplot(wc12.pls, ncomp = 15, newdata = wc12Test, line = TRUE,

pch = 16, main = "Prediction Ability on 2012 Data Alone",

xlab = expression(paste("True Yield ", "(bu ac"^"-1" ,")" )),

ylab = expression(paste("Predicted Yield ", "(bu ac"^"-1" ,")" )),

las=1)+

abline(a=0,b=1,col="firebrick3",lwd=2)

abline(regline, lwd=2)

legend(x=150, y=80, legend= paste("R2 =", format(summary(regline)$r.squared, digits=4)), bty="n")

legend(x=150, y=65, legend= noquote(paste("RMSEP=", (round(rmsep$val[1,1,2], digits=2)), "bu/ac")), bty="n")

legend(x=150, y=50, legend = c("regression", "m=1"),pch= "\_",

col = c("black", "firebrick3"), pt.cex = 2,bty="n" )

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

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

wc12TestSet.pedigree<-wc12[!wc12$pedigree %in% wc11$pedigree, ]

levels(droplevels(wc12TestSet.pedigree$pedigree))

#69

levels(droplevels(wc11$pedigree))

#275

wcTestSet<-wc12TestSet.pedigree

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

wc12.pls <- plsr(yield.buac~sg1, data= wc11, ncomp=25, validation="CV")

#PLSR 7

summary(wc12.pls)

plot(wc12.pls, ncomp = 15, asp = 1, line = TRUE)

RMSEP(wc12.pls, newdata=wcTestSet)

plot(RMSEP(wc12.pls), legendpos = "topright")

plot(RMSEP(wc12.pls, newdata=wcTestSet), legendpos = "topright")

######### PLOT

rmsep<-RMSEP(wc12.pls, newdata=wcTestSet, ncomp=10)

regline<- lm((predict(wc12.pls, ncomp=10, newdata=wcTestSet))~ wcTestSet$yield.buac)

predplot(wc12.pls, ncomp = 10, newdata = wcTestSet, line = TRUE,

pch = 16, main = "2011 Prediction Ability on 2012 Data - Different Pedigrees",

xlab = expression(paste("True Yield ", "(bu ac"^"-1" ,")" )),

ylab = expression(paste("Predicted Yield ", "(bu ac"^"-1" ,")" )),

las=1)+

abline(a=0,b=1,col="firebrick3",lwd=2)

abline(regline, lwd=2)

legend(x=150, y=80, legend= paste("R2 =", format(summary(regline)$r.squared, digits=4)), bty="n")

legend(x=150, y=65, legend= noquote(paste("RMSEP=", (round(rmsep$val[1,1,2], digits=2)), "bu/ac")), bty="n")

legend(x=150, y=50, legend = c("regression", "m=1"),pch= "\_",

col = c("black", "firebrick3"), pt.cex = 2,bty="n" )

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

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

## 2011 to predict 2012 ##

wc11.pls <- plsr(yield.buac~sg1, data= wc11, ncomp=25, validation="CV")

#PLSR5

summary(wc11.pls)

plot(wc11.pls, ncomp = 9, asp = 1, line = TRUE)

RMSEP(wc11.pls, newdata=wc12)

plot(RMSEP(wc11.pls), legendpos = "topright")

plot(RMSEP(wc11.pls, newdata=wc12), legendpos = "topright")

regline<- lm((predict(wc11.pls, ncomp=9, newdata=wc12))~ wc12$yield.buac)

predplot(wc11.pls, ncomp = 9, newdata = wc12, line = TRUE,

pch = 16, main = "2011 Prediction Ability on 2012 Data",

xlab = expression(paste("True Yield ", "(bu ac"^"-1" ,")" )),

ylab = expression(paste("Predicted Yield ", "(bu ac"^"-1" ,")" )),

las=1)+

abline(a=0,b=1,col="firebrick3",lwd=2)

abline(regline, lwd=2)

legend(x=150, y=75, legend= paste("R2 =", format(summary(regline)$r.squared, digits=4)), bty="n")

legend(x=150, y=55, legend = c("regression", "m=1"),pch= "\_",

col = c("black", "firebrick3"), pt.cex = 2,bty="n" )

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

## 2012 to predict 2011 ##

wc12.pls <- plsr(yield.buac~sg1, data= wc12, ncomp=15, validation="CV")

#PLSR 8

summary(wc12.pls)

plot(wc12.pls, ncomp = 15, asp = 1, line = TRUE)

RMSEP(wc12.pls, newdata=wc11)

plot(RMSEP(wc12.pls), legendpos = "topright")

plot(RMSEP(wc12.pls, newdata=wc11), legendpos = "topright")

regline<- lm((predict(wc12.pls, ncomp=11, newdata=wc11))~ wc11$yield.buac)

predplot(wc12.pls, ncomp = 11, newdata = wc11, line = TRUE,

pch = 16, main = "2012 Prediction Ability on 2011 Data",

xlab = expression(paste("True Yield ", "(bu ac"^"-1" ,")" )),

ylab = expression(paste("Predicted Yield ", "(bu ac"^"-1" ,")" )),

las=1)+

abline(a=0,b=1,col="firebrick3",lwd=2)

abline(regline, lwd=2)

legend(x=100, y=80, legend= paste("R2 =", format(summary(regline)$r.squared, digits=4)), bty="n")

legend(x=100, y=50, legend = c("regression", "m=1"),pch= "\_",

col = c("black", "firebrick3"), pt.cex = 2,bty="n" )

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

***Building PLSR Model 2 and LM 1-4***

## To convert yield from BU/AC to TONNES/HA, multiply yield by (0.06271)

setwd("C:/Users/holly.lane/Project/Manuscript Files") ## Set this for your system

library(reshape); library(reshape2);library(ggplot2); library(dplyr)

library(prospectr);library(RcppArmadillo);library(Rcpp); library(foreach) ; library(iterators)

library(resemble);require(prospectr);library(ggfortify);library(cluster); library(pls)

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

wholecorn<- read.csv("Supplemental 1.csv", header=TRUE, check.names = FALSE)

#check.names prevents R from placing an X in front of the wavelength column names

dim(wholecorn)

##forcing R to accept numeric column names makes the column names in valid

##extract spectra columns before fixing this

wholecornspc <- wholecorn[,83:3194]

wholespc <- data.matrix(wholecornspc)

##now fix column names

valid\_column\_names <- make.names(names=names(wholecorn), unique=TRUE, allow\_ = TRUE)

names(wholecorn) <- valid\_column\_names

#select informational columns of interest, then attach matrix of spectra as variable

wholecornfull<- select(wholecorn,c(5,7,8,9,21,22,23,25,32,43,48:61,77,78))

wholecornfull$year<-as.factor(wholecornfull$year)

wholecornfull$spc<-wholecornspc

###in example dataset, NIRsoil, spectra is variable spc as a matrix within "dataframe"

wholecornfull$raw<-wholespc

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

whole.sg <- savitzkyGolay(wholecornfull$spc, p = 3, w = 37, m = 1)

whole.sg2 <- savitzkyGolay(wholecornfull$spc, p = 3, w = 37, m = 2) ## adjust w

wholecornfull$sg1<-whole.sg

wholecornfull$sg2<-whole.sg2

wcFULL<-wholecornfull

WCcomp<- na.omit(wcFULL)

rownames(WCcomp) <- seq(length=nrow(WCcomp))

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

wc.pls <- plsr(yield.buac~sg1, data= WCcomp, ncomp=15, validation="CV")

summary(wc.pls)

plot(wc.pls, ncomp = 15, asp = 1, line = TRUE)

plot(RMSEP(wc.pls), legendpos = "topright")

plot(wc.pls, plottype = "scores", comps = 1:10)

plot(x= wc.pls$scores[,6], y= wc.pls$scores[,10])

#identify(x= wc.pls$scores[,6], y= wc.pls$scores[,10], plot=TRUE)

WCcomp2<- WCcomp[-c(1438, 1560),]

WCcomp<-WCcomp2

rownames(WCcomp) <- seq(length=nrow(WCcomp))

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

WCcomp.duplex<-duplex(X=WCcomp$sg1,k=100,pc=10) # takes awhile to run

plot(WCcomp.duplex$pc)

points(WCcomp.duplex$pc[WCcomp.duplex$model, 1],WCcomp.duplex$pc[WCcomp.duplex$model, 2], col = "red", pch = 19)

points(WCcomp.duplex$pc[WCcomp.duplex$test, 1], WCcomp.duplex$pc[WCcomp.duplex$test, 2], col = "blue", pch = 19)

legend("topright", legend = c("calibration", "validation"), pch = c(19, 19),

col = c("red", "blue"))

##result is list with row index for calibration(model)/and validation(test) samples

testrows<-WCcomp.duplex$test

testrows<-sort.int(testrows)

WCcompTest <- WCcomp[(testrows),]

WCcompTrain <-WCcomp[-testrows,]

levels(droplevels(WCcomp$pedigree))

# 345 represented -- NOTE: Some may be missing from yield adjustments

levels(droplevels(WCcompTest$pedigree))

# 82 represented, need to eliminate all from training set

levels(droplevels(WCcompTrain$pedigree))

# 345 represented

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

WCcompTest.pedigree<-WCcomp[WCcomp$pedigree %in% WCcompTest$pedigree, ]

levels(droplevels(WCcompTest.pedigree$pedigree))

dim(WCcompTest)

dim(WCcompTest.pedigree)

WCcompTest.pedigree[1:10,1:5]

WCcompTrain.pedigree<-WCcomp[!WCcomp$pedigree %in% WCcompTest$pedigree, ]

dim(WCcompTrain)

dim(WCcompTrain.pedigree)

WCcompTrain.pedigree[1:10,1:5]

levels(droplevels(WCcompTrain.pedigree$pedigree))

#263 pedigrees in training set

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

WCcompTest<-WCcompTest.pedigree

WCcompTrain<-WCcompTrain.pedigree

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

whole.pls.ya <- plsr(yield.buac~sg1, data= WCcompTrain, ncomp=25, validation="CV")

# PLSR 2 built

summary(whole.pls.ya)

plot(whole.pls.ya, ncomp = 15, asp = 1, line = TRUE)

####

#modelresults<-sort.default(whole.pls.ya$validation$pred[,,15])

##

#RMSEP(whole.pls.ya, newdata=WCcompTest)

plot(RMSEP(whole.pls.ya), legendpos = "topright")

ncomp.onesigma <- selectNcomp(whole.pls.ya, method = "onesigma", plot = TRUE)

ncomp.permut <- selectNcomp(whole.pls.ya, method = "randomization", plot = TRUE)

plot(RMSEP(whole.pls.ya, newdata=WCcompTest), legendpos = "topright")

##### PLOT

rmsep<-RMSEP(whole.pls.ya, newdata=WCcompTest, ncomp=13)

regline<- lm((predict(whole.pls.ya, ncomp=13, newdata=WCcompTest))~ WCcompTest$yield.buac)

predplot(whole.pls.ya, ncomp = 13, newdata = WCcompTest, line = TRUE,

pch = 16, main = "Global Prediction Ability (NIRS)",

xlab = expression(paste("True Yield ", "(bu ac"^"-1" ,")" )),

ylab = expression(paste("Predicted Yield ", "(bu ac"^"-1" ,")" )),

las=1)+

abline(a=0,b=1,col="firebrick3",lwd=2)

abline(regline, lwd=2)

legend(x=100, y=75, legend= paste("R2 =", format(summary(regline)$r.squared, digits=2)), bty="n")

legend(x=100, y=65, legend= noquote(paste("RMSEP=", (round(rmsep$val[1,1,2], digits=2)), "bu/ac")), bty="n")

legend(x=100, y=30, legend = c("regression", "1:1"),pch= "\_",

col = c("black", "firebrick3"), pt.cex = 2,bty="n" )

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

#protein

cor.test(WCcomp$yield.buac, WCcomp$crude.protein, method="pearson")

pp <- lm(yield.buac ~ crude.protein, data= WCcompTrain)

predictions<-predict(pp, newdata=WCcompTest)

regline <- lm(predictions ~ WCcompTest$yield.buac)

plot(WCcompTest$yield.buac, predictions)

abline(regline, lwd=2)

legend(x=180, y=75, legend= paste("R2 =", format(summary(regline)$r.squared, digits=2)), bty="n")

legend(x=180, y=65, legend= paste("R =", format(sqrt(summary(regline)$r.squared), digits=2)), bty="n")

## need to calculate RMSEP by hand....

WCcompTest$diff<-(predictions) - (WCcompTest$yield.buac)

WCcompTest$diff.sq<-(WCcompTest$diff)\*(WCcompTest$diff)

diff.sum<-sum(WCcompTest$diff.sq)

MSEP<-(diff.sum)/(nrow(WCcompTest))

RMSEP <- sqrt(MSEP)

RMSEP #in bu/ac

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

#starch.2

cor.test(WCcomp$yield.buac, WCcomp$starch.2, method="pearson")

sp <- lm(yield.buac ~ starch.2, data= WCcompTrain)

predictions<-predict(sp, newdata=WCcompTest)

regline <- lm(predictions ~ WCcompTest$yield.buac)

plot(WCcompTest$yield.buac, predictions)

abline(regline, lwd=2)

legend(x=180, y=75, legend= paste("R2 =", format(summary(regline)$r.squared, digits=2)), bty="n")

legend(x=180, y=65, legend= paste("R =", format(sqrt(summary(regline)$r.squared), digits=2)), bty="n")

WCcompTest$diff<-(predictions) - (WCcompTest$yield.buac)

WCcompTest$diff.sq<-(WCcompTest$diff)\*(WCcompTest$diff)

diff.sum<-sum(WCcompTest$diff.sq)

MSEP<-(diff.sum)/(nrow(WCcompTest))

RMSEP <- sqrt(MSEP)

RMSEP #in bu/ac

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

#fat.3

cor.test(WCcomp$yield.buac, WCcomp$fat.3, method="pearson")

fp <- lm(yield.buac ~ fat.3, data= WCcompTrain)

predictions<-predict(fp, newdata=WCcompTest)

regline <- lm(predictions ~ WCcompTest$yield.buac)

plot(WCcompTest$yield.buac, predictions)

abline(regline, lwd=2)

legend(x=180, y=90, legend= paste("R2 =", format(summary(regline)$r.squared, digits=2)), bty="n")

legend(x=180, y=92, legend= paste("R =", format(sqrt(summary(regline)$r.squared), digits=2)), bty="n")

WCcompTest$diff<-(predictions) - (WCcompTest$yield.buac)

WCcompTest$diff.sq<-(WCcompTest$diff)\*(WCcompTest$diff)

diff.sum<-sum(WCcompTest$diff.sq)

MSEP<-(diff.sum)/(nrow(WCcompTest))

RMSEP <- sqrt(MSEP)

RMSEP #in bu/ac

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

#all

ap <- lm(yield.buac ~ fat.3+starch.2+crude.protein, data= WCcompTrain)

predictions<-predict(ap, newdata=WCcompTest)

regline <- lm(predictions ~ WCcompTest$yield.buac)

plot(WCcompTest$yield.buac, predictions)

abline(regline, lwd=2)

legend(x=180, y=90, legend= paste("R2 =", format(summary(regline)$r.squared, digits=2)), bty="n")

legend(x=180, y=85, legend= paste("R =", format(sqrt(summary(regline)$r.squared), digits=2)), bty="n")

WCcompTest$diff<-(predictions) - (WCcompTest$yield.buac)

WCcompTest$diff.sq<-(WCcompTest$diff)\*(WCcompTest$diff)

diff.sum<-sum(WCcompTest$diff.sq)

MSEP<-(diff.sum)/(nrow(WCcompTest))

RMSEP <- sqrt(MSEP)

RMSEP #in bu/ac

***Running PLSR under CV00 Scheme***

setwd("C:/Users/holly.lane/Project/Manuscript Files")

library(reshape); library(reshape2);library(ggplot2); library(dplyr)

library(prospectr);library(RcppArmadillo);library(Rcpp); library(foreach) ; library(iterators)

library(resemble);require(prospectr);library(ggfortify);library(cluster); library(pls)

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

## Make sure you have run the other script that cleans and creates the wc object first

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

wc11 <- wc[(wc$year=="2011"),]

wc12 <-wc[(wc$year=="2012"),]

rownames(wc11) <- seq(length=nrow(wc11))

rownames(wc12) <- seq(length=nrow(wc12))

WW<-wc[(wc$treatment =="IRRI"),]

WS<-wc[(wc$treatment =="NIRRI"),]

WW11<-wc[(wc$trial.1 =="CS11\_WW"),]

WW12<-wc[(wc$trial.1 =="CS12\_WW"),]

WS11<-wc[(wc$trial.1 =="CS11\_WS"),]

WS12<-wc[(wc$trial.1 =="CS12\_WS"),]

rownames(WW) <- seq(length=nrow(WW))

rownames(WS) <- seq(length=nrow(WS))

rownames(WW11) <- seq(length=nrow(WW11))

rownames(WS11) <- seq(length=nrow(WS11))

rownames(WW12) <- seq(length=nrow(WW12))

rownames(WS12) <- seq(length=nrow(WS12))

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

wcTest<-WS12 ##Update and change plot title for each treatment

wcTrain<-WW

wcTrain<-wc[!(wc$trial.1 =="CS12\_WS"),] #Update and change plot title

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

whole.pls.ya <- plsr(yield.buac~sg1, data= wcTrain, ncomp=50, validation="CV")

#summary(whole.pls.ya)

#plot(whole.pls.ya, ncomp = 25, asp = 1, line = TRUE)

####

#modelresults<-sort.default(whole.pls.ya$validation$pred[,,15])

##

#RMSEP(whole.pls.ya, newdata=wcTest)

plot(RMSEP(whole.pls.ya), legendpos = "topright")

ncomp.onesigma <- selectNcomp(whole.pls.ya, method = "onesigma", plot = TRUE)

ncomp.permut <- selectNcomp(whole.pls.ya, method = "randomization", plot = TRUE)

plot(RMSEP(whole.pls.ya, newdata=wcTest), legendpos = "topright")

##### PLOT

rmsep<-RMSEP(whole.pls.ya, newdata=wcTest, ncomp=12)

R2(whole.pls.ya, newdata=wcTest, ncomp=12)

regline<- lm((predict(whole.pls.ya, ncomp=12, newdata=wcTest))~ wcTest$yield.buac)

predplot(whole.pls.ya, ncomp = 12, newdata = wcTest, line = TRUE,

cex.axis = 1.5, cex.lab = 1.5,cex.main=2, #for poster

pch = 16, main = "Fix",

xlab = expression(paste("True Yield ", "(bu ac"^"-1" ,")" )),

ylab = expression(paste("Predicted Yield ", "(bu ac"^"-1" ,")" )),

las=1)+

abline(a=0,b=1,col="firebrick3",lwd=2)

abline(regline, lwd=2)

legend(x=90, y=80, legend= paste("R2 =", format(summary(regline)$r.squared, digits=2)), bty="n")

legend(x=90, y=70, legend= noquote(paste("RMSEP=", (round(rmsep$val[1,1,2], digits=2)), "bu/ac")), bty="n")

legend(x=95, y=50, legend = c("regression", "1:1"),pch= "\_",

col = c("black", "firebrick3"), pt.cex = 2,bty="n" )

summary(regline)$r.squared

***R CODE FOR MODELS M1 AND M2***

***R code for the Near Infrared Spectroscopy (NIR)-BLUP model with interaction-M1***

### is the relationship matrices between lines, after taking for each line the average across ###environments of the wavenumber reflectance values.

rm(list=ls(all=TRUE))

library(reshape2)

dat\_F = read.csv(file='dat\_F.csv',h=T)

dat\_F = droplevels(dat\_F)

XF = read.csv(file='dat\_NIRs.csv',h=T)

XF$Id = as.character(XF$Id)

XF = data.frame(Env = as.character(XF$Site),XF)

XF = data.frame(Env\_Id = paste(XF$Env,XF$Id,sep='\_'),XF)

XF$Env = as.character(XF$Env)

XF$Env\_Id = as.character(XF$Env\_Id)

dat\_F$Id = as.character(dat\_F$Id)

dat\_F$Env = as.character(dat\_F$Site)

dat\_F = dat\_F[dat\_F$Env!='USDA',]

dat\_F = droplevels(dat\_F)

dat\_F1 = dat\_F[order(dat\_F$Env,dat\_F$Id),]

dat\_F$Env\_Id = paste(dat\_F$Env,dat\_F$Id,sep='\_')

head(dat\_F[1,1:8])

XF = XF[match(dat\_F$Env\_Id,XF$Env\_Id),]

ZL = model.matrix(~0+Id,data=dat\_F)

XE = model.matrix(~0+Env,data=dat\_F)

ZEL = model.matrix(~0+Id:Env,data=dat\_F)

library(BGLR)

library(epiR)

XF\_c = as.matrix(XF[,-(1:6)])

#-------------------------------------------------

#Genomic created by the Average of WLs in each Id

#-------------------------------------------------

Ids = unique(dat\_F$Id)

XF\_c\_a\_Id = data.frame()

for(j in 1:length(Ids))

{

XF\_j = matrix(XF\_c[dat\_F$Id==Ids[j],],nc=dim(XF\_c)[2])

if(dim(XF\_j)[1]==1)

{

XF\_c\_a\_Id = rbind(XF\_c\_a\_Id,

data.frame(Id=Ids[j],

matrix(XF\_j,nr=1)))

}

else

{

XF\_c\_a\_Id = rbind(XF\_c\_a\_Id,

data.frame(Id=Ids[j],

matrix(colMeans(XF\_j),nr=1)))

}

}

dim(XF\_c\_a\_Id)

head(XF\_c\_a\_Id)[,1:5]

XF\_c\_a = as.matrix(XF\_c\_a\_Id[,-1])

head(XF\_c\_a)[,1:5]

XF\_c\_a = scale(XF\_c\_a)

G = 1/dim(XF\_c)[2]\*tcrossprod(XF\_c\_a)

colnames(G) = paste('Id',XF\_c\_a\_Id$Id,sep='')

head(colnames(G))

Pos\_L = match(colnames(ZL),colnames(G))

G = G[Pos\_L,Pos\_L]

K = ZL%\*%G%\*%t(ZL)

#-------------------------------------------------

#W created by the Average of WLs in each Env-Id

#-------------------------------------------------

dat\_F$Env\_Id2 = paste('Id',dat\_F$Id,':','Env',dat\_F$Env,sep='')

Id\_Envs = unique(dat\_F$Env\_Id2)

XF\_c\_a\_Id\_Env = data.frame()

for(j in 1:length(Id\_Envs))

{

XF\_j = matrix(XF\_c[dat\_F$Env\_Id2==Id\_Envs[j],],nc=dim(XF\_c)[2])

if(dim(XF\_j)[1]==1)

{

XF\_c\_a\_Id\_Env = rbind(XF\_c\_a\_Id\_Env,

data.frame(Id\_Env=Id\_Envs[j],

matrix(XF\_j,nr=1)))

}

else

{

XF\_c\_a\_Id\_Env = rbind(XF\_c\_a\_Id\_Env,

data.frame(Id\_Env=Id\_Envs[j],

matrix(colMeans(XF\_j),nr=1)))

}

}

dim(XF\_c\_a\_Id\_Env)

head(XF\_c\_a\_Id\_Env)[,1:5]

XF\_c\_a\_EL = (XF\_c\_a\_Id\_Env[,-1])

XF\_c\_a\_EL = scale(XF\_c\_a\_EL)

GEL = 1/dim(XF\_c)[2]\*tcrossprod(as.matrix(XF\_c\_a\_EL))

colnames(GEL) = XF\_c\_a\_Id\_Env$Id\_Env

Pos\_EL = match(colnames(ZEL),colnames(GEL))

GEL = GEL[Pos\_EL,Pos\_EL]

KEL = ZEL%\*%GEL%\*%t(ZEL)

ETA = list(list(model='FIXED',X=XE[,-1]),list(model='RKHS',K=K),

list(model='RKHS',K=KEL))

Traits = 'y'

source('PT\_v4.R')

set.seed(1)

PT = PT\_f(dat\_F=data.frame(Env = dat\_F$Env,

Line = dat\_F$Id),K=5)

Tabp = data.frame()

for(t in 1:length(Traits))

{

yt = dat\_F[,Traits[t]]

for(p in 1:5)

{

yt\_NA = yt

Pos\_p = PT$g\_Pos\_ls[[paste('g',p,sep='')]]

yt\_NA[Pos\_p] = NA

A\_te = BGLR(y=yt\_NA,ETA = ETA,nIter = 2e4,burnIn = 1e4,verbose = FALSE)

yt\_p = A\_te$yHat

Tabp = rbind(Tabp,

data.frame(Trait = Traits[t],PT=p,Env=dat\_F$Env[Pos\_p],

Env = dat\_F$Env[Pos\_p],Year = dat\_F$Year[Pos\_p],

y = yt[Pos\_p],yp = yt\_p[Pos\_p]))

cat('PT',p,'\n')

}

}

library(dplyr)

Tab\_Env\_PT = Tabp%>%group\_by(Trait,Env,PT)%>%select(y,yp)%>%summarise(

Cor = cor(y,yp),

MSEP = mean((y-yp)\*\*2))

Tab\_Env = Tab\_Env\_PT%>%group\_by(Trait,Env)%>%

select(Cor,MSEP,MAAPE,CCC,Intercept,Slope)%>%

summarise(Cor\_Mean = mean(Cor),Cor\_SD=sd(Cor),

MSEP\_Mean = mean(MSEP), MSEP\_SD = sd(MSEP))

Tab\_Env = data.frame(Tab\_Env)

#Predicting one entire Env

Envs = unique(dat\_F$Env)

Tab2 = data.frame()

for(t in 1:length(Traits))

{

yt = dat\_F[,Traits[t]]

Envs = unique(dat\_F$Env)

for(p in 1:length(Envs))

{

yt\_NA = yt

Pos\_p = which(dat\_F$Env==Envs[p])

yt\_NA[Pos\_p] = NA

A\_te = BGLR(y=yt\_NA,ETA = ETA,nIter = 2e4,burnIn = 1e4,verbose = FALSE)

yt\_p = A\_te$yHat

Tab2 = rbind(Tab2,

data.frame(Trait = Traits[t],PT=p,Env=dat\_F$Env[Pos\_p],

Env = dat\_F$Env[Pos\_p],Year = dat\_F$Year[Pos\_p],

y = yt[Pos\_p],yp = yt\_p[Pos\_p]))

cat('PT',p,'\n')

}

}

Tab2\_Env = Tab2%>%group\_by(Trait,Env)%>%select(y,yp)%>%summarise(

Cor\_Mean = cor(y,yp), Cor\_SD=NA,

MSEP\_Mean = mean((y-yp)\*\*2),MSEP\_SD=NA)

Tab2\_Env = data.frame(Tab2\_Env)

***R code for the functional regression model with interaction – M2.***

rm(list=ls(all=TRUE))

#Building the desing matrix for functional regression with fourier basis

dat = read.csv("dat\_NIRs.csv",h=TRUE)

WLv = as.numeric(substring(names(dat)[-(1:4)],2))

plot(WLv,dat[1,][-(1:4)])

library(fda)

nb = 101

X\_WL = as.matrix(dat[,-(1:4)])

P = 1.02\*diff(range(c(WLv)))

bspF = create.fourier.basis(range(WLv)+c(0,0), nbasis = nb,

period = P)

n = dim(X\_WL)[1]

X\_WL\_F = matrix(NA, nrow = n, ncol = nb)

X\_DWL\_F = X\_WL\_F

for(i in 1:n)

{

xi = as.numeric(X\_WL[i,])

smf = smooth.basisPar(argvals = WLv,y =xi, fdobj = bspF,lambda=10,

Lfdobj = 1)

plot(WLv,xi)

lines(smf,col=2)

cv = smf$fd$coefs

I\_KL = inprod(bspF,bspF)

xai = t(I\_KL%\*%cv)

X\_WL\_F[i,] = xai

}

XF = data.frame(Year = dat$Year,Loc=dat$Loc,Site=dat$Site,Id=dat$Id,X\_WL\_F)

library(reshape2)

dat\_F = read.csv(file='dat\_F.csv',h=T)

dat\_F = droplevels(dat\_F)

XF$Id = as.character(XF$Id)

XF = data.frame(Env = as.character(XF$Site),XF)

XF = data.frame(Env\_Id = paste(XF$Env,XF$Id,sep='\_'),XF)

dat\_F$Id = as.character(dat\_F$Id)

dat\_F$Env = as.character(dat\_F$Site)

dat\_F = dat\_F[dat\_F$Env!='USDA',]

dat\_F = droplevels(dat\_F)

dat\_F1 = dat\_F[order(dat\_F$Env,dat\_F$Id),]

dat\_F$Env\_Id = paste(dat\_F$Env,dat\_F$Id,sep='\_')

XF = XF[match(dat\_F$Env\_Id,XF$Env\_Id),]

XF = droplevels(XF)

ZL = model.matrix(~0+Id,data=dat\_F)

XE = model.matrix(~0+Env,data=dat\_F)

ZEL = model.matrix(~0+Id:Env,data=dat\_F)

library(BGLR)

K = ZL%\*%t(ZL)

KEL = ZEL%\*%t(ZEL)

XF\_c = XF[,7:dim(XF)[2]]

head(colnames(XF\_c))

ETA = list(list(model='FIXED',X=XE[,-1]),list(model='RKHS',K=K),

list(model='RKHS',K=KEL),list(model='BRR',X=XF\_c))

Traits = 'y'

source('PT\_v4.R')

set.seed(1)

PT = PT\_f(dat\_F=data.frame(Env = dat\_F$Env,

Line = dat\_F$Id),K=5)

Tabp = data.frame()

for(t in 1:length(Traits))

{

yt = dat\_F[,Traits[t]]

for(p in 1:5)

{

yt\_NA = yt

Pos\_p = PT$g\_Pos\_ls[[paste('g',p,sep='')]]

yt\_NA[Pos\_p] = NA

A\_te = BGLR(y=yt\_NA,ETA = ETA,nIter = 2e4,burnIn = 1e4,verbose = FALSE)

yt\_p = A\_te$yHat

Tabp = rbind(Tabp,

data.frame(Trait = Traits[t],PT=p,Env=dat\_F$Env[Pos\_p],

Env = dat\_F$Env[Pos\_p],Year = dat\_F$Year[Pos\_p],

y = yt[Pos\_p],yp = yt\_p[Pos\_p]))

cat('PT',p,'\n')

}

}

library(dplyr)

Tab\_Env\_PT = Tabp%>%group\_by(Trait,Env,PT)%>%select(y,yp)%>%summarise(

Cor = cor(y,yp),

MSEP = mean((y-yp)\*\*2))

Tab\_Env = Tab\_Env\_PT%>%group\_by(Trait,Env)%>%

select(Cor,MSEP,MAAPE,CCC,Intercept,Slope)%>%

summarise(Cor\_Mean = mean(Cor),Cor\_SD=sd(Cor),

MSEP\_Mean = mean(MSEP), MSEP\_SD = sd(MSEP))

Tab\_Env = data.frame(Tab\_Env)

Envs = unique(dat\_F$Env)

#Predicting an entire Env

Tab2 = data.frame()

for(t in 1:length(Traits))

{

yt = dat\_F[,Traits[t]]

Envs = unique(dat\_F$Env)

for(p in 1:length(Envs))

{

yt\_NA = yt

Pos\_p = which(dat\_F$Env==Envs[p])

yt\_NA[Pos\_p] = NA

A\_te = BGLR(y=yt\_NA,ETA = ETA,nIter = 2e4,burnIn = 1e4,verbose = FALSE)

yt\_p = A\_te$yHat

Tab2 = rbind(Tab2,

data.frame(Trait = Traits[t],PT=p,Env=dat\_F$Env[Pos\_p],

Env = dat\_F$Env[Pos\_p],Year = dat\_F$Year[Pos\_p],

y = yt[Pos\_p],yp = yt\_p[Pos\_p]))

cat('PT',p,'\n')

}

}

Tab2\_Env = Tab2%>%group\_by(Trait,Env)%>%select(y,yp)%>%summarise(

Cor\_Mean = cor(y,yp), Cor\_SD=NA,

MSEP\_Mean = mean((y-yp)\*\*2),MSEP\_SD=NA)

Tab2\_Env = data.frame(Tab2\_Env)