An expanded PLSR example using leaf-level specrta and leaf mass per area (LMA) data from several CONUS NEON sites

Shawn P. Serbin

Overview

his is an R Markdown Notebook to illustrate how to conduct a basic model fit. This example shows you how to retrieve a dataset from the EcoSIS spectral database, choose the "optimal" number of plsr components, and fit a plsr model for leaf-mass area

When you click the **Knit** button in Rstudio a document will be generated that includes both content as well as the output of any embedded R code chunks within the document.

Getting Started

Installation

```
list.of.packages <- c("devtools","readr","RCurl","httr","pls","dplyr","reshape2",
                      "ggplot2","gridExtra")  # packages needed for script
# check for dependencies and install if needed
new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]
if(length(new.packages)) install.packages(new.packages)
```

Load libraries

```
#library(httr) #!! may not actually need this package
library(pls)
library(readr)
library(dplyr)
library(reshape2)
library(ggplot2)
library(gridExtra)
```

Setup other functions and options

```
# Source helper functions from GitHub
devtools::source_url("https://raw.githubusercontent.com/TESTgroup-BNL/How_to_PLSR/master/R_Scripts/funct

# not in
`%notin%` <- Negate(`%in%`)

# Script options
```

```
pls.options(plsralg = "oscorespls")
pls.options("plsralg")
```

$plsralg [1] “oscorespls”

```
pls.options()$parallel
```

NULL

```
# NULL

# What is the target variable?
inVar <- "LMA_gDW_m2"
```

Set working directory (scratch space)

```
outdir <- tempdir()
setwd(outdir) # set working directory
print(paste0("Output directory: ",getwd()))  # check wd
```

[1] “Output directory: /private/var/folders/xp/h3k9vf3n2jx181ts786_yjrn9c2gjq/T/RtmpP87nwE”

Grab data from EcoSIS

```
print("**** Downloading Ecosis data ****")
```

URL: https://ecosis.org/package/fresh-leaf-spectra-to-estimate-lma-over-neon-domains-in-eastern-united-states [1] “**** Downloading Ecosis data ****”

```
ecosis_id <- "5617da17-c925-49fb-b395-45a51291bd2d"  # NEON dataset
ecosis_file <- sprintf(
  "https://ecosis.org/api/package/%s/export?metadata=true",
  ecosis_id
)
message("Downloading data...")
dat_raw <- read_csv(ecosis_file)
message("Download complete!")
head(dat_raw)
```

A tibble: 6 x 2,162

Affiliation `Common Name` Domain Functional_type LMA `Latin Genus`
1 University~ black walnut D02 broadleaf 72.9 Juglans
2 University~ black walnut D02 broadleaf 72.9 Juglans
3 University~ black walnut D02 broadleaf 60.8 Juglans
4 University~ black walnut D02 broadleaf 60.8 Juglans
5 University~ black walnut D02 broadleaf 85.9 Juglans
6 University~ black walnut D02 broadleaf 85.9 Juglans
... with 2,156 more variables: `Latin Species` , PI , # Project , Sample_ID , `USDA Symbol` , 350 , # 351 , 352 , 353 , 354 , 355 , # 356 , 357 , 358 , 359 , 360 , # 361 , 362 , 363 , 364 , 365 , # 366 , 367 , 368 , 369 , 370 , # 371 , 372 , 373 , 374 , 375 , # 376 , 377 , 378 , 379 , 380 , # 381 , 382 , 383 , 384 ,

385 , # 386 , 387 , 388 , 389 , 390 , # 391 , 392 , 393 , 394 , 395 , # 396 , 397 , 398 , 399 , 400 , # 401 , 402 , 403 , 404 , 405 , # 406 , 407 , 408 , 409 , 410 , # 411 , 412 , 413 , 414 , 415 , # 416 , 417 , 418 , 419 , 420 , # 421 , 422 , 423 , 424 , 425 , # 426 , 427 , 428 , 429 , 430 , # 431 , 432 , 433 , 434 , 435 , # 436 , 437 , 438 , 439 , 440 , # 441 , 442 , 443 , 444 , ...

```
names(dat_raw)[1:40]
```

[1] "Affiliation" "Common Name" "Domain" "Functional_type" [5] "LMA" "Latin Genus" "Latin Species" "PI"
[9] "Project" "Sample_ID" "USDA Symbol" "350"
[13] "351" "352" "353" "354"
[17] "355" "356" "357" "358"
[21] "359" "360" "361" "362"
[25] "363" "364" "365" "366"
[29] "367" "368" "369" "370"
[33] "371" "372" "373" "374"
[37] "375" "376" "377" "378"

Create full plsr dataset

```
Start.wave <- 500
End.wave <- 2400
wv <- seq(Start.wave,End.wave,1)
spectra <- data.frame(dat_raw[,names(dat_raw) %in% wv])
names(spectra) <- c(paste0("Wave_",wv))
head(spectra)[1:6,1:10]
```

Wave_500 Wave_501 Wave_502 Wave_503 Wave_504 Wave_505 Wave_506 Wave_507 1 0.044226 0.044605 0.044927 0.045473 0.046241 0.046878 0.047826 0.049090 2 0.046855 0.047601 0.047944 0.048478 0.049381 0.050235 0.051161 0.052191 3 0.043758 0.044171 0.044869 0.045465 0.045984 0.046933 0.047993 0.049090 4 0.041154 0.041603 0.042088 0.042408 0.042639 0.043260 0.044140 0.045058 5 0.037296 0.037944 0.038209 0.038677 0.039388 0.039948 0.040630 0.041501 6 0.043878 0.044257 0.044723 0.045295 0.045949 0.046575 0.047378 0.048357 Wave_508 Wave_509 1 0.050268 0.051525 2 0.053322 0.054357 3 0.050168 0.051441 4 0.045700 0.046476 5 0.042613 0.043731 6 0.049392 0.050387

```
sample_info <- dat_raw[,names(dat_raw) %notin% seq(350,2500,1)]
head(sample_info)
```

A tibble: 6 x 11

Affiliation `Common Name` Domain Functional_type LMA `Latin Genus`
1 University~ black walnut D02 broadleaf 72.9 Juglans
2 University~ black walnut D02 broadleaf 72.9 Juglans
3 University~ black walnut D02 broadleaf 60.8 Juglans
4 University~ black walnut D02 broadleaf 60.8 Juglans
5 University~ black walnut D02 broadleaf 85.9 Juglans
6 University~ black walnut D02 broadleaf 85.9 Juglans
... with 5 more variables: `Latin Species` , PI , Project , # Sample_ID , `USDA Symbol`

```
sample_info2 <- sample_info %>%
  select(Domain,Functional_type,Sample_ID,USDA_Species_Code=`USDA Symbol`,LMA_gDW_m2=LMA)
head(sample_info2)
```

A tibble: 6 x 5

Domain Functional_type Sample_ID USDA_Species_Code LMA_gDW_m2 1 D02 broadleaf P0001 JUNI 72.9 2 D02 broadleaf L0001 JUNI 72.9 3 D02 broadleaf P0002 JUNI 60.8 4 D02 broadleaf L0002 JUNI 60.8 5 D02 broadleaf P0003 JUNI 85.9 6 D02 broadleaf L0003 JUNI 85.9

```
plsr_data <- data.frame(sample_info2,spectra)
head(plsr_data)[,1:10]
```

Domain Functional_type Sample_ID USDA_Species_Code LMA_gDW_m2 Wave_500 1 D02 broadleaf P0001 JUNI 72.87 0.044226 2 D02 broadleaf L0001 JUNI 72.87 0.046855 3 D02 broadleaf P0002 JUNI 60.77 0.043758 4 D02 broadleaf L0002 JUNI 60.77 0.041154 5 D02 broadleaf P0003 JUNI 85.92 0.037296 6 D02 broadleaf L0003 JUNI 85.92 0.043878 Wave_501 Wave_502 Wave_503 Wave_504 1 0.044605 0.044927 0.045473 0.046241 2 0.047601 0.047944 0.048478 0.049381 3 0.044171 0.044869 0.045465 0.045984 4 0.041603 0.042088 0.042408 0.042639 5 0.037944 0.038209 0.038677 0.039388 6 0.044257 0.044723 0.045295 0.045949

```
rm(sample_info,sample_info2,spectra)
```

Create cal/val datasets

```
set.seed(2356812)

#unique(plsr_data$USDA_Species_Code)
#unique(plsr_data$Domain)
# !!! this is messy and could likely be streamlined !!!
# !!! also we may want to split data by both domain and functional type or species !!!
domains <- unique(plsr_data$Domain)
cal.plsr.data <- 0
val.plsr.data <- 0
prop <- 0.80
j <- 1
for (i in domains){
  print(paste("Domain: ",i,sep=""))
  temp.data <- plsr_data[which(plsr_data$Domain==i),]
  rows <- sample(1:nrow(temp.data),floor(prop*nrow(temp.data)))
  cal_data = droplevels(temp.data[rows,])
  val_data = droplevels(temp.data[-rows,])

  if(j==1){
    cal.plsr.data <- cal_data
    val.plsr.data <- val_data
  } else {
    cal.plsr.data <- rbind(cal.plsr.data,cal_data)
    val.plsr.data <- rbind(val.plsr.data,val_data)
  }

  j <- j+1
}
```

[1] “Domain: D02” [1] “Domain: D03” [1] “Domain: D05” [1] “Domain: D06” [1] “Domain: D07” [1] “Domain: D08” [1] “Domain: D09”

```
rm(temp.data)
```

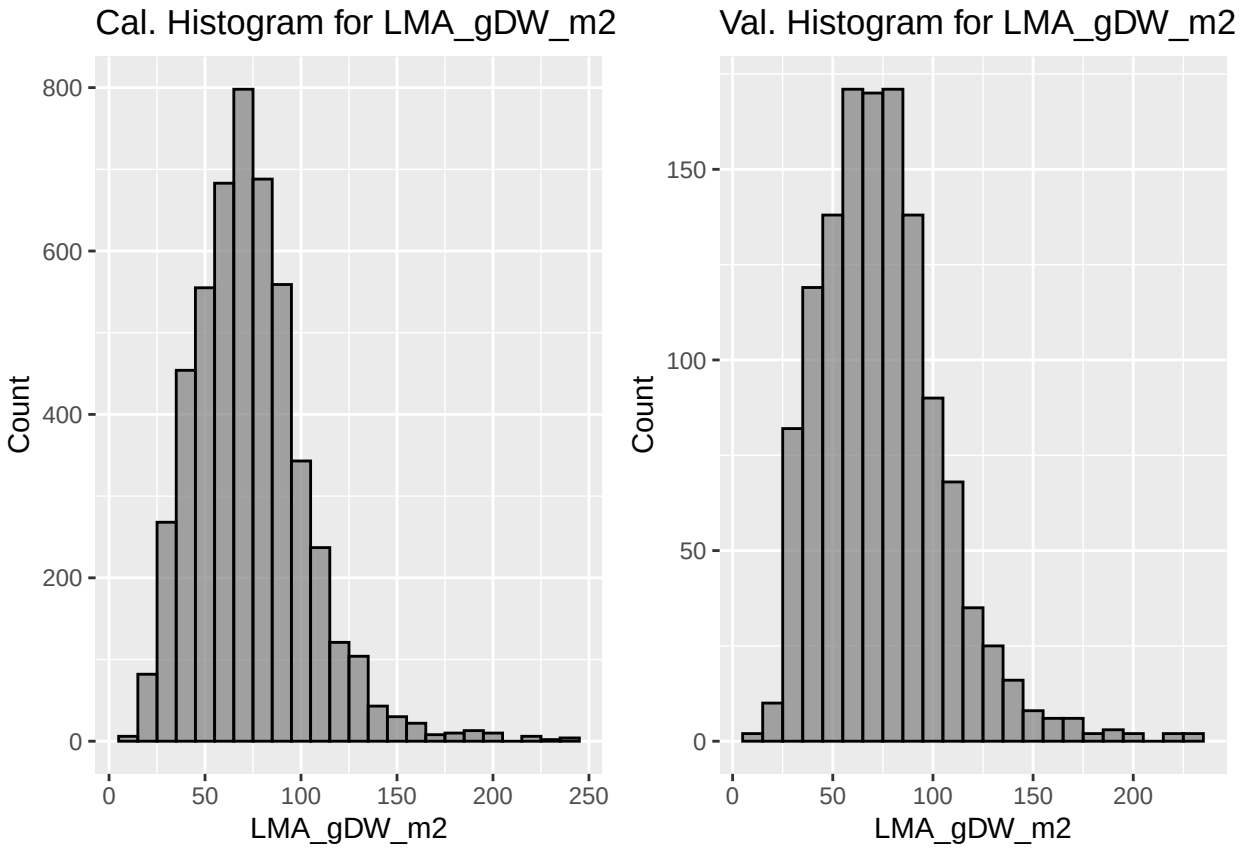
```
# Datasets:
print(paste("Cal observations: ",dim(cal.plsr.data)[1],sep=""))
```

[1] "Cal observations: 5046"

```
print(paste("Val observations: ",dim(val.plsr.data)[1],sep=""))
```

[1] "Val observations: 1266"

```
cal_hist_plot <- qplot(cal.plsr.data[,paste0(inVar)],geom="histogram",binwidth = 10,
      main = paste0("Cal. Histogram for ",inVar),
      xlab = paste0(inVar),ylab = "Count",fill=I("grey50"),col=I("black"),alpha=I(.7))
val_hist_plot <- qplot(val.plsr.data[,paste0(inVar)],geom="histogram",binwidth = 10,
      main = paste0("Val. Histogram for ",inVar),
      xlab = paste0(inVar),ylab = "Count",fill=I("grey50"),col=I("black"),alpha=I(.7))
grid.arrange(cal_hist_plot, val_hist_plot, ncol=2)
```




```
# !!  do we need to actually write any of this out to temp dir? !!
full_plsr_data <- rbind(cal.plsr.data,val.plsr.data)
write.csv(full_plsr_data,file=file.path(outdir,paste0(inVar,'_Full_PLSR_Dataset.csv')),row.names=FALSE)
write.csv(cal.plsr.data,file=file.path(outdir,paste0(inVar,'_Cal_PLSR_Dataset.csv')),row.names=FALSE)
write.csv(val.plsr.data,file=file.path(outdir,paste0(inVar,'_Val_PLSR_Dataset.csv')),row.names=FALSE)
rm(cal_data,val_data,i,j,prop,rows,domains)
```

Create calibration and validation PLSR datasets

```
spec_start <- which(names(cal.plsr.data)==paste0("Wave_",Start.wave))
cal.spec <- as.matrix(droplevels(cal.plsr.data[,spec_start:dim(cal.plsr.data)[2]]))
cal.plsr.data.2 <- data.frame(cal.plsr.data[,1:spec_start-1],Spectra=I(cal.spec))
cal.plsr.data <- cal.plsr.data.2
head(cal.plsr.data)[,1:5]
```

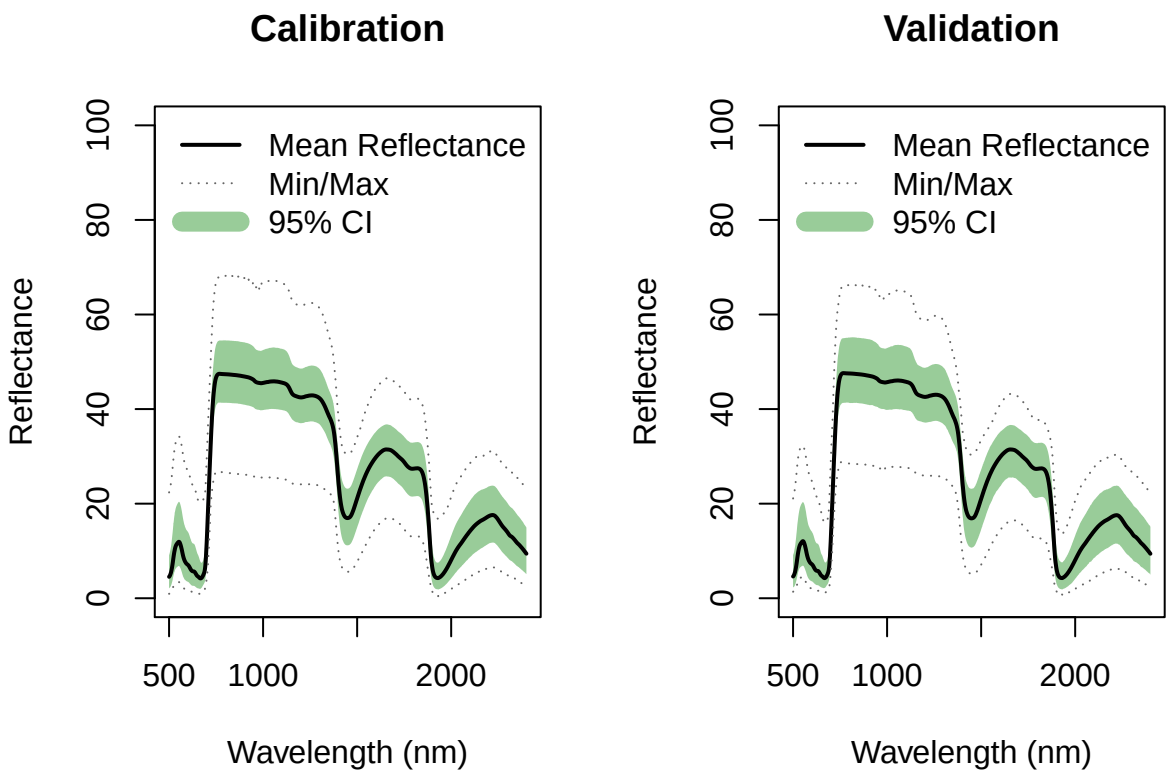
```
Domain Functional_type Sample_ID USDA_Species_Code LMA_gDW_m2
```

636 D02 broadleaf L0318 QUFA 111.19 460 D02 broadleaf L0230 FAGR 23.71 476 D02 broadleaf L0238 CATO 34.88 1 D02 broadleaf P0001 JUNI 72.87 364 D02 broadleaf L0182 QUAL 50.59 771 D02 broadleaf P0386 PLOC 50.91

```
rm(cal.plsr.data.2,cal.spec,spec_start)

spec_start <- which(names(val.plsr.data)==paste0("Wave_",Start.wave))
val.spec <- as.matrix(droplevels(val.plsr.data[,spec_start:dim(val.plsr.data)[2]]))
val.plsr.data.2 <- data.frame(val.plsr.data[,1:spec_start-1],Spectra=I(val.spec))
val.plsr.data <- val.plsr.data.2
head(val.plsr.data)[,1:5]
```

Domain Functional_type Sample_ID USDA_Species_Code LMA_gDW_m2 4 D02 broadleaf L0002 JUNI 60.77 5 D02 broadleaf P0003 JUNI 85.92 9 D02 broadleaf P0005 JUNI 48.76 17 D02 broadleaf P0009 QUVE 84.92 20 D02 broadleaf L0010 PRSE 78.82 21 D02 broadleaf P0011 PRSE 86.09

```
rm(val.plsr.data.2,val.spec,spec_start)

# plot cal and val spectra
par(mfrow=c(1,2)) # B, L, T, R
f.plot.spec(Z=cal.plsr.data$Spectra,wv=seq(Start.wave,End.wave,1),plot_label="Calibration")
f.plot.spec(Z=val.plsr.data$Spectra,wv=seq(Start.wave,End.wave,1),plot_label="Validation")
```



Use Jackknife permutation to determine optimal number of components

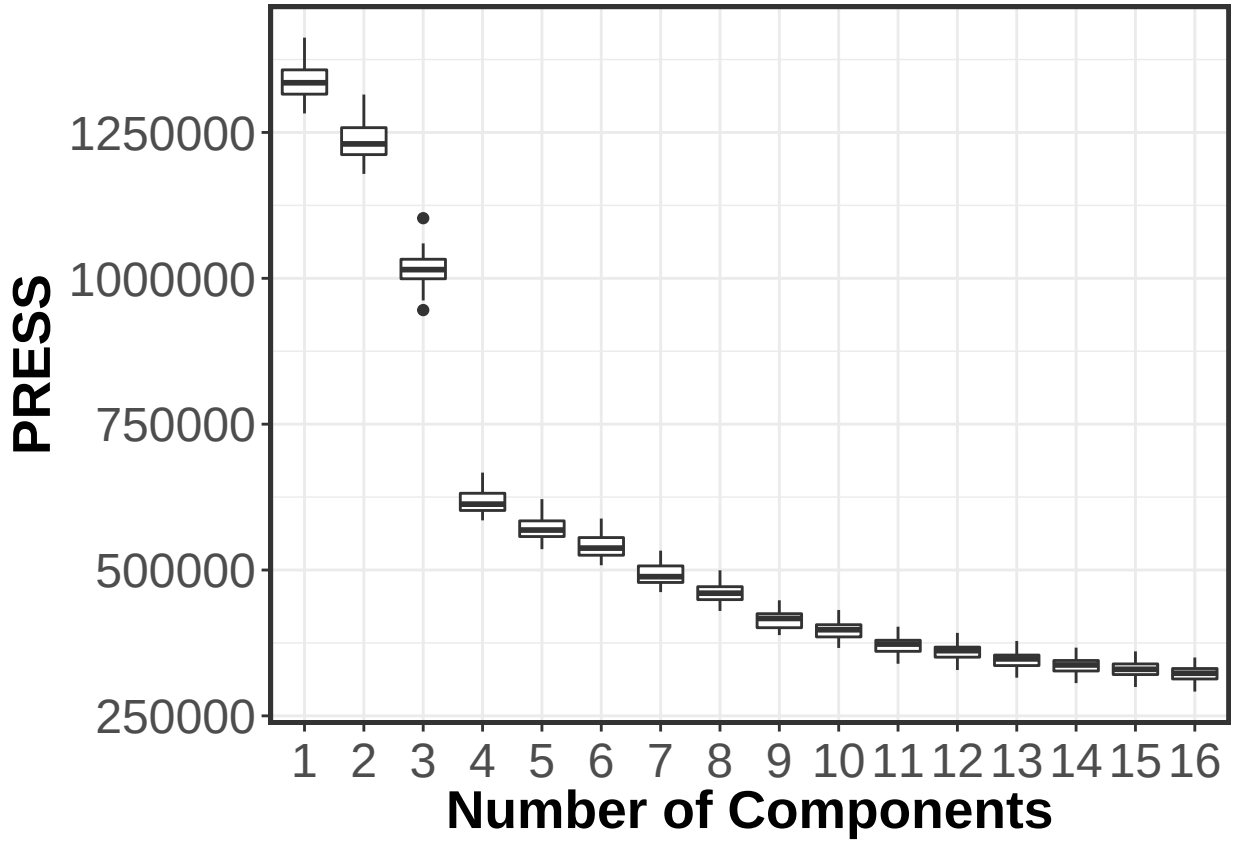
```
if(grepl("Windows", sessionInfo()$running)){
  pls.options(parallel =NULL)
} else {
  pls.options(parallel = parallel::detectCores()-1)
}
dims <- dim(plsr_data)
nComps <- 16
iterations <- 20
seg <- 15
prop <- 0.70
jk.out <- matrix(data=NA,nrow=iterations,ncol=nComps)
print("*** Running jacknife permutation test.  Please hang tight, this can take awhile ***")
```

[1] "*** Running jacknife permutation test. Please hang tight, this can take awhile ***"

```
start.time <- Sys.time()
for (i in 1:iterations) {
  rows <- sample(1:nrow(cal.plsr.data),floor(prop*nrow(cal.plsr.data)))
  sub.data <- cal.plsr.data[rows,]
  plsr.out <- plsr(as.formula(paste(inVar,"~","Spectra")), scale=FALSE, center=TRUE, ncomp=nComps,
                   validation="CV", segments = seg, segment.type="interleaved", trace=FALSE, data=sub.da
  resPRESS <- as.vector(plsr.out$validation$PRESS)
  jk.out[i,seq(plsr.out$validation$ncomp)]=resPRESS
}
end.time <- Sys.time()
end.time - start.time
```

Time difference of 6.226917 mins

```
# Jackknife PRESS plot
pressDF <- as.data.frame(jk.out)
names(pressDF) <- as.character(seq(nComps))
pressDFres <- melt(pressDF)
bp <- ggplot(pressDFres, aes(x=variable, y=value)) + theme_bw() +
  geom_boxplot(notch=FALSE) + labs(x="Number of Components", y="PRESS") +
  theme(axis.text=element_text(size=18), legend.position="none",
        axis.title=element_text(size=20, face="bold"),
        axis.text.x = element_text(angle = 0,vjust = 0.5),
        panel.border = element_rect(linetype = "solid", fill = NA, size=1.5))
bp
```




```
# conduct t.test across components to identify first minimum - just one of the ways to do this
j <-2 #!!! actually need to confirm we should be doing this....!!!
results <- as.vector(array(data="NA", dim=c(nComps-1,1)))
for (i in seq_along(1:nComps-1)) {
  comp1 <- i; comp2 <- j
  ttest <- t.test(pressDFres$value[which(pressDFres$variable==comp1)],
                  pressDFres$value[which(pressDFres$variable==comp2)])
  #print(i)
  results[i] <- round(unlist(ttest$p.value),8)
  j <- j+1
  if (j > nComps) {
    break
  }
}
```

```
results <- data.frame(seq(2,nComps,1),results)
names(results) <- c("Component", "P.value")
results
```

Component P.value 1 2 0 2 3 0 3 4 0 4 5 3e-08 5 6 3.296e-05 6 7 0 7 8 4.3e-06 8 9 0 9 10 0.00113696 10 11 1.629e-05 11 12 0.04018846 12 13 0.00867417 13 14 0.05253998 14 15 0.19815572 15 16 0.09891252

```
# Simple final model validated with cross-validation.  Segmented cross-validation used
# given the very large sample size.  For models with fewer observations (e.g. <100)
# LOO or leave-one-out cross validation is recommended
first <- min(which(as.numeric(as.character(results$P.value)) > 0.05))
nComps <- results$Component[first]
print(paste0("*** Optimal number of components based on t.test: ", nComps))
```

[1] "*** Optimal number of components based on t.test: 14"

```
segs <- 30
plsr.out <- plsr(as.formula(paste(inVar,"~","Spectra")),scale=FALSE,ncomp=nComps,validation="CV",
                 segments=segs, segment.type="interleaved",trace=TRUE,data=cal.plsr.data)
```

Segment:

```
fit <- plsr.out$fitted.values[,1,nComps]
pls.options(parallel = NULL)
```

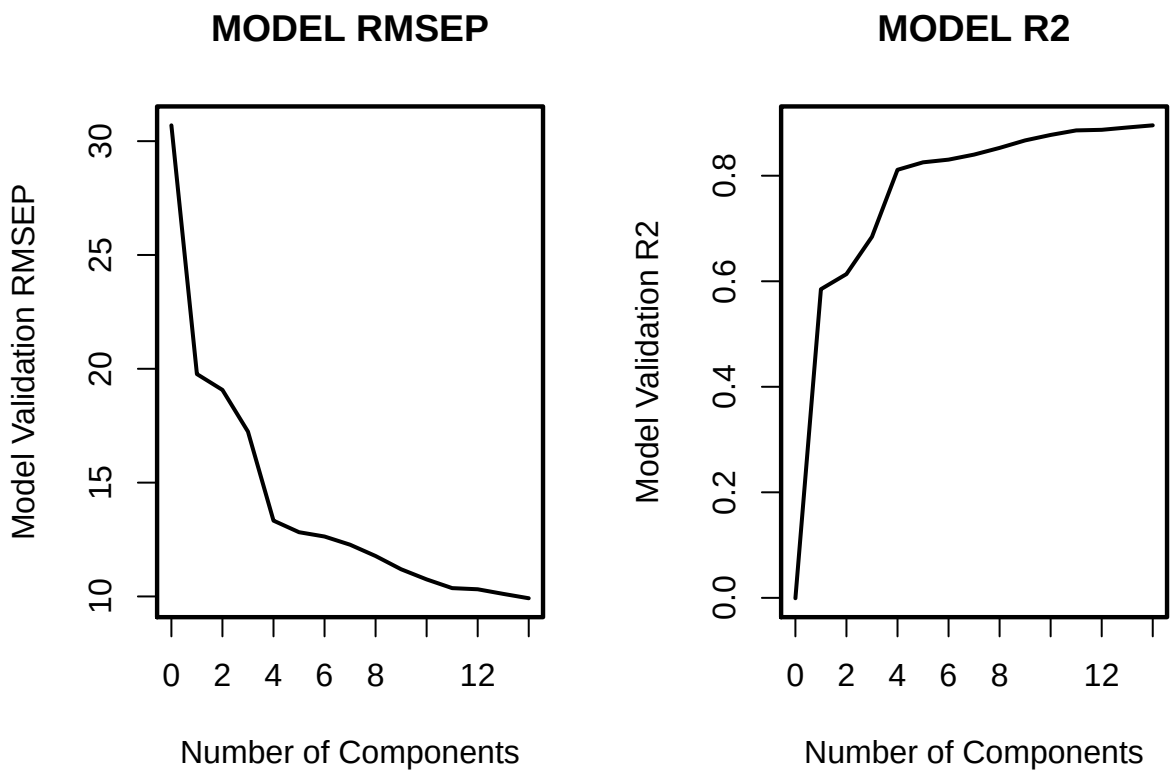
```
### Generate some initial PLSR results
# External validation
par(mfrow=c(1,2)) # B, L, T, R
RMSEP(plsr.out, newdata = val.plsr.data)
```

(Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps
30.69 19.77 19.07 17.24 13.33 12.82
6 comps 7 comps 8 comps 9 comps 10 comps 11 comps
12.63 12.27 11.78 11.19 10.75 10.37
12 comps 13 comps 14 comps
10.31 10.11 9.92

```
plot(RMSEP(plsr.out,estimate=c("test"),newdata = val.plsr.data), main="MODEL RMSEP",
     xlab="Number of Components",ylab="Model Validation RMSEP",lty=1,col="black",cex=1.5,lwd=2)
box(lwd=2.2)

R2(plsr.out, newdata = val.plsr.data)
```

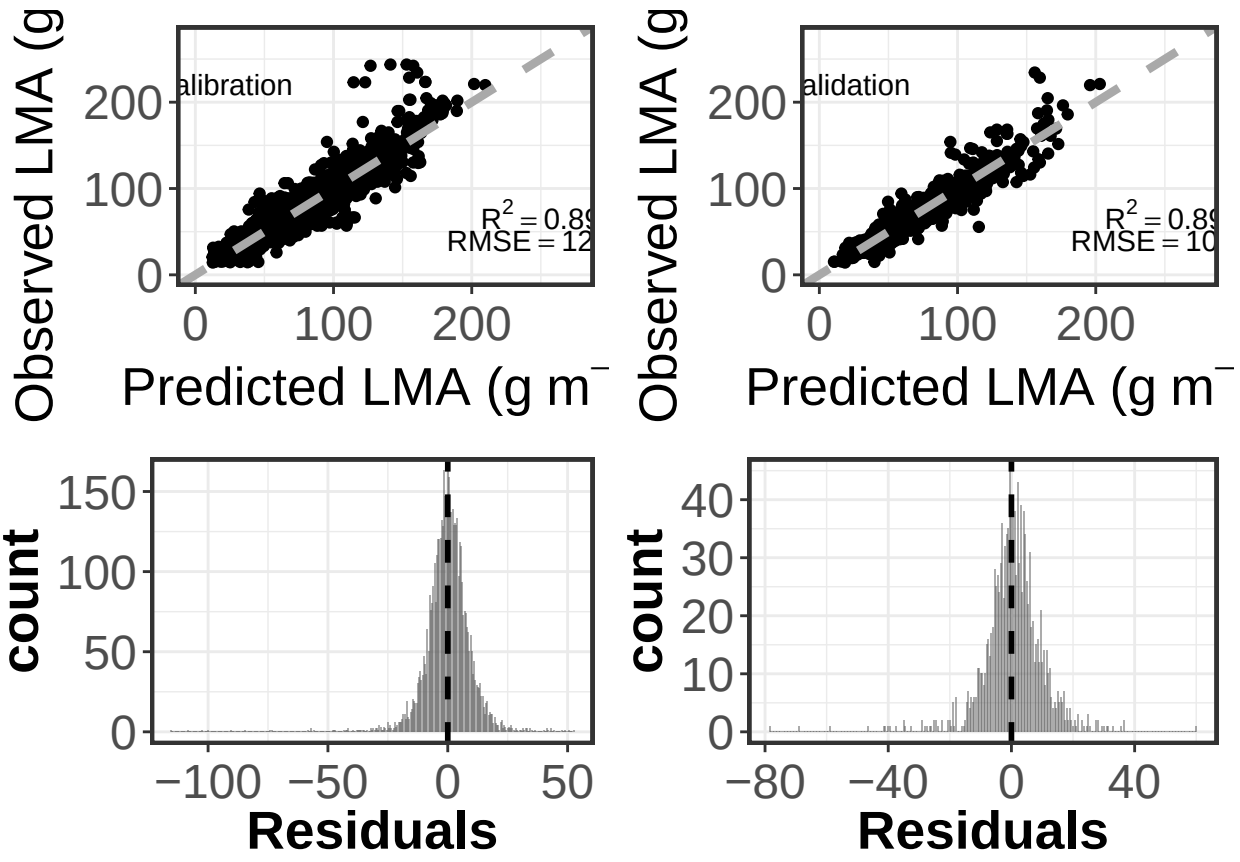
(Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps
-0.0007599 0.5849108 0.6135804 0.6842727 0.8112478 0.8253585
6 comps 7 comps 8 comps 9 comps 10 comps 11 comps
0.8305107 0.8400255 0.8526980 0.8669164 0.8772827 0.8858723
12 comps 13 comps 14 comps
0.8869825 0.8913726 0.8954615

```
plot(R2(plsr.out,estimate=c("test"),newdata = val.plsr.data), main="MODEL R2",
     xlab="Number of Components",ylab="Model Validation R2",lty=1,col="black",cex=1.5,lwd=2)
box(lwd=2.2)
```



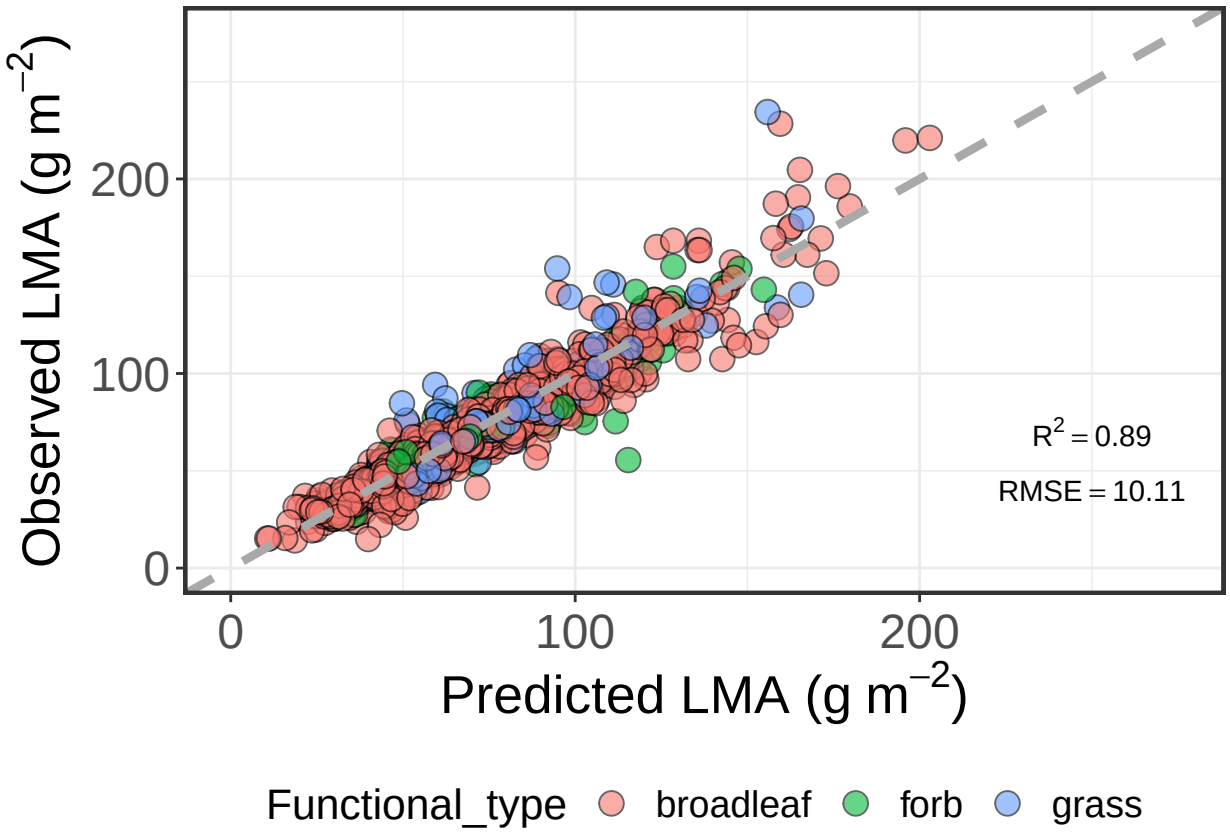

```
#calibration
cal_plot_data <- data.frame(cal.plsr.data[, which(names(cal.plsr.data) %notin% "Spectra")], Fitted=fit)
cal_plot_data <- cal_plot_data %>%
  mutate(Residuals = Fitted-LMA_gDW_m2)

# validation
val_plot_data <- data.frame(val.plsr.data[, which(names(val.plsr.data) %notin% "Spectra")],
                            Fitted=as.vector(predict(plsr.out, newdata = val.plsr.data, ncomp=nComps,
                                                     type="response")[,,1]))
val_plot_data <- val_plot_data %>%
  mutate(Residuals = Fitted-LMA_gDW_m2)

# cal
cal_scatter_plot <- ggplot(cal_plot_data, aes(x=Fitted, y=LMA_gDW_m2)) +
  theme_bw() + geom_point() + geom_abline(intercept = 0, slope = 1, color="dark grey",
                                          linetype="dashed", size=1.5) + xlim(0, 275) + ylim(0, 275) +
  labs(x=expression(paste("Predicted LMA (",g~m^{-2},")")),
       y=expression(paste("Observed LMA (",g~m^{-2},")"))) +
  annotate("text", x=250, y=70, label = paste0("R^2 == ", round(pls::R2(plsr.out)[[1]][nComps],2)), pars
  annotate("text", x=250, y=40, label = paste0("RMSE == ", round(pls::RMSEP(plsr.out)[[1]][nComps],2)),
  annotate("text",x=20,y=220,label="Calibration") +
  theme(axis.text=element_text(size=18), legend.position="none",
        axis.title=element_text(size=20, face="bold"),
        axis.text.x = element_text(angle = 0,vjust = 0.5),
        panel.border = element_rect(linetype = "solid", fill = NA, size=1.5))

cal_resid_histogram <- ggplot(cal_plot_data, aes(x=Residuals)) +
  geom_histogram(binwidth=.5, alpha=.5, position="identity") +
  geom_vline(xintercept = 0, color="black",
```

```
            linetype="dashed", size=1) + theme_bw() +
  theme(axis.text=element_text(size=18), legend.position="none",
        axis.title=element_text(size=20, face="bold"),
        axis.text.x = element_text(angle = 0,vjust = 0.5),
        panel.border = element_rect(linetype = "solid", fill = NA, size=1.5))

# val
val_scatter_plot <- ggplot(val_plot_data, aes(x=Fitted, y=LMA_gDW_m2)) +
  theme_bw() + geom_point() + geom_abline(intercept = 0, slope = 1, color="dark grey",
                                          linetype="dashed", size=1.5) + xlim(0, 275) + ylim(0, 275) +
  labs(x=expression(paste("Predicted LMA (",g~m^{-2},")")),
       y=expression(paste("Observed LMA (",g~m^{-2},")"))) +
  annotate("text", x=250, y=70, label = paste0("R^2 == ",
                                               round(pls::R2(plsr.out, newdata = val.plsr.data)[[1]][nC
  annotate("text", x=250, y=40, label = paste0("RMSE == ",
                                               round(pls::RMSEP(plsr.out, newdata = val.plsr.data)[[1]]
  annotate("text",x=20,y=220,label="Validation") +
  theme(axis.text=element_text(size=18), legend.position="none",
        axis.title=element_text(size=20, face="bold"),
        axis.text.x = element_text(angle = 0,vjust = 0.5),
        panel.border = element_rect(linetype = "solid", fill = NA, size=1.5))

val_resid_histogram <- ggplot(val_plot_data, aes(x=Residuals)) +
  geom_histogram(binwidth=.5, alpha=.5, position="identity") +
  geom_vline(xintercept = 0, color="black",
             linetype="dashed", size=1) + theme_bw() +
  theme(axis.text=element_text(size=18), legend.position="none",
        axis.title=element_text(size=20, face="bold"),
        axis.text.x = element_text(angle = 0,vjust = 0.5),
        panel.border = element_rect(linetype = "solid", fill = NA, size=1.5))

# plot cal/val side-by-side
grid.arrange(cal_scatter_plot, val_scatter_plot, cal_resid_histogram, val_resid_histogram,
             nrow=2,ncol=2)
```

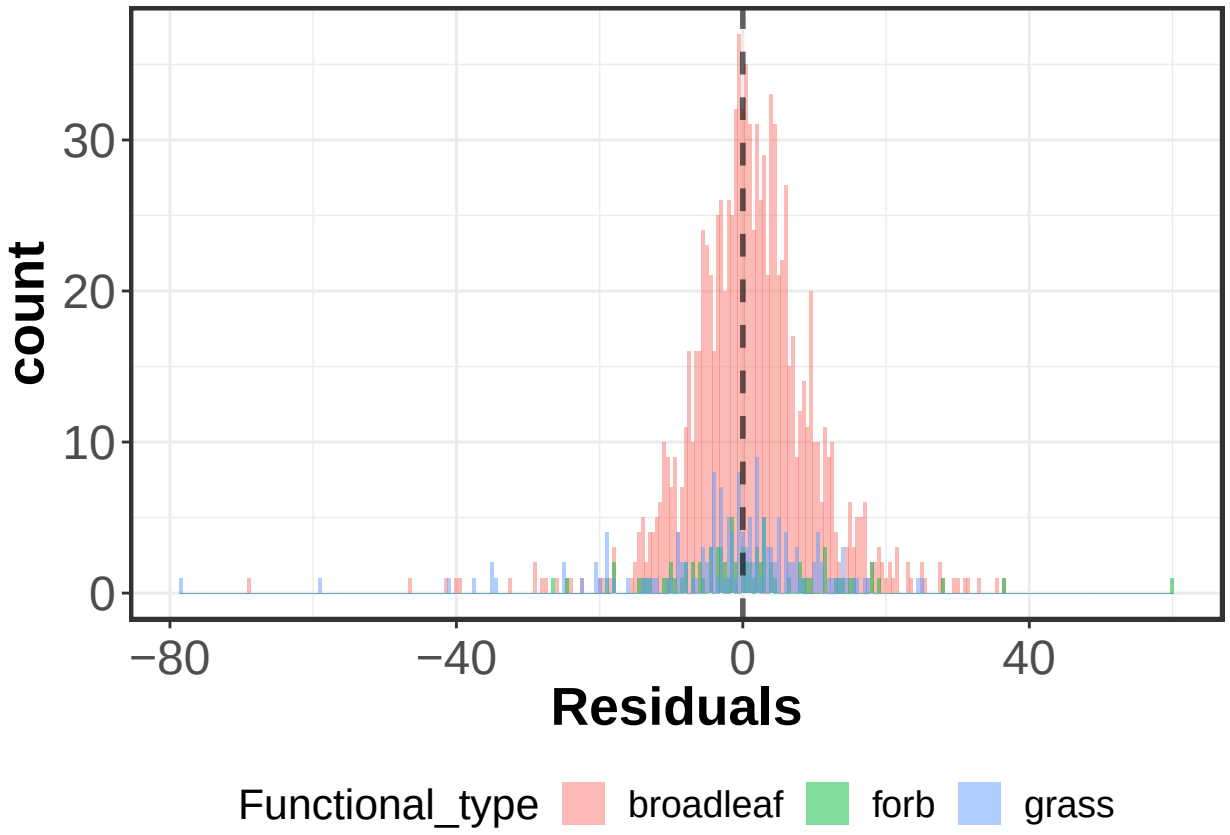


results by functional type and domain

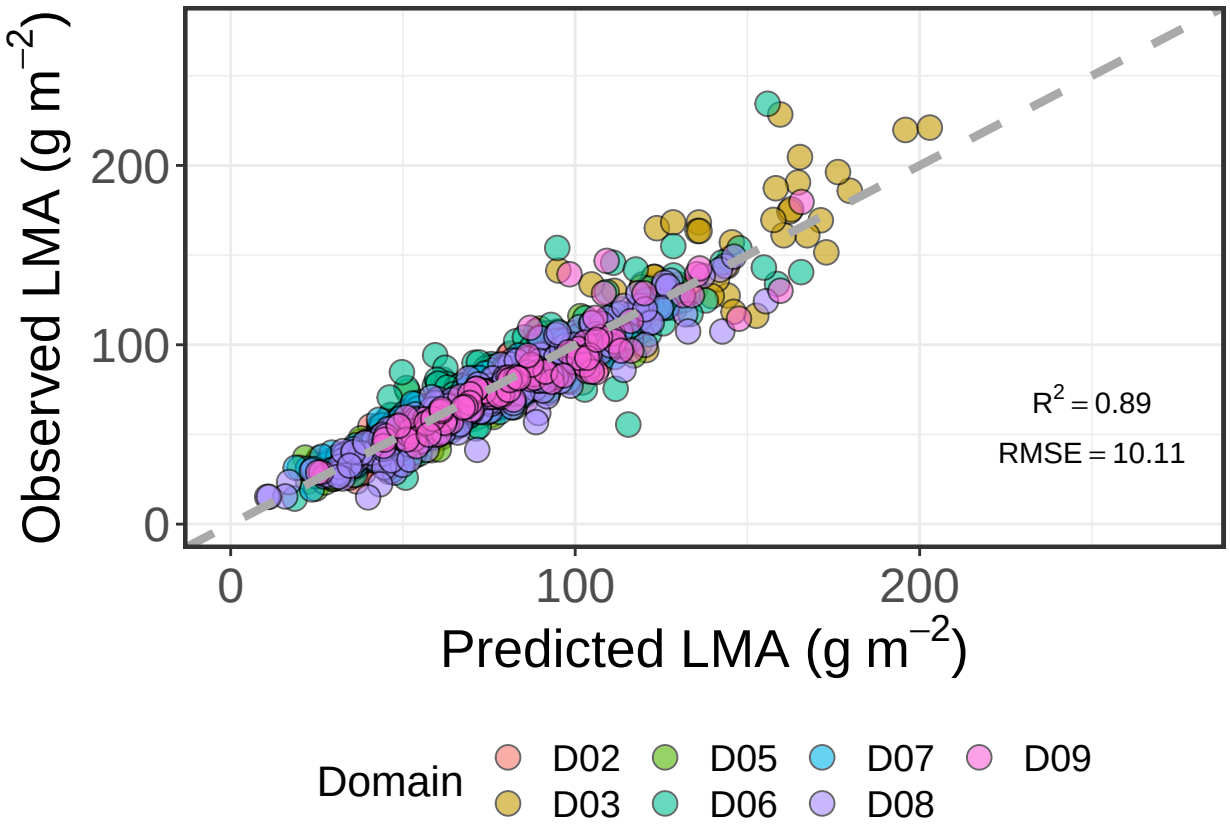
```
# validation by functional type
scatter_plot <- ggplot(val_plot_data, aes(x=Fitted, y=LMA_gDW_m2)) +
  theme_bw() + geom_point(aes(fill=Functional_type),alpha=0.6,colour="black", pch=21, size=4) +
  geom_abline(intercept = 0, slope = 1, color="dark grey",
              linetype="dashed", size=1.5) + xlim(0, 275) + ylim(0, 275) +
  labs(x=expression(paste("Predicted LMA (",g~m^{-2},")")),
       y=expression(paste("Observed LMA (",g~m^{-2},")"))) +
  annotate("text", x=250, y=70, label = paste0("R^2 == ",
                                               round(pls::R2(plsr.out, newdata = val.plsr.data)[[1]][nC
           parse=T) +
  annotate("text", x=250, y=40, label = paste0("RMSE == ",
                                               round(pls::RMSEP(plsr.out, newdata = val.plsr.data)[[1]]
           parse=T) +
  theme(axis.text=element_text(size=18), legend.position="bottom",legend.title=element_text(size=16),
        legend.text=element_text(size=14),
        axis.title=element_text(size=20, face="bold"),
        axis.text.x = element_text(angle = 0,vjust = 0.5),
        panel.border = element_rect(linetype = "solid", fill = NA, size=1.5))
scatter_plot
```



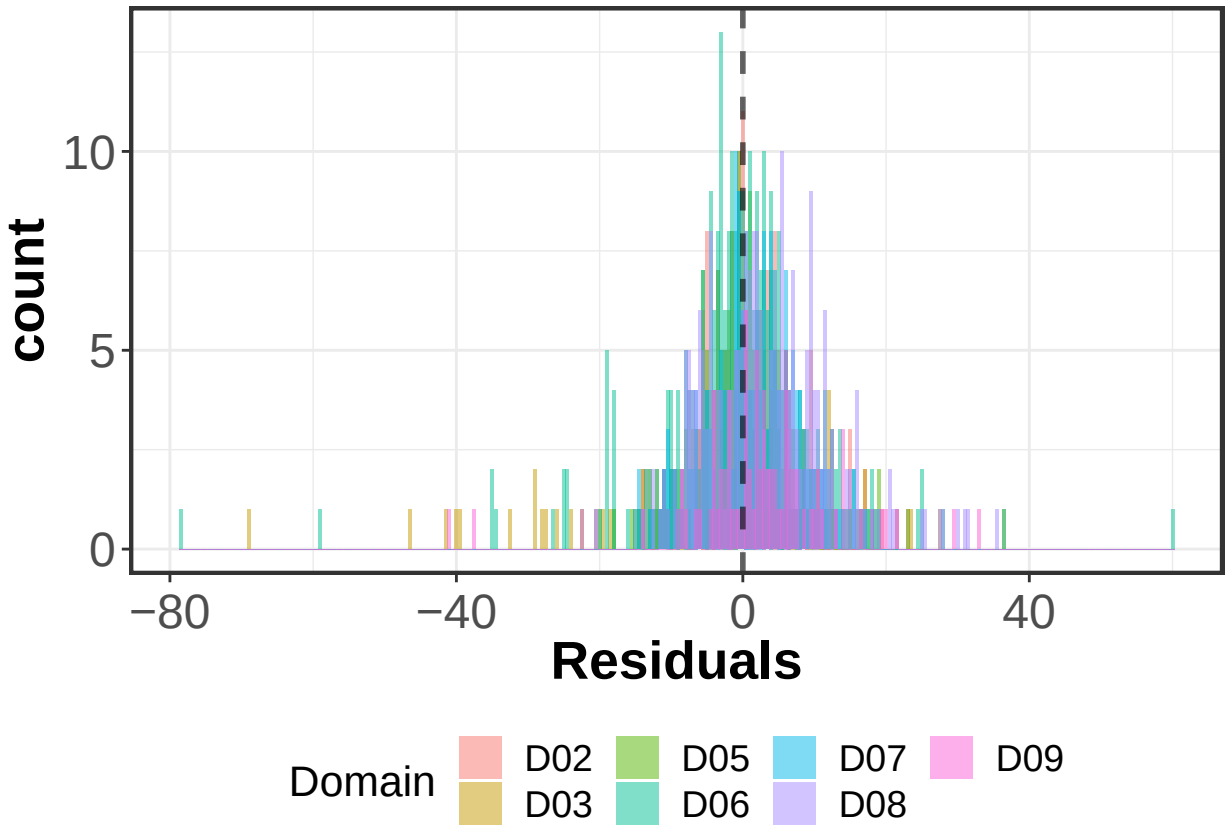

```
resid_histogram <- ggplot(val_plot_data, aes(x=Residuals, fill=Functional_type)) +
  geom_histogram(binwidth=.5, alpha=.5, position="identity") +
  geom_vline(xintercept = 0, color="black", alpha=0.6,
             linetype="dashed", size=1) + theme_bw() +
  theme(axis.text=element_text(size=18), legend.position="bottom",legend.title=element_text(size=16),
        legend.text=element_text(size=14),
        axis.title=element_text(size=20, face="bold"),
        axis.text.x = element_text(angle = 0,vjust = 0.5),
        panel.border = element_rect(linetype = "solid", fill = NA, size=1.5))
resid_histogram
```




```
# NEON domain
scatter_plot <- ggplot(val_plot_data, aes(x=Fitted, y=LMA_gDW_m2)) +
  theme_bw() + geom_point(aes(fill=Domain),alpha=0.6,colour="black", pch=21, size=4) +
  geom_abline(intercept = 0, slope = 1, color="dark grey",
              linetype="dashed", size=1.5) + xlim(0, 275) + ylim(0, 275) +
  labs(x=expression(paste("Predicted LMA (",g~m^{-2},")")),
       y=expression(paste("Observed LMA (",g~m^{-2},")"))) +
  annotate("text", x=250, y=70, label = paste0("R^2 == ",
                                               round(pls::R2(plsr.out, newdata = val.plsr.data)[[1]][nC
           parse=T) +
  annotate("text", x=250, y=40, label = paste0("RMSE == ",
                                               round(pls::RMSEP(plsr.out, newdata = val.plsr.data)[[1]]
           parse=T) +
  theme(axis.text=element_text(size=18), legend.position="bottom",legend.title=element_text(size=16),
        legend.text=element_text(size=14),
        axis.title=element_text(size=20, face="bold"),
        axis.text.x = element_text(angle = 0,vjust = 0.5),
        panel.border = element_rect(linetype = "solid", fill = NA, size=1.5))
scatter_plot
```




```
resid_histogram <- ggplot(val_plot_data, aes(x=Residuals, fill=Domain)) +
  geom_histogram(binwidth=.5, alpha=.5, position="identity") +
  geom_vline(xintercept = 0, color="black", alpha=0.6,
             linetype="dashed", size=1) + theme_bw() +
  theme(axis.text=element_text(size=18), legend.position="bottom",legend.title=element_text(size=16),
        legend.text=element_text(size=14),
        axis.title=element_text(size=20, face="bold"),
        axis.text.x = element_text(angle = 0,vjust = 0.5),
        panel.border = element_rect(linetype = "solid", fill = NA, size=1.5))
resid_histogram
```



Generate some useful outputs

```
cal.plsr.pred <- as.vector(plsr.out$fitted.values[,,nComps]) # Model fitted values. Predicted values
cal.plsr.CVpred <- as.vector(plsr.out$validation$pred[,,nComps]) # CV pred values
cal.CVresiduals <- as.vector(plsr.out$residuals[,,nComps]) # CV pred residuals
cal.output <- data.frame(cal.plsr.data[,which(names(cal.plsr.data) %notin% "Spectra")],
                         PLSR_Predicted=cal.plsr.pred, PLSR_CV_Predicted=cal.plsr.CVpred,
                         PLSR_CV_Residuals=cal.CVresiduals)
head(cal.output)
```

```
Domain Functional_type Sample_ID USDA_Species_Code LMA_gDW_m2
```

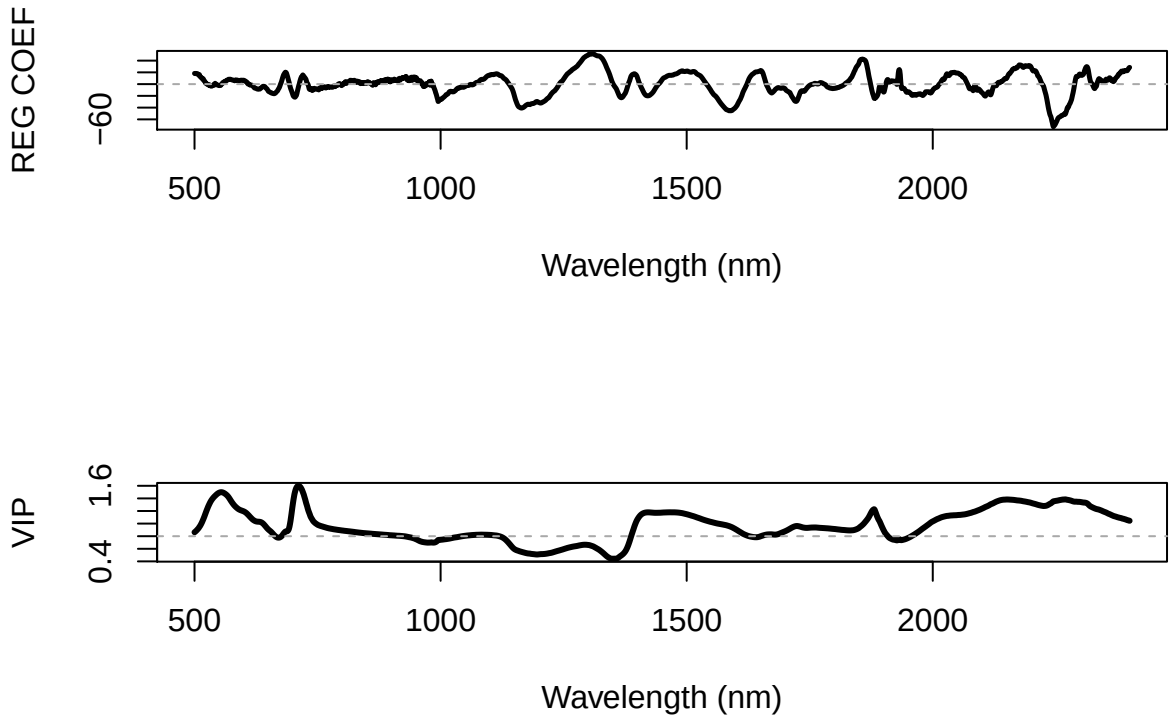
636 D02 broadleaf L0318 QUFA 111.19 460 D02 broadleaf L0230 FAGR 23.71 476 D02 broadleaf L0238 CATO 34.88 1 D02 broadleaf P0001 JUNI 72.87 364 D02 broadleaf L0182 QUAL 50.59 771 D02 broadleaf P0386 PLOC 50.91 PLSR_Predicted PLSR_CV_Predicted PLSR_CV_Residuals 636 122.79872 122.87748 -11.608720 460 34.06507 34.07325 -10.355069 476 37.50026 37.55687 -2.620262 1 76.96557 77.49305 -4.095565 364 53.52878 53.55954 -2.938778 771 52.76685 52.85561 -1.856847

```
rm(cal.plsr.pred,cal.plsr.CVpred,cal.CVresiduals) #cleanup

predicted_val <- as.vector(predict(plsr.out, newdata = val.plsr.data, ncomp=nComps, type="response")[,,
predicted_val_residuals <- predicted_val-val.plsr.data[,inVar]
val.output <- data.frame(val.plsr.data[,which(names(cal.plsr.data) %notin% "Spectra")],
                         PLSR_Predicted=predicted_val,PLSR_Residuals=predicted_val_residuals)
head(val.output)
```

Domain Functional_type Sample_ID USDA_Species_Code LMA_gDW_m2 PLSR_Predicted 4 D02

broadleaf L0002 JUNI 60.77 53.44240 5 D02 broadleaf P0003 JUNI 85.92 84.39148 9 D02 broadleaf P0005 JUNI 48.76 41.41186 17 D02 broadleaf P0009 QUVE 84.92 77.73390 20 D02 broadleaf L0010 PRSE 78.82 71.82721 21 D02 broadleaf P0011 PRSE 86.09 82.78854 PLSR_Residuals 4 -7.327603 5 -1.528519 9 -7.348142 17 -7.186101 20 -6.992792 21 -3.301464

```
rm(predicted_val,predicted_val_residuals) #cleanup

coefs <- coef(plsr.out,ncomp=nComps,intercept=FALSE)
vips <- VIP(plsr.out)[nComps,]

# Coefficient and VIP plot for PLSR model !! This plotting could be improved !!
par(mfrow=c(2,1))
plot(seq(Start.wave,End.wave,1),coefs,cex=0.01,xlab="Wavelength (nm)",ylab="REG COEF")
lines(seq(Start.wave,End.wave,1),coefs,lwd=2.5)
abline(h=0,lty=2,col="dark grey")

plot(seq(Start.wave,End.wave,1),vips,xlab="Wavelength (nm)",ylab="VIP",cex=0.01)
lines(seq(Start.wave,End.wave,1),vips,lwd=3)
abline(h=0.8,lty=2,col="dark grey")
```



Output results

```
print(paste("Output directory: ", getwd()))
```

[1] "Output directory: /private/var/folders/xp/h3k9vf3n2jx181ts786_yjrn9c2gjq/T/RtmpP87nwE"

```
# Observed versus predicted
write.csv(cal.output,file=file.path(outdir,paste0(inVar,'_Observed_PLSR_CV_Pred_',nComps,
                                  'comp.csv')),row.names=FALSE)

# Validation data
```

```
write.csv(val.output,file=file.path(outdir,paste0(inVar,'_Val_PLSR_Pred_',nComps,
                                    'comp.csv')),row.names=FALSE)

# Model coefficients
coefs <- coef(plsr.out,ncomp=nComps,intercept=TRUE)
write.csv(coefs,file=file.path(outdir,paste0(inVar,'_PLSR_Coefficients_',nComps,'comp.csv')),
          row.names=TRUE)

# PLSR VIP
write.csv(vips,file=file.path(outdir,paste0(inVar,'_PLSR_VIPs_',nComps,'comp.csv')))

# confirm files were written to temp space
print("**** PLSR output files: ")
```

[1] "**** PLSR output files:"

```
list.files(getwd())[grep(pattern = inVar, list.files(getwd()))]
```

[1] "LMA_gDW_m2_Cal_PLSR_Dataset.csv"
[2] "LMA_gDW_m2_Full_PLSR_Dataset.csv"
[3] "LMA_gDW_m2_Observed_PLSR_CV_Pred_14comp.csv" [4] "LMA_gDW_m2_PLSR_Coefficients_14comp.csv"
[5] "LMA_gDW_m2_PLSR_VIPs_14comp.csv"
[6] "LMA_gDW_m2_Val_PLSR_Dataset.csv"
[7] "LMA_gDW_m2_Val_PLSR_Pred_14comp.csv"

Jackknife Model Evaluation - !!!This section needs work!!!

```
nComps
```

[1] 14

```
resamples <- 100 #1000 !! this should be more like 100 or 1000, just set to 10 for now for testing !!!
output.jackknife.stats <- data.frame(Rsq=rep(NA,resamples),RMSEP=rep(NA,resamples),
                                     PERC_RMSEP=rep(NA,resamples), Bias=rep(NA,resamples))
output.jackknife.coefs <- array(data=NA,dim=c(resamples,
                                              dim(coef(plsr.out,ncomp=nComps,intercept=TRUE))[1]))
output.jackknife.coefs.scaled <- array(data=NA,dim=c(resamples,
                                                     dim(coef(plsr.out,ncomp=nComps,intercept=TRUE))[1]))
vips <- array(data=NA,dim=c(resamples,
                            dim(coef(plsr.out,ncomp=nComps,intercept=FALSE))[1]))

for (i in 1:resamples) {
  rows <- sample(1:nrow(cal.plsr.data),floor(0.7*nrow(cal.plsr.data)))
  cal.data.jk <- cal.plsr.data[rows,]
  val.data.jk <- cal.plsr.data[-rows,]

  dimsCal <- dim(cal.data.jk)
  dimsVal <- dim(val.data.jk)

  ### Build PLSR model with training data
  if(grepl("Windows", sessionInfo()$running)){
    pls.options(parallel =NULL)
  } else {
    pls.options(parallel = parallel::detectCores()-1)
```

```
  }
  pls.jack <- plsr(as.formula(paste0(inVar,"~","Spectra")), scale=FALSE, ncomp=nComps, validation="none
  pls.jack.scaled <- plsr(as.formula(paste0(inVar,"~","Spectra")), scale=TRUE, ncomp=nComps, validation=

  ### Estimate independent (Validation) samples
  plsr.val <- val.data.jk[,inVar]
  pred.val.data <- as.vector(predict(pls.jack,newdata=val.data.jk$Spectra,
                                     ncomp=nComps,type="response")[,,1])

  ### Coefficients and VIPs
  output.jackknife.coefs[i,] <- as.vector(coef(pls.jack,ncomp=nComps,intercept=TRUE))
  output.jackknife.coefs.scaled[i,] <- as.vector(coef(pls.jack.scaled,ncomp=nComps,intercept=TRUE))
  vips[i,] <- VIP(pls.jack)[nComps,]

  # Error statistics
  n <- length(plsr.val)
  Rsq.val <- R2(pls.jack,newdata = val.data.jk)$val[,,nComps+1]
  val.residuals <- pred.val.data-plsr.val
  Val.bias <- mean(pred.val.data)-mean(plsr.val)
  MSEP <- mean(val.residuals^2)
  RMSEP.val <- sqrt(MSEP)
  PERC_RMSEP <- (RMSEP.val/(max(plsr.val)-min(plsr.val)))*100

  ### Store results of iteration i
  output.jackknife.stats[i,1] <- Rsq.val
  output.jackknife.stats[i,2] <- RMSEP.val
  output.jackknife.stats[i,3] <- PERC_RMSEP
  output.jackknife.stats[i,4] <- Val.bias

  #print(paste("Running Iteration",i))
  #print(paste("Stats: ","Rsq ",round(Rsq.val,2)," / RMSEP ",round(RMSEP.val,2), " / %RMSEP ",
  #            round(PERC_RMSEP,2)," / Bias ",round(Val.bias,2), sep="" ) )
  #flush.console()  # force the output

  # Remove temp objects
  rm(cal.data.jk,val.data.jk,n,pls.jack,plsr.val,Rsq.val,pred.val.data,RMSEP.val,PERC_RMSEP,
     val.residuals,Val.bias)
}
```

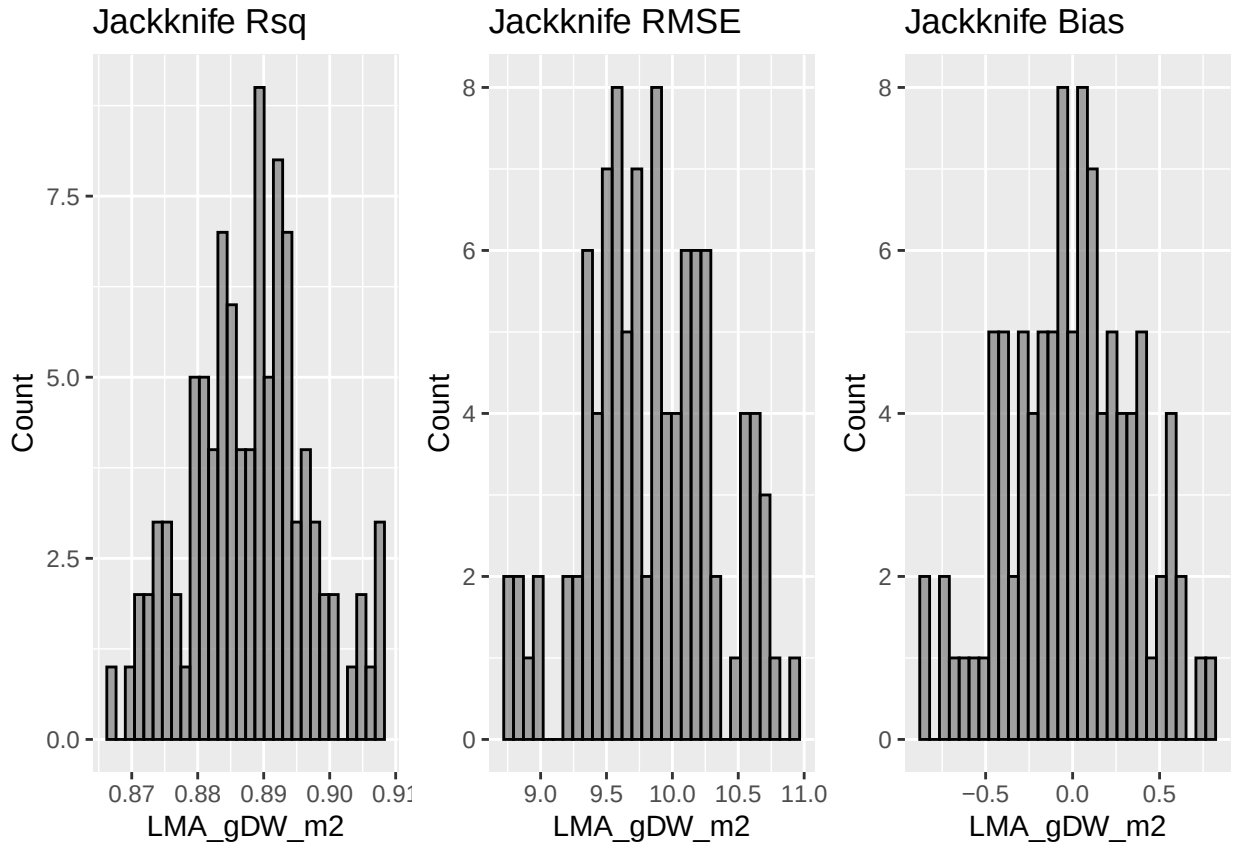
Histogram statistics

```
write.csv(output.jackknife.stats,file=file.path(outdir,paste0(inVar,'_Jackkife_PLSR_Resutls.csv')),
          row.names=FALSE)

head(output.jackknife.stats)
```

```
    Rsq     RMSEP PERC_RMSEP        Bias
```

1 0.8994387 9.000367 4.208926 0.19296664 2 0.8918579 9.816252 4.285637 0.08638655 3 0.9074502 8.739256 4.251231 0.81417043 4 0.8830323 9.980613 4.350747 0.54104277 5 0.8807645 10.014945 4.365713 0.21859792 6 0.8704313 10.545666 4.610531 -0.24629719

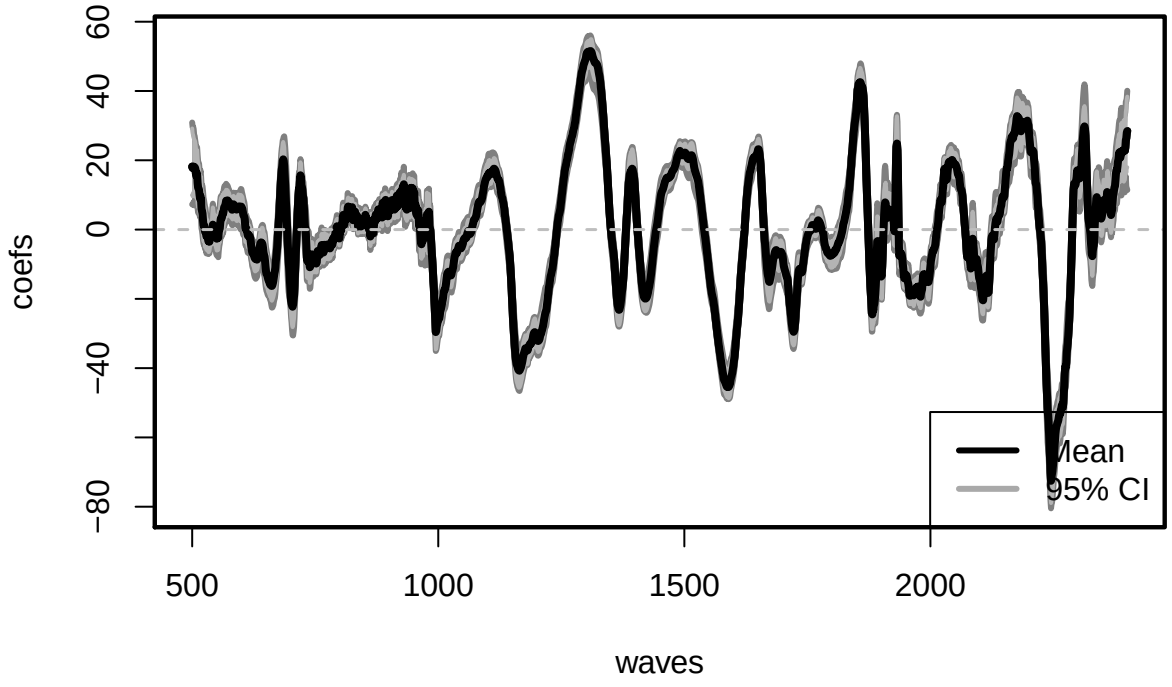
```
rsqHist <- qplot(output.jackknife.stats[,"Rsq"],geom="histogram",
                 main = "Jackknife Rsq",
                 xlab = paste0(inVar),ylab = "Count",fill=I("grey50"),col=I("black"),alpha=I(.7))
rmseHist <- qplot(output.jackknife.stats[,"RMSEP"],geom="histogram",
                 main = "Jackknife RMSE",
                 xlab = paste0(inVar),ylab = "Count",fill=I("grey50"),col=I("black"),alpha=I(.7))
biasHist <- qplot(output.jackknife.stats[,"Bias"],geom="histogram",
                  main = "Jackknife Bias",
                  xlab = paste0(inVar),ylab = "Count",fill=I("grey50"),col=I("black"),alpha=I(.7))
grid.arrange(rsqHist, rmseHist, biasHist, ncol=3)
```



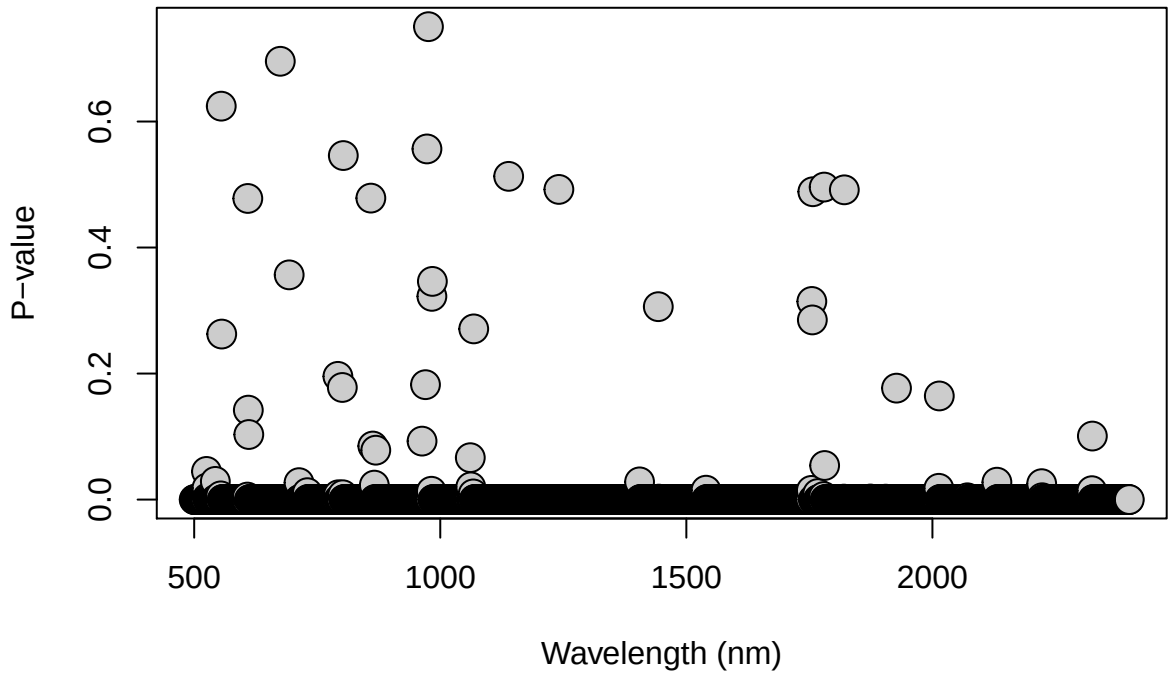
plot jackknife coefficients

```
dims <- dim(output.jackknife.coefs)
plot.coefs <- output.jackknife.coefs[,2:dims[2]]
plot.min <- min(output.jackknife.coefs[,2:dims[2]])
plot.max <- max(output.jackknife.coefs[,2:dims[2]])
jk.intercepts <- output.jackknife.coefs[,1]

# Stats
coef.means <- colMeans(plot.coefs)
sd.coef <- apply(plot.coefs,MARGIN=2,FUN=function(x)sd(x))
min.coef <- apply(plot.coefs,MARGIN=2,FUN=function(x)min(x))
max.coef <- apply(plot.coefs,MARGIN=2,FUN=function(x)max(x))
coef.quant <- apply(plot.coefs,2,quantile,probs=c(0.025,0.975))
```

```
intercepts.quant <- quantile(jk.intercepts,probs=c(0.025,0.975))

# T Test
x <- resamples # From Jackknife above
results <- apply(plot.coefs,2, function(plot.coefs) {
  t.test(x = plot.coefs[1:x])$p.value})

waves <- seq(Start.wave,End.wave,1)
coefs <- as.vector(coef(plsr.out,ncomp=nComps,intercept=FALSE))
plot(waves,coefs,type="l",lwd=4,ylim=c(plot.min,plot.max))
# Min/Max
polygon(c(waves ,rev(waves)),c(max.coef, rev(min.coef)),col="grey50",border=NA)
lines(waves,min.coef,lty=1,lwd=3,col="grey50")
lines(waves,max.coef,lty=1,lwd=3,col="grey50")
# 95% CIs
polygon(c(waves ,rev(waves)),c(coef.quant[2,], rev(coef.quant[1,])),col="grey70",border=NA)
lines(waves,coef.quant[1,],lty=1,lwd=2,col="grey70")
lines(waves,coef.quant[2,],lty=1,lwd=2,col="grey70")
# replot the mean and zero line
lines(waves,coefs,lwd=4)
abline(h=0,lty=2,col="grey",lwd=1.5)
legend("bottomright",legend=c("Mean","95% CI"),lty=c(1,1),
       col=c("black","dark grey"),lwd=3)
box(lwd=2.2)
```




```
# P-vals
plot(waves,results,pch=21,bg="grey80",ylab="P-value",xlab="Wavelength (nm)",
     cex=2)
```



Output jackknife results

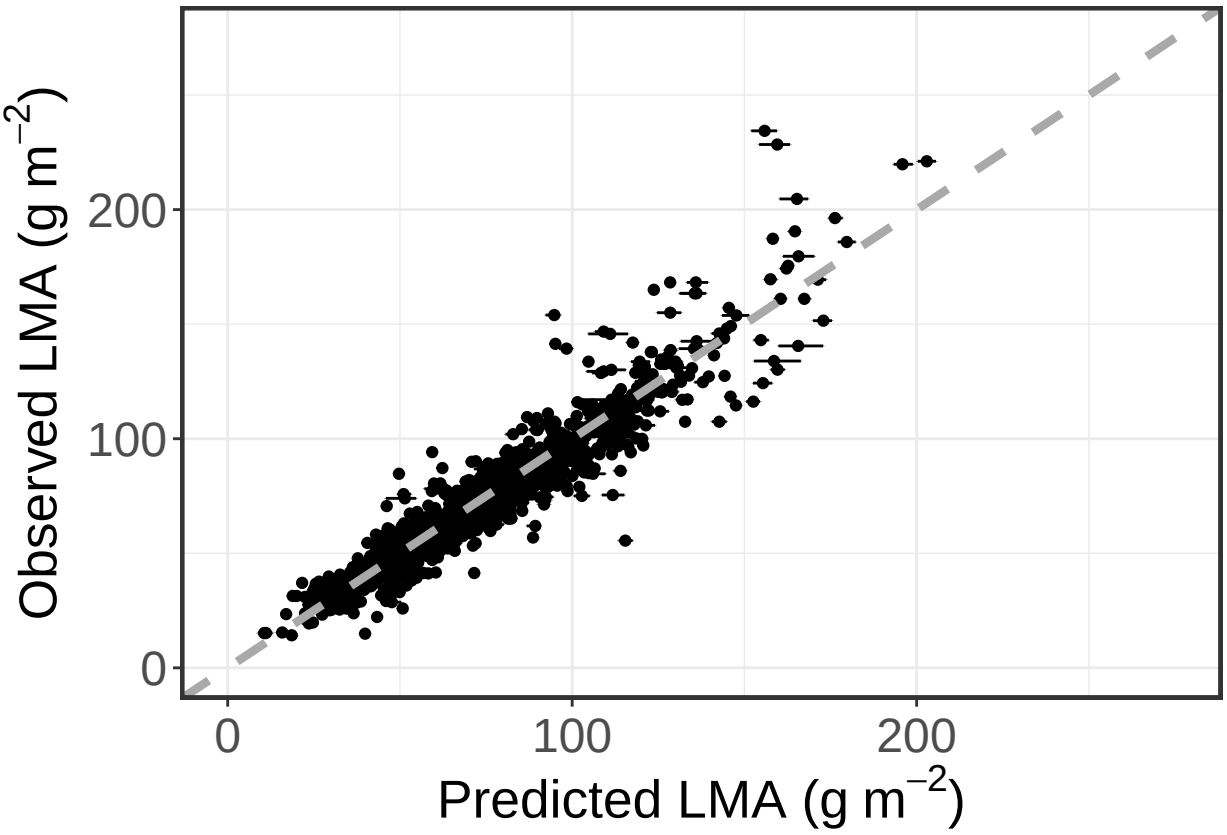
```
# JK Coefficents
out.jk.coefs <- data.frame(Iteration=seq(1,resamples,1),jk.intercepts,plot.coefs)
names(out.jk.coefs) <- c("Iteration","Intercept",paste("Wave_",seq(Start.wave,End.wave,1),sep=""))
write.csv(out.jk.coefs,file=file.path(outdir,paste0(inVar,'_Jackkife_PLSR_Coefficients.csv')),
          row.names=FALSE)

# VIPs
out.jk.vips <- data.frame(Iteration=seq(1,resamples,1),vips)
names(out.jk.vips) <- c("Iteration",paste("Wave_",seq(Start.wave,End.wave,1),sep=""))
write.csv(out.jk.vips,file=file.path(outdir,paste0(inVar,'_Jackkife_PLSR_VIPs.csv')),
          row.names=FALSE)

# Coeff quantiles
out.coef.quant <- array(data=NA,dim=c(2,dim(out.jk.coefs)[2]))
out.coef.quant[1,1] <- "5%"
out.coef.quant[2,1] <- "95%"
out.coef.quant[1,2] <- intercepts.quant[[1]]
out.coef.quant[2,2] <- intercepts.quant[[2]]
out.coef.quant[,3:dim(out.jk.coefs)[2]] <- coef.quant
out.coef.quant <- data.frame(out.coef.quant)

names(out.coef.quant) <- c("Quantile","Intercept",paste("Wave_",seq(Start.wave,End.wave,1),sep=""))

write.csv(out.coef.quant,file=file.path(outdir,paste0(inVar,'_Jackkife_PLSR_Coefficient_Quantiles.csv')),
          row.names=TRUE)
# P-vals
out.pvals <- data.frame(Wavelength=paste("Wave_",seq(Start.wave,End.wave,1),sep=""),Pval=results)
write.csv(out.pvals,file=file.path(outdir,paste0(inVar,'_Jackkife_PLSR_Coefficient_Pvals.csv')),
          row.names=FALSE)
```

JK Val plot

```
dims <- dim(output.jackknife.coefs)
intercepts <- output.jackknife.coefs[,1]
jk.coef.test.output <- array(data=NA,dim=c(dim(val.plsr.data)[1],dims[1]))
for (i in 1:length(intercepts)){
  coefs <- as.vector(output.jackknife.coefs[i,2:dims[2]])
  temp <- val.plsr.data$Spectra %*% coefs  # Updated: Using matrix mult.
  vals <- data.frame(rowSums(temp))+intercepts[i]
  jk.coef.test.output[,i] <- vals[,1]
}
pred.quant <- apply(jk.coef.test.output,1,quantile,probs=c(0.025,0.975))
pred.quant.ll <- pred.quant[1,]
pred.quant.ul <- pred.quant[2,]

jk_val_plot_data <- data.frame(val.output, LL=pred.quant.ll, UL=pred.quant.ul)
head(jk_val_plot_data)
```

Domain Functional_type Sample_ID USDA_Species_Code LMA_gDW_m2 PLSR_Predicted 4 D02 broadleaf L0002 JUNI 60.77 53.44240 5 D02 broadleaf P0003 JUNI 85.92 84.39148 9 D02 broadleaf P0005 JUNI 48.76 41.41186 17 D02 broadleaf P0009 QUVE 84.92 77.73390 20 D02 broadleaf L0010 PRSE 78.82 71.82721 21 D02 broadleaf P0011 PRSE 86.09 82.78854 PLSR_Residuals LL UL 4 -7.327603 52.74049 54.27251 5 -1.528519 83.10181 85.81704 9 -7.348142 40.18355 42.60245 17 -7.186101 76.76737 78.52871 20 -6.992792 70.99487 72.83655 21 -3.301464 81.48057 84.23899

```
# plot needs cleaning up. draft
jk_val_scatterplot <- ggplot(jk_val_plot_data, aes(x=PLSR_Predicted, y=LMA_gDW_m2)) +
  theme_bw() + geom_point() + geom_errorbar(aes(xmin = LL,xmax = UL), width = 0.2) +
  geom_abline(intercept = 0, slope = 1, color="dark grey",
              linetype="dashed", size=1.5) + xlim(0, 275) + ylim(0, 275) +
  labs(x=expression(paste("Predicted LMA (",g~m^{-2},")")),
       y=expression(paste("Observed LMA (",g~m^{-2},")"))) +
  theme(axis.text=element_text(size=18), legend.position="none",
        axis.title=element_text(size=20, face="bold"),
        axis.text.x = element_text(angle = 0,vjust = 0.5),
        panel.border = element_rect(linetype = "solid", fill = NA, size=1.5))
jk_val_scatterplot
```



Output JK Coefficient test results

```
jk.coef.test.output2 <- data.frame(jk.coef.test.output)
names(jk.coef.test.output2) <- paste("Iteration.",seq(1,resamples,1),sep="")
jk.coef.test.output2 <- data.frame(Observed.Values=val.output[,"LMA_gDW_m2"],
                                   jk.coef.test.output2)

write.csv(jk.coef.test.output2,file=file.path(outdir,paste0(inVar,'_Jackkife_PLSR_Val_Data_Output.csv')),
          row.names=FALSE)
```