Spectra-trait PLSR example using NEON AOP pixel spectra and field-sampled leaf nitrogen content from CONUS NEON sites

Shawn P. Serbin, Julien Lamour, & Jeremiah Anderson

Overview

This is an R Markdown Notebook to illustrate how to develop pixel-scale spectra-trait PLSR models. This example uses image data from NEON AOP and associated field measurements of leaf nitrogen content collected across a range of CONUS NEON sites. For more information refer to the dataset EcoSIS page: https://ecosis.org/package/canopy-spectra-to-map-foliar-functional-traits-over-neon-domains-in-eastern-united-states

Getting Started

Load libraries

```
list.of.packages <- c("pls","dplyr","reshape2","here","plotrix","ggplot2","gridExtra",
                      "spectratrait")
invisible(lapply(list.of.packages, library, character.only = TRUE))
```

```
##
## Attaching package: 'pls'

## The following object is masked from 'package:stats':
##
##     loadings

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

## here() starts at /Users/sserbin/Data/GitHub/PLSR_for_plant_trait_prediction

##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

Setup other functions and options

```r
### Setup other functions and options
# not in
`%notin%` <- Negate(`%in%`)

# Script options
pls::pls.options(plsralg = "oscorespls")
pls::pls.options("plsralg")
```

```
## $plsralg
## [1] "oscorespls"
```

```r
# Default par options
opar <- par(no.readonly = T)

# What is the target variable? What is the variable name in the input dataset?
inVar <- "Nitrogen"

# What is the source dataset from EcoSIS?
ecosis_id <- "b9dbf3db-5b9c-4ab2-88c2-26c8b39d0903"

# Specify output directory, output_dir
# Options:
# tempdir - use a OS-specified temporary directory
# user defined PATH - e.g. "~/scratch/PLSR"
output_dir <- "tempdir"
```

Set working directory (scratch space)

```
## [1] "/private/var/folders/xp/h3k9vf3n2jx181ts786_yjrn9c2gjq/T/RtmpigxOMb"
```

Grab data from EcoSIS

```r
print(paste0("Output directory: ",getwd()))  # check wd
```

```
## [1] "Output directory: /Users/sserbin/Data/GitHub/PLSR_for_plant_trait_prediction/vignettes"
```

```r
dat_raw <- spectratrait::get_ecosis_data(ecosis_id = ecosis_id)
```

```
## [1] "**** Downloading Ecosis data ****"
```

```
## Downloading data...
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double(),
##   Affiliation = col_character(),
##   PI = col_character(),
##   Plot_ID = col_character(),
##   Project = col_character()
## )
## i Use `spec()` for the full column specifications.
```

```
## Download complete!
```

```
head(dat_raw)
```

```
## # A tibble: 6 x 459
##   Affiliation   Boron Calcium Carbon Carotenoids_area Carotenoids_mass Cellulose
##   <chr>         <dbl>   <dbl>  <dbl>            <dbl>            <dbl>     <dbl>
## 1 University ~ 0.0420   24.2    463.             9.19             1.18      221.
## 2 University ~ 0.0361    6.90   558.            10.8              1.17      183.
## 3 University ~ 0.0407   16.7    532.            12.2              1.52      133.
## 4 University ~ 0.0461   13.9    461.             9.16             1.50      220.
## 5 University ~ 0.0401   13.7    510.            11.0              1.53      101.
## 6 University ~ 0.0456   14.5    557.             8.90             1.24      214.
## # ... with 452 more variables: Chlorophylls_area <dbl>,
## #   Chlorophylls_mass <dbl>, Copper <dbl>, EWT <dbl>, Fiber <dbl>,
## #   Flavonoids <dbl>, LMA <dbl>, Lignin <dbl>, Magnesium <dbl>,
## #   Manganese <dbl>, NSC <dbl>, Nitrogen <dbl>, PI <chr>, Phenolics <dbl>,
## #   Phosphorus <dbl>, Plot_ID <chr>, Potassium <dbl>, Project <chr>, SLA <dbl>,
## #   Sample_Year <dbl>, Starch <dbl>, Sugar <dbl>, Sulfur <dbl>, Water <dbl>,
## #   d13C <dbl>, d15N <dbl>, 384 <dbl>, 389 <dbl>, 394 <dbl>, 399 <dbl>,
## #   404 <dbl>, 409 <dbl>, 414 <dbl>, 419 <dbl>, 424 <dbl>, 429 <dbl>,
## #   434 <dbl>, 439 <dbl>, 444 <dbl>, 449 <dbl>, 454 <dbl>, 459 <dbl>,
## #   464 <dbl>, 469 <dbl>, 474 <dbl>, 479 <dbl>, 484 <dbl>, 489 <dbl>,
## #   494 <dbl>, 499 <dbl>, 504 <dbl>, 509 <dbl>, 514 <dbl>, 519 <dbl>,
## #   524 <dbl>, 529 <dbl>, 534 <dbl>, 539 <dbl>, 544 <dbl>, 549 <dbl>,
## #   554 <dbl>, 559 <dbl>, 564 <dbl>, 569 <dbl>, 574 <dbl>, 579 <dbl>,
## #   584 <dbl>, 589 <dbl>, 594 <dbl>, 599 <dbl>, 604 <dbl>, 609 <dbl>,
## #   614 <dbl>, 619 <dbl>, 624 <dbl>, 629 <dbl>, 634 <dbl>, 639 <dbl>,
## #   644 <dbl>, 649 <dbl>, 654 <dbl>, 659 <dbl>, 664 <dbl>, 669 <dbl>,
## #   674 <dbl>, 679 <dbl>, 684 <dbl>, 689 <dbl>, 694 <dbl>, 699 <dbl>,
## #   704 <dbl>, 709 <dbl>, 714 <dbl>, 719 <dbl>, 724 <dbl>, 729 <dbl>,
## #   734 <dbl>, 739 <dbl>, 744 <dbl>, 749 <dbl>, ...
```

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

```
##  [1] "Affiliation"       "Boron"             "Calcium"
##  [4] "Carbon"            "Carotenoids_area"  "Carotenoids_mass"
##  [7] "Cellulose"         "Chlorophylls_area" "Chlorophylls_mass"
## [10] "Copper"            "EWT"               "Fiber"
## [13] "Flavonoids"        "LMA"               "Lignin"
## [16] "Magnesium"         "Manganese"         "NSC"
## [19] "Nitrogen"          "PI"                "Phenolics"
## [22] "Phosphorus"        "Plot_ID"           "Potassium"
## [25] "Project"           "SLA"               "Sample_Year"
## [28] "Starch"            "Sugar"             "Sulfur"
## [31] "Water"             "d13C"              "d15N"
## [34] "384"               "389"               "394"
## [37] "399"               "404"               "409"
## [40] "414"
```

Create full plsr dataset

```
# identify the trait data and other metadata
sample_info <- dat_raw[,names(dat_raw) %notin% seq(300,2600,1)]
```

```r
head(sample_info)
```

```
## # A tibble: 6 x 33
##   Affiliation   Boron Calcium Carbon Carotenoids_area Carotenoids_mass Cellulose
##   <chr>         <dbl>   <dbl>  <dbl>            <dbl>            <dbl>     <dbl>
## 1 University ~ 0.0420   24.2    463.             9.19             1.18      221.
## 2 University ~ 0.0361    6.90   558.            10.8              1.17      183.
## 3 University ~ 0.0407   16.7    532.            12.2              1.52      133.
## 4 University ~ 0.0461   13.9    461.             9.16             1.50      220.
## 5 University ~ 0.0401   13.7    510.            11.0              1.53      101.
## 6 University ~ 0.0456   14.5    557.             8.90             1.24      214.
## # ... with 26 more variables: Chlorophylls_area <dbl>, Chlorophylls_mass <dbl>,
## #   Copper <dbl>, EWT <dbl>, Fiber <dbl>, Flavonoids <dbl>, LMA <dbl>,
## #   Lignin <dbl>, Magnesium <dbl>, Manganese <dbl>, NSC <dbl>, Nitrogen <dbl>,
## #   PI <chr>, Phenolics <dbl>, Phosphorus <dbl>, Plot_ID <chr>,
## #   Potassium <dbl>, Project <chr>, SLA <dbl>, Sample_Year <dbl>, Starch <dbl>,
## #   Sugar <dbl>, Sulfur <dbl>, Water <dbl>, d13C <dbl>, d15N <dbl>
```

```r
# spectra matrix
Spectra <- as.matrix(dat_raw[,names(dat_raw) %notin% names(sample_info)])

# set the desired spectra wavelength range to include
Start.wave <- 500
End.wave <- 2400
wv <- seq(Start.wave,End.wave,1)
final_spec <- Spectra[,round(as.numeric(colnames(Spectra))) %in% wv]
colnames(final_spec) <- c(paste0("Wave_",colnames(final_spec)))

## Drop bad spectra data - for canopy-scale reflectance, often the "water band" wavelengths
## are too noisy to use for trait estimation.  Its possible to remove these wavelengths
## prior to model fitting. Its best to first identify which wavelengths to drop
## before attempting PLSR, as these ranges may need to be considered on a case-by-case
## basis or generalized for multiple datasets
dropwaves <- c(1350:1440, 1826:1946)
final_spec <- final_spec[,colnames(final_spec) %notin% paste0("Wave_",dropwaves)]
wv <- as.numeric(gsub(pattern = "Wave_",replacement = "", x = colnames(final_spec)))

## Drop bad spectra data - for canopy-scale reflectance, often the "water band" wavelengths
## are too noisy to use for trait estimation.  Its possible to remove these wavelengths
## prior to model fitting. Its best to first identify which wavelengths to drop
## before attempting PLSR, as these ranges may need to be considered on a case-by-case
## basis or generalized for multiple datasets
dropwaves <- c(1350:1440, 1826:1946)
final_spec <- final_spec[,colnames(final_spec) %notin% paste0("Wave_",dropwaves)]
wv <- as.numeric(gsub(pattern = "Wave_",replacement = "", x = colnames(final_spec)))

# assemble example dataset
sample_info2 <- sample_info %>%
  select(Plot_ID,Sample_Year,SLA,Nitrogen)
site_plot <- data.frame(matrix(unlist(strsplit(sample_info2$Plot_ID, "_")),
                               ncol=2, byrow=TRUE))
colnames(site_plot) <- c("Plot_Num","SampleID")
sample_info3 <- data.frame(site_plot,sample_info2)
```

```r
plsr_data <- data.frame(sample_info3,final_spec*0.01)
rm(sample_info,sample_info2,sample_info3,Spectra, site_plot)
```

```r
# Example data cleaning.  End user needs to do what's appropriate for their
# data. This may be an iterative process.
# Keep only complete rows of inVar and spec data before fitting
#
plsr_data <- plsr_data %>%  # remove erroneously high values, or "bad spectra"
  filter(Nitrogen<50) %>%
  filter(Wave_859<80) %>%
  filter(Wave_859>15)
plsr_data <- plsr_data[complete.cases(plsr_data[,names(plsr_data) %in%
                                                   c(inVar,paste0("Wave_",wv))]),]
```

Example data cleaning.

Create cal/val datasets

```r
## Make a stratified random sampling in the strata USDA_Species_Code and Domain

method <- "base" #base/dplyr
# base R - a bit slow
# dplyr - much faster
split_data <- spectratrait::create_data_split(dataset=plsr_data, approach=method, split_seed=2356326,
                                              prop=0.8, group_variables="Plot_Num")
```

```
## D02   Cal: 80.4597701149425%

## D03   Cal: 80.327868852459%

## D05   Cal: 80%

## D06   Cal: 79.7297297297297%

## D07   Cal: 79.2452830188679%

## D08   Cal: 79.8165137614679%

## D09   Cal: 79.6296296296296%
```

```r
names(split_data)
```

```
## [1] "cal_data" "val_data"
```

```r
cal.plsr.data <- split_data$cal_data
head(cal.plsr.data)[1:8]
```

```
##   Plot_Num SampleID  Plot_ID Sample_Year      SLA Nitrogen  Wave_504  Wave_509
## 2      D02     0002 D02_0002        2017 10.77861 27.70598 1.2909576 1.4075910
## 3      D02     0003 D02_0003        2017 12.46154 34.63999 1.2976806 1.4257559
## 5      D02     0005 D02_0005        2017 17.27620 26.64623 1.7735714 1.9423405
## 6      D02     0006 D02_0006        2017 12.92806 20.69437 1.7786337 1.9621929
## 7      D02     0007 D02_0007        2017 10.21521 28.87526 1.7981043 1.9359032
## 8      D02     0008 D02_0008        2017 20.87397 33.63137 0.8780127 0.9454703
```

```r
val.plsr.data <- split_data$val_data
head(val.plsr.data)[1:8]
```

```
##    Plot_Num SampleID  Plot_ID Sample_Year      SLA Nitrogen Wave_504 Wave_509
## 1       D02     0001 D02_0001        2017 13.66366 31.18030 1.467240 1.654816
## 4       D02     0004 D02_0004        2017 16.63205 34.54034 1.551933 1.764580
## 16      D02     0016 D02_0016        2017 14.44765 22.87740 2.198174 2.403996
## 18      D02     0019 D02_0019        2017 14.47103 17.73126 1.961911 2.175771
## 19      D02     0020 D02_0020        2017 18.98522 21.32929 1.546430 1.873175
## 20      D02     0021 D02_0021        2017 12.12731 29.50256 1.936263 2.065204
```

```r
rm(split_data)

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

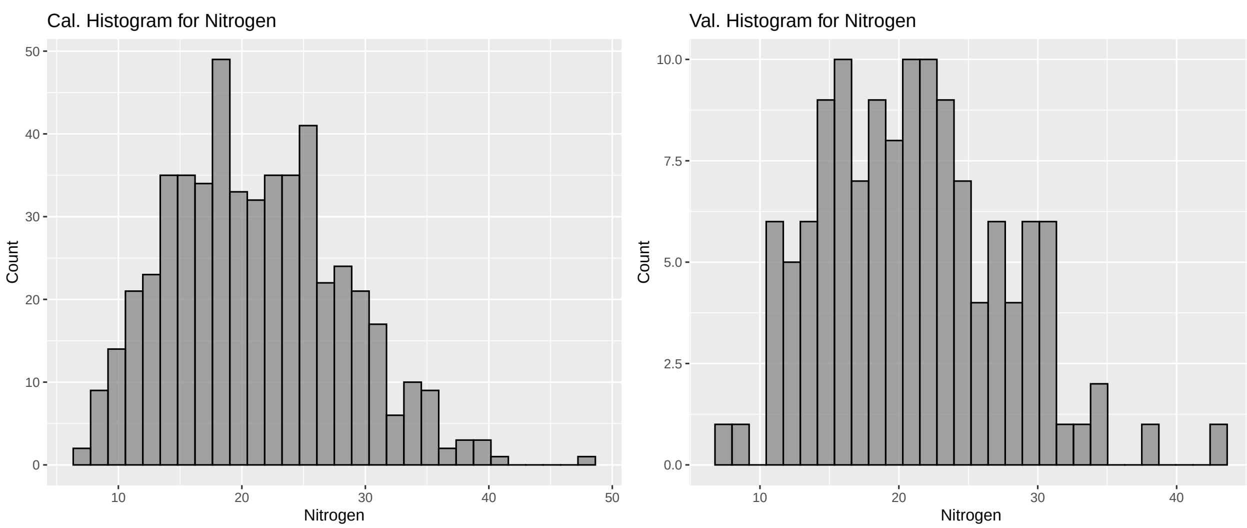
```
## [1] "Cal observations: 517"
```

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

```
## [1] "Val observations: 130"
```

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

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```r
ggsave(filename = file.path(outdir,paste0(inVar,"_Cal_Val_Histograms.png")), plot = histograms,
       device="png", width = 30,
       height = 12, units = "cm",
       dpi = 300)
```

```r
# output cal/val data
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)
```

Create calibration and validation PLSR datasets

```r
cal_spec <- as.matrix(cal.plsr.data[, which(names(cal.plsr.data) %in% paste0("Wave_",wv))])
cal.plsr.data <- data.frame(cal.plsr.data[, which(names(cal.plsr.data) %notin% paste0("Wave_",wv))],
                            Spectra=I(cal_spec))
head(cal.plsr.data)[1:5]
```

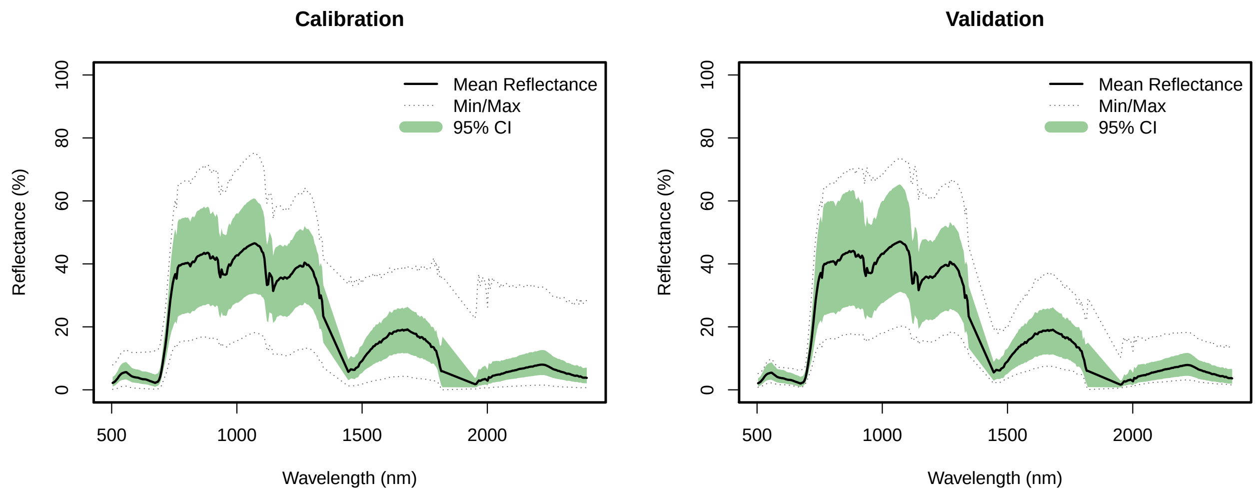
```
##   Plot_Num SampleID  Plot_ID Sample_Year      SLA
## 2      D02     0002 D02_0002        2017 10.77861
## 3      D02     0003 D02_0003        2017 12.46154
## 5      D02     0005 D02_0005        2017 17.27620
## 6      D02     0006 D02_0006        2017 12.92806
## 7      D02     0007 D02_0007        2017 10.21521
## 8      D02     0008 D02_0008        2017 20.87397
```

```r
val_spec <- as.matrix(val.plsr.data[, which(names(val.plsr.data) %in% paste0("Wave_",wv))])
val.plsr.data <- data.frame(val.plsr.data[, which(names(val.plsr.data) %notin% paste0("Wave_",wv))],
                            Spectra=I(val_spec))
head(val.plsr.data)[1:5]
```

```
##    Plot_Num SampleID  Plot_ID Sample_Year      SLA
## 1       D02     0001 D02_0001        2017 13.66366
## 4       D02     0004 D02_0004        2017 16.63205
## 16      D02     0016 D02_0016        2017 14.44765
## 18      D02     0019 D02_0019        2017 14.47103
## 19      D02     0020 D02_0020        2017 18.98522
## 20      D02     0021 D02_0021        2017 12.12731
```

plot cal and val spectra

```r
par(mfrow=c(1,2)) # B, L, T, R
spectratrait::f.plot.spec(Z=cal.plsr.data$Spectra,wv=wv,plot_label="Calibration")
spectratrait::f.plot.spec(Z=val.plsr.data$Spectra,wv=wv,plot_label="Validation")
```




```
dev.copy(png,file.path(outdir,paste0(inVar,'_Cal_Val_Spectra.png')),
         height=2500,width=4900, res=340)
```

```
## quartz_off_screen
##                  3
```

```
dev.off();
```

```
## pdf
##   2
```

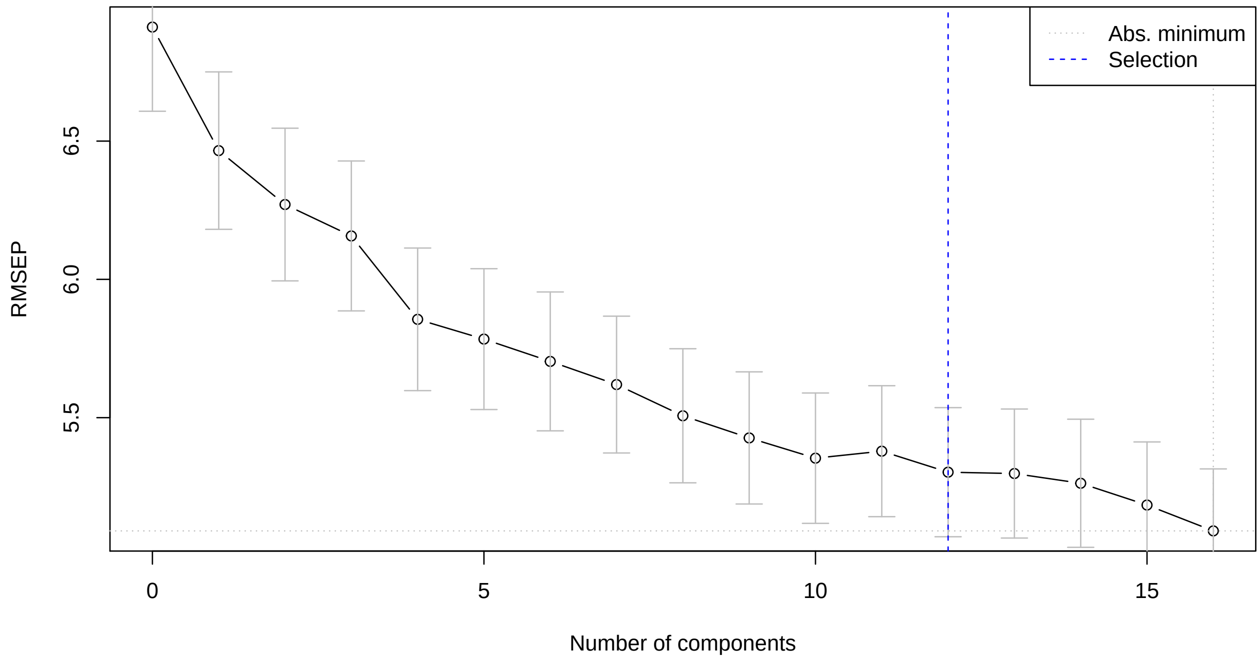
```
par(mfrow=c(1,1))
```

Use permutation to determine optimal number of components

```
if(grepl("Windows", sessionInfo()$running)){
  pls.options(parallel = NULL)
} else {
  pls.options(parallel = parallel::detectCores()-1)
}

method <- "pls" #pls, firstPlateau, firstMin
random_seed <- 1245565
seg <- 50
maxComps <- 16
iterations <- 80
prop <- 0.70
if (method=="pls") {
  # pls package approach - faster but estimates more components....
  nComps <- spectratrait::find_optimal_components(dataset=cal.plsr.data, method=method,
                                                  maxComps=maxComps, seg=seg,
                                                  random_seed=random_seed)
  print(paste0("*** Optimal number of components: ", nComps))
} else {
  nComps <- spectratrait::find_optimal_components(dataset=cal.plsr.data, method=method,
                                                  maxComps=maxComps, iterations=iterations,
                                                  seg=seg, prop=prop,
                                                  random_seed=random_seed)
```

```
}
```

```
## [1] "*** Running PLS permutation test ***"
```




```
## [1] "*** Optimal number of components: 12"
```

```
dev.copy(png,file.path(outdir,paste0(paste0(inVar,"_PLSR_Component_Selection.png"))),
         height=2800, width=3400,  res=340)
```

```
## quartz_off_screen
##                 3
```

```
dev.off();
```

```
## pdf
##   2
```

Fit final model

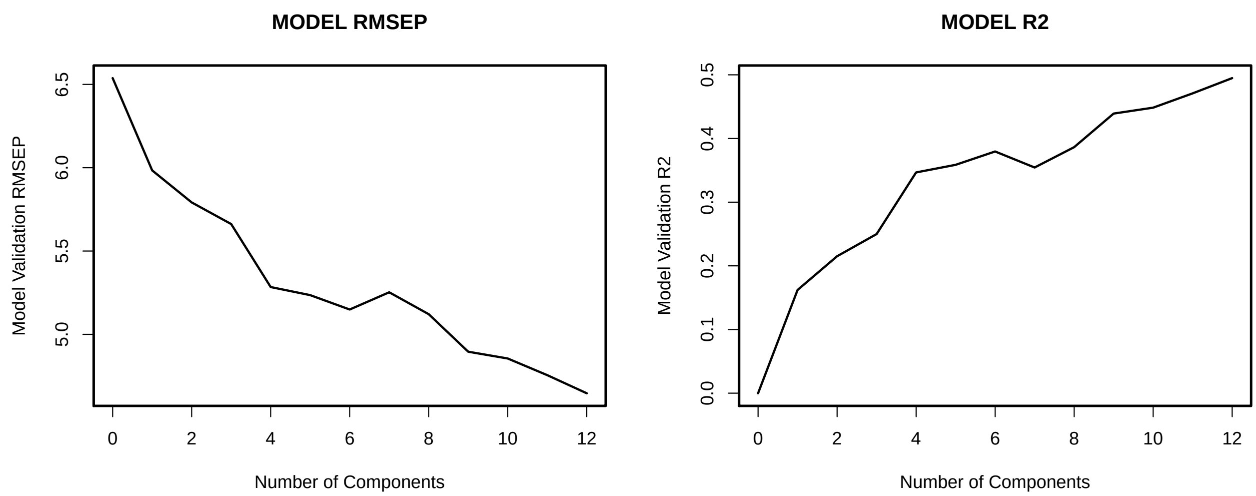
```
plsr.out <- plsr(as.formula(paste(inVar,"~","Spectra")),scale=FALSE,ncomp=nComps,validation="LOO",
                 trace=FALSE,data=cal.plsr.data)
fit <- plsr.out$fitted.values[,1,nComps]
pls.options(parallel = NULL)

# External validation fit stats
par(mfrow=c(1,2)) # B, L, T, R
pls::RMSEP(plsr.out, newdata = val.plsr.data)
```

```
## (Intercept)      1 comps      2 comps      3 comps      4 comps      5 comps
##       6.538        5.984        5.792        5.662        5.284        5.235
##     6 comps      7 comps      8 comps      9 comps     10 comps     11 comps
##       5.149        5.252        5.121        4.896        4.855        4.755
##    12 comps
##       4.646
```

```
plot(pls::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
##  -0.0001616    0.1621284    0.2150431    0.2498762    0.3467097    0.3586424
##     6 comps      7 comps      8 comps      9 comps     10 comps     11 comps
##   0.3796062    0.3544358    0.3863604    0.4391471    0.4484252    0.4708911
##    12 comps
##   0.4948347
```

```
plot(pls::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)
```




```
dev.copy(png,file.path(outdir,paste0(paste0(inVar,"_Validation_RMSEP_R2_by_Component.png"))),
         height=2800, width=4800,  res=340)
```

```
## quartz_off_screen
##                 3
```

```
dev.off();
```

```
## pdf
##   2
```

```
par(opar)
```

PLSR fit observed vs. predicted plot data

```
#calibration
cal.plsr.output <- data.frame(cal.plsr.data[, which(names(cal.plsr.data) %notin% "Spectra")],
                              PLSR_Predicted=fit,
                              PLSR_CV_Predicted=as.vector(plsr.out$validation$pred[,,nComps]))
cal.plsr.output <- cal.plsr.output %>%
  mutate(PLSR_CV_Residuals = PLSR_CV_Predicted-get(inVar))
head(cal.plsr.output)
```

```
##   Plot_Num SampleID  Plot_ID Sample_Year      SLA Nitrogen CalVal
## 2      D02     0002 D02_0002        2017 10.77861 27.70598    Cal
## 3      D02     0003 D02_0003        2017 12.46154 34.63999    Cal
## 5      D02     0005 D02_0005        2017 17.27620 26.64623    Cal
## 6      D02     0006 D02_0006        2017 12.92806 20.69437    Cal
## 7      D02     0007 D02_0007        2017 10.21521 28.87526    Cal
## 8      D02     0008 D02_0008        2017 20.87397 33.63137    Cal
##   PLSR_Predicted PLSR_CV_Predicted PLSR_CV_Residuals
## 2       24.65561          24.59452        -3.1114612
## 3       27.85223          27.64033        -6.9996606
## 5       29.36467          29.54595         2.8997194
## 6       21.66448          21.68116         0.9867955
## 7       23.04393          22.78554        -6.0897138
## 8       25.56637          25.29798        -8.3333884
```

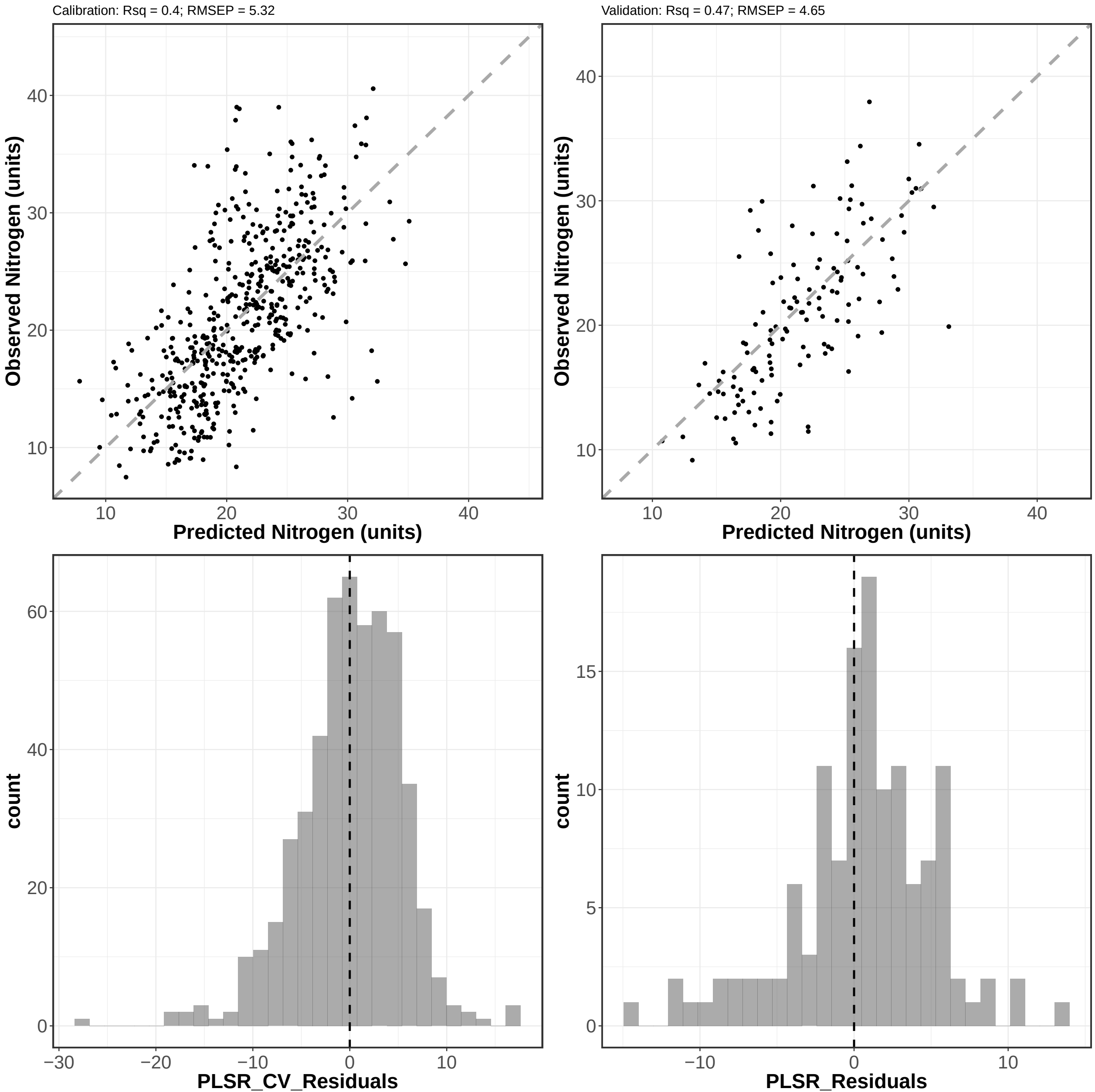
```
cal.R2 <- round(pls::R2(plsr.out)[[1]][nComps],2)
cal.RMSEP <- round(sqrt(mean(cal.plsr.output$PLSR_CV_Residuals^2)),2)

val.plsr.output <- data.frame(val.plsr.data[, which(names(val.plsr.data) %notin% "Spectra")],
                              PLSR_Predicted=as.vector(predict(plsr.out,
                                                                 newdata = val.plsr.data,
                                                                 ncomp=nComps, type="response")[,,1]))
val.plsr.output <- val.plsr.output %>%
  mutate(PLSR_Residuals = PLSR_Predicted-get(inVar))
head(val.plsr.output)
```

```
##    Plot_Num SampleID  Plot_ID Sample_Year      SLA Nitrogen CalVal
## 1       D02     0001 D02_0001        2017 13.66366 31.18030    Val
## 4       D02     0004 D02_0004        2017 16.63205 34.54034    Val
## 16      D02     0016 D02_0016        2017 14.44765 22.87740    Val
## 18      D02     0019 D02_0019        2017 14.47103 17.73126    Val
## 19      D02     0020 D02_0020        2017 18.98522 21.32929    Val
## 20      D02     0021 D02_0021        2017 12.12731 29.50256    Val
##    PLSR_Predicted PLSR_Residuals
## 1        22.55166      -8.628643
## 4        30.79494      -3.745399
## 16       29.14446       6.267060
## 18       23.47518       5.743923
## 19       23.00736       1.678070
## 20       31.93483       2.432274
```

```
val.R2 <- round(pls::R2(plsr.out,newdata=val.plsr.data)[[1]][nComps],2)
val.RMSEP <- round(sqrt(mean(val.plsr.output$PLSR_Residuals^2)),2)

rng_quant <- quantile(cal.plsr.output[,inVar], probs = c(0.001, 0.999))
cal_scatter_plot <- ggplot(cal.plsr.output, aes(x=PLSR_CV_Predicted, y=get(inVar))) +
  theme_bw() + geom_point() + geom_abline(intercept = 0, slope = 1, color="dark grey",
                                          linetype="dashed", size=1.5) + xlim(rng_quant[1],
                                                                                rng_quant[2]) +
  ylim(rng_quant[1], rng_quant[2]) +
  labs(x=paste0("Predicted ", paste(inVar), " (units)"),
       y=paste0("Observed ", paste(inVar), " (units)"),
       title=paste0("Calibration: ", paste0("Rsq = ", cal.R2), "; ", paste0("RMSEP = ",
                                                                              cal.RMSEP))) +
  theme(axis.text=element_text(size=18), legend.position="none",
```

```r
        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.plsr.output, aes(x=PLSR_CV_Residuals)) +
  geom_histogram(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))

rng_quant <- quantile(val.plsr.output[,inVar], probs = c(0.001, 0.999))
val_scatter_plot <- ggplot(val.plsr.output, aes(x=PLSR_Predicted, y=get(inVar))) +
  theme_bw() + geom_point() + geom_abline(intercept = 0, slope = 1, color="dark grey",
                                          linetype="dashed", size=1.5) + xlim(rng_quant[1],
                                                                              rng_quant[2]) +
  ylim(rng_quant[1], rng_quant[2]) +
  labs(x=paste0("Predicted ", paste(inVar), " (units)"),
       y=paste0("Observed ", paste(inVar), " (units)"),
       title=paste0("Validation: ", paste0("Rsq = ", val.R2), "; ", paste0("RMSEP = ",
                                                                           val.RMSEP))) +
  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.plsr.output, aes(x=PLSR_Residuals)) +
  geom_histogram(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
scatterplots <- grid.arrange(cal_scatter_plot, val_scatter_plot, cal_resid_histogram,
                             val_resid_histogram, nrow=2,ncol=2)
```

```
## Warning: Removed 5 rows containing missing values (geom_point).

## Warning: Removed 2 rows containing missing values (geom_point).

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

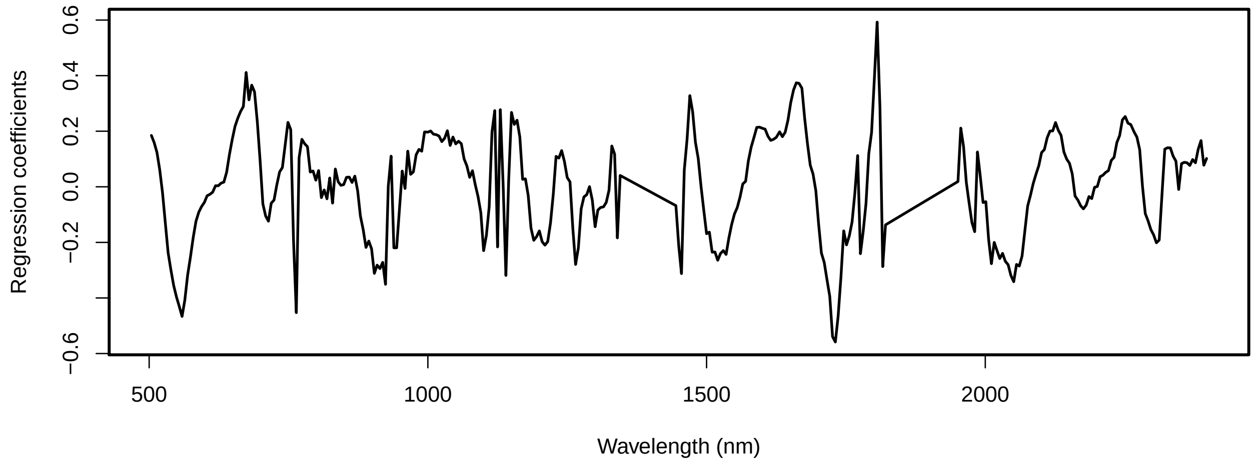


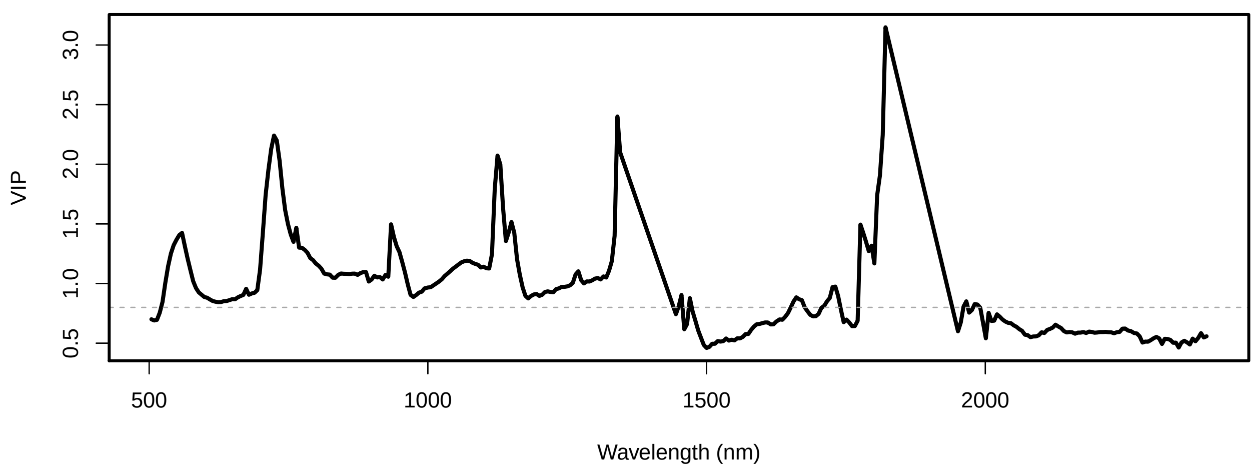

```
ggsave(filename = file.path(outdir,paste0(inVar,"_Cal_Val_Scatterplots.png")),
       plot = scatterplots, device="png", width = 32, height = 30, units = "cm",
       dpi = 300)
```

Generate Coefficient and VIP plots

```
vips <- spectratrait::VIP(plsr.out)[nComps,]

par(mfrow=c(2,1))
plot(plsr.out$coefficients[,,nComps], x=wv,xlab="Wavelength (nm)",
     ylab="Regression coefficients",lwd=2,type='l')
box(lwd=2.2)
plot(wv, vips, xlab="Wavelength (nm)",ylab="VIP",cex=0.01)
```

```
lines(wv, vips, lwd=3)
abline(h=0.8, lty=2, col="dark grey")
box(lwd=2.2)
```






```
dev.copy(png, file.path(outdir, paste0(inVar,'_Coefficient_VIP_plot.png')),
         height=3100, width=4100, res=340)
```

```
## quartz_off_screen
##                 3
```

```
dev.off();
```

```
## pdf
##   2
```

```
par(opar)
```

Bootstrap validation

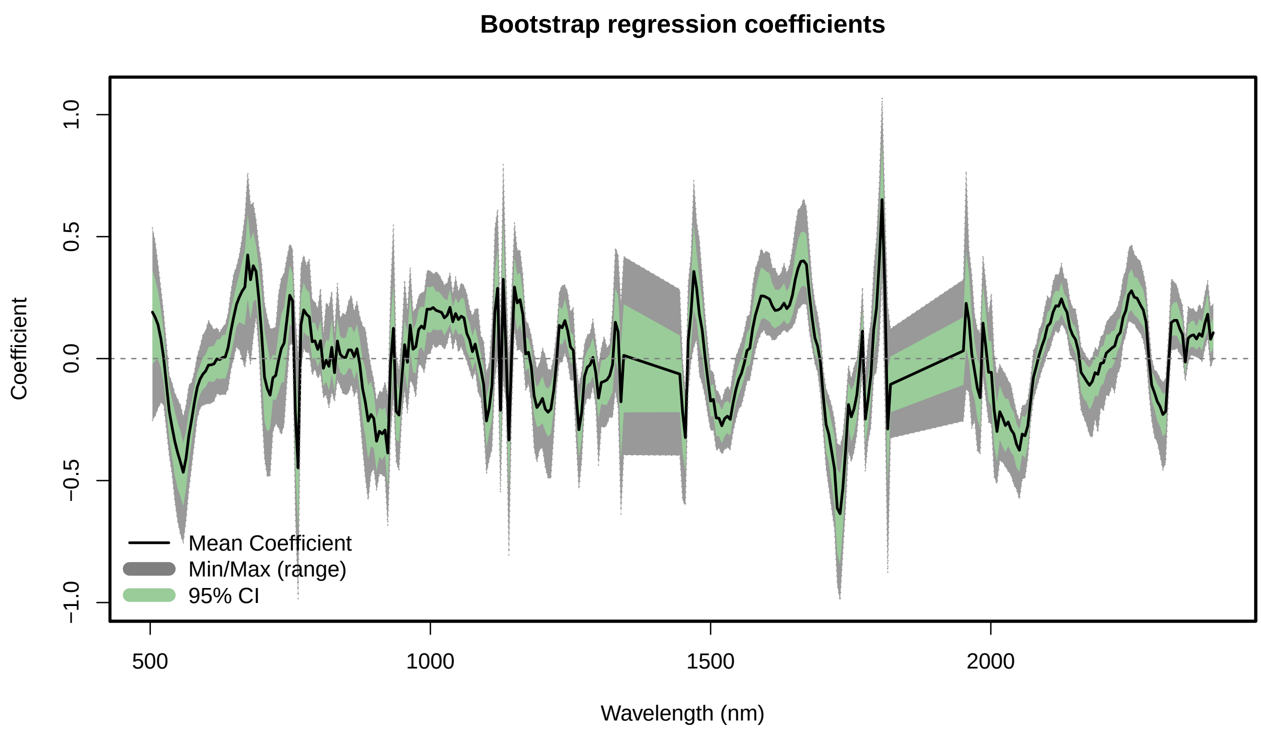
```
## [1] "*** Running permutation test.  Please hang tight, this can take awhile ***"
## [1] "Options:"
```

```
## [1] "Max Components: 12 Iterations: 500 Data Proportion (percent): 70"
## [1] "*** Providing PRESS and coefficient array output ***"
```

```
##    Plot_Num SampleID  Plot_ID Sample_Year      SLA Nitrogen CalVal
## 1       D02     0001 D02_0001        2017 13.66366 31.18030    Val
## 4       D02     0004 D02_0004        2017 16.63205 34.54034    Val
## 16      D02     0016 D02_0016        2017 14.44765 22.87740    Val
## 18      D02     0019 D02_0019        2017 14.47103 17.73126    Val
## 19      D02     0020 D02_0020        2017 18.98522 21.32929    Val
## 20      D02     0021 D02_0021        2017 12.12731 29.50256    Val
##    PLSR_Predicted PLSR_Residuals      LCI      UCI      LPI      UPI
## 1        22.55166      -8.628643 21.75139 23.67919 13.44246 31.66086
## 4        30.79494      -3.745399 29.24737 32.37867 21.60577 39.98412
## 16       29.14446       6.267060 27.57462 30.82609 19.93270 38.35621
## 18       23.47518       5.743923 21.73808 24.49326 14.31158 32.63878
## 19       23.00736       1.678070 20.70321 24.57934 13.73687 32.27785
## 20       31.93483       2.432274 30.75996 34.32739 22.69357 41.17610
```

Jackknife coefficient plot

```
spectratrait::f.plot.coef(Z = t(bootstrap_coef), wv = wv,
                          plot_label="Bootstrap regression coefficients",
                          position = 'bottomleft')
abline(h=0,lty=2,col="grey50")
box(lwd=2.2)
```




```
dev.copy(png,file.path(outdir,paste0(inVar,'_Bootstrap_Regression_Coefficients.png')),
         height=2100, width=3800, res=340)
```

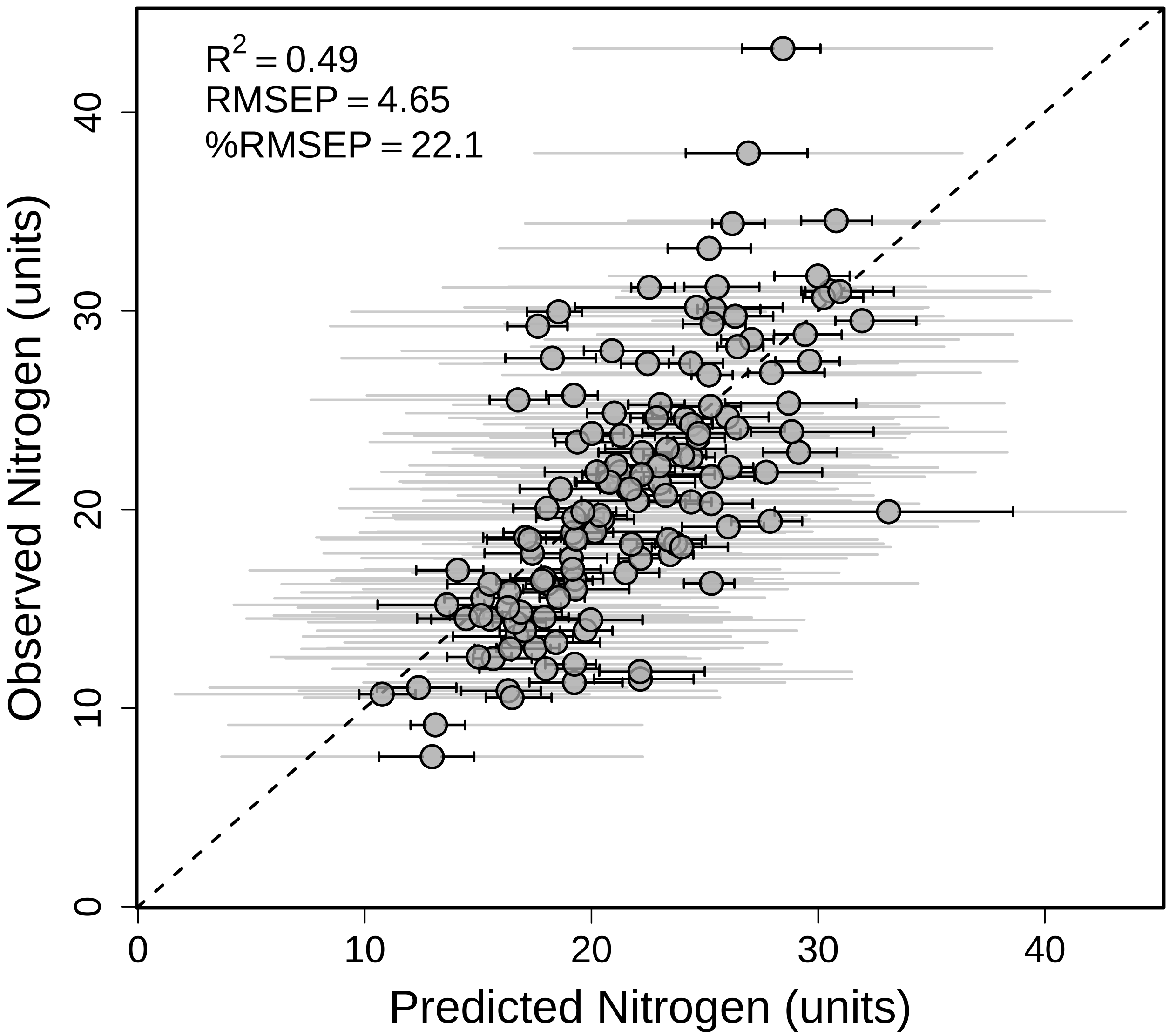
```
## quartz_off_screen
##                 3
```

```
dev.off();
```

```
## pdf
##   2
```

Bootstrap validation plot

```
RMSEP <- sqrt(mean(val.plsr.output$PLSR_Residuals^2))
pecr_RMSEP <- RMSEP/mean(val.plsr.output[,inVar])*100
r2 <- round(pls::R2(plsr.out, newdata = val.plsr.data)$val[nComps+1],2)
expr <- vector("expression", 3)
expr[[1]] <- bquote(R^2==.(r2))
expr[[2]] <- bquote(RMSEP==.(round(RMSEP,2)))
expr[[3]] <- bquote("%RMSEP"==.(round(pecr_RMSEP,2)))
rng_vals <- c(min(val.plsr.output$LPI), max(val.plsr.output$UPI))
par(mfrow=c(1,1), mar=c(4.2,5.3,1,0.4), oma=c(0, 0.1, 0, 0.2))
plotrix::plotCI(val.plsr.output$PLSR_Predicted,val.plsr.output[,inVar],
                li=val.plsr.output$LPI, ui=val.plsr.output$UPI, gap=0.009,sfrac=0.000,
                lwd=1.6, xlim=c(rng_vals[1], rng_vals[2]), ylim=c(rng_vals[1], rng_vals[2]),
                err="x", pch=21, col="black", pt.bg=scales::alpha("grey70",0.7), scol="grey80",
                cex=2, xlab=paste0("Predicted ", paste(inVar), " (units)"),
                ylab=paste0("Observed ", paste(inVar), " (units)"),
                cex.axis=1.5,cex.lab=1.8)
abline(0,1,lty=2,lw=2)
plotrix::plotCI(val.plsr.output$PLSR_Predicted,val.plsr.output[,inVar],
                li=val.plsr.output$LCI, ui=val.plsr.output$UCI, gap=0.009,sfrac=0.004,
                lwd=1.6, xlim=c(rng_vals[1], rng_vals[2]), ylim=c(rng_vals[1], rng_vals[2]),
                err="x", pch=21, col="black", pt.bg=scales::alpha("grey70",0.7), scol="black",
                cex=2, xlab=paste0("Predicted ", paste(inVar), " (units)"),
                ylab=paste0("Observed ", paste(inVar), " (units)"),
                cex.axis=1.5,cex.lab=1.8, add=T)
legend("topleft", legend=expr, bty="n", cex=1.5)
box(lwd=2.2)
```




```
dev.copy(png,file.path(outdir,paste0(inVar,"_PLSR_Validation_Scatterplot.png")),
         height=2800, width=3200,  res=340)
```

```
## quartz_off_screen
##                 3
```

```
dev.off();
```

```
## pdf
##   2
```

Output bootstrap results

```
out.jk.coefs <- data.frame(Iteration=seq(1,length(bootstrap_intercept),1),
                           Intercept=bootstrap_intercept,t(bootstrap_coef))
names(out.jk.coefs) <- c("Iteration","Intercept",paste0("Wave_",wv))
head(out.jk.coefs)[1:6]
```

```
##   Iteration Intercept   Wave_504   Wave_509   Wave_514    Wave_519
```

```
## 1         1  13.57171  0.2253380  0.1886856  0.1539993  0.09577521
## 2         2  15.24466  0.1921689  0.1596680  0.1200761  0.05273115
## 3         3  14.36148  0.2138642  0.1821139  0.1216748  0.06134136
## 4         4  12.28467  0.2444603  0.2089635  0.1558502  0.10461395
## 5         5  12.94807 -0.1358811 -0.1290176 -0.1109839 -0.09476558
## 6         6  14.56747  0.2983242  0.2627539  0.2313171  0.16354535
```

```r
write.csv(out.jk.coefs,file=file.path(outdir,paste0(inVar,'_Bootstrap_PLSR_Coefficients.csv')),
          row.names=FALSE)
```

Create core PLSR outputs

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

```
## [1] "Output directory:  /Users/sserbin/Data/GitHub/PLSR_for_plant_trait_prediction/vignettes"
```

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

# Validation data
write.csv(val.plsr.output,file=file.path(outdir,
                                          paste0(inVar,'_Validation_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

```r
print("**** PLSR output files: ")
```

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

```r
print(list.files(outdir)[grep(pattern = inVar,
                               list.files(outdir))])
```

```
##  [1] "Nitrogen_Bootstrap_PLSR_Coefficients.csv"
##  [2] "Nitrogen_Bootstrap_Regression_Coefficients.png"
##  [3] "Nitrogen_Cal_PLSR_Dataset.csv"
##  [4] "Nitrogen_Cal_Val_Histograms.png"
##  [5] "Nitrogen_Cal_Val_Scatterplots.png"
##  [6] "Nitrogen_Cal_Val_Spectra.png"
##  [7] "Nitrogen_Coefficient_VIP_plot.png"
##  [8] "Nitrogen_Observed_PLSR_CV_Pred_12comp.csv"
```

```
##  [9] "Nitrogen_PLSR_Coefficients_12comp.csv"
## [10] "Nitrogen_PLSR_Component_Selection.png"
## [11] "Nitrogen_PLSR_Validation_Scatterplot.png"
## [12] "Nitrogen_PLSR_VIPs_12comp.csv"
## [13] "Nitrogen_Val_PLSR_Dataset.csv"
## [14] "Nitrogen_Validation_PLSR_Pred_12comp.csv"
## [15] "Nitrogen_Validation_RMSEP_R2_by_Component.png"
```