

Spectra-trait PLSR example using leaf-level spectra and leaf nitrogen content (Narea, g/m²) data from 36 species growing in *Rosa rugosa* invaded coastal grassland communities in Belgium

Shawn P. Serbin, Julien Lamour, & Jeremiah Anderson

Overview

This is an R Markdown Notebook to illustrate how to retrieve a dataset from the EcoSIS spectral database, choose the “optimal” number of plsr components, and fit a plsr model for leaf nitrogen content (Narea, g/m²)

Getting Started

Installation

```
## Skipping install of 'spectratrait' from a github remote, the SHA1 (fbcb98da) has not changed since 1
##   Use `force = TRUE` to force installation

##
## 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

```

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

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

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

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

# What is the target variable?
inVar <- "Narea_g_m2"

# What is the source dataset from EcoSIS?
ecosys_id <- "9db4c5a2-7eac-4e1e-8859-009233648e89"

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

```

Set working directory (scratch space)

```
## [1] "/private/var/folders/xp/h3k9vf3n2jx181ts786_yjrn9c2gjQ/T/Rtmp73bCe2"
```

Grab data from EcoSIS

```

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

## [1] "Output directory: /Users/sserbin/Data/GitHub/PLSR_for_plant_trait_prediction/vignettes"
dat_raw <- spectratrait::get_ecosis_data(ecosis_id = ecosys_id)

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

## Downloading data...

##
## -- Column specification -----
## cols(
##   .default = col_double(),
##   `Latin Species` = col_character(),
##   ids = col_character(),
##   `plot code` = col_character(),
##   `species code` = col_character()
## )
## i Use `spec()` for the full column specifications.
## Download complete!

```

```
head(dat_raw)
```

```
## # A tibble: 6 x 2,164
##   `Cw/EWT (cm3/cm~` `Latin Species` `Leaf area (mm2~` `Leaf calcium c~
##           <dbl> <chr>           <dbl>           <dbl>
## 1      0.00887 Arrhenatherum ~      696.           0.0291
## 2      0.00824 Bromus sterilis      447.           0.0230
## 3      0.0280  Jacobaea vulga~     2418.          0.0950
## 4      0.0106  Rubus caesius      5719.          0.0700
## 5      0.00851 Arrhenatherum ~      671.           0.0286
## 6      0.0153  Crepis capilla~     1401.          0.0470
## # ... with 2,160 more variables: `Leaf magnesium content per leaf area
## #   (mg/mm2)` <dbl>, `Leaf mass per area (g/cm2)` <dbl>, `Leaf nitrogen content
## #   per leaf area (mg/mm2)` <dbl>, `Leaf phosphorus content per leaf area
## #   (mg/mm2)` <dbl>, `Leaf potassium content per leaf area (mg/mm2)` <dbl>,
## #   `Plant height vegetative (cm)` <dbl>, ids <chr>, `plot code` <chr>,
## #   `species code` <chr>, `350` <dbl>, `351` <dbl>, `352` <dbl>, `353` <dbl>,
## #   `354` <dbl>, `355` <dbl>, `356` <dbl>, `357` <dbl>, `358` <dbl>,
## #   `359` <dbl>, `360` <dbl>, `361` <dbl>, `362` <dbl>, `363` <dbl>,
## #   `364` <dbl>, `365` <dbl>, `366` <dbl>, `367` <dbl>, `368` <dbl>,
## #   `369` <dbl>, `370` <dbl>, `371` <dbl>, `372` <dbl>, `373` <dbl>,
## #   `374` <dbl>, `375` <dbl>, `376` <dbl>, `377` <dbl>, `378` <dbl>,
## #   `379` <dbl>, `380` <dbl>, `381` <dbl>, `382` <dbl>, `383` <dbl>,
## #   `384` <dbl>, `385` <dbl>, `386` <dbl>, `387` <dbl>, `388` <dbl>,
## #   `389` <dbl>, `390` <dbl>, `391` <dbl>, `392` <dbl>, `393` <dbl>,
## #   `394` <dbl>, `395` <dbl>, `396` <dbl>, `397` <dbl>, `398` <dbl>,
## #   `399` <dbl>, `400` <dbl>, `401` <dbl>, `402` <dbl>, `403` <dbl>,
## #   `404` <dbl>, `405` <dbl>, `406` <dbl>, `407` <dbl>, `408` <dbl>,
## #   `409` <dbl>, `410` <dbl>, `411` <dbl>, `412` <dbl>, `413` <dbl>,
## #   `414` <dbl>, `415` <dbl>, `416` <dbl>, `417` <dbl>, `418` <dbl>,
## #   `419` <dbl>, `420` <dbl>, `421` <dbl>, `422` <dbl>, `423` <dbl>,
## #   `424` <dbl>, `425` <dbl>, `426` <dbl>, `427` <dbl>, `428` <dbl>,
## #   `429` <dbl>, `430` <dbl>, `431` <dbl>, `432` <dbl>, `433` <dbl>,
## #   `434` <dbl>, `435` <dbl>, `436` <dbl>, `437` <dbl>, `438` <dbl>,
## #   `439` <dbl>, `440` <dbl>, ...
```

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

```
## [1] "Cw/EWT (cm3/cm2)"
## [2] "Latin Species"
## [3] "Leaf area (mm2)"
## [4] "Leaf calcium content per leaf area (mg/mm2)"
## [5] "Leaf magnesium content per leaf area (mg/mm2)"
## [6] "Leaf mass per area (g/cm2)"
## [7] "Leaf nitrogen content per leaf area (mg/mm2)"
## [8] "Leaf phosphorus content per leaf area (mg/mm2)"
## [9] "Leaf potassium content per leaf area (mg/mm2)"
## [10] "Plant height vegetative (cm)"
## [11] "ids"
## [12] "plot code"
## [13] "species code"
## [14] "350"
## [15] "351"
## [16] "352"
```

```
## [17] "353"
## [18] "354"
## [19] "355"
## [20] "356"
## [21] "357"
## [22] "358"
## [23] "359"
## [24] "360"
## [25] "361"
## [26] "362"
## [27] "363"
## [28] "364"
## [29] "365"
## [30] "366"
## [31] "367"
## [32] "368"
## [33] "369"
## [34] "370"
## [35] "371"
## [36] "372"
## [37] "373"
## [38] "374"
## [39] "375"
## [40] "376"
```

Create full pls dataset

Create pls dataset

```
Start.wave <- 500
End.wave <- 2400
wv <- seq(Start.wave,End.wave,1)
Spectra <- as.matrix(dat_raw[,names(dat_raw) %in% wv])
colnames(Spectra) <- c(paste0("Wave_",wv))
sample_info <- dat_raw[,names(dat_raw) %notin% seq(350,2500,1)]
head(sample_info)
```

```
## # A tibble: 6 x 13
##   `Cw/EWT (cm3/cm~` `Latin Species` `Leaf area (mm2~` `Leaf calcium c~
##   <dbl> <chr> <dbl> <dbl>
## 1 0.00887 Arrhenatherum ~ 696. 0.0291
## 2 0.00824 Bromus sterilis 447. 0.0230
## 3 0.0280 Jacobaea vulga~ 2418. 0.0950
## 4 0.0106 Rubus caesius 5719. 0.0700
## 5 0.00851 Arrhenatherum ~ 671. 0.0286
## 6 0.0153 Crepis capilla~ 1401. 0.0470
## # ... with 9 more variables: `Leaf magnesium content per leaf area
## # (mg/mm2)` <dbl>, `Leaf mass per area (g/cm2)` <dbl>, `Leaf nitrogen content
## # per leaf area (mg/mm2)` <dbl>, `Leaf phosphorus content per leaf area
## # (mg/mm2)` <dbl>, `Leaf potassium content per leaf area (mg/mm2)` <dbl>,
## # `Plant height vegetative (cm)` <dbl>, ids <chr>, `plot code` <chr>,
## # `species code` <chr>
```

```
sample_info2 <- sample_info %>%
  select(Plant_Species=`Latin Species`,Species_Code=`species code`,Plot=`plot code`,
         Narea_mg_mm2=`Leaf nitrogen content per leaf area (mg/mm2)`)
sample_info2 <- sample_info2 %>%
  # mutate(Narea_g_m2=Narea_mg_mm2*(0.001/1e-6)) # based on orig units should be this but conversion wro
  mutate(Narea_g_m2=Narea_mg_mm2*100) # this assumes orig units were g/mm2 or mg/cm2
head(sample_info2)
```

```
## # A tibble: 6 x 5
##   Plant_Species      Species_Code Plot  Narea_mg_mm2 Narea_g_m2
##   <chr>             <chr>      <chr>      <dbl>      <dbl>
## 1 Arrhenatherum elatius Arrela      DC1         0.0126      1.26
## 2 Bromus sterilis      Broste      DC1         0.00682     0.682
## 3 Jacobaea vulgaris    Jacvul      DC1         0.0102     1.02
## 4 Rubus caesius        Rubcae      DC1         0.0121     1.21
## 5 Arrhenatherum elatius Arrela      DC2         0.0117     1.17
## 6 Crepis capillaris    Creves      DC2         0.00877     0.877
```

```
plsr_data <- data.frame(sample_info2,Spectra)
rm(sample_info,sample_info2,Spectra)
```

```
#### 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[complete.cases(plsr_data[,names(plsr_data) %in%
                                     c(inVar,paste0("Wave_",wv))]),]
```

Example data cleaning.

Create cal/val datasets

```
### Create cal/val datasets
## Make a stratified random sampling in the strata USDA_Species_Code and Domain

method <- "dplyr" #base/dplyr
# base R - a bit slow
# dplyr - much faster
split_data <- spectratrait::create_data_split(dataset=plsr_data, approach=method, split_seed=1245565,
                                              prop=0.8, group_variables="Species_Code")
names(split_data)

## [1] "cal_data" "val_data"

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

```
##      Plant_Species Species_Code Plot Narea_mg_mm2 Narea_g_m2 Wave_500
## 1 Ammophila arenaria   Ammare   ZC3   0.03240495   3.240495 0.130885
## 2 Ammophila arenaria   Ammare   MC2   0.02806279   2.806279 0.135785
## 3 Ammophila arenaria   Ammare   ZC1   0.02041612   2.041612 0.147665
## 4 Ammophila arenaria   Ammare   MC1   0.02426549   2.426549 0.142765
## 5 Ammophila arenaria   Ammare   WC3   0.02807281   2.807281 0.151750
```

```
## 6 Ammophila arenaria      Ammare  WR3    0.02286678    2.286678 0.150850
##   Wave_501 Wave_502
## 1  0.13175 0.132750
## 2  0.13685 0.138150
## 3  0.14910 0.150330
## 4  0.14390 0.145200
## 5  0.15275 0.154150
## 6  0.15185 0.152815
```

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

```
##           Plant_Species Species_Code Plot Narea_mg_mm2 Narea_g_m2   Wave_500
## 184  Jacobaea vulgaris      Jacvul  WC2  0.008756996  0.8756996 0.06736887
## 185  Potentilla reptans      Potrep  WC2  0.010313464  1.0313464 0.07125000
## 186    Rubus caesius        Rubcae  WC2  0.007968454  0.7968454 0.05993560
## 187    Urtica dioica        Urtdio  WC2  0.012737560  1.2737560 0.06508300
## 188  Ammophila arenaria      Ammare  WC3  0.028072806  2.8072806 0.15175000
## 189  Jacobaea vulgaris      Jacvul  WC3  0.010251687  1.0251687 0.06805547
##           Wave_501   Wave_502
## 184 0.06870667 0.07014220
## 185 0.07235000 0.07368350
## 186 0.06162000 0.06352233
## 187 0.06625000 0.06758350
## 188 0.15275000 0.15415000
## 189 0.06938000 0.07093553
```

```
rm(split_data)
```

```
# Datasets:
```

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

```
## [1] "Cal observations: 183"
```

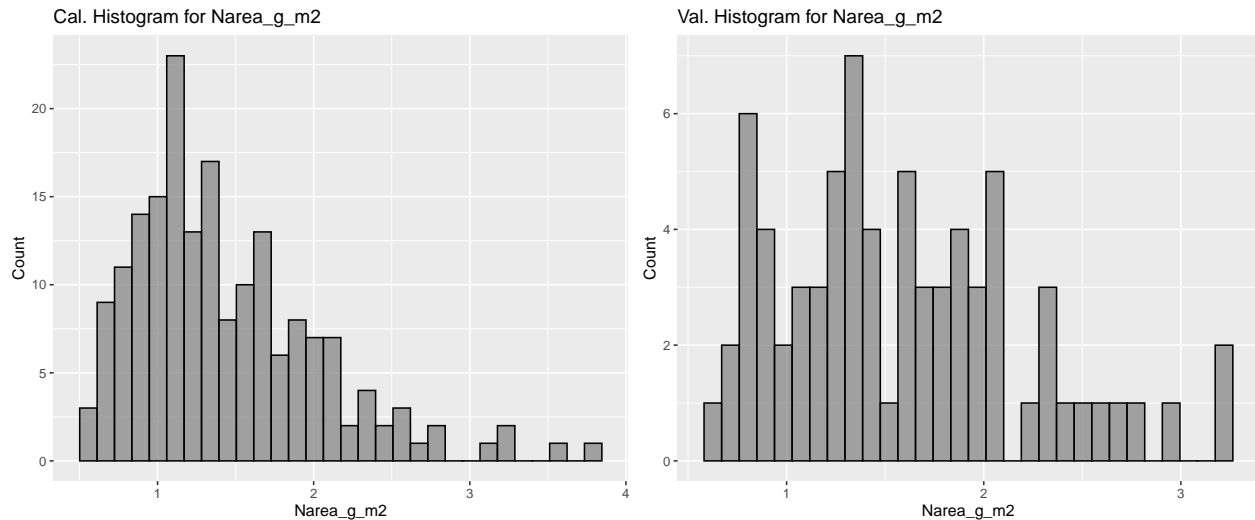
```
print(paste("Val observations: ",dim(val.plsr.data)[1],sep=""))
```

```
## [1] "Val observations: 73"
```

```
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`.
```



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

```
### Format PLSR data for model fitting
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]

##      Plant_Species Species_Code Plot Narea_mg_mm2 Narea_g_m2
## 1 Ammophila arenaria      Ammare ZC3  0.03240495  3.240495
## 2 Ammophila arenaria      Ammare MC2  0.02806279  2.806279
## 3 Ammophila arenaria      Ammare ZC1  0.02041612  2.041612
## 4 Ammophila arenaria      Ammare MC1  0.02426549  2.426549
## 5 Ammophila arenaria      Ammare WC3  0.02807281  2.807281
## 6 Ammophila arenaria      Ammare WR3  0.02286678  2.286678

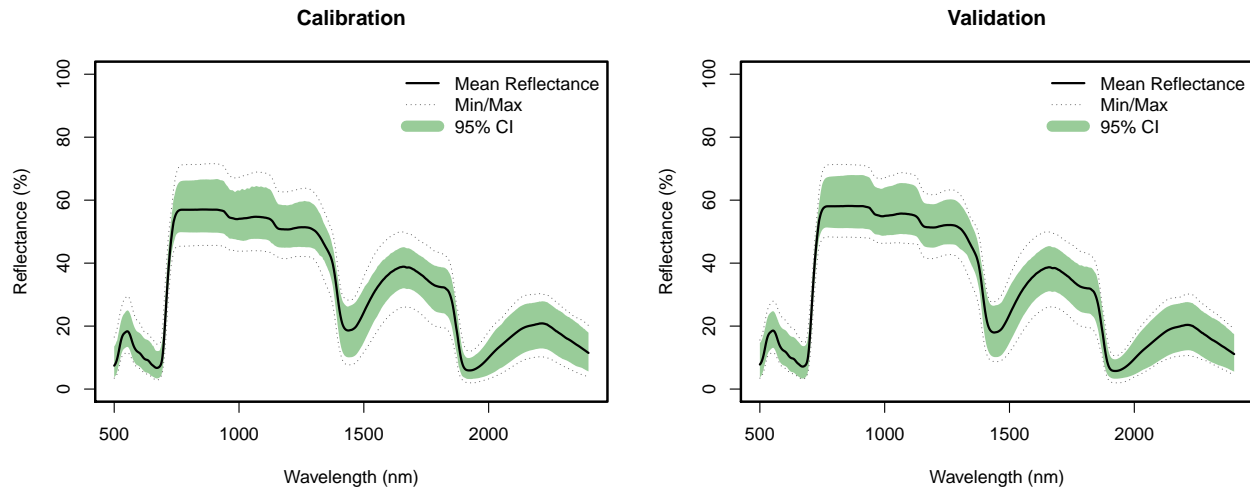
val_spec <- as.matrix(val.plsr.data[, which(names(val.plsr.data) %in% paste0("Wave_",wv))])
val.plsr.data <- data.frame(val.plsr.data[, which(names(val.plsr.data) %notin% paste0("Wave_",wv))],
                           Spectra=I(val_spec))
head(val.plsr.data)[1:5]

##      Plant_Species Species_Code Plot Narea_mg_mm2 Narea_g_m2
## 184 Jacobaea vulgaris      Jacvul WC2  0.008756996  0.8756996
## 185 Potentilla reptans      Potrep WC2  0.010313464  1.0313464
## 186 Rubus caesius          Rubcae WC2  0.007968454  0.7968454
## 187 Urtica dioica          Urtdio WC2  0.012737560  1.2737560
## 188 Ammophila arenaria      Ammare WC3  0.028072806  2.8072806
```

```
## 189  Jacobaea vulgaris      Jacvul  WC3  0.010251687  1.0251687
```

plot cal and val spectra

```
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

```
### Use permutation to determine the 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") {
```



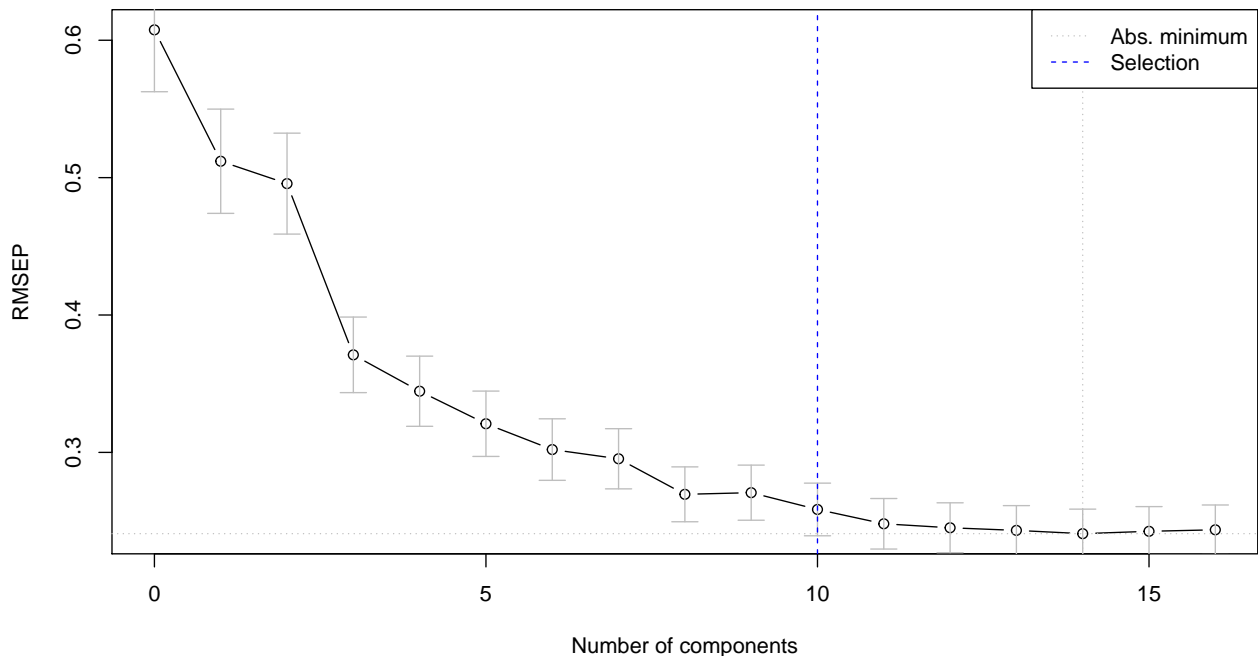
```

# 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: 10"
```

```

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="L00",
               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
##      0.6346      0.5045      0.4645      0.3415      0.3296      0.3037
##      6 comps      7 comps      8 comps      9 comps     10 comps
##      0.2703      0.2659      0.2524      0.2450      0.2452

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)

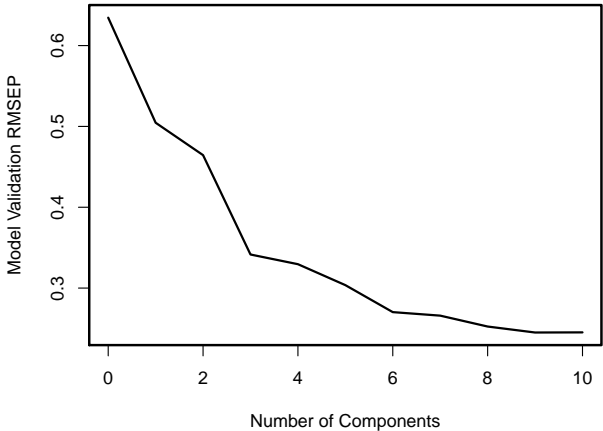
pls::R2(plsr.out, newdata = val.plsr.data)

## (Intercept)      1 comps      2 comps      3 comps      4 comps      5 comps
##     -0.05977      0.33000      0.43217      0.69298      0.71415      0.75732
##      6 comps      7 comps      8 comps      9 comps     10 comps
##      0.80776      0.81389      0.83228      0.84198      0.84176

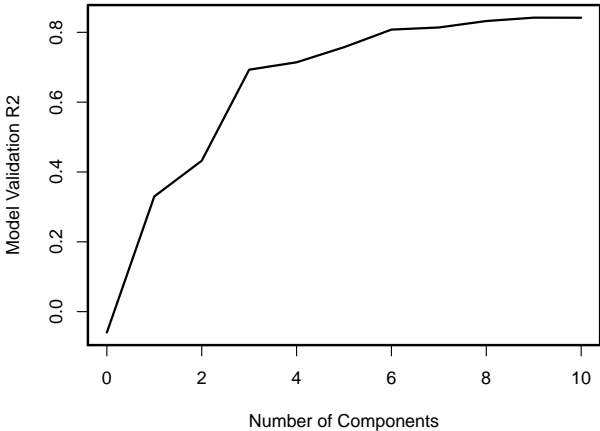
plot(pls::R2(plsr.out,estimate=c("test"),newdata = val.plsr.data), main="MODEL R2",
     xlab="Number of Components",ylab="Model Validation R2",lty=1,col="black",cex=1.5,lwd=2)
box(lwd=2.2)

```

MODEL RMSEP



MODEL R2



```

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)
```

	Plant_Species	Species_Code	Plot	Narea_mg_mm2	Narea_g_m2	PLSR_Predicted
## 1	Ammophila arenaria	Ammare	ZC3	0.03240495	3.240495	2.672029
## 2	Ammophila arenaria	Ammare	MC2	0.02806279	2.806279	2.651863
## 3	Ammophila arenaria	Ammare	ZC1	0.02041612	2.041612	2.178056
## 4	Ammophila arenaria	Ammare	MC1	0.02426549	2.426549	2.412013
## 5	Ammophila arenaria	Ammare	WC3	0.02807281	2.807281	2.452711
## 6	Ammophila arenaria	Ammare	WR3	0.02286678	2.286678	2.792340

```
##      PLSR_CV_Predicted PLSR_CV_Residuals
## 1          2.598245      -0.642250440
## 2          2.652066      -0.154212969
## 3          2.200588       0.158975634
## 4          2.435784       0.009234491
## 5          2.384049      -0.423231444
## 6          2.943186       0.656508493

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)
```

	Plant_Species	Species_Code	Plot	Narea_mg_mm2	Narea_g_m2	PLSR_Predicted
## 184	Jacobaea vulgaris	Jacvul	WC2	0.008756996	0.8756996	0.9462916
## 185	Potentilla reptans	Potrep	WC2	0.010313464	1.0313464	1.5386676
## 186	Rubus caesius	Rubcae	WC2	0.007968454	0.7968454	0.8790482
## 187	Urtica dioica	Urtdio	WC2	0.012737560	1.2737560	1.1241560
## 188	Ammophila arenaria	Ammare	WC3	0.028072806	2.8072806	2.4527108
## 189	Jacobaea vulgaris	Jacvul	WC3	0.010251687	1.0251687	1.1553688

```
##      PLSR_Residuals
## 184      0.07059201
## 185      0.50732119
## 186      0.08220284
## 187     -0.14959995
## 188     -0.35456980
## 189      0.13020008

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("Rsqr = ", cal.R2), "; ", paste0("RMSEP = ",
                                                                                   cal.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))

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("Rsqr = ", 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))

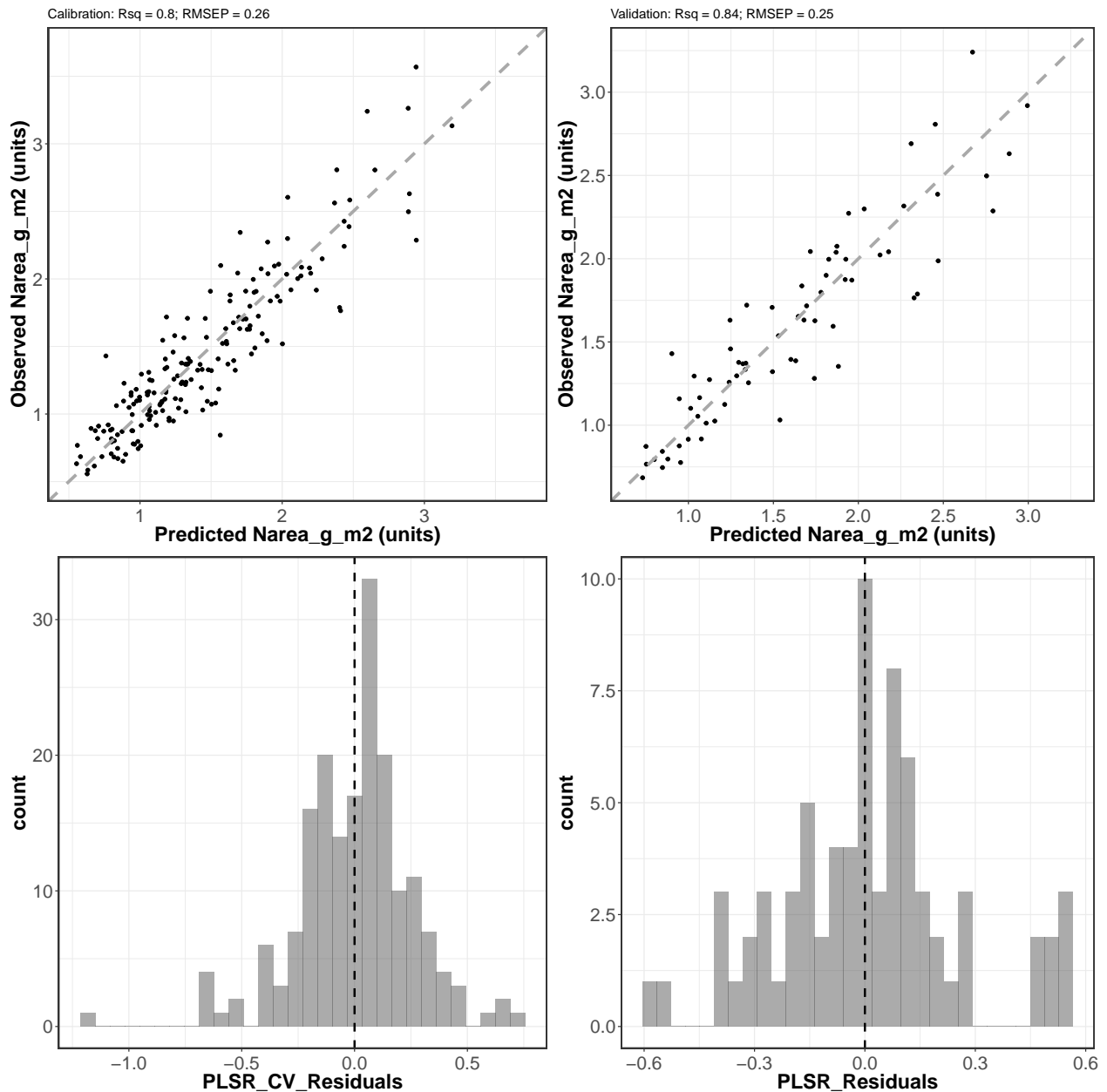
# plot cal/val side-by-side
scatterplots <- grid.arrange(cal_scatter_plot, val_scatter_plot, cal_resid_histogram,
                             val_resid_histogram, nrow=2, ncol=2)

```

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

```
## Warning: Removed 3 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)
```

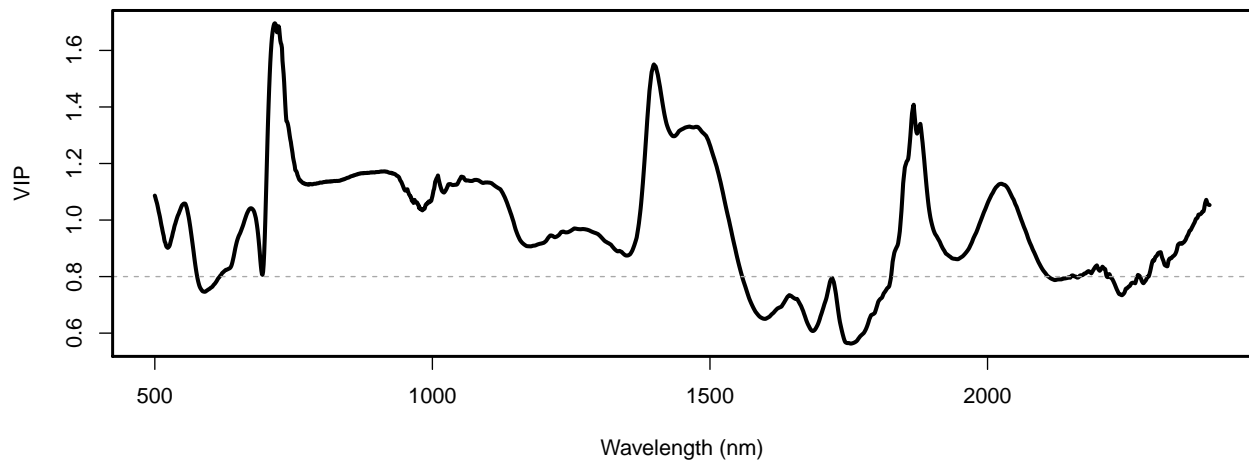
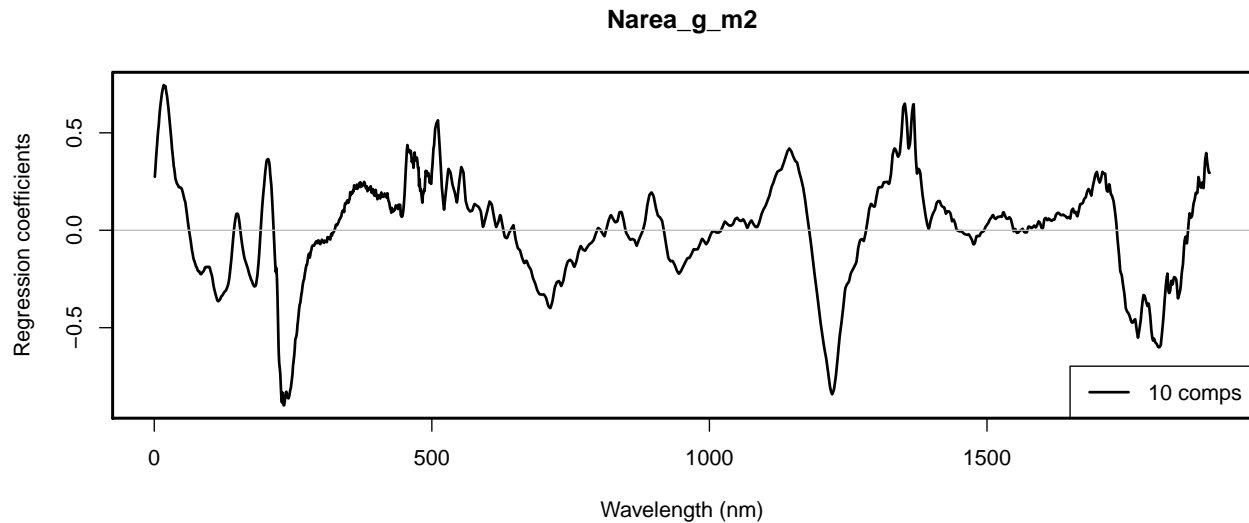
Generate Coefficient and VIP plots

```
vips <- spectratrait::VIP(plsr.out)[nComps,]
par(mfrow=c(2,1))
```

```

plot(plsr.out, plotype = "coef", xlab="Wavelength (nm)",
     ylab="Regression coefficients", legendpos = "bottomright",
     ncomp=nComps, lwd=2)
box(lwd=2.2)
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")
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

```

Bootstrap validation

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

### PLSR bootstrap permutation uncertainty analysis
iterations <- 500      # how many permutation iterations to run
prop <- 0.70           # fraction of training data to keep for each iteration
pls_permutation <- spectratrait::pls_permutation(dataset=cal.plsr.data, maxComps=nComps,
                                                  iterations=iterations, prop=prop)

## [1] "*** Running permutation test. Please hang tight, this can take awhile ***"
## [1] "Options: 10 500 100 0.7"

## Running interation 1
## Running interation 2
## Running interation 3
## Running interation 4
## Running interation 5
## Running interation 6
## Running interation 7
## Running interation 8
## Running interation 9
## Running interation 10
## Running interation 11
## Running interation 12
## Running interation 13
## Running interation 14
## Running interation 15
## Running interation 16
## Running interation 17
## Running interation 18
## Running interation 19
## Running interation 20
## Running interation 21
## Running interation 22
## Running interation 23
## Running interation 24
## Running interation 25
```

Running iteration 26
Running iteration 27
Running iteration 28
Running iteration 29
Running iteration 30
Running iteration 31
Running iteration 32
Running iteration 33
Running iteration 34
Running iteration 35
Running iteration 36
Running iteration 37
Running iteration 38
Running iteration 39
Running iteration 40
Running iteration 41
Running iteration 42
Running iteration 43
Running iteration 44
Running iteration 45
Running iteration 46
Running iteration 47
Running iteration 48
Running iteration 49
Running iteration 50
Running iteration 51
Running iteration 52
Running iteration 53
Running iteration 54
Running iteration 55
Running iteration 56
Running iteration 57
Running iteration 58
Running iteration 59
Running iteration 60
Running iteration 61

Running interation 62
Running interation 63
Running interation 64
Running interation 65
Running interation 66
Running interation 67
Running interation 68
Running interation 69
Running interation 70
Running interation 71
Running interation 72
Running interation 73
Running interation 74
Running interation 75
Running interation 76
Running interation 77
Running interation 78
Running interation 79
Running interation 80
Running interation 81
Running interation 82
Running interation 83
Running interation 84
Running interation 85
Running interation 86
Running interation 87
Running interation 88
Running interation 89
Running interation 90
Running interation 91
Running interation 92
Running interation 93
Running interation 94
Running interation 95
Running interation 96
Running interation 97

Running interation 98
Running interation 99
Running interation 100
Running interation 101
Running interation 102
Running interation 103
Running interation 104
Running interation 105
Running interation 106
Running interation 107
Running interation 108
Running interation 109
Running interation 110
Running interation 111
Running interation 112
Running interation 113
Running interation 114
Running interation 115
Running interation 116
Running interation 117
Running interation 118
Running interation 119
Running interation 120
Running interation 121
Running interation 122
Running interation 123
Running interation 124
Running interation 125
Running interation 126
Running interation 127
Running interation 128
Running interation 129
Running interation 130
Running interation 131
Running interation 132
Running interation 133

Running interation 134
Running interation 135
Running interation 136
Running interation 137
Running interation 138
Running interation 139
Running interation 140
Running interation 141
Running interation 142
Running interation 143
Running interation 144
Running interation 145
Running interation 146
Running interation 147
Running interation 148
Running interation 149
Running interation 150
Running interation 151
Running interation 152
Running interation 153
Running interation 154
Running interation 155
Running interation 156
Running interation 157
Running interation 158
Running interation 159
Running interation 160
Running interation 161
Running interation 162
Running interation 163
Running interation 164
Running interation 165
Running interation 166
Running interation 167
Running interation 168
Running interation 169

Running interation 170
Running interation 171
Running interation 172
Running interation 173
Running interation 174
Running interation 175
Running interation 176
Running interation 177
Running interation 178
Running interation 179
Running interation 180
Running interation 181
Running interation 182
Running interation 183
Running interation 184
Running interation 185
Running interation 186
Running interation 187
Running interation 188
Running interation 189
Running interation 190
Running interation 191
Running interation 192
Running interation 193
Running interation 194
Running interation 195
Running interation 196
Running interation 197
Running interation 198
Running interation 199
Running interation 200
Running interation 201
Running interation 202
Running interation 203
Running interation 204
Running interation 205

Running interation 206
Running interation 207
Running interation 208
Running interation 209
Running interation 210
Running interation 211
Running interation 212
Running interation 213
Running interation 214
Running interation 215
Running interation 216
Running interation 217
Running interation 218
Running interation 219
Running interation 220
Running interation 221
Running interation 222
Running interation 223
Running interation 224
Running interation 225
Running interation 226
Running interation 227
Running interation 228
Running interation 229
Running interation 230
Running interation 231
Running interation 232
Running interation 233
Running interation 234
Running interation 235
Running interation 236
Running interation 237
Running interation 238
Running interation 239
Running interation 240
Running interation 241

Running interation 242
Running interation 243
Running interation 244
Running interation 245
Running interation 246
Running interation 247
Running interation 248
Running interation 249
Running interation 250
Running interation 251
Running interation 252
Running interation 253
Running interation 254
Running interation 255
Running interation 256
Running interation 257
Running interation 258
Running interation 259
Running interation 260
Running interation 261
Running interation 262
Running interation 263
Running interation 264
Running interation 265
Running interation 266
Running interation 267
Running interation 268
Running interation 269
Running interation 270
Running interation 271
Running interation 272
Running interation 273
Running interation 274
Running interation 275
Running interation 276
Running interation 277

Running interation 278
Running interation 279
Running interation 280
Running interation 281
Running interation 282
Running interation 283
Running interation 284
Running interation 285
Running interation 286
Running interation 287
Running interation 288
Running interation 289
Running interation 290
Running interation 291
Running interation 292
Running interation 293
Running interation 294
Running interation 295
Running interation 296
Running interation 297
Running interation 298
Running interation 299
Running interation 300
Running interation 301
Running interation 302
Running interation 303
Running interation 304
Running interation 305
Running interation 306
Running interation 307
Running interation 308
Running interation 309
Running interation 310
Running interation 311
Running interation 312
Running interation 313

Running interation 314
Running interation 315
Running interation 316
Running interation 317
Running interation 318
Running interation 319
Running interation 320
Running interation 321
Running interation 322
Running interation 323
Running interation 324
Running interation 325
Running interation 326
Running interation 327
Running interation 328
Running interation 329
Running interation 330
Running interation 331
Running interation 332
Running interation 333
Running interation 334
Running interation 335
Running interation 336
Running interation 337
Running interation 338
Running interation 339
Running interation 340
Running interation 341
Running interation 342
Running interation 343
Running interation 344
Running interation 345
Running interation 346
Running interation 347
Running interation 348
Running interation 349

Running iteration 350
Running iteration 351
Running iteration 352
Running iteration 353
Running iteration 354
Running iteration 355
Running iteration 356
Running iteration 357
Running iteration 358
Running iteration 359
Running iteration 360
Running iteration 361
Running iteration 362
Running iteration 363
Running iteration 364
Running iteration 365
Running iteration 366
Running iteration 367
Running iteration 368
Running iteration 369
Running iteration 370
Running iteration 371
Running iteration 372
Running iteration 373
Running iteration 374
Running iteration 375
Running iteration 376
Running iteration 377
Running iteration 378
Running iteration 379
Running iteration 380
Running iteration 381
Running iteration 382
Running iteration 383
Running iteration 384
Running iteration 385

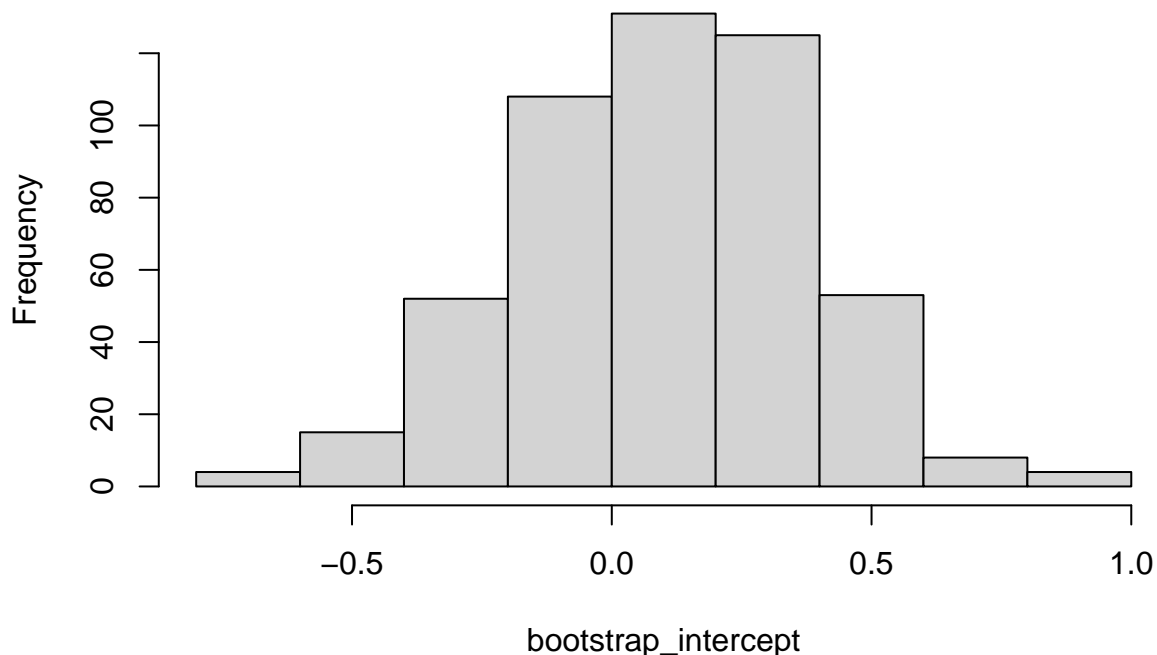
Running interation 386
Running interation 387
Running interation 388
Running interation 389
Running interation 390
Running interation 391
Running interation 392
Running interation 393
Running interation 394
Running interation 395
Running interation 396
Running interation 397
Running interation 398
Running interation 399
Running interation 400
Running interation 401
Running interation 402
Running interation 403
Running interation 404
Running interation 405
Running interation 406
Running interation 407
Running interation 408
Running interation 409
Running interation 410
Running interation 411
Running interation 412
Running interation 413
Running interation 414
Running interation 415
Running interation 416
Running interation 417
Running interation 418
Running interation 419
Running interation 420
Running interation 421

Running interation 422
Running interation 423
Running interation 424
Running interation 425
Running interation 426
Running interation 427
Running interation 428
Running interation 429
Running interation 430
Running interation 431
Running interation 432
Running interation 433
Running interation 434
Running interation 435
Running interation 436
Running interation 437
Running interation 438
Running interation 439
Running interation 440
Running interation 441
Running interation 442
Running interation 443
Running interation 444
Running interation 445
Running interation 446
Running interation 447
Running interation 448
Running interation 449
Running interation 450
Running interation 451
Running interation 452
Running interation 453
Running interation 454
Running interation 455
Running interation 456
Running interation 457

Running interation 458
Running interation 459
Running interation 460
Running interation 461
Running interation 462
Running interation 463
Running interation 464
Running interation 465
Running interation 466
Running interation 467
Running interation 468
Running interation 469
Running interation 470
Running interation 471
Running interation 472
Running interation 473
Running interation 474
Running interation 475
Running interation 476
Running interation 477
Running interation 478
Running interation 479
Running interation 480
Running interation 481
Running interation 482
Running interation 483
Running interation 484
Running interation 485
Running interation 486
Running interation 487
Running interation 488
Running interation 489
Running interation 490
Running interation 491
Running interation 492
Running interation 493

```
## Running iteration 494
## Running iteration 495
## Running iteration 496
## Running iteration 497
## Running iteration 498
## Running iteration 499
## Running iteration 500
bootstrap_intercept <- plsr_permutation$coef_array[1,,nComps]
hist(bootstrap_intercept)
```

Histogram of bootstrap_intercept



```
bootstrap_coef <- plsr_permutation$coef_array[2:length(plsr_permutation$coef_array[,1,nComps]),
,nComps]
rm(plsr_permutation)

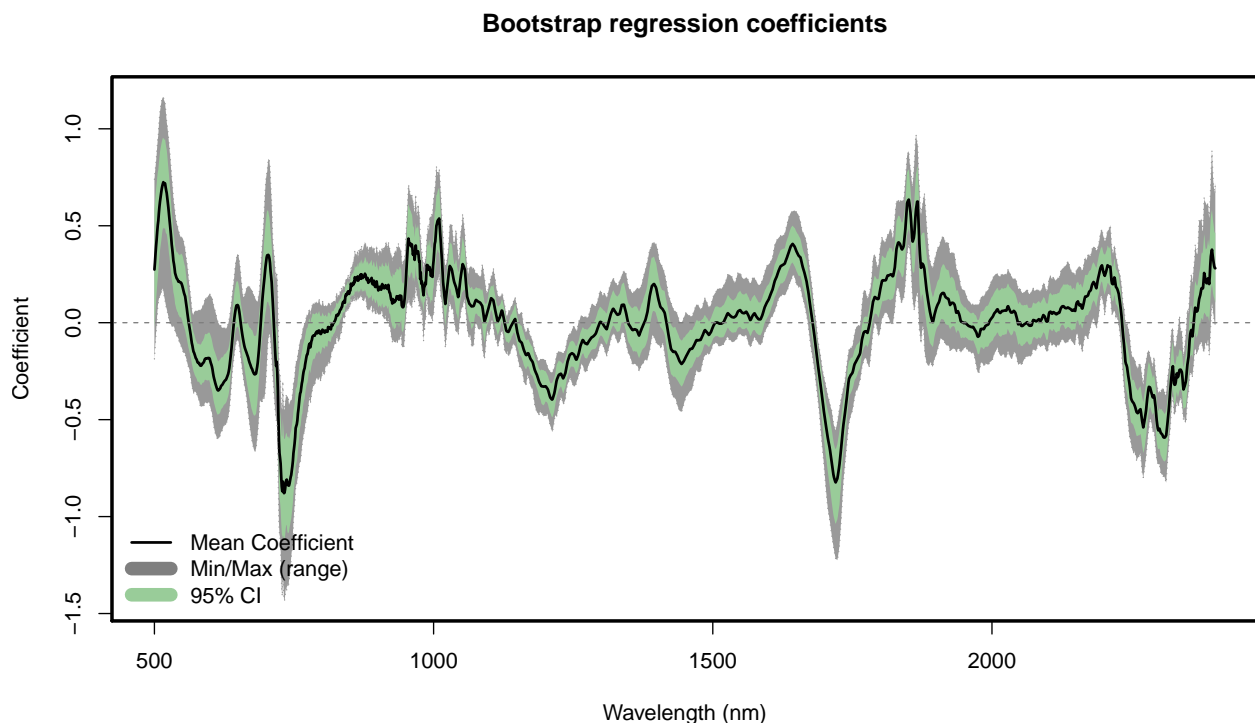
# apply coefficients to left-out validation data
interval <- c(0.025,0.975)
Bootstrap_Pred <- val.plsr.data$Spectra %*% bootstrap_coef +
  matrix(rep(bootstrap_intercept, length(val.plsr.data[,inVar])), byrow=TRUE,
    ncol=length(bootstrap_intercept))
Interval_Conf <- apply(X = Bootstrap_Pred, MARGIN = 1, FUN = quantile,
  probs=c(interval[1], interval[2]))
sd_mean <- apply(X = Bootstrap_Pred, MARGIN = 1, FUN = sd)
sd_res <- sd(val.plsr.output$PLSR_Residuals)
sd_tot <- sqrt(sd_mean^2+sd_res^2)
val.plsr.output$LCI <- Interval_Conf[1,]
val.plsr.output$UCI <- Interval_Conf[2,]
```

```
val.plsr.output$LPI <- val.plsr.output$PLSR_Predicted-1.96*sd_tot
val.plsr.output$UPI <- val.plsr.output$PLSR_Predicted+1.96*sd_tot
head(val.plsr.output)
```

```
##      Plant_Species Species_Code Plot Narea_mg_mm2 Narea_g_m2 PLSR_Predicted
## 184 Jacobaea vulgaris   Jacvul  WC2  0.008756996  0.8756996      0.9462916
## 185 Potentilla reptans   Potrep  WC2  0.010313464  1.0313464      1.5386676
## 186 Rubus caesius       Rubcae  WC2  0.007968454  0.7968454      0.8790482
## 187 Urtica dioica       Urtdio  WC2  0.012737560  1.2737560      1.1241560
## 188 Ammophila arenaria   Ammare  WC3  0.028072806  2.8072806      2.4527108
## 189 Jacobaea vulgaris   Jacvul  WC3  0.010251687  1.0251687      1.1553688
##      PLSR_Residuals      LCI      UCI      LPI      UPI
## 184  0.07059201  0.8915898  1.008806  0.4588988  1.433684
## 185  0.50732119  1.4007173  1.636452  1.0403747  2.036960
## 186  0.08220284  0.6861219  1.160030  0.3405908  1.417506
## 187 -0.14959995  0.9651982  1.245464  0.6198291  1.628483
## 188 -0.35456980  2.1911406  2.619696  1.9245720  2.980850
## 189  0.13020008  1.0735154  1.233082  0.6651011  1.645636
```

Jackknife coefficient plot

```
# Bootstrap regression 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)
```

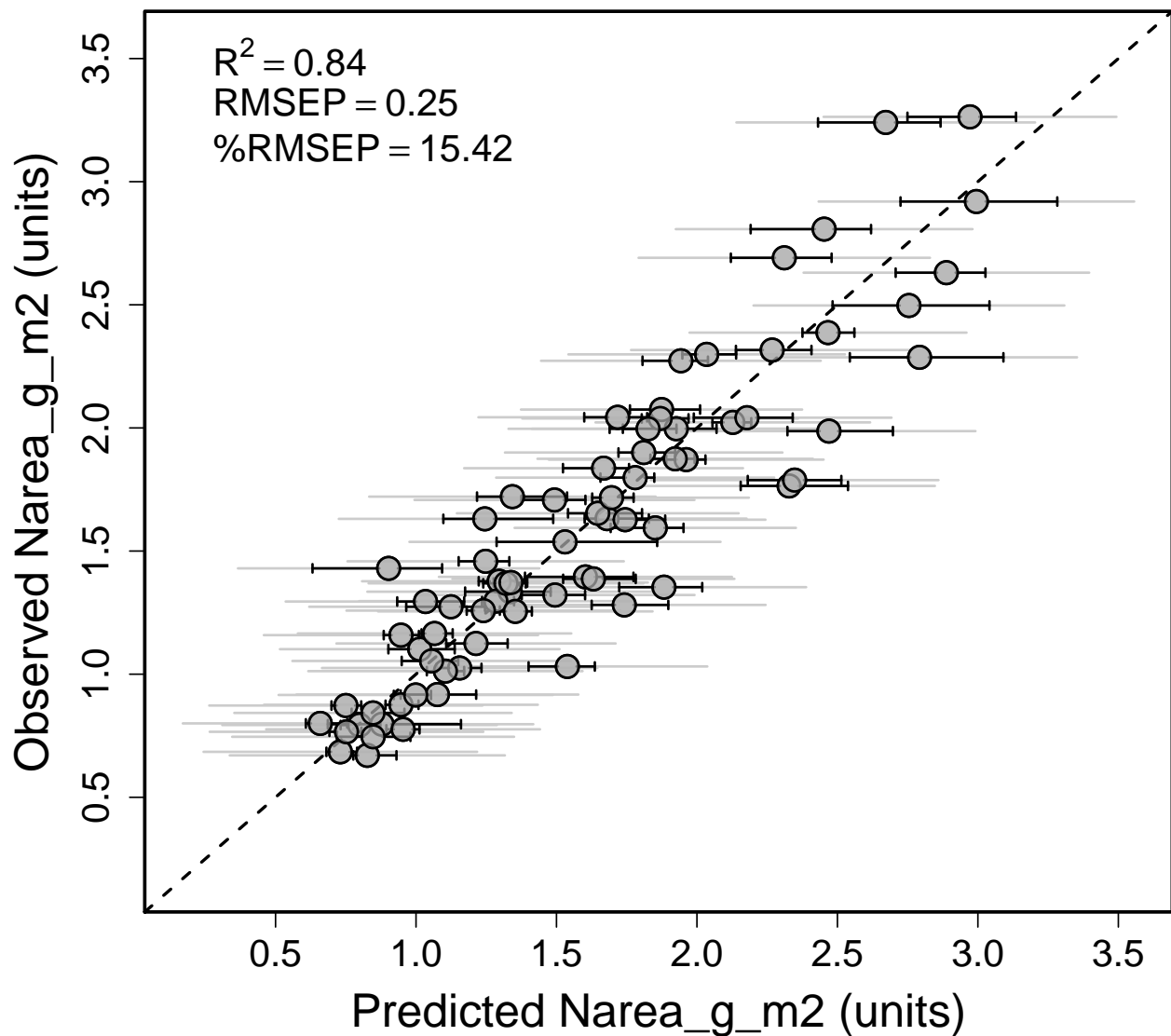
```
## quartz_off_screen
##          3

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$LPI, ui=val.plsr.output$UPI, 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

```
# Bootstrap Coefficients
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_500   Wave_501   Wave_502   Wave_503
## 1           1 -0.13686765 0.29141489 0.3287594 0.3654695 0.3999712
## 2           2 -0.17612080 0.24114488 0.2827001 0.3182992 0.3552505
## 3           3  0.34135463 0.21939317 0.2562451 0.2984578 0.3330333
## 4           4  0.01511507 0.09404839 0.1299058 0.1792805 0.2240698
## 5           5  0.06136605 0.12835311 0.1662322 0.2116938 0.2539193
## 6           6  0.10925409 0.28154095 0.3107509 0.3539714 0.3932620

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

Create core PLSR outputs

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

## [1] "Output directory: /var/folders/xp/h3k9vf3n2jx181ts786_yjrn9c2gjgq/T//Rtmp73bCe2"
# 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

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

## [1] "**** PLSR output files: "
print(list.files(outdir)[grep(pattern = inVar, list.files(outdir))])

## [1] "Narea_g_m2_Bootstrap_PLSR_Coefficients.csv"
## [2] "Narea_g_m2_Bootstrap_Regression_Coefficients.png"
## [3] "Narea_g_m2_Cal_PLSR_Dataset.csv"
```

```
## [4] "Narea_g_m2_Cal_Val_Histograms.png"
## [5] "Narea_g_m2_Cal_Val_Scatterplots.png"
## [6] "Narea_g_m2_Cal_Val_Spectra.png"
## [7] "Narea_g_m2_Coefficient_VIP_plot.png"
## [8] "Narea_g_m2_Observed_PLSR_CV_Pred_10comp.csv"
## [9] "Narea_g_m2_PLSR_Coefficients_10comp.csv"
## [10] "Narea_g_m2_PLSR_Component_Selection.png"
## [11] "Narea_g_m2_PLSR_Validation_Scatterplot.png"
## [12] "Narea_g_m2_PLSR_VIPs_10comp.csv"
## [13] "Narea_g_m2_Val_PLSR_Dataset.csv"
## [14] "Narea_g_m2_Validation_PLSR_Pred_10comp.csv"
## [15] "Narea_g_m2_Validation_RMSEP_R2_by_Component.png"
```