LPW

This RMD serves as a guide/lab journal to document my methods for processing and modelling FT-NIRS otolith spectra from juvenile Pacific cod (**Gadus macrocephalus**) in the Gulf of Alaska. These scans were acquired on a Bruker MPA II with an integrating sphere, 5mm. teflon disk taped over the scanning window and gold transflectance stamp placed over the top of samples. Spectra were collected between 11,500 and 4,000 cm^{-1} with a resolution of 16 cm^{-1} and 64 replicate scans.

LPW Scans refer to specimens collected near Little Port Walter (LPW), a NOAA research station located near the southern tip of Baranof Island in Southeast Alaska. Juvenile cod were collected using beach and purse seines from embayments near LPW on July 27-28, 2020. These fish were transferred into outdoor net pens at LPW to be reared in captivity. Specimens were collected approximately weekly (ADD MORE IN HERE LATER)

Load necessary packages

This code borrowed from **Dr. Esther Goldstein** at NOAA's AFSC in Seattle. Packages used include *dplyr* for easier data manipulation, *ggplot2* for clear looking figures, *opusreader2* to import FT-NIRS spectral data, *prospectr* for FT-NIRS preprocessing (ADD MORE IN HERE)

```
# Install packages not yet installed
packages <- c("dplyr", "tidyr", "EMSC", "purrr", "pls", "devtools", "hyperSpec", "prospectr", "data.tabl
installed_packages <- packages %in% rownames(installed.packages())
if (any(installed_packages == FALSE)) {
  utils::install.packages(pkgs = packages[!installed_packages])
}
invisible(lapply(packages, library, character.only = TRUE)) # load all packages in list

# install packages not on CRAN
if (!require("remotes")) install.packages("remotes")
if (!require("opusreader2")) {
  remotes::install_github("spectral-cockpit/opusreader2")
  library("opusreader2")
}
if (!require("simplerspec")) {
  remotes::install_github("philipp-baumann/simplerspec")
  library("simplerspec")
}
# not sure if I need this one?
# if (!require("hyperSpec.utils")) {
#  remotes::install_github("konradmayer/hyperSpec.utils",dependencies=TRUE)
#  library("hyperSpec.utils")
# }

rm(installed_packages) # remove objects from environment
rm(packages)
```

Grabbing filenames from .0 files for metadata .csv

The Bruker MPA outputs FT-NIRS spectra in a .0 format. Prior to loading spectral data, a metadata .csv (**LPW_metadata.csv**) was created to pair specimen numbers, lengths & otolith weights with their respective FT-NIRS scans. Included in this metadata file is a column with each scans .0 filename.

All file names in a folder can be generated with list.files(). For pulling file names to create a new .csv, you can use:

This output (without index numbers in square brackets) can be copied and pasted into Excel, then converted into columns using the "Data -> Text to Columns" function and selecting a "space" delimiter. These columns can then be transposed into a "filename" column.

Reading in FT-NIRS spectra from a Bruker MPA II

First read-in my metadata .csv and convert the session_title (i.e. scan #) to a factor. LPW FT-NIRS scans were taken in triplicate (NIR_LPW202001202_otoliths, . . . otoliths_rescan_1, . . . otoliths_rescan_2) to see if spectra changed after storage. Metadata for LPW contained an additional entry (. . . _rescan_3) with some notes and comments, however I will only be importing the metadata for actual scans.

```
setwd("/Users/zachstamplis/Desktop/Thesis and Otoliths/Pcod/LPW/LPW_FT-NIRS")
# source("LPW2020spectra.rmd", chdir = TRUE)
meta_LPW <- read.csv("LPW_metadata.csv")
names(meta_LPW)
```

```
##  [1] "specimen"         "length"           "weight"           "structure_weight"
##  [5] "region"           "collection_year"  "percent_affected" "other_problem"
##  [9] "broken"           "side"             "read_age"         "test_age"
## [13] "final_age"        "common_name"      "run_number"       "scan_name"
## [17] "file_name"        "session_title"    "comments"
```

```
levels(as.factor(as.character(meta_LPW$session_title))) # set session_title to factor (three scans take
```

```
## [1] "NIR_LPW202001202_otoliths"          "NIR_LPW202001202_otoliths_rescan_1"
## [3] "NIR_LPW202001202_otoliths_rescan_2" "NIR_LPW202001202_otoliths_rescan_3"
```

```
meta_LPW <- meta_LPW[meta_LPW$session_title %in% c("NIR_LPW202001202_otoliths", "NIR_LPW202001202_otoli

meta_LPW <- meta_LPW[meta_LPW$file_name != "", ] # remove any rows that are missing a file_name in the

# make sure you have the same number of specimens for each scan session (122 specimens, 366 total scans
nrow(meta_LPW[meta_LPW$session %in% c("NIR_LPW202001202_otoliths"), ])
```

```
## [1] 121
```

```
nrow(meta_LPW[meta_LPW$session %in% c("NIR_LPW202001202_otoliths_rescan_1"), ])
```

```
## [1] 122
```

```
nrow(meta_LPW[meta_LPW$session %in% c("NIR_LPW202001202_otoliths_rescan_2"), ])
```

```
## [1] 122
```

```
# the first scan of specimen 66 is missing and will be excluded for analysis.

# generate file paths to import with Opusreader
setwd("LPW_scans/") # setwd to folder containing FT-NIRS scans
meta_LPW$file_path <- paste0(getwd(), "/", meta_LPW$file_name)
Opusfiles <- as.vector(meta_LPW$file_path)
exists <- as.vector(lapply(Opusfiles, file.exists)) # check that I have all my files or else I get an e
meta_LPW$exists <- exists
meta_LPW1 <- meta_LPW[meta_LPW$exists == "TRUE", ] # filter the file list and data by otoliths with spe
Opusfiles <- as.vector(meta_LPW1$file_path) # I repeated this and wrote over it so I wouldn't have extr
rm(exists)
rm(meta_LPW1)
```

Once all metadata has been imported, .0 FT-NIRS scan files can be imported using opusreader2 and read_opus. First a single file is read in to make sure everything looks good; then loading all .0 files with the Opusfiles object. read_opus() creates a massive list of lists that includes information about MPA II settings, metadata generated during scans, specimen names, etc. See ?read_opus() from the opusreader2 package for more details on the list structure of .0 files.

```
# read a single file (one measurement) to check
file <- Opusfiles[1]
data_list <- opusreader2::read_opus(dsn = file) # NA's seem to be introduced due to a timestamp metadat
rm(data_list)
rm(file)
SPCfiles_nooffset <- suppressWarnings(lapply(Opusfiles, opusreader2::read_opus)) # Read in all files in

# uncomment to see additional output from .0 files

# str(SPCfiles_nooffset[[1]]) # check first element
# SPCfiles_nooffset[[1]][[1]]$ab$data # can see spc values this way I think
# SPCfiles_nooffset[[1]][[1]]$lab_and_process_param_raw$parameters #this has info about what setting wa
SPCfiles_nooffset[[1]][[1]]$lab_and_process_param_raw$parameters$FC2$parameter_value # species
```

```
## [1] "PACIFIC_COD"
```

```
SPCfiles_nooffset[[1]][[1]]$lab_and_process_param_raw$parameters$FD1$parameter_value # unique ID. Then
```

```
## [1] "LPW202001202_1_OA1"
```

```
# SPCfiles_nooffset[[1]][[1]]$ab$wavenumbers # wavenumbers scanned
SPCfiles_nooffset[[1]][[1]]$instrument_ref$parameters$INS$parameter_value # instrument name
```

```
## [1] "MPA II"
```

Once loaded, FT-NIRS scan data can be extracted from the lists.

```
spectra <- lapply(SPCfiles_nooffset, function(x) x[[1]]$ab$data) # FT-NIRS scan data
file_id <- lapply(SPCfiles_nooffset, function(x) x$lab_and_process_param_raw$parameters$FD1$parameter_va
species <- lapply(SPCfiles_nooffset, function(x) x$lab_and_process_param_raw$parameters$FC2$parameter_va
instrument <- lapply(SPCfiles_nooffset, function(x) x[[1]]$instrument_ref$parameters$INS$parameter_valu
```

```
wavenumber <- lapply(SPCfiles_nooffset, function(x) x[[1]]$ab$wavenumbers) # these could differ if sett

spectra <- lapply(spectra, as.data.frame)
# str(spectra[[1]])

for (i in 1:length(spectra)) {
  colnames(spectra[[i]]) <- wavenumber[[i]] # need to assign column  names first or else there will be
}
for (i in 1:length(spectra)) {
  spectra[[i]]$species <- species[[i]]
}
for (i in 1:length(spectra)) {
  spectra[[i]]$file_id <- file_id[[i]]
}
for (i in 1:length(spectra)) {
  spectra[[i]]$instrument <- instrument[[i]]
}
for (i in 1:length(spectra)) {
  spectra[[i]]$file_path <- Opusfiles[[i]]
}

try <- spectra[[1]] # test code on one file
splitstackshape::cSplit(as.data.frame(try$file_path), sep = "/", splitCols = "try$file_path", type.conv
```

```
##                     try$file_path_10
##                                <char>
## 1: PACIFIC_COD_LPW202001202_1_OA1.0
```

```
rm(try)

file_name <- lapply(spectra, function(x) splitstackshape::cSplit(as.data.frame(x$file_path), sep = "/",
file_name[[1]][[1, 1]]
```

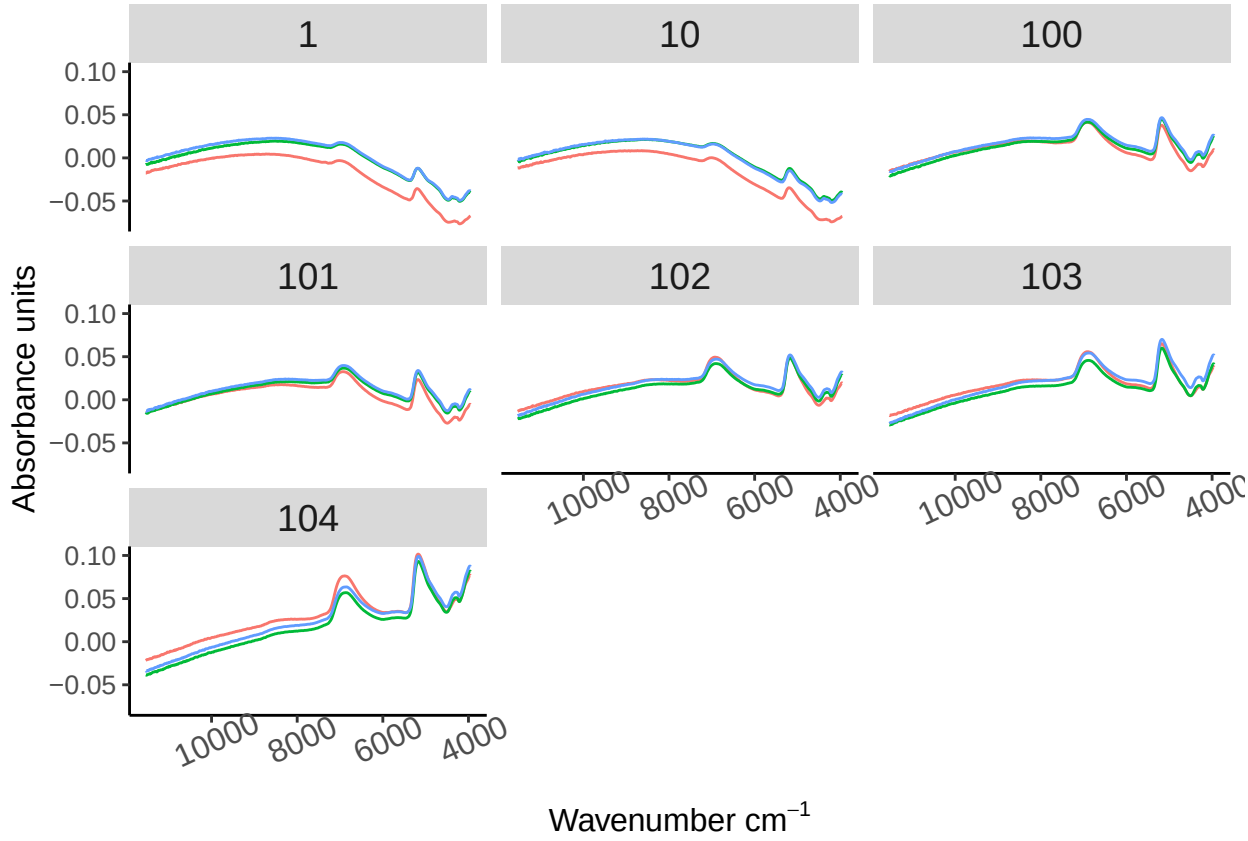
```
## [1] "PACIFIC_COD_LPW202001202_1_OA1.0"
```

```
for (i in 1:length(spectra)) {
  spectra[[i]]$file_name <- file_name[[i]][[1, 1]]
}
rm(file_id, file_name, instrument, species, wavenumber, Opusfiles, i)
```

Building a spectral dataframe after importing .0 scans

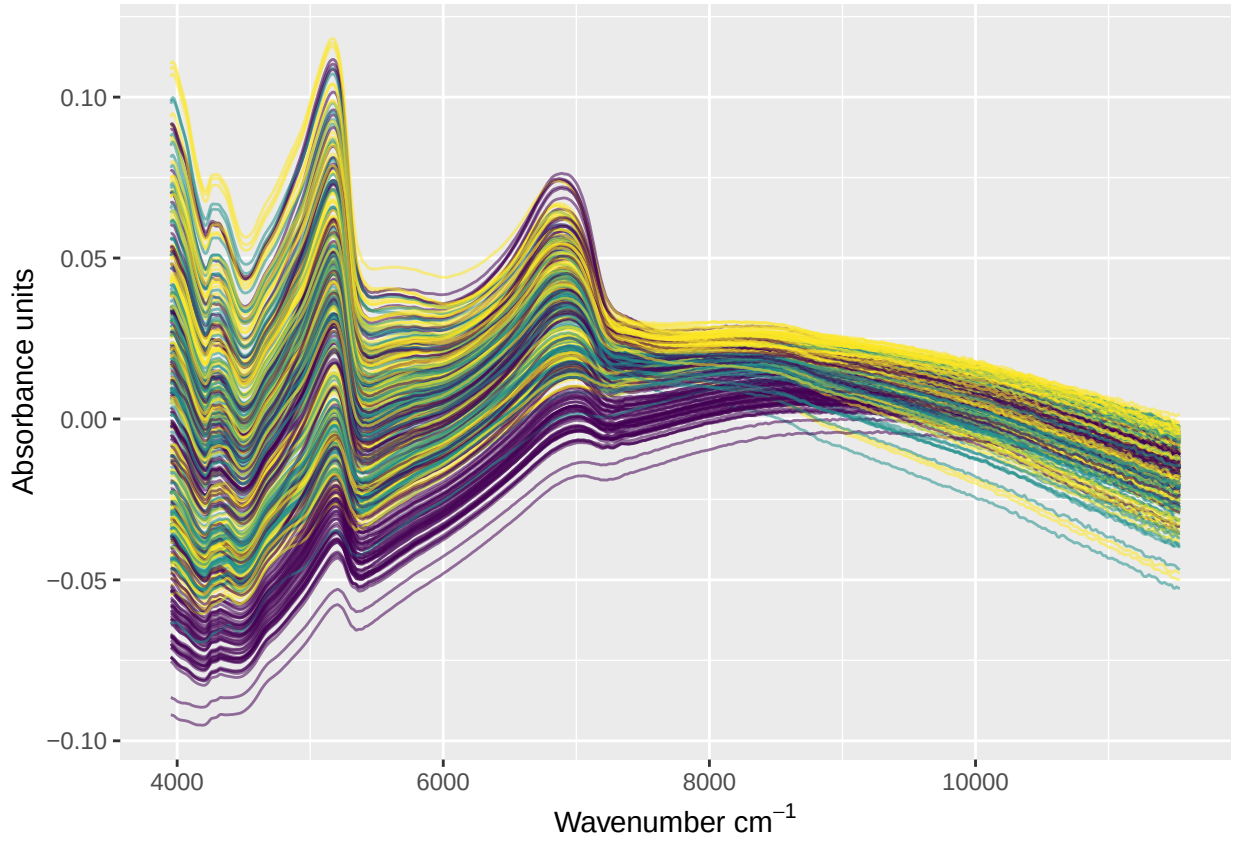
```
df <- as.data.frame(do.call(rbind, spectra))
dfmeta_LPW <- dplyr::left_join(meta_LPW, df, by = c("file_name", "file_path"))
dfmeta_LPW <- dfmeta_LPW %>% select(-c("exists", "instrument"))
rm(meta_LPW, df, spectra, SPCfiles_nooffset)
colnames(dfmeta_LPW) <- as.character(colnames(dfmeta_LPW))
dfmeta_longLPW <- pivot_longer(dfmeta_LPW, cols = -c(1:20)) # make long format, excluding all non-measu
dfmeta_longLPW <- dfmeta_longLPW %>% rename(., "wavenumber" = "name") # rename wavenumber column
```

```
# fix class of measurements to a numeric
dfmeta_longLPW$wavenumber <- as.numeric(as.character(dfmeta_longLPW$wavenumber))

# Plot to see if data loaded properly.
ggplot() +
  geom_path(data = dfmeta_longLPW[dfmeta_longLPW$specimen %in% c(1, 10, 100, 101, 102, 103, 104), ],
            aes(x = wavenumber, y = value, color = session_title, group = file_name), linewidth = .5) +
  scale_x_reverse() +
  labs(y = "Absorbance units", x = expression(paste("Wavenumber ", cm^-1))) +
  theme(
    axis.text = element_text(size = 10),
    axis.text.x =element_text(size=12,angle=25),
    axis.title = element_text(size = 12),
    legend.position = "none",
    strip.text = element_text(size = 14),
    legend.text = element_text(size = 10),
    legend.title = element_text(size = 12),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.background = element_blank(),
    axis.line = element_line(colour = "black")
  ) +
  facet_wrap(~specimen)
```

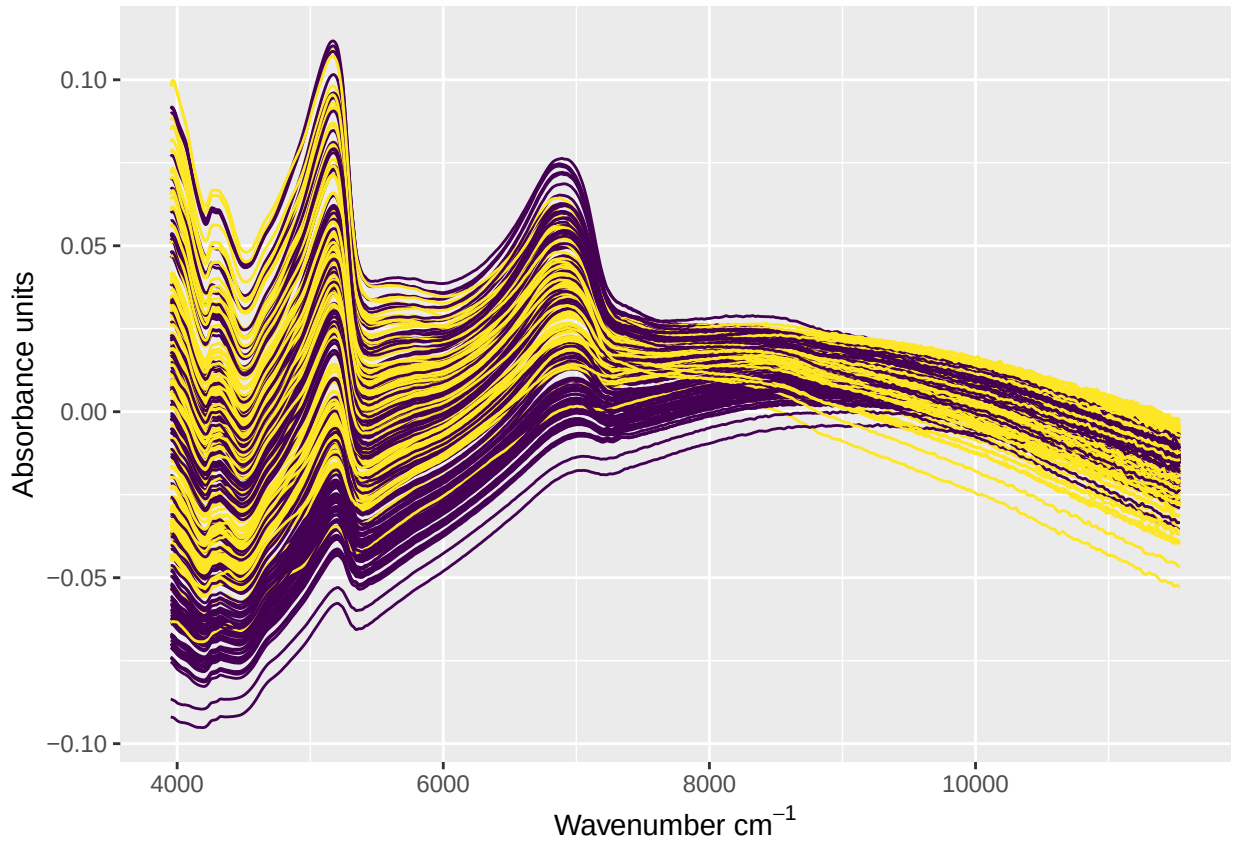


Comparison between scans & PCA of all scans

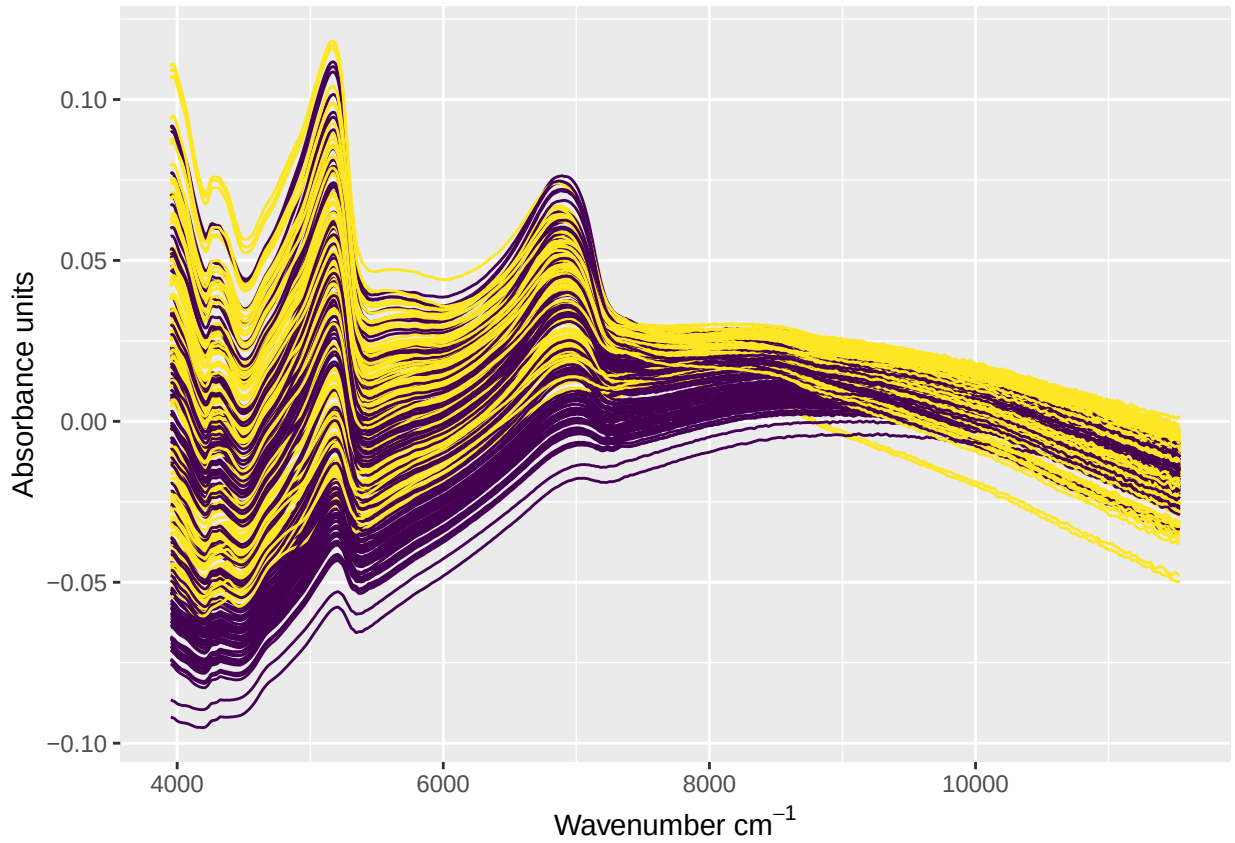
```r
# set run_number as factor for color groupings
dfmeta_longLPW$run_number <- as.factor(dfmeta_longLPW$run_number)
dfmeta_LPW$run_number <- as.factor(dfmeta_LPW$run_number)
# FT-NIRS scans, all 3 runs, colored by run_number
ggplot(dfmeta_longLPW) + geom_path(aes(x = wavenumber, y = value,
                                       color = run_number, group = file_name, alpha = 0.5)) +
  labs(y = "Absorbance units", x = expression(paste("Wavenumber ", cm^-1))) +
  theme(legend.position = "none") +
  scale_color_viridis(discrete = T)
```



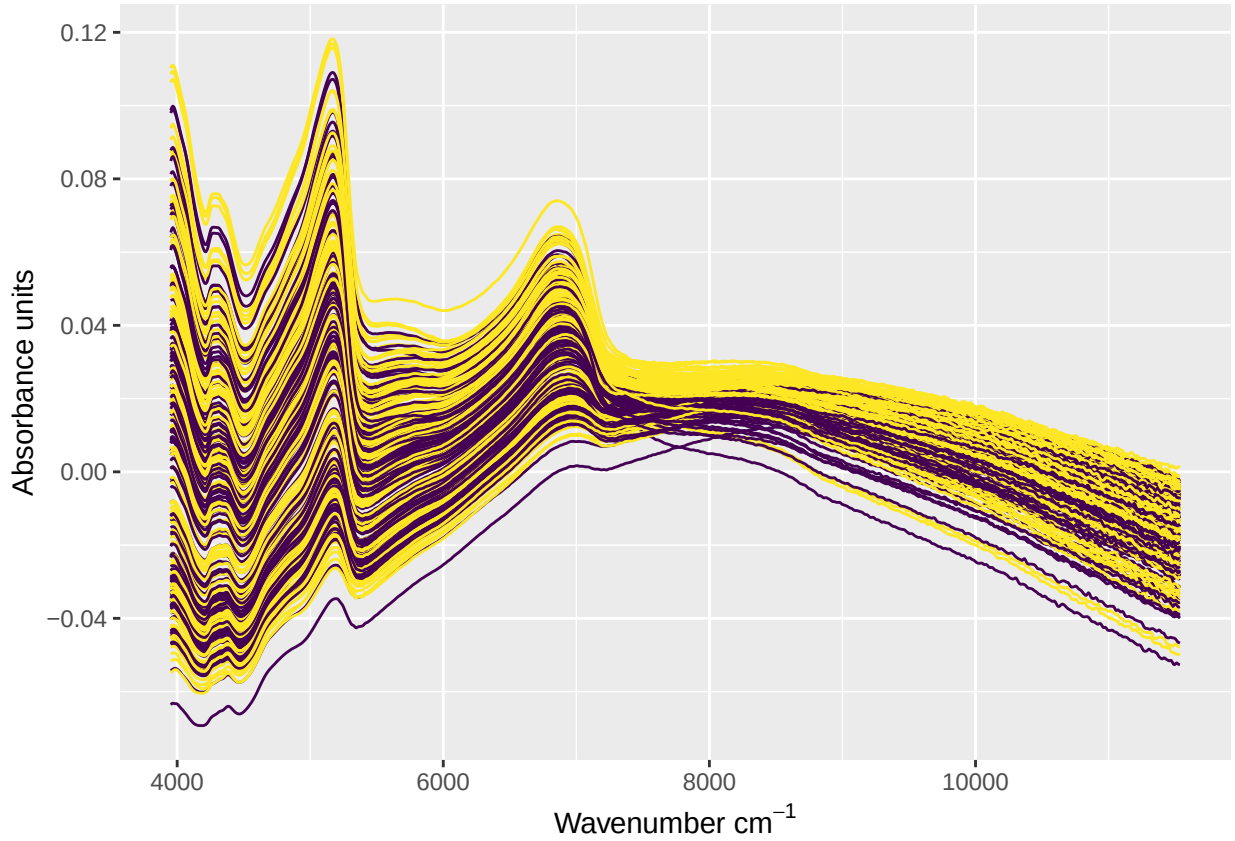

```r
# appears to be slightly different  values for each scan, look at 1 & 2 first
ggplot(dfmeta_longLPW[dfmeta_longLPW$run_number %in% c(1,2),]) + geom_path(aes(x = wavenumber, y = valu
  labs(y = "Absorbance units", x = expression(paste("Wavenumber ", cm^-1))) +
  theme(legend.position = "none") +
  scale_color_viridis(discrete = T)
```



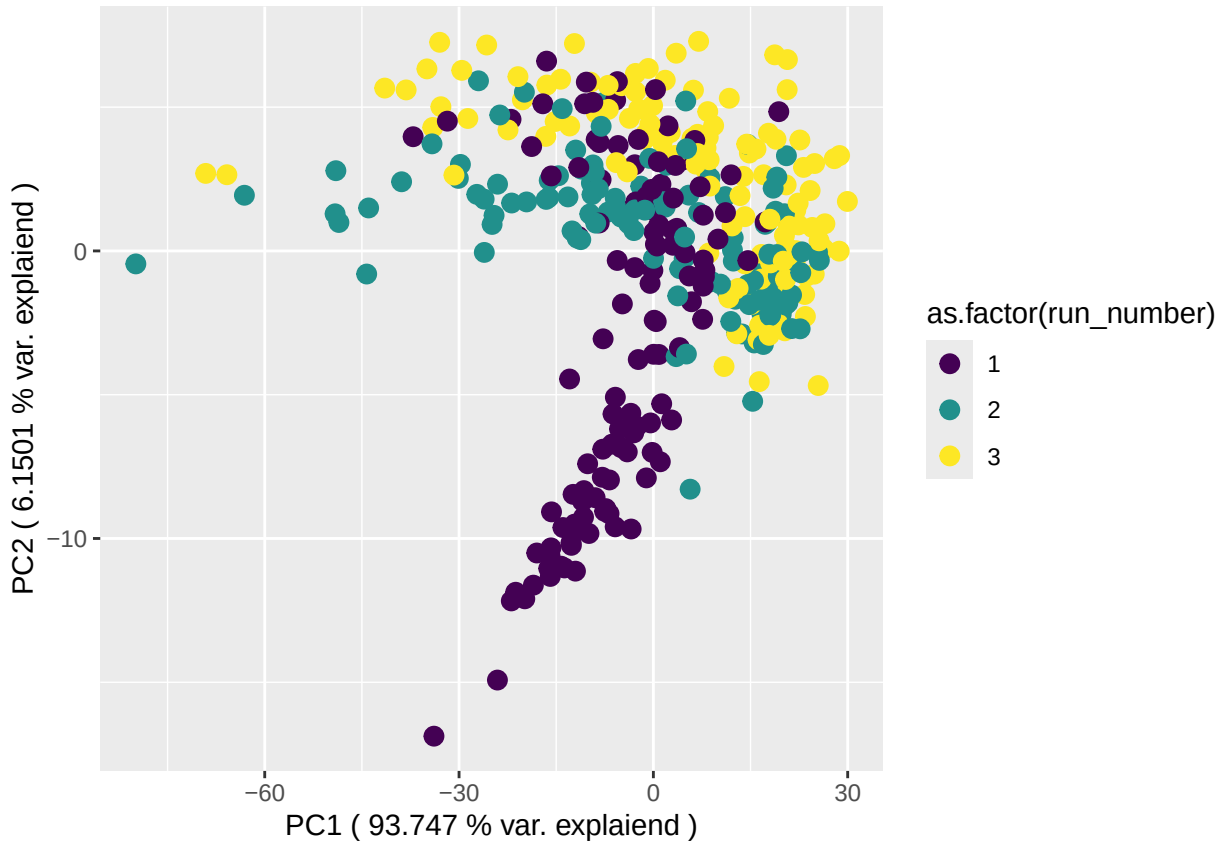

```
# clearly different, now 1 & 3
ggplot(dfmeta_longLPW[dfmeta_longLPW$run_number %in% c(1,3),]) + geom_path(aes(x = wavenumber, y = valu
  labs(y = "Absorbance units", x = expression(paste("Wavenumber ", cm^-1))) +
  theme(legend.position = "none") +
  scale_color_viridis(discrete = T)
```




```
# also different, finally 2 & 3
ggplot(dfmeta_longLPW[dfmeta_longLPW$run_number %in% c(2,3),]) + geom_path(aes(x = wavenumber, y = valu
  labs(y = "Absorbance units", x = expression(paste("Wavenumber ", cm^-1))) +
  theme(legend.position = "none") +
  scale_color_viridis(discrete = T)
```




```
# These should be triplicate scans with identical values for each of the 122 specimens.  To confirm, I'
pca_all_LPW <- pca(dfmeta_LPW[21:nrow(dfmeta_LPW)], scale = T)
pcs <- as.data.frame(cbind(pc2 = pca_all_LPW$calres$scores[,2], run_number = dfmeta_LPW$run_number)) #
pcs <- cbind(pc1 = pca_all_LPW$calres$scores[,1], pcs) # extract scores for PC1

# plot PC1 & PC2, colored by run_number
ggplot(pcs) +
  geom_point(aes(x = pc1, y = pc2, color = as.factor(run_number)), size = 3) +
  labs(x = paste("PC1 (",
                 round(pca_all_LPW$calres$expvar[1], digits = 4),
                 "% var. explaiend )"),
       y = paste("PC2 (",
                 round(pca_all_LPW$calres$expvar[2], digits = 4),
                 "% var. explaiend )")) +
  scale_color_viridis(discrete = T)
```




```
# run/scan #1 looks way different.  Something appears to have changed between scan periods, however it
```

Combining/averaging LPW FT-NRIS scans 2 & 3

```
scan_2 <- dfmeta_LPW %>% dplyr::filter(run_number == 2)
# scan_2_long <- pivot_longer(scan_2, cols = `11536`:`3952`, names_to = "name", values_to = "value")
# scan_2_long$name <- as.numeric(scan_2_long$name)
scan_3 <- dfmeta_LPW %>% dplyr::filter(run_number == 3)
# scan_3_long <- pivot_longer(scan_3, cols = `11536`:`3952`, names_to = "name", values_to = "value")
# scan_3_long$name <- as.numeric(scan_3_long$name)
scan_avg <- bind_cols(NULL, scan_2[, 1:20])
scan_avg <- bind_cols(scan_avg, (scan_2[, 21:ncol(scan_2)] + scan_3[, 21:ncol(scan_3)]) / 2)
rm(scan_2, scan_3)
scan_avg_long <- pivot_longer(scan_avg, cols = `11536`:`3952`, names_to = "name", values_to = "value")
scan_avg_long$name <- as.numeric(scan_avg_long$name)
```

Preprocessing Spectra

Initial preprocessing done with prospectr and a savitzkygolay filter. might make more sense to preprocess after selecting important variables? Some functions included to play around with preprocessing filters and select optimal filters.

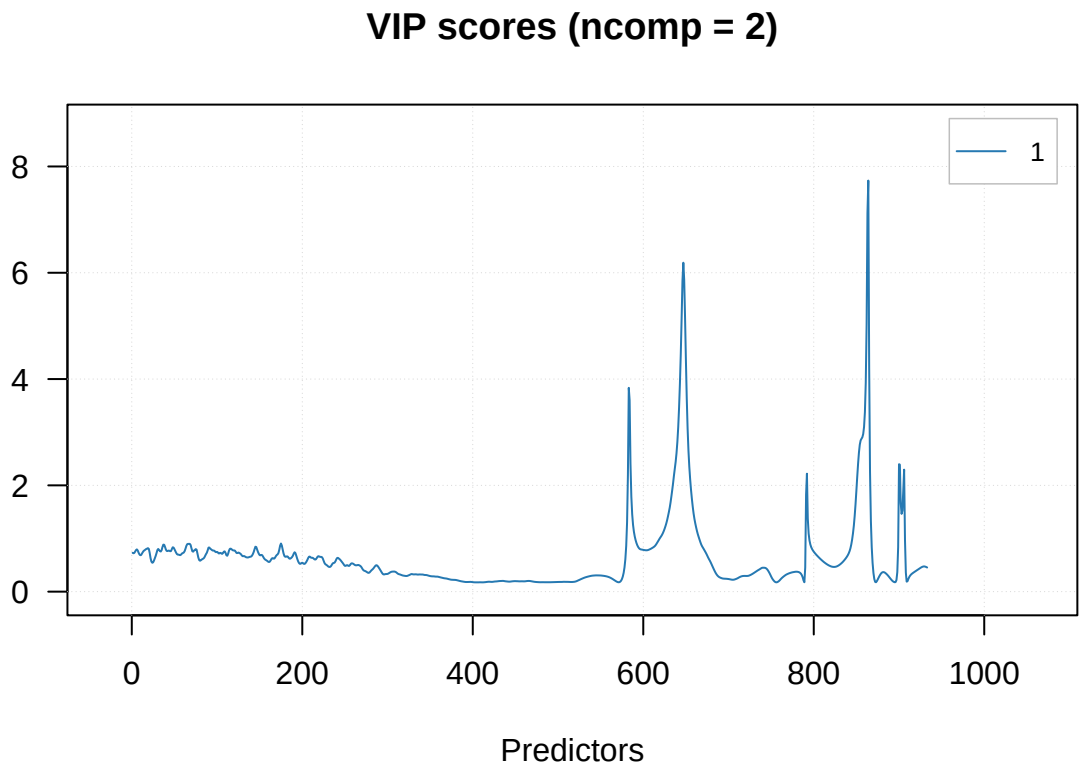
```r
spec.fig <- function(mydf, color) { # ggplot function to plot spectra, colored by some factor
  ggplot(mydf) +
    geom_path(aes(x = name, y = value, color = {{ color }}, group = file_name)) +
    scale_x_reverse() +
    scale_color_viridis() +
    labs(y = "Preprocessed absorbance", x = expression(paste("Wavenumber ", cm^-1)))
    #theme(axis.text = element_text(size = 16),
      # axis.text.x = element_text(size = 12, angle = 25),
      # axis.title = element_text(size = 14),
      # legend.position = "right",
      # strip.text = element_text(size = 14),
      # legend.text = element_text(size = 14),
      # legend.title = element_text(size = 14),
      # panel.grid.major = element_blank(),
      # panel.grid.minor = element_blank(),
      # panel.background = element_blank(),
      # axis.line = element_line(colour = "black")
    #)
}
# function to plot results of different savitzkyGolay filter params
sg_plotting <- function(color, m, p, w) {
  dftempproc <- as.data.frame(
    cbind(scan_avg[, c(1:20)],
          savitzkyGolay(scan_avg[,21:length(scan_avg)],
                        m = m, p = p, w = w)
  ))
  dftempproc_long <- tidyr::pivot_longer(dftempproc, cols = c(21:length(dftempproc)))
  dftempproc_long$name <- as.numeric(as.character(dftempproc_long$name))
  spec.fig(mydf = dftempproc_long, color = {{ color }}) +
    ggtitle(paste("diff = ", {{ m }}, "poly = ", {{ p}}, "window = ", {{ w }}))
}
# function to apply savitzkyGolay filter with selected parameters to dataframe and create new temp_proc
quickproc <- function(df, m, p, w){
  temp_proc <<- as.data.frame(
    cbind({{df}}[, c(1:20)],
          savitzkyGolay({{df}}[,21:length({{df}})],
                        m = m, p = p, w = w)
  ))
  temp_proc_long <<- tidyr::pivot_longer(temp_proc, cols = c(21:length(temp_proc)))
  temp_proc_long$name <<- as.numeric(as.character(temp_proc_long$name))
}
```

Variable/Wavelength Selection & PLS on length as a test

Using temp_proc df produced from quickproc() function above. VIP score to select wavelengths that are most informative to model

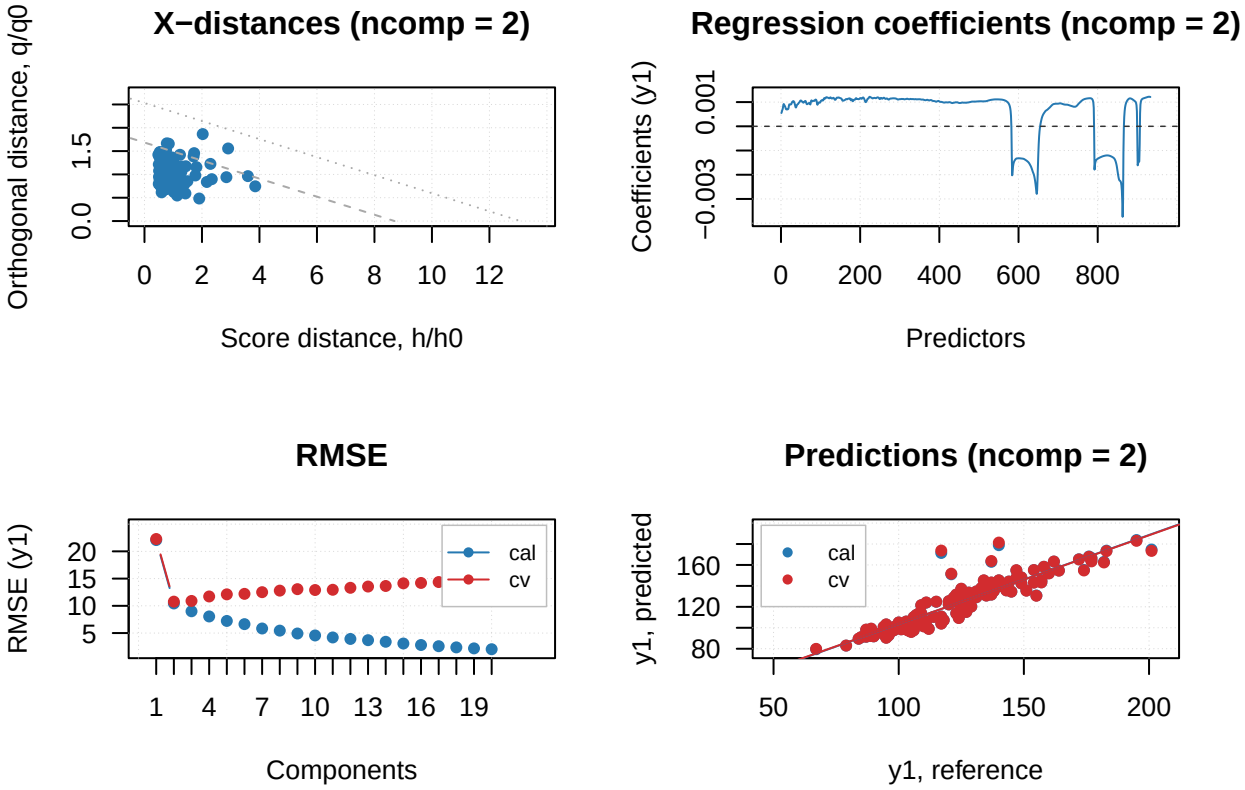
```r
quickproc(scan_avg,1,2,17) # SG of scan_avg DF to genereate temp_proc df for models below

test_pls_length <- pls(temp_proc[, 21:ncol(temp_proc)], temp_proc[, 2],
  scale = T, center = F, info = "Length Prediction Model",
  cv = 1
```

```
)

# pls only with wavenumebrs VIP > 1
length_VIP <- vipscores(test_pls_length) # extract VIP values for wavenumbers of above pls model
plotVIPScores(test_pls_length)
```

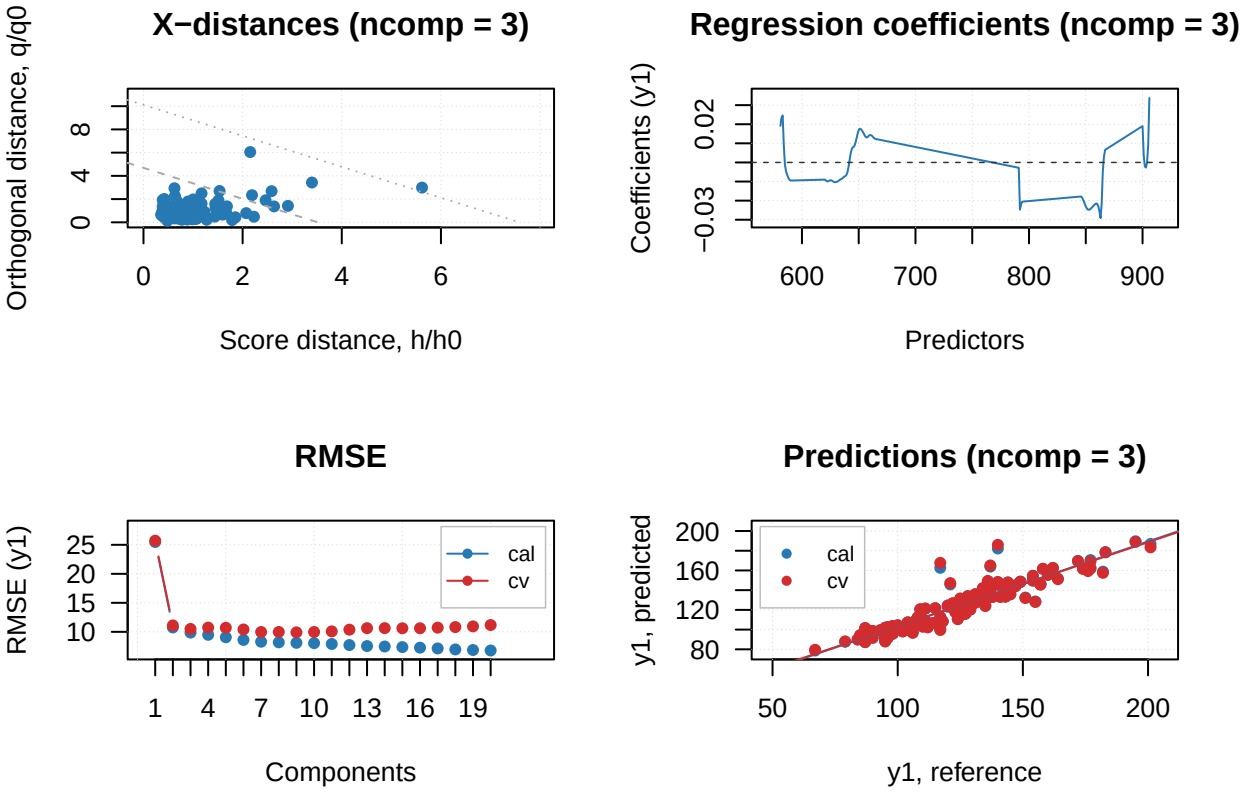



```
test_waves <- names(subset(length_VIP[,1], length_VIP[,1] > 1))
all_waves <- names(temp_proc[,21:length(temp_proc)])
bad_waves <- all_waves[!(all_waves %in% test_waves)]
test_pls_length_VIP <- pls(temp_proc[,21:ncol(temp_proc)], temp_proc[,2],
  scale = T, center = F, info = "Length Prediction Model",
  exclcols = bad_waves,
  cv = 1
)

# unprocessed df PLS
test_pls_length_unproc <- pls(scan_avg[, 21:ncol(scan_avg)], scan_avg[, 2],
  scale = T, center = F, info = "Length Prediction Model",
  cv = 1
)


# 90% train, 10% test PLS, no VIP
set.seed(1)
idx <- floor(nrow(temp_proc) * 0.9)
```

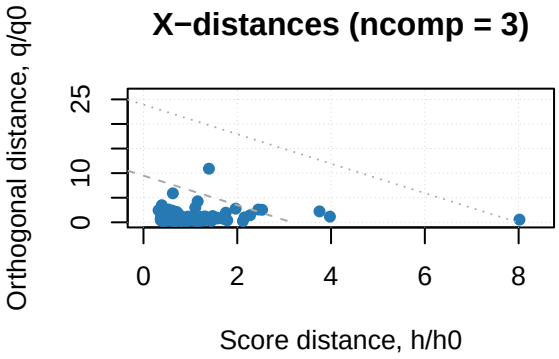
```r
train_idx <- sample(seq_len(nrow(temp_proc)), size = idx)
train <- temp_proc[train_idx, -c(1,3:20)]
test <- temp_proc[-train_idx, -c(1,3:20)]
test_pls_length_split <- pls(train[,-c(1)], train[,1],
  scale = T, center = F,
  info = "Length Prediction Model", cv = 1,
  x.test = test[,-c(1)], y.test = test[,1]
)

# 90% train, 10% test PLS, VIP
test_pls_length_split_VIP <- pls(train[,-c(1)], train[,1],
  scale = T, center = F,
  info = "Length Prediction Model", cv = 1,
  x.test = test[,-c(1)], y.test = test[,1],
  exclcols = bad_waves
)

plot(test_pls_length)
```

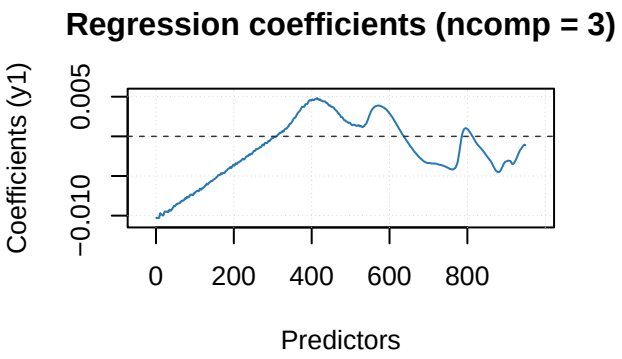


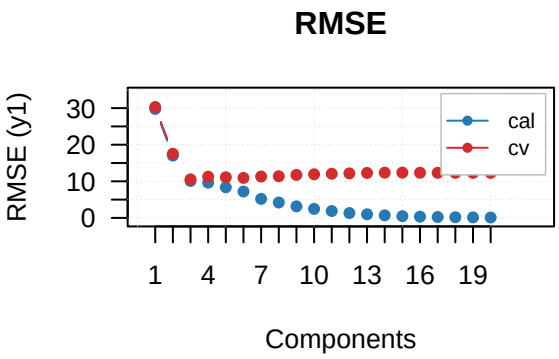

```r
plot(test_pls_length_VIP)
```

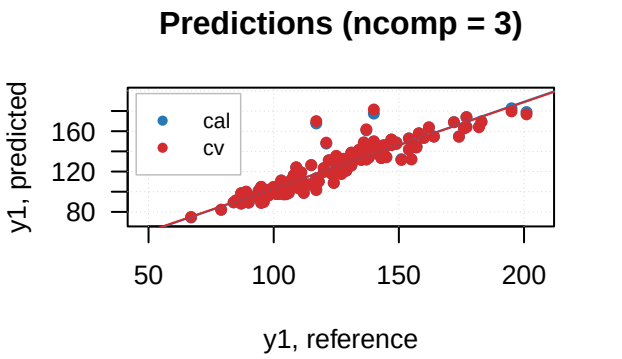



```
plot(test_pls_length_unproc)
```

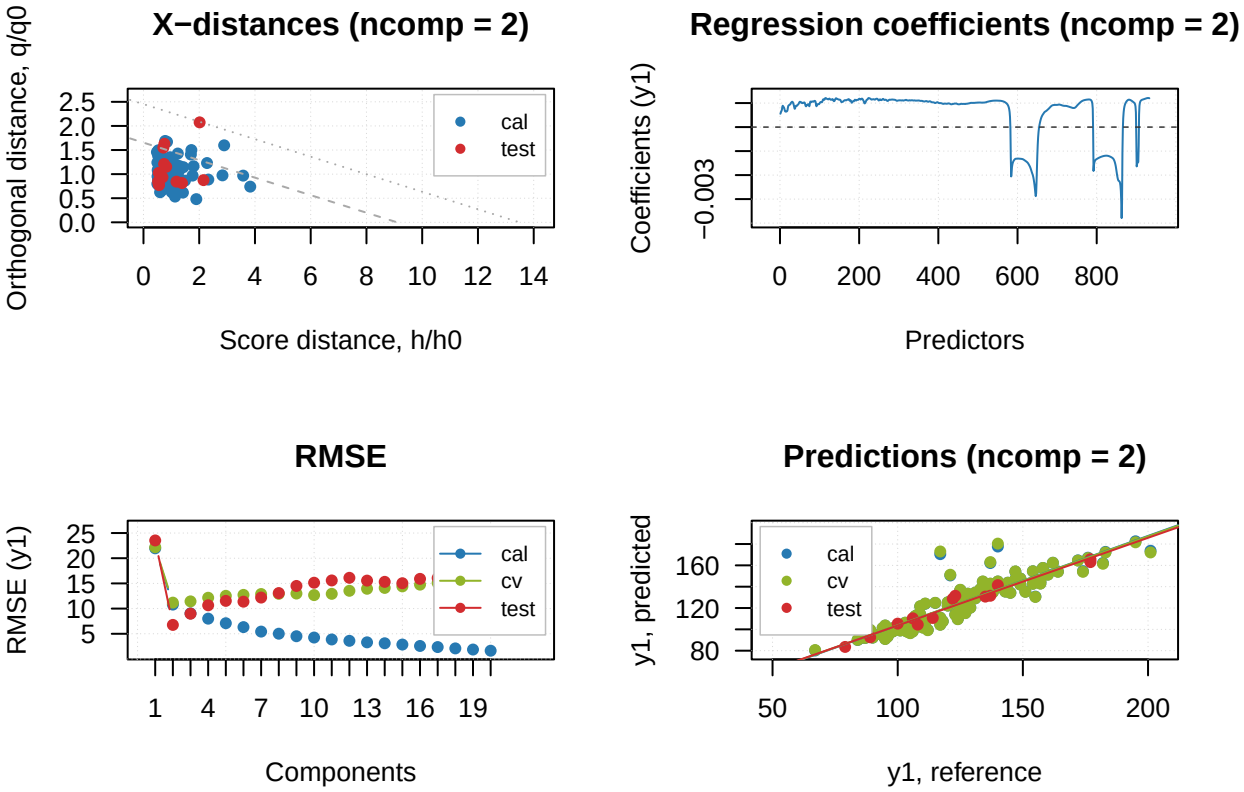




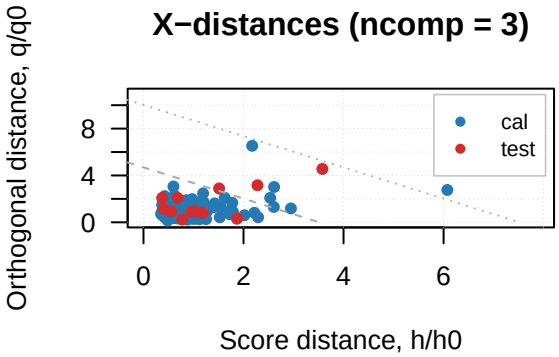


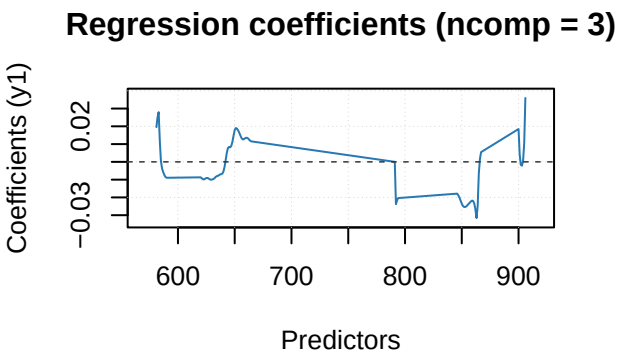


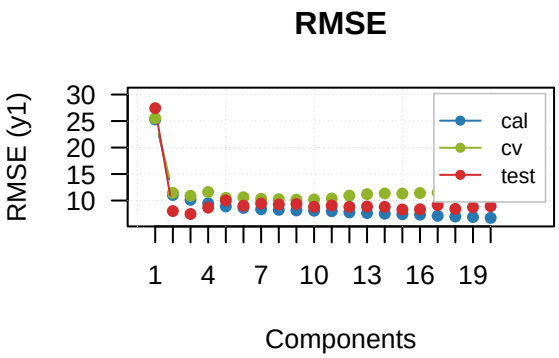

```
plot(test_pls_length_split)
```

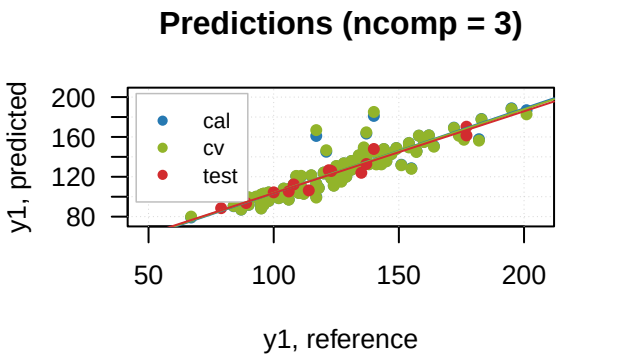



```
plot(test_pls_length_split_VIP)
```










```
test_pls_length$res$cal$rmse[2]
```

```
## [1] 10.43392
```

```
test_pls_length_VIP$res$cal$rmse[3]
```

```
## [1] 9.866577
```

```
test_pls_length_unproc$res$cal$rmse[3]
```

```
## [1] 10.07586
```

```
test_pls_length_split$res$test$rmse[2]
```

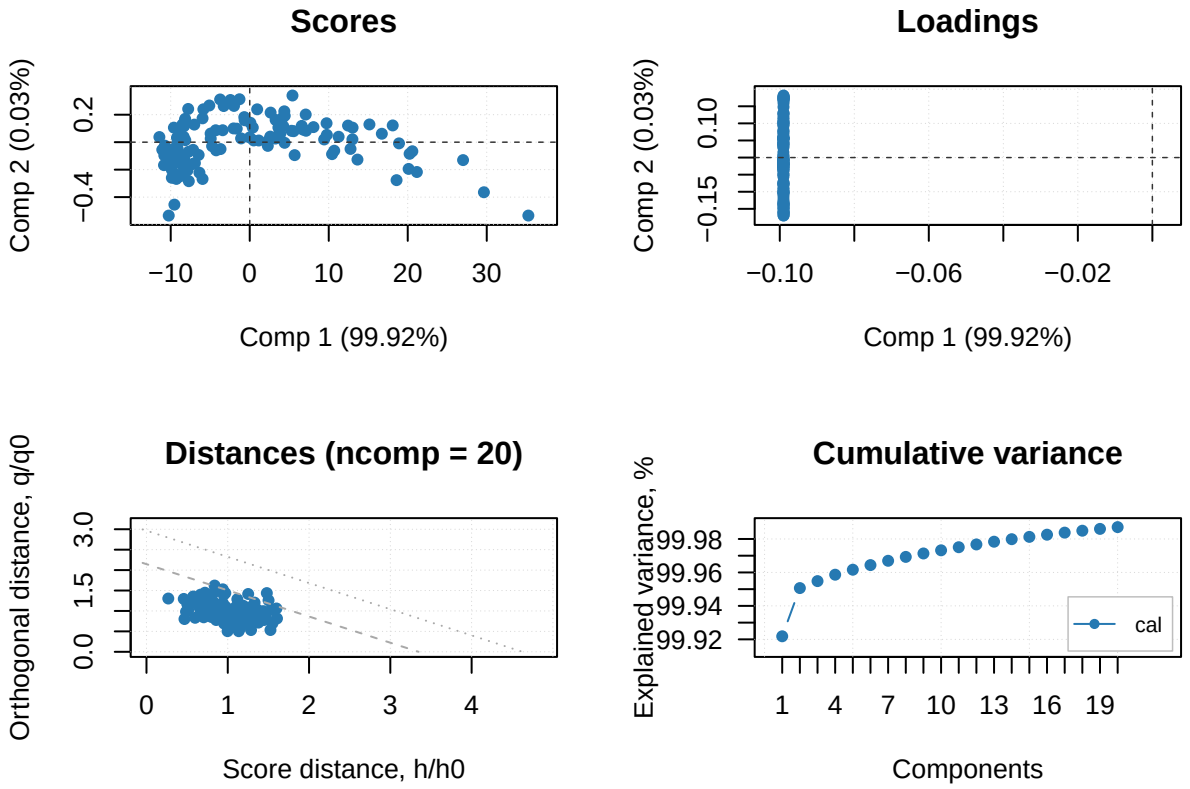
```
## [1] 6.759596
```

```
test_pls_length_split_VIP$res$test$rmse[3]
```

```
## [1] 7.46186
```

MLR & PCA using length

```r
pca_avg_LPW <- pca(scan_avg[21:nrow(scan_avg)], scale = T) # PCA for scan_avg
plot(pca_avg_LPW)
```




```r
mlr_pcs <- pca_avg_LPW$calres$scores[, 1:20] # extract PCs
mlr_pcs <- cbind(mlr_pcs[, 1:5], scan_avg[, 1:4]) # only store first 5 PC's
mlr_length <- lm(data = mlr_pcs, length ~ `Comp 1` + `Comp 2` + `Comp 3` + `Comp 4` + `Comp 5`)
summary(mlr_length) # only appear to need first 2 comps
```

```
##
## Call:
## lm(formula = length ~ `Comp 1` + `Comp 2` + `Comp 3` + `Comp 4` +
##     `Comp 5`, data = mlr_pcs)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -50.944  -5.600   0.456   6.679  23.577
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 123.27869    0.95139 129.577  < 2e-16 ***
## `Comp 1`      2.36945    0.09463  25.040  < 2e-16 ***
## `Comp 2`     43.68759    5.57116   7.842 2.39e-12 ***
## `Comp 3`    -18.43889   14.66028  -1.258    0.211
## `Comp 4`     -6.00035   15.29366  -0.392    0.696
```

```
## `Comp 5`     -7.21902   17.31533  -0.417    0.678
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.51 on 116 degrees of freedom
## Multiple R-squared:  0.8562, Adjusted R-squared:   0.85
## F-statistic: 138.1 on 5 and 116 DF,  p-value: < 2.2e-16
```

```
mlr_length <- update(mlr_length, length ~ `Comp 1` + `Comp 2`)
summary(mlr_length)
```

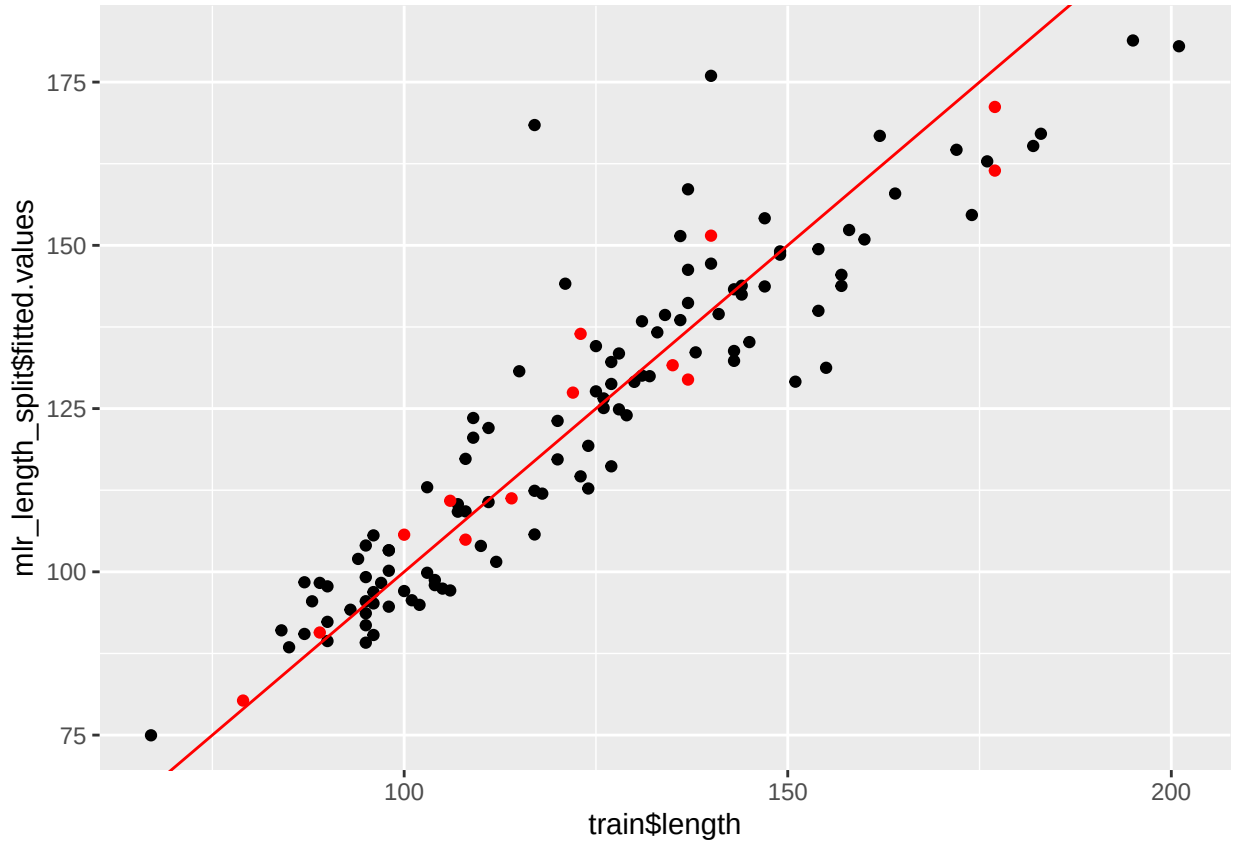
```
##
## Call:
## lm(formula = length ~ `Comp 1` + `Comp 2`, data = mlr_pcs)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -52.161  -5.205   0.403   5.730  23.797
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 123.27869    0.94702 130.175  < 2e-16 ***
## `Comp 1`      2.36945    0.09419  25.155  < 2e-16 ***
## `Comp 2`     43.68759    5.54558   7.878 1.75e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.46 on 119 degrees of freedom
## Multiple R-squared:  0.8538, Adjusted R-squared:  0.8513
## F-statistic: 347.4 on 2 and 119 DF,  p-value: < 2.2e-16
```

```
# MLR with split
set.seed(1)
idx <- floor(nrow(scan_avg) * 0.9) # 90% of indices needed for training set
train_idx <- sample(seq_len(nrow(scan_avg)), size = idx) # generate indices for training set
train <- mlr_pcs[train_idx, -c(6,8:9)] # wavenumber measurement columns only
test <- mlr_pcs[-train_idx, -c(6,8:9)]
mlr_length_split <- lm(data = train, length ~ `Comp 1` + `Comp 2`)
summary(mlr_length_split)
```

```
##
## Call:
## lm(formula = length ~ `Comp 1` + `Comp 2`, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -51.421  -5.305   0.432   6.022  23.753
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  123.327      1.031 119.593  < 2e-16 ***
## `Comp 1`       2.337      0.102  22.912  < 2e-16 ***
## `Comp 2`      45.640      6.275   7.274 6.31e-11 ***
```

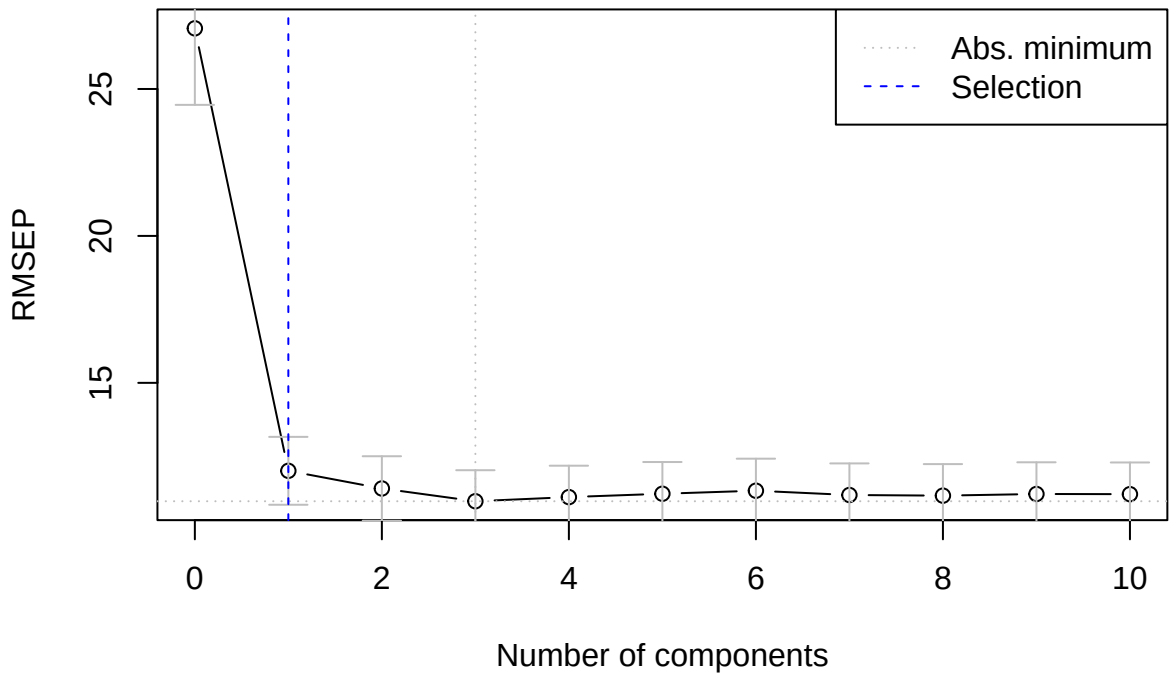
```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.76 on 106 degrees of freedom
## Multiple R-squared:  0.8433, Adjusted R-squared:  0.8403
## F-statistic: 285.2 on 2 and 106 DF,  p-value: < 2.2e-16
```

```
mlr_length_split_pred <- predict(mlr_length_split,test)
ggplot() +  # fitted values vs actual, out of sample prediction in red
  geom_point(aes(train$length,mlr_length_split$fitted.values)) +
  geom_point(aes(test$length,mlr_length_split_pred),col = "red") +
  geom_abline(slope=1, intercept = 0, col = "red")
```



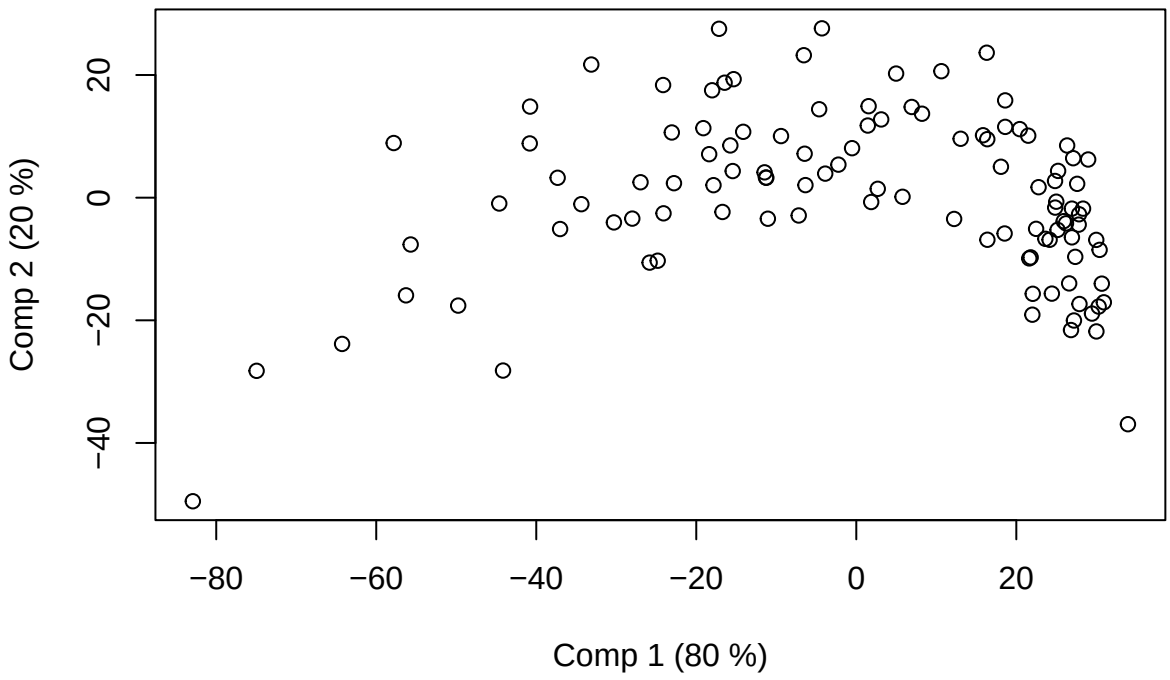

```
# Split for PCR
set.seed(1)
idx <- floor(nrow(scan_avg) * 0.9) # 90% of indices needed for training set
train_idx <- sample(seq_len(nrow(scan_avg)), size = idx) # generate indices for training set
train <- scan_avg[train_idx, -c(1, 3:20)] # length and wavenumber measurement columns only
test <- scan_avg[-train_idx, -c(1, 3:20)]

# PCR model
pcr_length_model <- pls::pcr(length ~ .,
  data = train,
  scale = T,
  validation = "CV",
```

```
  ncomp = 10
) # look at 10 components max
selectNcomp(pcr_length_model, "onesigma", plot = TRUE) # determine appropriate number of comps
```

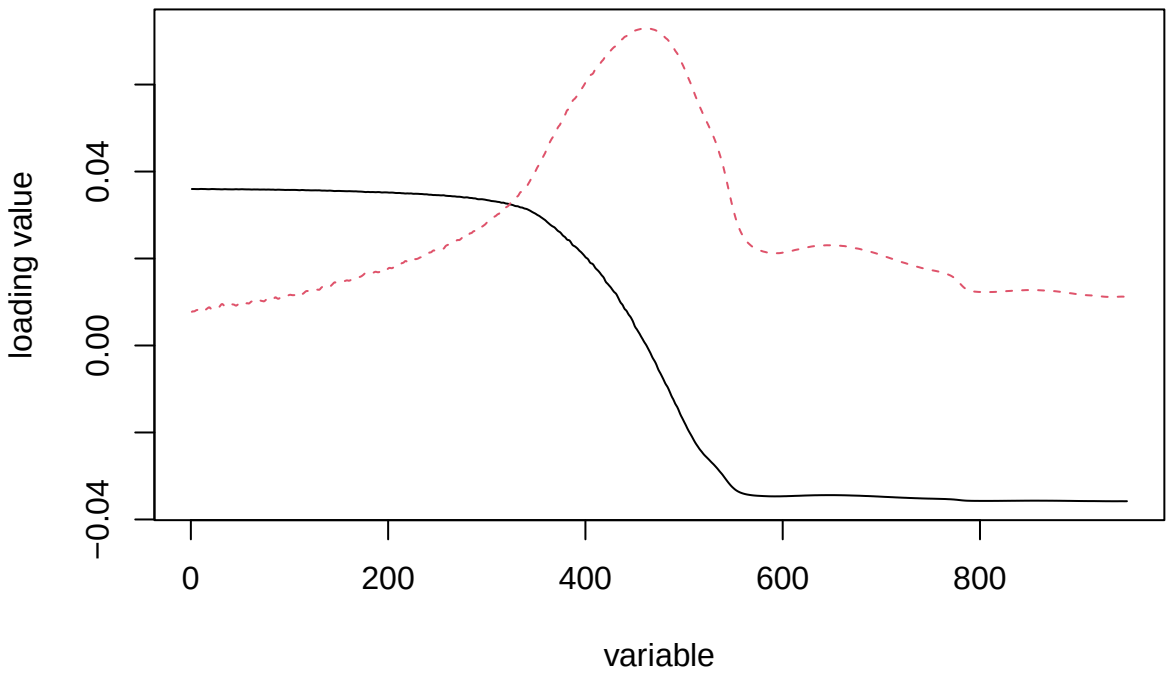



```
## [1] 1
```

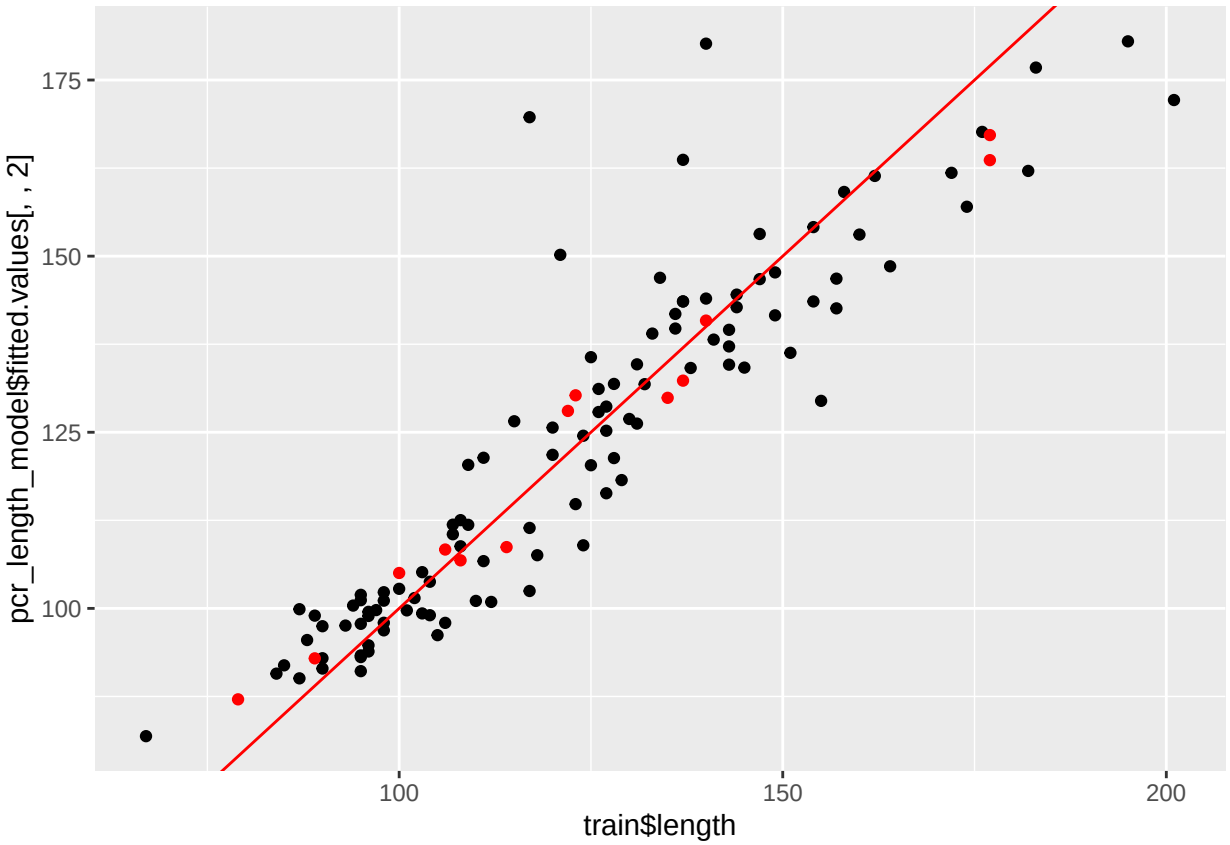
```
scoreplot(pcr_length_model) # plot of first 2 PCs
```




```
loadingplot(pcr_length_model)
```

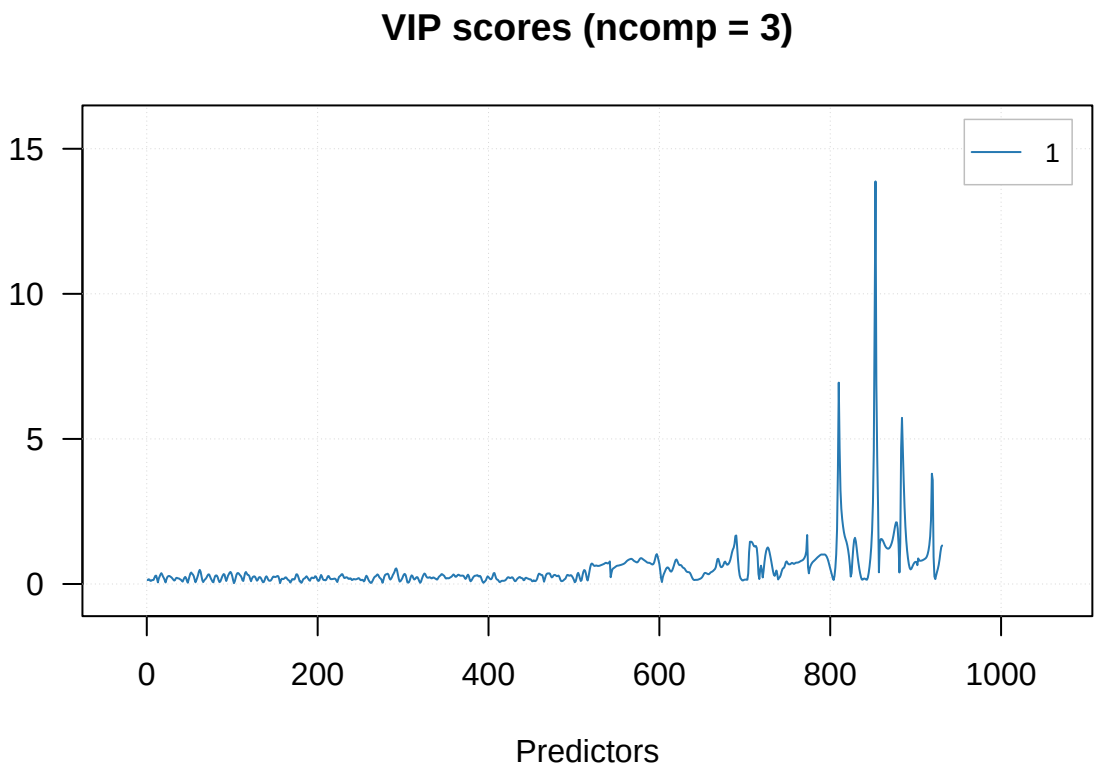



```
pcr_length_pred <- predict(pcr_length_model, newdata = test, ncomp = 2) # fit model to test data, store
ggplot() +
  geom_point(aes(x = train$length, y = pcr_length_model$fitted.values[, , 2])) + # extract fitted value
  geom_point(aes(x = test$length, y = pcr_length_pred), col = "red") +
  geom_abline(slope = 1, col = "red")
```



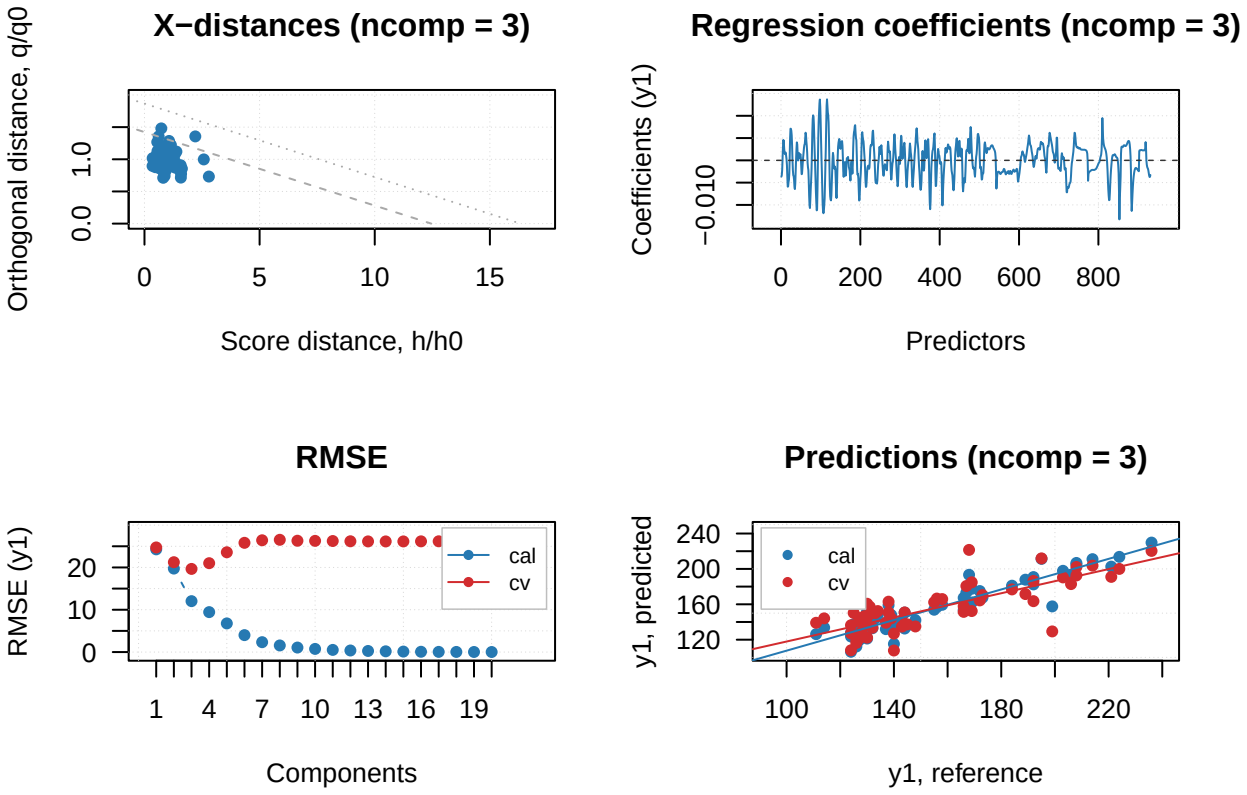
Remove specimens missing age data, pls for age

```
# df with only aged specimens
age_only <- scan_avg[complete.cases(scan_avg$read_age),]
# generate preprocessed df of ages, temp_proc & temp_proc_long
quickproc(age_only,2,3,19)
# test pls, no variable selection
test_pls_age <- pls(temp_proc[, 21:ncol(temp_proc)], temp_proc[, 11],
  scale = T, center = F, info = "Age Prediction Model",
  cv = 1
)

# pls with VIP > 1
test_vip_age <- vipscores(test_pls_age)
plotVIPScores(test_pls_age)
```

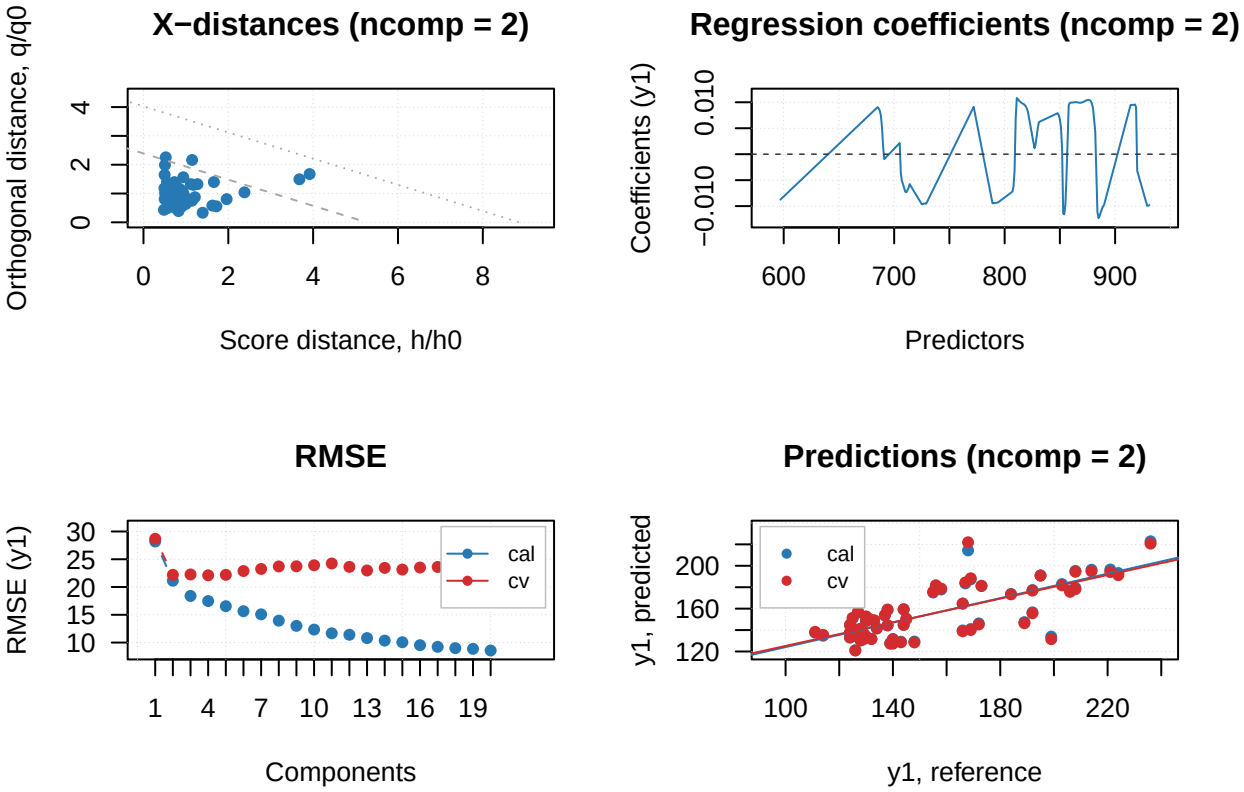



```
good_waves <- as.numeric(names(subset(test_vip_age[,1], test_vip_age[,1] > 1)))
all_waves <- names(temp_proc[,21:length(temp_proc)])
bad_waves <- all_waves[!(all_waves %in% good_waves)]

test_pls_age_VIP <- pls(temp_proc[, 21:ncol(temp_proc)], temp_proc[, 11],
 scale = T, center = F, info = "Age Prediction Model",
 cv = 1, exclcols = bad_waves
)

# 90/10 split for PLS, no VIP
set.seed(1)
idx <- floor(nrow(temp_proc) * 0.9)
train_idx <- sample(seq_len(nrow(temp_proc)), size = idx)
train <- temp_proc[train_idx, -c(1:10,12:20)]
test <- temp_proc[-train_idx, -c(1:10,12:20)]

test_pls_age_split <- pls(train[,-1], train[,1],
  scale = T, center = F,
  info = "Age Prediction Model", cv = 1,
  x.test = test[,-1], y.test = test[,1]
)

# 90/10 split for PLS, VIP > 1
test_pls_age_split_VIP <- pls(train[,-1], train[,1],
  scale = T, center = F,
  info = "Age Prediction Model", cv = 1,
```

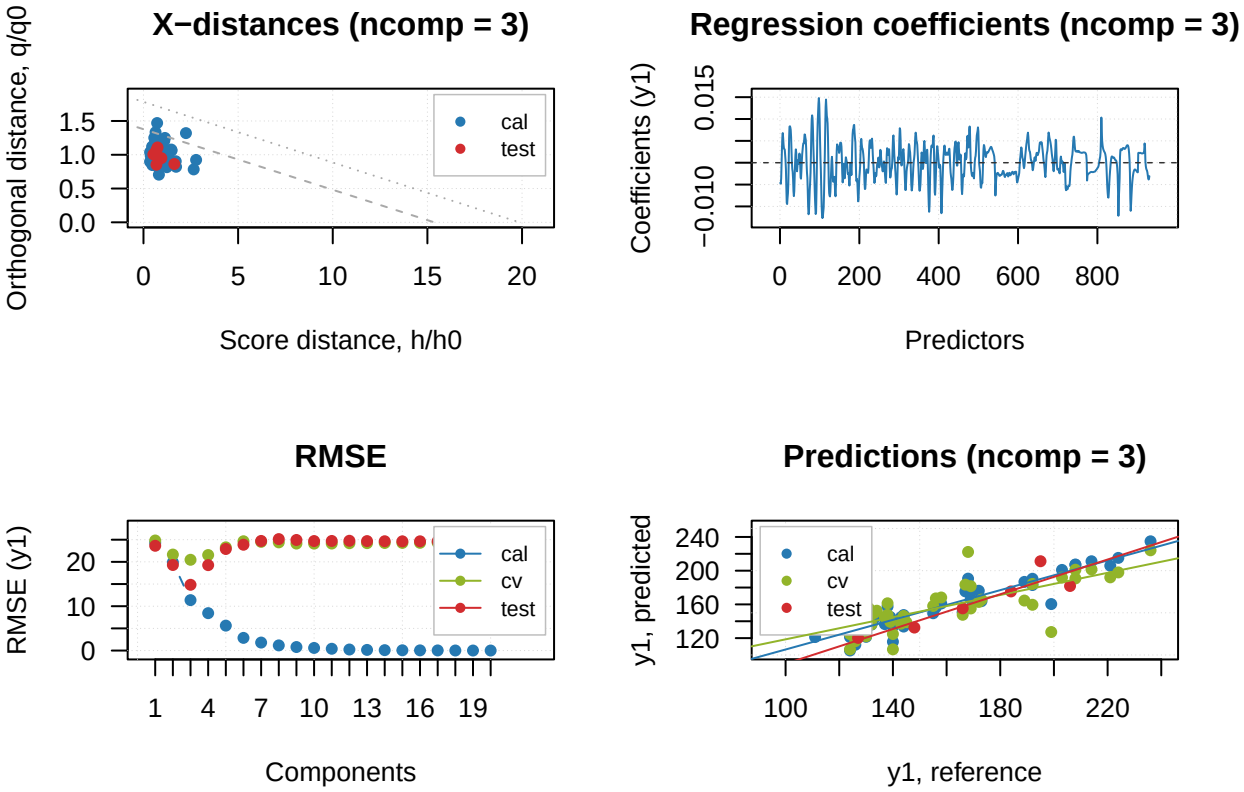
```
  x.test = test[,-c(1)], y.test = test[,1],
  exclcols = bad_waves
)

plot(test_pls_age)
```



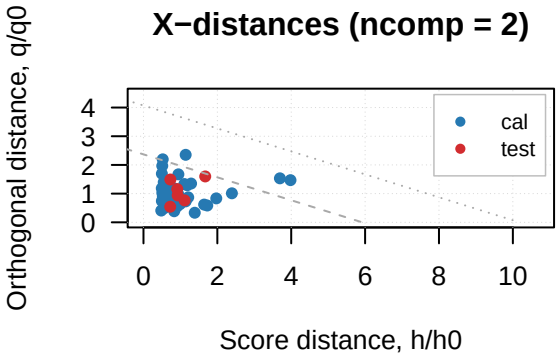

```
plot(test_pls_age_VIP)
```

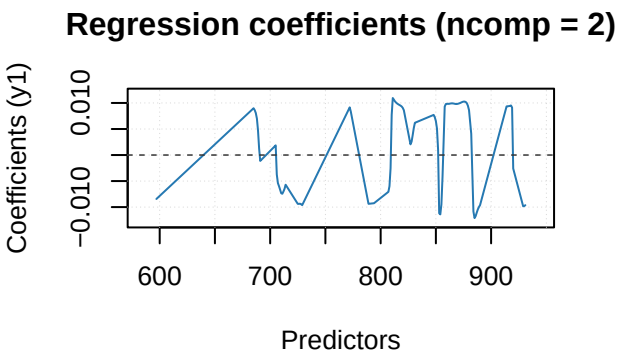


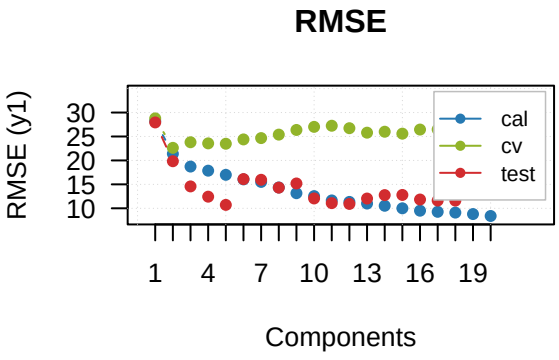

```
plot(test_pls_age_split)
```

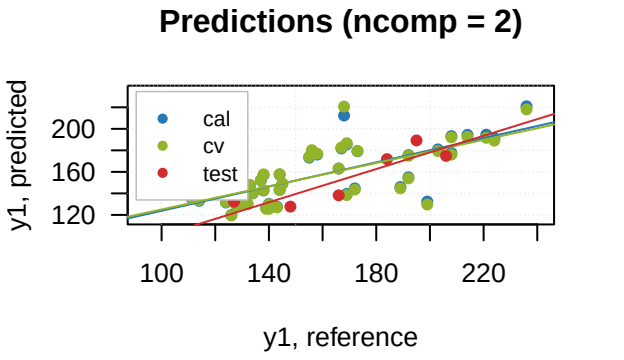



```
plot(test_pls_age_split_VIP)
```










```
test_pls_age$res$cal$rmse[3]
```

```
## [1] 12.02924
```

```
test_pls_age_VIP$res$cal$rmse[2]
```

```
## [1] 21.14238
```

```
test_pls_age_split$res$test$rmse[3]
```

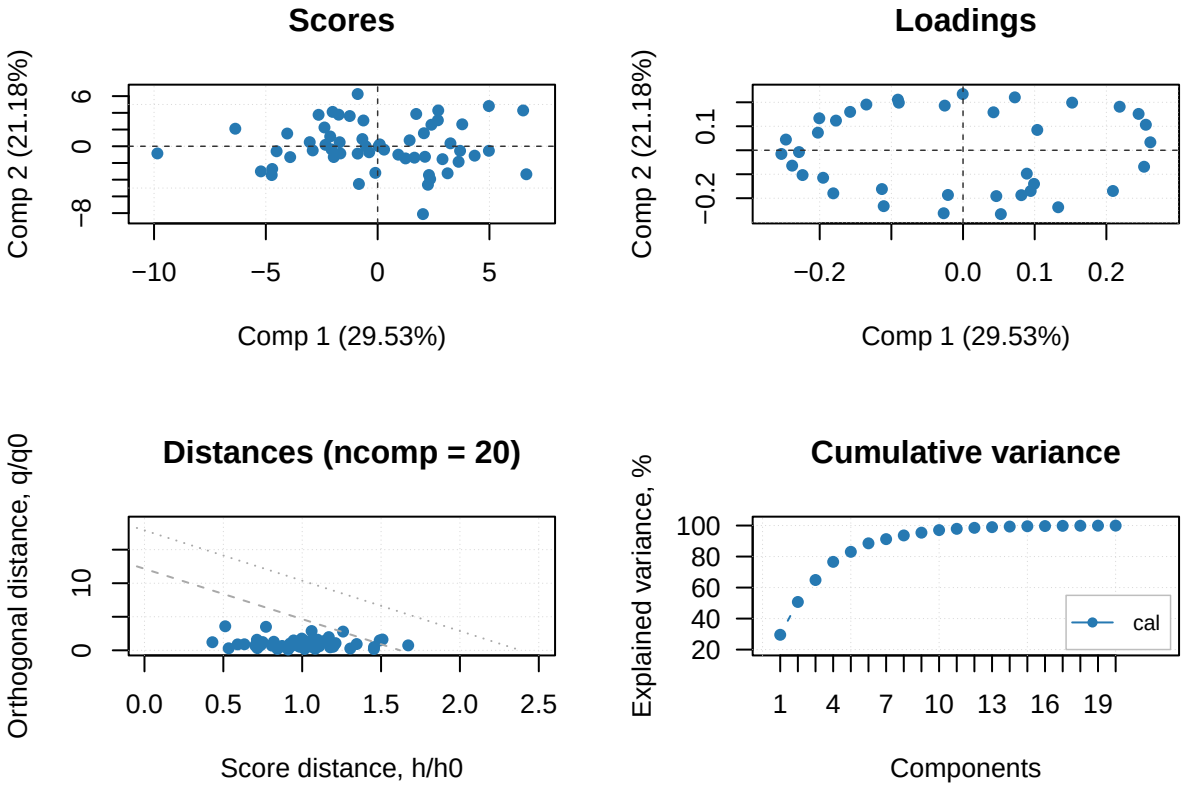
```
## [1] 14.83123
```

```
test_pls_age_split_VIP$res$cal$rmse[2]
```

```
## [1] 21.38504
```

PCR for age

```
quickproc(age_only,2,3,19)
pca_age_LPW <- pca(temp_proc[21:nrow(temp_proc)], scale = T)
plot(pca_age_LPW)
```




```
mlr_age_pcs <- pca_age_LPW$calres$scores[,1:20]
mlr_age_pcs <- cbind(mlr_age_pcs[,1:10], temp_proc[,1:11]) # extract first 10 PC's
mlr_age <- lm(data = mlr_age_pcs, read_age ~ `Comp 1` + `Comp 2` + `Comp 3` + `Comp 4` + `Comp 5`)
summary(mlr_age)
```

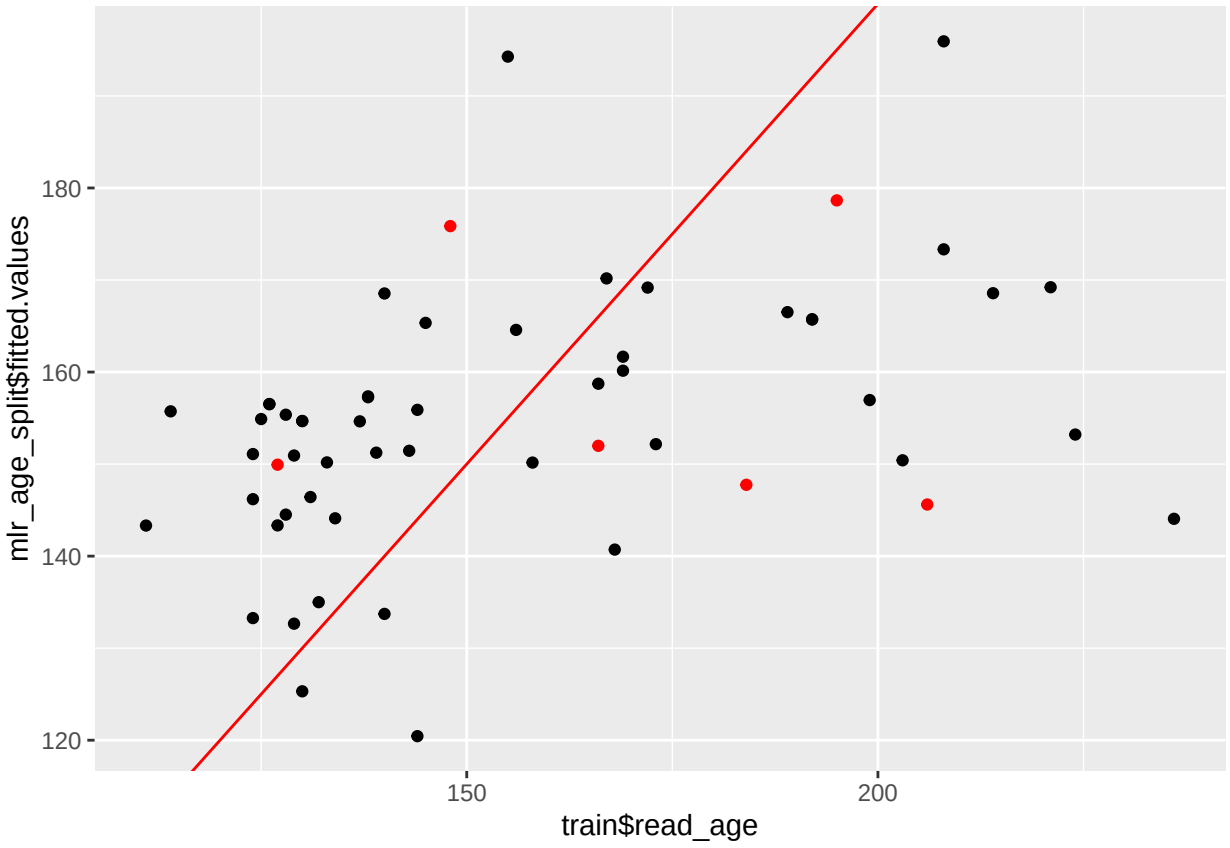
```
##
## Call:
## lm(formula = read_age ~ `Comp 1` + `Comp 2` + `Comp 3` + `Comp 4` +
##     `Comp 5`, data = mlr_age_pcs)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -41.820 -21.264  -7.735  17.378  90.137
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 156.2807     4.0967  38.148   <2e-16 ***
## `Comp 1`      1.4109     1.2503   1.128   0.2644
## `Comp 2`     -2.6409     1.4764  -1.789   0.0796 .
## `Comp 3`     -3.2187     1.8059  -1.782   0.0806 .
## `Comp 4`     -0.8587     1.9857  -0.432   0.6673
## `Comp 5`     -4.4479     2.6712  -1.665   0.1020
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 30.93 on 51 degrees of freedom
## Multiple R-squared:  0.1722, Adjusted R-squared:  0.09105
## F-statistic: 2.122 on 5 and 51 DF,  p-value: 0.0777
```

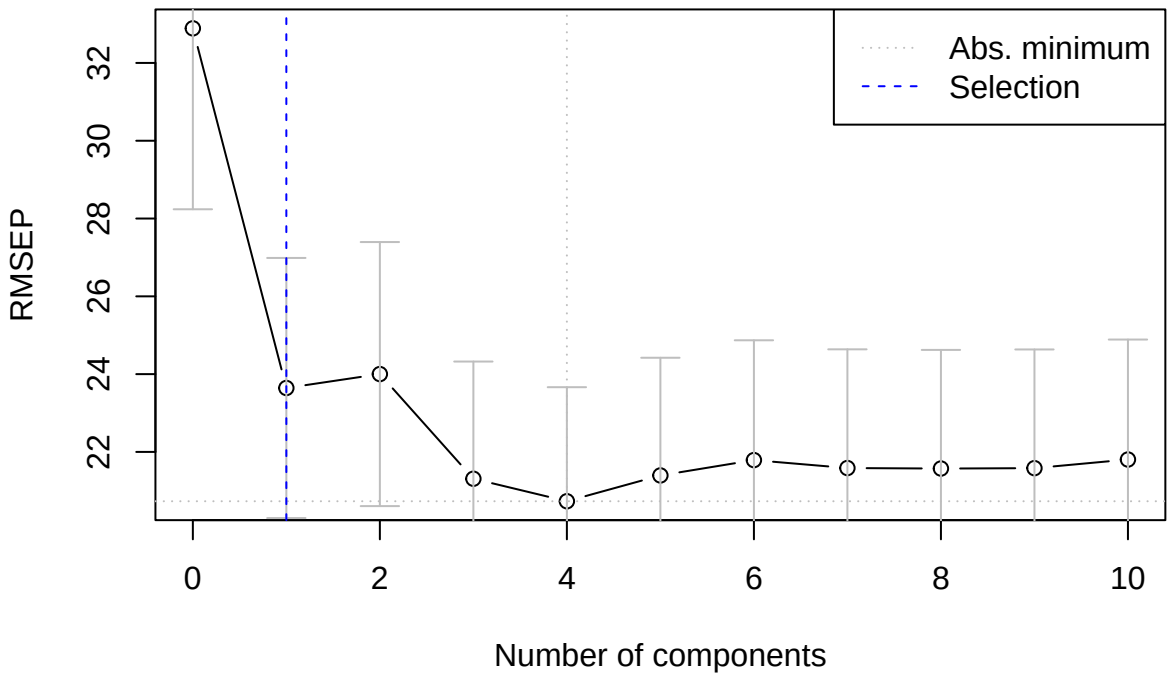
```
# MLR with split
set.seed(1)
idx <- floor(nrow(age_only) * 0.9) # 90% of indices needed for training set
train_idx <- sample(seq_len(nrow(age_only)), size = idx) # generate indices for training set
train <- mlr_age_pcs[train_idx, -c(11:20)] # wavenumber measurement columns only
test <- mlr_age_pcs[-train_idx, -c(11:20)]
mlr_age_split <- lm(data = train, read_age ~`Comp 1` + `Comp 2` + `Comp 3` + `Comp 4` + `Comp 5`)
summary(mlr_age_split)
```

```
##
## Call:
## lm(formula = read_age ~ `Comp 1` + `Comp 2` + `Comp 3` + `Comp 4` +
##     `Comp 5`, data = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -41.73 -21.14  -8.58  16.45  91.94
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 154.9436     4.3548  35.580   <2e-16 ***
## `Comp 1`      0.7844     1.3129   0.597   0.5532
## `Comp 2`     -3.2988     1.5725  -2.098   0.0416 *
## `Comp 3`     -2.8779     1.8872  -1.525   0.1343
## `Comp 4`     -0.3173     2.1300  -0.149   0.8822
## `Comp 5`     -5.3221     2.7197  -1.957   0.0566 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 30.84 on 45 degrees of freedom
## Multiple R-squared:  0.1928, Adjusted R-squared:  0.1031
## F-statistic:  2.15 on 5 and 45 DF,  p-value: 0.07666
```

```
mlr_age_split_pred <- predict(mlr_age_split,test)
ggplot() +  # fitted values vs actual, out of sample prediction in red
  geom_point(aes(train$read_age,mlr_age_split$fitted.values)) +
  geom_point(aes(test$read_age,mlr_age_split_pred),col = "red") +
  geom_abline(slope=1, intercept = 0, col = "red")
```

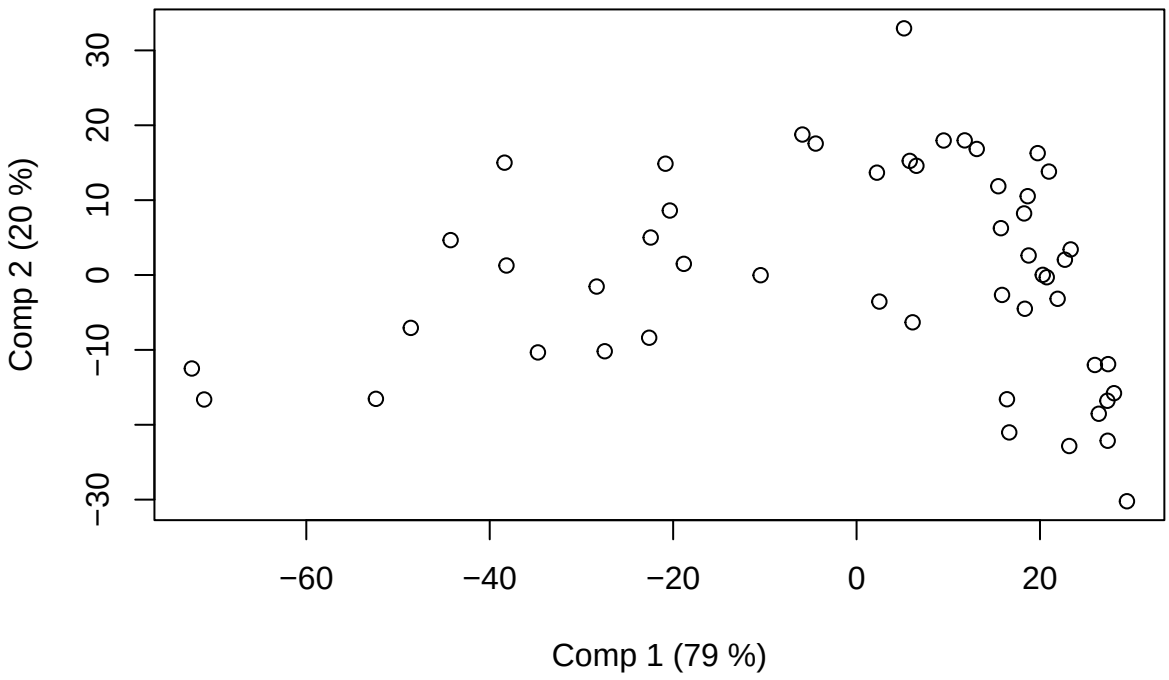



```
# Split for PCR
set.seed(1)
idx <- floor(nrow(temp_proc) * 0.9)
train_idx <- sample(seq_len(nrow(temp_proc)), size = idx)
train <- age_only[train_idx, -c(1:10,12:20)]
test <- age_only[-train_idx, -c(1:10,12:20)]

pcr_age_model <- pcr(read_age ~ .,
                     data = train,
                     scale = T,
                     validation = "CV",
                     ncomp = 10)
selectNcomp(pcr_age_model, "onesigma", plot = TRUE) # determine appropriate number of comps
```

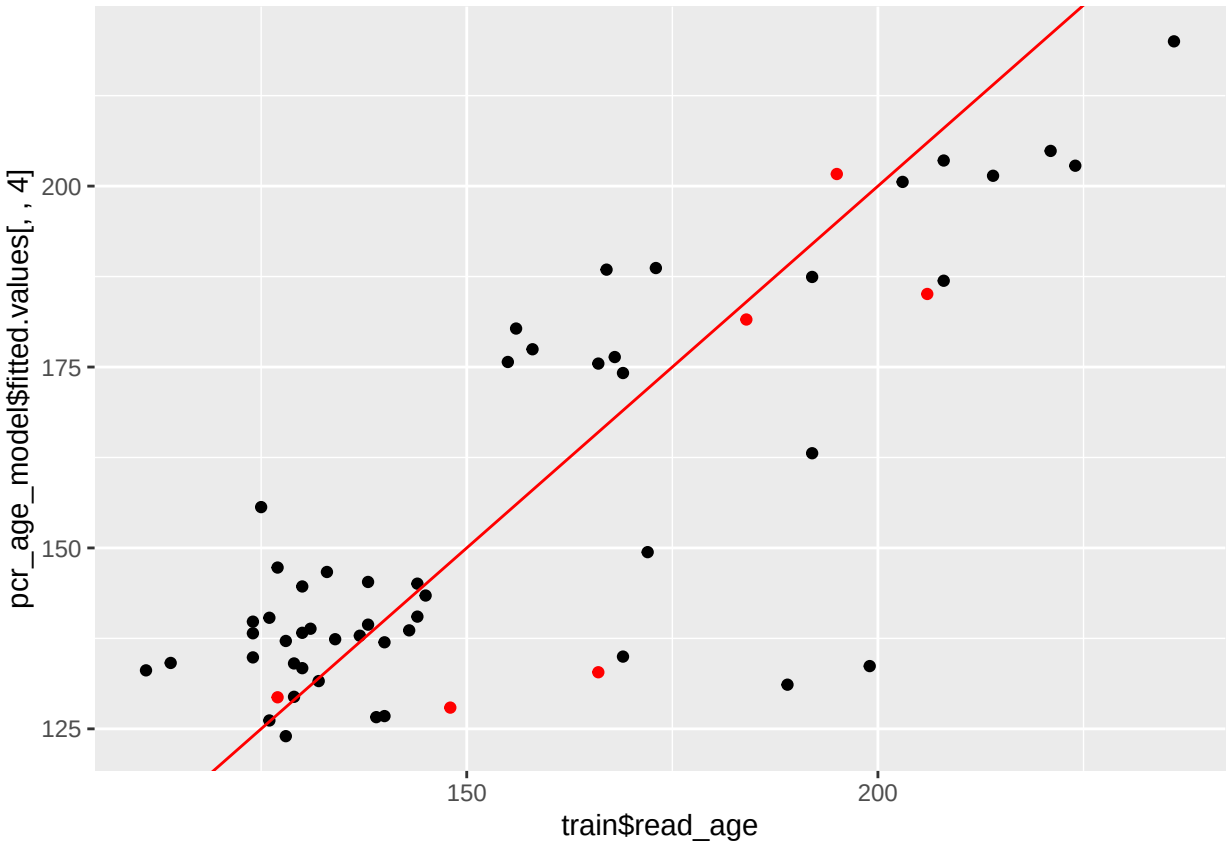



```
## [1] 1
```

```
scoreplot(pcr_age_model) # plot of first 2 PCs
```




```
pcr_age_pred <- predict(pcr_age_model, newdata = test, ncomp = 4) # fit model to test data, store fitte
ggplot() +
  geom_point(aes(x = train$read_age, y = pcr_age_model$fitted.values[, , 4])) + # extract fitted values
  geom_point(aes(x = test$read_age, y = pcr_age_pred), col = "red") +
  geom_abline(slope = 1, col = "red")
```




```
RMSEP(pcr_age_model)
```

```
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           32.89    23.64    24.00    21.31    20.73    21.40    21.79
## adjCV        32.89    23.54    23.88    21.21    20.63    21.26    21.63
##        7 comps  8 comps  9 comps  10 comps
## CV       21.59    21.57    21.58     21.81
## adjCV    21.38    21.45    21.49     22.05
```