

# 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 transreflectance stamp placed over the top of samples. Spectra were collected between 11,500 and 4,000 cm<sup>-1</sup> with a resolution of 16 cm<sup>-1</sup> 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", "tidyverse", "EMSC", "purrr", "devtools", "hyperSpec", "prospectr", "data.table", "m
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("simplespec")) {
  remotes::install_github("philipp-baumann/simplespec")
  library("simplespec")
}
# 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 taken)

## [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$session_title %in% c("NIR_LPW202001202_otoliths", "NIR_LPW202001202_otoliths_rescan_1")]
meta_LPW <- meta_LPW[meta_LPW$file_name != "", ] # remove any rows that are missing a file_name in the session

# 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$file_path)
exists <- as.vector(lapply(Opusfiles, file.exists)) # check that I have all my files or else I get an error
meta_LPW$exists <- exists
meta_LPW1 <- meta_LPW[meta_LPW$exists == "TRUE", ] # filter the file list and data by otoliths with species
Opusfiles <- as.vector(meta_LPW1$file_path) # I repeated this and wrote over it so I wouldn't have extra
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 metadata
rm(data_list)
rm(file)
SPCfiles_nooffset <- suppressWarnings(lapply(Opusfiles, opusreader2::read_opus)) # Read in all files in parallel

# 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 was used
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 we can use this to identify the specimen

## [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_value)
species <- lapply(SPCfiles_nooffset, function(x) x$lab_and_process_param_raw$parameters$FC2$parameter_value)
instrument <- lapply(SPCfiles_nooffset, function(x) x[[1]]$instrument_ref$parameters$INS$parameter_value)

```

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

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 a warning
}
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.convert = TRUE)

##                         try$file_path_10
##                               <char>
## 1: PACIFIC_COD_LPW202001202_1_0A1.0

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_0A1.0"

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

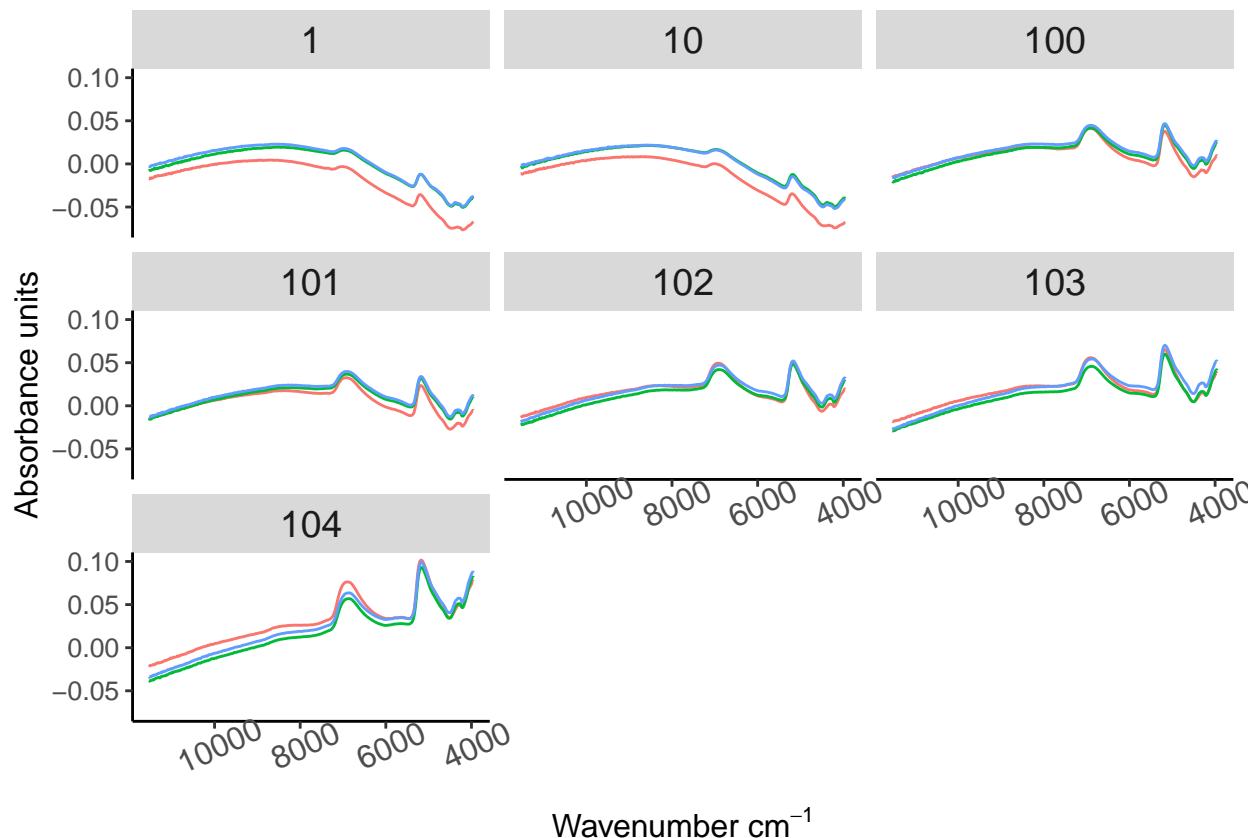
```
df <- as.data.frame(do.call(rbind, spectra))
dfmeta_LPW <- dplyr::left_join(meta_LPW, df, by = c("file_name", "file_path"))
dfmeta_LPW <- dfmeta_LPW %>% select(-c("exists", "instrument"))
rm(meta_LPW, df, spectra, SPCfiles_nooffset)
colnames(dfmeta_LPW) <- as.character(colnames(dfmeta_LPW))
dfmeta_longLPW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

```

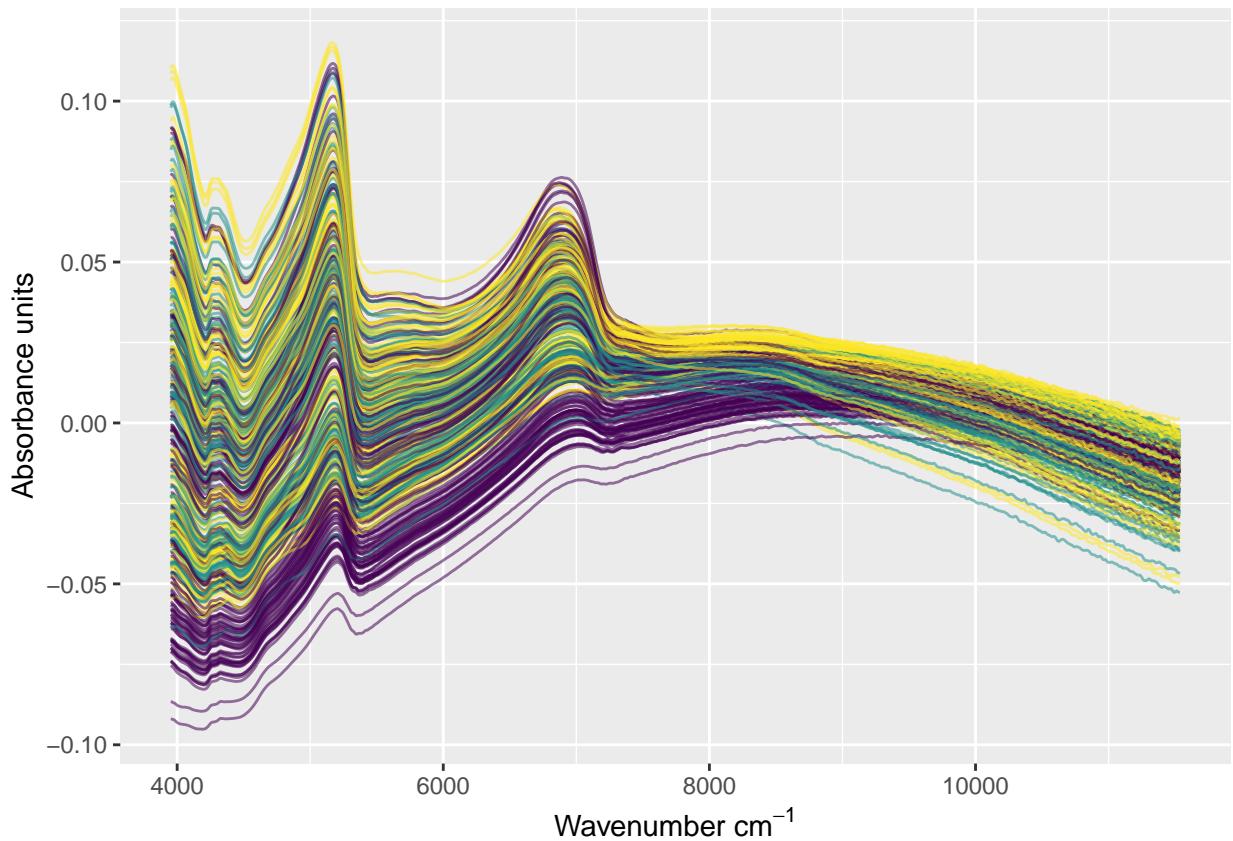


```
# Comparison between scans & PCA of all scans
```

```

# 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)

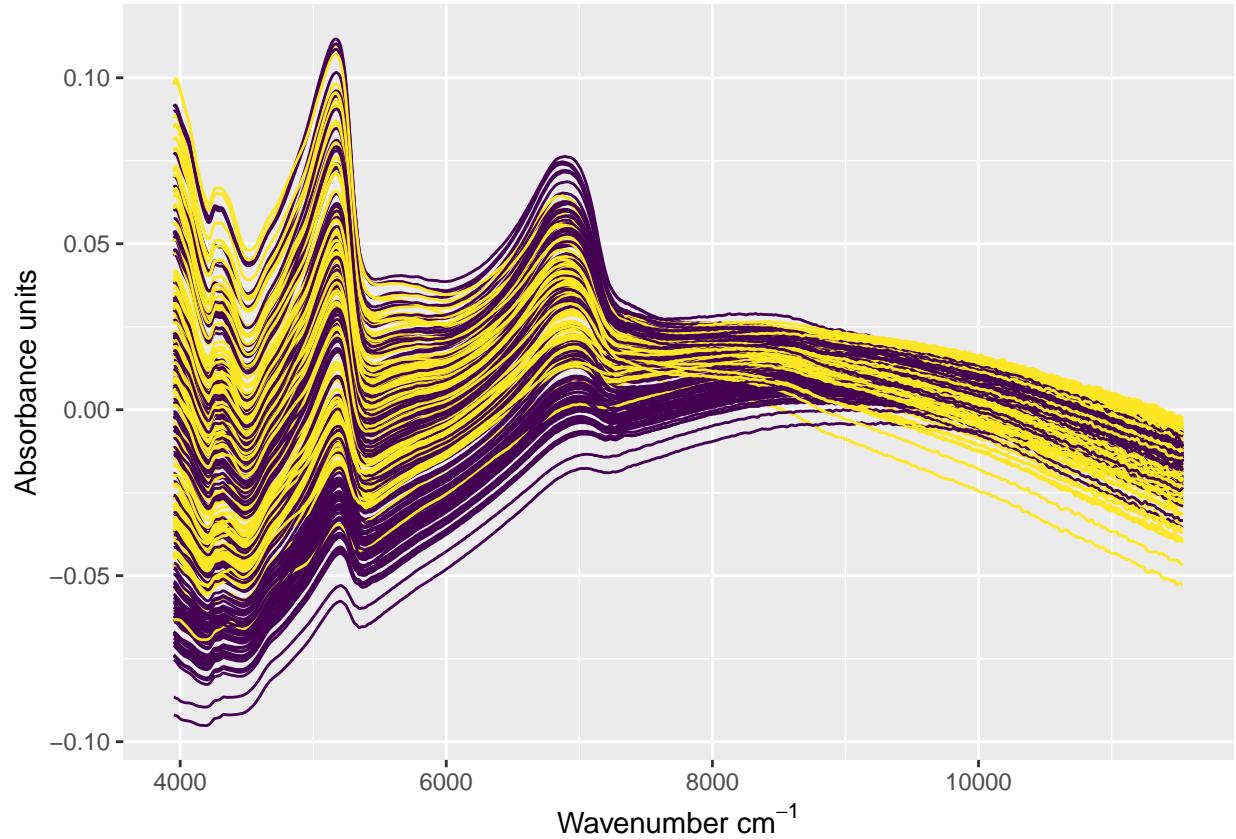
```



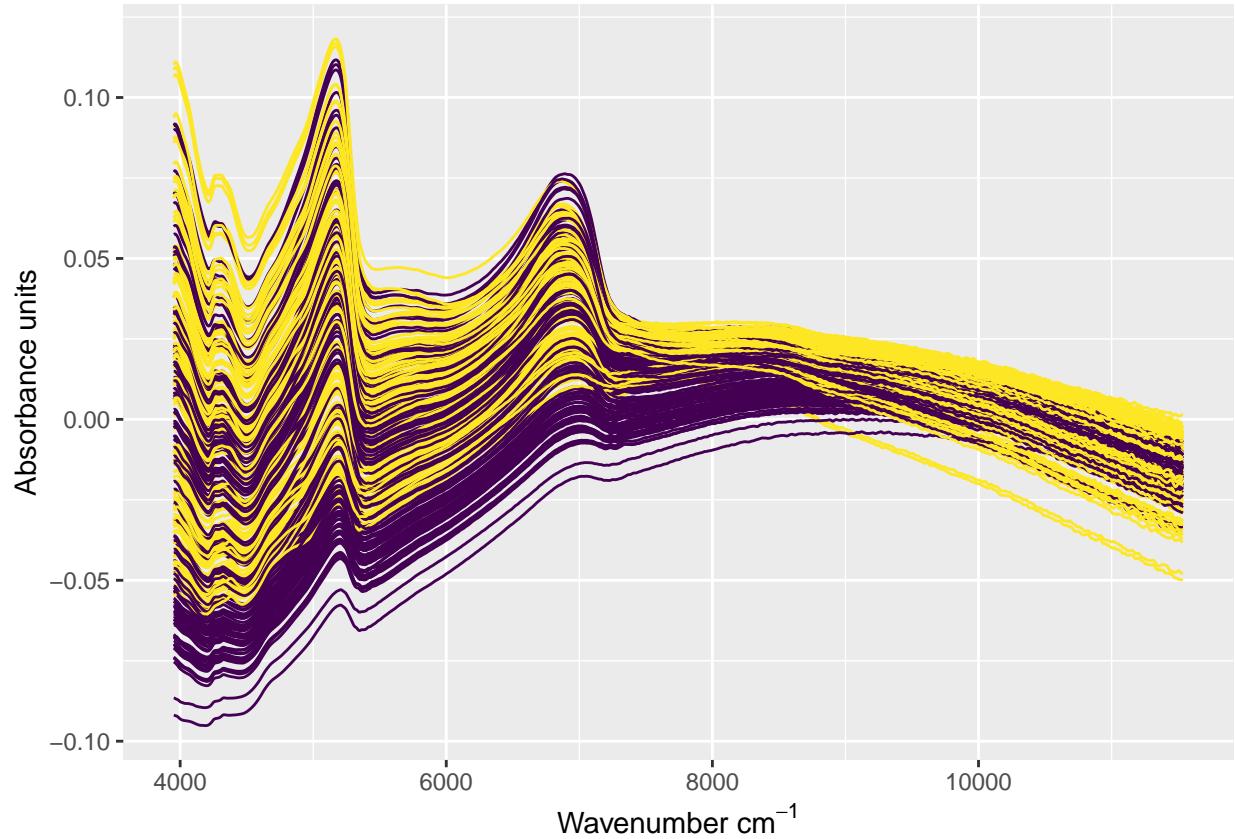
```

# 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 = value))
  labs(y = "Absorbance units", x = expression(paste("Wavenumber ", cm^-1))) +
  theme(legend.position = "none") +
  scale_color_viridis(discrete = T)

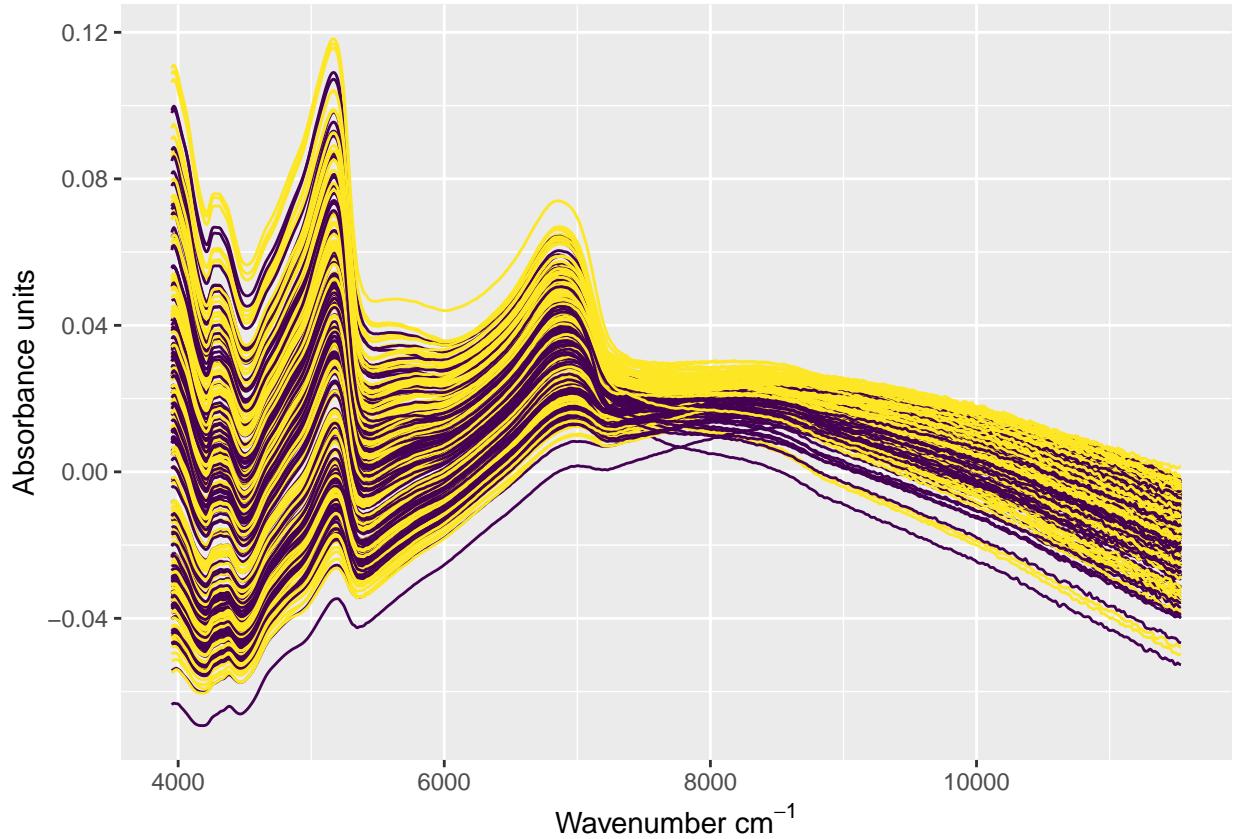
```



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

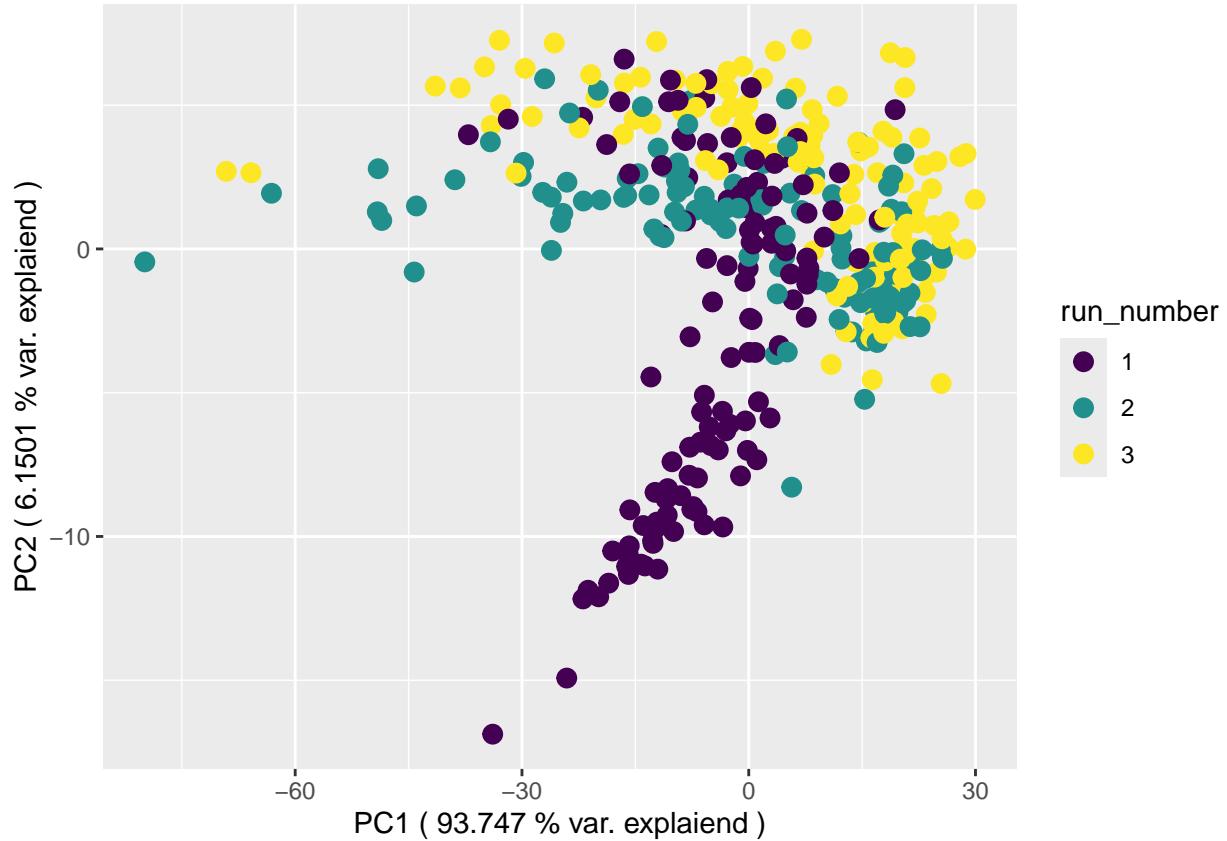


```

# 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)
dfmeta_LPW <- cbind(pc2 = pca_all_LPW$calres$scores[,2], dfmeta_LPW) # extract scores for PC2
dfmeta_LPW <- cbind(pc1 = pca_all_LPW$calres$scores[,1], dfmeta_LPW) # extract scores for PC1

# plot PC1 & PC2, colored by run_number
ggplot(dfmeta_LPW) +
  geom_point(aes(x = pc1, y = pc2, color = 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[, 1:22])
scan_avg <- bind_cols(scan_avg, (scan_2[, 23:ncol(scan_2)] + scan_3[, 23: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.

```

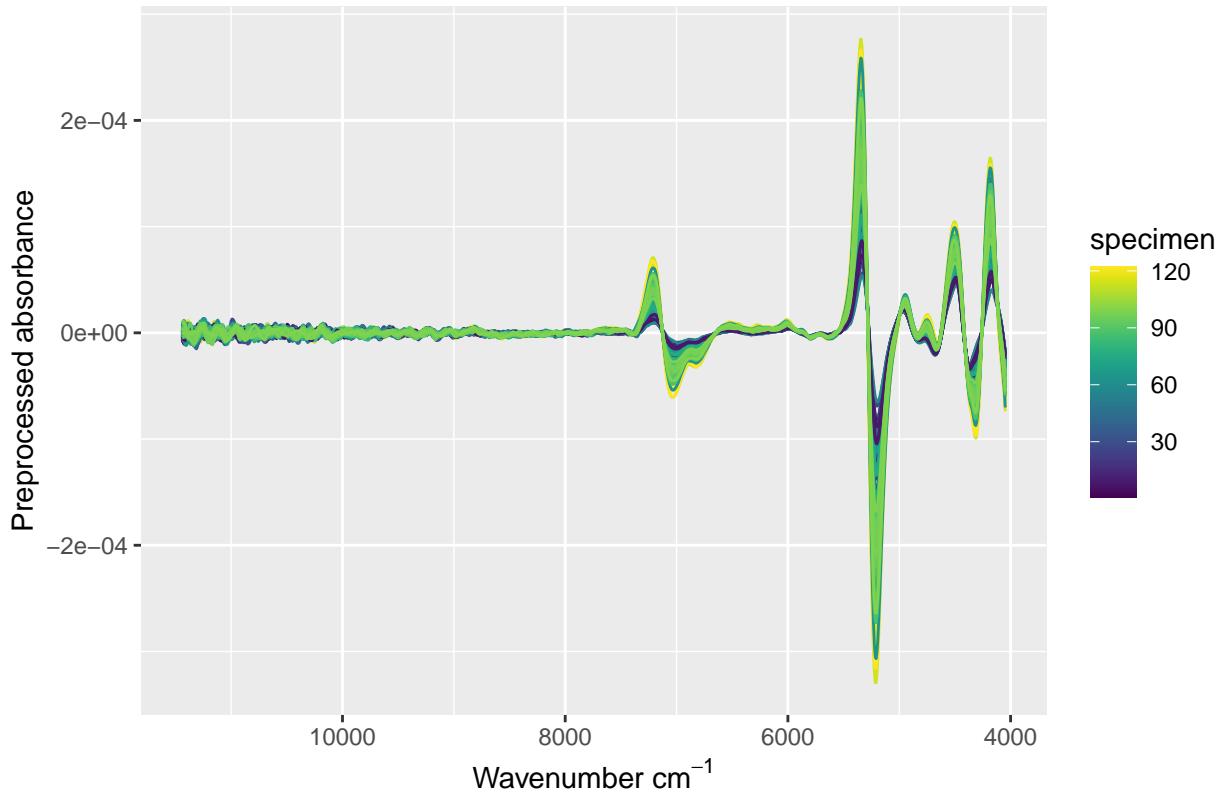
spec.fig <- function(mydf, color) { # ggplot function to swap out color
  ggplot(mydf) +
    geom_path(aes(x = name, y = value, color = {{ color }}), group = file_name)) +
    scale_x_reverse() +
    scale_color_viridis() +
    labs(y = "Preprocessed absorbance", x = expression(paste("Wavenumber ", cm^-1)))
  #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")
  #)
}

sg_plotting <- function(color, m, p, w) {
  dftempproc <- as.data.frame(
    cbind(scan_avg[, c(1:22)],
          savitzkyGolay(scan_avg[, 23:length(scan_avg)],
                        m = m, p = p, w = w)
    ))
  dftempproc_long <- tidyr::pivot_longer(dftempproc, cols = c(23: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 }}))
}

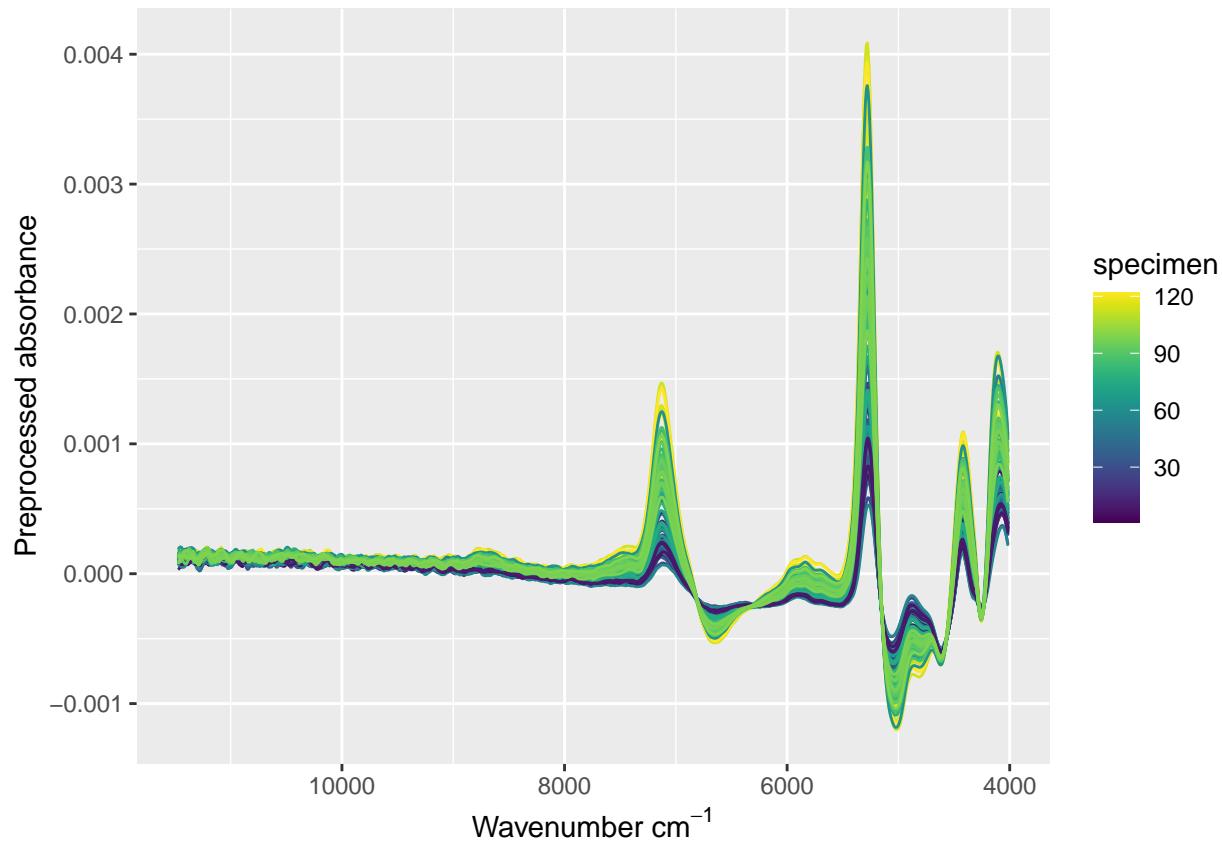
sg_plotting(specimen, 2, 3, 25)

```

diff = 2 poly = 3 window = 25

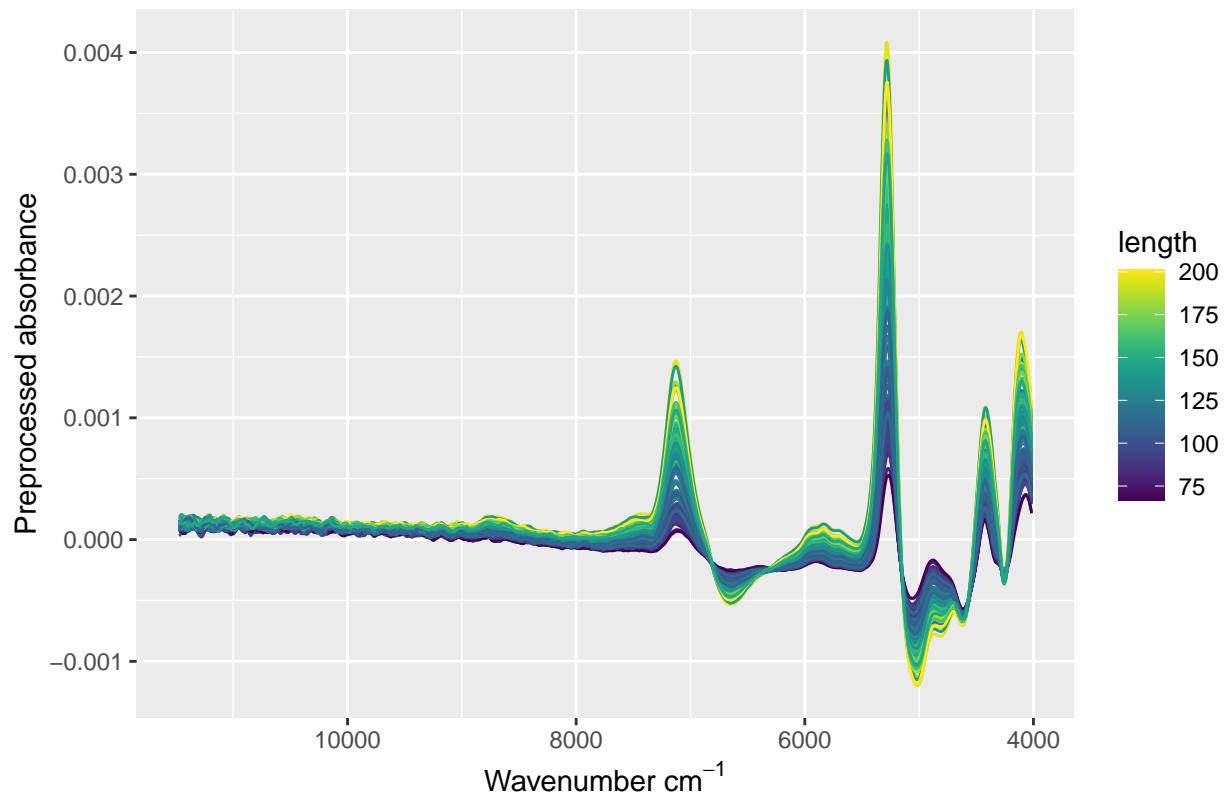


```
quickproc <- function(df, m, p, w){  
  temp_proc <- as.data.frame(  
    cbind({{df}}[, c(1:22)],  
          savitzkyGolay({{df}}[, 23:length({{df}})],  
                      m = m, p = p, w = w)  
  ))  
  temp_proc_long <- tidyr::pivot_longer(temp_proc, cols = c(23:length(temp_proc)))  
  temp_proc_long$name <- as.numeric(as.character(temp_proc_long$name))  
}  
  
quickproc(scan_avg, 1, 2, 17)  
spec.fig(temp_proc_long, specimen)
```



```
sg_plotting(length,1,2,17)
```

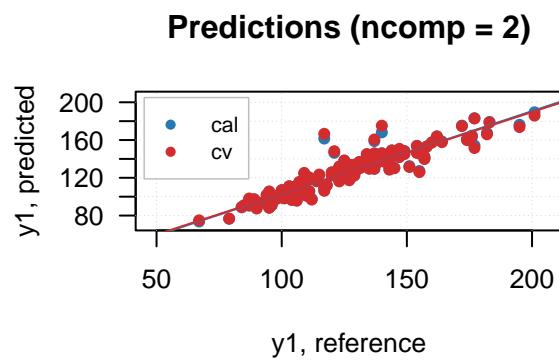
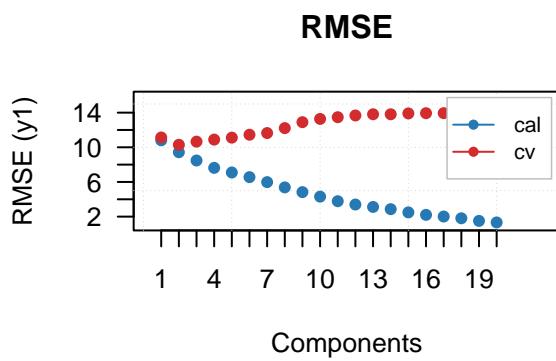
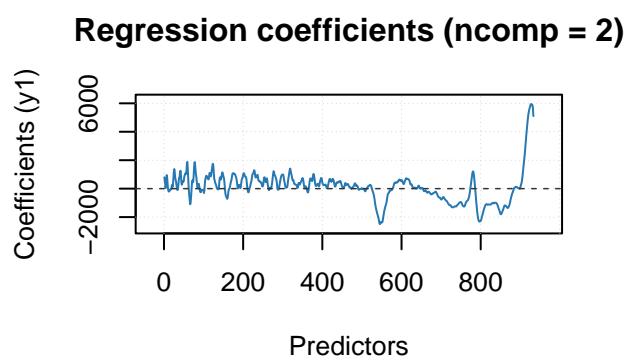
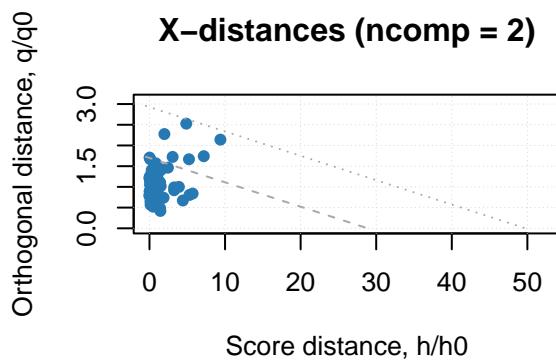
diff = 1 poly = 2 window = 17



## Variable/Wavelength Selection & PLS

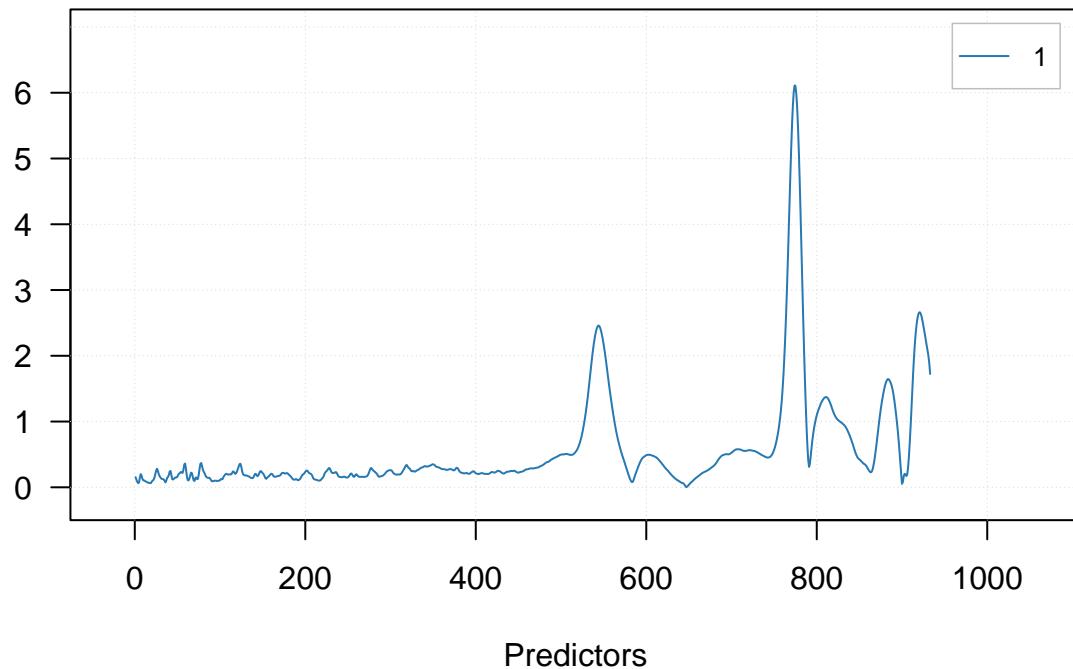
Using temp\_proc df produced from quickproc() function above. VIP score to select wavelengths that are most informative to model

```
# temp_proc df
test_pls <- pls(temp_proc[, 23:ncol(temp_proc)], temp_proc[, 4],
  scale = FALSE, center = TRUE, info = "Length Prediction Model",
  cv = 1
)
plot(test_pls)
```



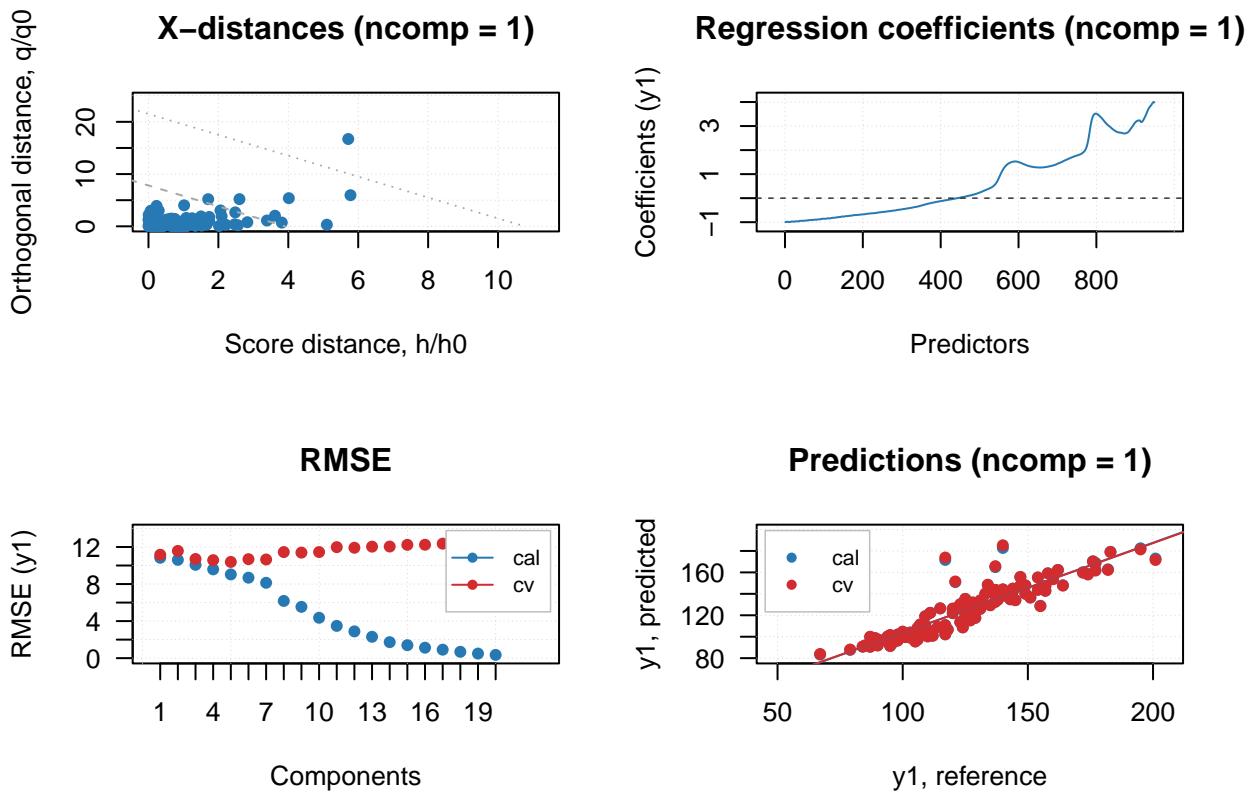
```
# pls w/ VIP > 1.5
test_vip <- vipscores(test_pls)
plotVIPScores(test_pls)
```

## VIP scores (ncomp = 2)



```
test_waves <- names(subset(test_vip[,1], test_vip[,1] > 1.5))
all_waves <- names(temp_proc[,23:length(temp_proc)])
bad_waves <- all_waves[!(all_waves %in% test_waves)]

# unprocessed df
test_unproc_pls <- pls(scan_avg[, 23:ncol(scan_avg)], scan_avg[, 4],
  scale = FALSE, center = TRUE, info = "Length Prediction Model",
  cv = 1
)
plot(test_unproc_pls)
```



```
# test_pls_VIP <- pls(temp_proc[, 23:ncol(temp_proc)], temp_proc[, 4],
#   scale = FALSE, center = TRUE, info = "Length Prediction Model",
#   cv = 1, exclcols = bad_waves
# )
# plot(test_pls_VIP)

# 90% train, 10% test
# 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(1:3,5:33)]
# test <- temp_proc[-train_idx, -c(1:3,5:33)]
# ###
# test_pls_split <- pls(train[,-c(1)], train[,1],
#   scale = T, center = T,
#   info = "Length Prediction Model", cv = 1,
#   x.test = test[,-c(1)], y.test = test[,1]
# )
# plot(test_pls_split)
# plot(test_pls_VIP)
# plot(test_pls)
```

## MLR & PCA

```
pca_avg_LPW <- pca(scan_avg[23:nrow(scan_avg)], scale = T)
mlr_pcs <- pca_avg_LPW$calres$scores[,1:20]
mlr_df <- cbind(mlr_pcs[,1:5], scan_avg[,1:4])

# split into test and train datasets
set.seed(1)
idx <- floor(nrow(scan_avg) * 0.9)
train_idx <- sample(seq_len(nrow(scan_avg)), size = idx)
train <- scan_avg[train_idx, -c(1:2,3,5:33)]
test <- scan_avg[-train_idx, -c(1:2,3,5:33)]

library(pls)

##
## Attaching package: 'pls'

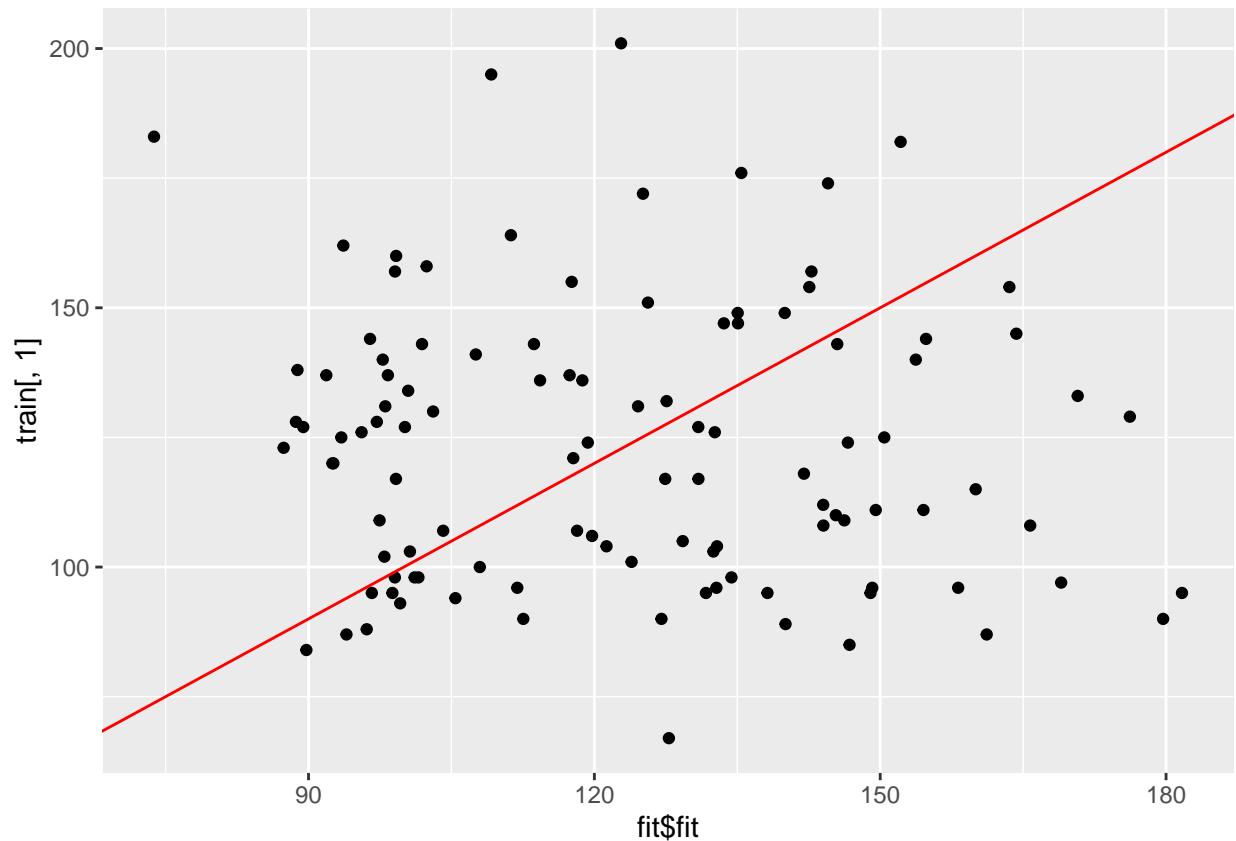
## The following object is masked from 'package:stats':
##      loadings

## The following object is masked from 'package:mdatools':
##      crossval

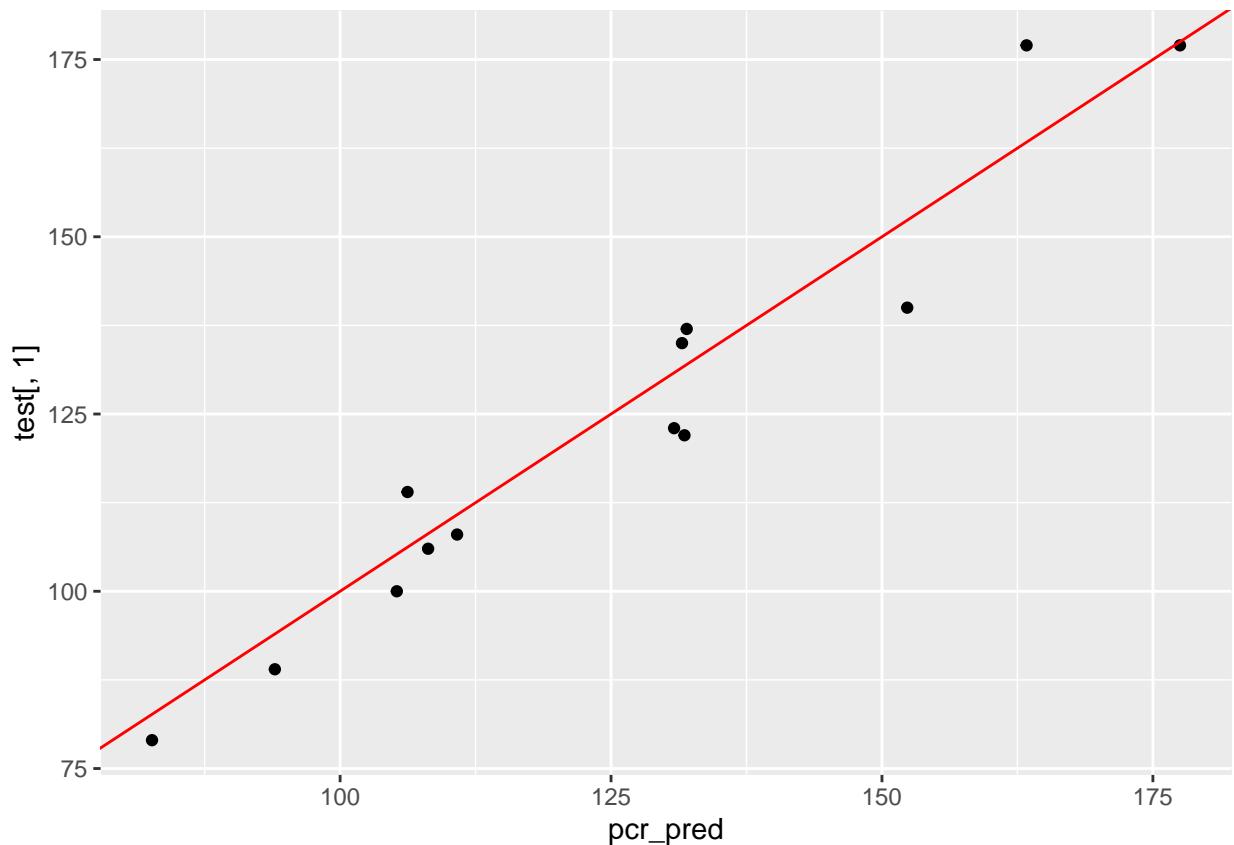
## The following object is masked from 'package:prospectr':
##      msc

pcr_model <- pcr(length ~ .,
                    data = train,
                    scale = T,
                    validation = "CV",
                    ncomp = 10)

fit <- as.data.frame(pcr_model$fitted.values[, , 3])
fit <- fit %>%
  arrange(as.numeric(row.names(.)))
names(fit)[1] <- "fit"
ggplot() + geom_point(aes(x = fit$fit, y = train[, 1])) +
  geom_abline(slope = 1, col = "red")
```



```
pcr_pred <- predict(pcr_model, newdata=test, ncomp = 3)
ggplot() + geom_point(aes(x = pcr_pred, y = test[,1])) +
  geom_abline(slope = 1, col = "red")
```

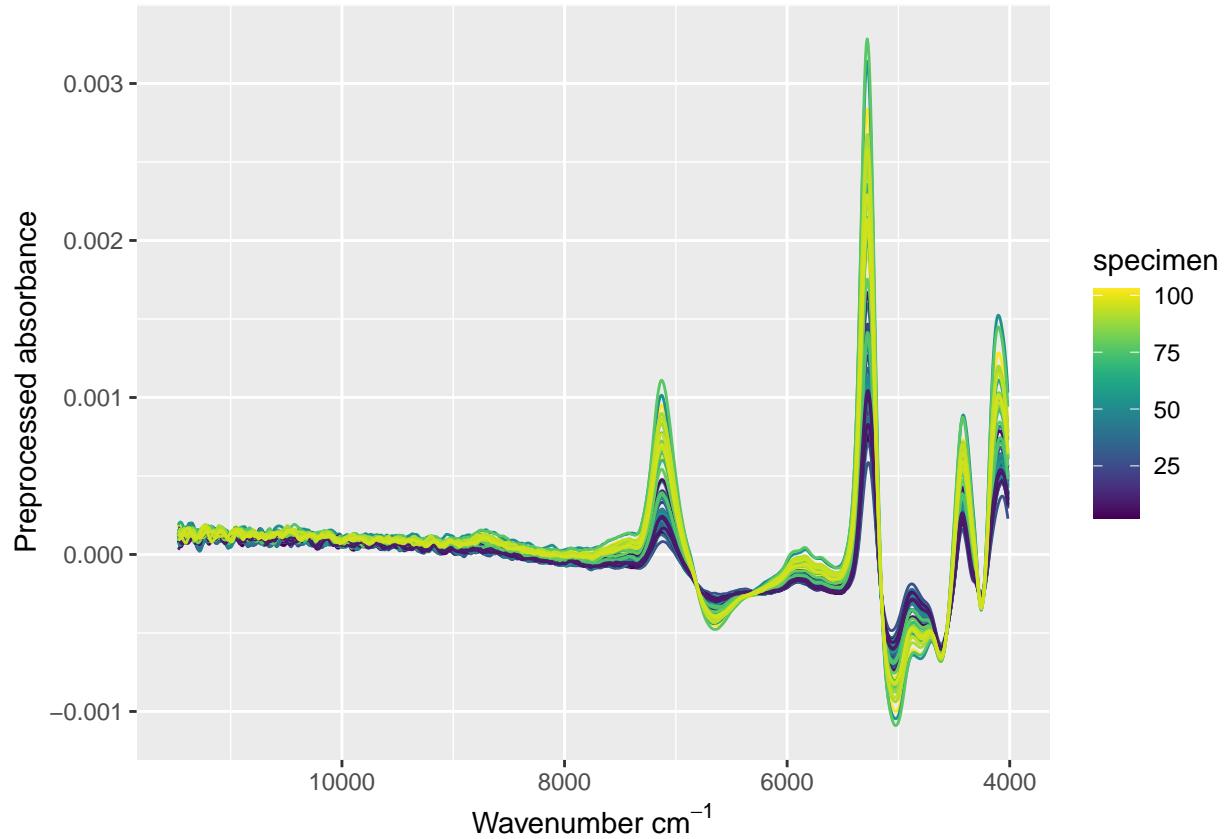


```
# Remove specimens missing age data, pls for age
```

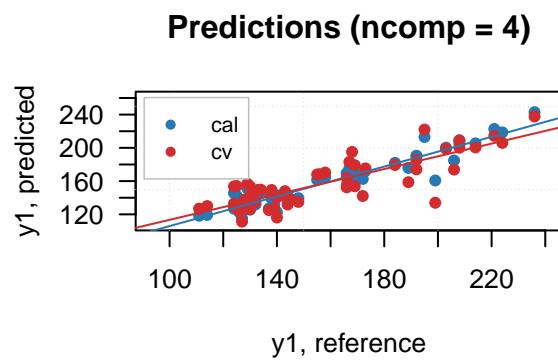
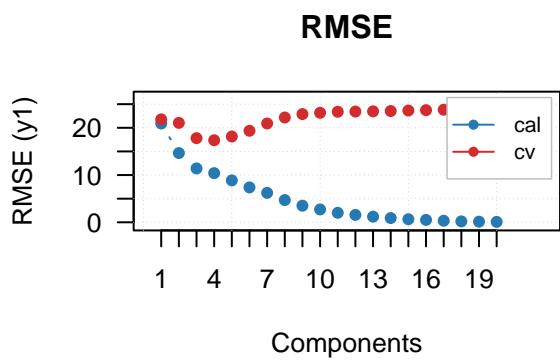
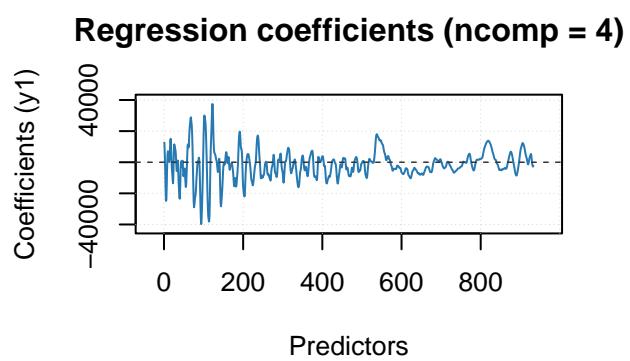
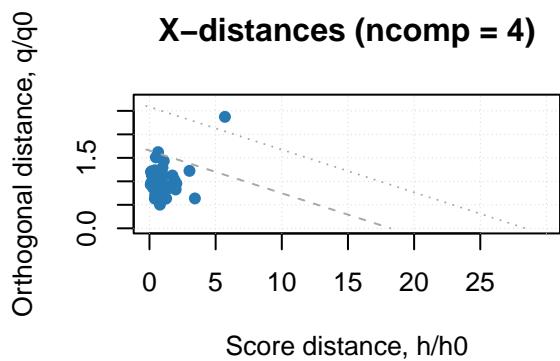
```
# df with only aged specimens
age_only <- scan_avg %>% dplyr::filter(is.na(read_age) == FALSE)

# generate preprocessed df of ages, temp_proc & temp_proc_long
quickproc(age_only,1,2,17)

spec.fig(temp_proc_long,specimen)
```

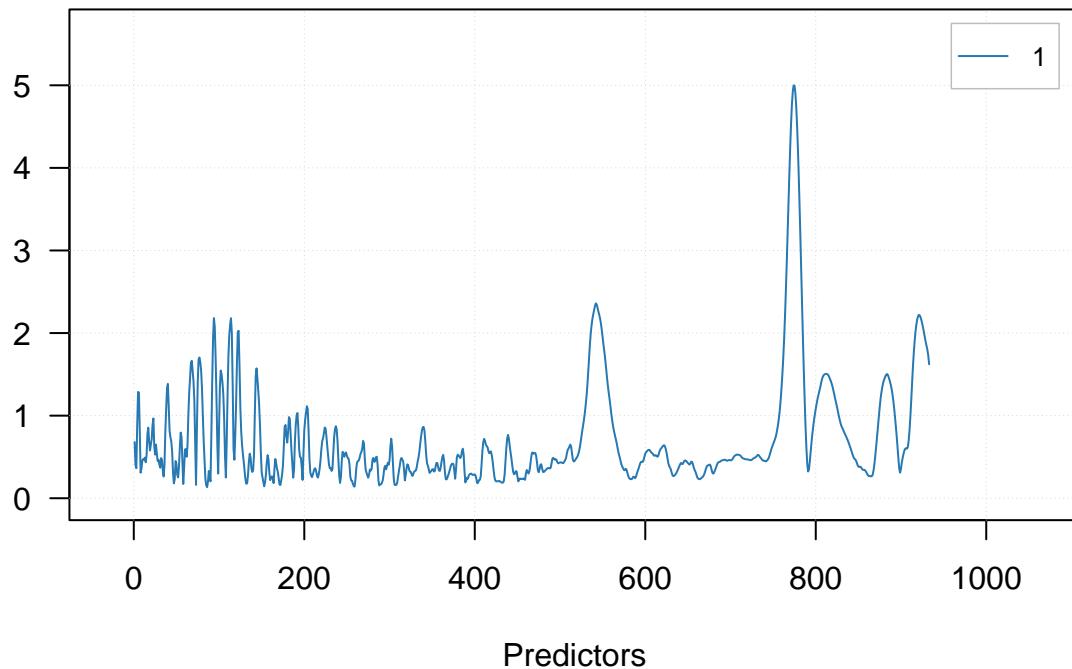


```
# test pls, no variable selection
test_pls_age <- pls(temp_proc[, 23:ncol(temp_proc)], temp_proc[, 13],
  scale = FALSE, center = TRUE, info = "Age Prediction Model",
  cv = 1
)
plot(test_pls_age)
```



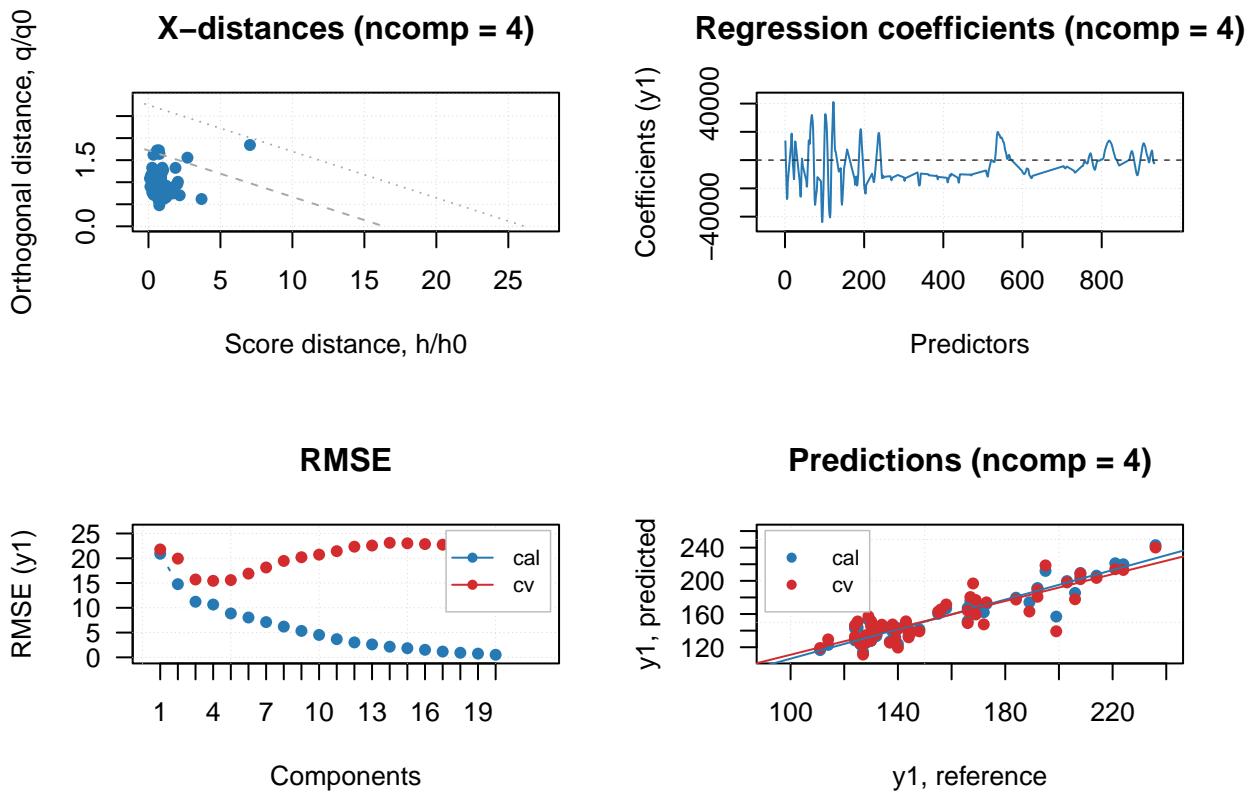
```
# pls with VIP > 0.5
test_vip_age <- vipscores(test_pls_age)
plotVIPScores(test_pls_age)
```

### VIP scores (ncomp = 4)



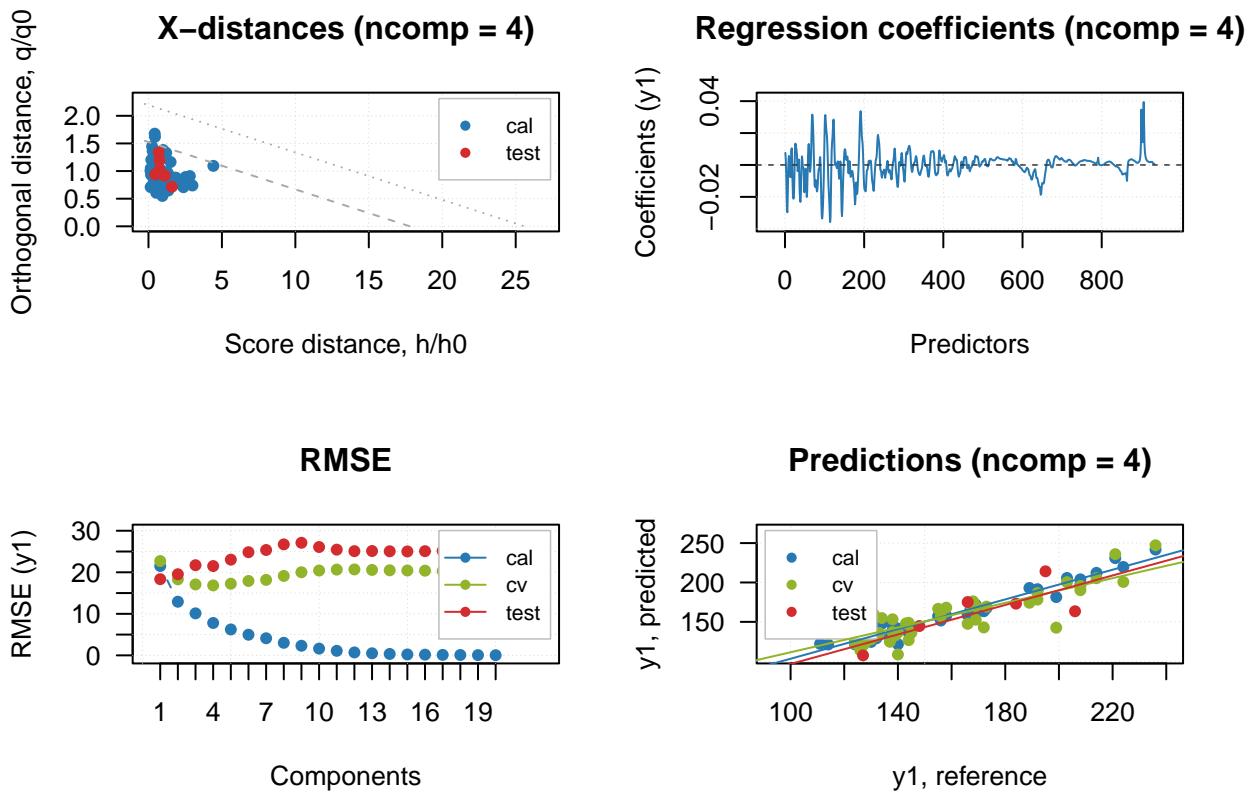
```
test_waves <- names(subset(test_vip_age[,1], test_vip_age[,1] > .5))
all_waves <- names(temp_proc[,23:length(temp_proc)])
bad_waves <- all_waves[!(all_waves %in% test_waves)]

test_pls_age_VIP <- pls(temp_proc[, 23:ncol(temp_proc)], temp_proc[, 13],
  scale = FALSE, center = TRUE, info = "Age Prediction Model",
  cv = 1, exclcols = bad_waves
)
plot(test_pls_age_VIP)
```

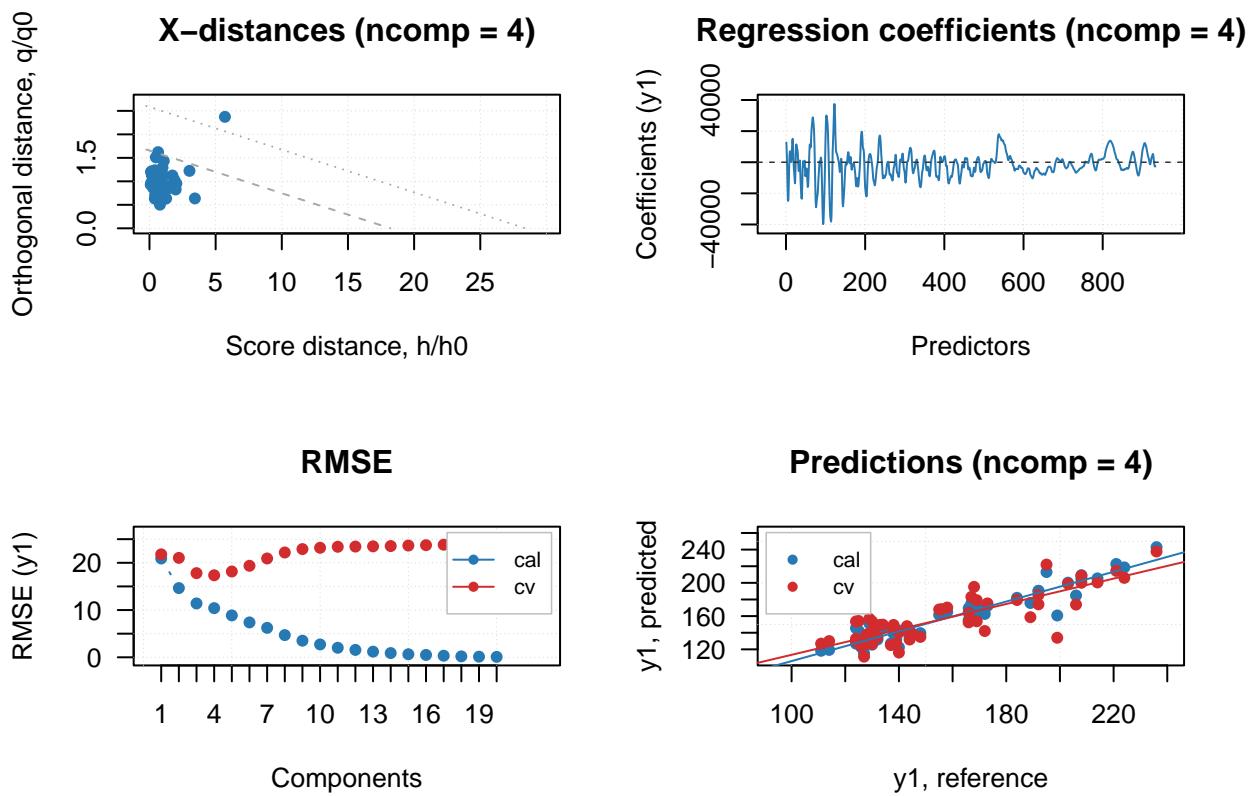


```
# 90/10 split for PLS, no VIP
set.seed(1)
idx <- floor(nrow(temp_proc) * 0.9)
train_idx <- sample(seq_len(nrow(temp_proc)), size = idx)
train <- temp_proc[train_idx, -c(1:12, 14:22)]
test <- temp_proc[-train_idx, -c(1:12, 14:22)]

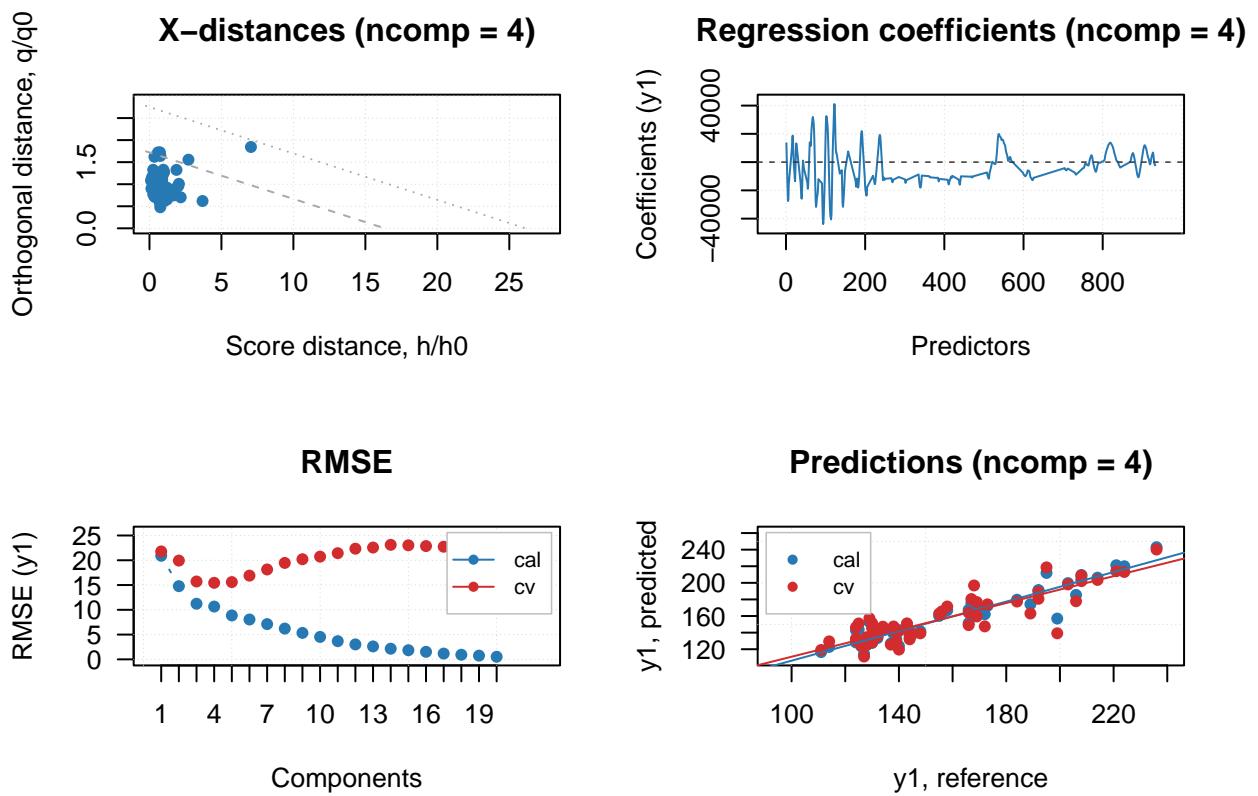
test_pls_age_split <- pls(train[,-1], train[,1],
  scale = T, center = T,
  info = "Age Prediction Model", cv = 1,
  x.test = test[,-1], y.test = test[,1])
)
plot(test_pls_age_split)
```



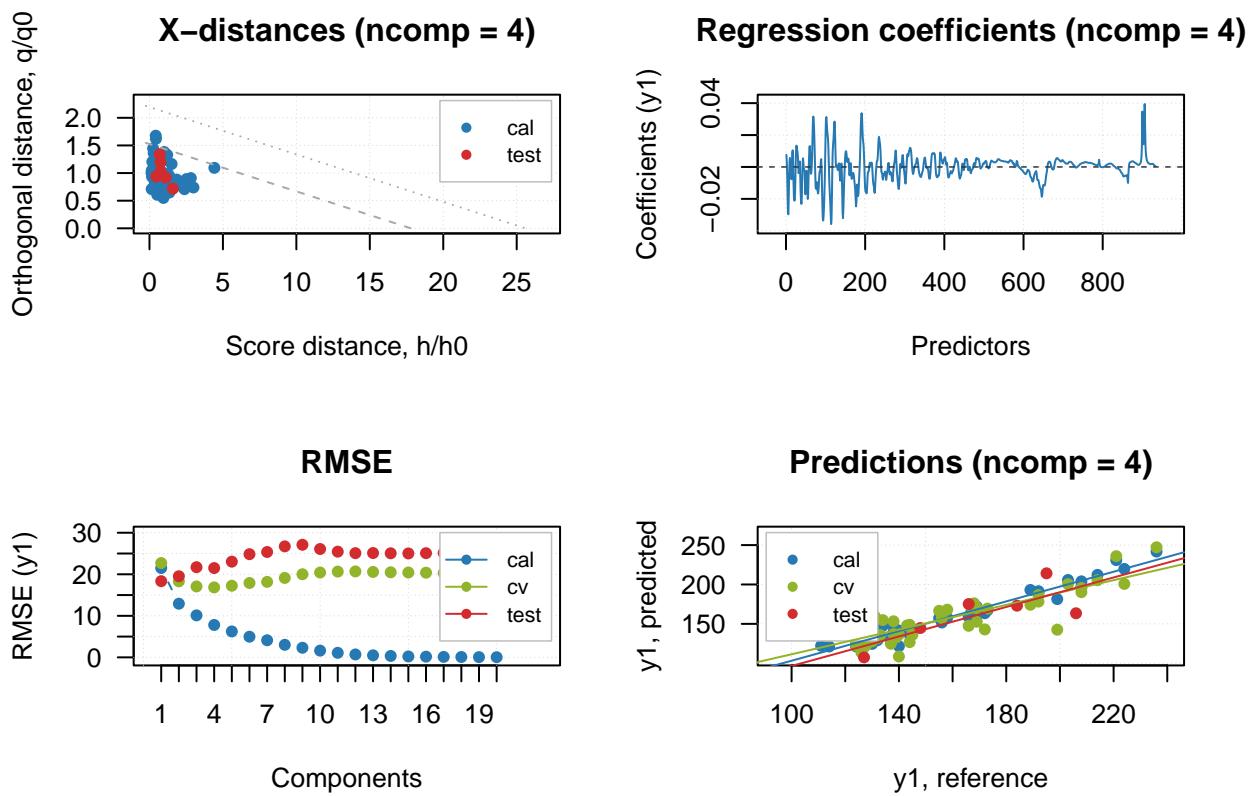
```
# 90/10 split for PLS, VIP > 0.5 (not working???)  
test_pls_age_split_VIP <- pls(train[,-1], train[,1],  
  scale = T, center = T,  
  info = "Age Prediction Model", cv = 1,  
  exclcols = bad_waves  
)  
  
plot(test_pls_age)
```



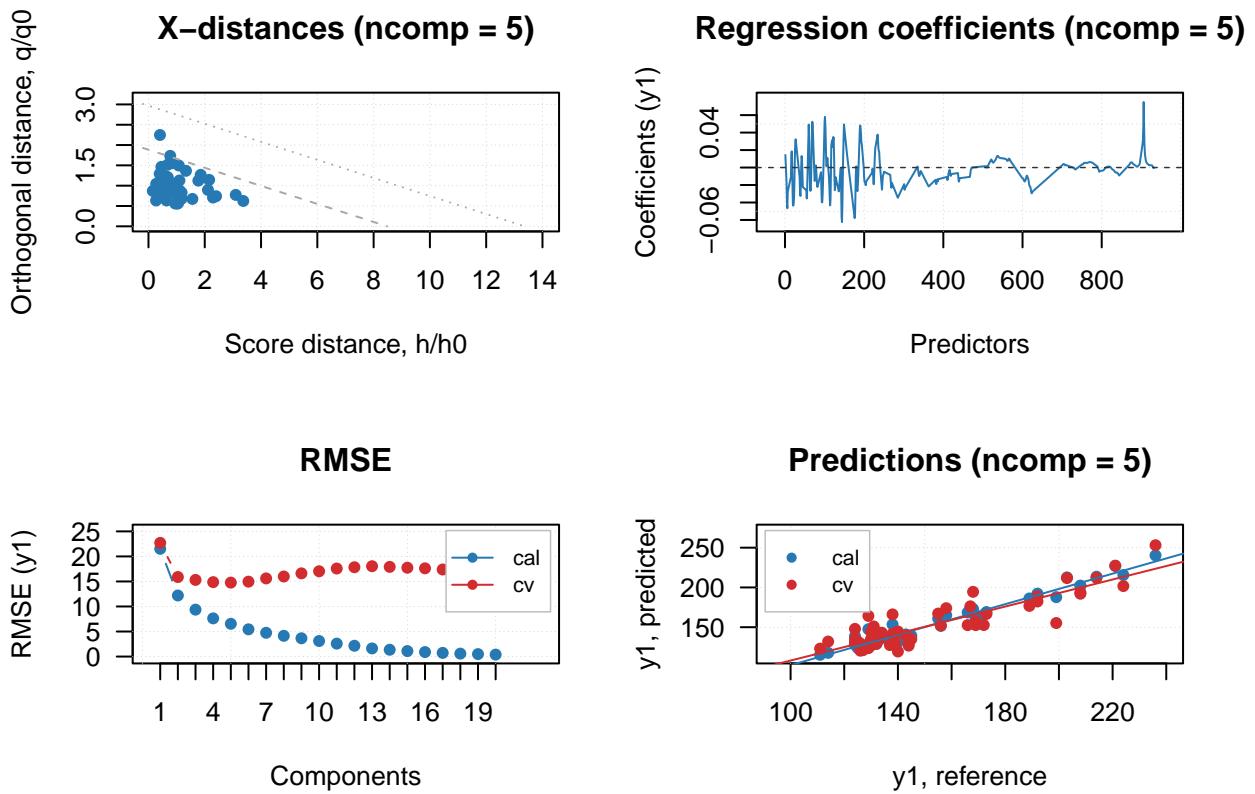
```
plot(test_pls_age_VIP)
```



```
plot(test_pls_age_split)
```



```
plot(test_pls_age_split_VIP)
```



```
test_pls_age$res$cal$rmse[1]
```

```
## [1] 20.90808
```

```
test_pls_age_VIP$res$cal$rmse[1]
```

```
## [1] 20.90652
```

```
test_pls_age_split$res$cal$rmse[4]
```

```
## [1] 7.799148
```

```
test_pls_age_split_VIP$res$cal$rmse[4]
```

```
## [1] 7.626065
```

PCR for age

```

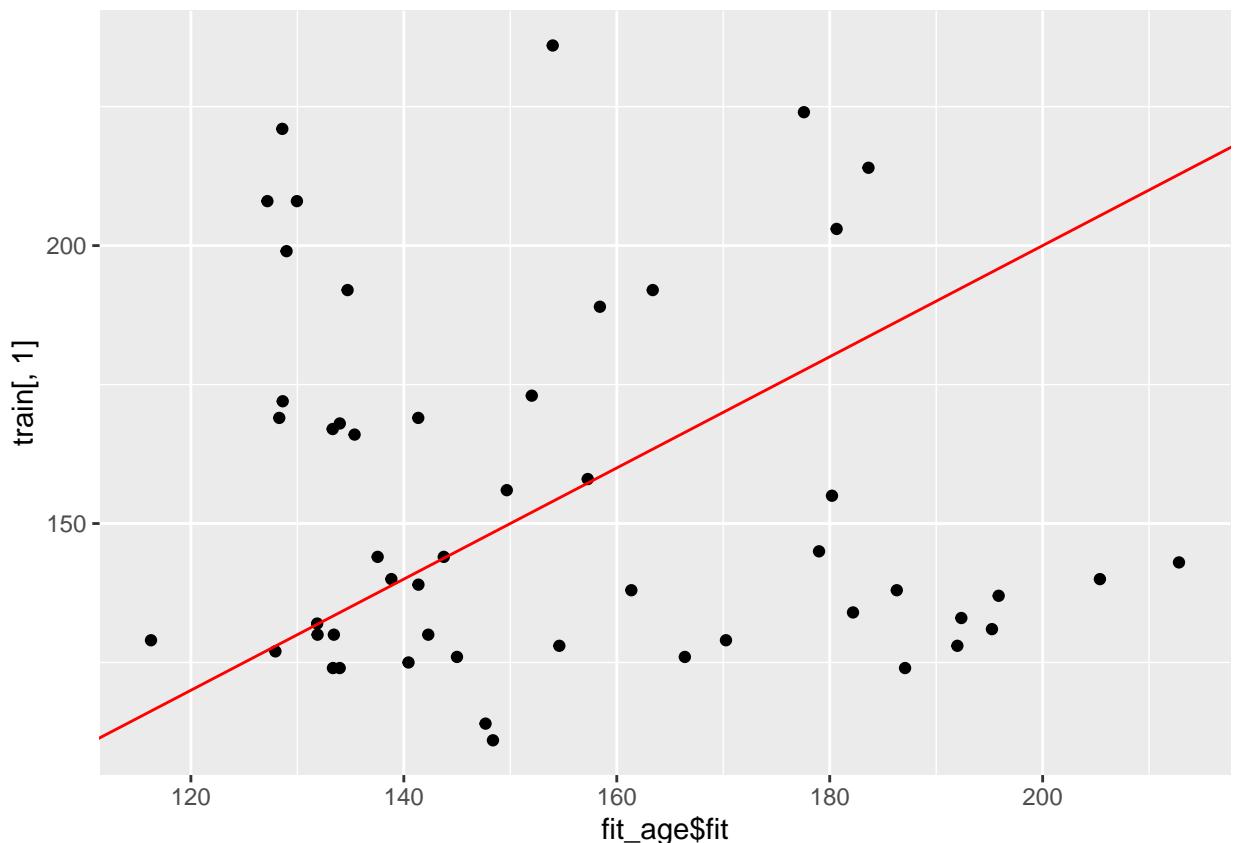
pca_age_LPW <- pca(temp_proc[23:nrow(temp_proc)], scale = T)
mlr_age_pcs <- pca_age_LPW$calres$scores[,1:20]
mlr_age_df <- cbind(mlr_age_pcs[,1:10], temp_proc[,3:13])

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(1:12,14:33)]
test <- temp_proc[-train_idx, -c(1:12,14:33)]

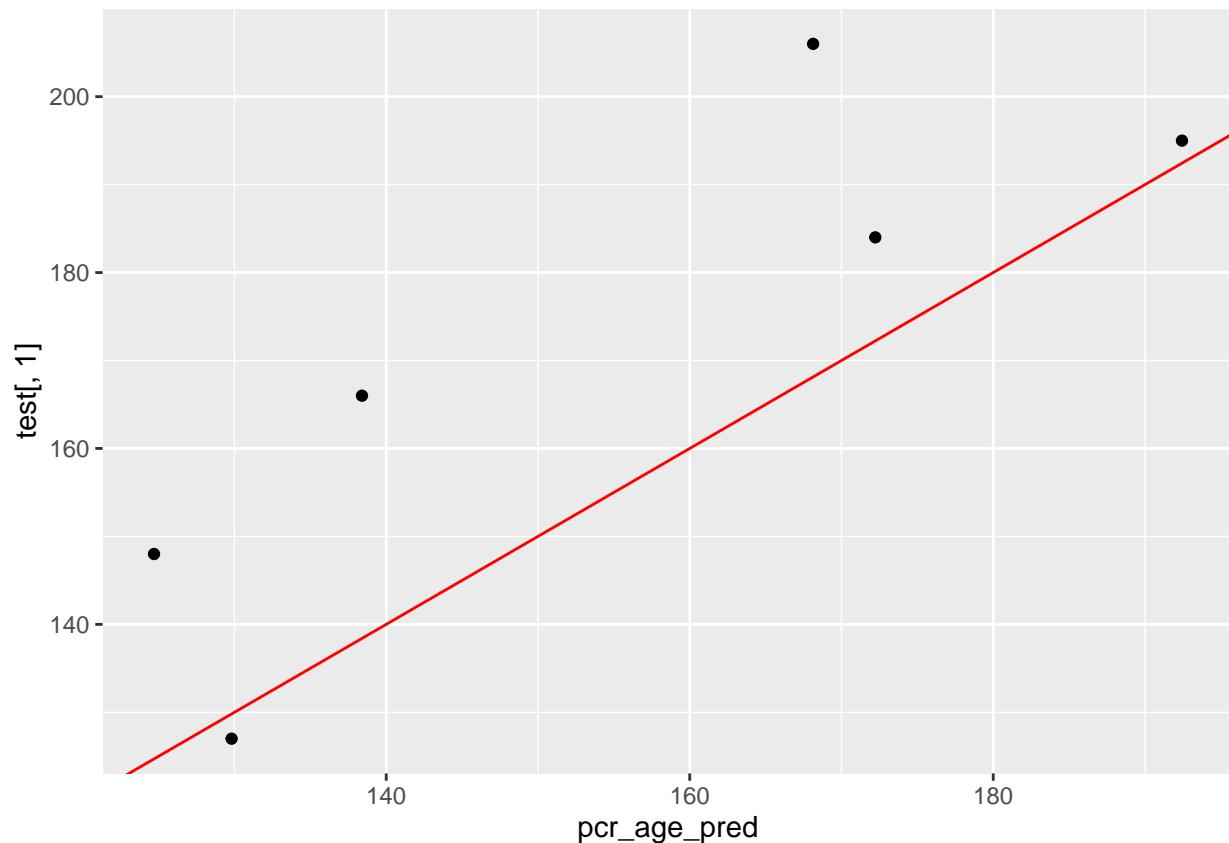
library(pls)
pcr_age_model <- pcr(read_age ~ .,
                       data = train,
                       scale = T,
                       validation = "CV",
                       ncomp = 10)

fit_age <- as.data.frame(pcr_age_model$fitted.values[, , 3])
fit_age <- fit_age %>%
  arrange(as.numeric(row.names(.)))
names(fit_age)[1] <- "fit"
ggplot() + geom_point(aes(x = fit_age$fit, y = train[, 1])) +
  geom_abline(slope = 1, col = "red")

```



```
pcr_age_pred <- predict(pcr_age_model, newdata=test, ncomp = 5)
ggplot() + geom_point(aes(x = pcr_age_pred, y = test[,1])) +
  geom_abline(slope = 1, col = "red")
```



```
rmsep_age_pcr <- as.data.frame(RMSEP(pcr_age_model))
rmsep_age_pcr[7:10,7] # RMSEP for 3 & 4 comps
```

```
## [1] 24.03883 23.98947 23.64716 23.47124
```