EDA_Analysis

George Acostalemus, Chunlin Liu, Sile Wang

2025-10-31

Load In Packages

```r
library(readxl)
library(dplyr)
```

```
##
## 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
```

```r
library(ggplot2)
library(boot)
library(patchwork)
library(lme4)
```

```
## Loading required package: Matrix
```

```r
library(lmtest)
```

```
## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
library(broom.mixed)
library(car)
```

```
## Loading required package: carData

##
## Attaching package: 'car'

## The following object is masked from 'package:boot':
##
##     logit

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

```r
library(MatchIt)
library(mgcv)
```

```
## Loading required package: nlme

##
## Attaching package: 'nlme'

## The following object is masked from 'package:lme4':
##
##     lmList

## The following object is masked from 'package:dplyr':
##
##     collapse

## This is mgcv 1.9-3. For overview type 'help("mgcv-package")'.
```

```r
library(gamm4)
```

```
## This is gamm4 0.2-7
```

```r
library(broom)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.4     v tibble    3.3.0
## v purrr     1.1.0     v tidyr     1.3.1
## v readr     2.1.5

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x nlme::collapse() masks dplyr::collapse()
## x tidyr::expand()  masks Matrix::expand()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x tidyr::pack()    masks Matrix::pack()
## x car::recode()    masks dplyr::recode()
## x purrr::some()    masks car::some()
## x tidyr::unpack()  masks Matrix::unpack()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(itsadug)
```

```
## Loading required package: plotfunctions
##
## Attaching package: 'plotfunctions'
##
## The following object is masked from 'package:ggplot2':
##
##     alpha
##
## Loaded package itsadug 2.4 (see 'help("itsadug")' ).
```

```r
library(purrr)
```

Read In The DataSet

You can also embed plots, for example:

```r
# Read in the dataset
scattering <- read_excel("radii.xlsx", sheet = "scattering")
retardance <- read_excel("radii.xlsx", sheet = "retardance")
orientation <- read_excel("radii.xlsx", sheet = "orientation")


scattering$property <- "scattering"
retardance$property <- "retardance"
orientation$property <- "orientation"

main_data <- bind_rows(scattering, retardance, orientation)

# Clean variable names for plotting
main_data$Groups <- factor(main_data$Groups, levels = c("control", "experimental"))
main_data$Region <- factor(main_data$Region, levels = c("front", "occipital"))
```

Counting

```r
# Counts per group and region
main_data %>%
  count(Groups, Region) %>%
  arrange(desc(n)) %>%
  arrange(desc(n))
```

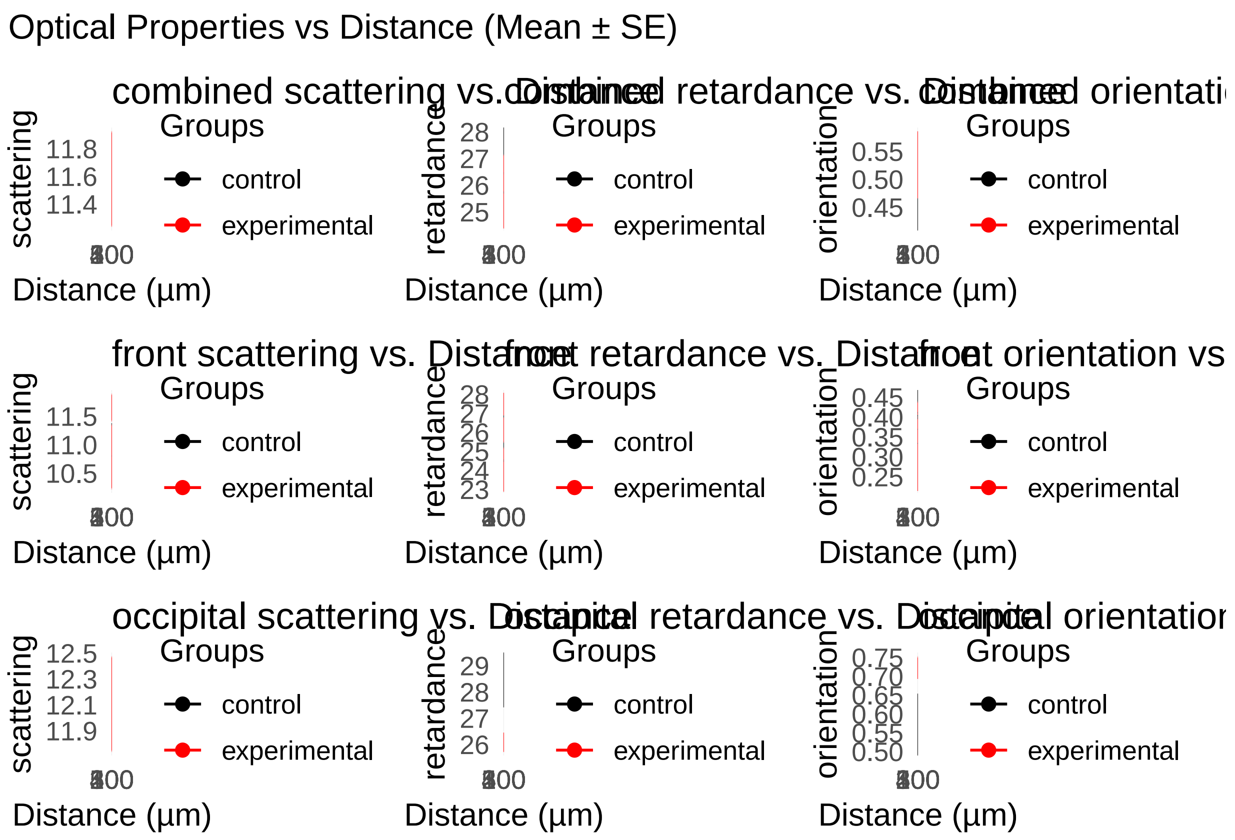
```
## # A tibble: 4 x 3
##   Groups       Region      n
##   <fct>        <fct>   <int>
## 1 control      <NA>   735359
## 2 control      front  225756
## 3 experimental <NA>    16038
## 4 experimental front   14190
```

```r
# Counts per property and group
main_data %>%
  count(property, Groups) %>%
  arrange(desc(n))
```

```
## # A tibble: 6 x 3
##   property    Groups            n
##   <chr>       <fct>         <int>
## 1 orientation control      320710
## 2 retardance  control      320407
## 3 scattering  control      319998
## 4 orientation experimental  10077
## 5 retardance  experimental  10077
## 6 scattering  experimental  10074
```

Recreate the graphs that was shown in the intake meeting

```r
# Create summmary (mean +- SE) by group, region, and distance
summary_df <- main_data %>%
  group_by(Groups, Region, property, distance) %>%
  summarise(
    mean_val = mean(OpticalProperty, na.rm = TRUE),
    sd_val = sd(OpticalProperty, na.rm = TRUE),
    n = n(),
    se = sd_val / sqrt(n),
    .groups = "drop"
  )
# Standarize Region Names
summary_df <- summary_df %>%
  mutate(
    Region = case_when(
      is.na(Region) ~ "occipital",
      TRUE ~ as.character(Region)
    ),
    Region = factor(Region, levels = c("front", "occipital"))
  )


# Define plot functions
plot_property <- function(df, region_filter, property_label) {
  df %>%
    filter(property == property_label, Region %in% region_filter) %>%
    ggplot(aes(x = distance, y = mean_val, color = Groups)) +
    geom_point(size = 2) +  # keep dots
    geom_errorbar(aes(ymin = mean_val - se, ymax = mean_val + se), width = 10) +
    # optional smooth trend line (remove if you want only dots + bars)
    # geom_smooth(method = "loess", se = FALSE, linetype = "dashed", alpha = 0.4) +
    scale_color_manual(values = c("black", "red")) +
    theme_minimal(base_size = 12) +
    labs(
      title = paste(paste(region_filter, collapse = " & "), property_label, "vs. Distance"),
      x = "Distance (µm)",
      y = property_label
    )
}



# Combined (all regions)
combined_summary <- summary_df %>%
  group_by(Groups, distance, property) %>%
  summarise(
    mean_val = mean(mean_val, na.rm = TRUE),
    se = sqrt(sum(se^2, na.rm = TRUE)) / 2,  # conservative SE estimate
    .groups = "drop"
  ) %>%
  mutate(Region = "combined")


summary_df_balanced <- bind_rows(summary_df, combined_summary)


p_combined_scatt <- plot_property(summary_df_balanced, "combined", "scattering")
p_combined_ret   <- plot_property(summary_df_balanced, "combined", "retardance")
p_combined_ori   <- plot_property(summary_df_balanced, "combined", "orientation")

p_front_scatt <- plot_property(summary_df_balanced, "front", "scattering")
p_front_ret   <- plot_property(summary_df_balanced, "front", "retardance")
p_front_ori   <- plot_property(summary_df_balanced, "front", "orientation")

p_occ_scatt <- plot_property(summary_df_balanced, "occipital", "scattering")
p_occ_ret   <- plot_property(summary_df_balanced, "occipital", "retardance")
p_occ_ori   <- plot_property(summary_df_balanced, "occipital", "orientation")

library(patchwork)
top_row <- p_combined_scatt + p_combined_ret + p_combined_ori
mid_row <- p_front_scatt + p_front_ret + p_front_ori
bottom_row <- p_occ_scatt + p_occ_ret + p_occ_ori

final_plot <- top_row / mid_row / bottom_row +
  plot_annotation(title = "Optical Properties vs Distance (Mean ± SE)")

final_plot
```



```
set.seed(1234)
main_data <- main_data %>%
  mutate(
    Groups = factor(Groups),              # control / experimental
    subID = factor(subID),
    distance = as.numeric(distance),
    OpticalProperty = as.numeric(OpticalProperty),
    property = factor(property, levels = c("scattering", "retardance", "orientation")),
    Region = tolower(as.character(Region))
  )

# Replace NA or blank with occipital, and recode synonyms properly
main_data <- main_data %>%
  mutate(
    Region = case_when(
      is.na(Region) ~ "occipital",
      Region %in% c("", "na") ~ "occipital",
      Region %in% c("front", "frontal") ~ "front",
      Region %in% c("occ", "occipital") ~ "occipital",
      TRUE ~ Region
    ),
    Region = factor(Region, levels = c("front", "occipital"))
  )

# Normalize Region factor to two levels: front and occipital
main_data <- main_data %>% mutate(Region = factor(ifelse(grepl("front", tolower(Region)), "front", "occ
                                        levels = c("front","occipital")))
```

```r
block_df <- main_data %>%
  group_by(Groups, subID, distance, property) %>%
  mutate(weight = n()) %>%               # weight = number of samples in that group-distance combo
  ungroup()



# Export basic counts for EDA
counts_table <- main_data %>%
  group_by(property, Region, Groups) %>%
  summarise(n_total = n(), n_blocks = n_distinct(subID), .groups = "drop")
```

dd

```r
# 4. Global LMM (distance continuous) per property
library(dplyr)
library(lme4)
library(tibble)

props <- unique(block_df$property)
global_results <- list()

for (prop in props) {
  datp <- block_df %>% filter(property == prop)
  datp <- datp %>% mutate(dist_c = distance - mean(distance, na.rm = TRUE))

  # Need enough subjects to estimate random effect
  if (n_distinct(datp$subID) < 4) {
    global_results[[as.character(prop)]] <- tibble(property = prop, model = NA, note = "too few subjects
    next
  }
  # Inlcude weights
  modg <- tryCatch(
    lmer(mean_val ~ Groups * dist_c + (1 | subID), data = datp, REML = TRUE),
    error = function(e) NULL
  )

  if (is.null(modg)) {
    global_results[[as.character(prop)]] <- tibble(property = prop, model = "lmer_failed", note = NA)
  } else {
    sm <- summary(modg)
    an <- anova(modg)

    # Display summary & ANOVA in console
    cat("\n========================================\n")
    cat("Model Summary for:", prop, "\n")
    cat("========================================\n")
    print(sm)
    cat("\n--- ANOVA Table ---\n")
    print(an)
    cat("\n\n")

    # Save summary & ANOVA to text file
    sink(file.path("~/Documents/MA 675/Consulting-Project-EPVs", paste0("global_lmm_summary_", prop, ".
    print(sm)
    print(an)
    sink()

    # Extract p-values from ANOVA
    p_group <- if ("Groups" %in% rownames(an)) an["Groups", "Pr(>F)"] else NA
    p_inter <- if ("Groups:dist_c" %in% rownames(an)) an["Groups:dist_c", "Pr(>F)"] else NA

    # Save results to the list
    global_results[[as.character(prop)]] <- tibble(
      property = prop,
      model = "lmer",
      p_group = p_group,
      p_interaction = p_inter,
      note = NA
    )
  }
}

# Combine results into a single dataframe
global_results_df <- bind_rows(global_results)

summary(modg)
```

```
## Length  Class   Mode
##      0   NULL   NULL
```

```
# Diagnostics for global LMM (example: for scattering)

diagnostic_plot_fn <- function(mod, label) {
  # Residuals vs Fitted plot
  dfres <- data.frame(fitted = fitted(mod), resid = resid(mod))

  p1 <- ggplot(dfres, aes(x = fitted, y = resid)) +
    geom_point(alpha = 0.6) +
    geom_smooth(se = FALSE, method = "loess", color = "blue") +
    geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
    labs(
      title = paste0("Residuals vs Fitted: ", label),
      x = "Fitted values",
      y = "Residuals"
    ) +
    theme_minimal(base_size = 13)

  print(p1)  # Display plot interactively

  # QQ plot (normality of residuals)
  qq_df <- data.frame(sample = resid(mod))
  p2 <- ggplot(qq_df, aes(sample = sample)) +
    stat_qq(alpha = 0.6) +
    stat_qq_line(color = "red") +
    labs(
```

```r
      title = paste0("QQ Plot: ", label),
      x = "Theoretical Quantiles",
      y = "Sample Quantiles"
    ) +
    theme_minimal(base_size = 13)

  print(p2)  # Display QQ plot interactively
}

# Run diagnostics for each property model if available
for (prop in props) {
  message("Showing diagnostics for: ", prop)

  datp <- block_df %>%
    filter(property == prop) %>%
    mutate(dist_c = distance - mean(distance, na.rm = TRUE))

  modg <- tryCatch(
    lmer(mean_val ~ Groups * dist_c + (1 | subID), data = datp, REML = TRUE),
    error = function(e) NULL
  )

  if (!is.null(modg)) {
    diagnostic_plot_fn(modg, paste0("global_", prop))
  } else {
    message("Model failed for: ", prop)
  }
}
```

```
## Showing diagnostics for: scattering

## Model failed for: scattering

## Showing diagnostics for: retardance

## Model failed for: retardance

## Showing diagnostics for: orientation

## Model failed for: orientation
```

REFIT MODEL WITH FACTORING

```r
# Helper to add subject-region block ID
add_block_id <- function(df) {
  df %>%
    mutate(
      Groups   = factor(Groups),
      Region   = factor(Region),
      distance = factor(distance, levels = sort(unique(distance))),
      block_id = interaction(subID, Region, drop = TRUE)
    )
}

# Diagnostics (same style as before)
plot_diagnostics <- function(mod, label){
  res <- residuals(mod, type = "pearson")
  fit <- fitted(mod)
  dfres <- data.frame(fitted = fit, resid = res)

  p1 <- ggplot(dfres, aes(fitted, resid)) +
    geom_point(alpha = 0.6) +
    geom_smooth(method = "loess", se = FALSE) +
    geom_hline(yintercept = 0, linetype = "dashed") +
    labs(title = paste("Residuals vs Fitted:", label),
         x = "Fitted values", y = "Pearson residuals") +
    theme_minimal(base_size = 12)

  p2 <- ggplot(dfres, aes(sample = resid)) +
    stat_qq(alpha = 0.6) + stat_qq_line() +
    labs(title = paste("QQ Plot:", label),
         x = "Theoretical Quantiles", y = "Sample Quantiles") +
    theme_minimal(base_size = 12)

  # Explicitly print so they render
  print(p1)
  print(p2)
}

# Main function: fit LME per property (distance as factor)
fit_factored_lme <- function(block_df,
                             outdir = "~/Documents/Consulting-Project-EPVS") {
  dir.create(outdir, showWarnings = FALSE, recursive = TRUE)

  df0 <- add_block_id(block_df)
  props <- unique(df0$property)

  results <- map(props, function(prop_i) {
    datp <- df0 %>% filter(property == prop_i) %>% droplevels()

    mod <- tryCatch(
      lme(
        fixed   = mean_val ~ Groups * distance,
        random  = ~1 | subID,                      # subject-level random effect
        weights = varIdent(form = ~1 | Groups),    # heteroskedasticity by group
        data    = datp,
        method  = "REML",
        na.action = na.omit
      ),
      error = function(e) NULL
    )

    if (is.null(mod)) {
      return(tibble(property = prop_i, model_status = "failed"))
```

```r
    }

    # Save text output
    sink(file.path(outdir, paste0("lme_factordist_summary_", prop_i, ".txt")))
    print(summary(mod))
    cat("\n--- ANOVA ---\n")
    print(anova(mod))
    sink()

    # Diagnostics
    plot_diagnostics(mod, paste("Factored model:", prop_i))

    # Compact p-value extraction
    an <- anova(mod)
    p_group <- if ("Groups" %in% rownames(an)) an["Groups", "p-value"] else NA
    p_inter <- {
      rn <- rownames(an)
      idx <- grepl("^Groups:distance", rn)
      if (any(idx)) an[which(idx)[1], "p-value"] else NA
    }

    tibble(property = prop_i,
           model_status = "ok",
           p_groups = p_group,
           p_interaction = p_inter)
  })

  bind_rows(results)
}
res_factored <- fit_factored_lme(block_df)
res_factored
```

```
## # A tibble: 3 x 2
##   property    model_status
##   <fct>       <chr>
## 1 scattering  failed
## 2 retardance  failed
## 3 orientation failed
```

SWP Overview and Goal

The Size-Weighted Proximity (SWP) metric is intended to quantify how strongly each voxel in the parenchyma is influenced by nearby EPVS, accounting for both distance and EPVS size. The question we are looking at is:
How do optical properties change as a function of log(SWP)?

Our goal within this analysis is not only to visualize the relationship, but also to quantify its shape, assess it strength, and determine whether the relationship appear linear or nonlinear. This helps evaluate whether changes in local tissue microstructure (reflected in optical properties) can be predicted from EPVS proximity and size

Begin SWP Relationship Analysis

Here are two figures based on what was presented in the intake meeting. It's clear that there is a non-linear patterns at higher log(SWP) values. Differences between frontal and occipital regions, suggesting a regional effect. In order look further into it we would need to fit a non-linear model. This because using any linear model would underestimate the relationship where it bends upward or downward and provides poor fit. Thus we needed a method that does *not* assume linearity, can adapt to the shape of the data, and still yields interpretable effects and significance tests.

```r
file <- "op_vs_swp_16-Oct-2025.xlsx"

sheets <- excel_sheets(file)

swp <- map_dfr(sheets, function(s) {
  read_excel(file, sheet = s) %>%
    rename(log_swp = 1, optical_value = 2) %>%  # rename first two columns consistently
    mutate(sheet_name = s)
})

# Label property + region
swp <- swp %>%
  mutate(
    property = case_when(
      str_detect(sheet_name, "mus") ~ "scattering",
      str_detect(sheet_name, "ret") ~ "retardance"
    ),
    region = case_when(
      str_detect(sheet_name, "front") ~ "frontal",
      str_detect(sheet_name, "occip") ~ "occipital",
      TRUE ~ "combined"
    )
  )

glimpse(swp)
```

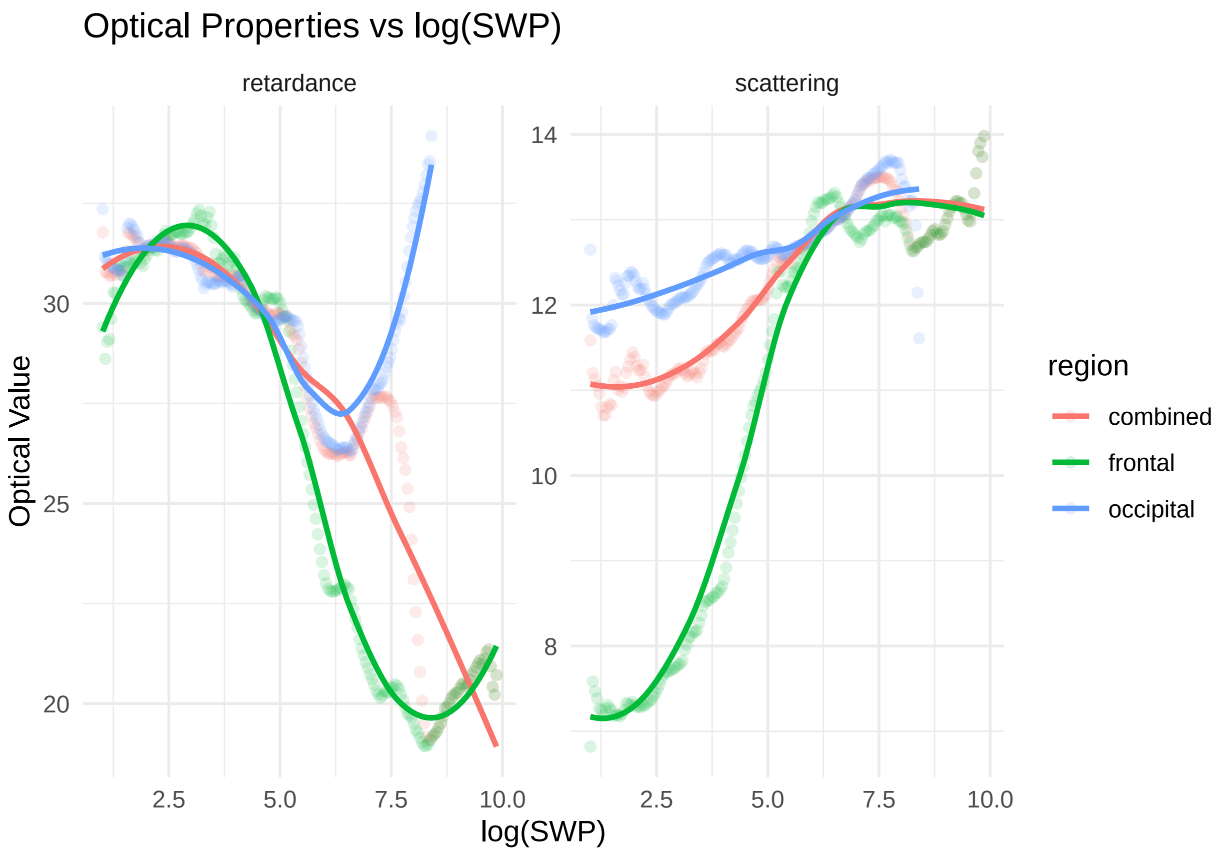
```
## Rows: 1,140
## Columns: 5
## $ log_swp       <dbl> 1.013489, 1.069199, 1.116735, 1.163813, 1.210787, 1.2577~
## $ optical_value <dbl> 11.58480, 11.20085, 11.12777, 11.04941, 10.96003, 10.805~
## $ sheet_name    <chr> "comb_mus", "comb_mus", "comb_mus", "comb_mus", "comb_mu~
## $ property      <chr> "scattering", "scattering", "scattering", "scattering", ~
## $ region        <chr> "combined", "combined", "combined", "combined", "combine~
```

```r
table(swp$property, swp$region)
```

```
##
##                combined frontal occipital
##   retardance        190     190       190
##   scattering        190     190       190
```

```r
ggplot(swp, aes(x = log_swp, y = optical_value, color = region)) +
  geom_point(alpha = 0.15) +
  geom_smooth(method = "loess", se = FALSE) +
  facet_wrap(~ property, scales = "free_y") +
  theme_minimal() +
  labs(title = "Optical Properties vs log(SWP)", y = "Optical Value", x = "log(SWP)")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



A Generalized Additive Model (GAM) is a type of statistical model that allows the relationship between a predictor and an outcome to be non-linear, instead of forcing it to be a straight line.

In a normal linear regression, we assume:

- Every unit increase in X changes the outcome by the same amount.
- The relationship is assumed to be a straight line.

But biological processes often aren't linear — especially tissue changes in disease.

A GAM replaces the straight line with a smooth curve:

We used a Generalized Additive Model because we expected the relationship between size-weighted proximity (SWP) and the optical properties to be non-linear.

A GAM allows us to model this relationship flexibly, without forcing it into a straight line or predefined curve.
This lets the data determine the true shape of how tissue optical properties change as proximity to EPVS increases, while still adjusting for differences between brain regions.

```
gam_scatter <- gam(optical_value ~ s(log_swp, k = 6) + region,
                   data = filter(swp, property == "scattering"))

gam_retard <- gam(optical_value ~ s(log_swp, k = 6) + region,
                  data = filter(swp, property == "retardance"))

summary(gam_scatter)
```

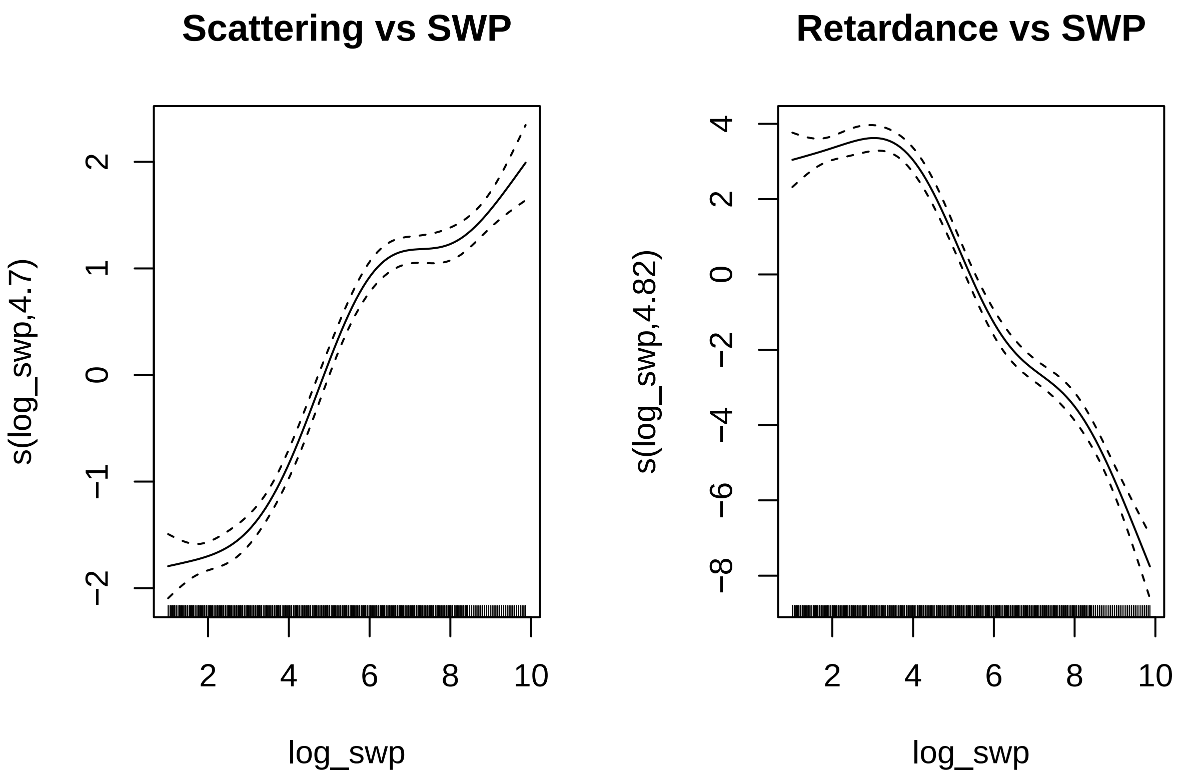
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## optical_value ~ s(log_swp, k = 6) + region
##
## Parametric coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       12.16170    0.06327 192.207  < 2e-16 ***
## regionfrontal     -1.43294    0.08908 -16.085  < 2e-16 ***
## regionoccipital    0.66085    0.09086   7.273 1.19e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##               edf Ref.df     F p-value
## s(log_swp) 4.699  4.955 243.9  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.744   Deviance explained = 74.7%
## GCV = 0.76424  Scale est. = 0.75392   n = 570
```

```
summary(gam_retard)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## optical_value ~ s(log_swp, k = 6) + region
##
## Parametric coefficients:
```

```
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       27.7548     0.1498 185.310  < 2e-16 ***
## regionfrontal     -1.3834     0.2109  -6.560 1.22e-10 ***
## regionoccipital    1.3557     0.2151   6.303 5.92e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##               edf Ref.df   F p-value
## s(log_swp) 4.816  4.983 294  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =   0.76   Deviance explained = 76.3%
## GCV = 4.2828  Scale est. = 4.2241    n = 570
```

```r
par(mfrow=c(1,2))
plot(gam_scatter, shade = TRUE, main = "Scattering vs SWP")
plot(gam_retard, shade = TRUE, main = "Retardance vs SWP")
```



We modeled scattering and retardance as smooth functions of SWP and included region as a fixed effect. In both cases, the smooth term was highly significant, confirming a non-linear relationship between SWP and tissue optical properties. Scattering increases and then decreases as SWP increases, consistent with an early increase in extra cellular tissue density followed by late-stage tissue degradation. Retardance decreases and then increases, which suggests initial myelin breakdown, followed by fiber compression or re-packing in more advanced disease. Additionally, both scattering and retardance are higher in the occipital region than in the frontal region, consistent with the known pattern of CAA-related white matter vulnerability. The models is a good fit with the adjusted r-squred value (0.74 and 0.76)

I visualized the relationship between EPVS Size-Weighted Proximity (SWP) and optical properties.

Both scattering and retardance show non-linear dependence on SWP, so linear models are not appropriate.

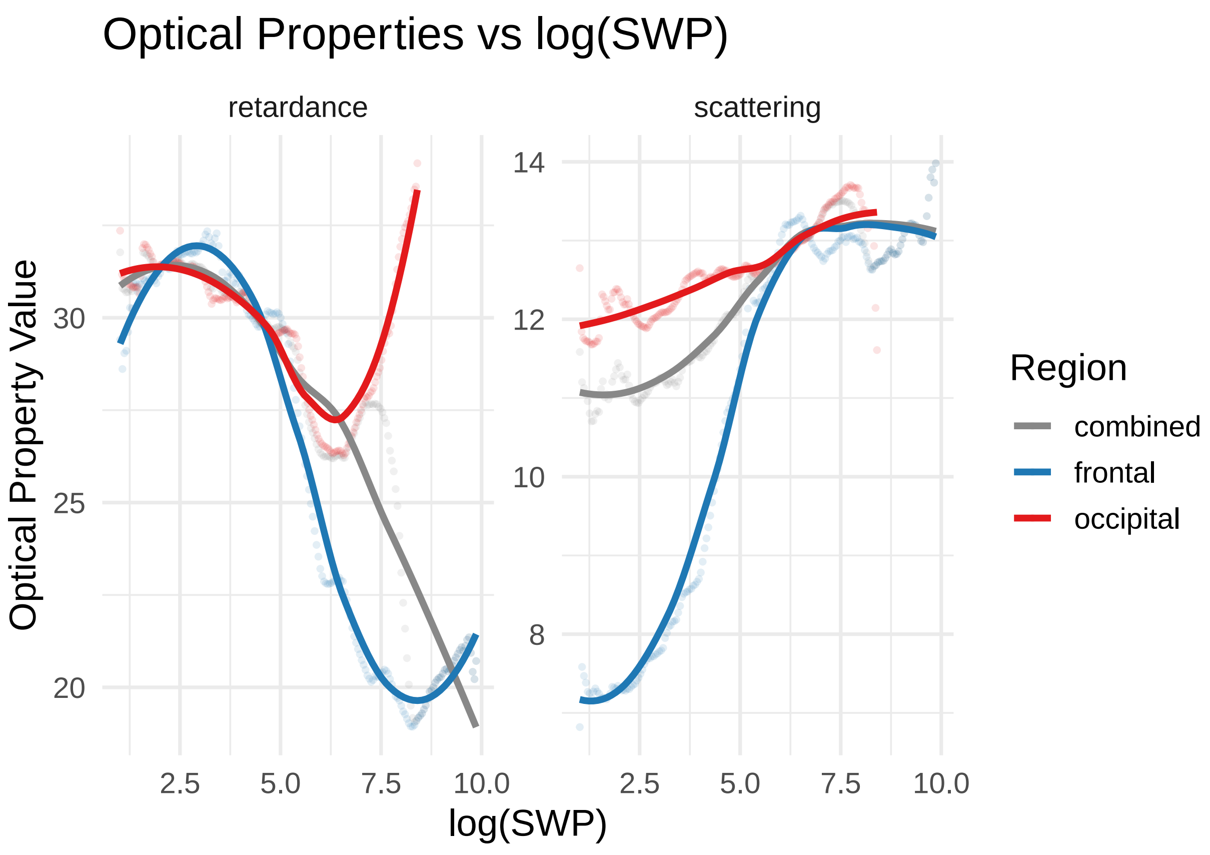
A GAM model captures the shape of the curve, which aligns with the biological interpretation:

- Scattering increases at moderate SWP, likely due to increased cellular density or ECM changes — then drops at high SWP, consistent with late-stage tissue breakdown.
- Retardance decreases initially, due to myelin degeneration — then rises again, suggesting myelin fiber compression in late disease progression.
- Regional effects (frontal vs occipital) will be tested next, but occipital is expected to show stronger pathology based on previous findings.

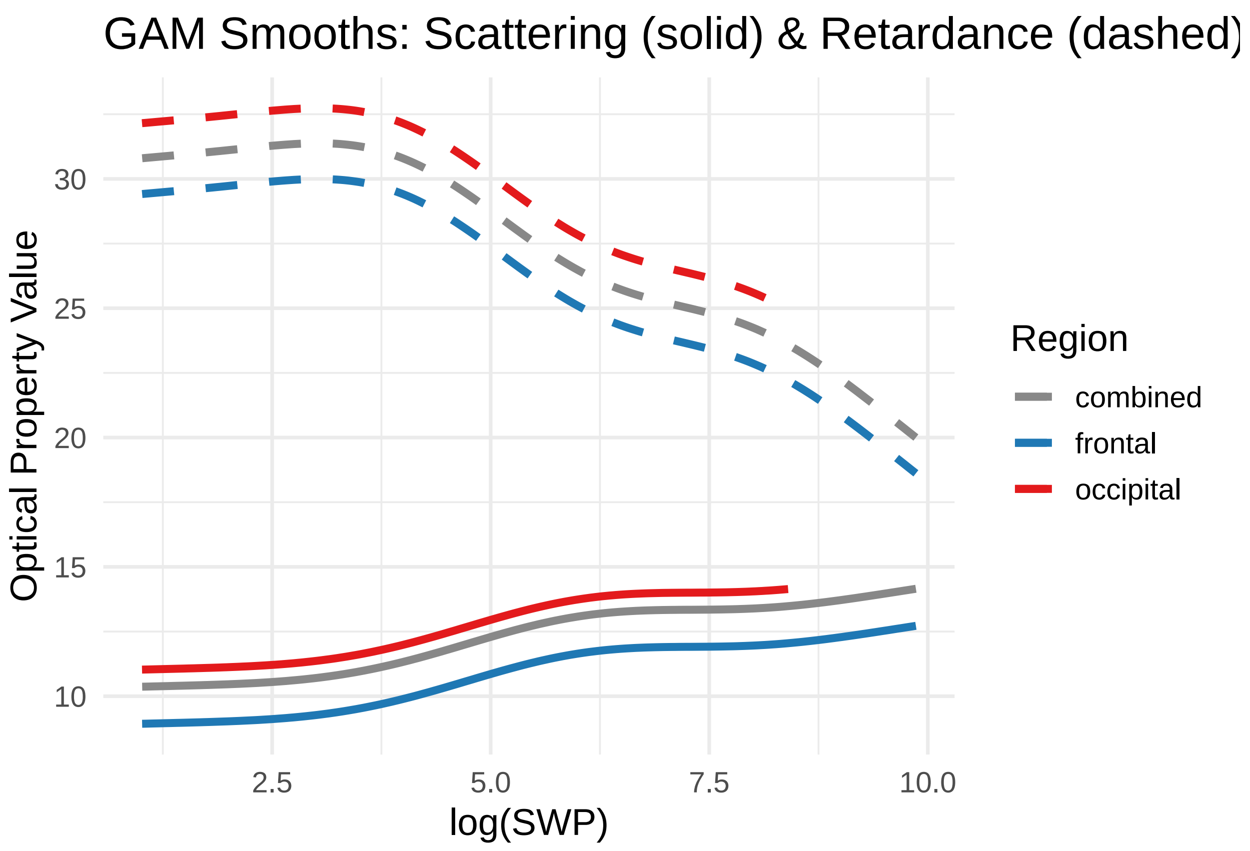
Below is a more advanced visualization

```r
p1 <- ggplot(swp, aes(x = log_swp, y = optical_value, color = region)) +
  geom_point(alpha = 0.12, size = 0.7) +
  geom_smooth(method = "loess", se = FALSE, linewidth = 1.2) +
  facet_wrap(~ property, scales = "free_y") +
  scale_color_manual(values = c("combined"="#888888", "frontal"="#1f78b4", "occipital"="#e31a1c")) +
  theme_minimal(base_size = 14) +
  labs(title = "Optical Properties vs log(SWP)",
       x = "log(SWP)",
       y = "Optical Property Value",
       color = "Region")
p1
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



```r
# Predict for smooth plotting
newdata <- swp %>%
  group_by(region) %>%
  reframe(log_swp = seq(min(log_swp), max(log_swp), length.out = 200), .groups="drop")

newdata$scatter_pred <- predict(gam_scatter, newdata)
newdata$ret_pred <- predict(gam_retard, newdata)
p2 <- ggplot(newdata, aes(x = log_swp, color = region)) +
  geom_line(aes(y = scatter_pred), linewidth = 1.4) +
  geom_line(aes(y = ret_pred), linetype="dashed", linewidth=1.4) +
  scale_color_manual(values = c("combined"="#888888", "frontal"="#1f78b4", "occipital"="#e31a1c")) +
  theme_minimal(base_size = 14) +
  labs(title = "GAM Smooths: Scattering (solid) & Retardance (dashed)",
       x = "log(SWP)",
       y = "Optical Property Value",
       color = "Region")

p2
```


GAM Smooths: Scattering (solid) & Retardance (dashed)
Optical Property Value
log(SWP)
Region
combined
frontal
occipital