Supplementary Material V: R Code

This document contains complete code for reproduction of simulation experiments reported in Sections 4.1 and 4.2 of the article “Meta-Analysis of Generalized Additive Models in Neuroimaging Studies” by Sørensen et al.

Simulation Study in Section 4.1: Function Estimation

Below is the complete R code necessary to reproduce the simulation experiment in Section 4.1. This script is self contained.

```
library(tidyverse); library(furrr); library(metagam); library(mgcv)

# Functional forms used:
dx <- seq(from = 0, to = 1, by = .001)
f0 <- function(x) 2 * sin(pi * x)/1.271967 - mean(2 * sin(pi * dx)/1.271967)
f1 <- function(x) exp(2 * x)/4.193528 - mean(exp(2 * dx)/4.193528)
f2 <- function(x) (0.2 * x^11 * (10 * (1 - x))^6 + 10 *
  (10 * x)^3 * (1 - x)^10)/5.526776 - mean( (0.2 * dx^11 * (10 * (1 - dx))^6 + 10 *
                         (10 * dx)^3 * (1 - dx)^10)/5.526776)
f3 <- function(x) 0 * x

# Simulation with univariate smooths
iteration <- 1:1000
n_tot <- 4000

# Grid for use in predictions
grid <- list(
  "s(x0)" = tibble(x0 = seq(from = 0, to = 1, by = .01), x1 = 0, x2 = 0, x3 = 0),
  "s(x1)" = tibble(x0 = 0, x1 = seq(from = 0, to = 1, by = .01), x2 = 0, x3 = 0),
  "s(x2)" = tibble(x0 = 0, x1 = 0, x2 = seq(from = 0, to = 1, by = .01), x3 = 0),
  "s(x3)" = tibble(x0 = 0, x1 = 0, x2 = 0, x3 = seq(from = 0, to = 1, by = .01))
)
# Model formula used
form <- as.formula(y ~ s(x0, k = 20, bs = "cr") +
                   s(x1, k = 10, bs = "cr") +
                   s(x2, k = 30, bs = "cr") +
                   s(x3, k = 5, bs = "cr"))
# Terms to meta analyze
terms <- c("s(x0)", "s(x1)", "s(x2)", "s(x3)")
# Standard deviations
noise <- c(1, 1.6)
# Settings
sample_settings <- list(subset = c("full", "unequal", "unequal", "equal"),
                        xrange = c("full", "full", "sub", "full"))

plan(multisession)
simres <- future_pmap_dfr(crossing(iteration, noise), function(iteration, noise){
  # Create the full dataset
  dat <- tibble(
```

```
  x0 = runif(n_tot), x1 = runif(n_tot), x2 = runif(n_tot), x3 = runif(n_tot),
  y = f0(x0) + f1(x1) + f2(x2) + rnorm(n_tot, sd = noise)
)

# Split the dataset and fit models to each
pmap_dfr(sample_settings, function(subset, xrange){
  datasplit <- if(subset == "full"){
    dat %>%
      mutate(ID = 1) %>%
      group_by(ID) %>%
      nest()
  } else if(subset == "equal"){
    dat %>%
      mutate(ID = ceiling(row_number() / 800)) %>%
      group_by(ID) %>%
      nest()
  } else if(subset == "unequal" && xrange == "full"){
    dat %>%
      mutate(ID = case_when(
        row_number() <= 300 ~ 1,
        row_number() <= 800 ~ 2,
        row_number() <= 1600 ~ 3,
        row_number() <= 2600 ~ 4,
        row_number() <= 4000 ~ 5,
        TRUE ~ NA_real_)
      ) %>%
      group_by(ID) %>%
      nest()
  } else if(subset == "unequal" && xrange == "sub"){
    cutoff <- .5

    dat_cp <- dat %>%
      mutate(ID = row_number())
    # The first cohort contains subjects with x2 < .5
    dat1 <- dat_cp %>%
      filter(x2 < cutoff) %>%
      sample_n(300)

    # The second cohort contains subjects with x2 > .5
    dat2 <- dat_cp %>%
      anti_join(dat1, by = "ID") %>%
      filter(x2 > 1 - cutoff) %>%
      sample_n(500)

    # The third cohort contains subjects with x1 < .5
    dat3 <- dat_cp %>%
      anti_join(dat1, by = "ID") %>%
      anti_join(dat2, by = "ID") %>%
      filter(x1 < cutoff) %>%
      sample_n(800)

    dat4 <- dat_cp %>%
      anti_join(dat1, by = "ID") %>%
      anti_join(dat2, by = "ID") %>%
      anti_join(dat3, by = "ID") %>%
      filter(x1 > 1 - cutoff) %>%
      sample_n(1000)

    dat5 <- dat_cp %>%
```

```
    anti_join(dat1, by = "ID") %>%
    anti_join(dat2, by = "ID") %>%
    anti_join(dat3, by = "ID") %>%
    anti_join(dat4, by = "ID")

  tibble(
    ID = 1:5,
    data = list(dat1, dat2, dat3, dat4, dat5)
  )
 }

 # Compute the curves on the grid
 if(nrow(datasplit) > 1){
  fits <- pmap(datasplit, function(ID, data){
   fit <- gam(form, data = data)
   strip_rawdata(fit)
  })

  estimated_curves <- map_dfr(terms, function(term){
   metagam(fits, grid[[term]], type = "iterms", terms = term,
        method = "FE", intercept = TRUE)$meta_estimates %>%
    select(x0, x1, x2, x3, term, estimate, se)
  })

 } else {
  fit <- gam(form, data = datasplit$data[[1]])
  estimated_curves <- map_dfr(terms, function(term){
   preds <- predict(fit, newdata = grid[[term]], se.fit = TRUE,
                type = "iterms", terms = term)

   grid[[term]] %>%
    mutate(
     term = !!term,
     estimate = c(preds$fit),
     se = c(preds$se.fit)
    )
  })
 }
 estimated_curves %>%
  mutate(
   subset = subset,
   xrange = xrange
  )
}) %>%
 mutate(
  iteration = iteration,
  noise = noise,
  sample = if_else(subset == "unequal" & xrange == "sub", "unequal_range", subset)
 )
}, .progress = TRUE, .options = future_options(seed = 123L))
```

Simulation Study in Section 4.2: Hypthesis Testing and Power

The snippet below shows the R code used in the simulation study of Section 4.2. The file "power_sim_functions.RData" is available upon request from the corresponding author.

```
library(tidyverse); library(furrr); library(mgcv); library(metagam)
# This file power_sim_functions.RData is required to rerun the simulations.
load("power_sim_functions.RData")

n_splits <- 6
nmc <- 1000 # Monte Carlo samples at each parameter setting
form <- as.formula(Volume ~ s(Age, bs = "cr") + s(Age, by = Group, bs = "cr") + Group)

simulate_data <- function(n, sd_eps, mod, use_interaction = TRUE){
 tibble(
  Age = runif(n, min = 4, max = 93),
  Group = ordered(sample(0:1, size = n, replace = TRUE))
 ) %>%
  mutate(
   Volume = predict(mod, newdata = .) + use_interaction * interact(Group, Age) + rnorm(n, sd =
sd_eps)
  )
}

# Grid for meta-analysis
grid <- tibble(Age = seq(from = 4, to = 94, by = 1), Group = factor(1, levels = 0:1))

meta_analyze <- function(n, n_splits, form, simdat, alpha, grid){
 # Sample row partitioning
 row_inds <- sample(1:n_splits, size = n, replace = TRUE)
 # Fit separate models
 models <- map(1:n_splits, function(sep_no){
  sep_fit <- gam(form, data = filter(simdat, row_inds == sep_no))
  strip_rawdata(sep_fit)
 })
 # Join the fits
 metagam(models, grid = grid, type = "iterms", terms = "s(Age):Group1")

}

param <- bind_rows(
 sd_eps_var = crossing(
  sd_eps = seq(from = 1000, to = 15000, by = 1000),
  i = 1:nmc,
  n = 3000,
  use_interaction = c(FALSE, TRUE)
 ),
 n_var = crossing(
  sd_eps = 3500,
  i = 1:nmc,
  n = seq(from = 900, to = 3000, by = 300),
  use_interaction = c(TRUE, FALSE)
 ),
 .id = "simname"
)

plan(multisession)
power_dat <- future_pmap_dfr(param, function(simname, sd_eps, i, n, use_interaction){
```

```
 simdat <- simulate_data(n, sd_eps, mod, use_interaction)
 simmod_full <- gam(form, data = simdat) # Full data

 simmod_sep <- gam(form, data = slice(simdat, 1:floor(n / n_splits))) # One dataset
 meta_test <- meta_analyze(n, n_splits, form, simdat, alpha, grid) # Meta analysis

 imap_dfr(list(full_data = simmod_full, sep_data = simmod_sep, meta_data = meta_test), function(x, n){
  if(inherits(x, "gam")) {
   pval <- tibble(name = n, value = c(summary(x)$s.table["s(Age):Group1", "p-value"]))
  } else {
   pval <- select(x$meta_pvals, term, ends_with("pval"))
   pval <- pivot_longer(pval, cols = -term)
   pval <- dplyr::select_at(pval, dplyr::vars(-"term"))
  }
 }) %>%
  mutate(iteration = i, sd_eps = sd_eps, n = n, use_interaction = use_interaction, simname = simname)
}, .options = future_options(seed = 123L), .progress = TRUE)

power_dat <- power_dat %>% mutate(
 name = recode(name,
           full_data = "Mega-analysis",
           sep_data = "Single dataset",
           maximump_pval = "Meta-analysis; Wilkinson",
           minimump_pval = "Meta-analysis; Tippet",
           logitp_pval = "Meta-analysis; logit",
           sumlog_pval = "Meta-analysis; Fisher",
           sump_pval = "Meta-analysis; Edgington",
           sumz_pval = "Meta-analysis; Stouffer")
)
```