Generalised Additive Model

Table of contents

To illustrate generalised additive models (GAM), we fit a model with a fixed smooth function of time, and random (person-specific) smooth functions of time.

To reproduce the results, it is necessary to prepare the data set, plot base, and training and test data sets, as outlined in the "Data Preparation" section.

Preparation

Loading Required Packages and Data

Load the necessary packages, data sets, and other supporting files. Each element serves a specific purpose:

- `tidyverse`: For data manipulation and visualisation.
- `mgcv`: To fit the generalised additive model.

- `gratia`: For obtaining the estimated slopes across the time points.
- `tidygam`: To compute the confidence intervals of the predictions.
- `caret`: To compute model performance indices.
- `plot_base`: A pre-configured ggplot object for visualisation.
- **Training and Test Data sets**: Required for cross-validation.

```
# Load necessary packages
library(tidyverse)
library(mgcv)
library(gratia)
library(tidygam)
library(caret)

# Load the data set
load("data/wido.rdata")

# Load the pre-configured plot base
plot_base <- readRDS("objects/plot_base.rds")

# Load training and test datasets for cross-validation
training_datasets <- readRDS("objects/training_datasets.rds")
test_datasets <- readRDS("objects/test_datasets.rds")
```

Analysis

Fitting the Model

Determine k. This defines the maximum possible complexity, or "wiggliness," of the model. The exact choice of k is not critical; it should be large enough to represent the underlying pattern but small enough to maintain computational efficiency (Wood, 2017). We use a k of one fewer than the levels of the time variable, to allow for a wiggle with each new level (see also Doré & Bolger, 2018). Fit the GAM. We use a fixed smooth function of time, specifying k; and random (person-specific) smooth functions of time, using `bs = "fs"`. The `m = 1` argument sets a penalty for the smooth. Alternatively, to estimate only random intercepts specify `s(id, bs = "re")`. To estimate random slopes specify `s(mnths, id, bs = "re")`.

```r
# Determine number of unique values of time to define k
n_distinct(wido$mnths)
```

```
[1] 334
```

```r
# Fit the generalised additive model
gam <- gam(lifesatisfaction ~ s(mnths, k = 333) + s(mnths, id, bs = "fs", m =
 ↪  1), data = wido, method = "REML")

# Display the summary of the model
summary(gam)
```

```
Family: gaussian
Link function: identity

Formula:
lifesatisfaction ~ s(mnths, k = 333) + s(mnths, id, bs = "fs",
    m = 1)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.98120    0.05945   83.79   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
               edf  Ref.df     F p-value
s(mnths)     17.13   21.26 8.587  <2e-16 ***
s(mnths,id) 504.75 1711.00 3.164  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.717   Deviance explained = 78.1%
-REML = 2661.5  Scale est. = 0.3478    n = 2322
```

Inspect Slope Changes Over Time

To get an indication of when and how change in life satisfaction is taking place, examine how the slope (first derivative) changes across the range of time. Time periods where the

confidence interval of the slope straddles zero indicate periods of stability. Positive slopes indicate increases, while negative slopes indicate declines. Compute the slopes and confidence intervals across the the values of time. To inspect them, the `slopes`-object created below can be printed, but here we only print the time intervals in which there is an increasing and a decreasing trend.

```
# Compute the slopes (first derivatives) across the values of time
slopes <- derivatives(gam, n = 333, n_sim = 100, select = "s(mnths)")

# Create a variable indicating whether there is an increasing or a decreasing
 ↪  trend, or stability.
slopes <- slopes %>%
    mutate(
        trend = case_when(
          .lower_ci > 0 & .upper_ci > 0 ~ "increasing",
          .lower_ci < 0 & .upper_ci < 0  ~ "decreasing",
          TRUE ~ "stability"))

# Print range of time when there is a decreasing trend
range(slopes$mnths[slopes$trend == "decreasing"])
```

```
[1] -18.855422   2.650602
```

```
# Print range of time when there is an increasing trend
range(slopes$mnths[slopes$trend == "increasing"])
```

```
[1] 10.17771 24.15663
```

Visualisation

Predicting Average and Individual Trajectories

Predict both the population-level (fixed effects) and individual-level (random effects) trajectories of life satisfaction.

```r
# Predict population-level trajectories based on fixed effects
wido$lifesatisfaction_gam_f <- predict(gam, newdata = wido,
 ↪  exclude="s(mnths,id)")

# Predict individual-level trajectories based on random effects
wido$lifesatisfaction_gam_r <- predict(gam, newdata = wido)
```

Compute Confidence Intervals

Compute the confidence intervals for the predicted values of the model using the `predict_gam` function from the `tidygam` package. `ci_z = 1.96` estimates 95% confidence intervals.

```r
# Compute confidence intervals for 400 values of time (mnths)
ci <- predict_gam(gam, exclude_terms = "s(mnths,id)", length_out = 400, ci_z
 ↪  = 1.96)

# In the object with the confidence intervals, round down the values of time
 ↪  (mnths)
ci$mnths <- round(ci$mnths, 0)

# Select the relevant columns of the confidence intervals object
ci <- ci %>% select(mnths, lower_ci, upper_ci)

# Merge the confidence intervals with the wido data frame based on the mnths
 ↪  column
wido <- wido %>%
  left_join(ci, by = "mnths")
```

Selecting a Random Sample for Plotting

For better visualisation, select a random sample of individuals to display their individual trajectories.

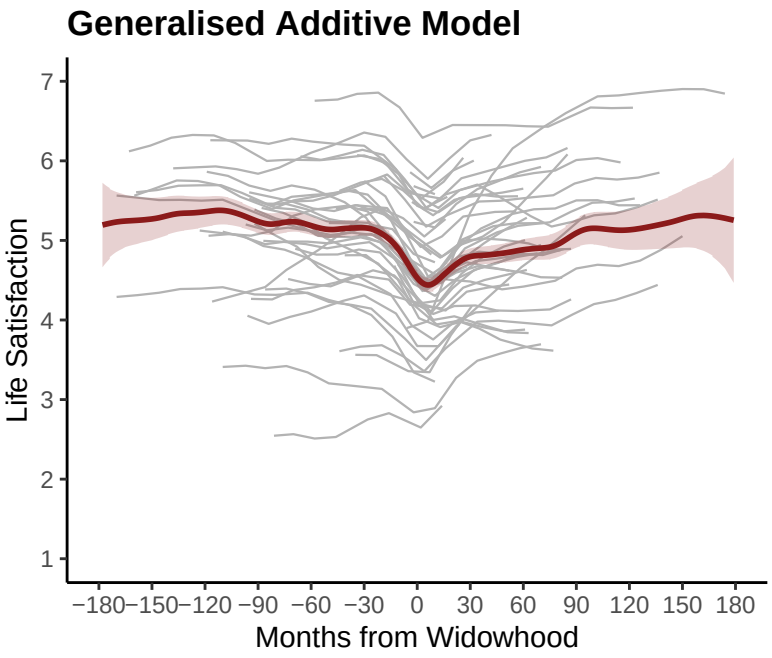
```r
# For reproducibility
set.seed(123)

# Randomly sample 50 participants
rsample_ids <- sample(unique(wido$id), 50)
```

```r
# Filter the data to include only the randomly selected participants
wido_rsample <- wido %>%
  filter(id %in% rsample_ids)
```

Creating the Plot

Combine all elements to create the plot, which includes individual trajectories, the population trajectory, and the confidence interval of the population trajectory.

```r
# Create the plot using the pre-configured plot base
plot_base +
  geom_line(
    data = wido_rsample,
    aes(mnths, lifesatisfaction_gam_r, group = id),
    color = "grey70",
    linewidth = 0.4
  ) +
  geom_line(
    data = wido,
    aes(mnths, lifesatisfaction_gam_f),
    color = "firebrick4",
    linewidth = 1
  ) +
  geom_ribbon(
    data = wido,
    aes(
      ymin = lower_ci,
      ymax = upper_ci,
      x = mnths
    ),
    alpha = 0.2,
    fill = "firebrick4"
  ) +
  ggtitle("Generalised Additive Model") +
  theme(plot.title = element_text(size = 13, face = "bold"))
```



Model Performance

Evaluating the Model

Assess the model's performance using the Bayesian Information Criterion (BIC), R-squared (R^2), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE).

```
# Compute BIC for the fitted model
round(BIC(gam), 2)
```

```
[1] 7626.69
```

```
# Calculate R², MAE, and RMSE for the fixed effects predictions
data.frame(
  R2_FE = round(R2(wido$lifesatisfaction_gam_f, wido$m_lifesat_per_mnth), 2),
  MAE_FE = round(MAE(wido$lifesatisfaction_gam_f, wido$m_lifesat_per_mnth),
  ↪  2),
  RMSE_FE = round(RMSE(wido$lifesatisfaction_gam_f, wido$m_lifesat_per_mnth),
  ↪  2)
)
```

```
  R2_FE MAE_FE RMSE_FE
1  0.33   0.28    0.38
```

```
# Calculate R², MAE, and RMSE for the random effects predictions
data.frame(
  R2_RE = round(R2(wido$lifesatisfaction_gam_r, wido$lifesatisfaction), 2),
  MAE_RE = round(MAE(wido$lifesatisfaction_gam_r, wido$lifesatisfaction), 2),
  RSME_RE = round(RMSE(wido$lifesatisfaction_gam_r, wido$lifesatisfaction),
   ↪  2)
)
```

```
  R2_RE MAE_RE RSME_RE
1  0.79   0.39    0.52
```

Cross-Validation

To assess the replicability of the model, perform cross-validation using the training and test datasets. For each training dataset, fit the model and compute performance metrics for the associated test dataset R^2, MAE, and RMSE.

```
# Determine the lowest number of unique values of time in the training
 ↪  datasets to define k
unique_timevalues_training_data <- numeric(length(training_datasets))

for (i in 1:length(training_datasets)) {
  training_data <- training_datasets[[i]]
  unique_timevalues_training_data[i] <- n_distinct(training_data$mnths)
}

min(unique_timevalues_training_data)
```

```
[1] 318
```

```
# Initialise vectors to store performance metrics
R2_values <- c()
MAE_values <- c()
RMSE_values <- c()

# Perform cross-validation
```

```r
for (i in 1:length(training_datasets)) {
  # Get the current training and test dataset
  training_data <- training_datasets[[i]]
  test_data <- test_datasets[[i]]

  # Fit the model
  gam <- gam(lifesatisfaction ~ s(mnths, k = 317) + s(mnths, id, bs = "fs", m
 ↪  = 1), data = training_data, method = "REML")

  # Predict fixed effects
  predictions <- predict(gam, newdata = test_data, exclude = "s(mnths,id)")

  # Compute average test trajectory
  test_data <- test_data %>%
    group_by(mnths) %>%
    mutate(mean_ls = mean(lifesatisfaction, na.rm = TRUE))

  # Compute performance metrics
  R2_value <- R2(predictions, test_data$mean_ls)
  RMSE_value <- RMSE(predictions, test_data$mean_ls)
  MAE_value <- MAE(predictions, test_data$mean_ls)

  # Store the metrics
  R2_values <- c(R2_values, R2_value)
  RMSE_values <- c(RMSE_values, RMSE_value)
  MAE_values <- c(MAE_values, MAE_value)
}

# Compute average performance metrics (mean)
  average_R2 <- mean(R2_values)
  average_MAE <- mean(MAE_values)
  average_RMSE <- mean(RMSE_values)

# Compute average performance metrics (SD)
  sd_R2 <- sd(R2_values)
  sd_MAE <- sd(MAE_values)
  sd_RMSE <- sd(RMSE_values)

# Combine the mean and standard deviation into one data.frame
combined_metrics <- data.frame(
  Metric = c("R²", "MAE", "RMSE"),
  Mean = round(c(average_R2, average_MAE, average_RMSE), 2),
```

```
  SD = round(c(sd_R2, sd_MAE, sd_RMSE), 2)
)

# Print the combined metrics
print(combined_metrics)
```

```
  Metric Mean   SD
1     R² 0.12 0.05
2    MAE 0.58 0.08
3   RMSE 0.76 0.12
```