2024 LT Electricity Consumption Functional Data Project

2025-05-25

```
set.seed(6558765)

consumption_data <- read_parquet("data/fda_duomenys.parquet")
consumption_data$Suvartojimo_laikotarpis <- as.Date(consumption_data$Suvartojimo_laikotarpis)

consumption_2024 <- consumption_data %>% filter(year(Suvartojimo_laikotarpis) == 2024)

population <- read_excel("data/savivaldybes_populiacija.xlsx")

consumption_by_population_2024 <- consumption_2024 %>%
  left_join(population, by = "Savivaldybe")

consumption_by_population_2024 <- consumption_by_population_2024 %>%
  mutate(kWh_per_person = Bendras_suvartojimas_mWh * 1000/ populiacija)
```

Loading data

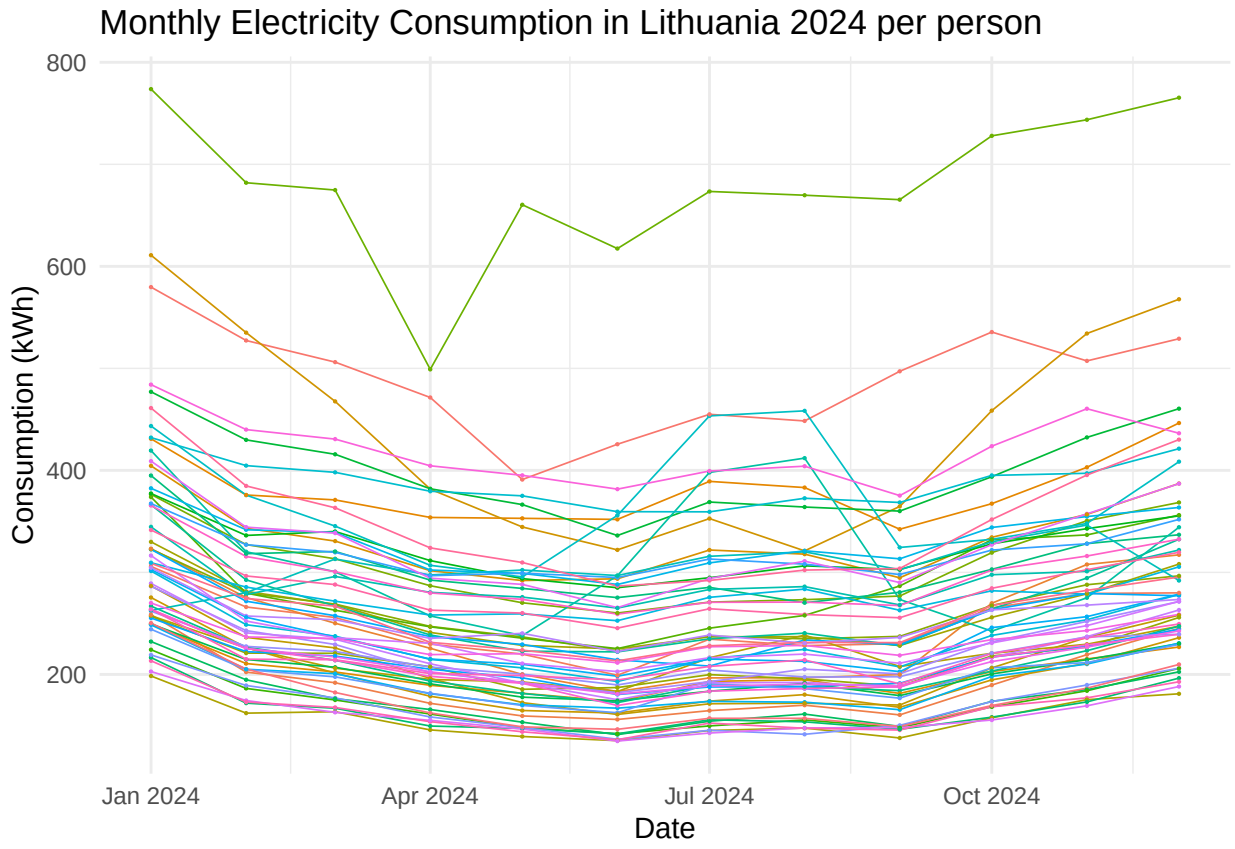
```
head(consumption_by_population_2024)
```

Exploring data

```
## # A tibble: 6 x 9
##   Suvartojimo_laikotarpis Savivaldybe     Bendras_suvartojimas_mWh populiacija
##   <date>                  <chr>                              <dbl>       <dbl>
## 1 2024-04-01              Akmenės r. sav.                    9356.       19845
## 2 2024-06-01              Akmenės r. sav.                    8447.       19845
## 3 2024-08-01              Akmenės r. sav.                    8899.       19845
## 4 2024-10-01              Akmenės r. sav.                   10629.       19845
## 5 2024-12-01              Akmenės r. sav.                   10499.       19845
## 6 2024-02-01              Biržų r. sav.                      4958.       23540
## # i 5 more variables: atlyginimas <dbl>, plotas <dbl>, tankis <dbl>,
## #   vid_nedarbas <dbl>, kWh_per_person <dbl>
```

```
plot_all_municipalities <- ggplot(consumption_by_population_2024, aes(
  x = Suvartojimo_laikotarpis,
  y = kWh_per_person,
  group = Savivaldybe,
  color = Savivaldybe
```

```
)) +
  geom_line(size = 0.3) +
  geom_point(size = 0.1) +
  labs(
    title = "Monthly Electricity Consumption in Lithuania 2024 per person",
    x = "Date",
    y = "Consumption (kWh)"
  ) +
  theme_minimal() +
  theme(legend.position = "none")

plot_all_municipalities
```



Smoothing

```
# prepare data
consumption_by_population_2024$Suvartojimo_laikotarpis <- as.Date(consumption_by_population_2024$Suvarto
consumption_by_population_2024$Month <- as.numeric(format(consumption_by_population_2024$Suvartojimo_la
  12 * (as.numeric(format(consumption_by_population_2024$Suvartojimo_laikotarpis, "%Y")) - min(as.numer
```

```r
# I found 6 to be optimal (not too smooth nor overfitting) maybe could be changed
num_basis <- 6
time_range <- range(consumption_by_population_2024$Month)
basis <- create.bspline.basis(time_range, nbasis = num_basis, norder = 4) # cubic splines
```

Cubic B-splines

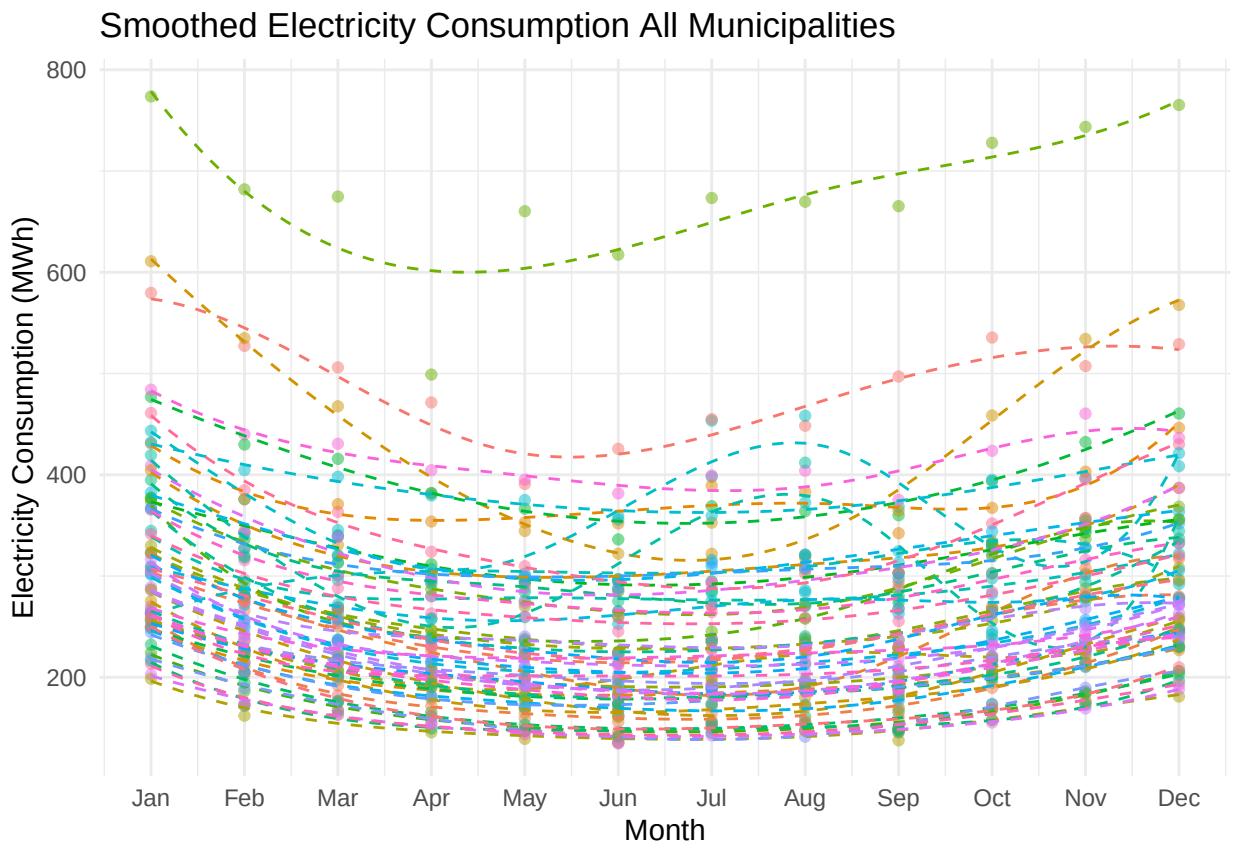
```r
energy_matrix <- consumption_by_population_2024 %>%
  dplyr::select(Savivaldybe, Month, kWh_per_person) %>%
  dplyr::arrange(Month, Savivaldybe) %>%  # ensure consistent order
  tidyr::pivot_wider(names_from = Savivaldybe, values_from = kWh_per_person) %>%
  dplyr::arrange(Month) %>%  # just to be safe
  dplyr::select(-Month) %>%
  as.matrix()


months <- sort(unique(consumption_by_population_2024$Month))

fd_obj_all_mun <- smooth.basis(argvals = months, y = energy_matrix, fdParobj = fdPar(basis, 2))$fd
# plot(fd_obj_all_mun)
```

```r
municipalities <- unique(consumption_by_population_2024$Savivaldybe)

smoothed_curves <- list()

for (municipality in municipalities) {
  subset <- consumption_by_population_2024 %>%
    filter(Savivaldybe == municipality) %>%
    arrange(Month)

  x <- subset$Month
  y <- subset$kWh_per_person

  # converting to functional data
  fd_obj <- smooth.basis(x, y, fdPar(basis, 2))$fd # second derivative smoothing


  smoothed_curves[[municipality]] <- data.frame(
    Month = seq(min(x), max(x), length.out = 100),
    Consumption = eval.fd(seq(min(x), max(x), length.out = 100), fd_obj),
    Savivaldybe = municipality
  )
}
```

```r
# combining smoothed results
smoothed_df <- bind_rows(smoothed_curves)
```

```r
# plotting all municipalities, I guess smoothing looks good?
ggplot() +
  geom_point(data = consumption_by_population_2024, aes(x = Month, y = kWh_per_person, color = Savivaldy
  geom_line(data = smoothed_df, aes(x = Month, y = Consumption, color = Savivaldybe), linetype = "dashe
  scale_x_continuous(breaks = 1:12, labels = month.abb) +  # converting 1-12 to month names
  labs(
    title = "Smoothed Electricity Consumption All Municipalities",
    x = "Month",
    y = "Electricity Consumption (MWh)"
  ) +
  theme_minimal() +
  theme(legend.position = "none")
```

fd object for all municipalities




```r
consumption_summary <- consumption_by_population_2024 %>%
  group_by(Savivaldybe) %>%
  summarise(yearly_consumption_per_person = sum(kWh_per_person, na.rm = TRUE))

top_municipalities <- consumption_summary %>%
  ungroup() %>%
  arrange(desc(yearly_consumption_per_person)) %>%
  mutate(Rank = row_number()) %>%
  slice(1:10) %>%
```

```
  dplyr::select(Rank, Savivaldybe, yearly_consumption_per_person)

kable(top_municipalities,
      col.names = c("Rank", "Municipality", "Yearly Consumption (kWh/person)"),
      caption = "Top 10 Municipalities by Total Electricity Consumption per Person in 2024")
```

Table 1: Top 10 Municipalities by Total Electricity Consumption per Person in 2024

Rank	Municipality	Yearly Consumption (kWh/person)
1	Kazlų Rūdos sav.	8152.377
2	Akmenės r. sav.	5873.570
3	Elektrėnų sav.	5260.728
4	Trakų r. sav.	5035.179
5	Klaipėdos r. sav.	4786.852
6	Panevėžio m. sav.	4663.038
7	Birštono sav.	4568.187
8	Palangos m. sav.	4448.107
9	Vilniaus r. sav.	4205.096
10	Druskininkų sav.	3978.873

Municipality with highest energy consumption per person is Kazlų Rūda, followed by Akmenė and Elektrėnai, which is likely due to significant industrial activity in these municipalities.

To evaluate whether the energy consumption is related with geographical location, average consumption per person was plotted on the Lithuanian municipality map.

```
library(sf)
```

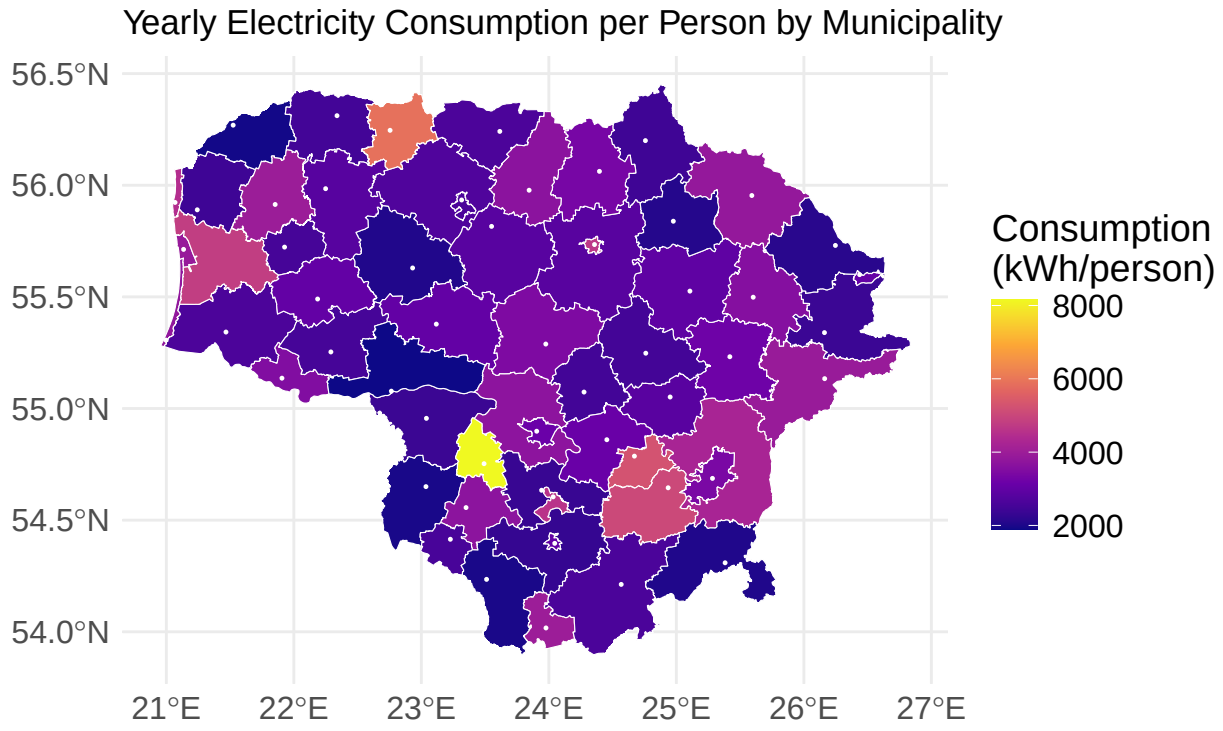
```
## Linking to GEOS 3.13.0, GDAL 3.10.1, PROJ 9.5.1; sf_use_s2() is TRUE
```

```
lt_map <- st_read("data/municipalities.geojson")
```

```
## Reading layer ‘municipalities’ from data source
##   ‘C:\Users\krist\Desktop\FDA\FDA_group_project\data\municipalities.geojson’
##   using driver ‘GeoJSON’
## Simple feature collection with 114 features and 39 fields
## Geometry type: GEOMETRY
## Dimension:     XY
## Bounding box:  xmin: 20.9537 ymin: 53.89679 xmax: 26.83552 ymax: 56.45042
## Geodetic CRS:  WGS 84
```

```
lt_map$name <- lt_map$name %>%
  gsub("miesto", "m.", .) %>%
  gsub("rajono", "r.", .) %>%
  gsub("savivaldybė", "sav.", .)

map_data <- lt_map %>%
  left_join(consumption_summary, by = c("name" = "Savivaldybe"))
```

```r
ggplot(map_data) +
  geom_sf(aes(fill = yearly_consumption_per_person), color = "white", size = 0.2) +
  scale_fill_viridis_c(option = "plasma", na.value = "lightgrey",
                       name = "Consumption\n(kWh/person)") +
  labs(title = "Yearly Electricity Consumption per Person by Municipality") +
  theme_minimal() +
  theme(
    axis.text = element_text(size = 12),
    axis.title = element_text(size = 14),
    legend.title = element_text(size = 14),
    legend.text = element_text(size = 12)
  )
```



It can be seen from the map that high consumption municipalities are distributed across the country instead of being clustered in one location.

Clustering

```r
set.seed(123)

res_v2 = funFEM(fd_obj_all_mun,K=2:5,model='AkjBk',init='kmeans',lambda=0,disp=TRUE)
```

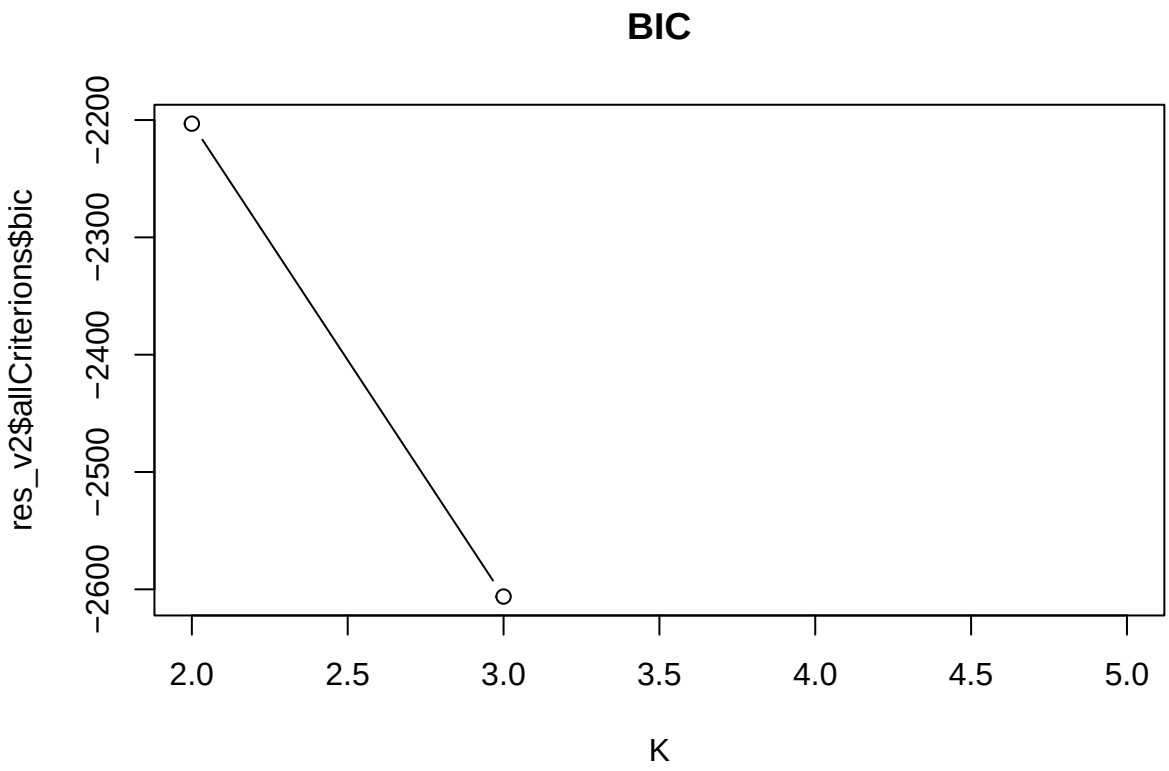
finding optimal K

```
## >> K = 2
## AkjBk    :    bic = -2203.093
## >> K = 3
## AkjBk    :    bic = -2606.155
## >> K = 4
## Error in .fstep(fd, T, lambda) : One cluster is almost empty!
## >> K = 5
## Error in .fstep(fd, T, lambda) : One cluster is almost empty!
## The best model is AkjBk with K = 2 ( bic = -2203.093 )
```

```
res_v2$allCriterions
```

```
##   K model       bic       aic       icl nbprm        ll
## 1 2 AkjBk -2203.093 -2190.527 -2202.551    12 -2178.527
## 2 3 AkjBk -2606.155 -2578.929 -2605.391    26 -2552.929
## 3 4 AkjBk        NA        NA        NA    NA        NA
## 4 5 AkjBk        NA        NA        NA    NA        NA
```

```
plot(res_v2$allCriterions$K,res_v2$allCriterions$bic,type='b',xlab='K',main='BIC')
```




```
res_v2$cls
```

```
##  [1] 1 2 2 2 1 2 2 1 2 2 2 2 2 2 2 2 1 2 2 1 2 2 2 2 2 2 2 1 2 2 1 2 2 2 2 2 2 2
## [39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

Optimal K was found to be 2.

Clustering K = 2 To identify distinct electricity consumption patters accross Lithuanian municiaplities, we employed performed k-means clustering with the AkjBk model, selecting $K = 2$ based on Bayesian Information Criterion (BIC) optimization. The regularization parameter λ was set to 0.

```
library(funFEM)

# Clustering
res_electricity <- funFEM(fd_obj_all_mun, K = 2, model = "AkjBk", init = "kmeans", lambda = 0, disp = TI
```

```
## >> K = 2
## AkjBk    :    bic = -2638.873
## The best model is AkjBk with K = 2 ( bic = -2638.873 )
```

```
# Cluster mean curves
fdmeans_electricity <- fd_obj_all_mun
fdmeans_electricity$coefs <- t(res_electricity$prms$my)

# Cluster assignment
municipality_clusters <- res_electricity$cls
municipality_names <- colnames(energy_matrix)

cluster_table <- data.frame(
  Savivaldybe = municipality_names,
  Cluster = municipality_clusters
)
print(cluster_table)
```

```
##            Savivaldybe Cluster
## 1      Akmenės r. sav.       1
## 2      Alytaus m. sav.       2
## 3      Alytaus r. sav.       2
## 4     Anykščių r. sav.       2
## 5        Birštono sav.       1
## 6        Biržų r. sav.       2
## 7     Druskininkų sav.       1
## 8       Elektrėnų sav.       1
## 9    Ignalinos r. sav.       2
## 10     Jonavos r. sav.       2
## 11    Joniškio r. sav.       1
## 12    Jurbarko r. sav.       2
## 13 Kaišiadorių r. sav.       2
## 14     Kalvarijos sav.       2
## 15       Kauno m. sav.       2
## 16       Kauno r. sav.       2
## 17    Kazlų Rūdos sav.       1
## 18      Kelmės r. sav.       2
## 19   Klaipėdos m. sav.       2
```

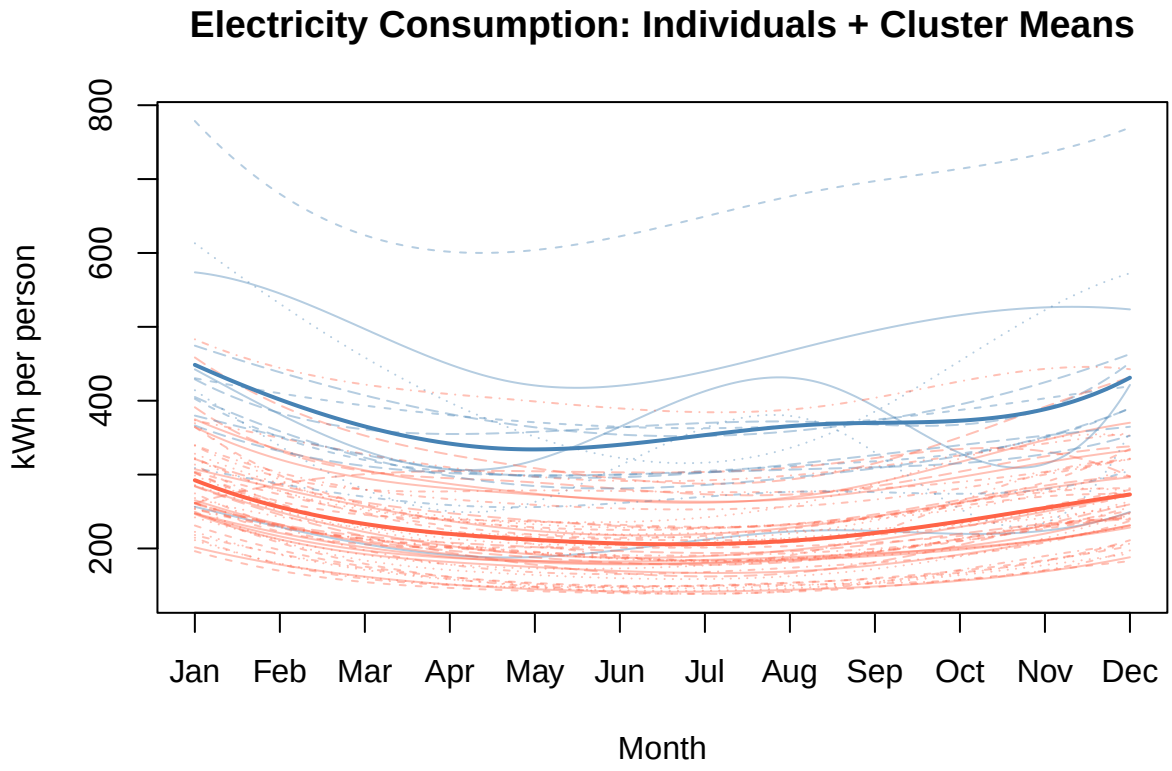
```
## 20   Klaipėdos r. sav.       1
## 21   Kretingos r. sav.       2
## 22    Kupiškio r. sav.       2
## 23    Kėdainių r. sav.       2
## 24     Lazdijų r. sav.       2
## 25   Marijampolės sav.       2
## 26    Mažeikių r. sav.       2
## 27      Molėtų r. sav.       2
## 28       Neringos sav.       1
## 29        Pagėgių sav.       2
## 30    Pakruojo r. sav.       2
## 31    Palangos m. sav.       1
## 32   Panevėžio m. sav.       1
## 33   Panevėžio r. sav.       2
## 34    Pasvalio r. sav.       1
## 35     Plungės r. sav.       1
## 36      Prienų r. sav.       2
## 37 Radviliškio r. sav.       2
## 38    Raseinių r. sav.       2
## 39        Rietavo sav.       2
## 40    Rokiškio r. sav.       1
## 41      Skuodo r. sav.       2
## 42    Tauragės r. sav.       2
## 43      Telšių r. sav.       2
## 44       Trakų r. sav.       2
## 45    Ukmergės r. sav.       2
## 46      Utenos r. sav.       2
## 47     Varėnos r. sav.       2
## 48 Vilkaviškio r. sav.       2
## 49    Vilniaus m. sav.       2
## 50    Vilniaus r. sav.       2
## 51       Visagino sav.       2
## 52      Zarasų r. sav.       2
## 53       Šakių r. sav.       2
## 54  Šalčininkų r. sav.       2
## 55     Šiaulių m. sav.       2
## 56     Šiaulių r. sav.       2
## 57     Šilalės r. sav.       2
## 58     Šilutės r. sav.       2
## 59    Širvintų r. sav.       2
## 60  Švenčionių r. sav.       1
```

```r
consumption_by_population_2024 <- consumption_by_population_2024 %>%
  left_join(cluster_table, by = "Savivaldybe")

# Colours
clusters <- res_electricity$cls
n_clusters <- length(unique(clusters))
cluster_colors <- c("steelblue", "tomato")


# Plot all curves colored by cluster
plot(fd_obj_all_mun, col = adjustcolor(cluster_colors[clusters], alpha.f = 0.4),
     main = "Electricity Consumption: Individuals + Cluster Means",
```

```
     xlab = "Month", ylab = "kWh per person", lwd = 1, xaxt = "n")
```

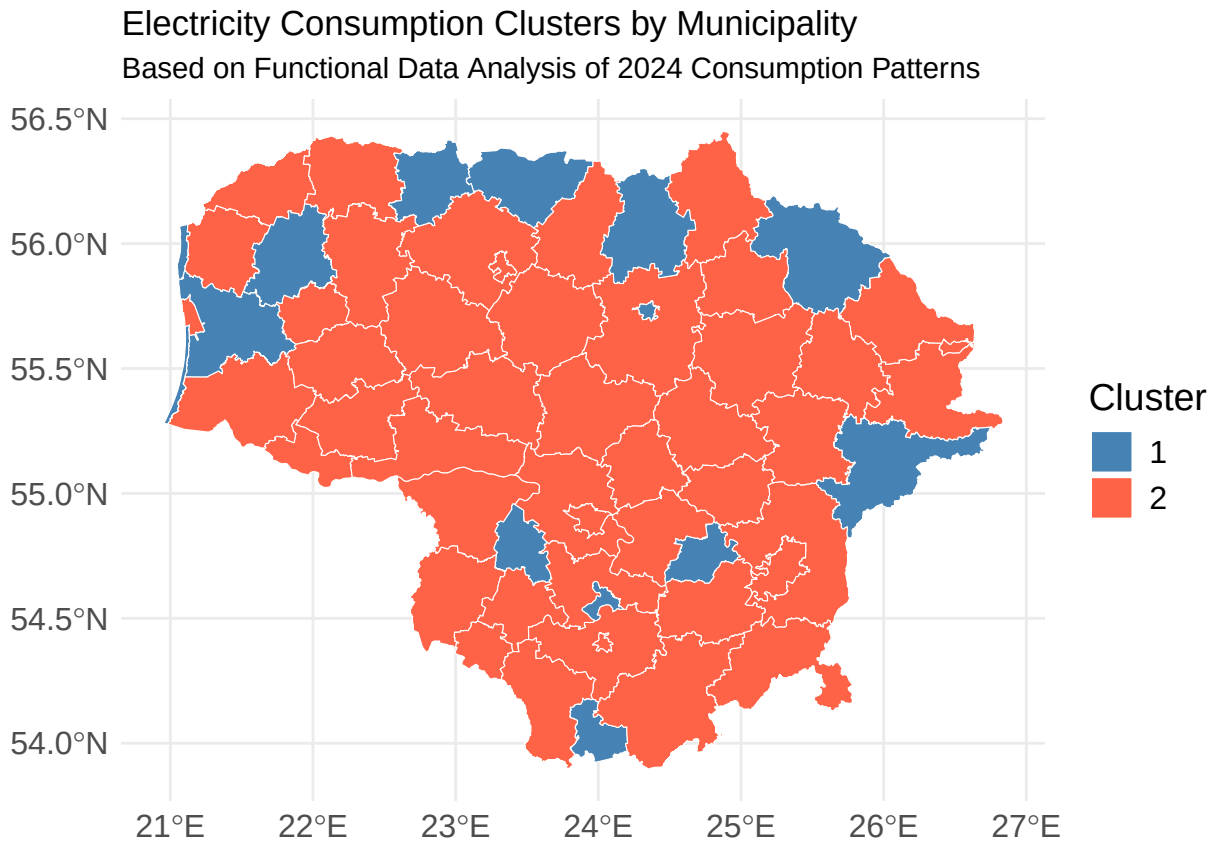
```
## [1] "done"
```

```
axis(1, at = 1:12, labels = month.abb)

# adding mean curves per cluster
for (i in 1:n_clusters) {
  lines(fdmeans_electricity[i], col = cluster_colors[i], lwd = 2)
}
```




```
# Cluster map
map_clusters <- lt_map %>%
  left_join(cluster_table, by = c("name" = "Savivaldybe"))

map_clusters <- map_clusters %>%
  filter(!is.na(Cluster))

cluster_palette <- c("steelblue", "tomato")

ggplot(map_clusters) +
  geom_sf(aes(fill = factor(Cluster)), color = "white", size = 0.2) +
  scale_fill_manual(values = cluster_palette,
                    name = "Cluster",
```

```
                    labels = c("1", "2")) +
labs(title = "Electricity Consumption Clusters by Municipality",
     subtitle = "Based on Functional Data Analysis of 2024 Consumption Patterns") +
theme_minimal() +
theme(
  axis.text = element_text(size = 12),
  axis.title = element_text(size = 14),
  legend.title = element_text(size = 14),
  legend.text = element_text(size = 12)
)
```



As can be seen in the map, municipalities in Cluster 1 are not clustered geographically.

Hypothesis testing

In order to identify the time periods where differences between the two clusters occur, pointwise functional ANOVA was carried out with the 0.05 alpha level. Since only two clusters are being compared, post-hoc test result is not shown as it provides identical results to the pointwise ANOVA itself.

fANOVA

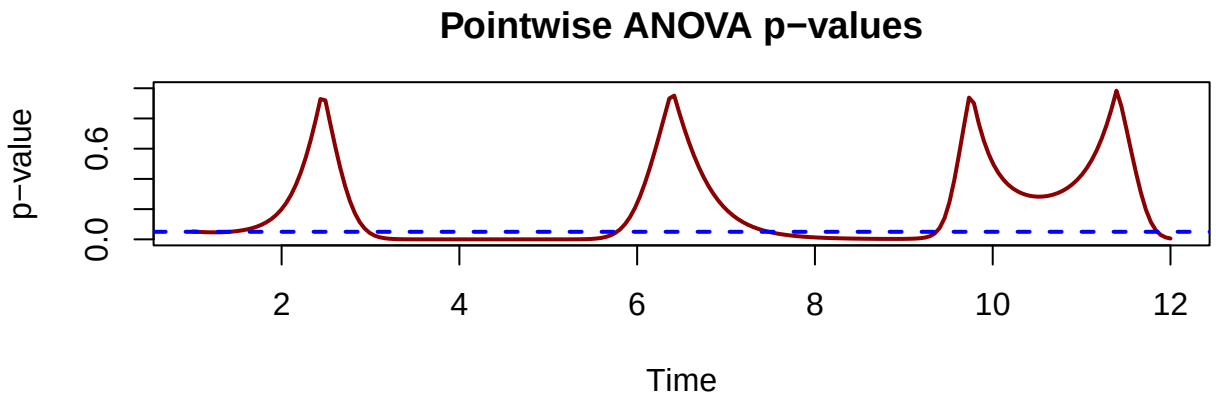
```r
fANOVA.pointwise <- function(data, groups, t.seq, alpha = 0.05) {
  if (length(groups) != ncol(data)) stop("Length of groups must match number of columns in data")
  if (!is.factor(groups)) groups <- factor(groups)
  library(dplyr)
  n_time <- nrow(data)
  n_groups <- length(levels(groups))
  group_levels <- levels(groups)
  pvals <- numeric(n_time)
  mean.p <- matrix(NA, ncol = n_groups, nrow = n_time)
  # Determine number of pairwise comparisons
  combs <- combn(group_levels, 2)
  perm <- ncol(combs)
  Tukey.posthoc <- matrix(NA, ncol = perm, nrow = n_time)
  colnames(Tukey.posthoc) <- apply(combs, 2, paste, collapse = " - ")
  # Main loop: pointwise ANOVA and Tukey HSD
  for (i in 1:n_time) {
  dt <- data.frame(values = data[i, ], groups = groups)
  av <- aov(values ~ groups, data = dt)
  pvals[i] <- summary(av)[[1]]["Pr(>F)"][1,1]
  mean.p[i, ] <- dt %>%
  group_by(groups) %>%
  summarise(mean_val = mean(values), .groups = "drop") %>%
  pull(mean_val)
  tukey_res <- TukeyHSD(av)$groups[, "p adj"]
  Tukey.posthoc[i, ] <- tukey_res
  }
  overall_mean <- rowMeans(data)
  ## --- Plotting --- ##
  opar <- par(mfrow = c(2, 1), mar = c(4, 4, 3, 1))
  # Plot ANOVA p-values
  plot(t.seq, pvals, type = "l", lwd = 2, col = "darkred",
  main = "Pointwise ANOVA p-values", xlab = "Time", ylab = "p-value", ylim = c(0, 1))
  abline(h = alpha, col = "blue", lty = 2, lwd = 2)
  # Plot group means
  col_set <- rainbow(n_groups)
  ylim_range <- range(c(mean.p, overall_mean)) * 1.05
  plot(t.seq, overall_mean, type = "l", lwd = 2, col = "black",
  main = "Group Mean Functions", xlab = "Time", ylab = "Mean", ylim = ylim_range)
  for (j in 1:n_groups) {
    lines(t.seq, mean.p[, j], col = col_set[j], lty = j + 1, lwd = 1.5)
  }
  legend("topright", legend = c("Overall", group_levels),
  col = c("black", col_set), lty = c(1, 2:(n_groups + 1)),
  lwd = c(2, rep(1.5, n_groups)))
  par(opar)
  ## --- Post-hoc Tukey plots --- ##
  # opar2 <- par(mfrow = c(1, 1), ask = TRUE)
  # for (i in 1:perm) {
  # plot(t.seq, Tukey.posthoc[, i], type = "l", col = "purple", lwd = 2,
  # main = paste("Tukey HSD p-values for", colnames(Tukey.posthoc)[i]),
  # xlab = "Time", ylab = "p-value", ylim = c(0, 1))
  # abline(h = alpha, col = "blue", lty = 2, lwd = 2)
```

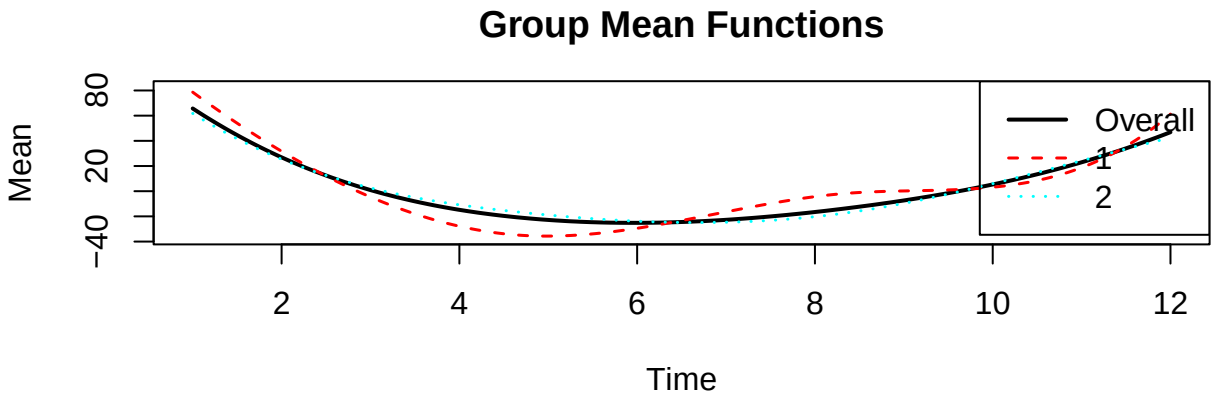
```
  # }
  # par(opar2)
  ## --- Return --- ##
  # return(list(
  #   p.values = pvals,
  #   TukeyHSD = Tukey.posthoc,
  #   group.means = mean.p,
  #   overall.mean = overall_mean,
  #   comparisons = colnames(Tukey.posthoc)
  #   )
  # )
}
```

function definition

```
time_grid <- seq(1, 12, length.out = 200)  # finer time points over months
elec_eval <- eval.fd(time_grid, fd_obj_all_mun)
# Center **each curve** separately
elec_eval_centered <- apply(elec_eval, 2, function(x) x - mean(x))

library(fda.usc)

fANOVA.pointwise(
  data = elec_eval_centered,
  groups = as.factor(clusters),  # clusters from funFEM
  t.seq = time_grid,
  alpha = 0.05
)
```

Identifying time periods that differ significantly





PCA analysis

```r
# Prepare data matrix for all municipalities (wide format)
energy_matrix <- consumption_by_population_2024 %>%
  arrange(Savivaldybe, Month) %>%
  dplyr::select(Savivaldybe, Month, kWh_per_person) %>%
  pivot_wider(names_from = Savivaldybe, values_from = kWh_per_person) %>%
  dplyr::select(-Month) %>%
  as.matrix()

months <- 1:12
```

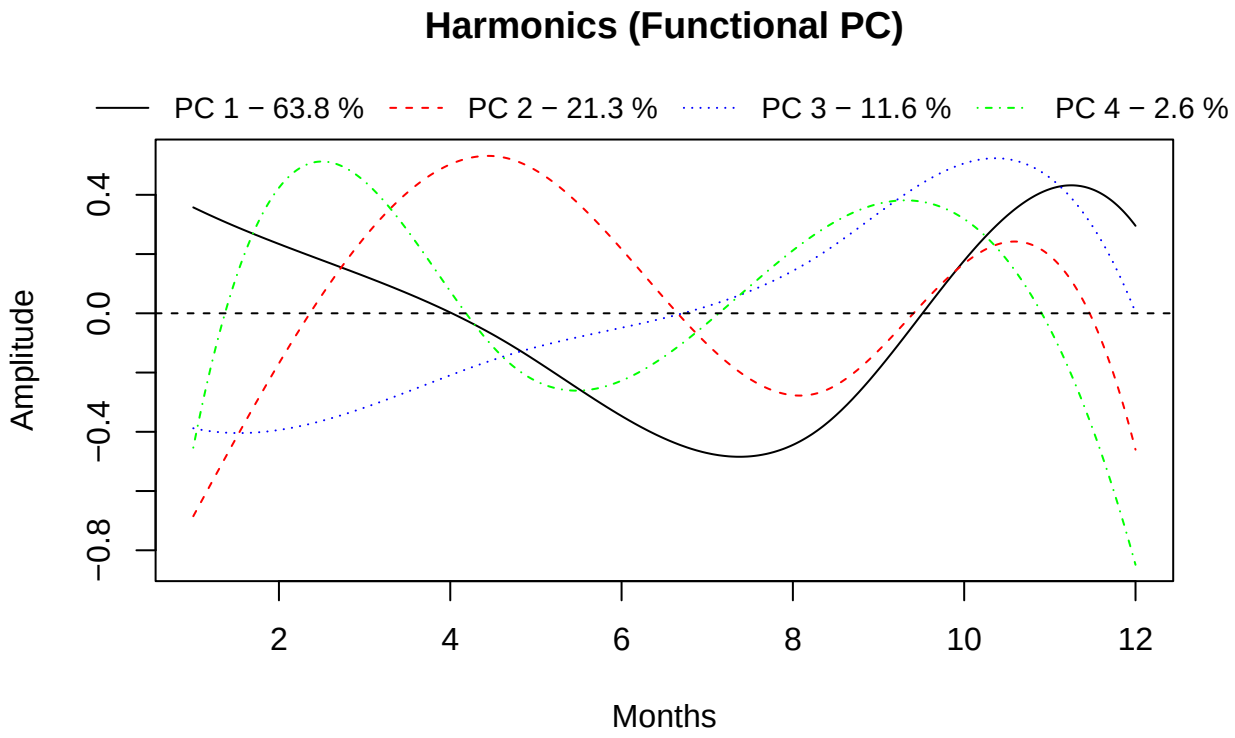
```r
# centering data by subtracting mean to concentrate on true seasonal consumption
# patterns instead of just finding which municipality consumes more overall

energy_matrix <- apply(energy_matrix, 2, function(x) x - mean(x))
fd_all <- smooth.basis(months, energy_matrix, basis)$fd
pca_results <- pca.fd(fd_all, nharm = 4, centerfns = TRUE)

explained_var <- round(pca_results$varprop * 100, 1)
colors <- c("black", "red", "blue", "green")
```

```r
line_types <- c(1, 2, 3, 4)
par(mar = c(5, 4, 6, 2))  # Increase top margin to make space for legend
plot(pca_results$harmonics, main = "Harmonics (Functional PC)",
     xlab = "Months", ylab = "Amplitude", lty = line_types, col = colors)
```

Harmonics, PCA scores and Reconstrunctions

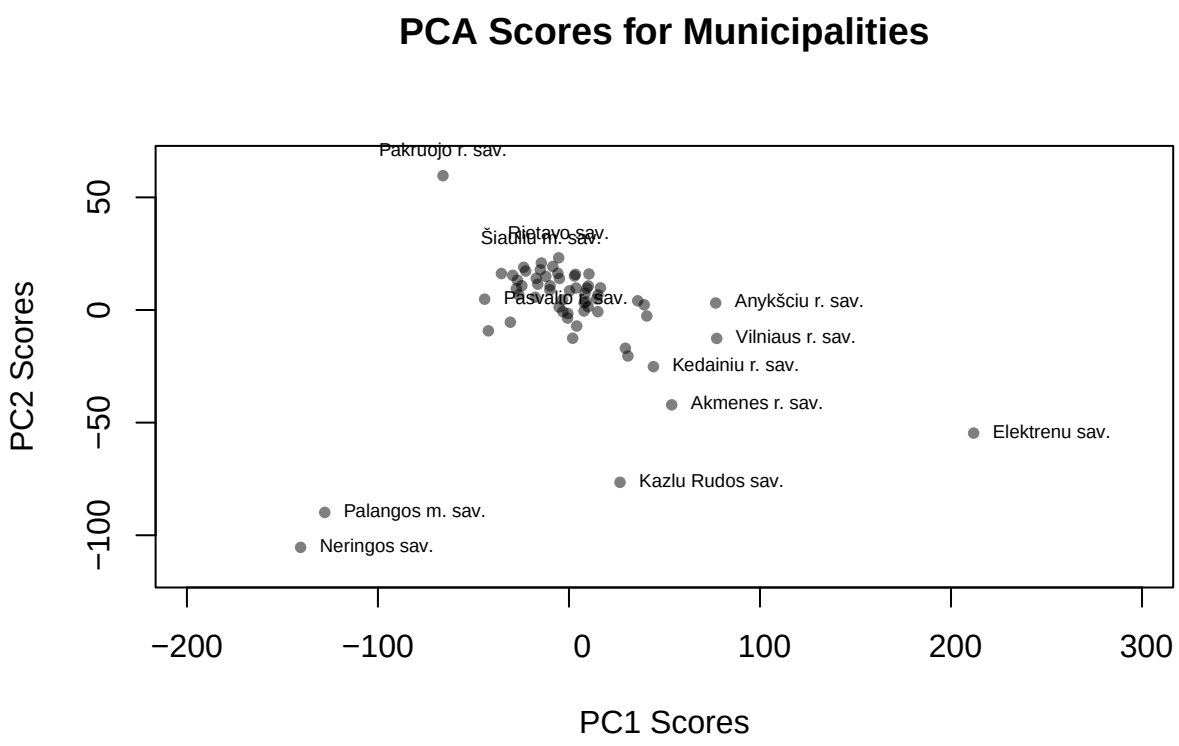
```
## [1] "done"
```

```r
par(xpd = TRUE)  # Allow plotting outside the plotting area
legend("top", inset = c(0, -0.15), legend = paste("PC", 1:4, "-", explained_var[1:4], "%"),
       col = colors, lty = line_types, cex = 0.9, horiz = TRUE, bty = "n")
```



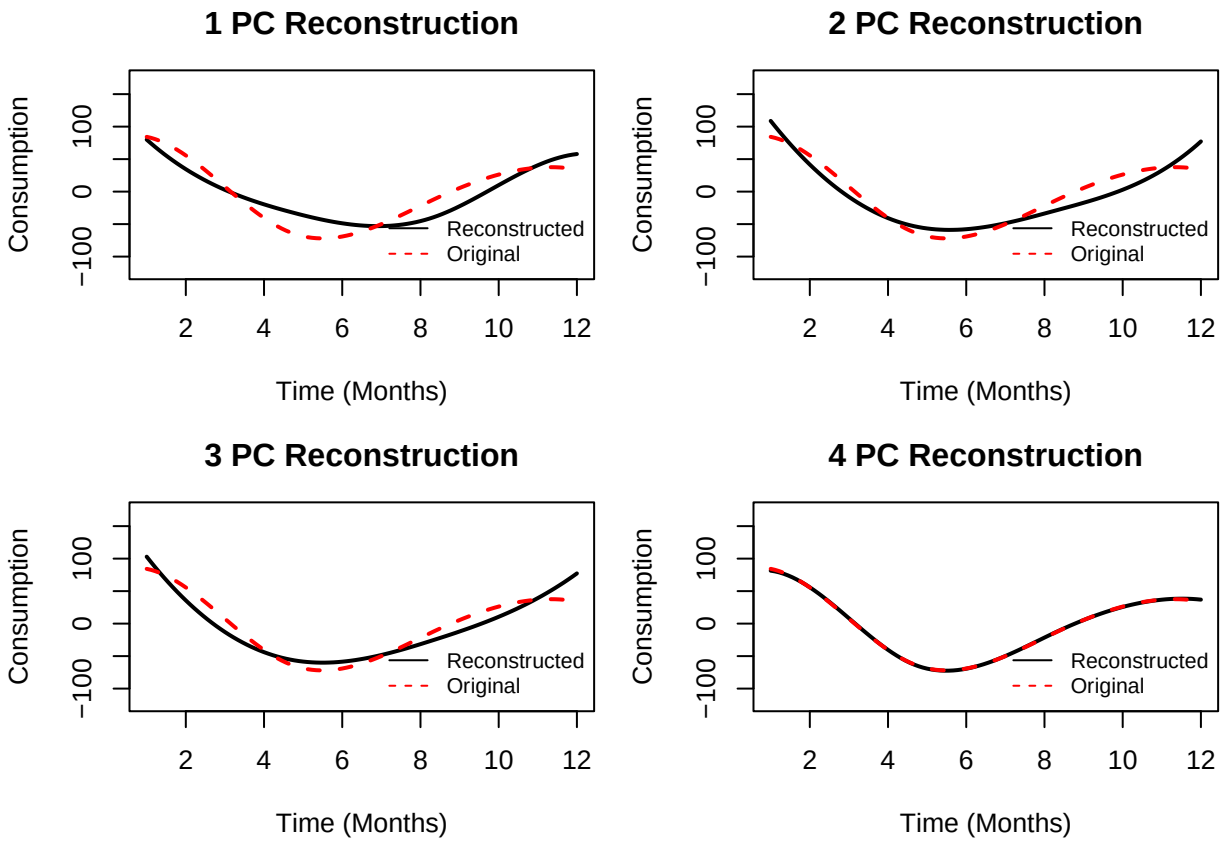

```r
municipality_labels <- colnames(energy_matrix)
x_limits <- range(pca_results$scores[,1]) * 1.4
y_limits <- range(pca_results$scores[,2]) * 1.1

plot(pca_results$scores[,1], pca_results$scores[,2],
     xlab = "PC1 Scores", ylab = "PC2 Scores",
     main = "PCA Scores for Municipalities",
     pch = 16, col = rgb(0, 0, 0, 0.5), cex = 0.8,
     xlim = x_limits, ylim = y_limits, xaxt = "n")

x_ticks <- pretty(pca_results$scores[,1])
```

```
x_labels <- format(x_ticks, big.mark = ",", scientific = FALSE)
axis(1, at = x_ticks, labels = x_labels)
text_positions <- ifelse(pca_results$scores[,2] > quantile(pca_results$scores[,2], 0.9), 3, 4)
label_indices <- abs(pca_results$scores[,1]) > quantile(abs(pca_results$scores[,1]), 0.85) |
                 abs(pca_results$scores[,2]) > quantile(abs(pca_results$scores[,2]), 0.85)
text(pca_results$scores[,1], pca_results$scores[,2],
     labels = ifelse(label_indices, municipality_labels, ""), pos = text_positions, cex = 0.6)
```




```
par(mfrow = c(2,2), mar = c(4, 4, 3, 1))

time_grid <- seq(1, 12, length.out = 100)
ylim_range <- range(eval.fd(time_grid, fd_all))

for(i in 1:4) {
  pc_curve <- pca_results$meanfd

  for(j in 1:i) {
    pc_curve <- pc_curve + pca_results$scores[1, j] * pca_results$harmonics[j]
  }

  plot(time_grid, eval.fd(time_grid, pc_curve), type = "l",
       ylim = ylim_range, main = paste(i, "PC Reconstruction"),
       xlab = "Time (Months)", ylab = "Consumption", col = "black", lwd = 2)

  lines(time_grid, eval.fd(time_grid, fd_all[1]), col = "red", lwd = 2, lty = 2)
```

```
  legend("bottomright", legend = c("Reconstructed", "Original"), col = c("black", "red"),
         lty = c(1, 2), bty = "n", cex = 0.8)
}
```



Harmonics (Functional PC) plot:

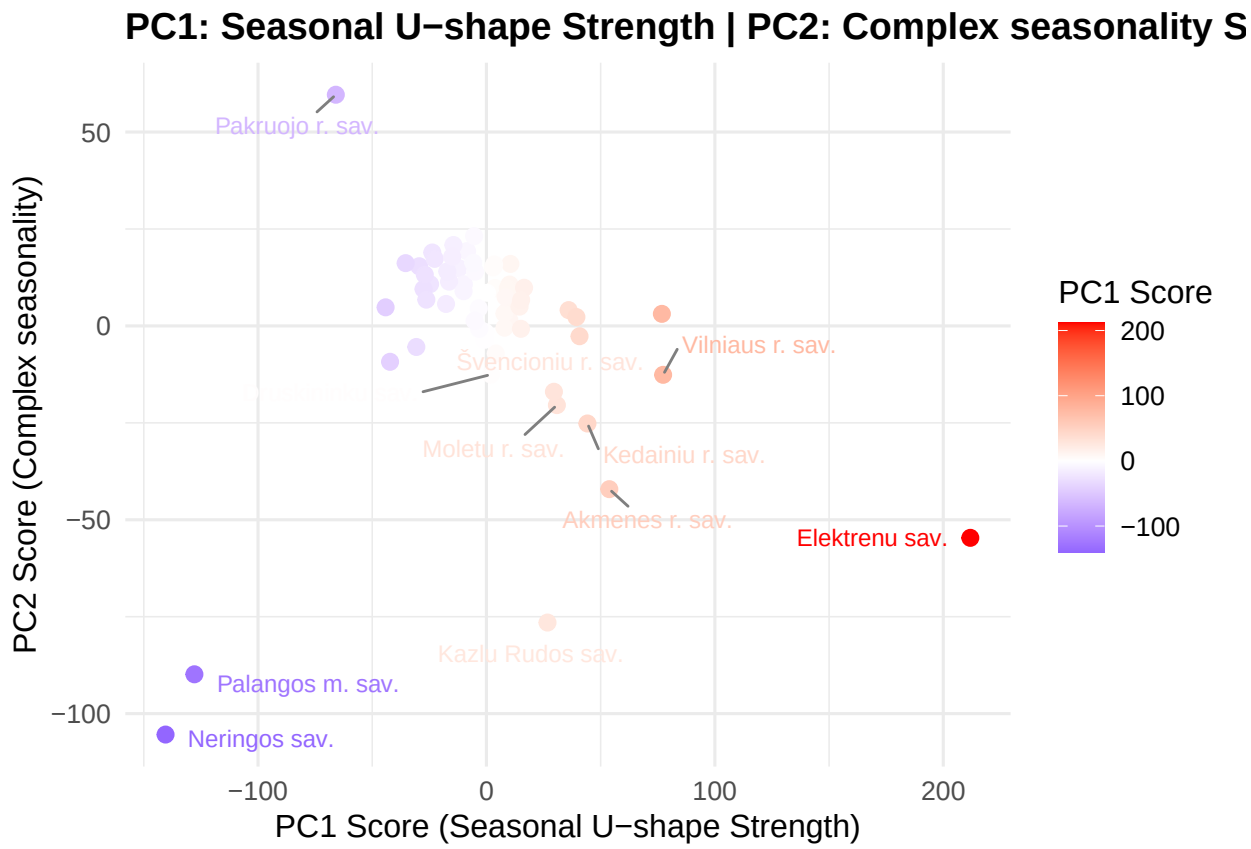
PC1 captures a broad seasonal trend, reflecting the overall winter-summer consumption cycle. PC2 describes more complex seasonal patterns, distinguishing municipalities with two seasonal peaks or dips across the year. PC3 captures a gradual increase in consumption toward the end of the year, highlighting municipalities where consumption steadily rises into autumn and winter. PC4 represents smaller local timing adjustments within the seasonal cycle.

Reconstruction plots:

The first principal component alone captures the main seasonal trend. Using all four principal components, the reconstructed curve nearly perfectly matches the original, showing that the four components can explain the variability quite precise.

```
municipalities_scores <- data.frame(
  PC1 = pca_results$scores[,1],
  PC2 =  pca_results$scores[,2],
  Municipality = municipality_labels
)

# Color by PC1
ggplot(municipalities_scores, aes(x = PC1, y = PC2, label = Municipality, color = PC1)) +
```

```
  geom_point(size = 2.5) +
  geom_text_repel(size = 3, max.overlaps = 20, box.padding = 0.5, segment.color = "grey50") +  # better
  scale_color_gradient2(low = "blue", mid = "white", high = "red", midpoint = 0) +
  labs(
    title = "PC1: Seasonal U-shape Strength | PC2: Complex seasonality Shift",
    x = "PC1 Score (Seasonal U-shape Strength)",
    y = "PC2 Score (Complex seasonality)",
    color = "PC1 Score"
  ) +
  theme_minimal(base_size = 12) +
  theme(
    plot.title = element_text(face = "bold"),
    legend.position = "right"
  )
```

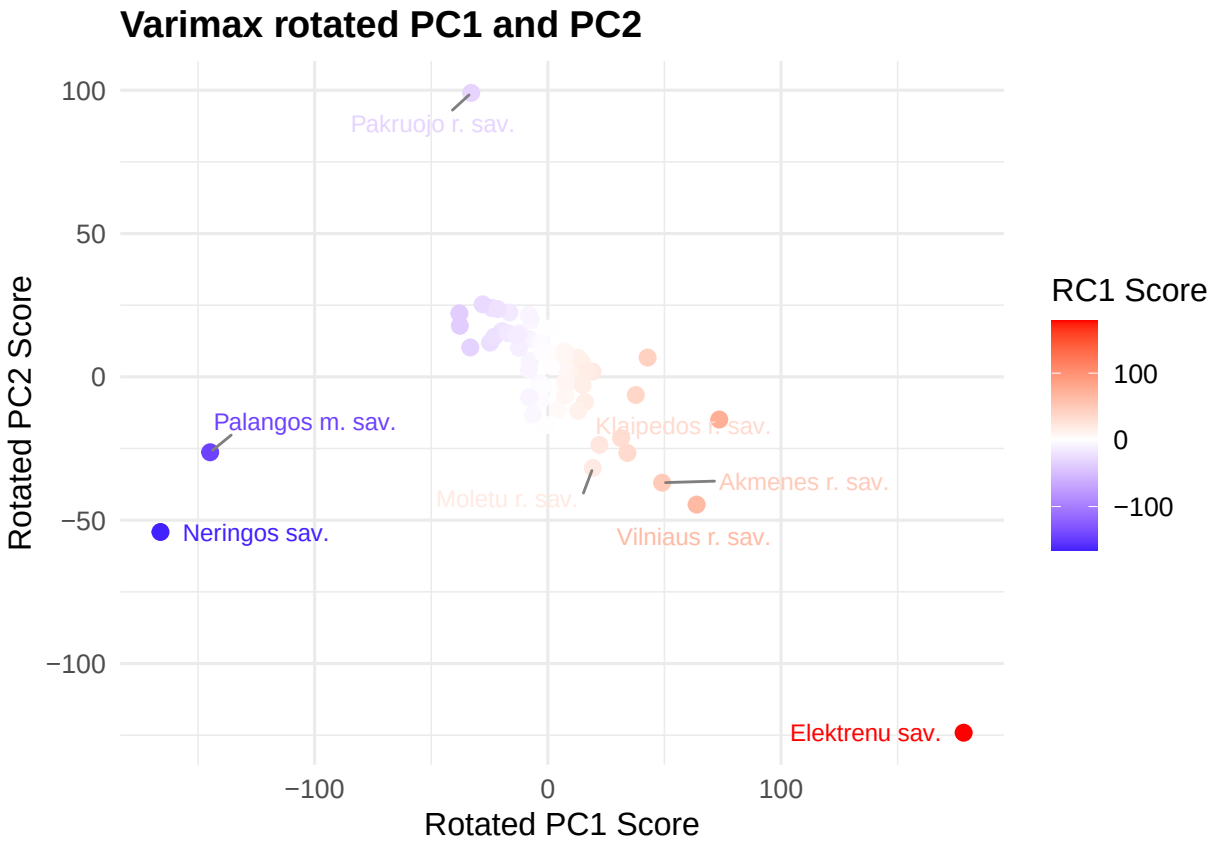


Seasonal U-shape Strength vs. Complex seasonality Shift plot:

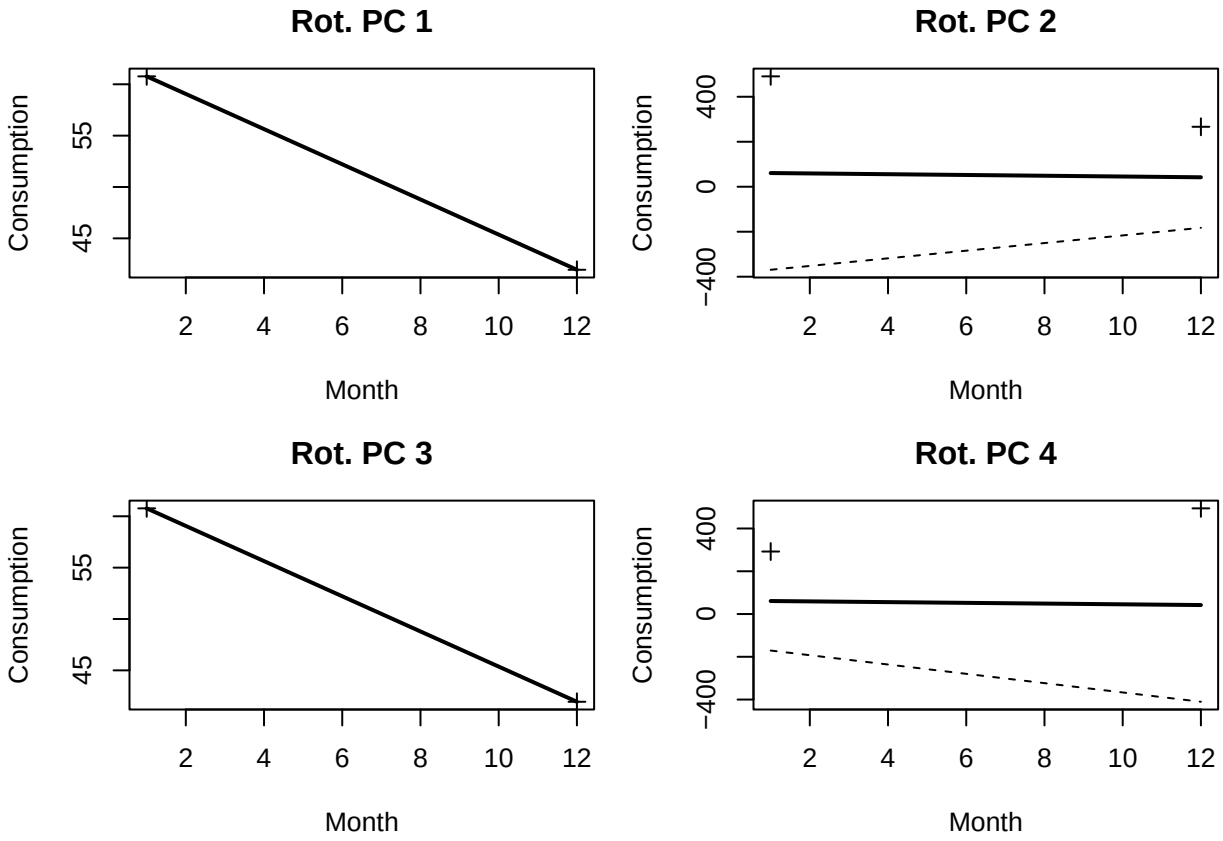
Most municipalities cluster near the center of the PC1–PC2 space, indicating broadly similar seasonal consumption patterns in 2024. Elektrėnų sav. stands out with extreme seasonality, showing strong winter and summer peaks compared to others. Tourism-affected municipalities like Neringos and Palangos m. sav. deviate, exhibiting alternative seasonal profiles.

```r
harmonics_matrix <- eval.fd(time_range, pca_results$harmonics)
varimax_result <- varimax(harmonics_matrix)
harmonics_varimax <- varimax_result$loadings

# Re-project municipalities' scores
rotated_scores <- as.matrix(pca_results$scores[, 1:4]) %*% varimax_result$rotmat

# Create new dataframe
municipalities_rotated <- data.frame(
  RC1 = rotated_scores[,1],
  RC2 = rotated_scores[,2],
  Municipality = municipality_labels
)

ggplot(municipalities_rotated, aes(x = RC1, y = RC2, label = Municipality, color = RC1)) +
  geom_point(size = 2.5) +
  geom_text_repel(size = 3, max.overlaps = 20, box.padding = 0.5, segment.color = "grey50") +  # better
  scale_color_gradient2(low = "blue", mid = "white", high = "red", midpoint = 0) +
  labs(
    title = "Varimax rotated PC1 and PC2",
    x = "Rotated PC1 Score",
    y = "Rotated PC2 Score",
    color = "RC1 Score"
  ) +
  theme_minimal(base_size = 12) +
  theme(
    plot.title = element_text(face = "bold"),
    legend.position = "right"
  )
```

Localizing PCs effects with VARIMAX rotation




```
mean_curve <- rowMeans(eval.fd(time_range, pca_results$meanfd))

amplification_factor <- 500  # Might need tuning

par(mfrow = c(2,2), mar = c(4,4,3,1))  # 2x2 layout

for (i in 1:4) {
  deform_plus <- mean_curve + harmonics_varimax[,i] * amplification_factor
  deform_minus <- mean_curve - harmonics_varimax[,i] * amplification_factor

  plot(time_range, mean_curve, type = "l", lwd = 2, ylim = range(c(deform_plus, deform_minus, mean_curve
       main = paste("Rot. PC", i),
       xlab = "Month", ylab = "Consumption")
  lines(time_range, deform_plus, lty = 2)
  points(time_range, deform_minus, pch = 3)  # plus signs
}
```



Because the consumption data were already normalized per person and centered per municipality, the rotated components mostly captured simple monotonic trends with limited seasonal structure. Therefore, for practical interpretation, we rely on the unrotated principal components which better reflect the true seasonal variation across municipalities.

Exploratory data analysis

```
data_list <- split(consumption_by_population_2024, consumption_by_population_2024$Savivaldybe)

fd_list <- lapply(data_list, function(df) {
  # converting to functional data
  smooth_fd <- smooth.basis(df$Month, df$kWh_per_person, fdPar(basis, 2))$fd

  return(smooth_fd)
})

# Combine coefficient matrices from each municipality
fd_coefs <- do.call(cbind, lapply(fd_list, function(fd) fd$coefs))

# Set column names as municipality names
colnames(fd_coefs) <- names(data_list)

# Rebuild the multivariate functional data object
smooth_fd <- fd(coef = fd_coefs, basisobj = basis)
```

```r
municipality_names <- colnames(smooth_fd$coefs)

cluster_mapping <- setNames(cluster_table$Cluster, cluster_table$Savivaldybe)

# municipality indices for each cluster
cluster1_indices <- which(municipality_names %in% names(cluster_mapping)[cluster_mapping == 1])
cluster2_indices <- which(municipality_names %in% names(cluster_mapping)[cluster_mapping == 2])

cluster1_fd <- smooth_fd[cluster1_indices]
cluster2_fd <- smooth_fd[cluster2_indices]
```

Derivatives (velocity and acceleration) First derivative shows the rate of change of electricity consumption (how fast consumption is increasing or decreasing). The second derivative shows acceleration or curvature (how fast the rate of change is changing).

```r
# Cluster 1
deriv1_c1 <- deriv.fd(cluster1_fd, 1) # first derivative
deriv2_c1 <- deriv.fd(cluster1_fd, 2) # second derivatives

# Cluster 2
deriv1_c2 <- deriv.fd(cluster2_fd, 1)
deriv2_c2 <- deriv.fd(cluster2_fd, 2)

time_grid <- seq(1, 12, length.out = 100)

par(mfrow=c(3,2), mar=c(4,1,3,1))

# Original curves
# Cluster 1
plot(cluster1_fd, xlab="Month", ylab="Consumption",
     main="Cluster 1: Original Curves", col="gray70", lwd=0.5)
```

```
## [1] "done"
```

```r
lines(mean.fd(cluster1_fd), col="blue", lwd=2)

# Cluster 2
plot(cluster2_fd, xlab="Month", ylab="Consumption",
     main="Cluster 2: Original Curves", col="gray70", lwd=0.5)
```

```
## [1] "done"
```

```r
lines(mean.fd(cluster2_fd), col="red", lwd=2)

# First derivatives (velocity)
# Cluster 1
plot(deriv1_c1, xlab="Month", ylab="Rate of Change",
     main="Cluster 1: First Derivatives", col="gray70", lwd=0.5)
```

```
## [1] "done"
```

```r
lines(mean.fd(deriv1_c1), col="blue", lwd=2)
abline(h=0, lty=3)

# Cluster 2
plot(deriv1_c2, xlab="Month", ylab="Rate of Change",
     main="Cluster 2: First Derivatives", col="gray70", lwd=0.5)
```

```
## [1] "done"
```

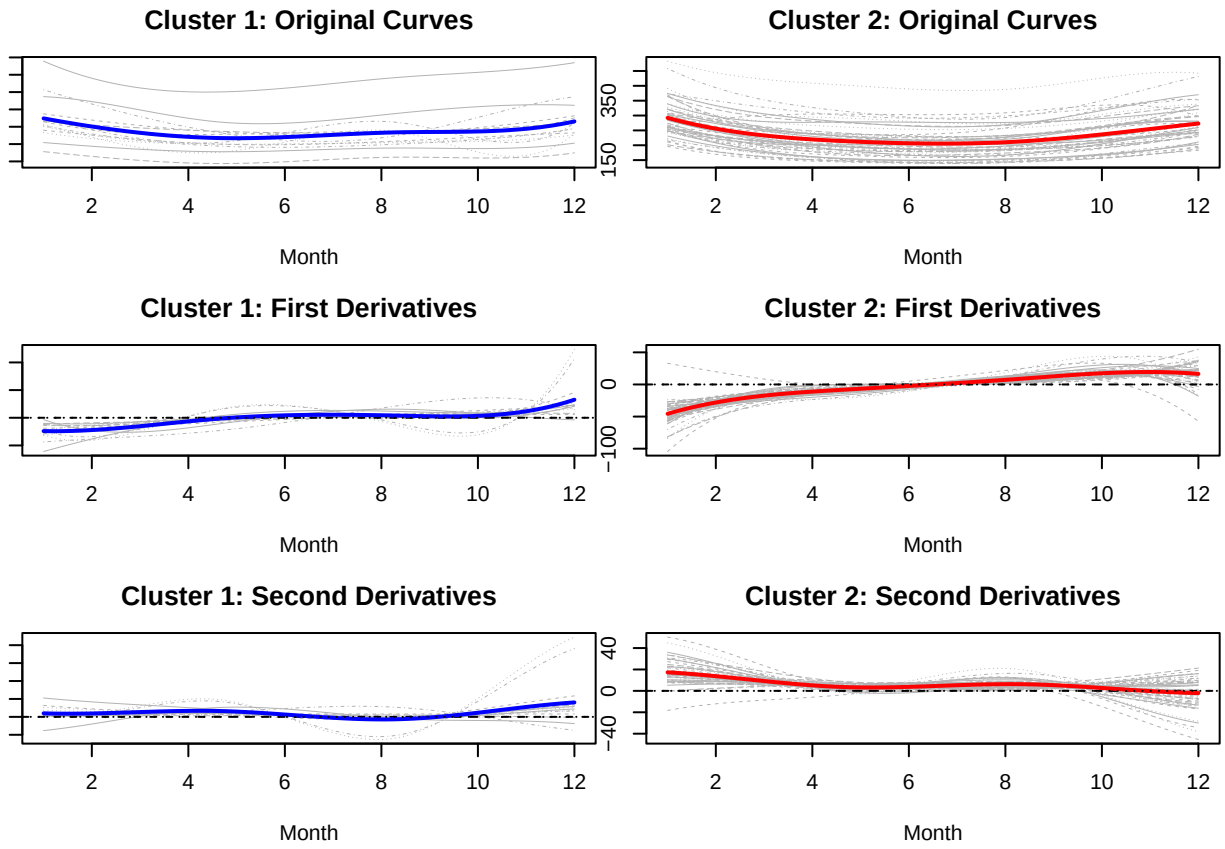
```r
lines(mean.fd(deriv1_c2), col="red", lwd=2)
abline(h=0, lty=3)

# Second derivatives (acceleration)
# Cluster 1
plot(deriv2_c1, xlab="Month", ylab="Acceleration",
     main="Cluster 1: Second Derivatives", col="gray70", lwd=0.5)
```

```
## [1] "done"
```

```r
lines(mean.fd(deriv2_c1), col="blue", lwd=2)
abline(h=0, lty=3)

# Cluster 2
plot(deriv2_c2, xlab="Month", ylab="Acceleration",
     main="Cluster 2: Second Derivatives", col="gray70", lwd=0.5)
```

```
## [1] "done"
```

```r
lines(mean.fd(deriv2_c2), col="red", lwd=2)
abline(h=0, lty=3)
```



Two municipalities in Cluster 1 have unusually high first and second derivative values at December.

```
# find the first and second derivatives at month 12 for all municipalities in Cluster 1
month_12 <- 12
time_for_eval <- month_12

deriv1_values <- eval.fd(time_for_eval, deriv1_c1)
deriv2_values <- eval.fd(time_for_eval, deriv2_c1)

deriv1_values <- as.numeric(deriv1_values)
deriv2_values <- as.numeric(deriv2_values)

# data frame with municipalities and their derivative values
cluster1_municipalities <- colnames(cluster1_fd$coefs)
derivatives_df <- data.frame(
  Municipality = cluster1_municipalities,
  First_Derivative = deriv1_values,
  Second_Derivative = deriv2_values
)

derivatives_df <- na.omit(derivatives_df)

# sort by first derivative
derivatives_sorted_first <- derivatives_df[order(derivatives_df$First_Derivative, decreasing = TRUE), ]
# sort by second derivative
derivatives_sorted_second <- derivatives_df[order(derivatives_df$Second_Derivative, decreasing = TRUE),
```

```r
# top municipalities by first derivative
cat("Top municipalities in Cluster 1 by First Derivative at December:\n")
```

```
## Top municipalities in Cluster 1 by First Derivative at December:
```

```r
print(head(derivatives_sorted_first, 5))
```

```
##          Municipality First_Derivative Second_Derivative
## 8       Neringos sav.        248.04922         222.48744
## 9    Palangos m. sav.        214.14278         190.47566
## 2       Birštono sav.         91.84954          58.52572
## 3    Druskininkų sav.         53.48942          26.18620
## 14 Švenčionių r. sav.         47.29856          16.44357
```

```r
# Print top municipalities by second derivative
cat("\nTop municipalities in Cluster 1 by Second Derivative at December:\n")
```

```
##
## Top municipalities in Cluster 1 by Second Derivative at December:
```

```r
print(head(derivatives_sorted_second, 5))
```

```
##        Municipality First_Derivative Second_Derivative
## 8     Neringos sav.        248.04922         222.48744
## 9  Palangos m. sav.        214.14278         190.47566
## 2     Birštono sav.         91.84954          58.52572
## 5  Joniškio r. sav.         44.74470          33.06836
## 11 Pasvalio r. sav.         41.62632          28.91159
```

The two municipalities with high first and second derivative values at December are two coastal resorts, Neringa and Palanga.

```r
# municipalities to highlight
highlight_municipalities <- c("Neringos sav.", "Palangos m. sav.")

cluster1_municipalities <- colnames(cluster1_fd$coefs)
highlight_indices <- match(highlight_municipalities, cluster1_municipalities)

par(mfrow=c(1,3), mar=c(4,4,3,2), oma=c(0,0,2,0), pty="s")

# 1. Original curves
plot(cluster1_fd, col="lightgray", lwd=0.5,
     main="Cluster 1: Original Curves",
     xlab="Month", ylab="Consumption per Person (kWh)")
```

```
## [1] "done"
```

```r
lines(mean.fd(cluster1_fd), col="blue", lwd=2)

if(!is.na(highlight_indices[1])) {
  lines(cluster1_fd[highlight_indices[1]], col="darkred", lwd=2)
}

if(!is.na(highlight_indices[2])) {
  lines(cluster1_fd[highlight_indices[2]], col="darkgreen", lwd=2)
}
grid()

# 2. First derivatives
plot(deriv1_c1, col="lightgray", lwd=0.5,
     main="Cluster 1: First Derivatives",
     xlab="Month", ylab="Rate of Change (kWh/month)")
```

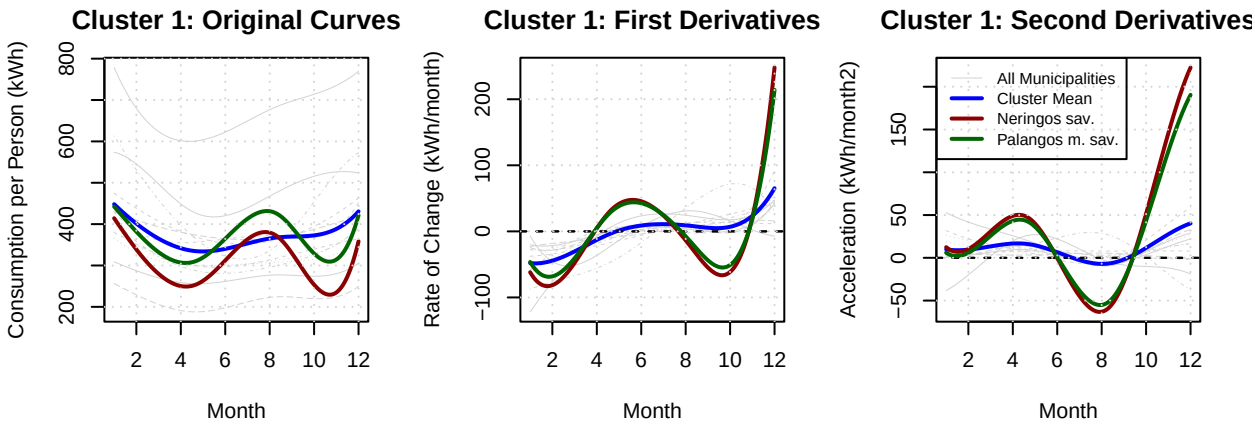
```
## [1] "done"
```

```r
lines(mean.fd(deriv1_c1), col="blue", lwd=2)
abline(h=0, lty=2)

if(!is.na(highlight_indices[1])) {
  lines(deriv1_c1[highlight_indices[1]], col="darkred", lwd=2)
}

if(!is.na(highlight_indices[2])) {
  lines(deriv1_c1[highlight_indices[2]], col="darkgreen", lwd=2)
}
grid()

# 3. Second derivatives
plot(deriv2_c1, col="lightgray", lwd=0.5,
     main="Cluster 1: Second Derivatives",
     xlab="Month", ylab="Acceleration (kWh/month2)")
```

```
## [1] "done"
```

```r
lines(mean.fd(deriv2_c1), col="blue", lwd=2)
abline(h=0, lty=2)

if(!is.na(highlight_indices[1])) {
  lines(deriv2_c1[highlight_indices[1]], col="darkred", lwd=2)
}

if(!is.na(highlight_indices[2])) {
  lines(deriv2_c1[highlight_indices[2]], col="darkgreen", lwd=2)
}
grid()

mtext("Electricity Consumption Dynamics for Cluster 1", outer=TRUE, cex=1.5)

legend("topleft",
```

```
        legend=c("All Municipalities", "Cluster Mean", highlight_municipalities),
        col=c("lightgray", "blue", "darkred", "darkgreen"),
        lwd=c(0.5, 2, 2, 2),
        cex=0.8)
```



Neringa and Palanga show more pronounced seasonal variation that the rest of the municipalities in Cluster 1 and are similar to each other. The rate of change for both highlighted municipalities becomes dramatically positive in December, indicating a rapid increase in consumption. They also show stronger positive rates at the end of spring and the beginning of summer, indicating that energy consumption increases rapidly during this time. These patterns suggest that Neringa and Palanga (both coastal resort towns) have distinctive seasonal electricity consumption behaviors with sharper seasonal transitions. This observation could be explained by the fact that both Neringa and Palanga tourism-oriented coastal areas, where seasonal population changes and tourism patterns strongly influence electricity consumption, especially during summer and around winter holidays.

Point-wise mean and standard deviation

All Municipalities To understand the central tendency and variability of electricity consumption across different municipalities, we calculated point-wise mean and standard deviation. This approach computes statistical measures at each time point throughout the year.

Additionallym, to analyze the centrality and variability measures further and check cluster homogeniety, we performed point-wise mean and standard deviation calculations for the two clusters separately.

```r
# elementary pointwise mean and standard deviation
mean_energy = mean.fd(smooth_fd)
stddev_energy = std.fd(smooth_fd)


par(mfrow=c(1,3), pty="s")

# Plot all
plot(smooth_fd, main = "All Municipalities", xlab = "Month", ylab = "Energy Consumption per Person (kWh
  ylim = c(0, 800), lty = 1, col = "gray80", lwd = 1)
```

```
## [1] "done"
```

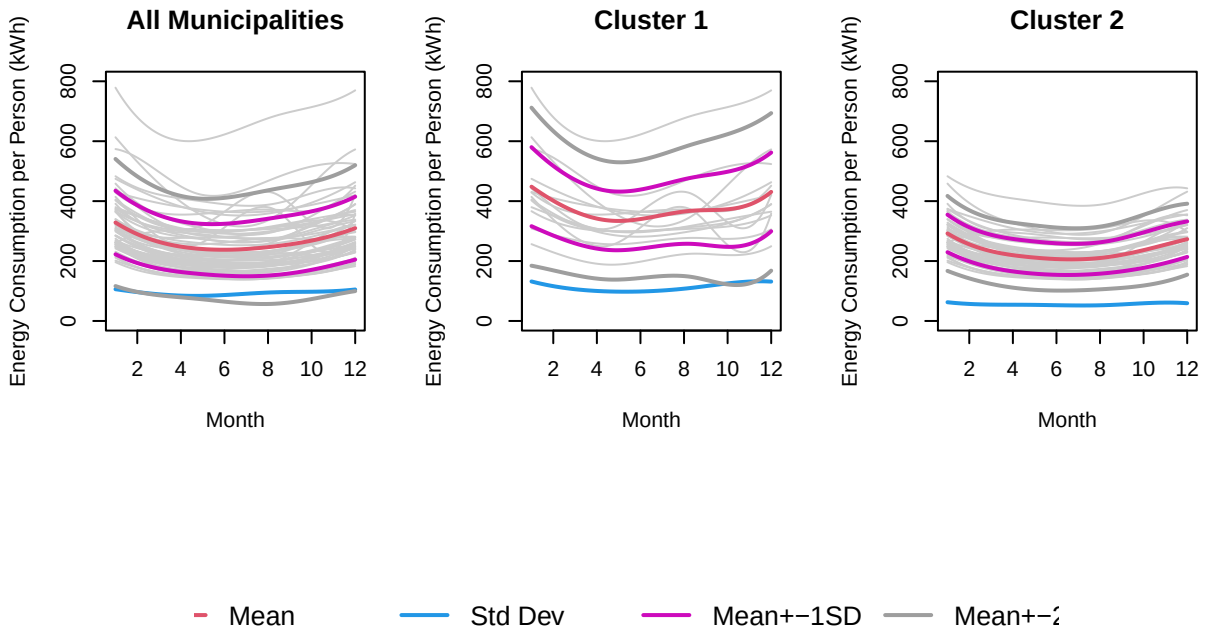
```r
# elementary pointwise mean and standard deviation
mean_energy = mean.fd(smooth_fd)
stddev_energy = std.fd(smooth_fd)

lines(mean_energy, lwd=2, lty=1, col=2)
lines(stddev_energy, lwd=2, lty=1, col=4)

lines(mean_energy-stddev_energy, lwd=2, lty=1, col=6)
lines(mean_energy+stddev_energy, lwd=2, lty=1, col=6)

lines(mean_energy-2*stddev_energy, lwd=2, lty=1, col=8)
lines(mean_energy+2*stddev_energy, lwd=2, lty=1, col=8)

# plot cluster 1
plot(cluster1_fd, main = "Cluster 1", xlab = "Month", ylab = "Energy Consumption per Person (kWh)",
  ylim = c(0, 800), lty = 1, col = "gray80", lwd = 1)
```

```
## [1] "done"
```

```r
# elementary pointwise mean and standard deviation
mean_energy = mean.fd(cluster1_fd)
stddev_energy = std.fd(cluster1_fd)

lines(mean_energy, lwd=2, lty=1, col=2)
lines(stddev_energy, lwd=2, lty=1, col=4)

lines(mean_energy-stddev_energy, lwd=2, lty=1, col=6)
lines(mean_energy+stddev_energy, lwd=2, lty=1, col=6)

lines(mean_energy-2*stddev_energy, lwd=2, lty=1, col=8)
lines(mean_energy+2*stddev_energy, lwd=2, lty=1, col=8)

# plot cluster 2
plot(cluster2_fd, main = "Cluster 2", xlab = "Month", ylab = "Energy Consumption per Person (kWh)",
  ylim = c(0, 800), lty = 1, col = "gray80", lwd = 1)
```

```
## [1] "done"
```

```r
# elementary pointwise mean and standard deviation
mean_energy = mean.fd(cluster2_fd)
stddev_energy = std.fd(cluster2_fd)

lines(mean_energy, lwd=2, lty=1, col=2)
lines(stddev_energy, lwd=2, lty=1, col=4)

lines(mean_energy-stddev_energy, lwd=2, lty=1, col=6)
lines(mean_energy+stddev_energy, lwd=2, lty=1, col=6)

lines(mean_energy-2*stddev_energy, lwd=2, lty=1, col=8)
lines(mean_energy+2*stddev_energy, lwd=2, lty=1, col=8)

par(mfrow=c(1,1), mar=c(0,0,0,0), new=TRUE)
plot(0, 0, type="n", axes=FALSE, xlab="", ylab="", xlim=c(0,1), ylim=c(0,1))

legend("bottom",
       legend = c("Mean", "Std Dev", "Mean+-1SD", "Mean+-2SD"),
       col = c(2, 4, 6, 8),
       lty = 1,
       lwd = 2,
       horiz = TRUE,
       cex = 0.8,
       bty = "n")
```



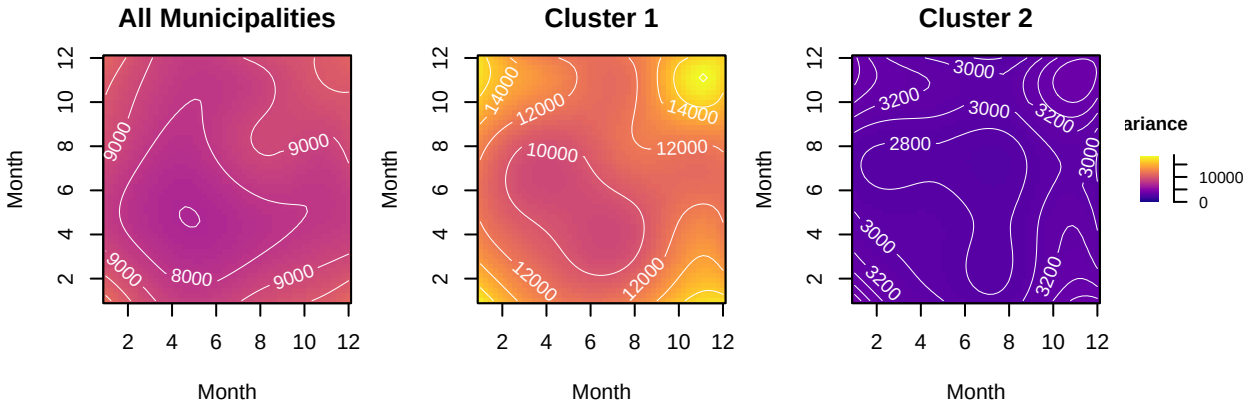
Judging from the first plot, the characteristic energy consumption in Lithuania peaks in Winter months and drops in Summer, forming in a U-shape mean curve. The blue standard deviation curve shows that variability is almost uniform during the whole year and there are no months with unusually high or low variability. The magenta curves (Mean +- 1 SD) capture the majority of municipalities with the most typical consumption patterns, while the dark gray curves (Mean +- 2 SD) reveal two most extreme outliers not falling into this range.

Comparison of plots showing centrality measures of Clusters 1 and 2 separately reveals that Cluster 1 has a higher baseline consumption and higher variability than Cluster 2. Outlier 1 also is also more complex seasonally, displaying more significant rise in consumption during Winter.

The Bivariate Covariance Function v(s, t) The bivariate covariance function analysis reveals how electricity consumption patterns at different time points relate to each other within and across months. It measures the covariance between consumption at time point s and time point t, creating a covariance surface. The diagonal of this surface represents variance (how much consumption varies within each time point), while the off-diagonal elements show how consumption in one time point relates to consumption in another time point. Like with the centrality measures, the covariance function analysis was performed for the entire dataset, and for Clusters 1 and 2 separately.

```
monthtime <- seq(min(time_range), max(time_range), length.out = 50)

# all municipalities
energy_var.bifd_all <- var.fd(smooth_fd)
energy_var_mat_all <- eval.bifd(monthtime, monthtime, energy_var.bifd_all)

# cluster 1
energy_var.bifd_c1 <- var.fd(cluster1_fd)
energy_var_mat_c1 <- eval.bifd(monthtime, monthtime, energy_var.bifd_c1)

# cluster 2
energy_var.bifd_c2 <- var.fd(cluster2_fd)
energy_var_mat_c2 <- eval.bifd(monthtime, monthtime, energy_var.bifd_c2)

library(viridis)

layout(matrix(c(1,2,3,4), nrow=1, ncol=4), widths=c(3,3,3,1))

par(mar=c(4,4,3,1), pty="s")

my_colors <- viridis(100, option="plasma")

max_val <- max(c(energy_var_mat_all, energy_var_mat_c1, energy_var_mat_c2))
max_val <- ceiling(max_val/100)*100

# All Municipalities
image(monthtime, monthtime, energy_var_mat_all,
      zlim=c(0, max_val),
      col=my_colors,
      xlab="Month", ylab="Month",
      main="All Municipalities")
box()
contour(monthtime, monthtime, energy_var_mat_all, add=TRUE, col="white", lwd=0.5, nlevels=5)

# Cluster 1
```

```r
image(monthtime, monthtime, energy_var_mat_c1,
      zlim=c(0, max_val),
      col=my_colors,
      xlab="Month", ylab="Month",
      main="Cluster 1")
box()
contour(monthtime, monthtime, energy_var_mat_c1, add=TRUE, col="white", lwd=0.5, nlevels=5)

# Cluster 2
image(monthtime, monthtime, energy_var_mat_c2,
      zlim=c(0, max_val),
      col=my_colors,
      xlab="Month", ylab="Month",
      main="Cluster 2")
box()
contour(monthtime, monthtime, energy_var_mat_c2, add=TRUE, col="white", lwd=0.5, nlevels=5)

# Color legend
par(mar=c(4,0,3,3))
plot(0, 0, type="n", xlim=c(0,1), ylim=c(0,max_val),
     axes=FALSE, xlab="", ylab="")
title("Variance", line=1, cex.main=0.9)

# Gradient color bar
y_positions <- seq(0, max_val, length.out=100)
rect_height <- max_val/100
for(i in 1:100) {
  rect(0.3, y_positions[i], 0.7, y_positions[i] + rect_height,
       col=my_colors[i], border=NA)
}

# Scale with fewer labels
axis(4, at=pretty(c(0,max_val), n=5), las=1, cex.axis=0.8)
```



The covariance surface plots reveal that the variance is higher in Winter months (January and December) and lower in the Summer. There is a positive correlation between consumption in adjacent months. Notably, Cluster 1 exhibits significantly higher variance than Cluster 2,though Cluster 1's seasonal variance pattern is more complex due to industrial activity and tourism. A flatter surface of Cluster 2 suggests that is more predictable temporal relationships and significantly more uniform covariance values across different month combinations.

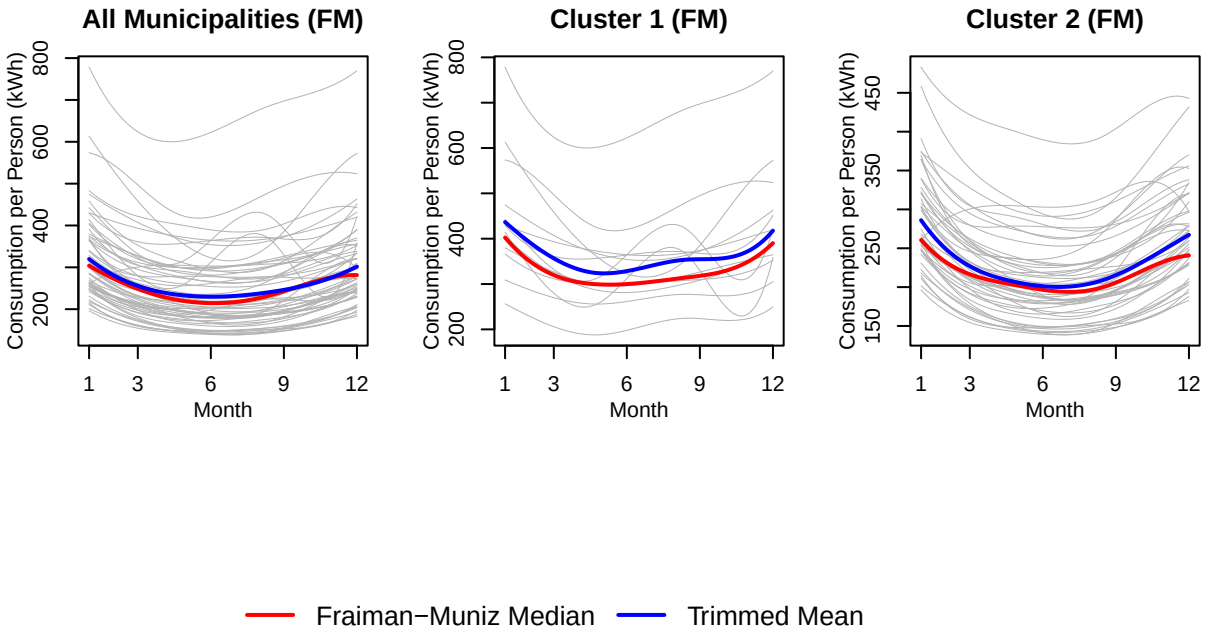
Depth To find municipalities with the most representative and the most unusual energy consumption curves, functional depth was evaluated for all municipalities and for Clusers 1 and 2 separately.

Fraiman-Muniz (FM) Depth, also known as Integrated Depth, computes the integration of an univariate depth along the x axis.

```
tt <- seq(1, 12, length.out = 100)
smooth_fdata <- fdata(smooth_fd, argvals = tt)
rownames(smooth_fdata$data) <- colnames(smooth_fd$coefs)

smooth_fdata_c1 <- fdata(cluster1_fd, argvals = tt)
rownames(smooth_fdata_c1$data) <- colnames(cluster1_fd$coefs)

smooth_fdata_c2 <- fdata(cluster2_fd, argvals = tt)
rownames(smooth_fdata_c2$data) <- colnames(cluster2_fd$coefs)

par(mfrow=c(1,3), mar=c(4,4,3,2))
```

```r
centrality_ylim <- c(170, 450)

# Fraiman-Muniz Depth
par(mfrow=c(1,3), pty="s", mar=c(4,4,3,1), oma=c(0,0,0,0), mgp=c(2,1,0))

# FM - All Municipalities
out.FM_all <- depth.FM(smooth_fdata, trim=0.1, draw=FALSE)
deepest_index_all <- which.max(out.FM_all$dep)
deepest_name_all <- rownames(smooth_fdata$data)[deepest_index_all]

plot(smooth_fdata$argvals, smooth_fdata$data[1,], type="n",
     ylim=range(smooth_fdata$data),
     main="All Municipalities (FM)", xlab="Month",
     ylab="Consumption per Person (kWh)",
     xaxt="n")
axis(1, at=c(1,3,6,9,12))

for(i in 1:nrow(smooth_fdata$data)) {
  lines(smooth_fdata$argvals, smooth_fdata$data[i,], col="gray70", lwd=0.5)
}

lines(smooth_fdata$argvals, smooth_fdata$data[deepest_index_all,], col="red", lwd=2)
lines(func.trim.FM(smooth_fdata, trim=0.1), col="blue", lwd=2)

# FM - Cluster 1
out.FM_c1 <- depth.FM(smooth_fdata_c1, trim=0.1, draw=FALSE)
deepest_index_c1 <- which.max(out.FM_c1$dep)
deepest_name_c1 <- rownames(smooth_fdata_c1$data)[deepest_index_c1]

plot(smooth_fdata_c1$argvals, smooth_fdata_c1$data[1,], type="n",
     ylim=range(smooth_fdata_c1$data),
     main="Cluster 1 (FM)", xlab="Month",
     ylab="Consumption per Person (kWh)",
     xaxt="n")
axis(1, at=c(1,3,6,9,12))

for(i in 1:nrow(smooth_fdata_c1$data)) {
  lines(smooth_fdata_c1$argvals, smooth_fdata_c1$data[i,], col="gray70", lwd=0.5)
}

lines(smooth_fdata_c1$argvals, smooth_fdata_c1$data[deepest_index_c1,], col="red", lwd=2)
lines(func.trim.FM(smooth_fdata_c1, trim=0.1), col="blue", lwd=2)

# FM - Cluster 2
out.FM_c2 <- depth.FM(smooth_fdata_c2, trim=0.1, draw=FALSE)
deepest_index_c2 <- which.max(out.FM_c2$dep)
deepest_name_c2 <- rownames(smooth_fdata_c2$data)[deepest_index_c2]

plot(smooth_fdata_c2$argvals, smooth_fdata_c2$data[1,], type="n",
     ylim=range(smooth_fdata_c2$data),
     main="Cluster 2 (FM)", xlab="Month",
     ylab="Consumption per Person (kWh)",
     xaxt="n")
```

```r
axis(1, at=c(1,3,6,9,12))

for(i in 1:nrow(smooth_fdata_c2$data)) {
  lines(smooth_fdata_c2$argvals, smooth_fdata_c2$data[i,], col="gray70", lwd=0.5)
}

lines(smooth_fdata_c2$argvals, smooth_fdata_c2$data[deepest_index_c2,], col="red", lwd=2)
lines(func.trim.FM(smooth_fdata_c2, trim=0.1), col="blue", lwd=2)

par(mfrow=c(1,1), mar=c(0,0,0,0), new=TRUE)
plot(0, 0, type="n", axes=FALSE, xlab="", ylab="", xlim=c(0,1), ylim=c(0,1))
legend("bottom", legend=c("Fraiman-Muniz Median", "Trimmed Mean"),
       col=c("red", "blue"), lwd=2, cex=0.8, bty="n", horiz=TRUE)
```




```r
cat("FM Deepest (All):", deepest_name_all, "\n")
```

```
## FM Deepest (All): Alytaus m. sav.
```

```r
cat("FM Deepest (Cluster 1):", deepest_name_c1, "\n")
```

```
## FM Deepest (Cluster 1): Druskininkų sav.
```

```r
cat("FM Deepest (Cluster 2):", deepest_name_c2, "\n")
```

```
## FM Deepest (Cluster 2): Šiaulių m. sav.
```

Outliers

```r
# grid of 100 evenly spaced time points over the time range
tt <- seq(min(time_range), max(time_range), length.out = 100)
el <- eval.fd(tt, smooth_fd)

#needs transpose
bd <- band_depth(dt = t(el))
names(bd) <- colnames(el)

mbd <- modified_band_depth(t(el))

fbplot_obj <- functional_boxplot(t(el), depth_method = "bd")
# fbplot_obj$outliers
outlier_indices <- fbplot_obj$outliers
municipality_names <- colnames(el)[outlier_indices]
print(municipality_names)
```

All municipalities

```
## [1] "Kazlų Rūdos sav."
```

```r
fbplot_obj <- functional_boxplot(t(el), depth_method = "mbd")
# fbplot_obj$outliers
outlier_indices <- fbplot_obj$outliers
municipality_names <- colnames(el)[outlier_indices]
print(municipality_names)
```

```
## [1] "Akmenės r. sav."  "Elektrėnų sav."   "Kazlų Rūdos sav."
```

Band Depth (BD) measures how "central" each curve is among all curves. The higher the band depth, the more typical the curve is. Modified band depth (MBD) tends to be more stable and less sensitive to crossing curves. `functional_boxplot()` uses BD and MBD to find curves that are "outliers" (very low depth).

```r
m <- muod(t(el), cut_method = c("boxplot"))
# m$outliers
colnames(el)[m$outliers$shape]
colnames(el)[m$outliers$amplitude]
colnames(el)[m$outliers$magnitude]
```

Massive Unsupervised Outlier Detection (MUOD) finds outliers by computing for each functional data, a magnitude, amplitude and shape index. Outliers identified in the magnitude indices are flagged as magnitude outliers. The same holds true for the amplitude and shape indices. Thus, the outliers are not only identified but also classified.

Shape putliers: Pakruojis, Neringa, Palanga, Kazlų Rūda, Joniškis, Birštonas, Kėdainiai, Pasvalys, Akmenė.

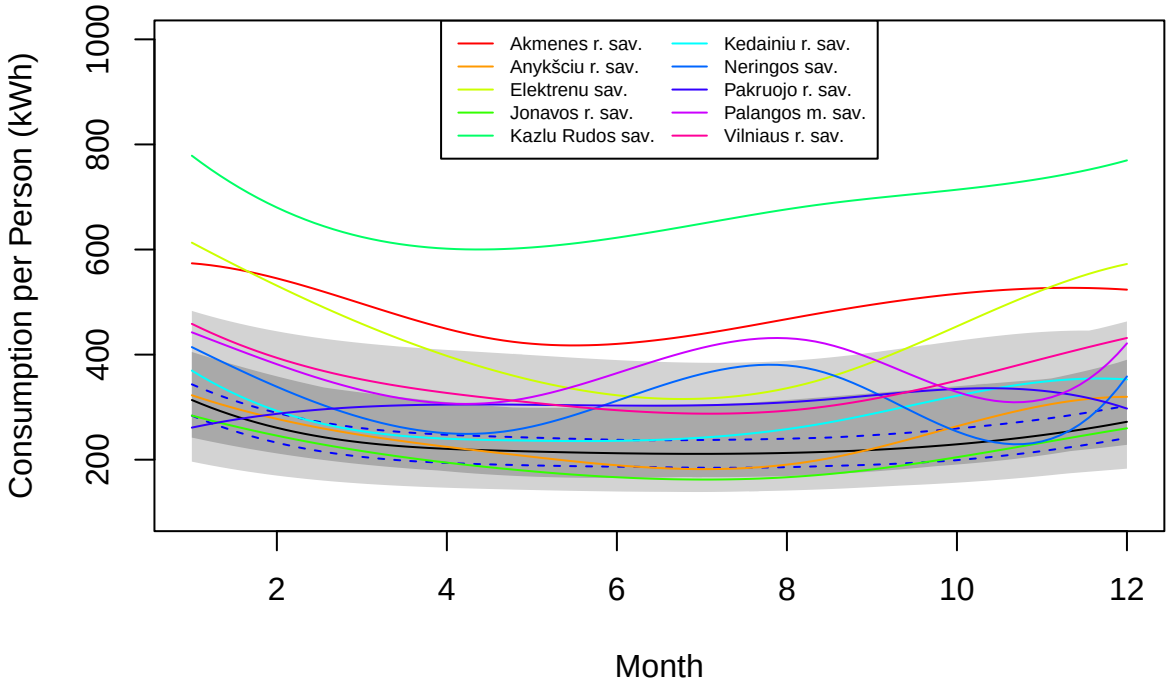
Amplitude outliers: Elektrėnai, Pakruojis.

Magnitude outliers: Elektrėnai, Pakruojis, Palanga, Kazlų Rūda, Neringa.

Functional Boxplots

```
data_mat <- t(smooth_fdata$data)  # now rows = time, cols = functions
time_grid <- smooth_fdata$argvals

curve_labels <- rownames(smooth_fdata$data)

colnames(data_mat) <- curve_labels
fts_object <- fts(y = data_mat, x = time_grid)

labels <- colnames(fts_object$y)

# functional boxplot
fboxplot(data = fts_object, plot.type = "functional",
         type = "bag", projmethod = "PCAproj",
         xlab = "Month", ylab = "Consumption per Person (kWh)",
         ylim = c(100, 1000),
         legendpos = "top",
         cex = 0.6)
```

All municipalities



The dark gray band shows the median 50% region.

Regression

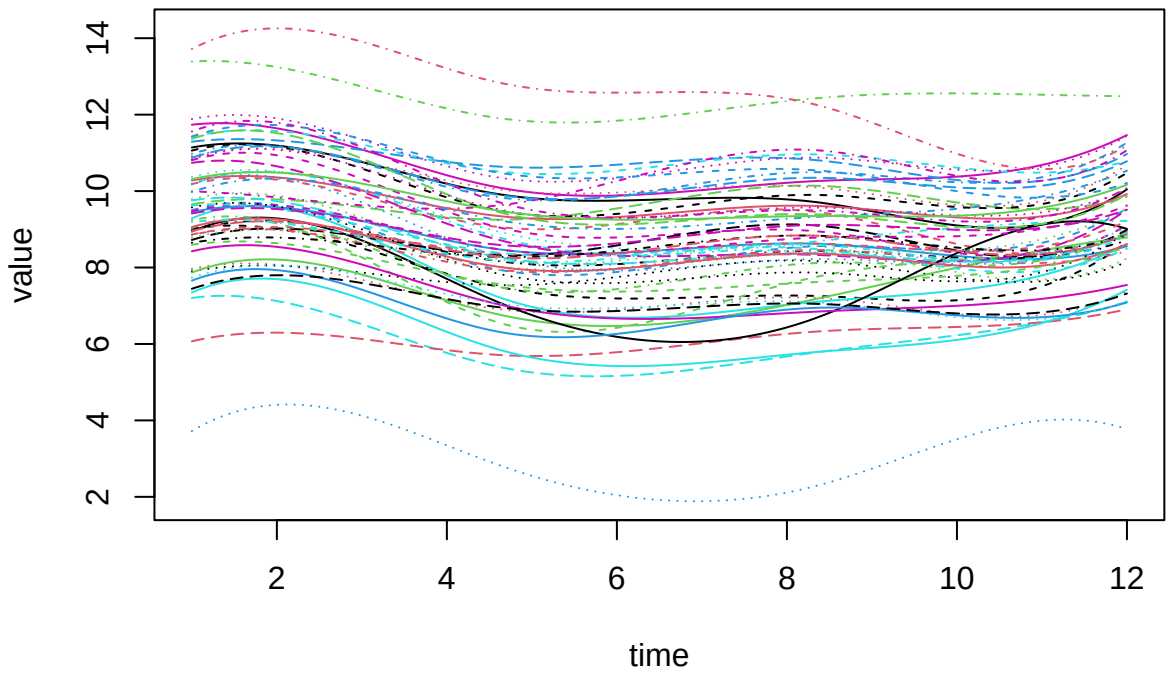
To understand how economic and industrial factors influence consumption behaviors throughout the year, we performed both function-on-scalar and function-on-function regression analysis. The dependant variable in our analysis was monthly per-capita energy consumption, while the predictors were unemployment rates, commercial and residential customer counts per person, and average wage in each municipality. The predictors were standardized to have mean 0 and standard deviation 1 to better capture the influence of predictors that have vastly different scales. Without standartization, wages would dominate the regression simply because of the scale, not beacuse of importance.

```
# convert to wide format
smoothed_wide <- smoothed_df %>%
  pivot_wider(id_cols = Savivaldybe,
              names_from = Month,
              values_from = Consumption)

# create a matrix
consumption_matrix <- as.matrix(smoothed_wide[, -1])
rownames(consumption_matrix) <- municipalities
```

```r
wages_ordered <- population %>%
  arrange(factor(Savivaldybe, levels = municipalities)) %>%
  pull(atlyginimas)

df <- data.frame(
  Savivaldybe = municipalities,
  ID = 1:length(municipalities),
  atlyginimas = as.numeric(wages_ordered)
)

df$consumption <- consumption_matrix

# convert to a refund object
consumption_df <- as_refundObj(df$consumption)

eval_points <- seq(min(months), max(months), length.out = 100)

consumption_basis <- create.bspline.basis(rangeval = range(eval_points),
  nbasis = 6, norder = 4) # cubic splines

consumption_fd <- Data2fd(eval_points, t(as.matrix(df$consumption)), consumption_basis)
```

```r
unemployment_2024 <- read.csv("data/nedarbas.csv")

matched_unemployment <- unemployment_2024 %>%
  filter(savivaldybe %in% df$Savivaldybe)

unemployment_matrix <- matched_unemployment %>%
  dplyr::select(savivaldybe, menesis, nedarbas) %>%
  dplyr::arrange(menesis, savivaldybe) %>%
  tidyr::pivot_wider(names_from = savivaldybe, values_from = nedarbas) %>%
  dplyr::arrange(menesis) %>%
  dplyr::select(-menesis) %>%
  as.matrix()

fd_obj_all_mun_unempl <- smooth.basis(argvals = months, y = unemployment_matrix,
  fdParobj = fdPar(basis, 2))$fd
plot(fd_obj_all_mun_unempl)
```

Preparing functional data (unemployment)



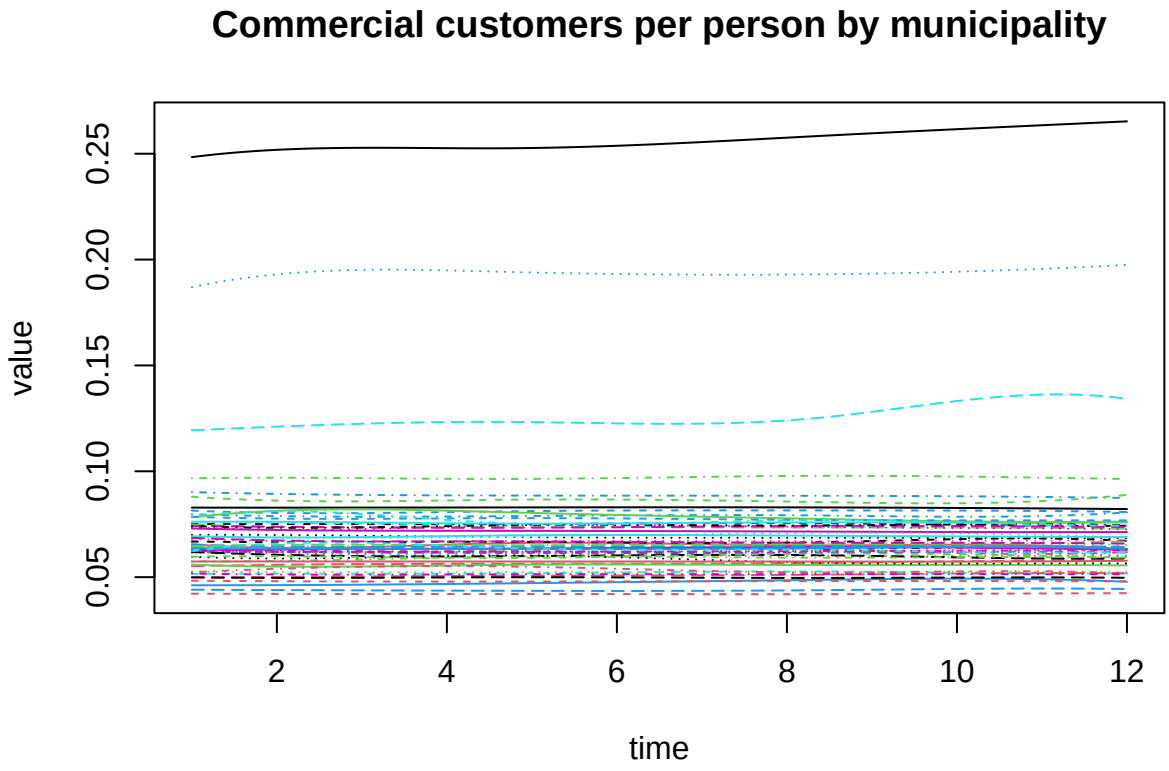

```
## [1] "done"
```

```
unemployment_smoothed <- eval.fd(eval_points, fd_obj_all_mun_unempl)

# transpose
unemployment_smoothed_t <- t(unemployment_smoothed)

df$unemployment <- unemployment_smoothed_t
colnames(df$unemployment) <- colnames(df$consumption)
```

```
customer_types <- read_parquet("data/fda_sutart_kiekis.parquet")

customer_types$date <- as.Date(substr(customer_types$Suvartojimo_laikotarpis, 1, 10))

customer_types <- customer_types %>%
  filter(format(date, "%Y") == "2024")

customer_types <- customer_types %>%
  filter(Savivaldybe %in% df$Savivaldybe)

customer_types_by_population <- customer_types %>% left_join(population, by = "Savivaldybe")
```

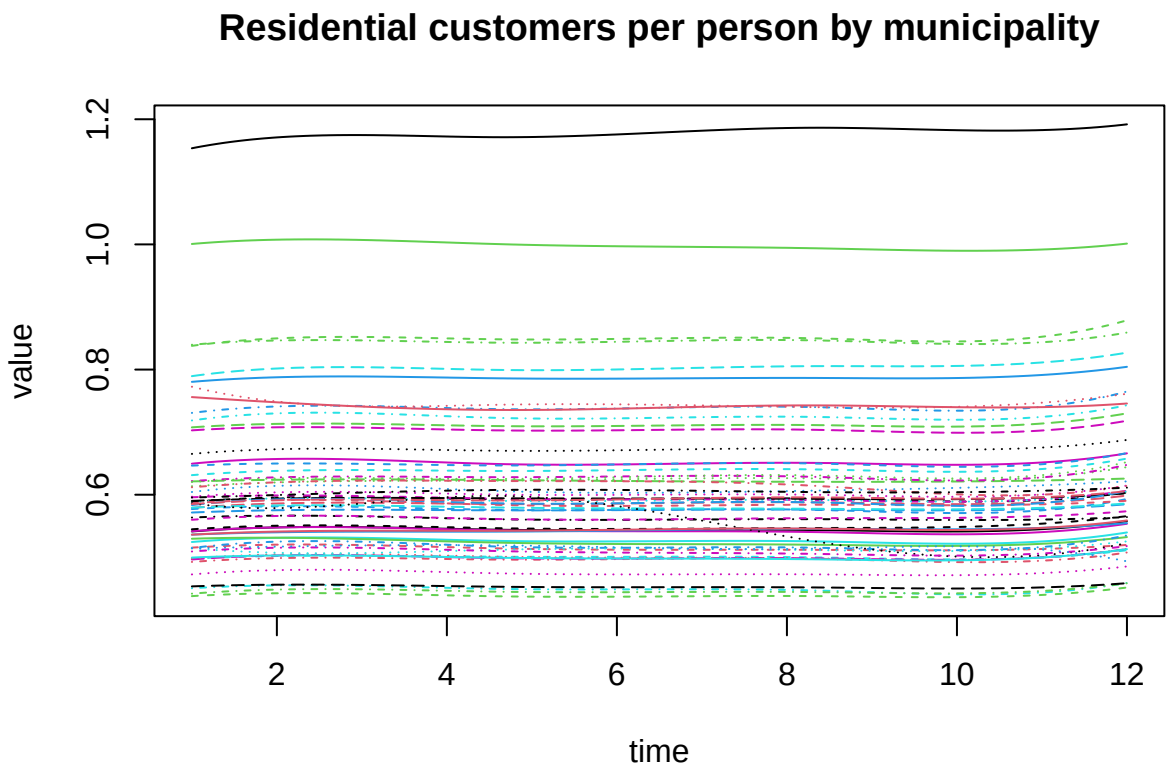
```r
customer_types_by_population <- customer_types_by_population %>%
  mutate(customers_per_person = kiekis / populiacija)

commercial_matrix <- customer_types_by_population %>%
  filter(Sutarties_tipas == "Komercinė") %>%
  dplyr::select(date, Savivaldybe, customers_per_person) %>%
  dplyr::arrange(date, Savivaldybe) %>%
  tidyr::pivot_wider(names_from = Savivaldybe, values_from = customers_per_person) %>%
  dplyr::arrange(date) %>%
  dplyr::select(-date) %>%
  as.matrix()

residential_matrix <- customer_types_by_population %>%
  filter(Sutarties_tipas == "Buitinė") %>%
  dplyr::select(date, Savivaldybe, customers_per_person) %>%
  dplyr::arrange(date, Savivaldybe) %>%
  tidyr::pivot_wider(names_from = Savivaldybe, values_from = customers_per_person) %>%
  dplyr::arrange(date) %>%
  dplyr::select(-date) %>%
  as.matrix()

unique_dates <- unique(customer_types_by_population$date)
unique_dates <- sort(unique_dates)
months <- seq(from = 1, to = length(unique_dates), by = 1)

# create fd objects
fd_obj_commercial <- smooth.basis(argvals = months, y = commercial_matrix,
  fdParobj = fdPar(basis, 2))$fd
fd_obj_residential <- smooth.basis(argvals = months, y = residential_matrix,
  fdParobj = fdPar(basis, 2))$fd

plot(fd_obj_commercial, main = "Commercial customers per person by municipality")
```

Preparing functional data (commercial/residential customer count per person)




```
## [1] "done"
```

```
plot(fd_obj_residential, main = "Residential customers per person by municipality")
```




```
## [1] "done"
```

```
commercial_smoothed <- eval.fd(eval_points, fd_obj_commercial)
residential_smoothed <- eval.fd(eval_points, fd_obj_residential)

# transpose
commercial_smoothed_t <- t(commercial_smoothed)
residential_smoothed_t <- t(residential_smoothed)

df$commercial_customers <- commercial_smoothed_t
df$residential_customers <- residential_smoothed_t

colnames(df$commercial_customers) <- colnames(df$consumption)
colnames(df$residential_customers) <- colnames(df$consumption)
```

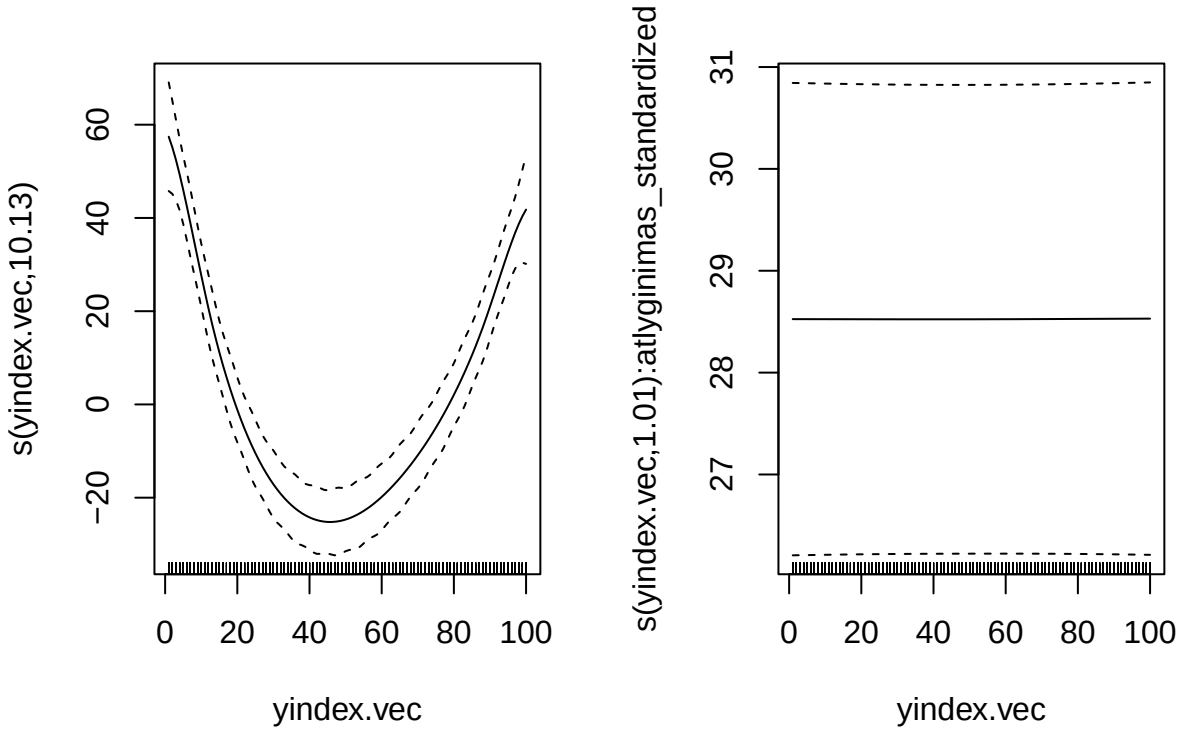
Functional-on-scalar regression with mean wages, unemployment and customer count values
For function-on-scalar regression analysis we performed penalized flexible functional regression (pffr) to investigate how municipal characteristics influence electricity consumption patterns throughout the year. This approach estimates time-varying coefficient functions to reveal how the effects of predictors change seasonally.

In this regression, unemployment rates, commercial and residential customers counts per capita for each municipality were first averaged across months, then standardized to allow a more direct comparison of predictor importance.

```r
# standardizing wages
df$atlyginimas_standardized <- (df$atlyginimas - mean(df$atlyginimas)) / sd(df$atlyginimas)

unempl_ordered <- population %>%
  arrange(factor(Savivaldybe, levels = municipalities)) %>%
  pull(vid_nedarbas)

df$vid_nedarbas <- as.numeric(unempl_ordered)

vid_nedarbas_std <- (df$vid_nedarbas - mean(df$vid_nedarbas)) / sd(df$vid_nedarbas)

df$vid_nedarbas_std <- as.numeric(vid_nedarbas_std)

commercial_vals <- eval.fd(months, fd_obj_commercial)
residential_vals <- eval.fd(months, fd_obj_residential)

commercial_avg <- colMeans(commercial_vals, na.rm = TRUE)
residential_avg <- colMeans(residential_vals, na.rm = TRUE)

commercial_avg_std <- (commercial_avg - mean(commercial_avg)) / sd(commercial_avg)
residential_avg_std <- (residential_avg - mean(residential_avg)) / sd(residential_avg)

df$commercial_avg <- commercial_avg
df$residential_avg <- residential_avg
df$commercial_avg_std <- commercial_avg_std
df$residential_avg_std <- residential_avg_std

# penalized flexible functional regression
fosr.fit <- pffr(consumption ~ atlyginimas_standardized, data = df)
summary(fosr.fit)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## consumption ~ atlyginimas_standardized
##
## Constant coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   263.21       1.14   230.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Smooth terms & functional coefficients:
##                                    edf Ref.df      F p-value
## Intercept(yindex)               10.130 19.000  22.76  <2e-16 ***
## atlyginimas_standardized(yindex)  1.006  1.012 607.77  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.149   Deviance explained =   15%
## -REML score =  35410  Scale est. = 7802.5    n = 6000 (60 x 100)
```

```r
plot(fosr.fit, pages=1, scale=0) # coefficients of regressors over time
```

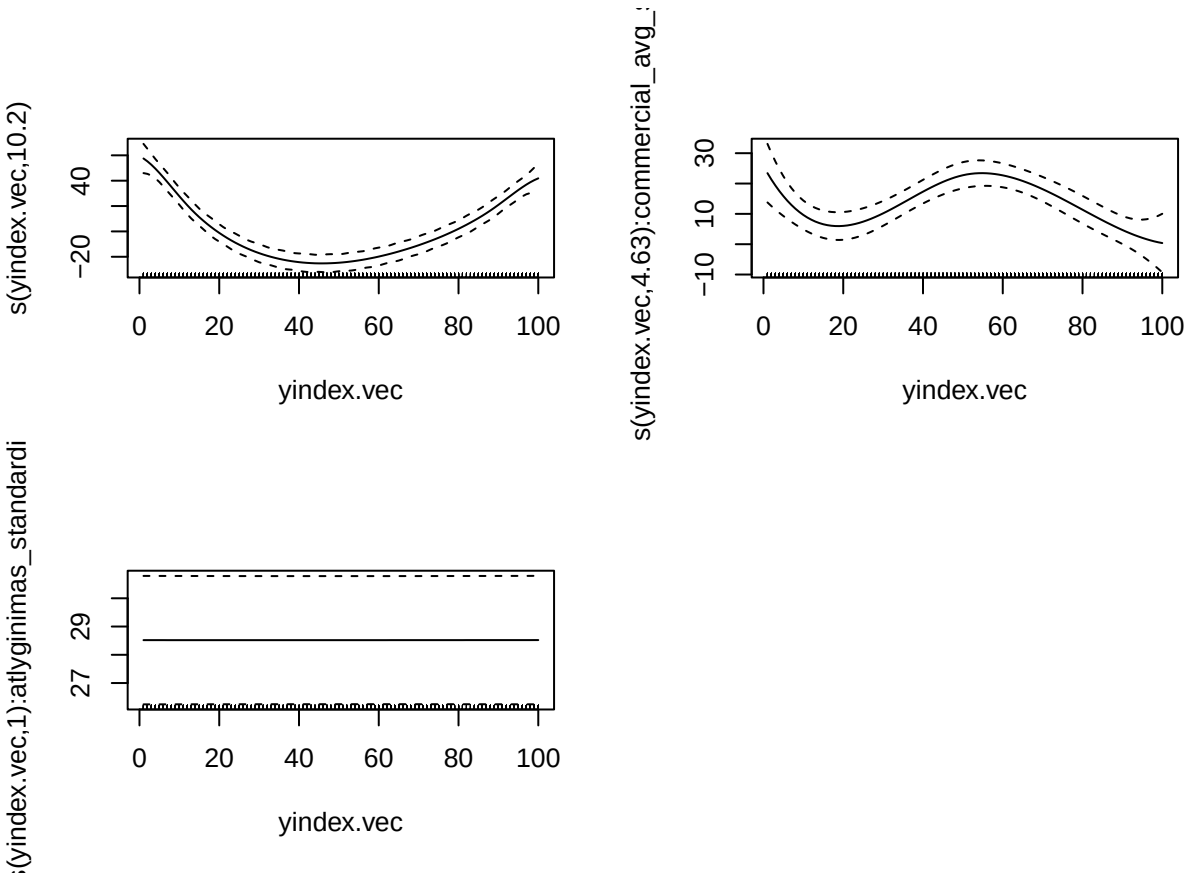



```r
fosr.fit2 <- pffr(consumption ~ commercial_avg_std + atlyginimas_standardized,
  data = df)
summary(fosr.fit2)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## consumption ~ commercial_avg_std + atlyginimas_standardized
##
## Constant coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  263.209      1.124   234.2   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Smooth terms & functional coefficients:
##                                     edf Ref.df      F p-value
## Intercept(yindex)                10.196 19.000  23.45  <2e-16 ***
## commercial_avg_std(yindex)        4.633  4.917  36.98  <2e-16 ***
## atlyginimas_standardized(yindex)  1.002  1.005 630.34  <2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.173   Deviance explained = 17.5%
## -REML score =  35329  Scale est. = 7578      n = 6000 (60 x 100)
```

```
plot(fosr.fit2, pages=1, scale=0)
```

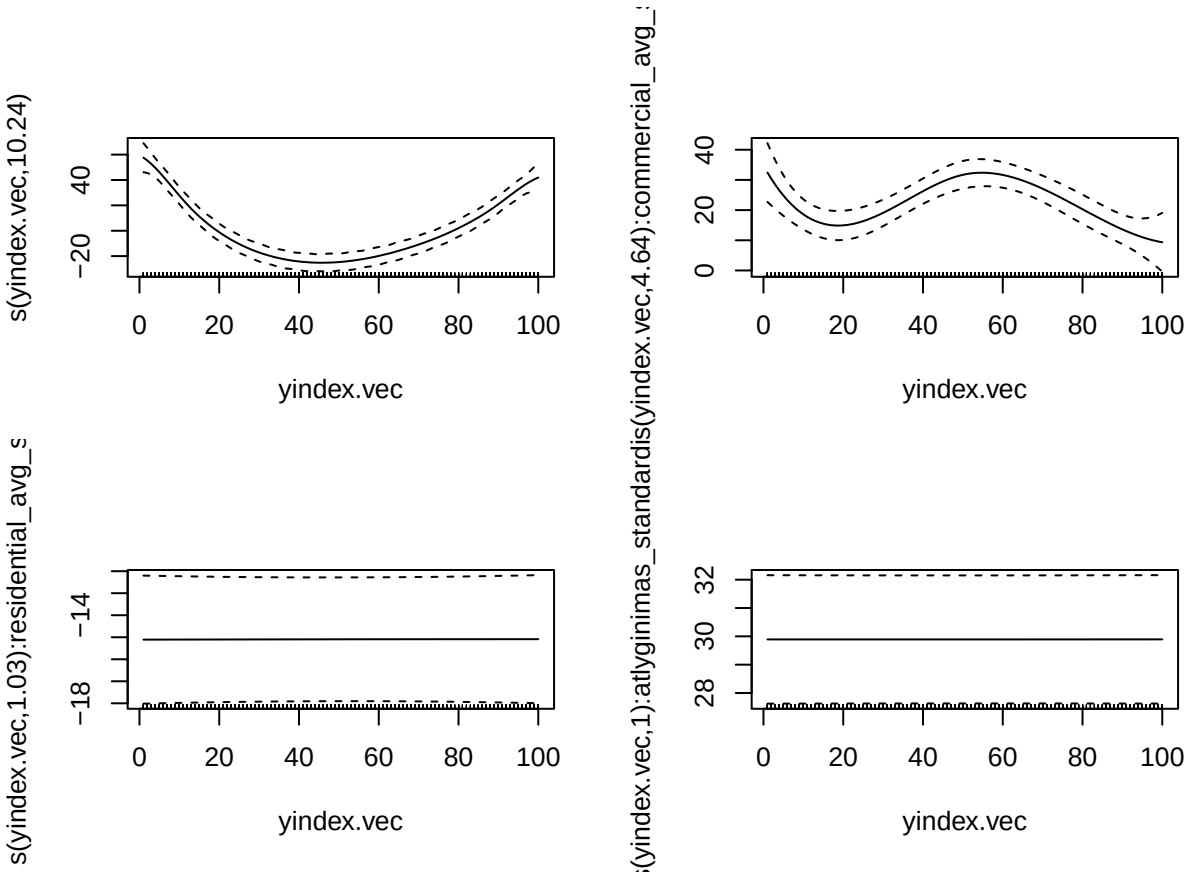



```
fosr.fit3 <- pffr(consumption ~ commercial_avg_std + residential_avg_std + atlyginimas_standardized,  da
summary(fosr.fit3)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## consumption ~ commercial_avg_std + residential_avg_std + atlyginimas_standardized
##
## Constant coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  263.209      1.113   236.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Smooth terms & functional coefficients:
##                                     edf Ref.df      F p-value
## Intercept(yindex)                10.239 19.000  23.91  <2e-16 ***
## commercial_avg_std(yindex)        4.641  4.920  60.70  <2e-16 ***
## residential_avg_std(yindex)       1.028  1.056 109.82  <2e-16 ***
## atlyginimas_standardized(yindex)  1.003  1.005 696.44  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.189   Deviance explained = 19.1%
## -REML score =  35269  Scale est. = 7435.2    n = 6000 (60 x 100)
```

```
plot(fosr.fit3, pages=1, scale=0)
```

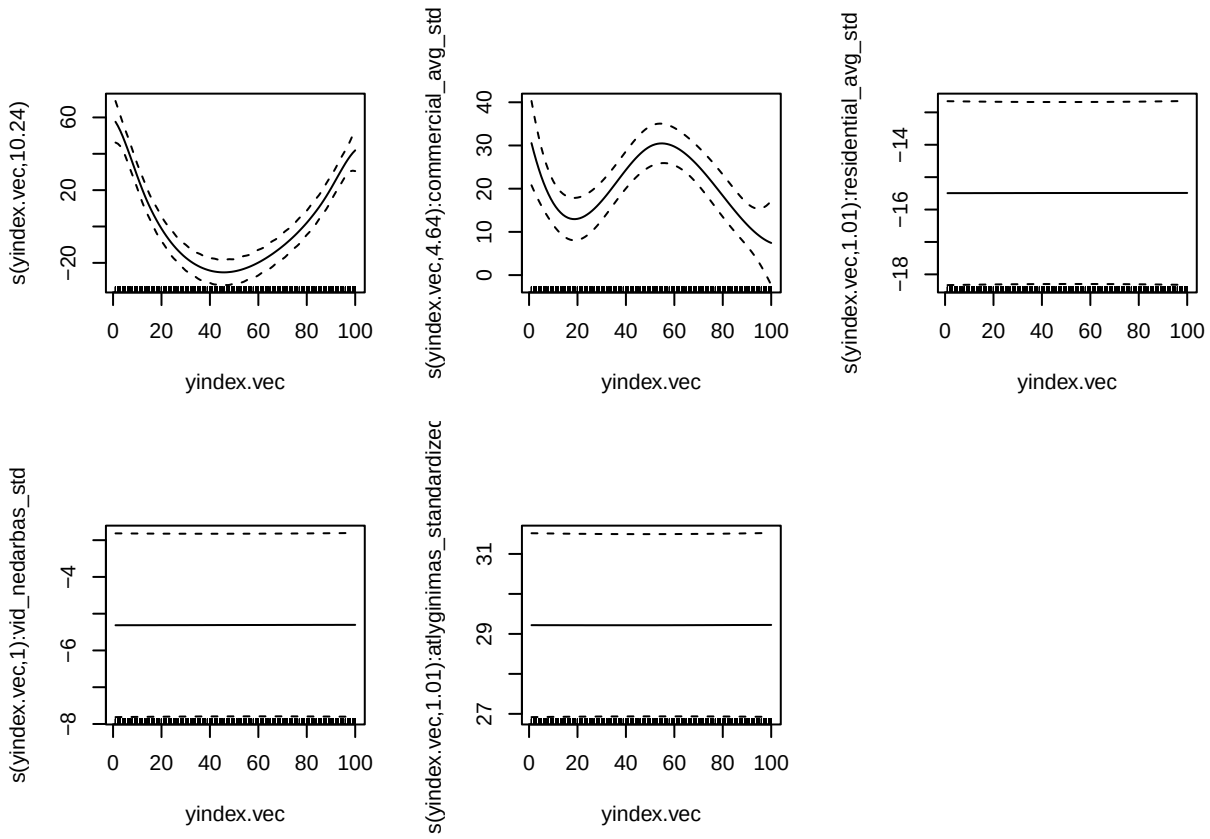



```
fosr.fit4 <- pffr(consumption ~ commercial_avg_std + residential_avg_std + vid_nedarbas_std +
    atlyginimas_standardized,  data = df)
summary(fosr.fit4)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## consumption ~ commercial_avg_std + residential_avg_std + vid_nedarbas_std +
```

```
##     atlyginimas_standardized
##
## Constant coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  263.209      1.112   236.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Smooth terms & functional coefficients:
##                                    edf Ref.df      F  p-value
## Intercept(yindex)                10.245 19.000  23.98  < 2e-16 ***
## commercial_avg_std(yindex)        4.642  4.920  48.35  < 2e-16 ***
## residential_avg_std(yindex)       1.009  1.017 119.83  < 2e-16 ***
## vid_nedarbas_std(yindex)          1.005  1.010  18.13 2.03e-05 ***
## atlyginimas_standardized(yindex)  1.007  1.015 648.38  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.191   Deviance explained = 19.3%
## -REML score =  35258  Scale est. = 7413.8    n = 6000 (60 x 100)
```

```
plot(fosr.fit4, pages=1, scale=0)
```



Four function-on-scalar regression models were compared:

- Model 1:

$$Y_i(t) = \beta_0(t) + \beta_1(t) \cdot \text{Wages}_i + \varepsilon_i(t)$$

- Model 2:

$$Y_i(t) = \beta_0(t) + \beta_1(t) \cdot \text{Wages}_i + \beta_2(t) \cdot \text{Commercial}_i + \varepsilon_i(t)$$

- Model 3:

$$Y_i(t) = \beta_0(t) + \beta_1(t) \cdot \text{Wages}_i + \beta_2(t) \cdot \text{Commercial}_i + \beta_3(t) \cdot \text{Residential}_i + \varepsilon_i(t)$$

- Model 4:

$$Y_i(t) = \beta_0(t) + \beta_1(t) \cdot \text{Wages}_i + \beta_2(t) \cdot \text{Commercial}_i + \beta_3(t) \cdot \text{Residential}_i + \beta_4(t) \cdot \text{Unemployment}_i + \varepsilon_i(t)$$

- $Y_i(t)$ = Electricity consumption for municipality i at time t
- $\beta_0(t)$ = Intercept function (baseline seasonal consumption pattern)

- $\beta_j(t)$ = Time-varying coefficient functions for predictor j
- Wages_i = Standardized average wages in municipality i
- Commercial_i = Standardized commercial customers per person in municipality i
- Residential_i = Standardized residential customers per person in municipality i
- Unemployment_i = Standardized unemployment rate in municipality i
- $\varepsilon_i(t)$ = Error function for municipality i at time t

Based on low p-values, all regressors are statistically significant.

Wages and count of commercial customers have a positive coefficient, indicating that higher values of these predictors are related to higher consumption levels. Meanwhile, unemployment and count of residential customers have negative coefficients, suggesting an opposite effect.

The intercept captures baseline seasonal consumption patterns. Out of all predictors, only the count of commercial customers coefficients show distinct seasonal patterns with peaks at the beginning of and the middle of the year, indicating that this regressor is more influential during those periods. This could be related to seasonal business cycle. Other independent variables maintain constant influence throughout the year.

```r
# get predictions and transpose to match observed format
get_pffr_predictions <- function(model) {
  return(t(predict(model)))
}

pred_wages_only <- get_pffr_predictions(fosr.fit)
pred_wages_commercial <- get_pffr_predictions(fosr.fit2)
pred_comm_res_wages <- get_pffr_predictions(fosr.fit3)
pred_all_vars <- get_pffr_predictions(fosr.fit4)

observed <- t(as.matrix(df$consumption))
```

```r
calculate_r2 <- function(observed, predicted) {
  sse <- sum((observed - predicted)^2)
  sst <- sum((observed - mean(observed))^2)
  return(1 - sse/sst)
}

# performance metrics
metrics <- data.frame(
  Model = c("Wages_Only", "Wages_Commercial", "Wages_Comm_Res", "All_Variables"),
  AIC = c(AIC(fosr.fit), AIC(fosr.fit2), AIC(fosr.fit3), AIC(fosr.fit4)),
  BIC = c(BIC(fosr.fit), BIC(fosr.fit2), BIC(fosr.fit3), BIC(fosr.fit4)),
  R2 = c(
    calculate_r2(as.vector(observed), as.vector(pred_wages_only)),
    calculate_r2(as.vector(observed), as.vector(pred_wages_commercial)),
    calculate_r2(as.vector(observed), as.vector(pred_comm_res_wages)),
    calculate_r2(as.vector(observed), as.vector(pred_all_vars))
  )
)

print(metrics)
```

```
##              Model      AIC      BIC        R2
## 1       Wages_Only 70815.28 70905.68 0.1501372
## 2 Wages_Commercial 70645.63 70770.24 0.1752350
## 3   Wages_Comm_Res 70533.47 70668.28 0.1909206
## 4    All_Variables 70517.10 70658.48 0.1933884
```

Of the four models, Model 4 (which included all predictors) has the lowest Akaike Information Criterion (AIC) and (Bayesian Information Criterion) BIC, and highest R^2. However, even this model has a low R^2 value of 0.19.

To understand how the relative influence of different predictors varies throughout the year, we plotted each predictor's proportional contribution to the total absolute effect at each time point.

```r
# Importance of each predictor from fosr.fit4 model
fosr_coefs <- coef(fosr.fit4)
```

```
## using seWithMean for  s(yindex.vec) .
```

```r
commercial_coef_fosr <- abs(fosr_coefs$smterms$`commercial_avg_std(yindex)`$value)
residential_coef_fosr <- abs(fosr_coefs$smterms$`residential_avg_std(yindex)`$value)
unemployment_coef_fosr <- abs(fosr_coefs$smterms$`vid_nedarbas_std(yindex)`$value)
wages_coef_fosr <- abs(fosr_coefs$smterms$`atlyginimas_standardized(yindex)`$value)

total_effect_fosr <- commercial_coef_fosr + residential_coef_fosr +
                     unemployment_coef_fosr + wages_coef_fosr

commercial_importance_fosr <- commercial_coef_fosr / total_effect_fosr * 100
residential_importance_fosr <- residential_coef_fosr / total_effect_fosr * 100
unemployment_importance_fosr <- unemployment_coef_fosr / total_effect_fosr * 100
wages_importance_fosr <- wages_coef_fosr / total_effect_fosr * 100

cat(sprintf("Mean importance from function-on-scalar model:\n"))
```

```
## Mean importance from function-on-scalar model:
```

```
cat(sprintf("Mean importance of the count of commercial customers per person: %.2f%%\n",
           mean(commercial_importance_fosr)))
```

```
## Mean importance of the count of commercial customers per person: 28.16%
```

```
cat(sprintf("Mean importance of the count of residential customers per person: %.2f%%\n",
           mean(residential_importance_fosr)))
```

```
## Mean importance of the count of residential customers per person: 22.25%
```

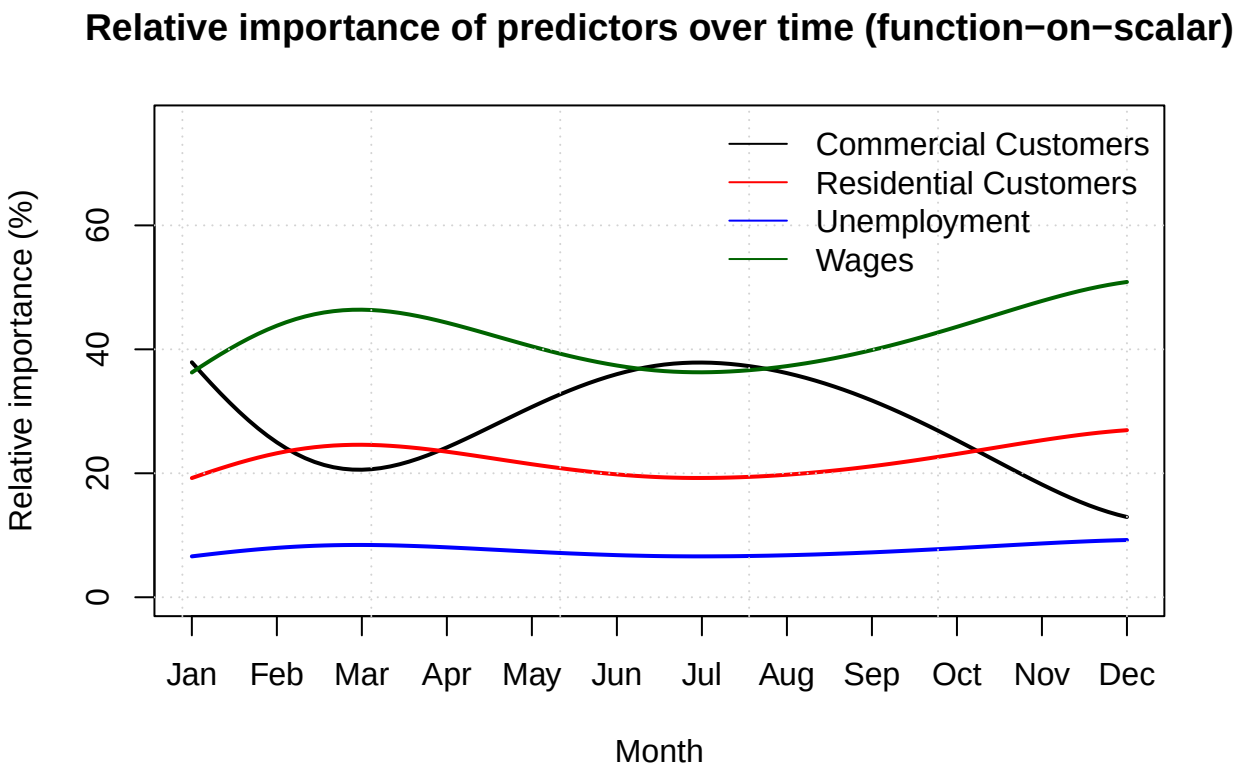
```
cat(sprintf("Mean importance of unemployment: %.2f%%\n",
           mean(unemployment_importance_fosr)))
```

```
## Mean importance of unemployment: 7.62%
```

```
cat(sprintf("Mean importance of wages: %.2f%%\n",
           mean(wages_importance_fosr)))
```

```
## Mean importance of wages: 41.96%
```

```
eval_points <- 1:length(commercial_coef_fosr)
month_positions <- seq(1, length(eval_points), length.out = 12)

plot(eval_points, commercial_importance_fosr, type="l", lwd=2, col="black",
     xlab="Month", ylab="Relative importance (%)",
     main="Relative importance of predictors over time (function-on-scalar)",
     ylim=c(0, max(commercial_importance_fosr, residential_importance_fosr,
                   unemployment_importance_fosr, wages_importance_fosr) * 1.5),
     xaxt="n")
axis(1, at=month_positions, labels=month.abb)
lines(eval_points, residential_importance_fosr, lwd=2, col="red", lty=1)
lines(eval_points, unemployment_importance_fosr, lwd=2, col="blue", lty=1)
lines(eval_points, wages_importance_fosr, lwd=2, col="darkgreen", lty=1)
grid(NULL, NULL, lty=3, col="lightgray")
legend("topright",
       legend=c("Commercial Customers", "Residential Customers", "Unemployment", "Wages"),
       lty=1,
       col=c("black", "red", "blue", "darkgreen"),
       bty="n")
```



From the plot we can see that wages has the biggest influence which rises towards the end of the year and around March. The influence of the count of commercial customer varies the most and peaks during the Summer and at the beginning of the year. The most influential predictor is the unemployment rate which is almost constant throughout the year.

Function-on-function regression model: unemployment (function), commercial/residential customer count per person (function), and wages (scalar) Next, we performed function-on-function regression, where not only the dependent variable but also the predictors were functions.

$$Y_i(t) = \beta_0(t) + \int_0^{12} \beta_1(t,s) \cdot \text{Commercial}_i(s)\, ds + \int_0^{12} \beta_2(t,s) \cdot \text{Residential}_i(s)\, ds$$

$$+ \int_0^{12} \beta_3(t,s) \cdot \text{Unemployment}_i(s)\, ds + \int_0^{12} \beta_4(t,s) \cdot \text{UnemploymentVel}_i(s)\, ds$$

$$+ \beta_5(t) \cdot \text{Wages}_i + \varepsilon_i(t)$$

Where: - $Y_i(t)$ = Electricity consumption for municipality i at time t - $\beta_0(t)$ = Intercept function (baseline seasonal consumption pattern) - $\beta_j(t,s)$ = Bivariate coefficient functions for functional predictors - $\beta_5(t)$ = Time-varying coefficient function for scalar predictor - $\text{Commercial}_i(s)$ = Commercial customers per person function for municipality i - $\text{Residential}_i(s)$ = Residential customers per person function for municipality i - $\text{Unemployment}_i(s)$ = Unemployment rate function for municipality i - $\text{UnemploymentVel}_i(s)$ = Unemployment velocity function for municipality i - Wages_i = Standardized average wages (scalar) for municipality i - $\varepsilon_i(t)$ = Error function for municipality i at time t - $s, t \in [0, 12]$ = Time points (months)

Integration captures how predictor values at time s affect response at time t.

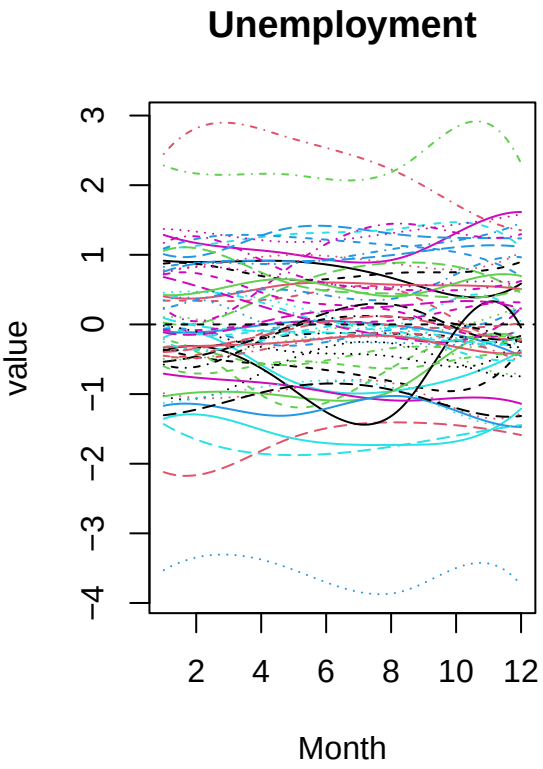
```r
eval_points <- seq(min(months), max(months), length.out = 100)

# function to standardize a functional data object - each month is
#  individually transformed to have mean 0 and standard deviation 1
standardize_fd <- function(fd_obj) {
  eval_points <- seq(min(months), max(months), length.out = 100)
  fd_values <- eval.fd(eval_points, fd_obj)

  means <- rowMeans(fd_values)
  sds <- apply(fd_values, 1, sd)

  fd_values_std <- sweep(fd_values, 1, means, "-")
  fd_values_std <- sweep(fd_values_std, 1, sds, "/")

  fd_basis <- fd_obj$basis
  fd_std <- Data2fd(argvals = eval_points, y = fd_values_std, basisobj = fd_basis)

  return(fd_std)
}

commercial_fd_std <- standardize_fd(fd_obj_commercial)
residential_fd_std <- standardize_fd(fd_obj_residential)
unemployment_fd_std <- standardize_fd(fd_obj_all_mun_unempl)
```

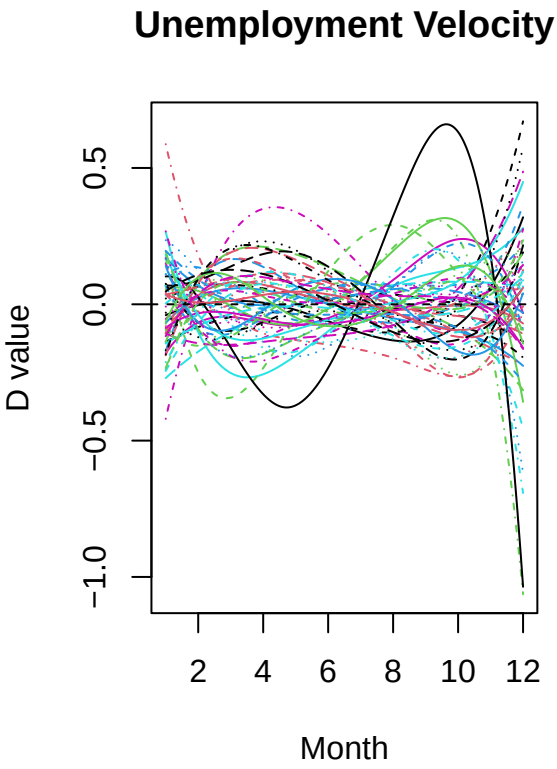
```r
# velocity for unemployment
unemployment_fd_vel <- deriv.fd(unemployment_fd_std, 1)

par(mfrow=c(1,2))
plot(unemployment_fd_std, main="Unemployment", xlab="Month")
```

Predictor standardization

```
## [1] "done"
```

```r
plot(unemployment_fd_vel, main="Unemployment Velocity", xlab="Month")
```






```
## [1] "done"
```

```
par(mfrow=c(1,1))

unemployment_vel_values <- eval.fd(eval_points, unemployment_fd_vel)

unemployment_vel_std_values <- (unemployment_vel_values - mean(unemployment_vel_values)) /
  sd(as.vector(unemployment_vel_values))
unemployment_vel_std <- Data2fd(argvals = eval_points,
                                y = unemployment_vel_std_values,
                                basisobj = unemployment_fd_vel$basis)

xfdlist <- list(
  const = rep(1, nrow(df)),
  commercial = commercial_fd_std,
  residential = residential_fd_std,
  unemployment = unemployment_fd_std,
  unemployment_vel = unemployment_vel_std,
  wages = df$atlyginimas_standardized
)
```

Additionally, after trying out several combinations of predictors, we included the first derivative (velocity) of unemployment into our model for better characterization of functional behavior.

To find the best smoothing parameters (λ and *nbasis*) for each predictor, we performed a systematic search based on Generalized Cross-Validation (GCV). Number of basis functions controls model complexity and flexibility (tested range: 8-24), while λ controls function smoothness (tested range: $10^{-5} - 10^{1}$).

```r
# find optimal parameters for a specific regressor
find_optimal_params <- function(regressor_name) {
  cat("\nFinding optimal parameters for:", regressor_name, "\n")

  nbasis_values <- seq(8, 24, 1)
  lambda_log_range <- seq(-5, 1, 1)
  n_nbasis <- length(nbasis_values)
  n_lambda <- length(lambda_log_range)

  # array to store GCV values
  smoothStats <- array(NA, dim=c(n_nbasis, n_lambda),
                       dimnames=list(nbasis_values, lambda_log_range))

  for (i in 1:n_nbasis) {
    nbasis <- nbasis_values[i]
    beta_basis <- create.bspline.basis(rangeval = range(eval_points), nbasis = nbasis)

    for (j in 1:n_lambda) {
      lambda <- 10^lambda_log_range[j]
      betafdPar <- fdPar(beta_basis, 2, lambda = lambda)

      default_basis <- create.bspline.basis(rangeval = range(eval_points), nbasis = 15)
      default_betafdPar <- fdPar(default_basis, 2, lambda = 1)

      betalist <- list(
        const = default_betafdPar,
        commercial = default_betafdPar,
        residential = default_betafdPar,
        unemployment = default_betafdPar,
        unemployment_vel = unemployment_vel_std,
        wages = default_betafdPar
      )
      betalist[[regressor_name]] <- betafdPar

      # fit model and calculate GCV
      tryCatch({
        model <- fRegress(consumption_fd, xfdlist, betalist)

        y_hat_matrix <- eval.fd(eval_points, model$yhatfd)
        y_matrix <- eval.fd(eval_points, consumption_fd)
        residuals <- y_matrix - y_hat_matrix
        RSS <- sum(residuals^2)
        df <- sum(diag(model$hatmat))
        n <- length(as.vector(y_matrix))
        GCV <- RSS / (n * (1 - df/n)^2)

        smoothStats[i, j] <- GCV

        if ((i %% 5 == 0) && (j %% 3 == 0)) {
          cat("  Completed nbasis =", nbasis, ", lambda =", lambda, ", GCV =", GCV, "\n")
        }
      }, error = function(e) {
```

```
        cat("  Error with nbasis =", nbasis, ", lambda =", lambda, ":", conditionMessage(e), "\n")
      })
    }
  }

  # minimum GCV
  min_idx <- which(smoothStats == min(smoothStats, na.rm = TRUE), arr.ind = TRUE)
  best_nbasis <- as.numeric(rownames(smoothStats)[min_idx[1]])
  best_log_lambda <- as.numeric(colnames(smoothStats)[min_idx[2]])
  best_lambda <- 10^best_log_lambda
  best_gcv <- smoothStats[min_idx]

  cat("\nBest parameters for", regressor_name, ":\n")
  cat("nbasis =", best_nbasis, "\n")
  cat("lambda =", best_lambda, "\n")
  cat("GCV =", best_gcv, "\n")

  return(list(
    nbasis = best_nbasis,
    lambda = best_lambda,
    log_lambda = best_log_lambda,
    gcv = best_gcv
  ))
}

const_params <- find_optimal_params("const")
commercial_params <- find_optimal_params("commercial")
residential_params <- find_optimal_params("residential")
unemployment_params <- find_optimal_params("unemployment")
unemployment_vel_params <- find_optimal_params("unemployment_vel")
wages_params <- find_optimal_params("wages")

# basis and parameter objects for each regressor
const_basis <- create.bspline.basis(rangeval = range(eval_points),
  nbasis = const_params$nbasis)
commercial_basis <- create.bspline.basis(rangeval = range(eval_points),
  nbasis = commercial_params$nbasis)
residential_basis <- create.bspline.basis(rangeval = range(eval_points),
  nbasis = residential_params$nbasis)
unemployment_basis <- create.bspline.basis(rangeval = range(eval_points),
  nbasis = unemployment_params$nbasis)
unemployment_vel_basis <- create.bspline.basis(rangeval = range(eval_points),
  nbasis = unemployment_vel_params$nbasis)
wages_basis <- create.bspline.basis(rangeval = range(eval_points),
  nbasis = wages_params$nbasis)

const_betafdPar <- fdPar(const_basis, 2, lambda = const_params$lambda)
commercial_betafdPar <- fdPar(commercial_basis, 2, lambda = commercial_params$lambda)
residential_betafdPar <- fdPar(residential_basis, 2, lambda = residential_params$lambda)
unemployment_betafdPar <- fdPar(unemployment_basis, 2, lambda = unemployment_params$lambda)
unemployment_vel_betafdPar <- fdPar(unemployment_basis, 2, lambda = unemployment_vel_params$lambda)
wages_betafdPar <- fdPar(wages_basis, 2, lambda = wages_params$lambda)
```

Parameter optimization All predictors have an optimal λ value of 10^{-5}, indicating that minimal smoothing is preferred. The optimal number of basis functions are as follows: for intercept 24, commercial and residential customer counts 20, unemployment 13, unemployment rate 17.

```
# functional predictors
betalist <- list(
  const = const_betafdPar,
  commercial = commercial_betafdPar,
  residential = residential_betafdPar,
  unemployment = unemployment_betafdPar,
  unemployment_vel = unemployment_vel_std,
  wages = wages_betafdPar
)

# function-on-function regression
fof_model <- fRegress(consumption_fd, xfdlist, betalist)

betaestlist <- fof_model$betaestlist

month_positions <- seq(min(eval_points), max(eval_points), length.out=12)

par(mfrow=c(2,3))

# plot intercept function
plot(betaestlist$const$fd, main="Baseline consumption",
     xlab="Month", ylab="Consumption (kWh/person)",
     xaxt="n")
```

```
## [1] "done"
```

```
axis(1, at=month_positions)

# plot commercial customers effect function
plot(betaestlist$commercial$fd,
     main="Effect of commercial customers",
     xlab="Month", ylab="Effect magnitude",
     xaxt="n")
```

```
## [1] "done"
```

```
axis(1, at=month_positions)
abline(h=0, lty=2)

# plot residential customers effect function
plot(betaestlist$residential$fd,
     main="Effect of residential customers",
     xlab="Month", ylab="Effect magnitude",
     xaxt="n")
```

```
## [1] "done"
```

```
axis(1, at=month_positions)
abline(h=0, lty=2)

# plot unemployment velocity effect function
plot(betaestlist$unemployment$fd,
     main="Effect of unemployment",
     xlab="Month", ylab="Effect magnitude",
     xaxt="n")
```

```
## [1] "done"
```

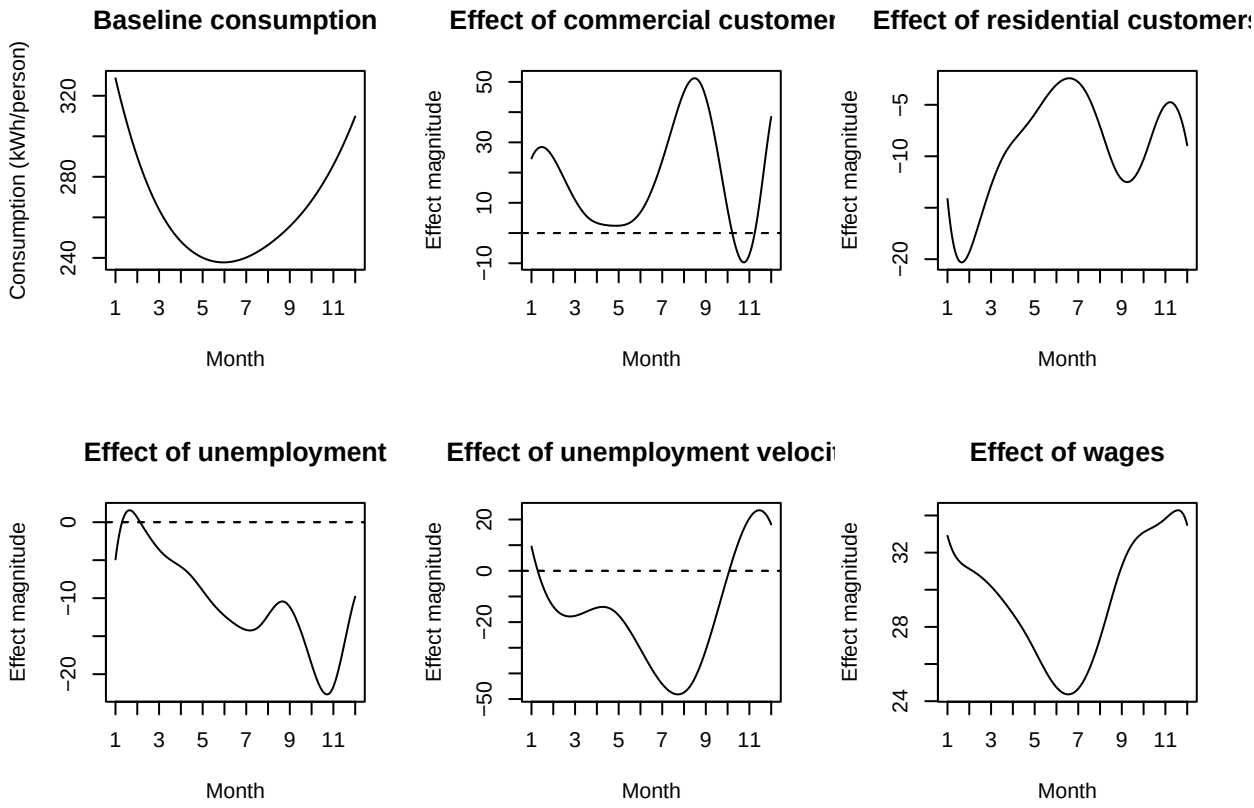
```
axis(1, at=month_positions)
abline(h=0, lty=2)

# plot unemployment velocity effect function
plot(betaestlist$unemployment_vel$fd,
     main="Effect of unemployment velocity",
     xlab="Month", ylab="Effect magnitude",
     xaxt="n")
```

```
## [1] "done"
```

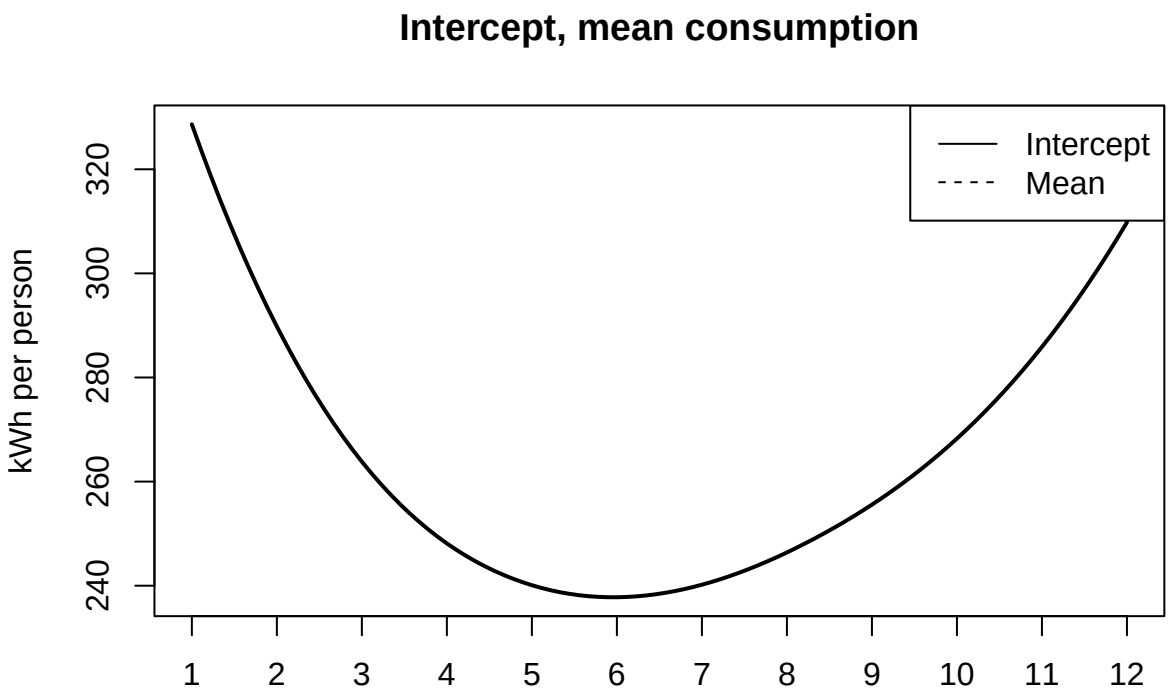
```
axis(1, at=month_positions)
abline(h=0, lty=2)

# plot wages effect function
plot(betaestlist$wages$fd,
     main="Effect of wages",
     xlab="Month", ylab="Effect magnitude",
     xaxt="n")
```

```
## [1] "done"
```

```
axis(1, at=month_positions)
abline(h=0, lty=2)
```

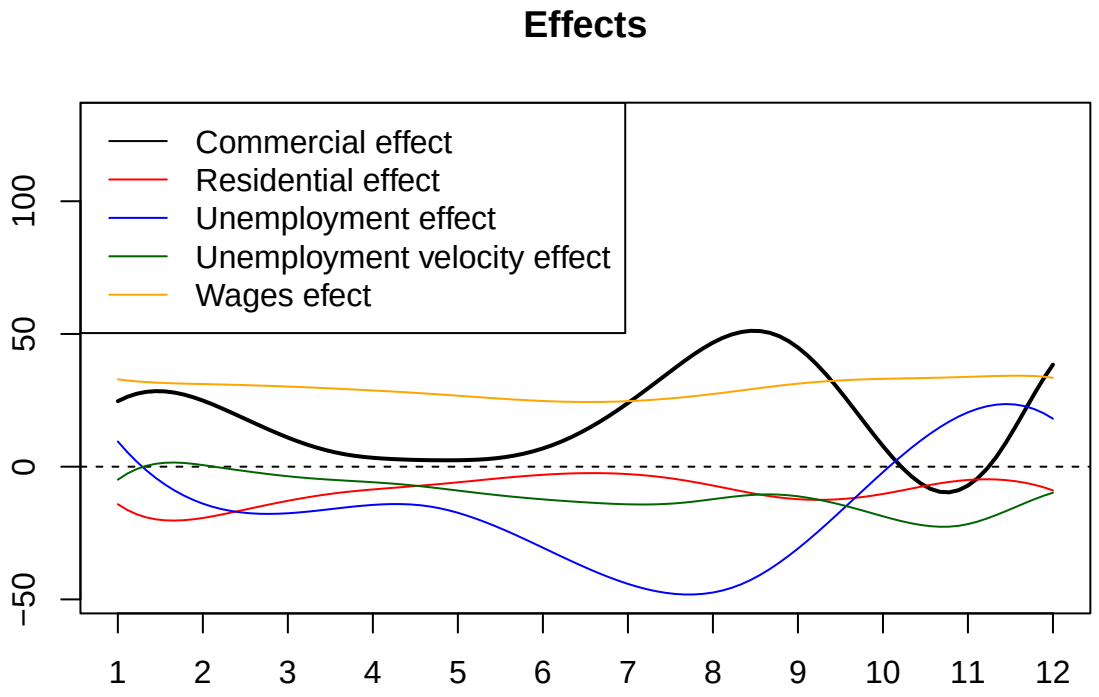



```
par(mfrow=c(1,1))
intercept_curve <- predict(betaestlist$const$fd, eval_points)
consumption_mean <- predict(mean.fd(consumption_fd), eval_points)

# intercept and mean consumption plot (overlap due to regressor standardization)
ylim1 <- range(consumption_mean, intercept_curve)

plot(eval_points, intercept_curve, ylim=ylim1, lwd=2,
     main="Intercept, mean consumption", type='l',
     xlab='', ylab='kWh per person', xaxt="n")
axis(1, at=month_positions)
lines(eval_points, consumption_mean, lty='dashed')
abline(h=0, lty='dashed')
legend("topright", legend=c("Intercept", "Mean"),
       lty=c(1,2), col=c("black", "black"))
```

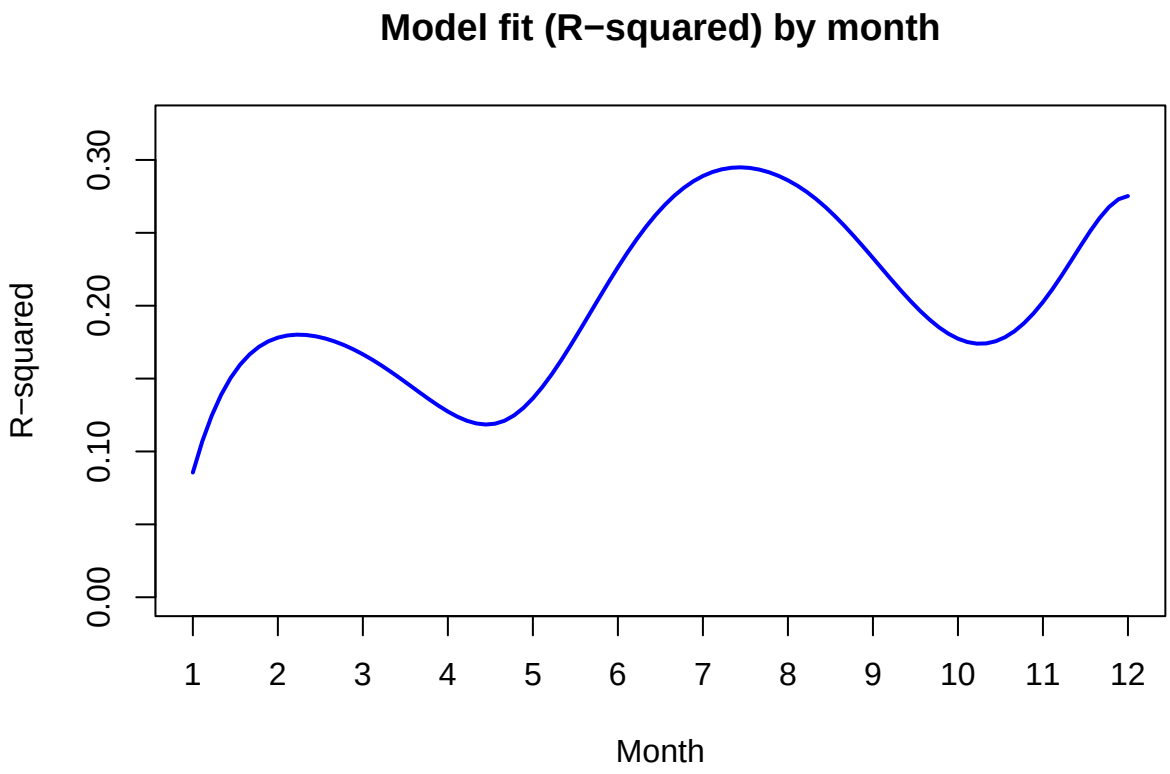



```
commercial_coef <- predict(betaestlist$commercial$fd, eval_points)
residential_coef <- predict(betaestlist$residential$fd, eval_points)
unemployment_coef <- predict(betaestlist$unemployment$fd, eval_points)
unemployment_vel_coef <- predict(betaestlist$unemployment_vel$fd, eval_points)
wages_coef <- predict(betaestlist$wages$fd, eval_points)

consumption_hat <- fof_model$yhatfd
consumption_hat_matrix <- eval.fd(eval_points, consumption_hat)
consumption_matrix <- eval.fd(eval_points, consumption_fd)
consumption_mean_matrix <- eval.fd(eval_points, mean.fd(consumption_fd))

# residuals
resmat <- consumption_matrix - consumption_hat_matrix
ncurve <- ncol(consumption_matrix)
resmat0 <- consumption_matrix - consumption_mean_matrix %*% matrix(1,1,ncurve)

# R-squared at each time point
SSE0 <- apply(resmat0^2, 1, sum)
SSE1 <- apply(resmat^2, 1, sum)
R2_by_time <- (SSE0-SSE1)/SSE0

ylim2 <- c(min(0, min(commercial_coef), min(residential_coef), min(unemployment_coef), min(unemployment_
           max(commercial_coef, residential_coef, R2_by_time, unemployment_coef, unemployment_vel_coef,
plot(eval_points, commercial_coef, lwd=2, xlab='', ylab='', ylim=ylim2, type='l',
     main='Effects', xaxt="n")
axis(1, at=month_positions)
```

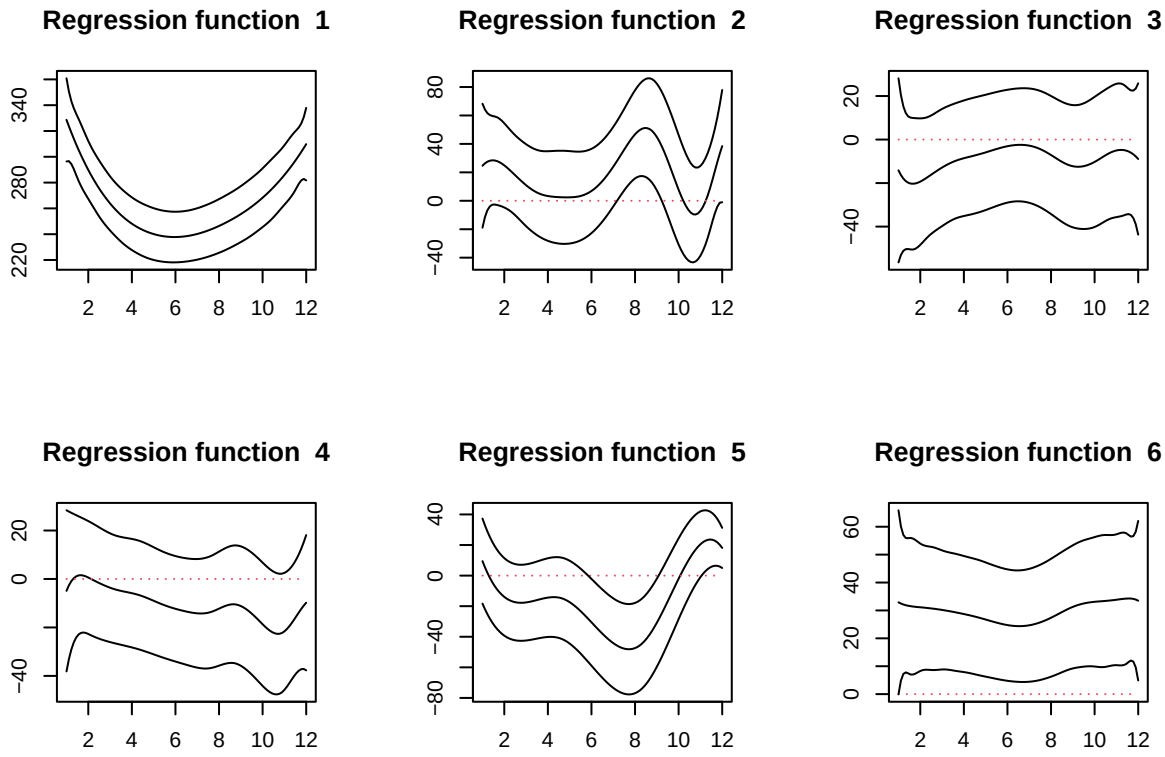
```
abline(h=0, lty='dashed')
lines(eval_points, residential_coef, lty=1, col="red")
lines(eval_points, unemployment_vel_coef, lty=1, col="blue")
lines(eval_points, unemployment_coef, lty=1, col="darkgreen")
lines(eval_points, wages_coef, lty=1, col="orange")
legend("topleft",
       legend=c("Commercial effect", "Residential effect", "Unemployment effect", "Unemployment velocity
       lty=1, col=c("black", "red", "blue", "darkgreen", "orange"))
```




```
op <- par(mfrow=c(1,1))
ylim_r2 <- c(0, max(R2_by_time) * 1.1)
plot(eval_points, R2_by_time, lwd=2, xlab='Month', ylab='R-squared',
     type='l', col="blue", main='Model fit (R-squared) by month',
     ylim=ylim_r2, xaxt="n")
axis(1, at=month_positions)
```




```
par(op)

y2cMap <- smooth.basis(eval_points, t(as.matrix(df$consumption)), fdPar(consumption_basis))$y2cMap
SigmaE <- cov(t(resmat))

# standard errors
fRegressList1 <- fRegress(consumption_fd, xfdlist, betalist, y2cMap=y2cMap, SigmaE=SigmaE)
fRegressList2 <- fRegress.stderr(fRegressList1, y2cMap, SigmaE)
betastderrlist <- fRegressList2$betastderrlist

# coefficients with confidence intervals
par(mfrow=c(2,3))
plotbeta(betaestlist, betastderrlist, eval_points)
```




```
# calculate R-squared
calculate_r2 <- function(model) {
  yhat <- eval.fd(eval_points, model$yhatfd)
  y <- eval.fd(eval_points, consumption_fd)
  SSE <- sum((y - yhat)^2)
  SST <- sum((y - mean(y))^2)
  R2 <- 1 - SSE/SST
  return(R2)
}

# R-squared
r2_optimized <- calculate_r2(fof_model)

cat(sprintf("R-squared:  %.4f\n", r2_optimized))
```

```
## R-squared:  0.2535
```

The function-on-function regression model achieves $R^2 = 0.25$, explaining 25% of variance in electricity consumption patterns, which is an improvement over the function-on-scalar approach (19% variance explained), demonstrating the value of allowing predictor relationships to vary in time.

Intercept coefficient curve (Regression function 1) is aligned with the mean consumption curve due to predictor standardization. It captures the baseline consumption when all standardized predictors equal to zero.

Commercial customer count (Regression function 2) has a positive effect on consumption almost year-round, with a small exception around November when it slightly drops below zero. It peaks in July-September and around January-February, possibly related to business/tourism peak seasons.

Residential customer effect (Regression function 3) is negative year-round, with the lowest values around January-February.

Unemployment rate (Regression function 4) has an overall negative effect on consumption, reaching the lowest value in November but quickly going back up towards the Winter holiday season. Interestingly, unemployment rate has a positive effect on energy consumption for a brief period in February.

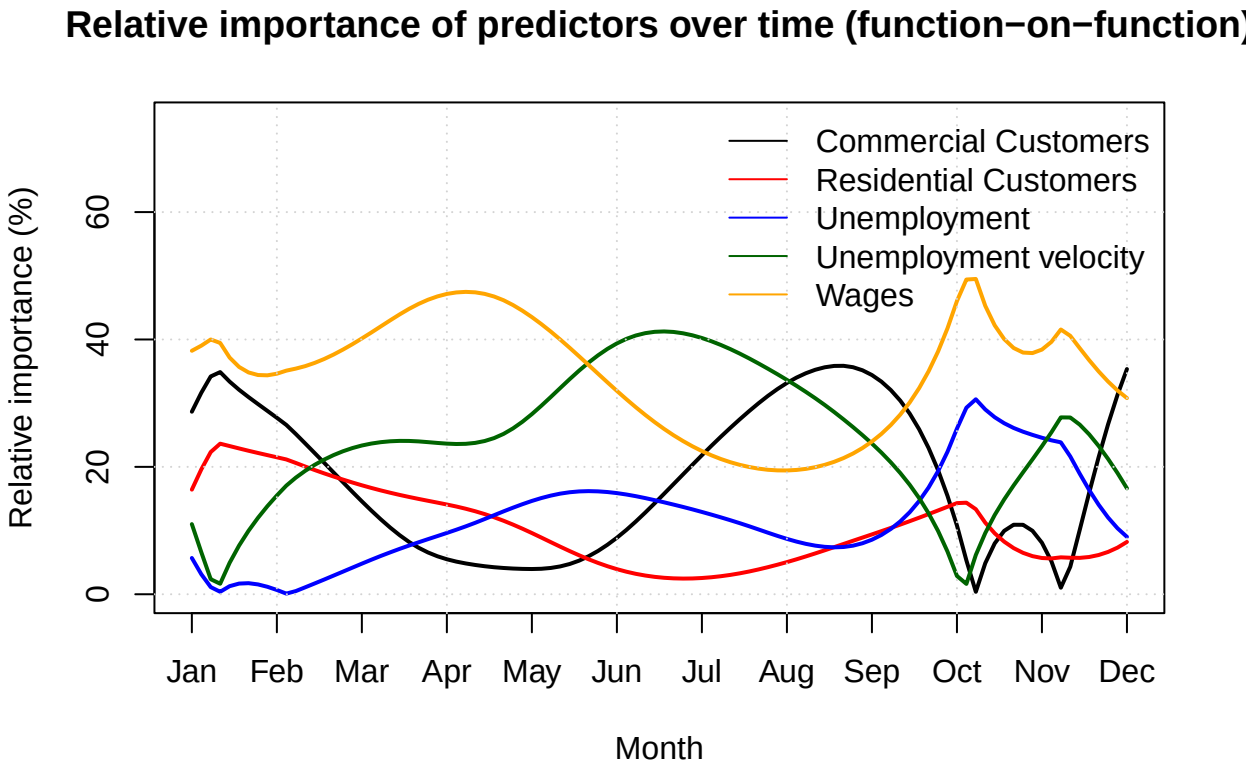
Unemployment velocity (Regression function 5) effect has a complex tendency, being generally negative (reaching extreme negative effect during the Summer) but attaining positive values towards the end of the year.

Effect of wages is positive throughout the year but drops in the middle of the year, resulting in a U-shaped seasonal pattern. This could be explained by the logic that wages effect is the highest when energy demand is the highest (during the heating season in Winter) and becomes less impactful when this demand is reduced.

The graph visualizing how R^2 varies across the months suggests that energy consumption is most predictable during summer when tourism, agriculture, and construction activities reach stable, peak operating patterns. In contrast, predicting energy consumption during the winter is more challenging because heating demand introduces weather-dependent variability. This suggests, that additional predictors (e.g., weather temperature) could potentially help to better predict energy consumption during those months.

The plot showing confidence intervals for all coefficient functions reveals that only the wages effect is consistently statistically significant throughout the year as its confidence intervals do not include zero at any point. Other predictors have a non-uniform significance throughout the year.

```
# relative importance of each predictor by month
total_effect <- abs(commercial_coef) + abs(residential_coef) + abs(unemployment_coef) +
  abs(unemployment_vel_coef) + abs(wages_coef)
commercial_importance <- abs(commercial_coef) / total_effect * 100
residential_importance <- abs(residential_coef) / total_effect * 100
unemployment_vel_importance <- abs(unemployment_vel_coef) / total_effect * 100
unemployment_importance <- abs(unemployment_coef) / total_effect * 100
wages_importance <- abs(wages_coef) / total_effect * 100

plot(eval_points, commercial_importance, type="l", lwd=2, col="black",
     xlab="Month", ylab="Relative importance (%)",
     main="Relative importance of predictors over time (function-on-function)",
     ylim=c(0, max(commercial_importance, residential_importance, unemployment_importance,
       unemployment_vel_importance, wages_importance) * 1.5),
     xaxt="n")
axis(1, at=month_positions, labels=month.abb)
lines(eval_points, residential_importance, lwd=2, col="red", lty=1)
lines(eval_points, unemployment_importance, lwd=2, col="blue", lty=1)
lines(eval_points, unemployment_vel_importance, lwd=2, col="darkgreen", lty=1)
lines(eval_points, wages_importance, lwd=2, col="orange", lty=1)
grid(NULL, NULL, lty=3, col="lightgray")

legend("topright",
       legend=c("Commercial Customers", "Residential Customers",
         "Unemployment", "Unemployment velocity", "Wages"),
       lty=1,
       col=c("black", "red", "blue", "darkgreen", "orange"),
       bty="n")
```




```
cat(sprintf("Mean importance of the count of commercial customers per person: %.2f%%\n",
  mean(commercial_importance)))
```

```
## Mean importance of the count of commercial customers per person: 18.22%
```

```
cat(sprintf("Mean importance of the count of residential customers per person: %.2f%%\n",
  mean(residential_importance)))
```

```
## Mean importance of the count of residential customers per person: 10.62%
```

```
cat(sprintf("Mean importance of unemployment: %.2f%%\n", mean(unemployment_importance)))
```

```
## Mean importance of unemployment: 11.96%
```

```
cat(sprintf("Mean importance of unemployment velocity: %.2f%%\n", mean(unemployment_vel_importance)))
```

```
## Mean importance of unemployment velocity: 24.47%
```

```
cat(sprintf("Mean importance of wages: %.2f%%\n", mean(wages_importance)))
```

```
## Mean importance of wages: 34.74%
```

The function-on-function relative importance analysis reveals significantly more complex seasonal dynamics compared to the function-on-scalar approach, with predictor hierarchies shifting substantially throughout the year. On average, the most important predictor is wages (34.74%), but with significant seasonal variations. The second most important predictor is unemployment velocity 24.47%, followed by commercial customer count per person (18.22%), unemployment (11.96%), and residential customer count per person (10.62%).