Cranes data analysis

Supplementary material for 'Efficient Curve Fitting with Penalized B-Splines for Oceanographic and Ecological Applications'

Ju-Seong Lee and Jae-Hwan Jhong

2025-05-29

```
rm(list = ls())
source("admm_source.R")
source("splineBox.R")

## packages load
library(dplyr)
library(ggplot2)
library(sf)
library(rnaturalearth)
library(rnaturalearthdata)
library(cowplot)
library(rworldmap)
library(sphereplot)
```

```
## data time setting
start_date <- as.POSIXct("2018-08-09 00:00:00")
end_date <- as.POSIXct("2019-08-09 23:59:59")


## pre-preocessing
df = read.csv("Data/Cranes.csv")
data = df[, c("individual.local.identifier", "timestamp", "location.long","location.lat")]
data = data[order(data$timestamp), ]

## time data extraction
time = data["timestamp"]
time = as.matrix(time)

## POSIXct transform
timestamps <- as.POSIXct(time, format = "%Y-%m-%d %H:%M:%S")

data <- data %>%
   filter(timestamps >= start_date & timestamps <= end_date)

## time pre-processing
time = data["timestamp"]
time = as.matrix(time)

# 1. raw_data
raw_data <- time
```

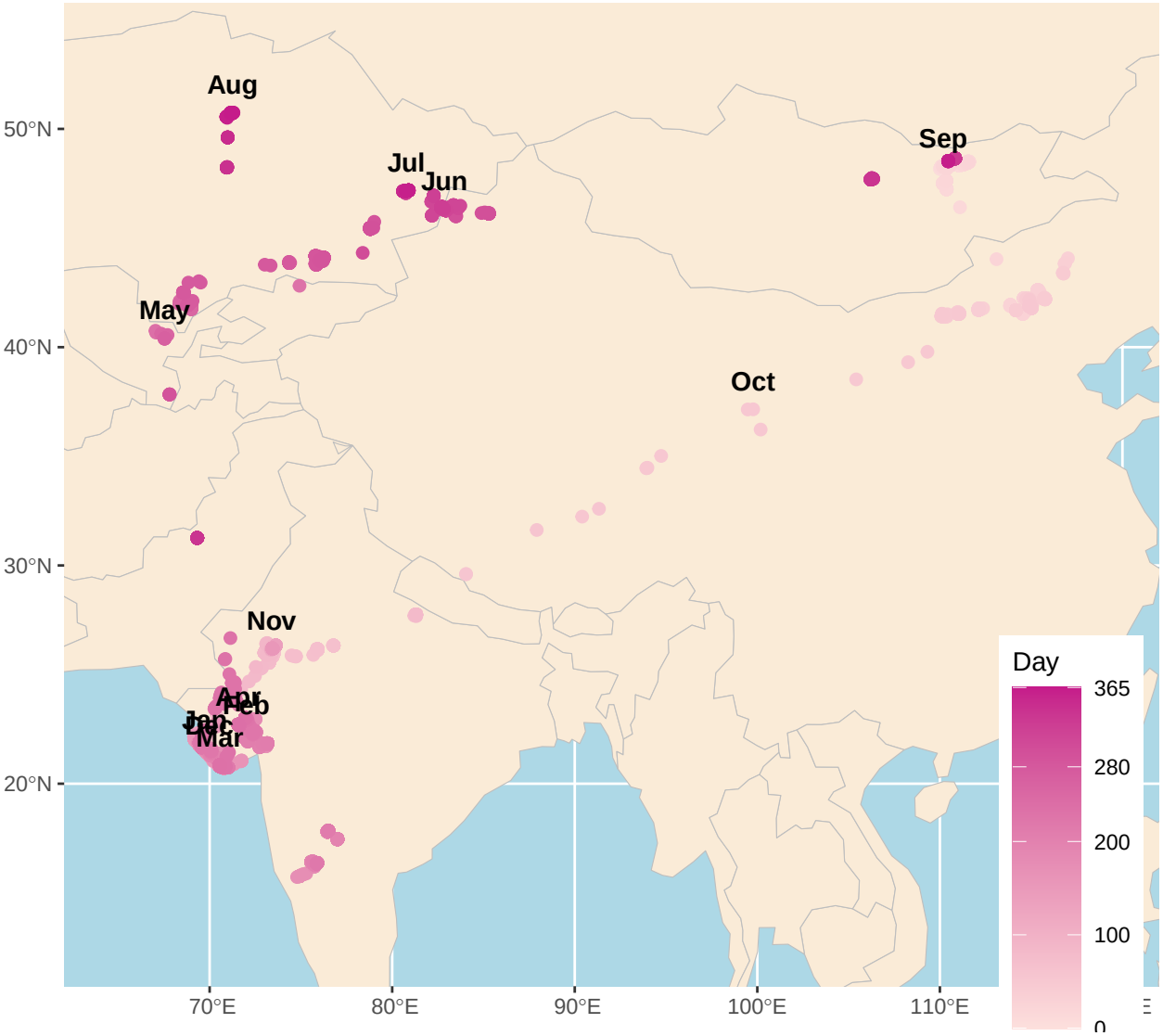
```r
timestamps <- as.POSIXct(raw_data, format = "%Y-%m-%d %H:%M:%S")

# set standard time
start_time <- timestamps[1]

# calculate time difference in minutes
time_diffs <- as.numeric(difftime(timestamps, start_time, units = "hours"))

df_time <- data.frame(
   OriginalTime = timestamps,
   TimeElapsed = time_diffs
)

t = df_time$TimeElapsed
t = as.matrix(t)

# make t range 0~365
t = t / max(t) * 365
data$time = t

# define y
y1 = data["location.long"]
y2 = data["location.lat"]

y = NULL
y = cbind(y1, y2)
y = as.matrix(y)


####################
## all_data fitting
####################
order = 3
dimension = 200
knots = knots_quantile(t, dimension, order)
B = bsplines(t, knots, order)
D = bspline_jump(knots, order)

# model fitting
fit = bspline.curve.admm_lambdas(y, D, B,
                                 lambdas = NULL,
                                 lam_max = 10,
                                 lam_min = 1e-5,
                                 n_lambda = 300,
                                 max_iter = 1000,
                                 epsilon = 1e-8,
                                 eta_c = 1)

best_index_bic = which.min(fit$bic)
best_index_aic = which.min(fit$aic)

best_index = which.min(fit$bic)
```

```r
#######################
## all_data world plot
#######################

## set world parameter
worldMap = getMap()
worldMap_sf = st_as_sf(worldMap)

mar = 5
xlim <- c(min(y[,1]) - mar, max(y[,1]) + mar)
ylim <- c(min(y[,2]) - mar, max(y[,2]) + mar)

my_data = data

# list of the first day of each month
monthly_t_values <- c(23, 53, 84, 114, 145, 176, 205, 236, 266, 298, 327, 358)
month_labels <- c("Sep", "Oct", "Nov", "Dec", "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug")

# find close index to each t
closest_indices <- sapply(monthly_t_values, function(t) {
  which.min(abs(my_data$time - t))
})

monthly_selected_data <- my_data[closest_indices, ]
monthly_selected_data$month <- month_labels


# raw_data
raw_data_plot <- ggplot() +
  geom_sf(data = worldMap_sf, fill = "antiquewhite", color = "grey") +
  coord_sf(xlim = xlim, ylim = ylim, expand = FALSE) +

  geom_point(data = my_data, aes(x = location.long, y = location.lat, color = time),
             size = 2) +

  scale_color_gradient(low = "#fde0dd", high = "#c51b8a", name = "Day",
                       breaks = c(365, 280, 200, 100, 0),
                       limits = c(min(my_data$time, na.rm = TRUE),
                                  max(my_data$time, na.rm = TRUE))) +

  geom_text(data = monthly_selected_data,
            aes(x = location.long, y = location.lat, label = month),
            color = "black", size = 4, vjust = -1, fontface = "bold") +

  theme(panel.background = element_rect(fill = "lightblue", color = NA),
        axis.title.x = element_blank(),
        axis.title.y = element_blank(),
        legend.position = c(0.92, 0.15),
        legend.key.size = unit(1, "cm"))
```

```
## Warning: A numeric `legend.position` argument in `theme()` was deprecated in ggplot2
## 3.5.0.
## i Please use the `legend.position.inside` argument of `theme()` instead.
```

```
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
raw_data_plot
```

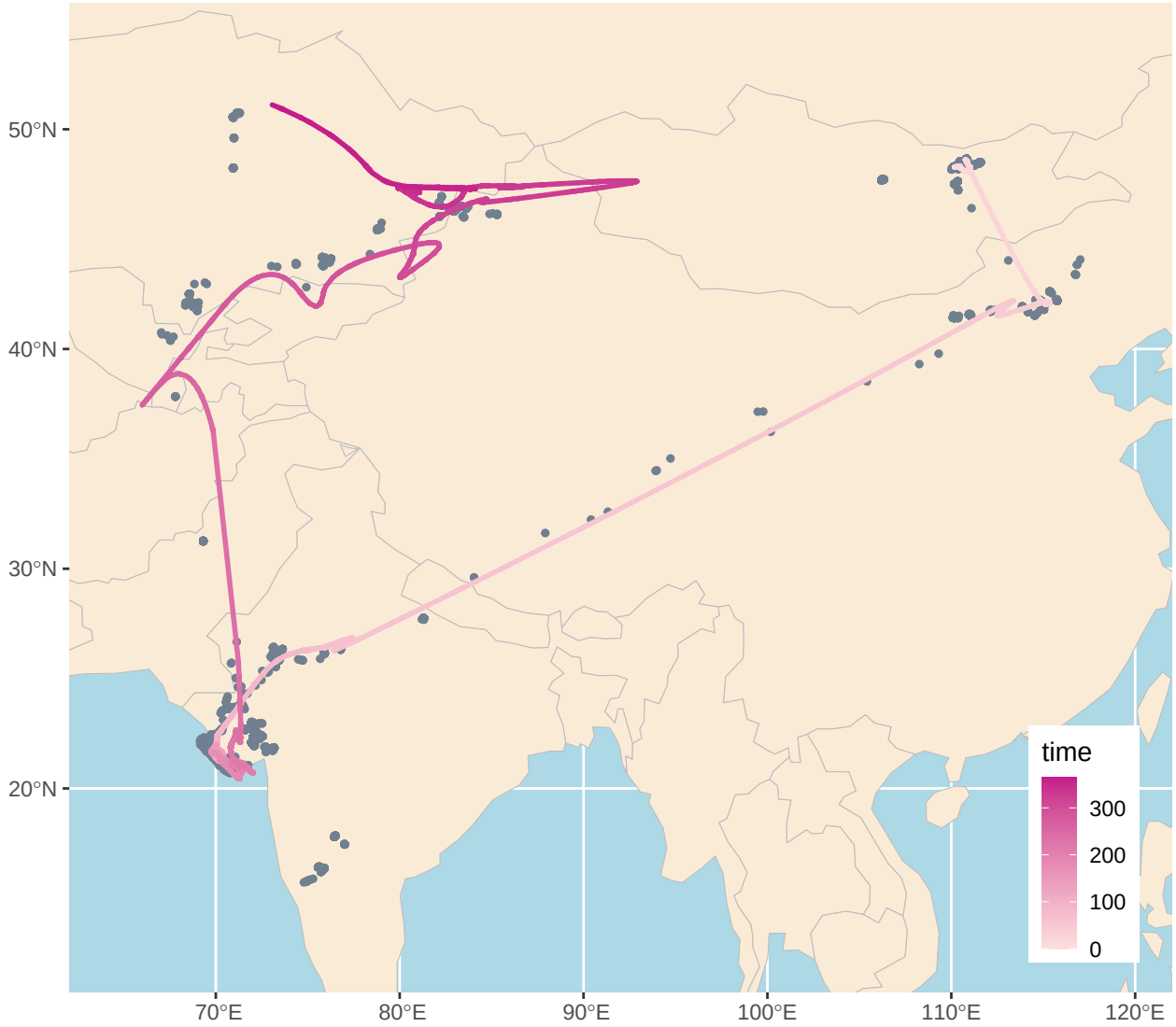



```
#####################
# best aic model plot
#####################
my_data = cbind(fit[[best_index_aic]]$Bb, data$time)
colnames(my_data)[3] = "time"
my_data = as.data.frame(my_data)

best_aic_plot = ggplot() +
   geom_sf(data = worldMap_sf, fill = "antiquewhite", color = "grey")  +
   coord_sf(xlim = xlim, ylim = ylim, expand = FALSE) +

   geom_point(data = data, aes(x = location.long, y = location.lat), color = 'slategray', size = 1) +
```

```
  geom_path(data = my_data, aes(x = location.long, y = location.lat, color = time),
            size = 1, lineend = "round") +

  scale_color_gradient(low = "#fde0dd", high = "#c51b8a") +

  theme(panel.background = element_rect(fill = "lightblue", color = NA),
        axis.title.x = element_blank(),
        axis.title.y = element_blank(),
        legend.position = c(0.92, 0.15),
        legend.key.size = unit(0.5, "cm"))
```

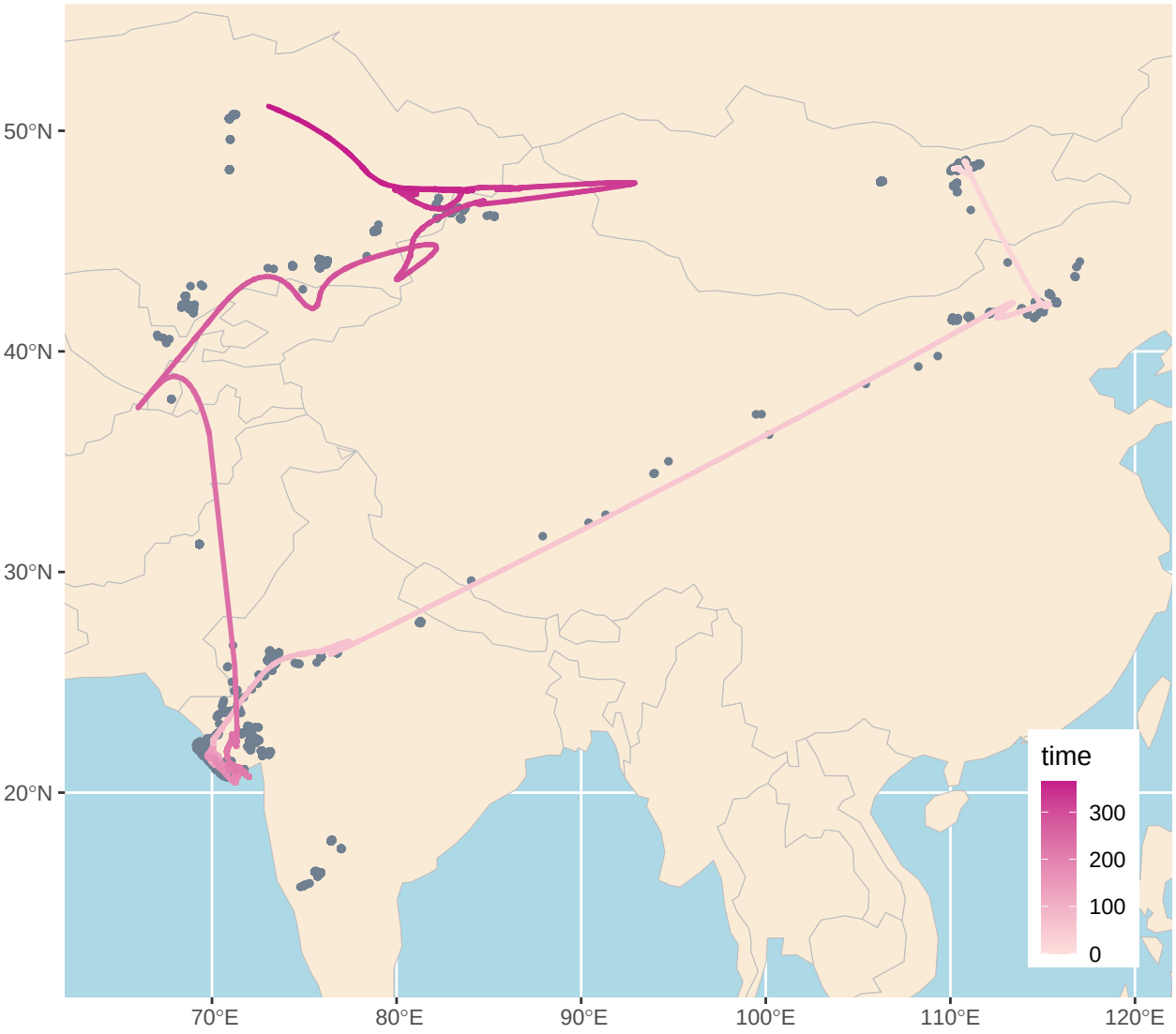
```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
best_aic_plot
```



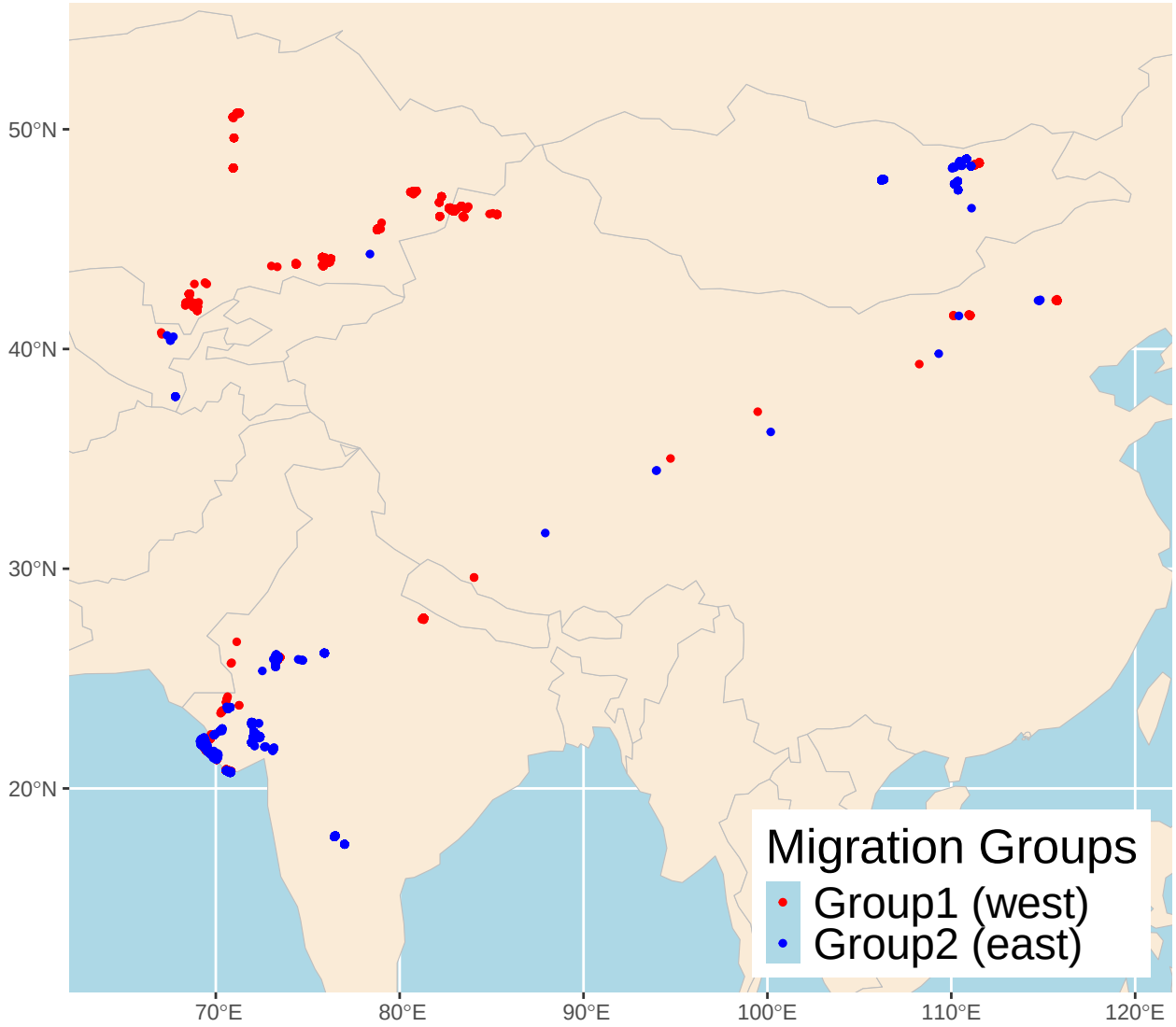

```
#####################
# best bic model plot
#####################
my_data = cbind(fit[[best_index_bic]]$Bb, data$time)
colnames(my_data)[3] = "time"
my_data = as.data.frame(my_data)

best_bic_plot = ggplot() +
   geom_sf(data = worldMap_sf, fill = "antiquewhite", color = "grey") +
   coord_sf(xlim = xlim, ylim = ylim, expand = FALSE) +
   geom_point(data = data, aes(x = location.long, y = location.lat), color = 'slategray', size = 1) +
      geom_path(data = my_data, aes(x = location.long, y = location.lat, color = time),
             size = 1, lineend = "round") +

   scale_color_gradient(low = "#fde0dd", high = "#c51b8a") +

   theme(panel.background = element_rect(fill = "lightblue", color = NA),
         axis.title.x = element_blank(),
         axis.title.y = element_blank(),
```

```
        legend.position = c(0.92, 0.15),
        legend.key.size = unit(0.5, "cm"))

best_bic_plot
```




```
########################
# left, right data split
########################
left_data <- subset(data, individual.local.identifier %in% c("H13-6280", "H17-6330"))

right_data <- subset(data, individual.local.identifier %in% c("H29-6269", "H33-6233", "H71-6237"))

left_data$group <- "Group1 (west)"
right_data$group <- "Group2 (east)"

# combine two data
combined_data <- rbind(left_data, right_data)
```

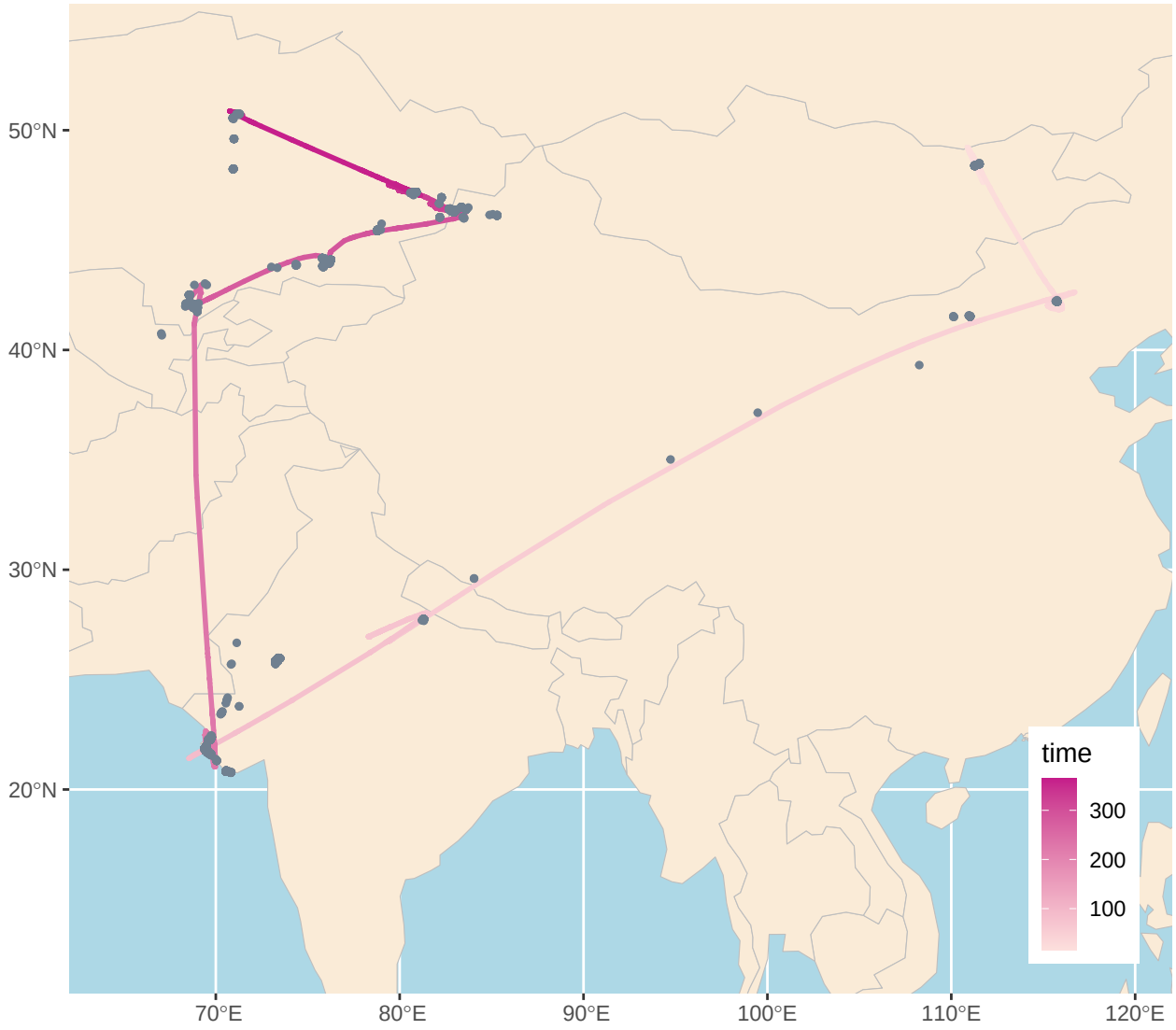
```
group_data_plot = ggplot() +
   geom_sf(data = worldMap_sf, fill = "antiquewhite", color = "grey") +
   coord_sf(xlim = xlim, ylim = ylim, expand = FALSE) +
   geom_point(data = combined_data, aes(x = location.long, y = location.lat, color = group), size = 1) +

   scale_color_manual(
      values = c("Group1 (west)" = "red",
                 "Group2 (east)" = "blue"),
      name = "Migration Groups"
   ) +
   theme(
      panel.background = element_rect(fill = "lightblue", color = NA),
      legend.position = c(0.8, 0.1),
      legend.background = element_rect(fill = "white", color = NA),
      legend.title = element_text(size = 20),
      legend.text = element_text(size = 18),
      axis.title.x = element_blank(),
      axis.title.y = element_blank(),
      legend.key.size = unit(0.5, "cm")
   )
group_data_plot
```




```
#####################
## left_model fitting
#####################
y1 = left_data["location.long"]; y2 = left_data["location.lat"]
y = NULL
y = cbind(y1, y2)
y = as.matrix(y)

t = left_data$time

order = 3
dimension = 100
knots = knots_quantile(t, dimension, order)
B = bsplines(t, knots, order)
D = bspline_jump(knots, order)

# model fitting
fit1 = bspline.curve.admm_lambdas(y, D, B,
                                  lambdas = NULL,
```

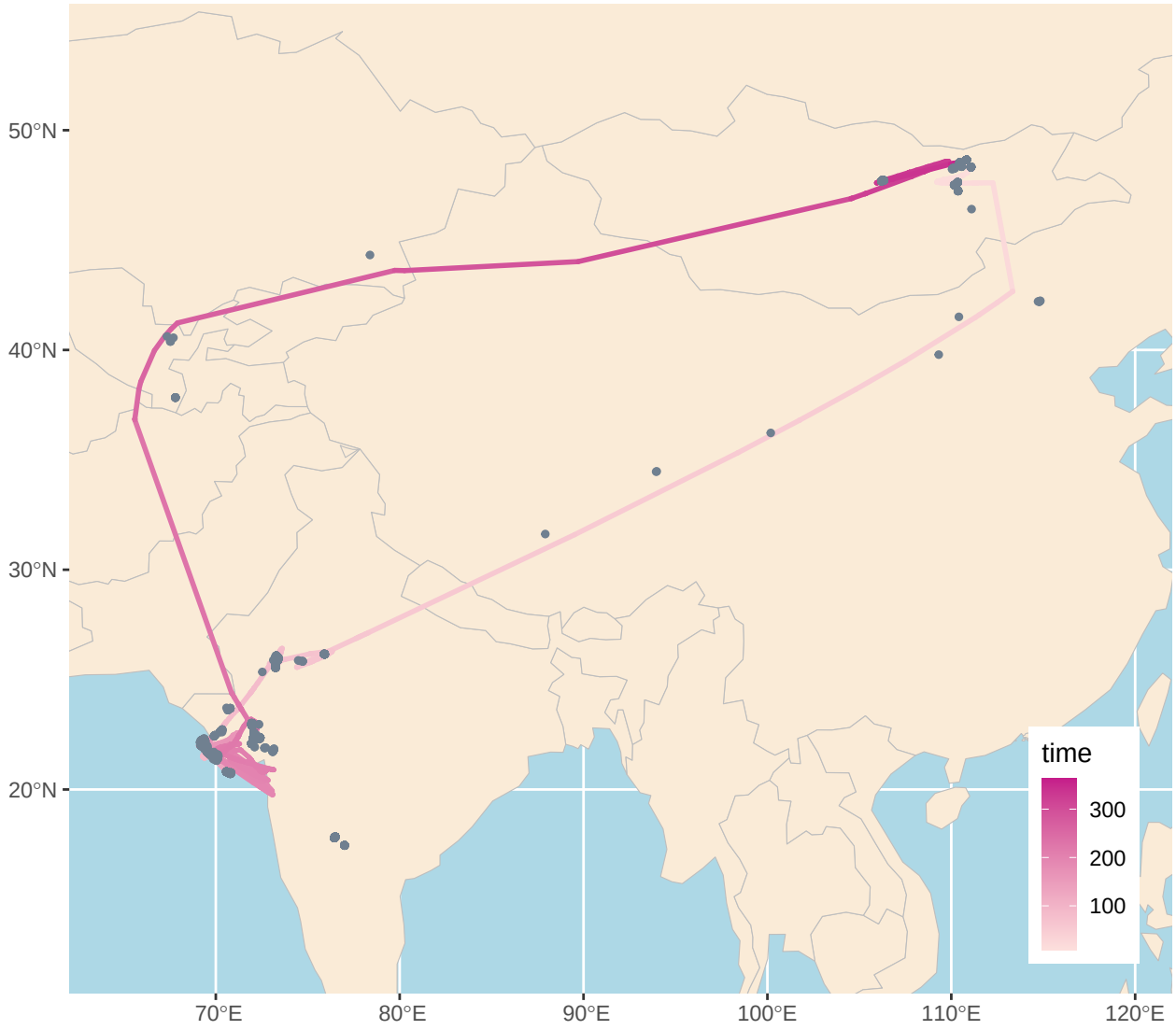
```
                                    lam_max = 10,
                                    lam_min = 1e-5,
                                    n_lambda = 300,
                                    max_iter = 1000,
                                    epsilon = 1e-8,
                                    eta_c = 1)

best_index1 = which.min(fit1$bic)

my_data1 = cbind(fit1[[best_index1]]$Bb, left_data$time)
colnames(my_data1)[3] = "time"
my_data1 = as.data.frame(my_data1)


######################
# right_model fitting
######################
y1 = right_data["location.long"]; y2 = right_data["location.lat"]
y = NULL
y = cbind(y1, y2)
y = as.matrix(y)
t = right_data$time

order = 3
dimension = 100
knots = knots_quantile(t, dimension, order)
B = bsplines(t, knots, order)
D = bspline_jump(knots, order)

# model fitting
fit2 = bspline.curve.admm_lambdas(y, D, B,
                                    lambdas = NULL,
                                    lam_max = 10,
                                    lam_min = 1e-5,
                                    n_lambda = 300,
                                    max_iter = 1000,
                                    epsilon = 1e-8,
                                    eta_c = 1)

best_index2 = which.min(fit2$bic)

my_data2 = cbind(fit2[[best_index2]]$Bb, right_data$time)
colnames(my_data2)[3] = "time"
my_data2 = as.data.frame(my_data2)


######################
## left model plot
######################
left_plot = ggplot() +
   geom_sf(data = worldMap_sf, fill = "antiquewhite", color = "grey") +
   coord_sf(xlim = xlim, ylim = ylim, expand = FALSE) +
```

```
   geom_path(data = my_data1, aes(x = location.long, y = location.lat, color = time),
             size = 1, lineend = "round") +
   scale_color_gradient(low = "#fde0dd", high = "#c51b8a") +

   geom_point(data = left_data, aes(x = location.long, y = location.lat),
              color = "slategrey", size = 1) +

   theme(panel.background = element_rect(fill = "lightblue", color = NA),
         axis.title.x = element_blank(),
         axis.title.y = element_blank(),
         legend.position = c(0.92, 0.15),
         legend.key.size = unit(0.5, "cm"))
left_plot
```




```
######################
## right model plot
######################
right_plot = ggplot() +
```

```
   geom_sf(data = worldMap_sf, fill = "antiquewhite", color = "grey") +
   coord_sf(xlim = xlim, ylim = ylim, expand = FALSE) +
   geom_path(data = my_data2, aes(x = location.long, y = location.lat, color = time),
             size = 1, lineend = "round") +
   scale_color_gradient(low = "#fde0dd", high = "#c51b8a") +
   geom_point(data = right_data, aes(x = location.long, y = location.lat),
              color = "slategrey", size = 1) +
   theme(panel.background = element_rect(fill = "lightblue", color = NA),
         axis.title.x = element_blank(),
         axis.title.y = element_blank(),
         legend.position = c(0.92, 0.15),
         legend.key.size = unit(0.5, "cm"))
right_plot
```




```
### overlay Stopover_Sites
stopover_site = read.csv("Data/Stopover_Sites_Data.csv")

stopover_plot <- ggplot() +
```

```
  geom_sf(data = worldMap_sf, fill = "antiquewhite", color = "grey") +
  coord_sf(xlim = xlim, ylim = ylim, expand = FALSE) +
  geom_point(data = data, aes(x = location.long, y = location.lat),
             color = 'slategray', size = 2) +
  geom_path(data = my_data, aes(x = location.long, y = location.lat),
            size = 1.5, lineend = "round", color = "blue") +
  geom_point(data = stopover_site, aes(x = Longitude, y = Latitude, color = as.factor(PTT_ID)),
             size = 5, shape = 21, fill = 'white', stroke = 1.2) +
  theme(panel.background = element_rect(fill = "lightblue", color = NA),
        axis.title.x = element_blank(),
        axis.title.y = element_blank(),
        legend.position = "right",
        legend.key.size = unit(1, "cm")) +
  labs(color = "PTT_ID")

stopover_plot
```