Semi-Markov Process Construction

Ronan

2022-10-27

```
## This script takes a Disco derived CSV output and generates a semi-markov process
## along with mean wait times and limiting probabilities for each state. It also finds
## the mean hitting time for state e.

#Load the packages
pacman::p_load(
  tidyverse,
  lubridate
)

#Define the file of interest
file <- "CSV_Files/BPI_Challenge_2013_closed_problems.csv"

#Read in the data
data <- read.csv(file) |>
  mutate(
    across(!where(is.factor),as_factor),
    Complete.Timestamp = as.character(Complete.Timestamp)
  )

#Add a time difference column to each row
data$time_diff <- 0

#Calculate the time taken in seconds for each step.
for(i in 2:dim(data)[1]){
  if(data$Case.ID[i-1] == data$Case.ID[i]){
    data$time_diff[i] <- dseconds(interval(
      data$Complete.Timestamp[i-1],
      data$Complete.Timestamp[i]
    )
    )
  }
}

#Add a state identifier to each row
data$state_id <- data |>
  group_indices(Activity)
```

```
## Warning: The `...` argument of `group_indices()` is deprecated as of dplyr 1.0.0.
## Please `group_by()` first
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

```r
#Get the unique cases
unique_cases <- unique(data$Case.ID)

#Get the state names and their numeric value
state_names <- data |>
  select(
    Activity,
    state_id
  ) |>
  distinct()

#Initialise the  count matrix and state counts.
counts <- matrix(0, nrow = max(data$state_id), ncol = max(data$state_id))
state_counts <- matrix(0, nrow = 1, ncol = max(data$state_id))
initial_state_counts <- matrix(0, nrow = 1, ncol = max(data$state_id))

#Let's get the transition matrix
for(i in 1:length(unique_cases)){
  #Filter the data per case
  temp_data <- data |>
    filter(
      Case.ID == unique_cases[i]
    ) |>
    select(
      state_id
    )

  #Get the counts of each transition
  for(j in 2:dim(temp_data)[1]){
    counts[temp_data$state_id[j-1], temp_data$state_id[j]] <-
      counts[temp_data$state_id[j-1], temp_data$state_id[j]] + 1
  }

  #Get the counts of each state
  for(j in 1:(dim(temp_data)[1] - 0)){
    state_counts[temp_data$state_id[j]] <- state_counts[temp_data$state_id[j]] + 1
  }

  #Get the initial state counts
  initial_state_counts[temp_data$state_id[1]] <-
    initial_state_counts[temp_data$state_id[1]] + 1
}

#Get the initial state transition probabilities
initial_state_probabilities <- initial_state_counts * (1/sum(initial_state_counts))

#Let's get the transition matrix and add an end state.
P <- rbind(cbind(counts, matrix(0, nrow = dim(counts)[1], ncol = 1)), #Initialisng
           matrix(0, nrow = 1, ncol = (dim(counts)[1] + 1)))
for(i in 1:dim(P)[1]){
  #Divide by state count
  P[i,] <- (1/state_counts[i]) * (P[i,])
}
```

```r
#Let's add the starting state to the transition matrix.
P <- rbind( matrix(0, nrow = 1, ncol = (dim(P)[2] + 1)),
            cbind(matrix(0, nrow = dim(P)[1], ncol = 1), P))

#Loop transitions to the end state.
for(i in 2:(dim(P)[1] - 1)){
  if(sum(P[i,]) != 1){
    P[i,dim(P)[2]] = 1 - sum(P[i,])
  }
}
#Add the start and end transition probabilities
P[1,] <- c(0, initial_state_probabilities, 0)
P[dim(P)[2],] <- c(1,rep(0,(dim(P)[2] - 1)))


#Initialise the waiting time matrix as a list.
F_wait <- list()

#Initialise the mean time spend in each state
mean_times <- c(rep(0,max(data$state_id)))

#Generate the F_wait list of pmfs.
list_index <- 1;
for(i in 1:max(data$state_id)){
  #Initialise the current mean times.
  current_mean_times <- c()

  for(j in 1:max(data$state_id)){
    #Grab the temporary data
    temp_data <- tibble()
    for(k in 2:dim(data)[1]){
      if((data$state_id[k] == j) &&
         (data$state_id[k-1] == i)){
        temp_data <- rbind(temp_data, data[k-1,], data[k,])
      }
    }

    temp_data <- temp_data[(seq_len(nrow(temp_data)) %% 2) == 0,]

    #Do some cleaning and summation
    if(is_empty(temp_data)){
      temp_f = "no transitions"
    } else{
      #Clean up a little bit
      temp_f <- temp_data |>
        tibble() |>
        select(
          state_id,
          time_diff
        ) |>
        mutate(
          yes = 1
        )
```

```
      #Add to the mean times vector
      current_mean_times <- c(current_mean_times, c(temp_f$time_diff))

      #Calculate some probabilities
      temp_f <- temp_f |>
        group_by(time_diff) |>
        summarise(
          sum(yes)
        )
      colnames(temp_f) <- c("time", "prob")

      #Get the probabilities
      temp_f$prob <- (temp_f$prob) * (1/sum(temp_f$prob))
    }

    #Add the PMF to the list
    F_wait[[list_index]] <- list(id = paste0("f",i,j), pmf = temp_f)
    list_index <- list_index + 1
  }

  #Get the mean time for state i
  mean_times[i] <- mean(current_mean_times)
}

#Get the limiting probabilities of the states
Pi <- eigen(t(P))$vectors[, 1]
Pi <- Re(Pi / sum(Pi))

#Get the mean hitting time for state e
e_state_mean <- (1/Pi[1]) * sum((Pi[2:(length(Pi) - 1)]*mean_times))
e_hit_time <- dseconds(e_state_mean)

#Get the worst state
worst_state <- c(state_names[which.max(Pi[2:(length(Pi) - 1)]*mean_times),])

#Let's suppose that we reduce the mean time for the worst state by 2
mean_times[worst_state[[2]]] <- mean_times[worst_state[[2]]]/2

#Recalculate the hitting time
e_state_mean <- (1/Pi[1]) * sum((Pi[2:(length(Pi) - 1)]*mean_times))
e_hit_time_half <- dseconds(e_state_mean)
```