Graph

Mingze Li 300137754

2025-02-13

```
library(traveltimeCLT)
library(data.table)
```

```
## Warning:   'data.table' R 4.3.3
```

```
library(traveltimeHMM)
```

```
##
##     'traveltimeHMM'
```

```
## The following objects are masked from 'package:traveltimeCLT':
##
##     rules2timebins, time_bins, time_bins_functional,
##     time_bins_readable, to7daybins
```

```
library(doParallel)
```

```
## Warning:   'doParallel' R 4.3.3
```

```
##      foreach
```

```
##      iterators
```

```
##      parallel
```

```
library(ggplot2)
```

```
## Warning:   'ggplot2' R 4.3.3
```

```
library(ggraph)
```

```
## Warning:   'ggraph' R 4.3.3
```

```
library(tidygraph)
```

```
## Warning:   'tidygraph' R 4.3.3
```

```
##
##     'tidygraph'

## The following object is masked from 'package:stats':
##
##     filter
```

```
library(igraph)
```

```
## Warning:   'igraph' R 4.3.3

##
##     'igraph'

## The following object is masked from 'package:tidygraph':
##
##     groups

## The following object is masked from 'package:traveltimeHMM':
##
##     time_bins

## The following object is masked from 'package:traveltimeCLT':
##
##     time_bins

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

```
trips = fread('data/trips.csv')
trips$time <- as.POSIXct( trips$time, format = "%Y-%m-%dT%H:%M:%OSZ")
trips$timeBin<-time_bins_readable(trips$time)
trips <-na.omit(trips)
trips[, duration_secs := as.numeric(difftime(shift(time, type = "lead"), time, units = "secs")), by = ti
trips[, log_duration := log(duration_secs)]
sd_na_is_0<-function(x){
  x=na.omit(x)
  if(length(x)==0)return(sd(x))
  if(length(x)>=2)return(sd(x))
  else return(0)
}
get_mode <- function(x) {
  x=na.omit(x)
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}
names(trips)[7]<-"length"
unique(trips$timeBin)
```

```
## [1] "EveningNight" "EveningRush"  "Weekday"      "MorningRush"  "Weekendday"
```

```
id = sample(unique(trips$trip),20)
cl <- makeCluster(8)
registerDoParallel(cl)
tripID <- unique(trips$trip)
link_net_list <- foreach(trip_id = tripID, .combine = rbind, .packages = "data.table") %dopar% {trip_li
  temp_dt <- data.table(linkID = integer(), nextLinkID = integer(), log_duration = numeric(), timeBin =
  if(length(trip_links$linkId)>1) for (i in 1:(length(trip_links$linkId) - 1)) {
    current_link <- trip_links$linkId[i]
    one_way_link <- trip_links$linkId[i + 1]
    log_duration <- trip_links$log_duration[i]
    timeBin <- trip_links$timeBin[i]
    Length <- trip_links$length[i]
    temp_dt <- rbind(temp_dt, list(linkID = current_link, nextLinkID = one_way_link,log_duration = log_
  }else return(data.table())
  temp_dt
}
sampled <- foreach(trip_id = id, .combine = rbind, .packages = "data.table") %dopar% {trip_links <- trip
  temp_dt <- data.table(linkID = integer(), nextLinkID = integer(), log_duration = numeric(), timeBin =
  if(length(trip_links$linkId)>1) for (i in 1:(length(trip_links$linkId) - 1)) {
    current_link <- trip_links$linkId[i]
    one_way_link <- trip_links$linkId[i + 1]
    log_duration <- trip_links$log_duration[i]
    timeBin <- trip_links$timeBin[i]
    temp_dt <- rbind(temp_dt, list(linkID = current_link, nextLinkID = one_way_link,log_duration = log_
  }else return(data.table())
  temp_dt
}
stopCluster(cl)
```

```
timeBin_stats <- link_net_list[,
  .(one_way_mean = mean(log_duration, na.rm = TRUE),
  one_way_sd = sd_na_is_0(log_duration),
  one_way_frequency = .N),
  by = .(linkID, nextLinkID, timeBin)]
length_stats <- link_net_list[, .(length = get_mode(length)), by = .(linkID, nextLinkID)]
timeBin_stats <- merge(timeBin_stats, length_stats, by = c("linkID", "nextLinkID"))
global_stats <- link_net_list[,
  .(  one_way_mean = mean(log_duration, na.rm = TRUE),
  one_way_sd = sd_na_is_0(log_duration),
  one_way_frequency = .N,
  length = get_mode(length)),,
  by = .(linkID, nextLinkID)]
global_stats[, timeBin := "Global"]
stats1 <- rbind(timeBin_stats, global_stats)
remove(timeBin_stats,global_stats,length_stats)

reverse_pairs <- link_net_list[, .(linkID = nextLinkID, nextLinkID = linkID, log_duration, timeBin,leng
link_net_two_way <- rbind(link_net_list, reverse_pairs)
two_way_stats <- link_net_two_way[,
  .(two_way_mean = mean(log_duration, na.rm = TRUE),
  two_way_sd = sd_na_is_0(log_duration),
  two_way_frequency = .N),
```

```
  by = .(linkID, nextLinkID, timeBin)]
length_stats <- link_net_two_way[, .(length = get_mode(length)), by = .(linkID, nextLinkID)]
two_way_stats <- merge(two_way_stats, length_stats, by = c("linkID", "nextLinkID"))
global_stats <- link_net_list[,
  .(  two_way_mean = mean(log_duration, na.rm = TRUE),
  two_way_sd = sd_na_is_0(log_duration),
  two_way_frequency = .N,
  length = get_mode(length)),,
  by = .(linkID, nextLinkID)]
global_stats[, timeBin := "Global"]
two_way_stats<-rbind(two_way_stats, global_stats)
net_stat <- merge(
  stats1,
  two_way_stats,
  by = c("linkID", "nextLinkID", "timeBin"),
  all = TRUE
)
net_stat[is.na(one_way_frequency), one_way_frequency := 0]
remove(reverse_pairs,link_net_two_way,two_way_stats,stats1,global_stats,length_stats)
net_stat[, length.x := NULL]
setnames(net_stat, "length.y", "length")
fwrite(net_stat,"data/net_stat.csv")
#fwrite(sampled,"data/sampled.csv")
```

```
edges <- unique(sampled, by = c("linkID", "nextLinkID"))

g <- graph_from_data_frame(edges, directed = TRUE)
tidy_g <- as_tbl_graph(g)

edge_alpha <- 1
filtered_trips <- trips[, if (all(trip %in% id)) .SD, by = trip]
start_nodes <- as.character(filtered_trips[, .( linkId[1]), by = trip]$V1)
end_nodes <- as.character(filtered_trips[, .( linkId[length(linkId)]), by = trip]$V1)
junction_nodes <- V(g)[degree(g, mode = "out") > 1 | degree(g, mode = "in") > 1]$name
node_label <- ifelse(V(g)$name %in% c(junction_nodes,end_nodes,start_nodes), V(g)$name, NA)
length(end_nodes)
```

```
## [1] 20
```

```
paths <- list()
for (i in 1:length(start_nodes)) {
    paths <- c(paths, all_simple_paths(g, from = start_nodes[i], to = end_nodes[i]))
}
```

```
shorten_segment <- function(segment) {
  l<-length(segment)
  if (l > 6) {
    new_length <- 5
    segment <- c(segment[1:new_length],segment[l])
  }
  return(segment)
}
```

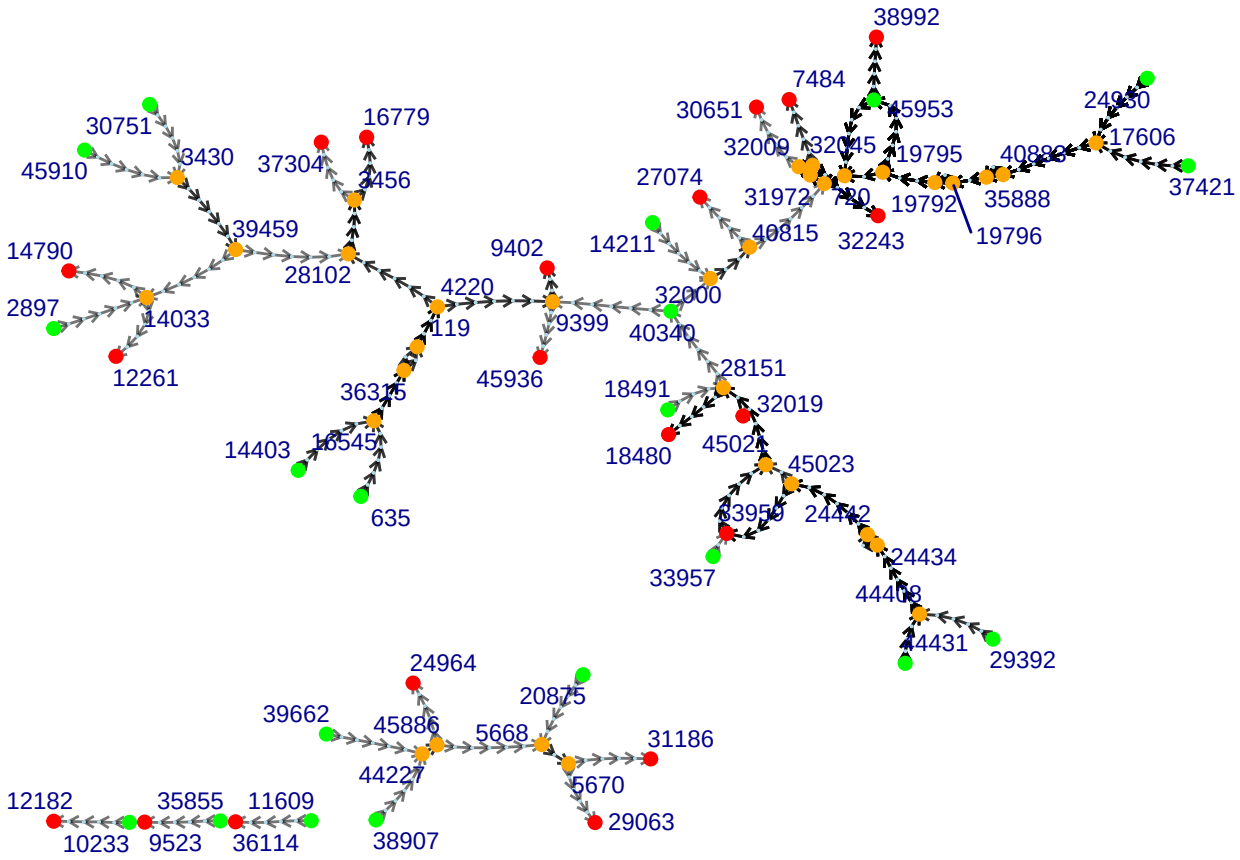
```
segmented_paths <- lapply(paths, function(path) {
  junctions_in_path <- intersect(names(path), junction_nodes)
  if (length(junctions_in_path) == 0) {
    return(shorten_segment(path))
  }
  segments <- list()
  start_index <- 1
  for (junction in junctions_in_path) {
    end_index <- which(names(path) == junction)
    segment <- path[start_index:(end_index-1)]
    segments <- c(segments, list(shorten_segment(segment)))
    start_index <- end_index
  }
  last_segment <- path[start_index:length(path)]
  segments <- c(segments, list(shorten_segment(last_segment)))
  return(unlist(segments))
})
new_edges <- do.call(rbind, lapply(segmented_paths, function(path) {
  path_names <- names(path)
  from <- path_names[-length(path_names)]
  to <- path_names[-1]
  data.frame(from = from, to = to)
}))

g <- graph_from_data_frame(new_edges, directed = TRUE)
tidy_g <- as_tbl_graph(g)
node_label <- ifelse(V(g)$name %in% c(junction_nodes,end_nodes,start_nodes), V(g)$name, NA)
```

```
p1<-ggraph(tidy_g, layout = "stress") +
  geom_edge_link(
    aes(alpha = edge_alpha),
    arrow = arrow(length = unit(1.5, "mm")),
    edge_color = "black"
  ) +
  geom_node_point(
  aes(
    color = ifelse(name %in% start_nodes, "Start",
                   ifelse(name %in% end_nodes, "End",
                          ifelse(name %in% junction_nodes, "Junction", "Normal"))),
    size = ifelse(name %in% start_nodes, "Start",
                  ifelse(name %in% end_nodes, "End",
                         ifelse(name %in% junction_nodes, "Junction", "Normal")))
  )
)+
geom_node_text(aes(label = node_label), size = 3, color = "darkblue", repel = TRUE, na.rm = TRUE) +
  scale_size_manual(values = c("Start" = 2, "End" = 2, "Junction" = 2, "Normal" = 0.01))+
  scale_color_manual(values = c("Start" = "green", "End" = "red", "Junction" = "orange", "Normal" = "lig
  ) +
  theme_void() +
  theme(legend.position = "none")
p1
```

```
## Warning: ggrepel: 1 unlabeled data points (too many overlaps). Consider
```

```
## increasing max.overlaps
```




```
#getwd()
#ggsave("plot/R_network.jpg",p1)
#fwrite(sampled,"data/sampled.csv")
```