On-Demand

Mingze Li 300137754

2025-02-04

```
library(mvtnorm)
```

```
## Warning: package 'mvtnorm' was built under R version 4.3.3
```

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

```
## Warning: package 'data.table' was built under R version 4.3.3
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:data.table':
##
##     between, first, last

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
library(traveltimeHMM)
```

```
##
## Attaching package: 'traveltimeHMM'

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

```r
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
library(patchwork)
```

```r
log_no_0 <- function(x) {
  l <- length(x)
  result <- c()
  for (i in 1:l) {
    if (x[i] == 0) {
      result <- c(result, 0)
    } else {
      result <- c(result, log(x[i]))
    }
  }
  result
}
sd_na_is_0 <- function(x) {
  if (length(x) >= 2) {
    return(sd(x, na.rm = T))
  } else {
    return(0)
  }
}
```

Sample data

```r
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[, duration_secs := as.numeric(difftime(shift(time, type = "lead"), time, units = "secs")), by = ti
trips[, log_duration := log(duration_secs)]
trips <- na.omit(trips)
timeBin_x_edges <- trips[,
  .(
    mean = mean(log_duration, na.rm = TRUE),
    sd = sd_na_is_0(log_duration),
    frequency = .N
  ),
```

```
  by = .(linkId, timeBin)
]
timeBin_x_edges[, edge := 1:.N]
# sample 1000 trips
set.seed(1234)
id <- sample(unique(trips$trip), 1000)
set.seed(NULL)
sampled_trips <- trips[trip %in% id, .(trip, linkId, timeBin, duration_secs, logspeed, length, time)]
sampled_trips <- merge(sampled_trips, timeBin_x_edges, by = c("linkId", "timeBin"), all.x = TRUE)
sampled_time <- sampled_trips[, .(sum(duration_secs, rm.na = T)), by = "trip"]
names(sampled_time)[2] <- "sampled_time"
sampled_trips <- data.frame(sampled_trips)
```

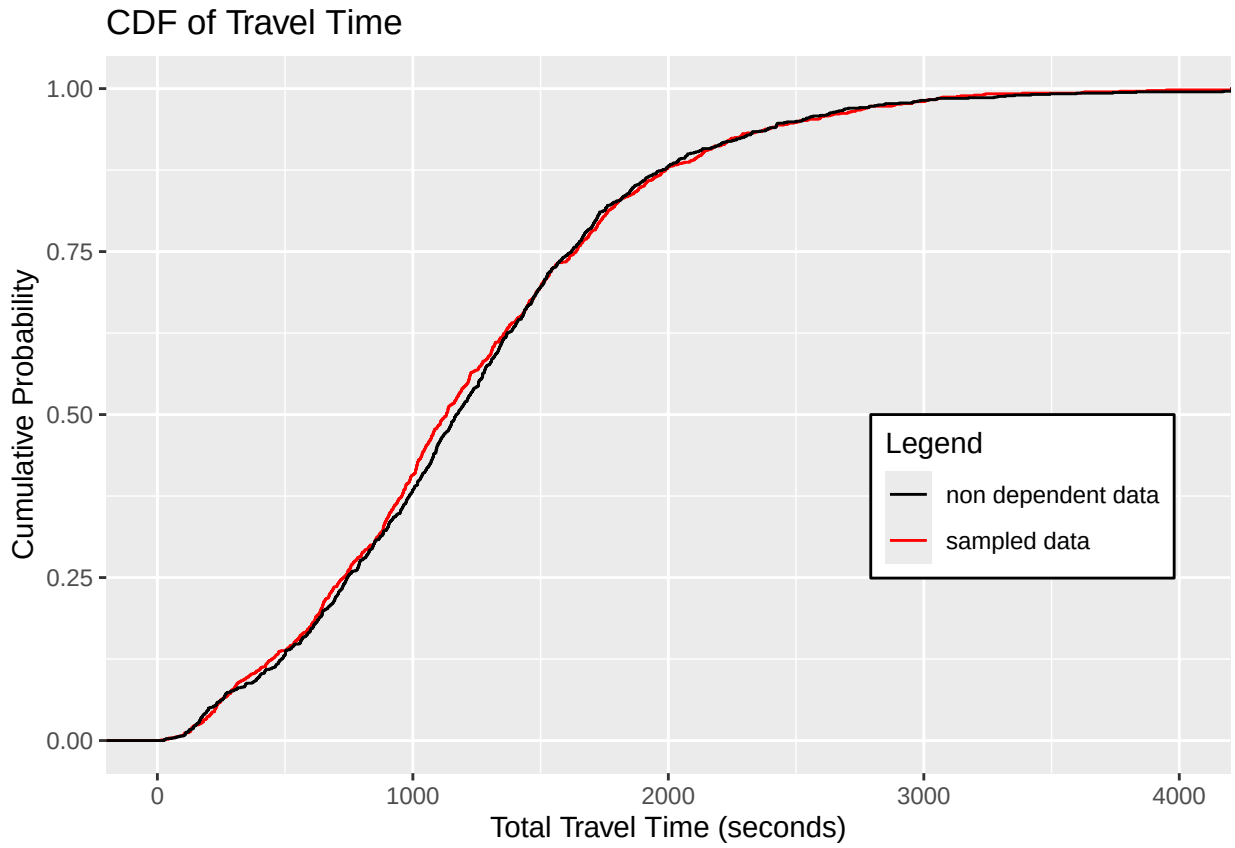
Non-dependence model

```
non_dependent_simulator <- function(edges, rho = 0.31) {
  l <- length(edges)
  U <- runif(l)
  mu <- (sampled_trips[match(edges, sampled_trips$edge), 8])
  sigma <- (sampled_trips[match(edges, sampled_trips$edge), 9])
  sum(exp(mu + sigma * qnorm(U)))
}
non_dependent_time <- sampled_trips %>%
  group_by(trip) %>%
  summarise(simulated_time = non_dependent_simulator(edge))
travel_time <- data.frame(sampled_time)
travel_time$non_dependent_time <- non_dependent_time$simulated_time
```

```
plot1 <- ggplot(travel_time) +
  stat_ecdf(aes(x = sampled_time, color = "sampled data")) +
  stat_ecdf(aes(x = non_dependent_time, color = "non dependent data")) +
  labs(title = "CDF of Travel Time", x = "Total Travel Time (seconds)", y = "Cumulative Probability") +
  coord_cartesian(xlim = c(0, 4000), ylim = c(0, 1)) +
  scale_color_manual(
    name = "Legend",
    values = c(
      "sampled data" = "red",
      "dependent data" = "black",
      "non dependent data" = "black",
      "first order data" = "black",
      "second order data" = "black",
      "population data" = "black",
      "HMM model simulation" = "black"
    )
  ) +
  theme(
    legend.position = c(0.95, 0.5),
    legend.justification = c(1, 1),
    legend.text.align = 0,
    legend.background = element_rect(color = "black", fill = "white")
  )
```

```
## Warning: The `legend.text.align` argument of `theme()` is deprecated as of ggplot2
## 3.5.0.
## i Please use theme(legend.text = element_text(hjust)) instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## 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.
```

```
plot1
```

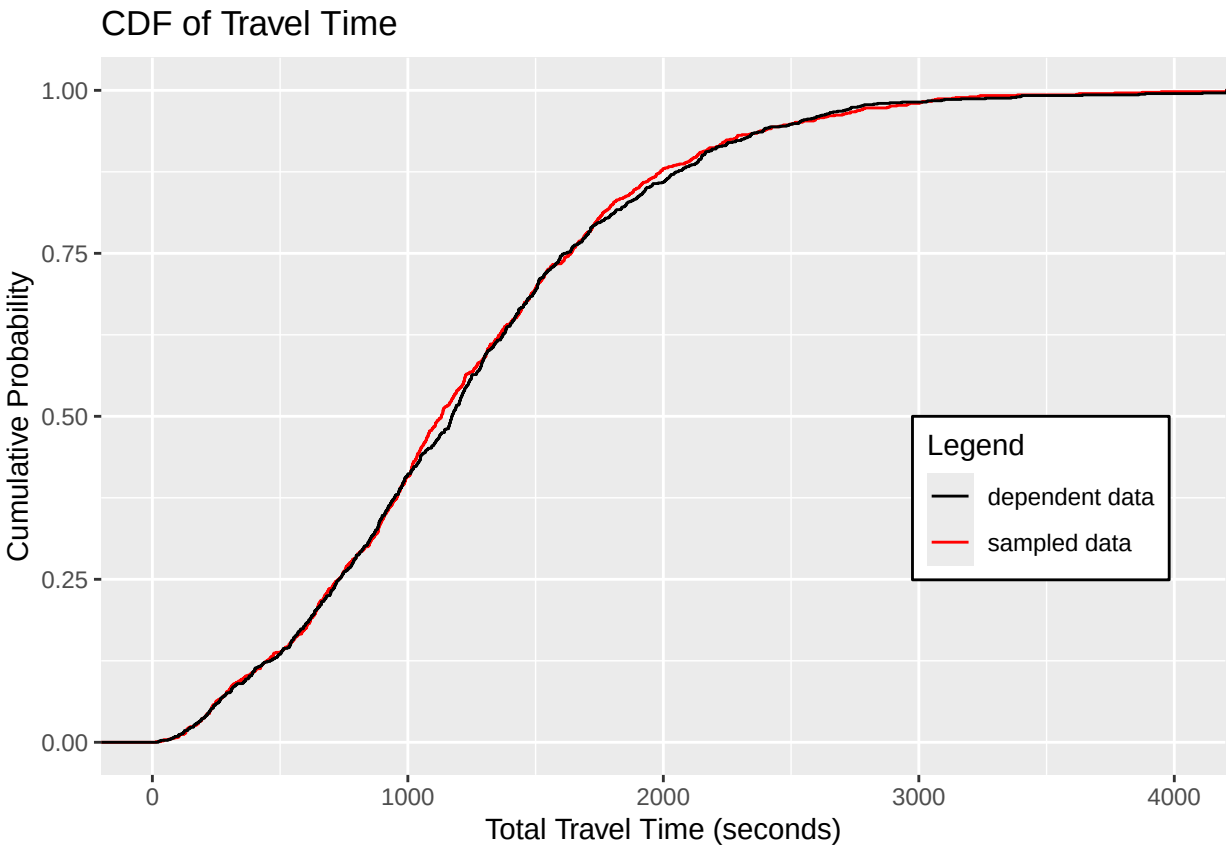


The Full Dependent Model

```
dependent_uniform <- function(n, rho = 0.31) {
  S <- diag(n)
  for (i in 1:n) {
    for (j in 2:n) {
      S[i, j] <- rho^(abs(i - j))
    }
  }
  S <- S + t(S)
```

```
  diag(S) <- 1
  St <- 2 * sin(S * pi / 6) # must be positive definite
  U <- c(pnorm(rmvnorm(1, sigma = St)))
  U
}
time_simulator <- function(edges, rho = 0.31) {
  l <- length(edges)
  if (l > 1) {
    U <- dependent_uniform(l, rho)
  } else {
    U <- runif(1)
  }
  mu <- (sampled_trips[match(edges, sampled_trips$edge), 8])
  sigma <- (sampled_trips[match(edges, sampled_trips$edge), 9])
  sum(exp(mu + sigma * qnorm(U)))
}
simulated_time <- sampled_trips %>%
  group_by(trip) %>%
  summarise(simulated_time = time_simulator(edge))
travel_time$dependent <- simulated_time$simulated_time
```

```
plot2 <- ggplot(travel_time) +
  stat_ecdf(aes(x = sampled_time, color = "sampled data")) +
  stat_ecdf(aes(x = dependent, color = "dependent data")) +
  labs(title = "CDF of Travel Time", x = "Total Travel Time (seconds)", y = "Cumulative Probability") +
  coord_cartesian(xlim = c(0, 4000), ylim = c(0, 1)) +
  scale_color_manual(
    name = "Legend",
    values = c(
      "sampled data" = "red",
      "dependent data" = "black",
      "non dependent data" = "black",
      "first order data" = "black",
      "second order data" = "black",
      "population data" = "black",
      "HMM model simulation" = "black"
    )
  ) +
  theme(
    legend.position = c(0.95, 0.5),
    legend.justification = c(1, 1),
    legend.text.align = 0,
    legend.background = element_rect(color = "black", fill = "white")
  )
plot2
```

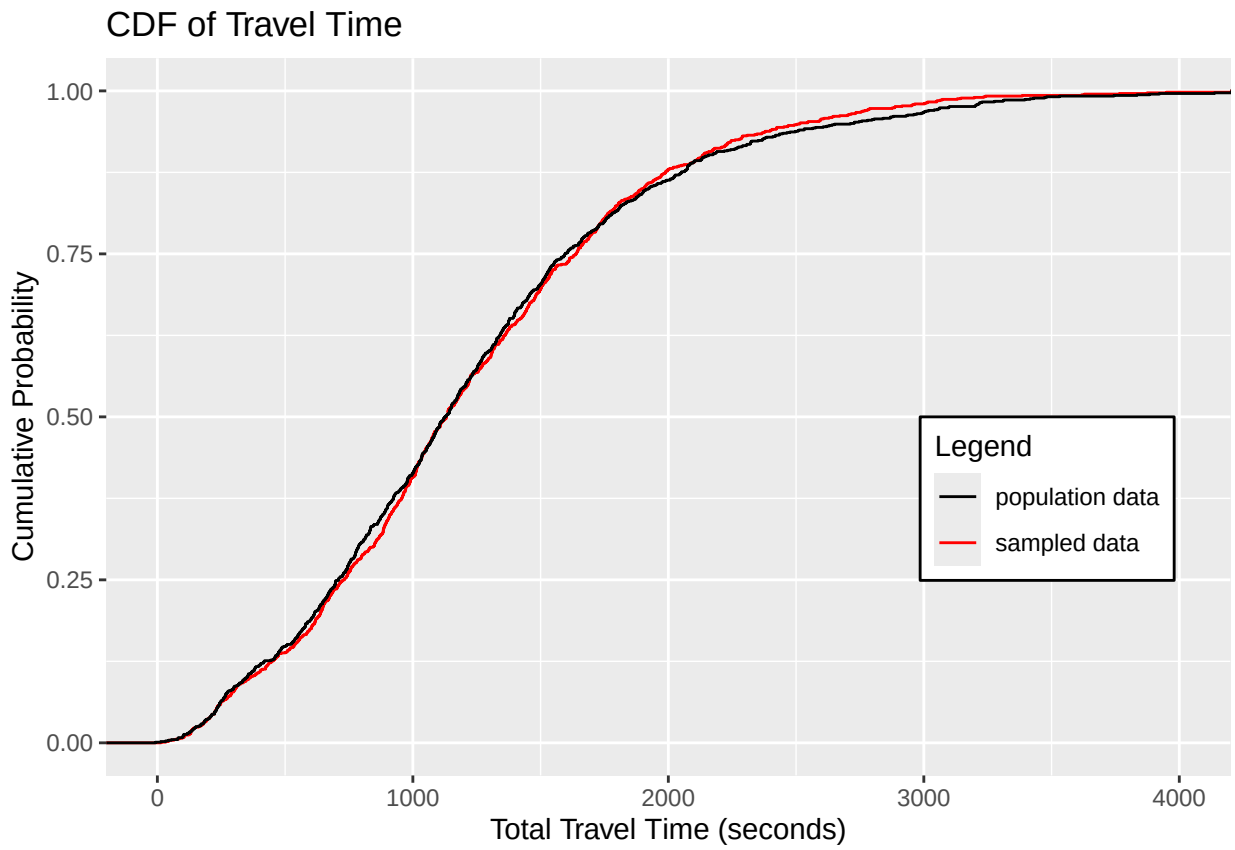


Population model

```
population_simulator <- function(duration_secs) {
  l <- length(duration_secs) + 1
  Z <- rnorm(l, 0, sqrt(1))
  mu <- mean(duration_secs)
  sigma <- sd_na_is_0(duration_secs)
  t <- numeric(l)
  for (i in 2:l) {
    t[i] <- t[i - 1] + mu + sigma * Z[i]
  }
  t[l]
}
population_time <- sampled_trips %>%
  group_by(trip) %>%
  summarise(simulated_time = population_simulator(duration_secs))
travel_time$population_time <- population_time$simulated_time
```

```
plot3 <- ggplot(travel_time) +
  stat_ecdf(aes(x = sampled_time, color = "sampled data")) +
  stat_ecdf(aes(x = population_time, color = "population data")) +
  labs(title = "CDF of Travel Time", x = "Total Travel Time (seconds)", y = "Cumulative Probability") +
  coord_cartesian(xlim = c(0, 4000), ylim = c(0, 1)) +
```

```
  scale_color_manual(
    name = "Legend",
    values = c(
      "sampled data" = "red",
      "dependent data" = "black",
      "non dependent data" = "black",
      "first order data" = "black",
      "second order data" = "black",
      "population data" = "black",
      "HMM model simulation" = "black"
    )
  ) +
  theme(
    legend.position = c(0.95, 0.5),
    legend.justification = c(1, 1),
    legend.text.align = 0,
    legend.background = element_rect(color = "black", fill = "white")
  )
plot3
```



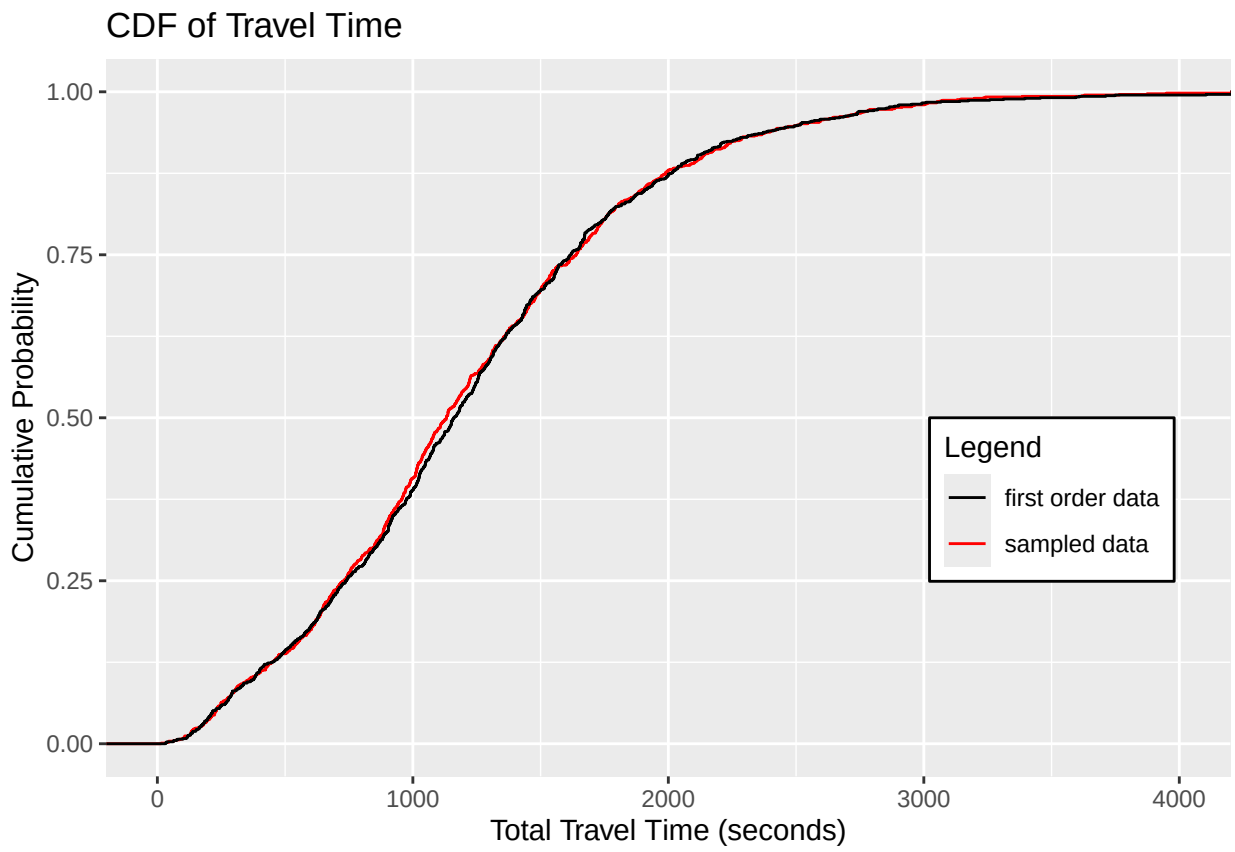
First Order Model

```
first_order <- function(n, rho = 0.31) {
  S <- diag(n)
  if (n > 1) {
    if (n > 2) {
```

```
      for (i in 2:(n - 1)) {
        for (j in (i - 1):(i + 1)) {
          S[i, j] <- 2 * rho^(abs(1))
        }
      }
    }
    S[n, n - 1] <- 2 * rho
    S[1, 2] <- rho
    S[2, 1] <- rho
    diag(S) <- 1
    eigen_values <- eigen(S, symmetric = TRUE)$values
    if (!all(eigen_values >= 0)) {
      S <- as.matrix(Matrix::nearPD(S, cor = TRUE)$mat)
    }
    U <- c(pnorm(rmvnorm(1, sigma = S)))
  } else {
    U <- runif(1)
  }
  U
}
first_order_simulator <- function(edges, rho = 0.31) {
  l <- length(edges)
  U <- first_order(l)
  mu <- (sampled_trips[match(edges, sampled_trips$edge), 8])
  sigma <- (sampled_trips[match(edges, sampled_trips$edge), 9])
  sum(exp(mu + sigma * qnorm(U)))
}
first_order_time <- sampled_trips %>%
  group_by(trip) %>%
  summarise(simulated_time = first_order_simulator(edge))
travel_time$first_order_time <- first_order_time$simulated_time
```

```
plot4 <- ggplot(travel_time) +
  stat_ecdf(aes(x = sampled_time, color = "sampled data")) +
  stat_ecdf(aes(x = first_order_time, color = "first order data")) +
  labs(title = "CDF of Travel Time", x = "Total Travel Time (seconds)", y = "Cumulative Probability") +
  coord_cartesian(xlim = c(0, 4000), ylim = c(0, 1)) +
  scale_color_manual(
    name = "Legend",
    values = c(
      "sampled data" = "red",
      "dependent data" = "black",
      "non dependent data" = "black",
      "first order data" = "black",
      "second order data" = "black",
      "population data" = "black",
      "HMM model simulation" = "black"
    )
  ) +
  theme(
    legend.position = c(0.95, 0.5),
    legend.justification = c(1, 1),
    legend.text.align = 0,
```

```
    legend.background = element_rect(color = "black", fill = "white")
  )
plot4
```

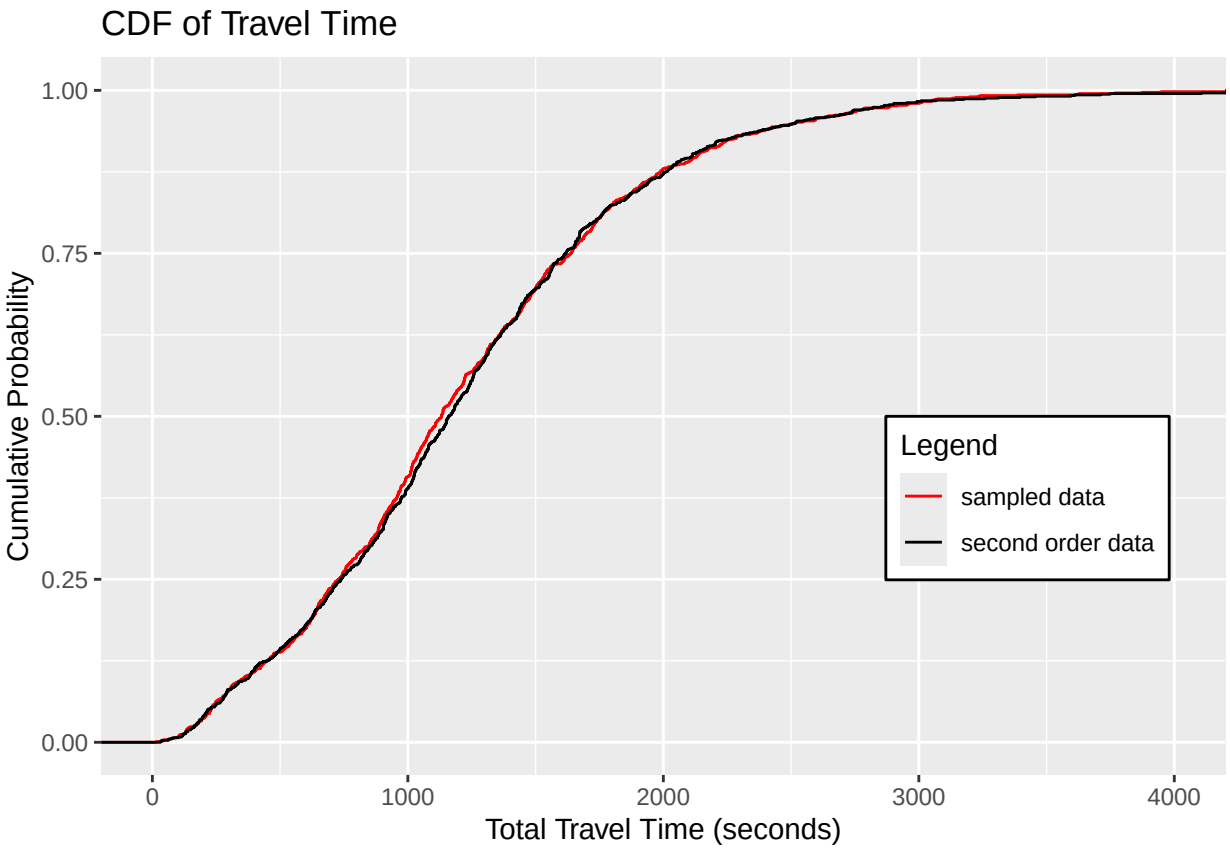


Second Order Model

```
second_order <- function(n, rho = 0.31) {
  S <- diag(n)
  if (n > 2) {
    for (i in 1:n) {
      if (i - 2 > 0) S[i, (i - 2)] <- 2 * rho
      if (i + 2 <= n) S[i, (i + 2)] <- 2 * rho
    }
    S[1, 3] <- rho
    S[3, 1] <- rho
    diag(S) <- 1
    eigen_values <- eigen(S, symmetric = TRUE)$values
    if (!all(eigen_values >= 0)) {
      S <- as.matrix(Matrix::nearPD(S, cor = TRUE)$mat)
    }
    U <- c(pnorm(rmvnorm(1, sigma = S)))
  } else {
    U <- runif(n)
  }
  U
}
```

```r
second_order_simulator <- function(edges, rho = 0.31) {
  l <- length(edges)
  U <- second_order(l)
  mu <- (sampled_trips[match(edges, sampled_trips$edge), 8])
  sigma <- (sampled_trips[match(edges, sampled_trips$edge), 9])
  sum(exp(mu + sigma * qnorm(U)))
}
second_order_time <- sampled_trips %>%
  group_by(trip) %>%
  summarise(simulated_time = second_order_simulator(edge))
travel_time$second_order_time <- first_order_time$simulated_time
```

```r
plot5 <- ggplot(travel_time) +
  stat_ecdf(aes(x = sampled_time, color = "sampled data")) +
  stat_ecdf(aes(x = second_order_time, color = "second order data")) +
  labs(title = "CDF of Travel Time", x = "Total Travel Time (seconds)", y = "Cumulative Probability") +
  coord_cartesian(xlim = c(0, 4000), ylim = c(0, 1)) +
  scale_color_manual(
    name = "Legend",
    values = c(
      "sampled data" = "red",
      "dependent data" = "black",
      "non dependent data" = "black",
      "first order data" = "black",
      "second order data" = "black",
      "population data" = "black",
      "HMM model simulation" = "black"
    )
  ) +
  theme(
    legend.position = c(0.95, 0.5),
    legend.justification = c(1, 1),
    legend.text.align = 0,
    legend.background = element_rect(color = "black", fill = "white")
  )
plot5
```




```
tripdata <- data.frame(
  logspeed = sampled_trips$logspeed,
  tripID = sampled_trips$trip,
  timeBin = sampled_trips$timeBin,
  linkID = sampled_trips$linkId,
  length = sampled_trips$length,
  time = (sampled_trips$time),
  traveltime = sampled_trips$duration_secs
)
tripdata <- tripdata %>%
  group_by(tripID) %>%
  arrange(time, .by_group = TRUE)
unique(tripdata$timeBin)
```

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

```
tripdata$time <- as.POSIXct(tripdata$time, format = "%Y-%m-%dT%H:%M:%OSZ")
tripdata$timeBin <- time_bins_readable(tripdata$time)
unique(tripdata$timeBin)
```

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

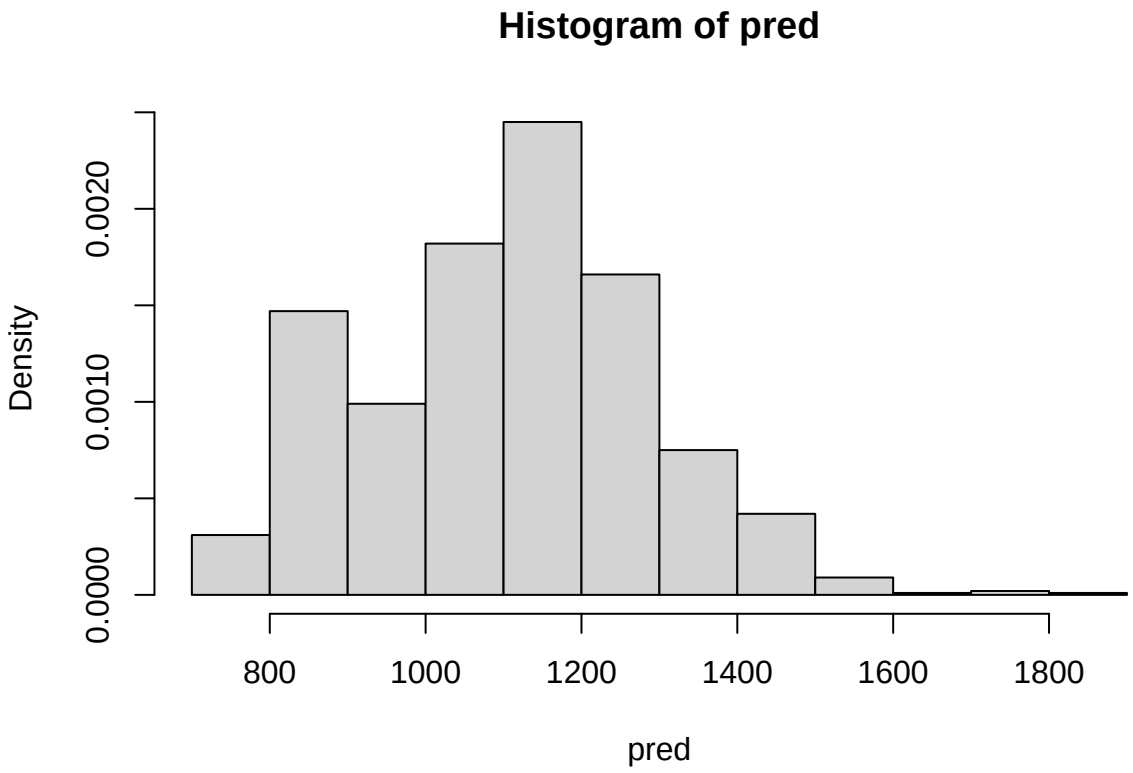
```
fit <- traveltimeHMM(
  data = tripdata,
```

```
  nQ = 2, max.it = 20, model = "HMM"
)
```

```
## max.speed is not specified, setting at default value: 130 km/h

## Warning: Many observations are higher than speed limit (130km/h)!, about 0.9%
## of observations.  It is advised to remove these observations.

## Model HMM with 1000 trips over 16890 roads and 5 time bins...

## Expected completion of 20 iterations in 99 secs

## Reached maximum number of iterations
```

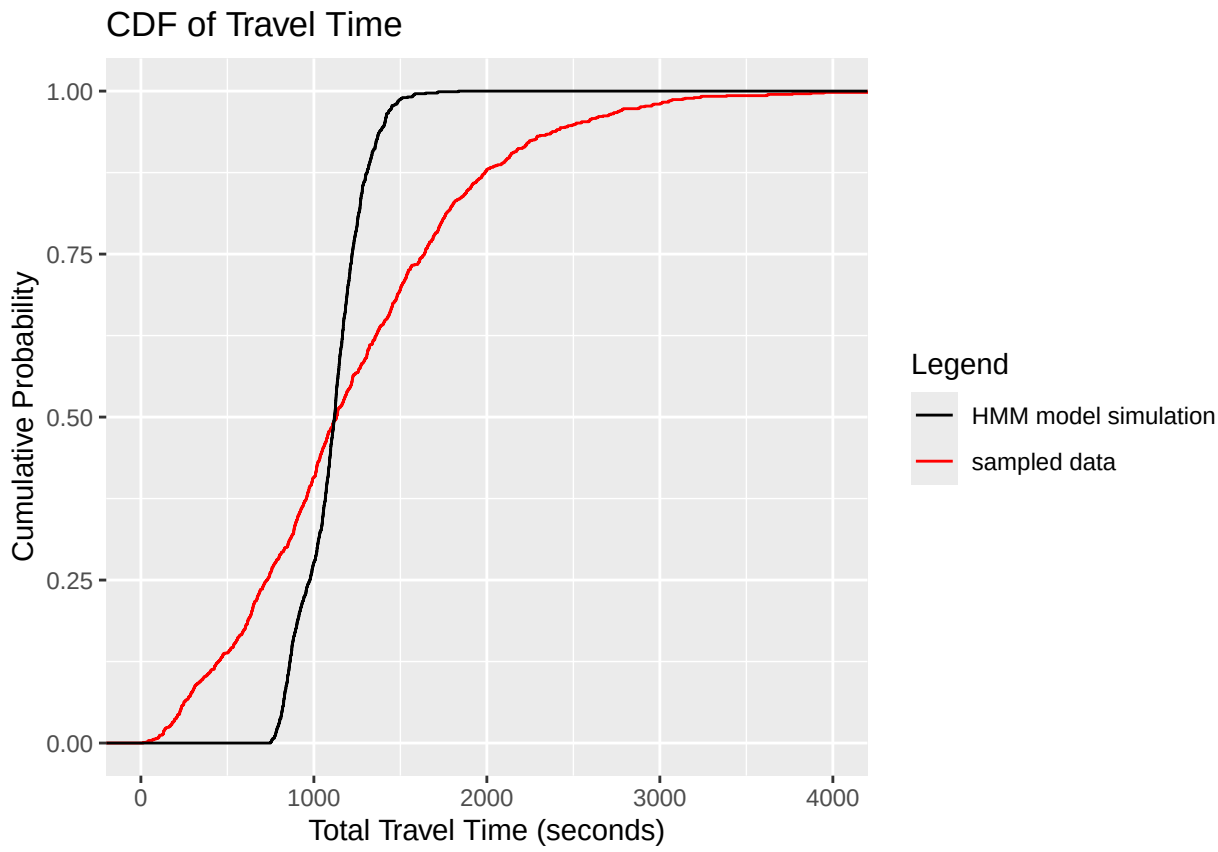
```
single_trip <- subset(tripdata, tripID == unique(tripdata$tripID)[1])
```

```
pred <- predict(
  object = fit,
  tripdata = single_trip,
  starttime = single_trip$time[1],
  n = 1000
)
hist(pred, freq = FALSE)
```



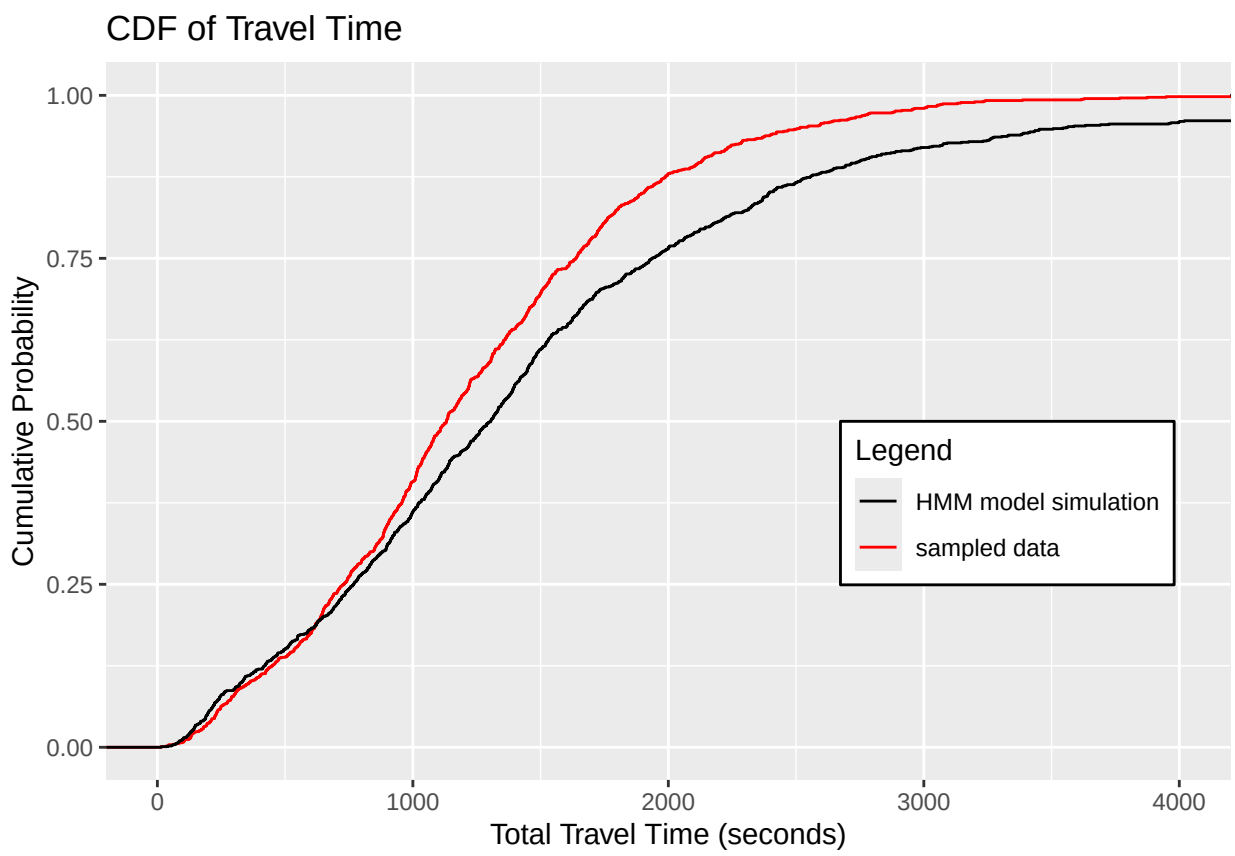
```
travel_time$HMM <- pred
```

```
ggplot(travel_time) +
  stat_ecdf(aes(x = sampled_time, color = "sampled data")) +
  stat_ecdf(aes(x = pred, color = "HMM model simulation")) +
  labs(title = "CDF of Travel Time", x = "Total Travel Time (seconds)", y = "Cumulative Probability") +
  coord_cartesian(xlim = c(0, 4000), ylim = c(0, 1)) +
  scale_color_manual(name = "Legend", values = c("black", "red"))
```




```
starttime <- tripdata %>%
  group_by(tripID) %>%
  summarise(first(time))
pred <- c()
for (i in 1:1000) {
  single_trip <- subset(tripdata, tripID == unique(tripdata$tripID)[i])
  newpred <- predict(
    object = fit,
    tripdata = single_trip,
    starttime = starttime$`first(time)`[i],
    n = 1
  )
  pred <- c(pred, newpred)
}
travel_time$HMM <- pred
```

```r
plot6 <- ggplot(travel_time) +
  stat_ecdf(aes(x = sampled_time, color = "sampled data")) +
  stat_ecdf(aes(x = pred, color = "HMM model simulation")) +
  labs(title = "CDF of Travel Time", x = "Total Travel Time (seconds)", y = "Cumulative Probability") +
  coord_cartesian(xlim = c(0, 4000), ylim = c(0, 1)) +
  scale_color_manual(
    name = "Legend",
    values = c(
      "sampled data" = "red",
      "dependent data" = "black",
      "non dependent data" = "black",
      "first order data" = "black",
      "second order data" = "black",
      "population data" = "black",
      "HMM model simulation" = "black"
    )
  ) +
  theme(
    legend.position = c(0.95, 0.5),
    legend.justification = c(1, 1),
    legend.text.align = 0,
    legend.background = element_rect(color = "black", fill = "white")
  )
plot6
```



```r
plot_list <- list(plot1, plot2, plot4, plot5, plot3, plot6)
combined_plot <- wrap_plots(plot_list, nrow = 3)


ggsave("plot/R_combined_simulation_plots2.jpg",
  plot = combined_plot,
  device = "jpg",
  width = 21,
  height = 29.7,
  units = "cm",
  dpi = 300
)
```

```r
plot7 <- ggplot(travel_time) +
  stat_ecdf(aes(x = sampled_time, color = "Real Data", linetype = "Real Data")) +
  stat_ecdf(aes(x = non_dependent_time, color = "Non-dependent", linetype = "Non-dependent")) +
  stat_ecdf(aes(x = dependent, color = "Full-dependent", linetype = "Full-dependent")) +
  stat_ecdf(aes(x = first_order_time, color = "1st-order", linetype = "1st-order")) +
  stat_ecdf(aes(x = second_order_time, color = "2nd-order", linetype = "2nd-order")) +
  stat_ecdf(aes(x = population_time, color = "Population", linetype = "Population")) +
  stat_ecdf(aes(x = HMM, color = "HMM", linetype = "HMM")) +
  labs(
    title = "Combined CDF Comparison",
    x = "Total Travel Time (seconds)",
    y = "Cumulative Probability"
  ) +
  coord_cartesian(xlim = c(0, 4000), ylim = c(0, 1)) +
  scale_color_manual(
    name = "Models",
    values = c(
      "Real Data" = "red",
      "Non-dependent" = "#1f77b4",
      "Full-dependent" = "#ff7f0e",
      "1st-order" = "#2ca02c",
      "2nd-order" = "#d19f9f",
      "Population" = "#7c48ac",
      "HMM" = "#7d3f33"
    )
  ) +
  scale_linetype_manual(
    name = "Models",
    values = c(
      "Real Data" = "solid",
      "Non-dependent" = "dashed",
      "Full-dependent" = "dotted",
      "1st-order" = "dotdash",
      "2nd-order" = "longdash",
      "Population" = "twodash",
      "HMM" = "F1"
    )
  ) +
  theme(
    panel.background = element_rect(fill = "white"),
```

```
    panel.grid = element_blank(),
    plot.background = element_rect(fill = "white"),
    legend.position = "bottom",
    legend.key = element_rect(fill = "white")
  )
plot7
```