Untitled

Mingze Li 300137754

2025-01-29

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

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

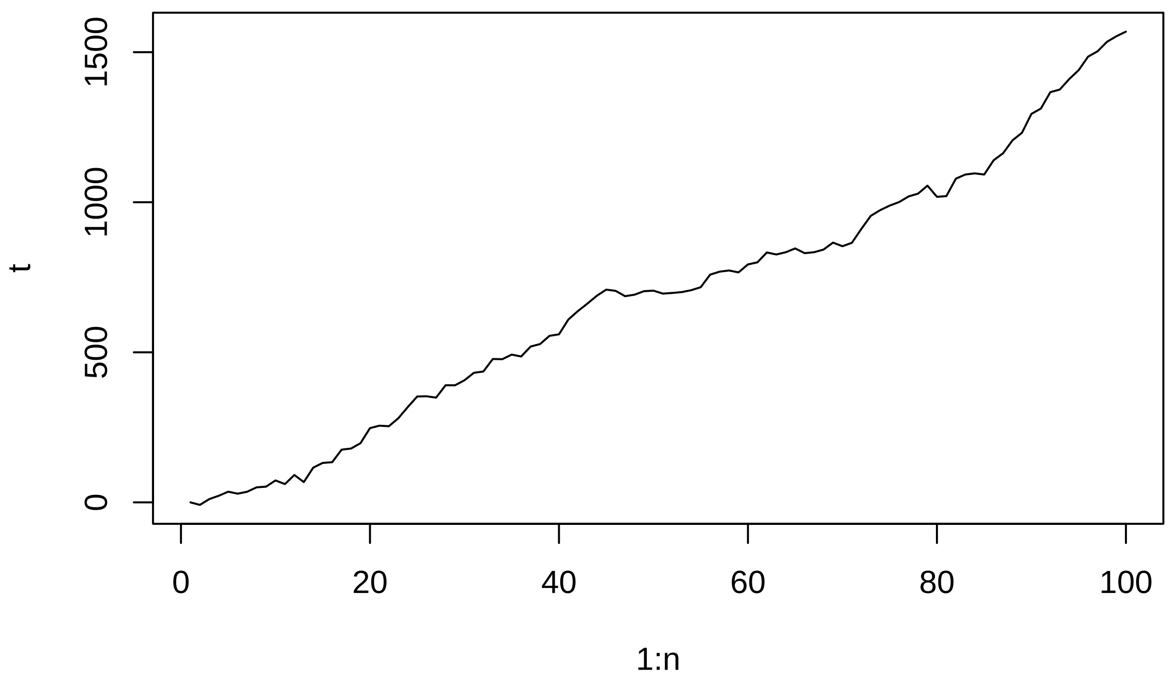
```r
library(dplyr)
```

```
##
##    '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
```

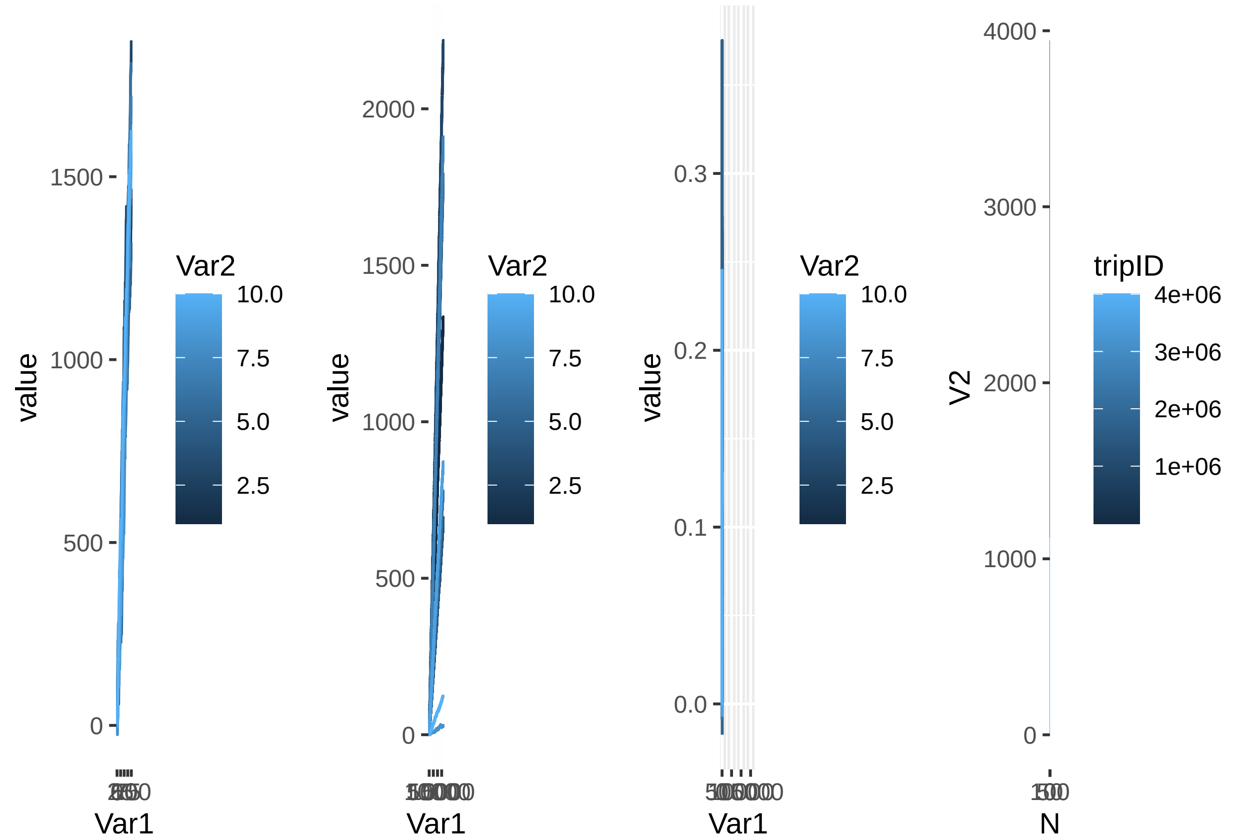
```r
source("sde_sim.R")
```

```
##
##    'reshape2'

## The following objects are masked from 'package:data.table':
##
##     dcast, melt

## Warning in sqrt(t[i - 1]):   NaNs

## Warning in sqrt(t[i - 1]):   NaNs

## Warning in sqrt(t[i - 1]):   NaNs

## Warning in sqrt(t[i - 1]):   NaNs
```

```
## Warning in sqrt(t[i - 1]):  NaNs

## Warning in sqrt(t[i - 1]):  NaNs

## Warning in sqrt(t[i - 1]):  NaNs

## Warning in sqrt(t[i - 1]):  NaNs

## Warning in sqrt(t[i - 1]):  NaNs

## Warning in sqrt(t[i - 1]):  NaNs

## Warning in sqrt(t[i - 1]):  NaNs
```



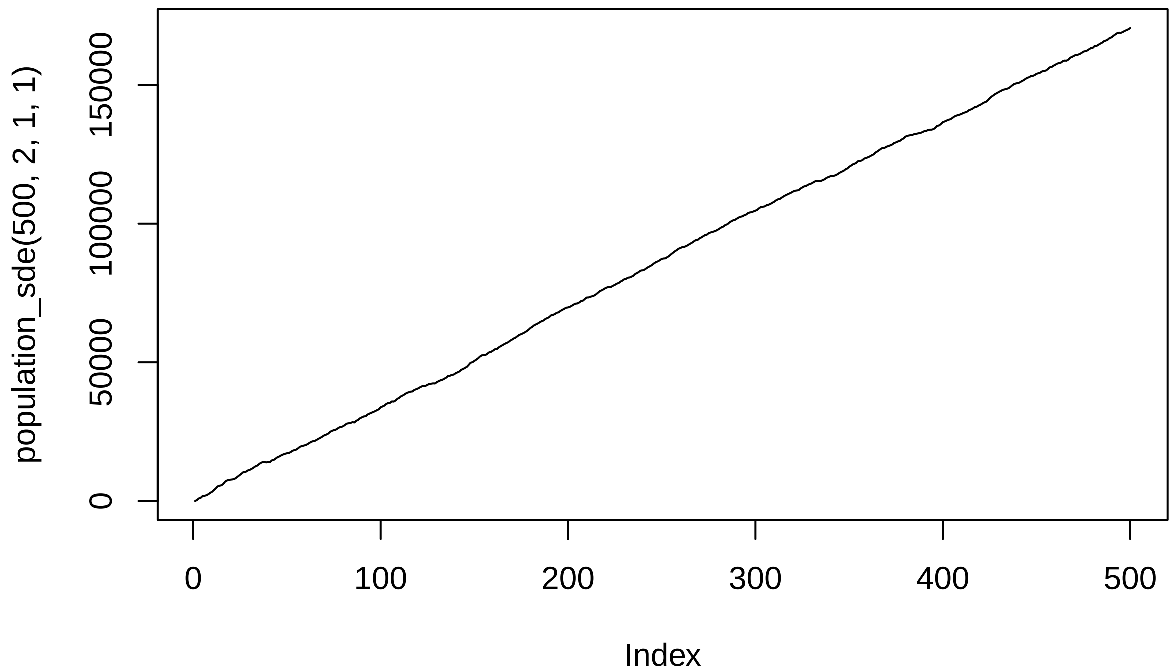

```
## Warning: Removed 169750 rows containing missing values (`geom_line()`).
```



##Study the random trip simulators

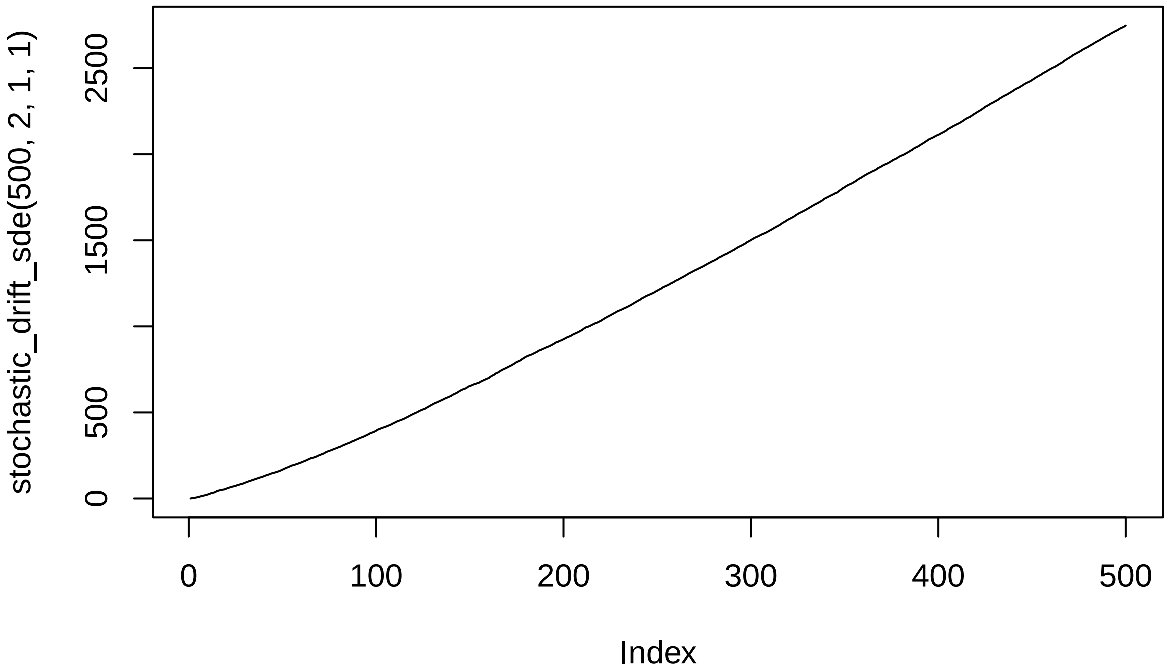
population_sde simulate the arrival time at every edge.The result may looks close to a sraight line points top right. Input : n is # of edges, mu sigma are the global estimated mean and standard error of the speed. Delta is the difference in standard error estimation.

```
plot(population_sde(500,2,1,1),type='l')
```

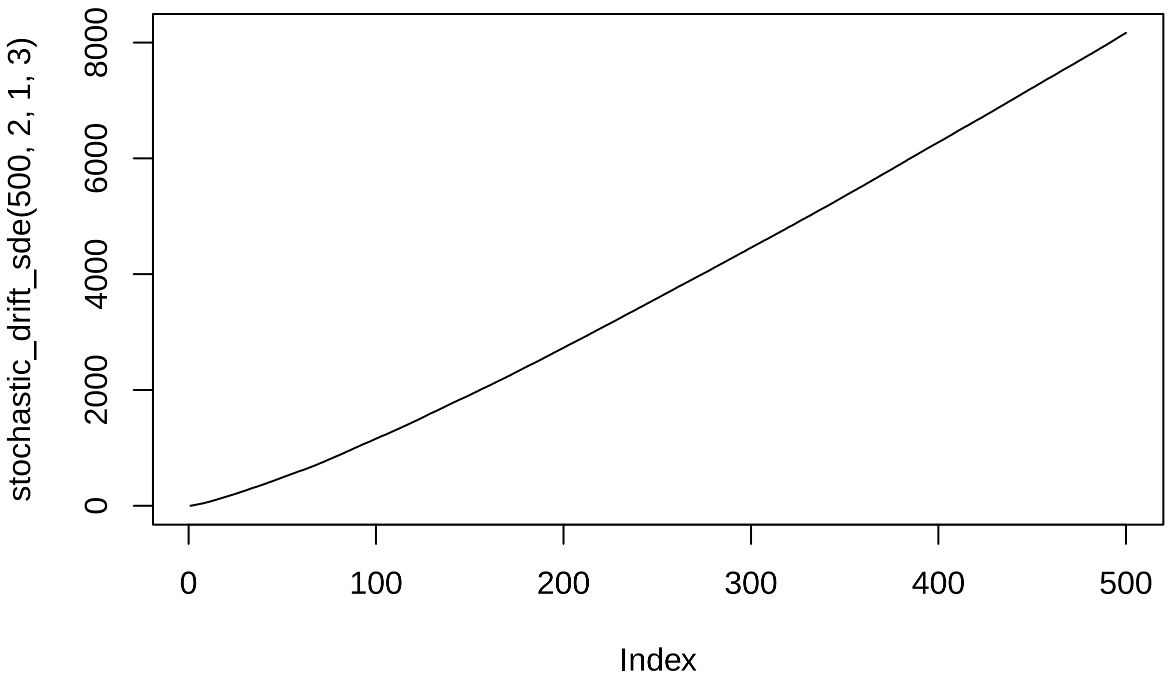


stochastic__drift__sde simulates the arrival time at every edge. When n is large, (1+t[i-1]/(1 + i*delta)) will be more close to 1. the new t will depends less on the previous simulated value t. The direction of the line will be more close to a steady slop.

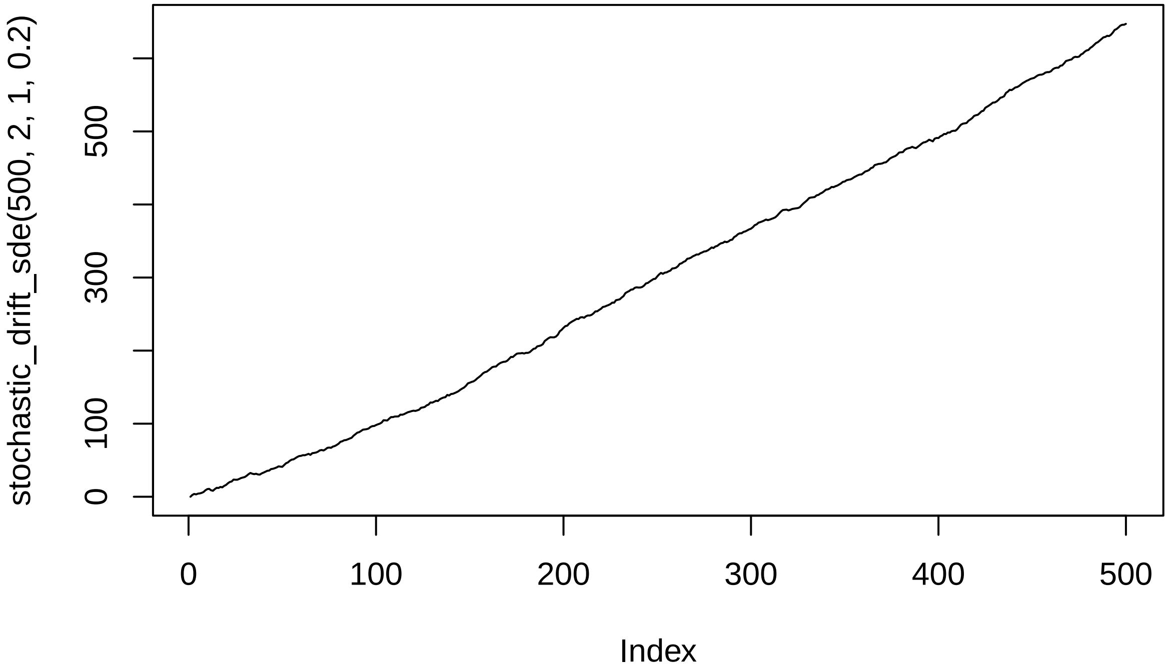
```
plot(stochastic_drift_sde(500,2,1,1),type='l')
```




```
plot(stochastic_drift_sde(500,2,1,3),type='l')
```

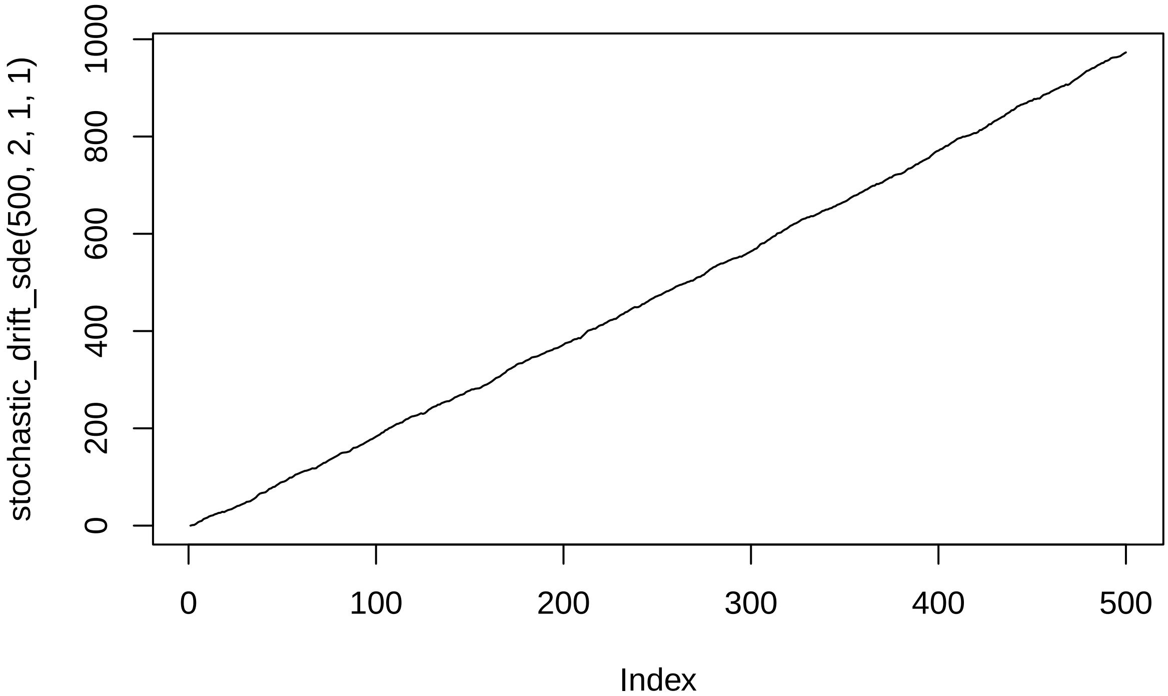



```r
plot(stochastic_drift_sde(500,2,1,0.2),type='l')
```

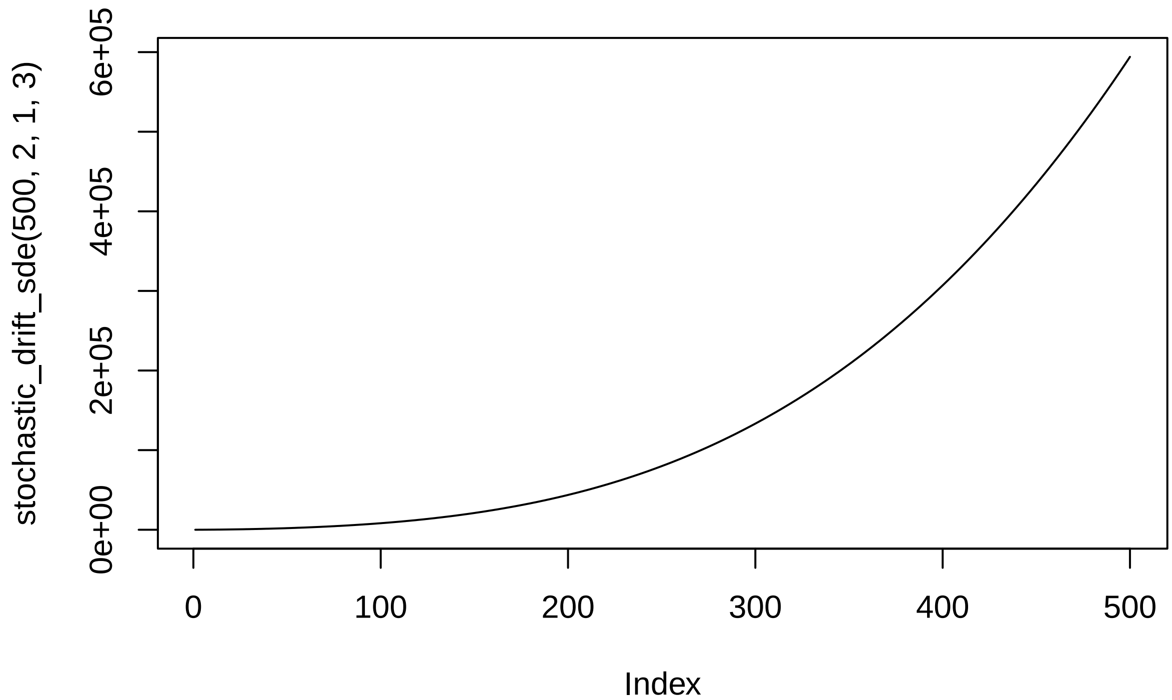


There is another version of stochastic_drift_sde. When n is large, lambda will be more close to 1. the new t will depend on the previous simulated value t. When delta is large, this simulator behave like exponential functions.

```
stochastic_drift_sde <- function(n,
                                 mu=mu,
                                 sigma=sigma,
                                 delta=delta) {
  B = rnorm(n , 0, sqrt(delta))
  t = numeric(n)
  for(i in 2:n) {
    lambda = exp(0.01*i)/(1+exp(0.01*i))
    mu_avrg = mu * (1-lambda) + (t[i-1]/(i-1)) * lambda
    t[i] = t[i-1] + mu_avrg *delta + sigma * B[i]
  }
  t
}
plot(stochastic_drift_sde(500,2,1,1),type='l')
```



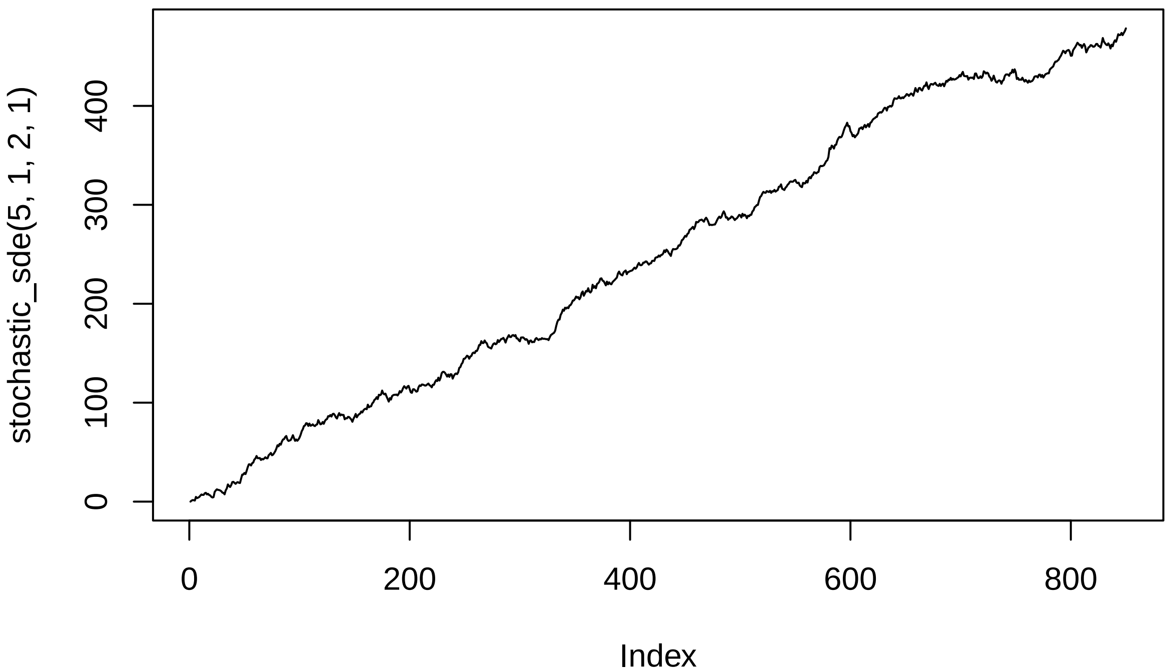

```
plot(stochastic_drift_sde(500,2,1,3),type='l')
```




```
plot(stochastic_drift_sde(500,2,1,0.1),type='l')
```

This simulator will jump to the mu in the second observation, and never become lower than mu. Every edges are expected to increase by half of the mu. Delta does not make any change. The resulting length is multipled by 170, as the avg_seg_length in sde_sim.R is set by 170. I suggest to modify this process slightly on the line "t[i] = max(t[i-1] + (t[i-1])/(i) + sigma * B[i], mu)", the new one ensures there is no backward movement. Also, I added a result vector to make sure the resulting length is the same as n.

```
plot(stochastic_sde(5,1,2,1),type='l')
```

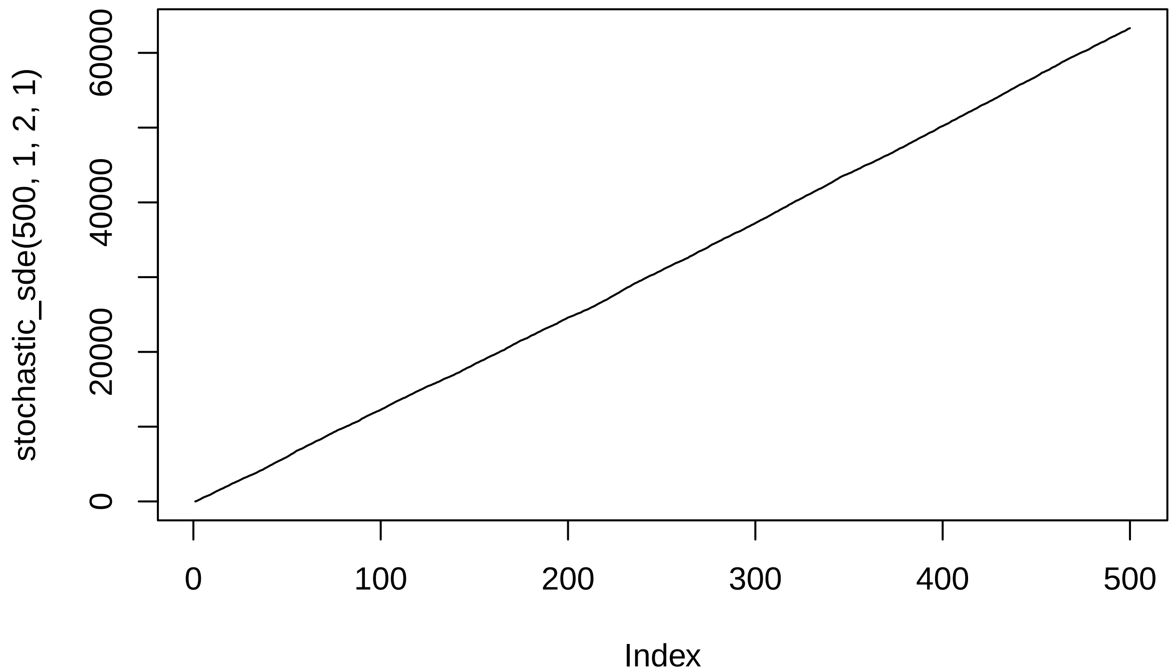



```
length(stochastic_sde(5,2,1,1))
```

```
## [1] 850
```

```
stochastic_sde <- function(n,
                           mu=mu,
                           sigma=sigma,
                           delta=delta) {
  m = floor(avg_seg_length)  * n-1
  mu <- mu*avg_seg_length/2
  B = rnorm(m , 0, 1)
  t = numeric(m)
  result = c(0)
  for(i in 2:m) {
    t[i] = t[i-1] +(t[i-1])/(i) + sigma * B[i]
    if(i%%floor(avg_seg_length)==0&
       result[length(result)]<=t[i]-mu)result<-c(result,t[i])
    else if (i%%floor(avg_seg_length)==0){
      result<-c(result,result[length(result)]+mu)
      t[i]<-result[length(result)]
      }
  }
  result
```

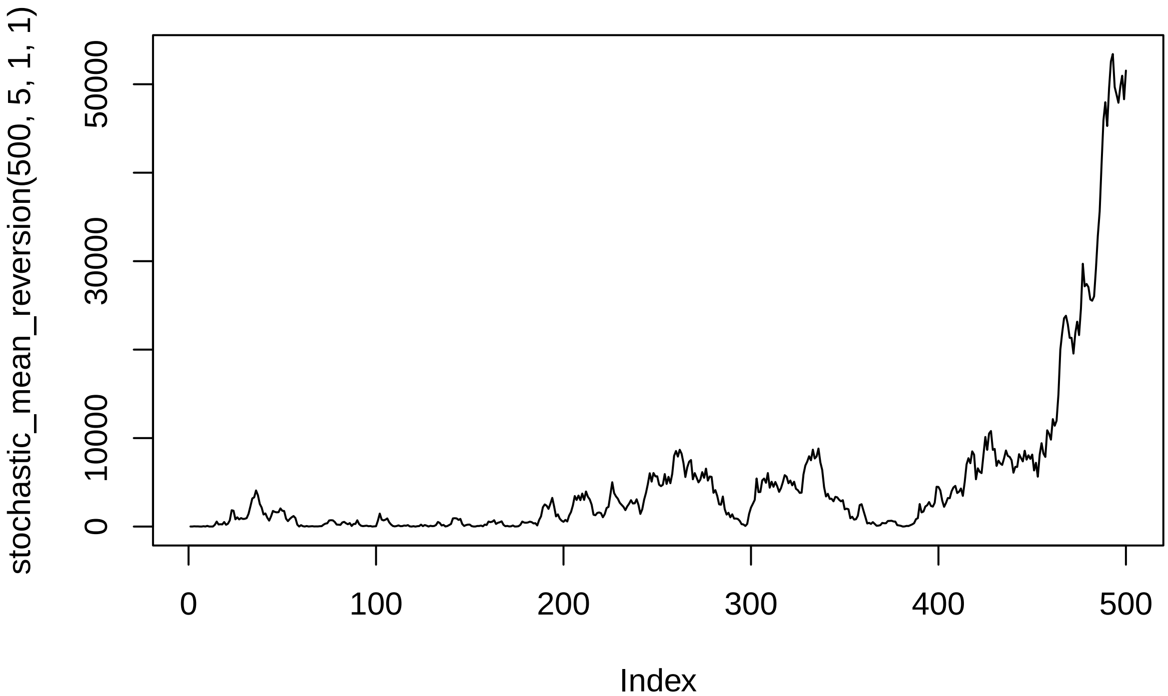
```
}
plot(stochastic_sde(500,1,2,1),type='l')
```



I modified this simulator, as the sqrt function cannot accept negative input so sometime produce NaN. I added a absolute value to make sure it works. The variance increases according to t, meaning no matter where the t is, the probability of t back to small persists. This simulator seems more suitable for financial data.

```
stochastic_mean_reversion <- function(n,
                                      mu=mu,
                                      sigma=sigma,
                                      delta=delta) {
  m = floor(avg_seg_length  * n)
  B = rnorm(m , 0, 1)
  t = numeric(m)
  result = c(mu)
  mu_function<-function(x,y, alpha=0.1) {
    alpha * (x - y)
  }

  for(i in 2:(m-1)) {
    #t[i] = max(t[i-1] + mu_function(mu, t[i-1]/i, 0.05)   + sigma * sqrt(t[i-1]) * B[i], mu)
    t[i] = t[i-1] + mu_function(mu, t[i-1]/i, 0.05)   + sigma * sqrt(abs(t[i-1])) * B[i]
    if(i%%floor(avg_seg_length)==0)result<-c(result,t[i])
  }
  result
```

```r
}
plot(stochastic_mean_reversion(500,5,1,1),type='l')
```

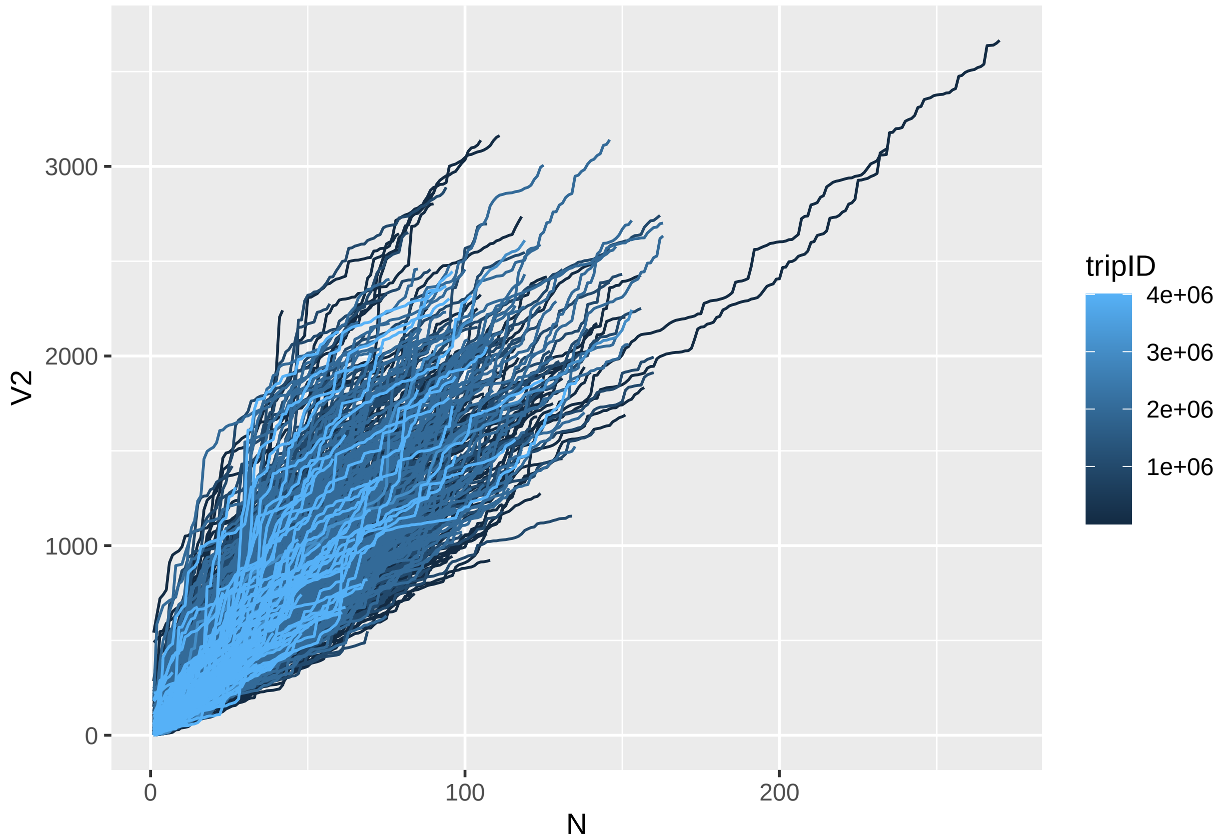


Calculate trip parameters.

First, randomly selected 1000 trips, then calculate the estimated global mean $\hat{\mu}$ and its unconditional variance $\hat{\mathbb{V}}(n^{-1}\tau_\rho)$. Their formulas are given by formula (16) and formula (19) of the paper Predictive Inference of Travel Time on Transportation Networks.

```r
id = sample(unique(trips$tripID),1000)
sampled_trips = trips[tripID %in% id]
a = sampled_trips[, .(N=1:.N, cumsum(duration_secs)),tripID]
groupmean <- sampled_trips[, mean(duration_secs),tripID]$V1
muhat = mean(groupmean)
vhat = var(groupmean)
trip_length <- sampled_trips[, length(duration_secs),tripID]$V1
```

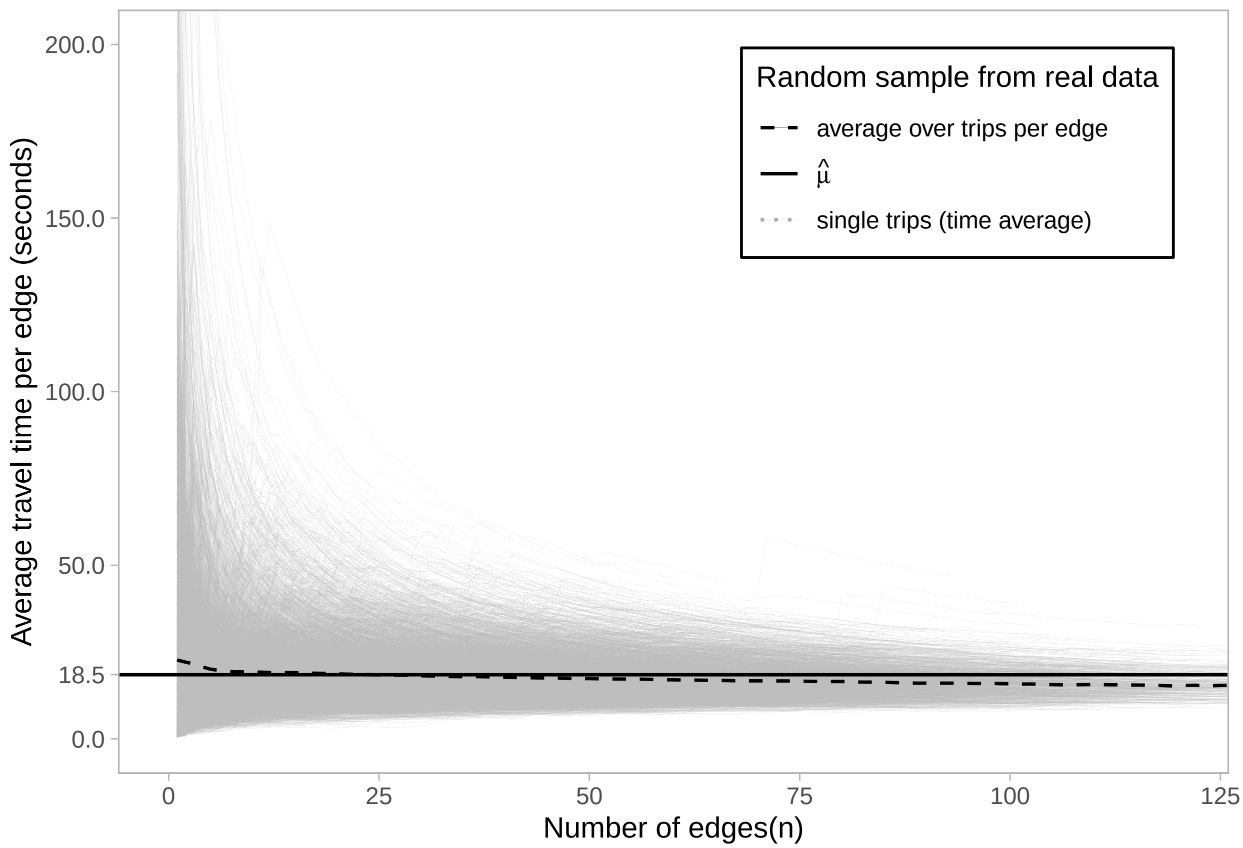
```r
ggplot(a, aes(x = N, y = V2)) +
    geom_line(aes(group=tripID,color=tripID))
```




```
#b = trips[, .(N=1:.N, duration_secs),tripID]
#ggplot(b, aes(x = N, y = duration_secs)) +
#  theme_light() +
#    geom_line(aes(group=tripID),color="grey")
c = trips[, .(N=1:.N, cummean(duration_secs)),tripID]
c=c[, ave_time := mean(cummean(V2)[1:.N]), N]
ggplot(c, aes(x = N, y = V2)) +
  theme_light() +
  geom_line(aes(group = tripID, linetype = "single trips (time average)",color="single trips (time avera
  geom_line(aes(y = ave_time, linetype = "average over trips per edge"),size=0.6) +
  labs(y = "Average travel time per edge (seconds)",
       x = "Number of edges(n)") +
  geom_hline(aes(yintercept = muhat, linetype = "mu"),size=0.6) +
  coord_cartesian(xlim = c(0, 120), ylim = c(0, 200)) +
  scale_color_manual(name = "Legend",values = c("mu" ="black","single trips (time average)" ="grey","ave
  scale_linetype_manual(
    name = "Random sample from real data",
    values = c("mu" = "solid",
      "single trips (time average)" = "dotted",
      "average over trips per edge" = "dashed"),
    labels = c("mu" = expression(hat(mu)),
      "single trips (time average)" = "single trips (time average)",
      "average over trips per edge" = "average over trips per edge")) +
    theme(legend.position = c(0.95, 0.95),
      legend.justification = c(1, 1),
      legend.text.align = 0,
```

```
      legend.background = element_rect(color = "black", fill = "white"),
      panel.grid.major = element_blank(),
      panel.grid.minor = element_blank()) +
    scale_y_continuous(breaks=c(seq(0,200,50),round(muhat,1)))+
  guides(linetype = guide_legend(
    override.aes = list(color = c("black", "black", "darkgrey"),
                        size = c(0.6, 0.6, 0.4))))
```

```
## 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.
```

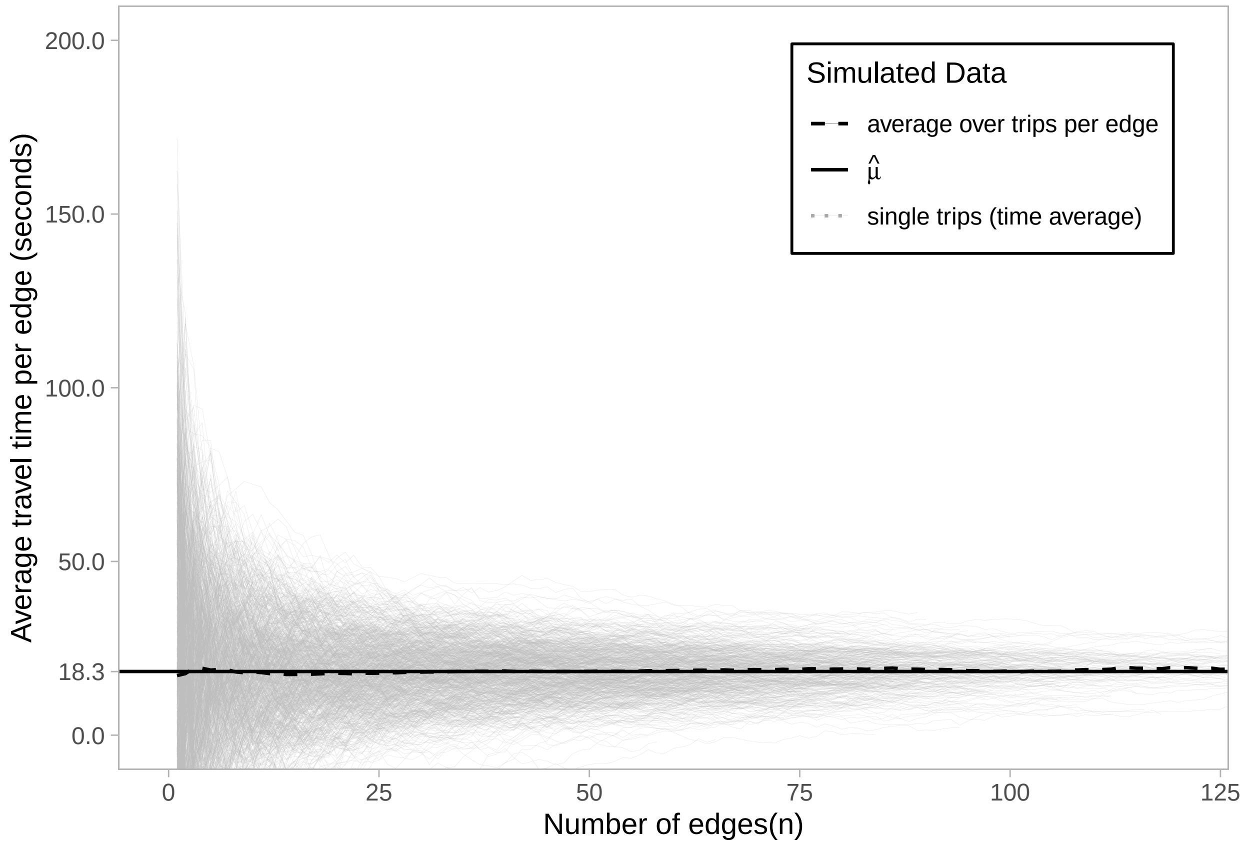


simulation using pop_sed. the data contains 1000 trips, trips length are the same as the real data. The trip simulator use the same estimated mu and variance in real sampled data.

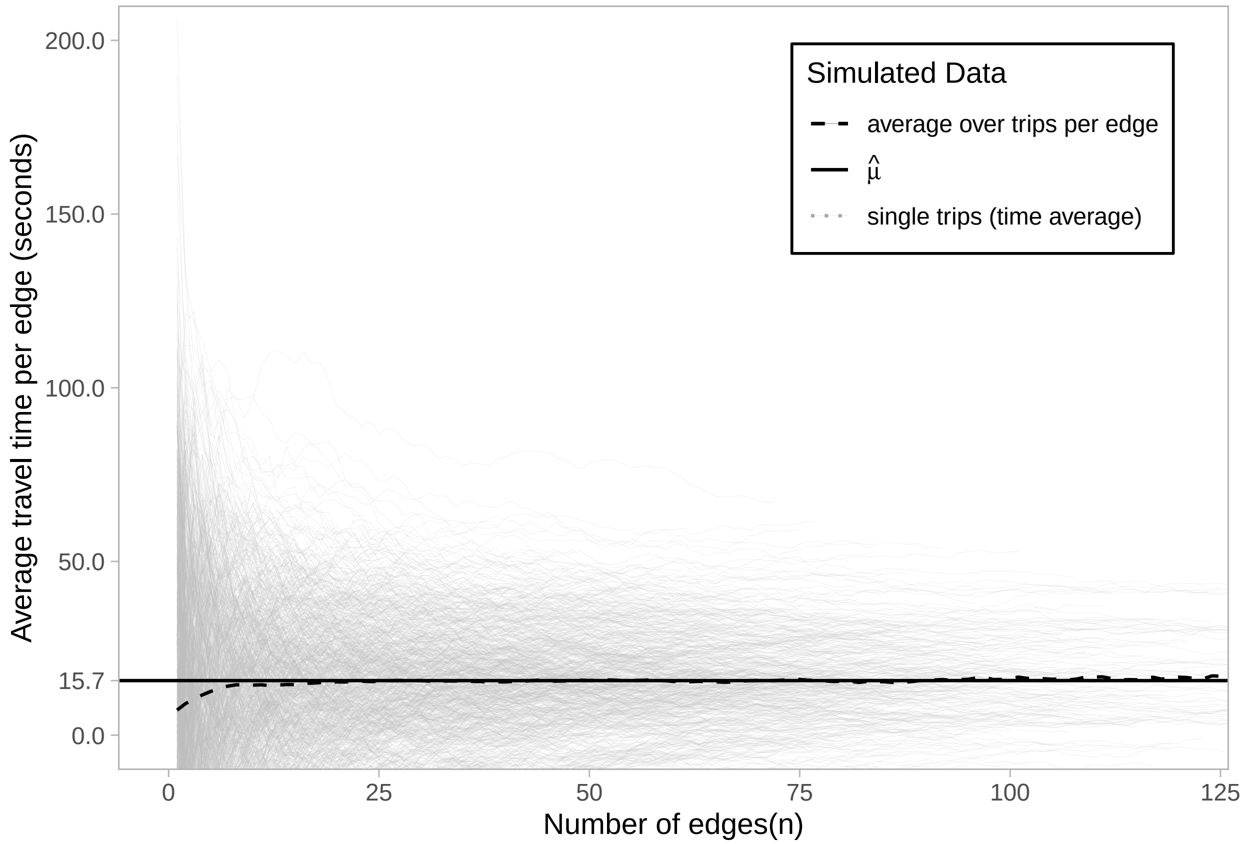
```
n=trip_length
index =1;tripID=c();duration=c()
while (index<=1000) {
  if(n[index]<2)n[index]<-2
  tripID<-c(tripID,rep(index,(n[index])))
  duration<-c(duration,diff(population_sde(n[index]+1,muhat/avg_seg_length,vhat/avg_seg_length,1)))
  index<-index+1
}
```

```r
simulated_trips <- data.table(tripID,duration)
d = simulated_trips[, .(N=1:.N, cummean(duration)),tripID]
d=d[, ave_time := mean(cummean(V2)[1:.N]), N]
simulated_groupmean <- simulated_trips[, mean(duration),tripID]$V1
simulated_mu = mean(simulated_groupmean)
```

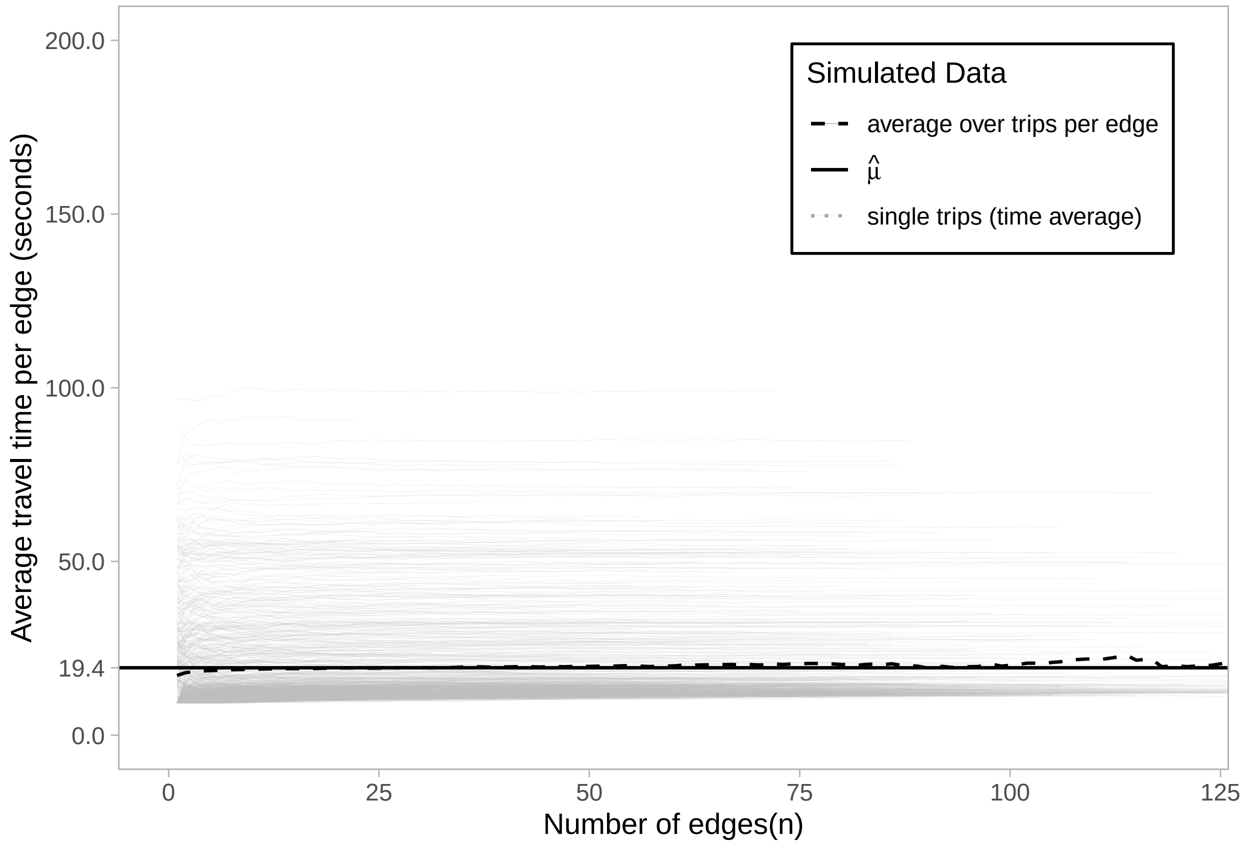
```r
dataploter<-function(data,title="Simulated Data"){
ggplot(data, aes(x = N, y = V2)) +
  theme_light() +
  geom_line(aes(group = tripID, linetype = "single trips (time average)",color="single trips (time avera
  geom_line(aes(y = ave_time, linetype = "average over trips per edge"),size=0.6) +
  labs(y = "Average travel time per edge (seconds)",
       x = "Number of edges(n)") +
  geom_hline(aes(yintercept = simulated_mu, linetype = "mu"),size=0.6) +
  coord_cartesian(xlim = c(0, 120), ylim = c(0, 200)) +
  scale_color_manual(name = "Legend",values = c("mu" ="black","single trips (time average)" ="grey","ave
  scale_linetype_manual(
    name = title,
    values = c("mu" = "solid",
      "single trips (time average)" = "dotted",
      "average over trips per edge" = "dashed"),
    labels = c("mu" = expression(hat(mu)),
      "single trips (time average)" = "single trips (time average)",
      "average over trips per edge" = "average over trips per edge")) +
    theme(legend.position = c(0.95, 0.95),
      legend.justification = c(1, 1),
      legend.text.align = 0,
      legend.background = element_rect(color = "black", fill = "white"),
      panel.grid.major = element_blank(),
      panel.grid.minor = element_blank()) +
    scale_y_continuous(breaks=c(seq(0,200,50),round(simulated_mu,1)))+
  guides(linetype = guide_legend(
    override.aes = list(color = c("black", "black", "darkgrey"),
                        size = c(0.6, 0.6, 0.4))))}
dataploter(d)
```



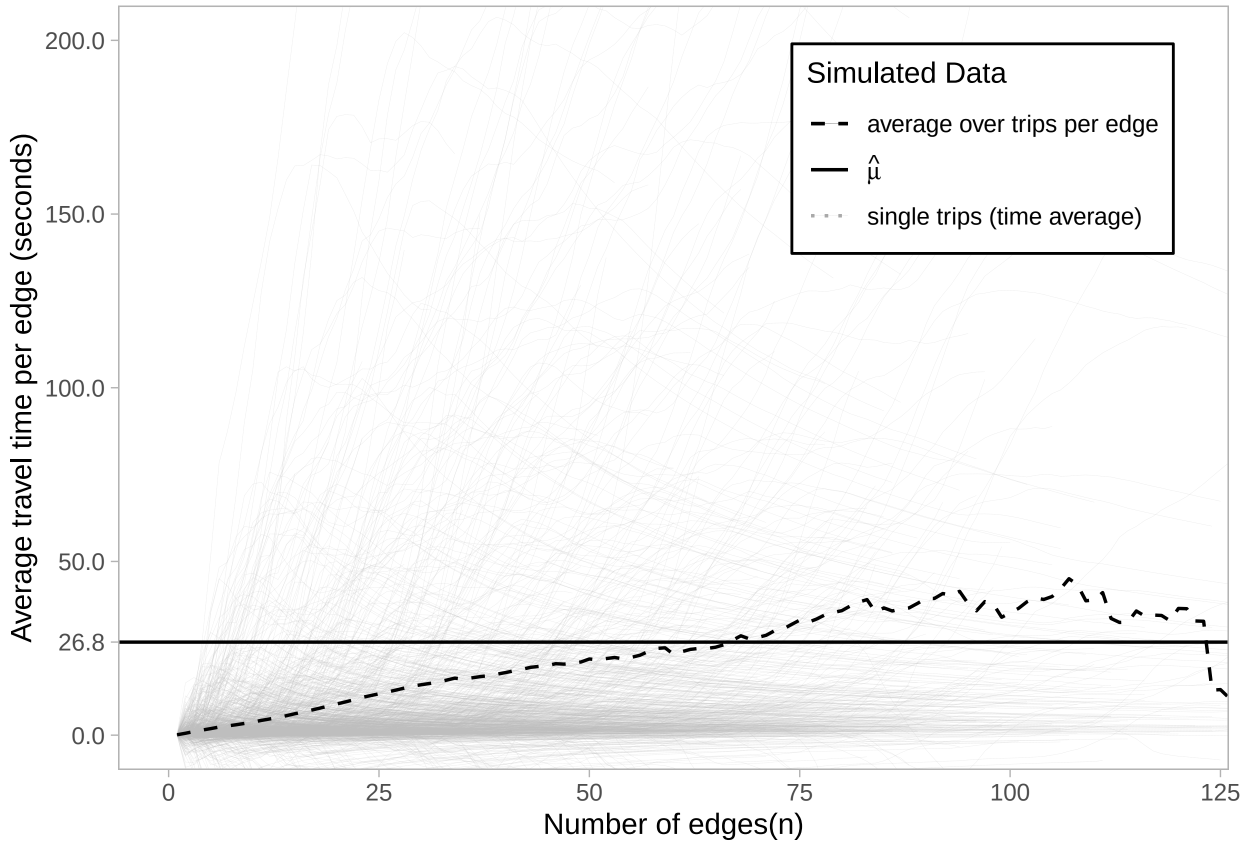

```
n=trip_length
index =1;tripID=c();duration=c()
while (index<=1000) {
  if(n[index]<2)n[index]<-2
  tripID<-c(tripID,rep(index,(n[index])))
  duration<-c(duration,diff(stochastic_drift_sde(n[index]+1,muhat,vhat,1)))
  index<-index+1
}
simulated_trips <- data.table(tripID,duration)
d = simulated_trips[, .(N=1:.N, cummean(duration)),tripID]
d=d[, ave_time := mean(cummean(V2)[1:.N]), N]
simulated_groupmean <- simulated_trips[, mean(duration),tripID]$V1
simulated_mu = mean(simulated_groupmean)
dataploter(d)
```




```
n=trip_length
index =1;tripID=c();duration=c()
while (index<=1000) {
  if(n[index]<2)n[index]<-2
  tripID<-c(tripID,rep(index,(n[index])))
  duration<-c(duration,diff(stochastic_sde(n[index]+1,muhat/avg_seg_length,vhat/avg_seg_length,1)))
  index<-index+1
}
simulated_trips <- data.table(tripID,duration)
d = simulated_trips[, .(N=1:.N, cummean(duration)),tripID]
d=d[, ave_time := mean(cummean(V2)[1:.N]), N]
simulated_groupmean <- simulated_trips[, mean(duration),tripID]$V1
simulated_mu = mean(simulated_groupmean)
dataploter(d)
```




```
n=trip_length
index =1;tripID=c();duration=c()
while (index<=1000) {
  if(n[index]<2)n[index]<-2
  tripID<-c(tripID,rep(index,(n[index])))
  duration<-c(duration,(stochastic_mean_reversion(n[index],muhat/avg_seg_length,vhat/avg_seg_length,1))
  index<-index+1
}
simulated_trips <- data.table(tripID,duration)
d = simulated_trips[, .(N=1:.N, cummean(duration)),tripID]
d=d[, ave_time := mean(cummean(V2)[1:.N]), N]
simulated_groupmean <- simulated_trips[, mean(duration),tripID]$V1
simulated_mu = mean(simulated_groupmean)
dataploter(d)
```


Simulated Data
average over trips per edge
$\hat{\mu}$
single trips (time average)
Average travel time per edge (seconds)
200.0
150.0
100.0
50.0
26.8
0.0
0
25
50
75
100
125
Number of edges(n)