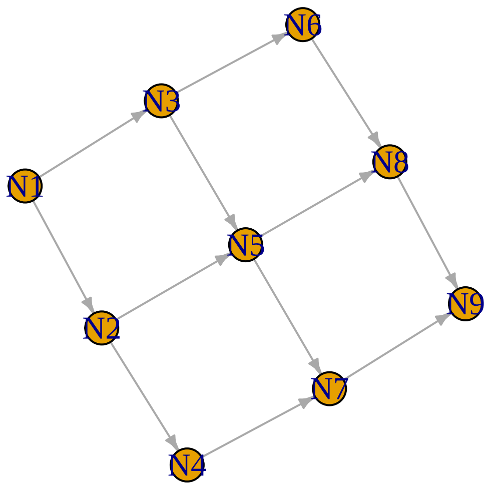
# Thompson sampling for shortest traveling time problem

*Xiao Chen*

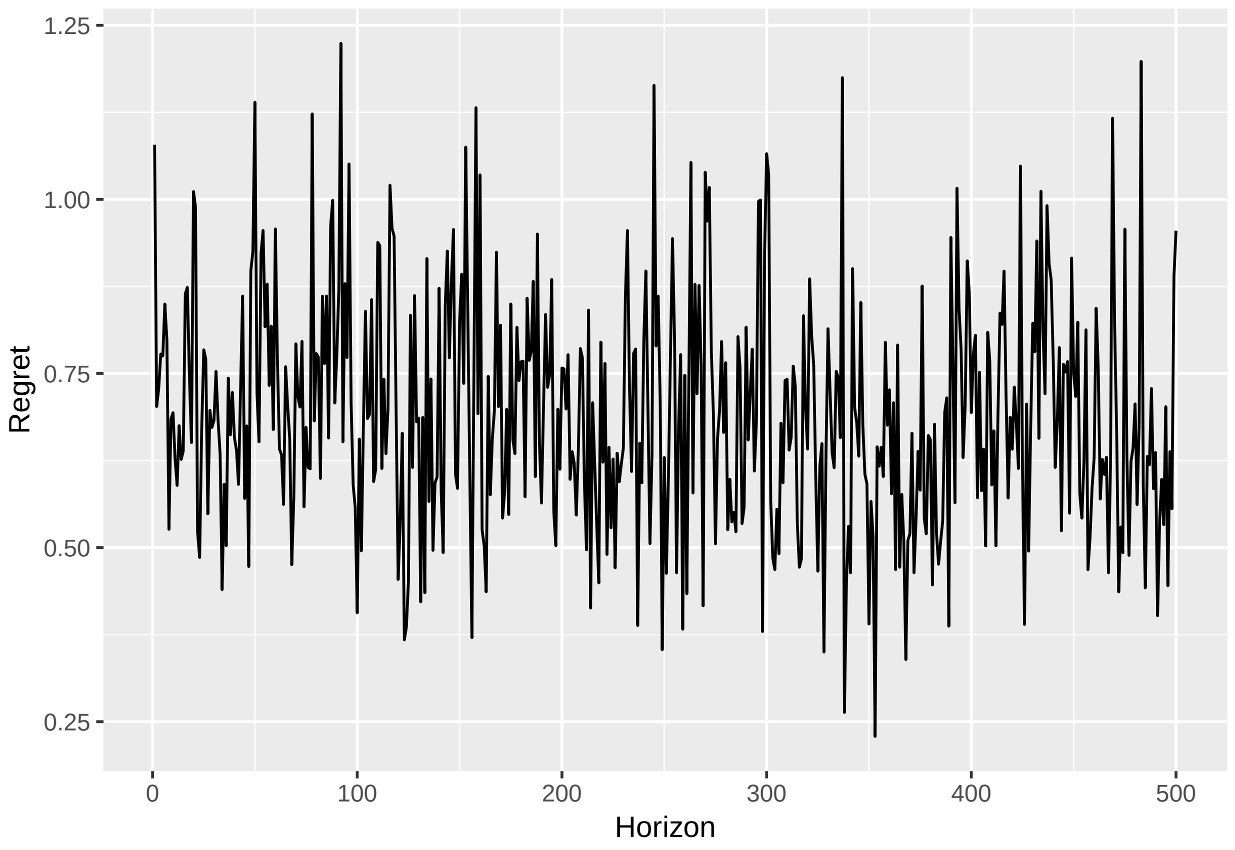
*February 3, 2019*

```r
library(igraph)

# ########################
# constructing network #
# ########################
s = c("N1", "N1", "N2", "N2", "N3", "N3", "N4", "N5", "N5", "N6", "N7", "N8")
t = c("N2", "N3", "N4", "N5", "N5", "N6", "N7", "N7", "N8", "N8", "N9", "N9")

df <- data.frame(s, t)
df.g <- graph.data.frame(d = df, directed = TRUE)
plot(df.g, edge.arrow.size=.4)
paraT = data.frame( edge = attr(E(df.g), "vnames"),
            theta = rep(0, length(E(df.g))) )

n.mat = as.matrix(df.g[,])
```



```r
################################
#' @find s-t paths in graph
#' built-in functions
#' V(df.g)
### pre setup the graph
mu_e = -1/2
sigma_e_sq = 1
sigma_tilde_sq = 1
all.paths = all_simple_paths(df.g, from=1, to=9)
#' path.1 = E(df.g, path = all.paths$res[[1]])
all.paths.N = length(all.paths)
all.edges.N = length(E(df.g))
#'
# horizon = 1000
set.seed(8087)
run_sim_travel <- function(n_iter=1, horizon = 1000 ){
      prior.theta =rlnorm(all.edges.N, meanlog = -1/2, sdlog = 1)
      # df.g = df.g %>% set_edge_attr("cost", value = rlnorm(gsize(.),  meanlog = , sdlog = 1))
      # edge_attr(df.g, "cost", index = E(df.g) )
      # attr(all.edges, "cost") = prior.theta
      # E(df.g) = all.edges
      theta.DT = prior.theta
        # attr(all.edges, "cost") = theta.DT
      regret.horizon.vector = rep(100, horizon)
      for (t in 1:horizon){
        # t-th step
        # edge_attr(df.g, "cost", index=E(df.g)) = theta.DT
        # a list of rewards
        reward.of.paths = sapply(seq(all.paths.N), function(i) {
                                   path.i.edges = E(df.g, path =all.paths[[i]] )
                                   # reward = -sum(edge_attr(df.g, "cost", index = E(df.g) )[match(path
                                   reward = -sum( theta.DT[match(path.i.edges, E(df.g))] )
                                   sapply(match(path.i.edges, E(df.g)), function(j) {
                                       numerater1 = (1/(sigma_e_sq))*mu_e + (1/sigma_tilde_sq) * (log(t
                                       denominator1 = (1/sigma_e_sq) + (1/sigma_e_sq)
                                       theta.DT[j] <<- rlnorm(1, mean= numerater1/denominator1, sd = 1/
                                     } )

                                   #' -------- discarded --------
                                   # print(edge_attr(df.g, "cost", index = E(df.g) )[match(path.i.edges
                                   # for (e in path.i.edges){
                                   #   print(which(e == E(df.g)))
                                   # }

                                   return(reward)

          } )

        # print(reward.of.paths)
        # print(E(df.g, path = all.paths[[which.max(reward.of.paths)]] ) )
        # print( paste("reward: ", max(reward.of.paths) ))
        #
        df.g.shortest.path = E(df.g, path= shortest_paths(df.g, from=1, to=9, weights = theta.DT)$vpath
        reward_opt = -sum( theta.DT[match(df.g.shortest.path, E(df.g))] )
        # print(paste( "regret:", (reward_opt - max(reward.of.paths))^2 ) )
        regret.horizon.vector[t] = abs(reward_opt - max(reward.of.paths) )

      }
      return(regret.horizon.vector)
}

# import parallel computing
library(parallel)
multiple.simulate = mclapply(seq(10), function(i) {run_sim_travel(n_iter=i, horizon = 500)} )
res = do.call("rbind", multiple.simulate)
# plot(x=seq(horizon), y= regret.horizon.vector, lty=1)
library(ggplot2)
# library(dplyr)
horizon = 500
```

```r
df.res = data.frame( Horizon=seq(horizon), Regret = colMeans(res) )
ggplot(df.res, aes(x=Horizon, y = Regret )) + geom_line()
```


Regret
Horizon