Data Simulation Project

Simulating Google Maps

By: Sama Amr & Mona Mahmoud

20th May, 2023

Table of contents

Question of Interest.. 2
Objective..2
Design of Experiment..3
Summary and Conclusion...7
Recommendation...7

Question of Interest

Our project aims to answer the question of how to identify the shortest route according to the one that takes the shortest time. That was mainly because with the expansion of roads' infrastructure in Egypt, a person can go to a specific destination using numerous different routes. However, the problem lies in identifying the most efficient route to minimize the time taken to travel from one place to another.

Objective

Our project aims to mimic Google Maps in finding the fastest route based on a set of parameters. By considering different variables that affect travel time, such as road characteristics, traffic lights, and speed limits we try to find the fastest route among the given set of roads.

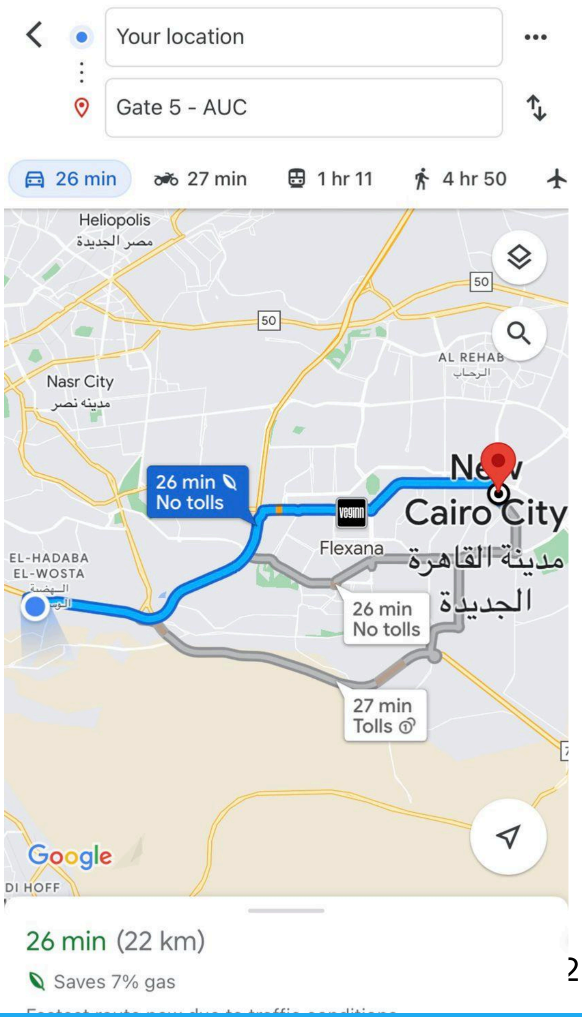

image source: screenshot from our location to AUC

Design of Experiment

Input

We have defined 6 variables, of which 1 is inputted by the user, which is “n,” corresponding to the number of roads the user would wish to compare, and the rest are randomly generated using simulation. We have taken into consideration that they must be generated from different distributions since we would need the generated numbers to have specific characteristics to be as much as the real-life data. Here is a description of the variables used throughout the simulation.

- n ➔ the number of roads the user would wish to compare
- total_distance ➔ corresponding to the total distance of each route to the same destination in kilometers
- no_traffic ➔ the number of traffic lights the user might encounter during the journey
- dlight ➔ the distance to the first traffic light encountered in kilometers
- tlight ➔ the time for which the traffic light would remain red from 0 to 60 seconds according to each traffic light
- splimit ➔ represents the speed limit in each of the routes in kilometers per hour

Explanation of the Algorithm

- Step 1: We calculated the time taken to reach the first traffic light by dividing the distance to the first traffic light by the speed limit and then multiplied it by 3600 to convert it to seconds
- Step 2: We calculated the time in second to reach the traffic time and that is to figure out if the user is going to stop at that traffic light or not.
- Step 3: Using a for-loop we stored the values of the waiting time at each traffic light in an array
- Step 4: We calculated the total time needed to reach the destination
- Step 5: We created another for-loop that stores the average time taken for each route in an array
- Step 6: using a nested if statements, we calculated the stopping time due to traffic and then added it to the total time taken to complete the journey.
- Step 7: We used print() to output the minimum route stored in the array using the min() and which.min() functions.
- These calculations were all done under the assumption that:
    - The car starts at time 0.
    - The car moves at a constant speed (randomly generated) throughout its journey.
    - The traffic lights are either red, green, or yellow lights.
    - All traffic lights (on each route) are connected.

Output

```
> # to call the function
> shortest_route(n)
[1] "The fastest route is route 38 and it takes 121.04 minutes to arrive"
> # to call the function
> shortest_route(n)
[1] "The fastest route is route 76 and it takes 41.64 minutes to arrive"
> # to call the function
> shortest_route(n)
[1] "The fastest route is route 80 and it takes 68.64 minutes to arrive"
```

The code outputs the shortest route that would take the least time out of all the routes to reach the destination. The output of the function isn't the same after each run because it delivers the shortest route, which varies because new data is randomly generated every time.

Code

```
n=100 #how many roads we are comparing

shortest_route= function(n){

  total_distance=rexp(n,0.003) #total distance in km

  no_traffic= runif(n,0,n) #number of traffic light from 0 to n

  dlight=rnorm(n,0,1) #distance to the first traffic light in km
```

```
 dlight=pmax(dlight,0) #to select only the positive numbers generated from the previous
normal distribution
 tlight=runif(n,30,60) #time of traffic light being red from 30 to 60 in seconds
 splimit= rexp(n,0.2) #speed limit (maximum speed) in km/h
 limit=runif(n,0,splimit)# speed of each car in km/h

avg_time= function(total_distance,no_traffic, dlight,tlight, limit){
   tttraffic= (dlight/limit)  #time in hours to reach first traffic light
   tttraffic= tttraffic *3600  #time in seconds to reach first traffic light

   #Based on the assumption that all traffic lights are connected, the time taken to reach
the first traffic light (‘tttraffic’) is used to determine whether the car stops or not, if the car
stops at the first traffic light, it implies that it will continue to encounter red lights at all
subsequent traffic lights along the route, and the same for green and yellow lights.
   x= (tttraffic %%tlight)/tlight
   stop= numeric(n) #creates an array containing the total time of the car stopping due to
traffic lights
   z= sample(1:3,n,replace=T) # takes a sample of size n with numbers from 1 = red, 2=
yellow, 3= green
   for (i in  1:n) {
     if((z[i]==1) || (x[i]< tlight[i]/60)){
       stop[i]=no_traffic[i]*tlight[i] # waiting time at red light
```

```
    }else if ((z[i]==2) || (x[i]== tlight[i]/60)){
      stop[i]=no_traffic[i]*tlight[i]*0.5 # waiting time at yellow light
    }else if ((z[i]==3) || (x[i]> tlight[i]/60)){
      stop[i]=0 #waiting time at green light
    } }
  stop= stop/3600 #converting stopping time to hours

  end_time= (total_distance/limit)
  end_time=end_time+stop  #total time of travel and stopping time

  return(end_time)
 }
 #storing all routes together in an array to compare them
 route <- numeric(n)
 for(i in 1:n){
  y= avg_time(total_distance,no_traffic, dlight,tlight, limit)
  route[i]= y[i]  }
 print(paste("The fastest route is route", which.min(route),"and it takes",
round(min(route)*60,2), "minutes to arrive"))
}
# to call the function
shortest_route(n)
```

Summary and conclusion

Our project aims to identify the most efficient route when it comes to time in order to get to one's destination faster. This was all done by the help of simulation and the random number generators to hypothetically test how it works and mimic the real life event.

Recommendation

The code provided is a simple simulation of a real-life problem; however, it doesn’t mimic the full functionality and complexity. That is due to it not incorporating realistic data, but synthetic (generated) data; therefore, in order to increase the accuracy, use real world data. Another recommendation could be adding more variables that impact the time taken to reach the destination such as rush hours, and road closures.