Analysis of Simulated Vance County EMS Response Data

September 18, 2023

In what follows, we carry out a basic analysis of the simulated data to test the R library mapsapi's Google API interface.

1 Input Data; Basic Data Processing

Here we import the data elements, carry out some basic data processing and display a few summaries of the data set:

```
rm(list=ls()) ## Completely clear the workspace.
opts_chunk$set(fig.path='./figs/',cache.path='./cache/')
library(mapsapi)
library(mgcv)
library(xtable)
library(lme4)
api.key<-scan("../api.key",what=" ")
```

```
##x<-read.csv("VanceModelDataset.csv")
x<-read.csv("VanceMockData1.csv")
head(x)

##    REF.GRID DISPATCH.PRIORITY.NAME REF.GPS.LAT REF.GPS.LON BASE.NAME VEH.GRID
## 1   3 South              Emergency     36.3085    -78.4563 Company 9  Medic 5
## 2 2 Central              Emergency     36.3306    -78.4040 Company 9  Medic 6
## 3 2 Central              Emergency     36.3335    -78.4399 Company 9  Medic 1
## 4 2 Central              Emergency     36.3351    -78.4410 Company 9  Medic 5
## 5 2 Central          Non Emergency     36.3401    -78.4017 Company 9  Medic 6
## 6 2 Central              Emergency     36.3315    -78.3929 Company 9  Medic 1
##             VEHCGPS             DT.DISP          DT.ENROUTE           DT.ARRIVE
## 1 36.345, -78.3905 01/01/1789 06:46:00 01/01/1789 06:46:00 01/01/1789 06:52:00
## 2 36.345, -78.3905 01/01/1789 08:30:00 01/01/1789 08:30:00 01/01/1789 08:34:00
## 3 36.345, -78.3905 01/01/1789 10:22:00 01/01/1789 10:22:00 01/01/1789 10:27:00
## 4 36.345, -78.3905 01/01/1789 11:38:00 01/01/1789 11:38:00 01/01/1789 11:44:00
## 5 36.345, -78.3905 01/01/1789 12:33:00 01/01/1789 12:33:00 01/01/1789 12:37:00
## 6 36.345, -78.3905 01/01/1789 14:18:00 01/01/1789 14:18:00 01/01/1789 14:22:00
##              DT.LVREF           DT.ARVREC        DT.AVAILABLE
## 1 01/01/1789 07:07:00 01/01/1789 07:13:00 01/01/1789 07:32:00
## 2 01/01/1789 08:39:00 01/01/1789 08:46:00 01/01/1789 09:00:00
## 3 01/01/1789 10:36:00 01/01/1789 10:39:00 01/01/1789 10:54:00
## 4                                         01/01/1789 12:08:00
## 5 01/01/1789 12:38:00 01/01/1789 12:45:00 01/01/1789 12:52:00
## 6 01/01/1789 14:38:00 01/01/1789 14:47:00 01/01/1789 15:11:00
##                 REC.NAME
## 1 Maria Parham Hospital
## 2 Maria Parham Hospital
## 3 Maria Parham Hospital
## 4
## 5 Maria Parham Hospital
## 6 Maria Parham Hospital
```

```
summary(x)
```

```
##    REF.GRID         DISPATCH.PRIORITY.NAME  REF.GPS.LAT     REF.GPS.LON
##  Length:499         Length:499             Min.   :35.96   Min.   :-78.81
##  Class :character   Class :character       1st Qu.:36.32   1st Qu.:-78.43
##  Mode  :character   Mode  :character       Median :36.33   Median :-78.40
##                                            Mean   :36.33   Mean   :-78.41
##                                            3rd Qu.:36.34   3rd Qu.:-78.39
##                                            Max.   :36.52   Max.   :-78.20
##                                            NA's   :2       NA's   :2
##   BASE.NAME           VEH.GRID           VEHCGPS            DT.DISP
##  Length:499         Length:499         Length:499         Length:499
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##   DT.ENROUTE         DT.ARRIVE           DT.LVREF          DT.ARVREC
##  Length:499         Length:499         Length:499         Length:499
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##  DT.AVAILABLE         REC.NAME
##  Length:499         Length:499
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
##
```

1.1 Get Station Coordinates

```
gps.cent<-unique(x$VEHCGPS[x$BASE.NAME=="Company 9"])
gps.south<-unique(x$VEHCGPS[x$BASE.NAME=="Company 1"])
gps.centN<-as.numeric(strsplit(gps.cent,",")[[1]])
gps.southN<-as.numeric(strsplit(gps.south,",")[[1]])
## First option for a north station
gps.northNN<-c(36.430596, -78.431689) ##NN=Near North
##Second option for north station
gps.northFN<-c(36.495537112943886,-78.42090194629898) ##FN=Far North
## GPS coordinates of Maria Parham Hospital
gps.hospital<-c(36.33089064918619, -78.44930886477614)
```

Destination hospital coordinates:

```
x$REC.LON<-rep(NA,nrow(x))
x$REC.LAT<-rep(NA,nrow(x))
x$REC.LON[x$REC.NAME=="Maria Parham Hospital"]<-(-78.44930886477614)
x$REC.LAT[x$REC.NAME=="Maria Parham Hospital"]<-(36.33089064918619)
x$REC.LON[x$REC.NAME=="Granville Medical Center"]<-(-78.59367173834997)
x$REC.LAT[x$REC.NAME=="Granville Medical Center"]<-(36.33043072571129)
x$REC.LON[x$REC.NAME=="Duke Health Duke University Medical Center"]<-(-78.93687608445487)
x$REC.LAT[x$REC.NAME=="Duke Health Duke University Medical Center"]<-(36.00643609468812)
```

Drop cases with missing call GPS coordinates:

```
table(is.na(x$REF.GPS.LON))

##
## FALSE  TRUE
##   497     2

x<-x[!is.na(x$REF.GPS.LON),]
```

1.2 Format Time Character Strings as Times

1.2.1 Google Needs Current/Future Times

Change the year from 1789 to 2024.

```
x$DT.DISP<-sub("1789","2024",x$DT.DISP)
x$DT.ENROUTE<-sub("1789","2024",x$DT.ENROUTE)
x$DT.ARRIVE<-sub("1789","2024",x$DT.ARRIVE)
x$DT.LVREF<-sub("1789","2024",x$DT.LVREF)
x$DT.ARVREC<-sub("1789","2024",x$DT.ARVREC)
x$DT.AVAILABLE<-sub("1789","2024",x$DT.AVAILABLE)
```

1.2.2 Google Needs Times in POSIXct Format

Convert times importated as character strings to times formated as R POSIX values.

```
x$DT.DISP<-strptime(x$DT.DISP,format="%m/%d/%Y %H:%M:%S",tz="EST")
x$DT.ENROUTE<-strptime(x$DT.ENROUTE,format="%m/%d/%Y %H:%M:%S",tz="EST")
x$DT.ARRIVE<-strptime(x$DT.ARRIVE,format="%m/%d/%Y %H:%M:%S",tz="EST")
x$DT.LVREF<-strptime(x$DT.LVREF,format="%m/%d/%Y %H:%M:%S",tz="EST")
x$DT.ARVREC<-strptime(x$DT.ARVREC,format="%m/%d/%Y %H:%M:%S",tz="EST")
x$DT.AVAILABLE<-strptime(x$DT.AVAILABLE,format="%m/%d/%Y %H:%M:%S",tz="EST")
```

2 Estimate Response Travel Times from Each Station

Compute travel times between the two existing and two proposed station locations and each destination. Do so assuming the pessimistic, best guess and optimistic traffic assumptions, in turn. In addition, save the ‘green light’ duration and distance.

2.1 Best Guess Scenario

```
travelTimeBG<-NULL
distance<-NULL
durationGL<-NULL
for (i in 1:nrow(x)){
    api.out<-mp_matrix(
        origins = rbind(gps.southN[c(2,1)],gps.centN[c(2,1)],
                        gps.northNN[c(2,1)],gps.northFN[c(2,1)]),
        destinations = cbind(x$REF.GPS.LON,x$REF.GPS.LAT)[i,],
        mode="driving",
        traffic_model="best_guess",
        departure_time=as.POSIXct(x$DT.ENROUTE[i]),  ##as POSIXct
        ##departure_time=Sys.time() + as.difftime(4, units = "hours"),
        key = api.key,
        quiet = TRUE)
    times.out<-mp_get_matrix(api.out,
                              value = "duration_in_traffic_s")
    gl.out<-mp_get_matrix(api.out,
                              value = "duration_s")
    dist.out<-mp_get_matrix(api.out,
                              value = "distance_m")
    travelTimeBG<-rbind(travelTimeBG,matrix(times.out,nrow=1))
    distance<-rbind(distance,matrix(dist.out,nrow=1))
    durationGL<-rbind(durationGL,matrix(gl.out,nrow=1))
    Sys.sleep(0.5)
}
```

Google API computed response travel times from each station, measured in seconds, where ‘So’ refers to the south station, ‘Ce’ refers to the central, ‘NN’ to the proposed near north station and ‘FN’ to the proposed far north station. ‘BG’ refers to the ‘best guess’ scenario.

```
colnames(travelTimeBG)<-c("eTT.BG.So","eTT.BG.Ce","eTT.BG.NN","eTT.BG.FN")
colnames(distance)<-c("Dist.So","Dist.Ce","Dist.NN","Dist.FN")
colnames(durationGL)<-c("eTT.GL.So","eTT.GL.Ce","eTT.GL.NN","eTT.GL.FN")
head(travelTimeBG)

##      eTT.BG.So eTT.BG.Ce eTT.BG.NN eTT.BG.FN
## [1,]       539       406       805      1173
## [2,]       549       220       633       993
## [3,]       752       346       743      1127
## [4,]       770       306       712      1085
## [5,]       766       181       670      1036
## [6,]       722       235       815      1186

summary(travelTimeBG)
```

```
##    eTT.BG.So        eTT.BG.Ce        eTT.BG.NN        eTT.BG.FN   
##  Min.   :  38.0   Min.   :   9.0   Min.   :  13.0   Min.   :  11  
##  1st Qu.: 491.0   1st Qu.: 280.0   1st Qu.: 679.0   1st Qu.:1054  
##  Median : 604.0   Median : 385.0   Median : 779.0   Median :1150  
##  Mean   : 661.3   Mean   : 440.2   Mean   : 816.7   Mean   :1177  
##  3rd Qu.: 737.0   3rd Qu.: 544.0   3rd Qu.: 937.0   3rd Qu.:1318  
##  Max.   :2890.0   Max.   :2412.0   Max.   :2900.0   Max.   :3284  
```

2.2 Pessimistic Scenario

```
travelTimePe<-NULL
for (i in 1:nrow(x)){
    api.out<-mp_matrix(
        origins = rbind(gps.southN[c(2,1)],gps.centN[c(2,1)],
                        gps.northNN[c(2,1)],gps.northFN[c(2,1)]),
        destinations = cbind(x$REF.GPS.LON,x$REF.GPS.LAT)[i,],
        mode="driving",
        traffic_model="pessimistic",
        departure_time=as.POSIXct(x$DT.ENROUTE[i]),  ##as POSIXct
        ##departure_time=Sys.time() + as.difftime(4, units = "hours"),
        key = api.key,
        quiet = TRUE)
    times.out<-mp_get_matrix(api.out,
                              value = "duration_in_traffic_s")
    travelTimePe<-rbind(travelTimePe,matrix(times.out,nrow=1))
    Sys.sleep(0.5)
}
```

Google API computed response travel times from each station, measured in seconds; ‘Pe’ refers to the ‘pessimistic’ scenario.

```
colnames(travelTimePe)<-c("eTT.Pe.So","eTT.Pe.Ce","eTT.Pe.NN","eTT.Pe.FN")
head(travelTimePe)

##      eTT.Pe.So eTT.Pe.Ce eTT.Pe.NN eTT.Pe.FN
## [1,]       616       440       859      1267
## [2,]       650       251       689      1076
## [3,]       918       384       796      1217
## [4,]      1026       363       784      1193
## [5,]       965       220       725      1123
## [6,]       890       250       963      1358

summary(travelTimePe)

##    eTT.Pe.So        eTT.Pe.Ce      eTT.Pe.NN        eTT.Pe.FN   
##  Min.   :  39.0   Min.   :   9   Min.   :  15.0   Min.   :  12  
##  1st Qu.: 556.0   1st Qu.: 313   1st Qu.: 730.0   1st Qu.:1128  
##  Median : 695.0   Median : 434   Median : 866.0   Median :1261  
##  Mean   : 756.8   Mean   : 488   Mean   : 896.8   Mean   :1279  
##  3rd Qu.: 891.0   3rd Qu.: 595   3rd Qu.:1042.0   3rd Qu.:1429  
##  Max.   :3174.0   Max.   :2680   Max.   :3204.0   Max.   :3655  
```

2.3 Optimistic Scenario

```
travelTimeOp<-NULL
for (i in 1:nrow(x)){
    api.out<-mp_matrix(
        origins = rbind(gps.southN[c(2,1)],gps.centN[c(2,1)],
                        gps.northNN[c(2,1)],gps.northFN[c(2,1)]),
        destinations = cbind(x$REF.GPS.LON,x$REF.GPS.LAT)[i,],
        mode="driving",
        traffic_model="optimistic",
        departure_time=as.POSIXct(x$DT.ENROUTE[i]),  ##as POSIXct
        ##departure_time=Sys.time() + as.difftime(4, units = "hours"),
        key = api.key,
        quiet = TRUE)
    times.out<-mp_get_matrix(api.out,
                             value = "duration_in_traffic_s")
    travelTimeOp<-rbind(travelTimeOp,matrix(times.out,nrow=1))
    Sys.sleep(0.5)
}
```

Google API computed response travel times from each station, measured in seconds; ‘Op’ refers to the ‘optimal’ scenario.

```
colnames(travelTimeOp)<-c("eTT.Op.So","eTT.Op.Ce","eTT.Op.NN","eTT.Op.FN")
head(travelTimeOp)

##      eTT.Op.So eTT.Op.Ce eTT.Op.NN eTT.Op.FN
## [1,]       507       372       758      1097
## [2,]       508       210       587       933
## [3,]       699       343       728      1090
## [4,]       679       283       671      1032
## [5,]       728       187       642       994
## [6,]       668       243       753      1124

summary(travelTimeOp)

##    eTT.Op.So        eTT.Op.Ce      eTT.Op.NN        eTT.Op.FN
##  Min.   :  38.0   Min.   :   9   Min.   :  13.0   Min.   :  10
##  1st Qu.: 483.0   1st Qu.: 277   1st Qu.: 653.0   1st Qu.:1014
##  Median : 586.0   Median : 370   Median : 749.0   Median :1113
##  Mean   : 637.1   Mean   : 428   Mean   : 780.3   Mean   :1132
##  3rd Qu.: 697.0   3rd Qu.: 529   3rd Qu.: 899.0   3rd Qu.:1258
##  Max.   :2815.0   Max.   :2340   Max.   :2824.0   Max.   :3150
```

2.4 Observed Times

Observed times, measured in seconds:

```
## Duration from dispatch to clear
x$dispToClearTime<-difftime(x$DT.AVAILABLE,x$DT.DISP,units="secs")
## Duration from dispatch to enroute
```

```r
x$timeToEnroute<-difftime(x$DT.ENROUTE,x$DT.DISP,units="secs")
## Response Time, Station to Scene
x$observedTT<-difftime(x$DT.ARRIVE,x$DT.ENROUTE,units="secs")
## Duration on Scene
x$onSceneDur<-difftime(x$DT.LVREF,x$DT.ARRIVE,units="secs")
## Scene to Hospital Travel Time
x$toHospitalTT<-difftime(x$DT.ARVREC,x$DT.LVREF,units="secs")
## Duration at Hospital
x$atHospitalDur<-difftime(x$DT.AVAILABLE,x$DT.ARVREC,units="secs")
## Time from arriving at scene to clear
x$arriveToClearTime<-difftime(x$DT.AVAILABLE,x$DT.ARRIVE,units="secs")
```

3 Estimate Travel Times to Hospital

Compute travel times between the call locations and destination hospital under each of the traffic scenarios. Save green light times and distances.

3.1 Best Guess Traffic Model

```
travelTime2bg<-rep(NA,nrow(x))
hosp.GL<-rep(NA,nrow(x))
hosp.Dist<-rep(NA,nrow(x))
for (i in 1:nrow(x)){
  if (!is.na(x$REC.LON[i])){
    api.out2<-mp_matrix(
      origins = cbind(x$REF.GPS.LON,x$REF.GPS.LAT)[i,],
      destinations = cbind(x$REC.LON,x$REC.LAT)[i,],
      mode="driving",
      traffic_model="best_guess",
      departure_time=as.POSIXct(x$DT.LVREF[i]),  ##as POSIXct
      ##departure_time=Sys.time() + as.difftime(4, units = "hours"),
      key = api.key,
      quiet = TRUE
    )
    travelTime2bg[i]<-mp_get_matrix(api.out2,
                                  value = "duration_in_traffic_s")
    hosp.GL[i]<-mp_get_matrix(api.out2,
                              value = "duration_s")
    hosp.Dist[i]<-mp_get_matrix(api.out2,
                              value = "distance_m")
    Sys.sleep(0.5)
  }
}
```

3.2 Pessimistic Traffic Model

```
travelTime2pe<-rep(NA,nrow(x))
for (i in 1:nrow(x)){
  if (!is.na(x$REC.LON[i])){
    api.out2<-mp_matrix(
      origins = cbind(x$REF.GPS.LON,x$REF.GPS.LAT)[i,],
      destinations = cbind(x$REC.LON,x$REC.LAT)[i,],
      mode="driving",
      traffic_model="pessimistic",
      departure_time=as.POSIXct(x$DT.LVREF[i]),  ##as POSIXct
      ##departure_time=Sys.time() + as.difftime(4, units = "hours"),
      key = api.key,
      quiet = TRUE
    )
    travelTime2pe[i]<-mp_get_matrix(api.out2,
                                  value = "duration_in_traffic_s")
    Sys.sleep(0.5)
  }
}
```

3.3 Optimistic Traffic Model

```
travelTime2op<-rep(NA,nrow(x))
for (i in 1:nrow(x)){
  if (!is.na(x$REC.LON[i])){
    api.out2<-mp_matrix(
      origins = cbind(x$REF.GPS.LON,x$REF.GPS.LAT)[i,],
      destinations = cbind(x$REC.LON,x$REC.LAT)[i,],
      mode="driving",
      traffic_model="optimistic",
      departure_time=as.POSIXct(x$DT.LVREF[i]),  ##as POSIXct
      ##departure_time=Sys.time() + as.difftime(4, units = "hours"),
      key = api.key,
      quiet = TRUE
    )
    travelTime2op[i]<-mp_get_matrix(api.out2,
                                  value = "duration_in_traffic_s")
    Sys.sleep(0.5)
  }
}
```

4 Assemble Travel Time Estimates

```
apiEstimates<-cbind(distance,durationGL,
                    travelTimePe,travelTimeBG,travelTimeOp,
                    hosp.Dist,hosp.GL,
                    eTT.Pe.Hosp=travelTime2pe,
                    eTT.BG.Hosp=travelTime2bg,
                    eTT.Op.Hosp=travelTime2op)
head(apiEstimates)

##      Dist.So Dist.Ce Dist.NN Dist.FN eTT.GL.So eTT.GL.Ce eTT.GL.NN eTT.GL.FN
## [1,]    9258    8434   17426   25709       561       411       827      1198
## [2,]    7048    2422   12212   20495       578       234       635      1007
## [3,]   10969    5301   12540   20823       759       366       752      1124
## [4,]    8781    5068   12307   20590       734       298       685      1056
## [5,]    8967    1516   12228   20511       770       191       656      1027
## [6,]    7800    2298   13267   21550       696       245       795      1166
##      eTT.Pe.So eTT.Pe.Ce eTT.Pe.NN eTT.Pe.FN eTT.BG.So eTT.BG.Ce eTT.BG.NN
## [1,]       616       440       859      1267       539       406       805
## [2,]       650       251       689      1076       549       220       633
## [3,]       918       384       796      1217       752       346       743
## [4,]      1026       363       784      1193       770       306       712
## [5,]       965       220       725      1123       766       181       670
## [6,]       890       250       963      1358       722       235       815
##      eTT.BG.FN eTT.Op.So eTT.Op.Ce eTT.Op.NN eTT.Op.FN hosp.Dist hosp.GL
## [1,]      1173       507       372       758      1097      3151     309
## [2,]       993       508       210       587       933      6060     443
## [3,]      1127       699       343       728      1090      1372     219
## [4,]      1085       679       283       671      1032        NA      NA
## [5,]      1036       728       187       642       994      6076     447
## [6,]      1186       668       243       753      1124      8416     557
##      eTT.Pe.Hosp eTT.BG.Hosp eTT.Op.Hosp
## [1,]         300         271         267
## [2,]         489         404         393
## [3,]         222         178         208
## [4,]          NA          NA          NA
## [5,]         557         438         414
## [6,]         661         553         531
```

4.1 Parsing the API–Computed Column Names

- **So** indicates the existing south EMS station
- **Ce** indicates the existing central EMS station
- **NN** indicates the proposed near–north EMS station
- **FN** indicates the proposed far–north EMS station
- **Dist** indicates distance travelled in meters.
- **eTT** indicates an estimated travel time.
- **GL** is the “green light” distance.
- **Pe** is the pessimistic travel time in traffic.
- **BG** is the best–guess travel time in traffic.
- **Op** is the optimistic travel time in traffic.
- **Hosp** is the hospital used, if such a trip is made.

4.2 Export Data Set for EDA

```r
x<-cbind(x,apiEstimates)
save(x,file="emsData.RData")
gc(); save.image()
```

```
##           used  (Mb) gc trigger  (Mb) limit (Mb) max used  (Mb)
## Ncells 2156672 115.2    3797542 202.9         NA  3797542 202.9
## Vcells 4065674  31.1   10350180  79.0      16384 10348928  79.0
```