Vehicular Messaging Simulation

Clustered Epidemiological Differential Equations

Jungyeol Kim

Contents

Contents

Contents

- Solving Differential Equations
  - Initial condition
  - Solution of differential equations
- Generating Figures
  - Fraction of overall informed vehicles over time
  - Fraction of informed and non-informed vehicles per cluster over time

Introduction

V2V technologies bridge two infrastructures: communication and transportation. These infrastructures are interconnected and interdependent. To capture this inter-dependence, which may vary in time and space, we propose a new methodology for modeling information propagation between V2V-enabled vehicles. The model is based on a continuous-time Markov chain which is shown to converge, under appropriate conditions, to a set of clustered epidemiological differential equations. The fraction of vehicles which have received a message, as a function of space and time may be obtained as a solution of these differential equations, which can be solved efficiently, independently of the number of vehicles.

Objective: Our goal is to model the spread of V2V messages and obatin the fraction of vehicles which have received a message in arbitrary transportation networks, as a function of space and time, using a set of *clustered epidemiological differential equations*.

Recommended citation

Please cite our paper below if you use this code.

J. Kim, S. Sarkar, S. S. Venkatesh, M. S. Ryerson, and D. Starobinski, 2019. An Epidemiological Diffusion Framework for Vehicular Messaging in General Transportation Networks. Transportation Research Part B: Methodological.

Requirements and installation

It requires R v3.2.0 (or above) and it is strongly recommended to download and install RStudio from https://rstudio.com/products/rstudio/.

Input files (required)

The following three files are required to use this software: `mobility-network.csv`, `communication-network.csv`, and `initial-condition.csv`

All the csv files should be located in your workspace.

File 1: `mobility-network.csv`

- The file contains four columns `clust_from, clust_to, routing_prob, lambda`. We will use the data set to import the given directed mobility network and mobility rates between clusters. Below is a quick description of all the features in the data set.

Feature	Description
clust_from	Origin of the directed edge in the mobility network
clust_to	Destination of the directed edge in the mobility network
routing_prob	Rounting probability that a vehicle in cluster i move to j; p_{ij}
lambda	Mobility constant; λ

- This file can be configured with different values depending on vehicle movement pattern and the (arbitrary) road topology you are considering.
- The table below is an example of this csv file. The mobility rate (set in this model) from cluster i to j is given by

$$\lambda_{ij}^I = \lambda_{ij}^S = p_{ij}\lambda.$$

The mobility constatnt λ can be determinded by the characteristic of each road segment. Speed limits are applied differently depending on the local characteristics of each city and the type of road, and as a result the average speed of the vehicles will depend on these characteristics.

from_clust	to_clust	routing_prob	lambda
1	2	0.5	0.05
1	36	0.5	0.05
2	3	0.5	0.05
2	41	0.5	0.05
3	4	0.5	0.05
3	46	0.5	0.05
4	5	0.5	0.05
4	51	0.5	0.05

File 2: `communication-network.csv`

- The file contains three columns `cluster_i; cluster_j; beta_ij`. We will use the data set to import the given undirected communication network and communication rates between clusters. Below is a quick description of all the features in the data set.

Feature	Description
cluster_i	One of the endpoints of the undirected edge in the communication network
cluster_j	The other endpoint of the same undirected edge in the communication network
beta_ij	Parameter for communications between cluster i and j; β_{ij}

- The table below is an example of this csv file. The communication parameter `beta_ij` is same as β_{ij} in the differential equations.

cluster_i	cluster_j	beta_ij
1	61	10
61	1	10
1	1	10
61	61	10

- Since this is an undirected network, this data should include both bidirectional edges between a pair of nodes. For example, as can be seen in the figure above, the edgelist must contain both {1,61} and {61,1}.
- Also, since vehicles in the same cluster can communicate with each other, {1,1} and {61,61} should be also included in the datat.

File 3: `initial-condition.csv`

- The file contains three columns `cluster; I_ini; S_ini`. We will use the data set to import the given initial condition. Below is a quick description of all the features in the data set.

Feature	Description
cluster	Cluster number
I_ini	Fraction of informed vehicles in each cluster at initial time; the number of informed vehicle per cluster divided by the total number of vehicles
S_ini	Fraction of non-informed vehicles in each cluster at initial time; the number of non-informed vehicle per cluster divided by the total number of vehicles

- Constraints: $I_{ini} \geq 0$, $S_{ini} \geq 0$, and $\sum_{i=1}^{J}(I_{ini} + S_{ini}) = 1$.
- The table below is an example of this csv file.

cluster	I_ini	S_ini
1	0.000833333	0.0075
2	0	0.008333333
3	0	0.008333333
4	0	0.008333333
5	0	0.008333333

Step-by-step instructions

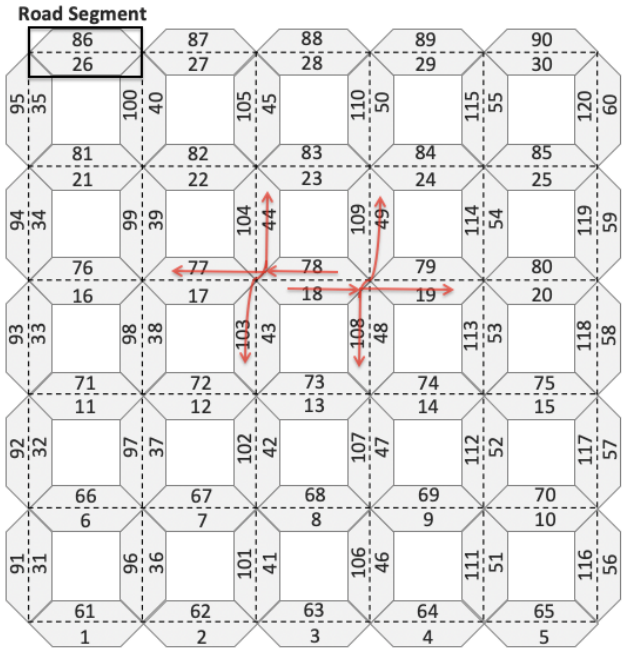
For first time users, here is the quick version of the instructions. Follow these steps:

1. Open RStudio.
2. Set working directory.
   - Go to `Tools > Change Working Dir...` menu (`Session > Set Working Directory` on a mac), or use `setwd()` function. eg., `setwd("/Users/jy/V2Vproject")`.
   - Check that the "home" directory for your project is the working directory of our current R process by typing `getwd()` in the Console.
3. Keep all the files (required input csv files, R scripts) associated with a project located together in the working directory.
4. Open the script file `V2Vsimulation.R` in the editor.

- Go to `File > Open > V2Vsimulation.R` or type `file.edit('V2Vsimulation.R')` in the Console.

5. Setting parameters values. (set values in lines 10,11,12 of the script file)
   - `total.clusters`: the total number of clusters
   - `sim.time`: end time of the time sequence
   - `step.size`: step, increment of time
   - e.g., if `step.size=1` and `sim.time=100`, the result will be generated every 1 time unit, from 0 to 100 units.
6. Check output
   - Type `source("V2Vsimulation.R", echo = TRUE)` in the Console.
7. Check output of this software
   - The output of this software consist of csv file representing the fraction of vehicles which have received a message, as a function of space and time.
   - The output files are saved in the the current working directory.
   - `fraction_of_informed_vehicles_per_cluster.csv`: the fraction of informed vehicles per clusters over time.
   - `fraction_of_non_formed_vehicles_per_cluster.csv`: the fraction of non-informed vehicles per clusters over time.
   - Most users will want to create graphs using the output files. Refer to the last section of this manual.

Case study: Grid road topology



As an example, we consider the grid road topology with six avenues and streets. In this network, we assume that all roads are two-way and allow vehicles to move in both directions, and a road segment consists of two clusters corresponding to the opposite directional roads. Since two clusters on the same road segment are sufficiently close to each other, we also assume that the vehicles located in these can communicate; there is an undirected edge between the two clusters on the same road segment.

Under these settings, we can create three aforementioned files that are essential for the vehicular messaging simulation using clustered epidemiological differential equations. The files I have already created under these assumptions can be downloaded from the following GitHub repository: https://github.com/jungyeol-kim/V2X-simulations.

However, for any road topologies, vehicle movement patterns, and communication environment, you can use the software code below to automatically generate a set of clustered epidemiological differential equations corresponding to the given conditions, and perform V2V message propagation simulation using the generated differential equations.

The following sections describe the software code structure in detail using the sample input files provided based on the above road topology.

Mobility and Communication Network

Importing edge list of mobility network

We import preset edge list of directed mobility network and corresponding mobility parameter λ_{ij}.

```
setwd("/Users/jy/V2Vproject") # set working directory
mobility_network<-read.csv("mobility-network.csv", header=T, as.is=T)
mobility_network$lambda_from_to<-mobility_network$routing_prob*mobility_network$lambda
```

```
head(mobility_network) #View(mobility_network)
```

```
##   from_clust to_clust routing_prob lambda lambda_from_to
## 1          1        2          0.5   0.05          0.025
## 2          1       36          0.5   0.05          0.025
## 3          2        3          0.5   0.05          0.025
## 4          2       41          0.5   0.05          0.025
## 5          3        4          0.5   0.05          0.025
## 6          3       46          0.5   0.05          0.025
```

Importing edge list of communication network

We import preset edge list of undirected communication network and corresponding communication parameter β_{ij}.

```
communication_network<-read.csv("communication-network.csv", header=T, as.is=T)
communication_network<-communication_network[order(communication_network$cluster_i,
                                                   communication_network$cluster_j),]
rownames(communication_network) <- NULL # reset row names
```

```
head(communication_network) #View(communication_network)
```

```
##   cluster_i cluster_j beta_ij
## 1         1         1      10
## 2         1        61      10
## 3         2         2      10
## 4         2        62      10
## 5         3         3      10
## 6         3        63      10
```

Defining a neighborhood of a cluster

We define a neighborhood of each cluster for both directed mobility network and undirected communication network.

```
total.clusters<-120
mob.edge.out<-vector(mode='list',length=total.clusters)
# outgoing edges from node i in the mobility network
mob.edge.out.rate<-vector(mode='list',length=total.clusters)
# mobility parameter of outgoing edges from node i in the mobility network
mob.edge.in<-vector(mode='list',length=total.clusters)
# incoming edges to node i in the mobility network
mob.edge.in.rate<-vector(mode='list',length=total.clusters)
# mobility parameter of incoming edges from node i in the mobility network
comm.edge<-vector(mode='list',length=total.clusters)
# undirected edges from or to node i in the communication network
comm.edge.rate<-vector(mode='list',length=total.clusters)
# communication parameter of undirected edges from or to node i in the communication network
for (i in 1:total.clusters) {
  temp1<-mobility_network[mobility_network$from_clust==i,]
  temp2<-communication_network[communication_network$cluster_i==i,]
  mob.edge.out[[i]]<-temp1$to_clust
  mob.edge.out.rate[[i]]<-temp1$lambda_from_to
  comm.edge[[i]]<-temp2$cluster_j
  comm.edge.rate[[i]]<-temp2$beta_ij
  for(j in 1:length(mob.edge.out[[i]])){
    mob.edge.in[[mob.edge.out[[i]][j]]]<-c(mob.edge.in[[mob.edge.out[[i]][j]]],i)
    mob.edge.in.rate[[mob.edge.out[[i]][j]]]<-c(mob.edge.in.rate[[mob.edge.out[[i]][j]]],
                                                mob.edge.out.rate[[i]][j])
  }
}
```

Quick exploration of the result

```
mob.edge.out[[1]] #View(end point of outgoing edges from a given cluster 1)
```

```
## [1]  2 36
```

```
mob.edge.out.rate[[1]] #View(mobility rates corresponding to outgoing edges from a given cluster 1)
```

```
## [1] 0.025 0.025
```

- Endpoints of outgoing edges for a given cluster (vertex) 1 are cluster 2 and 36.
- Mobility rate from cluster 1 to 2 is 0.25, and mobility rate from 1 to 36 is also 0.25.

```
mob.edge.in[[7]] #View(mobility_network)
```

```
## [1]  6 36 97
```

```
mob.edge.in.rate[[7]] #View(communication_network)
```

```
## [1] 0.01666667 0.01666667 0.01666667
```

- Incoming edges to cluster 7 come from cluster 6, 36, and 97.
- Mobility rate from cluster 6 to 7 is 0.01666667, mobility rate from 36 to 7 is 0.25, and mobility rate from 97 to 7 is 0.25.

```
comm.edge[[25]] #View(mobility_network)
```

```
## [1] 25 85
```

```
comm.edge.rate[[25]] #View(communication_network)
```

```
## [1] 10 10
```

- Vehicles in cluster 25 can communicate with other vehicles in the same cluster 25, and also communicate with vehicles in cluster 85. (intra- and inter-cluster communication)
- Intra-cluster communication parameter $\beta_{25,25}$ is 10, and inter-cluster communication parameter $\beta_{25,85}$=$\beta_{85,25}$ corresponding to communication between cluster 10 and 70 is also 10.

Generation of Differential Equations

Recall that under the conditions of our model, for a given choice of initial conditions $\big(\mathbf{I}(0), \mathbf{S}(0)\big)$, the time-evolution, $\big(\mathbf{I}(t), \mathbf{S}(t)\big)$, of the distribution of the asymptotic fraction of informed and non-informed vehicles across clusters is governed by the following system of ordinary differential equations:

$$\dot{I}_j(t) = -\sum_{k\neq j}^{J} \lambda^I_{jk}\left(\mathbf{I},\mathbf{S}\right)\cdot I_j + \sum_{k=1}^{J}\beta_{kj}\cdot I_k\cdot S_j + \sum_{k\neq j}^{J}\lambda^I_{kj}\left(\mathbf{I},\mathbf{S}\right)\cdot I_k \qquad (j=1,2,\ldots,J),$$

$$\dot{S}_j(t) = -\sum_{k\neq j}^{J} \lambda^S_{jk}\left(\mathbf{I},\mathbf{S}\right)\cdot S_j - \sum_{k=1}^{J}\beta_{kj}\cdot I_k\cdot S_j + \sum_{k\neq j}^{J}\lambda^S_{kj}\left(\mathbf{I},\mathbf{S}\right)\cdot S_k \qquad (j=1,2,\ldots,J).$$

We now create a set of *clustered epidemiological differential equations* for the given mobility and communicatoin networks. The number of variables and the total number of differential equations are $2J$ each (recall that J is the total number of clusters). The $2J$-dimensional vector $(y_1, y_2, ..., y_J; y_{J+1}, y_{J+2}, ..., y_{2J}) = (I_1, I_2, ..., I_J; S_1, S_2, ..., S_J)$ represent the instantaneous state of the system, semicolon and extra spacing have been added merely for visual separation of informed and non-informed vehicular counts in the various clusters.

We create the first summation term on right hand side The first summation terms on the RHS of the j-th equation ($\dot{I}_j$) and the $J+j$-th equation ($\dot{S}_j$) is related to the outgoing mobility from cluster j.

```
mob.out.text<-vector(mode='character',length=total.clusters*2)
for(i in 1:total.clusters){
  mob.out.text.temp.1<-c();  mob.out.text.temp.2<-c();
  mob.out.text.temp.3<-c();  mob.out.text.temp.4<-c()
  for(j in 1:length(mob.edge.out[[i]])){
    mob.out.text.temp.1<-paste("- ",mob.edge.out.rate[[i]][j],"*y[",i,"]", sep="")
    mob.out.text.temp.2<-paste(mob.out.text.temp.2,mob.out.text.temp.1,sep=" ")
    mob.out.text.temp.3<-paste("- ",mob.edge.out.rate[[i]][j],"*y[",
                               total.clusters+i,"]", sep="")
    mob.out.text.temp.4<-paste(mob.out.text.temp.4,mob.out.text.temp.3,sep=" ")
  }
  mob.out.text[i]<-mob.out.text.temp.2
  mob.out.text[total.clusters+i]<-mob.out.text.temp.4
}
```

```
head(mob.out.text) #View(the first summation term on the RHS of each differential equation)
```

```
## [1] " - 0.025*y[1] - 0.025*y[1]"
## [2] " - 0.025*y[2] - 0.025*y[2]"
## [3] " - 0.025*y[3] - 0.025*y[3]"
## [4] " - 0.025*y[4] - 0.025*y[4]"
## [5] " - 0.05*y[5]"
## [6] " - 0.01666666665*y[6] - 0.01666666665*y[6] - 0.01666666665*y[6]"
```

We create the second summation term on right hand side The second summation terms on the RHS of the j-th equation ($\dot{I}_j$) and the $J+j$-th equation ($\dot{S}_j$) is related to the intra- and inter-communication in cluster j.

```
comm.text<-vector(mode='character',length=total.clusters*2)
for(i in 1:total.clusters){
  comm.text.temp.1<-c();  comm.text.temp.2<-c();
  comm.text.temp.3<-c();  comm.text.temp.4<-c()
  for(j in 1:length(comm.edge[[i]])){
    comm.text.temp.1<-paste(comm.edge.rate[[i]][j],"*y[",comm.edge[[i]][j],"]*y[",
```

```
                            total.clusters+i,"]", sep="")
    comm.text.temp.2<-paste(comm.text.temp.2," + ",comm.text.temp.1,sep="")
    comm.text.temp.3<-paste(comm.edge.rate[[i]][j],"*y[",comm.edge[[i]][j],"]*y[",
                            total.clusters+i,"]", sep="")
    comm.text.temp.4<-paste(comm.text.temp.4," - ",comm.text.temp.3,sep="")
  }
  comm.text[i]<-comm.text.temp.2
  comm.text[total.clusters+i]<-comm.text.temp.4
}
```

```
head(comm.text) #View(the second summation term on the RHS of each differential equation)
```

```
## [1] " + 10*y[1]*y[121] + 10*y[61]*y[121]" " + 10*y[2]*y[122] + 10*y[62]*y[122]"
## [3] " + 10*y[3]*y[123] + 10*y[63]*y[123]" " + 10*y[4]*y[124] + 10*y[64]*y[124]"
## [5] " + 10*y[5]*y[125] + 10*y[65]*y[125]" " + 10*y[6]*y[126] + 10*y[66]*y[126]"
```

We create the third summation term on right hand side The third summation terms on the RHS of the j-th equation ($\dot{I}_j$) and the $J+j$-th equation ($\dot{S}_j$) is related to the incoming mobility to cluster j

```
mob.in.text<-vector(mode='character',length=total.clusters*2)
for(i in 1:total.clusters){
  mob.in.text.temp.1<-c();  mob.in.text.temp.2<-c();
  mob.in.text.temp.3<-c();  mob.in.text.temp.4<-c()
  for(j in 1:length(mob.edge.in[[i]])){
    mob.in.text.temp.1<-paste(mob.edge.in.rate[[i]][j],"*y[",
                              mob.edge.in[[i]][j],"]", sep="")
    mob.in.text.temp.2<-paste(mob.in.text.temp.2," + ",mob.in.text.temp.1,sep=" ")
    mob.in.text.temp.3<-paste(mob.edge.in.rate[[i]][j],"*y[",
                              total.clusters+mob.edge.in[[i]][j],"]", sep="")
    mob.in.text.temp.4<-paste(mob.in.text.temp.4," + ",mob.in.text.temp.3,sep=" ")
  }
  mob.in.text[i]<-mob.in.text.temp.2
  mob.in.text[total.clusters+i]<-mob.in.text.temp.4
}
```

```
head(mob.in.text) #View(the third summation term on the RHS of each differential equation)
```

```
## [1] "  +  0.05*y[91]"                  "  +  0.025*y[1]  +  0.025*y[96]"
## [3] "  +  0.025*y[2]  +  0.025*y[101]" "  +  0.025*y[3]  +  0.025*y[106]"
## [5] "  +  0.025*y[4]  +  0.025*y[111]" "  +  0.025*y[31]  +  0.025*y[92]"
```

We now combine the all terms to create the complete set of differential equations

```
dy<-vector(mode='character',length=total.clusters*2)
for (i in 1:(total.clusters*2)) {
   dy[i]<-paste("dy",i," <- ",mob.out.text[i],mob.in.text[i],comm.text[i],sep="")
}
```

```
head(dy) # View(complete set of differential equations)
```

```
## [1] "dy1 <-  - 0.025*y[1] - 0.025*y[1]  +  0.05*y[91] + 10*y[1]*y[121] + 10*y[61]*y[121]"
## [2] "dy2 <-  - 0.025*y[2] - 0.025*y[2]  +  0.025*y[1]  +  0.025*y[96] + 10*y[2]*y[122] + 10*y[62]*y[
## [3] "dy3 <-  - 0.025*y[3] - 0.025*y[3]  +  0.025*y[2]  +  0.025*y[101] + 10*y[3]*y[123] + 10*y[63]*y
## [4] "dy4 <-  - 0.025*y[4] - 0.025*y[4]  +  0.025*y[3]  +  0.025*y[106] + 10*y[4]*y[124] + 10*y[64]*y
## [5] "dy5 <-  - 0.05*y[5]  +  0.025*y[4]  +  0.025*y[111] + 10*y[5]*y[125] + 10*y[65]*y[125]"
## [6] "dy6 <-  - 0.01666666665*y[6] - 0.01666666665*y[6] - 0.01666666665*y[6]  +  0.025*y[31]  +  0.025
```

```
dy_name<-c()
for (i in 1:(total.clusters*2)) {
  if(i==1){dy_name<-paste(dy_name,"list(c(dy1",sep="")}
  else if(i==(total.clusters*2)){dy_name<-paste(dy_name,",dy",total.clusters*2,"))}",sep="")}
  else{dy_name<-paste(dy_name,",",paste("dy",i,sep=""),sep="")}
}
set.diff.eqn<-c("f <- function(t, y, parms) {",dy,dy_name)
write(set.diff.eqn, file = "set_diff_eqn.R")
source("set_diff_eqn.R")
```

Solving Differential Equations

Initial condition

We import preset initial conditions.

```
# import preset initial condition
initial_condition<-read.csv("initial-condition.csv", header=T, as.is=T)
# initial condition: 2J-dimensional vector
yini<-c(initial_condition$I_ini,initial_condition$S_ini)
```

The $2J$-dimensional vector `y_ini` represent the state of the system at initial time.

Solution of differential equations

We create a function to encode the set of differential equations in a form suitable for use as the `func` argument to `ode` (numerical methods provided by the `deSolve` package).

Before we run, we need to set what are the timestamps used. `times` denote time sequence for which output is wanted.

The example below shows that the result will be generated every `step.size=1` time unit, from `0` to `sim.time=100` units.

```
sim.time<-100
step.size<-1
times <- seq(from = 0, to = sim.time, by = step.size) # output wanted at these time intervals
print(times)
```

```
##   [1]   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
##  [19]  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35
##  [37]  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53
##  [55]  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71
##  [73]  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89
##  [91]  90  91  92  93  94  95  96  97  98  99 100
```

We then compute the fracton of informed vehicles over space and time by applying all into the `ODE` solver:

```
out <- ode(times = times, y=yini, func = f, parms = NULL) # numerically solve the set of
                                                          # differential equations
solution<-out[,-1]
rownames(solution)<-times

# fraction of informed vehicles per cluster
frac.inf.clust<-solution[,1:total.clusters]
```

```r
# fraction of non-informed vehicles per cluster
frac.non.inf.clust<-solution[,(1+total.clusters):(2*total.clusters)]
colnames(frac.non.inf.clust)<-1:total.clusters

write.table(frac.inf.clust, file = "fraction_of_informed_vehicles_per_cluster.csv",
            row.names=TRUE,col.names=TRUE, sep=",") # export a matrix to a file.
write.table(frac.non.inf.clust, file = "fraction_of_non_informed_vehicles_per_cluster.csv",
            row.names=TRUE,col.names=TRUE, sep=",") # export a matrix to a file.
```

Row names and Column names of `frac.inf.clust` and `frac.non.inf.clust` represent time and cluster respectively. For example, `frac.inf.clust[rownames(frac.inf.clust)==10,25]` (`frac.non.inf.clust[rownames(frac.non.inf.clust)==10,25]`) gives the fraction of informed (non-informed) vehicles at $t = 10$ in cluster 25. Naturally, multiplying the matrix `frac.inf.clust` (`frac.non.inf.clust`) by the total number of vehicles yields the number of informed (non-informed) vehicles in a given cluster at a given time.

```r
head(frac.inf.clust) # View(fraction of informed vehicles at a given time in each cluster)
```

```
##              1            2            3            4            5            6
## 0 0.0008333330 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## 1 0.0008570283 2.153502e-05 2.742762e-07 3.458794e-09 4.070297e-11 3.065389e-08
## 2 0.0008866241 4.467211e-05 1.129462e-06 2.138115e-08 3.472434e-10 1.887809e-07
## 3 0.0009224916 6.983034e-05 2.639884e-06 7.020836e-08 1.509739e-09 6.169624e-07
## 4 0.0009649812 9.741283e-05 4.886947e-06 1.667595e-07 4.435826e-09 1.458011e-06
## 5 0.0010145891 1.280043e-04 8.000834e-06 3.369939e-07 1.088772e-08 2.927579e-06
##              7            8            9           10           11           12
## 0 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## 1 1.831403e-07 3.843117e-09 5.728566e-11 3.262923e-15 3.587846e-10 1.537265e-09
## 2 7.554370e-07 2.377566e-08 4.888356e-10 5.218208e-12 3.050074e-09 9.538702e-09
## 3 1.770591e-06 7.815003e-08 2.126314e-09 3.916361e-11 1.321024e-08 3.147272e-08
## 4 3.289197e-06 1.858284e-07 6.250678e-09 1.508577e-10 3.866833e-08 7.514784e-08
## 5 5.410205e-06 3.760688e-07 1.535273e-08 4.579755e-10 9.448173e-08 1.528957e-07
##             13           14           15           16           17           18
## 0 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## 1 4.221049e-11 3.313121e-15 1.642140e-19 1.782041e-14 1.206014e-11 1.807157e-15
## 2 3.604032e-10 5.298492e-12 5.497413e-14 2.844247e-11 1.032339e-10 2.890101e-12
## 3 1.569299e-09 3.977367e-11 6.720586e-13 2.127203e-10 4.515584e-10 2.172475e-11
## 4 4.618798e-09 1.532387e-10 3.459331e-12 8.165112e-10 1.336009e-09 8.382470e-11
## 5 1.136243e-08 4.653261e-10 1.329592e-11 2.468061e-09 3.309011e-09 2.550324e-10
##             19           20           21           22           23           24
## 0 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## 1 1.278613e-19 3.244791e-24 6.650014e-19 4.015905e-16 5.515586e-20 2.033217e-24
## 2 4.280428e-14 3.254178e-18 2.226214e-13 6.422592e-13 1.846459e-14 2.039098e-18
## 3 5.233526e-13 6.935202e-15 2.712305e-12 4.858482e-12 2.259794e-13 4.345668e-15
## 4 2.694521e-12 5.560318e-14 1.391470e-11 1.887354e-11 1.165454e-12 3.485216e-14
## 5 1.035972e-11 2.834071e-13 5.326576e-11 5.792206e-11 4.491357e-12 1.777117e-13
##             25           26           27           28           29           30
## 0 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## 1 0.000000e+00 2.182922e-23 1.504251e-20 1.169796e-24 0.000000e+00 0.000000e+00
## 2 1.320640e-22 2.189216e-17 5.035799e-15 1.173180e-18 1.202166e-22 4.201052e-27
## 3 4.427394e-17 4.658482e-14 6.205064e-14 2.500277e-15 4.030214e-17 4.219934e-21
## 4 6.296406e-16 3.721500e-13 3.238040e-13 2.011311e-14 5.733459e-16 8.839941e-18
## 5 4.560960e-15 1.888573e-12 1.267853e-12 1.029676e-13 4.156429e-15 1.048587e-16
##             31           32           33           34           35           36
```

```
## 0 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## 1 1.815881e-06 2.527363e-08 3.165741e-10 1.641484e-14 6.299022e-19 2.153502e-05
## 2 7.440270e-06 1.556124e-07 2.690232e-09 2.619463e-11 2.108712e-13 4.467212e-05
## 3 1.726134e-05 5.084321e-07 1.164778e-08 1.958548e-10 2.568573e-12 6.983046e-05
## 4 3.167600e-05 1.201236e-06 3.408468e-08 7.515669e-10 1.317408e-11 9.741330e-05
## 5 5.128501e-05 2.411345e-06 8.325398e-08 2.270989e-09 5.041520e-11 1.280057e-04
##             37           38           39           40           41           42
## 0 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## 1 1.830680e-07 1.537261e-09 1.206014e-11 4.015905e-16 2.743184e-07 3.843115e-09
## 2 7.548209e-07 9.532279e-09 1.031869e-10 6.422572e-13 1.129821e-06 2.377244e-08
## 3 1.767922e-06 3.142466e-08 4.509846e-10 4.854196e-12 2.641444e-06 7.812604e-08
## 4 3.281385e-06 7.496329e-08 1.333062e-09 1.883929e-11 4.891517e-06 1.857364e-07
## 5 5.391130e-06 1.523377e-07 3.297720e-09 5.774828e-11 8.012010e-06 3.757912e-07
##             43           44           45           46           47           48
## 0 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## 1 4.221049e-11 1.807157e-15 5.515586e-20 3.458798e-09 5.728566e-11 3.313121e-15
## 2 3.603528e-10 2.890098e-12 1.846459e-14 2.138676e-08 4.888373e-10 5.298491e-12
## 3 1.568684e-09 2.171785e-11 2.259401e-13 7.025042e-08 2.126335e-09 3.977022e-11
## 4 4.615644e-09 8.376948e-11 1.164897e-12 1.669211e-07 6.250786e-09 1.532111e-10
## 5 1.135037e-08 2.547519e-10 4.487341e-12 3.374825e-07 1.535316e-08 4.651861e-10
##             49           50           51           52           53           54
## 0 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## 1 1.278613e-19 2.033217e-24 4.070297e-11 3.262923e-15 1.642140e-19 3.244791e-24
## 2 4.280428e-14 2.039098e-18 3.472992e-10 5.218205e-12 5.497413e-14 3.254178e-18
## 3 5.233120e-13 4.345667e-15 1.510420e-09 3.915701e-11 6.719527e-13 6.935192e-15
## 4 2.693944e-12 3.484867e-14 4.439320e-09 1.508049e-10 3.457828e-12 5.558229e-14
## 5 1.035556e-11 1.776706e-13 1.090109e-08 4.577077e-10 1.328507e-11 2.831602e-13
##             55           56           57           58           59           60
## 0 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## 1 0.000000e+00 4.066103e-15 2.745258e-19 7.311224e-24 0.000000e+00 0.000000e+00
## 2 1.320640e-22 6.502672e-12 9.190331e-14 7.332374e-18 3.961919e-22 8.796679e-27
## 3 4.427394e-17 4.876559e-11 1.122932e-12 1.562646e-14 1.328218e-16 8.836216e-21
## 4 6.295244e-16 1.876858e-10 5.774844e-12 1.251485e-13 1.887660e-15 1.851004e-17
## 5 4.558121e-15 5.691548e-10 2.216754e-11 6.369514e-13 1.365205e-14 2.191876e-16
##             61           62           63           64           65           66
## 0 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## 1 7.131157e-05 9.124209e-07 1.152921e-08 1.356765e-10 6.776839e-15 1.837915e-07
## 2 1.466226e-04 3.749033e-06 7.112955e-08 1.156364e-09 1.083775e-11 7.609573e-07
## 3 2.264721e-04 8.730399e-06 2.329732e-07 5.018877e-09 8.119620e-11 1.794405e-06
## 4 3.113426e-04 1.608604e-05 5.518143e-07 1.471636e-08 3.121723e-10 3.358600e-06
## 5 4.018877e-04 2.616890e-05 1.111061e-06 3.602558e-08 9.453568e-10 5.578838e-06
##             67           68           69           70           71           72
## 0 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## 1 9.992038e-09 1.778870e-10 1.089314e-14 5.597066e-19 1.537301e-09 9.045104e-11
## 2 6.172226e-08 1.516572e-09 1.742068e-11 1.873736e-13 9.595709e-09 7.719405e-10
## 3 2.024826e-07 6.585788e-09 1.305156e-10 2.288371e-12 3.189778e-08 3.358472e-09
## 4 4.804336e-07 1.932287e-08 5.017895e-10 1.175853e-11 7.677414e-08 9.875231e-09
## 5 9.695412e-07 4.734072e-08 1.519576e-09 4.508504e-11 1.577912e-07 2.426281e-08
##             73           74           75           76           77           78
## 0 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## 1 8.935388e-15 4.951492e-19 1.369636e-23 1.206014e-11 3.413519e-15 3.083714e-19
## 2 1.428981e-11 1.657617e-13 1.373598e-17 1.037157e-10 5.459077e-12 1.032339e-13
## 3 1.071369e-10 2.024752e-12 2.927351e-14 4.574175e-10 4.102966e-11 1.261608e-12
## 4 4.122287e-10 1.040689e-11 2.343078e-13 1.365993e-09 1.582865e-10 6.490091e-12
```

```
## 5 1.249628e-09 3.991804e-11 1.191602e-12 3.423410e-09 4.814714e-10 2.492406e-11
##             79           80           81           82           83           84
## 0 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## 1 8.926656e-24 0.000000e+00 4.015905e-16 9.526921e-20 4.484217e-24 0.000000e+00
## 2 8.952480e-18 6.803560e-22 6.422846e-13 3.189339e-14 4.497189e-18 3.355610e-22
## 3 1.907915e-14 2.280867e-16 4.912499e-12 3.904359e-13 9.584262e-15 1.124955e-16
## 4 1.527709e-13 3.240923e-15 1.930430e-11 2.014575e-12 7.682480e-14 1.598906e-15
## 5 7.773380e-13 2.342821e-14 6.010314e-11 7.768643e-12 3.914544e-13 1.156581e-14
##             85           86           87           88           89           90
## 0 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## 1 0.000000e+00 1.504251e-20 1.754694e-24 0.000000e+00 0.000000e+00 0.000000e+00
## 2 1.392807e-26 5.035801e-15 1.759770e-18 2.226620e-22 8.872112e-27 0.000000e+00
## 3 1.399068e-20 6.264001e-14 3.750440e-15 7.464657e-17 8.911988e-21 1.186747e-24
## 4 2.930752e-17 3.321510e-13 3.022088e-14 1.061923e-15 1.866881e-17 2.996425e-19
## 5 3.469172e-16 1.327939e-12 1.550541e-13 7.698084e-15 2.212120e-16 7.326893e-18
##             91           92           93           94           95           96
## 0 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## 1 7.886381e-08 1.013036e-09 5.331057e-14 2.064575e-18 3.549077e-23 9.125114e-07
## 2 4.845142e-07 8.599272e-09 8.506716e-11 6.911537e-13 3.559308e-17 3.749798e-06
## 3 1.578565e-06 3.716222e-08 6.350701e-10 8.412619e-12 7.573624e-14 8.733702e-06
## 4 3.717777e-06 1.085137e-07 2.432996e-09 4.309428e-11 6.046011e-13 1.609567e-05
## 5 7.431826e-06 2.643079e-07 7.336036e-09 1.646334e-10 3.065346e-12 2.619235e-05
##             97           98           99          100          101          102
## 0 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## 1 7.686190e-09 6.030069e-11 2.007952e-15 5.014169e-20 1.306645e-08 1.628119e-10
## 2 4.748859e-08 5.148216e-10 3.211239e-12 1.678599e-14 8.063097e-08 1.388138e-09
## 3 1.558299e-07 2.241347e-09 2.417029e-11 2.059713e-13 2.641669e-07 6.028752e-09
## 4 3.698483e-07 6.595598e-09 9.339051e-11 1.067086e-12 6.258896e-07 1.769088e-08
## 5 7.466584e-07 1.622146e-08 2.846415e-10 4.137650e-12 1.260709e-06 4.335000e-08
##            103          104          105          106          107          108
## 0 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## 1 7.429423e-15 2.356659e-19 3.314421e-24 1.658269e-10 1.094334e-14 4.587965e-19
## 2 1.188142e-11 7.889416e-14 3.324010e-18 1.413525e-09 1.750097e-11 1.535918e-13
## 3 8.909146e-11 9.642758e-13 7.084047e-15 6.136510e-09 1.311323e-10 1.876152e-12
## 4 3.428423e-10 4.961602e-12 5.683884e-14 1.799859e-08 5.042233e-10 9.643578e-12
## 5 1.039474e-09 1.905986e-11 2.899884e-13 4.407681e-08 1.527194e-09 3.699269e-11
##            109          110          111          112          113          114
## 0 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## 1 7.715082e-24 0.000000e+00 8.684394e-15 4.901350e-19 1.054209e-23 0.000000e+00
## 2 7.737400e-18 2.982764e-22 1.388837e-11 1.640831e-13 1.057259e-17 4.568229e-22
## 3 1.648963e-14 9.999603e-17 1.040474e-10 2.004029e-12 2.253183e-14 1.531481e-16
## 4 1.320408e-13 1.421528e-15 4.000102e-10 1.029840e-11 1.803605e-13 2.176236e-15
## 5 6.718894e-13 1.028751e-14 1.211289e-09 3.949146e-11 9.173385e-13 1.573392e-14
##            115          116          117          118          119          120
## 0 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## 1 0.000000e+00 6.299050e-19 1.798561e-23 0.000000e+00 0.000000e+00 0.000000e+00
## 2 9.332448e-27 2.108740e-13 1.803764e-17 1.036476e-21 2.419087e-26 0.000000e+00
## 3 9.374393e-21 2.574852e-12 3.844094e-14 3.474745e-16 2.429959e-20 1.416890e-24
## 4 1.963742e-17 1.322581e-11 3.075504e-13 4.936081e-15 5.090242e-17 3.577514e-19
## 5 2.325661e-16 5.068578e-11 1.563179e-12 3.566092e-14 6.021679e-16 8.746445e-18
```

We export the results to csv files in the current workspace.

```
write.table(frac.inf.clust, file = "fraction_of_informed_vehicles_per_cluster.csv",
            row.names=TRUE,col.names=TRUE, sep=",") # export a matrix to a file.
```

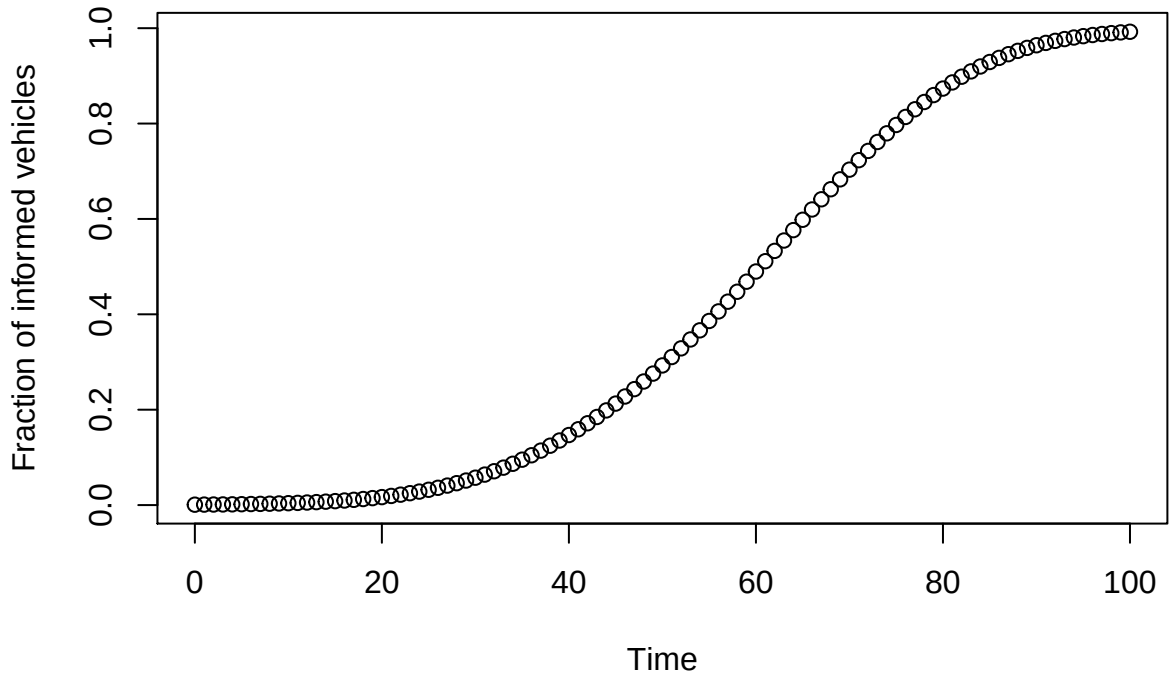
```
write.table(frac.non.inf.clust, file = "fraction_of_non_informed_vehicles_per_cluster.csv",
            row.names=TRUE,col.names=TRUE, sep=",") # export a matrix to a file.
```

Generating Figures

Fraction of overall informed vehicles over time

To study the degree of information propagation, we plot the fraction of overall vehicles that are informed at time t. A value of 1 on the y axis indicates that all vehicles in the system receive messages via V2V communication.

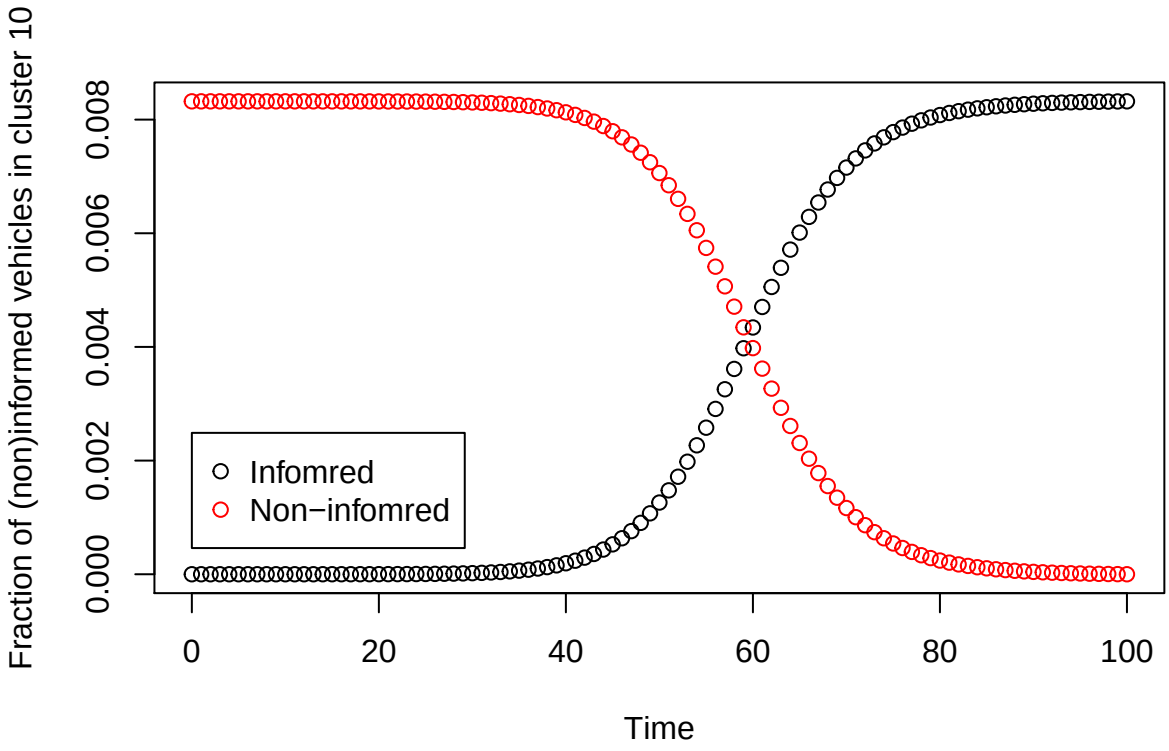
```
frac.inf.veh<-rowSums(solution[,1:total.clusters]) # fraction of overall vehicles
                                                   # that are informed over time.
plot(times,frac.inf.veh, xlab="Time", ylab="Fraction of informed vehicles")
```



Fraction of informed and non-informed vehicles over time per cluster

```
cluster.specific<-10 # determine the specific cluster of interest.
# fraction of informed vehicles over time in the particular cluster.
frac.inf.veh.clust<-frac.inf.clust[,cluster.specific]
# fraction of non-informed vehicles over time in the particular cluster.
frac.non.inf.veh.clust<-frac.non.inf.clust[,cluster.specific]

plot(times, frac.inf.veh.clust, xlab="Time", col="black",
     ylab=paste("Fraction of (non)informed vehicles in cluster ",cluster.specific,sep = ""))
par(new=T)
plot(times, frac.non.inf.veh.clust, xlab='', ylab='', col="red", axes=F)
par(new=F)
legend(0, 0.0025, legend=c("Infomred","Non-infomred"), pch = c(1, 1), col=c("black","red"))
```


Fraction of (non)informed vehicles in cluster 10
0.000
0.002
0.004
0.006
0.008
0
20
40
60
80
100
Time
Infomred
Non−infomred