Bearmod_Documentation

COVId19 TaskForce

4/1/2020

Descripción del código

Inicialización de la población

```
HPop = InitiatePop(pat_locator,
                   patnInf,
                   patnExp)
```

Inicialización de la población. Toma como parámetro una tabla con todas las regiones (*patches*), y el número inicial de infectados y expuestos por región.

Input

- `pat_locator` : Tabla de regiones (*patches*). Cada fila representa una región, de de cada una se incluye su identificador (columna `patIDs`), su nombre (columna `patNames`) y su población (columna `pop`).

	patNames	patIDs	pop
1	1	1	100
2	2	2	100
3	3	3	100
4	4	4	100
5	5	5	100

- `patnInf` : Tupla de **infectados** iniciales por región. Cada elemento de la tupla representa una región.

  `50 0 0 0 0`

- `patnExp` : Tupla de **expuestos** iniciales por región. Cada elemento de la tupla representa una región.

  `0 0 0 0 0`

Output

Como salida produce la población inicial, con las siguientes entradas:

```
$nInitialInf
[1] 50  0  0  0  0

$nInitialExp
[1] 0 0 0 0 0

$nInf
[1] 50  0  0  0  0

$nExp
[1] 0 0 0 0 0
```

```
$nRec
[1] 0 0 0 0 0

$nTotal
[1] 100 100 100 100 100

$names
[1] 1 2 3 4 5

$IDs
[1] 1 2 3 4 5

$relativeInf
[1] 1 1 1 1 1

$nRecoveredToday
[1] 0 0 0 0 0

$nInfectedToday
[1] 0 0 0 0 0

$nExposedToday
[1] 0 0 0 0 0

$nInfMovedToday
[1] 0 0 0 0 0

$controlled
[1] 0 0 0 0 0
```

Ejecución de la simulación

```
HPop_update = runSim(HPop,
                     pat_locator,
                     relative_move_data,
                     movement_data,
                     input_dates,
                     recover_df,
                     exposerate,
                     exposepd,
                     exposed_pop_inf_prop = 0,
                     TSinday = 1,
                     prob_move_per_TS=0)
```

Ejecuta una simulación a partir de la población inicial, la tabla de movilidad y las tasas de recuperación y exposición.

Input

- `HPop` : Población inicial.
- `pat_locator`: Tabla de regiones (*patches*)

```
  patNames patIDs pop
1        1      1 100
2        2      2 100
3        3      3 100
4        4      4 100
5        5      5 100
```

- `relative_move_data` : Permite reducir una fracción de la población que se mueve, respecto a los valores absolutos de la siguiente tabla. Se indica región de origen (`from`), región destino (`to`), y el factor por el que se multiplicará la movilidad entre dichas regiones (`relative_move`),

```
        date from to relative_move
1 2020-05-01    1  2             0
2 2020-05-01    1  3             0
3 2020-05-01    1  4             0
4 2020-05-01    1  5             0
5 2020-05-01    5  1             0
```

- `movement_data` : Tabla de movilidad. Se incluyen los movimiento que se aplicarán en cada fecha. Se indica región de origen (`fr_pat`), región destino (`to_pat`), personas que se desplazan (`movers`), población en el origen (`fr_users`) y en el destino (`to_users`).

```
        date from to movers fr_pat to_pat fr_users to_users
1 2020-05-01    1  2     10      1      2      100      100
2 2020-05-01    1  3      5      1      3      100      100
3 2020-05-01    1  4      4      1      4      100      100
4 2020-05-01    1  5      3      1      5      100      100
5 2020-05-01    5  1      8      5      1      100      100
```

- `input_dates` : Lista de fechas en las que se realizará la simulación.

```
2020-05-01  2020-05-02  2020-05-03  2020-05-04  2020-05-05    2020-05-06  2020-05-07  2020-05-08  2
```

- `recover_df`: *Data frame* con tasa de recuperación por fecha (1.0 / periodo de recuperación)

```
        date   recrate
1  2020-05-01 0.1666667
2  2020-05-02 0.1666667
3  2020-05-03 0.1666667
4  2020-05-04 0.1666667
5  2020-05-05 0.1666667
6  2020-05-06 0.1666667
7  2020-05-07 0.1666667
8  2020-05-08 0.1666667
9  2020-05-09 0.1666667
10 2020-05-10 0.1666667
```

- `exposerate` : Se puede especificar un valor único, para todas las fechas, o bien un *data frame* con diferentes tasas para cada fecha. Se calcula como (R0 / periodo de recuperación).

```
0.4466667
```

- `exposepd`: Periodo de exposición. La tasa diaria de conversión de exposición a infectado se calcula como (1.0 / exposepd)

```
3
```

- `exposed_pop_inf_prop = 0` : Permite indicar un porcentaje de individuos expuestos que se consideraría población infecciosa, junto a los infectados.

- `TSinday = 1` : *Time Steps* de ejecución para cada día. Por defecto 1. Permite realizar varios pasos de simulación para cada fecha.
- `prob_move_per_TS=0` : Se ignora si es `0` . En caso contrario, se aplica como factor multiplicativo a la matriz de movilidad, indicando la probabilidad de movimiento por cada *time step*.

Output

Genera los resultados de la simulación con las siguientes entradas:

- `epidemic_curve`: Tabla con el número de infectados por día.

```
         Date inf
1  2020-05-01  44
2  2020-05-02  48
3  2020-05-03  48
4  2020-05-04  43
5  2020-05-05  52
6  2020-05-06  58
7  2020-05-07  66
8  2020-05-08  77
9  2020-05-09  86
10 2020-05-10  96
```

- `all_spread`: Número de infectados por region (*patch*) y día.

```
         dates runday  1  2  3  4  5
1  2020-05-01      1 36  2  2  4  0
2  2020-05-02      2 32  6  4  6  0
3  2020-05-03      3 22 10  4 10  2
4  2020-05-04      4 13 13  4 10  3
5  2020-05-05      5 14 14  8 13  3
6  2020-05-06      6 10 18  9 17  4
7  2020-05-07      7 10 18 14 20  4
8  2020-05-08      8 13 18 20 22  4
9  2020-05-09      9 14 24 18 25  5
10 2020-05-10     10 13 25 21 29  8
```

Pasos de la simulación

Movimiento

```
HPop = movementTimeStep(HPop,
                        mobmat,
                        day,
                        control_df,
                        prob_move_per_TS)
```

A partir de la tabla de movilidad, calcula una matriz de probabilidad `P` de ir del patch `i` al `j`:

```
movers(1,1)/poblacion(1)  movers(1,2)/poblacion(1)  movers(1,3)/poblacion(1)
movers(2,1)/poblacion(2)  movers(2,2)/poblacion(2)  movers(2,3)/poblacion(2)
movers(3,1)/poblacion(3)  movers(3,2)/poblacion(3)  movers(3,3)/poblacion(3)
```

Para estimar el número de personas que se mueven realmente en cada posible transición `(i, j)`, genera valores aleatorios a partir de una distribución binomial con los siguientes parámetros:

- `prob` : Valor de la celda `(i, j)` de la matriz de probabilidad anterior (`P(i, j)`).
- `size` : Número actual de infectados del *patch* `i` de origen.

Con ello se obtiene una matriz de número de personas que se mueven de una región a otra. Por ejemplo:

```
0 0 0
1 0 1
0 0 0
```

En este caso de la región 2 habría una persona que se mueve a la región 1, y otra que se mueve a la región 3.

> **WARNING** En principio los valores aleatorios obtenidos a partir de dicha distribución deberían ser válidos como estimación de infectados que se mueven a otra región. Sin embargo, en el código se normalizan las filas de dicha matriz de movimientos, y tras ello se multiplica por el número de infectados de la región de origen de cada fila. En el ejemplo anterior tendríamos:
>
> ```
> 0           0           0
> nInf[2]*0.5 0           nInf[2]*0.5
> 0           0           0
> ```
>
> Esto debería revisarse, ya que en cuanto en un fila haya al menos uno que se mueva, lo que se estaría haciendo es mover toda la población de infectados de la región correspondiente a la fila a otras regiones. Debe revisarse si procede o no multiplicar esta matriz por el número de infectados.

Input

- `HPop` : Población actual.
- `mobmat` : Matriz de movilidad (*data frame*).
- `day` : Fecha del día a simular.
- `control_info` : *Data frame* de control de movilidad.
- `prob_move_per_TS` : Probabilidad de movimiento en cada *time step*.

Bearmod Code

Parameters model

```
#preprocess_ComVal.R
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
# ## 2013 - 2014 data
# movement_data2 = read.csv("baidu/7 feb/Baidu_LBS_flow_201312-201404.csv")
# cell_user_from_data = read.csv("baidu/7 feb/LBSusers_from_201312-201404.csv")
# cell_user_from_data$date = date(cell_user_from_data$date) + days(1)
# cell_user_to_data = read.csv("baidu/7 feb/LBSusers_to_201312-201404.csv")
# cell_user_to_data$date = date(cell_user_to_data$date) + days(1)
```

```
movement_data = read.table("testmove_CV.csv",sep=",",header=T)

patNames = unique(movement_data$to)[order(unique(movement_data$to))]
patIDs = 1:length(patNames)
pat_locator = data.frame(patNames,patIDs)
#convert dates to format R can read
movement_data$date = ymd("2020-02-25")+movement_data$date

#
# missing_dates = c(date("2014-1-17"), date("2014-2-2"),date("2014-2-18"),date("2014-2-20")
#     date("2014-3-1"),date("2014-3-2"))
# for (dates in 1:length(missing_dates)){
#   replaceday = subset(movement_data,Date == missing_dates[dates] - days(1))
#   replaceday$Date = replaceday$Date + days(1)
#   movement_data = rbind(movement_data,replaceday)
# }

recrate = 1/12 #daily probability of recovery
exposerate = 2.68/6 # R0 of 2.68, 5.8 days till seeking treatment
# How many people a single person potentially infects per day -- can be calculated from R0
#estimate if you divide R0 by infectious period
exposepd = 5.1 # incubation period
```

Model runnig

```
#run_model_ComVal.R
####Model running code for BEARmod v.0.6
rm(list=ls())
library(data.table) # fread - fastly reading data
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday,
##     week, yday, year
```

```
library(lubridate)

# setwd('//worldpop.files.soton.ac.uk/Worldpop/Projects/WP519091_Seasonality')
# setwd('D:/OneDrive - University of Southampton/Wuhan Coronavirus R0/Spread risk')
#setwd('C:/Users/sl4m18/OneDrive - University of Southampton/Wuhan Coronavirus R0/Spread risk')

source("bearmod_fx.R")
# source("bearmod/bearmod_fx.R")
source("preprocess_ComVal.R")
#Initial parameters
NPat = length(patNames)
patnInf = rep(0,NPat)
patnExp = c(rep(0,NPat))

#pat_locator$pop = 4941509
```

```r
#three provinces
pat_locator$pop = rep(0, NPat)
pat_locator$pop[1] = 575470
pat_locator$pop[2] = 2540707
pat_locator$pop[3] = 1825332

#start infection in Comunidad Valenciana
patnInf[which(patNames == 1)] = 5
patnInf[which(patNames == 2)] = 25
patnInf[which(patNames == 3)] = 5

#recovery rate variable for each available day
recover_df = data.frame(date = seq(from=min(movement_data$date),to=max(movement_data$date),by="days"),r

##load a new mobility scenario
relative_move_df=data.frame()
relative_move_data = data.frame()

#relative_move_data = read.table("file://C:/Users/nuria/OneDrive/myProjects/COVID-19/BEARmod-master/BEA

#convert dates to format R can read
#relative_move_data$date = ymd("2020-02-25")+30

#### Running the model  ####
HPop = InitiatePop(pat_locator,patnInf,patnExp)
###dates of simulation

#input_dates = rep("2020-02-26",30)
#input_dates=append(input_dates,rep("2020-03-27",15))
#input_dates=append(input_dates,rep("2020-04-11",50))
input_dates = seq(from=min(movement_data$date),to=max(movement_data$date),by="days") # unique(movement_

#input_dates = seq(from=min(movement_data$date),to=max(movement_data$date),by="days")
#input_dates = seq(date("2020-02-26"),date("2020-4-26"),by="days") # coresponding to the period from 20
# input_dates = seq(date("2013-12-02"),date("2014-2-27"),by="days") # coresponding to the period from 2
results = list()

HPop_update2 <- runSim(HPop,pat_locator,relative_move_data,movement_data, input_dates,recover_df, expos
```

```
## [1] "Day: "
## [1] "2020-02-26"
## [1] "Day 2020-02-26 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 0"
## [1] 0
## [1] 0
## [1] 0
## [1] 0
## [1] 0
## [1] 0
## [1] "Number of people newly infectious: 0"
## [1] "Number of people newly exposed: 14.9998771586388"
## [1] "Day: "
## [1] "2020-02-27"
## [1] "Day 2020-02-27 recovery rate: 0.0833333333333333"
```

```
## [1] "Number of people recovering: 1"
## [1] 2.999974
## [1] 2.999974
## [1] 8.999911
## [1] 8.999911
## [1] 2.999992
## [1] 2.999992
## [1] "Number of people newly infectious: 3"
## [1] "Number of people newly exposed: 22.9997281947993"
## [1] "Day: "
## [1] "2020-02-28"
## [1] "Day 2020-02-28 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 2"
## [1] 1.999969
## [1] 4.999943
## [1] 16.99978
## [1] 22.99969
## [1] 3.99998
## [1] 6.999972
## [1] "Number of people newly infectious: 9"
## [1] "Number of people newly exposed: 16.9996784614109"
## [1] "Day: "
## [1] "2020-02-29"
## [1] "Day 2020-02-29 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 3"
## [1] 2.999937
## [1] 6.99988
## [1] 13.99974
## [1] 30.99943
## [1] 0
## [1] 4.999972
## [1] "Number of people newly infectious: 11"
## [1] "Number of people newly exposed: 22.9994721134993"
## [1] "Day: "
## [1] "2020-03-01"
## [1] "Day 2020-03-01 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 4"
## [1] 2.999922
## [1] 7.999802
## [1] 17.99957
## [1] 40.999
## [1] 1.999982
## [1] 5.999955
## [1] "Number of people newly infectious: 15"
## [1] "Number of people newly exposed: 26.9992422920108"
## [1] "Day: "
## [1] "2020-03-02"
## [1] "Day 2020-03-02 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 7"
## [1] 4.999844
## [1] 11.99965
## [1] 17.99944
## [1] 45.99844
## [1] 3.999958
```

```
## [1] 8.999913
## [1] "Number of people newly infectious: 9"
## [1] "Number of people newly exposed: 25.9991517407436"
## [1] "Day: "
## [1] "2020-03-03"
## [1] "Day 2020-03-03 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 6"
## [1] 2.99987
## [1] 11.99952
## [1] 16.99936
## [1] 57.9978
## [1] 5.999924
## [1] 13.99984
## [1] "Number of people newly infectious: 16"
## [1] "Number of people newly exposed: 35.998480768604"
## [1] "Day: "
## [1] "2020-03-04"
## [1] "Day 2020-03-04 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 6"
## [1] 9.999513
## [1] 18.99903
## [1] 20.99905
## [1] 67.99685
## [1] 4.999918
## [1] 16.99976
## [1] "Number of people newly infectious: 24"
## [1] "Number of people newly exposed: 53.9973307021469"
## [1] "Day: "
## [1] "2020-03-05"
## [1] "Day 2020-03-05 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 8"
## [1] 7.999444
## [1] 23.99847
## [1] 35.99807
## [1] 86.99492
## [1] 9.999814
## [1] 22.99957
## [1] "Number of people newly infectious: 24"
## [1] "Number of people newly exposed: 43.9972200528603"
## [1] "Day: "
## [1] "2020-03-06"
## [1] "Day 2020-03-06 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 9"
## [1] 5.9995
## [1] 25.99797
## [1] 30.99789
## [1] 101.9928
## [1] 6.999831
## [1] 25.9994
## [1] "Number of people newly infectious: 23"
## [1] "Number of people newly exposed: 54.9960676209029"
## [1] "Day: "
## [1] "2020-03-07"
## [1] "Day 2020-03-07 recovery rate: 0.0833333333333333"
```

```
## [1] "Number of people recovering: 7"
## [1] 6.999343
## [1] 26.99732
## [1] 36.99704
## [1] 123.9899
## [1] 10.99968
## [1] 34.99908
## [1] "Number of people newly infectious: 36"
## [1] "Number of people newly exposed: 73.993462421865"
## [1] "Day: "
## [1] "2020-03-08"
## [1] "Day 2020-03-08 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 12"
## [1] 8.999046
## [1] 28.99636
## [1] 54.99476
## [1] 157.9846
## [1] 9.999655
## [1] 36.99874
## [1] "Number of people newly infectious: 48"
## [1] "Number of people newly exposed: 101.989462903347"
## [1] "Day: "
## [1] "2020-03-09"
## [1] "Day 2020-03-09 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 17"
## [1] 18.99769
## [1] 43.99405
## [1] 63.99254
## [1] 189.9772
## [1] 18.99923
## [1] 43.99797
## [1] "Number of people newly infectious: 62"
## [1] "Number of people newly exposed: 123.984620261662"
## [1] "Day: "
## [1] "2020-03-10"
## [1] "Day 2020-03-10 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 26"
## [1] 16.99728
## [1] 52.99133
## [1] 79.98873
## [1] 227.9659
## [1] 26.99861
## [1] 58.99657
## [1] "Number of people newly infectious: 81"
## [1] "Number of people newly exposed: 119.981376744304"
## [1] "Day: "
## [1] "2020-03-11"
## [1] "Day 2020-03-11 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 22"
## [1] 22.9956
## [1] 65.98694
## [1] 73.98736
## [1] 241.9532
## [1] 22.99841
```

```
## [1] 70.99499
## [1] "Number of people newly infectious: 66"
## [1] "Number of people newly exposed: 154.97123420237"
## [1] "Day: "
## [1] "2020-03-12"
## [1] "Day 2020-03-12 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 31"
## [1] 25.99399
## [1] 72.98093
## [1] 102.9794
## [1] 313.9327
## [1] 25.99783
## [1] 80.99282
## [1] "Number of people newly infectious: 64"
## [1] "Number of people newly exposed: 158.963892936039"
## [1] "Day: "
## [1] "2020-03-13"
## [1] "Day 2020-03-13 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 37"
## [1] 32.99094
## [1] 96.97187
## [1] 102.9752
## [1] 374.9079
## [1] 22.99772
## [1] 90.99054
## [1] "Number of people newly infectious: 105"
## [1] "Number of people newly exposed: 200.952652200533"
## [1] "Day: "
## [1] "2020-03-14"
## [1] "Day 2020-03-14 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 31"
## [1] 26.9909
## [1] 104.9628
## [1] 111.9687
## [1] 421.8766
## [1] 61.993
## [1] 131.9835
## [1] "Number of people newly infectious: 133"
## [1] "Number of people newly exposed: 243.929252102891"
## [1] "Day: "
## [1] "2020-03-15"
## [1] "Day 2020-03-15 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 43"
## [1] 40.98433
## [1] 129.9471
## [1] 142.9538
## [1] 476.8304
## [1] 59.99113
## [1] 162.9747
## [1] "Number of people newly infectious: 152"
## [1] "Number of people newly exposed: 302.893276762749"
## [1] "Day: "
## [1] "2020-03-16"
## [1] "Day 2020-03-16 recovery rate: 0.0833333333333333"
```

```
## [1] "Number of people recovering: 54"
## [1] 55.9748
## [1] 158.9219
## [1] 184.9298
## [1] 567.7603
## [1] 61.98866
## [1] 193.9633
## [1] "Number of people newly infectious: 170"
## [1] "Number of people newly exposed: 338.858968035718"
## [1] "Day: "
## [1] "2020-03-17"
## [1] "Day 2020-03-17 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 72"
## [1] 57.96826
## [1] 185.8902
## [1] 205.9072
## [1] 664.6674
## [1] 74.98353
## [1] 238.9469
## [1] "Number of people newly infectious: 203"
## [1] "Number of people newly exposed: 399.805076729464"
## [1] "Day: "
## [1] "2020-03-18"
## [1] "Day 2020-03-18 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 80"
## [1] 74.95166
## [1] 221.8418
## [1] 227.879
## [1] 767.5464
## [1] 96.97444
## [1] 296.9213
## [1] "Number of people newly infectious: 261"
## [1] "Number of people newly exposed: 524.694665920087"
## [1] "Day: "
## [1] "2020-03-19"
## [1] "Day 2020-03-19 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 92"
## [1] 74.94202
## [1] 254.7838
## [1] 344.7861
## [1] 947.3325
## [1] 104.9666
## [1] 347.8879
## [1] "Number of people newly infectious: 318"
## [1] "Number of people newly exposed: 610.571657910444"
## [1] "Day: "
## [1] "2020-03-20"
## [1] "Day 2020-03-20 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 118"
## [1] 109.8999
## [1] 314.6837
## [1] 369.7212
## [1] 1119.054
## [1] 130.9506
```

```
## [1] 408.8384
## [1] "Number of people newly infectious: 350"
## [1] "Number of people newly exposed: 709.409734524699"
## [1] "Day: "
## [1] "2020-03-21"
## [1] "Day 2020-03-21 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 136"
## [1] 108.8805
## [1] 367.5643
## [1] 445.5994
## [1] 1348.653
## [1] 154.9298
## [1] 485.7682
## [1] "Number of people newly infectious: 412"
## [1] "Number of people newly exposed: 789.223765307845"
## [1] "Day: "
## [1] "2020-03-22"
## [1] "Day 2020-03-22 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 151"
## [1] 123.8406
## [1] 412.4049
## [1] 483.4819
## [1] 1572.135
## [1] 181.9012
## [1] 594.6694
## [1] "Number of people newly infectious: 515"
## [1] "Number of people newly exposed: 980.882520045585"
## [1] "Day: "
## [1] "2020-03-23"
## [1] "Day 2020-03-23 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 206"
## [1] 148.7764
## [1] 471.1814
## [1] 580.2693
## [1] 1840.404
## [1] 251.8367
## [1] 733.5062
## [1] "Number of people newly infectious: 611"
## [1] "Number of people newly exposed: 1186.38707397581"
## [1] "Day: "
## [1] "2020-03-24"
## [1] "Day 2020-03-24 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 219"
## [1] 204.6384
## [1] 590.8197
## [1] 683.9845
## [1] 2139.389
## [1] 297.7642
## [1] 890.2703
## [1] "Number of people newly infectious: 705"
## [1] "Number of people newly exposed: 1359.80841628921"
## [1] "Day: "
## [1] "2020-03-25"
## [1] "Day 2020-03-25 recovery rate: 0.0833333333333333"
```

```
## [1] "Number of people recovering: 274"
## [1] 219.5306
## [1] 691.3503
## [1] 796.6088
## [1] 2523.998
## [1] 343.669
## [1] 1059.939
## [1] "Number of people newly infectious: 821"
## [1] "Number of people newly exposed: 1616.88246962226"
## [1] "Day: "
## [1] "2020-03-26"
## [1] "Day 2020-03-26 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 287"
## [1] 284.2847
## [1] 832.635
## [1] 953.0364
## [1] 2994.034
## [1] 379.5614
## [1] 1244.501
## [1] "Number of people newly infectious: 999"
## [1] "Number of people newly exposed: 2023.41784102061"
## [1] "Day: "
## [1] "2020-03-27"
## [1] "Day 2020-03-27 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 341"
## [1] 341.9673
## [1] 1014.602
## [1] 1176.147
## [1] 3573.181
## [1] 505.3035
## [1] 1507.804
## [1] "Number of people newly infectious: 1229"
## [1] "Number of people newly exposed: 2331.70827471055"
## [1] "Day: "
## [1] "2020-03-28"
## [1] "Day 2020-03-28 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 432"
## [1] 356.7131
## [1] 1172.315
## [1] 1404.945
## [1] 4238.127
## [1] 570.0497
## [1] 1787.854
## [1] "Number of people newly infectious: 1452"
## [1] "Number of people newly exposed: 2749.1553708326"
## [1] "Day: "
## [1] "2020-03-29"
## [1] "Day 2020-03-29 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 518"
## [1] 455.0748
## [1] 1385.39
## [1] 1625.409
## [1] 5018.535
## [1] 668.672
```

```
## [1] 2091.526
## [1] "Number of people newly infectious: 1666"
## [1] "Number of people newly exposed: 3307.45256300695"
## [1] "Day: "
## [1] "2020-03-30"
## [1] "Day 2020-03-30 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 634"
## [1] 530.3257
## [1] 1648.716
## [1] 1931.121
## [1] 5956.656
## [1] 846.0063
## [1] 2531.532
## [1] "Number of people newly infectious: 1966"
## [1] "Number of people newly exposed: 3885.48897542461"
## [1] "Day: "
## [1] "2020-03-31"
## [1] "Day 2020-03-31 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 788"
## [1] 616.2995
## [1] 1945.015
## [1] 2273.008
## [1] 7082.664
## [1] 996.1817
## [1] 3028.714
## [1] "Number of people newly infectious: 2377"
## [1] "Number of people newly exposed: 4728.72848754227"
## [1] "Day: "
## [1] "2020-04-01"
## [1] "Day 2020-04-01 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 844"
## [1] 742.7462
## [1] 2299.762
## [1] 2773.097
## [1] 8444.76
## [1] 1212.886
## [1] 3663.6
## [1] "Number of people newly infectious: 2842"
## [1] "Number of people newly exposed: 5568.50608051979"
## [1] "Day: "
## [1] "2020-04-02"
## [1] "Day 2020-04-02 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 1022"
## [1] 902.4354
## [1] 2730.197
## [1] 3236.907
## [1] 10051.67
## [1] 1429.164
## [1] 4352.764
## [1] "Number of people newly infectious: 3321"
## [1] "Number of people newly exposed: 6436.45654777039"
## [1] "Day: "
## [1] "2020-04-03"
## [1] "Day 2020-04-03 recovery rate: 0.0833333333333333"
```

```
## [1] "Number of people recovering: 1241"
## [1] 1017.837
## [1] 3255.034
## [1] 3695.052
## [1] 11779.72
## [1] 1723.568
## [1] 5215.331
## [1] "Number of people newly infectious: 3965"
## [1] "Number of people newly exposed: 7846.77025985452"
## [1] "Day: "
## [1] "2020-04-04"
## [1] "Day 2020-04-04 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 1458"
## [1] 1261.126
## [1] 3853.16
## [1] 4511.803
## [1] 13999.52
## [1] 2073.841
## [1] 6279.173
## [1] "Number of people newly infectious: 4706"
## [1] "Number of people newly exposed: 9039.75231179842"
## [1] "Day: "
## [1] "2020-04-05"
## [1] "Day 2020-04-05 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 1696"
## [1] 1493.999
## [1] 4566.159
## [1] 5117.871
## [1] 16424.39
## [1] 2427.883
## [1] 7475.055
## [1] "Number of people newly infectious: 5606"
## [1] "Number of people newly exposed: 10838.7733842428"
## [1] "Day: "
## [1] "2020-04-06"
## [1] "Day 2020-04-06 recovery rate: 0.0833333333333333"
## [1] "Number of people recovering: 2091"
## [1] 1789.12
## [1] 5406.279
## [1] 6241.365
## [1] 19434.76
## [1] 2808.288
## [1] 8857.343
## [1] "Number of people newly infectious: 6592"
## [1] "Number of people newly exposed: 12709.6927356084"
## [1] "Warning: row sums > 1 in movement matrix. Correcting..."
```

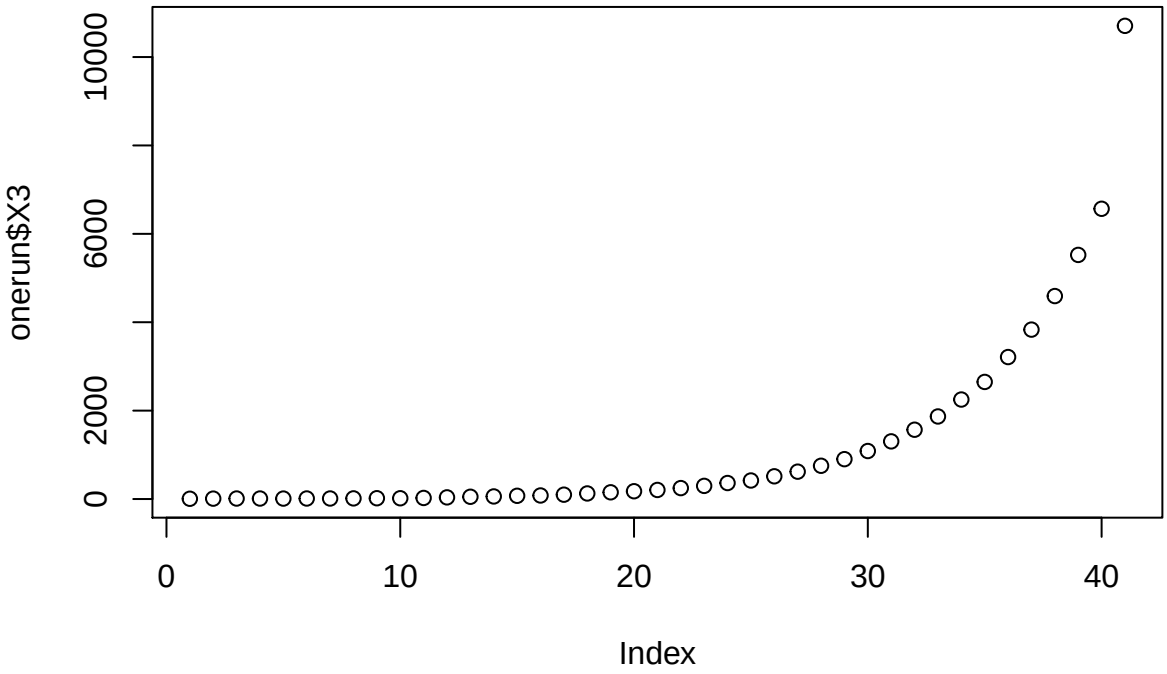
```r
run <- 1
results[[run]] <- HPop_update2$all_spread
onerun <- data.frame(results[run])

#Visualising results
# for (run in 1:2){
#
```

```
#   HPop_update2 = runSim(HPop,pat_locator,relative_move_data,movement_data, input_dates,recover_df, ex
#   #print(paste0("Run # ",run))
#   results[[run]] = HPop_update2$all_spread
# }

# this is just one run of the model instance 10
onerun <- data.frame(results[run])

#plot
#infected
plot(onerun$X3)
```




```
onerun$X3[35:50]
```

```
##  [1]  2649  3212  3833  4592  5523  6567 10706    NA    NA    NA    NA
## [12]    NA    NA    NA    NA    NA
```

```
save(results,file="results_ComVal.RData")
#
```

The core model file

```
######
#bearmode_fx.R
library(lubridate)

#This function creates the starting population
InitiatePop = function(pat_locator,initialInf,initialExp){
  NPat = dim(pat_locator)[1]
  list(
    nInitialInf = initialInf,
    nInitialExp = initialExp,
```

```
    nInf = initialInf,
    nExp = initialExp,
    nRec = rep(0,NPat),
    nTotal = pat_locator$pop,
    names = pat_locator$patNames,
    IDs = pat_locator$patIDs,
    relativeInf = rep(1,NPat),
    nRecoveredToday = rep(0,NPat),
    nInfectedToday = rep(0,NPat),
    nExposedToday = rep(0,NPat),
    nInfMovedToday = rep(0,NPat),
    controlled = rep(0,NPat)
  )
}
##########################################################################
##### Epidemic functions: exposure, infectivity, recovery  ####
##########################################################################
# Random number of people recovered R at current_day (cumulated binomial)
##########################################################################
recoveryTimeStep = function(HPop, recrate_values,current_day){
  recrate = subset(recrate_values,date == current_day)$recrate
  print(paste0("Day ",current_day, " recovery rate: ", recrate))
  for (i in 1:length(HPop$nInf)){
    # HPop$nRecoveredToday[i] is computed by using a random vector of size
    # HPop$nInf[i] from a Binomial Distribution with parameters n=1 and
    # p = recreate
  HPop$nRecoveredToday[i]= sum(rbinom(HPop$nInf[i],1,recrate))
  HPop$nInf[i] = HPop$nInf[i] - HPop$nRecoveredToday[i]
  HPop$nRec[i] = HPop$nRec[i] + HPop$nRecoveredToday[i]
  }
  print(paste0("Number of people recovering: ",sum(HPop$nRecoveredToday)))
  HPop
}
##########################################################################
# Random number of people infected I at current_day (cumulated binomial)
##########################################################################
exposedtoinfTimeStep = function(HPop, exp_to_infrate){
  #(exp_to_infrate)
  for (i in 1:length(HPop$nInf)){
    print(HPop$nExposedToday[i])
    print(HPop$nExp[i])
    # HPop$nInfectedToday[i] is computed by using a random vector of size
    # HPop$nExp[i] from a Binomial Distribution with parameters n=1 and
    # p = exp_to_infrate
    HPop$nInfectedToday[i]= sum(rbinom(HPop$nExp[i],1,exp_to_infrate))
    HPop$nInf[i] = HPop$nInf[i] + HPop$nInfectedToday[i]
    HPop$nExp[i] = HPop$nExp[i] - HPop$nInfectedToday[i]
  }
  print(paste0("Number of people newly infectious: ",sum(HPop$nInfectedToday)))
  HPop
}
##########################################################################
# Random number of people exposed E at current_day (transformed Poisson)
```

```
#########################################################################
exposedTimeStep = function(HPop, exposerate_df, current_day, exposed_pop_inf_prop){
  if (is.numeric(exposerate_df)){
    exposerate = exposerate_df
  }
  if (is.data.frame(exposerate_df)){
    exposerate = subset(exposerate_df, date == current_day)$exposerate
  }
  for (i in 1:length(HPop$nInf)){
    infectious_pop = HPop$nInf[i] + exposed_pop_inf_prop * HPop$nExp[i]
    #HPop$nExposedToday[i]= sum(rbinom(infectious_pop,1,exposerate)) * (1 - ( (HPop$nInf[i] + HPop$nExp
    HPop$nExposedToday[i]= sum(rpois(infectious_pop,exposerate)) * (1 - ( (HPop$nInf[i] + HPop$nExp[i]
    if (HPop$nExp[i] + HPop$nExposedToday[i] < HPop$nTotal[i] - HPop$nInf[i] - HPop$nRec[i] ) {
      HPop$nExp[i] = HPop$nExp[i] + HPop$nExposedToday[i]

    }
    else {
      HPop$nExposedToday[i] = max(0,HPop$nTotal[i] - HPop$nInf[i] - HPop$nExp[i]- HPop$nRec[i])
      HPop$nExp[i]  = max(0,HPop$nTotal[i] - HPop$nInf[i] - HPop$nRec[i])
    }
  }
  print(paste0("Number of people newly exposed: ",sum(HPop$nExposedToday)))
  HPop
}
#########################################################################################################
#
########################################
exposedTimeStep_timespent = function(HPop, exposerate_df, current_day, exposed_pop_inf_prop,ts_data){
  TS_matrix = matrix(0,NPat,NPat,dimnames=list(patIDs,patIDs))
  daily_move = subset(ts_data,date == current_day)
  daily_move = subset(daily_move,!is.na(fr_pat) & !is.na(to_pat) & !is.na(fr_users) & !is.na(movers))
  daily_move_mat = daily_move[,is.element(names(daily_move),c("fr_pat","to_pat","fr_users","movers"))]
  daily_move_mat = as.matrix(daily_move_mat)
  col1 = which(colnames(daily_move_mat) == "fr_pat")
  col2=which(colnames(daily_move_mat) == "to_pat")
  colmove = which(colnames(daily_move_mat) == "movers")
  colusers=which(colnames(daily_move_mat) == "fr_users")
  TS_matrix[daily_move_mat[,c(col1,col2)]] = daily_move_mat[,colmove]/daily_move_mat[,colusers]
  if (length(which(rowSums(TS_matrix)>1)) > 0){
    print("Warning: row sums > 1 in movement matrix. Correcting...")
    correctingrows = which(rowSums(TS_matrix)>1)
    for (i in correctingrows){
      TS_matrix[i,] = TS_matrix[i,] /sum(TS_matrix[i,] )
    }
  }
  for (i in 1:length(patIDs)){
    TS_matrix[i,i] = 1 - sum(TS_matrix[i,-i])
  }
  if (is.numeric(exposerate_df)){
    exposerate = exposerate_df
  }
  if (is.data.frame(exposerate_df)){
    exposerate = subset(exposerate_df, date == current_day)$exposerate
```

```
  }
  movement_adjusted_infectious_prop = rep(0,length(HPop$nInf))
  for (i in 1:length(HPop$nInf)){
    movement_adjusted_infectious_prop[i] = sum(((HPop$nInf * TS_matrix[,i]) + exposed_pop_inf_prop * su
  }
  susceptible_vec = round(HPop$nTotal - HPop$nInf - HPop$nExp - HPop$nRec)
  ##############################################################################################
  probability_infection = 1-exp(-exposerate * movement_adjusted_infectious_prop)
  ##############################################################################################
  for (i in 1:length(HPop$nInf)){
    susceptible_weighted_pop = round(susceptible_vec[i]*TS_matrix[i,])
    HPop$nExposedToday[i] = round(sum(rbinom(length(susceptible_weighted_pop),size = susceptible_weight
    if (HPop$nExp[i] + HPop$nExposedToday[i] < HPop$nTotal[i] - HPop$nInf[i] - HPop$nRec[i] ) {
      HPop$nExp[i] = HPop$nExp[i] + HPop$nExposedToday[i]

    } else {
      HPop$nExposedToday[i] = HPop$nTotal[i] - HPop$nInf[i] - HPop$nExp[i]- HPop$nRec[i]
      HPop$nExp[i]  = HPop$nTotal[i] - HPop$nInf[i] - HPop$nRec[i]

    }
  }
  #print(paste0("Number of people newly exposed: ",sum(HPop$nExposedToday)))
  HPop
}

####### Activity functions: Human movement ####
movementTimeStep = function(HPop, mobmat,day,control_df,prob_move_per_TS){

  # Crea una matrix de nPatches x nPatches
  movement_matrix = matrix(0,NPat,NPat,dimnames=list(patIDs,patIDs))
  # Obtiene el subconjunto de movimientos del dia actual
  daily_move = subset(mobmat,date == day)
  # Elimina filas con valores vacíos
  daily_move = subset(daily_move,!is.na(fr_pat) & !is.na(to_pat) & !is.na(fr_users) & !is.na(movers))
  # Filtra solo las columnas que nos interesan
  daily_move_mat = daily_move[,is.element(names(daily_move),c("fr_pat","to_pat","fr_users","movers"))]
  daily_move_mat = as.matrix(daily_move_mat)

  # Obtiene los indices de las diferentes columnas
  col1 = which(colnames(daily_move_mat) == "fr_pat")
  col2=which(colnames(daily_move_mat) == "to_pat")
  colmove = which(colnames(daily_move_mat) == "movers")
  colusers=which(colnames(daily_move_mat) == "fr_users")

  # Matriz de movimientos relativos. Cada elemento (i,j) representa que proporcion de usuarios de i se
  movement_matrix[daily_move_mat[,c(col1,col2)]] = daily_move_mat[,colmove]/daily_move_mat[,colusers]
  # Se comprueba que ninguna fila sume más de 1. Si esto ocurriese, se corrige normalizando
  if (length(which(rowSums(movement_matrix)>1)) > 0){
    print("Warning: row sums > 1 in movement matrix. Correcting...")
    correctingrows = which(rowSums(movement_matrix)>1)
    for (i in correctingrows){
    movement_matrix[i,] = movement_matrix[i,] /sum(movement_matrix[i,] )
    }
```

```
}
# En caso de indicar probabilidad de movimiento por time step, se multiplica la matriz de movimiento
if (prob_move_per_TS > 0){
  movement_matrix = movement_matrix*prob_move_per_TS
}
# En la diagonal se pone 1 menos la suma del resto de la fila
for (i in 1:length(patIDs)){
  movement_matrix[i,i] = 1 - sum(movement_matrix[i,-i])
}

# Crea un vector de poblacion con movilidad controlada
HPop$controlled = rep(0,length(HPop$names))
# Busca en la tabla de control de movilidad (si la hubiese) las entradas de la fecha actual
if (length(which(control_df$date == day)) > 0){
  # Filtra las filas de la tabla correspondientes al dia actual
  control_df_sub = subset(control_df,date == day)
  # Comprueba si el filtrado ha devuelto filas
  if (dim(control_df_sub)[1] > 0){
  # Pone en cada entrada (patch) del vector de movilidad controlada el ratio a aplicar
  for (i in 1:dim(control_df_sub)[1]){
    HPop$controlled[which(HPop$names == control_df_sub$from[i])] = control_df_sub$relative_move[i]

  }
}
}
# En caso de haber control de movilidad, se aplica la reduccion de movilidad con stopMovement
if (sum(HPop$controlled)>0){
  movement_matrix = stopMovement(HPop,movement_matrix,day)
}
#deterministic version
#HPop$nInfMovedToday = colSums(diag(HPop$nInf) %*% movement_matrix) - HPop$nInf
#HPop$nInf = colSums(diag(HPop$nInf) %*% movement_matrix)
HPop$nInf = ceiling(HPop$nInf)
# stochastic version

# Para cada entrada (i,j) de la matriz de movilidad:
#     - movement_matrix(i,j) representa la probabilidad p de que un invididuo de la region i se mueve
#     - Se toma el número de infectados de la region i como numero de ensayos de la distribucion bino
#     - Se genera un valor aleatorio a partir de dicha distribucion para cada entrada, que representa
  z <- rbinom(n=NPat^2,size = rep(HPop$nInf,each=NPat),prob = t(movement_matrix)[])
moved_matrix = t(matrix(z,NPat,NPat,dimnames=list(patIDs,patIDs)))
# Se normaliza cada fila de la matriz de movimientos y se multiplican por el número de infectados en
# WARNING: Esto es como suponer que todos los infectados salen de la region i, veo innecesario multip
for (i in 1:dim(moved_matrix)[1]){
   if (sum(moved_matrix[i,]) > 0){
   moved_matrix[i,] = moved_matrix[i,]/sum(moved_matrix[i,]) * HPop$nInf[i]
   }
}
# Se suman las columnas de la matrix (numero de infectados que recibe cada region j)
HPop$nInfMovedToday = ceiling(colSums(moved_matrix))

# A la población infectada de cada patch, se le restan los que salen y se suman los que entran
HPop$nInf = HPop$nInf - floor(rowSums(moved_matrix)) + ceiling(colSums(moved_matrix))
```

```
  #quick fix

  # Evita que el numero de infectados de una region pueda ser superior al numero total de habitantes
  for (i in 1:length(HPop$nInf)){
    if (HPop$nInf[i] > HPop$nTotal[i]){
      HPop$nInf[i] = HPop$nTotal[i]
    }
  }

  #print(paste0("Number of infected people moving: ",sum(abs(HPop$nInfMovedToday))/2))
  HPop
}


###### Response functions: Control
#relative_movement is the proportion of original movement out/in that we want to keep -- ie. .1 = 10% o
stopMovement = function(HPop,mobmat,current_date){
  stopping = which(HPop$controlled > 0)
    if (length(stopping) > 0){
     # print(paste("stopping movement in patches", HPop$names[stopping]))
      for (ctrl_pat in stopping){
    control_patches = HPop$IDs[ctrl_pat]
    # Para cada patch con movilidad controlada, se aplica el factor de control en toda la fila y columna
    mobmat[control_patches,] = mobmat[control_patches,] * HPop$controlled[ctrl_pat]
    mobmat[,control_patches] = mobmat[,control_patches] * HPop$controlled[ctrl_pat]
    # Recalcula la diagonal
    for (i in 1:length(HPop$IDs)){
      mobmat[i,i] = 1 - sum(mobmat[i,-i])
    }
      }
    }
    mobmat
}
########################################
# Revisado has aquí 1 de Abril 2020
###### Master function  ################
runSim = function(HPop,pat_info,control_info,mobmat,day_list,recrate_values,exposerate_df,exposepd,expos


  epidemic_curve <- data.frame(Date=as.Date(character()),
                               inf=c(),
                               stringsAsFactors=FALSE)

  if (TSinday > 1){
    #recrate_values$recrate = 1-(1-recrate_values$recrate)^(1/TSinday)
    exposetoinfrate = 1/exposepd
    exposepd = 1/(1 - exp(log(1-exposetoinfrate) / TSinday))
    #recrate_values$recrate = 1 - ((1 - recrate_values$recrate) ^ (1/TSinday))
    recrate_values$recrate = 1 - exp(log(1-recrate_values$recrate) / TSinday)
    if (is.numeric(exposerate_df)){
     # exposerate_df = 1-(1-exposerate_df)^(1/TSinday)
      exposerate_df = exposerate_df/TSinday
     # recrate_values$recrate = 1 - ((1 - recrate_values$recrate) ^ (1/TSinday))
```

```
    }
    if (is.data.frame(exposerate_df)){
     # exposerate_df$exposerate = 1-(1-exposerate_df$exposerate)^(1/TSinday)
      exposerate_df$exposerate = 1 - exp(log(1-exposerate_df$exposerate) / TSinday)
    }
  }

  all_spread = matrix(0,length(day_list),length(HPop$nInf))
  colnames(all_spread) = HPop$names
  #print(all_dates)
  for (current_day in 1:length(day_list)){
    for (current_TS in 1:TSinday){
    print("Day: ")
    print(day_list[current_day])
    HPop = recoveryTimeStep(HPop,recrate_values,day_list[current_day])
    HPop = exposedtoinfTimeStep(HPop,1/exposepd)
    HPop = exposedTimeStep(HPop,exposerate_df, day_list[current_day], exposed_pop_inf_prop)

    HPop = movementTimeStep(HPop,mobmat,day_list[current_day],control_info,prob_move_per_TS)
    }
    #save(HPop,file=paste(current_day,".RData"))
    epidemic_curve = rbind(epidemic_curve,data.frame(Date = day_list[current_day], inf = sum(HPop$nInf)
    all_spread[current_day,] = HPop$nInf

  }
  all_spread_2 = data.frame(dates = day_list,runday = 1:length(day_list))
  all_spread_2= cbind(all_spread_2,all_spread)
  list(HPop = HPop,epidemic_curve = epidemic_curve,all_spread=all_spread_2)
}


runSim_timespent = function(HPop,pat_info,control_info,TS_data,day_list,recrate_values,exposerate_df,ex


  epidemic_curve <- data.frame(Date=as.Date(character()),
                               inf=c(),
                               stringsAsFactors=FALSE)

  if (TSinday > 1){
    #recrate_values$recrate = 1-(1-recrate_values$recrate)^(1/TSinday)
    exposetoinfrate = 1/exposepd
    exposepd = 1/(1 - exp(log(1-exposetoinfrate) / TSinday))
    #recrate_values$recrate = 1 - ((1 - recrate_values$recrate) ^ (1/TSinday))
    recrate_values$recrate = 1 - exp(log(1-recrate_values$recrate) / TSinday)
    if (is.numeric(exposerate_df)){
      # exposerate_df = 1-(1-exposerate_df)^(1/TSinday)
      exposerate_df = exposerate_df/TSinday
      # recrate_values$recrate = 1 - ((1 - recrate_values$recrate) ^ (1/TSinday))
    }
    if (is.data.frame(exposerate_df)){
      # exposerate_df$exposerate = 1-(1-exposerate_df$exposerate)^(1/TSinday)
      exposerate_df$exposerate = 1 - exp(log(1-exposerate_df$exposerate) / TSinday)
    }
```

```
  }

  all_spread = matrix(0,length(day_list),length(HPop$nInf))
  colnames(all_spread) = HPop$names
  #print(all_dates)
  for (current_day in 1:length(day_list)){
    for (current_TS in 1:TSinday){
      print(day_list[current_day])
      HPop = recoveryTimeStep(HPop,recrate_values,day_list[current_day])
      HPop = exposedtoinfTimeStep(HPop,1/exposepd)
      HPop = exposedTimeStep_timespent(HPop,exposerate_df, day_list[current_day], exposed_pop_inf_prop,
    }
    #save(HPop,file=paste(current_day,".RData"))
    epidemic_curve = rbind(epidemic_curve,data.frame(Date = day_list[current_day], inf = sum(HPop$nInf)
    all_spread[current_day,] = HPop$nInf

  }
  all_spread_2 = data.frame(dates = day_list,runday = 1:length(day_list))
  all_spread_2= cbind(all_spread_2,all_spread)
  list(HPop = HPop,epidemic_curve = epidemic_curve,all_spread=all_spread_2)
}
```