```r
#################################################################################

#### BML Simulation Study

#### Put in this file the code to run the BML simulation study for a set of
input parameters.
#### Save some of the output data into an R object and use save() to save it to
disk for reference
#### when you write up your results.
#### The output can e.g. be how many steps the system took until it hit
gridlock or
#### how many steps you observered before concluding that it is in a free
flowing state.


##### Code #####




#### Initialization function.
## Input : size of grid [r and c] and density [p]
## Output : A matrix [m] with entries 0 (no cars) 1 (red cars) or 2 (blue cars)
## that stores the state of the system (i.e. location of red and blue cars)

bml.init <- function(r, c, p){
  grid.r = r
  grid.c = c
  density = p

  ncars = round(p * grid.r * grid.c, 0)
  grid.size = grid.r * grid.c

  car.sample = c(rep(0, grid.size - ncars), rep(1, ncars/2), rep(2, ncars/2))
  m = matrix(sample(car.sample, grid.size, replace = T), nrow = grid.r)


  return(m)
}

image(t(m)[,nrow(m):1], axes=FALSE, col = c("white", "red", "blue"))



#### Function to move the system one step (east and north)
## Input : a matrix [m] of the same type as the output from bml.init()
## Output : TWO variables, the updated [m] and a logical variable
## [grid.new] which should be TRUE if the system changed, FALSE otherwise.

## NOTE : the function should move the red cars once and the blue cars once,
## you can write extra functions that do just a step north or just a step east.


check.grid.east <- function(m,i,j){
```

```
  if (j == dim(m)[2]-1){
    if(m[i,j] == 1 & m[i,j+1] == 0){
      m[i,j] = 0
      m[i,j+1] = 1
    }
    else if(m[i,j] == 1 & m [i,j+1] == 1 & m[i,1] == 0){
      m[i,j+1] = 0
      m[i,1] = 1
    }
    else if(m[i,j] == 0 & m[i,j+1] == 1 & m[i,1] == 0){
      m[i,j+1] = 0
      m[i,1] = 1
    }
    else if(m[i,j] == 2 & m[i,j+1] == 1 & m[i,1] == 0){
      m[i,j+1] = 0
      m[i,1] = 1
    }
  }
  else if(j == dim(m)[2] & m[i,j] == 1 & m[i,1] == 0){
    m[i,j] = 0
    m[i,1] = 1
  }
  return(m)
}


check.grid.north <- function(m,i,j){
  if (i == 1 & m[1,j] == 2 & m[dim(m)[1],j] == 0){
    m[1,j] = 0
    m[dim(m)[1],j] = 2
  }
  else if ( i == 2 & m[2,j] == 0 & m[1,j] == 2 & m[dim(m)[1],j] == 0){
    m[dim(m)[1],j] = 2
    m[1,j] = 0
  }
  else if ( i == 2 & m[2,j] == 1 & m[1,j] == 2 & m[dim(m)[1],j] == 0){
    m[dim(m)[1],j] = 2
    m[1,j] = 0
  }
  else if ( i == 2 & m[2,j] == 2 & m[1,j] == 0){
    m[2,j] = 0
    m[1,j] = 2
  }
  else if ( i == 2 & m[1,j] == 2 & m[2,j] == 2 & m[dim(m)[1], j] == 0){
    m[dim(m)[1],j] = 2
    m[1,j] = 0
  }
  return(m)
}


bml.step.east <- function(m){
  i=1
  max=dim(m)[2]
  for(i in 1:dim(m)[1]){
```

```
    j=1
    while(j < dim(m)[2] - 1){
      if(m[i,j] == 1 & m[i,j+1] == 0){
        m[i,j] = 0
        m[i,j+1] = 1
        j = j+2
      }
      else j = j+1
    }
    m=check.grid.east(m,i,j)
  }

  return(m)
}
```

```
bml.step.north <- function(m){
  j=1
  for(j in 1:dim(m)[2]){
    i=dim(m)[1]
    while(i > 2){
      if(m[i,j] == 2 & m[i-1,j] == 0){
        m[i,j] = 0
        m[i-1,j] = 2
        i = i-2
      }
      else i = i-1
    }
    m=check.grid.north(m,i,j)

  }
  return(m)
}


bml.step <- function(m){
  if(length(m) == 1) return(list(m, as.logical('FALSE')))
  else{
    m1 = m
    m = bml.step.east(m)
    m = bml.step.north(m)
    grid.new = any(m1 != m)
    return(list(m,grid.new))
  }
}
```

```
## I have set bml.sim function to have the max iteration step at 20,000.
```

```
## This function returns the list with the 'initial grid', 'end grid' and
number of steps taken.
## Here the number of steps will be either 20,000 meaning that the grid has a
free-flow or
## some number of steps taken until the grid got gridlocked.

bml.sim <- function(r, c, p){
  steps=20000             #this is where we set the max number of iterations
  s=2
  m.initial = bml.init(r,c,p)
  x = bml.step(m.initial)
  m=x[[1]]
  check.gridlock=x[[2]]
  while(check.gridlock & s < steps){
    x=bml.step(m)
    m=x[[1]]
    check.gridlock=x[[2]]
    s=s+1
  }
  return(list(m.initial,m,s))      #output has initial grid, end grid and
number of iterations reached

}



## This function help us observe the behaviour of the 'n' different sample
grids
## of the same size ('r' x 'c') and same density 'p'. It helps us examine the
## structure of the grids that get gridlocked and those that aren't.
## I used this function to examine and plot initial and end grids for the free-
flow and
## gridlocked one.

bml.sim2 <- function(r,c,p,n){     #this function generates 'n' different
samples of the same grid size and density
  x=c()
  while (n > 0){
    x = c(x, bml.sim(r,c,p))
    n=n-1
  }
  return(list(x))        #this test function returns an initial, an end grid,
as well as the number of iterations for each grid for number of 'n' samples
}


## This functions serves us to observe the behaviour of different grid sizes
and different denisities.
## It generates 'n' sample grids of the same size and denisity and it returns
the number
## of iterations needed for each one of them.

bml.sim3 <- function(r,c,p,n){
  number.of.iterations = c()
  while (n > 0){
```

```
    number.of.iterations = c(number.of.iterations, bml.sim(r,c,p)[[3]])
    n=n-1
  }
  return(number.of.iterations)

}
```

```
################################ Analyzing the observations
################################
## Running my function bml.sim2(5,5,0.8,50) and observing the results, I am
finding that
## even it is very unlickely that the grids with denisity 80% are grid-lock-
free, these
## do occur. One matrix is

##       [,1] [,2] [,3] [,4] [,5]
## [1,]    1    1    2    2    1
## [2,]    2    1    2    0    2
## [3,]    0    0    2    2    2
## [4,]    1    1    1    2    0
## [5,]    1    1    1    2    0

initial.grid.lock.free =
matrix(c(1,2,0,1,1,1,1,0,1,1,2,2,2,1,1,2,0,2,2,2,1,2,2,0,0), nrow=5)
## or image given by:
image(t(initial.grid.lock.free)[,nrow(initial.grid.lock.free):1], axes=FALSE,
col = c("white", "red", "blue"))

## Which even it went through 20,000 iterations, is still grid-lock-free. This
bahaviour, even unlikely
## does occur when one of the columns (for blue cars) or rows (for red cars)
gets isolated
## by surrounding columns or rows which are gridlocked. This is very unlickely
to happen,
## and for the sample size 50, this chance we get this sample on the first draw
is around 4%
## as seen using my function bml.sim3 which gives us the number of iterations
taken for 'n' sampples
## of the same size and density:

# bml.sim3(5,5,0.8,50)
# [1]     6     8     7     5     7     6     9     7     8     4     4    10
8     8     4    10
# [17]    14     5    14     5    10 20000    10    10    14     7    11    17
13     6     4    15
# [33]     7    15     7     6    10     7    14     3    18 20000     5     5
12     4     7     5
# [49]    11     8
```

```
## Note: Those two sample with 20,000 iterations are grid-lock-free.
## This matrix from above, at the 20,000th step looks like this:


##       [,1] [,2] [,3] [,4] [,5]
## [1,]    1    1    2    2    1
## [2,]    2    1    2    0    2
## [3,]    0    0    2    2    2
## [4,]    1    1    1    2    0
## [5,]    1    1    1    2    0

end.grid.lock.free =
matrix(c(1,2,0,1,1,1,1,1,0,1,1,2,2,2,1,1,2,0,2,2,2,1,2,2,0,0),nrow=5)
## With the image:
image(t(end.grid.lock.free)[,nrow(end.grid.lock.free):1], axes=FALSE, col =
c("white", "red", "blue"))


## Much easiser to find and take as an example of 5x5 with p=0.8 are the grids
that get gridlocked.
## From the above results from bml.sim3(5,5,0.8,50) it is clear that without
outliers (20,000) this matrix
## gets gridlocked after 8.542 steps on average.

## To ilustrate this here is the initial grid of the grid-lock matrix:

##       [,1] [,2] [,3] [,4] [,5]
## [1,]    1    1    1    1    0
## [2,]    0    1    1    1    2
## [3,]    2    1    2    2    2
## [4,]    2    2    1    2    2
## [5,]    0    0    1    2    0

## This matrix got grid-locked after 10 steps.

## In its grid-lock it looked like this:

## [,1] [,2] [,3] [,4] [,5]
## [1,]    1    1    1    2    1
## [2,]    2    1    2    1    1
## [3,]    2    2    1    2    2
## [4,]    0    0    0    1    2
## [5,]    0    0    1    2    2

## Or graphically:
initial.grid.lock =
matrix(c(1,0,2,2,0,1,1,1,2,0,1,1,2,1,1,1,1,2,2,2,0,2,2,2,0), nrow=5)
end.grid.lock = matrix(c(1,2,2,0,0,1,1,2,0,0,1,2,1,0,1,2,1,2,1,2,1,1,2,2,2),
nrow=5)

image(t(initial.grid.lock)[,nrow(initial.grid.lock):1], axes=FALSE, col =
c("white", "red", "blue"))
image(t(end.grid.lock)[,nrow(end.grid.lock):1], axes=FALSE, col = c("white",
"red", "blue"))
```

```
############### More testings with different densities ##################

## Using my bml.sim3(r,c,p,n) function which gives the vector of the numbers of
itereted steps for each of
## 'n' samples, I have found that p=0.55 is an interesting density,
## as the percentage of the grid-lock matrixes is solidly high at 74%. This is
shown from 50 5x5
## sample grids with denisity p=0.55:

# > bml.sim3(5,5,0.55,50)
# [1] 20000 20000    23 20000 20000 20000 20000 20000 20000 20000 20000    24
20000 20000 20000 20000
# [17] 20000 20000    24 20000 20000 20000    11    15 20000 20000 20000    13
20000 20000    17    17
# [33] 20000 20000 20000 20000 20000 20000 20000    19 20000     9    44 20000    23
20000 20000 20000     8
# [49] 20000 20000


## Another density I was observing the behaviour is p=0.32 for the same grid
5x5 and using my bml.sim3
## function it seems that the grids are lock-free and that the traffic doesn't
get jammed:

# > bml.sim3(5,5,0.32,50)
# [1] 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000
20000 20000 20000 20000
# [17] 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000
20000 20000 20000 20000
# [33] 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000
20000 20000 20000 20000
# [49] 20000 20000

## Testing with density p=0.35 and using bml.sim3 function with 10 samples we
get:

# > bml.sim3(5,5,0.35,10)
# [1] 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000


## Exploring the lower densities p=0.3 and p=0.25, it is clear the the traffic
will flow
## undistrurbed and therefore grid never gets locked:

# > bml.sim3(5,5,0.3,50)
# [1] 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000
20000 20000 20000 20000
# [17] 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000
20000 20000 20000 20000
# [33] 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000
20000 20000 20000 20000
```

```
# [49] 20000 20000

# > bml.sim3(5,5,0.25,50)
# [1] 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000
20000 20000 20000 20000
# [17] 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000
20000 20000 20000 20000
# [33] 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000
20000 20000 20000 20000
# [49] 20000 20000


## Now examining the bahaviour of the bigger grid (60x60) with the same
densities: p=0.55, p=0.35,
## p=0.32, and p=0.28 we get these results using the bml.sim3 function for 5
samples:

#### for p=0.55 and 10 samples we have the average of

# > bml.sim3(25,25,0.55,10)
# [1]  64 117  67  60  79  52  43  81  65  90

#### for p=0.35 and 10 samples we have the results as:

# > bml.sim3(25,25,0.35,10)
# [1] 513 20000 1092 20000 2439 239 436 335 2517 159

#### for p=0.32 and 10 samples, we have the results:

# > bml.sim3(25,25,0.32,10)
# [1] 4946 20000 1596 20000 20000 1348 20000 3821 20000 20000

#### for p=0.3 and p=0.25 and 10 samples, average timestem needed for the
gridlock to occur is 20000

# > bml.sim3(25,25,0.3,10)
# [1] 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000

# > bml.sim3(25,25,0.25,10)
# [1] 20000 20000 20000 20000 20000 20000 20000 20000 20000 20000




############################## Conclusion ##############################

#### Now that we have obtained the denisities for two different sized grids, we
can make an conclusion
#### on how does the behaviour of the BML depends on the grid size and the
density.

#### Both 5x5 and 25x25 grids have similar behaviour when the densities are
bellow p=0.32.
#### In this spectrum both grids show very high likelihood that the BML will
```

```
avoid
#### grid lock which shows the results above.

#### When comparing two grids on the p=0.32 density, it seems that the bigger
grid size (25x25)
#### tends to create more frequent traffic jams than the smaller one (5x5).
Nevertheless
#### bigger grid size still tends to have a high likelihood of non-gridlocking
behaviour or
#### a very high number of iterations.

#### Comparing the densities above p=0.32, in this case p=0.35, and p=0.55, it
seems again
#### that the smaller grid sizes (5x5) tend to create less jams than the bigger
ones (25x25).
#### It is interesting though that for p=0.55 and 5x5 grid, even thou that in
many cases in our
#### sample of size 50 we had grid-lock-free system, we also recorded couple of
gridlocks
#### at a very low number of iterations. This is quite lokely due to the
randomness of the
#### car locatin on the small grid. This is an analogy when we were
experementing with the
#### same grid on the higher density (p=0.8) and still recorded a couple (4% in
our sample)
#### of systems which are grid-lock-free.

#### In addition, this bar plot of comparison between 5x5 and 25x25 in terms of
different
#### densities summarizes the behaviour of the densities and sizes of grids.

comparison.of.two.grids <-
matrix(c(20000,20000,20000,14804,20000,13171,4773,71.8),ncol=4,byrow=TRUE)
colnames(comparison.of.two.grids) <- c("p=0.3","p=0.32","p=0.35", "p=0.55")
rownames(comparison.of.two.grids) <- c("5x5","25x25")
plot.comparison <- as.table(comparison.of.two.grids)

barplot(pet.puta.pet5, beside=T, ylab="Number of Iterations", xlab="Densities",
legend.text= c("5x5", "25x25"), main="Comparison of 5x5 and 25x25 BML systems")
```