Sam Hatamian

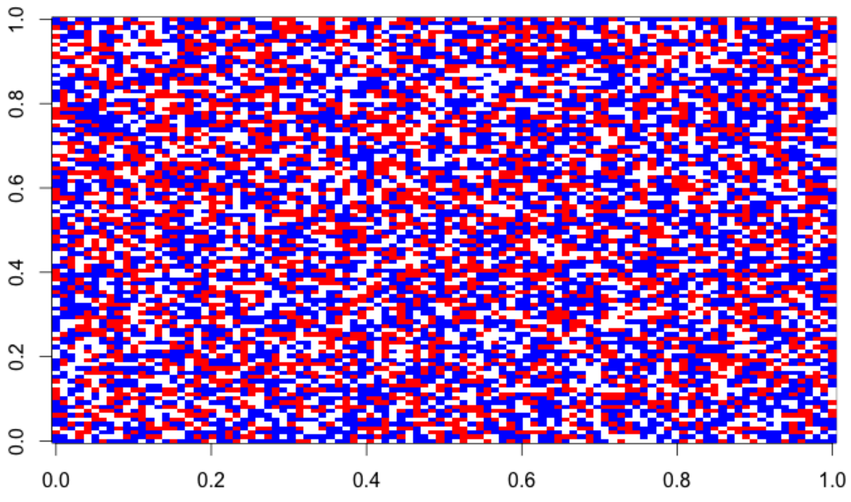
Traffic Simulation Modeling

Introduction:

The goal of this project was to create traffic simulation model that could take in a certain inputs. These inputs such as car color and size of grid were then used to run simulations in order to better understand the flow of traffic under constraints for specific car colors.

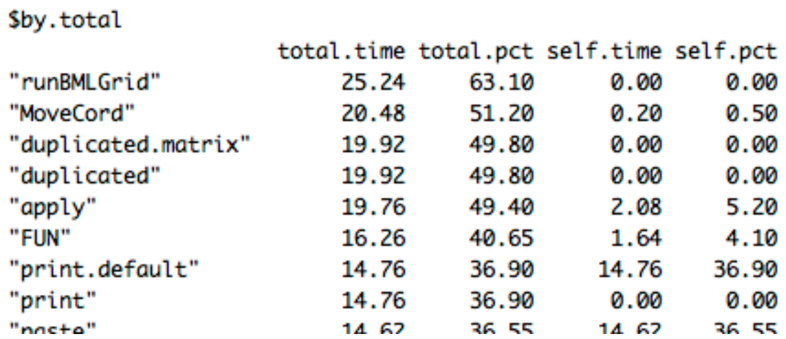
Some created functions:

Some of my main functions included createBMLGrid and runBMLGrid. The first allowed for me to create a grid with specific positions for blue and red cars randomly. The second function, runBMLGrid simulated the movement of cars for a specific period of time. runBMLGrid actually outputs a list of the positions at every single time (can be modified for just one time) so that I can then calculate number of cars moved, blocked and velocity (all individual functions that I created) at specific time intervals. Below is what my initial Grid would look like:



Optimizing functions:

I was initially using loops in my CordToPos and my PosToCord function which took either coordinates or positions and turned them into the other. However after running summary RPROF I realized that this was adding a lot of time to my function. I decided to vectorize my operations here using logical statements and this greatly decreased the time. Here is an example below with a 5000 grid and cars taking up 70% of the grid at an even split:



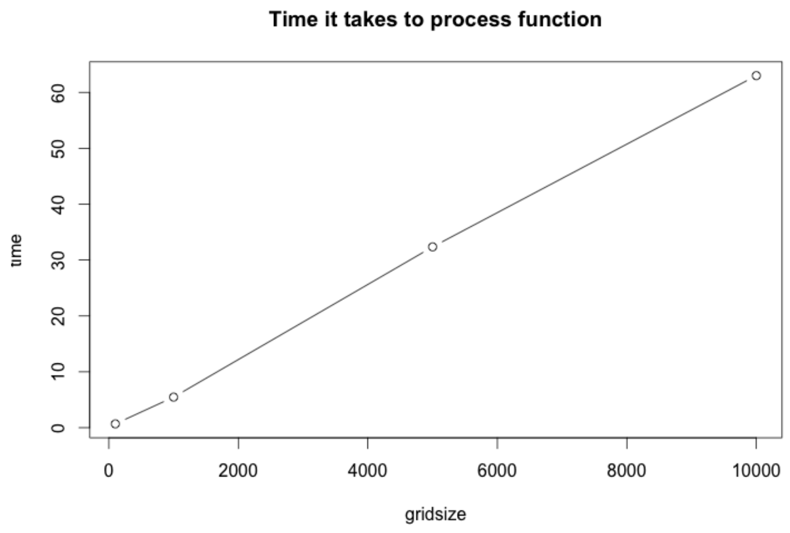
```
$by.total
                    total.time total.pct self.time self.pct
"runBMLGrid"             25.24     63.10      0.00     0.00
"MoveCord"               20.48     51.20      0.20     0.50
"duplicated.matrix"      19.92     49.80      0.00     0.00
"duplicated"             19.92     49.80      0.00     0.00
"apply"                  19.76     49.40      2.08     5.20
"FUN"                    16.26     40.65      1.64     4.10
"print.default"          14.76     36.90     14.76    36.90
"print"                  14.76     36.90      0.00     0.00
"paste"                  14.62     36.55     14.62    36.55
```

Quick Notes:

I just want to take the opportunity to mention that my runBMLGrid function saves every position at every single step! This is why the time might seem excessively long. I can easily modify the function to just take in a specific time but the reason that my function is set up this way is so that I can make a GIF that shows the plot changing over time. I will send you this GIF in the report.

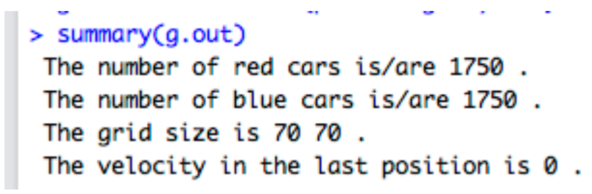
Timing of function:

I wanted to see how much function performed at various grid sizes using system.time(). Again please keep the note above in mind. I fixed the density of cars to be at 70% of the grid with the proportion of cars being at a 50-50 split. Below is the graph:



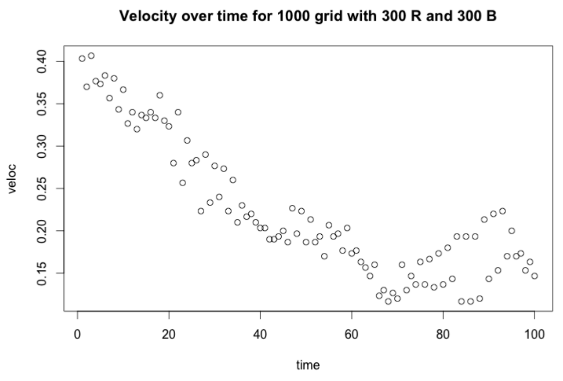
Some Tasks:

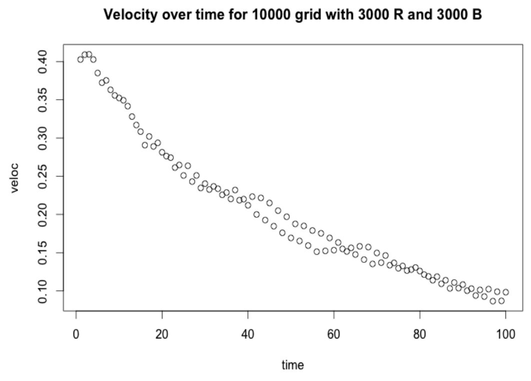
I wrote a method for the function plot and a method for the function summary that took in my output from runBMLGrid and plotted as well as summarized. My plot function actually plots all of the positions just so I can make a gif that I will send as well. The reason that I make the gif is because I want to see the pattern of how traffic moves over time. That is why I made my plot function iteratively output each grid. The summary function returns these values for the grid with parameters that I explained above:



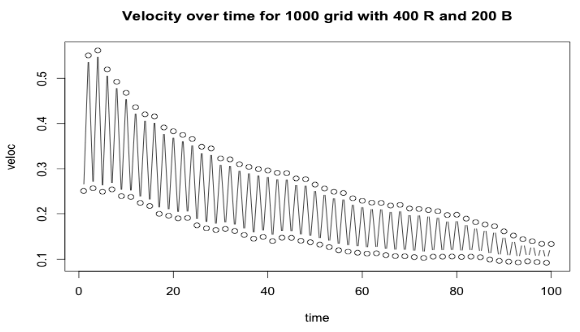
```
> summary(g.out)
 The number of red cars is/are 1750 .
 The number of blue cars is/are 1750 .
 The grid size is 70 70 .
 The velocity in the last position is 0 .
```

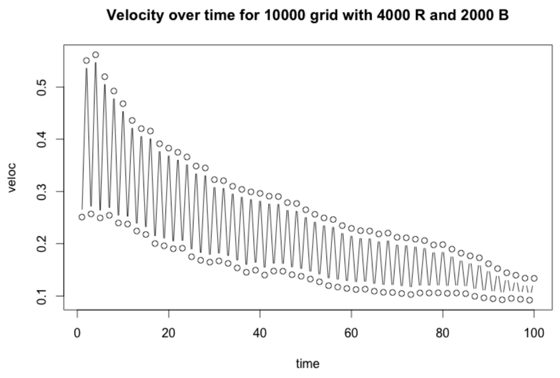
I created a blocked and moved function that shows how many cars were blocked and how many moved at a certain time period. I also did the same for velocity. I was curious to look at velocity over time for different grid sizes and proportions. I fixed my grid size with about a thousand and placed 300 red and blue cars. Below is the velocity over a time period of 100:





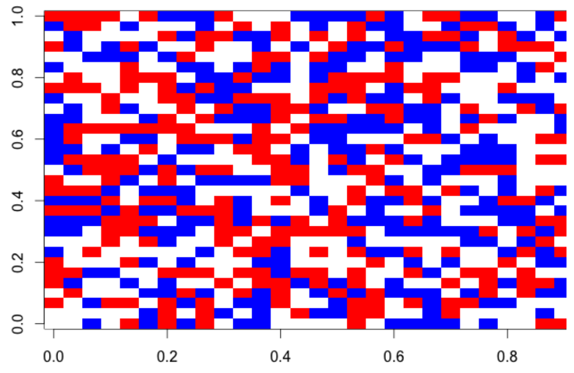
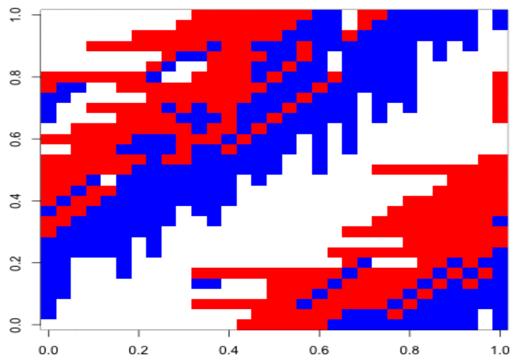
Now I double my proportion of reds but maintain my proportion of blues:



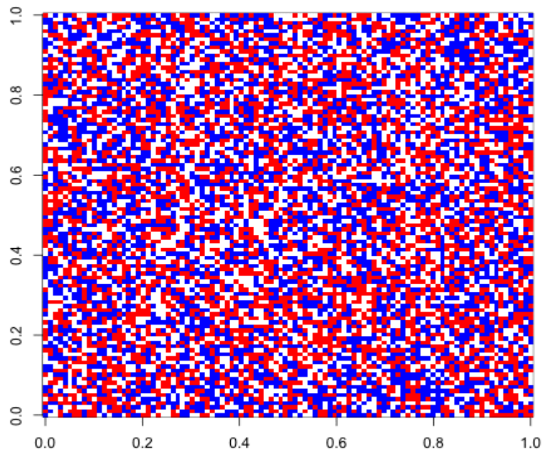
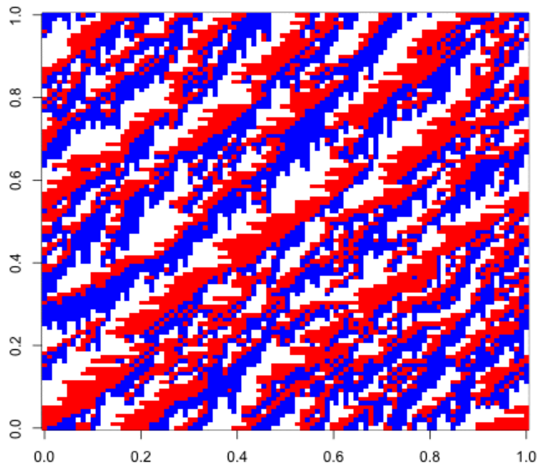


It is clear that the number of cars for a certain type is directly proportional to velocity. The more cars you have of one color the more chances you are going to have of getting blocked. Above we see when we double red that the velocity at each time period is almost double for red then blue cars. We also see that as we continue our simulation model the velocity decreases with time as well. However when the grid size is larger we see that this decrease is a lot more smooth.

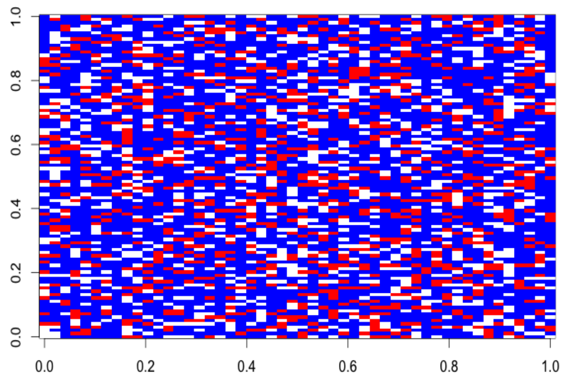
Now I ran a couple of simulations. NOTE: I programmed my cars so that if you have two red cars followed by a blank (R R BLANK) the move transitions into R BLANK R. Same applies for blue. The first was for a 1000 grid with 300 of each color over a 500 time interval. Here is the start and stop:

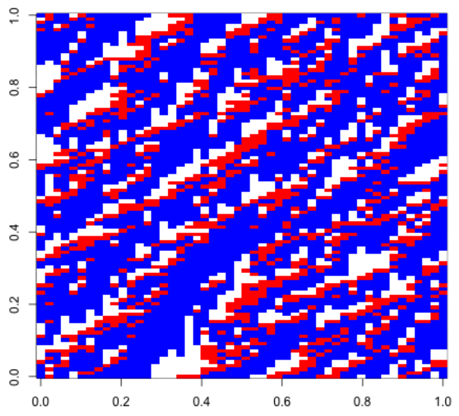



I did the same thing but with a grid of 10,000 and this is the result that I got.

We can see that the cars falls into this diagonal lines were reds can no longer move right and blue lines can no longer move up. Running this simulation over and over again provided similar results in shape. I also wanted to switch the proportions and shape of the grid. I changed the shape of my grid to a 100x50 and doubled my proportion of blue to red cars.

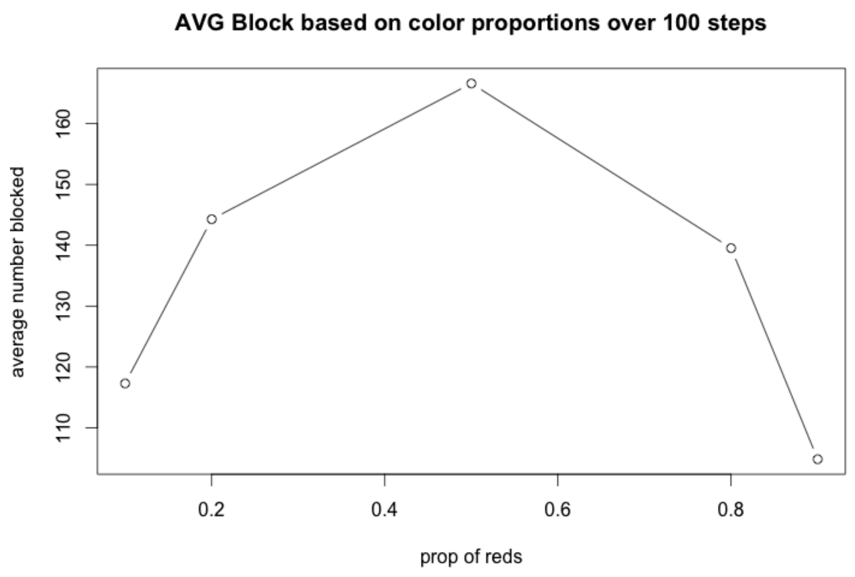




Again we see the same diagonal type of pattern occur but when we have a larger proportion of blues, they cluster together and are soon blocked by the reds on top similar to our previous image. The change in grid size was not extremely noticeable in the outcome compared to the previous grid size that we used.

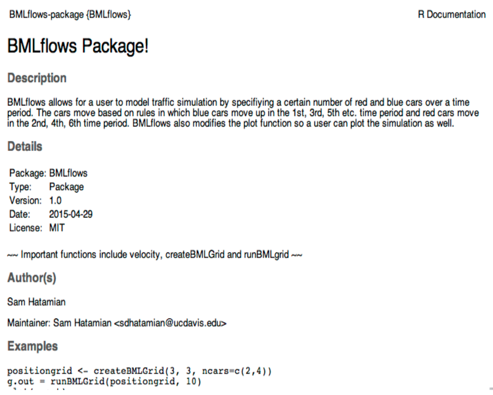
Looking at blocking:

I also wanted to look at fixing the grid and changing the proportions of red and blue in order to see how blocking average blocking over a certain time changed. We see that as you start with a low proportion of one and a higher of the other, average blocking starts low but increases as the proportions even out. Then as the proportions again begin to skew after you pass 50 percent mark, blocking once again decreases. It is clear that the cars of the same color with higher proportions experience less blocking (less diagonal lines that we mentioned above) but cluster together (as shown above) until at some point, limited movement is experienced. Below is the graph for this. Note that this is a 1000 space grid with total number of cars occupying 50% of the grid.



Library:

I also created a package where I stored all of these functions with documentation, called BMLflows. I created help documentation for functions including velocity(), blocked(), moved(), createBMLGrid() and runBMLgrid(). Below is an example:



BMLflows-package {BMLflows} R Documentation

BMLflows Package!

Description

BMLflows allows for a user to model traffic simulation by specifiying a certain number of red and blue cars over a time period. The cars move based on rules in which blue cars move up in the 1st, 3rd, 5th etc. time period and red cars move in the 2nd, 4th, 6th time period. BMLflows also modifies the plot function so a user can plot the simulation as well.

Details

Package:	BMLflows
Type:	Package
Version:	1.0
Date:	2015-04-29
License:	MIT

~~ Important functions include velocity, createBMLGrid and runBMLgrid ~~

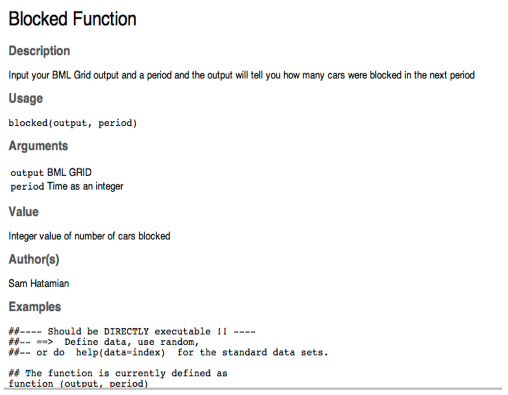
Author(s)

Sam Hatamian

Maintainer: Sam Hatamian <sdhatamian@ucdavis.edu>

Examples

```
positiongrid <- createBMLGrid(3, 3, ncars=c(2,4))
g.out = runBMLGrid(positiongrid, 10)
```



Blocked Function

Description

Input your BML Grid output and a period and the output will tell you how many cars were blocked in the next period

Usage

```
blocked(output, period)
```

Arguments

`output` BML GRID
`period` Time as an integer

Value

Integer value of number of cars blocked

Author(s)

Sam Hatamian

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##-- or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (output, period)
```