

STA242 - HW02 - BML Traffic Model

Hugh Crockford

February 4, 2013

1 Simulation

The effect of various levels of ρ , percent each car color, and size of map was investigated on the average velocity (Percent of cars moved that were supposed to).

1.1 Effect of ρ on Velocity.

Below plot shows the effect of rho (percent of available places covered by a car) on average velocity.

When the percent of map covered is low (between 0.1-0.3) the cars are fairly free to move and initial velocity is high. The Velocity increases fairly quickly to approach 1 as the cars form orderly diagonal patterns, allowing each group (red and blue) to move in step.

When ρ is around 0.4, the velocity is fairly constant with jams developing and resolving at random areas on the grid area. Higher levels of ρ quickly develop diagonal jams with layers of red and blue cars being unable to move.(See below)

1.2 Effect of map size on Velocity

Map size also had an impact on average velocity (for a fixed ρ), with smaller maps having lower average velocities due to conflicts occurring more often.

1.3 Effect of percent color mix on Velocity

The color mix had a small effect on average velocity, mostly at the beginning of the simulation, before the cars developed their ordered diagonal positions. (This graph is concerning as I would've thought 10 percent red should be the same as 90 percent red. I've had a good look at my code and cannot find the error.)

It would be interesting to investigate the combined effect of these three variables and assess their covariances (? - e.g. what if map is small, high rho but few red cars?)

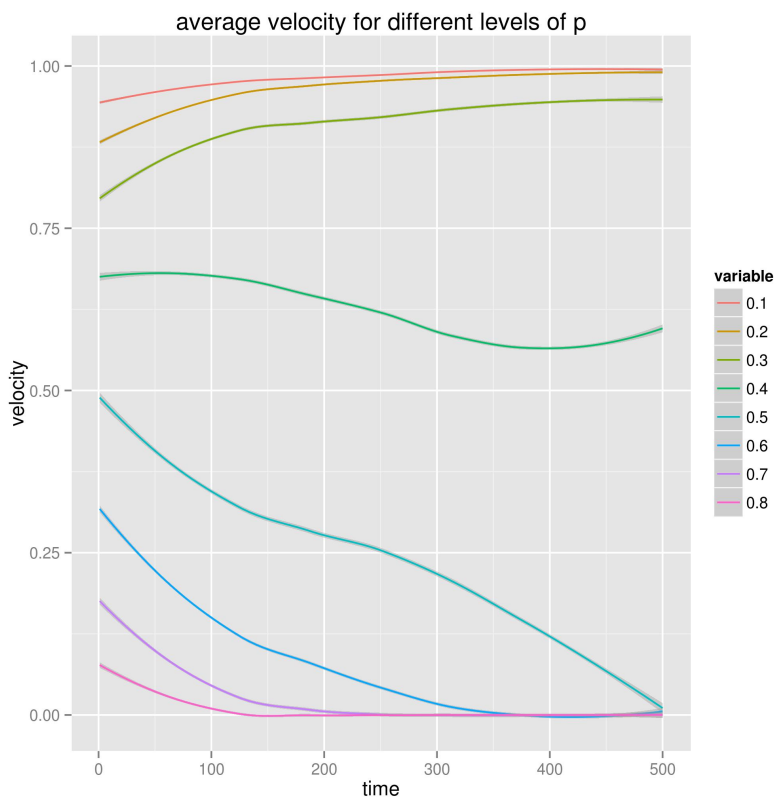
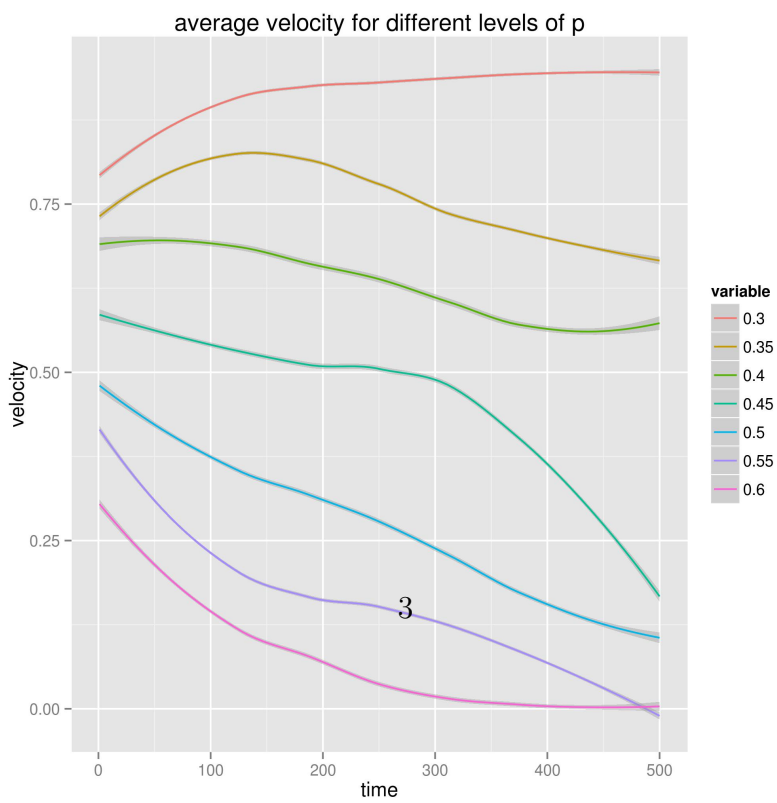


Figure 1: effect of ρ on average velocity



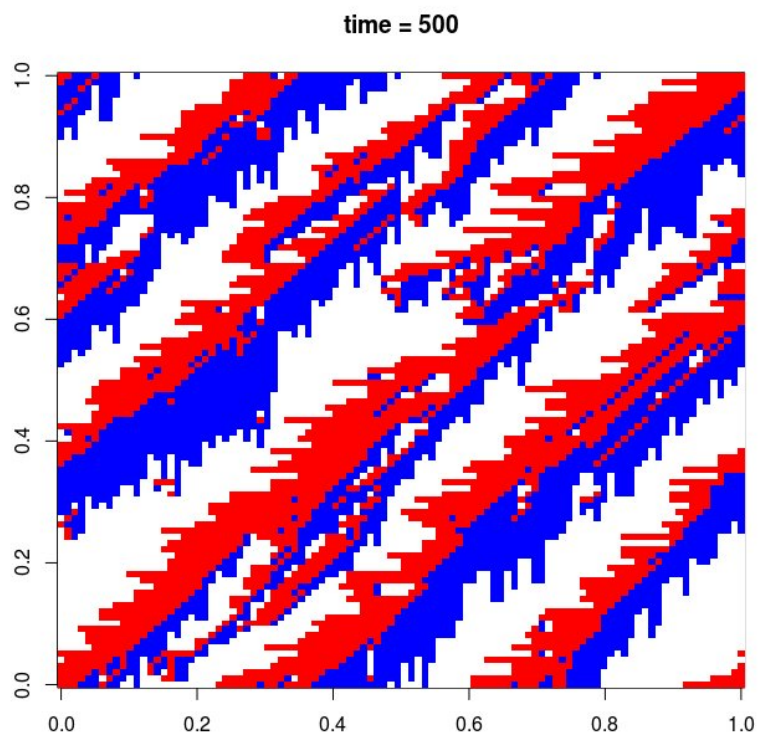


Figure 3: Typical jam pattern with high ρ

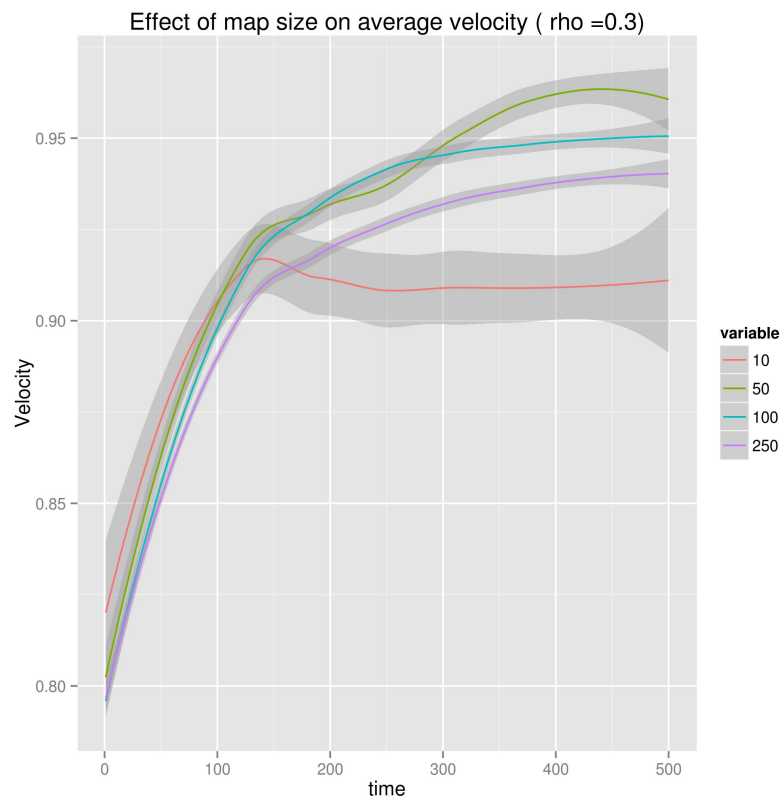


Figure 4: Effect of map size on average velocity

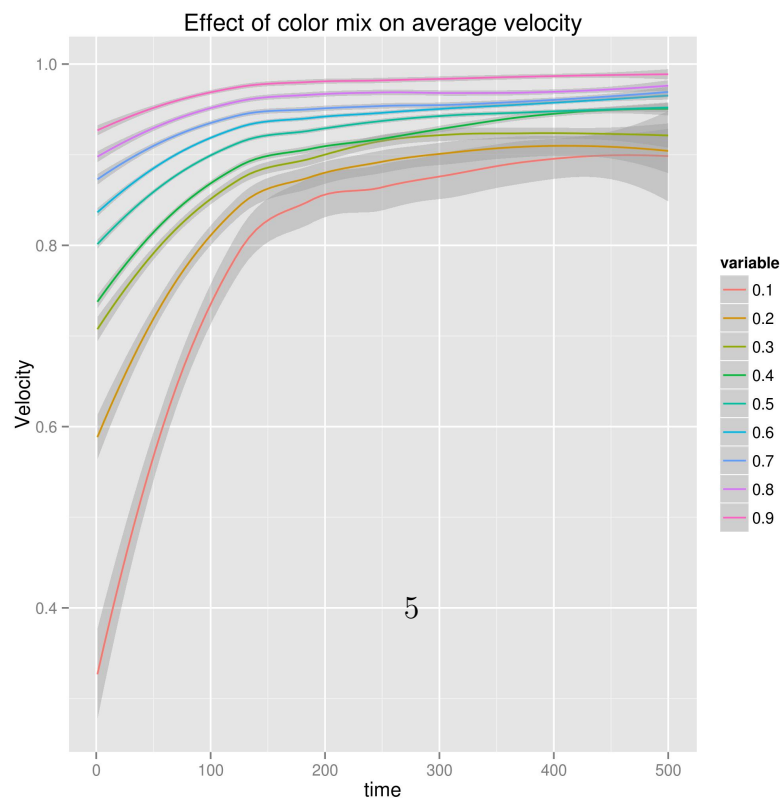


Figure 5: Effect of color mix on average velocity

2 Methods

Below is the output of plot and summary methods for my 'bml' class, as well as the dynamic plot.vhs that replays the simulation.

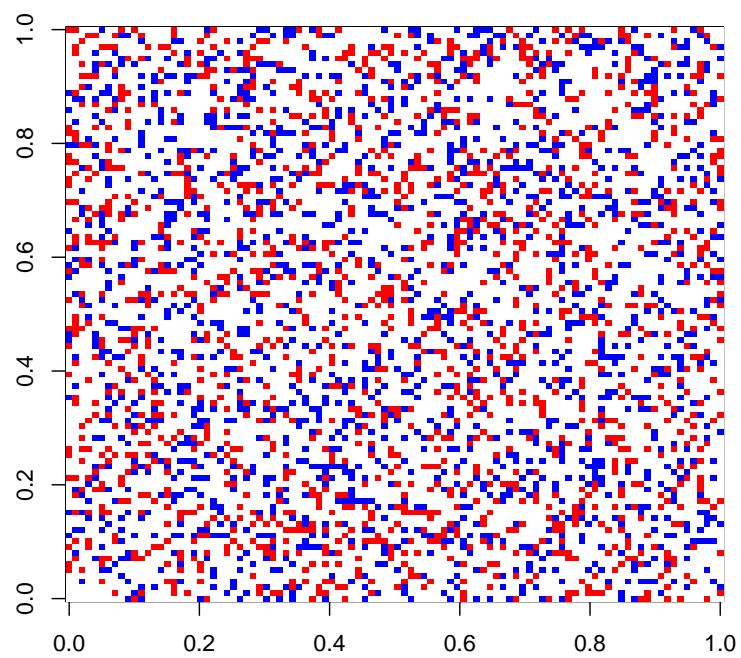
```
> source('BMLFn.R')
> m = map(0.3,.5,100,100)
> summary(m)

$dim
[1] "dimensions: 100 rows X 100 columns"

$n
[1] "number of cars: 3000"

$p
[1] "p = 0.3"

> plot(m)
> simulation = play(m,100)
> plot(simulation)
```



3 Code Profiling

3.1 Speeding up Functions

Investigating my functions with Rprof revealed much time spent in a call to paste in move function. This was leftover from another method of testing conflicts that I had tried out, but was not even used in the final version of move. Removing this call to paste significantly improved the speed of execution, however paste was still top of the by.self table, even though it did not occur in any of functions - perhaps it is being used internally by something?

```
> head(summaryRprof('testing.out')$by.self)           # initial version
```

	self.time	self.pct	total.time	total.pct
"paste"	12.94	65.55	13.02	65.96
"make.unique"	2.28	11.55	2.28	11.55
"matrix"	0.94	4.76	0.96	4.86
"print.default"	0.62	3.14	0.62	3.14
"[.data.frame"	0.48	2.43	0.96	4.86
"row.names.data.frame"	0.36	1.82	0.36	1.82

```
> head(summaryRprof('testing2.out')$by.self)          # after removing superfluous
```

	self.time	self.pct	total.time	total.pct
"paste"	6.80	54.23	6.82	54.39
"make.unique"	1.98	15.79	1.98	15.79
"matrix"	0.88	7.02	0.88	7.02
"row.names.data.frame"	0.50	3.99	0.50	3.99
"[.data.frame"	0.42	3.35	0.88	7.02
"rbind"	0.28	2.23	2.48	19.78

Considerable time was also spent in the call to make.unique, which I assume is part of 'duplicated', which I was using to test conflicts. I could not find another method of testing conflicts that did not use duplicated. (I tried pasting character strings and testing %in%, complicated logic statements, and table methods to test conflicts between new and old states.)

3.2 Effect of various parameters on speed of execution

The effect of map size, ρ , and number of times simulated on speed of execution was investigated, to see if the functions followed a linear or log time relationship. As can be seen on the below graph, increasing each of these parameters resulted in a linear increase in time taken, indicating code is well vectorised. <++>

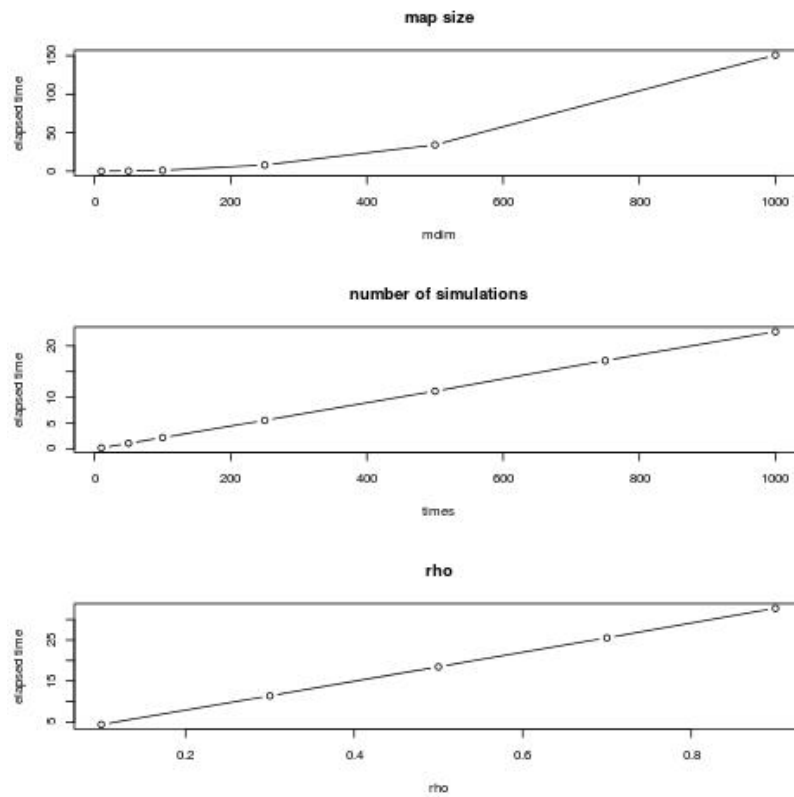


Figure 6: effect of various parameters on running time

4 CODE

```
_report.R"

# STA242 – HW02 – BML traffic model
# 20130125 – Hugh Crockford

library(reshape2)
library(ggplot2)
source("BMLFn.R")

m = map(.2,.5,100,100)
y = play(m,500)
plot(y)
plot(y$vel)

# velocities of different p
t = 500
p = seq(0.1,.8,.1)
o = sapply(p,function(i) play(map(i,.5,100,100),t))
      # returns list of vhs w diff
q = data.frame(1:t,o[2,])
names(q) = c("time", p)
m = melt(q , id.vars = "time")
qplot(data = m , x=time,y=value,color = variable , ylab
      = "velocity", main = "average velocity for different
      levels of p",geom = "smooth")      # vel drops off
      at ~ p = 0.4
ggsave("rhov.jpg")

four = o[,4]
class(four) = "vhs"
plot(four)

five = o[,5]
class(five) = "vhs"
```

```

plot(five)

six = o[,6]
class(six) = "vhs"
plot(six)          # locks up ~ 400

# investigate this more
t = 500
p = seq(0.3,.6,.05)
o = sapply(p,function(i) play(map(i,.5,100,100),t))
q = data.frame(1:t,o[2,])
names(q) = c("time", p)
m = melt(q , id.vars = "time")
qplot(data = m , x=time,y=value,color = variable , ylab
      = "velocity", main = "average velocity for different
      levels of p",geom = "smooth")          # vel drops off
      at ~ p = 0.4
ggsave("rhov22.jpg")

# effect of map size.
t = 500
mdim = c(10,50,100,250)
o = sapply(mdim,function(i) play(map(0.3,.5,i,i),t))
q = data.frame(1:t,o[2,])
names(q) = c("time", mdim)
m = melt(q , id.vars = "time")
qplot(data = m , x=time,y=value,color = variable , geom
      = "smooth",main = "Effect of map size on average
      velocity ( rho =0.3)", ylab = "Velocity")
ggsave("sizev.jpg")

# velocities of different percent red/blue.
t = 500
p = seq(0.1,.9,.1)
o = sapply(p,function(i) play(map(.3,i,100,100),t))
q = data.frame(1:t,o[2,])
names(q) = c("time", p)

```

```

m = melt(q , id.vars = "time")
qplot(data = m , x=time,y=value,color = variable , geom
      = "smooth", main = "Effect of color mix on average
      velocity",ylab = "Velocity")
ggsave("colorv.jpg")

one = o[,1]
class(one) = "vhs"
plot(one)

# code testing

Rprof("testing.out")
t = play(map(0.3,.5,100,100),500)
Rprof(NULL)
head(summaryRprof("testing.out")$by.self)

Rprof("testing2.out")
t = play(map(0.3,.5,100,100),500)
Rprof(NULL)
summaryRprof("testing2.out")
summaryRprof("testing2.out")$by.self

# is it size map or replications (t) that slow it up?
DO AFTER A REBOOT.

mdim = c(10,50,100,250,500,1000)
time.size = sapply(mdim, function(i) system.time({t=
  play(map(0.3,.5,i,i),50)})))
times = c(10,50,100,seq(250,1000,250))
time.time = sapply(times, function(i) system.time({t=
  play(map(0.3,.5,100,100),i)})))
rho = seq(0.1,.9,.2)
tim.rho = sapply(rho, function(i) system.time({t=play(
  map(i,.5,100,100),500)})))

```

```

qplot(mdim,time.size[3,],geom="line")
qplot(times,time.time[3,],geom="line")
qplot(rho,tim.rho[3,],geom="line")

# do plots of size, times, rho vs time.

library(profr)
out = profr(t = play(map(0.3,.5,100,100),500))

```