HW4

Shaoqian Chen 8831737894

3/30/2020

```
library(contextual)
```

```
visits = 250
simulations = 1000
```

Question 1

```
# define 1 options give a reward 10% of time, one option give a reward 30% of the time,
 one option give a reward 50% of the time
prob_per_arm = c(0.1,0.9)
bandit = BasicBernoulliBandit$new(prob_per_arm)
# define policies (they differ by epsilon value)
agents = list(Agent$new(EpsilonGreedyPolicy$new(0.1),bandit,'Epsilon = 0.1'),
              Agent$new(EpsilonGreedyPolicy$new(0.3),bandit,'Epsilon = 0.3'),
              Agent$new(EpsilonGreedyPolicy$new(0.5),bandit,'Epsilon = 0.5'))
simulation = Simulator$new(agents,visits,simulations)
history = simulation$run()
```

```
## Setting up parallel backend.
```

```
## Cores available: 8
```

```
## Workers assigned: 7
```

```
## Simulation horizon: 250
```

```
## Number of simulations: 3000
```

```
## Number of batches: 7
```

```
## Starting main loop.
```

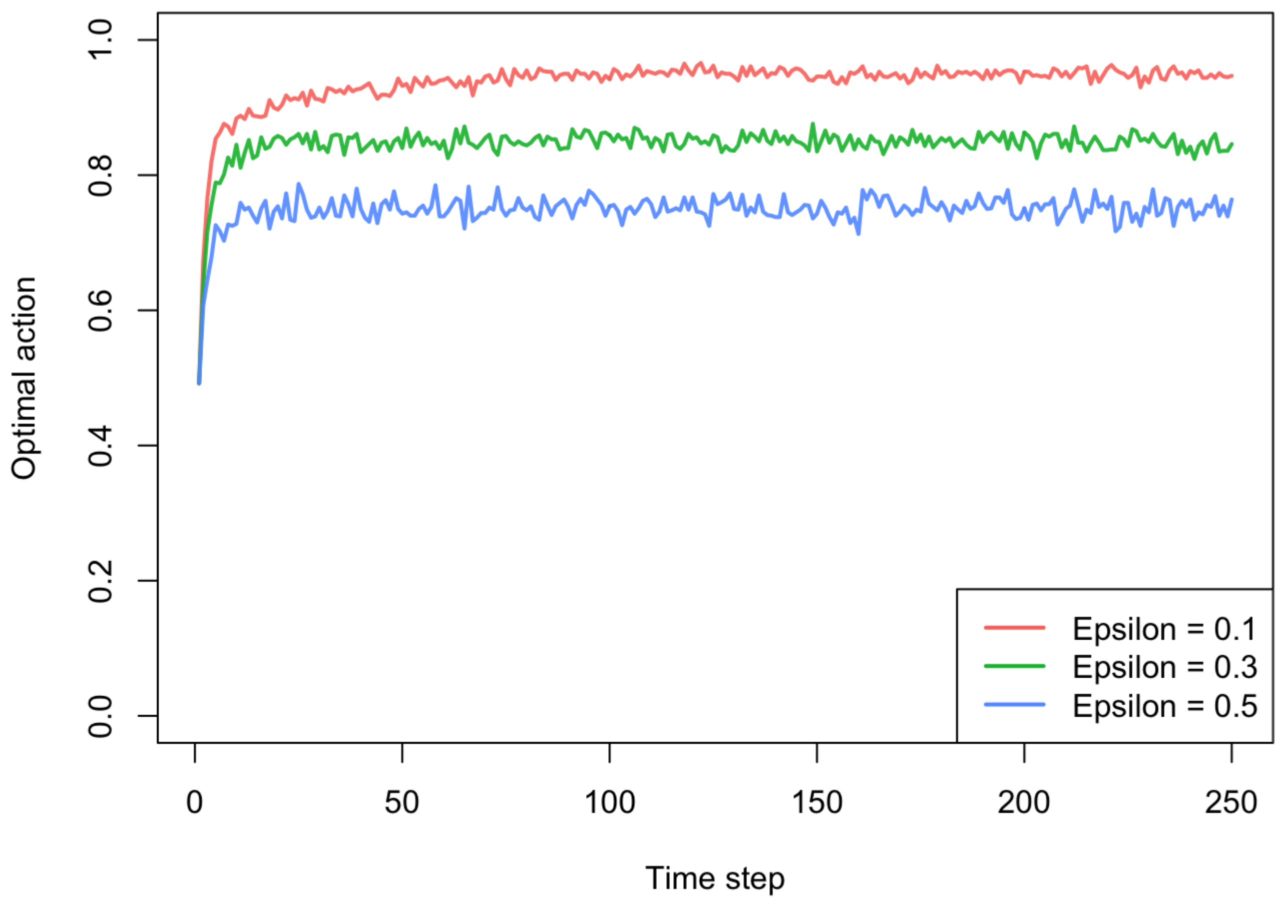
```
## Finished main loop.
```

```
## Completed simulation in 0:00:15.507
```

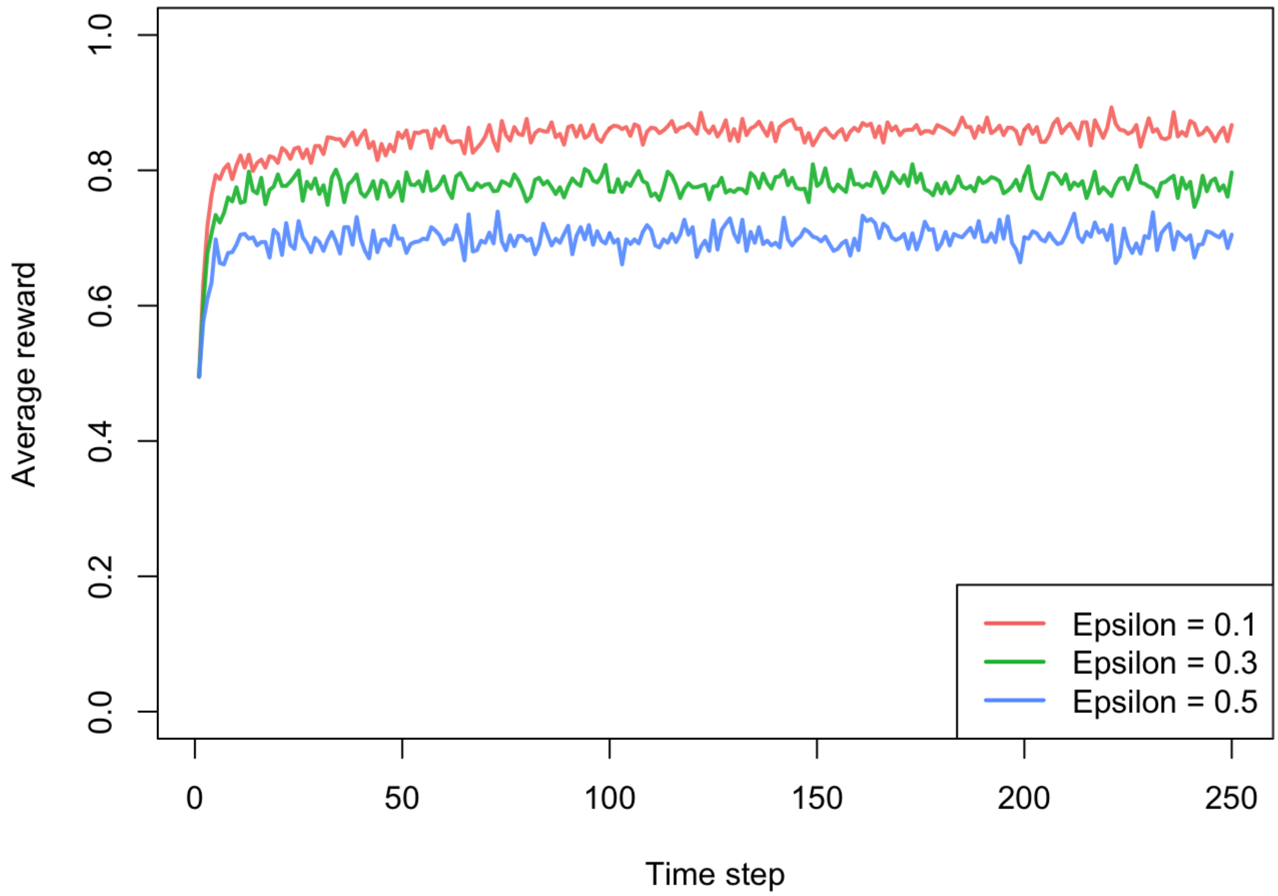
```
## Computing statistics.
```

Plot for 2 arms

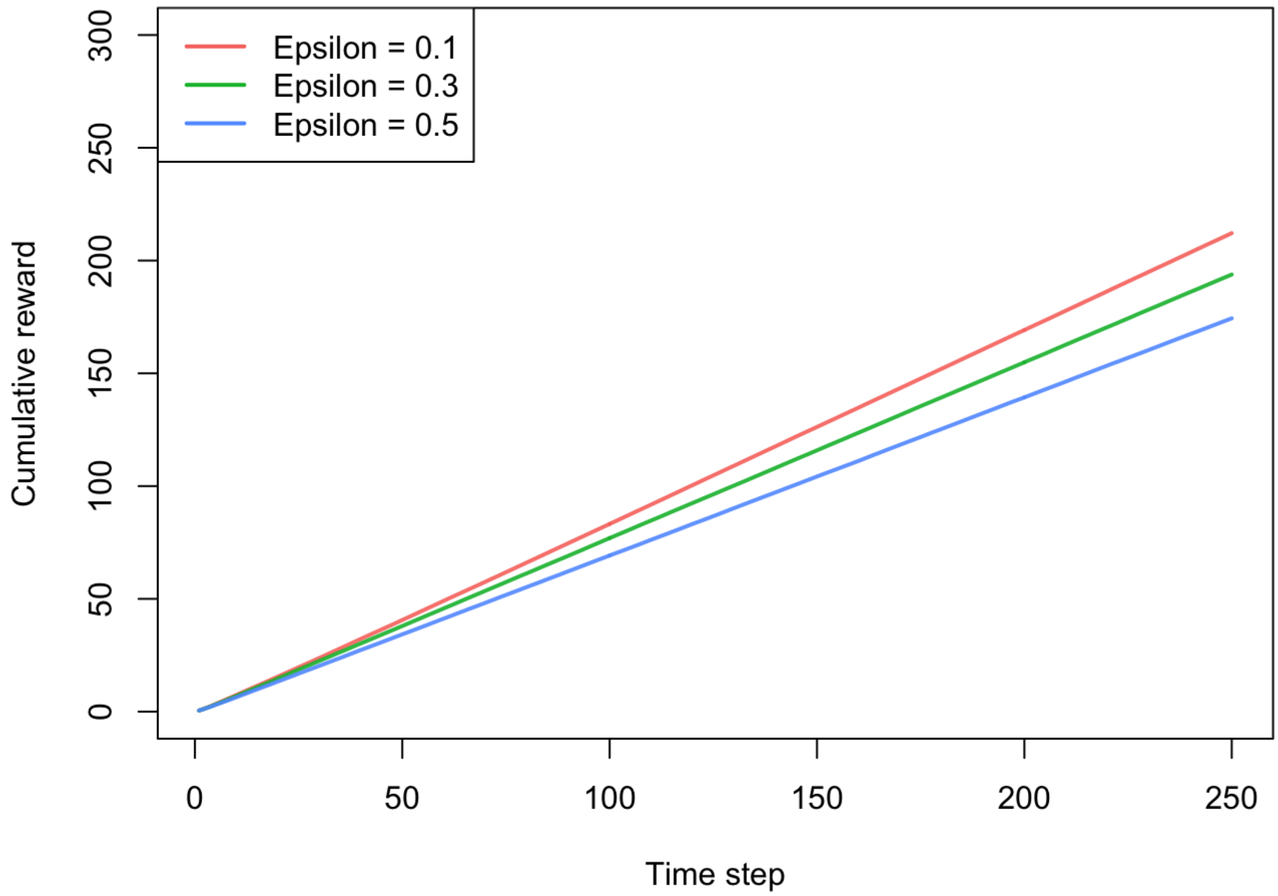
```
# probability of selecting the best design (arm)
# fraction of times the algorithm chooses best design
plot(history,type = 'optimal',legend_position='bottomright', ylim = c(0,1))
```




```
#
# increasing curve means the algorithm is learning
#
# average reward at each visit
plot(history,type = 'average',regret = F, legend_position='bottomright', ylim = c(0,1))
```




```
# fix legend overlay
plot(history,type = 'cumulative',regret =F,ylim = c(0,300))
```




```
visits = 250
simulations = 1000
# define 1 options give a reward 10% of time, one option give a reward 30% of the time,
 one option give a reward 50% of the time
prob_per_arm = c(0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1
,0.1,0.9)
bandit = BasicBernoulliBandit$new(prob_per_arm)
# define policies (they differ by epsilon value)
agents = list(Agent$new(EpsilonGreedyPolicy$new(0.1),bandit,'Epsilon = 0.1'),
              Agent$new(EpsilonGreedyPolicy$new(0.3),bandit,'Epsilon = 0.3'),
              Agent$new(EpsilonGreedyPolicy$new(0.5),bandit,'Epsilon = 0.5'))
simulation = Simulator$new(agents,visits,simulations)
history = simulation$run()
```

```
## Setting up parallel backend.
```

```
## Cores available: 8
```

```
## Workers assigned: 7
```

```
## Simulation horizon: 250
```

```
## Number of simulations: 3000
```

```
## Number of batches: 7
```

```
## Starting main loop.
```

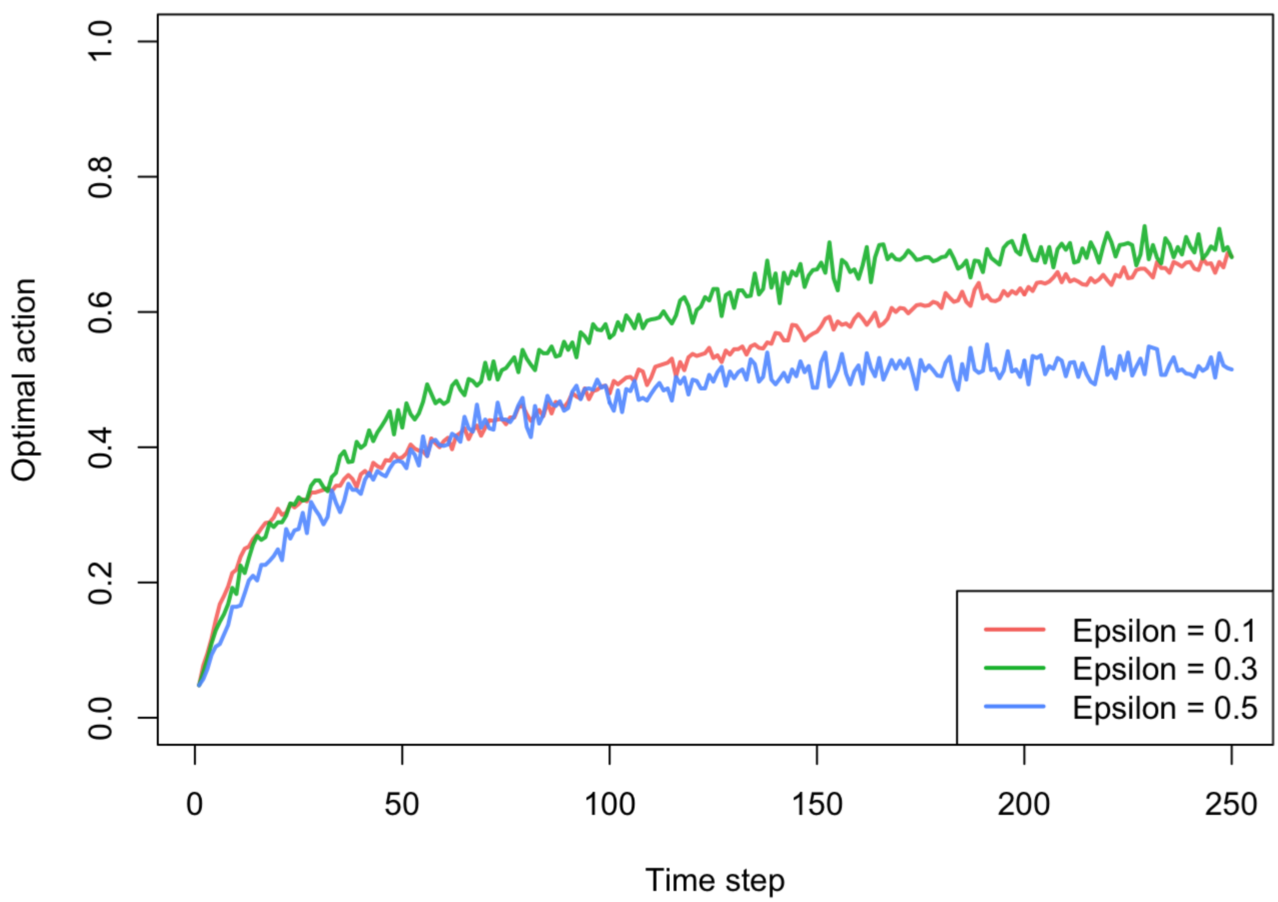
```
## Finished main loop.
```

```
## Completed simulation in 0:00:19.863
```

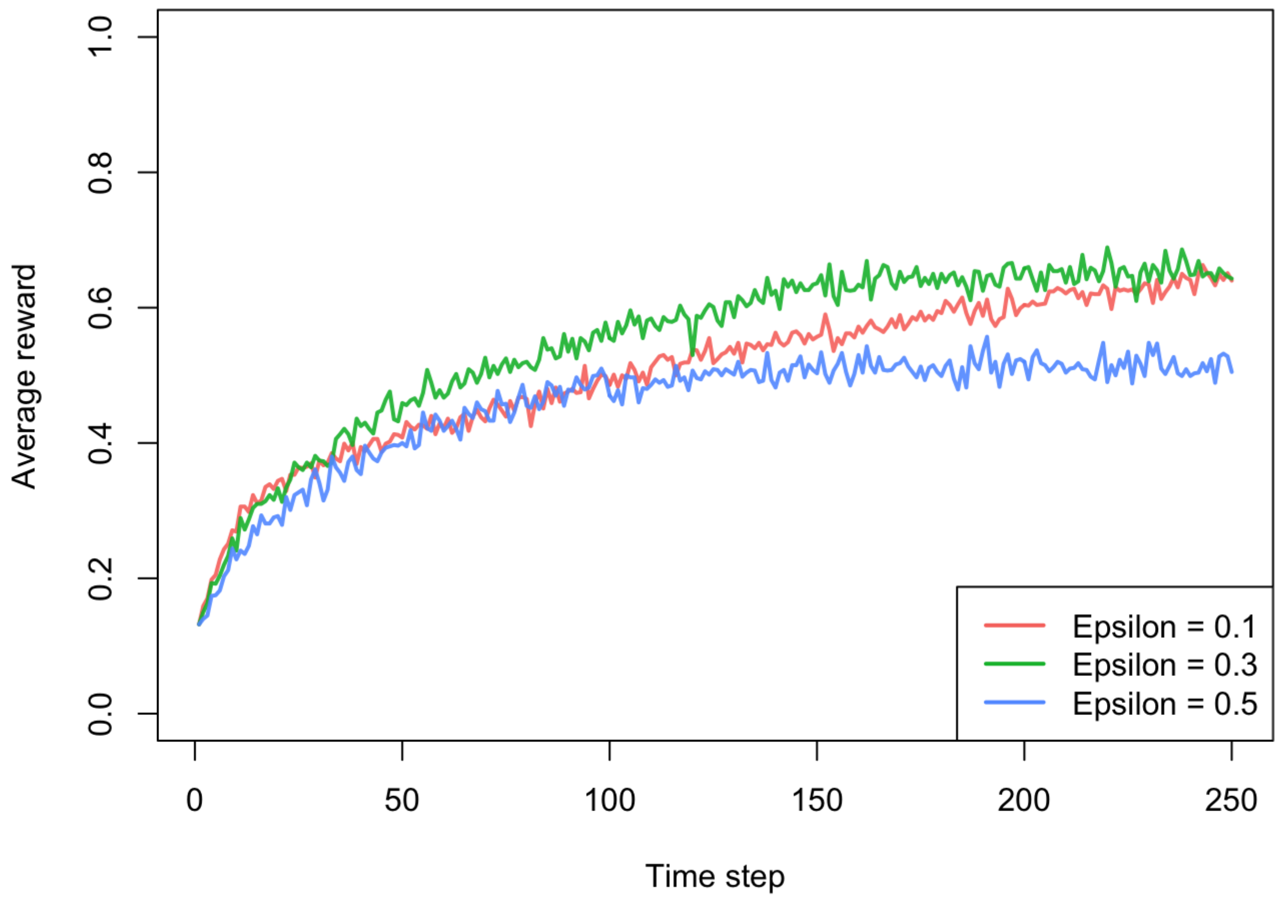
```
## Computing statistics.
```

Plot for 20 arms

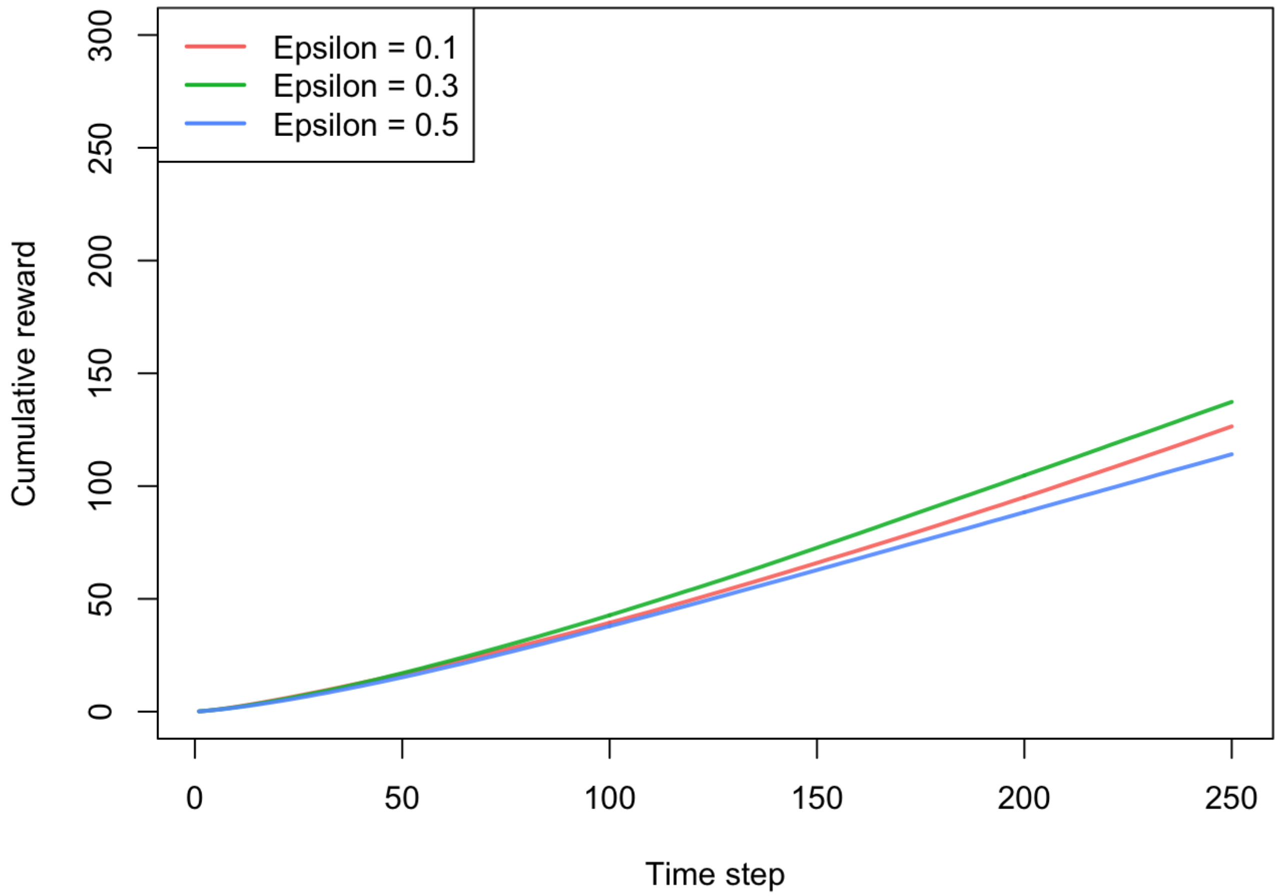
```
# probability of selecting the best design (arm)
# fraction of times the algorithm chooses best design
plot(history,type = 'optimal',legend_position='bottomright', ylim = c(0,1))
```



```
#
# increasing curve means the algorithm is learning
#
# average reward at each visit
plot(history,type = 'average',regret = F, legend_position='bottomright', ylim = c(0,1))
```




```
plot(history,type = 'cumulative',regret =F,ylim = c(0,300))
```




```
#
```

Question 2

(a)

```
# define 4 options give a reward 10% of time, and one option give a reward 90% of the ti
me
prob_per_arm = c(0.4,0.4,0.4,0.4,0.6)
bandit = BasicBernoulliBandit$new(prob_per_arm)

# define policies (they differ by epsilon value)
agents = list(Agent$new(EpsilonGreedyPolicy$new(0.1),bandit,'Epsilon = 0.1'),
              Agent$new(EpsilonGreedyPolicy$new(0.3),bandit,'Epsilon = 0.3'),
              Agent$new(EpsilonGreedyPolicy$new(0.5),bandit,'Epsilon = 0.5'))
simulation = Simulator$new(agents,visits,simulations)
history = simulation$run()
```

```
## Setting up parallel backend.
```

```
## Cores available: 8
```

```
## Workers assigned: 7
```

```
## Simulation horizon: 250
```

```
## Number of simulations: 3000
```

```
## Number of batches: 7
```

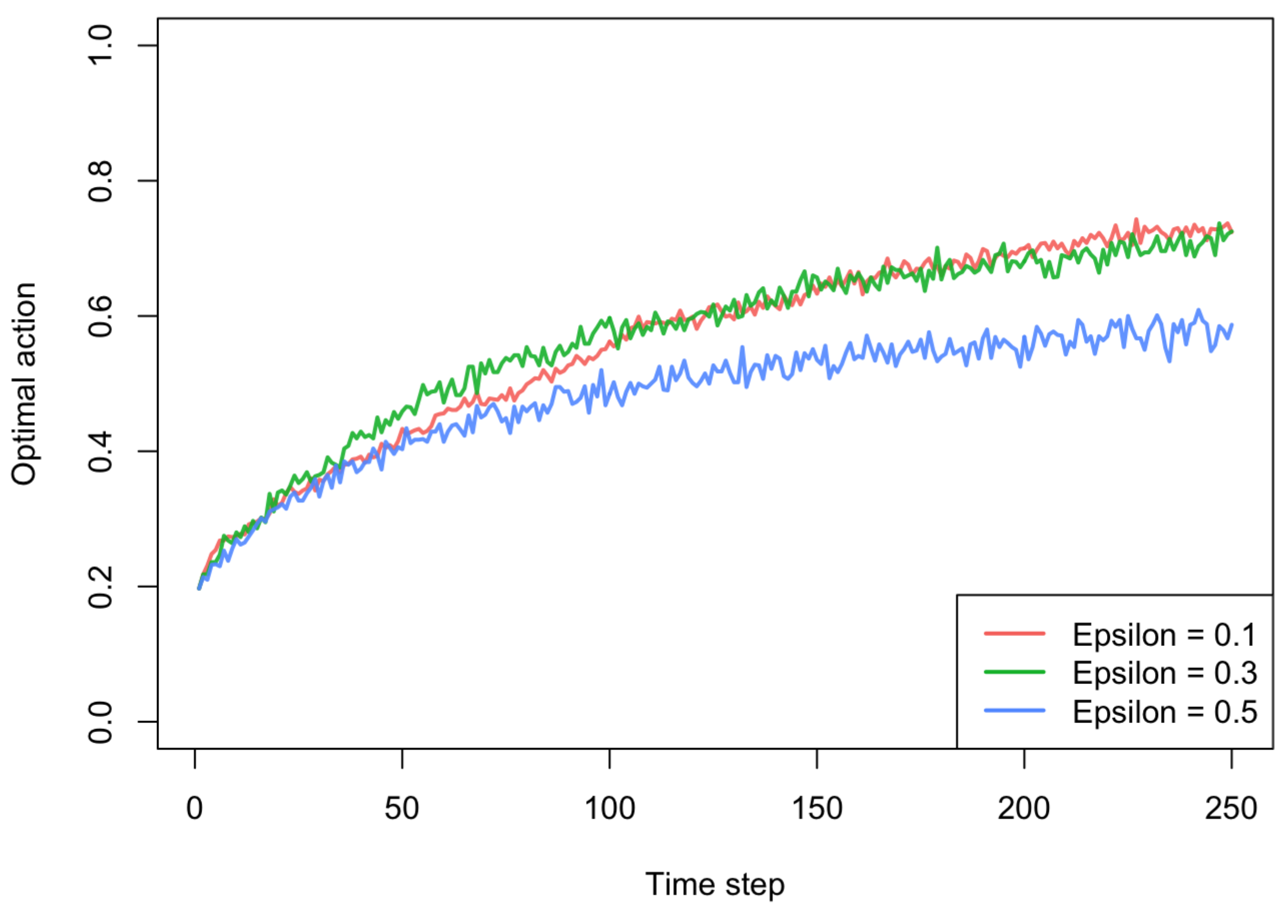
```
## Starting main loop.
```

```
## Finished main loop.
```

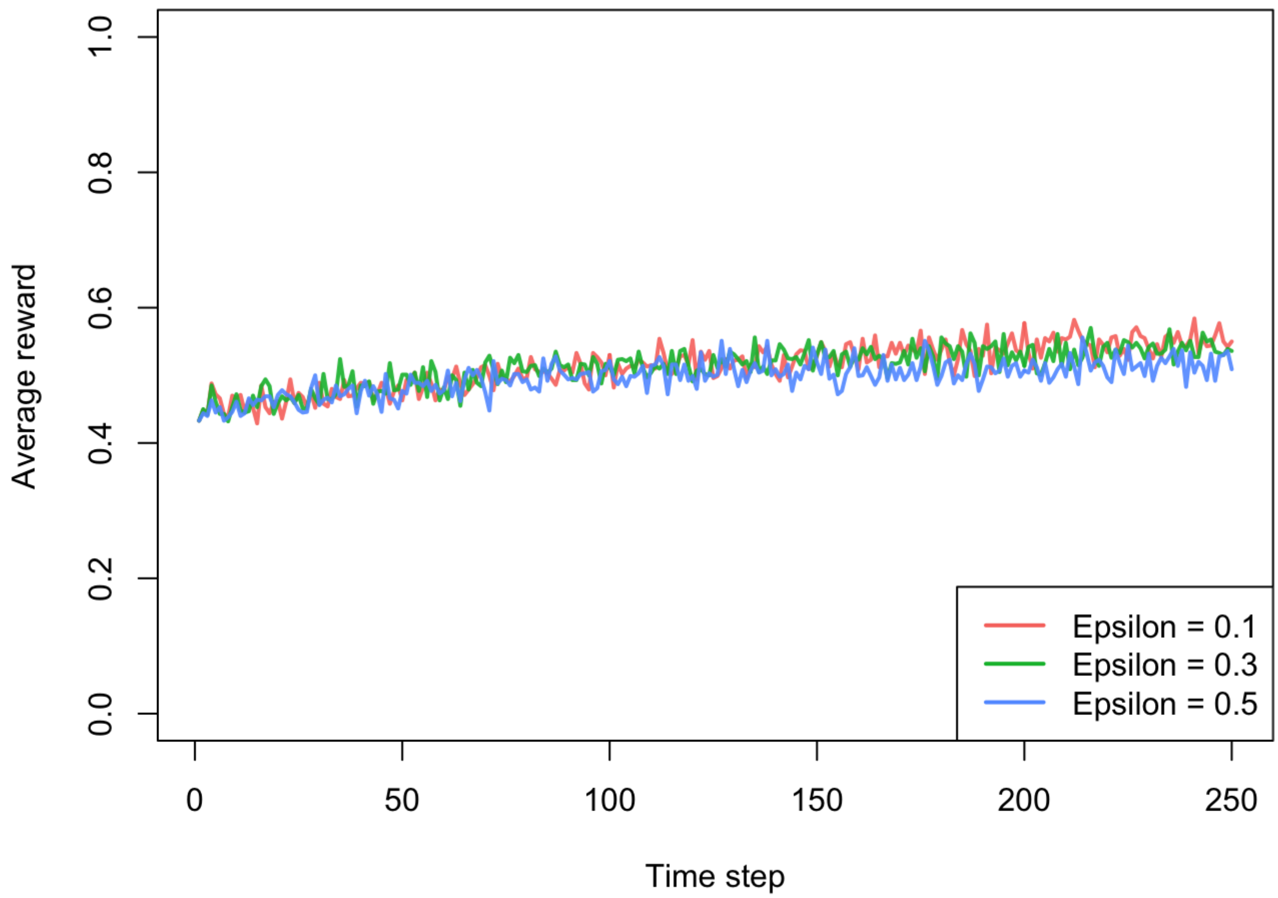
```
## Completed simulation in 0:00:16.572
```

```
## Computing statistics.
```

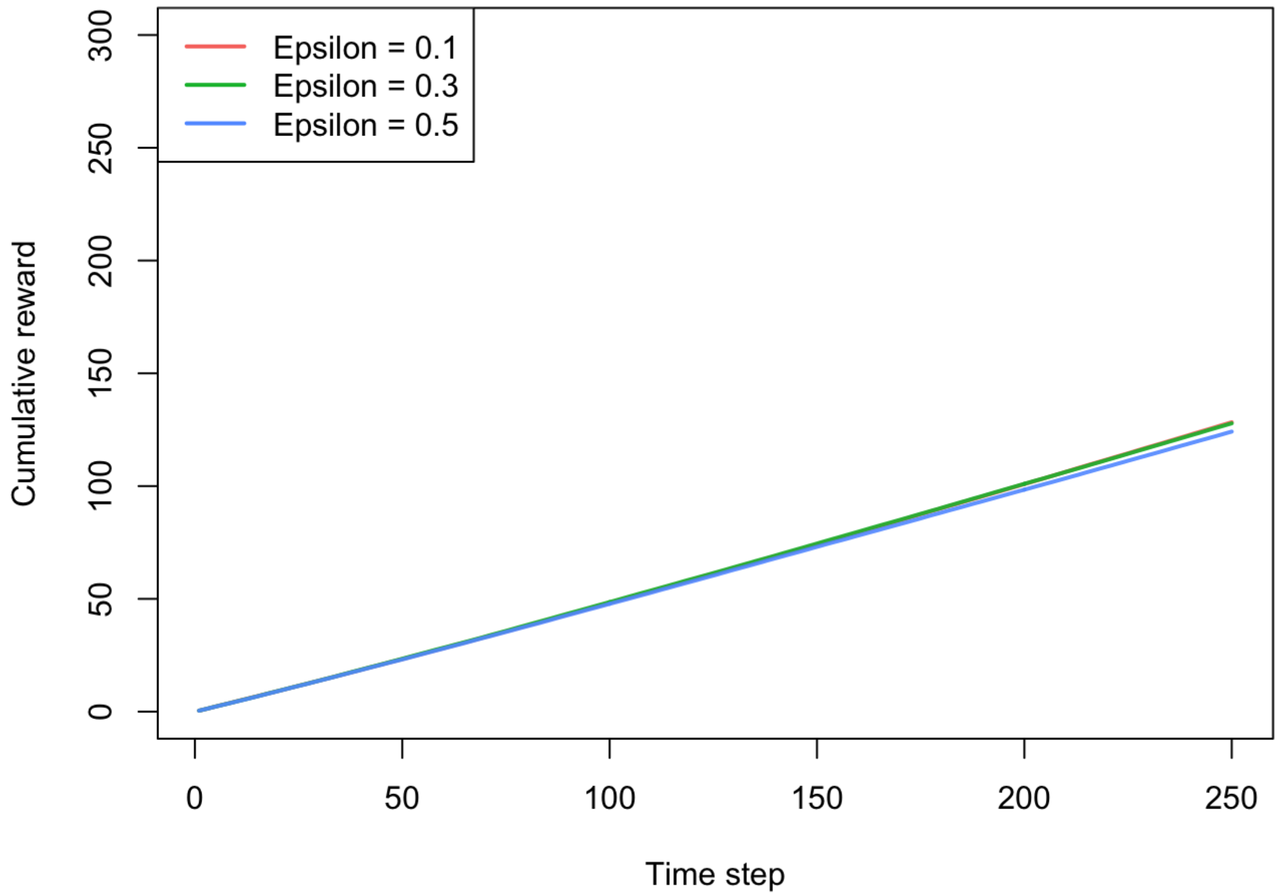
```
# probability of selecting the best design (arm)
# fraction of times the algorithm chooses best design
plot(history,type = 'optimal',legend_position='bottomright', ylim = c(0,1))
```



```
#
# increasing curve means the algorithm is learning
#
# average reward at each visit
plot(history,type = 'average',regret = F, legend_position='bottomright', ylim = c(0,1))
```




```
# fix legend overlay
plot(history,type = 'cumulative',regret =F,ylim = c(0,300))
```




```
#
```

(b)

```
# define 4 options give a reward 10% of time, and one option give a reward 90% of the ti
me
prob_per_arm = c(0.4,0.4,0.5,0.6,0.6)
bandit = BasicBernoulliBandit$new(prob_per_arm)

# define policies (they differ by epsilon value)
agents = list(Agent$new(EpsilonGreedyPolicy$new(0.1),bandit,'Epsilon = 0.1'),
              Agent$new(EpsilonGreedyPolicy$new(0.3),bandit,'Epsilon = 0.3'),
              Agent$new(EpsilonGreedyPolicy$new(0.5),bandit,'Epsilon = 0.5'))
simulation = Simulator$new(agents,visits,simulations)
history = simulation$run()
```

```
## Setting up parallel backend.
```

```
## Cores available: 8
```

```
## Workers assigned: 7
```

```
## Simulation horizon: 250
```

```
## Number of simulations: 3000
```

```
## Number of batches: 7
```

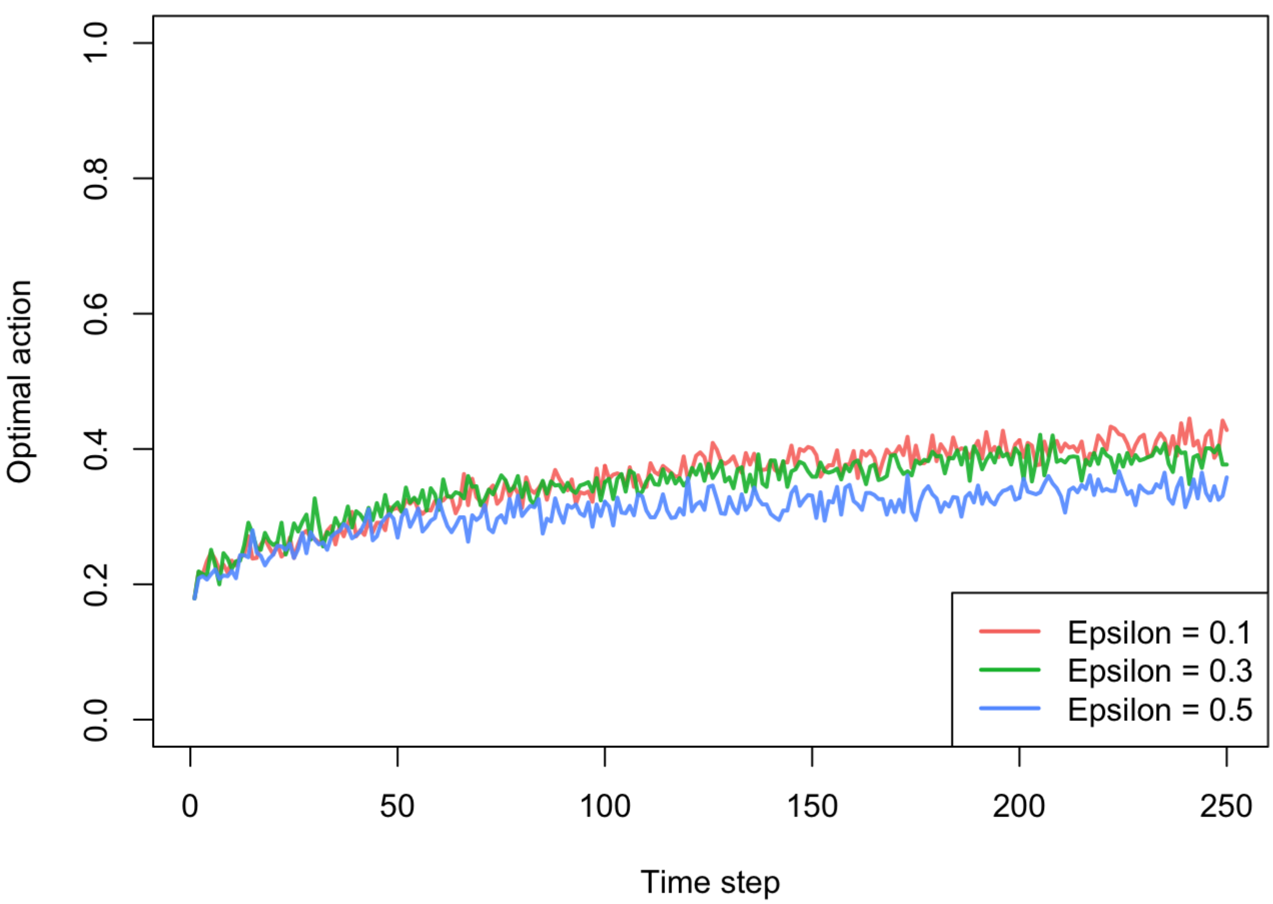
```
## Starting main loop.
```

```
## Finished main loop.
```

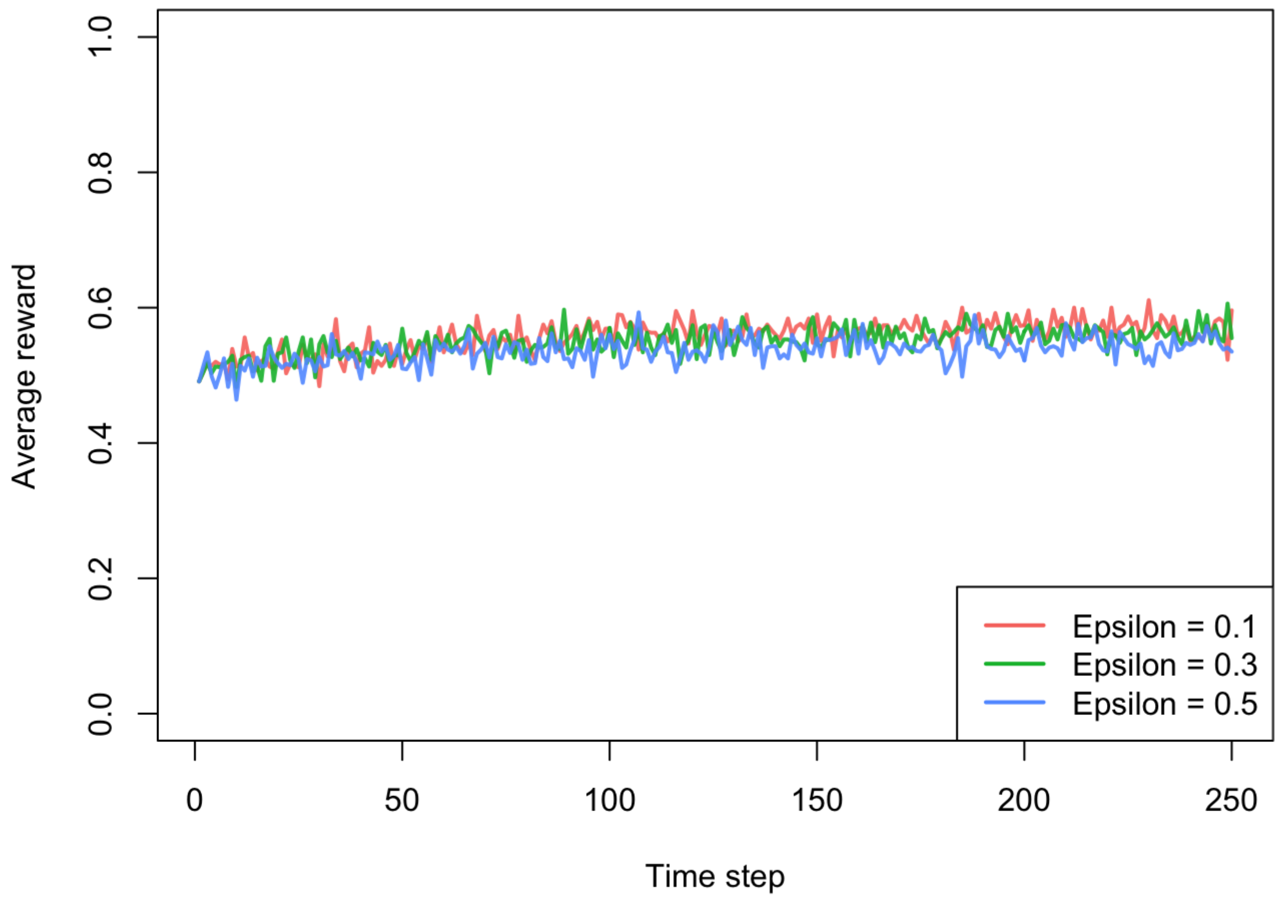
```
## Completed simulation in 0:00:18.473
```

```
## Computing statistics.
```

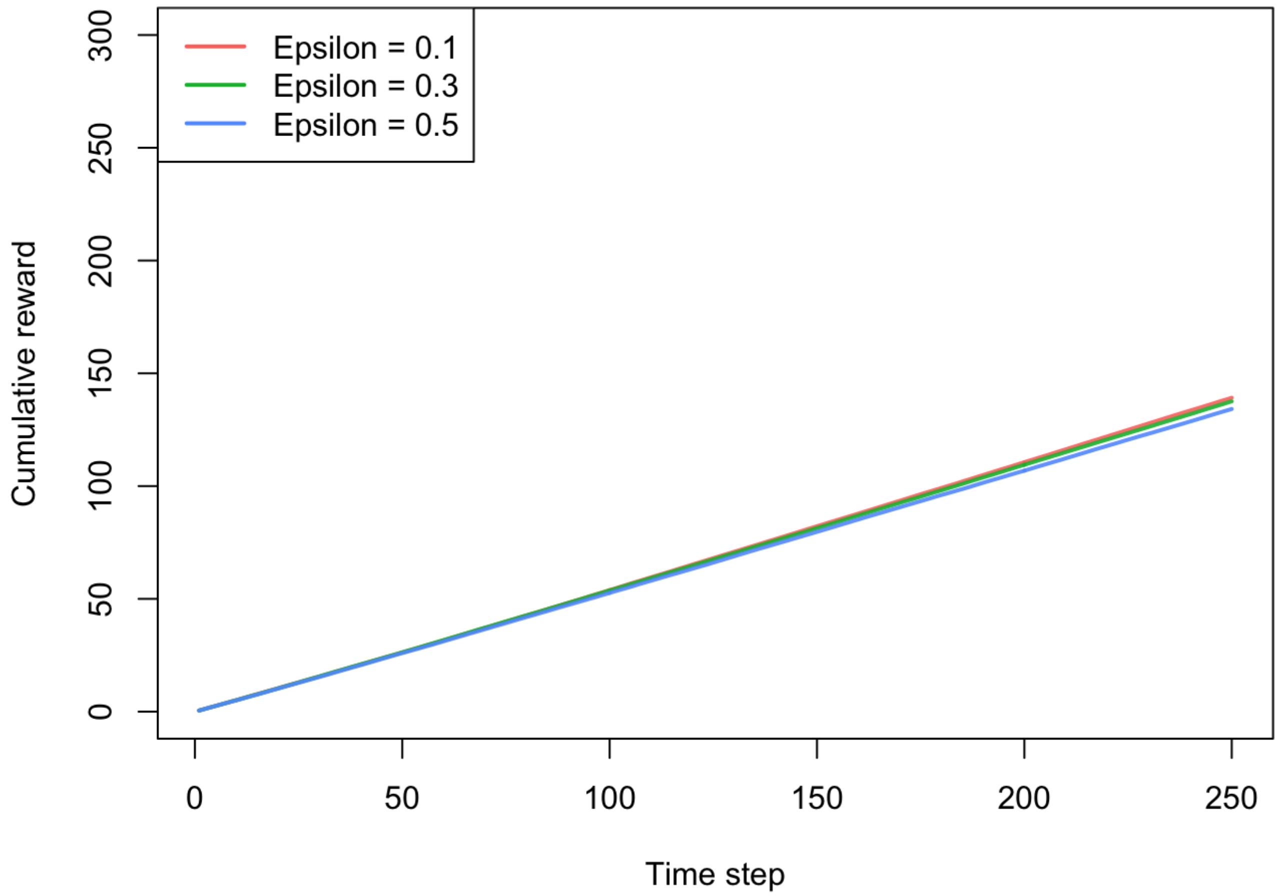
```
# probability of selecting the best design (arm)
# fraction of times the algorithm chooses best design
plot(history,type = 'optimal',legend_position='bottomright', ylim = c(0,1))
```



```
#
# increasing curve means the algorithm is learning
#
# average reward at each visit
plot(history,type = 'average',regret = F, legend_position='bottomright', ylim = c(0,1))
```




```
# fix legend overlay
plot(history,type = 'cumulative',regret =F,ylim = c(0,300))
```




```
#
```

(c)

```
# define 4 options give a reward 10% of time, and one option give a reward 90% of the ti
me
prob_per_arm = c(0.5,0.5,0.5,0.5,0.5)
bandit = BasicBernoulliBandit$new(prob_per_arm)

# define policies (they differ by epsilon value)
agents = list(Agent$new(EpsilonGreedyPolicy$new(0.1),bandit,'Epsilon = 0.1'),
              Agent$new(EpsilonGreedyPolicy$new(0.3),bandit,'Epsilon = 0.3'),
              Agent$new(EpsilonGreedyPolicy$new(0.5),bandit,'Epsilon = 0.5'))
simulation = Simulator$new(agents,visits,simulations)
history = simulation$run()
```

```
## Setting up parallel backend.
```

```
## Cores available: 8
```

```
## Workers assigned: 7
```

```
## Simulation horizon: 250
```

```
## Number of simulations: 3000
```

```
## Number of batches: 7
```

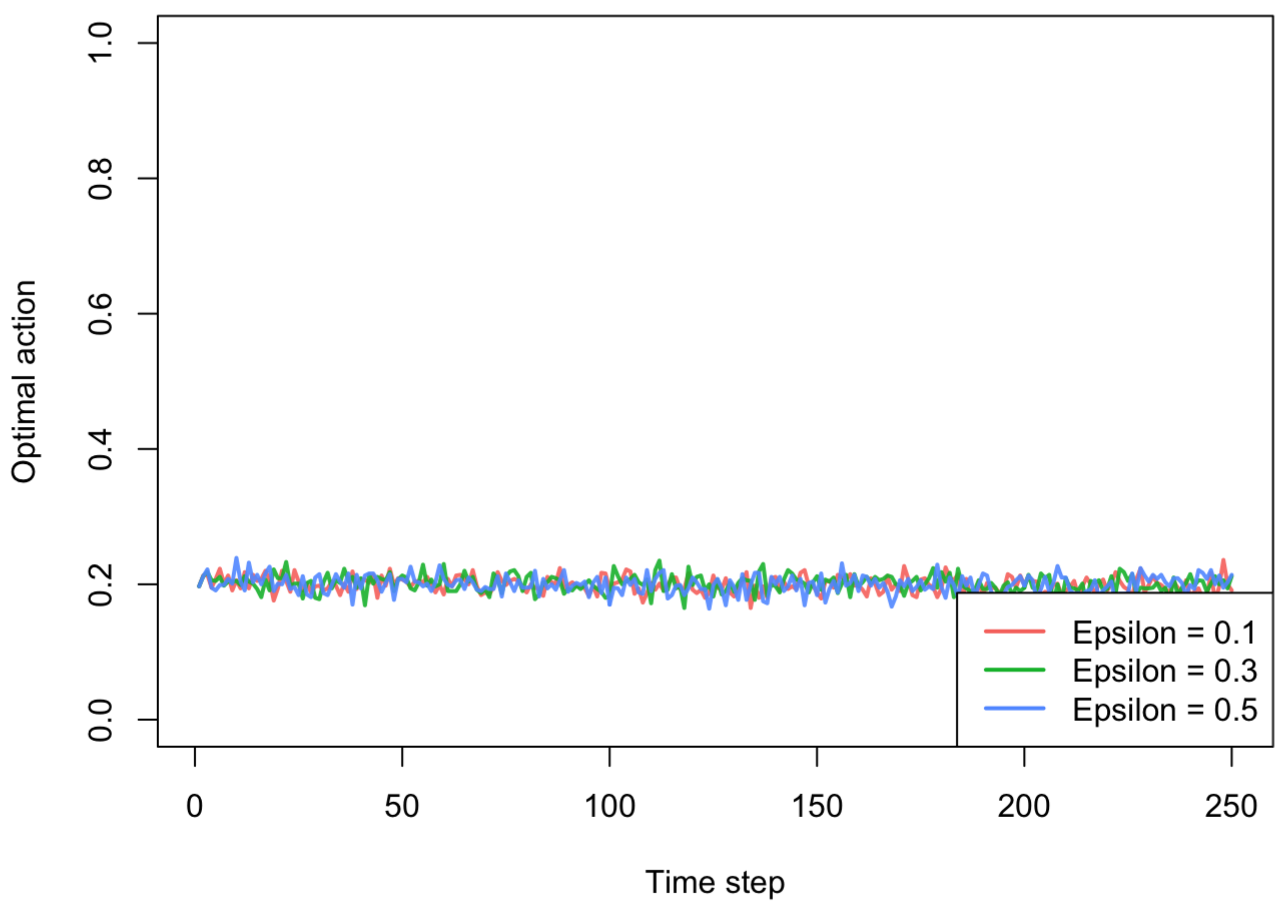
```
## Starting main loop.
```

```
## Finished main loop.
```

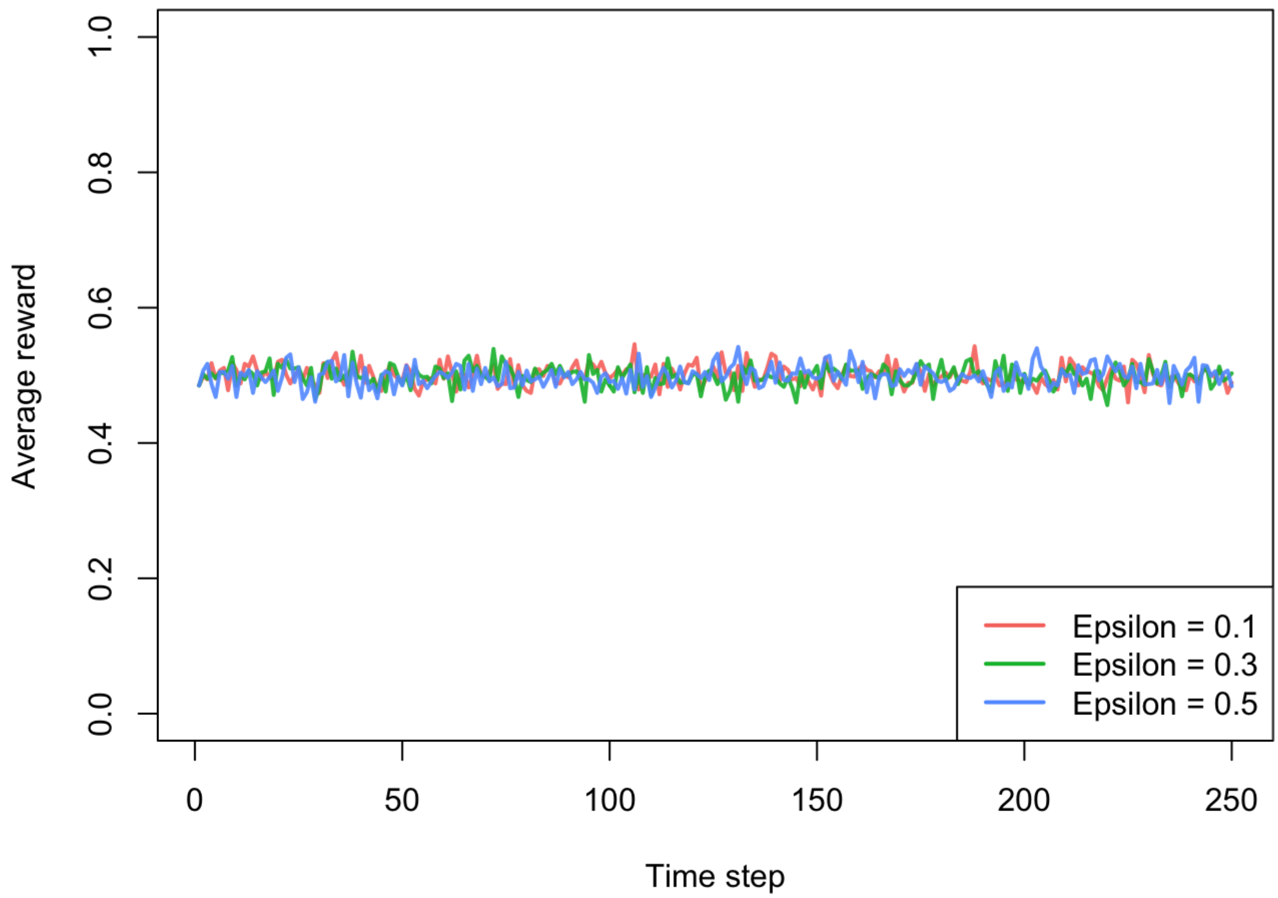
```
## Completed simulation in 0:00:19.526
```

```
## Computing statistics.
```

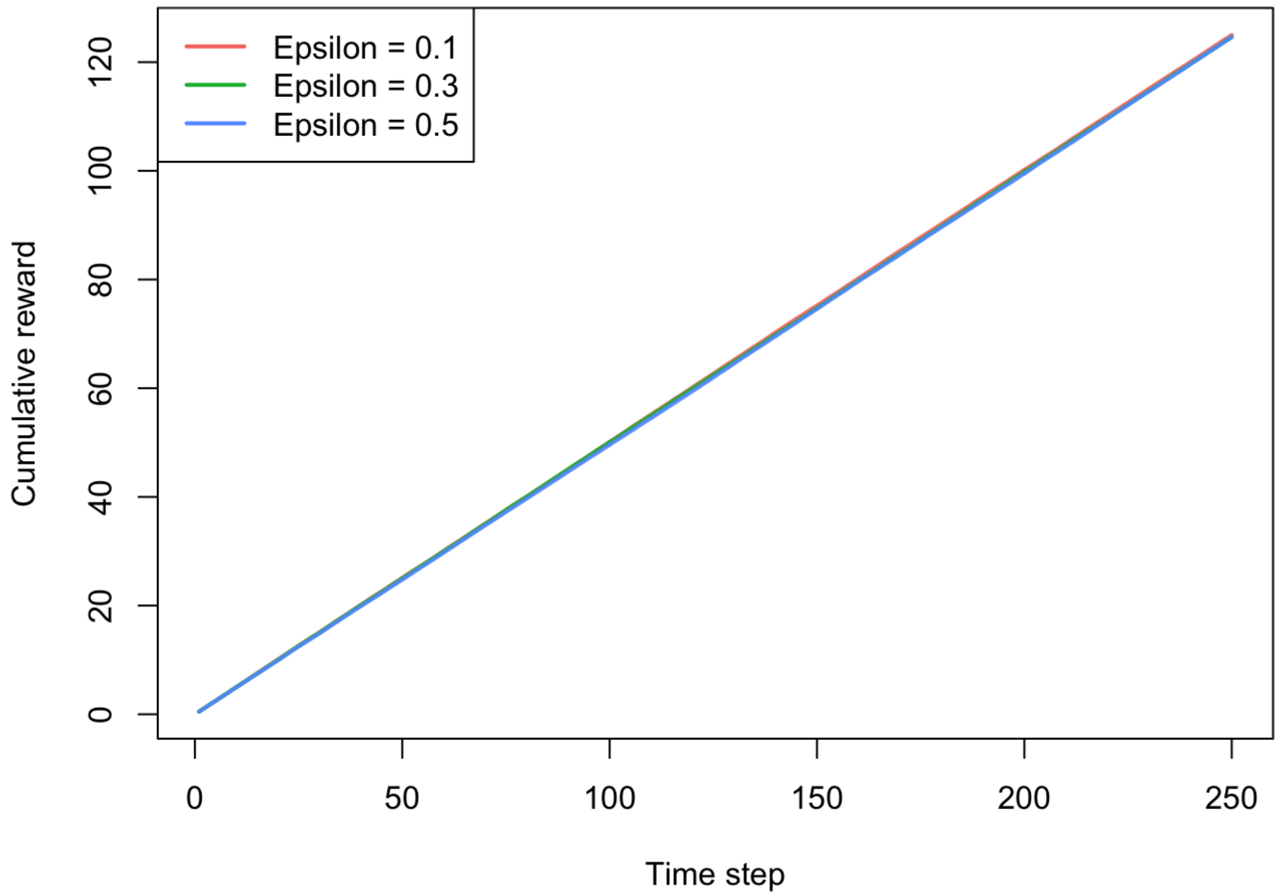
```
# probability of selecting the best design (arm)
# fraction of times the algorithm chooses best design
plot(history,type = 'optimal',legend_position='bottomright', ylim = c(0,1))
```



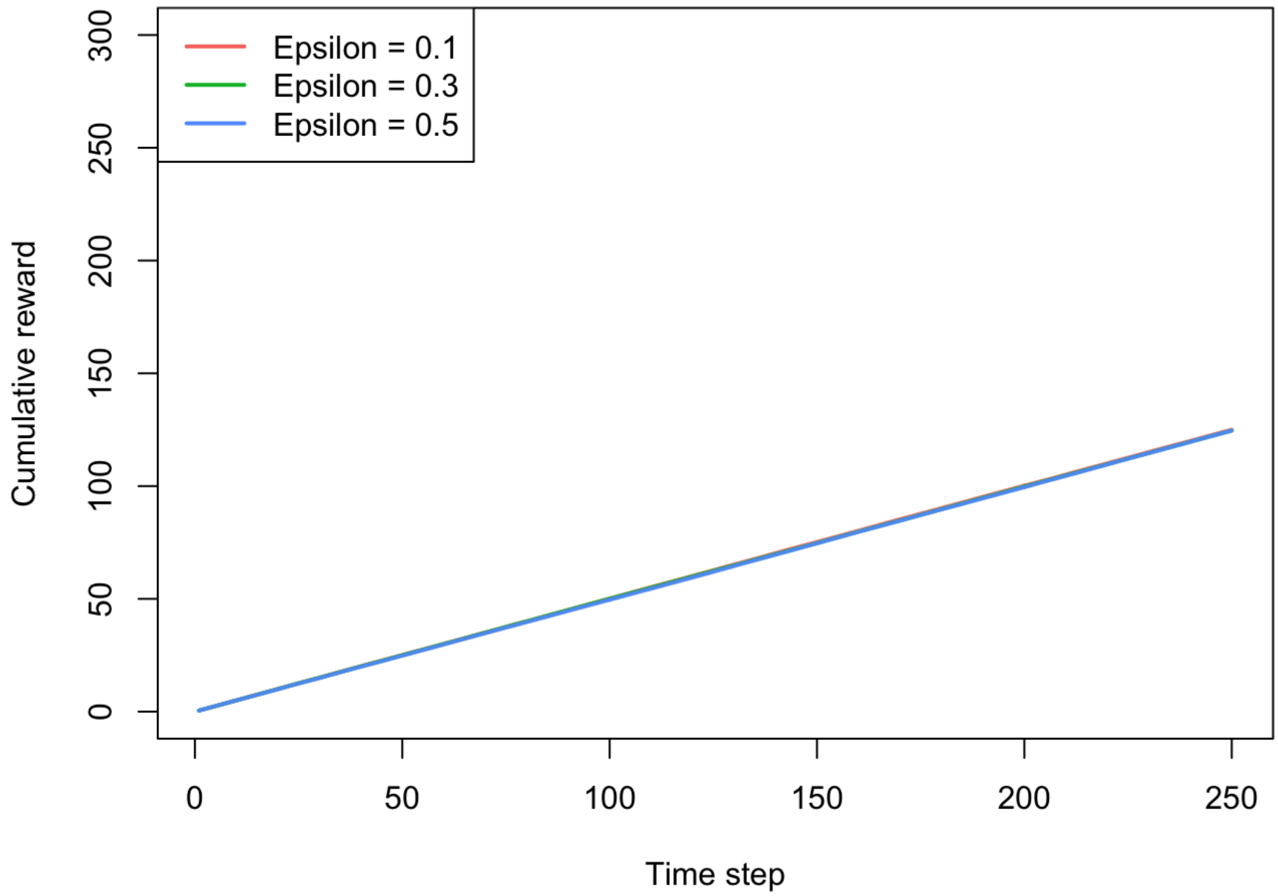
```
#
# increasing curve means the algorithm is learning
#
# average reward at each visit
plot(history,type = 'average',regret = F, legend_position='bottomright', ylim = c(0,1))
```




```
#
# average reward up to visit t
plot(history,type = 'cumulative',regret =F)
```




```r
# fix legend overlay
plot(history,type = 'cumulative',regret =F,ylim = c(0,300))
```




```
#
```

Question 3

```
library(ggplot2)
headlines= c('A','B','C','D','E')
clicks = c(500,1000,825,490,880)
visits = c(900,1800,1500,1100,1325)
d0 = rbind(clicks,visits)
d0 = data.frame(d0)
names(d0) = headlines
d0
```

```
##          A    B    C    D    E
## clicks 500 1000  825  490  880
## visits 900 1800 1500 1100 1325
```

```
#
# observed frequencies table
#
noclicks = visits - clicks
noclicks
```

```
## [1] 400 800 675 610 445
```

```
observed = data.frame(rbind(clicks,noclicks))
names(observed) = headlines
observed
```

```
##            A    B   C   D   E
## clicks   500 1000 825 490 880
## noclicks 400  800 675 610 445
```

```
#
# expected freqs
#
pooled = sum(clicks)/sum(visits)
pooled
```

```
## [1] 0.5577358
```

```
expected_clicks = pooled*visits
expected_noclicks = (1-pooled)*visits
expected = data.frame(rbind(expected_clicks,expected_noclicks))
names(expected) = headlines
expected
```

```
##                          A         B        C        D   E
## expected_clicks   501.9623 1003.9245 836.6038 613.5094 739
## expected_noclicks 398.0377  796.0755 663.3962 486.4906 586
```

```
#
# Chi-square test
#
chisquare = 0
for(i in 1:2)
{
  for(j in 1:5)
  {
    value = ((observed[i,j]-expected[i,j])^2)/expected[i,j]
    chisquare = chisquare + value
  }
}
chisquare
```

```
## [1] 117.466
```

```
# p-value
pvalue = 1 - pchisq(chisquare,2)
pvalue
```

```
## [1] 0
```

```
#
# reject Ho
```

```
# R function
#
prop.test(clicks,visits)
```

```
##
##  5-sample test for equality of proportions without continuity
##  correction
##
## data:  clicks out of visits
## X-squared = 117.47, df = 4, p-value < 2.2e-16
## alternative hypothesis: two.sided
## sample estimates:
##    prop 1    prop 2    prop 3    prop 4    prop 5
## 0.5555556 0.5555556 0.5500000 0.4454545 0.6641509
```

```
#
# conclude Ha: not all headline click-rates are equal
```

```
props = NULL
lls = NULL
uls = NULL

test = binom.test(500,900)
test
```

```
##
##  Exact binomial test
##
## data:  500 and 900
## number of successes = 500, number of trials = 900, p-value =
## 0.0009564
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.5223989 0.5883474
## sample estimates:
## probability of success
##              0.5555556
```

```
props = c(props,test$estimate)
int = test$conf.int
int
```

```
## [1] 0.5223989 0.5883474
## attr(,"conf.level")
## [1] 0.95
```

```
lls = c(lls,int[1])
uls = c(uls,int[2])

test = binom.test(1000,1800)
test
```

```
##
##  Exact binomial test
##
## data:  1000 and 1800
## number of successes = 1000, number of trials = 1800, p-value =
## 2.665e-06
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.5322475 0.5786823
## sample estimates:
## probability of success
##              0.5555556
```

```
props = c(props,test$estimate)
int = test$conf.int
int
```

```
## [1] 0.5322475 0.5786823
## attr(,"conf.level")
## [1] 0.95
```

```
lls = c(lls,int[1])
uls = c(uls,int[2])

test = binom.test(825,1500)
test
```

```
##
##  Exact binomial test
##
## data:  825 and 1500
## number of successes = 825, number of trials = 1500, p-value =
## 0.0001181
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.5244132 0.5753906
## sample estimates:
## probability of success
##                   0.55
```

```
props = c(props,test$estimate)
int = test$conf.int
int
```

```
## [1] 0.5244132 0.5753906
## attr(,"conf.level")
## [1] 0.95
```

```
lls = c(lls,int[1])
uls = c(uls,int[2])

test = binom.test(490,1100)
test
```

```
##
##  Exact binomial test
##
## data:  490 and 1100
## number of successes = 490, number of trials = 1100, p-value =
## 0.0003291
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.4158083 0.4753933
## sample estimates:
## probability of success
##              0.4454545
```

```
props = c(props,test$estimate)
int = test$conf.int
int
```

```
## [1] 0.4158083 0.4753933
## attr(,"conf.level")
## [1] 0.95
```

```
lls = c(lls,int[1])
uls = c(uls,int[2])

test = binom.test(880,1325)
test
```

```
##
##  Exact binomial test
##
## data:  880 and 1325
## number of successes = 880, number of trials = 1325, p-value <
## 2.2e-16
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.6380010 0.6895705
## sample estimates:
## probability of success
##              0.6641509
```

```
props = c(props,test$estimate)
int = test$conf.int
int
```

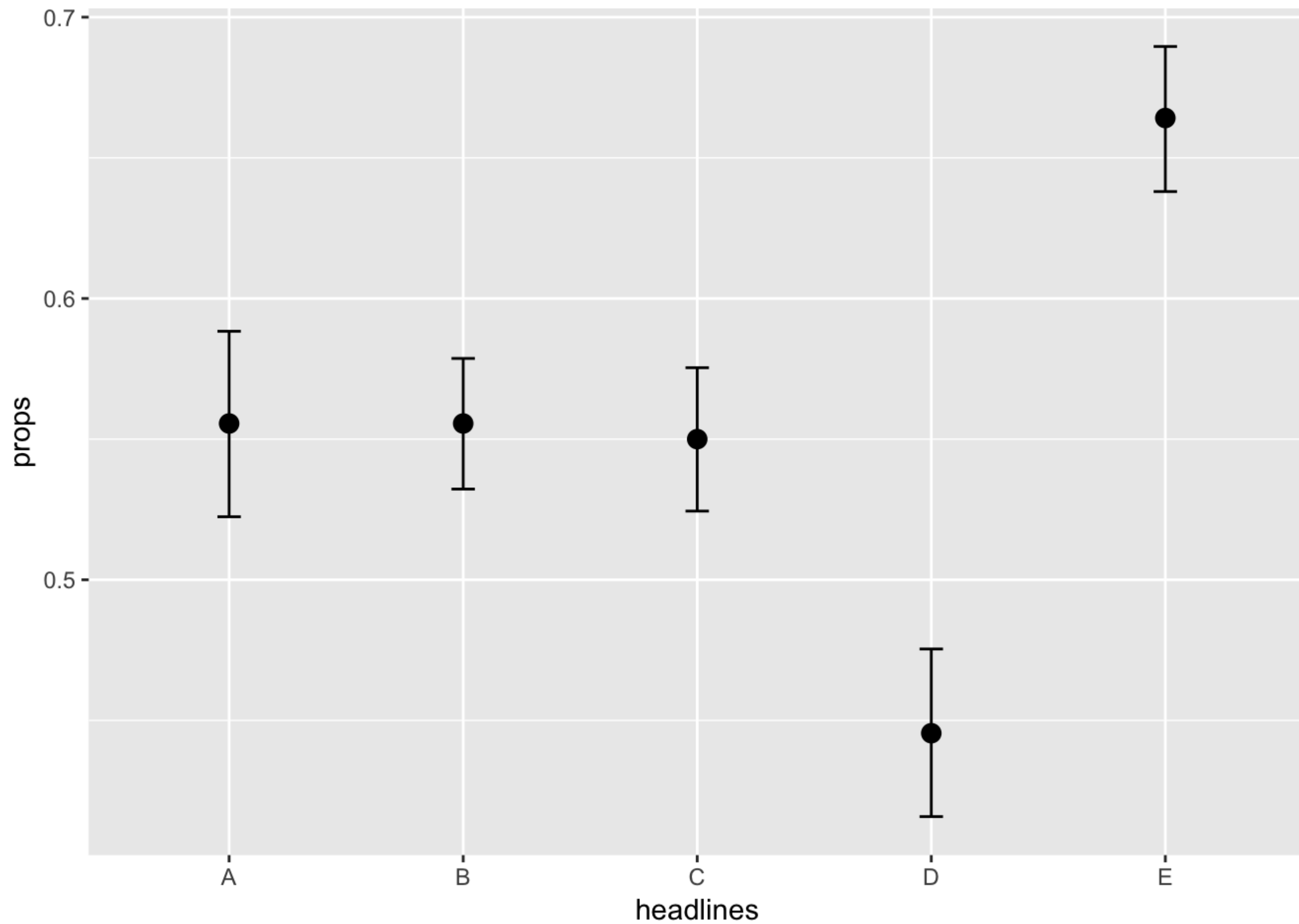
```
## [1] 0.6380010 0.6895705
## attr(,"conf.level")
## [1] 0.95
```

```
lls = c(lls,int[1])
uls = c(uls,int[2])

d = data.frame(headlines,props,lls,uls)
d
```

```
##   headlines     props       lls       uls
## 1         A 0.5555556 0.5223989 0.5883474
## 2         B 0.5555556 0.5322475 0.5786823
## 3         C 0.5500000 0.5244132 0.5753906
## 4         D 0.4454545 0.4158083 0.4753933
## 5         E 0.6641509 0.6380010 0.6895705
```

```
# plot
ggplot(data=d) +
  geom_errorbar(mapping = aes(x=headlines,ymin=lls,ymax=uls),width=0.1) +
  geom_point(mapping = aes(x=headlines,y=props),size = 3)
```




```
test = function(x,n){binom.test(x,n)}
out = mapply(test,clicks,visits)
out
```

```
##             [,1]                   [,2]
## statistic   500                    1000
## parameter   900                    1800
## p.value     0.0009564441           2.665343e-06
## conf.int    Numeric,2              Numeric,2
## estimate    0.5555556              0.5555556
## null.value  0.5                    0.5
## alternative "two.sided"            "two.sided"
## method      "Exact binomial test" "Exact binomial test"
## data.name   "x and n"              "x and n"
##             [,3]                   [,4]
## statistic   825                    490
## parameter   1500                   1100
## p.value     0.0001180582           0.0003291259
## conf.int    Numeric,2              Numeric,2
## estimate    0.55                   0.4454545
## null.value  0.5                    0.5
## alternative "two.sided"            "two.sided"
## method      "Exact binomial test" "Exact binomial test"
## data.name   "x and n"              "x and n"
##             [,5]
## statistic   880
## parameter   1325
## p.value     2.381536e-33
## conf.int    Numeric,2
## estimate    0.6641509
## null.value  0.5
## alternative "two.sided"
## method      "Exact binomial test"
## data.name   "x and n"
```

```
class(out)
```

```
## [1] "matrix"
```

```
class(out[1,1])
```

```
## [1] "list"
```

```
# using mapply this way is a matrix of lists
#
out[4,1]
```

```
## $conf.int
## [1] 0.5223989 0.5883474
## attr(,"conf.level")
## [1] 0.95
```

```r
out[4,1]$conf.int[1]
```

```
## [1] 0.5223989
```

```r
out[4,1]$conf.int[2]
```

```
## [1] 0.5883474
```

```r
#
# collect all CIs lower limits
for(i in 1:5) lls[i] = out[4,i]$conf.int[1]
# collect all CIs upper limits
for(i in 1:5) uls[i] = out[4,i]$conf.int[2]
#
d = data.frame(headlines,props,lls,uls)
d
```

```
##   headlines     props       lls       uls
## 1         A 0.5555556 0.5223989 0.5883474
## 2         B 0.5555556 0.5322475 0.5786823
## 3         C 0.5500000 0.5244132 0.5753906
## 4         D 0.4454545 0.4158083 0.4753933
## 5         E 0.6641509 0.6380010 0.6895705
```

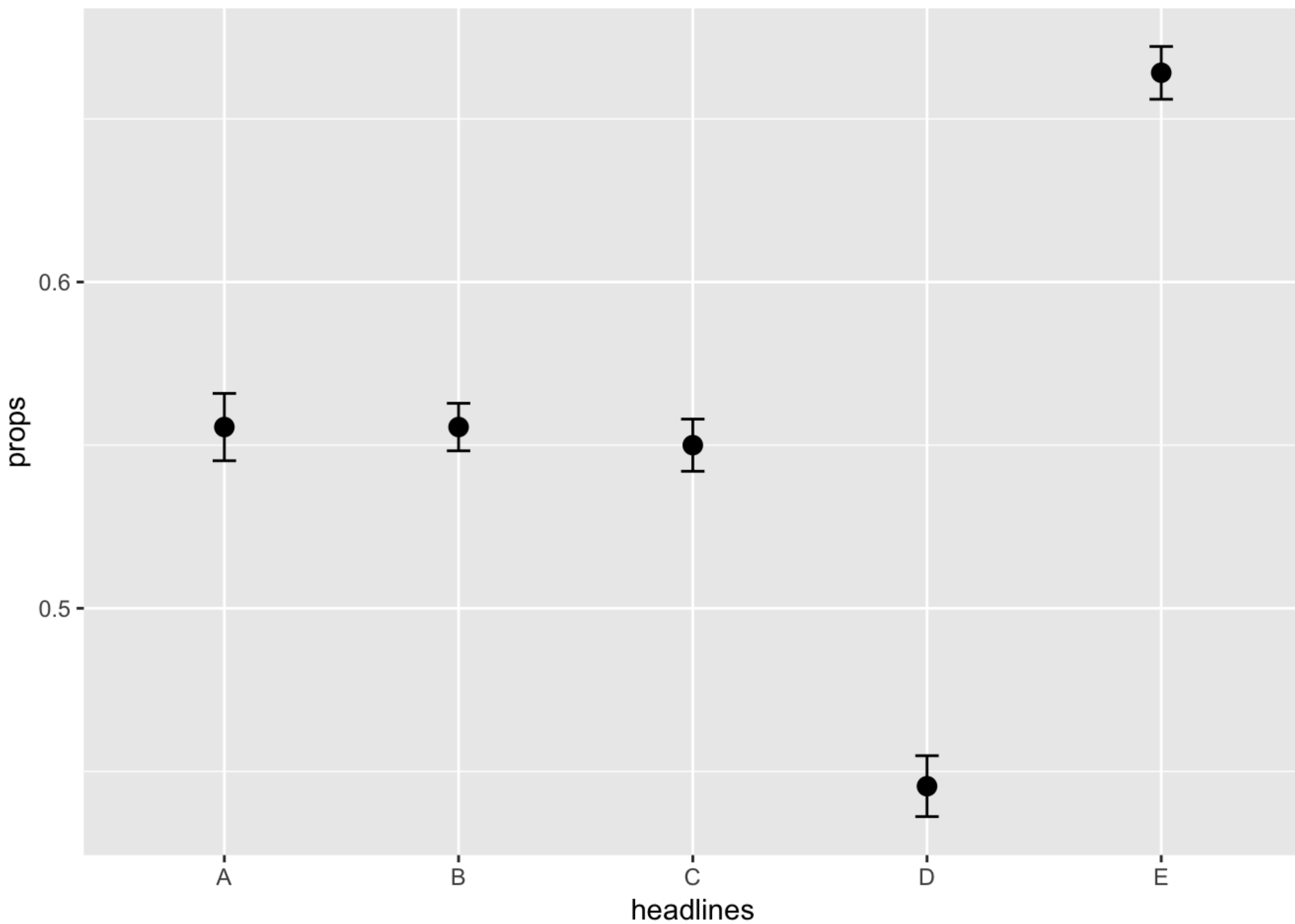
```r
#
# Suppose that we obtained larger samples
#
clicks = 10*clicks
visits = 10*visits
d0 = rbind(clicks,visits)
d0 = data.frame(d0)
names(d0) = headlines
d0
```

```
##           A     B     C     D     E
## clicks 5000 10000  8250  4900  8800
## visits 9000 18000 15000 11000 13250
```

```r
out = mapply(test,clicks,visits)
# collect all CIs limits
for(i in 1:5) lls[i] = out[4,i]$conf.int[1]
for(i in 1:5) uls[i] = out[4,i]$conf.int[2]
#
d3 = data.frame(headlines,props,lls,uls)
d3
```

```
##   headlines     props       lls       uls
## 1         A 0.5555556 0.5452174 0.5658577
## 2         B 0.5555556 0.5482602 0.5628330
## 3         C 0.5500000 0.5419962 0.5579844
## 4         D 0.4454545 0.4361366 0.4548014
## 5         E 0.6641509 0.6560362 0.6721936
```

```
# plot
ggplot(data=d3) +
  geom_errorbar(mapping = aes(x=headlines,ymin=lls,ymax=uls),width=0.1) +
  geom_point(mapping = aes(x=headlines,y=props),size = 3)
```



Question 4

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
d4 = read.csv('cereal.csv')
str(d4)
```

```
## 'data.frame':    1250 obs. of  3 variables:
##  $ Group    : int  1 2 2 4 2 2 2 3 3 4 ...
##  $ Spend    : num  14.77 8.15 8 9.31 12.09 ...
##  $ Breakfast: int  4 4 2 1 4 4 4 4 3 3 ...
```

```
n = nrow(d4)
table(d4$Group)
```

```
##
##   1   2   3   4
## 269 484 241 256
```

```
head(d4)
```

```
##   Group Spend Breakfast
## 1     1 14.77         4
## 2     2  8.15         4
## 3     2  8.00         2
## 4     4  9.31         1
## 5     2 12.09         4
## 6     2  7.13         4
```

```
d4 %>%
  group_by(d4$Group) %>%
  summarise(sum_group = sum(Spend))
```

```
## # A tibble: 4 x 2
##   `d4$Group` sum_group
##        <int>     <dbl>
## 1          1     3115.
## 2          2     4911.
## 3          3     3910.
## 4          4     1574.
```

```
tapply(d4$Spend, d4$Group, mean)
```

```
##         1         2         3         4
## 11.579257 10.145888 16.222780  6.147656
```

The total and the mean of the amount spent by these four segments is different

```
g1 <- d4$Group==1
group1 = d4[g1,]

g2 <- d4$Group==2
group2 = d4[g2,]

g3 <- d4$Group==3
group3 = d4[g3,]

g4 <- d4$Group==4
group4 = d4[g4,]
```

```
props = NULL
lls = NULL
uls = NULL

test = t.test(group1$Spend)
test
```

```
##
##  One Sample t-test
##
## data:  group1$Spend
## t = 62.351, df = 268, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  11.21362 11.94489
## sample estimates:
## mean of x
##  11.57926
```

```
props = c(props,test$estimate)
int = test$conf.int
int
```

```
## [1] 11.21362 11.94489
## attr(,"conf.level")
## [1] 0.95
```

```
lls = c(lls,int[1])
uls = c(uls,int[2])

test = t.test(group2$Spend)
test
```

```
##
##  One Sample t-test
##
## data:  group2$Spend
## t = 69.347, df = 483, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##   9.858414 10.433363
## sample estimates:
## mean of x
##  10.14589
```

```
props = c(props,test$estimate)
int = test$conf.int
int
```

```
## [1]  9.858414 10.433363
## attr(,"conf.level")
## [1] 0.95
```

```
lls = c(lls,int[1])
uls = c(uls,int[2])

test = t.test(group3$Spend)
test
```

```
##
##  One Sample t-test
##
## data:  group3$Spend
## t = 81.03, df = 240, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  15.82839 16.61717
## sample estimates:
## mean of x
##  16.22278
```

```
props = c(props,test$estimate)
int = test$conf.int
int
```

```
## [1] 15.82839 16.61717
## attr(,"conf.level")
## [1] 0.95
```

```
lls = c(lls,int[1])
uls = c(uls,int[2])

test = t.test(group4$Spend)
test
```

```
##
##  One Sample t-test
##
## data:  group4$Spend
## t = 53.462, df = 255, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  5.921201 6.374111
## sample estimates:
## mean of x
##  6.147656
```

```
props = c(props,test$estimate)
int = test$conf.int
int
```

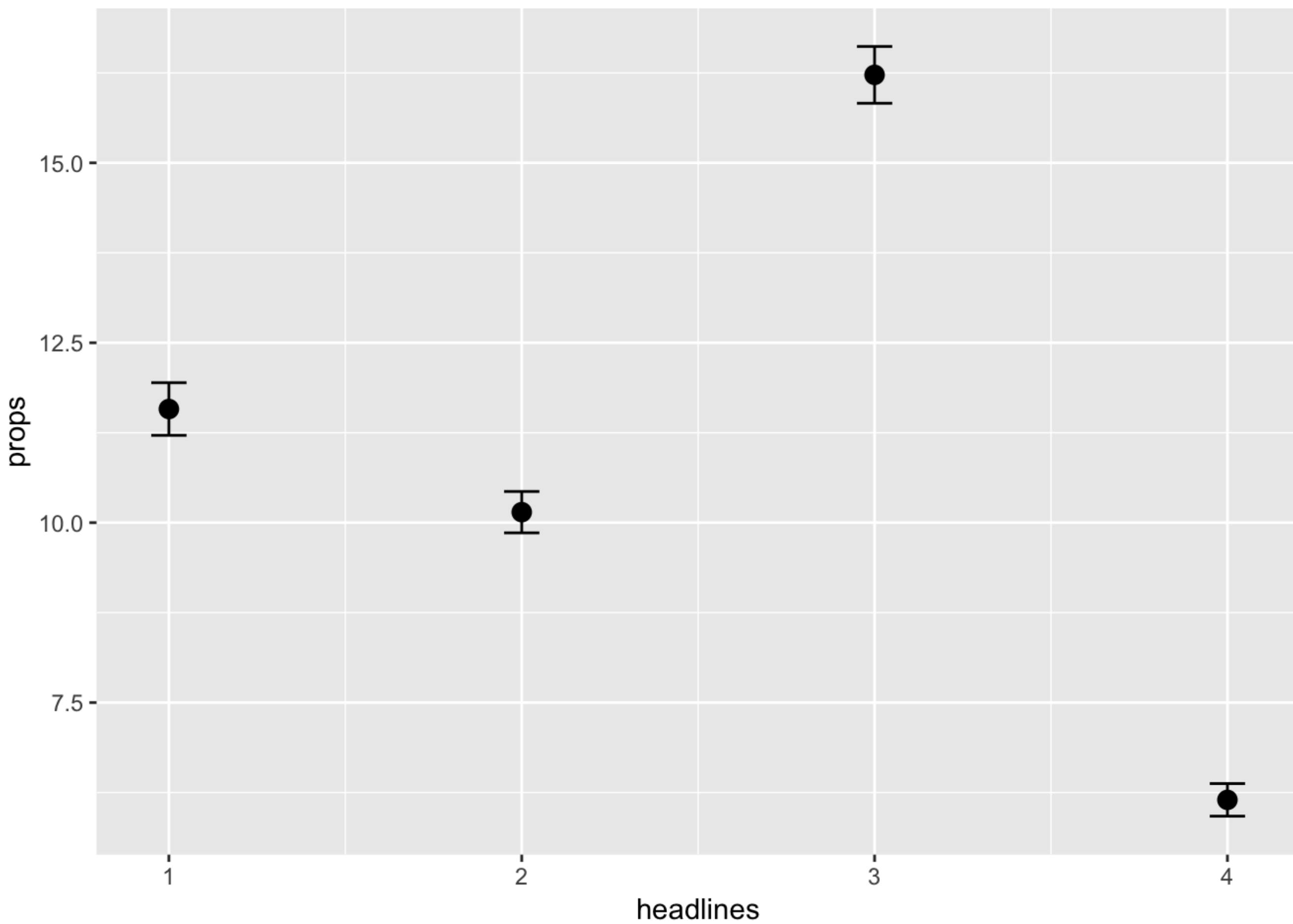
```
## [1] 5.921201 6.374111
## attr(,"conf.level")
## [1] 0.95
```

```
lls = c(lls,int[1])
uls = c(uls,int[2])
```

```
headlines = c(1,2,3,4)
d4_1 = data.frame(headlines,props,lls,uls)
d4_1
```

```
##   headlines     props       lls       uls
## 1         1 11.579257 11.213620 11.944893
## 2         2 10.145888  9.858414 10.433363
## 3         3 16.222780 15.828390 16.617170
## 4         4  6.147656  5.921201  6.374111
```

```
ggplot(data=d4_1) +
  geom_errorbar(mapping = aes(x=headlines,ymin=lls,ymax=uls),width=0.1) +
  geom_point(mapping = aes(x=headlines,y=props),size = 3)
```



The gragh above showing that people spent on segment 3 the most.