benchmark of linear solver

Yinan Zhu

September 28, 2017

```
library(microbenchmark)
```

```
## Warning: package 'microbenchmark' was built under R version 3.4.1
```

```
source('~/GitHub/SDS385-course-work/Excercise 1/linear equation solver/linear solver.R')
```

First, consider two solvers, of which "linearsolver" use inverse matrix and "linearsolverlu" use LU decomposition, we should check that they give the correct result, using data generated from a linear model with gaussian noise

```
n=3000
p=20
beta=matrix(1,nrow=p,ncol=1)
X=matrix(runif(n*p,0,1),nrow=n,ncol=p)
y=X%*%beta+rnorm(n,sd=1)
```

```
print(linearsolve(y,X))
```

```
##            [,1]
##  [1,] 1.0487508
##  [2,] 1.0272857
##  [3,] 1.0859725
##  [4,] 0.9591240
##  [5,] 1.0403490
##  [6,] 0.8836230
##  [7,] 0.9327977
##  [8,] 1.0646996
##  [9,] 0.9320475
## [10,] 0.9978795
## [11,] 0.9237362
## [12,] 0.9052198
## [13,] 1.0749832
## [14,] 0.8886502
## [15,] 0.9271126
## [16,] 1.0071691
## [17,] 1.1728008
## [18,] 1.0567865
## [19,] 1.1014499
## [20,] 0.9623982
```

```
print(linearsolvelu(y,X))
```

```
## 20 x 1 Matrix of class "dgeMatrix"
##            [,1]
##  [1,] 1.0487508
##  [2,] 1.0272857
##  [3,] 1.0859725
##  [4,] 0.9591240
##  [5,] 1.0403490
##  [6,] 0.8836230
```

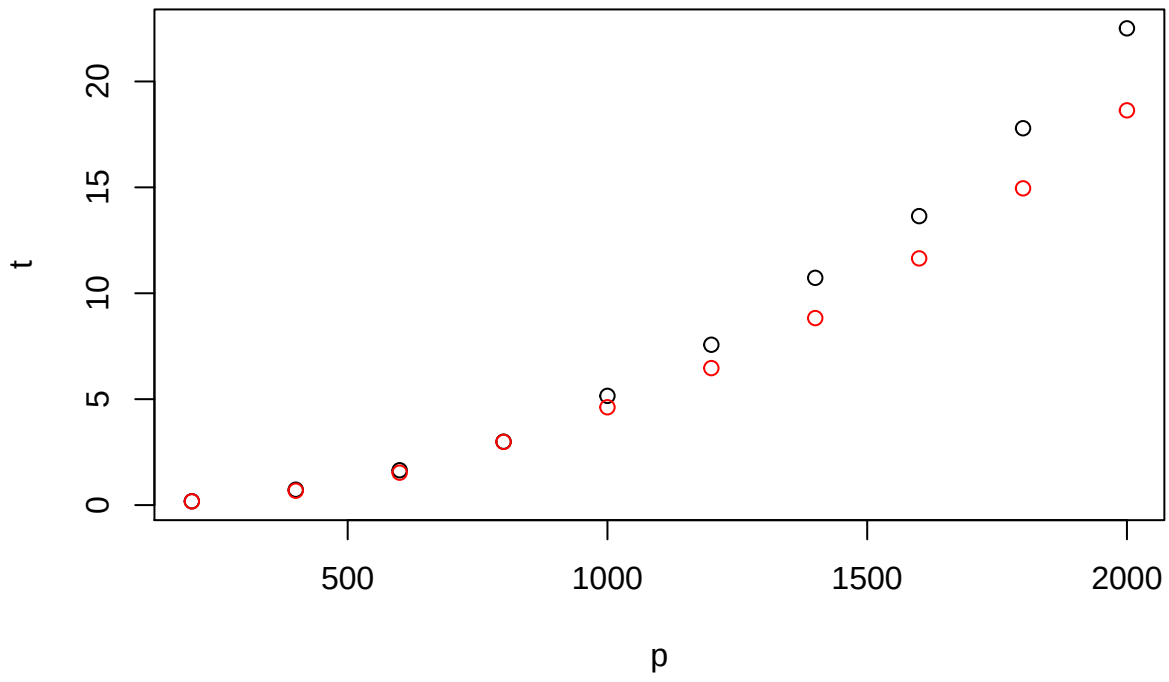
```
##  [7,] 0.9327977
##  [8,] 1.0646996
##  [9,] 0.9320475
## [10,] 0.9978795
## [11,] 0.9237362
## [12,] 0.9052198
## [13,] 1.0749832
## [14,] 0.8886502
## [15,] 0.9271126
## [16,] 1.0071691
## [17,] 1.1728008
## [18,] 1.0567865
## [19,] 1.1014499
## [20,] 0.9623982
```

Now we fix n, the number of points to be constant, and varies p, the number of features. Test the time it takes to solve

```
t=as.vector(rep(0,10))
tlu=t
for(i in 1:10){
p=200*i
beta=matrix(1,nrow=p,ncol=1)
X=matrix(runif(n*p,0,1),nrow=n,ncol=p)
y=X%*%beta+rnorm(n,sd=1)

l=microbenchmark(linearsolve(y,X),times=1L)
t[i]=l$time/1000000000
l=microbenchmark(linearsolvelu(y,X),times=1L)
tlu[i]=l$time/1000000000
}
```

```
plot(t~seq(from=200,to=2000,by=200),xlab='p')
points(tlu~seq(from=200,to=2000,by=200),col='red')
```



Now we add "linearsolvesparse" which use the sparse matrix class

```
p=20
mask=matrix(rbinom(n*p,1,0.05),nrow=n,ncol=p)
X=matrix(runif(n*p,0,1),nrow=n,ncol=p)
beta=matrix(1,nrow=p,ncol=1)
Xsparse=X*mask
y=Xsparse%*%beta+rnorm(n,sd=10)
```

```
print(linearsolve(y,Xsparse))
```

```
##             [,1]
##  [1,]  1.4806215
##  [2,]  0.9538296
##  [3,]  0.9434488
##  [4,] -0.4156347
##  [5,]  2.6230407
##  [6,]  0.4524291
##  [7,]  1.7185126
##  [8,] -1.2125243
##  [9,] -1.6212439
## [10,] -0.4231141
## [11,]  1.9036825
## [12,]  1.5183774
## [13,]  0.8445240
## [14,]  0.3829095
## [15,]  0.7728963
```

```
## [16,]  3.7913797
## [17,]  1.4177467
## [18,]  2.5337501
## [19,]  1.2001069
## [20,]  1.6588161
```

```
print(linearsolvelu(y,Xsparse))
```

```
## 20 x 1 Matrix of class "dgeMatrix"
##             [,1]
##  [1,]  1.4806215
##  [2,]  0.9538296
##  [3,]  0.9434488
##  [4,] -0.4156347
##  [5,]  2.6230407
##  [6,]  0.4524291
##  [7,]  1.7185126
##  [8,] -1.2125243
##  [9,] -1.6212439
## [10,] -0.4231141
## [11,]  1.9036825
## [12,]  1.5183774
## [13,]  0.8445240
## [14,]  0.3829095
## [15,]  0.7728963
## [16,]  3.7913797
## [17,]  1.4177467
## [18,]  2.5337501
## [19,]  1.2001069
## [20,]  1.6588161
```

```
print(linearsolvesparse(y,Xsparse))
```

```
##  [1]  1.4806215  0.9538296  0.9434488 -0.4156347  2.6230407  0.4524291
##  [7]  1.7185126 -1.2125243 -1.6212439 -0.4231141  1.9036825  1.5183774
## [13]  0.8445240  0.3829095  0.7728963  3.7913797  1.4177467  2.5337501
## [19]  1.2001069  1.6588161
```

To benchmark the performance, we keep the size of equation constant and varies the density of observed data

```
t=as.vector(rep(0,6))
tlu=t
ts=t
p=1600
X=matrix(runif(n*p,0,1),nrow=n,ncol=p)
beta=matrix(1,nrow=p,ncol=1)
for(i in 1:6){
  mask=matrix(rbinom(n*p,1,0.1*i),nrow=n,ncol=p)
  Xsparse=X*mask
  y=Xsparse%*%beta+rnorm(n,sd=10)
  l=microbenchmark(linearsolve(y,Xsparse),times=1L)
  t[i]=l$time/1000000000
  l=microbenchmark(linearsolvelu(y,Xsparse),times=1L)
  tlu[i]=l$time/1000000000
  l=microbenchmark(linearsolvesparse(y,Xsparse),times=1L)
  ts[i]=l$time/1000000000
```

```
}
```

```
plot(t~seq(from=0.1,to=0.6,by=0.1),ylim=c(0,15),xlab='density')
points(tlu~seq(from=0.1,to=0.6,by=0.1),col='red')
points(ts~seq(from=0.1,to=0.6,by=0.1),col='blue')
```