STAT243 Problem set 7

Tianyi Zhang

November 14, 2015

Problem 1

-What are the goals of their simulation study and what are the metrics that they consider in assessing their method?

The goal of their simulation study is to assess the accuracy of the asymptotic approximation in finite samples and to examine the power of the EM-test. The metrics they consider to use are looking at the power of EM-test (type II error) and the significance level (type I error) to assess the effectiveness of their method.

-What choices did the authors have to make in designing their simulation study? What are the key aspects of the data generating mechanism that likely aspect the statistical power of the test?

The choices that authors did included the choosing of sample sizes. The authors chose two sample sizes, which were 200 and 400. Also 5 percent and 1 percent significance level were chosen, and the number of repetitions were also included in the design of the simulation study.
The key aspects of data generating mechanism that impact the statistical power of the test includes the sample size of the data, and the randomness of the generation process. As the sample size increases, the power will tend to increase.
-Suggest some alternatives to how the authors designed their study. Are there data-generating scenarios that they did not consider that would be useful to consider?

A possible alternative would be, instead of setting Ha: m>2, we can set H0: m=2 vs Ha: m=3.
- Give some thoughts on how to set up a simulation study for their problem that uses principles of basic experimental design (see the Unit 10 notes) or if you think it would be difficult, say why.

For the set up of the experimental design, when we choose the inputs, such as parameters, sample sizes, and data generating mechanisms, we want to categorize them into small number of levels, so that we do not have too many level of treatments. Besides, another thing need to mention is to focus on the decomposition of sum squares.

-Do their figures/tables do a good job of presenting the simulation results and do you have any alternative suggestions for how to do this? Do the authors address the issue of simulation uncertainty/simulation standard errors and/or do they convince the reader they've done enough simulation replications?

Regardless of the functionality that generally presenting the study results, there are still aspects that this paper can improve in its tabulation and figures. First of all, the authors did not provide the detailed data of type I error results in the text of section 4, thus the box-plot should at least show the means and standard deviations of type I errors, perhaps histograms would be another good choice.
Additionally, the author did not provide sufficient information about the simulation uncertainty and standard errors because there are 12 models, and the authors only provided one type I error for each model with many replications. More reasonably, I think that the authors should box-plot the standard errors and the

means for each of 12 models instead of comparing 12 models.

-Interpret their tables on power (Tables 4 and 6) - do the results make sense in terms of how the power varies as a function of the data generating mechanism?

The results on power in Tables 4 and 6 are consistent with the data generating mechanism. First of all, the powers have been increasing significantly compared from sample size 200 to 400.
Secondly, as the elements within weight vectors and standard deviation vectors spread more, the power will be larger. This is consistent to the fact stated before that with the same weight vectors and standard deviation vectors, rejecting null hypothesis becomes difficult. Also, when order goes up, rejecting the null hypothesis becomes difficult.
-Discuss the extent to which they follow JASA's guidelines on simulation studies (see the end of the Unit 10 class notes for the JASA guidelines).
The most serious problem of this simulation study is about the reproducibility. Since the authors failed to provide descriptions of pseudo-code and the numerical algorithms of the simulation, so that other researchers are not able to replicate the experiment to do further research.

Problem 2

2A

Referring the the lecture notes in Unit7, the steps calculating each step are listed below:
Step 1: $U_{11} = \sqrt{A_{11}}$, 0 step.
Step 2: $j = 2...n, U_{1j} = A_{1j}/U_{11}$, $n-1$ steps of divison.
Step 3:
Part 1:$i = 2...n, U_{ii} = \sqrt{A_{ii} - \Sigma_{k=1}^{i-1} U_{ki}^2}$, $1 + ... + n - 1 = \frac{n(n-1)}{2} steps.$
Part 2: $j = i+1, ..., n. U_{ij} = \frac{A_{ij} - \Sigma_{k=1}^{i-1} U_{ki}U_{kj}}{U_{ii}}$, for each ij combination, ther are $i - 1 + 1 = i$ steps. For each i, there are $n - i$ number of js. Thus There are $\Sigma_{i=2}^{n} i(n-i) = \frac{1}{6}n(n+1)(n-1) - (n-1)$ steps.
Therefore, there are
$Total = \frac{1}{6}n(n+1)(n-1) + \frac{n(n-1)}{2} = \frac{n^3}{6} + \frac{n^2}{2} - \frac{2n}{3}$.

2B

In fact, we **can** "overwrite" the orginal matrix by the new upper triangular matrix U as time goes along, which means we do not keep another copy of the original in the storage.
The reason is straight forward. For any computation to get the U_{ij}, I do not need any information of the entry A_{mk} where $m < i \; and \; k < j$.
For instance, by looking at the step 2 and 3 in the lecture notes, if we needs to compute the entry U_{23} in the new upper triangular matrix, I only need like A_{23}, A_{24}..., but not A_{11}, A_{12}.... From this perspective, all entires sit left above the current entry could be already subsitituted to the entry of the new matrix, which saves memory.

2C

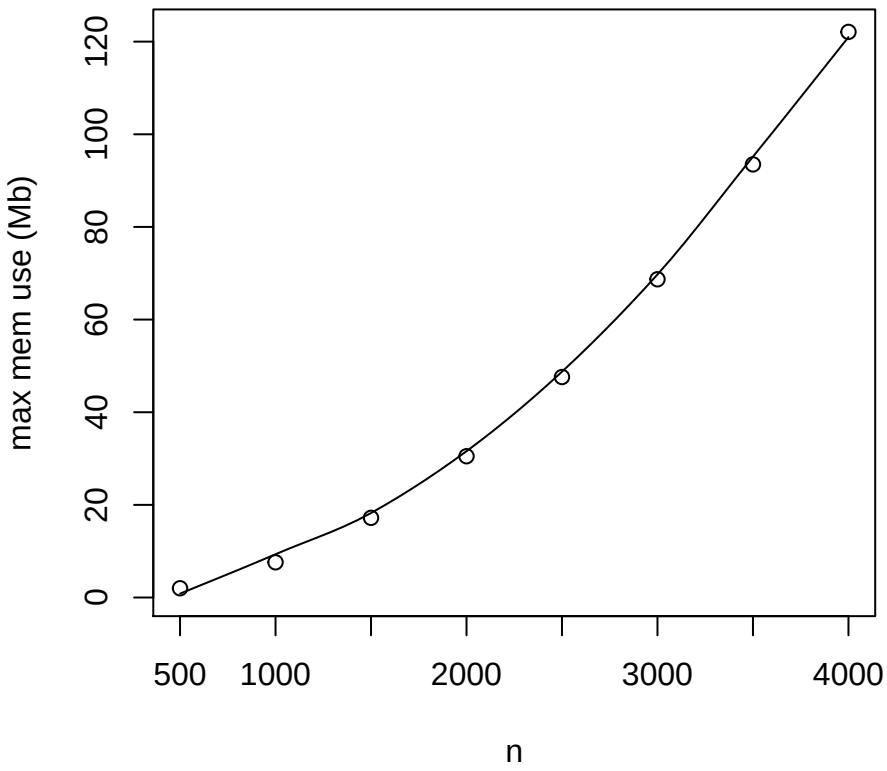
In this problem, we need first to use the implications in 5A, which states that for every X with $n * p$,XX^T is a positive definite matrix.
Then I find another much faster way to get a positive definite matrix using **diagnolally dominate matrix** which states that a symmetric diagonally dominant matrix is positive semi-definite. We know that the smallest eigenvalue equaling to zero is measure zero, and it almost surely will not happen. However, for conservative purpose, we can add one more identity matrix to the resulting matrix so that the smallest possible eigenvalue will be 1.
Reference: `http://mathworld.wolfram.com/DiagonallyDominantMatrix.html`

After creating the positive definite matrix with the function with a input of n, I use the built-in function gc to record the peak memory use in the running of chol function. Note that here I cannot simply use the subtraction of memused function becauses I repalce my object with a new object with final size the same, so the absoulte memory difference will be zero.However, because we assume the "sequential overwrite" in part b, so we are interested in the peak of memory use.
Surprisingly, the result does not match my answer in part b. By the difference of max mem use, we observed that the function still had a max memory use for a about the same size as the original matrix. Explicitly, it still creates another copy to save the original matrix while creating the new matrix, and replace that at the end, which might not be smart enough.
From the plots of peak of memory use and time spent for chol function, I find that the max memory use of the chol has a **qudratic relationship** with n, and the time spent has a **cubic relationship** with n. The result is consistent to the conclusion in part A, where the complexity of the Cholesky decomposition is $O(n^3)$.

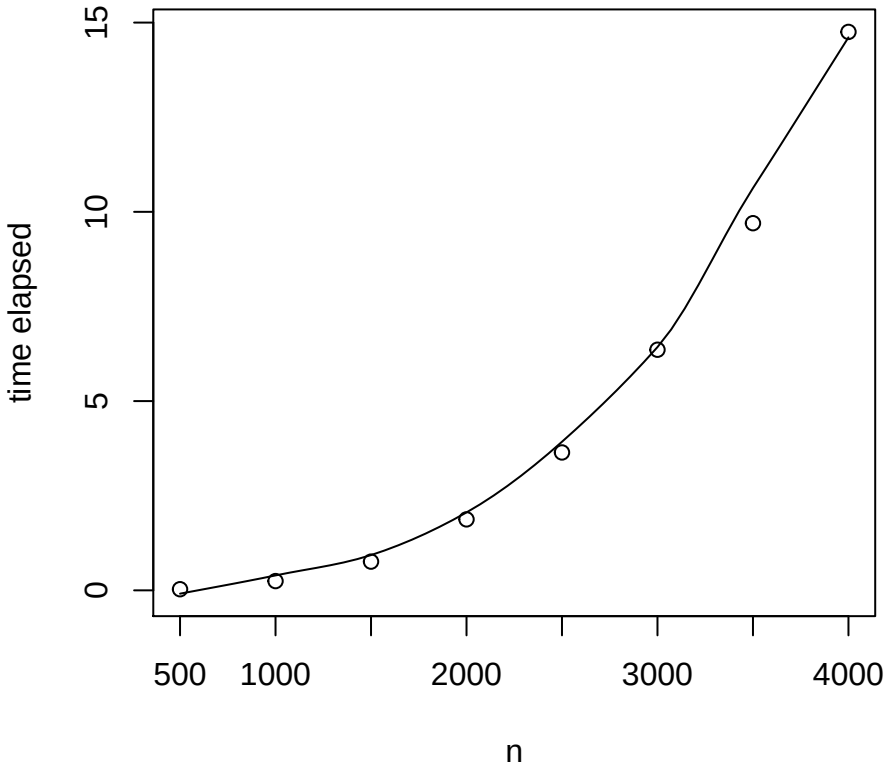
```
posdef_old<-function(n){
  A=matrix(rnorm(2*n*n),nc=n)
  M<-cor(A)
  return(M)
}
posdef<-function(n){
  A=matrix(runif(n^2),nc=n)
  pos_d<-A+t(A)+n*diag(n)
  return(pos_d)
}

mem_time<-function(n){
  C<-posdef(n)
  mem1<-sum(gc(reset=TRUE)[,6])
  time<-system.time(C<-chol(C))
  mem2<-sum(gc()[,6])
  result<-c(mem2-mem1,time[3])
  names(result)<-c("max mem use (Mb)","time spent")
  return(result)
}

run_list<-seq(from=500,to=4000,by=500)
result<-lapply(run_list,mem_time)
result<-do.call(rbind,result)
result<-cbind(run_list,result)
colnames(result)<-c("n","max mem use (Mb)","time spent")
scatter.smooth(result[,1],result[,2],ylab="max mem use (Mb)",xlab="n")
```




```
scatter.smooth(result[,1],result[,3],ylab="time elapsed",xlab="n")
```



Problem 3

3A and B

I have used MAC Air 1.7GHZ I5 in this problem, and different machines will carry out different speed, for instance, a Ubuntu with 2.5GHZ I5 will be approximately one time faster than the MAC machine.

For the illustration purpose, I use $eval = False$ and paste my result into the pdf file so that I won't spend 10 more minutes compiling my document. For this problem, we are comparing three different ways to calculate $b = X^{-1}U$. For n=5000,the full version takes about 300 seconds, and the second method (LU decomposition) elapses about 45 seconds, and the way using chol decomposition yields about 20 seconds for two steps. Approximately, the full version invovles $O(n^3)$ calculation, LU decomposition involves $O(\frac{n^3}{3})$ and the chol decomposition involves $O(\frac{n^3}{6})$ calculations. However, the chol method only spends approximately a half of the time of the LU decomposition.

These results are consistent with the relative order introduced in class, the chol spends approximately half time as the LU decompostion.

As for the precision of three different results, I pick up one element from each result and set the digits to be 22. I found that both of the tree methods: the full version, LU decomposition, and the cholesky decompsition gives me the same answer for the first 15 digits (14 if does not consider rounding), we can conclude that they have 15 digits accuracy up to machine precision. **In conclusion, three methods give the same precision**

We can relate this to the condition number, because condition number represents the loss of accuracy in

digits (compared to 16 digits accuracy of machine precision). By actually calculating the ratio of max and min eigenvalue ($2.02,10^1$),$16-1=15$, which is consistent to the 14 digits accuracy in the calculation.

```
X<-posdef(5000)
y<-rnorm(5000)
system.time(b1<-solve(X)%*%y)

##    user  system elapsed
## 334.080   2.132 338.345

##LU
system.time(b2<-solve(X,y))

##    user  system elapsed
##  44.709   0.341  45.407

## Chol Decomposition
system.time(U<-chol(X))

##    user  system elapsed
##  28.484   0.213  28.863

system.time(b3<-backsolve(U, backsolve(U, y, transpose = TRUE)))

##    user  system elapsed
##   0.071   0.000   0.071

rm(U)
options(digits=22)
b1[1]

## [1] -0.000353027600590731776425 4

b2[1]

## [1] -0.000353027600590731180114 2

b3[1]

## [1] -0.000353027600590730638013 1

### Remove these things to free the future memory use
eval<-eigen(X)
condnumber<-max(eval$values)/min(eval$values)
condnumber

## [1] 2.023582952339720630874

rm(b1,b2,b3)
rm(X)
rm(y)
```

Problem 4

The general framework of the way here I did was essentially utilizing the fact that Σ is a covariance matrix (positive definite and symmetric). Thus, I am able to use the cholesky decompostion to deal with inverses,

and aviod using solve, because cholesky function with backsolve spends about half time compared to the function solve(X,y). My algorithms mainly uses two cholesky decompositions, and the first cholesky decomposition and solving upper triangular system takes majority of time. Now let's first write out the pseudo-code and consider the order of the calculation in each step. The first step is: Do the cholesky decomposition for Σ to calculate the result of $\Sigma_{-1}X$, and because Σ is an $n*n$ matrix, thus the order of operations is $O(\frac{n^3}{6})$. Then we need to solve the upper triangular system like in problem 3 to use the backsolve twice, and the order of operations is $O(n^2p)+O(n^2p) = O(2n^2p)$ the result of this part will be denoted as P_1, and this step takes the majority time of the whole calculation.
The second step is: Calculate the $P_2 = X^T\Sigma^{-1}X = X^TP_1$, where the order is $O(np^2)$ because of the matrix multiplication between an $p*n$ and $n*p$ matrices.
The third step is: Do the Cholesky decomposition for $X^T\Sigma^{-1}X$, which is an $O(\frac{p^3}{6})$.Then we do the final part to backsolve, which is to solve the upper triangular linear systems and get the final answer. The order is $O(\frac{p^3}{6})+O(np)+O(p^2)$
Consider n is to thousands, and p is to hunderds, we may neglect some terms, the overall order is $O(n^3)$ (majorly I think it's $O(\frac{n^3}{6}+2n^2p)$. The more concise algorithm are listed below:

$$
\begin{aligned}
U_1U_1^T = A\ (U_1\ is\ an\ upper\ triangular\ matrix)\\
P_1 = \Sigma^{-1}X = U_1^{-1}(U_1^{T-1}X)\\
P_2 = X^TP_1\\
U_2U_2^T = P_2\ (U_2\ is\ upper\ triangular)\\
Result = U_2^{-1}(U_2^{T-1}P_1^Ty)\ Notice\ that\ P_1^T\ is\ X^T\Sigma^{-1}
\end{aligned}
$$

I use the case $n = 1000, p = 100$ for experiment, and it takes about 0.4 seconds compared to 9 seconds for just using solve function.

```
options(digits=7)
x<-matrix(rnorm(1000*100),nrow=1000)
sigma<-posdef(1000)
y<-rnorm(1000)

gls2<-function(x,sigma,y){
  U1<-chol(sigma)
  p1<-backsolve(U1, backsolve(U1, x, transpose = TRUE))
  p2<-t(x)%*%p1
  U2<-chol(p2)
  last<-t(p1)%*%y
  beta<-backsolve(U2, backsolve(U2, last, transpose = TRUE))
  return(beta)
}
system.time(gls2(x,sigma,y))

##    user  system elapsed
##   0.578   0.012   0.616
```

Problem 5

5A

By definition, if $\vec{v}$ is the right singular vector of X, and $\vec{u}$ is the left singular vector, then:
$X\vec{v} = \sigma\vec{u}$ and $X^T\vec{u} = \sigma\vec{v}$.

$$X^T X\vec{v} = X^T\sigma\vec{u}, \sigma scalar.$$
$$X^T X\vec{v} = \sigma X^T\vec{u}$$
$$X^T X\vec{v} = \sigma\sigma\vec{v} = \sigma^2\vec{v}$$

Thus $\vec{v}$ is an eigenvector of $X^T X$, and σ^2 is the eigenvalue.
Additionally, we can also show that $X^T X$ is positive semi definite, because it has eigenvalues that can be expressed as $\sigma^2 >= 0$. Detaily:
$M = X^T X$ is positive semi definite i f $Z^T MZ >= 0$ for every non-zero Z.
$Z^T MZ = Z^T X^T XZ = ||XZ||_2^2 >= 0$, thus $X^T X$ is positive semi definite.

5B

If $X\vec{v} = \lambda\vec{v}$, then λ is the eigenvalue of X.

$$(X + cI)\vec{v} = X\vec{v} + cI\vec{v} = \lambda\vec{v} + c\vec{v} = (\lambda + c)\vec{v}$$

Therefore, $\lambda + c$ is always the eigenvalue of $X + cI$. Since we have at most n distinct eigenvalues, then the calculation is in $O(n)$, actually, exactly n.