Homework 3

Steven Xu

Due @ 5pm on October 19, 2018

Part 1.

1. Let $\mathbf{A} \in \mathbb{R}^{m \times m}$ be a k-banded matrix, namely $a_{ij} = 0$ if $|i - j| > k$.

```
library(sgxuST758)
```

```
##
## Attaching package: 'sgxuST758'
```

```
## The following object is masked from 'package:base':
##
##     sweep
```

```
library(Matrix)
```

```
set.seed(12345)
k <- 3; m <- 10
A <- matrix(rnorm(m**2), m, m)
A <- A + t(A)
A <- round(10*A)/10
for (i in 1:m) {
  for (j in 1:m) {
    if (abs(i - j) > k) A[i,j] <- 0
  }
}
A
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]  1.2  0.6  0.7  0.4  0.0  0.0  0.0  0.0  0.0   0.0
##  [2,]  0.6  3.6  1.8  2.7 -3.1  0.0  0.0  0.0  0.0   0.0
##  [3,]  0.7  1.8 -1.3  0.5 -2.7  1.9  0.0  0.0  0.0   0.0
##  [4,]  0.4  2.7  0.5  3.3  1.2  0.8  1.3  0.0  0.0   0.0
##  [5,]  0.0 -3.1 -2.7  1.2  1.7  0.8 -2.0  2.0  0.0   0.0
##  [6,]  0.0  0.0  1.9  0.8  0.8  0.6 -1.1  1.8  4.0   0.0
##  [7,]  0.0  0.0  0.0  1.3 -2.0 -1.1  1.8  2.4  1.2  -1.8
##  [8,]  0.0  0.0  0.0  0.0  2.0  1.8  2.4 -1.6  0.2   3.2
##  [9,]  0.0  0.0  0.0  0.0  0.0  4.0  1.2  0.2  1.1   0.2
## [10,]  0.0  0.0  0.0  0.0  0.0  0.0 -1.8  3.2  0.2  -1.4
```

The R snippet above constructs a 3-banded 10-by-10 matrix $\mathbf{A}$. Suppose further that $\mathbf{A}$ is positive definite and therefore admits a unique Cholesky decomposition $\mathbf{A} = \mathbf{R}^\mathsf{T}\mathbf{R}$ where $\mathbf{R} \in \mathbb{R}^{m \times m}$ is upper triangular.

Prove that the banded Cholesky decomposition can be computed in $\mathcal{O}(mk^2)$ flops.

Answer:

For $\mathbf{A} \in \mathbb{R}^{m \times m}$, let $\mathbf{A}^{(i)}$ be the starting matrix of Cholesky factorization at i^{th} stage, $i = 1, ..., m$. Then $\mathbf{A}^{(i)} = \mathbf{A}[i:m, i:m]$.

$$\mathbf{A}^{(i)} = \begin{pmatrix} \alpha^{(i)} & a^{(i)\mathsf{T}} \\ a^{(i)} & \mathbf{A}_*{}^{(i)} \end{pmatrix}$$

By the k-banded construction, only entries $[i, j]$ s.t. $max(i-k, 1) \leq j \leq min(k+i, m)$ of $\mathbf{A}$ have nonzero values. Therefore the first row of $\mathbf{A}^{(i)}$ has nonzero values on columns $[i, max(k+i, m)]$. Therefore the first row of $\mathbf{A}^{(i)}$ has at most $k+1$ nonzero entries, which is equivalent to ${a^{(i)}}^{\mathsf{T}}$ having at most k nonzero elements, i.e. $nnz(a^{(i)}) \leq k$. The total flops of Cholesky factorization of a m by m $k-banded$ matrix at i^{th} stage can be calculated as follow:

$$
\begin{aligned}
&1.\ \rho = \sqrt{\alpha^{(i)}} \ \sim \mathcal{O}(1) \\
&2.\ {r^{(i)}}^{\mathsf{T}} = \rho^{-1}{a^{(i)}}^{\mathsf{T}} \sim \mathcal{O}(nnz(a^{(i)})) \sim \mathcal{O}(k) \\
&3.\mathbf{A}^{sub} = \mathbf{A}_{*}^{(i)} - r^{(i)}{r^{(i)}}^{\mathsf{T}} \sim \mathcal{O}(k^2)
\end{aligned}
$$

Step 3 is of $\mathcal{O}(k^2)$ because ${r^{(i)}}^{\mathsf{T}}$ has at most k^2 nonzero entries. Since we have to repeat the above steps m times. The total flops is of $\mathcal{O}(mk^2)$.

2. Prove that solving a linear system with the Cholesky factor matrix $\mathbf{R}$ of a k-banded matrix $\mathbf{A}$ is $\mathcal{O}(mk)$ flops.

Answer:

Suppose we have

$$
\begin{aligned}
&\mathbf{R}^{\mathsf{T}}\mathbf{R}b = y,\ \text{let } \mathbf{R}b = u \\
&1.\ \mathbf{R}^{\mathsf{T}}u = y \\
&2.\ \mathbf{R}b = u
\end{aligned}
$$

From question 1 we know that $\mathbf{R}$ will have at most k nonzero elements on each row. Therefore when solving $\mathbf{R}^{\mathsf{T}}u = y$, each step of back substitution require at most $2k-1$ flops and we have m steps. Therefore $\mathbf{R}^{\mathsf{T}}u = y \ \sim \mathcal{O}(mk)$. Similarly $\mathbf{R}b = u \ \sim \mathcal{O}(mk)$. Thus solving a linear system with $\mathbf{R}$ is of $\mathcal{O}(mk)$.

Part 2. Banded Cholesky Decomposition

You will next add an implementation of the banded Cholesky decomposition to your R package.

Please complete the following steps.

Step 0: Make a file called `cholesky_banded.R` in your R package. Put it in the R subdirectory, namely we should be able to see the file at github.ncsu.edu/unityidST758/unityidST758/R/cholesky_banded.R

Step 1: Write a function `check_k_banded` that returns `True` if an input matrix `A` is k-banded and `False` otherwise. It should return an error message if $k < 0$ or if k is not an integer. Please use the `stop` function. You may also find the function `is.integer` helpful.

```
#' Check Matrix is k-banded
#'
#' \code{check_k_banded} returns a Boolean variable indicating whether or
#' not a matrix is k-banded.
#'
#' @param A The input matrix
#' @param k bandwidth parameter
#' @export
# check_k_banded <- function(A, k) {
#
# }
```

Step 2: Write a recursive function `chol_banded` that computes the k-banded Cholesky decomposition. **Note:** It is also okay if you don't write a recursive version. In fact, a non-recursive version will probably be better as R has a limit on the number of times a recursive function can be called within itself. But it's fine if you write a recursive version as originally requested too.

```
#' Banded Cholesky
#'
#' \code{chol_banded} computes the Cholesky decomposition of a banded
#' positive definite matrix.
#'
#' @param A The input matrix
#' @param k bandwidth parameter
#' @param checks_off Boolean variable to turn on / off checks
#' @export
# chol_banded <- function(A, k, checks_off=FALSE) {
#
# }
```

- Use sparse matrices via `Matrix` package. Be sure to put the line

  `Depends: Matrix`

  in your `DESCRIPTION` file.
- Your function should return an error message if the input matrix is not k-banded (**Hint:** Use your `check_k_banded` function you wrote in Step 1).
- Your function should return an error message if the input matrix is not symmetric
- Your function should return a sparse uppertriangular matrix `R`.

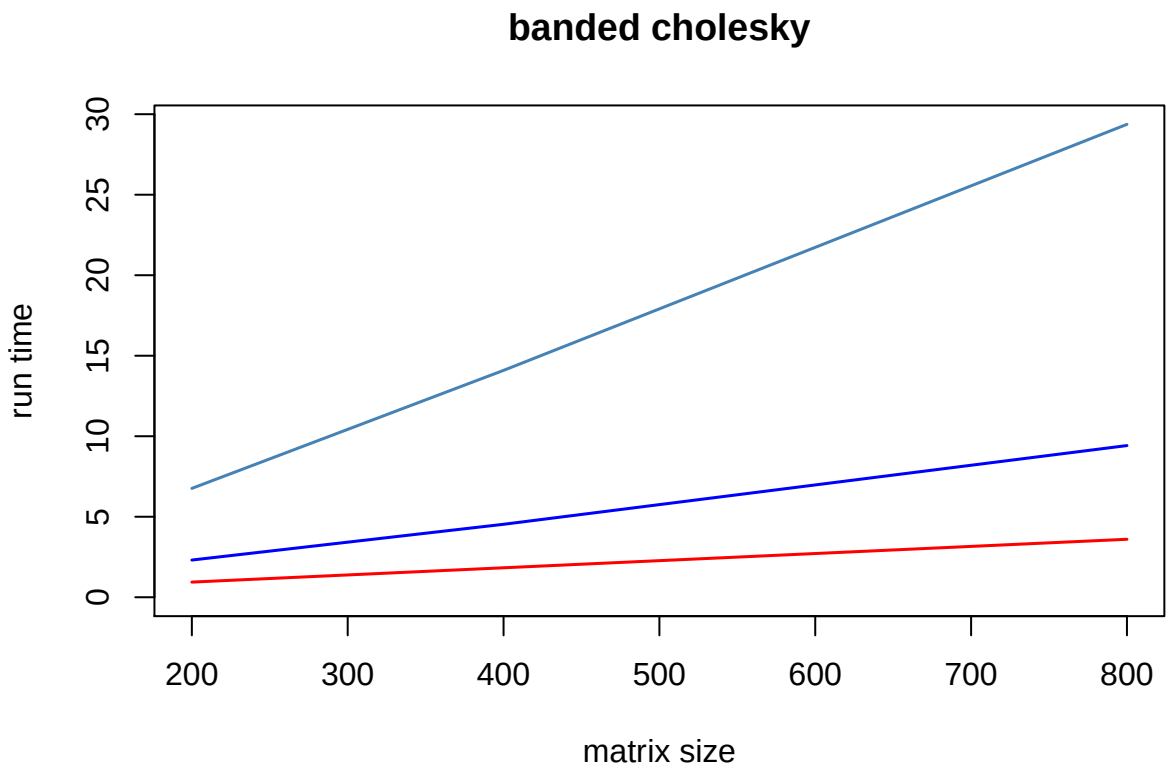
Step 3: Write a unit test function `test-cholesky` that

- checks the error messages for `check_k_banded`
- checks the correctness of your `check_k_banded` function
- checks the error messages for `chol_banded`

- checks the correctness of your `chol_banded` output matrix with that obtained by R's `chol` function.

Step 4: Apply your `chol_banded` function on k-banded positive definite matrices of several sizes n and k, and plot the run times using `system.out`. Plot runtime versus n and overlay the curves for $k = 2, 4, 8$. Does the run time scale as you'd expect?

```
x = c(200,400,800)
time = function(k,size){
  y = matrix(rnorm(size),ncol=1)
  A <- matrix(rgamma(size,5,5)+1, ncol=1)
  A <- tcrossprod(A)
  A[lower.tri(A)] <- 0
  for (i in 1:size) {
    for (j in 1:size) {
      if (abs(i - j) > k) A[i,j] <- 0
    }
  }
  A <- round(10*A)/10
  A = t(A)%*%A
  t1 = as.numeric(system.time(R<-chol_banded(A,k,TRUE))[3])
  return(t1)
}
time_vec2 = sapply(x,k=2,time)
time_vec4 = sapply(x,k=4,time)
time_vec8 = sapply(x,k=8,time)
plot(x,time_vec8,xlab = "matrix size",ylab="run time",main="banded cholesky",type="l",lwd=1.5,col="steel
lines(x,time_vec2,lwd=1.5,col=2)
lines(x,time_vec4,lwd=1.5,col=4)
```



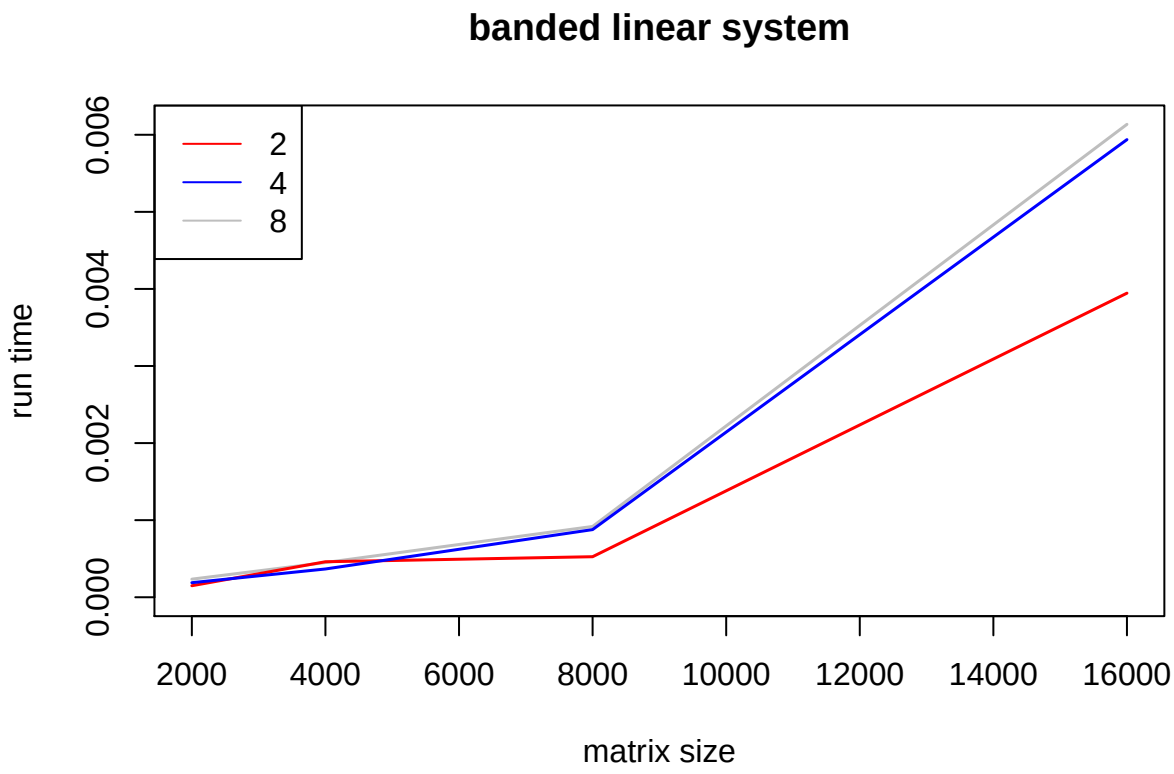

```
size_vec = rep(x,3)
k_vec = rep(c(2,4,8),each=length(x))
nk2 = size_vec*(k_vec)^2
time_vec = c(time_vec2,time_vec4,time_vec8)
summary(fit <- lm(time_vec~nk2))
```

```
##
## Call:
## lm(formula = time_vec ~ nk2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.18798 -0.41718  0.03343  0.33145  1.47202
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 9.178e-01  3.928e-01   2.336   0.0521 .
## nk2         5.492e-04  1.945e-05  28.231  1.8e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8984 on 7 degrees of freedom
## Multiple R-squared:  0.9913, Adjusted R-squared:   0.99
## F-statistic:   797 on 1 and 7 DF,  p-value: 1.798e-08
```

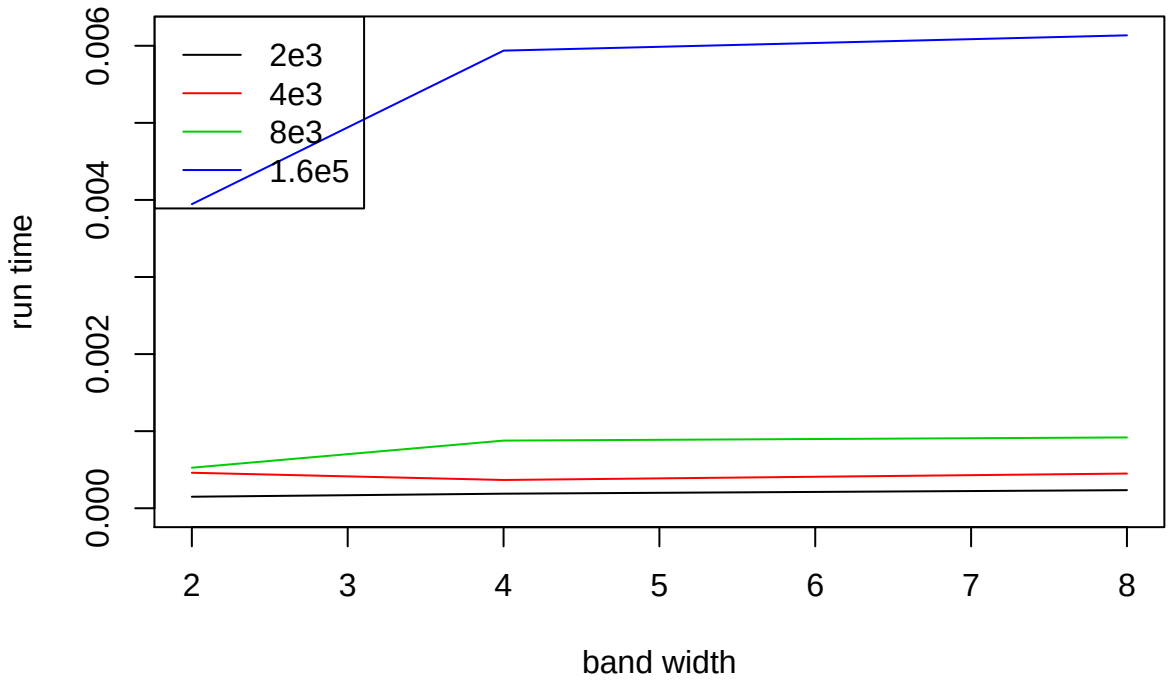
We can see that the linear regression fit of runtime$\sim mk^2$ has a R-squared of close to 1, proving that the banded cholesky has a computation complexity of $\mathcal{O}(nk^2)$.

Step 5: Generate random right hand side vectors $\mathbf{y}$ of several sizes n. Using the Cholesky factor matrices R obtained for k-banded positive definite matrices of sizes n and $k = 2, 4, 8$, plot the run times solving the linear system $\mathbf{R}^{\mathsf{T}}\mathbf{u} = \mathbf{y}$ plus the run time of solving the linear system $\mathbf{R}\mathbf{x} = \mathbf{u}$ as a function of n for $k = 2, 4, 8$. Use the function `Matrix::solve` with the parameter `system` = `"L"` and `"Lt"` respectively. Does the run time scale as you'd expect?

```
rm(list=ls())
x = c(2000,4000,8000,16000)
time = function(k,size){
   y = matrix(rnorm(size),ncol=1)
   A <- matrix(rgamma(size,5,5)+1, ncol=1)
   A <- tcrossprod(A)
   A[lower.tri(A)] <- 0
   for (i in 1:size) {
     for (j in 1:size) {
       if (abs(i - j) > k) A[i,j] <- 0
     }
   }
   A <- round(10*A)/10
   A = Matrix(A,sparse=TRUE)
   t_start = Sys.time()
   for(i in 1:1000) {
      u = solve(t(A),y)
      s = solve(A,u)
   }
   t_end = Sys.time()
   t2 = as.numeric(t_end-t_start)/1000
   return(t2)
}
time_vec2 = sapply(x,k=2,time)
time_vec4 = sapply(x,k=4,time)
time_vec8 = sapply(x,k=8,time)
time_vec = c(time_vec2,time_vec4,time_vec8)
k_vec = rep(c(2,4,8),each=length(x))
plot(x,time_vec8,xlab = "matrix size",ylab="run time",main="banded linear system",type="l",lwd=1.5,col=8
lines(x,time_vec2,lwd=1.5,col=2)
lines(x,time_vec4,lwd=1.5,col=4)
legend("topleft",legend=c(2,4,8),lty=1,col=c(2,4,8))
```




```
plot(c(2,4,8),time_vec[c(1,5,9)],type="l",ylim=c(0,max(time_vec)),xlab="band width",ylab="run time")
lines(c(2,4,8),time_vec[c(2,6,10)],col=2)
lines(c(2,4,8),time_vec[c(3,7,11)],col=3)
lines(c(2,4,8),time_vec[c(4,8,12)],col=4)
legend("topleft",legend=c("2e3","4e3","8e3","1.6e5"),lty=1,col=c(1,2,3,4))
```



We can see that the run time doesnot seem to be perfectly linear in neither matrix size and band width which contradicts with our mathematical result. This might be bcause that the matrix size is still not big enough to make th algorithm to perform stable enough. Or, the built-in solve function might have run time not specifically designed for k banded matrices.