Timing for linear algebra

Short Course

Timing the spatialProcess function

Motivation for problems as problem size gets large

Note: loading **LatticeKrig** package automatically loads **fields**.

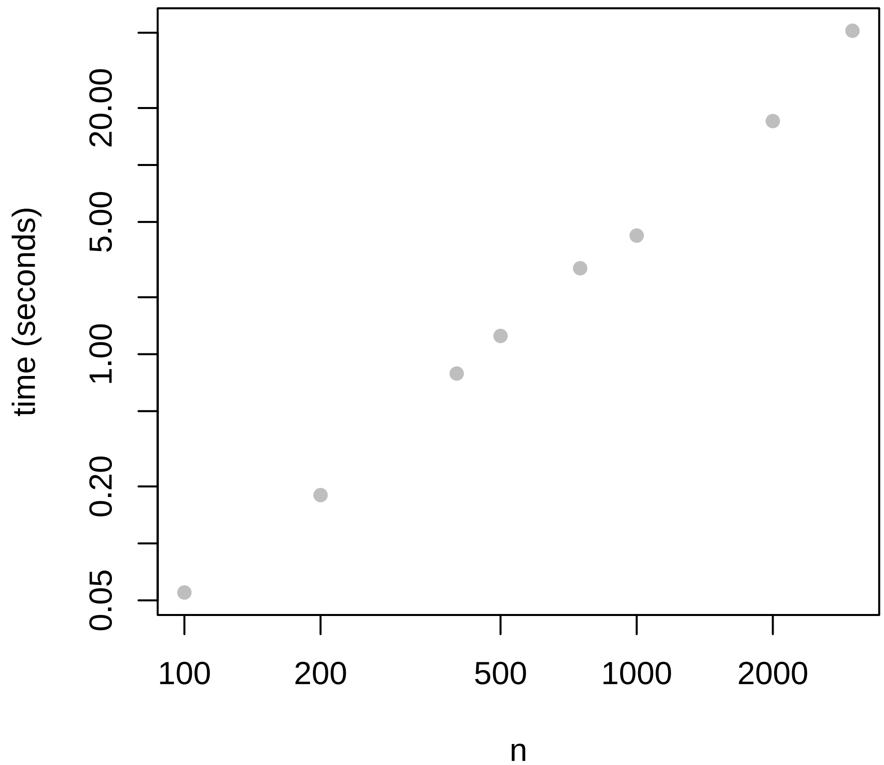
```
set.seed(222)
tabSP<- NULL
nObs<-
  c(100, 200, 400,  500, 750, 1000, 2000, 3000)
for( n in nObs ){
  x<- cbind( seq( 0,1,,n))
  Sigma<- Matern( rdist(x,x)/.15, smoothness = 1.0)
  U<- rnorm(n)
  E<- rnorm(n)
# add in a linear part too to match fitting
  y<-   (1 + x) + t(chol(Sigma))%*%U  + .05*E
  # fix the range to compare with example later on
  elapsedTime<- system.time(
              out<- spatialProcess( x,y, aRange=.15 )
        )
  #print(c( n,elapsedTime[3]) )
  tabSP<-  rbind( tabSP, c( n,elapsedTime[3]) )
}

print( tabSP)
```

```
##           elapsed
## [1,]  100   0.055
## [2,]  200   0.180
## [3,]  400   0.790
## [4,]  500   1.249
## [5,]  750   2.845
## [6,] 1000   4.236
## [7,] 2000  17.063
## [8,] 3000  51.230
```

Take a look at a log-log plot. Linear in log-log means a polynomial relationship.

```
plot( tabSP, log="xy", xlab="n", ylab="time (seconds)",
        col="grey", pch=16)
```




```
y<- log10(tabSP[,2])
x<- log10(tabSP[,1])
lm( y~x)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)            x
##      -5.277        1.987
```

Timing just the Cholesky

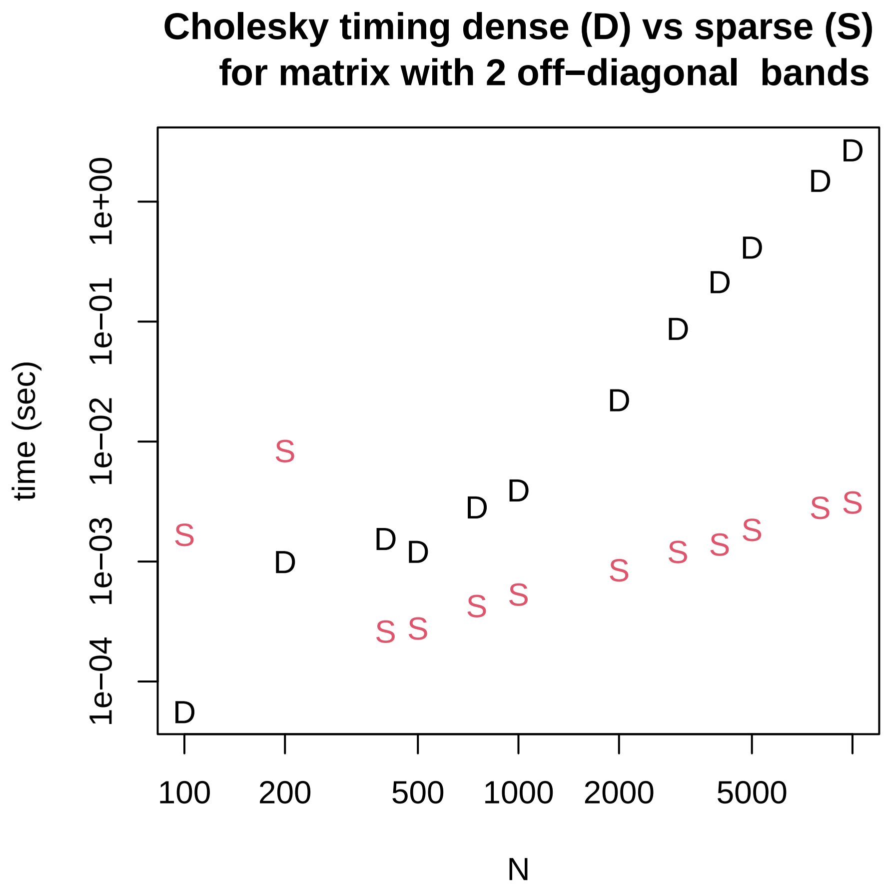
Focusing just on the Cholesky decoposition – theoretically the most time consuming step. Use a test matrix that has lots of zeroes and compare timing to sparse Cholesky decomposition.

```
sizes<- c(100, 200, 400,  500, 750, 1000, 2000, 3000, 4000, 5000, 8000, 10000)
NTotal<- length( sizes)
tabChol<- matrix( NA, nrow= NTotal, ncol=4)
dimnames(tabChol)<- list( NULL, c("N","Dense",
                              "Sparse","speedup"))
for(k in 1:NTotal) {
 N<- sizes[k]
#weights are a 4th differece
# sparse matrix construction using LatticeKrig utility
SMat <- LKDiag( c(1,  -10,   27,  -10,    1), N)
# convert to full ( now the zeroes are consider real values)
FMat <- spam2full(SMat)
# dense matrix Cholesky
startTime <- Sys.time() #
FChol <- chol(FMat)
deltaF<- as.numeric(Sys.time() - startTime) #
# sparse matrix Cholesky
startTime <- Sys.time()
SChol <- chol(SMat)
deltaS<- as.numeric(Sys.time() - startTime )
tabChol[k,]<- c(N,deltaF, deltaS, deltaF/deltaS )
}
print( tabChol)
```

```
##           N        Dense       Sparse      speedup
##  [1,]   100 5.602837e-05 0.0016839504   0.03327198
##  [2,]   200 9.989738e-04 0.0084080696   0.11881132
##  [3,]   400 1.554966e-03 0.0002629757   5.91296464
##  [4,]   500 1.215935e-03 0.0002789497   4.35897436
##  [5,]   750 2.849817e-03 0.0004289150   6.64424680
##  [6,]  1000 3.949881e-03 0.0005350113   7.38279857
##  [7,]  2000 2.229905e-02 0.0008530617  26.14002236
##  [8,]  3000 8.770394e-02 0.0012121201  72.35582219
##  [9,]  4000 2.151361e-01 0.0013971329 153.98395904
## [10,]  5000 4.174101e-01 0.0018510818 225.49523442
## [11,]  8000 1.502816e+00 0.0028290749 531.20402832
## [12,] 10000 2.699010e+00 0.0031311512 861.98644636
```

Log- log plot to look for polynomial dependence

```
matplot( tabChol[,1], tabChol[,2:3], type="p", pch=c( "D","S"),
xlab="N", ylab="time (sec)", log="xy")
title("Cholesky timing dense (D) vs sparse (S)
      for matrix with 2 off-diagonal  bands ")
```



Cholesky timing dense (D) vs sparse (S)
for matrix with 2 off−diagonal bands
time (sec)
1e−04
1e−03
1e−02
1e−01
1e+00
100
200
500
1000
2000
5000
N

On your own . . .

1. Extrapolating from the smaller sample results in **tabS** estimate the time for **spatialProcess** to handle a problem of size 26000 (about the size of the CO2 data set.)

2. Is the time for spatialProcess (**tabSP[,2]**) linearly related to the time for the Cholesky decomposition (**tabChol[1:6, 2]**)?