Vecchia Approximation

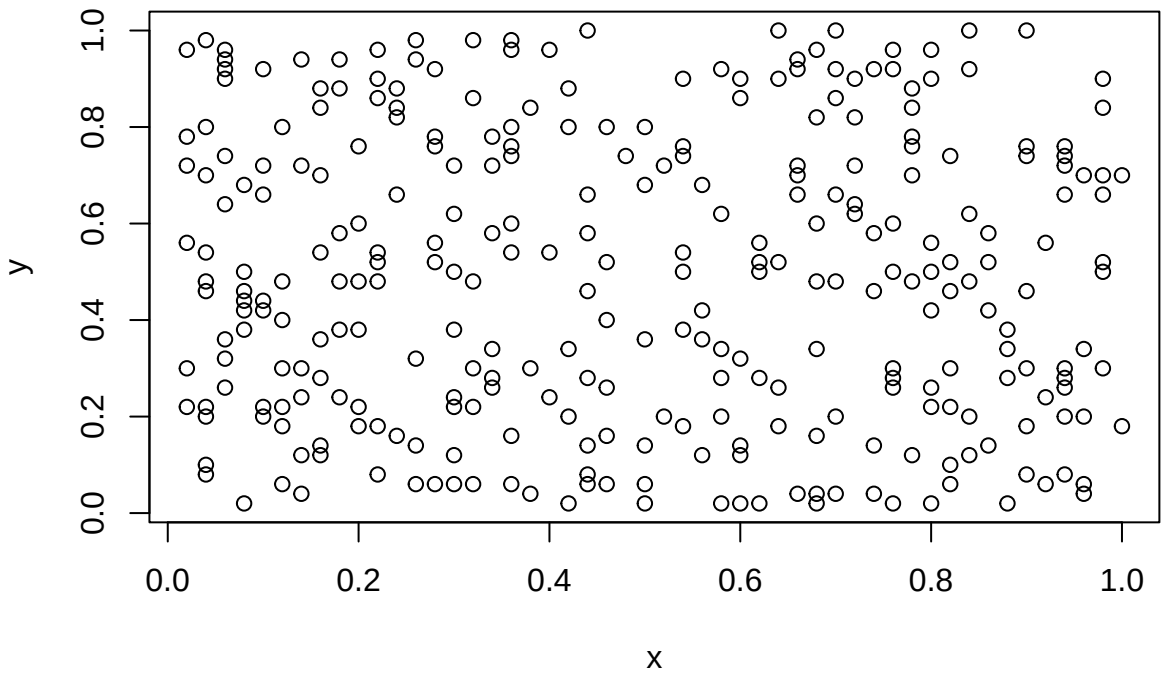
2022-12-12

Load packages:

```
library(GpGp)
library(monomvn)
library(Matrix)
library(SparseM)
library(vecchia)
```

We will use $n = 300$, although the approximation ideas here are meant for much larger n. The code below defines and plots $t_1, \ldots, t_n \in [0,1]^2$. Row i of `locs` contains the (x, y) coordinates of t_i.

```
nvec <- c(50,50)
n <- prod(nvec)
locs <- as.matrix(expand.grid((1:nvec[1])/nvec[1],(1:nvec[2])/nvec[2]))
locs <- locs[sample(n, 300),]
n <- nrow(locs)
plot( locs[,1], locs[,2], ylab = "y", xlab = "x")
```



Next, we generate a covariance matrix

$$\Sigma_{ij} = C(t_i, t_j)$$

We use the exponential covariance

$$C(t_1, t_2) = \sigma^2 \exp(-\|t_1 - t_2\|/\alpha)$$

with

$$(\sigma^2, \alpha) = (1, 0.4)$$

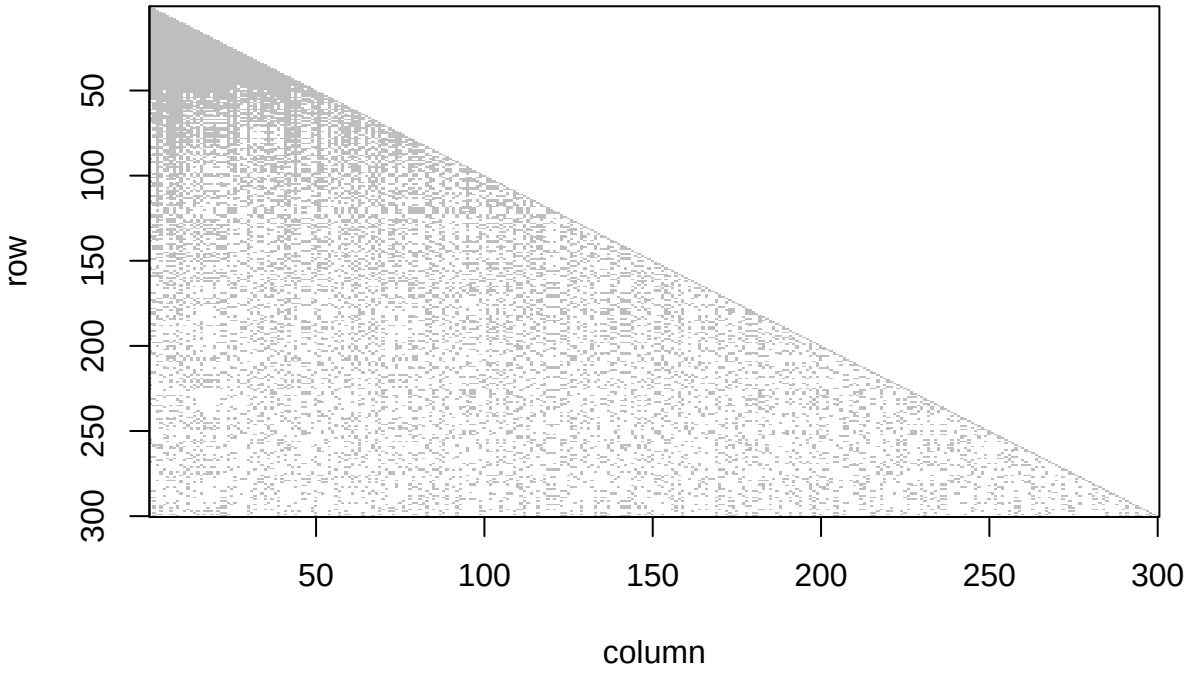
```
covparms <- c(1,0.4,0)
Sigma <- exponential_isotropic( covparms, locs )
```

We define and visualize a sparsity pattern using the nearest neighbor approach and construct the Vecchia approximation based on this sparsity pattern.

```
m = 45
NNarray <- find_ordered_nn(locs, m)
system.time(GammaNN <- vecchia_Linv(covparms, "exponential_isotropic", locs, NNarray))
```

```
##    user  system elapsed
##   0.009   0.001   0.008
```

```
image(as.matrix.csr(expandm(GammaNN, NNarray)))
```



We evaluate the approximation using KL-divergence

```
SigmaNN <- solve(t(expandm(GammaNN, NNarray))%*%expandm(GammaNN, NNarray) )
log10(kl.norm(rep(0,n), Sigma, rep(0,n), SigmaNN, quiet<-FALSE))
```

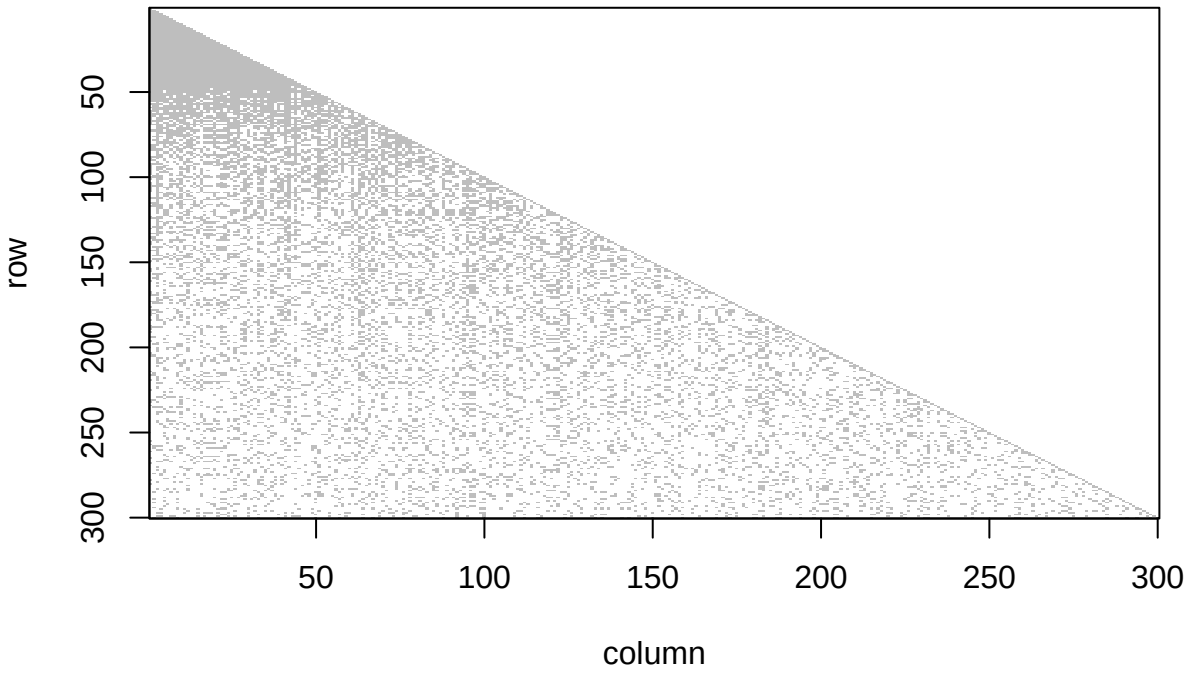
```
## [1] -2.595672
```

Next, we construct the Vecchia approximation with a dynamically obtained sparsity pattern. First we use DyanmicF, which is slow because it searches over a lot of things.

```
NNarray2 <- find_ordered_nn(locs, n)
system.time(obj <- FSPAI(get("exponential_isotropic"), covparms, Sigma, NNarray2, locs, m ))
```

```
##    user  system elapsed
##   9.077   0.083   9.164
```

```
Gamma_DynamicF <- obj$Linv
NNarrayF <- obj$NNarray
image(as.matrix.csr(expandm(Gamma_DynamicF, NNarrayF)))
```



The Kl-divergence is lower than before (approximation is better)

```
Sigma_DynmaicF <-  solve(t(expandm(Gamma_DynamicF, NNarrayF))%*%expandm(Gamma_DynamicF, NNarrayF) )
log10(kl.norm(rep(0,n), Sigma, rep(0,n), Sigma_DynmaicF, quiet<-FALSE))
```

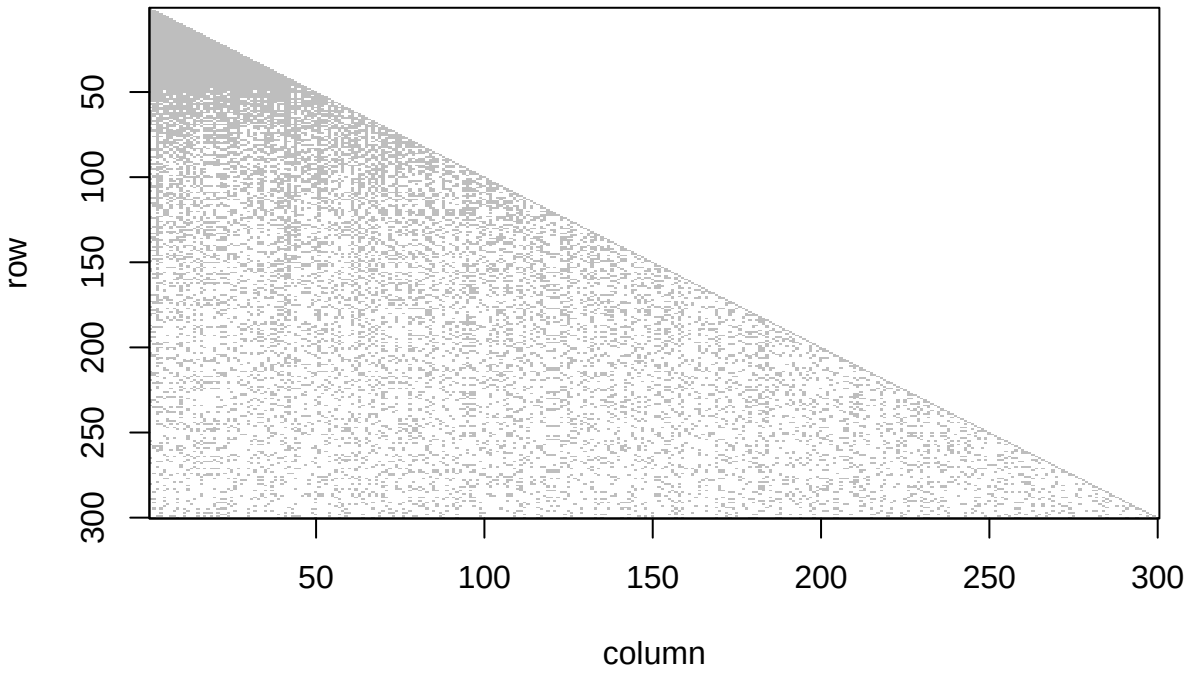
```
## [1] -3.549901
```

Now we use DyanmicS, which is faster than DynamicF because it searches over less things

```
NNarray2 <- find_ordered_nn(locs, round(1.5*m))
system.time(obj <- FSPAI(get("exponential_isotropic"), covparms, Sigma, NNarray2, locs, m ))
```

```
##    user  system elapsed
##   3.545   0.018   3.565
```

```
Gamma_DynamicS <- obj$Linv
NNarrayS <- obj$NNarray
image(as.matrix.csr(expandm(Gamma_DynamicS, NNarrayS)))
```




```
Sigma_DynmaicS <- solve(t(expandm(Gamma_DynamicS, NNarrayS))%*%expandm(Gamma_DynamicS, NNarrayS) )
log10(kl.norm(rep(0,n), Sigma, rep(0,n), Sigma_DynmaicS, quiet <-FALSE))
```

```
## [1] -3.348042
```

The approximation is essentially as good as the DynamicF approach, and it takes less time to compute. It is still not as fast as the nearest neighbor approach but the approximation is better.