Random Correlation Matrix

Contents

The problem

Suppose that we have a partial defined symmetric matrix $A(x) \in \mathbb{S}^n$ depending on the vector of missed entries $x \in \mathbb{R}^m$. Suppose that diagonal elements are fixed to one. Denote by F_A the set $\{x \in \mathbb{R}^m : A(x) \in \mathbb{S}^n_+\}$ of all feasible solutions for the missed correlation problem. We are interested on finding a procedure to draw a random sample of points from F_A when such set is non-empty.

Proposed Solutions

I figured out two procedures to construct a random point of F_A, one based on Sylvester's criterion and the other based on the fact that the set F_A is convex and that we can always compute the maximum determinant point.

Using the maxdet point

Consider the partial defined symmetric matrix $A(x)$ defined by the rule

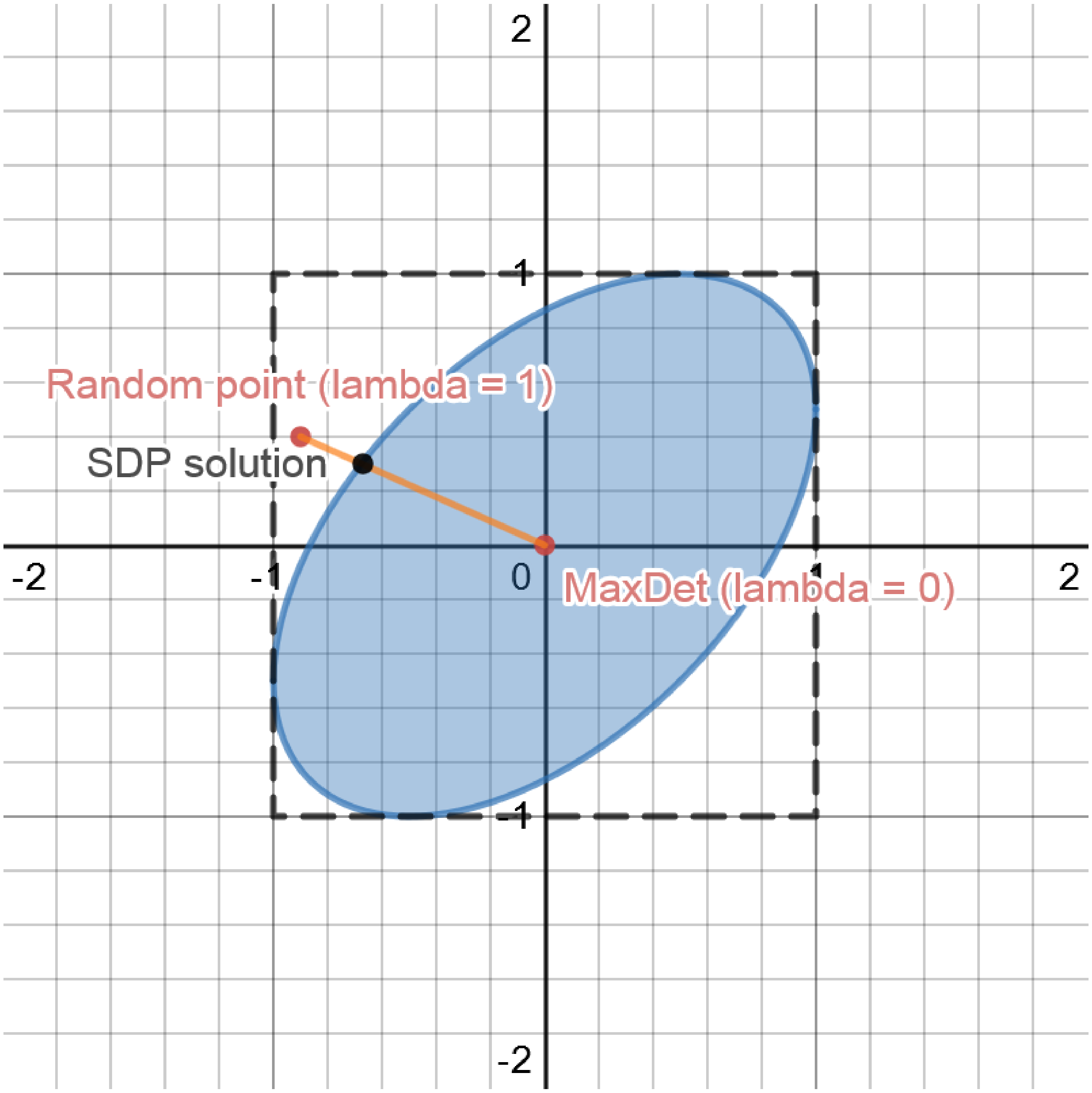
$$(x_1, x_2) \in \mathbb{R}^2 \longrightarrow \begin{bmatrix} 1 & x_1 & 0.5 \\ x_1 & 1 & x_2 \\ 0.5 & x_2 & 1 \end{bmatrix}$$

Let's call x_{MaxDet} the solution to the MDMCP (Maximum Determinant Missed Correlation Problem). For this case, this point is equal to the origin.

We first begin by drawing a random point x_0 from the cube $[-1, 1]^2$ and if $A(x_0)$ is not a correlation matrix, then we proceed as follow:

- Consider the line segment that joins x_0 and x_{MaxDet}, i.e. the set of all convex combinations between these points ($\{\lambda x_0 + (1 - \lambda)x_{MaxDet} : \lambda \in [0, 1]\}$).
- We know that x_{MaxDet} is a point from the relative interior of F_A, so the intersection between the line segment considered above and the feasible set F_A must be non-empty; moreover, if F_A has Lebesgue measure (in R^2) greater than 0, such intersection is non-empty and uncountable as well (this will get clear by looking at the figure). This is a necessary condition in order to be able to draw an uniform sample from F_A.
- From now on, we can proceed in two different ways: first, we can calculate the maximum λ ($\lambda \in [0, 1]$) that makes $A(\lambda x_0 + (1 - \lambda)x_{MaxDet})$ a correlation matrix. This can be done by solving (again!) a Semidefinite program. The alternative is to draw a random λ_0 from a uniform distribution $U(0, 1)$ and look if $A(\lambda_0 x_0 + (1 - \lambda_0)x_{MaxDet})$ is a correlation matrix; if it is not, we draw again a $lambda_1$ from $[0, \lambda_0]$ and repeat the process until we have a correlation matrix.

For this example, by applying Sylvester's criterion, $A(x)$ is a correlation matrix iff $x_1^2 + x_2^2 - 2x_1x_2 \leq 0.75$. The figure below illustrates the above procedure.



Using Sylverster's criterion

An alternative is to use Sylvester's criterion to construct such correlation matrix.

- Consider the 3×3 principal leading minor (upper left corner of $A(x)$); replace the missed correlations with uniform random variables and check if the determinant of such matrix is non-negative. If the resulting matrix has negative determinant, repeat until obtaining a principal minor with positive determinant
- Now repeat the above step but considering the 4×4 principal leading minor.
- Repeat until considering the $n \times n$ principal leading minor (i.e $A(x)$)

This procedure is faster than random sampling and then discarding the non-correlation matrices.

Examples with code Let's draw a random sample of 1000 points from the F_A given above, using the first method.

```
source("randcorr.R")

A = matrix(rep(NA,9),3,3)
diag(A) = 1
A[1,3] = A[3,1] = 0.5
```

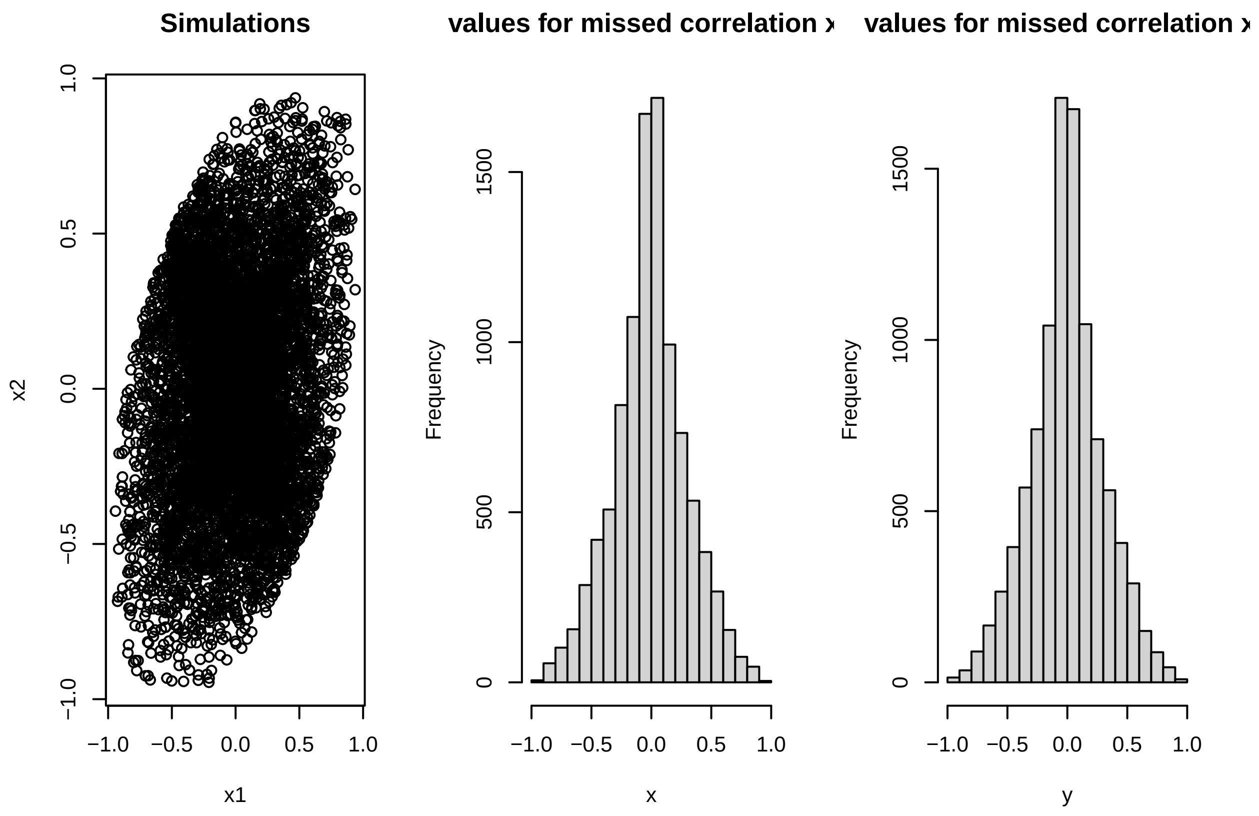
```
N = 10000
sim1 = randcorr_maxdet_maxlambda(N,A, verbose = F) #let's simulate 10000 correlation matrices
```

```
## [1] "Solver succesfully converged!!!"
```

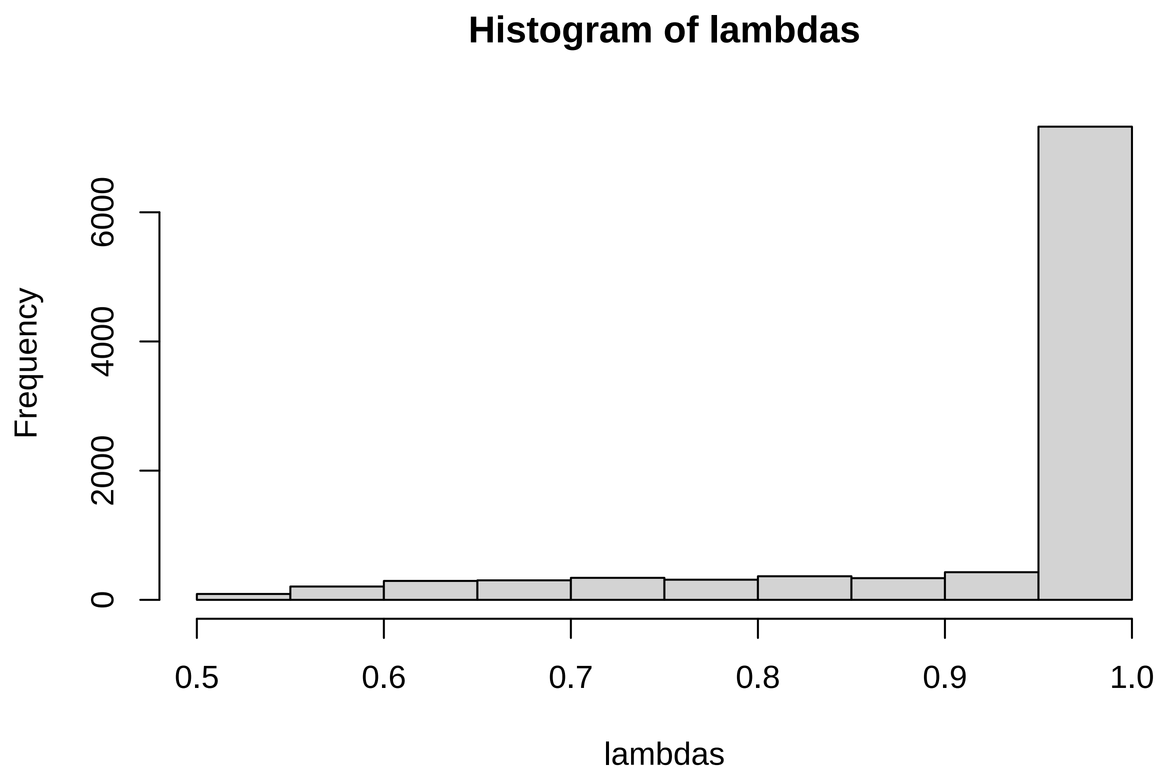
```
sum(sapply(1:N, function(k) is.PD(sim1$matrices[[k]])))
```

```
## [1] 10000
```

```
x = sapply(1:N,function(i) sim1$matrices[[i]][1,2])
y = sapply(1:N,function(i) sim1$matrices[[i]][3,2])

par(mfrow = c(1,3))

plot(x,y, main = "Simulations", xlab = "x1", ylab = "x2")
hist(x, main = "values for missed correlation x1")
hist(y, main = "values for missed correlation x2")
```



```r
lambdas = sapply(1:N, function(k) sim1$lambdas[[k]])
hist(lambdas)
```



We can see that for small n, the maxdet method is not that different from a simple random sampling.

The function that solves the SDP looking for a λ is very computationally expensive. So let's look at the alternatives:

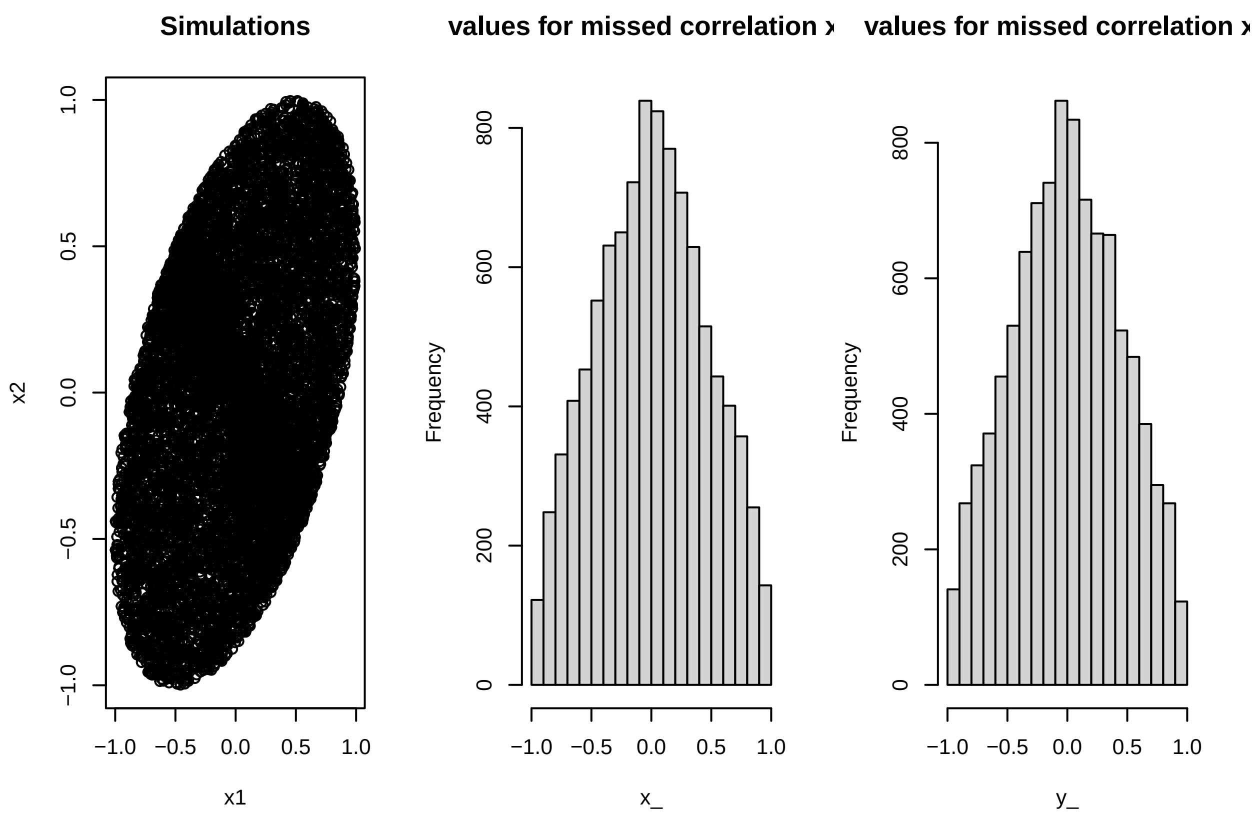
```r
sim2 = randcorr_maxdet_unif(N,A, verbose = F) #let's simulate 10000 correlation matrices using maxdet
```

```
## [1] "Solver succesfully converged!!!"
```

```r
sum(sapply(1:N, function(k) is.PD(sim2$matrices[[k]])))
```

```
## [1] 10000
```

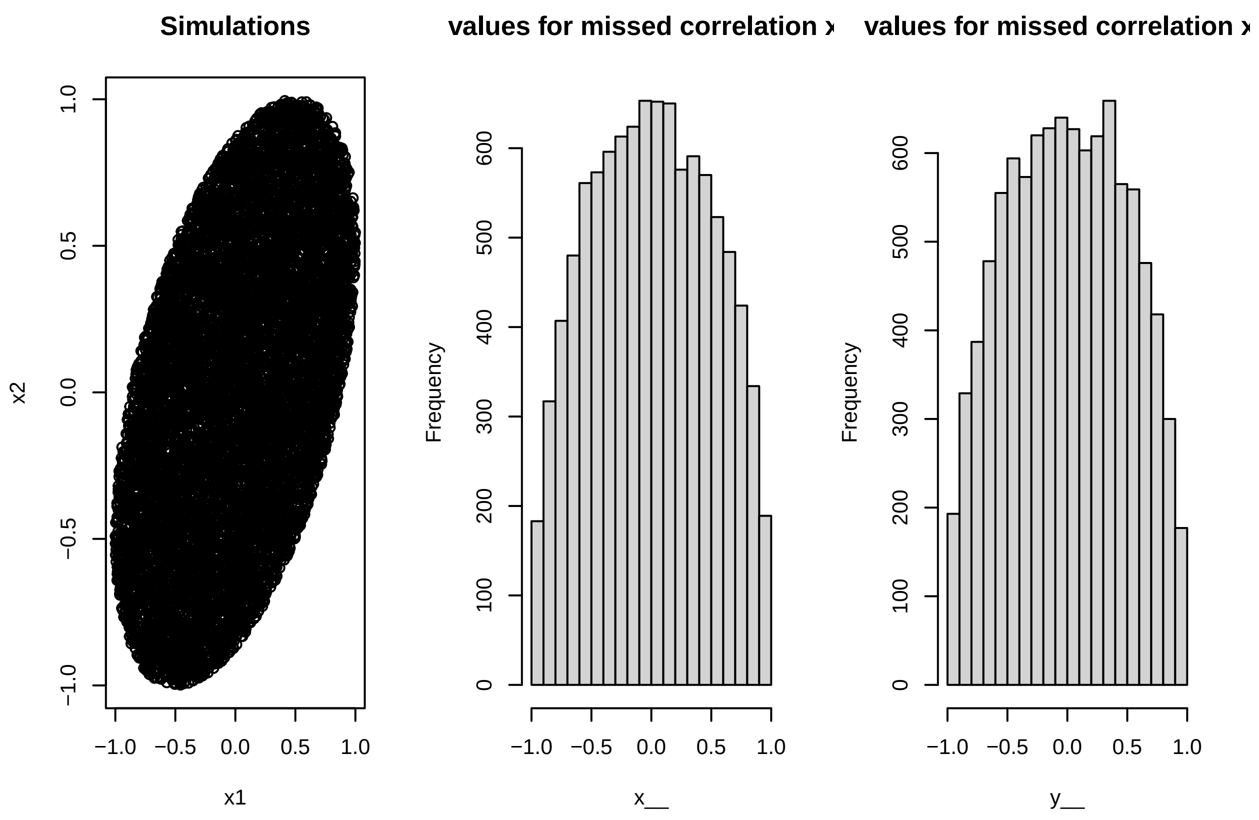
```r
x_ = sapply(1:N,function(i) sim2$matrices[[i]][1,2])
y_ = sapply(1:N,function(i) sim2$matrices[[i]][3,2])

par(mfrow = c(1,3))


plot(x_,y_, main = "Simulations", xlab = "x1", ylab = "x2")
hist(x_, main = "values for missed correlation x1")
hist(y_, main = "values for missed correlation x2")
```



```
lambdas = sapply(1:N, function(k) sim2$lambdas[[k]])

sum(lambdas==1)
```

```
## [1] 6840
```

```
sim3 = randcorr_sylv(10000,A, verbose = F) #let's simulate 10000 correlation matrices

x__ = sapply(1:N,function(i) sim3[[i]][1,2])
y__ = sapply(1:N,function(i) sim3[[i]][3,2])

par(mfrow = c(1,3))


plot(x__,y__, main = "Simulations", xlab = "x1", ylab = "x2")
hist(x__, main = "values for missed correlation x1")
hist(y__, main = "values for missed correlation x2")
```



By looking the plots above, we can see that the method that samples more uniformly from the feasible set F_A is the algorithm based on Sylvester's criterion and rejection sampling.

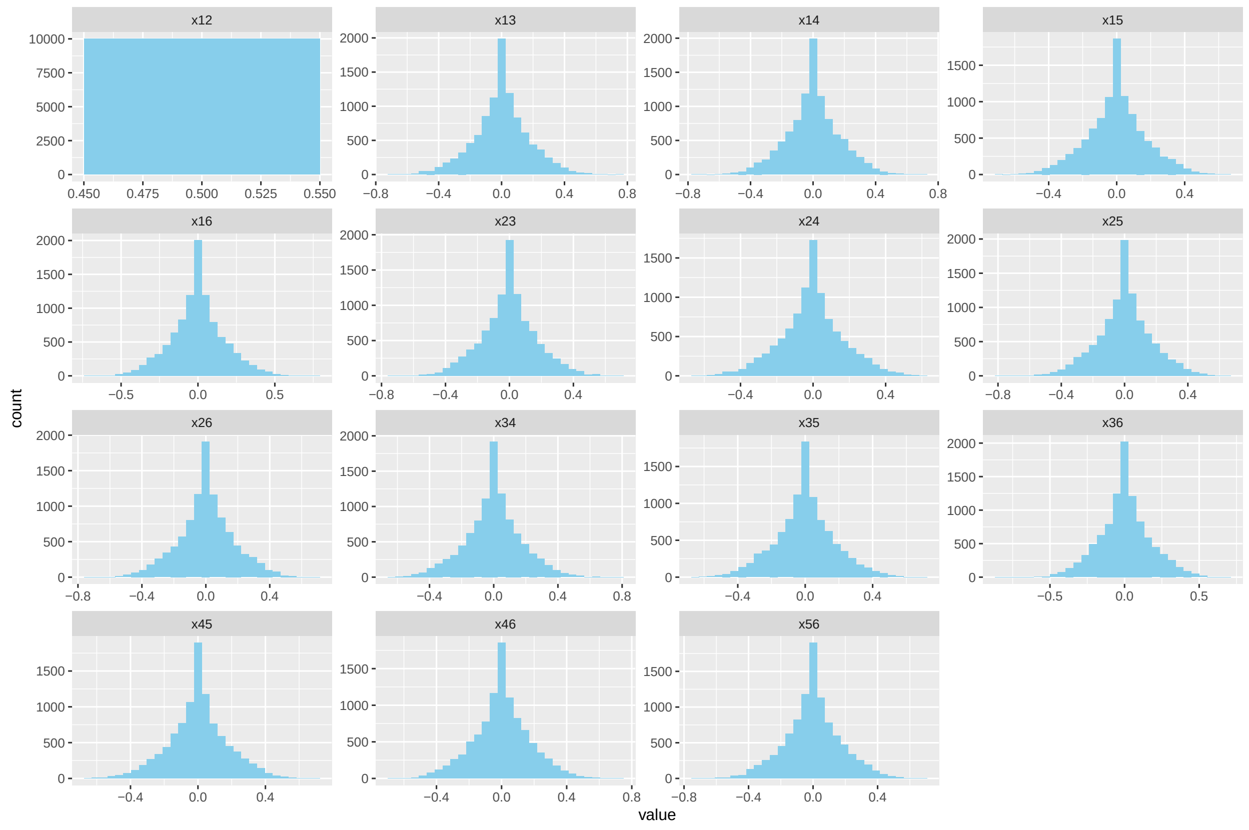
For the example above, the probability of obtaining a point of F_A by uniformly sampling the missed entries from $[-1,1]^2$ is about 68%. Let's look the performance when n is relatively large, so simple random sampling will tend to fail.

What if n is bigger?

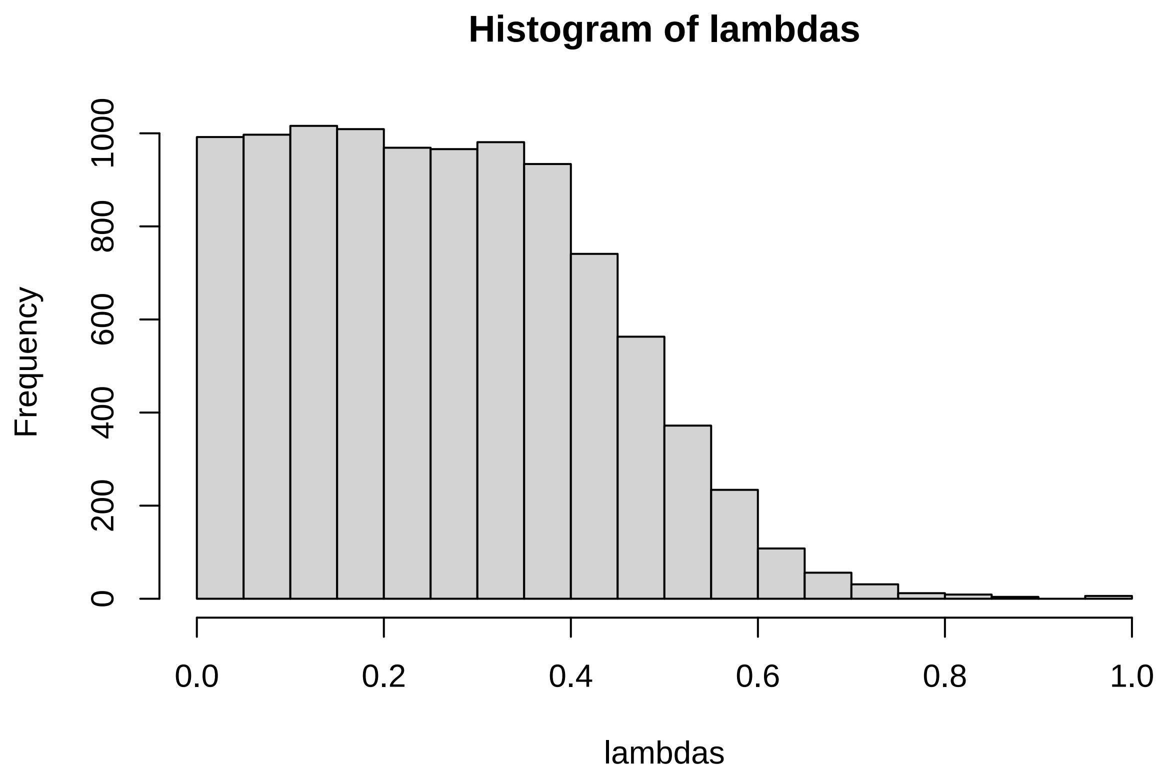
```
library(tidyverse)

n = 6

A_6 = matrix(rep(NA,n^2),n,n)
diag(A_6) = 1
A_6[1,2] = A_6[2,1] = 0.5

sim4 = randcorr_maxdet_unif(N,A_6, verbose = F) #let's simulate 1000 correlation matrices
```

```
## [1] "Solver succesfully converged!!!"
```

```
df = tibble(.rows = N)

for (i in 1:6){
  for (j in 1:6 ){
    if (j<i) {
      df[,paste("x",j,i,sep="")] = sapply(1:N, function(k) sim4$matrices[[k]][i,j])

          }
  }
}

df %>%
  pivot_longer(1:15) %>%
  ggplot(aes(value))+
  geom_histogram(fill = "skyblue")+
  facet_wrap(~name, scales = "free")
```

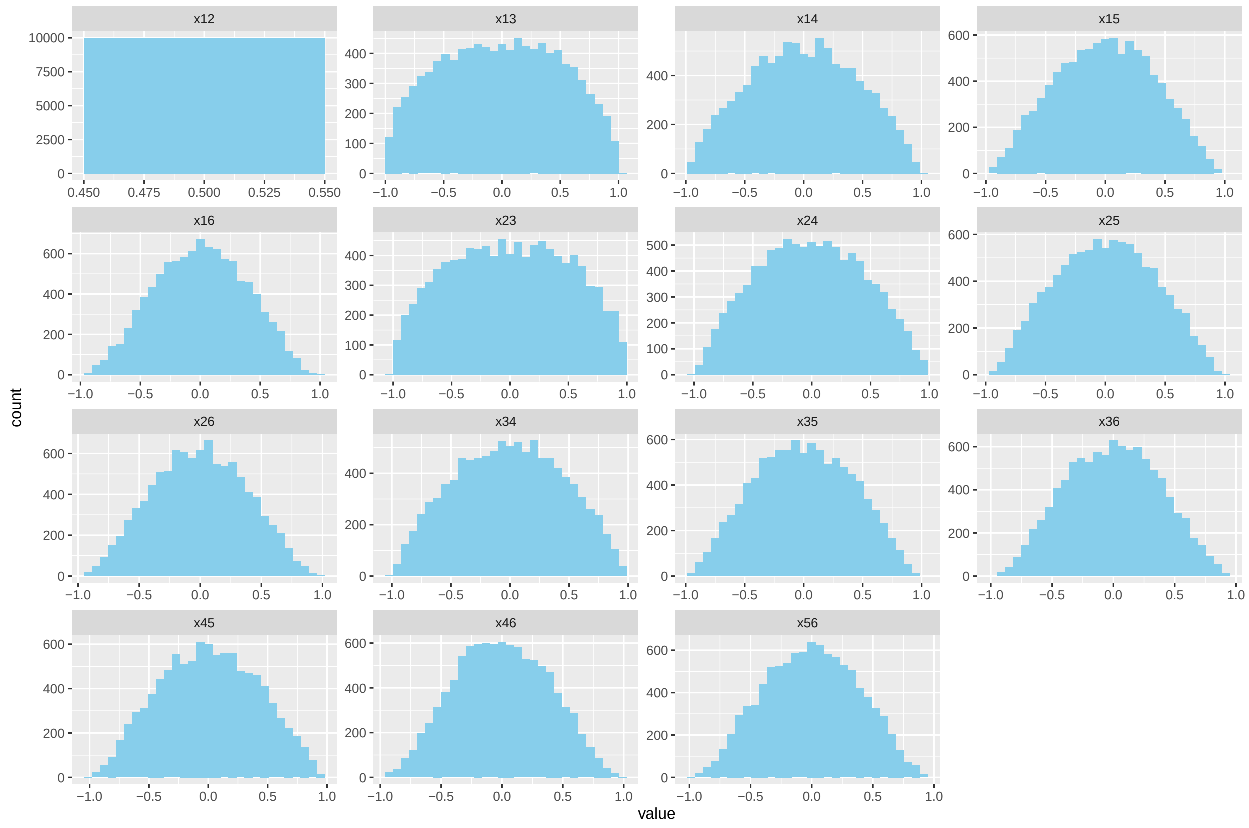


```
lambdas = sapply(1:N, function(k) sim4$lambdas[[k]])
hist(lambdas)
```



As n increases, the shrinking method leads to lambdas points near to 0, so the resulting distribution of missed entries has high density near at the maxdet point.

```
sim5 = randcorr_sylv(N, A_6, verbose = F)


df = tibble(.rows = N)

for (i in 1:6){
  for (j in 1:6 ){
    if (j<i) {

      df[,paste("x",j,i,sep="")] = sapply(1:N, function(k) sim5[[k]][i,j])

          }
  }
}

df %>%
  pivot_longer(1:15) %>%
  ggplot(aes(value))+
  geom_histogram(fill = "skyblue")+
  facet_wrap(~name, scales = "free")
```



From graphs above, we can observe that the distribution of the missed entries varies depending on the method used to generate the sample; the maxdet method leads to marginal distributions with higher densities at the respective entry of the maxdet point.

When the method fails Now, what would happen if F_A has Lebesgue measure of 0?

```
A = matrix(rep(NA,9),3,3)
diag(A) = 1
A[1,3] = A[3,1] = 1 #the feasible set for this matrix is a line segment, with no area.

try(randcorr_maxdet_unif(1000,A))
```

```
## [1] "Warning!!! (The solution is  optimal_inaccurate )"
```

```
A = matrix(rep(NA,16),4,4)
diag(A) = 1

A[1,2] = A[2,1] = 0.6
A[1,3] = A[3,1] = 0.7
A[1,4] = A[4,1] = 0.8
A[4,2] = A[2,4] = 0.9

fill.correlations(A)
```

```
## [1] "Solver succesfully converged!!!"
```

```
## $Sigma_maxdet
##      [,1]      [,2]      [,3]      [,4]
## [1,]  1.0 0.6000000 0.7000000 0.8000000
## [2,]  0.6 1.0000000 0.4200365 0.9000000
## [3,]  0.7 0.4200365 1.0000000 0.5600422
## [4,]  0.8 0.9000000 0.5600422 1.0000000
##
## $Sigma
##      [,1] [,2] [,3] [,4]
## [1,]  1.0  0.6  0.7  0.8
## [2,]  0.6  1.0   NA  0.9
## [3,]  0.7   NA  1.0   NA
## [4,]  0.8  0.9   NA  1.0
##
## $status
## [1] "optimal"
```