Cross Validated

Use R to generate random positive definite matrix with zero constraints

Asked yesterday Active today Viewed 81 times

6

How to use R to generate a random symmetric positive definite matrix with zero constraints?

For example, I would like to generate a 4 by 4 random symmetric positive definite matrix $\Omega \in \mathbb{R}^{4\times 4}$, and we know $\Omega_{1,2} = \Omega_{2,1} = \Omega_{1,3} = \Omega_{3,1} = 0$. How can I do that in R?

What I had in mind is something like Cholesky decomposition $LL^T = \Omega$, where row L_i and row L_j are orthogonal if $\Omega_{ij} = 0$. Possibly solve by the Lagrangian multiplier. But I am not really sure how to implement this. Or if this is possible at all.

r matrix

Share Cite Edit Follow Flag edited yesterday asked yesterday Tan 1,318 11

3 I haven't thought about it deeply, but what I have in my mind is that you cannot randomly generate a PD matrix with an equal probability. Think that any positive constant times the identity matrix is positive definite and we cannot even randomly choose a positive number from positive real numbers unless you employ a proper distribution on the range. – Stati yesterday

@Stati Thanks. I am looking for a general way to generate PD with zero constraints. Of course, cI satisfies the zeros, but it is not general. Also, I understand that some zero structures are simply less "likely", or are hard, to generate a PD random matrix. – Tan yesterday

Yes, cI is an extreme case. However, if you cannot generate it even in a restrictive situation, it would be much harder to do that in general. – Stati yesterday

Generate the non-zero elements in the subdiagonal part of the matrix, eg from an Exponential distribution, complete by symmetry and check for definite positivity. If not, repeat, &tc. – Xi'an yesterday

3 Answers

Active Oldest Votes

3

I'm not sure if this is what you want, but for the *specific example you gave* (this doesn't necessarily generalize easily to *arbitrary* zero constraints, as the algebra can get messy!)

- if L is a lower-triangular matrix with positive values on the diagonal then $\Omega = LL^\top$ is positive definite (requiring the diagonal to be positive is not necessary for positive definiteness, but makes the decomposition unique: see Pinheiro and Bates 1996 "Unconstrained parametrizations for variance-covariance matrices").

- $\Omega_{12} = L_{11}L_{21}$ and $\Omega_{13} = L_{11}L_{31}$. Thus, I *think* that without any further loss of generality, a lower-triangular matrix with a positive diagonal and $L_{21} = L_{31} = 0$ will give you the constraint pattern you want. (Setting $L_{11} = 0$ would give you a singular matrix.)
- "random" is pretty vague. (You didn't say "uniform" ...) We could for example pick $\theta_{ii} \sim U(0, 20)$, $\theta_{ij} \sim U(-10, 10)$ (for $i \neq j$ and $\{i, j\}$ not equal to $\{2, 1\}$ or $\{3, 1\}$).

```
set.seed(101)
m <- matrix(0, 4, 4)
diag(m) <- runif(4, max=20)
m[lower.tri(m)] <- runif(6, min=-10, max=10)
m[2,1] <- m[3,1] <- 0
S <- m %*% t(m)
S
         [,1]       [,2]       [,3]       [,4]
[1,] 55.41265  0.0000000   0.000000  12.634888
[2,]  0.00000  0.7682458  -2.919309   2.138861
[3,]  0.00000 -2.9193087 212.553839   4.881917
[4,] 12.63489  2.1388607   4.881917 182.698471

eigen(S)$values

[1] 213.387898 183.174454  54.170033   0.700823
```

Share Cite Edit Follow Flag

edited 21 hours ago

answered yesterday

Ben Bolker
32.9k 2 90 121

Is positive diagonal in L even required for $\Sigma = LL^T$ to be PD? – Firebug yesterday

I get 0.02% non-PD matrices when not restricting diagonals vs 0.05% non-PD matrices when restricting the diagonal to be positive (in 10x10 matrices, to exacerbate numerical errors). – Firebug yesterday

1 See edits ..... – Ben Bolker yesterday

Great! I always searched for that reference as well, was needing it for a project actually, so thank you! – Firebug yesterday

---

0

Every $d \times d$ symmetric positive (semi)definite matrix Σ can be factored as

$$\Sigma = \Lambda' \, Q' \, Q \, \Lambda$$

where Q is an orthonormal matrix and Λ is a diagonal matrix with non-negative(positive) entries $\lambda_1, \ldots, \lambda_d$. ($\Sigma$ is always the covariance matrix of *some* d-variate distribution and QQ' will be its correlation matrix; the λ_i are the standard deviations of the marginal distributions.)

Let's interpret this formula. The (i, j) entry $\Sigma_{i,j}$ is the dot product of columns i and j of Q, multiplied by $\lambda_i \lambda_j$. Thus, *the zero-constraints on Σ are orthogonality constraints on the dot products of the columns of Q.*

products of the columns of Q.

(Notice that all diagonal entries of a positive-definite matrix must be nonzero, so I assume the zero-constraints are all off the diagonal. I also extend any constraint on the (i,j) entry to a constraint on the (j,i) entry, to assure symmetry of the result.)

One (completely general) way to impose such constraints is to generate the columns of Q sequentially. Use any method you please to create a $d\times d$ matrix of initial values. At step $i=1,2,\ldots,d,$ alter column i regressing it on all the columns $1,2,\ldots,i-1$ of Q that need to be orthogonal to it and retaining the residuals. Normalize those results so their dot product (sum of squares) is unity. That is column i of Q.

Having created an instance of Q, randomly generate the diagonal of Λ any way you please (as discussed in the closely related answer at https://stats.stackexchange.com/a/215647/919).

The following `R` function `rQ` uses *iid* standard Normal variates for the initial values by default. I have tested it extensively with dimensions $d=1$ through $200,$ checking systematically that the intended constraints hold. I also tested it with Poisson(0.1) variates, which--because they are likely to be zero--generate highly problematic initial solutions.

The principal input to `rQ` is a logical matrix indicating where the zero-constraints are to be applied. Here is an example with the constraints specified in the question.

```
set.seed(17)
Q <- matrix(c(FALSE, TRUE, TRUE, FALSE,
              TRUE, FALSE, FALSE, FALSE,
              TRUE, FALSE, FALSE, FALSE,
              FALSE, FALSE, FALSE, FALSE), 4)
Lambda <- rexp(4)
zapsmall(rQ(Q, Lambda))
```

```
         [,1]      [,2]      [,3]      [,4]
[1,] 2.646156  0.000000  0.000000  2.249189
[2,] 0.000000  0.079933  0.014089 -0.360013
[3,] 0.000000  0.014089  0.006021 -0.055590
[4,] 2.249189 -0.360013 -0.055590  4.167296
```

As a convenience, you may pass the diagonal of Λ as the second argument to `rQ`. Its third argument, `f`, must be a random number generator (or any other function for which `f(n)` returns a numeric vector of length `n`).

```
rQ <- function(Q, Lambda, f=rnorm) {
  normalize <- function(x) {
    v <- zapsmall(c(1, sqrt(sum(x * x))))[2]
    if (v == 0) v <- 1
    x / v
  }
  Q <- Q | t(Q)                    # Force symmetry by applying all constraints
  d <- nrow(Q)
  if (missing(Lambda)) Lambda <- rep(1, d)
  R <- matrix(f(d^2), d, d)        # An array of column vectors
  for (i in seq_len(d)) {
    j <- which(Q[seq_len(i-1), i]) # Indices of the preceding orthogonal vectors
    R[, i] <- normalize(residuals(.lm.fit(R[, j, drop=FALSE], R[, i])))
  }
```

```
  }
  R <- R %*% diag(Lambda)
  crossprod(R)
}
```

Share Cite Edit Follow Flag

answered 25 secs ago

whuber ♦
270k 53 606 1060

-1

Want to improve this post? Provide detailed answers to this question, including citations and an explanation of why your answer is correct. Answers without enough detail may be edited or deleted.

First, generate the random symmetric matrix. Second, apply ledoit wolf regularization to make it SPD.

Share Cite Edit Follow Flag

answered yesterday

Aksakal
52.2k 5 81 166

Does ledoit wolf regularization keeps sparsity? – Tan yesterday