

SimJoint: simulate joint distribution given marginals and correlation matrix

Charlie Wusuo Liu

June 9, 2019

Abstract

This R package simulates joint distribution given nonparametric marginals and their covariance structure characterized by a Pearson or Spearman correlation matrix. The simulator engages the problem from a purely computational perspective. It assumes no statistical models such as copulas or parametric distributions, and can approximate the target correlations regardless of theoretical feasibility. The algorithm integrates and advances the Iman-Conover (1982) approach [1] and the Ruscio-Kaczetow (2008) iteration [2]. Additionally, a simple heuristic algorithm is designed to optimize the joint distribution in the post-simulation stage. This heuristic demonstrated not only strong capability of cost reduction, but also good potential of achieving the same level of precision of approximation without the enhanced Iman-Conover-Ruscio-Kaczetow.

Contents

1 Iman-Conover review	1
1.1 Enhancement	2
2 Ruscio-Kaczetow iteration	2
2.1 Enhancement	2
3 Post-simulation optimization	4
4 Implementation	4

1 Iman-Conover review

This section assumes readers are familiar with the Iman-Conover procedure. Let X be an $N \times K$ matrix where each column contains samples from a marginal distribution. Let Σ be a $K \times K$ Pearson correlation matrix, and let S be an $N \times K$ uncorrelated (between columns) noise matrix. Reordering elements in each columns of X in the following way can let X have a column-wise correlation matrix close to Σ :

1. Cholesky-decompose $\Sigma = U^T U$ where U is the upper-triangle.
2. Let $Y = SU$. Y will be a $N \times K$ matrix whose column-wise correlations equal Σ .

3. Reorder elements in the k th column of X such that it perfectly rank-correlates the k th column of Y , $k = 0, 1, \dots, K - 1$. Denote the reordered X by $X^{(1)}$.

Since $X^{(1)}$ and Y have identical Spearman rank-correlation matrices, the two should have similar Pearson correlation matrices — this is merely an assumption due to the close relationship between Pearson and Spearman correlations.

1.1 Enhancement

An eigen-decomposition can substitute the Cholesky decomposition if Σ is not positive-definite: let $\Sigma = (\Lambda^{\frac{1}{2}}Q)^{\top}(\Lambda^{\frac{1}{2}}Q)$ and set $Y = S(\Lambda^{\frac{1}{2}}Q)$. In fact, any decomposition that results in a matrix left-multiplied by its transpose, i.e. $\Sigma = A^{\top}A$, guarantees $Y = SA$ has correlation matrix Σ . Eigen-decomposition is chosen due to its relative cheap computational cost over symmetric matrices.

2 Ruscio-Kaczetow iteration

There are two error sources that prevent $X^{(1)}$ having a correlation matrix exactly equal to Σ : (i) columns of the noise matrix S can hardly be uncorrelated (limited by random number generator), (ii) $X^{(1)}$ and Y having identical rank-correlation matrices does not imply they have the same Pearson correlation matrix.

Let $\Sigma^{(1)}$ be the Pearson correlation matrix of $X^{(1)}$. Let $e(\cdot)$ be a cost function and calculate $e(\Sigma - \Sigma^{(1)})$. Feeding a different correlation matrix Σ_{target} to Steps 1 - 3 in Section 1 may end up with a lower cost $e(\Sigma - \Sigma^{(1)})$. Ruscio and Kaczetow (2008) created a program that iteratively adjusts Σ_{target} according to $\Delta = \Sigma - \Sigma^{(1)}$ until $e(\Delta)$ converges.

2.1 Enhancement

For some reason, Ruscio and Kaczetow embedded a quite expensive factor analysis to obtain the matrix that is equivalent to Y in Iman-Conover. My guess is that they might be concerned by the limitation of Cholesky decomposition to positive definite matrix, or they simply intended to threshold the dimensionality with principal components — the number of factors for reproducing Σ are less than K . Such a dimension reduction however seems no impact on the end result, since eventually, we need K correlated vectors spanned from those factors to rank-order the marginals.

Another drawback of Ruscio and Kaczetow’s iteration is that the correlation adjustments, analogous to the step sizes in gradient descent, are deterministic and identical for every entry of the correlation matrix. Simulations have shown random step sizes can largely lower the converged $e(\Delta)$. This package uses a uniform kernel to generate the stochastic steps. Further investigation on the choice of kernels might lead to better convergence.

Ruscio and Kaczetow’s codes published with their paper were erroneous. Instructions for correction are made in the example section for `SJpearson()` in the package manual.

The full improved algorithm is depicted in Algorithm 1.

Algorithm 1 Simulate joint distribution given marginals and correlation matrix

Input: (i) an $N \times K$ matrix X of samples from K marginal distributions; (ii) a $K \times K$ correlation matrix Σ ; (iii) maximal iteration i_{\max} ; (iv) range of the step size for correcting correlations $[l, u]$ — self-explanatory in the algorithm, default $[0, 1]$; (v) convergence tail size h — self-explanatory in the algorithm.

Optional input: an $N \times K$ noise matrix S where columns are uncorrelated (conceptually). One can populate this matrix with uniform random numbers.

Operator and function definitions:

\odot : element-wise multiplication for vectors or matrices.

$\sigma(X)$: export the column-wise Pearson correlation matrix of X .

$\phi(\mathbf{x})$: given a vector \mathbf{x} of size $\frac{1}{2}(K-1)K$, export a $K \times K$ symmetric matrix where the diagonal equals 1 and the lower triangle entries equal \mathbf{x} .

$e(\Delta)$: given a matrix Δ , export a scalar such as the sum of all squared elements in Δ .

$\pi(X, Y)$: given $N \times K$ matrices X and Y , reorder elements in the k th column of X such that it perfectly rank-correlates the k th column of Y , $k = 0, 1, \dots, K-1$. Export the reordered X .

```
1: If  $S$  is not given, copy  $X$  to  $S$  and permute each column of  $S$  at random.
2:  $\epsilon^{\text{optimal}} \leftarrow \infty$ ;  $i \leftarrow 0$ ;  $\Sigma_{\text{target}}^{(i)} \leftarrow \Sigma$ ;
3: Fill an array  $\{\epsilon_0, \dots, \epsilon_{h-1}\}$  with arbitrary unequal values.
4: Allocate vector  $\mathbf{r}$  of size  $\frac{1}{2}(K-1)K$  for storing stochastic step ratio.
5: while  $i < i_{\max}$  do
6:   Cholesky (primary) or eigen (secondary) decomposition:  $\Sigma_{\text{target}}^{(i)} = A^{\top} A$ .
7:    $Y \leftarrow SA$ .
8:    $X^{(i)} \leftarrow \pi(X, Y)$ .
9:    $\Sigma^{(i)} \leftarrow \sigma(X^{(i)})$ .
10:   $\Delta \leftarrow \Sigma - \Sigma^{(i)}$ .
11:   $\epsilon \leftarrow e(\Delta)$ .
12:   $\{\epsilon_0, \dots, \epsilon_{h-1}\} \leftarrow \{\epsilon_1, \dots, \epsilon_{h-2}, \epsilon\}$ .
13:  if  $\epsilon_0 = \epsilon_1 = \dots = \epsilon_{h-1}$  then
14:    break  $\triangleright$  Algorithm converged.
15:  end if
16:  loop
17:    if  $\epsilon < \epsilon^{\text{optimal}}$  then
18:       $\Sigma^{\text{optimal}} \leftarrow \Sigma^{(i)}$ .
19:       $\epsilon^{\text{optimal}} \leftarrow \epsilon$ .
20:       $\Sigma_{\text{target}}^{\text{base}} \leftarrow \Sigma_{\text{target}}^{(i)}$ .
21:       $\mathbf{r} \leftarrow \mathbf{1}$ .
22:    else
23:      Populate a vector  $\mathbf{x}$  of size  $\frac{1}{2}(K-1)K$  with random uniforms in  $[l, u]$ .
24:       $\mathbf{r} \leftarrow \mathbf{r} \odot \mathbf{x}$ 
25:    end if
26:     $i \leftarrow i + 1$ .
27:     $\Sigma_{\text{target}}^{(i)} \leftarrow \min\left(\mathbf{1}, \max\left(-\mathbf{1}, \Sigma_{\text{target}}^{\text{base}} + \phi(\mathbf{r}) \odot (\Sigma - \Sigma^{\text{optimal}})\right)\right)$ .
28:    if  $\Sigma_{\text{target}}^{(i)}$  is positive semi-definite then  $\triangleright$  merged with Step 6.
29:      break.
30:    end if
31:  end loop
32: end while
33: return  $X^{(i)}$ .
```

3 Post-simulation optimization

We propose a simple stochastic optimization procedure to further reduce the final cost ϵ in Algorithm 1:

1. From the output correlation matrix $\Sigma^{(i)}$, identify the κ entries that have the κ largest absolute difference against the target correlation matrix Σ .
2. Select one of the κ entries with a probability. Identify which two columns in $X^{(i)}$ result in this correlation.
 - We set the probability as an increasing function of the absolute difference.
3. According to certain criteria (below), from the two columns in Step 2, sample two rows so we have four elements.
 - (a) Let $X^{(i)}[s, u]$, $X^{(i)}[t, u]$, $X^{(i)}[s, v]$, $X^{(i)}[t, v]$ represent the four elements. Let ρ^* be the correlation between columns $X^{(i)}[:, u]$ and $X^{(i)}[:, v]$.
 - (b) We keep sampling s, t until

$$(X^{(i)}[s, u] - X^{(i)}[t, u]) \cdot (X^{(i)}[s, v] - X^{(i)}[t, v]) < 0 \text{ if } \rho^* \text{ is below target,} \quad (1)$$

or,

$$(X^{(i)}[s, u] - X^{(i)}[t, u]) \cdot (X^{(i)}[s, v] - X^{(i)}[t, v]) > 0 \text{ if } \rho^* \text{ is above target.} \quad (2)$$

4. Let w be u or v at random. Swap $X^{(i)}[s, w]$ and $X^{(i)}[t, w]$ if this would reduce the final cost ϵ .
5. Repeat Steps 1 to 4 until the last consecutive h iterations failed to lower the cost.

In various experiments, feeding the input of Algorithm 1 to this procedure achieves the same level of precision of approximation.

4 Implementation

The package is implemented in C++ and is carefully programmed for computing speed. Memory consumption is dominated by three $N \times K$ matrices: the input X , the input S or a permuted X , and Y . The input X is normalized (column shifted and scaled) such that its Gramian matrix equals the correlation matrix. The permuted X or S is in row-major for promoting cache locality while left-multiplying the factorization A . Matrix multiplications are hand-optimized to exploit matrix symmetries and triangularities. All major steps except for Cholesky and eigen decompositions, which employed C++ library *Armadillo*, are crafted for multithreading.

Imposing Spearman correlations is equivalent to imposing the same Pearson correlations on ranks. If a rank correlation matrix is given, the algorithm populates three $N \times K$ single-precision matrices: X , a permuted X , and Y with normalized ranks. Therefore the memory usage stays below three $N \times K$ double-precision matrices. Single-precision floats are sufficiently accurate for operations on ranks due to their uniformity.

For the post-simulation optimization, the key to high computing speed is to not recompute, but rather to regionally update, the correlation matrix and the cost function in each iteration. Programming details are presented in C++ source comments.

The package imports Rcpp, RcppArmadillo, RcppParallel for bridging C++ and R.

References

- [1] R. L. Iman and W. J. Conover, “A distribution-free approach to inducing rank correlation among input variables,” *Communications in Statistics - Simulation and Computation*, 1982. [Online]. Available: <https://doi.org/10.1080/03610918208812265>
- [2] J. Ruscio and W. Kacetow, “Simulating multivariate nonnormal data using an iterative algorithm,” *Multivariate Behavioral Research*, 2008. [Online]. Available: <https://doi.org/10.1080/00273170802285693>