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 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 is an integration and advancement of the Iman-Conover (1982) approach [1] and the Ruscio-Kaczetow (2008) iteration [2]. Additionally, a simple heuristic algorithm initially designed to optimize the simulated joint distribution has demonstrated not only the capability of significant error reduction, but also the potential of achieving the same level of precision of approximation without Iman-Conover-Ruscio-Kaczetow as a primer.

Contents

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

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

Since $X^{(1)}$ and Y have identical Spearman rank-correlation 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 postive-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 are not perfectly uncorrelated (limited by random number generator), (ii) $X^{(1)}$ and Y being perfectly rank-correlated 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 equivalent to Y in Iman-Conover. They might be concerned by Cholesky decomposition's limitation to positive definite matrix, or simply intended to threshold the dimensionality with principal components — the number of factors for reproducing Σ are less than K. Such a dimension reduction in the middle however seems no impact on the end result, since we still 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 adjustment, analogous to the step size in gradient descent, is deterministic and identical for every entry in the correlation matrix. Experiments show stochastic adjustments can largely reduce the converged $e(\Delta)$. Currently a uniform kernel is used to generate the stochastic steps. Further investigation on the choice of kernels might lead to better convergence.

Ruscio and Kaczetow's code published with their paper was erroneous. The issue is further addressed in benchmark tests for `SJpearson()`.

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 kth column of X such that it perfectly rank-correlates the kth column of Y, $k = 0, 1, \ldots, K-1$. Export the reordered X.

```
1: If $S$ is not given, copy $X$ to $S$ and permute each column of $S$ at random.
2: $\epsilon^{\text{optimal}} \leftarrow \infty$; $i \leftarrow 0$; $\Sigma^{(i)}_{\text{target}} \leftarrow \Sigma$;
3: Fill an array $\{\epsilon_0, \ldots, \epsilon_{h-1}\}$ 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: $\Sigma^{(i)}_{\text{target}} = A^\top A$ .
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:  $\{\epsilon_0, \ldots, \epsilon_{h-1}\} \leftarrow \{\epsilon_1, \ldots, \epsilon_{h-2}, \epsilon\}$ .
13:  if $\epsilon_0 = \epsilon_1 = \ldots = \epsilon_{h-1}$ then
14:    break ▷ 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:      $\Sigma^{\text{base}}_{\text{target}} \leftarrow \Sigma^{(i)}_{\text{target}}$ .
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:    $\Sigma^{(i)}_{\text{target}} \leftarrow \min\Big(\mathbf{1},\, \max\big(-\mathbf{1},\, \Sigma^{\text{base}}_{\text{target}} + \phi(\mathbf{r}) \odot (\Sigma - \Sigma^{\text{optimal}})\big)\Big)$ .
28:    if $\Sigma^{(i)}_{\text{target}}$ is positive semi-definite then ▷ 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 errors in the approximated correlation matrix from Algorithm 1. The main steps are as follows:

1. From the current correlation matrix $\Sigma^{(i)}$, identify κ entries having the largest absolute errors (against target correlation matrix Σ).
2. Select one of the κ entries with certain probability. Identify the two associated columns in $X^{(i)}$.
3. According to certain criteria, from the columns sample four elements such that they are in only two different rows.
4. Swap the two elements in one of the columns if such a move would reduce the overall error, i.e. the cost function in Algorithm 1, of $\Sigma^{(i)}$.
5. Repeat Steps 1 to 4 until the last consecutive h iterations failed to lower the cost.

In Step 2, probabilities should be an increasing function of the entries' absolute errors.

In Step 3, let $X^{(i)}[s,\,u]$, $X^{(i)}[t,\,u]$, $X^{(i)}[s,\,v]$, $X^{(i)}[t,\,v]$ denote the four elements sampled at random. If the correlation between columns $X^{(i)}[,u]$, $X^{(i)}[,v]$ is below target, we keep sampling until

$$(X^{(i)}[s,\,u]-X^{(i)}[t,\,u])\cdot(X^{(i)}[s,\,v]-X^{(i)}[t,\,v])<0\;. \tag{1}$$

If the correlation between columns $X^{(i)}[,u]$, $X^{(i)}[,v]$ is above target, we keep sampling until

$$(X^{(i)}[s,\,u]-X^{(i)}[t,\,u])\cdot(X^{(i)}[s,\,v]-X^{(i)}[t,\,v])>0\;. \tag{2}$$

In various experiments, feeding the raw input to this algorithm achieves the same level of precision of approximation as the enhanced Iman-Conover-Ruscio-Kaczetow. A little heartbreaking yet humored to see the elegant and sophisticated outperformed by the brute and simple.

4 Implementation

The package is implemented in C++ and is carefully programmed for high 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 handmade to exploit matrix symmetries and triangularities . All major operations except for Cholesky and eigen decompositions, which utilize 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, thus 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 post-simulation optimization, the key to high computing speed is to avoid recomputing the entire correlation matrix and cost function in each iteration. Complexity of the update of correlation matrix and cost due to swapping elements in Step 3, Section 3 can be easily reduced. Programming details are presented in the C++ source code 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. Kaczetow, "Simulating multivariate nonnormal data using an iterative algorithm," *Multivariate Behavioral Research*, 2008. [Online]. Available: https://doi.org/10.1080/00273170802285693