Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *doi: 10.18637/jss.v000.i00*

Simulating High-Dimensional Multivariate Data using the Bigsimr Package

Abstract

It is critical to realistically simulate data when employing Monte Carlo techniques and evaluating statistical methodology. Measurements are often correlated and high dimensional in this era of big data, such as data obtained through high-throughput biomedical experiments. Due to computational complexity and a lack of user-friendly software available to simulate these massive multivariate constructions, researchers resort to simulation designs that posit independence or perform arbitrary data transformations. This article introduces the \pkg{Bigsimr} R package. This high-level package provides a flexible and scalable procedure to simulate high-dimensional random vectors with arbitrary marginal distributions and known Pearson, Spearman, or Kendall correlation matrix. \pkg{Bigsimr} contains additional high-performance features, including multi-core algorithms to simulate random vectors, estimate correlation, and compute the nearest correlation matrix. Monte Carlo studies quantify the accuracy and scalability of our approach, up to $d = 10,000$. Finally, we demonstrate example applications enabled via \pkg{Bigsimr} by applying these methods to a motivating dataset — RNA-sequencing data obtained from breast cancer tumor samples.

Keywords: multivariate simulation, high-dimensional data, nonparametric correlation, Gaussian copula, RNA-sequencing data, breast cancer.

1. Introduction

Massive high-dimensional (HD) data sets are now commonplace in many areas of scientific inquiry. As new methods are developed for data, a fundamental challenge lies in designing and conducting simulation studies to assess the operating characteristics of analyzing such proposed methodology — such as false positive rates, statistical power, interval coverage, and robustness — often with comparison to existing methods. Further, efficient simulation empowers statistical computing strategies, such as the parametric bootstrap (Chernick 2008) to simulate from a hypothesized null model, providing inference in analytically challenging settings. Such Monte Carlo (MC) techniques become difficult for high-dimensional data using existing algorithms and tools. This is particularly true when simulating massive multivariate, non-normal distributions, arising naturally in many fields of study.

As many have noted, it can be vexing to simulate dependent, non-normal/discrete data, even for low dimensional settings (Madsen and Birkes 2013; Xiao and Zhou 2019a). For continuous non-normal multivariate data, the well-known NORmal To Anything (NORTA) algorithm (Cario and Nelson 1997) and other copula approaches (Nelsen 2007) are well-studied, with flexible, robust software available (Yan 2007; Chen 2001). Yet these approaches do not scale in a timely fashion to high-dimensional problems (Li, Schissler, Wu, Barford, Harris, Fredrick C., and Harris 2019). For discrete data, early simulation strategies had major flaws, such as failing to obtain the full range of possible correlations (e.g., admitting only positive correlations: see Park, Park, and Shin (1996)). While more recent approaches (Madsen and Birkes 2013; Xiao 2017; Barbiero and Ferrari 2017) have largely remedied this issue for low-dimensional problems, the existing tools are not designed to scale to high dimensions.

Another central issue lies in characterizing dependence between components in the high-dimensional random vector. The choice of correlation in practice usually relates to the eventual analytic goal and distributional assumptions of the data (e.g., non-normal, discrete, infinite support, etc). For normal data, the Pearson product-moment correlation describes the dependency perfectly. As we will see, however, simulating arbitrary random vectors that match a target Pearson correlation matrix is computationally intense (Chen 2001; Xiao 2017). On the other hand, an analyst might consider use of nonparametric correlation measures to better characterize monotone, non-linear dependence, such as Spearman's ρ and Kendall's τ. Throughout, we focus on matching these nonparametric dependence measures, as our aim lies in modeling non-normal data and these rank-based measures possess invariance properties favorable in our proposed methodology. We do, however, provide Pearson matching with some caveats.

With all this in mind, we present a scalable, flexible multivariate simulation algorithm. The crux of the method lies in the construction of a Gaussian copula in the spirit of the NORTA procedure. Further, we introduce the `bigsimr` R package that provides high-performance software implementing our algorithm. The algorithm design leverages useful properties of nonparametric correlation measures, namely invariance under monotone transformation and well-known closed-form relationships between dependence measures for the multivariate normal (MVN) distribution.

Our study proceeds by providing background information, including a description of a motivating example application: RNA-sequencing (RNA-seq) breast cancer data. Then we describe and justify our simulation methodology and related algorithms. Next, we detail an illustrative low-dimensional example of basic use of the `bigsimr` R package. Then we proceed with Monte Carlo studies under various bivariate distributional assumptions to assess accuracy. We conclude the Monte Carlo evaluations by summarizing the computation time for increasingly higher dimensional vectors. After the MC evaluations, we simulate random vectors motivated by our RNA-seq example, evaluate the accuracy, and provide example statistical computing tasks, namely MC estimation of joint probabilities and evaluating HD correlation estimation efficiency. Finally, we discuss the method's utility, limitations, and future directions.

2. Background

The **bigsimr** R package presented here provides multiple algorithms that operate with high-dimensional multivariate data; however, all these algorithms were originally designed to support a single task: to generate random vectors drawn from multivariate probability distributions with given marginal distributions and dependency metrics. Specifically, our goal is to efficiently simulate a large number, B, of HD random vectors $\mathbf{Y} = (Y_1, \ldots, Y_d)^\top$ with *correlated* components and heterogeneous marginal distributions, described via cumulative distribution functions (CDFs) F_i.

When designing this methodology, we developed the following properties to guide our effort. We divide the properties into two categories: (1) basic properties (BP) and "scalability" properties (SP). The BPs are adapted from an existing criteria due to Nikoloulopoulos (2013). A suitable simulation strategy should possess the following properties:

- BP1: A wide range of dependences, allowing both positive and negative values, and, ideally, admitting the full range of possible values.
- BP2: Flexible dependence, meaning that the number of bivariate marginals can be equal to the number of dependence parameters.
- BP3: Flexible marginal modeling, generating heterogeneous data — possibly from differing probability families.

Moreover, the simulation method must *scale* to high dimensions:

- SP1: Procedure must scale to high dimensions with practical compute times.
- SP2: Procedure must scale to high dimensions while maintaining accuracy.

2.1. Motivating example: RNA-seq data

Simulating high-dimensional, non-normal, correlated data motivates this work — in pursuit of modeling RNA-sequencing (RNA-seq) data (Wang, Gerstein, and Snyder 2009; Conesa, Madrigal, Tarazona, Gomez-Cabrero, Cervera, McPherson, Szcześniak, Gaffney, Elo, Zhang, and Mortazavi 2016) derived from breast cancer patients. The RNA-seq data-generating process involves counting how often a particular form of human messenger RNA (mRNA) is expressed in a biological sample. RNA-seq platforms typically quantify the entire transcriptome in one experimental run, resulting in high-dimensional data. For human-derived samples, this results in count data corresponding to over 20,000 genes (protein-coding genomic regions) or even over 77,000 isoforms when alternatively-spliced mRNA are counted (Schissler, Aberasturi, Kenost, and Lussier 2019). Importantly, due to inherent biological processes, gene expression data exhibit correlation (co-expression) across genes (Efron 2007; Schissler, Piegorsch, and Lussier 2018).

We illustrate our methodology using a well-studied Breast Invasive Carcinoma (BRCA) data set housed in The Cancer Genome Atlas (TCGA; see Acknowledgments). For ease of modeling and simplicity of exposition, we only consider high expressing genes. In turn, we begin by filtering to retain the top 1000 of the highest-expressing genes (in terms of median expression) of the over 20,000 gene measurements from $N = 878$ patients' tumor samples. This gives a great number of pairwise dependencies among the marginals (specifically, 4.995×10^5 correlation parameters). Table 1 displays RNA-seq counts for three selected high-expressing genes for the first five patients' breast tumor samples. To help visualize the bivariate relationships for these three selected genes across all patients, Figure 1 displays the marginal distributions and estimated Spearman's correlations.

```
R> set.seed(2020-02-25)
R> num_genes    <- 3
R> num_patients <- 5
R> gene_sample  <- sample(example_genes[1:1000], num_genes)
R>
R> cap <- paste0("mRNA expression for three selected high-expressing genes, ",
+               paste(gene_sample, collapse = ", "),
+               ", for the first five patients in the TCGA BRCA data set.")
R>
R> small_brca <- example_brca %>%
+   select(all_of(gene_sample)) %>%
+   head(n = num_patients) %>%
+   as_tibble(rownames = "Patient ID") %>%
+   mutate(`Patient ID` = str_sub(`Patient ID`, end = 12))
R>
R> small_brca %>%
+   kable(format = "latex", booktabs = TRUE, caption = cap)

R> ggpairs(
+   data = example_brca[, gene_sample],
+   upper = list(continuous = wrap('cor', method = "spearman"))
```

Table 1: mRNA expression for three selected high-expressing genes, STAU1, FKBP1A, NME2, for the first five patients in the TCGA BRCA data set.

Patient ID	STAU1	FKBP1A	NME2
TCGA-A1-A0SB	10440	11354	17655
TCGA-A1-A0SD	21523	20221	14653
TCGA-A1-A0SE	21733	22937	35251
TCGA-A1-A0SF	11866	19650	16551
TCGA-A1-A0SG	12486	12089	10434

```
+ ) +

+   theme_bw() +

+   theme(axis.text.x = element_text(angle = -90, vjust = 0.5))
```

```
Warning in cor.test.default(x, y, method = method, use = use): Cannot compute

exact p-value with ties


Warning in cor.test.default(x, y, method = method, use = use): Cannot compute

exact p-value with ties


Warning in cor.test.default(x, y, method = method, use = use): Cannot compute

exact p-value with ties
```

2.2. Measures of dependency

In multivariate analysis, an analyst must select a metric to quantify dependency. The most widely-known is the Pearson (product-moment) correlation coefficient that describes the linear association between two random variables X and Y, and, it is given by

$$\rho_P(X, Y) = \frac{E(XY) - E(X)E(Y)}{[\mathrm{Var}(X)\mathrm{Var}(Y)]^{1/2}}. \tag{1}$$

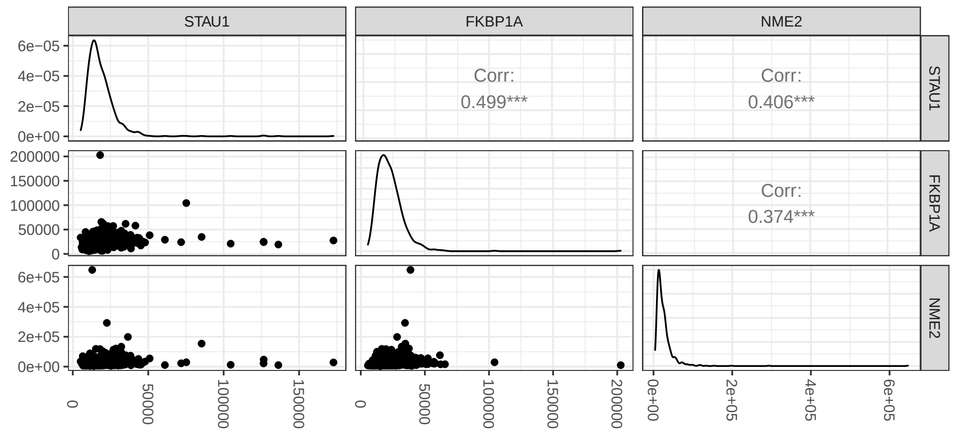

Figure 1: Marginal scatterplots, densities, and estimated pairwise Spearman's correlations for three example genes from Table 1. The data possess outliers, heavy-right tails, are discrete, and have non-trivial intergene correlations. Modeling these data motivate our simulation methodology.

As Madsen and Birkes (2013) and Mari and Kotz (2001) discuss, for a bivariate normal (X, Y) random vector, the Pearson correlation completely describes the dependency between the components. For non-normal marginals with monotone correlation patterns, however, ρ_P suffers some drawbacks and may mislead or fail to capture important relationships (Mari and Kotz 2001). Alternatively in these settings, analysts often prefer rank-based correlation measures to describe the degree of monotonic association.

Two nonparametric, rank-based measures common in practice are Spearman's correlation (denoted ρ_S) and Kendall's τ. Spearman's ρ_S has an appealing correspondence as the Pearson correlation coefficient on *ranks* of the values, thereby captures nonlinear yet monotone relationships. Kendall's τ, on the other hand, is the difference in probabilities of concordant and discordant pairs of observations, (X_i, Y_i) and (X_j, Y_j), with concordance meaning that orderings have the same direction (e.g., if $X_i < X_j$, then $Y_i < Y_j$). Note that concordance is determined by the ranks of the values, not the values themselves.

Both τ and ρ_S are *invariant under monotone transformations* of the underlying random variates. As we will describe more fully in the Algorithms section, this property enables matching rank-based correlations with speed (SP1) and accuracy (SP2).

Correspondence among Pearson, Spearman, and Kendall correlations

There is no closed form, general correspondence among the rank-based measures and the Pearson correlation coefficient, as the marginal distributions F_i are intrinsic in their calculation. For *bivariate normal vectors*, however, the correspondence is well-known:

$$\rho_P = \sin\left(\tau \times \frac{\pi}{2}\right), \tag{2}$$

and similarly for Spearman's ρ (Kruskal 1958),

$$\rho_P = 2 \times \sin\left(\rho_S \times \frac{\pi}{6}\right). \tag{3}$$

Marginal-dependent bivariate correlation bounds

Given two marginal distributions, ρ_P is not free to vary over the entire range of possible correlations $[-1, 1]$. The so-called *Frechet-Hoeffding bounds* are well-studied (Nelsen 2007; Barbiero and Ferrari 2017). These constraints cannot be overcome through algorithm design. In general, the bounds are given by

$$\rho_P^{max} = \rho_P\left(F_1^{-1}(U), F_2^{-1}(U)\right), \quad \rho_P^{min} = \rho_P\left(F_1^{-1}(U), F_2^{-1}(1-U)\right) \tag{4}$$

where U is a uniform random variable on $(0, 1)$ and F_1^{-1}, F_2^{-1} are the inverse CDFs of X_1 and

X_2, respectively, definitely by (5) when the variables are discrete.

$$F_i^{-1} = \inf\{y : F_i(y) \geq u\}. \tag{5}$$

2.3. Gaussian copulas

There is a strong connection of our simulation strategy to Gaussian *copulas* (see Nelsen (2007) for a technical introduction). A copula is a distribution function on $[0,1]^d$ that describes a multivariate probability distribution with standard uniform marginals. This provides a powerful, natural way to characterize joint probability structures. Consequently, the study of copulas is an important and active area of statistical theory and practice.

For any random vector $\mathbf{X} = (X_1, \ldots, X_d)^\top$ with CDF F and marginal CDFs F_i there is a copula function $C(u_1, \ldots, u_d)$ satisfying

$$F(x_1, \ldots, x_d) = \mathbb{P}(X_1 \leq x_1, \ldots, X_d \leq x_d) = C(F_1(x_1), \ldots, F_d(x_d)), \tag{6}$$

$x_i \in \mathbb{R}, i = 1, \ldots, d.$

A Gaussian copula has marginal CDFs that are all standard normal, $F_i = \Phi, \forall\, i$. This representation corresponds to a multivariate normal (MVN) distribution with standard normal marginal distributions and covariance matrix $\mathbf{R_P}$. As the marginals are standardized to have unit variance, however, $\mathbf{R_P}$ is a Pearson correlation matrix. If $F_{\mathbf{R}}$ is the CDF of such a multivariate normal distribution, then the corresponding Gaussian copula $C_{\mathbf{R}}$ is defined through

$$F_{\mathbf{R}}(x_1, \ldots, x_d) = C_{\mathbf{R}}(\Phi(x_1), \ldots, \Phi(x_d)), \tag{7}$$

where $\Phi(\cdot)$ is the standard normal CDF. Note that the copula $C_{\mathbf{R}}$ is the familiar multivariate normal CDF of the random vector $(\Phi(X_1), \ldots, \Phi(X_d))$, where $(X_1, \ldots, X_d) \sim N_d(\mathbf{0}, \mathbf{R_P})$.

Sklar's Theorem (Sklar 1959; Úbeda-Flores and Fernández-Sánchez 2017) guarantees that given inverse CDFs F_i^{-1}s and a valid correlation matrix (within the Frechet bounds) a random vector can be obtained via transformations involving copula functions. For example, using Gaussian copulas, we can construct a random vector $\mathbf{Y} = (Y_1, \ldots, Y_d)^\top$ with $Y_i \sim F_i$, viz. $Y_i = F_i^{-1}(\Phi(X_i)), i = 1, \ldots, d$, where $(X_1, \ldots, X_d) \sim N_d(\mathbf{0}, \mathbf{R_P})$.

3. Algorithms

This section describes our methods involved in simulating a random vector $\mathbf{Y}$ with Y_i components for $i = 1, \ldots, d$. Each Y_i has a specified marginal CDF F_i and its inverse F_i^{-1}. To characterize dependency, every pair (Y_i, Y_j) has a given Pearson correlation ρ_P, Spearman correlation ρ_S, and/or Kendall's τ. The method can be described as a *high-performance Gaussian copula* (Equation (7)) providing a high-dimensional NORTA-inspired algorithm.

3.1. NORmal To Anything (NORTA)

The well-known NORTA algorithm (Cario and Nelson 1997) simulates a random vector $\mathbf{Y}$ with variance-covariance matrix $\Sigma_{\mathbf{Y}}$. Specifically, NORTA algorithm proceeds as:

1. Simulate a random vector $\mathbf{Z}$ with d *independent and identically distributed* (iid) standard normal components.

2. Determine the input matrix $\Sigma_{\mathbf{Z}}$ that corresponds with the specified output $\Sigma_{\mathbf{Y}}$.
3. Produce a Cholesky factor M of $\Sigma_{\mathbf{Z}}$ such that $MM' = \Sigma_{\mathbf{Z}}$.
4. Set X by $X \leftarrow MZ$.
5. Return Y where $Y_i \leftarrow F_i^{-1}[\Phi(X_i)],\ i = 1, ..., d$.

With modern parallel computing, steps 1, 3, 4, 5 are readily implemented as high-performance, multi-core and/or graphical-processing-unit (GPU) accelerated algorithms — providing fast scalability using readily-available hardware.

Matching specified Pearson correlation coefficients exactly (step 2 above), however, is computationally costly. In general, there is no closed-form correspondence between the components of the input $\Sigma_{\mathbf{Z}}$ and target $\Sigma_{\mathbf{Y}}$. Matching the correlations involves evaluating or approximating $\binom{d}{2}$ integrals of the form

$$\mathrm{E}\left[Y_i Y_j\right] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F_i^{-1}\left[\Phi(z_i)\right] F_j^{-1}\left[\Phi(z_j)\right] \phi(z_i, z_j, \rho_z) dz_i dz_j, \tag{8}$$

where $\phi(\cdot)$ is the joint probability distribution function of two correlated standard normal variables. For HD data, (nearly) exact evaluation may become too costly to enable practical simulation studies. For low-dimensional problems, however, methods and tools exist to match Pearson correlations precisely; see (Xiao 2017) and the publicly available `nortaRA` R package (Chen 2001). As described later, to enable HD Pearson matching we approximate these integrals.

NORTA in higher dimensions

Sklar's theorem provides a useful characterization of multivariate distributions through copulas. Yet the choice of copula-based simulation algorithm affects which joint distributions may be simulated. Even in low-dimensional spaces (e.g., $d = 3$), there exist valid multivariate distributions with *feasible* Pearson correlation matrices that NORTA cannot match exactly (Li and Hammond 1975). This occurs when the bivariate transformations are applied to find the input correlation matrix, yet when combined the resultant matrix is indefinite. These situations do occur, even using exact analytic calculations. Such problematic target correlation matrices are termed *NORTA defective*.

Ghosh and Henderson (2002) conducted a Monte Carlo study to estimate the probability of encountering NORTA defective matrices while increasing the dimension d. They found that for what is now considered low-to-moderate dimensions ($d \approx 20$), almost *all* feasible matrices are NORTA defective. This stems from the concentration of measure near the boundary of the space of all possible correlation matrices as dimension increases. Unfortunately, it is precisely near this boundary that NORTA defective matrices reside.

There is hope, however, as Ghosh and Henderson (2002) also showed that replacing an indefinite input correlation matrix with a close proxy will give approximate matching to the target — with adequate performance for moderate d. This provides evidence that our nearest positive definite (PD) augmented approach will maintain reasonable accuracy if our input matching scheme returns an indefinite matrix, at least for the rank-based matching scheme described above.

3.2. Random vector generator

We now describe our algorithm to generate random vectors, which mirrors the classical NORTA algorithm above with some modifications for rank-based dependency matching:

1. Mapping step
    - (i) Convert the target Spearman correlation matrix R_S to the corresponding MVN Pearson correlation R_X. Alternatively,

- (i') Convert the target Kendall τ matrix R_K to the corresponding MVN Pearson correlation R_X. Alternatively,
- (i' ') Convert the target Pearson correlation matrix to R_P to the corresponding, approximate MVN Pearson correlation R_X.

2. Check admissibility and, if needed, compute the nearest correlation matrix.

- (i) Check that R_X is a correlation matrix, a positive definite matrix with 1's along the diagonal.
- (ii) If R_X is a correlation matrix, the input matrix is *admissible* in this scheme. Otherwise
- (ii') Replace R_X with the nearest correlation matrix $\tilde{R}_X$, in the Frobenius norm.

3. Gaussian copula

- (i) Generate $\mathbf{X} = (X_1, \ldots, X_d) \sim N_d(\mathbf{0}, R_X)$.
- (ii) Transform $\mathbf{X}$ to $\mathbf{U} = (U_1, \ldots, U_d)$ viz. $U_i = \Phi(X_i)$, $i = 1, \ldots, d$.
- (iii) Return $\mathbf{Y} = (Y_1, \ldots, Y_d)$, where $Y_i = F_i^{-1}(U_i)$, $i = 1, \ldots, d$.

Step 1

The first two descriptions of the *Mapping step* employ the closed-form relationships between ρ_S and τ with ρ_P for bivariate normal random variables via Equations (2) and (3), respectively (implemented as `cor_covert`). Initializing our algorithm to match the nonparametric correlations by computing these equations for all pairs is computationally trivial.

For the computationally expensive process to match Pearson correlations, we approximate Equation (8) for all pairs of margins. To this end, we implement the approximation scheme introduced by Xiao and Zhou (2019b). Briefly, the many double integrals of the form in (8) are approximated by weighted sums of Hermite polynomials. Matching coefficients for pairs of continuous distributions is made tractable by this method, but for discrete distributions (especially discrete distributions with large support sets or infinite support), the problem is not so simple or efficient.

Our solution is to approximate discrete distributions by a continuous distribution. The question then becomes that of which distribution to use. We found that Generalized S-Distributions (Muino, Voit, and Sorribas 2006) solve this problem by approximating a wide range of unimodal distributions, both continuous and discrete. Using the GS-Distribution approximation of discrete distributions in Pearson matching scheme above yields favorable results.

Step 2

Once R_X has been determined, we check admissibility of the adjusted correlation via the steps described above. If R_X is not a valid correlation matrix, then we compute the nearest correlation matrix. Finding the nearest correlation matrix is a common statistical computing problem. The defacto default function in `R` is `Matrix::nearPD`, an alternating projection algorithm due to Higham (2002). As implemented the function fails to scale to HD. Instead, we provide the quadratically convergent algorithm based on the theory of strongly semi-smooth matrix functions (Qi and Sun (2006)). The nearest correlation matrix problem can be written down as the following convex optimization problem:

$$\begin{aligned} \min \quad & \frac{1}{2}\|G - X\|^2 \\ \text{s.t.} \quad & X_{ii} = 1, \quad i = 1, \ldots, n, \\ & X \in S_+^n \end{aligned}$$

For nonparametric correlation measures, our algorithm allows the generation of high-dimensional multivariate data with arbitrary marginal distributions with a broad class of admissible Spearman correlation matrices and Kendall τ matrices. The admissible classes consist of the matrices that map to a Pearson correlation matrix for a MVN. In particular, if we let X be MVN with d components and set

$$\Omega_P = \{R_P : R_P \text{ is a correlation matrix for } X\}$$

$$\Omega_K = \{R_K : R_K \text{ is a correlation matrix for } X\}$$

$$\Omega_S = \{R_S : R_S \text{ is a correlation matrix for } X\}$$

then there are 1-1 mappings between these sets. We conjecture informally that the sets of admissible R_S and R_K are not highly restrictive. In particular, R_P is approximately R_S for a MVN, suggesting that the admissible set Ω_S should be flexible as R_P can be any PD matrix with 1's along the diagonal. We provide methods to check whether a target R_S is an element of the *admissible set* Ω_S (and similarly for R_K).

There is an increasing probability of encountering a non-admissible correlation matrix as dimension increases. In our experience, the mapping step for large d almost always produces a R_X that is not a correlation matrix. In Section 4 we provide a basic description of how to quantify and control the approximation error. Further, the RNA-seq example in Section 6 provides an illustration of this in practice.

Step 3

The final step implements a NORTA-inspired, Gaussian copula approach to produce the desired margins. Steps 1 and 2 determine the MVN Pearson correlation values that will eventually match the target correlation. Then all that is required is a fast MVN simulator, a standard normal CDF, and well-defined quantile functions for marginals. The MVN is transformed to a copula (distribution with standard uniform margins) by applying the normal CDF $\phi(\cdot)$. The quantile functions F^{-1} are applied across the margins to return the desired random vector $\mathbf{Y}$.

4. The bigsimr R package

```
R> box::use(
+   patchwork[...],
+   dplyr[...],
+   ggplot2[...],
+   tidyr[drop_na],
+   bigsimr[bigsimr_setup, distributions_setup],
+   JuliaCall[julia_call]
+ )
R>
R> Sys.setenv(JULIA_NUM_THREADS = parallel::detectCores())
R> bs <- bigsimr_setup(pkg_check = FALSE)

Julia version 1.5.4 at location /home/alex/julia-1.5.4/bin will be used.

Loading setup script for JuliaCall...
```

```
Finish loading setup script for JuliaCall.
```

```
R> dist <- distributions_setup()
```

This section describes a low-dimensional (2D) random vector simulation workflow via the `bigsimr` R package (`https://github.com/SchisslerGroup/r-bigsimr`). The package `bigsimr` provides an interface to the native code written in Julia, registered as the `Bigsimr` Julia package (`https://github.com/adknudson/Bigsimr.jl`). In addition to the native Julia `Bigsimr` package and `R` interface `bigsimr`, we also provide a python interface `bigsimr` (`https://github.com/SchisslerGroup/python-bigsimr/`) that interfaces with the Julia `Bigsimr` package. The Julia package provides a high-performance implementation of our proposed random vector generation algorithm and associated functions (see Section 3).

The subsections below describe the basic use of `bigsimr` by stepping through an example workflow using the data set `airquality` that contains daily air quality measurements in New York, May to September 1973 (Chambers, Cleveland, Kleiner, and Tukey 1983). This workflow proceeds from setting up the computing environment, to data wrangling, estimation, simulation configuration, random vector generation, and, finally, result visualization.

4.1. Bivariate example description

We illustrate the use of `bigsimr` motivated by the New York air quality data set (`airquality`) included in the R `datasets` package. First, we load the `bigsimr` library and a few other convenient data science packages, including the syntactically-elegant `tidyverse` suite of `R` packages. The code chunk below prepares the computing environment:

```
R> library("tidyverse")
R> library("bigsimr")
```

```
R> # Activate multithreading in Julia
R> Sys.setenv(JULIA_NUM_THREADS = parallel::detectCores())
R> # Load the Bigsimr and Distributions Julia packages
R> bs <- bigsimr_setup()
R> dist <- distributions_setup()
```

For simplicity and to provide a minimal working example, we consider bivariate simulation of the two `airquality` variables `Temperature`, in degrees Fahrenheit, and `Ozone` level, in parts per billion.

```
R> df <- airquality %>% select(Temp, Ozone) %>% drop_na()
```

```
R> glimpse(df)
```

```
Rows: 116
Columns: 2
$ Temp  <int> 67, 72, 74, 62, 66, 65, 59, 61, 74, 69, 66, 68, 58, 64, 66, 57, ~
$ Ozone <int> 41, 36, 12, 18, 28, 23, 19, 8, 7, 16, 11, 14, 18, 14, 34, 6, 30,~
```

Figure 2 visualizes the bivariate relationship between Ozone and Temperature. We aim to simulate random two-component vectors mimicking this structure. The margins are not normally distributed; particularly the Ozone level exhibits a strong positive skew.

```
R> p0 <- ggplot(df, aes(Temp, Ozone)) +
+   geom_point(size = 1) +
+   theme(legend.position = "none") +
+   labs(x = "Temperature")
```

```
R>
R> pTemp <- ggplot(df, aes(Temp)) +
+   geom_density() +
+   theme(axis.title.x = element_blank(),
+         axis.title.y = element_blank(),
+         axis.text = element_blank(),
+         axis.ticks = element_blank())
R>
R> pOzone <- ggplot(df, aes(Ozone)) +
+     geom_density() +
+   theme(axis.title.y = element_blank(),
+         axis.title.x = element_blank(),
+         axis.ticks = element_blank(),
+         axis.text = element_blank()) +
+   coord_flip()
R>
R> pTemp + plot_spacer() + p0 + pOzone +
+   plot_layout(widths = c(3,1), heights = c(1, 3))
```

Next, we specify the marginal distributions and correlation coefficient (both type and magnitude). Here the analyst is free to be creative. For this example, we avoid goodness-of-fit considerations to determine the marginal distributions. But it seems sensible without domain knowledge to estimate these quantities from the data, and `bigsimr` contains fast functions

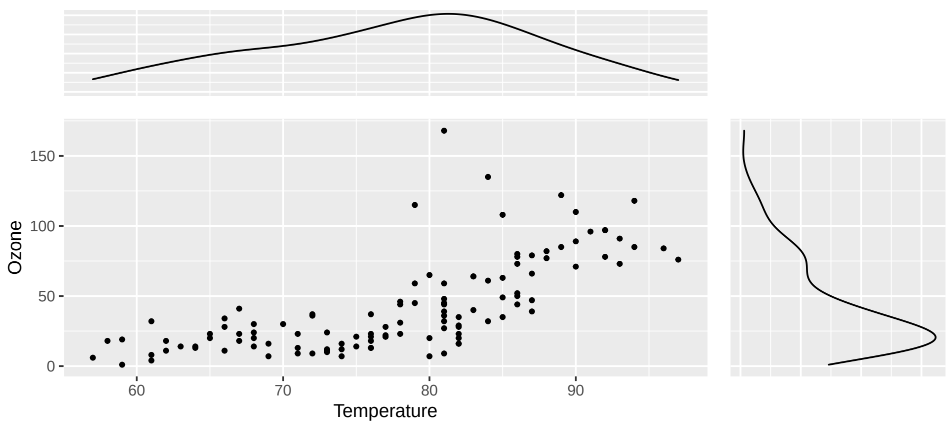

Figure 2: Bivariate scatterplot of Ozone vs. Temp with estimated marginal densities. We model the Ozone data as marginally log-normal and the Temperature data as normal.

designed for this task.

4.2. Specifying marginal distributions

Based on the estimated densities in Figure 2, we assume `Temp` is normally distributed and `Ozone` is log-normally distributed, as the latter values are positive and skewed. We use the well-known, unbiased estimators for the normal distribution's parameters and maximum likelihood estimators for the log-normal parameters:

```
R> df %>% select(Temp) %>%
+   summarise_all(.funs = c(mean = mean, sd = sd))
```

```
   mean    sd
1 77.87 9.485
```

```
R> mle_mean <- function(x) mean(log(x))
R> mle_sd <- function(x) mean( sqrt( (log(x) - mean(log(x)))^2 ) )
R> df %>%
+   select(Ozone) %>%
+   summarise_all(.funs = c(meanlog = mle_mean, sdlog = mle_sd))
```

```
  meanlog  sdlog
1   3.419 0.6967
```

Next, we configure the input marginals for later input into `rvec`. The marginal distributions are specified using `Julia`'s `Distributions` package and stored in a vector.

```
R> margins <- c(dist$Normal(mean(df$Temp), sd(df$Temp)),
+               dist$LogNormal(mle_mean(df$Ozone), mle_sd(df$Ozone)))
```

4.3. Specifying correlation

As mentioned, the user must decide how to describe correlation based on the particulars of the problem. For non-normal data and for improved simulation accuracy/scalability in our scheme, we advocate the use of Spearman's ρ correlation matrix R_S and Kendall's τ correlation matrix R_K. We also support Pearson correlation coefficient matching, while cautioning the user to check the performance for their parametric multivariate model (see Monte Carlo evaluations below for evaluation strategies and guidance). These estimation methods are classical approaches, not designed for high-dimensional correlation estimation (see the Conclusion and Discussion sections for more on this).

```
R> (R_S <- bs$cor(as.matrix(df), bs$Spearman))
```

```
      [,1]  [,2]
[1,] 1.000 0.774
[2,] 0.774 1.000
```

4.4. Checking target correlation matrix admissibility

First we use `cor_bounds` to ensure that the pairwise target correlation values are valid prior to the mapping step which constructs R_X for the MVN input into NORTA (see Section 3). `cor_bounds` estimates the pairwise lower and upper correlation bounds using the Generate, Sort, and Correlate algorithm of Demirtas and Hedeker (2011).

```
R> bounds <- bs$cor_bounds(margins)
R> bounds$lower

        [,1]    [,2]
[1,]  1.0000 -0.8821
[2,] -0.8821  1.0000

R> bounds$upper

       [,1]   [,2]
[1,] 1.0000 0.8812
[2,] 0.8812 1.0000
```

Since our single estimated Spearman correlation is within the theoretical bounds, the correlation is valid as input to `rvec`. But even if the 2-dimensional bounds are satisfied for each pair of margins, this does not guarantee the feasibility of a $d-$variate distribution (Barbiero and Ferrari 2017).

To provide higher dimensional feasibility/admissibility checking, we begin by mapping using `cor_convert` and check admissibility.

```
R> # Step 1. Mapping
R> (R_X <- bs$cor_convert(R_S, bs$Spearman, bs$Pearson))

       [,1]   [,2]
[1,] 1.0000 0.7886
[2,] 0.7886 1.0000

R> # Step 2. Check admissibility
R> bs$iscorrelation(R_X)

[1] TRUE
```

The bounds on the Pearson correlation coefficient R_P between these margins are restricted as seen below in our MC estimated correlation bounds:

```
R> bs$cor_bounds(margins[1], margins[2], bs$Pearson, n_samples = 1e6)

Julia Object of type NamedTuple{(:lower, :upper),Tuple{Float64,Float64}}.
(lower = -0.8814915390483397, upper = 0.8817369743444273)
```

Our MC estimate of the bounds slightly overestimates the theoretic bounds of $(-0.881, 0.881)$. An analytic derivation is presented as an Appendix.

4.5. Simulating random vectors

Finally, we arrive at the main function of `bigsimr`: `rvec`. We now simulate $B = 10,000$ random vectors from the assumed joint distribution of Ozone levels and Temp.

```
R> x <- bs$rvec(10000, R_X, margins)
R> df_sim <- as.data.frame(x)
R> colnames(df_sim) <- colnames(df)
```

Figure 3 plots the 10,000 simulated points.

```
R> p1 <- df_sim %>%
+   ggplot(aes(Temp, Ozone)) +
+   geom_density_2d_filled() +
+   theme(legend.position = "none") +
+   labs(x = "Simulated Temperature", y = "Simulated Ozone") +
+   scale_y_continuous(limits = c(0, max(df$Ozone))) +
+   scale_x_continuous(limits = range(df$Temp))
R>
R> p1Temp <- ggplot(df_sim, aes(Temp)) +
+   geom_density(alpha = 0.5, fill = "lightseagreen") +
+   theme(axis.title.x = element_blank(),
+         axis.title.y = element_blank(),
+         axis.text = element_blank(),
+         axis.ticks = element_blank())
R>
R> p1Ozone <- ggplot(df_sim, aes(Ozone)) +
+   geom_density(alpha = 0.5, fill = "lightseagreen") +
+   theme(axis.title.y = element_blank(),
+         axis.title.x = element_blank(),
```

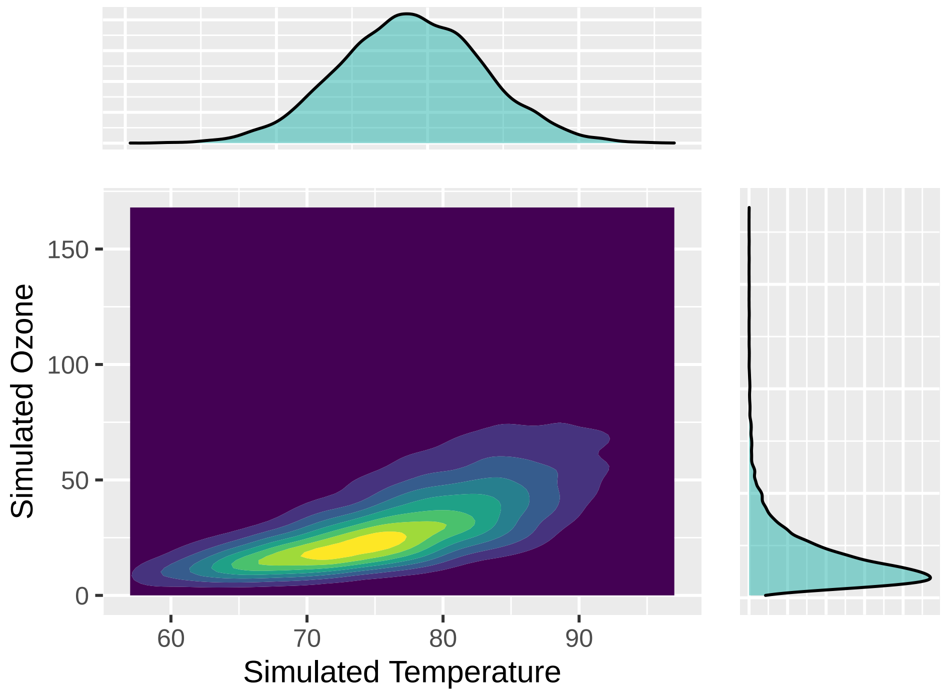

Figure 3: Contour plot and marginal densities for the simulated bivariate distribution of Air Quality Temperatures and Ozone levels. The simulated points mimic the observed data with respect to both the marginal characteristics and bivariate association.

```
+          axis.ticks = element_blank(),

+          axis.text = element_blank()) +

+    coord_flip()

R>

R> p1Temp + plot_spacer() + p1 + p1Ozone +

+    plot_layout(widths = c(3,1), heights = c(1, 3))


Warning: Removed 392 rows containing non-finite values (stat_density2d_filled).
```

5. Monte Carlo evaluations

Before applying our methodology to real data simulation, we conduct several Monte Carlo studies to investigate method performance. Since marginal parameter matching in our scheme is essentially a sequence of univariate inverse probability transforms, the challenging aspects are the accuracy of dependency matching and computational efficiency at high dimensions. To evaluate our methods in those respects, we design the following numerical experiments to first assess accuracy of matching dependency parameters in bivariate simulations and then time the procedure in increasingly large dimension d.

5.1. Bivariate experiments

We select bivariate simulation configurations to ultimately simulate our motivating discrete-valued RNA-seq example, and, so we proceed in increasing complexity, leading to the model in our motivating application in Section 6. We begin with empirically evaluating the dependency matching across all three supported correlations — Pearson's, Spearman's, and Kendall's — in identical, bivariate marginal configurations. For each pair of identical margins, we vary the target correlation across Ω, the set of possible admissible values for correlation type, to evaluate the simulation's ability to obtain the theoretic bounds. The simulations progress from bivariate normal, to bivariate gamma (non-normal yet continuous), and bivariate negative binomial (mimicking RNA-seq counts).

Table 5.1 lists our identical-marginal, bivariate simulation configurations. We increase the simulate replicates B to check that our results converge to the target correlations and gauge statistical efficiency. We select distributions beginning with a standard multivariate normal (MVN) as we expect the performance to be exact (up to MC error) for all correlation types. Then, we select a non-symmetric continuous distribution: a standard (rate =1) two-component multivariate gamma (MVG). Finally, we select distributions and marginal parameter values that are motivated by our RNA-seq data, namely values proximal to probabilities and sizes estimated from the data (see Example applications for estimation details). Thus we

Simulation Reps B	Correlation Types	Identical-margin 2D distribution
1000	Pearson (ρ_P)	$\mathbf{Y} \sim MVN(\mu = 0, \sigma = 1, \rho_i), i = 1, \ldots, 100$
$10,000$	Spearman (ρ_S)	$\mathbf{Y} \sim MVG(shape = 10, rate = 1, \rho_i), i = 1, \ldots, 100$
$100,000$	Kendall (τ)	$\mathbf{Y} \sim MVNB(p = 3 \times 10^{-4}, r = 4, \rho_i), i = 1, \ldots, 100$

arrive at a multivariate negative binomial (MVNB) $p_1 = p_2 = 3 \times 10^{-4}, r_1 = r_2 = 4, \rho \in \Omega$.

Table: Identical margin, bivariate simulation configurations to evaluate correlation matching accuracy and efficiency.

For each of the unique 9 simulation configurations described above, we estimate the correlation bounds and vary the correlations along a sequence of 100 points evenly placed within the bounds, aiming to explore Ω. Specifically, we set correlations $\{\rho_1 = (\hat{l} + \epsilon), \rho_2 = (\hat{l} + \epsilon) + \delta, \ldots, \rho_{100} = (\hat{u} - \epsilon)\}$, with $\hat{l}$ and $\hat{u}$ being the estimated lower and upper bounds, respectively, and increment value δ. The adjustment factor, $\epsilon = 0.01$, is introduced to handle numeric issues when the bound is specified exactly.

```
R> bivariate_normal_sims %>%
+     ggplot(aes(rho, rho_hat, color = type)) +
+     geom_point() +
+     geom_abline(slope = 1) +
+     facet_wrap(~ type + N) +
+     theme_bw()

R> bivariate_gamma_sims %>%
+     ggplot(aes(rho, rho_hat, color = type)) +
+     geom_point() +
+     geom_abline(slope = 1) +
+     facet_wrap(~ type + N) +
+     theme_bw()

R> bivariate_nbinom_sims %>%
+     ggplot(aes(rho, rho_hat, color = type)) +
+     geom_point() +
+     geom_abline(slope = 1) +
+     facet_wrap(~ type + N) +
+     theme_bw()

R> allDat <- bind_rows(
+     select(bivariate_normal_sims, margins, type, N, rho, rho_hat),
+     select(bivariate_gamma_sims, margins, type, N, rho, rho_hat),
+     select(bivariate_nbinom_sims, margins, type, N, rho, rho_hat)
+ ) %>%
+     mutate(margins = factor(margins, levels=c("norm", "gamma", "nbinom")),
+            type = factor(type, levels=c("Pearson", "Spearman", "Kendall")),
+            N = factor(N))
```

Figure 4 displays the aggregated bivariate simulation results. Table 2 contains the mean absolute error (MAE) in reproducing the desired dependency measures for the three bivariate scenarios.

```
R> tabMAE <- allDat %>%
+     group_by(N, type, margins) %>%
+     summarize(MAE = mean(abs(rho - rho_hat))) %>%
+     ungroup()
```

```
`summarise()` has grouped output by 'N', 'type'. You can override using the `.groups` argu
```

```
R> kable(tabMAE, booktabs = TRUE, format = "latex",
+       linesep = c("", "", "\\addlinespace"),
+       col.names = c("No. of random vectors",
+                     "Correlation type",
+                     "Distribution",
+                     "Mean abs. error"),
+       caption = "Average abolute error in matching the target dependency across the enti
```

Overall, the studies show that our methodology is generally accurate across the entire range of possible correlation values all three dependency measures, at least in these limited simulation settings for the rank-based correlations. Our Pearson matching performs nearly as well as Spearman or Kendall, except for a slight increase in error for negative binomial case. This is due the particularly large counts generated with our choice of parameters for p and r.

```
R> # https://www.datanovia.com/en/blog/how-to-change-ggplot-facet-labels/
R> # New facet label names
R> repsLabs <- paste0("B=", c("1,000", "10,000", "100,000") )
R> names(repsLabs) <- c(1000, 10000, 100000)
R> typeLabs <- c("Pearson", "Spearman", "Kendall")
R> names(typeLabs) <- c("Pearson", "Spearman", "Kendall")
R>
```

Table 2: Average abolute error in matching the target dependency across the entire range of possible correlations for each bivariate marginal.

No. of random vectors	Correlation type	Distribution	Mean abs. error
1000	Pearson	norm	0.0168
1000	Pearson	gamma	0.0159
1000	Pearson	nbinom	0.0188
1000	Spearman	norm	0.0203
1000	Spearman	gamma	0.0180
1000	Spearman	nbinom	0.0165
1000	Kendall	norm	0.0118
1000	Kendall	gamma	0.0099
1000	Kendall	nbinom	0.0089
10000	Pearson	norm	0.0048
10000	Pearson	gamma	0.0055
10000	Pearson	nbinom	0.0066
10000	Spearman	norm	0.0058
10000	Spearman	gamma	0.0054
10000	Spearman	nbinom	0.0060
10000	Kendall	norm	0.0034
10000	Kendall	gamma	0.0031
10000	Kendall	nbinom	0.0037
1e+05	Pearson	norm	0.0016
1e+05	Pearson	gamma	0.0016
1e+05	Pearson	nbinom	0.0032
1e+05	Spearman	norm	0.0018
1e+05	Spearman	gamma	0.0019
1e+05	Spearman	nbinom	0.0013
1e+05	Kendall	norm	0.0011
1e+05	Kendall	gamma	0.0010
1e+05	Kendall	nbinom	0.0010

```
R> # Set colors
R> ## RColorBrewer::display.brewer.all()
R> numColors <- 4
R> numGroups <- length(levels(allDat$margins))
R> myColors <- rev(RColorBrewer::brewer.pal(n = numColors, name = "Greys")[ ((numColors -
R>
R> allDat %>%
+     ggplot(aes(x = rho, y = rho_hat, color = margins)) +
+     geom_point(size = 2) +
+     scale_color_manual(values = myColors ) +
+     geom_abline(slope = 1, linetype = "dashed") +
+     labs(x = "Specified Correlation", y = "Estimated Correlation") +
+     facet_wrap(~ type + N, labeller = labeller(N = repsLabs, type = typeLabs)) +
+     theme_bw() +
+     theme(legend.position = "bottom", legend.direction = "horizontal")
```

5.2. Scale up to High Dimensions

With information of our method's accuracy from a low-dimensional perspective, we now turn to assessing whether `bigsimr` can scale to larger dimensional problems with practical computation times. Specifically, we ultimately generate $B = 1,000$ random vectors for $d = \{100, 250, 500, 1000, 2500, 5000, 10000\}$ for each correlation type, $\{Pearson, Spearman, Kendall\}$ while timing the algorithm's major steps.

To mimic the workflow, we first produce a synthetic "data set" by completing the following

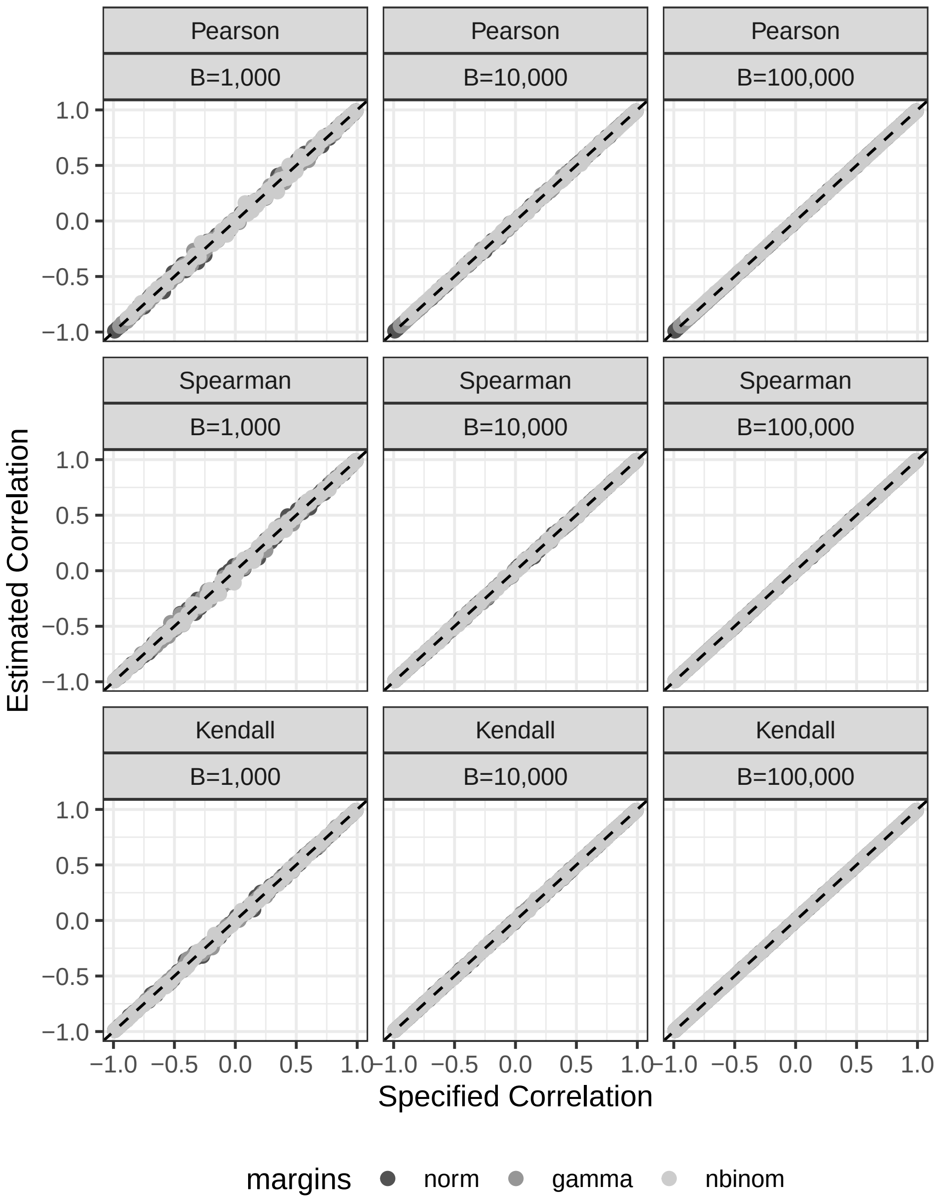

Figure 4: Bivariate simulations match target correlations across the entire range of feasible correlations. The horizontal axis plots the specified target correlations for each bivariate margin. Normal margins are plotted in dark dark grey, gamma in medium grey, and negative binomial in light grey. As the number of simulated vectors B increases from left to right, the variation in estimated correlations (vertical axis) decreases. The dashed line indicates equality between the specified and estimated correlations.

steps:

1. Produce heterogeneous gamma marginals by randomly selecting the j^{th} gamma shape parameter from $U_j \sim uniform(1, 10), j = 1, \ldots, d$ and the j^{th} rate parameter from $V_j \sim exp(1/5), j = 1, \ldots, d$, with the constant parameters determined arbitrarily.
2. Produce a random full-rank Pearson correlation matrix via `cor_randPD` of size $d \times d$.
3. Simulate a "data set" of $1,000 \times d$ random vectors via `rvec` (without matching the Pearson exactly).

With the synthetic data set in hand, we complete and time the following 4 steps:

i. Estimate the correlation matrix from the "data" in the *Compute Correlation* step.

1. Map the correlations to initialize the algorithm (Pearson to Pearson, Spearman to Pearson, or Kendall to Pearson) in the *Adjust Correlation* step.
2. Check whether the mapping produces a valid correlation matrix and, if not, find the nearest PD correlation matrix in the *Check Admissibility* step.
3. Simulate $1,000$ vectors via the Inverse Transform method in the *Simulate Data* step.

The experiments are conducted on a MacBook Pro carrying a 2.4 GHz 8-Core Intel Core i9 processor, with all 16 threads employed during computation. The results for $d \leq 500$ are fast with all times (except Kendall) under 3 seconds. Table 3 displays the total computation time of the algorithms steps 1 through 3, including estimation step i.

```
R> dat_long <- benchmark_dependences %>%
+   select(-total_time, -n_sim, -needed_near_pd) %>%
+   pivot_longer(cols = c(corr_time:sim_time), names_to = "Step", values_to = "Time") %>%
+   mutate(dim = factor(dim),
+          Step = factor(Step,
+                        levels = c("corr_time", "adj_time", "admiss_time", "sim_time"),
+                        labels = c("Compute Correlation",
+                                   "Adjust Correlation",
```

Table 3: Total time to produce 10,000 random vectors with a random correlation matrix and hetereogeneous gamma margins.

Dimension	Correlation type	Total Time (Seconds)
100	Pearson	0.080
100	Spearman	0.036
100	Kendall	0.116
250	Pearson	0.286
250	Spearman	0.371
250	Kendall	0.366
500	Pearson	2.932
500	Spearman	0.176
500	Kendall	2.183

```
+                                   "Check Admissibility",
+                                   "Simulate Data"))) %>%
+     rename(Correlation = corr_type, Dimensions = dim)
R>
R> tabTimes  <- dat_long %>%
+     filter(Dimensions %in% c(100, 250, 500)) %>%
+     group_by(Dimensions, Correlation) %>%
+     summarize('Total Time(Seconds)' = sum(Time)) %>%
+     ungroup()

`summarise()` has grouped output by 'Dimensions'. You can override using the `.groups` arg

R> tabTimes %>%
+     kable(booktabs = TRUE,
+           linesep = c("", "", "\\addlinespace"),
+           col.names = c("Dimension",
+                         "Correlation type",
+                         "Total Time (Seconds)"),
+           caption = 'Total time to produce 10,000 random vectors with a random correlati
```

The results with larger dimensional vectors show scalability to ultra-high dimensions for all three correlation types, although the total times do become much larger. Figure 5 displays computation times for $d = \{1000, 2500, 5000, 10000\}$. For d equal to 1000 and 2500, the total time is under a couple of minutes. At d of 5000 and 10,000, Pearson correlation matching in the *Adjust Correlation* step becomes costly. Interestingly, Pearson is actually faster than Kendall for $d = 10,000$ due to bottlenecks in *Compute Correlation* and *Check Admissibility.* Uniformly, matching Spearman correlations is faster, with total times under 5 minutes for $d = 10,000$, making Spearman correlations the most computationally-friendly dependency type. With this in mind, we scaled the simulation to $d = 20,000$ for the Spearman type and obtained the 1,000 vectors in under an hour (data not shown). In principle, this would enable the simulation of an entire human-derived RNA-seq data set. We note that for a given target correlation matrix and margins, steps i, 1, and 2 only need to be computed once and the third step, *Simulate Data*, is fast under all schemes for all d under consideration.

```
R> # Set colors
R> numColors <- 5
R> numGroups <- 4
R> myColors <- rev(brewer.pal(n=numColors, name="Greys")[((numColors-numGroups)+1):numColo
R>
R> dat_long %>%
+     filter(Dimensions %in% c(1000, 2500, 5000, 10000, 20000)) %>%
+     mutate(`Time (minutes)` = Time / 60) %>%
+     ggplot(aes(Correlation, `Time (minutes)`, fill=Step)) +
+     geom_bar(position = "stack", stat = "identity") +
+     scale_fill_manual(values = myColors ) +
+     scale_y_continuous(breaks = seq(0, 80, 10),
+                        minor_breaks = NULL) +
+     facet_grid(. ~ Dimensions) +
+     theme(axis.text.x = element_text(angle = -90))
```

Limitations, conclusions, and recommendations

In the bivariate studies, we chose arbitrary simulation parameters for three distributions, moving from the Gaussian to the discrete and non-normal, multivariate negative binomial. Under these conditions, the simulated random vectors sample the desired bivariate distribu-

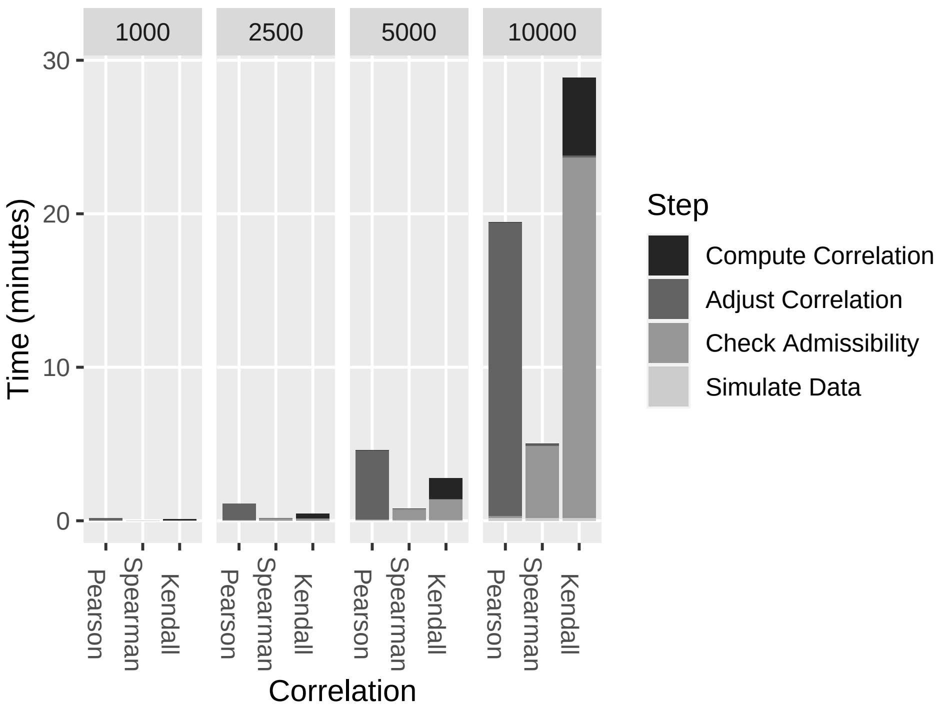

Figure 5: Computation times as d increases.

tion across the entire range of pairwise correlations for the three dependency measures. The simulation results could differ for other choices of simulation settings. Specifying extreme correlations near the boundary or Frechet bounds could result in poor simulation performance. Fortunately, it is straightforward to evaluate simulation performance by using strategies similar to those completed above. We expect our random vector generation to perform well for the vast majority of NORTA-feasible correlation matrices, but advise to check the performance before making inferences/further analyses.

Somewhat surprising, Kendall estimation and nearest PD computation scale poorly compared to Spearman and, even, approximate Pearson matching. In our experience, Kendall computation times are sensitive to the number of cores, benefiting from multi-core parallelization. This could mitigate some of the current algorithmic/implementation shortcomings. Despite this, Kendall matching is still feasible for most high-dimensional data sets. Finally, we note that one could use our single-pass algorithm `cor_fastPD` to produce a 'close' (not nearest

PD) to scale to even higher dimensions with greater loss of accuracy.

6. RNA-seq data applications

This section demonstrates how to simulate multivariate data using `bigsimr`, aiming to replicate the structure of high-dimensional dependent count data. In an illustration of our proposed methodology, we seek to simulate RNA-sequencing data by producing simulated random vectors mimicking the observed data and its generating process. Modeling RNA-seq using multivariate probability distributions is natural as inter-gene correlation is an inherent part of biological processes (Wang *et al.* 2009). And yet, many models do not account for this, leading to major disruptions to the operating characteristics of statistical estimation, testing, and prediction. See Efron (2012) for a detailed discussion with related methods and Wu and Smyth (2012), Schissler *et al.* (2018); Schissler *et al.* (2019) for applied examples. The following subsections apply `bigsimr`'s methods to real RNA-seq data, including replicating an estimated parametric structure, MC probability estimation, and MC evaluation of correlation estimation efficiency.

6.1. Simulating High-Dimensional RNA-seq data

```
R> d <- 1000
R> brca1000 <- example_brca %>%
+   select(all_of(1:d)) %>%
+   mutate(across(everything(), as.double))
```

We begin by estimating the structure of the TCGA BRCA RNA-seq data set (see Background). Ultimately, we will simulate $B = 10,000$ random vectors $\mathbf{Y} = (Y_1, \ldots, Y_d)^\top$ with $d = 1000$. We assume a multivariate negative binomial (MVNB) model as RNA-seq counts are often over-dispersed and correlated. Since all d selected genes exhibit over-dispersion (data not shown), we proceed to estimate the NB parameters $(r_i, p_i), i = 1, \ldots, d$, to determine the target marginal PMFs f_i. To complete specification of the simulation algorithm inputs, we estimate the Spearman correlation matrix $\mathbf{R}_S$ to characterize dependency.

With this goal in mind, we first estimate the desired correlation matrix using the fast implementation provided by `bigsimr`:

```
R> # Estimate Spearman's correlation on the count data
R> R_S <- bs$cor(as.matrix(brca1000), bs$Spearman)
```

Next, we estimate the marginal parameters. We use the method of moments (MoM) to estimate the marginal parameters for the multivariate negative binomial model. While, marginal distributions are from the same probability family (NB), they are heterogeneous in terms of the parameters probability and size (p_i, n_i) for $i, \ldots, d$. The functions below support this estimation for later use in `rvec`.

```
R> make_nbinom_margins <- function(sizes, probs) {
+   margins <- lapply(1:length(sizes), function(i) {
+     dist$NegativeBinomial(sizes[i], probs[i])
+   })
+   do.call(c, margins)
+ }
```

We apply these estimators to all 1000 genes across the 878 patients:

```
R> nbinom_fit <- apply(brca1000, 2, mom_nbinom)
R> sizes <- nbinom_fit["size",]
R> probs <- nbinom_fit["prob",]
R> nb_margins <- make_nbinom_margins(sizes, probs)
```

Notably, the estimated marginal NB probabilities $\{\hat{p}_i\}$ are small — ranging in the interval $[7.5305 \times 10^{-7}, 5.6592 \times 10^{-4}]$. This gives rise to highly variable counts and, typically, less restriction on potential pairwise correlation pairs. Once the functions are defined/executed to complete marginal estimation, we specify targets and generate the desired random vectors using `rvec`. Now we check admissibility of the specified correlation matrix.

```
R> # 1. Mapping step first
R> R_X <- bs$cor_convert(R_S, bs$Spearman, bs$Pearson)
R> # 2a. Check admissibility
R> (is_valid_corr <- bs$iscorrelation(R_X))
```

```
[1] FALSE
```

```
R> # 2b. compute nearest correlation
R> if (!is_valid_corr) {
+   R_X_pd <- bs$cor_nearPD(R_X)
+   ## Quantify the error
+   targets      <- R_X[lower.tri(R_X, diag = FALSE)]
+   approximates <- R_X_pd[lower.tri(R_X_pd, diag = FALSE)]
+   R_X          <- R_X_pd
+ }
R> summary(abs(targets - approximates))
```

```
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.000000 0.000201 0.000432 0.000534 0.000753 0.006931
```

While the exact $d \times d$ Spearman correlation matrix is not strictly admissible in our scheme (as seen by the non-positive definite result above), the approximation is close with a maximum absolute error of 0.0069 and average absolute error of 5.3366×10^{-4} across the 4.995×10^5 correlations.

```
R> sim_nbinom <- bs$rvec(10000, R_X, nb_margins)
R> colnames(sim_nbinom) <- colnames(brca1000)
```

Figure 6 displays the simulated counts and pairwise relationships for our example genes from Table 1. Simulated counts roughly mimic the observed data but with a smoother appearance due to the assumed parametric form and with less extreme points then the observed data in Figure 1. Figure 7 displays the aggregated results of our simulation by comparing the specified target parameter (horizontal axes) with the corresponding quantities estimated from the simulated data (vertical axes). The evaluation shows that the simulated counts approximately match the target parameters and exhibit the full range of estimated correlation from the data.

```
R> set.seed(2020-02-25)
R> num_genes <- 3
```

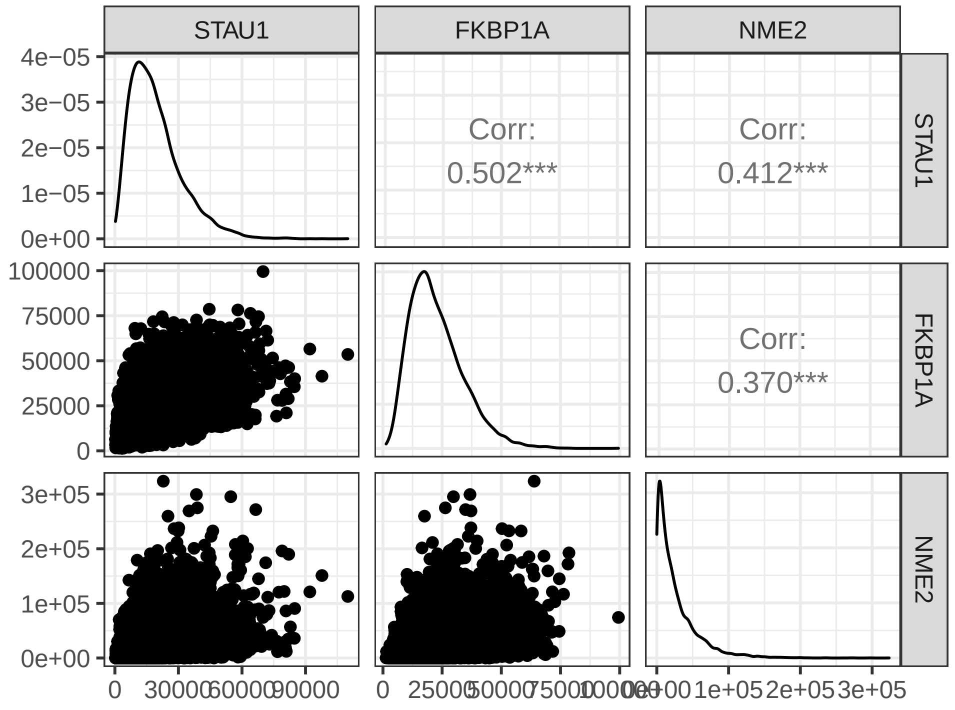

Figure 6: Simulated data for three selected high-expressing genes generally replicates the estimated data structure. The data do not exhibit outlying points, but do possess the desired Spearman correlations, central tendencies, and discrete values.

```
R> gene_sample <- sample(example_genes[1:1000], num_genes)
R>
R> ggpairs(data = as.data.frame(sim_nbinom[, gene_sample]),
+         upper = list(continuous = wrap('cor', method = "spearman"))) +
+   theme_bw()
```

```
Warning in cor.test.default(x, y, method = method, use = use): Cannot compute
exact p-value with ties

Warning in cor.test.default(x, y, method = method, use = use): Cannot compute
exact p-value with ties

Warning in cor.test.default(x, y, method = method, use = use): Cannot compute
exact p-value with ties
```

```
R> include_graphics('fig/ch050-figBRCA.png')
```

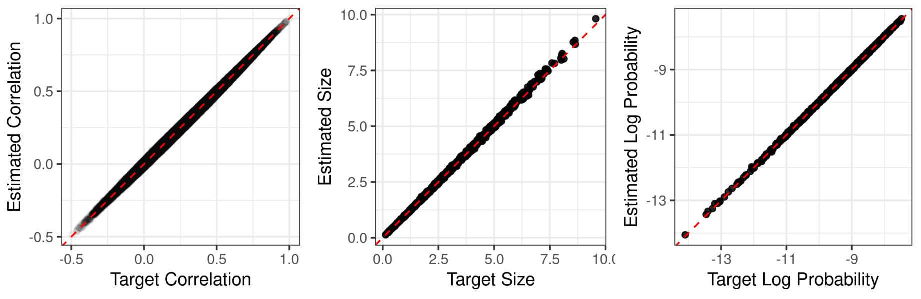

Figure 7: Simulated random vectors from a multivariate negative binomial replicate the estimated structure from an RNA-seq data set. The dashed red lines indicate equality between estimated parameters from simulated data (vertical axes) and the specified target parameters (horizontal axes).

Limitations, conclusions, and recommendations

The results show overall good simulation performance for our choice of parameters settings. Our settings were motivated by modeling high-expressing genes from the TCGA BRCA data set. In general, the ability to match marginal and dependence parameters depends on the particular joint probability model. We recommend to evaluate and tune your simulation until you can be assured of the accuracy.

6.2. Simulation-based joint probability calculations

Many statistical tasks require evaluation of a joint probability mass (or density) function:

$$P(\mathbf{Y} = \mathbf{y}),\ y_i \in \chi_i.$$

where χ_i is the sample space for the i^{th} component of the random vector $\mathbf{Y}$. Compact representations with convenient computational forms are rare for high-dimensional constructions, especially with heterogeneous, correlated marginal distributions (or margins of mixed data types). Given a large number of simulated vectors as produced above, estimated probabilities are readily given by counting the proportion of simulated vectors meeting the desired condition. In our motivating application, one may ask what is the probability that all genes expressed greater than a certain threshold value $\mathbf{y}_0$. This can be estimated as

$$\hat{P}(\mathbf{Y} \geq \mathbf{y_0}) = \frac{1}{B}\sum_{b=1}^{B} I(\mathbf{Y}^{(b)} \geq \mathbf{y_0}),$$

where $\mathbf{Y}^{(b)}$ is the b^{th} simulated vector from a total of B simulation replicates and $I(\cdot)$ is the indicator function. For example, we can estimate from our $B = 10,000$ simulated vectors that the probability of all genes expressing (i.e., $\mathbf{y}_i \geq 1, \forall\ i$) is 0.5979.

```
R> d <- ncol(sim_nbinom)
R> B <- nrow(sim_nbinom)
R> y0 <- 1
R> pHat <- mean(apply(sim_nbinom, 1, function(Y) {
+   all(Y >= y0)
+ }))
R> pHat
```

```
[1] 0.5979
```

6.3. Evaluation of correlation estimation efficiency

MC methods are routinely used in many statistical inferential tasks including estimation, hypothesis testing, error rates, and empirical interval coverage rates. To conclude the example applications, we demonstrate how `bigsimr` can be used to evaluate estimation efficiency. In particular, we wish to assess the error in our correlation estimation above. We used a conventional method, based on classical statistical theory which was not designed for high-dimensional data. Indeed, high-dimensional covariance estimation (and precision matrices) is an active area of statistical science, .e.g., (Won, Lim, Kim, and Rajaratnam 2013; Van Wieringen and Peeters 2016).

In this small example, we simulate $m = 30$ data sets with the number of simulated vectors matching the number of patients in the BRCA data set, $N = 878$. Since our simulation is much faster for the Pearson correlation type (see Figure 5), we only convert the Spearman correlation matrix once (and ensure it is positive definite). At each iteration, we estimate the quadratic loss (residual sum of squared errors) from the specified $\mathbf{R}_S$, producing a distribution of loss values.

```
R> # Simulate random vectors equal to the sample size
R> N <- nrow(example_brca)
R> # create m random vectors and estimate correlation
R> simRho <- replicate(n = m, expr = {
+   tmpSim <- bs$rvec(N , R_X, nb_margins)
+   bs$cor(tmpSim, bs$Spearman)
+ }, simplify = FALSE)
R> # Evaluate the residual sum of squared error
R> sapply(simRho, function(R) sum((R - R_S)^2))

R> frobenius_loss <- sapply(simRho, function(R) sum((R - R_S)^2))
```

The R summary function supplies the mean-augmented five-number summary of the quadratic loss distribution computed above.

```
R> summary(frobenius_loss)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    691     829     890     925     986    1224
```

This distribution could be compared to high-dimensional designed covariance estimators for guidance on whether the additional complexity and computation time are warranted.

7. Conclusion and discussion

We have introduced a general-purpose high-dimensional multivariate simulation algorithm and provide a user-friendly, high-performance R package [bigsimr]. The random vector generation method is inspired by NORTA (Cario and Nelson 1997) and Gaussian copula-based approaches (Madsen and Birkes 2013, Barbiero and Ferrari (2017), Xiao (2017)). The major contributions of this work are methods and software for flexible, scalable simulation of HD multivariate probability distributions with broad potential data analytic applications for modern, big-data statistical computing. For example, one could simulate high-resolution time series data, such as those consistent with an auto-regressive moving average model exhibiting a specified Spearman structure. Or our methods could be used to simulate sparsely correlated data, as many HD methods assume, via specifying a *spiked correlation matrix*.

It is customary to compare new tools and algorithms directly to existing competing methods and software. In this study, however, we only employ our proposed methodology, as our previous work has shown that existing `R` tools are simply not designed to meet our high-dimensional goal (see Li *et al.* (2019) for evaluations of the `R` `copula` package and others). For the bivariate simulations, existing packages such as `nortaRA` work well to match Pearson correlations exactly.

There are limitations to the methodology and implementation. We could only investigate selected multivariate distributions in our Monte Carlo studies. There may be instances that the methods do not perform well. Along those lines, we expect that correlation values close to the boundary of the feasible region could result in algorithm failure. Another issue is that for discrete distributions, we use continuous approximations when the support set is large. This could limit the scalability/accuracy for particular discrete/mixed multivariate distributions. Our method would also benefit computationally from faster Kendall estimation and more finely tuned nearest PD calculation for non-admissible Kendall matrices.

Supplementary Materials

We provide an open-source implementation of our methodology as the `Bigsimr.jl` Julia package, hosted on github at `https://github.com/adknudson/Bigsimr.jl`. Additionally we provide R and Python interfaces to `Bigsimr`, respectively at `https://github.com/SchisslerGroup/r-bigsimr` and `https://github.com/SchisslerGroup/python-bigsimr`. `Bigsimr.jl` is an ongoing project, and feature requests or issues can be submitted to `https://github.com/adknudson/Bigsimr.jl/issues`.

```
R> knitr::include_graphics('images/hex-bigsimr.png')
```

Acknowledgment(s)

The authors gratefully acknowledge Heather Knudson's graphic design for the `bigsimr` R Package. The results published here are in whole or part based upon data generated by the TCGA Research Network: https://www.cancer.gov/tcga.

Disclosure statement

The authors report no conflict of interest.

Funding

Research reported in this publication was supported by grants from the National Institutes of General Medical Sciences (5 U54 GM104944, GM103440) from the National Institutes of Health.

Appendix

Consider a bivariate example where we have a correlated $(Y_1, Y_2)^\top$ with $Y_1 \sim N(\mu_1, \sigma_1^2)$ (normal) and $Y_2 \sim LN(\mu_2, \sigma_2^2)$ (lognormal). First, let us note that when we get such a vector from the `bigsimr` package then in fact it can be represented as

$$(Y_1, Y_2)^\top \overset{d}{=} \left(X_1, e^{X_2}\right)^\top, \tag{9}$$

where $(X_1, X_2)^\top \sim N_2(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, which is bivariate normal with mean vector $\boldsymbol{\mu} = (\mu_1, \mu_2)^\top$ and variance-covariance matrix

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix} \tag{10}$$

To see this, consider the three steps of the NORTA construction:

1. Generate $(Z_1, Z_2)^\top \sim N_2(\mathbf{0}, \boldsymbol{R})$, where

$$\boldsymbol{R} = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}. \tag{11}$$

2. Transform $(Z_1, Z_2)^\top$ to $(U_1, U_2)^\top$ viz. $U_i = \Phi(Z_i)$, $i = 1, 2$, where $\Phi(\cdot)$ is the standard normal CDF.

3. Return $(Y_1, Y_2)^\top$, where $Y_i = F_i^{-1}(U_i)$, $i = 1, 2$, and $F_i(\cdot)$ is the CDF of Y_i and $F_i^{-1}(\cdot)$ is its inverse (the quantile function).

In this example, F_1 is the CDF of $N(\mu_1, \sigma_1^2)$ while F_2 is the CDF of $LN(\mu_2, \sigma_2^2)$, so that the two required quantile functions are

$$F_1^{-1}(u) = \mu_1 + \sigma_1 \Phi^{-1}(u) \text{ and } F_2^{-1}(u) = e^{\mu_2 + \sigma_2 \Phi^{-1}(u)}, \tag{12}$$

as can be seen by standard calculation. Thus, when we apply Step 3 of the NORTA algorithm, we get

$$F_1^{-1}(U_1) = \mu_1 + \sigma_1 \Phi^{-1}(\Phi(Z_1)) = \mu_1 + \sigma_1 Z_1 \text{ and } F_2^{-1}(U_2) = e^{\mu_2 + \sigma_2 \Phi^{-1}(\Phi(Z_2))} = e^{\mu_2 + \sigma_2 Z_2}, \tag{13}$$

where $(X_1, X_2)^\top = (\mu_1 + \sigma_1 Z_1, \mu_2 + \sigma_2 Z_2)^\top \sim N_2(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Consequently, the vector $(Y_1, Y_2)^\top$ obtained in Step 3 of the algorithm has the structure provided by (**??**).

Next, we provide the exact covariance structure of the random vector $(Y_1, Y_2)^\top$ given by (**??**), and relate the correlation of Y_1 and Y_2 to ρ, which is the correlation of the normal variables Z_1, Z_2 (and also X_1 and X_2). A straightforward albeit somewhat tedious algebra produces the following result.

Lemma 1. *Let* $\mathbf{Y} = (Y_1, Y_2)^\top$ *admit the stochastic representation (**??**), where* $\mathbf{X} = (X_1, X_2)^\top \sim N_2(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ *with* $\boldsymbol{\mu}$ *and* $\boldsymbol{\Sigma}$ *as above. Then the mean vector and the variance-covariance matrix* *of* $\mathbf{Y}$ *are given by*

$$\boldsymbol{\mu}_{\mathbf{Y}} = \left(\mu_1, e^{\mu_2+\sigma_2^2/2}\right)^\top \tag{14}$$

and

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 e^{\mu_2+\sigma_2^2/2} \\ \rho\sigma_1\sigma_2 e^{\mu_2+\sigma_2^2/2} & \left[e^{\mu_2+\sigma_2^2/2}\right]^2 \left[e^{\sigma_2^2} - 1\right], \end{bmatrix} \tag{15}$$

respectively.

Proof. The values of the means and the variances are obtained immediately from normal marginal distribution of Y_1 and lognormal marginal distribution of Y_2. It remains to establish the covariance of Y_1 and Y_2,

$$\mathrm{Cov}(Y_1, Y_2) = \mathbb{E}(Y_1Y_2) - \mathbb{E}(Y_1)\mathbb{E}(Y_2), \tag{16}$$

and in particular the first expectation on the right-hand-side above. By using the tower property of expectations, the latter expectation can be expressed as

$$\mathbb{E}(Y_1Y_2) = \mathbb{E}\left(X_1e^{X_2}\right) = \mathbb{E}\left\{\mathbb{E}\left(X_1e^{X_2}|X_1\right)\right\} = \mathbb{E}\left\{X_1\mathbb{E}\left(e^{X_2}|X_1\right)\right\}. \tag{17}$$

Further, by using the fact that the conditional distribution of X_2 given $X_1 = x_1$ is normal with mean $\mathbb{E}(X_2|X_1 = x_1) = \mu_2 + \rho\sigma_2(x_1 - \mu_1)/\sigma_1$ and variance $\sigma_2^2(1 - \rho^2)$, one can relate the inner expectation on the far right in (**??**) to the moment generating function $M(t)$ of this conditional distribution evaluated at $t = 1$, leading to

$$E\left(e^{X_2}|X_1\right) = e^{\mu_2+\rho\sigma_2(x_1-\mu_1)/\sigma_1+\sigma_2^2(1-\rho^2)/2}, \tag{18}$$

so that

$$\mathbb{E}(Y_1Y_2) = e^{\mu_2+\frac{1}{2}\sigma_2^2(1-\rho^2)}\mathbb{E}\left\{X_1e^{\rho\frac{\sigma_2}{\sigma_1}(X_1-\mu_1)}\right\}. \tag{19}$$

Since X_1 is normal with mean μ_1 and variance σ_1^2, the expectation in (**??**) becomes

$$\mathbb{E}\left\{X_1 e^{\rho\frac{\sigma_2}{\sigma_1}(X_1-\mu_1)}\right\} = \int_{-\infty}^{\infty} x_1 e^{\rho\frac{\sigma_2}{\sigma_1}(x_1-\mu_1)} \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{1}{2\sigma_1^2}(x_1-\mu_1)^2}. \tag{20}$$

Upon substituting $u = x_1 - \mu_1$, followed by some algebra, we arrive at

$$\mathbb{E}\left\{X_1 e^{\rho\frac{\sigma_2}{\sigma_1}(X_1-\mu_1)}\right\} = e^{\frac{1}{2}\sigma_2^2\rho^2} \int_{-\infty}^{\infty} (u+\mu_1) \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{1}{2\sigma_1^2}(u-\rho\sigma_1\sigma_2)^2}. \tag{21}$$

Since the integral in (**??**) is the expectation $\mathbb{E}(X + \mu_1)$, where X is normal with mean $\rho\sigma_1\sigma_2$ and variance σ_1^2, we conclude that

$$\mathbb{E}\left\{X_1 e^{\rho\frac{\sigma_2}{\sigma_1}(X_1-\mu_1)}\right\} = e^{\frac{1}{2}\sigma_2^2\rho^2}(\rho\sigma_1\sigma_2 + \mu_1), \tag{22}$$

which, in view of (**??**), leads to

$$\mathbb{E}(Y_1 Y_2) = e^{\mu_2 + \frac{1}{2}\sigma_2^2}(\rho\sigma_1\sigma_2 + \mu_1). \tag{23}$$

Finally, (**??**), along with the expressions for the means of Y_1 and Y_2, produce the covariance of Y_1 and Y_2:

$$\mathrm{Cov}(Y_1, Y_2) = e^{\mu_2 + \frac{1}{2}\sigma_2^2}(\rho\sigma_1\sigma_2 + \mu_1) - \mu_1 e^{\mu_2 + \frac{1}{2}\sigma_2^2} = \rho\sigma_1\sigma_2 e^{\mu_2 + \frac{1}{2}\sigma_2^2}. \tag{24}$$

This concludes the proof □

Using the above result, we can directly relate the correlation coefficient of Y_1 and Y_2 with that of X_1 and X_2,

$$\rho_{Y_1,Y_2} = \rho \frac{\sigma_2}{\sqrt{e^{\sigma_2^2} - 1}}, \tag{25}$$

where ρ is the correlation of X_1 and X_2. The above result is useful when studying the possible range of correlation of Y_1 and Y_2 in the above example. It can be shown that the factor on the far right in (**??**) is a monotonically decreasing function of σ_2 on $(0, \infty)$, with the limits of 1 and 0 at zero and infinity, respectively. Thus, in principle, the range of correlation of

Y_1 and Y_2 is the same as that of X_1 and X_2, as the factor on the far right in (**??**) can be made arbitrarily close to 1. However, by changing this factor we may affect the marginal distributions of Y_1 and Y_2. It can be shown that if the marginal distributions of Y_1 and Y_2 are fixed, then the relation (**??**) becomes

$$\rho_{Y_1,Y_2} = \rho\sqrt{\frac{\log(1+c_2^2)}{c_2^2}}, \tag{26}$$

where $c_2 = \sigma_{Y_2}/\mu_{Y_2}$ is the *coefficient of variation* (CV) of the variable Y_2. Thus, the range of possible correlations in this model is not affected by the distribution of Y_1, and is determined by the CV of Y_2 as follows:

$$-\sqrt{\frac{\log(1+c_2^2)}{c_2^2}} \leq \rho_{Y_1,Y_2} \leq \sqrt{\frac{\log(1+c_2^2)}{c_2^2}}. \tag{27}$$

Plugging in the estimates of these quantities from Section 4, we see that

$$\frac{\hat{\sigma}_2}{\sqrt{e^{\hat{\sigma}_2^2}-1}} = 0.881. \tag{28}$$

Thus, the possible range of correlation becomes $(-0.881, 0.881)$. This approximately agrees with the MC results provided.

As an invited speaker at CMStatistics 2020, I was delighted to learn of the opportunity to have our work included in the first issue of CSDA Annals of Statistical Data Science.

On behalf of my co-authors, I'm pleased to submit our manuscript entitled *Simulating High-Dimensional Multivariate Data using the `Bigsimr` Package* for your consideration. We feel that the work is well suited to the indicated themes and hope that it will be considered a valuable contribution to this prestigious journal.

The entire article can be reproduced from code and is housed at `https://github.com/SchisslerGroup/article_bigsimr`, which we are glad to make available upon request to the reviewers and, after acceptance, the general public.

I hope that everything is in order. If not, or if you have any additional needs, please do not hesitate to contact me. Thank you and all those involved in this process.

References

Barbiero A, Ferrari PA (2017). "An R package for the simulation of correlated discrete variables." *Communications in Statistics - Simulation and Computation*, **46**(7), 5123–5140. ISSN 0361-0918. doi:10.1080/03610918.2016.1146758. URL https://www.tandfonline.com/doi/full/10.1080/03610918.2016.1146758.

Cario MC, Nelson BL (1997). "Modeling and generating random vectors with arbitrary marginal distributions and correlation matrix." *Technical report*.

Chambers JM, Cleveland WS, Kleiner B, Tukey PA (1983). *Graphical Methods for Data Analysis*. Wadsworth & Brooks, Belmont, CA.

Chen H (2001). "Initialization for NORTA: Generation of Random Vectors with Specified Marginals and Correlations." *INFORMS Journal on Computing*, **13**(4), 312–331.

Chernick MR (2008). *Bootstrap Methods: A Guide for Practitioners and Researchers*. 2nd edition. Hoboken, NJ.

Conesa A, Madrigal P, Tarazona S, Gomez-Cabrero D, Cervera A, McPherson A, Szcześniak MW, Gaffney DJ, Elo LL, Zhang X, Mortazavi A (2016). "A survey of best practices for RNA-seq data analysis." doi:10.1186/s13059-016-0881-8.

Demirtas H, Hedeker D (2011). "A practical way for computing approximate lower and upper correlation bounds." *American Statistician*, **65**(2), 104–109. ISSN 00031305. doi:10.1198/tast.2011.10090.

Efron B (2007). "Correlation and large-scale simultaneous significance testing." *Journal of the American Statistical Association*, **102**(477), 93–103. ISSN 01621459. doi:10.1198/016214506000001211.

Efron B (2012). *Large-Scale Inference: Empirical Bayes Methods for Estimation, Testing and Prediction*. 1 edition. Cambridge University Press, Cambridge.

Ghosh S, Henderson SG (2002). "Properties Of The Norta Method In Higher Dimensions." *Proceedings of the 2002 Winter Simulation Conference*, pp. 263–269.

Higham NJ (2002). "Computing the nearest correlation matrix—a problem from finance." *IMA journal of Numerical Analysis*, **22**(3), 329–343.

Kruskal WH (1958). "Ordinal Measures of Association." *Journal of the American Statistical Association*, **53**(284), 814–861. ISSN 1537274X. doi:10.1080/01621459.1958.10501481.

Li ST, Hammond JL (1975). "Generation Of Pseudorandom Numbers With Specified Univariate Distributions And Correlation Coefficients." *IEEE Transactions on Systems, Man and Cybernetics*, **5**, 557–61. ISSN 21682909. doi:10.1109/TSMC.1975.5408380.

Li X, Schissler AG, Wu R, Barford L, Harris, Fredrick C J, Harris FC (2019). "A Graphical Processing Unit Accelerated NORmal to Anything Algorithm for High Dimensional Multivariate Simulation." *Advances in Intelligent Systems and Computing*, pp. 339–345. doi:10.1007/978-3-030-14070-0_46.

Madsen L, Birkes D (2013). "Simulating dependent discrete data." *Journal of Statistical Computation and Simulation*. ISSN 00949655. doi:10.1080/00949655.2011.632774.

Mari DD, Kotz S (2001). *Correlation and dependence*. World Scientific. ISBN 1860942644.

Muino J, Voit EO, Sorribas A (2006). "GS-distributions: A new family of distributions for continuous unimodal variables." *Computational statistics & data analysis*, **50**(10), 2769–2798.

Nelsen RB (2007). *An Introduction to copulas*. 2 edition. Springer Science & Business Media, New York. ISBN 9781475719062.

Nikoloulopoulos AK (2013). "Copula-based models for multivariate discrete response data." In *Lecture Notes in Statistics: Copulae in Mathematical and Quantitative Finance*, 213 edition, pp. 231–249. Springer, Heidelberg.

Park CG, Park T, Shin DW (1996). "A Simple Method for Generating Correlated Binary Variates." *American Statistician*. ISSN 15372731. doi:10.1080/00031305.1996.10473557.

Qi H, Sun D (2006). "A quadratically convergent Newton method for computing the nearest correlation matrix." *SIAM journal on matrix analysis and applications*, **28**(2), 360–385.

Schissler AG, Aberasturi D, Kenost C, Lussier YA (2019). "A Single-Subject Method to Detect Pathways Enriched With Alternatively Spliced Genes." *Frontiers in Genetics*, **10**(414). ISSN 1664-8021. doi:10.3389/fgene.2019.00414. URL https://www.frontiersin.org/article/10.3389/fgene.2019.00414/full.

Schissler AG, Piegorsch WW, Lussier YA (2018). "Testing for differentially expressed genetic pathways with single-subject N-of-1 data in the presence of inter-gene correlation." *Statistical Methods in Medical Research*, **27**(12), 3797–3813. ISSN 0962-2802. doi:10.1177/0962280217712271. URL http://journals.sagepub.com/doi/10.1177/0962280217712271.

Sklar A (1959). "Fonctions de R{é}partition {à} n Dimensions et Leurs Marges." *Publications de L'Institut de Statistique de L'Universit{é} de Paris*.

Úbeda-Flores M, Fernández-Sánchez J (2017). "Sklar's theorem: The cornerstone of the Theory of Copulas." In *Copulas and Dependence Models with Applications*. doi:10.1007/978-3-319-64221-5_15.

Van Wieringen WN, Peeters CF (2016). "Ridge estimation of inverse covariance matrices from high-dimensional data." *Computational Statistics and Data Analysis*. ISSN 01679473. doi:10.1016/j.csda.2016.05.012. 1403.0904.

Wang Z, Gerstein M, Snyder M (2009). "RNA-Seq: A revolutionary tool for transcriptomics." doi:10.1038/nrg2484.

Won JH, Lim J, Kim SJ, Rajaratnam B (2013). "Condition-number-regularized covariance estimation." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **75**(3), 427–450. ISSN 13697412. doi:10.1111/j.1467-9868.2012.01049.x. URL http://doi.wiley.com/10.1111/j.1467-9868.2012.01049.x.

Wu D, Smyth GK (2012). "Camera: A competitive gene set test accounting for inter-gene correlation." *Nucleic Acids Research*, **40**(17), e133–e133. ISSN 03051048. doi:10.1093/nar/gks461. URL https://academic.oup.com/nar/article/40/17/e133/2411151.

Xiao Q (2017). "Generating correlated random vector involving discrete variables." *Commu-*

nications in Statistics - Theory and Methods. ISSN 1532415X. doi:10.1080/03610926.2015.1024860.

Xiao Q, Zhou S (2019a). "Matching a correlation coefficient by a Gaussian copula." *Communications in Statistics - Theory and Methods*, **48**(7), 1728–1747. ISSN 1532415X. doi:10.1080/03610926.2018.1439962.

Xiao Q, Zhou S (2019b). "Matching a correlation coefficient by a Gaussian copula." *Communications in Statistics-Theory and Methods*, **48**(7), 1728–1747.

Yan J (2007). "Enjoy the Joy of Copulas : With a Package copula." *Journal Of Statistical Software*, **21**(4), 1–21. ISSN 15487660. URL http://www.jstatsoft.org/v21/i04.

Affiliation:

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd