Simulating High-Dimensional Multivariate Data

using the bigsimr R Package

Alexander D. Knudson, Tomasz J. Kozubowski, Anna K. Panorska, Juli Petereit, and Alfred G. Schissle

Abstract

It is critical to realistically simulate data when conducting Monte Carlo studies and methods. But 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 may resort to simulation designs that posit independence or perform arbitrary data transformations. This greatly diminishes insights into the empirical operating characteristics of any proposed methodology, such as false positive rates, statistical power, interval coverage, and robustness. This article introduces the `bigsimr` R package that provides a flexible, scaleable procedure to simulate high-dimensional random vectors with given marginal characteristics and dependency measures. We'll describe the functions included in the package, including multi-core and graphical-processing-unit accelerated algorithms to simulate random vectors, estimate correlation matrices, and find close positive semi-definite matrices. Finally, we demonstrate the power of `bigsimr` by applying these functions to our motivating dataset — RNA-sequencing data obtained from breast cancer tumor samples with sample size $n = 1212$ patients and dimension $d > 1000$.

Contents

1 Introduction

Massive high-dimensional data sets are now commonplace in many areas of scientific inquiry. This is particularly true when simulating massive *multivariate*, *non-normal* distributions, arising naturally in many fields of study in this era of big data. As new methods are developed for these data, a fundamental challenge lies in designing and conducting simulation studies to assess the operating characteristics of proposed methodology, such as false positive rates, statistical power, interval coverage, and robustness — often in comparison to existing methods. Further, efficient simulation empowers statistical computing strategies, such as the parameteric bootstrap (Rizzo 2007) to simulate from a hypothesized null model, providing inference in analytically challenging settings. Such Monte Carlo techniques become difficult for high-dimensional data with the current existing algorithms and tools.

As others 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 2019). For continous non–normal data, the well-known NORmal To Anything (NORTA) algorithm (Cario and Nelson 1997) and other copula (Nelsen 2007) approaches 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 (see Li et al. (2019)). For discrete data, multiple techniques have been proposed. But some schemes have major flaws such as failing to obtain the full range of possible dependencies (for example, feasible for only positive correlations Park, Park, and Shin (1996)). While more recent approaches (Madsen and Birkes 2013; Xiao 2017; Barbiero and Ferrari 2017) have largely rememdied this issue for low dimesional problems, the existing tools are not designed to scale to high dimensions.

Another central issue lies characterizing dependency 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 completely. As we will see, however, simulating arbitrary random vectors that match a target Pearson correlation matrix exactly is computationally intense (Chen 2001; Xiao 2017). On the other hand, an analyst may consider the use of nonparametric correlation measures to better characterize monotone dependency, such as Spearman's ρ and Kendall's τ. Throughout, we'll emphasize matching these nonparametric dependency measures as our aim lies in modeling non-normal data.

In the study, we present scalable, flexible multivariate simulation algorithms. 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 parallelized, high-performance software. The algorithm design relies on useful properties of nonparameteric correlation measures, namely invariance under monotone transformation and well-known closed form relationships between dependency measures for the multivariate normal (MVN) distribution.

The study proceeds with background and notation, including a description of our motivating example in RNA-sequencing (RNA-seq) breast cancer data. Then we describe and justify our simulation methodology and related algorithms. Next, a detailed illustrative low-dimensional example of basic and advanced use of the `bigsimr` R package is provided. Then we proceed with extensive Monte Carlo studies under various distributional assumptions. After evaluating *in silico*, we revisit our high-dimensional motivating RNA-seq example and employ our methods to commonplace statistical computing tasks. Finally, we'll discuss the method's utility, limitations, and future directions.

2 Background

The article presents several algorithms to work with high-dimesional multivariate data, but all `bigsimr` algorithms were originally designed to support a single task: to generate random vectors drawn from multivariate probability distribution with given marginal distributions and component-wise dependency metrics. Specifically, our goal is to efficiently simulate a large number, B, of random vectors $\mathbf{Y} = (Y_1, \ldots, Y_d)^\top$ with **correlated** components and hetereogeneous marginal distributions, described via cumulative distribution functions (CDFs) F_i, and d can be very large.

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 "scaleability" properties (**SP**). The BPs are adapted from an existing criteria due to Nikoloulopoulos (2013). Our simulation strategy should allow:

- 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, computable in a reasonable amount time.
- SP2: Procedure must scale to high dimensions while maintaining accuracy.

To fix ideas and provide example applications enabled via `bigsimr`, the next section describes a motivating data set that originally inspired the authors' interest in developing this methodology.

2.1 Motivating example: RNA-seq data

Simulating high-dimensional, non-normal, correleated data motivates this work — in pursuit of modeling RNA-sequencing (RNA-seq) data (Wang, Gerstein, and Snyder 2009; Conesa et al. 2016) derived from breast cancer patients. The RNA-seq data-generating process involves counting how often a particular 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. This results in count data corresponding to over 20,000 genes (coding genomic regions) or even over 77,000 isoforms when alternating spliced mRNA are counted (for human derived samples). Yet due to inherent biological processes, gene expression data exhibits correlation (co-expression) across genes (Efron 2007; Schissler, Piegorsch, and Lussier 2018).

Specifically, we'll illustrate our methodology using the Breast Invasive Carcinoma (BRCA) data set housed in The Cancer Genome Atlas (TCGA; see Acknowledgements). For simplicity, we begin by filtering to retain the top 1% highest expressing (in terms of median expression) genes of the 20,501 gene measurements from $N = 1212$ patients' tumor samples, resulting in $d = 206$ genes. This results in 2.1115×10^4 pairwise dependencies among the bivariate marginals. To illustrate our methodology's flexible/robust simulation scheme and better align with the actual data-generating process, we remove the statistical adjustment for read alignment ambiguity (RNA-seq by Expectation Maximization; RSEM; Li and Dewey (2011)) by simple rounding to integer values. Table 1 displays counts for three selected genes for the first five patients' breast tumor samples.

Figure 1 plots the marginal distributions and estimated Spearman's correlations (see Equation (2) below).

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

Table 1: mRNA counts for three selected high-expressing genes from the first five observations of the BRCA data set.

RPL5	TXNIP	VIM
20283	13401	26883
18614	11365	28806
31378	5365	22221
37861	5873	26871
17902	12564	15985

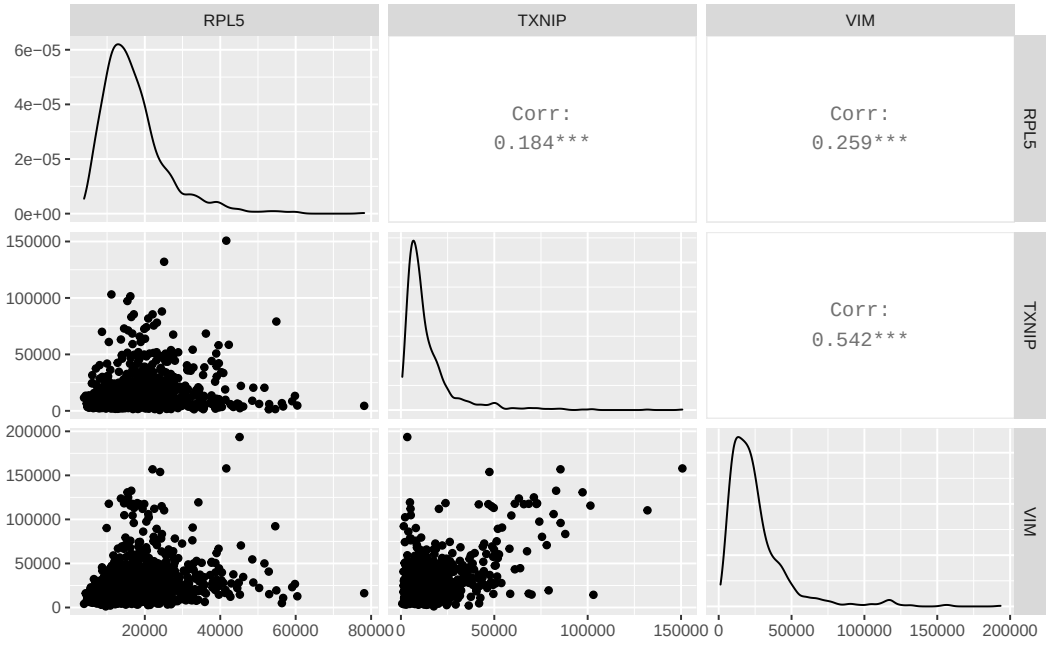

Figure 1: Marginal scatterplots, densities, and estimated pairwise Spearman's *rho* for three example genes. The data possess heavy-right tails, are discrete, and have non-trivial intergene correlations. Modeling these data motivate our simulation methodology.

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

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, 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 nonparameteric, rank-based measures common in practice are Spearman's correlation (denoted ρ_S) and Kendall's τ. Formally, define

$$\rho_S(X,Y) = 3\left[P\left[(X_1 - X_2)(Y_1 - Y_3) > 0\right] - P\left[(X_1 - X_2)(Y_1 - Y_3) < 0\right]\right] \tag{2}$$

where $(X_1, Y_1) \stackrel{d}{=} (X, Y), X_2 \stackrel{d}{=} X, Y_3 \stackrel{d}{=} Y$ with X_2 and Y_3 are independent of one other and of (X_1, Y_1). Spearman's ρ_S has an appealing correspondence as the Pearson's correlaiton 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). By concordance we mean that orderings have the same direction (e.g., if $X_i < X_j$, then $Y_i < Y_j$) and is determinded by the ranks of the values, not the values themselves.

Both τ and ρ_S are **invariant to under monotone transformations** of the underlying random variates. As we will describe more fully in the Algorithms section, this property is preferred in our scheme to satistify properties SP1 and SP2, above.

Correspondence among Pearson, Spearman, τ correlations. This is no closed form, general correspondence among the rank-based measures and the Pearson correlation coefficent, as the marginal distributions F_i are intrinstic in their calculation. But for **bivariate normal vectors**, however, the correspondence is well-known:

$$\rho_P = sin\left(\tau \times \frac{\pi}{2}\right), \tag{3}$$

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

$$\rho_P = 2 \times sin\left(\rho_S \times \frac{\pi}{6}\right). \tag{4}$$

These facts are also critical in our simulation algorithm to broaden the dependency measures supported by `bigsimr`, in a computationally effective manner.

Discrete marginal considerations. Spearman's correlation for discrete marginal suffers some issues due to the nonzero probability of ties (for example, the Spearman's correlation of a discrete-valued random variable X with itself could be less than 1; Madsen and Birkes (2013)). One remedy for this issue is to rescale Equation (2). For two random variables X, Y with probability mass functions (PMFs) or probability densities functions (PDFs) $p(x)$ and $q(y)$, respectively, define the rescaled Spearman's correlation as

$$\rho_{RS}(X,Y) = \frac{\rho_s(X,Y)}{\left[\left[1 - \sum_x p(x)^3\right]\left[1 - \sum_y q(y)^3\right]\right]^{1/2}}. \tag{5}$$

Note that with continuous marginals the rescaling returns ρ_S. For discrete marginals with large or infinite support, computing the adjustment factors $\sum_x p(x)^3$, $\sum_y p(y)^3$ over all large number of pairs becomes

expensive (often violating desired property SP1). And further the infinite sums must be approximately for count-valued data and potentially violating desired property SP2.

Also we'll note that for a discrete random variable Y_i, some care must be taken to define the quantile function F_i^{-1}. Let

$$F_i^{-1} = inf\{y : F_i(y) \geq u\}. \tag{6}$$

Marginal-dependent bivariate correlation bounds. Given two marginal distributions, ρ_P is not free to vary the entire range of possible correlations $[-1, 1]$. The so-called *Frechet-Hoeffding bounds* are well-studied (for example, see Nelsen (2007) and Barbiero and Ferrari (2017)). This situation gives strict restraints on the possible bounds and cannot be overcome through algorithm design. In general the bounds are given by

$$\rho_P^{max} = \rho_P\left(F_1^{-1}(U), F_2^{-1}(U)\right), \quad \rho_P^{min} = \rho_P\left(F_1^{-1}(U), F_2^{-1}(1-U)\right) \tag{7}$$

where U is a uniform random variable in $(0, 1)$, and F_1^{-1}, F_2^{-1} are the inverse cdf of random variables X_1 and X_2, respectively. For discrete random variables, define F^{-1} as in Equation (6).

2.3 Gaussian copulas

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

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

$$F(x_1, \ldots, x_d) = \mathbb{P}(X_1 \leq x_1, \ldots, X_d \leq x_d) = C(F_1(x_1), \ldots, F_d(x_d)), \;\; x_i \in \mathbb{R}, i = 1, \ldots, d.$$

A Gaussian copula is the case where all marginal CDFs F_i are the standard normal cdf, Φ. This representation corresponds to a multivariate normal distribution with standard normal marginal distributions and covariance matrix $\mathbf{R_P}$. But since the marginals are standardized to have unit variance, this $\mathbf{R_P}$ 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

$$F_{\mathbf{R}}(x_1, \ldots, x_d) = C_{\mathbf{R}}(\Phi(x_1), \ldots, \Phi(x_d)), \tag{8}$$

where $\Phi(\cdot)$ is the standard normal CDF. Note that the copula $C_{\mathbf{R}}$ is the familar 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) *any* random vector can be obtained via transformations involving copula functions. For example, using Gaussian copulas, we can construct a random vector $\mathbf{Y} = (Y_1, \ldots, Y_d)$ with $Y_i = F_i^{-1}(U_i)$, $i = 1, \ldots, d$, via $\mathbf{U} = (U_1, \ldots, U_d)$ viz $U_i = \Phi(X_i)$, $i = 1, \ldots, d$ povides $Y_i \sim F_i$, $\forall\, i$.

3 Algorithms

This section describes our methods involved in simulating a random vector $\mathbf{Y}$ with Y_i components for $i = 1, 2, \ldots, d$. 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, (rescaled) Spearman correlation ρ_S, and/or Kendall's

τ. The method is best understand as a **high-performance Gaussian copula** (Equation (8)) providing a high-dimensianl NORTA-inspired algorithm.

3.1 NORmal To Anything (NORTA)

The well-known NORTA algorithm (Cario and Nelson 1997) can be used simulate a random vector $\mathbf{Y}$ with variance-covariance matrix $\Sigma_{\mathbf{Y}}$. Specifically, the NORTA algorithm follows like this:

1. Simulate a random vector $\mathbf{Z}$ with d **independent** and **identical** standard normal components.
2. Determine the input matrix $\Sigma_{\mathbf{Z}}$ that corresponds with the specified output $\Sigma_{\mathbf{Y}}$.
3. Produce the Cholesky factor M of $\Sigma_{\mathbf{Z}}$ so that $MM' = \Sigma_{\mathbf{Z}}$.
4. Set X by $X \leftarrow MZ$.
5. Return Y where $Y_i \leftarrow F_{Y_i}^{-1}[\Phi(X_i)],\ i = 1, 2, ..., d$.

With modern parallelized computing, steps 1, 3, 4, 5 are readily implemented as high-performance (multicore and/or graphical-processing-unit (GPU) acceleration) algorithms, providing the greatest scaleability in a practical manner.

Matching specified Pearson correlation coefficients exactly (step 2 above), however, is problematic. 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 $EY_iY_j = \int\int y_i y_j f_{X|r}(F_i^{-1}(\Phi(z_i)), F_j^{-1}(\Phi(z_j))dy_i dy_j$, for $i, j = 1, 2, \ldots, d$. For high-dimensional data, these evaluations are often too costly to enable feasible comprehensive MC studies. For low-dimensional problems, methods and tools exist to match exactly (see Chen (2001); Xiao (2017); Madsen and Birkes (2013)), including the publicly available `nortaRA` R package.

Our solution is to essentially avoid this complication in Pearson matching. Since our goal is to simulate non-normal marginals, we greatly prefer the use of rank-based measures ρ_S and τ from a modeling standpoint. Further, ρ_S and τ invariance under monotone transformation (see Background), perserves the correlation coefficients through steps 3, 4, and 5 in the NORTA algorithm above. This eliminates the need for computing the $\binom{d}{2}$ integrals to match exactly (nothing is for free, however, as discussed in below in Section 3.2).

Despite all this, if one does desire to characterize dependency using Pearson correlations, simply using the target Pearson correlation matrix as the initial conditions to our proposed algorithm will lead to approximate matching in the resultant distribution (Song (2000)). The quality of this approximation depends on the modeling goal, but in practice, for high-dimensional count data we find the accuracy to be adequate. Later, we'll study the robustness of our method to this limitation in selected Monte Carlo evalutations.

3.2 Random vector generation via `bigsimr::rvec()`

Now we describe `bigsimr::rvec()`, our algorithm to generate random vectors. It mirrors the classical NORTA algorithm above with some modifications for rank-based dependency matching:

1. Pre-processing for nonparameteric dependency matching.
   (i) Convert from either $\mathbf{R}_{\mathbf{Spearman}}$ or $\mathbf{R}_{\mathbf{Kendall}}$ into the corresponding MVN input correlation $\mathbf{R}_{\mathbf{Pearson}}$.
   (ii) Check that $\mathbf{R}_{\mathbf{Pearson}}$ is semi-positive definite.
   (iii) If not compute a close semi-positive $\widetilde{\mathbf{R}}_{\mathbf{Pearson}}$.
2. Gaussian copula construction.
   (i) Generate $\mathbf{X} = (X_1, \ldots, X_d) \sim N_d(\mathbf{0}, \mathbf{R}_{\mathbf{Pearson}})$;
   (ii) Transform $\mathbf{X}$ to $\mathbf{U} = (U_1, \ldots, U_d)$ viz $U_i = \Phi(X_i)$, $i = 1, \ldots, d$;
3. Quantile evaluations. Return $\mathbf{Y} = (Y_1, \ldots, Y_d)$, where $Y_i = F_i^{-1}(U_i)$, $i = 1, \ldots, d$;

The pre-processing (Step1) takes advantage of the closed form relationships between ρ_S and τ with ρ_P for bivariate normal random variables via Equations (4) or (3), respectively (implemented as `bigsimr::cor_covert()`).

A complication often arises at this stage: the parallelized pairwise conversions may create a (spuriously) non-positive definite correlation matrix, especially in high dimensions. Researchers working in multivariate computation frequently encounter such difficulties and need to find a closest positive definiteness matrix. The most widely used routine for this task in R is called `matrix::nearPD()` and is not suitable for high dimensions. To overcome this issue, we've implemented `bigsimr::nearPSD()`, a quadractically convergent Netwon method for finding the nearest correlation matrix, developed by Qi and Sun (2006). We hope that this routine could be useful in many applications aside from our primary goal of random vector generation.

Once the target margins and algorithm inputs are determined, steps 2 and 3 are essentially a NORTA algorithm with modern high-performance computing implementations. Specifically, step 2i uses either an efficient multicore multivariate normal simulator `mvnfast`(Fasiolo 2016) or a using Google's JAX python library NumPy for graphical-processing-unit (GPU) acceleration of the Cholesky factorization and matrix multiplication (steps 3, 4 in the NORTA algorithm in the preceeding section). (JAX is a not acronym and can be thought of a high-performance NumPy).

4 The `bigsimr` R package

The `bigsimr` package is a high-performance implementation of the proposed random vector generation algorithm and associated functions (see Section 3 for details). When designing `bigsimr`, we aimed to conveniently provide parallelized computation, through multi-core and GPU acceleration.

This section describes basic use of `bigsimr`, by stepping through a low-dimensional (2D) simulation workflow for the often-used example data set `airquality`. This workflow proceeds from data, to estimation, simulation configuration, random vector generation, and result vizualization. For this low-dimensional setting, we compute using a single central processing unit (CPU) as the overhead in forking the tasks to multiple cores outways the computational gains. Then we transition to advanced use where we briefly describe some of the high-performance features and syntax. The section concludes with a short description of how to use `bigsimr` on a computer clusters through slurm scheduling via the `rslurm` package.

4.1 Basic use illustrated through a minimal example

We'll demonstrate the basic use and syntax of `bigsimr` through an example workflow applied to 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 syntacically-pleasing `tidyverse` suite of `R` packages.

```
library(bigsimr)
library(tidyverse)
library(patchwork)
```

For simplicity and to provide a minimal working example, we'll consider bivariate simulation of temperature, in degrees Fahrenheit, and ozone level, in parts per billion.

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

```
Rows: 116
Columns: 2
$ Temp  <int> 67, 72, 74, 62, 66, 65, 59, 61, 74, 69, 66, 68, 58, 64, 66, 5...
$ Ozone <int> 41, 36, 12, 18, 28, 23, 19, 8, 7, 16, 11, 14, 18, 14, 34, 6, ...
```

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, particulary the

ozone level exhibits a strong left skew.

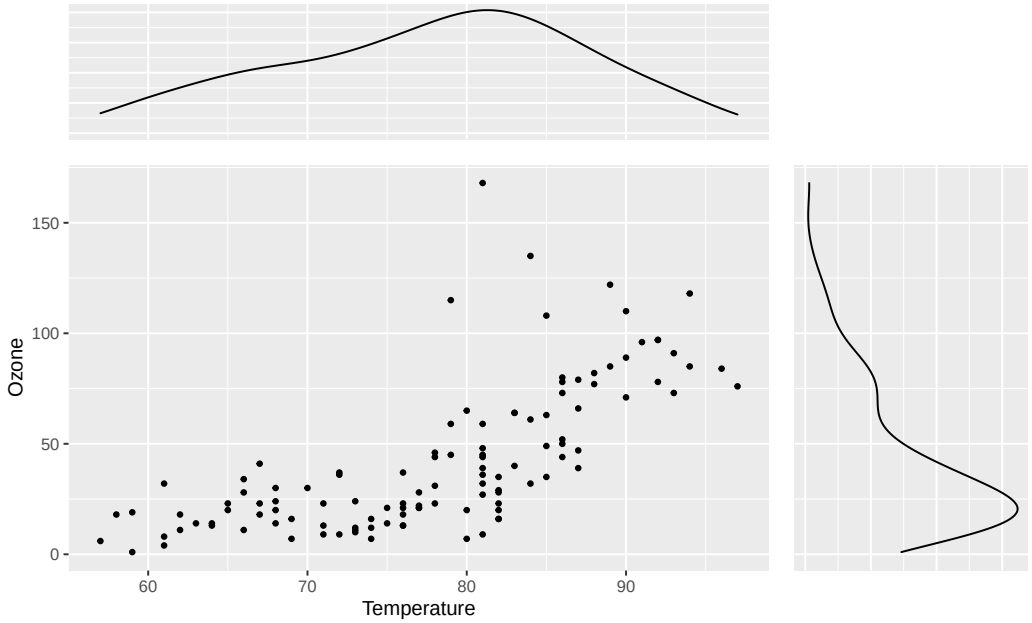

Figure 2: Bivariate scatterplot of Ozone vs. Temperature with estimated marginal densities. The data are left skewed tails and appear to be correlation.

Next we must specify the marginal distributions and correlation coefficient (type and magnitude). Here the analyst is free to be creative. For this example, we will not take up goodness-of-fit considerations. But it seems sensible without domain knowledge to estimate these quantities from the data and we have a suite of functions designed for this task.

Specifying marginal distributions. Of course, we don't know the true distribution of ozone levels or air temperatures. Based on the estimated densities in Figure 2, we'll assume `Temp` is normally ditsributed and `Ozone` is log-normally distributed since its values are positive and skewed. We'll use the well-known unbiased estimators for normal distribution and maximum likelihood estimators for the lognormal parameters:

```
df %>%
  select(Temp) %>%
  summarise_all(.funs = c(mean = mean, sd = sd))
   mean    sd
1 77.87 9.485
```

```
mle_mean <- function(x) mean(log(x))
mle_sd <- function(x) mean( (log(x) - mean(log(x)))^2 )

df %>%
  select(Ozone) %>%
  summarise_all(.funs = c(meanlog = mle_mean, sdlog = mle_sd))
  meanlog  sdlog
1   3.419 0.7426
```

Next, we'll configure the input marginals later input into `bigsimr::rvec`. The marginal distributions are specifying using `R`'s special `alist` function. This allows one to enter the distributions without evaluating anything (yet).

```
margins <- alist(
    qnorm(mean = 77.871, sd = 9.4855),
    qlnorm(meanlog = 3.419, sdlog = 0.7426),
)
```

Notice that we use the *quantile* function for the marginals, as that is how the marginal distributions F_i enter into the `bigsimr::rvec` algorithm. This implementation strategy supports all `R base` probability distributions. And allows flexible extensions using other `R` packages that adhere to conventions, such as `extraDistr` Further, by using `alist`, users can specify their own custom distributions (see below in creating custom margins).

It is a bit inconvenient to have to fill in the parameter values manually each time, so we provide a convenience function called `mlist` which behaves similarly to `alist`, except that it will evaluate the right hand side of argument values within the list. This is intended to help when scaling up your code to high dimensions.

```
margins <- mlist(
  qnorm(mean = mean(df$Temp), sd = sd(df$Temp)),
  qlnorm(meanlog = mle_mean(df$Ozone), sdlog = mle_sd(df$Ozone))
)
margins
[[1]]
qnorm(mean = 77.8706896551724, sd = 9.48548563759966)

[[2]]
qlnorm(meanlog = 3.41851510081201, sdlog = 0.742588880315024)
```

Specfying correlation. Again the user must decide how to describe correlation, based on the particulars of the problem. For non-normal data and for improved simulation accuracy in our scheme, we advocate the use of rank-based correlations Spearman's ρ_S and Kendall's τ. But we also support approximate Pearson correlation coefficient matching, while cautioning the user to check the performance for their parameteric multivariate model (see Monte Carlo evaluations for evaluation strategies). To aid in correlation specification, and estimation in general, we provide a high-performance function `bigsimr::cor_fast` which can estimate Pearson, Spearman, or Kendall correlation using the fastest methods available. (Anyone who has tried estimating Kendall's τ using `stats::cor` can attest that the routine does not scale to even moderate dimensions). Notably, these estimation methods are the standard approaches, not designed specifically designed for high-dimensional correlation estimation (see Conclusion and Discussion for more on this).

```
type <- 'spearman'
(rho <- cor_fast(df, method = "spearman"))
       Temp Ozone
Temp  1.000 0.774
Ozone 0.774 1.000
```

Checking the theoretical correlation bounds As discussed in Section 2, given a pair of marignal distributions the possible correlations are not free to vary between $[-1, 1]$. To ensure that the simulation is not configured to impossible settings, we provide the `bigsimr::cor_bounds` fuction provides Monte Carlo estimated theoretical lower and upper bounds (using the Generate, Sort, and Correlate algorithm of Demirtas and Hedeker (2011)).

```
cor_bounds(margins = margins, type = type)
$lower
     [,1] [,2]
[1,]    1   -1
[2,]   -1    1

$upper
     [,1] [,2]
[1,]    1    1
```

```
[2,]    1    1
```

Since our estimated Spearm correlation $\hat{\rho}_S$ is within the theoretical bounds, the correlation is valid as input to `bigsimr:rvec`. By comparison, the Pearson correlation ρ_P is restricted to [-0.8648, 0.8727] for these margins (see `bigsimr::cor_bounds`output below).

```
cor_bounds(margins = margins, type = 'pearson')
$lower
        [,1]    [,2]
[1,]  1.0000 -0.8699
[2,] -0.8699  1.0000

$upper
       [,1]   [,2]
[1,] 1.0000 0.8677
[2,] 0.8677 1.0000
```

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

```
x <- rvec(10000, rho, margins, type)
df_sim <- as.data.frame(x)
colnames(df_sim) <- colnames(df)
```

Figure 3 plots the 10,000 simulated points.

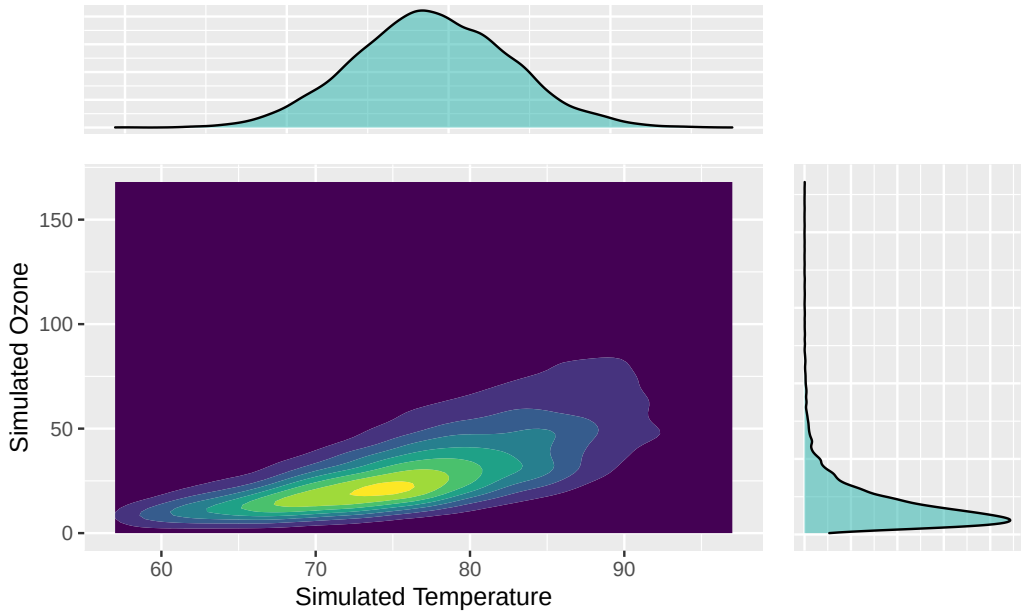

Figure 3: Contour plot and marginal densities for the simulated bivariate distribution of Air Quality Temperatures and Ozone levels. The simulated points mimick the observed data with respect to both the marginal characteristics and bivariate association.

4.2 Advanced use

Creating a custom marginal distribution. Because `bigsimr` uses an `alist` to store the margins, any kind of probability distribution can be used including custom marginal distributions not provided in base `R`. This

accommodation is possible because the main function `rvec` uses the inverse CDF of a margin to transform from correlated uniform margins to the target distribution. Therefore, to specify a marginal distribution absent from an existing packge, the user simply needs to provide a closed form expression from the corresponding quantile function.

It is important to follow `R`'s naming convention for probability distributions. Users should prefix distributions with `r` and `q` for *random* and *quantile* respectively. Only the quantile function is necessary to simulate random vectors, but to compute the theoretical correlation bounds, it is required to supply univariate random number generator as well.

For an example, let's provide a custom *Pareto* distribution for use with `bigsimr`. The Pareto CDF is

$$F(x) = 1 - \left(\frac{x_m}{x}\right)^{\alpha}$$

for scale $x_m > 0$ and shape $\alpha > 0$, with support $x \in [x_m, \infty)$. From the CDF, we compute the inverse CDF

$$F^{-1}(p) = \frac{x_m}{(1-p)^{1/\alpha}}$$

Next we define the Pareto quantile function in R.

```
qpareto <- function(p, scale, shape) {
  scale / (1 - p)^(1/shape)
}
```

For example, writing random number generating function for our target marginal can be accomplished by calling the quantile function on a uniformly distributed random variable (the inverse transform method Rizzo (2007)).

```
rpareto <- function(n, scale, shape) {
  qpareto(runif(n), scale, shape)
}
```

Now with the quantile and RNG Pareto functions, we can use the distribution in `bigsimr` just like the other built-in distributions.

```
margins <- alist(
  qnorm(mean = 3.14, sd = 0.1),
  qbeta(shape1 = 1, shape2 = 4),
  qnbinom(size = 10, prob = 0.75),
  qpareto(scale = 1.11, shape = 5.55)
)
cor_bounds(margins, "pearson")
rho <- cor_randPD(4)
x <- rvec(10, rho, margins)
```

Using `bigsimr` on a computing cluster via `rslurm`. Though `bigsimr` runs quickly, at large d users may want to run jobs on a shared computing server. The R package `rslurm` makes it easy to run embarrassingly large parallel `rvec` calls. This example assumes that `bigsimr` is installed on a system with a slurm scheduler installed.

```
library(rslurm)
sjob <- slurm_call(rvec, jobname = 'rvec',
                   alist(n=1e6,
                         rho = rho,
                       margins = margins,
```

```
                        type = "spearman"),
                    submit = TRUE)
```

Now, let's show off the real power of combining `bigsimr` and `rslurm` by simulating many correlation structures. The `rslurm::slumr_map` syntax mirrors the familiar `base::lapply` and `purrr::map` functions.

```
library(rslurm)
simReps <- 100
rhoList <- replicate( n = simReps, bigsimr::cor_randPSD(d = length(margins)), simplify=FALSE )
sjob <- slurm_map(x = rhoList,
                  f = rvec,
                  jobname = 'rvecMap',
                  n=1e6,
                  margins = margins,
                  type = "spearman",
                  nodes = 4,
                  cpus_per_node = 4,
                  cores = 4,
                  submit = TRUE)
```

On a cluster carrying 24 nodes with 48 threads, these 100 jobs completed in about a minute.

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. And, so, we'll focus the numerical experiments on assessing how well the procedure scales to high dimension with respect to reasonable computation times (property S1 above) and accurately m atching marginal and dependency parameters.

Further, we select simulaiton configurations to ultimately simulate discrete-valued RNA-seq data and the simulations will proceed in increasing complexity, leading to our motivating application in Section 6. We begin wiith empirically evaluating the dependency matching across all three supported correlations — Pearson's, Spearman's, and Kendall's —- in identical, bivariate marginal configurations. The simulations progress from bivariate normal, to bivariate gamma (non-normal yet continuous), and bivariate negative binomial (mimicking RNA-seq counts).

5.1 Bivariate experiments

Bivariate Normal. Let's simulate a bivariate normal and check our correlation matching performance as N increases. Standard bivariate normal Here we have BVN($\mu_1 = \mu_2 = 0, \rho_{type}$). We vary ρ across the entire possible range of correlations for each correlation type. Figure 4 `bigsimr` recovers the Pearson specified correlations for MVN.

Bivariate Gamma. Similarly, let's check the performance for a non-symmetric continuous distribution: a standard (rate =1) bivariate gamma. Here we have a Bivariate Gamma with $shape_1 = shape_2 = 10, \rho_{type}$. We vary ρ across the entire possible range of correlations for each correlation type. Figure 5 `bigsimr` recovers the Pearson specified correlations for bivariate gamma.

Bivariate Negative Binomial. Let's check the performance for a discrete distribution: a bivariate negative binomial. We use values that are motivated by our RNA-seq data, namely the mean probability and sizes estimated from the data (see Example applications for our motivating data for estimation details): Bivariate Negative Binomial ($prob_1 = prob_2 = 3 \times 10^{-4}, size_1 = size_2 = 4, \rho_{type}$). We vary ρ across the entire possible range of correlations for each correlation type. Figure 6 `bigsimr` recovers the Pearson specified correlations for bivariate negative binomial.

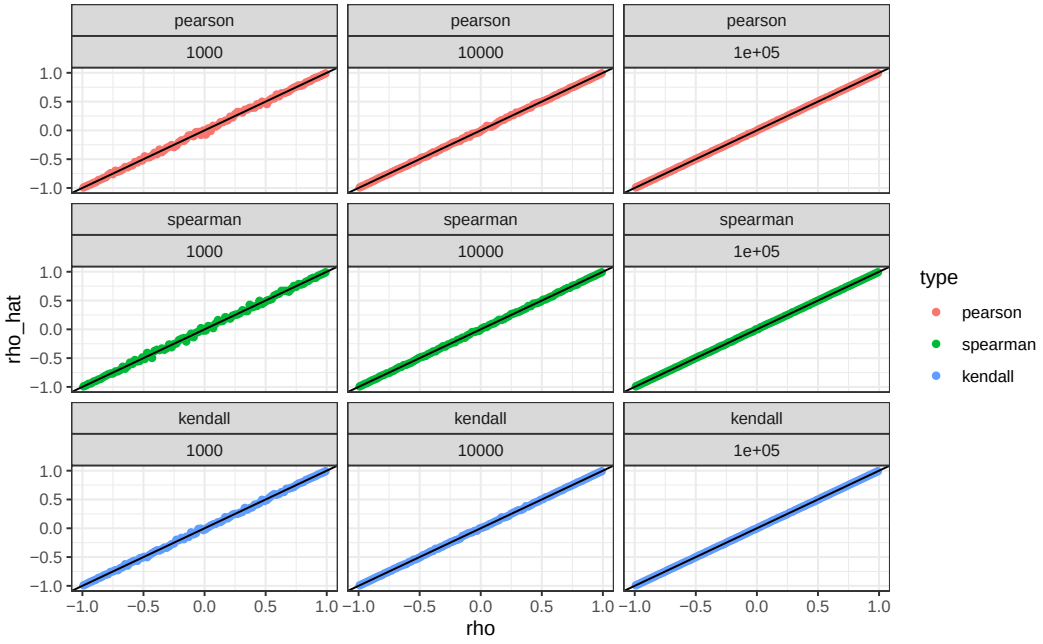

Figure 4: 'bigsimr' recovers the Pearson specified correlations for MVN.

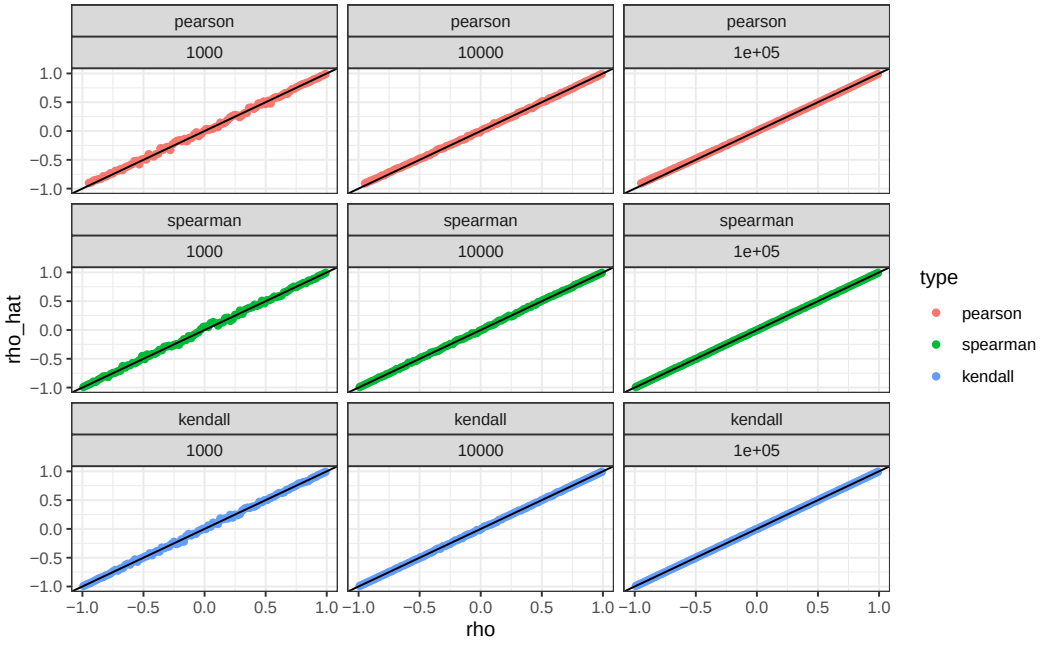

Figure 5: 'bigsimr' recovers the Pearson specified correlations for Bivariate Gamma.

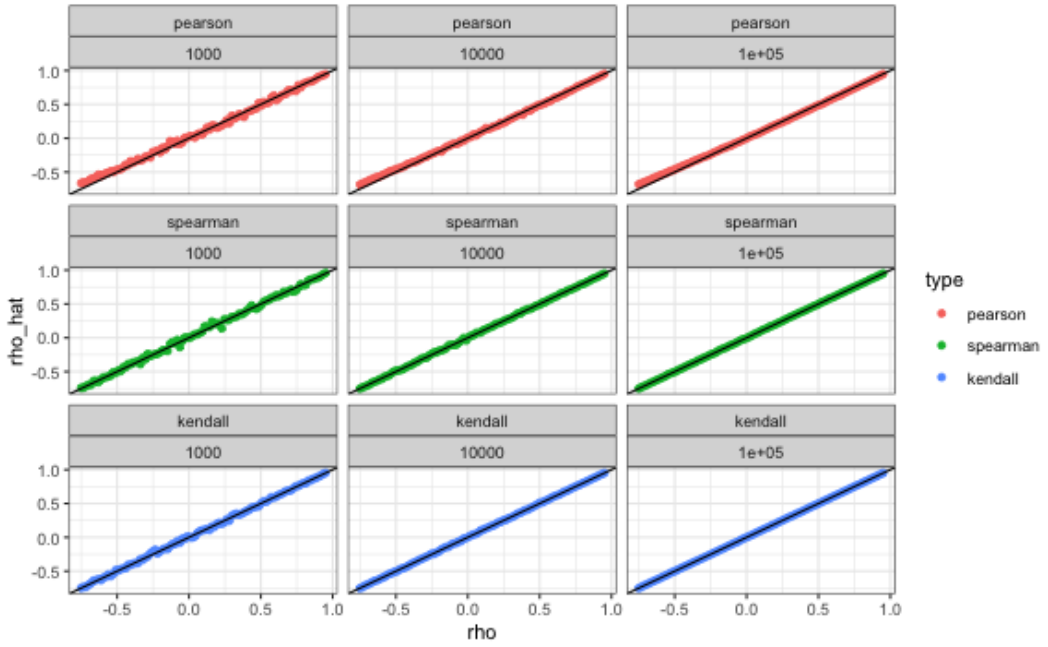

Figure 6: 'bigsimr' recovers the correlations for bivariate negative binomial only approximately for Pearson but (nearly) exactly for the rank-based correlations.

Taken together, the studies show our methology are accuracy across the entire range of possible correlation values for the rank-based dependency measures in continuous marginals. For the two non-normal bivariate marginals, the Pearson correlation matching is approximate. Table XX contains the average absolute error (MAE) in reproducing the desired dependency measures for the three bivariate scenarios. For discrete margins, matching the dependency measures was somewhat less accurate, even for the rank-based metrics, especially near the correlation bounds. Since the bivariate normal exactly matches all dependency measures in our scheme, we also report relative MSE of the non-normal cases to the normal simulations.

Generally, the accuracy seems adequate for most applications, at least in these limited simulation settings. Of course, one could choose marginals that lead to poor matching by construction. In partice, we recommend users to always evaluate the accuracy for their application, using methods similar to those presented above. See Discusion for future directions for fast Pearson matching and discrete-specific modifications.

5.2 Scale up to Ultra-High Dimensions

With some assurance of the method's accuracy from the low-dimensional perspective in the bivariate setting, we now turn to assessing whether the `bigsimr` can scale to large dimesional problems. Now we seek evidence to determine whether the method can scale to higher dimensions in a practical time. Using our motivating RNA-seq data, described in Background, we filtered the orignal 20,501 genes to the high-expressing genes at increasing percentiles, $1, 5, 10, 15, 20, 25\%$, to obtain $d = \{206, 1026, 2051, 3076, 4101, 5127\}$ marginals and $\binom{d}{2}$ pairwise correlations at each setting. For example, for $d = 5127$ there are $13,140,501$ correlation coefficients. We estimated the marginal negative binomial parameters and the correlation coefficients from the RNA-seq data to seed our simulations. (See Example applications for our motivating data for a detailed descrition of estimation). Figure 7 displays computation times using various high-performance settings to produce $B = 10,000$ random vectors. The Pearson simulations are much faster since the correlation conversion steps are avoided (pre-processing step; see Algorithms), but as we known from above the accuracy will suffer slightly, especially near the negative boundary of the possible correlations. Matching Spearman's correlation at larger d gets costly if one wants to produce $B = 10,000$ random vectors at many different simulation settings. We recommend computing this type of simulations on a computing cluster (possibly using `rslurm`;

Table 2: Average abolute error in matching the target dependency and

N	type	margins	mae
1000	kendall	norm	0.0111
1000	kendall	gamma	0.0122
1000	kendall	nbinom	0.0118
1000	spearman	norm	0.0182
1000	spearman	gamma	0.0169
1000	spearman	nbinom	0.0187
1000	pearson	norm	0.0151
1000	pearson	gamma	0.0217
1000	pearson	nbinom	0.0306
10000	kendall	norm	0.0040
10000	kendall	gamma	0.0033
10000	kendall	nbinom	0.0041
10000	spearman	norm	0.0056
10000	spearman	gamma	0.0056
10000	spearman	nbinom	0.0065
10000	pearson	norm	0.0055
10000	pearson	gamma	0.0120
10000	pearson	nbinom	0.0198
1e+05	kendall	norm	0.0012
1e+05	kendall	gamma	0.0011
1e+05	kendall	nbinom	0.0011
1e+05	spearman	norm	0.0017
1e+05	spearman	gamma	0.0018
1e+05	spearman	nbinom	0.0020
1e+05	pearson	norm	0.0017
1e+05	pearson	gamma	0.0102
1e+05	pearson	nbinom	0.0186

see the Advance Use).

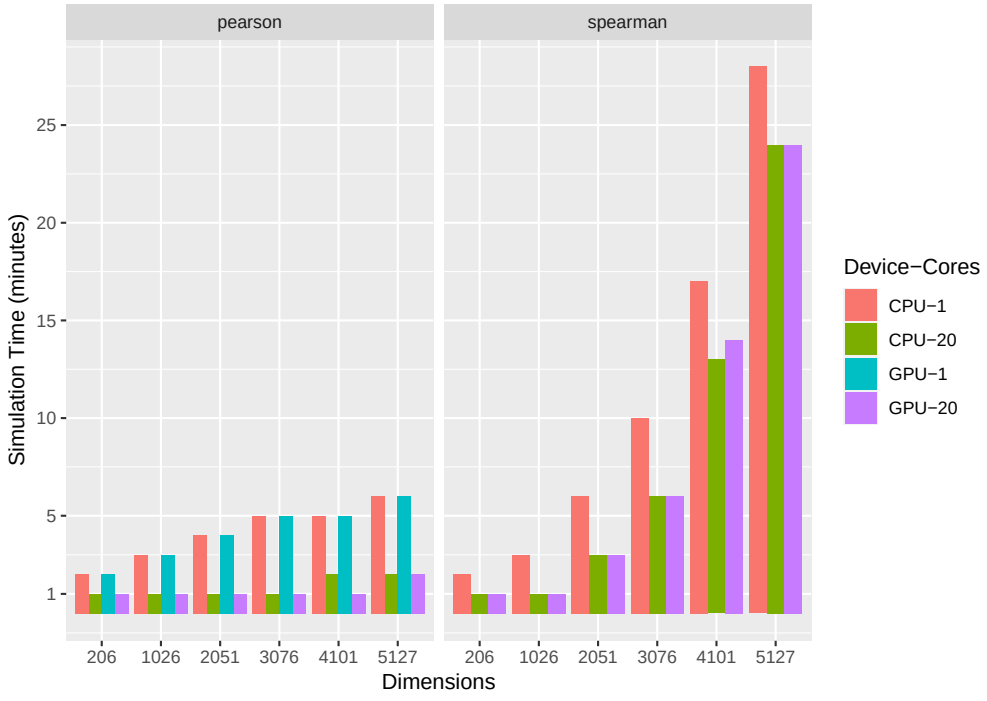

Figure 7: Computation times as d increases. We filter to the top 1, 5, 10, 15, 20, 25% expressing genes (in terms of median expression.)

6 Example applications for RNA-seq data

This section demonstrates how to simulate multivariate data using `bigsimr`, aiming to replicate the structure of high-dimensional dependent count data. Simulating RNA-sequencing (RNA-seq) data is a primary motivating application of the proposed methodology, seeking scaleable Monte Carlo (MC) methods for realistic multivariate simulation for these data. Intergene correlation is an inherent part of biological processes. Yet many models do not account for this, leading to major distruptions to the operating characteristics of statistical estimation, testing, and prediction. See Efron (2012) for a detailed discussion with related methods and see Wu and Smyth (2012), Schissler, Piegorsch, and Lussier (2018), Schissler et al. (2019) for applied examples. The following subsections apply `bigsimr`'s methods to real RNA-seq data, including replicated an estimated parameteric structure, MC probability estimation, and MC evaluation of correlation estimation efficiency.

6.1 Simulating High-Dimensional RNA-seq data

In an illustration of our proposed methodology applied to real data, we seek to simulate RNA-sequencing data by producing simulated random vectors with assumed marginal distributions with estimated parameters. Our goal is to replicate the structure of a breast cancer data set (BRCA data set from The Cancer Genome Atlas). Specifically, we will simulate $B = 10,000$ random vectors $\mathbf{Y} = (Y_1, \ldots, Y_d)^\top$ All these genes exhibit over-dispersion and, so, we proceed to estimate the NB parameters $(r_i, p_i), i = 1, \ldots, d$ to determine the target marginal PMFs $g_i(y_i)$ (via method of moments). Often researchers posit a negative binomial (NB) model as RNA-seq counts are often over-dispersed that a Poisson model would suggest. This suggests simulating high-dimensional multivariate NB (MVB) with heterogeneous marginals would be useful tool in the development and evaluation of RNA-seq analytics. This procedure results in count data with infinite support, since RNA-sequencing platforms measure gene expression by enumerating the number of reads

aligned to genomic regions. To specify the simulation algorithm inputs, we estimate the Spearman correlation matrix $\mathbf{R}_{\mathbf{Y}}^{\mathbf{Spearman}}$ and marginal NB parameters.

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

```
## Estimate Spearman's correlation on the count data
corType <- 'spearman'
system.time( nb_Rho <- bigsimr::cor_fast( brca, method = corType ) )
   user  system elapsed
  0.057   0.005   0.061
```

Next we estimate the marginal parameters. Here we use the basic method of moments (MoM) to estimate the marginal parameters for the multivariate negative binomial model. We model the marginals distributions as coming from the same probability family (NB) yet are hetereogeneous in terms of the parameters probability and size (p_i, n_i) for $i, \ldots, d$. The code below features some advanced `R` utilities for concise, rapid, and generalizable target marginal specification:

```
make_nbinom_alist <- function(sizes, probs) {
  lapply(1:length(sizes), function(i) {
    substitute(qnbinom(size = s, prob = p),
               list(s = sizes[i], p = probs[i]))
  })
}
## make_nbinom_alist(c(20, 21, 22), c(0.3, 0.4, 0.5))
nbinom_mom <- function(x) {
  m <- mean(x)
  s <- sd(x)
  s2 <- s^2
  p <- m/s2
  r <- m^2 / (s2 - m)
  c(r, p)
}
```

Once the functions are defined to complete marginal estimation, we specify the desired multivariate negative binomial distribution and generate the desired random vectors:

```
sizes <- apply( unname(as.matrix(brca)), 2, nbinom_mom )[1, ]
probs <- apply( unname(as.matrix(brca)), 2, nbinom_mom )[2, ]
```

Notably, the marginal NB probabilities $\hat{p}_i's$ are small — ranging in $[6.4313 \times 10^{-6}, 0.0023]$. 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`:

```
## Set the number of random vectors
n <- 10000
## construct margins
nb_margins <- make_nbinom_alist(sizes, probs)
## run sims
sim_nbinom <- rvec(n, nb_Rho, nb_margins, type = corType,
                   ensure_PSD = TRUE, cores = cores)
colnames(sim_nbinom) <- names(brca)
```

Figure 8 displays the simulated counts and pairwise relationships for our example genes in 1. Simulated counts roughly mimick the observed data but with a smoothier appearance due to the assumed parameter form.

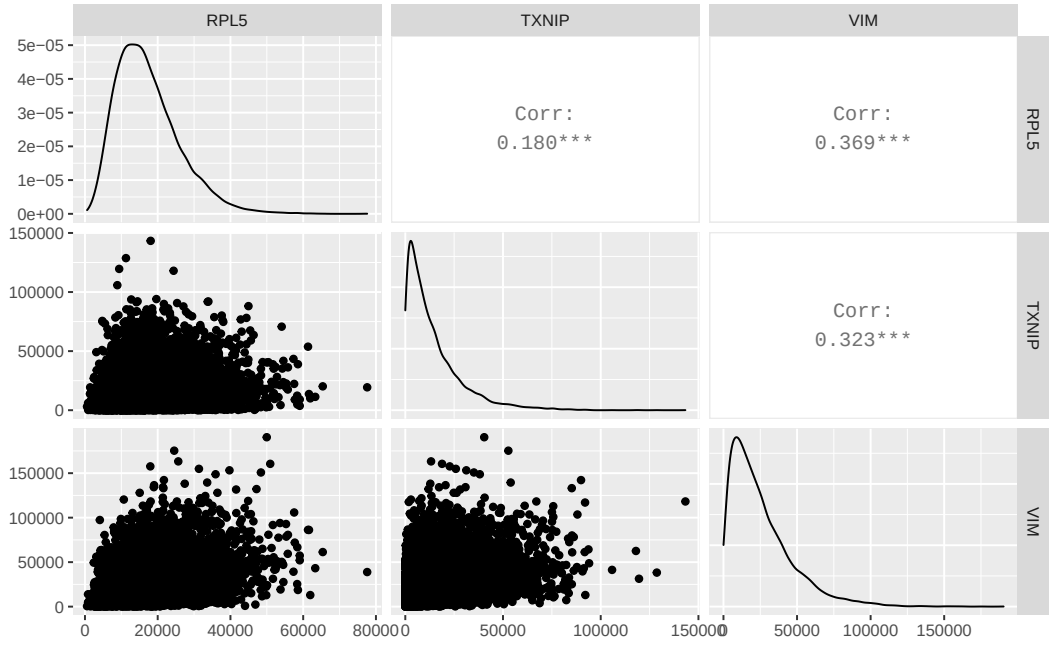

Figure 8: Simulated data for 3 selected high-expressing genes, replicating the observed data structure.

Figure 9 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. Utilizing 15 CPU threads in a MacBook Pro carrying a 2.4 GHz 8-Core Intel Core i9 processor, the simulation completed just shy of 7.5 minutes.

6.2 Simulation-based UHD joint probability calculations

To conduct statistical inference a critical task is to evaluate the 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 hetereogeneous, correlated marginal distributions (or margins of mixed data types). Given a large number 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 $\mathbf{y}_0$.

Then we estimate

$$\hat{P}(\mathbf{Y} >= \mathbf{y_0}) = \sum_{b=1}^{B} I(\mathbf{Y^{(b)}} > \mathbf{y_0})/B$$

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

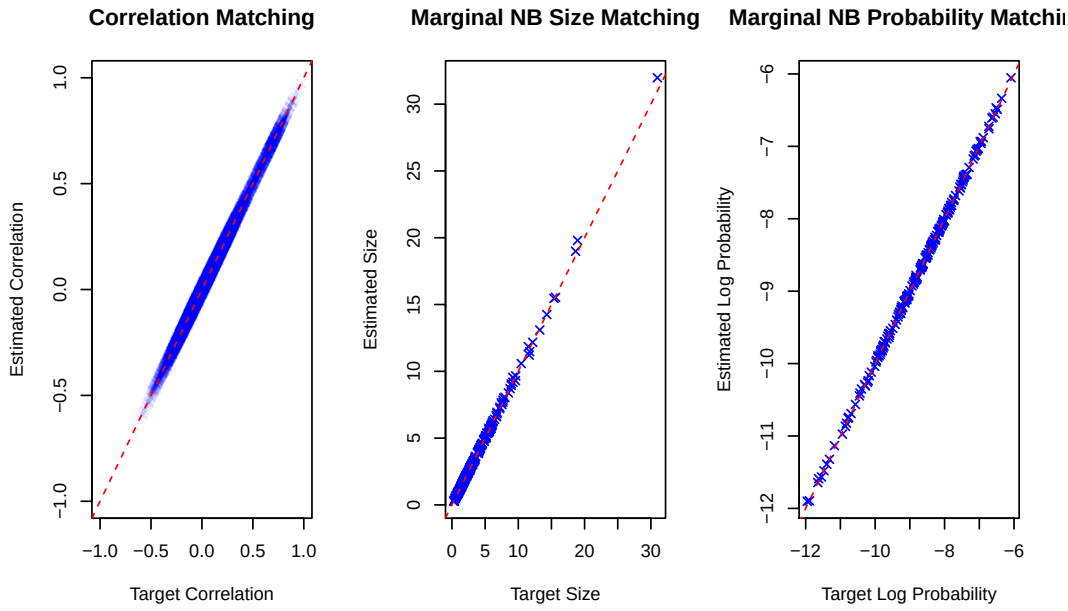

Figure 9: 'bigsimr' produces simulated random vectors from a multivariate negative binomial that replicate the estimated structure from an RNA-seq data set. The dashed red lines indicated equality between estimated parameters from simulated data(vertical axes) and the specified target parameters (horizontal axes).

```
d <- ncol(sim_nbinom)
B <- nrow(sim_nbinom)
threshold <- rep( 1, d)
mean(apply( sim_nbinom,  1,
           function(X, y0=threshold) {
               all( X > y0) }
           ))
[1] 0.7551
```

6.3 MC 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. For an concise introduction to these methods, see, for example Rizzo (2007), Ch. 6. Now, we demonstrate how `bigsimr` can be used evaluate estimation efficiency. In particular, we'd like to assess the error our correlation estimation above. We used a conventional method, based on classical statistical theory. Yet this method was not designed for high-dimensional data. Indeed, high-dimensional covariance estimation (and precision matrices) is an active area of statistical science (see, for example, Won et al. (2013) and Van Wieringen and Peeters (2016)). In this example, we simulate $m = 100$ data sets with the number of simulated vectors matching the number of patients in the BRCA data set, $N = 1212$. At each iteration we estimate the quadratic loss from the specified $\mathbf{R}_{\mathbf{Y}}^{\mathbf{Spearman}}$ and average over the m iteration.

```
quadLossRDS <- paste0( "results/brca", round(myProb*100), "quadLoss.rds" )
if ( !file.exists( quadLossRDS) )  {
    n <- nrow(brca)
    simRho <- replicate(n = m, expr = { tmpSim <- rvec(n = nrow(brca) , nb_Rho, nb_margins,
                                                       type = corType, ensure_PSD = TRUE, cores = cores)
                                        tmpRho <- bigsimr::cor_fast( x = tmpSim )} ,
```

```
                        simplify = FALSE)

    quadLoss <- unlist( lapply( simRho, rags2ridges::loss, T = nb_Rho, type = "quadratic"  ) )
    saveRDS(quadLoss, quadLossRDS)
}
quadLoss <- readRDS( quadLossRDS )
```

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

```
summary(quadLoss)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
    495     543     572     575     609     681       2
```

This distribution could be compared to more elaborate, high-dimensional estimators to guide analysts in deciding whether the additional complexity and computation time are warranted for their application.

7 Conclusion and discussion

We've introduced a general-purpose high-dimensional multivariate simulation algorithm and provide a high-performance implementation called **bigsimr**. The high-performance — efficient, multi-core and GPU enabled — algorithms and software provide The random vector generation method is largely 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 our work presented here involve high-dimensional scalability, model flexibility, and high-performance implementation with broad potential data analytic applications for modern big data challenges.

It is usually customary to compare new tools and algorthms directly to existing competing methods. In our MC studies, however, we only employ our proposed methodology, since our previous work has show that existing tools are simply not designed or feasible to meet our HD goal (see Li et al. (2019) for evalutions 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. The most obvious missing feature the proposed methodology is the inability to match a Pearson correlation matrix exactly. As discussed in Algorithms and extensively by Xiao and Zhou (2019), this is a computational intense procedure and not a natural choice for characterizing dependency for non-normal marginals, especially in the high-dimensional setting. While we do not provide an implementation directly supporting Pearson matching, user can always supply their own input Pearson after using a supplementary matching scheme (Cario and Nelson 1997; Xiao and Zhou 2019).

Future work includes developing scalable algorithms to match the Pearson correlation matrix more precisely, discrete-margin specific modifications including fast Spearman's correlation rescaling (see Equation (5)), and *bona fide* high-dimensional correlation estimation (for example, see Won et al. (2013)). From an implementation standpoint, **bigsimr** only supports Nvidia GPUs and refactoring the code using OpenCL would broaden the user base. As problems grow even larger, multi-GPU support is a logical progression in this highly parallelization algorthmic scheme and warrant further development.

8 Supplementary Materials

We provide an open-source implementation of our methology as the **bigsimr** R package, hosted on github, https://schisslergroup.github.io/bigsimr/.

9 Acknowledgement(s)

The authors gratefully acknowledge the helpful discussions with University of Arizona's Professor Walter W. Piegorsch and Professor Edward J. Bedrick during this project's conception. We also 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.

10 Disclosure statement

The authors report no conflict of interest.

11 Funding

Research reported in this publication was supported by MW-CTR-IN of the National Institutes of Health under award number NIH 1U54GM104944.

Barbiero, Alessandro, and Pier Alda Ferrari. 2017. "An R package for the simulation of correlated discrete variables." *Communications in Statistics - Simulation and Computation* 46 (7): 5123–40. https://doi.org/10.1080/03610918.2016.1146758.

Cario, Marne C., and Barry L. Nelson. 1997. "Modeling and generating random vectors with arbitrary marginal distributions and correlation matrix." Citeseer. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.48.281{\&}rep=rep1

Chen, Huifen. 2001. "Initialization for NORTA: Generation of Random Vectors with Specified Marginals and Correlations." *INFORMS Journal on Computing* 13 (4): 312–31.

Conesa, Ana, Pedro Madrigal, Sonia Tarazona, David Gomez-Cabrero, Alejandra Cervera, Andrew McPherson, Michal Wojciech Szcześniak, et al. 2016. "A survey of best practices for RNA-seq data analysis." https://doi.org/10.1186/s13059-016-0881-8.

Demirtas, Hakan, and Donald Hedeker. 2011. "A practical way for computing approximate lower and upper correlation bounds." *American Statistician* 65 (2): 104–9. https://doi.org/10.1198/tast.2011.10090.

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

———. 2012. *Large-Scale Inference: Empirical Bayes Methods for Estimation, Testing and Prediction*. 1st ed. Cambridge: Cambridge University Press. http://statweb.stanford.edu/{~}ckirby/brad/LSI/monograph{_}CUP.pdf.

Fasiolo, Matteo. 2016. *An introduction to mvnfast. R package version 0.1.6.* https://cran.r-project.org/package=mvnfast.

Kruskal, William H. 1958. "Ordinal Measures of Association." *Journal of the American Statistical Association* 53 (284): 814–61. https://doi.org/10.1080/01621459.1958.10501481.

Li, Bo, and Colin N. Dewey. 2011. "RSEM: Accurate transcript quantification from RNA-Seq data with or without a reference genome." *BMC Bioinformatics*. https://doi.org/10.1186/1471-2105-12-323.

Li, Xiang, A. Grant Schissler, Rui Wu, Lee Barford, Jr. Harris, Fredrick C., and Frederick C. Harris. 2019. "A Graphical Processing Unit Accelerated NORmal to Anything Algorithm for High Dimensional Multivariate Simulation." *Advances in Intelligent Systems and Computing*, 339–45. https://doi.org/10.1007/978-3-030-14070-0_46.

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

Mari, Dominique Drouet, and Samuel Kotz. 2001. *Correlation and dependence*. World Scientific.

Nelsen, Roger B. 2007. *An Introduction to copulas*. 2nd ed. New York: Springer Science & Business Media.

Nikoloulopoulos, Aristidis K. 2013. "Copula-based models for multivariate discrete response data." In *Lecture Notes in Statistics: Copulae in Mathematical and Quantitative Finance*, 213th ed., 231–49. Heidelberg: Springer.

Park, Chul Gyu, Taesung Park, and Dong Wan Shin. 1996. "A Simple Method for Generating Correlated Binary Variates." *American Statistician*. https://doi.org/10.1080/00031305.1996.10473557.

Qi, Houduo, and Defeng Sun. 2006. "Computing the A Quadratically Convergent Newton Method For Computing The Nearest Correlation Matrix." *SIAM Journal on Matrix Analysis and Applications* 28 (2): 360–85.

Rizzo, Maria L. 2007. *Statistical Computing with R*. https://doi.org/10.1201/9781420010718.

Schissler, A Grant, Walter W Piegorsch, and Yves A Lussier. 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. https://doi.org/10.1177/0962280217712271.

Schissler, Alfred Grant, Dillon Aberasturi, Colleen Kenost, and Yves A. Lussier. 2019. "A Single-Subject Method to Detect Pathways Enriched With Alternatively Spliced Genes." *Frontiers in Genetics* 10 (414). https://doi.org/10.3389/fgene.2019.00414.

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*.

Song, Peter Xue-kun. 2000. "Multivariate Dispersion Models Generated from Gaussian Copula." *Scandinavian Journal of Statistics* 27 (2): 305–20.

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

Van Wieringen, Wessel N., and Carel F.W. Peeters. 2016. "Ridge estimation of inverse covariance matrices from high-dimensional data." *Computational Statistics and Data Analysis*. https://doi.org/10.1016/j.csda.2016.05.012.

Wang, Zhong, Mark Gerstein, and Michael Snyder. 2009. "RNA-Seq: A revolutionary tool for transcriptomics." https://doi.org/10.1038/nrg2484.

Won, Joong-Ho, Johan Lim, Seung-Jean Kim, and Bala Rajaratnam. 2013. "Condition-number-regularized covariance estimation." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 75 (3). Blackwell Publishing Ltd: 427–50. https://doi.org/10.1111/j.1467-9868.2012.01049.x.

Wu, Di, and Gordon K. Smyth. 2012. "Camera: A competitive gene set test accounting for inter-gene correlation." *Nucleic Acids Research* 40 (17). Oxford University Press: e133–e133. https://doi.org/10.1093/nar/gks461.

Xiao, Qing. 2017. "Generating correlated random vector involving discrete variables." *Communications in Statistics - Theory and Methods*. https://doi.org/10.1080/03610926.2015.1024860.

Xiao, Qing, and Shaowu Zhou. 2019. "Matching a correlation coefficient by a Gaussian copula." *Communications in Statistics - Theory and Methods* 48 (7): 1728–47. https://doi.org/10.1080/03610926.2018.1439962.

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