Simulating High-Dimensional Multivariate Data

using the bigsimr R Package

true

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. 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 and notation

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 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, where d can be very large (possibly millions of variables).

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: 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 is (approximately) 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`, we the next section describes our motivating data that originally inspired the authors' interest in creating this methodology.

2.1 Motivating example: RNA-seq data from breast cancer tumor samples

Simulating high-dimensional non-normal data motivates this work — in pursuit of modeling RNA-sequencing data (Wang, Gerstein, and Snyder 2009; Conesa et al. 2016) derived from breast cancer patients. This laboratory 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. Often researchers posit a negative-binomial (NB) model as counts are often over-dispersed that a Poisson model would suggest. Yet due to inherent biological processes, gene expression data exhibits correlation (co-expression) across genes (Efron 2007; Schissler, Piegorsch, and Lussier 2018).

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

Figure X.

2.2 Measures of dependency

The population correlation coefficient, commonly called the Pearson (product-moment) correlation coefficient, describes the linear association between two random variables X and Y and is given by

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

As Madsen and Birkes (2013) and Mari and Kotz (2001) discuss, for a bivariate normal (X, Y) random vector the Pearson correlation adequately describes the dependency between the components. ρ's utility in non-normal or non-linear associations is lacking (Mari and Kotz (2001)). Rank-based (ordinal) approaches performance better in these settings, such as Spearman's (denoted ρ_s) and Kendall's τ. Define

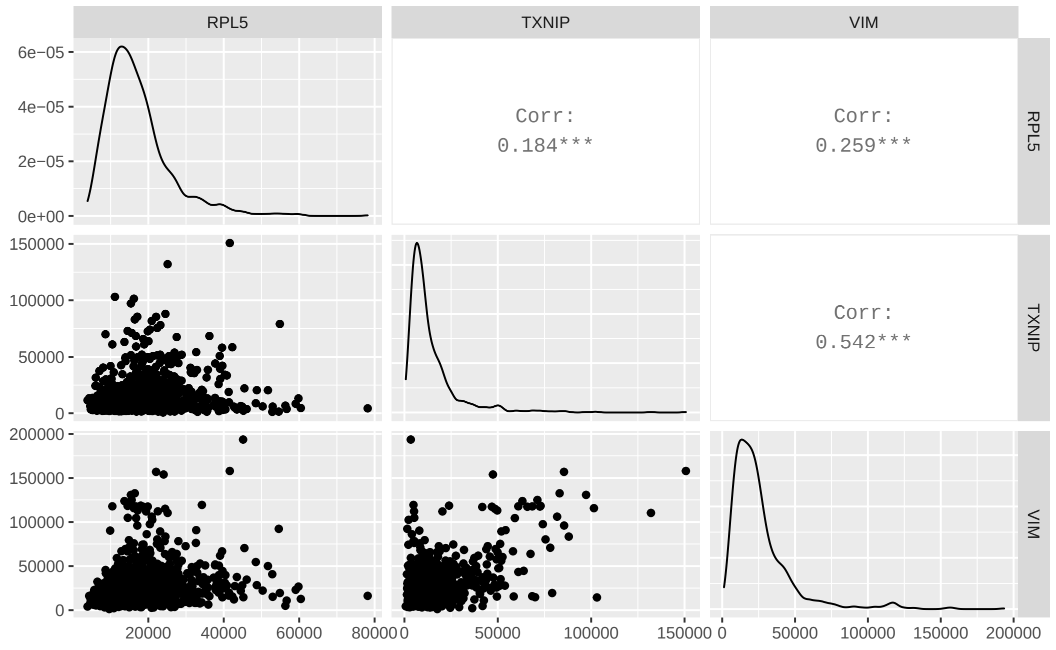


Figure 1: Real data for the 3 selected high-expressing genes

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

where $(X_1, Y_1) \overset{d}{=} (X, Y), X_2 \overset{d}{=} X, Y_3 \overset{d}{=} Y$ with X_2 and Y_3 are independent of one other and of (X_1, Y_1). For continuous marginals, the measure works well due to zero probability of ties.

To do this, we take advantage of a closed form relationship [ref?] between Kendall's τ and Pearson's correlation coefficient for bivariate normal random variables:

$$r_{Pearson} = sin\left(\tau_{Kendall} \times \frac{\pi}{2}\right), \tag{3}$$

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

$$\rho_{Pearson} = 2 \times sin\left(\rho_{Spearman} \times \frac{\pi}{6}\right). \tag{4}$$

Marginal-dependent bivariate correlation bounds. Adding to the complexity is the well-known, marginal-dependent constraints on the possible bivariate correlation. This is a consequence of the bounds of bivariate distribution functions — the *Frechet-Hoeffding bounds* (Nelsen 2007)). Denote the pointwise upper bound as $M(y_i, y_{i'})$ These bounds give strict restraints on the possible bounds and cannot be overcome through algorithm design. Yet not all multivariate simulation approaches obtain these bounds [for example REF]. Computationally, algorithms that rely on serial computation scale poorly to high dimensions (refs - hierarchy) and fail to make use of modern high-performance computing including parallelization and graphical processing unit acceleration (Li et al. 2019).

The pairwise correlation between two correlated random variables cannot in general obtain the full range of possible values, $[-1, -1]$. The range, called the Frechet(-Hoeffding) bounds, is a well-known function of the marginal distributions and are given by (Barbiero and Ferrari 2017):

$$\rho^{max} = \rho\left(F_1^{-1}(U), F_2^{-1}(U)\right), \quad \rho^{min} = \rho\left(F_1^{-1}(U), F_2^{-1}(1-U)\right) \tag{5}$$

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 (8).

For discrete marginals, however, one could perform a rescaled version of ρ_s for random variables X, Y with pmfs (or pdfs) $p(x)$ and $q(y)$, respectively.

$$\rho_{RS}(X,Y) = \frac{\rho_s(X,Y)}{\left[[1-\sum_x p(x)^3]\left[1-\sum_y q(y)^3\right]\right]^{1/2}} \tag{6}$$

2.3 Gaussian copulas

The crux of the method is construction of a Gaussian copula. This idea is well known [refs]. We chose a Gaussian copula among the many available copula functions primarily for two reasons: 1) the computationally convient closed expression to convert among the most common dependency measures when the margins are Gaussian and 2) the availability of high-performance multivariate normal simulators in R ('mvnfast' ref).

A copula is a distribution function on $[0,1]^d$ describing a random vector with standard uniform marginals. Moreover, for any random vector $\mathbf{X} = (X_1, \ldots, X_d)$ with cumulative distribution function (CDF) F and marginal CDFs F_i there is a copula function $C(u_1, \ldots, u_d)$ so that

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

A Gaussian copula is the case where all marginal CDFs F_i are the standard normal cdf, Φ. The Gaussian copula is the one that corresponds to a multivariate normal distribution with standard normal marginal distributions and covariance matrix $\mathbf{R}$. (Since the marginals are standard normal, this $\mathbf{R}$ is also the correlation matrix). If $F_{\mathbf{R}}$ is the CDF of such 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 simply the CDF of the random vector $(\Phi(X_1), \ldots, \Phi(X_d))$, where $(X_1, \ldots, X_d) \sim N_d(\mathbf{0}, \mathbf{R})$.

Sklar's Theorem (ref) guarantee's that any random vector with computational feasible inverse CDFs (P4 above) and obtainable correlation matrix (within the Frechet bounds) can be obtained via transformations involving copula functions. Namely, to simulate a random vector $\mathbf{Y} = (Y_1, \ldots, Y_d)$, where $Y_i = F_i^{-1}(U_i)$, $i = 1, \ldots, d$, we can construct a Gaussian copula $\mathbf{U} = (U_1, \ldots, U_d)$ viz $U_i = \Phi(X_i)$, $i = 1, \ldots, d$. When an Y_i is discrete, some care must be taken to define F_i. Letting

$$F_i^{-1} = inf\{y : F_i(y) \ge u\} \tag{8}$$

ensures that $Y_i \sim F_i$.

Simulation of a general multivariate random vector $\mathbf{T}$ with margins F_i based on this copula is quite simple. Ensuring that one obtains a certain dependence among the marginal distributions, however, proves challenging in our proposed Gaussian-copula-based scheme. Two core issues: 1) marginal characteristics induce bounds on the possible bivariate correlations and 2) monotone transformations deform the Pearson correlation which no closed form expression exists in general (ref? Chen2001). The Frechet bounds are well-known.

3 Algorithms

This section describes the method for simulating a random vector $\mathbf{Y}$ with Y_i components for $i = 1, 2, \ldots, d$. Each Y_i has a specified marginal distribution function F_i and its inverse. To characterize dependency, every pair (Y_i, Y_j) has either a specified Pearson correlation (1), (rescaled) Spearman correlation , or Kendall's τ). The method only approximately matches the Pearson correlation in general, whereas the rank-based methods are exact.

The method is best understand as a **parallelized Gaussian copula** (see Equation (7) to provide a high-performance NORTA (NORmal To Anything) algorithm.

`mvnfast` High-performance multivariate normal simulator (Fasiolo 2016)

`nortaRA` Implements exact Pearson matching (step 2 in NORTA) (Chen 2001)

To simulate a random vector $\mathbf{Y}$ with variance-covariance matrix $\Sigma_{\mathbf{Y}}$, there is the well-known NORTA algorithm (Cario and Nelson 1997). It follows like this:

1. Simulate a random vector $\mathbf{Z}$ with d **independent** and **identical** standard normal components.
2. Determine the input matrix $\Sigma_{\mathbf{Z}}$ to corresponds with the specified output $\Sigma_{\mathbf{Y}}$ (Chen 2001; Xiao 2017)
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 $Y_i \leftarrow F_{Y_i}^{-1}[\Phi(X_i)],\ i = 1, 2, ..., d$.

We shall see that constructing continuous joint distributions that match a target Spearman or Kendall's correlations computes easily when employing Gaussian copulas, since this measures are invariant under the monotone transformations involved [refs].

In contrast the rank-based correlations, matching specified Pearson correlation coefficients exactly is computational intense in this scheme. In general, there is no closed form correspondence and involving computing 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 $i, j = 1, 2, \ldots, d$ (see Xia17 and MB13). For accurate numeric approximation of these integrals, the functions must be evaluated hundreds of times. Others have used efficient Monte Carlo integration schemes (see Chen (2001)), but scale poorly to large dimension in reasonable times (property **S2**). Despite all this, if one does desire to characterize dependency using Pearson correlations, we often see in practice — and it is theoretically justified under certain conditions (Song (2000)) — that simply using the target Pearson correlation matrix as the initial conditions to our proposed algorithm will lead to approximate matching in the resultant distribution. We'll study the robustness of our method to this limitation later in the application section (Section BLAH).

Putting the together the facts provided in the equations above, we come the following proposed simulation algorithm to produce a random vector $\mathbf{Y}$ with specified Spearman's correlation and marginal distributions. Note that all computational steps can be parallelized as each operation can be done or either the d marginals or the $\binom{d}{2}$ pairs for the correlation values. Even the generation of multivariate normal random vectors is parallelized through optimized matrix multiplication/decomposition routines.

3.1 Random vector generation `bigsimr::rvec` algorithm

make this algorithm pretty later

1. Preprocessing for nonparameteric dependency matching.
   (i) Convert from either $\mathbf{R_{Spearman}}$ or $\mathbf{R_{Kendall}}$ into the corresponding MVN input correlation $\mathbf{R_{Pearson}}$ via (4) or (3), respectivity.

   (ii) Check that $\mathbf{R_{Pearson}}$ is semi-positive definite.

   (iii) If not find a close semi-positive $\widetilde{\mathbf{R}}_{\mathbf{Pearson}}$ via `cor_nearPSD()`.

2. NORTA transformation
   (1) Generate $\mathbf{X} = (X_1, \ldots, X_d) \sim N_d(\mathbf{0}, \mathbf{R_{Pearson}})$;

   (2) Transform $\mathbf{X}$ to $\mathbf{U} = (U_1, \ldots, U_d)$ viz $U_i = \Phi(X_i)$, $i = 1, \ldots, d$;

3. Return step
   (3) Return $\mathbf{Y} = (Y_1, \ldots, Y_d)$, where $Y_i = F_i^{-1}(U_i)$, $i = 1, \ldots, d$;

3.2 Other high-performance `bigsimr` algorithms

ALEX, briefly discuss other algorithms here

- `cor_bounds()`

- `cor_covert()` make sure that you are clear this is for only MVN.

- `cor_fast()`

- `cor_nearPSD()`

4 The `bigsimr` R package

`bigsimr` is an R package for simulating high-dimensional multivariate data with arbitrary marginal distributions. The efficiency behind generating multivariate samples comes from the ability to utilize a GPU during the generation of multivariate normal samples and the subsequent transformation to uniform samples (see section 3.1). A GPU is not necessary in order to use `bigsimr`, and the speed benefits generally only come when simulating very high dimensional data ($d > 10000$).

The next computational step after obtaining the correlated uniform marginals is the inverse transform into the target marginal distributions. Since this consists of d independent transformations, we utilize parallelization to achieve higher throughput. As with many parallel algorithms, there is overhead associated with forking the task to utilize multiple cores, however cost is negligible compared to the total run-time.

4.1 Basic use

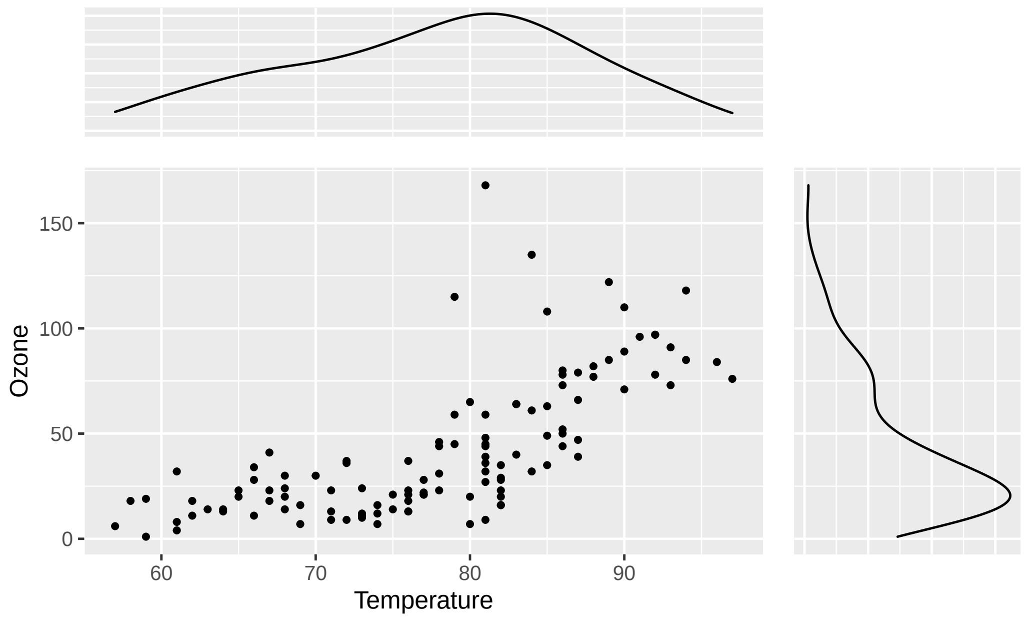
We're going to show the basic use and syntax of `bigsimr` by using the New York air quality data set (`airquality`) included in the R `datasets` package. We will focus specifically on the temperature (degrees Fahrenheit) and ozone level (parts per billion).

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

```r
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, ...
```

Let's look at the joint distribution of the Ozone and Temperature



We can see that not all margins are normally distributed - the ozone level is highly skewed. Though we don't know the true distribution of ozone levels, we can go forward assuming that it is log-normally distributed.

To simulate observations from this joint distribution, we need to estimate the correlation and the marginal parameters.

4.1.1 Estimating Correlation

To estimate the correlation, we can use `R`'s function `cor` or we can use the convenience function `bigsimr::cor_fast` which can estimate Pearson, Spearman, or Kendall correlation using the fastest methods available from other packages.

```
(rho <- cor_fast(df, method = "pearson"))
        Temp  Ozone
Temp  1.0000 0.6984
Ozone 0.6984 1.0000
```

4.1.2 Defining Marginal Distributions

Next we can estimate the marginal parameters. Assuming that `Temperature` is normally distributed, it has parameters

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

and assuming that `Ozone` is log-normally distributed, it has parameters

```
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
```

We need a list of marginals (and their parameters), and a correlation structure (matrix). The marginal distributions can be built up using R's special `alist` function. This allows one to enter the distributions without evaluating anything (yet).

```
margins <- alist(
  qnorm(mean = 77.9, sd = 9.49),
  qlnorm(meanlog = 3.42, sdlog = 0.743)
)
```

Notice that we use the *quantile* function for the marginals, as that is the workhorse of the `rvec` function. By using an `alist`, users can also specify their own custom distributions (see the section on creating custom margins).

- all the base distributions are supported
- one could extend using `extraDistr` .https://cran.r-project.org/web/packages/extraDistr/index.html

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.

```
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)
```

4.1.3 Correlation Bounds

Given a list of margins, the theoretical lower and upper correlation coefficients can be estimated.

```
cor_bounds(margins = margins, type = "pearson")
$lower
        [,1]    [,2]
[1,]  1.0000 -0.8642
[2,] -0.8642  1.0000

$upper
       [,1]   [,2]
[1,] 1.0000 0.8641
[2,] 0.8641 1.0000
```

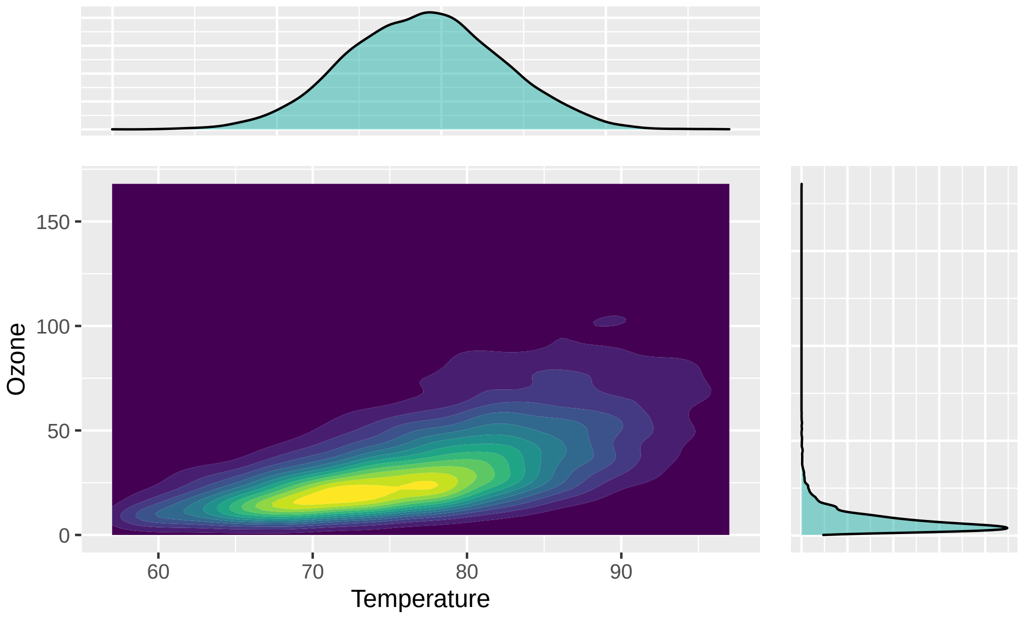
Since our estimated correlation is within the theoretical bounds (assuming the distributions are correct), we should be able to achieve a similar correlation in the simulated data.

4.1.4 Simulating Multivariate Distributions

Let's now simulate 10,000 observations from the joint distribution using `bigsimr::rvec`

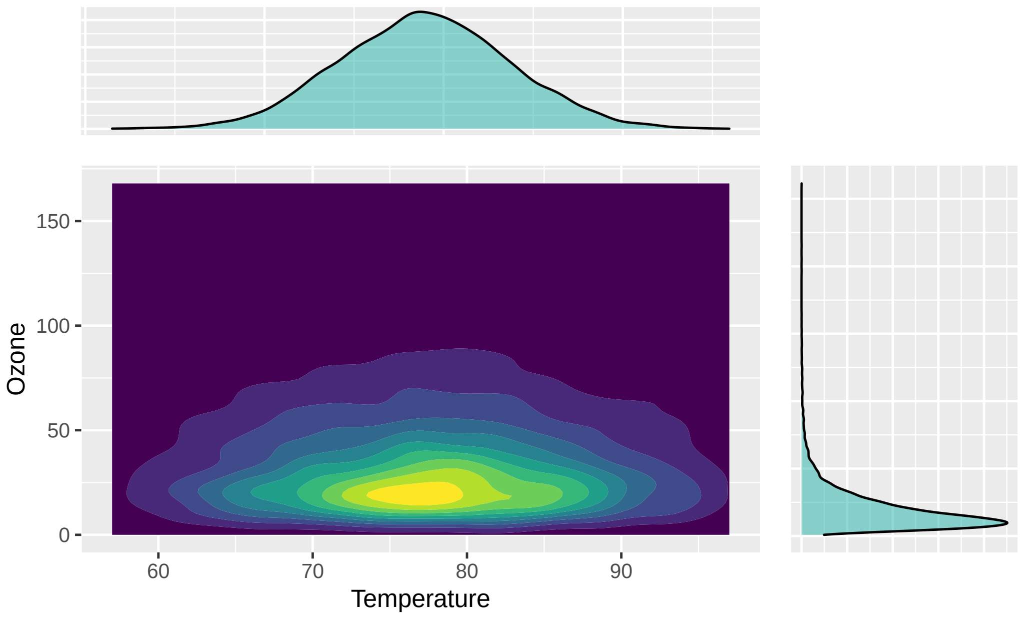
```r
x <- rvec(10000, rho, margins, "pearson")

df_sim <- as.data.frame(x)
colnames(df_sim) <- colnames(df)
```

and visualize the joint distribution



4.1.5 Comparison to Uncorrelated Samples

We can compare the bivariate distribution above to one where no correlation is taken into account.



Notice that this uncorrelated bivariate distribution has the same marginal distributions as the correlated example above.

4.2 Advanced use

4.2.1 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.

It is highly recommended that users follow R's naming convention for probability distributions - I.e. prefixing distributions with `r` and `q` for *random* and *quantile* respectively. Only the quantile function is necessary to generate multivariate data, but if one wishes to compute the theoretical correlation bounds, then one should also specify the random number generator (RNG) function for their custom distribution.

For an example, R does not have any functions for the *Pareto* distribution. We can write a quantile function and RNG function and use it in `rvec`.

The Pareto CDF is defined as

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

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. Note that extra checks should be taken to ensure that the arguments are valid. We omit such checks for clarity.

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

Writing a RNG function for our new distribution can be accomplished by calling the quantile function on the output of a random unit uniform variable.

```r
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.

```r
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)
```

4.2.2 Calculating the nearest positive (semi)definite correlation matrix

4.2.3 Using multicore

4.2.4 Utilizing the GPU

The `bigsimr` packages uses Google's `jax` python library to interface with an Nvidia CUDA enabled GPU.

4.2.5 Simulation-based computation of correlation bounds

4.2.6 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.

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

On a cluster carrying 24 nodes with 48 threads, these 100 jobs completed in about a minute. Let's evaluate the quality of simulation performance.

```
# A tibble: 300 x 4
   simNum   rho rhoHat pair
    <int> <dbl>  <dbl> <fct>
 1      1 0.551  0.551 pair1_2
 2      1 0.551  0.545 pair1_3
 3      1 0.551  0.545 pair2_3
 4      2 0.350  0.349 pair1_2
 5      2 0.350  0.345 pair1_3
 6      2 0.350  0.346 pair2_3
 7      3 0.778  0.778 pair1_2
 8      3 0.778  0.769 pair1_3
 9      3 0.778  0.769 pair2_3
10      4 0.352  0.351 pair1_2
# ... with 290 more rows
```

5 Monte Carlo evaluations

Before applying our methodology to real data simulation, we conduct several Monte Carlo studies to investigate method performance in comparison to other existing implementations. We 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 matching marginal and dependency parameters. The simulations will proceed in increasing complexity — leading up to the setting in our motivating example. We begin by exploring simple exchangeable (constant) correlation structures under a large number of simulation replicates while applying the algorithm above to first continuous then discrete marginal distributions to test the robustness of our algorithm since exact matching is not guaranteed.

5.1 Bivariate experiments

Bivariate Normal. Let's simulate a bivariate normal and check our correlation matching performance as N increases. Here we have BVN($\mu_1 = \mu_2 = 10, \rho_{type}$). We vary ρ across the entire possible range of correlations for each correlation type.

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.

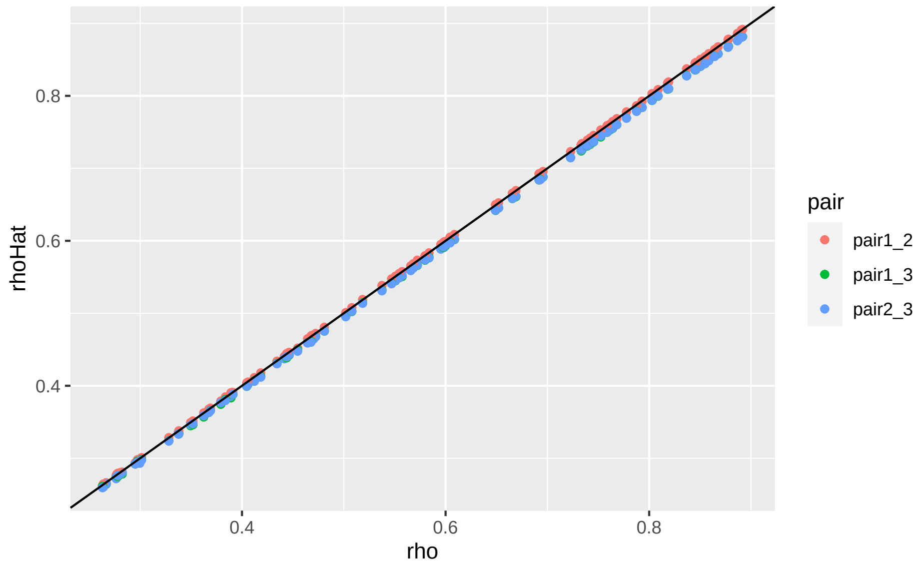


Figure 2: The continous marginal pairs exact reproduce the specified correlation whereas the discrete pairs (green nad blue) show a downward bias. Future work with employ a discrete adjusted procedures using *rescaling* on the traditional correlation coefficient.

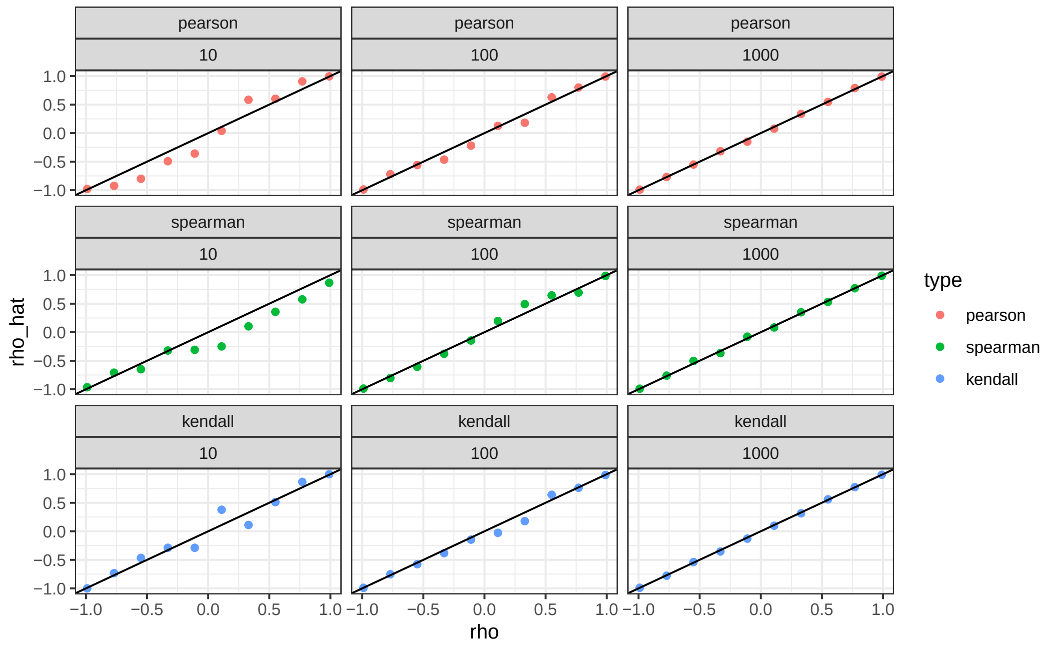


Figure 3: ‘bigsimr‘ recovers the Pearson specified correlations for MVN.

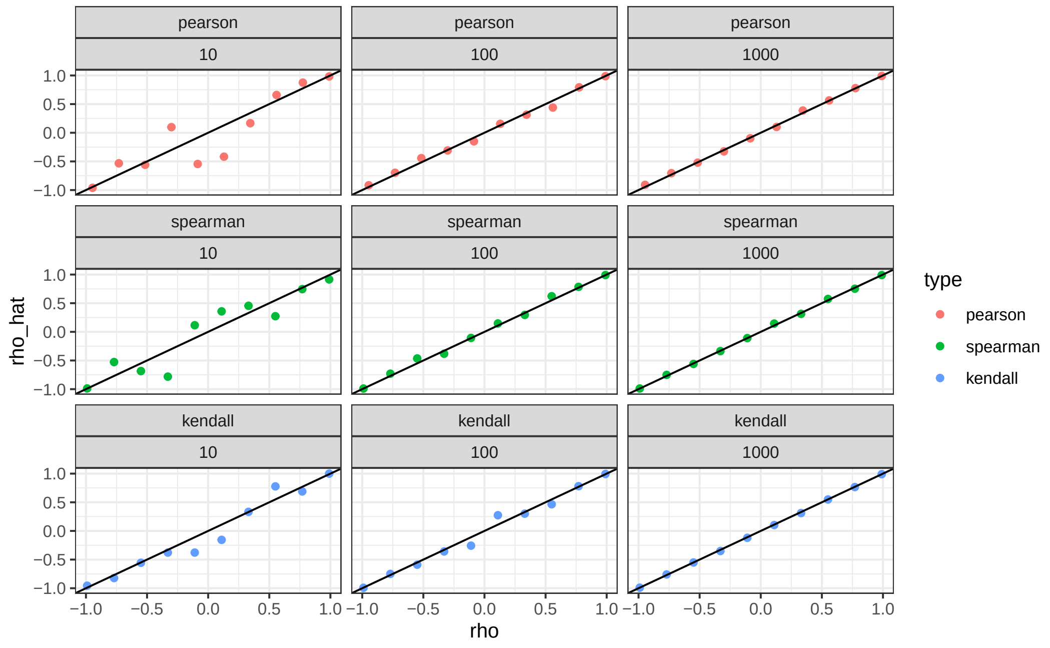


Figure 4: ‘bigsimr‘ recovers the Pearson specified correlations for Bivariate Gamma.

Bivariate Negative Binomial. Let’s check the performance for a discrete distribution: a bivariate negative binomial. Here we have Bivariate Negative Binomial ($prob_1 = prob_2 = 0.5, size_1 = size_2 = 4, \rho_{type}$). We vary ρ across the entire possible range of correlations for each correlation type.

5.2 Scale up to Ultra-High Dimensions

6 Example applications for our motivating data

This section demonstrates how to simulate high-dimensional multivariate data using `bigsimr`, aiming to replicate the structure of high dimensional dependent count data. In fact, simulating RNA-sequencing (RNA-seq) data is a one of the primary motivating applications of the proposed methodology, seeking scaleable Monte Carlo methods for realistic multivariate simulation (for example, see Schissler, Piegorsch, and Lussier 2018).

The RNA-seq data generating process involves counting how often a particular messenger RNA (mRNA) is expressed in a biological sample. Since this is a counting processing with no upper bound, many modeling approaches discrete random variables with infinite support. Often the counts exhibit over-dispersion and so the negative binomial arises as a sensible model for the expression levels (*gene counts*). Moreover, the counts are correlated (co-expressed) and cannot be assumed to behave independently. RNA-seq platforms quantify the entire transcriptome in one experimental run, resulting in high dimensional data. In humans, 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. This suggests simulating high-dimensional multivariate NB with heterogeneous marginals would be useful tool in the development and evaluation of RNA-seq analytics.

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). For simplicity of illustration, we begin by filtering to retain the top 5% highest expressing genes of

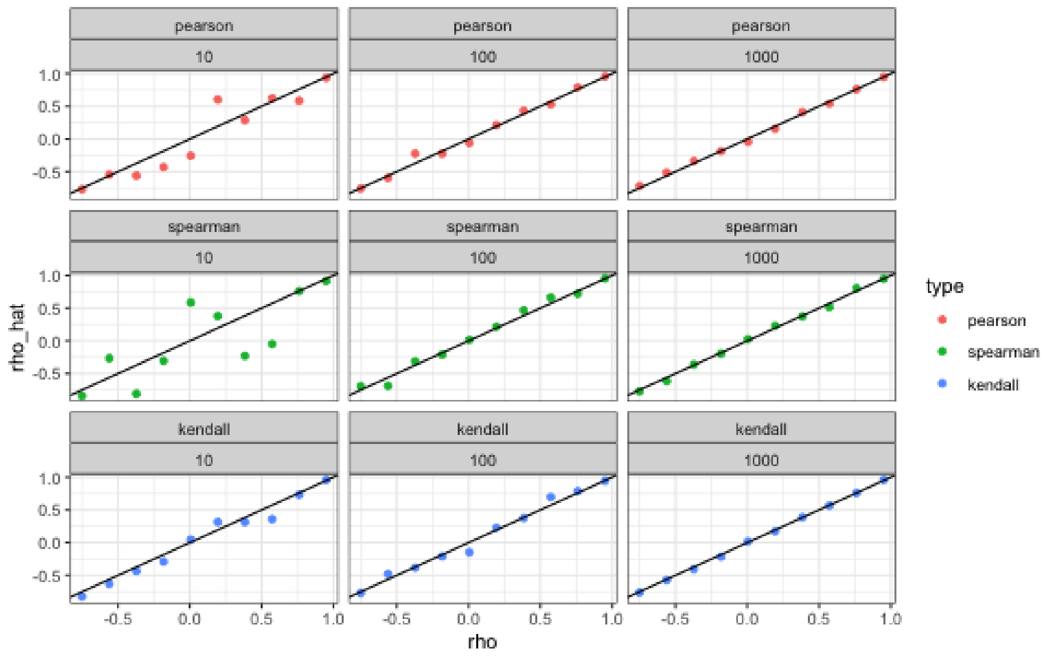


Figure 5: ‘bigsimr‘ recovers the correlations for bivariate negative binomial only approximately for Pearson but (nearly) exactly for the rank-based correlations.

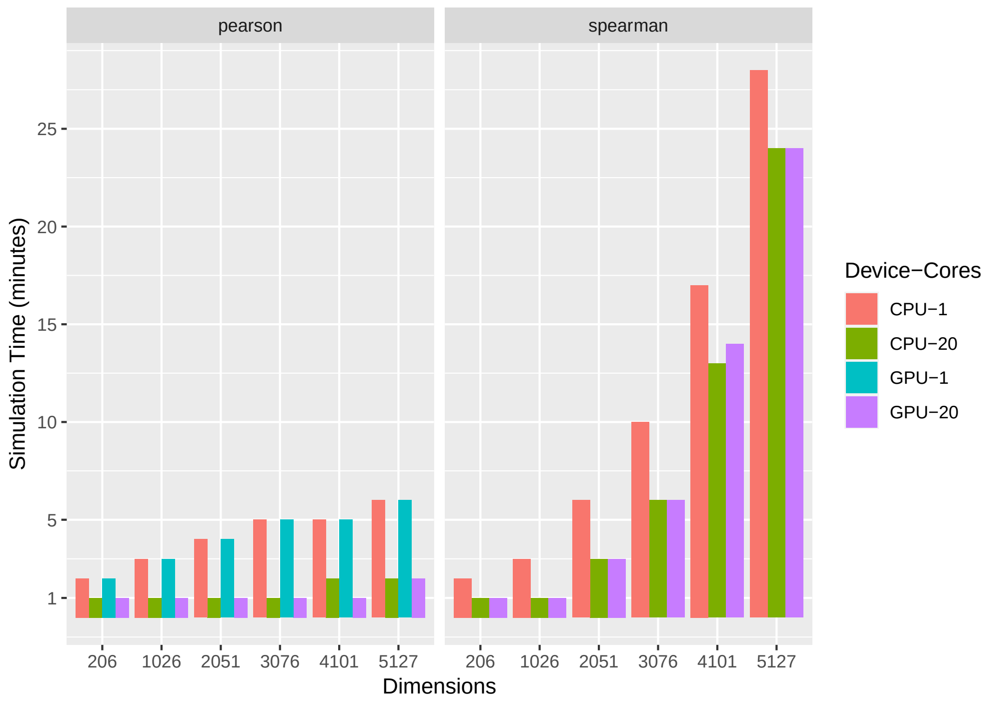


Figure 6: Computation times as d increases. We filter to the top 1, 5, 10, 15, 20, 25% expressing genes (in terms of median expression.)

the 20,501 gene measurements from $N = 1212$ patients' tumor samples, resulting in $d = 1026$ genes.

Let's walk through a workflow simulating RNA-seq data

```
## 1. Estimate Spearman's correlation on the count data
## corType <- 'pearson'
corType <- 'spearman'
system.time( nb_Rho <- bigsimr::cor_fast( brca, method = corType ) )
   user  system elapsed 
  0.051   0.003   0.055 
```

Here we use the basic method of moments (MoM) to estimate the marginal parameters for the multivariate negative binomial model. We model 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 $i, \ldots, d$.

TYPESTE MoM estimators here.

Now specify the desired multivariate negative binomial distribution.

```
## Set the number of random vectors
n <- 10000
## construct margins
nb_margins <- make_nbinom_alist(sizes, probs)
## run sims
system.time( sim_nbinom <- rvec(n, nb_Rho, nb_margins, type = corType,
                                ensure_PSD = TRUE, cores = cores) )
   user  system elapsed 
  1.592   0.326  53.716 
colnames(sim_nbinom) <- names(brca)
```

Figure X.

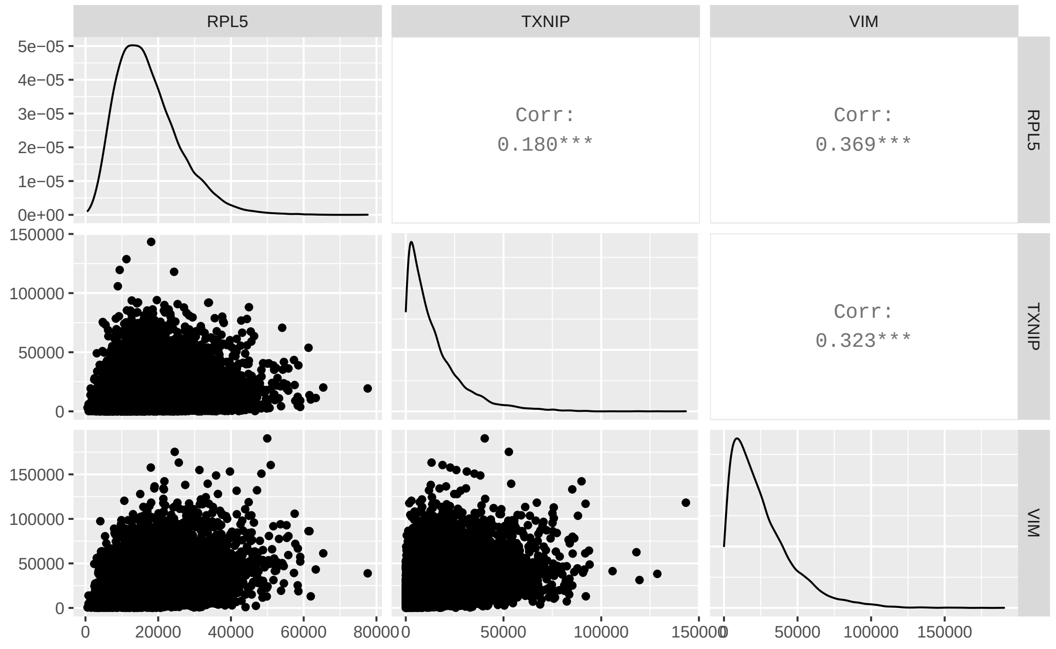


Figure 7: Simulated data for 3 randomly selected high-expressing genes, replicating the real data structure describing in the Introduction.

Figure X shows the 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 in less than 30 seconds.

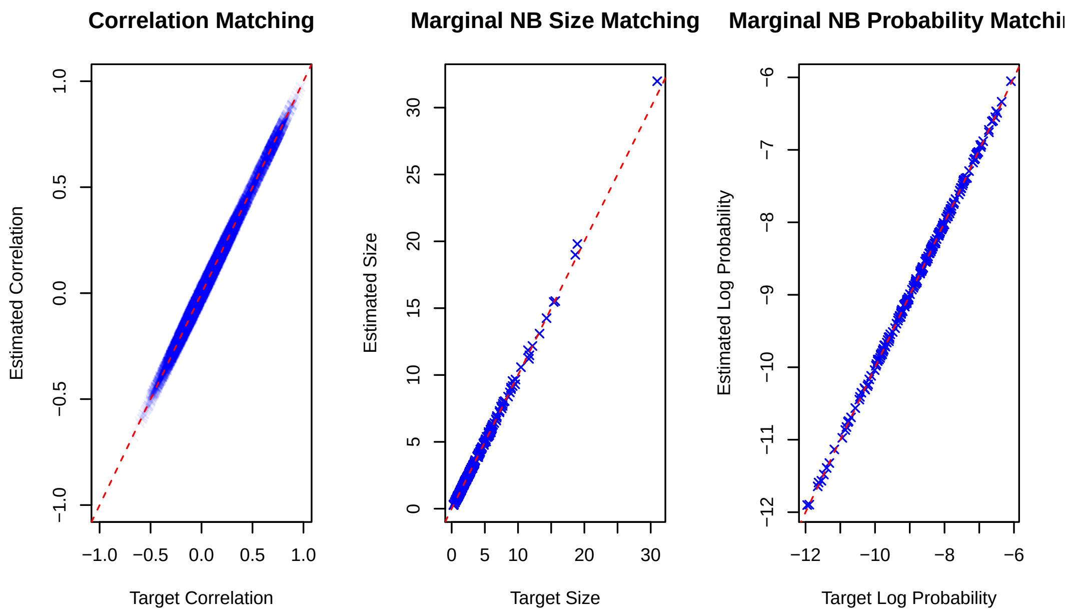


Figure 8: bigsimr produces simulated random vectors from a multivariate negative binomial that replicate the estimated structure from an RNA-seq data set. The dashed red lines indicated equality between estimated parameters (vertical axes; derived from the simulated data) and the specified target parameters (horizontal axes).

6.2 Simulation-based UHD joint probability calculations

To conduct statistical inference, via maximum likelihood for example, a critical task is to evaluate the joint probability mass (or density) function:

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

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

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

where $\mathbf{Y^{(b)}}$ is the b^{th} simulated vector in a total of B simulation replicates and $I(\dot{})$ is the indicator function. For example, we can estimate from our $B = 10,000$ simulated vectors the probability that all genes express more than $\mathbf{y_0} = \mathbf{1000}$.

```r
d <- ncol(sim_nbinom)
B <- nrow(sim_nbinom)
threshold <- rep( 1000, d)
mean(apply( sim_nbinom,  1,
           function(X, y0=threshold) {
               all( X > y0) }
           ))
[1] 0.0368
```

6.3 Monte Carlo estimation of MSE

Monte Carlo (MC) methods offer flexible and straightfoward statistical inferenec in this typically difficult analytic setting. MC methods are routinely used many statistical inferential tasks including estimation, hypothesis testing, error rates, and empirical interval coverage rates. For an concise introduction to these Methos, see, for example Rizzo (2007), Ch. 6. As a simple example, let's now employ `bigsimr` to estimate the mean squared error (MSE) of the method of moments estimators we used in section X.

Specifically, TYPESET THE MATH HERE. REF to MoM above.

```r
## system( "rm results/mseSizes.rds")
if ( !file.exists( 'results/mseSizes.rds') )  {
    n <- nrow(brca)
    m <- 100
    momSizes <- replicate(n = m, expr =
                                     { tmpSim <- rvec(n, nb_Rho, nb_margins, type = corType, ensure_PSD
                                         tmpMom <- apply( unname(as.matrix(tmpSim)), 2, estimateNegBinM
                                         tmpSizes <- unlist( lapply( X = tmpMom, function(x) x$size ) )
                                     }
                          )
    sqDiffSizes <- apply( momSizes, 2, function(x) { (x  - sizes)^2 } )
    mseSizes <- rowMeans( sqDiffSizes )
    saveRDS(mseSizes, 'results/mseSizes.rds')
}
```

Figure X.

One could then compare the MSE of the baisc MoM estimation scheme to other strategies (such as MLE, trimmed approaches, etc.) and decide whether a more computational costly approach is warrented or if the MoM will suffice.

7 Conclusion/Discussion

- Although we recommend to use the rank-based measures
- discuss the omission (or inclusion) of high dimensional covariance estimators?
- We've shown that the multivariate NB simulations outperform the Independent sims. But the agreement is still poor. This could motivate other more appropriate models for high-expressing RNA-seq data than the negative binomial distribution. This algorithm lends itself to exploration of flexible probability models to inspire model selection in the development novel RNA-seq analytic methodology.
- Simulating correlated discrete and count data takes more care (Xiao 2017), including slightly redefining the inverse.
- a user can always supply their own input Pearson after using the existing matching schemes (Xiao and Zhou 2019).

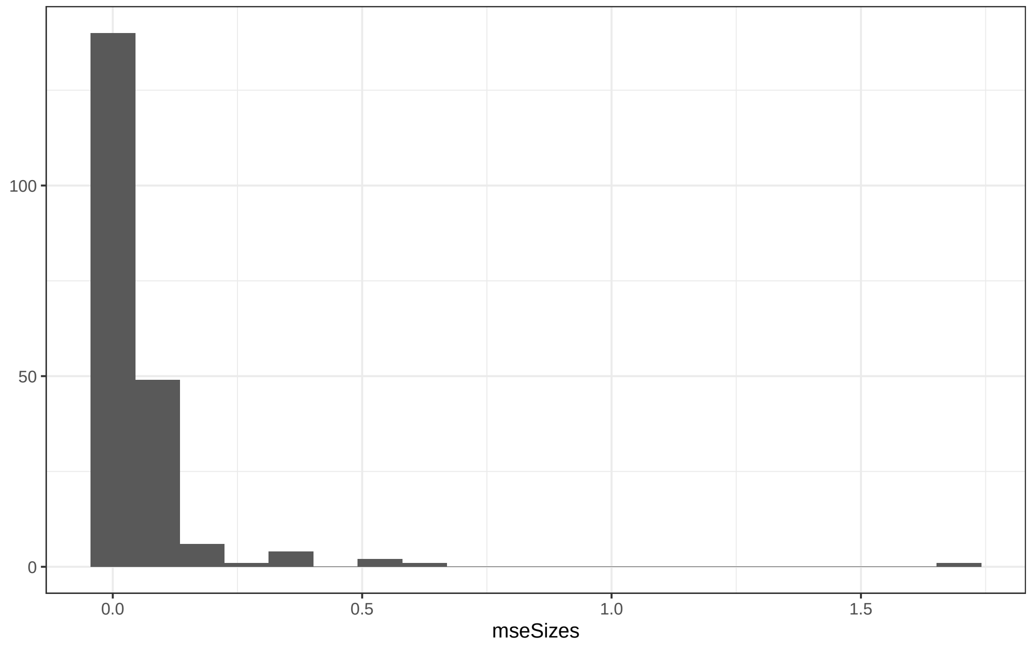


Figure 9: Real data for the 3 selected high-expressing genes

We've introduced a general-purpose high-dimensional multivariate simulation algorithm and provide a high-performance implementation called `bigsimr` (github url). The parallelized (multi-core) algorithm is simple and easy to understand as an application of Gaussian copulas. This method is largely inspired by Madsen and Birkes (2013), but with contributions with regard to scalability, implementation, and application in high-throughput biomedical data (RNA-sequencing). We advocate the use of Kendall's τ as it better captures correlation among components of non-normal, as well as non-linear patterns of association. Moreover, the matching Kendall's τ is nearly trivial due to the invariant of monotone transformations, after an adjustment to the input correlation matrix. Interestingly, our simulation studies show the use of Kendall's τ to works as well as using Pearson correlation coefficient when simulating from multivariate normal distributions.

We also show utility in our methodology through an application to differential gene expression analysis from RNA-sequencing data. The application results show that correlations indeed matter in the large-scale hypothesis testing, as many others have noted (for example, see Efron (2007), Wu and Smyth (2012)). We also hope that the application provides an example workflow — and other strategy to use in simulation design. Even the best-performing simulations we provide show a gap from the empirical distribution. To keep the demonstration straightforward, we did not attempt to match the empirical distribution of test statistics as precisely as possible. Yet our methodology could be more creatively applied to meet that goal. One could consider marginal distributions from different families, such as Poisson for a subset of genes. One could imagine finding a best fitting probability distribution among a class of distributions for each gene and this could perhaps a better fit. One could imagine additional structures/features in the data that an analyst could model to improve the correspondence with the empirical values, for example row correlations and high-dimensional covariance estimators (see Won et al. (2013)).

Mention `rslurm`

There are of course 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 above, this is a computational intense procedure and not a natural choice for the posed problem (seeking to simulate non-normal margins in a scaleable fashion). Yet it is an important aspect that paralleziable, approximation based apporaches are promising [ref Hermite polynomials]. Further, while we provide the ability of the user to specify discrete marginals, exact matching a desired dependency measure is not yet obtained. One potential solution for this is *rescale* to account for ties [ref]. Finally we note, that the algorithm requires invertible marginal cdfs. This gives some restriction to the available marginal probability distributions (DOES IT? FOR EXAMPLE?).

From a practical computing standpoint, the user's computing resource provides the ultimate limit on how large a dimension can be simulated using the `bigsimr` R package. A user must consider carefully the available memory and cores when conducting a Monte Carlo experiment. The algorithm does amends itself well to parallelization and graphical processing unit acceleration (Li et al. 2019) and advanced users may take advantage of those techniques.

Future work includes developing scalable algorithms to match the Pearson correlation matrix exactly and more methods developed specifically for discrete distributions. Further, more sophisticated approaches could involve simulation algorithms that are "estimation aware" and so could explore the role of covariance estimation within the simulation. Lastly, these may reveal new approaches to estimating and evaluating high dimensional regression and Bayesian models. On the implementation side, employing graphical process unit acceleration could produce substantial speedups over our multi-core approach.

8 Supplementary Materials

We provide an open-source implementation of our methology as the `bigsimr` R package, hosted on github.

9 Acknowledgement(s)

The authors gratefully acknowledge the helpful discussions with Professor Walter W. Piegorsch and Professor Edward J. Bedrick during this project's conception. Alex's sister graphic design. TCGA acknowledgement.

10 Disclosure statement

The authors report no conflict of interest. DO WE?

11 Funding

FUNDING FROM EVERYONE HERE. CP3 grant.

12 Nomenclature/Notation

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=rep

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.

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.

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, 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.

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.

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

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.