trunmnt: An R package for calculating moments in a truncated multivariate normal distribution

Seung-Chun Lee*

November 7, 2025

Abstract

The calculation of moments for a truncated multivariate normal distribution has been a persistent challenge in statistical computation. Recently, Kan and Robotti (2017) developed an algorithm, implemented in the R package **MomTrunc** by Galarza *et al.* (2025), which is capable of calculating moments of all orders under various types of truncation. However, its practical utility is often hampered by the heavy computational burden, which increases exponentially with the order of the moment or the dimension of the random vector. Meanwhile, Lee (2020) presented an effective numerical method based on Gauss-Hermite quadrature, which offers superior accuracy and computational efficiency.

This article introduces **trunmnt**, an R package that implements Lee's method. **trunmnt** is designed for moment calculations in most practical statistical problems, offering superior speed and accuracy, particularly in low-dimensional settings.

1 Introduction

The package **trunmnt** computes the moment:

$$\mathrm{E}\left(Y_1^{\kappa_1}\cdots Y_n^{\kappa_n} \mid a_i < Y_i < b_i, i = 1,\ldots,n\right) \tag{1}$$

where $\mathbf{Y} = (Y_1,\ldots,Y_n)' \sim \mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$ is a random vector and the κ_i are nonnegative integers. This calculation yields the κ^{th} order product moment ($\kappa = \sum_{i=1}^{n}\kappa_i$) of a truncated multivariate normal distribution, where each component Y_i is truncated between a lower limit a_i and an upper limit b_i. In what follows, we will use the simplified notation $\mathrm{E}\left(\mathbf{Y}^{\boldsymbol{\kappa}} \mid \mathbf{a} < \mathbf{Y} < \mathbf{b}\right)$ for equation (1).

In this general formulation, some or all of the a_i's may be $-\infty$ and some or all of the b_i's may be ∞. Specific truncation types are defined as follows:

- Lower Truncation: When all b_i are ∞, the random vector is $\mathbf{Y} > \mathbf{a}$.
- Upper Truncation: When all a_i are $-\infty$, the random vector is $\mathbf{Y} < \mathbf{b}$.
- Double Truncation: When $\mathbf{a} < \mathbf{Y} < \mathbf{b}$, implying both lower and upper truncation points.

A challenge in calculating the moments of a multivariate truncated normal distribution arises from the fact that its marginal distributions are not truncated normal. Consequently, calculating the marginal κ^{th} order moment requires computing the κ^{th} order product moment. For instance, computing the second order moment of Y_1, $E(Y_1^2 \mid a_i < Y_i < b_i, i = 1,\ldots,n)$, requires calculating the second order product moment $E(Y_1^2 Y_2^0 \cdots Y_n^0 \mid a_i < Y_i < b_i, i = 1,\ldots,n)$.

The calculation of these product moments has been addressed in numerous studies under varying conditions, including the type of truncation (one-sided or double-sided) and the number of variables (univariate, bivariate, or multivariate). However, currently, two R packages are available for the moment calculation. The first is based on the work of Manjunath and Wilhelm (2012), who provided

*Emeritus Professor, Department of Applied Statistics, Hanshin University, 137 Hanshindae-gil, Osan, Gyunggi-Do, 18101, Korea.

explicit representations for the mean and variance of a multivariate normal distribution with arbitrary double truncation using the moment generating function. These results were implemented in the package **tmvtnorm** (Wilhelm and Manjunath, 2010), which can consequently calculate moments up to the second order. It is worth noting that likelihood-based inferences for some limited dependent variable mixed-effects models require fourth-order moments. The second package, **MomTrunc** (Galarza *et al.*, 2025), uses the iterative relationship developed by Kan and Robotti (2017) for integrals involving the multivariate normal density. Algorithms that rely on recursive relationships are generally known to be slow. This is reflected in **MomTrunc**, which is quite slow even for moderately large dimensions or moment orders. In contrast, the algorithm proposed by Lee (2020) does not depend heavily on the dimension (n); its effectiveness depends primarily on the covariance structure of the variables.

One advantage of both the **tmvtnorm** and **MomTrunc** packages is their capacity to compute moments without imposing structural assumptions on the variance-covariance matrix of the parent multivariate normal distribution. This matrix, for an n-dimensional random vector, comprises $n(n+1)/2$ unique elements. In most statistical models, this matrix is generally unknown, and its estimation typically necessitates assumptions regarding its structure. The **trunmnt** package, while designed to efficiently compute product moments by leveraging the specific variance-covariance matrix structures common in statistical models, also retains the capability to compute moments without such structural assumptions.

2 Computation of moments

2.1 Variance-covariance structure in mixed effects linear models

The general form of the model equation for a linear mixed model is,

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \boldsymbol{\epsilon}.$$

Here $\mathbf{X}_{n\times p}$ is a matrix of explanatory variables, $\boldsymbol{\beta}$ is a vector of unknown parameters, $\mathbf{Z}_{n\times q}$ is the design matrix for the random effects $\mathbf{u}$, and $\boldsymbol{\epsilon}$ is the vector of errors. Customarily, the random effects $\mathbf{u}$ are assumed to follow a multivariate normal distribution with mean $\mathbf{0}$ and symmetric positive definite variance-covariance matrix $\mathbf{D}$, and the error terms are assumed to be independent but not necessarily 'homoscedastic, thus allowing for more flexibility. Specifically, the distribution of $\boldsymbol{\epsilon}$ is assumed to be $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{E})$, where $\mathbf{E}$ is a diagonal matrix with elements $(\sigma_1^2, \ldots, \sigma_n^2)$. This is a generalization of the usual assumption $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \sigma_\epsilon^2\mathbf{I})$.

Assuming $\mathbf{u}$ and $\boldsymbol{\epsilon}$ are independent and letting $\boldsymbol{\mu} = \mathbf{X}\boldsymbol{\beta}$, the response vector $\mathbf{Y}$ has the marginal distribution:

$$\mathbf{Y} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{Z}\mathbf{D}\mathbf{Z}' + \mathbf{E}). \tag{2}$$

The **trunmnt** package focuses on calculating the truncated moment under this variance structure.

Let the conditional κ-th moment of a truncated univariate normal random variable be defined as $m^{\kappa}_{(a,b)}(\mu, \sigma^2) = \mathrm{E}(Y^{\kappa} |\ a < Y < b)$, and the probability of truncation as $\Phi(a, b; \mu, \sigma^2) = \Pr[a < Y < b]$, where $Y \sim N(\mu, \sigma^2)$. Furthermore, let $\phi_n(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ denote the joint density of an n-variate normal distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.

Using Theorem 1 from Lee (2020), the product moment of the truncated multivariate normal distribution, $\mathrm{E}(Y_1^{\kappa_1} \cdots Y_n^{\kappa_n} |\ a_i < Y_i < b_i, i = 1, \ldots, n)$, equals:

$$\frac{1}{\Pr[\mathbf{a} < \mathbf{Y} < \mathbf{b}]} \int_{-\infty}^{\infty} \left(\prod_{i=1}^{n} m^{\kappa_i}_{(a_i,b_i)} \left(\tilde{\mu}_i, \sigma_i^2\right) \Phi\left(a_i, b_i; \tilde{\mu}_i, \sigma_i^2\right) \right) \phi_q(\mathbf{u}; \mathbf{0}, \mathbf{D}) \mathrm{d}\mathbf{u}, \tag{3}$$

where $\tilde{\mu}_i = \mu_i + \mathbf{z}_i'\mathbf{u}$, and $\mathbf{z}_i'$ is the i^{th} row of $\mathbf{Z}$.

Equation (3) is significant because it reduces an n-dimensional integral problem to a q-dimensional integral, where q is typically much less than n. In practice, q is typically 1 or 2 in many statistical models, allowing these low-dimensional integrals to be evaluated efficiently using the Gauss-Hermite quadrature method. Since the moment of a truncated univariate normal random variable,

$m^{\kappa_i}_{(a_i,b_i)}(\tilde{\mu}_i, \sigma_i^2)$, can be obtained via an explicit formula (Burkardt, 2014) (which is implemented in the `utrunmnt()` function within the **trunmnt** package), (3) can be successfully approximated by the Gauss-Hermite method. The **trunmnt** package specifically utilizes the multivariate Gauss-Hermite quadrature described in Jäckel (2005).

A related result of (3) is:

$$\Pr[\mathbf{a} < \mathbf{Y} < \mathbf{b}] = \int_{-\infty}^{\infty} \left(\prod_{i=1}^{n} \Phi\left(a_i, b_i; \tilde{\mu}_i, \sigma_i^2\right) \right) \phi_q(\mathbf{u}; \mathbf{0}, \mathbf{D}) \mathrm{d}\mathbf{u}. \tag{4}$$

The evaluation of $\Pr[\mathbf{a} < \mathbf{Y} < \mathbf{b}]$ constitutes another n-dimensional integral problem that is analytically intractable. This quantity is required by both the **tmvtnorm** and **MomTrunc** packages. Both packages rely on **mvtnorm** (Genz *et al.*, 2025)—an `R` package for computing multivariate normal and t probabilities, quantiles, random deviates, and densities—to calculate this probability after truncation.

The methods used by **mvtnorm** and **trunmnt** differ: **mvtnorm** utilizes a randomized quasi-Monte Carlo procedure (Genz, 1992, 1993), while **trunmnt** evaluates equation (4) using multivariate Gauss-Hermite quadrature. It is generally acknowledged that quasi-Monte Carlo integration performs better than quadrature methods for high-dimensional integral problems, although its convergence properties are typically slow. Conversely, quadrature methods are known to be very fast and efficient for low-dimensional integration.

2.2 Arbitrary positive-definite variance-covariance matrix

Suppose the parent distribution has an arbitrary positive-definite variance-covariance matrix $\boldsymbol{\Sigma}$. Using the spectral decomposition, we can write:

$$\boldsymbol{\Sigma} = \mathbf{P}\boldsymbol{\Lambda}\mathbf{P}'$$

where $\mathbf{P}$ is an orthogonal matrix and $\boldsymbol{\Lambda}$ is a diagonal matrix whose entries are the eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ of $\boldsymbol{\Sigma}$. Since $\boldsymbol{\Sigma}$ is positive-definite, the smallest eigenvalue λ_n is strictly greater than zero.

If all eigenvalues are identical ($\boldsymbol{\Sigma} = \lambda \mathbf{I}_n$), the components of the parent distribution are uncorrelated, and independent if the distribution is multivariate normal. In this special case, computing the product moment simplifies to computing the product of the marginal moments of the independent components, and thus one may use `utrunmnt()` for computing the required univariate moments.

Suppose now that the eigenvalues are not all identical, and that λ_n is an r-fold eigenvalue, where $1 \leq r \leq n-1$. We then partition $\mathbf{P}$ into an $n \times (n-r)$ matrix $\mathbf{P}_1$ and an $n \times r$ matrix $\mathbf{P}_2$. Correspondingly, we partition $\boldsymbol{\Lambda}$ into $\boldsymbol{\Lambda}_1$ and $\lambda_n \mathbf{I}_r$, where $\boldsymbol{\Lambda}_1$ is the diagonal matrix of the remaining eigenvalues $\lambda_1, \ldots, \lambda_{n-r}$. This allows us to re-express $\boldsymbol{\Sigma}$ as:

$$\begin{aligned}
\boldsymbol{\Sigma} &= \left(\mathbf{P}_1, \mathbf{P}_2\right) \begin{bmatrix} \boldsymbol{\Lambda}_1 & \mathbf{0} \\ \mathbf{0} & \lambda_n \mathbf{I}_r \end{bmatrix} \begin{pmatrix} \mathbf{P}_1' \\ \mathbf{P}_2' \end{pmatrix} = \mathbf{P}_1 \boldsymbol{\Lambda}_1 \mathbf{P}_1' + \lambda_n \mathbf{P}_2 \mathbf{P}_2' \\
&= \mathbf{P}_1 \boldsymbol{\Lambda}_1 \mathbf{P}_1' + \lambda_n \left(\mathbf{I}_n - \mathbf{P}_1 \mathbf{P}_1'\right) = \mathbf{P}_1 \left(\boldsymbol{\Lambda}_1 - \lambda_n \mathbf{I}_{n-r}\right) \mathbf{P}_1' + \lambda_n \mathbf{I}_n,
\end{aligned}$$

Note that $\boldsymbol{\Lambda}_1 - \lambda_n \mathbf{I}_{n-r}$ is a diagonal matrix whose entries are strictly positive. Consequently, any arbitrary positive-definite variance-covariance matrix can be written in the form of (2).

This reformulation enables the use of (3) for computing the product moments with an arbitrary variance-covariance matrix. The computation using equation (3) becomes highly efficient when r, the multiplicity of the smallest eigenvalue, is large. This efficiency is significant because many patterned covariance matrices commonly used in statistical applications exhibit a large value of r (*e.g.*, the compound symmetry variance matrix has a multiplicity of $r = n - 1$ for its smallest eigenvalue).

Table 1: Summary of the functions in the **trunmnt** package

Function	Description
`meanvar`	Calculate the mean and variance
`mtrunmnt`	Create a mtrunmnt object with given parameters
`prodmnt`	Calculate the product moment
`probntrun`	Calculate the probability of not being truncated
`utrunmnt`	Calculate the moment of univariate truncated normal distribution

3 The trunmnt package

3.1 Structure and functionality

The **trunmnt** package consists of five main functions for calculating the moments of a truncated multivariate normal distribution. Table 1 provides a summary of these functions. Their functionality will be demonstrated with numerical examples below. Most of the functional parts of **trunmnt** were written using `RcppArmadillo` (Eddelbuettel and Sanderson, 2014) for computational efficiency.

The calculation of moments for a truncated univariate normal distribution is implemented in the `utrunmnt()` function.

```
utrunmnt(k, mu = 0., lower = -Inf, upper = Inf, sd = 1.)
```

Here, the `k` argument specifies the order of the moment and must be a nonnegative integer. The other arguments are self-explanatory. For example, if a normal distribution with mean 5 and standard deviation 1 is upper truncated at 10, the fourth moment is computed with the command:

```
> utrunmnt(4, mu = 5, upper = 10)
[1] 777.9971
```

To calculate the moment $\mathrm{E}(\mathbf{y}^{\kappa} \mid \mathbf{a} < \mathbf{Y} < \mathbf{b})$, you must first create an `mtrunmnt` object using the following syntax:

```
obj <- mtrunmnt(mu, lower = -Inf, upper = Inf, Sigma = 1, Sigmae = 1,
                Z = matrix(rep(1, length(mu)), ncol = 1),
                D = matrix(1, ncol = 1, nrow = 1), nGH = 35)
```

The mean vector of the parent distribution must be passed to the `mu` argument, which has no default value. The lower limits and upper limits are specified by `lower` and `upper`, respectively. These limits can be provided as a scalar (if they are all the same) or as a vector with the same length as `mu`. The variance-covariance matrix must be specified in one of two ways: either by providing `Sigmae`, `D`, and `Z` in (2), or by providing `Sigma` directly. The former approach is preferable for leveraging the computational advantages of the package.

Variance-Covariance Specification:

- `Sigmae`: Specifies the variances of the error terms (ϵ). It must be a scalar or a vector of length equal to `mu`. If a scalar is provided, the variances are assumed to be equal.
- `Z` and `D`: The design matrix for the random effects ($\mathbf{u}$) and its variance-covariance matrix are passed via the `Z` and `D` arguments, respectively.
  - By default, `Z` is a vector of ones in matrix format with the same length as `mu`. It should be changed as necessary so that the number of columns of `Z` and the dimension of the random components are the same.
  - For `D`, you can pass a scalar (for independent random components ($\mathbf{u}$) with equal variances), a vector (for independent components with unequal variances), or a positive-definite matrix. If `D` is provided as a scalar or a vector, it is converted into a diagonal matrix whose diagonal entries are determined by the given value(s), with a dimension equal to the number of columns in `Z`.

- `Sigma`: When `Sigma` is specified, a positive-definite symmetric matrix should be passed to this argument.

The `nGH` argument specifies the number of Gauss-Hermite quadrature points, with a default value of 40. Note that while increasing the number of quadrature points increases the precision of the calculation, the computational burden grows rapidly if the dimension of the random components q is large. For example, when $q = 5$ and `nGH` = 35, the Gauss-Hermite quadrature method requires $35^5 = 52,521,875$ quadrature points to calculate (3). While this number may not seem excessively large for computing a particular product moment, it could be large enough to cause computational issues when calling the `meanvar()` function, which involves calculating $n(n+1)/2 + n$ moments. Therefore, it may be more practical to reduce the value of `nGH` when q is large.

Once a `mtrunmnt` object is created, it can then be passed to `probntrun()`, `prodmnt()`, and `meanvar()` using the following syntax:

- `probntrun(obj)`
- `meanvar(obj)`
- `prodmnt(obj, kappa)`

to compute $\Pr(\mathbf{a} < \mathbf{Y} < \mathbf{b}), \mathrm{E}(\mathbf{y}^{\kappa} \mid \mathbf{a} < \mathbf{Y} < \mathbf{b})$ and the mean and variance-covariance matrix of the truncated variables, respectively. Here, `kappa` is a vector of nonnegative integers with the same length as the mean vector (*i.e.*, the `mu` attribute of `obj`).

3.2 Numerical example

A probit regression model for panel data, an example of a model that relies on truncated multivariate normal random variables, is specified as follows:

$$Y_{it} = \mathbf{x}_{it}'\boldsymbol{\beta} + u_i + \epsilon_{it}, \qquad \text{(Latent variable)}$$

$$W_{it} = \begin{cases} 1 & \text{if } y_{it} \geq a \\ 0 & \text{if } y_{it} < a \end{cases} \qquad \text{(Observed variable)}$$

Here the subscript $i = 1, \ldots, q$ indicates the individual, and the subscript $t = 1, \ldots, n_i$ indicates the time period. u_i is a time-invariant individual specific effect that follows a normal distribution with mean 0 and variance σ_u^2, and ϵ_{it} is the remaining disturbance that also follows a normal distribution with mean 0 and variance σ_e^2, independently of u_i. The parameter a is a known number.

The probit regression model occurs when $\mathbf{Y} = (\mathbf{Y}_1', \ldots, \mathbf{Y}_q')'$, where $\mathbf{Y}_i = (Y_{i1}, \ldots, Y_{in_i})'$, is an unobservable latent vector, and inference about the model parameters can be analyzed through the observable vector $\mathbf{W}$, where $\mathbf{W} = (\mathbf{W}_1', \ldots, \mathbf{W}_q')'$ and $\mathbf{W}_i = (W_{i1}, \ldots, W_{in_1})'$. That is, $\mathbf{Y}|\mathbf{W} = \mathbf{w}$ is a truncated multivariate normal random vector. For each Y_{it}, the truncation limits are determined by the observed value w_{it} of W_{it}: the lower and upper limits are $-\infty$ and a if w_{it} is 0, whereas the lower and upper limits become a and ∞ if $W_{it} = 1$.

The Maximum Likelihood (ML) method requires computing the product moments of the latent variables, which constitutes a q-dimensional integration problem in this probit regression model. The **trunmnt** package addresses this by utilizing the multivariate Gauss-Hermite quadrature method, thereby inheriting the same advantages and limitations. The major trade-off is between its high accuracy in low dimensions and its prohibitive computational cost in high dimensions. For integrals in $q \leq 3$ dimensions, employing a small number of quadrature points often yields extremely accurate results with a relatively low computational cost. Conversely, if $q \geq 4$, the method suffers from the curse of dimensionality; that is, as noted previously, for a q-dimensional integral with m points per dimension, the total number of evaluation points is m^q. This cost becomes prohibitively high even for moderate dimensions, rendering the method infeasible for high-dimensional problems ($q > 5$ or so), where Monte Carlo methods or sparse grid techniques are preferred. We recommend using **trunmnt** when $q \leq 3$. Under this condition, the default value of `nGH` $= 35$ remains computationally tractable on modern computers. If $q \geq 4$, caution is advised when calling the `meanvar()` function, especially when n is large.

In general, q is a given number that cannot be controlled; however, in some cases, we can decompose the high-dimensional integration problem into lower-dimensional subproblems. For example, in the random effect panel probit regression model, $\mathbf{Y}$ is composed of independent vectors $\mathbf{Y}_i$, where $i = 1, \ldots, q$. Consequently, the conditional expectation can be written as a product of conditional expectations:

$$\mathrm{E}\left(\mathbf{Y}^{\boldsymbol{\kappa}} \mid \mathbf{a} < \mathbf{Y} < \mathbf{b}\right) = \prod_{i=1}^{q} \mathrm{E}\left(\mathbf{Y}_i^{\boldsymbol{\kappa}_i} \mid \mathbf{a}_i < \mathbf{Y}_i < \mathbf{b}_i\right) \tag{5}$$

where $\boldsymbol{\kappa}_i, \mathbf{a}_i$ and $\mathbf{b}_i$ are the appropriate partitions of $\boldsymbol{\kappa}, \mathbf{a}$ and $\mathbf{b}$, respectively. Since only 1-dimensional integration is required for each term $\mathrm{E}\left(\mathbf{Y}_i^{\boldsymbol{\kappa}_i} \mid \mathbf{a}_i < \mathbf{Y}_i < \mathbf{b}_i\right)$, we can effectively transform the original q-dimensional integration problem into a set of one-dimensional problems, allowing for efficient and highly accurate computation.

The following R script, based on the R environment (R Core Team, 2025), mimics a random-effects panel probit regression model. It generates a vector of latent variables $\mathbf{Y}' = (\mathbf{Y}_1', \mathbf{Y}_2', \mathbf{Y}_3')$, where the 5-dimensional subvectors $\mathbf{Y}_i$ are independently and identically distributed as $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Here, $\boldsymbol{\mu}$ is a 5-element vector of equally spaced values ranging from -1 to 1. The covariance matrix is defined as $\boldsymbol{\Sigma} = 2\mathbf{J}_5 + \mathbf{I}_5$, where $\mathbf{J}_5$ is the 5×5 matrix of ones and $\mathbf{I}_5$ is the 5×5 identity matrix, respectively. Binary observations w_{it} are then generated via the rule $w_{it} = 1$ if $Y_{it} \geq 0$. This transformation defines the truncation rules for the observed data:

- If $w_{it} = 1$, the latent variable Y_{it} is lower-truncated (sometimes called left-censored), implying $Y_{it} \in [0, \infty)$.
- If $w_{it} = 0$, Y_{it} is upper-truncated (sometimes called right-censored), implying $Y_{it} \in (-\infty, 0)$.

```
set.seed(123)
sigma2e <- 1 ; sigma2u <- 2  # Error variance and random effect variance
n  <- 5 ; k <- 3
u  <- rnorm(k, sd = sqrt(sigma2u))
Z  <- model.matrix(~ factor(rep(1:k, each = n)) - 1) # Design matrix for
D  <- diag(sigma2u, k)                               # random effects
S  <- Z %*% D %*% t(Z) + diag(sigma2e, n*k)
mu <- rep(seq(-1,1, length.out = n), k)              # Mean vector mu
y  <- mu + Z %*% u + rnorm(n*k, sd = sqrt(sigma2e))
w  <- ifelse(y >= 0, 1, 0)
a  <- rep(-Inf, n*k)
b  <- rep( Inf, n*k)
a[w == 1] <- 0
b[w == 0] <- 0
```

The following examples demonstrate the main functions in the trunmnt package, specifically using the conditional distribution of $\mathbf{Y}_1$ given $\mathbf{w}_1$.

```
> obj <- mtrunmnt(mu[1:5], a[1:5], b[1:5], Sigmae = 1, D = 2)
> probntrun(obj)
[1] 0.01465224
> meanvar(obj)
$tmean
[1] -1.5460538 -1.2227138  0.7203610  0.9030988 -0.6186135

$tvar
           [,1]       [,2]       [,3]       [,4]       [,5]
[1,] 0.87417118 0.13333969 0.06742012 0.08987795 0.05916922
[2,] 0.13333969 0.69024283 0.05456682 0.07276999 0.04880507
[3,] 0.06742012 0.05456682 0.34753567 0.04263978 0.02377110
[4,] 0.08987795 0.07276999 0.04263978 0.47601312 0.03170960
[5,] 0.05916922 0.04880507 0.02377110 0.03170960 0.27720637
```

```
> c(prodmnt(obj, c(4,0,0,0,0)), prodmnt(obj, c(3,1,0,0,0)),
+   prodmnt(obj, c(2,2,0,0,0)), prodmnt(obj, c(2,1,1,0,0)),
+   prodmnt(obj, c(1,1,1,1,0)))
[1] 23.352856 11.732113  8.706671 -2.734212  1.079536
```

You can also obtain the `mtrunmnt` object using an explicit covariance matrix:

```
> obj <- mtrunmnt(mu[1:5], a[1:5], b[1:5], Sigma = S[1:5, 1:5])
```

Applying this object to the functions will yield exactly the same results as the previous example.

The probability $\Pr[\mathbf{a} < \mathbf{Y} < \mathbf{b}]$ can also be calculated using the function `pmvnorm()` from the **mvtnorm** package (Genz *et al.*, 2025). It employs the randomized quasi-Monte Carlo procedure proposed by Genz (1992, 1993) to compute the probability mass. Consequently, the calculated values may fluctuate slightly from run to run, even under identical conditions. Nevertheless, the functions `probntrun()` (with the default value of `nGH`) and `pmvnorm()` yielded values that agreed up to the fifth decimal place in most runs. Note also that the functions `ptmvnorm` (from the **tmvnorm** package) and `pmvnormt` (from the **MomTrunc** package) can compute the probability mass. These functions are essentially aliases of `pmvnorm()`.

The accurate calculation of this probability is particularly important when n is large, as the probability value decreases with increasing n. When computing the product moment, the integral value within (3) is divided by this small probability; thus, a small change in the probability calculation may lead to a large change (or instability) in the final moment.

The **trunmnt** package computes several parameters of a truncated multivariate normal distribution for which analytical solutions are unavailable, rendering their true values unknown. Consequently, assessing the accuracy of its results is challenging.

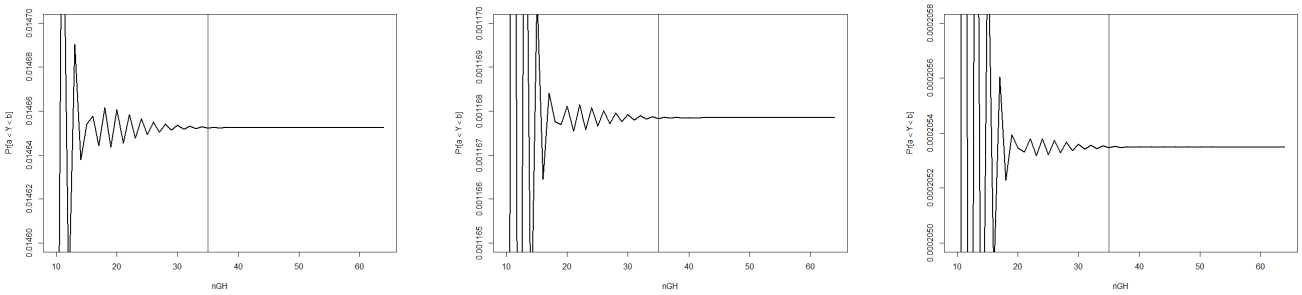
The Gauss-Hermite quadrature method is known to yield an exact integral if the integrand is a polynomial of degree at most $2m-1$, where m is the number of quadrature points. The function enclosed in parentheses of Equation (3) is a re-expression of

$$f(\mathbf{u}) = \prod_{i=1}^{n} \int_{a_i}^{b_i} y^{\kappa_i} \phi_1\left(y; \tilde{\mu}_i, \sigma_i^2\right) dy,$$

where $\tilde{\mu} = \mu_i + \mathbf{z}_i'\mathbf{u}$ and $\mathbf{z}_i'$ is the i^{th} row of $\mathbf{Z}$). This function is easily shown to be infinitely differentiable (C^∞) with respect to $\mathbf{u}$ over the domain $\mathcal{R}^q$. Similarly, the function enclosed in parentheses in (4) is also C^∞. Although their complexity increases as n increases, this C^∞ continuity ensures that the integrand for the Gauss-Hermite method is smooth and, crucially, accurately approximable by a polynomial. Therefore, we can expect the Gauss-Hermite method to perform well with a moderate number of quadrature points.

We tested various `nGH` values when computing the probability $\Pr[\mathbf{a} < \mathbf{Y} < \mathbf{b}]$ to determine the appropriate number of quadrature points for an accurate calculated value. Figure 1 shows the calculated values of $\Pr[\mathbf{a}_q^* < \mathbf{Y}_q^* < \mathbf{b}_q^*]$ against the number of quadrature points `nGH`, for $q = 1, 2, 3$. Here, $\mathbf{Y}_q^*$, $\mathbf{a}_q^*$, and $\mathbf{b}_q^*$ are defined as $\mathbf{Y}_q^* = (\mathbf{Y}_1', \ldots, \mathbf{Y}_q')'$, $\mathbf{a}_q^* = (\mathbf{a}_1', \ldots, \mathbf{a}_q')'$, and $\mathbf{b}_q^* = (\mathbf{b}_1', \ldots, \mathbf{b}_q')'$, where q serves as the dimension of the integral in (4). The calculated values stabilize after `nGH` $= 35$. Beyond this point, the values for all three cases exhibit a maximum variation of 1.0×10^{-7}.

Recall that the **tmvnorm** package can compute up to the second-order moments based on the moment generating functions (MGF). Specifically, the `mtmvnorm()` function within **tmvnorm** produces the mean vector and the variance-covariance matrix of a truncated multivariate normal random vector. The `meanvarTMD()` function in **MomTrunc** offers the same functionality. Note that **MomTrunc** contains two functions, `meanvarTMD` and `momentsTMD`, for the computation of product moments. While both functions are rooted in the Kan and Robotti (2017) recurrence relations, they use different computational implementations based on the order of the moment, primarily for efficiency. For the first and second moments (the mean vector and the variance-covariance matrix), the general recurrence relation is explicitly solved and simplified. This optimized formula avoids the full iterative complexity required for higher orders, making the calculation of the mean and variance faster and more computationally stable than applying the general recursive procedure directly.



(a) $q = 1, \Pr(\mathbf{a}_1 < \mathbf{Y}_1 < \mathbf{b}_1)$ (b) $q = 2, \Pr(\mathbf{a}_2^* < \mathbf{Y}_2^* < \mathbf{b}_2^*)$ (c) $q = 3, \Pr(\mathbf{a}_3^* < \mathbf{Y}_3^* < \mathbf{b}_3^*)$

Figure 1: Calculated probability mass after truncation as a function of `nGH`

We aim to verify the accuracy of `meanvar` by comparing it with `mtmvnorm` and `meanvar TMD`. Although the moment calculations in both `mtmvnorm` (MGF) and `meanvarTMD` (closed-form solution of the recurrence relation) rely on deterministic analytic formulas for the moment calculation, they use a randomized quasi-Monte Carlo procedure to calculate the truncated probability (given in Equation (2.2)) required for the computation. Consequently, their outputs will not be identical with every run. It was observed, however, that the outputs of `mtmvnorm()` and `meanvarTMD()` typically match only within 2 to 3 decimal places in most runs. We cannot identify the true value of the moments. But, it was observed that the output of `meanvar` is closer to that of `meanvarTMD`. They match within 3 to 4 decimal places, but `meanvar` and `mtmvnorm` match within 2 to 3 decimal places. Since the true value is unknown, it is not possible to conclude which function yields the moment closest to the true value.

To assess the accuracy of **trunmnt**, we leverage the theoretical independence of $\mathbf{Y}_1$, $\mathbf{Y}_2$, and $\mathbf{Y}_3$ in the unconditional distribution. A partial output of the mean vector and variance-covariance matrix of the truncated vector $\mathbf{Y}|\mathbf{w}$ is presented below:

```
> obj3 <- mtrunmnt(mu, a, b, Sigma = S)
> meanvar(obj3)
$tmean
 [1] -1.5460538 -1.2227138  0.7203610  0.9030988 -0.6186135 -1.1630524 ...

$tvar
               [,1]          [,2]          [,3]          [,4]          [,5]
 [1,]  8.741712e-01  1.333397e-01  6.742012e-02  8.987795e-02  5.916922e-02
 [2,]  1.333397e-01  6.902428e-01  5.456682e-02  7.276999e-02  4.880507e-02
 [3,]  6.742012e-02  5.456682e-02  3.475357e-01  4.263978e-02  2.377110e-02
 [4,]  8.987795e-02  7.276999e-02  4.263978e-02  4.760131e-01  3.170960e-02
 [5,]  5.916922e-02  4.880507e-02  2.377110e-02  3.170960e-02  2.772064e-01
 [6,] -3.108624e-15  1.332268e-15  1.332268e-15 -1.332268e-15  4.440892e-16
 [7,]  8.881784e-16  3.108624e-15  2.220446e-16 -1.110223e-15  1.221245e-15
 [8,]  3.996803e-15 -1.998401e-15 -1.554312e-15  3.330669e-15 -6.661338e-16
 [9,] -6.661338e-16 -7.327472e-15 -1.998401e-15  3.330669e-15 -1.776357e-15
[10,] -5.329071e-15 -8.881784e-15 -4.440892e-16  5.551115e-15 -2.997602e-15
[11,] -8.881784e-15 -1.265654e-14  6.661338e-16  7.105427e-15 -6.106227e-15
[12,]  8.881784e-16 -8.659740e-15 -1.332268e-15 -8.881784e-16 -4.107825e-15
[13,] -3.552714e-15  8.881784e-16  1.110223e-15  3.774758e-15 -2.220446e-16
[14,] -1.776357e-15 -1.021405e-14 -4.440892e-15  1.110223e-14 -5.329071e-15
[15,]  6.217249e-15 -1.687539e-14 -6.661338e-15 -4.440892e-15 -3.552714e-15
```

The computed mean vector and variance-covariance matrix of $\mathbf{Y}_1|\mathbf{w}$ are identical to the previously reported output. More importantly, inter-subvector covariance matrices $\mathrm{Cov}(\mathbf{Y}_1, \mathbf{Y}_2|\mathbf{w})$ are computed

to be negligibly small ($\sim 10^{-15}$), aligning with their theoretical true value of **0**. Given that the calculation of these covariances involves three-dimensional integrals, this result strongly suggests that the Gauss-Hermite quadrature performs exceptionally well, at least for integrals up to this dimension. It is noteworthy that the **tmvtnorm** and **MomTrunc** packages fail to achieve this level of approximation for the covariances, offering accuracy only up to two or three decimal places.

Another key advantage of `trunmnt` is its ability to compute higher-order moments significantly faster than `MomTrunc`. For instance, the output of `prodmnt` in the preceding example demonstrates the computation of the 4^{th} order moments. Comparing this result with the output of the `momentsTMD` function in the `MomTrunc` package shows agreement only up to the first or second decimal place. However, the difference in computational speed is dramatic, specifically, when the dimension of **Y** is greater than 10. `momentsTMD()` is too slow to be practically used, rendering it infeasible for such high-dimensional problems.

4 Summary

This paper introduces the **trunmnt** package, an `R` implementation designed for calculating the product moment of a truncated multivariate normal distribution. The package can compute high-order moments of an arbitrarily truncated multivariate normal distribution, but is particularly efficient when the variance-covariance matrix is a patterned matrix. **trunmnt** offers two major advantages: it can yield more accurate values than the others in some cases, and it is significantly faster. For example, when computing the mean and variance-covariance matrices for the example in Section 3.2, **trunmnt** was approximately 98 times faster than **tmvtnorm** and 19 times faster than **MomTrunc** in average of 1000 replications. This speed advantage is particularly dramatic with high-dimensional variables. This slow computational speed is a major impediment to using both competitor packages in real-world problems. Furthermore, the computation of higher-order moments is more problematic: **tmvtnorm** lacks the functionality entirely, while **MomTrunc** is unusably slow. Considering these points, **trunmnt** is a powerful alternative for certain problems.

It should be noted, however, that the **trunmnt** package shares the limitations of the multivariate Gauss-Hermite quadrature. Specifically, the computational burden grows very rapidly with the increasing dimension (q) of the random components. For example, when $q = 5$ and `nGH` $= 35$, the method requires the calculation of $35^5 = 52,521,875$ quadrature points. While this number is not excessively large for computing a particular product moment, it can lead to computational issues when calling the `meanvar()` function, which requires calculating a substantially larger number of moments.

References

Burkardt J (2014). The truncated normal distribution, *Online document*, Available from: `http://people.sc.fsu.edu/~jburkardt/presentations/truncated_normal.pdf`

Eddelbuettel D and Sanderson C (2014). RcppArmadillo: Accelerating R with high-performance C++ linear algebra, *Computational Statistics & Data Analysis*, **71**, 1054–1063.

Galarza CE, Kan R, and Lachos VH (2021). MomTrunc: Moments of folded and doubly truncated multivariate distributions, *Online document,* Available from: `https://cran.r-project.org/web/packages/MomTrunc/MomTrunc.pdf`

Genz A (1992). Numerical computation of multivariate normal probabilities, *Journal of Computational and Graphical Statistics*, **1**, 141-–150.

Genz A (1993). Comparison of methods for the computation of multivariate normal probabilities, *Computing Science and Statistics*, **25**, 400-–405.

Genz *et al.* (2021). mvtvnorm: Multivariate Normal and t Distributions, R package version 1.1, Available from: `https://cran.r-project.org/web/packages/mvtnorm/mvtnorm.pdf`

Jäckel P (2005). A note on multivariate Gauss-Hermite quadrature, *London: ABN-Amro*, Available from: urlhttp://www.jaeckel.org/ANoteOnMultivariateGaussHermiteQuadrature.pdf

Kan R and Robotti C (2017). On moments of folded and truncated multivariate normal distributions, *Journal of Computational and Graphical Statistics*, **26**, 930–934.

Lee S.-C. (2021). Moments calculation for truncated multivariate normal in nonlinear generalized mixed models, *Communications for Statistical Applications and Methods*, **27**, 377–383.

Manjunath BG and Wilhelm S (2012). Moments calculation for the doubly truncated multivariate normal density, *SSRN Electronic Journal*, arXiv preprint arXiv:1206.5387.

R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. `https://www.R-project.org/`.

Wilhelm S and Manjunath BG (2010). tmvtnorm: Truncated multivariate normal and Student t distribution, *The R Journal*, **2**.