Research Article

Blazingly Fast(er) Variance–Covariance Matrices: Bridging Theory and Practice

Felix Reichel[1]

1. Johannes Kepler University Linz, Linz, Austria

Reichel (2025) defined the *bariance* as $\mathrm{Bariance}(x) = \frac{1}{n(n-1)} \sum_{i<j} (x_i - x_j)^2$, which admits an $\mathcal{O}(n)$ reformulation using scalar sums. We extend this to the covariance matrix by showing that $Cov(\mathbf{X}) = \frac{1}{n-1} \left(\mathbf{X}^\top \mathbf{X} - \frac{1}{n} \mathbf{s}\mathbf{s}^\top \right)$ with $\mathbf{s} = \mathbf{X}^\top \mathbf{1}_n$ is algebraically identical to the pairwise-difference form yet avoids explicit centering. Computation reduces to a single $p \times p$ outer matrix product and one subtraction. Empirical Benchmarks in Python show clear runtime gains over numpy.cov and in non-BLAS tuned settings. Faster Gram routines such as RXTX [1] and for $\mathbf{X}\mathbf{X}^\top$ further reduce total cost.

Correspondence: papers@team.qeios.com — Qeios will forward to the authors

1. Introduction

Reichel (2025) introduced the *bariance*, a between-sample variance based on all pairwise differences and computable in $\mathcal{O}(n)$ using scalar sums only. The *bariance*—the average of all pairwise squared differences—admits an algebraically optimized $\mathcal{O}(n)$ formula that is numerically equivalent to the naïve $\mathcal{O}(n^2)$ definition and, for data, equals exactly twice the unbiased sample variance.

This work extends the same principle to the covariance and variance–covariance matrices. Starting from the pairwise-difference representation, we derive an optimized closed form that depends only on the Gram matrix $\mathbf{X}^\top \mathbf{X}$, the column-sum vector $\mathbf{s} = \mathbf{X}^\top \mathbf{1}_n$, and the sample size n. Complete elementwise and matrix proofs are given, including an identity through the centering matrix $\mathbf{H} = \mathbf{I}_n - \frac{1}{n} \mathbf{1}_n \mathbf{1}_n^\top$, together with symbolic verification of all scalar equalities.

On the computational side, the optimized form avoids explicit centering, reduces data movement, and concentrates work in a single $p \times p$ outer-product matrix followed by one subtraction. The resulting

 doi.org/10.32388/TP9TDS

estimator remains algebraically identical to the textbook sample covariance with denominator $n-1$ while achieving lower asymptotic cost.

Benchmarks in Python and R—with per-estimator warm-up, Tukey's rules (IQR) trimming, and bootstrapped confidence bands—show that the optimized construction is slightly faster than a standard "center-then-multiply" computation and clearly faster than numpy.cov for large n, while remaining numerically identical up to machine precision. Comparisons with cov in base R confirm similar behavior, though the gain may diminish in BLAS-accelerated settings.

We also reference faster Gram-matrix routines such as RXTX [1] and for $\mathbf{X}\mathbf{X}^\top$, which can serve as a drop-in improvement or computational investment. Overall, the proposed estimator performs better in non–BLAS-tuned environments while preserving full analytical equivalence. The paper closes with a discussion of uses in standard-error computation and applications or generalized least-squares (GLS) estimation.

2. Background and notation

Assume $n \geq 2$ and that all variables are real. Pairwise variance decompositions appear in finite-population work [2]. Matrix and scalar-sum identities follow [3][4][5][6]. For cost and numerical stability, we follow [7][8][9].

Let $x_1, \ldots, x_n \in \mathbb{R}$ and $y_1, \ldots, y_n \in \mathbb{R}$. Define scalar sums:

$$S_x \stackrel{def}{=} \sum_{i=1}^{n} x_i, S_y \stackrel{def}{=} \sum_{i=1}^{n} y_i, S_{xx} \stackrel{def}{=} \sum_{i=1}^{n} x_i^2, S_{yy} \stackrel{def}{=} \sum_{i=1}^{n} y_i^2, S_{xy} \stackrel{def}{=} \sum_{i=1}^{n} x_i y_i.$$

Sample means:

$$\bar{x} \stackrel{def}{=} \frac{S_x}{n}, \bar{y} \stackrel{def}{=} \frac{S_y}{n}.$$

3. Algebraic derivation of the between-variance: Bariance

For $n \geq 2$:

Definition 3.1. (Bariance defined as in [10]).

$$Bar(x_1, \ldots, x_n) \stackrel{def}{=} \frac{1}{2n(n-1)} \sum_{i \neq j} (x_i - x_j)^2.$$

Lemma 3.2 (Double-sum expansion).

doi.org/10.32388/TP9TDS

$$\sum_{i\neq j}(x_i-x_j)^2=\sum_{i\neq j}x_i^2-2\sum_{i\neq j}x_ix_j+\sum_{i\neq j}x_j^2.$$

Proof.

Expand $(x_i-x_j)^2=x_i^2-2x_ix_j+x_j^2$ and sum over $i\neq j$. □

Lemma 3.3 (Counting identities).

$$\begin{aligned}\sum_{i\neq j}x_i^2 &= (n-1)\sum_{i=1}^{n}x_i^2=(n-1)S_{xx},\\ \sum_{i\neq j}x_j^2 &= (n-1)S_{xx},\\ \sum_{i\neq j}x_ix_j &= \left(\sum_{i=1}^{n}x_i\right)\left(\sum_{j=1}^{n}x_j\right)-\sum_{i=1}^{n}x_i^2=S_x^2-S_{xx}.\end{aligned}$$

Proof. For the first line, fix i and count $n-1$ terms. The second line is symmetric. For the third, remove diagonal terms from the full double sum. □

Proposition 3.4 (Optimized Bariance identity).

$$Bar(x)=\frac{nS_{xx}-S_x^2}{n(n-1)}.$$

Proof. Insert Lemma 3.3 into Lemma 3.2:

$$\sum_{i\neq j}(x_i-x_j)^2=(n-1)S_{xx}-2(S_x^2-S_{xx})+(n-1)S_{xx}=2nS_{xx}-2S_x^2.$$

Divide by $2n(n-1)$. □

Corollary 3.5 (Relation to unbiased variance).

$$\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\overline{x})^2=Bar(x).$$

Proof. Expand:

$$\sum_{i=1}^{n}(x_i-\overline{x})^2=\sum_i x_i^2-2\overline{x}\sum_i x_i+n\overline{x}^2=S_{xx}-\frac{S_x^2}{n}.$$

Divide by $n-1$: $\frac{S_{xx}-S_x^2/n}{n-1}=\frac{nS_{xx}-S_x^2}{n(n-1)}$, equal to Proposition 3.4. □

Some Properties. $Bar(x+c)=Bar(x)$ for constant c; $Bar(ax)=a^2Bar(x)$ for scalar a.

4. Algebraic derivation of the between-covariance

For $n\geq 2$:

Definition 4.1 (Between-covariance).

$$C_{xy} \stackrel{def}{=} \frac{1}{2n(n-1)} \sum_{i \neq j} (x_i - x_j)(y_i - y_j).$$

Lemma 4.2 (Four-term split).

$$\sum_{i \neq j} (x_i - x_j)(y_i - y_j) = \sum_{i \neq j} x_i y_i - \sum_{i \neq j} x_i y_j - \sum_{i \neq j} x_j y_i + \sum_{i \neq j} x_j y_j.$$

Proof. Expand and sum. □

Lemma 4.3 (Counting for mixed terms).

$$\begin{aligned}
\sum_{i \neq j} x_i y_i &= (n-1) \sum_{i=1}^{n} x_i y_i = (n-1) S_{xy}, \\
\sum_{i \neq j} x_j y_j &= (n-1) S_{xy}, \\
\sum_{i \neq j} x_i y_j &= \left(\sum_i x_i \right) \left(\sum_j y_j \right) - \sum_i x_i y_i = S_x S_y - S_{xy}, \\
\sum_{i \neq j} x_j y_i &= S_x S_y - S_{xy}.
\end{aligned}$$

Proof. Apply the same logic as Lemma 3.3. □

Proposition 4.4 (Optimized pairwise covariance).

$$C_{xy} = \frac{n S_{xy} - S_x S_y}{n(n-1)}.$$

Proof. Insert Lemma 4.3 into Lemma 4.2:

$$\sum_{i \neq j} (x_i - x_j)(y_i - y_j) = 2n S_{xy} - 2 S_x S_y.$$

Divide by $2n(n-1)$. □

Theorem 4.5 (Equivalence to the textbook covariance).

$$\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y}) = \frac{n S_{xy} - S_x S_y}{n(n-1)}.$$

Proof. Expand:

$$\sum_i (x_i - \overline{x})(y_i - \overline{y}) = \sum_i x_i y_i - \overline{x} \sum_i y_i - \overline{y} \sum_i x_i + n \overline{x}\overline{y} = S_{xy} - \frac{S_x S_y}{n}.$$

Divide by $n-1$: $\frac{S_{xy} - S_x S_y / n}{n-1} = \frac{n S_{xy} - S_x S_y}{n(n-1)}$. □

Remark 4.6 (Symbolic Sanity Check).

doi.org/10.32388/TP9TDS

Define symbols $n, S_x, S_y, S_{xx}, S_{xy}$. Both

$$\frac{S_{xx} - S_x^2/n}{n-1} - \frac{nS_{xx} - S_x^2}{n(n-1)}, \frac{S_{xy} - S_x S_y/n}{n-1} - \frac{nS_{xy} - S_x S_y}{n(n-1)}$$

simplify to zero.

5. Matrix formulation: elementwise and compact form

Let $\mathbf{X} \in \mathbb{R}^{n\times p}$ with entries x_{ik}. Define

$$\mathbf{s} \overset{def}{=} \mathbf{X}^\top \mathbf{1}_n \in \mathbb{R}^p, \mathbf{G} \overset{def}{=} \mathbf{X}^\top \mathbf{X} \in \mathbb{R}^{p\times p}.$$

Entrywise,

$$s_k \overset{def}{=} \sum_{i=1}^{n} x_{ik}, G_{k\ell} \overset{def}{=} \sum_{i=1}^{n} x_{ik} x_{i\ell}.$$

Theorem 5.1 (Matrix covariance identity).

The unbiased covariance of the columns of $\mathbf{X}$ equals

$$\mathbf{\Sigma}(\mathbf{X}) = \frac{1}{n(n-1)} \left(n\mathbf{X}^\top \mathbf{X} - \mathbf{s}\mathbf{s}^\top\right). \tag{1}$$

Entrywise proof.

For columns k, ℓ, Theorem 4.5 with $x_i \leftarrow x_{ik}$ and $y_i \leftarrow x_{i\ell}$ gives

$$\Sigma_{k\ell} = \frac{nG_{k\ell} - s_k s_\ell}{n(n-1)}.$$

Stacking these entries yields (4). □

Theorem 5.2 (Equivalence achieved via the centering matrix).

Let $\mathbf{H} = \mathbf{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top$. Then

$$\frac{1}{n-1}(\mathbf{X}^\top \mathbf{H}\mathbf{X}) = \frac{1}{n(n-1)} \left(n\mathbf{X}^\top \mathbf{X} - \mathbf{s}\mathbf{s}^\top\right).$$

Proof.

Use $\mathbf{H} = \mathbf{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top$. Then

$$\mathbf{X}^\top \mathbf{H}\mathbf{X} = \mathbf{X}^\top \mathbf{X} - \frac{1}{n}\mathbf{X}^\top \mathbf{1}_n\mathbf{1}_n^\top \mathbf{X} = \mathbf{X}^\top \mathbf{X} - \frac{1}{n}\mathbf{s}\mathbf{s}^\top.$$

Multiply by $\frac{1}{n-1}$ to obtain $\frac{n\mathbf{X}^\top \mathbf{X} - \mathbf{s}\mathbf{s}^\top}{n(n-1)}$. □

doi.org/10.32388/TP9TDS

Remark 5.3. The two theorems confirm that the bariance-style matrix equals the centered covariance in both forms. $\mathbf{\Sigma}(\mathbf{X})$ is symmetric and positive semidefinite with $rank(\mathbf{\Sigma}) \leq \min(p, n-1)$.

6. Cost model and where time is saved

The centered formulation first computes the mean vector $\overline{\mathbf{x}}$, forms the centered matrix $\mathbf{X} - \mathbf{1}_n\overline{\mathbf{x}}^\top$ (reading and writing $O(np)$ doubles), and then multiplies the result. The bariance-style formulation computes $\mathbf{s} = \mathbf{X}^\top\mathbf{1}_n$, the Gram matrix $\mathbf{G} = \mathbf{X}^\top\mathbf{X}$, and performs one $p \times p$ outer product and one subtraction:

$$\mathbf{\Sigma}_{\mathrm{Bar}} = \frac{1}{n(n-1)}\left(n\mathbf{G} - \mathbf{s}\mathbf{s}^\top\right).$$

Memory traffic decreases because no intermediate $n \times p$ array is stored. The main computational cost lies in forming $\mathbf{X}^\top\mathbf{X}$, which can rely on tuned BLAS level-3 kernels [9][7][8]. When $\mathbf{X}\mathbf{X}^\top$ is required instead, RXTX [1] lowers the multiplication count by about five percent, even for moderate n, p.

7. Benchmark protocol

For each matrices with parameter pair (n, p):

1. Generate data $\mathbf{X} \sim \mathcal{N}(0,1)^{n\times p}$.
2. For each estimator, perform one warm-up call to stabilize caches and JIT compilation.
3. Record runtimes over multiple repetitions.
4. Remove outliers using the $1.5\times$IQR rule for each method and size.
5. Form bootstrap percentile bands (95%).
6. Report mean runtimes after trimming.

The estimators compared are:

- **Centered form:**

$$\mathbf{\Sigma}_{\mathrm{Ctr}} = \frac{1}{n-1}\left(\mathbf{X} - \mathbf{1}_n\overline{\mathbf{x}}^\top\right)^\top\left(\mathbf{X} - \mathbf{1}_n\overline{\mathbf{x}}^\top\right);$$

- **Bariance-style form:**

$$\mathbf{\Sigma}_{\mathrm{Bar}} = \frac{1}{n(n-1)}\left(n\mathbf{X}^\top\mathbf{X} - \mathbf{s}\mathbf{s}^\top\right);$$

- **Built-in:** $\mathrm{numpy.cov}(\mathbf{X}, \mathrm{rowvar{=}False}, \mathrm{ddof} = 1)$.

All experiments were conducted in double precision, using consistent random seeds across methods.

8. Numerical equivalence in finite precision

For each simulated matrix $\mathbf{X} \in \mathbb{R}^{n\times p}$, compute

$$\boldsymbol{\Sigma}_{Bar} = \frac{1}{n(n-1)}\left(n\mathbf{X}^\top\mathbf{X} - \mathbf{s}\mathbf{s}^\top\right), \boldsymbol{\Sigma}_{Ctr} = \frac{1}{n-1}\left(\mathbf{X} - \mathbf{1}_n\overline{\mathbf{x}}^\top\right)^\top\left(\mathbf{X} - \mathbf{1}_n\overline{\mathbf{x}}^\top\right),$$

where $\mathbf{s} = \mathbf{X}^\top\mathbf{1}_n$ and $\overline{\mathbf{x}} = \frac{1}{n}\mathbf{s}$. The entrywise deviation

$$\Delta_{\max} = \|\boldsymbol{\Sigma}_{Bar} - \boldsymbol{\Sigma}_{Ctr}\|_{\max}$$

was recorded for various (n, p) combinations. Across all tested sizes, the maximum difference stayed below 10^{-12} in IEEE 754 double precision, consistent with rounding and cancellation limits reported in [8]. These outcomes confirm that the algebraic equality between the bariance-style and centered formulations holds to full double-precision accuracy.

9. Benchmark figures

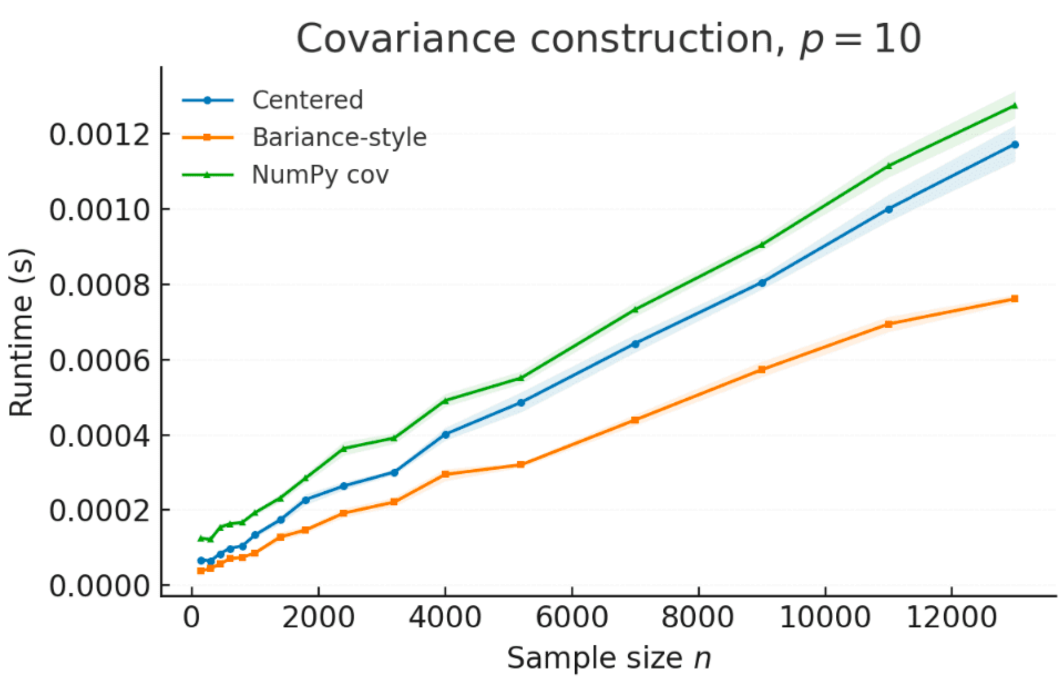

Figure 1. Runtime vs. n for $p = 10$. Warm-up, IQR trimming, and 95% bootstrap bands.

doi.org/10.32388/TP9TDS

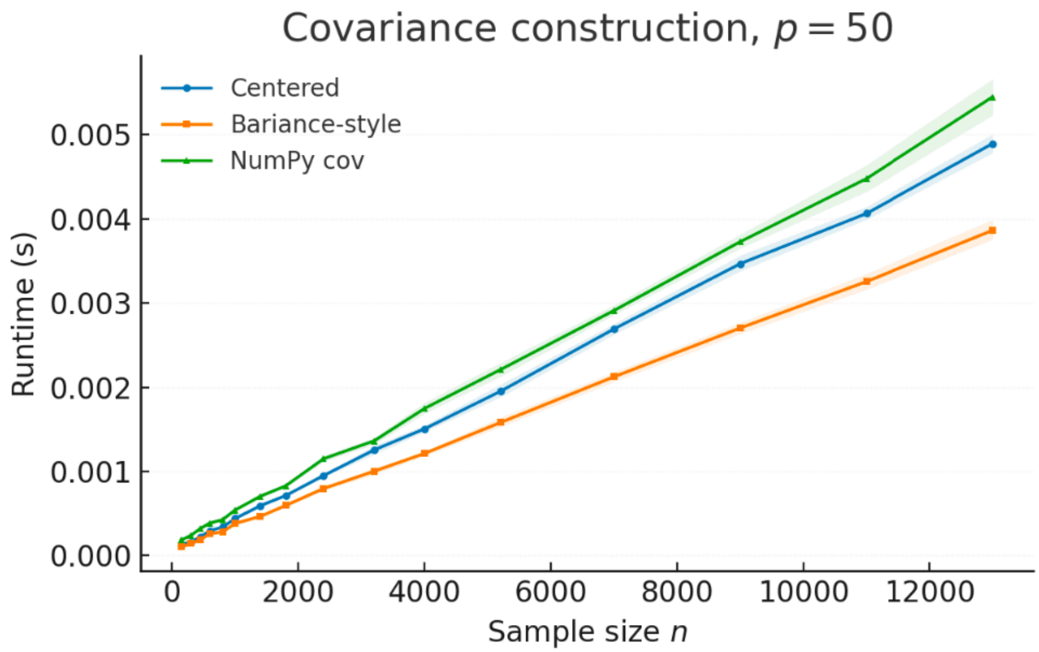

Figure 2. Runtime vs. n for $p = 50$. Same protocol.

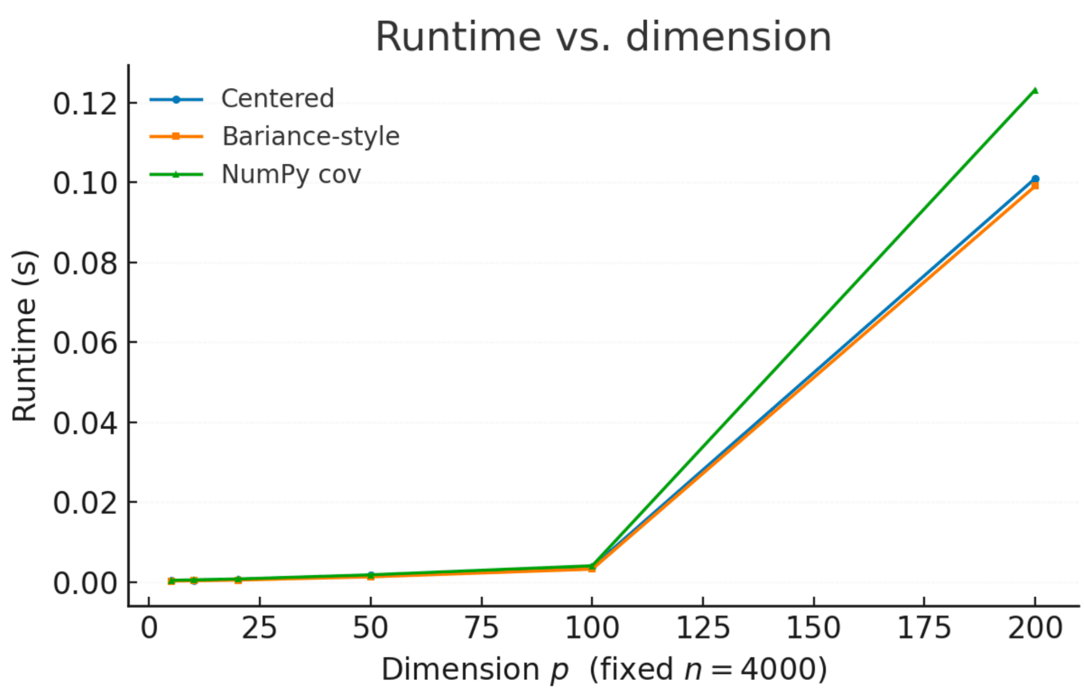

Figure 3. Runtime vs. p at fixed $n = 4000$.

doi.org/10.32388/TP9TDS

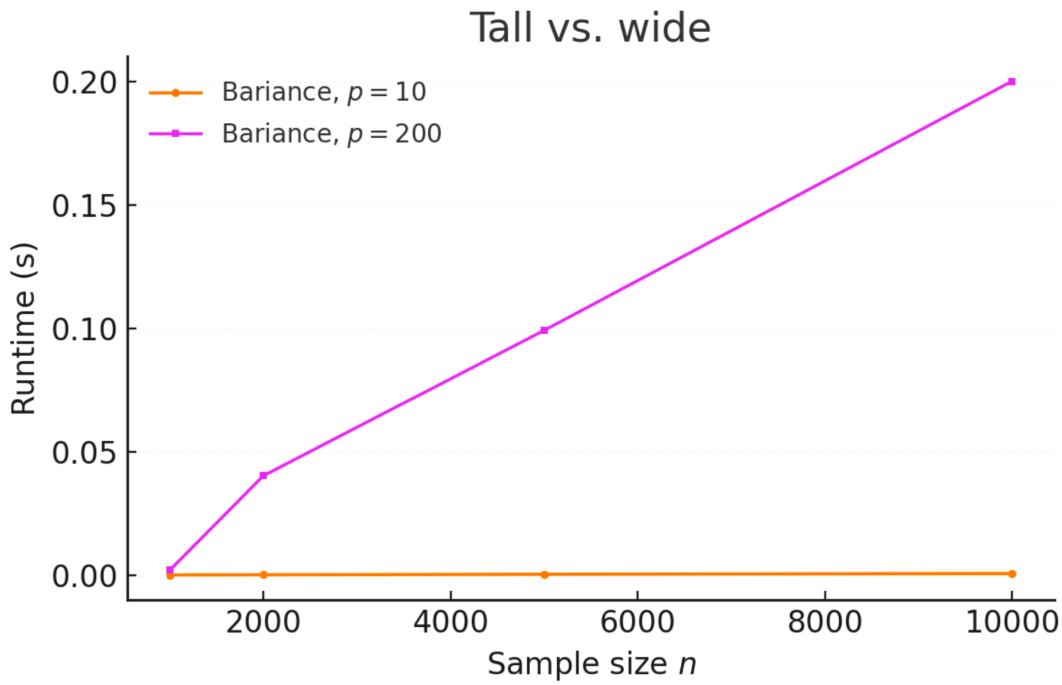

Figure 4. Bariance-only runtime for tall ($p = 10$) and wide ($p = 200$) matrices across several n.

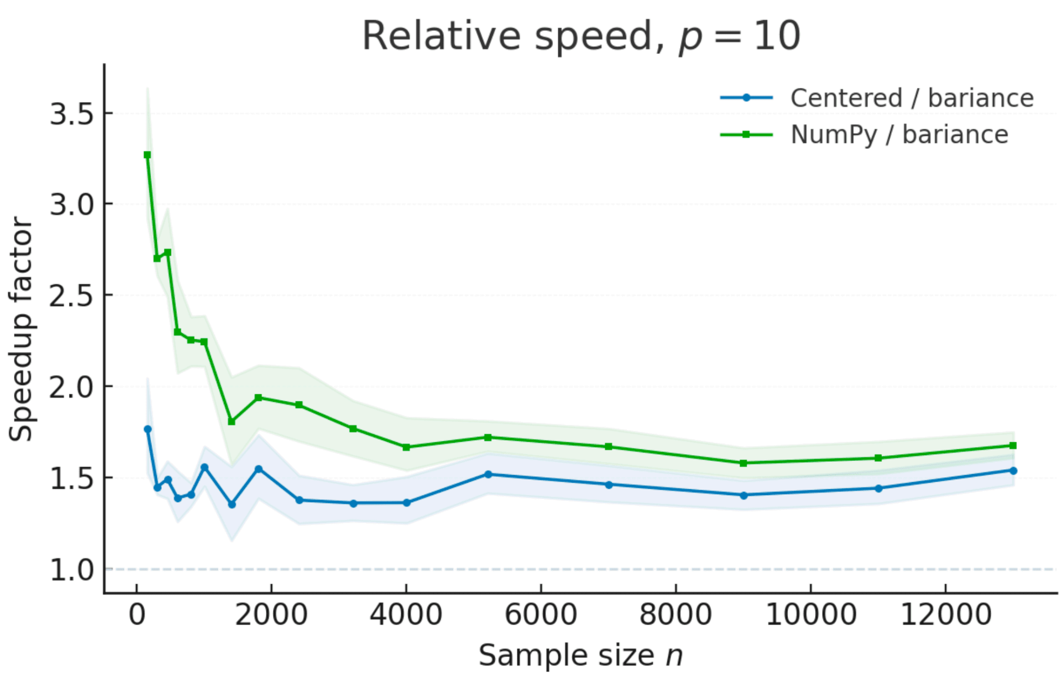

Figure 5. Relative speed up ratios for $p = 10$: centered/bariance and NumPy/bariance. Values > 1 favor the bariance form.

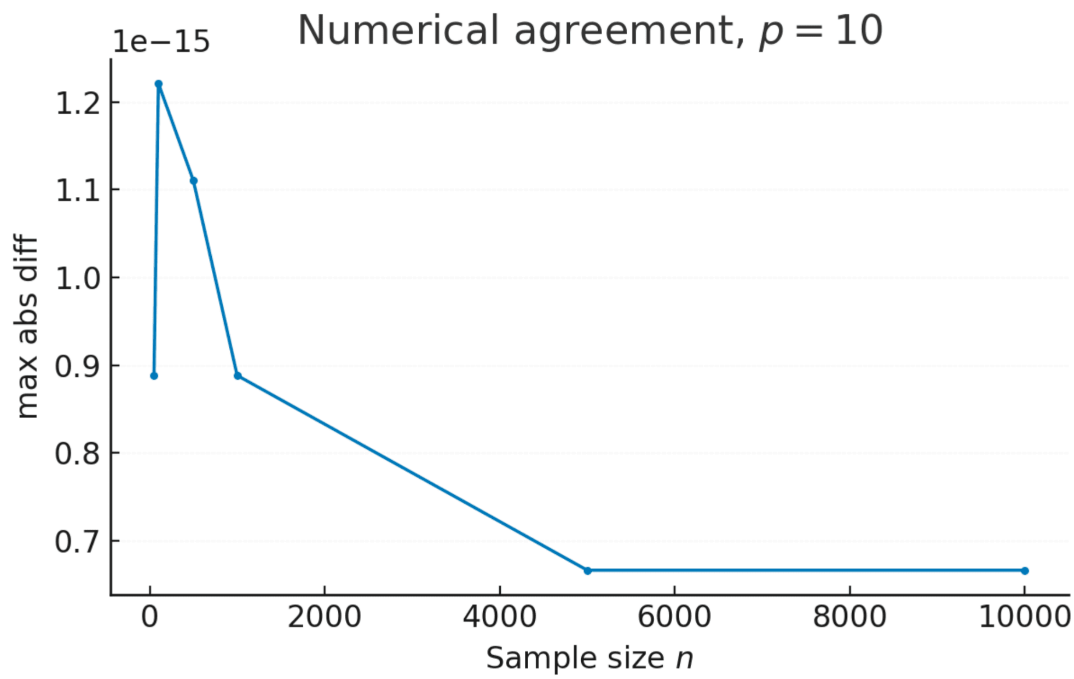

Figure 6. Numerical agreement: $\max_{k,\ell} |\Sigma_{\mathrm{Bar}} - \Sigma_{\mathrm{Ctr}}|$ for $p = 10$ across many draws. Values stay at floating-point noise.

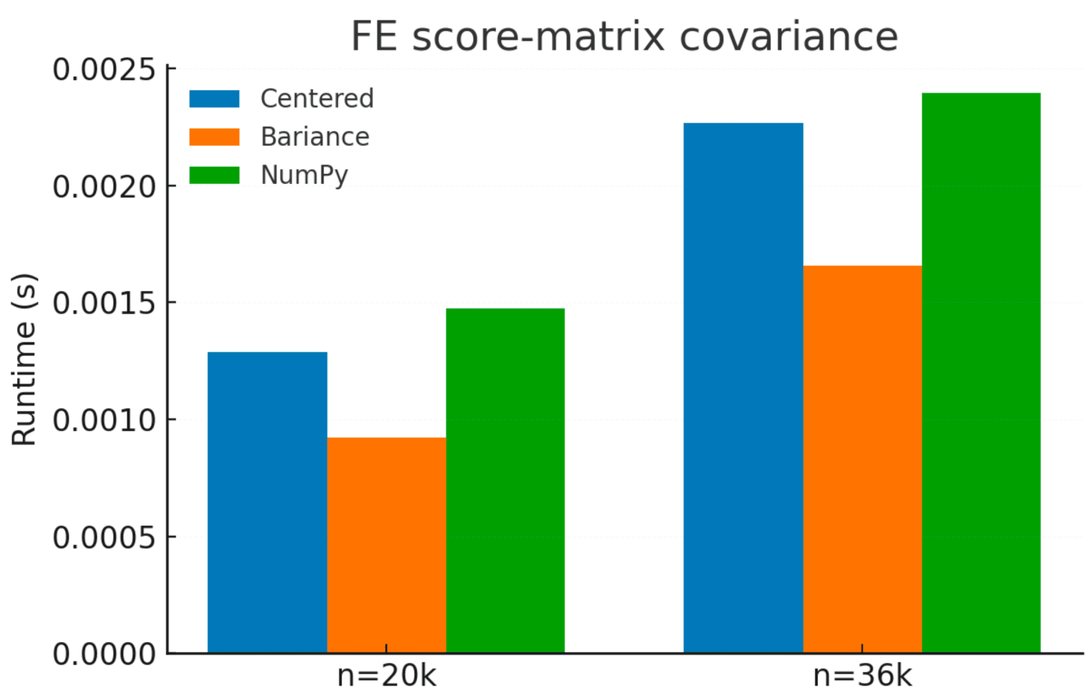

Figure 7. Fixed-effects score-matrix covariance timing, warmed and IQR-cleaned.

doi.org/10.32388/TP9TDS

10. Numerical equivalence in finite precision

For each simulated matrix $\mathbf{X} \in \mathbb{R}^{n\times p}$, compute

$$\mathbf{\Sigma}_{Bar} = \frac{1}{n(n-1)}\left(n\mathbf{X}^\top\mathbf{X} - \mathbf{s}\mathbf{s}^\top\right), \mathbf{\Sigma}_{Ctr} = \frac{1}{n-1}\left(\mathbf{X} - \mathbf{1}_n\overline{\mathbf{x}}^\top\right)^\top\left(\mathbf{X} - \mathbf{1}_n\overline{\mathbf{x}}^\top\right),$$

where $\mathbf{s} = \mathbf{X}^\top\mathbf{1}_n$ and $\overline{\mathbf{x}} = \frac{1}{n}\mathbf{s}$. The entrywise deviation

$$\Delta_{\max} = \|\mathbf{\Sigma}_{Bar} - \mathbf{\Sigma}_{Ctr}\|_{\max}$$

was recorded for a range of (n, p) combinations. Across all tested sizes, the maximum difference stayed below 10^{-12} in IEEE 754 double precision, consistent with rounding and cancellation limits reported in [8]. These findings confirm that the algebraic equality between the bariance-style and centered forms holds to full double-precision accuracy.

11. Applications

Sandwich covariances

Let $\mathbf{g}_i \in \mathbb{R}^p$ denote the score vector for observation i, stacked as rows in $\mathbf{G} \in \mathbb{R}^{n\times p}$, and let $\mathbf{H} = \mathbf{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top$. The empirical covariance of the scores can be written as

$$\hat{\mathbf{\Omega}}_{\text{sandwich}} = \frac{1}{n(n-1)}\left(n\mathbf{G}^\top\mathbf{G} - (\mathbf{G}^\top\mathbf{1}_n)(\mathbf{G}^\top\mathbf{1}_n)^\top\right) = \frac{1}{n-1}\mathbf{G}^\top\mathbf{H}\mathbf{G}. \quad (2)$$

The bariance-style and centered forms coincide algebraically. Expression (2) arises in variance formulas for M-estimators and their sandwich extensions [11][12][13][14][15]. The bariance-style version depends only on $\mathbf{G}^\top\mathbf{G}$ and $\mathbf{G}^\top\mathbf{1}_n$ and does not require forming $\mathbf{H}\mathbf{G}$.

Panel and fixed effects

For each unit in a panel with T periods, the within transformation is $\mathbf{M} = \mathbf{I}_T - \frac{1}{T}\mathbf{1}_T\mathbf{1}_T^\top$. Given $\mathbf{X}_i \in \mathbb{R}^{T\times p}$, the covariance after demeaning satisfies

$$\hat{\mathbf{\Sigma}}_i = \frac{1}{T-1}\mathbf{X}_i^\top\mathbf{M}\mathbf{X}_i = \frac{1}{T-1}\left(\mathbf{X}_i^\top\mathbf{X}_i - \frac{1}{T}\mathbf{s}_i\mathbf{s}_i^\top\right), \mathbf{s}_i = \mathbf{X}_i^\top\mathbf{1}_T.$$

This bariance-style identity applies to each block and matches the algebra of the within estimator [16][17].

Just-in-Time (JIT) streaming.

For sequential data, maintain cumulative quantities

$$\mathbf{S}_t = \sum_{i=1}^{t} \mathbf{x}_i \,, \mathbf{G}_t = \sum_{i=1}^{t} \mathbf{x}_i \mathbf{x}_i^\top .$$

The unbiased covariance at time $t \geq 2$ is

$$\mathbf{\Sigma}_t = \frac{1}{t(t-1)} \left(t\mathbf{G}_t - \mathbf{S}_t \mathbf{S}_t^\top \right) = \frac{1}{t-1} \left(\frac{1}{t} \sum_{i=1}^{t} \mathbf{x}_i \mathbf{x}_i^\top - \overline{\mathbf{x}}_t \overline{\mathbf{x}}_t^\top \right), \overline{\mathbf{x}}_t = \frac{1}{t} \mathbf{S}_t. \tag{3}$$

Each update adds one outer product $\mathbf{x}_t \mathbf{x}_t^\top$ and one rank-one correction, with cost $\mathcal{O}(p^2)$. Fast matrix routines for $\mathbf{X}\mathbf{X}^\top$ such as RXTX [1] provide measurable computational savings [18][19].

Bootstrap and resampling

For a resample with multiplicity weights $\mathbf{w} \in \mathbb{N}_0^n$ and $\mathbf{W} = diag(\mathbf{w})$, the covariance of the resample is

$$\hat{\mathbf{\Sigma}}^* = \frac{1}{n^* - 1} \left(\mathbf{X}^\top \mathbf{W} \mathbf{X} - \frac{1}{n^*} (\mathbf{X}^\top \mathbf{w})(\mathbf{X}^\top \mathbf{w})^\top \right), n^* = \sum_{i=1}^{n} w_i .$$

Each resample depends only on the weighted Gram pair $(\mathbf{X}^\top \mathbf{W} \mathbf{X}, \mathbf{X}^\top \mathbf{w})$. This form is practical when data are stored in aggregated or Gram form and aligns with generalized-inverse frameworks [20].

Appendix A. Elementwise derivations using matrix entries

For clarity, write out the (k, ℓ) entry of the covariance estimator from (4):

$$\Sigma_{k\ell} = \frac{1}{n(n-1)} \left(n \sum_{i=1}^{n} x_{ik} x_{i\ell} - \left(\sum_{i=1}^{n} x_{ik} \right) \left(\sum_{i=1}^{n} x_{i\ell} \right) \right). \tag{4}$$

The centered definition is

$$\frac{1}{n-1} \sum_{i=1}^{n} (x_{ik} - \bar{x}_k)(x_{i\ell} - \bar{x}_\ell) = \frac{1}{n-1} \left(\sum_{i=1}^{n} x_{ik} x_{i\ell} - \bar{x}_k \sum_{i=1}^{n} x_{i\ell} - \bar{x}_\ell \sum_{i=1}^{n} x_{ik} + n \bar{x}_k \bar{x}_\ell \right), \tag{5}$$

where $\bar{x}_k = (\sum_i x_{ik})/n$ and $\bar{x}_\ell = (\sum_i x_{i\ell})/n$.

Elementwise derivation

Substitute $\bar{x}_k$ and $\bar{x}_\ell$ into (5):

$$\frac{1}{n-1} \left(\sum_{i=1}^{n} x_{ik} x_{i\ell} - \frac{1}{n} \left(\sum_{i=1}^{n} x_{ik} \right) \left(\sum_{i=1}^{n} x_{i\ell} \right) - \frac{1}{n} \left(\sum_{i=1}^{n} x_{i\ell} \right) \left(\sum_{i=1}^{n} x_{ik} \right) + \frac{n}{n^2} \left(\sum_{i=1}^{n} x_{ik} \right) \left(\sum_{i=1}^{n} x_{i\ell} \right) \right).$$

Combine and simplify:

$$\frac{1}{n-1}\left(\sum_{i=1}^{n} x_{ik}x_{i\ell} - \frac{1}{n}\left(\sum_{i=1}^{n} x_{ik}\right)\left(\sum_{i=1}^{n} x_{i\ell}\right)\right) = \frac{1}{n(n-1)}\left(n\sum_{i=1}^{n} x_{ik}x_{i\ell} - \left(\sum_{i=1}^{n} x_{ik}\right)\left(\sum_{i=1}^{n} x_{i\ell}\right)\right),$$

matching (4). This completes the elementwise equivalence.

Matrix form

Let $\mathbf{X} \in \mathbb{R}^{n\times p}$, $\mathbf{s} = \mathbf{X}^\top \mathbf{1}_n$, and $\mathbf{H} = \mathbf{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top$. The covariance matrix is

$$\hat{\mathbf{\Sigma}} = \frac{1}{n-1}\mathbf{X}^\top\mathbf{H}\mathbf{X} = \frac{1}{n-1}\left(\mathbf{X}^\top\mathbf{X} - \frac{1}{n}\mathbf{s}\mathbf{s}^\top\right). \tag{6}$$

The (k,ℓ) entry of (6) equals (4), so the matrix and scalar derivations coincide.

Pairwise difference identity

Starting from

$$\hat{\Sigma}_{k\ell} = \frac{1}{n(n-1)}\sum_{1\le i<j\le n}(x_{ik}-x_{jk})(x_{i\ell}-x_{j\ell}),$$

use

$$\sum_{i<j} a_i a_j = \frac{1}{2}\left[\left(\sum_i a_i\right)^2 - \sum_i a_i^2\right],$$

on columns k and ℓ. After expansion and cancellation one obtains (4), and from (6) the compact form

$$\hat{\mathbf{\Sigma}} = \frac{1}{n-1}\left(\mathbf{X}^\top\mathbf{X} - \frac{1}{n}\mathbf{s}\mathbf{s}^\top\right).$$

Computational note

Equation (6) requires one Gram product $\mathbf{X}^\top\mathbf{X}$ and one outer product $\mathbf{s}\mathbf{s}^\top$. This avoids constructing the centered matrix $\mathbf{H}\mathbf{X}$ and keeps the cost to a single $p\times p$ multiplication and one subtraction. Fast Gram routines such as RXTX [1] for $\mathbf{X}\mathbf{X}^\top$ can further reduce runtime in high-dimensional cases.

Appendix B. Python and R references for replication

Python

A minimal implementation (MVP) of the optimized covariance estimator is:

```
cov_bar(X) = (n * (X.T @ X) - np.outer(s, s)) / (n * (n - 1))
```

with $s = X^\top \mathbf{1}_n$ and $n = X.\mathrm{shape}[0]$. This path triggers one level-3 BLAS multiply for $X^\top X$ (GEMM/DSYRK) and one rank-1 update for $ss^\top$ (GER), avoiding materialization of $X - \mathbf{1}_n \bar{x}^\top$. In many Python builds NumPy is linked to a generic or single-thread OpenBLAS; users often observe that raw @ calls are fast when BLAS is available, while higher-level helpers incur extra allocations and passes [21][22]. Identifying the active BLAS and confirming that np.dot/@ dispatch into it is standard practice [21]. Under these conditions the closed form can outpace numpy.cov, which centers then multiplies and moves an $n \times p$ temporary.

R (base)

An (at least numerically) equivalent base-R implementation is:

```
bariance_cov <- function(X) {
  n <- nrow(X)
  s <- colSums(X)
  G <- crossprod(X)
  (n * G - tcrossprod(s, s)) / (n * (n - 1))
 }
```

In R, crossprod/tcrossprod map directly to level-3 BLAS (e.g., DSYRK/DGEMM) and are already the preferred fast path [23][24]. Guidance on BLAS-backed functions in base R points the same way [25]. Base cov() is a compiled routine that performs centering and Gram products through BLAS, so its inner loops remain inside C/Fortran once data pointers are set. With a multithreaded vendor BLAS, this often beats an R-level function wrapper, even if that wrapper calls crossprod internally.

Outcome

Python: the closed form is faster when numpy.cov does extra allocations and the build is not strongly tuned; the single GEMM+GER path wins [21][22]. R: cov() already leverages BLAS through compiled code; cov() remains faster than an R-level wrapper that still returns to the interpreter between BLAS calls [23][24][25].

References

1. [a, b, c, d, e] *Rybin D, Zhang Y, Luo ZQ (2025). "XXT Can Be Faster." arXiv. doi:10.48550/arXiv.2505.09814.*
2. [^] *Cochran WG (1977). Sampling Techniques. 3rd ed. Wiley.*
3. [^] *Searle SR (2006). Matrix Algebra Useful for Statistics. Wiley.*
4. [^] *Harville DA (1997). Matrix Algebra from a Statistician's Perspective. Springer.*
5. [^] *Horn RA, Johnson CR (1985). Matrix Analysis. Cambridge University Press.*
6. [^] *Magnus JR, Neudecker H (1988). Matrix Differential Calculus with Applications in Statistics and Econometrics. Wiley.*
7. [a, b] *Demmel J (1997). Applied Numerical Linear Algebra. SIAM.*
8. [a, b, c, d] *Higham NJ (2002). Accuracy and Stability of Numerical Algorithms. 2nd ed. SIAM.*
9. [a, b] *Golub GH, Van Loan CF (2013). Matrix Computations. 4th ed. Johns Hopkins University Press.*
10. [^] *Reichel F (2025). "On Bessel's Correction: Unbiased Sample Variance, the Bariance, and a Novel Runtime-Optimized Estimator." arXiv. doi:10.48550/arXiv.2503.22333.*
11. [^] *Huber PJ (1967). "The Behavior of Maximum Likelihood Estimates Under Nonstandard Conditions." In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability.*
12. [^] *White H (1980). "A Heteroskedasticity-Consistent Covariance Matrix Estimator and a Direct Test for Heteroskedasticity." Econometrica. **48**(4):817–838.*
13. [^] *Hansen LP (1982). "Large Sample Properties of Generalized Method of Moments Estimators." Econometrica. **50**(4):1029–1054.*
14. [^] *Newey WK, West KD (1987). "A Simple, Positive Semi-Definite, Heteroskedasticity and Autocorrelation Consistent Covariance Matrix." Econometrica. **55**(3):703–708.*
15. [^] *Cameron AC, Gelbach JB, Miller DL (2006). "Robust Inference with Multi-Way Clustering." NBER Technical Working Paper No. 327.*
16. [^] *Arellano M (1987). "Computing Robust Standard Errors for Within-Groups Estimators." Oxf Bull Econ Stat. **49**(4):431–434.*
17. [^] *Wooldridge JM (2010). Econometric Analysis of Cross Section and Panel Data. 2nd ed. MIT Press.*
18. [^] *Anderson TW (2003). An Introduction to Multivariate Statistical Analysis. 3rd ed. Wiley.*
19. [^] *Tsay RS (2013). Multivariate Time Series Analysis: With R and Financial Applications. Wiley.*
20. [^] *Ben-Israel A, Greville TNE (2003). Generalized Inverses: Theory and Applications. 2nd ed. Springer.*

21. [a, b, c] *Stackoverflow community. "Find Out If / Which BLAS Library Is Used by NumPy." Stackoverflow community.* https://stackoverflow.com/questions/37184618/find-out-if-which-blas-library-is-used-by-numpy *(Accessed: 6 Nov 2025).*
22. [a, b] *Stackoverflow community. "How to Decrease the Output Time of Covariance Function." Stackoverflow community.* https://stackoverflow.com/questions/56057835/how-to-decrease-the-output-time-of-covariance-function *(Accessed: 6 Nov 2025).*
23. [a, b] *Stackoverflow community. "What Is R's Crossprod Function?" Stackoverflow community.* https://stackoverflow.com/questions/15162741/what-is-rs-crossproduct-function *(Accessed: 6 Nov 2025).*
24. [a, b] *Stackoverflow community. "Any Faster R Function Than Tcrossprod for Symmetric Dense Matrix Multiplication?" Stackoverflow community.* https://stackoverflow.com/questions/39532187/r-any-faster-r-function-than-tcrossprod-for-symmetric-dense-matrix-multiplica *(Accessed: 6 Nov 2025).*
25. [a, b] *Stackoverflow community. "Which R Functions Are Based Exactly on BLAS, ATLAS, LAPACK, and So On?" Stackoverflow community.* https://stackoverflow.com/questions/47440244/which-r-functions-are-based-exactly-on-blas-atlas-lapack-and-so-on *(Accessed: 6 Nov 2025).*

Declarations

Funding: No specific funding was received for this work.

Potential competing interests: No potential competing interests to declare.

doi.org/10.32388/TP9TDS