TRAGULA
Volume 0, Issue 0, 0000
doi: 10.5555/12345678
https://github.com/RobinHankin/quadform

# Quadratic forms in R: introducing the quadform package

**Robin K. S. Hankin**
University of Stirling

**Abstract**

The **quadform** package evaluates a range of quadratic forms, using efficient methods. Unnecessary transposes are not performed. Complex values are handled consistently.

*Keywords*: Quadratic forms.



## 1. Overview


Quadratic forms are polynomials with all terms of degree 2. Given a column vector $\mathbf{x} = (x_1, \ldots, x_n)^\top$ and an $n \times n$ matrix $M$ then the function $f\colon \mathbb{R}^n \longrightarrow \mathbb{R}$ given by $f(\mathbf{x}) = \mathbf{x}^\top M \mathbf{x}$ is a quadratic form (Gentle 2024); we extend to complex vectors by mapping $\mathbf{z} = (z_1, \ldots, z_n)^\top$ to $\mathbf{z}^* M \mathbf{z}$, where $\mathbf{z}^*$ means the complex conjugate of $\mathbf{z}^\top$, that is $\overline{\mathbf{z}}^\top$. These are implemented in the package with `quad.form(M, x)` which is essentially

```
> function(M,x){crossprod(crossprod(M, Conj(x)), x)}
```

Using `quad.form(M, x)` is preferable to `t(x) %*% M %*% x` on several grounds. Firstly, it streamlines and simplifies code; secondly, it is more efficient; and thirdly, the package handles the complex case consistently in that matrix transposes become Hermitian transposes. For example, `cprod(x, y)` returns `Conj(t(x)) %*% y`, as one would usually want—compare `crossprod(x, y)` which evaluates `t(x) %*% M`. Note that **quadform** functions generally avoid `%*%` and use `crossprod()` and `tcrossprod()` whenever possible: `cprod()` is essentially `crossprod(Conj(x), y)`. This general design principle ensures that if $M$ is Hermitian symmetric, `quad.form(M, x)` is real for any complex vector `x`: `quad.form(M, x)` returns `Conj(t(x)) %*% M %*% x`.

The package includes similar functionality for other related expressions such as $\mathbf{x}^* M \mathbf{y}$ and $\mathbf{x}^* M^{-1} \mathbf{x}$. The package also has functionality for evaluating quadratic forms of matrices.

| package idiom | definition | base R idiom | terse form |
|---|---|---|---|
| `ht(x)` | $\mathbf{x}^* = \overline{\mathbf{x}^T}$ | `Conj(t(x))` | `ht(x)` |
| `cprod(x,y)` | $\mathbf{x}^*\mathbf{y}$ | `ht(x) %*% y` | `cp()` |
| `tcprod(x,y)` | $\mathbf{x}\mathbf{y}^*$ | `x %*% ht(y)` | `tcp()` |
| `quad.form(M,x)` | $\mathbf{x}^*M\mathbf{x}$ | `ht(x) %*% M %*% x` | `qf()` |
| `quad.form.inv(M,x)` | $\mathbf{x}^*M^{-1}\mathbf{x}$ | `ht(x) %*% solve(M) %*% x` | `qfi()` |
| `quad.tform(M,x)` | $\mathbf{x}M\mathbf{x}^*$ | `x %*% A %*% ht(x)` | `qt()` |
| `quad.tform.inv(M,x)` | $\mathbf{x}M^{-1}\mathbf{x}^*$ | `x %*% solve(M) %*% ht(x)` | `qti()` |
| `quad3.form(M,l,r)` | $\mathbf{l}^*M\mathbf{r}$ | `t(l) %*% M %*% r` | `q3()` |
| `quad3.form.inv(M,l,r)` | $\mathbf{l}^*M^{-1}\mathbf{r}$ | `t(l) %*% solve(M) %*% r` | `q3i()` |
| `quad3.tform(M,l,r)` | $\mathbf{l}M\mathbf{r}^*$ | `l %*% M %*% t(r)` | `q3t()` |
| `quad.diag(M,x)` | $\mathrm{diag}(\mathbf{x}^*M\mathbf{x})$ | `diag(quad.form(M,x))` | `qd()` |
| `quad.tdiag(M,x)` | $\mathrm{diag}(\mathbf{x}M\mathbf{x}^*)$ | `diag(quad.tform(M,x))` | `qtd()` |
| `quad3.diag(M,l,r)` | $\mathrm{diag}(\mathbf{l}^*M\mathbf{r})$ | `diag(quad3.form(M,l,r))` | `q3d()` |
| `quad3.tdiag(M,l,r)` | $\mathrm{diag}(\mathbf{l}M\mathbf{r}^*)$ | `diag(quad3.tform(M,l,r))` | `q3td()` |
| `quad.trace(M,x)` | $\mathrm{tr}(\mathbf{x}^*M\mathbf{x})$ | `tr(quad.form(M,x))` | `qt()` |
| `quad.ttrace(M,x)` | $\mathrm{tr}(\mathbf{x}M\mathbf{x}^*)$ | `tr(quad.tform(M,x))` | `qtt()` |

Table 1: Quadratic forms. Here, $\mathbf{x}^*$ consistently denotes the *complex conjugate* of the transpose, also known as the Hermitian transpose

Given $M_{[n\times n]}$ and $\mathbf{x}_{[n\times k]}$, then $F = \mathbf{x}^*M\mathbf{x}$ is $[k \times k]$, and sometimes one needs $\mathrm{diag}(F)$ or $\mathrm{trace}(F)$. These are implemented as `quad.diag()` and `quad.trace()` respectively. A list is provided in table 1.

The main motivation for the package is nicer code. For example, the **emulator** package (Hankin 2005) has to manipulate the following expression:

$$\left[H_x - H^\top A^{-1}U\right]^\top \left[H^\top \left(H^\top A^{-1}H\right)^{-1} H\right] \left[H_x - H^\top A^{-1}U\right].$$

Direct R idiom would be:

```
> t(Hx - t(H) %*% solve(A) %*% U) %*%
+ t(H) %*% solve(t(H) %*% solve(A) %*% H) %*% H %*%
+ (Hx - t(H) %*% solve(A) %*% U)
```

But **quadform** idiom is:

```
> quad.form(quad.form.inv(quad.form.inv(A,H),H), Hx - quad3.form.inv(A,H,U))
```

and in terse form becomes:

```
> qf(qfi(qfi(A,H),H), Hx - q3fi(A,H,U))
```

which is certainly shorter, arguably more elegant, and possibly faster; it also operates correctly in the complex case, as needed for working with the complex multivariate Gaussian distribution (Hankin 2015).

## References

Gentle JE (2024). *Matrix Algebra*. Third edition. Springer.

Hankin RKS (2005). "Introducing **BACCO**, an R bundle for Bayesian analysis of computer code output." *Journal of Statistical Software*, **14**(16).

Hankin RKS (2015). "The complex multivariate Gaussian distribution." *The R Journal*, **7**(1), 73–80.

### Affiliation:

Robin K. S. Hankin
University of Stirling
Scotland
email: hankin.robin@gmail.com


*Submitted:* 0000-00-00
*Accepted:* 0000-00-00