Package 'GFT'

September 22, 2026

Type Package

Title Generalized Fisher Transformation of Correlation Matrices

Version 1.2.0

Description Forward and inverse generalized Fisher transformation (GFT) of correlation matrices, gamma = vecl(log C), which maps the positive definite correlation matrices one-to-one onto the Euclidean space of dimension n(n-1)/2, see Archakov and Hansen (2021) <doi:10.3982/ECTA16910>. The inverse is computed from a variational characterization by the GFT-FP+N algorithm: matrix-free inexact Newton steps for the log-diagonal residual, solved by preconditioned conjugate gradients, with fixed-point safeguards that guarantee global convergence. Sequential inversion with a tangent predictor, a certified quadrature preconditioner, and comparison solvers (the plain fixed point, Broyden's method, full Newton, Anderson acceleration, and limited-memory BFGS) are included. Uses base R only.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 3.5.0)

Suggests testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

URL https://github.com/reinhardhansen/GFT

BugReports https://github.com/reinhardhansen/GFT/issues

NeedsCompilation no

Author Ilya Archakov [aut],
Peter Reinhard Hansen [aut, cre]

Maintainer Peter Reinhard Hansen <hansen@unc.edu>

Repository CRAN

Date/Publication 2026-09-22 20:10:49 UTC

Contents

`GFT-package` *Generalized Fisher Transformation of Correlation Matrices*

Description

Forward and inverse generalized Fisher transformation (GFT) of correlation matrices. The GFT maps a non-singular $n \times n$ correlation matrix C to the real vector $\gamma = \mathrm{vecl}(\log C)$ of below-diagonal elements of the matrix logarithm of C. The map is a bijection between the set of positive definite correlation matrices and R^d with $d = n(n-1)/2$ (Archakov and Hansen, 2021), and generalizes Fisher's z-transformation, to which it reduces for $n = 2$.

Details

The forward map is computed by `gft`. The inverse is computed from the variational characterization

$$x^*(z) = \arg\min_x \ \mathrm{tr}\, e^{A[x;z]} - \sum_i x_i,$$

where $A[x; z]$ is symmetric with off-diagonal elements z and diagonal x, by the following solvers:

`inv_gft` GFT-FP+N (recommended): matrix-free inexact Newton steps for the log-diagonal residual via preconditioned conjugate gradients, with fixed-point safeguards and exact normalization; optional certified quadrature preconditioner.

`inv_gft_path` Sequential inversion with warm starts and the tangent predictor `gft_predict`.

`inv_gft_fp` The Archakov-Hansen fixed point.

`inv_gft_broyden` Broyden's method as in Chen, Fei and Yu (2025).

`inv_gft_newton` Full Newton with the exact $O(n^4)$ Hessian.

`inv_gft_anderson` Anderson acceleration of the fixed point, plain or guarded.

`inv_gft_lbfgs` Limited-memory BFGS on the objective.

The implementation is a line-faithful port of version 1.2.0 of the Julia reference implementation by the same authors (`https://github.com/reinhardhansen/GFT`) and uses base R only.

Author(s)

Ilya Archakov and Peter Reinhard Hansen.

Maintainer: Peter Reinhard Hansen <hansen@unc.edu>

References

Archakov, I. and Hansen, P. R. (2021). A new parametrization of correlation matrices. *Econometrica*, **89**, 1699–1715. doi:10.3982/ECTA16910

Archakov, I. and Hansen, P. R. (2026). Fast inversion of the generalized Fisher transformation of correlation matrices. Working paper, arXiv:2609.19028.

Chen, H., Fei, Y. and Yu, J. (2025). Multivariate stochastic volatility models based on generalized Fisher transformation. *Journal of Econometrics*, **251**, 106041.

Examples

```
C <- 0.9^abs(outer(1:5, 1:5, "-"))   # Toeplitz correlation matrix
z <- gft(C)                          # forward transformation
r <- inv_gft(z)                      # inverse transformation
max(abs(r$C - C))                    # round trip at machine precision
```

---

gft *Generalized Fisher Transformation*

Description

Computes the generalized Fisher transformation $\gamma = \mathrm{vecl}(\log C)$: the below-diagonal elements, stacked column by column, of the matrix logarithm of a positive definite correlation matrix C.

Usage

```
gft(C)
```

Arguments

C	a positive definite correlation matrix (square, symmetric, numeric). The symmetric part $(C + C')/2$ is used.

Details

The matrix logarithm is computed from the eigendecomposition of C. For $n = 2$ the transformation reduces to Fisher's classical z-transformation $z = \mathrm{atanh}(\rho)$. The map is a bijection between the positive definite correlation matrices and $R^{n(n-1)/2}$; its inverse is computed by `inv_gft`.

Value

A numeric vector of length $n(n-1)/2$ containing $\mathrm{vecl}(\log C)$.

References

Archakov, I. and Hansen, P. R. (2021). A new parametrization of correlation matrices. *Econometrica*, **89**(4), 1699–1715. doi:10.3982/ECTA16910

See Also

`inv_gft`, `vecl`, `unvecl`.

Examples

```
# n = 2: reduces to Fisher's z-transformation
C <- matrix(c(1, 0.5, 0.5, 1), 2, 2)
all.equal(gft(C), atanh(0.5))

C <- 0.9^abs(outer(1:5, 1:5, "-"))
z <- gft(C)
max(abs(inv_gft(z)$C - C))
```

`inv_gft` *Inverse Generalized Fisher Transformation (GFT-FP+N)*

Description

Reconstructs the unique positive definite correlation matrix C with $\mathrm{vecl}(\log C) = z$ by the GFT-FP+N algorithm (recommended solver).

Usage

```
inv_gft(z, x0 = NULL, tol = 1e-13, maxit = 500, delta = 1,
        exact_hess = FALSE, residual = c("log", "gradient"),
        forcing = c("quad", "sqrt"), normalize = TRUE, phase = FALSE,
        adaptive = TRUE,
        preconditioner = c("diagonal", "quadrature", "auto"),
        kappa = 2, rmin = 2, rmax = 8, nmin = 0)
```

Arguments

`z`	numeric vector of length $n(n-1)/2$: the below-diagonal elements of $\log C$, stacked column by column.
`x0`	optional numeric vector of length n: starting value for the diagonal of $\log C$ (e.g. from a previous solution, for warm starts). Defaults to zero.
`tol`	convergence tolerance on $\|\mathrm{diag}(e^A) - 1\|_\infty$.
`maxit`	maximum number of iterations.
`delta`	threshold of the optional initial fixed-point phase (`phase = TRUE`): fixed-point steps are used while $\|\mathrm{diag}(e^A) - 1\|_\infty > \delta$.

`exact_hess` if `TRUE`, solve the Newton system with the explicit $O(n^4)$ Hessian and a Cholesky factorization instead of matrix-free conjugate gradients. Identical algorithm otherwise; mainly for benchmarking.

`residual` `"log"` (default): Newton steps for the equation $\ell(x) = 0$, $\ell = \log \operatorname{diag}(e^A)$, i.e. the system $H\delta = -D\ell$; `"gradient"`: Newton steps for $\nabla f = 0$, the system $H\delta = -g$.

`forcing` inexact-Newton forcing tolerance: $\eta = \min(1/2, \|r\|)$ (`"quad"`, locally quadratic) or $\min(1/2, \|r\|^{1/2})$ (`"sqrt"`, order 3/2), with r the outer residual.

`normalize` if `TRUE`, every evaluated point is shifted along the vector of ones so that $\operatorname{tr} e^A = n$, the exact minimizer of f along that direction; costs no eigendecomposition and removes overflow.

`phase` if `TRUE`, take fixed-point steps while $\max_i \ell_i > \log(1 + \delta)$ before the Newton phase.

`adaptive` if `TRUE`, start each Armijo backtracking at $\min(1, 2t_{\text{prev}})$, with t_{prev} the previously accepted step; otherwise at 1.

`preconditioner` preconditioner of the conjugate-gradient solve: the diagonal $D = \operatorname{diag}(e^\ell)$ (`"diagonal"`, default); the certified quadrature model M_r at every Newton step (`"quadrature"`); or the selection rule (`"auto"`): M_r whenever an order in `[rmin, rmax]` certifies $\kappa(M_r^{-1}H) \le$ `kappa` and $n \ge$ `nmin`, the diagonal otherwise. The forcing test is unchanged.

`kappa`, `rmin`, `rmax`, `nmin`
parameters of the quadrature preconditioner and its selection rule; see `preconditioner`.

Details

The solution solves the strictly convex problem

$$x^*(z) = \arg\min_x \ \operatorname{tr} e^{A[x;z]} - \textstyle\sum_i x_i,$$

where $A[x; z]$ is symmetric with off-diagonal elements z and diagonal x. Every evaluated point is normalized by the exact minimization of f along the vector of ones. Fixed-point steps $x \leftarrow x - \ell$, $\ell = \log \operatorname{diag}(e^A)$, are taken while some $\ell_i < -700$; otherwise an inexact Newton step for the equation $\ell(x) = 0$ is computed: the system $H\delta = -D\ell$, $D = \operatorname{diag}(e^\ell)$, is solved matrix-free by conjugate gradients with preconditioner D, exact Hessian-vector products (two matrix multiplications each), and forcing tolerance $\eta = \min(1/2, \|\ell\|)$ on the preconditioned residual (locally quadratic; Dembo, Eisenstat and Steihaug, 1982), capped at $2n$ products. A non-finite or non-descent direction is replaced by a fixed-point step; the full step is taken untested when its predicted decrease is below the floating-point resolution of f and $\|\ell\|_\infty < 10^{-3}$; otherwise Armijo backtracking on f, with a fixed-point step substituted if the search fails. Near the rounding floor the computation is finished by fixed-point steps. Every fixed-point step and every tested Newton step decreases f, which gives global convergence.

The options `residual`, `forcing`, `normalize`, `phase` and `adaptive` reproduce the variants of the ablation in the paper; `residual = "gradient"`, `forcing = "sqrt"`, `normalize = FALSE`, `phase = TRUE`, `adaptive = FALSE` is the version of GFT-FP+N in package versions 1.0.x.

The quadrature preconditioner replaces D by $M_r = \sum_j a_j e^{t_j A} \circ e^{(1-t_j)A}$ for the r-point Gauss-Legendre rule on $[0, 1]$, built from the same eigendecomposition at r matrix multiplications and

one Cholesky factorization. It satisfies $M_r \preceq H \preceq \phi_r(\Delta) M_r$ with Δ the spectral spread of A, so the order is chosen from the extreme eigenvalues; it pays whenever two or more nodes are needed, increasingly with n.

Value

An object of class `"gft_inv"`: a list with components

`x`	the solution: diagonal of $\log C$.
`C`	the reconstructed correlation matrix $e^{A[x^*;z]}$.
`iters`	number of iterations.
`eighs`	number of eigendecompositions, the dominant $O(n^3)$ kernel.
`hvs`	number of Hessian-vector products.
`err`	final value of $\|\mathrm{diag}(e^A) - 1\|_\infty$.
`converged`	logical.
`hist`	the error after each iteration.

References

Archakov, I. and Hansen, P. R. (2021). A new parametrization of correlation matrices. *Econometrica*, **89**(4), 1699–1715. doi:10.3982/ECTA16910

Archakov, I. and Hansen, P. R. (2026). Fast inversion of the generalized Fisher transformation of correlation matrices. Working paper, arXiv:2609.19028.

Dembo, R. S., Eisenstat, S. C. and Steihaug, T. (1982). Inexact Newton methods. *SIAM Journal on Numerical Analysis*, **19**, 400–408.

Eisenstat, S. C. and Walker, H. F. (1996). Choosing the forcing terms in an inexact Newton method. *SIAM Journal on Scientific Computing*, **17**, 16–32.

See Also

`gft` for the forward map; `inv_gft_fp`, `inv_gft_broyden`, `inv_gft_newton`, `inv_gft_anderson`, `inv_gft_lbfgs` for the comparison solvers; `inv_gft_path` for sequential inversion with the tangent predictor.

Examples

```
C <- 0.9^abs(outer(1:5, 1:5, "-"))
z <- gft(C)
r <- inv_gft(z)
r
max(abs(r$C - C))

# warm start from a nearby solution
z2 <- z + 0.01
r2 <- inv_gft(z2, x0 = r$x)

# an arbitrary vector in R^d is always a valid input
set.seed(1)
```

```
ra <- inv_gft(rnorm(45, sd = 2))   # n = 10
range(diag(ra$C))                  # unit diagonal
min(eigen(ra$C)$values) > 0        # positive definite

# the quadrature preconditioner halves the Hessian-vector products on
# ill-conditioned inputs
b <- 0.8 + 0.195 * runif(60)
zb <- gft(b %*% t(b) + diag(1 - b^2))
c(diagonal = inv_gft(zb)$hvs,
  quadrature = inv_gft(zb, preconditioner = "quadrature")$hvs)
```

`inv_gft_anderson` *Anderson Acceleration of the GFT Fixed Point*

Description

Anderson acceleration (type II, window `m`) of the fixed-point map $x \mapsto x - \log \operatorname{diag}(e^{A[x;z]})$, with an optional safeguard that makes it globally convergent. A Jacobian-free comparator for `inv_gft`.

Usage

```
inv_gft_anderson(z, x0 = NULL, tol = 1e-13, maxit = 5000, m = 5,
                 guarded = FALSE, theta = 0.25)
```

Arguments

`z`, `x0`, `tol`, `maxit`	as in `inv_gft`.
`m`	memory: number of stored differences of iterates and residuals.
`guarded`	if `TRUE`, an Anderson proposal y is accepted only if $f(y) \le f(x) - \theta V(x)$, where $V(x) = \sum_i (e^{\ell_i} - 1 - \ell_i)$ is the guaranteed decrease of the fixed-point step; otherwise the memory is cleared and a fixed-point step is taken. Proposals are accepted untested once $\theta V(x)$ is below the floating-point resolution of f.
`theta`	fraction of the fixed-point decrease required by the safeguard.

Details

In the difference form of Walker and Ni (2011) each iteration costs one eigendecomposition and a small least-squares solve. The ordinary scheme has no convergence guarantee and can diverge for large `m`; with `guarded = TRUE` every step decreases f by at least $\theta V(x)$, which implies convergence from every starting point (Archakov and Hansen, 2026, Section S5).

Value

An object of class `"gft_inv"`; see `inv_gft`. The `hvs` component is zero.

References

Walker, H. F. and Ni, P. (2011). Anderson acceleration for fixed-point iterations. *SIAM Journal on Numerical Analysis*, **49**, 1715–1735.

Archakov, I. and Hansen, P. R. (2026). Fast inversion of the generalized Fisher transformation of correlation matrices. Working paper, arXiv:2609.19028.

See Also

inv_gft, inv_gft_fp.

Examples

```
set.seed(2)
z <- rnorm(45, sd = 2)                # n = 10
r <- inv_gft_anderson(z, guarded = TRUE)
r
max(abs(r$x - inv_gft(z)$x)) < 1e-9
```

---

inv_gft_broyden *Inverse GFT by Broyden's Method*

Description

Reconstructs the correlation matrix C with $\mathrm{vecl}(\log C) = z$ by Broyden's method applied to the residual $F(x) = \log \mathrm{diag}(e^{A[x;z]})$, as in Chen, Fei and Yu (2025). Reference implementation for benchmarking against inv_gft.

Usage

```
inv_gft_broyden(z, x0 = NULL, tol = 1e-13, maxit = 500, warm = 1,
                globalized = FALSE)
```

Arguments

z	numeric vector of length $n(n-1)/2$.
x0	optional starting value of length n; defaults to zero.
tol	convergence tolerance on $\|\mathrm{diag}(e^A) - 1\|_\infty$.
maxit	maximum number of iterations.
warm	number of initial fixed-point steps before the Jacobian is formed (ignored when globalized = TRUE).
globalized	if TRUE, replace the one-step initialization with the same log-domain fixed-point phase as GFT-FP+N (fixed-point steps until $\max_i \ell_i \le \log 2$), and only then form the Jacobian.

Details

The exact Jacobian is computed once (an $O(n^4)$ Hessian), then updated by rank-one Sherman-Morrison updates of its inverse, with one eigendecomposition per iteration and no line search. Without globalization the method can diverge for large $\|z\|$; divergence is reported gracefully via `converged = FALSE`.

Value

An object of class `"gft_inv"`; see `inv_gft` for the components. On divergence the result has `converged = FALSE` and `err = Inf`.

References

Chen, H., Fei, Y. and Yu, J. (2025). Multivariate stochastic volatility models based on generalized Fisher transformation. *Journal of Econometrics*, **251**, 106041.

See Also

`inv_gft`.

Examples

```
z <- gft(0.9^abs(outer(1:5, 1:5, "-")))
r <- inv_gft_broyden(z)
r$converged
```

`inv_gft_fp` *Inverse GFT by the Archakov-Hansen Fixed Point*

Description

Reconstructs the correlation matrix C with $\mathrm{vecl}(\log C) = z$ by the fixed-point iteration of Archakov and Hansen (2021), $x \leftarrow x - \log \mathrm{diag}(e^{A[x;z]})$, evaluated in the log domain throughout.

Usage

```
inv_gft_fp(z, x0 = NULL, tol = 1e-13, maxit = 5000)
```

Arguments

`z`	numeric vector of length $n(n-1)/2$.
`x0`	optional starting value of length n; defaults to zero.
`tol`	convergence tolerance on $\|\mathrm{diag}(e^A) - 1\|_\infty$.
`maxit`	maximum number of iterations.

Details

Globally convergent, with one eigendecomposition per iteration. The local linear rate degrades as the spectrum of C spreads; `inv_gft` switches to an inexact Newton phase precisely to avoid this slowdown while retaining the fixed point's robustness.

Value

An object of class `"gft_inv"`; see `inv_gft` for the components.

References

Archakov, I. and Hansen, P. R. (2021). A new parametrization of correlation matrices. *Econometrica*, **89**(4), 1699–1715. doi:10.3982/ECTA16910

See Also

`inv_gft`.

Examples

```
z <- gft(0.9^abs(outer(1:5, 1:5, "-")))
r <- inv_gft_fp(z)
r$eighs
```

---

inv_gft_lbfgs *Limited-Memory BFGS for the Inverse GFT*

Description

Limited-memory BFGS on the objective $f(x) = \operatorname{tr} e^{A[x;z]} - \sum_i x_i$, with two-loop recursion, Armijo backtracking on f and the untested-step rule of `inv_gft` near the rounding floor. A general-purpose comparator.

Usage

```
inv_gft_lbfgs(z, x0 = NULL, tol = 1e-13, maxit = 500, m = 10,
              globalized = FALSE)
```

Arguments

`z`, `x0`, `tol`, `maxit`	as in `inv_gft`.
`m`	memory of the quasi-Newton update.
`globalized`	if `TRUE`, precede the iteration by fixed-point steps in the log domain until $\max_i \ell_i \le \log 2$.

Value

An object of class `"gft_inv"`; see `inv_gft`.

References

Liu, D. C. and Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, **45**, 503–528.

See Also

inv_gft, inv_gft_anderson.

Examples

```
set.seed(3)
z <- rnorm(45, sd = 2)
r <- inv_gft_lbfgs(z, globalized = TRUE)
c(lbfgs = r$eighs, gft_fpn = inv_gft(z)$eighs)
```

---

inv_gft_newton *Inverse GFT by Full Newton*

Description

Reconstructs the correlation matrix C with $\mathrm{vecl}(\log C) = z$ by a full Newton method with the exact $O(n^4)$ Hessian recomputed at every iteration, Armijo backtracking on the objective, and an optional fixed-point warm start. Reference implementation for benchmarking against inv_gft.

Usage

```
inv_gft_newton(z, x0 = NULL, tol = 1e-13, maxit = 500, warm = 1,
               safeguard = TRUE)
```

Arguments

z	numeric vector of length $n(n-1)/2$.
x0	optional starting value of length n; defaults to zero.
tol	convergence tolerance on $\|\mathrm{diag}(e^A) - 1\|_\infty$.
maxit	maximum number of iterations.
warm	number of initial fixed-point steps.
safeguard	logical. If TRUE (the default), apply the two rounding-floor safeguards described below. If FALSE, reproduce the published comparator with only the Armijo line search added, which may stagnate short of tol.

Details

Each iteration forms the exact Hessian in $O(n^4)$ time and solves the Newton system by Cholesky factorization; if the factorization fails or the line search cannot make progress, a fixed-point step is substituted. `inv_gft` obtains the same fast local convergence at $O(n^3)$ cost per iteration by solving the Newton system inexactly and matrix-free.

Near the solution the objective $f = \mathrm{tr}(e^A) - 1'x$ is flat to within cancellation, so the Armijo condition carries no information: the line search exhausts all of its halvings, the fixed-point fallback is taken, and the iteration can lock at an error several orders of magnitude above the attainable floor. With `safeguard = TRUE` two further rules prevent this. The full Newton step is taken untested once the predicted decrease falls below the floating-point resolution of f, and a persistent lack of progress below 10^{-9} hands the iteration to the contractive fixed point.

`safeguard = FALSE` disables both and reproduces the comparator benchmarked in the reference below, namely the method of Chen, Fei and Yu (2025) with only the Armijo line search added. In that mode `converged` can be `FALSE` on a small fraction of otherwise well conditioned problems; this is the documented behaviour of that variant, not a defect of the implementation.

Value

An object of class `"gft_inv"`; see `inv_gft` for the components.

References

Archakov, I. and Hansen, P. R. (2026). Fast inversion of the generalized Fisher transformation of correlation matrices. Working paper, arXiv:2609.19028.

See Also

`inv_gft`.

Examples

```
z <- gft(0.9^abs(outer(1:5, 1:5, "-")))
r <- inv_gft_newton(z)
r$converged
```

`inv_gft_path` *Sequential Inversion with Warm Starts and the Tangent Predictor*

Description

`inv_gft_path` inverts a sequence of GFT vectors, starting each solve from the previous solution or from its first-order (tangent) prediction; `gft_predict` computes that prediction.

Usage

```
inv_gft_path(zs, predictor = TRUE, rtol = 1e-3, tol = 1e-13, ...)
gft_predict(z_prev, x_prev, lam, Q, z_new, rtol = 1e-3)
```

Arguments

zs	a list of numeric vectors of length $n(n-1)/2$.
predictor	if TRUE, start each solve from the tangent prediction; otherwise from the previous solution.
rtol	relative residual to which the predictor's linear system is solved.
tol, ...	passed to inv_gft.
z_prev, x_prev	the previous input and its solution.
lam, Q	eigenvalues and eigenvectors of $A[x_{prev}; z_{prev}]$.
z_new	the new input.

Details

Differentiating $\ell(x^*(z); z) = 0$ gives the prediction $\hat{x} = x_{prev} + p$ with $Hp = -(B\Delta z + D\ell)$, where $B\Delta z$ is the derivative of $\mathrm{diag}(e^A)$ in the direction of the off-diagonal change (three matrix multiplications) and $D\ell$ corrects for the residual left by the previous solve. The system is solved by preconditioned conjugate gradients with the stored eigendecomposition, so the prediction costs no eigendecomposition. The prediction error is $O(h^2)$ in the step size. The predictor pays at moderate step sizes; for very small steps warm starts alone are nearly as good, and for large steps its linear solve can cost more than the eigendecomposition it saves.

Value

inv_gft_path: a list of "gft_inv" objects, one per input, whose hvs components include the predictor's Hessian-vector products, with attribute "nbdz" counting the predictor evaluations. gft_predict: a list with the predicted diagonal x and the number hvs of products used.

See Also

inv_gft.

Examples

```
set.seed(4)
b <- 0.85 + 0.149 * runif(30)
z0 <- gft(b %*% t(b) + diag(1 - b^2))
zs <- lapply(0:5, function(k) z0 + 0.02 * k * rnorm(length(z0)) / sqrt(length(z0)))
out <- inv_gft_path(zs)
sapply(out, `[[`, "eighs")
```

print.gft_inv *Print an Inverse GFT Result*

Description

Compact display of an object of class "gft_inv" as returned by inv_gft and the other inverse-GFT solvers.

Usage

```
## S3 method for class 'gft_inv'
print(x, ...)
```

Arguments

x	an object of class "gft_inv".
...	ignored.

Value

x, invisibly.

Examples

```
r <- inv_gft(gft(0.9^abs(outer(1:5, 1:5, "-"))))
print(r)
```

vecl *Stack and Unstack Below-Diagonal Elements*

Description

vecl stacks the below-diagonal elements of a square matrix column by column. unvecl is its right inverse on symmetric matrices with zero diagonal: it forms the symmetric matrix with zero diagonal and below-diagonal elements z.

Usage

```
vecl(M)
unvecl(z)
```

Arguments

M	a square matrix.
z	a numeric vector of length $n(n-1)/2$ for some integer n.

Value

`vecl` returns a numeric vector of length $n(n-1)/2$. `unvecl` returns an $n \times n$ symmetric numeric matrix with zero diagonal.

Examples

```
M <- matrix(1:9, 3, 3)
vecl(M)                 # c(2, 3, 6)
unvecl(c(0.1, 0.2, 0.3))
```

Index