Package: landgraph (via r-universe)

October 30, 2026

Type Package

Title Graphs and Covariance for Landscape Genetics

Version 0.0.4

Date 2026-10-29

Description Shared, dependency-light primitives for landscape-genetic network methods: a lightweight deme/landscape graph (vertex coordinates and an undirected edge list) with constructors from coordinates; genetic covariance and distance from biallelic or multiallelic data (the normalized-dosage covariance of Yang et al. (2010) <doi:10.1038/ng.608>, and the multivariate covariance of Dyer and Nason (2004) <doi:10.1111/j.1365-294X.2004.02177.x> built by the Gower (1966) <doi:10.1093/biomet/53.3-4.325> double-centering); pairwise F_ST by the estimator of Bhatia et al. (2013) <doi:10.1101/gr.154831.113>; and antisymmetric per-edge directional covariate builders (the gradient of a scalar potential, and the projection of a vector flow field). Used by 'terradish' (symmetric resistance).

License BSD_3_clause + file LICENSE

Encoding UTF-8

Depends R (>= 4.0.0)

Imports stats

Suggests terra, deldir, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

URL https://github.com/wpeterman/landgraph

BugReports https://github.com/wpeterman/landgraph/issues

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Bill Peterman [aut, cre, cph], Nathaniel S. Pope [aut, cph]

Maintainer Bill Peterman <peterman.73@osu.edu>

Repository https://cran.r-universe.dev

Date/Publication 2026-10-30 14:25:28 UTC

RemoteUrl https://github.com/cran/landgraph

RemoteRef HEAD

RemoteSha a276411b27ddb1df55be47cff6c1908e931604a7

Contents

cov_from_biallelic 2
cov_from_genetic_data 4
deme_graph 8
dist_from_biallelic 9
dist_from_cov 10
edge_flow 11
edge_gradient 12
fst_from_biallelic 13

Index **15**

`cov_from_biallelic` *Genetic covariance from biallelic allele counts*

Description

Computes a sample covariance matrix from counts of the derived allele across biallelic (SNP) markers. Rows may be individuals, populations, or any other sampled genetic unit.

Usage

```
cov_from_biallelic(
  Y,
  N = NULL,
  ploidy = 2,
  monomorphic = c("drop", "error"),
  tol = sqrt(.Machine$double.eps)
)
```

Arguments

`Y`	Numeric matrix of derived-allele counts with sampled units in rows and loci in columns. For diploid individuals genotyped as 0/1/2, this is the standard dosage matrix.
`N`	Optional haploid sample-size specification. Controls how many chromosomes were sampled at each locus for each unit:

- NULL: use ploidy for all cells.
- A scalar: recycled to all cells.
- A vector of length nrow(Y): row-specific sample sizes (same at every locus for that row).
- A vector of length ncol(Y): locus-specific sample sizes (same across all rows).
- A matrix matching dim(Y): fully general cell-level sizes.

ploidy — Haploid chromosome count per unit, used when N = NULL. The default 2 corresponds to diploid individuals with dosage coding 0/1/2 (so each individual contributes two chromosomes at each locus).

monomorphic — How to handle loci with pooled allele frequency 0 or 1 (fixed across all rows). "drop" (default) removes them and emits a warning; "error" stops with an error to preserve historical strict behavior.

tol — Tolerance for identifying monomorphic loci. A locus is treated as monomorphic when its pooled frequency is within tol of 0 or 1.

Details

Let $p_\ell = \sum_i Y_{i\ell} / \sum_i N_{i\ell}$ be the pooled derived-allele frequency at locus ℓ. The normalized allele dosage is:

$$Z_{i\ell} = \frac{Y_{i\ell} - N_{i\ell} p_\ell}{\sqrt{N_{i\ell} p_\ell (1 - p_\ell)}}$$

The function returns $ZZ^\top / L$, where L is the number of retained (polymorphic) loci. The resulting matrix is the sample covariance in normalized allele-frequency space, analogous to the genomic relationship matrix of Yang et al. (2010).

The function standardizes counts, not frequencies, so entries grow with the haploid sample size: $Z_{i\ell} = \sqrt{N_{i\ell}}(\hat{p}_{i\ell} - p_\ell)/\sqrt{p_\ell(1-p_\ell)}$. For individual 0/1/2 dosage this is the genomic relationship matrix scale. For population counts with equal sample sizes, cov_from_biallelic(Y, N) / N[1] gives the standardized allele-frequency covariance that terradish fits. Do not use that shortcut with unequal sizes; rows then carry unequal sampling variance that a common nugget does not absorb. Y and N cannot contain NA. For population counts, record missing genotypes by lowering the matching cells of N. Every cell of N must be positive, so for individual dosage you must filter loci or individuals first. Counting either allele gives the same covariance.

Per-locus standardization gives greater weight to rare variants, including their contribution to the diagonal. Filter loci by minor allele frequency before constructing the covariance, report the filtering rule, and check that conclusions hold across thresholds. Unequal chromosome counts also imply unequal sampling variance. With unequal counts across rows, pooled-frequency centering does not generally produce zero row sums in the covariance.

To convert the covariance to a pairwise distance matrix, use dist_from_cov. For non-biallelic or multiallelic data, use cov_from_genetic_data.

Value

A symmetric positive-semidefinite nrow(Y) by nrow(Y) matrix of pairwise genetic covariances on the count-standardized scale (see Details). Row and column names match the row names of

`Y` when present. The `diagonal` attribute is `"normalized_dosage"`. The `centered` attribute is `"sites"` when chromosome counts are equal across rows at every retained locus; otherwise it is `"pooled_allele_frequency"`. These labels describe the construction without changing the returned numerical values.

References

Yang J, Benyamin B, McEvoy BP, Gordon S, Henders AK, Nyholt DR, Madden PA, Heath AC, Martin NG, Montgomery GW, Goddard ME, Visscher PM. 2010. Common SNPs explain a large proportion of the heritability for human height. Nature Genetics 42(7):565-569. doi:10.1038/ng.608

See Also

`cov_from_genetic_data`, `fst_from_biallelic`, `dist_from_biallelic`, `dist_from_cov`, `terradish::wishart_covari`

Examples

```
# Individual diploid dosage matrix: 3 individuals, 3 SNPs
Y <- matrix(c(2, 1, 0,
              1, 1, 1,
              0, 1, 2), nrow = 3, byrow = TRUE)
cov_from_biallelic(Y)          # ploidy = 2 by default

# Population-level counts with unequal sampling
Y2 <- matrix(c(10, 4, 1,
               5,  5, 3), nrow = 2, byrow = TRUE)
N2 <- matrix(c(20, 20, 10,
               10, 10, 6), nrow = 2, byrow = TRUE)
cov_from_biallelic(Y2, N = N2)
```

`cov_from_genetic_data` *Covariance from multivariate genetic data*

Description

Calculates individual- or population-level genetic covariance from multivariate genetic data. This is useful for microsatellite, multiallelic, SNP dosage, PCA score, or other numeric encodings.

Usage

```
cov_from_genetic_data(
  x,
  groups = NULL,
  input = c("features", "allele_calls"),
  loci = NULL,
  center = TRUE,
  scale = TRUE,
  tol = sqrt(.Machine$double.eps),
```

```
  diagonal = c("gower", "within", "auto"),
  normalize = c("none", "features")
)
```

Arguments

x	Genetic data. For `input = "features"`, a numeric matrix or data frame with individuals in rows and genetic features in columns. For `input = "allele_calls"`, a matrix or data frame of allele calls with one column per allele copy.
groups	Optional factor, character, or integer vector assigning each row of `x` to a population. If `NULL`, each row is treated as an individual sampled unit and an individual-level covariance matrix is returned.
input	Input format. `"features"` treats `x` as an already numeric feature matrix. `"allele_calls"` converts allele calls to per-locus allele dosage columns before calculating covariance.
loci	Required for `input = "allele_calls"`; a vector with one entry per column of `x` identifying the locus for each allele-copy column. For a diploid microsatellite matrix with two columns per locus, the two columns for each locus should have the same `loci` value.
center	Should feature columns be centered by their global mean before covariance calculation? Default `TRUE`. The covariance depends only on differences among units, so centering changes only the `centroids`, `unit_features`, and `feature_center` attributes.
scale	Should feature columns be divided by their global standard deviation? Default `TRUE`. Scaling divides each feature by its standard deviation, so rare alleles get the same variance as common ones and therefore more relative weight. Constant features are dropped with a warning.
tol	Tolerance used to identify constant features when `scale = TRUE`.
diagonal	How to set the returned covariance diagonal. `"gower"` (default) keeps the double-centered diagonal, so every entry shares one scale; use it for Wishart models. `"within"` substitutes summed within-population variances, as in population graphs; terradish warns when it receives one. `"auto"` restores the default used before 0.0.3: `"within"` for grouped data with replication and `"gower"` otherwise.
normalize	Should covariance scale be normalized? `"none"` returns sums over retained features. `"features"` divides the covariance, squared distances, and within-population variances by the number of retained features. The divisor counts expanded allele features, not loci, and is not `nu`.

Details

The function first obtains a numeric feature matrix. Microsatellite data can be supplied as allele calls by setting `input = "allele_calls"`; these are converted to allele dosage columns, one column per locus-allele combination.

Wishart models in terradish, `terradish::wishart_covariance()` and `terradish::generalized_wishart()`, need `nu`, an effective-information value that you supply. This function does not compute it, and the

number of SNPs, loci, or allele features does not supply it: linked loci share population history, so they carry far less independent information than their count suggests. Hold `nu` fixed across compared models and report sensitivity with `terradish::terradish_rescale_nu()`.

If `groups = NULL`, rows of the processed feature matrix are used directly and squared Euclidean distances among individuals are transformed to a covariance matrix by Gower double-centering. This produces a positive semidefinite row-level covariance matrix up to numerical tolerance.

If `groups` is supplied, population centroids are calculated in feature space before the Gower transform. With `diagonal = "within"`, the diagonal is replaced by the sum of within-population feature variances. This matches the covariance construction used before population-graph partial-correlation filtering in Dyer-style population graph workflows.

A numeric feature matrix cannot contain `NA`. For allele calls, each missing copy becomes the most frequent allele at its locus; the `imputed_allele_calls` attribute records the count. Modal imputation shrinks distances, so report it. `terradish::pca_dist()` imputes allele means instead. Code missing calls as `NA`: the function reads codes such as 0 or -9 as real alleles.

The within-population diagonal is on a different scale from the covariance among population centroids and is unsuitable for a Wishart likelihood. Use `diagonal = "gower"` for that purpose. Replacing the diagonal also removes the guarantee of positive semidefiniteness and zero row sums. Unequal group sizes imply unequal sampling variance. Standardizing allele features gives greater weight to rare variants; filter by minor allele frequency before constructing the covariance and report the filtering rule.

Value

A symmetric numeric matrix with one row and column per individual (when `groups = NULL`) or population (when `groups` is supplied). The matrix is positive semi-definite up to numerical tolerance, except when `diagonal = "within"` is used (see Details). The following attributes are attached. For a result `S`, use `S[, ]` to view the matrix without attributes.

`centroids`, `unit_features` Feature matrix of population centroids (or individual feature vectors when ungrouped), one row per returned unit. The two attributes hold the same matrix.

`centroid_distance2`, `unit_distance2` Squared Euclidean distance matrix among the rows of `centroids`, divided by `normalizer`. The two attributes hold the same matrix.

`within_variance` Named vector of summed within-population feature variances, divided by `normalizer`. Attached whenever `groups` is supplied, with 0 for single-member groups. It fills the diagonal only when `diagonal = "within"`.

`unit_size` Named vector of group sizes: numeric when `groups` is supplied, integer (all 1) when ungrouped.

`feature_center`, `feature_scale` Centering and scaling constants applied to each retained feature.

`retained_features` Names of the features kept after constant features were dropped.

`input` The `input` format used.

`imputed_allele_calls`, `imputed_modal_alleles` Only for `input = "allele_calls"`: the number of imputed calls per locus and the modal allele used to fill them, for loci with missing calls (zero-length when no call was missing).

`level` Either `"individual"` or `"population"`.

`diagonal` The `diagonal` method actually used.

`centered` `"sites"` for the Gower covariance, or `"sites_before_diagonal_replacement"` when the within-population diagonal replaces the centered covariance diagonal.

`normalize` The `normalize` option used.

`normalizer` Divisor applied to the covariance and distances (`1` for `normalize = "none"`, number of retained features for `normalize = "features"`).

References

Gower JC. 1966. Some distance properties of latent root and vector methods used in multivariate analysis. Biometrika 53(3-4):325-338. doi:10.1093/biomet/53.34.325

Dyer RJ, Nason JD. 2004. Population graphs: the graph theoretic shape of genetic structure. Molecular Ecology 13(7):1713-1727. doi:10.1111/j.1365294X.2004.02177.x

Dyer RJ. 2015. Population graphs and landscape genetics. Annual Review of Ecology, Evolution, and Systematics 46:327-342. doi:10.1146/annurevecolsys112414054150

See Also

`terradish::wishart_covariance()`, `cov_from_biallelic`, `dist_from_cov`, `terradish::simulate_covariance_resp`

Examples

```
# Numeric allele dosage / feature matrix
x <- matrix(c(0, 1,
              1, 1,
              2, 0,
              2, 1,
              0, 2,
              1, 2),
            ncol = 2, byrow = TRUE)
cov_from_genetic_data(x)

groups <- rep(c("pop1", "pop2", "pop3"), each = 2)
cov_from_genetic_data(x, groups = groups)

# Microsatellite-style allele-call columns: two allele copies per locus
alleles <- data.frame(
  loc1_a = c(100, 100, 102, 102, 104, 104),
  loc1_b = c(100, 102, 102, 104, 104, 100),
  loc2_a = c(200, 202, 200, 202, 204, 204),
  loc2_b = c(202, 202, 204, 204, 204, 200)
)
cov_from_genetic_data(
  alleles,
  groups = groups,
  input = "allele_calls",
  loci = c("loc1", "loc1", "loc2", "loc2")
)
```

deme_graph *Lightweight deme / landscape graph from coordinates*

Description

Builds the minimal spatial graph that landscape-genetic network methods use: a set of vertex coordinates and an undirected edge list. For deme-scale problems (dozens to a few hundred nodes) this avoids the heavier raster / conductance graph builders. The class lets `edge_gradient` and `edge_flow` accept the object. It lacks the covariates, Laplacian, and Cholesky factor that `terradish::terradish()` needs; build fitting graphs with `terradish::conductance_surface()`.

Usage

```
deme_graph(
  coords,
  neighbours = c("delaunay", "knn", "lattice"),
  k = 6L,
  queen = FALSE
)
```

Arguments

`coords`	A two-column numeric matrix or data frame of node (deme) coordinates, one row per node (columns x, y).
`neighbours`	Edge construction. `"delaunay"` (the Delaunay triangulation; needs the **deldir** package), `"knn"` (symmetric k-nearest-neighbor adjacency), or `"lattice"` (rook or queen adjacency for points lying on a regular grid with uniform spacing).
`k`	Number of neighbors for `neighbours = "knn"`; a single positive whole number. Values larger than the number of other nodes are reduced to it.
`queen`	For `neighbours = "lattice"`, also connect diagonal (queen) neighbors; the default rook adjacency reproduces `terradish::conductance_surface(directions = 4)`.

Value

An object of class `c("landgraph", "terradish_graph")` with `vertex_coordinates` (an `n x 2` matrix), `edge_pairs` (an `m x 2` integer matrix of 1-based undirected edges, each pair once, `a < b`), `coords` (a copy of `vertex_coordinates`), and `n_vertices`.

See Also

`edge_gradient`, `edge_flow`

Examples

```
coords <- as.matrix(expand.grid(x = 0:4, y = 0:4))
g <- deme_graph(coords, neighbours = "lattice")
nrow(g$edge_pairs)
```

dist_from_biallelic *Genetic distance matrix from biallelic allele counts*

Description

Computes pairwise squared genetic distances from biallelic (SNP) allele counts. This is a convenience wrapper for dist_from_cov(cov_from_biallelic(Y, N)).

Usage

```
dist_from_biallelic(Y, N = NULL, ...)
```

Arguments

Y	Numeric matrix of derived-allele counts (populations/individuals in rows, loci in columns).
N	Optional haploid sample sizes in any form accepted by cov_from_biallelic. The default NULL uses ploidy (2 unless set through ...) for every cell.
...	Further arguments passed to cov_from_biallelic, such as ploidy, monomorphic, or tol.

Value

A symmetric matrix of pairwise squared distances with a zero diagonal, on the count scale of cov_from_biallelic() and named by rownames(Y). With equal sample sizes, the terradish response dist_from_cov(cov_from_biallelic(Y, N) / N[1]) equals this matrix divided by N[1]. Suitable for terradish::mlpe() or terradish::generalized_wishart().

See Also

cov_from_biallelic, dist_from_cov, fst_from_biallelic

Examples

```
Y <- matrix(c(2, 1, 0,
              1, 1, 1,
              0, 1, 2), nrow = 3, byrow = TRUE)
N <- matrix(2, nrow = 3, ncol = 3)
dist_from_biallelic(Y, N)
dist_from_biallelic(Y)   # N = NULL uses ploidy = 2 for every cell
```

dist_from_cov *Squared-distance matrix from a covariance matrix*

Description

Converts a covariance to the squared Euclidean distances it implies. For cov_from_genetic_data output with diagonal = "gower", the result equals its centroid_distance2 attribute; with "within", it does not.

Usage

```
dist_from_cov(Cov)
```

Arguments

Cov	A square numeric covariance matrix. Does not need to be full-rank (e.g. the output of cov_from_biallelic is positive semi-definite).

Details

The squared distance between units i and j is:

$$D_{ij} = C_{ii} + C_{jj} - 2C_{ij}$$

which is the law-of-cosines identity relating a covariance matrix to its implied squared Euclidean distances. The diagonal of the returned matrix is zero.

Squared distances from a coherent covariance suit terradish::generalized_wishart(), which gives the same likelihood as terradish::wishart_covariance() and needs the same nu. terradish::mlpe() accepts any pairwise distance and needs no nu.

The result keeps the diagonal and centered attributes of Cov, so terradish can still warn about a "within" diagonal. Other attributes are dropped.

Value

A symmetric matrix with the dimnames of Cov and a zero diagonal. Off-diagonal entries are non-negative when Cov is positive semidefinite.

See Also

cov_from_biallelic, cov_from_genetic_data, dist_from_biallelic

Examples

```
Cov <- matrix(c(2.0, 1.2, 0.8,
                1.2, 1.5, 0.7,
                0.8, 0.7, 1.1), nrow = 3, byrow = TRUE)
dist_from_cov(Cov)

# Round-trip: covariance -> distance -> (verify non-negative, zero diag)
Y <- matrix(c(2, 1, 0, 1, 1, 1, 0, 1, 2), nrow = 3, byrow = TRUE)
S_cov  <- cov_from_biallelic(Y)
S_dist <- dist_from_cov(S_cov)
diag(S_dist)   # all zeros
```

---

edge_flow *Circulation (flow-field) edge covariate from a spatial vector field*

Description

Builds an antisymmetric per-edge covariate from a spatial vector field (for example wind or current), for use in directed gene-flow models. edge_gradient takes the gradient of a scalar potential, which is curl-free and yields a reversible generator whose stationary distribution is collinear with the potential. A vector flow field can carry a rotational/curl component instead, making the directed generator non-reversible and its stationary distribution non-collinear with any scalar covariate.

Usage

```
edge_flow(field, data)
```

Arguments

field	The vector field at the active graph cells: a two-column numeric matrix or data frame (x- and y-components, in vertex order), a two-layer terra::SpatRaster (sampled at the vertex coordinates), or a function of a two-column coordinate matrix returning such a two-column field.
data	A terradish_graph from deme_graph (must carry vertex_coordinates).

Details

For undirected edge (a, b) the covariate is the mean field along the edge projected onto the edge direction, $c_{ab} = \frac{1}{2}(f_a + f_b) \cdot (xy_b - xy_a)$; it is antisymmetric by construction, and a directed model applies $-c_{ab}$ to the reverse edge $b \to a$.

Value

A numeric vector with one entry per undirected edge in data$edge_pairs, ready to pass to a directed gene-flow model.

See Also

edge_gradient

Examples

```
coords <- as.matrix(expand.grid(x = 0:2, y = 0:2))
g <- deme_graph(coords, neighbours = "lattice")
field <- matrix(c(1, 0), nrow = nrow(coords), ncol = 2, byrow = TRUE)
edge_flow(field, g)
```

---

edge_gradient *Directional edge covariates from a spatial layer*

Description

Builds the antisymmetric covariate $d_{ab} = x_a - x_b$ for directional gene-flow models; it is positive from high to low x.

Usage

```
edge_gradient(x, data)
```

Arguments

x	A terra::SpatRaster layer (or a numeric vector with one value per active graph cell) giving the potential whose gradient drives direction.
data	A terradish_graph from deme_graph.

Value

A list with edges (an integer matrix of directed edges, columns a, b, 1-based node indices) and d (the matching numeric vector $d_{ab} = x_a - x_b$). With m rows in data$edge_pairs, rows 1 to m of edges follow edge_pairs and rows $m + 1$ to $2m$ reverse them, so $d_{ab} = -d_{ba}$.

See Also

edge_flow, deme_graph

Examples

```
coords <- as.matrix(expand.grid(x = 0:2, y = 0:2))
g <- deme_graph(coords, neighbours = "lattice")
elevation <- c(10, 20, 30, 25, 15, 5, 8, 18, 28)
edge_gradient(elevation, g)
```

`fst_from_biallelic` *F_ST from biallelic allele counts*

Description

Estimates pairwise F_{ST} from counts of the derived allele across biallelic markers using the ratio-of-averages estimator of Bhatia et al. (2013).

Usage

```
fst_from_biallelic(Y, N)
```

Arguments

`Y`	Numeric matrix of derived-allele counts with rows as populations and columns as loci.
`N`	Haploid sample sizes in any form accepted by `cov_from_biallelic`; every entry must exceed 1. For diploid data without missing genotypes, `N = 2 * n_individuals` for each cell.

Details

The estimator computes, for each locus, the squared allele-frequency difference between populations, corrects for within-population heterozygosity, and then forms a ratio of locus-averages (Bhatia et al. 2013). Diagonal entries are set to 0.

The result is not guaranteed to lie in $[0, 1]$ for every pair; small negative values can arise from sampling noise in very similar populations.

To use F_{ST} as the response in `terradish`, pass the matrix directly to the formula left-hand side with `terradish::mlpe()`. This ratio estimator is not a Wishart response and has no Wishart degrees of freedom. Use covariance-derived squared distances for Wishart models.

Value

A symmetric matrix named by `rownames(Y)` holding pairwise Hudson F_{ST}, with a zero diagonal.

References

Bhatia G, Patterson N, Sankararaman S, Price AL. 2013. Estimating and interpreting F_{ST}: the impact of rare variants. Genome Research 23(9):1514-1521. doi:10.1101/gr.154831.113

See Also

`cov_from_biallelic`, `dist_from_biallelic`

Examples

```
Y <- matrix(c(2, 1, 0,
              1, 1, 1,
              0, 1, 2), nrow = 3, byrow = TRUE)
N <- matrix(2, nrow = 3, ncol = 3)
fst_from_biallelic(Y, N)

# A scalar or a locus-length vector also works
fst_from_biallelic(Y, 2)
```

Index

cov_from_biallelic, 2, *7*, *9, 10*, *13*
cov_from_genetic_data, *3*, 4, *4*, *10*

deme_graph, 8, *11, 12*
dist_from_biallelic, *4*, 9, *10*, *13*
dist_from_cov, *3, 4*, *7*, *9*, 10

edge_flow, *8*, 11, *12*
edge_gradient, *8*, *11*, 12, *12*

fst_from_biallelic, *4*, *9*, 13