Package ‘visPedigree’

February 23, 2026

Type Package

Title Tidying and Visualizing Animal Pedigrees

Version 1.0.1

Description Built on graph theory and the high-performance 'data.table' framework, this package provides a comprehensive suite of tools for tidying, analyzing, and visualizing animal pedigrees. By modeling pedigrees as directed acyclic graphs using 'igraph', it ensures robust loop detection, efficient generation assignment, and optimal sub-population splitting. Key features include standardizing pedigree formats, flexible ancestry tracing, and generating legible vector-based PDF graphs. A unique compaction algorithm enables the visualization of massive pedigrees by grouping full-sib families. Furthermore, the package implements high-performance C++ algorithms for calculating and visualizing genetic relationship matrices (A, D, AA, and their inverses) and inbreeding coefficients.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 4.1.0)

Imports data.table (>= 1.14.0), igraph (>= 1.3.0), Matrix, Rcpp, lattice

LinkingTo Rcpp, RcppArmadillo

Suggests nadiv (>= 2.18.0), testthat (>= 3.0.0), knitr, rmarkdown

URL `https://github.com/luansheng/visPedigree`, `https://luansheng.github.io/visPedigree/`

BugReports `https://github.com/luansheng/visPedigree/issues`

VignetteBuilder knitr

Config/testthat/edition 3

RoxygenNote 7.3.3

NeedsCompilation yes

Author Sheng Luan [aut, cre]

Maintainer Sheng Luan <luansheng@gmail.com>

Repository CRAN

Date/Publication 2026-02-23 08:30:02 UTC

Contents

big_family_size_ped 2
deep_ped 3
expand_pedmat 3
inbreed 4
loop_ped 5
pedmat 5
plot.tidyped 9
print.summary.tidyped 10
print.tidyped 10
query_relationship 11
simple_ped 12
small_ped 12
splitped 13
summary.tidyped 14
summary_pedmat 15
tidyped 16
vismat 18
visped 22

Index **27**

`big_family_size_ped` *A large pedigree with big family sizes*

Description

A dataset containing a pedigree with many full-sib individuals per family.

Usage

```
big_family_size_ped
```

Format

A data.table with 8 columns:

Ind Individual ID

Sire Sire ID

Dam Dam ID

Sex Sex of the individual

Year Year of birth

IndNum Numeric ID for individual

SireNum Numeric ID for sire

DamNum Numeric ID for dam

`deep_ped` *A deep pedigree*

Description

A dataset containing a pedigree with many generations.

Usage

```
deep_ped
```

Format

A data.table with 4 columns:

Ind Individual ID

Sire Sire ID

Dam Dam ID

Sex Sex of the individual

`expand_pedmat` *Expand a Compact Pedigree Matrix to Full Dimensions*

Description

Restores a compact pedmat to its original dimensions by mapping each individual to their family representative's values. For non-compact matrices, returns the matrix unchanged.

Usage

```
expand_pedmat(x)
```

Arguments

`x`	A pedmat object from `pedmat`.

Details

For compact matrices, full-siblings within the same family will have identical relationship values in the expanded matrix because they shared the same representative during calculation.

Value

Matrix or vector with original pedigree dimensions:

- Matrices: Row and column names set to all individual IDs
- Vectors (e.g., method="f"): Names set to all individual IDs

The result is **not** a pedmat object (S3 class stripped).

See Also

pedmat, query_relationship

Examples

```
tped <- tidyped(small_ped)

# Compact matrix
A_compact <- pedmat(tped, method = "A", compact = TRUE)
dim(A_compact)  # Reduced dimensions

# Expand to full size
A_full <- expand_pedmat(A_compact)
dim(A_full)  # Original dimensions restored

# Non-compact matrices are returned unchanged
A <- pedmat(tped, method = "A", compact = FALSE)
A2 <- expand_pedmat(A)
identical(dim(A), dim(A2))  # TRUE
```

---

inbreed — *Calculate inbreeding coefficients*

Description

`inbreed` function calculates the inbreeding coefficients for all individuals in a tidied pedigree.

Usage

```
inbreed(ped, ...)
```

Arguments

`ped`	A `tidyped` object.
`...`	Additional arguments (currently ignored).

Details

This function takes a pedigree tidied by the `tidyped` function and calculates the inbreeding coefficients using optimized C++ code based on the Meuwissen & Luo (1992) algorithm. It is the core engine used by both `tidyped(..., inbreed = TRUE)` and `pedmat(..., method = "f")`, ensuring consistent results across the package. It is significantly faster than standard R implementations for large pedigrees.

Value

A `tidyped` object with an additional column **f**.

`loop_ped` *A pedigree with loops*

Description

A dataset containing a pedigree with circular mating loops.

Usage

```
loop_ped
```

Format

A data.table with 3 columns:

Ind Individual ID

Sire Sire ID

Dam Dam ID

`pedmat` *Genetic Relationship Matrices and Inbreeding Coefficients*

Description

Optimized calculation of additive (A), dominance (D), epistatic (AA) relationship matrices, their inverses, and inbreeding coefficients (f). Uses Rcpp with Meuwissen & Luo (1992) algorithm for efficient computation.

Usage

```
pedmat(
  ped,
  method = "A",
  sparse = TRUE,
  invert_method = "auto",
  threads = 0,
  compact = FALSE
)
```

Arguments

`ped`	A tidied pedigree from `tidyped`. Must be a single pedigree, not a splitped object. For splitped results, use `pedmat(ped_split$GP1, ...)` to process individual groups.
`method`	Character, one of: • `"A"`: Additive (numerator) relationship matrix (default) • `"f"`: Inbreeding coefficients (returns named vector) • `"Ainv"`: Inverse of A using Henderson's rules (O(n) complexity) • `"D"`: Dominance relationship matrix • `"Dinv"`: Inverse of D (requires matrix inversion) • `"AA"`: Additive-by-additive epistatic matrix (A # A) • `"AAinv"`: Inverse of AA
`sparse`	Logical, if `TRUE` returns sparse Matrix (recommended for large pedigrees). Default is `TRUE`.
`invert_method`	Character, method for matrix inversion (Dinv/AAinv only): • `"auto"`: Auto-detect and use optimal method (default) • `"sympd"`: Force Cholesky decomposition (faster for SPD matrices) • `"general"`: Force general LU decomposition
`threads`	Integer. Number of OpenMP threads to use. Use 0 to keep the system/default setting. Currently, multi-threading is explicitly implemented for: • `"D"`: Dominance relationship matrix (significant speedup). • `"Ainv"`: Inverse of A (only for large pedigrees, n >= 5000). For `"Dinv"`, `"AA"`, and `"AAinv"`, parallelism depends on the linked BLAS/LAPACK library (e.g., OpenBLAS, MKL, Accelerate) and is not controlled by this parameter. Methods `"A"` and `"f"` are single-threaded.
`compact`	Logical, if `TRUE` compacts full-sibling families by selecting one representative per family. This dramatically reduces matrix dimensions for pedigrees with large full-sib groups. See Details.

Details

API Design:

Only a single method may be requested per call. This design prevents accidental heavy computations. If multiple matrices are needed, call `pedmat()` separately for each method.

Compact Mode (`compact = TRUE`):

Full-siblings share identical relationships with all other individuals. Compact mode exploits this by selecting one representative per full-sib family, dramatically reducing matrix size. For example, a pedigree with 170,000 individuals might compact to 1,800 unique relationship patterns.

Key features:

- `query_relationship(x, id1, id2)`: Query any individual pair, including merged siblings (automatic lookup)
- `expand_pedmat(x)`: Restore full matrix dimensions
- `vismat(x)`: Visualize directly (auto-detects compact)

Performance Notes:

- **Ainv**: O(n) complexity using Henderson's rules. Fast for any size.
- **Dinv/AAinv**: O(n^3) matrix inversion. Practical limits:
  - n < 500: ~10-20 ms
  - n = 1,000: ~40-60 ms
  - n = 2,000: ~130-150 ms
  - n > 2,000: Consider using `compact = TRUE`
- **Memory**: Sparse matrices use ~O(nnz) memory; dense use O(n^2)

Value

Returns a matrix or vector with S3 class `"pedmat"`.

Object type by method:

- `method="f"`: Named numeric vector of inbreeding coefficients
- All other methods: Sparse or dense matrix (depending on `sparse`)

S3 Methods:

- `print(x)`: Display matrix with metadata header
- `summary_pedmat(x)`: Detailed statistics (size, compression, mean, density)
- `dim(x)`, `length(x)`, `Matrix::diag(x)`, `t(x)`: Standard operations
- `x[i, j]`: Subsetting (behaves like underlying matrix)
- `as.matrix(x)`: Convert to base matrix

Accessing Metadata (use `attr()`, not `$`):

- `attr(x, "ped")`: The pedigree used (or compact pedigree if `compact=TRUE`)
- `attr(x, "ped_compact")`: Compact pedigree (when `compact=TRUE`)
- `attr(x, "method")`: Calculation method used
- `attr(x, "call_info")`: Full calculation metadata (timing, sizes, etc.)

- `names(attributes(x))`: List all available attributes

Additional attributes when `compact = TRUE`**:**

- `attr(x, "compact_map")`: data.table mapping individuals to representatives
- `attr(x, "family_summary")`: data.table summarizing merged families
- `attr(x, "compact_stats")`: Compression statistics (ratio, n_families, etc.)

References

Meuwissen, T. H. E., & Luo, Z. (1992). Computing inbreeding coefficients in large populations. Genetics Selection Evolution, 24(4), 305-313.

Henderson, C. R. (1976). A simple method for computing the inverse of a numerator relationship matrix used in prediction of breeding values. Biometrics, 32(1), 69-83.

See Also

`tidyped` for preparing pedigree data, `query_relationship` for querying individual pairs, `expand_pedmat` for restoring full dimensions, `vismat` for visualization, `inbreed` for simple inbreeding calculation

Examples

```
# Basic usage with small pedigree
library(visPedigree)
tped <- tidyped(small_ped)

# --- Additive Relationship Matrix (default) ---
A <- pedmat(tped)
A["A", "B"]      # Relationship between A and B
Matrix::diag(A)  # Diagonal = 1 + F (inbreeding)

# --- Inbreeding Coefficients ---
f <- pedmat(tped, method = "f")
f["Z1"]  # Inbreeding of individual Z1

# --- Using summary_pedmat() ---
summary_pedmat(A)   # Detailed matrix statistics

# --- Accessing Metadata ---
attr(A, "ped")              # Original pedigree
attr(A, "method")           # "A"
names(attributes(A))        # All available attributes

# --- Compact Mode (for large full-sib families) ---
A_compact <- pedmat(tped, method = "A", compact = TRUE)

# Query relationships (works for any individual, including merged sibs)
query_relationship(A_compact, "Z1", "Z2")  # Full-sibs Z1 and Z2

# View compression statistics
attr(A_compact, "compact_stats")
attr(A_compact, "family_summary")

# Expand back to full size
A_full <- expand_pedmat(A_compact)
dim(A_full)  # Original dimensions restored

# --- Inverse Matrices ---
Ainv <- pedmat(tped, method = "Ainv")  # Henderson's rules (fast)

# --- Dominance and Epistatic ---
D <- pedmat(tped, method = "D")
AA <- pedmat(tped, method = "AA")

# --- Visualization (requires display device) ---
## Not run:
vismat(A)                        # Heatmap of relationship matrix
vismat(A_compact)                # Works with compact matrices
vismat(A, grouping = "Gen")      # Group by generation

## End(Not run)
```

`plot.tidyped` *Plot a tidy pedigree*

Description

Plot a tidy pedigree

Usage

```
## S3 method for class 'tidyped'
plot(x, ...)
```

Arguments

`x`	A `tidyped` object.
`...`	Additional arguments passed to `visped`.

Value

Invisibly returns a list of graph data from `visped` (node/edge data and layout components) used to render the pedigree; the primary result is the plot drawn on the current device.

`print.summary.tidyped` *Print method for summary.tidyped*

Description

Print method for summary.tidyped

Usage

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

Arguments

`x`	A summary.tidyped object.
`...`	Additional arguments (ignored).

Value

The input object, invisibly.

`print.tidyped` *Print method for tidyped pedigree*

Description

Print method for tidyped pedigree

Usage

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

Arguments

`x`	A tidyped object
`...`	Additional arguments passed to the data.table print method

Value

The input object, invisibly.

`query_relationship` *Query Relationship Coefficients from a Pedigree Matrix*

Description

Retrieves relationship coefficients between individuals from a pedmat object. For compact matrices, automatically handles lookup of merged full-siblings.

Usage

```
query_relationship(x, id1, id2 = NULL)
```

Arguments

`x`	A pedmat object created by `pedmat`.
`id1`	Character, first individual ID.
`id2`	Character, second individual ID. If `NULL`, returns the entire row of relationships for `id1`.

Details

For compact matrices (`compact = TRUE`), this function automatically maps individuals to their family representatives. For methods A, D, and AA, it can compute the correct relationship even between merged full-siblings using the formula:

- **A**: $a_{ij} = 0.5 * (a_{is} + a_{id})$ where s, d are parents
- **D**: $d_{ij} = a_{ij}^2$ (for full-sibs in same family)
- **AA**: $aa_{ij} = a_{ij}^2$

Value

- If `id2` is provided: numeric value (relationship coefficient)
- If `id2` is `NULL`: named numeric vector (id1's row)
- Returns `NA` if individual not found

Note

Inverse matrices (Ainv, Dinv, AAinv) are **not supported** because inverse matrix elements do not represent meaningful relationship coefficients.

See Also

`pedmat`, `expand_pedmat`

Examples

```
tped <- tidyped(small_ped)
A <- pedmat(tped, method = "A", compact = TRUE)

# Query specific pair
query_relationship(A, "A", "B")

# Query merged full-siblings (works with compact)
query_relationship(A, "Z1", "Z2")

# Get all relationships for one individual
query_relationship(A, "A")
```

`simple_ped` *A simple pedigree*

Description

A small dataset containing a simple pedigree for demonstration.

Usage

```
simple_ped
```

Format

A data.table with 4 columns:

Ind Individual ID

Sire Sire ID

Dam Dam ID

Sex Sex of the individual

`small_ped` *A small pedigree*

Description

A small dataset containing a pedigree with some missing parents.

Usage

```
small_ped
```

Format

A data.frame with 3 columns:

Ind Individual ID

Sire Sire ID

Dam Dam ID

`splitped` *Split Pedigree into Disconnected Groups*

Description

Detects and splits a tidyped object into disconnected groups (connected components). Uses igraph to efficiently find groups of individuals that have no genetic relationships with each other. Isolated individuals (Gen = 0, those with no parents and no offspring) are excluded from group splitting and stored separately.

Usage

```
splitped(ped)
```

Arguments

`ped`	A tidyped object created by `tidyped`.

Details

This function identifies connected components in the pedigree graph where edges represent parent-offspring relationships. Two individuals are in the same group if they share any ancestry (direct or indirect).

Isolated individuals (Gen = 0 in tidyped output) are those who:

- Have no known parents (Sire and Dam are both NA)
- Are not parents of any other individual in the pedigree

These isolated individuals are excluded from splitting and stored in the `isolated` attribute. Each resulting group contains at least 2 individuals (at least one parent-offspring relationship).

The function always returns a list, even if there is only one group (i.e., the pedigree is fully connected). Groups are sorted by size in descending order.

Each group in the result is a valid tidyped object with:

- Renumbered IndNum (1 to n for each group)
- Updated SireNum and DamNum referencing the new IndNum
- Recalculated Gen (generation) based on the group's structure

Value

A list of class "splitped" containing:

`GP1, GP2, ...`	tidyped objects for each disconnected group (with at least 2 individuals), with renumbered IndNum, SireNum, DamNum

The returned object has the following attributes:

`n_groups`	Number of disconnected groups found (excluding isolated individuals)
`sizes`	Named vector of group sizes
`total`	Total number of individuals in groups (excluding isolated)
`isolated`	Character vector of isolated individual IDs (Gen = 0)
`n_isolated`	Number of isolated individuals

See Also

`tidyped` for pedigree tidying

Examples

```
# Load example data
library(visPedigree)
data(small_ped)

# First tidy the pedigree
tped <- tidyped(small_ped)

# Split into groups
result <- splitped(tped)
print(result)

# Access individual groups (each is a tidyped object)
result$GP1

# Check isolated individuals
attr(result, "isolated")
```

`summary.tidyped` *Summary method for tidyped objects*

Description

Summary method for tidyped objects

Usage

```
## S3 method for class 'tidyped'
summary(object, ...)
```

Arguments

`object`	A tidyped object.
`...`	Additional arguments (ignored).

Value

A summary.tidyped object (list) containing:

- `n_ind`: Total number of individuals.
- `n_male`, `n_female`, `n_unknown_sex`: Sex composition counts.
- `n_founders`: Number of individuals with no known parents.
- `n_both_parents`: Count of individuals with complete parentage.
- `max_gen`, `gen_dist`: (Optional) Maximum generation and its distribution.
- `n_families`, `family_sizes`, `top_families`: (Optional) Family statistics.
- `f_stats`, `n_inbred`: (Optional) Inbreeding coefficient statistics.
- `n_cand`, `cand_f_stats`: (Optional) Candidate-specific statistics.

`summary_pedmat` *Summary Statistics for Pedigree Matrices*

Description

Computes and displays summary statistics for a pedmat object.

Usage

```
summary_pedmat(x)
```

Arguments

`x`	A pedmat object from `pedmat`.

Details

Since pedmat objects are often S4 sparse matrices with custom attributes, use this function instead of the generic `summary()` to ensure proper display of pedigree matrix statistics.

Value

An object of class `"summary.pedmat"` with statistics including method, dimensions, compression ratio (if compact), mean relationship, and matrix density.

See Also

`pedmat`

Examples

```
tped <- tidyped(small_ped)
A <- pedmat(tped, method = "A")
summary_pedmat(A)
```

---

tidyped *Tidy and prepare a pedigree using graph theory*

Description

This function takes a pedigree, checks for duplicate and bisexual individuals, detects pedigree loops using graph theory, adds missing founders, assigns generation numbers, sorts the pedigree, and traces the pedigree of specified candidates. If the `cand` parameter contains individual IDs, only those individuals and their ancestors or descendants are retained. Tracing direction and the number of generations can be specified using the `trace` and `tracegen` parameters.

Usage

```
tidyped(
  ped,
  cand = NULL,
  trace = "up",
  tracegen = NULL,
  addgen = TRUE,
  addnum = TRUE,
  inbreed = FALSE,
  genmethod = "top",
  ...
)
```

Arguments

`ped`	A data.table or data frame containing the pedigree. The first three columns must be **individual**, **sire**, and **dam** IDs. Additional columns, such as sex or generation, can be included. Column names can be customized, but their order must remain unchanged. Individual IDs should not be coded as "", " ", "0", "*", or "NA"; otherwise, they will be removed. Missing parents should be denoted by "NA", "0", or "*". Spaces and empty strings ("") are also treated as missing parents but are not recommended.
`cand`	A character vector of individual IDs, or NULL. If provided, only the candidates and their ancestors/descendants are retained.
`trace`	A character value specifying the tracing direction: "**up**", "**down**", or "**all**". "up" traces ancestors; "down" traces descendants; "all" traces the union of ancestors and descendants. This parameter is only used if `cand` is not NULL. Default is "up".

`tracegen`	An integer specifying the number of generations to trace. This parameter is only used if `trace` is not NULL. If NULL or 0, all available generations are traced.
`addgen`	A logical value indicating whether to generate generation numbers. Default is TRUE, which adds a **Gen** column to the output.
`addnum`	A logical value indicating whether to generate a numeric pedigree. Default is TRUE, which adds **IndNum**, **SireNum**, and **DamNum** columns to the output.
`inbreed`	A logical value indicating whether to calculate inbreeding coefficients. Default is FALSE. If TRUE, an **f** column is added to the output. This uses the same optimized engine as `pedmat(..., method = "f")`.
`genmethod`	A character value specifying the generation assignment method: "**top**" or "**bottom**". "top" (top-aligned) assigns generations from parents to offspring, starting founders at Gen 1. "bottom" (bottom-aligned) assigns generations from offspring to parents, aligning terminal nodes at the bottom. Default is "top".
`...`	Additional arguments passed to `inbreed`.

Details

Compared to the legacy version, this function handles cyclic pedigrees more robustly by detecting and reporting loops, and it is generally faster for large pedigrees due to the use of sparse graph algorithms from the `igraph` package. Generation assignment can be performed using either a top-down approach (default, aligning founders at the top) or a bottom-up approach (aligning terminal nodes at the bottom).

Value

A `tidyped` object (which inherits from `data.table`). Individual, sire, and dam ID columns are renamed to **Ind**, **Sire**, and **Dam**. Missing parents are replaced with **NA**. The **Sex** column contains "male", "female", or NA. The **Cand** column is included if `cand` is not NULL. The **Gen** column is included if `addgen` is TRUE. The **IndNum**, **SireNum**, and **DamNum** columns are included if `addnum` is TRUE. The **Family** and **FamilySize** columns identify full-sibling families (e.g., "A x B" for offspring of sire A and dam B). The **f** column is included if `inbreed` is TRUE.

See Also

`summary.tidyped` for summarizing tidyped objects `visped` for visualizing pedigree structure `pedmat` for computing relationship matrices `vismat` for visualizing relationship matrices `splitped` for splitting pedigree into connected components `inbreed` for calculating inbreeding coefficients

Examples

```
library(visPedigree)
library(data.table)

# Tidy a simple pedigree
tidy_ped <- tidyped(simple_ped)
head(tidy_ped)

# Trace ancestors of a specific individual within 2 generations
tidy_ped_tracegen <- tidyped(simple_ped, cand = "J5X804", trace = "up", tracegen = 2)
```

```
head(tidy_ped_tracegen)

# Trace both ancestors and descendants for multiple candidates
# This is highly optimized and works quickly even on 100k+ individuals
cand_list <- c("J5X804", "J3Y620")
tidy_ped_all <- tidyped(simple_ped, cand = cand_list, trace = "all")

# Check for loops (will error if loops exist)
try(tidyped(loop_ped))

# Example with a large pedigree: extract 2 generations of ancestors for 2007 candidates
cand_2007 <- big_family_size_ped[Year == 2007, Ind]

tidy_big <- tidyped(big_family_size_ped, cand = cand_2007, trace = "up", tracegen = 2)
summary(tidy_big)
```

`vismat` *Visualize Relationship Matrices*

Description

`vismat` provides visualization tools for relationship matrices (A, D, AA), supporting individual-level heatmaps and relationship coefficient histograms. This function is useful for exploring population genetic structure, identifying inbred individuals, and analyzing kinship between families.

Usage

```
vismat(
  mat,
  ped = NULL,
  type = "heatmap",
  ids = NULL,
  reorder = TRUE,
  grouping = NULL,
  labelcex = NULL,
  ...
)
```

Arguments

`mat` A relationship matrix. Can be one of the following types:

- A `pedmat` object returned by `pedmat`
- A named list containing matrices (preferring A, D, AA)
- A `tidyped` object (automatically calculates additive relationship matrix A)
- A standard `matrix` or `Matrix` object

	Note: Inverse matrices (Ainv, Dinv, AAinv) are not supported for visualization because their elements do not represent meaningful relationship coefficients.
`ped`	Optional. A tidied pedigree object (`tidyped`), used for extracting labels or grouping information. Required when using the `grouping` parameter. If `mat` is a `pedmat` object, the pedigree can be automatically extracted from its attributes.
`type`	Character, type of visualization. Supported options: • `"heatmap"`: Relationship matrix heatmap (default). Uses a Nature Genetics style color palette (white-orange-red-dark red), with optional hierarchical clustering and group aggregation. • `"histogram"`: Distribution histogram of relationship coefficients. Shows the frequency distribution of lower triangular elements (pairwise kinship).
`ids`	Character vector specifying individual IDs to display. Used to filter and display a submatrix of specific individuals. If `NULL` (default), all individuals are shown.
`reorder`	Logical. If `TRUE` (default), rows and columns are reordered using hierarchical clustering (Ward.D2 method) to bring closely related individuals together. Only affects heatmap visualization. Automatically skipped for large matrices (N > 2000) to improve performance. **Clustering principle**: Based on relationship profile distance (Euclidean). Full-sibs have nearly identical relationship profiles with the population, so they cluster tightly together.
`grouping`	Optional. Column name in `ped` to group by (e.g., `"Family"`, `"Gen"`, `"Year"`). When grouping is enabled: • Individual-level matrix is aggregated to group-level matrix (computing mean relationship coefficients between groups) • For `"Family"` grouping, founders without family assignment are excluded • For other grouping columns, NA values are assigned to `"Unknown"` group This is useful for analyzing the structure of large populations.
`labelcex`	Numeric. Manual control for font size of individual labels. If `NULL` (default), uses dynamic font size that adjusts automatically based on matrix dimensions (range 0.2-0.7). For matrices with more than 500 individuals, labels are automatically hidden.
`...`	Additional arguments passed to the plotting function: • Heatmap uses `levelplot`: can set `main`, `xlab`, `ylab`, `col.regions`, `colorkey`, `scales`, etc. • Histogram uses `histogram`: can set `main`, `xlab`, `ylab`, `nint` (number of bins), etc.

Details

Visualization Types:

Heatmap:

- Uses Nature Genetics style color palette (white to orange to red to dark red)
- Hierarchical clustering reordering is enabled by default to group similar individuals
- Matrix[1,1] is displayed at top-left corner

- Grid lines shown when N <= 100
- Individual labels shown when N <= 500

Histogram:

- Shows distribution of lower triangular elements (excluding diagonal)
- X-axis: relationship coefficient values; Y-axis: frequency percentage
- Useful for checking population inbreeding levels and kinship structure

Performance Considerations:

- N > 2000: Hierarchical clustering reordering is automatically skipped
- N > 500: Individual labels are automatically hidden
- N > 100: Grid lines are automatically hidden
- Grouping functionality uses optimized matrix algebra, suitable for large matrices

Interpreting Relationship Coefficients: For additive relationship matrix A:

- Diagonal elements = 1 + F (where F is the inbreeding coefficient)
- Off-diagonal elements = 2 x kinship coefficient
- Value 0: No relationship (unrelated)
- Value 0.25: Half-sibs or grandparent-grandchild
- Value 0.5: Full-sibs or parent-offspring
- Value 1.0: Same individual

Value

Invisibly returns the `lattice` plot object. The plot is generated on the current graphics device.

See Also

`pedmat` for computing relationship matrices `tidyped` for tidying pedigree data `visped` for visualizing pedigree structure graphs `levelplot` underlying plotting function for heatmaps `histogram` underlying plotting function for histograms

Examples

```
# ============================================================
# Basic Usage
# ============================================================

# Load example data
data(simple_ped)
ped <- tidyped(simple_ped)

# Method 1: Plot directly from tidyped object (auto-computes A matrix)
vismat(ped)

# Method 2: Plot from pedmat object
A <- pedmat(ped)
vismat(A)
```

```
# Method 3: Plot from plain matrix
A_dense <- as.matrix(A)
vismat(A_dense)

# ============================================================
# Heatmap Customization
# ============================================================

# Custom title and axis labels
vismat(A, main = "Additive Relationship Matrix", xlab = "Individual", ylab = "Individual")

# Disable clustering reorder (preserve original order)
vismat(A, reorder = FALSE)

# Custom label font size
vismat(A, labelcex = 0.5)

# Custom color palette (blue-white-red)
vismat(A, col.regions = colorRampPalette(c("blue", "white", "red"))(100))

# ============================================================
# Select Specific Individuals
# ============================================================

# Display only a subset of individuals
target_ids <- rownames(A)[1:8]
vismat(A, ids = target_ids)

# ============================================================
# Histogram Visualization
# ============================================================

# Relationship coefficient distribution histogram
vismat(A, type = "histogram")

# Custom number of bins
vismat(A, type = "histogram", nint = 30)

# ============================================================
# Group Aggregation (for large populations)
# ============================================================

# Group by generation
vismat(A, ped = ped, grouping = "Gen",
       main = "Mean Relationship Between Generations")

# Group by family (if pedigree has Family column)
# vismat(A, ped = ped, grouping = "Family")

# ============================================================
# Different Types of Relationship Matrices
# ============================================================
# Dominance relationship matrix
D <- pedmat(ped, method = "D")
vismat(D, main = "Dominance Relationship Matrix")

# Inbreeding coefficient distribution (diagonal elements - 1)
A_mat <- as.matrix(A)
f_values <- Matrix::diag(A_mat) - 1
hist(f_values, main = "Inbreeding Coefficient Distribution", xlab = "Inbreeding (F)")
```

visped *Visualize a tidy pedigree*

Description

visped function draws a graph of a full or compact pedigree.

Usage

```
visped(
  ped,
  compact = FALSE,
  outline = FALSE,
  cex = NULL,
  showgraph = TRUE,
  file = NULL,
  highlight = NULL,
  trace = FALSE,
  showf = FALSE,
  pagewidth = 200,
  symbolsize = 1,
  maxiter = 1000,
  ...
)
```

Arguments

ped	A tidyped object (which inherits from data.table). It is recommended that the pedigree is tidied and pruned by candidates using the tidyped function with the non-null parameter cand.
compact	A logical value indicating whether IDs of full-sib individuals in one generation will be removed and replaced with the number of full-sib individuals. For example, if there are 100 full-sib individuals in one generation, they will be replaced with a single label "100" when compact = TRUE. The default value is FALSE.
outline	A logical value indicating whether shapes without labels will be shown. A graph of the pedigree without individual labels is shown when setting outline = TRUE.

	This is useful for viewing the pedigree outline and identifying immigrant individuals in each generation when the graph width exceeds the maximum PDF width (500 inches). The default value is FALSE.
`cex`	NULL or a numeric value changing the size of individual labels shown in the graph. *cex* is an abbreviation for 'character expansion factor'. The `visped` function will attempt to estimate (`cex=NULL`) the appropriate cex value and report it in the messages. Based on the reported cex from a previous run, this parameter should be increased if labels are wider than their shapes in the PDF; conversely, it should be decreased if labels are narrower than their shapes. The default value is NULL.
`showgraph`	A logical value indicating whether a plot will be shown in the default graphic device (e.g., the Plots panel in RStudio). This is useful for quick viewing without opening a PDF file. However, the graph on the default device may not be legible (e.g., overlapping labels or aliasing lines) due to size restrictions. It is recommended to set `showgraph = FALSE` for large pedigrees. The default value is TRUE.
`file`	NULL or a character value specifying whether the pedigree graph will be saved as a PDF file. The PDF output is a legible vector drawing where labels do not overlap, even with many individuals or long labels. It is recommended to save the pedigree graph as a PDF file. The default value is NULL.
`highlight`	NULL, a character vector of individual IDs, or a list specifying individuals to highlight. If a character vector is provided, individuals will be highlighted with a purple border while preserving their sex-based fill color. If a list is provided, it should contain: • `ids`: (required) character vector of individual IDs to highlight. • `frame.color`: (optional) hex color for the border of focal individuals. • `color`: (optional) hex color for the fill of focal individuals. • `rel.frame.color`: (optional) hex color for the border of relatives (used when `trace` is not NULL). • `rel.color`: (optional) hex color for the fill of relatives (used when `trace` is not NULL). For example: `c("A", "B")` or `list(ids = c("A", "B"), frame.color = "#9c27b0")`. The function will check if the specified individuals exist in the pedigree and issue a warning for any missing IDs. The default value is NULL.
`trace`	A logical value or a character string. If TRUE, all ancestors and descendants of the individuals specified in `highlight` will be highlighted. If a character string, it specifies the tracing direction: "**up**" (ancestors), "**down**" (descendants), or "**all**" (union of ancestors and descendants). This is useful for focusing on specific families within a large pedigree. The default value is FALSE.
`showf`	A logical value indicating whether inbreeding coefficients will be shown in the graph. If `showf = TRUE` and the column **f** exists in the pedigree, the inbreeding coefficient will be appended to the individual label, e.g., "ID (0.05)". The default value is FALSE.
`pagewidth`	A numeric value specifying the width of the PDF file in inches. This controls the horizontal scaling of the layout. The default value is 200.

`symbolsize`	A numeric value specifying the scaling factor for node size relative to the label size. Values greater than 1 increase the node size (adding padding around the label), while values less than 1 decrease it. This is useful for fine-tuning the whitespace and legibility of dense graphs. The default value is 1.
`maxiter`	An integer specifying the maximum number of iterations for the Sugiyama layout algorithm to minimize edge crossings. Higher values (e.g., 2000 or 5000) may result in fewer crossed lines for complex pedigrees but will increase computation time. The default value is 1000.
`...`	Additional arguments passed to `plot.igraph`.

Details

This function takes a pedigree tidied by the `tidyped` function and outputs a hierarchical graph for all individuals in the pedigree. The graph can be shown on the default graphic device or saved as a PDF file. The PDF output is a legible vector drawing that is legible and avoids overlapping labels. It is especially useful when the number of individuals is large and individual labels are long.

Rendering is performed using a Two-Pass strategy: edges are drawn first to ensure center-to-center connectivity, followed by nodes and labels. This ensures perfect visual alignment in high-resolution vector outputs. The function also supports real-time ancestry and descendant highlighting.

This function can draw the graph of a very large pedigree (> 10,000 individuals per generation) by compacting full-sib individuals. It is highly effective for aquatic animal pedigrees, which usually include many full-sib families per generation in nucleus breeding populations. The outline of a pedigree without individual labels is still shown if the width of a pedigree graph exceeds the maximum width (500 inches) of the PDF file.

In the graph, two shapes and three colors are used. Circles represent individuals, and squares represent families. Dark sky blue indicates males, dark goldenrod indicates females, and dark olive green indicates unknown sex. For example, a dark sky blue circle represents a male individual; a dark goldenrod square represents all female individuals in a full-sib family when `compact = TRUE`.

Value

The function mainly produces a plot on the current graphics device and/or a PDF file. It invisibly returns a list containing the graph object, layout coordinates, and node sizes.

Note

Isolated individuals (those with no parents and no progeny, assigned Gen 0) are automatically filtered out and not shown in the plot. A message will be issued if any such individuals are removed.

See Also

`tidyped` for tidying pedigree data (required input) `vismat` for visualizing relationship matrices as heatmaps `pedmat` for computing relationship matrices `splitped` for splitting pedigree into connected components `plot.igraph` underlying plotting function

Examples

```
library(visPedigree)
library(data.table)
# Drawing a simple pedigree
simple_ped_tidy <- tidyped(simple_ped)
visped(simple_ped_tidy,
       cex=0.25,
       symbolsize=5.5)

# Highlighting an individual and its ancestors and descendants
visped(simple_ped_tidy,
       highlight = "J5X804",
       trace = "all",
       cex=0.25,
       symbolsize=5.5)

# Showing inbreeding coefficients in the graph
simple_ped_tidy_inbreed <- tidyped(simple_ped, inbreed = TRUE)
visped(simple_ped_tidy_inbreed,
       showf = TRUE,
       cex=0.25,
       symbolsize=5.5)

# Adjusting page width and symbol size for better layout
# Increase pagewidth to spread nodes horizontally in the pdf file
# Increase symbolsize for more padding around individual labels
visped(simple_ped_tidy,
       cex=0.25,
       symbolsize=5.5,
       pagewidth = 100,
       file = tempfile(fileext = ".pdf"))

# Highlighting multiple individuals with custom colors
visped(simple_ped_tidy,
       highlight = list(ids = c("J3Y620", "J1X971"),
                        frame.color = "#4caf50",
                        color = "#81c784"),
       cex=0.25,
       symbolsize=5.5)

# Handling large pedigrees: Saving to PDF is recommended for legibility
# The 'trace' and 'tracegen' parameters in tidyped() help prune the graph
cand_labels <- big_family_size_ped[(Year == 2007) & (substr(Ind,1,2) == "G8"), Ind]

big_ped_tidy <- tidyped(big_family_size_ped,
                        cand = cand_labels,
                        trace = "up",
                        tracegen = 2)
# Use compact = TRUE for large families
visped(big_ped_tidy,
       compact = TRUE,
       cex=0.08,
       symbolsize=5.5,
       file = tempfile(fileext = ".pdf"))

# Use outline = TRUE if individual labels are not required
visped(big_ped_tidy,
       compact = TRUE,
       outline = TRUE,
       file = tempfile(fileext = ".pdf"))
```

Index

∗ **datasets**
    big_family_size_ped, 2
    deep_ped, 3
    loop_ped, 5
    simple_ped, 12
    small_ped, 12

big_family_size_ped, 2

deep_ped, 3

expand_pedmat, 3, *7*, *8*, *11*

histogram, *19*, *20*

inbreed, 4, *8*, *17*

levelplot, *19*, *20*
loop_ped, 5

pedmat, *3*, *4*, 5, *11*, *15*, *17*, *18*, *20*, *24*
plot.igraph, *24*
plot.tidyped, 9
print.summary.tidyped, 10
print.tidyped, 10

query_relationship, *4*, *7*, *8*, 11

simple_ped, 12
small_ped, 12
splitped, 13, *17*, *24*
summary.tidyped, 14, *17*
summary_pedmat, *7*, 15

tidyped, *5*, *6*, *8*, *13*, *14*, 16, *18*, *20*, *22*, *24*

vismat, *7*, *8*, *17*, 18, *24*
visped, *9*, *17*, *20*, 22