Package 'kmer'

January 23, 2026

Type Package

Title Fast K-Mer Counting and Clustering for Biological Sequence Analysis

Version 1.1.3

Author Shaun Wilkinson [aut, cre]

Maintainer Shaun Wilkinson <shaunpwilkinson@gmail.com>

Description Contains tools for rapidly computing distance matrices and clustering large sequence datasets using fast alignment-free k-mer counting and recursive k-means partitioning. See Vinga and Almeida (2003) <doi:10.1093/bioinformatics/btg005> for a review of k-mer counting methods and applications for biological sequence analysis.

License GPL-3

Encoding UTF-8

URL https://github.com/shaunpwilkinson/kmer

BugReports https://github.com/shaunpwilkinson/kmer/issues

Imports openssl, phylogram, Rcpp (>= 0.12.13)

Suggests ape (>= 4.0), dendextend, knitr, rmarkdown, testthat

LinkingTo Rcpp

VignetteBuilder knitr

RoxygenNote 7.3.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2026-01-23 06:40:52 UTC

Contents

`cluster` *Divisive k-means clustering.*

Description

This function recursively splits a sequence set into smaller and smaller subsets, returning a "dendrogram" object.

Usage

```
cluster(x, k = 5, residues = NULL, gap = "-", ...)
```

Arguments

`x`	a list or matrix of sequences, possibly an object of class `"DNAbin"` or `"AAbin"`.
`k`	integer. The k-mer size required.
`residues`	either NULL (default; emitted residues are automatically detected from the sequences), a case sensitive character vector specifying the residue alphabet, or one of the character strings "RNA", "DNA", "AA", "AMINO". Note that the default option can be slow for large lists of character vectors. Specifying the residue alphabet is therefore recommended unless the sequence list is a "DNAbin" or "AAbin" object.
`gap`	the character used to represent gaps in the alignment matrix (if applicable). Ignored for `"DNAbin"` or `"AAbin"` objects. Defaults to "-" otherwise.
`...`	further arguments to be passed to `kmeans` (not including `centers`).

Details

This function creates a tree by successively splitting the dataset into smaller and smaller subsets (recursive partitioning). This is a divisive, or "top-down" approach to tree-building, as opposed to agglomerative "bottom-up" methods such as neighbor joining and UPGMA. It is particularly useful for large large datasets with many sequences ($n > 10{,}000$) since the need to compute a large $n * n$ distance matrix is circumvented. Instead, a matrix of k-mer counts is computed, and split recursively row-wise using a k-means clustering algorithm ($k = 2$). This effectively reduces the time and memory complexity from quadratic to linear, while generally maintaining comparable accuracy.

If a more accurate tree is required, users can increase the value of `nstart` passed to `kmeans` *via* the `...` argument. While this can increase computation time, it can improve tree accuracy considerably.

DNA and amino acid sequences can be passed to the function either as a list of non-aligned sequences or a matrix of aligned sequences, preferably in the "DNAbin" or "AAbin" raw-byte format

(Paradis et al 2004, 2012; see the ape package documentation for more information on these S3 classes). Character sequences are supported; however ambiguity codes may not be recognized or treated appropriately, since raw ambiguity codes are counted according to their underlying residue frequencies (e.g. the 5-mer "ACRGT" would contribute 0.5 to the tally for "ACAGT" and 0.5 to that of "ACGGT").

To minimize computation time when counting longer k-mers (k > 3), amino acid sequences in the raw "AAbin" format are automatically compressed using the Dayhoff-6 alphabet as detailed in Edgar (2004). Note that amino acid sequences will not be compressed if they are supplied as a list of character vectors rather than an "AAbin" object, in which case the k-mer length should be reduced (k < 4) to avoid excessive memory use and computation time.

Value

Returns an object of class `"dendrogram"`.

Author(s)

Shaun Wilkinson

References

Edgar RC (2004) Local homology recognition and distance measures in linear time using compressed amino acid alphabets. *Nucleic Acids Research*, **32**, 380-385.

Paradis E, Claude J, Strimmer K, (2004) APE: analyses of phylogenetics and evolution in R language. *Bioinformatics* **20**, 289-290.

Paradis E (2012) Analysis of Phylogenetics and Evolution with R (Second Edition). Springer, New York.

See Also

kcount

Examples

```
## Not run:
## Cluster the woodmouse dataset (ape package)
library(ape)
data(woodmouse)
## trim gappy ends to subset global alignment
woodmouse <- woodmouse[, apply(woodmouse, 2, function(v) !any(v == 0xf0))]
## build tree divisively
suppressWarnings(RNGversion("3.5.0"))
set.seed(999)
woodmouse.tree <- cluster(woodmouse, nstart = 5)
## plot tree
op <- par(no.readonly = TRUE)
par(mar = c(5, 2, 4, 8) + 0.1)
plot(woodmouse.tree, main = "Woodmouse phylogeny", horiz = TRUE)
par(op)
```

```
## End(Not run)
```

---

kcount *K-mer counting.*

Description

Count all k-letter words in a sequence or set of sequences with a sliding window of length k.

Usage

```
kcount(
  x,
  k = 5,
  residues = NULL,
  gap = "-",
  named = TRUE,
  compress = TRUE,
  encode = FALSE
)
```

Arguments

x	a matrix of aligned sequences, a list of unaligned sequences, or a vector representing a single sequence. Accepted modes are "character" and "raw" (the latter being applicable for "DNAbin" and "AAbin" objects).
k	integer representing the k-mer size. Defaults to 5. Note that high values of k may be slow to compute and use a lot of memory due to the large numbers of calculations required, particularly when the residue alphabet is also large.
residues	either NULL (default; the residue alphabet is automatically detected from the sequences), a case sensitive character vector specifying the residue alphabet, or one of the character strings "RNA", "DNA", "AA", "AMINO". Note that the default option can be slow for large lists of character vectors. Specifying the residue alphabet is therefore recommended unless x is a "DNAbin" or "AAbin" object.
gap	the character used to represent gaps in the alignment matrix (if applicable). Ignored for `"DNAbin"` and `"AAbin"` objects. Defaults to "-" otherwise.
named	logical. Should the k-mers be returned as column names in the returned matrix? Defaults to TRUE.
compress	logical indicating whether to compress AAbin sequences using the Dayhoff(6) alphabet for k-mer sizes exceeding 4. Defaults to TRUE to avoid memory overflow and excessive computation time.
encode	logical indicating if the resulting matrix should be encoded in raw bytes (output matrix can be decoded with `kmer:::.decodekc()`). Note that the output will be rounded and have maximum k-mer count of 15.

Details

This function computes a vector or matrix of k-mer counts from a sequence or set of sequences using a sliding a window of length k. DNA and amino acid sequences can be passed to the function either as a list of non-aligned sequences or a matrix of aligned sequences, preferably in the "DNAbin" or "AAbin" raw-byte format (Paradis et al 2004, 2012; see the ape package documentation for more information on these S3 classes). Character sequences are supported; however ambiguity codes may not be recognized or treated appropriately, since raw ambiguity codes are counted according to their underlying residue frequencies (e.g. the 5-mer "ACRGT" would contribute 0.5 to the tally for "ACAGT" and 0.5 to that of "ACGGT").

To minimize computation time when counting longer k-mers (k > 3), amino acid sequences in the raw "AAbin" format are automatically compressed using the Dayhoff-6 alphabet as detailed in Edgar (2004). Note that amino acid sequences will not be compressed if they are supplied as a list of character vectors rather than an "AAbin" object, in which case the k-mer length should be reduced (k < 4) to avoid excessive memory use and computation time.

Value

Returns a matrix of k-mer counts with one row for each sequence and *n^k* columns (where *n* is the size of the residue alphabet and *k* is the k-mer size)

Author(s)

Shaun Wilkinson

References

Edgar RC (2004) Local homology recognition and distance measures in linear time using compressed amino acid alphabets. *Nucleic Acids Research*, **32**, 380-385.

Paradis E, Claude J, Strimmer K, (2004) APE: analyses of phylogenetics and evolution in R language. *Bioinformatics* **20**, 289-290.

Paradis E (2012) Analysis of Phylogenetics and Evolution with R (Second Edition). Springer, New York.

See Also

kdistance for k-mer distance matrix computation.

Examples

```
## compute a matrix of k-mer counts for the woodmouse
## data (ape package) using a k-mer size of 3
library(ape)
data(woodmouse)
x <- kcount(woodmouse, k = 3)
x
## 64 columns for nucleotide 3-mers AAA, AAC, ... TTT
## convert to AAbin object and repeat the operation
y <- kcount(ape::trans(woodmouse, 2), k = 2)
y
```

```
## 400 columns for amino acid 2-mers AA, AB, ... , YY
```

---

kdistance *K-mer distance matrix computation.*

Description

Computes the matrix of k-mer distances between all pairwise comparisons of a set of sequences.

Usage

```
kdistance(
  x,
  k = 5,
  method = "edgar",
  residues = NULL,
  gap = "-",
  compress = TRUE,
  ...
)
```

Arguments

x	a matrix of aligned sequences or a list of unaligned sequences. Accepted modes are "character" and "raw" (the latter being applicable for "DNAbin" and "AAbin" objects).
k	integer representing the k-mer size to be used for calculating the distance matrix. Defaults to 5. Note that high values of k may be slow to compute and use a lot of memory due to the large numbers of calculations required, particularly when the residue alphabet is also large.
method	a character string giving the k-mer distance measure to be used. Currently the available options are "edgar" (default; see Edgar (2004) for details) and the standard methods available for the base function "dist" ("euclidean", "maximum", "manhattan", "canberra", "binary" and "minkowski").
residues	either NULL (default; the residue alphabet is automatically detected from the sequences), a case sensitive character vector specifying the residue alphabet, or one of the character strings "RNA", "DNA", "AA", "AMINO". Note that the default option can be slow for large lists of character vectors. Specifying the residue alphabet is therefore recommended unless x is a "DNAbin" or "AAbin" object.
gap	the character used to represent gaps in the alignment matrix (if applicable). Ignored for "DNAbin" or "AAbin" objects. Defaults to "-" otherwise.
compress	logical indicating whether to compress AAbin sequences using the Dayhoff(6) alphabet for k-mer sizes exceeding 4. Defaults to TRUE to avoid memory overflow and excessive computation time.
...	further arguments to be passed to "as.dist".

Details

This function computes the *n* * *n* k-mer distance matrix (where *n* is the number of sequences), returning an object of class `"dist"`. DNA and amino acid sequences can be passed to the function either as a list of non-aligned sequences or as a matrix of aligned sequences, preferably in the "DNAbin" or "AAbin" raw-byte format (Paradis et al 2004, 2012; see the ape package documentation for more information on these S3 classes). Character sequences are supported; however ambiguity codes may not be recognized or treated appropriately, since raw ambiguity codes are counted according to their underlying residue frequencies (e.g. the 5-mer "ACRGT" would contribute 0.5 to the tally for "ACAGT" and 0.5 to that of "ACGGT").

To minimize computation time when counting longer k-mers (k > 3), amino acid sequences in the raw "AAbin" format are automatically compressed using the Dayhoff-6 alphabet as detailed in Edgar (2004). Note that amino acid sequences will not be compressed if they are supplied as a list of character vectors rather than an "AAbin" object, in which case the k-mer length should be reduced (k < 4) to avoid excessive memory use and computation time.

Value

an object of class `"dist"`.

Author(s)

Shaun Wilkinson

References

Edgar RC (2004) Local homology recognition and distance measures in linear time using compressed amino acid alphabets. *Nucleic Acids Research*, **32**, 380-385.

Paradis E, Claude J, Strimmer K, (2004) APE: analyses of phylogenetics and evolution in R language. *Bioinformatics* **20**, 289-290.

Paradis E (2012) Analysis of Phylogenetics and Evolution with R (Second Edition). Springer, New York.

See Also

`kcount` for k-mer counting, and `mbed` for leaner distance matrices

Examples

```
## compute a k-mer distance matrix for the woodmouse
## dataset (ape package) using a k-mer size of 5
library(ape)
data(woodmouse)
### subset global alignment by removing gappy ends
woodmouse <- woodmouse[, apply(woodmouse, 2, function(v) !any(v == 0xf0))]
### compute the distance matrix
woodmouse.dist <- kdistance(woodmouse, k = 5)
### cluster and plot UPGMA tree
woodmouse.tree <- as.dendrogram(hclust(woodmouse.dist, "average"))
plot(woodmouse.tree)
```

kmer *Fast K-mer Counting and Clustering for Biological Sequence Analysis.*

Description

The kmer package contains tools for rapidly computing distance matrices, building large trees, and clustering operational taxonomic units using fast alignment-free k-mer counting and divisive clustering techniques.

Functions

A breif description of the primary **kmer** functions are provided with links to their help pages below.

K-mer counting

- `kcount` counts all k-letter words in a sequence or set of sequences using a sliding window of length k

Distance matrix computation

- `kdistance` calculates pairwise distances between sequences based on k-mer counts
- `mbed` embeds sequences as vectors of k-mer distances to a set of seed' sequences

Alignment-free clustering

- `cluster` builds a phylogenetic tree by successively splitting a set of sequences (recursive partitioning) based on k-mer counts
- `otu` heirarchically clusters a set of sequences until a predefined furthest neighbor dissimilarity threshold is reached.

Author(s)

Maintainer: Shaun Wilkinson <shaunpwilkinson@gmail.com>

See Also

Useful links:

- https://github.com/shaunpwilkinson/kmer
- Report bugs at https://github.com/shaunpwilkinson/kmer/issues

mbed *Convert sequences to vectors of distances to a subset of seed sequences.*

Description

This function computes a matrix of distances from each sequence to a subset of 'seed' sequences using the method outlined in Blacksheilds et al (2010).

Usage

```
mbed(x, seeds = NULL, k = 5, residues = NULL, gap = "-", counts = FALSE)
```

Arguments

x	a matrix of aligned sequences or a list of unaligned sequences. Accepted modes are "character" and "raw" (the latter is for "DNAbin" and "AAbin" objects).
seeds	optional integer vector indicating which sequences should be used as the seed sequences. If `seeds = NULL` a set of log(*n*, 2)^2 non-identical sequences is randomly selected from the sequence set (where *n* is the number of sequences; see Blacksheilds et al. 2010). Alternatively, if `seeds = 'all'` a standard *n* * *n* distance matrix is computed.
k	integer representing the k-mer size to be used for calculating the distance matrix. Defaults to 5. Note that high values of k may be slow to compute and use a lot of memory due to the large numbers of calculations required, particularly when the residue alphabet is also large.
residues	either NULL (default; emitted residues are automatically detected from the sequences), a case sensitive character vector specifying the residue alphabet, or one of the character strings "RNA", "DNA", "AA", "AMINO". Note that the default option can be slow for large lists of character vectors. Specifying the residue alphabet is therefore recommended unless x is a "DNAbin" or "AAbin" object.
gap	the character used to represent gaps in the alignment matrix (if applicable). Ignored for `"DNAbin"` or `"AAbin"` objects. Defaults to "-" otherwise.
counts	logical indicating whether the (usually large) matrix of k-mer counts should be returned as an attribute of the returned object. Defaults to FALSE.

Details

This function computes a *n* * log(*n*, 2)^2 k-mer distance matrix (where *n* is the number of sequences), returning an object of class `"mbed"`. If the number of sequences is less than or equal to 19, the full *n* * *n* distance matrix is produced (since the rounded up value of log(*19*, 2)^2 is 19). Currently the only distance measure supported is that of Edgar (2004).

For maximum information retention following the embedding process it is generally desirable to select the seed sequences based on their uniqueness, rather than simply selecting a random subset

(Blackshields et al. 2010). Hence if 'seeds' is set to NULL (the default setting) the the 'mbed' function selects the subset by clustering the sequence set into t groups using the k-means algorithm ($k = t$), and choosing one representative from each group. Users can alternatively pass an integer vector (as in the above example) to specify the seeds manually. See Blackshields et al (2010) for other seed selection options.

DNA and amino acid sequences can be passed to the function either as a list of non-aligned sequences or as a matrix of aligned sequences, preferably in the "DNAbin" or "AAbin" raw-byte format (Paradis et al 2004, 2012; see the ape package documentation for more information on these S3 classes). Character sequences are supported; however ambiguity codes may not be recognized or treated appropriately, since raw ambiguity codes are counted according to their underlying residue frequencies (e.g. the 5-mer "ACRGT" would contribute 0.5 to the tally for "ACAGT" and 0.5 to that of "ACGGT").

To minimize computation time when counting longer k-mers (k > 3), amino acid sequences in the raw "AAbin" format are automatically compressed using the Dayhoff-6 alphabet as detailed in Edgar (2004). Note that amino acid sequences will not be compressed if they are supplied as a list of character vectors rather than an "AAbin" object, in which case the k-mer length should be reduced (k < 4) to avoid excessive memory use and computation time.

Note that agglomerative (bottom-up) tree-building methods such as neighbor-joining and UPGMA depend on a full $n * n$ distance matrix. See the kdistance function for details on computing symmetrical distance matrices.

Value

Returns an object of class "mbed", whose primary object is an $n * \log(n, 2)^2$ matrix (where n is the number of sequences). The returned object contains additional attributes including an integer vector of seed sequence indices ("seeds"), a logical vector identifying the duplicated sequences ("duplicates"), an integer vector giving the matching indices of the non-duplicated sequences ("pointers"), a character vector of MD5 digests of the sequences ("hashes"), an integer vector of sequence lengths ("seqlengths"), and if counts = TRUE, the matrix of k-mer counts ("kcounts"; see kcount for details).

Author(s)

Shaun Wilkinson

References

Blackshields G, Sievers F, Shi W, Wilm A, Higgins DG (2010) Sequence embedding for fast construction of guide trees for multiple sequence alignment. *Algorithms for Molecular Biology*, **5**, 21.

Edgar RC (2004) Local homology recognition and distance measures in linear time using compressed amino acid alphabets. *Nucleic Acids Research*, **32**, 380-385.

Paradis E, Claude J, Strimmer K, (2004) APE: analyses of phylogenetics and evolution in R language. *Bioinformatics* **20**, 289-290.

Paradis E (2012) Analysis of Phylogenetics and Evolution with R (Second Edition). Springer, New York.

See Also

kdistance for full *n* * *n* distance matrix computation.

Examples

```
## compute an embedded k-mer distance matrix for the woodmouse
## dataset (ape package) using a k-mer size of 5
library(ape)
data(woodmouse)
## randomly select three sequences as seeds
suppressWarnings(RNGversion("3.5.0"))
set.seed(999)
seeds <- sample(1:15, size = 3)
## embed the woodmouse dataset in three dimensions
woodmouse.mbed <- mbed(woodmouse, seeds = seeds, k = 5)
## print the distance matrix (without attributes)
print(woodmouse.mbed[,], digits = 2)
```

---

otu *Cluster sequences into operational taxonomic units.*

Description

This function performs divisive heirarchical clustering on a set of DNA sequences using sequential k-means partitioning, returning an integer vector of OTU membership.

Usage

```
otu(
  x,
  k = 5,
  threshold = 0.97,
  method = "central",
  residues = NULL,
  gap = "-",
  ...
)
```

Arguments

x	a "DNAbin" object.
k	integer giving the k-mer size used to generate the input matrix for k-means clustering.
threshold	numeric between 0 and 1 giving the OTU identity cutoff. Defaults to 0.97.

`method` the maximum distance criterion to use for terminating the recursive partitioning procedure. Accepted options are "central" (splitting stops if the similarity between the central sequence and its farthest neighbor within the cluster is greater than the threshold), "centroid" (splitting stops if the similarity between the centroid and its farthest neighbor within the cluster is greater than the threshold), and "farthest" (splitting stops if the similarity between the two farthest sequences within the cluster is greater than the threshold). Defaults to "central".

`residues` either NULL (default; emitted residues are automatically detected from the sequences), a case sensitive character vector specifying the residue alphabet, or one of the character strings "RNA", "DNA", "AA", "AMINO". Note that the default option can be slow for large lists of character vectors. Specifying the residue alphabet is therefore recommended unless the sequence list is a "DNAbin" or "AAbin" object.

`gap` the character used to represent gaps in the alignment matrix (if applicable). Ignored for `"DNAbin"` or `"AAbin"` objects. Defaults to "-" otherwise.

`...` further arguments to be passed to `kmeans` (not including `centers`).

Details

This function clusters sequences into OTUs by first generating a matrix of k-mer counts, and then splitting the matrix into two subsets (row-wise) using the k-means algorithm ($k = 2$). The splitting continues recursively until the farthest k-mer distance in every cluster is below the threshold value.

This is a divisive, or "top-down" approach to OTU clustering, as opposed to agglomerative "bottom-up" methods. It is particularly useful for large large datasets with many sequences ($n > 10,000$) since the need to compute a large $n * n$ distance matrix is circumvented. This effectively reduces the time and memory complexity from quadratic to linear, while generally maintaining comparable accuracy.

It is recommended to increase the value of `nstart` passed to `kmeans` *via* the `...` argument to at least 20. While this can increase computation time, it can improve clustering accuracy considerably.

DNA and amino acid sequences can be passed to the function either as a list of non-aligned sequences or a matrix of aligned sequences, preferably in the "DNAbin" or "AAbin" raw-byte format (Paradis et al 2004, 2012; see the ape package documentation for more information on these S3 classes). Character sequences are supported; however ambiguity codes may not be recognized or treated appropriately, since raw ambiguity codes are counted according to their underlying residue frequencies (e.g. the 5-mer "ACRGT" would contribute 0.5 to the tally for "ACAGT" and 0.5 to that of "ACGGT").

To minimize computation time when counting longer k-mers ($k > 3$), amino acid sequences in the raw "AAbin" format are automatically compressed using the Dayhoff-6 alphabet as detailed in Edgar (2004). Note that amino acid sequences will not be compressed if they are supplied as a list of character vectors rather than an "AAbin" object, in which case the k-mer length should be reduced ($k < 4$) to avoid excessive memory use and computation time.

Value

a named integer vector of cluster membership with values ranging from 1 to the total number of OTUs. Asterisks indicate the representative sequence within each cluster.

Author(s)

Shaun Wilkinson

References

Edgar RC (2004) Local homology recognition and distance measures in linear time using compressed amino acid alphabets. *Nucleic Acids Research*, **32**, 380-385.

Paradis E, Claude J, Strimmer K, (2004) APE: analyses of phylogenetics and evolution in R language. *Bioinformatics* **20**, 289-290.

Paradis E (2012) Analysis of Phylogenetics and Evolution with R (Second Edition). Springer, New York.

Examples

```
## Not run:
## Cluster the woodmouse dataset (from the ape package) into OTUs
library(ape)
data(woodmouse)
## trim gappy ends to subset global alignment
woodmouse <- woodmouse[, apply(woodmouse, 2, function(v) !any(v == 0xf0))]
## cluster sequences into OTUs at 0.97 threshold with kmer size = 5
suppressWarnings(RNGversion("3.5.0"))
set.seed(999)
woodmouse.OTUs <- otu(woodmouse, k = 5, threshold = 0.97, nstart = 20)
woodmouse.OTUs

## End(Not run)
```

Index