Package 'microclass'

October 13, 2022

Encoding UTF-8

Type Package

Title Methods for Taxonomic Classification of Prokaryotes

Version 1.2

Date 2020-08-11

Author Kristian Hovde Liland, Hilde Vinje, Lars Snipen

Maintainer Lars Snipen <lars.snipen@nmbu.no>

Description Functions for assigning 16S sequence data to a taxonomic level in the tree-of-life for prokaryotes.

License GPL (>= 2)

Depends R (>= 3.5.0), microseq, microcontax, dplyr, stringr, rlang

Imports Rcpp (>= 0.11.1), RcppParallel, tibble

LinkingTo Rcpp (>= 0.11.1), RcppEigen, RcppParallel

SystemRequirements GNU make

RoxygenNote 7.1.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-08-28 12:20:08 UTC

R topics documented:

`microclass-package` *Methods for 16S based taxonomic classification of prokaryotes*

Description

The package provides functions for assigning 16S sequence data to a taxonomic level in the tree-of-life for prokaryotes.

Usage

```
microclass()
```

Details

Package:	microclass
Type:	Package
Version:	1.0.1
Date:	2016-08-31
License:	GPL-2

Author(s)

Hilde Vinje, Kristian Hovde Liland, Lars Snipen.
Maintainer: Lars Snipen <lars.snipen@nmbu.no>

`blastClassify16S` *Classifying using BLAST*

Description

A 16S based classification based on BLAST.

Usage

```
blastClassify16S(sequence, bdb)
```

Arguments

sequence	Character vector of 16S sequences to classify.
bdb	Name of BLAST data base, see blastDbase16S.

Details

A vector of 16S sequences (DNA) are classified by first using BLAST `blastn` against a database of 16S DNA sequences, and then classify according to the nearest-neighbour principle. The nearest neighbour of a query sequence is the hit with the largest bitscore. The blast+ software https://blast.ncbi.nlm.nih.gov/Blast.cgi?PAGE_TYPE=BlastDocs&DOC_TYPE=Download must be installed on the system. Type `system("blastn -help")` in the Console window, and a sensible Help-text should appear.

The database must contain 16S sequences where the Header starts with a token specifying the taxon. More specifically, the tokens must look like:

<taxon>_1

<taxon>_2

...etc

where <taxon> is some proper taxon name. Use blastDbase16S to make such databases.

The identity of each alignment is also computed. This should be close to 1.0 for a classification to be trusted. Identity values below 0.95 could indicate uncertain classifications, but this will vary between taxa.

Value

A `data.frame` with two columns: Taxon is the predicted taxon for each `sequence` and Identity is the corresponding identity-value. If no BLAST hit is seen, the sequence is `"unclassified"`.

Author(s)

Lars Snipen.

See Also

blastDbase16S.

Examples

```
data("small.16S")
## Not run:
dbase <- blastDbase16S("test", small.16S$Sequence, word(small.16S$Header, 2, 2))
reads <- str_sub(small.16S$Sequence, 100, 550)
blastClassify16S(reads, dbase) %>%
  bind_cols(small.16S) -> tbl

## End(Not run)
```

`blastDbase16S` *Building a BLAST database*

Description

Building a BLAST database for 16S based classification.

Usage

```
blastDbase16S(name, sequence, taxon)
```

Arguments

`name`	The name of the database (text).
`sequence`	A character vector with 16S sequence data.
`taxon`	A character vector with taxon information.

Details

This functions builds a database using the `makeblastdb` program of the BLAST+ software https://blast.ncbi.nlm.nih.gov/Blast.cgi?PAGE_TYPE=BlastDocs&DOC_TYPE=Download. Thus, this software must be available on the system when using this function. If you type `system("makeblastdb -help")` in the Console window some meaningful Help-text should be displayed.

This function is most typically used prior to `blastClassify16S` to set up the database before searching and classifying. It can be seen as the 'training step' of a BLAST-based classification procedure.

The `sequence` must be a vector of DNA-sequences (16S sequences). The `taxon` is a vector of the same length as `sequence`, containing the correpsonding taxon information.

Value

The database files are created, and the name of the database (`name`) is returned.

Author(s)

Lars Snipen.

See Also

`blastClassify16S`.

Examples

```
# See examples for blastClassify16S.
```

KmerCount *K-mer counting*

Description

Counting overlapping words of length K in DNA/RNA sequences.

Usage

```
KmerCount(sequences, K = 1, col.names = FALSE)
```

Arguments

sequences	Vector of sequences (text).
K	Word length (integer).
col.names	Logical indicating if the words should be added as columns names.

Details

For each input sequence, the frequency of every word of length K is counted. Counting is done with overlap. The counting itself is done by a C++ function.

With col.names=TRUE the K-mers are added as column names, but this makes the computations slower.

Value

A matrix with one row for each sequence in sequences and one column for each possible word of lengthK.

Author(s)

Kristian Hovde Liland and Lars Snipen.

See Also

multinomTrain, multinomClassify.

Examples

```
KmerCount("ATGCCTGAACTGACCTGC",K=2)
```

`multinomClassify` *Classifying with a Multinomial model*

Description

Classifying sequences by a trained Multinomial model.

Usage

```
multinomClassify(sequence, trained.model, post.prob = FALSE, prior = FALSE)
```

Arguments

`sequence`	Character vector of 16S sequences to classify.
`trained.model`	A list with a trained model, see `multinomTrain`.
`post.prob`	Logical indicating if posterior log-probabilities should be returned.
`prior`	Logical indicating if classification should be done by flat priors (default) or with empirical priors (prior=TRUE).

Details

The classification step of the Multinomial method (Vinje et al, 2015) means counting K-mers on all sequences, and computing the posterior probabilities for each taxon in the trained model. The predicted taxon for each input sequence is the one with the maximum posterior probability for that sequence.

By setting `post.prob=TRUE` you will get the log-probability of the best and second best taxon for each sequence. This can be used for evaluating the certainty in the classifications, see `taxMachine`.

The classification is parallelized through RcppParallel employing Intel TBB and TinyThread. By default all available processing cores are used. This can be changed using the function `setParallel`.

Value

If `post.prob=FALSE` a character vector of predicted taxa is returned.

If `post.prob=TRUE` a `data.frame` with three columns is returned. Taxon is the vector of predicted taxa, one for each sequence in `sequence`. The Post.prob.1 and Post.prob.2 are vectors with the maximum and second largest posterior log-probabilities for each sequence.

Author(s)

Kristian Hovde Liland and Lars Snipen.

References

Vinje, H, Liland, KH, Almøy, T, Snipen, L. (2015). Comparing K-mer based methods for improved classification of 16S sequences. BMC Bioinformatics, 16:205.

See Also

KmerCount, multinomTrain.

Examples

```
data("small.16S")
seq <- small.16S$Sequence
tax <- sapply(strsplit(small.16S$Header,split=" "),function(x){x[2]})
## Not run:
trn <- multinomTrain(seq,tax)
primer.515f <- "GTGYCAGCMGCCGCGGTAA"
primer.806rB <- "GGACTACNVGGGTWTCTAAT"
reads <- amplicon(seq, primer.515f, primer.806rB)
predicted <- multinomClassify(unlist(reads[nchar(reads)>0]),trn)
print(predicted)

## End(Not run)
```

---

multinomTrain *Training multinomial model*

Description

Training the multinomial K-mer method on sequence data.

Usage

```
multinomTrain(sequence, taxon, K = 8, col.names = FALSE, n.pseudo = 100)
```

Arguments

sequence	Character vector of 16S sequences.
taxon	Character vector of taxon labels for each sequence.
K	Word length (integer).
col.names	Logical indicating if column names should be added to the trained model matrix.
n.pseudo	Number of pseudo-counts to use (positive numerics, need not be integer). Special case -1 will only return word counts, not log-probabilities.

Details

The training step of the multinomial method (Vinje et al, 2015) means counting K-mers on all sequences and compute the multinomial probabilities for each K-mer for each unique taxon. n.pseudo pseudo-counts are added, divided equally over all K-mers, before probabilities are estimated. The optimal choice of n.pseudo will depend on K and the training data set. The default value n.pseudo=100 has proven good for K=8 and the contax.trim data set (see the microcontax R-package).

Adding the actual K-mers as column names (col.names=TRUE) will slow down the computations.

The relative taxon sizes are also computed, and may be used as an empirical prior in the classification step (see "prior" below).

Value

A list with two elements. The first element is `Method`, which is the text `"multinom"` in this case. The second element is `Fitted`, which is a matrix of probabilities with one row for each unique `taxon` and one column for each possible word of length`K`. The sum of each row is 1.0. No probabilities are 0 if `n.pseudo`>0.0.

The matrix `Fitted` has an attribute `attr("prior",)`, that contains the relative taxon sizes.

Author(s)

Kristian Hovde Liland and Lars Snipen.

References

Vinje, H, Liland, KH, Almøy, T, Snipen, L. (2015). Comparing K-mer based methods for improved classification of 16S sequences. BMC Bioinformatics, 16:205.

See Also

`KmerCount`, `multinomClassify`.

Examples

```
# See examples for multinomClassify
```

`rdpClassify` *Classifying with the RDP classifier*

Description

Classifying sequences by a trained presence/absence K-mer model.

Usage

```
rdpClassify(sequence, trained.model, post.prob = FALSE, prior = FALSE)
```

Arguments

`sequence`	Character vector of sequences to classify.
`trained.model`	A list with a trained model, see `rdpTrain`.
`post.prob`	Logical indicating if posterior log-probabilities should be returned.
`prior`	Logical indicating if classification should be done by flat priors (default) or with empirical priors (prior=TRUE).

Details

The classification step of the presence/absence method known as the RDP classifier (Wang et al 2007) means looking for K-mers on all sequences, and computing the posterior probabilities for each taxon using a trained model and a naive Bayes assumption. The predicted taxon is the one producing the maximum posterior probability, for each sequence.

The classification is parallelized through RcppParallel employing Intel TBB and TinyThread. By default all available processing cores are used. This can be changed using the function setParallel.

Value

A character vector with the predicted taxa, one for each sequence.

Author(s)

Kristian Hovde Liland and Lars Snipen.

References

Wang, Q, Garrity, GM, Tiedje, JM, Cole, JR (2007). Naive Bayesian Classifier for Rapid Assignment of rRNA Sequences into the New Bacterial Taxonomy. Applied and Enviromental Microbiology, 73: 5261-5267.

See Also

rdpTrain.

Examples

```
data("small.16S")
seq <- small.16S$Sequence
tax <- sapply(strsplit(small.16S$Header,split=" "),function(x){x[2]})
## Not run:
trn <- rdpTrain(seq,tax)
primer.515f <- "GTGYCAGCMGCCGCGGTAA"
primer.806rB <- "GGACTACNVGGGTWTCTAAT"
reads <- amplicon(seq, primer.515f, primer.806rB)
predicted <- rdpClassify(unlist(reads[nchar(reads)>0]),trn)
print(predicted)

## End(Not run)
```

rdpTrain *Training the RDP classifier*

Description

Training the RDP presence/absence K-mer method on sequence data.

Usage

```
rdpTrain(sequence, taxon, K = 8, cnames = FALSE)
```

Arguments

sequence	Character vector of 16S sequences.
taxon	Character vector of taxon labels for each sequence.
K	Word length (integer).
cnames	Logical indicating if column names should be added to the trained model matrix.

Details

The training step of the RDP method means looking for K-mers on all sequences, and computing the probability of each K-mer being present for each unique taxon. This is an attempt to re-implement the method described by Wang et tal (2007), but without the bootstrapping. See that publications for all details.

The word-length K is by default 8, since this is the value used by Wang et al. Larger values may lead to memory-problems since the trained model is a matrix with 4^K columns. Adding the K-mers as column names will slow down all computations.

The relative taxon sizes are also computed, and returned as an attribute to the model matrix. They may be used as empirical priors in the classification step.

Value

A list with two elements. The first element is Method, which is the text "RDPclassifier" in this case. The second element is Fitted, which is a matrix with one row for each unique taxon and one column for each possible word of length K. The value in row i and column j is the probability that word j is present in taxon i.

Author(s)

Kristian Hovde Liland and Lars Snipen.

References

Wang, Q, Garrity, GM, Tiedje, JM, Cole, JR (2007). Naive Bayesian Classifier for Rapid Assignment of rRNA Sequences into the New Bacterial Taxonomy. Applied and Enviromental Microbiology, 73: 5261-5267.

See Also

rdpClassify.

Examples

```
# See examples for rdpClassify.
```

---

setParallel *Set number of parallel threads*

Description

Simple function to set the number of threads to use in parallel computations. The default equals all available logical cores. An integer is interpreted as the number of threads. A numeric < 1 is interpreted as a proportion of the avialable logical cores.

Usage

```
setParallel(C = NULL)
```

Arguments

C	a scalar indicating the number of threads, default = NULL (#available logical cores)

Value

NULL, returned silently.

Examples

```
## Not run:
setParallel() # Use all available logical cores.

## End(Not run)
```

small.16S *A small example data set*

Description

A tibble object (data.frame) with some 16S sequences with taxon information.

Usage

```
data(small.16S)
```

Details

This is a tibble object (data.frame) with 71 sequences used in some examples. The taxonomy information for each sequence follows the ConTax format, see the microcontax package for more details.

Author(s)

Hilde Vinje, Kristian Hovde Liland, Lars Snipen.

Examples

```
data(small.16S)
str(small.16S)
```

taxMachine *Classifying 16S sequences*

Description

Optimized classification of 16S sequence data.

Usage

```
taxMachine(
  sequence,
  model.in.memory = TRUE,
  model.on.disk = FALSE,
  verbose = TRUE,
  chunk.size = 10000
)
```

Arguments

sequence	Character vector with DNA sequences.
model.in.memory	Logical indicating if model should be cached in memory (default=TRUE).
model.on.disk	Logical or text, for reading/saving models, see Deatils below (default=FALSE).
verbose	Logical, if TRUE progress is reported during computations (default=TRUE).
chunk.size	The number of sequence to classify in each iteration of the loop (default=10000).

Details

This function provides optimized taxonomy classifications from 16S sequence data.

All sequences are classified to the genus level based on a Multinomial model (see multinomTrain) trained on the designed consensus taxonomy data set contax.trim found in the R-package microcontax. The word length K=8 has been used in the model.

To avoid saving fitted models in the package, a model is trained the first time you run taxMachine in an R session. This takes only a few seconds, and the result is cached for latter use if model.in.memory is TRUE.

If a path to an existing file with a trained model is supplied in model.on.disk, this Multinomial model is read from the file and used. If a path to a new file is supplied, the trained Multinomial model will be saved to that file. The default (model.on.disk=FALSE), means no files are read/saved, while model.on.disk=TRUE will attempt to load/save models from the microclass/extdata directory.

Both verbose and chunk.size are used to monitor the progress, which is nice when classifying huge data sets, since this will take some time.

Value

A data.frame with one row for each sequence. The columns are Genus, D.score, R.score and P.recognize.

Genus is the predicted genus for each sequence. Note that all sequences get a prediction, but may still be more or less reliable.

The D.score is a measure of how the predicted genus wins over all other genera in the race for being the chosen one. A large D.score means the winner stands out clearly, and we can be confident it is the correct genus. A D.score close to 0 means we have an uncertain classification. Only D.scores below 1.0, should be of any concern, see Liland et al (2016) for details.

The R.score is a measure of the models ability to recognize the sequence. The more negative the R.score gets, the more unusual the sequence is compared to the training set (the contax.trim data set). The P.recognize is a rough probability of seing an R.score this small, or smaller, given the training data. Thus, a very small P.recognize means the sequence is not really recognized, and the classification is worthless. A very negative R.score indicates either not 16S at all, many sequencing errors that has destroyed the read, or a completely new taxon never seen before. See Liland et al (2016) for details.

Author(s)

Lars Snipen and Kristian Hovde Liland

References

Liland, KH, Vinje, H, Snipen, L (2016). microclass - An R-package for 16S taxonomy classification. BMC Bioinformatics, xx:yy.

See Also

KmerCount, multinomClassify.

Examples

```
## Not run:
data(small.16S)
tax.tab <- taxMachine(small.16S$Sequence)

## End(Not run)
```

Index