Package ‘seqR’

October 6, 2021

Type Package
Title Fast and Comprehensive K-Mer Counting Package
Version 1.0.1
Date 2021-09-27
Description A fast implementation of k-mer (subsequences of length k) counting with the support for all types of biological sequences and k-mer variants.
License GPL (>= 3)
Imports Matrix, Rcpp (>= 1.0.4), RcppParallel (>= 5.1.2), rlang, slam
Depends R(>= 3.6)
LinkingTo Rcpp, RcppParallel
Suggests covr, ggplot2, knitr, microbenchmark, pryr, rmarkdown, spelling, testthat
VignetteBuilder knitr
SystemRequirements c++17, GNU make
NeedsCompilation yes
RoxygenNote 7.1.1
Encoding UTF-8
Language en-US
URL https://github.com/slowikj/seqR
BugReports https://github.com/slowikj/seqR/issues
LazyData TRUE
Collate 'CsgA.R' 'RcppExports.R' 'atomic_validators.R' 'common.R'
  'seqR_result.R' 'kmer_TYPED_FUNCTIONS.R'
  'kmer_FUNCTIONS_PROVIDER.R' 'count_kmers.R' 'count_multimers.R'
  'package.R' 'zzz.R'

Author Jadwiga Słowik [aut, cre] (<https://orcid.org/0000-0003-3466-8933>),
  Michał Burdukiewicz [ths] (<https://orcid.org/0000-0001-8926-582X>)
Maintainer Jadwiga Słowik <jadwiga.slowik5@gmail.com>
Repository CRAN
Date/Publication 2021-10-05 23:20:13 UTC
R topics documented:

seqR-package ............................................................... 2
count_kmers ............................................................... 3
count_multimers .......................................................... 5
CsgA ................................................................. 7
rbind_columnwise ......................................................... 8

Description

The seqR package provides in-memory, probabilistic, highly-optimized, and multi-threaded implementation of k-mer counting.

Author(s)

Jadwiga Słowik

Examples

# Load exemplary sequences
data(CsgA)

# Counting 1-mers (amino acid composition)
count_kmers(
  CsgA,
  k = 1,
  batch_size = 1)

# Counting 1-mers and 2-mers
count_multimers(
  CsgA,
  k_vector = c(1, 2),
  batch_size = 1)
**count_kmers**

**Count k-mers of one, particular type for a given collection of sequences**

---

**Description**

This is an in-memory, probabilistic, highly-optimized, and multi-threaded implementation of k-mer counting algorithm.

The function supports:

1. several types of k-mers (contiguous, gapped, and positional variants)
2. all biological sequences (in particular, nucleic acids and proteins)
3. two common in-memory representations of sequences, i.e., string vectors and list of string vectors

Moreover, several extra features are provided (for more information see details’):

1. configurable k-mer alphabet (i.e., which elements of a sequence should be considered during the k-mer counting procedure)
2. verbose mode
3. configurable batch size (i.e., how many sequences are processed in a single step)
4. configurable dimension of the hash value of a k-mer
5. possibility to compute k-mers with or without their frequencies
6. possibility to compute a result k-mer matrix with or without human-readable k-mer (column) names

**Usage**

```r
count_kmers(
  sequences,
  k = length(kmer_gaps) + 1,
  kmer_alphabet = getOption("seqR_kmer_alphabet_default"),
  positional = getOption("seqR_positional_default"),
  kmer_gaps = c(),
  with_kmer_counts = getOption("seqR_with_kmer_counts_default"),
  with_kmer_names = getOption("seqR_with_kmer_names_default"),
  batch_size = getOption("seqR_batch_size_default"),
  hash_dim = getOption("seqR_hash_dim_default"),
  verbose = getOption("seqR_verbose_default")
)
```
Arguments

- **sequences**: input sequences of one of two supported types, either string vector or list of string vectors.
- **k**: an integer representing the length of a k-mer.
- **kmer_alphabet**: a string vector representing the elements that should be used during the construction of k-mers. By default, all elements that are present in sequences are taking into account.
- **positional**: a single logical value that determines whether positional k-mer variant should be considered.
- **kmer_gaps**: an integer vector representing the lengths of gaps between consecutive k-mer elements. The length of the vector should be equal to k - 1.
- **with_kmer_counts**: a single logical value that determines whether the result should contain k-mer frequencies.
- **with_kmer_names**: a single logical value that determines whether the result should contain human-readable k-mer names.
- **batch_size**: a single integer value that represents the number of sequences that are being processed in a single step.
- **hash_dim**: a single integer value (1 <= hash_dim <= 500) representing the length of a hash vector that is internally used in the algorithm.
- **verbose**: a single logical value that denotes whether a user wants to get extra information on the current state of computations.

Details

The comprehensive description of supported features is available in vignette("features-overview", package = "seqR").

Value

A Matrix value that represents a result k-mer matrix. The result is a sparse matrix in order to reduce memory consumption. The i-th row of the matrix represents k-mers found in the i-th input sequence. Each column represents a distinct k-mer. The names of columns conform to human-readable schema for k-mers, if parameter with_kmer_names = TRUE.

See Also

Function that counts many k-mer variants in the single invocation: `count_multimers`

Function that merges several k-mer matrices (rbind): `rbind_columnwise`

Examples

```r
batch_size <- 1
```
Count k-mers of various types for a given collection of sequences

Description

This is a wrapper over `count_kmers` function in order to enable the computation of many types of k-mers in a single invocation of the function.

A user can input multiple k-mer configurations in the following way. Each parameter that is related to the configuration (i.e., `k_vector`, `positional_vector`, and `kmer_gaps_list`) is represented in a sequential form (i.e., a list or a vector). The i-th entry of each sequence corresponds to the i-th configuration.
count_multimers

Usage

```r
count_multimers(
  sequences,
  k_vector,
  kmer_alphabet = getOption("seqR_kmer_alphabet_default"),
  positional_vector = rep(getOption("seqR_positional_default"), length(k_vector)),
  kmer_gaps_list = rep(list(c()), length(k_vector)),
  with_kmer_counts = getOption("seqR_with_kmer_counts_default"),
  with_kmer_names = getOption("seqR_with_kmer_names_default"),
  batch_size = getOption("seqR_batch_size_default"),
  hash_dim = getOption("seqR_hash_dim_default"),
  verbose = getOption("seqR_verbose_default")
)
```

Arguments

- **sequences**: input sequences of one of two supported types, either string vector or list of string vectors
- **k_vector**: an integer vector that represents the lengths of k-mers. The i-th element corresponds to the value of \( k \) for the i-th k-mer configuration.
- **kmer_alphabet**: a string vector representing the elements that should be used during the construction of k-mers. By default, all elements that are present in sequences are taking into account
- **positional_vector**: a logical vector that consists of k-mer configurations related to the positional part. The i-th element corresponds to the i-th k-mer configuration (i.e., whether the k-mer is positional or not)
- **kmer_gaps_list**: a list of integer vectors that represents the lengths of k-mer gaps for each configuration separately. The i-th element of the list corresponds to the lengths of gaps of the i-th k-mer configuration
- **with_kmer_counts**: a single logical value that determines whether the result should contain k-mer frequencies
- **with_kmer_names**: a single logical value that determines whether the result should contain human-readable k-mer names
- **batch_size**: a single integer value that represents the number of sequences that are being processed in a single step
- **hash_dim**: a single integer value (1 \( \leq \) hash_dim \( \leq \) 500) representing the length of a hash vector that is internally used in the algorithm
- **verbose**: a single logical value that denotes whether a user wants to get extra information on the current state of computations

Details

The comprehensive description of supported features is available in vignette("features-overview", package = "seqR").
Value

A Matrix value that represents a result k-mer matrix. The result is a sparse matrix in order to reduce memory consumption. The i-th row of the matrix represents k-mers found in the i-th input sequence. Each column represents a distinct k-mer. The names of columns conform to human-readable schema for k-mers, if parameter with_kmer_names = TRUE.

See Also

Function that count k-mers of one type: count_kmers
Function that merges several k-mer matrices (rbind): rbind_columnwise

Examples

```r
batch_size <- 1

# Counting 1-mers
count_multimers(  
c("AAAACFVV", "AAAAAA", "AAAAD"),  
k_vector = c(1),  
batch_size=batch_size)

# Counting 1-mers and 2-mers
count_multimers(  
c("AAAACFVV", "AAAAAA", "AAAAD"),  
k_vector = c(1, 2),  
batch_size=batch_size)

# Counting 1-mers, 2-mers, and gapped 2-mers with the length of the gap = 1
count_multimers(  
c("AAAACFVV", "AAAAAA", "AAAAD"),  
k_vector = c(1, 2, 2),  
kmer_gaps = list(NULL, NULL, c(1)),  
batch_size=batch_size)

# Counting 3-mers, positional 3-mers, and positional gapped 2-mers with the length of the gap = 1
count_multimers(  
c("AAAACFVV", "AAAAAA", "AAAAD"),  
k_vector = c(3, 3, 2),  
kmer_gaps_list = list(NULL, NULL, c(1)),  
positionai_vector = c(FALSE, TRUE, TRUE),  
batch_size=batch_size)
```

CsgA data set

Description

5 reviewed sequences of CsgA proteins obtained from UniProt on 13-09-2021.
Usage
CsgA

Format
a list of 5 proteins.

---

rbind_columnwise  Bind rows of several k-mer matrices

Description
The function binds rows of several input k-mer matrices (of type Matrix), which are results of `count_kmers` and `count_multimers`. This implementation also handles properly k-mer matrices that do not have the same columns, as opposed to the implementation of `rbind`.

Usage
rbind_columnwise(...)

Arguments
...

k-mer matrices of type Matrix

Value
a k-mer matrix of type Matrix that is the result of the rbind operation

See Also
Function that count k-mers of one type: `count_kmers`
Function that counts many k-mer variants in the single invocation: `count_multimers`

Examples

```r
batch_size <- 1
# k-mer counting
resA <- count_kmers(c("AAAA", "ASASSSSASSA"), k=5, batch_size=batch_size)
resB <- count_multimers(c("HWHSHS", "AASDCASD"), k_vector=c(3, 5), batch_size=batch_size)

# rbind
res <- rbind_columnwise(resA, resB)
```
Index

* datasets
  CsgA, 7
  count_kmers, 3, 5, 7, 8
  count_multimers, 4, 5, 8
  CsgA, 7
  Matrix, 4, 7, 8
  rbind, 8
  rbind_columnwise, 4, 7, 8
  seqR (seqR-package), 2
  seqR-package, 2