Package: QuickBLAST (via r-universe)

July 21, 2026

Type Package

Title High-Performance Sequence Alignment using BLAST and Apache Arrow

Version 1.99.8

Date 2026-07-21

Maintainer Vishvesh Karthik <vishveshkarthik@gmail.com>

Description A BLAST Wrapper to call BLAST directly from R (interoperable and without Sys.call()) coupled with Apache Arrow's columnar format with parquet, IPC and csv file support.

License GPL (>= 2)

Depends R (>= 4.4.0)

Imports Rcpp (>= 1.0.10), RcppProgress (>= 0.4.2), dplyr (>= 1.1.4), tidyr (>= 1.3.1), tzdb (>= 0.5.0), iterators (>= 1.0.14), arrow (>= 21.0.0.1), tools (>= 4.3.1), compiler (>= 4.5.2), parallel (>= 4.4.1), fs (>= 1.6.6), RcppThread (>= 2.3.0), future (>= 1.69.0), future.apply (>= 1.20.2), future.callr (>= 0.10.2), xfun (>= 0.56), BH (>= 1.90.0), magrittr (>= 2.0.5)

LinkingTo Rcpp, RcppProgress, RcppThread, BH

biocViews Software, Alignment, SequenceMatching, Sequencing, DataImport

Encoding UTF-8

StagedInstall no

UseLTO yes

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Language en-US

Suggests curl (>= 7.0.0), formatR (>= 1.14), knitr, rmarkdown, remotes

VignetteBuilder knitr

Collate 'QuickBLAST-package.R' 'RcppExports.R' 'QuickBLAST.R'

SystemRequirements cmake (>= 3.25), sqlite3 (deb: libsqlite3-dev, rpm: sqlite-devel), eigen (deb: libeigen3-dev, rpm: eigen3-devel), boost (deb: libboost-dev, rpm: boost-devel), fontconfig, libcurl, harfbuzz, fribidi, freetype2, libpng, libtiff, libjpeg, libuv (brew: libuv, deb: libuv1-dev, rpm: libuv-devel), nghttp2 (brew: nghttp2, deb: libnghttp2-dev, rpm: libnghttp2-devel), libunwind-dev (deb), libcurl4-openssl-dev (deb), pkg-config

URL https://github.com/vizkidd/QuickBLAST

BugReports https://github.com/vizkidd/QuickBLAST/issues

Config/pak/sysreqs libboost-all-dev cmake libeigen3-dev libfontconfig1-dev libfreetype6-dev libfribidi-dev make libharfbuzz-dev libicu-dev libjpeg-dev libpng-dev libtiff-dev libuv1-dev libssl-dev pkg-config libsqlite3-dev

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

Date/Publication 2026-07-21 17:46:06 UTC

RemoteUrl https://github.com/BiocStaging/QuickBLAST

RemoteRef HEAD

RemoteSha 4efe6de52220eb5488add8d54e8c1f89e0b8b9ed

Contents

`.GetBinPath` *Get file path of file inside "bin" folder of package*

Description

Get file path of file inside "bin" folder of package

Usage

```
.GetBinPath(file_name = "")
```

Arguments

`file_name`	file_name inside "bin" folder

Value

File path inside QuickBLAST package

`.GetLibsPath` *Globals Get file path of file inside "libs" folder of package*

Description

Globals Get file path of file inside "libs" folder of package

Usage

```
.GetLibsPath(file_name = "")
```

Arguments

`file_name`	file_name inside "libs" folder

Value

File path inside QuickBLAST package

all2all *Execute all2all QuickBLAST*

Description

Executes All-to-All QuickBLAST between two lists of organisms/genes/clusters. Output BLAST files are bi-directional and are stored in the filename filename1.filename2.all2all under output_dir. (All-to-All is simply Many-to-Many association)

Usage

```
all2all(
  first_list,
  second_list,
  blast_fun,
  seq_type,
  strand,
  blast_program,
  file_ext = ".fa",
  input_prefix_path = NULL,
  output_dir = "./",
  ...
)
```

Arguments

first_list	Vector of FASTA Filenames or Strings
second_list	Vector of FASTA Filenames or Strings
blast_fun	One of QuickBLAST::BLAST2Seqs, QuickBLAST::BLAST2Files, QuickBLAST::BLAST2Folders, QuickBLAST::BLAST2DBs
seq_type	(int) Sequence Type. Check QuickBLAST::GetQuickBLASTEnums()$ESeqType for available enums.
strand	(int) Strand. Check QuickBLAST::GetQuickBLASTEnums()$EStrand for available enums.
blast_program	Give the name of the BLAST program to use (if in $PATH) or give the absolute path to the BLAST program.
file_ext	File extension of input files. eg- ".cds" or ".fa", Unused if input_type is GetQuickBLASTEnums()$EInputType$eSequencesString
input_prefix_path	If input lists/vectors are filenames, then provide input folder to prefix path
output_dir	Path to BLAST output
...	Extended options passed to internal functions, including: • blast_options: BLAST Options to use - QuickBLAST::GetAvailableBLASTOptions() • save_sequences: (bool) Save full sequences to result?

- save_hsp_sequences: (bool) Save HSP sequences to result?
- return_values: (bool) Return values back to R?
- min_batch_size: Minimum batch size. (Default: 256)
- n_threads: Number of threads. (Default: 8)
- out_format: Output format. ipc/csv/parquet (Default: "parquet")
- extension: File extension. (Only for QuickBLAST::BLAST2Folders())
- reciprocal_hits: (bool) Reciprocal (Bi-directional) Hits?
- verbose: (bool) Print DEBUG Messages?

Value

If return_values = TRUE, returns a list of data frames corresponding to each alignment query. Otherwise, returns invisible(NULL) or outputs directly to files.

See Also

GetAvailableBLASTOptions(), GetQuickBLASTEnums()

Examples

```
QuickBLAST::all2all(
  first_list = fs::path_package("QuickBLAST", "extdata", "protein_query.fasta"),
  second_list = fs::path_package("QuickBLAST", "extdata", "protein_subject.fasta"),
  blast_fun = QuickBLAST::BLAST2Files,
  seq_type = 0,
  strand = 0,
  output_dir = "./",
  n_threads = 8,
  blast_program = "tblastx",
  save_sequences = FALSE,
  save_hsp_sequences = FALSE,
  return_values = TRUE,
  min_batch_size = 256,
  out_format = "parquet",
  blast_options = "",
  verbose = TRUE
)
```

BLAST1Folder *BLAST Files within a Folder*

Description

BLAST FASTA files containing nucleotide or protein sequences, within a folder with a QuickBLAST instance. The files from the folder are selected with the extension parameter and BLAST'd against each other.

Usage

```
BLAST1Folder(
  ptr,
  input_folder,
  extension,
  out_folder,
  out_format = NULL,
  num_threads = 0L,
  reciprocal_hits = FALSE,
  min_batch_size = 0L,
  verbose = TRUE
)
```

Arguments

ptr	(Rcpp::XPtr<QuickBLAST>) or (unsigned int) Pointer/ID of QuickBLAST instance
input_folder	(string) Input folder
extension	(string) Extension of files in folder.
out_folder	(string) Ouput Folder (Required).
out_format	(string) Ouput Format. 'ipc'/'csv'/'parquet' (Optional) (Default: 'parquet').
num_threads	(unsigned int) Number of threads. (Optional)
reciprocal_hits	(bool) Perform Bi-directional (Reciprocal => query <-> subject) BLAST? (Default: FALSE) (Optional)
min_batch_size	(unsigned int) Minimum batch size - Size of file write buffer (Optional).
verbose	(bool) Verbosity (Defulat: TRUE).

Value

(bool) TRUE - on success, FALSE - Otherwise. (Results are not returned as R Lists to reduce overhead)

Note

Only FASTA files are supported by this function, use BLAST2DBs() if inputs are BLAST DBs.

See Also

GetInstanceID(), GetQuickBLASTInstance(), BLAST2Files(), BLAST2Seqs(), BLAST2Folders(), BLAST1Folder(), RemoteBLAST()

BLAST2DBs *BLAST 2 on-disk BLAST DBs*

Description

Calls makeblastdb to create a BLAST DB of a FASTA file

Usage

```
BLAST2DBs(
  ptr,
  query,
  subject,
  out_file = NULL,
  out_format = NULL,
  num_threads = 0L,
  refresh_db = FALSE,
  return_values = TRUE,
  min_batch_size = 0L,
  enable_chunking = FALSE,
  chunk_size = 50000L,
  overlap = 1000L,
  verbose = TRUE
)
```

Arguments

ptr	(Rcpp::XPtr<QuickBLAST>) or (unsigned int) Pointer/ID of QuickBLAST instance
query	(string) Query DB
subject	(string) Subject DB
out_file	(string) Ouput file (Optional)
out_format	(string) Ouput Format. 'ipc'/'csv'/'parquet' (Optional) (Default: 'parquet').
num_threads	(unsigned int) Number of threads. (Optional)
refresh_db	(bool) If TRUE, re-creates the DBs
return_values	(bool) Return BLAST Hits as Rcpp::List (Default: TRUE) (Optional)
min_batch_size	(unsigned int) Minimum batch size - Size of file write buffer (Optional).
enable_chunking	(bool) Chunk large sequences? (Default: FALSE)
chunk_size	(int) Size of chunks (Default: 50000)
overlap	(int) Overlap between chunks (Default: 1000)
verbose	(bool) Verbose? (Default: TRUE)

Value

(SEXP) Rcpp::List - if return_values == TRUE, out_file - Otherwise.

See Also

GetInstanceID(), GetQuickBLASTInstance(), MakeBLASTDB(), isBLASTDB()

Examples

```
## Not run:
blastp_inst <- QuickBLAST::CreateQuickBLASTInstance(
  seq_type = 1,
  strand = 0,
  program = "blastp",
  save_sequences = F,
  save_hsp_sequences = F
)
QuickBLAST::BLAST2DBs(
  ptr=blastp_inst,
  query=system.file(
    "extdata",
    "protein_query.fasta",
    package = "QuickBLAST",
    mustWork = T
  ),
  subject=system.file(
    "extdata",
    "protein_subject.fasta",
    package = "QuickBLAST",
    mustWork = T
  ),
  num_threads=24,
  out_file="test.db.arrow",
  return_values = T
)
QuickBLAST::MakeBLASTDB(
  blastp_inst,
  system.file(
    "extdata",
    "protein_query.fasta",
    package = "QuickBLAST",
    mustWork = T
  ),
  "protein_query.db"
)
QuickBLAST::MakeBLASTDB(
  blastp_inst,
  system.file(
    "extdata",
    "protein_subject.fasta",
    package = "QuickBLAST",
    mustWork = T
```

```
  ),
  "protein_subject.db"
)
QuickBLAST::BLAST2DBs(
  ptr=blastp_inst,
  query="protein_query.db",
  subject="protein_subject.db",
  num_threads=24,
  out_file="test.db.arrow",
  return_values = T
)

## End(Not run)
```

BLAST2Files *BLAST 2 Files*

Description

BLAST 2 FASTA files containing nucleotide or protein sequences with a QuickBLAST instance.

Usage

```
BLAST2Files(
  ptr,
  query,
  subject,
  out_file = NULL,
  out_format = NULL,
  num_threads = 0L,
  return_values = TRUE,
  min_batch_size = 0L,
  verbose = TRUE
)
```

Arguments

ptr	(Rcpp::XPtr<QuickBLAST>) or (unsigned int) Pointer/ID of QuickBLAST instance
query	(string) Query file
subject	(string) Subject file
out_file	(string) Ouput file (Optional)
out_format	(string) Ouput Format. 'ipc'/'csv'/'parquet' (Optional) (Default: 'parquet').
num_threads	(unsigned int) Number of threads. (Optional)
return_values	(bool) Return BLAST Hits as Rcpp::List (Default: TRUE) (Optional)
min_batch_size	(unsigned int) Minimum batch size - Size of file write buffer (Optional).
verbose	(bool) Verbosity (Default: TRUE).

Value

(SEXP) Rcpp::List - if return_values == TRUE, out_file - Otherwise.

Note

Only FASTA files are supported by this function, use BLAST2DBs() if inputs are BLAST DBs.

See Also

GetInstanceID(), GetQuickBLASTInstance(), BLAST2Files(), BLAST2DBs(), BLAST2Seqs(), BLAST2Folders(), BLAST1Folder(), RemoteBLAST()

Examples

```
## Not run:
blastp_inst <- QuickBLAST::CreateQuickBLASTInstance(
  seq_type = 1,
  strand = 0,
  program = "blastp",
  save_sequences = F,
  save_hsp_sequences = F
)
QuickBLAST::BLAST2Files(
  ptr = blastp_inst,
  query = system.file(
    "extdata",
    "protein_query.fasta",
    package = "QuickBLAST",
    mustWork = T
  ),
  subject = system.file(
    "extdata",
    "protein_subject.fasta",
    package = "QuickBLAST",
    mustWork = T
  ),
  out_file = "test.arrow",
  out_format = "parquet",
  return_values = F,
  min_batch_size = 1024
)
QuickBLAST::BLAST2Files(
  ptr = blastp_inst,
  query = system.file(
    "extdata",
    "protein_query.fasta",
    package = "QuickBLAST",
    mustWork = T
  ),
  subject = system.file(
    "extdata",
    "protein_subject.fasta",
```

```
    package = "QuickBLAST",
    mustWork = T
  ),
  out_file = "test.arrow",
  return_values = T,
  min_batch_size = 0,
  seq_limit = 0
)

## End(Not run)
```

`BLAST2Folders` *BLAST 2 Folders*

Description

BLAST 2 Folders with FASTA files containing nucleotide or protein sequences with a QuickBLAST instance. The files from query and subject folders are selected with the extension parameter

Usage

```
BLAST2Folders(
  ptr,
  query,
  subject,
  extension,
  out_folder,
  out_format = NULL,
  num_threads = 0L,
  reciprocal_hits = FALSE,
  min_batch_size = 0L,
  verbose = TRUE
)
```

Arguments

`ptr`	(`Rcpp::XPtr<QuickBLAST>`) or (unsigned int) Pointer/ID of QuickBLAST instance
`query`	(string) Query folder
`subject`	(string) Subject folder.
`extension`	(string) Extension of files in folder.
`out_folder`	(string) Ouput Folder (Required).
`out_format`	(string) Ouput Format. 'ipc'/'csv'/'parquet' (Optional) (Default: 'parquet').
`num_threads`	(unsigned int) Number of threads. (Optional)
`reciprocal_hits`	(bool) Perform Bi-directional (Reciprocal => query <-> subject) BLAST? (Default: FALSE) (Optional)

min_batch_size	(unsigned int) Minimum batch size - Size of file write buffer (Optional).
verbose	(bool) Verbosity (Defaut: TRUE).

Value

(bool) TRUE - on success, FALSE - Otherwise. (Results are not returned as R Lists to reduce overhead)

Note

Only FASTA files are supported by this function, use BLAST2DBs() if inputs are BLAST DBs.

See Also

GetInstanceID(), GetQuickBLASTInstance(), BLAST2Files(), BLAST2Seqs(), BLAST2Folders(), BLAST1Folder(), RemoteBLAST()

BLAST2Seqs *BLAST 2 Sequence strings*

Description

BLAST 2 nucleotide or protein strings with a QuickBLAST instance.

Usage

```
BLAST2Seqs(ptr, query, subject, verbose = TRUE)
```

Arguments

ptr	(Rcpp::XPtr<QuickBLAST>) or (unsigned int) Pointer/ID of QuickBLAST instance
query	(string) Query sequence
subject	(string) Subject sequence.
verbose	(bool) Verbosity (Default: TRUE).

Value

(Rcpp::XPtr<QuickBLAST>) Pointer to a QuickBLAST Instance (Cannot be used in R)

See Also

GetInstanceID(), GetQuickBLASTInstance(), BLAST2Files(), BLAST2Seqs(), BLAST2Folders(), BLAST1Folder(), RemoteBLAST()

Examples

```
## Not run:
blastp_inst <- QuickBLAST::CreateQuickBLASTInstance(
  seq_type = 1,
  strand = 0,
  program = "blastp",
  save_sequences = F,
  save_hsp_sequences = F
)
QuickBLAST::BLAST2Seqs(
  blastp_inst,
  "MQILLVEDDNTLFQELKKELEQWDFNV
  AGIEDFGKVMDTFESFNPEIVILDVQLP
  KYDGFYWCRKMREVSNVPILFLSSRDNP
  MDQVMSMELGADDYMQKPFYTNVLIAKL
  QAIYRRVYEFTAEEKRTLTWQDAVVDLS
  KDSIQKGDDTIFLSKTEMIILEILITKK
  NQIVSRDTIITALWDDEAFVSDNTLTVN
  VNRLRKKLSEISMDSAIETKVGKGYMAHE",
  "MQILLVEDDNTLFQELKKELEQWDFNV
  AGIEDFGKVMDTFESFNPEIVILDVQLP
  KYDGFYWCRKMREVSNVPILFLSSRDNP
  MDQVMSMELGADDYMQKPFYTNVLIAKL
  QAIYRRVYEFTAEEKRTLTWQDAVVDLS
  KDSIQKGDDTIFLSKTEMIILEILITKK
  NQIVSRDTIITALWDDEAFVSDNTLTVN
  VNRLRKKLSEISMDSAIETKVGKGYMAHE"
)

## End(Not run)
```

BLASTFile2DB *BLAST File to a on-disk BLAST DB*

Description

Runs BLAST using the ptr

Usage

```
BLASTFile2DB(
  ptr,
  query,
  subject,
  out_file = NULL,
  out_format = NULL,
  num_threads = 0L,
  return_values = TRUE,
  min_batch_size = 0L
)
```

Arguments

ptr	(Rcpp::XPtr<QuickBLAST>) or (unsigned int) Pointer/ID of QuickBLAST instance
query	(string) Query DB
subject	(string) Subject DB
out_file	(string) Ouput file (Optional)
out_format	(string) Ouput Format. 'ipc'/'csv'/'parquet' (Optional) (Default: 'parquet').
num_threads	(unsigned int) Number of threads. (Optional)
return_values	(bool) Return BLAST Hits as Rcpp::List (Default: TRUE) (Optional)
min_batch_size	(unsigned int) Minimum batch size - Size of file write buffer (Optional).

Value

(SEXP) Rcpp::List - if return_values == TRUE, out_file - Otherwise.

Note

Calls makeblastdb to create a BLAST DB of subject if it is not a DB

See Also

GetInstanceID(), GetQuickBLASTInstance(), MakeBLASTDB(), isBLASTDB()

Examples

```
## Not run:
blastp_inst <- QuickBLAST::CreateQuickBLASTInstance(
  seq_type = 1,
  strand = 0,
  program = "blastp",
  save_sequences = F,
  save_hsp_sequences = F
)
QuickBLAST::BLASTFile2DB(
  ptr=blastp_inst,
  query=system.file(
    "extdata","protein_query.fasta",
    package = "QuickBLAST",
    mustWork = T
  ),
  subject=system.file(
    "extdata",
    "protein_subject.fasta",
    package = "QuickBLAST",
    mustWork = T
  ),
  num_threads=24,
  out_file="test.db.arrow",
  return_values = T
```

```
)
QuickBLAST::MakeBLASTDB(
  blastp_inst,
  system.file(
    "extdata",
    "protein_subject.fasta",
    package = "QuickBLAST",
    mustWork = T
  ),
  "protein_subject.db"
)
QuickBLAST::BLASTFile2DB(
  ptr=blastp_inst,
  query="protein_query.fasta",
  subject="protein_subject.db",
  num_threads=24,
  out_file="test.db.arrow",
  return_values = T
)

## End(Not run)
```

`CreateQuickBLASTInstance` *Create new QuickBLAST instance with seq_type, strand, program and BLAST options.*

Description

Create a new QuickBLAST C++ object with seq_type, strand, program and BLAST options, which can be used in QuickBLAST::BLAST2Files() and QuickBLAST::BLAST2Seqs()

Usage

```
CreateQuickBLASTInstance(
  seq_type,
  strand,
  program,
  options = NULL,
  save_sequences = FALSE,
  save_hsp_sequences = FALSE
)
```

Arguments

`seq_type`	(int) 0 - (eNucleotide) (OR) 1 - (eProtein)
`strand`	(int) 0 - (ePlus) (OR) 1 - (eMinus)
`program`	(string) Name of the BLAST program

options	(string (or) Named List) List of BLAST options - check QuickBLAST::GetAvailableBLASTOptions(). String should be of the format "-option1 value1 -option2 value2". If empty, default values (per program) are used.
save_sequences	(bool) Save Full Sequences to output?. (Default: FALSE)
save_hsp_sequences	(bool) Save HSP Sequences to output?. (Default: FALSE)

Value

(Rcpp::XPtr<QuickBLAST>) Pointer to a QuickBLAST Instance (Cannot be used in R)

Note

Set save_sequences AND/OR save_hsp_sequences when using Genomes

See Also

GetAvailableBLASTOptions(), GetQuickBLASTEnums(), BLAST2Files(), BLAST2Seqs(), BLAST2Folders(), BLAST1Folder(), RemoteBLAST()

Examples

```
blastp_inst <- QuickBLAST::CreateQuickBLASTInstance(
  seq_type = 1,
  strand = 0,
  program = "blastp",
  save_sequences = F,
  save_hsp_sequences = F
)
```

DeleteQuickBLASTInstance

Delete a QuickBLAST instance stored in C++ side

Description

This function deletes a QuickBLAST instance based on the instance ID

Usage

```
DeleteQuickBLASTInstance(ptr)
```

Arguments

ptr	(Rcpp::XPtr<QuickBLAST>) or (unsigned int) Pointer/ID of QuickBLAST instance

Value

TRUE - if the instance is deleted successfully, throws error otherwise

Examples

```
## Not run:
blastp_inst <- QuickBLAST::CreateQuickBLASTInstance(
  seq_type = 1,
  strand = 0,
  program = "blastp",
  save_sequences = F,
  save_hsp_sequences = F
)
QuickBLAST::DeleteQuickBLASTInstance(
  QuickBLAST::GetInstanceID(
    blastp_inst
  )
)

## End(Not run)
```

GetAvailableBLASTOptions

Get a List of Available BLAST options

Description

Use this function in blast_options to set BLAST defaults for the chosen BLAST program.

Usage

```
GetAvailableBLASTOptions()
```

Value

A List of Available BLAST options

Note

CREATE a NEW LIST with ONLY the OPTIONS THAT YOU NEED

Examples

```
opts <- QuickBLAST::GetAvailableBLASTOptions()
print(opts)
```

GetFASTAHeaders *Get FASTA Headers*

Description

Get the header strings of a FASTA file as a String Vector

Usage

```
GetFASTAHeaders(path, keep_gt = FALSE)
```

Arguments

path	(std::string) Path to FASTA file
keep_gt	(bool) Keep the '>' symbol? (Default: FALSE)

Value

(SEXP) Rcpp::StringVector - on success, FALSE - Otherwise.

See Also

MakeBLASTDB(), isBLASTDB()

Examples

```
## Not run:
QuickBLAST::GetFASTAHeaders(
  system.file(
    "extdata",
    "protein_query.fasta",
    package = "QuickBLAST",
    mustWork = T
  ),
  keep_gt = F
)

## End(Not run)
```

GetInstanceCount *Get count of QuickBLAST instances stored in C++ side*

Description

This function gives the size of the list of QuickBLAST C++ object list

Usage

```
GetInstanceCount()
```

Value

Count of QuickBLAST instances

Examples

```
QuickBLAST::GetInstanceCount()
```

GetInstanceID *Get ID/Index of a QuickBLAST instance stored in C++ side*

Description

This function fetches the ID/Index of a QuickBLAST instance of a `Rcpp::XPtr<QuickBLAST>` stored in C++ side.

Usage

```
GetInstanceID(ptr)
```

Arguments

`ptr`	(`Rcpp::XPtr<QuickBLAST>`) or (unsigned int) Pointer/ID of QuickBLAST instance

Value

(unsigned int) ID/Index of the QuickBLAST instance pointer, FALSE otherwise

Examples

```
blastp_inst <- QuickBLAST::CreateQuickBLASTInstance(
  seq_type = 1,
  strand = 0,
  program = "blastp",
  save_sequences = F,
  save_hsp_sequences = F,
  num_threads=24
)
QuickBLAST::GetInstanceID(
  blastp_inst
)
```

GetQuickBLASTEnums *Get a list of Enums used by QuickBLAST*

Description

Get a list of Enums used by QuickBLAST

Usage

```
GetQuickBLASTEnums()
```

Value

A List of Enums used by QuickBLAST

Examples

```
enums <- QuickBLAST::GetQuickBLASTEnums()
print(names(enums))
print(enums$ESeqType$eNucleotide)
```

GetQuickBLASTInstance *Get QuickBLAST instance stored in C++ side at ID/Index*

Description

This function fetches the QuickBLAST instance of a `Rcpp::XPtr<QuickBLAST>` at ID/Index stored in C++ side.

Usage

```
GetQuickBLASTInstance(ptr_id)
```

Arguments

ptr_id	(unsigned int) ID/Index of Pointer to a QuickBLAST instance (in C++ side).

Value

(Rcpp::XPtr<QuickBLAST>) Pointer to a QuickBLAST instance, FALSE otherwise

Examples

```
# QuickBLAST::GetQuickBLASTInstance(0)
```

GetQuickBLASTOptions *Get BLAST options for a QuickBLAST instance as a string.*

Description

Get the BLAST options for a QuickBLAST instance.

Usage

```
GetQuickBLASTOptions(ptr)
```

Arguments

ptr	(Rcpp::XPtr<QuickBLAST>) or (unsigned int) Pointer/ID of QuickBLAST instance

Value

(string) BLAST options as std::string

See Also

GetAvailableBLASTOptions(), GetQuickBLASTEnums(), SetQuickBLASTOptions()

Examples

```
## Not run:
blastp_inst <- QuickBLAST::CreateQuickBLASTInstance(
  seq_type = 1,
  strand = 0,
  program = "tblastn",
  save_sequences = F,
  save_hsp_sequences = F
)
QuickBLAST::GetQuickBLASTOptions(
  blastp_inst
)
```

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

`isBLASTDB` *Check BLAST DB Files*

Description

Check whether a path/name corresponds to a BLAST DB (heuristic). Check if all the files of a BLAST DB exist (.psq .pog .pin .phr .pos .pto .pot .pdb .ptf .pjs). This checks the directory for filenames that start with the provided basename and have extensions commonly produced by makeblastdb:

- protein: .phr .pin .psq
- nucleotide: .nhr .nin .nsq It returns a list with a boolean `is_db`, a guessed `type` ("Protein"/"Nucleotide"/"Mixed"/"Unknown"), a character vector of matching `files`, and the `dir` and `name` used.

Usage

```
isBLASTDB(ptr, input_db)
```

Arguments

`ptr`	(`Rcpp::XPtr<QuickBLAST>`) or (unsigned int) Pointer/ID of QuickBLAST instance
`input_db`	character(1) path to db (path + name) or a bare name (current directory assumed)

Value

list with keys: is_db (logical), type (string), files (character vector), dir (string), name (string), message (string)

See Also

`GetInstanceID()`, `GetQuickBLASTInstance()`, `MakeBLASTDB()`, `BLAST2DBs()`

Examples

```
## Not run:
QuickBLAST::MakeBLASTDB(
  blastp_inst,
  system.file(
    "extdata",
    "protein_query.fasta",
    package = "QuickBLAST",
    mustWork = T
  ),
  "protein_query.db"
```

```
)
QuickBLAST::isBLASTDB(
  tools::file_path_sans_ext(
    system.file(
      "extdata",
      "protein_query.db.pin",
      package = "QuickBLAST",
      mustWork = T
    )
  )
)

## End(Not run)
```

`isQuickBLASTLoaded` *Check R <-> C++ (FFI) connection*

Description

This function does nothing than check the connection between the R package and C++ libraries

Usage

```
isQuickBLASTLoaded()
```

Value

String that successfully confirms when the package is loaded properly

Examples

```
QuickBLAST::isQuickBLASTLoaded()
```

`LoadBLASTHits` *Load BLAST Hits into a data.frame*

Description

Give the path to a BLAST Hits file (table/ipc/parquet) to load it into a tibble (BLAST HITs Table).

Usage

```
LoadBLASTHits(infile, sep = "\t", header = F, format = "parquet")
```

Arguments

infile	BLAST hits filename (not a connection) (Gzipped files supported)
sep	Delimiter of the BLAST File columns. (Only applies when format == 'table'). Default - '\t'
header	Does the file have a header? (Only applies when format == 'table'). Default - FALSE
format	Input Format (Required) - 'table' (CSV/TSV)/'ipc' (arrow::ipc)/'parquet' (arrow::parquet) - Default : 'parquet'

Value

Data Frame with BLAST Results

Examples

```
# Assuming 'blast_results.parquet' exists in your working directory
# results <- QuickBLAST::LoadBLASTHits("blast_results.parquet", format = "parquet")
```

MakeBLASTDB *Make on-disk BLAST DB from a FASTA file*

Description

Calls makeblastdb to create a BLAST DB of a FASTA file

Usage

```
MakeBLASTDB(ptr, input_file, database_name, parse_seqids = FALSE)
```

Arguments

ptr	(Rcpp::XPtr<QuickBLAST>) or (unsigned int) Pointer/ID of QuickBLAST instance
input_file	(string) Path to FASTA file.
database_name	(string) Name of the output DB.
parse_seqids	(bool) TRUE - Checks FASTA headers for malformations (Default: FALSE)

Value

(bool) DB name on success, FALSE - Otherwise.

Note

Faitful re-implementation of makeblastdb seemed pointless, hence the system.call() to a the program.

See Also

GetInstanceID(), GetQuickBLASTInstance(), BLAST2DBs(), isBLASTDB()

Examples

```
## Not run:
blastp_inst <- QuickBLAST::CreateQuickBLASTInstance(
  seq_type = 1,
  strand = 0,
  program = "blastp",
  save_sequences = F,
  save_hsp_sequences = F
)
QuickBLAST::MakeBLASTDB(
  blastp_inst,
  system.file(
    "extdata",
    "protein_query.fasta",
    package = "QuickBLAST",
    mustWork = T
  ),
  "protein_query.db"
)
QuickBLAST::MakeBLASTDB(
  blastp_inst,
  system.file(
    "extdata",
    "protein_subject.fasta",
    package = "QuickBLAST",
    mustWork = T
  ),
  "protein_subject.db"
)
QuickBLAST::BLAST2DBs(
  ptr=blastp_inst,
  query="protein_query.db",
  subject="protein_subject.db",
  num_threads=24,
  out_file="test.db.arrow",
  return_values = T
)

## End(Not run)
```

---

one2one *Execute one2one QuickBLAST*

Description

Executes One-to-One QuickBLAST between two lists of organisms/genes/clusters. The BLAST Hits are stored in Arrow::Feather/Parquet format.

Usage

```
one2one(
  first_list,
  second_list,
  blast_fun,
  seq_type,
  strand,
  blast_program,
  file_ext = ".fa",
  input_prefix_path = NULL,
  output_dir = "./",
  ...
)
```

Arguments

first_list	Vector of FASTA Filenames or Strings
second_list	Vector of FASTA Filenames or Strings
blast_fun	One of QuickBLAST::BLAST2Seqs, QuickBLAST::BLAST2Files, QuickBLAST::BLAST2Folders, QuickBLAST::BLAST2DBs
seq_type	(int) Sequence Type. Check QuickBLAST::GetQuickBLASTEnums()$ESeqType for available enums.
strand	(int) Strand. Check QuickBLAST::GetQuickBLASTEnums()$EStrand for available enums.
blast_program	Give the name of the BLAST program to use (if in $PATH) or give the absolute path to the BLAST program.
file_ext	File extension of input files. eg- ".cds" or ".fa", Unused if input_type is GetQuickBLASTEnums()$EInputType$eSequencesString
input_prefix_path	If input lists/vectors are filenames, then provide input folder to prefix path
output_dir	Path to BLAST output
...	Extended options passed to internal functions, including: • blast_options: BLAST Options to use - QuickBLAST::GetAvailableBLASTOptions() • save_sequences: (bool) Save full sequences to result? • save_hsp_sequences: (bool) Save HSP sequences to result? • return_values: (bool) Return values back to R? • min_batch_size: Minimum batch size. (Default: 256) • n_threads: Number of threads. (Default: 8) • out_format: Output format. ipc/csv/parquet (Default: "parquet") • extension: File extension. (Only for QuickBLAST::BLAST2Folders()) • reciprocal_hits: (bool) Reciprocal (Bi-directional) Hits? • verbose: (bool) Print DEBUG Messages?

Value

If `return_values = TRUE`, returns a list of data frames corresponding to each alignment query. Otherwise, returns `invisible(NULL)` or outputs directly to files.

See Also

`GetAvailableBLASTOptions()`, `GetQuickBLASTEnums()`

Examples

```
QuickBLAST::one2one(
  first_list = fs::path_package("QuickBLAST", "extdata", "protein_query.fasta"),
  second_list = fs::path_package("QuickBLAST", "extdata", "protein_subject.fasta"),
  blast_fun = QuickBLAST::BLAST2Files,
  seq_type = 0,
  strand = 0,
  output_dir = "./",
  n_threads = 8,
  blast_program = "tblastx",
  save_sequences = FALSE,
  save_hsp_sequences = FALSE,
  return_values = FALSE,
  min_batch_size = 256,
  out_format = "parquet",
  blast_options = "",
  verbose = TRUE
)
```

RemoteBLAST *BLAST query against remote NCBI DBs*

Description

BLAST the input query against remote NCBI DBs (one sequence at a time - to respect rate limits)

Usage

```
RemoteBLAST(
  ptr,
  database,
  query_input,
  input_type,
  outFile = NULL,
  outFormat = NULL,
  return_values = TRUE,
  max_poll_seconds = 360L,
```

```
  poll_interval_ms = 4000L,
  verbose = TRUE
)
```

Arguments

ptr	(Rcpp::XPtr<QuickBLAST>) or (unsigned int) Pointer/ID of QuickBLAST instance
database	(string) Name of the remote NCBI DB - Check note for reference and supported values.
query_input	(Rcpp::List) (Named) List of input queries (Sequences, Files, Folders - type is determined by input_type parameter)
input_type	(QuickBLAST::EInputType) Input type (Check GetQuickBLASTEnums())
outFile	(string) Output file name (Optional)
outFormat	(string) Format of Output File (Required for outFile) (Default: 'parquet')
return_values	(bool) Return BLAST Hits as Rcpp::List (Default: TRUE) (Optional)
max_poll_seconds	(int) Max seconds to wait for RemoteBLAST (Default: 360) (Optional)
poll_interval_ms	(int) Milliseconds wait-time between polling RemoteBLAST service (Default(4s): 4000) (Optional)
verbose	(bool) Verbosity (Default: TRUE).

Value

(SEXP) Rcpp::List - if return_values == TRUE, outFile - Otherwise.

Note

Check BLAST Guide (https://blast.ncbi.nlm.nih.gov/BLAST_guide.pdf) and NCBI BLAST (https://blast.ncbi.nlm.nih.gov/Blast.cgi) (Program -> Choose DB/Search set) for database names.

See Also

GetInstanceID(), GetQuickBLASTInstance(), BLAST2Files(), BLAST2Seqs(), BLAST2Folders(), BLAST1Folder(), RemoteBLAST()

Examples

```
## Not run:
blastp_inst <- QuickBLAST::CreateQuickBLASTInstance(
  seq_type = 1,
  strand = 0,
  program = "blastp",
  save_sequences = F,
  save_hsp_sequences = F
)
```

```
QuickBLAST::RemoteBLAST(
  blastp_inst,
  query_input="MQILLVEDDNTLFQELKKELEQWDFN
  VAGIEDFGKVMDTFESFNPEIVILDVQLPKYDGFYWCRK
  MREVSNVPILFLSSRDNPMDQVMSMELGADDYMQKPFYT
  NVLIAKLQAIYRRVYEFTAEEKRTLTWQDAVVDLSKDSI
  QKGDDTIFLSKTEMIILEILITKKNQIVSRDTIITALWD
  DEAFVSDNTLTVNVNRLRKKLSEISMDSAIETKVGKGYMAHE",
  database= "pdb",
  input_type=1,
  return_values=T
)

## End(Not run)
```

`SetQuickBLASTOptions` *Set BLAST options for a QuickBLAST instance.*

Description

Set/Modify the BLAST options for a QuickBLAST instance.

Usage

```
SetQuickBLASTOptions(ptr, program_name, options, verbose = TRUE)
```

Arguments

`ptr`	(`Rcpp::XPtr<QuickBLAST>`) or (unsigned int) Pointer/ID of QuickBLAST instance
`program_name`	(string) Name of the BLAST program
`options`	(string (or) Named List) List of BLAST options - check QuickBLAST::GetAvailableBLASTOptions(). String should be of the format "-option1 value1 -option2 value2"
`verbose`	(bool) Verbose?

Value

(bool) TRUE - if options set for the QuickBLAST instance, FALSE otherwise

See Also

`GetAvailableBLASTOptions()`, `GetQuickBLASTEnums()`

Examples

```
## Not run:
blastp_inst <- QuickBLAST::CreateQuickBLASTInstance(
  seq_type = 1,
  strand = 0,
  program = "tblastn",
  save_sequences = F,
  save_hsp_sequences = F
)
QuickBLAST::SetQuickBLASTOptions(
  blastp_inst,
  "blastp",
  "-evalue 1"
)

## End(Not run)
```

Index