Package: rBLAST (via r-universe)

July 14, 2024

Title R Interface for the Basic Local Alignment Search Tool

Description Seamlessly interfaces the Basic Local Alignment Search Tool (BLAST) to search genetic sequence data bases. This work was partially supported by grant no. R21HG005912 from the National Human Genome Research Institute.

Version 1.1.1

Date 2024-04-30

biocViews Genetics, Sequencing, SequenceMatching, Alignment, DataImport

Depends Biostrings (>= 2.26.2)

Imports methods, utils, BiocFileCache

Suggests knitr, rmarkdown, testthat

URL https://github.com/mhahsler/rBLAST

BugReports https://github.com/mhahsler/rBLAST/issues

SystemRequirements ncbi-blast+

License GPL-3

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.3.1

Roxygen list(markdown = TRUE)

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

RemoteUrl https://github.com/bioc/rBLAST

RemoteRef HEAD

RemoteSha 39e28cec59647b0ff7afa24c7f0472eccfcefefe

Contents

`blast` *Basic Local Alignment Search Tool (BLAST)*

Description

Open a BLAST database and execute blastn (blastp or blastx) from blast+ to find sequences matches.

Usage

```
blast(db = NULL, remote = FALSE, type = "blastn")

blast_help(type = "blastn")

## S3 method for class 'BLAST'
print(x, info = TRUE, ...)

## S3 method for class 'BLAST'
predict(
  object,
  newdata,
  BLAST_args = "",
  custom_format = "",
  verbose = FALSE,
  keep_tmp = FALSE,
  ...
)

has_blast()
```

Arguments

`db`	the database file to be searched (without file extension).
`remote`	logical execute the query remotely on the NCBI server. `db` needs to be the name of a database available in the server.
`type`	BLAST program to use (e.g., `blastn`, `blastp`, `blastx`).
`info`	show additional data base information.
`...`	additional arguments are ignored.
`object`, `x`	An open BLAST database as a BLAST object created with `blast()`.
`newdata`	the query as an object of class Biostrings::XStringSet.
`BLAST_args`	additional arguments in command-line style.
`custom_format`	custom format specified by space delimited format specifiers.
`verbose`	logical; print progress and debugging information.
`keep_tmp`	logical; keep temporary files for debugging.

Value

- `blast()` returns a BLAST database object which can be used for queries (via `predict`).
- `predict` returns a data.frame containing the BLAST results.
- `has_blast()` returns `TRUE` if the blast software installation can be found and `FALSE` otherwise.

Installing BLAST+

The BLAST+ software needs to be installed on your system. Installation instructions are available in this package's INSTALL file and at https://www.ncbi.nlm.nih.gov/books/NBK569861/.

R needs to be able to find the executable. After installing the software, try in R

```
Sys.which("blastn")
```

If the command returns "" instead of the path to the executable, then you need to set the environment variable called PATH. In R

```
Sys.setenv(PATH = paste(Sys.getenv("PATH"),
   "path_to_your_BLAST_installation", sep=.Platform$path.sep))
```

BLAST Databases

You will also need a database. NCBI BLAST databases are updated daily and may be downloaded via FTP from https://ftp.ncbi.nlm.nih.gov/blast/db/. See `blast_db_cache()` on how to manage a local cache of database files.

BLAST databases are a set of database files with different extensions. All files start with the same database name. For example, `16S_ribosomal_RNA.tar.gz` contains files starting with `16S_ribosomal_RNA` which is the database name used for calling `blast()`.

Large databases are separated into several archives numbered `00`, `01`, etc. Download all archives and extract the files in the same directory. All files will have a common name which is the database name used for calling `blast()`.

Author(s)

Michael Hahsler

References

BLAST Help - BLAST+ Executable: https://blast.ncbi.nlm.nih.gov/doc/blast-help/downloadblastdata.html

BLAST Command Line Applications User Manual, https://www.ncbi.nlm.nih.gov/books/NBK279690/

See Also

Other blast: `blast_db_cache()`, `makeblastdb()`

Examples

```
## check if blastn is correctly installed. Should return the path to the
##   executable
Sys.which("blastn")

## only run if blast is installed
if (has_blast()) {
    ## check version you should have version 1.8.1+
    system2("blastn", "-version")

    ## download the latest version of the 16S Microbial
    ##  rRNA data base from NCBI using the local chache
    tgz_file <- blast_db_get("16S_ribosomal_RNA.tar.gz")

    ## extract the database files
    untar(tgz_file, exdir = "./16S_rRNA_DB")

    ## Note the database file can also downloaded without using a
    ##    cache using download.file
    # download.file(paste("https://ftp.ncbi.nlm.nih.gov/blast/db",
    #    "16S_ribosomal_RNA.tar.gz", sep = "/"),
    #    "16S_ribosomal_RNA.tar.gz", mode = "wb")
    # untar("16S_ribosomal_RNA.tar.gz", exdir = "./16S_rRNA_DB")

    ## A BLAST database is just a set of files. It is a good idea to
    ## organize the files in a directory.
    list.files("./16S_rRNA_DB")

    ## load a BLAST database (replace db with the location + name of
    ##   the BLAST DB without the extension)
    bl <- blast(db = "./16S_rRNA_DB/16S_ribosomal_RNA")
    bl

    ## read a single example sequence to BLAST
    seq <- readRNAStringSet(system.file("examples/RNA_example.fasta",
        package = "rBLAST"
    ))[1]
    seq

    ## query a sequence using BLAST
    cl <- predict(bl, seq)
    cl[1:5, ]

    ## Pass on BLAST arguments (99% identity) and use a custom format
    ## (see BLAST documentation)
    fmt <- paste(
        "qaccver saccver pident length mismatch gapopen qstart qend",
        "sstart send evalue bitscore qseq sseq"
    )
    cl <- predict(bl, seq,
        BLAST_args = "-perc_identity 99",
        custom_format = fmt
```

```
    )
    cl

    ## cleanup the example: delete the database files
    unlink("./16S_rRNA_DB", recursive = TRUE)
}
```

`blast_db_cache` *Manage BLAST Database Downloads using BioCFileCache*

Description

Use BiocFileCache::BiocFileCache to manage local copies of BLAST database downloads. NCBI BLAST databases are updated daily and may be downloaded via FTP from https://ftp.ncbi.nlm.nih.gov/blast/db/.

Usage

```
blast_db_cache()

blast_db_get(
  file = "16S_ribosomal_RNA.tar.gz",
  baseURL = "https://ftp.ncbi.nlm.nih.gov/blast/db/",
  check_update = TRUE,
  verbose = TRUE
)
```

Arguments

`file`	the filename of the database.
`baseURL`	URL to download blast databases from. The default is NCBI's ftp server.
`check_update`	logical; update the local cache if there is a newer version of the file available on the server. This may take some time.
`verbose`	logical; display download information.

Details

The package maintains its own local cache which can be accessed using `blast_db_cache()`.

Value

- `blast_db_cache()` returns the path to the local BiocFileCache::BiocFileCache cache.
- `blast_db_get()` returns the file path to a downloaded BLAST database file.

Author(s)

Michael Hahsler

See Also

Other blast: blast(), makeblastdb()

Examples

```
## get a database file (will be downloaded if the
##     local copy is not up-to-date)
db_16S <- blast_db_get("16S_ribosomal_RNA.tar.gz")
db_16S

## directly interacting with the local cache
library(BiocFileCache)

## show the package's cache directory
local_cache <- blast_db_cache()
local_cache

## bfc functions can be used to manage the local cache
bfcinfo(local_cache)
```

---

makeblastdb *Create BLAST Databases*

Description

Call the makeblastdb utility to create a BLAST database from a FASTA file.

Usage

```
makeblastdb(
  file,
  db_name = NULL,
  dbtype = "nucl",
  hash_index = TRUE,
  args = "",
  verbose = TRUE
)
```

Arguments

file	input file/database name. **Note** that the filename and path cannot contain whitespaces.
db_name	name of the database (files).
dbtype	molecule type of target db ("nucl" or "prot").
hash_index	logical; create index of sequence hash values.
args	string including additional arguments passed on to makeblastdb.
verbose	logical; show the progress report produced by makeblastdb?

Details

R needs to be able to find the executable (mostly an issue with Windows). Try `Sys.which("makeblastdb")` to see if the program is properly installed.

Use `blast_help("makeblastdb")` to see all possible extra arguments. Arguments need to be formated in exactly the way as they would be used for the command line tool.

Value

Nothing but creates a BLAST database directory.

Author(s)

Michael Hahsler

See Also

Other blast: `blast()`, `blast_db_cache()`

Examples

```
## check if makeblastdb is correctly installed
Sys.which("makeblastdb")

## only run if blast is installed
if (has_blast()) {
    ## see possible arguments
    blast_help("makeblastdb")

    ## read some example sequences
    seq <- readRNAStringSet(system.file("examples/RNA_example.fasta",
        package = "rBLAST"
    ))

    ## 1. write the FASTA file
    writeXStringSet(seq, filepath = "seqs.fasta")

    ## 2. make database
    makeblastdb(file = "seqs.fasta", db_name = "db/small", dbtype = "nucl")

    ## 3. open database
    db <- blast("db/small")
    db

    ## 4. perform search (first sequence in the db should be a perfect match)
    predict(db, seq[1])

    ## clean up
    unlink("seqs.fasta")
    unlink("db", recursive = TRUE)
}
```

Index