

Package: rBLAST (via r-universe)

October 11, 2026

Title R Interface for the Basic Local Alignment Search Tool

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

Version 1.9.4

Date 2026-xx-xx

biocViews Genetics, Sequencing, SequenceMatching, Alignment, MultipleSequenceAlignment, DataImport

Depends Biostrings (>= 2.26.2)

Imports methods, utils, BiocFileCache

Suggests data.table, knitr, rmarkdown, testthat, withr

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

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

SystemRequirements ncbi-blast+

License GPL-3

VignetteBuilder knitr

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libicu-dev libssl-dev zlib1g-dev

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

Date/Publication 2026-09-11 16:39:36 UTC

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

RemoteRef HEAD

RemoteSha 6e6073d0e92c933cab053dc42998f37fcc912bd9

Contents

blast	2
blast_db_cache	5
makeblastdb	7
Index	9

blast	<i>Basic Local Alignment Search Tool (BLAST)</i>
-------	--

Description

Open a local or remote BLAST database and search it with a BLAST+ command-line program such as blastn, blastp, or blastx.

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	Character scalar naming the database to search. For a local database, supply the path and database prefix without an index-file extension. For a remote search, supply an NCBI database name such as "nt" or "nr".
remote	Logical scalar. If TRUE, execute the query on an NCBI server. Remote searches use a shared service and can be substantially slower than local searches.
type	Character scalar naming the BLAST+ search program, for example "blastn", "blastp", or "blastx". The program must be compatible with the query sequences and database type.

info	Logical scalar. For <code>print.BLAST()</code> , query and display database metadata using <code>blastdbcmd</code> . Set to <code>FALSE</code> to print only the database path and BLAST program.
...	Additional arguments; currently ignored.
object, x	An open BLAST database as a BLAST object created with <code>blast()</code> .
newdata	Query sequences as a <code>Biostrings::XStringSet</code> object. Use a DNA or RNA string set with nucleotide searches and an amino-acid string set with protein searches.
BLAST_args	Character scalar containing additional BLAST+ command-line options, for example <code>"-perc_identity 99 -num_threads 4"</code> . The database, query, output file, and output format are managed by <code>predict.BLAST()</code> and should not be supplied here.
custom_format	Character scalar containing space-separated BLAST output field specifiers, such as <code>"qseqid sseqid pident length"</code> . An empty string uses the standard 12-column tabular format. See the BLAST+ documentation for available specifiers.
verbose	Logical scalar; report the generated files, command, and number of matches.
keep_tmp	Logical scalar; retain the query FASTA and BLAST output files in <code>tempdir()</code> for debugging. Retained files can be large, especially for repeated or parallel searches, and must be managed by the caller.

Value

- `blast()` returns an object of class `BLAST` containing the database, search program, and remote-search setting.
- `predict.BLAST()` returns a data frame with one row per hit. Column names follow `custom_format`, or the standard 12 BLAST fields when it is empty. A search without hits returns a zero-row data frame.
- `print.BLAST()` returns `NULL` invisibly and is called for its side effect.
- `blast_help()` returns the exit status from the BLAST+ help command.
- `has_blast()` returns a logical scalar indicating whether at least `blastn`, `makeblastdb`, and `blastdbcmd` can be found on `PATH`.

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()`.

`blast()` validates a local database with `blastdbcmd -info`. It does not download or create a database; use `blast_db_get()` for NCBI archives or `makeblastdb()` to build a database from a FASTA file.

`predict.BLAST()` writes newdata to a temporary FASTA file, invokes the selected BLAST+ program, and parses delimiter-separated tabular output. A FASTA write failure, a nonzero BLAST exit status, or a missing output file produces an error before result parsing. Use `verbose = TRUE` together with `keep_tmp = TRUE` when diagnosing a failed command. If the suggested package `data.table` is installed, output is read with `data.table::fread()` for improved performance; otherwise, `utils::read.table()` is used automatically.

BLAST's own multithreading is controlled with the `-num_threads` option in `BLAST_args`. This is distinct from running several `predict()` calls in parallel.

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 BLAST is installed
Sys.which(c("blastn", "makeblastdb", "blastdbcmd"))

## Build and query a small local database (only if BLAST is installed).
if (has_blast()) {
  ## Create a BLAST DB
  seq <- readRNAStringSet(system.file(
    "examples/RNA_example.fasta",
    package = "rBLAST"
  ))
  work_dir <- tempfile("rBLAST-example-")
```

```

dir.create(work_dir)
fasta <- file.path(work_dir, "sequences.fasta")
db_path <- file.path(work_dir, "database")

writeXStringSet(seq, fasta)
makeblastdb(fasta, db_name = db_path, dbtype = "nucl", verbose = FALSE)

## DB files
list.files(work_dir)

## Open and query the DB
db <- blast(db_path)
db

hits <- predict(
  db, seq[1],
  BLAST_args = "-perc_identity 99 -num_threads 2",
  custom_format = "qseqid sseqid pident length eval evalue bitscore"
)
hits

## Cleanup
unlink(work_dir, recursive = TRUE)
}

```

blast_db_cache

*Cache BLAST Database Archives***Description**

Download BLAST database archives and manage local copies with [BiocFileCache::BiocFileCache](#).

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	Character scalar giving the archive filename, for example "16S_ribosomal_RNA.tar.gz".
baseURL	Character scalar giving the directory URL from which file is downloaded. It must end in /.

check_update	Logical scalar; check whether the remote file is newer than the cached copy and download it when necessary.
verbose	Logical scalar; report whether a file is downloaded or a cached copy is returned.

Details

`blast_db_cache()` opens rBLAST's user-level cache. Its location is selected by `tools::R_user_dir()` and therefore depends on the operating system and environment.

`blast_db_get()` returns a cached archive when available and downloads it otherwise. With `check_update = TRUE`, the remote resource is checked and an outdated cached copy is refreshed. Downloaded archives still need to be extracted before their database prefix can be passed to `blast()`. Large NCBI databases can consist of multiple numbered archives; download and extract every part into the same directory.

Value

- `blast_db_cache()` returns a `BiocFileCache::BiocFileCache` object.
- `blast_db_get()` returns a character vector containing the local path to the cached archive.

Author(s)

Michael Hahsler

See Also

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

Examples

```
## Download or retrieve an existing cached copy
## (downloading ~70MB will take a little).
blast_db_get("16S_ribosomal_RNA.tar.gz")

## Extract the cached archive, then open the database using its prefix.
db_dir <- tempfile("16S-rRNA-")
dir.create(db_dir)
untar(blast_db_get("16S_ribosomal_RNA.tar.gz"), exdir = db_dir)
db <- blast(file.path(db_dir, "16S_ribosomal_RNA"))
db

## Inspect the underlying cache with BiocFileCache functions.
cache <- blast_db_cache()
BiocFileCache::bfcinfo(cache)
```

makeblastdb*Create BLAST Databases*

Description

Run the BLAST+ makeblastdb utility to create a local nucleotide or protein database from a FASTA file.

Usage

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

Arguments

file	Character scalar giving the input FASTA path.
db_name	Optional character scalar giving the output database prefix, including its directory if needed.
dbtype	Character scalar specifying the database molecule type: "nucl" or "prot".
hash_index	Logical scalar; add -hash_index to create an index of sequence hash values.
args	Character scalar containing additional makeblastdb command-line options.
verbose	Logical scalar; show output 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 available options. Additional options supplied through `args` are passed to the command-line program.

When `db_name` is `NULL`, BLAST uses `file` as the output database prefix. Otherwise, database index files are created using the `db_name` prefix. The destination directory must already exist. Paths containing whitespace are not supported by this interface.

Value

The integer exit status returned by makeblastdb; zero indicates success. The function is primarily called for its side effect of creating the database files.

Author(s)

Michael Hahsler

See Also

Other blast: [blast\(\)](#), [blast_db_cache\(\)](#)

Examples

```
## Only run if BLAST is installed
if (has_blast()) {
  ## Create an example BLAST DB
  seq <- readRNAStringSet(system.file("examples/RNA_example.fasta",
    package = "rBLAST"
  ))
  work_dir <- tempfile("rBLAST-db-")
  dir.create(work_dir)
  fasta <- file.path(work_dir, "sequences.fasta")
  db_path <- file.path(work_dir, "database")
  writeXStringSet(seq, fasta)

  status <- makeblastdb(
    fasta,
    db_name = db_path, dbtype = "nucl", verbose = FALSE
  )
  status

  ## Created DB files
  Sys.glob(paste0(db_path, ".*"))

  ## Cleanup
  unlink(work_dir, recursive = TRUE)
}
```

Index

* **blast**

- blast, [2](#)
- blast_db_cache, [5](#)
- makeblastdb, [7](#)

* **model**

- blast, [2](#)
- makeblastdb, [7](#)

BiocFileCache::BiocFileCache, [5](#), [6](#)

Biostrings::XStringSet, [3](#)

BLAST (blast), [2](#)

blast, [2](#)

blast(), [3](#), [6](#), [8](#)

blast_db_cache, [5](#)

blast_db_cache(), [4](#), [8](#)

blast_db_get (blast_db_cache), [5](#)

blast_db_get(), [4](#)

blast_help (blast), [2](#)

data.table::fread(), [4](#)

has_blast (blast), [2](#)

makeblastdb, [7](#)

makeblastdb(), [4](#), [6](#)

predict.BLAST (blast), [2](#)

print.BLAST (blast), [2](#)

tempdir(), [3](#)

tools::R_user_dir(), [6](#)