Package 'blit'

May 7, 2026

Title Bioinformatics Library for Integrated Tools

Version 0.2.0

Description An all-encompassing R toolkit designed to streamline the process of calling various bioinformatics software and then performing data analysis and visualization in R. With 'blit', users can easily integrate a wide array of bioinformatics command line tools into their workflows, leveraging the power of R for sophisticated data manipulation and graphical representation.

License GPL (>= 3)

Imports cli, processx, R6 (>= 2.4.0), rlang (>= 1.1.0), utils

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

URL https://github.com/WangLabCSU/blit

BugReports https://github.com/WangLabCSU/blit/issues

NeedsCompilation no

Author Yun Peng [aut, cre] (ORCID: <https://orcid.org/0000-0003-2801-3332>),
Shixiang Wang [aut] (ORCID: <https://orcid.org/0000-0001-9855-7357>),
Jennifer Lu [cph] (Author of the included scripts from Kraken2 and KrakenTools libraries),
Li Song [cph] (Author of included scripts from TRUST4 library),
X. Shirley Liu [cph] (Author of included scripts from TRUST4 library)

Maintainer Yun Peng <yunyunp96@163.com>

Repository CRAN

Date/Publication 2025-03-29 12:00:02 UTC

Contents

`allele_counter` *Run alleleCount*

Description

The `alleleCount` program primarily exists to prevent code duplication between some other projects, specifically `AscatNGS` and `Battenberg`.

Usage

```
allele_counter(
  hts_file,
  loci_file,
  ofile,
  ...,
  odir = getwd(),
  alleleCounter = NULL
)
```

Arguments

hts_file	A string of path to sample HTS file.
loci_file	A string of path to loci file.
ofile	A string of path to the output file.
...	<dynamic dots> Additional arguments passed to alleleCounter command. Empty arguments are automatically trimmed. If a single argument, such as a file path, contains spaces, it must be quoted, for example using shQuote(). Details see: cmd_help(alleleCounter()).
odir	A string of path to the output directory.
alleleCounter	A string of path to alleleCounter command.

Value

A command object.

See Also

- https://github.com/cancerit/alleleCount
- cmd_wd()/cmd_envvar()/cmd_envpath()/cmd_conda()
- cmd_on_start()/cmd_on_exit()
- cmd_on_succeed()/cmd_on_fail()
- cmd_parallel()

Other commands: cellranger(), conda(), fastq_pair(), gistic2(), kraken2(), kraken_tools(), perl(), pyscenic(), python(), samtools(), seqkit(), trust4()

---

appmamba *Manage Environment with* micromamba

Description

Manage Environment with micromamba

Usage

```
appmamba(...)

install_appmamba(force = FALSE)

uninstall_appmamba()

appmamba_rc(edit = FALSE)
```

Arguments

...	<dynamic dots> Additional arguments passed to micromamba. Run appmamba() for more details.
force	A logical value indicating whether to reinstall appmamba if it is already installed.
edit	A logical value indicating whether to open the config file for editing.

Functions

- appmamba(): blit utilizes micromamba to manage environments. This function simply executes the specified micromamba command.
- install_appmamba(): Install appmamba (micromamba).
- uninstall_appmamba(): Remove appmamba (micromamba).
- appmamba_rc(): Get the run commands config file of the micromamba.

Examples

```
install_appmamba()
appmamba()
appmamba("env", "list")
# uninstall_appmamba() # Uninstall the `micromamba`
```

---

arg *Deliver arguments of command*

Description

arg() is intended for user use, while arg0() is for developers and does not perform argument validation.

Usage

```
arg(tag, value, indicator = FALSE, lgl2int = FALSE, format = "%s", sep = " ")

arg0(
  tag,
  value,
  indicator = FALSE,
  lgl2int = FALSE,
  format = "%s",
  sep = " ",
  allow_null = FALSE,
  arg = caller_arg(value),
  call = caller_call()
)
```

Arguments

tag	A string specifying argument tag, like "-i", "-o".
value	Value passed to the argument.
indicator	A logical value specifying whether value should be an indicator of tag. If TRUE, logical value will explain the set or unset of tag.
lgl2int	A logical value indicates whether transfrom value TRUE to 1 or FALSE to 0. If TRUE, format will always be set to "%d".
format	The format of the value, details see sprintf.
sep	A character string used to separate "tag" and "value", usually " " or "=".
allow_null	A single logical value indicates whether value can be NULL.
arg	An argument name as a string. This argument will be mentioned in error messages as the input that is at the origin of a problem.
call	The execution environment of a currently running function.

Value

A string.

---

cellranger *Run cellranger*

Description

Run cellranger

Usage

```
cellranger(subcmd = NULL, ..., cellranger = NULL)
```

Arguments

subcmd	Sub-Command of cellranger.
...	<dynamic dots> Additional arguments passed to cellranger command. Empty arguments are automatically trimmed. If a single argument, such as a file path, contains spaces, it must be quoted, for example using shQuote(). Details see: cmd_help(cellranger()).
cellranger	A string of path to cellranger command.

Value

A command object.

See Also

- cmd_wd()/cmd_envvar()/cmd_envpath()/cmd_conda()
- cmd_on_start()/cmd_on_exit()
- cmd_on_succeed()/cmd_on_fail()
- cmd_parallel()

Other commands: allele_counter(), conda(), fastq_pair(), gistic2(), kraken2(), kraken_tools(), perl(), pyscenic(), python(), samtools(), seqkit(), trust4()

---

cmd_on_start *Schedule expressions to run*

Description

Schedule expressions to run

Usage

```
cmd_on_start(command, ...)

cmd_on_exit(command, ...)

cmd_on_fail(command, ...)

cmd_on_succeed(command, ...)
```

Arguments

command	A command object.
...	The expressions input will be captured with enquos(). If your expressions depend on global data, you may want to unquote objects with !! to prevent unintended changes due to delayed evaluation. • cmd_on_start: Expression to be evaluated when the command started. • cmd_on_exit: Expression to be evaluated when the command finished. • cmd_on_fail: Expression to be evaluated when the command failed. • cmd_on_succeed: Expression to be evaluated when the command succeeded.

Value

- cmd_on_start: The command object itself, with the start code updated.
- cmd_on_exit: The command object itself, with the exit code updated.
- cmd_on_fail: The command object itself, with the failure code updated.
- cmd_on_succeed: The command object itself, with the successful code updated.

Functions

- `cmd_on_start()`: define the startup code of the command
- `cmd_on_exit()`: define the exit code of the command
- `cmd_on_fail()`: define the failure code of the command
- `cmd_on_succeed()`: define the successful code of the command

`cmd_parallel` *Execute a list of commands*

Description

Execute a list of commands

Usage

```
cmd_parallel(
  ...,
  stdouts = FALSE,
  stderrs = FALSE,
  stdins = NULL,
  stdout_callbacks = NULL,
  stderr_callbacks = NULL,
  timeouts = NULL,
  threads = NULL,
  verbose = TRUE
)
```

Arguments

`...` A list of `command` object.

`stdouts, stderrs`

Specifies how the output/error streams of the child process are handled. One of or a list of following values:

- `FALSE/NULL`: Suppresses the output/error stream.
- `TRUE`: Prints the child process output/error to the R console. If a standard output/error stream exists, `""` is used; otherwise, `"|"` is used.
- **string**: An empty string `""` inherits the standard output/error stream from the main R process (Printing in the R console). If the main R process lacks a standard output/error stream, such as in `RGui` on Windows, an error is thrown. A string `"|"` prints to the standard output connection of R process (Using `cat()`). Alternative, a file name or path to redirect the output/error. If a relative path is specified, it remains relative to the current working directory, even if a different directory is set using `cmd_wd()`.
- `connection`: A writable R `connection` object. If the connection is not `open()`, it will be automatically opened.

For `stderrs`, use string `"2>&1"` to redirect it to the same connection (i.e. pipe or file) as `stdout`.

When a single file path is specified, the stdout/stderr of all commands will be merged into this single file.

`stdins` should the input be diverted? One of or a list of following values:

- `FALSE`/`NULL`: no standard input.
- `TRUE`: If a standard input stream exists, `""` is used; otherwise, `NULL` is used.
- **string**: An empty string `""` inherits the standard input stream from the main R process. If the main R process lacks a standard input stream, such as in `RGui` on Windows, an error is thrown. Alternative, a file name or path to redirect the input. If a relative path is specified, it remains relative to the current working directory, even if a different directory is set using `cmd_wd()`.

`stdout_callbacks`, `stderr_callbacks`
One of or a list of following values:

- `NULL`: no callback function.
- `function`: A function invoked for each line of standard output or error. Non-text (non-character) output will be ignored. The function should accept two arguments: one for the standard output or error and another for the running `process` object.

`timeouts` Timeout in seconds. Can be a single value or a list, specifying the maximum elapsed time for running the command in the separate process.

`threads` Number of threads to use.

`verbose` A single boolean value indicating whether the command execution should be verbose.

Value

A list of exit status invisiblely.

See Also

- `cmd_wd()`/`cmd_envvar()`/`cmd_envpath()`/`cmd_conda()`
- `cmd_on_start()`/`cmd_on_exit()`
- `cmd_on_succeed()`/`cmd_on_fail()`
- `cmd_parallel()`

`cmd_run` *Execute command*

Description

- `cmd_run`: Run the command.
- `cmd_help`: Print the help document for this command.
- `cmd_background`: Run the command in the background. This function is provided for completeness. Instead of using this function, we recommend using `cmd_parallel()`, which can run multiple commands in the background while ensuring that all processes are properly cleaned up when the process exits.

Usage

```
cmd_run(
  command,
  stdout = TRUE,
  stderr = TRUE,
  stdin = TRUE,
  stdout_callback = NULL,
  stderr_callback = NULL,
  timeout = NULL,
  spinner = FALSE,
  verbose = TRUE
)

cmd_help(
  command,
  stdout = TRUE,
  stderr = TRUE,
  stdout_callback = NULL,
  stderr_callback = NULL,
  verbose = TRUE
)

cmd_background(
  command,
  stdout = FALSE,
  stderr = FALSE,
  stdin = NULL,
  verbose = TRUE
)
```

Arguments

`command`	A `command` object.
`stdout`, `stderr`	Specifies how the output/error streams of the child process are handled. Possible values include: • `FALSE`/`NULL`: Suppresses the output/error stream. • `TRUE`: Prints the child process output/error to the R console. If a standard output/error stream exists, `""` is used; otherwise, `"|"` is used.

- **string**: An empty string `""` inherits the standard output/error stream from the main R process (Printing in the R console). If the main R process lacks a standard output/error stream, such as in `RGui` on Windows, an error is thrown. A string `"|"` prints to the standard output connection of R process (Using `cat()`). Alternative, a file name or path to redirect the output/error. If a relative path is specified, it remains relative to the current working directory, even if a different directory is set using `cmd_wd()`.
- `connection`: A writable R `connection` object. If the connection is not `open()`, it will be automatically opened.

For `stderr`, use string `"2>&1"` to redirect it to the same connection (i.e. pipe or file) as `stdout`.

For `cmd_help()`, use `FALSE/NULL` will do nothing, since it always want to display the help document.

For `cmd_background()`, `connection` cannot be used, and `TRUE` and `"|"` will fallback to the empty string `""`.

When using a `connection` (if not already open) or a `string`, wrapping it with `I()` prevents overwriting existing content.

`stdin` should the input be diverted? Possible values include:

- `FALSE/NULL`: no standard input.
- `TRUE`: If a standard input stream exists, `""` is used; otherwise, `NULL` is used.
- **string**: An empty string `""` inherits the standard input stream from the main R process. If the main R process lacks a standard input stream, such as in `RGui` on Windows, an error is thrown. Alternative, a file name or path to redirect the input. If a relative path is specified, it remains relative to the current working directory, even if a different directory is set using `cmd_wd()`.

`stdout_callback, stderr_callback`

Possible values include:

- `NULL`: no callback function.
- `function`: A function invoked for each line of standard output or error. Non-text (non-character) output will be ignored. The function should accept two arguments: one for the standard output or error and another for the running `process` object.

`timeout` Timeout in seconds. This is a limit for the elapsed time running command in the separate process.

`spinner` Whether to show a reassuring spinner while the process is running.

`verbose` A single boolean value indicating whether the command execution should be verbose.

Value

- `cmd_run`: Exit status invisiblely.
- `cmd_help`: The input `command` invisiblely.
- `cmd_background`: A `process` object.

See Also

- cmd_wd()/cmd_envvar()/cmd_envpath()/cmd_conda()
- cmd_on_start()/cmd_on_exit()
- cmd_on_succeed()/cmd_on_fail()
- cmd_parallel()

---

cmd_wd *Setup the context for the command*

Description

Setup the context for the command

Usage

```
cmd_wd(command, wd = NULL)

cmd_envvar(command, ..., action = "replace", sep = NULL)

cmd_envpath(command, ..., action = "prefix", name = "PATH")

cmd_conda(command, ..., root = NULL, action = "prefix")
```

Arguments

command	A command object.
wd	A string or NULL define the working directory of the command.
...	<dynamic dots>: • cmd_envvar: Named character define the environment variables. • cmd_envpath: Unnamed character to define the PATH-like environment variables name. • cmd_conda: Unnamed character to specify the name of conda environment.
action	Should the new values "replace", "prefix" or "suffix" existing environment variables?
sep	A string to separate new and old value when action is "prefix" or "suffix". By default, " " will be used.
name	A string define the PATH environment variable name. You can use this to define other PATH-like environment variable such as PYTHONPATH.
root	A string specifying the path to the conda root prefix. By default, it utilizes the environment variable BLIT_CONDA_ROOT or the option blit.conda.root. If neither is set, the root prefix of appmamba() will be used.

Value

- cmd_wd: The command object itself, with working directory updated.
- cmd_envvar: The command object itself, with running environment variable updated.
- cmd_envpath: The command object itself, with running environment variable specified in name updated.
- cmd_conda: The command object itself, with running environment variable PATH updated.

Functions

- cmd_wd(): define the working directory.
- cmd_envvar(): define the environment variables.
- cmd_envpath(): define the PATH-like environment variables.
- cmd_conda(): Set conda-like path to the PATH environment variables.

See Also

- cmd_run()/cmd_help()/cmd_background()
- cmd_on_start()/cmd_on_exit()
- cmd_on_succeed()/cmd_on_fail()
- cmd_parallel()

Command *R6 Class to prepare command parameters.*

Description

Command is an R6 class used by developers to create new command. It should not be used by end users.

Methods

Public methods:

- Command$new()
- Command$build_command()
- Command$get_on_start()
- Command$get_on_exit()
- Command$get_on_fail()
- Command$get_on_succeed()
- Command$print()
- Command$clone()

Method `new()`**:** Create a new `Command` object.

Usage:

```
Command$new(...)
```

Arguments:

`...` Additional argument passed into command.

Method `build_command()`**:** Build the command line

Usage:

```
Command$build_command(help = FALSE, verbose = TRUE)
```

Arguments:

`help` A boolean value indicating whether to build parameters for help document or not.
`verbose` A boolean value indicating whether the command execution should be verbose.
`envir` An environment used to Execute command.

Returns: An atomic character combine the command and parameters.

Method `get_on_start()`**:** Get the command startup code

Usage:

```
Command$get_on_start()
```

Returns: A list of `quosures`.

Method `get_on_exit()`**:** Get the command exit code

Usage:

```
Command$get_on_exit()
```

Returns: A list of `quosures`.

Method `get_on_fail()`**:** Get the command failure code

Usage:

```
Command$get_on_fail()
```

Returns: A list of `quosures`.

Method `get_on_succeed()`**:** Get the command succeessful code

Usage:

```
Command$get_on_succeed()
```

Returns: A list of `quosures`.

Method `print()`**:** Build parameters to run command.

Usage:

```
Command$print(indent = NULL)
```

Arguments:

`indent` A single integer number giving the space of indent.

Returns: The object itself.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Command$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

make_command

---

conda *Run conda*

Description

Run conda

Usage

```
conda(subcmd = NULL, ..., conda = NULL)
```

Arguments

subcmd	Sub-Command of conda.
...	<dynamic dots> Additional arguments passed to conda command. Empty arguments are automatically trimmed. If a single argument, such as a file path, contains spaces, it must be quoted, for example using `shQuote()`. Details see: `cmd_help(conda())`.
conda	A string of path to conda command.

Value

A command object.

See Also

- `cmd_wd()`/`cmd_envvar()`/`cmd_envpath()`/`cmd_conda()`
- `cmd_on_start()`/`cmd_on_exit()`
- `cmd_on_succeed()`/`cmd_on_fail()`
- `cmd_parallel()`

Other commands: `allele_counter()`, `cellranger()`, `fastq_pair()`, `gistic2()`, `kraken2()`, `kraken_tools()`, `perl()`, `pyscenic()`, `python()`, `samtools()`, `seqkit()`, `trust4()`

exec *Invoke a System Command*

Description

Invoke a System Command

Usage

```
exec(cmd, ...)
```

Arguments

cmd	Command to be invoked, as a character string.
...	<dynamic dots> Additional arguments passed to cmd command. Empty arguments are automatically trimmed. If a single argument, such as a file path, contains spaces, it must be quoted, for example using shQuote().

Value

A command object.

command collections

- allele_counter()
- cellranger()
- conda()
- fastq_pair()
- gistic2()
- kraken_tools()
- kraken2()
- perl()
- pyscenic()
- python()
- samtools()
- seqkit()
- trust4()

See Also

- cmd_wd()/cmd_envvar()/cmd_envpath()/cmd_conda()
- cmd_on_start()/cmd_on_exit()
- cmd_on_succeed()/cmd_on_fail()
- cmd_parallel()

Examples

```
cmd_run(exec("echo", "$PATH"))
```

`fastq_pair` *FASTQ PAIR*

Description

Rewrite paired end fastq files to make sure that all reads have a mate and to separate out singletons.

Usually when you get paired end read files you have two files with a /1 sequence in one and a /2 sequence in the other (or a /f and /r or just two reads with the same ID). However, often when working with files from a third party source (e.g. the SRA) there are different numbers of reads in each file (because some reads fail QC). Spades, bowtie2 and other tools break because they demand paired end files have the same number of reads.

Usage

```
fastq_pair(
  fq1,
  fq2,
  ...,
  hash_table_size = NULL,
  max_hash_table_size = NULL,
  fastq_pair = NULL
)

fastq_read_pair(fastq_files)
```

Arguments

`fq1`, `fq2`	A string of fastq file path.
`...`	<dynamic dots> Additional arguments passed to `fastq_pair` command. Empty arguments are automatically trimmed. If a single argument, such as a file path, contains spaces, it must be quoted, for example using `shQuote()`. Details see: `cmd_help(fastq_pair())`.
`hash_table_size`	Size of hash table to use.
`max_hash_table_size`	Maximal hash table size to use.
`fastq_pair`	A string of path to `fastq_pair` command.
`fastq_files`	A character of the fastq file paths.

Value

A `command` object.

See Also

- https://github.com/linsalrob/fastq-pair
- cmd_wd()/cmd_envvar()/cmd_envpath()/cmd_conda()
- cmd_on_start()/cmd_on_exit()
- cmd_on_succeed()/cmd_on_fail()
- cmd_parallel()

Other commands: allele_counter(), cellranger(), conda(), gistic2(), kraken2(), kraken_tools(), perl(), pyscenic(), python(), samtools(), seqkit(), trust4()

---

gistic2 *Run GISTIC2*

Description

The GISTIC module identifies regions of the genome that are significantly amplified or deleted across a set of samples. Each aberration is assigned a G-score that considers the amplitude of the aberration as well as the frequency of its occurrence across samples. False Discovery Rate q-values are then calculated for the aberrant regions, and regions with q-values below a user-defined threshold are considered significant. For each significant region, a "peak region" is identified, which is the part of the aberrant region with greatest amplitude and frequency of alteration. In addition, a "wide peak" is determined using a leave-one-out algorithm to allow for errors in the boundaries in a single sample. The "wide peak" boundaries are more robust for identifying the most likely gene targets in the region. Each significantly aberrant region is also tested to determine whether it results primarily from broad events (longer than half a chromosome arm), focal events, or significant levels of both. The GISTIC module reports the genomic locations and calculated q-values for the aberrant regions. It identifies the samples that exhibit each significant amplification or deletion, and it lists genes found in each "wide peak" region.

Usage

```
gistic2(seg, refgene, ..., odir = getwd(), gistic2 = NULL)
```

Arguments

seg	A data.frame of segmented data.
refgene	Path to reference genome data input file (REQUIRED, see below for file description).
...	<dynamic dots> Additional arguments passed to gistic2 command. Empty arguments are automatically trimmed. If a single argument, such as a file path, contains spaces, it must be quoted, for example using shQuote(). Details see: cmd_help(gistic2()).
odir	A string of path to the output directory.
gistic2	A string of path to gistic2 command.

Value

A command object.

See Also

- https://broadinstitute.github.io/gistic2/
- cmd_wd()/cmd_envvar()/cmd_envpath()/cmd_conda()
- cmd_on_start()/cmd_on_exit()
- cmd_on_succeed()/cmd_on_fail()
- cmd_parallel()

Other commands: allele_counter(), cellranger(), conda(), fastq_pair(), kraken2(), kraken_tools(), perl(), pyscenic(), python(), samtools(), seqkit(), trust4()

---

kraken2 *Running Kraken2*

Description

Kraken is a taxonomic sequence classifier that assigns taxonomic labels to DNA sequences. Kraken examines the k-mers within a query sequence and uses the information within those k-mers to query a database. That database maps k-mers to the lowest common ancestor (LCA) of all genomes known to contain a given k-mer.

Usage

```
kraken2(
  fq1,
  ...,
  fq2 = NULL,
  ofile = "kraken_output.txt",
  report = "kraken_report.txt",
  classified_out = NULL,
  unclassified_out = NULL,
  odir = getwd(),
  kraken2 = NULL
)
```

Arguments

fq1, fq2	A string of fastq file path.
...	<dynamic dots> Additional arguments passed to kraken2 command. Empty arguments are automatically trimmed. If a single argument, such as a file path, contains spaces, it must be quoted, for example using shQuote(). Details see: cmd_help(kraken2()).
ofile	A string of path to save kraken2 output.

report	A string of path to save kraken2 report.
classified_out	A string of path to save classified sequences, which should be a fastq file.
unclassified_out	A string of path to save unclassified sequences, which should be a fastq file.
odir	A string of path to the output directory.
kraken2	A string of path to kraken2 command.

Value

A command object.

See Also

- https://github.com/DerrickWood/kraken2/wiki/Manual
- https://benlangmead.github.io/aws-indexes/k2
- cmd_wd()/cmd_envvar()/cmd_envpath()/cmd_conda()
- cmd_on_start()/cmd_on_exit()
- cmd_on_succeed()/cmd_on_fail()
- cmd_parallel()

Other commands: allele_counter(), cellranger(), conda(), fastq_pair(), gistic2(), kraken_tools(), perl(), pyscenic(), python(), samtools(), seqkit(), trust4()

---

kraken_tools *KrakenTools is a suite of scripts to be used alongside the Kraken, KrakenUniq, Kraken 2, or Bracken programs.*

Description

These scripts are designed to help Kraken users with downstream analysis of Kraken results.

Usage

```
kraken_tools(script, ..., python = NULL)
```

Arguments

script	Name of the kraken2 script. One of "combine_kreports", "combine_mpa", "extract_kraken_reads", "filter_bracken_out", "fix_unmapped", "kreport2krona", "kreport2mpa", "make_kreport", and "make_ktaxonomy".
...	<dynamic dots> Additional arguments passed to kraken_tools command. Empty arguments are automatically trimmed. If a single argument, such as a file path, contains spaces, it must be quoted, for example using shQuote(). Details see: cmd_help(kraken_tools()).
python	A string of path to python command.

Value

A command object.

See Also

- https://github.com/jenniferlu717/KrakenTools
- cmd_wd()/cmd_envvar()/cmd_envpath()/cmd_conda()
- cmd_on_start()/cmd_on_exit()
- cmd_on_succeed()/cmd_on_fail()
- cmd_parallel()

Other commands: allele_counter(), cellranger(), conda(), fastq_pair(), gistic2(), kraken2(), perl(), pyscenic(), python(), samtools(), seqkit(), trust4()

---

make_command *Helper function to create new command.*

Description

make_command is a helper function used by developers to create function for a new Command object. It should not be used by end users.

Usage

```
make_command(name, fun, envir = caller_env())
```

Arguments

name	A string of the function name.
fun	A function used to initialize the Command object.
envir	A environment used to bind the created function.

Value

A function.

See Also

Command

`perl` *Perl is a highly capable, feature-rich programming language with over 36 years of development.*

Description

Perl is a highly capable, feature-rich programming language with over 36 years of development.

Usage

```
perl(..., perl = NULL)
```

Arguments

`...`	<dynamic dots> Additional arguments passed to `perl` command. Empty arguments are automatically trimmed. If a single argument, such as a file path, contains spaces, it must be quoted, for example using `shQuote()`. Details see: `cmd_help(perl())`.
`perl`	A string of path to `perl` command.

Value

A command object.

See Also

- https://www.perl.org/
- `cmd_wd()`/`cmd_envvar()`/`cmd_envpath()`/`cmd_conda()`
- `cmd_on_start()`/`cmd_on_exit()`
- `cmd_on_succeed()`/`cmd_on_fail()`
- `cmd_parallel()`

Other commands: `allele_counter()`, `cellranger()`, `conda()`, `fastq_pair()`, `gistic2()`, `kraken2()`, `kraken_tools()`, `pyscenic()`, `python()`, `samtools()`, `seqkit()`, `trust4()`

`pyscenic` *Run pyscenic*

Description

Run pyscenic

Usage

```
pyscenic(subcmd = NULL, ..., pyscenic = NULL)
```

Arguments

subcmd	Sub-Command of pyscenic.
...	<dynamic dots> Additional arguments passed to pyscenic subcmd command. Empty arguments are automatically trimmed. If a single argument, such as a file path, contains spaces, it must be quoted, for example using shQuote(). Details see: cmd_help(pyscenic subcmd()).
pyscenic	A string of path to pyscenic command.

Value

A command object.

See Also

- https://github.com/aertslab/pySCENIC
- cmd_wd()/cmd_envvar()/cmd_envpath()/cmd_conda()
- cmd_on_start()/cmd_on_exit()
- cmd_on_succeed()/cmd_on_fail()
- cmd_parallel()

Other commands: allele_counter(), cellranger(), conda(), fastq_pair(), gistic2(), kraken2(), kraken_tools(), perl(), python(), samtools(), seqkit(), trust4()

---

python *Python is a programming language that lets you work quickly and integrate systems more effectively.*

Description

Python is a programming language that lets you work quickly and integrate systems more effectively.

Usage

```
python(..., python = NULL)
```

Arguments

...	<dynamic dots> Additional arguments passed to python command. Empty arguments are automatically trimmed. If a single argument, such as a file path, contains spaces, it must be quoted, for example using shQuote(). Details see: cmd_help(python()).
python	A string of path to python command.

Value

A command object.

See Also

- https://www.python.org/
- cmd_wd()/cmd_envvar()/cmd_envpath()/cmd_conda()
- cmd_on_start()/cmd_on_exit()
- cmd_on_succeed()/cmd_on_fail()
- cmd_parallel()

Other commands: allele_counter(), cellranger(), conda(), fastq_pair(), gistic2(), kraken2(), kraken_tools(), perl(), pyscenic(), samtools(), seqkit(), trust4()

---

samtools *Python is a programming language that lets you work quickly and integrate systems more effectively.*

Description

Python is a programming language that lets you work quickly and integrate systems more effectively.

Usage

```
samtools(subcmd = NULL, ..., samtools = NULL)
```

Arguments

subcmd	Sub-Command of samtools. Details see: cmd_help(samtools()).
...	<dynamic dots> Additional arguments passed to samtools command. Empty arguments are automatically trimmed. If a single argument, such as a file path, contains spaces, it must be quoted, for example using shQuote(). Details see: cmd_help(samtools()).
samtools	A string of path to samtools command.

Value

A command object.

See Also

- https://www.htslib.org/
- cmd_wd()/cmd_envvar()/cmd_envpath()/cmd_conda()
- cmd_on_start()/cmd_on_exit()
- cmd_on_succeed()/cmd_on_fail()
- cmd_parallel()

Other commands: allele_counter(), cellranger(), conda(), fastq_pair(), gistic2(), kraken2(), kraken_tools(), perl(), pyscenic(), python(), seqkit(), trust4()

`seqkit` *Run seqkit*

Description

Run seqkit

Usage

```
seqkit(subcmd = NULL, ..., seqkit = NULL)
```

Arguments

subcmd	Sub-Command of seqkit.
...	<dynamic dots> Additional arguments passed to `seqkit subcmd` command. Empty arguments are automatically trimmed. If a single argument, such as a file path, contains spaces, it must be quoted, for example using shQuote(). Details see: `cmd_help(seqkit subcmd())`.
seqkit	A string of path to `seqkit` command.

Value

A command object.

See Also

- https://bioinf.shenwei.me/seqkit/
- cmd_wd()/cmd_envvar()/cmd_envpath()/cmd_conda()
- cmd_on_start()/cmd_on_exit()
- cmd_on_succeed()/cmd_on_fail()
- cmd_parallel()

Other commands: allele_counter(), cellranger(), conda(), fastq_pair(), gistic2(), kraken2(), kraken_tools(), perl(), pyscenic(), python(), samtools(), trust4()

trust4 *TRUST4: immune repertoire reconstruction from bulk and single-cell RNA-seq data*

Description

TRUST4: immune repertoire reconstruction from bulk and single-cell RNA-seq data

Usage

```
trust4(
  file1,
  ref_coordinate,
  ...,
  file2 = NULL,
  mode = NULL,
  ref_annot = NULL,
  ofile = NULL,
  odir = getwd(),
  trust4 = NULL
)

trust4_imgt_annot(
  species = "Homo_sapien",
  ...,
  ofile = "IMGT+C.fa",
  odir = getwd(),
  perl = NULL
)

trust4_gene_names(imgt_annot, ofile = "bcr_tcr_gene_name.txt", odir = getwd())
```

Arguments

file1	Path to bam file or fastq file.
ref_coordinate	Path to the fasta file coordinate and sequence of V/D/J/C genes.
...	• trust4: <dynamic dots> Additional arguments passed to run-trust4 command. Empty arguments are automatically trimmed. If a single argument, such as a file path, contains spaces, it must be quoted, for example using shQuote(). Details see: cmd_help(run-trust4()). • trust4_imgt_annot: <dynamic dots> Additional arguments passed to trust4_imgt_annot command. Empty arguments are automatically trimmed. If a single argument, such as a file path, contains spaces, it must be quoted, for example using shQuote(). Details see: cmd_help(trust4_imgt_annot()).
file2	Path to the second paired-end read fastq file, only used for mode = "fastq".
mode	One of "bam" or "fastq". If NULL, will be inferred from file1.

ref_annot	Path to detailed V/D/J/C gene reference file, such as from IMGT database. (default: not used). (recommended).
ofile	• trust4: Prefix of output files. (default: inferred from file prefix). • trust4_imgt_annot: Output file name. • trust4_gene_names: Output file name.
odir	A string of path to the output directory.
trust4	A string of path to run-trust4 command.
species	Species to extract IMGT annotation, details see https://www.imgt.org//download/V-QUEST/IMGT_V-QUEST_reference_directory/.
perl	A string of path to perl command.
imgt_annot	Path of IMGT annotation file, created via trust4_imgt_annot.

Value

A command object.

See Also

- https://github.com/liulab-dfci/TRUST4
- cmd_wd()/cmd_envvar()/cmd_envpath()/cmd_conda()
- cmd_on_start()/cmd_on_exit()
- cmd_on_succeed()/cmd_on_fail()
- cmd_parallel()

Other commands: allele_counter(), cellranger(), conda(), fastq_pair(), gistic2(), kraken2(), kraken_tools(), perl(), pyscenic(), python(), samtools(), seqkit()

Index