Package 'btw'

November 4, 2025

Title A Toolkit for Connecting R and Large Language Models

Version 1.0.0

Description A complete toolkit for connecting 'R' environments with Large Language Models (LLMs). Provides utilities for describing 'R' objects, package documentation, and workspace state in plain text formats optimized for LLM consumption. Supports multiple workflows: interactive copy-paste to external chat interfaces, programmatic tool registration with 'ellmer' chat clients, batteries-included chat applications via 'shinychat', and exposure to external coding agents through the Model Context Protocol. Project configuration files enable stable, repeatable conversations with project-specific context and preferred LLM settings.

License MIT + file LICENSE

URL https://github.com/posit-dev/btw, https://posit-dev.github.io/btw/

BugReports https://github.com/posit-dev/btw/issues

Depends R (>= 4.1.0)

Imports cli, clipr, dplyr, ellmer (>= 0.3.0), fs, jsonlite, lifecycle, mcptools, pkgsearch, rlang (>= 1.1.0), rmarkdown, rstudioapi, S7, sessioninfo, skimr, tibble, utils, withr, xml2

Suggests bslib (>= 0.7.0), chromote, DBI, duckdb, gert, gh, htmltools, pandoc, shiny, shinychat (>= 0.2.0), testthat (>= 3.0.0), usethis

Config/Needs/website tidyverse/tidytemplate

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

RoxygenNote 7.3.3

Collate 'aaa-tools.R' 'addins.R' 'btw-package.R' 'btw.R' 'btw_client.R' 'btw_client_app.R' 'btw_this.R' 'clipboard.R' 'edit_btw_md.R' 'import-standalone-obj-type.R' 'import-standalone-purrr.R' 'import-standalone-types-check.R'

'mcp.R' 'task_create_btw_md.R' 'task_create_readme.R'
'tool-data-frame.R' 'tool-result.R' 'tool-docs-news.R'
'tool-docs.R' 'tool-environment.R' 'tool-files-code-search.R'
'tool-files.R' 'tool-git.R' 'tool-github.R' 'tool-rstudioapi.R'
'tool-search-packages.R' 'tool-session-package-installed.R'
'tool-sessioninfo.R' 'tool-web.R' 'tools.R' 'utils-ellmer.R'
'utils-gitignore.R' 'utils-md.R' 'utils-r.R' 'utils.R' 'zzz.R'

NeedsCompilation no

Author Garrick Aden-Buie [aut, cre] (ORCID:
<https://orcid.org/0000-0002-7111-0077>),
Simon Couch [aut] (ORCID: <https://orcid.org/0000-0001-5676-5107>),
Joe Cheng [aut],
Posit Software, PBC [cph, fnd],
Google [cph] (Material Design Icons),
Jamie Perkins [cph] (countUp.js author)

Maintainer Garrick Aden-Buie <garrick@adenbuie.com>

Repository CRAN

Date/Publication 2025-11-04 19:10:07 UTC

Contents

btw . . . 3
btw_client . . . 4
btw_task_create_btw_md . . . 6
btw_task_create_readme . . . 8
btw_this . . . 9
btw_this.character . . . 10
btw_this.data.frame . . . 12
btw_this.environment . . . 13
btw_tools . . . 14
btw_tool_docs_package_news . . . 17
btw_tool_env_describe_data_frame . . . 18
btw_tool_env_describe_environment . . . 19
btw_tool_files_code_search . . . 20
btw_tool_files_list_files . . . 22
btw_tool_files_read_text_file . . . 23
btw_tool_files_write_text_file . . . 24
btw_tool_github . . . 25
btw_tool_git_branch_checkout . . . 27
btw_tool_git_branch_create . . . 28
btw_tool_git_branch_list . . . 29
btw_tool_git_commit . . . 30
btw_tool_git_diff . . . 31
btw_tool_git_log . . . 32
btw_tool_git_status . . . 33
btw_tool_ide_read_current_editor . . . 34

btw_tool_package_docs 35
btw_tool_search_packages 36
btw_tool_search_package_info 37
btw_tool_session_check_package_installed 38
btw_tool_session_package_info 39
btw_tool_session_platform_info 40
btw_tool_web_read_url 41
mcp 42
use_btw_md 44

Index **48**

`btw` *Plain-text descriptions of R objects*

Description

This function allows you to quickly describe your computational environment to a model by concatenating plain-text descriptions of "R stuff", from data frames to packages to function documentation.

There are two key ways to use `btw()`:

1. Use it interactively at the console to gather information about your environment into prompt text that you can paste into the chat interface of an LLM, like ChatGPT or Claude. By default, `btw()` copies the prompt to the clipboard for you.

   ```
   btw(vignette("colwise", "dplyr"), dplyr::across, dplyr::starwars)
   #> btw copied to the clipboard!
   ```

2. Pair `btw()` with ellmer::Chat during a chat session to create a prompt that includes additional context drawn from your environment and help pages.

   ```
   library(ellmer)

   chat <- chat_anthropic() # requires an Anthropic API key
   chat <- chat_ollama(model = "llama3.1:8b") # requires ollama and a local model

   chat$chat(btw(
     vignette("colwise", "dplyr"),
     dplyr::across,
     dplyr::starwars,
     "Create a few interesting examples that use `dplyr::across()`",
     "with the `starwars` data set."
   ))
   ```

Additional examples:

1. Use `btw()` without arguments to describe all objects in your workspace:

   ```
   btw()
   #> btw copied to the clipboard!
   ```

2. Describe a function (it's documentation) and a data frame:

```
btw(dplyr::mutate, mtcars)
#> btw copied to the clipboard!
```

3. Use `btw()` to give additional context to an ellmer::Chat session:

```
library(ellmer)

chat <- chat_ollama(model = "llama3.1:8b")
chat$chat(
  btw(mtcars, "Are there cars with 8 cylinders in this dataset?")
)
```

Usage

```
btw(..., clipboard = TRUE)
```

Arguments

`...`	Objects to describe from your R environment. You can pass objects themselves, like data frames or functions, or the function also accepts output from `btw_tool_*()` functions like `btw_tool_docs_package_help_topics()`, `btw_tool_docs_help_page()` etc. If omitted, this function will just describe the elements in your global R environment.
`clipboard`	Whether to write the results to the clipboard. A single logical value; will default to `TRUE` when run interactively.

Value

Returns an ellmer::ContentText object with the collected prompt. If `clipboard = TRUE`, the prompt text is copied to the clipboard when the returned object is printed for the first time (e.g. calling `btw()` without assignment).

Examples

```
# See documentation for detailed examples
btw(mtcars)
```

`btw_client` *Create a btw-enhanced ellmer chat client*

Description

Creates an ellmer::Chat client, enhanced with the tools from `btw_tools()`. Use `btw_client()` to create the chat client for general or interactive use at the console, or `btw_app()` to create a chat client and launch a Shiny app for chatting with a btw-enhanced LLM in your local workspace.

Project Context:

You can keep track of project-specific rules, guidance and context by adding a `btw.md` file or `AGENTS.md` in your project directory. See `use_btw_md()` for help creating a `btw.md` file in your project, or use `path_btw` to tell `btw_client()` to use a specific context file.

`btw_client()` will also include context from an `llms.txt` file in the system prompt, if one is found in your project directory or as specified by the `path_llms_txt` argument.

Client Settings with User-Level Fallback:

Client settings in `client` and `tools` from a project-level `btw.md` or `AGENTS.md` file take precedence. If a project file doesn't specify a setting, btw will fall back to settings in a user-level `btw.md` file (typically in `~/btw.md` or `~/.config/btw/btw.md`). Project-level btw tool options under the `options` key are merged with user-level options, with project-level options taking precedence.

Project-specific instructions from both files are combined with a divider, allowing you to maintain global guidelines in your user file and project-specific rules in your project file.

Client Options:

The following R options are consulted when creating a new btw chat client and take precedence over settings in a `btw.md` file:

- `btw.client`: The ellmer::Chat client or a `provider/model` string (see `ellmer::chat()`) to use as the basis for new `btw_client()` or `btw_app()` chats.
- `btw.tools`: The btw tools to include by default when starting a new btw chat, see `btw_tools()` for details.‘

Usage

```
btw_client(
  ...,
  client = NULL,
  tools = NULL,
  path_btw = NULL,
  path_llms_txt = NULL
)

btw_app(..., client = NULL, tools = NULL, path_btw = NULL, messages = list())
```

Arguments

`...`	In `btw_app()`, additional arguments are passed to `shiny::shinyApp()`. In `btw_client()`, additional arguments are ignored.
`client`	An ellmer::Chat client or a `provider/model` string to be passed to `ellmer::chat()` to create a chat client. Defaults to `ellmer::chat_anthropic()`. You can use the `btw.client` option to set a default client for new `btw_client()` calls, or use a `btw.md` project file for default chat client settings, like provider and model. We check the `client` argument, then the `btw.client` R option, and finally the `btw.md` project file (falling back to user-level `btw.md` if needed), using only the client definition from the first of these that is available.

`tools`	A list of tools to include in the chat, defaults to `btw_tools()`. Join `btw_tools()` with additional tools defined by `ellmer::tool()` to include additional tools in the chat client. Alternatively, you can use a character values to refer to specific btw tools by name or by group. For example, use `tools = "docs"` to include only the documentation related tools, or `tools = c("env", "docs")` to include the environment and documentation tools, and so on. You can also refer to btw tools by name, e.g. `tools = "btw_tool_docs_help_page"` or alternatively in the shorter form `tools = "docs_help_page"`. Finally, set `tools = FALSE` to skip registering **btw** tools with the chat client.
`path_btw`	A path to a `btw.md` or `AGENTS.md` project context file. If `NULL`, btw will find a project-specific `btw.md` or `AGENTS.md` file in the parents of the current working directory, with fallback to user-level `btw.md` if no project file is found. Set `path_btw = FALSE` to create a chat client without using a `btw.md` file.
`path_llms_txt`	A path to an `llms.txt` file containing context about the current project. By default, btw will look for an `llms.txt` file in the your current working directory or its parents. Set `path_llms_txt = FALSE` to skip looking for an `llms.txt` file.
`messages`	A list of initial messages to show in the chat, passed to `shinychat::chat_mod_ui()`.

Value

Returns an ellmer::Chat object with additional tools registered from `btw_tools()`. `btw_app()` returns the chat object invisibly, and the chat object with the messages added during the chat session.

Functions

- `btw_client()`: Create a btw-enhanced ellmer::Chat client
- `btw_app()`: Create a btw-enhanced client and launch a Shiny app to chat

Examples

```
withr::local_options(list(
  btw.client = ellmer::chat_ollama(model="llama3.1:8b")
))

chat <- btw_client()
chat$chat(
  "How can I replace `stop()` calls with functions from the cli package?"
)
```

`btw_task_create_btw_md`

Task: Initialize Project Context File

Description

Create a comprehensive context `btw.md` or `AGENTS.md` file for your project. If launched in app or console mode, this task will start an interactive chat session to guide you through the process of creating a context file.

This task focuses on documenting project context for developers and agents. See `btw_client()` for additional details about the format and usage of the `btw.md` context file, including choosing the default LLM provider and model or the default set of tools to use with `btw_client()`.

Usage

```
btw_task_create_btw_md(
  ...,
  path = "btw.md",
  client = NULL,
  mode = c("app", "console", "client", "tool")
)
```

Arguments

`...`	Additional context to provide to the AI. This can be any text or R objects that can be converted to text using `btw()`.
`path`	The path to the context file to create. Defaults to `btw.md`.
`client`	An ellmer::Chat client or a `provider/model` string to be passed to `ellmer::chat()` to create a chat client. Defaults to `ellmer::chat_anthropic()`. You can use the `btw.client` option to set a default client for new `btw_client()` calls, or use a `btw.md` project file for default chat client settings, like provider and model. We check the `client` argument, then the `btw.client` R option, and finally the `btw.md` project file (falling back to user-level `btw.md` if needed), using only the client definition from the first of these that is available.
`mode`	The mode to run the task in, which affects what is returned from this function. `"app"` and `"console"` modes launch interactive sessions, while `"client"` and `"tool"` modes return objects for programmatic use.

Value

When `mode` is `"app"` or `"console"`, this function launches an interactive session in the browser or the R console, respectively. The ellmer chat object with the conversation history is returned invisibly when the session ends.

When `mode` is `"client"`, this function returns the configured ellmer chat client object. When `mode` is `"tool"`, this function returns an ellmer tool object that can be used in other chat instances.

See Also

Other task and agent functions: `btw_task_create_readme()`

Examples

```
withr::with_envvar(list(ANTHROPIC_API_KEY = "example"), {
  btw_task_create_btw_md(mode = "tool", client = "anthropic")
})
```

`btw_task_create_readme`

Task: Create a Polished README

Description

Create a compelling, user-focused README file for your project. If launched in app or console mode, this task will start an interactive chat session to guide you through the process of creating a polished README that clearly communicates value and helps potential users make informed decisions.

This task focuses on creating READMEs for END USERS, not developers, with emphasis on clarity, accessibility, and authentic communication of value. The process involves exploring your project files, understanding your target audience and goals, proposing a structure, and then iteratively drafting each section with your input.

Usage

```
btw_task_create_readme(
  ...,
  client = NULL,
  mode = c("app", "console", "client", "tool")
)
```

Arguments

`...`	Additional context to provide to the AI. This can be any text or R objects that can be converted to text using `btw()`.
`client`	An ellmer::Chat client or a `provider/model` string to be passed to `ellmer::chat()` to create a chat client. Defaults to `ellmer::chat_anthropic()`. You can use the `btw.client` option to set a default client for new `btw_client()` calls, or use a `btw.md` project file for default chat client settings, like provider and model. We check the `client` argument, then the `btw.client` R option, and finally the `btw.md` project file (falling back to user-level `btw.md` if needed), using only the client definition from the first of these that is available.
`mode`	The mode to run the task in, which affects what is returned from this function. `"app"` and `"console"` modes launch interactive sessions, while `"client"` and `"tool"` modes return objects for programmatic use.

Value

When `mode` is `"app"` or `"console"`, this function launches an interactive session in the browser or the R console, respectively. The ellmer chat object with the conversation history is returned invisibly when the session ends.

When `mode` is `"client"`, this function returns the configured ellmer chat client object. When `mode` is `"tool"`, this function returns an ellmer tool object that can be used in other chat instances.

See Also

Other task and agent functions: `btw_task_create_btw_md()`

Examples

```
withr::with_envvar(list(ANTHROPIC_API_KEY = "example"), {
  btw_task_create_readme(mode = "tool", client = "anthropic")
})
```

---

btw_this *Describe something for use by an LLM*

Description

A generic function used to describe an object for use by LLM.

Usage

```
btw_this(x, ...)
```

Arguments

`x`	The thing to describe.
`...`	Additional arguments passed down to underlying methods. Unused arguments are silently ignored.

Value

A character vector of lines describing the object.

See Also

Other btw formatting methods: `btw_this.character()`, `btw_this.data.frame()`, `btw_this.environment()`

Examples

```
btw_this(mtcars) # describe the mtcars dataset
btw_this(dplyr::mutate) # include function source
```

`btw_this.character` *Describe objects*

Description

Character strings in `btw_this()` are used as shortcuts to many underlying methods. `btw_this()` detects specific formats in the input string to determine which method to call, or by default it will try to evaluate the character string as R code and return the appropriate object description.

`btw_this()` knows about the following special character string formats:

- `"./path"`
  Any string starting with `./` is treated as a relative path. If the path is a file, we call `btw_tool_files_read_text_file()` and if the path is a directory we call `btw_tool_files_list_files()` on the path.
  - `btw_this("./data")` lists the files in `data/`.
  - `btw_this("./R/load_data.R")` reads the source of the `R/load_data.R` file.
- `"{pkgName}"` or `"@pkg pkgName"`
  A package name wrapped in braces, or using the `@pkg` command. Returns the list of help topics (`btw_tool_docs_package_help_topics()`) and, if it exists, the introductory vignette for the package (`btw_tool_docs_vignette()`).
  - `btw_this("{dplyr}")` or `btw_this("@pkg dplyr")` includes dplyr's introductory vignette.
  - `btw_this("{btw}")` returns only the package help index (because `btw` doesn't have an intro vignette, yet).
- `"?help_topic"` or `"@help topic"`
  When the string starts with `?` or `@help`, btw searches R's help topics using `btw_tool_docs_help_page()`. Supports multiple formats:
  - `btw_this("?dplyr::across")` or `btw_this("@help dplyr::across")`
  - `btw_this("@help dplyr across")` - space-separated format
  - `btw_this("@help across")` - searches all packages
- `"@news {{package_name}} {{search_term}}"`
  Include the release notes (NEWS) from the latest package release, e.g. `"@news dplyr"`, or that match a search term, e.g. `"@news dplyr join_by"`.
- `"@url {{url}}"`
  Include the contents of a web page at the specified URL as markdown, e.g. `"@url https://cran.r-project.org/doc/`
  Requires the **chromote** package to be installed.
- `"@git status"`, `"@git diff"`, `"@git log"`
  Git commands for viewing repository status, diffs, and commit history. Requires **gert** package and a git repository.
  - `btw_this("@git status")` - show working directory status
  - `btw_this("@git status staged")` - show only staged files
  - `btw_this("@git diff")` - show unstaged changes
  - `btw_this("@git diff HEAD")` - show staged changes

  - `btw_this("@git log")` - show recent commits (default 10)
  - `btw_this("@git log main 20")` - show 20 commits from main branch

- `"@issue #number"` or `"@pr #number"`
  Fetch a GitHub issue or pull request. Automatically detects the current repository, or you can specify `owner/repo#number` or `owner/repo number`. Requires **gh** package and GitHub authentication.
  - `btw_this("@issue #65")` - issue from current repo
  - `btw_this("@pr posit-dev/btw#64")` - PR from specific repo
  - `btw_this("@issue tidyverse/dplyr 1234")` - space-separated format
- `"@current_file"` or `"@current_selection"`
  When used in RStudio or Positron, or anywhere else that the **rstudioapi** is supported, `btw("@current_file")` includes the contents of the file currently open in the editor using `rstudioapi::getSourceEditorContext()`.
- `"@clipboard"`
  Includes the contents currently stored in your clipboard.
- `"@platform_info"`
  Includes information about the current platform, such as the R version, operating system, IDE or UI being used, as well as language, locale, timezone and current date.
- `"@attached_packages"`, `"@loaded_packages"`, `"@installed_packages"`
  Includes information about the attached, loaded, or installed packages in your R session, using `sessioninfo::package_info()`.
- `"@last_error"`
  Includes the message from the last error that occurred in your session. To reliably capture the last error, you need to enable `rlang::global_entrace()` in your session.
- `"@last_value"`
  Includes the `.Last.value`, i.e. the result of the last expression evaluated in your R console.

Usage

```
## S3 method for class 'character'
btw_this(x, ..., caller_env = parent.frame())
```

Arguments

`x`	A character string
`...`	Ignored.
`caller_env`	The caller environment.

Value

A character vector of lines describing the object.

See Also

Other btw formatting methods: `btw_this()`, `btw_this.data.frame()`, `btw_this.environment()`

Examples

```
mtcars[1:3, 1:4]
cat(btw_this("@last_value"))
```

`btw_this.data.frame` *Describe a data frame in plain text*

Description

Describe a data frame in plain text

Usage

```
## S3 method for class 'data.frame'
btw_this(
  x,
  ...,
  format = c("skim", "glimpse", "print", "json"),
  max_rows = 5,
  max_cols = 100,
  package = NULL
)

## S3 method for class 'tbl'
btw_this(
  x,
  ...,
  format = c("skim", "glimpse", "print", "json"),
  max_rows = 5,
  max_cols = 100,
  package = NULL
)
```

Arguments

`x`	A data frame or tibble.
`...`	Additional arguments are silently ignored.
`format`	One of `"skim"`, `"glimpse"`, `"print"`, or `"json"`.

- `"skim"` is the most information-dense format for describing the data. It uses and returns the same information as `skimr::skim()` but formatting as a JSON object that describes the dataset.
- To glimpse the data column-by-column, use `"glimpse"`. This is particularly helpful for getting a sense of data frame column names, types, and distributions, when pairings of entries in individual rows aren't particularly important.

- To just print out the data frame, use `print()`.
- To get a json representation of the data, use `"json"`. This is particularly helpful when the pairings among entries in specific rows are important to demonstrate.

`max_rows` The maximum number of rows to show in the data frame. Only applies when `format = "json"`.

`max_cols` The maximum number of columns to show in the data frame. Only applies when `format = "json"`.

`package` The name of the package that provides the data set. If not provided, `data_frame` must be loaded in the current environment, or may also be inferred from the name of the data frame, e.g. `"dplyr::storms"`.

Value

A character vector containing a representation of the data frame. Will error if the named data frame is not found in the environment.

Functions

- `btw_this(data.frame)`: Summarize a data frame.
- `btw_this(tbl)`: Summarize a `tbl`.

See Also

`btw_tool_env_describe_data_frame()`

Other btw formatting methods: `btw_this()`, `btw_this.character()`, `btw_this.environment()`

Other btw formatting methods: `btw_this()`, `btw_this.character()`, `btw_this.environment()`

Examples

```
btw_this(mtcars)

btw_this(mtcars, format = "print")

btw_this(mtcars, format = "json")
```

---

btw_this.environment *Describe the contents of an environment*

Description

Describe the contents of an environment

Usage

```
## S3 method for class 'environment'
btw_this(x, ..., items = NULL)
```

Arguments

x	An environment.
...	Additional arguments are silently ignored.
items	Optional. A character vector of objects in the environment to describe.

Value

A string describing the environment contents with #> prefixing each object's printed representation.

See Also

btw_tool_env_describe_environment()

Other btw formatting methods: btw_this(), btw_this.character(), btw_this.data.frame()

Examples

```
env <- new.env()
env$cyl_6 <- mtcars[mtcars$cyl == 6, ]
env$gear_5 <- mtcars[mtcars$gear == 5, ]
btw_this(env)
```

---

btw_tools *Tools: Register tools from btw*

Description

The btw_tools() function provides a list of tools that can be registered with an ellmer chat via chat$set_tools() that allow the chat to interface with your computational environment. Chats returned by this function have access to the tools:

Group: docs:

Name	Description
btw_tool_docs_available_vignettes()	List available vignettes for an R package.
btw_tool_docs_help_page()	Get help page from package.
btw_tool_docs_package_help_topics()	Get available help topics for an R package.
btw_tool_docs_package_news()	Read the release notes (NEWS) for a package.
btw_tool_docs_vignette()	Get a package vignette in plain text.

Group: env:

Name	Description
btw_tool_env_describe_data_frame()	Show the data frame or table or get information about the structure of a data fra
btw_tool_env_describe_environment()	List and describe items in the R session's global environment.

Group: files:

Name	Description
btw_tool_files_code_search()	Search code files in the project.
btw_tool_files_list_files()	List files or directories in the project.
btw_tool_files_read_text_file()	Read an entire text file.
btw_tool_files_write_text_file()	Write content to a text file.

Group: git:

Name	Description
btw_tool_git_branch_checkout()	Switch to a different git branch.
btw_tool_git_branch_create()	Create a new git branch.
btw_tool_git_branch_list()	List git branches in the repository.
btw_tool_git_commit()	Stage files and create a git commit.
btw_tool_git_diff()	View changes in the working directory or a commit.
btw_tool_git_log()	Show the commit history for a repository.
btw_tool_git_status()	Show the status of the git working directory.

Group: github:

Name	Description
btw_tool_github()	Execute R code that calls the GitHub API using gh().

Group: ide:

Name	Description
btw_tool_ide_read_current_editor()	Read the contents of the editor that is currently open in the user's IDE.

Group: search:

Name	Description
btw_tool_search_package_info()	Describe a CRAN package.
btw_tool_search_packages()	Search for an R package on CRAN.

Group: session:

Name	Description
btw_tool_session_check_package_installed()	Check if a package is installed in the current session.
btw_tool_session_package_info()	Verify that a specific package is installed, or find out which packages a
btw_tool_session_platform_info()	Describes the R version, operating system, language and locale settings

Group: web:

Name	Description
btw_tool_web_read_url()	Read a web page and convert it to Markdown format.

Usage

```
btw_tools(...)
```

Arguments

...	Optional names of tools or tool groups to include when registering tools. By default all btw tools are included. For example, use "docs" to include only the documentation related tools, or "env", "docs", "session" for the collection of environment, documentation and session tools, and so on. The names provided can be: 1. The name of a tool, such as "btw_tool_env_describe_data_frame". 2. The name of a tool group, such as "env", which will include all tools in that group. 3. The tool name without the btw_tool_ prefix, such as "env_describe_data_frame".

Value

Registers the tools with chat, updating the chat object in place. The chat input is returned invisibly.

See Also

Other Tools: btw_tool_docs_package_news(), btw_tool_env_describe_data_frame(), btw_tool_env_describe_env
btw_tool_files_code_search(), btw_tool_files_list_files(), btw_tool_files_read_text_file(),
btw_tool_files_write_text_file(), btw_tool_ide_read_current_editor(), btw_tool_package_docs,
btw_tool_search_packages(), btw_tool_session_package_info(), btw_tool_session_platform_info(),
btw_tool_web_read_url()

Examples

```
# requires an ANTHROPIC_API_KEY
ch <- ellmer::chat_anthropic()

# register all of the available tools
ch$set_tools(btw_tools())
```

```
# or register only the tools related to fetching documentation
ch$set_tools(btw_tools("docs"))

# ensure that the current tools persist
ch$set_tools(c(ch$get_tools(), btw_tools()))
```

`btw_tool_docs_package_news`

Tool: Package Release Notes

Description

Include release notes for a package, either the release notes for the most recent package release or release notes matching a search term.

Usage

```
btw_tool_docs_package_news(package_name, search_term = "", `_intent` = "")
```

Arguments

`package_name`	The name of the package as a string, e.g. `"shiny"`.
`search_term`	A regular expression to search for in the NEWS entries. If empty, the release notes of the current installed version is included.
`_intent`	An optional string describing the intent of the tool use. When the tool is used by an LLM, the model will use this argument to explain why it called the tool.

Value

Returns the release notes for the currently installed version of the package, or the release notes matching the search term.

See Also

`btw_tools()`

Other Tools: `btw_tool_env_describe_data_frame()`, `btw_tool_env_describe_environment()`, `btw_tool_files_code_search()`, `btw_tool_files_list_files()`, `btw_tool_files_read_text_file()`, `btw_tool_files_write_text_file()`, `btw_tool_ide_read_current_editor()`, `btw_tool_package_docs`, `btw_tool_search_packages()`, `btw_tool_session_package_info()`, `btw_tool_session_platform_info()`, `btw_tool_web_read_url()`, `btw_tools()`

Examples

```
# Copy release notes to the clipboard for use in any AI app
btw("@news dplyr", clipboard = FALSE)

btw("@news dplyr join_by", clipboard = FALSE)

if (interactive()) { # can be slow
  if (R.version$major == 4 && R.version$minor > "2.0") {
    # Search through R's release notes.
    # This should find a NEWS entry from R 4.2
    btw("@news R dynamic rd content", clipboard = FALSE)
  }
}

# Tool use by LLMs via ellmer or MCP ----
btw_tool_docs_package_news("dplyr")

btw_tool_docs_package_news("dplyr", "join_by")
```

---

btw_tool_env_describe_data_frame

Tool: Describe data frame

Description

Tool: Describe data frame

Usage

```
btw_tool_env_describe_data_frame(
  data_frame,
  format = c("skim", "glimpse", "print", "json"),
  max_rows = 5,
  max_cols = 100,
  package = NULL,
  `_intent` = ""
)
```

Arguments

`data_frame`	The data frame to describe
`format`	One of `"skim"`, `"glimpse"`, `"print"`, or `"json"`.

- `"skim"` is the most information-dense format for describing the data. It uses and returns the same information as `skimr::skim()` but formatting as a JSON object that describes the dataset.

- To glimpse the data column-by-column, use `"glimpse"`. This is particularly helpful for getting a sense of data frame column names, types, and distributions, when pairings of entries in individual rows aren't particularly important.
- To just print out the data frame, use `print()`.
- To get a json representation of the data, use `"json"`. This is particularly helpful when the pairings among entries in specific rows are important to demonstrate.

`max_rows` The maximum number of rows to show in the data frame. Only applies when `format = "json"`.

`max_cols` The maximum number of columns to show in the data frame. Only applies when `format = "json"`.

`package` The name of the package that provides the data set. If not provided, `data_frame` must be loaded in the current environment, or may also be inferred from the name of the data frame, e.g. `"dplyr::storms"`.

`_intent` An optional string describing the intent of the tool use. When the tool is used by an LLM, the model will use this argument to explain why it called the tool.

Value

A character vector containing a representation of the data frame. Will error if the named data frame is not found in the environment.

See Also

`btw_this.data.frame()`, `btw_tools()`

Other Tools: `btw_tool_docs_package_news()`, `btw_tool_env_describe_environment()`, `btw_tool_files_code_sear`
`btw_tool_files_list_files()`, `btw_tool_files_read_text_file()`, `btw_tool_files_write_text_file()`,
`btw_tool_ide_read_current_editor()`, `btw_tool_package_docs`, `btw_tool_search_packages()`,
`btw_tool_session_package_info()`, `btw_tool_session_platform_info()`, `btw_tool_web_read_url()`,
`btw_tools()`

Examples

```
btw_tool_env_describe_data_frame(mtcars)
```

---

btw_tool_env_describe_environment

Tool: Describe an environment

Description

This tool can be used by the LLM to describe the contents of an R session, i.e. the data frames and other objects loaded into the global environment. This tool will only see variables that you've named and created in the global environment, it cannot reach into package namespaces, see which packages you have loaded, or access files on your computer.

Usage

```
btw_tool_env_describe_environment(items = NULL, `_intent` = "")
```

Arguments

items	Optional. A character vector of objects in the environment to describe.
_intent	An optional string describing the intent of the tool use. When the tool is used by an LLM, the model will use this argument to explain why it called the tool.

Value

A string describing the environment contents with #> prefixing each object's printed representation.

See Also

btw_this.environment(), btw_tools()

Other Tools: btw_tool_docs_package_news(), btw_tool_env_describe_data_frame(), btw_tool_files_code_searc
btw_tool_files_list_files(), btw_tool_files_read_text_file(), btw_tool_files_write_text_file(),
btw_tool_ide_read_current_editor(), btw_tool_package_docs, btw_tool_search_packages(),
btw_tool_session_package_info(), btw_tool_session_platform_info(), btw_tool_web_read_url(),
btw_tools()

Examples

```
my_cars <- mtcars[mtcars$mpg > 25, ]
btw_tool_env_describe_environment("my_cars")
```

---

btw_tool_files_code_search

Tool: Code Search in Project

Description

Search through code files in the project directory for specific terms.

Usage

```
btw_tool_files_code_search(
  term,
  limit = 100,
  case_sensitive = TRUE,
  use_regex = FALSE,
  show_lines = FALSE,
  `_intent` = ""
)
```

Arguments

term	The term to search for in the code files.
limit	Maximum number of matching lines to return (between 1 and 1000, default 100).
case_sensitive	Whether the search should be case-sensitive (default is FALSE).
use_regex	Whether to interpret the search term as a regular expression (default is FALSE).
show_lines	Whether to show the matching lines in the results. Defaults to FALSE, which means only the file names and count of matching lines are returned.
_intent	An optional string describing the intent of the tool use. When the tool is used by an LLM, the model will use this argument to explain why it called the tool.

Details

Options:

You can configure which file extensions are included and which paths are excluded from code search by using two options:

- `btw.files_code_search.extensions`: A character vector of file extensions to search in (default includes R, Python, JavaScript, TypeScript, Markdown, SCSS, and CSS files).
- `btw.files_code_search.exclusions`: A character vector of gitignore-style patterns to exclude paths and directories from the search. The default value includes a set of common version control, IDE, and cache folders.

Alternatively, you can also set these options in your `btw.md` file under the `options` section, like this:

```
---
client:
  provider: anthropic
tools: [files_code_search]
options:
  files_code_search:
    extensions: ["R", "Rmd", "py", "qmd"]
    exclusions: ["DEFAULT", ".quarto/"]
---
```

Include `"DEFAULT"` in the `exclusions` option to use btw's default exclusions, which cover common directories like `.git/`, `.vscode/`.

If the **gert** package is installed and the project is a Git repository, the tool will also respect the `.gitignore` file and exclude any ignored paths, regardless of the `btw.files_code_search.exclusions` option.

Value

Returns a tool result with a data frame of search results, with columns for `filename`, `size`, `last_modified`, `content` and `line`.

See Also

Other Tools: btw_tool_docs_package_news(), btw_tool_env_describe_data_frame(), btw_tool_env_describe_env
btw_tool_files_list_files(), btw_tool_files_read_text_file(), btw_tool_files_write_text_file(),
btw_tool_ide_read_current_editor(), btw_tool_package_docs, btw_tool_search_packages(),
btw_tool_session_package_info(), btw_tool_session_platform_info(), btw_tool_web_read_url(),
btw_tools()

Examples

```
withr::with_tempdir({
  writeLines(state.name[1:25], "state_names_1.md")
  writeLines(state.name[26:50], "state_names_2.md")

  tools <- btw_tools("files_code_search")
  tools$btw_tool_files_code_search(
    term = "kentucky",
    case_sensitive = FALSE,
    show_lines = TRUE
  )
})
```

---

btw_tool_files_list_files

Tool: List files

Description

Tool: List files

Usage

```
btw_tool_files_list_files(
  path = NULL,
  type = c("any", "file", "directory"),
  regexp = "",
  `_intent` = ""
)
```

Arguments

path	Path to a directory or file for which to get information. The path must be in the current working directory. If path is a directory, we use fs::dir_info() to list information about files and directories in path (use type to pick only one or the other). If path is a file, we show information about that file.
type	File type(s) to return, one of "any" or "file" or "directory".
regexp	A regular expression (e.g. [.]csv$) passed on to grep() to filter paths.
_intent	An optional string describing the intent of the tool use. When the tool is used by an LLM, the model will use this argument to explain why it called the tool.

Value

Returns a character table of file information.

See Also

Other Tools: btw_tool_docs_package_news(), btw_tool_env_describe_data_frame(), btw_tool_env_describe_env
btw_tool_files_code_search(), btw_tool_files_read_text_file(), btw_tool_files_write_text_file(),
btw_tool_ide_read_current_editor(), btw_tool_package_docs, btw_tool_search_packages(),
btw_tool_session_package_info(), btw_tool_session_platform_info(), btw_tool_web_read_url(),
btw_tools()

Examples

```
withr::with_tempdir({
  write.csv(mtcars, "mtcars.csv")

  btw_tool_files_list_files(type = "file")
})
```

---

btw_tool_files_read_text_file

Tool: Read a file

Description

Tool: Read a file

Usage

```
btw_tool_files_read_text_file(
  path,
  line_start = 1,
  line_end = 1000,
  `_intent` = ""
)
```

Arguments

path	Path to a file for which to get information. The path must be in the current working directory.
line_start	Starting line to read, defaults to 1 (starting from the first line).
line_end	Ending line to read, defaults to 1000. Change only this value if you want to read more or fewer lines. Use in combination with line_start to read a specific line range of the file.
_intent	An optional string describing the intent of the tool use. When the tool is used by an LLM, the model will use this argument to explain why it called the tool.

Value

Returns a character vector of lines from the file.

See Also

Other Tools: btw_tool_docs_package_news(), btw_tool_env_describe_data_frame(), btw_tool_env_describe_env
btw_tool_files_code_search(), btw_tool_files_list_files(), btw_tool_files_write_text_file(),
btw_tool_ide_read_current_editor(), btw_tool_package_docs, btw_tool_search_packages(),
btw_tool_session_package_info(), btw_tool_session_platform_info(), btw_tool_web_read_url(),
btw_tools()

Examples

```
withr::with_tempdir({
  write.csv(mtcars, "mtcars.csv")

  btw_tool_files_read_text_file("mtcars.csv", line_end = 5)
})
```

---

btw_tool_files_write_text_file

Tool: Write a text file

Description

Tool: Write a text file

Usage

```
btw_tool_files_write_text_file(path, content, `_intent` = "")
```

Arguments

path	Path to the file to write. The path must be in the current working directory.
content	The text content to write to the file. This should be the complete content as the file will be overwritten.
_intent	An optional string describing the intent of the tool use. When the tool is used by an LLM, the model will use this argument to explain why it called the tool.

Value

Returns a message confirming the file was written.

See Also

Other Tools: btw_tool_docs_package_news(), btw_tool_env_describe_data_frame(), btw_tool_env_describe_env
btw_tool_files_code_search(), btw_tool_files_list_files(), btw_tool_files_read_text_file(),
btw_tool_ide_read_current_editor(), btw_tool_package_docs, btw_tool_search_packages(),
btw_tool_session_package_info(), btw_tool_session_platform_info(), btw_tool_web_read_url(),
btw_tools()

Examples

```
withr::with_tempdir({
  btw_tool_files_write_text_file("example.txt", "Hello\nWorld!")
  readLines("example.txt")
})
```

---

btw_tool_github *Tool: GitHub*

Description

Execute R code that calls the GitHub API using gh::gh().

This tool is designed such that models can write very limited R code to call gh::gh() and protections are inserted to prevent the model from calling unsafe or destructive actions via the API. The **Endpoint Validation** section below describes how API endpoints are validated to ensure safety.

While this tool *can* execute R code, the code is evaluated in an environment where only a limited set of functions and variables are available. In particular, only the gh() and gh_whoami() functions from the gh package are available, along with owner and repo variables that are pre-defined to point to the current repository (if detected). This allows models to focus on writing GitHub API calls without needing to load packages or manage authentication.

Endpoint Validation:

This tool uses endpoint validation to ensure only safe GitHub API operations are performed. By default, most read operations and low-risk write operations (like creating issues or PRs) are allowed, while dangerous operations (like merging PRs or deleting repositories) are blocked.

To customize which endpoints are allowed or blocked, use the btw.github.allow and btw.github.block options:

```
# Allow a specific endpoint
options(btw.github.allow = c(
  getOption("btw.github.allow"),
  "GET /repos/*/*/topics"
))

# Block a specific endpoint
options(btw.github.block = c(
  getOption("btw.github.block"),
  "GET /repos/*/*/branches"
))
```

You can also set these options in your btw.md file under the options field:

```
tools: github
options:
  github:
    allow:
      - "PATCH /repos/*/*/pulls/*" # Allow converting PRs to/from draft
      - "POST /repos/*/*/git/refs" # Allow creating branches
    block:
      - "DELETE /repos/**" # Block any delete action under /repos
```

The precedence order for rules is:

1. User block rules (checked first, highest priority)
2. User allow rules
3. Built-in block rules
4. Built-in allow rules
5. Default: reject (if no rules match)

Additional Examples:

```
# Get an issue
btw_tool_github(
  code = 'gh("/repos/{owner}/{repo}/issues/123", owner = owner, repo = repo)'
)

# Create an issue
btw_tool_github(code = r"(
  gh(
    "POST /repos/{owner}/{repo}/issues",
    title = \"Bug report\",
    body = \"Description of bug\",
    owner = owner,
    repo = repo
  )
)")

# Target a different repository
btw_tool_github(code = 'gh("/repos/tidyverse/dplyr/issues/123")')
```

Usage

```
btw_tool_github(code, fields = "default", `_intent` = "")
```

Arguments

code	R code that calls gh() or gh_whoami(). The code will be evaluated in an environment where owner and repo variables are predefined (defaulting to the current repository if detected). The gh() function is available without needing to load the gh package.

fields	Optional character vector of GitHub API response fields to retain. If provided, only these fields will be included in the result. Defaults to a curated set of commonly used fields.
_intent	An optional string describing the intent of the tool use. When the tool is used by an LLM, the model will use this argument to explain why it called the tool.

Value

A btw_tool_result containing the result of the GitHub API call.

Examples

```
# This tool requires the gh package and authentication to GitHub.
# See additional examples in the documentation above.
```

---

btw_tool_git_branch_checkout

Tool: Git Branch Checkout

Description

Allows an LLM to switch to a different git branch using gert::git_branch_checkout(), equivalent to git checkout <branch> in the terminal.

Usage

```
btw_tool_git_branch_checkout(branch, force = FALSE, `_intent` = "")
```

Arguments

branch	Name of branch to check out.
force	Whether to force checkout even with uncommitted changes. Defaults to FALSE.
_intent	An optional string describing the intent of the tool use. When the tool is used by an LLM, the model will use this argument to explain why it called the tool.

Value

Returns a confirmation message.

See Also

Other git tools: btw_tool_git_branch_create(), btw_tool_git_branch_list(), btw_tool_git_commit(), btw_tool_git_diff(), btw_tool_git_log(), btw_tool_git_status()

Examples

```
withr::with_tempdir({
  gert::git_init()
  gert::git_config_set("user.name", "R Example")
  gert::git_config_set("user.email", "ex@example.com")

  fs::file_touch("hello.md")

  gert::git_add("hello.md")
  gert::git_commit("Initial commit")

  gert::git_branch_create("feature-1")

  # LLM checks out an existing branch
  res <- btw_tool_git_branch_checkout(branch = "feature-1")

  # What the LLM sees
  cat(res@value)
})
```

---

btw_tool_git_branch_create

Tool: Git Branch Create

Description

Allows an LLM to create a new git branch using `gert::git_branch_create()`, equivalent to `git branch <branch>` in the terminal.

Usage

```
btw_tool_git_branch_create(
  branch,
  ref = "HEAD",
  checkout = TRUE,
  `_intent` = ""
)
```

Arguments

branch	Name of the new branch to create.
ref	Optional reference point for the new branch. Defaults to "HEAD".
checkout	Whether to check out the new branch after creation. Defaults to TRUE.
_intent	An optional string describing the intent of the tool use. When the tool is used by an LLM, the model will use this argument to explain why it called the tool.

Value

Returns a confirmation message.

See Also

Other git tools: btw_tool_git_branch_checkout(), btw_tool_git_branch_list(), btw_tool_git_commit(), btw_tool_git_diff(), btw_tool_git_log(), btw_tool_git_status()

Examples

```
withr::with_tempdir({
  gert::git_init()
  gert::git_config_set("user.name", "R Example")
  gert::git_config_set("user.email", "ex@example.com")

  fs::file_touch("hello.md")
  gert::git_add("hello.md")
  gert::git_commit("Initial commit")

  # LLM creates a new branch
  res <- btw_tool_git_branch_create(branch = "feature/new-analysis")

  # What the LLM sees
  cat(res@value)
})
```

---

btw_tool_git_branch_list

Tool: Git Branch List

Description

This tool allows an LLM to list git branches in the repository using gert::git_branch_list(), equivalent to `git branch` in the terminal.

Usage

```
btw_tool_git_branch_list(include = c("local", "remote", "all"), `_intent` = "")
```

Arguments

include	Once of "local" (default), "remote", or "all" to filter branches to local branches only, remote branches only, or all branches.
_intent	An optional string describing the intent of the tool use. When the tool is used by an LLM, the model will use this argument to explain why it called the tool.

Value

Returns a character table of branches.

See Also

Other git tools: btw_tool_git_branch_checkout(), btw_tool_git_branch_create(), btw_tool_git_commit(), btw_tool_git_diff(), btw_tool_git_log(), btw_tool_git_status()

Examples

```
withr::with_tempdir({
  gert::git_init()
  gert::git_config_set("user.name", "R Example")
  gert::git_config_set("user.email", "ex@example.com")

  fs::file_touch("hello.md")
  gert::git_add("hello.md")
  gert::git_commit("Initial commit")

  gert::git_branch_create("feature-1")
  gert::git_branch_create("feature-2")

  # What the LLM sees
  cat(btw_tool_git_branch_list()@value)
})
```

---

btw_tool_git_commit *Tool: Git Commit*

Description

This tool allows an LLM stage files and create a git commit. This tool uses a combination of gert::git_add() to stage files and gert::git_commit() to commit them, which is equivalent to `git add` and `git commit` in the terminal, respectively.

Usage

```
btw_tool_git_commit(message, files = NULL, `_intent` = "")
```

Arguments

message	A commit message describing the changes.
files	Optional character vector of file paths to stage and commit. Use "." to stage all changed files. If NULL, commits currently staged files.
_intent	An optional string describing the intent of the tool use. When the tool is used by an LLM, the model will use this argument to explain why it called the tool.

Value

Returns the commit SHA.

See Also

Other git tools: btw_tool_git_branch_checkout(), btw_tool_git_branch_create(), btw_tool_git_branch_list(), btw_tool_git_diff(), btw_tool_git_log(), btw_tool_git_status()

Examples

```
withr::with_tempdir({
  gert::git_init()
  gert::git_config_set("user.name", "R Example")
  gert::git_config_set("user.email", "ex@example.com")

  writeLines("hello, world", "hello.md")

  res <- btw_tool_git_commit("Initial commit", files = "hello.md")

  # What the LLM sees
  cat(res@value)
})
```

---

btw_tool_git_diff *Tool: Git Diff*

Description

This tool allows an LLM to run gert::git_diff_patch(), equivalent to `git diff` in the terminal, and to see the detailed changes made in a commit.

Usage

```
btw_tool_git_diff(ref = NULL, `_intent` = "")
```

Arguments

ref	a reference such as "HEAD", or a commit id, or NULL to the diff the working directory against the repository index.
_intent	An optional string describing the intent of the tool use. When the tool is used by an LLM, the model will use this argument to explain why it called the tool.

Value

Returns a diff patch as a formatted string.

See Also

Other git tools: btw_tool_git_branch_checkout(), btw_tool_git_branch_create(), btw_tool_git_branch_list(), btw_tool_git_commit(), btw_tool_git_log(), btw_tool_git_status()

Examples

```
withr::with_tempdir({
  gert::git_init()
  gert::git_config_set("user.name", "R Example")
  gert::git_config_set("user.email", "ex@example.com")

  writeLines("hello, world", "hello.md")
  gert::git_add("hello.md")
  gert::git_commit("Initial commit")

  writeLines("hello, universe", "hello.md")

  # What the LLM sees
  cat(btw_tool_git_diff()@value)
})
```

---

btw_tool_git_log *Tool: Git Log*

Description

This tool allows an LLM to run gert::git_log(), equivalent to `git log` in the terminal, and to see the commit history of a repository.

Usage

```
btw_tool_git_log(ref = "HEAD", max = 10, after = NULL, `_intent` = "")
```

Arguments

ref	Revision string with a branch/tag/commit value. Defaults to "HEAD".
max	Maximum number of commits to retrieve. Defaults to 10.
after	Optional date or timestamp: only include commits after this date.
_intent	An optional string describing the intent of the tool use. When the tool is used by an LLM, the model will use this argument to explain why it called the tool.

Value

Returns a character table of commit history.

See Also

Other git tools: btw_tool_git_branch_checkout(), btw_tool_git_branch_create(), btw_tool_git_branch_list(), btw_tool_git_commit(), btw_tool_git_diff(), btw_tool_git_status()

Examples

```
withr::with_tempdir({
  gert::git_init()
  gert::git_config_set("user.name", "R Example")
  gert::git_config_set("user.email", "ex@example.com")

  writeLines("hello, world", "hello.md")
  gert::git_add("hello.md")
  gert::git_commit("Initial commit")

  writeLines("hello, universe", "hello.md")
  gert::git_add("hello.md")
  gert::git_commit("Update hello.md")

  # What the LLM sees
  cat(btw_tool_git_log()@value)
})
```

---

btw_tool_git_status *Tool: Git Status*

Description

This tool allows the LLM to run gert::git_status(), equivalent to `git status` in the terminal, and to see the current status of the working directory.

Usage

```
btw_tool_git_status(
  include = c("both", "staged", "unstaged"),
  pathspec = NULL,
  `_intent` = ""
)
```

Arguments

include	One of "both", "staged", or "unstaged". Use "both" to show both staged and unstaged files (default).
pathspec	Optional character vector with paths to match.
_intent	An optional string describing the intent of the tool use. When the tool is used by an LLM, the model will use this argument to explain why it called the tool.

Value

Returns a character table of file statuses.

See Also

Other git tools: btw_tool_git_branch_checkout(), btw_tool_git_branch_create(), btw_tool_git_branch_list(), btw_tool_git_commit(), btw_tool_git_diff(), btw_tool_git_log()

Examples

```
withr::with_tempdir({
  gert::git_init()
  gert::git_config_set("user.name", "R Example")
  gert::git_config_set("user.email", "ex@example.com")

  writeLines("hello, world", "hello.md")

  # What the LLM sees
  cat(btw_tool_git_status()@value)
})
```

---

btw_tool_ide_read_current_editor

Tool: Read current file

Description

Reads the current file using the **rstudioapi**, which works in RStudio, Positron and VS Code (with the vscode-r extension).

Usage

```
btw_tool_ide_read_current_editor(
  selection = TRUE,
  consent = FALSE,
  `_intent` = ""
)
```

Arguments

selection	Should only the selected text be included? If no text is selected, the full file contents are returned.
consent	Boolean indicating whether the user has consented to reading the current file. The tool definition includes language to induce LLMs to confirm with the user before calling the tool. Not all models will follow these instructions. Users can also include the string `@current_file` to induce the tool.
_intent	An optional string describing the intent of the tool use. When the tool is used by an LLM, the model will use this argument to explain why it called the tool.

Value

Returns the contents of the current editor.

See Also

Other Tools: btw_tool_docs_package_news(), btw_tool_env_describe_data_frame(), btw_tool_env_describe_env
btw_tool_files_code_search(), btw_tool_files_list_files(), btw_tool_files_read_text_file(),
btw_tool_files_write_text_file(), btw_tool_package_docs, btw_tool_search_packages(),
btw_tool_session_package_info(), btw_tool_session_platform_info(), btw_tool_web_read_url(),
btw_tools()

Examples

```
btw_tool_ide_read_current_editor(consent = TRUE)
```

---

btw_tool_package_docs *Tool: Describe R package documentation*

Description

These functions describe package documentation in plain text.

Additional Examples:

Show a list of available vignettes in the dplyr package:

```
btw_tool_docs_available_vignettes("dplyr")
```

Get the introductory vignette for the dplyr package:

```
btw_tool_docs_vignette("dplyr")
```

Get a specific vignette, such as the programming vignette for the dplyr package:

```
btw_tool_docs_vignette("dplyr", "programming")
```

Usage

```
btw_tool_docs_package_help_topics(package_name, `_intent` = "")

btw_tool_docs_help_page(topic, package_name = "", `_intent` = "")

btw_tool_docs_available_vignettes(package_name, `_intent` = "")

btw_tool_docs_vignette(package_name, vignette = package_name, `_intent` = "")
```

Arguments

package_name	The name of the package as a string, e.g. "shiny".
_intent	An optional string describing the intent of the tool use. When the tool is used by an LLM, the model will use this argument to explain why it called the tool.
topic	The topic_id or alias of the help page, e.g. "withProgress" or "incProgress". Find topic_ids or aliases using get_package_help().
vignette	The name (or index) of the vignette to retrieve. Defaults to the "intro" vignette to the package (by the same rules as pkgdown.)

Value

- btw_tool_docs_package_help_topics() returns the topic_id, title, and aliases fields for every topic in a package's documentation as a json-formatted string.
- btw_tool_docs_help_page() returns the help-page for a package topic as a string.

See Also

btw_tools()

Other Tools: btw_tool_docs_package_news(), btw_tool_env_describe_data_frame(), btw_tool_env_describe_env
btw_tool_files_code_search(), btw_tool_files_list_files(), btw_tool_files_read_text_file(),
btw_tool_files_write_text_file(), btw_tool_ide_read_current_editor(), btw_tool_search_packages(),
btw_tool_session_package_info(), btw_tool_session_platform_info(), btw_tool_web_read_url(),
btw_tools()

Examples

```
btw_tool_docs_package_help_topics("btw")

btw_tool_docs_help_page("btw", "btw")
```

---

btw_tool_search_packages

Tool: Search for an R package on CRAN

Description

Uses pkgsearch::pkg_search() to search for R packages on CRAN.

Usage

```
btw_tool_search_packages(
  query,
  format = c("short", "long"),
  n_results = NULL,
  `_intent` = ""
)
```

Arguments

query	Search query string. If this argument is missing or NULL, then the results of the last query are printed, in *short* and *long* formats, in turns for successive pkg_search() calls. If this argument is missing, then all other arguments are ignored.
format	Default formatting of the results. *short* only outputs the name and title of the packages, *long* also prints the author, last version, full description and URLs. Note that this only affects the default printing, and you can still inspect the full results, even if you specify *short* here.
n_results	Number of search results to include. Defaults to 10 for 'short' format and 5 for 'long' format.
_intent	An optional string describing the intent of the tool use. When the tool is used by an LLM, the model will use this argument to explain why it called the tool.

Value

A listing of packages matching the search term.

See Also

btw_tools()

Other Tools: btw_tool_docs_package_news(), btw_tool_env_describe_data_frame(), btw_tool_env_describe_env
btw_tool_files_code_search(), btw_tool_files_list_files(), btw_tool_files_read_text_file(),
btw_tool_files_write_text_file(), btw_tool_ide_read_current_editor(), btw_tool_package_docs,
btw_tool_session_package_info(), btw_tool_session_platform_info(), btw_tool_web_read_url(),
btw_tools()

Examples

```
# Copy pkgsearch results to the clipboard for use in any LLM app
btw(
  pkgsearch::pkg_search("network visualization", size = 1),
  clipboard = FALSE
)
btw(
  pkgsearch::pkg_search("network visualization", format = "long", size = 1),
  clipboard = FALSE
)
```

---

btw_tool_search_package_info

Tool: Describe a CRAN package

Description

Describes a CRAN package using pkgsearch::cran_package().

Usage

```
btw_tool_search_package_info(package_name, `_intent` = "")
```

Arguments

package_name	The name of a package on CRAN.
_intent	An optional string describing the intent of the tool use. When the tool is used by an LLM, the model will use this argument to explain why it called the tool.

Value

An info sheet about the package.

Examples

```
cli::cat_line(
  btw_this(pkgsearch::cran_package("anyflights"))
)
```

---

btw_tool_session_check_package_installed

Tool: Check if a package is installed

Description

Checks if a package is installed in the current session. If the package is installed, it returns the version number. If not, it suggests packages with similar names to help the LLM resolve typos.

Usage

```
btw_tool_session_check_package_installed(package_name, `_intent` = "")
```

Arguments

package_name	The name of the package.
_intent	An optional string describing the intent of the tool use. When the tool is used by an LLM, the model will use this argument to explain why it called the tool.

Value

A message indicating whether the package is installed and its version, or an error indicating that the package is not installed.

See Also

btw_tools()

Examples

```
btw_tool_session_check_package_installed("dplyr")@value

tryCatch(
  btw_tool_session_check_package_installed("dplry"),
  error = function(err) {
    cat(conditionMessage(err))
  }
)
```

`btw_tool_session_package_info`

Tool: Gather information about a package or currently loaded packages

---

Description

Uses `sessioninfo::package_info()` to provide information about the loaded, attached, or installed packages. The primary use case is to verify that a package is installed; check the version number of a specific packages; or determine which packages are already in use in a session.

Usage

```
btw_tool_session_package_info(
  packages = "attached",
  dependencies = "",
  `_intent` = ""
)
```

Arguments

`packages`	Which packages to show, or `"loaded"` to show all loaded packages, `"attached"` to show all attached packages, or `"installed"` to show all installed packages.
`dependencies`	Whether to include the dependencies when listing package information.
`_intent`	An optional string describing the intent of the tool use. When the tool is used by an LLM, the model will use this argument to explain why it called the tool.

Value

Returns a string describing the selected packages.

See Also

btw_tools(), btw_tool_session_platform_info()

Other Tools: btw_tool_docs_package_news(), btw_tool_env_describe_data_frame(), btw_tool_env_describe_env
btw_tool_files_code_search(), btw_tool_files_list_files(), btw_tool_files_read_text_file(),
btw_tool_files_write_text_file(), btw_tool_ide_read_current_editor(), btw_tool_package_docs,
btw_tool_search_packages(), btw_tool_session_platform_info(), btw_tool_web_read_url(),
btw_tools()

Examples

```
btw_tool_session_package_info("btw")
```

---

btw_tool_session_platform_info

Tool: Describe user's platform

Description

Describes the R version, operating system, and language and locale settings for the user's system. When using btw_client() or btw_app(), this information is automatically included in the system prompt.

Usage

```
btw_tool_session_platform_info(`_intent` = "")
```

Arguments

_intent	An optional string describing the intent of the tool use. When the tool is used by an LLM, the model will use this argument to explain why it called the tool.

Value

Returns a string describing the user's platform.

See Also

btw_tools()

Other Tools: btw_tool_docs_package_news(), btw_tool_env_describe_data_frame(), btw_tool_env_describe_env
btw_tool_files_code_search(), btw_tool_files_list_files(), btw_tool_files_read_text_file(),
btw_tool_files_write_text_file(), btw_tool_ide_read_current_editor(), btw_tool_package_docs,
btw_tool_search_packages(), btw_tool_session_package_info(), btw_tool_web_read_url(),
btw_tools()

Examples

```
btw_tool_session_platform_info()
```

`btw_tool_web_read_url` *Tool: Read a Web Page as Markdown*

Description

Tool: Read a Web Page as Markdown

Usage

```
btw_tool_web_read_url(url, `_intent` = "")
```

Arguments

`url`	The URL of the web page to read.
`_intent`	An optional string describing the intent of the tool use. When the tool is used by an LLM, the model will use this argument to explain why it called the tool.

Details

You can control the maximum time to wait for the page to load by setting the `btw.max_wait_for_page_load_s` option globally in your R session.

Value

Returns a `BtwWebPageResult` object that inherits from ellmer::ContentToolResult containing the markdown content of the web page.

See Also

Other Tools: `btw_tool_docs_package_news()`, `btw_tool_env_describe_data_frame()`, `btw_tool_env_describe_env`
`btw_tool_files_code_search()`, `btw_tool_files_list_files()`, `btw_tool_files_read_text_file()`,
`btw_tool_files_write_text_file()`, `btw_tool_ide_read_current_editor()`, `btw_tool_package_docs`,
`btw_tool_search_packages()`, `btw_tool_session_package_info()`, `btw_tool_session_platform_info()`,
`btw_tools()`

Examples

```
btw_tool_web_read_url("https://www.r-project.org/")
btw_tool_web_read_url(
  "https://posit.co/blog/easy-tool-calls-with-ellmer-and-chatlas/"
)
```

mcp *Start a Model Context Protocol server with btw tools*

Description

`btw_mcp_server()` starts an MCP server with tools from `btw_tools()`, which can provide MCP clients like Claude Desktop or Claude Code with additional context. The function will block the R process it's called in and isn't intended for interactive use.

To give the MCP server access to a specific R session, run `btw_mcp_session()` in that session. If there are no sessions configured, the server will run the tools in its own session, meaning that e.g. the `btw_tools(tools = "env")` tools will describe R objects in *that* R environment.

Usage

```
btw_mcp_server(tools = btw_tools())

btw_mcp_session()
```

Arguments

`tools`	A list of `ellmer::tool()`s to use in the MCP server, defaults to the tools provided by `btw_tools()`. Use `btw_tools()` to subset to specific list of **btw** tools that can be augmented with additional tools. Alternatively, you can pass a path to an R script that returns a list of tools as supported by `mcptools::mcp_server()`.

Value

Returns the result of `mcptools::mcp_server()` or `mcptools::mcp_session()`.

Configuration

To configure this server with MCP clients, use the command `Rscript` and the args `-e "btw::btw_mcp_server()"`. For example, in Claude Desktop's configuration format:

```
{
  "mcpServers": {
    "r-btw": {
      "command": "Rscript",
      "args": ["-e", "btw::btw_mcp_server()"]
    }
  }
}
```

For Claude Code:

```
claude mcp add -s "user" r-btw -- Rscript -e "btw::btw_mcp_server()"
```

For Continue, include the following in your config file:

```
"experimental": {
  "modelContextProtocolServers": [
    {
      "transport": {
        "name": "r-btw",
        "type": "stdio",
        "command": "Rscript",
        "args": [
          "-e",
          "btw::btw_mcp_server()"
        ]
      }
    }
  ]
}
```

Additional Examples

`btw_mcp_server()` should only be run non-interactively, as it will block the current R process once called.

To start a server with btw tools:

```
btw_mcp_server()
```

Or to only do so with a subset of btw's tools, e.g. those that fetch package documentation:

```
btw_mcp_server(tools = btw_tools("docs"))
```

To allow the server to access variables in specific sessions, call `btw_mcp_session()` in that session:

```
btw_mcp_session()
```

See Also

These functions use `mcptools::mcp_server()` and `mcptools::mcp_session()` under the hood. To configure arbitrary tools with an MCP client, see the documentation of those functions.

Examples

```
# btw_mcp_server() and btw_mcp_session() are only intended to be run in
# non-interactive R sessions, e.g. when started by an MCP client like
# Claude Desktop or Claude Code. Therefore, we don't run these functions
# in examples.

# See above for more details and examples.
```

use_btw_md *Create or edit a btw.md context file*

Description

Create or edit a `btw.md` or `AGENTS.md` context file for your project or user-level configuration. These functions help you set up the context files that `btw_client()` and `btw_app()` use to configure chat clients.

`use_btw_md()` creates a new context file with a default template. If the file already exists, it will not overwrite it, but will still ensure the file is added to `.Rbuildignore` if you're in an R package.

`edit_btw_md()` opens an existing context file for editing. Without arguments, it opens the same file that `btw_client()` would use by default.

Usage

```
use_btw_md(scope = "project")

edit_btw_md(scope = NULL)
```

Arguments

scope	The scope of the context file. Can be: • `"project"` (default): Creates/opens `btw.md` (by default) or `AGENTS.md` in the project root • `"user"`: Creates/opens `btw.md` in your home directory • A directory path: Creates/opens `btw.md` in that directory • A file path: Creates/opens that specific file For `edit_btw_md()`, `scope = NULL` (default) will find and open the context file that `btw_client()` would use, searching first for `btw.md` and then `AGENTS.md` in the project directory and then for `btw.md` in your home directory.

Value

`use_btw_md()` returns the path to the context file, invisibly. `edit_btw_md()` is called for its side effect of opening the file.

Functions

- `use_btw_md()`: Create a new `btw.md` or `AGENTS.md` context file in the current directory, the project directory or your home directory.
- `edit_btw_md()`: Open an existing `btw.md` or `AGENTS.md` context file for editing.

Additional Examples

```
# Create a project-level btw.md
use_btw_md()

# Create a user-level btw.md
use_btw_md("user")

# Create an AGENTS.md file
use_btw_md("AGENTS.md")

# Edit the context file that btw_client() would use
edit_btw_md()

# Edit a specific context file
edit_btw_md("user")
```

Project Context

You can use a `btw.md` or `AGENTS.md` file to keep track of project-specific rules, guidance and context in your project. Either file name will work, so we'll refer primarily to `btw.md`. These files are used automatically by `btw_client()` and `btw_app()`: they look first for `btw.md` and then for `AGENTS.md`. If both files are present, only the `btw.md` file will be used.

Any time you start a chat client with `btw_client()` or launch a chat session with `btw_app()`, btw will automatically find and include the contents of the `btw.md` or `AGENTS.md` file in the system prompt of your chat. This helps maintain context and consistency across chat sessions.

Use `btw.md` to inform the LLM of your preferred code style, to provide domain-specific terminology or definitions, to establish project documentation, goals and constraints, to include reference materials such or technical specifications, or more. Storing this kind of information in `btw.md` may help you avoid repeating yourself and can be used to maintain coherence across many chat sessions.

Write in markdown and structure the file in any way you wish, or use `btw_task_create_btw_md()` to help you create a project context file for an existing project with the help of an AI agent.

Chat Settings

You can also use the `btw.md` file to choose default chat settings for your project in a YAML front matter block at the top of the file. In this YAML block you can choose settings for the default ellmer chat `client`, e.g. `provider`, `model`, as well as choose which `btw_tools()` to use in `btw_client()` or `btw_app()`.

Chat client settings

Use the `client` field to set options for the chat client. This can be a single string in `provider` or `provider/model` format – as used by `ellmer::chat()` – or a list of client options with `provider` and `model` fields, as well as any other options supported by the underlying `ellmer::chat_*()` function you choose. Note that `provider` maps to the `ellmer::chat_*()` function, while `model` maps to the `model` argument of that function.

- Using ellmer's default model for a provider:

```
---
client: openai
---

---
client:
  provider: openai
---
```

- Using a specific model:

```
---
client: anthropic/claude-4-5-sonnet-latest
---

---
client:
  provider: anthropic
  model: claude-4-5-sonnet-latest
---
```

- Using additional client options:

```
---
client:
  provider: ollama
  model: "gpt-oss:20b"
  echo: output
  base_url: "http://my-company.example.com:11434"
---
```

Tool Settings

The top-level `tools` field is used to specify which btw tools are included in the chat. This should be a list of tool groups or tool names (with or without the `btw_tool_` prefix). See `btw_tools()` for a list of available tools and tool groups.

Here's an example `btw.md` file:

```
---
client: claude/claude-4-5-sonnet-latest
tools: [docs, env, files, git, ide, search, session, web]
---

Follow these important style rules when writing R code:

* Prefer solutions that use {tidyverse}
* Always use `<-` for assignment
* Always use the native base-R pipe `|>` for piped expressions
```

Selective Context

One use-case for `btw.md` is to provide stable context for an on-going task that might span multiple chat sessions. In this case, you can use `btw.md` to hold the complete project plan, with background information, requirements, and specific tasks to be completed. This can help maintain continuity across chat sessions, especially if you update the `btw.md` file as the project progresses.

In this use case, however, you might want to hide parts of the project plan from the system prompt, for example to hide completed or future tasks when their description would distract the LLM from the current task.

You can hide parts of the `btw.md` file from the system prompt by wrapping them in HTML `<!-- HIDE -->` and `<!-- /HIDE -->` comment tags. A single `<!-- HIDE -->` comment tag will hide all content after it until the next `<!-- /HIDE -->` tag, or the end of the file. This is particularly useful when your system prompt contains notes to yourself or future tasks that you do not want to be included in the system prompt.

Project or User Scope

For project-specific configuration, store your `btw.md` file in the root of your project directory. You can even have multiple `btw.md` files in your project, in which case the one closest to your current working directory will be used. This makes it easy to have different `btw.md` files for different sub-projects or sub-directories within a larger project.

For global configuration, you can maintain a `btw.md` file in your home directory (at `btw.md` or `.config/btw/btw.md` in your home directory, using `fs::path_home()`). This file will be used by default when a project-specific `btw.md` file is not found. Note that **btw** only looks for `btw.md` in your home directory if no project-specific `btw.md` or `AGENTS.md` file is present. It also does not look for `AGENTS.md` in your home directory.

Interactive Setup

For an interactive guided setup, consider using `btw_task_create_btw_md()` to use an LLM to help you create a `btw.md` file for your project.

See Also

Project context files are discovered automatically and included in the system prompt by `btw_client()`. See `btw_tools()` for a list of available tools.

Examples

```
# See additional examples in the sections above

withr::with_tempdir({
  withr::with_tempfile("btw_md_tmp", fileext = ".md", {
    use_btw_md(btw_md_tmp)
  })
})
```

Index

∗ **Tools**
btw_tool_docs_package_news, 17
btw_tool_env_describe_data_frame, 18
btw_tool_env_describe_environment, 19
btw_tool_files_code_search, 20
btw_tool_files_list_files, 22
btw_tool_files_read_text_file, 23
btw_tool_files_write_text_file, 24
btw_tool_ide_read_current_editor, 34
btw_tool_package_docs, 35
btw_tool_search_packages, 36
btw_tool_session_package_info, 39
btw_tool_session_platform_info, 40
btw_tool_web_read_url, 41
btw_tools, 14
∗ **btw formatting methods**
btw_this, 9
btw_this.character, 10
btw_this.data.frame, 12
btw_this.environment, 13
∗ **git tools**
btw_tool_git_branch_checkout, 27
btw_tool_git_branch_create, 28
btw_tool_git_branch_list, 29
btw_tool_git_commit, 30
btw_tool_git_diff, 31
btw_tool_git_log, 32
btw_tool_git_status, 33
∗ **github tools**
btw_tool_github, 25
∗ **task and agent functions**
btw_task_create_btw_md, 6
btw_task_create_readme, 8
∗ **tools**
btw_tool_session_check_package_installed, 38
btw, 3
btw(), *7*, *8*
btw.md, *26*
btw_app *(*btw_client*)*, 4
btw_app(), *40*, *44*, *45*
btw_client, 4
btw_client(), *7*, *40*, *44*, *45*, *47*
btw_mcp_server *(*mcp*)*, 42
btw_mcp_session *(*mcp*)*, 42
btw_task_create_btw_md, 6, *9*
btw_task_create_btw_md(), *45*, *47*
btw_task_create_readme, *7*, 8
btw_this, 9, *11*, *13*, *14*
btw_this.character, *9*, 10, *13*, *14*
btw_this.data.frame, *9*, *11*, 12, *14*
btw_this.data.frame(), *19*
btw_this.environment, *9*, *11*, *13*, 13
btw_this.environment(), *20*
btw_this.tbl *(*btw_this.data.frame*)*, 12
btw_tool_docs_available_vignettes *(*btw_tool_package_docs*)*, 35
btw_tool_docs_available_vignettes(), *14*
btw_tool_docs_help_page *(*btw_tool_package_docs*)*, 35
btw_tool_docs_help_page(), *4*, *10*, *14*
btw_tool_docs_package_help_topics *(*btw_tool_package_docs*)*, 35
btw_tool_docs_package_help_topics(), *4*, *10*, *14*
btw_tool_docs_package_news, *16*, 17, *19*, *20*, *22–25*, *35–37*, *40*, *41*
btw_tool_docs_package_news(), *14*
btw_tool_docs_vignette *(*btw_tool_package_docs*)*, 35
btw_tool_docs_vignette(), *10*, *14*
btw_tool_env_describe_data_frame, *16*, *17*, 18, *20*, *22–25*, *35–37*, *40*, *41*
btw_tool_env_describe_data_frame(), *13*,

15
btw_tool_env_describe_environment, *16, 17*, *19*, 19, *22–25*, *35–37*, *40, 41*
btw_tool_env_describe_environment(), *14, 15*
btw_tool_files_code_search, *16, 17*, *19, 20*, 20, *23–25*, *35–37*, *40, 41*
btw_tool_files_code_search(), *15*
btw_tool_files_list_files, *16, 17*, *19, 20*, *22*, 22, *24, 25*, *35–37*, *40, 41*
btw_tool_files_list_files(), *10*, *15*
btw_tool_files_read_text_file, *16, 17*, *19, 20*, *22, 23*, 23, *25*, *35–37*, *40, 41*
btw_tool_files_read_text_file(), *10*, *15*
btw_tool_files_write_text_file, *16, 17*, *19, 20*, *22–24*, 24, *35–37*, *40, 41*
btw_tool_files_write_text_file(), *15*
btw_tool_git_branch_checkout, 27, *29–34*
btw_tool_git_branch_checkout(), *15*
btw_tool_git_branch_create, *27*, 28, *30–34*
btw_tool_git_branch_create(), *15*
btw_tool_git_branch_list, *27*, *29*, 29, *31–34*
btw_tool_git_branch_list(), *15*
btw_tool_git_commit, *27*, *29, 30*, 30, *32–34*
btw_tool_git_commit(), *15*
btw_tool_git_diff, *27*, *29–31*, 31, *33, 34*
btw_tool_git_diff(), *15*
btw_tool_git_log, *27*, *29–32*, 32, *34*
btw_tool_git_log(), *15*
btw_tool_git_status, *27*, *29–33*, 33
btw_tool_git_status(), *15*
btw_tool_github, 25
btw_tool_github(), *15*
btw_tool_ide_read_current_editor, *16, 17*, *19, 20*, *22–25*, 34, *36, 37*, *40, 41*
btw_tool_ide_read_current_editor(), *15*
btw_tool_package_docs, *16, 17*, *19, 20*, *22–25*, *35*, 35, *37*, *40, 41*
btw_tool_search_package_info, 37
btw_tool_search_package_info(), *15*
btw_tool_search_packages, *16, 17*, *19, 20*, *22–25*, *35, 36*, 36, *40, 41*
btw_tool_search_packages(), *15*
btw_tool_session_check_package_installed, 38
btw_tool_session_check_package_installed(), *16*
btw_tool_session_package_info, *16, 17*, *19, 20*, *22–25*, *35–37*, 39, *40, 41*
btw_tool_session_package_info(), *16*
btw_tool_session_platform_info, *16, 17*, *19, 20*, *22–25*, *35–37*, *40*, 40, *41*
btw_tool_session_platform_info(), *16*, *40*
btw_tool_web_read_url, *16, 17*, *19, 20*, *22–25*, *35–37*, *40*, 41
btw_tool_web_read_url(), *16*
btw_tools, 14, *17*, *19, 20*, *22–25*, *35–37*, *40, 41*
btw_tools(), *4–6*, *17*, *19, 20*, *36–38*, *40*, *42*, *45–47*

edit_btw_md *(*use_btw_md*)*, 44
ellmer::Chat, *3–8*
ellmer::chat(), *5*, *7, 8*, *45*
ellmer::chat_anthropic(), *5*, *7, 8*
ellmer::ContentText, *4*
ellmer::ContentToolResult, *41*
ellmer::tool(), *6*, *42*

fs::dir_info(), *22*

gert::git_add(), *30*
gert::git_branch_checkout(), *27*
gert::git_branch_create(), *28*
gert::git_branch_list(), *29*
gert::git_commit(), *30*
gert::git_diff_patch(), *31*
gert::git_log(), *32*
gert::git_status(), *33*
gh::gh(), *25*
grep(), *22*

mcp, 42
mcptools::mcp_server(), *42, 43*
mcptools::mcp_session(), *42, 43*

pkgsearch::cran_package(), *37*
pkgsearch::pkg_search(), *36*

rlang::global_entrace(), *11*
rstudioapi::getSourceEditorContext(), *11*

sessioninfo::package_info(), *11*, *39*
shiny::shinyApp(), *5*

shinychat::chat_mod_ui(), *6*
skimr::skim(), *12*, *18*

use_btw_md, 44
use_btw_md(), *5*