Package 'mcptools'

July 18, 2025

Title Model Context Protocol Servers and Clients

Version 0.1.0

Description Implements the Model Context Protocol (MCP). Users can start 'R'-based servers, serving functions as tools for large language models to call before responding to the user in MCP-compatible apps like 'Claude Desktop' and 'Claude Code', with options to run those tools inside of interactive 'R' sessions. On the other end, when 'R' is the client via the 'ellmer' package, users can register tools from third-party MCP servers to integrate additional context into chats.

License MIT + file LICENSE

Suggests knitr, rmarkdown, testthat (>= 3.0.0), withr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

Imports cli, ellmer (>= 0.2.0), jsonlite, nanonext (>= 1.6.0), processx, promises, rlang

Depends R (>= 4.1.0)

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

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

Config/Needs/website tidyverse/tidytemplate

VignetteBuilder knitr

NeedsCompilation no

Author Simon Couch [aut, cre] (ORCID: <https://orcid.org/0000-0001-5676-5107>), Winston Chang [aut], Charlie Gao [aut] (ORCID: <https://orcid.org/0000-0002-0750-061X>), Posit Software, PBC [cph, fnd] (ROR: <https://ror.org/03wc8by49>)

Maintainer Simon Couch <simon.couch@posit.co>

Repository CRAN

Date/Publication 2025-07-18 15:10:07 UTC

Contents

`client` *R as a client: Define ellmer tools from MCP servers*

Description

These functions implement R as an MCP *client*, so that ellmer chats can register functionality from third-party MCP servers such as those listed here: https://github.com/modelcontextprotocol/servers.

`mcp_tools()` fetches tools from MCP servers configured in the mcptools server config file and converts them to a list of tools compatible with the `$set_tools()` method of ellmer::Chat objects.

Usage

```
mcp_tools(config = NULL)
```

Arguments

`config`	A single string indicating the path to the mcptools MCP servers configuration file. If one is not supplied, mcptools will look for one at the file path configured with the option `.mcptools_config`, falling back to `file.path("~", ".config", "mcptools", "config.json")`.

Value

- `mcp_tools()` returns a list of ellmer tools that can be passed directly to the `$set_tools()` method of an ellmer::Chat object. If the file at `config` doesn't exist, an error.

Configuration

mcptools uses the same .json configuration file format as Claude Desktop; most MCP servers will define example .json to configure the server with Claude Desktop in their README files. By default, mcptools will look to `file.path("~", ".config", "mcptools", "config.json")`; you can edit that file with `file.edit(file.path("~", ".config", "mcptools", "config.json"))`.

The mcptools config file should be valid .json with an entry `mcpServers`. That entry should contain named elements, each with at least a `command` and `args` entry.

For example, to configure `mcp_tools()` with GitHub's official MCP Server https://github.com/github/github-mcp-server, you could write the following in that file:

```
{
  "mcpServers": {
    "github": {
      "command": "docker",
      "args": [
        "run",
        "-i",
        "--rm",
        "-e",
        "GITHUB_PERSONAL_ACCESS_TOKEN",
        "ghcr.io/github/github-mcp-server"
      ],
      "env": {
        "GITHUB_PERSONAL_ACCESS_TOKEN": "<add_your_github_pat_here>"
      }
    }
  }
}
```

See Also

This function implements R as an MCP *client*. To use R as an MCP *server*, i.e. to provide apps like Claude Desktop or Claude Code with access to R-based tools, see `mcp_server()`.

Examples

```
# setup
config_file <- tempfile(fileext = "json")
file.create(config_file)

# usually, `config` would be a persistent, user-level
# configuration file for a set of MCP server
mcp_tools(config = config_file)

# teardown
file.remove(config_file)
```

`server` *R as a server: Configure R-based tools with LLM-enabled apps*

Description

`mcp_server()` implements a model context protocol server with arbitrary R functions as its tools. Optionally, calling `mcp_session()` in an interactive R session allows those tools to execute inside of that session.

Usage

```
mcp_server(tools = NULL)

mcp_session()
```

Arguments

tools	A list of tools created with `ellmer::tool()` that will be available from the server or a file path to an .R file that, when sourced, will return a list of tools. Any list that could be passed to `Chat$set_tools()` can be passed here. By default, the package won't serve any tools other than those needed to communicate with interactive R sessions.

Value

`mcp_server()` and `mcp_session()` are both called primarily for side-effects.

- `mcp_server()` blocks the R process it's called in indefinitely and isn't intended for interactive use.
- `mcp_session()` makes the interactive R session it's called in available to MCP servers. It returns a promise via `promises::promise()`.

Configuration

`mcp_server()` should be configured with the MCP clients via the `Rscript` command. For example, to use with Claude Desktop, paste the following in your Claude Desktop configuration (on macOS, at `file.edit("~/Library/Application Support/Claude/claude_desktop_config.json")`):

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

Or, to use with Claude Code, you might type in a terminal:

```
claude mcp add -s "user" r-mcptools Rscript -e "mcptools::mcp_server()"
```

mcp_server() is not intended for interactive use.

The server interfaces with the MCP client. If you'd like tools to have access to variables inside of an interactive R session, call `mcp_session()` to make your R session available to the server. Place a call to `mcptools::mcp_session()` in your `.Rprofile`, perhaps with `usethis::edit_r_profile()`, to make every interactive R session you start available to the server.

On Windows, you may need to configure the full path to the Rscript executable. Examples for Claude Code on WSL and Claude Desktop on Windows are shown at https://github.com/posit-dev/mcptools/issues/41#issuecomment-3036617046.

See Also

- The "R as an MCP server" vignette at `vignette("server", package = "mcptools")` delves into further detail on setup and customization.
- These functions implement R as an MCP *server*. To use R as an MCP *client*, i.e. to configure tools from third-party MCP servers with ellmer chats, see `mcp_tools()`.

Examples

```
# should only be run non-interactively, and will block the current R process
# once called.
if (identical(Sys.getenv("MCPTOOLS_CAN_BLOCK_PROCESS"), "true")) {
# to start a server with a tool to draw numbers from a random normal:
library(ellmer)

tool_rnorm <- tool(
  rnorm,
  "Draw numbers from a random normal distribution",
  n = type_integer("The number of observations. Must be a positive integer."),
  mean = type_number("The mean value of the distribution."),
 sd = type_number("The standard deviation of the distribution. Must be a non-negative number.")
)

mcp_server(tools = list(tool_rnorm))

# can also supply a file path as `tools`
readLines(system.file("example-ellmer-tools.R", package = "mcptools"))

mcp_server(tools = system.file("example-ellmer-tools.R", package = "mcptools"))
}

if (interactive()) {
  mcp_session()
}
```

Index