Package: miner (via r-universe)

June 6, 2026

Title Control Minecraft with RaspberryJuice API

Version 0.3.2

Date 2026-05-20

Description Following the code in py3minepi, allows one to connect to the minecraft from R.

Depends R (>= 3.5.0)

Imports stats, utils

Suggests testthat, knitr, rmarkdown

License MIT + file LICENSE

Encoding UTF-8

LazyData true

URL https://github.com/kbroman/miner

BugReports https://github.com/kbroman/miner/issues

VignetteBuilder knitr

Roxygen list(markdown=TRUE)

Config/roxygen2/version 8.0.0

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

Date/Publication 2026-06-06 11:45:29 UTC

RemoteUrl https://github.com/kbroman/miner

RemoteRef HEAD

RemoteSha 206577a76648b2e5bd744a0c19bb50fee91cca7d

Contents

`chatPost` *Post to chat*

Description

Post a message to Minecraft chat.

Usage

```
chatPost(text)
```

Arguments

`text`	A character string with the message you would like to post.

Examples

```
## Not run:
chatPost('foobar')

## Post the first few digits of pi to chat
for (p in strsplit(as.character(round(pi, 4)), '')[[1]]) {
    chatPost(p)
  }
```

```
## End(Not run)
```

`find_entity` *Find entity by name*

Description

Find a Minecraft entity by name.

Usage

```
find_entity(name)
```

Arguments

`name`	Character string with a Minecraft entity name (can be a partial match)

Details

We first look to see whether there is an exact match to the `namex` column in the output of `getEntityTypes()`. If there is, we return that row. If not, we use `grep()` with `ignore.case=TRUE` and return matching rows.

Value

Data frame with a row or set of rows from the output of `getEntityTypes()` that match the queried name.

See Also

`getEntityTypes()`, `find_item()`

Examples

```
## Not run:
find_entity("HORSE")
find_entity("horse")

## End(Not run)
```

`find_item` *Find item by name or ID/style*

Description

Find a Minecraft item by name or ID. If querying an item by ID, the search can also specify item style.

Usage

```
find_item(name = NULL, id = NULL, style = 0)
```

Arguments

`name`	Character string with the name of a Minecraft item (specify either `name` or `id`, not both)
`id`	A numeric or character string with the ID of a Minecraft item (specify either `name` or `id`, not both)
`style`	A numeric or character string with the style of a Minecraft item (use this argument only if querying by `id` is provided)

Details

If `name` is provided, we first look to see whether there is an exact match to the `name` column in `mc_items`. If there is, we return that row. If not, we use `grep()` with `ignore.case=TRUE` and return matching rows.

If instead `id` is provided, we return the row with that id. The default is to return the row with that ID and `style==0`, or whatever style was provided. If `style` is NULL, we return all rows with that ID.

Value

Data frame with a row or set of rows from `mc_items` that match the queried name, ID, and / or style.

See Also

`find_entity()`, `mc_items` x

Examples

```
find_item(name = "Oak")
find_item(id = 5)
find_item(id = 5, style = 5)
```

getBlock *Determine block type and style at some position*

Description

Determine the type of the block at position x (north / south), y (height), z (east / west). By default, the block's style is also given, although the style can be excluded from the output using the `include_style` parameter.

Usage

```
getBlock(x, y, z, include_style = TRUE)
```

Arguments

`x`	A numeric string with north/south position
`y`	A numeric string with height
`z`	A numeric string with east/west position
`include_style`	A logical value of whether the block's style should also be included in the output (defaults to TRUE).

Value

A numeric vector of length one or two with the type ID and style, if `include_style` is `TRUE`, of the block at position (x, y, z). You can use `find_item()` to find the name of the block type based on this returned ID.

Examples

```
## Not run:
mc_connect()
h <- getHeight(0,0)
b_type <- getBlock(0,h,0)
b_type

find_item(id = b_type[1])
find_item(id = b_type[1], style = b_type[2])

getBlock(0,h,0, include_style = FALSE)

## End(Not run)
```

getBlockHits *Get most recent block hits*

Description

Return the most recent block hits made in the Minecraft world by an iron sword.

Usage

```
getBlockHits()
```

Value

A dataframe with columns for the coordinates, block type, and player id of recent block hits.

Note

Only right clicks with an iron sword are logged.

See Also

getChatPosts()

Examples

```
## Not run:
getBlockHits()

library(futile.logger)
while (TRUE) {
    flog.info(getEventsBlockHits())
    Sys.sleep(1)
}

## End(Not run)
```

getBlocks *Determine block types in a cuboid*

Description

Determine block types in a cuboid for which one corner is at the position (x0, y0, z0) and the opposite corner is at the position (x1, y1, z1).

Usage

```
getBlocks(x0, y0, z0, x1, y1, z1)
```

Arguments

x0	A numeric value giving the starting north / south position of the cuboid.
y0	A numeric value giving the starting height of the cuboid.
z0	A numeric value giving the starting east / west position of the cuboid.
x1	A numeric value giving the north / south position of the opposite corner of the cuboid.
y1	A numeric value giving the ending height of the opposite corner of the cuboid.
z1	A numeric value giving the ending east / west position of the opposite corner of the cuboid.

Value

A 3-D array of integers where each integer gives the ID of the type of a block in the cuboid.

Examples

```
## Not run:
mc_connect()
h <- getHeight(0,0)
block_types <- getBlocks(0, h, 0, 1, h + 3, 2)
block_types

find_item(id = block_types[1, 1, 1])

## End(Not run)
```

---

getChatPosts *Pull the most recent chat message*

Description

Returns the chat posts that have been posted recently and since the last time the function was run. It is possible that this does not return chat posts that were posted using chatPost().

Usage

```
getChatPosts()
```

Value

A dataframe of recent chat posts with columns for the player id of the poster and the message posted

See Also

getBlockHits()

Examples

```
## Not run:
mc_connect()
getChatPosts()

## End(Not run)
```

`getEntityTypes` *Get entity types*

Description

Get a table of available entity types and their IDs.

Usage

```
getEntityTypes()
```

Value

a data frame with two rows, id and type.

See Also

`mc_items()`, `spawnEntity()`, `find_entity()`

Examples

```
## Not run:
mc_connect()
getEntityTypes()

## End(Not run)
```

`getHeight` *Get height of the world at a given position*

Description

Get height of the world at a given position, where the world's height is defined as the height of the highest non-air point at that location.

Usage

```
getHeight(x, z)
```

Arguments

x	A numeric string with north/south position
z	A numeric string with east/west position

Value

Integer with height of block just above the last bit of non-air at the specified x / z position.

Examples

```
## Not run:
mc_connect()
getHeight(0,0)

## End(Not run)
```

---

getPlayerDirection *Get player direction as unit vector*

Description

Returns a unit vector describing the current direction a player is facing. The default is to get the direction for the first player spawned in the Minecraft, world, but the directions of other players can be gotten using the `player_id` argument.

Usage

```
getPlayerDirection(player_id = NULL)
```

Arguments

player_id	Integer giving the ID of a player. You can find IDs of all current players using `getPlayerIds()`.

Details

x is east/west with east being the positive direction. y is up/down with up being the positive direction, and z is north/south with south being the positive direction.

Value

A numeric vector of length 3 with coordinates of the player's current direction (gaze) as a unit vector, (x,y,z).

See Also

`getPlayerRotation()`, `getPlayerPitch()`, `getPlayerPos()`

Examples

```
## Not run:
getPlayerDirection()

example_playerId <- getPlayerIds()[1]
getPlayerDirection(example_playerId)

## End(Not run)
```

`getPlayerId` *Get player ID*

Description

Get the ID for a player with a given name.

Usage

```
getPlayerId(player_name)
```

Arguments

`player_name`	Character string with name of a player.

Value

A number representing the ID of the player.

See Also

`getPlayerIds()`, `getPlayerName()`

Examples

```
## Not run:
mc_connect()
getPlayerIds()
getPlayerId("wyldstyle")

## End(Not run)
```

getPlayerIds *Get player IDs*

Description

Get the IDs of all player currently in the world.

Usage

```
getPlayerIds()
```

Value

A numeric vector with the IDs of each player currently in the MineCraft world. Player ids are listed in the order they joined the game world.

See Also

`getPlayerId()`, `getPlayerName()`

Examples

```
## Not run:
mc_connect()
getPlayerIds()

## End(Not run)
```

getPlayerName *Get player name*

Description

Get the name of a player with a given ID

Usage

```
getPlayerName(player_id)
```

Arguments

`player_id`	Integer giving the ID of a player. You can find IDs of all current players using `getPlayerIds()`.

Value

A character string with the name of the identified player.

See Also

`getPlayerIds()`, `getPlayerId()`

Examples

```
## Not run:
mc_connect()
getPlayerIds()
getPlayerName(322)

## End(Not run)
```

`getPlayerPitch` *Get player pitch*

Description

Return the player's pitch (angle in the up / down direction). The default is to get the rotation for the first player spawned in the Minecraft world, but this can be run for a different player by using the `player_id` argument.

Usage

```
getPlayerPitch(player_id = NULL)
```

Arguments

`player_id`	Integer giving the ID of a player. You can find IDs of all current players using `getPlayerIds()`.

Value

A numeric value between -90 and +90, giving the pitch of the player's viewpoint, with -90 indicating the player is looking straight up and +90 indicating that the player is looking straight down.

See Also

`getPlayerDirection()`, `getPlayerRotation()`, `getPlayerPos()`

Examples

```
## Not run:
getPlayerPitch()

example_playerId <- getPlayerIds()[1]
getPlayerPitch(example_playerId)

## End(Not run)
```

`getPlayerPos` *Get player position*

Description

Get entity position. The default is to get the position of the first player spawned in the game, but the positions of other players can be gotten using the `player_id` argument.

Usage

```
getPlayerPos(player_id = NULL, tile = FALSE)
```

Arguments

`player_id`	Integer giving the ID of a player. You can find IDs of all current players using `getPlayerIds()`.
`tile`	Logical value specifying whether to truncate the output position to an integer (i.e., the location of the tile on which the player is positioned).

Details

x is east/west with east being the positive direction. y is up/down with up being the positive direction, and z is north/south with south being the positive direction.

Value

A numeric vector of length three giving the position (x, y, and z) of the requested player.

See Also

`setPlayerPos()`, `getPlayerRotation()`, `getPlayerPitch()`, and `getPlayerDirection()`

Examples

```
## Not run:
mc_connect()
getPlayerPos()
getPlayerPos(tile = TRUE)

example_entity <- getPlayerIds()[1]
getPlayerPos(example_entity)

## End(Not run)
```

`getPlayerRotation` *Get player rotation*

Description

Get the current rotation of a player. The default is to get the rotation for the first player spawned in the Minecraft world, but this can be run for a different player by using the `player_id` argument.

Usage

```
getPlayerRotation(player_id = NULL)
```

Arguments

`player_id`	Integer giving the ID of a player. You can find IDs of all current players using `getPlayerIds()`.

Value

A numeric value between 0 and +360 indicating the direction that the player is facing, with 0 being south (positive z), 90 being west (negative x), 180 being north (negative z), 270 being east (positive x), and 360 being back at the south again.

See Also

`getPlayerDirection()`, `getPlayerPitch()`, `getPlayerPos()`

Examples

```
## Not run:
getPlayerRotation()

example_playerId <- getPlayerIds()[1]
getPlayerRotation(example_playerId)

## End(Not run)
```

mc_close *Close cached connection to Minecraft server*

Description

Close the current connection to a Minecraft server.

Usage

```
mc_close()
```

Value

None.

Examples

```
## Not run:
  mc_connect()
  getPlayerIds()
  mc_close()

## End(Not run)
```

mc_connect *Create a connection to a Minecraft server*

Description

Create a connection to a Minecraft server

Usage

```
mc_connect(
  hostname = Sys.getenv("SPIGOT_HOSTNAME", "localhost"),
  port = Sys.getenv("SPIGOT_PORT", "4711")
)
```

Arguments

hostname	A character string with the hostname or IP address for the Minecraft Spigot server to which you want to connect.
port	An integer giving the port to use for the connection.

Value

Nothing returned, the connection is cached within the package namespace.

Examples

```
## Not run:
mc_connect()
getPlayerIds()
mc_close()

## End(Not run)
```

`mc_items` *Minecraft item information*

Description

Dataset with the name, ID, and style of possible Minecraft items.

Usage

```
data(mc_items)
```

Format

Data frame with columns `name`, `id`, and `style`.

Source

https://minecraft-ids.grahamedgecombe.com/

See Also

`find_item()`, `find_entity()`

Examples

```
data(mc_items)
mc_items[grep("Dirt", mc_items$name),]
```

setBlock *Place a block*

Description

Place a block at position (x,y,z) by type id

Usage

```
setBlock(x, y, z, id, style = 0)
```

Arguments

x	A numeric string with north/south position
y	A numeric string with height
z	A numeric string with east/west position
id	A numeric or character string with the ID of a Minecraft item (specify either name or id, not both)
style	A numeric or character string with the style of a Minecraft item (use this argument only if querying by id is provided)

Value

None.

Examples

```
## Not run:
mc_connect()
h <- getHeight(0, 0)
setBlock(0, h, 0, 46)

## End(Not run)
```

setBlocks *Place blocks in a cuboid*

Description

Place blocks of a single type (specified by id) in the cuboid with opposite corners at the positions (x0, y0, z0) and (x1, y1, z1).

Usage

```
setBlocks(x0, y0, z0, x1, y1, z1, id)
```

Arguments

x0	A numeric value giving the starting north / south position of the cuboid.
y0	A numeric value giving the starting height of the cuboid.
z0	A numeric value giving the starting east / west position of the cuboid.
x1	A numeric value giving the north / south position of the opposite corner of the cuboid.
y1	A numeric value giving the ending height of the opposite corner of the cuboid.
z1	A numeric value giving the ending east / west position of the opposite corner of the cuboid.
id	A numeric or character string with the ID of a Minecraft item (specify either name or id, not both)

Value

None.

Examples

```
## Not run:
mc_connect()

ice <- find_item(name = "Ice")

h <- getHeight(0,0)
setBlocks(0, h, 0, 1, h + 3, 2, id = ice$id)

## End(Not run)
```

---

setPlayerDirection *Set a player's direction*

Description

Set a player's direction vector (x,y,z)

Usage

```
setPlayerDirection(x, y, z, player_id = NULL)
```

Arguments

x	east/west direction
y	up/down direction
z	north/south direction
player_id	Player or entity ID

Details

(x, y, z) define a unit vector to which the player will now point.

Note

Only works with RaspberryJuice version 1.11 or later.

See Also

getPlayerDirection(), setPlayerRotation(), setPlayerPitch(), setPlayerPos()

Examples

```
## Not run:
mc_connect()
getPlayerIds()
setPlayerDirection(1, 0, 1)

## End(Not run)
```

---

setPlayerPitch *Set a player's pitch*

Description

Set a player's pitch

Usage

```
setPlayerPitch(angle, player_id = NULL)
```

Arguments

angle	Angle of pitch (-90 is straight up and +90 is straight down)
player_id	Player or entity ID

Details

Angles < -90 degrees are treated as -90 degrees (straight up), and angles > 90 degrees are treated as +90 degrees (straight down).

Note

Only works with RaspberryJuice version 1.11 or later.

See Also

getPlayerPitch(), setPlayerRotation(), setPlayerDirection(), setPlayerPos()

Examples

```
## Not run:
mc_connect()
getPlayerIds()
setPlayerPitch(45, 355)

## End(Not run)
```

`setPlayerPos` *Change player position*

Description

Move player to position (x,y,z). The default is to move the first player who was spawned in the Minecraft world, but other players can also be moved using the `player_id` argument. If the `tile` argument is set to `TRUE`, the player will be moved to the position specified by truncating the specified x, y, and z to integers.

Usage

```
setPlayerPos(x, y, z, player_id = NULL, tile = FALSE)
```

Arguments

`x`	A numeric string with north/south position
`y`	A numeric string with height
`z`	A numeric string with east/west position
`player_id`	Integer giving the ID of a player. You can find IDs of all current players using `getPlayerIds()`.
`tile`	Logical value specifying whether to truncate the output position to an integer (i.e., the location of the tile on which the player is positioned).

Value

None.

Examples

```
## Not run:
mc_connect()
p <- getPlayerPos()
setPlayerPos(0, p + 5, 0)

example_entity <- getPlayerIds()[1]
getPlayerPos(example_entity)
setPlayerPos(0, p, 0, example_entity)
```

```
getPlayerPos(example_entity)

## End(Not run)
```

`setPlayerRotation` *Set a player's rotation*

Description

Set a player's rotation

Usage

```
setPlayerRotation(angle, player_id = NULL)
```

Arguments

`angle`	Angle of rotation (0-360)
`player_id`	Player or entity ID

Note

Only works with RaspberryJuice version 1.11 or later.

See Also

`getPlayerRotation()`, `setPlayerPitch()`, `setPlayerDirection()`, `setPlayerPos()`

Examples

```
## Not run:
mc_connect()
getPlayerIds()
current <- getPlayerRotation(355)
setPlayerRotation(current + 90, 355)

## End(Not run)
```

spawnEntity *Spawn an entity*

Description

Spawn an entity of a given type at a given position.

Usage

```
spawnEntity(x, y, z, typeid)
```

Arguments

x	east/west position (east is positive)
y	up/down position (up is positive)
z	north/south position (south is positive)
typeid	Integer giving the type of entity. See the output of `getEntityTypes()` or use `find_entity()` to search.

Value

A number giving the spawned entity's ID.

Note

Only works with RaspberryJuice version 1.11 or later.

See Also

`getEntityTypes()`, `find_entity()`

Examples

```
## Not run:
mc_connect()
witch_typeid <- find_entity("witch")$id
witch_id <- spawnEntity(0, 20, 0, witch_typeid)
getPlayerName(witch_id)

## End(Not run)
```

Index