# Package 'QBMS'

May 7, 2026

**Type** Package

**Title** Query the Breeding Management System(s)

**Version** 2.0.0

**Date** 2025-08-02

**Author** Khaled Al-Shamaa [aut, cre],
Mariano Omar Crimi [ctb],
Zakaria Kehel [ctb],
Johan Aparicio [ctb],
ICARDA [cph]

**Maintainer** Khaled Al-Shamaa <k.el-shamaa@cgiar.org>

**Description** This R package assists breeders in linking data systems with their analytic pipelines, a crucial step in digitizing breeding processes. It supports querying and retrieving phenotypic and genotypic data from systems like 'EBS' <https://ebs.excellenceinbreeding.org/>, 'BMS' <https://bmspro.io>, 'BreedBase' <https://breedbase.org>, 'GIGWA' <https://github.com/SouthGreenPlatform/Gigwa2> (using 'BrAPI' <https://brapi.org> calls), , and 'Germinate' <https://germinateplatform.github.io/get-germinate/>. Extra helper functions support environmental data sources, including 'TerraClimate' <https://www.climatologylab.org/terraclimate.html> and 'FAO' 'HWSDv2' <https://gaez.fao.org/pages/hwsd> soil database.

**License** GPL (>= 3)

**URL** https://icarda.github.io/QBMS/

**BugReports** https://github.com/icarda/QBMS/issues

**Depends** R (>= 3.1.0)

**Imports** httr2, jsonlite, tcltk, utils, RNetCDF, stats, terra, RSQLite, DBI, future, future.apply

**Suggests** knitr, rmarkdown, remotes

**VignetteBuilder** knitr

**Encoding** UTF-8

**NeedsCompilation** no

**RoxygenNote** 7.3.2

**Repository** CRAN

**Date/Publication** 2025-07-31 16:40:08 UTC

# Contents

---

`brapi_get_call` *Internal Function for Core BrAPI GET Calls*

---

## Description

Fetches data from an API endpoint, handles pagination by retrieving all pages, and consolidates the results into a single data frame.

## Usage

```
brapi_get_call(call_url, nested = TRUE, caller_func = NA)
```

## Arguments

| | |
|---|---|
| `call_url` | Character string specifying the base URL of the API endpoint to request. |
| `nested` | Logical value indicating whether to flatten nested lists in the JSON responses. Defaults to `TRUE`. |
| `caller_func` | Character string identifying the name of the function that invoked `brapi_get_call()`. |

## Details

This function performs the following steps:

1. Fetches the first page synchronously to determine the total number of pages.
2. If multiple pages exist, it asynchronously fetches the remaining pages using `get_async_pages()`.
3. Consolidates the data from all pages into a single data frame.
4. Updates global state variables with pagination information.

It relies on global variables from `qbms_globals` to manage state and configuration.

## Value

A list containing the consolidated data and associated metadata from the API response.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

---

# brapi_map *BrAPI Mapping Table*

---

## Description

This internal table maps QBMS function names to corresponding BrAPI calls for different versions of the BrAPI standard. It is used internally by the QBMS system to translate function calls into BrAPI-compliant API requests.

## Usage

```
brapi_map
```

## Format

An object of class `data.frame` with 37 rows and 3 columns.

## Details

This table supports both BrAPI v1 and v2 endpoints. It is regularly updated to reflect changes in the BrAPI standard and the inclusion of new API calls.

`brapi_post_search_allelematrix` *Internal Function Used for Core BrAPI POST Calls (Allele Matrix Search)*

## Description

This function is used internally to execute POST calls for retrieving the allele matrix via BrAPI. It handles the post request, waits for the results asynchronously if required, and processes the results.

## Usage

```
brapi_post_search_allelematrix(call_url, call_body, nested = TRUE)
```

## Arguments

| | |
|---|---|
| `call_url` | BrAPI URL for the POST request. |
| `call_body` | The request body to send with the POST request. |
| `nested` | Logical indicating whether to flatten the nested structure. Default is TRUE. |

## Value

A list of results obtained from the BrAPI POST call.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

---

`brapi_post_search_call` *Internal Function Used for Core BrAPI POST Calls*

## Description

This function is used internally to execute POST calls to BrAPI endpoints and retrieve the results while handling pagination and long-running tasks.

## Usage

```
brapi_post_search_call(call_url, call_body, nested = TRUE)
```

## Arguments

| | |
|---|---|
| `call_url` | BrAPI URL for the POST request. |
| `call_body` | The request body to send with the POST request. |
| `nested` | Logical indicating whether to flatten the nested structure. Default is TRUE. |

## Value

A list of results obtained from the BrAPI POST call.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

---

`build_pedigree_table` *Building Pedigree Table Recursively*

---

## Description

Recursively builds a pedigree table by extracting and tracking parents for each genotype/germplasm in the provided list. The function handles backcross cases and updates the pedigree data frame with parent information for multiple generations.

## Usage

```
build_pedigree_table(
  geno_list = NULL,
  pedigree_list = NULL,
  pedigree_df = NULL
)
```

## Arguments

| | |
|---|---|
| `geno_list` | A character vector of genotype/germplasm names. |
| `pedigree_list` | A character vector of associated pedigree strings, corresponding to the genotypes in `geno_list`. |
| `pedigree_df` | A data frame of pedigrees from a previous iteration, used to accumulate pedigree data. If NULL, a new data frame is created. |

## Value

A data frame with three columns: - 'Variety': The identifier for the individual genotype. - 'Female': The identifier for the female parent. - 'Male': The identifier for the male parent. The pedigree is built recursively, with individuals listed before any appearance as a parent.

## Author(s)

Khaled Al-Shamaa, <k.el-shamaa@cgiar.org>

`calc_biovars` *Calculate the Bioclimatic Variables*

## Description

This function calculates the 19 standard bioclimatic variables derived from monthly temperature and precipitation data. Bioclimatic variables are often used in ecological modeling and species distribution modeling to capture biologically meaningful patterns in climate data, including annual trends, seasonality, and extreme environmental factors.

The bioclimatic variables represent metrics such as the annual mean temperature, temperature seasonality, and precipitation patterns (e.g., wettest or driest quarter). These metrics help to model species distributions and analyze ecological dynamics.

The bioclimatic variables are coded as follows:

- **BIO1** = Annual Mean Temperature
- **BIO2** = Mean Diurnal Range (Mean of monthly (max temp - min temp))
- **BIO3** = Isothermality (BIO2/BIO7) (* 100)
- **BIO4** = Temperature Seasonality (standard deviation * 100)
- **BIO5** = Max Temperature of Warmest Month
- **BIO6** = Min Temperature of Coldest Month
- **BIO7** = Temperature Annual Range (BIO5 - BIO6)
- **BIO8** = Mean Temperature of Wettest Quarter
- **BIO9** = Mean Temperature of Driest Quarter
- **BIO10** = Mean Temperature of Warmest Quarter
- **BIO11** = Mean Temperature of Coldest Quarter
- **BIO12** = Annual Precipitation
- **BIO13** = Precipitation of Wettest Month
- **BIO14** = Precipitation of Driest Month
- **BIO15** = Precipitation Seasonality (Coefficient of Variation)
- **BIO16** = Precipitation of Wettest Quarter
- **BIO17** = Precipitation of Driest Quarter
- **BIO18** = Precipitation of Warmest Quarter
- **BIO19** = Precipitation of Coldest Quarter

These variables are computed using temperature and precipitation data in a standard format, and are critical for understanding species habitats and the effects of climate on ecosystems.

This work is derived from the dismo R package.

## Usage

```
calc_biovars(data)
```

### Arguments

| | |
|---|---|
| data | A data frame containing monthly climate data. The data frame must include:<br>• **year:** The year for each set of monthly data.<br>• **ppt:** Monthly precipitation values (in mm).<br>• **tmin:** Monthly minimum temperature values (in degrees Celsius).<br>• **tmax:** Monthly maximum temperature values (in degrees Celsius).<br>The data should contain 12 rows (one for each month from January to December) per year, with the columns sorted in the order of year, ppt, tmin, and tmax. |

### Value

A data frame with 19 columns representing the bioclimatic variables (BIO1 to BIO19) and an additional column for the year. The output data frame provides one row per year, with each column corresponding to one of the bioclimatic variables described above.

### Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

Robert Hijmans, Museum of Vertebrate Zoology, UC Berkeley

### References

Nix, 1986. A biogeographic analysis of Australian elapid snakes. In: R. Longmore (ed.). Atlas of elapid snakes of Australia. Australian Flora and Fauna Series 7. Australian Government Publishing Service, Canberra.

---

## debug_qbms *Debug Internal QBMS Status Object*

---

### Description

Retrieves the internal QBMS status object for debugging purposes. This object contains the current configuration and state of the QBMS session, including connection settings and active tokens.

### Usage

```
debug_qbms()
```

### Value

An environment object that holds the current QBMS configuration and state.

### Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## Examples

```
if (interactive()) {
  obj <- debug_qbms()
  obj$config
  obj$state
}
```

---

`get_async_page` *Asynchronously Fetch a Single API Page*

---

## Description

Sends an asynchronous HTTP GET request to fetch data from a single API page.

## Usage

```
get_async_page(full_url, nested)
```

## Arguments

| | |
|---|---|
| `full_url` | Character string specifying the full URL of the API endpoint to request. |
| `nested` | Logical value indicating whether to flatten nested lists in the JSON response. |

## Details

This function uses `future::future()` to perform the HTTP GET request asynchronously. It retrieves the content from the specified URL, checks for HTTP errors, and parses the JSON response.

## Value

A future representing the asynchronous operation, which will resolve to a list containing the parsed JSON response.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## get_async_pages *Asynchronously Fetch Multiple API Pages*

### Description

Sends asynchronous HTTP GET requests to fetch data from multiple API pages concurrently.

### Usage

```
get_async_pages(pages, nested)
```

### Arguments

| | |
|---|---|
| pages | Character vector of full URLs specifying the API endpoints to request. |
| nested | Logical value indicating whether to flatten nested lists in the JSON responses. |

### Details

This function uses `future.apply::future_lapply()` to perform concurrent HTTP GET requests for multiple pages. It retrieves and parses the JSON responses from each URL provided.

### Value

A list of parsed JSON responses from each page.

### Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## get_brapi_url *Get the BrAPI Endpoint URL for a given QBMS function*

### Description

Constructs the BrAPI endpoint URL for a given QBMS function based on the configured server, crop, and BrAPI version. The function name is mapped to the corresponding BrAPI call using internal mapping.

### Usage

```
get_brapi_url(func_name)
```

### Arguments

| | |
|---|---|
| func_name | (string) The name of the QBMS function for which the BrAPI endpoint URL is required. |

### Value

A string representing the BrAPI endpoint URL.

### Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

---

`get_germplasm_attributes` *Retrieve Attributes for a Specified Germplasm*

---

### Description

Retrieves a detailed list of attributes for a given germplasm, such as its origin, donors, and taxonomic information.

### Usage

```
get_germplasm_attributes(germplasm_name = "")
```

### Arguments

| | |
|---|---|
| `germplasm_name` | The name of the germplasm. |

### Value

A data frame containing the attributes associated with the specified germplasm.

### Author(s)

Johan Steven Aparicio, <j.aparicio@cgiar.org>

### See Also

`login`, `set_crop`, `get_germplasm_data`

### Examples

```
if (interactive()) {
  set_qbms_config("https://bms.icarda.org/ibpworkbench")
  login_bms()
  set_crop("wheat")
  germplasm_attributes <- get_germplasm_attributes("Jabal")
}
```

get_germplasm_data *Retrieve Observations Data for a Specified Germplasm.*

## Description

Retrieves all available observations data for the given germplasm in the current active crop. This data is aggregated across all trials in the crop database.

## Usage

```
get_germplasm_data(germplasm_name = "")
```

## Arguments

| | |
|---|---|
| germplasm_name | The name of the germplasm. |

## Value

A data frame containing all available observations data for the specified germplasm.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## See Also

login, set_crop, get_germplasm_attributes

## Examples

```
if (interactive()) {
  set_qbms_config("https://bms.icarda.org/ibpworkbench")
  login_bms()
  set_crop("wheat")
  germplasm_observations <- get_germplasm_data("Jabal")
  head(germplasm_observations)
}
```

## get_germplasm_id *Get Germplasm ID for a Specified Germplasm Name*

### Description

Retrieves the unique germplasm ID associated with the specified germplasm name for the current active crop.

### Usage

```
get_germplasm_id(germplasm_name = "")
```

### Arguments

| | |
|---|---|
| germplasm_name | The name of the germplasm. |

### Value

A string representing the germplasm's unique ID (germplasmDbId).

### Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

### See Also

set_crop, get_germplasm_data, get_germplasm_attributes

## get_germplasm_list *Get the Germplasm List of the Current Active Study*

### Description

Retrieves the list of germplasm (genetic material) used in the currently active study, which must be set using the set_study function.

### Usage

```
get_germplasm_list()
```

### Value

A data frame containing the germplasm list for the active study.

### Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## See Also

login, set_crop, set_program, set_trial, set_study for related operations on crops and studies.

## Examples

```
if (interactive()) {
  set_qbms_config("https://bms.icarda.org/ibpworkbench")
  login_bms()
  set_crop("wheat")
  set_program("Wheat International Nurseries")
  set_trial("IDYT39")
  set_study("IDYT39 Environment Number 9")
  germplasm <- get_germplasm_list()
  head(germplasm)
}
```

---

# get_hwsd2 *Get HWSD v2 Soil Data for a Given Location(s)*

---

## Description

Queries the HWSD v2 database to retrieve soil information for specific locations based on their coordinates. For each location, the function extracts the Soil Mapping Unit (SMU) code and retrieves soil attributes based on the specified sequence (soil dominance) and depth layer. The function returns the input data frame augmented with soil data from the HWSDv2 dataset.

The HWSD2_SMU table contains general information for each of the soil units occurring in any given SMU code (dominant soil unit and up to 11 associated soils).

The SEQUENCE column refers to the sequence in which soil units within the SMU are presented (in order of percentage share). The dominant soil has sequence 1. The sequence can range between 1 and 12.

The SHARE column refers to the share of the soil unit within the mapping unit in percentage. Shares of soil units within a mapping unit always sum up to 100 percent.

The HWSD2_LAYERS table provides soil attributes per depth layer for each of the seven depth layers (D1 to D7) separately (represented in the LAYER column in the HWSD2_LAYERS table). The depth of the top and bottom of each layer is defined in the TOPDEP and BOTDEP columns, respectively.

## Usage

```
get_hwsd2(df, con, x = "longitude", y = "latitude", sequence = 1, layer = "D1")
```

## Arguments

| | |
|---|---|
| df | A data frame containing location information, including longitude and latitude in decimal degrees. |
| con | The HWSDv2 object returned by the ini_hwsd2() function, containing the raster and SQLite connection. |
| x | The column name in the data frame representing longitude (default is 'Longitude'). |
| y | The column name in the data frame representing latitude (default is 'Latitude'). |
| sequence | Integer indicating the soil unit's dominance order within the SMU (default is 1 for the dominant soil). Valid values range from 1 to 12. |
| layer | String indicating the depth layer for which soil attributes should be retrieved (default is 'D1', with layers ranging from 'D1' to 'D7'). |

## Value

A data frame with the original location data augmented by soil attributes for the specified sequence and layer. The data frame includes additional columns such as 'smu_id', 'SEQUENCE', 'LAYER', and other soil attributes.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## See Also

ini_hwsd2

## Examples

```
if (interactive()) {
  Location  <- c('Tel-Hadya', 'Terbol', 'Marchouch')
  Latitude  <- c(36.016, 33.808, 33.616)
  Longitude <- c(36.943, 35.991, -6.716)
  sites     <- data.frame(Location, Latitude, Longitude)

  hwsd2 <- ini_hwsd2(data_path = 'C:/Users/Kel-shamaa/Downloads/HWSD v2/')
  sites <- get_hwsd2(df = sites, con = hwsd2, x = 'Longitude', y = 'Latitude',
                     sequence = 1, layer = 'D1')
}
```

## get_login_details *Login Pop-Up Window*

### Description

Opens a GUI pop-up window using Tcl/Tk to prompt the user for their username and password. The window title and prompt message adapt based on the type of server being used (e.g., BMS, GIGWA).

### Usage

```
get_login_details()
```

### Value

A vector containing the inserted username and password, with names `usr` and `pwd` respectively.

### Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## get_marker_map *Get marker position information*

### Description

Retrieves the genetic marker map associated with the currently active study. This includes the marker name, chromosome (or linkage group), and position.

### Usage

```
get_marker_map()
```

### Value

A data frame with three columns:

**rs#** Marker name (variant name)

**chrom** Chromosome or linkage group name

**pos** Genetic or physical position of the marker

### Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

### See Also

`set_qbms_config`, `set_study`

## get_parents *Get Direct Parents*

### Description

Utility function to split a given pedigree string and retrieve the pedigrees of the direct parents (female and male). The function handles different formats of cross representations, such as single slashes (/), double slashes (//), or numbered crosses (e.g., /3/). It extracts the highest cross order when available and returns the sub-pedigree for the immediate parents.

### Usage

```
get_parents(pedigree)
```

### Arguments

| | |
|---|---|
| pedigree | A string providing the parentage through which a cultivar was obtained. |

### Value

A vector of two items representing the direct female and male parents. If parent information is unavailable or unknown, 'NA' is returned for the respective parent.

### Author(s)

Khaled Al-Shamaa, <k.el-shamaa@cgiar.org>

## get_pedigree_table *Get the Pedigree Table*

### Description

Retrieves a comprehensive pedigree table for the given dataset, which contains genotype names and pedigree strings. The function recursively traces parentage across generations and builds a pedigree table where each row corresponds to an individual, with columns for the female and male parents. It also handles cases of similar genotype names by standardizing them.

### Usage

```
get_pedigree_table(
  data,
  geno_column = "germplasmName",
  pedigree_column = "pedigree"
)
```

## Arguments

| | |
|---|---|
| data | A data frame containing genotype/germplasm data, including names and pedigree strings. |
| geno_column | The name of the column that identifies the genotype/germplasm names. |
| pedigree_column | The name of the column that contains the pedigree strings. |

## Value

A data frame with three columns: - 'Variety': The identifier for the individual genotype. - 'Female': The identifier for the female parent. - 'Male': The identifier for the male parent. The pedigree table is sorted such that individuals appear before any row where they are listed as a parent. For founders (i.e., individuals with no parent information), 'NA' is used for the parental columns.

## Author(s)

Khaled Al-Shamaa, <k.el-shamaa@cgiar.org>

## Examples

```
if (interactive()) {
  set_qbms_config("https://bms.icarda.org/ibpworkbench")
  login_bms()
  set_crop("wheat")
  set_program("Wheat International Nurseries")
  set_trial("IDYT39")
  set_study("IDYT39 Environment Number 9")
  germplasm <- get_germplasm_list()
  pedigree_table <- get_pedigree_table(germplasm, "germplasmName", "pedigree")

  ############################
  # nadiv package way
  # library(nadiv)

  # Get additive relationship matrix in sparse matrix format
  # A <- nadiv::makeA(pedigree_table)

  # Get A inverse matrix using base R function
  # AINV <- solve(as.matrix(A))

  ############################
  # ASReml-R package way
  # library(asreml)

  # Represent A inverse matrix in an efficient way using i, j index and Ainverse value
  # Actual genotype names of any given index are in the attr(ainv, "rowNames")
  # ainv <- asreml::ainverse(pedigree_table)

  ############################
  # Dummy data set for testing
  test <- data.frame(genotype = c("X", "Y"),
```

```
                    pedigree = c("A//B/D/2/C", "B/C/3/A//B/C/2/D"))

  pedigree_table <- get_pedigree_table(test, "genotype", "pedigree")
}
```

---

`get_program_studies` *Get the List of Trials, Studies, and Locations Information for the Current Selected Program*

---

## Description

Retrieves comprehensive information about the trials, studies, and environments/locations within the current active breeding program, as configured in the internal state object using the `set_program` function. This includes test and check entry counts for each study.

## Usage

```
get_program_studies()
```

## Value

A data frame containing detailed information for each study within the program's trials, including trial names, study names, location information, and entry counts.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## See Also

`login`, `set_crop`, `set_program`

## Examples

```
if (interactive()) {
  set_qbms_config("https://bms.icarda.org/ibpworkbench")
  login_bms()
  set_crop("wheat")
  set_program("Wheat International Nurseries")
  program_studies <- get_program_studies()
  head(program_studies)
}
```

## get_program_trials *Retrieve the List of Trials for the Active Breeding Program*

### Description

This internal function retrieves the list of trials for the currently active breeding program and crop combination. The crop and program must be set using `set_crop` and `set_program` prior to calling this function.

### Usage

```
get_program_trials()
```

### Value

A data frame containing information on trials for the active breeding program.

### Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

### See Also

`login`, `set_crop`, `set_program`, `list_trials`

## get_qbms_connection *Get the QBMS Connection*

### Description

Retrieves the current QBMS connection object, which contains the server's configuration and state, including any active sessions and tokens. This can be used to save and restore connections between sessions.

### Usage

```
get_qbms_connection()
```

### Value

A list containing the current QBMS configuration and state.

### Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## See Also

set_qbms_connection

## Examples

```
if(interactive()) {
  set_qbms_config("https://bms.icarda.org/ibpworkbench")

  login_bms()
  set_crop("wheat")
  set_program("Wheat International Nurseries")

  df1  <- get_germplasm_data("Jabal")
  con1 <- get_qbms_connection()

  set_qbms_config("https://gigwa.southgreen.fr/gigwa/", engine = "gigwa", no_auth = TRUE)

  gigwa_set_db("DIVRICE_NB")
  gigwa_set_project("refNB")
  gigwa_set_run("03052022")

  df2  <- gigwa_get_metadata()
  con2 <- get_qbms_connection()

  set_qbms_connection(con1)
  df3 <- get_germplasm_attributes("Jabal")
}
```

---

# get_study_data *Get the Observations Data of the Current Active Study*

---

## Description

Retrieves the observations data (e.g., measurements, variables) for the active study, which must be set using the set_study function.

## Usage

```
get_study_data()
```

## Value

A data frame containing the observation data for the active study, or NULL if no data is available.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## See Also

login, set_crop, set_program, set_trial, set_study for related study operations.

## Examples

```
if (interactive()) {
  set_qbms_config("https://bms.icarda.org/ibpworkbench")
  login_bms()
  set_crop("wheat")
  set_program("Wheat International Nurseries")
  set_trial("IDYT39")
  set_study("IDYT39 Environment Number 9")
  data <- get_study_data()
  head(data)
}
```

---

# get_study_info *Get the Details/Metadata of the Current Active Study*

---

## Description

Retrieves detailed metadata for the currently active study. The study must be set using the set_study function, and its details will be fetched from the BrAPI server.

## Usage

```
get_study_info()
```

## Value

A data frame containing the metadata of the active study. Returns NULL if no study metadata is available.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## See Also

login, set_crop, set_program, set_trial, set_study for related crop and study management.

## Examples

```
if (interactive()) {
  set_qbms_config("https://bms.icarda.org/ibpworkbench")
  login_bms()
  set_crop("wheat")
  set_program("Wheat International Nurseries")
  set_trial("IDYT39")
  set_study("IDYT39 Environment Number 9")
  info <- get_study_info()
}
```

---

`get_terraclimate` *Get TerraClimate Data for a Given Coordinate(s)*

---

## Description

This function allows you to extract climate variables from the TerraClimate dataset for specific geographic coordinates. TerraClimate is a global dataset of monthly climate data covering the years 1958-present. The function retrieves climate variables directly from the hosting server provided by the University of Idaho, avoiding the need to download large raster files in netCDF format. Additionally, the function calculates bioclimatic variables using the `calc_biovars` function, derived from the dismo R package.

The TerraClimate dataset is compared with WorldClim in several aspects:

- TerraClimate: 1958-present vs. WorldClim: 1970-2000
- 14 climate variables vs. 7 climate variables in WorldClim
- Spatial resolution: ~4 km (TerraClimate) vs. ~1 km (WorldClim)
- Need to calculate bioclimatic variables (TerraClimate) vs. pre-calculated (WorldClim)

## Usage

```
get_terraclimate(
  lat,
  lon,
  from = "1958-01-01",
  to = "2022-12-31",
  clim_vars = NULL,
  month_mask = NULL,
  offline = FALSE,
  data_path = "./data/"
)
```

## Arguments

| | |
|---|---|
| `lat` | Vector of Latitude(s) in decimal degree format. Each entry corresponds to a location of interest. |
| `lon` | Vector of Longitude(s) in decimal degree format. Each entry corresponds to a location of interest. |
| `from` | Start date as a string in 'YYYY-MM-DD' format. Defines the beginning of the time range for data extraction. |
| `to` | End date as a string in 'YYYY-MM-DD' format. Defines the end of the time range for data extraction. |
| `clim_vars` | List of climate variables to extract. Valid options include: *aet, def, pet, ppt, q, soil, srad, swe, tmax, tmin, vap, ws, vpd, and PDSI*. Default is `NULL`, which retrieves all variables. |
| `month_mask` | A vector specifying the months of interest, e.g., for specific seasons (e.g., planting season: `c(10:12, 1:5)`). Default is `NULL`, which retrieves data for all months. |
| `offline` | Logical value indicating whether to extract TerraClimate data from pre-downloaded netCDF files. Default is `FALSE`, meaning data is fetched from the remote server. |
| `data_path` | String specifying the directory path where downloaded netCDF files are stored when working offline. Default is './data/'. |

## Value

A list of two data frames for each coordinate pair (latitude and longitude):

- **climate:** A data frame containing the requested climate variables for each month and location.
- **biovars:** A data frame with calculated bioclimatic variables, based on the extracted climate data.

Each data frame is in a format ready for further analysis in R.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## References

Abatzoglou, J., Dobrowski, S., Parks, S. *et al.* TerraClimate, a high-resolution global dataset of monthly climate and climatic water balance from 1958-2015. *Sci Data* **5**, 170191 (2018). doi:10.1038/sdata.2017.191

## See Also

`ini_terraclimate`, `calc_biovars`

## Examples

```
if (interactive()) {
  # data <- get_terraclimate(36.016, 36.943, '1979-09-01', '2012-06-30',
  #                          c('ppt', 'tmin', 'tmax'), c(10:12,1:5))
  data <- get_terraclimate(36.016, 36.943, '1979-09-01', '2012-06-30')

  View(data$climate[[1]])  # View the climate data
  View(data$biovars[[1]])  # View the bioclimatic variables
}
```

---

`get_trial_data` *Get the Observations Data of the Current Active Trial*

---

## Description

Retrieves the combined observations data (including all studies) for the current active trial, as configured in the internal state object using the `set_trial` function. This function iterates over all studies within the active trial and aggregates their observation data.

## Usage

```
get_trial_data()
```

## Value

A data frame containing the combined observations data from all studies in the active trial.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## See Also

`login`, `set_crop`, `set_program`, `set_trial`

## Examples

```
if (interactive()) {
  set_qbms_config("https://bms.icarda.org/ibpworkbench")
  login_bms()
  set_crop("wheat")
  set_program("Wheat International Nurseries")
  set_trial("IDYT39")
  MET <- get_trial_data()
  head(MET)
}
```

## get_trial_obs_ontology *Get the Traits Ontology/Metadata of the Current Active Trial*

### Description

Retrieves the traits ontology or metadata for the current active trial, which includes detailed information about the observation variables used in the trial.

### Usage

```
get_trial_obs_ontology()
```

### Value

A data frame containing the traits ontology or metadata, filtered by the observation variables used in the current trial.

### Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

### See Also

login, set_crop, set_program, set_trial, get_study_data for retrieving study observations.

### Examples

```
if (interactive()) {
  set_qbms_config("https://bms.icarda.org/ibpworkbench")
  login_bms()
  set_crop("wheat")
  set_program("Wheat International Nurseries")
  set_trial("IDYT39")
  ontology <- get_trial_obs_ontology()
}
```

## get_trial_pedigree *Get the Pedigree table for the Selected Trial*

### Description

Get the pedigree table representing the pedigree tree for the currently active trial. Each row corresponds to a germplasm entry and includes identifiers and details for both parents.

## Usage

```
get_trial_pedigree()
```

## Value

A data frame with germplasm and parent details for the selected trial.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## See Also

login, set_crop, set_program, set_trial for related operations on crops, programs, and trials.

---

# get_variants *Get Marker Matrix from the Selected Variant Set*

---

## Description

Get a two-dimensional marker matrix for all samples in the currently active variant set. This function is used as an alternative to the `get_variantset()` function when the server does not support it.

Note: This approach is significantly slower than `get_variantset` because it fetches data page by page through the BrAPI API, rather than downloading the full matrix in one Falpjack format file.

## Usage

```
get_variants()
```

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## See Also

set_qbms_config, set_variantset

## get_variantset *Get Marker Matrix from the Selected Variant Set*

### Description

Downloads a two-dimensional marker matrix for all samples in the currently active variant set, using the efficient Flapjack file format from the server. Row names correspond to marker IDs and column names correspond to unique sample IDs. Genotype calls are returned in character format.

### Usage

```
get_variantset()
```

### Value

A data frame of genotype calls in character format, with markers as rows and samples as columns.

### Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

### See Also

set_qbms_config, set_variantset

## gigwa_get_allelematrix *Get Markers Matrix in the Selected GIGWA Run*

### Description

Retrieve a two-dimensional matrix of genotype data from the selected GIGWA run. This matrix is returned based on filters for regions, samples, or variants. The data can be simplified to use numeric coding for genotypes, or returned in its raw VCF-like format.

### Usage

```
gigwa_get_allelematrix(
  samples = NULL,
  start = 0,
  end = "",
  chrom = NULL,
  snps = NULL,
  snps_pageSize = 10000,
  samples_pageSize = 100,
  simplify = TRUE
)
```

### Arguments

| | |
|---|---|
| samples | A list of sample names to include (optional). If NULL, all samples will be included. |
| start | Start position of the query region (zero-based, inclusive). |
| end | End position of the query region (zero-based, exclusive). |
| chrom | Reference sequence name (e.g., chromosome or contig). |
| snps | A list of variant names to filter (optional). |
| snps_pageSize | Number of variants to fetch per page (default is 10,000). |
| samples_pageSize | Number of samples to fetch per page (default is 100). |
| simplify | Whether to simplify the returned data using numeric coding (default is TRUE). |

### Value

A data frame with rows representing SNP markers and columns representing samples. Values are numeric codings (0: reference allele, 1: heterozygous, 2: alternative allele).

### Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

### Examples

```
if (interactive()) {
  set_qbms_config("https://gigwa.southgreen.fr/gigwa/",
                  time_out = 300, engine = "gigwa", no_auth = TRUE)
  gigwa_set_db("Sorghum-JGI_v1")
  gigwa_set_project("Nelson_et_al_2011")
  gigwa_set_run("run1")
  samples <- gigwa_get_samples()
  chroms <- gigwa_get_sequences()
  geno_data <- gigwa_get_allelematrix(samples = samples[1:5],
                                      start   = 0,
                                      end     = 1234567,
                                      chrom   = chroms[1:3])
}
```

---

## gigwa_get_markers *Get Markers Map in the Selected GIGWA Run*

---

### Description

Retrieve a filtered list of SNP variants from the selected run. This function allows users to query variants based on chromosomal regions and return results in simplified format.

## Usage

```
gigwa_get_markers(start = NULL, end = NULL, chrom = NULL, simplify = TRUE)
```

## Arguments

| | |
|---|---|
| start | Start position of the query region (zero-based, inclusive). |
| end | End position of the query region (zero-based, exclusive). |
| chrom | Reference sequence name (e.g., chromosome). |
| simplify | Logical, if TRUE (default) returns data in a simplified HapMap-like format with columns for rs#, alleles, chromosome, and position. |

## Value

A data frame of SNP markers, optionally simplified to include rs#, alleles, chromosome, and position.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## Examples

```
if (interactive()) {
  set_qbms_config("https://gigwa.southgreen.fr/gigwa/",
                  time_out = 300, engine = "gigwa", no_auth = TRUE)
  gigwa_set_db("Sorghum-JGI_v1")
  gigwa_set_project("Nelson_et_al_2011")
  gigwa_set_run("run1")
  chroms <- gigwa_get_sequences()
  geno_map <- gigwa_get_markers(start = 0, end = 12345678, chrom = chroms[7])
}
```

---

# gigwa_get_metadata *Get the Metadata of the Current Active GIGWA Run*

---

## Description

Retrieve metadata associated with the samples in the current active run, set using the gigwa_set_run() function. The metadata provides additional information about the samples in the selected run.

## Usage

```
gigwa_get_metadata()
```

## Value

A data frame containing metadata attributes for each sample in the active run.

### Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

### See Also

set_qbms_config, gigwa_set_run

### Examples

```
if (interactive()) {
  set_qbms_config("https://gigwa.southgreen.fr/gigwa/",
                  time_out = 300, engine = "gigwa", no_auth = TRUE)
  gigwa_set_db("3kG_10M")
  gigwa_set_project("3003_ind")
  gigwa_set_run("1")
  metadata <- gigwa_get_metadata()
}
```

---

## gigwa_get_samples *Get the Samples List of the Current Active GIGWA Project*

---

### Description

Retrieve the list of samples associated with the currently active GIGWA project, set using `gigwa_set_project()`.

### Usage

```
gigwa_get_samples()
```

### Value

A vector of sample names in the selected project.

### Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

### See Also

set_qbms_config, gigwa_set_project

### Examples

```
if (interactive()) {
  set_qbms_config("https://gigwa.southgreen.fr/gigwa/",
                  time_out = 300, engine = "gigwa", no_auth = TRUE)
  gigwa_set_db("Sorghum-JGI_v1")
  gigwa_set_project("Nelson_et_al_2011")
  samples <- gigwa_get_samples()
}
```

## gigwa_get_sequences *Get the Sequences of the Current Active GIGWA Project*

### Description

Retrieve the list of sequences (e.g., chromosomes) associated with the currently active project in GIGWA, which has been set using the 'gigwa_set_project()' function.

### Usage

```
gigwa_get_sequences()
```

### Value

A vector of sequence names (e.g., chromosome names) for the selected project.

### Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

### See Also

set_qbms_config, gigwa_set_project

### Examples

```
if (interactive()) {
  set_qbms_config("https://gigwa.southgreen.fr/gigwa/",
                  time_out = 300, engine = "gigwa", no_auth = TRUE)
  gigwa_set_db("Sorghum-JGI_v1")
  gigwa_set_project("Nelson_et_al_2011")
  chroms <- gigwa_get_sequences()
}
```

## gigwa_get_variants *Get Available Variants in the Selected GIGWA Run*

### Description

Retrieve variant data (e.g., SNP markers) for the selected GIGWA run based on filtering criteria, including minor allele frequency, missing data threshold, and sample subset.

## Usage

```
gigwa_get_variants(
  max_missing = 1,
  min_maf = 0.5,
  samples = NULL,
  start = NULL,
  end = NULL,
  referenceName = NULL
)
```

## Arguments

| | |
|---|---|
| max_missing | The maximum allowable missing data ratio, between 0 and 1 (default is 1, meaning up to 100% missing data). |
| min_maf | Minimum Minor Allele Frequency (MAF) between 0 and 0.5 (default is 0). |
| samples | A list of sample names to include in the query (optional). If NULL, all samples will be included. |
| start | Start position of the query region (zero-based, inclusive). |
| end | End position of the query region (zero-based, exclusive). |
| referenceName | The reference sequence name (e.g., chromosome). |

## Value

A data frame where the first 4 columns describe the SNP (rs# variant name, alleles, chrom, pos), and subsequent columns contain numerical genotyping information (0 for reference allele, 1 for heterozygous, and 2 for minor allele).

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## Examples

```
if (interactive()) {
  set_qbms_config("https://gigwa.southgreen.fr/gigwa/",
                  time_out = 300, engine = "gigwa", no_auth = TRUE)
  gigwa_set_db("Sorghum-JGI_v1")
  gigwa_set_project("Nelson_et_al_2011")
  gigwa_set_run("run1")
  marker_matrix <- gigwa_get_variants(max_missing = 0.2,
                                      min_maf = 0.35,
                                      samples = c("ind1", "ind3", "ind7"))
}
```

## gigwa_list_dbs *List GIGWA Databases*

### Description

Retrieve the list of available databases from the connected GIGWA server. An active connection is required. If not connected, the function will throw an error.

### Usage

```
gigwa_list_dbs()
```

### Value

A list of databases available on the connected GIGWA server.

### Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

### See Also

set_qbms_config

### Examples

```
if (interactive()) {
  set_qbms_config("https://gigwa.southgreen.fr/gigwa/",
                  time_out = 300, engine = "gigwa", no_auth = TRUE)
  gigwa_list_dbs()
}
```

## gigwa_list_projects *Get the List of All Projects in the Selected GIGWA Database*

### Description

Retrieve the list of projects available in the currently active GIGWA database, set using `gigwa_set_db()`. If no database is selected, the function will throw an error.

### Usage

```
gigwa_list_projects()
```

### Value

A list of project names in the selected database.

### Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

### See Also

set_qbms_config, gigwa_set_db

### Examples

```
if (interactive()) {
  set_qbms_config("https://gigwa.southgreen.fr/gigwa/",
                  time_out = 300, engine = "gigwa", no_auth = TRUE)
  gigwa_set_db("Sorghum-JGI_v1")
  gigwa_list_projects()
}
```

---

## gigwa_list_runs — *Get the List of the Run Names Available in the Selected GIGWA Project*

---

### Description

Retrieve the list of available runs in the currently active GIGWA project, set using `gigwa_set_project()`. If no project is selected, an error will be raised.

### Usage

```
gigwa_list_runs()
```

### Value

A list of run names associated with the selected project.

### Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

### See Also

set_qbms_config, gigwa_set_project

### Examples

```
if (interactive()) {
  set_qbms_config("https://gigwa.southgreen.fr/gigwa/",
                  time_out = 300, engine = "gigwa", no_auth = TRUE)
  gigwa_set_db("Sorghum-JGI_v1")
  gigwa_set_project("Nelson_et_al_2011")
  gigwa_list_runs()
}
```

## gigwa_set_db *Set the Current Active GIGWA Database by Name*

### Description

Select a GIGWA database as the active database for subsequent operations. This updates the internal configuration object and resets any previously selected projects or runs.

### Usage

```
gigwa_set_db(db_name)
```

### Arguments

| | |
|---|---|
| db_name | The name of the database to set as active. |

### Value

No return value. Updates the internal configuration with the selected database.

### Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

### See Also

set_qbms_config, gigwa_list_dbs

### Examples

```
if (interactive()) {
  set_qbms_config("https://gigwa.southgreen.fr/gigwa/",
                  time_out = 300, engine = "gigwa", no_auth = TRUE)
  gigwa_set_db("Sorghum-JGI_v1")
}
```

## gigwa_set_project *Set the Current Active GIGWA Project*

### Description

Select a project from the active GIGWA database and set it as the current active project in the internal state. This selection is used for retrieving related data, such as runs or samples.

### Usage

```
gigwa_set_project(project_name)
```

### Arguments

| | |
|---|---|
| project_name | The name of the project to set as active. |

### Value

No return value. Updates the internal state with the selected project.

### Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

### See Also

set_qbms_config, gigwa_set_db, gigwa_list_projects

### Examples

```
if (interactive()) {
  set_qbms_config("https://gigwa.southgreen.fr/gigwa/",
                  time_out = 300, engine = "gigwa", no_auth = TRUE)
  gigwa_set_db("Sorghum-JGI_v1")
  gigwa_set_project("Nelson_et_al_2011")
}
```

---

## gigwa_set_run *Set the Current Active GIGWA Run*

---

### Description

Select a run from the active GIGWA project and set it as the current active run in the internal state, enabling further data retrieval operations.

### Usage

```
gigwa_set_run(run_name)
```

### Arguments

| | |
|---|---|
| run_name | The name of the run to set as active. |

### Value

No return value. Updates the internal state with the selected run.

### Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## See Also

set_qbms_config, gigwa_set_project, gigwa_list_runs

## Examples

```
if (interactive()) {
  set_qbms_config("https://gigwa.southgreen.fr/gigwa/",
                  time_out = 300, engine = "gigwa", no_auth = TRUE)
  gigwa_set_db("Sorghum-JGI_v1")
  gigwa_set_project("Nelson_et_al_2011")
  gigwa_set_run("run1")
}
```

---

# ini_hwsd2 *Download and Setup HWSD v2.0 Data Files to Extract their Data Offline*

---

## Description

Downloads and sets up the HWSD v2.0 data files required to extract soil data offline. The function retrieves the HWSD raster soil unit map and the SQLite database containing soil attributes. If the files already exist in the specified directory, they are used directly. The function returns an object with the raster and SQLite connection for further queries.

## Usage

```
ini_hwsd2(data_path = "./data/", timeout = 300)
```

## Arguments

| | |
|---|---|
| data_path | String specifying the directory path where HWSD v2.0 data files are stored or should be downloaded (default is './data/'). |
| timeout | Timeout in seconds for downloading each HWSD v2.0 data file (default is 300). |

## Value

A list object ('con') containing two items: - 'raster': HWSDv2 raster object for spatial queries. - 'sqlite': Connection to the HWSDv2 SQLite database.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## See Also

get_hwsd2

## Examples

```
if (interactive()) {
  hwsd2 <- ini_hwsd2(data_path = 'C:/Users/Kel-shamaa/Downloads/HWSD v2/')
}
```

---

`ini_terraclimate` *Download TerraClimate netCDF Data Files to Extract their Data Offline*

---

## Description

This function facilitates the download of TerraClimate netCDF files for a specified time period and climate variables. TerraClimate data provides monthly climate data for global terrestrial surfaces, and this function allows you to store the data locally for offline extraction and analysis without the need to download the entire dataset.

Users can specify a range of climate variables such as precipitation, temperature, evapotranspiration, soil moisture, and more. The downloaded files are saved in netCDF format, making them accessible for subsequent offline analysis.

## Usage

```
ini_terraclimate(
  from = "2019-09-01",
  to = "2022-06-30",
  clim_vars = c("ppt", "tmin", "tmax"),
  data_path = "./data/",
  timeout = 300
)
```

## Arguments

| | |
|---|---|
| `from` | Start date as a string in the 'YYYY-MM-DD' format. This defines the beginning of the time period for which you want to download the climate data. |
| `to` | End date as a string in the 'YYYY-MM-DD' format. This defines the end of the time period for which you want to download the climate data. |
| `clim_vars` | A list of climate variables to download. Valid options include: *aet* (Actual Evapotranspiration), *def* (Climate Water Deficit), *pet* (Potential Evapotranspiration), *ppt* (Precipitation), *q* (Runoff), *soil* (Soil Moisture), *srad* (Solar Radiation), *swe* (Snow Water Equivalent), *tmax* (Maximum Temperature), *tmin* (Minimum Temperature), *vap* (Vapor Pressure), *ws* (Wind Speed), *vpd* (Vapor Pressure Deficit), and *PDSI* (Palmer Drought Severity Index). If NULL (default), all available variables will be downloaded. |
| `data_path` | A string containing the directory path where the downloaded netCDF files will be stored. The default path is './data/'. |
| `timeout` | Timeout in seconds for downloading each netCDF raster file. The default value is 300 seconds. |

## Value

No explicit return value. The downloaded netCDF files are saved to the specified directory for offline use with the get_terraclimate function to extract data for a given coordinate or set of coordinates.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## See Also

get_terraclimate

## Examples

```
if (interactive()) {
  # Initialize TerraClimate data download for specific climate variables between two dates
  ini_terraclimate('2018-09-01', '2019-06-30', c('ppt', 'tmin', 'tmax'))

  # Coordinates for the location(s) of interest
  x <- c(-6.716, 35.917, 76.884)
  y <- c(33.616, 33.833, 23.111)

  # Extract TerraClimate data for the specified coordinates (online mode)
  a <- get_terraclimate(y, x, '2018-09-01', '2019-06-30', c('ppt', 'tmin', 'tmax'))

  # View the extracted climate data and bioclimatic variables
  View(a$climate[[1]])
  View(a$biovars[[1]])

  # Extract TerraClimate data for the specified coordinates (offline mode)
 b <- get_terraclimate(y, x, '2018-09-01', '2019-06-30', c('ppt', 'tmin', 'tmax'), offline = TRUE)

  # View the offline-extracted data
  View(b$climate[[1]])
  View(b$biovars[[1]])
}
```

---

# list_crops *Retrieve Supported Crops from the Server*

---

## Description

Retrieves the list of crops supported by the connected server. If the crop list is cached in the internal state, it returns the cached data; otherwise, it sends a BrAPI GET request to fetch the crop list.

## Usage

```
list_crops()
```

## Value

A character vector containing the names of supported crops.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## See Also

login, set_crop to configure and set the current active crop.

## Examples

```
if (interactive()) {
  set_qbms_config("https://bms.icarda.org/ibpworkbench")
  login_bms()  # Log in to the server
  list_crops()  # Retrieve list of supported crops
}
```

---

# list_locations *Get the List of Locations Information of the Current Selected Crop*

---

## Description

Retrieves a list of locations associated with the current active crop, as configured in the internal state object using the set_crop function.

## Usage

```
list_locations()
```

## Value

A data frame containing information about locations relevant to the current crop.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## See Also

login, set_crop for related crop operations.

## list_programs *Retrieve Breeding Programs for the Active Crop*

### Description

Retrieves the list of breeding programs available for the currently selected crop. The crop must be set using the set_crop function prior to calling this.

### Usage

```
list_programs()
```

### Value

A data frame containing the names of breeding programs available for the active crop.

### Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

### See Also

login, set_crop, list_crops for managing server connection and crop selection.

### Examples

```
if (interactive()) {
  set_qbms_config("https://bms.icarda.org/ibpworkbench")
  login_bms()  # Log in to the server
  set_crop("wheat")  # Set "wheat" as the active crop
  list_programs()  # Retrieve breeding programs for the active crop
}
```

## list_studies *Get the List of Studies in the Current Active Trial*

### Description

Retrieves a list of studies (and associated locations) for the currently active trial, as configured using the set_trial function.

### Usage

```
list_studies()
```

### Value

A data frame containing study names and associated location names. If no studies are available, an error is thrown.

### Note

This function must be called after a trial has been set using `set_trial`.

### Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

### See Also

`login`, `set_crop`, `set_program`, `set_trial` for related operations on crops, programs, and trials.

### Examples

```
if (interactive()) {
  set_qbms_config("https://bms.icarda.org/ibpworkbench")
  login_bms()  # Log in to the server
  set_crop("wheat")  # Set crop
  set_program("Wheat International Nurseries")  # Set breeding program
  set_trial("IDYT39")  # Set trial
  list_studies()  # List studies
}
```

---

## `list_trials` *List Trials in the Current Active Breeding Program*

---

### Description

Retrieves the list of trials for the current active breeding program. Optionally, filters trials by their starting year if specified.

### Usage

```
list_trials(year = NULL)
```

### Arguments

| | |
|---|---|
| `year` | Numeric. An optional parameter to filter trials by their starting year. If not provided, all trials for the active program are returned. |

### Value

A data frame containing the names of trials for the active breeding program. If no trials match the query, a warning is issued, and NA is returned.

## Note

The year filter is only supported for BMS databases.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## See Also

login, set_crop, set_program for related operations involving crop and program selection.

## Examples

```
if (interactive()) {
  set_qbms_config("https://bms.icarda.org/ibpworkbench")
  login_bms()  # Log in to the server
  set_crop("wheat")  # Set crop
  set_program("Wheat International Nurseries")  # Set breeding program
  list_trials()  # List trials
  list_trials(2022)  # List trials starting in 2022
}
```

---

# list_variantsets *List Variant Sets in the Selected Study*

---

## Description

Retrieves the names of all variant sets available in the currently selected study, as set by `set_study()`. If no study is selected, the function returns an error.

## Usage

```
list_variantsets()
```

## Value

A list of the names of variant sets associated with the selected study.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## See Also

set_qbms_config, set_study

## login *Login to a BrAPI Server*

### Description

Authenticates with a BrAPI server using a username and password. If credentials are not provided, interactive prompt will request them from the user. This function serves as a wrapper for all `login_*` authentication methods and accepts additional parameters via `...`

### Usage

```
login(username = NULL, password = NULL, ...)
```

### Arguments

| | |
|---|---|
| `username` | The username (optional, default is NULL). |
| `password` | The password (optional, default is NULL). |
| `...` | Additional arguments passed to `login_oauth2`. |

### Value

No return value. On success, the access token is stored internally for future use.

### Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

### See Also

`login_oauth2`

## login_bms *Login to the Server*

### Description

Connects to the BMS or related server using a username and password. If these are not provided, a pop-up window will prompt the user to enter their credentials. The function handles authentication and stores the resulting access token internally for subsequent requests.

### Usage

```
login_bms(username = NULL, password = NULL, encoding = "json")
```

### Arguments

| | |
|---|---|
| username | The username (optional, default is NULL). If not provided, the pop-up window is triggered. |
| password | The password (optional, default is NULL). If not provided, the pop-up window is triggered. |
| encoding | Specifies how the request body should be encoded: `form` (application/x-www-form-urlencoded), `multipart` (multipart/form-data), or `json` (application/json). Default is "json". |

### Value

No return value. The access token is stored internally for future use.

### Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

### Examples

```
if(interactive()) {
  set_qbms_config("https://bms.icarda.org/ibpworkbench", engine = "bms")

  # Login using your BMS account (interactive mode)
  login_bms()

  # You can pass BMS username and password as parameters (batch mode)
  # login_bms("username", "password")
}
```

---

## login_breedbase *Login to the BreedBase Server*

---

### Description

Logs in to the BreedBase server using a username and password. If credentials are not provided, a pop-up window will prompt the user. The function is a wrapper around the `login_bms()` function, with encoding set to `form`.

### Usage

```
login_breedbase(username = NULL, password = NULL)
```

### Arguments

| | |
|---|---|
| username | The username (optional, default is NULL). |
| password | The password (optional, default is NULL). |

## Value

No return value. The access token is stored internally for future use.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## Examples

```
if(interactive()) {
  set_qbms_config("https://cassavabase.org/", engine = "breedbase")

  # Login using your BreedBase account (interactive mode)
  login_breedbase()

   # You can pass BreedBase username and password as parameters (batch mode)
   # login_breedbase("username", "password")
}
```

---

`login_germinate` *Login to the Germinate Server*

---

## Description

Login to the Germinate Server

## Usage

```
login_germinate(username = NULL, password = NULL)
```

## Arguments

| | |
|---|---|
| `username` | The username (optional, default is NULL). |
| `password` | The password (optional, default is NULL). |

## Value

No return value. The access token is stored internally for future use.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

login_gigwa *Login to the GIGWA Server*

## Description

Connect to the GIGWA server. If the username or password parameters are missing, a login window will be triggered to capture these details.

All connection settings (server URL, port, API path, and protocol) are read from the qbms_config() list. The function will request an authentication token from the server and update the qbms_state() list with the token.

## Usage

```
login_gigwa(username = NULL, password = NULL)
```

## Arguments

| | |
|---|---|
| username | The GIGWA username (optional, default is NULL). |
| password | The GIGWA password (optional, default is NULL). |

## Value

No return value. The authentication token will be stored internally.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## Examples

```
if (interactive()) {
 set_qbms_config("http://localhost:59395/gigwa/index.jsp", time_out = 300, engine = "gigwa")

  # Login using your GIGWA account (interactive mode)
  login_gigwa()

  # You can pass GIGWA username and password as parameters (batch mode)
  # login_gigwa("gigwadmin", "nimda")
}
```

`login_oauth2` *Login using OAuth 2.0 Authentication*

## Description

Performs OAuth 2.0 authentication by sending the user to the authorization URL and exchanging the authorization code for an access token. This function supports caching of tokens for subsequent requests.

## Usage

```
login_oauth2(
  authorize_url,
  access_url,
  client_id,
  client_secret = NULL,
  redirect_uri = "http://localhost:1410",
  scope = NULL
)
```

## Arguments

| | |
|---|---|
| `authorize_url` | The URL where the client is redirected for user authorization. |
| `access_url` | The URL used to exchange an authorization code for an access token. |
| `client_id` | The client ID (consumer key) provided by the authorization server. |
| `client_secret` | The client secret provided by the authorization server (optional). |
| `redirect_uri` | The URL where the user will be redirected after authorization (default is http://localhost:1410). |
| `scope` | Scopes to be requested from the resource owner. |

## Value

No return value. Updates the internal state with the access token and additional details.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## rbindlistx *Make One Data.Table from a List of Many*

### Description

Performs the equivalent of do.call("rbind", x) on data.frames, but much faster.

### Usage

```
rbindlistx(x)
```

### Arguments

| | |
|---|---|
| x | A list containing data.table, data.frame, or list objects. |

### Value

An unkeyed data.table containing a concatenation of all the items passed in.

## rbindx *Combine Data Frames by Row, Filling in Missing Columns*

### Description

Combines a list of data frames by row, filling in missing columns with NA.

### Usage

```
rbindx(..., dfs = list(...))
```

### Arguments

| | |
|---|---|
| ... | The first argument data frame. |
| dfs | Input data frames to row bind together. |

### Value

A single data frame.

## scan_brapi_endpoints *Scan BrAPI Endpoints*

### Description

Scans the available BrAPI endpoints on the configured source server and checks their accessibility. This function allows users to verify which BrAPI endpoints are available based on the provided IDs.

### Usage

```
scan_brapi_endpoints(programDbId = 0, trialDbId = 0, studyDbId = 0)
```

### Arguments

| | |
|---|---|
| programDbId | (numeric) The programDbId to scan specific program-related endpoints (default is 0). |
| trialDbId | (numeric) The trialDbId to scan specific trial-related endpoints (default is 0). |
| studyDbId | (numeric) The studyDbId to scan specific study-related endpoints (default is 0). |

### Value

A data frame listing the QBMS function, BrAPI endpoint URL, and availability status for each endpoint.

### Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## set_crop *Set the Current Active Crop*

### Description

Updates the internal configuration to set the selected crop as the active one. This must be called before performing crop-specific operations such as retrieving breeding programs.

### Usage

```
set_crop(crop_name)
```

### Arguments

| | |
|---|---|
| crop_name | A string specifying the name of the crop to set as active. |

## Value

No return value. The function updates the global state with the selected crop.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## See Also

login, list_crops to validate and retrieve the list of supported crops.

## Examples

```
if (interactive()) {
  set_qbms_config("https://bms.icarda.org/ibpworkbench")
  login_bms()  # Log in to the server
  set_crop("wheat")  # Set "wheat" as the active crop
}
```

---

# set_program *Set the Current Active Breeding Program*

---

## Description

Updates the internal state to set the selected breeding program as active using the associated programDbId. This allows subsequent operations to be carried out within the context of this program.

## Usage

```
set_program(program_name)
```

## Arguments

| | |
|---|---|
| program_name | A string specifying the name of the breeding program to set as active. |

## Value

No return value. The internal state is updated with the selected program.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## See Also

login, set_crop, list_programs for related operations in the crop and program selection process.

## Examples

```
if (interactive()) {
  set_qbms_config("https://bms.icarda.org/ibpworkbench")
  login_bms()  # Log in to the server
  set_crop("wheat")  # Set crop
  set_program("Wheat International Nurseries")  # Set breeding program
}
```

---

`set_qbms_config` *Configure BMS Server Settings*

---

## Description

Configures the BMS server connection settings, including URL, API path, page size, and timeout. This function allows you to set up the connection for different server backends like BMS, Gigwa, EBS, and Breedbase, and choose the appropriate BrAPI version.

## Usage

```
set_qbms_config(
  url = "http://localhost",
  path = NULL,
  page_size = 1000,
  time_out = 120,
  no_auth = FALSE,
  engine = "bms",
  brapi_ver = "v1",
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| `url` | The URL of the BMS login page or API base (default is "http://localhost"). |
| `path` | The API path to use (default is NULL, which sets a path based on the engine). |
| `page_size` | The number of records per page when making API calls (default is 1000). |
| `time_out` | The maximum number of seconds to wait for a response (default is 120). |
| `no_auth` | Logical, whether the server requires authentication (default is FALSE). |
| `engine` | The backend system (default is "bms"). Options include "bms", "gigwa", "breedbase", "ebs", "germinate". |
| `brapi_ver` | The version of BrAPI to use, either "v1" or "v2" (default is "v1"). |
| `verbose` | Logical, indicating whether to display progress information when making API calls (default is TRUE). |

## Value

No return value.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## Examples

```
set_qbms_config("https://bms.icarda.org/ibpworkbench")
```

---

`set_qbms_connection` *Set the QBMS Connection*

---

## Description

Sets the QBMS connection object in the current environment, allowing users to restore a saved connection, including configuration settings and session tokens.

## Usage

```
set_qbms_connection(env)
```

## Arguments

| | |
|---|---|
| `env` | A list containing the saved connection configuration and state. |

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## See Also

`get_qbms_connection`

## Examples

```
if(interactive()) {
  set_qbms_config("https://bms.icarda.org/ibpworkbench")

  login_bms()
  set_crop("wheat")
  set_program("Wheat International Nurseries")

  df1  <- get_germplasm_data("Jabal")
  con1 <- get_qbms_connection()

 set_qbms_config("https://gigwa.southgreen.fr/gigwa/", engine = "gigwa", no_auth = TRUE)
```

```
  gigwa_set_db("DIVRICE_NB")
  gigwa_set_project("refNB")
  gigwa_set_run("03052022")

  df2  <- gigwa_get_metadata()
  con2 <- get_qbms_connection()

  set_qbms_connection(con1)
  df3 <- get_germplasm_attributes("Jabal")
}
```

---

`set_study` *Set the Current Active Study*

---

## Description

Updates the internal state to set the selected study as the current active study using the associated studyDbId. This allows operations to be performed within the context of the selected study.

## Usage

```
set_study(study_name)
```

## Arguments

| | |
|---|---|
| `study_name` | A string specifying the name of the study to set as active. |

## Value

No return value. The internal state is updated with the selected study.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## See Also

`login`, `set_crop`, `set_program`, `set_trial`, `list_studies` for related operations on crops, programs, trials, and studies.

## Examples

```
if (interactive()) {
  set_qbms_config("https://bms.icarda.org/ibpworkbench")
  login_bms()  # Log in to the server
  set_crop("wheat")  # Set crop
  set_program("Wheat International Nurseries")  # Set breeding program
```

```
  set_trial("IDYT39")  # Set trial
  set_study("IDYT39 Environment Number 9")  # Set study
}
```

---

`set_token` *Set Access Token Response*

---

## Description

Stores the access token and associated details (such as username and expiration time) in the internal state. The token is typically retrieved from the server during login and used for subsequent API requests.

## Usage

```
set_token(token, user = "", expires_in = 3600)
```

## Arguments

| | |
|---|---|
| `token` | The access token string issued by the authorization server. |
| `user` | The username associated with the token (optional). |
| `expires_in` | The lifetime of the access token in seconds (optional, default is 3600 seconds). |

## Value

No return value. Updates the internal state with the token info.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

---

`set_trial` *Set the Current Active Trial*

---

## Description

Updates the internal state to set the selected trial as the current active trial using the associated trialDbId. This enables operations to be carried out within the context of the selected trial.

## Usage

```
set_trial(trial_name)
```

## Arguments

| | |
|---|---|
| `trial_name` | A string specifying the name of the trial to set as active. |

## Value

No return value. The internal state is updated with the selected trial.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## See Also

login, set_crop, set_program, list_trials for operations involving crops, programs, and trials.

## Examples

```
if (interactive()) {
  set_qbms_config("https://bms.icarda.org/ibpworkbench")
  login_bms()  # Log in to the server
  set_crop("wheat")  # Set crop
  set_program("Wheat International Nurseries")  # Set breeding program
  set_trial("IDYT39")  # Set trial
}
```

---

# set_variantset *Set the Active Variant Set*

---

## Description

Selects a variant set by name from the currently active study and updates the internal state to make it the active variant set. This selection is required for subsequent data retrieval operations related to variant sets.

## Usage

```
set_variantset(variantset_name)
```

## Arguments

| | |
|---|---|
| variantset_name | The name of the variant set to be active. |

## Value

No return value. Updates the internal state with the selected variant set.

## Author(s)

Khaled Al-Shamaa (<k.el-shamaa@cgiar.org>)

## See Also

set_qbms_config, set_study, list_variantsets

# Index