Package: hubUtils (via r-universe)

September 7, 2024

Title Utility Functions for Infectious Disease Modeling Hubs

Version 0.1.2

Description A set of utility functions for downloading, plotting, and scoring forecast and truth data from Infectious Disease Modeling Hubs.

License MIT + file LICENSE

URL https://github.com/hubverse-org/hubUtils, https://hubverse-org.github.io/hubUtils/

BugReports https://github.com/hubverse-org/hubUtils/issues

Depends R (>= 2.10)

Imports checkmate, cli, curl, fs, gh, glue, jsonlite, lifecycle, magrittr, memoise, purrr, rlang, stringr, tibble, utils

Suggests dplyr, hubData, knitr, rmarkdown, testthat (>= 3.2.0)

Additional_repositories https://hubverse-org.r-universe.dev/

Remotes hubverse-org/hubData

Config/Needs/website hubverse-org/hubStyle

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Repository https://hubverse-org.r-universe.dev

RemoteUrl https://github.com/hubverse-org/hubUtils

RemoteRef HEAD

RemoteSha 0c8b37aa2e20172cf85df76ebe3fbd74adab5fb6

Contents

`as_model_out_tbl` *Convert model output to a* `model_out_tbl` *class object.*

Description

Convert model output to a `model_out_tbl` class object.

Usage

```
as_model_out_tbl(
  tbl,
  model_id_col = NULL,
  output_type_col = NULL,
  output_type_id_col = NULL,
  value_col = NULL,
  sep = "-",
  trim_to_task_ids = FALSE,
  hub_con = NULL,
  task_id_cols = NULL,
  remove_empty = FALSE
)
```

Arguments

tbl	a data.frame or tibble of model output data returned from a query to a <hub_connection> object.
model_id_col	character string. If a model_id column does not already exist in tbl, the tbl column name containing model_id data. Alternatively, if both a team_abbr and a model_abbr column exist, these will be merged automatically to create a single model_id column.
output_type_col	character string. If an output_type column does not already exist in tbl, the tbl column name containing output_type data.
output_type_id_col	character string. If an output_type_id column does not already exist in tbl, the tbl column name containing output_type_id data.
value_col	character string. If a value column does not already exist in tbl, the tbl column name containing value data.
sep	character string. Character used as separator when concatenating team_abbr and model_abbr column values into a single model_id string. Only applicable if model_id column not present and team_abbr and model_abbr columns are.
trim_to_task_ids	logical. Whether to trim tbl to task ID columns only. Task ID columns can be specified by providing a <hub_connection> class object to hub_con or manually through task_id_cols.
hub_con	a <hub_connection> class object. Only used if trim_to_task_ids = TRUE and tasks IDs should be determined from the hub config.
task_id_cols	a character vector of column names. Only used if trim_to_task_ids = TRUE to manually specify task ID columns to retain. Overrides hub_con argument if provided.
remove_empty	Logical. Whether to remove columns containing only NA.

Value

A model_out_tbl class object.

Examples

```
if (requireNamespace("hubData", quietly = TRUE)) {
  library(dplyr)
  hub_path <- system.file("testhubs/flusight", package = "hubUtils")
  hub_con <- hubData::connect_hub(hub_path)
  hub_con %>%
    filter(output_type == "quantile", location == "US") %>%
    collect() %>%
    filter(forecast_date == max(forecast_date)) %>%
    as_model_out_tbl()
}
```

check_deprecated_schema

Check whether a config file is using a deprecated schema

Description

Function compares the current schema version in a config file to a valid version, If config file version deprecated compared to valid version, the function issues a lifecycle warning to prompt user to upgrade.

Usage

```
check_deprecated_schema(
  config_version,
  config,
  valid_version = "v2.0.0",
  hubutils_version = "0.0.0.9010"
)
```

Arguments

`config_version`	Character string of the schema version.
`config`	List representation of config file.
`valid_version`	Character string of minimum valid schema version.
`hubutils_version`	The version of the hubUtils package in which deprecation of the schema version below `valid_version` is introduced.

Value

Invisibly, `TRUE` if the schema version is deprecated, `FALSE` otherwise. Primarily used for the side effect of issuing a lifecycle warning.

extract_schema_version

Extract the schema version from a schema `id` *or config* `schema_version` *property character string*

Description

Extract the schema version from a schema `id` or config `schema_version` property character string

Usage

```
extract_schema_version(id)
```

Arguments

id	A schema id or config schema_version property character string.

Value

The schema version number as a character string.

---

get_config_tid *Get the name of the output type id column based on the schema version*

Description

Version can be provided either directly through the config_version argument or extracted from a config_tasks object.

Usage

```
get_config_tid(config_version, config_tasks)
```

Arguments

config_version	Character string of the schema version.
config_tasks	a list version of the content's of a hub's tasks.json config file, accessed through the "config_tasks" attribute of a <hub_connection> object or function read_config().

Value

character string of the name of the output type id column

Examples

```
get_config_tid("v1.0.0")
get_config_tid("v2.0.0")
```

get_round_idx *Utilities for accessing round ID metadata*

Description

Utilities for accessing round ID metadata

Usage

```
get_round_idx(config_tasks, round_id)

get_round_ids(
  config_tasks,
  flatten = c("all", "model_task", "task_id", "none")
)
```

Arguments

config_tasks	a list version of the content's of a hub's `tasks.json` config file, accessed through the `"config_tasks"` attribute of a <hub_connection> object or function `read_config()`.
round_id	Character string. Round identifier. If the round is set to `round_id_from_variable: true`, IDs are values of the task ID defined in the round's `round_id` property of `config_tasks`. Otherwise should match round's `round_id` value in config. Ignored if hub contains only a single round.
flatten	Character. Whether and how much to flatten output. • `"all"`: Complete flattening. Returns a character vector of unique round IDs across all rounds. • `"model_task"`: Flatten model tasks. Returns a list with an element for each round. Each round element contains a character vector of unique round IDs across all round model tasks. Only applicable if `round_id_from_variable` is `TRUE`. • `"task_id"`: Flatten task ID. Returns a nested list with an element for each round. Each round element contains a list with an element for each model task. Each model task element contains a character vector of unique round IDs. across `required` and `optional` properties. Only applicable if `round_id_from_variable` is `TRUE` • `"none"`: No flattening. If `round_id_from_variable` is `TRUE`, returns a nested list with an element for each round. Each round element contains a nested element for each model task. Each model task element contains a nested list of `required` and `optional` character vectors of round IDs. If `round_id_from_variable` is `FALSE`,a list with a round ID for each round is returned.

Value

the integer index of the element in `config_tasks$rounds` that a character round identifier maps to

a list or character vector of hub round IDs

- A character vector is returned only if `flatten = "all"`
- A list is returned otherwise (see `flatten` for more details)

Functions

- `get_round_idx()`: Get an integer index of the element in `config_tasks$rounds` that a character round identifier maps to.
- `get_round_ids()`: Get a list or character vector of hub round IDs. For each round, if `round_id_from_variable` is `TRUE`, round IDs returned are the values of the task ID defined in the `round_id` property. Otherwise, if `round_id_from_variable` is `FALSE`, the value of the `round_id` property is returned.

Examples

```
if (requireNamespace("hubData", quietly = TRUE)) {
  hub_con <- hubData::connect_hub(system.file("testhubs/simple",
                                              package = "hubUtils"))
  config_tasks <- attr(hub_con, "config_tasks")
  # Get round IDs
  get_round_ids(config_tasks)
  get_round_ids(config_tasks, flatten = "model_task")
  get_round_ids(config_tasks, flatten = "task_id")
  get_round_ids(config_tasks, flatten = "none")
  # Get round integer index using a round_id
  get_round_idx(config_tasks, "2022-10-01")
  get_round_idx(config_tasks, "2022-10-29")
}
```

---

get_round_model_tasks *Get the model tasks for a given round*

Description

Get the model tasks for a given round

Usage

```
get_round_model_tasks(config_tasks, round_id)
```

Arguments

config_tasks	a list version of the content's of a hub's tasks.json config file, accessed through the "config_tasks" attribute of a <hub_connection> object or function read_config().
round_id	Character string. Round identifier. If the round is set to round_id_from_variable: true, IDs are values of the task ID defined in the round's round_id property of config_tasks. Otherwise should match round's round_id value in config. Ignored if hub contains only a single round.

Value

a list representation of model tasks for a given round.

Examples

```
hub_path <- system.file("testhubs/simple", package = "hubUtils")
config_tasks <- read_config(hub_path, "tasks")
get_round_model_tasks(config_tasks, round_id = "2022-10-08")
get_round_model_tasks(config_tasks, round_id = "2022-10-15")
```

---

get_round_task_id_names

Get task ID names for a given round

Description

Get task ID names for a given round

Usage

```
get_round_task_id_names(config_tasks, round_id)
```

Arguments

config_tasks	a list version of the content's of a hub's tasks.json config file, accessed through the "config_tasks" attribute of a <hub_connection> object or function read_config().
round_id	Character string. Round identifier. If the round is set to round_id_from_variable: true, IDs are values of the task ID defined in the round's round_id property of config_tasks. Otherwise should match round's round_id value in config. Ignored if hub contains only a single round.

Value

a character vector of task ID names

Examples

```
hub_path <- system.file("testhubs/simple", package = "hubUtils")
config_tasks <- read_config(hub_path, "tasks")
get_round_task_id_names(config_tasks, round_id = "2022-10-08")
get_round_task_id_names(config_tasks, round_id = "2022-10-15")
```

`get_schema` *Download a schema*

Description

Download a schema

Usage

```
get_schema(schema_url)
```

Arguments

`schema_url`	The download URL for a given config schema version.

Value

Contents of the JSON schema as a character string.

See Also

Other functions supporting config file validation: `get_schema_url()`, `get_schema_valid_versions()`

Examples

```
schema_url <- get_schema_url(config = "tasks", version = "v0.0.0.9")
get_schema(schema_url)
```

`get_schema_url` *Get the JSON schema download URL for a given config file version*

Description

Get the JSON schema download URL for a given config file version

Usage

```
get_schema_url(config = c("tasks", "admin", "model"), version, branch = "main")
```

Arguments

config	Name of config file to validate. One of "tasks" or "admin".
version	A valid version of hubverse schema (e.g. "v0.0.1").
branch	The branch of the hubverse schemas repository from which to fetch schema. Defaults to "main".

Value

The JSON schema download URL for a given config file version.

See Also

Other functions supporting config file validation: get_schema(), get_schema_valid_versions()

Examples

```
get_schema_url(config = "tasks", version = "v0.0.0.9")
```

---

get_schema_valid_versions

Get a vector of valid schema version

Description

Get a vector of valid schema version

Usage

```
get_schema_valid_versions(branch = "main")
```

Arguments

branch	The branch of the hubverse schemas repository from which to fetch schema. Defaults to "main".

Value

a character vector of valid versions of hubverse schema.

See Also

Other functions supporting config file validation: get_schema(), get_schema_url()

Examples

```
get_schema_valid_versions()
```

get_schema_version_latest *Get the latest schema version*

Description

Get the latest schema version from the schema repository if "latest" requested (default) or ignore if specific version provided.

Usage

```
get_schema_version_latest(schema_version = "latest", branch = "main")
```

Arguments

schema_version	A character vector. Either "latest" or a valid schema version.
branch	The branch of the hubverse schemas repository from which to fetch schema. Defaults to "main".

Value

a schema version string. If schema_version is "latest", the latest schema version from the schema repository. If specific version provided to schema_version, the same version is returned.

Examples

```
get_schema_version_latest()
get_schema_version_latest(schema_version = "v1.0.0")
```

get_task_id_names *Get hub task IDs*

Description

Get hub task IDs

Usage

```
get_task_id_names(config_tasks)
```

Arguments

config_tasks	a list version of the content's of a hub's tasks.json config file, accessed through the "config_tasks" attribute of a <hub_connection> object or function read_config().

Value

a character vector of all unique task ID names across all rounds.

Examples

```
hub_path <- system.file("testhubs/simple", package = "hubUtils")
config_tasks <- read_config(hub_path, "tasks")
get_task_id_names(config_tasks)
```

---

is_v3_config *Is config list representation using v3.0.0 schema?*

Description

Is config list representation using v3.0.0 schema?

Usage

```
is_v3_config(config)
```

Arguments

config	List representation of the JSON config file.

Value

Logical, whether the config list representation is using v3.0.0 schema.

Examples

```
config <- read_config_file(
  system.file("config", "tasks.json", package = "hubUtils")
)
is_v3_config(config)
```

is_v3_config_file *Is config file using v3.0.0 schema?*

Description

Is config file using v3.0.0 schema?

Usage

```
is_v3_config_file(config_path)
```

Arguments

config_path	Path to the config file.

Value

Logical, whether the config file is using v3.0.0 schema.

Examples

```
config_path <- system.file("config", "tasks.json", package = "hubUtils")
is_v3_config_file(config_path)
```

is_v3_hub *Is hub configured using v3.0.0 schema?*

Description

Is hub configured using v3.0.0 schema?

Usage

```
is_v3_hub(hub_path, config = c("tasks", "admin"))
```

Arguments

hub_path	Either a character string path to a local Modeling Hub directory or an object of class <SubTreeFileSystem> created using functions hubData::s3_bucket() or hubData::gs_bucket() by providing a string S3 or GCS bucket name or path to a Modeling Hub directory stored in the cloud. For more details consult the Using cloud storage (S3, GCS) in the arrow package.
config	Name of config file to validate. One of "tasks" or "admin".

Value

Logical, whether the hub is configured using v3.0.0 schema.

Examples

```
is_v3_hub(hub_path = system.file("testhubs", "flusight", package = "hubUtils"))
```

`model_id_merge` *Merge/Split model output tbl* `model_id` *column*

Description

Merge/Split model output tbl `model_id` column

Usage

```
model_id_merge(tbl, sep = "-")

model_id_split(tbl, sep = "-")
```

Arguments

`tbl`	a `data.frame` or `tibble` of model output data returned from a query to a `<hub_connection>` object.
`sep`	character string. Character used as separator when concatenating `team_abbr` and `model_abbr` values into a single `model_id` string or splitting `model_id` into component `team_abbr` and `model_abbr`. When splitting, if multiple instances of the separator exist in a `model_id` stringing, splitting occurs occurs on the first instance.

Value

`tbl` with either `team_abbr` and `model_abbr` merged into a single `model_id` column or `model_id` split into columns `team_abbr` and `model_abbr`.

a tibble with `model_id` column split into separate `team_abbr` and `model_abbr` columns

Functions

- `model_id_merge()`: merge `team_abbr` and `model_abbr` into a single `model_id` column.
- `model_id_split()`: split `model_id` column into separate `team_abbr` and `model_abbr` columns.

Examples

```
if (requireNamespace("hubData", quietly = TRUE)) {
  hub_con <- hubData::connect_hub(system.file("testhubs/flusight", package = "hubUtils"))
  tbl <- hub_con %>%
    dplyr::filter(output_type == "quantile", location == "US") %>%
    dplyr::collect() %>%
    dplyr::filter(forecast_date == max(forecast_date))

  tbl_split <- model_id_split(tbl)
```

```
  tbl_split

  # Merge model_id
  tbl_merged <- model_id_merge(tbl_split)
  tbl_merged

  # Split / Merge using custom separator
  tbl_sep <- tbl
  tbl_sep$model_id <- gsub("-", "_", tbl_sep$model_id)
  tbl_sep <- model_id_split(tbl_sep, sep = "_")
  tbl_sep
  tbl_sep <- model_id_merge(tbl_sep, sep = "_")
  tbl_sep
}
```

`read_config` *Read a hub config file into R*

Description

Read a hub config file into R

Usage

```
read_config(hub_path, config = c("tasks", "admin", "model-metadata-schema"))
```

Arguments

`hub_path`	Either a character string path to a local Modeling Hub directory or an object of class `<SubTreeFileSystem>` created using functions `hubData::s3_bucket()` or `hubData::gs_bucket()` by providing a string S3 or GCS bucket name or path to a Modeling Hub directory stored in the cloud. For more details consult the Using cloud storage (S3, GCS) in the `arrow` package.
`config`	Name of config file to validate. One of `"tasks"` or `"admin"`.

Value

The contents of the config as an R list.

Examples

```
# Read config files from local hub
hub_path <- system.file("testhubs/simple", package = "hubUtils")
read_config(hub_path, "tasks")
read_config(hub_path, "admin")


# Read config file from AWS S3 bucket hub
hub_path <- hubData::s3_bucket("hubverse/hubutils/testhubs/simple/")
```

```
read_config(hub_path, "admin")
```

`read_config_file` *Read a JSON config file from a path*

Description

Read a JSON config file from a path

Usage

```
read_config_file(config_path)
```

Arguments

`config_path`	path to JSON config file

Value

a list representation of the JSON config file

Examples

```
read_config_file(system.file("config", "tasks.json", package = "hubUtils"))
```

`std_colnames` *Hubverse model output standard column names*

Description

A named character string of standard column names used in hubverse model output data files. The terms currently used for standard column names in the hubverse are English. In future, however, this could be expanded to provide the basis for hub terminology localisation.

Usage

```
std_colnames
```

Format

An object of class `character` of length 4.

validate_model_out_tbl *Validate a* model_out_tbl *object.*

Description

Validate a model_out_tbl object.

Usage

```
validate_model_out_tbl(tbl)
```

Arguments

tbl	a model_out_tbl S3 class object.

Value

If valid, returns a model_out_tbl class object. Otherwise, throws an error.

Examples

```
if (requireNamespace("hubData", quietly = TRUE)) {
  library(dplyr)
  hub_path <- system.file("testhubs/flusight", package = "hubUtils")
  hub_con <- hubData::connect_hub(hub_path)
  md_out <- hub_con %>%
    filter(output_type == "quantile", location == "US") %>%
    collect() %>%
    filter(forecast_date == max(forecast_date)) %>%
    as_model_out_tbl()

  validate_model_out_tbl(md_out)
}
```

Index