Package 'hubUtils'

October 29, 2026

Version 1.3.0

Title Core 'hubverse' Utilities

Description Core set of low-level utilities common across the 'hubverse'. Used to interact with 'hubverse' schema, Hub configuration files and model outputs and designed to be primarily used internally by other 'hubverse' packages. See Reich et al. (2022) <doi:10.2105/AJPH.2022.306831> for an overview of Collaborative 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 (>= 4.1.0)

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

Suggests arrow (>= 17.0.0), dplyr, knitr, rmarkdown, testthat (>= 3.2.0), withr

Config/Needs/website hubverse-org/hubStyle

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

NeedsCompilation no

Author Anna Krystalli [aut, cre] (ORCID: <https://orcid.org/0000-0002-2378-4915>), Li Shandross [aut], Nicholas G. Reich [ctb] (ORCID: <https://orcid.org/0000-0003-3503-9899>), Evan L. Ray [ctb], Zhian N. Kamvar [ctb] (ORCID: <https://orcid.org/0000-0003-1458-7108>), Consortium of Infectious Disease Modeling Hubs [cph]

Maintainer Anna Krystalli <annakrystalli@googlemail.com>

Repository CRAN

Date/Publication 2026-10-29 09:30:02 UTC

Contents

as_config . . . 3
as_model_out_tbl . . . 3
check_deprecated_schema . . . 5
convert_output_type . . . 6
create_s3_url . . . 7
extract_schema_version . . . 8
get_config_tid . . . 8
get_derived_task_ids_config . . . 9
get_hub_timezone . . . 10
get_round_config . . . 12
get_round_idx . . . 12
get_round_model_tasks . . . 14
get_round_output_type_names . . . 15
get_round_task_id_names . . . 15
get_round_value_sets_config . . . 16
get_schema . . . 18
get_schema_url . . . 19
get_schema_valid_versions . . . 20
get_schema_version_latest . . . 20
get_task_id_names . . . 21
get_task_id_type . . . 22
get_task_id_values . . . 23
get_version_config . . . 24
hub_con_output . . . 25
is_github_repo_url . . . 26
is_github_url . . . 27
is_s3_base_fs . . . 27
is_url . . . 28
is_v3_config . . . 29
is_v3_config_file . . . 29
is_v3_hub . . . 30
is_valid_url . . . 30
model_id_merge . . . 31
read_config . . . 32
read_config_file . . . 33
read_model_metadata . . . 34
std_colnames . . . 35
subset_task_id_cols . . . 35
subset_task_id_names . . . 36
target-data-utils . . . 36
validate_model_out_tbl . . . 39

version_equal 39

Index **42**

`as_config` *Coerce a config list to a config class object*

Description

Coerce a config list to a config class object

Usage

```
as_config(x)
```

Arguments

`x`	a list representation of the contents a `tasks.json` config file.

Value

a config list object with subclass `<config>`.

Examples

```
config_tasks <- read_config(
  hub_path = system.file("testhubs/simple", package = "hubUtils")
)
# Remove all attributes except names to demonstrate functionality
attributes(config_tasks) <- attributes(config_tasks)[
  names(attributes(config_tasks)) == "names"
]
# Convert to config object
as_config(config_tasks)
```

`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

```
as_model_out_tbl(hub_con_output)
```

`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.

convert_output_type *Transform between output types*

Description

Transform between output types for each unique combination of task IDs for each model. Conversion must be from a single initial output type to one or more to output types, and the resulting output will only contain the to output types. See details for supported conversions.

Usage

```
convert_output_type(model_out_tbl, to)
```

Arguments

model_out_tbl	an object of class model_out_tbl containing predictions with a single, unique value in the output_type column.
to	a named list indicating the desired output types and associated output type IDs. List item name and value pairs may be as follows: • mean: NA (no associated output type ID) • median: NA (no associated output type ID) • quantile: a numeric vector of probability levels OR a dataframe of probability levels and the task ID variables they depend upon. (See examples section for an example of each.) Note that any task ID variable value must appear in the associated model_out_tbl task ID column

Details

Currently, only "sample" can be converted to "mean", "median", or "quantile"

Value

object of class model_out_tbl containing (only) predictions of the to output_type(s) for each unique combination of task IDs for each model

Examples

```
# We illustrate the conversion between output types using normal distributions
ex_quantiles <- c(0.25, 0.5, 0.75)
model_out_tbl <- expand.grid(
  stringsAsFactors = FALSE,
  group1 = c(1, 2),
  model_id = "A",
  output_type = "sample",
  output_type_id = 1:100
) |>
  dplyr::mutate(value = rnorm(200, mean = group1))
```

```
# Output type conversions with vector `to` elements
convert_output_type(model_out_tbl,
  to = list("quantile" = ex_quantiles, "median" = NA)
)

# Output type conversion with dataframe `to` element
# Output type ID values (quantile levels) are determined by group1 value
quantile_levels <- rbind(
  data.frame(group1 = 1, output_type_id = 0.5),
  data.frame(group1 = 2, output_type_id = c(0.25, 0.5, 0.75))
)
convert_output_type(model_out_tbl,
  to = list("quantile" = quantile_levels)
)
```

`create_s3_url` *Create a URL to a file in an S3 bucket*

Description

Create a URL to a file in an S3 bucket

Usage

```
create_s3_url(base_fs, base_path)
```

Arguments

`base_fs`	character string. Path of the base s3 file system (bucket) in the cloud. Can be extracted from the object of class `<SubTreeFileSystem>` using the `$base_fs` field, followed by the `$base_path`.
`base_path`	character string. Path to the file in relation to `base_fs`. Can be extracted from the object of class `<SubTreeFileSystem>` using the `$base_path`.

Value

A character string of the URL to the file in s3.

Examples

```
create_s3_url(
  base_fs = "hubverse/hubutils/testhubs/simple/",
  base_path = "hub-config/admin.json"
)

# Create a URL from an object of class `<SubTreeFileSystem>` of an s3 hub
hub_path <- arrow::s3_bucket("hubverse/hubutils/testhubs/simple/")
```

```
create_s3_url(hub_path$base_path, "hub-config/admin.json")
config_path <- hub_path$path("hub-config/admin.json")
# Create a URL from an object of class `<SubTreeFileSystem>` of the path to
# a config file in an s3 hub
create_s3_url(config_path$base_fs$base_path, config_path$base_path)
```

`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.

Examples

```
extract_schema_version("schema_version: v3.0.0")
extract_schema_version("refs/heads/main/v3.0.0")
```

`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("v3.0.0")
get_config_tid("v2.0.0")

# this will produce a warning because support for schema version 1.0.0
# has been dropped.
get_config_tid("v1.0.0")
```

---

get_derived_task_ids_config

Get hub or round level derived task IDs

Description

get_config_derived_task_ids() is an alias of get_derived_task_ids_config().

Usage

```
get_derived_task_ids_config(
  config_tasks,
  round_id = NULL,
  call = rlang::current_env()
)

get_config_derived_task_ids(
  config_tasks,
  round_id = NULL,
  call = rlang::current_env()
)
```

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.

call — The execution environment of the function to name in error and warning messages. By default, messages name this function. Supply another environment, such as a wrapper function's environment, to name that function instead.

Value

a character vector of hub or round level derived task ID names. If round_id is NULL or the round does not have a round level derived_task_ids setting, returns the hub level derived_task_ids setting.

See Also

get_hub_derived_task_ids() to get derived task IDs from a hub path.

Examples

```
hub_path <- system.file("testhubs/v4/flusight", package = "hubUtils")
config_tasks <- read_config(hub_path, "tasks")
get_derived_task_ids_config(config_tasks)
get_derived_task_ids_config(config_tasks, round_id = "2023-05-08")
```

---

get_hub_timezone *Get hub configuration fields*

Description

Get hub configuration fields

Usage

```
get_hub_timezone(hub_path)

get_hub_model_output_dir(hub_path)

get_hub_file_formats(hub_path, round_id = NULL, call = rlang::current_env())

get_hub_derived_task_ids(
  hub_path,
  round_id = NULL,
  call = rlang::current_env()
)
```

Arguments

hub_path — Either a character string path to a local Modeling Hub directory, a character string of a URL to a GitHub repository or an object of class <SubTreeFileSystem> created using functions arrow::s3_bucket() or arrow::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.

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.

call — The execution environment of the function to name in error and warning messages. By default, messages name this function. Supply another environment, such as a wrapper function's environment, to name that function instead.

Value

- get_hub_timezone: The timezone of the hub
- get_hub_model_output_dir: The model output directory name
- get_hub_file_formats: character vector accepted hub or round level file formats. If round_id is NULL or the round does not have a round level file_format setting, returns the hub level file_format setting.
- get_hub_derived_task_ids: character vector of hub or round level derived task ID names. If round_id is NULL or the round does not have a round level derived_task_ids setting, returns the hub level derived_task_ids setting. See get_derived_task_ids_config() to get the same from a tasks config object.

Functions

- get_hub_timezone(): Get the hub timezone
- get_hub_model_output_dir(): Get the model output directory name
- get_hub_file_formats(): Get the hub or round level file formats
- get_hub_derived_task_ids(): Get the hub or round level derived_task_ids

Examples

```
hub_path <- system.file("testhubs", "flusight", package = "hubUtils")
get_hub_timezone(hub_path)
get_hub_model_output_dir(hub_path)
get_hub_file_formats(hub_path)
get_hub_file_formats(hub_path, "2022-12-12")
```

get_round_config *Get the configuration of a given round*

Description

Get the configuration of a given round

Usage

```
get_round_config(config_tasks, round_id, call = rlang::current_env())
```

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.
call	The execution environment of the function to name in error and warning messages. By default, messages name this function. Supply another environment, such as a wrapper function's environment, to name that function instead.

Value

a list representation of the round's element in the `rounds` property of the tasks config.

Examples

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

get_round_idx *Utilities for accessing round ID metadata*

Description

Utilities for accessing round ID metadata

Usage

```
get_round_idx(config_tasks, round_id, call = rlang::current_env())

get_round_ids(
  config_tasks,
  flatten = c("all", "model_task", "task_id", "none"),
  call = rlang::current_env()
)
```

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.
call	The execution environment of the function to name in error and warning messages. By default, messages name this function. Supply another environment, such as a wrapper function's environment, to name that function instead.
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

```
config_tasks <- read_config(
  hub_path = system.file("testhubs/simple", package = "hubUtils")
)
# 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, call = rlang::current_env())
```

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.
call	The execution environment of the function to name in error and warning messages. By default, messages name this function. Supply another environment, such as a wrapper function's environment, to name that function instead.

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_output_type_names`

Get output type names for a given round

Description

Get output type names for a given round

Usage

```
get_round_output_type_names(
  config_tasks,
  round_id,
  call = rlang::current_env()
)
```

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.
`call`	The execution environment of the function to name in error and warning messages. By default, messages name this function. Supply another environment, such as a wrapper function's environment, to name that function instead.

Value

a character vector of output type names

Examples

```
hub_path <- system.file("testhubs/v6/target_dir", package = "hubUtils")
config_tasks <- read_config(hub_path, "tasks")
get_round_output_type_names(config_tasks, round_id = "2022-10-22")
```

`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, call = rlang::current_env())
```

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.
call	The execution environment of the function to name in error and warning messages. By default, messages name this function. Supply another environment, such as a wrapper function's environment, to name that function instead.

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_round_value_sets_config

Get the values each modeling task in a round allows

Description

Reads a round of a tasks.json config into one list per modeling task. Each list holds the values that modeling task allows in each task ID column and, for each output type, in the output_type_id column:

Usage

```
get_round_value_sets_config(
  config_tasks,
  round_id,
  required_vals_only = FALSE,
  force_output_types = FALSE,
  output_types = NULL,
  derived_task_ids = NULL,
  call = rlang::current_env()
)
```

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.
required_vals_only	Logical. Whether to return only required values.
force_output_types	Logical. Whether to treat all output types as required, regardless of their is_required setting. Only affects the result when required_vals_only = TRUE.
output_types	Character vector of output type names to include. If NULL, all output types in the round are included.
derived_task_ids	Character vector of derived task ID names (task IDs whose values depend on other task IDs). Their values are returned as NA.
call	The execution environment of the function to name in error and warning messages. By default, messages name this function. Supply another environment, such as a wrapper function's environment, to name that function instead.

Details

```
[[1]]
  $task_ids
    $target   "wk flu hosp rate category"
    $horizon  0 1 2 3
    $location "US" "01" "02" ...
  $output_type_ids
    $pmf      "low" "moderate" "high" ...
```

The values are read from the config as follows:

- required and optional values are combined into one vector, required values first. With required_vals_only = TRUE, only required values are kept.
- A task ID a modeling task does not use is either listed as null or left out of that modeling task. Both are returned as NA, the value model output holds in that column. So are task IDs that have no required values when required_vals_only = TRUE, and derived task IDs.
- In a round with round_id_from_variable: true, the task ID holding round IDs is set to round_id.
- Output type IDs are read the same way for every schema version. From schema v4.0.0 onwards, an output type's is_required setting determines whether its output type IDs are required. Output types with no output type ID values are returned as NA. For mean and median, NA is the only valid value. For sample, NA stands in for sample IDs, which the config does not list.

Values keep the type they have in the config.

Value

A list with one element per modeling task in the round. Each element is a list of two:

- `task_ids`: a named list with a vector of allowed values for each task ID in the round.
- `output_type_ids`: a named list with a vector of allowed output type ID values for each output type in the modeling task. An output type with no `required` values is left out when `required_vals_only = TRUE`.

Examples

```
hub_path <- system.file("testhubs/v6/target_dir", package = "hubUtils")
config_tasks <- read_config(hub_path)
get_round_value_sets_config(config_tasks, round_id = "2022-10-22")
# Set derived task IDs to NA
get_round_value_sets_config(
  config_tasks,
  round_id = "2022-10-22",
  derived_task_ids = "target_end_date"
)
# Required values of a single output type only
get_round_value_sets_config(
  config_tasks,
  round_id = "2022-10-22",
  required_vals_only = TRUE,
  output_types = "pmf"
)
```

`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", "target-data"),
  version,
  branch = "main"
)
```

Arguments

`config`	Name of config file to validate. One of `"tasks"`, `"admin"`, `"model"` or `"target-data"`.
`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 the latest version of the schema

get_schema_version_latest()
get_schema_version_latest(schema_version = "v3.0.0")
```

---

get_task_id_names *Get hub task IDs*

Description

Get hub task IDs

Usage

```
get_task_id_names(config_tasks, call = rlang::current_env())
```

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().
call	The execution environment of the function to name in error and warning messages. By default, messages name this function. Supply another environment, such as a wrapper function's environment, to name that function instead.

Value

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

See Also

get_round_task_id_names() for the task ID names of a single round.

Examples

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

`get_task_id_type` *Get the data type of a task ID*

Description

The data type is the simplest R type that represents every value the task ID takes in the config. Character values that are all ISO 8601 dates (`YYYY-MM-DD`) are reported as `"Date"`.

Usage

```
get_task_id_type(
  config_tasks,
  task_id_name,
  round_id = NULL,
  call = rlang::current_env()
)
```

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()`.
`task_id_name`	character string. The name of a task ID.
`round_id`	character string. Round identifier. If `NULL`, the default, values are collected across all rounds.
`call`	The execution environment of the function to name in error and warning messages. By default, messages name this function. Supply another environment, such as a wrapper function's environment, to name that function instead.

Value

a character string, one of `"character"`, `"double"`, `"integer"`, `"logical"` or `"Date"`. `NA` if the task ID's `required` and `optional` values are both `null` in every modeling task in scope, so the config declares no type for it there.

See Also

`get_task_id_values()` for the values the type is derived from.

Examples

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

---

get_task_id_values *Get the values a task ID takes*

Description

Collects the `required` and `optional` values of a task ID from every modeling task in the config, or in one round.

Usage

```
get_task_id_values(
  config_tasks,
  task_id_name,
  round_id = NULL,
  call = rlang::current_env()
)
```

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()`.
`task_id_name`	character string. The name of a task ID.
`round_id`	character string. Round identifier. If `NULL`, the default, values are collected across all rounds.
`call`	The execution environment of the function to name in error and warning messages. By default, messages name this function. Supply another environment, such as a wrapper function's environment, to name that function instead.

Value

a vector of the unique values the task ID takes, with the type the values were read from the config as. Dates are character strings. `NULL` if the task ID's `required` and `optional` values are both `null` in every modeling task in scope.

Examples

```
hub_path <- system.file("testhubs/simple", package = "hubUtils")
config_tasks <- read_config(hub_path, "tasks")
get_task_id_values(config_tasks, "origin_date")
get_task_id_values(config_tasks, "origin_date", round_id = "2022-10-01")
get_task_id_values(config_tasks, "horizon")
```

`get_version_config` *Get hub config schema versions*

Description

Get hub config schema versions

Usage

```
get_version_config(config)

get_version_file(config_path)

get_version_hub(hub_path, config_type = c("tasks", "admin", "target-data"))
```

Arguments

`config`	A <config> class object. Usually the output of `read_config` or `read_config_file`.
`config_path`	Either a character string of a path to a local JSON config file, a character string of the URL to the **raw contents** of a JSON config file (e.g on GitHub) or an object of class <SubTreeFileSystem> created using functions `arrow::s3_bucket()` and associated methods for creating paths to JSON config files within the bucket.
`hub_path`	Either a character string path to a local Modeling Hub directory, a character string of a URL to a GitHub repository or an object of class <SubTreeFileSystem> created using functions `arrow::s3_bucket()` or `arrow::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_type`	Character vector specifying the type of config file to read. One of "tasks", "admin" or "target-data". Default is "tasks".

Value

The schema version number as a character string.

Functions

- `get_version_config()`: Get schema version from config list representation.
- `get_version_file()`: Get schema version from config file at specific path.
- `get_version_hub()`: Get schema version from config file at specific path.

Examples

```
config <- read_config_file(
  system.file("config", "tasks.json", package = "hubUtils")
)
get_version_config(config)
config_path <- system.file("config", "tasks.json", package = "hubUtils")
get_version_file(config_path)
# Get version from a URL of a hub config file
url <- paste0(
  "https://raw.githubusercontent.com/hubverse-org/",
  "example-simple-forecast-hub/refs/heads/main/hub-config/tasks.json"
)
get_version_file(url)

# Get version from an AWS S3 cloud hub config file
hub_path <- arrow::s3_bucket("hubverse/hubutils/testhubs/simple/")
config_path <- hub_path$path("hub-config/admin.json")
get_version_file(config_path)

hub_path <- system.file("testhubs/simple", package = "hubUtils")
get_version_hub(hub_path)
get_version_hub(hub_path, "admin")

# Get version from an AWS S3 cloud hub config file
hub_path <- arrow::s3_bucket("hubverse/hubutils/testhubs/simple/")
get_version_hub(hub_path)
```

---

hub_con_output *Example Hub model output data*

Description

A subset of model output data accessed using `hubData` from the simple example hub contained in the `hubUtils` package. The subset consists of `"quantile"` output type data for `"US"` location and the most recent forecast date.

Usage

```
hub_con_output
```

Format

A `tbl` with 92 rows and 8 columns:

- `forecast_date`: Origin date of the forecast.
- `horizon`: Forecast horizon relative to the `forecast_date`.
- `target`: Target variable.

- `location`: Location of the forecast.
- `output_type`: Output type of forecast.
- `output_type_id`: Forecast output type level/identifier. In this case, quantile level.
- `value`: Forecast value.
- `model_id`: Model identifier.

`is_github_repo_url` *Detect if a URL is a GitHub repository URL*

Description

Detect if a URL is a GitHub repository URL

Usage

```
is_github_repo_url(url)
```

Arguments

`url`	character string of the URL to check.

Value

Logical. `TRUE` if the URL is a GitHub repository URL, `FALSE` otherwise.

Examples

```
is_github_repo_url("https://github.com/hubverse-org/example-simple-forecast-hub")
raw_url <- paste0(
  "https://raw.githubusercontent.com/hubverse-org/",
  "example-simple-forecast-hub/refs/heads/main/hub-config/tasks.json"
)
is_github_repo_url(raw_url)
url_to_blob <- "https://github.com/hubverse-org/example-simple-forecast-hub/blob/main/README.md"
is_github_repo_url(url_to_blob)
```

is_github_url *Detect a URL on github.com*

Description

Detect a URL on github.com

Usage

```
is_github_url(url)
```

Arguments

url	character string of the URL to check.

Value

Logical. TRUE if the URL on github.com, FALSE otherwise.

Examples

```
# Returns TRUE
is_github_url("https://github.com/hubverse-org/example-simple-forecast-hub")
is_github_url("https://github.com/hubverse-org/schemas/tree/main/v5.0.0")
# Returns FALSE
is_github_url("https://gitlab.com/hubverse-org/schemas/tree/main/v5.0.0")
raw_url <- paste0(
  "https://raw.githubusercontent.com/hubverse-org/",
  "example-simple-forecast-hub/refs/heads/main/hub-config/tasks.json"
)
is_github_url(raw_url)
```

is_s3_base_fs *Detect whether An object of class* <SubTreeFileSystem> *represents the base path of an S3 file system (i.e. the root of a cloud hub)*

Description

Detect whether An object of class <SubTreeFileSystem> represents the base path of an S3 file system (i.e. the root of a cloud hub)

Usage

```
is_s3_base_fs(s3_fs)
```

Arguments

s3_fs	An object of class <SubTreeFileSystem>.

Value

Logical. TRUE if the object represents the base path of an S3 file, FALSE otherwise.

Examples

```
hub_path <- arrow::s3_bucket("hubverse/hubutils/testhubs/simple/")
config_path <- hub_path$path("hub-config/admin.json")
is_s3_base_fs(hub_path)
is_s3_base_fs(config_path)
```

---

is_url *Determine if a string is a URL*

Description

Determine if a string is a URL

Usage

```
is_url(x)
```

Arguments

x	character string to check if it is a URL. Must contain a protocol to be considered a URL.

Value

Logical. TRUE if x is a URL, FALSE otherwise.

Examples

```
is_url("https://docs.hubverse.io")
is_url("www.hubverse.io")
```

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 or greater.

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	Either a character string of a path to a local JSON config file, a character string of the URL to the **raw contents** of a JSON config file (e.g on GitHub) or an object of class <SubTreeFileSystem> created using functions arrow::s3_bucket() and associated methods for creating paths to JSON config files within the bucket.

Value

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

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", "target-data"))
```

Arguments

`hub_path`	Either a character string path to a local Modeling Hub directory, a character string of a URL to a GitHub repository or an object of class `<SubTreeFileSystem>` created using functions `arrow::s3_bucket()` or `arrow::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`	Type of config file to read. One of `"tasks"`, `"admin"` or `"model-metadata-schema"`. Default is `"tasks"`.

Value

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

Examples

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

`is_valid_url` *Determine if a URL is valid and reachable*

Description

Determine if a URL is valid and reachable

Usage

```
is_valid_url(url)
```

Arguments

url	character string of the URL to check.

Value

Logical. TRUE if the URL is valid and reachable, FALSE otherwise.

Examples

```
is_valid_url("https://docs.hubverse.io")
is_valid_url("https://docs.hubverse.io/invalid")
```

---

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

```
tbl_split <- model_id_split(hub_con_output)
tbl_split

# Merge model_id
tbl_merged <- model_id_merge(tbl_split)
tbl_merged

# Split / Merge using custom separator
tbl_sep <- hub_con_output
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", "target-data"),
  silent = TRUE
)
```

Arguments

hub_path	Either a character string path to a local Modeling Hub directory, a character string of a URL to a GitHub repository or an object of class <SubTreeFileSystem> created using functions arrow::s3_bucket() or arrow::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	Type of config file to read. One of "tasks", "admin" or "model-metadata-schema". Default is "tasks".
silent	Logical. If TRUE, suppress warnings. Default is FALSE.

Value

The contents of the config file as an R list. If possible, the output is further converted to a <config> class object before returning. Note that "model-metadata-schema" files are never converted to a <config> object.

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 a GitHub hub repository
github_url <- "https://github.com/hubverse-org/example-simple-forecast-hub"
read_config(github_url)
read_config(github_url, "admin")

# Read config file from AWS S3 bucket hub
hub_path <- arrow::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, silent = TRUE)
```

Arguments

config_path	Either a character string of a path to a local JSON config file, a character string of the URL to the **raw contents** of a JSON config file (e.g on GitHub) or an object of class <SubTreeFileSystem> created using functions arrow::s3_bucket() and associated methods for creating paths to JSON config files within the bucket.
silent	Logical. If TRUE, suppress warnings. Default is FALSE.

Value

The contents of the config file as an R list. If possible, the output is further converted to a <config> class object before returning. Note that "model-metadata-schema" files are never converted to a <config> object.

Examples

```
# Read local config file
read_config_file(system.file("config", "tasks.json", package = "hubUtils"))
# Read config file from URL
url <- paste0(
  "https://raw.githubusercontent.com/hubverse-org/",
```

```
  "example-simple-forecast-hub/refs/heads/main/hub-config/tasks.json"
)
read_config_file(url)

# Read config file from AWS S3 bucket hub
hub_path <- arrow::s3_bucket("hubverse/hubutils/testhubs/simple/")
config_path <- hub_path$path("hub-config/admin.json")
read_config_file(config_path)
```

`read_model_metadata` *Read a model metadata file into R*

Description

Reads a model metadata YAML file into a named list.

Usage

```
read_model_metadata(path)
```

Arguments

`path`	A character string of the path to a local model metadata YAML file.

Details

Every array in the file is returned as a list, including an array with a single element. For example, `methods: ["one"]` is returned as `list("one")` and not as the string `"one"`.

Value

The contents of the metadata file as a named list, or `NULL` for an empty file.

Examples

```
hub_path <- system.file("testhubs/simple", package = "hubUtils")
path <- fs::path(hub_path, "model-metadata", "team1-goodmodel.yaml")
read_model_metadata(path)
```

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.

subset_task_id_cols *Subset a* `model_out_tbl` *or submission* `tbl`.

Description

Subset a `model_out_tbl` or submission `tbl`.

Usage

```
subset_task_id_cols(model_out_tbl)

subset_std_cols(model_out_tbl)
```

Arguments

`model_out_tbl`	A `model_out_tbl` or submission `tbl` object. Must inherit from class `data.frame`.

Value

- `subset_task_id_cols`: an object of the same class as `model_out_tbl` which contains only task ID columns.
- `subset_std_cols`: an object of the same class as `model_out_tbl` which contains only hubverse standard columns (i.e. columns that are not task_id columns).

Functions

- `subset_task_id_cols()`: subset a `model_out_tbl` or submission `tbl` to only include task_id columns
- `subset_std_cols()`: subset a `model_out_tbl` or submission `tbl` to only include hubverse standard columns (i.e. columns that are not task_id columns)

Examples

```
model_out_tbl_path <- system.file("testhubs", "v4", "simple",
  "model-output", "hub-baseline", "2022-10-15-hub-baseline.parquet",
  package = "hubUtils"
)
model_out_tbl <- arrow::read_parquet(model_out_tbl_path)
subset_task_id_cols(model_out_tbl)
subset_std_cols(model_out_tbl)
```

---

subset_task_id_names *Subset a vector of column names to only include task IDs*

Description

Subset a vector of column names to only include task IDs

Usage

```
subset_task_id_names(x)
```

Arguments

x	character vector of column names

Value

a character vector of task ID names

Examples

```
x <- c(
  "origin_date", "horizon", "target_date",
  "location", "output_type", "output_type_id", "value"
)
subset_task_id_names(x)
```

---

target-data-utils *Get target data configuration properties*

Description

Utility functions for extracting properties from target-data.json configuration files (v6.0.0 schema). These functions handle defaults and inheritance patterns for target data configuration.

Usage

```
get_date_col(config_target_data)

get_observable_unit(
  config_target_data,
  dataset = c("time-series", "oracle-output")
)

get_versioned(config_target_data, dataset = c("time-series", "oracle-output"))

get_has_output_type_ids(config_target_data)

get_non_task_id_schema(config_target_data)

has_target_data_config(hub_path)

## Default S3 method:
has_target_data_config(hub_path)

## S3 method for class 'SubTreeFileSystem'
has_target_data_config(hub_path)
```

Arguments

config_target_data	A target-data config object created by `read_config(hub_path, "target-data")`.
dataset	Character string specifying the dataset type: either `"time-series"` or `"oracle-output"`. Used for functions that extract dataset-specific properties.
hub_path	Path to a hub. Can be a local directory path or cloud URL (S3, GCS).

Details

Inheritance and Defaults:

Some properties can be specified at both the global level and the dataset level:

- **observable_unit**: Dataset-specific values override global when specified, otherwise the global value is used.
- **versioned**: Dataset-specific values override global when specified, otherwise inherits from global (default `FALSE` if not specified anywhere).

Other properties are dataset-specific only:

- **has_output_type_ids**: Only for oracle-output dataset (default `FALSE`)
- **non_task_id_schema**: Only for time-series dataset (default `NULL`)

Value

`get_date_col()` returns a character string: the name of the date column that stores the date on which observed data actually occurred.

`get_observable_unit()` returns a character vector: column names whose unique value combinations define the minimum observable unit.

`get_versioned()` returns a logical value: whether the dataset is versioned using `as_of` dates.

`get_has_output_type_ids()` returns a logical value: whether oracle-output data has `output_type` and `output_type_id` columns (default `FALSE` if not specified).

`get_non_task_id_schema()` returns a named list: key-value pairs of non-task ID column names and their data types, or `NULL` if not specified.

`has_target_data_config()` returns a logical value: `TRUE` if the target-data.json file exists in the `hub-config` directory of the hub, `FALSE` otherwise.

Functions

- `get_date_col()`: Get the name of the date column across hub data.
- `get_observable_unit()`: Get observable unit column names. Returns dataset-specific observable_unit if configured, otherwise falls back to global.
- `get_versioned()`: Get whether target data is versioned for the specified dataset. Returns dataset-specific setting if configured, otherwise inherits from global (default `FALSE` if not specified).
- `get_has_output_type_ids()`: Get whether oracle-output data has output_type/output_type_id columns.
- `get_non_task_id_schema()`: Get the schema for non-task ID columns in time-series data.
- `has_target_data_config()`: Check if target data config file exists in hub.

Examples

```
hub_path <- system.file("testhubs/v6/target_dir", package = "hubUtils")
config <- read_config(hub_path, "target-data")

# Get the date column name
get_date_col(config)

# Get observable unit (uses dataset-specific or falls back to global)
get_observable_unit(config, dataset = "time-series")
get_observable_unit(config, dataset = "oracle-output")

# Get versioned setting (inherits from global if not specified)
get_versioned(config, dataset = "time-series")

# Get oracle-output specific property
get_has_output_type_ids(config)

# Get time-series specific property
get_non_task_id_schema(config)

# Check if target data config exists
has_target_data_config(hub_path)
no_config_hub <- system.file("testhubs/v5/target_file/", package = "hubUtils")
has_target_data_config(no_config_hub)
```

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

```
md_out <- as_model_out_tbl(hub_con_output)
validate_model_out_tbl(md_out)
```

version_equal *Compare hub config* schema_version*s to specific version numbers from a variety of sources*

Description

Compare hub config schema_versions to specific version numbers from a variety of sources

Usage

```
version_equal(
  version,
  config = NULL,
  config_path = NULL,
  hub_path = NULL,
  schema_version = NULL
)

version_gte(
  version,
  config = NULL,
```

```
  config_path = NULL,
  hub_path = NULL,
  schema_version = NULL
)

version_gt(
  version,
  config = NULL,
  config_path = NULL,
  hub_path = NULL,
  schema_version = NULL
)

version_lte(
  version,
  config = NULL,
  config_path = NULL,
  hub_path = NULL,
  schema_version = NULL
)

version_lt(
  version,
  config = NULL,
  config_path = NULL,
  hub_path = NULL,
  schema_version = NULL
)
```

Arguments

version	Character string. Version number to compare against, must be in the format "v#.#.#".
config	A <config> class object. Usually the output of read_config or read_config_file.
config_path	Either a character string of a path to a local JSON config file, a character string of the URL to the **raw contents** of a JSON config file (e.g on GitHub) or an object of class <SubTreeFileSystem> created using functions arrow::s3_bucket() and associated methods for creating paths to JSON config files within the bucket.
hub_path	Either a character string path to a local Modeling Hub directory, a character string of a URL to a GitHub repository or an object of class <SubTreeFileSystem> created using functions arrow::s3_bucket() or arrow::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.
schema_version	Character string. A config schema_version property to compare against.

Value

TRUE or FALSE depending on how the schema version compares to the version number specified.

Functions

- version_equal(): Check whether a schema version property is equal to a specific version number.
- version_gte(): Check whether a schema version property is equal to or greater than a specific version number.
- version_gt(): Check whether a schema version property is greater than a specific version number.
- version_lte(): Check whether a schema version property is equal to or less than a specific version number.
- version_lt(): Check whether a schema version property is less than a specific version number.

Examples

```
# Actual version "v2.0.0"
hub_path <- system.file("testhubs/simple", package = "hubUtils")
# Actual version "v3.0.0"
config_path <- system.file("config", "tasks.json", package = "hubUtils")
config <- read_config_file(config_path)
schema_version <- config$schema_version
# Check whether schema_version equal to v3.0.0
version_equal("v3.0.0", config = config)
version_equal("v3.0.0", config_path = config_path)
version_equal("v3.0.0", hub_path = hub_path)
version_equal("v3.0.0", schema_version = schema_version)
# Check whether schema_version equal to or greater than v3.0.0
version_gte("v3.0.0", config = config)
version_gte("v3.0.0", config_path = config_path)
version_gte("v3.0.0", hub_path = hub_path)
version_gte("v3.0.0", schema_version = schema_version)
# Check whether schema_version greater than v3.0.0
version_gt("v3.0.0", config = config)
version_gt("v3.0.0", config_path = config_path)
version_gt("v3.0.0", hub_path = hub_path)
version_gt("v3.0.0", schema_version = schema_version)
# Check whether schema_version equal to or less than v3.0.0
version_lte("v3.0.0", config = config)
version_lte("v3.0.0", config_path = config_path)
version_lte("v3.0.0", hub_path = hub_path)
version_lte("v3.0.0", schema_version = schema_version)
# Check whether schema_version less than v3.0.0
version_lt("v3.0.0", config = config)
version_lt("v3.0.0", config_path = config_path)
version_lt("v3.0.0", hub_path = hub_path)
version_lt("v3.0.0", schema_version = schema_version)
```

Index

∗ **datasets**
hub_con_output, 25
std_colnames, 35
∗ **functions supporting config file validation**
get_schema, 18
get_schema_url, 19
get_schema_valid_versions, 20

arrow::gs_bucket(), *10*, *24*, *30*, *32*, *40*
arrow::s3_bucket(), *10*, *24*, *29*, *30*, *32*, *33*, *40*
as_config, 3
as_model_out_tbl, 3

check_deprecated_schema, 5
convert_output_type, 6
create_s3_url, 7

extract_schema_version, 8

get_config_derived_task_ids (get_derived_task_ids_config), 9
get_config_tid, 8
get_date_col (target-data-utils), 36
get_derived_task_ids_config, 9
get_derived_task_ids_config(), *11*
get_has_output_type_ids (target-data-utils), 36
get_hub_derived_task_ids (get_hub_timezone), 10
get_hub_derived_task_ids(), *10*
get_hub_file_formats (get_hub_timezone), 10
get_hub_model_output_dir (get_hub_timezone), 10
get_hub_timezone, 10
get_non_task_id_schema (target-data-utils), 36
get_observable_unit (target-data-utils), 36
get_round_config, 12
get_round_ids (get_round_idx), 12
get_round_idx, 12
get_round_model_tasks, 14
get_round_output_type_names, 15
get_round_task_id_names, 15
get_round_task_id_names(), *21*
get_round_value_sets_config, 16
get_schema, 18, *19*, *20*
get_schema_url, *18*, 19, *20*
get_schema_valid_versions, *18*, *19*, 20
get_schema_version_latest, 20
get_task_id_names, 21
get_task_id_type, 22
get_task_id_values, 23
get_task_id_values(), *22*
get_version_config, 24
get_version_file (get_version_config), 24
get_version_hub (get_version_config), 24
get_versioned (target-data-utils), 36

has_target_data_config (target-data-utils), 36
hub_con_output, 25

is_github_repo_url, 26
is_github_url, 27
is_s3_base_fs, 27
is_url, 28
is_v3_config, 29
is_v3_config_file, 29
is_v3_hub, 30
is_valid_url, 30

model_id_merge, 31
model_id_split (model_id_merge), 31

read_config, 32
read_config(), *9*, *12–17*, *21–23*

read_config_file, 33
read_model_metadata, 34

std_colnames, 35
subset_std_cols (subset_task_id_cols), 35
subset_task_id_cols, 35
subset_task_id_names, 36

target-data-utils, 36
tibble, *31*

validate_model_out_tbl, 39
version_equal, 39
version_gt (version_equal), 39
version_gte (version_equal), 39
version_lt (version_equal), 39
version_lte (version_equal), 39