Package: hubValidations (via r-universe)

September 11, 2024

Title Testing framework for hubverse hub validations

Version 0.6.2

Description This package aims at providing a simple interface to run validations on data and metadata submitted to a hubverse modeling hub. Validation tests can be run at different levels (single file, single folder, whole repository) and locally as well as part of a continuous integration workflow.

License MIT + file LICENSE

Imports arrow (>= 17.0.0), checkmate, cli, config, dplyr (>= 1.1.0), fs, gh, hubAdmin (>= 1.0.0), hubData (>= 1.1.0), hubUtils (>= 0.1.2), jsonlite, jsonvalidate, lifecycle, lubridate, magrittr, purrr, rlang, stringr, tibble, yaml

Suggests covr, gert, kableExtra, mockery, readr, rmarkdown, testthat (>= 3.2.0), testthis, withr

Remotes hubverse-org/hubUtils, hubverse-org/hubData, hubverse-org/hubAdmin

Config/testthat/edition 3

Config/Needs/website pkgdown, hubverse-org/hubStyle

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

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

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

Depends R (>= 3.5.0)

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

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

RemoteRef HEAD

RemoteSha 640f68083dc1e5246856bf9d103d838ed17154fc

Contents

capture_check_cnd . . . 3
capture_check_info . . . 4
capture_exec_error . . . 5
capture_exec_warning . . . 6
check_config_hub_valid . . . 6
check_file_exists . . . 7
check_file_format . . . 8
check_file_location . . . 8
check_file_name . . . 9
check_file_read . . . 10
check_for_errors . . . 10
check_metadata_file_exists . . . 11
check_metadata_file_ext . . . 12
check_metadata_file_location . . . 12
check_metadata_file_name . . . 13
check_metadata_matches_schema . . . 14
check_metadata_schema_exists . . . 14
check_submission_metadata_file_exists . . . 15
check_submission_time . . . 16
check_tbl_colnames . . . 17
check_tbl_col_types . . . 17
check_tbl_match_round_id . . . 19
check_tbl_rows_unique . . . 20
check_tbl_spl_compound_taskid_set . . . 20
check_tbl_spl_compound_tid . . . 22
check_tbl_spl_n . . . 23
check_tbl_spl_non_compound_tid . . . 25
check_tbl_unique_round_id . . . 26
check_tbl_values . . . 27
check_tbl_values_required . . . 28
check_tbl_value_col . . . 29
check_tbl_value_col_ascending . . . 30
check_tbl_value_col_sum1 . . . 31
check_valid_round_id . . . 31
check_valid_round_id_col . . . 32
combine . . . 33
expand_model_out_grid . . . 33
get_tbl_compound_taskid_set . . . 37
is_success . . . 38
match_tbl_to_model_task . . . 39
new_hub_validations . . . 41
opt_check_metadata_team_max_model_n . . . 42
opt_check_tbl_col_timediff . . . 43
opt_check_tbl_counts_lt_popn . . . 44
opt_check_tbl_horizon_timediff . . . 45
parse_file_name . . . 47
print.hub_validations . . . 48

print.pr_hub_validations 48
read_model_out_file 49
submission_tmpl 50
try_check 53
validate_model_data 53
validate_model_file 55
validate_model_metadata 56
validate_pr 57
validate_submission 59
validate_submission_time 61

Index **63**

`capture_check_cnd` *Capture a condition of the result of validation check.*

Description

Capture a condition of the result of validation check.

Usage

```
capture_check_cnd(
  check,
  file_path,
  msg_subject,
  msg_attribute,
  msg_verbs = c("is", "must be"),
  error = FALSE,
  details = NULL,
  ...
)
```

Arguments

`check`	logical, the result of a validation check. If `check` is `FALSE`, validation has failed. If `check` is `TRUE`, validation has succeeded.
`file_path`	character string. Path to the file being validated. Must be the relative path to the hub's `model-output` (or equivalent) directory.
`msg_subject`	character string. The subject of the validation.
`msg_attribute`	character string. The attribute of subject being validated.
`msg_verbs`	character vector of length 2. The verbs describing the state of the attribute in relation to the validation subject. The first element describes the state when validation succeeds, the second element, when validation fails.

`error`	logical. In the case of validation failure, whether the function should return an object of class `<error/check_error>` (`TRUE`) or `<error/check_failure>` (`FALSE`, default).
`details`	further details to be appended to the output message.
`...`	<dynamic> Named data fields stored inside the condition object.

Details

Arguments `msg_subject`, `msg_attribute`, `msg_verbs` and `details` accept text that can interpreted and formatted by `cli::format_inline()`.

Value

Depending on whether validation has succeeded and the value of the `error` argument, one of:

- `<message/check_success>` condition class object.
- `<error/check_failure>` condition class object.
- `<error/check_error>` condition class object.

Returned object also inherits from subclass `<hub_check>`.

Examples

```
capture_check_cnd(
  check = TRUE, file_path = "test/file.csv",
  msg_subject = "{.var round_id}", msg_attribute = "valid.", error = FALSE
)
capture_check_cnd(
  check = FALSE, file_path = "test/file.csv",
  msg_subject = "{.var round_id}", msg_attribute = "valid.", error = FALSE,
  details = "Must be one of 'A' or 'B', not 'C'"
)
capture_check_cnd(
  check = FALSE, file_path = "test/file.csv",
  msg_subject = "{.var round_id}", msg_attribute = "valid.", error = TRUE,
  details = "Must be one of {.val {c('A', 'B')}}, not {.val C}"
)
```

---

capture_check_info *Capture a simple info message condition*

Description

Capture a simple info message condition. Useful for communicating when a check is ignored or skipped.

Usage

```
capture_check_info(file_path, msg, call = rlang::caller_call())
```

Arguments

`file_path`	character string. Path to the file being validated. Must be the relative path to the hub's `model-output` (or equivalent) directory.
`msg`	Character string. Accepts text that can interpreted and formatted by `cli::format_inline()`.
`call`	The defused call of the function that generated the message. Use to override default which uses the caller call. See rlang::stack for more details.

Value

A `<message/check_info>` condition class object. Returned object also inherits from subclass `<hub_check>`.

`capture_exec_error` *Capture an execution error condition*

Description

Capture an execution error condition. Useful for communicating when a check execution has failed. Usually used in conjunction with `try`.

Usage

```
capture_exec_error(file_path, msg, call = NULL)
```

Arguments

`file_path`	character string. Path to the file being validated. Must be the relative path to the hub's `model-output` (or equivalent) directory.
`msg`	Character string.
`call`	Character string. Name of the parent call that failed to execute. If `NULL` (default), the caller's call name is captured.

Value

A `<error/check_exec_error>` condition class object. Returned object also inherits from subclass `<hub_check>`.

`capture_exec_warning` *Capture an execution warning condition*

Description

Capture an execution warning condition. Useful for communicating when a check execution has failed. Usually used in conjunction with `try`.

Usage

```
capture_exec_warning(file_path, msg, call = NULL)
```

Arguments

`file_path`	character string. Path to the file being validated. Must be the relative path to the hub's `model-output` (or equivalent) directory.
`msg`	Character string.
`call`	Character string. Name of the parent call that failed to execute. If `NULL` (default), the caller's call name is captured.

Value

A `<warning/check_exec_warn>` condition class object. Returned object also inherits from subclass `<hub_check>`.

`check_config_hub_valid`
Check hub correctly configured

Description

Checks that `admin` and `tasks` configuration files in directory `hub-config` are valid.

Usage

```
check_config_hub_valid(hub_path)
```

Arguments

`hub_path`	Either a character string path to a local Modeling Hub directory or an object of class `<SubTreeFileSystem>` created using functions `s3_bucket()` or `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. The hub must be fully configured with valid `admin.json` and `tasks.json` files within the `hub-config` directory.

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_error>` condition class object.

Returned object also inherits from subclass `<hub_check>`.

`check_file_exists` *Check file exists at the file path specified*

Description

Check file exists at the file path specified

Usage

```
check_file_exists(
  file_path,
  hub_path = ".",
  subdir = c("model-output", "model-metadata", "hub-config")
)
```

Arguments

`file_path`	character string. Path to the file being validated relative to the hub's model-output directory.
`hub_path`	Either a character string path to a local Modeling Hub directory or an object of class `<SubTreeFileSystem>` created using functions `s3_bucket()` or `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. The hub must be fully configured with valid `admin.json` and `tasks.json` files within the `hub-config` directory.
`subdir`	subdirectory within the hub

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_error>` condition class object.

Returned object also inherits from subclass `<hub_check>`.

`check_file_format` *Check file format is accepted by hub.*

Description

Check file format is accepted by hub.

Usage

```
check_file_format(file_path, hub_path, round_id)
```

Arguments

`file_path`	character string. Path to the file being validated relative to the hub's model-output directory.
`hub_path`	Either a character string path to a local Modeling Hub directory or an object of class `<SubTreeFileSystem>` created using functions `s3_bucket()` or `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. The hub must be fully configured with valid `admin.json` and `tasks.json` files within the `hub-config` directory.
`round_id`	character string. The round identifier.

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_error>` condition class object.

Returned object also inherits from subclass `<hub_check>`.

`check_file_location` *Check file is being submitted to the correct folder*

Description

Checks that the `model_id` metadata in the file name matches the directory name the file is being submitted to.

Usage

```
check_file_location(file_path)
```

Arguments

`file_path`	character string. Path to the file being validated relative to the hub's model-output directory.

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_failure>` condition class object.

Returned object also inherits from subclass `<hub_check>`.

`check_file_name` *Check a model output file name can be correctly parsed.*

Description

Check a model output file name can be correctly parsed.

Usage

```
check_file_name(file_path)
```

Arguments

`file_path`	character string. Path to the file being validated relative to the hub's model-output directory.

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_error>` condition class object.

Returned object also inherits from subclass `<hub_check>`.

check_file_read *Check file can be read successfully*

Description

Check file can be read successfully

Usage

```
check_file_read(file_path, hub_path = ".")
```

Arguments

file_path	character string. Path to the file being validated relative to the hub's model-output directory.
hub_path	Either a character string path to a local Modeling Hub directory or an object of class <SubTreeFileSystem> created using functions s3_bucket() or 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. The hub must be fully configured with valid admin.json and tasks.json files within the hub-config directory.

Value

Depending on whether validation has succeeded, one of:

- <message/check_success> condition class object.
- <error/check_error> condition class object.

Returned object also inherits from subclass <hub_check>.

check_for_errors *Raise conditions stored in a* hub_validations *S3 object*

Description

This is meant to be used in CI workflows to raise conditions from hub_validations objects but can also be useful locally to summarise the results of checks contained in a hub_validations S3 object.

Usage

```
check_for_errors(x, verbose = FALSE)
```

Arguments

`x`	A `hub_validations` object
`verbose`	Logical. If `TRUE`, print the results of all checks prior to raising condition and summarising `hub_validations` S3 object check results.

Value

An error if one of the elements of `x` is of class `check_failure`, `check_error`, `check_exec_error` or `check_exec_warning`. `TRUE` invisibly otherwise.

`check_metadata_file_exists` *Check whether a metadata schema file exists*

Description

Check whether a metadata schema file exists

Usage

```
check_metadata_file_exists(hub_path = ".", file_path)
```

Arguments

`hub_path`	Either a character string path to a local Modeling Hub directory or an object of class `<SubTreeFileSystem>` created using functions `s3_bucket()` or `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. The hub must be fully configured with valid `admin.json` and `tasks.json` files within the `hub-config` directory.
`file_path`	character string. Path to the file being validated relative to the hub's model-metadata directory.

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_error>` condition class object.

Returned object also inherits from subclass `<hub_check>`.

`check_metadata_file_ext`
Check file is being submitted to the correct folder

Description

Checks that the `model_id` metadata in the file name matches the directory name the file is being submitted to.

Usage

```
check_metadata_file_ext(file_path)
```

Arguments

`file_path`	character string. Path to the file being validated relative to the hub's model-output directory.

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_failure>` condition class object.

Returned object also inherits from subclass `<hub_check>`.

`check_metadata_file_location`
Check that the metadata file is being submitted to the correct folder

---

Description

Check that the metadata file is being submitted to the correct folder

Usage

```
check_metadata_file_location(file_path)
```

Arguments

`file_path`	character string. Path to the file being validated relative to the hub's model-metadata directory.

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_failure>` condition class object.

Returned object also inherits from subclass `<hub_check>`.

`check_metadata_file_name`

Check whether the file name of a metadata file matches the model_id or combination of team_abbr and model_abbr specified within the metadata file

---

Description

Check whether the file name of a metadata file matches the model_id or combination of team_abbr and model_abbr specified within the metadata file

Usage

```
check_metadata_file_name(file_path, hub_path = ".")
```

Arguments

`file_path`	character string. Path to the file being validated relative to the hub's model-metadata directory.
`hub_path`	Either a character string path to a local Modeling Hub directory or an object of class `<SubTreeFileSystem>` created using functions `s3_bucket()` or `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. The hub must be fully configured with valid `admin.json` and `tasks.json` files within the `hub-config` directory.

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_failure>` condition class object.

Returned object also inherits from subclass `<hub_check>`.

`check_metadata_matches_schema`
Check whether a metadata file matches the schema provided by the hub

---

Description

Check whether a metadata file matches the schema provided by the hub

Usage

```
check_metadata_matches_schema(file_path, hub_path = ".")
```

Arguments

`file_path`	character string. Path to the file being validated relative to the hub's model-metadata directory.
`hub_path`	Either a character string path to a local Modeling Hub directory or an object of class `<SubTreeFileSystem>` created using functions `s3_bucket()` or `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. The hub must be fully configured with valid `admin.json` and `tasks.json` files within the `hub-config` directory.

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_failure>` condition class object.

Returned object also inherits from subclass `<hub_check>`.

`check_metadata_schema_exists`
Check whether a metadata schema file exists

Description

Check whether a metadata schema file exists

Usage

```
check_metadata_schema_exists(hub_path = ".")
```

Arguments

`hub_path`	Either a character string path to a local Modeling Hub directory or an object of class `<SubTreeFileSystem>` created using functions `s3_bucket()` or `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. The hub must be fully configured with valid `admin.json` and `tasks.json` files within the `hub-config` directory.

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_failure>` condition class object.

Returned object also inherits from subclass `<hub_check>`.

`check_submission_metadata_file_exists`

Check whether a metadata file for the given model exists

Description

Check whether a metadata file for the given model exists

Usage

```
check_submission_metadata_file_exists(file_path, hub_path = ".")
```

Arguments

`file_path`	character string. Path to the file being validated relative to the hub's model-output directory.
`hub_path`	Either a character string path to a local Modeling Hub directory or an object of class `<SubTreeFileSystem>` created using functions `s3_bucket()` or `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. The hub must be fully configured with valid `admin.json` and `tasks.json` files within the `hub-config` directory.

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_error>` condition class object.

Returned object also inherits from subclass `<hub_check>`.

`check_submission_time` *Checks submission is within the valid submission window for a given round.*

Description

Checks submission is within the valid submission window for a given round.

Usage

```
check_submission_time(
  hub_path,
  file_path,
  ref_date_from = c("file", "file_path")
)
```

Arguments

`hub_path`	Either a character string path to a local Modeling Hub directory or an object of class `<SubTreeFileSystem>` created using functions `s3_bucket()` or `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. The hub must be fully configured with valid `admin.json` and `tasks.json` files within the `hub-config` directory.
`file_path`	character string. Path to the file being validated relative to the hub's model-output directory.
`ref_date_from`	whether to get the reference date around which relative submission windows will be determined from the file's `file_path` round ID or the `file` contents themselves. `file` requires that the file can be read. Only applicable when a round is configured to determine the submission windows relative to the value in a date column in model output files. Not applicable when explicit submission window start and end dates are provided in the hub's config.

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_failure>` condition class object.

Returned object also inherits from subclass `<hub_check>`.

check_tbl_colnames *Check column names of model output data*

Description

Checks that a tibble/data.frame of data read in from the file being validated contains the expected task ID and standard column names according the round configuration being validated against.

Usage

```
check_tbl_colnames(tbl, round_id, file_path, hub_path = ".")
```

Arguments

tbl	a tibble/data.frame of the contents of the file being validated.
round_id	character string. The round identifier.
file_path	character string. Path to the file being validated relative to the hub's model-output directory.
hub_path	Either a character string path to a local Modeling Hub directory or an object of class <SubTreeFileSystem> created using functions s3_bucket() or 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. The hub must be fully configured with valid admin.json and tasks.json files within the hub-config directory.

Value

Depending on whether validation has succeeded, one of:

- <message/check_success> condition class object.
- <error/check_error> condition class object.

Returned object also inherits from subclass <hub_check>.

check_tbl_col_types *Check model data column data types*

Description

Check that model output data column datatypes conform to those define in the hub config.

Usage

```
check_tbl_col_types(
  tbl,
  file_path,
  hub_path,
  output_type_id_datatype = c("from_config", "auto", "character", "double", "integer",
    "logical", "Date")
)
```

Arguments

`tbl`	a tibble/data.frame of the contents of the file being validated.
`file_path`	character string. Path to the file being validated relative to the hub's model-output directory.
`hub_path`	Either a character string path to a local Modeling Hub directory or an object of class `<SubTreeFileSystem>` created using functions `s3_bucket()` or `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. The hub must be fully configured with valid `admin.json` and `tasks.json` files within the `hub-config` directory.
`output_type_id_datatype`	character string. One of `"from_config"`, `"auto"`, `"character"`, `"double"`, `"integer"`, `"logical"`, `"Date"`. Defaults to `"from_config"` which uses the setting in the `output_type_id_datatype` property in the `tasks.json` config file if available. If the property is not set in the config, the argument falls back to `"auto"` which determines the `output_type_id` data type automatically from the `tasks.json` config file as the simplest data type required to represent all output type ID values across all output types in the hub. Other data type values can be used to override automatic determination. Note that attempting to coerce `output_type_id` to a data type that is not valid for the data (e.g. trying to coerce`"character"` values to `"double"`) will likely result in an error or potentially unexpected behaviour so use with care.

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_failure>` condition class object.

Returned object also inherits from subclass `<hub_check>`.

`check_tbl_match_round_id` *Check model output data tbl round ID matches submission round ID.*

Description

Check model output data tbl round ID matches submission round ID.

Usage

```
check_tbl_match_round_id(tbl, file_path, hub_path, round_id_col = NULL)
```

Arguments

`tbl`	a tibble/data.frame of the contents of the file being validated.
`file_path`	character string. Path to the file being validated relative to the hub's model-output directory.
`hub_path`	Either a character string path to a local Modeling Hub directory or an object of class `<SubTreeFileSystem>` created using functions `s3_bucket()` or `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. The hub must be fully configured with valid `admin.json` and `tasks.json` files within the `hub-config` directory.
`round_id_col`	Character string. The name of the column containing `round_id`s. Usually, the value of round property `round_id` in hub `tasks.json` config file.

Details

This check only applies to files being submitted to rounds where `round_id_from_variable: true` or where a `round_id_col` name is explicitly provided. Skipped otherwise.

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_error>` condition class object.

If `round_id_from_variable: false` and no `round_id_col` name is provided, check is skipped and a `<message/check_info>` condition class object is returned. If no valid `round_id_col` name is provided or can extracted from config (check through `check_valid_round_id_col`), a `<message/check_error>` condition class object is returned and the rest of the check skipped.

`check_tbl_rows_unique` *Check model data rows are all unique*

Description

Checks that combinations of task ID, output type and output type ID value combinations are unique, by checking that there are no duplicate rows across all `tbl` columns excluding the `value` column.

Usage

```
check_tbl_rows_unique(tbl, file_path, hub_path)
```

Arguments

`tbl`	a tibble/data.frame of the contents of the file being validated. Column types must **all be character**.
`file_path`	character string. Path to the file being validated relative to the hub's model-output directory.
`hub_path`	Either a character string path to a local Modeling Hub directory or an object of class `<SubTreeFileSystem>` created using functions `s3_bucket()` or `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. The hub must be fully configured with valid `admin.json` and `tasks.json` files within the `hub-config` directory.

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_failure>` condition class object.

Returned object also inherits from subclass `<hub_check>`.

`check_tbl_spl_compound_taskid_set` *Check model output data tbl sample compound task id sets for each modeling task match or are coarser than the expected set defined in the config.*

Description

This check detects the compound task ID sets of samples, implied by the `output_type_id` and task ID values, and checks them for internal consistency and compliacance with the `compound_taskid_set` defined for each round modeling task in the `tasks.json` config.

Usage

```
check_tbl_spl_compound_taskid_set(
  tbl,
  round_id,
  file_path,
  hub_path,
  derived_task_ids = NULL
)
```

Arguments

`tbl`	a tibble/data.frame of the contents of the file being validated. Column types must **all be character**.
`round_id`	character string. The round identifier.
`file_path`	character string. Path to the file being validated relative to the hub's model-output directory.
`hub_path`	Either a character string path to a local Modeling Hub directory or an object of class `<SubTreeFileSystem>` created using functions `s3_bucket()` or `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. The hub must be fully configured with valid `admin.json` and `tasks.json` files within the `hub-config` directory.
`derived_task_ids`	Character vector of derived task ID names (task IDs whose values depend on other task IDs) to ignore. Columns for such task ids will contain `NA`s.

Details

If the check fails, the output of the check includes an `errors` element, a list of items, one for each modeling task failing validation. The structure depends on the reason the check failed.

If the check failed because more that a single unique `compound_taskid_set` was found for a given model task, the `errors` object will be a list with one element for each `compound_taskid_set` detected and will have the following structure:

- `tbl_comp_tids`: a compound task id set detected in the the tbl.
- `output_type_ids`: The output type ID of the sample that does not contain a single, unique value for each compound task ID.

If the check failed because task IDs which is not allowed in the config, were identified as compound task ID (i.e. samples describe "finer" compound modeling tasks) for a given

model task, the `errors` object will be a list with the structure described above as well as the additional following elements:

- `config_comp_tids`: the allowed `compound_taskid_set` defined in the modeling task config.
- `invalid_tbl_comp_tids`: the names of invalid compound task IDs.

The name of each element is the index identifying the config modeling task the sample is associated with `mt_id`. See hubverse documentation on samples for more details.

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_error>` condition class object.

Returned object also inherits from subclass `<hub_check>`.

---

check_tbl_spl_compound_tid

Check model output data tbl samples contain single unique values for each compound task ID within individual samples

---

Description

Check model output data tbl samples contain single unique values for each compound task ID within individual samples

Usage

```
check_tbl_spl_compound_tid(
  tbl,
  round_id,
  file_path,
  hub_path,
  compound_taskid_set = NULL,
  derived_task_ids = NULL
)
```

Arguments

`tbl`	a tibble/data.frame of the contents of the file being validated. Column types must **all be character**.
`round_id`	character string. The round identifier.
`file_path`	character string. Path to the file being validated relative to the hub's model-output directory.

`hub_path` Either a character string path to a local Modeling Hub directory or an object of class `<SubTreeFileSystem>` created using functions `s3_bucket()` or `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. The hub must be fully configured with valid `admin.json` and `tasks.json` files within the `hub-config` directory.

`compound_taskid_set`
a list of `compound_taskid_sets` (characters vector of compound task IDs), one for each modeling task. Used to override the compound task ID set in the config file, for example, when validating coarser samples.

`derived_task_ids`
Character vector of derived task ID names (task IDs whose values depend on other task IDs) to ignore. Columns for such task ids will contain `NA`s.

Details

Output of the check includes an `errors` element, a list of items, one for each sample failing validation, with the following structure:

- `mt_id`: Index identifying the config modeling task the sample is associated with.
- `output_type_id`: The output type ID of the sample that does not contain a single, unique value for each compound task ID.
- `values`: The unique values of each compound task ID. See hubverse documentation on samples for more details.

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_error>` condition class object.

Returned object also inherits from subclass `<hub_check>`.

`check_tbl_spl_n` *Check model output data tbl samples contain the appropriate number of samples for a given compound idx.*

Description

Check model output data tbl samples contain the appropriate number of samples for a given compound idx.

Usage

```
check_tbl_spl_n(
  tbl,
  round_id,
  file_path,
  hub_path,
  compound_taskid_set = NULL,
  derived_task_ids = NULL
)
```

Arguments

`tbl`	a tibble/data.frame of the contents of the file being validated. Column types must **all be character**.
`round_id`	character string. The round identifier.
`file_path`	character string. Path to the file being validated relative to the hub's model-output directory.
`hub_path`	Either a character string path to a local Modeling Hub directory or an object of class `<SubTreeFileSystem>` created using functions `s3_bucket()` or `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. The hub must be fully configured with valid `admin.json` and `tasks.json` files within the `hub-config` directory.
`compound_taskid_set`	a list of `compound_taskid_sets` (characters vector of compound task IDs), one for each modeling task. Used to override the compound task ID set in the config file, for example, when validating coarser samples.
`derived_task_ids`	Character vector of derived task ID names (task IDs whose values depend on other task IDs) to ignore. Columns for such task ids will contain `NA`s.

Details

Output of the check includes an `errors` element, a list of items, one for each compound_idx failing validation, with the following structure:

- `compound_idx`: the compound idx that failed validation of number of samples.
- `n`: the number of samples counted for the compound idx.
- `min_samples_per_task`: the minimum number of samples required for the compound idx.
- `max_samples_per_task`: the maximum number of samples required for the compound idx.
- `compound_idx_tbl`: a tibble of the expected structure for samples belonging to the compound idx. See hubverse documentation on samples for more details.

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_failure>` condition class object.

Returned object also inherits from subclass `<hub_check>`.

`check_tbl_spl_non_compound_tid` *Check model output data tbl samples contain single unique combination of non-compound task ID values across all samples*

Description

Check model output data tbl samples contain single unique combination of non-compound task ID values across all samples

Usage

```
check_tbl_spl_non_compound_tid(
  tbl,
  round_id,
  file_path,
  hub_path,
  compound_taskid_set = NULL,
  derived_task_ids = NULL
)
```

Arguments

`tbl`	a tibble/data.frame of the contents of the file being validated. Column types must **all be character**.
`round_id`	character string. The round identifier.
`file_path`	character string. Path to the file being validated relative to the hub's model-output directory.
`hub_path`	Either a character string path to a local Modeling Hub directory or an object of class `<SubTreeFileSystem>` created using functions `s3_bucket()` or `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. The hub must be fully configured with valid `admin.json` and `tasks.json` files within the `hub-config` directory.
`compound_taskid_set`	a list of `compound_taskid_set`s (characters vector of compound task IDs), one for each modeling task. Used to override the compound task ID set in the config file, for example, when validating coarser samples.

`derived_task_ids`
: Character vector of derived task ID names (task IDs whose values depend on other task IDs) to ignore. Columns for such task ids will contain `NA`s.

Details

Output of the check includes an `errors` element, a list of items, one for each modeling task containing samples failing validation, with the following structure:

- `mt_id`: Index identifying the config modeling task the samples are associated with.
- `output_type_ids`: The output type IDs of samples that do not match the most frequent non-compound task ID value combination across all samples in the modeling task.
- `frequent`: The most frequent non-compound task ID value combination across all samples in the modeling task to which all samples were compared. See hubverse documentation on samples for more details.

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_error>` condition class object.

Returned object also inherits from subclass `<hub_check>`.

`check_tbl_unique_round_id`

Check model output data tbl contains a single unique round ID.

Description

Check model output data tbl contains a single unique round ID.

Usage

```
check_tbl_unique_round_id(tbl, file_path, hub_path, round_id_col = NULL)
```

Arguments

`tbl`	a tibble/data.frame of the contents of the file being validated.
`file_path`	character string. Path to the file being validated relative to the hub's model-output directory.
`hub_path`	Either a character string path to a local Modeling Hub directory or an object of class `<SubTreeFileSystem>` created using functions `s3_bucket()` or `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. The hub must be fully configured with valid `admin.json` and `tasks.json` files within the `hub-config` directory.

`round_id_col` Character string. The name of the column containing `round_id`s. Usually, the value of round property `round_id` in hub `tasks.json` config file.

Details

This check only applies to files being submitted to rounds where `round_id_from_variable: true` or where a `round_id_col` name is explicitly provided. Skipped otherwise.

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_error>` condition class object.

If `round_id_from_variable: false` and no `round_id_col` name is provided, check is skipped and a `<message/check_info>` condition class object is returned. If no valid `round_id_col` name is provided or can extracted from config (check through `check_valid_round_id_col`), a `<message/check_error>` condition class object is returned and the rest of the check skipped.

`check_tbl_values` *Check model output data tbl contains valid value combinations*

Description

Check model output data tbl contains valid value combinations

Usage

```
check_tbl_values(tbl, round_id, file_path, hub_path, derived_task_ids = NULL)
```

Arguments

`tbl` a tibble/data.frame of the contents of the file being validated. Column types must **all be character**.

`round_id` character string. The round identifier.

`file_path` character string. Path to the file being validated relative to the hub's model-output directory.

`hub_path` Either a character string path to a local Modeling Hub directory or an object of class `<SubTreeFileSystem>` created using functions `s3_bucket()` or `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. The hub must be fully configured with valid `admin.json` and `tasks.json` files within the `hub-config` directory.

`derived_task_ids` Character vector of derived task ID names (task IDs whose values depend on other task IDs) to ignore. Columns for such task ids will contain `NA`s.

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_error>` condition class object.

Returned object also inherits from subclass `<hub_check>`.

`check_tbl_values_required`

Check all required task ID/output type/output type ID value combinations present in model data.

Description

Check all required task ID/output type/output type ID value combinations present in model data.

Usage

```
check_tbl_values_required(
  tbl,
  round_id,
  file_path,
  hub_path,
  derived_task_ids = NULL
)
```

Arguments

`tbl`	a tibble/data.frame of the contents of the file being validated. Column types must **all be character**.
`round_id`	character string. The round identifier.
`file_path`	character string. Path to the file being validated relative to the hub's model-output directory.
`hub_path`	Either a character string path to a local Modeling Hub directory or an object of class `<SubTreeFileSystem>` created using functions `s3_bucket()` or `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. The hub must be fully configured with valid `admin.json` and `tasks.json` files within the `hub-config` directory.
`derived_task_ids`	Character vector of derived task ID names (task IDs whose values depend on other task IDs) to ignore. Columns for such task ids will contain `NA`s.

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_failure>` condition class object.

Returned object also inherits from subclass `<hub_check>`.

`check_tbl_value_col` *Check output type values of model output data against config*

Description

Checks that values in the `value` column of a tibble/data.frame of data read in from the file being validated conform to the configuration for each output type of the appropriate model task.

Usage

```
check_tbl_value_col(
  tbl,
  round_id,
  file_path,
  hub_path,
  derived_task_ids = NULL
)
```

Arguments

`tbl`	a tibble/data.frame of the contents of the file being validated.
`round_id`	character string. The round identifier.
`file_path`	character string. Path to the file being validated relative to the hub's model-output directory.
`hub_path`	Either a character string path to a local Modeling Hub directory or an object of class `<SubTreeFileSystem>` created using functions `s3_bucket()` or `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. The hub must be fully configured with valid `admin.json` and `tasks.json` files within the `hub-config` directory.
`derived_task_ids`	Character vector of derived task ID names (task IDs whose values depend on other task IDs) to ignore. Columns for such task ids will contain `NA`s.

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_failure>` condition class object.

Returned object also inherits from subclass `<hub_check>`.

`check_tbl_value_col_ascending`
Check that `quantile` *and* `cdf` *output type values of model output data are non-descending*

Description

Checks that values in the `value` column for `quantile` and `cdf` output type data for each unique task ID/output type combination are non-descending when arranged by increasing `output_type_id` order. Check only performed if `tbl` contains `quantile` or `cdf` output type data. If not, the check is skipped and a `<message/check_info>` condition class object is returned.

Usage

```
check_tbl_value_col_ascending(tbl, file_path)
```

Arguments

`tbl`	a tibble/data.frame of the contents of the file being validated.
`file_path`	character string. Path to the file being validated relative to the hub's model-output directory.

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_failure>` condition class object.

Returned object also inherits from subclass `<hub_check>`.

`check_tbl_value_col_sum1` *Check that* `pmf` *output type values of model output data sum to 1.*

Description

Checks that values in the `value` column of `pmf` output type data for each unique task ID combination sum to 1. Check only performed if `tbl` contains `pmf` output type data. If not, the check is skipped and a `<message/check_info>` condition class object is returned.

Usage

```
check_tbl_value_col_sum1(tbl, file_path)
```

Arguments

`tbl`	a tibble/data.frame of the contents of the file being validated.
`file_path`	character string. Path to the file being validated relative to the hub's model-output directory.

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_failure>` condition class object.

Returned object also inherits from subclass `<hub_check>`.

`check_valid_round_id` *Check whether the* `round_id` *determined for the submission is valid*

Description

Check whether the `round_id` determined for the submission is valid

Usage

```
check_valid_round_id(round_id, file_path, hub_path = ".")
```

Arguments

round_id	character string. The round identifier.
file_path	character string. Path to the file being validated relative to the hub's model-output directory.
hub_path	Either a character string path to a local Modeling Hub directory or an object of class <SubTreeFileSystem> created using functions s3_bucket() or 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. The hub must be fully configured with valid admin.json and tasks.json files within the hub-config directory.

Value

Depending on whether validation has succeeded, one of:

- <message/check_success> condition class object.
- <error/check_error> condition class object.

Returned object also inherits from subclass <hub_check>.

---

check_valid_round_id_col

Check that any round_id_col name provided or extracted from the hub config is valid.

Description

Check that any round_id_col name provided or extracted from the hub config is valid.

Usage

```
check_valid_round_id_col(tbl, file_path, hub_path, round_id_col = NULL)
```

Arguments

tbl	a tibble/data.frame of the contents of the file being validated.
file_path	character string. Path to the file being validated relative to the hub's model-output directory.
hub_path	Either a character string path to a local Modeling Hub directory or an object of class <SubTreeFileSystem> created using functions s3_bucket() or 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. The hub must be fully configured with valid admin.json and tasks.json files within the hub-config directory.
round_id_col	Character string. The name of the column containing round_ids. Usually, the value of round property round_id in hub tasks.json config file.

Details

This check only applies to files being submitted to rounds where `round_id_from_variable: true` or where a `round_id_col` name is explicitly provided. Skipped otherwise.

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_failure>` condition class object.

If `round_id_from_variable: false` and no `round_id_col` name is provided, check is skipped and a `<message/check_info>` condition class object is returned. Returned object also inherits from subclass `<hub_check>`.

`combine` *Concatenate* `hub_validations` *S3 class objects*

Description

Concatenate `hub_validations` S3 class objects

Usage

```
combine(...)
```

Arguments

`...`	`hub_validations` S3 class objects to be concatenated.

Value

a `hub_validations` S3 class object.

`expand_model_out_grid` *Create expanded grid of valid task ID and output type value combinations*

Description

Create expanded grid of valid task ID and output type value combinations

Usage

```
expand_model_out_grid(
  config_tasks,
  round_id,
  required_vals_only = FALSE,
  all_character = FALSE,
  output_type_id_datatype = c("from_config", "auto", "character", "double", "integer",
    "logical", "Date"),
  as_arrow_table = FALSE,
  bind_model_tasks = TRUE,
  include_sample_ids = FALSE,
  compound_taskid_set = NULL,
  output_types = NULL,
  derived_task_ids = NULL
)
```

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 `hubUtils::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.
`required_vals_only`	Logical. Whether to return only combinations of Task ID and related output type ID required values.
`all_character`	Logical. Whether to return all character column.
`output_type_id_datatype`	character string. One of `"from_config"`, `"auto"`, `"character"`, `"double"`, `"integer"`, `"logical"`, `"Date"`. Defaults to `"from_config"` which uses the setting in the `output_type_id_datatype` property in the `tasks.json` config file if available. If the property is not set in the config, the argument falls back to `"auto"` which determines the `output_type_id` data type automatically from the `tasks.json` config file as the simplest data type required to represent all output type ID values across all output types in the hub. Other data type values can be used to override automatic determination. Note that attempting to coerce `output_type_id` to a data type that is not valid for the data (e.g. trying to coerce`"character"` values to `"double"`) will likely result in an error or potentially unexpected behaviour so use with care.
`as_arrow_table`	Logical. Whether to return an arrow table. Defaults to `FALSE`.
`bind_model_tasks`	Logical. Whether to bind expanded grids of values from multiple modeling tasks into a single tibble/arrow table or return a list.

include_sample_ids
: Logical. Whether to include sample identifiers in the output_type_id column.

compound_taskid_set
: List of character vectors, one for each modeling task in the round. Can be used to override the compound task ID set defined in the config. If NULL is provided for a given modeling task, a compound task ID set of all task IDs is used.

output_types
: Character vector of output type names to include. Use to subset for grids for specific output types.

derived_task_ids
: Character vector of derived task ID names (task IDs whose values depend on other task IDs) to ignore. Columns for such task ids will contain NAs.

Details

When a round is set to round_id_from_variable: true, the value of the task ID from which round IDs are derived (i.e. the task ID specified in round_id property of config_tasks) is set to the value of the round_id argument in the returned output.

When sample output types are included in the output and include_sample_ids = TRUE, the output_type_id column contains example sample indexes which are useful for identifying the compound task ID structure of multivariate sampling distributions in particular, i.e. which combinations of task ID values represent individual samples.

Value

If bind_model_tasks = TRUE (default) a tibble or arrow table containing all possible task ID and related output type ID value combinations. If bind_model_tasks = FALSE, a list containing a tibble or arrow table for each round modeling task.

Columns are coerced to data types according to the hub schema, unless all_character = TRUE. If all_character = TRUE, all columns are returned as character which can be faster when large expanded grids are expected. If required_vals_only = TRUE, values are limited to the combinations of required values only.

Examples

```
hub_con <- hubData::connect_hub(
  system.file("testhubs/flusight", package = "hubUtils")
)
config_tasks <- attr(hub_con, "config_tasks")
expand_model_out_grid(config_tasks, round_id = "2023-01-02")
expand_model_out_grid(
  config_tasks,
  round_id = "2023-01-02",
  required_vals_only = TRUE
)
# Specifying a round in a hub with multiple round configurations.
hub_con <- hubData::connect_hub(
  system.file("testhubs/simple", package = "hubUtils")
)
```

```
config_tasks <- attr(hub_con, "config_tasks")
expand_model_out_grid(config_tasks, round_id = "2022-10-01")
# Later round_id maps to round config that includes additional task ID 'age_group'.
expand_model_out_grid(config_tasks, round_id = "2022-10-29")
# Coerce all columns to character
expand_model_out_grid(config_tasks,
  round_id = "2022-10-29",
  all_character = TRUE
)
# Return arrow table
expand_model_out_grid(config_tasks,
  round_id = "2022-10-29",
  all_character = TRUE,
  as_arrow_table = TRUE
)
# Hub with sample output type
config_tasks <- hubUtils::read_config_file(system.file("config", "tasks.json",
  package = "hubValidations"
))
expand_model_out_grid(config_tasks,
  round_id = "2022-12-26"
)
# Include sample IDS
expand_model_out_grid(config_tasks,
  round_id = "2022-12-26",
  include_sample_ids = TRUE
)
# Hub with sample output type and compound task ID structure
config_tasks <- hubUtils::read_config_file(
  system.file("config", "tasks-comp-tid.json", package = "hubValidations")
)
expand_model_out_grid(config_tasks,
  round_id = "2022-12-26",
  include_sample_ids = TRUE
)
# Override config compound task ID set
# Create coarser compound task ID set for the first modeling task which contains
# samples
expand_model_out_grid(config_tasks,
  round_id = "2022-12-26",
  include_sample_ids = TRUE,
  compound_taskid_set = list(
    c("forecast_date", "target"),
    NULL
  )
)
expand_model_out_grid(config_tasks,
  round_id = "2022-12-26",
  include_sample_ids = TRUE,
  compound_taskid_set = list(
    NULL,
    NULL
  )
```

```
)
# Subset output types
config_tasks <- hubUtils::read_config(
  system.file("testhubs", "samples", package = "hubValidations")
)
expand_model_out_grid(config_tasks,
  round_id = "2022-10-29",
  include_sample_ids = TRUE,
  bind_model_tasks = FALSE,
  output_types = c("sample", "pmf"),
)
expand_model_out_grid(config_tasks,
  round_id = "2022-10-29",
  include_sample_ids = TRUE,
  bind_model_tasks = TRUE,
  output_types = "sample",
)
# Ignore derived task IDs
expand_model_out_grid(config_tasks,
  round_id = "2022-10-29",
  include_sample_ids = TRUE,
  bind_model_tasks = FALSE,
  output_types = "sample",
  derived_task_ids = "target_end_date"
)
```

`get_tbl_compound_taskid_set` *Detect the compound_taskid_set for a tbl for each modeling task in a given round.*

Description

Detect the compound_taskid_set for a tbl for each modeling task in a given round.

Usage

```
get_tbl_compound_taskid_set(
  tbl,
  config_tasks,
  round_id,
  compact = TRUE,
  error = TRUE,
  derived_task_ids = NULL
)
```

Arguments

`tbl`	a tibble/data.frame of the contents of the file being validated. Column types must **all be character**.

config_tasks	a list representantion of the tasks.json config file.
round_id	Character string. The round ID.
compact	Logical. If TRUE, the output will be compacted to remove NULL elements.
error	Logical. If TRUE, an error will be thrown if the compound task ID set is not valid. If FALSE and an error is detected, the detected compound task ID set will be returned with error attributes attached.
derived_task_ids	Character vector of derived task ID names (task IDs whose values depend on other task IDs) to ignore. Columns for such task ids will contain NAs.

Value

A list of vectors of compound task IDs detected in the tbl, one for each modeling task in the round. If compact is TRUE, modeling tasks returning NULL elements will be removed.

Examples

```
hub_path <- system.file("testhubs/samples", package = "hubValidations")
file_path <- "flu-base/2022-10-22-flu-base.csv"
round_id <- "2022-10-22"
tbl <- read_model_out_file(
  file_path = file_path,
  hub_path = hub_path,
  coerce_types = "chr"
)
config_tasks <- hubUtils::read_config(hub_path, "tasks")
get_tbl_compound_taskid_set(tbl, config_tasks, round_id)
get_tbl_compound_taskid_set(tbl, config_tasks, round_id,
  compact = FALSE
)
```

---

is_success *Get status of a hub check*

Description

Get status of a hub check

Usage

```
is_success(x)

is_failure(x)

is_error(x)
```

```
is_info(x)

not_pass(x)

is_exec_error(x)

is_exec_warn(x)

is_any_error(x)
```

Arguments

x	an object that inherits from class <hub_check> to test.

Value

Logical. Is given status of check TRUE?

Functions

- is_success(): Is check success?
- is_failure(): Is check failure?
- is_error(): Is check error?
- is_info(): Is check info?
- not_pass(): Did check not pass?
- is_exec_error(): Is exec error?
- is_exec_warn(): Is exec warning?
- is_any_error(): Is error or exec error?

---

match_tbl_to_model_task

Match model output tbl *data to their model tasks in* config_tasks.

Description

Split and match model output tbl data to their corresponding model tasks in config_tasks. Useful for performing model task specific checks on model output. For v3 samples, the output_type_id column is set to NA for sample outputs.

Usage

```
match_tbl_to_model_task(
  tbl,
  config_tasks,
  round_id,
  output_types = NULL,
  derived_task_ids = NULL,
  all_character = TRUE
)
```

Arguments

tbl	a tibble/data.frame of the contents of the file being validated.
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 `hubUtils::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.
output_types	Character vector of output type names to include. Use to subset for grids for specific output types.
derived_task_ids	Character vector of derived task ID names (task IDs whose values depend on other task IDs) to ignore. Columns for such task ids will contain `NA`s.
all_character	Logical. Whether to return all character column.

Value

A list containing a `tbl_df` of model output data matched to a model task with one element per round model task.

Examples

```
hub_path <- system.file("testhubs/samples", package = "hubValidations")
tbl <- read_model_out_file(
  file_path = "flu-base/2022-10-22-flu-base.csv",
  hub_path, coerce_types = "chr"
)
config_tasks <- hubUtils::read_config(hub_path, "tasks")
match_tbl_to_model_task(tbl, config_tasks, round_id = "2022-10-22")
match_tbl_to_model_task(tbl, config_tasks,
  round_id = "2022-10-22",
  output_types = "sample"
)
```

`new_hub_validations` *Create new or convert list to* `hub_validations` *S3 class object*

Description

Create new or convert list to `hub_validations` S3 class object

Usage

```
new_hub_validations(...)

as_hub_validations(x)
```

Arguments

`...`	named elements to be included. Each element must be an object which inherits from class `<hub_check>`.
`x`	a list of named elements. Each element must be an object which inherits from class `<hub_check>`.

Value

an S3 object of class `<hub_validations>`.

Functions

- `new_hub_validations()`: Create new `<hub_validations>` S3 class object
- `as_hub_validations()`: Convert list to `<hub_validations>` S3 class object

Examples

```
new_hub_validations()

hub_path <- system.file("testhubs/simple", package = "hubValidations")
file_path <- "team1-goodmodel/2022-10-08-team1-goodmodel.csv"
new_hub_validations(
  file_exists = check_file_exists(file_path, hub_path),
  file_name = check_file_name(file_path)
)
x <- list(
  file_exists = check_file_exists(file_path, hub_path),
  file_name = check_file_name(file_path)
)
as_hub_validations(x)
```

`opt_check_metadata_team_max_model_n`
Check that submitting team does not exceed maximum number of allowed models per team

---

Description

Check that submitting team does not exceed maximum number of allowed models per team

Usage

```
opt_check_metadata_team_max_model_n(file_path, hub_path, n_max = 2L)
```

Arguments

`file_path`	character string. Path to the file being validated relative to the hub's model-metadata directory.
`hub_path`	Either a character string path to a local Modeling Hub directory or an object of class `<SubTreeFileSystem>` created using functions `s3_bucket()` or `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. The hub must be fully configured with valid `admin.json` and `tasks.json` files within the `hub-config` directory.
`n_max`	Integer. Number of maximum allowed models per team.

Details

Should be deployed as part of `validate_model_metadata` optional checks.

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_failure>` condition class object.

Returned object also inherits from subclass `<hub_check>`.

`opt_check_tbl_col_timediff` *Check time difference between values in two date columns equal a defined period.*

Description

Check time difference between values in two date columns equal a defined period.

Usage

```
opt_check_tbl_col_timediff(
  tbl,
  file_path,
  hub_path,
  t0_colname,
  t1_colname,
  timediff = lubridate::weeks(2),
  output_type_id_datatype = c("from_config", "auto", "character", "double", "integer",
    "logical", "Date")
)
```

Arguments

`tbl`	a tibble/data.frame of the contents of the file being validated.
`file_path`	character string. Path to the file being validated relative to the hub's model-output directory.
`hub_path`	Either a character string path to a local Modeling Hub directory or an object of class `<SubTreeFileSystem>` created using functions `s3_bucket()` or `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. The hub must be fully configured with valid `admin.json` and `tasks.json` files within the `hub-config` directory.
`t0_colname`	Character string. The name of the time zero date column.
`t1_colname`	Character string. The name of the time zero + 1 time step date column.
`timediff`	an object of class `lubridate` `Period` and length 1.
`output_type_id_datatype`	character string. One of `"from_config"`, `"auto"`, `"character"`, `"double"`, `"integer"`, `"logical"`, `"Date"`. Defaults to `"from_config"` which uses the setting in the `output_type_id_datatype` property in the `tasks.json` config file if available. If the property is not set in the config, the argument falls back to `"auto"` which determines the `output_type_id` data type automatically from the `tasks.json` config file as the simplest data type required to represent all output type ID values across all output types

in the hub. Other data type values can be used to override automatic determination. Note that attempting to coerce `output_type_id` to a data type that is not valid for the data (e.g. trying to coerce`"character"` values to `"double"`) will likely result in an error or potentially unexpected behaviour so use with care.

Details

Should be deployed as part of `validate_model_data` optional checks.

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_failure>` condition class object.

Returned object also inherits from subclass `<hub_check>`.

---

opt_check_tbl_counts_lt_popn

Check that predicted values per location are less than total location population.

Description

Check that predicted values per location are less than total location population.

Usage

```
opt_check_tbl_counts_lt_popn(
  tbl,
  file_path,
  hub_path,
  targets = NULL,
  popn_file_path = "auxiliary-data/locations.csv",
  popn_col = "population",
  location_col = "location"
)
```

Arguments

`tbl`	a tibble/data.frame of the contents of the file being validated.
`file_path`	character string. Path to the file being validated relative to the hub's model-output directory.

hub_path	Either a character string path to a local Modeling Hub directory or an object of class <SubTreeFileSystem> created using functions s3_bucket() or 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. The hub must be fully configured with valid admin.json and tasks.json files within the hub-config directory.
targets	Either a single target key list or a list of multiple target key lists.
popn_file_path	Character string. Path to population data relative to the hub root. Defaults to auxiliary-data/locations.csv.
popn_col	Character string. The name of the population size column in the population data set.
location_col	Character string. The name of the location column. Used to join population data to submission file data. Must be shared by both files.

Details

Should only be applied to rows containing count predictions. Use argument targets to filter tbl data to appropriate count target rows.

Should be deployed as part of validate_model_data optional checks.

Value

Depending on whether validation has succeeded, one of:

- <message/check_success> condition class object.
- <error/check_failure> condition class object.

Returned object also inherits from subclass <hub_check>.

Examples

```
hub_path <- system.file("testhubs/flusight", package = "hubValidations")
file_path <- "hub-ensemble/2023-05-08-hub-ensemble.parquet"
tbl <- hubValidations::read_model_out_file(file_path, hub_path)
# Single target key list
targets <- list("target" = "wk ahead inc flu hosp")
opt_check_tbl_counts_lt_popn(tbl, file_path, hub_path, targets = targets)
```

---

opt_check_tbl_horizon_timediff

Check time difference between values in two date columns equals a defined time period defined by values in a horizon column

---

Description

Check time difference between values in two date columns equals a defined time period defined by values in a horizon column

Usage

```
opt_check_tbl_horizon_timediff(
  tbl,
  file_path,
  hub_path,
  t0_colname,
  t1_colname,
  horizon_colname = "horizon",
  timediff = lubridate::weeks(),
  output_type_id_datatype = c("from_config", "auto", "character", "double", "integer",
    "logical", "Date")
)
```

Arguments

`tbl`	a tibble/data.frame of the contents of the file being validated.
`file_path`	character string. Path to the file being validated relative to the hub's model-output directory.
`hub_path`	Either a character string path to a local Modeling Hub directory or an object of class `<SubTreeFileSystem>` created using functions `s3_bucket()` or `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. The hub must be fully configured with valid `admin.json` and `tasks.json` files within the `hub-config` directory.
`t0_colname`	Character string. The name of the time zero date column.
`t1_colname`	Character string. The name of the time zero + 1 time step date column.
`horizon_colname`	Character string. The name of the horizon column. Defaults to `"horizon"`.
`timediff`	an object of class `lubridate` `Period` and length 1. The period of a single horizon. Default to 1 week.
`output_type_id_datatype`	character string. One of `"from_config"`, `"auto"`, `"character"`, `"double"`, `"integer"`, `"logical"`, `"Date"`. Defaults to `"from_config"` which uses the setting in the `output_type_id_datatype` property in the `tasks.json` config file if available. If the property is not set in the config, the argument falls back to `"auto"` which determines the `output_type_id` data type automatically from the `tasks.json` config file as the simplest data type required to represent all output type ID values across all output types in the hub. Other data type values can be used to override automatic determination. Note that attempting to coerce `output_type_id` to a data

type that is not valid for the data (e.g. trying to coerce`"character"` values to `"double"`) will likely result in an error or potentially unexpected behaviour so use with care.

Details

Should be deployed as part of `validate_model_data` optional checks.

Value

Depending on whether validation has succeeded, one of:

- `<message/check_success>` condition class object.
- `<error/check_failure>` condition class object.

Returned object also inherits from subclass `<hub_check>`.

`parse_file_name` *Parse model output file metadata from file name*

Description

Parse model output file metadata from file name

Usage

```
parse_file_name(file_path, file_type = c("model_output", "model_metadata"))
```

Arguments

`file_path`	Character string. A model output file name. Can include parent directories which are ignored.
`file_type`	Character string. Type of file name being parsed. One of `"model_output"` or `"model_metadata"`.

Details

File names are allowed to contain the following compression extension prefixes: .snappy, .gzip, .gz, .brotli, .zstd, .lz4, .lzo, .bz2. These extension prefixes are now extracted when parsing the file name and returned as `compression_ext` element if present.

Value

A list with the following elements:

- `round_id`: The round ID the model output is associated with (`NA` for model metadata files.)
- `team_abbr`: The team responsible for the model.
- `model_abbr`: The name of the model.

- `model_id`: The unique model ID derived from the concatenation of `<team_abbr>-<model_abbr>`.
- `ext`: The file extension.
- `compression_ext`: optional. The compression extension if present.

Examples

```
parse_file_name("hub-baseline/2022-10-15-hub-baseline.csv")
parse_file_name("hub-baseline/2022-10-15-hub-baseline.gzip.parquet")
```

---

print.hub_validations

Print results of `validate_...()` *function as a bullet list*

Description

Print results of `validate_...()` function as a bullet list

Usage

```
## S3 method for class 'hub_validations'
print(x, ...)
```

Arguments

`x`	An object of class `hub_validations`
`...`	Unused argument present for class consistency

---

print.pr_hub_validations

Print results of `validate_pr()` *function as a bullet list*

Description

Print results of `validate_pr()` function as a bullet list

Usage

```
## S3 method for class 'pr_hub_validations'
print(x, ...)
```

Arguments

`x`	An object of class `pr_hub_validations`
`...`	Unused argument present for class consistency

`read_model_out_file` *Read a model output file*

Description

Read a model output file

Usage

```
read_model_out_file(
  file_path,
  hub_path = ".",
  coerce_types = c("hub", "chr", "none"),
  output_type_id_datatype = c("from_config", "auto", "character", "double", "integer",
    "logical", "Date")
)
```

Arguments

`file_path`	character string. Path to the file being validated relative to the hub's model-output directory.
`hub_path`	Either a character string path to a local Modeling Hub directory or an object of class `<SubTreeFileSystem>` created using functions `s3_bucket()` or `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. The hub must be fully configured with valid `admin.json` and `tasks.json` files within the `hub-config` directory.
`coerce_types`	character. What to coerce column types to on read. • `hub`: (default) read in (`csv`) or coerce (`parquet`, `arrow`) to hub schema. When coercing data types using the `hub` schema, the `output_type_id_datatype` can also be used to set the `output_type_id` column data type manually. • `chr`: read in (`csv`) or coerce (`parquet`, `arrow`) all columns to character. • `none`: No coercion. Use `arrow` `read_*` function defaults.
`output_type_id_datatype`	character string. One of `"from_config"`, `"auto"`, `"character"`, `"double"`, `"integer"`, `"logical"`, `"Date"`. Defaults to `"from_config"` which uses the setting in the `output_type_id_datatype` property in the `tasks.json` config file if available. If the property is not set in the config, the argument falls back to `"auto"` which determines the `output_type_id` data type automatically from the `tasks.json` config file as the simplest data type required to represent all output type ID values across all output types in the hub. Other data type values can be used to override automatic determination. Note that attempting to coerce `output_type_id` to a data

type that is not valid for the data (e.g. trying to coerce`"character"` values to `"double"`) will likely result in an error or potentially unexpected behaviour so use with care.

Value

a tibble of contents of the model output file.

`submission_tmpl` *Create a model output submission file template*

Description

Create a model output submission file template

Usage

```
submission_tmpl(
  hub_con,
  config_tasks,
  round_id,
  required_vals_only = FALSE,
  complete_cases_only = TRUE,
  compound_taskid_set = NULL,
  output_types = NULL,
  derived_task_ids = NULL
)
```

Arguments

`hub_con`	A `<hub_connection>` class object.
`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 `hubUtils::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.
`required_vals_only`	Logical. Whether to return only combinations of Task ID and related output type ID required values.
`complete_cases_only`	Logical. If `TRUE` (default) and `required_vals_only = TRUE`, only rows with complete cases of combinations of required values are returned. If `FALSE`, rows with incomplete cases of combinations of required values are included in the output.

`compound_taskid_set`
: List of character vectors, one for each modeling task in the round. Can be used to override the compound task ID set defined in the config. If `NULL` is provided for a given modeling task, a compound task ID set of all task IDs is used.

`output_types`
: Character vector of output type names to include. Use to subset for grids for specific output types.

`derived_task_ids`
: Character vector of derived task ID names (task IDs whose values depend on other task IDs) to ignore. Columns for such task ids will contain `NA`s.

Details

For task IDs or output_type_ids where all values are optional, by default, columns are included as columns of `NA`s when `required_vals_only = TRUE`. When such columns exist, the function returns a tibble with zero rows, as no complete cases of required value combinations exists. *(Note that determination of complete cases does excludes valid* `NA` `output_type_id` *values in* `"mean"` *and* `"median"` *output types).* To return a template of incomplete required cases, which includes `NA` columns, use `complete_cases_only = FALSE`.

When sample output types are included in the output, the `output_type_id` column contains example sample indexes which are useful for identifying the compound task ID structure of multivariate sampling distributions in particular, i.e. which combinations of task ID values represent individual samples.

When a round is set to `round_id_from_variable: true`, the value of the task ID from which round IDs are derived (i.e. the task ID specified in `round_id` property of `config_tasks`) is set to the value of the `round_id` argument in the returned output.

Value

a tibble template containing an expanded grid of valid task ID and output type ID value combinations for a given submission round and output type. If `required_vals_only = TRUE`, values are limited to the combination of required values only.

Examples

```
hub_con <- hubData::connect_hub(
  system.file("testhubs/flusight", package = "hubUtils")
)
submission_tmpl(hub_con, round_id = "2023-01-02")
submission_tmpl(
  hub_con,
  round_id = "2023-01-02",
  required_vals_only = TRUE
)
submission_tmpl(
  hub_con,
  round_id = "2023-01-02",
  required_vals_only = TRUE,
  complete_cases_only = FALSE
)
```

```
# Specifying a round in a hub with multiple rounds
hub_con <- hubData::connect_hub(
  system.file("testhubs/simple", package = "hubUtils")
)
submission_tmpl(hub_con, round_id = "2022-10-01")
submission_tmpl(hub_con, round_id = "2022-10-29")
submission_tmpl(hub_con,
  round_id = "2022-10-29",
  required_vals_only = TRUE
)
submission_tmpl(hub_con,
  round_id = "2022-10-29",
  required_vals_only = TRUE,
  complete_cases_only = FALSE
)
# Hub with sample output type
config_tasks <- hubUtils::read_config_file(system.file("config", "tasks.json",
  package = "hubValidations"
))
submission_tmpl(
  config_tasks = config_tasks,
  round_id = "2022-12-26"
)
# Hub with sample output type and compound task ID structure
config_tasks <- hubUtils::read_config_file(system.file("config", "tasks-comp-tid.json",
  package = "hubValidations"
))
submission_tmpl(
  config_tasks = config_tasks,
  round_id = "2022-12-26"
)
# Override config compound task ID set
# Create coarser compound task ID set for the first modeling task which contains
# samples
submission_tmpl(
  config_tasks = config_tasks,
  round_id = "2022-12-26",
  compound_taskid_set = list(
    c("forecast_date", "target"),
    NULL
  )
)
# Subsetting for a single output type
submission_tmpl(
  config_tasks = config_tasks,
  round_id = "2022-12-26",
  output_types = "sample"
)
# Derive a template with ignored derived task ID. Useful to avoid creating
# a template with invalid derived task ID value combinations.
config_tasks <- hubUtils::read_config(
  system.file("testhubs", "flusight", package = "hubValidations")
)
```

```
submission_tmpl(
  config_tasks = config_tasks,
  round_id = "2022-12-12",
  output_types = "pmf",
  derived_task_ids = "target_end_date",
  complete_cases_only = FALSE
)
```

`try_check` *Wrap check expression in try to capture check execution errors*

Description

Wrap check expression in try to capture check execution errors

Usage

```
try_check(expr, file_path)
```

Arguments

`expr`	check function expression to run.
`file_path`	character string. Path to the file being validated relative to the hub's model-output directory.

Value

If `expr` executes correctly, the output of `expr` is returned. If execution fails, and object of class `<error/check_exec_error>` is returned. The execution error message is attached as attribute `msg`.

`validate_model_data` *Validate the contents of a submitted model data file*

Description

Validate the contents of a submitted model data file

Usage

```
validate_model_data(
  hub_path,
  file_path,
  round_id_col = NULL,
  output_type_id_datatype = c("from_config", "auto", "character", "double", "integer",
    "logical", "Date"),
  validations_cfg_path = NULL,
  derived_task_ids = NULL
)
```

Arguments

hub_path	Either a character string path to a local Modeling Hub directory or an object of class <SubTreeFileSystem> created using functions s3_bucket() or 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. The hub must be fully configured with valid admin.json and tasks.json files within the hub-config directory.
file_path	character string. Path to the file being validated relative to the hub's model-output directory.
round_id_col	Character string. The name of the column containing round_ids. Usually, the value of round property round_id in hub tasks.json config file.
output_type_id_datatype	character string. One of "from_config", "auto", "character", "double", "integer", "logical", "Date". Defaults to "from_config" which uses the setting in the output_type_id_datatype property in the tasks.json config file if available. If the property is not set in the config, the argument falls back to "auto" which determines the output_type_id data type automatically from the tasks.json config file as the simplest data type required to represent all output type ID values across all output types in the hub. Other data type values can be used to override automatic determination. Note that attempting to coerce output_type_id to a data type that is not valid for the data (e.g. trying to coerce"character" values to "double") will likely result in an error or potentially unexpected behaviour so use with care.
validations_cfg_path	Path to validations.yml file. If NULL defaults to hub-config/validations.yml.
derived_task_ids	Character vector of derived task ID names (task IDs whose values depend on other task IDs) to ignore. Columns for such task ids will contain NAs.

Details

Details of checks performed by validate_model_data()

Value

An object of class hub_validations. Each named element contains a hub_check class object reflecting the result of a given check. Function will return early if a check returns an error.

For more details on the structure of <hub_validations> objects, including how to access more information on individual checks, see article on <hub_validations> S3 class objects.

Examples

```
hub_path <- system.file("testhubs/simple", package = "hubValidations")
file_path <- "team1-goodmodel/2022-10-08-team1-goodmodel.csv"
validate_model_data(hub_path, file_path)
```

validate_model_file *Valid file level properties of a submitted model output file.*

Description

Valid file level properties of a submitted model output file.

Usage

```
validate_model_file(hub_path, file_path, validations_cfg_path = NULL)
```

Arguments

hub_path	Either a character string path to a local Modeling Hub directory or an object of class <SubTreeFileSystem> created using functions s3_bucket() or 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. The hub must be fully configured with valid admin.json and tasks.json files within the hub-config directory.
file_path	character string. Path to the file being validated relative to the hub's model-output directory.
validations_cfg_path	Path to validations.yml file. If NULL defaults to hub-config/validations.yml.

Details

Details of checks performed by validate_model_file()

Value

An object of class hub_validations. Each named element contains a hub_check class object reflecting the result of a given check. Function will return early if a check returns an error.

For more details on the structure of <hub_validations> objects, including how to access more information on individual checks, see article on <hub_validations> S3 class objects.

Examples

```
hub_path <- system.file("testhubs/simple", package = "hubValidations")
validate_model_file(hub_path,
  file_path = "team1-goodmodel/2022-10-08-team1-goodmodel.csv"
)
validate_model_file(hub_path,
  file_path = "team1-goodmodel/2022-10-15-team1-goodmodel.csv"
)
```

`validate_model_metadata` *Valid properties of a metadata file.*

Description

Valid properties of a metadata file.

Usage

```
validate_model_metadata(
  hub_path,
  file_path,
  round_id = "default",
  validations_cfg_path = NULL
)
```

Arguments

`hub_path`	Either a character string path to a local Modeling Hub directory or an object of class `<SubTreeFileSystem>` created using functions `s3_bucket()` or `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. The hub must be fully configured with valid `admin.json` and `tasks.json` files within the `hub-config` directory.
`file_path`	character string. Path to the file being validated relative to the hub's model-output directory.
`round_id`	character string. The round identifier. Used primarily to indicate whether the "default" or a round specific configuration should be used for custom validations.
`validations_cfg_path`	Path to `validations.yml` file. If `NULL` defaults to `hub-config/validations.yml`.

Details

Details of checks performed by `validate_model_metadata()`

Value

An object of class `hub_validations`. Each named element contains a `hub_check` class object reflecting the result of a given check. Function will return early if a check returns an error.

Examples

```
hub_path <- system.file("testhubs/simple", package = "hubValidations")
validate_model_metadata(hub_path,
  file_path = "hub-baseline.yml"
)
validate_model_metadata(hub_path,
  file_path = "team1-goodmodel.yaml"
)
```

---

validate_pr *Validate Pull Request*

Description

Validates model output and model metadata files in a Pull Request.

Usage

```
validate_pr(
  hub_path = ".",
  gh_repo,
  pr_number,
  round_id_col = NULL,
  output_type_id_datatype = c("from_config", "auto", "character", "double", "integer",
    "logical", "Date"),
  validations_cfg_path = NULL,
  skip_submit_window_check = FALSE,
  file_modification_check = c("error", "failure", "warn", "message", "none"),
  allow_submit_window_mods = TRUE,
  submit_window_ref_date_from = c("file", "file_path"),
  derived_task_ids = NULL
)
```

Arguments

hub_path	Either a character string path to a local Modeling Hub directory or an object of class <SubTreeFileSystem> created using functions s3_bucket() or 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. The hub must be fully configured with valid admin.json and tasks.json files within the hub-config directory.
gh_repo	GitHub repository address in the format username/repo
pr_number	Number of the pull request to validate
round_id_col	Character string. The name of the column containing round_ids. Only required if files contain a column that contains round_id details but has not been configured via round_id_from_variable: true and round_id: in in hub tasks.json config file.

`output_type_id_datatype`
: character string. One of `"from_config"`, `"auto"`, `"character"`, `"double"`, `"integer"`, `"logical"`, `"Date"`. Defaults to `"from_config"` which uses the setting in the `output_type_id_datatype` property in the `tasks.json` config file if available. If the property is not set in the config, the argument falls back to `"auto"` which determines the `output_type_id` data type automatically from the `tasks.json` config file as the simplest data type required to represent all output type ID values across all output types in the hub. Other data type values can be used to override automatic determination. Note that attempting to coerce `output_type_id` to a data type that is not valid for the data (e.g. trying to coerce`"character"` values to `"double"`) will likely result in an error or potentially unexpected behaviour so use with care.

`validations_cfg_path`
: Path to `validations.yml` file. If `NULL` defaults to `hub-config/validations.yml`.

`skip_submit_window_check`
: Logical. Whether to skip the submission window check.

`file_modification_check`
: Character string. Whether to perform check and what to return when modification/deletion of a previously submitted model output file or deletion of a previously submitted model metadata file is detected in PR:

  - `"error"`: Appends a `<error/check_error>` condition class object for each applicable modified/deleted file.
  - `"warning"`: Appends a `<error/check_failure>` condition class object for each applicable modified/deleted file.
  - `"message"`: Appends a `<message/check_info>` condition class object for each applicable modified/deleted file.
  - `"none"`: No modification/deletion checks performed.

`allow_submit_window_mods`
: Logical. Whether to allow modifications/deletions of model output files within their submission windows. Defaults to `TRUE`.

`submit_window_ref_date_from`
: whether to get the reference date around which relative submission windows will be determined from the file's `file_path` round ID or the `file` contents themselves. `file` requires that the file can be read. Only applicable when a round is configured to determine the submission windows relative to the value in a date column in model output files. Not applicable when explicit submission window start and end dates are provided in the hub's config.

`derived_task_ids`
: Character vector of derived task ID names (task IDs whose values depend on other task IDs) to ignore. Columns for such task ids will contain `NA`s.

Details

Only model output and model metadata files are individually validated using `validate_submission()` or `validate_model_metadata()` respectively although as part of checks, hub config files are also validated. Any other files included in the PR are ignored but flagged in a message.

By default, modifications (which include renaming) and deletions of previously submitted model output files and deletions or renaming of previously submitted model metadata files are not allowed and return a `<error/check_error>` condition class object for each applicable modified/deleted file. This behaviour can be modified through arguments `file_modification_check`, which controls whether modification/deletion checks are performed and what is returned if modifications/deletions are detected, and `allow_submit_window_mods`, which controls whether modifications/deletions of model output files are allowed within their submission windows.

Note that to establish **relative** submission windows when performing modification/deletion checks and `allow_submit_window_mods` is `TRUE`, the reference date is taken as the `round_id` extracted from the file path (i.e. `submit_window_ref_date_from` is always set to `"file_path"`). This is because we cannot extract dates from columns of deleted files. If hub submission window reference dates do not match round IDs in file paths, currently `allow_submit_window_mods` will not work correctly and is best set to `FALSE`. This only relates to hubs/rounds where submission windows are determined relative to a reference date and not when explicit submission window start and end dates are provided in the config.

Checks on model output files:

Details of checks performed by `validate_submission()`

Checks on model metadata files:

Details of checks performed by `validate_model_metadata()`

Value

An object of class `hub_validations`.

Examples

```
## Not run:
validate_pr(
  hub_path = ".",
  gh_repo = "hubverse-org/ci-testhub-simple",
  pr_number = 3
)

## End(Not run)
```

`validate_submission` *Validate a submitted model data file.*

Description

Checks both file level properties like file name, extension, location etc as well as model output data, i.e. the contents of the file.

Usage

```
validate_submission(
  hub_path,
  file_path,
  round_id_col = NULL,
  validations_cfg_path = NULL,
  output_type_id_datatype = c("from_config", "auto", "character", "double", "integer",
    "logical", "Date"),
  skip_submit_window_check = FALSE,
  skip_check_config = FALSE,
  submit_window_ref_date_from = c("file", "file_path"),
  derived_task_ids = NULL
)
```

Arguments

`hub_path`	Either a character string path to a local Modeling Hub directory or an object of class `<SubTreeFileSystem>` created using functions `s3_bucket()` or `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. The hub must be fully configured with valid `admin.json` and `tasks.json` files within the `hub-config` directory.
`file_path`	character string. Path to the file being validated relative to the hub's model-output directory.
`round_id_col`	Character string. The name of the column containing `round_id`s. Usually, the value of round property `round_id` in hub `tasks.json` config file.
`validations_cfg_path`	Path to `validations.yml` file. If `NULL` defaults to `hub-config/validations.yml`.
`output_type_id_datatype`	character string. One of `"from_config"`, `"auto"`, `"character"`, `"double"`, `"integer"`, `"logical"`, `"Date"`. Defaults to `"from_config"` which uses the setting in the `output_type_id_datatype` property in the `tasks.json` config file if available. If the property is not set in the config, the argument falls back to `"auto"` which determines the `output_type_id` data type automatically from the `tasks.json` config file as the simplest data type required to represent all output type ID values across all output types in the hub. Other data type values can be used to override automatic determination. Note that attempting to coerce `output_type_id` to a data type that is not valid for the data (e.g. trying to coerce`"character"` values to `"double"`) will likely result in an error or potentially unexpected behaviour so use with care.
`skip_submit_window_check`	Logical. Whether to skip the submission window check.
`skip_check_config`	Logical. Whether to skip the hub config validation check. check.

`submit_window_ref_date_from`
: whether to get the reference date around which relative submission windows will be determined from the file's `file_path` round ID or the `file` contents themselves. `file` requires that the file can be read. Only applicable when a round is configured to determine the submission windows relative to the value in a date column in model output files. Not applicable when explicit submission window start and end dates are provided in the hub's config.

`derived_task_ids`
: Character vector of derived task ID names (task IDs whose values depend on other task IDs) to ignore. Columns for such task ids will contain `NA`s.

Details

Details of checks performed by `validate_submission()`

Value

An object of class `hub_validations`. Each named element contains a `hub_check` class object reflecting the result of a given check. Function will return early if a check returns an error.

For more details on the structure of `<hub_validations>` objects, including how to access more information on individual checks, see article on `<hub_validations>` S3 class objects.

Examples

```
hub_path <- system.file("testhubs/simple", package = "hubValidations")
file_path <- "team1-goodmodel/2022-10-08-team1-goodmodel.csv"
validate_submission(hub_path, file_path)
```

`validate_submission_time` *Validate a submitted model data file submission time.*

Description

Validate a submitted model data file submission time.

Usage

```
validate_submission_time(
  hub_path,
  file_path,
  ref_date_from = c("file_path", "file")
)
```

Arguments

hub_path	Either a character string path to a local Modeling Hub directory or an object of class <SubTreeFileSystem> created using functions s3_bucket() or 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. The hub must be fully configured with valid admin.json and tasks.json files within the hub-config directory.
file_path	character string. Path to the file being validated relative to the hub's model-output directory.
ref_date_from	whether to get the reference date around which relative submission windows will be determined from the file's file_path round ID or the file contents themselves. file requires that the file can be read. Only applicable when a round is configured to determine the submission windows relative to the value in a date column in model output files. Not applicable when explicit submission window start and end dates are provided in the hub's config.

Value

An object of class hub_validations. Each named element contains a hub_check class object reflecting the result of a given check. Function will return early if a check returns an error.

For more details on the structure of <hub_validations> objects, including how to access more information on individual checks, see article on <hub_validations> S3 class objects.

Examples

```
hub_path <- system.file("testhubs/simple", package = "hubValidations")
file_path <- "team1-goodmodel/2022-10-08-team1-goodmodel.csv"
validate_submission_time(hub_path, file_path)
```

Index

as_hub_validations
(*new_hub_validations*), 41

capture_check_cnd, 3
capture_check_info, 4
capture_exec_error, 5
capture_exec_warning, 6
check_config_hub_valid, 6
check_file_exists, 7
check_file_format, 8
check_file_location, 8
check_file_name, 9
check_file_read, 10
check_for_errors, 10
check_metadata_file_exists, 11
check_metadata_file_ext, 12
check_metadata_file_location, 12
check_metadata_file_name, 13
check_metadata_matches_schema, 14
check_metadata_schema_exists, 14
check_submission_metadata_file_exists, 15
check_submission_time, 16
check_tbl_col_types, 17
check_tbl_colnames, 17
check_tbl_match_round_id, 19
check_tbl_rows_unique, 20
check_tbl_spl_compound_taskid_set, 20
check_tbl_spl_compound_tid, 22
check_tbl_spl_n, 23
check_tbl_spl_non_compound_tid, 25
check_tbl_unique_round_id, 26
check_tbl_value_col, 29
check_tbl_value_col_ascending, 30
check_tbl_value_col_sum1, 31
check_tbl_values, 27
check_tbl_values_required, 28
check_valid_round_id, 31
check_valid_round_id_col, 32
cli::format_inline(), *4*, *5*

combine, 33

dynamic, *4*

expand_model_out_grid, 33

get_tbl_compound_taskid_set, 37
gs_bucket(), *6–8*, *10*, *11*, *13–21*, *23–29*, *32*, *42*, *43*, *45*, *46*, *49*, *54–57*, *60*, *62*

hubUtils::read_config(), *34*, *40*, *50*

is_any_error (*is_success*), 38
is_error (*is_success*), 38
is_exec_error (*is_success*), 38
is_exec_warn (*is_success*), 38
is_failure (*is_success*), 38
is_info (*is_success*), 38
is_success, 38

match_tbl_to_model_task, 39

new_hub_validations, 41
not_pass (*is_success*), 38

opt_check_metadata_team_max_model_n, 42
opt_check_tbl_col_timediff, 43
opt_check_tbl_counts_lt_popn, 44
opt_check_tbl_horizon_timediff, 45

parse_file_name, 47
Period, *43*, *46*
print.hub_validations, 48
print.pr_hub_validations, 48

read_model_out_file, 49
rlang::stack, *5*

s3_bucket(), *6–8*, *10*, *11*, *13–21*, *23–29*, *32*, *42*, *43*, *45*, *46*, *49*, *54–57*, *60*, *62*

submission_tmpl, 50

try, *5, 6*
try_check, 53

validate_model_data, 53
validate_model_file, 55
validate_model_metadata, 56
validate_pr, 57
validate_submission, 59
validate_submission_time, 61