Package 'pointblank'

May 2, 2018

Type Package

Title Validation of Local and Remote Data Tables

Version 0.2.0

Description Validate data in data frames, 'tibble' objects, in 'CSV' and 'TSV' files, and in database tables ('PostgreSQL' and 'MySQL'). Validation pipelines can be made using easily-readable, consecutive validation steps and such pipelines allow for switching of the data table context. Upon execution of the validation plan, several reporting options are available. User-defined thresholds for failure rates allow for the determination of appropriate reporting actions.

Depends R (>= 3.1.2)

License MIT + file LICENSE

Imports commonmark (>= 1.4), DBI (>= 0.8), dplyr (>= 0.7.4), glue (>= 1.2.0), httr (>= 1.3.1), magrittr (>= 1.5), purrr (>= 0.2.4), readr (>= 1.1.1), rlang (>= 0.2.0), rmarkdown (>= 1.9), RMySQL (>= 0.10.14), RPostgreSQL (>= 0.6-2), stringr (>= 1.3.0), tibble (>= 1.4.2), tidyr (>= 0.8.0)

Suggests testthat

URL https://github.com/rich-iannone/pointblank

BugReports https://github.com/rich-iannone/pointblank/issues

Encoding UTF-8

RoxygenNote 6.0.1

NeedsCompilation no

Author Richard Iannone [aut, cre] (<https://orcid.org/0000-0003-3925-190X>)

Maintainer Richard Iannone <riannone@me.com>

Repository CRAN

Date/Publication 2018-05-02 07:51:25 UTC

R topics documented:

all_cols *Perform validations on all table columns When used in conjunction with the* column *argument, the* all_cols() *helper function will indicate that the validation should occur for each and every column in the table.*

Description

Perform validations on all table columns When used in conjunction with the column argument, the all_cols() helper function will indicate that the validation should occur for each and every column in the table.

Usage

```
all_cols()
```

all_passed *Did all of the validations pass?*

Description

Given a completed validation pass, this function will return TRUE or FALSE on whether all of the validations passed without any fails.

Usage

```
all_passed(agent)
```

Arguments

agent	an agent object of class ptblank_agent.

Value

an agent object.

col_exists *Verify that one or more columns exist*

Description

Set a verification step that checks whether one or several specified columns exist in the target table.

Usage

```
col_exists(agent, column, brief = NULL, warn_count = 1,
  notify_count = NULL, warn_fraction = NULL, notify_fraction = NULL,
  tbl_name = NULL, db_type = NULL, creds_file = NULL,
  initial_sql = NULL, file_path = NULL, col_types = NULL)
```

Arguments

agent	an agent object of class ptblank_agent.
column	the name of a single table column or multiple columns in the same table.
brief	an optional, text-based description for the validation step.
warn_count	the threshold number for individual validations returning a FALSE result before applying the warn flag.
notify_count	the threshold number for individual validations returning a FALSE result before applying the notify flag.
warn_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.
notify_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.
tbl_name	the name of the local or remote table.
db_type	if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.
creds_file	if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.
initial_sql	when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').

`file_path` an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

`col_types` if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: `c` -> character, `i` -> integer, `n` -> number, `d` -> double, `l` -> logical, `D` -> date, `T` -> date time, `t` -> time, `?` -> guess, or `_/-`, which skips the column.

Value

an agent object.

Examples

```
# Validate that column `a` exists in
# the `small_table` CSV file; do this
# by creating an agent, focussing on
# that table, creating a `col_exists()`
# step, and then interrogating the table
agent <-
  create_agent() %>%
  focus_on(
    file_name =
      system.file(
        "extdata", "small_table.csv",
        package = "pointblank"),
    col_types = "TDicidlc") %>%
  col_exists(column = a) %>%
  interrogate()

# Determine if this column validation
# passed by using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

`col_is_character` *Verify that a column contains character/string data*

Description

Set a verification step where a table column is expected to consist of string data.

Usage

```
col_is_character(agent, column, brief = NULL, warn_count = 1,
  notify_count = NULL, warn_fraction = NULL, notify_fraction = NULL,
  tbl_name = NULL, db_type = NULL, creds_file = NULL,
  initial_sql = NULL, file_path = NULL, col_types = NULL)
```

Arguments

agent	an agent object of class ptblank_agent.
column	the name of a single table column, multiple columns in the same table, or, a helper function such as all_cols().
brief	an optional, text-based description for the validation step.
warn_count	the threshold number for individual validations returning a FALSE result before applying the warn flag.
notify_count	the threshold number for individual validations returning a FALSE result before applying the notify flag.
warn_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.
notify_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.
tbl_name	the name of the local or remote table.
db_type	if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.
creds_file	if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.
initial_sql	when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').
file_path	an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.
col_types	if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.

Value

an agent object.

Examples

```
# Create a simple data frame
# with a column containing data
```

```
# classed as `character`
df <-
  data.frame(
    a = c("one", "two"),
    stringsAsFactors = FALSE)

# Validate that column `a`
# in the data frame is classed
# as `character`
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_is_character(
    column = a) %>%
  interrogate()

# Determine if these column
# validations have all passed
# by using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

`col_is_date` *Verify that a column contains R Date objects*

Description

Set a verification step where a table column is expected to consist entirely of R `Date` objects.

Usage

```
col_is_date(agent, column, brief = NULL, warn_count = 1,
  notify_count = NULL, warn_fraction = NULL, notify_fraction = NULL,
  tbl_name = NULL, db_type = NULL, creds_file = NULL,
  initial_sql = NULL, file_path = NULL, col_types = NULL)
```

Arguments

`agent`	an agent object of class `ptblank_agent`.
`column`	the name of a single table column, multiple columns in the same table, or, a helper function such as `all_cols()`.
`brief`	an optional, text-based description for the validation step.
`warn_count`	the threshold number for individual validations returning a `FALSE` result before applying the `warn` flag.
`notify_count`	the threshold number for individual validations returning a `FALSE` result before applying the `notify` flag.

warn_fraction — the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.

notify_fraction — the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.

tbl_name — the name of the local or remote table.

db_type — if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.

creds_file — if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.

initial_sql — when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').

file_path — an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

col_types — if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.

Value

an agent object.

Examples

```
# Create a simple data frame
# with a column containing data
# classed as `Date`
df <-
  data.frame(
    a = as.Date("2017-08-15"))

# Validate that column `a`
# in the data frame is classed
# as `Date`
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_is_date(
```

```
    column = a) %>%
  interrogate()

# Determine if these column
# validations have all passed
# by using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

---

col_is_factor *Verify that a column contains R factor objects*

Description

Set a verification step where a table column is expected to consist entirely of R `factor` objects.

Usage

```
col_is_factor(agent, column, brief = NULL, warn_count = 1,
  notify_count = NULL, warn_fraction = NULL, notify_fraction = NULL,
  tbl_name = NULL, db_type = NULL, creds_file = NULL,
  initial_sql = NULL, file_path = NULL, col_types = NULL)
```

Arguments

`agent`	an agent object of class `ptblank_agent`.
`column`	the name of a single table column, multiple columns in the same table, or, a helper function such as `all_cols()`.
`brief`	an optional, text-based description for the validation step.
`warn_count`	the threshold number for individual validations returning a `FALSE` result before applying the `warn` flag.
`notify_count`	the threshold number for individual validations returning a `FALSE` result before applying the `notify` flag.
`warn_fraction`	the threshold fraction for individual validations returning a `FALSE` over all the entire set of individual validations. Beyond this threshold, the `warn` flag will be applied.
`notify_fraction`	the threshold fraction for individual validations returning a `FALSE` over all the entire set of individual validations. Beyond this threshold, the `notify` flag will be applied.
`tbl_name`	the name of the local or remote table.
`db_type`	if the table is located in a database, the type of database is required here. Currently, this can be either `PostgreSQL` or `MySQL`.

creds_file — if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.

initial_sql — when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').

file_path — an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

col_types — if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.

Value

an agent object.

Examples

```
# Create a simple data frame
# with a column containing data
# classed as `factor`
df <-
  data.frame(
    a = c("one", "two"))

# Validate that column `a`
# in the data frame is classed
# as `factor`
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_is_factor(column = a) %>%
  interrogate()

# Determine if these column
# validations have all passed
# by using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

col_is_integer *Verify that a column contains integer values*

Description

Set a verification step where a table column is expected to consist of integer values.

Usage

```
col_is_integer(agent, column, brief = NULL, warn_count = 1,
  notify_count = NULL, warn_fraction = NULL, notify_fraction = NULL,
  tbl_name = NULL, db_type = NULL, creds_file = NULL,
  initial_sql = NULL, file_path = NULL, col_types = NULL)
```

Arguments

agent	an agent object of class ptblank_agent.
column	the name of a single table column, multiple columns in the same table, or, a helper function such as all_cols().
brief	an optional, text-based description for the validation step.
warn_count	the threshold number for individual validations returning a FALSE result before applying the warn flag.
notify_count	the threshold number for individual validations returning a FALSE result before applying the notify flag.
warn_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.
notify_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.
tbl_name	the name of the local or remote table.
db_type	if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.
creds_file	if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.
initial_sql	when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').

file_path	an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.
col_types	if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: `c` -> character, `i` -> integer, `n` -> number, `d` -> double, `l` -> logical, `D` -> date, `T` -> date time, `t` -> time, `?` -> guess, or `_/-`, which skips the column.

Value

an agent object.

Examples

```
# Create a simple data frame
# with a column containing data
# classed as `integer`
df <-
  data.frame(
    a = as.integer(c(5, 9, 3)))

# Validate that column `a`
# in the data frame is classed
# as `integer`
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_is_integer(
    column = a) %>%
  interrogate()

# Determine if these column
# validations have all passed
# by using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

---

col_is_logical *Verify that a column contains logical values*

Description

Set a verification step where a table column is expected to consist of `logical` values.

Usage

```
col_is_logical(agent, column, brief = NULL, warn_count = 1,
  notify_count = NULL, warn_fraction = NULL, notify_fraction = NULL,
  tbl_name = NULL, db_type = NULL, creds_file = NULL,
  initial_sql = NULL, file_path = NULL, col_types = NULL)
```

Arguments

agent	an agent object of class ptblank_agent.
column	the name of a single table column, multiple columns in the same table, or, a helper function such as all_cols().
brief	an optional, text-based description for the validation step.
warn_count	the threshold number for individual validations returning a FALSE result before applying the warn flag.
notify_count	the threshold number for individual validations returning a FALSE result before applying the notify flag.
warn_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.
notify_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.
tbl_name	the name of the local or remote table.
db_type	if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.
creds_file	if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.
initial_sql	when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').
file_path	an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.
col_types	if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.

Value

an agent object.

Examples

```
# Create a simple data frame
# with a column containing data
```

```
# classed as `logical`
df <-
  data.frame(
    a = c(TRUE, FALSE))

# Validate that column `a` in
# the data frame is classed as
# `logical`
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_is_logical(column = a) %>%
  interrogate()

# Determine if this column
# validation has passed by using
# `all_passed()`
all_passed(agent)
#> [1] TRUE
```

`col_is_numeric` *Verify that a column contains numeric values*

Description

Set a verification step where a table column is expected to consist of floating point values.

Usage

```
col_is_numeric(agent, column, brief = NULL, warn_count = 1,
  notify_count = NULL, warn_fraction = NULL, notify_fraction = NULL,
  tbl_name = NULL, db_type = NULL, creds_file = NULL,
  initial_sql = NULL, file_path = NULL, col_types = NULL)
```

Arguments

`agent`	an agent object of class `ptblank_agent`.
`column`	the name of a single table column, multiple columns in the same table, or, a helper function such as `all_cols()`.
`brief`	an optional, text-based description for the validation step.
`warn_count`	the threshold number for individual validations returning a `FALSE` result before applying the `warn` flag.
`notify_count`	the threshold number for individual validations returning a `FALSE` result before applying the `notify` flag.
`warn_fraction`	the threshold fraction for individual validations returning a `FALSE` over all the entire set of individual validations. Beyond this threshold, the `warn` flag will be applied.

notify_fraction
: the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.

tbl_name
: the name of the local or remote table.

db_type
: if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.

creds_file
: if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.

initial_sql
: when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').

file_path
: an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

col_types
: if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.

Value

an agent object.

Examples

```
# Create a simple data frame
# with a column containing data
# classed as `numeric`
df <-
  data.frame(
    a = c(5.1, 2.9),
    stringsAsFactors = FALSE)

# Validate that column `a`
# in the data frame is classed
# as `numeric`
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_is_numeric(
    column = a) %>%
  interrogate()
```

```
# Determine if this column
# validation has passed by using
# `all_passed()`
all_passed(agent)
#> [1] TRUE
```

`col_is_posix` *Verify that a column contains POSIXct dates*

Description

Set a verification step where a table column is expected to consist entirely of POSIXct dates.

Usage

```
col_is_posix(agent, column, brief = NULL, warn_count = 1,
  notify_count = NULL, warn_fraction = NULL, notify_fraction = NULL,
  tbl_name = NULL, db_type = NULL, creds_file = NULL,
  initial_sql = NULL, file_path = NULL, col_types = NULL)
```

Arguments

`agent`	an agent object of class `ptblank_agent`.
`column`	the name of a single table column, multiple columns in the same table, or, a helper function such as `all_cols()`.
`brief`	an optional, text-based description for the validation step.
`warn_count`	the threshold number for individual validations returning a `FALSE` result before applying the `warn` flag.
`notify_count`	the threshold number for individual validations returning a `FALSE` result before applying the `notify` flag.
`warn_fraction`	the threshold fraction for individual validations returning a `FALSE` over all the entire set of individual validations. Beyond this threshold, the `warn` flag will be applied.
`notify_fraction`	the threshold fraction for individual validations returning a `FALSE` over all the entire set of individual validations. Beyond this threshold, the `notify` flag will be applied.
`tbl_name`	the name of the local or remote table.
`db_type`	if the table is located in a database, the type of database is required here. Currently, this can be either `PostgreSQL` or `MySQL`.
`creds_file`	if a connection to a database is required for reaching the table specified in `tbl_name`, then a path to a credentials file can be used to establish that connection. The credentials file is an `RDS` containing a character vector with the following items in the specified order: (1) database name (`dbname`), (2) the `host` name, (3) the `port`, (4) the username (`user`), and (5) the `password`. This file can be easily created using the `create_creds_file()` function.

initial_sql — when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial `SELECT...` statement can be omitted for simple queries (e.g., `WHERE a > 1 AND b = 'one'`).

file_path — an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

col_types — if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: `c` -> character, `i` -> integer, `n` -> number, `d` -> double, `l` -> logical, `D` -> date, `T` -> date time, `t` -> time, `?` -> guess, or `_/-`, which skips the column.

Value

an agent object.

Examples

```
# Create a simple data frame
# with a column containing data
# classed as `POSIXct`
df <-
  data.frame(
    a = as.POSIXct(
      strptime(
        "2011-03-27 01:30:00",
        "%Y-%m-%d %H:%M:%S")))

# Validate that column `a` in
# the data frame is classed as
# `POSIXct`
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_is_posix(column = a) %>%
  interrogate()

# Determine if this column
# validation has passed by
# using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

---

col_vals_between *Verify whether column data are between two values*

Description

Set a verification step where column data should be between two values.

Usage

```
col_vals_between(agent, column, left, right, preconditions = NULL,
  brief = NULL, warn_count = 1, notify_count = NULL,
  warn_fraction = NULL, notify_fraction = NULL, tbl_name = NULL,
  db_type = NULL, creds_file = NULL, initial_sql = NULL,
  file_path = NULL, col_types = NULL)
```

Arguments

agent	an agent object of class ptblank_agent.
column	the column (or a set of columns, provided as a character vector) to which this validation should be applied. Aside from a single column name, column operations can be used to create one or more computed columns (e.g., a + b or a + sum(a)).
left	the lower bound for the range. The validation includes this bound value in addition to values greater than left.
right	the upper bound for the range. The validation includes this bound value in addition to values lower than right.
preconditions	an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return TRUE or FALSE for every row evaluated, where rows evaluated as TRUE are the rows that are retained for the validation step). For example, if a table has columns a, b, and c, and, column a has numerical data, we can write a statement a < 5 that filters all rows in the table where values in column a are less than five.
brief	an optional, text-based description for the validation step.
warn_count	the threshold number for individual validations returning a FALSE result before applying the warn flag.
notify_count	the threshold number for individual validations returning a FALSE result before applying the notify flag.
warn_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.
notify_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.
tbl_name	the name of the local or remote table.
db_type	if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.
creds_file	if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host

name, (3) the `port`, (4) the username (`user`), and (5) the `password`. This file can be easily created using the `create_creds_file()` function.

`initial_sql` when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial `SELECT...` statement can be omitted for simple queries (e.g., `WHERE a > 1 AND b = 'one'`).

`file_path` an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

`col_types` if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: `c` -> character, `i` -> integer, `n` -> number, `d` -> double, `l` -> logical, `D` -> date, `T` -> date time, `t` -> time, `?` -> guess, or `_/-`, which skips the column.

Value

an agent object.

Examples

```
# Create a simple data frame
# with a column of numerical
# values
df <-
  data.frame(
    a = c(5.6, 8.2, 6.3, 7.8, 3.4))

# Validate that values in
# column `a` are all between
# 1 and 9
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_vals_between(
    column = a,
    left = 1,
    right = 9) %>%
  interrogate()

# Determine if this column
# validation has passed by using
# `all_passed()`
all_passed(agent)
#> [1] TRUE
```

col_vals_equal *Verify whether numerical column data are equal to a specified value*

Description

Set a verification step where numeric values in a table column should be equal to a specified value.

Usage

```
col_vals_equal(agent, column, value, preconditions = NULL, brief = NULL,
  warn_count = 1, notify_count = NULL, warn_fraction = NULL,
  notify_fraction = NULL, tbl_name = NULL, db_type = NULL,
  creds_file = NULL, initial_sql = NULL, file_path = NULL,
  col_types = NULL)
```

Arguments

agent	an agent object of class ptblank_agent.
column	the column (or a set of columns, provided as a character vector) to which this validation should be applied. Aside from a single column name, column operations can be used to create one or more computed columns (e.g., a + b or a + sum(a)).
value	a numeric value used to test for equality.
preconditions	an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return TRUE or FALSE for every row evaluated, where rows evaluated as TRUE are the rows that are retained for the validation step). For example, if a table has columns a, b, and c, and, column a has numerical data, we can write a statement a < 5 that filters all rows in the table where values in column a are less than five.
brief	an optional, text-based description for the validation step.
warn_count	the threshold number for individual validations returning a FALSE result before applying the warn flag.
notify_count	the threshold number for individual validations returning a FALSE result before applying the notify flag.
warn_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.
notify_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.
tbl_name	the name of the local or remote table.

db_type — if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.

creds_file — if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.

initial_sql — when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').

file_path — an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

col_types — if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.

Value

an agent object.

Examples

```
# Create a simple data frame
# with 2 columns of numerical values
df <-
  data.frame(
    a = c(1, 1, 1, 2, 2, 2),
    b = c(5, 5, 5, 3, 6, 3))

# Validate that values in column
# `b` are equal to 5 when values
# in column `a` are equal to 1
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_vals_equal(
    column = b,
    value = 5,
    preconditions = a == 1) %>%
  interrogate()

# Determine if these column
# validations have all passed
# by using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

`col_vals_gt` *Verify whether numerical column data are greater than a specified value*

Description

Set a verification step where numeric values in a table column should be greater than a specified value.

Usage

```
col_vals_gt(agent, column, value, preconditions = NULL, brief = NULL,
  warn_count = 1, notify_count = NULL, warn_fraction = NULL,
  notify_fraction = NULL, tbl_name = NULL, db_type = NULL,
  creds_file = NULL, initial_sql = NULL, file_path = NULL,
  col_types = NULL)
```

Arguments

`agent`	an agent object of class `ptblank_agent`.
`column`	the column (or a set of columns, provided as a character vector) to which this validation should be applied. Aside from a single column name, column operations can be used to create one or more computed columns (e.g., `a + b` or `a + sum(a)`).
`value`	a numeric value used for this test. Any column values `> value` are considered passing.
`preconditions`	an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return `TRUE` or `FALSE` for every row evaluated, where rows evaluated as `TRUE` are the rows that are retained for the validation step). For example, if a table has columns `a`, `b`, and `c`, and, column `a` has numerical data, we can write a statement `a < 5` that filters all rows in the table where values in column a are less than five.
`brief`	an optional, text-based description for the validation step.
`warn_count`	the threshold number for individual validations returning a `FALSE` result before applying the `warn` flag.
`notify_count`	the threshold number for individual validations returning a `FALSE` result before applying the `notify` flag.
`warn_fraction`	the threshold fraction for individual validations returning a `FALSE` over all the entire set of individual validations. Beyond this threshold, the `warn` flag will be applied.
`notify_fraction`	the threshold fraction for individual validations returning a `FALSE` over all the entire set of individual validations. Beyond this threshold, the `notify` flag will be applied.

tbl_name	the name of the local or remote table.
db_type	if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.
creds_file	if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.
initial_sql	when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').
file_path	an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.
col_types	if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.

Value

an agent object.

Examples

```
# Create a simple data frame
# with a column of numerical values
df <-
  data.frame(
    a = c(5, 7, 6, 5, 8, 7))

# Validate that values in column
# `a` are always greater than 4
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_vals_gt(
    column = a,
    value = 4) %>%
  interrogate()

# Determine if these column
# validations have all passed
# by using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

col_vals_gte *Verify whether numerical column data are greater than or equal to a specified value*

Description

Set a verification step where numeric values in a table column should be greater than or equal to a specified value.

Usage

```
col_vals_gte(agent, column, value, preconditions = NULL, brief = NULL,
  warn_count = 1, notify_count = NULL, warn_fraction = NULL,
  notify_fraction = NULL, tbl_name = NULL, db_type = NULL,
  creds_file = NULL, initial_sql = NULL, file_path = NULL,
  col_types = NULL)
```

Arguments

agent	an agent object of class ptblank_agent.
column	the column (or a set of columns, provided as a character vector) to which this validation should be applied. Aside from a single column name, column operations can be used to create one or more computed columns (e.g., a + b or a + sum(a)).
value	a numeric value used for this test. Any column values >= value are considered passing.
preconditions	an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return TRUE or FALSE for every row evaluated, where rows evaluated as TRUE are the rows that are retained for the validation step). For example, if a table has columns a, b, and c, and, column a has numerical data, we can write a statement a < 5 that filters all rows in the table where values in column a are less than five.
brief	an optional, text-based description for the validation step.
warn_count	the threshold number for individual validations returning a FALSE result before applying the warn flag.
notify_count	the threshold number for individual validations returning a FALSE result before applying the notify flag.
warn_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.
notify_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.

tbl_name	the name of the local or remote table.
db_type	if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.
creds_file	if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.
initial_sql	when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').
file_path	an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.
col_types	if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.

Value

an agent object.

Examples

```
# Create a simple data frame
# with a column of numerical
# values
df <-
  data.frame(
    a = c(5, 7, 6, 5, 8, 7))

# Validate that values in column
# `a` are always greater than or
# equal to 5
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_vals_gte(
    column = a,
    value = 5) %>%
  interrogate()

# Determine if this column
# validation has passed by using
# `all_passed()`
all_passed(agent)
#> [1] TRUE
```

col_vals_in_set *Verify whether column data are part of a set of values*

Description

Set a verification step where numeric values in a table column should be part of a set of values.

Usage

```
col_vals_in_set(agent, column, set, preconditions = NULL, brief = NULL,
  warn_count = 1, notify_count = NULL, warn_fraction = NULL,
  notify_fraction = NULL, tbl_name = NULL, db_type = NULL,
  creds_file = NULL, initial_sql = NULL, file_path = NULL,
  col_types = NULL)
```

Arguments

agent	an agent object of class ptblank_agent.
column	the column (or a set of columns, provided as a character vector) to which this validation should be applied. Aside from a single column name, column operations can be used to create one or more computed columns (e.g., a + b or a + sum(a)).
set	a vector of numeric or string-based elements, where column values found within this set will be considered as passing.
preconditions	an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return TRUE or FALSE for every row evaluated, where rows evaluated as TRUE are the rows that are retained for the validation step). For example, if a table has columns a, b, and c, and, column a has numerical data, we can write a statement a < 5 that filters all rows in the table where values in column a are less than five.
brief	an optional, text-based description for the validation step.
warn_count	the threshold number for individual validations returning a FALSE result before applying the warn flag.
notify_count	the threshold number for individual validations returning a FALSE result before applying the notify flag.
warn_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.
notify_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.

`tbl_name` the name of the local or remote table.

`db_type` if the table is located in a database, the type of database is required here. Currently, this can be either `PostgreSQL` or `MySQL`.

`creds_file` if a connection to a database is required for reaching the table specified in `tbl_name`, then a path to a credentials file can be used to establish that connection. The credentials file is an `RDS` containing a character vector with the following items in the specified order: (1) database name (`dbname`), (2) the `host` name, (3) the `port`, (4) the username (`user`), and (5) the `password`. This file can be easily created using the `create_creds_file()` function.

`initial_sql` when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial `SELECT...` statement can be omitted for simple queries (e.g., `WHERE a > 1 AND b = 'one'`).

`file_path` an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

`col_types` if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: `c` -> character, `i` -> integer, `n` -> number, `d` -> double, `l` -> logical, `D` -> date, `T` -> date time, `t` -> time, `?` -> guess, or `_/-`, which skips the column.

Value

an agent object.

Examples

```
# Create a simple data frame with
# 2 columns: one with numerical
# values and the other with strings
df <-
  data.frame(
    a = c(1, 2, 3, 4),
    b = c("one", "two", "three", "four"),
    stringsAsFactors = FALSE)

# Validate that all numerical values
# in column `a` belong to a numerical
# set, and, create an analogous
# validation check for column `b` with
# a set of string values
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_vals_in_set(
    column = a,
    set = 1:4) %>%
  col_vals_in_set(
    column = b,
```

```
    set = c("one", "two",
            "three", "four")) %>%
  interrogate()

# Determine if these column
# validations have all passed
# by using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

---

col_vals_lt *Verify whether numerical column data are less than a specified value*

Description

Set a verification step where numeric values in a table column should be less than a specified value.

Usage

```
col_vals_lt(agent, column, value, preconditions = NULL, brief = NULL,
  warn_count = 1, notify_count = NULL, warn_fraction = NULL,
  notify_fraction = NULL, tbl_name = NULL, db_type = NULL,
  creds_file = NULL, initial_sql = NULL, file_path = NULL,
  col_types = NULL)
```

Arguments

agent	an agent object of class ptblank_agent.
column	the column (or a set of columns, provided as a character vector) to which this validation should be applied. Aside from a single column name, column operations can be used to create one or more computed columns (e.g., a + b or a + sum(a)).
value	a numeric value used for this test. Any column values < value are considered passing.
preconditions	an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return TRUE or FALSE for every row evaluated, where rows evaluated as TRUE are the rows that are retained for the validation step). For example, if a table has columns a, b, and c, and, column a has numerical data, we can write a statement a < 5 that filters all rows in the table where values in column a are less than five.
brief	an optional, text-based description for the validation step.
warn_count	the threshold number for individual validations returning a FALSE result before applying the warn flag.

notify_count — the threshold number for individual validations returning a FALSE result before applying the notify flag.

warn_fraction — the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.

notify_fraction — the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.

tbl_name — the name of the local or remote table.

db_type — if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.

creds_file — if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.

initial_sql — when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').

file_path — an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

col_types — if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.

Value

an agent object.

Examples

```
# Create a simple data frame
# with a column of numerical
# values
df <-
  data.frame(
    a = c(5, 4, 3, 5, 1, 2))

# Validate that values in
# column `a` are always less
# than 6
agent <-
  create_agent() %>%
```

```
  focus_on(tbl_name = "df") %>%
  col_vals_lt(
    column = a,
    value = 6) %>%
  interrogate()

# Determine if this column
# validation has passed by using
# `all_passed()`
all_passed(agent)
#> [1] TRUE
```

`col_vals_lte` *Verify whether numerical column data are less than or equal to a specified value*

Description

Set a verification step where numeric values in a table column should be less than or equal to a specified value.

Usage

```
col_vals_lte(agent, column, value, preconditions = NULL, brief = NULL,
  warn_count = 1, notify_count = NULL, warn_fraction = NULL,
  notify_fraction = NULL, tbl_name = NULL, db_type = NULL,
  creds_file = NULL, initial_sql = NULL, file_path = NULL,
  col_types = NULL)
```

Arguments

`agent`	an agent object of class `ptblank_agent`.
`column`	the column (or a set of columns, provided as a character vector) to which this validation should be applied. Aside from a single column name, column operations can be used to create one or more computed columns (e.g., `a + b` or `a + sum(a)`).
`value`	a numeric value used for this test. Any column values `<= value` are considered passing.
`preconditions`	an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return `TRUE` or `FALSE` for every row evaluated, where rows evaluated as `TRUE` are the rows that are retained for the validation step). For example, if a table has columns `a`, `b`, and `c`, and, column `a` has numerical data, we can write a statement `a < 5` that filters all rows in the table where values in column a are less than five.
`brief`	an optional, text-based description for the validation step.

warn_count — the threshold number for individual validations returning a FALSE result before applying the warn flag.

notify_count — the threshold number for individual validations returning a FALSE result before applying the notify flag.

warn_fraction — the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.

notify_fraction — the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.

tbl_name — the name of the local or remote table.

db_type — if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.

creds_file — if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.

initial_sql — when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').

file_path — an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

col_types — if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.

Value

an agent object.

Examples

```
# Create a simple data frame
# with a column of numerical
# values
df <-
  data.frame(
    a = c(5, 4, 3, 5, 1, 2),
    b = c(3, 2, 4, 3, 5, 6))

# Validate that the sum of
```

```
# values across columns `a`
# and `b` are always less
# than or equal to 10
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_vals_lte(
    column = a + b,
    value = 10) %>%
  interrogate()

# Determine if this column
# validation has passed by using
# `all_passed()`
all_passed(agent)
#> [1] TRUE
```

`col_vals_not_between` *Verify that column data are not between two values*

Description

Set a verification step where column data should not be between two values.

Usage

```
col_vals_not_between(agent, column, left, right, preconditions = NULL,
  brief = NULL, warn_count = 1, notify_count = NULL,
  warn_fraction = NULL, notify_fraction = NULL, tbl_name = NULL,
  db_type = NULL, creds_file = NULL, initial_sql = NULL,
  file_path = NULL, col_types = NULL)
```

Arguments

`agent`	an agent object of class `ptblank_agent`.
`column`	the column (or a set of columns, provided as a character vector) to which this validation should be applied. Aside from a single column name, column operations can be used to create one or more computed columns (e.g., `a + b` or `a + sum(a)`).
`left`	the lower bound for the range. The validation includes this bound value in addition to values greater than `left`. Any values `>= left` and `<= right` will be considered as failing.
`right`	the upper bound for the range. The validation includes this bound value in addition to values lower than `right`. Any values `<= right` and `>= left` will be considered as failing.

preconditions — an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return TRUE or FALSE for every row evaluated, where rows evaluated as TRUE are the rows that are retained for the validation step). For example, if a table has columns a, b, and c, and, column a has numerical data, we can write a statement a < 5 that filters all rows in the table where values in column a are less than five.

brief — an optional, text-based description for the validation step.

warn_count — the threshold number for individual validations returning a FALSE result before applying the warn flag.

notify_count — the threshold number for individual validations returning a FALSE result before applying the notify flag.

warn_fraction — the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.

notify_fraction — the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.

tbl_name — the name of the local or remote table.

db_type — if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.

creds_file — if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.

initial_sql — when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').

file_path — an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

col_types — if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.

Value

an agent object.

Examples

```
# Create a simple data frame
# with a column a numerical values
df <-
  data.frame(
    a = c(5.6, 8.2, 6.3, 7.8, 3.4))

# Validate that none of the values
# in column `a` are between 9 and 10,
# or, between 0 and 2
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_vals_not_between(
    column = a,
    left = 9,
    right = 10) %>%
  col_vals_not_between(
    column = a,
    left = 0,
    right = 2) %>%
  interrogate()

# Determine if these column
# validations have all passed by
# using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

`col_vals_not_equal` *Verify whether numerical column data are not equal to a specified value*

Description

Set a verification step where numeric values in a table column should not be equal to a specified value.

Usage

```
col_vals_not_equal(agent, column, value, preconditions = NULL, brief = NULL,
  warn_count = 1, notify_count = NULL, warn_fraction = NULL,
  notify_fraction = NULL, tbl_name = NULL, db_type = NULL,
  creds_file = NULL, initial_sql = NULL, file_path = NULL,
  col_types = NULL)
```

Arguments

agent	an agent object of class ptblank_agent.
column	the column (or a set of columns, provided as a character vector) to which this validation should be applied. Aside from a single column name, column operations can be used to create one or more computed columns (e.g., a + b or a + sum(a)).
value	a numeric value used to test for non-equality.
preconditions	an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return TRUE or FALSE for every row evaluated, where rows evaluated as TRUE are the rows that are retained for the validation step). For example, if a table has columns a, b, and c, and, column a has numerical data, we can write a statement a < 5 that filters all rows in the table where values in column a are less than five.
brief	an optional, text-based description for the validation step.
warn_count	the threshold number for individual validations returning a FALSE result before applying the warn flag.
notify_count	the threshold number for individual validations returning a FALSE result before applying the notify flag.
warn_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.
notify_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.
tbl_name	the name of the local or remote table.
db_type	if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.
creds_file	if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.
initial_sql	when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').
file_path	an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

col_types — if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.

Value

an agent object.

Examples

```
# Create a simple data frame
# with 2 columns of numerical values
df <-
  data.frame(
    a = c(1, 1, 1, 2, 2, 2),
    b = c(5, 5, 5, 3, 6, 3))

# Validate that values in
# column `b` are not equal to 5
# when values in column `a`
# are equal to 2
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_vals_not_equal(
    column = b,
    value = 5,
    preconditions = a == 2) %>%
  interrogate()

# Determine if this column
# validation has passed by using
# `all_passed()`
all_passed(agent)
#> [1] TRUE
```

---

col_vals_not_in_set *Verify that column data are not part of a set of values*

Description

Set a verification step where numeric values in a table column should be part of a set of values.

Usage

```
col_vals_not_in_set(agent, column, set, preconditions = NULL, brief = NULL,
  warn_count = 1, notify_count = NULL, warn_fraction = NULL,
```

```
  notify_fraction = NULL, tbl_name = NULL, db_type = NULL,
  creds_file = NULL, initial_sql = NULL, file_path = NULL,
  col_types = NULL)
```

Arguments

agent	an agent object of class ptblank_agent.
column	the column (or a set of columns, provided as a character vector) to which this validation should be applied. Aside from a single column name, column operations can be used to create one or more computed columns (e.g., a + b or a + sum(a)).
set	a vector of numeric or string-based elements, where column values found within this set will be considered as failing.
preconditions	an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return TRUE or FALSE for every row evaluated, where rows evaluated as TRUE are the rows that are retained for the validation step). For example, if a table has columns a, b, and c, and, column a has numerical data, we can write a statement a < 5 that filters all rows in the table where values in column a are less than five.
brief	an optional, text-based description for the validation step.
warn_count	the threshold number for individual validations returning a FALSE result before applying the warn flag.
notify_count	the threshold number for individual validations returning a FALSE result before applying the notify flag.
warn_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.
notify_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.
tbl_name	the name of the local or remote table.
db_type	if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.
creds_file	if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.
initial_sql	when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').

`file_path` an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

`col_types` if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: `c` -> character, `i` -> integer, `n` -> number, `d` -> double, `l` -> logical, `D` -> date, `T` -> date time, `t` -> time, `?` -> guess, or `_/-`, which skips the column.

Value

an agent object.

Examples

```
# Create a simple data frame with 2 columns: one
# with numerical values and the other with strings
df <-
  data.frame(
    a = c(1, 2, 3, 4),
    b = c("one", "two", "three", "four"),
    stringsAsFactors = FALSE)

# Validate that all numerical
# values in column `a` do not
# belong to a specified numerical
# set, and, create an analogous
# validation check for column `b`
# within a set of string values
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_vals_not_in_set(
    column = a,
    set = 7:10) %>%
  col_vals_not_in_set(
    column = b,
    set = c("seven", "eight",
            "nine", "ten")) %>%
  interrogate()

# Determine if these column
# validations have all passed
# by using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

`col_vals_not_null` *Verify whether all column values are not NULL*

Description

Set a verification step where no values in a table column are expected to be NULL.

Usage

```
col_vals_not_null(agent, column, preconditions = NULL, brief = NULL,
  warn_count = 1, notify_count = NULL, warn_fraction = NULL,
  notify_fraction = NULL, tbl_name = NULL, db_type = NULL,
  creds_file = NULL, initial_sql = NULL, file_path = NULL,
  col_types = NULL)
```

Arguments

agent	an agent object of class ptblank_agent.
column	the column (or a set of columns, provided as a character vector) to which this validation should be applied. Aside from a single column name, column operations can be used to create one or more computed columns (e.g., a + b or a + sum(a)).
preconditions	an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return TRUE or FALSE for every row evaluated, where rows evaluated as TRUE are the rows that are retained for the validation step). For example, if a table has columns a, b, and c, and, column a has numerical data, we can write a statement a < 5 that filters all rows in the table where values in column a are less than five.
brief	an optional, text-based description for the validation step.
warn_count	the threshold number for individual validations returning a FALSE result before applying the warn flag.
notify_count	the threshold number for individual validations returning a FALSE result before applying the notify flag.
warn_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.
notify_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.
tbl_name	the name of the local or remote table.
db_type	if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.
creds_file	if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host

name, (3) the `port`, (4) the username (`user`), and (5) the `password`. This file can be easily created using the `create_creds_file()` function.

`initial_sql` when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial `SELECT...` statement can be omitted for simple queries (e.g., `WHERE a > 1 AND b = 'one'`).

`file_path` an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

`col_types` if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: `c` -> character, `i` -> integer, `n` -> number, `d` -> double, `l` -> logical, `D` -> date, `T` -> date time, `t` -> time, `?` -> guess, or `_/-`, which skips the column.

Value

an agent object.

Examples

```
# Create a simple data frame with
# 2 columns: one with numerical
# values and the other with strings
df <-
  data.frame(
    a = c(1, 2, NA, NA),
    b = c(2, 2, 5, 5),
    stringsAsFactors = FALSE)

# Validate that all values in
# column `a` are not NULL when
# values in column `b` are equal
# to 2
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_vals_not_null(
    column = a,
    preconditions = b == 2) %>%
  interrogate()

# Determine if these column
# validations have all passed
# by using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

col_vals_null *Verify whether all column values are NULL*

Description

Set a verification step where all values in a table column are expected to be NULL.

Usage

```
col_vals_null(agent, column, preconditions = NULL, brief = NULL,
  warn_count = 1, notify_count = NULL, warn_fraction = NULL,
  notify_fraction = NULL, tbl_name = NULL, db_type = NULL,
  creds_file = NULL, initial_sql = NULL, file_path = NULL,
  col_types = NULL)
```

Arguments

agent	an agent object of class ptblank_agent.
column	the column (or a set of columns, provided as a character vector) to which this validation should be applied. Aside from a single column name, column operations can be used to create one or more computed columns (e.g., a + b or a + sum(a)).
preconditions	an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return TRUE or FALSE for every row evaluated, where rows evaluated as TRUE are the rows that are retained for the validation step). For example, if a table has columns a, b, and c, and, column a has numerical data, we can write a statement a < 5 that filters all rows in the table where values in column a are less than five.
brief	an optional, text-based description for the validation step.
warn_count	the threshold number for individual validations returning a FALSE result before applying the warn flag.
notify_count	the threshold number for individual validations returning a FALSE result before applying the notify flag.
warn_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.
notify_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.
tbl_name	the name of the local or remote table.
db_type	if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.

creds_file — if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.

initial_sql — when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').

file_path — an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

col_types — if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.

Value

an agent object.

Examples

```
# Create a simple data frame with
# 2 columns: one with numerical
# values and the other with strings
df <-
  data.frame(
    a = c(1, 2, NA, NA),
    b = c(2, 2, 5, 5),
    stringsAsFactors = FALSE)

# Validate that all values in
# column `a` are NULL when
# values in column `b` are
# equal to 5
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_vals_null(
    column = a,
    preconditions = b == 5) %>%
  interrogate()

# Determine if these column
# validations have all passed
# by using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

`col_vals_regex` *Verify whether string column data corresponds to a regex matching expression*

Description

Set a verification step where string column data should correspond to a regex matching expression.

Usage

```
col_vals_regex(agent, column, regex, preconditions = NULL, brief = NULL,
  warn_count = 1, notify_count = NULL, warn_fraction = NULL,
  notify_fraction = NULL, tbl_name = NULL, db_type = NULL,
  creds_file = NULL, initial_sql = NULL, file_path = NULL,
  col_types = NULL)
```

Arguments

`agent`	an agent object of class `ptblank_agent`.
`column`	the column (or a set of columns, provided as a character vector) to which this validation should be applied. Aside from a single column name, column operations can be used to create one or more computed columns (e.g., `a + b` or `a + sum(a)`).
`regex`	a regex pattern to test for matching strings.
`preconditions`	an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return `TRUE` or `FALSE` for every row evaluated, where rows evaluated as `TRUE` are the rows that are retained for the validation step). For example, if a table has columns `a`, `b`, and `c`, and, column `a` has numerical data, we can write a statement `a < 5` that filters all rows in the table where values in column a are less than five.
`brief`	an optional, text-based description for the validation step.
`warn_count`	the threshold number for individual validations returning a `FALSE` result before applying the `warn` flag.
`notify_count`	the threshold number for individual validations returning a `FALSE` result before applying the `notify` flag.
`warn_fraction`	the threshold fraction for individual validations returning a `FALSE` over all the entire set of individual validations. Beyond this threshold, the `warn` flag will be applied.
`notify_fraction`	the threshold fraction for individual validations returning a `FALSE` over all the entire set of individual validations. Beyond this threshold, the `notify` flag will be applied.

tbl_name	the name of the local or remote table.
db_type	if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.
creds_file	if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.
initial_sql	when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').
file_path	an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.
col_types	if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.

Value

an agent object.

Examples

```
# Create a simple data frame with a column
# containing strings
df <-
  data.frame(
    a = c("s_0131", "s_0231",
          "s_1389", "s_2300"),
    stringsAsFactors = FALSE)

# Validate that all string values in
# column `a` match a regex statement
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_vals_regex(
    column = a,
    regex = "^s_[0-9]{4}$") %>%
  interrogate()

# Determine if these column
# validations have all passed
# by using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

create_agent *Create a pointblank agent object*

Description

Creates an agent object.

Usage

```
create_agent(validation_name = NULL)
```

Arguments

validation_name
: an optional name for the validation pipeline that the agent will eventually carry out during the interrogation process. If no value is provided, a name will be generated based on the current system time.

Value

an agent object.

Examples

```
# Create a simple data frame
# with a column of numerical values
df <-
  data.frame(
    a = c(5, 7, 6, 5, 8, 7))

# Create a pointblank `agent` object
agent <- create_agent()

# Then, as with any `ptblank_agent`
# object, we can focus on a table,
# add validation steps, and then
# eventually use `interrogate()`
# to perform the validations;
# here, in a single validation
# step, we expect that values in
# column `a` are always greater
# than 4
agent <-
  agent %>%
  focus_on(tbl_name = "df") %>%
  col_vals_gt(
    column = a,
    value = 4) %>%
  interrogate()
```

```
# A summary can be produced using
# `get_interrogation_summary()`; we
# we will just obtain the first
# 7 columns of its output
(agent %>%
  get_interrogation_summary())[, 1:7]
#> # A tibble: 1 x 7
#>   tbl_name  db_type assertion_type column value regex all_passed
#>      <chr>    <chr>          <chr>  <chr> <dbl> <chr>      <lgl>
#> 1       df local_df    col_vals_gt      a     4  <NA>       TRUE
```

`create_creds_file` *Create a file with DB access credentials*

Description

Creates a file containing access credentials for a database.

Usage

```
create_creds_file(file, dbname, host, port, user, password)
```

Arguments

`file`	a file path for the credentials file to be stored on disk.
`dbname`	the database name.
`host`	the `host` name.
`port`	the port number.
`user`	the username for the database
`password`	the password associated with the `user`.

Examples

```
## Not run:
# Create a credentials file for access to
# remote databases (where the tables to be
# validated reside); place in the user's
# home directory
create_creds_file(
  file = "~/.pb_credentials",
  dbname = "********",
  host = "**************",
  port = ***,
  user = "********",
  password = "*****************")

## End(Not run)
```

create_email_creds_file *Create a file with email access credentials*

Description

Creates a file with access credentials for the purpose of automatically emailing notification messages.

Usage

```
create_email_creds_file(file, sender, host, port, user, password,
  use_ssl = TRUE, authenticate = TRUE)
```

Arguments

file	a file path for the credentials file to be stored on disk.
sender	the sender name.
host	the host name.
port	the port number.
user	the username for the email account.
password	the password associated with the user's email address.
use_ssl	an option as to whether to use SSL; supply a TRUE or FALSE value (TRUE is the default value).
authenticate	an option as to whether to authenticate; supply a TRUE or FALSE value (TRUE is the default value).

Examples

```
## Not run:
# Create a credentials file for automatic
# email notifications; place in the user's
# home directory
create_email_creds_file(
  file = "~/.pb_notify",
  sender = "point@blank.org",
  host = "smtp.blank.org",
  port = 465,
  user = "point@blank.org",
  password = "************")

## End(Not run)
```

db_creds_env_vars *Bind environment variable names for database access*

Description

Associates environment variables as credentials for a database. Used to generate a list of environment variable names, which is used as an input value for the `db_creds_env_vars` argument of the `focus_on()` function.

Usage

```
db_creds_env_vars(dbname, host, port, user, password)
```

Arguments

`dbname`	the name of the environment variable storing the database name.
`host`	the name of the environment variable storing the `host` name.
`port`	the name of the environment variable storing the port number.
`user`	the name of the environment variable storing a username for the database.
`password`	the name of the environment variable storing the password associated with the `user`.

focus_on *Place certain access details more to the fore*

Description

Allows for a change of the focus on the table name, database type, and the location of the credentials file.

Usage

```
focus_on(agent, tbl_name = NULL, file_name = NULL, col_types = NULL,
  db_type = NULL, initial_sql = NULL, brief = NULL, creds_file = NULL,
  db_creds_env_vars = NULL)
```

Arguments

`agent`	an agent object of class `ptblank_agent`.
`tbl_name`	the name of the local or remote table.
`file_name`	the name of a file to be loaded as a table. Valid types are CSV and TSV files.

col_types
: if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: `c` -> character, `i` -> integer, `n` -> number, `d` -> double, `l` -> logical, `D` -> date, `T` -> date time, `t` -> time, `?` -> guess, or `_/-`, which skips the column.

db_type
: if the table is located in a database, the type of database is required here. Currently, this can be either `PostgreSQL` or `MySQL`.

initial_sql
: when accessing a table in a database (MySQL and PostgreSQL), this provides an option to provide an initial SQL query that is applied to the table before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial `SELECT...` statement can be omitted for simple queries that filter the table in focus (e.g., `WHERE a > 1 AND b = 'one'`).

brief
: an optional, text-based description for the new focus.

creds_file
: if a connection to a database is required for reaching the table specified in `tbl_name`, then a path to a credentials file can be used to establish that connection. The credentials file is an `RDS` containing a character vector with the following items in the specified order: (1) database name (`dbname`), (2) the `host` name, (3) the `port`, (4) the username (`user`), and (5) the `password`. This file can be easily created using the `create_creds_file()` function.

db_creds_env_vars
: if a connection to a database is required for reaching the table specified in `tbl_name`, then a set of environment variables can be used to establish that connection. Separate environment variables with the following items should be available: (1) database name (`dbname`), (2) the `host` name, (3) the `port`, (4) the username (`user`), and (5) the `password`. To pass the names of the environment variables to the `agent` object, one can use the `db_creds_env_vars()` function directly.

Value

an agent object.

Examples

```
# Create a simple data frame with a column
# of numerical values
df <-
  data.frame(
    a = c(5, 4, 3, 5, 1, 2))

# Validate that values in column `a` are
# always less than 6; this requires the
# use of `create_agent()` (to begin the
# validation process) and `focus_on()`
# to point to the table that will undergo
# validation in subsequent steps
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
```

```
  col_vals_lt(
    column = a,
    value = 6) %>%
  interrogate()

# Determine if this column validation has
# passed by using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

`get_html_summary` *Create an HTML summary file for the interrogation*

Description

Get the essential information from an agent object after an interrogation is complete and then generates an HTML file for a visual summary.

Usage

```
get_html_summary(agent, output_file = NULL, output_dir = NULL,
  intro_text = NULL, pre_results_text = NULL, footer_text = NULL)
```

Arguments

`agent`	an agent object of class `ptblank_agent`.
`output_file`	an optional filename to use for the output HTML file. If not provided, the filename `validation_report.html` will be used.
`output_dir`	an optional path to place the output HTML file and the subfolder of CSV data (if row sample data has been collected). If the path does not exist, the directory will be created.
`intro_text`	HTML text to be placed at the top of the summary. Can be provided as plaintext or as markdown text.
`pre_results_text`	HTML text to be placed just before the tables of validation results. Can be provided as plaintext or as markdown text.
`footer_text`	HTML text to be placed in the footer of the summary. Can be provided as plaintext or as markdown text.

Value

an agent object.

get_interrogation_summary *Get a simple summary of the interrogation*

Description

Gets the essential information from an agent object after an interrogation is complete.

Usage

```
get_interrogation_summary(agent)
```

Arguments

agent	an agent object of class ptblank_agent.

Value

an agent object.

get_row_sample_data *Get non-passing sample rows from a validation step*

Description

Get row data that didn't pass a validation step. The amount of row data available depends on both the fraction of rows that didn't pass a validation step and the level of sampling or explicit collection from that set of rows (this is defined within the interrogate() call).

Usage

```
get_row_sample_data(agent, step)
```

Arguments

agent	an agent object of class ptblank_agent. It should have had interrogate() called on it, such that the validation steps were carried out and any sample rows from non-passing validations could potentially be available in the object.
step	the validation step number, which is assigned to each validation step in the order of definition. To determine which validation steps produced sample row data, one can use the get_row_sample_info() function. The data frame output provides the step number and the number of rows in the sample.

Examples

```
# Set a seed
set.seed(23)

# Create a simple data frame with a
# column of numerical values
df <-
  data.frame(
    a = rnorm(
      n = 100,
      mean = 5,
      sd = 2))

# Create 2 simple validation steps
# that test whether values within
# column `a`
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_vals_between(
    column = a,
    left = 4,
    right = 6) %>%
  col_vals_lte(
    column = a,
    value = 10) %>%
  interrogate(
    get_problem_rows = TRUE,
    get_first_n = 10)

# Find out which validation steps
# contain sample row data
get_row_sample_info(agent)
#>   step   assertion_type n_failed rows_in_sample
#> 1    1 col_vals_between       65             10

# Get row sample data for those rows
# in `df` that did not pass the first
# validation step (`col_vals_between`);
# the leading column `pb_step_` is
# applied to provide context on the
# validation step for which these rows
# failed to pass
agent %>%
  get_row_sample_data(step = 1)
#> # A tibble: 10 x 2
#>    pb_step_        a
#>       <int>    <dbl>
#>  1        1 6.826534
#>  2        1 8.586776
#>  3        1 6.993210
#>  4        1 7.214981
```

```
#>  5        1 7.038411
#>  6        1 8.151559
#>  7        1 2.906929
#>  8        1 2.567247
#>  9        1 3.959643
#> 10        1 3.801374
```

`get_row_sample_info` *Get information on sample rows from non-passing validations*

Description

Get information on which validation steps produced sample rows from non-passing validations.

Usage

```
get_row_sample_info(agent)
```

Arguments

`agent`	an agent object of class `ptblank_agent`. It should have had `interrogate()` called on it, such that the validation steps were carried out and any sample rows from non-passing validations could potentially be available in the object.

Examples

```
# Set a seed
set.seed(23)

# Create a simple data frame with a
# column of numerical values
df <-
  data.frame(
    a = rnorm(
      n = 100,
      mean = 5,
      sd = 2))

# Create 2 simple validation steps
# that test whether values within
# column `a`
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_vals_between(
    column = a,
    left = 4,
    right = 6) %>%
  col_vals_lte(
    column = a,
```

```
    value = 10) %>%
  interrogate(
    get_problem_rows = TRUE,
    get_first_n = 10)

# Find out which validation steps
# contain sample row data
get_row_sample_info(agent)
#>   step   assertion_type n_failed rows_in_sample
#> 1    1 col_vals_between       65             10
```

`get_validation_name` *Get the pointblank validation name*

Description

Gets the name of a pointblank validation from an agent object. This name is optionally assigned during the `create_agent()` call using the `validation_name` argument.

Usage

```
get_validation_name(agent)
```

Arguments

`agent`	an agent object of class `ptblank_agent`.

Value

a character object with the agent name.

`interrogate` *Given an agent that is fully loaded with tasks, perform an interrogation*

Description

The agent has all the information on what to do, so now all interrogations can proceed efficiently, and, according to plan.

Usage

```
interrogate(agent, get_problem_rows = TRUE, get_first_n = NULL,
  sample_n = NULL, sample_frac = NULL, sample_limit = 5000)
```

Arguments

agent	an agent object of class ptblank_agent.
get_problem_rows	an option to collect rows that didn't pass a particular validation step. The default is TRUE and further options allow for fine control of how these rows are collected.
get_first_n	if the option to collect non-passing rows is chosen, there is the option here to collect the first n rows here. Supply the number of rows to extract from the top of the non-passing rows table (the ordering of data from the original table is retained).
sample_n	if the option to collect non-passing rows is chosen, this option allows for the sampling of n rows. Supply the number of rows to sample from the non-passing rows table. If n is greater than the number of non-passing rows, then all the rows will be returned.
sample_frac	if the option to collect non-passing rows is chosen, this option allows for the sampling of a fraction of those rows. Provide a number in the range of 0 and 1. The number of rows to return may be extremely large (and this is especially when querying remote databases), however, the sample_limit option will apply a hard limit to the returned rows.
sample_limit	a value that limits the possible number of rows returned when sampling non-passing rows using the sample_frac option.

Value

an agent object.

Examples

```
# Create 2 simple data frames
# with 2 columns of numerical
# values in each
df_1 <-
  data.frame(
    a = c(5, 7, 6, 5, 8, 7),
    b = c(7, 1, 0, 0, 0, 3))

df_2 <-
  data.frame(
    c = c(8, 8, 8, 6, 1, 3),
    d = c(9, 8, 7, 2, 3, 3))

# Validate that values in column
# `a` from `df_1` are always >= 5,
# and also validate that, in `df_2`,
# values in `c` are always == 8
# when values in `d` are >= 5
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df_1") %>%
  col_vals_gte(
```

```
    column = a,
    value = 5) %>%
  focus_on(tbl_name = "df_2") %>%
  col_vals_equal(
    column = c,
    value = 8,
    preconditions = d >= 5) %>%
  interrogate()

# Get a basic summary with
# `get_interrogation_summary()`
get_interrogation_summary(agent)[, 1:7]
#> # A tibble: 2 x 7
#>   tbl_name  db_type assertion_type column value regex all_passed
#>      <chr>    <chr>          <chr>  <chr> <dbl> <chr>      <lgl>
#> 1     df_1 local_df   col_vals_gte      a     5  <NA>       TRUE
#> 2     df_2 local_df col_vals_equal      c     8  <NA>       TRUE
```

---

is_ptblank_agent *Is the object a pointblank agent?*

Description

Determines whether the object is actually a pointbank agent object (i.e., of type `ptblank_agent`).

Usage

```
is_ptblank_agent(object)
```

Arguments

`object`	the object to test for whether it is of class `ptblank_agent`.

Value

an logical value.

---

rows_not_duplicated *Verify that row data are not duplicated*

Description

Set a verification step where row data should contain no duplicates.

Usage

```
rows_not_duplicated(agent, cols = NULL, preconditions = NULL,
  brief = NULL, warn_count = 1, notify_count = NULL,
  warn_fraction = NULL, notify_fraction = NULL, tbl_name = NULL,
  db_type = NULL, creds_file = NULL, initial_sql = NULL,
  file_path = NULL, col_types = NULL)
```

Arguments

agent	an agent object of class ptblank_agent.
cols	an optional grouping of columns to check for duplication. If not provided, the validation checks for duplicate records using data across all columns.
preconditions	an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return TRUE or FALSE for every row evaluated, where rows evaluated as TRUE are the rows that are retained for the validation step). For example, if a table has columns a, b, and c, and, column a has numerical data, we can write a statement a < 5 that filters all rows in the table where values in column a are less than five.
brief	an optional, text-based description for the validation step.
warn_count	the threshold number for individual validations returning a FALSE result before applying the warn flag.
notify_count	the threshold number for individual validations returning a FALSE result before applying the notify flag.
warn_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.
notify_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.
tbl_name	the name of the local or remote table.
db_type	if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.
creds_file	if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.
initial_sql	when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').

file_path	an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.
col_types	if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.

Value

an agent object.

Examples

```
# Validate that column `a` exists in
# the `small_table` CSV file; do this
# by creating an agent, focussing on
# that table, creating a `col_exists()`
# step, and then interrogating the table
agent <-
  create_agent() %>%
  focus_on(
    file_name =
      system.file(
        "extdata", "small_table.csv",
        package = "pointblank"),
    col_types = "TDicidlc") %>%
  rows_not_duplicated(
    cols = a & b) %>%
  interrogate()

# Determine if these column
# validations have all passed
# by using `all_passed()`
all_passed(agent)
#> [1] FALSE
```

---

run_validation_files *Process multiple validation files*

Description

Run multiple validation scripts in a single function call. This is useful when validation scripts are developed over time since this function could be simply pointed to a directory containing validation script files.

Usage

```
run_validation_files(input_dir = NULL, file_pattern = NULL,
  input_files = NULL)
```

Arguments

input_dir	a path to where the validation scripts reside. Any files that match the pattern specified in file_pattern are included in the file list for processing.
file_pattern	a file pattern to distinguish which files are to be included in the file list for processing. If no pattern is provided, the default pattern will caputure any R script files in input_dir.
input_files	a vector of input files that are to be processed. If filenames are provided here, any values provided to input_dir and file_pattern are ignored.

---

set_email_prefs *Set email credentials and enable email reporting*

Description

Grants email credentials to a pointblank agent object and provides an opportunity to set email reporting options. This function is to be used in a pointblank pipeline any time before an interrogate() call.

Usage

```
set_email_prefs(agent, notify_active = FALSE, email_recipients = NULL,
  creds_file = NULL)
```

Arguments

agent	an agent object of class ptblank_agent.
notify_active	an option to enable notification emails whenever any of the validation steps in the agent object have triggered a notify status.
email_recipients	an optional vector of email addresses to which notification emails should be sent.
creds_file	an optional path to an email credentials file. Such a file can be generated using the create_email_creds_file() function. notify status.

Value

an agent object.

Examples

```
## Not run:
# Generate an email credentials
# file using the function
# `create_email_creds_file()`
create_email_creds_file(
  file = "~/.pb_email",
```

```
  sender = "point@blank.org",
  host = "smtp.blank.org",
  port = 465,
  user = "point@blank.org",
  password = "************")

# Create a simple data frame
# with a column of numerical values
df <-
  data.frame(
    a = c(5, 7, 6, 5, 8, 7))

# Create a pointblank `agent`,
# set up the email notification
# preferences, and conduct a
# simple validation; because
# `notify_count` (in the step
# where `col_vals_lt()` is called)
# has a value of `1` the
# `email_recipients` will be
# notified when there are one or
# more non-passing validations (in
# this case, non-passing rows)
agent <-
  create_agent() %>%
  set_email_prefs(
    notify_active = TRUE,
    email_recipients =
      c("a@b.net", "c@d.com"),
    creds_file = "~/.pb_email") %>%
  focus_on(tbl_name = "df") %>%
  col_vals_lt(
    column = a,
    value = 6,
    notify_count = 1) %>%
  interrogate()

## End(Not run)
```

`set_slack_prefs` *Set Slack credentials and enable Slack notifications*

Description

Grants Slack credentials to a pointblank agent object and provides an opportunity to set Slack reporting options. This function is to be used in a pointblank pipeline any time before an `interrogate()` call.

Usage

```
set_slack_prefs(agent, notify_active = FALSE, slack_webhook_url,
  slack_channel, slack_username, slack_author_name = NULL,
  slack_title = NULL, slack_report_url = NULL,
  slack_footer_thumb_url = NULL, slack_footer_text = NULL)
```

Arguments

agent	an agent object of class ptblank_agent.
notify_active	an option to enable Slack notifications whenever any of the validation steps in the agent object have triggered a notify status.
slack_webhook_url	the URL that is the endpoint for the API POST request.
slack_channel	the slack channel to which the notification message will be posted.
slack_username	the custom username associated with the webhook integration.
slack_author_name	an optional author name for the notification. If not provided, then pointblank will be used as fallback text.
slack_title	the title text for the notification. If not provided, the name of the validation will be used.
slack_report_url	an optional URL for a validation report that is associated with the notification message. The link is embedded in the slack_title text.
slack_footer_thumb_url	an optional URL that is associated with the thumbnail image in the notification footer.
slack_footer_text	an optional snippet of text that will be part of the notification footer. notify status.

Value

an agent object.

Examples

```
## Not run:
# Create a simple data frame
# with a column of numerical values
df <-
  data.frame(
    a = c(5, 7, 6, 5, 8, 7))

# Create a pointblank `agent`,
# set up the Slack notification
# preferences, and conduct a
# simple validation; because
# `notify_count` (in the step
```

```
# where `col_vals_lt()` is called)
# has a value of `1` the
# `slack_channel` will be
# notified when there are one or
# more non-passing validations (in
# this case, non-passing rows)
agent <-
  create_agent() %>%
  set_slack_prefs(
    notify_active = TRUE,
    slack_webhook_url =
      "https://hooks.slack.com/services/XXXXX/XXXXX/XXXXX",
    slack_channel =
      "#table-validation",
    slack_username = "table_validator",
    slack_report_url =
      "https:://my.company.com/reports/df_validation") %>%
  focus_on(tbl_name = "df") %>%
  col_vals_lt(
    column = a,
    value = 6,
    notify_count = 1) %>%
  interrogate()

## End(Not run)
```

`%>%` *The magrittr pipe*

Description

pointblank uses the pipe function, `%>%` to turn function composition into a series of imperative statements.

Index