Package: affirm (via r-universe)

November 14, 2024

Title Secular affirmations against data

Version 0.2.0.9001

Description What the package does (one paragraph).

License MIT + file LICENSE

URL https://github.com/pages/pcctc/affirm/,
https://pcctc.github.io/affirm/

BugReports https://github.com/pcctc/affirm/issues

Depends R (>= 4.2)

Imports base64enc (>= 0.1-3), cli (>= 3.6.1), dplyr (>= 1.1.2), glue (>= 1.6.2), gt (>= 0.9.0), htmltools (>= 0.5.5), openxlsx2 (>= 1.1), readr (>= 2.1.4), rlang (>= 1.1.1)

Suggests knitr, rmarkdown, testthat (>= 3.0.0), webshot2 (>= 0.1.0), tibble (>= 3.2.1)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Config/pak/sysreqs make libicu-dev libxml2-dev libssl-dev libnode-dev libx11-dev

Repository https://ddsjoberg.r-universe.dev

RemoteUrl https://github.com/pcctc/affirm

RemoteRef HEAD

RemoteSha 6fe15ddb3b8d5162218eecfc5c33756d495668d3

Contents

`affirm_false` *Affirm False*

Description

A wrapper for `affirm_true()`. The `condition` argument is process and passed to `affirm_true(condition = !condition)`

Usage

```
affirm_false(
  data,
  label,
  condition,
  id = NA_integer_,
  priority = NA_integer_,
  data_frames = NA_character_,
  columns = NA_character_,
  report_listing = NULL,
  data_action = NULL,
  error = getOption("affirm.error", default = FALSE)
)
```

Arguments

`data`	a data frame
`label`	a string used to describe the affirmation
`condition`	expression to check that evaluates to a logical vector, e.g. `cyl %in% c(4, 6, 8)`. Use the dot (`.`) to reference the passed data frame. If condition results in a missing value, it is interpreted as `FALSE`.
`id`, `priority`, `data_frames`, `columns`	Optional additional information that will be passed to affirmation report.

- `id` must be an integer, e.g. `id = 1L`
- `priority` must be an integer, e.g. `priority = 1L`
- `data_frames` string of data frame names used in affirmation, e.g. `data_frames = "RAND, DM"`
- `columns` string of column names used in affirmation. default is `all.vars(condition)`

`report_listing` an expression selecting/filtering rows from `data=` to return in the issue listing report. The default is to return the result from `create_report_listing()`, which are the rows that do *not* met in `condition=` and columns included in the `condition=` expression along with any columns set in `option('affirm.id_cols')`. The `'affirm.id_cols'` option must be a character vector of column names, where columns will be selected with `dplyr::select(any_of(getOption('affirm.id_cols')))`.

`data_action` this expression is executed at the end of the function call when supplied.

- Default is NULL, and the passed data frame in `data=` is returned unaltered.
- Perhaps you'll need to remove problematic rows: `data_action = dplyr::filter(., !(!!condition))`

`error` Logical indicating whether to throw an error when condition is not met. Default is `FALSE`.

Value

data frame

See Also

Other Data Affirmations: `affirm_na()`, `affirm_no_dupes()`, `affirm_range()`, `affirm_true()`, `affirm_values()`

Examples

```
affirm_init(replace = TRUE)

dplyr::as_tibble(mtcars) |>
 affirm_false(
   label = "No. cylinders must be 4, 6, or 8",
   condition = !cyl %in% c(4, 6, 8)
 )

affirm_close()
```

---

affirm_init *Begin Affirmations*

Description

Run this function to initialize a new affirmation report

Usage

```
affirm_init(replace = NA)

affirm_close()
```

Arguments

replace	logical indicating whether to replace/delete an existing or in-progress affirmation report. Default is NA, and user will interactively be asked whether to replace a report if it exists.

Examples

```
affirm_init()

affirm_close()
```

---

affirm_na *Affirm Class*

Description

A wrapper for affirm_true(). Reports columns that do not inherit class, e.g. dplyr::select(data, all_of(columns) &

A wrapper for affirm_true(). Reports columns whose names end with ".x" or ".y", indicating a sloppy merge.

A wrapper for affirm_true(). The columns argument is used to construct the affirm_true(condition = is.na(column)) argument.

Usage

```
affirm_class(
  data,
  label,
  columns,
  class,
  id = NA_integer_,
  priority = NA_integer_,
  data_frames = NA_character_,
  report_listing = NULL,
  data_action = NULL,
  error = getOption("affirm.error", default = FALSE)
)

affirm_clean_join(
  data,
  label,
```

```
  id = NA_integer_,
  priority = NA_integer_,
  data_frames = NA_character_,
  report_listing = NULL,
  data_action = NULL,
  error = getOption("affirm.error", default = FALSE)
)

affirm_na(
  data,
  label,
  column,
  id = NA_integer_,
  priority = NA_integer_,
  data_frames = NA_character_,
  report_listing = NULL,
  data_action = NULL,
  error = getOption("affirm.error", default = FALSE)
)

affirm_not_na(
  data,
  label,
  column,
  id = NA_integer_,
  priority = NA_integer_,
  data_frames = NA_character_,
  report_listing = NULL,
  data_action = NULL,
  error = getOption("affirm.error", default = FALSE)
)
```

Arguments

`data`	a data frame
`label`	a string used to describe the affirmation
`columns`	columns to check class
`class`	character class to affirm
`id`, `priority`, `data_frames`	Optional additional information that will be passed to affirmation report. • `id` must be an integer, e.g. `id = 1L` • `priority` must be an integer, e.g. `priority = 1L` • `data_frames` string of data frame names used in affirmation, e.g. `data_frames = "RAND, DM"`
`report_listing`	an expression selecting/filtering rows from `data=` to return in the issue listing report. The default is to return the result from `create_report_listing()`, which are the rows that do *not* met in `condition=` and columns included in the

condition= expression along with any columns set in option('affirm.id_cols'). The 'affirm.id_cols' option must be a character vector of column names, where columns will be selected with dplyr::select(any_of(getOption('affirm.id_cols'))).

data_action this expression is executed at the end of the function call when supplied.

- Default is NULL, and the passed data frame in data= is returned unaltered.
- Perhaps you'll need to remove problematic rows: data_action = dplyr::filter(., !(!!condition))

error Logical indicating whether to throw an error when condition is not met. Default is FALSE.

column column to check NA values against

Value

data frame

data frame

data frame

See Also

Other Data Affirmations: affirm_false(), affirm_no_dupes(), affirm_range(), affirm_true(), affirm_values()

Other Data Affirmations: affirm_false(), affirm_no_dupes(), affirm_range(), affirm_true(), affirm_values()

Other Data Affirmations: affirm_false(), affirm_no_dupes(), affirm_range(), affirm_true(), affirm_values()

Examples

```
affirm_init(replace = TRUE)

affirm_class(
  dplyr::as_tibble(iris),
  label = "all cols are numeric (but Species really isn't)",
  columns = everything(),
  class = "numeric"
)

affirm_close()
affirm_init(replace = TRUE)

df <-
  dplyr::tibble(lgl = c(NA, TRUE, NA, FALSE, NA)) |>
  dplyr::mutate(id = dplyr::row_number())

affirm_clean_join(
  dplyr::full_join(df, df, by = "id"),
  label = "Checking for clean merge"
)
```

```
affirm_close()
affirm_init(replace = TRUE)
```

`affirm_no_dupes` *Affirm Range*

Description

A wrapper for `affirm_true()`. The columns argument is used to construct the `affirm_true(condition = dplyr::select(., all_of(columns)) |> duplicated())` argument.

Usage

```
affirm_no_dupes(
  data,
  label,
  columns,
  id = NA_integer_,
  priority = NA_integer_,
  data_frames = NA_character_,
  report_listing = NULL,
  data_action = NULL,
  error = getOption("affirm.error", default = FALSE)
)
```

Arguments

`data`	a data frame
`label`	a string used to describe the affirmation
`columns`	columns to check duplicates among
`id`, `priority`, `data_frames`	Optional additional information that will be passed to affirmation report. • `id` must be an integer, e.g. `id = 1L` • `priority` must be an integer, e.g. `priority = 1L` • `data_frames` string of data frame names used in affirmation, e.g. `data_frames = "RAND, DM"`
`report_listing`	an expression selecting/filtering rows from `data=` to return in the issue listing report. The default is to return the result from `create_report_listing()`, which are the rows that do *not* met in `condition=` and columns included in the `condition=` expression along with any columns set in `option('affirm.id_cols')`. The `'affirm.id_cols'` option must be a character vector of column names, where columns will be selected with `dplyr::select(any_of(getOption('affirm.id_cols')))`.
`data_action`	this expression is executed at the end of the function call when supplied. • Default is NULL, and the passed data frame in `data=` is returned unaltered.

- Perhaps you'll need to remove problematic rows: `data_action = dplyr::filter(., !(!!condition))`

`error` Logical indicating whether to throw an error when condition is not met. Default is `FALSE`.

Value

data frame

See Also

Other Data Affirmations: `affirm_false()`, `affirm_na()`, `affirm_range()`, `affirm_true()`, `affirm_values()`

Examples

```
affirm_init(replace = TRUE)

dplyr::as_tibble(mtcars) |>
 affirm_no_dupes(
   label = "No duplicates in the number of cylinders",
   columns = cyl
 )

affirm_close()
```

---

affirm_range *Affirm Range*

Description

A wrapper for `affirm_true()`. The column, range, and boundaries arguments are used to construct the `affirm_true(condition = column >= range[1] & column <= range[2])` argument.

Usage

```
affirm_range(
  data,
  label,
  column,
  range,
  boundaries = c(TRUE, TRUE),
  id = NA_integer_,
  priority = NA_integer_,
  data_frames = NA_character_,
  report_listing = NULL,
  data_action = NULL,
  error = getOption("affirm.error", default = FALSE)
)
```

Arguments

`data`	a data frame
`label`	a string used to describe the affirmation
`column`	a single column to check values of
`range`	vector of length two indicating the upper and lower bounds of the range. The class of the `range` must be compatible with the `column`, e.g. if `column` is numeric, `range` must also be numeric; if `column` is a date, range must be a date; if `column` is an integer, `range` must be an integer, etc.
`boundaries`	logical vector of length 2 indicating whether to include UB and LB in the range check. Default is `c(TRUE, TRUE)`
`id`, `priority`, `data_frames`	Optional additional information that will be passed to affirmation report. • `id` must be an integer, e.g. `id = 1L` • `priority` must be an integer, e.g. `priority = 1L` • `data_frames` string of data frame names used in affirmation, e.g. `data_frames = "RAND, DM"`
`report_listing`	an expression selecting/filtering rows from `data=` to return in the issue listing report. The default is to return the result from `create_report_listing()`, which are the rows that do *not* met in `condition=` and columns included in the `condition=` expression along with any columns set in `option('affirm.id_cols')`. The `'affirm.id_cols'` option must be a character vector of column names, where columns will be selected with `dplyr::select(any_of(getOption('affirm.id_cols')))`.
`data_action`	this expression is executed at the end of the function call when supplied. • Default is NULL, and the passed data frame in `data=` is returned unaltered. • Perhaps you'll need to remove problematic rows: `data_action = dplyr::filter(., !(!!condition))`
`error`	Logical indicating whether to throw an error when condition is not met. Default is `FALSE`.

Value

data frame

See Also

Other Data Affirmations: `affirm_false()`, `affirm_na()`, `affirm_no_dupes()`, `affirm_true()`, `affirm_values()`

Examples

```
affirm_init(replace = TRUE)

dplyr::as_tibble(mtcars) |>
 affirm_range(
   label = "MPG is >0 and <=30",
   column = mpg,
```

```
    range = c(0, 30),
    boundaries = c(FALSE, TRUE)
  )

affirm_close()
```

`affirm_report` *Affirmation Report*

Description

- `affirm_report_gt()` returns styled gt table summarizing results of affirmation session.
- `affirm_report_excel()` returns excel file with one sheet per affirmation (excluding those with no errors)
- `affirm_report_raw_data()` returns raw data used to generate summary in `affirm_report_gt()`

Usage

```
affirm_report_gt()

affirm_report_excel(
  file,
  affirmation_name = "{data_frames}{id}",
  overwrite = TRUE
)

affirm_report_raw_data()
```

Arguments

`file`	A file path to save the xlsx file
`affirmation_name`	A string for affirmation names; the item name in curly brackets is replaced with the item value (see glue::glue). Item names accepted include: `id`, `label`, `priority`, `data_frames`, `columns`, `error_n`, `total_n`. Defaults to `"{data_frames}{id}"`.
`overwrite`	Overwrite existing file (Defaults to `TRUE` as with `write.table`)

Value

gt table

Examples

```
affirm_init(replace = TRUE)

dplyr::as_tibble(mtcars) |>
 affirm_true(
   label = "No. cylinders must be 4, 6, or 8",
   condition = cyl %in% c(4, 6, 8)
 ) |>
 affirm_true(
    label = "MPG should be less than 33",
    condition = mpg < 33
 )

gt_report <- affirm_report_gt()

affirm_close()
```

`affirm_true` *Affirm True*

Description

Use this function to affirm an expression is true.

Usage

```
affirm_true(
  data,
  label,
  condition,
  id = NA_integer_,
  priority = NA_integer_,
  data_frames = NA_character_,
  columns = NA_character_,
  report_listing = NULL,
  data_action = NULL,
  error = getOption("affirm.error", default = FALSE)
)
```

Arguments

`data`	a data frame
`label`	a string used to describe the affirmation
`condition`	expression to check that evaluates to a logical vector, e.g. `cyl %in% c(4, 6, 8)`. Use the dot (`.`) to reference the passed data frame. If condition results in a missing value, it is interpreted as `FALSE`.

`id`, `priority`, `data_frames`, `columns`
: Optional additional information that will be passed to affirmation report.
  - `id` must be an integer, e.g. `id = 1L`
  - `priority` must be an integer, e.g. `priority = 1L`
  - `data_frames` string of data frame names used in affirmation, e.g. `data_frames = "RAND, DM"`
  - `columns` string of column names used in affirmation. default is `all.vars(condition)`

`report_listing`
: an expression selecting/filtering rows from `data=` to return in the issue listing report. The default is to return the result from `create_report_listing()`, which are the rows that do *not* met in `condition=` and columns included in the `condition=` expression along with any columns set in `option('affirm.id_cols')`. The `'affirm.id_cols'` option must be a character vector of column names, where columns will be selected with `dplyr::select(any_of(getOption('affirm.id_cols')))`.

`data_action`
: this expression is executed at the end of the function call when supplied.
  - Default is NULL, and the passed data frame in `data=` is returned unaltered.
  - Perhaps you'll need to remove problematic rows: `data_action = dplyr::filter(., !(!!condition))`

`error`
: Logical indicating whether to throw an error when condition is not met. Default is `FALSE`.

Details

When passing expressions to arguments `report_listing=` and `data_action=`, there are a few things to keep in mind.

- The expression passed in `condition=` can be used, but note that it has been captured as an expression inside the function. This means that to use it, you'll need to use `!!` (bang-bang) to pass it inside a function.
- In addition to being able to use the `condition=` expression, you can simplify your code somewhat by referring to `lgl_condition`, which is an evaluated logical vector of the `condition=` expression.

Value

data frame

See Also

Other Data Affirmations: `affirm_false()`, `affirm_na()`, `affirm_no_dupes()`, `affirm_range()`, `affirm_values()`

Examples

```
affirm_init(replace = TRUE)

dplyr::as_tibble(mtcars) |>
 affirm_true(
   label = "No. cylinders must be 4, 6, or 8",
```

```
  condition = cyl %in% c(4, 6, 8)
)

affirm_close()
```

`affirm_values` *Affirm Values*

Description

A wrapper for `affirm_true()`. The column and value arguments are used to construct the `affirm_true(condition = column %in% value)` argument.

Usage

```
affirm_values(
  data,
  label,
  column,
  values,
  id = NA_integer_,
  priority = NA_integer_,
  data_frames = NA_character_,
  report_listing = NULL,
  data_action = NULL,
  error = getOption("affirm.error", default = FALSE)
)
```

Arguments

`data`	a data frame
`label`	a string used to describe the affirmation
`column`	a single column to check values of
`values`	vector of values the `column=` may take on
`id`, `priority`, `data_frames`	Optional additional information that will be passed to affirmation report. • `id` must be an integer, e.g. `id = 1L` • `priority` must be an integer, e.g. `priority = 1L` • `data_frames` string of data frame names used in affirmation, e.g. `data_frames = "RAND, DM"`
`report_listing`	an expression selecting/filtering rows from `data=` to return in the issue listing report. The default is to return the result from `create_report_listing()`, which are the rows that do *not* met in `condition=` and columns included in the `condition=` expression along with any columns set in `option('affirm.id_cols')`. The `'affirm.id_cols'` option must be a character vector of column names, where columns will be selected with `dplyr::select(any_of(getOption('affirm.id_cols')))`.

data_action	this expression is executed at the end of the function call when supplied. • Default is NULL, and the passed data frame in data= is returned unaltered. • Perhaps you'll need to remove problematic rows: data_action = dplyr::filter(., !(!!condition))
error	Logical indicating whether to throw an error when condition is not met. Default is FALSE.

Value

data frame

See Also

Other Data Affirmations: affirm_false(), affirm_na(), affirm_no_dupes(), affirm_range(), affirm_true()

Examples

```
affirm_init(replace = TRUE)

dplyr::as_tibble(mtcars) |>
 affirm_values(
   label = "No. cylinders must be 4, 6, or 8",
   column = cyl,
   values = c(4, 6, 8)
 )

affirm_close()
```

DM *Subject Demographics*

Description

A data set containing demographics for enrolled subjects in a trial.

Usage

```
DM
```

Format

A data frame

SUBJECT Subject ID

AGE Age at Randomization

RACE Race

prepend_df_name *Prepend DF Name to Column Names*

Description

Prepend DF Name to Column Names

Usage

```
prepend_df_name(
  data,
  df_name = NULL,
  include = c(everything(), -any_of(getOption("affirm.id_cols")))
)
```

Arguments

data	a data frame
df_name	string indicating the data frame name to prepend to the column names. If not supplied, function will try to identify the data frame name. NOTE: We can only get the correct name if the data frame has been piped directly to this function without any other piped function between.
include	tidyselect expression to identify columns to modify name. Default is all columns, except those identified in `options("affirm.id_cols")`.

Value

a data frame

Examples

```
DM |>
 prepend_df_name()
```

RAND *Subject Randomization*

Description

A data set containing randomization assignment from a trial.

Usage

```
RAND
```

Format

A data frame

SUBJECT Subject ID

RAND_GROUP Randomization Assignment

RAND_STRATA Randomization Strata Value

Index