Package ‘datadiff’

March 18, 2026

Title Data Validation Based on YAML Rules

Version 0.4.4

Description A comprehensive data validation package that allows comparing datasets using configurable validation rules defined in 'YAML' files. Built on top of the 'pointblank' package for robust data validation, it supports exact matching, tolerance-based numeric comparisons, text normalization, and row count validation.

License MIT + file LICENSE

Depends R (>= 4.0.0)

Imports dplyr (>= 1.0.0), pointblank (>= 0.12.3), rlang (>= 0.4.0), stats, tidyselect (>= 1.0.0), utils, yaml (>= 2.2.0)

Suggests arrow, DBI, dbplyr, duckdb, knitr, rmarkdown, RSQLite, stringr, testthat (>= 3.0.0), tibble

VignetteBuilder knitr

URL `https://github.com/ThinkR-open/datadiff`

BugReports `https://github.com/ThinkR-open/datadiff/issues`

Encoding UTF-8

RoxygenNote 7.3.3

Config/testthat/edition 3

NeedsCompilation no

Author Vincent Guyader [cre, aut] (ORCID: <`https://orcid.org/0000-0003-0671-9270`>),
ThinkR [cph],
Agence technique de l'information sur l'hospitalisation [spn]

Maintainer Vincent Guyader <`vincent@thinkr.fr`>

Repository CRAN

Date/Publication 2026-03-18 11:00:02 UTC

Contents

`add_tolerance_columns` *Add tolerance columns for numeric comparisons*

Description

Creates additional columns in the comparison dataframe to handle numeric tolerance validation. For each numeric column with tolerance rules, adds absolute difference and threshold columns, plus a boolean column indicating if values are within acceptable tolerance.

Usage

```
add_tolerance_columns(cmp, tol_cols, col_rules, ref_suffix, na_equal)
```

Arguments

`cmp`	Comparison dataframe containing candidate and reference columns
`tol_cols`	Character vector of column names that have tolerance rules
`col_rules`	List of column-specific validation rules
`ref_suffix`	Suffix used for reference columns (default: "__reference")
`na_equal`	Logical indicating if NA values should be considered equal

Value

Modified dataframe with added tolerance columns

Examples

```
cmp <- data.frame(value = c(1.1, 2.2), value__reference = c(1.0, 2.0))
rules <- list(value = list(abs = 0.2, rel = 0.1))
add_tolerance_columns(cmp, "value", rules, "__reference", TRUE)
```

`analyze_columns` *Analyze column differences between datasets*

Description

Compares column names between reference and candidate datasets to identify common columns, missing columns, and extra columns. Can ignore specified columns.

Usage

```
analyze_columns(data_reference, data_candidate, ignore_columns = character(0))
```

Arguments

`data_reference`	Reference dataframe
`data_candidate`	Candidate dataframe to compare
`ignore_columns`	Character vector of column names to ignore during comparison

Value

A list containing:

`cols_reference`	Column names in reference data (excluding ignored)
`cols_candidate`	Column names in candidate data (excluding ignored)
`missing_in_candidate`	Columns present in reference but missing in candidate
`extra_in_candidate`	Columns present in candidate but not in reference
`common_cols`	Columns present in both datasets
`ignored_cols`	Columns that were ignored from comparison

Examples

```
ref <- data.frame(a = 1, b = 2, c = 3, d = 4)
cand <- data.frame(a = 1, b = 2, c = 3, e = 5)
analyze_columns(ref, cand, ignore_columns = c("d", "e"))
```

compare_datasets_from_yaml

Compare datasets using YAML validation rules

Description

Main function for comparing reference and candidate datasets using configurable validation rules defined in a YAML file. Supports exact matching, tolerance-based comparisons, text normalization, and row count validation.

Usage

```
compare_datasets_from_yaml(
  data_reference,
  data_candidate,
  key = NULL,
  path = NULL,
  warn_at = 1e-14,
  stop_at = 1e-14,
  ref_suffix = "__reference",
  label = NULL,
 error_msg_no_key = "Without keys, both tables must have the same number of rows.",
  lang = getOption("datadiff.lang", "en"),
  locale = getOption("datadiff.locale", "en_US"),
  extract_failed = TRUE,
  get_first_n = NULL,
  sample_n = NULL,
  sample_frac = NULL,
  sample_limit = 5000,
  duckdb_memory_limit = "8GB"
)
```

Arguments

`data_reference`	Reference dataframe, tibble, or lazy table (tbl_lazy)
`data_candidate`	Candidate dataframe to validate against reference
`key`	Optional character vector of column names to use as join keys for ordered comparison
`path`	Path to YAML file containing validation rules. If NULL, default rules are generated automatically based on the reference dataset structure.
`warn_at`	Warning threshold as fraction of failing tests (default: 1e-14)
`stop_at`	Stop threshold as fraction of failing tests (default: 1e-14)
`ref_suffix`	Suffix for reference columns in comparison dataframe (default: "__reference")
`label`	Descriptive label for the validation report

`error_msg_no_key`	Error message when datasets have different row counts without keys
`lang`	Language code for pointblank reports. Defaults to the `datadiff.lang` option if set, otherwise `"en"`. Set globally with `options(datadiff.lang = "fr")`. Supported values include "en" (English), "fr" (French), "de" (German), "it" (Italian), "es" (Spanish), "pt" (Portuguese), "zh" (Chinese), "ja" (Japanese), "ru" (Russian), etc. See pointblank documentation for full list.
`locale`	Locale code for number and date formatting. Defaults to the `datadiff.locale` option if set, otherwise `"en_US"`. Set globally with `options(datadiff.locale = "fr_FR")`. Examples: "en_GB", "fr_FR", "de_DE", "es_ES", "pt_BR", "zh_CN", "ja_JP".
`extract_failed`	Logical indicating whether to collect rows that failed validation (default: TRUE). Set to FALSE to reduce memory usage for large datasets with many errors.
`get_first_n`	Integer specifying the maximum number of failed rows to extract per validation step (default: NULL, meaning all). Useful to limit memory when many rows fail.
`sample_n`	Integer specifying a fixed number of failed rows to randomly sample per validation step (default: NULL). Alternative to get_first_n for random sampling.
`sample_frac`	Numeric between 0 and 1 specifying the fraction of failed rows to sample (default: NULL). Used with sample_limit for proportional sampling.
`sample_limit`	Integer specifying the maximum number of rows when using sample_frac (default: 5000). Acts as a ceiling for sampled rows.
`duckdb_memory_limit`	Character string passed to DuckDB's `SET memory_limit` when Arrow datasets are used (default: `"8GB"`). Controls how much RAM DuckDB may use before spilling intermediate results to `tempdir()`. The default leaves headroom for R, Arrow, and the OS alongside DuckDB. Raise it (e.g. `"16GB"`) on machines with ample free RAM to reduce disk I/O; lower it (e.g. `"4GB"`) when memory is very constrained. Has no effect when both inputs are plain `data.frames` or `tbl_lazy` objects.

Value

A list containing:

`agent`	Configured pointblank agent with validation results
`reponse`	Interrogation results from pointblank
`missing_in_candidate`	Columns missing in candidate data
`extra_in_candidate`	Extra columns in candidate data
`applied_rules`	Final column-specific rules applied

Examples

```
# Create test data
ref <- data.frame(id = 1:3, value = c(1.0, 2.0, 3.0))
cand <- data.frame(id = 1:3, value = c(1.1, 2.1, 3.1))

# Compare datasets without YAML (uses default rules, positional comparison)
result <- compare_datasets_from_yaml(ref, cand)

# Compare datasets with key but without YAML
result <- compare_datasets_from_yaml(ref, cand, key = "id")

# Compare datasets with custom YAML rules
tmp <- tempfile(fileext = ".yaml")
write_rules_template(ref, key = "id", path = tmp)
result <- compare_datasets_from_yaml(ref, cand, key = "id", path = tmp)
result$reponse
```

`derive_column_rules` *Derive column-specific validation rules*

Description

Combines type-based and column-specific validation rules into a unified list of rules for each column in the dataset.

Usage

```
derive_column_rules(data_reference, rules)
```

Arguments

`data_reference`	Reference dataframe used to determine column types
`rules`	List of validation rules from read_rules()

Value

List of column-specific rules with type and name-based rules merged

Examples

```
df <- data.frame(a = 1:3, b = c(1.1, 2.2, 3.3))
rules <- list(by_type = list(numeric = list(abs = 0.1)), by_name = list(a = list(abs = 0.5)))
derive_column_rules(df, rules)
```

`detect_column_types` *Detect column types in a dataframe*

Description

Analyzes each column of a dataframe to determine its data type (integer, numeric, date, datetime, logical, or character)

Usage

```
detect_column_types(data)
```

Arguments

`data`	A dataframe or tibble

Value

A named character vector with column names as names and types as values

Examples

```
df <- data.frame(a = 1:3, b = c(1.1, 2.2, 3.3), c = c("x", "y", "z"))
detect_column_types(df)
```

`normalize_text` *Normalize text for comparison*

Description

Applies text normalization transformations including case conversion and whitespace trimming. Non-character inputs are returned unchanged.

Usage

```
normalize_text(x, case_insensitive = FALSE, trim = FALSE)
```

Arguments

`x`	Vector to normalize (typically character)
`case_insensitive`	Logical indicating whether to convert to lowercase
`trim`	Logical indicating whether to trim leading/trailing whitespace

Value

Normalized vector of the same type as input

Examples

```
normalize_text(c("  Hello ", "WORLD  "), case_insensitive = TRUE, trim = TRUE)
# Returns: c("hello", "world")
```

---

or_operator *Operator for default values*

Description

Returns y if x is NULL, otherwise returns x

Usage

```
x %||% y
```

Arguments

`x`	Value to check
`y`	Default value to return if x is NULL

Value

x if not NULL, otherwise y

---

preprocess_dataframe *Preprocess dataframe according to column rules*

Description

Applies preprocessing transformations to dataframe columns based on validation rules, such as text normalization for character columns. Supports both local data.frames and lazy tables (tbl_lazy) via dplyr::mutate().

Usage

```
preprocess_dataframe(df, col_rules, schema = NULL)
```

Arguments

`df`	A dataframe or lazy table to preprocess
`col_rules`	A list of column-specific rules from derive_column_rules()
`schema`	Optional local data.frame with 0 rows used to determine column types when `df` is a lazy table. Obtained via `dplyr::collect(utils::head(df, 0L))`.

Value

Preprocessed dataframe (or lazy table) with transformations applied

Examples

```
df <- data.frame(text_col = c("  HELLO  ", "world"))
rules <- list(text_col = list(equal_mode = "normalized", case_insensitive = TRUE, trim = TRUE))
preprocess_dataframe(df, rules)
```

---

read_rules   *Read and validate YAML rules file*

Description

Loads validation rules from a YAML file and ensures the format is valid. Adds default values for missing configuration sections.

Usage

```
read_rules(path)
```

Arguments

`path`	Character string specifying the path to the YAML rules file

Value

A list containing the parsed and validated rules configuration

Examples

```
tmp <- tempfile(fileext = ".yaml")
write_rules_template(data.frame(id = 1L, value = 1.0), key = "id", path = tmp)
rules <- read_rules(tmp)
```

`setup_pointblank_agent` *Setup pointblank agent for data validation*

Description

Creates and configures a pointblank validation agent with all necessary validation steps including column existence checks, exact value comparisons, and tolerance validations.

Usage

```
setup_pointblank_agent(
  cmp,
  cols_reference,
  common_cols,
  tol_cols,
  row_validation_info = NULL,
  ref_suffix,
  warn_at,
  stop_at,
  label,
  na_equal,
  lang = getOption("datadiff.lang", "en"),
  locale = getOption("datadiff.locale", "en_US"),
  missing_in_candidate = character(0),
  type_mismatch_cols = character(0),
  add_col_exists_steps = TRUE
)
```

Arguments

`cmp`	Comparison dataframe with candidate and reference data
`cols_reference`	Character vector of reference column names
`common_cols`	Character vector of columns present in both datasets
`tol_cols`	Character vector of columns with tolerance rules
`row_validation_info`	List with row validation information from validate_row_counts
`ref_suffix`	Suffix for reference columns
`warn_at`	Warning threshold (fraction of failing tests)
`stop_at`	Stop threshold (fraction of failing tests)
`label`	Descriptive label for the validation
`na_equal`	Logical indicating if NA values are considered equal

`lang`	Language code for pointblank reports. Defaults to the `datadiff.lang` option if set, otherwise `"en"`. Set globally with `options(datadiff.lang = "fr")`. Supported values include "en" (English), "fr" (French), "de" (German), "it" (Italian), "es" (Spanish), "pt" (Portuguese), "zh" (Chinese), "ja" (Japanese), "ru" (Russian), etc. See pointblank documentation for full list.
`locale`	Locale code for number and date formatting. Defaults to the `datadiff.locale` option if set, otherwise `"en_US"`. Set globally with `options(datadiff.locale = "fr_FR")`. Examples: "en_GB", "fr_FR", "de_DE", "es_ES", "pt_BR", "zh_CN", "ja_JP".
`missing_in_candidate`	Character vector of columns missing in candidate dataset
`type_mismatch_cols`	Character vector of columns whose type differs between reference and candidate (e.g. numeric in reference, character in candidate). A dedicated failing validation step labelled `type_mismatch: <column>` is added for each such column.
`add_col_exists_steps`	Logical indicating whether to add `col_exists` validation steps for common columns (default: `TRUE`). Set to `FALSE` for the non-local (lazy table) path where `cmp` only contains pre-computed boolean columns, not the original data columns.

Value

Configured pointblank agent ready for interrogation

Examples

```
cmp <- data.frame(a = 1:3, a__reference = 1:3, b = c(1.1, 2.2, 3.3),
 b__reference = c(1.0, 2.0, 3.0))
row_info <- list(check_count = FALSE)
setup_pointblank_agent(cmp, c("a", "b"), c("a", "b"), "b", row_info,
 "__reference", 0.1, 0.1, "Test", TRUE)
```

`validate_row_counts` *Validate row counts according to rules*

Description

Checks if the number of rows in candidate data matches expectations defined in rules. Can validate against a fixed expected count or against reference data count with tolerance. Returns validation information that can be used by pointblank agents.

Usage

```
validate_row_counts(data_reference_p, data_candidate_p, rules)
```

Arguments

`data_reference_p`	Preprocessed reference dataframe
`data_candidate_p`	Preprocessed candidate dataframe
`rules`	List of validation rules containing row_validation settings

Value

A list with validation information for pointblank integration

Examples

```
ref <- data.frame(a = 1:3)
cand <- data.frame(a = 1:3)
rules <- list(row_validation = list(check_count = TRUE, expected_count = 3, tolerance = 0))
validate_row_counts(ref, cand, rules)
```

`write_rules_template` *Create a YAML rules template for data validation*

Description

Generates a comprehensive YAML configuration file with default validation rules based on the structure and types of the reference dataset. The template includes rules for different data types, column-specific rules, and row validation settings.

Usage

```
write_rules_template(
  data_reference,
  key = NULL,
  label = NULL,
  path = "rules.yaml",
  version = 1L,
  na_equal_default = TRUE,
  ignore_columns_default = character(0),
  check_count_default = TRUE,
  expected_count_default = NULL,
  row_count_tolerance_default = 0,
  numeric_abs = 1e-09,
  numeric_rel = 0,
  integer_abs = 0L,
  character_equal_mode = "exact",
  character_case_insensitive = FALSE,
  character_trim = FALSE,
  date_equal_mode = "exact",
  datetime_equal_mode = "exact",
  logical_equal_mode = "exact"
)
```

Arguments

`data_reference`	A dataframe or tibble used as reference for rule generation
`key`	Character vector specifying column name(s) to use as join key(s) for data comparison. If NULL, comparison is positional (row by row).
`label`	Descriptive label for the validation report
`path`	Character string specifying the output YAML file path (default: "rules.yaml")
`version`	Numeric version of the rules format (default: 1)
`na_equal_default`	Logical indicating if NA values should be considered equal by default
`ignore_columns_default`	Character vector of column names to ignore during comparison by default
`check_count_default`	Logical indicating if row count validation should be enabled by default
`expected_count_default`	Numeric value specifying expected row count (NULL uses reference count)
`row_count_tolerance_default`	Numeric tolerance for row count validation
`numeric_abs`	Default absolute tolerance for numeric columns
`numeric_rel`	Default relative tolerance for numeric columns
`integer_abs`	Default absolute tolerance for integer columns
`character_equal_mode`	Default comparison mode for character columns ("exact", "normalized")
`character_case_insensitive`	Logical for case-insensitive character comparison
`character_trim`	Logical for trimming whitespace in character comparison
`date_equal_mode`	Default comparison mode for date columns
`datetime_equal_mode`	Default comparison mode for datetime columns
`logical_equal_mode`	Default comparison mode for logical columns

Value

The path to the created YAML file, invisibly.

Examples

```
df <- data.frame(id = 1:3, value = c(1.1, 2.2, 3.3), name = c("A", "B", "C"))
write_rules_template(df, key = "id", path = tempfile(fileext = ".yaml"))
```

Index