Package: dtaudit (via r-universe)

May 20, 2026

Title Audit and Diagnostic Tools for 'data.table' Workflows

Version 0.1.1

Description Diagnostic tools for auditing data analysis workflows built on 'data.table'. Provides functions to validate join operations, compare data.tables, filter with diagnostic output, summarize data quality, check primary keys and variable relationships, and diagnose string columns. Designed to help analysts understand and document data transformations.

License LGPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports data.table, stringi

URL https://github.com/fpcordeiro/dtaudit

BugReports https://github.com/fpcordeiro/dtaudit/issues

Suggests knitr, pbapply, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/pak/sysreqs libicu-dev

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

Date/Publication 2026-02-17 12:46:08 UTC

RemoteUrl https://github.com/fpcordeiro/dtaudit

RemoteRef HEAD

RemoteSha 5d00194a55efc5da163e7f9d65165cc527bfb257

Contents

`audit_clean` *Audit String Cleaning Operation*

Description

Applies a cleaning function to a character vector and reports what changed. Provides transparency about the cleaning operation by showing counts and before/after examples.

Usage

```
audit_clean(x, clean_fn, name = NULL)
```

Arguments

`x`	Character vector to clean.
`clean_fn`	A function that takes a character vector and returns a cleaned character vector of the same length.
`name`	Optional name for the variable (used in output). If NULL, attempts to capture the variable name from the call.

Value

An S3 object of class `audit_clean` containing:

name Name of the variable

clean_fn_name Name of the cleaning function used

n_total Total number of elements

n_changed Count of values that changed

n_unchanged Count of values that stayed the same

n_na Count of NA values (unchanged by definition)

pct_changed Percentage of non-NA values that changed

change_examples data.table with sample before/after pairs

cleaned The cleaned character vector

See Also

`diagnose_strings()` for string quality diagnostics, `clean_var_names()` and `clean_firm_name()` for built-in cleaning functions

Examples

```
library(data.table)
firms <- c("Apple Inc.", "MICROSOFT CORP", "Alphabet LLC", NA)
result <- audit_clean(firms, clean_firm_name)
result$cleaned
```

---

check_date_coverage *Check Date Coverage*

Description

Verifies whether a date vector contains data for all periods within a specified date range. Reports any missing periods.

Usage

```
check_date_coverage(
  date_var,
  start_date,
  end_date,
  by = "month",
  quiet = FALSE
)
```

Arguments

date_var	A vector of dates to check.
start_date	Character or Date. Start of the expected date range (format: "YYYY-MM-DD").
end_date	Character or Date. End of the expected date range (format: "YYYY-MM-DD").
by	Character. Period granularity: one of "day", "week", "month", "quarter", or "year". Defaults to "month".
quiet	Logical. If TRUE, suppresses printed output. Defaults to FALSE.

Value

An IDate vector of missing periods, returned invisibly.

See Also

check_months_coverage() for a convenience wrapper with by = "month"

Examples

```
library(data.table)
dates <- as.IDate(c("2023-01-15", "2023-02-20", "2023-04-10"))
check_date_coverage(dates, "2023-01-01", "2023-04-30")
check_date_coverage(dates, "2023-01-01", "2023-04-30", by = "quarter")
```

---

check_months_coverage *Check Monthly Date Coverage*

Description

Verifies whether a date vector contains data for all months within a specified date range. Reports any missing months.

Usage

```
check_months_coverage(date_var, start_date, end_date, quiet = FALSE)
```

Arguments

date_var	A vector of dates to check.
start_date	Character or Date. Start of the expected date range (format: "YYYY-MM-DD").
end_date	Character or Date. End of the expected date range (format: "YYYY-MM-DD").
quiet	Logical. If TRUE, suppresses printed output. Defaults to FALSE.

Details

This is a convenience wrapper around check_date_coverage() with by = "month".

Value

An IDate vector of missing months, returned invisibly.

See Also

`check_date_coverage()` for other period granularities

Examples

```
library(data.table)
dates <- as.IDate(c("2023-01-15", "2023-02-20", "2023-04-10"))
check_months_coverage(dates, "2023-01-01", "2023-04-30")
```

---

clean_firm_name *Clean Firm Names for Matching*

Description

Normalizes firm names by converting to uppercase ASCII, removing common suffixes (Corp, LLC, Inc, etc.), and standardizing whitespace. Useful for fuzzy matching or deduplication of company names.

Usage

```
clean_firm_name(text)
```

Arguments

`text`	Character vector of firm names to clean.

Value

Character vector of cleaned firm names.

Examples

```
clean_firm_name(c("Apple, Inc.", "MICROSOFT CORP.", "Alphabet LLC"))
```

clean_var_names *Clean Variable Names*

Description

Standardizes variable names by trimming whitespace, converting to lowercase ASCII, replacing all non-alphanumeric characters with underscores, and removing leading/trailing underscores.

Usage

```
clean_var_names(text)
```

Arguments

text	Character vector of variable names to clean.

Value

Character vector of cleaned variable names containing only lowercase letters, digits, and underscores.

Examples

```
clean_var_names(c("Sales Revenue", "cost-of-goods", " margin "))
# Returns: c("sales_revenue", "cost_of_goods", "margin")
```

compare_datatables *Compare Two Data Tables*

Description

Compares two data.tables by examining column names, row counts, key overlap, and numeric discrepancies. Useful for validating data processing pipelines.

Usage

```
compare_datatables(dt1, dt2, key_cols = NULL)
```

Arguments

dt1	First data.table to compare.
dt2	Second data.table to compare.
key_cols	Character vector of column names to use as keys for matching rows. If NULL (default), automatically detects character, factor, and integer columns as keys.

Value

An S3 object of class `compare_dt` containing:

name1, name2 Names of the compared objects

common_columns Column names present in both tables

only_dt1 Column names only in dt1

only_dt2 Column names only in dt2

type_mismatches Data.table of columns with same name but different types, with columns: column, type_dt1, type_dt2. NULL if no mismatches.

nrow_dt1 Number of rows in dt1

nrow_dt2 Number of rows in dt2

key_summary Summary of key overlap (if keys found)

numeric_summary Data.table of numeric column discrepancies

numeric_method How numeric columns were compared ("keys", "row_index", or NA)

rows_matched Number of rows matched on keys (when method is "keys")

See Also

`validate_join()` for analyzing join relationships, `validate_primary_keys()` for key uniqueness validation

Examples

```
library(data.table)
dt1 <- data.table(id = 1:3, value = c(10.0, 20.0, 30.0))
dt2 <- data.table(id = 1:3, value = c(10.1, 20.0, 30.5))
compare_datatables(dt1, dt2)
```

---

diagnose_nas *Diagnose Missing Values*

Description

Reports NA counts and percentages for each column in a data.table, sorted by missing percentage in descending order.

Usage

```
diagnose_nas(dt)
```

Arguments

`dt`	A data.table to diagnose.

Value

An S3 object of class `diagnose_na` containing:

table A data.table with columns `variable`, `n_na`, `pct_na`, and `n_valid`, sorted by `pct_na` descending.

n_cols Total number of columns in the input.

n_with_na Number of columns that have at least one NA.

See Also

`get_summary_table()` for comprehensive column summaries, `diagnose_strings()` for string column quality

Examples

```
library(data.table)
dt <- data.table(
  a = c(1, NA, 3),
  b = c(NA, NA, "x"),
  c = c(TRUE, FALSE, TRUE)
)
diagnose_nas(dt)
```

`diagnose_strings` *Diagnose String Column Quality*

Description

Audits a character vector for common data quality issues including missing values, empty strings, whitespace problems, non-ASCII characters, and case inconsistencies. Useful for understanding string data before cleaning.

Usage

```
diagnose_strings(x, name = NULL)
```

Arguments

`x`	Character vector to diagnose.
`name`	Optional name for the variable (used in output). If NULL, attempts to capture the variable name from the call.

Value

An S3 object of class `diagnose_strings` containing:

name Name of the variable

n_total Total number of elements

n_na Count of NA values

n_empty Count of empty strings ("")

n_whitespace_only Count of strings containing only whitespace

n_leading_ws Count of non-empty strings with leading whitespace

n_trailing_ws Count of non-empty strings with trailing whitespace

n_non_ascii Count of strings containing non-ASCII characters

n_case_variants Number of unique values that have case variants

case_variant_groups Number of groups of case-insensitive duplicates

case_variant_examples data.table with examples of case variants

See Also

`audit_clean()` for auditing the effect of cleaning functions, `diagnose_nas()` for missing value diagnostics

Examples

```
library(data.table)
firms <- c("Apple", "APPLE", "apple", "  Microsoft ", "Google", NA, "")
diagnose_strings(firms)
```

`embed_into_cartesian` *Embed Data into Cartesian Product Frame*

Description

Expands a data.table to include all combinations of the specified grouping variables (cartesian product). Missing combinations are filled with NA or a specified value.

Usage

```
embed_into_cartesian(dt, group_vars, dt_frame = NULL, fill = NA)
```

Arguments

`dt`	A data.table to expand.
`group_vars`	Character vector of column names defining the grouping structure.
`dt_frame`	Optional data.table containing the target cartesian frame. If NULL (default), creates cartesian product from unique values in dt.
`fill`	Value to fill for missing combinations. Default is NA. Set to NA to leave gaps unfilled.

Value

A data.table with all combinations of group_vars, with original data merged in.

Examples

```
library(data.table)
dt <- data.table(
  year = c(2020, 2020, 2021),
  region = c("A", "B", "A"),
  value = c(10, 20, 30)
)
embed_into_cartesian(dt, c("year", "region"))
```

`filter_drop` *Drop Data with Diagnostic Statistics*

Description

Filters a data.table by DROPPING rows where the expression is TRUE, while reporting statistics about dropped rows and optionally the sum of a statistic column that was dropped.

Usage

```
filter_drop(x, ...)

## S3 method for class 'data.table'
filter_drop(
  x,
  expr,
  stat = NULL,
  na_as = FALSE,
  quiet = FALSE,
  warn_threshold = NULL,
  ...
)
```

Arguments

`x`	A data.table or other object.
`...`	Arguments passed to methods.
`expr`	A filtering expression written in terms of columns of x. Rows where expr is TRUE are DROPPED; others are kept.
`stat`	An unquoted column or expression to total, e.g., sales, price*qty, etc. Reports the amount dropped and its share of total.
`na_as`	Logical. Treat NA results of expr as this value (default FALSE: drop rows where expr is NA).

quiet	Logical. If TRUE, suppress printing diagnostics.
warn_threshold	Numeric between 0 and 1. If set and the proportion of dropped rows exceeds this threshold, a warning is issued.

Value

The filtered data.table.

Methods (by class)

- filter_drop(data.table): Method for data.table objects

Examples

```
library(data.table)
DT <- data.table(
  id = 1:5,
  bad = c(FALSE, TRUE, FALSE, TRUE, FALSE),
  sales = 10:14
)

# Drop rows where bad == TRUE; report dropped statistics
DT2 <- filter_drop(DT, bad == TRUE)

# Also report dropped sales value
DT3 <- filter_drop(DT, bad == TRUE, stat = sales)
```

---

filter_keep *Filter Data with Diagnostic Statistics*

Description

Filters a data.table while reporting statistics about dropped rows and optionally the sum of a statistic column that was dropped.

Usage

```
filter_keep(x, ...)

## S3 method for class 'data.table'
filter_keep(
  x,
  expr,
  stat = NULL,
  na_as = FALSE,
  quiet = FALSE,
  warn_threshold = NULL,
  ...
)
```

Arguments

x	A data.table or other object.
...	Arguments passed to methods.
expr	A filtering expression written in terms of columns of x. Rows where expr is TRUE are KEPT; others are dropped.
stat	An unquoted column or expression to total, e.g., sales, price*qty, etc. Reports the amount dropped and its share of total.
na_as	Logical. Treat NA results of expr as this value (default FALSE: drop rows where expr is NA).
quiet	Logical. If TRUE, suppress printing diagnostics.
warn_threshold	Numeric between 0 and 1. If set and the proportion of dropped rows exceeds this threshold, a warning is issued.

Value

The filtered data.table.

Methods (by class)

- `filter_keep(data.table)`: Method for data.table objects

Examples

```
library(data.table)
DT <- data.table(
  id = 1:6,
  keep = c(TRUE, FALSE, TRUE, NA, TRUE, FALSE),
  sales = c(100, 50, 200, 25, NA, 75)
)

# Keep rows where keep == TRUE; report dropped statistics
DT2 <- filter_keep(DT, keep == TRUE)

# Also report dropped sales value
DT3 <- filter_keep(DT, keep == TRUE, stat = sales)
```

---

get_summary_table *Generate Summary Table for a Data Table*

Description

Creates a comprehensive summary of all columns in a data.table, including type, missing values, descriptive statistics, and example values.

Usage

```
get_summary_table(dt, cols = NULL)
```

Arguments

dt	A data.table to summarize.
cols	Optional character vector of column names to summarize. If NULL (the default), all columns are summarized.

Value

A data.table with one row per column containing summary statistics.

See Also

summarize_vector() for single-vector summaries, diagnose_nas() for missing value diagnostics

Examples

```
library(data.table)
dt <- data.table(
  id = 1:100,
  value = rnorm(100),
  category = sample(letters[1:5], 100, replace = TRUE)
)
get_summary_table(dt)
get_summary_table(dt, cols = c("value", "category"))
```

---

print.audit_clean *Print Method for audit_clean Objects*

Description

Print Method for audit_clean Objects

Usage

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

Arguments

x	An audit_clean object.
...	Additional arguments (ignored).

Value

Invisibly returns the input object.

print.compare_dt *Print Method for compare_dt Objects*

Description

Print Method for compare_dt Objects

Usage

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

Arguments

x	A compare_dt object from compare_datatables().
...	Additional arguments (ignored).

Value

The compare_dt object, invisibly.

print.diagnose_na *Print Method for diagnose_na Objects*

Description

Print Method for diagnose_na Objects

Usage

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

Arguments

x	A diagnose_na object from diagnose_nas().
...	Additional arguments (ignored).

Value

The diagnose_na object, invisibly.

print.diagnose_strings *Print Method for diagnose_strings Objects*

Description

Print Method for diagnose_strings Objects

Usage

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

Arguments

x	A diagnose_strings object.
...	Additional arguments (ignored).

Value

Invisibly returns the input object.

print.validate_join *Print Method for validate_join Objects*

Description

Displays a compact summary of join diagnostics.

Usage

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

Arguments

x	A validate_join object.
...	Additional arguments (ignored).

Value

Invisibly returns the input object.

print.validate_pk *Print Method for validate_pk Objects*

Description

Displays a compact summary of primary key validation results.

Usage

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

Arguments

x	A validate_pk object.
...	Additional arguments (ignored).

Value

Invisibly returns the input object.

print.validate_var_rel *Print Method for validate_var_rel Objects*

Description

Displays a compact summary of variable relationship validation results.

Usage

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

Arguments

x	A validate_var_rel object.
...	Additional arguments (ignored).

Value

Invisibly returns the input object.

summarize_vector *Summarize a Single Vector*

Description

Computes summary statistics for a vector. Handles numeric, character, factor, logical, Date, and other types with appropriate statistics for each.

Usage

```
summarize_vector(x)
```

Arguments

x	A vector to summarize.

Value

A named character vector with summary statistics including: type, unique count, missing count, most frequent value (for non-numeric), mean, sd, min, quartiles (q25, q50, q75), max, and three example values.

Examples

```
summarize_vector(c(1, 2, 3, NA, 5))
summarize_vector(c("a", "b", "a", "c"))
```

summary.validate_join *Summary Method for validate_join Objects*

Description

Returns the summary table from a validate_join object.

Usage

```
## S3 method for class 'validate_join'
summary(object, ...)
```

Arguments

object	A validate_join object.
...	Additional arguments (ignored).

Value

Invisibly returns the summary data.table.

validate_join *Validate Join Operations Between Two Data Tables*

Description

Analyzes a potential join between two data.tables without performing the full join between original tables. Reports relationship type (one-to-one, one-to-many, etc.), match rates, duplicate keys, and unmatched rows. Optionally tracks a numeric statistic column through the join to quantify impact.

Usage

```
validate_join(
  x,
  y,
  by = NULL,
  by.x = NULL,
  by.y = NULL,
  stat = NULL,
  stat.x = NULL,
  stat.y = NULL
)
```

Arguments

x	A data.table (left table).
y	A data.table (right table).
by	Character vector of column names to join on (used for both tables).
by.x	Character vector of column names in x to join on.
by.y	Character vector of column names in y to join on.
stat	Character string naming a numeric column present in both tables. Reports total, matched, and unmatched sums for each table.
stat.x	Character string naming a numeric column in x.
stat.y	Character string naming a numeric column in y.

Value

An S3 object of class validate_join containing:

x_name, y_name Names of the input tables from the original call

by.x, by.y Key columns used for the join

counts List with row counts, match rates, and overlap statistics

stat When stat, stat.x, or stat.y is provided, a list with elements stat_col_x and/or stat_col_y (column names) and sublists x and/or y each containing total, matched, only, rate, and n_na. NULL when no stat is provided.

duplicates List with duplicate key information for each table

summary_table A data.table summarizing the join diagnostics

relation Character string: "one-to-one", "one-to-many", "many-to-one", "many-to-many", or "no matches"

keys_only_in_x Keys present in x but not in y

keys_only_in_y Keys present in y but not in x

See Also

`validate_primary_keys()` for key uniqueness validation, `validate_var_relationship()` for variable relationship analysis, `compare_datatables()` for structural comparison

Examples

```
library(data.table)
dt1 <- data.table(id = c(1, 2, 3, 3), value = c("a", "b", "c", "d"))
dt2 <- data.table(id = c(2, 3, 4), score = c(10, 20, 30))
result <- validate_join(dt1, dt2, by = "id")
print(result)

# Track a numeric column through the join
orders <- data.table(id = 1:4, revenue = c(100, 200, 300, 400))
products <- data.table(id = 2:5, cost = c(10, 20, 30, 40))
validate_join(orders, products, by = "id", stat.x = "revenue", stat.y = "cost")
```

---

validate_primary_keys *Validate Primary Keys*

Description

Tests whether a set of columns constitute primary keys of a data.table, i.e., whether they uniquely identify every row in the table.

Usage

```
validate_primary_keys(dt, keys)
```

Arguments

dt	A data.table.
keys	Character vector of column names to test as primary keys.

Details

A warning is issued if any key column is numeric (double), as floating-point values can cause unexpected behavior in exact matching operations.

Value

An S3 object of class `validate_pk` containing:

table_name Name of the input table from the original call

keys Character vector of column names tested

is_primary_key Logical: TRUE if keys uniquely identify all rows

n_rows Total number of rows in the table

n_unique_keys Number of distinct key combinations

n_duplicate_keys Number of key combinations that appear more than once

duplicate_keys A data.table of duplicated key values with their counts

has_numeric_keys Logical: TRUE if any key column is of type double

See Also

`validate_join()` for join relationship analysis, `validate_var_relationship()` for variable relationship analysis

Examples

```
library(data.table)
dt <- data.table(
  id = c(1L, 2L, 3L, 4L),
  group = c("A", "A", "B", "B"),
  value = c(10, 20, 30, 40)
)

# Single column that IS a primary key
validate_primary_keys(dt, "id")

# Single column that is NOT a primary key
validate_primary_keys(dt, "group")

# Composite key that IS a primary key
validate_primary_keys(dt, c("group", "id"))
```

`validate_var_relationship`

Validate Variable Relationship

Description

Determines the relationship between two variables in a data.table: one-to-one, one-to-many, many-to-one, or many-to-many.

Usage

```
validate_var_relationship(dt, var1, var2)
```

Arguments

dt	A data.table.
var1	Character string: name of the first variable.
var2	Character string: name of the second variable.

Details

This function only accepts variables of type character, integer, or factor. Numeric (double) variables are not allowed due to potential floating-point comparison issues.

The relationship is determined as follows:

- **one-to-one**: Each value of var1 maps to exactly one value of var2, and vice versa.
- **one-to-many**: Each value of var1 maps to exactly one value of var2, but some var2 values map to multiple var1 values.
- **many-to-one**: Some var1 values map to multiple var2 values, but each var2 value maps to exactly one var1 value.
- **many-to-many**: Both variables have values that map to multiple values of the other.

Value

An S3 object of class `validate_var_rel` containing:

table_name Name of the input table from the original call

var1, var2 Names of the variables analyzed

relation Character string: "one-to-one", "one-to-many", "many-to-one", or "many-to-many"

var1_unique Number of distinct values in var1

var2_unique Number of distinct values in var2

n_combinations Number of unique (var1, var2) pairs

var1_has_dups Logical: does any var1 value map to multiple var2 values?

var2_has_dups Logical: does any var2 value map to multiple var1 values?

See Also

`validate_primary_keys()` for key uniqueness validation, `validate_join()` for join relationship analysis

Examples

```
library(data.table)
dt <- data.table(
  person_id = c(1L, 2L, 3L, 4L),
  department = c("Sales", "Sales", "Engineering", "Engineering"),
  country = c("US", "US", "US", "UK")
)

# Many-to-one: multiple persons per department
validate_var_relationship(dt, "person_id", "department")

# Many-to-many: departments and countries have complex mapping
validate_var_relationship(dt, "department", "country")
```

Index