Package 'joinspy'

June 13, 2026

Title Diagnostic Tools for Data Frame Joins

Version 0.8.3

Language en-US

Description Provides diagnostic tools for understanding and debugging data frame joins. Analyzes key columns before joining to detect duplicates, mismatches, encoding issues, and other common problems. Explains unexpected row count changes and provides safe join wrappers with cardinality enforcement. Concepts and diagnostics build on tidy data principles as described in 'Wickham' (2014) <doi:10.18637/jss.v059.i10>.

License MIT + file LICENSE

Encoding UTF-8

Imports cli, rlang

Suggests dplyr, data.table, tibble, testthat (>= 3.0.0), knitr, rmarkdown, shiny, miniUI

VignetteBuilder knitr

Config/testthat/edition 3

URL https://gillescolling.com/joinspy/, https://github.com/gcol33/joinspy

BugReports https://github.com/gcol33/joinspy/issues

Depends R (>= 4.1)

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Gilles Colling [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-3070-6066>)

Maintainer Gilles Colling <gilles.colling051@gmail.com>

Repository CRAN

Date/Publication 2026-06-13 16:10:18 UTC

Contents

`analyze_join_chain` *Analyze Multi-Table Join Chain*

Description

Analyzes a sequence of joins to identify potential issues in the chain. Useful for debugging complex multi-table joins.

Usage

```
analyze_join_chain(tables, joins)
```

Arguments

`tables`	A named list of data frames to join.
`joins`	A list of join specifications, each with elements: **left** Name of left table **right** Name of right table **by** Join column(s)

Value

A summary of the join chain analysis.

See Also

join_spy(), check_cartesian()

Examples

```
orders <- data.frame(order_id = 1:3, customer_id = c(1, 2, 2))
customers <- data.frame(customer_id = 1:3, region_id = c(1, 1, 2))
regions <- data.frame(region_id = 1:2, name = c("North", "South"))

analyze_join_chain(
  tables = list(orders = orders, customers = customers, regions = regions),
  joins = list(
    list(left = "orders", right = "customers", by = "customer_id"),
    list(left = "result", right = "regions", by = "region_id")
  )
)
```

---

check_cartesian *Detect Potential Cartesian Product*

Description

Warns if a join will produce a very large result due to many-to-many relationships (Cartesian product explosion).

Usage

```
check_cartesian(x, y, by, threshold = 10)
```

Arguments

x	A data frame (left table).
y	A data frame (right table).
by	Column names to join by.
threshold	Warn if result will exceed this many times the larger input. Default 10.

Value

A list with explosion analysis.

See Also

join_spy(), join_strict()

Examples

```
# Dangerous: both tables have duplicates
x <- data.frame(id = c(1, 1, 2, 2), val_x = 1:4)
y <- data.frame(id = c(1, 1, 2, 2), val_y = 1:4)

check_cartesian(x, y, by = "id")
```

`detect_cardinality` *Detect Join Relationship Type*

Description

Determines the actual cardinality relationship between two tables.

Usage

```
detect_cardinality(x, y, by)
```

Arguments

x	A data frame (left table).
y	A data frame (right table).
by	Column names to join by.

Value

Character string: "1:1", "1:n", "n:1", or "n:m".

See Also

`join_strict()`, `join_spy()`

Examples

```
# 1:1 relationship
x <- data.frame(id = 1:3, val = 1:3)
y <- data.frame(id = 1:3, name = c("A", "B", "C"))
detect_cardinality(x, y, "id")

# 1:n relationship
x <- data.frame(id = 1:3, val = 1:3)
y <- data.frame(id = c(1, 1, 2, 3), name = c("A1", "A2", "B", "C"))
detect_cardinality(x, y, "id")
```

full_join_spy *Full Join with Diagnostics*

Description

Performs a full join and automatically prints diagnostic information.

Usage

```
full_join_spy(x, y, by, verbose = TRUE, .quiet = FALSE, backend = NULL, ...)
```

Arguments

x	A data frame (left table).
y	A data frame (right table).
by	A character vector of column names to join by.
verbose	Logical. If TRUE (default), prints diagnostic summary.
.quiet	Logical. If TRUE, suppresses all output (overrides verbose). Useful for silent pipeline operations. Use last_report() to access the diagnostics afterward.
backend	Character or NULL. The join backend to use. If NULL (default), auto-detects from input class: data.table inputs use data.table, tibble inputs use dplyr, otherwise base R merge(). Explicit values: "base", "dplyr", "data.table".
...	Additional arguments passed to the underlying join function.

Value

The joined data frame with a "join_report" attribute.

See Also

left_join_spy(), join_spy(), last_report()

get_log_file *Get Current Log File*

Description

Returns the current automatic log file path, if set.

Usage

```
get_log_file()
```

Value

The log file path, or NULL if not set.

See Also

set_log_file()

---

inner_join_spy *Inner Join with Diagnostics*

Description

Performs an inner join and automatically prints diagnostic information.

Usage

```
inner_join_spy(x, y, by, verbose = TRUE, .quiet = FALSE, backend = NULL, ...)
```

Arguments

x	A data frame (left table).
y	A data frame (right table).
by	A character vector of column names to join by.
verbose	Logical. If TRUE (default), prints diagnostic summary.
.quiet	Logical. If TRUE, suppresses all output (overrides verbose). Useful for silent pipeline operations. Use last_report() to access the diagnostics afterward.
backend	Character or NULL. The join backend to use. If NULL (default), auto-detects from input class: data.table inputs use data.table, tibble inputs use dplyr, otherwise base R merge(). Explicit values: "base", "dplyr", "data.table".
...	Additional arguments passed to the underlying join function.

Value

The joined data frame with a "join_report" attribute.

See Also

left_join_spy(), join_spy(), last_report()

is_join_report *Check if Object is a JoinReport*

Description

Check if Object is a JoinReport

Usage

```
is_join_report(x)
```

Arguments

x	An object to test.

Value

TRUE if x is a JoinReport, FALSE otherwise.

join_diff *Compare Data Frame Before and After Join*

Description

Shows a side-by-side comparison of key statistics before and after a join operation.

Usage

```
join_diff(before, after, by = NULL)
```

Arguments

before	The original data frame (before joining).
after	The result data frame (after joining).
by	Optional. Column names to analyze for key statistics.

Value

Invisibly returns a comparison summary. Prints a formatted comparison.

See Also

join_explain(), join_spy()

Examples

```
before <- data.frame(id = 1:3, x = letters[1:3])
after <- data.frame(id = c(1, 2, 2, 3), x = c("a", "b", "b", "c"), y = 1:4
)

join_diff(before, after)
```

`join_explain` *Explain Row Count Changes After a Join*

Description

After performing a join, use this function to understand why the row count changed. It analyzes the original tables and the result to explain the difference.

Usage

```
join_explain(result, x, y, by, type = NULL)
```

Arguments

`result`	The result of a join operation.
`x`	The original left data frame.
`y`	The original right data frame.
`by`	A character vector of column names used in the join.
`type`	Character. The type of join that was performed. One of `"left"`, `"right"`, `"inner"`, `"full"`. If `NULL` (default), attempts to infer the join type from row counts.

Value

Invisibly returns a list with explanation details. Prints a human-readable explanation.

See Also

`join_spy()`, `join_diff()`

Examples

```
orders <- data.frame(id = c(1, 2, 2, 3), value = 1:4)
customers <- data.frame(id = c(1, 2, 2, 4), name = c("A", "B1", "B2", "D"))

result <- merge(orders, customers, by = "id", all.x = TRUE)

# Explain why we got more rows than expected
join_explain(result, orders, customers, by = "id", type = "left")
```

`join_repair` *Repair Common Key Issues*

Description

Automatically fixes trivial join key issues like whitespace and case mismatches. Returns the repaired data frame(s) with a summary of changes.

Usage

```
join_repair(
  x,
  y = NULL,
  by,
  trim_whitespace = TRUE,
  standardize_case = NULL,
  remove_invisible = TRUE,
  empty_to_na = FALSE,
  dry_run = FALSE
)
```

Arguments

`x`	A data frame (left table).
`y`	A data frame (right table). If NULL, only repairs x.
`by`	A character vector of column names to repair.
`trim_whitespace`	Logical. Trim leading/trailing whitespace. Default TRUE.
`standardize_case`	Character. Standardize case to "lower", "upper", or NULL (no change). Default NULL.
`remove_invisible`	Logical. Remove invisible Unicode characters. Default TRUE.
`empty_to_na`	Logical. Convert empty strings to NA. Default FALSE.
`dry_run`	Logical. If TRUE, only report what would be changed without modifying data. Default FALSE.

Value

If `y` is NULL, returns the repaired `x`. If both are provided, returns a list with `x` and `y`. In dry_run mode, returns a summary of proposed changes.

See Also

`join_spy()`, `key_check()`

Examples

```
# Data with whitespace issues
orders <- data.frame(
  id = c(" A", "B ", "C"),
  value = 1:3,
  stringsAsFactors = FALSE
)

# Dry run to see what would change
join_repair(orders, by = "id", dry_run = TRUE)

# Actually repair
orders_fixed <- join_repair(orders, by = "id")
```

---

join_spy *Comprehensive Pre-Join Diagnostic Report*

Description

Analyzes two data frames before joining to detect potential issues and predict the outcome. Returns a detailed report of key quality, match rates, and detected problems.

Usage

```
join_spy(x, y, by, sample = NULL, ...)
```

Arguments

x	A data frame (left table in the join).
y	A data frame (right table in the join).
by	A character vector of column names to join by, or a named character vector for joins where column names differ (e.g., `c("id" = "customer_id")`).
sample	Integer or NULL. If provided, randomly sample this many rows from each table for faster diagnostics on large datasets. Default NULL (analyze all rows).
...	Reserved for future use.

Details

This function detects the following common join issues:

- **Duplicate keys**: Keys appearing multiple times, which cause row multiplication during joins
- **Whitespace**: Leading or trailing spaces that prevent matches
- **Case mismatches**: Keys that differ only by case (e.g., "ABC" vs "abc")
- **Encoding issues**: Different character encodings or invisible Unicode characters
- **NA values**: Missing values in key columns

Value

A `JoinReport` object with the following components:

x_summary Summary statistics for keys in the left table

y_summary Summary statistics for keys in the right table

match_analysis Details of which keys will/won't match

issues List of detected problems (duplicates, whitespace, etc.)

expected_rows Predicted row counts for each join type

memory_estimate Heuristic estimate of the result size for each join type, as human-readable strings

See Also

`key_check()`, `join_explain()`, `join_strict()`

Examples

```
# Create sample data with issues
orders <- data.frame(
  order_id = 1:5,
  customer_id = c("A", "B", "B", "C", "D ")
)
customers <- data.frame(
  customer_id = c("A", "B", "C", "E"),
  name = c("Alice", "Bob", "Carol", "Eve")
)

# Get diagnostic report
join_spy(orders, customers, by = "customer_id")
```

`join_strict` *Strict Join with Cardinality Enforcement*

Description

Performs a join operation that fails if the specified cardinality constraint is violated. Use this to catch unexpected many-to-many relationships early.

Usage

```
join_strict(
  x,
  y,
  by,
  type = c("left", "right", "inner", "full"),
  expect = c("1:1", "1:n", "1:many", "n:1", "many:1", "n:m", "many:many"),
```

```
  backend = NULL,
  ...
)
```

Arguments

x	A data frame (left table).
y	A data frame (right table).
by	A character vector of column names to join by.
type	Character. The type of join to perform. One of `"left"` (default), `"right"`, `"inner"`, `"full"`.
expect	Character. The expected cardinality relationship. One of: **"1:1"** Each key in x matches at most one key in y, and vice versa **"1:n" or "1:many"** Each key in x can match multiple keys in y, but each key in y matches at most one key in x **"n:1" or "many:1"** Each key in y can match multiple keys in x, but each key in x matches at most one key in y **"n:m" or "many:many"** No cardinality constraints (allows all relationships)
backend	Character or `NULL`. The join backend to use. If `NULL` (default), auto-detects from input class. See `left_join_spy()` for details.
...	Additional arguments passed to the underlying join function.

Value

The joined data frame if the cardinality constraint is satisfied. Throws an error if the constraint is violated.

See Also

`join_spy()`, `left_join_spy()`

Examples

```
orders <- data.frame(id = 1:3, product = c("A", "B", "C"))
customers <- data.frame(id = 1:3, name = c("Alice", "Bob", "Carol"))

# This succeeds (1:1 relationship)
join_strict(orders, customers, by = "id", expect = "1:1")

# This fails if customers had duplicate ids (wrapped in try to show error)
customers_dup <- data.frame(id = c(1, 1, 2), name = c("A1", "A2", "B"))
try(join_strict(orders, customers_dup, by = "id", expect = "1:1"))
```

key_check *Quick Key Quality Check*

Description

A fast check of join key quality that returns a simple pass/fail status with a brief summary. Use this for quick validation; use `join_spy()` for detailed diagnostics.

Usage

```
key_check(x, y, by, warn = TRUE)
```

Arguments

x	A data frame (left table in the join).
y	A data frame (right table in the join).
by	A character vector of column names to join by.
warn	Logical. If TRUE (default), prints warnings for detected issues. Set to FALSE for silent operation.

Value

Invisibly returns a logical: TRUE if no issues detected, FALSE otherwise. Also prints a brief status message unless warn = FALSE.

See Also

`join_spy()`, `key_duplicates()`

Examples

```
orders <- data.frame(id = c(1, 2, 2, 3), value = 1:4)
customers <- data.frame(id = c(1, 2, 4), name = c("A", "B", "D"))

# Quick check
key_check(orders, customers, by = "id")

# Silent check
is_ok <- key_check(orders, customers, by = "id", warn = FALSE)
```

key_duplicates *Find Duplicate Keys*

Description

Identifies rows with duplicate values in the specified key columns. Returns a data frame containing only the rows with duplicated keys, along with a count of occurrences.

Usage

```
key_duplicates(data, by, keep = c("all", "first", "last"))
```

Arguments

data	A data frame.
by	A character vector of column names to check for duplicates.
keep	Character. Which duplicates to return: **"all"** Return all rows with duplicated keys (default) **"first"** Return only the first occurrence of each duplicate **"last"** Return only the last occurrence of each duplicate

Value

A data frame containing the duplicated rows, with an additional column `.n_duplicates` showing how many times each key appears. Returns an empty data frame (0 rows) if no duplicates found.

See Also

key_check(), join_spy()

Examples

```
df <- data.frame(
  id = c(1, 2, 2, 3, 3, 3, 4),
  value = letters[1:7]
)

# Find all duplicates
key_duplicates(df, by = "id")

# Find first occurrence only
key_duplicates(df, by = "id", keep = "first")
```

last_report *Get the Last Join Report*

Description

Retrieves the most recent `JoinReport` object from any `*_join_spy()` call. Useful when using `.quiet = TRUE` in pipelines and wanting to inspect the diagnostics afterward.

Usage

```
last_report()
```

Value

The last `JoinReport` object, or `NULL` if no join has been performed.

See Also

`left_join_spy()`, `join_spy()`

Examples

```
orders <- data.frame(id = 1:3, value = c(10, 20, 30))
customers <- data.frame(id = c(1, 2, 4), name = c("A", "B", "D"))

# Silent join in a pipeline
result <- left_join_spy(orders, customers, by = "id", .quiet = TRUE)

# Inspect the report afterward
last_report()
```

left_join_spy *Left Join with Diagnostics*

Description

Performs a left join and automatically prints diagnostic information about the operation. The diagnostic report is also attached as an attribute.

Usage

```
left_join_spy(x, y, by, verbose = TRUE, .quiet = FALSE, backend = NULL, ...)
```

Arguments

x	A data frame (left table).
y	A data frame (right table).
by	A character vector of column names to join by.
verbose	Logical. If TRUE (default), prints diagnostic summary.
.quiet	Logical. If TRUE, suppresses all output (overrides verbose). Useful for silent pipeline operations. Use last_report() to access the diagnostics afterward.
backend	Character or NULL. The join backend to use. If NULL (default), auto-detects from input class: data.table inputs use data.table, tibble inputs use dplyr, otherwise base R merge(). Explicit values: "base", "dplyr", "data.table".
...	Additional arguments passed to the underlying join function.

Value

The joined data frame with a "join_report" attribute containing the diagnostic information.

See Also

join_spy(), join_strict(), last_report()

Examples

```
orders <- data.frame(id = 1:3, value = c(10, 20, 30))
customers <- data.frame(id = c(1, 2, 4), name = c("A", "B", "D"))

result <- left_join_spy(orders, customers, by = "id")

# Access the diagnostic report
attr(result, "join_report")

# Silent mode for pipelines
result2 <- left_join_spy(orders, customers, by = "id", .quiet = TRUE)
last_report()  # Access diagnostics afterward
```

---

log_report *Log Join Report to File*

Description

Writes a JoinReport object to a file for audit trails and reproducibility. Supports plain text, JSON, and RDS formats.

Usage

```
log_report(report, file, append = FALSE, timestamp = TRUE)
```

Arguments

report	A JoinReport object from join_spy() or retrieved via last_report().
file	File path to write to. Extension determines format: • .txt or .log: Plain text (human-readable) • .json: JSON format (machine-readable) • .rds: R binary format (preserves all data)
append	Logical. If TRUE, appends to existing file (text/log only). Default FALSE.
timestamp	Logical. If TRUE (default), includes timestamp in output.

Value

Invisibly returns the file path.

See Also

join_spy(), last_report()

Examples

```
orders <- data.frame(id = 1:3, value = c(10, 20, 30))
customers <- data.frame(id = c(1, 2, 4), name = c("A", "B", "D"))

report <- join_spy(orders, customers, by = "id")

# Log to temporary file
tmp <- tempfile(fileext = ".log")
log_report(report, tmp, append = TRUE)
unlink(tmp)
```

---

plot.JoinReport *Plot Method for JoinReport*

Description

Creates a Venn diagram showing key overlap between tables.

Usage

```
## S3 method for class 'JoinReport'
plot(
  x,
  file = NULL,
  width = 6,
  height = 5,
  colors = c("#4A90D9", "#D94A4A"),
  ...
)
```

Arguments

x	A JoinReport object.
file	Optional file path to save the plot (PNG, SVG, or PDF based on extension). If NULL (default), displays in the current graphics device.
width	Width in inches (default 6).
height	Height in inches (default 5).
colors	Character vector of length 2 for left and right circle colors.
...	Additional arguments (ignored).

Value

Invisibly returns the plot data (left_only, both, right_only counts).

Examples

```
orders <- data.frame(id = 1:5, val = 1:5)
customers <- data.frame(id = 3:7, name = letters[3:7])

report <- join_spy(orders, customers, by = "id")
plot(report)
```

---

print.JoinReport *Print Method for JoinReport*

Description

Print Method for JoinReport

Usage

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

Arguments

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

Value

Invisibly returns x.

right_join_spy *Right Join with Diagnostics*

Description

Performs a right join and automatically prints diagnostic information.

Usage

```
right_join_spy(x, y, by, verbose = TRUE, .quiet = FALSE, backend = NULL, ...)
```

Arguments

x	A data frame (left table).
y	A data frame (right table).
by	A character vector of column names to join by.
verbose	Logical. If TRUE (default), prints diagnostic summary.
.quiet	Logical. If TRUE, suppresses all output (overrides verbose). Useful for silent pipeline operations. Use last_report() to access the diagnostics afterward.
backend	Character or NULL. The join backend to use. If NULL (default), auto-detects from input class: data.table inputs use data.table, tibble inputs use dplyr, otherwise base R merge(). Explicit values: "base", "dplyr", "data.table".
...	Additional arguments passed to the underlying join function.

Value

The joined data frame with a "join_report" attribute.

See Also

left_join_spy(), join_spy(), last_report()

set_log_file *Configure Automatic Logging*

Description

Sets up automatic logging of all join reports to a specified file. When enabled, every *_join_spy() call will append its report to the log.

Usage

```
set_log_file(file, format = c("text", "json"))
```

Arguments

file	File path for automatic logging. Set to NULL to disable.
format	Log format: "text" (default) or "json".

Value

Invisibly returns the previous log file setting.

See Also

log_report(), get_log_file()

Examples

```
# Enable automatic logging to temp file
tmp <- tempfile(fileext = ".log")
old <- set_log_file(tmp)

# Disable logging and clean up
set_log_file(NULL)
unlink(tmp)
```

---

suggest_repairs *Suggest Repair Code*

Description

Analyzes join issues and returns R code snippets to fix them.

Usage

```
suggest_repairs(report)
```

Arguments

report	A JoinReport object from join_spy().

Value

Character vector of R code snippets to fix detected issues.

See Also

join_repair(), join_spy()

Examples

```
orders <- data.frame(id = c("A ", "B"), val = 1:2, stringsAsFactors = FALSE)
customers <- data.frame(id = c("a", "b"), name = c("Alice", "Bob"), stringsAsFactors = FALSE)

report <- join_spy(orders, customers, by = "id")
suggest_repairs(report)
```

---

summary.JoinReport *Summary Method for JoinReport*

Description

Returns a compact summary data frame of the join diagnostic report.

Usage

```
## S3 method for class 'JoinReport'
summary(object, format = c("data.frame", "text", "markdown"), ...)
```

Arguments

object	A JoinReport object.
format	Output format: "data.frame" (default), "text", or "markdown".
...	Additional arguments (ignored).

Value

A data frame with key metrics (or printed output for text/markdown).

Examples

```
orders <- data.frame(id = 1:5, val = 1:5)
customers <- data.frame(id = 3:7, name = letters[3:7])

report <- join_spy(orders, customers, by = "id")
summary(report)
summary(report, format = "markdown")
```

Index