Package 'clinCompare'

February 18, 2026

Type Package

Title Dataset Comparison with 'CDISC' Validation for Clinical Trial Data

Version 1.0.0

Description A general-purpose toolkit for comparing any two data frames with optional 'CDISC' (Clinical Data Interchange Standards Consortium) validation for clinical trial data. Core comparison functions work on arbitrary datasets: variable-level and observation-level comparison, data type checking, metadata attribute analysis (types, labels, lengths, formats), missing value handling, key-based row matching, tolerance-based numeric comparisons, and group-wise comparisons. Optional z-score outlier detection is available when enabled. When working with clinical data, the package additionally validates 'SDTM' (Study Data Tabulation Model) and 'ADaM' (Analysis Data Model) datasets against CDISC standards (SDTM IG 3.3/3.4, ADaM IG 1.1/1.2/1.3), automatically detecting domains and flagging non-conformant variables. Generates unified comparison reports in text or HTML format with interactive dashboards. For CDISC standards, see <https://www.cdisc.org/standards>.

License MIT + file LICENSE

URL https://github.com/siddharthlokineni/clinCompare

BugReports https://github.com/siddharthlokineni/clinCompare/issues

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 3.5.0)

Imports dplyr (>= 1.0.0), haven (>= 2.0.0), rlang (>= 0.4.0), tidyr (>= 1.0.0), methods, stats, tools, utils

Suggests ggplot2 (>= 3.0.0), openxlsx (>= 4.0.0), testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Siddharth Lokineni [aut, cre]

Maintainer Siddharth Lokineni <sidhu871@gmail.com>

Repository CRAN

Date/Publication 2026-02-18 19:00:07 UTC

Contents

clinCompare-package 2
cdisc_compare 3
clean_dataset 6
compare_by_group 7
compare_datasets 7
compare_observations 9
compare_submission 10
compare_variables 11
detect_cdisc_domain 11
export_report 12
generate_cdisc_report 14
generate_detailed_report 15
generate_summary_report 16
get_all_differences 17
prepare_datasets 18
print.cdisc_comparison 19
print.dataset_comparison 19
print_cdisc_validation 20
summary.cdisc_comparison 21
validate_cdisc 21

Index **23**

`clinCompare-package` *clinCompare: Dataset Comparison with CDISC Validation*

Description

A comprehensive toolkit for comparing clinical trial datasets. Provides functions for dataset comparison including variable-level and observation-level differences, data type checking, and missing value analysis. Integrates CDISC validation for SDTM and ADaM datasets.

Main Functions

`compare_datasets` High-level comparison of two datasets

`compare_variables` Compare variable names and types

`compare_observations` Row-wise value comparison

`cdisc_compare` Compare datasets with CDISC validation

`validate_cdisc` Validate a dataset against CDISC standards

`detect_cdisc_domain` Auto-detect CDISC domain or ADaM dataset

CDISC Standards Supported

SDTM DM, AE, LB, VS, EX, CM, MH, DS, SV, TA, TE domains

ADaM ADSL, ADAE, ADLB, ADTTE, ADEFF datasets

Author(s)

Maintainer: Siddharth Lokineni <sidhu871@gmail.com>

See Also

Useful links:

- https://github.com/siddharthlokineni/clinCompare
- Report bugs at https://github.com/siddharthlokineni/clinCompare/issues

---

cdisc_compare *Compare Two Datasets with CDISC Validation*

Description

Flagship function that compares two datasets AND runs CDISC validation on both. Combines dataset comparison with CDISC conformance analysis to provide comprehensive insights into both differences and regulatory compliance.

Usage

```
cdisc_compare(
  df1,
  df2,
  domain = NULL,
  standard = NULL,
  id_vars = NULL,
  vars = NULL,
  ts_data = NULL,
  detect_outliers = FALSE,
  tolerance = 0,
  where = NULL
)
```

Arguments

df1	First data frame to compare, or a file path (character string ending in .xpt, .sas7bdat, .csv, or .rds). When a file path is provided, the dataset is loaded automatically. Domain is auto-detected from filename if not specified (e.g., "dm.xpt" sets domain to "DM").
df2	Second data frame to compare, or a file path.
domain	Optional character string specifying the CDISC domain code or dataset name (e.g., "DM", "AE", "ADSL"). Strongly recommended – auto-detection can be ambiguous for datasets with common columns. If NULL, auto-detected from df1.
standard	Optional character string: "SDTM" or "ADaM". If NULL, auto-detected from df1.
id_vars	Optional character vector of ID variable names (e.g., c("USUBJID", "VISITNUM")) used to match rows between datasets. When provided, rows are joined by these keys instead of matched by position. Unmatched rows are reported separately. When NULL (default) and domain is known, CDISC-standard keys are auto-detected (e.g., STUDYID + USUBJID + \<DOMAIN\>SEQ for SDTM). Only variables present in both datasets are used. To add extra keys on top of the defaults, prefix with "+": e.g., id_vars = c("+", "AETOXGR") appends AETOXGR to the standard keys. To override completely, pass without "+".
vars	Optional character vector of variable names to compare. Only these columns are included in value comparison. Structural and CDISC validation still covers all columns.
ts_data	Optional data frame of the TS (Trial Summary) domain. When provided, CDISC standard versions (e.g., SDTM IG 3.4, ADaM IG 1.3) are extracted and included in the results and reports. If NULL (default), version information is omitted.
detect_outliers	Logical. When TRUE, runs z-score outlier detection on numeric columns and includes results in the output. Defaults to FALSE.
tolerance	Numeric tolerance value for floating-point comparisons (default 0). When tolerance > 0, numeric values are considered equal if their absolute difference is within the tolerance threshold. Character and factor columns always use exact matching regardless of tolerance.
where	Optional filter expression as a string (e.g., "AESEV == 'SEVERE'"). Applied to both datasets before comparison. Equivalent to a WHERE clause.

Value

A list containing:

domain	Character: detected or supplied CDISC domain
standard	Character: detected or supplied CDISC standard (SDTM/ADaM)
nrow_df1	Integer: number of rows in df1
ncol_df1	Integer: number of columns in df1
nrow_df2	Integer: number of rows in df2

`ncol_df2` Integer: number of columns in df2

`id_vars` Character vector of ID variables used for matching (NULL if positional matching was used)

`comparison` Result of `compare_datasets()` function

`variable_comparison` Result of `compare_variables()` function

`metadata_comparison` List of metadata differences: type_mismatches, label_mismatches, length_mismatches, format_mismatches, column ordering

`observation_comparison` Result of `compare_observations()` if dimensions match, otherwise NULL with explanatory message

`unified_comparison` Data frame combining attribute and value differences per variable. Columns: variable, attribute, base_value, compare_value, and optionally id columns and row when value differences exist

`unmatched_rows` List with df1_only and df2_only data frames of rows that could not be matched by id_vars (NULL when id_vars is not used)

`cdisc_validation_df1` CDISC validation results for df1

`cdisc_validation_df2` CDISC validation results for df2

`cdisc_conformance_comparison` Data frame showing which CDISC issues are unique to df1, unique to df2, or common to both

`outlier_notes` Data frame of z-score outliers ($|z| > 3$) found in numeric columns of either dataset (NULL when detect_outliers is FALSE)

`cdisc_version` List of CDISC version information extracted from TS domain (NULL when ts_data is not provided). See `extract_cdisc_version()`

Examples

```
# Create sample SDTM DM domains
dm1 <- data.frame(
  STUDYID = "STUDY001",
  USUBJID = c("SUBJ001", "SUBJ002"),
  DMSEQ = c(1, 1),
  RACE = c("WHITE", "BLACK OR AFRICAN AMERICAN"),
  stringsAsFactors = FALSE
)

dm2 <- data.frame(
  STUDYID = "STUDY001",
  USUBJID = c("SUBJ001", "SUBJ003"),
  DMSEQ = c(1, 1),
  RACE = c("WHITE", "ASIAN"),
  ETHNIC = c("NOT HISPANIC", "NOT HISPANIC"),
```

```
  stringsAsFactors = FALSE
)

# Positional matching (default)
result <- cdisc_compare(dm1, dm2, domain = "DM", standard = "SDTM")

# Key-based matching by ID variables
result <- cdisc_compare(dm1, dm2, domain = "DM", id_vars = c("USUBJID"))
names(result)
```

---

clean_dataset *Clean Dataset*

Description

Removes duplicate rows, standardizes column names and text values to uppercase or lowercase, and performs basic data cleaning on a data frame.

Usage

```
clean_dataset(
  df,
  variables = NULL,
  remove_duplicates = TRUE,
  convert_to_case = NULL
)
```

Arguments

df	A data frame to be cleaned.
variables	Optional; a vector of variable names to specifically clean. If NULL, applies cleaning to all variables.
remove_duplicates	Logical; whether to remove duplicate rows.
convert_to_case	Optional; convert character variables to "lower" or "upper" case.

Value

A cleaned data frame.

Examples

```
df <- data.frame(name = c("Alice", "Bob", "Alice"),
                 score = c(90, 85, 90),
                 stringsAsFactors = FALSE)
clean_dataset(df, remove_duplicates = TRUE, convert_to_case = "upper")
```

compare_by_group *Compare Two Datasets by Group*

Description

Compares two datasets within subgroups defined by grouping variables. Performs separate comparisons for each group and returns results organized by group.

Usage

```
compare_by_group(df1, df2, group_vars)
```

Arguments

df1	A data frame representing the first dataset.
df2	A data frame representing the second dataset.
group_vars	A character vector of column names to group by.

Value

A list of comparison results for each group.

Examples

```
df1 <- data.frame(region = c("A", "A", "B"), value = c(10, 20, 30),
                  stringsAsFactors = FALSE)
df2 <- data.frame(region = c("A", "A", "B"), value = c(10, 25, 30),
                  stringsAsFactors = FALSE)
compare_by_group(df1, df2, group_vars = "region")
```

compare_datasets *Compare Two Datasets*

Description

Compares two datasets at three levels in a single call:

1. **Dataset level** – dimensions, column overlap, missing-value totals.
2. **Variable level** – column name discrepancies and data-type mismatches (delegates to `compare_variables()`).
3. **Observation level** – row-by-row value differences on common columns. Uses positional matching by default, or key-based matching when `id_vars` is provided.

The return value is a list with class `"dataset_comparison"`, which has a tidy `print()` method. The same object is accepted by `generate_summary_report()`, `generate_detailed_report()`, and `compare_by_group()`.

Usage

```
compare_datasets(df1, df2, tolerance = 0, vars = NULL, id_vars = NULL)
```

Arguments

df1	A data frame (the *base* dataset).
df2	A data frame (the *compare* dataset).
tolerance	Numeric tolerance value for floating-point comparisons (default 0). When tolerance > 0, numeric values are considered equal if their absolute difference is within the tolerance threshold. Character and factor columns always use exact matching regardless of tolerance.
vars	Optional character vector of variable names to compare. When provided, only these columns are included in the observation-level comparison. Structural comparison (extra columns, type mismatches) still covers all columns. Default is NULL (compare all common columns).
id_vars	Optional character vector of column names to use as matching keys. When provided, rows are matched by these key columns instead of by position. This allows comparison of datasets with different row counts or different row orders. Rows that exist in only one dataset are reported in unmatched_rows. Default is NULL (positional matching).

Value

A dataset_comparison list containing:

nrow_df1, ncol_df1	Dimensions of df1.
nrow_df2, ncol_df2	Dimensions of df2.
common_columns	Character vector of columns present in both.
extra_in_df1	Columns only in df1.
extra_in_df2	Columns only in df2.
type_mismatches	Data frame of columns whose class differs (columns: column, type_df1, type_df2), or NULL if none.
missing_values	Data frame summarising NA counts per column per dataset (columns: column, na_df1, na_df2), or NULL if no missingness.
variable_comparison	Output of compare_variables().
observation_comparison	Output of compare_observations(), or a list with a message element when row counts differ.
id_vars	Character vector of key columns used for matching, or NULL if positional matching was used.
unmatched_rows	List with df1_only and df2_only data frames of rows with no match in the other dataset (key-based matching only), or NULL.

Examples

```
# Positional matching (default)
df1 <- data.frame(id = 1:3, val = c(10, 20, 30))
df2 <- data.frame(id = 1:3, val = c(10, 25, 30))
result <- compare_datasets(df1, df2)
result

# Key-based matching (for different row counts or row orders)
df1 <- data.frame(id = c(1, 2, 3), val = c(10, 20, 30))
df2 <- data.frame(id = c(2, 3, 4), val = c(20, 35, 40))
result <- compare_datasets(df1, df2, id_vars = "id")
result
result$unmatched_rows
```

---

compare_observations *Compare Observations of Two Datasets*

Description

Performs row-by-row comparison of two datasets on common columns, identifying specific value differences at the cell level. Returns discrepancy counts and details showing which rows differ and how their values diverge.

Usage

```
compare_observations(df1, df2, tolerance = 0)
```

Arguments

df1	A data frame representing the first dataset.
df2	A data frame representing the second dataset.
tolerance	Numeric tolerance value for floating-point comparisons (default 0). When tolerance > 0, numeric values are considered equal if their absolute difference is within the tolerance threshold. Character and factor columns always use exact matching regardless of tolerance.

Value

A list containing discrepancy counts and details of row differences.

Examples

```
df1 <- data.frame(id = 1:3, value = c(1.0, 2.0, 3.0))
df2 <- data.frame(id = 1:3, value = c(1.0, 2.5, 3.0))
compare_observations(df1, df2)
compare_observations(df1, df2, tolerance = 0.00001)
```

`compare_submission` *Batch Compare CDISC Datasets Across Submission Directories*

Description

Scans two directories for matching dataset files, runs `cdisc_compare()` on each pair, and optionally generates a consolidated Excel report.

Usage

```
compare_submission(
  base_dir,
  compare_dir,
  format = NULL,
  id_vars = NULL,
  tolerance = 0,
  output_file = NULL
)
```

Arguments

`base_dir`	Path to directory containing base/reference files.
`compare_dir`	Path to directory containing comparison files.
`format`	File format to match: "xpt", "sas7bdat", "csv", or "rds". When NULL (default), auto-detected from the most common file type in base_dir.
`id_vars`	Optional character vector of ID variables (passed to each comparison). When NULL, CDISC-standard keys are auto-detected per domain.
`tolerance`	Numeric tolerance for floating-point comparisons (default 0).
`output_file`	Optional path to Excel (.xlsx) file for consolidated report.

Value

Named list of cdisc_compare() results, one per matched domain.

Examples

```
## Not run:
  # Auto-detects format from directory contents
  results <- compare_submission("v1/", "v2/",
                                output_file = "submission_diff.xlsx")

  # Explicit format
  results <- compare_submission("v1/", "v2/", format = "csv")

## End(Not run)
```

compare_variables *Compare Variables of Two Datasets*

Description

Compares the structural attributes of two datasets including column names, data types, and variable ordering. Identifies common columns and reports columns that exist in only one dataset.

Usage

```
compare_variables(df1, df2)
```

Arguments

df1	A data frame representing the first dataset.
df2	A data frame representing the second dataset.

Value

A list containing variable comparison details and discrepancy count.

Examples

```
df1 <- data.frame(id = 1:3, name = c("A", "B", "C"))
df2 <- data.frame(id = 1:3, name = c("A", "B", "C"), score = c(90, 80, 70))
compare_variables(df1, df2)
```

detect_cdisc_domain *Detect CDISC Domain Type*

Description

Detects whether a data frame looks like an SDTM domain or ADaM dataset by comparing column names against known CDISC standards. Calculates a confidence score based on the percentage of expected variables present.

Auto-detection is a convenience for exploratory use. For anything important – validation reports, regulatory submissions, scripted pipelines – always pass `domain` and `standard` explicitly. Datasets with common columns (STUDYID, USUBJID, etc.) can match multiple domains, and a warning is issued when the top two candidates score within 10 percentage points of each other.

Usage

```
detect_cdisc_domain(df, name_hint = NULL)
```

Arguments

df	A data frame to analyze.
name_hint	Optional character string with the dataset name (e.g., "DM", "ADLB", or a filename like "adlb.xpt"). When provided and it matches a known CDISC domain, that candidate receives a strong confidence boost. This makes detection much more accurate when the filename is available.

Value

A list containing:

standard	Character: "SDTM", "ADaM", or "Unknown"
domain	Character: domain code (e.g., "DM", "AE") or dataset name (e.g., "ADSL"), or NA
confidence	Numeric between 0 and 1 indicating match quality
message	Character: human-readable explanation

Examples

```
# Create a sample SDTM DM domain
dm <- data.frame(
  STUDYID = "STUDY001",
  USUBJID = "SUBJ001",
  SUBJID = "001",
  DMSEQ = 1,
  RACE = "WHITE",
  ETHNIC = "NOT HISPANIC OR LATINO",
  ARMCD = "ARM01",
  ARM = "Treatment A",
  stringsAsFactors = FALSE
)

result <- detect_cdisc_domain(dm)
print(result)
```

---

export_report *Export Comparison Report to File*

Description

Exports a dataset or CDISC comparison result to a file in multiple formats. Automatically detects format from file extension (.html, .txt, .xlsx).

Usage

```
export_report(result, file, format = NULL)
```

Arguments

result	A list from compare_datasets() or cdisc_compare().
file	Character string specifying the output file path. File extension determines format: .html, .txt, or .xlsx.
format	Character string specifying output format: "html", "text", or "excel". If NULL (default), format is auto-detected from file extension.

Details

Supported formats:

- **HTML** (.html): Self-contained HTML report with styling and interactive charts.
- **Text** (.txt): Plain text report suitable for console review.
- **Excel** (.xlsx): Multi-sheet workbook with tabbed data:
  - "Summary": Dataset dimensions, domain, standard, matching type, tolerance
  - "Variable Diffs": Metadata attribute differences
  - "Value Diffs": Unified diff data frame from get_all_differences()
  - "CDISC Validation": Combined validation results (for CDISC comparisons only)

The result object can be either a dataset_comparison (from compare_datasets()) or cdisc_comparison (from cdisc_compare()). All features are supported for both.

Value

Invisibly returns the input result (useful for piping).

Examples

```
# Create sample datasets
df1 <- data.frame(
  ID = c(1, 2, 3),
  NAME = c("Alice", "Bob", "Charlie"),
  AGE = c(25, 30, 35)
)

df2 <- data.frame(
  ID = c(1, 2, 3),
  NAME = c("Alice", "Bob", "Charles"),
  AGE = c(25, 30, 36)
)

# Compare datasets
result <- compare_datasets(df1, df2)

# Export to different formats (write to tempdir)
export_report(result, file.path(tempdir(), "report.html"))
export_report(result, file.path(tempdir(), "report.txt"))

# Explicit format specification
```

```
export_report(result, file.path(tempdir(), "report.xlsx"), format = "excel")
```

`generate_cdisc_report` *Generate CDISC Validation Report*

Description

Generates a formatted report from the results of `cdisc_compare()`. Supports both text-based console output and HTML reports with professional styling and color-coding.

Usage

```
generate_cdisc_report(cdisc_results, output_format = "text", file_name = NULL)
```

Arguments

`cdisc_results`	A list output from `cdisc_compare()`.
`output_format`	Character string: either "text" (default) for console output or "html" for HTML report.
`file_name`	Optional character string specifying the output file path. For text format, the report is appended to this file. For HTML format, must be explicitly provided by the user. If NULL, output is not written to file.

Details

The report includes:

- Dataset Comparison Summary
- CDISC Compliance for each dataset
- CDISC Conformance Comparison

For text output, formatting uses console-friendly layout. For HTML output, a self-contained report is generated with color-coded severity levels: red for ERROR, orange for WARNING, blue for INFO.

Value

Invisibly returns the input `cdisc_results` (useful for piping).

Examples

```
## Not run:
# Create sample datasets
dm1 <- data.frame(
  STUDYID = "STUDY001",
  USUBJID = c("SUBJ001", "SUBJ002"),
  DMSEQ = c(1, 1),
  RACE = c("WHITE", "BLACK OR AFRICAN AMERICAN")
)

dm2 <- data.frame(
  STUDYID = "STUDY001",
  USUBJID = c("SUBJ001", "SUBJ003"),
  DMSEQ = c(1, 1),
  RACE = c("WHITE", "ASIAN")
)

result <- cdisc_compare(dm1, dm2, domain = "DM")

# Generate text report to console
generate_cdisc_report(result, output_format = "text")

# Generate HTML report to file
out <- file.path(tempdir(), "report.html")
generate_cdisc_report(result, output_format = "html", file_name = out)

## End(Not run)
```

---

generate_detailed_report

Generate a Detailed Report of Dataset Comparison

Description

Creates a detailed report outlining all the differences found in the comparison, including variable differences, observation differences, and group-based discrepancies.

Usage

```
generate_detailed_report(
  comparison_results,
  output_format = "text",
  file_name = NULL
)
```

Arguments

`comparison_results`
: A list containing the results of dataset comparisons.

output_format	Format of the output ('text' or 'html').
file_name	Name of the file to save the report to (applicable for 'html' format).

Value

The detailed report. For 'text', prints to console. For 'html', writes to file.

Examples

```
## Not run:
  generate_detailed_report(comparison_results, output_format = "text")

## End(Not run)
```

---

generate_summary_report

Generate a Summary Report of Dataset Comparison

Description

Provides a summary of the comparison results, highlighting key points such as the number of differing observations and variables.

Usage

```
generate_summary_report(
  comparison_results,
  detail_level = "high",
  output_format = "text",
  file_name = NULL
)
```

Arguments

comparison_results	A list containing the results of dataset comparisons.
detail_level	The level of detail ('high', 'medium', 'low') for the summary.
output_format	Format of the output ('text' or 'html').
file_name	Name of the file to save the report to (applicable for 'html' format).

Value

The summary report. For 'text', prints to console. For 'html', writes to file.

Examples

```
## Not run:
 generate_summary_report(comparison_results, detail_level = "high", output_format = "text")

## End(Not run)
```

---

get_all_differences *Extract All Differences as a Unified Data Frame*

Description

Converts per-variable observation differences into a single long-format data frame suitable for filtering with dplyr, writing to CSV, or programmatic analysis. This is the R equivalent of SAS PROC COMPARE's OUT= dataset with _TYPE_ and _DIF_ variables.

Accepts output from compare_datasets(), cdisc_compare(), or any list containing an observation_comparison element with the standard discrepancies / details / id_details structure.

Usage

```
get_all_differences(comparison_results)
```

Arguments

comparison_results	A dataset_comparison or cdisc_comparison object, or any list with an observation_comparison element.

Value

A data frame with one row per differing cell. Columns:

Variable Character: column name where the difference was found.

Row Integer: row index in df1 (positional matching).

Base The value in df1 (base dataset).

Compare The value in df2 (compare dataset).

Diff Numeric: Base - Compare (NA for character columns).

PctDiff Numeric: absolute percentage difference relative to Base (NA when Base is 0 or column is character).

When key-based matching was used (id_vars), the ID columns are prepended to the left of the data frame.

Returns an empty data frame with the expected columns when no differences exist or observation comparison was skipped.

Examples

```
df1 <- data.frame(id = 1:3, value = c(10, 20, 30), name = c("A", "B", "C"))
df2 <- data.frame(id = 1:3, value = c(10, 25, 30), name = c("A", "B", "D"))
result <- compare_datasets(df1, df2)
diffs <- get_all_differences(result)
head(diffs)
```

`prepare_datasets` *Prepare Datasets for Comparison*

Description

Prepares two datasets for comparison by optionally sorting by specified columns and filtering rows based on a condition.

Usage

```
prepare_datasets(df1, df2, sort_columns = NULL, filter_criteria = NULL)
```

Arguments

`df1`	First dataset to be prepared.
`df2`	Second dataset to be prepared.
`sort_columns`	Columns to sort the datasets by.
`filter_criteria`	Criteria for filtering the datasets.

Value

A list containing two prepared datasets.

Examples

```
df1 <- data.frame(id = c(3, 1, 2), score = c(70, 90, 80))
df2 <- data.frame(id = c(2, 3, 1), score = c(80, 75, 90))
prepare_datasets(df1, df2, sort_columns = "id", filter_criteria = "score > 75")
```

print.cdisc_comparison

Print CDISC Comparison Results

Description

Prints a concise summary of CDISC comparison results. Shows dataset dimensions, domain, number of differences, and a pass/fail verdict based on CDISC validation errors.

Usage

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

Arguments

x	A cdisc_comparison object returned by `cdisc_compare()`.
...	Additional arguments (ignored).

Value

Invisibly returns x.

print.dataset_comparison

Print Dataset Comparison Results

Description

Print Dataset Comparison Results

Usage

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

Arguments

x	A `dataset_comparison` object from `compare_datasets()`.
...	Ignored.

Value

Invisibly returns `x`.

print_cdisc_validation *Print CDISC Validation Results*

Description

Pretty-prints CDISC validation results to the console with a summary and grouped output by category. Displays counts of errors, warnings, and info messages.

Usage

```
print_cdisc_validation(validation_result)
```

Arguments

validation_result	A data frame from validate_cdisc().

Details

Output includes:

- Summary counts of errors, warnings, and info messages
- Issues grouped by category
- Each issue displayed with its variable name and message

Value

Invisibly returns the input (useful for piping).

Examples

```
## Not run:
# Validate a dataset
dm <- data.frame(
  STUDYID = "STUDY001",
  USUBJID = c("SUBJ001", "SUBJ002"),
  DMSEQ = c(1, 1),
  RACE = c("WHITE", "BLACK OR AFRICAN AMERICAN")
)

validation_result <- validate_cdisc(dm, domain = "DM", standard = "SDTM")
print_cdisc_validation(validation_result)

## End(Not run)
```

summary.cdisc_comparison *Summarize CDISC Comparison Results*

Description

Returns a concise one-row data frame summarizing the comparison: domain, standard, row/col counts, number of differences, and CDISC error/warning counts.

Usage

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

Arguments

object	A cdisc_comparison object returned by cdisc_compare().
...	Additional arguments (ignored).

Value

A one-row data frame with summary metrics.

validate_cdisc *Validate CDISC Compliance*

Description

Main validation entry point that checks whether a data frame conforms to CDISC standards. If domain and standard are not provided, they are automatically detected via detect_cdisc_domain(). Dispatches to validate_sdtm() or validate_adam() as appropriate.

Usage

```
validate_cdisc(df, domain = NULL, standard = NULL)
```

Arguments

df	A data frame to validate.
domain	Optional character string specifying the CDISC domain code (e.g., "DM", "AE") or ADaM dataset name (e.g., "ADSL", "ADAE"). If NULL, auto-detected.
standard	Optional character string: "SDTM" or "ADaM". If NULL, auto-detected.

Value

A data frame with columns:

`category`	Character: type of validation issue ("Missing Required Variable", "Missing Expected Variable", "Type Mismatch", "Non-Standard Variable", "Variable Info")
`variable`	Character: variable name
`message`	Character: description of the issue
`severity`	Character: "ERROR", "WARNING", or "INFO"

Examples

```
# Auto-detect domain
dm <- data.frame(
  STUDYID = "STUDY001",
  USUBJID = "SUBJ001",
  DMSEQ = 1,
  RACE = "WHITE",
  stringsAsFactors = FALSE
)
results <- validate_cdisc(dm)
print(results)

# Validate with explicit domain specification
results <- validate_cdisc(dm, domain = "DM", standard = "SDTM")
```

Index

cdisc_compare, *3*, 3
cdisc_compare(), *10*, *13, 14*, *17*, *19*, *21*
clean_dataset, 6
clinCompare *(*clinCompare-package*)*, 2
clinCompare-package, 2
compare_by_group, 7
compare_by_group(), *7*
compare_datasets, *3*, 7
compare_datasets(), *5*, *13*, *17*, *19*
compare_observations, *3*, 9
compare_observations(), *5*, *8*
compare_submission, 10
compare_variables, *3*, 11
compare_variables(), *5*, *7, 8*

detect_cdisc_domain, *3*, 11
detect_cdisc_domain(), *21*

export_report, 12
extract_cdisc_version(), *5*

generate_cdisc_report, 14
generate_detailed_report, 15
generate_detailed_report(), *7*
generate_summary_report, 16
generate_summary_report(), *7*
get_all_differences, 17
get_all_differences(), *13*

prepare_datasets, 18
print(), *7*
print.cdisc_comparison, 19
print.dataset_comparison, 19
print_cdisc_validation, 20

summary.cdisc_comparison, 21

validate_adam(), *21*
validate_cdisc, *3*, 21
validate_cdisc(), *20*
validate_sdtm(), *21*