Package 'dqcheckr'

June 8, 2026

Type Package

Title Automated Data Quality Checks for Recurring Dataset Deliveries

Version 0.2.1

Date 2026-06-03

Description Automates quality verification of recurring external dataset deliveries. For each new file arrival, it runs single-snapshot quality checks, compares the file to the previous delivery, writes a self-contained 'HTML' report, and records summary statistics in a local 'SQLite' database for long-term trend tracking. Supports 'CSV' and fixed-width formats. Custom organisation-specific checks can be supplied as plain R files.

License MIT + file LICENSE

URL https://github.com/mickmioduszewski/dqcheckr

BugReports https://github.com/mickmioduszewski/dqcheckr/issues

Encoding UTF-8

Language en-GB

Depends R (>= 4.2)

Imports readr, DBI, RSQLite, quarto, knitr, kableExtra, ggplot2, gridExtra, dplyr, tidyr, yaml, rlang

Suggests testthat (>= 3.1.0), withr, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Mick Mioduszewski [aut, cre]

Maintainer Mick Mioduszewski <mick@mioduszewski.net>

Repository CRAN

Date/Publication 2026-06-08 05:00:02 UTC

Contents

`check_allowed_values` *QC-09: Check for values outside the allowed set*

Description

For each column that has `allowed_values` configured in `config$column_rules`, returns a `dq_result` flagging any non-empty values not in the allowed list. Returns an empty list when no `allowed_values` rules are configured.

Usage

```
check_allowed_values(df, config)
```

Arguments

df	A data frame with all columns as character vectors (as returned by read_dataset).
config	Named list. Merged configuration as returned by load_config.

Value

A list of dq_result objects, one per configured column. Status is "FAIL" when unexpected values are found; "PASS" otherwise. Returns an empty list if no allowed_values rules are configured.

Examples

```
cfg_dir <- system.file("demonstrations/config", package = "dqcheckr")
cfg  <- load_config("starwars_csv", config_dir = cfg_dir)
path <- system.file("demonstrations/data/starwars.csv", package = "dqcheckr")
df   <- read_dataset(path, cfg)
check_allowed_values(df, cfg)
```

---

check_col_count *QC-05: Report column count*

Description

Returns a single "INFO" dq_result recording the number of columns in the data frame. Never fails or warns.

Usage

```
check_col_count(df, config)
```

Arguments

df	A data frame with all columns as character vectors (as returned by read_dataset).
config	Named list. Merged configuration as returned by load_config. Currently unused; present for API consistency.

Value

A list containing one dq_result with status "INFO".

Examples

```
cfg_dir <- system.file("demonstrations/config", package = "dqcheckr")
cfg  <- load_config("starwars_csv", config_dir = cfg_dir)
path <- system.file("demonstrations/data/starwars.csv", package = "dqcheckr")
df   <- read_dataset(path, cfg)
check_col_count(df, cfg)
```

check_distinct_counts *QC-08: Report distinct value counts for character columns*

Description

For each column whose resolved type is `"character"`, returns one `"INFO"` `dq_result` with the count of distinct non-empty values. Columns inferred as numeric or date are silently skipped.

Usage

```
check_distinct_counts(df, config)
```

Arguments

`df`	A data frame with all columns as character vectors (as returned by `read_dataset`).
`config`	Named list. Merged configuration as returned by `load_config`.

Value

A list of `dq_result` objects (one per character column), all with status `"INFO"`. Returns an empty list if no character columns are found.

Examples

```
cfg_dir <- system.file("demonstrations/config", package = "dqcheckr")
cfg  <- load_config("starwars_csv", config_dir = cfg_dir)
path <- system.file("demonstrations/data/starwars.csv", package = "dqcheckr")
df   <- read_dataset(path, cfg)
check_distinct_counts(df, cfg)
```

check_duplicate_rows *QC-03: Check for fully-duplicate rows*

Description

Returns a single `dq_result` for the whole table. A row is considered a duplicate when every column value is identical to another row.

Usage

```
check_duplicate_rows(df, config)
```

Arguments

df	A data frame with all columns as character vectors (as returned by read_dataset).
config	Named list. Merged configuration as returned by load_config. Currently unused; present for API consistency.

Value

A list containing one dq_result. Status is "WARN" if any duplicate rows exist; "PASS" otherwise.

Examples

```
cfg_dir <- system.file("demonstrations/config", package = "dqcheckr")
cfg  <- load_config("starwars_csv", config_dir = cfg_dir)
path <- system.file("demonstrations/data/starwars.csv", package = "dqcheckr")
df   <- read_dataset(path, cfg)
check_duplicate_rows(df, cfg)
```

---

check_empty_column *QC-02: Check for entirely empty columns*

Description

Returns a dq_result per column. A column is considered empty when every value is NA or the empty string "".

Usage

```
check_empty_column(df, config)
```

Arguments

df	A data frame with all columns as character vectors (as returned by read_dataset).
config	Named list. Merged configuration as returned by load_config.

Value

A list of dq_result objects, one per column. Status is "FAIL" for entirely empty columns; "PASS" otherwise.

Examples

```
cfg_dir <- system.file("demonstrations/config", package = "dqcheckr")
cfg  <- load_config("starwars_csv", config_dir = cfg_dir)
path <- system.file("demonstrations/data/starwars.csv", package = "dqcheckr")
df   <- read_dataset(path, cfg)
check_empty_column(df, cfg)
```

check_inferred_types *QC-06: Report inferred column types*

Description

Returns one "INFO" dq_result per column recording the type resolved by resolve_col_type ("date", "numeric", "character", or "unknown"). Per-column overrides from config$column_types are respected.

Usage

```
check_inferred_types(df, config)
```

Arguments

df	A data frame with all columns as character vectors (as returned by read_dataset).
config	Named list. Merged configuration as returned by load_config.

Value

A list of dq_result objects, one per column, all with status "INFO".

Examples

```
cfg_dir <- system.file("demonstrations/config", package = "dqcheckr")
cfg  <- load_config("starwars_csv", config_dir = cfg_dir)
path <- system.file("demonstrations/data/starwars.csv", package = "dqcheckr")
df   <- read_dataset(path, cfg)
check_inferred_types(df, cfg)
```

check_key_uniqueness *QC-12: Check uniqueness of key column(s)*

Description

Checks that the column(s) listed in config$key_columns have no duplicate values. When key_columns is a single string, one result is returned for that column. When it is a character vector of length > 1, a single result covering the composite key is returned. Returns an empty list if key_columns is not configured.

Usage

```
check_key_uniqueness(df, config)
```

Arguments

df	A data frame with all columns as character vectors (as returned by read_dataset).
config	Named list. Merged configuration as returned by load_config.

Value

A list of dq_result objects. Status is "FAIL" when duplicates or missing key columns are detected; "PASS" otherwise. Returns an empty list if key_columns is not configured.

Examples

```
cfg_dir <- system.file("demonstrations/config", package = "dqcheckr")
cfg  <- load_config("starwars_csv", config_dir = cfg_dir)
path <- system.file("demonstrations/data/starwars.csv", package = "dqcheckr")
df   <- read_dataset(path, cfg)
check_key_uniqueness(df, cfg)
```

---

check_min_row_count *QC-14: Check row count bounds and optional file size*

Description

Runs up to three sub-checks, each returning a separate dq_result:

1. **File size** – only when file_path is supplied and max_file_size_mb is configured in rules: FAIL if the file exceeds the size limit.
2. **Minimum row count** – FAIL if row_count < min_row_count. Skipped (PASS with a note) when min_row_count is 0.
3. **Maximum row count** – only when max_row_count is configured in rules: FAIL if row_count > max_row_count.

Usage

```
check_min_row_count(df, config, file_path = NULL)
```

Arguments

df	A data frame with all columns as character vectors (as returned by read_dataset).
config	Named list. Merged configuration as returned by load_config.
file_path	Character or NULL. Absolute path to the file on disk, required for the optional file-size sub-check.

Value

A list of dq_result objects (one to three entries depending on which sub-checks are active).

Examples

```
cfg_dir <- system.file("demonstrations/config", package = "dqcheckr")
cfg  <- load_config("starwars_csv", config_dir = cfg_dir)
path <- system.file("demonstrations/data/starwars.csv", package = "dqcheckr")
df   <- read_dataset(path, cfg)
check_min_row_count(df, cfg, file_path = path)
```

`check_missing_rate` *QC-01: Check missing rate per column*

Description

Returns a `dq_result` per column flagging columns whose proportion of missing or empty values exceeds `max_missing_rate`.

Usage

```
check_missing_rate(df, config)
```

Arguments

`df`	A data frame with all columns as character vectors.
`config`	Named list as returned by `load_config`.

Value

A list of `dq_result` objects, one per column.

Examples

```
cfg_dir <- system.file("demonstrations/config", package = "dqcheckr")
cfg  <- load_config("starwars_csv", config_dir = cfg_dir)
path <- system.file("demonstrations/data/starwars.csv", package = "dqcheckr")
df   <- read_dataset(path, cfg)
check_missing_rate(df, cfg)
```

check_non_numeric *QC-11: Check non-numeric rate in numeric columns*

Description

For each column whose resolved type is "numeric", computes the proportion of non-empty values that cannot be coerced to numeric. Returns "FAIL" when the rate exceeds max_non_numeric_rate (default 0.01), "WARN" when it exceeds warn_non_numeric_rate (default 0), and "PASS" otherwise. Both thresholds support per-column overrides via config$column_rules.

Usage

```
check_non_numeric(df, config)
```

Arguments

df	A data frame with all columns as character vectors (as returned by read_dataset).
config	Named list. Merged configuration as returned by load_config.

Value

A list of dq_result objects, one per numeric column. Returns an empty list if no numeric columns are found.

Examples

```
cfg_dir <- system.file("demonstrations/config", package = "dqcheckr")
cfg  <- load_config("starwars_csv", config_dir = cfg_dir)
path <- system.file("demonstrations/data/starwars.csv", package = "dqcheckr")
df   <- read_dataset(path, cfg)
check_non_numeric(df, cfg)
```

check_numeric_bounds *QC-10: Check for out-of-range numeric values*

Description

For each column that has min_value or max_value configured in config$column_rules, returns a dq_result flagging any values that fall outside the specified range. Returns an empty list when no bound rules are configured.

Usage

```
check_numeric_bounds(df, config)
```

Arguments

df	A data frame with all columns as character vectors (as returned by read_dataset).
config	Named list. Merged configuration as returned by load_config.

Value

A list of dq_result objects, one per configured column. Status is "FAIL" when out-of-range values are found; "PASS" otherwise. Returns an empty list if no bound rules are configured.

Examples

```
cfg_dir <- system.file("demonstrations/config", package = "dqcheckr")
cfg  <- load_config("starwars_csv", config_dir = cfg_dir)
path <- system.file("demonstrations/data/starwars.csv", package = "dqcheckr")
df   <- read_dataset(path, cfg)
check_numeric_bounds(df, cfg)
```

---

check_numeric_stats *QC-07: Report numeric summary statistics*

Description

For each column whose resolved type is "numeric", returns one "INFO" dq_result containing min, max, mean, and standard deviation of the parseable values. Columns inferred as non-numeric are silently skipped.

Usage

```
check_numeric_stats(df, config)
```

Arguments

df	A data frame with all columns as character vectors (as returned by read_dataset).
config	Named list. Merged configuration as returned by load_config.

Value

A list of dq_result objects (one per numeric column), all with status "INFO". Returns an empty list if no numeric columns are found.

Examples

```
cfg_dir <- system.file("demonstrations/config", package = "dqcheckr")
cfg  <- load_config("starwars_csv", config_dir = cfg_dir)
path <- system.file("demonstrations/data/starwars.csv", package = "dqcheckr")
df   <- read_dataset(path, cfg)
check_numeric_stats(df, cfg)
```

`check_outliers` *QC-15: Detect statistical outliers in numeric columns*

Description

For each column whose resolved type is `"numeric"`, applies up to two outlier detection methods (combined with logical OR):

1. **Z-score**: values whose absolute Z-score exceeds `max_z_score` are flagged.
2. **IQR fence**: values below `Q1 - k * IQR` or above `Q3 + k * IQR` (where `k = iqr_fence_multiplier`) are flagged.

Both thresholds support per-column overrides via `config$column_rules`. A column is skipped (PASS with a note) when neither threshold is configured or when it has fewer than four parseable values.

Usage

```
check_outliers(df, config)
```

Arguments

`df`	A data frame with all columns as character vectors (as returned by `read_dataset`).
`config`	Named list. Merged configuration as returned by `load_config`.

Value

A list of `dq_result` objects, one per numeric column. Status is `"FAIL"` when outliers are detected; `"PASS"` otherwise. Returns an empty list if no numeric columns are found.

Examples

```
cfg_dir <- system.file("demonstrations/config", package = "dqcheckr")
cfg  <- load_config("starwars_csv", config_dir = cfg_dir)
path <- system.file("demonstrations/data/starwars.csv", package = "dqcheckr")
df   <- read_dataset(path, cfg)
check_outliers(df, cfg)
```

check_pattern *QC-13: Check values against a regex pattern*

Description

For each column that has a `pattern` configured in `config$column_rules`, returns a `dq_result` reporting how many non-empty values do not match the Perl-compatible regular expression. Returns an empty list when no pattern rules are configured.

Usage

```
check_pattern(df, config)
```

Arguments

`df`	A data frame with all columns as character vectors (as returned by `read_dataset`).
`config`	Named list. Merged configuration as returned by `load_config`.

Value

A list of `dq_result` objects, one per configured column. Status is `"FAIL"` when any values violate the pattern; `"PASS"` otherwise. Returns an empty list if no pattern rules are configured.

Examples

```
cfg_dir <- system.file("demonstrations/config", package = "dqcheckr")
cfg  <- load_config("starwars_csv", config_dir = cfg_dir)
path <- system.file("demonstrations/data/starwars.csv", package = "dqcheckr")
df   <- read_dataset(path, cfg)
check_pattern(df, cfg)
```

check_row_count *QC-04: Report row count*

Description

Returns a single `"INFO"` `dq_result` recording the number of rows in the data frame. Never fails or warns; use `check_min_row_count` for threshold-based row count checks.

Usage

```
check_row_count(df, config)
```

Arguments

df	A data frame with all columns as character vectors (as returned by read_dataset).
config	Named list. Merged configuration as returned by load_config. Currently unused; present for API consistency.

Value

A list containing one dq_result with status "INFO".

Examples

```
cfg_dir <- system.file("demonstrations/config", package = "dqcheckr")
cfg  <- load_config("starwars_csv", config_dir = cfg_dir)
path <- system.file("demonstrations/data/starwars.csv", package = "dqcheckr")
df   <- read_dataset(path, cfg)
check_row_count(df, cfg)
```

---

check_schema_contract *SC-01 / SC-02: Check columns against the expected schema contract*

Description

Compares the columns present in df against config$expected_columns:

- **SC-01**: one "FAIL" result per column present in the file but not listed in expected_columns.
- **SC-02**: one "FAIL" result per column listed in expected_columns but absent from the file.

Returns an empty list if expected_columns is not configured.

Usage

```
check_schema_contract(df, config)
```

Arguments

df	A data frame with all columns as character vectors (as returned by read_dataset).
config	Named list. Merged configuration as returned by load_config.

Value

A list of dq_result objects. Each schema violation produces one "FAIL" result; a "PASS" result is emitted for each sub-check when no violations are found. Returns an empty list if expected_columns is not configured.

Examples

```
cfg_dir <- system.file("demonstrations/config", package = "dqcheckr")
cfg  <- load_config("starwars_csv", config_dir = cfg_dir)
path <- system.file("demonstrations/data/starwars.csv", package = "dqcheckr")
df   <- read_dataset(path, cfg)
check_schema_contract(df, cfg)
```

---

compare_snapshots *Compare two snapshots from the SQLite database*

Description

Reads two historical snapshot records (by ID) from the SQLite database and computes table-level, schema, and per-column statistical drift. Optionally renders an HTML drift report.

Usage

```
compare_snapshots(
  dataset_name,
  snapshot_id_prev = NULL,
  snapshot_id_curr = NULL,
  db_path = NULL,
  config_dir = ".",
  report = TRUE,
  open_report = interactive()
)
```

Arguments

dataset_name	Character. Dataset name to compare.
snapshot_id_prev	Integer or NULL. ID of the earlier snapshot. If NULL, defaults to the second-most-recent snapshot by ID.
snapshot_id_curr	Integer or NULL. ID of the later snapshot. If NULL, defaults to the most-recent snapshot by ID.
db_path	Character or NULL. Path to the SQLite snapshot database. If NULL (the default), the path is read from snapshot_db in dqcheckr.yml.
config_dir	Character. Path to the directory containing dqcheckr.yml. Used to read thresholds, report_output_dir, and (when db_path is NULL) snapshot_db.
report	Logical. Whether to render an HTML drift report.
open_report	Logical. Whether to open the HTML report in the browser after rendering (only takes effect in interactive sessions).

Value

Invisibly, a named list with elements `dataset_name`, `snap_prev`, `snap_curr`, `table_drift`, `schema_changes`, `missing_rate_changes`, `non_numeric_changes`, `mean_shifts`, `distinct_changes`.

Examples

```
tmp     <- tempdir()
db_path <- file.path(tmp, "snap.sqlite")
cfg_yml <- file.path(tmp, "dqcheckr.yml")
ds_yml  <- file.path(tmp, "starwars_csv.yml")
dat     <- system.file("demonstrations/data/starwars.csv", package = "dqcheckr")
writeLines(c(
  paste0('snapshot_db: "', db_path, '"'),
  paste0('report_output_dir: "', tmp, '"'),
  'default_rules:',
  '  max_missing_rate: 0.60',
  '  min_row_count: 80'
), cfg_yml)
writeLines(c(
  'dataset_name: "starwars_csv"',
  paste0('current_file: "', dat, '"'),
  'format: csv',
  'encoding: "UTF-8"',
  'delimiter: ","'
), ds_yml)
run_dq_check("starwars_csv", config_dir = tmp, open_report = FALSE)
run_dq_check("starwars_csv", config_dir = tmp, open_report = FALSE)
drift <- compare_snapshots("starwars_csv", config_dir = tmp, report = FALSE)
names(drift)
```

---

detect_files *Detect current and previous dataset files*

Description

Resolves the current and previous file paths from the configuration. If `current_file` is set explicitly, it is used directly. Otherwise the two most recently modified files in `folder` are used.

Usage

```
detect_files(config)
```

Arguments

`config`	Named list. Merged configuration as returned by `load_config`.

Value

A named list with elements `current` (character path) and `previous` (character path or `NULL`).

Examples

```
cfg_dir <- system.file("demonstrations/config", package = "dqcheckr")
cfg <- load_config("starwars_csv", config_dir = cfg_dir)
cfg$current_file <- system.file("demonstrations/data/starwars.csv",
                                package = "dqcheckr")
files <- detect_files(cfg)
files$current
```

`dq_result` *Construct a data quality result object*

Description

Creates the atomic result unit returned by every check function.

Usage

```
dq_result(
  check_id,
  check_name,
  column = NA_character_,
  status,
  observed,
  threshold = NA_character_,
  message
)
```

Arguments

`check_id`	Character. Short identifier for the check (e.g. `"QC-01"`).
`check_name`	Character. Human-readable name of the check.
`column`	Character. Column the check applies to, or `NA_character_` for row-level or file-level checks.
`status`	Character. One of `"PASS"`, `"WARN"`, `"FAIL"`, or `"INFO"`.
`observed`	Character. What was observed (e.g. `"5.2% missing"`).
`threshold`	Character. The configured threshold, or `NA_character_` if not applicable.
`message`	Character. Human-readable description of the result.

Value

A named list with seven elements: `check_id`, `check_name`, `column`, `status`, `observed`, `threshold`, `message`.

Examples

```
dq_result("QC-01", "Missing rate", column = "age",
          status = "PASS", observed = "0% missing",
          message = "No missing values.")
```

`infer_col_type` *Infer the logical type of a character column*

Description

Classifies a character vector as `"date"`, `"numeric"`, `"character"`, or `"unknown"` by applying rules in priority order.

Usage

```
infer_col_type(x, threshold = 0.9)
```

Arguments

`x`	Character vector to classify (as read from a CSV or FWF file).
`threshold`	Numeric. Minimum proportion of non-empty values that must parse as numeric for the column to be classified as `"numeric"`. Defaults to `0.90`. Configurable via `type_inference_threshold` in `rule_overrides`.

Value

A single character string: `"date"`, `"numeric"`, `"character"`, or `"unknown"`.

Examples

```
infer_col_type(c("2024-01-01", "2024-06-15"))   # "date"
infer_col_type(c("1.5", "2.0", "3.1"))          # "numeric"
infer_col_type(c("high", "low", "medium"))       # "character"
infer_col_type(c(NA, "", NA))                    # "unknown"
infer_col_type(c(rep("1", 17), "a", "b", "c"), threshold = 0.80)  # "numeric"
```

list_snapshots *List snapshots available in the database*

Description

Returns a data frame of snapshot records for the given dataset (or all datasets if `dataset_name` is `NULL`), ordered by dataset name and snapshot ID.

Usage

```
list_snapshots(dataset_name = NULL, db_path = NULL)
```

Arguments

`dataset_name`	Character or `NULL`. If supplied, only snapshots for that dataset are returned. If `NULL`, all datasets are returned.
`db_path`	Character. Path to the SQLite snapshot database. Required; there is no default (a relative default would be path-sensitive).

Value

A data frame with columns `id`, `dataset_name`, `file_name`, `run_timestamp`, `row_count`, `overall_status`. Returns an empty data frame if the database does not exist or contains no matching records.

Examples

```
list_snapshots(db_path = tempfile(fileext = ".sqlite"))
```

load_config *Load and merge dataset configuration*

Description

Reads the global `dqcheckr.yml` and the dataset-specific YAML, merging `rule_overrides` from the dataset config on top of `default_rules` from the global config. Top-level keys `snapshot_db` and `report_output_dir` are inherited from the global config when absent from the dataset config.

Usage

```
load_config(dataset_name, config_dir)
```

Arguments

`dataset_name`	Character. Dataset name; must match `<dataset_name>.yml` in `config_dir`.
`config_dir`	Character. Path to the directory containing both YAML files.

Value

A named list representing the merged configuration.

Examples

```
cfg_dir <- system.file("demonstrations/config", package = "dqcheckr")
cfg <- load_config("starwars_csv", config_dir = cfg_dir)
cfg$format
```

`overall_status` *Compute the worst status across a list of dq_result objects*

Description

Returns the single worst status in precedence order: `"FAIL"` > `"WARN"` > `"PASS"` > `"INFO"`.

Usage

```
overall_status(results)
```

Arguments

`results`	A list of `dq_result` objects.

Value

A single character string: `"FAIL"`, `"WARN"`, `"PASS"`, or `"INFO"`.

Examples

```
r1 <- dq_result("QC-01", "test", status = "PASS", observed = "ok", message = "ok")
r2 <- dq_result("QC-02", "test", status = "WARN", observed = "ok", message = "ok")
overall_status(list(r1, r2))  # "WARN"
```

read_dataset *Read a dataset file into a data frame*

Description

Reads a CSV or fixed-width file, coercing all columns to character and trimming whitespace. Encoding and delimiter are taken from `config`.

Usage

```
read_dataset(path, config)
```

Arguments

`path`	Character. Path to the file to read.
`config`	Named list. Merged configuration as returned by `load_config`. Must include `format` (`"csv"` or `"fwf"`). For FWF files, `fwf_widths` is required and `fwf_col_names` and `fwf_skip` are optional.

Value

A data frame with all columns as character vectors.

Examples

```
cfg_dir <- system.file("demonstrations/config", package = "dqcheckr")
cfg  <- load_config("starwars_csv", config_dir = cfg_dir)
path <- system.file("demonstrations/data/starwars.csv", package = "dqcheckr")
df   <- read_dataset(path, cfg)
```

read_recent_snapshots *Read recent snapshot history from the SQLite database*

Description

Retrieves the `n` most recent run records for a given dataset from the snapshot database, ordered newest-first.

Usage

```
read_recent_snapshots(db_path, dataset_name, n = 10)
```

Arguments

db_path	Character. Path to the SQLite database file.
dataset_name	Character. Dataset name to filter on.
n	Integer. Maximum number of records to return. Defaults to 10.

Value

A data frame with one row per run and columns including id, dataset_name, run_timestamp, file_name, row_count, col_count, overall_status, check_pass_count, check_warn_count, check_fail_count, check_info_count, new_cols_vs_previous, missing_cols_vs_previous, new_cols_vs_schema, missing_cols_vs_schema, comparison_mode, render_status, and type_changed_cols_vs_pre
Returns an empty data frame if the database does not exist or contains no records for the dataset.

Examples

```
history <- read_recent_snapshots(tempfile(fileext = ".sqlite"), "starwars_csv")
```

---

resolve_col_type *Resolve the effective type of a column, respecting config overrides*

Description

Returns the type for col from the column_types map in config if one is set, otherwise falls back to infer_col_type. Use this in custom check scripts instead of calling infer_col_type() directly so that type overrides are respected.

Usage

```
resolve_col_type(col, x, config)
```

Arguments

col	Character. Column name.
x	Character vector. The column's values (as read from the file).
config	Named list. Merged configuration as returned by load_config.

Value

A single character string: "date", "numeric", "character", or "unknown".

Examples

```
cfg_dir <- system.file("demonstrations/config", package = "dqcheckr")
cfg <- load_config("starwars_csv", config_dir = cfg_dir)
resolve_col_type("name", c("Luke", "Leia", "Han"), cfg)   # "character"
```

run_comparison_checks *Run all version comparison checks between two dataset snapshots*

Description

Runs CP-01 to CP-08 comparing a current delivery against the previous one.

Usage

```
run_comparison_checks(df_current, df_previous, config)
```

Arguments

df_current	A data frame. The current delivery.
df_previous	A data frame. The previous delivery.
config	Named list. Merged configuration as returned by load_config.

Value

A list of dq_result objects. The list carries attributes new_cols, dropped_cols, and type_changed_cols (character vectors) for use by the snapshot writer.

Examples

```
cfg_dir   <- system.file("demonstrations/config", package = "dqcheckr")
cfg       <- load_config("starwars_csv", config_dir = cfg_dir)
curr_path <- system.file("demonstrations/data2/starwars_v2.csv", package = "dqcheckr")
prev_path <- system.file("demonstrations/data2/starwars_v1.csv", package = "dqcheckr")
curr      <- read_dataset(curr_path, cfg)
prev      <- read_dataset(prev_path, cfg)
results   <- run_comparison_checks(curr, prev, cfg)
```

run_custom_checks *Run organisation-specific custom checks*

Description

Sources the R file specified by config$custom_checks_file, which must define a function custom_checks(df) returning a list of dq_result objects. Returns an empty list if custom_checks_file is not set in the config.

Usage

```
run_custom_checks(df, config)
```

Arguments

df	A data frame. The current delivery.
config	Named list. Merged configuration as returned by load_config.

Details

The file is sourced into an isolated environment whose parent is baseenv(), so only base R functions are available by default. dq_result is explicitly injected and can be called without qualification. All other dqcheckr exports (e.g. resolve_col_type, infer_col_type) must be qualified: dqcheckr::resolve_col_type(). Any error – missing file, undefined function, or runtime failure – stops the run with a clear message.

Value

A list of dq_result objects (may be empty).

Examples

```
cfg_dir <- system.file("demonstrations/config", package = "dqcheckr")
cfg     <- load_config("starwars_csv", config_dir = cfg_dir)
path    <- system.file("demonstrations/data/starwars.csv", package = "dqcheckr")
df      <- read_dataset(path, cfg)
results <- run_custom_checks(df, cfg)
```

---

run_dq_check *Run a full data quality check pipeline*

Description

Orchestrates the complete dqcheckr pipeline: loads configuration, detects files, runs QC and comparison checks, writes a snapshot to SQLite, and renders an HTML report.

Usage

```
run_dq_check(dataset_name, config_dir = ".", open_report = TRUE)
```

Arguments

dataset_name	Character. Name of the dataset; must match a YAML config file <dataset_name>.yml in config_dir.
config_dir	Character. Path to the directory containing dqcheckr.yml and the dataset YAML file. Defaults to ".".
open_report	Logical. Whether to open the HTML report in the browser after rendering (only takes effect in interactive sessions).

Value

Invisibly, a named list with:

status Overall status string: `"PASS"`, `"WARN"`, `"FAIL"`, or `"INFO"`.

report_path Absolute path to the rendered HTML report, or `NULL` if rendering was skipped.

snapshot_id Integer row ID of the snapshot written to SQLite, or `NULL` if the write failed.

Examples

```
tmp <- gsub("\\\\", "/", tempdir())
dat <- system.file("demonstrations/data/starwars.csv", package = "dqcheckr")
writeLines(c(
  paste0('snapshot_db: "',       tmp, '/snap.sqlite"'),
  paste0('report_output_dir: "', tmp, '"'),
  'default_rules:',
  '  max_missing_rate: 0.60',
  '  min_row_count: 80'
), file.path(tmp, "dqcheckr.yml"))
writeLines(c(
  'dataset_name: "starwars_csv"',
  paste0('current_file: "', dat, '"'),
  'format: csv',
  'encoding: "UTF-8"',
  'delimiter: ","'
), file.path(tmp, "starwars_csv.yml"))
result <- run_dq_check("starwars_csv", config_dir = tmp, open_report = FALSE)
result$status
```

`run_qc_checks` *Run all generic quality checks on a dataset*

Description

Runs the full QC check suite (QC-01 to QC-15, SC-01, SC-02) against a single data frame snapshot.

Usage

```
run_qc_checks(df, config, file_path = NULL)
```

Arguments

`df`	A data frame with all columns as character vectors (as returned by `read_dataset`).
`config`	Named list. Merged configuration as returned by `load_config`.
`file_path`	Character or `NULL`. Absolute path to the file, used for the optional `max_file_size_mb` check in QC-14.

Value

A list of dq_result objects.

Examples

```
cfg_dir <- system.file("demonstrations/config", package = "dqcheckr")
cfg     <- load_config("starwars_csv", config_dir = cfg_dir)
path    <- system.file("demonstrations/data/starwars.csv", package = "dqcheckr")
df      <- read_dataset(path, cfg)
results <- run_qc_checks(df, cfg)
```

Index