Package 'datagrade'

November 28, 2026

Type Package

Title Data Quality Assessment on the ISO/IEC 25012 Model

Version 0.3.0

Description A single-call data quality assessment scoring a data set on the five inherent characteristics of the ISO/IEC 25012 data quality model (accuracy, completeness, consistency, credibility and currentness) through the fifteen quality properties beneath them, ISO/IEC (2008) <https://www.iso.org/standard/35736.html>. Each property is measured as a ratio of counted data items, the form ISO/IEC 25024 measurement functions take, ISO/IEC (2015) <https://www.iso.org/standard/35749.html>. Six properties are computable from a table alone; the rest compare the data against expectations declared through a specification object, and are reported as not applicable rather than approximated when none is supplied. Also reports missing-data patterns, duplicates, redundant columns, outliers, multicollinearity and distributional deviations, compares two deliveries of the same data set measure by measure, and returns a structured object with plots rather than console text alone. The statistics implement the methodology of the author's master's dissertation, Demirhan (2024) ``Assessing Data Quality Management Issues in Open-Sourced Data Sets with Statistical Methodology'' <https://halukdemirhan.com/assets/msc-thesis-haluk-demirhan-public.pdf>.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0)

Imports cli, ggplot2 (>= 3.4.0), stats, tools, utils

Suggests knitr, randomForest, rmarkdown, readxl, testthat (>= 3.0.0)

URL https://github.com/ohalukdemirhan/datagrade

BugReports https://github.com/ohalukdemirhan/datagrade/issues

Config/testthat/edition 3

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Omer Haluk Demirhan [aut, cre]

Maintainer Omer Haluk Demirhan <hlk01dn@gmail.com>

Repository CRAN

Date/Publication 2026-11-28 00:30:02 UTC

Contents

analyze_correlation . . . 2
analyze_distribution . . . 3
analyze_missing_data_patterns . . . 4
as.data.frame.dg_report . . . 4
calculate_consistency_score . . . 5
check_unique_identifiers . . . 5
clean_text_columns . . . 6
detect_data_types . . . 6
detect_duplicates . . . 7
detect_outliers . . . 8
dg_assess . . . 8
dg_compare . . . 11
dg_plots . . . 12
dg_save_plots . . . 13
dg_spec . . . 14
dg_vif . . . 16
flag_anomalies_in_relationships . . . 17
flag_redundant_columns . . . 18
impute_missing_values . . . 18
print.dg_comparison . . . 19
print.dg_report . . . 20
summary.dg_report . . . 20
theme_dg . . . 21
validate_dates . . . 21

Index **22**

`analyze_correlation` *Correlation matrix, optionally per category*

Description

Correlation matrix, optionally per category

Usage

```
analyze_correlation(data, category_column = NULL)
```

Arguments

`data`	A data frame.
`category_column`	Optional grouping column.

Value

A list with the overall correlation matrix and, if requested, one matrix per group.

Examples

```
analyze_correlation(iris, "Species")$overall
```

`analyze_distribution` *Assess the distribution of every numeric column*

Description

Reports skewness, excess kurtosis and a Shapiro-Wilk normality test. The test is limited to 5,000 observations by its own definition, so larger columns are tested on a deterministic subsample and the sample size used is reported.

Usage

```
analyze_distribution(data, sample_size = 5000L, seed = 1L)
```

Arguments

`data`	A data frame.
`sample_size`	Observations used for the normality test.
`seed`	Seed for the subsample, so repeated runs agree.

Value

A data frame with one row per numeric column.

Examples

```
analyze_distribution(iris)
```

analyze_missing_data_patterns

Summarise missing-data patterns

Description

Summarise missing-data patterns

Usage

```
analyze_missing_data_patterns(data, high_threshold = 0.3)
```

Arguments

data	A data frame.
high_threshold	Share of missing values above which a column or row is flagged as heavily incomplete.

Value

A list with a per-column summary data frame, the names of heavily incomplete columns, the count of heavily incomplete rows, and the number of rows that are complete in every column.

Examples

```
analyze_missing_data_patterns(airquality)$columns
```

as.data.frame.dg_report

Coerce a quality report to a one-row data frame

Description

Convenient for assessing many files and stacking the results.

Usage

```
## S3 method for class 'dg_report'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	A ‘dg_report‘.
row.names	Passed to [base::as.data.frame()].
optional	Passed to [base::as.data.frame()].
...	Unused.

Value

A one-row data frame.

`calculate_consistency_score` *Consistency score from multicollinearity*

Description

Consistency score from multicollinearity

Usage

```
calculate_consistency_score(data, vif_threshold = 5)
```

Arguments

`data`	A data frame.
`vif_threshold`	VIF above which a column counts as inconsistent.

Value

A list with the score as a ratio in ‘[0, 1]‘, the VIF vector, and the names of the offending columns.

Examples

```
calculate_consistency_score(iris)$score
```

`check_unique_identifiers` *Identify columns that could serve as a unique key*

Description

Identify columns that could serve as a unique key

Usage

```
check_unique_identifiers(data)
```

Arguments

`data`	A data frame.

Value

A character vector of candidate key columns, possibly empty.

clean_text_columns *Normalise text columns without destroying information*

Description

Trims whitespace and collapses internal runs of spaces. Case folding and punctuation stripping are opt-in and, when enabled, are Unicode-aware.

Usage

```
clean_text_columns(data, lowercase = FALSE, strip_punctuation = FALSE)
```

Arguments

data	A data frame.
lowercase	Fold to lower case.
strip_punctuation	Remove characters that are neither alphanumeric nor whitespace. Unicode letters and digits are preserved.

Details

The dissertation's implementation applied ‘gsub("[^a-zA-Z0-9]", "")‘ unconditionally, which deleted every accented and non-Latin character, every date separator and every ‘@‘ in an e-mail address. Assessment is a diagnosis, not a treatment, so nothing here is destructive by default.

Value

The data frame with its character columns normalised.

Examples

```
clean_text_columns(data.frame(city = c("  Agri ", "Corum")))
```

detect_data_types *Detect the semantic type of every column*

Description

Unlike a plain ‘class()‘ lookup this distinguishes an identifier column from an ordinary numeric or text column, and a low-cardinality text column from free text.

Usage

```
detect_data_types(data, categorical_max = 20L)
```

Arguments

`data`	A data frame.
`categorical_max`	Columns with at most this many distinct values are reported as ‘"Categorical"‘ rather than ‘"Text"‘.

Details

The uniqueness test is evaluated **before** the storage-type tests. In the dissertation's pseudocode (Appendix A.3) it came last, which made the ‘"ID"‘ branch unreachable: a character key matched ‘is.character()‘ and returned ‘"Text"‘ first.

Value

A named character vector, one entry per column, with values ‘"ID"‘, ‘"Numeric"‘, ‘"Text"‘, ‘"Date"‘, ‘"Categorical"‘ or ‘"Logical"‘.

Examples

```
detect_data_types(data.frame(id = c("a", "b"), n = 1:2))
```

`detect_duplicates` *Count duplicate rows*

Description

Returns a count and a small sample rather than the full set of duplicated rows. On a data set with millions of near-identical rows, materialising every duplicate can be larger than the input itself.

Usage

```
detect_duplicates(data, columns = NULL, max_examples = 5L)
```

Arguments

`data`	A data frame.
`columns`	Optional subset of columns defining a duplicate.
`max_examples`	Number of example rows to retain.

Value

A list with ‘n‘ (count), ‘rate‘ (share of rows) and ‘examples‘.

detect_outliers *Detect outliers in numeric columns*

Description

Detect outliers in numeric columns

Usage

```
detect_outliers(
  data,
  numeric_columns = NULL,
  method = c("z_score", "iqr"),
  category_column = NULL,
  threshold = 3
)
```

Arguments

data	A data frame.
numeric_columns	Columns to test. Defaults to every numeric column.
method	‘"z_score"‘ (default) or ‘"iqr"‘.
category_column	Optional grouping column; outliers are then counted within each group.
threshold	Z-score cut-off. Ignored by the IQR method, which uses the conventional 1.5 x IQR fence.

Value

A list with the total count, the share of tested values that were flagged, and a per-column breakdown.

Examples

```
detect_outliers(iris)$total
```

dg_assess *Assess the quality of a data set*

Description

Runs every check in the package and returns a ‘dg_report‘ object scoring the data on the five inherent characteristics of the ISO/IEC 25012 data quality model — accuracy, completeness, consistency, credibility and currentness — through the fifteen quality properties beneath them. Each property is measured as a ratio ‘A / B‘ over counted data items, the form ISO/IEC 25024 measurement functions take.

Usage

```
dg_assess(
  data = NULL,
  path = NULL,
  spec = NULL,
  outlier_method = c("z_score", "iqr"),
  imputation_method = c("mean", "median", "none"),
  correlation_threshold = 0.5,
  vif_threshold = 5,
  category_column = NULL,
  weights = c(accuracy = 1, completeness = 1, consistency = 1, credibility = 1,
    currentness = 1),
  clean_text = TRUE,
  strip_punctuation = FALSE,
  parse_dates = TRUE,
  check_duplicates = TRUE,
  anomaly_detection = FALSE,
  sample_size = 1e+05,
  seed = 1L,
  keep_data = FALSE,
  verbose = TRUE
)

data_quality_assessment(...)
```

Arguments

data	A data frame. Either ‘data‘ or ‘path‘ must be supplied.
path	Path to a ‘.csv‘, ‘.xls‘ or ‘.xlsx‘ file.
spec	A [dg_spec()] declaring what the data is supposed to look like. Optional; without it the nine specification-dependent measures report as not applicable.
outlier_method	‘"z_score"‘ (default) or ‘"iqr"‘.
imputation_method	‘"mean"‘ (default), ‘"median"‘ or ‘"none"‘.
correlation_threshold	Absolute correlation above which a column pair is reported as redundant.
vif_threshold	Variance inflation factor above which a column counts against the consistency score.
category_column	Optional grouping column for outlier and correlation analysis.
weights	Named weights for the five characteristics in the overall score. Characteristics that do not apply to the data are dropped and the remaining weights renormalised. ‘timeliness‘ is accepted as an alias for ‘currentness‘.
clean_text	Normalise whitespace in character columns.
strip_punctuation	Also remove punctuation. Unicode-aware; off by default because it is destructive.

parse_dates	Convert character columns that parse cleanly as dates. Required for the timeliness dimension to be measurable from a CSV.
check_duplicates	Whether to run the duplicate scan, the only check whose cost is superlinear in row count.
anomaly_detection	Fit a random forest to flag anomalous rows.
sample_size	Row budget for the checks that subsample.
seed	Seed used by every subsampling step.
keep_data	Retain the imputed data frame on the returned object.
verbose	Print progress to the console.
...	Passed to [dg_assess()].

Details

Six of the fifteen properties are computable from a table alone. The other nine have a denominator that counts *the items for which an expectation is declared*, which no data set contains; supply a [dg_spec()] to make them measurable. Without one they are reported as not applicable and dropped from the aggregate, never scored as zero.

A sixth, non-ISO score is reported alongside them. ‘plausibility‘ is the outlier-based measure this package used before it followed the standard: it compares values against an interval *derived* from the data rather than one declared in a specification. That makes it useful and cheap on data with no documentation, but it is not accuracy in the ISO sense — an outlier need not be wrong, and a wrong value can sit exactly on the mean — so it is reported separately and excluded from the overall score.

The function never modifies the caller’s data. Text normalisation and imputation are applied to an internal copy used for the statistics, and the imputed frame is returned only when ‘keep_data = TRUE‘.

‘data_quality_assessment()‘ is the name used in the dissertation and is kept as an alias.

Value

An object of class ‘dg_report‘. See [print.dg_report()], [summary.dg_report()] and [dg_plots()].

Examples

```
report <- dg_assess(iris, verbose = FALSE)
report$scores
report$measures[, c("code", "value", "a", "b")]

# Declaring an expectation makes the accuracy measures computable.
spec <- dg_spec(ranges = list(Sepal.Length = c(4, 8)),
                domains = list(Species = levels(iris$Species)))
dg_assess(iris, spec = spec, verbose = FALSE)$scores
```

dg_compare *Compare two deliveries of the same data set*

Description

Assesses two versions of a data set — last month's extract and this month's, the file before and after a pipeline change — and reports what changed, property by property, on the ISO/IEC 25012 model.

Usage

```
dg_compare(old, new, spec = NULL, tolerance = 0.01, ...)

## S3 method for class 'dg_comparison'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

old, new	Two data frames, or two 'dg_report' objects from [dg_assess()]. 'old' is the reference.
spec	A [dg_spec()] applied to both versions. Ignored when reports are passed.
tolerance	Smallest absolute change in a measure, characteristic or overall score that counts as a change. Defaults to one percentage point.
...	Further arguments passed to [dg_assess()] for both versions, such as 'outlier_method' or 'check_duplicates'. Ignored when reports are passed.
x	A 'dg_comparison'.
row.names, optional	Unused; present for compatibility with the generic.

Details

Both versions are scored against the same [dg_spec()] and the same settings, so a change in a score is a change in the data and not in the yardstick. When two existing 'dg_report' objects are passed instead, they are compared as they are, and a warning is raised if they were produced with different specifications or settings.

A measure is reported as 'declined' or 'improved' only when its value moves by more than 'tolerance'. A measure that is computable in one version and not the other — a date column that disappeared, a declared column that was dropped — is reported as 'lost' or 'gained' rather than as a numeric change, because "not applicable" is not a score of zero.

Alongside the scores the comparison reports the structural changes that most often explain them: columns added or removed, columns whose detected type changed, the row count, the duplicate rate and the missing-value rate of every column the two versions share.

Value

An object of class ‘dg_comparison‘: a list with ‘overall‘, ‘scores‘ and ‘measures‘ (each with ‘old‘, ‘new‘, ‘change‘ and ‘status‘ columns), ‘schema‘ (added, removed and retyped columns), ‘rows‘, ‘duplicates‘, ‘missing‘ (per shared column) and the two reports as ‘old‘ and ‘new‘. ‘as.data.frame()‘ returns the measure table.

Examples

```
old <- airquality
new <- airquality
new$Ozone[1:40] <- NA          # a sensor stopped reporting
new$Wind <- NULL               # a column was dropped upstream
cmp <- dg_compare(old, new)
cmp
cmp$scores
cmp$missing[cmp$missing$status != "unchanged", ]

# With a declared expectation, the accuracy measures are compared too.
spec <- dg_spec(ranges = list(Temp = c(50, 100)))
hot <- airquality
hot$Temp[1:10] <- 120
dg_compare(airquality, hot, spec = spec)$measures[1:3, ]
```

---

dg_plots *Figures for a quality report*

Description

Returns a named list of **ggplot2** objects. Every figure is built from statistics precomputed during [dg_assess()] — bin counts, five-number summaries, a correlation matrix — so the plot objects never carry the raw observations and rendering cost does not grow with the number of rows.

Usage

```
dg_plots(x, which = NULL)

## S3 method for class 'dg_report'
plot(x, y, ...)
```

Arguments

x	A ‘dg_report‘.
which	Names of the figures to build. Defaults to all available.
y	Unused.
...	Unused.

Value

A named list of ggplot objects: ‘scores‘, ‘missing‘, ‘correlation‘, ‘outliers‘, ‘distribution‘.

Examples

```
report <- dg_assess(airquality, verbose = FALSE)
figures <- dg_plots(report)
names(figures)
```

---

dg_save_plots *Save every report figure to a directory*

Description

Save every report figure to a directory

Usage

```
dg_save_plots(x, dir, width = 9, height = 5.5, dpi = 150, device = "png")
```

Arguments

x	A ‘dg_report‘.
dir	Output directory, created if needed. There is no default, so nothing is written unless a path is given.
width, height, dpi	Passed to [ggplot2::ggsave()].
device	File format.

Value

The file paths, invisibly.

Examples

```
report <- dg_assess(iris, verbose = FALSE)
out <- file.path(tempdir(), "datagrade-figures")
paths <- dg_save_plots(report, out)
basename(paths)
unlink(out, recursive = TRUE)
```

dg_spec *Declare what the data is supposed to look like*

Description

Nine of the fifteen ISO/IEC 25012 quality properties cannot be computed from a table alone, because their denominator counts *the data items for which an expectation is defined* — and an expectation is something a person declares, not something a data set contains. ‘dg_spec()‘ is where those declarations go. Pass the result to [dg_assess()] as ‘spec‘.

Usage

```
dg_spec(
  required = NULL,
  expected_records = NULL,
  null_equivalents = NULL,
  patterns = NULL,
  domains = NULL,
  ranges = NULL,
  reference = NULL,
  reference_key = NULL,
  foreign_keys = NULL,
  rules = NULL,
  formats = NULL,
  credible_ranges = NULL,
  credible_domains = NULL,
  source_trust = NULL,
  trusted_sources = NULL,
  update_column = NULL,
  update_interval = NULL,
  max_age = 3652.5,
  as_of = Sys.Date()
)
```

Arguments

required	Character vector of columns that must be populated. Drives record completeness (‘COMP_REG‘) and data value completeness (‘COMP_VAL_ESP‘). Defaults to every column when absent.
expected_records	Expected number of records in a complete file. Drives file completeness (‘COMP_FICH‘).
null_equivalents	Character vector of strings that are structurally present but semantically empty. Drives false completeness (‘FAL_COMP_FICH‘). A conservative default list is used when absent; pass ‘character(0)‘ to disable the check.
patterns	Named list of regular expressions, one per column. A value is syntactically accurate when it matches. Drives ‘EXAC_SINT‘.

`domains` Named list of permitted value sets, one per column. Also drives ‘EXAC_SINT‘; a column may have a pattern, a domain, or both.

`ranges` Named list of ‘c(min, max)‘ numeric intervals, one per column. Drives accuracy range (‘RAN_EXAC‘). This is the *declared* interval ISO requires, as distinct from the interval [dg_assess()] derives from the data for its non-ISO ‘plausibility‘ measure.

`reference` A data frame holding known-correct values, and ‘reference_key‘ the column(s) joining it to the assessed data. Drives semantic accuracy (‘EXAC_SEMAN‘).

`reference_key` Character vector of join columns for ‘reference‘.

`foreign_keys` Named list mapping a column in the data to the permitted value set it must draw from — either a vector of values, or a ‘list(data = <data frame>, column = <name>)‘ pair. Drives referential integrity (‘INT_REF‘).

`rules` Named list of one-sided formulas evaluated per record in the data, each of which must be ‘TRUE‘ for a record to be semantically consistent, e.g. ‘list(order = ~ start_date <= end_date)‘. Drives ‘CONS_SEMAN‘.

`formats` Named list of regular expressions describing the required format of a column. Drives format consistency (‘CONS_FORM‘). Absent columns fall back to the dominant observed format, which is what makes ‘CONS_FORM‘ computable with no spec at all.

`credible_ranges` Named list of ‘c(min, max)‘ intervals outside which a value is not believable even if it is inside the valid range. Drives data values credibility (‘CRED_VAL_DAT‘).

`credible_domains` Named list of believable value sets per column. Also drives ‘CRED_VAL_DAT‘.

`source_trust` Named list, or named character vector, giving the provenance of each column, together with ‘trusted_sources‘ naming the ones considered credible. Drives source credibility (‘CRED_FUEN‘).

`trusted_sources` Character vector of source names deemed credible.

`update_column` Name of the column carrying the update timestamp used by the currentness measures.

`update_interval` Longest acceptable gap between consecutive updates, as a ‘difftime‘ or a number of days. Drives update frequency (‘FREC_ACT‘).

`max_age` Longest acceptable age of a record, as a ‘difftime‘ or a number of days. Drives timeliness of update (‘CONV_ACT‘). Defaults to ten years, which is what makes ‘CONV_ACT‘ computable with no spec.

`as_of` Date the assessment is made relative to. Defaults to today.

Details

Every argument is optional. Each one you supply switches on the measures that depend on it; each one you leave out leaves those measures reported as *not applicable* rather than guessed at. Supplying nothing is legal and gives the same six zero-configuration measures [dg_assess()] computes on its own.

Value

An object of class ‘dg_spec‘.

Examples

```
spec <- dg_spec(
  required = c("Sepal.Length", "Species"),
  ranges   = list(Sepal.Length = c(4, 8)),
  domains  = list(Species = c("setosa", "versicolor", "virginica"))
)
spec
dg_assess(iris, spec = spec, verbose = FALSE)$scores
```

---

dg_vif *Variance inflation factors*

Description

Computed from the diagonal of the inverted correlation matrix, which is algebraically identical to ‘1 / (1 - R^2_j)‘ — the definition given in the dissertation — but costs ‘O(p^3)‘ in the number of columns and is independent of the number of rows. Fitting ‘p‘ auxiliary regressions instead, as the original implementation did through a dummy response, is ‘O(n p^2)‘ and does not finish in reasonable time on millions of rows.

Usage

```
dg_vif(data)
```

Arguments

data	A data frame or numeric matrix.

Value

A named numeric vector of VIF values. Perfectly collinear columns are reported as ‘Inf‘.

Examples

```
dg_vif(iris)
```

`flag_anomalies_in_relationships` *Flag rows whose variable relationships look anomalous*

Description

Fits a random forest predicting the first numeric column from the others and flags rows whose residual exceeds ‘sd_threshold‘ standard deviations. The model is always fitted on a deterministic subsample, then applied to the full data, so cost is bounded by ‘sample_size‘ rather than by the input size.

Usage

```
flag_anomalies_in_relationships(
  data,
  target = NULL,
  sample_size = 20000L,
  sd_threshold = 2,
  ntree = 100L,
  seed = 1L
)
```

Arguments

`data`	A data frame.
`target`	Column to predict. Defaults to the first numeric column.
`sample_size`	Rows used to fit the model.
`sd_threshold`	Residual cut-off in standard deviations.
`ntree`	Number of trees.
`seed`	Seed for both the subsample and the forest.

Details

Requires the suggested **randomForest** package.

Value

A list with the number of anomalies, their share, and example rows; or ‘NULL‘ when the input has too few numeric columns.

flag_redundant_columns *Flag redundant columns by pairwise correlation*

Description

Flag redundant columns by pairwise correlation

Usage

```
flag_redundant_columns(data, correlation_threshold = 0.5)
```

Arguments

data	A data frame.
correlation_threshold	Absolute correlation above which a pair is considered redundant.

Value

A list with the correlation matrix, a data frame of offending pairs, and the columns proposed for removal.

Examples

```
flag_redundant_columns(iris, 0.9)$columns
```

impute_missing_values *Impute missing numeric values*

Description

Impute missing numeric values

Usage

```
impute_missing_values(
  data,
  method = c("mean", "median", "none"),
  group_by = NULL
)
```

Arguments

data	A data frame.
method	One of ‘"mean"‘, ‘"median"‘ or ‘"none"‘.
group_by	Optional column name. When supplied, imputation is performed within each level of that column, falling back to the overall statistic for groups that are entirely missing.

Value

The data frame with numeric ‘NA‘s filled.

Examples

```
colSums(is.na(impute_missing_values(airquality, "median")))
```

---

print.dg_comparison *Print a comparison of two deliveries*

Description

Print a comparison of two deliveries

Usage

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

Arguments

x	A ‘dg_comparison‘ from [dg_compare()].
...	Unused.

Value

‘x‘, invisibly.

print.dg_report *Print a quality report*

Description

Print a quality report

Usage

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

Arguments

x	A ‘dg_report‘.
...	Unused.

Value

‘x‘, invisibly.

summary.dg_report *Tabular summary of a quality report*

Description

Tabular summary of a quality report

Usage

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

Arguments

object	A ‘dg_report‘.
...	Unused.

Value

A list of data frames: ‘scores‘, ‘measures‘, ‘columns‘, ‘redundant_pairs‘ and ‘distribution‘.

theme_dg *A recessive ggplot2 theme for quality-report figures*

Description

Grid and axes are deliberately low-contrast so the marks carry the reading.

Usage

```
theme_dg(base_size = 12)
```

Arguments

base_size	Base font size in points.

Value

A [ggplot2::theme()] object.

validate_dates *Check date columns for impossible and stale values*

Description

Check date columns for impossible and stale values

Usage

```
validate_dates(data, stale_years = 10)
```

Arguments

data	A data frame.
stale_years	Dates older than this many years are counted as stale.

Value

A data frame with one row per date column: number of future dates, number of stale dates, and the observed range.

Index

analyze_correlation, 2
analyze_distribution, 3
analyze_missing_data_patterns, 4
as.data.frame.dg_comparison
        *(*dg_compare*)*, 11
as.data.frame.dg_report, 4

calculate_consistency_score, 5
check_unique_identifiers, 5
clean_text_columns, 6

data_quality_assessment *(*dg_assess*)*, 8
detect_data_types, 6
detect_duplicates, 7
detect_outliers, 8
dg_assess, 8
dg_compare, 11
dg_plots, 12
dg_save_plots, 13
dg_spec, 14
dg_vif, 16

flag_anomalies_in_relationships, 17
flag_redundant_columns, 18

impute_missing_values, 18

plot.dg_report *(*dg_plots*)*, 12
print.dg_comparison, 19
print.dg_report, 20

summary.dg_report, 20

theme_dg, 21

validate_dates, 21