Package: daiquiri (via r-universe)

July 7, 2024

Type Package

Title Data Quality Reporting for Temporal Datasets

Version 1.1.1.9000

Description Generate reports that enable quick visual review of temporal shifts in record-level data. Time series plots showing aggregated values are automatically created for each data field (column) depending on its contents (e.g. min/max/mean values for numeric data, no. of distinct values for categorical data), as well as overviews for missing values, non-conformant values, and duplicated rows. The resulting reports are shareable and can contribute to forming a transparent record of the entire analysis process. It is designed with Electronic Health Records in mind, but can be used for any type of record-level temporal data (i.e. tabular data where each row represents a single ``event", one column contains the ``event date", and other columns contain any associated values for the event).

URL https://github.com/ropensci/daiquiri, https://ropensci.github.io/daiquiri/

BugReports https://github.com/ropensci/daiquiri/issues

License GPL (>= 3)

Encoding UTF-8

Imports data.table (>= 1.12.8), readr (>= 2.0.0), ggplot2 (>= 3.1.0), scales (>= 1.1.0), cowplot (>= 0.9.3), rmarkdown, reactable (>= 0.2.3), utils, stats, xfun (>= 0.15)

RoxygenNote 7.3.1

Suggests covr, knitr, testthat (>= 3.0.0), codemetar

VignetteBuilder knitr

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

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

RemoteUrl https://github.com/ropensci/daiquiri

RemoteRef master

RemoteSha 9f3828e08795ebcc55db076bf329840c70f22182

Contents

`aggregate_data` *Aggregate source data*

Description

Aggregates a `daiquiri_source_data` object based on the `field_types()` specified at load time. Default time period for aggregation is a calendar day

Usage

```
aggregate_data(source_data, aggregation_timeunit = "day", show_progress = TRUE)
```

Arguments

`source_data`	A `daiquiri_source_data` object returned from `prepare_data()` function
`aggregation_timeunit`	Unit of time to aggregate over. Specify one of `"day"`, `"week"`, `"month"`, `"quarter"`, `"year"`. The `"week"` option is Monday-based. Default = `"day"`
`show_progress`	Print progress to console. Default = `TRUE`

Value

A `daiquiri_aggregated_data` object

See Also

prepare_data(), report_data()

Examples

```
# load example data into a data.frame
raw_data <- read_data(
  system.file("extdata", "example_prescriptions.csv", package = "daiquiri"),
  delim = ",",
  col_names = TRUE
)

# validate and prepare the data for aggregation
source_data <- prepare_data(
  raw_data,
  field_types = field_types(
    PrescriptionID = ft_uniqueidentifier(),
    PrescriptionDate = ft_timepoint(),
    AdmissionDate = ft_datetime(includes_time = FALSE),
    Drug = ft_freetext(),
    Dose = ft_numeric(),
    DoseUnit = ft_categorical(),
    PatientID = ft_ignore(),
    Location = ft_categorical(aggregate_by_each_category = TRUE)
  ),
  override_column_names = FALSE,
  na = c("", "NULL")
)

# aggregate the data
aggregated_data <- aggregate_data(
  source_data,
  aggregation_timeunit = "day"
)

aggregated_data
```

---

close_log *Close any active log file*

Description

Close any active log file

Usage

```
close_log()
```

Value

If a log file was found, the path to the log file that was closed, otherwise an empty string

Examples

```
close_log()
```

`daiquiri_report` *Create a data quality report from a data frame*

Description

Accepts record-level data from a data frame, validates it against the expected type of content of each column, generates a collection of time series plots for visual inspection, and saves a report to disk.

Usage

```
daiquiri_report(
  df,
  field_types,
  override_column_names = FALSE,
  na = c("", "NA", "NULL"),
  dataset_description = NULL,
  aggregation_timeunit = "day",
  report_title = "daiquiri data quality report",
  save_directory = ".",
  save_filename = NULL,
  show_progress = TRUE,
  log_directory = NULL
)
```

Arguments

`df`	A data frame. Rectangular data can be read from file using `read_data()`. See Details.
`field_types`	`field_types()` object specifying names and types of fields (columns) in the supplied `df`. See also field_types_available.
`override_column_names`	If `FALSE`, column names in the supplied `df` must match the names specified in `field_types` exactly. If `TRUE`, column names in the supplied `df` will be replaced with the names specified in `field_types`. The specification must therefore contain the columns in the correct order. Default = `FALSE`
`na`	vector containing strings that should be interpreted as missing values, Default = `c("","NA","NULL")`.

dataset_description
: Short description of the dataset being checked. This will appear on the report. If blank, the name of the data frame object will be used

aggregation_timeunit
: Unit of time to aggregate over. Specify one of "day", "week", "month", "quarter", "year". The "week" option is Monday-based. Default = "day"

report_title
: Title to appear on the report

save_directory
: String specifying directory in which to save the report. Default is current directory.

save_filename
: String specifying filename for the report, excluding any file extension. If no filename is supplied, one will be automatically generated with the format daiquiri_report_YYMMDD_HHMMSS

show_progress
: Print progress to console. Default = TRUE

log_directory
: String specifying directory in which to save log file. If no directory is supplied, progress is not logged.

Value

A list containing information relating to the supplied parameters as well as the resulting daiquiri_source_data and daiquiri_aggregated_data objects.

Details

In order for the package to detect any non-conformant values in numeric or datetime fields, these should be present in the data frame in their raw character format. Rectangular data from a text file will automatically be read in as character type if you use the read_data() function. Data frame columns that are not of class character will still be processed according to the field_types specified.

See Also

read_data(), field_types(), field_types_available()

Examples

```
# load example data into a data.frame
raw_data <- read_data(
  system.file("extdata", "example_prescriptions.csv", package = "daiquiri"),
  delim = ",",
  col_names = TRUE
)

# create a report in the current directory
daiq_obj <- daiquiri_report(
  raw_data,
  field_types = field_types(
    PrescriptionID = ft_uniqueidentifier(),
    PrescriptionDate = ft_timepoint(),
    AdmissionDate = ft_datetime(includes_time = FALSE, na = "1800-01-01"),
    Drug = ft_freetext(),
```

```
    Dose = ft_numeric(),
    DoseUnit = ft_categorical(),
    PatientID = ft_ignore(),
    Location = ft_categorical(aggregate_by_each_category = TRUE)
  ),
  override_column_names = FALSE,
  na = c("", "NULL"),
  dataset_description = "Example data provided with package",
  aggregation_timeunit = "day",
  report_title = "daiquiri data quality report",
  save_directory = ".",
  save_filename = "example_data_report",
  show_progress = TRUE,
  log_directory = NULL
)
```

`export_aggregated_data` *Export aggregated data*

Description

Export aggregated data to disk. Creates a separate file for each aggregated field in dataset.

Usage

```
export_aggregated_data(
  aggregated_data,
  save_directory,
  save_file_prefix = "",
  save_file_type = "csv"
)
```

Arguments

`aggregated_data`	A `daiquiri_aggregated_data` object
`save_directory`	String. Full or relative path for save folder
`save_file_prefix`	String. Optional prefix for the exported filenames
`save_file_type`	String. Filetype extension supported by `readr`, currently only csv allowed

Value

(invisibly) The `daiquiri_aggregated_data` object that was passed in

Examples

```
raw_data <- read_data(
  system.file("extdata", "example_prescriptions.csv", package = "daiquiri"),
  delim = ",",
  col_names = TRUE
)

source_data <- prepare_data(
  raw_data,
  field_types = field_types(
    PrescriptionID = ft_uniqueidentifier(),
    PrescriptionDate = ft_timepoint(),
    AdmissionDate = ft_datetime(includes_time = FALSE),
    Drug = ft_freetext(),
    Dose = ft_numeric(),
    DoseUnit = ft_categorical(),
    PatientID = ft_ignore(),
    Location = ft_categorical(aggregate_by_each_category = TRUE)
  ),
  override_column_names = FALSE,
  na = c("", "NULL")
)

aggregated_data <- aggregate_data(
  source_data,
  aggregation_timeunit = "day"
)

export_aggregated_data(
  aggregated_data,
  save_directory = ".",
  save_file_prefix = "ex_"
)
```

`field_types` *Create field_types specification*

Description

Specify the names and types of fields in the source data frame. This is important because the data in each field will be aggregated in different ways, depending on its `field_type`. See field_types_available

Usage

```
field_types(...)
```

Arguments

...	names and types of fields (columns) in source data.

Value

A `field_types` object

See Also

`field_types_available()`, `template_field_types()`

Examples

```
fts <- field_types(
  PatientID = ft_uniqueidentifier(),
  TestID = ft_ignore(),
  TestDate = ft_timepoint(),
  TestName = ft_categorical(aggregate_by_each_category = FALSE),
  TestResult = ft_numeric(),
  ResultDate = ft_datetime(),
  ResultComment = ft_freetext(),
  Location = ft_categorical()
)

fts
```

`field_types_advanced` *Create field_types_advanced specification*

Description

Specify only a subset of the names and types of fields in the source data frame. The remaining fields will be given the same 'default' type.

Usage

```
field_types_advanced(..., .default_field_type = ft_simple())
```

Arguments

...	names and types of fields (columns) in source data.
`.default_field_type`	`field_type` to use for any remaining fields (columns) in source data. Note, this means there can not be a field in the data named `.default_field_type`

Value

A `field_types` object

See Also

field_types(), field_types_available(), template_field_types()

Examples

```
fts <- field_types_advanced(
  PrescriptionDate = ft_timepoint(),
  PatientID = ft_ignore(),
  .default_field_type = ft_simple()
)

fts
```

---

field_types_available *Types of data fields available for specification*

Description

Each column in the source dataset must be assigned to a particular ft_xx depending on the type of data that it contains. This is done through a field_types() specification.

Usage

```
ft_timepoint(includes_time = TRUE, format = "", na = NULL)

ft_uniqueidentifier(na = NULL)

ft_categorical(aggregate_by_each_category = FALSE, na = NULL)

ft_numeric(na = NULL)

ft_datetime(includes_time = TRUE, format = "", na = NULL)

ft_freetext(na = NULL)

ft_simple(na = NULL)

ft_strata(na = NULL)

ft_ignore()
```

Arguments

includes_time	If TRUE, additional aggregated values will be generated using the time portion (and if no time portion is present then midnight will be assumed). If FALSE, aggregated values will ignore any time portion. Default = TRUE

format — Where datetime values are not in the format YYYY-MM-DD or YYYY-MM-DD HH:MM:SS, an alternative format can be specified at the per field level, using readr::col_datetime() format specifications, e.g. format = "%d/%m/%Y". When a format is supplied, it must match the complete string.

na — Column-specific vector of strings that should be interpreted as missing values (in addition to those specified at dataset level)

aggregate_by_each_category — If TRUE, aggregated values will be generated for each distinct subcategory as well as for the field overall. If FALSE, aggregated values will only be generated for the field overall. Default = FALSE

Value

A field_type object denoting the type of data in the column

Details

ft_timepoint() - identifies the data field which should be used as the independent time variable. There should be one and only one of these specified.

ft_uniqueidentifier() - identifies data fields which contain a (usually computer-generated) identifier for an entity, e.g. a patient. It does not need to be unique within the dataset.

ft_categorical() - identifies data fields which should be treated as categorical.

ft_numeric() - identifies data fields which contain numeric values that should be treated as continuous. Any values which contain non-numeric characters (including grouping marks) will be classed as non-conformant

ft_datetime() - identifies data fields which contain date values that should be treated as continuous.

ft_freetext() - identifies data fields which contain free text values. Only presence/missingness will be evaluated.

ft_simple() - identifies data fields where you only want presence/missingness to be evaluated (but which are not necessarily free text).

ft_strata() - identifies a categorical data field which should be used to stratify the rest of the data.

ft_ignore() - identifies data fields which should be ignored. These will not be loaded.

See Also

field_types(), template_field_types()

Examples

```
fts <- field_types(
  PatientID = ft_uniqueidentifier(),
  TestID = ft_ignore(),
  TestDate = ft_timepoint(),
  TestName = ft_categorical(aggregate_by_each_category = FALSE),
  TestResult = ft_numeric(),
  ResultDate = ft_datetime(),
```

```
  ResultComment = ft_freetext(),
  Location = ft_categorical()
)

ft_simple()
```

`initialise_log` *Initialise a log file*

Description

Choose a directory in which to save the log file. If this is not called, no log file is created.

Usage

```
initialise_log(log_directory)
```

Arguments

`log_directory`	String containing directory to save log file

Value

Character string containing the full path to the newly-created log file

Examples

```
log_name <- initialise_log(".")

log_name
```

`prepare_data` *Prepare source data*

Description

Validate a data frame against a `field_types()` specification, and prepare for aggregation.

Usage

```
prepare_data(
  df,
  field_types,
  override_column_names = FALSE,
  na = c("", "NA", "NULL"),
  dataset_description = NULL,
  show_progress = TRUE
)
```

Arguments

df	A data frame
field_types	field_types() object specifying names and types of fields (columns) in the supplied df. See also field_types_available.
override_column_names	If FALSE, column names in the supplied df must match the names specified in field_types exactly. If TRUE, column names in the supplied df will be replaced with the names specified in field_types. The specification must therefore contain the columns in the correct order. Default = FALSE
na	vector containing strings that should be interpreted as missing values. Default = c("","NA","NULL"). Additional column-specific values can be specified in the field_types() object
dataset_description	Short description of the dataset being checked. This will appear on the report. If blank, the name of the data frame object will be used
show_progress	Print progress to console. Default = TRUE

Value

A daiquiri_source_data object

See Also

field_types(), field_types_available(), aggregate_data(), report_data(), daiquiri_report()

Examples

```
# load example data into a data.frame
raw_data <- read_data(
  system.file("extdata", "example_prescriptions.csv", package = "daiquiri"),
  delim = ",",
  col_names = TRUE
)

# validate and prepare the data for aggregation
source_data <- prepare_data(
  raw_data,
  field_types = field_types(
    PrescriptionID = ft_uniqueidentifier(),
    PrescriptionDate = ft_timepoint(),
    AdmissionDate = ft_datetime(includes_time = FALSE),
    Drug = ft_freetext(),
    Dose = ft_numeric(),
    DoseUnit = ft_categorical(),
    PatientID = ft_ignore(),
    Location = ft_categorical(aggregate_by_each_category = TRUE)
  ),
  override_column_names = FALSE,
  na = c("", "NULL"),
```

```
  dataset_description = "Example data provided with package"
)

source_data
```

`read_data` *Read delimited data for optimal use with daiquiri*

Description

Popular file readers such as `readr::read_delim()` perform datatype conversion by default, which can interfere with daiquiri's ability to detect non-conformant values. Use this function instead to ensure optimal compatibility with daiquiri's features.

Usage

```
read_data(
  file,
  delim = NULL,
  col_names = TRUE,
  quote = "\"",
  trim_ws = TRUE,
  comment = "",
  skip = 0,
  n_max = Inf,
  show_progress = TRUE
)
```

Arguments

`file`	A string containing path of file containing data to load, or a URL starting `http://`, `file://`, etc. Compressed files with extension `.gz`, `.bz2`, `.xz` and `.zip` are supported.
`delim`	Single character used to separate fields within a record. E.g. `","` or `"\t"`
`col_names`	Either `TRUE`, `FALSE` or a character vector of column names. If `TRUE`, the first row of the input will be used as the column names, and will not be included in the data frame. If `FALSE`, column names will be generated automatically. Default = `TRUE`
`quote`	Single character used to quote strings.
`trim_ws`	Should leading and trailing whitespace be trimmed from each field?
`comment`	A string used to identify comments. Any text after the comment characters will be silently ignored
`skip`	Number of lines to skip before reading data. If `comment` is supplied any commented lines are ignored after skipping
`n_max`	Maximum number of lines to read.
`show_progress`	Display a progress bar? Default = `TRUE`

Details

This function is aimed at non-expert users of R, and operates as a restricted implementation of `readr::read_delim()`. If you prefer to use `read_delim()` directly, ensure you set the following parameters: `col_types = readr::cols(.default = "c")` and `na = character()`

Value

A data frame

See Also

`field_types()`, `field_types_available()`, `aggregate_data()`, `report_data()`, `daiquiri_report()`

Examples

```
raw_data <- read_data(
  system.file("extdata", "example_prescriptions.csv", package = "daiquiri"),
  delim = ",",
  col_names = TRUE
)

head(raw_data)
```

---

report_data *Generate report from existing objects*

Description

Generate report from previously-created `daiquiri_source_data` and `daiquiri_aggregated_data` objects

Usage

```
report_data(
  source_data,
  aggregated_data,
  report_title = "daiquiri data quality report",
  save_directory = ".",
  save_filename = NULL,
  format = "html",
  show_progress = TRUE,
  ...
)
```

Arguments

source_data	A daiquiri_source_data object returned from prepare_data() function
aggregated_data	A daiquiri_aggregated_data object returned from aggregate_data() function
report_title	Title to appear on the report
save_directory	String specifying directory in which to save the report. Default is current directory.
save_filename	String specifying filename for the report, excluding any file extension. If no filename is supplied, one will be automatically generated with the format daiquiri_report_YYMMDD_HHMMSS
format	File format of the report. Currently only "html" is supported
show_progress	Print progress to console. Default = TRUE
...	Further parameters to be passed to rmarkdown::render(). Cannot include any of input, output_dir, output_file, params, quiet.

Value

A string containing the name and path of the saved report

See Also

prepare_data(), aggregate_data(), daiquiri_report()

Examples

```
# load example data into a data.frame
raw_data <- read_data(
  system.file("extdata", "example_prescriptions.csv", package = "daiquiri"),
  delim = ",",
  col_names = TRUE
)

# validate and prepare the data for aggregation
source_data <- prepare_data(
  raw_data,
  field_types = field_types(
    PrescriptionID = ft_uniqueidentifier(),
    PrescriptionDate = ft_timepoint(),
    AdmissionDate = ft_datetime(includes_time = FALSE),
    Drug = ft_freetext(),
    Dose = ft_numeric(),
    DoseUnit = ft_categorical(),
    PatientID = ft_ignore(),
    Location = ft_categorical(aggregate_by_each_category = TRUE)
  ),
  override_column_names = FALSE,
  na = c("", "NULL"),
  dataset_description = "Example data provided with package",
  show_progress = TRUE
```

```
)

# aggregate the data
aggregated_data <- aggregate_data(
  source_data,
  aggregation_timeunit = "day",
  show_progress = TRUE
)

# save a report in the current directory using the previously-created objects
report_data(
  source_data,
  aggregated_data,
  report_title = "daiquiri data quality report",
  save_directory = ".",
  save_filename = "example_data_report",
  show_progress = TRUE
)
```

`template_field_types` *Print a template field_types() specification to console*

Description

Helper function to generate template code for a `field_types()` specification, based on the supplied data frame. All fields (columns) in the specification will be defined using the `default_field_type`, and the console output can be copied and edited before being used as input to `daiquiri_report()` or `prepare_data()`.

Usage

```
template_field_types(df, default_field_type = ft_ignore())
```

Arguments

`df`	data frame including the column names for the template specification
`default_field_type`	`field_type` to be used for each column. Default = `ft_ignore()`. See `field_types_available()`

Value

(invisibly) Character string containing the template code

See Also

`field_types()`

Examples

```
df <- data.frame(
  col1 = rep("2022-01-01", 5),
  col2 = rep(1, 5),
  col3 = 1:5,
  col4 = rnorm(5)
)

template_field_types(df, default_field_type = ft_numeric())
```

Index