Package: vivaglint (via r-universe)

June 3, 2026

Type Package

Title Analysis Tools for 'Viva Glint' Survey Data

Version 0.1.1

Description Provides functions for importing, validating, and analyzing 'Viva Glint' survey data exports, with optional API-based import via the 'Microsoft Graph' API. Includes tools for data reshaping, question-level analysis, multi-cycle comparisons, organizational hierarchy analysis, factor analysis, and correlation analysis. Harman (1960, ISBN: 0226316513); Husser (2017) <doi:10.1002/9781118901731.iecrm0048>.

License MIT + file LICENSE

URL https://github.com/microsoft/vivaglint

BugReports https://github.com/microsoft/vivaglint/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports dplyr (>= 1.0.0), tidyr (>= 1.0.0), readr (>= 2.0.0), stringr (>= 1.4.0), lubridate (>= 1.7.0), purrr (>= 0.3.0), rlang (>= 0.4.0), httr (>= 1.4.0)

Suggests testthat (>= 3.0.0), knitr, rmarkdown, psych (>= 2.0.0), ggplot2 (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/pak/sysreqs libicu-dev libssl-dev libx11-dev

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

Date/Publication 2026-06-03 13:41:35 UTC

RemoteUrl https://github.com/microsoft/vivaglint

RemoteRef HEAD

RemoteSha 2af479d7e273907b8d59580cb21a20118572cf63

Contents

`aggregate_by_manager` *Aggregate Responses by Manager*

Description

Rolls up survey responses to the manager level, calculating the same metrics as summarize_survey() for each manager's team.

Usage

```
aggregate_by_manager(
  survey,
  scale_points,
  emp_id_col = NULL,
  manager_id_col,
  full_tree = FALSE,
  plot = FALSE
)
```

Arguments

`survey`	A glint_survey object or data frame containing survey data
`scale_points`	Integer specifying the number of scale points (2-11)
`emp_id_col`	Character string specifying the employee ID column name
`manager_id_col`	Character string specifying the manager ID column name

`full_tree` Logical indicating whether to include full subtree (all indirect reports) or only direct reports (default: FALSE)

`plot` Logical. If `TRUE`, prints a ranked lollipop chart of team Glint Scores by manager and returns the data invisibly. When multiple questions are present the chart is faceted by question. Requires **ggplot2**. Default: `FALSE`.

Value

A tibble with one row per manager-question combination containing:

manager_id The manager's employee ID

manager_name The manager's full name (First Name + Last Name)

question The question text

team_size Number of employees in the team (direct reports only, or full subtree when `full_tree = TRUE`)

mean, sd, glint_score, n_responses, n_skips, n_total Descriptive statistics for this manager's team on this question

pct_favorable, pct_neutral, pct_unfavorable Favorability percentages for this manager's team on this question

When `plot = TRUE`, the same tibble is returned invisibly after printing the plot.

Examples

```
survey_path <- system.file("extdata", "survey_export.csv", package = "vivaglint")
survey <- read_glint_survey(survey_path, emp_id_col = "EMP ID")

# Direct reports only
manager_summary <- aggregate_by_manager(survey, scale_points = 5,
                                        emp_id_col = "EMP ID",
                                        manager_id_col = "Manager ID")

# Full organizational tree
manager_summary_full <- aggregate_by_manager(survey, scale_points = 5,
                                             emp_id_col = "EMP ID",
                                             manager_id_col = "Manager ID",
                                             full_tree = TRUE)

# With ranked dot plot
if (requireNamespace("ggplot2", quietly = TRUE)) {
  aggregate_by_manager(survey, scale_points = 5, emp_id_col = "EMP ID",
                       manager_id_col = "Manager ID", plot = TRUE)
}
```

`analyze_attrition` *Analyze Employee Attrition*

Description

Analyzes the relationship between survey responses and employee attrition, calculating how much more (or less) likely employees are to leave within specified time periods if they respond unfavorably vs favorably to survey questions.

Usage

```
analyze_attrition(
  survey,
  attrition_file,
  emp_id_col = NULL,
  term_date_col,
  scale_points,
  time_periods = c(90, 180, 365),
  attribute_cols = NULL,
  min_group_size = 5,
  plot = FALSE
)
```

Arguments

`survey`	A glint_survey object or data frame containing survey data
`attrition_file`	Path to CSV file containing employee attrition data
`emp_id_col`	Character string specifying the column name for employee ID
`term_date_col`	Character string specifying the column name for termination date
`scale_points`	Integer specifying the number of scale points (2-11)
`time_periods`	Integer vector specifying time periods in days to analyze (default: c(90, 180, 365))
`attribute_cols`	Optional character vector of attribute column names to segment results by (e.g., `c("Department", "Gender")`). Attributes must already be joined to the survey via `join_attributes()`. When `NULL` (default), results are returned for the overall population.
`min_group_size`	Minimum number of employees required in an attribute group for it to be included in results (default: 5). Ignored when `attribute_cols = NULL`.
`plot`	Logical. If `TRUE`, prints a grouped bar chart of favorable vs. unfavorable attrition rates faceted by time period and returns the data invisibly. When `attribute_cols` is supplied, the first attribute column is used as an additional facet dimension. Requires **ggplot2**. Default: `FALSE`.

Value

A tibble with one row per (attribute group)-question-time period combination containing:

attribute columns One column per entry in `attribute_cols` (only present when `attribute_cols` is supplied)

group_size Number of employees in the attribute group (only present when `attribute_cols` is supplied)

question The question text

days The time period in days

favorable_n Number of employees who responded favorably

favorable_attrition Proportion who left within time period (favorable)

unfavorable_n Number of employees who responded unfavorably

unfavorable_attrition Proportion who left within time period (unfavorable)

attrition_ratio Ratio of unfavorable to favorable attrition rates

When `plot = TRUE`, the same tibble is returned invisibly after printing the plot.

Examples

```
survey_path <- system.file("extdata", "survey_export.csv", package = "vivaglint")
attrition_file <- system.file("extdata", "attrition.csv", package = "vivaglint")
attr_path <- system.file("extdata", "employee_attributes.csv", package = "vivaglint")
survey <- read_glint_survey(survey_path, emp_id_col = "EMP ID")

# Overall attrition analysis
attrition <- analyze_attrition(
  survey,
  attrition_file = attrition_file,
  emp_id_col = "EMP ID",
  term_date_col = "Termination Date",
  scale_points = 5
)

# Attrition segmented by Department and Gender
survey_enriched <- join_attributes(survey, attr_path, emp_id_col = "EMP ID")
attrition_by_dept <- analyze_attrition(
  survey_enriched,
  attrition_file = attrition_file,
  emp_id_col = "EMP ID",
  term_date_col = "Termination Date",
  scale_points = 5,
  attribute_cols = c("Department", "Gender"),
  min_group_size = 2
)

# With attrition chart
if (requireNamespace("ggplot2", quietly = TRUE)) {
  analyze_attrition(survey, attrition_file = attrition_file,
                    emp_id_col = "EMP ID", term_date_col = "Termination Date",
```

```
                    scale_points = 5, plot = TRUE)
}
```

`analyze_by_attributes` *Analyze Survey Responses by Attributes*

Description

Aggregates survey responses by employee attributes (e.g., Department, Gender, Tenure) and calculates the same metrics as `summarize_survey()` for each attribute group combination. Only groups meeting the minimum size threshold are included in the results.

Usage

```
analyze_by_attributes(
  survey,
  attribute_file = NULL,
  scale_points,
  attribute_cols,
  emp_id_col = NULL,
  min_group_size = 5,
  plot = FALSE
)
```

Arguments

`survey`	A `glint_survey` object or data frame. If attributes have already been joined via `join_attributes()`, `attribute_file` can be omitted.
`attribute_file`	Optional. A file path (character string) or data frame containing employee attributes to join. If `NULL` (default), the survey must already contain the columns named in `attribute_cols`.
`scale_points`	Integer specifying the number of scale points (2-11)
`attribute_cols`	Character vector of column names to group by (e.g., `c("Department", "Gender", "Tenure Group")`)
`emp_id_col`	Character string specifying the employee ID column name in both the survey data and the attribute data
`min_group_size`	Integer specifying the minimum number of employees required for a group to be included in results (default: 5)
`plot`	Logical. If `TRUE`, prints a faceted dot plot of Glint Scores by attribute group (one facet per question) and returns the data invisibly. When multiple `attribute_cols` are supplied, only the first is plotted. Requires **ggplot2**. Default: `FALSE`.

Details

Attributes can be supplied in two ways:

1. Pass `attribute_file` (a file path or data frame) and the join is performed internally on each call.
2. Pre-join attributes once with `join_attributes()` and pass the enriched survey directly — omit `attribute_file` entirely. This is more efficient when calling `analyze_by_attributes()` multiple times or when you want to filter the data before analysis.

Value

A tibble with one row per attribute-group-question combination containing:

attribute columns Values for each attribute grouping variable

group_size Number of employees in this attribute group

question, mean, sd, n_responses, n_skips, n_total Descriptive statistics for this group on this question

pct_favorable, pct_neutral, pct_unfavorable Favorability percentages for this group on this question

When `plot = TRUE`, the same tibble is returned invisibly after printing the plot.

Examples

```
survey_path <- system.file("extdata", "survey_export.csv", package = "vivaglint")
attr_file <- system.file("extdata", "employee_attributes.csv", package = "vivaglint")
survey <- read_glint_survey(survey_path, emp_id_col = "EMP ID")

# Option 1: provide attribute_file directly
results <- analyze_by_attributes(
  survey,
  attribute_file = attr_file,
  scale_points = 5,
  attribute_cols = "Department",
  emp_id_col = "EMP ID",
  min_group_size = 2
)

# Option 2: pre-join with join_attributes(), then omit attribute_file
survey_enriched <- join_attributes(survey, attr_file, emp_id_col = "EMP ID")
results <- analyze_by_attributes(survey_enriched, scale_points = 5,
                                 attribute_cols = "Department",
                                 emp_id_col = "EMP ID",
                                 min_group_size = 2)

# With dot plot
if (requireNamespace("ggplot2", quietly = TRUE)) {
  analyze_by_attributes(survey_enriched, scale_points = 5,
                        attribute_cols = "Department",
                        emp_id_col = "EMP ID", plot = TRUE,
```

```
                        min_group_size = 2)
}
```

`compare_cycles` *Compare Survey Cycles*

Description

Compares question-level metrics across multiple survey cycles, calculating change scores and trends over time.

Usage

```
compare_cycles(..., scale_points, cycle_names = NULL, plot = FALSE)
```

Arguments

`...`	Two or more glint_survey objects or data frames to compare
`scale_points`	Integer specifying the number of scale points (2-11)
`cycle_names`	Optional character vector of names for each cycle. If not provided, will use "Cycle 1", "Cycle 2", etc.
`plot`	Logical. If `TRUE`, prints a line chart of Glint Score over cycles (one line per question) and returns the data invisibly. Requires **ggplot2**. Default: `FALSE`.

Value

A tibble with one row per question-cycle combination containing:

cycle The cycle name or number

question The question text

mean, sd, n_responses, n_skips, n_total Descriptive statistics for this question in this cycle (see `summarize_survey()`)

pct_favorable, pct_neutral, pct_unfavorable Favorability percentages for this question in this cycle

change_from_previous Change in raw mean score from the previous cycle (NA for the first cycle)

pct_change_from_previous Percentage change in raw mean score from the previous cycle (NA for the first cycle)

glint_score_change_from_previous Change in Glint Score (0-100 scale) from the previous cycle (NA for the first cycle)

When `plot = TRUE`, the same tibble is returned invisibly after printing the plot.

Examples

```
survey_path <- system.file("extdata", "survey_export.csv", package = "vivaglint")
survey1 <- read_glint_survey(survey_path, emp_id_col = "EMP ID")
survey2 <- read_glint_survey(survey_path, emp_id_col = "EMP ID")
survey3 <- read_glint_survey(survey_path, emp_id_col = "EMP ID")

comparison <- compare_cycles(survey1, survey2, survey3,
                             scale_points = 5,
                             cycle_names = c("Q1 2024", "Q2 2024", "Q3 2024"))
print(comparison)

# With trend chart
if (requireNamespace("ggplot2", quietly = TRUE)) {
  compare_cycles(survey1, survey2, survey3, scale_points = 5,
                 cycle_names = c("Q1 2024", "Q2 2024", "Q3 2024"),
                 plot = TRUE)
}
```

---

extract_questions *Extract Questions from Glint Survey*

Description

Parses column names to extract unique questions and their associated column names.

Usage

```
extract_questions(data, emp_id_col = NULL)
```

Arguments

data	A data frame or glint_survey object containing survey data
emp_id_col	Character string specifying the employee ID column name. When data is a glint_survey object this is resolved automatically from data$metadata$emp_id_col. Required when data is a plain data frame.

Value

A tibble with one row per question containing:

question The question text

response_col Column name for numeric responses

comment_col Column name for comments

topics_col Column name for comment topics

flag_col Column name for sensitive comment flags

Examples

```
survey_path <- system.file("extdata", "survey_export.csv", package = "vivaglint")
survey <- read_glint_survey(survey_path, emp_id_col = "EMP ID")
questions <- extract_questions(survey)
print(questions)
```

`extract_survey_factors` *Extract Survey Factors*

Description

Performs factor analysis on survey question responses to identify underlying latent factors. Supports multiple rotation methods with oblique rotation as default.

Usage

```
extract_survey_factors(
  survey,
  n_factors = NULL,
  rotation = "oblimin",
  min_loading = 0.3,
  fm = "minres",
  plot = FALSE
)
```

Arguments

`survey`	A glint_survey object or data frame containing survey data
`n_factors`	Integer indicating the number of factors to extract. If NULL (default), will use parallel analysis to determine optimal number of factors
`rotation`	Character string indicating rotation method: "oblimin" (default), "varimax", "promax", "quartimax", "equamax", or "none"
`min_loading`	Minimum factor loading to display in results (default: 0.3)
`fm`	Character string indicating factoring method: "minres" (minimum residuals, default), "ml" (maximum likelihood), "pa" (principal axis), "wls" (weighted least squares), "gls" (generalized least squares), or "uls" (unweighted least squares)
`plot`	Logical. If `TRUE`, prints a factor loading heatmap and returns the result list invisibly. Requires **ggplot2**. Default: `FALSE`.

Value

A list containing:

factor_summary A tibble with one row per question-factor combination (filtered to `abs(loading) >= min_loading`), containing: `question` (item text), `factor` (factor name), `loading` (loading coefficient), `loading_label` ("Strong" >= 0.75 / "Medium" 0.60-0.74 / "Weak" < 0.60), `communality` (proportion of item variance explained by all factors), and `factor_variance_pct` (percentage of total variance explained by this factor). Sorted by factor then descending loading strength.

fa_object Original fa object from psych package for further analysis

When `plot = TRUE`, the same list is returned invisibly after printing the plot.

Examples

```
survey_path <- system.file("extdata", "survey_export.csv", package = "vivaglint")
survey <- read_glint_survey(survey_path, emp_id_col = "EMP ID")

if (requireNamespace("psych", quietly = TRUE)) {
  # Extract factors with fixed count to keep runtime small
  factors <- extract_survey_factors(survey, n_factors = 1)
  print(factors$factor_summary)

  # Filter to strong loaders only
  strong <- dplyr::filter(factors$factor_summary, loading_label == "Strong")

  # With factor loading heatmap
  if (requireNamespace("ggplot2", quietly = TRUE)) {
    extract_survey_factors(survey, n_factors = 1, plot = TRUE)
  }
}
```

---

get_correlations *Calculate Question Correlations*

Description

Calculates correlations between all question response columns in the survey. Supports multiple correlation methods and output formats.

Usage

```
get_correlations(
  survey,
  method = "pearson",
  format = "long",
  use = "pairwise.complete.obs",
  plot = FALSE
)
```

Arguments

survey	A glint_survey object or data frame containing survey data
method	Character string indicating the correlation method: "pearson" (default), "spearman", or "kendall"
format	Character string indicating output format: "long" for long format with one row per question pair (default), or "matrix" for traditional correlation matrix
use	Character string indicating how to handle missing values, passed to cor() function (default: "pairwise.complete.obs")
plot	Logical. If TRUE, prints a correlation heatmap and returns the data invisibly. Only supported when format = "long" (the default); ignored with a warning when format = "matrix". Requires **ggplot2**. Default: FALSE.

Value

If format = "long", a tibble with columns:

question1 First question text

question2 Second question text

correlation Correlation coefficient

p_value P-value for test of correlation significance

n Number of complete pairs used in calculation

If format = "matrix", a matrix with questions as rows and columns. When plot = TRUE and format = "long", the tibble is returned invisibly after printing the plot.

Examples

```
survey_path <- system.file("extdata", "survey_export.csv", package = "vivaglint")
survey <- read_glint_survey(survey_path, emp_id_col = "EMP ID")

# Get Pearson correlations in long format (default)
correlations <- get_correlations(survey)

# Get Spearman correlations
correlations_spearman <- get_correlations(survey, method = "spearman")

# Get correlation matrix
cor_matrix <- get_correlations(survey, format = "matrix")

# With correlation heatmap
if (requireNamespace("ggplot2", quietly = TRUE)) {
  get_correlations(survey, plot = TRUE)
}
```

`get_response_dist` *Get Response Distribution*

Description

Calculates the distribution of response values for survey questions, returning counts and percentages for each response value in tidy format.

Usage

```
get_response_dist(survey, questions = "all", plot = FALSE)
```

Arguments

`survey`	A glint_survey object or data frame containing survey data
`questions`	Character vector of question text(s) to analyze, or "all" to analyze all questions (default: "all")
`plot`	Logical. If `TRUE`, prints a stacked bar chart of response value distributions (red → blue gradient) and returns the data invisibly. Requires **ggplot2**. Default: `FALSE`.

Value

A tibble with one row per question containing:

question The question text

count_X Count of responses with value X (for each unique response value)

pct_X Percentage of responses with value X (for each unique response value)

When `plot = TRUE`, the same tibble is returned invisibly after printing the plot.

Examples

```
survey_path <- system.file("extdata", "survey_export.csv", package = "vivaglint")
survey <- read_glint_survey(survey_path, emp_id_col = "EMP ID")

# Get response distribution for all questions
dist <- get_response_dist(survey)

# Get response distribution for specific questions
dist_subset <- get_response_dist(survey,
  questions = c("My work is meaningful", "I feel valued"))

# With distribution chart
if (requireNamespace("ggplot2", quietly = TRUE)) {
  get_response_dist(survey, plot = TRUE)
}
```

`glint_setup` *Configure Viva Glint API credentials*

Description

Stores Viva Glint API credentials in environment variables for the current R session. Optionally writes the values to ~/.Renviron for persistence.

Usage

```
glint_setup(
  tenant_id,
  client_id,
  client_secret,
  experience_name,
  save_to_renviron = FALSE
)
```

Arguments

`tenant_id`	Azure AD tenant ID
`client_id`	Azure AD app (client) ID
`client_secret`	Azure AD app client secret
`experience_name`	Viva Glint experience name (e.g., "contoso@demo")
`save_to_renviron`	Logical; if TRUE, append values to ~/.Renviron

Details

Required environment variables:

- GLINT_TENANT_ID
- GLINT_CLIENT_ID
- GLINT_CLIENT_SECRET
- GLINT_EXPERIENCE_NAME

Value

Invisibly returns TRUE after saving credentials

Examples

```
## Not run:
glint_setup(
  tenant_id = "your-tenant-id",
  client_id = "your-client-id",
  client_secret = "your-client-secret",
  experience_name = "your-experience-name",
  save_to_renviron = TRUE
)

## End(Not run)
```

`join_attributes` *Join Employee Attributes to Survey Data*

Description

Reads employee attribute data from a CSV file or data frame and joins it to a survey object by employee ID. Returns an enriched `glint_survey` object that can be passed directly to `analyze_by_attributes()` or any other package function, eliminating the need to re-read and re-join the attribute file on every call.

Usage

```
join_attributes(survey, attribute_source, emp_id_col = NULL)
```

Arguments

`survey`	A `glint_survey` object or data frame containing survey data
`attribute_source`	Either a character string file path to a CSV file, or a data frame containing employee attributes. All columns are coerced to character to avoid type conflicts during joining.
`emp_id_col`	Character string specifying the employee ID column name. Must match the column name in both the survey data and the attribute data. When `survey` is a `glint_survey` object this defaults to `survey$metadata$emp_id_col` (set at import via `read_glint_survey()`), so you can omit this argument if you loaded the survey with the correct `emp_id_col`.

Value

If `survey` is a `glint_survey` object, returns an enriched `glint_survey` with the attribute columns appended to `$data` and the names of all joined attribute columns stored in `$metadata$attribute_cols`. If `survey` is a plain data frame, returns the joined data frame.

Respondents with no match in the attribute data will have `NA` for all attribute columns; a message is emitted indicating how many were affected.

Examples

```
survey_path <- system.file("extdata", "survey_export.csv", package = "vivaglint")
attr_path <- system.file("extdata", "employee_attributes.csv", package = "vivaglint")
survey <- read_glint_survey(survey_path, emp_id_col = "EMP ID")

survey_enriched <- join_attributes(survey, attr_path, emp_id_col = "EMP ID")
results <- analyze_by_attributes(survey_enriched, scale_points = 5,
                                 attribute_cols = "Department")
```

`pivot_long` *Pivot Survey Data to Long Format*

Description

Transforms survey data from wide format (one row per respondent) to long format (one row per respondent-question combination). Can return all responses, only comments, or both depending on the data_type parameter.

Usage

```
pivot_long(
  survey,
  data_type = "all",
  include_empty = FALSE,
  include_standard_cols = TRUE
)
```

Arguments

`survey`	A glint_survey object or data frame containing survey data
`data_type`	Character string indicating what data to return: "all" for all responses including those without comments, "comments" for only responses with comments, or "both" for separate tibbles (default: "all")
`include_empty`	Logical indicating whether to include empty comments when data_type = "comments" (default: FALSE)
`include_standard_cols`	Logical indicating whether to include standard columns (EMP ID, Manager ID, etc.) in the output (default: TRUE)

Value

When `data_type = "all"` or `"comments"`, a single tibble in long format with columns:

Standard columns EMP ID, First Name, Last Name, etc. (if `include_standard_cols = TRUE`)

question The question text

response Numeric response value

comment Comment text (NA if not provided)

comment_topics Comma-separated topic tags (NA if not provided)

sensitive_flag Sensitivity flag value

When `data_type = "both"`, a named list with two elements:

all Long-format tibble of all responses (same structure as above)

comments Long-format tibble filtered to rows with non-empty comments

Examples

```
survey_path <- system.file("extdata", "survey_export.csv", package = "vivaglint")
survey <- read_glint_survey(survey_path, emp_id_col = "EMP ID")

# Get all responses (with and without comments)
all_data <- pivot_long(survey)

# Get only responses with comments
comments_only <- pivot_long(survey, data_type = "comments")

# Get both separately
both <- pivot_long(survey, data_type = "both")
all_responses <- both$all
comments <- both$comments
```

---

read_glint_survey *Read Viva Glint Survey Data*

Description

Reads a Viva Glint survey export CSV file with automatic validation and parsing. This function validates the file structure, parses dates, and organizes the data into a structured format ready for analysis. To import directly from the Viva Glint API, use `read_glint_survey_api()`.

Usage

```
read_glint_survey(
  file_path,
  emp_id_col = NULL,
  first_name_col = "First Name",
  last_name_col = "Last Name",
  email_col = "Email",
  status_col = "Status",
  completion_date_col = "Survey Cycle Completion Date",
  sent_date_col = "Survey Cycle Sent Date",
  encoding = "UTF-8"
)
```

Arguments

file_path	Character string specifying the path to the CSV file
emp_id_col	Character string specifying the name of the employee ID column in the survey export (e.g., "Employee ID", "EmployeeID", "emp_id"). The column name varies by organization. The value is stored in survey$metadata$emp_id_col and used automatically by downstream functions so you do not need to repeat it on every call.
first_name_col	Column name for first name (default: "First Name")
last_name_col	Column name for last name (default: "Last Name")
email_col	Column name for email (default: "Email")
status_col	Column name for status (default: "Status")
completion_date_col	Column name for survey completion date (default: "Survey Cycle Completion Date")
sent_date_col	Column name for survey sent date (default: "Survey Cycle Sent Date")
encoding	Character string specifying file encoding (default: "UTF-8")

Value

A glint_survey object (an S3 class extending list) with two elements:

data A tibble containing all survey responses. Date columns are parsed automatically from DD-MM-YYYY HH:MM format to POSIXct.

metadata A list with five elements: standard_columns (the eight standard Glint column names), questions (a tibble from extract_questions() with one row per question), n_respondents (total row count), n_questions (number of questions), and file_path (the path originally supplied).

Examples

```
survey_path <- system.file("extdata", "survey_export.csv", package = "vivaglint")
survey <- read_glint_survey(survey_path, emp_id_col = "EMP ID")
head(survey$data)
survey$metadata$questions
```

---

read_glint_survey_api *Read Viva Glint Survey Data via API*

Description

Exports survey data through the Microsoft Graph beta API, downloads the resulting ZIP archive, and returns either a glint_survey object or a named list of them depending on what the export contains. This is an alternative to read_glint_survey() when you want to pull data directly from Viva Glint instead of importing a local CSV export.

Usage

```
read_glint_survey_api(
  survey_uuid = NULL,
  cycle_id = NULL,
  mode = NULL,
  emp_id_col = NULL,
  first_name_col = "First Name",
  last_name_col = "Last Name",
  email_col = "Email",
  status_col = "Status",
  completion_date_col = "Survey Cycle Completion Date",
  sent_date_col = "Survey Cycle Sent Date",
  start_date = NULL,
  end_date = NULL,
  encoding = "UTF-8",
  poll_interval = 10,
  max_attempts = 60,
  save_zip_to = NULL,
  experience_name = NULL
)
```

Arguments

survey_uuid	Survey UUID from Viva Glint. Falls back to GLINT_SURVEY_UUID. Required for "cycle" and "survey" modes.
cycle_id	Survey cycle ID. Falls back to GLINT_CYCLE_ID. Required for "cycle" mode.
mode	One of "cycle", "survey", "daterange". Falls back to GLINT_MODE. If still unset, inferred from which identifiers are populated. Explicit arguments always win.
emp_id_col	Character string specifying the employee ID column name.
first_name_col	Column name for first name (default: "First Name")
last_name_col	Column name for last name (default: "Last Name")
email_col	Column name for email (default: "Email")
status_col	Column name for status (default: "Status")
completion_date_col	Column name for survey completion date (default: "Survey Cycle Completion Date")
sent_date_col	Column name for survey sent date (default: "Survey Cycle Sent Date")
start_date	Optional start date/time for the export window. Falls back to GLINT_START_DATE. Can be a character string in ISO 8601 format, or a Date/POSIXct value.
end_date	Optional end date/time for the export window. Falls back to GLINT_END_DATE.
encoding	Character string specifying file encoding (default: "UTF-8")
poll_interval	Seconds to wait between status checks (default: 10)
max_attempts	Maximum number of polling attempts (default: 60)

`save_zip_to` Optional path. When set, the raw export zip Microsoft Graph returns is also written to disk before being parsed into R data frames — useful for audit trails or feeding other tools. If the path is an existing directory (or ends with a path separator), the file is named `glint-export-{job_id}.zip` inside it; otherwise the path is treated as a full file path. Falls back to `GLINT_SAVE_ZIP_TO` when the argument is omitted. Defaults to `NULL` (no zip is persisted).

`experience_name` Optional Viva Glint experience name to override the GLINT_EXPERIENCE_NAME environment variable

Value

Single-CSV exports (typical for `"cycle"` mode) return a `glint_survey` object with the same structure as `read_glint_survey()` produces. Multi-CSV exports (typical for `"survey"` and `"daterange"` modes) return a named list of `glint_survey` objects keyed by source CSV filename. CSVs that do not fit the standard GlintSurvey schema (e.g. supplementary metadata or attribute files) are returned as plain `data.frame` entries in that list with a warning.

Modes

Three export shapes are supported via the `mode` argument:

- `"cycle"` — exports a single survey cycle. Requires `survey_uuid` and `cycle_id`. This is the existing behavior.
- `"survey"` — exports every cycle of one survey. Requires `survey_uuid`.
- `"daterange"` — exports every survey in the experience that has activity between `start_date` and `end_date`. Both dates are optional; if both are omitted, the API applies its default window (about the last six months).

When `mode` is `NULL`, it is read from the `GLINT_MODE` env var; if that is also unset, it is inferred from which identifiers are populated (both IDs => `"cycle"`, survey UUID only => `"survey"`, neither => `"daterange"`). This keeps the existing positional call `read_glint_survey_api(survey_uuid, cycle_id)` working unchanged.

Environment variable fallbacks

Any input not supplied as a function argument is read from the matching environment variable, so the typical call only specifies what differs from the values in `.Renviron`:

- `survey_uuid` <- `GLINT_SURVEY_UUID`
- `cycle_id` <- `GLINT_CYCLE_ID`
- `mode` <- `GLINT_MODE`
- `start_date` <- `GLINT_START_DATE`
- `end_date` <- `GLINT_END_DATE`
- `save_zip_to` <- `GLINT_SAVE_ZIP_TO`

Explicit arguments always win over env vars.

Examples

```
## Not run:
glint_setup(
  tenant_id = "your-tenant-id",
  client_id = "your-client-id",
  client_secret = "your-client-secret",
  experience_name = "your-experience-name"
)

# Cycle mode (also the default if you pass both IDs):
cycle_survey <- read_glint_survey_api(
  survey_uuid = "your-survey-uuid",
  cycle_id = "your-cycle-id",
  emp_id_col = "Employment ID"
)

# Survey mode: every cycle of one survey, returned as a named list.
all_cycles <- read_glint_survey_api(
  mode = "survey",
  survey_uuid = "your-survey-uuid",
  emp_id_col = "Employment ID"
)

# Date-range mode, with everything else read from .Renviron
# (GLINT_START_DATE, GLINT_END_DATE):
recent <- read_glint_survey_api(
  mode = "daterange",
  emp_id_col = "Employment ID"
)

## End(Not run)
```

---

search_comments *Search Survey Comments*

Description

Searches through all survey comment text and returns matching responses with their associated question text, numeric response values, comments, and topics. Supports both exact substring matching and fuzzy (approximate) matching that tolerates minor spelling differences.

Usage

```
search_comments(survey, query, exact = FALSE, max_distance = 0.2)
```

Arguments

survey	A glint_survey object or data frame containing survey data
query	Character string to search for within comments

exact	Logical. If TRUE, performs case-sensitive exact substring matching. If FALSE (default), performs case-insensitive approximate matching that also tolerates minor spelling differences.
max_distance	Numeric between 0 and 1 controlling fuzzy match tolerance when exact = FALSE. Higher values allow more differences (e.g. more typos). Default is 0.2.

Value

A tibble with one row per matching comment containing:

question The survey question text

response The numeric survey response value

comment The comment text that matched the query

topics The comment topic(s) associated with the comment

Returns an empty tibble with the same columns if no matches are found.

Examples

```
survey_path <- system.file("extdata", "survey_export.csv", package = "vivaglint")
survey <- read_glint_survey(survey_path, emp_id_col = "EMP ID")

# Fuzzy search (default) - finds "manager", "Manager", "managers", etc.
results <- search_comments(survey, "manager")

# Exact search - finds only the literal string "Manager" (case-sensitive)
results_exact <- search_comments(survey, "Manager", exact = TRUE)

# Wider fuzzy tolerance to catch more spelling variations
results_wide <- search_comments(survey, "management", max_distance = 0.3)
```

---

split_survey_data *Split Survey Data into Quantitative and Qualitative Components*

Description

Separates a survey dataset into two data frames: one containing only numeric response data (for statistical analysis) and one containing only comment and topic data (for qualitative analysis). Both outputs retain the employee ID column so they can be rejoined at any time.

Usage

```
split_survey_data(survey, emp_id_col = NULL)
```

Arguments

survey	A glint_survey object or data frame containing survey data
emp_id_col	Character string specifying the employee ID column name

Value

A named list with two elements:

quantitative A tibble with all standard respondent columns plus one numeric response column per question. Comment, topic, and sensitive flag columns are excluded.

qualitative A tibble with the employee ID column plus the comment, topic, and sensitive flag columns for every question. Numeric response columns are excluded.

Examples

```
survey_path <- system.file("extdata", "survey_export.csv", package = "vivaglint")
survey <- read_glint_survey(survey_path, emp_id_col = "EMP ID")
parts  <- split_survey_data(survey)

# Analyze numeric responses
summary <- summarize_survey(parts$quantitative, scale_points = 5,
                            emp_id_col = "EMP ID")

# Work with comments separately
comments <- parts$qualitative
```

`summarize_survey` *Summarize Survey Questions*

Description

Calculates comprehensive metrics for survey questions, returning a summary analysis in tidy format. This includes mean, standard deviation, Glint Score, response counts, skip counts, and favorability percentages.

Usage

```
summarize_survey(
  survey,
  scale_points,
  questions = "all",
  emp_id_col = NULL,
  plot = FALSE
)
```

Arguments

`survey`	A glint_survey object or data frame containing survey data
`scale_points`	Integer specifying the number of scale points (2-11)
`questions`	Character vector of question text(s) to analyze, or "all" to analyze all questions (default: "all")
`emp_id_col`	Character string specifying the employee ID column name
`plot`	Logical. If `TRUE`, prints a favorability stacked bar chart sorted by Glint Score and returns the data invisibly. Requires **ggplot2**. Default: `FALSE`.

Value

A tibble with one row per question containing:

question The question text

mean Mean of numeric responses (raw scale)

sd Standard deviation of numeric responses

glint_score Mean transformed to a 0-100 scale, matching the score displayed in the Viva Glint UI: `round(((mean - 1) / (scale_points - 1)) * 100)`

n_responses Count of non-blank, non-null responses

n_skips Count of blank or null responses

n_total Total number of respondents

pct_favorable Percentage of responses classified as favorable

pct_neutral Percentage of responses classified as neutral

pct_unfavorable Percentage of responses classified as unfavorable

When `plot = TRUE`, the same tibble is returned invisibly after printing the plot.

Examples

```
survey_path <- system.file("extdata", "survey_export.csv", package = "vivaglint")
survey <- read_glint_survey(survey_path, emp_id_col = "EMP ID")

# Summarize all questions (5-point scale)
summary <- summarize_survey(survey, scale_points = 5)

# Summarize specific questions
summary_subset <- summarize_survey(survey, scale_points = 5,
  questions = c("My work is meaningful", "I feel valued"))

# With favorability chart
if (requireNamespace("ggplot2", quietly = TRUE)) {
  summarize_survey(survey, scale_points = 5, plot = TRUE)
}
```

Index