Package 'vivainsights'

May 8, 2026

Type Package

Title Analyze and Visualize Data from 'Microsoft Viva Insights'

Version 0.7.2

Maintainer Martin Chan <martin.chan@microsoft.com>

Description
Provides a versatile range of functions, including exploratory data analysis, time-series analysis, organizational network analysis, and data validation, whilst at the same time implements a set of best practices in analyzing and visualizing data specific to 'Microsoft Viva Insights'.

RoxygenNote 7.3.3

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Language en-US

Depends R (>= 4.1.0)

Imports dplyr, stats, utils, tidyr, tibble, tidyselect (>= 1.0.0), magrittr, ggplot2, reshape2, scales, ggrepel, purrr, data.table, methods, htmltools, markdown, networkD3, rmarkdown, wpa, ggraph, lifecycle, glue, igraph, rlang, tidytext

Suggests flexdashboard, testthat (>= 3.0.0), lmtest, sandwich, slider, ggwordcloud

URL https://microsoft.github.io/vivainsights/

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

Config/testthat/edition 3

NeedsCompilation no

Author Martin Chan [aut, cre], Carlos Morales [aut]

Repository CRAN

Date/Publication 2026-04-28 16:50:02 UTC

Contents

`afterhours_dist` *Distribution of After-hours Collaboration Hours as a 100% stacked bar*

Description

Analyse the distribution of weekly after-hours collaboration time. Returns a stacked bar plot by default. Additional options available to return a table with distribution elements.

Usage

```
afterhours_dist(
  data,
  hrvar = "Organization",
  mingroup = 5,
  return = "plot",
  cut = c(1, 2, 3)
)
```

Arguments

`data`	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns `PersonId` and `MetricDate` present. If your data is already aggregated (e.g. one row per group), use the equivalent `*_asis()` variant of this function instead.
`hrvar`	String containing the name of the HR Variable by which to split metrics. Defaults to `"Organization"`. To run the analysis on the total instead of splitting by an HR attribute, supply `NULL` (without quotes).
`mingroup`	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
`return`	String specifying what to return. This must be one of the following strings: • `"plot"` • `"table"` See `Value` for more information.
`cut`	A vector specifying the cuts to use for the data, accepting "default" or "range-cut" as character vector, or a numeric value of length three to specify the exact breaks to use. e.g. c(1, 3, 5)

Details

Uses the metric `After_hours_collaboration_hours`. See `create_dist()` for applying the same analysis to a different metric.

Value

A different output is returned depending on the value passed to the `return` argument:

- `"plot"`: 'ggplot' object. A stacked bar plot for the metric.
- `"table"`: data frame. A summary table for the metric.

See Also

Other Visualization: `afterhours_fizz()`, `afterhours_line()`, `afterhours_rank()`, `afterhours_summary()`, `afterhours_trend()`, `collaboration_area()`, `collaboration_dist()`, `collaboration_fizz()`, `collaboration_line()`, `collaboration_rank()`, `collaboration_sum()`, `collaboration_trend()`, `create_bar()`, `create_bar_asis()`, `create_boxplot()`, `create_bubble()`, `create_dist()`, `create_fizz()`, `create_inc()`, `create_line()`, `create_line_asis()`, `create_period_scatter()`, `create_radar()`, `create_rank()`, `create_rogers()`, `create_sankey()`, `create_scatter()`, `create_stacked()`, `create_survival()`, `create_tracking()`, `create_trend()`, `email_dist()`, `email_fizz()`, `email_line()`, `email_rank()`, `email_summary()`, `email_trend()`, `external_dist()`, `external_fizz()`, `external_line()`, `external_rank()`, `external_sum()`, `hr_trend()`, `hrvar_count()`, `hrvar_trend()`, `keymetrics_scan()`, `meeting_dist()`, `meeting_fizz()`, `meeting_line()`, `meeting_rank()`, `meeting_summary()`, `meeting_trend()`, `one2one_dist()`, `one2one_fizz()`, `one2one_freq()`, `one2one_line()`, `one2one_rank()`, `one2one_sum()`, `one2one_trend()`

Other After-hours Collaboration: `afterhours_fizz()`, `afterhours_line()`, `afterhours_rank()`, `afterhours_summary()`, `afterhours_trend()`, `external_rank()`

Examples

```
# Return plot
afterhours_dist(pq_data, hrvar = "Organization")

# Return summary table
afterhours_dist(pq_data, hrvar = "Organization", return = "table")

# Return result with a custom specified breaks
afterhours_dist(pq_data, hrvar = "LevelDesignation", cut = c(4, 7, 9))
```

`afterhours_fizz` *Distribution of After-hours Collaboration Hours (Fizzy Drink plot)*

Description

Analyze weekly after-hours collaboration hours distribution, and returns a 'fizzy' scatter plot by default. Additional options available to return a table with distribution elements.

Usage

```
afterhours_fizz(data, hrvar = "Organization", mingroup = 5, return = "plot")
```

Arguments

`data`	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns `PersonId` and `MetricDate` present. If your data is already aggregated (e.g. one row per group), use the equivalent `*_asis()` variant of this function instead.
`hrvar`	String containing the name of the HR Variable by which to split metrics. Defaults to `"Organization"`. To run the analysis on the total instead of splitting by an HR attribute, supply `NULL` (without quotes).
`mingroup`	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
`return`	String specifying what to return. This must be one of the following strings: • `"plot"` • `"table"` See `Value` for more information.

Details

Uses the metric `After_hours_collaboration_hours`. See `create_fizz()` for applying the same analysis to a different metric.

Value

A different output is returned depending on the value passed to the `return` argument:

- `"plot"`: 'ggplot' object. A jittered scatter plot for the metric.
- `"table"`: data frame. A summary table for the metric.

See Also

Other Visualization: `afterhours_dist()`, `afterhours_line()`, `afterhours_rank()`, `afterhours_summary()`, `afterhours_trend()`, `collaboration_area()`, `collaboration_dist()`, `collaboration_fizz()`, `collaboration_line()`, `collaboration_rank()`, `collaboration_sum()`, `collaboration_trend()`, `create_bar()`, `create_bar_asis()`, `create_boxplot()`, `create_bubble()`, `create_dist()`, `create_fizz()`, `create_inc()`, `create_line()`, `create_line_asis()`, `create_period_scatter()`, `create_radar()`, `create_rank()`, `create_rogers()`, `create_sankey()`, `create_scatter()`, `create_stacked()`, `create_survival()`, `create_tracking()`, `create_trend()`, `email_dist()`, `email_fizz()`, `email_line()`, `email_rank()`, `email_summary()`, `email_trend()`, `external_dist()`, `external_fizz()`, `external_line()`, `external_rank()`, `external_sum()`, `hr_trend()`, `hrvar_count()`, `hrvar_trend()`, `keymetrics_scan()`, `meeting_dist()`, `meeting_fizz()`, `meeting_line()`, `meeting_rank()`, `meeting_summary()`, `meeting_trend()`, `one2one_dist()`, `one2one_fizz()`, `one2one_freq()`, `one2one_line()`, `one2one_rank()`, `one2one_sum()`, `one2one_trend()`

Other After-hours Collaboration: `afterhours_dist()`, `afterhours_line()`, `afterhours_rank()`, `afterhours_summary()`, `afterhours_trend()`, `external_rank()`

Examples

```
# Return plot
afterhours_fizz(pq_data, hrvar = "LevelDesignation", return = "plot")

# Return summary table
afterhours_fizz(pq_data, hrvar = "Organization", return = "table")
```

`afterhours_line` *After-hours Collaboration Time Trend - Line Chart*

Description

Provides a week by week view of after-hours collaboration time, visualized as line charts. By default returns a line chart for after-hours collaboration hours, with a separate panel per value in the HR attribute. Additional options available to return a summary table.

Usage

```
afterhours_line(
  data,
  hrvar = "Organization",
  mingroup = 5,
  return = "plot",
  label = FALSE
)
```

Arguments

`data`	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns `PersonId` and `MetricDate` present. If your data is already aggregated (e.g. one row per group), use the equivalent `*_asis()` variant of this function instead.
`hrvar`	String containing the name of the HR Variable by which to split metrics. Defaults to `"Organization"`. To run the analysis on the total instead of splitting by an HR attribute, supply `NULL` (without quotes).
`mingroup`	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
`return`	String specifying what to return. This must be one of the following strings: • `"plot"` • `"table"` See `Value` for more information.
`label`	Logical value to determine whether to show data point labels on the plot. If `TRUE`, both `geom_point()` and `geom_text()` are added to display data labels rounded to 1 decimal place above each data point. Defaults to `FALSE`.

Details

Uses the metric `After_hours_collaboration_hours`.

Value

A different output is returned depending on the value passed to the `return` argument:

- `"plot"`: 'ggplot' object. A faceted line plot for the metric.
- `"table"`: data frame. A summary table for the metric.

See Also

`create_line()` for applying the same analysis to a different metric.

Other Visualization: `afterhours_dist()`, `afterhours_fizz()`, `afterhours_rank()`, `afterhours_summary()`, `afterhours_trend()`, `collaboration_area()`, `collaboration_dist()`, `collaboration_fizz()`, `collaboration_line()`, `collaboration_rank()`, `collaboration_sum()`, `collaboration_trend()`, `create_bar()`, `create_bar_asis()`, `create_boxplot()`, `create_bubble()`, `create_dist()`, `create_fizz()`, `create_inc()`, `create_line()`, `create_line_asis()`, `create_period_scatter()`, `create_radar()`, `create_rank()`, `create_rogers()`, `create_sankey()`, `create_scatter()`, `create_stacked()`, `create_survival()`, `create_tracking()`, `create_trend()`, `email_dist()`, `email_fizz()`, `email_line()`, `email_rank()`, `email_summary()`, `email_trend()`, `external_dist()`, `external_fizz()`, `external_line()`, `external_rank()`, `external_sum()`, `hr_trend()`, `hrvar_count()`, `hrvar_trend()`, `keymetrics_scan()`, `meeting_dist()`, `meeting_fizz()`, `meeting_line()`, `meeting_rank()`, `meeting_summary()`, `meeting_trend()`, `one2one_dist()`, `one2one_fizz()`, `one2one_freq()`, `one2one_line()`, `one2one_rank()`, `one2one_sum()`, `one2one_trend()`

Other After-hours Collaboration: `afterhours_dist()`, `afterhours_fizz()`, `afterhours_rank()`, `afterhours_summary()`, `afterhours_trend()`, `external_rank()`

Examples

```
# Return a line plot
afterhours_line(pq_data, hrvar = "LevelDesignation")

# Return summary table
afterhours_line(pq_data, hrvar = "LevelDesignation", return = "table")
```

`afterhours_rank` *Rank groups with high After-Hours Collaboration Hours*

Description

This function scans a Standard Person Query for groups with high levels of After-Hours Collaboration. Returns a plot by default, with an option to return a table with all groups (across multiple HR attributes) ranked by hours of After-Hours Collaboration Hours.

Usage

```
afterhours_rank(
  data,
  hrvar = extract_hr(data),
  mingroup = 5,
  mode = "simple",
  plot_mode = 1,
  return = "plot"
)
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present. If your data is already aggregated (e.g. one row per group), use the equivalent *_asis() variant of this function instead.
hrvar	String containing the name of the HR Variable by which to split metrics. Defaults to "Organization". To run the analysis on the total instead of splitting by an HR attribute, supply NULL (without quotes).
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
mode	String to specify calculation mode. Must be either: • "simple" • "combine"
plot_mode	Numeric vector to determine which plot mode to return. Must be either 1 or 2, and is only used when return = "plot". • 1: Top and bottom five groups across the data population are highlighted • 2: Top and bottom groups *per* organizational attribute are highlighted
return	String specifying what to return. This must be one of the following strings: • "plot" (default) • "table" See Value for more information.

Details

Uses the metric After_hours_collaboration_hours. See create_rank() for applying the same analysis to a different metric.

Value

When 'table' is passed in return, a summary table is returned as a data frame.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other After-hours Collaboration: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_summary(), afterhours_trend(), external_rank()

Examples

```
# Return plot
afterhours_rank(pq_data, return = "plot")

# Return summary table
afterhours_rank(pq_data, return = "table")
```

---

afterhours_summary *Summary of After-Hours Collaboration Hours*

Description

Provides an overview analysis of after-hours collaboration time. Returns a bar plot showing average weekly after-hours collaboration hours by default. Additional options available to return a summary table.

Usage

```
afterhours_summary(data, hrvar = "Organization", mingroup = 5, return = "plot")

afterhours_sum(data, hrvar = "Organization", mingroup = 5, return = "plot")
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present. If your data is already aggregated (e.g. one row per group), use the equivalent *_asis() variant of this function instead.

hrvar	String containing the name of the HR Variable by which to split metrics. Defaults to "Organization". To run the analysis on the total instead of splitting by an HR attribute, supply NULL (without quotes).
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
return	String specifying what to return. This must be one of the following strings: • "plot" • "table" See Value for more information.

Details

Uses the metric After_hours_collaboration_hours.

Value

A different output is returned depending on the value passed to the return argument:

- "plot": 'ggplot' object. A bar plot for the metric.
- "table": data frame. A summary table for the metric.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other After-hours Collaboration: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_trend(), external_rank()

Examples

```
# Return a ggplot bar chart
afterhours_summary(pq_data, hrvar = "LevelDesignation")

# Return a summary table
afterhours_summary(pq_data, hrvar = "LevelDesignation", return = "table")
```

`afterhours_trend` *After-Hours Time Trend*

Description

Provides a week by week view of after-hours collaboration time. By default returns a week by week heatmap, highlighting the points in time with most activity. Additional options available to return a summary table.

Usage

```
afterhours_trend(data, hrvar = "Organization", mingroup = 5, return = "plot")
```

Arguments

`data`	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns `PersonId` and `MetricDate` present. If your data is already aggregated (e.g. one row per group), use the equivalent `*_asis()` variant of this function instead.
`hrvar`	String containing the name of the HR Variable by which to split metrics. Defaults to `"Organization"`. To run the analysis on the total instead of splitting by an HR attribute, supply `NULL` (without quotes).
`mingroup`	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
`return`	Character vector specifying what to return, defaults to `"plot"`. Valid inputs are "plot" and "table".

Details

Uses the metric `After_hours_collaboration_hours`.

Value

Returns a 'ggplot' object by default, where 'plot' is passed in `return`. When 'table' is passed, a summary table is returned as a data frame.

See Also

Other Visualization: `afterhours_dist()`, `afterhours_fizz()`, `afterhours_line()`, `afterhours_rank()`, `afterhours_summary()`, `collaboration_area()`, `collaboration_dist()`, `collaboration_fizz()`, `collaboration_line()`, `collaboration_rank()`, `collaboration_sum()`, `collaboration_trend()`, `create_bar()`, `create_bar_asis()`, `create_boxplot()`, `create_bubble()`, `create_dist()`, `create_fizz()`, `create_inc()`, `create_line()`, `create_line_asis()`, `create_period_scatter()`, `create_radar()`, `create_rank()`, `create_rogers()`, `create_sankey()`, `create_scatter()`, `create_stacked()`, `create_survival()`, `create_tracking()`, `create_trend()`, `email_dist()`, `email_fizz()`, `email_line()`, `email_rank()`, `email_summary()`, `email_trend()`, `external_dist()`,

external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other After-hours Collaboration: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), external_rank()

Examples

```
# Run plot
afterhours_trend(pq_data)

# Run table
afterhours_trend(pq_data, hrvar = "LevelDesignation", return = "table")
```

---

anonymise *Anonymise a categorical variable by replacing values*

Description

Anonymize categorical variables such as HR variables by replacing values with dummy team names such as 'Team A'. The behaviour is to make 1 to 1 replacements by default, but there is an option to completely randomise values in the categorical variable.

Usage

```
anonymise(x, scramble = FALSE, replacement = NULL)

anonymize(x, scramble = FALSE, replacement = NULL)
```

Arguments

x	Character vector to be passed through.
scramble	Logical value determining whether to randomise values in the categorical variable.
replacement	Character vector containing the values to replace original values in the categorical variable. The length of the vector must be at least as great as the number of unique values in the original variable. Defaults to NULL, where the replacement would consist of "Team A", "Team B", etc.

Value

Character vector with the same length as input x, replaced with values provided in replacement.

See Also

jitter

Examples

```
unique(anonymise(pq_data$Organization))

rep <- c("Manager+", "Manager", "IC")
unique(anonymise(pq_data$Layer), replacement = rep)
```

---

any_idate *Identify whether variable is an IDate class.*

Description

This function checks whether the variable is an IDate class.

Usage

```
any_idate(x)
```

Arguments

x	Variable to test whether an IDate class.

Value

logical value indicating whether the string is of an IDate class.

See Also

Other Support: camel_clean(), check_inputs(), cut_hour(), extract_date_range(), extract_hr(), heat_colours(), is_date_format(), maxmin(), pairwise_count(), read_preamble(), rgb2hex(), totals_bind(), totals_col(), tstamp(), us_to_space(), wrap()

Examples

```
any_idate("2023-12-15")
```

camel_clean *Convert "CamelCase" to "Camel Case"*

Description

Convert a text string from the format "CamelCase" to "Camel Case". This is used for converting variable names such as "LevelDesignation" to "Level Designation" for the purpose of prettifying plot labels.

Usage

```
camel_clean(string)
```

Arguments

string	A string vector in 'CamelCase' format to format

Value

Returns a formatted string.

See Also

Other Support: any_idate(), check_inputs(), cut_hour(), extract_date_range(), extract_hr(), heat_colours(), is_date_format(), maxmin(), pairwise_count(), read_preamble(), rgb2hex(), totals_bind(), totals_col(), tstamp(), us_to_space(), wrap()

Examples

```
camel_clean("NoteHowTheStringIsFormatted")
```

check_inputs *Check whether a data frame contains all the required variable*

Description

Checks whether a data frame contains all the required variables. Matching works via variable names, and used to support individual functions in the package. Not used directly.

Usage

```
check_inputs(input, requirements, return = "stop")
```

Arguments

input	Pass a data frame for checking
requirements	A character vector specifying the required variable names
return	A character string specifying what to return. The default value is "stop". Also accepts "names" and "warning".

Value

The default behaviour is to return an error message, informing the user what variables are not included. When return is set to "names", a character vector containing the unmatched variable names is returned.

See Also

Other Support: any_idate(), camel_clean(), cut_hour(), extract_date_range(), extract_hr(), heat_colours(), is_date_format(), maxmin(), pairwise_count(), read_preamble(), rgb2hex(), totals_bind(), totals_col(), tstamp(), us_to_space(), wrap()

Examples

```
# Return error message
## Not run:
check_inputs(iris, c("Sepal.Length", "mpg"))

## End(Not run)

#' # Return warning message
check_inputs(iris, c("Sepal.Length", "mpg"), return = "warning")

# Return variable names
check_inputs(iris, c("Sepal.Length", "Sepal.Width", "RandomVariable"), return = "names")
```

---

check_query *Check a query to ensure that it is suitable for analysis*

Description

Prints diagnostic data about the data query to the R console, with information such as date range, number of employees, HR attributes identified, etc.

Usage

```
check_query(data, return = "message", validation = FALSE)
```

Arguments

`data`	A person-level query in the form of a data frame. This includes: • Standard Person Query • Ways of Working Assessment Query • Hourly Collaboration Query All person-level query have a `PersonId` column and a `MetricDate` column.
`return`	String specifying what to return. This must be one of the following strings: • `"message"` (default) • `"text"` See `Value` for more information.
`validation`	Logical value to specify whether to show summarized version. Defaults to `FALSE`. To hide checks on variable names, set `validation` to `TRUE`.

Details

This can be used with any person-level query, such as the standard person query, Ways of Working assessment query, and the hourly collaboration query. When run, this prints diagnostic data to the R console.

Value

A different output is returned depending on the value passed to the `return` argument:

- `"message"`: a message is returned to the console.
- `"text"`: string containing the diagnostic message.

See Also

Other Data Validation: `extract_hr()`, `flag_ch_ratio()`, `flag_em_ratio()`, `flag_extreme()`, `flag_outlooktime()`, `hr_trend()`, `hrvar_count()`, `hrvar_count_all()`, `hrvar_trend()`, `identify_churn()`, `identify_holidayweeks()`, `identify_inactiveweeks()`, `identify_nkw()`, `identify_outlier()`, `identify_privacythreshold()`, `identify_shifts()`, `identify_tenure()`, `track_HR_change()`, `validation_report()`

Examples

```
check_query(pq_data)
```

collaboration_area *Collaboration - Stacked Area Plot*

Description

Provides an overview analysis of Weekly Digital Collaboration. Returns an stacked area plot of Email and Meeting Hours by default. Additional options available to return a summary table.

Usage

```
collaboration_area(data, hrvar = NULL, mingroup = 5, return = "plot")

collab_area(data, hrvar = NULL, mingroup = 5, return = "plot")
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present. A Ways of Working assessment dataset may also be provided, in which Unscheduled call hours would be included in the output.
hrvar	HR Variable by which to split metrics, defaults to NULL, but accepts any character vector, e.g. "LevelDesignation". If NULL is passed, the organizational attribute is automatically populated as "Total".
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
return	String specifying what to return. This must be one of the following strings: • "plot" • "table" See Value for more information.

Details

Uses the metrics Meeting_hours, Email_hours, Unscheduled_Call_hours, and Instant_Message_hours.

Value

A different output is returned depending on the value passed to the return argument:

- "plot": 'ggplot' object. A stacked area plot for the metric.
- "table": data frame. A summary table for the metric.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Collaboration: collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend()

Examples

```
## Not run:
# Return plot with total (default)
collaboration_area(pq_data)

# Return plot with hrvar split
collaboration_area(pq_data, hrvar = "Organization")

# Return summary table
collaboration_area(pq_data, return = "table")

## End(Not run)
```

---

collaboration_dist *Distribution of Collaboration Hours as a 100% stacked bar*

Description

Analyze the distribution of Collaboration Hours. Returns a stacked bar plot by default. Additional options available to return a table with distribution elements.

Usage

```
collaboration_dist(
  data,
  hrvar = "Organization",
  mingroup = 5,
  return = "plot",
  cut = c(15, 20, 25)
```

```
)

collab_dist(
  data,
  hrvar = "Organization",
  mingroup = 5,
  return = "plot",
  cut = c(15, 20, 25)
)
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present. If your data is already aggregated (e.g. one row per group), use the equivalent *_asis() variant of this function instead.
hrvar	String containing the name of the HR Variable by which to split metrics. Defaults to "Organization". To run the analysis on the total instead of splitting by an HR attribute, supply NULL (without quotes).
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
return	String specifying what to return. This must be one of the following strings: • "plot" • "table" See Value for more information.
cut	A numeric vector of length three to specify the breaks for the distribution, e.g. c(10, 15, 20)

Value

A different output is returned depending on the value passed to the return argument:

- "plot": 'ggplot' object. A stacked bar plot for the metric.
- "table": data frame. A summary table for the metric.

Metrics used

The metric Collaboration_hours is used in the calculations. Please ensure that your query contains a metric with the exact same name.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(),

create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Collaboration: collaboration_area(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend()

Examples

```
# Return plot
collaboration_dist(pq_data, hrvar = "Organization")

# Return summary table
collaboration_dist(pq_data, hrvar = "Organization", return = "table")
```

---

collaboration_fizz *Distribution of Collaboration Hours (Fizzy Drink plot)*

Description

Analyze weekly collaboration hours distribution, and returns a 'fizzy' scatter plot by default. Additional options available to return a table with distribution elements.

Usage

```
collaboration_fizz(data, hrvar = "Organization", mingroup = 5, return = "plot")

collab_fizz(data, hrvar = "Organization", mingroup = 5, return = "plot")
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present. If your data is already aggregated (e.g. one row per group), use the equivalent *_asis() variant of this function instead.
hrvar	String containing the name of the HR Variable by which to split metrics. Defaults to "Organization". To run the analysis on the total instead of splitting by an HR attribute, supply NULL (without quotes).
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
return	String specifying what to return. This must be one of the following strings:

- "plot"
- "table"

See Value for more information.

Value

A different output is returned depending on the value passed to the return argument:

- "plot": 'ggplot' object. A jittered scatter plot for the metric.
- "table": data frame. A summary table for the metric.

Metrics used

The metric Collaboration_hours is used in the calculations. Please ensure that your query contains a metric with the exact same name.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Collaboration: collaboration_area(), collaboration_dist(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend()

Examples

```
# Return plot
collaboration_fizz(pq_data, hrvar = "Organization", return = "plot")

# Return summary table
collaboration_fizz(pq_data, hrvar = "Organization", return = "table")
```

`collaboration_line` *Collaboration Time Trend - Line Chart*

Description

Provides a week by week view of collaboration time, visualised as line charts. By default returns a line chart for collaboration hours, with a separate panel per value in the HR attribute. Additional options available to return a summary table.

Usage

```
collaboration_line(
  data,
  hrvar = "Organization",
  mingroup = 5,
  return = "plot",
  label = FALSE
)

collab_line(
  data,
  hrvar = "Organization",
  mingroup = 5,
  return = "plot",
  label = FALSE
)
```

Arguments

`data`	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns `PersonId` and `MetricDate` present. If your data is already aggregated (e.g. one row per group), use the equivalent `*_asis()` variant of this function instead.
`hrvar`	String containing the name of the HR Variable by which to split metrics. Defaults to `"Organization"`. To run the analysis on the total instead of splitting by an HR attribute, supply `NULL` (without quotes).
`mingroup`	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
`return`	String specifying what to return. This must be one of the following strings: • `"plot"` • `"table"` See `Value` for more information.
`label`	Logical value to determine whether to show data point labels on the plot. If `TRUE`, both `geom_point()` and `geom_text()` are added to display data labels rounded to 1 decimal place above each data point. Defaults to `FALSE`.

Value

A different output is returned depending on the value passed to the `return` argument:

- `"plot"`: 'ggplot' object. A faceted line plot for the metric.
- `"table"`: data frame. A summary table for the metric.

Metrics used

The metric `Collaboration_hours` is used in the calculations. Please ensure that your query contains a metric with the exact same name.

See Also

Other Visualization: `afterhours_dist()`, `afterhours_fizz()`, `afterhours_line()`, `afterhours_rank()`, `afterhours_summary()`, `afterhours_trend()`, `collaboration_area()`, `collaboration_dist()`, `collaboration_fizz()`, `collaboration_rank()`, `collaboration_sum()`, `collaboration_trend()`, `create_bar()`, `create_bar_asis()`, `create_boxplot()`, `create_bubble()`, `create_dist()`, `create_fizz()`, `create_inc()`, `create_line()`, `create_line_asis()`, `create_period_scatter()`, `create_radar()`, `create_rank()`, `create_rogers()`, `create_sankey()`, `create_scatter()`, `create_stacked()`, `create_survival()`, `create_tracking()`, `create_trend()`, `email_dist()`, `email_fizz()`, `email_line()`, `email_rank()`, `email_summary()`, `email_trend()`, `external_dist()`, `external_fizz()`, `external_line()`, `external_rank()`, `external_sum()`, `hr_trend()`, `hrvar_count()`, `hrvar_trend()`, `keymetrics_scan()`, `meeting_dist()`, `meeting_fizz()`, `meeting_line()`, `meeting_rank()`, `meeting_summary()`, `meeting_trend()`, `one2one_dist()`, `one2one_fizz()`, `one2one_freq()`, `one2one_line()`, `one2one_rank()`, `one2one_sum()`, `one2one_trend()`

Other Collaboration: `collaboration_area()`, `collaboration_dist()`, `collaboration_fizz()`, `collaboration_rank()`, `collaboration_sum()`, `collaboration_trend()`

Examples

```
# Return a line plot
collaboration_line(pq_data, hrvar = "LevelDesignation")

# Return summary table
collaboration_line(pq_data, hrvar = "LevelDesignation", return = "table")
```

---

collaboration_rank *Collaboration Ranking*

Description

This function scans a standard query output for groups with high levels of 'Weekly Digital Collaboration'. Returns a plot by default, with an option to return a table with a all of groups (across multiple HR attributes) ranked by hours of digital collaboration.

Usage

```
collaboration_rank(
  data,
  hrvar = extract_hr(data),
  mingroup = 5,
  mode = "simple",
  plot_mode = 1,
  return = "plot"
)

collab_rank(
  data,
  hrvar = extract_hr(data),
  mingroup = 5,
  mode = "simple",
  plot_mode = 1,
  return = "plot"
)
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present. If your data is already aggregated (e.g. one row per group), use the equivalent *_asis() variant of this function instead.
hrvar	String containing the name of the HR Variable by which to split metrics. Defaults to "Organization". To run the analysis on the total instead of splitting by an HR attribute, supply NULL (without quotes).
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
mode	String to specify calculation mode. Must be either: • "simple" • "combine"
plot_mode	Numeric vector to determine which plot mode to return. Must be either 1 or 2, and is only used when return = "plot". • 1: Top and bottom five groups across the data population are highlighted • 2: Top and bottom groups *per* organizational attribute are highlighted
return	String specifying what to return. This must be one of the following strings: • "plot" (default) • "table" See Value for more information.

Details

Uses the metric Collaboration_hours. See create_rank() for applying the same analysis to a different metric.

Value

A different output is returned depending on the value passed to the `return` argument:

- `"plot"`: 'ggplot' object. A bubble plot where the x-axis represents the metric, the y-axis represents the HR attributes, and the size of the bubbles represent the size of the organizations. Note that there is no plot output if `mode` is set to `"combine"`.
- `"table"`: data frame. A summary table for the metric.

See Also

Other Visualization: `afterhours_dist()`, `afterhours_fizz()`, `afterhours_line()`, `afterhours_rank()`, `afterhours_summary()`, `afterhours_trend()`, `collaboration_area()`, `collaboration_dist()`, `collaboration_fizz()`, `collaboration_line()`, `collaboration_sum()`, `collaboration_trend()`, `create_bar()`, `create_bar_asis()`, `create_boxplot()`, `create_bubble()`, `create_dist()`, `create_fizz()`, `create_inc()`, `create_line()`, `create_line_asis()`, `create_period_scatter()`, `create_radar()`, `create_rank()`, `create_rogers()`, `create_sankey()`, `create_scatter()`, `create_stacked()`, `create_survival()`, `create_tracking()`, `create_trend()`, `email_dist()`, `email_fizz()`, `email_line()`, `email_rank()`, `email_summary()`, `email_trend()`, `external_dist()`, `external_fizz()`, `external_line()`, `external_rank()`, `external_sum()`, `hr_trend()`, `hrvar_count()`, `hrvar_trend()`, `keymetrics_scan()`, `meeting_dist()`, `meeting_fizz()`, `meeting_line()`, `meeting_rank()`, `meeting_summary()`, `meeting_trend()`, `one2one_dist()`, `one2one_fizz()`, `one2one_freq()`, `one2one_line()`, `one2one_rank()`, `one2one_sum()`, `one2one_trend()`

Other Collaboration: `collaboration_area()`, `collaboration_dist()`, `collaboration_fizz()`, `collaboration_line()`, `collaboration_sum()`, `collaboration_trend()`

Examples

```
# Return rank table
collaboration_rank(
  data = pq_data,
  return = "table"
)

# Return plot
collaboration_rank(
  data = pq_data,
  return = "plot"
)
```

`collaboration_sum` *Collaboration Summary*

Description

Provides an overview analysis of 'Weekly Digital Collaboration'. Returns a stacked bar plot of Email and Meeting Hours by default. Additional options available to return a summary table.

Usage

```
collaboration_sum(data, hrvar = "Organization", mingroup = 5, return = "plot")

collab_sum(data, hrvar = "Organization", mingroup = 5, return = "plot")

collaboration_summary(
  data,
  hrvar = "Organization",
  mingroup = 5,
  return = "plot"
)

collab_summary(data, hrvar = "Organization", mingroup = 5, return = "plot")
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present. If your data is already aggregated (e.g. one row per group), use the equivalent *_asis() variant of this function instead.
hrvar	String containing the name of the HR Variable by which to split metrics. Defaults to "Organization". To run the analysis on the total instead of splitting by an HR attribute, supply NULL (without quotes).
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
return	Character vector specifying what to return, defaults to "plot". Valid inputs are "plot" and "table".

Details

Uses the metrics Meeting_hours, Email_hours, Unscheduled_Call_hours, and Instant_Message_hours.

Value

Returns a 'ggplot' object by default, where 'plot' is passed in return. When 'table' is passed, a summary table is returned as a data frame.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(),

external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Collaboration: collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_trend()

Examples

```
# Return a ggplot bar chart
collaboration_sum(pq_data, hrvar = "LevelDesignation")

# Return a summary table
collaboration_sum(pq_data, hrvar = "LevelDesignation", return = "table")
```

---

collaboration_trend *Collaboration Time Trend*

Description

Provides a week by week view of collaboration time. By default returns a week by week heatmap, highlighting the points in time with most activity. Additional options available to return a summary table.

Usage

```
collaboration_trend(
  data,
  hrvar = "Organization",
  mingroup = 5,
  return = "plot"
)
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present. If your data is already aggregated (e.g. one row per group), use the equivalent *_asis() variant of this function instead.
hrvar	String containing the name of the HR Variable by which to split metrics. Defaults to "Organization". To run the analysis on the total instead of splitting by an HR attribute, supply NULL (without quotes).
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
return	Character vector specifying what to return, defaults to "plot". Valid inputs are "plot" and "table".

Value

Returns a 'ggplot' object by default, where 'plot' is passed in `return`. When 'table' is passed, a summary table is returned as a data frame.

Metrics used

The metric `Collaboration_hours` is used in the calculations. Please ensure that your query contains a metric with the exact same name.

See Also

Other Visualization: `afterhours_dist()`, `afterhours_fizz()`, `afterhours_line()`, `afterhours_rank()`, `afterhours_summary()`, `afterhours_trend()`, `collaboration_area()`, `collaboration_dist()`, `collaboration_fizz()`, `collaboration_line()`, `collaboration_rank()`, `collaboration_sum()`, `create_bar()`, `create_bar_asis()`, `create_boxplot()`, `create_bubble()`, `create_dist()`, `create_fizz()`, `create_inc()`, `create_line()`, `create_line_asis()`, `create_period_scatter()`, `create_radar()`, `create_rank()`, `create_rogers()`, `create_sankey()`, `create_scatter()`, `create_stacked()`, `create_survival()`, `create_tracking()`, `create_trend()`, `email_dist()`, `email_fizz()`, `email_line()`, `email_rank()`, `email_summary()`, `email_trend()`, `external_dist()`, `external_fizz()`, `external_line()`, `external_rank()`, `external_sum()`, `hr_trend()`, `hrvar_count()`, `hrvar_trend()`, `keymetrics_scan()`, `meeting_dist()`, `meeting_fizz()`, `meeting_line()`, `meeting_rank()`, `meeting_summary()`, `meeting_trend()`, `one2one_dist()`, `one2one_fizz()`, `one2one_freq()`, `one2one_line()`, `one2one_rank()`, `one2one_sum()`, `one2one_trend()`

Other Collaboration: `collaboration_area()`, `collaboration_dist()`, `collaboration_fizz()`, `collaboration_line()`, `collaboration_rank()`, `collaboration_sum()`

Examples

```
# Run plot
collaboration_trend(pq_data)

# Run table
collaboration_trend(pq_data, hrvar = "LevelDesignation", return = "table")
```

---

comma *Add comma separator for thousands*

Description

Takes a numeric value and returns a character value which is rounded to the whole number, and adds a comma separator at the thousands. A convenient wrapper function around `round()` and `format()`.

Usage

```
comma(x)
```

Arguments

x	A numeric value

Value

Returns a formatted string.

`copy_df` *Copy a data frame to clipboard for pasting in Excel*

Description

This is a pipe-optimised function, that feeds into `vivainsights::export()`, but can be used as a stand-alone function.

Based on the original function from https://github.com/martinctc/surveytoolbox.

Usage

```
copy_df(x, row.names = FALSE, col.names = TRUE, quietly = FALSE, ...)
```

Arguments

x	Data frame to be passed through. Cannot contain list-columns or nested data frames.
row.names	A logical vector for specifying whether to allow row names. Defaults to FALSE.
col.names	A logical vector for specifying whether to allow column names. Defaults to FALSE.
quietly	Set this to TRUE to not print data frame on console
...	Additional arguments for write.table().

Value

Copies a data frame to the clipboard with no return value.

See Also

Other Import and Export: `create_dt()`, `export()`, `import_query()`, `prep_query()`

create_bar *Mean Bar Plot for any metric*

Description

Provides an overview analysis of a selected metric by calculating a mean per metric. Returns a bar plot showing the average of a selected metric by default. Additional options available to return a summary table.

Usage

```
create_bar(
  data,
  metric,
  hrvar = "Organization",
  mingroup = 5,
  return = "plot",
  bar_colour = "default",
  na.rm = FALSE,
  percent = FALSE,
  plot_title = us_to_space(metric),
  plot_subtitle = paste("Average by", tolower(camel_clean(hrvar))),
  legend_lab = NULL,
  rank = "descending",
  xlim = NULL,
  text_just = 0.5,
  text_colour = "#FFFFFF"
)
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns `PersonId` and `MetricDate` present. If your data is already aggregated (e.g. one row per group), use the equivalent `*_asis()` variant of this function instead.
metric	Character string containing the name of the metric, e.g. "Collaboration_hours"
hrvar	String containing the name of the HR Variable by which to split metrics. Defaults to `"Organization"`. To run the analysis on the total instead of splitting by an HR attribute, supply `NULL` (without quotes).
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
return	String specifying what to return. This must be one of the following strings: • `"plot"` • `"table"`

	See Value for more information.
bar_colour	String to specify colour to use for bars. In-built accepted values include "default" (default), "alert" (red), and "darkblue". Otherwise, hex codes are also accepted. You can also supply RGB values via rgb2hex().
na.rm	A logical value indicating whether NA should be stripped before the computation proceeds. Defaults to FALSE.
percent	Logical value to determine whether to show labels as percentage signs. Defaults to FALSE.
plot_title	An option to override plot title.
plot_subtitle	An option to override plot subtitle.
legend_lab	String. Option to override legend title/label. Defaults to NULL, where the metric name will be populated instead.
rank	String specifying how to rank the bars. Valid inputs are: • "descending" - ranked highest to lowest from top to bottom (default). • "ascending" - ranked lowest to highest from top to bottom. • NULL - uses the original levels of the HR attribute.
xlim	An option to set max value in x axis.
text_just	**[Experimental]** A numeric value controlling for the horizontal position of the text labels. Defaults to 0.5.
text_colour	**[Experimental]** String to specify colour to use for the text labels. Defaults to "#FFFFFF".

Value

A different output is returned depending on the value passed to the return argument:

- "plot": 'ggplot' object. A bar plot for the metric.
- "table": data frame. A summary table for the metric.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Flexible: create_bar_asis(), create_boxplot(), create_bubble(), create_density(), create_dist(), create_fizz(), create_hist(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend()

Examples

```
# Return a ggplot bar chart
create_bar(pq_data, metric = "Collaboration_hours", hrvar = "LevelDesignation")

# Change bar colour
create_bar(pq_data,
           metric = "After_hours_collaboration_hours",
           bar_colour = "alert")

# Custom data label positions and formatting
pq_data %>%
  create_bar(
    metric = "Meetings",
    text_just = 1.1,
    text_colour = "black",
    xlim = 20)

# Return a summary table
create_bar(pq_data,
           metric = "Collaboration_hours",
           hrvar = "LevelDesignation",
           return = "table")
```

---

create_bar_asis *Create a bar chart without aggregation for any metric*

Description

This function creates a bar chart directly from the aggregated / summarised data. Unlike `create_bar()` which performs a person-level aggregation, there is no calculation for `create_bar_asis()` and the values are rendered as they are passed into the function.

Usage

```
create_bar_asis(
  data,
  group_var,
  bar_var,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  ylab = group_var,
  xlab = bar_var,
  percent = FALSE,
  bar_colour = "default",
  rounding = 1
)
```

Arguments

data	Aggregated or summarised data as a data frame. Unlike create_bar(), this function does **not** require panel data and can accept any pre-aggregated data frame (i.e. PersonId and MetricDate are not required).
group_var	String containing name of variable for the group.
bar_var	String containing name of variable representing the value of the bars.
title	Title of the plot.
subtitle	Subtitle of the plot.
caption	Caption of the plot.
ylab	Y-axis label for the plot (group axis)
xlab	X-axis label of the plot (bar axis).
percent	Logical value to determine whether to show labels as percentage signs. Defaults to FALSE.
bar_colour	String to specify colour to use for bars. In-built accepted values include "default" (default), "alert" (red), and "darkblue". Otherwise, hex codes are also accepted. You can also supply RGB values via rgb2hex().
rounding	Numeric value to specify number of digits to show in data labels

Value

'ggplot' object. A horizontal bar plot.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Flexible: create_bar(), create_boxplot(), create_bubble(), create_density(), create_dist(), create_fizz(), create_hist(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend()

Examples

```
# Creating a custom bar plot without mean aggregation
library(dplyr)
```

```
pq_data %>%
  group_by(Organization) %>%
  summarise(across(.cols = Meeting_hours,
                   .fns = ~sum(., na.rm = TRUE))) %>%
  create_bar_asis(group_var = "Organization",
                  bar_var = "Meeting_hours",
                  title = "Total Meeting Hours over period",
                  subtitle = "By Organization",
                  caption = extract_date_range(pq_data, return = "text"),
                  bar_colour = "darkblue",
                  rounding = 0)

library(dplyr)

# Summarise Non-person-average median `Emails_sent`
med_df <-
  pq_data %>%
  group_by(Organization) %>%
  summarise(Emails_sent_median = median(Emails_sent))

med_df %>%
  create_bar_asis(
    group_var = "Organization",
    bar_var = "Emails_sent_median",
    title = "Emails sent by organization",
    subtitle = "Median values",
    bar_colour = "darkblue",
    caption = extract_date_range(pq_data, return = "text")
  )
```

`create_boxplot` *Box Plot for any metric*

Description

Analyzes a selected metric and returns a box plot by default. Additional options available to return a table with distribution elements.

Usage

```
create_boxplot(
  data,
  metric,
  hrvar = "Organization",
  mingroup = 5,
  return = "plot"
)
```

Arguments

`data`	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns `PersonId` and `MetricDate` present. If your data is already aggregated (e.g. one row per group), use the equivalent `*_asis()` variant of this function instead.
`metric`	Character string containing the name of the metric, e.g. "Collaboration_hours"
`hrvar`	String containing the name of the HR Variable by which to split metrics. Defaults to `"Organization"`. To run the analysis on the total instead of splitting by an HR attribute, supply `NULL` (without quotes).
`mingroup`	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
`return`	String specifying what to return. This must be one of the following strings: • `"plot"` • `"table"` • `"data"` See `Value` for more information.

Details

This is a general purpose function that powers all the functions in the package that produce box plots.

Value

A different output is returned depending on the value passed to the `return` argument:

- `"plot"`: 'ggplot' object. A box plot for the metric.
- `"table"`: data frame. A summary table for the metric, containing the following columns:
  - `group`: The HR variable by which the metric is split.
  - `mean`: The mean of the metric.
  - `min`: The minimum value of the metric.
  - `p10`: The 10th percentile of the metric.
  - `p25`: The 25th percentile of the metric.
  - `p50`: The 50th percentile of the metric.
  - `p75`: The 75th percentile of the metric.
  - `p90`: The 90th percentile of the metric.
  - `max`: The maximum value of the metric.
  - `sd`: The standard deviation of the metric.
  - `range`: The range of the metric.
  - `n`: The number of observations.
- `"data"`: data frame. A data frame containing the metric and group.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Flexible: create_bar(), create_bar_asis(), create_bubble(), create_density(), create_dist(), create_fizz(), create_hist(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend()

Examples

```
# Create a box plot for Collaboration_hours by Level Designation
create_boxplot(pq_data, metric = "Collaboration_hours", hrvar = "LevelDesignation", return = "plot")

# Create a box plot for Collaboration_hours by Organization
create_boxplot(pq_data, metric = "Collaboration_hours", hrvar = "Organization", return = "plot")

# Create a summary statistics table for Collaboration_hoursby Organization
create_boxplot(pq_data, metric = "Collaboration_hours", hrvar = "Organization", return = "table")
```

---

create_bubble *Create a bubble plot with two selected Viva Insights metrics (General Purpose), with size representing the number of employees in the group.*

Description

Returns a bubble plot of two selected metrics, using size to map the number of employees.

Usage

```
create_bubble(
  data,
  metric_x,
  metric_y,
  hrvar = "Organization",
  mingroup = 5,
  return = "plot",
```

```
  bubble_size = c(1, 10)
)
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present.
metric_x	Character string containing the name of the metric, e.g. "Collaboration_hours"
metric_y	Character string containing the name of the metric, e.g. "Collaboration_hours"
hrvar	HR Variable by which to split metrics, defaults to "Organization" but accepts any character vector, e.g. "LevelDesignation"
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
return	String specifying what to return. This must be one of the following strings: - "plot" - "table"
bubble_size	A numeric vector of length two to specify the size range of the bubbles

Details

This is a general purpose function that powers all the functions in the package that produce bubble plots.

Value

A different output is returned depending on the value passed to the return argument:

- "plot": 'ggplot' object. A bubble plot for the metric.
- "table": data frame. A summary table for the metric.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Flexible: create_bar(), create_bar_asis(), create_boxplot(), create_density(), create_dist(), create_fizz(), create_hist(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend()

Examples

```
create_bubble(pq_data, "Collaboration_hours", "Multitasking_hours", hrvar ="Organization")
```

`create_density` *Create a density plot for any metric*

Description

Provides an analysis of the distribution of a selected metric. Returns a faceted density plot by default. Additional options available to return the underlying frequency table.

Usage

```
create_density(
  data,
  metric,
  hrvar = "Organization",
  mingroup = 5,
  ncol = NULL,
  return = "plot"
)
```

Arguments

`data`	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns `PersonId` and `MetricDate` present. If your data is already aggregated (e.g. one row per group), use the equivalent `*_asis()` variant of this function instead.
`metric`	String containing the name of the metric, e.g. "Collaboration_hours"
`hrvar`	String containing the name of the HR Variable by which to split metrics. Defaults to `"Organization"`. To run the analysis on the total instead of splitting by an HR attribute, supply `NULL` (without quotes).
`mingroup`	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
`ncol`	Numeric value setting the number of columns on the plot. Defaults to `NULL` (automatic).
`return`	String specifying what to return. This must be one of the following strings: • `"plot"` • `"table"` • `"data"` • `"frequency"` See `Value` for more information.

Value

A different output is returned depending on the value passed to the `return` argument:

- `"plot"`: 'ggplot' object. A faceted density plot for the metric.
- `"table"`: data frame. A summary table for the metric, containing the following columns:
  - `group`: The HR variable by which the metric is split.
  - `mean`: The mean of the metric.
  - `min`: The minimum value of the metric.
  - `p10`: The 10th percentile of the metric.
  - `p25`: The 25th percentile of the metric.
  - `p50`: The 50th percentile of the metric.
  - `p75`: The 75th percentile of the metric.
  - `p90`: The 90th percentile of the metric.
  - `max`: The maximum value of the metric.
  - `sd`: The standard deviation of the metric.
  - `range`: The range of the metric.
  - `n`: The number of observations.
- `"data"`: data frame. Data with calculated person averages.
- `"frequency`: list of data frames. Each data frame contains the frequencies used in each panel of the plotted histogram.

See Also

Other Flexible: `create_bar()`, `create_bar_asis()`, `create_boxplot()`, `create_bubble()`, `create_dist()`, `create_fizz()`, `create_hist()`, `create_inc()`, `create_line()`, `create_line_asis()`, `create_period_scatter()`, `create_radar()`, `create_rank()`, `create_sankey()`, `create_scatter()`, `create_stacked()`, `create_survival()`, `create_tracking()`, `create_trend()`

Examples

```
# Return plot for whole organization
create_density(pq_data, metric = "Collaboration_hours", hrvar = NULL)

# Return plot
create_density(pq_data, metric = "Collaboration_hours", hrvar = "Organization")

# Return plot but coerce plot to three columns
create_density(pq_data, metric = "Collaboration_hours", hrvar = "Organization", ncol = 3)

# Return summary table
create_density(pq_data, metric = "Collaboration_hours", hrvar = "Organization", return = "table")
```

create_dist *Horizontal 100 percent stacked bar plot for any metric*

Description

Provides an analysis of the distribution of a selected metric. Returns a stacked bar plot by default. Additional options available to return a table with distribution elements.

Usage

```
create_dist(
  data,
  metric,
  hrvar = "Organization",
  mingroup = 5,
  return = "plot",
  cut = c(15, 20, 25),
  dist_colours = c("#facebc", "#fcf0eb", "#b4d5dd", "#bfe5ee"),
  unit = "hours",
  lbound = 0,
  ubound = 200,
  sort_by = NULL,
  labels = NULL
)
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present. If your data is already aggregated (e.g. one row per group), use the equivalent *_asis() variant of this function instead.
metric	String containing the name of the metric, e.g. "Collaboration_hours"
hrvar	String containing the name of the HR Variable by which to split metrics. Defaults to "Organization". To run the analysis on the total instead of splitting by an HR attribute, supply NULL (without quotes).
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
return	String specifying what to return. This must be one of the following strings: • "plot" • "table" See Value for more information.
cut	A numeric vector of length three to specify the breaks for the distribution, e.g. c(10, 15, 20)

dist_colours	A character vector of length four to specify colour codes for the stacked bars.
unit	String to specify what unit to use. This defaults to "hours" but can accept any custom string. See cut_hour() for more details.
lbound	Numeric. Specifies the lower bound (inclusive) value for the minimum label. Defaults to 0.
ubound	Numeric. Specifies the upper bound (inclusive) value for the maximum label. Defaults to 100.
sort_by	String to specify the bucket label to sort by. Defaults to NULL (no sorting).
labels	Character vector to override labels for the created categorical variables. Must be a named vector - see examples.

Value

A different output is returned depending on the value passed to the return argument:

- "plot": 'ggplot' object. A stacked bar plot for the metric.
- "table": data frame. A summary table for the metric.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Flexible: create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_density(), create_fizz(), create_hist(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend()

Examples

```
# Return plot
create_dist(pq_data, metric = "Collaboration_hours", hrvar = "Organization")

# Return summary table
create_dist(pq_data, metric = "Collaboration_hours", hrvar = "Organization", return = "table")

# Use custom labels by providing a label vector
eh_labels <- c(
  "Fewer than fifteen" = "< 15 hours",
```

```
  "Between fifteen and twenty" = "15 - 20 hours",
  "Between twenty and twenty-five" = "20 - 25 hours",
  "More than twenty-five" = "25+ hours"
)

pq_data %>% create_dist(metric = "Meeting_hours", labels = eh_labels, return = "plot")

# Sort by a category
pq_data %>%  create_dist(metric = "Collaboration_hours", sort_by = "25+ hours")
```

`create_dt` *Create interactive tables in HTML with 'download' buttons.*

Description

See https://martinctc.github.io/blog/vignette-downloadable-tables-in-rmarkdown-with-the-dt-package/ for more.

Usage

```
create_dt(x, rounding = 1, freeze = 2, percent = FALSE)
```

Arguments

`x`	Data frame to be passed through.
`rounding`	Numeric vector to specify the number of decimal points to display
`freeze`	Number of columns from the left to 'freeze'. Defaults to 2, which includes the row number column.
`percent`	Logical value specifying whether to display numeric columns as percentages.

Details

This is exported from `wpa::create_dt()`.

Value

Returns an HTML widget displaying rectangular data.

See Also

Other Import and Export: `copy_df()`, `export()`, `import_query()`, `prep_query()`

Examples

```
output <- hrvar_count(pq_data, return = "table")
create_dt(output)
```

`create_fizz` *Fizzy Drink / Jittered Scatter Plot for any metric*

Description

Analyzes a selected metric and returns a 'fizzy' scatter plot by default. Additional options available to return a table with distribution elements.

Usage

```
create_fizz(
  data,
  metric,
  hrvar = "Organization",
  mingroup = 5,
  return = "plot"
)
```

Arguments

`data`	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns `PersonId` and `MetricDate` present. If your data is already aggregated (e.g. one row per group), use the equivalent `*_asis()` variant of this function instead.
`metric`	Character string containing the name of the metric, e.g. `"Collaboration_hours"`
`hrvar`	String containing the name of the HR Variable by which to split metrics. Defaults to `"Organization"`. To run the analysis on the total instead of splitting by an HR attribute, supply `NULL` (without quotes).
`mingroup`	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
`return`	String specifying what to return. This must be one of the following strings: • `"plot"` • `"table"` See `Value` for more information.

Details

This is a general purpose function that powers all the functions in the package that produce 'fizzy drink' / jittered scatter plots.

Value

A different output is returned depending on the value passed to the `return` argument:

- `"plot"`: 'ggplot' object. A jittered scatter plot for the metric.
- `"table"`: data frame. A summary table for the metric.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Flexible: create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_density(), create_dist(), create_hist(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend()

Examples

```
# Create a fizzy plot for Collaboration hours by Level Designation
create_fizz(pq_data, metric = "Collaboration_hours", hrvar = "LevelDesignation", return = "plot")

# Create a summary statistics table for Collaboration hours by Organization
create_fizz(pq_data, metric = "Collaboration_hours", hrvar = "Organization", return = "table")
```

---

create_hist *Create a histogram plot for any metric*

Description

Provides an analysis of the distribution of a selected metric. Returns a faceted histogram by default. Additional options available to return the underlying frequency table.

Usage

```
create_hist(
  data,
  metric,
  hrvar = "Organization",
  mingroup = 5,
  binwidth = 1,
  ncol = NULL,
  return = "plot"
)
```

Arguments

`data`	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns `PersonId` and `MetricDate` present. If your data is already aggregated (e.g. one row per group), use the equivalent `*_asis()` variant of this function instead.
`metric`	String containing the name of the metric, e.g. "Collaboration_hours"
`hrvar`	String containing the name of the HR Variable by which to split metrics. Defaults to `"Organization"`. To run the analysis on the total instead of splitting by an HR attribute, supply `NULL` (without quotes).
`mingroup`	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
`binwidth`	Numeric value for setting `binwidth` argument within `ggplot2::geom_histogram()`. Defaults to 1.
`ncol`	Numeric value setting the number of columns on the plot. Defaults to `NULL` (automatic).
`return`	String specifying what to return. This must be one of the following strings: • `"plot"` • `"table"` • `"data"` • `"frequency"` See `Value` for more information.

Value

A different output is returned depending on the value passed to the `return` argument:

- `"plot"`: 'ggplot' object. A faceted histogram for the metric.
- `"table"`: data frame. A summary table for the metric, containing the following columns:
  - `group`: The HR variable by which the metric is split.
  - `mean`: The mean of the metric.
  - `min`: The minimum value of the metric.
  - `p10`: The 10th percentile of the metric.
  - `p25`: The 25th percentile of the metric.
  - `p50`: The 50th percentile of the metric.
  - `p75`: The 75th percentile of the metric.
  - `p90`: The 90th percentile of the metric.
  - `max`: The maximum value of the metric.
  - `sd`: The standard deviation of the metric.
  - `range`: The range of the metric.
  - `n`: The number of observations.
- `"data"`: data frame. Data with calculated person averages.
- `"frequency`: list of data frames. Each data frame contains the frequencies used in each panel of the plotted histogram.

See Also

Other Flexible: create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_density(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend()

Examples

```
# Return plot for whole organization
create_hist(pq_data, metric = "Collaboration_hours", hrvar = NULL)

# Return plot
create_hist(pq_data, metric = "Collaboration_hours", hrvar = "Organization")

# Return plot but coerce plot to 3 columns
create_hist(pq_data, metric = "Collaboration_hours", hrvar = "Organization", ncol = 3)

# Return summary table
create_hist(pq_data,  metric = "Collaboration_hours", hrvar = "Organization", return = "table")
```

---

create_inc *Create an incidence analysis reflecting proportion of population scoring above or below a threshold for a metric*

Description

An incidence analysis is generated, with each value in the table reflecting the proportion of the population that is above or below a threshold for a specified metric. There is an option to only provide a single hrvar in which a bar plot is generated, or two hrvar values where an incidence table (heatmap) is generated.

Usage

```
create_inc(
  data,
  metric,
  hrvar,
  mingroup = 5,
  threshold,
  position,
  return = "plot"
)

create_incidence(
  data,
  metric,
  hrvar,
  mingroup = 5,
```

```
  threshold,
  position,
  return = "plot"
)
```

Arguments

data	A Standard Person Query dataset in the form of a data frame.
metric	Character string containing the name of the metric, e.g. "Collaboration_hours"
hrvar	Character vector of at most length 2 containing the name of the HR Variable by which to split metrics. Accepts NULL, where the total population is used for the analysis.
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
threshold	Numeric value specifying the threshold.
position	String containing the below valid values: • "above": show incidence of those equal to or above the threshold • "below": show incidence of those equal to or below the threshold
return	String specifying what to return. This must be one of the following strings: • "plot" • "table" See Value for more information.

Value

A different output is returned depending on the value passed to the return argument:

- "plot": 'ggplot' object. A heat map.
- "table": data frame. A summary table.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Flexible: create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_density(), create_dist(), create_fizz(), create_hist(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend()

Examples

```
# Only a single HR attribute
create_inc(
  data = pq_data,
  metric = "After_hours_collaboration_hours",
  hrvar = "Organization",
  threshold = 4,
  position = "above"
)

# Total population (NULL hrvar)
create_inc(
  data = pq_data,
  metric = "Collaboration_hours",
  hrvar = NULL,
  threshold = 20,
  position = "below"
)

# Two HR attributes
create_inc(
  data = pq_data,
  metric = "Collaboration_hours",
  hrvar = c("LevelDesignation", "Organization"),
  threshold = 20,
  position = "below"
)
```

---

create_itsa *Estimate an effect of intervention on every Viva Insights metric in input file by applying single-group Interrupted Time-Series Analysis (ITSA)*

Description

r lifecycle::badge('experimental')

This function implements ITSA method described in the paper 'Conducting interrupted time-series analysis for single- and multiple-group comparisons', Ariel Linden, The Stata Journal (2015), 15, Number 2, pp. 480-500

This function further requires the installation of 'sandwich' and 'lmtest' in order to work. These packages can be installed from CRAN using `install.packages()`.

Usage

```
create_itsa(
  data,
  metrics = NULL,
  before_start = NULL,
```

```
  before_end = NULL,
  after_start = NULL,
  after_end = NULL,
  ac_lags_max = 7,
  return = "table"
)
```

Arguments

data	Person Query as a dataframe in **panel format** (one row per employee per time period), with the columns PersonId and MetricDate present. This function assumes the date format is %Y-%m-%d as is standard in a Viva Insights query output.
metrics	A character vector containing the variable names to perform the interrupted time series analysis for.
before_start	String specifying the start date of the 'before' time period in %Y-%m-%d format. The 'before' time period refers to the period before the intervention (e.g. training program, re-org, shift to remote work) occurs and bounded by before_start and before_end parameters. Longer period increases likelihood of achieving more statistically significant results. Defaults to earliest date in dataset. If not provided, this defaults to the earliest date in the dataset.
before_end	String specifying the end date of 'before' time period in %Y-%m-%d format. If NULL, an error will be raised, as this value is required.
after_start	String specifying the start date of the 'after' time period in %Y-%m-%d format. If NULL, this will default to the value of before_end. The 'after' time period refers to the period after the intervention occurs and is bounded by after_start and after_end parameters. Longer periods increase the likelihood of achieving more statistically significant results.
after_end	String specifying the end date of the 'after' time period in %Y-%m-%d format. Defaults to the latest date in the dataset.
ac_lags_max	Numeric value specifying the maximum lag for the autocorrelation test. The Ljung-Box test is used to check for autocorrelation in the model residuals up to this specified number of lags. Higher values check for longer-term dependencies in the time series data.
return	String specifying what output to return. Defaults to "table". Valid return options include: • 'plot': return a list of plots. • 'table': return data.frame with estimated models' coefficients and their corresponding p-values You should look for significant p-values in beta_2 to indicate an immediate treatment effect, and/or in beta_3 to indicate a treatment effect over time

Details

This function uses the additional package dependencies 'sandwich' and 'lmtest'. Please install these separately from CRAN prior to running the function.

As of May 2022, the 'portes' package was archived from CRAN. The dependency has since been removed and dependent functions `Ljungbox()` incorporated into the **wpa** package.

Value

When 'data' is passed to `return`, a data frame with the following columns:

- `metric_name`: Name of the metric being analyzed.
- `beta_2`: Coefficient for the immediate treatment effect.
- `beta_3`: Coefficient for the treatment effect over time.
- `beta_2_pvalue`: P-value for the immediate treatment effect.
- `beta_3_pvalue`: P-value for the treatment effect over time.
- `AR_flag`: Logical flag indicating whether autocorrelation was detected.
- `error_warning`: Error or warning message if applicable.

Author(s)

Aleksey Ashikhmin alashi@microsoft.com

Examples

```
## Not run:
# Returns summary table
create_itsa(
  data = pq_data,
  metrics = c("Collaboration_span", "Internal_network_size"),
  before_end = "2024-07-01",
  after_start = "2024-07-01",
  ac_lags_max = 7,
  return = "table"
)

# Returns list of plots
plot_list <-
  create_itsa(
    data = pq_data,
    metrics = c("Collaboration_span", "Internal_network_size"),
    before_end = "2024-07-01",
    after_start = "2024-07-01",
    ac_lags_max = 7,
    return = "plot"
  )

# Extract a plot as an example
plot_list$Collaboration_span

## End(Not run)
```

create_IV *Compute Information Value for Predictive Variables*

Description

This function calculates the Information Value (IV) for the selected numeric predictor variables in the dataset, given a specified outcome variable. The Information Value provides a measure of the predictive power of each variable in relation to the outcome variable, which can be useful in feature selection for predictive modeling.

Usage

```
create_IV(
  data,
  predictors = NULL,
  outcome,
  bins = 5,
  siglevel = 0.05,
  exc_sig = FALSE,
  return = "plot"
)
```

Arguments

data	A Person Query dataset in the form of a data frame.
predictors	A character vector specifying the columns to be used as predictors. Defaults to NULL, where all numeric vectors in the data will be used as predictors.
outcome	String specifying the column name for a binary variable, containing only the values 1 or 0.
bins	Number of bins to use, defaults to 5.
siglevel	Significance level to use in comparing populations for the outcomes, defaults to 0.05
exc_sig	Logical value determining whether to exclude values where the p-value lies below what is set at siglevel. Defaults to FALSE, where p-value calculation does not happen altogether.
return	String specifying what to return. This must be one of the following strings: • "plot" • "summary" • "list" • "plot-WOE" • "IV" See Value for more information.

Details

This is a wrapper around `wpa::create_IV()`.

Value

A different output is returned depending on the value passed to the `return` argument:

- `"plot"`: 'ggplot' object. A bar plot showing the IV value of the top (maximum 12) variables.
- `"summary"`: data frame. A summary table for the metric.
- `"list"`: list. A list of outputs for all the input variables.
- `"plot-WOE"`: A list of 'ggplot' objects that show the WOE for each predictor used in the model.
- `"IV"` returns a list object which mirrors the return in `Information::create_infotables()`.

See Also

Other Variable Association: `IV_report()`

Other Information Value: `IV_report()`

Examples

```
# Return a summary table of IV
pq_data %>%
  dplyr::mutate(X = ifelse(Internal_network_size > 40, 1, 0)) %>%
  create_IV(outcome = "X",
            predictors = c("Email_hours",
                           "Meeting_hours",
                           "Chat_hours"),
            return = "plot")


# Return summary
pq_data %>%
  dplyr::mutate(X = ifelse(Internal_network_size > 40, 1, 0)) %>%
  create_IV(outcome = "X",
            predictors = c("Email_hours", "Meeting_hours"),
            return = "summary")
```

---

create_line *Time Trend - Line Chart for any metric*

Description

Provides a week by week view of a selected metric, visualised as line charts. By default returns a line chart for the defined metric, with a separate panel per value in the HR attribute. Additional options available to return a summary table.

Usage

```
create_line(
  data,
  metric,
  hrvar = "Organization",
  mingroup = 5,
  ncol = NULL,
  label = FALSE,
  return = "plot"
)
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present. If your data is already aggregated (e.g. one row per group), use the equivalent *_asis() variant of this function instead.
metric	Character string containing the name of the metric, e.g. "Collaboration_hours"
hrvar	String containing the name of the HR Variable by which to split metrics. Defaults to "Organization". To run the analysis on the total instead of splitting by an HR attribute, supply NULL (without quotes).
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
ncol	Numeric value setting the number of columns on the plot. Defaults to NULL (automatic).
label	Logical value to determine whether to show data point labels on the plot. If TRUE, both geom_point() and geom_text() are added to display data labels rounded to 1 decimal place above each data point. Defaults to FALSE.
return	String specifying what to return. This must be one of the following strings: • "plot" • "table" See Value for more information.

Details

This is a general purpose function that powers all the functions in the package that produce faceted line plots.

Value

A different output is returned depending on the value passed to the return argument:

- "plot": 'ggplot' object. A faceted line plot for the metric.
- "table": data frame. A summary table for the metric.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Flexible: create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_density(), create_dist(), create_fizz(), create_hist(), create_inc(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend()

Other Time-series: create_line_asis(), create_period_scatter(), create_trend()

Examples

```
# Return plot of Email Hours
pq_data %>% create_line(metric = "Email_hours", return = "plot")

# Return plot of Collaboration Hours
pq_data %>% create_line(metric = "Collaboration_hours", return = "plot")

# Return plot but coerce plot to two columns
pq_data %>%
  create_line(
    metric = "Collaboration_hours",
    hrvar = "Organization",
    ncol = 2
    )

# Return plot of email hours and cut by `LevelDesignation`
pq_data %>% create_line(metric = "Email_hours", hrvar = "LevelDesignation")

# Return plot with data point labels
pq_data %>% create_line(metric = "Email_hours", label = TRUE)
```

---

create_line_asis *Create a line chart without aggregation for any metric*

Description

This function creates a line chart directly from the aggregated / summarised data. Unlike `create_line()` which performs a person-level aggregation, there is no calculation for `create_line_asis()` and the values are rendered as they are passed into the function. The only requirement is that a `date_var` is provided for the x-axis.

Usage

```
create_line_asis(
  data,
  date_var = "MetricDate",
  metric,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  ylab = date_var,
  xlab = metric,
  line_colour = rgb2hex(0, 120, 212),
  label = FALSE
)
```

Arguments

`data`	Aggregated or summarised data as a data frame. Unlike `create_line()`, this function does **not** require panel data and can accept any pre-aggregated data frame (i.e. `PersonId` and `MetricDate` are not required).
`date_var`	String containing name of variable for the horizontal axis.
`metric`	String containing name of variable representing the line.
`title`	Title of the plot.
`subtitle`	Subtitle of the plot.
`caption`	Caption of the plot.
`ylab`	Y-axis label for the plot (group axis)
`xlab`	X-axis label of the plot (bar axis).
`line_colour`	String to specify colour to use for the line. Hex codes are accepted. You can also supply RGB values via `rgb2hex()`.
`label`	Logical value to determine whether to show data point labels on the plot. If `TRUE`, both `geom_point()` and `geom_text()` are added to display data labels rounded to 1 decimal place above each data point. Defaults to `FALSE`.

Value

Returns a 'ggplot' object representing a line plot.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Flexible: create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_density(), create_dist(), create_fizz(), create_hist(), create_inc(), create_line(), create_period_scatter(), create_radar(), create_rank(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend()

Other Time-series: create_line(), create_period_scatter(), create_trend()

Examples

```
library(dplyr)

# Median `Emails_sent` grouped by `MetricDate`
# Without Person Averaging
med_df <-
  pq_data %>%
  group_by(MetricDate) %>%
  summarise(Emails_sent_median = median(Emails_sent))

med_df %>%
  create_line_asis(
    date_var = "MetricDate",
    metric = "Emails_sent_median",
    title = "Median Emails Sent",
    subtitle = "Person Averaging Not Applied",
    caption = extract_date_range(pq_data, return = "text")
  )

# Create line plot with data point labels
med_df %>%
  create_line_asis(
    date_var = "MetricDate",
    metric = "Emails_sent_median",
    title = "Median Emails Sent",
    subtitle = "Person Averaging Not Applied",
    caption = extract_date_range(pq_data, return = "text"),
    label = TRUE
  )
```

`create_lorenz` *Calculate the Lorenz Curve and Gini Coefficient in a Person Query*

Description

This function computes the Gini coefficient and plots the Lorenz curve based on a selected metric from a Person Query data frame. It provides a way to measure inequality in the distribution of the selected metric.This function can be integrated into a larger analysis pipeline to assess inequality in metric distribution.

Usage

```
create_lorenz(data, metric, return = "plot")
```

Arguments

`data`	Data frame containing a Person Query.
`metric`	Character string identifying the metric to be used for the Lorenz curve and Gini coefficient calculation.
`return`	Character string identifying the return type. Options are: • `"gini"` - Numeric value representing the Gini coefficient. • `"table"` - Data frame containing a summary table of population share and value share. • `"plot"` (default) - `ggplot` object representing a plot of the Lorenz curve.

Gini coefficient

The Gini coefficient is a measure of statistical dispersion most commonly used to represent income inequality within a population. It is calculated as the ratio of the area between the Lorenz curve and the line of perfect equality (the 45-degree line) to the total area under the line of perfect equality. It has a range of 0 to 1, where 0 represents perfect equality and 1 represents perfect inequality. It can be applied to any Viva Insights metric where inequality is of interest.

Examples

```
create_lorenz(data = pq_data, metric = "Emails_sent", return = "gini")

create_lorenz(data = pq_data, metric = "Emails_sent", return = "plot")

create_lorenz(data = pq_data, metric = "Emails_sent", return = "table")
```

`create_period_scatter` *Period comparison scatter plot for any two metrics*

Description

Returns two side-by-side scatter plots representing two selected metrics, using colour to map an HR attribute and size to represent number of employees. Returns a faceted scatter plot by default, with additional options to return a summary table.

Usage

```
create_period_scatter(
  data,
  hrvar = "Organization",
  metric_x = "Large_and_long_meeting_hours",
  metric_y = "Meeting_hours",
  before_start = min(as.Date(data$MetricDate, "%m/%d/%Y")),
  before_end,
  after_start = as.Date(before_end) + 1,
  after_end = max(as.Date(data$MetricDate, "%m/%d/%Y")),
  before_label = "Period 1",
  after_label = "Period 2",
  mingroup = 5,
  return = "plot"
)
```

Arguments

`data`	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns `PersonId` and `MetricDate` present.
`hrvar`	HR Variable by which to split metrics. Accepts a character vector, defaults to "Organization" but accepts any character vector, e.g. "LevelDesignation"
`metric_x`	Character string containing the name of the metric, e.g. "Collaboration_hours"
`metric_y`	Character string containing the name of the metric, e.g. "Collaboration_hours"
`before_start`	Start date of "before" time period in YYYY-MM-DD
`before_end`	End date of "before" time period in YYYY-MM-DD
`after_start`	Start date of "after" time period in YYYY-MM-DD
`after_end`	End date of "after" time period in YYYY-MM-DD
`before_label`	String to specify a label for the "before" period. Defaults to "Period 1".
`after_label`	String to specify a label for the "after" period. Defaults to "Period 2".
`mingroup`	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
`return`	Character vector specifying what to return, defaults to "plot". Valid inputs are "plot" and "table".

Details

This is a general purpose function that powers all the functions in the package that produce faceted scatter plots.

Value

Returns a 'ggplot' object showing two scatter plots side by side representing the two periods.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Flexible: create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_density(), create_dist(), create_fizz(), create_hist(), create_inc(), create_line(), create_line_asis(), create_radar(), create_rank(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend()

Other Time-series: create_line(), create_line_asis(), create_trend()

Examples

```
# Return plot
create_period_scatter(pq_data,
                      hrvar = "LevelDesignation",
                      before_start = "2024-05-01",
                      before_end = "2024-05-31",
                      after_start = "2024-06-01",
                      after_end = "2024-07-03")

# Return a summary table
create_period_scatter(pq_data, before_end = "2024-05-31", return = "table")
```

`create_radar` *Radar Chart for multiple metrics*

Description

Creates a multi-group radar (spider) chart across a set of metrics.

Core pipeline:

1. Person-level aggregation within group
2. Group-level aggregation
3. Privacy filtering via `mingroup`
4. Optional indexing modes: "total", "none", "ref_group", "minmax"

Optional auto-segmentation:

- If `hrvar` is not supplied, the function will call `identify_usage_segments()` and infer the resulting segment column.

Usage

```
create_radar(
  data,
  metrics,
  hrvar = "Organization",
  mingroup = 5,
  agg = "mean",
  index_mode = "total",
  index_ref_group = NULL,
  na.rm = FALSE,
  return = "plot"
)
```

Arguments

`data`	A Standard Person Query dataset in the form of a data frame.
`metrics`	Character vector of metric column names.
`hrvar`	Character string specifying the grouping column. Defaults to `"Organization"`. If `NULL`, usage segments will be derived via `identify_usage_segments()`.
`mingroup`	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
`agg`	String specifying aggregation method. Either `"mean"` (default) or `"median"`.
`index_mode`	String specifying indexing mode. One of: • `"total"` (default): Total = 100 for each metric • `"none"`: no indexing (raw group values)

- "ref_group": reference group = 100 (requires index_ref_group)
- "minmax": scale to [0, 100] within observed group ranges per metric

index_ref_group
: Character string specifying reference group name when index_mode = "ref_group".

na.rm
: Logical value indicating whether NA rows in required columns are removed prior to aggregation. Defaults to FALSE.

return
: String specifying what to return. One of:

- "plot" (default)
- "table"

Value

A different output is returned depending on the value passed to return:

- "plot": ggplot object (radar chart)
- "table": data frame (group-level indexed table)

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Flexible: create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_density(), create_dist(), create_fizz(), create_hist(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_rank(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend()

Examples

```
create_radar(
  data = pq_data,
  metrics = c("Collaboration_hours", "Email_hours", "Meeting_hours"),
  hrvar = "Organization",
  mingroup = 1
)

# Return the indexed table instead of a plot
create_radar(
  data = pq_data,
```

```
  metrics = c("Collaboration_hours", "Email_hours", "Meeting_hours"),
  hrvar = "LevelDesignation",
  mingroup = 1,
  return = "table"
)
```

`create_radar_calc` *Radar Chart (Calculation)*

Description

Computes group-level metric values and applies optional indexing.

Usage

```
create_radar_calc(
  data,
  metrics,
  hrvar,
  id_col = "PersonId",
  mingroup = 5,
  agg = "mean",
  index_mode = "total",
  index_ref_group = NULL,
  na.rm = FALSE
)
```

Arguments

`data`	data.frame.
`metrics`	character vector of metric column names.
`hrvar`	character string specifying grouping column.
`id_col`	character string specifying person id column.
`mingroup`	numeric minimum unique people per group.
`agg`	"mean" or "median".
`index_mode`	"total","none","ref_group","minmax".
`index_ref_group`	reference group name when index_mode="ref_group".
`na.rm`	logical.

Value

list(table=group_level_table, ref=reference_used)

create_radar_viz *Radar Chart (Visualization)*

Description

Renders a multi-group radar chart using ggplot2 + coord_polar().

Usage

```
create_radar_viz(data, metrics, hrvar, fill_missing = "zero", caption = NULL)
```

Arguments

data	Output table from create_radar_calc().
metrics	Character vector of metric column names.
hrvar	Character string of grouping column name.
fill_missing	Character string specifying how to handle missing values. If "zero" (default), fill NA values as 0 for plotting. This ensures polygons close properly in the radar visualization.
caption	Character string for the plot caption. Typically the output of extract_date_range(data, return = "text") plus an index-mode label, as constructed by create_radar(). Defaults to NULL (no caption).

Value

ggplot object.

create_rank *Rank all groups across HR attributes on a selected Viva Insights metric*

Description

This function scans a standard Person query output for groups with high levels of a given Viva Insights Metric. Returns a plot by default, with an option to return a table with all groups (across multiple HR attributes) ranked by the specified metric.

Usage

```
create_rank(
  data,
  metric,
  hrvar = extract_hr(data, exclude_constants = TRUE),
  mingroup = 5,
  return = "table",
  mode = "simple",
  plot_mode = 1
)
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present. If your data is already aggregated (e.g. one row per group), use the equivalent *_asis() variant of this function instead.
metric	Character string containing the name of the metric, e.g. "Collaboration_hours"
hrvar	String containing the name of the HR Variable by which to split metrics. Defaults to "Organization". To run the analysis on the total instead of splitting by an HR attribute, supply NULL (without quotes).
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
return	String specifying what to return. This must be one of the following strings: • "plot" (default) • "table" See Value for more information.
mode	String to specify calculation mode. Must be either: • "simple" • "combine"
plot_mode	Numeric vector to determine which plot mode to return. Must be either 1 or 2, and is only used when return = "plot". • 1: Top and bottom five groups across the data population are highlighted • 2: Top and bottom groups *per* organizational attribute are highlighted

Value

A different output is returned depending on the value passed to the return argument:

- "plot": 'ggplot' object. A bubble plot where the x-axis represents the metric, the y-axis represents the HR attributes, and the size of the bubbles represent the size of the organizations. Note that there is no plot output if mode is set to "combine".
- "table": data frame. A summary table for the metric.

Author(s)

Carlos Morales Torrado carlos.morales@microsoft.com

Martin Chan martin.chan@microsoft.com

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(),

create_radar(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Flexible: create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_density(), create_dist(), create_fizz(), create_hist(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend()

Examples

```
pq_data_small <- dplyr::slice_sample(pq_data, prop = 0.1)

# Plot mode 1 - show top and bottom five groups
create_rank(
  data = pq_data_small,
  hrvar = c("FunctionType", "LevelDesignation"),
  metric = "Emails_sent",
  return = "plot",
  plot_mode = 1
)

# Plot mode 2 - show top and bottom groups per HR variable
create_rank(
  data = pq_data_small,
  hrvar = c("FunctionType", "LevelDesignation"),
  metric = "Emails_sent",
  return = "plot",
  plot_mode = 2
)

# Return a table
create_rank(
  data = pq_data_small,
  metric = "Emails_sent",
  return = "table"
)


# Return a table - combination mode
create_rank(
  data = pq_data_small,
  metric = "Emails_sent",
  mode = "combine",
  return = "table"
)
```

`create_rank_combine` *Create combination pairs of HR variables and run 'create_rank()'*

Description

Create pairwise combinations of HR variables and compute an average of a specified advanced insights metric.

Usage

```
create_rank_combine(data, hrvar = extract_hr(data), metric, mingroup = 5)
```

Arguments

`data`	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns `PersonId` and `MetricDate` present. If your data is already aggregated (e.g. one row per group), use the equivalent `*_asis()` variant of this function instead.
`hrvar`	String containing the name of the HR Variable by which to split metrics. Defaults to `"Organization"`. To run the analysis on the total instead of splitting by an HR attribute, supply `NULL` (without quotes).
`metric`	Character string containing the name of the metric, e.g. "Collaboration_hours"
`mingroup`	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.

Details

This function is called when the `mode` argument in `create_rank()` is specified as `"combine"`.

Value

Data frame containing the following variables:

- `hrvar`: placeholder column that denotes the output as `"Combined"`.
- `group`: pairwise combinations of HR attributes with the HR attribute in square brackets followed by the value of the HR attribute.
- Name of the metric (as passed to `metric`)
- `n`

Examples

```
# Use a small sample for faster runtime
pq_data_small <- dplyr::slice_sample(pq_data, prop = 0.1)

create_rank_combine(
  data = pq_data_small,
  metric = "Email_hours",
  hrvar = c("Organization", "FunctionType", "LevelDesignation")
)
```

`create_rogers` *Generate Rogers Adoption Curve plots for Copilot usage*

Description

Creates various visualizations based on the Rogers adoption curve theory, analyzing the adoption patterns of Copilot usage. The function identifies habitual users using the `identify_habit()` function and then creates adoption curve visualizations based on different time frames and organizational groupings.

Usage

```
create_rogers(
  data,
  hrvar = NULL,
  metric,
  width = 9,
  max_window = 12,
  threshold = 1,
  start_metric = NULL,
  return = "plot",
  plot_mode = 1,
  label = FALSE
)
```

Arguments

`data`	Data frame containing Person Query data to be analyzed. Must contain `PersonId`, `MetricDate`, and the specified metrics.
`hrvar`	Character string specifying the HR attribute or organizational variable to group by. Default is `NULL`, for no grouping.
`metric`	Character string containing the name of the metric to analyze for habit identification, e.g. "Total_Copilot_actions". This is passed to `identify_habit()`.
`width`	Integer specifying the number of qualifying counts to consider for a habit. Passed to `identify_habit()`. Default is 9.

`max_window` Integer specifying the maximum unit of dates to consider a qualifying window for a habit. Passed to `identify_habit()`. Default is 12.

`threshold` Numeric value specifying the minimum threshold for the metric to be considered a qualifying count. Passed to `identify_habit()`. Default is 1.

`start_metric` Character string containing the name of the metric used for determining enablement start date. This metric should track when users first gained access to the technology being analyzed. The function identifies the earliest date where this metric is greater than 0 for each user as their "enablement date". This is then used in plot modes 3 and 4 to calculate time-to-adoption and Rogers segment classifications. The suggested variable is "Total_Copilot_enabled_days", but any metric that indicates access or licensing status can be used (e.g., "License_assigned_days", "Access_granted"). This parameter is optional for plot modes 1 and 2, but required for plot modes 3 and 4. When `return = "data"` and `start_metric` is provided, Rogers segment classifications will be included in the returned data frame. Default is `NULL`.

`return` Character vector specifying what to return. Valid inputs are "plot", "data", and "table". Default is "plot".

`plot_mode` Integer or character string determining which plot to return. Valid inputs are:

- 1 or "cumulative": Rogers Adoption Curve showing cumulative adoption
- 2 or "weekly": Weekly Rate of adoption showing new habitual users
- 3 or "enablement": Enablement-based adoption rate with Rogers segments
- 4 or "cumulative_enablement": Cumulative adoption adjusted for enablement

Default is 1.

`label` Logical value to determine whether to show data point labels on the plot for cumulative adoption curves (plot modes 1 and 4). If `TRUE`, both `geom_point()` and `geom_text()` are added to display data labels rounded to 1 decimal place above each data point. Defaults to `FALSE`.

Details

This function provides four distinct plot modes to analyze adoption patterns:

Plot Mode 1 - Cumulative Adoption Curve: Shows the classic Rogers adoption curve with cumulative percentage of habitual users over time. This S-shaped curve helps identify the pace of adoption and when saturation begins. Steep sections indicate rapid adoption periods, while flat sections suggest slower uptake or natural limits.

Plot Mode 2 - Weekly Adoption Rate: Displays the number of new habitual users identified each week, with a 3-week moving average line to smooth volatility. This view helps identify adoption spikes, seasonal patterns, and the natural ebb and flow of user onboarding. High bars indicate successful onboarding periods.

Plot Mode 3 - Enablement-Based Adoption: Analyzes adoption relative to when users were first enabled (had access). Users are classified into Rogers segments (Innovators, Early Adopters, Early/Late Majority, Laggards) based on how quickly they adopted after enablement. This helps understand the natural distribution of adoption speed within your organization.

Plot Mode 4 - Cumulative Enablement-Adjusted: Similar to Mode 1 but only includes users who had enablement data, providing a more accurate view of adoption among those who actually had access to the technology. This removes noise from users who may not have been properly enabled.

Interpretation Guidelines:

- Early steep curves suggest strong product-market fit
- Plateaus may indicate training needs or feature limitations
- Seasonal patterns often reflect organizational training cycles
- Rogers segments help identify user personas for targeted interventions

Value

Returns a 'ggplot' object by default when 'plot' is passed in `return`. When 'table' is passed, a summary table is returned as a data frame. When 'data' is passed, the processed data with habit classifications is returned.

When `return = "data"`, the returned data frame includes:

- All original columns from the input data
- `IsHabit`: Binary indicator of whether the user has developed a habit
- `adoption_week`: The week when the user first exhibited habitual behavior
- `enable_week`: (if `start_metric` provided) The week when the user was first enabled
- `weeks_to_adopt`: (if `start_metric` provided) Number of weeks from enablement to adoption
- `RogersSegment`: (if `start_metric` provided) Rogers adoption segment classification:
  - "Innovators" (fastest 2.5\
  - "Early Adopters" (next 13.5\
  - "Early Majority" (next 34\
  - "Late Majority" (next 34\
  - "Laggards" (slowest 16\

Author(s)

Chris Gideon chris.gideon@microsoft.com

See Also

Other Visualization: `afterhours_dist()`, `afterhours_fizz()`, `afterhours_line()`, `afterhours_rank()`, `afterhours_summary()`, `afterhours_trend()`, `collaboration_area()`, `collaboration_dist()`, `collaboration_fizz()`, `collaboration_line()`, `collaboration_rank()`, `collaboration_sum()`, `collaboration_trend()`, `create_bar()`, `create_bar_asis()`, `create_boxplot()`, `create_bubble()`, `create_dist()`, `create_fizz()`, `create_inc()`, `create_line()`, `create_line_asis()`, `create_period_scatter()`, `create_radar()`, `create_rank()`, `create_sankey()`, `create_scatter()`, `create_stacked()`, `create_survival()`, `create_tracking()`, `create_trend()`, `email_dist()`, `email_fizz()`, `email_line()`, `email_rank()`, `email_summary()`, `email_trend()`, `external_dist()`, `external_fizz()`, `external_line()`, `external_rank()`, `external_sum()`, `hr_trend()`, `hrvar_count()`, `hrvar_trend()`, `keymetrics_scan()`, `meeting_dist()`, `meeting_fizz()`, `meeting_line()`, `meeting_rank()`, `meeting_summary()`, `meeting_trend()`, `one2one_dist()`, `one2one_fizz()`, `one2one_freq()`, `one2one_line()`, `one2one_rank()`, `one2one_sum()`, `one2one_trend()`

Examples

```
# Basic Rogers adoption curve
create_rogers(
  data = pq_data,
  metric = "Copilot_actions_taken_in_Teams",
  plot_mode = 1
)

# Weekly adoption rate by organization
create_rogers(
  data = pq_data,
  hrvar = "Organization",
  metric = "Copilot_actions_taken_in_Teams",
  plot_mode = 2
)

# Enablement-based adoption
create_rogers(
  data = pq_data,
  metric = "Copilot_actions_taken_in_Teams",
  start_metric = "Total_Copilot_enabled_days",
  plot_mode = 3
)

# Return data with Rogers segments
rogers_data <- create_rogers(
  data = pq_data,
  metric = "Copilot_actions_taken_in_Teams",
  start_metric = "Total_Copilot_enabled_days",
  return = "data"
)

# Rogers adoption curve with data point labels
create_rogers(
  data = pq_data,
  metric = "Copilot_actions_taken_in_Teams",
  plot_mode = 1,
  label = TRUE
)
```

`create_sankey` *Create a sankey chart from a two-column count table*

Description

Create a 'networkD3' style sankey chart based on a long count table with two variables. The input data should have three columns, where each row is a unique group:

1. Variable 1

2. Variable 2
3. Count

Usage

```
create_sankey(data, var1, var2, count = "n")
```

Arguments

data	Data frame of the long count table.
var1	String containing the name of the variable to be shown on the left.
var2	String containing the name of the variable to be shown on the right.
count	String containing the name of the count variable.

Value

A 'sankeyNetwork' and 'htmlwidget' object containing a two-tier sankey plot. The output can be saved locally with `htmlwidgets::saveWidget()`.

See Also

Other Visualization: `afterhours_dist()`, `afterhours_fizz()`, `afterhours_line()`, `afterhours_rank()`, `afterhours_summary()`, `afterhours_trend()`, `collaboration_area()`, `collaboration_dist()`, `collaboration_fizz()`, `collaboration_line()`, `collaboration_rank()`, `collaboration_sum()`, `collaboration_trend()`, `create_bar()`, `create_bar_asis()`, `create_boxplot()`, `create_bubble()`, `create_dist()`, `create_fizz()`, `create_inc()`, `create_line()`, `create_line_asis()`, `create_period_scatter()`, `create_radar()`, `create_rank()`, `create_rogers()`, `create_scatter()`, `create_stacked()`, `create_survival()`, `create_tracking()`, `create_trend()`, `email_dist()`, `email_fizz()`, `email_line()`, `email_rank()`, `email_summary()`, `email_trend()`, `external_dist()`, `external_fizz()`, `external_line()`, `external_rank()`, `external_sum()`, `hr_trend()`, `hrvar_count()`, `hrvar_trend()`, `keymetrics_scan()`, `meeting_dist()`, `meeting_fizz()`, `meeting_line()`, `meeting_rank()`, `meeting_summary()`, `meeting_trend()`, `one2one_dist()`, `one2one_fizz()`, `one2one_freq()`, `one2one_line()`, `one2one_rank()`, `one2one_sum()`, `one2one_trend()`

Other Flexible: `create_bar()`, `create_bar_asis()`, `create_boxplot()`, `create_bubble()`, `create_density()`, `create_dist()`, `create_fizz()`, `create_hist()`, `create_inc()`, `create_line()`, `create_line_asis()`, `create_period_scatter()`, `create_radar()`, `create_rank()`, `create_scatter()`, `create_stacked()`, `create_survival()`, `create_tracking()`, `create_trend()`

Examples

```
pq_data %>%
  dplyr::count(Organization, FunctionType) %>%
  create_sankey(var1 = "Organization", var2 = "FunctionType")
```

`create_scatter` *Create a Scatter plot with two selected Viva Insights metrics (General Purpose)*

Description

Returns a scatter plot of two selected metrics, using colour to map an HR attribute. Returns a scatter plot by default, with additional options to return a summary table.

Usage

```
create_scatter(
  data,
  metric_x,
  metric_y,
  hrvar = "Organization",
  mingroup = 5,
  return = "plot"
)
```

Arguments

`data`	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns `PersonId` and `MetricDate` present.
`metric_x`	Character string containing the name of the metric, e.g. "Collaboration_hours"
`metric_y`	Character string containing the name of the metric, e.g. "Collaboration_hours"
`hrvar`	HR Variable by which to split metrics, defaults to "Organization" but accepts any character vector, e.g. "LevelDesignation"
`mingroup`	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
`return`	Character vector specifying what to return, defaults to "plot". Valid inputs are "plot" and "table".

Details

This is a general purpose function that powers all the functions in the package that produce scatter plots.

Value

Returns a 'ggplot' object by default, where 'plot' is passed in `return`. When 'table' is passed, a summary table is returned as a data frame.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Flexible: create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_density(), create_dist(), create_fizz(), create_hist(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_sankey(), create_stacked(), create_survival(), create_tracking(), create_trend()

Examples

```
create_scatter(
  pq_data,
  metric_x = "Collaboration_hours",
  metric_y = "Multitasking_hours",
  hrvar = "Organization"
  )

create_scatter(
  pq_data,
  metric_x = "Collaboration_hours",
  metric_y = "Multitasking_hours",
  hrvar = "Organization",
  mingroup = 100,
  return = "plot"
)
```

---

create_stacked *Horizontal stacked bar plot for any metric*

Description

Creates either a single bar plot, of a stacked bar using selected metrics (where the typical use case is to create different definitions of collaboration hours). Returns a plot by default. Additional options available to return a summary table.

Usage

```
create_stacked(
  data,
  hrvar = "Organization",
  metrics = c("Meeting_hours", "Email_hours"),
  mingroup = 5,
  return = "plot",
  stack_colours = c("#1d627e", "#34b1e2", "#b4d5dd", "#adc0cb"),
  percent = FALSE,
  plot_title = "Collaboration Hours",
  plot_subtitle = paste("Average by", tolower(camel_clean(hrvar))),
  legend_lab = NULL,
  rank = "descending",
  xlim = NULL,
  text_just = 0.5,
  text_colour = "#FFFFFF"
)
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns `PersonId` and `MetricDate` present. If your data is already aggregated (e.g. one row per group), use the equivalent `*_asis()` variant of this function instead.
hrvar	String containing the name of the HR Variable by which to split metrics. Defaults to `"Organization"`. To run the analysis on the total instead of splitting by an HR attribute, supply `NULL` (without quotes).
metrics	A character vector to specify variables to be used in calculating the "Total" value, e.g. c("Meeting_hours", "Email_hours"). The order of the variable names supplied determine the order in which they appear on the stacked plot.
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
return	Character vector specifying what to return, defaults to "plot". Valid inputs are "plot" and "table".
stack_colours	A character vector to specify the colour codes for the stacked bar charts.
percent	Logical value to determine whether to show labels as percentage signs. Defaults to `FALSE`.
plot_title	String. Option to override plot title.
plot_subtitle	String. Option to override plot subtitle.
legend_lab	String. Option to override legend title/label. Defaults to `NULL`, where the metric name will be populated instead.
rank	String specifying how to rank the bars. Valid inputs are: • `"descending"` - ranked highest to lowest from top to bottom (default). • `"ascending"` - ranked lowest to highest from top to bottom.

- NULL - uses the original levels of the HR attribute.

`xlim` An option to set max value in x axis.

`text_just` **[Experimental]** A numeric value controlling for the horizontal position of the text labels. Defaults to 0.5.

`text_colour` **[Experimental]** String to specify colour to use for the text labels. Defaults to `"#FFFFFF"`.

Value

Returns a 'ggplot' object by default, where 'plot' is passed in `return`. When 'table' is passed, a summary table is returned as a data frame.

See Also

Other Visualization: `afterhours_dist()`, `afterhours_fizz()`, `afterhours_line()`, `afterhours_rank()`, `afterhours_summary()`, `afterhours_trend()`, `collaboration_area()`, `collaboration_dist()`, `collaboration_fizz()`, `collaboration_line()`, `collaboration_rank()`, `collaboration_sum()`, `collaboration_trend()`, `create_bar()`, `create_bar_asis()`, `create_boxplot()`, `create_bubble()`, `create_dist()`, `create_fizz()`, `create_inc()`, `create_line()`, `create_line_asis()`, `create_period_scatter()`, `create_radar()`, `create_rank()`, `create_rogers()`, `create_sankey()`, `create_scatter()`, `create_survival()`, `create_tracking()`, `create_trend()`, `email_dist()`, `email_fizz()`, `email_line()`, `email_rank()`, `email_summary()`, `email_trend()`, `external_dist()`, `external_fizz()`, `external_line()`, `external_rank()`, `external_sum()`, `hr_trend()`, `hrvar_count()`, `hrvar_trend()`, `keymetrics_scan()`, `meeting_dist()`, `meeting_fizz()`, `meeting_line()`, `meeting_rank()`, `meeting_summary()`, `meeting_trend()`, `one2one_dist()`, `one2one_fizz()`, `one2one_freq()`, `one2one_line()`, `one2one_rank()`, `one2one_sum()`, `one2one_trend()`

Other Flexible: `create_bar()`, `create_bar_asis()`, `create_boxplot()`, `create_bubble()`, `create_density()`, `create_dist()`, `create_fizz()`, `create_hist()`, `create_inc()`, `create_line()`, `create_line_asis()`, `create_period_scatter()`, `create_radar()`, `create_rank()`, `create_sankey()`, `create_scatter()`, `create_survival()`, `create_tracking()`, `create_trend()`

Examples

```
pq_data %>%
  create_stacked(hrvar = "LevelDesignation",
                 metrics = c("Meeting_hours", "Email_hours"),
                 return = "plot")

pq_data %>%
  create_stacked(hrvar = "FunctionType",
                 metrics = c("Meeting_hours",
                             "Email_hours",
                             "Call_hours",
                             "Chat_hours"),
                 return = "plot",
                 rank = "ascending")

pq_data %>%
  create_stacked(hrvar = "FunctionType",
```

```
metrics = c("Meeting_hours",
            "Email_hours",
            "Call_hours",
            "Chat_hours"),
return = "table")
```

`create_survival` *Kaplan–Meier Survival Curve*

Description

Computes Kaplan–Meier survival curves and returns a step-function plot by default, with an option to return the underlying long-format survival table.

Although this function is built on **survival analysis**, the framing applies equally to positive milestones in a workforce context. In classical survival analysis the "event" is typically something negative (death, failure); here it is often a positive outcome — first use of a tool, first week as a power user, first week crossing a collaboration threshold. The curve is therefore better read as a **time-to-adoption**, **conversion**, or **graduation** curve:

- The y-axis ("survival probability") represents the share of people who have *not yet* reached the milestone.
- A curve that drops steeply early means most people converted quickly.
- A curve that stays high means many people had not yet converted by the end of the observation window.

Supports:

- Flexible grouping via `hrvar`
- Privacy filtering via `mingroup`

This function expects **one row per person** with a pre-computed time-to-event column and an event indicator. Use `create_survival_prep` to derive these from a Standard Person Query panel dataset.

Usage

```
create_survival(
  data,
  time_col,
  event_col,
  hrvar = NULL,
  mingroup = 5,
  na.rm = TRUE,
  return = "plot"
)
```

Arguments

data	A person-level data frame with one row per person, as produced by create_survival_prep. Must contain time_col, event_col, and (when hrvar is not NULL) the grouping column.
time_col	Character string containing the name of the time-to-event column.
event_col	Character string containing the name of the event indicator column. Accepted forms: • Logical (TRUE/FALSE) • Numeric 0/1 • Numeric event-like (>=0): values >0 treated as event, 0 as censored • Character tokens ("true"/"false", "yes"/"no", "1"/"0")
hrvar	Character string containing the name of the grouping column. Supply NULL (without quotes) to compute a single overall curve.
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
na.rm	A logical value indicating whether NA should be stripped before computation proceeds. Defaults to TRUE.
return	String specifying what to return. This must be one of: • "plot" (default) • "table"

Value

A different output is returned depending on the value passed to the return argument:

- "plot": 'ggplot' object. A Kaplan–Meier survival curve by group.
- "table": data frame. A long-format survival table by group.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Flexible: create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_density(), create_dist(), create_fizz(), create_hist(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_sankey(), create_scatter(), create_stacked(), create_tracking(), create_trend()

Examples

```
## Not run:
library(vivainsights)
data("pq_data", package = "vivainsights")

# Step 1: convert panel data to person-level survival format
surv_data <- create_survival_prep(
  data = pq_data,
  metric = "Copilot_actions_taken_in_Teams"
)

# Step 2: plot Kaplan-Meier curves by organisation
create_survival(
  data = surv_data,
  time_col = "time",
  event_col = "event",
  hrvar = "Organization"
)

# Return the survival table instead
create_survival(
  data = surv_data,
  time_col = "time",
  event_col = "event",
  hrvar = "Organization",
  return = "table"
)

## End(Not run)
```

---

create_survival_calc *Kaplan–Meier Survival Curve (Calculation)*

Description

Computes a long-format Kaplan–Meier survival table by group and applies a minimum group size threshold (`mingroup`).

Usage

```
create_survival_calc(
  data,
  time_col,
  event_col,
  hrvar = NULL,
  id_col = "PersonId",
  mingroup = 5,
  na.rm = TRUE
)
```

Arguments

data	data.frame.
time_col	Character. Time-to-event column name.
event_col	Character. Event indicator column name.
hrvar	Character or NULL. Grouping column name.
id_col	Character. Optional id column name for distinct counts.
mingroup	Numeric. Minimum group size.
na.rm	Logical. Drop missing values in required columns.

Value

A list with elements:

- table: long survival table with columns (group, time, survival, at_risk, events, n)
- counts: group size table

---

create_survival_prep *Prepare Survival Data from a Panel Dataset*

Description

Converts a Standard Person Query (panel) dataset into a person-level survival table suitable for direct use with create_survival.

Each person's weekly rows are reduced to a single row containing:

- time: the number of periods from the first observation until the event condition is first met. If the condition is never met, time equals the total number of observed periods (censored).
- event: 1 if the condition was met at any point in the observation window, 0 if the person was censored (never met it).

Although rooted in survival analysis, the event_condition is typically a **positive milestone** in a workforce context — first use of a tool, first week as a power user, first week crossing a collaboration threshold. The resulting curve is therefore often better described as a **time-to-adoption**, **conversion**, or **graduation** curve. See create_survival for further framing guidance.

Usage

```
create_survival_prep(
  data,
  metric,
  event_condition = function(x) x > 0,
  hrvar = "Organization"
)
```

Arguments

data	A Standard Person Query dataset in the form of a data frame.
metric	Character string containing the name of the metric column to evaluate the event condition on.
event_condition	A function that takes a numeric vector and returns a logical vector. Applied to metric to mark whether the event occurred in a given period. Defaults to function(x) x > 0 (any non-zero value is treated as the event).
hrvar	Character string containing the name of the HR attribute column to carry through into the output. The most recent (last observed) value per person is used. Defaults to "Organization". Set to NULL to omit.

Value

A data frame with one row per person containing:

- PersonId
- time: numeric — periods until event, or total observed periods if censored
- event: integer — 1 (event) or 0 (censored)
- The hrvar column, if supplied

See Also

create_survival

Examples

```
# \dontrun{
# library(vivainsights)
# data("pq_data", package = "vivainsights")
#
# # Step 1: derive person-level survival data
# surv_data <- create_survival_prep(
#   data = pq_data,
#   metric = "Copilot_actions_taken_in_Teams",
#   event_condition = function(x) x > 0
# )
#
# # Step 2: visualise
# create_survival(
#   data = surv_data,
#   time_col = "time",
#   event_col = "event",
#   hrvar = "Organization"
# )
# }
```

create_survival_viz *Kaplan–Meier Survival Curve (Visualization)*

Description

Renders a Kaplan–Meier step curve by group from a long-format survival table.

Usage

```
create_survival_viz(
  data,
  hrvar = "group",
  title = "Survival Curve",
  subtitle = "Kaplan-Meier estimate",
  caption = NULL
)
```

Arguments

data	Long survival table produced by create_survival_calc().
hrvar	Character. Group column name in data.
title, subtitle	Character. Plot annotations.
caption	Character string for the plot caption. Typically the output of extract_date_range(data, return = "text"), as constructed by create_survival(). Defaults to NULL (no caption).

Value

ggplot object.

create_tracking *Create a line chart that tracks metrics over time with a 4-week rolling average*

Description

[Experimental]

Create a two-series line chart that visualizes a set of metric over time for the selected population, with one of the series being a four-week rolling average.

Usage

```
create_tracking(
  data,
  metric,
  plot_title = us_to_space(metric),
  plot_subtitle = "Measure over time",
  percent = FALSE
)
```

Arguments

data	A Standard Person Query dataset in the form of a data frame.
metric	Character string containing the name of the metric, e.g. "Collaboration_hours" percentage signs. Defaults to FALSE.
plot_title	An option to override plot title.
plot_subtitle	An option to override plot subtitle.
percent	Logical value to determine whether to show labels as percentage signs. Defaults to FALSE.

Value

A different output is returned depending on the value passed to the return argument:

- "plot": 'ggplot' object. A time-series plot for the metric.
- "table": data frame. A summary table for the metric.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Flexible: create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_density(), create_dist(), create_fizz(), create_hist(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_trend()

Examples

```
pq_data %>%
  create_tracking(
    metric = "Collaboration_hours",
    percent = FALSE
  )
```

`create_trend` *Heat mapped horizontal bar plot over time for any metric*

Description

Provides a week by week view of a selected Viva Insights metric. By default returns a week by week heatmap bar plot, highlighting the points in time with most activity. Additional options available to return a summary table.

Usage

```
create_trend(
  data,
  metric,
  hrvar = "Organization",
  mingroup = 5,
  palette = c("steelblue4", "aliceblue", "white", "mistyrose1", "tomato1"),
  return = "plot",
  legend_title = "Hours"
)
```

Arguments

`data`	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns `PersonId` and `MetricDate` present. If your data is already aggregated (e.g. one row per group), use the equivalent `*_asis()` variant of this function instead.
`metric`	Character string containing the name of the metric, e.g. "Collaboration_hours"
`hrvar`	String containing the name of the HR Variable by which to split metrics. Defaults to `"Organization"`. To run the analysis on the total instead of splitting by an HR attribute, supply `NULL` (without quotes).
`mingroup`	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
`palette`	Character vector containing colour codes, ranked from the lowest value to the highest value. This is passed directly to `ggplot2::scale_fill_gradientn()`.
`return`	Character vector specifying what to return, defaults to `"plot"`. Valid inputs are "plot" and "table".
`legend_title`	String to be used as the title of the legend. Defaults to `"Hours"`.

Value

Returns a 'ggplot' object by default, where 'plot' is passed in `return`. When 'table' is passed, a summary table is returned as a data frame.

See Also

Other Visualization: `afterhours_dist()`, `afterhours_fizz()`, `afterhours_line()`, `afterhours_rank()`, `afterhours_summary()`, `afterhours_trend()`, `collaboration_area()`, `collaboration_dist()`, `collaboration_fizz()`, `collaboration_line()`, `collaboration_rank()`, `collaboration_sum()`, `collaboration_trend()`, `create_bar()`, `create_bar_asis()`, `create_boxplot()`, `create_bubble()`, `create_dist()`, `create_fizz()`, `create_inc()`, `create_line()`, `create_line_asis()`, `create_period_scatter()`, `create_radar()`, `create_rank()`, `create_rogers()`, `create_sankey()`, `create_scatter()`, `create_stacked()`, `create_survival()`, `create_tracking()`, `email_dist()`, `email_fizz()`, `email_line()`, `email_rank()`, `email_summary()`, `email_trend()`, `external_dist()`, `external_fizz()`, `external_line()`, `external_rank()`, `external_sum()`, `hr_trend()`, `hrvar_count()`, `hrvar_trend()`, `keymetrics_scan()`, `meeting_dist()`, `meeting_fizz()`, `meeting_line()`, `meeting_rank()`, `meeting_summary()`, `meeting_trend()`, `one2one_dist()`, `one2one_fizz()`, `one2one_freq()`, `one2one_line()`, `one2one_rank()`, `one2one_sum()`, `one2one_trend()`

Other Flexible: `create_bar()`, `create_bar_asis()`, `create_boxplot()`, `create_bubble()`, `create_density()`, `create_dist()`, `create_fizz()`, `create_hist()`, `create_inc()`, `create_line()`, `create_line_asis()`, `create_period_scatter()`, `create_radar()`, `create_rank()`, `create_sankey()`, `create_scatter()`, `create_stacked()`, `create_survival()`, `create_tracking()`

Other Time-series: `create_line()`, `create_line_asis()`, `create_period_scatter()`

Examples

```
create_trend(pq_data, metric = "Collaboration_hours", hrvar = "LevelDesignation")

# custom colours
create_trend(
  pq_data,
  metric = "Collaboration_hours",
  hrvar = "LevelDesignation",
  palette = c(
    "#FB6107",
    "#F3DE2C",
    "#7CB518",
    "#5C8001"
  )
  )
```

`cut_hour` *Convert a numeric variable for hours into categorical*

Description

Supply a numeric variable, e.g. `Collaboration_hours`, and return a character vector.

Usage

```
cut_hour(metric, cuts, unit = "hours", lbound = 0, ubound = 100)
```

Arguments

metric	A numeric variable representing hours.
cuts	A numeric vector of minimum length 3 to represent the cut points required. The minimum and maximum values provided in the vector are inclusive.
unit	String to specify the unit of the labels. Defaults to "hours".
lbound	Numeric. Specifies the lower bound (inclusive) value for the minimum label. Defaults to 0.
ubound	Numeric. Specifies the upper bound (inclusive) value for the maximum label. Defaults to 100.

Details

This is used within `create_dist()` for numeric to categorical conversion.

Value

Character vector representing a converted categorical variable, appended with the label of the unit. See `examples` for more information.

See Also

Other Support: `any_idate()`, `camel_clean()`, `check_inputs()`, `extract_date_range()`, `extract_hr()`, `heat_colours()`, `is_date_format()`, `maxmin()`, `pairwise_count()`, `read_preamble()`, `rgb2hex()`, `totals_bind()`, `totals_col()`, `tstamp()`, `us_to_space()`, `wrap()`

Examples

```
# Direct use
cut_hour(1:30, cuts = c(15, 20, 25))

# Use on a query
cut_hour(pq_data$Collaboration_hours, cuts = c(10, 15, 20), ubound = 150)
```

---

email_dist *Distribution of Email Hours as a 100% stacked bar*

Description

Analyze Email Hours distribution. Returns a stacked bar plot by default. Additional options available to return a table with distribution elements.

Usage

```
email_dist(
  data,
  hrvar = "Organization",
  mingroup = 5,
  return = "plot",
  cut = c(0.5, 1, 1.5)
)
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present. If your data is already aggregated (e.g. one row per group), use the equivalent *_asis() variant of this function instead.
hrvar	String containing the name of the HR Variable by which to split metrics. Defaults to "Organization". To run the analysis on the total instead of splitting by an HR attribute, supply NULL (without quotes).
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
return	String specifying what to return. This must be one of the following strings: • "plot" • "table" See Value for more information.
cut	A numeric vector of length three to specify the breaks for the distribution, e.g. c(10, 15, 20)

Value

A different output is returned depending on the value passed to the return argument:

- "plot": 'ggplot' object. A stacked bar plot for the metric.
- "table": data frame. A summary table for the metric.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(),

meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Emails: email_fizz(), email_line(), email_rank(), email_summary(), email_trend()

Examples

```
# Return plot
email_dist(pq_data, hrvar = "Organization")

# Return summary table
email_dist(pq_data, hrvar = "Organization", return = "table")

# Return result with a custom specified breaks
email_dist(pq_data, hrvar = "LevelDesignation", cut = c(1, 2, 3))
```

---

email_fizz *Distribution of Email Hours (Fizzy Drink plot)*

Description

Analyze weekly email hours distribution, and returns a 'fizzy' scatter plot by default. Additional options available to return a table with distribution elements.

Usage

```
email_fizz(data, hrvar = "Organization", mingroup = 5, return = "plot")
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present. If your data is already aggregated (e.g. one row per group), use the equivalent *_asis() variant of this function instead.
hrvar	String containing the name of the HR Variable by which to split metrics. Defaults to "Organization". To run the analysis on the total instead of splitting by an HR attribute, supply NULL (without quotes).
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
return	String specifying what to return. This must be one of the following strings: • "plot" • "table" See Value for more information.

Value

A different output is returned depending on the value passed to the `return` argument:

- `"plot"`: 'ggplot' object. A jittered scatter plot for the metric.
- `"table"`: data frame. A summary table for the metric.

See Also

Other Visualization: `afterhours_dist()`, `afterhours_fizz()`, `afterhours_line()`, `afterhours_rank()`, `afterhours_summary()`, `afterhours_trend()`, `collaboration_area()`, `collaboration_dist()`, `collaboration_fizz()`, `collaboration_line()`, `collaboration_rank()`, `collaboration_sum()`, `collaboration_trend()`, `create_bar()`, `create_bar_asis()`, `create_boxplot()`, `create_bubble()`, `create_dist()`, `create_fizz()`, `create_inc()`, `create_line()`, `create_line_asis()`, `create_period_scatter()`, `create_radar()`, `create_rank()`, `create_rogers()`, `create_sankey()`, `create_scatter()`, `create_stacked()`, `create_survival()`, `create_tracking()`, `create_trend()`, `email_dist()`, `email_line()`, `email_rank()`, `email_summary()`, `email_trend()`, `external_dist()`, `external_fizz()`, `external_line()`, `external_rank()`, `external_sum()`, `hr_trend()`, `hrvar_count()`, `hrvar_trend()`, `keymetrics_scan()`, `meeting_dist()`, `meeting_fizz()`, `meeting_line()`, `meeting_rank()`, `meeting_summary()`, `meeting_trend()`, `one2one_dist()`, `one2one_fizz()`, `one2one_freq()`, `one2one_line()`, `one2one_rank()`, `one2one_sum()`, `one2one_trend()`

Other Emails: `email_dist()`, `email_line()`, `email_rank()`, `email_summary()`, `email_trend()`

Examples

```
# Return plot
email_fizz(pq_data, hrvar = "Organization", return = "plot")

# Return summary table
email_fizz(pq_data, hrvar = "Organization", return = "table")
```

---

email_line *Email Time Trend - Line Chart*

Description

Provides a week by week view of email time, visualised as line charts. By default returns a line chart for email hours, with a separate panel per value in the HR attribute. Additional options available to return a summary table.

Usage

```
email_line(
  data,
  hrvar = "Organization",
  mingroup = 5,
  return = "plot",
  label = FALSE
)
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present. If your data is already aggregated (e.g. one row per group), use the equivalent *_asis() variant of this function instead.
hrvar	String containing the name of the HR Variable by which to split metrics. Defaults to "Organization". To run the analysis on the total instead of splitting by an HR attribute, supply NULL (without quotes).
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
return	String specifying what to return. This must be one of the following strings: • "plot" • "table" See Value for more information.
label	Logical value to determine whether to show data point labels on the plot. If TRUE, both geom_point() and geom_text() are added to display data labels rounded to 1 decimal place above each data point. Defaults to FALSE.

Value

A different output is returned depending on the value passed to the return argument:

- "plot": 'ggplot' object. A faceted line plot for the metric.
- "table": data frame. A summary table for the metric.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Emails: email_dist(), email_fizz(), email_rank(), email_summary(), email_trend()

Examples

```
# Return a line plot
email_line(pq_data, hrvar = "LevelDesignation")
```

```
# Return summary table
email_line(pq_data, hrvar = "LevelDesignation", return = "table")
```

`email_rank` *Email Hours Ranking*

Description

This function scans a standard query output for groups with high levels of 'Weekly Email Collaboration'. Returns a plot by default, with an option to return a table with a all of groups (across multiple HR attributes) ranked by hours of digital collaboration.

Usage

```
email_rank(
  data,
  hrvar = extract_hr(data),
  mingroup = 5,
  mode = "simple",
  plot_mode = 1,
  return = "plot"
)
```

Arguments

`data`	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns `PersonId` and `MetricDate` present. If your data is already aggregated (e.g. one row per group), use the equivalent `*_asis()` variant of this function instead.
`hrvar`	String containing the name of the HR Variable by which to split metrics. Defaults to `"Organization"`. To run the analysis on the total instead of splitting by an HR attribute, supply `NULL` (without quotes).
`mingroup`	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
`mode`	String to specify calculation mode. Must be either: • `"simple"` • `"combine"`
`plot_mode`	Numeric vector to determine which plot mode to return. Must be either `1` or `2`, and is only used when `return = "plot"`. • `1`: Top and bottom five groups across the data population are highlighted • `2`: Top and bottom groups *per* organizational attribute are highlighted
`return`	String specifying what to return. This must be one of the following strings:

- "plot" (default)
- "table"

See Value for more information.

Details

Uses the metric Email_hours. See create_rank() for applying the same analysis to a different metric.

Value

A different output is returned depending on the value passed to the return argument:

- "plot": 'ggplot' object. A bubble plot where the x-axis represents the metric, the y-axis represents the HR attributes, and the size of the bubbles represent the size of the organizations. Note that there is no plot output if mode is set to "combine".
- "table": data frame. A summary table for the metric.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Emails: email_dist(), email_fizz(), email_line(), email_summary(), email_trend()

Examples

```
# Return rank table
email_rank(
  data = pq_data,
  return = "table"
)

# Return plot
email_rank(
  data = pq_data,
  return = "plot"
)
```

`email_summary` *Email Summary*

Description

Provides an overview analysis of weekly email hours. Returns a bar plot showing average weekly email hours by default. Additional options available to return a summary table.

Usage

```
email_summary(data, hrvar = "Organization", mingroup = 5, return = "plot")

email_sum(data, hrvar = "Organization", mingroup = 5, return = "plot")
```

Arguments

`data`	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns `PersonId` and `MetricDate` present. If your data is already aggregated (e.g. one row per group), use the equivalent `*_asis()` variant of this function instead.
`hrvar`	String containing the name of the HR Variable by which to split metrics. Defaults to `"Organization"`. To run the analysis on the total instead of splitting by an HR attribute, supply `NULL` (without quotes).
`mingroup`	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
`return`	String specifying what to return. This must be one of the following strings: • `"plot"` • `"table"` See `Value` for more information.

Value

A different output is returned depending on the value passed to the `return` argument:

- `"plot"`: 'ggplot' object. A bar plot for the metric.
- `"table"`: data frame. A summary table for the metric.

See Also

Other Visualization: `afterhours_dist()`, `afterhours_fizz()`, `afterhours_line()`, `afterhours_rank()`, `afterhours_summary()`, `afterhours_trend()`, `collaboration_area()`, `collaboration_dist()`, `collaboration_fizz()`, `collaboration_line()`, `collaboration_rank()`, `collaboration_sum()`, `collaboration_trend()`, `create_bar()`, `create_bar_asis()`, `create_boxplot()`, `create_bubble()`, `create_dist()`, `create_fizz()`, `create_inc()`, `create_line()`, `create_line_asis()`, `create_period_scatter()`, `create_radar()`, `create_rank()`, `create_rogers()`, `create_sankey()`, `create_scatter()`,

create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Emails: email_dist(), email_fizz(), email_line(), email_rank(), email_trend()

Examples

```
# Return a ggplot bar chart
email_summary(pq_data, hrvar = "LevelDesignation")

# Return a summary table
email_summary(pq_data, hrvar = "LevelDesignation", return = "table")
```

`email_trend` *Email Hours Time Trend*

Description

Provides a week by week view of email time. By default returns a week by week heatmap, highlighting the points in time with most activity. Additional options available to return a summary table.

Usage

```
email_trend(data, hrvar = "Organization", mingroup = 5, return = "plot")
```

Arguments

`data`	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns `PersonId` and `MetricDate` present. If your data is already aggregated (e.g. one row per group), use the equivalent `*_asis()` variant of this function instead.
`hrvar`	String containing the name of the HR Variable by which to split metrics. Defaults to `"Organization"`. To run the analysis on the total instead of splitting by an HR attribute, supply `NULL` (without quotes).
`mingroup`	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
`return`	Character vector specifying what to return, defaults to `"plot"`. Valid inputs are "plot" and "table".

Details

Uses the metric `Email_hours`.

Value

Returns a 'ggplot' object by default, where 'plot' is passed in `return`. When 'table' is passed, a summary table is returned as a data frame.

See Also

Other Visualization: `afterhours_dist()`, `afterhours_fizz()`, `afterhours_line()`, `afterhours_rank()`, `afterhours_summary()`, `afterhours_trend()`, `collaboration_area()`, `collaboration_dist()`, `collaboration_fizz()`, `collaboration_line()`, `collaboration_rank()`, `collaboration_sum()`, `collaboration_trend()`, `create_bar()`, `create_bar_asis()`, `create_boxplot()`, `create_bubble()`, `create_dist()`, `create_fizz()`, `create_inc()`, `create_line()`, `create_line_asis()`, `create_period_scatter()`, `create_radar()`, `create_rank()`, `create_rogers()`, `create_sankey()`, `create_scatter()`, `create_stacked()`, `create_survival()`, `create_tracking()`, `create_trend()`, `email_dist()`, `email_fizz()`, `email_line()`, `email_rank()`, `email_summary()`, `external_dist()`, `external_fizz()`, `external_line()`, `external_rank()`, `external_sum()`, `hr_trend()`, `hrvar_count()`, `hrvar_trend()`, `keymetrics_scan()`, `meeting_dist()`, `meeting_fizz()`, `meeting_line()`, `meeting_rank()`, `meeting_summary()`, `meeting_trend()`, `one2one_dist()`, `one2one_fizz()`, `one2one_freq()`, `one2one_line()`, `one2one_rank()`, `one2one_sum()`, `one2one_trend()`

Other Emails: `email_dist()`, `email_fizz()`, `email_line()`, `email_rank()`, `email_summary()`

Examples

```
# Run plot
email_trend(pq_data)

# Run table
email_trend(pq_data, hrvar = "LevelDesignation", return = "table")
```

---

export *Export 'vivainsights' outputs to CSV, clipboard, or save as images*

Description

A general use function to export 'vivainsights' outputs to CSV, clipboard, or save as images. By default, `export()` copies a data frame to the clipboard. If the input is a 'ggplot' object, the default behaviour is to export a PNG.

Usage

```
export(
  x,
  method = "clipboard",
  path = "insights export",
  timestamp = TRUE,
  width = 12,
  height = 9
)
```

Arguments

x	Data frame or 'ggplot' object to be passed through.
method	Character string specifying the method of export. Valid inputs include: • "clipboard" (default if input is data frame) • "csv" • "png" (default if input is 'ggplot' object) • "svg" • "jpeg" • "pdf"
path	If exporting a file, enter the path and the desired file name, *excluding the file extension*. For example, "Analysis/SQ Overview".
timestamp	Logical vector specifying whether to include a timestamp in the file name. Defaults to TRUE.
width	Width of the plot
height	Height of the plot

Value

A different output is returned depending on the value passed to the method argument:

- "clipboard": no return - data frame is saved to clipboard.
- "csv": CSV file containing data frame is saved to specified path.
- "png": PNG file containing 'ggplot' object is saved to specified path.
- "svg": SVG file containing 'ggplot' object is saved to specified path.
- "jpeg": JPEG file containing 'ggplot' object is saved to specified path.
- "pdf": PDF file containing 'ggplot' object is saved to specified path.

Author(s)

Martin Chan martin.chan@microsoft.com

See Also

Other Import and Export: copy_df(), create_dt(), import_query(), prep_query()

`external_dist` *Distribution of External Collaboration Hours as a 100% stacked bar*

Description

Analyze the distribution of External Collaboration Hours. Returns a stacked bar plot by default. Additional options available to return a table with distribution elements.

Usage

```
external_dist(
  data,
  hrvar = "Organization",
  mingroup = 5,
  return = "plot",
  cut = c(5, 10, 15)
)
```

Arguments

`data`	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns `PersonId` and `MetricDate` present. If your data is already aggregated (e.g. one row per group), use the equivalent `*_asis()` variant of this function instead.
`hrvar`	String containing the name of the HR Variable by which to split metrics. Defaults to `"Organization"`. To run the analysis on the total instead of splitting by an HR attribute, supply `NULL` (without quotes).
`mingroup`	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
`return`	String specifying what to return. This must be one of the following strings: • `"plot"` • `"table"` See `Value` for more information.
`cut`	A numeric vector of length three to specify the breaks for the distribution, e.g. c(10, 15, 20)

Details

Uses the metric `External_collaboration_hours`. See `create_dist()` for applying the same analysis to a different metric.

Value

A different output is returned depending on the value passed to the `return` argument:

- `"plot"`: 'ggplot' object. A stacked bar plot for the metric.
- `"table"`: data frame. A summary table for the metric.

See Also

Other Visualization: `afterhours_dist()`, `afterhours_fizz()`, `afterhours_line()`, `afterhours_rank()`, `afterhours_summary()`, `afterhours_trend()`, `collaboration_area()`, `collaboration_dist()`, `collaboration_fizz()`, `collaboration_line()`, `collaboration_rank()`, `collaboration_sum()`, `collaboration_trend()`, `create_bar()`, `create_bar_asis()`, `create_boxplot()`, `create_bubble()`, `create_dist()`, `create_fizz()`, `create_inc()`, `create_line()`, `create_line_asis()`, `create_period_scatter()`, `create_radar()`, `create_rank()`, `create_rogers()`, `create_sankey()`, `create_scatter()`, `create_stacked()`, `create_survival()`, `create_tracking()`, `create_trend()`, `email_dist()`, `email_fizz()`, `email_line()`, `email_rank()`, `email_summary()`, `email_trend()`, `external_fizz()`, `external_line()`, `external_rank()`, `external_sum()`, `hr_trend()`, `hrvar_count()`, `hrvar_trend()`, `keymetrics_scan()`, `meeting_dist()`, `meeting_fizz()`, `meeting_line()`, `meeting_rank()`, `meeting_summary()`, `meeting_trend()`, `one2one_dist()`, `one2one_fizz()`, `one2one_freq()`, `one2one_line()`, `one2one_rank()`, `one2one_sum()`, `one2one_trend()`

Other External Collaboration: `external_fizz()`, `external_line()`, `external_sum()`

Examples

```
# Return plot
external_dist(pq_data, hrvar = "Organization")

# Return summary table
external_dist(pq_data, hrvar = "Organization", return = "table")

# Return result with a custom specified breaks
external_dist(pq_data, hrvar = "LevelDesignation", cut = c(2, 4, 6))
```

`external_fizz` *Distribution of External Collaboration Hours (Fizzy Drink plot)*

Description

Analyze weekly External Collaboration hours distribution, and returns a 'fizzy' scatter plot by default. Additional options available to return a table with distribution elements.

Usage

```
external_fizz(data, hrvar = "Organization", mingroup = 5, return = "plot")
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present. If your data is already aggregated (e.g. one row per group), use the equivalent *_asis() variant of this function instead.
hrvar	String containing the name of the HR Variable by which to split metrics. Defaults to "Organization". To run the analysis on the total instead of splitting by an HR attribute, supply NULL (without quotes).
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
return	String specifying what to return. This must be one of the following strings: • "plot" • "table" See Value for more information.

Details

Uses the metric Collaboration_hours_external. See create_fizz() for applying the same analysis to a different metric.

Value

A different output is returned depending on the value passed to the return argument:

- "plot": 'ggplot' object. A jittered scatter plot for the metric.
- "table": data frame. A summary table for the metric.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other External Collaboration: external_dist(), external_line(), external_sum()

Examples

```
# Return plot
external_fizz(pq_data, hrvar = "LevelDesignation", return = "plot")

# Return summary table
external_fizz(pq_data, hrvar = "Organization", return = "table")
```

`external_line` *External Collaboration Hours Time Trend - Line Chart*

Description

Provides a week by week view of External collaboration time, visualized as line chart. By default returns a separate panel per value in the HR attribute. Additional options available to return a summary table.

Usage

```
external_line(
  data,
  hrvar = "Organization",
  mingroup = 5,
  return = "plot",
  label = FALSE
)
```

Arguments

`data`	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns `PersonId` and `MetricDate` present. If your data is already aggregated (e.g. one row per group), use the equivalent `*_asis()` variant of this function instead.
`hrvar`	String containing the name of the HR Variable by which to split metrics. Defaults to `"Organization"`. To run the analysis on the total instead of splitting by an HR attribute, supply `NULL` (without quotes).
`mingroup`	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
`return`	String specifying what to return. This must be one of the following strings: • `"plot"` • `"table"` See `Value` for more information.
`label`	Logical value to determine whether to show data point labels on the plot. If `TRUE`, both `geom_point()` and `geom_text()` are added to display data labels rounded to 1 decimal place above each data point. Defaults to `FALSE`.

Details

Uses the metric `Collaboration_hours_external`.

Value

A different output is returned depending on the value passed to the `return` argument:

- `"plot"`: 'ggplot' object. A faceted line plot for the metric.
- `"table"`: data frame. A summary table for the metric.

See Also

`create_line()` for applying the same analysis to a different metric.

Other Visualization: `afterhours_dist()`, `afterhours_fizz()`, `afterhours_line()`, `afterhours_rank()`, `afterhours_summary()`, `afterhours_trend()`, `collaboration_area()`, `collaboration_dist()`, `collaboration_fizz()`, `collaboration_line()`, `collaboration_rank()`, `collaboration_sum()`, `collaboration_trend()`, `create_bar()`, `create_bar_asis()`, `create_boxplot()`, `create_bubble()`, `create_dist()`, `create_fizz()`, `create_inc()`, `create_line()`, `create_line_asis()`, `create_period_scatter()`, `create_radar()`, `create_rank()`, `create_rogers()`, `create_sankey()`, `create_scatter()`, `create_stacked()`, `create_survival()`, `create_tracking()`, `create_trend()`, `email_dist()`, `email_fizz()`, `email_line()`, `email_rank()`, `email_summary()`, `email_trend()`, `external_dist()`, `external_fizz()`, `external_rank()`, `external_sum()`, `hr_trend()`, `hrvar_count()`, `hrvar_trend()`, `keymetrics_scan()`, `meeting_dist()`, `meeting_fizz()`, `meeting_line()`, `meeting_rank()`, `meeting_summary()`, `meeting_trend()`, `one2one_dist()`, `one2one_fizz()`, `one2one_freq()`, `one2one_line()`, `one2one_rank()`, `one2one_sum()`, `one2one_trend()`

Other External Collaboration: `external_dist()`, `external_fizz()`, `external_sum()`

Examples

```
# Return a line plot
external_line(pq_data, hrvar = "LevelDesignation")

# Return summary table
external_line(pq_data, hrvar = "LevelDesignation", return = "table")
```

---

external_rank *Rank groups with high External Collaboration Hours*

Description

This function scans a Standard Person Query for groups with high levels of External Collaboration. Returns a plot by default, with an option to return a table with all groups (across multiple HR attributes) ranked by hours of External Collaboration.

Usage

```
external_rank(
  data,
  hrvar = extract_hr(data),
  mingroup = 5,
  mode = "simple",
  plot_mode = 1,
  return = "plot"
)
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present. If your data is already aggregated (e.g. one row per group), use the equivalent *_asis() variant of this function instead.
hrvar	String containing the name of the HR Variable by which to split metrics. Defaults to "Organization". To run the analysis on the total instead of splitting by an HR attribute, supply NULL (without quotes).
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
mode	String to specify calculation mode. Must be either: • "simple" • "combine"
plot_mode	Numeric vector to determine which plot mode to return. Must be either 1 or 2, and is only used when return = "plot". • 1: Top and bottom five groups across the data population are highlighted • 2: Top and bottom groups *per* organizational attribute are highlighted
return	String specifying what to return. This must be one of the following strings: • "plot" (default) • "table" See Value for more information.

Details

Uses the metric Collaboration_hours_external. See create_rank() for applying the same analysis to a different metric.

Value

When 'table' is passed in return, a summary table is returned as a data frame.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other After-hours Collaboration: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend()

Examples

```
# Return rank table
external_rank(data = pq_data, return = "table")

# Return plot
external_rank(data = pq_data, return = "plot")
```

`external_sum` *External Collaboration Summary*

Description

Provides an overview analysis of 'External Collaboration'. Returns a stacked bar plot of internal and external collaboration. Additional options available to return a summary table.

Usage

```
external_sum(
  data,
  hrvar = "Organization",
  mingroup = 5,
  stack_colours = c("#1d327e", "#1d7e6a"),
  return = "plot"
)

external_summary(
  data,
  hrvar = "Organization",
  mingroup = 5,
```

```
  stack_colours = c("#1d327e", "#1d7e6a"),
  return = "plot"
)
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present. If your data is already aggregated (e.g. one row per group), use the equivalent *_asis() variant of this function instead.
hrvar	String containing the name of the HR Variable by which to split metrics. Defaults to "Organization". To run the analysis on the total instead of splitting by an HR attribute, supply NULL (without quotes).
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
stack_colours	A character vector to specify the colour codes for the stacked bar charts.
return	Character vector specifying what to return, defaults to "plot". Valid inputs are "plot" and "table".

Value

Returns a 'ggplot' object by default, where 'plot' is passed in return. When 'table' is passed, a summary table is returned as a data frame.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other External Collaboration: external_dist(), external_fizz(), external_line()

Examples

```
# Return a plot
external_sum(pq_data, hrvar = "LevelDesignation")

# Return summary table
external_sum(pq_data, hrvar = "LevelDesignation", return = "table")
```

extract_date_range *Extract date period*

Description

Return a data frame with the start and end date of the query data by default. There are options to return a descriptive string, which is used in the caption of plots in this package.

Usage

```
extract_date_range(data, return = "table")
```

Arguments

data	Data frame containing a query to pass through. The data frame must contain a Date column. Accepts a Person query or a Meeting query.
return	String specifying what output to return. Returns a table by default ("table"), but allows returning a descriptive string ("text").

Value

A different output is returned depending on the value passed to the return argument:

- "table": data frame. A summary table containing the start and end date for the dataset.
- "text": string. Contains a descriptive string on the start and end date for the dataset.

See Also

Other Support: any_idate(), camel_clean(), check_inputs(), cut_hour(), extract_hr(), heat_colours(), is_date_format(), maxmin(), pairwise_count(), read_preamble(), rgb2hex(), totals_bind(), totals_col(), tstamp(), us_to_space(), wrap()

extract_hr *Extract HR attribute variables*

Description

This function uses a combination of variable class, number of unique values, and regular expression matching to extract HR / organisational attributes from a data frame.

Usage

```
extract_hr(data, max_unique = 50, exclude_constants = TRUE, return = "names")
```

Arguments

`data`	A data frame to be passed through.
`max_unique`	A numeric value representing the maximum number of unique values to accept for an HR attribute. Defaults to 50. Any column with `max_unique` or more unique values is excluded, and a message is displayed listing those columns and their unique-value counts.
`exclude_constants`	Logical value to specify whether single-value HR attributes are to be excluded. Defaults to `TRUE`.
`return`	String specifying what to return. This must be one of the following strings: • `"names"` • `"vars"` See `Value` for more information.

Value

A different output is returned depending on the value passed to the `return` argument:

- `"names"`: character vector identifying all the names of HR variables present in the data.
- `"vars"`: data frame containing all the columns of HR variables present in the data.

See Also

Other Support: `any_idate()`, `camel_clean()`, `check_inputs()`, `cut_hour()`, `extract_date_range()`, `heat_colours()`, `is_date_format()`, `maxmin()`, `pairwise_count()`, `read_preamble()`, `rgb2hex()`, `totals_bind()`, `totals_col()`, `tstamp()`, `us_to_space()`, `wrap()`

Other Data Validation: `check_query()`, `flag_ch_ratio()`, `flag_em_ratio()`, `flag_extreme()`, `flag_outlooktime()`, `hr_trend()`, `hrvar_count()`, `hrvar_count_all()`, `hrvar_trend()`, `identify_churn()`, `identify_holidayweeks()`, `identify_inactiveweeks()`, `identify_nkw()`, `identify_outlier()`, `identify_privacythreshold()`, `identify_shifts()`, `identify_tenure()`, `track_HR_change()`, `validation_report()`

Examples

```
pq_data %>% extract_hr(return = "names")

pq_data %>% extract_hr(return = "vars")
```

`flag_ch_ratio` *Flag unusual high collaboration hours to after-hours collaboration hours ratio*

Description

This function flags persons who have an unusual ratio of collaboration hours to after-hours collaboration hours. Returns a character string by default.

Usage

```
flag_ch_ratio(data, threshold = c(1, 30), return = "message")
```

Arguments

`data`	A data frame containing a Person Query.
`threshold`	Numeric value specifying the threshold for flagging. Defaults to 30.
`return`	String to specify what to return. Options include: • `"message"` • `"text"` • `"data"`

Value

A different output is returned depending on the value passed to the `return` argument:

- `"message"`: message in the console containing diagnostic summary
- `"text"`: string containing diagnostic summary
- `"data"`: data frame. Person-level data with flags on unusually high or low ratios

Metrics used

The metric `Collaboration_hours` is used in the calculations. Please ensure that your query contains a metric with the exact same name.

See Also

Other Data Validation: `check_query()`, `extract_hr()`, `flag_em_ratio()`, `flag_extreme()`, `flag_outlooktime()`, `hr_trend()`, `hrvar_count()`, `hrvar_count_all()`, `hrvar_trend()`, `identify_churn()`, `identify_holidayweeks()`, `identify_inactiveweeks()`, `identify_nkw()`, `identify_outlier()`, `identify_privacythreshold()`, `identify_shifts()`, `identify_tenure()`, `track_HR_change()`, `validation_report()`

Examples

```
flag_ch_ratio(pq_data)


data.frame(PersonId = c("Alice", "Bob"),
           Collaboration_hours = c(30, 0.5),
           After_hours_collaboration_hours = c(0.5, 30)) %>%
  flag_ch_ratio()
```

`flag_em_ratio` *Flag Persons with unusually high Email Hours to Emails Sent ratio*

Description

This function flags persons who have an unusual ratio of email hours to emails sent. If the ratio between Email Hours and Emails Sent is greater than the threshold, then observations tied to a `PersonId` is flagged as unusual.

Usage

```
flag_em_ratio(data, threshold = 1, return = "text")
```

Arguments

`data`	A data frame containing a Person Query.
`threshold`	Numeric value specifying the threshold for flagging. Defaults to 1.
`return`	String specifying what to return. This must be one of the following strings: • `"text"` • `"data"` See `Value` for more information.

Value

A different output is returned depending on the value passed to the `return` argument:

- `"text"`: string. A diagnostic message.
- `"data"`: data frame. Person-level data with those flagged with unusual ratios.

See Also

Other Data Validation: `check_query()`, `extract_hr()`, `flag_ch_ratio()`, `flag_extreme()`, `flag_outlooktime()`, `hr_trend()`, `hrvar_count()`, `hrvar_count_all()`, `hrvar_trend()`, `identify_churn()`, `identify_holidayweeks()`, `identify_inactiveweeks()`, `identify_nkw()`, `identify_outlier()`, `identify_privacythreshold()`, `identify_shifts()`, `identify_tenure()`, `track_HR_change()`, `validation_report()`

Examples

```
flag_em_ratio(pq_data)
```

`flag_extreme` *Warn for extreme values by checking against a threshold*

Description

This is used as part of data validation to check if there are extreme values in the dataset.

Usage

```
flag_extreme(
  data,
  metric,
  person = TRUE,
  threshold,
  mode = "above",
  return = "message"
)
```

Arguments

`data`	A Standard Person Query dataset in the form of a data frame.
`metric`	A character string specifying the metric to test.
`person`	A logical value to specify whether to calculate person-averages. Defaults to `TRUE` (person-averages calculated).
`threshold`	Numeric value specifying the threshold for flagging.
`mode`	String determining mode to use for identifying extreme values. • `"above"`: checks whether value is great than the threshold (default) • `"equal"`: checks whether value is equal to the threshold • `"below"`: checks whether value is below the threshold
`return`	String specifying what to return. This must be one of the following strings: • `"text"` • `"message"` • `"table"` See `Value` for more information.

Value

A different output is returned depending on the value passed to the `return` argument:

- `"text"`: string. A diagnostic message.
- `"message"`: message on console. A diagnostic message.
- `"table"`: data frame. A person-level table with `PersonId` and the extreme values of the selected metric.

See Also

Other Data Validation: check_query(), extract_hr(), flag_ch_ratio(), flag_em_ratio(), flag_outlooktime(), hr_trend(), hrvar_count(), hrvar_count_all(), hrvar_trend(), identify_churn(), identify_holidayweeks(), identify_inactiveweeks(), identify_nkw(), identify_outlier(), identify_privacythreshold(), identify_shifts(), identify_tenure(), track_HR_change(), validation_report()

Examples

```
# The threshold values are intentionally set low to trigger messages.
flag_extreme(pq_data, "Email_hours", threshold = 15)

# Return a summary table
flag_extreme(pq_data, "Email_hours", threshold = 15, return = "table")

# Person-week level
flag_extreme(pq_data, "Email_hours", person = FALSE, threshold = 15)

# Check for values equal to threshold
flag_extreme(pq_data, "Email_hours", person = TRUE, mode = "equal", threshold = 0)

# Check for values below threshold
flag_extreme(pq_data, "Email_hours", person = TRUE, mode = "below", threshold = 5)
```

---

flag_outlooktime *Flag unusual outlook time settings for work day start and end time*

Description

This function flags unusual outlook calendar settings for start and end time of work day.

Usage

```
flag_outlooktime(data, threshold = c(4, 15), return = "message")
```

Arguments

data	A data frame containing a Person Query.
threshold	A numeric vector of length two, specifying the hour threshold for flagging. Defaults to c(4, 15).
return	String specifying what to return. This must be one of the following strings: • "text" (default) • "message" • "data"

Value

A different output is returned depending on the value passed to the `return` argument:

- `"text"`: string. A diagnostic message.
- `"message"`: message on console. A diagnostic message.
- `"data"`: data frame. Data where flag is present.

See `Value` for more information.

See Also

Other Data Validation: `check_query()`, `extract_hr()`, `flag_ch_ratio()`, `flag_em_ratio()`, `flag_extreme()`, `hr_trend()`, `hrvar_count()`, `hrvar_count_all()`, `hrvar_trend()`, `identify_churn()`, `identify_holidayweeks()`, `identify_inactiveweeks()`, `identify_nkw()`, `identify_outlier()`, `identify_privacythreshold()`, `identify_shifts()`, `identify_tenure()`, `track_HR_change()`, `validation_report()`

Examples

```
# Demo with `pq_data` example where Outlook Start and End times are imputed
spq_df <- pq_data

spq_df$WorkingStartTimeSetInOutlook <- "6:30"

spq_df$WorkingEndTimeSetInOutlook <- "23:30"

# Return a message
flag_outlooktime(spq_df, threshold = c(5, 13))

# Return data
flag_outlooktime(spq_df, threshold = c(5, 13), return = "data")
```

---

g2g_data *Sample Group-to-Group dataset*

Description

A demo dataset representing a Group-to-Group Query. The grouping organizational attribute used here is `Organization`, where the variable have been prefixed with `PrimaryCollaborator_` and `SecondaryCollaborator_` to represent the direction of collaboration.

Usage

```
g2g_data
```

Format

A data frame with 150 rows and 11 variables:

PrimaryCollaborator_Organization

PrimaryCollaborator_GroupSize

SecondaryCollaborator_Organization

SecondaryCollaborator_GroupSize

MetricDate

Percent_Group_collaboration_time_invested

Group_collaboration_time_invested

Group_email_sent_count

Group_email_time_invested

Group_meeting_count

Group_meeting_time_invested ...

Value

data frame.

Source

https://analysis.insights.cloud.microsoft/analyst/analysis/

See Also

Other Data: `mt_data`, `p2p_data`, `p2p_data_sim()`, `pq_data`

Other Network: `network_g2g()`, `network_p2p()`, `network_summary()`, `p2p_data`, `p2p_data_sim()`

`generate_report` *Generate HTML report with list inputs*

Description

This is a support function using a list-pmap workflow to create a HTML document, using RMarkdown as the engine.

Usage

```
generate_report(
  title = "My minimal HTML generator",
  filename = "minimal_html",
  outputs = output_list,
  titles,
  subheaders,
  echos,
  levels,
  theme = "united",
  preamble = ""
)
```

Arguments

`title`	Character string to specify the title of the chunk.
`filename`	File name to be used in the exported HTML.
`outputs`	A list of outputs to be added to the HTML report. Note that `outputs`, `titles`, `echos`, and `levels` must have the same length
`titles`	A list/vector of character strings to specify the title of the chunks.
`subheaders`	A list/vector of character strings to specify the subheaders for each chunk.
`echos`	A list/vector of logical values to specify whether to display code.
`levels`	A list/vector of numeric value to specify the header level of the chunk.
`theme`	Character vector to specify theme to be used for the report. E.g. `"united"`, `"default"`.
`preamble`	A preamble to appear at the beginning of the report, passed as a text string.

Value

An HTML report with the same file name as specified in the arguments is generated in the working directory. No outputs are directly returned by the function.

Creating a custom report

Below is an example on how to set up a custom report.

The first step is to define the content that will go into a report and assign the outputs to a list.

```
# Step 1: Define Content
output_list <-
  list(pq_data %>% workloads_summary(return = "plot"),
       pq_data %>% workloads_summary(return = "table")) %>%
  purrr::map_if(is.data.frame, create_dt)
```

The next step is to add a list of titles for each of the objects on the list:

```
# Step 2: Add Corresponding Titles
title_list <- c("Workloads Summary - Plot", "Workloads Summary - Table")
n_title <- length(title_list)
```

The final step is to run `generate_report()`. This can all be wrapped within a function such that the function can be used to generate a HTML report.

```
# Step 3: Generate Report
generate_report(title = "My First Report",
                filename = "My First Report",
                outputs = output_list,
                titles = title_list,
                subheaders = rep("", n_title),
                echos = rep(FALSE, n_title
```

Author(s)

Martin Chan martin.chan@microsoft.com

See Also

Other Reports: `IV_report()`, `meeting_tm_report()`, `read_preamble()`, `validation_report()`

`generate_report2` *Generate HTML report based on existing RMarkdown documents*

Description

This is a support function that accepts parameters and creates a HTML document based on an RMarkdown template. This is an alternative to `generate_report()` which instead creates an RMarkdown document from scratch using individual code chunks.

Usage

```
generate_report2(
 output_format = rmarkdown::html_document(toc = TRUE, toc_depth = 6, theme = "cosmo"),
  output_file = "report.html",
  output_dir = getwd(),
  report_title = "Report",
  rmd_dir = system.file("rmd_template/minimal.rmd", package = "vivainsights"),
  ...
)
```

Arguments

output_format	output format in rmarkdown::render(). Default is rmarkdown::html_document(toc = TRUE, toc_depth = 6, theme = "cosmo").
output_file	output file name in rmarkdown::render(). Default is "report.html".
output_dir	output directory for report in rmarkdown::render(). Default is user's current directory.
report_title	report title. Default is "Report".
rmd_dir	string specifying the path to the directory containing the RMarkdown template files.
...	other arguments to be passed to params. For instance, pass hrvar if the RMarkdown document requires a 'hrvar' parameter.

Note

The implementation of this function was inspired by the 'DataExplorer' package by boxuancui, with credits due to the original author.

---

heat_colours *Generate a vector of* n *contiguous colours, as a red-yellow-green palette.*

Description

Takes a numeric value n and returns a character vector of colour HEX codes corresponding to the heat map palette.

Usage

```
heat_colours(n, alpha, rev = FALSE)

heat_colors(n, alpha, rev = FALSE)
```

Arguments

n	the number of colors (>= 1) to be in the palette.
alpha	an alpha-transparency level in the range of 0 to 1 (0 means transparent and 1 means opaque)
rev	logical indicating whether the ordering of the colors should be reversed.

Value

A character vector containing the HEX codes and the same length as n is returned.

See Also

Other Support: any_idate(), camel_clean(), check_inputs(), cut_hour(), extract_date_range(), extract_hr(), is_date_format(), maxmin(), pairwise_count(), read_preamble(), rgb2hex(), totals_bind(), totals_col(), tstamp(), us_to_space(), wrap()

Examples

```
barplot(rep(10, 50), col = heat_colours(n = 50), border = NA)

barplot(rep(10, 50), col = heat_colours(n = 50, alpha = 0.5, rev = TRUE),
border = NA)
```

---

hrvar_count *Create a count of distinct people in a specified HR variable*

Description

This function enables you to create a count of the distinct people by the specified HR attribute.The default behaviour is to return a bar chart as typically seen in 'Analysis Scope'.

Usage

```
hrvar_count(data, hrvar = "Organization", return = "plot")

analysis_scope(data, hrvar = "Organization", return = "plot")
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present.
hrvar	HR Variable by which to split metrics, defaults to "Organization" but accepts any character vector, e.g. "LevelDesignation". If a vector with more than one value is provided, the HR attributes are automatically concatenated.
return	String specifying what to return. This must be one of the following strings: • "plot" • "table" See Value for more information.

Value

A different output is returned depending on the value passed to the return argument:

- "plot": 'ggplot' object containing a bar plot.
- "table": data frame containing a count table.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Data Validation: check_query(), extract_hr(), flag_ch_ratio(), flag_em_ratio(), flag_extreme(), flag_outlooktime(), hr_trend(), hrvar_count_all(), hrvar_trend(), identify_churn(), identify_holidayweeks(), identify_inactiveweeks(), identify_nkw(), identify_outlier(), identify_privacythreshold(), identify_shifts(), identify_tenure(), track_HR_change(), validation_report()

Examples

```
# Return a bar plot
hrvar_count(pq_data, hrvar = "LevelDesignation")

# Return a summary table
hrvar_count(pq_data, hrvar = "LevelDesignation", return = "table")
```

---

hrvar_count_all *Create count of distinct fields and percentage of employees with missing values for all HR variables*

Description

[Experimental]

This function enables you to create a summary table to validate organizational data. This table will provide a summary of the data found in the Viva Insights *Data sources* page. This function will return a summary table with the count of distinct fields per HR attribute and the percentage of employees with missing values for that attribute. See `hrvar_count()` function for more detail on the specific HR attribute of interest.

Usage

```
hrvar_count_all(
  data,
  n_var = 50,
```

```
  return = "message",
  threshold = 100,
  maxna = 20
)
```

Arguments

data	A Standard Person Query dataset in the form of a data frame.
n_var	number of HR variables to include in report as rows. Default is set to 50 HR variables.
return	String to specify what to return
threshold	The max number of unique values allowed for any attribute. Default is 100.
maxna	The max percentage of NAs allowable for any column. Default is 20.

Value

Returns an error message by default, where 'text' is passed in return.

- 'table': data frame. A summary table listing the number of distinct fields and percentage of missing values for the specified number of HR attributes will be returned.
- 'message': outputs a message indicating which values are beyond the specified thresholds.

See Also

Other Data Validation: check_query(), extract_hr(), flag_ch_ratio(), flag_em_ratio(), flag_extreme(), flag_outlooktime(), hr_trend(), hrvar_count(), hrvar_trend(), identify_churn(), identify_holidayweeks(), identify_inactiveweeks(), identify_nkw(), identify_outlier(), identify_privacythreshold(), identify_shifts(), identify_tenure(), track_HR_change(), validation_report()

Examples

```
# Return a summary table of all HR attributes
hrvar_count_all(pq_data, return = "table")
```

---

hrvar_trend *Track count of distinct people over time in a specified HR variable*

Description

This function provides a week by week view of the count of the distinct people by the specified HR attribute.The default behaviour is to return a week by week heatmap bar plot.

Usage

```
hrvar_trend(data, hrvar = "Organization", return = "plot")
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present.
hrvar	HR Variable by which to split metrics, defaults to "Organization" but accepts any character vector, e.g. "LevelDesignation". If a vector with more than one value is provided, the HR attributes are automatically concatenated.
return	String specifying what to return. This must be one of the following strings: • "plot" • "table" See Value for more information.

Value

A different output is returned depending on the value passed to the return argument:

- "plot": 'ggplot' object containing a bar plot.
- "table": data frame containing a count table.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Data Validation: check_query(), extract_hr(), flag_ch_ratio(), flag_em_ratio(), flag_extreme(), flag_outlooktime(), hr_trend(), hrvar_count(), hrvar_count_all(), identify_churn(), identify_holidayweeks(), identify_inactiveweeks(), identify_nkw(), identify_outlier(), identify_privacythreshold(), identify_shifts(), identify_tenure(), track_HR_change(), validation_report()

Examples

```
# Return a bar plot
hrvar_trend(pq_data, hrvar = "LevelDesignation")

# Return a summary table
hrvar_trend(pq_data, hrvar = "LevelDesignation", return = "table")
```

hr_trend *Employee count over time*

Description

Returns a line chart showing the change in employee count over time. Part of a data validation process to check for unusual license growth / declines over time.

Usage

```
hr_trend(data, return = "plot")
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present.
return	String specifying what to return. This must be one of the following strings: • "plot" • "table" See Value for more information.

Value

A different output is returned depending on the value passed to the return argument:

- "plot": ggplot object. A line plot showing employee count over time.
- "table": data frame containing a summary table.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Data Validation: check_query(), extract_hr(), flag_ch_ratio(), flag_em_ratio(), flag_extreme(), flag_outlooktime(), hrvar_count(), hrvar_count_all(), hrvar_trend(), identify_churn(), identify_holidayweeks(), identify_inactiveweeks(), identify_nkw(), identify_outlier(), identify_privacythreshold(), identify_shifts(), identify_tenure(), track_HR_change(), validation_report()

Examples

```
# Return plot
hr_trend(pq_data)

# Return summary table
hr_trend(pq_data, return = "table")
```

`identify_churn` *Identify employees who have churned from the dataset*

Description

This function identifies and counts the number of employees who have churned from the dataset by measuring whether an employee who is present in the first `n` (n1) weeks of the data is present in the last `n` (n2) weeks of the data.

Usage

```
identify_churn(data, n1 = 6, n2 = 6, return = "message", flip = FALSE)
```

Arguments

`data`	A Person Query as a data frame. Must contain a `PersonId`.
`n1`	A numeric value specifying the number of weeks at the beginning of the period that defines the measured employee set. Defaults to 6.
`n2`	A numeric value specifying the number of weeks at the end of the period to calculate whether employees have churned from the data. Defaults to 6.
`return`	String specifying what to return. This must be one of the following strings: • `"message"` (default) • `"text"` • `"data"` See `Value` for more information.
`flip`	Logical, defaults to FALSE. This determines whether to reverse the logic of identifying the non-overlapping set. If set to `TRUE`, this effectively identifies new-joiners, or those who were not present in the first n weeks of the data but were present in the final n weeks.

Details

An additional use case of this function is the ability to identify "new-joiners" by using the argument `flip`.

If an employee is present in the first `n` weeks of the data but not present in the last `n` weeks of the data, the function considers the employee as churned. As the measurement period is defined by the

number of weeks from the start and the end of the passed data frame, you may consider filtering the dates accordingly before running this function.

Another assumption that is in place is that any employee whose `PersonId` is not available in the data has churned. Note that there may be other reasons why an employee's `PersonId` may not be present, e.g. maternity/paternity leave, Viva Insights license has been removed, shift to a low-collaboration role (to the extent that he/she becomes inactive).

Value

A different output is returned depending on the value passed to the `return` argument:

- `"message"`: Message on console. A diagnostic message.
- `"text"`: String. A diagnostic message.
- `"data"`: Character vector containing the the `PersonId` of employees who have been identified as churned.

See Also

Other Data Validation: `check_query()`, `extract_hr()`, `flag_ch_ratio()`, `flag_em_ratio()`, `flag_extreme()`, `flag_outlooktime()`, `hr_trend()`, `hrvar_count()`, `hrvar_count_all()`, `hrvar_trend()`, `identify_holidayweeks()`, `identify_inactiveweeks()`, `identify_nkw()`, `identify_outlier()`, `identify_privacythreshold()`, `identify_shifts()`, `identify_tenure()`, `track_HR_change()`, `validation_report()`

Examples

```
pq_data %>% identify_churn(n1 = 3, n2 = 3, return = "message")
```

---

identify_datefreq *Identify date frequency based on a series of dates*

Description

[Experimental]

Takes a vector of dates and identify whether the frequency is 'daily', 'weekly', or 'monthly'. The primary use case for this function is to provide an accurate description of the query type used and for raising errors should a wrong date grouping be used in the data input.

Usage

```
identify_datefreq(x)
```

Arguments

`x`	Vector containing a series of dates.

Details

Date frequency detection works as follows:

- If at least three days of the week are present (e.g., Monday, Wednesday, Thursday) in the series, then the series is classified as 'daily'
- If the total number of months in the series is equal to the length, then the series is classified as 'monthly'
- If the total number of sundays in the series is equal to the length of the series, then the series is classified as 'weekly

Value

String describing the detected date frequency, i.e.:

- `'daily'`
- `'weekly'`
- `'monthly'`

Limitations

One of the assumptions made behind the classification is that weeks are denoted with Sundays, hence the count of sundays to measure the number of weeks. In this case, weeks where a Sunday is missing would result in an 'unable to classify' error.

Another assumption made is that dates are evenly distributed, i.e. that the gap between dates are equal. If dates are unevenly distributed, e.g. only two days of the week are available for a given week, then the algorithm will fail to identify the frequency as 'daily'.

Examples

```
start_date <- as.Date("2022/06/26")
end_date <- as.Date("2022/11/27")

# Daily
day_seq <-
  seq.Date(
    from = start_date,
    to = end_date,
    by = "day"
  )

identify_datefreq(day_seq)

# Weekly
week_seq <-
  seq.Date(
    from = start_date,
    to = end_date,
    by = "week"
  )
```

```
identify_datefreq(week_seq)

# Monthly
month_seq <-
  seq.Date(
    from = start_date,
    to = end_date,
    by = "month"
  )
identify_datefreq(month_seq)
```

`identify_habit` *Identify whether a habitual behaviour exists over a given interval of time*

Description

[Experimental]

Based on the principle of consistency, this function identifies whether a habit exists over a given interval of time. A habit is defined as a behaviour (action taken) that is repeated at least x number of times consistently over n weeks.

Usage

```
identify_habit(
  data,
  metric,
  threshold = 1,
  width,
  max_window,
  hrvar = NULL,
  return = "plot",
  plot_mode = "time",
  fill_col = c("#E5E5E5", "#0078D4")
)
```

Arguments

`data`	Data frame containing Person Query to be analysed. The data frame must have a `PersonId`, `MetricDate` and a column containing a metric for classifying behaviour.
`metric`	Character string specifying the metric to be analysed.
`threshold`	Numeric value specifying the minimum number of times the metric sum up to in order to be a valid count. A 'greater than or equal to' logic is used.
`width`	Integer specifying the number of qualifying counts to consider for a habit. The function assumes a **weekly** interval is used.

max_window	Integer specifying the maximum unit of dates to consider a qualifying window for a habit. If your data is grouped at a weekly level, then max_window = 12 would consider 12 weeks.
hrvar	Character string specifying the HR attribute or organisational variable to group by. Default is NULL.
return	Character string specifying the type of output to be returned. Valid options include: • "data": Returns the data frame with the habit classification. • "plot": Returns a ggplot object of a boxplot, showing the percentage of periods with where habitual behaviour occurred. • "summary": Returns a summary table of the habit analysis.
plot_mode	Character string specifying the type of plot to be returned. Only applicable when return = "plot". Valid options include: • "time": Returns a time series plot with the breakdown of users with habitual behaviour. • "boxplot": Returns a boxplot of the percentage of periods with habitual behaviour.
fill_col	Character vector of length 2 specifying the colours to be used in the plot. Only applicable when return = "plot" and plot_mode = "time".

Details

Each week is considered as a binary variable on whether sufficient action has been taken for that given week (a qualifying count). Sufficiency is determined by the threshold parameter. For instance, if the threshold is set to 2, this means that there must be 2 qualifying actions (e.g. summarise meeting in Copilot) in a week for there to be a qualifying count for the week. One way of determining the parameters would be to consider, *how many counts of* width *should occur within a* max_window *period for it to be considered a habit?*

Examples

```
# Return a plot
identify_habit(
  pq_data,
  metric = "Multitasking_hours",
  threshold = 1,
  width = 9,
  max_window = 12,
  return = "plot"
)

# Return a summary
identify_habit(
  pq_data,
  metric = "Multitasking_hours",
  threshold = 1,
  width = 9,
  max_window = 12,
```

```
  return = "summary"
)
```

`identify_holidayweeks` *Identify Holiday Weeks based on outliers*

Description

This function scans a standard query output for weeks where collaboration hours is far outside the mean. Returns a list of weeks that appear to be holiday weeks and optionally an edited dataframe with outliers removed. By default, missing values are excluded.

As best practice, run this function prior to any analysis to remove atypical collaboration weeks from your dataset.

Usage

```
identify_holidayweeks(data, sd = 1, return = "message")
```

Arguments

`data`	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns `PersonId` and `MetricDate` present.
`sd`	The standard deviation below the mean for collaboration hours that should define an outlier week. Enter a positive number. Default is 1 standard deviation.
`return`	String specifying what to return. This must be one of the following strings: • `"message"` (default) • `"data"` • `"data_cleaned"` • `"data_dirty"` • `"plot"` See `Value` for more information.

Value

A different output is returned depending on the value passed to the `return` argument:

- `"message"`: message on console. a message is printed identifying holiday weeks.
- `"data"`: data frame. A dataset with outlier weeks flagged in a new column is returned as a dataframe.
- `"data_cleaned"`: data frame. A dataset with outlier weeks removed is returned.
- `"data_dirty"`: data frame. A dataset with only outlier weeks is returned.
- `"plot"`: ggplot object. A line plot of Collaboration Hours with holiday weeks highlighted.

Metrics used

The metric `Collaboration_hours` is used in the calculations. Please ensure that your query contains a metric with the exact same name.

See Also

Other Data Validation: `check_query()`, `extract_hr()`, `flag_ch_ratio()`, `flag_em_ratio()`, `flag_extreme()`, `flag_outlooktime()`, `hr_trend()`, `hrvar_count()`, `hrvar_count_all()`, `hrvar_trend()`, `identify_churn()`, `identify_inactiveweeks()`, `identify_nkw()`, `identify_outlier()`, `identify_privacythresho`
`identify_shifts()`, `identify_tenure()`, `track_HR_change()`, `validation_report()`

Examples

```
# Return a message by default
identify_holidayweeks(pq_data)

# Return plot
identify_holidayweeks(pq_data, return = "plot")
```

---

identify_inactiveweeks

Identify Inactive Weeks

Description

This function scans a standard query output for weeks where collaboration hours is far outside the mean for any individual person in the dataset. Returns a list of weeks that appear to be inactive weeks and optionally an edited dataframe with outliers removed.

As best practice, run this function prior to any analysis to remove atypical collaboration weeks from your dataset.

Usage

```
identify_inactiveweeks(data, sd = 2, return = "text")
```

Arguments

`data`	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns `PersonId` and `MetricDate` present.
`sd`	The standard deviation below the mean for collaboration hours that should define an outlier week. Enter a positive number. Default is 1 standard deviation.
`return`	String specifying what to return. This must be one of the following strings: • `"text"` • `"data_cleaned"`

- "data_clean"
- "data_dirty"

See Value for more information.

Value

Returns an error message by default, where 'text' is returned. When 'data_cleaned' or 'data_clean' is passed, a dataset with outlier weeks removed is returned as a dataframe. When 'data_dirty' is passed, a dataset with outlier weeks is returned as a dataframe.

See Also

Other Data Validation: check_query(), extract_hr(), flag_ch_ratio(), flag_em_ratio(), flag_extreme(), flag_outlooktime(), hr_trend(), hrvar_count(), hrvar_count_all(), hrvar_trend(), identify_churn(), identify_holidayweeks(), identify_nkw(), identify_outlier(), identify_privacythreshold(), identify_shifts(), identify_tenure(), track_HR_change(), validation_report()

---

identify_nkw *Identify Non-Knowledge workers in a Person Query using Collaboration Hours*

Description

This function scans a standard query output to identify employees with consistently low collaboration signals. Returns the % of non-knowledge workers identified by Organization, and optionally an edited data frame with non-knowledge workers removed, or the full data frame with the kw/nkw flag added.

Usage

```
identify_nkw(data, collab_threshold = 5, return = "data_summary")
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present.
collab_threshold	Positive numeric value representing the collaboration hours threshold that should be exceeded as an average for the entire analysis period for the employee to be categorized as a knowledge worker ("kw"). Default is set to 5 collaboration hours. Any versions after v1.4.3, this uses a "greater than or equal to" logic (>=), in which case persons with exactly 5 collaboration hours will pass.
return	String specifying what to return. This must be one of the following strings: • "text" • "data_with_flag"

- "data_clean"
- "data_cleaned"
- "data_summary"

See Value for more information.

Value

A different output is returned depending on the value passed to the return argument:

- "text": string. Returns a diagnostic message.
- "data_with_flag": data frame. Original input data with an additional column containing the kw/nkw flag.
- "data_clean" or "data_cleaned": data frame. Data frame with non-knowledge workers excluded.
- "data_summary": data frame. A summary table by organization listing the number and % of non-knowledge workers.

See Also

Other Data Validation: check_query(), extract_hr(), flag_ch_ratio(), flag_em_ratio(), flag_extreme(), flag_outlooktime(), hr_trend(), hrvar_count(), hrvar_count_all(), hrvar_trend(), identify_churn(), identify_holidayweeks(), identify_inactiveweeks(), identify_outlier(), identify_privacythreshold(), identify_shifts(), identify_tenure(), track_HR_change(), validation_report()

---

identify_outlier *Identify metric outliers over a date interval*

Description

This function takes in a selected metric and uses z-score (number of standard deviations) to identify outliers across time. There are applications in this for identifying weeks with abnormally low collaboration activity, e.g. holidays. Time as a grouping variable can be overridden with the group_var argument.

Usage

```
identify_outlier(
  data,
  group_var = "MetricDate",
  metric = "Collaboration_hours"
)
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present.
group_var	A string with the name of the grouping variable. Defaults to Date.
metric	Character string containing the name of the metric, e.g. "Collaboration_hours"

Value

Returns a data frame with MetricDate (if grouping variable is not set), the metric, and the corresponding z-score.

See Also

Other Data Validation: check_query(), extract_hr(), flag_ch_ratio(), flag_em_ratio(), flag_extreme(), flag_outlooktime(), hr_trend(), hrvar_count(), hrvar_count_all(), hrvar_trend(), identify_churn(), identify_holidayweeks(), identify_inactiveweeks(), identify_nkw(), identify_privacythreshold(), identify_shifts(), identify_tenure(), track_HR_change(), validation_report()

Examples

```
identify_outlier(pq_data, metric = "Collaboration_hours")
```

---

identify_privacythreshold

Identify groups under privacy threshold

Description

This function scans a standard query output for groups with of employees under the privacy threshold. The method consists in reviewing each individual HR attribute, and count the distinct people within each group.

Usage

```
identify_privacythreshold(
  data,
  hrvar = extract_hr(data),
  mingroup = 5,
  return = "table"
)
```

Arguments

data	A Standard Person Query dataset in the form of a data frame.
hrvar	A list of HR Variables to consider in the scan. Defaults to all HR attributes identified.
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
return	String specifying what to return. This must be one of the following strings: • "table" • "text" See Value for more information.

Value

A different output is returned depending on the value passed to the return argument:

- "table": data frame. A summary table of groups that fall below the privacy threshold.
- "text": string. A diagnostic message.

Returns a ggplot object by default, where 'plot' is passed in return. When 'table' is passed, a summary table is returned as a data frame.

See Also

Other Data Validation: check_query(), extract_hr(), flag_ch_ratio(), flag_em_ratio(), flag_extreme(), flag_outlooktime(), hr_trend(), hrvar_count(), hrvar_count_all(), hrvar_trend(), identify_churn(), identify_holidayweeks(), identify_inactiveweeks(), identify_nkw(), identify_outlier(), identify_shifts(), identify_tenure(), track_HR_change(), validation_report()

Examples

```
## Not run:
# Return a summary table
pq_data %>% identify_privacythreshold(return = "table")

# Return a diagnostic message
pq_data %>% identify_privacythreshold(return = "text")

## End(Not run)
```

identify_retention *Identify Retention Rate Between Two Time Periods*

Description

[Experimental]

This function calculates the retention rate of individuals who meet a specific category condition in one time period and checks whether they still meet that condition in a subsequent time period. This function complements `identify_usage_segments()`.

Usage

```
identify_retention(
  data,
  start_x,
  end_x,
  start_y,
  end_y = NULL,
  category,
  category_values = TRUE,
  return = "summary"
)
```

Arguments

`data`	A data frame containing person-level time series data with at minimum `PersonId`, `MetricDate`, and the column specified in `category`.
`start_x`	Start date for the first time period (Date or character in `"YYYY-MM-DD"` format).
`end_x`	End date for the first time period (exclusive).
`start_y`	Start date for the second time period.
`end_y`	End date for the second time period (exclusive). If `NULL`, uses all data from `start_y` onwards.
`category`	A string specifying the column name containing the binary/categorical variable to evaluate (e.g. `"IsHabitualUser"`).
`category_values`	A character vector of values in `category` that define the "positive" condition (e.g. `c("Power User", "Habitual User")`). Defaults to `TRUE` for binary columns.
`return`	A string specifying the return type. Valid options are: • `"summary"` (default): A tibble with retention statistics. • `"data"`: A list with person-level retention status and breakdown. • `"message"`: Prints a formatted message and returns the summary tibble invisibly.

Value

Depending on `return`:

- `"summary"`: A tibble with the following columns:
  - `PeriodX_Start`, `PeriodX_End`: Start and end dates of period X.
  - `PeriodY_Start`, `PeriodY_End`: Start and end dates of period Y.
  - `Category`: The category column name.
  - `CategoryValues`: The category values used.
  - `N_Original`: Number of individuals meeting the condition in period X.
  - `N_Tracked`: Number of those individuals found in period Y.
  - `N_Lost`: Number of individuals not found in period Y.
  - `N_Retained`: Number still meeting the condition in period Y.
  - `RetentionPct`: Retention percentage.
- `"data"`: A list containing `summary`, `person_status`, and `breakdown`.
- `"message"`: Prints a formatted message; returns the summary tibble invisibly.

Author(s)

Martin Chan

Examples

```
## Not run:
# Retention of Power/Habitual users from Nov 2025 to Jan 2026
identify_retention(
  data = pq_week_df,
  start_x = "2025-11-01",
  end_x = "2025-12-01",
  start_y = "2026-01-01",
  end_y = "2026-02-01",
  category = "UsageSegments_12w",
  category_values = c("Power User", "Habitual User")
)

## End(Not run)
```

---

identify_shifts *Identify shifts based on outlook time settings for work day start and end time*

Description

This function uses outlook calendar settings for start and end time of work day to identify work shifts. The relevant variables are `WorkingStartTimeSetInOutlook` and `WorkingEndTimeSetInOutlook`.

Usage

```
identify_shifts(data, return = "plot")
```

Arguments

data	A data frame containing data from the Hourly Collaboration query.
return	String specifying what to return. This must be one of the following strings: • "plot" • "table" • "data" See Value for more information.

Value

A different output is returned depending on the value passed to the return argument:

- "plot": ggplot object. A bar plot for the weekly count of shifts.
- "table": data frame. A summary table for the count of shifts.
- "data: data frame. Input data appended with the Shifts columns.

See Also

Other Data Validation: check_query(), extract_hr(), flag_ch_ratio(), flag_em_ratio(), flag_extreme(), flag_outlooktime(), hr_trend(), hrvar_count(), hrvar_count_all(), hrvar_trend(), identify_churn(), identify_holidayweeks(), identify_inactiveweeks(), identify_nkw(), identify_outlier(), identify_privacythreshold(), identify_tenure(), track_HR_change(), validation_report()

Examples

```
# Demo with `pq_data` example where Outlook Start and End times are imputed
# Use a small sample for faster runtime
pq_data_small <- dplyr::slice_sample(pq_data, prop = 0.1)

pq_data_small$WorkingStartTimeSetInOutlook <- "6:30"
pq_data_small$WorkingEndTimeSetInOutlook <- "23:30"

# Return plot
pq_data_small %>% identify_shifts()

# Return summary table
pq_data_small %>% identify_shifts(return = "table")
```

`identify_tenure` *Tenure calculation based on different input dates, returns data summary table or histogram*

Description

This function calculates employee tenure based on different input dates. `identify_tenure` uses the latest Date available if user selects "MetricDate", but also have flexibility to select a specific date, e.g. "1/1/2020".

Usage

```
identify_tenure(
  data,
  end_date = "MetricDate",
  beg_date = "HireDate",
  maxten = 40,
  return = "message"
)
```

Arguments

`data`	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns `PersonId` and `MetricDate` present.
`end_date`	A string specifying the name of the date variable representing the latest date. Defaults to "MetricDate".
`beg_date`	A string specifying the name of the date variable representing the hire date. Defaults to "HireDate".
`maxten`	A numeric value representing the maximum tenure. If the tenure exceeds this threshold, it would be accounted for in the flag message.
`return`	String specifying what to return. This must be one of the following strings: • `"message"` • `"text"` • `"plot"` • `"data_cleaned"` • `"data_dirty"` • `"data"` See `Value` for more information.

Value

A different output is returned depending on the value passed to the `return` argument:

- `"message"`: message on console with a diagnostic message.

- "text": string containing a diagnostic message.
- "plot": 'ggplot' object. A line plot showing tenure.
- "data_cleaned": data frame filtered only by rows with tenure values lying within the threshold.
- "data_dirty": data frame filtered only by rows with tenure values lying outside the threshold.
- "data": data frame with the PersonId and a calculated variable called TenureYear is returned.

See Also

Other Data Validation: check_query(), extract_hr(), flag_ch_ratio(), flag_em_ratio(), flag_extreme(), flag_outlooktime(), hr_trend(), hrvar_count(), hrvar_count_all(), hrvar_trend(), identify_churn(), identify_holidayweeks(), identify_inactiveweeks(), identify_nkw(), identify_outlier(), identify_privacythreshold(), identify_shifts(), track_HR_change(), validation_report()

Examples

```
library(dplyr)
# Add HireDate to `pq_data`
pq_data2 <-
  pq_data %>%
  mutate(HireDate = as.Date("1/1/2015", format = "%m/%d/%Y"))

identify_tenure(pq_data2)
```

---

identify_usage_segments

Identify Usage Segments based on a metric

Description

[Experimental]

This function identifies users into usage segments based on their usage volume and consistency. The segments 'Power Users', 'Habitual Users', 'Novice Users', 'Low Users', and 'Non-users' are created. There are two versions, one based on a rolling 12-week average (version = "12w") and the other on a rolling 4-week average (version = "4w"). While a main use case is for Copilot metrics e.g. 'Total_Copilot_actions', this function can be applied to other metrics, such as 'Chats_sent'.

Usage

```
identify_usage_segments(
  data,
  metric = NULL,
  metric_str = NULL,
```

```
  version = "12w",
  threshold = NULL,
  width = NULL,
  max_window = NULL,
  power_thres = 15,
  return = "data"
)
```

Arguments

`data`	A data frame with a Person query containing the metric to be classified. The data frame must include a `PersonId` column and a `MetricDate` column.
`metric`	A string representing the name of the metric column to be classified. This parameter is used when a single column represents the metric.
`metric_str`	A character vector representing the names of multiple columns to be aggregated for calculating a target metric, using row sum for aggregation. This is used when `metric` is not provided.
`version`	A string indicating the version of the classification to be used. Valid options are `"12w"` for a 12-week rolling average, `"4w"` for a 4-week rolling average, or `NULL` when using custom parameters. Defaults to `"12w"`.
`threshold`	Numeric value specifying the minimum number of times the metric sum up to in order to be a valid count. A 'greater than or equal to' logic is used. Only used when `version` is `NULL`.
`width`	Integer specifying the number of qualifying counts to consider for a habit. Only used when `version` is `NULL`.
`max_window`	Integer specifying the maximum unit of dates to consider a qualifying window for a habit. Only used when `version` is `NULL`.
`power_thres`	Numeric value specifying the minimum weekly average actions required to be classified as a 'Power User'. Defaults to 15.
`return`	A string indicating what to return from the function. Valid options are: • `"data"`: Returns the data frame with usage segments. • `"plot"`: Returns a plot of the usage segments. • `"table"`: Returns a summary table with usage segments as columns.

Details

There are three ways to use this function for usage segments classification:

1. **12-week version** (`version = "12w"`): Based on a rolling 12-week period
2. **4-week version** (`version = "4w"`): Based on a rolling 4-week period
3. **Custom parameters** (`version = NULL`): Based on user-defined parameters

This function assumes that the input dataset is grouped at the weekly level by the `MetricDate` column.

The definitions of the segments as per the 12-week definition are as follows:

- **Power User**: Averaging 15+ weekly actions (customizable via `power_thres`) and any actions in at least 9 out of past 12 weeks
- **Habitual User**: Any action in at least 9 out of past 12 weeks
- **Novice User**: Averaging at least one action over the last 12 weeks
- **Low User**: Any action in the past 12 weeks
- **Non-user**: No actions in the past 12 weeks

The definitions of the segments as per the 4-week definition are as follows:

- **Power User**: Averaging 15+ weekly actions (customizable via `power_thres`) and any actions in at least 4 out of past 4 weeks
- **Habitual User**: Any action in at least 4 out of past 4 weeks
- **Novice User**: Averaging at least one action over the last 4 weeks
- **Low User**: Any action in the past 4 weeks
- **Non-user**: No actions in the past 4 weeks

When using custom parameters (`version = NULL`), you must provide values for `threshold`, `width`, `max_window`, and optionally `power_thres`. The segment definitions become:

- **Power User**: Minimum of `threshold` actions per week in at least `width` out of past `max_window` weeks, with 15+ average weekly actions (customizable via `power_thres`)
- **Habitual User**: Minimum of `threshold` actions per week in at least `width` out of past `max_window` weeks
- **Novice User**: Average of at least one action over the last `max_window` weeks
- **Low User**: Any action in the past `max_window` weeks
- **Non-user**: No actions in the past `max_window` weeks

Value

Depending on the `return` parameter, either a data frame with usage segments or a plot visualizing the segments over time. If `"data"` is passed to `return`, the following additional columns are appended:

- When `version` is `"12w"` or `"4w"`:
  - `IsHabit12w`: Indicates whether the user has a habit based on the 12-week rolling average.
  - `IsHabit4w`: Indicates whether the user has a habit based on the 4-week rolling average.
  - `UsageSegments_12w`: The usage segment classification based on the 12-week rolling average.
  - `UsageSegments_4w`: The usage segment classification based on the 4-week rolling average.
- When `version` is `NULL`:
  - `IsHabit`: Indicates whether the user has a habit based on the provided parameters.
  - `UsageSegments`: The usage segment classification based on the provided parameters.
- `IsHabit12w`: Indicates whether the user has a habit based on the 12-week rolling average.
- `IsHabit4w`: Indicates whether the user has a habit based on the 4-week rolling average.

- `UsageSegments_12w`: The usage segment classification based on the 12-week rolling average.
- `UsageSegments_4w`: The usage segment classification based on the 4-week rolling average.

If `"table"` is passed to `return`, a summary table is returned with one row per `MetricDate` and usage segments as columns containing percentages. The table includes:

- `MetricDate`: The date of the metric
- Segment columns (in order): `Non-user`, `Low User`, `Novice User`, `Habitual User`, `Power User` (only segments present in the data are included)
- `n`: The total number of distinct persons for that date

@import slider slide_dbl @import tidyr

Examples

```
# Example usage with a single metric column
identify_usage_segments(
  data = pq_data,
  metric = "Emails_sent",
  version = "12w",
  return = "plot"
)

# Example usage with multiple metric columns
identify_usage_segments(
  data = pq_data,
  metric_str = c(
    "Copilot_actions_taken_in_Teams",
    "Copilot_actions_taken_in_Outlook",
    "Copilot_actions_taken_in_Excel",
    "Copilot_actions_taken_in_Word",
    "Copilot_actions_taken_in_Powerpoint"
  ),
  version = "4w",
  return = "plot"
)

# Example usage with custom parameters
identify_usage_segments(
  data = pq_data,
  metric = "Emails_sent",
  version = NULL,
  threshold = 2,
  width = 5,
  max_window = 8,
  return = "plot"
)

# Example usage with custom power user threshold
identify_usage_segments(
  data = pq_data,
  metric = "Emails_sent",
```

```
  version = "12w",
  power_thres = 20,
  return = "plot"
)

# Return summary table
identify_usage_segments(
  data = pq_data,
  metric = "Emails_sent",
  version = "12w",
  return = "table"
)
```

`import_query` *Import a query from Viva Insights Analyst Experience*

Description

Import a Viva Insights Query from a .csv file, with variable classifications optimised for other functions in the package.

Usage

```
import_query(
  x,
  pid = NULL,
  dateid = NULL,
  date_format = "%m/%d/%Y",
  convert_date = TRUE,
  encoding = "UTF-8"
)
```

Arguments

`x`	String containing the path to the Viva Insights query to be imported. The input file must be a .csv file, and the file extension must be explicitly entered, e.g. `"/files/standard query.csv"`
`pid`	String specifying the unique person or individual identifier variable. `import_query` renames this to `PersonId` so that this is compatible with other functions in the package. Defaults to `NULL`, where no action is taken.
`dateid`	String specifying the date variable. `import_query` renames this to `MetricDate` so that this is compatible with other functions in the package. Defaults to `NULL`, where no action is taken.
`date_format`	String specifying the date format for converting any variable that may be a date to a Date variable. Defaults to `"%m/%d/%Y"`.
`convert_date`	Logical. Defaults to `TRUE`. When set to `TRUE`, any variable that matches true with `is_date_format()` gets converted to a Date variable. When set to `FALSE`, this step is skipped.

encoding — String to specify encoding to be used within `data.table::fread()`. See `data.table::fread()` documentation for more information. Defaults to `'UTF-8'`.

Details

`import_query()` uses `data.table::fread()` to import .csv files for speed, and by default `stringsAsFactors` is set to FALSE. A data frame is returned by the function (not a `data.table`). Column names are automatically cleaned, replacing spaces and special characters with underscores.

Value

A `tibble` is returned.

See Also

Other Import and Export: `copy_df()`, `create_dt()`, `export()`, `prep_query()`

---

is_date_format *Identify whether string is a date format*

Description

This function uses regular expression to determine whether a string is of the format `"mdy"`, separated by `"-"`, `"/"`, or `"."`, returning a logical vector.

Usage

```
is_date_format(string)
```

Arguments

string — Character string to test whether is a date format.

Value

logical value indicating whether the string is a date format.

See Also

Other Support: `any_idate()`, `camel_clean()`, `check_inputs()`, `cut_hour()`, `extract_date_range()`, `extract_hr()`, `heat_colours()`, `maxmin()`, `pairwise_count()`, `read_preamble()`, `rgb2hex()`, `totals_bind()`, `totals_col()`, `tstamp()`, `us_to_space()`, `wrap()`

Examples

```
is_date_format("1/5/2020")
```

`IV_report` *Generate a Information Value HTML Report*

Description

The function generates an interactive HTML report using Standard Person Query data as an input. The report contains a full Information Value analysis, a data exploration technique that helps determine which columns in a data set have predictive power or influence on the value of a specified dependent variable.

Usage

```
IV_report(
  data,
  predictors = NULL,
  outcome,
  bins = 5,
  max_var = 9,
  path = "IV report",
  timestamp = TRUE
)
```

Arguments

`data`	A Standard Person Query dataset in the form of a data frame.
`predictors`	A character vector specifying the columns to be used as predictors. Defaults to NULL, where all numeric vectors in the data will be used as predictors.
`outcome`	A string specifying a binary variable, i.e. can only contain the values 1 or 0.
`bins`	Number of bins to use in `Information::create_infotables()`, defaults to 10.
`max_var`	Numeric value to represent the maximum number of variables to show on plots.
`path`	Pass the file path and the desired file name, *excluding the file extension*. For example, `"IV report"`.
`timestamp`	Logical vector specifying whether to include a timestamp in the file name. Defaults to TRUE.

Value

An HTML report with the same file name as specified in the arguments is generated in the working directory. No outputs are directly returned by the function.

Creating a report

Below is an example on how to run the report.

```
library(dplyr)

pq_data %>%
 mutate(CH_binary = ifelse(Collaboration_hours > 12, 1, 0)) %>% # Simulate binary variable
  IV_report(outcome =  "CH_binary",
            predictors = c("Email_hours", "Meeting_hours"))
```

See Also

Other Reports: `generate_report()`, `meeting_tm_report()`, `read_preamble()`, `validation_report()`

Other Variable Association: `create_IV()`

Other Information Value: `create_IV()`

`jitter_metrics` *Jitter metrics in a data frame*

Description

Convenience wrapper around `jitter()` to add a layer of anonymity to a query. This can be used in combination with `anonymise()` to produce a demo dataset from real data.

Usage

```
jitter_metrics(data, cols = NULL, ...)
```

Arguments

`data`	Data frame containing a query.
`cols`	Character vector containing the metrics to jitter. When set to `NULL` (default), all numeric columns in the data frame are jittered.
`...`	Additional arguments to pass to `jitter()`.

Value

data frame where numeric columns specified by `cols` are jittered using the function `jitter()`.

See Also

anonymise

Examples

```
jittered <- jitter_metrics(pq_data, cols = "Collaboration_hours")

# compare jittered vs original results of top rows
head(
  data.frame(
    original = pq_data$Collaboration_hours,
    jittered = jittered$Collaboration_hours
  )
)
```

---

keymetrics_scan *Run a summary of Key Metrics from the Standard Person Query data*

Description

Returns a heatmapped table by default, with options to return a table.

Usage

```
keymetrics_scan(
  data,
  hrvar = "Organization",
  mingroup = 5,
  metrics = c("Collaboration_span", "Collaboration_hours",
    "After_hours_collaboration_hours", "Meetings", "Meeting_hours",
    "After_hours_meeting_hours", "Meeting_and_call_hours_with_manager_1_1",
    "Meeting_and_call_hours_with_manager", "Emails_sent", "Email_hours",
   "After_hours_email_hours", "Internal_network_size", "External_network_size"),
  return = "plot",
  low = rgb2hex(7, 111, 161),
  mid = rgb2hex(241, 204, 158),
  high = rgb2hex(216, 24, 42),
  textsize = 2
)
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns `PersonId` and `MetricDate` present. If your data is already aggregated (e.g. one row per group), use the equivalent `*_asis()` variant of this function instead.
hrvar	String containing the name of the HR Variable by which to split metrics. Defaults to `"Organization"`. To run the analysis on the total instead of splitting by an HR attribute, supply `NULL` (without quotes).

mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
metrics	A character vector containing the variable names to calculate averages of.
return	Character vector specifying what to return, defaults to "plot". Valid inputs are "plot" and "table".
low	String specifying colour code to use for low-value metrics. Arguments are passed directly to ggplot2::scale_fill_gradient2().
mid	String specifying colour code to use for mid-value metrics. Arguments are passed directly to ggplot2::scale_fill_gradient2().
high	String specifying colour code to use for high-value metrics. Arguments are passed directly to ggplot2::scale_fill_gradient2().
textsize	A numeric value specifying the text size to show in the plot.

Value

Returns a ggplot object by default, when 'plot' is passed in return. When 'table' is passed, a summary table is returned as a data frame.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Examples

```
## Not run:
# Heatmap plot is returned by default
keymetrics_scan(pq_data)

# Heatmap plot with custom colours
keymetrics_scan(pq_data, low = "purple", high = "yellow")

# Return summary table
keymetrics_scan(pq_data, hrvar = "LevelDesignation", return = "table")

## End(Not run)
```

keymetrics_scan_asis *Run a summary of Key Metrics without aggregation*

Description

Return a heatmapped table directly from the aggregated / summarised data. Unlike `keymetrics_scan()` which performs a person-level aggregation, there is no calculation for `keymetrics_scan_asis()` and the values are rendered as they are passed into the function.

Usage

```
keymetrics_scan_asis(
  data,
  row_var,
  col_var,
  group_var = col_var,
  value_var = "value",
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  ylab = row_var,
  xlab = "Metrics",
  rounding = 1,
  low = rgb2hex(7, 111, 161),
  mid = rgb2hex(241, 204, 158),
  high = rgb2hex(216, 24, 42),
  textsize = 2
)
```

Arguments

`data`	Aggregated or summarised data frame to plot. Unlike `keymetrics_scan()`, this function does **not** require panel data (i.e. `PersonId` and `MetricDate` are not required). It is recommended to provide data in a 'long' table format where one grouping column forms the rows, a second column forms the columns, and a third numeric columns forms the
`row_var`	String containing name of the grouping variable that will form the rows of the heatmapped table.
`col_var`	String containing name of the grouping variable that will form the columns of the heatmapped table.
`group_var`	String containing name of the grouping variable by which heatmapping would apply. Defaults to `col_var`.
`value_var`	String containing name of the value variable that will form the values of the heatmapped table. Defaults to `"value"`.
`title`	Title of the plot.

subtitle	Subtitle of the plot.
caption	Caption of the plot.
ylab	Y-axis label for the plot (group axis)
xlab	X-axis label of the plot (bar axis).
rounding	Numeric value to specify number of digits to show in data labels
low	String specifying colour code to use for low-value metrics. Arguments are passed directly to ggplot2::scale_fill_gradient2().
mid	String specifying colour code to use for mid-value metrics. Arguments are passed directly to ggplot2::scale_fill_gradient2().
high	String specifying colour code to use for high-value metrics. Arguments are passed directly to ggplot2::scale_fill_gradient2().
textsize	A numeric value specifying the text size to show in the plot.

Value

ggplot object for a heatmap table.

Examples

```
library(dplyr)

# Compute summary table
out_df <-
  pq_data %>%
  group_by(Organization) %>%
  summarise(
    across(
      .cols = c(
        Email_hours,
        Collaboration_hours
        ),
      .fns = ~median(., na.rm = TRUE)
      ),
      .groups = "drop"
    ) %>%
tidyr::pivot_longer(
  cols = c("Email_hours", "Collaboration_hours"),
  names_to = "metrics"
)

keymetrics_scan_asis(
  data = out_df,
  col_var = "metrics",
  row_var = "Organization"
)

# Show data the other way round
keymetrics_scan_asis(
  data = out_df,
```

```
  col_var = "Organization",
  row_var = "metrics",
  group_var = "metrics"
)
```

`maxmin` *Max-Min Scaling Function*

Description

This function allows you to scale vectors or an entire data frame using the max-min scaling method A numeric vector is always returned.

Usage

```
maxmin(x)
```

Arguments

`x`	Pass a vector or the required columns of a data frame through this argument.

Details

This is used within `keymetrics_scan()` to enable row-wise heatmapping. Originally implemented in https://github.com/martinctc/surveytoolbox.

Value

Returns a numeric vector with the input rescaled.

See Also

Other Support: `any_idate()`, `camel_clean()`, `check_inputs()`, `cut_hour()`, `extract_date_range()`, `extract_hr()`, `heat_colours()`, `is_date_format()`, `pairwise_count()`, `read_preamble()`, `rgb2hex()`, `totals_bind()`, `totals_col()`, `tstamp()`, `us_to_space()`, `wrap()`

Examples

```
numbers <- c(15, 40, 10, 2)
maxmin(numbers)
```

meeting_dist *Distribution of Meeting Hours as a 100% stacked bar*

Description

Analyze Meeting Hours distribution. Returns a stacked bar plot by default. Additional options available to return a table with distribution elements.

Usage

```
meeting_dist(
  data,
  hrvar = "Organization",
  mingroup = 5,
  return = "plot",
  cut = c(5, 10, 15)
)
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present. If your data is already aggregated (e.g. one row per group), use the equivalent *_asis() variant of this function instead.
hrvar	String containing the name of the HR Variable by which to split metrics. Defaults to "Organization". To run the analysis on the total instead of splitting by an HR attribute, supply NULL (without quotes).
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
return	String specifying what to return. This must be one of the following strings: • "plot" • "table" See Value for more information.
cut	A numeric vector of length three to specify the breaks for the distribution, e.g. c(10, 15, 20)

Value

A different output is returned depending on the value passed to the return argument:

- "plot": 'ggplot' object. A stacked bar plot for the metric.
- "table": data frame. A summary table for the metric.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Meetings: meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_tm_report(), meeting_trend()

Examples

```
# Return plot
meeting_dist(pq_data, hrvar = "Organization")

# Return summary table
meeting_dist(pq_data, hrvar = "Organization", return = "table")

# Return result with a custom specified breaks
meeting_dist(pq_data, hrvar = "LevelDesignation", cut = c(4, 7, 9))
```

---

meeting_fizz *Distribution of Meeting Hours (Fizzy Drink plot)*

Description

Analyze weekly meeting hours distribution, and returns a 'fizzy' scatter plot by default. Additional options available to return a table with distribution elements.

Usage

```
meeting_fizz(data, hrvar = "Organization", mingroup = 5, return = "plot")
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present. If your data is already aggregated (e.g. one row per group), use the equivalent *_asis() variant of this function instead.

hrvar	String containing the name of the HR Variable by which to split metrics. Defaults to "Organization". To run the analysis on the total instead of splitting by an HR attribute, supply NULL (without quotes).
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
return	String specifying what to return. This must be one of the following strings: • "plot" • "table" See Value for more information.

Details

Uses the metric Meeting_hours.

Value

A different output is returned depending on the value passed to the return argument:

- "plot": 'ggplot' object. A jittered scatter plot for the metric.
- "table": data frame. A summary table for the metric.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Meetings: meeting_dist(), meeting_line(), meeting_rank(), meeting_summary(), meeting_tm_report(), meeting_trend()

Examples

```
# Return plot
meeting_fizz(pq_data, hrvar = "Organization", return = "plot")

# Return summary table
meeting_fizz(pq_data, hrvar = "Organization", return = "table")
```

meeting_line *Meeting Time Trend - Line Chart*

Description

Provides a week by week view of meeting time, visualised as line charts. By default returns a line chart for meeting hours, with a separate panel per value in the HR attribute. Additional options available to return a summary table.

Usage

```
meeting_line(
  data,
  hrvar = "Organization",
  mingroup = 5,
  return = "plot",
  label = FALSE
)
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present. If your data is already aggregated (e.g. one row per group), use the equivalent *_asis() variant of this function instead.
hrvar	String containing the name of the HR Variable by which to split metrics. Defaults to "Organization". To run the analysis on the total instead of splitting by an HR attribute, supply NULL (without quotes).
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
return	String specifying what to return. This must be one of the following strings: • "plot" • "table" See Value for more information.
label	Logical value to determine whether to show data point labels on the plot. If TRUE, both geom_point() and geom_text() are added to display data labels rounded to 1 decimal place above each data point. Defaults to FALSE.

Value

A different output is returned depending on the value passed to the return argument:

- "plot": 'ggplot' object. A faceted line plot for the metric.
- "table": data frame. A summary table for the metric.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Meetings: meeting_dist(), meeting_fizz(), meeting_rank(), meeting_summary(), meeting_tm_report(), meeting_trend()

Examples

```
# Return a line plot
meeting_line(pq_data, hrvar = "LevelDesignation")

# Return summary table
meeting_line(pq_data, hrvar = "LevelDesignation", return = "table")
```

---

meeting_rank *Meeting Hours Ranking*

Description

This function scans a standard query output for groups with high levels of Weekly Meeting Collaboration. Returns a plot by default, with an option to return a table with a all of groups (across multiple HR attributes) ranked by hours of digital collaboration.

Usage

```
meeting_rank(
  data,
  hrvar = extract_hr(data),
  mingroup = 5,
  mode = "simple",
  plot_mode = 1,
  return = "plot"
)
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present. If your data is already aggregated (e.g. one row per group), use the equivalent *_asis() variant of this function instead.
hrvar	String containing the name of the HR Variable by which to split metrics. Defaults to "Organization". To run the analysis on the total instead of splitting by an HR attribute, supply NULL (without quotes).
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
mode	String to specify calculation mode. Must be either: • "simple" • "combine"
plot_mode	Numeric vector to determine which plot mode to return. Must be either 1 or 2, and is only used when return = "plot". • 1: Top and bottom five groups across the data population are highlighted • 2: Top and bottom groups *per* organizational attribute are highlighted
return	String specifying what to return. This must be one of the following strings: • "plot" (default) • "table" See Value for more information.

Details

Uses the metric Meeting_hours. See create_rank() for applying the same analysis to a different metric.

Value

A different output is returned depending on the value passed to the return argument:

- "plot": 'ggplot' object. A bubble plot where the x-axis represents the metric, the y-axis represents the HR attributes, and the size of the bubbles represent the size of the organizations. Note that there is no plot output if mode is set to "combine".
- "table": data frame. A summary table for the metric.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(),

create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Meetings: meeting_dist(), meeting_fizz(), meeting_line(), meeting_summary(), meeting_tm_report(), meeting_trend()

Examples

```
# Return rank table
meeting_rank(data = pq_data, return = "table")

# Return plot
meeting_rank(data = pq_data, return = "plot")
```

---

meeting_summary *Meeting Summary*

Description

Provides an overview analysis of weekly meeting hours. Returns a bar plot showing average weekly meeting hours by default. Additional options available to return a summary table.

Usage

```
meeting_summary(data, hrvar = "Organization", mingroup = 5, return = "plot")

meeting_sum(data, hrvar = "Organization", mingroup = 5, return = "plot")
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present. If your data is already aggregated (e.g. one row per group), use the equivalent *_asis() variant of this function instead.
hrvar	String containing the name of the HR Variable by which to split metrics. Defaults to "Organization". To run the analysis on the total instead of splitting by an HR attribute, supply NULL (without quotes).
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
return	String specifying what to return. This must be one of the following strings: • "plot" • "table" See Value for more information.

Value

A different output is returned depending on the value passed to the `return` argument:

- `"plot"`: 'ggplot' object. A bar plot for the metric.
- `"table"`: data frame. A summary table for the metric.

See Also

Other Visualization: `afterhours_dist()`, `afterhours_fizz()`, `afterhours_line()`, `afterhours_rank()`, `afterhours_summary()`, `afterhours_trend()`, `collaboration_area()`, `collaboration_dist()`, `collaboration_fizz()`, `collaboration_line()`, `collaboration_rank()`, `collaboration_sum()`, `collaboration_trend()`, `create_bar()`, `create_bar_asis()`, `create_boxplot()`, `create_bubble()`, `create_dist()`, `create_fizz()`, `create_inc()`, `create_line()`, `create_line_asis()`, `create_period_scatter()`, `create_radar()`, `create_rank()`, `create_rogers()`, `create_sankey()`, `create_scatter()`, `create_stacked()`, `create_survival()`, `create_tracking()`, `create_trend()`, `email_dist()`, `email_fizz()`, `email_line()`, `email_rank()`, `email_summary()`, `email_trend()`, `external_dist()`, `external_fizz()`, `external_line()`, `external_rank()`, `external_sum()`, `hr_trend()`, `hrvar_count()`, `hrvar_trend()`, `keymetrics_scan()`, `meeting_dist()`, `meeting_fizz()`, `meeting_line()`, `meeting_rank()`, `meeting_trend()`, `one2one_dist()`, `one2one_fizz()`, `one2one_freq()`, `one2one_line()`, `one2one_rank()`, `one2one_sum()`, `one2one_trend()`

Other Meetings: `meeting_dist()`, `meeting_fizz()`, `meeting_line()`, `meeting_rank()`, `meeting_tm_report()`, `meeting_trend()`

Examples

```
# Return a ggplot bar chart
meeting_summary(pq_data, hrvar = "LevelDesignation")

# Return a summary table
meeting_summary(pq_data, hrvar = "LevelDesignation", return = "table")
```

---

meeting_tm_report *Generate a Meeting Text Mining report in HTML*

Description

Create a text mining report in HTML based on Meeting Subject Lines

Usage

```
meeting_tm_report(
  data,
  path = "meeting text mining report",
  stopwords = NULL,
  timestamp = TRUE,
  keep = 100,
  seed = 100
)
```

Arguments

data	A Meeting Query dataset in the form of a data frame.
path	Pass the file path and the desired file name, *excluding the file extension*. For example, "meeting text mining report".
stopwords	A character vector OR a single-column data frame labelled 'word' containing custom stopwords to remove.
timestamp	Logical vector specifying whether to include a timestamp in the file name. Defaults to TRUE.
keep	A numeric vector specifying maximum number of words to keep.
seed	A numeric vector to set seed for random generation.

Details

Note that the column Subject must be available within the input data frame in order to run.d

Value

An HTML report with the same file name as specified in the arguments is generated in the working directory. No outputs are directly returned by the function.

How to run

```
meeting_tm_report(mt_data)
```

This will generate a HTML report as specified in path.

See Also

Other Reports: IV_report(), generate_report(), read_preamble(), validation_report()

Other Meetings: meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend()

Other Text-mining: pairwise_count(), tm_clean(), tm_cooc(), tm_freq(), tm_wordcloud()

---

meeting_trend *Meeting Hours Time Trend*

Description

Provides a week by week view of meeting time. By default returns a week by week heatmap, highlighting the points in time with most activity. Additional options available to return a summary table.

Usage

```
meeting_trend(data, hrvar = "Organization", mingroup = 5, return = "plot")
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present. If your data is already aggregated (e.g. one row per group), use the equivalent *_asis() variant of this function instead.
hrvar	String containing the name of the HR Variable by which to split metrics. Defaults to "Organization". To run the analysis on the total instead of splitting by an HR attribute, supply NULL (without quotes).
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
return	Character vector specifying what to return, defaults to "plot". Valid inputs are "plot" and "table".

Details

Uses the metric Meeting_hours.

Value

Returns a 'ggplot' object by default, where 'plot' is passed in return. When 'table' is passed, a summary table is returned as a data frame.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Meetings: meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_tm_report()

Examples

```
# Run plot
meeting_trend(pq_data)

# Run table
meeting_trend(pq_data, hrvar = "LevelDesignation", return = "table")
```

`mt_data` *Sample Meeting Query dataset*

Description

A dataset generated from a Meeting Query from Viva Insights.

Usage

```
mt_data
```

Format

A data frame with 612 rows and 41 variables:

MeetingId

Attendee_meeting_hours

Number_of_attendees

Number_of_attendees_multitasking

Number_of_attendees_who_didn_t_end_the_meeting_on_time

Number_of_attendees_who_didn_t_join_the_meeting_on_time

Number_of_attendees_who_ended_the_meeting_on_time

Number_of_attendees_who_joined_the_meeting_on_time

Number_of_chats_sent_during_the_meeting

Number_of_emails_sent_during_the_meeting

Number_of_redundant_attendees

Subject

All_Day_Meeting

Cancelled

Recurring

Accept_count

No_response_count

Decline_count

Tentatively_accepted_count

Intended_participant_count

Collaboration_start_time

Organizer

zId

attainment

TimeZone

SupervisorIndicator

Region

Population_Type

Organization

OnsiteDays

Number_of_directs

LevelDesignation

Layer

HireDate

GroupNum

GroupName

FunctionType

Domain

ADO_PersonSK

ADO_PersonIndicator

Duration

Value

data frame.

Source

https://learn.microsoft.com/en-us/viva/insights/advanced/analyst/meeting-query/

See Also

Other Data: g2g_data, p2p_data, p2p_data_sim(), pq_data

`network_g2g` *Create a network plot with the group-to-group query*

Description

Pass a data frame containing a group-to-group query and return a network plot. Automatically handles "Within Group" and "Other_collaborators" values within query data.

Usage

```
network_g2g(
  data,
  primary = NULL,
  secondary = NULL,
  metric = "Group_collaboration_time_invested",
  algorithm = "fr",
  node_colour = "lightblue",
  exc_threshold = 0.1,
  org_count = NULL,
  subtitle = "Collaboration Across Organizations",
  return = "plot"
)
```

Arguments

`data`	Data frame containing a group-to-group query.
`primary`	String containing the variable name for the Primary Collaborator column.
`secondary`	String containing the variable name for the Secondary Collaborator column.
`metric`	String containing the variable name for metric. Defaults to `Group_collaboration_time_invested`.
`algorithm`	String to specify the node placement algorithm to be used. Defaults to `"fr"` for the force-directed algorithm of Fruchterman and Reingold. See https://rdrr.io/cran/ggraph/man/layout_tbl_graph_igraph.html for a full list of options.
`node_colour`	String or named vector to specify the colour to be used for displaying nodes. Defaults to `"lightblue"`. • If `"vary"` is supplied, a different colour is shown for each node at random. • If a named vector is supplied, the names must match the values of the variable provided for the `primary` and `secondary` columns. See example section for details.
`exc_threshold`	Numeric value between 0 and 1 specifying the exclusion threshold to apply. Defaults to 0.1, which means that the plot will only display collaboration above 10% of a node's total collaboration. This argument has no impact on `"data"` or `"table"` return.
`org_count`	Optional data frame to provide the size of each organization in the `secondary` attribute. The data frame should contain only two columns: • Name of the `secondary` attribute excluding any prefixes, e.g. `"Organization"`. Must be of character or factor type. • `"n"`. Must be of numeric type. Defaults to `NULL`, where node sizes will be fixed.
`subtitle`	String to override default plot subtitle.
`return`	String specifying what to return. This must be one of the following strings: • `"plot"` • `"table"`

- "network"
- "data"

See Value for more information.

Value

A different output is returned depending on the value passed to the return argument:

- "plot": 'ggplot' object. A group-to-group network plot.
- "table": data frame. An interactive matrix of the network.
- "network: 'igraph' object used for creating the network plot.
- "data": data frame. A long table of the underlying data.

See Also

Other Network: g2g_data, network_p2p(), network_summary(), p2p_data, p2p_data_sim()

Examples

```
# Return a network plot
g2g_data %>% network_g2g()

# Return a network plot - Meeting hours and 5% threshold
network_g2g(
  data = g2g_data,
  primary = "PrimaryCollaborator_Organization",
  secondary = "SecondaryCollaborator_Organization",
  exc_threshold = 0.05
)

# Return a network plot - custom-specific colours
# Get labels of orgs and assign random colours
org_str <- unique(g2g_data$PrimaryCollaborator_Organization)

col_str <-
  sample(
    x = heat_colours(n = length(org_str)), # generate colour codes for each one
    size = length(org_str),
    replace = TRUE
  )

# Create and supply a named vector to `node_colour`
names(col_str) <- org_str

g2g_data %>%
  network_g2g(node_colour = col_str)


# Return a network plot with circle layout
# Vary node colours and add org sizes
org_tb <-
```

```
  data.frame(
    Organization = c(
      "G&A East",
      "G&A West",
      "G&A North",
      "South Sales",
      "North Sales",
      "G&A South"
    ),
    n = sample(30:1000, size = 6)
  )

g2g_data %>%
  network_g2g(algorithm = "circle",
              node_colour = "vary",
              org_count = org_tb)

# Return an interaction matrix
# Minimum arguments specified
g2g_data %>%
  network_g2g(return = "table")
```

---

network_p2p *Perform network analysis with the person-to-person query*

Description

[Experimental]

Analyse a person-to-person (P2P) network query, with multiple visualisation and analysis output options. Pass a data frame containing a person-to-person query and return a network visualization. Options are available for community detection using either the Louvain or the Leiden algorithms.

Note: The data frame must only contain a single `MetricDate` value, as the network represents a snapshot at a specific point in time. If multiple date values are present, filter the data frame to a specific date before using this function.

Usage

```
network_p2p(
  data,
  hrvar = "Organization",
  return = "plot",
  centrality = NULL,
  community = NULL,
  weight = NULL,
  comm_args = NULL,
  layout = "mds",
  path = paste("p2p", community, sep = "_"),
```

```
  style = "igraph",
  bg_fill = "#FFFFFF",
  font_col = "grey20",
  legend_pos = "right",
  palette = "rainbow",
  node_alpha = 0.7,
  edge_alpha = 1,
  edge_col = "#777777",
  node_sizes = c(1, 20),
  seed = 1
)
```

Arguments

`data`	Data frame containing a person-to-person query.
`hrvar`	String containing the label for the HR attribute.
`return`	A different output is returned depending on the value passed to the `return` argument: • `'plot'` (default) • `'plot-pdf'` • `'sankey'` • `'table'` • `'data'` • `'network'`
`centrality`	string to determines which centrality measure is used to scale the size of the nodes. All centrality measures are automatically calculated when it is set to one of the below values, and reflected in the `'network'` and `'data'` outputs. Measures include: • `betweenness` • `closeness` • `degree` • `eigenvector` • `pagerank` When `centrality` is set to NULL, no centrality is calculated in the outputs and all the nodes would have the same size.
`community`	String determining which community detection algorithms to apply. Valid values include: • `NULL` (default): compute analysis or visuals without computing communities. • `"louvain"` • `"leiden"` • `"edge_betweenness"` • `"fast_greedy"` • `"fluid_communities"`

- "infomap"
- "label_prop"
- "leading_eigen"
- "optimal"
- "spinglass"
- "walk_trap"

These values map to the community detection algorithms offered by igraph. For instance, "leiden" is based on igraph::cluster_leiden(). Please see the bottom of https://igraph.org/r/html/1.3.0/cluster_leiden.html on all applications and parameters of these algorithms. .

weight — String to specify which column to use as weights for the network. To create a graph without weights, supply NULL to this argument.

comm_args — list containing the arguments to be passed through to igraph's clustering algorithms. Arguments must be named. See examples section on how to supply arguments in a named list.

layout — String to specify the node placement algorithm to be used. Defaults to "mds" for the deterministic multi-dimensional scaling of nodes. See https://rdrr.io/cran/ggraph/man/layout_tbl_graph_igraph.html for a full list of options.

path — File path for saving the PDF output. Defaults to a timestamped path based on current parameters.

style — String to specify which plotting style to use for the network plot. Valid values include:

- "igraph"
- "ggraph"

bg_fill — String to specify background fill colour.

font_col — String to specify font colour.

legend_pos — String to specify position of legend. Defaults to "right". See ggplot2::theme(). This is applicable for both the 'ggraph' and the fast plotting method. Valid inputs include:

- "bottom"
- "top"
- "left" -"right"

palette — String specifying the function to generate a colour palette with a single argument n. Uses "rainbow" by default.

node_alpha — A numeric value between 0 and 1 to specify the transparency of the nodes. Defaults to 0.7.

edge_alpha — A numeric value between 0 and 1 to specify the transparency of the edges (only for 'ggraph' mode). Defaults to 1.

edge_col — String to specify edge link colour.

node_sizes — Numeric vector of length two to specify the range of node sizes to rescale to, when centrality is set to a non-null value.

seed — Seed for the random number generator passed to either set.seed() when the louvain or leiden community detection algorithm is used, to ensure consistency. Only applicable when community is set to one of the valid non-null values.

Value

A different output is returned depending on the value passed to the `return` argument:

- `'plot'`: return a network plot, interactively within R.
- `'plot-pdf'`: save a network plot as PDF. This option is recommended when the graph is large, which make take a long time to run if `return = 'plot'` is selected. Use this together with `path` to control the save location.
- `'sankey'`: return a sankey plot combining communities and HR attribute. This is only valid if a community detection method is selected at `community`.
- `'table'`: return a vertex summary table with counts in communities and HR attribute. When `centrality` is non-NULL, the average centrality values are calculated per group.
- `'data'`: return a vertex data file that matches vertices with communities and HR attributes.
- `'network'`: return 'igraph' object.

See Also

Other Network: `g2g_data`, `network_g2g()`, `network_summary()`, `p2p_data`, `p2p_data_sim()`

Examples

```
p2p_df <- p2p_data_sim(dim = 1, size = 100)

# default - ggraph visual
network_p2p(data = p2p_df, style = "ggraph")

# return vertex table
network_p2p(data = p2p_df, return = "table")


# return vertex table with community detection
network_p2p(data = p2p_df, community = "leiden", return = "table")

# leiden - igraph style with custom resolution parameters
network_p2p(data = p2p_df, community = "leiden", comm_args = list("resolution" = 0.1))

# louvain - ggraph style, using custom palette
network_p2p(
  data = p2p_df,
  style = "ggraph",
  community = "louvain",
  palette = "heat_colors"
)

# leiden - return a sankey visual with custom resolution parameters
network_p2p(
  data = p2p_df,
  community = "leiden",
  return = "sankey",
  comm_args = list("resolution" = 0.1)
)
```

```
# using `fluid_communities` algorithm with custom parameters
network_p2p(
  data = p2p_df,
  community = "fluid_communities",
  comm_args = list("no.of.communities" = 5)
)

# Calculate centrality measures and leiden communities, return at node level
network_p2p(
  data = p2p_df,
  centrality = "betweenness",
  community = "leiden",
  return = "data"
) %>%
  dplyr::glimpse()
```

`network_summary` *Summarise node centrality statistics with an igraph object*

Description

Pass an igraph object to the function and obtain centrality statistics for each node in the object as a data frame. This function works as a wrapper of the centralization functions in 'igraph'.

Usage

```
network_summary(graph, hrvar = NULL, return = "table")
```

Arguments

`graph`	'igraph' object that can be returned from `network_g2g()` or `network_p2p()` when the `return` argument is set to `"network"`.
`hrvar`	String containing the name of the HR Variable by which to split metrics. Defaults to `NULL`.
`return`	String specifying what output to return. Valid inputs include: • `"table"` • `"network"` • `"plot"` See `Value` for more information.

Value

By default, a data frame containing centrality statistics. Available statistics include:

- betweenness: number of shortest paths going through a node.
- closeness: number of steps required to access every other node from a given node.
- degree: number of connections linked to a node.
- eigenvector: a measure of the influence a node has on a network.
- pagerank: calculates the PageRank for the specified vertices. Please refer to the igraph package documentation for the detailed technical definition.

When `"network"` is passed to `"return"`, an 'igraph' object is returned with additional node attributes containing centrality scores.

When `"plot"` is passed to `"return"`, a summary table is returned showing the average centrality scores by HR attribute. This is currently available if there is a valid HR attribute.

See Also

Other Network: `g2g_data`, `network_g2g()`, `network_p2p()`, `p2p_data`, `p2p_data_sim()`

Examples

```
# Simulate a p2p network
p2p_data <- p2p_data_sim(size = 100)
g <- network_p2p(data = p2p_data, return = "network")

# Return summary table
network_summary(graph = g, return = "table")

# Return network with node centrality statistics
network_summary(graph = g, return = "network")

# Return summary plot
network_summary(graph = g, return = "plot", hrvar = "Organization")

# Simulate a g2g network and return table
g2 <- g2g_data %>% network_g2g(return = "network")
network_summary(graph = g2, return = "table")
```

---

one2one_dist *Distribution of Manager 1:1 Time as a 100% stacked bar*

Description

Analyze Manager 1:1 Time distribution. Returns a stacked bar plot of different buckets of 1:1 time. Additional options available to return a table with distribution elements.

Usage

```
one2one_dist(
  data,
  hrvar = "Organization",
  mingroup = 5,
  dist_colours = c("#facebc", "#fcf0eb", "#b4d5dd", "#bfe5ee"),
  return = "plot",
  cut = c(5, 15, 30)
)
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present. If your data is already aggregated (e.g. one row per group), use the equivalent *_asis() variant of this function instead.
hrvar	String containing the name of the HR Variable by which to split metrics. Defaults to "Organization". To run the analysis on the total instead of splitting by an HR attribute, supply NULL (without quotes).
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
dist_colours	A character vector of length four to specify colour codes for the stacked bars.
return	String specifying what to return. This must be one of the following strings: • "plot" • "table" See Value for more information.
cut	A numeric vector of length three to specify the breaks for the distribution, e.g. c(10, 15, 20)

Value

A different output is returned depending on the value passed to the return argument:

- "plot": 'ggplot' object. A stacked bar plot for the metric.
- "table": data frame. A summary table for the metric.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(),

email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Managerial Relations: one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Examples

```
# Return plot
one2one_dist(pq_data, hrvar = "Organization", return = "plot")

# Return summary table
one2one_dist(pq_data, hrvar = "Organization", return = "table")
```

---

one2one_fizz *Distribution of Manager 1:1 Time (Fizzy Drink plot)*

Description

Analyze weekly Manager 1:1 Time distribution, and returns a 'fizzy' scatter plot by default. Additional options available to return a table with distribution elements.

Usage

```
one2one_fizz(data, hrvar = "Organization", mingroup = 5, return = "plot")
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present. If your data is already aggregated (e.g. one row per group), use the equivalent *_asis() variant of this function instead.
hrvar	String containing the name of the HR Variable by which to split metrics. Defaults to "Organization". To run the analysis on the total instead of splitting by an HR attribute, supply NULL (without quotes).
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
return	String specifying what to return. This must be one of the following strings: • "plot" • "table" See Value for more information.

Value

A different output is returned depending on the value passed to the `return` argument:

- `"plot"`: 'ggplot' object. A jittered scatter plot for the metric.
- `"table"`: data frame. A summary table for the metric.

See Also

Other Visualization: `afterhours_dist()`, `afterhours_fizz()`, `afterhours_line()`, `afterhours_rank()`, `afterhours_summary()`, `afterhours_trend()`, `collaboration_area()`, `collaboration_dist()`, `collaboration_fizz()`, `collaboration_line()`, `collaboration_rank()`, `collaboration_sum()`, `collaboration_trend()`, `create_bar()`, `create_bar_asis()`, `create_boxplot()`, `create_bubble()`, `create_dist()`, `create_fizz()`, `create_inc()`, `create_line()`, `create_line_asis()`, `create_period_scatter()`, `create_radar()`, `create_rank()`, `create_rogers()`, `create_sankey()`, `create_scatter()`, `create_stacked()`, `create_survival()`, `create_tracking()`, `create_trend()`, `email_dist()`, `email_fizz()`, `email_line()`, `email_rank()`, `email_summary()`, `email_trend()`, `external_dist()`, `external_fizz()`, `external_line()`, `external_rank()`, `external_sum()`, `hr_trend()`, `hrvar_count()`, `hrvar_trend()`, `keymetrics_scan()`, `meeting_dist()`, `meeting_fizz()`, `meeting_line()`, `meeting_rank()`, `meeting_summary()`, `meeting_trend()`, `one2one_dist()`, `one2one_freq()`, `one2one_line()`, `one2one_rank()`, `one2one_sum()`, `one2one_trend()`

Other Managerial Relations: `one2one_dist()`, `one2one_freq()`, `one2one_line()`, `one2one_rank()`, `one2one_sum()`, `one2one_trend()`

Examples

```
# Return plot
one2one_fizz(pq_data, hrvar = "Organization", return = "plot")

# Return a summary table
one2one_fizz(pq_data, hrvar = "Organization", return = "table")
```

---

one2one_freq *Frequency of Manager 1:1 Meetings as bar or 100% stacked bar chart*

Description

[Experimental]

This function calculates the average number of weeks (cadence) between of 1:1 meetings between an employee and their manager. Returns a distribution plot for typical cadence of 1:1 meetings. Additional options available to return a bar plot, tables, or a data frame with a cadence of 1 on 1 meetings metric.

Usage

```
one2one_freq(
  data,
  hrvar = "Organization",
  mingroup = 5,
  return = "plot",
  mode = "dist",
  sort_by = NULL
)
```

Arguments

`data`	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns `PersonId` and `MetricDate` present. If your data is already aggregated (e.g. one row per group), use the equivalent `*_asis()` variant of this function instead.
`hrvar`	String containing the name of the HR Variable by which to split metrics. Defaults to `"Organization"`. To run the analysis on the total instead of splitting by an HR attribute, supply `NULL` (without quotes).
`mingroup`	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
`return`	String specifying what to return. This must be one of the following strings: • `"plot"` • `"table"`
`mode`	String specifying what method to use. This must be one of the following strings: • `"dist"` • `"sum"`
`sort_by`	String to specify the bucket label to sort by. Defaults to `NULL` (no sorting).

Value

A different output is returned depending on the value passed to the `return` argument:

- `"plot"`: 'ggplot' object. A stacked bar plot for the metric.
- `"table"`: data frame. A summary table for the metric.

Distribution view

For this view, there are four categories of cadence:

- Weekly (once per week)
- Twice monthly or more (up to 3 weeks)
- Monthly (3 - 6 weeks)
- Every two months (6 - 10 weeks)

- Quarterly or less (> 10 weeks)

In the occasion there are zero 1:1 meetings with managers, this is included into the last category, i.e. 'Quarterly or less'. Note that when mode is set to "sum", these rows are simply excluded from the calculation.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Other Managerial Relations: one2one_dist(), one2one_fizz(), one2one_line(), one2one_rank(), one2one_sum(), one2one_trend()

Examples

```
# Return plot, mode dist
one2one_freq(pq_data, hrvar = "Organization", return = "plot", mode = "dist")

# Return plot, mode sum
one2one_freq(pq_data,
             hrvar = "Organization",
             return = "plot",
             mode = "sum")

# Return summary table
one2one_freq(pq_data, hrvar = "Organization", return = "table")
```

---

one2one_line *Manager 1:1 Time Trend - Line Chart*

Description

Provides a week by week view of 1:1 time with managers, visualised as line charts. By default returns a line chart for 1:1 meeting hours, with a separate panel per value in the HR attribute. Additional options available to return a summary table.

Usage

```
one2one_line(
  data,
  hrvar = "Organization",
  mingroup = 5,
  return = "plot",
  label = FALSE
)
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present. If your data is already aggregated (e.g. one row per group), use the equivalent *_asis() variant of this function instead.
hrvar	String containing the name of the HR Variable by which to split metrics. Defaults to "Organization". To run the analysis on the total instead of splitting by an HR attribute, supply NULL (without quotes).
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
return	String specifying what to return. This must be one of the following strings: • "plot" • "table" See Value for more information.
label	Logical value to determine whether to show data point labels on the plot. If TRUE, both geom_point() and geom_text() are added to display data labels rounded to 1 decimal place above each data point. Defaults to FALSE.

Details

Uses the metric Meeting_and_call_hours_with_manager_1_1.

Value

A different output is returned depending on the value passed to the return argument:

- "plot": 'ggplot' object. A faceted line plot for the metric.
- "table": data frame. A summary table for the metric.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(),

create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_rank(), one2one_sum(), one2one_trend()

Other Managerial Relations: one2one_dist(), one2one_fizz(), one2one_freq(), one2one_rank(), one2one_sum(), one2one_trend()

Examples

```
# Return a line plot
one2one_line(pq_data, hrvar = "LevelDesignation")

# Return summary table
one2one_line(pq_data, hrvar = "LevelDesignation", return = "table")
```

---

one2one_rank *Manager 1:1 Time Ranking*

Description

This function scans a standard query output for groups with high levels of 'Manager 1:1 Time'. Returns a plot by default, with an option to return a table with a all of groups (across multiple HR attributes) ranked by manager 1:1 time.

Usage

```
one2one_rank(
  data,
  hrvar = extract_hr(data),
  mingroup = 5,
  mode = "simple",
  plot_mode = 1,
  return = "plot"
)
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present. If your data is already aggregated (e.g. one row per group), use the equivalent *_asis() variant of this function instead.

`hrvar` String containing the name of the HR Variable by which to split metrics. Defaults to `"Organization"`. To run the analysis on the total instead of splitting by an HR attribute, supply `NULL` (without quotes).

`mingroup` Numeric value setting the privacy threshold / minimum group size. Defaults to 5.

`mode` String to specify calculation mode. Must be either:

- `"simple"`
- `"combine"`

`plot_mode` Numeric vector to determine which plot mode to return. Must be either `1` or `2`, and is only used when `return = "plot"`.

- `1`: Top and bottom five groups across the data population are highlighted
- `2`: Top and bottom groups *per* organizational attribute are highlighted

`return` String specifying what to return. This must be one of the following strings:

- `"plot"` (default)
- `"table"`

See `Value` for more information.

Details

Uses the metric `Meeting_and_call_hours_with_manager_1_1`. See `create_rank()` for applying the same analysis to a different metric.

Value

A different output is returned depending on the value passed to the `return` argument:

- `"plot"`: 'ggplot' object. A bubble plot where the x-axis represents the metric, the y-axis represents the HR attributes, and the size of the bubbles represent the size of the organizations. Note that there is no plot output if `mode` is set to `"combine"`.
- `"table"`: data frame. A summary table for the metric.

See Also

Other Visualization: `afterhours_dist()`, `afterhours_fizz()`, `afterhours_line()`, `afterhours_rank()`, `afterhours_summary()`, `afterhours_trend()`, `collaboration_area()`, `collaboration_dist()`, `collaboration_fizz()`, `collaboration_line()`, `collaboration_rank()`, `collaboration_sum()`, `collaboration_trend()`, `create_bar()`, `create_bar_asis()`, `create_boxplot()`, `create_bubble()`, `create_dist()`, `create_fizz()`, `create_inc()`, `create_line()`, `create_line_asis()`, `create_period_scatter()`, `create_radar()`, `create_rank()`, `create_rogers()`, `create_sankey()`, `create_scatter()`, `create_stacked()`, `create_survival()`, `create_tracking()`, `create_trend()`, `email_dist()`, `email_fizz()`, `email_line()`, `email_rank()`, `email_summary()`, `email_trend()`, `external_dist()`, `external_fizz()`, `external_line()`, `external_rank()`, `external_sum()`, `hr_trend()`, `hrvar_count()`, `hrvar_trend()`, `keymetrics_scan()`, `meeting_dist()`, `meeting_fizz()`, `meeting_line()`, `meeting_rank()`, `meeting_summary()`, `meeting_trend()`, `one2one_dist()`, `one2one_fizz()`, `one2one_freq()`, `one2one_line()`, `one2one_sum()`, `one2one_trend()`

Other Managerial Relations: `one2one_dist()`, `one2one_fizz()`, `one2one_freq()`, `one2one_line()`, `one2one_sum()`, `one2one_trend()`

Examples

```
# Return rank table
one2one_rank(data = pq_data, return = "table")

# Return plot
one2one_rank(data = pq_data, return = "plot")
```

`one2one_sum` *Manager 1:1 Time Summary*

Description

Provides an overview analysis of Manager 1:1 Time. Returns a bar plot showing average weekly minutes of Manager 1:1 Time by default. Additional options available to return a summary table.

Usage

```
one2one_sum(data, hrvar = "Organization", mingroup = 5, return = "plot")

one2one_summary(data, hrvar = "Organization", mingroup = 5, return = "plot")
```

Arguments

`data`	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns `PersonId` and `MetricDate` present. If your data is already aggregated (e.g. one row per group), use the equivalent `*_asis()` variant of this function instead.
`hrvar`	String containing the name of the HR Variable by which to split metrics. Defaults to `"Organization"`. To run the analysis on the total instead of splitting by an HR attribute, supply `NULL` (without quotes).
`mingroup`	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
`return`	String specifying what to return. This must be one of the following strings: • `"plot"` • `"table"` See `Value` for more information.

Value

A different output is returned depending on the value passed to the `return` argument:

- `"plot"`: 'ggplot' object. A bar plot for the metric.
- `"table"`: data frame. A summary table for the metric.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_trend()

Other Managerial Relations: one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_trend()

Examples

```
# Return a ggplot bar chart
one2one_sum(pq_data, hrvar = "LevelDesignation")

# Return a summary table
one2one_sum(pq_data, hrvar = "LevelDesignation", return = "table")
```

---

one2one_trend *Manager 1:1 Time Trend*

Description

Provides a week by week view of scheduled manager 1:1 Time. By default returns a week by week heatmap, highlighting the points in time with most activity. Additional options available to return a summary table.

Usage

```
one2one_trend(data, hrvar = "Organization", mingroup = 5, return = "plot")
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present. If your data is already aggregated (e.g. one row per group), use the equivalent *_asis() variant of this function instead.

hrvar	String containing the name of the HR Variable by which to split metrics. Defaults to "Organization". To run the analysis on the total instead of splitting by an HR attribute, supply NULL (without quotes).
mingroup	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
return	Character vector specifying what to return, defaults to "plot". Valid inputs are "plot" and "table".

Details

Uses the metric Meeting_and_call_hours_with_manager_1_1.

Value

Returns a 'ggplot' object by default, where 'plot' is passed in return. When 'table' is passed, a summary table is returned as a data frame.

See Also

Other Visualization: afterhours_dist(), afterhours_fizz(), afterhours_line(), afterhours_rank(), afterhours_summary(), afterhours_trend(), collaboration_area(), collaboration_dist(), collaboration_fizz(), collaboration_line(), collaboration_rank(), collaboration_sum(), collaboration_trend(), create_bar(), create_bar_asis(), create_boxplot(), create_bubble(), create_dist(), create_fizz(), create_inc(), create_line(), create_line_asis(), create_period_scatter(), create_radar(), create_rank(), create_rogers(), create_sankey(), create_scatter(), create_stacked(), create_survival(), create_tracking(), create_trend(), email_dist(), email_fizz(), email_line(), email_rank(), email_summary(), email_trend(), external_dist(), external_fizz(), external_line(), external_rank(), external_sum(), hr_trend(), hrvar_count(), hrvar_trend(), keymetrics_scan(), meeting_dist(), meeting_fizz(), meeting_line(), meeting_rank(), meeting_summary(), meeting_trend(), one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum()

Other Managerial Relations: one2one_dist(), one2one_fizz(), one2one_freq(), one2one_line(), one2one_rank(), one2one_sum()

Examples

```
# Run plot
one2one_trend(pq_data)

# Run table
one2one_trend(pq_data, hrvar = "LevelDesignation", return = "table")
```

p2p_data *Sample person-to-person dataset*

Description

A demo dataset representing a person-to-person query, structured as an edgelist. The identifier variable for each person is `PersonId`, where the variables have been prefixed with `PrimaryCollaborator_` and `SecondaryCollaborator_` to represent the direction of collaboration.

Usage

```
p2p_data
```

Format

A data frame with 11550 rows and 13 variables:

PrimaryCollaborator_PersonId

SecondaryCollaborator_PersonId

MetricDate

Diverse_tie_score

Diverse_tie_type

Strong_tie_score

Strong_tie_type

PrimaryCollaborator_Organization

SecondaryCollaborator_Organization

PrimaryCollaborator_LevelDesignation

SecondaryCollaborator_LevelDesignation

PrimaryCollaborator_FunctionType

SecondaryCollaborator_FunctionType ...

Value

data frame.

Source

https://analysis.insights.cloud.microsoft/analyst/analysis/

See Also

Other Data: `g2g_data`, `mt_data`, `p2p_data_sim()`, `pq_data`

Other Network: `g2g_data`, `network_g2g()`, `network_p2p()`, `network_summary()`, `p2p_data_sim()`

p2p_data_sim *Simulate a person-to-person query using a Watts-Strogatz model*

Description

Generate an person-to-person query / edgelist based on the graph according to the Watts-Strogatz small-world network model. Organizational data fields are also simulated for `Organization`, `LevelDesignation`, and `City`.

Usage

```
p2p_data_sim(dim = 1, size = 300, nei = 5, p = 0.05)
```

Arguments

`dim`	Integer constant, the dimension of the starting lattice.
`size`	Integer constant, the size of the lattice along each dimension.
`nei`	Integer constant, the neighborhood within which the vertices of the lattice will be connected.
`p`	Real constant between zero and one, the rewiring probability.

Details

This is a wrapper around `igraph::watts.strogatz.game()`. See igraph documentation for details on methodology. Loop edges and multiple edges are disabled. Size of the network can be changing the arguments `size` and `nei`.

Value

data frame with the same column structure as a person-to-person flexible query. This has an edgelist structure and can be used directly as an input to `network_p2p()`.

See Also

Other Data: `g2g_data`, `mt_data`, `p2p_data`, `pq_data`

Other Network: `g2g_data`, `network_g2g()`, `network_p2p()`, `network_summary()`, `p2p_data`

Examples

```
# Simulate a p2p dataset with 800 edges
p2p_data_sim(size = 200, nei = 4)
```

pad2 *Create the two-digit zero-padded format*

Description

Create the two-digit zero-padded format

Usage

```
pad2(x)
```

Arguments

x	numeric value or vector with maximum two characters.

Value

Numeric value containing two-digit zero-padded values.

pairwise_count *Perform a pairwise count of words by id*

Description

This is a 'data.table' implementation that mimics the output of `pairwise_count()` from 'widyr' to reduce package dependency. This is used internally within `tm_cooc()`.

Usage

```
pairwise_count(data, id = "line", word = "word")
```

Arguments

data	Data frame output from `tm_clean()`.
id	String to represent the id variable. Defaults to `"line"`.
word	String to represent the word variable. Defaults to `"word"`.

Value

data frame with the following columns representing a pairwise count:

- `"item1"`
- `"item2"`
- `"n"`

See Also

Other Support: any_idate(), camel_clean(), check_inputs(), cut_hour(), extract_date_range(), extract_hr(), heat_colours(), is_date_format(), maxmin(), read_preamble(), rgb2hex(), totals_bind(), totals_col(), tstamp(), us_to_space(), wrap()

Other Text-mining: meeting_tm_report(), tm_clean(), tm_cooc(), tm_freq(), tm_wordcloud()

Examples

```
td <- data.frame(line = c(1, 1, 2, 2),
                 word = c("work", "meeting", "catch", "up"))

pairwise_count(td, id = "line", word = "word")
```

---

plot_ts_us *Plot Usage Segments over time*

Description

Returns a vertical stacked bar plot that displays the proportion of the Usage Segments over time. This visualization helps to understand the distribution of user segments across different time periods. While a main use case is for Copilot metrics, this function can be applied to other metrics, such as 'Chats_sent'.

Usage

```
plot_ts_us(
  data,
  metric,
  cus,
  caption,
  threshold = NULL,
  width = NULL,
  max_window = NULL,
  power_thres = 15,
  version = NULL
)
```

Arguments

data	A data frame with a column containing the Usage Segments, denoted by cus. The data frame must also include a MetricDate column.
metric	A string representing the name of the metric column to be classified.
cus	A string representing the name of the column containing the usage segment classifications (e.g., "UsageSegments_12w").

caption	A string representing the caption for the plot. This is typically used to provide additional context or information about the visualization.
threshold	Numeric value specifying the minimum threshold for a valid count. Only used when creating custom parameter captions. Defaults to NULL.
width	Integer specifying the number of qualifying counts to consider for a habit. Only used when creating custom parameter captions. Defaults to NULL.
max_window	Integer specifying the maximum window to consider for a habit. Only used when creating custom parameter captions. Defaults to NULL.
power_thres	Numeric value specifying the minimum weekly average actions required to be classified as a 'Power User'. Defaults to 15.
version	A string indicating the version of the classification. Valid options are "12w", "4w", or NULL for custom parameters. Used to determine which definitions to show in the caption.

Value

A ggplot object representing the stacked bar plot of usage segments.

---

pq_data *Sample Person Query dataset*

Description

A dataset generated from a Person Query from Viva Insights.

Usage

```
pq_data
```

Format

A data frame with 6900 rows and 73 variables:

PersonId

MetricDate

Collaboration_hours

Copilot_actions_taken_in_Teams

Meeting_and_call_hours

Internal_network_size

Email_hours

Channel_message_posts

Conflicting_meeting_hours

Large_and_long_meeting_hours

External_collaboration_hours
Active_connected_hours
Meetings
After_hours_collaboration_hours
Call_hours
Calls
Channel_message_hours
Chat_hours
Collaboration_span
Emails_read
Emails_sent
External_network_size
Meeting_and_call_hours_with_manager
Meeting_and_call_hours_with_manager_1_1
Meeting_and_call_hours_with_skip_level
Meeting_hours
Multitasking_hours
Network_outside_company
Network_outside_organisation
Time_with_leadership
Unscheduled_call_hours
Weekend_collaboration_hours
Copilot_actions_taken_in_Copilot_chat__work_
Copilot_actions_taken_in_Excel
Copilot_actions_taken_in_Outlook
Copilot_actions_taken_in_Powerpoint
Copilot_actions_taken_in_Word
Days_of_active_Copilot_chat__work__usage
Days_of_active_Copilot_usage_in_Excel
Days_of_active_Copilot_usage_in_Loop
Days_of_active_Copilot_usage_in_OneNote
Days_of_active_Copilot_usage_in_Outlook
Days_of_active_Copilot_usage_in_Powerpoint
Days_of_active_Copilot_usage_in_Teams
Days_of_active_Copilot_usage_in_Word
Total_Copilot_active_days
Total_Copilot_enabled_days

Barriers_to_Execution

Change_Adaptation

Collaboration

Communication_Flow

Continuous_Improvement

eSat

Initiative

Manager_Recommend

Resources

Speak_My_Mind

Wellbeing

Work_Life_Balance

Workload

Create_Excel_formula_actions_taken_using_Copilot

Create_presentation_actions_taken_using_Copilot

Generate_email_draft_actions_taken_using_Copilot_in_Outlook

Summarise_chat_actions_taken_using_Copilot_in_Teams

Summarise_email_thread_actions_taken_using_Copilot_in_Outlook

Summarise_meeting_actions_taken_using_Copilot_in_Teams

Summarise_presentation_actions_taken_using_Copilot_in_PowerPoint

Summarise_Word_document_actions_taken_using_Copilot_in_Word

FunctionType

SupervisorIndicator

Level

Organization

LevelDesignation

Value

data frame.

Source

https://learn.microsoft.com/en-us/viva/insights/advanced/analyst/person-query/

See Also

Other Data: `g2g_data`, `mt_data`, `p2p_data`, `p2p_data_sim()`

prep_query *Prepare variable names and types in query data frame for analysis*

Description

For applying to data frames that are read into R using *any other method* other than `import_query()`, this function cleans variable names by replacing special characters and converting the relevant variable types so that they are compatible with the rest of the functions in **vivainsights**.

Usage

```
prep_query(data, convert_date = TRUE, date_format = "%m/%d/%Y")
```

Arguments

`data`	A Standard Person Query dataset in the form of a data frame. You should pass the data frame that is read into R using *any other method* other than `import_query()`, as `import_query()` automatically performs the same variable operations.
`convert_date`	Logical. Defaults to `TRUE`. When set to `TRUE`, any variable that matches true with `is_date_format()` gets converted to a Date variable. When set to `FALSE`, this step is skipped.
`date_format`	String specifying the date format for converting any variable that may be a date to a Date variable. Defaults to `"%m/%d/%Y"`.

Value

A `tibble` with the cleaned data frame is returned.

Examples

The following shows when and how to use `prep_query()`:

```
pq_df <- read.csv("path_to_query.csv")
cleaned_df <- pq_df |> prep_query()
```

You can then run checks to see that the variables are of the correct type:

```
dplyr::glimpse(cleaned_df)
```

See Also

Other Import and Export: `copy_df()`, `create_dt()`, `export()`, `import_query()`

read_preamble *Read preamble*

Description

Read in a preamble to be used within each individual reporting function. Reads from the Markdown file installed with the package.

Usage

```
read_preamble(path)
```

Arguments

path	Text string containing the path for the appropriate Markdown file.

Value

String containing the text read in from the specified Markdown file.

See Also

Other Support: any_idate(), camel_clean(), check_inputs(), cut_hour(), extract_date_range(), extract_hr(), heat_colours(), is_date_format(), maxmin(), pairwise_count(), rgb2hex(), totals_bind(), totals_col(), tstamp(), us_to_space(), wrap()

Other Reports: IV_report(), generate_report(), meeting_tm_report(), validation_report()

rgb2hex *Convert rgb to HEX code*

Description

Convert rgb to HEX code

Usage

```
rgb2hex(r, g, b)
```

Arguments

r, g, b	Values that correspond to the three RGB parameters

Value

Returns a string containing a HEX code.

See Also

Other Support: any_idate(), camel_clean(), check_inputs(), cut_hour(), extract_date_range(), extract_hr(), heat_colours(), is_date_format(), maxmin(), pairwise_count(), read_preamble(), totals_bind(), totals_col(), tstamp(), us_to_space(), wrap()

---

theme_wpa *Main theme for 'vivainsights' visualisations*

Description

A theme function applied to 'ggplot' visualisations in 'vivainsights'. Install and load 'extrafont' to use custom fonts for plotting.

Usage

```
theme_wpa(font_size = 12, font_family = "Segoe UI")
```

Arguments

font_size	Numeric value that prescribes the base font size for the plot. The text elements are defined relatively to this base font size. Defaults to 12.
font_family	Character value specifying the font family to be used in the plot. The default value is "Segoe UI". To ensure you can use this font, install and load 'extrafont' prior to plotting. There is an initialisation process that is described by: https://stackoverflow.com/questions/34522732/changing-fonts-in-ggplot2

Value

Returns a ggplot object with the applied theme.

See Also

Other Themes: theme_wpa_basic()

---

theme_wpa_basic *Basic theme for 'vivainsights' visualisations*

Description

A theme function applied to 'ggplot' visualisations in 'vivainsights'. Based on theme_wpa() but has no font requirements.

Usage

```
theme_wpa_basic(font_size = 12)
```

Arguments

font_size	Numeric value that prescribes the base font size for the plot. The text elements are defined relatively to this base font size. Defaults to 12.

Value

Returns a ggplot object with the applied theme.

See Also

Other Themes: theme_wpa()

---

tm_clean *Clean subject line text prior to analysis*

Description

This function processes the Subject column in a Meeting Query by applying tokenisation using tidytext::unnest_tokens(), and removing any stopwords supplied in a data frame (using the argument stopwords). This is a sub-function that feeds into tm_freq(), tm_cooc(), and tm_wordcloud(). The default is to return a data frame with tokenised counts of words or ngrams.

Usage

```
tm_clean(data, token = "words", stopwords = NULL, ...)
```

Arguments

data	A Meeting Query dataset in the form of a data frame.
token	A character vector accepting either "words" or "ngrams", determining type of tokenisation to return.
stopwords	A character vector OR a single-column data frame labelled 'word' containing custom stopwords to remove.
...	Additional parameters to pass to tidytext::unnest_tokens().

Value

data frame with two columns:

- line
- word

See Also

Other Text-mining: meeting_tm_report(), pairwise_count(), tm_cooc(), tm_freq(), tm_wordcloud()

Examples

```
# words
tm_clean(mt_data)

# ngrams
tm_clean(mt_data, token = "ngrams")
```

`tm_cooc` *Analyse word co-occurrence in subject lines and return a network plot*

Description

This function generates a word co-occurrence network plot, with options to return a table. This function is used within `meeting_tm_report()`.

Usage

```
tm_cooc(data, stopwords = NULL, seed = 100, return = "plot", lmult = 0.05)
```

Arguments

`data`	A Meeting Query dataset in the form of a data frame.
`stopwords`	A character vector OR a single-column data frame labelled `'word'` containing custom stopwords to remove.
`seed`	A numeric vector to set seed for random generation.
`return`	String specifying what to return. This must be one of the following strings: • `"plot"` • `"table"` See `Value` for more information.
`lmult`	A multiplier to adjust the line width in the output plot. Defaults to 0.05.

Details

This function uses `tm_clean()` as the underlying data wrangling function. There is an option to remove stopwords by passing a data frame into the `stopwords` argument.

Value

A different output is returned depending on the value passed to the `return` argument:

- `"plot"`: 'ggplot' and 'ggraph' object. A network plot.
- `"table"`: data frame. A summary table.

Example

The function can be run with subject lines from `mt_data`, as per below.

```
mt_data %>%
  tm_cooc(lmult = 0.01)
```

Author(s)

Carlos Morales carlos.morales@microsoft.com

See Also

Other Text-mining: `meeting_tm_report()`, `pairwise_count()`, `tm_clean()`, `tm_freq()`, `tm_wordcloud()`

Examples

```
# Demo using a subset of `mt_data`
```

`tm_freq` *Perform a Word or Ngram Frequency Analysis and return a Circular Bar Plot*

Description

Generate a circular bar plot with frequency of words / ngrams. This function is used within `meeting_tm_report()`.

Usage

```
tm_freq(data, token = "words", stopwords = NULL, keep = 100, return = "plot")
```

Arguments

`data`	A Meeting Query dataset in the form of a data frame.
`token`	A character vector accepting either `"words"` or `"ngram"`, determining type of tokenisation to return.
`stopwords`	A character vector OR a single-column data frame labelled `'word'` containing custom stopwords to remove.
`keep`	A numeric vector specifying maximum number of words to keep.
`return`	String specifying what to return. This must be one of the following strings: • `"plot"` • `"table"` See `Value` for more information.

Details

This function uses `tm_clean()` as the underlying data wrangling function. There is an option to remove stopwords by passing a data frame into the `stopwords` argument.

Value

A different output is returned depending on the value passed to the `return` argument:

- `"plot"`: 'ggplot' object. A circular bar plot.
- `"table"`: data frame. A summary table.

See Also

Other Text-mining: `meeting_tm_report()`, `pairwise_count()`, `tm_clean()`, `tm_cooc()`, `tm_wordcloud()`

Examples

```
# circular network plot with words
tm_freq(mt_data, token = "words")

# circular network plot with ngrams
tm_freq(mt_data, token = "ngrams")

# summary table of text frequency
tm_freq(mt_data, token = "words", return = "table")
```

---

tm_wordcloud *Generate a wordcloud with meeting subject lines*

Description

Generate a wordcloud with the meeting query. This is a sub-function that feeds into `meeting_tm_report()`.

Usage

```
tm_wordcloud(
  data,
  stopwords = NULL,
  seed = 100,
  keep = 100,
  return = "plot",
  ...
)
```

Arguments

data	A Meeting Query dataset in the form of a data frame.
stopwords	A character vector OR a single-column data frame labelled 'word' containing custom stopwords to remove.
seed	A numeric vector to set seed for random generation.
keep	A numeric vector specifying maximum number of words to keep.
return	String specifying what to return. This must be one of the following strings: • "plot" • "table" See Value for more information.
...	Additional parameters to be passed to ggwordcloud::geom_text_wordcloud()

Details

Uses the 'ggwordcloud' package for the underlying implementation, thus returning a 'ggplot' object. Additional layers can be added onto the plot using a ggplot + syntax. The recommendation is not to return over 100 words in a word cloud.

This function uses tm_clean() as the underlying data wrangling function. There is an option to remove stopwords by passing a data frame into the stopwords argument.

Value

A different output is returned depending on the value passed to the return argument:

- "plot": 'ggplot' object containing a word cloud.
- "table": data frame returning the data used to generate the word cloud.

See Also

Other Text-mining: meeting_tm_report(), pairwise_count(), tm_clean(), tm_cooc(), tm_freq()

Examples

```
tm_wordcloud(mt_data, keep = 30)

# Removing stopwords
tm_wordcloud(mt_data, keep = 30, stopwords = c("weekly", "update"))
```

totals_bind *Row-bind an identical data frame for computing grouped totals*

Description

Row-bind an identical data frame and impute a specific column with the `target_value`, which defaults as "Total". The purpose of this is to enable to creation of summary tables with a calculated "Total" row. See example below on usage.

Usage

```
totals_bind(data, target_col, target_value = "Total")
```

Arguments

`data`	data frame
`target_col`	Character value of the column in which to impute `"Total"`. This is usually the intended grouping column.
`target_value`	Character value to impute in the new data frame to row-bind. Defaults to `"Total"`.

Value

data frame with twice the number of rows of the input data frame, where half of those rows will have the `target_col` column imputed with the value from `target_value`.

See Also

Other Support: `any_idate()`, `camel_clean()`, `check_inputs()`, `cut_hour()`, `extract_date_range()`, `extract_hr()`, `heat_colours()`, `is_date_format()`, `maxmin()`, `pairwise_count()`, `read_preamble()`, `rgb2hex()`, `totals_col()`, `tstamp()`, `us_to_space()`, `wrap()`

Examples

```
pq_data %>%
  totals_bind(target_col = "LevelDesignation", target_value = "Total") %>%
  create_bar(hrvar = "LevelDesignation", metric = "Email_hours", return = "table")
```

totals_col *Fabricate a 'Total' HR variable*

Description

Create a 'Total' column of character type comprising exactly of one unique value. This is a convenience function for returning a no-HR attribute view when NULL is supplied to the hrvar argument in functions.

Usage

```
totals_col(data, total_value = "Total")
```

Arguments

data	data frame
total_value	Character value defining the name and the value of the "Total" column. Defaults to "Total". An error is returned if an existing variable has the same name as the supplied value.

Value

data frame containing an additional 'Total' column on top of the input data frame.

See Also

Other Support: any_idate(), camel_clean(), check_inputs(), cut_hour(), extract_date_range(), extract_hr(), heat_colours(), is_date_format(), maxmin(), pairwise_count(), read_preamble(), rgb2hex(), totals_bind(), tstamp(), us_to_space(), wrap()

Examples

```
# Create a visual without HR attribute breaks
pq_data %>%
  totals_col() %>%
  create_fizz(hrvar = "Total", metric = "Email_hours")
```

`track_HR_change` *Sankey chart of organizational movement between HR attributes and missing values (outside company move) (Data Overview)*

Description

Creates a list of everyone at a specified start date and a specified end date then aggregates up people who have moved between organizations between this to points of time and visualizes the move through a sankey chart.

Through this chart you can see:

- The HR attribute/orgs that have the highest move out
- The HR attribute/orgs that have the highest move in
- The number of people that do not have that HR attribute or if they are no longer in the system

Usage

```
track_HR_change(
  data,
  start_date = min(data$MetricDate),
  end_date = max(data$MetricDate),
  hrvar = "Organization",
  mingroup = 5,
  return = "plot",
  NA_replacement = "Out of Company"
)
```

Arguments

`data`	A Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns `PersonId` and `MetricDate` present.
`start_date`	A start date to compare changes. See `end_date`.
`end_date`	An end date to compare changes. See `start_date`.
`hrvar`	HR Variable by which to compare changes between, defaults to `"Organization"` but accepts any character vector, e.g. `"LevelDesignation"`
`mingroup`	Numeric value setting the privacy threshold / minimum group size. Defaults to 5.
`return`	Character vector specifying what to return, defaults to `"plot"`. Valid inputs are `"plot"` and `"table"`.
`NA_replacement`	Character replacement for NA defaults to "out of company"

Value

Returns a 'NetworkD3' object by default, where 'plot' is passed in `return`. When 'table' is passed, a summary table is returned as a data frame.

Author(s)

Tannaz Sattari Tabrizi Tannaz.Sattari@microsoft.com

See Also

Other Data Validation: check_query(), extract_hr(), flag_ch_ratio(), flag_em_ratio(), flag_extreme(), flag_outlooktime(), hr_trend(), hrvar_count(), hrvar_count_all(), hrvar_trend(), identify_churn(), identify_holidayweeks(), identify_inactiveweeks(), identify_nkw(), identify_outlier(), identify_privacythreshold(), identify_shifts(), identify_tenure(), validation_report()

Examples

```
pq_data %>% track_HR_change()
```

---

tstamp *Generate a time stamp*

Description

This function generates a time stamp of the format 'yymmdd_hhmmss'. This is a support function and is not intended for direct use.

Usage

```
tstamp()
```

Value

String containing the timestamp in the format 'yymmdd_hhmmss'.

See Also

Other Support: any_idate(), camel_clean(), check_inputs(), cut_hour(), extract_date_range(), extract_hr(), heat_colours(), is_date_format(), maxmin(), pairwise_count(), read_preamble(), rgb2hex(), totals_bind(), totals_col(), us_to_space(), wrap()

us_to_space *Replace underscore with space*

Description

Convenience function to convert underscores to space

Usage

```
us_to_space(x)
```

Arguments

x	String to replace all occurrences of _ with a single space

Value

Character vector containing the modified string.

See Also

Other Support: any_idate(), camel_clean(), check_inputs(), cut_hour(), extract_date_range(), extract_hr(), heat_colours(), is_date_format(), maxmin(), pairwise_count(), read_preamble(), rgb2hex(), totals_bind(), totals_col(), tstamp(), wrap()

Examples

```
us_to_space("Meeting_and_call_hours_with_manager_1_on_1")
```

validation_report *Generate a Data Validation report in HTML*

Description

The function generates an interactive HTML report using Standard Person Query data as an input. The report contains checks on Viva Insights query outputs to provide diagnostic information for the Analyst prior to analysis.

An additional Standard Meeting Query can be provided to perform meeting subject line related checks. This is optional and the validation report can be run without it.

Usage

```
validation_report(
  data,
  meeting_data = NULL,
  hrvar = "Organization",
  path = "validation report",
  hrvar_threshold = 150,
  timestamp = TRUE
)
```

Arguments

data	A Standard Person Query dataset in the form of a data frame. This must be a **panel dataset** where each row represents one employee per time period, with the columns PersonId and MetricDate present.
meeting_data	An optional Meeting Query dataset in the form of a data frame.
hrvar	HR Variable by which to split metrics, defaults to "Organization" but accepts any character vector, e.g. "Organization"
path	Pass the file path and the desired file name, *excluding the file extension*.
hrvar_threshold	Numeric value determining the maximum number of unique values to be allowed to qualify as a HR variable. This is passed directly to the threshold argument within hrvar_count_all().
timestamp	Logical vector specifying whether to include a timestamp in the file name. Defaults to TRUE.

Details

For your input to data or meeting_data, please use the function vivainsights::import_query() to import your csv query files into R. This function will standardize format and prepare the data as input for this report.

For most variables, a note is returned in-line instead of an error if the variable is not available.

Value

An HTML report with the same file name as specified in the arguments is generated in the working directory. No outputs are directly returned by the function.

Checking functions within validation_report()

- check_query()
- flag_ch_ratio()
- hrvar_count_all()
- identify_privacythreshold()
- identify_nkw()
- identify_holidayweeks()

- subject_validate() (available in 'wpa')
- identify_tenure()
- flag_outlooktime()
- identify_shifts()
- track_HR_change()

You can browse each individual function for details on calculations.

Creating a report

Below is an example on how to run the report.

```
validation_report(pq_data,
                  hrvar = "Organization")
```

See Also

Other Reports: IV_report(), generate_report(), meeting_tm_report(), read_preamble()

Other Data Validation: check_query(), extract_hr(), flag_ch_ratio(), flag_em_ratio(), flag_extreme(), flag_outlooktime(), hr_trend(), hrvar_count(), hrvar_count_all(), hrvar_trend(), identify_churn(), identify_holidayweeks(), identify_inactiveweeks(), identify_nkw(), identify_outlier(), identify_privacythreshold(), identify_shifts(), identify_tenure(), track_HR_change()

---

wrap *Add a character at the start and end of a character string*

Description

This function adds a character at the start and end of a character string, where the default behaviour is to add a double quote.

Usage

```
wrap(string, wrapper = "\"")
```

Arguments

string	Character string to be wrapped around
wrapper	Character to wrap around string

Value

Character vector containing the modified string.

See Also

Other Support: any_idate(), camel_clean(), check_inputs(), cut_hour(), extract_date_range(), extract_hr(), heat_colours(), is_date_format(), maxmin(), pairwise_count(), read_preamble(), rgb2hex(), totals_bind(), totals_col(), tstamp(), us_to_space()

---

wrap_text *Wrap text based on character threshold*

Description

Wrap text in visualizations according to a preset character threshold. The next space in the string is replaced with \n, which will render as next line in plots and messages.

Usage

```
wrap_text(x, threshold = 15)
```

Arguments

x	String to wrap text
threshold	Numeric, defaults to 15. Number of character units by which the next space would be replaced with \n to move text to next line.

Value

String output representing a processed version of x, with spaces replaced by \n.

Examples

```
wrapped <- wrap_text(
  "The total entropy of an isolated system can never decrease."
  )
message(wrapped)
```

---

xicor *Calculate Chatterjee's Rank Correlation Coefficient*

Description

This function calculates Chatterjee's rank correlation coefficient, which measures the association between two variables. It is particularly useful for identifying monotonic relationships between variables, even if they are not linear.

Usage

```
xicor(x, y, ties = FALSE)
```

Arguments

x	A numeric vector representing the independent variable.
y	A numeric vector representing the dependent variable.
ties	A logical value indicating whether to handle ties in the data. Default is FALSE. If ties = TRUE, the function adjusts for tied ranks (repeated values in the data). This is important when there are many tied values in either x or y, as it ensures accurate calculation by considering the maximum rank for tied observations. If ties = FALSE, the function assumes that there are no ties, or that ties can be handled without additional computational effort. This option can offer better performance when ties are rare or absent.

Details

Unlike Pearson's correlation (which measures linear relationships), Chatterjee's coefficient can handle non-linear monotonic relationships. It is robust to outliers and can handle tied ranks, making it versatile for datasets with ordinal data or tied ranks. This makes it a valuable alternative to Spearman's and Kendall's correlations, especially when the data may not meet the assumptions required by these methods.

By default, ties = FALSE is set to prioritize computational efficiency, as handling ties requires additional processing. In cases where ties are present or likely (such as when working with ordinal or categorical data), it is recommended to set ties = TRUE.

Value

A numeric value representing Chatterjee's rank correlation coefficient.

Examples

```
xicor(x = pq_data$Collaboration_hours, y = pq_data$Internal_network_size, ties = TRUE)
xicor(x = pq_data$Collaboration_hours, y = pq_data$Internal_network_size, ties = FALSE)
```

Index