Package 'headliner'

December 20, 2022

Title Compose Sentences to Describe Comparisons

Version 0.0.3

Description Create dynamic, data-driven text. Given two values, a list of talking points is generated and can be combined using string interpolation. Based on the 'glue' package.

License MIT + file LICENSE

URL https://rjake.github.io/headliner/, https://github.com/rjake/headliner/

BugReports https://github.com/rjake/headliner/issues/

Depends R (>= 4.1)

Imports dplyr, glue, lubridate, purrr, rlang, tibble, tidyr

Suggests ggplot2, knitr, nycflights13, rmarkdown, testthat

VignetteBuilder knitr

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.2.0

NeedsCompilation no

Author Jake Riley [aut, cre]

Maintainer Jake Riley <rjake@sas.upenn.edu>

Repository CRAN

Date/Publication 2022-12-20 04:00:02 UTC

R topics documented:

add_article 2
add_date_columns 3
add_headline_column 4
compare_conditions 6

compare_values 8
demo_data 10
headline 11
pixar_films 14
plural_phrasing 16
trend_terms 17
view_list 18

Index **20**

`add_article` *Append a/an to word*

Description

Append a/an to word

Usage

```
add_article(x)
```

Arguments

`x`	string or numeric value

Details

This function uses crude logic to append 'a' or 'an' to numbers and phrases.

- words that start with aeiou
- negative numbers always start with 'a', ex: 'a -3' or 'a -8'
- decimals always start with 'a' ex: 0.4 is usually pronounced 'a zero point four' or 'a point four'
- numbers starting with 8 are always 'an'
- if the integer that comes after thousand or million is 11 or 18 then 'an'
  - 18,000 becomes 18 and that becomes 'an 18'
- if the integer that comes after thousand or million is in 1, 10, 12, 13, 14, 15, 16, 17, 19 then 'a'
  - 15,500 becomes 15 and that becomes 'a 15'
- otherwise 'a'

Value

Returns a vector the same length as the input.

Examples

```
add_article("increase")

add_article("decrease")

add_article(c(1, 8, 10, 11, 18, 20, 80))

add_article(18123)

stats::setNames(
  add_article(1.8 * 10^(1:7)),
  prettyNum(1.8 * 10^(1:7), big.mark = ",")
)
```

---

add_date_columns *Add columns with date calculations based on reference date*

Description

Using a reference date (defaults to current date), columns are appended to the data set describing the number of days, weeks, months, quarters, calendar years and fiscal years since the reference date. If the new columns share names with an existing column, the function will show a warning.

Usage

```
add_date_columns(
  df,
  date_col,
  ref_date = Sys.Date(),
  fiscal_year_offset = 6,
  week_start = 1,
  drop = FALSE
)
```

Arguments

df	data frame
date_col	column with class of 'date'
ref_date	reference date for calculations, defaults to current date
fiscal_year_offset	the number of months to offset date, if fiscal year ends in June, use 6
week_start	integer for start of week where Monday = 1 and Sunday = 7
drop	some of the generated fields may match the input data frame. When TRUE, the original columns will be removed and replaced with the new field of the same name. Otherwise, columns with the same name will be appended with a '1'

Value

Returns a data frame with columns appended to describe date distances from a reference date.

Examples

```
demo_data() |>
  add_date_columns(date_col = date)

# if columns overlap, you will see a warning
demo_data() |>
  dplyr::mutate(week = 1) |>
  add_date_columns(date_col = date)

# to drop the old column and keep the new column use `drop = TRUE`
demo_data() |>
  dplyr::mutate(week = 1) |>
  add_date_columns(date_col = date, drop = TRUE)
```

---

add_headline_column *Add column of headlines*

Description

This works similar to `headline()` but acts on and returns a data frame.

Usage

```
add_headline_column(
  df,
  x,
  y,
  headline = "{trend} of {delta} ({orig_values})",
  ...,
  .name = "headline",
  if_match = "There was no difference",
  trend_phrases = headliner::trend_terms(),
  plural_phrases = NULL,
  orig_values = "{x} vs. {y}",
  n_decimal = 1,
  round_all = TRUE,
  multiplier = 1,
  return_cols = .name
)
```

Arguments

df	data frame, must be a single row
x	a numeric value to compare to the reference value of 'y'
y	a numeric value to act as a control for the 'x' value
headline	a string to format the final output. Uses glue syntax
...	arguments passed to glue_data
.name	string value for the name of the new column to create
if_match	string to display if numbers match, uses glue syntax
trend_phrases	list of values to use for when x is more than y or x is less than y. You can pass it just trend_terms (the default) and call the result with "...{trend}..." or pass is a named list (see examples)
plural_phrases	named list of values to use when difference (delta) is singular (delta = 1) or plural (delta != 1)
orig_values	a string using glue syntax. example: ({x} vs {y})
n_decimal	numeric value to limit the number of decimal places in the returned values.
round_all	logical value to indicate if all values should be rounded. When FALSE, the values will return with no modification. When TRUE (default) all values will be round to the length specified by 'n_decimal'.
multiplier	number indicating the scaling factor. When multiplier = 1 (default), 0.25 will return 0.25. When multiplier = 100, 0.25 will return 25.
return_cols	arguments that can be passed to select, ex: c("a", "b"), starts_with, etc.

Details

What is nice about this function is you can return some of the "talking points" used in the headline calculation. For example, if you want to find the most extreme headlines, you can use add_headline_column(..., return_cols = delta) This will bring back a headline column as well as the delta talking point (the absolute difference between x and y). With this result, you can sort in descending order and filter for the biggest difference.

Value

Returns the original data frame with columns appended.

Examples

```
# You can use 'add_headline_column()' to reference values in an existing data set.
# Here is an example comparing the box office sales of different Pixar films
head(pixar_films) |>
  dplyr::select(film, bo_domestic, bo_intl) |>
  add_headline_column(
    x = bo_domestic,
    y = bo_intl,
    headline = "{film} was ${delta}M higher {trend} (${x}M vs ${y}M)",
```

```
    trend_phrases = trend_terms(more = "domestically", less = "internationally")
  ) |>
  knitr::kable("pandoc")

# You can also use 'return_cols' to return any and all "talking points".
# You can use tidyselect helpers like 'starts_with("delta")' or
# 'everything()'. In this example, I returned the 'raw_delta' & 'trend' columns
# and then identified the records at the extremes
pixar_films |>
  dplyr::select(film, bo_domestic, bo_intl) |>
  add_headline_column(
    x = bo_domestic,
    y = bo_intl,
    headline = "${delta}M {trend} (${x}M vs ${y}M)",
    trend_phrases = trend_terms(more = "higher", less = "lower"),
    return_cols = c(raw_delta, trend)
  ) |>
  dplyr::filter(raw_delta %in% range(raw_delta)) |>
  knitr::kable("pandoc")
```

`compare_conditions` *Compare two conditions within a data frame*

Description

Using logic that `filter` can interpret, `compare_conditions()` will summarize the data aggregating condition `x` and condition `y`

Usage

```
compare_conditions(df, x, y, .cols = everything(), .fns = lst(mean))
```

Arguments

`df`	data frame
`x`	condition for comparison, same criteria you would use in 'dplyr::filter', used in contrast to the reference group 'y'
`y`	condition for comparison, same criteria you would use in 'dplyr::filter', used in contrast to the reference group 'x'
`.cols`	columns to use in comparison
`.fns`	named list of the functions to use, ex: list(avg = mean, sd = sd) 'purrr' style phrases are also supported like list(mean = ~mean(.x, na.rm = TRUE), sd = sd) and dplyr::lst(mean, sd) will create a list(mean = mean, sd = sd)

Details

`compare_conditions()` passes its arguments to `across`. The `.cols` and `.fns` work the same. For clarity, it is helpful to use the `lst` function for the `.fns` parameter. Using `compare_conditions(..., .cols = my_var, .fns = lst(mean, sd))` will return the values `mean_my_var_x`, `mean_my_var_y`, `sd_my_var_x` and `sd_my_var_x`

Value

Returns a data frame that is either 1 row, or if grouped, 1 row per group.

Examples

```
# compare_conditions works similar to dplyr::across()
pixar_films |>
  compare_conditions(
    x = (rating == "G"),
    y = (rating == "PG"),
    .cols = rotten_tomatoes
  )


# because data frames are just fancy lists, you pass the result to headline_list()
pixar_films |>
  compare_conditions(
    x = (rating == "G"),
    y = (rating == "PG"),
    .cols = rotten_tomatoes
  ) |>
 headline_list("a difference of {delta} points")


 # you can return multiple objects to compare
 # 'view_List()' is a helper to see list objects in a compact way
 pixar_films |>
  compare_conditions(
    x = (rating == "G"),
    y = (rating == "PG"),
    .cols = c(rotten_tomatoes, metacritic),
    .fns = dplyr::lst(mean, sd)
  ) |>
  view_list()


# you can use any of the `tidyselect` helpers
pixar_films |>
  compare_conditions(
    x = (rating == "G"),
    y = (rating == "PG"),
    .cols = dplyr::starts_with("bo_")
  )
```

```
# if you want to compare x to the overall average, use y = TRUE
pixar_films |>
  compare_conditions(
    x = (rating == "G"),
    y = TRUE,
    .cols = rotten_tomatoes
  )


# to get the # of observations use length() instead of n()
# note: don't pass the parentheses
pixar_films |>
  compare_conditions(
    x = (rating == "G"),
    y = (rating == "PG"),
    .cols = rotten_tomatoes, # can put anything here really
    .fns = list(n = length)
  )


# you can also use purrr-style lambdas
pixar_films |>
  compare_conditions(
    x = (rating == "G"),
    y = (rating == "PG"),
    .cols = rotten_tomatoes,
    .fns = list(avg = ~ sum(.x) / length(.x))
  )

# you can compare categorical data with functions like dplyr::n_distinct()
pixar_films |>
  compare_conditions(
    x = (rating == "G"),
    y = (rating == "PG"),
    .cols = film,
    .fns = list(distinct = dplyr::n_distinct)
  )
```

`compare_values` *Compare two values and get talking points*

Description

A function to create "talking points" that describes the difference between two values.

Usage

```
compare_values(
  x,
```

```
  y,
  trend_phrases = headliner::trend_terms(),
  orig_values = "{x} vs. {y}",
  plural_phrases = NULL,
  n_decimal = 1,
  round_all = TRUE,
  multiplier = 1,
  check_rounding = TRUE
)
```

Arguments

x	a numeric value to compare to the reference value of 'y'
y	a numeric value to act as a control for the 'x' value
trend_phrases	list of values to use for when x is more than y or x is less than y. You can pass it just trend_terms (the default) and call the result with "...{trend}..." or pass is a named list (see examples)
orig_values	a string using glue syntax. example: ({x} vs {y})
plural_phrases	named list of values to use when difference (delta) is singular (delta = 1) or plural (delta != 1)
n_decimal	numeric value to limit the number of decimal places in the returned values.
round_all	logical value to indicate if all values should be rounded. When FALSE, the values will return with no modification. When TRUE (default) all values will be round to the length specified by 'n_decimal'.
multiplier	number indicating the scaling factor. When multiplier = 1 (default), 0.25 will return 0.25. When multiplier = 100, 0.25 will return 25.
check_rounding	when TRUE (default) inputs will be checked to confirm if a difference of zero may be due to rounding. Ex: 0.16 and 0.24 with 'n_decimal = 1' will both return 0.2. Because this will show no difference, a message will be displayed

Details

Given `compare_values(x = 8, y = 10)` the following items will be returned in the list:

item	value	description
x	2	original x value to compare against y
y	10	original y value
delta	8	absolute difference between x & y
delta_p	80	% difference between x & y
article_delta	"an 8"	delta with the article included
article_delta_p	"an 80"	delta_p with the article included
raw_delta	-8	true difference between x & y
raw_delta_p	-80	true % difference between x & y
article_raw_delta	"a -8"	raw_delta with the article
article_raw_delta_p	"a -80"	raw_delta_p with the article
sign	-1	the direction, 1 (increase), -1 (decrease), or 0 (no change)
orig_values	"2 vs. 10"	shorthand for {x} vs {y}

trend	"decrease"	influenced by the values in trend_phrases argument

Value

compare_values() returns a list object that can be used with glue syntax

See Also

headline(), trend_terms(), plural_phrasing() and view_list()

Examples

```
# the values can be manually entered

compare_values(10, 8) |> head(2)
# percent difference (10-8)/8
compare_values(10, 8)$delta_p

# trend_phrases returns an object called trend if nothing is passed
compare_values(10, 8)$trend

# or if one argument is passed using trend_terms()
compare_values(10, 8, trend_phrases = trend_terms(more = "higher"))$trend

# if a named list is used, the objects are called by their names
compare_values(
  10, 8,
  trend_phrases = list(
    more = trend_terms(),
    higher = trend_terms("higher", "lower")
  )
)$higher

# a phrase about the comparison can be edited by providing glue syntax
# 'c' = the 'compare' value, 'r' = 'reference'
compare_values(10, 8, orig_values = "{x} to {y} people")$orig_values

# you can also adjust the rounding, although the default is 1
compare_values(0.1234, 0.4321)$orig_values
compare_values(0.1234, 0.4321, n_decimal = 3)$orig_values
# or add a multiplier
compare_values(0.1234, 0.4321, multiplier = 100)$orig_values
```

---

demo_data *Small data set referencing the current date*

Description

Small data set referencing the current date

Usage

```
demo_data(n = 10, by = "-2 month")
```

Arguments

n	number of rows to return
by	string indicating the unit of time between dates in `seq.Date(..., by = )`

Value

Returns a data frame of size `n`.

Examples

```
demo_data()

demo_data(n = 8, by = "1 day")
```

---

headline *Compose phrases that describe differences in the data*

Description

Given two values, `headline()` will use glue syntax to string together "talking points". For example `headline(8, 10)` will describe a difference of 2 and can be expressed as `headline(8, 10, headline = "changed by {delta} ({raw_delta_p}%)")`. This returns "changed by 2 (-20%)".

Usage

```
headline(
  x,
  y,
  headline = "{trend} of {delta} ({orig_values})",
  ...,
  if_match = "There was no difference",
  trend_phrases = headliner::trend_terms(),
  plural_phrases = NULL,
  orig_values = "{x} vs. {y}",
  n_decimal = 1,
  round_all = TRUE,
  multiplier = 1,
  return_data = FALSE
)

headline_list(
  l,
  headline = "{trend} of {delta} ({orig_values})",
```

```
  x,
  y,
  ...,
  if_match = "There was no difference.",
  trend_phrases = headliner::trend_terms(),
  plural_phrases = NULL,
  orig_values = "{x} vs. {y}",
  n_decimal = 1,
  round_all = TRUE,
  multiplier = 1,
  return_data = FALSE
)
```

Arguments

x	a numeric value to compare to the reference value of 'y'
y	a numeric value to act as a control for the 'x' value
headline	a string to format the final output. Uses glue syntax
...	arguments passed to glue_data
if_match	string to display if numbers match, uses glue syntax
trend_phrases	list of values to use for when x is more than y or x is less than y. You can pass it just trend_terms (the default) and call the result with "...{trend}..." or pass is a named list (see examples)
plural_phrases	named list of values to use when difference (delta) is singular (delta = 1) or plural (delta != 1)
orig_values	a string using glue syntax. example: ({x} vs {y})
n_decimal	numeric value to limit the number of decimal places in the returned values.
round_all	logical value to indicate if all values should be rounded. When FALSE, the values will return with no modification. When TRUE (default) all values will be round to the length specified by 'n_decimal'.
multiplier	number indicating the scaling factor. When multiplier = 1 (default), 0.25 will return 0.25. When multiplier = 100, 0.25 will return 25.
return_data	logical to indicate whether function should return the talking points used to compose the headline
l	a list with values to compare, if named, can call by name

Details

headline() relies heavily on glue_data. Objects can be combined into a headline using the following search path: If given

```
delta <- 123
headline(1, 3, delta = "abc")

## decrease of abc (1 vs. 3)
```

delta is one of the "talking points" from compare_values() and would usually return "2" but because we passed the named variable delta = "none", headline() (really glue_data) will look first at the named variables, then at the result of compare_values() then in the global environment. So in the example above, the output will return "decrease of xxxxxx (1 vs. 3)"

Value

Returns a character vector the same length as the input,

See Also

compare_values(), trend_terms(), and add_article()

Examples

```
# values can be manually entered, some headlines are provided by default
headline(10, 8)
headline(8, 10)
headline(1:3, 3:1)

# most likely you'll edit the headline by hand
headline(
  x = 10,
  y = 8,
  headline = "There was a ${delta} {trend} vs last year"
)

# you can also adjust the phrasing of higher/lower values
headline(
  x = 10,
  y = 8,
  headline = "Group A was {trend} by {delta_p}%.",
  trend_phrases = trend_terms(more = "higher", less = "lower")
 )

# a phrase about the comparion can be edited by providing glue syntax
# 'c' = the 'compare' value, 'r' = 'reference'
headline(10, 8, orig_values = "{x} to {y} people")

# you can also add phrases for when the difference = 1 or not
headline(
  x = 10,
  y = 8,
  plural_phrases = list(
    were = plural_phrasing(single = "was", multi = "were"),
    people = plural_phrasing(single = "person", multi = "people")
  ),
  headline = "there {were} {delta} {people}"
)

# you can also adjust the rounding, the default is 1
headline(0.1234, 0.4321)
```

```
headline(0.1234, 0.4321, n_decimal = 3)
# or use a multiplier
headline(0.1234, 0.4321, multiplier = 100)

# there are many components you can assemble
headline(
  x = 16,
  y = 8,
  headline = "there was {article_delta_p}% {trend}, \\
  {add_article(trend)} of {delta} ({orig_values})"
)


# compare_conditions() produces a one-row data frame that can be
# passed to headline_list()
pixar_films |>
  compare_conditions(
    x = (rating == "G"),
    y = (rating == "PG"),
    rotten_tomatoes
  ) |>
  headline_list(
    headline = "On average, G-rated films score {delta} points {trend} than \\
    PG films on Rotten Tomatoes",
    trend_phrases = trend_terms(more = "higher", less = "lower")
  )

# if you have more than 2 list items, you can specify them by name
list(
  x = 1,
  y = 2,
  z = 3
 ) |>
  headline_list(
    x = x,
    y = z
  )
```

`pixar_films` *This data comes from* R*hrefhttps://github.com/erictleung/pixarfilms/*`pixarfilms` *package by Eric Leung (2022)*

Description

The data has box office sales, audience ratings, and release dates for each Pixar film

Usage

```
pixar_films
```

Format

A tibble with 22 rows and 10 columns:

order order of release

film name of film

release_date date film premiered

year the year the film premiered

run_time film length in minutes

film_rating rating based on Motion Picture Association (MPA) film rating system

rotten_tomatoes score from the American review-aggregation website Rotten Tomatoes; scored out of 100

metacritic score from Metacritic where scores are weighted average of reviews; scored out of 100

bo_domestic box office gross amount in U.S. dollars (millions) for U.S. and Canada

bo_intl box office gross amount in U.S. dollars (millions) for other territories

Examples

```
pixar_films

library(ggplot2)

headline(
  x = min(pixar_films$run_time),
  y = max(pixar_films$run_time),
  headline =
   "The shortest film was {delta} minutes less than the longest film ({orig_values} minutes)"
)

ggplot(pixar_films, aes(bo_intl, rating)) +
  geom_boxplot() +
  xlim(0, NA) +
  labs(title = "International Box Office by MPA Rating")


ggplot(pixar_films, aes(release_date, run_time)) +
  geom_line() +
  geom_point() +
  ylim(0, NA) +
  labs(title = "Film runtimes by release date")


ggplot(pixar_films, aes(y = reorder(film, rotten_tomatoes))) +
 geom_linerange(aes(xmin = rotten_tomatoes, xmax = metacritic), size = 2, color = "grey85") +
  geom_point(aes(x = rotten_tomatoes, color = "rotten_tomatoes")) +
  geom_point(aes(x = metacritic, color = "metacritic")) +
  scale_color_manual(values = c("steelblue1", "coral2")) +
  theme_minimal(base_size = 9) +
  labs(
```

```
  title = "Rotten Tomatoes vs Metacritic by film",
  color = NULL,
  y = NULL,
  x = "Audience Score"
)
```

`plural_phrasing` *List of values to use when change is plural (or singular)*

Description

`plural_phrasing()` returns a list object describing the value to use when display when `x - y` is 1 (single) or not one (multiple or fraction). This helps write "1 person" vs "2 people"

Usage

```
plural_phrasing(single, multi)
```

Arguments

`single`	string to use when delta = 1
`multi`	string to use when delta > 1

Details

`plural_phrasing()` will primarily be used in `headline()` and passed along to `compare_conditions()`. Similar to `trend_terms()`. Plural phrases can be passed in a list. See examples below.

Value

Returns a list object.

Examples

```
plural_phrasing(single = "person", multi = "people")

headline(
  x = 1:2,
  y = 0,
  headline = "a difference of {delta} {people}",
  plural_phrases = list(people = plural_phrasing("person", "people"))
)


# a complex example passing multiple trends and plural phrases
headline(
  35, 30,
  headline =
```

```
    "We had {an_increase} of {delta} {people}.
    That is {delta} {more} {employees} \\
    than the same time last year ({orig_values}).",
  trend_phrases = list(
    an_increase = trend_terms("an increase", "a decrease"),
    more = trend_terms("more", "less")
  ),
  plural_phrases =
    list(
      people = plural_phrasing("person", "people"),
      employees = plural_phrasing("employee", "employees")
    )
)
```

`trend_terms` *Phrases for direction of difference*

Description

`trend_terms()` returns a list object describing the values to display when `x` is greater than `y` or `x` is less than `y`.

Usage

```
trend_terms(more = "increase", less = "decrease")
```

Arguments

`more`	string to use when x > y
`less`	string to use when x < y

Details

`trend_terms()` will primarily be used in `headline()` and passed along to `compare_conditions()`. Similar to `plural_phrasing()` Trend terms can be passed in a list. See examples below.

Value

Returns a list object.

See Also

`compare_values()`

Examples

```
headline(
  x = c(9, 11),
  y = 10,
  headline = "{trend} by {delta_p}%",
  trend_phrases = trend_terms("higher", "lower")
)

# a complex example passing multiple trends and plural phrases
headline(
  35, 30,
  headline =
    "We had {an_increase} of {delta} {people}.
    That is {delta} {more} {employees} \\
    than the same time last year ({orig_values}).",
  trend_phrases = list(
    an_increase = trend_terms("an increase", "a decrease"),
    more = trend_terms("more", "less")
  ),
  plural_phrases =
    list(
      people = plural_phrasing("person", "people"),
      employees = plural_phrasing("employee", "employees")
    )
)
```

---

view_list *Compact view of list values*

Description

Compact view of list values

Usage

```
view_list(x)
```

Arguments

x	a vector or list to be transposed

Value

Returns a data frame to display a list or vector vertically.

See Also

compare_values()

Examples

```
compare_values(10, 8) |>
  view_list()

add_article(c(1,8,10, 11, 18)) |>
  view_list()
```

Index

∗ **datasets**
pixar_films, 14

across, *7*
add_article, 2
add_article(), *13*
add_date_columns, 3
add_headline_column, 4

compare_conditions, 6
compare_values, 8
compare_values(), *13*, *17, 18*

demo_data, 10

filter, *6*

glue, *5*, *9–12*
glue_data, *5*, *12, 13*

headline, 11
headline(), *10*
headline_list *(*headline*)*, 11

lst, *7*

pixar_films, 14
plural_phrasing, 16
plural_phrasing(), *10*

select, *5*
starts_with, *5*

trend_terms, *5*, *9*, *12*, 17
trend_terms(), *10*, *13*

view_list, 18
view_list(), *10*