Package: epitabulate (via r-universe)

January 6, 2025

Title Tables for Epidemiological Analysis

Version 0.0.0.9007

Description Produces tables for descriptive epidemiological analysis. These tables describe counts of variables in either line-list or survey data (with appropriate confidence intervals), with additional functionality to calculate odds, risk, and incidence rate ratios directly from a linelist across several variables. This package is part of the 'R4EPIs' project <https://R4epis.netlify.com>.

License GPL-3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

URL https://R4EPI.github.io/epitabulate/, https://github.com/R4EPI/epitabulate/

BugReports https://github.com/R4EPI/epitabulate/issues

Imports dplyr (>= 1.0.2), rlang (>= 0.4.0), tidyr (>= 1.0.0), stats, glue, purrr, tidyselect (>= 1.0.0), tibble (>= 3.0.0), survey, srvyr, forcats, epikit (>= 0.1.2), gtsummary

Suggests testthat, matchmaker (>= 0.1.0), epidict, covr

Remotes R4EPI/epidict, R4EPI/epikit

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

RemoteUrl https://github.com/r4epi/epitabulate

RemoteRef HEAD

RemoteSha 56370b87dd5cf339a65d0b28315e757dd779297b

Contents

`add_mr` *A mortality rate wrapper function (using the gtsummary package)that takes a gtsummary object and returns a gtsummary object with attack rate (per given multiple) with 95% confidence interval*

Description

A mortality rate wrapper function (using the gtsummary package)that takes a gtsummary object and returns a gtsummary object with attack rate (per given multiple) with 95% confidence interval

An attack rate wrapper function (using the gtsummary package)that takes a gtsummary object and returns a gtsummary object withattack rate (per given multiple) with 95% confidence interval

An case fatality rate wrapper function (using the gtsummary package) that takes a gtsummary object and returns a gtsummary object with number of deaths, case fatality rate, and 95% confidence interval.

Usage

```
add_mr(
  gts_object,
  deaths_var,
  population = NULL,
  multiplier = 10^4,
  drop_tblsummary_stat = FALSE
)

add_ar(
  gts_object,
  case_var,
  population = NULL,
  multiplier = 10^4,
  drop_tblsummary_stat = FALSE
)

add_cfr(gts_object, deaths_var)
```

Arguments

gts_object	A data frame, passed by the gtsummary::add_stat function.
deaths_var	the name of a logical column in the data that indicates that the case died, is passed as the first argument to epitabulate::case_fatality_rate_df
multiplier	The base by which to multiply the output:
data	A data frame, passed by the gtsummary::add_stat function.
variable	Name of a variable as the outcome of interest, passed by the gtsummary::add_stat function (e.g. illness).
by	Name of a variable for stratifying, passed by the gtsummary::add_stat function (e.g. illness).
...	additional params that may be passed from gtsummary functions.

Value

a single-row gtsummary object with attack rate results cases, population, attack rate, and 95% confidence interval.

a single-row gtsummary object with attack rate results cases, population, attack rate, and 95% confidence interval.

a single row gtsummary object with case fatality rate results for deaths, cases, cfr, and 95% confidence interval.

---

attack_rate *Rates and Ratios*

Description

Calculate attack rate, case fatality rate, and mortality rate

Usage

```
attack_rate(
  cases,
  population,
  conf_level = 0.95,
  multiplier = 100,
  mergeCI = FALSE,
  digits = 2
)

case_fatality_rate(
  deaths,
  population,
  conf_level = 0.95,
  multiplier = 100,
```

```
  mergeCI = FALSE,
  digits = 2
)

case_fatality_rate_df(
  x,
  deaths,
  group = NULL,
  conf_level = 0.95,
  multiplier = 100,
  mergeCI = FALSE,
  digits = 2,
  add_total = FALSE
)

mortality_rate(
  deaths,
  population,
  conf_level = 0.95,
  multiplier = 10^4,
  mergeCI = FALSE,
  digits = 2
)
```

Arguments

cases, deaths	number of cases or deaths in a population. For _df functions, this can be the name of a logical column OR an evaluated logical expression (see examples).
population	the number of individuals in the population.
conf_level	a number representing the confidence level for which to calculate the confidence interval. Defaults to 0.95, representing a 95% confidence interval using binom::binom.wilson()
multiplier	The base by which to multiply the output: • multiplier = 1: ratio between 0 and 1 • multiplier = 100: proportion • multiplier = 10^4: x per 10,000 people
mergeCI	Whether or not to put the confidence intervals in one column (default is FALSE)
digits	if mergeCI = TRUE, this determines how many digits are printed
x	a data frame
group	the bare name of a column to use for stratifying the output
add_total	if group is not NULL, then this will add a row containing the total value across all groups.

Value

a data frame with five columns that represent the numerator, denominator, rate, lower bound, and upper bound.

- `attack_rate()`: cases, population, ar, lower, upper
- `case_fatality_rate()`: deaths, population, cfr, lower, upper

Examples

```
# Attack rates can be calculated with just two numbers
print(ar <- attack_rate(10, 50), digits = 4) # 20% attack rate

# print them inline using `fmt_ci_df()`
epikit::fmt_ci_df(ar)

# Alternatively, if you want one column for the CI, use `mergeCI = TRUE`
attack_rate(10, 50, mergeCI = TRUE, digits = 2) # 20% attack rate

print(cfr <- case_fatality_rate(1, 100), digits = 2) # CFR of 1%
epikit::fmt_ci_df(cfr)

# using a data frame
if (require("outbreaks")) {
  withAutoprint({
  e <- outbreaks::ebola_sim$linelist
  case_fatality_rate_df(e,
    outcome == "Death",
    group = gender,
    add_total = TRUE,
    mergeCI = TRUE
  )
  })
}
```

`data_frame_from_2x2` *create a data frame from a 2x2 matrix*

Description

create a data frame from a 2x2 matrix

Usage

```
data_frame_from_2x2(x)
```

Arguments

x	a 2x2 matrix or 3D array with exposure variable in rows and outcome in columns

Value

a data frame with the important combinations:

- A_exp_cases
- B_exp_controls
- C_unexp_cases
- D_unexp_controls
- total_cases (A + B)
- total_controls (C + D)
- total_exposed (A + C)
- total_unexposed (B + D)
- total (A + B + C + D)

Examples

```
arr <- c(10, 35, 90, 465, 36, 25, 164, 175)
arr <- array(arr,
  dim = c(2, 2, 2),
  dimnames = list(
    risk = c(TRUE, FALSE),
    outcome = c(TRUE, FALSE),
    old = c(FALSE, TRUE)
  )
)
arr
data_frame_from_2x2(arr)
```

`gt_remove_stat` *A gtsummary wrapper function that takes a gtsummary object and removes a column from the table body by column name*

Description

A gtsummary wrapper function that takes a gtsummary object and removes a column from the table body by column name

A gtsummary wrapper function that takes a data frame and adds cross tabs by exposure and outcome

A function that adds mh odds ratio to an existing gtsummary object with same dimensions (will add to this later.)

Usage

```
gt_remove_stat(gts_object, col_name = "stat_0")

add_crosstabs(
  data,
  exposure,
  outcome,
  case_reference = "outcome",
  var_name = NULL,
  show_overall = TRUE,
  exposure_label = NULL,
  outcome_label = NULL,
  var_label = NULL,
  two_by_two = FALSE,
  gt_statistic = "{n}",
  show_N_header = FALSE
)

gt_mh_odds(
  data,
  exposure,
  outcome,
  strata,
  exposure_label = NULL,
  outcome_label = NULL,
  strata_label = NULL
)
```

Arguments

gts_object	A data frame, passed by the gtsummary::add_stat function
col_name	the column name from the gtsummary object's table_body to remove
data	A data frame with linelist-style individual-level case data
exposure	column name to use as the exposure variable, must be logical class
outcome	column name to use as the outcome variable, must be logical class
show_overall	Logical argument to include overall column in gtsummary output; defaults to TRUE
exposure_label	label for exposure variable
outcome_label	label for outcome variable
variable	Name of a variable as the outcome of interest, passed by the gtsummary::add_stat function (e.g. illness)
by	Name of a variable for stratifying, passed by the gtsummary::add_stat function (e.g. illness). #'@param population the number of individuals in the population, passed to `epitabulate::mortality_rate`
...	additional params that may be passed from gtsummary functions.

Value

a gtsummary object without the named column

gtsummary object with case and control counts tabulated by exposure, along with a crude overall odds ratio and odds using the Cochran-Mantel-Haenszel test with 95% confidence interval (https://cran.r-project.org/web/packages/samplesizeCMH/vignettes/samplesizeCMH-introduction.html)

`tab_linelist` *Tabulate counts and proportions*

Description

Tabulate counts and proportions

Usage

```
tab_linelist(
  x,
  ...,
  strata = NULL,
  keep = TRUE,
  drop = NULL,
  na.rm = TRUE,
  prop_total = FALSE,
  row_total = FALSE,
  col_total = FALSE,
  wide = TRUE,
  transpose = NULL,
  digits = 1,
  pretty = TRUE
)

tab_survey(
  x,
  ...,
  strata = NULL,
  keep = TRUE,
  drop = NULL,
  na.rm = TRUE,
  prop_total = FALSE,
  row_total = FALSE,
  col_total = FALSE,
  wide = TRUE,
  transpose = NULL,
  digits = 1,
  method = "logit",
  deff = FALSE,
```

```
  pretty = TRUE
)
```

Arguments

x	a `data.frame()` or tbl_svy object
...	categorical variables to tabulate
strata	a stratifier to split the data
keep	a character vector specifying which values to retain in the tabulation. Defaults to TRUE, which keeps all the values.
drop	a character vector specifying which values to drop in the tabulation. Defaults to NULL, which keeps all values.
na.rm	When TRUE (default), missing (NA) values present in `var` will be removed from the data set with a warning, causing a change in denominator for the tabulations. Setting this to FALSE creates an explicit missing value called "(Missing)".
prop_total	if TRUE and `strata` is not NULL, then the totals of the rows will be reported as proportions of the total data set, otherwise, they will be proportions within the stratum (default).
row_total	create a new column with the total counts for each row of stratified data.
col_total	create a new row with the total counts for each column of stratified data.
wide	if TRUE (default) and strata is defined, then the results are presented in a wide table with each stratification counts and estimates in separate columns. If FALSE, then the data will be presented in a long format where the counts and estimates are presented in single columns. This has no effect if strata is not defined.
transpose	if `wide = TRUE`, then this will transpose the columns to the rows, which is useful when you stratify by age group. Default is NULL, which will not transpose anything. You have three options for transpose: • `transpose = "variable"`: uses the variable column, (dropping values if strata exists). Use this if you know that your values are all identical or at least identifiable by the variable name. • `transpose = "value"` : uses the value column, (dropping variables if strata exists). Use this if your values are important and the variable names are generic placeholders. • `transpose = "both"` : combines the variable and value columns. Use this if both the variables and values are important.
digits	(survey only) if `pretty = FALSE`, this indicates the number of digits used for proportion and CI
pretty	(survey only) if TRUE, default, the proportion and CI are merged
method	(survey only) a method from `survey::svyciprop()` to calculate the confidence interval. Defaults to "logit".
deff	a logical indicating if the design effect should be reported. Defaults to TRUE.

Value

a tibble::tibble() with a column for variables, a column for values, and counts and proportions. If strata is not NULL and wide = TRUE, then there will be separate columns for each strata for the counts and proportions. Survey data will report confidence intervals.

Examples

```
have_packages <- require("matchmaker") & require("epidict")

if (have_packages) {
  withAutoprint({

    # Simulating linelist data

    linelist <- epidict::gen_data("Measles", numcases = 1000, org = "MSF")
    measles_dict <- epidict::msf_dict("Measles", compact = FALSE)

    # Cleaning linelist data
    linelist_clean <- matchmaker::match_df(
      x = linelist,
      dictionary = measles_dict,
      from = "option_code",
      to = "option_name",
      by = "data_element_shortname",
      order = "option_order_in_set"
    )

    # get a descriptive table by sex
    tab_linelist(linelist_clean, sex)

    # describe prenancy statistics, but remove missing data from the tally
    tab_linelist(linelist_clean, trimester, na.rm = TRUE)

    # describe by symptom

    tab_linelist(linelist_clean,
      cough, nasal_discharge, severe_oral_lesions,
      transpose = "value"
    )
    # describe prenancy statistics, stratifying by vitamin A perscription
    tab_linelist(linelist_clean, trimester, sex,
      strata = prescribed_vitamin_a,
      na.rm = TRUE, row_total = TRUE
    )
  })
}

have_survey_packages <- require("survey") && require("srvyr")
if (have_survey_packages) {
  withAutoprint({
    data(api)
```

```
  # stratified sample
  surv <- apistrat %>%
    as_survey_design(strata = stype, weights = pw)

  s <- surv %>%
   tab_survey(awards, strata = stype, col_total = TRUE, row_total = TRUE, deff = TRUE)
  s

  # making things pretty
  s %>%
    # wrap all "n" variables in braces (note space before n).
    epikit::augment_redundant(" (n)" = " n") %>%
    # relabel all columns containing "prop" to "% (95% CI)"
    epikit::rename_redundant(
      "% (95% CI)" = ci,
      "Design Effect" = deff
    )

  # long data
  surv %>%
    tab_survey(awards, strata = stype, wide = FALSE)

  # tabulate binary variables
  surv %>%
    tab_survey(yr.rnd, sch.wide, awards, keep = "Yes")

  # stratify the binary variables
  surv %>%
    tab_survey(yr.rnd, sch.wide, awards,
      strata    = stype,
      keep      = "Yes"
    )

  # invert the tabulation
  surv %>%
    tab_survey(yr.rnd, sch.wide, awards,
      strata = stype,
      drop = "Yes",
      deff = TRUE,
      row_total = TRUE
    )
 })
}
```

---

tab_univariate *Produce odds ratios, risk ratios or incidence rate ratios*

Description

Produce odds ratios, risk ratios or incidence rate ratios

Usage

```
tab_univariate(
  x,
  outcome,
  ...,
  perstime = NULL,
  strata = NULL,
  measure = "OR",
  extend_output = TRUE,
  digits = 3,
  mergeCI = FALSE,
  woolf_test = FALSE
)
```

Arguments

x	A data frame
outcome	Name of A TRUE/FALSE variable as your outcome of interest (e.g. illness)
...	Names of TRUE/FALSE variables as exposures of interest (e.g. risk factors)
perstime	A numeric variable containing the observation time for each individual
strata	Name of a TRUE/FALSE variable to be used for stratifying results. Note that this results in a different output table - giving you a table of crude measure, measures for each strata and the mantel-haeszel adjusted measure for each exposure variable listed in ...
measure	Specify what you would like to calculated, options are "OR", "RR" or "IRR" default is "OR"
extend_output	TRUE/FALSE to specify whether would like all columns in the outputs (default is TRUE) Non-extended output drops group odds or risk calculations as well as p-values
digits	Specify number of decimal places (default is 3)
mergeCI	Whether or not to put the confidence intervals in one column (default is FALSE)
woolf_test	Only if strata specified and measure is "RR" or "OR". TRUE/FALSE to specify whether to include woolf test for homogeneity p-value. Tests whether there is a significant difference in the estimates between strata.

References

Inspired by Daniel Gardiner, see github repo Real data set for example from http://sphweb.bumc.bu.edu/otlt/mph-modules/bs/bs704-ep713_confounding-em/BS704-EP713_Confounding-EM7.html

See Also

data_frame_from_2x2()

Examples

```
# set up data set, first as 2x2x2 table
arr <- array(
  data = c(10, 35, 90, 465, 36, 25, 164, 175),
  dim  = c(2 , 2 , 2),
  dimnames = list(
    risk    = c(TRUE , FALSE),
    outcome = c(TRUE , FALSE),
    old     = c(FALSE, TRUE)
  )
)
arr

# Create data frame from 2x2x2 table
library("tidyr")
a <- arr %>%
  as.data.frame.table() %>%
  tidyr::uncount(weights = Freq) %>%
  dplyr::mutate_all(as.logical) %>%
  tibble::as_tibble()

# get the results from tab_univariate function
tab_univariate(a, outcome, risk, strata = old, digits = 6, measure = "OR")
tab_univariate(a, outcome, risk, strata = old, digits = 6, measure = "RR")
```

Index