Package: testerror (via r-universe)

February 26, 2025

Title Uncertainty in Multiplex Panel Testing

Version 0.1.0

Description Provides methods to support the estimation of epidemiological parameters based on the results of multiplex panel tests.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Imports magrittr, dplyr, tidyr, tibble, tidyselect, ggplot2, memoise, glue, digest, rappdirs, fs, forcats, purrr, rlang, scales, stringr, rstan (>= 2.18.1), stats, extraDistr, interfacer (>= 0.1.7), pkgutils (>= 0.1.0), BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), StanHeaders (>= 2.18.0)

Remotes github::bristol-vaccine-centre/interfacer, github::bristol-vaccine-centre/pkgutils

Suggests tidyverse, knitr, rmarkdown, binom, huxtable, here, testthat, devtools, ggh4x

VignetteBuilder knitr

Config/testthat/edition 3

URL https://bristol-vaccine-centre.github.io/testerror/index.html, https://github.com/bristol-vaccine-centre/testerror, https://doi.org/10.5281/zenodo.7691196

BugReports https://github.com/bristol-vaccine-centre/testerror/issues

Depends R (>= 3.4.0)

LazyData true

Config/pak/sysreqs make libicu-dev libxml2-dev

Repository https://bristol-vaccine-centre.r-universe.dev

RemoteUrl https://github.com/bristol-vaccine-centre/testerror

RemoteRef 0.1.0

RemoteSha 4420c57e703087827a0c63ccf5f2229da26b66f8

Contents

`.input_data` *Dataframe format for component test results*

Description

A dataframe containing the following columns:

- id (character) - the patient identifier
- test (factor) - the test type
- result (logical) - the test result

Ungrouped.

No default value.

Usage

```
.input_data
```

Format

An object of class `iface` (inherits from `tbl_df`, `tbl`, `data.frame`) with 3 rows and 3 columns.

.input_panel_data *Dataframe format for panel test results*

Description

A dataframe containing the following columns:

- id (character) - the patient identifier
- result (logical) - the panel result

Ungrouped.

No default value.

Usage

```
.input_panel_data
```

Format

An object of class `iface` (inherits from `tbl_df`, `tbl`, `data.frame`) with 2 rows and 3 columns.

.output_data *Dataframe format for true prevalence results*

Description

A dataframe containing the following columns:

- test (character) - the name of the test or panel
- prevalence.lower (numeric) - the lower estimate
- prevalence.median (numeric) - the median estimate
- prevalence.upper (numeric) - the upper estimate
- prevalence.method (character) - the method of estimation
- prevalence.label (character) - a fomatted label of the true prevalence estimate with CI

Ungrouped.

No default value.

Usage

```
.output_data
```

Format

An object of class `iface` (inherits from `tbl_df`, `tbl`, `data.frame`) with 6 rows and 3 columns.

apparent_prevalence *Apparent prevalence from known prevalence*

Description

The observed counts of disease is going to be a binomial but with the apparent prevalence as a probability. This will never be less than (1-specificity) of the test (and never more than the sensitivity). When either of those quantities are uncertain the shape of the distribution of observed counts is not clear cut.

Usage

```
apparent_prevalence(p, sens, spec)
```

Arguments

p	the true value of the prevalence
sens	the sensitivity of the test
spec	the specificity of the test

Value

the expected value of apparent prevalence

Examples

```
apparent_prevalence(0, 0.75, 0.97)
apparent_prevalence(1, 0.75, 0.97)
```

as_tibble.beta_dist *convert a beta distribution to a tibble*

Description

convert a beta distribution to a tibble

Usage

```
## S3 method for class 'beta_dist'
as_tibble(x, prefix = NULL, confint = 0.95, ...)
```

Arguments

x	the beta distribution
prefix	name to output columns prefix.lower, prefix.upper etc
confint	confidence intervals
...	not used

`as_tibble.beta_dist_list` *convert a list of betas to a tibble*

Description

convert a list of betas to a tibble

Usage

```
## S3 method for class 'beta_dist_list'
as_tibble(x, ...)
```

Arguments

`x`	a beta dist list
`...`	Arguments passed on to `as_tibble.beta_dist` `prefix` name to output columns prefix.lower, prefix.upper etc `confint` confidence intervals

Value

a tibble

`bayesian_component_logit_model` *Bayesian simpler model true prevalence for component*

Description

Bayesian simpler model true prevalence for component

Usage

```
bayesian_component_logit_model(
  pos_obs,
  n_obs,
  false_pos_controls = NULL,
  n_controls = NULL,
  false_neg_diseased = NULL,
  n_diseased = NULL,
  ...,
  sens = sens_prior(),
  spec = spec_prior(),
  confint = 0.95,
```

```
  fmt = "%1.2f%% [%1.2f%% - %1.2f%%]",
  chains = 4,
  warmup = 1000,
  iter = 2000,
  cache_result = TRUE
)
```

Arguments

pos_obs	the number of positive observations for a given test
n_obs	the number of observations for a given test
false_pos_controls	the number of positives that appeared in the specificity disease-free control group. These are by definition false positives. This is (1-specificity)*n_controls
n_controls	the number of controls in the specificity disease-free control group.
false_neg_diseased	the number of negatives that appeared in the sensitivity confirmed disease group. These are by definition false negatives. This is (1-sensitivity)*n_diseased
n_diseased	the number of confirmed disease cases in the sensitivity control group.
...	not used
sens	the prior sensitivity of the test as a `beta_dist`.
spec	the prior specificity of the test as a `beta_dist`.
confint	confidence interval limits
fmt	a `sprintf` formatting string accepting 3 numbers
chains	A positive integer specifying the number of Markov chains. The default is 4.
warmup	A positive integer specifying the number of warmup (aka burnin) iterations per chain. If step-size adaptation is on (which it is by default), this also controls the number of iterations for which adaptation is run (and hence these warmup samples should not be used for inference). The number of warmup iterations should be smaller than `iter` and the default is `iter/2`.
iter	A positive integer specifying the number of iterations for each chain (including warmup). The default is 2000.
cache_result	save the result of the sampling in memory for the current session

Value

a list of dataframes containing the prevalence, sensitivity, and specificity estimates, and a `stanfit` object with the raw fit data

`bayesian_component_simpler_model` *Bayesian simpler model true prevalence for component*

Description

Bayesian simpler model true prevalence for component

Usage

```
bayesian_component_simpler_model(
  pos_obs,
  n_obs,
  false_pos_controls = NULL,
  n_controls = NULL,
  false_neg_diseased = NULL,
  n_diseased = NULL,
  ...,
  sens = uniform_prior(),
  spec = uniform_prior(),
  confint = 0.95,
  fmt = "%1.2f%% [%1.2f%% - %1.2f%%]",
  chains = 4,
  warmup = 1000,
  iter = 2000,
  cache_result = TRUE
)
```

Arguments

`pos_obs`	the number of positive observations for a given test
`n_obs`	the number of observations for a given test
`false_pos_controls`	the number of positives that appeared in the specificity disease-free control group. These are by definition false positives. This is (1-specificity)*n_controls
`n_controls`	the number of controls in the specificity disease-free control group.
`false_neg_diseased`	the number of negatives that appeared in the sensitivity confirmed disease group. These are by definition false negatives. This is (1-sensitivity)*n_diseased
`n_diseased`	the number of confirmed disease cases in the sensitivity control group.
`...`	not used
`sens`	the prior sensitivity of the test as a `beta_dist`.
`spec`	the prior specificity of the test as a `beta_dist`.
`confint`	confidence interval limits

fmt	a sprintf formatting string accepting 3 numbers
chains	A positive integer specifying the number of Markov chains. The default is 4.
warmup	A positive integer specifying the number of warmup (aka burnin) iterations per chain. If step-size adaptation is on (which it is by default), this also controls the number of iterations for which adaptation is run (and hence these warmup samples should not be used for inference). The number of warmup iterations should be smaller than iter and the default is iter/2.
iter	A positive integer specifying the number of iterations for each chain (including warmup). The default is 2000.
cache_result	save the result of the sampling in memory for the current session

Value

a list of dataframes containing the prevalence, sensitivity, and specificity estimates, and a stanfit object with the raw fit data

---

bayesian_panel_complex_model

Bayesian models true prevalence for panel

Description

Uses resampling to incorporate uncertainty of sensitivity and specificity into an estimate of true prevalence from a given value of apparent prevalence.

Usage

```
bayesian_panel_complex_model(
  test_results = testerror::.input_data,
  false_pos_controls = NULL,
  n_controls = NULL,
  false_neg_diseased = NULL,
  n_diseased = NULL,
  ...,
  sens = uniform_prior(),
  spec = uniform_prior(),
  panel_sens = uniform_prior(),
  panel_spec = uniform_prior(),
  panel_name = "Panel",
  confint = 0.95,
  fmt = "%1.2f%% [%1.2f%% - %1.2f%%]",
  chains = 4,
  warmup = 1000,
  iter = 2000,
  cache_result = TRUE
)
```

Arguments

test_results	A dataframe containing the following columns: • id (character) - the patient identifier • test (factor) - the test type • result (logical) - the test result Ungrouped. No default value.
false_pos_controls	the number of positives that appeared in the specificity disease-free control group. These are by definition false positives. This is (1-specificity)*n_controls
n_controls	the number of controls in the specificity disease-free control group.
false_neg_diseased	the number of negatives that appeared in the sensitivity confirmed disease group. These are by definition false negatives. This is (1-sensitivity)*n_diseased
n_diseased	the number of confirmed disease cases in the sensitivity control group.
...	not used
sens	the prior sensitivity of the test as a beta_dist.
spec	the prior specificity of the test as a beta_dist.
panel_sens	the prior sensitivity of the panel as a beta_dist (optional)
panel_spec	the prior specificity of the panel as a beta_dist (optional)
panel_name	the name of the panel for combined result
confint	confidence interval limits
fmt	a sprintf formatting string accepting 3 numbers
chains	A positive integer specifying the number of Markov chains. The default is 4.
warmup	A positive integer specifying the number of warmup (aka burnin) iterations per chain. If step-size adaptation is on (which it is by default), this also controls the number of iterations for which adaptation is run (and hence these warmup samples should not be used for inference). The number of warmup iterations should be smaller than iter and the default is iter/2.
iter	A positive integer specifying the number of iterations for each chain (including warmup). The default is 2000.
cache_result	save the result of the sampling in memory for the current session

Details

This is not vectorised

Value

a list of dataframes containing the prevalence, sensitivity, and specificity estimates, and a stanfit object with the raw fit data

`bayesian_panel_logit_model` *Bayesian logit model true prevalence for panel*

Description

The beta distribution priors in this model will actually be converted to logit_normal distributions

Usage

```
bayesian_panel_logit_model(
  panel_pos_obs,
  panel_n_obs,
  pos_obs,
  n_obs,
  test_names,
  false_pos_controls = NULL,
  n_controls = NULL,
  false_neg_diseased = NULL,
  n_diseased = NULL,
  ...,
  sens = sens_prior(),
  spec = spec_prior(),
  panel_sens = sens_prior(),
  panel_spec = spec_prior(),
  panel_name = "Panel",
  confint = 0.95,
  fmt = "%1.2f%% [%1.2f%% - %1.2f%%]",
  chains = 4,
  warmup = 1000,
  iter = 2000,
  cache_result = TRUE
)
```

Arguments

`panel_pos_obs`	the number of positive observations for a given panel of tests
`panel_n_obs`	the number of observations for each component test
`pos_obs`	the number of positive observations for a given test
`n_obs`	the number of observations for a given test
`test_names`	a vector of the component test names in desired order
`false_pos_controls`	the number of positives that appeared in the specificity disease-free control group. These are by definition false positives. This is (1-specificity)*n_controls
`n_controls`	the number of controls in the specificity disease-free control group.

false_neg_diseased	the number of negatives that appeared in the sensitivity confirmed disease group. These are by definition false negatives. This is (1-sensitivity)*n_diseased
n_diseased	the number of confirmed disease cases in the sensitivity control group.
...	not used
sens	the prior sensitivity of the test as a beta_dist.
spec	the prior specificity of the test as a beta_dist.
panel_sens	the prior sensitivity of the panel as a beta_dist (optional)
panel_spec	the prior specificity of the panel as a beta_dist (optional)
panel_name	the name of the panel for combined result
confint	confidence interval limits
fmt	a sprintf formatting string accepting 3 numbers
chains	A positive integer specifying the number of Markov chains. The default is 4.
warmup	A positive integer specifying the number of warmup (aka burnin) iterations per chain. If step-size adaptation is on (which it is by default), this also controls the number of iterations for which adaptation is run (and hence these warmup samples should not be used for inference). The number of warmup iterations should be smaller than iter and the default is iter/2.
iter	A positive integer specifying the number of iterations for each chain (including warmup). The default is 2000.
cache_result	save the result of the sampling in memory for the current session

Value

a list of dataframes containing the prevalence, sensitivity, and specificity estimates, and a stanfit object with the raw fit data

---

bayesian_panel_simpler_model

Bayesian simpler model true prevalence for panel

Description

Bayesian simpler model true prevalence for panel

Usage

```
bayesian_panel_simpler_model(
  panel_pos_obs,
  panel_n_obs,
  pos_obs,
  n_obs,
  test_names,
```

```
  false_pos_controls = NULL,
  n_controls = NULL,
  false_neg_diseased = NULL,
  n_diseased = NULL,
  ...,
  sens = uniform_prior(),
  spec = uniform_prior(),
  panel_sens = uniform_prior(),
  panel_spec = uniform_prior(),
  panel_name = "Panel",
  confint = 0.95,
  fmt = "%1.2f%% [%1.2f%% - %1.2f%%]",
  chains = 4,
  warmup = 1000,
  iter = 2000,
  cache_result = TRUE
)
```

Arguments

panel_pos_obs	the number of positive observations for a given panel of tests
panel_n_obs	the number of observations for each component test
pos_obs	the number of positive observations for a given test
n_obs	the number of observations for a given test
test_names	a vector of the component test names in desired order
false_pos_controls	the number of positives that appeared in the specificity disease-free control group. These are by definition false positives. This is (1-specificity)*n_controls
n_controls	the number of controls in the specificity disease-free control group.
false_neg_diseased	the number of negatives that appeared in the sensitivity confirmed disease group. These are by definition false negatives. This is (1-sensitivity)*n_diseased
n_diseased	the number of confirmed disease cases in the sensitivity control group.
...	not used
sens	the prior sensitivity of the test as a beta_dist.
spec	the prior specificity of the test as a beta_dist.
panel_sens	the prior sensitivity of the panel as a beta_dist (optional)
panel_spec	the prior specificity of the panel as a beta_dist (optional)
panel_name	the name of the panel for combined result
confint	confidence interval limits
fmt	a sprintf formatting string accepting 3 numbers
chains	A positive integer specifying the number of Markov chains. The default is 4.

`warmup` A positive integer specifying the number of warmup (aka burnin) iterations per chain. If step-size adaptation is on (which it is by default), this also controls the number of iterations for which adaptation is run (and hence these warmup samples should not be used for inference). The number of warmup iterations should be smaller than `iter` and the default is `iter/2`.

`iter` A positive integer specifying the number of iterations for each chain (including warmup). The default is 2000.

`cache_result` save the result of the sampling in memory for the current session

Value

a list of dataframes containing the prevalence, sensitivity, and specificity estimates, and a `stanfit` object with the raw fit data

---

`bayesian_panel_true_prevalence_model` *Execute one of a set of bayesian models*

Description

Execute one of a set of bayesian models

Usage

```
bayesian_panel_true_prevalence_model(
  ...,
  model_type = c("logit", "simpler", "complex")
)
```

Arguments

`...` Arguments passed on to `bayesian_panel_complex_model`, `bayesian_panel_simpler_model`, `bayesian_panel_logit_model`

`test_results` A dataframe containing the following columns:

- id (character) - the patient identifier
- test (factor) - the test type
- result (logical) - the test result

Ungrouped.
No default value.

`panel_sens` the prior sensitivity of the panel as a `beta_dist` (optional)

`panel_spec` the prior specificity of the panel as a `beta_dist` (optional)

`panel_name` the name of the panel for combined result

`cache_result` save the result of the sampling in memory for the current session

`false_pos_controls` the number of positives that appeared in the specificity disease-free control group. These are by definition false positives. This is (1-specificity)*n_controls

`n_controls` the number of controls in the specificity disease-free control group.

`false_neg_diseased` the number of negatives that appeared in the sensitivity confirmed disease group. These are by definition false negatives. This is (1-sensitivity)*n_diseased

`n_diseased` the number of confirmed disease cases in the sensitivity control group.

`sens` the prior sensitivity of the test as a `beta_dist`.

`spec` the prior specificity of the test as a `beta_dist`.

`confint` confidence interval limits

`fmt` a `sprintf` formatting string accepting 3 numbers

`chains` A positive integer specifying the number of Markov chains. The default is 4.

`iter` A positive integer specifying the number of iterations for each chain (including warmup). The default is 2000.

`warmup` A positive integer specifying the number of warmup (aka burnin) iterations per chain. If step-size adaptation is on (which it is by default), this also controls the number of iterations for which adaptation is run (and hence these warmup samples should not be used for inference). The number of warmup iterations should be smaller than `iter` and the default is `iter/2`.

`panel_pos_obs` the number of positive observations for a given panel of tests

`panel_n_obs` the number of observations for each component test

`test_names` a vector of the component test names in desired order

`pos_obs` the number of positive observations for a given test

`n_obs` the number of observations for a given test

`model_type` The bayesian model used one of "logit", "simpler", "complex"

Value

A dataframe containing the following columns:

- test (character) - the name of the test or panel
- prevalence.lower (numeric) - the lower estimate
- prevalence.median (numeric) - the median estimate
- prevalence.upper (numeric) - the upper estimate
- prevalence.method (character) - the method of estimation
- prevalence.label (character) - a fomatted label of the true prevalence estimate with CI

Ungrouped.

No default value.

`bayesian_true_prevalence_model` *Execute one of a set of bayesian models*

Description

Execute one of a set of bayesian models

Usage

```
bayesian_true_prevalence_model(..., model_type = c("logit", "simpler"))
```

Arguments

`...`	Arguments passed on to `bayesian_component_simpler_model`, `bayesian_component_logit_model` `cache_result` save the result of the sampling in memory for the current session `pos_obs` the number of positive observations for a given test `n_obs` the number of observations for a given test `false_pos_controls` the number of positives that appeared in the specificity disease-free control group. These are by definition false positives. This is (1-specificity)*n_controls `n_controls` the number of controls in the specificity disease-free control group. `false_neg_diseased` the number of negatives that appeared in the sensitivity confirmed disease group. These are by definition false negatives. This is (1-sensitivity)*n_diseased `n_diseased` the number of confirmed disease cases in the sensitivity control group. `sens` the prior sensitivity of the test as a `beta_dist`. `spec` the prior specificity of the test as a `beta_dist`. `confint` confidence interval limits `fmt` a `sprintf` formatting string accepting 3 numbers `chains` A positive integer specifying the number of Markov chains. The default is 4. `iter` A positive integer specifying the number of iterations for each chain (including warmup). The default is 2000. `warmup` A positive integer specifying the number of warmup (aka burnin) iterations per chain. If step-size adaptation is on (which it is by default), this also controls the number of iterations for which adaptation is run (and hence these warmup samples should not be used for inference). The number of warmup iterations should be smaller than `iter` and the default is `iter/2`.
`model_type`	The bayesian model used - one of "logit" or "simpler"

Value

A dataframe containing the following columns:

- test (character) - the name of the test or panel
- prevalence.lower (numeric) - the lower estimate
- prevalence.median (numeric) - the median estimate
- prevalence.upper (numeric) - the upper estimate
- prevalence.method (character) - the method of estimation
- prevalence.label (character) - a fomatted label of the true prevalence estimate with CI

Ungrouped.

No default value.

`beta_dist` *Generate a beta distribution out of probabilities, or positive and negative counts*

Description

Generate a beta distribution out of probabilities, or positive and negative counts

Usage

```
beta_dist(..., p = NULL, q = NULL, n = NULL, shape1 = NULL, shape2 = NULL)
```

Arguments

`...`	not used
`p`	the first shape / the probability or count of success
`q`	(optional) the second shape / the probability or count of failure
`n`	(optional) the number of trials.
`shape1`	the first shape parameter (use this to force interpretation as shape)
`shape2`	the second shape parameter (use this to force interpretation as shape)

Value

either a single `beta_dist` object or a list of `beta_dists`

Examples

```
beta_dist(shape1 = c(1,2,3),shape2 = c(3,2,1))
beta_dist(p = 0.7, n = 2)
```

beta_fit — *Fit a beta distribution to data using method of moments*

Description

Fit a beta distribution to data using method of moments

Usage

```
beta_fit(samples, na.rm = FALSE)
```

Arguments

samples	a set of probabilities
na.rm	should we ignore NA values

Value

a beta_dist S3 object fitted to the data.

Examples

```
beta_fit(stats::rbeta(10000,40,60))
beta_fit(stats::rbeta(10000,1,99))
```

beta_params — *Generate concave beta distribution parameters from mean and confidence intervals*

Description

Generate concave beta distribution parameters from mean and confidence intervals

Usage

```
beta_params(median, lower, upper, confint = 0.95, widen = 1, limit = 1, ...)
```

Arguments

median	the median of the probability given
lower	the lower ci of the probability given
upper	the upper ci of the probability given
confint	the ci limits
widen	widen the spread of the final beta by this factor
limit	the lowest possible value for the shape parameters of the resulting beta_dist (1 enforces that the distribution is convex)
...	not used

Value

a list with shape1, shape2 values, and d, p, q and r functions

Examples

```
beta = beta_params(0.25, 0.1, 0.3)
```

`ci_to_logitnorm` *Generate mu and sigma parameters for a logitnormal distribution*

Description

The resulting logitnorm distribution will have a set median. The confidence intervals will not match those provided as they are used as a inter-quartile range.

Usage

```
ci_to_logitnorm(median, lower, upper, ci = 0.95, fix_median = TRUE, ...)
```

Arguments

`median`	the median of the
`lower`	the lower CI
`upper`	the upper CI
`ci`	the confidence limits
`fix_median`	make the median of the logitnorm be the same as the median given. This can cause issues when very skewed distributions are used
`...`	not used

Value

a tibble with mu and sigma columns

format.beta_dist *Format a beta distribution*

Description

Format a beta distribution

Usage

```
## S3 method for class 'beta_dist'
format(x, glue = .default_beta_dist_format(), ...)
```

Arguments

x	the beta distribution
glue	a glue spec taking any of shape1, shape2, conc, mean, median, upper, lower
...	not used

Value

nothing

Examples

```
format(beta_dist(shape1=3,shape2=6), "{format(mean*100, digits=3)}%")
```

format.beta_dist_list *Format a beta distribution list*

Description

Format a beta distribution list

Usage

```
## S3 method for class 'beta_dist_list'
format(x, ...)
```

Arguments

x	the beta distribution list
...	Arguments passed on to format.beta_dist glue a glue spec taking any of shape1, shape2, conc, mean, median, upper, lower

Value

nothing

`fp_p_value` *Significance of an uncertain test result*

Description

Calculates a p-value for a count of positive test results based on false positive (specificity) controls. The null hypothesis is that the prevalence of the disease is zero.

Usage

```
fp_p_value(
  pos_obs,
  n_obs,
  false_pos_controls,
  n_controls,
  format = "%1.3g",
  lim = 1e-04,
  bonferroni = NULL,
  ...
)
```

Arguments

`pos_obs`	the number of positive observations for a given test
`n_obs`	the number of observations for a given test
`false_pos_controls`	the number of positives that appeared in the specificity disease-free control group. These are by definition false positives. This is `(1-specificity)*n_controls`
`n_controls`	the number of controls in the specificity disease-free control group.
`format`	a sprintf fmt string for the p-value
`lim`	a lower value to display
`bonferroni`	the number of simultaneous hypotheses that are being tested
`...`	not used

Details

This p_value does not tell you whether this count can be trusted only if the prevalence of this disease is significantly more than zero after this observation.

Value

a vector of p-values for the count

Examples

```
# calculate p-values for counts derived from 300 samples
# 10 observations is within noise of test
# 20 observations is unlikely on 1200 observations
fp_p_value(c(10,2,4,3,10,20), 1200, c(0,0,2,0,2,0)+2, 800)

# if the same observations are made against a smaller group then we get
# a positive result for 10
fp_p_value( c(10,2,4,3,10,20), 1000, c(2,2,4,2,4,2), 800)

tibble::tibble(
  x = c(1,2,5,10,20,40,20,20,20,20,20),
  n = 1000,
  fp_controls = c(0,0,0,0,0,0,0,1,2,3,4)+2,
  n_controls = 800
) %>% dplyr::mutate(
  p_value = fp_p_value(x, n, fp_controls, n_controls)
) %>% dplyr::glimpse()
```

`fp_signif_level` *Identify the minimum number of positive test result observations needed to be confident the disease has a non-zero prevalence.*

Description

Identify the minimum number of positive test result observations needed to be confident the disease has a non-zero prevalence.

Usage

```
fp_signif_level(
  n_obs,
  false_pos_controls,
  n_controls,
  bonferroni = NULL,
  ...,
  spec = NULL
)
```

Arguments

`n_obs`	the number of tests performed.
`false_pos_controls`	the number of positives that appeared in the specificity disease-free control group. These are by definition false positives. This is `(1-specificity)*n_controls`
`n_controls`	the number of controls in the specificity disease-free control group.
`bonferroni`	the number of simultaneous tests considered.
`...`	not used
`spec`	a prior value for specificity as a `beta`

Value

a vector of test positive counts which are the lowest significant value that could be regarded as not due to chance.

Examples

```
# lowest significant count of positives in 1000 tests
fp_signif_level(1000, false_pos_controls = 0:5, n_controls=800)
fp_signif_level(c(1000,800,600,400), false_pos_controls = 1:4, n_controls=800)
```

---

get_beta_shape *Get a parameter of the* beta_dist

Description

Get a parameter of the beta_dist

Usage

```
get_beta_shape(x, type = c("shape1", "shape2", "conc"))
```

Arguments

x	a beta_dist or beta_dist_list acting as the prior
type	the parameter to extract one of shape1 or shape2 or conc

Value

a vector of doubles

Examples

```
get_beta_shape(beta_dist(shape1=1,shape2=1))
get_beta_shape(beta_dist(shape1=2:5,shape2=1:4))
```

`get_beta_shape.beta_dist`
Get a parameter of the `beta_dist`

Description

Get a parameter of the `beta_dist`

Usage

```
## S3 method for class 'beta_dist'
get_beta_shape(x, type = c("shape1", "shape2", "conc"))
```

Arguments

`x`	a `beta_dist` or `beta_dist_list` acting as the prior
`type`	the parameter to extract one of `shape1` or `shape2` or `conc`

Value

a vector of doubles

Examples

```
get_beta_shape(beta_dist(shape1=1,shape2=1))
get_beta_shape(beta_dist(shape1=2:5,shape2=1:4))
```

`get_beta_shape.beta_dist_list`
Get a parameter of the `beta_dist`

Description

Get a parameter of the `beta_dist`

Usage

```
## S3 method for class 'beta_dist_list'
get_beta_shape(x, type = c("shape1", "shape2", "conc"))
```

Arguments

`x`	a `beta_dist` or `beta_dist_list` acting as the prior
`type`	the parameter to extract one of `shape1` or `shape2` or `conc`

Value

a vector of doubles

Examples

```
get_beta_shape(beta_dist(shape1=1,shape2=1))
get_beta_shape(beta_dist(shape1=2:5,shape2=1:4))
```

---

inv_logit *The inverse logit function*

Description

The inverse logit function

Usage

```
inv_logit(y)
```

Arguments

y	a number between -Inf and Inf

Value

a number between 0 and 1

---

length.beta_dist *Detect the length of a beta distribution*

Description

Detect the length of a beta distribution

Usage

```
## S3 method for class 'beta_dist'
length(x, ...)
```

Arguments

x	the beta distribution
...	not used

Value

always 1

length.beta_dist_list *Detect the length of a beta distribution list*

Description

Detect the length of a beta distribution list

Usage

```
## S3 method for class 'beta_dist_list'
length(x, ...)
```

Arguments

x	the beta distribution list
...	not used

Value

the length of the list

logit *The logit function*

Description

The logit function

Usage

```
logit(x)
```

Arguments

x	a number between 0 and 1

Value

a number between -Inf and Inf

odds_ratio_ve *Calculate a vaccine effectiveness estimate based on an odds ratio*

Description

This assumes that OR ~ RR which is only true if controls » cases The OR method can be used in test negative designs where disease positive relates to vaccine treatable disease and disease negative relates to non vaccine treatable disease

This assumes that OR ~ RR which is only true if controls » cases The OR method can be used in test negative designs where disease positive relates to vaccine treatable disease and disease negative relates to non vaccine treatable disease

Usage

```
odds_ratio_ve(
  vaccinatedCase,
  unvaccinatedCase,
  vaccinatedControl,
  unvaccinatedControl,
  confint = c(0.025, 0.975)
)

odds_ratio_ve(
  vaccinatedCase,
  unvaccinatedCase,
  vaccinatedControl,
  unvaccinatedControl,
  confint = c(0.025, 0.975)
)
```

Arguments

vaccinatedCase	count of disease positive vaccine positive
unvaccinatedCase	count of disease positive vaccine negative
vaccinatedControl	count of disease negative vaccine positive
unvaccinatedControl	count of disease negative vaccine positive
confint	the confidence intervals

Value

a dataframe

a dataframe

Examples

```
tibble::tibble(
  N_vacc = 42240,
  N_unvacc = 42256,
  N_vacc_pn_pos = 49,
  N_unvacc_pn_pos = 90
) %>% dplyr::mutate(
 odds_ratio_ve(N_vacc_pn_pos, N_unvacc_pn_pos, N_vacc-N_vacc_pn_pos, N_unvacc-N_unvacc_pn_pos)
)


tibble::tibble(
  N_vacc = 42240,
  N_unvacc = 42256,
  N_vacc_pn_pos = 49,
  N_unvacc_pn_pos = 90
) %>% dplyr::mutate(
 odds_ratio_ve(N_vacc_pn_pos, N_unvacc_pn_pos, N_vacc-N_vacc_pn_pos, N_unvacc-N_unvacc_pn_pos)
)
```

---

optimal_performance *Test optimal performance*

Description

For a given combination of prevalence, sensitivity and specificity this gives the critical threshold at which true prevalence equals apparent prevalence

Usage

```
optimal_performance(p = NULL, sens = NULL, spec = NULL)
```

Arguments

p	the prevalence or apparent prevalence
sens	the sensitivity of the test
spec	the specificity of the test

Value

the combination of sensitivity and specificity where apparent prevalence equals true prevalence

Examples

```
optimal_performance(p=0.1, sens=0.75)
optimal_performance(p=0.005, spec=0.9975)
```

panel_prevalence *Expected test panel prevalence assuming independence*

Description

Expected test panel prevalence assuming independence

Usage

```
panel_prevalence(p, na.rm = FALSE)
```

Arguments

p	a vector of prevalences of the component tests
na.rm	remove NA values?

Value

a single value for the effective specificity of the combination of the tests

Examples

```
panel_prevalence(p = rep(0.01,24))
```

panel_sens *Test panel combination sensitivity*

Description

Calculate the sensitivity of a combination of tests, where the tests are testing for different conditions and positive results are combined into a panel using a logical OR. Because false negatives from each component of a panel can be cancelled out by true positives, or false positives from other components of the test depending on the prevalence of the underlying conditions, the combined false negative rate is lower the more cases there arecombine the false positive rate for the panel is higher than the individual components (and hence the true negative rate a.k.a specificity is lower).

Usage

```
panel_sens(p, sens, spec, na.rm = FALSE)
```

Arguments

p	the true prevalence (one of p or ap must be given)
sens	a vector of sensitivities of the component tests
spec	a vector of specificity of the component tests
na.rm	remove NA values?

Value

an effective specificity for the combination of the tests

Examples

```
#TODO
```

`panel_sens_estimator` *Estimate test panel combination sensitivity*

Description

Estimate the sensitivity of a combination of tests, where the tests are testing for different conditions and positive results are combined into a panel using a logical OR. Because false negatives from each component of a panel can be cancelled out by true positives, or false positives from other components of the test depending on the prevalence of the underlying conditions, the combined false negative rate is lower the more cases there arecombine the false positive rate for the panel is higher than the individual components (and hence the true negative rate a.k.a specificity is lower).

Usage

```
panel_sens_estimator(ap, sens, spec, na.rm = FALSE)
```

Arguments

`ap`	the apparent prevalence or test positivity (one of p or ap must be given)
`sens`	a vector of sensitivities of the component tests
`spec`	a vector of specificity of the component tests
`na.rm`	remove NA values?

Value

an effective specificity for the combination of the tests

Examples

```
#TODO
```

panel_spec *Test panel combination specificity*

Description

Calculate the specificity of a combination of tests, where the tests are testing for different conditions and positive results are combined into a panel using a logical OR. Because false positives from each component of a panel combine the false positive rate for the panel is higher than the individual components (and hence the true negative rate a.k.a specificity is lower).

Usage

```
panel_spec(spec, na.rm = FALSE)
```

Arguments

spec	a vector of specificity of the component tests
na.rm	remove NA values?

Value

a single value for the effective specificity of the combination of the tests

Examples

```
panel_spec(spec = rep(0.9975,24))
```

prevalence_lang_reiczigel *True prevalence from apparent prevalence with uncertainty*

Description

Uses lang-reiczigel estimators to incorporate uncertainty of sensitivity and specificity into an estimate of true prevalence from a given value of apparent prevalence.

Usage

```
prevalence_lang_reiczigel(
  pos_obs,
  n_obs,
  false_pos_controls = NULL,
  n_controls = NULL,
  false_neg_diseased = NULL,
  n_diseased = NULL,
```

```
  ...,
  spec = spec_prior(),
  sens = sens_prior(),
  confint = 0.95,
  fmt = "%1.2f%% [%1.2f%% - %1.2f%%]"
)
```

Arguments

pos_obs	the number of positive observations for a given test
n_obs	the number of observations for a given test
false_pos_controls	the number of positives that appeared in the specificity disease-free control group. These are by definition false positives. This is (1-specificity)*n_controls
n_controls	the number of controls in the specificity disease-free control group.
false_neg_diseased	the number of negatives that appeared in the sensitivity confirmed disease group. These are by definition false negatives. This is (1-sensitivity)*n_diseased
n_diseased	the number of confirmed disease cases in the sensitivity control group.
...	not used
spec	the prior specificity of the test as a `beta_dist`.
sens	the prior sensitivity of the test as a `beta_dist`.
confint	confidence interval limits
fmt	a `sprintf` formatting string accepting 3 numbers

Value

the expected value of apparent prevalence

---

prevalence_panel_lang_reiczigel

Lang-Reiczigel true prevalence for panel

Description

Uses resampling to incorporate uncertainty of sensitivity and specificity into an estimate of true prevalence from a given value of apparent prevalence.

Usage

```
prevalence_panel_lang_reiczigel(
  panel_pos_obs,
  panel_n_obs,
  pos_obs,
  n_obs,
  false_pos_controls = NULL,
  n_controls = NULL,
  false_neg_diseased = NULL,
  n_diseased = NULL,
  ...,
  spec = spec_prior(),
  sens = sens_prior(),
  confint = 0.95,
  fmt = "%1.2f%% [%1.2f%% – %1.2f%%]",
  samples = 1000
)
```

Arguments

panel_pos_obs	the number of positive observations for a given panel of tests
panel_n_obs	a vector of the number of observations for each component test
pos_obs	the number of positive observations for a given test
n_obs	the number of observations for a given test
false_pos_controls	the number of positives that appeared in the specificity disease-free control group. These are by definition false positives. This is (1-specificity)*n_controls
n_controls	the number of controls in the specificity disease-free control group.
false_neg_diseased	the number of negatives that appeared in the sensitivity confirmed disease group. These are by definition false negatives. This is (1-sensitivity)*n_diseased
n_diseased	the number of confirmed disease cases in the sensitivity control group.
...	Arguments passed on to prevalence_lang_reiczigel
spec	the prior specificity of the test as a beta_dist.
sens	the prior sensitivity of the test as a beta_dist.
confint	confidence interval limits
fmt	a sprintf formatting string accepting 3 numbers
samples	number of random draws of sensitivity and specificity (optional - default 1000)

Details

This is not vectorised

Value

the expected value of apparent prevalence

Examples

```
#TODO
```

`print.beta_dist` *Print a beta distribution*

Description

Print a beta distribution

Usage

```
## S3 method for class 'beta_dist'
print(x, ...)
```

Arguments

`x`	the beta distribution
`...`	not used

Value

nothing

`print.beta_dist_list` *Print a beta distribution*

Description

Print a beta distribution

Usage

```
## S3 method for class 'beta_dist_list'
print(x, ...)
```

Arguments

`x`	the beta distribution
`...`	not used

Value

nothing

`relative_risk_ve` *Calculate a vaccine effectiveness estimate based on a risk ratio*

Description

The RR method cannot be used in test negative designs where disease positive relates to vaccine treatable disease and disease negative relates to non vaccine treatable disease. It is only relevant in prospective designs with a vaccinated and unvaccinated group.

Usage

```
relative_risk_ve(
  vaccinatedCase,
  unvaccinatedCase,
  vaccinatedControl,
  unvaccinatedControl,
  confint = c(0.025, 0.975)
)
```

Arguments

`vaccinatedCase`	count of disease positive vaccine positive
`unvaccinatedCase`	count of disease positive vaccine negative
`vaccinatedControl`	count of disease negative vaccine positive
`unvaccinatedControl`	count of disease negative vaccine positive
`confint`	the confidence intervals

Value

a dataframe

Examples

```
tibble::tibble(
  N_vacc = 42240,
  N_unvacc = 42256,
  N_vacc_pn_pos = 49,
  N_unvacc_pn_pos = 90
) %>% dplyr::mutate(
 relative_risk_ve(N_vacc_pn_pos, N_unvacc_pn_pos, N_vacc-N_vacc_pn_pos, N_unvacc-N_unvacc_pn_pos)
)

# dplyr::bind_rows(lapply(
#   c("katz.log", "adj.log", "bailey", "koopman", "noether",  "sinh-1", "boot"),
```

```
#   function(m) {tibble::as_tibble(
#    1-DescTools::BinomRatioCI(N_vacc_pn_pos, N_vacc, N_unvacc_pn_pos, N_unvacc, method = m)
#   ) %>% dplyr::mutate(
#     method = m
#   )}
# ))
```

`rep.beta_dist` *Repeat a* `beta_dist`

Description

Repeat a `beta_dist`

Usage

```
## S3 method for class 'beta_dist'
rep(x, times, ...)
```

Arguments

`x`	a `beta_dist`
`times`	n
`...`	not used

Value

a `beta_dist_list`

`rogan_gladen` *True prevalence from apparent prevalence*

Description

This estimator runs into problems with small AP as the Rogan-Gladen conversion is really using expected apparent prevalence. Getting the expected value of the AP distribution is complex and the expected value given a single observation is not in general the ratio of positives / count. The expected apparent prevalence is never less than the specificity but the observed value often is. To deal with this the R-G estimator truncates at zero.

Usage

```
rogan_gladen(ap, sens, spec)
```

Arguments

ap	the expected apparent prevalence.
sens	the sensitivity of the test
spec	the specificity of the test

Value

the estimate of 'true prevalence'

Examples

```
rogan_gladen(50/200, 0.75, 0.97)
```

---

sens_prior *The default prior for specificity*

Description

If undefined this is 0.70 (0.11 - 1.00). This can be set with `options(testerror.sens_prior = beta_dist(p=??, n=??))`

Usage

```
sens_prior()
```

Value

a `beta_dist`

---

spec_prior *The default prior for specificity*

Description

If undefined this is 0.98 (0.71 - 1.00). This can be set with `options(testerror.spec_prior = beta_dist(p=??, n=??))`

Usage

```
spec_prior()
```

Value

a `beta_dist`

`true_panel_prevalence` *Calculate an estimate of true prevalence for a single panel and components*

Description

Uses apparent prevalence, and uncertain estimates of test sensitivity and test specificity for the 3 methods described in Supplementary 2. This function works for a single panel per dataframe, multiple panels will need to call this function multiple times in a `group_modify`.

Usage

```
true_panel_prevalence(
  test_results = testerror::.input_data,
  false_pos_controls = NULL,
  n_controls = NULL,
  false_neg_diseased = NULL,
  n_diseased = NULL,
  ...,
  sens = NULL,
  spec = NULL,
  panel_name = "Panel",
  confint = 0.95,
  method = c("rogan-gladen", "lang-reiczigel", "bayes"),
  na.rm = TRUE
)
```

Arguments

`test_results` A dataframe containing the following columns:

- id (character) - the patient identifier
- test (factor) - the test type
- result (logical) - the test result

Ungrouped.

No default value.

`false_pos_controls`
the number of positives that appeared in the specificity disease-free control group. These are by definition false positives. This is (1-specificity)*n_controls

`n_controls` the number of controls in the specificity disease-free control group.

`false_neg_diseased`
the number of negatives that appeared in the sensitivity confirmed disease group. These are by definition false negatives. This is (1-sensitivity)*n_diseased

`n_diseased` the number of confirmed disease cases in the sensitivity control group.

`...` Arguments passed on to uncertain_panel_rogan_gladen, prevalence_panel_lang_reiczigel, bayesian_panel_complex_model, bayesian_panel_true_prevalence_model, bayesian_panel_simpler_model, bayesian_panel_logit_model

`samples` number fo random draws of sensitivity and specificity

`fmt` a `sprintf` formatting string accepting 3 numbers

`panel_sens` the prior sensitivity of the panel as a `beta_dist` (optional)

`panel_spec` the prior specificity of the panel as a `beta_dist` (optional)

`cache_result` save the result of the sampling in memory for the current session

`chains` A positive integer specifying the number of Markov chains. The default is 4.

`iter` A positive integer specifying the number of iterations for each chain (including warmup). The default is 2000.

`warmup` A positive integer specifying the number of warmup (aka burnin) iterations per chain. If step-size adaptation is on (which it is by default), this also controls the number of iterations for which adaptation is run (and hence these warmup samples should not be used for inference). The number of warmup iterations should be smaller than `iter` and the default is `iter/2`.

`model_type` The bayesian model used one of "logit", "simpler", "complex"

`sens` the prior sensitivity of the test as a `beta_dist`.

`spec` the prior specificity of the test as a `beta_dist`.

`panel_name` the name of the panel for combined result

`confint` confidence interval limits

`method` one of:

- "lang-reiczigel": Frequentist confidence limits
- "bayes": Bayesian analysis
- "rogan-gladen": Rogan gladen with uncertainty

`na.rm` exclude patients with missing results

Value

A dataframe containing the following columns:

- test (character) - the name of the test or panel
- prevalence.lower (numeric) - the lower estimate
- prevalence.median (numeric) - the median estimate
- prevalence.upper (numeric) - the upper estimate
- prevalence.method (character) - the method of estimation
- prevalence.label (character) - a fomatted label of the true prevalence estimate with CI

Ungrouped.

No default value.

Examples

```
tmp = testerror:::panel_example()
true_panel_prevalence(
  test_results = tmp$samples %>% dplyr::select(id,test,result = observed),
  false_pos_controls = tmp$performance$false_pos_controls,
  n_controls = tmp$performance$n_controls,
  false_neg_diseased = tmp$performance$false_neg_diseased,
  n_diseased = tmp$performance$n_diseased,
  method = "rogan-gladen"
)
```

---

true_prevalence *Vectorised true prevalence estimates*

Description

Calculate an estimate of true prevalence from apparent prevalence, and uncertain estimates of test sensitivity and test specificity, using one of 3 methods.

Usage

```
true_prevalence(
  pos_obs,
  n_obs,
  false_pos_controls = NULL,
  n_controls = NULL,
  false_neg_diseased = NULL,
  n_diseased = NULL,
  confint = 0.95,
  method = c("lang-reiczigel", "rogan-gladen", "bayes"),
  ...,
  spec = NULL,
  sens = NULL
)
```

Arguments

pos_obs	the number of positive observations for a given test
n_obs	the number of observations for a given test
false_pos_controls	the number of positives that appeared in the specificity disease-free control group. These are by definition false positives. This is (1-specificity)*n_controls
n_controls	the number of controls in the specificity disease-free control group.
false_neg_diseased	the number of negatives that appeared in the sensitivity confirmed disease group. These are by definition false negatives. This is (1-sensitivity)*n_diseased

n_diseased	the number of confirmed disease cases in the sensitivity control group.
confint	confidence interval limits
method	one of: • "lang-reiczigel": Frequentist confidence limits: see prevalence_lang_reiczigel() • "rogan-gladen": Rogan gladen incorporating uncertainty with resampling: see uncertain_rogan_gladen() • "bayes": Bayesian analysis: see bayesian_component_simpler_model()
...	Arguments passed on to uncertain_rogan_gladen samples number fo random draws of sensitivity and specificity fmt a sprintf formatting string accepting 3 numbers seed set seed for reproducibility
spec	the prior specificity of the test as a beta_dist.
sens	the prior sensitivity of the test as a beta_dist.

Value

A dataframe containing the following columns:

- test (character) - the name of the test or panel
- prevalence.lower (numeric) - the lower estimate
- prevalence.median (numeric) - the median estimate
- prevalence.upper (numeric) - the upper estimate
- prevalence.method (character) - the method of estimation
- prevalence.label (character) - a fomatted label of the true prevalence estimate with CI

Ungrouped.

No default value.

Examples

```
true_prevalence(c(1:50), 200, 2, 800, 25, 75)
true_prevalence(c(1:10)*2, 200, 25, 800, 1, 6, method="rogan-gladen")
true_prevalence(c(1:10)*2, 200, 5, 800, 1, 6, method="bayes")
```

---

uncertain_panel_rogan_gladen

Rogan-Gladen true prevalence for panel with resampling

Description

Uses resampling to incorporate uncertainty of sensitivity and specificity into an estimate of true prevalence from a given value of apparent prevalence.

Usage

```
uncertain_panel_rogan_gladen(
  panel_pos_obs,
  panel_n_obs,
  pos_obs,
  n_obs,
  false_pos_controls = NULL,
  n_controls = NULL,
  false_neg_diseased = NULL,
  n_diseased = NULL,
  ...,
  sens = sens_prior(),
  spec = spec_prior(),
  confint = 0.95,
  fmt = "%1.2f%% [%1.2f%% - %1.2f%%]",
  samples = 1000
)
```

Arguments

panel_pos_obs	the number of positive observations for a given panel of tests
panel_n_obs	the number of observations for each component test
pos_obs	the number of positive observations for a given test
n_obs	the number of observations for a given test
false_pos_controls	the number of positives that appeared in the specificity disease-free control group. These are by definition false positives. This is (1-specificity)*n_controls
n_controls	the number of controls in the specificity disease-free control group.
false_neg_diseased	the number of negatives that appeared in the sensitivity confirmed disease group. These are by definition false negatives. This is (1-sensitivity)*n_diseased
n_diseased	the number of confirmed disease cases in the sensitivity control group.
...	Arguments passed on to uncertain_rogan_gladen seed set seed for reproducibility
sens	the prior sensitivity of the test as a beta_dist.
spec	the prior specificity of the test as a beta_dist.
confint	confidence interval limits
fmt	a sprintf formatting string accepting 3 numbers
samples	number fo random draws of sensitivity and specificity

Details

This is not vectorised

Value

the expected value of apparent prevalence

`uncertain_panel_sens_estimator`

Propagate component test sensitivity and specificity into panel specificity assuming a known set of observations of component apparent prevalence

---

Description

Propagate component test sensitivity and specificity into panel specificity assuming a known set of observations of component apparent prevalence

Usage

```
uncertain_panel_sens_estimator(
  pos_obs,
  n_obs,
  false_pos_controls = NULL,
  n_controls = NULL,
  false_neg_diseased = NULL,
  n_diseased = NULL,
  ...,
  sens = sens_prior(),
  spec = spec_prior(),
  samples = 1000,
  fit_beta = FALSE
)
```

Arguments

`pos_obs`	the number of positive observations for a given test
`n_obs`	the number of observations for a given test
`false_pos_controls`	the number of positives that appeared in the specificity disease-free control group. These are by definition false positives. This is (1-specificity)*n_controls
`n_controls`	the number of controls in the specificity disease-free control group.
`false_neg_diseased`	the number of negatives that appeared in the sensitivity confirmed disease group. These are by definition false negatives. This is (1-sensitivity)*n_diseased
`n_diseased`	the number of confirmed disease cases in the sensitivity control group.
`...`	not used
`sens`	the prior sensitivity of the test as a `beta_dist`.

spec	the prior specificity of the test as a beta_dist.
samples	number fo random draws of sensitivity and specificity
fit_beta	return the result as a beta_dist object?

Value

a vector of possible sensitivity values

Examples

```
uncertain_panel_sens_estimator(
  pos_obs = c(30,10,20,10,5), n_obs=1000,
  false_pos_controls = c(20,15,15,15,15), n_controls = c(800,800,800,800,800),
  false_neg_diseased = c(20,25,20,20,15), n_diseased = c(100,100,100,100,100),
  fit_beta = TRUE)
```

---

uncertain_panel_spec *Propagate component test specificity into panel specificity*

Description

Propagate component test specificity into panel specificity

Usage

```
uncertain_panel_spec(
  false_pos_controls = NULL,
  n_controls = NULL,
  ...,
  spec = spec_prior(),
  samples = 1000,
  na.rm = FALSE,
  fit_beta = FALSE
)
```

Arguments

false_pos_controls	the number of positives that appeared in the specificity disease-free control group. These are by definition false positives. This is (1-specificity)*n_controls
n_controls	the number of controls in the specificity disease-free control group.
...	not used
spec	the prior specificity of the test as a beta_dist.
samples	number fo random draws of sensitivity and specificity
na.rm	remove missing values
fit_beta	return the result as a beta_dist object?

Value

a vector of possible specificities for the panel or a fitted `beta_dist`

Examples

```
uncertain_panel_spec(c(2,3,4,2,2), c(800,800,800,800,800), fit_beta=TRUE)
```

---

uncertain_rogan_gladen

True prevalence from apparent prevalence with uncertainty

Description

Uses resampling to incorporate uncertainty of sensitivity and specificity into an estimate of true prevalence from a given value of apparent prevalence.

Usage

```
uncertain_rogan_gladen(
  pos_obs,
  n_obs,
  false_pos_controls = NULL,
  n_controls = NULL,
  false_neg_diseased = NULL,
  n_diseased = NULL,
  ...,
  spec = spec_prior(),
  sens = sens_prior(),
  samples = 1000,
  confint = 0.95,
  fmt = "%1.2f%% [%1.2f%% – %1.2f%%]",
  seed = NA
)
```

Arguments

`pos_obs`	the number of positive observations for a given test
`n_obs`	the number of observations for a given test
`false_pos_controls`	the number of positives that appeared in the specificity disease-free control group. These are by definition false positives. This is (1-specificity)*n_controls
`n_controls`	the number of controls in the specificity disease-free control group.
`false_neg_diseased`	the number of negatives that appeared in the sensitivity confirmed disease group. These are by definition false negatives. This is (1-sensitivity)*n_diseased

n_diseased	the number of confirmed disease cases in the sensitivity control group.
...	not used
spec	the prior specificity of the test as a beta_dist.
sens	the prior sensitivity of the test as a beta_dist.
samples	number fo random draws of sensitivity and specificity
confint	confidence interval limits
fmt	a sprintf formatting string accepting 3 numbers
seed	set seed for reproducibility

Value

the expected value of apparent prevalence

Examples

```
uncertain_rogan_gladen(
  pos_obs = 20, n_obs = 1000,
  false_pos_controls = 10, n_controls = 800,
  false_neg_diseased = 20, n_diseased = 100)

uncertain_rogan_gladen(
  pos_obs = 5, n_obs = 1000,
  sens = beta_dist(p=0.75,n=200),
  spec = beta_dist(p=0.9975, n=800))

uncertain_rogan_gladen(
  pos_obs = c(5,10), n_obs = c(1000,1000),
  false_pos_controls = c(2,1), n_controls = c(800,800),
  false_neg_diseased = c(25,20),n_diseased = c(100,100))
```

---

underestimation_threshold

Test underestimation limit

Description

For a given sensitivity and specificity this give the critical threshold after which test error introduces underestimation rather than over estimation

Usage

```
underestimation_threshold(sens, spec)
```

Arguments

sens	the sensitivity of the test
spec	the specificity of the test

Value

the value where apparent prevalence equals true prevalence

Examples

```
tmp1 = underestimation_threshold(0.75, 0.97)
tmp2 = rogan_gladen(tmp1, 0.75, 0.97)
if (abs(tmp1-tmp2) > 0.0000001) stop("error")
```

---

uniform_prior *A uniform prior*

Description

A uniform prior

Usage

```
uniform_prior()
```

Value

a `beta_dist`

---

uninformed_prior *Uninformative prior*

Description

Uninformative prior

Usage

```
uninformed_prior()
```

Value

a `beta_dist`

update_posterior *Update the posterior of a* beta_dist

Description

Update the posterior of a beta_dist

Usage

```
update_posterior(x, ..., pos = NULL, neg = NULL, n = NULL)
```

Arguments

x	a beta_dist or beta_dist_list acting as the prior
...	not used
pos	positive observation(s)
neg	negative observation(s)
n	number observations

Value

a new beta_dist o beta_dist_list

Examples

```
update_posterior(beta_dist(shape1=1,shape2=1), neg=10, n=30)
```

update_posterior.beta_dist *Update the posterior of a* beta_dist

Description

Update the posterior of a beta_dist

Usage

```
## S3 method for class 'beta_dist'
update_posterior(x, ..., pos = NULL, neg = NULL, n = NULL)
```

Arguments

x	a beta_dist or beta_dist_list acting as the prior
...	not used
pos	positive observation(s)
neg	negative observation(s)
n	number observations

Value

a new beta_dist o beta_dist_list

Examples

```
update_posterior(beta_dist(shape1=1,shape2=1), neg=10, n=30)
```

---

update_posterior.beta_dist_list
Update the posterior of a beta_dist

Description

Update the posterior of a beta_dist

Usage

```
## S3 method for class 'beta_dist_list'
update_posterior(x, ..., pos = NULL, neg = NULL, n = NULL)
```

Arguments

x	a beta_dist or beta_dist_list acting as the prior
...	not used
pos	positive observation(s)
neg	negative observation(s)
n	number observations

Value

a new beta_dist o beta_dist_list

Examples

```
update_posterior(beta_dist(shape1=1,shape2=1), neg=10, n=30)
```

Index