

# Package: testerror (via r-universe)

October 29, 2024

**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 ( $\geq 2.18.1$ ), stats, extraDistr, interfacr ( $\geq 0.1.7$ ), pkgutils ( $\geq 0.1.0$ ), BH ( $\geq 1.66.0$ ), Rcpp ( $\geq 0.12.0$ ), RcppEigen ( $\geq 0.3.3.3.0$ ), RcppParallel ( $\geq 5.0.1$ ), StanHeaders ( $\geq 2.18.0$ )

**Remotes** github::bristol-vaccine-centre/interfacr,  
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 ( $\geq 3.4.0$ )

**LazyData** true

**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	3
.input_panel_data	3
.output_data	4
apparent_prevalence	5
as_tibble.beta_dist	5
as_tibble.beta_dist_list	6
bayesian_component_logit_model	6
bayesian_component_simpler_model	8
bayesian_panel_complex_model	9
bayesian_panel_logit_model	11
bayesian_panel_simpler_model	12
bayesian_panel_true_prevalence_model	14
bayesian_true_prevalence_model	16
beta_dist	17
beta_fit	18
beta_params	18
ci_to_logitnorm	19
format.beta_dist	20
format.beta_dist_list	20
fp_p_value	21
fp_signif_level	22
get_beta_shape	23
get_beta_shape.beta_dist	24
get_beta_shape.beta_dist_list	24
inv_logit	25
length.beta_dist	25
length.beta_dist_list	26
logit	26
odds_ratio_ve	27
optimal_performance	28
panel_prevalence	29
panel_sens	29
panel_sens_estimator	30
panel_spec	31
prevalence_lang_reiczigel	31
prevalence_panel_lang_reiczigel	32
print.beta_dist	34
print.beta_dist_list	34
relative_risk_ve	35
rep.beta_dist	36
rogan_gladen	36
sens_prior	37
spec_prior	37
true_panel_prevalence	38
true_prevalence	40
uncertain_panel_rogan_gladen	41

<i>.input_data</i>	3
uncertain_panel_sens_estimator . . . . .	43
uncertain_panel_spec . . . . .	44
uncertain_rogan_gladen . . . . .	45
underestimation_threshold . . . . .	46
uniform_prior . . . . .	47
uninformed_prior . . . . .	47
update_posterior . . . . .	48
update_posterior.beta_dist . . . . .	48
update_posterior.beta_dist_list . . . . .	49
<b>Index</b>	<b>50</b>

---

<i>.input_data</i>	<i>Dataframe format for component test results</i>
--------------------	--

---

**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.

---

<i>.input_panel_data</i>	<i>Dataframe format for panel test results</i>
--------------------------	--

---

**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	<i>Dataframe format for true prevalence results</i>
--------------	---

---

**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 formatted 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 <a href="#">as_tibble.beta_dist</a> 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

<code>pos_obs</code>	the number of positive observations for a given test
<code>n_obs</code>	the number of observations for a given test
<code>false_pos_controls</code>	the number of positives that appeared in the specificity disease-free control group. These are by definition false positives. This is $(1 - \text{specificity}) * n\_controls$
<code>n_controls</code>	the number of controls in the specificity disease-free control group.
<code>false_neg_diseased</code>	the number of negatives that appeared in the sensitivity confirmed disease group. These are by definition false negatives. This is $(1 - \text{sensitivity}) * n\_diseased$
<code>n_diseased</code>	the number of confirmed disease cases in the sensitivity control group.
<code>...</code>	not used
<code>sens</code>	the prior sensitivity of the test as a <code>beta_dist</code> .
<code>spec</code>	the prior specificity of the test as a <code>beta_dist</code> .
<code>confint</code>	confidence interval limits
<code>fmt</code>	a <code>sprintf</code> formatting string accepting 3 numbers
<code>chains</code>	A positive integer specifying the number of Markov chains. The default is 4.
<code>warmup</code>	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 <code>iter</code> and the default is <code>iter/2</code> .
<code>iter</code>	A positive integer specifying the number of iterations for each chain (including warmup). The default is 2000.
<code>cache_result</code>	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 $(1 - \text{specificity}) * n\_controls$
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 $(1 - \text{sensitivity}) * n\_diseased$
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



<code>fmt</code>	a <code>sprintf</code> formatting string accepting 3 numbers
<code>chains</code>	A positive integer specifying the number of Markov chains. The default is 4.
<code>warmup</code>	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 <code>iter</code> and the default is <code>iter/2</code> .
<code>iter</code>	A positive integer specifying the number of iterations for each chain (including warmup). The default is 2000.
<code>cache_result</code>	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**

<code>test_results</code>	A dataframe containing the following columns: <ul style="list-style-type: none"> <li>• <code>id</code> (character) - the patient identifier</li> <li>• <code>test</code> (factor) - the test type</li> <li>• <code>result</code> (logical) - the test result</li> </ul> Ungrouped. No default value.
<code>false_pos_controls</code>	the number of positives that appeared in the specificity disease-free control group. These are by definition false positives. This is $(1 - \text{specificity}) * n_{\text{controls}}$
<code>n_controls</code>	the number of controls in the specificity disease-free control group.
<code>false_neg_diseased</code>	the number of negatives that appeared in the sensitivity confirmed disease group. These are by definition false negatives. This is $(1 - \text{sensitivity}) * n_{\text{diseased}}$
<code>n_diseased</code>	the number of confirmed disease cases in the sensitivity control group.
<code>...</code>	not used
<code>sens</code>	the prior sensitivity of the test as a <code>beta_dist</code> .
<code>spec</code>	the prior specificity of the test as a <code>beta_dist</code> .
<code>panel_sens</code>	the prior sensitivity of the panel as a <code>beta_dist</code> (optional)
<code>panel_spec</code>	the prior specificity of the panel as a <code>beta_dist</code> (optional)
<code>panel_name</code>	the name of the panel for combined result
<code>confint</code>	confidence interval limits
<code>fmt</code>	a <code>sprintf</code> formatting string accepting 3 numbers
<code>chains</code>	A positive integer specifying the number of Markov chains. The default is 4.
<code>warmup</code>	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 <code>iter</code> and the default is <code>iter/2</code> .
<code>iter</code>	A positive integer specifying the number of iterations for each chain (including warmup). The default is 2000.
<code>cache_result</code>	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 $(1 - \text{specificity}) * n\_controls$
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 $(1 - \text{sensitivity}) * n_{\text{diseased}}$
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 $(1 - \text{specificity}) * n_{\text{controls}}$
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 $(1 - \text{sensitivity}) * n_{\text{diseased}}$
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 <code>iter</code> and the default is <code>iter/2</code> .
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  $(1 - \text{specificity}) * n\_controls$

<code>n_controls</code>	the number of controls in the specificity disease-free control group.
<code>false_neg_diseased</code>	the number of negatives that appeared in the sensitivity confirmed disease group. These are by definition false negatives. This is $(1 - \text{sensitivity}) * n_{\text{diseased}}$
<code>n_diseased</code>	the number of confirmed disease cases in the sensitivity control group.
<code>sens</code>	the prior sensitivity of the test as a <code>beta_dist</code> .
<code>spec</code>	the prior specificity of the test as a <code>beta_dist</code> .
<code>confint</code>	confidence interval limits
<code>fmt</code>	a <code>sprintf</code> formatting string accepting 3 numbers
<code>chains</code>	A positive integer specifying the number of Markov chains. The default is 4.
<code>iter</code>	A positive integer specifying the number of iterations for each chain (including warmup). The default is 2000.
<code>warmup</code>	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 <code>iter</code> and the default is <code>iter/2</code> .
<code>panel_pos_obs</code>	the number of positive observations for a given panel of tests
<code>panel_n_obs</code>	the number of observations for each component test
<code>test_names</code>	a vector of the component test names in desired order
<code>pos_obs</code>	the number of positive observations for a given test
<code>n_obs</code>	the number of observations for a given test
<code>model_type</code>	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  $(1 - \text{specificity}) * n\_controls$

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  $(1 - \text{sensitivity}) * n\_diseased$

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	<i>Generate a beta distribution out of probabilities, or positive and negative counts</i>
-----------	---

---

**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	<i>Fit a beta distribution to data using method of moments</i>
----------	--

---

**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	<i>Generate concave beta distribution parameters from mean and confidence intervals</i>
-------------	---

---

**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	<i>Generate mu and sigma parameters for a logitnormal distribution</i>
-----------------	--

---

**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 <a href="#">format.beta_dist</a>
glue	a glue spec taking any of shape1, shape2, conc, mean, median, upper, lower

**Value**

nothing

---

fp_p_value	<i>Significance of an uncertain test result</i>
------------	---

---

### 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 $(1 - \text{specificity}) * n\_controls$
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	<i>Identify the minimum number of positive test result observations needed to be confident the disease has a non-zero prevalence.</i>
-----------------	---

---

**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 $(1 - \text{specificity}) * n\_controls$
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	<i>Get a parameter of the beta_dist</i>
----------------	---

---

**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	<i>The inverse logit function</i>
-----------	-----------------------------------

---

**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	<i>Detect the length of a beta distribution</i>
------------------	---

---

**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**

<code>x</code>	the beta distribution list
<code>...</code>	not used

**Value**

the length of the list

---

`logit` *The logit function*

---

**Description**

The logit function

**Usage**

```
logit(x)
```

**Arguments**

<code>x</code>	a number between 0 and 1
----------------	--------------------------

**Value**

a number between  $-\text{Inf}$  and  $\text{Inf}$

odds\_ratio\_ve

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

This assumes that  $OR \sim RR$  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  $OR \sim RR$  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	<i>Expected test panel prevalence assuming independence</i>
------------------	---

---

**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	<i>Test panel combination sensitivity</i>
------------	---

---

**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 are 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_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 are 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_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	<i>Test panel combination specificity</i>
------------	---

---

### 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	<i>True prevalence from apparent prevalence with uncertainty</i>
---------------------------	--

---

### 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 $(1 - \text{specificity}) * n_{\text{controls}}$
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 $(1 - \text{sensitivity}) * n_{\text{diseased}}$
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 $(1 - \text{specificity}) * n\_controls$
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 $(1 - \text{sensitivity}) * n\_diseased$
n_diseased	the number of confirmed disease cases in the sensitivity control group.
...	Arguments passed on to <a href="#">prevalence_lang_reiczigel</a>
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	<i>Calculate a vaccine effectiveness estimate based on a risk ratio</i>
------------------	---

---

### 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	<i>Repeat a beta_dist</i>
---------------	---------------------------

---

### 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	<i>True prevalence from apparent prevalence</i>
--------------	---

---

### 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	<i>The default prior for specificity</i>
------------	--

---

**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	<i>The default prior for specificity</i>
------------	--

---

**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

<code>test_results</code>	A dataframe containing the following columns: <ul style="list-style-type: none"> <li>• <code>id</code> (character) - the patient identifier</li> <li>• <code>test</code> (factor) - the test type</li> <li>• <code>result</code> (logical) - the test result</li> </ul> Ungrouped. No default value.
<code>false_pos_controls</code>	the number of positives that appeared in the specificity disease-free control group. These are by definition false positives. This is $(1 - \text{specificity}) * n_{\text{controls}}$
<code>n_controls</code>	the number of controls in the specificity disease-free control group.
<code>false_neg_diseased</code>	the number of negatives that appeared in the sensitivity confirmed disease group. These are by definition false negatives. This is $(1 - \text{sensitivity}) * n_{\text{diseased}}$
<code>n_diseased</code>	the number of confirmed disease cases in the sensitivity control group.

...	Arguments passed on to <a href="#">uncertain_panel_rogan_gladen</a> , <a href="#">prevalence_panel_lang_reiczigel</a> , <a href="#">bayesian_panel_complex_model</a> , <a href="#">bayesian_panel_true_prevalence_model</a> , <a href="#">bayesian_panel_simpler_model</a> , <a href="#">bayesian_panel_logit_model</a>
	<code>samples</code> number fo random draws of sensitivity and specificity
	<code>fmt</code> a <code>sprintf</code> formatting string accepting 3 numbers
	<code>panel_sens</code> the prior sensitivity of the panel as a <code>beta_dist</code> (optional)
	<code>panel_spec</code> the prior specificity of the panel as a <code>beta_dist</code> (optional)
	<code>cache_result</code> save the result of the sampling in memory for the current session
	<code>chains</code> A positive integer specifying the number of Markov chains. The default is 4.
	<code>iter</code> A positive integer specifying the number of iterations for each chain (including warmup). The default is 2000.
	<code>warmup</code> 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 <code>iter</code> and the default is <code>iter/2</code> .
	<code>model_type</code> The bayesian model used one of "logit", "simpler", "complex"
<code>sens</code>	the prior sensitivity of the test as a <code>beta_dist</code> .
<code>spec</code>	the prior specificity of the test as a <code>beta_dist</code> .
<code>panel_name</code>	the name of the panel for combined result
<code>confint</code>	confidence interval limits
<code>method</code>	one of: <ul style="list-style-type: none"> <li>• "lang-reiczigel": Frequentist confidence limits</li> <li>• "bayes": Bayesian analysis</li> <li>• "rogan-gladen": Rogan gladen with uncertainty</li> </ul>
<code>na.rm</code>	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	<i>Vectorised true prevalence estimates</i>
-----------------	---

---

**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 $(1 - \text{specificity}) * n\_controls$
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 $(1 - \text{sensitivity}) * n\_diseased$



n_diseased	the number of confirmed disease cases in the sensitivity control group.
confint	confidence interval limits
method	one of: <ul style="list-style-type: none"> <li>• "lang-reiczigel": Frequentist confidence limits: see prevalence_lang_reiczigel()</li> <li>• "rogan-gladen": Rogan gladen incorporating uncertainty with resampling: see uncertain_rogan_gladen()</li> <li>• "bayes": Bayesian analysis: see bayesian_component_simpler_model()</li> </ul>
...	Arguments passed on to <a href="#">uncertain_rogan_gladen</a>
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 $(1 - \text{specificity}) * n_{\text{controls}}$
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 $(1 - \text{sensitivity}) * n_{\text{diseased}}$
n_diseased	the number of confirmed disease cases in the sensitivity control group.
...	Arguments passed on to <a href="#">uncertain_rogan_gladen</a>
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 $(1 - \text{specificity}) * n\_controls$
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 $(1 - \text{sensitivity}) * n\_diseased$
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  $(1 - \text{specificity}) * n\_controls$

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 $(1-\text{specificity}) \times n_{\text{controls}}$
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 $(1-\text{sensitivity}) \times n_{\text{diseased}}$

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) > 0.000001) stop("error")
```

---

uniform_prior	<i>A uniform prior</i>
---------------	------------------------

---

**Description**

A uniform prior

**Usage**

```
uniform_prior()
```

**Value**

a beta\_dist

---

uninformed_prior	<i>Uninformative prior</i>
------------------	----------------------------

---

**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 or 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

- \* **data**
  - .input\_data, 3
  - .input\_panel\_data, 3
  - .output\_data, 4
- .input\_data, 3
- .input\_panel\_data, 3
- .output\_data, 4
- apparent\_prevalence, 5
- as\_tibble.beta\_dist, 5, 6
- as\_tibble.beta\_dist\_list, 6
  
- bayesian\_component\_logit\_model, 6, 16
- bayesian\_component\_simpler\_model, 8, 16
- bayesian\_panel\_complex\_model, 9, 14, 39
- bayesian\_panel\_logit\_model, 11, 14, 39
- bayesian\_panel\_simpler\_model, 12, 14, 39
- bayesian\_panel\_true\_prevalence\_model, 14, 39
- bayesian\_true\_prevalence\_model, 16
- beta\_dist, 17
- beta\_fit, 18
- beta\_params, 18
  
- ci\_to\_logitnorm, 19
  
- format.beta\_dist, 20, 20
- format.beta\_dist\_list, 20
- fp\_p\_value, 21
- fp\_signif\_level, 22
  
- get\_beta\_shape, 23
- get\_beta\_shape.beta\_dist, 24
- get\_beta\_shape.beta\_dist\_list, 24
  
- inv\_logit, 25
  
- length.beta\_dist, 25
- length.beta\_dist\_list, 26
- logit, 26
  
- odds\_ratio\_ve, 27
- optimal\_performance, 28
  
- panel\_prevalence, 29
- panel\_sens, 29
- panel\_sens\_estimator, 30
- panel\_spec, 31
- prevalence\_lang\_reiczigel, 31, 33
- prevalence\_panel\_lang\_reiczigel, 32, 39
- print.beta\_dist, 34
- print.beta\_dist\_list, 34
  
- relative\_risk\_ve, 35
- rep.beta\_dist, 36
- rogan\_gladen, 36
  
- sens\_prior, 37
- spec\_prior, 37
  
- true\_panel\_prevalence, 38
- true\_prevalence, 40
  
- uncertain\_panel\_rogan\_gladen, 39, 41
- uncertain\_panel\_sens\_estimator, 43
- uncertain\_panel\_spec, 44
- uncertain\_rogan\_gladen, 41, 42, 45
- underestimation\_threshold, 46
- uniform\_prior, 47
- uninformed\_prior, 47
- update\_posterior, 48
- update\_posterior.beta\_dist, 48
- update\_posterior.beta\_dist\_list, 49