Package 'riskyr'

February 19, 2018

Type Package

Title Rendering Risk Literacy more Transparent

Version 0.1.0

Date 2018-02-16

Author Hansjoerg Neth [aut, cre],
Felix Gaisbauer [aut],
Nico Gradwohl [aut],
Wolfgang Gaissmaier [aut]

Maintainer Hansjoerg Neth <h.neth@uni.kn>

Description Risk-related information can be expressed in terms of probabilities or frequencies. By providing a toolbox of methods and metrics, we compute, translate, and represent risk-related information in a variety of ways. By offering different, but complementary perspectives on the interplay between key parameters, 'riskyr' renders teaching and training of risk literacy more transparent.

Depends R (>= 3.2)

Imports diagram (>= 1.6.4), grid (>= 3.4.3), vcd (>= 1.4), utils (>= 3.4.3)

Suggests devtools, rmarkdown, knitr, roxygen2

Collate 'comp_util.R' 'init_txt.R' 'init_pal.R' 'init_prob.R' 'comp_prob_prob.R' 'init_freq.R' 'comp_freq_prob.R' 'init_num.R' 'init_prob_num.R' 'init_freq_num.R' 'comp_freq_freq.R' 'comp_prob_freq.R' 'comp_xxxx_prob.R' 'comp_popu.R' 'comp_accu.R' 'plot_mosaic.R' 'plot_icons.R' 'plot_tree.R' 'plot_fnet.R' 'plot_curve.R' 'plot_plane.R' 'data.R' 'read_data.R' 'riskyr_class.R' 'start_riskyr.R'

Encoding UTF-8

LazyData true

License GPL-2 | GPL-3

BugReports https://github.com/hneth/riskyr/issues

VignetteBuilder knitr

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-02-19 09:22:19 UTC

R topics documented:

`accu` *A list containing current accuracy information.*

Description

`accu` contains current accuracy information returned by the corresponding generating function `comp_accu`.

Usage

```
accu
```

Format

An object of class `list` of length 5.

Details

Current metrics include:

1. `acc`: Overall accuracy as the proportion (or probability) of correctly classifying cases or of `dec.cor` cases:
   `acc = dec.cor/N = (hi + cr)/(hi + mi + fa + cr)`
   Values range from 0 (no correct prediction) to 1 (perfect prediction).
2. `wacc`: Weighted accuracy, as a weighted average of the sensitivity `sens` (aka. hit rate `HR`, `TPR`, `power` or `recall`) and the the specificity `spec` (aka. `TNR`) in which `sens` is multiplied by a weighting parameter `w` (ranging from 0 to 1) and `spec` is multiplied by `w`'s complement `(1 - w)`:
   `wacc = (w * sens) + ((1 - w) * spec)`
   If `w = .50`, `wacc` becomes *balanced* accuracy `bacc`.
3. `mcc`: The Matthews correlation coefficient (with values ranging from -1 to +1):
   `mcc = ((hi * cr) - (fa * mi)) / sqrt((hi + fa) * (hi + mi) * (cr + fa) * (cr + mi))`
   A value of `mcc = 0` implies random performance; `mcc = 1` implies perfect performance.
   See Wikipedia: Matthews correlation coefficient for additional information.
4. `f1s`: The harmonic mean of the positive predictive value `PPV` (aka. `precision`) and the sensitivity `sens` (aka. hit rate `HR`, `TPR`, `power` or `recall`):
   `f1s =  2 * (PPV * sens) / (PPV + sens)`
   See Wikipedia: F1 score for additional information.

Note that some accuracy metrics can be interpreted as probabilities (e.g., `acc`) or correlations (e.g., `mcc`).

See Also

The corresponding generating function `comp_accu`; `num` for basic numeric parameters; `freq` for current frequency information; `prob` for current probability information; `txt` for current text settings.

Other lists containing current scenario information: `freq`, `num`, `pal`, `prob`, `txt`

Other metrics: `comp_accu`, `comp_acc`

Examples

```
accu <- comp_accu()  # => computes current accuracy information and saves results in accu
accu                 # => shows current accuracy information
```

as_pb *Display a percentage as a (rounded) probability.*

Description

as_pb is a function that displays a percentage perc as a probability (rounded to n.digits decimals).

Usage

```
as_pb(perc, n.digits = 4)
```

Arguments

perc	A percentage (as a scalar or vector of numeric values from 0 to 100).
n.digits	Number of decimal places to which percentage is rounded. Default: n.digits = 4.

Details

as_pb and its complement function as_pc allow toggling the display of numeric values between percentages and probabilities.

Value

A probability (as a numeric value).

See Also

is_perc verifies a percentage; is_prob verifies a probability; is_valid_prob_set verifies the validity of probability inputs; num contains basic numeric variables; init_num initializes basic numeric variables; prob contains current probability information; comp_prob computes current probability information; freq contains current frequency information; comp_freq computes current frequency information; comp_complement computes a probability's complement; comp_comp_pair computes pairs of complements.

Other utility functions: as_pc

Other display functions: as_pc

Examples

```
as_pb(1/3)          # => 0.0033
as_pb(as_pc(2/3))   # => 0.6667 (rounded to 4 decimals)

prob.seq <- seq(0, 1, by = 1/10)
perc.seq <- seq(0, 100, by = 10)

as_pc(prob.seq)  # =>   0  10  20  30  40  50  60  70  80  90 100
as_pb(perc.seq)  # => 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

```
perc.seq == as_pc(as_pb(perc.seq))            # => all TRUE
prob.seq == as_pb(as_pc(prob.seq))            # => some FALSE due to rounding errors!
round(prob.seq, 4) == as_pb(as_pc(prob.seq))  # => all TRUE (both rounded to 4 decimals)
```

---

as_pc *Display a probability as a (rounded) percentage.*

Description

as_pc is a function that displays a probability prob as a percentage (rounded to n.digits decimals).

Usage

```
as_pc(prob, n.digits = 2)
```

Arguments

prob	A probability (as a scalar or vector of numeric values from 0 to 1).
n.digits	Number of decimal places to which percentage is rounded. Default: n.digits = 2.

Details

as_pc and its complement function as_pb allow toggling the display of numeric values between percentages and probabilities.

Value

A percentage (as a numeric value).

See Also

is_prob verifies a probability; is_perc verifies a percentage; is_valid_prob_set verifies the validity of probability inputs; num contains basic numeric variables; init_num initializes basic numeric variables; prob contains current probability information; comp_prob computes current probability information; freq contains current frequency information; comp_freq computes current frequency information; comp_complement computes a probability's complement; comp_comp_pair computes pairs of complements.

Other utility functions: as_pb

Other display functions: as_pb

Examples

```
as_pc(.50)               # =>  50
as_pc(1/3)               # =>  33.33
as_pc(1/3, n.digits = 0)  # =>  33

as_pc(pi)                # => 314.16 + Warning that prob is not in range.
as_pc(as_pb(12.3))       # =>  12.3

prob.seq <- seq(0, 1, by = 1/10)
perc.seq <- seq(0, 100, by = 10)

as_pc(prob.seq)  # =>   0  10  20  30  40  50  60  70  80  90 100
as_pb(perc.seq)  # => 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

perc.seq == as_pc(as_pb(perc.seq))           # => all TRUE
prob.seq == as_pb(as_pc(prob.seq))           # => some FALSE due to rounding errors!
round(prob.seq, 4) == as_pb(as_pc(prob.seq))  # => all TRUE (both rounded to 4 decimals)
```

`comp_acc` *Compute overall accuracy (acc) from probabilities.*

Description

`comp_acc` computes overall accuracy `acc` from 3 essential probabilities `prev`, `sens`, and `spec`.

Usage

```
comp_acc(prev, sens, spec)
```

Arguments

`prev`	The condition's prevalence `prev` (i.e., the probability of condition being `TRUE`).
`sens`	The decision's sensitivity `sens` (i.e., the conditional probability of a positive decision provided that the condition is `TRUE`).
`spec`	The decision's specificity value `spec` (i.e., the conditional probability of a negative decision provided that the condition is `FALSE`).

Details

`comp_acc` uses probabilities (not frequencies) as inputs and returns a proportion (probability) without rounding.

Definition: `acc` is the overall accuracy as the proportion (or probability) of correctly classifying cases or of `dec.cor` cases:

`acc = dec.cor/N = (hi + cr)/(hi + mi + fa + cr)`

Values range from 0 (no correct prediction) to 1 (perfect prediction).

Importantly, correct decisions `dec.cor` are not necessarily positive decisions `dec.pos`.

Value

Overall accuracy acc as a proportion (probability). A warning is provided for NaN values.

See comp_accu and accu for accuracy metrics based on frequencies.

See Also

comp_sens and comp_PPV compute related probabilities; is_extreme_prob_set verifies extreme cases; comp_complement computes a probability's complement; is_complement verifies probability complements; comp_prob computes current probability information; prob contains current probability information; is_prob verifies probabilities.

Other functions computing probabilities: comp_FDR, comp_FOR, comp_NPV, comp_PPV, comp_accu, comp_comp_pair, comp_complement, comp_complete_prob_set, comp_fart, comp_mirt, comp_ppod, comp_prob_freq, comp_prob, comp_sens, comp_spec

Other metrics: accu, comp_accu

Examples

```
# ways to work:
comp_acc(.10, .200, .300)  # => acc = 0.29
comp_acc(.50, .333, .666)  # => acc = 0.4995

# watch out for vectors:
prev.range <- seq(0, 1, by = .1)
comp_acc(prev.range, .5, .5)  # => 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5

# watch out for extreme values:
comp_acc(1, 1, 1)  #  => 1
comp_acc(1, 1, 0)  #  => 1

comp_acc(1, 0, 1)  #  => 0
comp_acc(1, 0, 0)  #  => 0

comp_acc(0, 1, 1)  #  => 1
comp_acc(0, 1, 0)  #  => 0

comp_acc(0, 0, 1)  #  => 1
comp_acc(0, 0, 0)  #  => 0
```

---

comp_accu *Compute acccuracy of current classification results.*

Description

comp_accu computes current accuracy metrics from the 4 essential frequencies (hi, mi, fa, cr) that constitute the current confusion matrix and are contained in freq.

Usage

```
comp_accu(hi = freq$hi, mi = freq$mi, fa = freq$fa, cr = freq$cr,
  w = 0.5)
```

Arguments

hi	The number of hits hi (or true positives).
mi	The number of misses mi (or false negatives).
fa	The number of false alarms fa (or false positives).
cr	The number of correct rejections cr (or true negatives).
w	The weighting parameter w (from 0 to 1) for computing weighted accuracy wacc. Default: w = .50 (i.e., yielding balanced accuracy bacc).

Details

Currently computed metrics include:

1. acc: Overall accuracy as the proportion (or probability) of correctly classifying cases or of dec.cor cases:
   acc = dec.cor/N = (hi + cr)/(hi + mi + fa + cr)
   Values range from 0 (no correct prediction) to 1 (perfect prediction).
2. wacc: Weighted accuracy, as a weighted average of the sensitivity sens (aka. hit rate HR, TPR, power or recall) and the the specificity spec (aka. TNR) in which sens is multiplied by a weighting parameter w (ranging from 0 to 1) and spec is multiplied by w's complement (1 - w):
   wacc = (w * sens) + ((1 - w) * spec)
   If w = .50, wacc becomes *balanced* accuracy bacc.
3. mcc: The Matthews correlation coefficient (with values ranging from -1 to +1):
   mcc = ((hi * cr) - (fa * mi)) / sqrt((hi + fa) * (hi + mi) * (cr + fa) * (cr + mi))
   A value of mcc = 0 implies random performance; mcc = 1 implies perfect performance.
   See Wikipedia: Matthews correlation coefficient for additional information.
4. f1s: The harmonic mean of the positive predictive value PPV (aka. precision) and the sensitivity sens (aka. hit rate HR, TPR, power or recall):
   f1s = 2 * (PPV * sens) / (PPV + sens)
   See Wikipedia: F1 score for additional information.

Note that some accuracy metrics can be interpreted as probabilities (e.g., acc) or correlations (e.g., mcc).

Value

A list accu containing current accuracy metrics.

References

Consult Wikipedia: Confusion matrix for additional information.

See Also

The corresponding data frame ; num for basic numeric parameters; freq for current frequency information; txt for current text settings; pal for current color settings; popu for a table of the current population.

Other metrics: accu, comp_acc

Other functions computing probabilities: comp_FDR, comp_FOR, comp_NPV, comp_PPV, comp_acc, comp_comp_pair, comp_complement, comp_complete_prob_set, comp_fart, comp_mirt, comp_ppod, comp_prob_freq, comp_prob, comp_sens, comp_spec

Examples

```
comp_accu()  # => computes accuracy metrics for current default scenario
comp_accu(hi = 1, mi = 2, fa = 3, cr = 4)  # medium accuracy, but cr > hi

# Extreme cases:
comp_accu(hi = 1, mi = 1, fa = 1, cr = 1)  # random performance
comp_accu(hi = 1, mi = 0, fa = 0, cr = 1)  # perfect accuracy/optimal performance
comp_accu(hi = 0, mi = 1, fa = 1, cr = 0)  # zero accuracy/worst performance, but see f1s
comp_accu(hi = 1, mi = 0, fa = 0, cr = 0)  # perfect accuracy, but see wacc and mcc

# Effects of w:
comp_accu(hi = 3, mi = 2, fa = 1, cr = 4, w = 1/2)  # equal weights to sens and spec
comp_accu(hi = 3, mi = 2, fa = 1, cr = 4, w = 2/3)  # more weight to sens
comp_accu(hi = 3, mi = 2, fa = 1, cr = 4, w = 1/3)  # more weight to spec
```

---

comp_complement *Compute a probability's complement probability.*

Description

comp_complement computes the probability complement of a given probability prob.

Usage

```
comp_complement(prob)
```

Arguments

prob	A numeric probability value (in range from 0 to 1).

Details

The type and range of prob is verified with is_prob.

Value

A numeric probability value (in range from 0 to 1).

See Also

`is_complement` verifies numeric complements; `comp_comp_pair` returns a probability and its complement; `is_prob` verifies probabilities.

Other functions computing probabilities: `comp_FDR`, `comp_FOR`, `comp_NPV`, `comp_PPV`, `comp_accu`, `comp_acc`, `comp_comp_pair`, `comp_complete_prob_set`, `comp_fart`, `comp_mirt`, `comp_ppod`, `comp_prob_freq`, `comp_prob`, `comp_sens`, `comp_spec`

Examples

```
comp_complement(0)    # => 1
comp_complement(1)    # => 0

comp_complement(2)    # => NA + warning (beyond range)
comp_complement("p")  # => NA + warning (non-numeric)
```

---

comp_complete_prob_set

Compute a complete set of probabilities from valid probability inputs.

Description

`comp_complete_prob_set` is a function takes a valid set of (3 to 5) probabilities as inputs (as a vector) and returns the complete set of (3 essential and 2 optional) probabilities.

Usage

```
comp_complete_prob_set(prev, sens = NA, mirt = NA, spec = NA, fart = NA)
```

Arguments

`prev`	The condition's prevalence `prev` (i.e., the probability of condition being `TRUE`).
`sens`	The decision's sensitivity `sens` (i.e., the conditional probability of a positive decision provided that the condition is `TRUE`). `sens` is optional when its complement `mirt` is provided.
`mirt`	The decision's miss rate `mirt` (i.e., the conditional probability of a negative decision provided that the condition is `TRUE`). `mirt` is optional when its complement `sens` is provided.
`spec`	The decision's specificity value `spec` (i.e., the conditional probability of a negative decision provided that the condition is `FALSE`). `spec` is optional when its complement `fart` is provided.

`fart` The decision's false alarm rate `fart` (i.e., the conditional probability of a positive decision provided that the condition is `FALSE`). `fart` is optional when its complement `spec` is provided.

Details

Assuming that `is_valid_prob_set = TRUE` this function uses `comp_comp_pair` on the two optional pairs (i.e., `sens` and `mirt`, and `spec` and `fart`) and returns the complete set of 5 probabilities.

Value

A vector of 5 probabilities: `c(prev, sens, mirt, spec, fart)`.

See Also

`is_valid_prob_set` verifies a set of probability inputs; `is_extreme_prob_set` verifies extreme cases; `comp_comp_pair` computes pairs of complements; `is_complement` verifies numeric complements; `is_prob` verifies probabilities; `comp_prob` computes current probability information; `prob` contains current probability information; `init_num` initializes basic numeric variables; `num` contains basic numeric variables.

Other functions computing probabilities: `comp_FDR`, `comp_FOR`, `comp_NPV`, `comp_PPV`, `comp_accu`, `comp_acc`, `comp_comp_pair`, `comp_complement`, `comp_fart`, `comp_mirt`, `comp_ppod`, `comp_prob_freq`, `comp_prob`, `comp_sens`, `comp_spec`

Examples

```
# ways to work:
comp_complete_prob_set(1, .8, NA, .7, NA) # => 1.0 0.8 0.2 0.7 0.3
comp_complete_prob_set(1, NA, .8, NA, .4) # => 1.0 0.2 0.8 0.6 0.4

# watch out for:
comp_complete_prob_set(8)                  # => 8 NA NA NA NA + warnings
comp_complete_prob_set(8, 7, 6, 5, 4)     # => 8 7 6 5 4 + no warning (valid set assumed)
comp_complete_prob_set(8, .8, NA, .7, NA)  # => 8.0 0.8 0.2 0.7 0.3 + no warning (sic)
comp_complete_prob_set(8, 2, NA, 3, NA)    # => 8 2 NA 3 NA + no warning (sic)
```

`comp_comp_pair` *Compute a probability's (missing) complement and return both.*

Description

`comp_comp_pair` is a function that takes 0, 1, or 2 probabilities (`p1` and `p2`) as inputs. If either of them is missing (`NA`), it computes the complement of the other one and returns both probabilities.

Usage

```
comp_comp_pair(p1 = NA, p2 = NA)
```

Arguments

p1	A numeric probability value (in range from 0 to 1). p1 is optional when p2 is provided.
p2	A numeric probability value (in range from 0 to 1). p2 is optional when p1 is provided.

Details

comp_comp_pair does *nothing* when both arguments are provided (i.e., !is.na(p1) & !is.na(p2)) and only issues a warning if both arguments are missing (i.e., is.na(p1) & is.na(p2)).

Inputs are *not* verified: Use is_prob to verify that an input is a probability and is_complement to verify that two provided values actually are complements.

Value

A vector v containing 2 numeric probability values (in range from 0 to 1): v = c(p1, p2).

See Also

is_complement verifies numeric complements; is_valid_prob_set verifies sets of probabilities; comp_complete_prob_set completes valid sets of probabilities; is_extreme_prob_set verifies extreme cases; comp_prob computes current probability information; prob contains current probability information; is_prob verifies probabilities.

Other functions computing probabilities: comp_FDR, comp_FOR, comp_NPV, comp_PPV, comp_accu, comp_acc, comp_complement, comp_complete_prob_set, comp_fart, comp_mirt, comp_ppod, comp_prob_freq, comp_prob, comp_sens, comp_spec

Examples

```
# ways to work:
comp_comp_pair(1, 0)   # => 1 0
comp_comp_pair(0, 1)   # => 0 1
comp_comp_pair(1, NA)  # => 1 0
comp_comp_pair(NA, 1)  # => 0 1

# watch out for:
comp_comp_pair(NA, NA) # => NA NA + warning
comp_comp_pair(8, 8)   # => 8 8 + NO warning (as is_prob is not verified)
comp_comp_pair(1, 1)   # => 1 1 + NO warning (as is_complement is not verified)
```

comp_fart *Compute a decision's false alarm rate from its specificity.*

Description

comp_fart is a conversion function that takes a specificity spec – given as a probability (i.e., a numeric value in the range from 0 to 1) – as its input, and returns the corresponding false alarm rate fart – also as a probability – as its output.

Usage

```
comp_fart(spec)
```

Arguments

spec	The decision's specificity value spec as a probability.

Details

The false alarm rate fart and specificity spec are complements (fart = (1 - spec)) and both features of the decision process (e.g., a diagnostic test).

The function comp_fart is complementary to the conversion function comp_spec and uses the generic function comp_complement.

Value

The decision's false alarm rate fart as a probability.

See Also

comp_complement computes a probability's complement; is_complement verifies probability complements; comp_prob computes current probability information; prob contains current probability information; is_prob verifies probabilities.

Other functions computing probabilities: comp_FDR, comp_FOR, comp_NPV, comp_PPV, comp_accu, comp_acc, comp_comp_pair, comp_complement, comp_complete_prob_set, comp_mirt, comp_ppod, comp_prob_freq, comp_prob, comp_sens, comp_spec

Examples

```
comp_fart(2)                      # => NA + warning (beyond range)
comp_fart(1/3)                    # => 0.6666667
comp_fart(comp_complement(0.123))  # => 0.123
```

comp_FDR *Compute a decision's false detection rate (FDR) from probabilities.*

Description

comp_FDR computes the false detection rate FDR from 3 essential probabilities prev, sens, and spec.

Usage

```
comp_FDR(prev, sens, spec)
```

Arguments

prev	The condition's prevalence prev (i.e., the probability of condition being TRUE).
sens	The decision's sensitivity sens (i.e., the conditional probability of a positive decision provided that the condition is TRUE).
spec	The decision's specificity value spec (i.e., the conditional probability of a negative decision provided that the condition is FALSE).

Details

comp_FDR uses probabilities (not frequencies) and does not round results.

Value

The false detection rate FDR as a probability. A warning is provided for NaN values.

See Also

comp_sens and comp_PPV compute related probabilities; is_extreme_prob_set verifies extreme cases; comp_complement computes a probability's complement; is_complement verifies probability complements; comp_prob computes current probability information; prob contains current probability information; is_prob verifies probabilities.

Other functions computing probabilities: comp_FOR, comp_NPV, comp_PPV, comp_accu, comp_acc, comp_comp_pair, comp_complement, comp_complete_prob_set, comp_fart, comp_mirt, comp_ppod, comp_prob_freq, comp_prob, comp_sens, comp_spec

Examples

```
# (1) Ways to work:
comp_FDR(.50, .500, .500)  # => FDR = 0.5    = (1 - PPV)
comp_FDR(.50, .333, .666)  # => FDR = 0.5007 = (1 - PPV)
```

comp_FOR *Compute a decision's false omission rate (FOR) from probabilities.*

Description

comp_FOR computes the false omission rate FOR from 3 essential probabilities prev, sens, and spec.

Usage

```
comp_FOR(prev, sens, spec)
```

Arguments

prev	The condition's prevalence prev (i.e., the probability of condition being TRUE).
sens	The decision's sensitivity sens (i.e., the conditional probability of a positive decision provided that the condition is TRUE).
spec	The decision's specificity value spec (i.e., the conditional probability of a negative decision provided that the condition is FALSE).

Details

comp_FOR uses probabilities (not frequencies) and does not round results.

Value

The false omission rate FOR as a probability. A warning is provided for NaN values.

See Also

comp_spec and comp_NPV compute related probabilities; is_extreme_prob_set verifies extreme cases; comp_complement computes a probability's complement; is_complement verifies probability complements; comp_prob computes current probability information; prob contains current probability information; is_prob verifies probabilities.

Other functions computing probabilities: comp_FDR, comp_NPV, comp_PPV, comp_accu, comp_acc, comp_comp_pair, comp_complement, comp_complete_prob_set, comp_fart, comp_mirt, comp_ppod, comp_prob_freq, comp_prob, comp_sens, comp_spec

Examples

```
# (1) Ways to work:
comp_FOR(.50, .500, .500)  # => FOR = 0.5    = (1 - NPV)
comp_FOR(.50, .333, .666)  # => FOR = 0.5004 = (1 - NPV)
```

comp_freq *Compute frequencies from (3 essential) probabilities.*

Description

comp_freq computes frequencies (typically as rounded integers) given 3 basic probabilities – prev, sens, and spec – – for a population of N individuals. It returns a list of 9 frequencies freq as its output.

Usage

```
comp_freq(prev = num$prev, sens = num$sens, spec = num$spec, N = num$N,
  round = TRUE)
```

Arguments

prev	The condition's prevalence prev (i.e., the probability of condition being TRUE).
sens	The decision's sensitivity sens (i.e., the conditional probability of a positive decision provided that the condition is TRUE).
spec	The decision's specificity value spec (i.e., the conditional probability of a negative decision provided that the condition is FALSE).
N	The number of individuals in the population. If N is unknown (NA), a suitable minimum value is computed by comp_min_N.
round	A Boolean value that determines whether frequencies are rounded to the nearest integer. Default: round = TRUE.

Details

In addition to prev, both sens and spec are necessary arguments. If only their complements mirt or fart are known, use the wrapper function comp_freq_prob which also accepts mirt and fart as inputs (but requires that the entire set of provided probabilities is sufficient and consistent). Alternatively, use comp_complement, comp_comp_pair, or comp_complete_prob_set to obtain the 3 essential probabilities.

comp_freq is the frequency counterpart to the probability function comp_prob.

By default, comp_freq rounds frequencies to nearest integers to avoid decimal values in freq. Use round = FALSE to switch off rounding.

Key relationships:

1. to probabilities: A population of N individuals can be split into 2 subsets in 2 different ways:
   (a) by condition: The frequency cond.true depends on the prevalence prev and the frequency cond.false depends on the prevalence's complement 1 - prev.
   (b) by decision: The frequency dec.pos depends on the proportion of positive decisions ppod and the frequency dec.neg depends on the proportion of negative decisions 1 - ppod.

The population size N is a free parameter (independent of the essential probabilities prev, sens, and spec).

If N is unknown, a suitable minimum value can be computed by comp_min_N.

2. to other frequencies: In a population of size N the following relationships hold:
   - N = cond.true + cond.false (by condition)
   - N = dec.pos + dec.neg (by decision)
   - N = dec.cor + dec.err (by correspondence of decision to condition)
   - N = hi + mi + fa + cr (by condition x decision)

Value

A list freq containing 9 frequency values.

See Also

num contains basic numeric variables; init_num initializes basic numeric variables; freq contains current frequency information; prob contains current probability information; comp_prob computes current probability information; comp_complement computes a probability's complement; comp_comp_pair computes pairs of complements; comp_complete_prob_set completes valid sets of probabilities; comp_min_N computes a suitable population size N (if missing).

Other functions computing frequencies: comp_freq_freq, comp_freq_prob, comp_min_N, comp_popu, comp_prob_prob

Examples

```
comp_freq()                 # => ok, using current defaults
length(comp_freq())         # => 11

# Ways to succeed:
comp_freq(prev = 1, sens = 1, spec = 1, 100)  # => ok, N hits (TP)
comp_freq(prev = 1, sens = 1, spec = 0, 100)  # => ok, N hits
comp_freq(prev = 1, sens = 0, spec = 1, 100)  # => ok, N misses (FN)
comp_freq(prev = 1, sens = 0, spec = 0, 100)  # => ok, N misses
comp_freq(prev = 0, sens = 1, spec = 1, 100)  # => ok, N correct rejections (TN)
comp_freq(prev = 0, sens = 1, spec = 0, 100)  # => ok, N false alarms (FP)

# Watch out for:
comp_freq(prev = 1, sens = 1, spec = 1, N = NA)  # => ok, but warning that N = 1 was computed
comp_freq(prev = 1, sens = 1, spec = 1, N =  0)  # => ok, but all 0 + warning (extreme case: N hits)
comp_freq(prev = .5, sens = .5, spec = .5, N = 10, round = TRUE)   # => ok, rounded (see mi and fa)
comp_freq(prev = .5, sens = .5, spec = .5, N = 10, round = FALSE)  # => ok, not rounded

# Ways to fail:
comp_freq(prev = NA,  sens = 1, spec = 1,  100)   # => NAs + warning (prev NA)
comp_freq(prev = 1,  sens = NA, spec = 1,  100)   # => NAs + warning (sens NA)
comp_freq(prev = 1,  sens = 1,  spec = NA, 100)   # => NAs + warning (spec NA)
comp_freq(prev = 8,  sens = 1,  spec = 1,  100)   # => NAs + warning (prev beyond range)
comp_freq(prev = 1,  sens = 8,  spec = 1,  100)   # => NAs + warning (sens beyond range)
```

comp_freq_freq *Compute frequencies from (4 essential) frequencies.*

Description

comp_freq_freq computes current frequency information from 4 essential frequencies (hi, mi, fa, cr). It returns a list of 9 frequencies freq for a population of N individuals as its output.

Usage

```
comp_freq_freq(hi = freq$hi, mi = freq$mi, fa = freq$fa, cr = freq$cr)
```

Arguments

hi	The number of hits hi (or true positives).
mi	The number of misses mi (or false negatives).
fa	The number of false alarms fa (or false positives).
cr	The number of correct rejections cr (or true negatives).

Details

Key relationships:

- Other functions translating between representational formats:
  1. comp_freq_freq (defined here) is an analog to 3 other format conversion functions:
  2. comp_freq_prob computes current *frequency* information contained in freq from 3 essential probabilities (prev, sens, spec).
  3. comp_prob_freq computes current *probability* information contained in prob from 4 essential frequencies (hi, mi, fa, cr).
  4. comp_prob_prob computes current *probability* information contained in prob from 3 essential probabilities (prev, sens, spec).
- Two perspectives:
  A population of N individuals can be split into 2 subsets in 2 different ways:
  1. by condition:
     The frequency cond.true depends on the prevalence prev and the frequency cond.false depends on the prevalence's complement 1 - prev.
  2. by decision:
     The frequency dec.pos depends on the proportion of positive decisions ppod and the frequency dec.neg depends on the proportion of negative decisions 1 - ppod.

  The population size N is a free parameter (independent of the essential probabilities prev, sens, and spec).

  If N is unknown (NA), a suitable minimum value can be computed by comp_min_N.
- Combinations of frequencies:
  In a population of size N the following relationships hold:

1. N = cond.true + cond.false = (hi + mi) + (fa + cr) (by condition)
2. N = dec.pos + dec.neg = (hi + fa) + (mi + cr) (by decision)
3. N = hi + mi + fa + cr (by condition x decision)

The two perspectives (by condition vs. by decision) combine the 4 essential frequencies (i.e., hi, mi, fa, cr) in 2 different ways.

- Defining probabilities in terms of frequencies:

  Probabilities *are* – determine, describe, or are defined as – the relationships between frequencies. Thus, they can be computed as ratios between frequencies.

  The following relationships hold (and are used in computations):

  1. prevalence prev:
     `prev = cond.true/N = (hi + mi) / (hi + mi + fa + cr)`
  2. sensitivity sens:
     `sens = hi/cond.true = hi / (hi + mi) = (1 - mirt)`
  3. miss rate mirt:
     `mirt = mi/cond.true = mi / (hi + mi) = (1 - sens)`
  4. specificity spec:
     `spec = cr/cond.false = cr / (fa + cr) = (1 - fart)`
  5. false alarm rate fart:
     `fart = fa/cond.false = fa / (fa + cr) = (1 - spec)`
  6. proportion of positive decisions ppod:
     `ppod = dec.pos/N = (hi + fa) / (hi + mi + fa + cr)`
  7. positive predictive value PPV:
     `PPV = hi/dec.pos = hi / (hi + fa) = (1 - FDR)`
  8. negative predictive value NPV:
     `NPV = cr/dec.neg = cr / (mi + cr) = (1 - FOR)`
  9. false detection rate FDR:
     `FDR = fa/dec.pos = fa / (hi + fa) = (1 - PPV)`
  10. false omission rate FOR:
      `FOR = mi/dec.neg = mi / (mi + cr) = (1 - NPV)`

See Also

comp_freq_prob computes current frequency information from (3 essential) probabilities; comp_prob_freq computes current probability information from (4 essential) frequencies; comp_prob_prob computes current probability information from (3 essential) probabilities; num contains basic numeric parameters; init_num initializes basic numeric parameters; prob contains current probability information; comp_prob computes current probability information; freq contains current frequency information; comp_freq computes current frequency information; is_prob verifies probability inputs; is_freq verifies frequency inputs.

Other functions computing frequencies: comp_freq_prob, comp_freq, comp_min_N, comp_popu, comp_prob_prob

Other format conversion functions: comp_freq_prob, comp_prob_freq, comp_prob_prob

Examples

```
## Basics:
comp_freq_freq()
all.equal(freq, comp_freq_freq())  # => should be TRUE


## Circular chain:
# 1. Current numeric parameters:
num

# 2. Compute all 10 probabilities in prob (from essential probabilities):
prob <- comp_prob()
prob

# 3. Compute 9 frequencies in freq from probabilities:
freq <- comp_freq(round = FALSE)   # no rounding (to obtain same probabilities later)
freq

# 4. Compute 9 frequencies AGAIN (but now from frequencies):
freq_freq <- comp_freq_freq()

# 5. Check equality of results (steps 2. and 4.):
all.equal(freq, freq_freq)  # => should be TRUE!
```

`comp_freq_prob` *Compute frequencies from (3 essential) probabilities.*

Description

`comp_freq_prob` computes current frequency information from a sufficient and valid set of 3 essential probabilities (`prev`, and `sens` or its complement `mirt`, and `spec` or its complement `fart`). It returns a list of 9 frequencies `freq` for a population of `N` individuals as its output.

Usage

```
comp_freq_prob(prev = prob$prev, sens = prob$sens, mirt = NA,
  spec = prob$spec, fart = NA, tol = 0.01, N = freq$N, round = TRUE)
```

Arguments

`prev`	The condition's prevalence `prev` (i.e., the probability of condition being `TRUE`).
`sens`	The decision's sensitivity `sens` (i.e., the conditional probability of a positive decision provided that the condition is `TRUE`). `sens` is optional when its complement `mirt` is provided.
`mirt`	The decision's miss rate `mirt` (i.e., the conditional probability of a negative decision provided that the condition is `TRUE`). `mirt` is optional when its complement `sens` is provided.

spec	The decision's specificity value spec (i.e., the conditional probability of a negative decision provided that the condition is FALSE). spec is optional when its complement fart is provided.
fart	The decision's false alarm rate fart (i.e., the conditional probability of a positive decision provided that the condition is FALSE). fart is optional when its complement spec is provided.
tol	A numeric tolerance value for is_complement. Default: tol = .01.
N	The number of individuals in the population. If N is unknown (NA), a suitable minimum value is computed by comp_min_N.
round	A Boolean value that determines whether frequencies are rounded to the nearest integer. Default: round = TRUE.

Details

By default, the values of prev, sens, and spec are initialized to the probability information currently contained in prob.

comp_freq_prob is a wrapper function for the more basic function comp_freq, which only accepts the 3 essential probabilities (prev, sens, spec) as inputs.

When using comp_freq_prob with the arguments mirt and fart, their complements sens and spec must either be valid complements (as in is_complement) or NA.

The value of N uses the frequency information currently contained in freq as its default. If N is unknown (NA), a suitable minimum value can be computed by comp_min_N.

In addition to prev, both sens and spec are necessary arguments. If only their complements mirt or fart are known, first use comp_complement, comp_comp_pair, or comp_complete_prob_set to obtain the 3 essential probabilities.

By default, comp_freq_prob and comp_freq round frequencies to nearest integers to avoid decimal values in freq. Using the option round = FALSE turns off rounding.

Key relationships:

- Other functions translating between representational formats:
  1. comp_freq_prob (defined here) is a wrapper function for comp_freq and an analog to 3 other format conversion functions:
  2. comp_freq_freq computes current *frequency* information contained in freq from 4 essential frequencies (hi, mi, fa, cr).
  3. comp_prob_freq computes current *probability* information contained in prob from 4 essential frequencies (hi, mi, fa, cr).
  4. comp_prob_prob computes current *probability* information contained in prob from 3 essential probabilities (prev, sens, spec).
- Two perspectives:

  A population of N individuals can be split into 2 subsets in 2 different ways:

  1. by condition:
     The frequency cond.true depends on the prevalence prev and the frequency cond.false depends on the prevalence's complement 1 - prev.

2. by decision:
   The frequency dec.pos depends on the proportion of positive decisions ppod and the frequency dec.neg depends on the proportion of negative decisions 1 - ppod.

The population size N is a free parameter (independent of the essential probabilities prev, sens, and spec).

If N is unknown (NA), a suitable minimum value can be computed by comp_min_N.

- Combinations of frequencies:
  In a population of size N the following relationships hold:
  1. N = cond.true + cond.false = (hi + mi) + (fa + cr) (by condition)
  2. N = dec.pos + dec.neg = (hi + fa) + (mi + cr) (by decision)
  3. N = hi + mi + fa + cr (by condition x decision)

  The two perspectives (by condition vs. by decision) combine the 4 essential frequencies (i.e., hi, mi, fa, cr) in 2 different ways.
- Defining probabilities in terms of frequencies:
  Probabilities *are* – determine, describe, or are defined as – the relationships between frequencies. Thus, they can be computed as ratios between frequencies:
  1. prevalence prev:
     prev = cond.true/N = (hi + mi) / (hi + mi + fa + cr)
  2. sensitivity sens:
     sens = hi/cond.true = hi / (hi + mi) = (1 - mirt)
  3. miss rate mirt:
     mirt = mi/cond.true = mi / (hi + mi) = (1 - sens)
  4. specificity spec:
     spec = cr/cond.false = cr / (fa + cr) = (1 - fart)
  5. false alarm rate fart:
     fart = fa/cond.false = fa / (fa + cr) = (1 - spec)
  6. proportion of positive decisions ppod:
     ppod = dec.pos/N = (hi + fa) / (hi + mi + fa + cr)
  7. positive predictive value PPV:
     PPV = hi/dec.pos = hi / (hi + fa) = (1 - FDR)
  8. negative predictive value NPV:
     NPV = cr/dec.neg = cr / (mi + cr) = (1 - FOR)
  9. false detection rate FDR:
     FDR = fa/dec.pos = fa / (hi + fa) = (1 - PPV)
  10. false omission rate FOR:
      FOR = mi/dec.neg = mi / (mi + cr) = (1 - NPV)

Value

A list freq containing 9 frequency values.

See Also

comp_freq_freq computes current frequency information from (4 essential) frequencies; comp_prob_freq computes current probability information from (4 essential) frequencies; comp_prob_prob computes current probability information from (3 essential) probabilities; num contains basic numeric

variables; init_num initializes basic numeric variables; freq contains current frequency information; comp_freq computes current frequency information; prob contains current probability information; comp_prob computes current probability information; comp_complement computes a probability's complement; comp_comp_pair computes pairs of complements; comp_complete_prob_set completes valid sets of probabilities; comp_min_N computes a suitable population size N (if missing).

Other functions computing frequencies: comp_freq_freq, comp_freq, comp_min_N, comp_popu, comp_prob_prob

Other format conversion functions: comp_freq_freq, comp_prob_freq, comp_prob_prob

Examples

```
# Basics:
comp_freq_prob(prev = .1, sens = .9, spec = .8, N = 100)         # => ok: hi = 9, ... cr = 72.
# Same case with complements (using NAs to prevent defaults):
comp_freq_prob(prev = .1, sens = NA, mirt = .1, spec = NA, fart = .2, N = 100)  # => same result

comp_freq_prob()               # => ok, using probability info currently contained in prob
length(comp_freq_prob())           # => a list containing 9 frequencies
all.equal(freq, comp_freq_prob())  # => TRUE, unless prob has been changed after computing freq
freq <- comp_freq_prob()           # => computes frequencies and stores them in freq

# Ways to work:
comp_freq_prob(prev = 1, sens = 1, spec = 1, N = 101)  # => ok + warning: N hits (TP)

# Same case with complements (using NAs to prevent defaults):
comp_freq_prob(prev = 1, sens = NA, mirt = 0, spec = NA, fart = 0, N = 101)

comp_freq_prob(prev = 1, sens = 1, spec = 0, N = 102)  # => ok + warning: N hits (TP)
comp_freq_prob(prev = 1, sens = 0, spec = 1, N = 103)  # => ok + warning: N misses (FN)
comp_freq_prob(prev = 1, sens = 0, spec = 0, N = 104)  # => ok + warning: N misses (FN)
comp_freq_prob(prev = 0, sens = 1, spec = 1, N = 105)  # => ok + warning: N correct rejections (TN)

comp_freq_prob(prev = 0, sens = 1, spec = 0, N = 106)  # => ok + warning: N false alarms (FP)

# Same case with complements (using NAs to prevent defaults):
comp_freq_prob(prev = 0, sens = NA, mirt = 0,
               spec = NA, fart = 1, N = 106)  # => ok + warning: N false alarms (FP)

# Watch out for:
comp_freq_prob(prev = 1, sens = 1, spec = 1, N = NA)  # => ok + warning: N = 1 computed
comp_freq_prob(prev = 1, sens = 1, spec = 1, N =  0)  # => ok, but all 0 + warning (NPV = NaN)
comp_freq_prob(prev = .5, sens = .5, spec = .5, N = 10, round = TRUE)  # => ok, but all rounded
comp_freq_prob(prev = .5, sens = .5, spec = .5, N = 10, round = FALSE) # => ok, but not rounded

# Ways to fail:
comp_freq_prob(prev = NA, sens = 1, spec = 1, 100)  # => NAs + no warning (prev NA)
comp_freq_prob(prev = 1, sens = NA, spec = 1, 100)  # => NAs + no warning (sens NA)
comp_freq_prob(prev = 1, sens = 1, spec = NA, 100)  # => NAs + no warning (spec NA)
comp_freq_prob(prev = 8, sens = 1, spec = 1,  100)  # => NAs + warning (prev beyond range)
comp_freq_prob(prev = 1, sens = 8, spec = 1,  100)  # => NAs + warning (sens & spec beyond range)
```

comp_min_N *Compute a suitable minimum population size value N.*

Description

comp_min_N is a function that computes a population size value N (an integer as a power of 10) so that the frequencies of the 4 combinations of conditions and decisions (i.e., the cells of the confusion table, or bottom row of boxes in the natural frequency tree) reach or exceed a minimum value min.freq given the basic parameters prev, sens, and spec (spec = 1 - fart).

Usage

```
comp_min_N(prev, sens, spec, min.freq = 1)
```

Arguments

prev	The condition's prevalence value prev (i.e., the probability of condition being TRUE).
sens	The decision's sensitivity value sens (i.e., the conditional probability of a positive decision provided that the condition is TRUE).
spec	The specificity value spec (i.e., the conditional probability of a negative decision provided that the condition is FALSE).
min.freq	The minimum frequency of each combination of a condition and a decision (i.e., hits, misses, false alarms, and correct rejections). Default: min.freq = 1.

Details

Using this function helps avoiding excessively small decimal values in categories (esp. true positives, false negatives, false positives, and true negatives) when expressing combinations of conditions and decisions as natural frequencies. As values of zero (0) are tolerable, the function only increases N (in powers of 10) while the current value of any frequency (cell in confusion table or leaf of tree) is positive but below min.freq.

Note that comp_freq still needs to round to avoid decimal values in frequencies freq.

Value

An integer value N (as a power of 10).

See Also

population size N; num contains basic numeric parameters; freq contains current frequency information; comp_freq computes frequencies from probabilities; prob contains current probability information; comp_prob computes probabilities from probabilities; comp_freq_freq computes current frequency information from (4 essential) frequencies; comp_freq_prob computes current frequency information from (3 essential) probabilities; comp_prob_freq computes current probability information from (4 essential) frequencies; comp_prob_prob computes current probability information from (3 essential) probabilities.

Other functions computing frequencies: comp_freq_freq, comp_freq_prob, comp_freq, comp_popu, comp_prob_prob

Examples

```
comp_min_N(0, 0, 0)  # => 1
comp_min_N(1, 1, 1)  # => 1

comp_min_N(1, 1, 1, min.freq = 10)  # =>  10
comp_min_N(1, 1, 1, min.freq = 99)  # => 100

comp_min_N(.1, .1, .1)        # =>       100 = 10^2
comp_min_N(.001, .1, .1)      # =>    10 000 = 10^4
comp_min_N(.001, .001, .1)    # => 1 000 000 = 10^6
comp_min_N(.001, .001, .001)  # => 1 000 000 = 10^6
```

---

comp_mirt *Compute a decision's miss rate from its sensitivity.*

Description

comp_mirt is a conversion function that takes a sensitivity sens – given as a probability (i.e., a numeric value in the range from 0 to 1) – as its input, and returns the corresponding miss rate mirt – also as a probability – as its output.

Usage

```
comp_mirt(sens)
```

Arguments

sens	The decision's sensitivity sens as a probability.

Details

The miss rate mirt and sensitivity sens are complements (mirt = (1 - sens)) and both features of the decision process (e.g., a diagnostic test).

The function comp_mirt is complementary to the conversion function comp_sens and uses the generic function comp_complement.

Value

The decision's miss rate mirt as a probability.

See Also

comp_complement computes a probability's complement; is_complement verifies probability complements; comp_prob computes current probability information; prob contains current probability information; is_prob verifies probabilities.

Other functions computing probabilities: comp_FDR, comp_FOR, comp_NPV, comp_PPV, comp_accu, comp_acc, comp_comp_pair, comp_complement, comp_complete_prob_set, comp_fart, comp_ppod, comp_prob_freq, comp_prob, comp_sens, comp_spec

Examples

```
comp_mirt(2)                      # => NA + warning (beyond range)
comp_mirt(1/3)                    # => 0.6666667
comp_mirt(comp_complement(0.123))  # => 0.123
```

---

comp_NPV *Compute a decision's negative predictive value (NPV) from probabilities.*

Description

comp_NPV computes the negative predictive value NPV from 3 essential probabilities prev, sens, and spec.

Usage

```
comp_NPV(prev, sens, spec)
```

Arguments

prev	The condition's prevalence prev (i.e., the probability of condition being TRUE).
sens	The decision's sensitivity sens (i.e., the conditional probability of a positive decision provided that the condition is TRUE).
spec	The decision's specificity value spec (i.e., the conditional probability of a negative decision provided that the condition is FALSE).

Details

comp_NPV uses probabilities (not frequencies) and does not round results.

Value

The negative predictive value NPV as a probability. A warning is provided for NaN values.

See Also

comp_spec and comp_PPV compute related probabilities; is_extreme_prob_set verifies extreme cases; comp_complement computes a probability's complement; is_complement verifies probability complements; comp_prob computes current probability information; prob contains current probability information; is_prob verifies probabilities.

Other functions computing probabilities: comp_FDR, comp_FOR, comp_PPV, comp_accu, comp_acc, comp_comp_pair, comp_complement, comp_complete_prob_set, comp_fart, comp_mirt, comp_ppod, comp_prob_freq, comp_prob, comp_sens, comp_spec

Examples

```
# (1) Ways to work:
comp_NPV(.50, .500, .500)  # => NPV = 0.5
comp_NPV(.50, .333, .666)  # => NPV = 0.4996

# (2) Watch out for vectors:
prev <- seq(0, 1, .1)
comp_NPV(prev, .5, .5)  # => without NaN values
comp_NPV(prev,  1,  0)  # => with NaN values

# (3) Watch out for extreme values:
comp_NPV(1, 1, 1)   # => NaN, as cr = 0 and mi = 0: 0/0
comp_NPV(1, 1, 0)   # => NaN, as cr = 0 and mi = 0: 0/0
comp_NPV(.5, sens = 1, spec = 0)  # => NaN, no dec.neg cases:  NPV = 0/0 = NaN
is_extreme_prob_set(.5, sens = 1, spec = 0)  # => verifies extreme cases
```

---

comp_popu *Compute a population table from frequencies.*

Description

comp_popu is a function that computes a table popu (as an R data frame) from the current frequency information (contained in freq).

Usage

```
comp_popu(hi = freq$hi, mi = freq$mi, fa = freq$fa, cr = freq$cr,
  cond.true.lbl = txt$cond.true.lbl, cond.false.lbl = txt$cond.false.lbl,
  dec.pos.lbl = txt$dec.pos.lbl, dec.neg.lbl = txt$dec.neg.lbl,
  hi.lbl = txt$hi.lbl, mi.lbl = txt$mi.lbl, fa.lbl = txt$fa.lbl,
  cr.lbl = txt$cr.lbl)
```

Arguments

hi	The number of hits hi (or true positives).
mi	The number of misses mi (or false negatives).
fa	The number of false alarms fa (or false positives).
cr	The number of correct rejections cr (or true negatives).
cond.true.lbl	Text label for cond.true cases.
cond.false.lbl	Text label for cond.false cases.
dec.pos.lbl	Text label for dec.pos cases.
dec.neg.lbl	Text label for dec.neg cases.
hi.lbl	Text label for hi cases.
mi.lbl	Text label for mi cases.
fa.lbl	Text label for fa cases.
cr.lbl	Text label for cr cases.

Format

An object of class data.frame with N rows and 3 columns ("Truth", "Decision", "SDT").

Details

comp_popu also uses the current text settings contained in txt.

A visualization of the current population contained in popu is provided by plot_icon.

Value

A data frame popu containing N rows (individual cases) and 3 columns ("Truth", "Decision", "SDT") encoded as ordered factors (with 2, 2, and 4 levels, respectively).

See Also

The corresponding data frame popu; num for basic numeric parameters; freq for current frequency information; txt for current text settings; pal for current color settings.

Other functions computing frequencies: comp_freq_freq, comp_freq_prob, comp_freq, comp_min_N, comp_prob_prob

Examples

```
comp_popu(hi = 4, mi = 1, fa = 2, cr = 3)  # => computes a table of N = 10 cases

popu <- comp_popu()  # => initializes popu (with current values of freq and txt)
dim(popu)            # => N x 3
head(popu)           # => shows head of data frame
```

comp_ppod *Compute the proportion of positive decisions (ppod) from probabilities.*

Description

comp_ppod computes the proportion of positive decisions ppod from 3 essential probabilities prev, sens, and spec.

Usage

```
comp_ppod(prev, sens, spec)
```

Arguments

prev	The condition's prevalence prev (i.e., the probability of condition being TRUE).
sens	The decision's sensitivity sens (i.e., the conditional probability of a positive decision provided that the condition is TRUE).
spec	The decision's specificity value spec (i.e., the conditional probability of a negative decision provided that the condition is FALSE).

Details

comp_ppod uses probabilities (not frequencies) as inputs and returns a proportion (probability) without rounding.

Definition: ppod is proportion (or probability) of positive decisions:

`ppod = dec.pos/N = (hi + fa)/(hi + mi + fa + cr)`

Values range from 0 (only negative decisions) to 1 (only positive decisions).

Importantly, positive decisions dec.pos are not necessariliy correct decisions dec.cor.

Value

The proportion of positive decisions ppod as a probability. A warning is provided for NaN values.

See Also

comp_sens and comp_NPV compute related probabilities; is_extreme_prob_set verifies extreme cases; comp_complement computes a probability's complement; is_complement verifies probability complements; comp_prob computes current probability information; prob contains current probability information; is_prob verifies probabilities.

Other functions computing probabilities: comp_FDR, comp_FOR, comp_NPV, comp_PPV, comp_accu, comp_acc, comp_comp_pair, comp_complement, comp_complete_prob_set, comp_fart, comp_mirt, comp_prob_freq, comp_prob, comp_sens, comp_spec

Examples

```
# (1) ways to work:
comp_ppod(.10, .200, .300)  # => ppod = 0.65
comp_ppod(.50, .333, .666)  # => ppod = 0.3335

# (2) watch out for vectors:
prev <- seq(0, 1, .1)
comp_ppod(prev, .8, .5)  # => 0.50 0.53 0.56 0.59 0.62 0.65 0.68 0.71 0.74 0.77 0.80
comp_ppod(prev,  0,  1)  # => 0 0 0 0 0 0 0 0 0 0 0

# (3) watch out for extreme values:
comp_ppod(1, 1, 1)  #  => 1
comp_ppod(1, 1, 0)  #  => 1

comp_ppod(1, 0, 1)  #  => 0
comp_ppod(1, 0, 0)  #  => 0

comp_ppod(0, 1, 1)  #  => 0
comp_ppod(0, 1, 0)  #  => 1

comp_ppod(0, 0, 1)  #  => 0
comp_ppod(0, 0, 0)  #  => 1
```

`comp_PPV` *Compute a decision's positive predictive value (PPV) from probabilities.*

Description

`comp_PPV` computes the positive predictive value `PPV` from 3 essential probabilities `prev`, `sens`, and `spec`.

Usage

```
comp_PPV(prev, sens, spec)
```

Arguments

`prev`	The condition's prevalence `prev` (i.e., the probability of condition being `TRUE`).
`sens`	The decision's sensitivity `sens` (i.e., the conditional probability of a positive decision provided that the condition is `TRUE`).
`spec`	The decision's specificity value `spec` (i.e., the conditional probability of a negative decision provided that the condition is `FALSE`).

Details

`comp_PPV` uses probabilities (not frequencies) and does not round results.

Value

The positive predictive value PPV as a probability. A warning is provided for NaN values.

See Also

comp_sens and comp_NPV compute related probabilities; is_extreme_prob_set verifies extreme cases; comp_complement computes a probability's complement; is_complement verifies probability complements; comp_prob computes current probability information; prob contains current probability information; is_prob verifies probabilities.

Other functions computing probabilities: comp_FDR, comp_FOR, comp_NPV, comp_accu, comp_acc, comp_comp_pair, comp_complement, comp_complete_prob_set, comp_fart, comp_mirt, comp_ppod, comp_prob_freq, comp_prob, comp_sens, comp_spec

Examples

```
# (1) Ways to work:
comp_PPV(.50, .500, .500)  # => PPV = 0.5
comp_PPV(.50, .333, .666)  # => PPV = 0.499

# (2) Watch out for vectors:
prev <- seq(0, 1, .1)
comp_PPV(prev, .5, .5)  # => without NaN values
comp_PPV(prev,  0,  1)  # => with NaN values

# (3) Watch out for extreme values:
comp_PPV(prev = 1, sens = 0, spec = .5)  # => NaN, only mi: hi = 0 and fa = 0: PPV = 0/0 = NaN
is_extreme_prob_set(prev = 1, sens = 0, spec = .5)  # => verifies extreme cases

comp_PPV(prev = 0, sens = .5, spec = 1)  # => NaN, only cr: hi = 0 and fa = 0: PPV = 0/0 = NaN
is_extreme_prob_set(prev = 0, sens = .5, spec = 1)  # => verifies extreme cases

comp_PPV(prev = .5, sens = 0, spec = 1)  # => NaN, only cr: hi = 0 and fa = 0: PPV = 0/0 = NaN
is_extreme_prob_set(prev = .5, sens = 0, spec = 1)  # => verifies extreme cases
```

---

comp_prev *Compute the condition's prevalence (baseline probability) from frequencies.*

Description

comp_prev computes a condition's prevalence value prev (or baseline probability) from 4 essential frequencies (hi, mi, fa, cr).

Usage

```
comp_prev(hi = freq$hi, mi = freq$mi, fa = freq$fa, cr = freq$cr)
```

Arguments

hi	The number of hits hi (or true positives).
mi	The number of misses mi (or false negatives).
fa	The number of false alarms fa (or false positives).
cr	The number of correct rejections cr (or true negatives).

Details

A condition's prevalence value prev is the probability of the condition being TRUE.

The probability prev can be computed from frequencies as the the ratio of cond.true (i.e., hi + mi) divided by N (i.e., hi + mi + fa + cr):

```
prev = cond.true/N = (hi + mi)/(hi + mi + fa + cr)
```

See Also

num contains basic numeric parameters; init_num initializes basic numeric parameters; prob contains current probability information; comp_prob computes current probability information; freq contains current frequency information; comp_freq computes current frequency information; is_prob verifies probability inputs; is_freq verifies frequency inputs.

---

comp_prob *Compute probabilities from (3 essential) probabilities.*

Description

comp_prob computes current probability information from 3 essential probabilities (prev, sens or mirt, spec or fart). It returns a list of 10 probabilities prob as its output.

Usage

```
comp_prob(prev = num$prev, sens = num$sens, mirt = NA, spec = num$spec,
  fart = NA, tol = 0.01)
```

Arguments

prev	The condition's prevalence value prev (i.e., the probability of the condition being TRUE).
sens	The decision's sensitivity value sens (i.e., the conditional probability of a positive decision provided that the condition is TRUE). sens is optional when its complement mirt is provided.
mirt	The decision's miss rate value mirt (i.e., the conditional probability of a negative decision provided that the condition is TRUE). mirt is optional when its complement sens is provided.

spec	The decision's specificity value spec (i.e., the conditional probability of a negative decision provided that the condition is FALSE). spec is optional when its complement fart is provided.
fart	The decision's false alarm rate fart (i.e., the conditional probability of a positive decision provided that the condition is FALSE). fart is optional when its complement spec is provided.
tol	A numeric tolerance value for is_complement. Default: tol = .01.

Details

comp_prob assumes that a sufficient and consistent set of essential probabilities (i.e., prev and either sens or its complement mirt, and either spec or its complement fart) is provided.

comp_prob computes and returns a full set of basic and various derived probabilities (e.g., the probability of a positive decision ppod, the predictive values PPV and NPV, as well as their complements FDR and FOR) in its output of a list prob.

Extreme probabilities (sets containing two or more probabilities of 0 or 1) may yield unexpected values (e.g., predictive values PPV or NPV turning NaN when is_extreme_prob_set evaluates to TRUE).

comp_prob is the probability counterpart to the frequency function comp_freq.

Key relationships:

- Other functions translating between representational formats:
  1. comp_prob_prob (defined here) is a wrapper function for comp_prob and an analog to 3 other format conversion functions:
  2. comp_prob_freq computes current *probability* information contained in prob from 4 essential frequencies (hi, mi, fa, cr).
  3. comp_freq_prob computes current *frequency* information contained in freq from 3 essential probabilities (prev, sens, spec).
  4. comp_freq_freq computes current *frequency* information contained in freq from 4 essential frequencies (hi, mi, fa, cr).
- Two perspectives:
  A population of N individuals can be split into 2 subsets in 2 different ways:
  1. by condition:
     The frequency cond.true depends on the prevalence prev and the frequency cond.false depends on the prevalence's complement 1 - prev.
  2. by decision:
     The frequency dec.pos depends on the proportion of positive decisions ppod and the frequency dec.neg depends on the proportion of negative decisions 1 - ppod.

  The population size N is a free parameter (independent of the essential probabilities prev, sens, and spec).

  If N is unknown (NA), a suitable minimum value can be computed by comp_min_N.
- Combinations of frequencies:
  In a population of size N the following relationships hold:
  1. N = cond.true + cond.false = (hi + mi) + (fa + cr) (by condition)

2. N = dec.pos + dec.neg = (hi + fa) + (mi + cr) (by decision)
3. N = hi + mi + fa + cr (by condition x decision)

The two perspectives (by condition vs. by decision) combine the 4 essential frequencies (i.e., hi, mi, fa, cr) in 2 different ways.

- Defining probabilities in terms of frequencies:
  Probabilities *are* numbers from 0 to 1 and determine, describe, or are defined as relationships between frequencies. The following probabilities can be computed as ratios between frequencies:
  1. prevalence prev:
     prev = cond.true/N = (hi + mi) / (hi + mi + fa + cr)
  2. sensitivity sens:
     sens = hi/cond.true = hi / (hi + mi) = (1 - mirt)
  3. miss rate mirt:
     mirt = mi/cond.true = mi / (hi + mi) = (1 - sens)
  4. specificity spec:
     spec = cr/cond.false = cr / (fa + cr) = (1 - fart)
  5. false alarm rate fart:
     fart = fa/cond.false = fa / (fa + cr) = (1 - spec)
  6. proportion of positive decisions ppod:
     ppod = dec.pos/N = (hi + fa) / (hi + mi + fa + cr)
  7. positive predictive value PPV:
     PPV = hi/dec.pos = hi / (hi + fa) = (1 - FDR)
  8. false detection rate FDR:
     FDR = fa/dec.pos = fa / (hi + fa) = (1 - PPV)
  9. negative predictive value NPV:
     NPV = cr/dec.neg = cr / (mi + cr) = (1 - FOR)
  10. false omission rate FOR:
      FOR = mi/dec.neg = mi / (mi + cr) = (1 - NPV)

Value

A list prob containing 10 probability values.

See Also

prob contains current probability information; num contains basic numeric parameters; init_num initializes basic numeric parameters; pal contains current color information; txt contains current text information; freq contains current frequency information; comp_freq computes frequencies from probabilities; is_valid_prob_set verifies sets of probability inputs; is_extreme_prob_set verifies sets of extreme probabilities; comp_min_N computes a suitable minimum population size N; comp_freq_freq computes current frequency information from (4 essential) frequencies; comp_freq_prob computes current frequency information from (3 essential) probabilities; comp_prob_freq computes current probability information from (4 essential) frequencies; comp_prob_prob computes current probability information from (3 essential) probabilities.

Other functions computing probabilities: comp_FDR, comp_FOR, comp_NPV, comp_PPV, comp_accu, comp_acc, comp_comp_pair, comp_complement, comp_complete_prob_set, comp_fart, comp_mirt, comp_ppod, comp_prob_freq, comp_sens, comp_spec

Examples

```
# Basics:
comp_prob(prev = .11, sens = .88, spec = .77)                    # => ok: PPV = 0.3210614
comp_prob(prev = .11, sens = NA, mirt = .12, spec = NA, fart = .23)  # => ok: PPV = 0.3210614
comp_prob()          # => ok, using current defaults
length(comp_prob())  # => 10 probabilities

# Ways to work:
comp_prob(.99, sens = .99, spec = .99)              # => ok: PPV = 0.999898
comp_prob(.99, sens = .90, spec = NA, fart = .10)   # => ok: PPV = 0.9988789

# Watch out for extreme cases:
comp_prob(1, sens = 0, spec = 1)      # => ok, but with warnings (as PPV & FDR are NaN)
comp_prob(1, sens = 0, spec = 0)      # => ok, but with warnings (as PPV & FDR are NaN)
comp_prob(1, sens = 0, spec = NA, fart = 0)  # => ok, but with warnings (as PPV & FDR are NaN)
comp_prob(1, sens = 0, spec = NA, fart = 1)  # => ok, but with warnings (as PPV & FDR are NaN)

# Watch out for extreme cases:
comp_prob(1, sens = 0, spec = 1)      # => ok, but with warnings (as PPV & FDR are NaN)
comp_prob(1, sens = 0, spec = 0)      # => ok, but with warnings (as PPV & FDR are NaN)
comp_prob(1, sens = 0, spec = NA, fart = 0)  # => ok, but with warnings (as PPV & FDR are NaN)
comp_prob(1, sens = 0, spec = NA, fart = 1)  # => ok, but with warnings (as PPV & FDR are NaN)

comp_prob(1, sens = 1, spec = 0)      # => ok, but with warnings (as NPV & FOR are NaN)
comp_prob(1, sens = 1, spec = 1)      # => ok, but with warnings (as NPV & FOR are NaN)
comp_prob(1, sens = 1, spec = NA, fart = 0)  # => ok, but with warnings (as NPV & FOR are NaN)
comp_prob(1, sens = 1, spec = NA, fart = 1)  # => ok, but with warnings (as NPV & FOR are NaN)

# Ways to fail:
comp_prob(NA, 1, 1, NA)  # => only warning: invalid set (prev not numeric)
comp_prob(8,  1, 1, NA)  # => only warning: prev no probability
comp_prob(1,  8, 1, NA)  # => only warning: sens no probability
comp_prob(1,  1, 1,  1)  # => only warning: is_complement not in tolerated range
```

`comp_prob_freq` *Compute probabilities from (4 essential) frequencies.*

Description

`comp_prob_freq` computes current probability information from 4 essential frequencies (`hi`, `mi`, `fa`, `cr`). It returns a list of 10 probabilities `prob` as its output.

Usage

```
comp_prob_freq(hi = freq$hi, mi = freq$mi, fa = freq$fa, cr = freq$cr)
```

Arguments

hi	The number of hits hi (or true positives).
mi	The number of misses mi (or false negatives).
fa	The number of false alarms fa (or false positives).
cr	The number of correct rejections cr (or true negatives).

Details

Key relationships:

- Other functions translating between representational formats:
  1. comp_prob_freq (defined here) is an analog to 3 other format conversion functions:
  2. comp_freq_freq computes current *frequency* information contained in freq from 4 essential frequencies (hi, mi, fa, cr).
  3. comp_freq_prob computes current *frequency* information contained in freq from 3 essential probabilities (prev, sens, spec).
  4. comp_prob_prob computes current *probability* information contained in prob from 3 essential probabilities (prev, sens, spec).
- Two perspectives:

  A population of N individuals can be split into 2 subsets in 2 different ways:

  1. by condition:
     The frequency cond.true depends on the prevalence prev and the frequency cond.false depends on the prevalence's complement 1 - prev.
  2. by decision:
     The frequency dec.pos depends on the proportion of positive decisions ppod and the frequency dec.neg depends on the proportion of negative decisions 1 - ppod.

  The population size N is a free parameter (independent of the essential probabilities prev, sens, and spec).

  If N is unknown (NA), a suitable minimum value can be computed by comp_min_N.
- Combinations of frequencies:

  In a population of size N the following relationships hold:

  1. N = cond.true + cond.false = (hi + mi) + (fa + cr) (by condition)
  2. N = dec.pos + dec.neg = (hi + fa) + (mi + cr) (by decision)
  3. N = hi + mi + fa + cr (by condition x decision)

  The two perspectives (by condition vs. by decision) combine the 4 essential frequencies (i.e., hi, mi, fa, cr) in 2 different ways.
- Defining probabilities in terms of frequencies:

  Probabilities *are* – determine, describe, or are defined as – the relationships between frequencies. Thus, they can be computed as ratios between frequencies.

  The following relationships hold (and are used in computations):

  1. prevalence prev:
     prev = cond.true/N = (hi + mi) / (hi + mi + fa + cr)

2. sensitivity sens:
   sens = hi/cond.true = hi / (hi + mi) = (1 - mirt)
3. miss rate mirt:
   mirt = mi/cond.true = mi / (hi + mi) = (1 - sens)
4. specificity spec:
   spec = cr/cond.false = cr / (fa + cr) = (1 - fart)
5. false alarm rate fart:
   fart = fa/cond.false = fa / (fa + cr) = (1 - spec)
6. proportion of positive decisions ppod:
   ppod = dec.pos/N = (hi + fa) / (hi + mi + fa + cr)
7. positive predictive value PPV:
   PPV = hi/dec.pos = hi / (hi + fa) = (1 - FDR)
8. negative predictive value NPV:
   NPV = cr/dec.neg = cr / (mi + cr) = (1 - FOR)
9. false detection rate FDR:
   FDR = fa/dec.pos = fa / (hi + fa) = (1 - PPV)
10. false omission rate FOR:
    FOR = mi/dec.neg = mi / (mi + cr) = (1 - NPV)

See Also

comp_freq_freq computes current frequency information from (4 essential) frequencies; comp_freq_prob computes current frequency information from (3 essential) probabilities; comp_prob_prob computes current probability information from (3 essential) probabilities; num contains basic numeric parameters; init_num initializes basic numeric parameters; prob contains current probability information; comp_prob computes current probability information; freq contains current frequency information; comp_freq computes current frequency information; is_prob verifies probability inputs; is_freq verifies frequency inputs.

Other functions computing probabilities: comp_FDR, comp_FOR, comp_NPV, comp_PPV, comp_accu, comp_acc, comp_comp_pair, comp_complement, comp_complete_prob_set, comp_fart, comp_mirt, comp_ppod, comp_prob, comp_sens, comp_spec

Other format conversion functions: comp_freq_freq, comp_freq_prob, comp_prob_prob

Examples

```
## Basics:
comp_prob_freq()
all.equal(prob, comp_prob_freq())  # => should be TRUE!


## Circular chain:
# 1. Current numeric parameters:
num

# 2. Compute all 10 probabilities in prob (from essential probabilities):
prob <- comp_prob()

# 3. Compute 9 frequencies in freq from probabilities:
```

```
freq <- comp_freq(round = FALSE)   # prevent rounding (to obtain same probabilities later)
freq

# 4. Compute all 10 probabilities again (but now from frequencies):
prob_freq <- comp_prob_freq()
prob_freq

# 5. Check equality of results (steps 2. and 4.):
all.equal(prob, prob_freq)  # => should be TRUE
```

---

comp_prob_prob *Compute probabilities from (3 essential) probabilities.*

Description

comp_prob_prob computes current probability information from 3 essential probabilities (prev, sens or mirt, spec or fart). It returns a list of 10 probabilities prob as its output.

Usage

```
comp_prob_prob(prev = prob$prev, sens = prob$sens, mirt = NA,
  spec = prob$spec, fart = NA, tol = 0.01)
```

Arguments

prev	The condition's prevalence value prev (i.e., the probability of condition being TRUE).
sens	The decision's sensitivity value sens (i.e., the conditional probability of a positive decision provided that the condition is TRUE). sens is optional when its complement mirt is provided.
mirt	The decision's miss rate value mirt (i.e., the conditional probability of a negative decision provided that the condition is TRUE). mirt is optional when its complement sens is provided.
spec	The decision's specificity value spec (i.e., the conditional probability of a negative decision provided that the condition is FALSE). spec is optional when its complement fart is provided.
fart	The decision's false alarm rate fart (i.e., the conditional probability of a positive decision provided that the condition is FALSE). fart is optional when its complement spec is provided.
tol	A numeric tolerance value for is_complement. Default: tol = .01.

Details

comp_prob assumes that a sufficient and consistent set of essential probabilities (i.e., prev and either sens or its complement mirt, and either spec or its complement fart) is provided.

comp_prob computes and returns a full set of basic and various derived probabilities (e.g., the probability of a positive decision ppod, the predictive values PPV and NPV, as well as their complements FDR and FOR) in its output of a list prob.

Extreme probabilities (sets containing two or more probabilities of 0 or 1) may yield unexpected values (e.g., predictive values PPV or NPV turning NaN when is_extreme_prob_set evaluates to TRUE).

Key relationships:

- Other functions translating between representational formats:
  1. comp_prob_prob (defined here) is a wrapper function for comp_prob and an analog to 3 other format conversion functions:
  2. comp_prob_freq computes current *probability* information contained in prob from 4 essential frequencies (hi, mi, fa, cr).
  3. comp_freq_prob computes current *frequency* information contained in freq from 3 essential probabilities (prev, sens, spec).
  4. comp_freq_freq computes current *frequency* information contained in freq from 4 essential frequencies (hi, mi, fa, cr).
- Two perspectives:
  A population of N individuals can be split into 2 subsets in 2 different ways:
  1. by condition:
     The frequency cond.true depends on the prevalence prev and the frequency cond.false depends on the prevalence's complement 1 - prev.
  2. by decision:
     The frequency dec.pos depends on the proportion of positive decisions ppod and the frequency dec.neg depends on the proportion of negative decisions 1 - ppod.

  The population size N is a free parameter (independent of the essential probabilities prev, sens, and spec).

  If N is unknown (NA), a suitable minimum value can be computed by comp_min_N.
- Combinations of frequencies:
  In a population of size N the following relationships hold:
  1. N = cond.true + cond.false = (hi + mi) + (fa + cr) (by condition)
  2. N = dec.pos + dec.neg = (hi + fa) + (mi + cr) (by decision)
  3. N = hi + mi + fa + cr (by condition x decision)

  The two perspectives (by condition vs. by decision) combine the 4 essential frequencies (i.e., hi, mi, fa, cr) in 2 different ways.
- Defining probabilities in terms of frequencies:
  Probabilities *are* – determine, describe, or are defined as – the relationships between frequencies. Thus, they can be computed as ratios between frequencies:
  1. prevalence prev:
     prev = cond.true/N = (hi + mi) / (hi + mi + fa + cr)

2. sensitivity sens:
   sens = hi/cond.true = hi / (hi + mi) = (1 - mirt)
3. miss rate mirt:
   mirt = mi/cond.true = mi / (hi + mi) = (1 - sens)
4. specificity spec:
   spec = cr/cond.false = cr / (fa + cr) = (1 - fart)
5. false alarm rate fart:
   fart = fa/cond.false = fa / (fa + cr) = (1 - spec)
6. proportion of positive decisions ppod:
   ppod = dec.pos/N = (hi + fa) / (hi + mi + fa + cr)
7. positive predictive value PPV:
   PPV = hi/dec.pos = hi / (hi + fa) = (1 - FDR)
8. negative predictive value NPV:
   NPV = cr/dec.neg = cr / (mi + cr) = (1 - FOR)
9. false detection rate FDR:
   FDR = fa/dec.pos = fa / (hi + fa) = (1 - PPV)
10. false omission rate FOR:
    FOR = mi/dec.neg = mi / (mi + cr) = (1 - NPV)

Value

A list prob containing 10 probability values.

See Also

comp_freq_prob computes current frequency information from (3 essential) probabilities; comp_freq_freq computes current frequency information from (4 essential) frequencies; comp_prob_freq computes current probability information from (4 essential) frequencies; num contains basic numeric variables; init_num initializes basic numeric variables; freq contains current frequency information; comp_freq computes current frequency information; prob contains current probability information; comp_prob computes current probability information; comp_complement computes a probability's complement; comp_comp_pair computes pairs of complements; comp_complete_prob_set completes valid sets of probabilities; comp_min_N computes a suitable population size N (if missing).

Other functions computing frequencies: comp_freq_freq, comp_freq_prob, comp_freq, comp_min_N, comp_popu

Other format conversion functions: comp_freq_freq, comp_freq_prob, comp_prob_freq

Examples

```
# Basics:
comp_prob_prob(prev = .11, sens = .88, spec = .77)                    # => ok: PPV = 0.3210614
comp_prob_prob(prev = .11, sens = NA, mirt = .12, spec = NA, fart = .23)  # => ok: PPV = 0.3210614
comp_prob_prob()          # => ok, using current defaults
length(comp_prob_prob())  # => 10 probabilities

# Ways to work:
comp_prob_prob(.99, sens = .99, spec = .99)              # => ok: PPV = 0.999898
comp_prob_prob(.99, sens = .90, spec = NA, fart = .10)   # => ok: PPV = 0.9988789
```

```
# Watch out for extreme cases:
comp_prob_prob(1, sens = 0, spec = 1)     # => ok, but with warnings (as PPV & FDR are NaN)
comp_prob_prob(1, sens = 0, spec = 0)     # => ok, but with warnings (as PPV & FDR are NaN)
comp_prob_prob(1, sens = 0, spec = NA, fart = 0)  # => ok, but with warnings (as PPV & FDR are NaN)
comp_prob_prob(1, sens = 0, spec = NA, fart = 1)  # => ok, but with warnings (as PPV & FDR are NaN)

comp_prob_prob(1, sens = 1, spec = 0)     # => ok, but with warnings (as NPV & FOR are NaN)
comp_prob_prob(1, sens = 1, spec = 1)     # => ok, but with warnings (as NPV & FOR are NaN)
comp_prob_prob(1, sens = 1, spec = NA, fart = 0)  # => ok, but with warnings (as NPV & FOR are NaN)
comp_prob_prob(1, sens = 1, spec = NA, fart = 1)  # => ok, but with warnings (as NPV & FOR are NaN)

# Ways to fail:
comp_prob_prob(NA, 1, 1, NA)  # => only warning: invalid set (prev not numeric)
comp_prob_prob(8,  1, 1, NA)  # => only warning: prev no probability
comp_prob_prob(1,  8, 1, NA)  # => only warning: sens no probability
comp_prob_prob(1,  1, 1,  1)  # => only warning: is_complement not in tolerated range
```

`comp_sens` *Compute a decision's sensitivity from its miss rate.*

Description

`comp_sens` is a conversion function that takes a miss rate `mirt` – given as a probability (i.e., a numeric value in the range from 0 to 1) – as its input, and returns the corresponding sensitivity `sens` – also as a probability – as its output.

Usage

```
comp_sens(mirt)
```

Arguments

`mirt`	The decision's miss rate `mirt` as a probability.

Details

The sensitivity `sens` and miss rate `mirt` are complements (`sens = (1 - mirt)`) and both features of the decision process (e.g., a diagnostic test).

The function `comp_sens` is complementary to the conversion function `comp_mirt` and uses the generic function `comp_complement`.

Value

The decision's sensitivity `sens` as a probability.

See Also

comp_complement computes a probability's complement; is_complement verifies probability complements; comp_prob computes current probability information; prob contains current probability information; is_prob verifies probabilities.

Other functions computing probabilities: comp_FDR, comp_FOR, comp_NPV, comp_PPV, comp_accu, comp_acc, comp_comp_pair, comp_complement, comp_complete_prob_set, comp_fart, comp_mirt, comp_ppod, comp_prob_freq, comp_prob, comp_spec

Examples

```
comp_sens(2)                      # => NA + warning (beyond range)
comp_sens(1/3)                    # => 0.6666667
comp_sens(comp_complement(0.123))  # => 0.123
```

---

comp_spec *Compute a decision's specificity from its false alarm rate.*

Description

comp_spec is a conversion function that takes a false alarm rate fart – given as a probability (i.e., a numeric value in the range from 0 to 1) – as its input, and returns the corresponding specificity spec – also as a probability – as its output.

Usage

```
comp_spec(fart)
```

Arguments

fart	The decision's false alarm rate fart as a probability.

Details

The specificity spec and the false alarm rate fart are complements (spec = (1 - fart)) and both features of the decision process (e.g., a diagnostic test).

The function comp_spec is complementary to the conversion function comp_fart and uses the generic function comp_complement.

Value

The decision's specificity spec as a probability.

See Also

comp_complement computes a probability's complement; is_complement verifies probability complements; comp_prob computes current probability information; prob contains current probability information; is_prob verifies probabilities.

Other functions computing probabilities: comp_FDR, comp_FOR, comp_NPV, comp_PPV, comp_accu, comp_acc, comp_comp_pair, comp_complement, comp_complete_prob_set, comp_fart, comp_mirt, comp_ppod, comp_prob_freq, comp_prob, comp_sens

Examples

```
comp_spec(2)                      # => NA + warning (beyond range)
comp_spec(1/3)                    # => 0.6666667
comp_spec(comp_complement(0.123))  # => 0.123
```

---

cond.false *Number of individuals for which the condition is false.*

Description

cond.false is a frequency that describes the number of individuals in the current population N for which the condition is FALSE (i.e., actually false cases).

Usage

```
cond.false
```

Format

An object of class numeric of length 1.

Details

Key relationships:

1. to probabilities: The frequency of cond.false individuals depends on the population size N and the complement of the condition's prevalence 1 - prev and is split further into two subsets of fa by the false alarm rate fart and cr by the specificity spec.

   Perspectives:

   (a) by condition:
   The frequency cond.false is determined by the population size N times the complement of the prevalence (1 - prev):
   cond.false= N x (1 - prev)

(b) by decision:
a. The frequency fa is determined by cond.false times the false alarm rate fart = (1 - spec) (aka. FPR):
fa = cond.false x fart = cond.false x (1 - spec)
b. The frequency cr is determined by cond.false times the specificity spec = (1 - fart):
cr = cond.false x spec = cond.false x (1 - fart)

2. to other frequencies: In a population of size N the following relationships hold:
   - N = cond.true + cond.false (by condition)
   - N = dec.pos + dec.neg (by decision)
   - N = dec.cor + dec.err (by correspondence of decision to condition)
   - N = hi + mi + fa + cr (by condition x decision)

Current frequency information is computed by comp_freq and contained in a list freq.

References

Consult Wikipedia: Confusion matrix for additional information.

See Also

is_freq verifies frequencies; num contains basic numeric parameters; init_num initializes basic numeric parameters; freq contains current frequency information; comp_freq computes current frequency information; prob contains current probability information; comp_prob computes current probability information.

Other frequencies: N, cond.true, cr, dec.cor, dec.err, dec.neg, dec.pos, fa, hi, mi

Examples

```
cond.false <- 1000 * .90  # => sets cond.false to 90% of 1000 = 900 cases.
is_freq(cond.false)       # => TRUE
is_prob(cond.false)     # => FALSE, as cond.false is no probability [but (1 - prev) and spec are]
```

---

cond.true *Number of individuals for which the condition is true.*

Description

cond.true is a frequency that describes the number of individuals in the current population N for which the condition is TRUE (i.e., actually true cases).

Usage

```
cond.true
```

Format

An object of class numeric of length 1.

Details

Key relationships:

1. to probabilities: The frequency of `cond.true` individuals depends on the population size `N` and the condition's prevalence `prev` and is split further into two subsets of `hi` by the sensitivity `sens` and `mi` by the miss rate `mirt`.

   Perspectives:

   (a) by condition:
   The frequency `cond.true` is determined by the population size `N` times the prevalence `prev`:
   `cond.true = N x prev`

   (b) by decision:
   a. The frequency `hi` is determined by `cond.true` times the sensitivity `sens` (aka. hit rate `HR`):
   `hi = cond.true x sens`
   b. The frequency `mi` is determined by `cond.true` times the miss rate `mirt = (1 - sens)`:
   `mi = cond.true x mirt = cond.true x (1 - sens)`

2. to other frequencies: In a population of size `N` the following relationships hold:

   - `N = cond.true + cond.false` (by condition)
   - `N = dec.pos + dec.neg` (by decision)
   - `N = dec.cor + dec.err` (by correspondence of decision to condition)
   - `N = hi + mi + fa + cr` (by condition x decision)

Current frequency information is computed by `comp_freq` and contained in a list `freq`.

References

Consult Wikipedia: Confusion matrix for additional information.

See Also

`is_freq` verifies frequencies; `num` contains basic numeric parameters; `init_num` initializes basic numeric parameters; `freq` contains current frequency information; `comp_freq` computes current frequency information; `prob` contains current probability information; `comp_prob` computes current probability information.

Other frequencies: `N`, `cond.false`, `cr`, `dec.cor`, `dec.err`, `dec.neg`, `dec.pos`, `fa`, `hi`, `mi`

Examples

```
cond.true <- 1000 * .10   # => sets cond.true to 10% of 1000 = 100 cases.
is_freq(cond.true)        # => TRUE
is_prob(cond.true)      # => FALSE, as cond.true is no probability (but prev and sens are)
```

cr *Frequency of correct rejections or true negatives (TN).*

Description

cr is the frequency of correct rejections or true negatives (TN) in a population of N individuals.

Usage

```
cr
```

Format

An object of class numeric of length 1.

Details

Definition: cr is the frequency of individuals for which Condition = FALSE and Decision = FALSE (negative).

cr is a measure of correct classifications, not an individual case.

Relationships:

1. to probabilities: The frequency cr depends on the specificity spec (aka. true negative rate, TNR) and is conditional on the prevalence prev.
2. to other frequencies: In a population of size N the following relationships hold:
   - N = cond.true + cond.false (by condition)
   - N = dec.pos + dec.neg (by decision)
   - N = dec.cor + dec.err (by correspondence of decision to condition)
   - N = hi + mi + fa + cr (by condition x decision)

See Also

spec is the specificity or correct rejection rate (aka. true negative rate TNR); num contains basic numeric parameters; init_num initializes basic numeric parameters; freq contains current frequency information; comp_freq computes current frequency information; prob contains current probability information; comp_prob computes current probability information; is_freq verifies frequencies.

Other essential parameters: fa, hi, mi, prev, sens, spec

Other frequencies: N, cond.false, cond.true, dec.cor, dec.err, dec.neg, dec.pos, fa, hi, mi

dec.cor *Number of individuals for which the decision is correct.*

Description

dec.cor is a frequency that describes the number of individuals in the current population N for which the decision is correct (i.e., cases in which the decision corresponds to the condition).

Usage

```
dec.cor
```

Format

An object of class numeric of length 1.

Details

Key relationships:

1. to probabilities: The frequency of dec.cor individuals depends on the population size N and is equal to the sum of true positives hi and true negatives cr.
2. to other frequencies: In a population of size N the following relationships hold:
   - N = cond.true + cond.false (by condition)
   - N = dec.pos + dec.neg (by decision)
   - N = dec.cor + dec.err (by correspondence of decision to condition)
   - dec.cor = hi + cr
   - dec.err = mi + fa
   - N = hi + mi + fa + cr (by condition x decision)

Current frequency information is computed by comp_freq and contained in a list freq.

References

Consult Wikipedia: Confusion matrix for additional information.

See Also

is_freq verifies frequencies; num contains basic numeric parameters; init_num initializes basic numeric parameters; freq contains current frequency information; comp_freq computes current frequency information; prob contains current probability information; comp_prob computes current probability information.

Other frequencies: N, cond.false, cond.true, cr, dec.err, dec.neg, dec.pos, fa, hi, mi

Examples

```
dec.cor <- 1000 * .50   # => sets dec.cor to 50% of 1000 = 500 cases.
is_freq(dec.cor)        # => TRUE
is_prob(dec.cor)       # => FALSE, as dec.cor is no probability (but acc, bacc/wacc ARE)
```

---

dec.err *Number of individuals for which the decision is erroneous.*

Description

`dec.err` is a frequency that describes the number of individuals in the current population `N` for which the decision is incorrect or erroneous (i.e., cases in which the decision does not correspond to the condition).

Usage

```
dec.err
```

Format

An object of class `numeric` of length 1.

Details

Key relationships:

1. to probabilities: The frequency of `dec.err` individuals depends on the population size `N` and is equal to the sum of false negatives `mi` and false positives `fa`.
2. to other frequencies: In a population of size `N` the following relationships hold:
   - `N = cond.true + cond.false` (by condition)
   - `N = dec.pos + dec.neg` (by decision)
   - `N = dec.cor + dec.err` (by correspondence of decision to condition)
   - `dec.cor = hi + cr`
   - `dec.err = mi + fa`
   - `N = hi + mi + fa + cr` (by condition x decision)

Current frequency information is computed by `comp_freq` and contained in a list `freq`.

References

Consult Wikipedia: Confusion matrix for additional information.

See Also

is_freq verifies frequencies; num contains basic numeric parameters; init_num initializes basic numeric parameters; freq contains current frequency information; comp_freq computes current frequency information; prob contains current probability information; comp_prob computes current probability information.

Other frequencies: N, cond.false, cond.true, cr, dec.cor, dec.neg, dec.pos, fa, hi, mi

Examples

```
dec.err <- 1000 * .50   # => sets dec.err to 50% of 1000 = 500 cases.
is_freq(dec.err)         # => TRUE
is_prob(dec.err)        # => FALSE, as dec.err is no probability (but acc, bacc/wacc ARE)
```

---

dec.neg *Number of individuals for which the decision is negative.*

Description

dec.neg is a frequency that describes the number of individuals in the current population N for which the decision is negative (i.e., cases not called or not predicted).

Usage

```
dec.neg
```

Format

An object of class numeric of length 1.

Details

Key relationships:

1. to probabilities: The frequency of dec.neg individuals depends on the population size N and the decision's proportion of negative decisions (1 - ppod) and is split further into two subsets of cr by the negative predictive value NPV and mi by the false omission rate FOR = 1 - NPV.

   Perspectives:

   (a) by condition:
   The frequency dec.neg is determined by the population size N times the proportion of negative decisions (1 - ppod):
   dec.neg = N x (1 - ppod)

   (b) by decision:
   a. The frequency cr is determined by dec.neg times the negative predictive value NPV:
   cr = dec.neg x NPV
   b. The frequency mi is determined by dec.neg times the false omission rate FOR = (1 - NPV):
   mi = dec.neg x FOR = dec.neg x (1 - NPV)

2. to other frequencies: In a population of size N the following relationships hold:
   - N = cond.true + cond.false (by condition)
   - N = dec.pos + dec.neg (by decision)
   - N = dec.cor + dec.err (by correspondence of decision to condition)
   - N = hi + mi + fa + cr (by condition x decision)

Current frequency information is computed by comp_freq and contained in a list freq.

References

Consult Wikipedia: Confusion matrix for additional information.

See Also

is_freq verifies frequencies; num contains basic numeric parameters; init_num initializes basic numeric parameters; freq contains current frequency information; comp_freq computes current frequency information; prob contains current probability information; comp_prob computes current probability information.

Other frequencies: N, cond.false, cond.true, cr, dec.cor, dec.err, dec.pos, fa, hi, mi

Examples

```
dec.neg <- 1000 * .67   # => sets dec.neg to 67% of 1000 = 670 cases.
is_freq(dec.neg)         # => TRUE
is_prob(dec.neg)       # => FALSE, as dec.neg is no probability (but ppod, NPV and FOR are)
```

---

dec.pos *Number of individuals for which the decision is positive.*

Description

dec.pos is a frequency that describes the number of individuals in the current population N for which the decision is positive (i.e., called or predicted cases).

Usage

```
dec.pos
```

Format

An object of class numeric of length 1.

Details

Key relationships:

1. to probabilities: The frequency of `dec.pos` individuals depends on the population size `N` and the decision's proportion of positive decisions `ppod` and is split further into two subsets of `hi` by the positive predictive value `PPV` and `fa` by the false detection rate `FDR = 1 - PPV`.

   Perspectives:

   (a) by condition:
   The frequency `dec.pos` is determined by the population size `N` times the proportion of positive decisions `ppod`:
   `dec.pos = N x ppod`

   (b) by decision:
   a. The frequency `hi` is determined by `dec.pos` times the positive predictive value `PPV` (aka. `precision`):
   `hi = dec.pos x PPV`
   b. The frequency `fa` is determined by `dec.pos` times the false detection rate `FDR = (1 - PPV)`:
   `fa = dec.pos x FDR = dec.pos x (1 - PPV)`

2. to other frequencies: In a population of size `N` the following relationships hold:

   - `N = cond.true + cond.false` (by condition)
   - `N = dec.pos + dec.neg` (by decision)
   - `N = dec.cor + dec.err` (by correspondence of decision to condition)
   - `N = hi + mi + fa + cr` (by condition x decision)

Current frequency information is computed by `comp_freq` and contained in a list `freq`.

References

Consult Wikipedia: Confusion matrix for additional information.

See Also

`is_freq` verifies frequencies; `num` contains basic numeric parameters; `init_num` initializes basic numeric parameters; `freq` contains current frequency information; `comp_freq` computes current frequency information; `prob` contains current probability information; `comp_prob` computes current probability information.

Other frequencies: `N`, `cond.false`, `cond.true`, `cr`, `dec.cor`, `dec.err`, `dec.neg`, `fa`, `hi`, `mi`

Examples

```
dec.pos <- 1000 * .33   # => sets dec.pos to 33% of 1000 = 330 cases.
is_freq(dec.pos)        # => TRUE
is_prob(dec.pos)        # => FALSE, as dec.pos is no probability (but ppod and PPV are)
```

`df.scenarios` *A collection of riskyr scenarios from various sources.*

Description

`df.scenarios` is an R data frame that contains a collection of scenarios from the scientific literature and other sources.

Usage

```
df.scenarios
```

Format

A data frame with currently 26 rows (i.e., scenarios) and 21 columns (variables describing each scenario):

Scenarios:

1. Mammografie 1
2. Nackenfaltentest (NFT)
3. HIV 1 (f)
4. HIV 2 (f)
5. Mammography 2
6. Sepsis
7. Cab problem
8. Sigmoidoskopie 1
9. Sigmoidoskopie 1
10. Brustkrebs 1
11. Brustkrebs 2 (BRCA1)
12. Brustkrebs 3 (BRCA1+pos. Mam.)
13. HIV 3 (m)
14. HIV 4 (m)
15. Nackenfaltentest 2 (NFT)
16. Amniozentese (pos. NFT)
17. Musical town
18. Mushrooms
19. Bowel cancer (FOB screening)
20. PSA test 1 (high prev)
21. PSA test 2 (low prev)
22. Colorectal cancer

23. Psylicraptis screening
24. Mammography 6 (prob)
25. Mammography 6 (freq)

Describing variables:

1. `scen.lbl` Text label for current scenario.
2. `scen.lng` Language of current scenario.
3. `scen.txt` Description text of current scenario.
4. `popu.lbl` Text label for current population.
5. `cond.lbl` Text label for current condition.
6. `cond.true.lbl` Text label for `cond.true` cases.
7. `cond.false.lbl` Text label for `cond.false` cases.
8. `dec.lbl` Text label for current decision.
9. `dec.pos.lbl` Text label for `dec.pos` cases.
10. `dec.neg.lbl` Text label for `dec.neg` cases.
11. `hi.lbl` Text label for cases of hits `hi`.
12. `mi.lbl` Text label for cases of misses `mi`.
13. `fa.lbl` Text label for cases of false alarms `fa`.
14. `cr.lbl` Text label for cases of correct rejections `cr`.
15. `prev` Value of current prevalence `prev`.
16. `sens` Value of current sensitivity `sens`.
17. `spec` Value of current specificity `spec`.
18. `fart` Value of current false alarm rate `fart`.
19. `N` Current population size `N`.
20. `scen.src` Source information for current scenario.
21. `scen.apa` Source information in APA format.

Note that names of variables (columns) correspond to `init_txt` (to initialize `txt`) and `init_num` (to initialize `num`).

Details

When loading `riskyr`, all scenarios contained in `df.scenarios` are also converted into a list of `riskyr` objects `scenarios`.

Source

See columns `scen.src` and `scen.apa` for a scenario's source information.

`fa` *Frequency of false alarms or false positives (FP).*

Description

`fa` is the frequency of false alarms or false positives (`FP`) in a population of `N` individuals.

Usage

```
fa
```

Format

An object of class `numeric` of length 1.

Details

Definition: `fa` is the frequency of individuals for which `Condition = FALSE` and `Decision = TRUE` (positive).

`fa` is a measure of incorrect classifications (type-I-errors), not an individual case.

Relationships:

1. to probabilities: The frequency `fa` depends on the false alarm rate `fart` (aka. false positive rate, FPR) and is conditional on the prevalence `prev`.
2. to other frequencies: In a population of size `N` the following relationships hold:
   - `N = cond.true + cond.false` (by condition)
   - `N = dec.pos + dec.neg` (by decision)
   - `N = dec.cor + dec.err` (by correspondence of decision to condition)
   - `N = hi + mi + fa + cr` (by condition x decision)

See Also

`fart` is the probability of false alarms (aka. false positive rate `FPR` or `fallout`); `num` contains basic numeric parameters; `init_num` initializes basic numeric parameters; `freq` contains current frequency information; `comp_freq` computes current frequency information; `prob` contains current probability information; `comp_prob` computes current probability information; `is_freq` verifies frequencies.

Other essential parameters: `cr`, `hi`, `mi`, `prev`, `sens`, `spec`

Other frequencies: `N`, `cond.false`, `cond.true`, `cr`, `dec.cor`, `dec.err`, `dec.neg`, `dec.pos`, `hi`, `mi`

`fart` *The false alarm rate (or false positive rate) of a decision process or diagnostic procedure.*

Description

`fart` defines a decision's false alarm rate (or the rate of false positives): The conditional probability of the decision being positive if the condition is FALSE.

Usage

```
fart
```

Format

An object of class `numeric` of length 1.

Details

Understanding or obtaining the false alarm rate `fart`:

- Definition: `fart` is the conditional probability for an incorrect positive decision given that the condition is `FALSE`:
  `fart = p(decision = positive | condition = FALSE)`
  or the probability of a false alarm.
- Perspective: `fart` further classifies the subset of `cond.false` individuals by decision (`fart = fa/cond.false`).
- Alternative names: false positive rate (`FPR`), rate of type-I errors (`alpha`), statistical significance level, `fallout`
- Relationships:
  a. `fart` is the complement of the specificity `spec`:
  `fart = 1 - spec`
  b. `fart` is the opposite conditional probability – but not the complement – of the false discovery rate or false detection rate `FDR`:
  `FDR = p(condition = FALSE | decision = positive)`
- In terms of frequencies, `fart` is the ratio of `fa` divided by `cond.false` (i.e., `fa + cr`):
  `fart = fa/cond.false = fa/(fa + cr)`
- Dependencies: `fart` is a feature of a decision process or diagnostic procedure and a measure of incorrect decisions (false positives).
  However, due to being a conditional probability, the value of `fart` is not intrinsic to the decision process, but also depends on the condition's prevalence value `prev`.

References

Consult Wikipedia for additional information.

See Also

comp_fart computes fart as the complement of spec prob contains current probability information; comp_prob computes current probability information; num contains basic numeric parameters; init_num initializes basic numeric parameters; comp_freq computes current frequency information; is_prob verifies probability inputs.

Other probabilities: FDR, FOR, NPV, PPV, mirt, ppod, prev, sens, spec

Examples

```
fart <- .25     # => sets a false alarm rate of 25%
fart <- 25/100  # => (decision = positive) for 25 out of 100 people with (condition = FALSE)
is_prob(fart)   # => TRUE (as fart is a probability)
```

FDR *The false detection rate of a decision process or diagnostic procedure.*

Description

FDR defines a decision's false detection (or false discovery) rate (FDR): The conditional probability of the condition being FALSE provided that the decision is positive.

Usage

```
FDR
```

Format

An object of class numeric of length 1.

Details

Understanding or obtaining the false detection fate or false discovery rate (FDR):

- Definition: FDR is the conditional probability for the condition being FALSE given a positive decision:

  FDR = p(condition = FALSE | decision = positive)

- Perspective: FDR further classifies the subset of dec.pos individuals by condition (FDR = fa/dec.pos = fa/(hi + fa))
- Alternative names: false discovery rate
- Relationships:

  a. FDR is the complement of the positive predictive value PPV:

  FDR = 1 - PPV

  b. FDR is the opposite conditional probability – but not the complement – of the false alarm rate fart:

  fart = p(decision = positive | condition = FALSE)

- In terms of frequencies, FDR is the ratio of fa divided by dec.pos (i.e., hi + fa):

  FDR = fa/dec.pos = fa/(hi + fa)

- Dependencies: FDR is a feature of a decision process or diagnostic procedure and a measure of incorrect decisions (positive decisions that are actually FALSE).

  However, due to being a conditional probability, the value of FDR is not intrinsic to the decision process, but also depends on the condition's prevalence value prev.

References

Consult Wikipedia for additional information.

See Also

comp_FDR computes FDR as the complement of PPV; prob contains current probability information; comp_prob computes current probability information; num contains basic numeric parameters; init_num initializes basic numeric parameters; comp_freq computes current frequency information; is_prob verifies probability inputs.

Other probabilities: FOR, NPV, PPV, fart, mirt, ppod, prev, sens, spec

Examples

```
FDR <- .45     # => sets a false discovery rate (FDR) of 45%
FDR <- 45/100  # => (condition = FALSE) for 45 out of 100 people with (decision = positive)
is_prob(FDR)   # => TRUE (as FDR is a probability)
```

FOR *The false omission rate (FOR) of a decision process or diagnostic procedure.*

Description

FOR defines a decision's false omission rate (FOR): The conditional probability of the condition being TRUE provided that the decision is negative.

Usage

```
FOR
```

Format

An object of class numeric of length 1.

Details

Understanding or obtaining the false omission rate FOR:

- Definition: FOR is the so-called false omission rate: The conditional probability for the condition being TRUE given a negative decision:

  FOR = p(condition = TRUE | decision = negative)

- Perspective: FOR further classifies the subset of dec.neg individuals by condition (FOR = mi/dec.neg = mi/(mi + cr)
- Alternative names: none?
- Relationships:

  a. FOR is the complement of the negative predictive value NPV:

  FOR = 1 - NPV

  b. FOR is the opposite conditional probability – but not the complement – of the miss rate mirt (aka. false negative rate FDR):

  mirt = p(decision = negative | condition = TRUE)

- In terms of frequencies, FOR is the ratio of mi divided by dec.neg (i.e., mi + cr):

  NPV = mi/dec.neg = mi/(mi + cr)

- Dependencies: FOR is a feature of a decision process or diagnostic procedure and a measure of incorrect decisions (negative decisions that are actually FALSE).

  However, due to being a conditional probability, the value of FOR is not intrinsic to the decision process, but also depends on the condition's prevalence value prev.

References

Consult Wikipedia for additional information.

See Also

comp_FOR computes FOR as the complement of NPV; prob contains current probability information; comp_prob computes current probability information; num contains basic numeric parameters; init_num initializes basic numeric parameters; comp_freq computes current frequency information; is_prob verifies probability inputs.

Other probabilities: FDR, NPV, PPV, fart, mirt, ppod, prev, sens, spec

Examples

```
FOR <- .05     # => sets a false omission rate of 5%
FOR <- 5/100   # => (condition = TRUE) for 5 out of 100 people with (decision = negative)
is_prob(FOR)   # => TRUE (as FOR is a probability)
```

freq *List current frequency information.*

Description

freq is a list of named numeric variables containing 11 frequencies:

Usage

```
freq
```

Format

An object of class list of length 11.

Details

1. the population size N
2. the number of cases for which cond.true
3. the number of cases for which cond.false
4. the number of cases for which dec.pos
5. the number of cases for which dec.neg
6. the number of cases for which dec.cor
7. the number of cases for which dec.err
8. the number of true positives, or hits hi
9. the number of false negatives, or misses mi
10. the number of false positives, or false alarms fa
11. the number of true negatives, or correct rejections cr

These frequencies are computed from basic parameters (contained in num) and computed by using comp_freq.

The list freq is the frequency counterpart to the list containing probability information prob.

Natural frequencies are always expressed in relation to the current population of size N.

Key relationships:

1. to probabilities: A population of N individuals can be split into 2 subsets in 2 different ways:
   (a) by condition: The frequency cond.true depends on the prevalence prev and the frequency cond.false depends on the prevalence's complement (1 - prev).
   (b) by decision: The frequency dec.pos depends on the proportion of positive decisions ppod and the frequency dec.neg depends on the proportion of negative decisions (1 - ppod).

   The population size N is a free parameter (independent of the essential probabilities prev, sens, and spec).

   If N is unknown, a suitable minimum value can be computed by comp_min_N.

2. to other frequencies: In a population of size N the following relationships hold:
   - N = cond.true + cond.false (by condition)
   - N = dec.pos + dec.neg (by decision)
   - N = dec.cor + dec.err (by correspondence of decision to condition)
   - N = hi + mi + fa + cr (by condition x decision)

Visualizations of the current frequency information are provided by plot_tree and plot_mosaic.

See Also

comp_freq computes current frequency information; num contains basic numeric variables; init_num initializes basic numeric variables; prob contains current probability information; num contains basic numeric parameters; init_num initializes basic numeric parameters; txt contains current text information; init_txt initializes text information; pal contains current color information; init_pal initializes color information.

Other lists containing current scenario information: accu, num, pal, prob, txt

Examples

```
freq <- comp_freq()  # => initialize freq to default parameters
freq                 # => show current values
length(freq)         # => 11
```

---

hi *Frequency of hits or true positives (TP).*

Description

hi is the frequency of hits or true positives (TP) in a population of N individuals.

Usage

```
hi
```

Format

An object of class numeric of length 1.

Details

Definition: hi is the frequency of individuals for which Condition = TRUE and Decision = TRUE (positive).

hi is a measure of correct classifications, not an individual case.

Relationships:

1. to probabilities: The frequency hi depends on the sensitivity sens (aka. hit rate or true positive rate, TPR) and is conditional on the prevalence prev.
2. to other frequencies: In a population of size N the following relationships hold:
   - N = cond.true + cond.false (by condition)
   - N = dec.pos + dec.neg (by decision)
   - N = dec.cor + dec.err (by correspondence of decision to condition)
   - N = hi + mi + fa + cr (by condition x decision)

See Also

sens is the probability of hits or hit rate HR; num contains basic numeric parameters; init_num initializes basic numeric parameters; freq contains current frequency information; comp_freq computes current frequency information; prob contains current probability information; comp_prob computes current probability information; is_freq verifies frequencies.

Other frequencies: N, cond.false, cond.true, cr, dec.cor, dec.err, dec.neg, dec.pos, fa, mi

Other essential parameters: cr, fa, mi, prev, sens, spec

---

init_num *Initialize basic numeric variables.*

Description

init_num initializes basic numeric variables to define num as a list of named elements containing four basic probabilities (prev, sens, spec, and fart) and one frequency parameter (the population size N).

Usage

```
init_num(prev = num.def$prev, sens = num.def$sens, spec = num.def$spec,
  fart = num.def$fart, N = num.def$N)
```

Arguments

prev	The condition's prevalence value prev (i.e., the probability of condition being TRUE).
sens	The decision's sensitivity value sens (i.e., the conditional probability of a positive decision provided that the condition is TRUE).
spec	The decision's specificity value spec (i.e., the conditional probability of a negative decision provided that the condition is FALSE). spec is optional when is complement fart is provided.
fart	The decision's false alarm rate fart (i.e., the conditional probability of a positive decision provided that the condition is FALSE). fart is optional when its complement spec is provided.
N	The population size N.

Details

If spec is provided, its complement fart is optional. If fart is provided, its complement spec is optional. If no N is provided, a suitable minimum value is computed by comp_min_N.

Value

A list containing a valid quadruple of probabilities (prev, sens, spec, and fart) and one frequency (population size N).

See Also

num contains basic numeric parameters; pal contains current color settings; txt contains current text settings; freq contains current frequency information; comp_freq computes frequencies from probabilities; prob contains current probability information; comp_prob computes current probability information; is_valid_prob_set verifies sets of probability inputs; is_extreme_prob_set verifies sets of extreme probabilities; comp_min_N computes a suitable minimum population size N.

Other functions initializing scenario information: init_pal, init_txt

Examples

```
# ways to succeed:
init_num(1, 1, 1, 0, 100)  # => succeeds
init_num(1, 1, 0, 1, 100)  # => succeeds

# watch out for:
init_num(1, 1, 0, 1)           # => succeeds (with N computed)
init_num(1, 1, NA, 1, 100)     # => succeeds (with spec computed)
init_num(1, 1, 0, NA, 100)     # => succeeds (with fart computed)
init_num(1, 1, NA, 1)          # => succeeds (with spec and N computed)
init_num(1, 1, 0, NA)          # => succeeds (with fart and N computed)
init_num(1, 1, .51, .50, 100)  # => succeeds (as spec and fart are within tolarated range)

# ways to fail:
init_num(prev = NA)                                  # => NAs + warning (NA)
init_num(prev = 88)                                  # => NAs + warning (beyond range)
init_num(prev =  1, sens = NA)                       # => NAs + warning (NA)
init_num(prev =  1, sens = 1, spec = NA, fart = NA)  # => NAs + warning (NAs)
init_num(1, 1, .52, .50, 100)   # => NAs + warning (complements beyond range)
```

---

init_pal *Initialize basic color information.*

Description

init_pal initializes basic color information (i.e., all colors corresponding to functional roles in the current scenario and used throughout the riskyr package).

Usage

```
init_pal(col.N = pal.def["N"], col.true = pal.def["true"],
  col.false = pal.def["false"], col.pos = pal.def["pos"],
  col.neg = pal.def["neg"], col.hi = pal.def["hi"],
  col.mi = pal.def["mi"], col.fa = pal.def["fa"], col.cr = pal.def["cr"],
  col.ppv = pal.def["ppv"], col.npv = pal.def["npv"])
```

Arguments

col.N	Color representing the *population* of N cases or individuals.
col.true	Color representing cases of cond.true, for which the current condition is TRUE.
col.false	Color representing cases of in cond.false, for which the current condition is FALSE.
col.pos	Color representing cases of dec.pos, for which the current decision is positive.
col.neg	Color representing cases in dec.neg, for which the current decision is negative.
col.hi	Color representing *hits* or true positives in hi (i.e., correct cases for which the current condition is TRUE and the decision is positive).
col.mi	Color representing *misses* or false negatives in mi (i.e., incorrect cases for which the current condition is TRUE but the decision is negative).
col.fa	Color representing *false alarms* or false positives in fa (i.e., incorrect cases for which the current condition is FALSE but the decision is positive).
col.cr	Color representing *correct rejections* or true negatives in cr (i.e., correct cases for which the current condition is FALSE and the decision is negative).
col.ppv	Color representing *positive predictive values* PPV (i.e., the conditional probability that the condition is TRUE, provided that the decision is positive).
col.npv	Color representing *negative predictive values* NPV (i.e., the conditional probability that the condition is FALSE, provided that the decision is negative).

Details

All color information of the current scenario is stored as named colors in a list pal. init_pal allows changing colors by assigning new colors to existing names.

See Also

num contains basic numeric parameters; init_num initializes basic numeric parameters; txt contains current text information; init_txt initializes text information; pal contains current color information; init_pal initializes color information; freq contains current frequency information; comp_freq computes current frequency information; prob contains current probability information; comp_prob computes current probability information.

Other functions initializing scenario information: init_num, init_txt

Examples

```
init_pal()          # => define and return a vector of current (default) colors
length(init_pal())  # => 11 colors
pal <- init_pal(col.false = "firebrick2")  # => change current color (stored in pal)
```

`init_txt` *Initialize basic text elements.*

Description

`init_txt` initializes basic text elements (i.e., all titles and labels corresponding to the current scenario and used throughout the `riskyr` package).

Usage

```
init_txt(scen.lbl = txt.def$scen.lbl, scen.txt = txt.def$scen.txt,
  scen.src = txt.def$scen.src, scen.apa = txt.def$scen.apa,
  scen.lng = txt.def$scen.lng, popu.lbl = txt.def$popu.lbl,
  cond.lbl = txt.def$cond.lbl, cond.true.lbl = txt.def$cond.true.lbl,
  cond.false.lbl = txt.def$cond.false.lbl, dec.lbl = txt.def$dec.lbl,
  dec.pos.lbl = txt.def$dec.pos.lbl, dec.neg.lbl = txt.def$dec.neg.lbl,
  hi.lbl = txt.def$hi.lbl, mi.lbl = txt.def$mi.lbl,
  fa.lbl = txt.def$fa.lbl, cr.lbl = txt.def$cr.lbl)
```

Arguments

`scen.lbl`	The current scenario title (sometimes in Title Caps).
`scen.txt`	A longer text description of the current scenario (which may extend over several lines).
`scen.src`	The source information for the current scenario.
`scen.apa`	Source information in APA format.
`scen.lng`	Language of the current scenario (as character code). Options: `"en"`...English, `"de"`... German.
`popu.lbl`	A brief description of the current target population `popu` or sample.
`cond.lbl`	A name for the *condition* or feature (e.g., some disease) currently considered.
`cond.true.lbl`	A label for the *presence* of the current condition or `cond.true` cases (the condition's true state of TRUE).
`cond.false.lbl`	A label for the *absence* of the current condition or `cond.false` cases (the condition's true state of FALSE).
`dec.lbl`	A name for the *decision* or judgment (e.g., some diagnostic test) currently made.
`dec.pos.lbl`	A label for *positive* decisions or `dec.pos` cases (e.g., predicting the presence of the condition).

dec.neg.lbl	A label for *negative* decisions or dec.neg cases (e.g., predicting the absence of the condition).
hi.lbl	A label for *hits* or *true positives* hi (i.e., correct decisions of the presence of the condition, when the condition is actually present).
mi.lbl	A label for *misses* or *false negatives* mi (i.e., incorrect decisions of the absence of the condition when the condition is actually present).
fa.lbl	A label for *false alarms* or *false positives* fa (i.e., incorrect decisions of the presence of the condition when the condition is actually absent).
cr.lbl	A label for *correct rejections* or *true negatives* cr (i.e., a correct decision of the absence of the condition, when the condition is actually absent).

Details

All textual elements that specify titles and details of the current scenario are stored as named elements (of type character) in a list txt. init_txt allows changing elements by assigning new character objects to existing names.

See Also

txt for current text settings; pal for current color settings; num for basic numeric parameters

Other functions initializing scenario information: init_num, init_pal

Examples

```
init_txt()          # => defines a list of (default) text elements
length(init_txt())  # => 16

# Customizing current text elements:
txt <- init_txt(scen.lbl = "US or Them",
                scen.src = "Some stable genius",
                popu.lbl = "We, the people")
```

---

is_complement *Verify that two numbers are complements.*

Description

is_complement is a function that takes 2 numeric arguments (typically probabilities) as inputs and verifies that they are *complements* (i.e., add up to 1).

Usage

```
is_complement(p1, p2, tol = 0.01)
```

Arguments

p1	A numeric argument (typically probability in range from 0 to 1).
p2	A numeric argument (typically probability in range from 0 to 1).
tol	A numeric tolerance value. Default: `tol = .01`.

Details

Both p1 and p2 are necessary arguments. If one or both arguments are NA, `is_complement` returns NA (i.e., neither TRUE nor FALSE).

The argument `tol` is optional (with a default value of .01) Numeric near-complements that differ by less than this value are still considered to be complements.

This function does not verify the type, range, or sufficiency of the inputs provided. See `is_prob` and `is_suff_prob_set` for this purpose.

Value

NA or a Boolean value: NA if one or both arguments are NA; TRUE if both arguments are provided and complements (in `tol` range); otherwise FALSE.

See Also

`comp_complement` computes a probability's complement; `comp_comp_pair` computes pairs of complements; `num` contains basic numeric variables; `init_num` initializes basic numeric variables; `prob` contains current probability information; `comp_prob` computes current probability information; `freq` contains current frequency information; `comp_freq` computes current frequency information; `is_valid_prob_set` verifies the validity of probability inputs; `as_pc` displays a probability as a percentage; `as_pb` displays a percentage as probability.

Other verification functions: `is_extreme_prob_set`, `is_freq`, `is_perc`, `is_prob`, `is_suff_prob_set`, `is_valid_prob_pair`, `is_valid_prob_set`, `is_valid_prob_triple`

Examples

```
# Basics:
is_complement(0, 1)           # => TRUE
is_complement(1/3, 2/3)       # => TRUE
is_complement(.33, .66)       # => TRUE  (as within default tol = .01)
is_complement(.33, .65)       # => FALSE (as beyond default tol = .01)

# watch out for:
is_complement(NA, NA)            # => NA (but not FALSE)
is_complement(1, NA)             # => NA (but not FALSE)
is_complement(2, -1)             # => TRUE + warnings (p1 and p2 beyond range)
is_complement(8, -7)             # => TRUE + warnings (p1 and p2 beyond range)
is_complement(.3, .6)            # => FALSE + warning (beyond tolerance)
is_complement(.3, .6, tol = .1)  # => TRUE (due to increased tolerance)

## ways to fail:
# is_complement(0, 0)            # => FALSE + warning (beyond tolerance)
# is_complement(1, 1)            # => FALSE + warning (beyond tolerance)
```

```
# is_complement(8, 8)           # => FALSE + warning (beyond tolerance)
```

`is_extreme_prob_set` *Verify that a set of probabilities describes an extreme case.*

Description

`is_extreme_prob_set` verifies that a set of probabilities (i.e., `prev`, and `sens` or `mirt`, and `spec` or `fart`) describe an extreme case.

Usage

```
is_extreme_prob_set(prev, sens = NA, mirt = NA, spec = NA, fart = NA)
```

Arguments

`prev`	The condition's prevalence value `prev` (i.e., the probability of condition being `TRUE`).
`sens`	The decision's sensitivity `sens` (i.e., the conditional probability of a positive decision provided that the condition is `TRUE`). `sens` is optional when is complement `mirt` is provided.
`mirt`	The decision's miss rate `mirt` (i.e., the conditional probability of a negative decision provided that the condition is `TRUE`). `mirt` is optional when is complement `sens` is provided.
`spec`	The decision's specificity `spec` (i.e., the conditional probability of a negative decision provided that the condition is `FALSE`). `spec` is optional when is complement `fart` is provided.
`fart`	The decision's false alarm rate `fart` (i.e., the conditional probability of a positive decision provided that the condition is `FALSE`). `fart` is optional when its complement `spec` is provided.

Details

If `TRUE`, a warning message describing the nature of the extreme case is printed to allow anticipating peculiar effects (e.g., that `PPV` or `NPV` values cannot be computed or are `NaN`).

This function does not verify the type, range, sufficiency, or consistency of its arguments. See `is_prob`, `is_suff_prob_set`, `is_complement`, `is_valid_prob_pair` and `is_valid_prob_set` for these purposes.

Value

A Boolean value: `TRUE` if an extreme case is identified; otherwise `FALSE`.

See Also

is_valid_prob_pair verifies that a pair of probabilities can be complements; is_valid_prob_set verifies the validity of a set of probability inputs; num contains basic numeric variables; init_num initializes basic numeric variables; prob contains current probability information; comp_prob computes current probability information; freq contains current frequency information; comp_freq computes current frequency information; as_pc displays a probability as a percentage; as_pb displays a percentage as probability

Other verification functions: is_complement, is_freq, is_perc, is_prob, is_suff_prob_set, is_valid_prob_pair, is_valid_prob_set, is_valid_prob_triple

Examples

```
# Identify 6 extreme cases (+ 4 variants):
is_extreme_prob_set(1, 1, NA, 1, NA)       # => TRUE + warning: N true positives
plot_tree(1, 1, NA, 1, NA, N = 100)        # => illustrates this case

is_extreme_prob_set(1, 0, NA, 1, NA)       # => TRUE + warning: N false negatives
plot_tree(1, 0, NA, 1, NA, N = 200)        # => illustrates this case

sens <- .50
is_extreme_prob_set(0, sens, NA, 0, NA)    # => TRUE + warning: N false positives
plot_tree(0, sens, NA, 0, N = 300)         # => illustrates this case
# Variant:
is_extreme_prob_set(0, sens, NA, NA, 1)    # => TRUE + warning: N false positives
plot_tree(0, sens, NA, NA, 1, N = 350)     # => illustrates this case

sens <- .50
is_extreme_prob_set(0, sens, NA, 1)        # => TRUE + warning: N true negatives
plot_tree(0, sens, NA, NA, 1, N = 400)     # => illustrates this case
# Variant:
is_extreme_prob_set(0, sens, NA, NA, 0)    # => TRUE + warning: N true negatives
plot_tree(0, sens, NA, NA, 0, N = 450)     # => illustrates this case

prev <- .50
is_extreme_prob_set(prev, 0, NA, 1, NA)   # => TRUE + warning: 0 hi and 0 fa (0 dec.pos cases)
plot_tree(prev, 0, NA, 1, NA, N = 500)     # => illustrates this case
# # Variant:
is_extreme_prob_set(prev, 0, 0, NA, 0)    # => TRUE + warning: 0 hi and 0 fa (0 dec.pos cases)
plot_tree(prev, 0, NA, 1, NA, N = 550)     # => illustrates this case

prev <- .50
is_extreme_prob_set(prev, 1, NA, 0, NA)   # => TRUE + warning: 0 mi and 0 cr (0 dec.neg cases)
plot_tree(prev, 1, NA, 0, NA, N = 600)     # => illustrates this case
# # Variant:
is_extreme_prob_set(prev, 1, NA, 0, NA)   # => TRUE + warning: 0 mi and 0 cr (0 dec.neg cases)
plot_tree(prev, 1, NA, 0, NA, N = 650)     # => illustrates this case
```

is_freq *Verify that input is a frequency (positive integer value).*

Description

is_freq is a function that checks whether its single argument freq is a frequency (i.e., a positive numeric integer value).

Usage

```
is_freq(freq)
```

Arguments

freq	A single (typically numeric) argument.

Value

A Boolean value: TRUE if freq is a frequency (positive integer), otherwise FALSE.

See Also

num contains basic numeric variables; init_num initializes basic numeric variables; prob contains current probability information; comp_prob computes current probability information; freq contains current frequency information; comp_freq computes current frequency information; is_valid_prob_set verifies the validity of probability inputs; as_pc displays a probability as a percentage; as_pb displays a percentage as probability

Other verification functions: is_complement, is_extreme_prob_set, is_perc, is_prob, is_suff_prob_set, is_valid_prob_pair, is_valid_prob_set, is_valid_prob_triple

Examples

```
# ways to succeed:
is_freq(2)             # => TRUE, but does NOT return the frequency 2.
is_freq(1:3)           # => TRUE (for vector)

# ways to fail:
is_freq(-1)            # => FALSE + warning (negative values)
is_freq(1:-1)          # => FALSE (for vector) + warning (negative values)
is_freq(c(1, 1.5, 2))  # => FALSE (for vector) + warning (non-integer values)

## Note that:
is.integer(2)          # => FALSE!
```

is_perc *Verify that input is a percentage (numeric value from 0 to 100).*

Description

is_perc is a function that checks whether its single argument perc is a percentage (proportion, i.e., a numeric value in the range from 0 to 100).

Usage

```
is_perc(perc)
```

Arguments

perc	A single (typically numeric) argument.

Value

A Boolean value: TRUE if perc is a percentage (proportion), otherwise FALSE.

See Also

num contains basic numeric variables; init_num initializes basic numeric variables; prob contains current probability information; comp_prob computes current probability information; freq contains current frequency information; comp_freq computes current frequency information; is_valid_prob_set verifies the validity of probability inputs; as_pc displays a probability as a percentage; as_pb displays a percentage as probability

Other verification functions: is_complement, is_extreme_prob_set, is_freq, is_prob, is_suff_prob_set, is_valid_prob_pair, is_valid_prob_set, is_valid_prob_triple

Examples

```
# ways to succeed:
is_perc(2)           # => TRUE, but does NOT return the percentage 2.
is_perc(1/2)         # => TRUE, but does NOT return the percentage 0.5.
pc.sq <- seq(0, 100, by = 10)
is_perc(pc.sq)       # => TRUE (for vector)

# ways to fail:
is_perc(NA)          # => FALSE + warning (NA values)
is_perc(NaN)         # => FALSE + warning (NaN values)
is_perc("Bernoulli") # => FALSE + warning (non-numeric values)
is_perc(101)         # => FALSE + warning (beyond range)
```

is_prob *Verify that input is a probability (numeric value from 0 to 1).*

Description

is_prob is a function that checks whether its argument prob is a probability (i.e., a numeric value in the range from 0 to 1).

Usage

```
is_prob(prob, NA.warn = FALSE)
```

Arguments

prob	A numeric argument (scalar or vector) that is to be checked.
NA.warn	Boolean value determining whether a warning is shown for NA values. Default: NA.warn = FALSE.

Value

A Boolean value: TRUE if prob is a probability, otherwise FALSE.

See Also

num contains basic numeric variables; init_num initializes basic numeric variables; prob contains current probability information; comp_prob computes current probability information; freq contains current frequency information; comp_freq computes current frequency information; is_valid_prob_set verifies the validity of probability inputs; as_pc displays a probability as a percentage; as_pb displays a percentage as probability.

Other verification functions: is_complement, is_extreme_prob_set, is_freq, is_perc, is_suff_prob_set, is_valid_prob_pair, is_valid_prob_set, is_valid_prob_triple

Examples

```
# ways to succeed:
is_prob(1/2)                  # => TRUE
p.seq <- seq(0, 1, by = .1)
is_prob(p.seq)                # => TRUE (for vector)

# watch out for:
is_prob(NA)                   # => FALSE + NO warning!
is_prob(0/0)                  # => FALSE + NO warning (NA + NaN values)
is_prob(0/0, NA.warn = TRUE)  # => FALSE + warning (NA values)

# ways to fail:
is_prob(8)                    # => FALSE + warning (outside range)
is_prob(c(.5, 8))             # => FALSE + warning (for vector)
is_prob("Laplace")            # => FALSE + warning (non-numeric values)
```

`is_suff_prob_set` *Verify a sufficient set of probability inputs.*

Description

`is_suff_prob_set` is a function that takes 3 to 5 probabilities as inputs and verifies that they are sufficient to compute all derived probabilities and combined frequencies for a population of `N` individuals.

Usage

```
is_suff_prob_set(prev, sens = NA, mirt = NA, spec = NA, fart = NA)
```

Arguments

`prev`	The condition's prevalence `prev` (i.e., the probability of condition being `TRUE`).
`sens`	The decision's sensitivity `sens` (i.e., the conditional probability of a positive decision provided that the condition is `TRUE`). `sens` is optional when its complement `mirt` is provided.
`mirt`	The decision's miss rate `mirt` (i.e., the conditional probability of a negative decision provided that the condition is `TRUE`). `mirt` is optional when its complement `sens` is provided.
`spec`	The decision's specificity value `spec` (i.e., the conditional probability of a negative decision provided that the condition is `FALSE`). `spec` is optional when its complement `fart` is provided.
`fart`	The decision's false alarm rate `fart` (i.e., the conditional probability of a positive decision provided that the condition is `FALSE`). `fart` is optional when its complement `spec` is provided.

Details

While no alternative input option for frequencies is provided, specification of the essential probability `prev` is always necessary.

However, for two other essential probabilities there is a choice:

1. Either `sens` or `mirt` is necessary (as both are complements).
2. Either `spec` or `fart` is necessary (as both are complements).

`is_suff_prob_set` does not verify the type, range, or consistency of its arguments. See `is_prob` and `is_complement` for this purpose.

Value

A Boolean value: `TRUE` if the probabilities provided are sufficient, otherwise `FALSE`.

See Also

num contains basic numeric variables; init_num initializes basic numeric variables; prob contains current probability information; comp_prob computes current probability information; freq contains current frequency information; comp_freq computes current frequency information; is_valid_prob_set verifies the validity of probability inputs; as_pc displays a probability as a percentage; as_pb displays a percentage as probability.

Other verification functions: is_complement, is_extreme_prob_set, is_freq, is_perc, is_prob, is_valid_prob_pair, is_valid_prob_set, is_valid_prob_triple

Examples

```
# ways to work:
is_suff_prob_set(prev = 1, sens = 1, spec = 1)  # => TRUE
is_suff_prob_set(prev = 1, mirt = 1, spec = 1)  # => TRUE
is_suff_prob_set(prev = 1, sens = 1, fart = 1)  # => TRUE
is_suff_prob_set(prev = 1, mirt = 1, fart = 1)  # => TRUE

# watch out for:
is_suff_prob_set(prev = 1, sens = 2, spec = 3)  # => TRUE, but is_prob is FALSE
is_suff_prob_set(prev = 1, mirt = 2, fart = 4)  # => TRUE, but is_prob is FALSE
is_suff_prob_set(prev = 1, sens = 2, spec = 3, fart = 4)  # => TRUE, but is_prob is FALSE

## ways to fail:
# is_suff_prob_set()                    # => FALSE + warning (prev missing)
# is_suff_prob_set(prev = 1)            # => FALSE + warning (sens or mirt missing)
# is_suff_prob_set(prev = 1, sens = 1)  # => FALSE + warning (spec or fart missing)
```

---

is_valid_prob_pair *Verify that a pair of probability inputs can be a pair of complementary probabilities.*

Description

is_valid_prob_pair is a function that verifies that a pair of 2 numeric inputs p1 and p2 can be interpreted as a valid pair of probabilities.

Usage

```
is_valid_prob_pair(p1, p2, tol = 0.01)
```

Arguments

p1	A numeric argument (typically probability in range from 0 to 1).
p2	A numeric argument (typically probability in range from 0 to 1).
tol	A numeric tolerance value.

Details

is_valid_prob_pair is a wrapper function that combines is_prob and is_complement in one function.

Either p1 or p2 must be a probability (verified via is_prob). If both arguments are provided they must be probabilities and complements (verified via is_complement).

The argument tol is optional (with a default value of .01) Numeric near-complements that differ by less than this value are still considered to be complements.

Value

A Boolean value: TRUE if exactly one argument is a probability, if both arguments are probabilities and complements, otherwise FALSE.

See Also

is_valid_prob_set uses this function to verify sets of probability inputs; is_complement verifies numeric complements; is_prob verifies probabilities; num contains basic numeric variables; init_num initializes basic numeric variables; prob contains current probability information; comp_prob computes current probability information; freq contains current frequency information; comp_freq computes current frequency information; as_pc displays a probability as a percentage; as_pb displays a percentage as probability.

Other verification functions: is_complement, is_extreme_prob_set, is_freq, is_perc, is_prob, is_suff_prob_set, is_valid_prob_set, is_valid_prob_triple

Examples

```
# ways to succeed:
is_valid_prob_pair(1, 0)      # => TRUE
is_valid_prob_pair(0, 1)      # => TRUE
is_valid_prob_pair(1, NA)     # => TRUE + warning (NA)
is_valid_prob_pair(NA, 1)     # => TRUE + warning (NA)
is_valid_prob_pair(.50, .51)  # => TRUE (as within tol)

# ways to fail:
is_valid_prob_pair(.50, .52)  # => FALSE (as beyond tol)
is_valid_prob_pair(1, 2)      # => FALSE + warning (beyond range)
is_valid_prob_pair(NA, NA)    # => FALSE + warning (NA)
```

---

is_valid_prob_set *Verify that a set of probability inputs is valid.*

Description

is_valid_prob_set is a function that verifies that a set of (3 to 5) numeric inputs can be interpreted as a valid set of (3 essential and 2 optional) probabilities.

Usage

```
is_valid_prob_set(prev, sens = NA, mirt = NA, spec = NA, fart = NA,
  tol = 0.01)
```

Arguments

prev	The condition's prevalence prev (i.e., the probability of condition being TRUE).
sens	The decision's sensitivity sens (i.e., the conditional probability of a positive decision provided that the condition is TRUE). sens is optional when its complement mirt is provided.
mirt	The decision's miss rate mirt (i.e., the conditional probability of a negative decision provided that the condition is TRUE). mirt is optional when its complement sens is provided.
spec	The decision's specificity value spec (i.e., the conditional probability of a negative decision provided that the condition is FALSE). spec is optional when its complement fart is provided.
fart	The decision's false alarm rate fart (i.e., the conditional probability of a positive decision provided that the condition is FALSE). fart is optional when its complement spec is provided.
tol	A numeric tolerance value used by is_complement.

Details

is_valid_prob_set is a wrapper function that combines is_prob, is_suff_prob_set, and is_complement in one function.

While no alternative input option for frequencies is provided, specification of the essential probability prev is always necessary. However, for 2 other essential probabilities there is a choice:

1. Either sens or mirt is necessary (as both are complements).
2. Either spec or fart is necessary (as both are complements).

The argument tol is optional (with a default value of .01) and used as the tolerance value of is_complement.

is_valid_prob_set verifies the validity of inputs, but does not compute or return numeric variables. Use is_extreme_prob_set to verify sets of probabilities that describe extreme cases and init_num for initializing basic parameters.

Value

A Boolean value: TRUE if the probabilities provided are valid; otherwise FALSE.

See Also

is_valid_prob_pair verifies that probability pairs are complements; num contains basic numeric variables; init_num initializes basic numeric variables; prob contains current probability information; comp_prob computes current probability information; freq contains current frequency information; comp_freq computes current frequency information; as_pc displays a probability as a percentage; as_pb displays a percentage as probability.

Other verification functions: is_complement, is_extreme_prob_set, is_freq, is_perc, is_prob, is_suff_prob_set, is_valid_prob_pair, is_valid_prob_triple

Examples

```
## ways to succeed:
is_valid_prob_set(1, 1, 0, 1, 0)                # => TRUE
is_valid_prob_set(.3, .9, .1, .8, .2)           # => TRUE
is_valid_prob_set(.3, .9, .1, .8, NA)           # => TRUE + warning (NA)
is_valid_prob_set(.3, .9, NA, .8, NA)           # => TRUE + warning (NAs)
is_valid_prob_set(.3, .9, NA, NA, .8)           # => TRUE + warning (NAs)
is_valid_prob_set(.3, .8, .1, .7, .2, tol = .1)  # => TRUE (due to increased tol)

## watch out for:
is_valid_prob_set(1, 0, 1, 0, 1)    # => TRUE, but NO warning about extreme case!
is_valid_prob_set(1, 1, 0, 1, 0)    # => TRUE, but NO warning about extreme case!
is_valid_prob_set(1, 1, 0, 1, NA)   # => TRUE, but NO warning about extreme case!
is_valid_prob_set(1, 1, 0, NA, 1)   # => TRUE, but NO warning about extreme case!
is_valid_prob_set(1, 1, 0, NA, 0)   # => TRUE, but NO warning about extreme case!

## ways to fail:
is_valid_prob_set(8, 1, 0, 1, 0)      # => FALSE + warning (is_prob fails)
is_valid_prob_set(1, 1, 8, 1, 0)      # => FALSE + warning (is_prob fails)
is_valid_prob_set(2, 1, 3, 1, 4)      # => FALSE + warning (is_prob fails)
is_valid_prob_set(1, .8, .2, .7, .2)  # => FALSE + warning (beyond complement range)
is_valid_prob_set(1, .8, .3, .7, .3)  # => FALSE + warning (beyond complement range)
is_valid_prob_set(1, 1, 1, 1, 1)      # => FALSE + warning (beyond complement range)
is_valid_prob_set(1, 1, 0, 1, 1)      # => FALSE + warning (beyond complement range)
```

---

is_valid_prob_triple *Verify that a triple of essential probability inputs is valid.*

Description

is_valid_prob_triple is a **deprecated** function that verifies that a set of 3 numeric inputs can be interpreted as a valid set of 3 probabilities.

Usage

```
is_valid_prob_triple(prev, sens, spec)
```

Arguments

prev	The condition's prevalence prev (i.e., the probability of condition being TRUE).
sens	The decision's sensitivity sens (i.e., the conditional probability of a positive decision provided that the condition is TRUE).
spec	The decision's specificity value spec (i.e., the conditional probability of a negative decision provided that the condition is FALSE).

Details

is_valid_prob_triple is a simplified version of is_valid_prob_set. It is a quick wrapper function that only verifies is_prob for all of its 3 arguments.

is_valid_prob_triple does not compute or return numeric variables. Use is_extreme_prob_set to verify extreme cases and comp_complete_prob_set to complete sets of valid probabilities.

Value

A Boolean value: TRUE if the probabilities provided are valid; otherwise FALSE.

See Also

is_extreme_prob_set verifies extreme cases; is_valid_prob_set verifies sets of probability inputs; is_valid_prob_pair verifies that probability pairs are complements; num contains basic numeric variables; init_num initializes basic numeric variables; prob contains current probability information; comp_prob computes current probability information; freq contains current frequency information; comp_freq computes current frequency information; as_pc displays a probability as a percentage; as_pb displays a percentage as probability.

Other verification functions: is_complement, is_extreme_prob_set, is_freq, is_perc, is_prob, is_suff_prob_set, is_valid_prob_pair, is_valid_prob_set

Examples

```
# ways to work:
is_valid_prob_triple(0, 0, 0)    # => TRUE
is_valid_prob_triple(1, 1, 1)    # => TRUE

## ways to fail:
# is_valid_prob_triple(0, 0)       # => ERROR (as no triple)
# is_valid_prob_triple(0, 0, 7)    # => FALSE + warning (beyond range)
# is_valid_prob_triple(0, NA, 0)   # => FALSE + warning (NA)
# is_valid_prob_triple("p", 0, 0)  # => FALSE + warning (non-numeric)
```

---

mi *Frequency of misses or false negatives (FN).*

Description

mi is the frequency of misses or false negatives (FN) in a population of N individuals.

Usage

```
mi
```

Format

An object of class numeric of length 1.

Details

Definition: `mi` is the frequency of individuals for which `Condition = TRUE` and `Decision = FALSE` (negative).

`mi` is a measure of incorrect classifications (type-II errors), not an individual case.

Relationships:

1. to probabilities: The frequency `mi` depends on the miss rate `mirt` (aka. false negative rate, FNR) and is conditional on the prevalence `prev`.
2. to other frequencies: In a population of size `N` the following relationships hold:
   - `N = cond.true + cond.false` (by condition)
   - `N = dec.pos + dec.neg` (by decision)
   - `N = dec.cor + dec.err` (by correspondence of decision to condition)
   - `N = hi + mi + fa + cr` (by condition x decision)

See Also

`mirt` is the probability or rate of misses; `num` contains basic numeric parameters; `init_num` initializes basic numeric parameters; `freq` contains current frequency information; `comp_freq` computes current frequency information; `prob` contains current probability information; `comp_prob` computes current probability information; `is_freq` verifies frequencies.

Other essential parameters: `cr`, `fa`, `hi`, `prev`, `sens`, `spec`

Other frequencies: `N`, `cond.false`, `cond.true`, `cr`, `dec.cor`, `dec.err`, `dec.neg`, `dec.pos`, `fa`, `hi`

---

mirt *The miss rate of a decision process or diagnostic procedure.*

Description

`mirt` defines a decision's miss rate value: The conditional probability of the decision being negative if the condition is `TRUE`.

Usage

```
mirt
```

Format

An object of class `numeric` of length 1.

Details

Understanding or obtaining the miss rate `mirt`:

- Definition: `sens` is the conditional probability for an incorrect negative decision given that the condition is `TRUE`:

  `mirt = p(decision = negative | condition = TRUE)`

  or the probability of failing to detect true cases (`condition = TRUE`).
- Perspective: `mirt` further classifies the subset of `cond.true` individuals by decision (`mirt = mi/cond.true`).
- Alternative names: false negative rate (`FNR`), rate of type-II errors (`beta`)
- Relationships:

  a. `mirt` is the complement of the sensitivity `sens` (aka. hit rate `HR`):

  `mirt = (1 - sens) = (1 - HR)`

  b. `mirt` is the _opposite_ conditional probability – but not the complement – of the false omission rate `FOR`:

  `FOR = p(condition = TRUE | decision = negative)`
- In terms of frequencies, `mirt` is the ratio of `mi` divided by `cond.true` (i.e., `hi + mi`):

  `mirt = mi/cond.true = mi/(hi + mi)`
- Dependencies: `mirt` is a feature of a decision process or diagnostic procedure and a measure of incorrect decisions (false negatives).

  However, due to being a conditional probability, the value of `mirt` is not intrinsic to the decision process, but also depends on the condition's prevalence value `prev`.

References

Consult Wikipedia for additional information.

See Also

`comp_mirt` computes `mirt` as the complement of `sens`; `prob` contains current probability information; `comp_prob` computes current probability information; `num` contains basic numeric parameters; `init_num` initializes basic numeric parameters; `comp_freq` computes current frequency information; `is_prob` verifies probability inputs.

Other probabilities: `FDR`, `FOR`, `NPV`, `PPV`, `fart`, `ppod`, `prev`, `sens`, `spec`

Examples

```
mirt <- .15     # => sets a miss rate of 15%
mirt <- 15/100  # => (decision = negative) for 15 out of 100 people with (condition = TRUE)
is_prob(mirt)   # => TRUE (as mirt is a probability)
```

N *Number of individuals in the population.*

Description

N is a frequency that describes the number of individuals in the current population (i.e., the overall number of cases considered).

Usage

```
N
```

Format

An object of class numeric of length 1.

Details

Key relationships:

1. to probabilities: A population of N individuals can be split into 2 subsets in 2 different ways:
   (a) by condition: The frequency cond.true depends on the prevalence prev and the frequency cond.false depends on the prevalence's complement 1 - prev.
   (b) by decision: The frequency dec.pos depends on the proportion of positive decisions ppod and the frequency dec.neg depends on the proportion of negative decisions 1 - ppod.

   The population size N is a free parameter (independent of the essential probabilities prev, sens, and spec).

   If N is unknown, a suitable minimum value can be computed by comp_min_N.
2. to other frequencies: In a population of size N the following relationships hold:
   - N = cond.true + cond.false (by condition)
   - N = dec.pos + dec.neg (by decision)
   - N = dec.cor + dec.err (by correspondence of decision to condition)
   - N = hi + mi + fa + cr (by condition x decision)

Current frequency information is computed by comp_freq and contained in a list freq.

References

Consult Wikipedia: Statistical population for additional information.

See Also

is_freq verifies frequencies; num contains basic numeric parameters; init_num initializes basic numeric parameters; freq contains current frequency information; comp_freq computes current frequency information; prob contains current probability information; comp_prob computes current probability information.

Other frequencies: cond.false, cond.true, cr, dec.cor, dec.err, dec.neg, dec.pos, fa, hi, mi

Examples

```
N <- 1000   # => sets a population size of 1000
is_freq(N)  # => TRUE
is_prob(N)  # => FALSE (as N is no probability)
```

NPV *The negative predictive value of a decision process or diagnostic procedure.*

Description

NPV defines some decision's negative predictive value (NPV): The conditional probability of the condition being FALSE provided that the decision is negative.

Usage

```
NPV
```

Format

An object of class numeric of length 1.

Details

Understanding or obtaining the negative predictive value NPV:

- Definition: NPV is the conditional probability for the condition being FALSE given a negative decision:
  `NPV = p(condition = FALSE | decision = negative)`
  or the probability of a negative decision being correct.
- Perspective: NPV further classifies the subset of dec.neg individuals by condition (`NPV = cr/dec.neg = cr/(mi + cr)`)
- Alternative names: true omission rate
- Relationships:
  a. NPV is the complement of the false omission rate FOR:
  `NPV = 1 - FOR`
  b. NPV is the opposite conditional probability – but not the complement – of the specificity spec:
  `spec = p(decision = negative | condition = FALSE)`
- In terms of frequencies, NPV is the ratio of cr divided by dec.neg (i.e., `cr + mi`):
  `NPV = cr/dec.neg = cr/(cr + mi)`
- Dependencies: NPV is a feature of a decision process or diagnostic procedure and – similar to the specificity spec – a measure of correct decisions (negative decisions that are actually FALSE).
  However, due to being a conditional probability, the value of NPV is not intrinsic to the decision process, but also depends on the condition's prevalence value prev.

References

Consult Wikipedia for additional information.

See Also

comp_NPV computes NPV; prob contains current probability information; comp_prob computes current probability information; num contains basic numeric parameters; init_num initializes basic numeric parameters; comp_freq computes current frequency information; is_prob verifies probability inputs.

Other probabilities: FDR, FOR, PPV, fart, mirt, ppod, prev, sens, spec

Examples

```
NPV <- .95     # => sets a negative predictive value of 95%
NPV <- 95/100  # => (condition = FALSE) for 95 out of 100 people with (decision = negative)
is_prob(NPV)   # => TRUE (as NPV is a probability)
```

---

num *List current values of basic numeric variables.*

Description

num is a list of named numeric variables containing 4 basic probabilities (prev, sens, spec, and fart) and 1 frequency parameter (the population size N).

Usage

```
num
```

Format

An object of class list of length 5.

See Also

init_num initializes basic numeric parameters; txt contains current text information; init_txt initializes text information; pal contains current color information; init_pal initializes color information; freq contains current frequency information; comp_freq computes current frequency information; prob contains current probability information; comp_prob computes current probability information.

Other lists containing current scenario information: accu, freq, pal, prob, txt

Examples

```
num <- init_num()  # => initialize num to default parameters
num                # => show defaults
length(num)        # => 5
```

pal *List current values of basic color information.*

Description

pal is initialized to a vector of named elements (colors) to define the color scheme for the current scenario that is used throughout the riskyr package.

Usage

```
pal
```

Format

An object of class character of length 11.

Details

All color information corresponding to the current scenario is stored as named colors in a vector pal. To change a color, assign a new color to an existing element name.

pal currently contains colors with the following names:

1. N Color representing the *population* of N cases or individuals.
2. true Color representing cases of cond.true, for which the current condition is TRUE.
3. false Color representing cases of in cond.false, for which the current condition is FALSE.
4. pos Color representing cases of dec.pos, for which the current decision is positive.
5. neg Color representing cases in dec.neg, for which the current decision is negative.
6. hi Color representing *hits* or true positives in hi (i.e., correct cases for which the current condition is TRUE and the decision is positive).
7. mi Color representing *misses* or false negatives in mi (i.e., incorrect cases for which the current condition is TRUE but the decision is negative).
8. fa Color representing *false alarms* or false positives in fa (i.e., incorrect cases for which the current condition is FALSE but the decision is positive).
9. cr Color representing *correct rejections* or true negatives in cr (i.e., correct cases for which the current condition is FALSE and the decision is negative).
10. ppv Color representing *positive predictive values* PPV (i.e., the conditional probability that the condition is TRUE, provided that the decision is positive).
11. npv Color representing *negative predictive values* NPV (i.e., the conditional probability that the condition is FALSE, provided that the decision is negative).

See Also

init_pal initializes color information; num contains basic numeric parameters; init_num initializes basic numeric parameters; txt contains current text information; init_txt initializes text information; freq contains current frequency information; comp_freq computes current frequency information; prob contains current probability information; comp_prob computes current probability information.

Other lists containing current scenario information: accu, freq, num, prob, txt

Examples

```
pal       # displays the vector of all current color names and values
pal["hi"] # displays the current color for hits (true positives)
pal["hi"] <- "green3" # defines a new color for hits (true positives)
```

---

plot.riskyr *Plot information of a riskyr object.*

Description

plot.riskyr is a method that allows to generate different plot types from a "riskyr" object.

Usage

```
## S3 method for class 'riskyr'
plot(x = NULL, plot.type = "network", ...)
```

Arguments

x	An object of class "riskyr", usually a result of a call to riskyr. Pre-defined scenarios are also of type "riskyr".
plot.type	The type of plot to be generated.

1. plot.type = "fnet" or plot.type = "network": Risk information is plotted in a network diagram of frequencies and probabilities (default). See plot_fnet for further options.
2. plot.type = "ftree" or plot.type = "ftree": Risk information is plotted in a frequency tree. See plot_tree for further options.
3. plot.type = "icons" or plot.type = "iconarray": The underlying population is plotted as icons. See plot_icons for further options.
4. plot.type = "mosaic" or plot.type = "mosaicplot": Risk information is plotted as a mosaicplot. See plot_mosaic for further options.
5. plot.type = "curve" or plot.type = "curves": Draws curves of selected values (including PPV, NPV) See plot_curve for further options.

6. `plot.type = "plane"` or `plot.type = "planes"`: Draws a 3D-plane of selected values (e.g., predictive values PPV or NPV) See `plot_plane` for further options.

`...` Additional parameters to be passed to the underlying plotting functions.

Details

`plot.riskyr` also uses the text settings specified in the "riskyr" object.

See Also

Other visualization functions: `plot_curve`, `plot_fnet`, `plot_icons`, `plot_mosaic`, `plot_plane`, `plot_tree`

Examples

```
# Select a scenario from list of scenarios:
s25 <- scenarios$n25  # select scenario 25 from scenarios

# Plot different types:
plot(s25)  # => default plot (fnet)
plot(s25, plot.type = "fnet")  # => network diagram (same as default)
plot(s25, plot.type = "tree", area = "vr") # => tree diagram (with vertical rectangles)
plot(s25, plot.type = "curve", what = "all")
plot(s25, plot.type = "icons")
plot(s25, plot.type = "icons", type = "mosaic")  # passing on additional parameter to create.
plot(s25, plot.type = "mosaic")
plot(s25, plot.type = "plane", what = "NPV")
```

---

plot_curve *Plot curves of selected values (e.g., PPV or NPV) as a function of prevalence.*

Description

`plot_curve` draws curves of selected values (including PPV, NPV) as a function of the prevalence (`prev`) for given values of sensitivity `sens` (or miss rate `mirt`) and specificity `spec` (or false alarm rate `fart`).

Usage

```
plot_curve(prev = num$prev, sens = num$sens, mirt = NA, spec = num$spec,
  fart = NA, what = c("prev", "PPV", "NPV"), what.col = pal,
  show.points = TRUE, log.scale = FALSE, title.lbl = txt$scen.lbl,
  cex.lbl = 0.85)
```

Arguments

prev	The condition's prevalence prev (i.e., the probability of condition being TRUE).
sens	The decision's sensitivity sens (i.e., the conditional probability of a positive decision provided that the condition is TRUE). sens is optional when its complement mirt is provided.
mirt	The decision's miss rate mirt (i.e., the conditional probability of a negative decision provided that the condition is TRUE). mirt is optional when its complement sens is provided.
spec	The decision's specificity spec (i.e., the conditional probability of a negative decision provided that the condition is FALSE). spec is optional when its complement fart is provided.
fart	The decision's false alarm rate fart (i.e., the conditional probability of a positive decision provided that the condition is FALSE). fart is optional when its complement spec is provided.
what	A vector of character codes that specify the selection of curves to be plotted. Currently available options are c("prev", "PPV", "NPV", "ppod", "acc") (shortcut: what = "all"). Default: what = c("prev", "PPV", "NPV").
what.col	A vector of colors corresponding to the elements specified in what. Default: what.col = pal.
show.points	Boolean option for showing the point of intersection with the current prevalence prev in all selected curves. Default: show.points = TRUE.
log.scale	Boolean value for switching from a linear to a logarithmic x-axis. Default: log.scale = FALSE.
title.lbl	The title of the current plot. Default: title.lbl = txt$scen.lbl.
cex.lbl	Scaling factor for the size of text labels (e.g., on axes, legend, margin text). Default: cex.lbl = .85.

Details

plot_curve is a generalization of plot_PV (see legacy code) that allows for additional dependent values.

See Also

comp_prob computes current probability information; prob contains current probability information; comp_freq computes current frequency information; freq contains current frequency information; num for basic numeric parameters; txt for current text settings; pal for current color settings.

Other visualization functions: plot.riskyr, plot_fnet, plot_icons, plot_mosaic, plot_plane, plot_tree

Examples

```
# Basics:
plot_curve()                    # => default: what = ("prev", "PPV", "NPV")
```

```
plot_curve(show.points = FALSE)  # => default without points

# all curves:
plot_curve(what = "all") # => all curves: what = ("prev", "PPV", "NPV", "ppod", "acc")
plot_curve(what = "all", show.points = FALSE)  # => all curves, no points

# selected curves:
plot_curve(what = c("PPV", "NPV"))                  # => PPV and NPV
plot_curve(what = c("prev", "PPV", "NPV", "acc"))   # => prev, PPV, NPV, and acc
plot_curve(what = c("prev", "PPV", "NPV", "ppod"))  # => prev, PPV, NPV, and acc

# X-axis as linear vs. log scale:
plot_curve(prev = .01, sens = .9, spec = .8)                    # => linear scale
plot_curve(prev = .01, sens = .9, spec = .8, log.scale = TRUE)   # => log scale

plot_curve(prev = .0001, sens = .7, spec = .6)                  # => linear scale
plot_curve(prev = .0001, sens = .7, spec = .6, log.scale = TRUE) # => log scale

# Other options:
plot_curve(title.lbl = "Testing smaller text labels", cex.lbl = .60)
plot_curve(what = "all", what.col = c("grey", "red3", "green3", "blue3", "gold"))
```

---

plot_fnet *Plot a network diagram of frequencies and probabilities.*

Description

plot_fnet draws a network diagram of frequencies (as nodes) and probabilities (as edges) from a sufficient and valid set of 3 essential probabilities (prev, and sens or its complement mirt, and spec or its complement fart) or existing frequency information freq and a population size of N individuals.

Usage

```
plot_fnet(prev = num$prev, sens = num$sens, mirt = NA, spec = num$spec,
  fart = NA, N = freq$N, round = TRUE, by = "cddc", area = "sq",
  p.lbl = "num", show.accu = TRUE, w.acc = 0.5,
  title.lbl = txt$scen.lbl, popu.lbl = txt$popu.lbl,
  cond.true.lbl = txt$cond.true.lbl, cond.false.lbl = txt$cond.false.lbl,
  dec.pos.lbl = txt$dec.pos.lbl, dec.neg.lbl = txt$dec.neg.lbl,
  hi.lbl = txt$hi.lbl, mi.lbl = txt$mi.lbl, fa.lbl = txt$fa.lbl,
  cr.lbl = txt$cr.lbl, col.txt = grey(0.01, alpha = 0.99), box.cex = 0.85,
  col.boxes = pal, col.border = grey(0.33, alpha = 0.99), lwd = 1.5,
  box.lwd = 1.5, col.shadow = grey(0.11, alpha = 0.99), cex.shadow = 0)
```

Arguments

prev	The condition's prevalence prev (i.e., the probability of condition being TRUE).
sens	The decision's sensitivity sens (i.e., the conditional probability of a positive decision provided that the condition is TRUE). sens is optional when its complement mirt is provided.
mirt	The decision's miss rate mirt (i.e., the conditional probability of a negative decision provided that the condition is TRUE). mirt is optional when its complement sens is provided.
spec	The decision's specificity value spec (i.e., the conditional probability of a negative decision provided that the condition is FALSE). spec is optional when its complement fart is provided.
fart	The decision's false alarm rate fart (i.e., the conditional probability of a positive decision provided that the condition is FALSE). fart is optional when its complement spec is provided.
N	The number of individuals in the population. A suitable value of N is computed, if not provided.
round	A Boolean option specifying whether computed frequencies are rounded to integers. Default: round = TRUE.
by	A character code specifying the perspective (or 1st category by which the population is split into subsets) with 4 options: 1. "cd" ... by condition; 2. "dc" ... by decision; 3. "cddc" ... 1st by condition, 2nd by decision; 4. "dccd" ... 1st by decision, 2nd by condition.
area	A character code specifying the area of the boxes (or their relative sizes) with 4 options: 1. "no" ... all boxes are shown with the same size; 2. "sq" ... boxes are squares with area sizes scaled proportional to frequencies (default); 3. "hr" ... boxes are horizontal rectangles with area sizes scaled proportional to frequencies; 4. "vr" ... boxes are vertical rectangles with area sizes scaled proportional to frequencies.
p.lbl	A character code specifying the type of probability information (on edges) with 4 options: 1. "nam" ... names of probabilities; 2. "num" ... numeric values of probabilities (rounded to 3 decimals) (default); 3. "mix" ... names of essential probabilities, values of complements; 4. "min" ... minimal labels: names of essential probabilities.
show.accu	Option for showing current accuracy metrics accu in the plot. Default: show.accu = TRUE.
w.acc	Weigthing parameter w used to compute weighted accuracy w.acc in comp_accu. Default: w.acc = .50.
	Various other options allow the customization of text labels and colors:

title.lbl	Text label for current plot title. Default: title.lbl = txt$scen.lbl.
popu.lbl	Text label for current population popu.
cond.true.lbl	Text label for current cases of cond.true.
cond.false.lbl	Text label for current cases of cond.false.
dec.pos.lbl	Text label for current cases of dec.pos.
dec.neg.lbl	Text label for current cases of dec.neg.
hi.lbl	Text label for hits hi.
mi.lbl	Text label for misses mi.
fa.lbl	Text label for false alarms fa.
cr.lbl	Text label for correct rejections cr.
col.txt	Color for text labels (in boxes).
box.cex	Scaling factor for text (in boxes). Default: box.cex = .90.
col.boxes	Colors of boxes (a single color or a vector with named colors matching the number of current boxes). Default: Current color information contained in pal.
col.border	Color of borders. Default: col.border = grey(.33, alpha = .99).
lwd	Width of arrows.
box.lwd	Width of boxes.
col.shadow	Color of box shadows. Default: col.shadow = grey(.11, alpha = .99).
cex.shadow	Scaling factor of shadows (values > 0 showing shadows). Default: cex.shadow = 0.

Details

plot_fnet is a generalization of plot_tree and offers the additional option of plotting the interplay between the 9 frequencies of freq and and the 10 probabilities of prob in a single network diagram.

The option by (as 2 or 4 characters) allows specifying 4 different ways of arranging frequencies:

1. "cd" plots a tree diagram in which the population is split by condition;
2. "dc" plots a tree diagram in which the population is split by decision;
3. "cddc" plots a network diagram in which the population is split 1st by condition, 2nd by decision (default);
4. "dccd" is yet to be implemented.

The option area (as 2 characters) allows specifying 4 different box shapes and sizes:

1. "no" shows all boxes in the same size (default);
2. "sq" shows boxes as squares with area sizes proportional to frequencies;
3. "hr" shows boxes as horizontal rectangles of area sizes proportional to frequencies;
4. "vr" shows boxes as vertical rectangles of area sizes proportional to frequencies. The resulting shapes and their relative proportions correspond to the areas in plot_mosaic.

If a prevalence value prev is provided, a new list of natural frequencies freq is computed by comp_freq. By contrast, if no prevalence value prev is provided, the values currently contained in freq are used. By default, comp_freq rounds frequencies to nearest integers to avoid decimal values in freq.

plot_fnet requires and uses the R package "diagram" (library("diagram")).

Value

Nothing (NULL).

See Also

num contains basic numeric parameters; init_num initializes basic numeric parameters; freq contains current frequency information; comp_freq computes current frequency information; prob contains current probability information; comp_prob computes current probability information; pal contains current color settings; txt contains current text settings; comp_min_N computes a suitable minimum population size N.

Other visualization functions: plot.riskyr, plot_curve, plot_icons, plot_mosaic, plot_plane, plot_tree

Examples

```
# Plotting existing freq:
plot_fnet()  # => plot current freq with default options
plot_fnet(by = "dccd")
plot_fnet(area = "no")
plot_fnet(p.lbl = "num")
plot_fnet(title.lbl = "")
plot_fnet(N = 33)
plot_fnet(N = NA)

# Computing and plotting new frequencies from probabilities:
plot_fnet(prev = 1/3)  # => changes prev, but uses current defaults of sens and spec
plot_fnet(prev = 1/3, N = 55)
plot_fnet(prev = 1/3, N = NA)
plot_fnet(prev = 1/3, round = FALSE)
plot_fnet(prev = .10, sens = .90, spec = 1/3, N = 100)
plot_fnet(prev = .10, sens = .90, spec = NA, fart = 1/3, N = 33)
plot_fnet(prev = .10, sens = .90, spec = 1/3, fart = NA, N = NA)
plot_fnet(prev = .10, sens = .90, spec = NA, fart = 1/3, N = NA)

# Perspective options:
plot_fnet(by = "cd")    # => 1. Tree diagram (by condition)
plot_fnet(by = "dc")    # => 2. Tree diagram (by decision)
plot_fnet(by = "cddc")  # => 3. Network diagram (1st by cond, 2nd by dec) (default)
plot_fnet(by = "dccd")  # => 4. Network diagram (1st by dec, 2nd by cond)

# Area options:
plot_fnet(area = "sq")  # => (default)
plot_fnet(area = "no")
plot_fnet(area = "sq", round = FALSE)
plot_fnet(area = "hr")
plot_fnet(area = "vr", round = FALSE)

# Accuracy:
plot_fnet(show.accu = TRUE)               # => default w = .5 (balanced accuracy "bacc")
plot_fnet(show.accu = TRUE, w.acc = 1/3)  # => (weighted accuracy "wacc")
plot_fnet(show.accu = FALSE)              # => no accuracy info.
```

```
# Rounding:
plot_fnet(prev = .1, sens = .7, spec = .9, N = 10, by = "cddc", area = "sq",
          p.lbl = "num", round = TRUE)    # => mi = 0
plot_fnet(prev = .1, sens = .7, spec = .9, N = 10, by = "cddc", area = "sq",
          p.lbl = "num", round = FALSE)   # => mi = 0.3

# Combining perspectives, areas, and label options:
plot_fnet(by = "cd", area = "sq", p.lbl = "nam")  # => by cond + sq + prob names
plot_fnet(by = "cd", area = "hr", p.lbl = "num")  # => by cond + hr + prob numbers
plot_fnet(by = "dc", area = "sq", p.lbl = "num")  # => by dec  + sq + mix names and numbers
plot_fnet(by = "dc", area = "vr", p.lbl = "mix")  # => by dec  + vr + min. labels

# Custom colors and shadows:
plot_fnet(prev = .08, sens = .92, spec = .95, N = 10000, area = "hr")
plot_fnet(area = "sq", col.boxes = "gold", col.border = "steelblue4",
          col.shadow = "steelblue4", cex.shadow = .008)
plot_fnet(N = NA, area = "vr", col.txt = "steelblue4", col.boxes = "lightyellow",
          col.border = grey(.3, .7), cex.shadow = .008, col.shadow = grey(.1, .9))
```

`plot_icons` *Plot an icon array of a population.*

Description

`plot_icons` plots a population of which individual's condition has been classified correctly or incorrectly as icons from a sufficient and valid set of 3 essential probabilities (`prev`, and `sens` or its complement `mirt`, and `spec` or its complement `fart`) or existing frequency information `freq` and a population size of `N` individuals.

Usage

```
plot_icons(prev = num$prev, sens = num$sens, mirt = NA, spec = num$spec,
  fart = NA, N = freq$N, type = "array", ident.order = c("hi", "mi",
  "fa", "cr"), icon.colors = pal[c("hi", "mi", "fa", "cr")],
  icon.types = 22, icon.border.col = grey(0.1, 0.5),
  icon.border.lwd = 1.5, transparency = 0.5, icon.size = NULL,
  block.d = NULL, border.d = 0.1, block.size.row = 10,
  block.size.col = 10, nblocks.row = NULL, nblocks.col = NULL,
  fill.array = "left", fill.blocks = "rowwise", show.accu = TRUE,
  w.acc = 0.5, title.lbl = txt$scen.lbl, type.lbls = txt[c("hi.lbl",
  "mi.lbl", "fa.lbl", "cr.lbl")], cex.lbl = 0.85)
```

Arguments

`prev`	The condition's prevalence `prev` (i.e., the probability of condition being `TRUE`).

`sens` The decision's sensitivity `sens` (i.e., the conditional probability of a positive decision provided that the condition is `TRUE`). `sens` is optional when its complement `mirt` is provided.

`mirt` The decision's miss rate `mirt` (i.e., the conditional probability of a negative decision provided that the condition is `TRUE`). `mirt` is optional when its complement `sens` is provided.

`spec` The decision's specificity value `spec` (i.e., the conditional probability of a negative decision provided that the condition is `FALSE`). `spec` is optional when its complement `fart` is provided.

`fart` The decision's false alarm rate `fart` (i.e., the conditional probability of a positive decision provided that the condition is `FALSE`). `fart` is optional when its complement `spec` is provided.

`N` The number of individuals in the population. A suitable value of `N` is computed, if not provided. If N is 100,000 or greater it is reduced to 10,000 for the array types if the frequencies allow it.

`type` The icons can be arranged in different ways resulting in different types of displays:

1. `type = "array"`: Icons are plotted in a classical icon array (default). Icons can be arranged in blocks using `block.d`. The order of filling the array can be customized using `fill.array` and `fill.blocks`.
2. `type = "shuffledarray"`: Icons are plotted in an icon array, but positions are shuffled (randomized). Icons can be arranged in blocks using `block.d`. The order of filling the array can be customized using `fill.array` and `fill.blocks`.
3. `type = "mosaic"`: Icons are ordered like in a mosaic plot. The area size displays the relative proportions of their frequencies.
4. `type = "fillequal"`: Icons are positioned into equally sized blocks. Thus, their density reflects the relative proportions of their frequencies.
5. `type = "fillleft"`: Icons are randomly filled from the left.
6. `type = "filltop"`: Icons are randomly filled from the top.
7. `type = "scatter"`: Icons are randomly scattered into the plot.

`ident.order` The order in which icon identities (i.e, hi, mi, fa, and cr) are plotted. Default: `ident.order = c("hi", "mi", "fa", "cr")`

`icon.colors` Specifies the icon colors as a vector.

`icon.types` Specifies the appearance of the icons as a vector. Accepts values from 1 to 25 (see `?points`).

`icon.border.col`
Specifies the border color of icons (if applicable).

`icon.border.lwd`
Specifies the border width of icons (if applicable).

`transparency` Specifies the transparency for overlapping icons (not `type` "array" and "shuffledarray").

`icon.size` Manually specifies the size of the icons via `cex` (calculated dynamically by default.)

block.d	The distance between blocks (does not apply to "filleft", "filltop", and "scatter")
border.d	The distance of icons to the border.
	Additional options allow to control the arrangement of the arrays (type "array" and "shuffledarray"):
block.size.row	specifies how many icons should be in each block row.
block.size.col	specifies how many icons should be in each block column.
nblocks.row	specifies how many blocks there are in each row. Is calculated by default.
nblocks.col	specifies how many blocks are there in each column. Is calculated by default.
fill.array	specifies how the blocks are filled into the array (Options "left" (default) and "top").
fill.blocks	specifies how icons within blocks are filled (Options: fill.blocks = "rowwise" (default) and fill.blocks = "colwise")
show.accu	Option for showing current accuracy metrics accu in the plot. Default: show.accu = TRUE.
w.acc	Weigthing parameter w used to compute weighted accuracy w.acc in comp_accu. Default: w.acc = .50.
	Various other options allow the customization of text labels and colors:
title.lbl	Text label to set plot title.
type.lbls	Text labels for icon types to be displayed in legend.
cex.lbl	Scaling factor for the size of text labels (e.g., on axes, legend, margin text). Default: cex.lbl = .85.

Details

If probabilities are provided, a new list of natural frequencies freq is computed by comp_freq. By contrast, if no probabilities are provided, the values currently contained in freq are used. By default, comp_freq rounds frequencies to nearest integers to avoid decimal values in freq.

See Also

Other visualization functions: plot.riskyr, plot_curve, plot_fnet, plot_mosaic, plot_plane, plot_tree

Examples

```
# ways to work:
plot_icons()  # => plots icon array for default population (with default type = "array")
plot_icons(type = "shuffledarray")  # => icon array with shuffled IDs

plot_icons(type = "mosaic",    N = 1000)  # => areas as in mosaic plot
plot_icons(type = "fillequal", N = 1000)  # => areas of equal size (density reflects probability)
plot_icons(type = "fillleft",  N = 1000)  # => icons filled from left to right (in columns)
plot_icons(type = "filltop",   N = 1000)  # => icons filled from top to bottom (in rows)

plot_icons(icon.types = c(21,23,24,23),
              block.size.row = 5, block.size.col = 5, #nblocks.row = 2, nblocks.col = 2,
               block.d = 0.5, border.d = 0.9)
```

```
plot_icons(type = "scatter",   N = 1000)  # => icons randomly scattered.

# some variants:
plot_icons(N = 800, type = "array", icon.types = c(21,22,23,24),
           block.d = 0.5, border.d = 0.5)

plot_icons(N = 1250, sens = 0.9, spec = 0.9, prev = 0.9,
               icon.types = c(21,23,24,23),
               block.size.row = 10, block.size.col = 5,
               nblocks.row = 5, nblocks.col = 5,
               block.d = 0.8,
               border.d = 0.2,
               fill.array = "top")
plot_icons(N = 800, type = "shuffledarray", icon.types = c(21,23,24,22),
           block.d = 0.5, border.d = 0.5)

plot_icons(N = 800, type = "shuffledarray", icon.types = c(21,23,24,22),
           icon.border.col = grey(.33, .99), icon.border.lwd = 3)

plot_icons(N = 800, type = "fillequal", icon.types = c(21,22,22,21),
           icon.border.lwd = .5, cex = 3)
```

`plot_mosaic` *Plot a mosaic plot of population frequencies.*

Description

`plot_mosaic` draws a mosaic plot that represents the proportions of frequencies in the current population popu as relatives sizes of rectangular areas.

Usage

```
plot_mosaic(prev = num$prev, sens = num$sens, mirt = NA,
  spec = num$spec, fart = NA, N = num$N, vsplit = TRUE,
  show.accu = TRUE, w.acc = 0.5, title.lbl = txt$scen.lbl,
  col.sdt = c(pal["hi"], pal["mi"], pal["fa"], pal["cr"]))
```

Arguments

`prev`	The condition's prevalence `prev` (i.e., the probability of condition being `TRUE`).
`sens`	The decision's sensitivity `sens` (i.e., the conditional probability of a positive decision provided that the condition is `TRUE`). `sens` is optional when its complement `mirt` is provided.
`mirt`	The decision's miss rate `mirt` (i.e., the conditional probability of a negative decision provided that the condition is `TRUE`). `mirt` is optional when its complement `sens` is provided.

spec	The decision's specificity value spec (i.e., the conditional probability of a negative decision provided that the condition is FALSE). spec is optional when its complement fart is provided.
fart	The decision's false alarm rate fart (i.e., the conditional probability of a positive decision provided that the condition is FALSE). fart is optional when its complement spec is provided.
N	The number of individuals in the population. (This value is not represented in the plot, but used when new frequency information freq and a new population table popu are computed from scratch from current probabilities.)
vsplit	Option for toggling between vertical and horizontal split. Default: vsplit = TRUE.
show.accu	Option for showing current accuracy metrics accu in the plot. Default: show.accu = TRUE.
w.acc	Weigthing parameter w used to compute weighted accuracy w.acc in comp_accu. Default: w.acc = .50.
title.lbl	Text label for current plot title. Default: title.lbl = txt$scen.lbl.
col.sdt	Colors for cases of 4 essential frequencies. Default: col.sdt = c(pal["hi"], pal["mi"], pal["fa"]

Details

If a sufficient and valid set of 3 essential probabilities (prev, and sens or its complement mirt, and spec or its complement fart) is provided, new frequency information freq and a new population table popu are computed from scratch. Otherwise, the existing population popu is shown.

Rectangles corresponding to the areas of the mosaic plot can be visualized byopting for vertical rectangles (by selecting the option box = "vr") in plot_tree and plot_fnet.

plot_mosaic requires and uses the R packages "vcd" and "grid" (library("vcd", "grid")).

See Also

comp_popu computes the current population; popu contains the current population; comp_freq computes current frequency information; freq contains current frequency information; num for basic numeric parameters; txt for current text settings; pal for current color settings

Other visualization functions: plot.riskyr, plot_curve, plot_fnet, plot_icons, plot_plane, plot_tree

Examples

```
plot_mosaic()               # => default options
plot_mosaic(title.lbl = "")  # => no title
plot_mosaic(vsplit = FALSE)  # => horizontal split
plot_mosaic(title.lbl = "My favorite scenario", col.sdt = "goldenrod")

# Accuracy:
plot_mosaic(show.accu = TRUE)              # => default w = .5 (balanced accuracy "bacc")
plot_mosaic(show.accu = TRUE, w.acc = 1/3)  # => (weighted accuracy "wacc")
plot_mosaic(show.accu = FALSE)              # => no accuracy info.
```

plot_plane *Plot a plane of selected values (e.g., PPV or NPV) as a function of sensitivity and specificity.*

Description

plot_plane draws a 3D-plane of selected values (e.g., predictive values PPV or NPV) as a function of a decision's sensitivity sens and specificity value spec for a given prevalence (prev).

Usage

```
plot_plane(prev = num$prev, sens = num$sens, mirt = NA, spec = num$spec,
  fart = NA, what = "PPV", what.col = pal, show.point = TRUE,
  step.size = 0.05, theta = -45, phi = 0, title.lbl = txt$scen.lbl,
  cex.lbl = 0.85)
```

Arguments

prev	The condition's prevalence prev (i.e., the probability of condition being TRUE).
sens	The decision's sensitivity sens (i.e., the conditional probability of a positive decision provided that the condition is TRUE). sens is optional when its complement mirt is provided.
mirt	The decision's miss rate mirt (i.e., the conditional probability of a negative decision provided that the condition is TRUE). mirt is optional when its complement sens is provided.
spec	The decision's specificity value spec (i.e., the conditional probability of a negative decision provided that the condition is FALSE). spec is optional when its complement fart is provided.
fart	The decision's false alarm rate fart (i.e., the conditional probability of a positive decision provided that the condition is FALSE). fart is optional when its complement spec is provided.
what	A character code that specifies one metric to be plotted as a plane. Currently available options are c("PPV", "NPV", "ppod", "acc"). Default: what = "PPV".
what.col	A color corresponding to the metric specified in what. Default: what.col = pal.
show.point	Boolean option for showing the current value of the selected metric for the current conditions (prev, sens, spec) as a point on the plane. Default: show.point = TRUE.
step.size	Sets the granularity of the sens-by-spec grid. Default: step.size = .05.
theta	Horizontal rotation angle (used by persp). Default: theta = -45.
phi	Vertical rotation angle (used by persp). Default: phi = 0.
title.lbl	The title of the current plot. Default: title.lbl = txt$scen.lbl.
cex.lbl	Scaling factor for the size of text labels (e.g., on axes, legend, margin text). Default: cex.lbl = .85.

Details

plot_plane is a generalization of plot_PV3d (see legacy code) that allows for additional dependent values.

See Also

comp_popu computes the current population; popu contains the current population; comp_freq computes current frequency information; freq contains current frequency information; num for basic numeric parameters; txt for current text settings; pal for current color settings

Other visualization functions: plot.riskyr, plot_curve, plot_fnet, plot_icons, plot_mosaic, plot_tree

Examples

```
# Basics:
plot_plane()  # => current defaults (what = "PPV")
plot_plane(what = "PPV")  # => plane of PPV
plot_plane(what = "NPV")  # => plane of NPV
plot_plane(what = "ppod") # => plane of ppod
plot_plane(what = "acc")  # => plane of acc

# Options:
plot_plane(title.lbl = "Testing smaller text labels", cex.lbl = .60)
plot_plane(show.point = FALSE)  # => no point shown on plane
plot_plane(step.size = .333, what.col = "firebrick")  # => coarser granularity + color
plot_plane(step.size = .025, what.col = "chartreuse4")  # => finer granularity + color
plot_plane(what.col = "steelblue4", theta = -90, phi = 45)  # => rotated, from above
```

---

plot_tree *Plot a tree diagram of frequencies and probabilities.*

Description

plot_tree draws a tree diagram of frequencies (as nodes) and probabilities (as edges) from a sufficient and valid set of 3 essential probabilities (prev, and sens or its complement mirt, and spec or its complement fart) or existing frequency information freq and a population size of N individuals.

Usage

```
plot_tree(prev = num$prev, sens = num$sens, mirt = NA, spec = num$spec,
  fart = NA, N = freq$N, round = TRUE, by = "cd", area = "no",
  p.lbl = "mix", show.accu = TRUE, w.acc = 0.5,
  title.lbl = txt$scen.lbl, popu.lbl = txt$popu.lbl,
  cond.true.lbl = txt$cond.true.lbl, cond.false.lbl = txt$cond.false.lbl,
  dec.pos.lbl = txt$dec.pos.lbl, dec.neg.lbl = txt$dec.neg.lbl,
```

```
hi.lbl = txt$hi.lbl, mi.lbl = txt$mi.lbl, fa.lbl = txt$fa.lbl,
cr.lbl = txt$cr.lbl, col.txt = grey(0.01, alpha = 0.99), box.cex = 0.9,
col.boxes = pal, col.border = grey(0.33, alpha = 0.99), lwd = 1.6,
box.lwd = 1.8, col.shadow = grey(0.11, alpha = 0.99), cex.shadow = 0)
```

Arguments

prev	The condition's prevalence prev (i.e., the probability of condition being TRUE).
sens	The decision's sensitivity sens (i.e., the conditional probability of a positive decision provided that the condition is TRUE). sens is optional when its complement mirt is provided.
mirt	The decision's miss rate mirt (i.e., the conditional probability of a negative decision provided that the condition is TRUE). mirt is optional when its complement sens is provided.
spec	The decision's specificity value spec (i.e., the conditional probability of a negative decision provided that the condition is FALSE). spec is optional when its complement fart is provided.
fart	The decision's false alarm rate fart (i.e., the conditional probability of a positive decision provided that the condition is FALSE). fart is optional when its complement spec is provided.
N	The number of individuals in the population. A suitable value of N is computed, if not provided.
round	A Boolean option specifying whether computed frequencies are rounded to integers. Default: round = TRUE.
by	A character code specifying the perspective (or category by which the population is split into subsets) with 2 options: 1. "cd" ... by condition; 2. "dc" ... by decision.
area	A character code specifying the area of the boxes (or their relative sizes) with 4 options: 1. "no" ... all boxes are shown with the same size (default); 2. "sq" ... boxes are squares with area sizes scaled proportional to frequencies; 3. "hr" ... boxes are horizontal rectangles with area sizes scaled proportional to frequencies; 4. "vr" ... boxes are vertical rectangles with area sizes scaled proportional to frequencies.
p.lbl	A character code specifying the type of probability information (on edges) with 4 options: 1. "nam" ... names of probabilities; 2. "num" ... numeric values of probabilities (rounded to 3 decimals); 3. "mix" ... names of essential probabilities, values of complements (default); 4. "min" ... minimal labels: names of essential probabilities.
show.accu	Option for showing current accuracy metrics accu in the plot. Default: show.accu = TRUE.

w.acc	Weigthing parameter w used to compute weighted accuracy w.acc in comp_accu. Default: w.acc = .50.
	Various other options allow the customization of text labels and colors:
title.lbl	Text label for current plot title. Default: title.lbl = txt$scen.lbl.
popu.lbl	Text label for current population popu.
cond.true.lbl	Text label for current cases of cond.true.
cond.false.lbl	Text label for current cases of cond.false.
dec.pos.lbl	Text label for current cases of dec.pos.
dec.neg.lbl	Text label for current cases of dec.neg.
hi.lbl	Text label for hits hi.
mi.lbl	Text label for misses mi.
fa.lbl	Text label for false alarms fa.
cr.lbl	Text label for correct rejections cr.
col.txt	Color for text labels (in boxes).
box.cex	Scaling factor for text (in boxes). Default: box.cex = .90.
col.boxes	Colors of boxes (a single color or a vector with named colors matching the number of current boxes). Default: Current color information contained in pal.
col.border	Color of borders. Default: col.border = grey(.33, alpha = .99).
lwd	Width of arrows.
box.lwd	Width of boxes.
col.shadow	Color of box shadows. Default: col.shadow = grey(.11, alpha = .99).
cex.shadow	Scaling factor of shadows (values > 0 showing shadows). Default: cex.shadow = 0.

Details

The option area (as 2 characters) allows specifying 4 different box shapes and sizes:

1. "no" shows all boxes in the same size (default);
2. "sq" shows boxes as squares with area sizes proportional to frequencies;
3. "hr" shows boxes as horizontal rectangles of area sizes proportional to frequencies;
4. "vr" shows boxes as vertical rectangles of area sizes proportional to frequencies. The resulting shapes and their relative proportions correspond to the areas in plot_mosaic.

If a prevalence value prev is provided, a new list of natural frequencies freq is computed by comp_freq. By contrast, if no prevalence value prev is provided, the values currently contained in freq are used. By default, comp_freq rounds frequencies to nearest integers to avoid decimal values in freq.

plot_tree requires and uses the R package "diagram" (library("diagram")).

Value

Nothing (NULL).

See Also

num contains basic numeric parameters; init_num initializes basic numeric parameters; freq contains current frequency information; comp_freq computes current frequency information; prob contains current probability information; comp_prob computes current probability information; pal contains current color settings; txt contains current text settings; comp_min_N computes a suitable minimum population size N.

Other visualization functions: plot.riskyr, plot_curve, plot_fnet, plot_icons, plot_mosaic, plot_plane

Examples

```
# Plotting existing freq:
plot_tree()
plot_tree(by = "dc")
plot_tree(area = "sq")
plot_tree(p.lbl = "num")
plot_tree(title.lbl = "")
plot_tree(N = 33)
plot_tree(N = NA)

# Computing and plotting new frequencies:
plot_tree(prev = 1/3)
plot_tree(prev = 1/3, N = 55)
plot_tree(prev = 1/3, N = NA)
plot_tree(prev = 1/3, round = FALSE)
plot_tree(prev = .10, sens = .90, spec = 1/3, N = 100)
plot_tree(prev = .10, sens = .90, spec = NA, fart = 1/3, N = 33)
plot_tree(prev = .10, sens = .90, spec = 1/3, fart = NA, N = NA)
plot_tree(prev = .10, sens = .90, spec = NA, fart = 1/3, N = NA)

# Area options:
plot_tree(area = "sq")
plot_tree(area = "sq", round = FALSE)
plot_tree(area = "hr")
plot_tree(area = "vr", round = FALSE)

# Accuracy:
plot_tree(show.accu = TRUE)               # => default w = .5 (balanced accuracy "bacc")
plot_tree(show.accu = TRUE, w.acc = 1/3)  # => (weighted accuracy "wacc")
plot_tree(show.accu = FALSE)              # => no accuracy info.

# Perspectives, areas, and label options:
plot_tree(by = "cd", area = "sq", p.lbl = "nam")  # => by cond + sq + prob names
plot_tree(by = "cd", area = "hr", p.lbl = "num")  # => by cond + hr + prob numbers
plot_tree(by = "dc", area = "sq", p.lbl = "num")  # => by dec  + sq + names and numbers
plot_tree(by = "dc", area = "vr", p.lbl = "mix")  # => by dec  + vr + min. labels

# Custom colors and shadows:
plot_tree(prev = .08, sens = .92, spec = .95, N = 10000, area = "hr")
plot_tree(area = "sq", col.boxes = "gold", col.border = "steelblue4",
          col.shadow = "steelblue4", cex.shadow = .008)
```

```
plot_tree(N = NA, area = "vr", col.txt = "steelblue4", col.boxes = "lightyellow",
          col.border = grey(.3, .7), cex.shadow = .008, col.shadow = grey(.1, .9))
```

---

popu *A population table based on current frequencies.*

Description

popu is an R data frame that is computed by comp_popu from the current frequency information (contained in freq). Each individual is represented as a row; columns represent the individual's condition (TRUE or FALSE), a corresponding decision (also encoded as TRUE = positive or FALSE = negative), and its classification (in SDT terms) as either true positive (an individual hit hi), false negative (an individual miss mi), false positive (an individual false alarm fa), or true negative (an individual correct rejection cr).

Usage

```
popu
```

Format

An object of class NULL of length 0.

Details

#' popu is initialized to NULL and needs to be computed by calling comp_popu with current parameter settings.

comp_popu uses the current text information contained in txt to define the labels of conditions, decisions, and SDT classifications.

A visualization of the current population popu is provided by plot_icons.

Value

A data frame popu containing N rows (individual cases) and 3 columns ("Truth", "Decision", "SDT") encoded as ordered factors (with 2, 2, and 4 levels, respectively).

See Also

The corresponding generating function comp_popu; num for basic numeric parameters; freq for current frequency information; txt for current text settings.

Examples

```
popu <- comp_popu()  # => initializes popu with current values of freq and txt
dim(popu)            # => N x 3
head(popu)           # => shows head of data frame
```

ppod *The proportion (or baseline) of a positive decision.*

Description

ppod defines the proportion (baseline probability or rate): a decision being `positive` (but not necessarily true).

Usage

```
ppod
```

Format

An object of class `numeric` of length 1.

Details

Understanding or obtaining the proportion of positive decisions ppod:

- Definition: ppod is the (non-conditional) probability:
  `ppod = p(decision = positive)`
  or the base rate (or baseline probability) of a decision being positive (but not necessarily true).
- Perspective: ppod classifies a population of `N` individuals by decision (`ppod = dec.pos/N`).
  ppod is the "by decision" counterpart to `prev` (which adopts a "by condition" perspective).
- Alternative names: base rate of positive decisions (`PR`), proportion predicted or diagnosed, rate of decision = `positive` cases
- In terms of frequencies, ppod is the ratio of `dec.pos` (i.e., `hi + fa`) divided by `N` (i.e., `hi + mi + fa + cr`):
  `ppod = dec.pos/N = (hi + fa)/(hi + mi + fa + cr)`
- Dependencies: ppod is a feature of the decision process or diagnostic procedure.
  However, the conditional probabilities `sens`, `mirt`, `spec`, `fart`, `PPV`, and `NPV` also depend on the condition's prevalence `prev`.

References

Consult Wikipedia for additional information.

See Also

`prob` contains current probability information; `comp_prob` computes current probability information; `num` contains basic numeric parameters; `init_num` initializes basic numeric parameters; `freq` contains current frequency information; `comp_freq` computes current frequency information; `is_prob` verifies probability inputs.

Other probabilities: `FDR`, `FOR`, `NPV`, `PPV`, `fart`, `mirt`, `prev`, `sens`, `spec`

Examples

```
ppod <- .50     # => sets a rate of positive decisions of 50%
ppod <- 50/100  # => (decision = TRUE) for 50 out of 100 individuals
is_prob(ppod)   # => TRUE (as ppod is a probability)
```

PPV *The positive predictive value of a decision process or diagnostic procedure.*

Description

PPV defines some decision's positive predictive value (PPV): The conditional probability of the condition being TRUE provided that the decision is positive.

Usage

```
PPV
```

Format

An object of class numeric of length 1.

Details

Understanding or obtaining the positive predictive value PPV:

- Definition: PPV is the conditional probability for the condition being TRUE given a positive decision:
  `PPV = p(condition = TRUE | decision = positive)`
  or the probability of a positive decision being correct.
- Perspective: PPV further classifies the subset of dec.pos individuals by condition (`PPV = hi/dec.pos = hi/(hi + fa)`)
- Alternative names: precision
- Relationships:
  a. PPV is the complement of the false discovery or false detection rate FDR:
  `PPV = 1 - FDR`
  b. PPV is the opposite conditional probability – but not the complement – of the sensitivity sens:
  `sens = p(decision = positive | condition = TRUE)`
- In terms of frequencies, PPV is the ratio of hi divided by dec.pos (i.e., `hi + fa`):
  `PPV = hi/dec.pos = hi/(hi + fa)`
- Dependencies: PPV is a feature of a decision process or diagnostic procedure and – similar to the sensitivity sens – a measure of correct decisions (positive decisions that are actually TRUE).
  However, due to being a conditional probability, the value of PPV is not intrinsic to the decision process, but also depends on the condition's prevalence value prev.

References

Consult Wikipedia for additional information.

See Also

comp_PPV computes PPV; prob contains current probability information; comp_prob computes current probability information; num contains basic numeric parameters; init_num initializes basic numeric parameters; comp_freq computes current frequency information; is_prob verifies probability inputs.

Other probabilities: FDR, FOR, NPV, fart, mirt, ppod, prev, sens, spec

Examples

```
PPV <- .55     # => sets a positive predictive value of 55%
PPV <- 55/100  # => (condition = TRUE) for 55 out of 100 people with (decision = positive)
is_prob(PPV)   # => TRUE (as PPV is a probability)
```

---

prev *The prevalence (baseline probability) of a condition.*

Description

prev defines a condition's prevalence value (or baseline probability): The probability of the condition being TRUE.

Usage

```
prev
```

Format

An object of class numeric of length 1.

Details

Understanding or obtaining the prevalence value prev:

- Definition: prev is the (non-conditional) probability:
  prev = p(condition = TRUE)
  or the base rate (or baseline probability) of the condition's occurrence.
- In terms of frequencies, prev is the ratio of cond.true (i.e., hi + mi) divided by N (i.e., hi + mi + fa + cr):
  prev = cond.true/N = (hi + mi)/(hi + mi + fa + cr)
- Perspective: prev classifies a population of N individuals by condition (prev = cond.true/N). prev is the "by condition" counterpart to ppod (which adopts a "by decision" perspective).

- Alternative names: base rate of condition, proportion affected, rate of condition = TRUE cases

  prev is often distinguished from the *incidence rate* (i.e., the rate of new cases within a certain time period).
- Dependencies: prev is a feature of the population and condition, but independent of the decision process or diagnostic procedure.

  The value of prev does *not* depend on features of the decision process or diagnostic procedure. However, prev must be taken into account when computing the conditional probabilities sens, mirt, spec, fart, PPV, and NPV (as they partly depend on prev).

References

Consult Wikipedia for additional information.

See Also

num contains basic numeric variables; init_num initializes basic numeric variables; is_prob verifies probability inputs; comp_prob computes derived probabilities; comp_freq computes natural frequencies from probabilities

Other probabilities: FDR, FOR, NPV, PPV, fart, mirt, ppod, sens, spec

Other essential parameters: cr, fa, hi, mi, sens, spec

Examples

```
prev <- .10     # => sets a prevalence value of 10%
prev <- 10/100  # => (condition = TRUE) for 10 out of 100 individuals
is_prob(prev)   # => TRUE (as prev is a probability)
```

---

print.summary.riskyr *Printing summarized risk information.*

Description

print.summary.riskyr provides a print method for objects of class "summary.riskyr".

Usage

```
## S3 method for class 'summary.riskyr'
print(x = NULL, ...)
```

Arguments

x	An object of class "summary.riskyr", usually a result of a call to summary.riskyr.
...	Additional parameters to be passed to the generic print function.

Format

Printed output of a "summary.riskyr" object.

Examples

```
summary(scenarios$n4)
```

---

prob *List current probability information.*

Description

prob is a list of named numeric variables containing 3 essential (1 non-conditional and 2 conditional) probabilities and 7 derived (1 non-conditional and 6 conditional) probabilities:

Usage

```
prob
```

Format

An object of class list of length 10.

Details

1. the condition's prevalence prev (i.e., the probability of the condition being TRUE): prev = cond.true/N.
2. the decision's sensitivity sens (i.e., the conditional probability of a positive decision provided that the condition is TRUE).
3. the decision's miss rate mirt (i.e., the conditional probability of a negative decision provided that the condition is TRUE).
4. the decision's specificity spec (i.e., the conditional probability of a negative decision provided that the condition is FALSE).
5. the decision's false alarm rate fart (i.e., the conditional probability of a positive decision provided that the condition is FALSE).
6. the proportion (baseline probability or rate) of the decision being positive ppod (but not necessarily true): ppod = dec.pos/N.
7. the decision's positive predictive value PPV (i.e., the conditional probability of the condition being TRUE provided that the decision is positive).
8. the decision's false detection (or false discovery) rate FDR (i.e., the conditional probability of the condition being FALSE provided that the decision is positive).
9. the decision's negative predictive value NPV (i.e., the conditional probability of the condition being FALSE provided that the decision is negative).

10. the decision's false omission rate FOR (i.e., the conditional probability of the condition being TRUE provided that the decision is negative).

These probabilities are computed from basic probabilities (contained in num) and computed by using comp_prob.

The list prob is the probability counterpart to the list containing frequency information freq.

Note that inputs of extreme probabilities (of 0 or 1) may yield unexpected values (e.g., an NPV value of NaN when is_extreme_prob_set evaluates to TRUE).

See Also

num contains basic numeric parameters; init_num initializes basic numeric parameters; txt contains current text information; init_txt initializes text information; pal contains current color information; init_pal initializes color information; freq contains current frequency information; comp_freq computes current frequency information; prob contains current probability information; comp_prob computes current probability information.

Other lists containing current scenario information: accu, freq, num, pal, txt

Examples

```
prob <- comp_prob()  # => initialize prob to default parameters
prob                 # => show current values
length(prob)         # => 8
```

---

riskyr *Create riskyr scenarios.*

Description

The instantiation function riskyr is used to create scenarios of class "riskyr", which can then be visualized by the plot method plot.riskyr and summarized by the summary method summary.riskyr.

Usage

```
riskyr(scen.lbl = "", scen.lng = txt$scen.lng, scen.txt = txt$scen.txt,
  popu.lbl = txt$popu.lbl, cond.lbl = txt$cond.lbl,
  cond.true.lbl = txt$cond.true.lbl, cond.false.lbl = txt$cond.false.lbl,
  dec.lbl = txt$dec.lbl, dec.pos.lbl = txt$dec.pos.lbl,
  dec.neg.lbl = txt$dec.neg.lbl, hi.lbl = txt$hi.lbl, mi.lbl = txt$mi.lbl,
  fa.lbl = txt$fa.lbl, cr.lbl = txt$cr.lbl, prev = num$prev,
  sens = num$sens, spec = num$spec, fart = NA, N = NA,
  scen.src = txt$scen.src, scen.apa = txt$scen.apa)
```

Arguments

scen.lbl	The current scenario title (sometimes in Title Caps).
scen.lng	Language of the current scenario (as character code). Options: "en" for English, "de" for German.
scen.txt	A longer text description of the current scenario (which may extend over several lines).
popu.lbl	A brief description of the current target population popu or sample.
cond.lbl	A name for the *condition* or feature (e.g., some disease) currently considered.
cond.true.lbl	A label for the *presence* of the current condition or cond.true cases (the condition's true state of TRUE).
cond.false.lbl	A label for the *absence* of the current condition or cond.false cases (the condition's true state of FALSE).
dec.lbl	A name for the *decision* or judgment (e.g., some diagnostic test) currently made.
dec.pos.lbl	A label for *positive* decisions or dec.pos cases (e.g., predicting the presence of the condition).
dec.neg.lbl	A label for *negative* decisions or dec.neg cases (e.g., predicting the absence of the condition).
hi.lbl	A label for *hits* or *true positives* hi (i.e., correct decisions of the presence of the condition, when the condition is actually present).
mi.lbl	A label for *misses* or *false negatives* mi (i.e., incorrect decisions of the absence of the condition when the condition is actually present).
fa.lbl	A label for *false alarms* or *false positives* fa (i.e., incorrect decisions of the presence of the condition when the condition is actually absent).
cr.lbl	A label for *correct rejections* or *true negatives* cr (i.e., a correct decision of the absence of the condition, when the condition is actually absent).
	Numeric elements:
prev	The condition's prevalence prev (i.e., the probability of condition being TRUE).
sens	The decision's sensitivity sens (i.e., the conditional probability of a positive decision provided that the condition is TRUE). sens is optional when its complement mirt is provided.
spec	The decision's specificity value spec (i.e., the conditional probability of a negative decision provided that the condition is FALSE). spec is optional when its complement fart is provided.
fart	The decision's false alarm rate fart (i.e., the conditional probability of a positive decision provided that the condition is FALSE). fart is optional when its complement spec is provided.
	Source information:
N	The number of individuals in the scenario's population. A suitable value of N is computed, if not provided.
scen.src	Source information for the current scenario.
scen.apa	Source information for the current scenario in the style of the American Psychological Association (APA style).

Format

An object of class "riskyr" with 21 entries on textual and numeric information on a riskyr scenario.

Details

Beyond basic scenario information only the population size `N` and the essential probabilities `prev`, `sens`, `spec`, and `fart` are used and returned.

Value

A list `object` of class "riskyr" containing information on a risky scenario.

Text elements (all elements of `txt`:

Examples

```
# Defining a scenario:
custom.scenario <- riskyr(scen.lbl = "Identify reoffenders",
  cond.lbl = "Being a reoffender", popu.lbl = "Prisoners",
  cond.true.lbl = "Has reoffended", cond.false.lbl = "Has not reoffended",
  dec.lbl = "Test result",
  dec.pos.lbl = "will reoffend", dec.neg.lbl = "will not reoffend",
  hi.lbl = "Reoffender found", mi.lbl = "Reoffender missed",
  fa.lbl = "False accusation", cr.lbl = "Correct release",
  prev = .45,  # prevalence of being a reoffender.
  sens = .98, spec = .46, fart = NA, N = 753,
  scen.src = "Ficticious example scenario")

# Using a scenario:
summary(custom.scenario)
plot(custom.scenario)
```

---

riskyr.guide *Opens the riskyr package guides*

Description

Opens the riskyr package guides

Usage

```
riskyr.guide()
```

scenarios *A collection of riskyr scenarios from various sources.*

Description

scenarios is a list that contains a collection of scenarios of class "riskyr" from the scientific literature and other sources that can be used directly in the visualization and summary functions.

Usage

```
scenarios
```

Format

A list with currently 26 objects of class "riskyr" (i.e., scenarios) which are each described by 21 variables:

Details

scenarios currently contains the following scenarios:

1. Mammografie 1
2. Nackenfaltentest (NFT)
3. HIV 1 (f)
4. HIV 2 (f)
5. Mammography 2
6. Sepsis
7. Cab problem
8. Sigmoidoskopie 1
9. Sigmoidoskopie 2
10. Brustkrebs 1
11. Brustkrebs 2 (BRCA1)
12. Brustkrebs 3 (BRCA1 + pos. Mam.)
13. HIV 3 (m)
14. HIV 4 (m)
15. Nackenfaltentest 2 (NFT)
16. Amniozentese (pos. NFT)
17. Musical town
18. Mushrooms
19. Bowel cancer (FOB screening)
20. PSA test 1 (high prev)

21. PSA test 2 (low prev)
22. Colorectal cancer
23. Psylicraptis screening
24. Mammography 6 (prob)
25. Mammography 6 (freq)

Variables describing each scenario:

1. `scen.lbl` Text label for current scenario.
2. `scen.lng` Language of current scenario.
3. `scen.txt` Description text of current scenario.
4. `popu.lbl` Text label for current population.
5. `cond.lbl` Text label for current condition.
6. `cond.true.lbl` Text label for `cond.true` cases.
7. `cond.false.lbl` Text label for `cond.false` cases.
8. `dec.lbl` Text label for current decision.
9. `dec.pos.lbl` Text label for `dec.pos` cases.
10. `dec.neg.lbl` Text label for `dec.neg` cases.
11. `hi.lbl` Text label for cases of hits `hi`.
12. `mi.lbl` Text label for cases of misses `mi`.
13. `fa.lbl` Text label for cases of false alarms `fa`.
14. `cr.lbl` Text label for cases of correct rejections `cr`.
15. `prev` Value of current prevalence `prev`.
16. `sens` Value of current sensitivity `sens`.
17. `spec` Value of current specificity `spec`.
18. `fart` Value of current false alarm rate `fart`.
19. `N` Current population size `N`.
20. `scen.src` Source information for current scenario.
21. `scen.apa` Source information in APA format.

Note that names of variables (columns) correspond to `init_txt` (to initialize `txt`) and `init_num` (to initialize `num`).

See columns `scen.src` and `scen.apa` for a scenario's source information.

The information of `scenarios` is also contained in an R data frame `df.scenarios` (and generated from the corresponding `.rda` file in `/data/`).

sens	*The sensitivity (or hit rate) of a decision process or diagnostic procedure.*

Description

sens defines a decision's sensitivity (or hit rate) value: The conditional probability of the decision being positive if the condition is TRUE.

Usage

```
sens
```

Format

An object of class numeric of length 1.

Details

Understanding or obtaining the sensitivity sens (or hit rate HR):

- Definition: sens is the conditional probability for a (correct) positive decision given that the condition is TRUE:
  `sens = p(decision = positive | condition = TRUE)`
  or the probability of correctly detecting true cases (`condition = TRUE`).
- Perspective: sens further classifies the subset of cond.true individuals by decision (`sens = hi/cond.true`).
- Alternative names: true positive rate (TPR), hit rate (HR), probability of detection, `power = 1 - beta`, `recall`
- Relationships:
  a. sens is the complement of the miss rate mirt (aka. false negative rate FNR or the rate of Type-II errors):
  `sens = (1 - miss rate) = (1 - FNR)`
  b. sens is the opposite conditional probability – but not the complement – of the positive predictive value PPV:
  `PPV = p(condition = TRUE | decision = positive)`
- In terms of frequencies, sens is the ratio of hi divided by cond.true (i.e., `hi + mi`):
  `sens = hi/cond.true = hi/(hi + mi)`
- Dependencies: sens is a feature of a decision process or diagnostic procedure and a measure of correct decisions (true positives).
  Due to being a conditional probability, the value of sens is not intrinsic to the decision process, but also depends on the condition's prevalence value prev.

References

Consult Wikipedia for additional information.

See Also

comp_sens computes sens as the complement of mirt; prob contains current probability information; comp_prob computes current probability information; num contains basic numeric parameters; init_num initializes basic numeric parameters; comp_freq computes current frequency information; is_prob verifies probability inputs.

Other probabilities: FDR, FOR, NPV, PPV, fart, mirt, ppod, prev, spec

Other essential parameters: cr, fa, hi, mi, prev, spec

Examples

```
sens <- .85     # => sets a sensitivity value of 85%
sens <- 85/100  # => (decision = positive) for 85 out of 100 people with (condition = TRUE)
is_prob(sens)   # => TRUE (as sens is a probability)
```

---

spec *The specificity of a decision process or diagnostic procedure.*

Description

spec defines a decision's specificity value (or correct rejection rate): The conditional probability of the decision being negative if the condition is FALSE.

Usage

```
spec
```

Format

An object of class numeric of length 1.

Details

Understanding or obtaining the specificity value spec:

- Definition: spec is the conditional probability for a (correct) negative decision given that the condition is FALSE:
  `spec = p(decision = negative | condition = FALSE)`
  or the probability of correctly detecting false cases (`condition = FALSE`).
- Perspective: spec further classifies the subset of cond.false individuals by decision (`spec = cr/cond.false`).
- Alternative names: true negative rate (TNR), correct rejection rate, `1 - alpha`
- Relationships:
  a. spec is the complement of the false alarm rate fart:
  `spec = 1 - fart`
  b. spec is the opposite conditional probability – but not the complement – of the negative predictive value NPV:
  `NPV = p(condition = FALSE | decision = negative)`

- In terms of frequencies, spec is the ratio of cr divided by cond.false (i.e., fa + cr):
  spec = cr/cond.false = cr/(fa + cr)
- Dependencies: spec is a feature of a decision process or diagnostic procedure and a measure of correct decisions (true negatives).
  However, due to being a conditional probability, the value of spec is not intrinsic to the decision process, but also depends on the condition's prevalence value prev.

References

Consult Wikipedia for additional information.

See Also

comp_spec computes spec as the complement of fart; prob contains current probability information; comp_prob computes current probability information; num contains basic numeric parameters; init_num initializes basic numeric parameters; comp_freq computes current frequency information; is_prob verifies probability inputs.

Other probabilities: FDR, FOR, NPV, PPV, fart, mirt, ppod, prev, sens

Other essential parameters: cr, fa, hi, mi, prev, sens

Examples

```
spec <- .75     # => sets a specificity value of 75%
spec <- 75/100  # => (decision = negative) for 75 out of 100 people with (condition = FALSE)
is_prob(spec)   # => TRUE (as spec is a probability)
```

---

summary.riskyr *Summarizing a riskyr scenario.*

Description

summary.riskyr provides a summary method for objects of class "riskyr".

Usage

```
## S3 method for class 'riskyr'
summary(object = NULL, summarize = "all", ...)
```

Arguments

object	An object of class "riskyr", usually a result of a call to riskyr. Inbuilt scenarios are also of type "riskyr".
summarize	What is summarized as a vector consisting of c("freq", "prob", "accu") for frequencies, probabilities, and accuracy respectively. The default "all" is an alias to all three.
...	Additional parameters to be passed to the underlying summary functions.

Format

An object of class `summary.riskyr` with up to 9 entries.

Value

A summary list `obj.sum` with up to 9 entries, dependent on which information is requested by `summarize`.

Scenario name, relevant condition , and `N` are summarized by default.

Examples

```
summary(scenarios$n4)
```

`txt` *List current values of basic text elements.*

Description

`txt` is initialized to a list of named elements to define all titles and labels corresponding to the current scenario and used throughout the `riskyr` package.

Usage

```
txt
```

Format

An object of class `list` of length 16.

Details

All textual elements that specify titles and details of the current scenario are stored as named elements (of type character) in a list `txt`. To change an element, assign a new character object to an existing name.

`txt` currently contains the following text labels:

1. `scen.lbl` The current scenario title (sometimes in Title Caps).
2. `scen.txt` A longer text description of the current scenario (which may extend over several lines).
3. `scen.src` The source information for the current scenario.
4. `scen.apa` The source information in APA format.
5. `scen.lng` The language of the current scenario (as character code). Options: `"en"`...English, `"de"`... German.
6. `popu.lbl` A brief description of the current target population popu or sample.

7. `cond.lbl` A name for the *condition* or feature (e.g., some disease) currently considered.
8. `cond.true.lbl` A label for the *presence* of the current condition or `cond.true` cases (the condition's true state of TRUE).
9. `cond.false.lbl` A label for the *absence* of the current condition or `cond.false` cases (the condition's true state of FALSE).
10. `dec.lbl` A name for the *decision* or judgment (e.g., some diagnostic test) currently made.
11. `dec.pos.lbl` A label for *positive* decisions or `dec.pos` cases (e.g., predicting the presence of the condition).
12. `dec.neg.lbl` A label for *negative* decisions or `dec.neg` cases (e.g., predicting the absence of the condition).
13. `hi.lbl` A label for *hits* or *true positives* `hi` (i.e., correct decisions of the presence of the condition, when the condition is actually present).
14. `mi.lbl` A label for *misses* or *false negatives* `mi` (i.e., incorrect decisions of the absence of the condition when the condition is actually present).
15. `fa.lbl` A label for *false alarms* or *false positives* `fa` (i.e., incorrect decisions of the presence of the condition when the condition is actually absent).
16. `cr.lbl` A label for *correct rejections* or *true negatives* `cr` (i.e., a correct decision of the absence of the condition, when the condition is actually absent).

See Also

`init_txt` initializes text information; `num` contains basic numeric parameters; `init_num` initializes basic numeric parameters; `pal` contains current color information; `init_pal` initializes color information; `freq` contains current frequency information; `comp_freq` computes current frequency information; `prob` contains current probability information; `comp_prob` computes current probability information.

Other lists containing current scenario information: `accu`, `freq`, `num`, `pal`, `prob`

Examples

```
txt           # => show  all current names and elements
txt$scen.lbl  # => show the current scenario label (e.g., used in plot titles)
txt$scen.lbl <- "My favorite example"  # => set a new scenario title
```

Index