Package 'riskyr'

August 15, 2022

Type Package

Title Rendering Risk Literacy more Transparent

Version 0.4.0

Date 2022-08-15

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

Description Risk-related information (like the prevalence of conditions, the sensitivity and specificity of diagnostic tests, or the effectiveness of interventions or treatments) can be expressed in terms of frequencies or probabilities. By providing a toolbox of corresponding metrics and representations, 'riskyr' computes, translates, and visualizes risk-related information in a variety of ways. Adopting multiple complementary perspectives provides insights into the interplay between key parameters and renders teaching and training programs on risk literacy more transparent.

Depends R (>= 3.4.0)

Imports utils (>= 3.4.0)

Suggests knitr, rmarkdown, spelling

Collate 'comp_util.R' 'init_txt.R' 'init_pal.R' 'init_prob.R' 'comp_prob_prob.R' 'init_freq.R' 'comp_min_N.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_util.R' 'plot_area.R' 'plot_tab.R' 'plot_prism.R' 'plot_fnet.R' 'plot_bar.R' 'plot_icons.R' 'plot_curve.R' 'plot_crisk.R' 'plot_plane.R' 'plot_tree.R' 'plot_mosaic.R' 'data.R' 'read_data.R' 'riskyr_class.R' 'riskyr_sims.R' 'FFTrees_riskyr.R' 'start_riskyr.R'

Encoding UTF-8

LazyData true

License GPL-2 | GPL-3

URL https://riskyr.org/, https://CRAN.R-project.org/package=riskyr, https://github.com/hneth/riskyr/

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

VignetteBuilder knitr

RoxygenNote 7.1.2

Language en-US

NeedsCompilation no

Author Hansjoerg Neth [aut, cre] (<https://orcid.org/0000-0001-5427-3141>),
Felix Gaisbauer [aut] (<https://orcid.org/0000-0002-1285-1246>),
Nico Gradwohl [aut] (<https://orcid.org/0000-0002-8703-905X>),
Wolfgang Gaissmaier [aut] (<https://orcid.org/0000-0001-6273-178X>)

Repository CRAN

Date/Publication 2022-08-15 07:20:05 UTC

R topics documented:

`acc` *Accuracy (acc) is the probability of a correct decision.*

Description

`acc` defines overall accuracy as the probability of correspondence between a positive decision and true condition (i.e., the proportion of correct classification decisions or of `dec_cor` cases).

Usage

```
acc
```

Format

An object of class numeric of length 1.

Details

Importantly, correct decisions dec_cor are not necessarily positive decisions dec_pos.

Understanding or obtaining the accuracy metric acc:

- Definition: acc is the (non-conditional) probability:
  acc = p(dec_cor) = dec_cor/N
  or the base rate (or baseline probability) of a decision being correct, but not necessarily positive.
  acc values range from 0 (no correct decision/prediction) to 1 (perfect decision/prediction).
- Computation: acc can be computed in several ways:
  (a) from prob: acc = (prev x sens) + [(1 - prev) x spec]
  (b) from freq: acc = dec_cor/N = (hi + cr)/(hi + mi + fa + cr)
  (c) as complement of the error rate err: acc = 1 - err
  When frequencies in freq are not rounded, (b) coincides with (a) and (c).
- Perspective: acc classifies a population of N individuals by accuracy/correspondence (acc = dec_cor/N).
  acc is the "by accuracy" or "by correspondence" counterpart to prev (which adopts a "by condition" perspective) and to ppod (which adopts a "by decision" perspective).
- Alternative names: base rate of correct decisions, non-erroneous cases
- In terms of frequencies, acc is the ratio of dec_cor (i.e., hi + cr) divided by N (i.e., hi + mi + fa + cr):
  acc = dec_cor/N = (hi + cr)/(hi + mi + fa + cr)
- Dependencies: acc is a feature of both the environment (true condition) and of the decision process or diagnostic procedure. It reflects the correspondence of decisions to conditions.

See accu for other accuracy metrics and several possible interpretations of accuracy.

References

Consult Wikipedia:Accuracy_and_precision for additional information.

See Also

comp_acc computes accuracy from probabilities; accu lists all accuracy metrics; comp_accu_prob computes exact accuracy metrics from probabilities; comp_accu_freq computes accuracy metrics from frequencies; 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 probabilities: FDR, FOR, NPV, PPV, err, fart, mirt, ppod, prev, sens, spec

Other metrics: accu, comp_accu_freq(), comp_accu_prob(), comp_acc(), comp_err(), err

Examples

```
acc <- .50     # sets a rate of correct decisions of 50%
acc <- 50/100  # (dec_cor) for 50 out of 100 individuals
is_prob(acc)   # TRUE
```

---

accu — *A list containing current accuracy information.*

Description

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

Usage

```
accu
```

Format

An object of class `list` of length 5.

Details

Current metrics include:

1. acc: Overall accuracy as the probability (or proportion) of correctly classifying cases or of dec_cor cases:

   See acc for definition and explanations.

   acc 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.

Notes:

- Accuracy metrics describe the *correspondence* of decisions (or predictions) to actual conditions (or truth).

  There are several possible interpretations of accuracy:

  1. as *probabilities* (i.e., acc being the probability or proportion of correct classifications, or the ratio dec_cor/N),
  2. as *frequencies* (e.g., as classifying a population of N individuals into cases of dec_cor vs. dec_err),
  3. as *correlations* (e.g., see mcc in accu).

- Computing exact accuracy values based on probabilities (by comp_accu_prob) may differ from accuracy values computed from (possibly rounded) frequencies (by comp_accu_freq).

  When frequencies are rounded to integers (see the default of round = TRUE in comp_freq and comp_freq_prob) the accuracy metrics computed by comp_accu_freq correspond to these rounded values. Use comp_accu_prob to obtain exact accuracy metrics from probabilities.

See Also

The corresponding generating function comp_accu_prob computes exact accuracy metrics from probabilities; acc defines accuracy as a probability; comp_accu_freq computes accuracy metrics from frequencies; 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_bwp, pal_bw, pal_kn, pal_mbw, pal_mod, pal_org, pal_rgb, pal_unikn, pal_vir, pal, prob, txt_TF, txt_org, txt

Other metrics: acc, comp_accu_freq(), comp_accu_prob(), comp_acc(), comp_err(), err

Examples

```
accu <- comp_accu_prob()  # => compute exact accuracy metrics (from probabilities)
accu                      # => current accuracy information

## Contrasting comp_accu_freq and comp_accu_prob:
# (a) comp_accu_freq (based on rounded frequencies):
freq1 <- comp_freq(N = 10, prev = 1/3, sens = 2/3, spec = 3/4)   # => rounded frequencies!
accu1 <- comp_accu_freq(freq1$hi, freq1$mi, freq1$fa, freq1$cr)  # => accu1 (based on rounded freq).
# accu1
#
# (b) comp_accu_prob (based on probabilities):
accu2 <- comp_accu_prob(prev = 1/3, sens = 2/3, spec = 3/4)     # => exact accu (based on prob).
# accu2
all.equal(accu1, accu2)  # => 4 differences!
#
# (c) comp_accu_freq (exact values, i.e., without rounding):
freq3 <- comp_freq(N = 10, prev = 1/3, sens = 2/3, spec = 3/4, round = FALSE)
accu3 <- comp_accu_freq(freq3$hi, freq3$mi, freq3$fa, freq3$cr)  # => accu3 (based on EXACT freq).
# accu3
all.equal(accu2, accu3)  # => TRUE (qed).
```

as_pb *Display a percentage as a (numeric and 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(), plot.box()

Other display functions: as_pc()

Examples

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

as_pc — *Display a probability as a (numeric and 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(), plot.box()

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(as_pb(12.3))       # 12.3
```

BRCA1 *Cumulative risk of breast cancer in women with the BRCA1 mutation.*

Description

BRCA1 provides the cumulative risk of breast cancer in a population of women with the BRCA1 mutation as a function of their age (in years).

Usage

```
BRCA1
```

Format

A data frame (11 x 2).

x: age (in years).

y: cumulative risk of developing breast cancer in this (BRCA1) population.

See Also

plot_crisk plots cumulative risk curves.

Other datasets: BRCA1_mam, BRCA1_ova, BRCA2_mam, BRCA2_ova, BRCA2, df_scenarios, t_A, t_B, t_I

BRCA1_mam *Cumulative risk of breast cancer in women with the BRCA1 mutation.*

Description

BRCA1_mam provides the cumulative risk of breast cancer in a population of women with the BRCA1 mutation as a function of their age (in years).

Usage

```
BRCA1_mam
```

Format

A data frame (63 x 2).

age: age (in years).

cumRisk: cumulative risk of developing breast cancer in this (BRCA1) population.

Source

Based on Figure 2 (p. 2408) of Kuchenbaecker, K. B., Hopper, J. L., Barnes, D. R., Phillips, K. A., Mooij, T. M., Roos-Blom, M. J., ... & BRCA1 and BRCA2 Cohort Consortium (2017). Risks of breast, ovarian, and contralateral breast cancer for BRCA1 and BRCA2 mutation carriers. *JAMA*, *317* (23), 2402–2416. doi: 10.1001/jama.2017.7112

See Also

`plot_crisk` plots cumulative risk curves.

Other datasets: `BRCA1_ova`, `BRCA1`, `BRCA2_mam`, `BRCA2_ova`, `BRCA2`, `df_scenarios`, `t_A`, `t_B`, `t_I`

BRCA1_ova — *Cumulative risk of ovarian cancer in women with the BRCA1 mutation.*

Description

`BRCA1_ova` provides the cumulative risk of ovarian cancer in a population of women with the BRCA1 mutation as a function of their age (in years).

Usage

```
BRCA1_ova
```

Format

A data frame (63 x 2).

`age`: age (in years).

`cumRisk`: cumulative risk of developing ovarian cancer in this (BRCA1) population.

Source

Based on Figure 2 (p. 2408) of Kuchenbaecker, K. B., Hopper, J. L., Barnes, D. R., Phillips, K. A., Mooij, T. M., Roos-Blom, M. J., ... & BRCA1 and BRCA2 Cohort Consortium (2017). Risks of breast, ovarian, and contralateral breast cancer for BRCA1 and BRCA2 mutation carriers. *JAMA*, *317* (23), 2402–2416. doi: 10.1001/jama.2017.7112

See Also

`plot_crisk` plots cumulative risk curves.

Other datasets: `BRCA1_mam`, `BRCA1`, `BRCA2_mam`, `BRCA2_ova`, `BRCA2`, `df_scenarios`, `t_A`, `t_B`, `t_I`

BRCA2 *Cumulative risk of breast cancer in women with the BRCA2 mutation.*

Description

BRCA2 provides the cumulative risk of breast cancer in a population of women with the BRCA2 mutation as a function of their age (in years).

Usage

```
BRCA2
```

Format

A data frame (11 x 2).

x: age (in years).

y: cumulative risk of developing breast cancer in this (BRCA2) population.

See Also

plot_crisk plots cumulative risk curves.

Other datasets: BRCA1_mam, BRCA1_ova, BRCA1, BRCA2_mam, BRCA2_ova, df_scenarios, t_A, t_B, t_I

BRCA2_mam *Cumulative risk of breast cancer in women with the BRCA2 mutation.*

Description

BRCA2_mam provides the cumulative risk of breast cancer in a population of women with the BRCA2 mutation as a function of their age (in years).

Usage

```
BRCA2_mam
```

Format

A data frame (63 x 2).

age: age (in years).

cumRisk: cumulative risk of developing breast cancer in this (BRCA2) population.

Source

Based on Figure 2 (p. 2408) of Kuchenbaecker, K. B., Hopper, J. L., Barnes, D. R., Phillips, K. A., Mooij, T. M., Roos-Blom, M. J., ... & BRCA1 and BRCA2 Cohort Consortium (2017). Risks of breast, ovarian, and contralateral breast cancer for BRCA1 and BRCA2 mutation carriers. *JAMA*, *317* (23), 2402–2416. doi: 10.1001/jama.2017.7112

See Also

`plot_crisk` plots cumulative risk curves.

Other datasets: `BRCA1_mam`, `BRCA1_ova`, `BRCA1`, `BRCA2_ova`, `BRCA2`, `df_scenarios`, `t_A`, `t_B`, `t_I`

`BRCA2_ova` *Cumulative risk of ovarian cancer in women with the BRCA2 mutation.*

Description

`BRCA2_ova` provides the cumulative risk of ovarian cancer in a population of women with the BRCA2 mutation as a function of their age (in years).

Usage

```
BRCA2_ova
```

Format

A data frame (63 x 2).

`age`: age (in years).

`cumRisk`: cumulative risk of developing ovarian cancer in this (BRCA2) population.

Source

Based on Figure 2 (p. 2408) of Kuchenbaecker, K. B., Hopper, J. L., Barnes, D. R., Phillips, K. A., Mooij, T. M., Roos-Blom, M. J., ... & BRCA1 and BRCA2 Cohort Consortium (2017). Risks of breast, ovarian, and contralateral breast cancer for BRCA1 and BRCA2 mutation carriers. *JAMA*, *317* (23), 2402–2416. doi: 10.1001/jama.2017.7112

See Also

`plot_crisk` plots cumulative risk curves.

Other datasets: `BRCA1_mam`, `BRCA1_ova`, `BRCA1`, `BRCA2_mam`, `BRCA2`, `df_scenarios`, `t_A`, `t_B`, `t_I` 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 an exact probability (proportion) without rounding.

Understanding the probability acc:

- Definition: acc is the (non-conditional) probability:
  acc = p(dec_cor) = dec_cor/N
  or the base rate (or baseline probability) of a decision being correct, but not necessarily positive.
  acc values range from 0 (no correct decision/prediction) to 1 (perfect decision/prediction).
- Computation: acc can be computed in 2 ways:
  (a) from prob: acc = (prev x sens) + [(1 - prev) x spec]
  (b) from freq: acc = dec_cor/N = (hi + cr)/(hi + mi + fa + cr)
  When frequencies in freq are not rounded, (b) coincides with (a).
- Perspective: acc classifies a population of N individuals by accuracy/correspondence (acc = dec_cor/N).
  acc is the "by accuracy" or "by correspondence" counterpart to prev (which adopts a "by condition" perspective) and to ppod (which adopts a "by decision" perspective).
- Alternative names of acc: base rate of correct decisions, non-erroneous cases
- In terms of frequencies, acc is the ratio of dec_cor (i.e., hi + cr) divided by N (i.e., hi + mi + fa + cr):
  acc = dec_cor/N = (hi + cr)/(hi + mi + fa + cr)
- Dependencies: acc is a feature of both the environment (true condition) and of the decision process or diagnostic procedure. It reflects the correspondence of decisions to conditions.

See accu for other accuracy metrics and several possible interpretations of accuracy.

Value

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

See acc for definition and accu for other accuracy metrics. comp_accu_freq and comp_accu_prob compute accuracy metrics from frequencies and probabilities.

See Also

acc defines accuracy as a probability; accu lists all accuracy metrics; comp_accu_prob computes exact accuracy metrics from probabilities; comp_accu_freq computes accuracy metrics from frequencies; 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_freq(), comp_accu_prob(), comp_comp_pair(), comp_complement(), comp_complete_prob_set(), comp_err(), comp_fart(), comp_mirt(), comp_ppod(), comp_prob_freq(), comp_prob(), comp_sens(), comp_spec()

Other metrics: accu, acc, comp_accu_freq(), comp_accu_prob(), comp_err(), err

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_freq *Compute accuracy metrics of current classification results.*

Description

comp_accu_freq computes a list of 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_freq(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 accuracy 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.

Notes:

- Accuracy metrics describe the *correspondence* of decisions (or predictions) to actual conditions (or truth).

  There are several possible interpretations of accuracy:

1. as *probabilities* (i.e., acc being the proportion of correct classifications, or the ratio dec_cor/N),
2. as *frequencies* (e.g., as classifying a population of N individuals into cases of dec_cor vs. dec_err),
3. as *correlations* (e.g., see mcc in accu).

- Computing exact accuracy values based on probabilities (by comp_accu_prob) may differ from accuracy values computed from (possibly rounded) frequencies (by comp_accu_freq).

  When frequencies are rounded to integers (see the default of round = TRUE in comp_freq and comp_freq_prob) the accuracy metrics computed by comp_accu_freq correspond to these rounded values. Use comp_accu_prob to obtain exact accuracy metrics from probabilities.

Value

A list accu containing current accuracy metrics.

References

Consult Wikipedia: Confusion matrix for additional information.

See Also

accu for all accuracy metrics; comp_accu_prob computes exact accuracy metrics from probabilities; 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, acc, comp_accu_prob(), comp_acc(), comp_err(), err

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

Examples

```
comp_accu_freq()  # => accuracy metrics for freq of current scenario
comp_accu_freq(hi = 1, mi = 2, fa = 3, cr = 4)  # medium accuracy, but cr > hi

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

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

## Contrasting comp_accu_freq and comp_accu_prob:
# (a) comp_accu_freq (based on rounded frequencies):
```

```
freq1 <- comp_freq(N = 10, prev = 1/3, sens = 2/3, spec = 3/4)  # => hi = 2, mi = 1, fa = 2, cr = 5
accu1 <- comp_accu_freq(freq1$hi, freq1$mi, freq1$fa, freq1$cr)  # => accu1 (based on rounded freq).
# accu1
#
# (b) comp_accu_prob (based on probabilities):
accu2 <- comp_accu_prob(prev = 1/3, sens = 2/3, spec = 3/4)     # => exact accu (based on prob).
# accu2
all.equal(accu1, accu2)  # => 4 differences!
#
# (c) comp_accu_freq (exact values, i.e., without rounding):
freq3 <- comp_freq(N = 10, prev = 1/3, sens = 2/3, spec = 3/4, round = FALSE)
accu3 <- comp_accu_freq(freq3$hi, freq3$mi, freq3$fa, freq3$cr)  # => accu3 (based on EXACT freq).
# accu3
all.equal(accu2, accu3)  # => TRUE (qed).
```

`comp_accu_prob` *Compute exact accuracy metrics based on probabilities.*

Description

`comp_accu_prob` computes a list of exact accuracy metrics from a sufficient and valid set of 3 essential probabilities (`prev`, and `sens` or its complement `mirt`, and `spec` or its complement `fart`).

Usage

```
comp_accu_prob(
  prev = prob$prev,
  sens = prob$sens,
  mirt = NA,
  spec = prob$spec,
  fart = NA,
  tol = 0.01,
  w = 0.5
)
```

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.
w	The weighting parameter w (from 0 to 1) for computing weighted accuracy wacc. Default: w = .50 (i.e., yielding balanced accuracy bacc).

Notes:

- Accuracy metrics describe the *correspondence* of decisions (or predictions) to actual conditions (or truth).
  There are several possible interpretations of accuracy:
  1. as *probabilities* (i.e., acc being the proportion of correct classifications, or the ratio dec_cor/N),
  2. as *frequencies* (e.g., as classifying a population of N individuals into cases of dec_cor vs. dec_err),
  3. as *correlations* (e.g., see mcc in accu).
- Computing exact accuracy values based on probabilities (by comp_accu_prob) may differ from accuracy values computed from (possibly rounded) frequencies (by comp_accu_freq).
  When frequencies are rounded to integers (see the default of round = TRUE in comp_freq and comp_freq_prob) the accuracy metrics computed by comp_accu_freq correspond to these rounded values. Use comp_accu_prob to obtain exact accuracy metrics from probabilities.

Details

Currently computed accuracy metrics include:

1. acc: Overall accuracy as the proportion (or probability) of correctly classifying cases or of dec_cor cases:
   (a) from prob: acc = (prev x sens) + [(1 - prev) x spec]
   (b) from freq: acc = dec_cor/N = (hi + cr)/(hi + mi + fa + cr)
   When frequencies in freq are not rounded, (b) coincides with (a).
   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) and some as correlations (e.g., mcc).

Also, accuracy can be viewed as a probability (e.g., the ratio of or link between dec_cor and N) or as a frequency type (containing dec_cor and dec_err).

comp_accu_prob computes exact accuracy metrics from probabilities. When input frequencies were rounded (see the default of round = TRUE in comp_freq and comp_freq_prob) the accuracy metrics computed by comp_accu correspond these rounded values.

Value

A list accu containing current accuracy metrics.

References

Consult Wikipedia: Confusion matrix for additional information.

See Also

accu for all accuracy metrics; comp_accu_freq computes accuracy metrics from frequencies; 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, acc, comp_accu_freq(), comp_acc(), comp_err(), err

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

Examples

```
comp_accu_prob()  # => accuracy metrics for prob of current scenario
comp_accu_prob(prev = .2, sens = .5, spec = .5)  # medium accuracy, but cr > hi.

# Extreme cases:
comp_accu_prob(prev = NaN, sens = NaN, spec = NaN)  # returns list of NA values
comp_accu_prob(prev = 0, sens = NaN, spec = 1)      # returns list of NA values
comp_accu_prob(prev = 0, sens = 0, spec = 1)     # perfect acc = 1, but f1s is NaN
comp_accu_prob(prev = .5, sens = .5, spec = .5)  # random performance
comp_accu_prob(prev = .5, sens = 1,  spec = 1)   # perfect accuracy
comp_accu_prob(prev = .5, sens = 0,  spec = 0)   # zero accuracy, but f1s is NaN
comp_accu_prob(prev = 1,  sens = 1,  spec = 0)   # perfect, but see wacc (0.5) and mcc (0)

# Effects of w:
comp_accu_prob(prev = .5, sens = .6, spec = .4, w = 1/2)  # equal weights to sens and spec
comp_accu_prob(prev = .5, sens = .6, spec = .4, w = 2/3)  # more weight on sens: wacc up
```

```
comp_accu_prob(prev = .5, sens = .6, spec = .4, w = 1/3)  # more weight on spec: wacc down

# Contrasting comp_accu_freq and comp_accu_prob:
# (a) comp_accu_freq (based on rounded frequencies):
freq1 <- comp_freq(N = 10, prev = 1/3, sens = 2/3, spec = 3/4)   # => rounded frequencies!
accu1 <- comp_accu_freq(freq1$hi, freq1$mi, freq1$fa, freq1$cr)  # => accu1 (based on rounded freq).
# accu1

# (b) comp_accu_prob (based on probabilities):
accu2 <- comp_accu_prob(prev = 1/3, sens = 2/3, spec = 3/4)    # => exact accu (based on prob).
# accu2
all.equal(accu1, accu2)  # => 4 differences!
#
# (c) comp_accu_freq (exact values, i.e., without rounding):
freq3 <- comp_freq(N = 10, prev = 1/3, sens = 2/3, spec = 3/4, round = FALSE)
accu3 <- comp_accu_freq(freq3$hi, freq3$mi, freq3$fa, freq3$cr)  # => accu3 (based on EXACT freq).
# accu3
all.equal(accu2, accu3)  # => TRUE (qed).
```

`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_freq(), comp_accu_prob(), comp_acc(), comp_comp_pair(), comp_complete_prob_set(), comp_err(), 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_freq(), comp_accu_prob(), comp_acc(), comp_comp_pair(), comp_complement(), comp_err(), 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_freq(), comp_accu_prob(), comp_acc(), comp_complement(), comp_complete_prob_set(), comp_err(), 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_err — *Compute overall error rate (err) from probabilities.*

Description

comp_err computes overall error rate err from 3 essential probabilities prev, sens, and spec.

Usage

```
comp_err(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_err uses comp_acc to compute err as the complement of acc:

err = 1 - acc

See comp_acc and acc for further details and accu for other accuracy metrics and several possible interpretations of accuracy.

Value

Overall error rate err as a probability (proportion). A warning is provided for NaN values.

See Also

comp_acc computes overall accuracy acc from probabilities; accu lists all accuracy metrics; comp_accu_prob computes exact accuracy metrics from probabilities; comp_accu_freq computes accuracy metrics from frequencies; 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_freq(), comp_accu_prob(), 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()

Other metrics: accu, acc, comp_accu_freq(), comp_accu_prob(), comp_acc(), err

Examples

```
# ways to work:
comp_err(.10, .200, .300)  # => err = 0.71
comp_err(.50, .333, .666)  # => err = 0.5005

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

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

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

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

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_freq(), comp_accu_prob(), comp_acc(), comp_comp_pair(), comp_complement(), comp_complete_prob_set(), comp_err(), 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_freq(), comp_accu_prob(), comp_acc(), comp_comp_pair(), comp_complement(), comp_complete_prob_set(), comp_err(), 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_freq(), comp_accu_prob(), comp_acc(), comp_comp_pair(), comp_complement(), comp_complete_prob_set(), comp_err(), 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 11 key frequencies freq as its output.

Usage

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

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
: Boolean value that determines whether frequency values are rounded to the nearest integer. Default: round = TRUE.

  Note: Removed n_digits parameter: Number of digits to which frequency values are to be rounded when round = FALSE. Default: n_digits = 5.

sample
: Boolean value that determines whether frequency values are sampled from N, given the probability values of prev, sens, and spec. Default: sample = FALSE.

  Note: Sampling uses sample() and returns integer values.

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 and its wrapper function comp_freq_prob round frequencies to nearest integers to avoid decimal values in freq (i.e., round = TRUE by default). When frequencies are rounded, probabilities computed from freq may differ from exact probabilities. Using the option round = FALSE turns off rounding.

Key relationships between probabilities and frequencies:

- Three perspectives on a population:

  A population of N individuals can be split into 2 subsets of frequencies in 3 different ways:

  1. by condition:
     N = cond_true + cond_false
     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:
     N = dec_pos + dec_neg
     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.
  3. by accuracy (i.e., correspondence of decision to condition):
     N = dec_cor + dec_err

  Each perspective combines 2 pairs of the 4 essential probabilities (hi, mi, fa, cr).

  When providing probabilities, 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.

- 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)`
11. accuracy `acc`:
    `acc` = `dec_cor`/`N` = `(hi + cr) / (hi + mi + fa + cr)`
12. rate of hits, given accuracy `p_acc_hi`:
    `p_acc_hi` = `hi`/`dec_cor` = `(1 - cr/dec_cor)`
13. rate of false alarms, given inaccuracy `p_err_fa`:
    `p_err_fa` = `fa`/`dec_err` = `(1 - mi/dec_err)`

Beware of rounding and sampling issues! If frequencies are rounded (by `round = TRUE` in `comp_freq`) or sampled from probabilities (by `sample = TRUE`), then any probabilities computed from `freq` may differ from original and exact probabilities.

Functions translating between representational formats: `comp_prob_prob`, `comp_prob_freq`, `comp_freq_prob`, `comp_freq_freq` (see documentation of `comp_prob_prob` for details).

Value

A list `freq` containing 11 key frequency values.

See Also

`comp_freq_prob` corresponding wrapper function; `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_prob_prob()`

Examples

```
comp_freq()          # ok, using current defaults
length(comp_freq())  # 11 key frequencies

# Rounding:
comp_freq(prev = .5, sens = .5, spec = .5, N = 1)   # yields fa = 1 (see ?round for reason)
comp_freq(prev = .1, sens = .9, spec = .8, N = 10)  # 1 hit (TP, rounded)
comp_freq(prev = .1, sens = .9, spec = .8, N = 10, round = FALSE)    # hi = .9
comp_freq(prev = 1/3, sens = 6/7, spec = 2/3, N = 1, round = FALSE)  # hi = 0.2857143

# Sampling (from probabilistic description):
comp_freq_prob(prev = .5, sens = .5, spec = .5, N = 100, sample = TRUE)  # freq values vary

# Extreme cases:
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 11 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 between frequencies and probabilities (see documentation of comp_freq or comp_prob for details):

- Three perspectives on a population:
  by condition / by decision / by accuracy.
- Defining probabilities in terms of frequencies:
  Probabilities can be computed as ratios between frequencies, but beware of rounding issues.

Functions translating between representational formats: comp_prob_prob, comp_prob_freq, comp_freq_prob, comp_freq_freq (see documentation of comp_prob_prob for details).

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_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 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 11 key frequencies (freq) 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,
  sample = FALSE
)
```

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.
sample	Boolean value that determines whether frequency values are sampled from N, given the probability values of prev, sens, and spec. Default: sample = FALSE. Note: Sampling uses sample() and returns integer values.

Details

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

Defaults and constraints:

- Initial values:

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

  Similarly, the population size N uses the frequency information currently contained in freq as its default. If N is unknown (NA), a suitable minimum value is computed by comp_min_N.

- Constraints:

  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 set to NA.

  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 compute the 3 essential probabilities.

- Rounding:

  By default, comp_freq_prob and its basic function comp_freq round frequencies to nearest integers to avoid decimal values in freq (i.e., round = TRUE by default).

  When frequencies are rounded, probabilities computed from freq may differ from exact probabilities.

  Using the option round = FALSE turns off rounding.

Key relationships between frequencies and probabilities (see documentation of comp_freq or comp_prob for details):

- Three perspectives on a population:

  by condition / by decision / by accuracy.

- Defining probabilities in terms of frequencies:

  Probabilities can be computed as ratios between frequencies, but beware of rounding and sampling issues!

Functions translating between representational formats: comp_prob_prob, comp_prob_freq, comp_freq_prob, comp_freq_freq (see documentation of comp_prob_prob for details).

Value

A list freq containing 11 key 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_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())           # list of 11 key 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 (note NAs to prevent default arguments):
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)

# Rounding:
comp_freq_prob(prev = .5, sens = .5, spec = .5, N = 1)   # yields fa = 1 (see ?round for reason)
comp_freq_prob(prev = .1, sens = .9, spec = .8, N = 10)  # 1 hit (TP, rounded)
comp_freq_prob(prev = .1, sens = .9, spec = .8, N = 10, round = FALSE)  # hi = .9
```

```
# Sampling (from probabilistic description):
comp_freq_prob(prev = .5, sens = .5, spec = .5, N = 100, sample = TRUE)  # freq values vary

# 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 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 center row of boxes in the frequency prism) 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 – especially hi, mi, fa, cr – 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 a frequency tree) is positive but below min_freq.

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

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_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_freq(), comp_accu_prob(), comp_acc(), comp_comp_pair(), comp_complement(), comp_complete_prob_set(), comp_err(), 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_freq(), comp_accu_prob(), comp_acc(), comp_comp_pair(), comp_complement(), comp_complete_prob_set(), comp_err(), 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 (data) from frequencies (description).*

Description

comp_popu 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_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,
  sdt_lbl = txt$sdt_lbl,
  hi_lbl = txt$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_lbl	Text label for condition dimension ("by cd" perspective).
cond_true_lbl	Text label for cond_true cases.
cond_false_lbl	Text label for cond_false cases.
dec_lbl	Text label for decision dimension ("by dc" perspective).
dec_pos_lbl	Text label for dec_pos cases.
dec_neg_lbl	Text label for dec_neg cases.
sdt_lbl	Text label for 4 cases/combinations (SDT classifications).
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 (e.g., `"X/truth/cd"`, `"Y/test/dc"`, `"SDT/cell/class"`).

Details

By default, `comp_popu` uses the text settings contained in `txt`.

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 (e.g., `"X/truth/cd"`, `"Y/test/dc"`, `"SDT/cell/class"`). encoded as ordered factors (with 2, 2, and 4 levels, respectively).

See Also

`read_popu` creates a scenario (description) from data (as df); `write_popu` creates data (as df) from a riskyr scenario (description); `popu` for data format; `num` for basic numeric parameters; `freq` for current frequency information; `txt` for current text settings; `pal` for current color settings.

Other functions converting data/descriptions: `read_popu()`, `write_popu()`

Examples

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

# (A) Diagnostic/screening scenario (using default labels):
comp_popu(hi = 4, mi = 1, fa = 2, cr = 3)  # => computes a table of N = 10 cases.

# (B) Intervention/treatment scenario:
comp_popu(hi = 3, mi = 2, fa = 1, cr = 4,
          cond_lbl = "Treatment", cond_true_lbl = "pill", cond_false_lbl = "placebo",
          dec_lbl = "Health status", dec_pos_lbl = "healthy", dec_neg_lbl = "sick")

# (C) Prevention scenario (e.g., vaccination):
comp_popu(hi = 3, mi = 2, fa = 1, cr = 4,
          cond_lbl = "Vaccination", cond_true_lbl = "yes", cond_false_lbl = "no",
          dec_lbl = "Disease", dec_pos_lbl = "no flu", dec_neg_lbl = "flu")
```

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 necessarily 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_freq(), comp_accu_prob(), comp_acc(), comp_comp_pair(), comp_complement(), comp_complete_prob_set(), comp_err(), 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_freq(), comp_accu_prob(), comp_acc(), comp_comp_pair(), comp_complement(), comp_complete_prob_set(), comp_err(), 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 13 key 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 probability of a correct decision acc, 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 between probabilities and frequencies:

- Three perspectives on a population:
  A population of N individuals can be split into 2 subsets of frequencies in 3 different ways:
  1. by condition:
     N = cond_true + cond_false
     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:
     N = dec_pos + dec_neg
     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.
  3. by accuracy (i.e., correspondence of decision to condition):
     N = dec_cor + dec_err

  Each perspective combines 2 pairs of the 4 essential probabilities (hi, mi, fa, cr).
  When providing probabilities, 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.
- 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)
11. accuracy acc:
    acc = dec_cor/N = (hi + cr) / (hi + mi + fa + cr)
12. rate of hits, given accuracy p_acc_hi:
    p_acc_hi = hi/dec_cor = (1 - cr/dec_cor)
13. rate of false alarms, given inaccuracy p_err_fa:
    p_err_fa = fa/dec_err = (1 - mi/dec_err)

Note: When frequencies are rounded (by round = TRUE in comp_freq), probabilities computed from freq may differ from exact probabilities.

Functions translating between representational formats: comp_prob_prob, comp_prob_freq, comp_freq_prob, comp_freq_freq (see documentation of comp_prob_prob for details).

Value

A list prob containing 13 key probability values.

See Also

prob contains current probability information; accu contains current accuracy 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_freq(), comp_accu_prob(), comp_acc(), comp_comp_pair(), comp_complement(), comp_complete_prob_set(), comp_err(), 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())  # => 13 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 11 frequencies freq for a population of N individuals 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 between frequencies and probabilities (see documentation of comp_freq or comp_prob for details):

- Three perspectives on a population:
  by condition / by decision / by accuracy.
- Defining probabilities in terms of frequencies:
  Probabilities can be computed as ratios between frequencies, but beware of rounding issues.

Functions translating between representational formats: comp_prob_prob, comp_prob_freq, comp_freq_prob, comp_freq_freq (see documentation of comp_prob_prob for details).

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_freq(), comp_accu_prob(), comp_acc(), comp_comp_pair(), comp_complement(), comp_complete_prob_set(), comp_err(), 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()  # => computes prob from current freq

## Beware of rounding:
all.equal(prob, comp_prob_freq())  # => would be TRUE (IF freq were NOT rounded)!
fe <- comp_freq(round = FALSE)     # compute exact freq (not rounded)
all.equal(prob, comp_prob_freq(fe$hi, fe$mi, fe$fa, fe$cr))  # is TRUE (qed).
```

```
## Explain by circular chain (compute prob 1. from num and 2. from freq)
# 0. Inspect current numeric parameters:
num

# 1. Compute currently 11 probabilities in prob (from essential probabilities):
prob <- comp_prob()
prob

# 2. Compute currently 11 frequencies in freq (from essential probabilities):
freq <- comp_freq(round = FALSE)   # no rounding (to obtain same probabilities later)
freq

# 3. Compute currently 11 probabilities again (but now from frequencies):
prob_freq <- comp_prob_freq()
prob_freq

# 4. Check equality of probabilities (in steps 1. and 3.):
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 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 13 key 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_prob is a wrapper function for the more basic function comp_prob.

Extreme probabilities (sets containing 2 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 between frequencies and probabilities (see documentation of comp_freq or comp_prob for details):

- Three perspectives on a population:
  by condition / by decision / by accuracy.
- Defining probabilities in terms of frequencies:
  Probabilities can be computed as ratios between frequencies, but beware of rounding issues.

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).

Value

A list prob containing 13 key 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()

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())  # 13 key probability values

# 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_freq(), comp_accu_prob(), comp_acc(), comp_comp_pair(), comp_complement(), comp_complete_prob_set(), comp_err(), 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_freq(), comp_accu_prob(), comp_acc(), comp_comp_pair(), comp_complement(), comp_complete_prob_set(), comp_err(), 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/accurate (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 the accuracy acc.
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)
3. correspondence: When not rounding the frequencies of freq then
   dec_cor = N x acc = hi + cr
   (i.e., dec_cor corresponds to the sum of true positives hi and true negatives cr.

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 (as df).*

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 25 rows (i.e., scenarios) and 21 columns (variables describing each scenario):

See scenarios for a list of scenarios and the variables currently contained in df_scenarios.

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

The variables scen_src and scen_apa provide a scenario's source information.

Details

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

See Also

scenarios contains all scenarios as riskyr objects; riskyr initializes a riskyr scenario; txt contains basic text information; 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.

Other datasets: BRCA1_mam, BRCA1_ova, BRCA1, BRCA2_mam, BRCA2_ova, BRCA2, t_A, t_B, t_I

err — *Error rate (err) as the probability of an incorrect decision.*

Description

err defines the error rate as the complement of accuracy acc or lack of correspondence of decisions to conditions.

Usage

```
err
```

Format

An object of class numeric of length 1.

Details

Definition:

err = (1 - acc)

When freq are not rounded (round = FALSE) then

err = dec_err/N = (mi + fa)/N

err is currently not included in prob, but shown in plots.

See err's complement of accuracy acc for computation and accu for current accuracy metrics and several possible interpretations of accuracy.

See Also

acc provides overall accuracy; comp_acc computes accuracy from probabilities; accu lists current accuracy metrics; comp_accu_prob computes exact accuracy metrics from probabilities; comp_accu_freq computes accuracy metrics from frequencies; 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 probabilities: FDR, FOR, NPV, PPV, acc, fart, mirt, ppod, prev, sens, spec

Other metrics: accu, acc, comp_accu_freq(), comp_accu_prob(), comp_acc(), comp_err()

Examples

```
err <- .50     # sets a rate of incorrect decisions of 50%
err <- 50/100  # (dec_err) for 50 out of 100 individuals
is_prob(err)   # TRUE
```

---

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 probabilities.

Other probabilities: `FDR`, `FOR`, `NPV`, `PPV`, `acc`, `err`, `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
```

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

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 probabilities.

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

Examples

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

FFTrees_riskyr *Convert from FFTrees to riskyr objects.*

Description

FFTrees_riskyr converts an FFTrees object — as generated by the **FFTrees** package — into a corresponding riskyr object.

Usage

```
FFTrees_riskyr(x, data = "train", tree = 1)
```

Arguments

x	An FFTrees object (generated by **FFTrees**).
data	The type of data to consider (as a character string). Must be either "train" (for training/fitting data) or "test" (for test/prediction data). Default: data = "train".
tree	An integer specifying the tree to consider (as an integer). Default: tree = 1.

Details

FFTrees_riskyr essentially allows using **riskyr** functions to visualize a fast-and-frugal tree (FFT)'s performance information (as contained in a 2x2 matrix of frequency counts).

The R package **FFTrees** creates, visualizes, and evaluates fast-and-frugal trees (FFTs) for solving binary classification problems in an efficient and transparent fashion.

Value

A **riskyr** scenario (as riskyr object).

References

See https://CRAN.R-project.org/package=FFTrees or https://github.com/ndphillips/FFTrees for information on the R package **FFTrees**.

See Also

riskyr initializes a riskyr scenario.

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 probabilities.

Other probabilities: `FDR`, `NPV`, `PPV`, `acc`, `err`, `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
```

---

freq — *List current frequency information.*

Description

`freq` is a list of named numeric variables containing 11 key frequencies (and their values):

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 between frequencies and probabilities (see documentation of comp_freq or comp_prob for details):

- Three perspectives on a population:
  by condition / by decision / by accuracy.
- Defining probabilities in terms of frequencies:
  Probabilities can be computed as ratios between frequencies, but beware of rounding issues.

Functions translating between representational formats: comp_prob_prob, comp_prob_freq, comp_freq_prob, comp_freq_freq (see documentation of comp_prob_prob for details).

Visualizations of current frequency information are provided by plot_prism and plot_icons.

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_bwp, pal_bw, pal_kn, pal_mbw, pal_mod, pal_org, pal_rgb, pal_unikn, pal_vir, pal, prob, txt_TF, txt_org, txt

Examples

```
freq <- comp_freq()  # initialize freq to default parameters
freq                 # show current values
length(freq)         # 11 known frequencies
names(freq)          # show names of known frequencies
```

---

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(), riskyr()

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(
  N_col = pal_def["N"],
  cond_true_col = pal_def["cond_true"],
  cond_false_col = pal_def["cond_false"],
  dec_pos_col = pal_def["dec_pos"],
  dec_neg_col = pal_def["dec_neg"],
  dec_cor_col = pal_def["dec_cor"],
  dec_err_col = pal_def["dec_err"],
  hi_col = pal_def["hi"],
  mi_col = pal_def["mi"],
  fa_col = pal_def["fa"],
  cr_col = pal_def["cr"],
  PPV_col = pal_def["ppv"],
  NPV_col = pal_def["npv"],
  txt_col = pal_def["txt"],
  bg_col = pal_def["bg"],
  brd_col = pal_def["brd"]
)
```

Arguments

`N_col`	Color representing the *population* of `N` cases or individuals.
`cond_true_col`	Color representing cases of `cond_true`, for which the current condition is `TRUE`.
`cond_false_col`	Color representing cases of in `cond_false`, for which the current condition is `FALSE`.
`dec_pos_col`	Color representing cases of `dec_pos`, for which the current decision is `positive`.
`dec_neg_col`	Color representing cases in `dec_neg`, for which the current decision is `negative`.
`dec_cor_col`	Color representing cases of correct decisions `dec_cor`, for which the current decision is `accurate`.
`dec_err_col`	Color representing cases in erroneous decisions `dec_err`, for which the current decision is `inaccurate`.

`hi_col`	Color representing *hits* or true positives in `hi` (i.e., correct cases for which the current condition is TRUE and the decision is positive).
`mi_col`	Color representing *misses* or false negatives in `mi` (i.e., incorrect cases for which the current condition is TRUE but the decision is negative).
`fa_col`	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).
`cr_col`	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).
`PPV_col`	Color representing *positive predictive values* `PPV` (i.e., the conditional probability that the condition is TRUE, provided that the decision is positive).
`NPV_col`	Color representing *negative predictive values* `NPV` (i.e., the conditional probability that the condition is FALSE, provided that the decision is negative).
`txt_col`	Color used for text labels.
`bg_col`	Background color of plot (used to set `par(bg = bg_col)`).
`brd_col`	Color used for borders (e.g., around bars or boxes).

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()`, `riskyr()`

Examples

```
init_pal()          # => define and return a vector of current (default) colors
length(init_pal())  # => 15 named colors
pal <- init_pal(N_col = "steelblue4")  # => change a color (stored in pal)
pal <- init_pal(brd_col = NA)          # => remove a color
```

`init_txt` *Initialize basic text elements.*

Description

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

Usage

```
init_txt(
  scen_lbl = txt_lbl_def$scen_lbl,
  scen_txt = txt_lbl_def$scen_txt,
  scen_src = txt_lbl_def$scen_src,
  scen_apa = txt_lbl_def$scen_apa,
  scen_lng = txt_lbl_def$scen_lng,
  popu_lbl = txt_lbl_def$popu_lbl,
  N_lbl = txt_lbl_def$N_lbl,
  cond_lbl = txt_lbl_def$cond_lbl,
  cond_true_lbl = txt_lbl_def$cond_true_lbl,
  cond_false_lbl = txt_lbl_def$cond_false_lbl,
  dec_lbl = txt_lbl_def$dec_lbl,
  dec_pos_lbl = txt_lbl_def$dec_pos_lbl,
  dec_neg_lbl = txt_lbl_def$dec_neg_lbl,
  acc_lbl = txt_lbl_def$acc_lbl,
  dec_cor_lbl = txt_lbl_def$dec_cor_lbl,
  dec_err_lbl = txt_lbl_def$dec_err_lbl,
  sdt_lbl = txt_lbl_def$sdt_lbl,
  hi_lbl = txt_lbl_def$hi_lbl,
  mi_lbl = txt_lbl_def$mi_lbl,
  fa_lbl = txt_lbl_def$fa_lbl,
  cr_lbl = txt_lbl_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 general name describing the current *population*.
`N_lbl`	A brief label for the current population `popu` or sample.
`cond_lbl`	A general name for the *condition* dimension currently considered (e.g., some clinical condition).
`cond_true_lbl`	A short label for the *presence* of the current condition or `cond_true` cases (the condition's true state of TRUE).
`cond_false_lbl`	A short label for the *absence* of the current condition or `cond_false` cases (the condition's true state of FALSE).
`dec_lbl`	A general name for the *decision* dimension (e.g., some diagnostic test) currently made.

dec_pos_lbl
: A short label for *positive* decisions or dec_pos cases (e.g., predicting the presence of the condition).

dec_neg_lbl
: A short label for *negative* decisions or dec_neg cases (e.g., predicting the absence of the condition).

acc_lbl
: A general name for the *accuracy* dimension (e.g., correspondence of decision to condition).

dec_cor_lbl
: A short label for *correct* decisions or dec_cor cases (e.g., accurately predicting the condition).

dec_err_lbl
: A short label for *erroneous* decisions or dec_err cases (e.g., inaccurately predicting the condition).

sdt_lbl
: A name for the case/category/cell dimension in the 2x2 contingency table (SDT: condition x decision).

hi_lbl
: A short 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 short 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 short 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 short 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.

However, you can directly specify scenario-specific text elements when defining a scenario with the riskyr function.

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(), riskyr()

Examples

```
init_txt()          # defines a list of (default) text elements
length(init_txt())  # 21

# Customizing current text elements:
txt <- init_txt(scen_lbl = "My scenario",
                scen_src = "My source",
                N_lbl = "My population")
```

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, within some tolerance range tol).

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_integer(), is_matrix(), 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_integer(), is_matrix(), 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_integer(), is_matrix(), 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(0:3)  # => 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:
# is.integer(2)          # => FALSE!
```

`is_integer` *Test for inters (i.e., whole numbers).*

Description

`is_integer` tests if `x` contains *only* integer numbers.

Usage

```
is_integer(x, tol = .Machine$double.eps^0.5)
```

Arguments

`x`	Number(s) to test (required, accepts numeric vectors).
`tol`	Numeric tolerance value. Default: `tol = .Machine$double.eps^0.5` (see `?.Machine` for details).

Details

Thus, `is_integer` does what the **base** R function `is.integer` is *not* designed to do:

- `is_integer()` returns TRUE or FALSE depending on whether its numeric argument `x` is an integer value (i.e., a "whole" number).
- `is.integer()` returns TRUE or FALSE depending on whether its argument is of type "integer", and FALSE if its argument is a factor.

See the documentation of `is.integer` for definition and details.

See Also

`is.integer` function of the R **base** package.

Other verification functions: `is_complement()`, `is_extreme_prob_set()`, `is_freq()`, `is_matrix()`, `is_perc()`, `is_prob()`, `is_suff_prob_set()`, `is_valid_prob_pair()`, `is_valid_prob_set()`, `is_valid_prob_triple()`

Examples

```
is_integer(2)    # TRUE
is_integer(2/1)  # TRUE
is_integer(2/3)  # FALSE
x <- seq(1, 2, by = 0.5)
is_integer(x)

# Note contrast to base R:
is.integer(2/1)  # FALSE!

# Compare:
is.integer(1 + 2)
is_integer(1 + 2)
```

---

is_matrix — *Verify a 2x2 matrix as a numeric contingency table.*

Description

`is_matrix` verifies that `mx` is a valid 2x2 matrix (i.e., a numeric contingency table).

Usage

```
is_matrix(mx)
```

Arguments

`mx`	An object to verify (required).

Details

`is_matrix` is more restrictive than `is.matrix`, as it also requires that `mx` `is.numeric`, `is.table`, `nrows(mx) == 2`, and `ncols(mx) == 2`.

Value

A Boolean value: `TRUE` if `mx` is a numeric matrix and 2x2 contingency table; otherwise `FALSE`.

References

Neth, H., Gradwohl, N., Streeb, D., Keim, D.A., & Gaissmaier, W. (2021). Perspectives on the 2×2 matrix: Solving semantically distinct problems based on a shared structure of binary contingencies. *Frontiers in Psychology*, *11*, 567817. doi: doi: 10.3389/fpsyg.2020.567817

See Also

Other verification functions: `is_complement()`, `is_extreme_prob_set()`, `is_freq()`, `is_integer()`, `is_perc()`, `is_prob()`, `is_suff_prob_set()`, `is_valid_prob_pair()`, `is_valid_prob_set()`, `is_valid_prob_triple()`

Examples

```
is_matrix(1:4)
is_matrix(matrix("A"))
is_matrix(matrix(1:4))
is_matrix(as.table(matrix(1:4, nrow = 1, ncol = 4)))
is_matrix(as.table(matrix(1:4, nrow = 4, ncol = 1)))
is_matrix(as.table(matrix(1:4, nrow = 2, ncol = 2)))
```

---

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_integer()`, `is_matrix()`, `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.

## note:
# 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` (a scalar or a vector) 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_integer()`, `is_matrix()`, `is_perc()`, `is_suff_prob_set()`, `is_valid_prob_pair()`, `is_valid_prob_set()`, `is_valid_prob_triple()`

Examples

```
is_prob(1/2)                 # TRUE
is_prob(2)                   # FALSE

# vectors:
p_seq <- seq(0, 1, by = .1)  # Vector of probabilities
is_prob(p_seq)               # TRUE (as scalar, not: TRUE TRUE etc.)
is_prob(c(.1, 2, .9))        # FALSE (as scalar, not: TRUE FALSE etc.)

## 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, NA_warn = TRUE)         # => FALSE + warning (outside range element)
# is_prob(c(.5, 8), NA_warn = TRUE)  # => FALSE + warning (outside range vector element)
# is_prob("Laplace", NA_warn = TRUE) # => 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 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).

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_integer(), is_matrix(), 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_integer()`, `is_matrix()`, `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; is_prob verifies probabilities; prob contains current probability information; num contains basic numeric variables; init_num initializes basic numeric variables; 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_integer(), is_matrix(), 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_integer(), is_matrix(), 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 probabilities.

Other probabilities: FDR, FOR, NPV, PPV, acc, err, 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)
```

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 between frequencies and probabilities (see documentation of comp_freq or comp_prob for details):

- Three perspectives on a population:
  by condition / by decision / by accuracy.
- Defining probabilities in terms of frequencies:
  Probabilities can be computed as ratios between frequencies, but beware of rounding issues.

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 probabilities.

Other probabilities: FDR, FOR, PPV, acc, err, 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
```

---

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_bwp, pal_bw, pal_kn, pal_mbw, pal_mod, pal_org, pal_rgb, pal_unikn, pal_vir, pal, prob, txt_TF, txt_org, txt

Examples

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

---

pal — *List current values of scenario color palette.*

Description

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

Usage

```
pal
```

Format

An object of class character of length 16.

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. cond_true Color representing cases of cond_true, for which the current condition is TRUE.
3. cond_false Color representing cases of in cond_false, for which the current condition is FALSE.
4. dec_pos Color representing cases of dec_pos, for which the current decision is positive.
5. dec_neg Color representing cases in dec_neg, for which the current decision is negative.
6. dec_cor Color representing cases of correct decisions dec_cor, for which the current decision is accurate.

7. dec_err Color representing cases of erroneous decisions dec_err, for which the current decision is inaccurate.
8. 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).
9. 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).
10. 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).
11. 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).
12. ppv Color representing *positive predictive values* PPV (i.e., the conditional probability that the condition is TRUE, provided that the decision is positive).
13. npv Color representing *negative predictive values* NPV (i.e., the conditional probability that the condition is FALSE, provided that the decision is negative).
14. txt Color used for text labels.
15. brd Color used for borders.
16. bg Color used for plot background (used to set par(bg = bg_col)).

Note that color names for frequencies correspond to frequency names, but are different for probabilities (which are written in lowercase and only PPV and NPV have assigned colors).

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, pal_bwp, pal_bw, pal_kn, pal_mbw, pal_mod, pal_org, pal_rgb, pal_unikn, pal_vir, prob, txt_TF, txt_org, txt

Examples

```
pal        # shows all color names and current values
pal["hi"]  # shows the current color for hits (true positives, TP)
pal["hi"] <- "gold"  # defines a new color for hits (true positives, TP)
```

pal_bw *Alternative color palette for black-and-white (greyscale) graphs.*

Description

pal_bw is initialized to a vector of named elements (colors) to define an alternative (black-and-white, b/w) scenario color scheme.

Usage

```
pal_bw
```

Format

An object of class character of length 16.

Details

Note that pal_bw uses various shades of grey for frequency boxes so that their bounds remain visible on a white background when f_lwd = 0 (as per default for most graphs).

See pal_bwp for a stricter version that enforces black text and lines on white boxes (e.g., for printing purposes).

See pal for default color information.

Assign pal <- pal_bw to use as default color scheme throughout the **riskyr** package.

See Also

pal contains current color information; init_pal initializes color information.

Other color palettes: pal_bwp, pal_crisk, pal_kn, pal_mbw, pal_mod, pal_org, pal_rgb, pal_unikn, pal_vir

Other lists containing current scenario information: accu, freq, num, pal_bwp, pal_kn, pal_mbw, pal_mod, pal_org, pal_rgb, pal_unikn, pal_vir, pal, prob, txt_TF, txt_org, txt

Examples

```
pal_bw        # shows all color names and current values
pal_bw["hi"]  # shows the current color for hits (true positives, TP)
pal_bw["hi"] <- "gold" # defines a new color for hits (true positives, TP)
```

pal_bwp *Alternative color palette for black-and-white graphs (for printing purposes).*

Description

pal_bwp is initialized to a vector of named elements (colors) to define a strict (black-and-white, b/w) scenario color scheme that is suited for printing graphs in black-and-white.

Usage

```
pal_bwp
```

Format

An object of class character of length 16.

Details

pal_bwp is a stricter version of the greyscale palette pal_bw that enforces black text and lines on white boxes. Thus, the bounds of frequency boxes are invisible on white backgrounds unless the default of f_lwd = 0 is changed (e.g., to f_lwd = 1).

Some background colors (of frequencies) are also used as foreground colors (of probabilities, e.g., in plot_curve and plot_plane). For this reason, the plotting functions detect and adjust colors and/or line settings when pal_bwp is used.

See pal_bw for a more permissible black-and-white palette that uses various shades of grey for frequency boxes so that their bounds remain visible on a white background when f_lwd = 0 (as per default for most graphs).

See pal for default color information.

Assign pal <- pal_bwp to use as default color scheme throughout the **riskyr** package.

See Also

pal contains current color information; init_pal initializes color information.

Other color palettes: pal_bw, pal_crisk, pal_kn, pal_mbw, pal_mod, pal_org, pal_rgb, pal_unikn, pal_vir

Other lists containing current scenario information: accu, freq, num, pal_bw, pal_kn, pal_mbw, pal_mod, pal_org, pal_rgb, pal_unikn, pal_vir, pal, prob, txt_TF, txt_org, txt

Examples

```
pal_bwp        # shows all color names and current values
pal_bwp["hi"]  # shows the current color for hits (true positives, TP)
pal_bwp["hi"] <- "gold" # defines a new color for hits (true positives, TP)
```

pal_crisk *Color palette for cumulative risk curve.*

Description

pal_crisk defines a default color palette for the plot_crisk function (as a named vector).

Usage

```
pal_crisk
```

Format

An object of class character of length 10.

Details

Color names and referents in plots generated by plot_crisk:

1. "cum": Cumulative risk curve
2. "rinc": Relative risk increments
3. "txt": Text labels
4. "aux": Auxiliary labels and lines
5. "high": Highlighting elements
6. "pas": Past/passed risk
7. "rem": Remaining risk
8. "delta": Delta-X- and -Y increments
9. "poly": Polygon of increments
10. "popu": Population partitions

See Also

plot_crisk plots cumulative risk curves; pal contains current color information; init_pal initializes color information.

Other color palettes: pal_bwp, pal_bw, pal_kn, pal_mbw, pal_mod, pal_org, pal_rgb, pal_unikn, pal_vir

Examples

```
pal_crisk  # show color palette (and names)
```

pal_kn *Alternative color palette for kn.*

Description

pal_kn is initialized to a vector of named elements (colors) to define an alternative (kn) scenario color scheme.

Usage

```
pal_kn
```

Format

An object of class character of length 16.

Details

See pal for default color information.

Assign pal <- pal_kn to use as default color scheme throughout the **riskyr** package.

See Also

pal_unikn contains more **unikn** colors; pal contains current color information; init_pal initializes color information.

Other color palettes: pal_bwp, pal_bw, pal_crisk, pal_mbw, pal_mod, pal_org, pal_rgb, pal_unikn, pal_vir

Other lists containing current scenario information: accu, freq, num, pal_bwp, pal_bw, pal_mbw, pal_mod, pal_org, pal_rgb, pal_unikn, pal_vir, pal, prob, txt_TF, txt_org, txt

Examples

```
pal_kn        # shows all color names and current values
pal_kn["hi"]  # shows the current color for hits (true positives, TP)
pal_kn["hi"] <- "grey" # defines a new color for hits (true positives, TP)
```

pal_mbw *Modern and reduced color palette (in green/blue/bw).*

Description

pal_mod is initialized to a vector of named colors to define a reduced modern scenario color scheme (in green/blue/bw).

Usage

```
pal_mbw
```

Format

An object of class character of length 16.

Details

See pal_org for original color information; pal_mod for a richer modern color palette; and pal_bw for a more reduced black-and-white color palette.

Assign pal <- pal_mbw to use as default color scheme throughout the **riskyr** package.

See Also

pal contains current color information; init_pal initializes color information; pal_org for original color palette; pal_mod for a richer modern color palette; pal_bw for a more reduced black-and-white color palette.

Other color palettes: pal_bwp, pal_bw, pal_crisk, pal_kn, pal_mod, pal_org, pal_rgb, pal_unikn, pal_vir

Other lists containing current scenario information: accu, freq, num, pal_bwp, pal_bw, pal_kn, pal_mod, pal_org, pal_rgb, pal_unikn, pal_vir, pal, prob, txt_TF, txt_org, txt

Examples

```
pal_mbw        # shows all color names and current values
pal_mbw["hi"]  # shows the current color for hits (true positives, TP)
pal_mbw["hi"] <- "gold" # defines a new color for hits (true positives, TP)
```

pal_mod *Modern color palette (in green/blue/orange).*

Description

pal_mod is initialized to a vector of named colors to define a modern scenario color scheme (in green/blue/orange).

Usage

```
pal_mod
```

Format

An object of class character of length 16.

Details

See pal for default color information.

Assign pal <- pal_mod to use as default color scheme throughout the **riskyr** package.

See Also

pal contains current color information; init_pal initializes color information.

Other color palettes: pal_bwp, pal_bw, pal_crisk, pal_kn, pal_mbw, pal_org, pal_rgb, pal_unikn, pal_vir

Other lists containing current scenario information: accu, freq, num, pal_bwp, pal_bw, pal_kn, pal_mbw, pal_org, pal_rgb, pal_unikn, pal_vir, pal, prob, txt_TF, txt_org, txt

Examples

```
pal_mod        # shows all color names and current values
pal_mod["hi"]  # shows the current color for hits (true positives, TP)
pal_mod["hi"] <- "gold" # defines a new color for hits (true positives, TP)
```

pal_org — *Original color palette.*

Description

pal_org is a copy of pal (to retrieve original set of colors in case pal is changed).

Usage

```
pal_org
```

Format

An object of class character of length 16.

Details

See pal for default color information.

Assign pal <- pal_org to re-set default color scheme throughout the **riskyr** package.

See Also

pal contains current color information; init_pal initializes color information.

Other color palettes: pal_bwp, pal_bw, pal_crisk, pal_kn, pal_mbw, pal_mod, pal_rgb, pal_unikn, pal_vir

Other lists containing current scenario information: accu, freq, num, pal_bwp, pal_bw, pal_kn, pal_mbw, pal_mod, pal_rgb, pal_unikn, pal_vir, pal, prob, txt_TF, txt_org, txt

Examples

```
pal_org        # shows all color names and current values
pal_org["hi"]  # shows the current color for hits (true positives, TP)
pal_org["hi"] <- "gold" # defines a new color for hits (true positives, TP)
```

pal_rgb — *Alternative color palette for graphs (with RGB colors).*

Description

pal_rgb is initialized to a vector of named elements (colors) to define an alternative (reduced) scenario color scheme (using red, green, and blue colors).

Usage

```
pal_rgb
```

Format

An object of class `character` of length 16.

Details

See `pal` for default color information.

Assign `pal <- pal_rgb` to use as default color scheme throughout the **riskyr** package.

See Also

`pal` contains current color information; `init_pal` initializes color information.

Other color palettes: `pal_bwp`, `pal_bw`, `pal_crisk`, `pal_kn`, `pal_mbw`, `pal_mod`, `pal_org`, `pal_unikn`, `pal_vir`

Other lists containing current scenario information: `accu`, `freq`, `num`, `pal_bwp`, `pal_bw`, `pal_kn`, `pal_mbw`, `pal_mod`, `pal_org`, `pal_unikn`, `pal_vir`, `pal`, `prob`, `txt_TF`, `txt_org`, `txt`

Examples

```
pal_rgb        # shows all color names and current values
pal_rgb["hi"]  # shows the current color for hits (true positives, TP)
pal_rgb["hi"] <- "gold" # defines a new color for hits (true positives, TP)
```

---

pal_unikn — *Alternative color palette for unikn.*

Description

`pal_unikn` is initialized to a vector of named elements (colors) to define an alternative (unikn) scenario color scheme.

Usage

```
pal_unikn
```

Format

An object of class `character` of length 16.

Details

See `pal` for default color information.

Assign `pal <- pal_unikn` to use as default color scheme throughout the **riskyr** package.

See Also

pal_kn contains fewer **unikn** colors; pal contains current color information; init_pal initializes color information.

Other color palettes: pal_bwp, pal_bw, pal_crisk, pal_kn, pal_mbw, pal_mod, pal_org, pal_rgb, pal_vir

Other lists containing current scenario information: accu, freq, num, pal_bwp, pal_bw, pal_kn, pal_mbw, pal_mod, pal_org, pal_rgb, pal_vir, pal, prob, txt_TF, txt_org, txt

Examples

```
pal_unikn        # shows all color names and current values
pal_unikn["hi"]  # shows the current color for hits (true positives, TP)
pal_unikn["hi"] <- "grey" # defines a new color for hits (true positives, TP)
```

---

pal_vir — *Alternative color palette using viridis colors.*

Description

pal_vir is initialized to a vector of named elements (colors) to define a scenario color scheme modeled on the viridis color scale.

Usage

```
pal_vir
```

Format

An object of class character of length 16.

Details

These colors are select by the Matplotlib viridis color map created by Stéfan van der Walt and Nathaniel Smith. See the viridisLite package (maintained by Simon Garnier) for further information.

Assign pal <- pal_vir to use as default color scheme throughout the **riskyr** package.

See Also

pal contains current color information; init_pal initializes color information.

Other color palettes: pal_bwp, pal_bw, pal_crisk, pal_kn, pal_mbw, pal_mod, pal_org, pal_rgb, pal_unikn

Other lists containing current scenario information: accu, freq, num, pal_bwp, pal_bw, pal_kn, pal_mbw, pal_mod, pal_org, pal_rgb, pal_unikn, pal, prob, txt_TF, txt_org, txt

Examples

```
pal_vir        # shows all color names and current values
pal_vir["hi"]  # shows the current color for hits (true positives, TP)
pal_vir["hi"] <- "green3" # defines a new color for hits (true positives, TP)
```

`plot.box` *Plot a frequency box object.*

Description

`plot.box` is a utility method that allows to plot low level boxes for `riskyr` plots.

Usage

```
## S3 method for class 'box'
plot(x, cur_freq = freq, lbl_txt = txt, col_pal = pal, ...)
```

Arguments

`x`	The box (i.e., an object of class box) to be plotted.
`cur_freq`	Current frequency information (see `freq` for details).
`lbl_txt`	Current text information (see `txt` for details).
`col_pal`	Current color palette (see `pal` for details).
`...`	Additional (graphical) 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 utility functions: `as_pb()`, `as_pc()`

plot.riskyr — *Plot a riskyr scenario.*

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, type = "prism", main = NULL, sub = NULL, ...)
```

Arguments

- x — An object of class "riskyr", usually a result of a call to riskyr. Pre-defined scenarios are also of type "riskyr".
- type — The type of plot to be generated.
- main — Text label for main plot title. Default: main = NULL (using x$scen_lbl per default).
- sub — Text label for plot subtitle (on 2nd line). Default: sub = NULL (using sub = "type" shows plot type).

  The following plot types are currently available:

  1. type = "prism" or type = "net" or type = "tree": Risk information is plotted in a network diagram of frequencies and probabilities (default). See plot_prism for further options.
  2. type = "tab" or type = "ftab": Risk information is plotted as a 2-by-2 frequency or contingency table. See plot_tab for further options.
  3. type = "area" or type = "mosaic": Risk information is plotted as a mosaic plot (scaled area). See plot_area for further options.
  4. type = "bar" or type = "fbar": Risk information is plotted as a bar chart. See plot_bar for further options.
  5. type = "icons" or type = "iconarray": The underlying population is plotted as an array of icons. See plot_icons for further options.
  6. type = "curve" or type = "curves": Draws curves of selected values (including PPV, NPV) See plot_curve for further options.
  7. type = "plane" or 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

riskyr initializes a riskyr scenario.

Other visualization functions: plot_area(), plot_bar(), plot_crisk(), plot_curve(), plot_fnet(), plot_icons(), plot_mosaic(), plot_plane(), plot_prism(), plot_tab(), plot_tree()

Other riskyr scenario functions: riskyr(), summary.riskyr()

Examples

```
# Select a scenario (from list of scenarios):
s1 <- scenarios$n1  # select scenario 1 from scenarios
plot(s1)  # default plot (type = "prism")

# Plot types currently available:
plot(s1, type = "prism")                # prism/network diagram (default)
plot(s1, type = "tree", by = "cd")      # tree diagram (only 1 perspective)
plot(s1, type = "area")                 # area/mosaic plot
plot(s1, type = "tab")                  # 2x2 frequency/contingency table
plot(s1, type = "bar", dir = 2)         # bar plot
plot(s1, type = "icons")                # icon array
plot(s1, type = "curve", what = "all")  # curves as fn. of prev
plot(s1, type = "plane", what = "NPV")  # plane as function of sens & spec
plot(s1, type = "default")              # unknown type: use default plot
```

---

plot_area *Plot an area diagram of probabilities or frequencies.*

Description

plot_area assigns the total probability or population frequency to an area (square or rectangle) and shows the probability or frequency of 4 classification cases (hi, mi, fa, cr) as relative proportions of this area.

Usage

```
plot_area(
  prev = num$prev,
  sens = num$sens,
  mirt = NA,
  spec = num$spec,
  fart = NA,
  N = num$N,
  by = "cddc",
  p_split = "v",
  area = "sq",
  scale = "p",
  round = TRUE,
```

```
    sample = FALSE,
    sum_w = 0.1,
    gaps = c(NA, NA),
    f_lbl = "num",
    f_lbl_sep = NA,
    f_lbl_sum = "num",
    f_lbl_hd = "nam",
    f_lwd = 0,
    p_lbl = NA,
    arr_c = -3,
    col_p = c(grey(0.15, 0.99), "yellow", "yellow"),
    brd_dis = 0.06,
    lbl_txt = txt,
    main = txt$scen_lbl,
    sub = "type",
    title_lbl = NULL,
    cex_lbl = 0.9,
    cex_p_lbl = NA,
    col_pal = pal,
    mar_notes = FALSE,
    ...
  )
```

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. Note: N is not represented in the plot, but used for computing frequency information freq from current probabilities prob.
- **by** — A character code specifying 2 perspectives that split the population into subsets, with 6 options:
  1. "cddc": by condition (cd) and by decision (dc) (default);
  2. "cdac": by condition (cd) and by accuracy (ac);
  3. "dccd": by decision (dc) and by condition (cd);

4. "dcac": by decision (dc) and by accuracy (ac);
5. "accd": by accuracy (ac) and by condition (cd);
6. "acdc": by accuracy (ac) and by decision (dc).

p_split
: Primary perspective for population split, with 2 options:
1. "v": vertical (default);
2. "h": horizontal.

area
: A character code specifying the shape of the main area, with 2 options:
1. "sq": main area is scaled to a square (default);
2. "no": no scaling (rectangular area fills plot size).

scale
: Scale probabilities and corresponding area dimensions either by exact probability or by (rounded or non-rounded) frequency, with 2 options:
1. "p": scale main area dimensions by exact probability (default);
2. "f": re-compute probabilities from (rounded or non-rounded) frequencies and scale main area dimensions by their frequency.

Note: scale setting matters for the display of probability values and for area plots with small population sizes N when round = TRUE.

round
: A Boolean option specifying whether computed frequencies are rounded to integers. Default: round = TRUE.

sample
: Boolean value that determines whether frequency values are sampled from N, given the probability values of prev, sens, and spec. Default: sample = FALSE.

sum_w
: Border width of 2 perspective summaries (on top and left borders) of main area as a proportion of area size (i.e., in range 0 <= sum_w <= 1). Default: sum_w = .10. Setting sum_w = 0, NA, or NULL removes summaries; setting sum_w = 1 scales summaries to same size as main areas.

gaps
: Size of gaps (as binary numeric vector) specifying the width of vertical and horizontal gaps as proportions of area size. Defaults: gaps = c(.02, .00) for p_split = "v" and gaps = c(.00, .02) for p_split = "h".

f_lbl
: Type of label for showing frequency values in 4 main areas, with 6 options:
1. "def": abbreviated names and frequency values;
2. "abb": abbreviated frequency names only (as specified in code);
3. "nam": names only (as specified in lbl_txt = txt);
4. "num": numeric frequency values only (default);
5. "namnum": names (as specified in lbl_txt = txt) and numeric values;
6. "no": no frequency labels (same for f_lbl = NA or NULL).

f_lbl_sep
: Label separator for main frequencies (used for f_lbl = "def" OR "namnum"). Use f_lbl_sep = ":\n" to add a line break between name and numeric value. Default: f_lbl_sep = NA (set to " = " or ":\n" based on f_lbl).

f_lbl_sum
: Type of label for showing frequency values in summary cells, with same 6 options as f_lbl (above). Default: f_lbl_sum = "num": numeric values only.

f_lbl_hd
: Type of label for showing frequency values in header, with same 6 options as f_lbl (above). Default: f_lbl_hd = "nam": names only (as specified in lbl_txt = txt).

f_lwd Line width of areas. Default: f_lwd = 0.

p_lbl Type of label for showing 3 key probability links and values, with 7 options:
1. "def": show links and abbreviated names and probability values;
2. "abb": show links and abbreviated probability names;
3. "nam": show links and probability names (as specified in code);
4. "num": show links and numeric probability values;
5. "namnum": show links with names and numeric probability values;
6. "no": show links with no labels;
7. NA: show no labels or links (same for p_lbl = NULL, default).

arr_c Arrow code for symbols at ends of probability links (as a numeric value -3 <= arr_c <= +6), with the following options:
- -1 to -3: points at one/other/both end/s;
- 0: no symbols;
- +1 to +3: V-arrow at one/other/both end/s;
- +4 to +6: T-arrow at one/other/both end/s.

Default: arr_c = -3 (points at both ends).

col_p Colors of probability links (as vector of 3 colors). Default: col_p = c(grey(.15, .99), "yellow", "yellow"). (Also consider: "black", "cornsilk", "whitesmoke").

brd_dis Distance of probability links from area border (as proportion of area width). Default: brd_dis = .06. Note: Adjust to avoid overlapping labels. Negative values show links outside of main area.

lbl_txt Default label set for text elements. Default: lbl_txt = txt.

main Text label for main plot title. Default: main = txt$scen_lbl.

sub Text label for plot subtitle (on 2nd line). Default: sub = "type" shows information on current plot type.

title_lbl **Deprecated** text label for current plot title. Replaced by main.

cex_lbl Scaling factor for text labels (frequencies and headers). Default: cex_lbl = .90.

cex_p_lbl Scaling factor for text labels (probabilities). Default: cex_p_lbl = cex_lbl - .05.

col_pal Color palette. Default: col_pal = pal.

mar_notes Boolean option for showing margin notes. Default: mar_notes = FALSE.

... Other (graphical) parameters.

Details

plot_area computes probabilities prob and frequencies freq 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.

plot_area generalizes and replaces plot_mosaic. by removing the dependency on the R packages vcd and grid and providing many additional options.

Value

Nothing (NULL).

See Also

plot_mosaic for older (obsolete) version; plot_tab for plotting table (without scaling area dimensions); pal contains current color settings; txt contains current text settings.

Other visualization functions: plot.riskyr(), plot_bar(), plot_crisk(), plot_curve(), plot_fnet(), plot_icons(), plot_mosaic(), plot_plane(), plot_prism(), plot_tab(), plot_tree()

Examples

```
## Basics:
# (1) Using global prob and freq values:
plot_area()  # default area plot,
# same as:
# plot_area(by = "cddc", p_split = "v", area = "sq", scale = "p")

# (2) Providing values:
plot_area(prev = .5, sens = 4/5, spec = 3/5, N = 10)

# (3) Rounding and sampling:
plot_area(N = 100, prev = 1/3, sens = 2/3, spec = 6/7, area = "hr", round = FALSE)
plot_area(N = 100, prev = 1/3, sens = 2/3, spec = 6/7, area = "hr", sample = TRUE, scale = "freq")

# (4) Custom colors and text:
plot_area(prev = .2, sens = 4/5, spec = 3/5, N = 10,
          by = "cddc", p_split = "v", scale = "p",
          main = "Custom text and color:",
          lbl_txt = txt_org, f_lbl = "namnum",
          f_lbl_sep = ":\n", f_lwd = 2, col_pal = pal_rgb)

## Versions:
## by x p_split (= [3 x 2 x 2] = 12 versions):
plot_area(by = "cddc", p_split = "v")  # v01 (see v07)
plot_area(by = "cdac", p_split = "v")  # v02 (see v11)
# plot_area(by = "cddc", p_split = "h")  # v03 (see v05)
# plot_area(by = "cdac", p_split = "h")  # v04 (see v09)

# plot_area(by = "dccd", p_split = "v")  # v05 (is v03 rotated)
plot_area(by = "dcac", p_split = "v")  # v06 (see v12)
# plot_area(by = "dccd", p_split = "h")  # v07 (is v01 rotated)
# plot_area(by = "dcac", p_split = "h")  # v08 (see v10)

# plot_area(by = "accd", p_split = "v")  # v09 (is v04 rotated)
# plot_area(by = "acdc", p_split = "v")  # v10 (is v08 rotated)
# plot_area(by = "accd", p_split = "h")  # v11 (is v02 rotated)
# plot_area(by = "acdc", p_split = "h")  # v12 (is v06 rotated)

## Options:
# area:
```

```
plot_area(area = "sq")  # main area as square (by scaling x-values)
plot_area(area = "no")  # rectangular main area (using full plotting region)

# scale (matters for small N):
plot_area(N = 5, prev = .5, sens = .8, spec = .6,
        by = "cddc", p_split = "v", scale = "p", p_lbl = "def")  # scaled by prob (default)
plot_area(N = 5, prev = .5, sens = .8, spec = .6,
       by = "cddc", p_split = "v", scale = "f", p_lbl = "def")  # scaled by freq (for small N)
plot_area(N = 4, prev = .4, sens = .8, spec = .6,
        by = "cdac", p_split = "h", scale = "p", p_lbl = "def")  # scaled by prob (default)
plot_area(N = 4, prev = .4, sens = .8, spec = .6,
       by = "cdac", p_split = "h", scale = "f", p_lbl = "def")  # scaled by freq (for small N)

# gaps (sensible range: 0--.10):
plot_area(gaps = NA)             # default gaps (based on p_split)
plot_area(gaps = c(0, 0))        # no gaps
# plot_area(gaps = c(.05, .01))  # v_gap > h_gap

# freq labels:
plot_area(f_lbl = "def", f_lbl_sep = " = ")  # default
plot_area(f_lbl = NA)       # NA/NULL: no freq labels (in main area & top/left boxes)
plot_area(f_lbl = "abb")    # abbreviated name (i.e., variable name)
# plot_area(f_lbl = "nam")  # only freq name
# plot_area(f_lbl = "num")  # only freq number
plot_area(f_lbl = "namnum", f_lbl_sep = ":\n", cex_lbl = .75)  # explicit & smaller

# prob labels:
plot_area(p_lbl = NA)      # default: no prob labels, no links
# plot_area(p_lbl = "no")  # show links, but no labels
plot_area(p_lbl = "namnum", cex_lbl = .70)  # explicit & smaller labels

# prob arrows:
plot_area(arr_c = +3, p_lbl = "def", f_lbl = NA)  # V-shape arrows
# plot_area(arr_c = +6, p_lbl = "def", f_lbl = NA)  # T-shape arrows
# plot_area(arr_c = +6, p_lbl = "def", f_lbl = NA,
#           brd_dis = -.02, col_p = c("black")) # adjust arrow type/position

# f_lwd:
plot_area(f_lwd =  3)    # thicker lines
plot_area(f_lwd = .5)    # thinner lines
# plot_area(f_lwd =  0)  # no lines (if f_lwd = 0/NULL/NA: lty = 0)

# sum_w:
# plot_area(sum_w = .10)  # default (showing top and left freq panels & labels)
plot_area(sum_w =  0)   # remove top and left freq panels
plot_area(sum_w =  1,         # top and left freq panels scaled to size of main areas
          col_pal = pal_org)  # custom colors

## Plain and suggested plot versions:
plot_area(sum_w = 0, f_lbl = "abb", p_lbl = NA)  # no compound indicators (on top/left)
plot_area(gap = c(0, 0), sum_w = 0, f_lbl = "num", p_lbl = "num",  # no gaps, numeric labels
          f_lwd = .5, col_pal = pal_bw, main = "Black-and-white")  # b+w print version
# plot_area(f_lbl = "nam", p_lbl = NA, col_pal = pal_mod)  # plot with freq labels
```

```
plot_area(f_lbl = "num", p_lbl = NA, col_pal = pal_rgb)  # no borders around boxes
```

---

plot_bar *Plot bar charts of population frequencies.*

Description

plot_bar draws bar charts that represent the proportions of frequencies in the current population popu as relatives sizes of rectangular areas.

Usage

```
plot_bar(
  prev = num$prev,
  sens = num$sens,
  mirt = NA,
  spec = num$spec,
  fart = NA,
  N = num$N,
  by = "all",
  dir = 1,
  scale = "f",
  round = TRUE,
  sample = FALSE,
  f_lbl = "num",
  f_lwd = 1,
  lty = 0,
  lbl_txt = txt,
  main = txt$scen_lbl,
  sub = "type",
  title_lbl = NULL,
  col_pal = pal,
  mar_notes = FALSE,
  ...
)
```

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.)

by
: A character code specifying the perspective (or the dimension by which the population is split into 2 subsets) with the following options:

1. by = "cd": by condition;
2. by = "dc": by decision;
3. by = "ac": by accuracy;
4. by = "all" combines perspectives (5 bars, default).

dir
: Number of directions in which bars are plotted. Options:

1. dir = 1: uni-directional bars (all up, default);
2. dir = 2: bi-directional bars (up vs. down).

scale
: Scale the heights of bars either by current frequencies (scale = "f") or by exact probabilities (scale = "p"). Default: scale = "f". For large population sizes N and when round = FALSE, both settings yield the same bar heights.

round
: Boolean option specifying whether computed frequencies are to be rounded to integers. Default: round = TRUE.

sample
: Boolean value that determines whether frequency values are sampled from N, given the probability values of prev, sens, and spec. Default: sample = FALSE.

f_lbl
: Type of frequency labels, as character code with the following options:

1. f_lbl = "nam": names;
2. f_lbl = "num": numeric values (default);
3. f_lbl = "abb": abbreviated names;
4. f_lbl = NA/NULL/"no": no labels;
5. f_lbl = "any": abbreviated names and numeric values (abb = num).

f_lwd
: Line width of frequency box (border). Values of NA/NULL/0 set lwd to invisible tiny_lwd <- .001 and lty <- 0 ("blank"). Default: f_lwd = 1.

lty
: Line type of frequency box (border). Values of NA/NULL/0 set lty to lty <- 0. Default: lty = 0 (i.e., no line).

lbl_txt
: Current text information (for labels, titles, etc.). Default: lbl_txt = txt (see init_txt).

main
: Text label for main plot title. Default: main = txt$scen_lbl.

sub
: Text label for plot subtitle (on 2nd line). Default: sub = "type" shows information on current plot type.

title_lbl
: **Deprecated** text label for current plot title. Replaced by main.

col_pal
: Current color palette. Default: col_pal = pal (see init_pal).

mar_notes
: Boolean option for showing margin notes. Default: mar_notes = FALSE.

...
: Other (graphical) parameters (e.g., cex, font, lty, etc.).

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.

By default, plot_bar uses current frequencies (i.e., rounded or not rounded, depending on the value of round) as bar heights, rather than using exact probabilities to scale bar heights (i.e., default scaling is scale = "f"). Using the option scale = "p" scales bar heights by probabilities (e.g., showing bars for non-natural frequencies even when frequencies are rounded). When round = FALSE, bar heights for scale = "f" and for scale = "p" are identical.

The distinction between scale = "f" and scale = "p" matters mostly for small populations sizes N (e.g., when N < 100). For rounded and small frequency values (e.g., freq < 10) switching from scale = "f" to scale = "p" yields different plots.

plot_bar contrasts compound frequencies along 1 dimension (height). See plot_mosaic for 2-dimensional visualizations (as areas) and various box) options in plot_tree and plot_fnet for related functions.

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_area(), plot_crisk(), plot_curve(), plot_fnet(), plot_icons(), plot_mosaic(), plot_plane(), plot_prism(), plot_tab(), plot_tree()

Examples

```
# Basics:
# (1) Using global prob and freq values:
plot_bar()

# (2) Providing values:
plot_bar(prev = .33, sens = .75, spec = .66, main = "Test 1")
plot_bar(N = 1000, prev = .33, sens = .75, spec = .60, main = "Test 2")  # by "all" (default)

# (3) Rounding and sampling:
plot_bar(N = 100, prev = 1/3, sens = 2/3, spec = 6/7, area = "hr", round = FALSE)
plot_bar(N = 100, prev = 1/3, sens = 2/3, spec = 6/7, area = "hr", sample = TRUE, scale = "freq")

# Perspectives (by):
# plot_bar(N = 1000, prev = .33, sens = .75, spec = .60, by = "cd",
#          main = "Test 3a")  # by condition
plot_bar(N = 1000, prev = .33, sens = .75, spec = .60, by = "cd", dir = 2,
         main = "Test 3b", f_lbl = "num")  # bi-directional

# plot_bar(N = 1000, prev = .33, sens = .75, spec = .60, by = "dc",
#          main = "Test 4a")  # by decision
plot_bar(N = 1000, prev = .33, sens = .75, spec = .60, by = "dc", dir = 2,
         main = "Test 4b", f_lbl = "num")  # bi-directional
```

```
# plot_bar(N = 1000, prev = .33, sens = .75, spec = .60, by = "ac",
#          main = "Test 5a")  # by accuracy
plot_bar(N = 1000, prev = .33, sens = .75, spec = .60, by = "ac", dir = 2,
         main = "Test 5b", f_lbl = "num")  # bi-directional

# Customize colors and text:
plot_bar(dir = 1, f_lbl = "num", col_pal = pal_org)
# plot_bar(dir = 2, f_lbl = "nam", col_pal = pal_bw)

# Frequency labels (f_lbl):
# plot_bar(f_lbl = "def")  # default labels: name = num
plot_bar(f_lbl = "nam")  # name only
plot_bar(f_lbl = "num")  # numeric value only
# plot_bar(f_lbl = "abb")  # abbreviated name
# plot_bar(f_lbl = NA)     # no labels (NA/NULL/"no")

# Scaling and rounding effects:
plot_bar(N = 3, prev = .1, sens = .7, spec = .6, dir = 2,
         scale = "f", round = TRUE,
         main = "Rounding (1)") # => Scale by freq and round freq.
plot_bar(N = 3, prev = .1, sens = .7, spec = .6, dir = 2,
         scale = "p", round = TRUE,
         main = "Rounding (2)") # => Scale by prob and round freq.
plot_bar(N = 3, prev = .1, sens = .7, spec = .6, dir = 2,
         scale = "f", round = FALSE,
         main = "Rounding (3)") # => Scale by freq and do NOT round freq.
plot_bar(N = 3, prev = .1, sens = .7, spec = .6, dir = 2,
         scale = "p", round = FALSE,
         main = "Rounding (4)") # => Scale by prob and do NOT round freq.
```

---

plot_crisk *Plot a cumulative risk curve.*

Description

plot_crisk creates visualizations of cumulative risks.

Usage

```
plot_crisk(
  x,
  y = NULL,
  x_from = NA,
  x_to = NA,
  fit_curve = FALSE,
  show_pas = FALSE,
  show_rem = FALSE,
  show_pop = FALSE,
```

```
  show_aux = FALSE,
  show_num = FALSE,
  show_inc = FALSE,
  show_grid = FALSE,
  col_pal = pal_crisk,
  arr_c = -3,
  main = txt$scen_lbl,
  sub = "type",
  title_lbl = NULL,
  x_lbl = "Age (in years)",
  y_lbl = "Population risk",
  y2_lbl = "",
  mar_notes = FALSE,
  ...
)
```

Arguments

`x`	Data or values of an x-dimension on which risk is expressed (required). If `x` but not `y` is provided, `xy.coords` from **grDevices** is used to determine `x`- and `y`-values.
`y`	Values of cumulative risks on a y-dimension (optional, if `x` is an appropriate structure), as monotonically increasing percentage values (ranging from 0 to 100). Default: `y = NULL`.
`x_from`	Start value of risk increment. Default: `x_from = NA`.
`x_to`	End value of risk increment. Default: `x_to = NA`.
`fit_curve`	Boolean: Fit a curve to `x`-`y`-data? Default: `fit_curve = FALSE`.
`show_pas`	Boolean: Show past/passed risk? Default: `show_pas = FALSE`.
`show_rem`	Boolean: Show remaining risk? Default: `show_rem = FALSE`.
`show_pop`	Boolean: Show population partitions? Default: `show_pop = FALSE`.
`show_aux`	Boolean: Show auxiliary elements (i.e., explanatory lines, points, and labels)? Default: `show_aux = FALSE`.
`show_num`	Boolean: Show numeric values, provided that `show_aux = TRUE`. Default: `show_num = FALSE`.
`show_inc`	Boolean: Show risk increments? Default: `show_inc = FALSE`.
`show_grid`	Boolean: Show grid lines? Default: `show_grid = FALSE`.
`col_pal`	Color palette (as a named vector). Default: `col_pal = pal_crisk`.
`arr_c`	Arrow code for symbols at ends of population links (as a numeric value `-3 <= arr_c <= +6`), with the following options: • `-1` to `-3`: points at one/other/both end/s; • `0`: no symbols; • `+1` to `+3`: V-arrow at one/other/both end/s; • `+4` to `+6`: T-arrow at one/other/both end/s. Default: `arr_c = -3` (points at both ends).

main	Text label for main plot title. Default: main = txt$scen_lbl.
sub	Text label for plot subtitle (on 2nd line). Default: sub = "type" shows information on current plot type.
title_lbl	**Deprecated** text label for current plot title. Replaced by main.
x_lbl	Text label of x-axis (at bottom). Default: x_lbl = "Age (in years)".
y_lbl	Text label of y-axis (on left). Default: y_lbl = "Population risk".
y2_lbl	Text label of 2nd y-axis (on right). Default: y2_lbl = "" (formerly "Remaining risk").
mar_notes	Boolean option for showing margin notes. Default: mar_notes = FALSE.
...	Other (graphical) parameters.

Details

plot_crisk assumes data inputs x and y that correspond to each other so that y is a (monotonically increasing) probability density function (over cumulative risk amounts represented by y as a function of x).

Inputs to x and y must typically be of the same length. If x but not y is provided, xy.coords from **grDevices** is used to determine x- and y-values.

The risk events quantified by the cumulative risk values in y are assumed to be uni-directional, non-reversible, and expressed as percentages (ranging from 0 to 100). Thus, an element in the population can only switch its status once (from 'unaffected' to 'affected' by the risk factor).

A cumulative risk increment is computed for an interval ranging from x_from to x_to. If risk values for x_from or x_to are not provided (i.e., in x and y), a curve is fitted to predict y by x (by fit_curve = TRUE).

Note that naive interpretations allow for both overestimation (e.g., reading off population values) and underestimation (e.g., reading off future risk increases without re-scaling to remaining population).

For instructional purposes, plot_crisk provides options for showing/hiding various elements required for computing or comprehending cumulative risk increments.

Color information is based on a vector with named colors col_pal = pal_crisk.

Value

Nothing (NULL).

See Also

pal_crisk corresponding color palette.

Other visualization functions: plot.riskyr(), plot_area(), plot_bar(), plot_curve(), plot_fnet(), plot_icons(), plot_mosaic(), plot_plane(), plot_prism(), plot_tab(), plot_tree()

Examples

```
# Data:
x <- seq(0, 100, by = 10)
y <- c(0, 0, 0, 8, 24, 50, 70, 80, 83, 85, 85)

# Basic versions:
plot_crisk(x, y)  # using data provided
plot_crisk(x, y, x_from = 40)  # use and mark 1 provided point
plot_crisk(x, y, x_from = 44)  # use and mark 1 predicted point
plot_crisk(x, y, x_from = 40, x_to = 60)  # use 2 provided points
plot_crisk(x, y, x_from = 44, x_to = 64)  # use 2 predicted points
plot_crisk(x, y, fit_curve = TRUE)  # fitting curve to provided data

# Training versions:
plot_crisk(x, y, 44, 64, show_pas = TRUE)  # past/passed risk only
plot_crisk(x, y, 44, 64, show_rem = TRUE)  # remaining risk only
plot_crisk(x, y, 44, 64, show_pas = TRUE, show_rem = TRUE) # both risks
plot_crisk(x, y, 44, 64, show_aux = TRUE)  # auxiliary lines + axis
plot_crisk(x, y, 44, 64, show_aux = TRUE, show_pop = TRUE)  # + population parts
plot_crisk(x, y, 44, 64, show_aux = TRUE, show_num = TRUE)  # + numeric values
plot_crisk(x, y, 44, 85, show_aux = TRUE, show_pop = TRUE, show_num = TRUE) # + aux/pop/num

# Note: Showing ALL is likely to overplot/overwhelm:
plot_crisk(x, y, x_from = 47, x_to = 67, fit_curve = TRUE,
           main = "The main title", sub = "Some subtitle",
           show_pas = TRUE, show_rem = TRUE, show_aux = TRUE, show_pop = TRUE,
           show_num = TRUE, show_inc = TRUE, show_grid = TRUE, mar_notes = TRUE)

# Small x- and y-values and linear increases:
plot_crisk(x = 2:10, y = seq(12, 28, by = 2), x_from = 4.5, x_to = 8.5,
           show_pas = TRUE, show_rem = TRUE, show_aux = TRUE, show_pop = TRUE,
           show_num = TRUE, show_inc = TRUE)
```

`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$spec,
  fart = NA,
  what = c("prev", "PPV", "NPV"),
  p_lbl = "def",
  p_lwd = 2,
  what_col = pal,
  uc = 0,
  show_points = TRUE,
  log_scale = FALSE,
  prev_range = c(0, 1),
  lbl_txt = txt,
  main = txt$scen_lbl,
  sub = "type",
  title_lbl = NULL,
  cex_lbl = 0.85,
  col_pal = pal,
  mar_notes = FALSE,
  ...
)
```

Arguments

prev The condition's prevalence `prev` (i.e., the probability of condition being `TRUE`). If `prev = NA`, the curves in `what` are plotted without points (i.e., `show_points = FALSE`).

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 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")`.

p_lbl Type of label for shown probability values, with the following options:

1. `"abb"`: show abbreviated probability names;
2. `"def"`: show abbreviated probability names and values (default);
3. `"nam"`: show only probability names (as specified in code);
4. `"num"`: show only numeric probability values;

5. "namnum": show names and numeric probability values;
6. "no": hide labels (same for p_lbl = NA or NULL).

p_lwd
: Line widths of probability curves plotted. Default: p_lwd = 2.

what_col
: Vector of colors corresponding to the elements specified in what. Default: what_col = pal.

uc
: Uncertainty range, given as a percentage of the current prev, sens, and spec values (added in both directions). Default: uc = .00 (i.e., no uncertainty). Plausible ranges are 0 < uc < .25.

show_points
: Boolean value 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.

prev_range
: Range (minimum and maximum) of prev values on x-axis (i.e., values in c(0, 1) range). Default: prev_range = c(0, 1).

lbl_txt
: Labels and text elements. Default: lbl_txt = txt.

main
: Text label for main plot title. Default: main = txt$scen_lbl.

sub
: Text label for plot subtitle (on 2nd line). Default: sub = "type" shows information on current plot type.

title_lbl
: **Deprecated** text label for current plot title. Replaced by main.

cex_lbl
: Scaling factor for the size of text labels (e.g., on axes, legend, margin text). Default: cex_lbl = .85.

col_pal
: Color palette (if what_col is unspecified). Default: col_pal = pal.

mar_notes
: Boolean value for showing margin notes. Default: mar_notes = FALSE.

...
: Other (graphical) parameters.

Details

If no prevalence value is provided (i.e., prev = NA), the desired probability curves are plotted without showing specific points (i.e., show_points = FALSE).

Note that a population size N is not needed for computing probability information prob. (An arbitrary value can be used when computing frequency information freq from current probabilities prob.)

plot_curve is a generalization of plot_PV (see legacy code) that allows plotting 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_area(), plot_bar(), plot_crisk(), plot_fnet(), plot_icons(), plot_mosaic(), plot_plane(), plot_prism(), plot_tab(), plot_tree()

Examples

```
# Basics:
plot_curve()  # default curve plot, same as:
# plot_curve(what = c("prev", "PPV", "NPV"), uc = 0, prev_range = c(0, 1))

# Showing no/multiple prev values/points and uncertainty ranges:
plot_curve(prev = NA)  # default curves without prev value (and point) shown
plot_curve(show_points = FALSE, uc = .10)  # curves w/o points, 10% uncertainty range
plot_curve(prev = c(.10, .33, .75))  # 3 prev values, with numeric point labels
plot_curve(prev = c(.10, .33, .75), p_lbl = "no", uc = .10) # 3 prev, no labels, 10% uc

# Provide local parameters and select curves:
plot_curve(prev = .2, sens = .8, spec = .6, what = c("PPV", "NPV", "acc"), uc = .2)

# Selecting curves: what = ("prev", "PPV", "NPV", "ppod", "acc") = "all"
plot_curve(prev = .3, sens = .9, spec = .8, what = "all")  # all 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 ppod

# Visualizing uncertainty (uc as percentage range):
plot_curve(prev = .2, sens = .9, spec = .8, what = "all",
           uc = .10)  # all with a 10% uncertainty range
# plot_curve(prev = .3, sens = .9, spec = .8, what = c("prev", "PPV", "NPV"),
#            uc = .05)  # prev, PPV and NPV with a 5% uncertainty range

# X-axis on 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
# Several small prev values:
plot_curve(prev = c(.00001, .0001, .001, .01, .05),
           sens = .9, spec = .8, log_scale = TRUE)

# Zooming in by setting prev_range (of prevalence values):
plot_curve(prev = c(.25, .33, .40), prev_range = c(.20, .50),
           what = "all", uc = .05)

# Probability labels:
plot_curve(p_lbl = "abb", what = "all")     # abbreviated names
plot_curve(p_lbl = "nam", what = "all")     # names only
plot_curve(p_lbl = "num", what = "all")     # numeric values only
plot_curve(p_lbl = "namnum", what = "all")  # names and values

# Text and color settings:
plot_curve(main = "Tiny text labels", p_lbl = "namnum", cex_lbl = .60)
plot_curve(main = "Specific colors", what = "all",
           uc = .1, what_col = c("grey", "red3", "green3", "blue3", "gold"))
plot_curve(main = "Black-and-white print version",
           what = "all", col_pal = pal_bwp)
```

plot_fnet *Plot frequency net diagram of frequencies and probabilities.*

Description

plot_fnet plots a frequency net of 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$spec,
  fart = NA,
  N = num$N,
  by = "cddc",
  area = "no",
  scale = "p",
  round = TRUE,
  sample = FALSE,
  f_lbl = "num",
  f_lbl_sep = NA,
  f_lwd = 0,
  p_lwd = 1,
  p_scale = FALSE,
  p_lbl = "mix",
  arr_c = NA,
  joint_p = TRUE,
  lbl_txt = txt,
  main = txt$scen_lbl,
  sub = "type",
  title_lbl = NULL,
  cex_lbl = 0.9,
  cex_p_lbl = NA,
  col_pal = pal,
  mar_notes = FALSE,
  ...
)
```

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. Note that a population size `N` is not needed for computing current probability information `prob`, but is needed for computing frequency information `freq` from current probabilities `prob`.

by
: A character code specifying 1 or 2 perspective(s) that split(s) the population into 2 subsets. Specifying 1 perspective plots a frequency tree (single tree) with 3 options:

  1. `"cd"`: by condition only;
  2. `"dc"`: by decision only;
  3. `"ac"`: by accuracy only.

  Specifying 2 perspectives plots a frequency prism (double tree) with 6 options:

  1. `"cddc"`: by condition (cd) and by decision (dc) (default);
  2. `"cdac"`: by condition (cd) and by accuracy (ac);
  3. `"dccd"`: by decision (dc) and by condition (cd);
  4. `"dcac"`: by decision (dc) and by accuracy (ac);
  5. `"accd"`: by accuracy (ac) and by condition (cd);
  6. `"acdc"`: by accuracy (ac) and by decision (dc).

area
: A character code specifying the shapes of the frequency boxes, with 2 options:

  1. `"no"`: rectangular frequency boxes, not scaled (default);
  2. `"sq"`: frequency boxes are squares (scaled relative to N).

scale
: Scale probabilities and corresponding area dimensions either by exact probability or by (rounded or non-rounded) frequency, with 2 options:

  1. `"p"`: scale main area dimensions by exact probability (default);
  2. `"f"`: re-compute probabilities from (rounded or non-rounded) frequencies and scale main area dimensions by their frequency.

  Note: `scale` setting matters for the display of probability values and for area plots with small population sizes `N` when `round = TRUE`.

round
: Boolean option specifying whether computed frequencies are rounded to integers. Default: `round = TRUE`.

sample
: Boolean value that determines whether frequency values are sampled from `N`, given the probability values of `prev`, `sens`, and `spec`. Default: `sample = FALSE`.

f_lbl
: Type of label for showing frequency values in 4 main areas, with 6 options:

1. "def": abbreviated names and frequency values;
2. "abb": abbreviated frequency names only (as specified in code);
3. "nam": names only (as specified in lbl_txt = txt);
4. "num": numeric frequency values only (default);
5. "namnum": names (as specified in lbl_txt = txt) and numeric values;
6. "no": no frequency labels (same for f_lbl = NA or NULL).

f_lbl_sep
: Label separator for main frequencies (used for f_lbl = "def" OR "namnum"). Use f_lbl_sep = ":\n" to add a line break between name and numeric value. Default: f_lbl_sep = NA (set to " = " or ":\n" based on f_lbl).

f_lwd
: Line width of areas. Default: f_lwd = 0.

p_lwd
: Line width of probability links. Default: p_lwd = 1, but consider increasing when setting p_scale = TRUE.

p_scale
: Boolean option for scaling current widths of probability links (as set by p_lwd) by the current probability values. Default: p_scale = FALSE.

p_lbl
: Type of label for showing probability links and values, with many options:

1. "abb": show links and abbreviated probability names;
2. "def": show links and abbreviated probability names and values;
3. "min": show links and minimum (prominent) probability names;
4. "mix": show links and prominent probability names and all values (default);
5. "nam": show links and probability names (as specified in code);
6. "num": show links and numeric probability values;
7. "namnum": show links with names and numeric probability values;
8. "no": show links with no labels (same for p_lbl = NA or NULL).

arr_c
: Arrow code for symbols at ends of probability links (as a numeric value -3 <= arr_c <= +6), with the following options:

- -1 to -3: points at one/other/both end/s;
- 0: no symbols;
- +1 to +3: V-arrow at one/other/both end/s;
- +4 to +6: T-arrow at one/other/both end/s.

Default: arr_c = NA, but adjusted by area.

joint_p
: Boolean options for showing links to joint probabilities (i.e., diagonals from N in center to joint frequencies in 4 corners). Default: joint_p = TRUE.

lbl_txt
: Default label set for text elements. Default: lbl_txt = txt.

main
: Text label for main plot title. Default: main = txt$scen_lbl.

sub
: Text label for plot subtitle (on 2nd line). Default: sub = "type" shows information on current plot type.

title_lbl
: **Deprecated** text label for current plot title. Replaced by main.

cex_lbl
: Scaling factor for text labels (frequencies and headers). Default: cex_lbl = .90.

cex_p_lbl	Scaling factor for text labels (probabilities). Default: cex_p_lbl = cex_lbl - .05.
col_pal	Color palette. Default: col_pal = pal.
mar_notes	Boolean option for showing margin notes. Default: mar_notes = FALSE.
...	Other (graphical) parameters.

Details

plot_fnet shows frequencies as nodes and probabilities as links (like trees and double trees generated by plot_prism), but combines elements from 2x2 tables (see plot_tab) and tree diagrams.

Similar to other 2D-visualizations (e.g., , plot_area, plot_prism and plot_tab), the frequency net selects and combines two perspectives (e.g., by = "cddc"). However, the frequency net is similar to a 2x2 table insofar as its perspectives (shown by arranging marginal frequencies in a vertical vs. horizontal fashion) do not suggest an order or dependency (in contrast to trees or mosaic plots). Additionally, the frequency net allows showing 3 kinds of (marginal, conditional, and joint) probabilities.

See the article by Binder K, Krauss S and Wiesner P (2020). A new visualization for probabilistic situations containing two binary events: The frequency net. Frontiers in Psychology, 11, 750. doi: 10.3389/fpsyg.2020.00750 for analysis and details.

Value

Nothing (NULL).

Source

Binder, K., Krauss, S., and Wiesner, P. (2020). A new visualization for probabilistic situations containing two binary events: The frequency net. Frontiers in Psychology, 11, 750. doi: 10.3389/fpsyg.2020.00750

See Also

plot_prism for plotting prism plot (double tree); plot_area for plotting mosaic plot (scaling area dimensions); plot_bar for plotting frequencies as vertical bars; plot_tab for plotting table (without scaling area dimensions); pal contains current color settings; txt contains current text settings.

Other visualization functions: plot.riskyr(), plot_area(), plot_bar(), plot_crisk(), plot_curve(), plot_icons(), plot_mosaic(), plot_plane(), plot_prism(), plot_tab(), plot_tree()

Examples

```
# (1) Basics: ----
# A. Using global prob and freq values:
plot_fnet()  # default frequency net, same as:
# plot_fnet(by = "cddc", area = "no", scale = "p",
#           f_lbl = "num", f_lwd = 0, cex_lbl = .90,
#           p_lbl = "mix", arr_c = -2, cex_p_lbl = NA)

# B. Providing values:
```

```
plot_fnet(N = 10000, prev = .02, sens = .8, spec = .9)  # Binder et al. (2020, Fig. 3)

# C. Rounding and sampling:
plot_fnet(N = 100, prev = 1/3, sens = 2/3, spec = 6/7, area = "sq", round = FALSE)
plot_fnet(N = 100, prev = 1/3, sens = 2/3, spec = 6/7, area = "sq", sample = TRUE, scale = "freq")

# Variants:
plot_fnet(N = 10000, prev = .02, sens = .8, spec = .9, by = "cdac")
plot_fnet(N = 10000, prev = .02, sens = .8, spec = .9, by = "dccd")
# plot_fnet(N = 10000, prev = .02, sens = .8, spec = .9, by = "dcac")
# plot_fnet(N = 10000, prev = .02, sens = .8, spec = .9, by = "accd")
# plot_fnet(N = 10000, prev = .02, sens = .8, spec = .9, by = "acdc")

# Trees (only 1 dimension):
plot_fnet(N = 10000, prev = .02, sens = .8, spec = .9, by = "cd")
# plot_fnet(N = 10000, prev = .02, sens = .8, spec = .9, by = "dc")
# plot_fnet(N = 10000, prev = .02, sens = .8, spec = .9, by = "ac")

# Area and margin notes:
plot_fnet(N = 10, prev = 1/4, sens = 3/5, spec = 2/5, area = "sq", mar_notes = TRUE)


# (2) Use case (highlight horizontal vs. vertical perspectives: ----
# Define scenario:
mammo <- riskyr(N = 10000, prev = .01, sens = .80, fart = .096,
                scen_lbl = "Mammography screening", N_lbl = "Women",
                cond_lbl = "Breast cancer", dec_lbl = "Test result",
                cond_true_lbl = "Cancer (C+)", cond_false_lbl = "no Cancer (C-)",
                dec_pos_lbl = "positive (T+)", dec_neg_lbl = "negative (T-)",
                hi_lbl = "C+ and T+", mi_lbl = "C+ and T-",
                fa_lbl = "C- and T+", cr_lbl = "C- and T-")
# Colors:
my_non <- "grey95"
my_red <- "orange1"
my_blu <- "skyblue1"

# A. Emphasize condition perspective (rows):
my_col_1 <- init_pal(N_col = my_non,
                     cond_true_col = my_blu, cond_false_col = my_red,
                     dec_pos_col = my_non, dec_neg_col = my_non,
                     hi_col = my_blu, mi_col = my_blu,
                     fa_col = my_red, cr_col = my_red)
plot(mammo, type = "fnet", col_pal = my_col_1,
     f_lbl = "namnum", f_lwd = 2, p_lbl = "no", arr_c = 0)

# B. Emphasize decision perspective (columns):
my_col_2 <- init_pal(N_col = my_non,
                     cond_true_col = my_non, cond_false_col = my_non,
                     dec_pos_col = my_red, dec_neg_col = my_blu,
                     hi_col = my_red, mi_col = my_blu,
                     fa_col = my_red, cr_col = my_blu)
plot(mammo, type = "fnet", col_pal = my_col_2,
     f_lbl = "namnum", f_lwd = 2, p_lbl = "no", arr_c = 0)
```

```
# (3) Custom color and text settings: ----
plot_fnet(col_pal = pal_bw, f_lwd = .5, p_lwd = .5, lty = 2, # custom fbox color, prob links,
          font = 3, cex_p_lbl = .75)                         # and text labels

plot_fnet(N = 7, prev = 1/2, sens = 3/5, spec = 4/5, round = FALSE,
          by = "cdac", lbl_txt = txt_org, f_lbl = "namnum", f_lbl_sep = ":\n",
          f_lwd = 1, col_pal = pal_rgb)  # custom colors

# plot_fnet(N = 5, prev = 1/2, sens = .8, spec = .5, scale = "p",   # Note scale!
#           by = "cddc", area = "hr", col_pal = pal_bw, f_lwd = 1)  # custom colors

plot_fnet(N = 3, prev = .50, sens = .50, spec = .50, scale = "p",            # Note scale!
        area = "sq", lbl_txt = txt_org, f_lbl = "namnum", f_lbl_sep = ":\n", # custom text
         col_pal = pal_kn, f_lwd = .5)                                      # custom colors

# (4) Other options: ----
plot_fnet(N = 4, prev = .2, sens = .7, spec = .8,
          area = "sq", scale = "p")  # areas scaled by prob (matters for small N)
# plot_fnet(N = 4, prev = .2, sens = .7, spec = .8,
#           area = "sq", scale = "f")  # areas scaled by (rounded or non-rounded) freq

## Frequency boxes (f_lbl):
# plot_fnet(f_lbl = NA)       # no freq labels
# plot_fnet(f_lbl = "abb")    # abbreviated freq names (variable names)
plot_fnet(f_lbl = "nam")      # only freq names
plot_fnet(f_lbl = "num")      # only numeric freq values (default)
# plot_fnet(f_lbl = "namnum") # names and numeric freq values
plot_fnet(f_lbl = "namnum", cex_lbl = .75)  # smaller freq labels
# plot_fnet(f_lbl = "def")   # informative default: short name and numeric value (abb = num)

# f_lwd:
# plot_fnet(f_lwd =  1)  # basic lines
# plot_fnet(f_lwd =  0)  # no lines (default), set to tiny_lwd = .001, lty = 0 (same if NA/NULL)
# plot_fnet(f_lwd = .5)  # thinner lines
plot_fnet(f_lwd =  3)    # thicker lines

## Probability links (p_lbl, p_lwd, p_scale):
# plot_fnet(p_lbl = NA)       # no prob labels (NA/NULL/"none")
plot_fnet(p_lbl = "mix")      # abbreviated names with numeric values (abb = num)
# plot_fnet(p_lbl = "min")    # minimal names (of key probabilities)
# plot_fnet(p_lbl = "nam")    # only prob names
plot_fnet(p_lbl = "num")      # only numeric prob values
# plot_fnet(p_lbl = "namnum") # names and numeric prob values

plot_fnet(p_lwd = 6, p_scale = TRUE)
plot_fnet(area = "sq", f_lbl = "num", p_lbl = NA, col_pal = pal_bw, p_lwd = 6, p_scale = TRUE)

# arr_c:
# plot_fnet(arr_c =  0)  # acc_c = 0: no arrows
# plot_fnet(arr_c = -3)  # arr_c = -1 to -3: points at both ends
# plot_fnet(arr_c = -2)  # point at far end
```

```
plot_fnet(arr_c = +2)    # crr_c = 1-3: V-shape arrows at far end

plot_fnet(by = "cd", joint_p = FALSE)      # tree without joint probability links
# plot_fnet(by = "cddc", joint_p = FALSE)  # fnet ...

## Plain plot versions:
plot_fnet(area = "no", f_lbl = "def", p_lbl = "num", col_pal = pal_mod, f_lwd = 1,
          main = "", mar_notes = FALSE)  # remove titles and margin notes
plot_fnet(area = "no", f_lbl = "nam", p_lbl = "min", col_pal = pal_rgb)

plot_fnet(area = "sq", f_lbl = "nam", p_lbl = "num", col_pal = pal_rgb)
# plot_fnet(area = "sq", f_lbl = "def", f_lbl_sep = ":\n", p_lbl = NA, f_lwd = 1, col_pal = pal_kn)

## Suggested combinations:
# plot_fnet(f_lbl = "nam", p_lbl = "mix")  # basic plot
plot_fnet(f_lbl = "namnum", p_lbl = "num", cex_lbl = .80, cex_p_lbl = .75)
# plot_fnet(area = "no", f_lbl = "def", p_lbl = "abb",           # def/abb labels
#           f_lwd = .8, p_lwd = .8, lty = 2, col_pal = pal_bwp)  # black-&-white

# plot_fnet(area = "sq", f_lbl = "nam", p_lbl = "abb", lbl_txt = txt_TF, col_pal = pal_bw)
plot_fnet(area = "sq", f_lbl = "num", p_lbl = "num", f_lwd = 1, col_pal = pal_rgb)
plot_fnet(area = "sq", f_lbl = "nam", p_lbl = "num", f_lwd = .5, col_pal = pal_rgb)
```

---

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,
  sample = FALSE,
  arr_type = "array",
  by = "all",
  ident_order = c("hi", "mi", "fa", "cr"),
  icon_types = 22,
  icon_size = NULL,
```

```
    icon_brd_lwd = 1.5,
    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",
    lbl_txt = txt,
    main = txt$scen_lbl,
    sub = "type",
    title_lbl = NULL,
    cex_lbl = 0.9,
    col_pal = pal,
    transparency = 0.5,
    mar_notes = FALSE,
    ...
)
```

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.

`sample` Boolean value that determines whether frequency values are sampled from `N`, given the probability values of `prev`, `sens`, and `spec`. Default: `sample = FALSE`.

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

1. `arr_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. `arr_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. `arr_type = "mosaic"`: Icons are ordered like in a mosaic plot. The area size displays the relative proportions of their frequencies.
4. `arr_type = "fillequal"`: Icons are positioned into equally sized blocks. Thus, their density reflects the relative proportions of their frequencies.
5. `arr_type = "fillleft"`: Icons are randomly filled from the left.
6. `arr_type = "filltop"`: Icons are randomly filled from the top.
7. `arr_type = "scatter"`: Icons are randomly scattered into the plot.

`by` A character code specifying a perspective to split the population into subsets, with 4 options:

1. `"all"`: by condition (cd) and by decision (dc): `hi`, `mi`, `fa`, `cr` cases (default);
2. `"cd"`: by condition (cd) only: `cond_true` vs. `cond_false` cases;
3. `"dc"`: by decision (dc) only: `dec_pos` vs. `dec_neg` cases;
4. `"ac"`: by accuracy (ac) only: `dec_cor` vs. `dec_err` cases.

`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_types` specifies the appearance of the icons as a vector. Default: `icon_types = 11` (i.e., squares with border). Accepts values from 1 to 25 (see `?points`).

`icon_size` specifies the size of the icons via `cex` Default: `icon_size = NULL` for automatic calculation.

`icon_brd_lwd` specifies the border width of icons (if applicable). Default: `icon_brd_lwd = 1.5`. Set to `NA` for no border.

`block_d` The distance between blocks. Default: `block_d = NULL` for automatic calculation; (does not apply to "filleft", "filltop", and "scatter")

`border_d` The distance of icons to the border. Default: `border_d = 0.1`.

Additional options for controlling the arrangement of arrays (for `arr_type = "array"` and `"shuffledarray"`):

`block_size_row` specifies how many icons should be in each block row. Default: `block_size_row = 10`.

`block_size_col` specifies how many icons should be in each block column. Default: `block_size_col = 10`.

`nblocks_row` Number of blocks per row. Default: `nblocks_row = NULL` for automatic calculation.

`nblocks_col` Number of blocks per column. Default: `nblocks_col = NULL` for automatic calculation.

`fill_array` specifies how the blocks are filled into the array. Options: `fill_array = "left"` (default) vs. `"top"`.

fill_blocks — specifies how icons within blocks are filled. Options: fill_blocks = "rowwise" (default) and "colwise".

Generic text and color options:

lbl_txt — Default label set for text elements. Default: lbl_txt = txt.

main — Text label for main plot title. Default: main = txt$scen_lbl.

sub — Text label for plot subtitle (on 2nd line). Default: sub = "type" shows information on current plot type.

title_lbl — **Deprecated** text label for current plot title. Replaced by main.

cex_lbl — Scaling factor for text labels. Default: cex_lbl = .90.

col_pal — Color palette. Default: col_pal = pal.

transparency — Specifies the transparency for overlapping icons (not for arr_type = "array" and "shuffledarray").

mar_notes — Boolean option for showing margin notes. Default: mar_notes = FALSE.

... — Other (graphical) parameters.

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.

Value

Nothing (NULL).

See Also

Other visualization functions: plot.riskyr(), plot_area(), plot_bar(), plot_crisk(), plot_curve(), plot_fnet(), plot_mosaic(), plot_plane(), plot_prism(), plot_tab(), plot_tree()

Examples

```
# Basics:
plot_icons(N = 1000)  # icon array with default settings (arr_type = "array")
plot_icons(arr_type = "shuffledarray", N = 1000)  # icon array with shuffled IDs

# Sampling:
plot_icons(N = 1000, prev = 1/2, sens = 2/3, spec = 6/7, sample = TRUE)

# array types:
plot_icons(arr_type = "mosaic",    N = 1000)  # areas as in mosaic plot
plot_icons(arr_type = "fillequal", N = 1000)  # areas of equal size (probability as density)
plot_icons(arr_type = "fillleft",  N = 1000)  # icons filled from left to right (in columns)
plot_icons(arr_type = "filltop",   N = 1000)  # icons filled from top to bottom (in rows)
plot_icons(arr_type = "scatter",   N = 1000)  # icons randomly scattered

# by:
plot_icons(N = 1000, by = "all")  # hi, mi, fa, cr (TP, FN, FP, TN) cases
```

```
plot_icons(N = 1000, by = "cd", main = "Cases by condition")  # (hi + mi) vs. (fa + cr)
plot_icons(N = 1000, by = "dc", main = "Cases by decision")   # (hi + fa) vs. (mi + cr)
plot_icons(N = 1000, by = "ac", main = "Cases by accuracy")   # (hi + cr) vs. (fa + mi)

# Custom icon types and colors:
plot_icons(N = 800, arr_type = "array", icon_types = c(21, 22, 23, 24),
           block_d = 0.5, border_d = 0.5, col_pal = pal_vir)

plot_icons(N = 800, arr_type = "shuffledarray", icon_types = c(21, 23, 24, 22),
           block_d = 0.5, border_d = 0.5)

plot_icons(N = 800, arr_type = "fillequal", icon_types = c(21, 22, 22, 21),
           icon_brd_lwd = .5, cex = 1, cex_lbl = 1.1)

# Text and color options:
plot_icons(N = 1000, prev = .5, sens = .5, spec = .5, arr_type = "shuffledarray",
         main = "My title", sub = NA, lbl_txt = txt_TF, col_pal = pal_vir, mar_notes = TRUE)

plot_icons(N = 1000, prev = .5, sens = .5, spec = .5, arr_type = "shuffledarray",
           main = "Green vs. red", col_pal = pal_rgb, transparency = .5)
```

---

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

Description

`plot_mosaic` drew a mosaic plot that represents the proportions of frequencies in the current population 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,
  by = "cddc",
  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.
sens	The decision's sensitivity sens.
mirt	The decision's miss rate mirt.
spec	The decision's specificity value spec.
fart	The decision's false alarm rate fart.
N	The number of individuals in the population.
by	A character code specifying the perspective (or categories by which the population is split into subsets) with 3 options: 1. "cddc" ... by condition x decision; 2. "dccd" ... by decision x condition; 3. "cdac" ... by condition x accuracy.
show_accu	Option for showing current and exact accuracy metrics accu in the plot.
w_acc	Weighting parameter w used to compute weighted accuracy.
title_lbl	Text label for current plot title.
col_sdt	Colors for cases of 4 essential frequencies. Default: col_sdt = c(pal["hi"], pal["mi"], pal["fa"], pal["cr"]).

Details

plot_mosaic is deprecated – please use plot_area instead.

See Also

plot_area is the new version of this function.

Other visualization functions: plot.riskyr(), plot_area(), plot_bar(), plot_crisk(), plot_curve(), plot_fnet(), plot_icons(), plot_plane(), plot_prism(), plot_tab(), plot_tree()

Examples

```
plot_mosaic()  # plot with default options
```

---

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$spec,
  fart = NA,
  what = "PPV",
  what_col = pal,
  line_col = "grey85",
  sens_range = c(0, 1),
  spec_range = c(0, 1),
  step_size = 0.05,
  show_points = TRUE,
  point_col = "yellow",
  theta = -45,
  phi = 0,
  p_lbl = "def",
  lbl_txt = txt,
  main = txt$scen_lbl,
  sub = "type",
  title_lbl = NULL,
  cex_lbl = 0.85,
  col_pal = pal,
  mar_notes = FALSE,
  ...
)
```

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. If `sens = NA`, then `show_points = FALSE`.
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. If `spec = NA`, then `show_points = FALSE`.
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	Color for surface facets corresponding to the metric specified in `what`. Default: `what_col` uses color corresponding to `what` in current `col_pal`.

line_col	Color for lines between surface facets. Default: line_col = "grey85".
sens_range	Range (minimum and maximum) of sens values on x-axis (i.e., values in c(0, 1) range). Default: sens_range = c(0, 1).
spec_range	Range (minimum and maximum) of spec values on y-axis (i.e., values in c(0, 1) range). Default: spec_range = c(0, 1).
step_size	Sets the granularity of the sens-by-spec grid. (in range .01 <= step_size <= 1). Default: step_size = .05.
show_points	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_points = TRUE.
point_col	Fill color for showing current value on plane. Default: point_col = "yellow".
theta	Horizontal rotation angle (used by persp). Default: theta = -45.
phi	Vertical rotation angle (used by persp). Default: phi = 0.
p_lbl	Type of label for shown probability values, with the following options: 1. "abb": show abbreviated probability names; 2. "def": show abbreviated probability names and values (default); 3. "nam": show only probability names (as specified in code); 4. "num": show only numeric probability values; 5. "namnum": show names and numeric probability values; 6. "no": hide labels (same for p_lbl = NA or NULL).
lbl_txt	Labels and text elements. Default: lbl_txt = txt.
main	Text label for main plot title. Default: main = txt$scen_lbl.
sub	Text label for plot subtitle (on 2nd line). Default: sub = "type" shows information on current plot type.
title_lbl	**Deprecated** text label for current plot title. Replaced by main.
cex_lbl	Scaling factor for the size of text labels (e.g., on axes, legend, margin text). Default: cex_lbl = .85.
col_pal	Color palette (if what_col is unspecified). Default: col_pal = pal.
mar_notes	Boolean value for showing margin notes. Default: mar_notes = FALSE.
...	Other (graphical) parameters.

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_area(), plot_bar(), plot_crisk(), plot_curve(), plot_fnet(), plot_icons(), plot_mosaic(), plot_prism(), plot_tab(), plot_tree()

Examples

```
# Basics:
plot_plane()              # => default plot (what = "PPV")
# same as:
# 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

# Plane with/out points:
# plot_plane(prev = .5, sens = NA, spec = NA, what = "ppv")          # plane with 0 points
plot_plane(prev = .5, sens = c(.2, .5, .8), spec = .6, what = "npv")  # plane with 3 points

# Zooming into sens and spec ranges:
# plot_plane(prev = .02, sens = c(.8, .9), spec = c(.8, .8, .9, .9))  # default ranges
plot_plane(prev = .02, sens = c(.8, .9), spec = c(.8, .8, .9, .9),
           sens_range = c(.7, 1), spec_range = c(.7, 1), step_size = .02)  # zooming in

# Options:
# plot_plane(main = "No point and smaller labels", show_points = FALSE, cex_lbl = .60)

plot_plane(main = "Testing plot colors", what_col = "royalblue4", line_col = "sienna2")
plot_plane(main = "Testing b/w plot", what = "npv", what_col = "white", line_col = "black")
plot_plane(main = "Testing color pal_bwp", col_pal = pal_bwp)

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 = 50)  # => rotated, from above
```

`plot_prism` *Plot prism diagram of frequencies and probabilities.*

Description

`plot_prism` plots a network diagram of 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_prism(
  prev = num$prev,
  sens = num$sens,
  mirt = NA,
  spec = num$spec,
  fart = NA,
  N = num$N,
```

```
  by = "cddc",
  area = "no",
  scale = "p",
  round = TRUE,
  sample = FALSE,
  f_lbl = "num",
  f_lbl_sep = NA,
  f_lwd = 0,
  p_lwd = 1,
  p_scale = FALSE,
  p_lbl = "mix",
  arr_c = NA,
  lbl_txt = txt,
  main = txt$scen_lbl,
  sub = "type",
  title_lbl = NULL,
  cex_lbl = 0.9,
  cex_p_lbl = NA,
  col_pal = pal,
  mar_notes = FALSE,
  ...
)
```

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. Note that a population size `N` is not needed for computing current probability information `prob`, but is needed for computing frequency information `freq` from current probabilities `prob`.

`by` A character code specifying 1 or 2 perspective(s) that split(s) the population into 2 subsets. Specifying 1 perspective plots a frequency tree (single tree) with 3 options:

1. `"cd"`: by condition only;
2. `"dc"`: by decision only;

3. "ac": by accuracy only.

Specifying 2 perspectives plots a frequency prism (double tree) with 6 options:

1. "cddc": by condition (cd) and by decision (dc) (default);
2. "cdac": by condition (cd) and by accuracy (ac);
3. "dccd": by decision (dc) and by condition (cd);
4. "dcac": by decision (dc) and by accuracy (ac);
5. "accd": by accuracy (ac) and by condition (cd);
6. "acdc": by accuracy (ac) and by decision (dc).

area — A character code specifying the shapes of the frequency boxes, with 3 options:

1. "no": rectangular frequency boxes, not scaled (default);
2. "hr": frequency boxes are horizontal rectangles (scaled relative to N).
3. "sq": frequency boxes are squares (scaled relative to N).

scale — Scale probabilities and corresponding node dimensions either by exact probability or by (rounded or non-rounded) frequency, with 2 options:

1. "p": scale node dimensions by exact probability (default);
2. "f": re-compute probabilities from (rounded or non-rounded) frequencies and scale node dimensions by their frequency.

Note: scale setting matters for the display of probability values and for area plots with small population sizes N when round = TRUE.

round — Boolean option specifying whether computed frequencies are rounded to integers. Default: round = TRUE.

sample — Boolean value that determines whether frequency values are sampled from N, given the probability values of prev, sens, and spec. Default: sample = FALSE.

f_lbl — Type of label for showing frequency values in nodes, with 6 options:

1. "def": abbreviated names and frequency values;
2. "abb": abbreviated frequency names only (as specified in code);
3. "nam": names only (as specified in lbl_txt = txt);
4. "num": numeric frequency values only (default);
5. "namnum": names (as specified in lbl_txt = txt) and numeric values;
6. "no": no frequency labels (same for f_lbl = NA or NULL).

f_lbl_sep — Separator for frequency labels (used for f_lbl = "def" OR "namnum"). Use f_lbl_sep = ":\n" to add a line break between name and numeric value. Default: f_lbl_sep = NA (set to " = " or ":\n" based on f_lbl).

f_lwd — Line width of areas. Default: f_lwd = 0.

p_lwd — Line width of probability links. Default: p_lwd = 1, but consider increasing when setting p_scale = TRUE.

p_scale — Boolean option for scaling current widths of probability links (as set by p_lwd) by the current probability values. Default: p_scale = FALSE.

p_lbl — Type of label for showing 3 key probability links and values, with many options:

1. "abb": show links and abbreviated probability names;
2. "def": show links and abbreviated probability names and values;

3. "min": show links and minimum (prominent) probability names;
4. "mix": show links and prominent probability names and all values (default);
5. "nam": show links and probability names (as specified in code);
6. "num": show links and numeric probability values;
7. "namnum": show links with names and numeric probability values;
8. "no": show links with no labels (same for p_lbl = NA or NULL).

arr_c
: Arrow code for symbols at ends of probability links (as a numeric value -3 <= arr_c <= +6), with the following options:
  - -1 to -3: points at one/other/both end/s;
  - 0: no symbols;
  - +1 to +3: V-arrow at one/other/both end/s;
  - +4 to +6: T-arrow at one/other/both end/s.

  Default: arr_c = NA, but adjusted by area.

lbl_txt
: Default label set for text elements. Default: lbl_txt = txt.

main
: Text label for main plot title. Default: main = txt$scen_lbl.

sub
: Text label for plot subtitle (on 2nd line). Default: sub = "type" shows information on current plot type.

title_lbl
: **Deprecated** text label for current plot title. Replaced by main.

cex_lbl
: Scaling factor for text labels (frequencies and headers). Default: cex_lbl = .90.

cex_p_lbl
: Scaling factor for text labels (probabilities). Default: cex_p_lbl = cex_lbl - .05.

col_pal
: Color palette. Default: col_pal = pal.

mar_notes
: Boolean option for showing margin notes. Default: mar_notes = FALSE.

...
: Other (graphical) parameters.

Details

plot_prism generalizes and replaces plot_fnet by removing the dependency on the R package diagram and providing many additional options.

Value

Nothing (NULL).

See Also

plot_fnet for older (obsolete) version; plot_area for plotting mosaic plot (scaling area dimensions); plot_bar for plotting frequencies as vertical bars; plot_tab for plotting table (without scaling area dimensions); pal contains current color settings; txt contains current text settings.

Other visualization functions: plot.riskyr(), plot_area(), plot_bar(), plot_crisk(), plot_curve(), plot_fnet(), plot_icons(), plot_mosaic(), plot_plane(), plot_tab(), plot_tree()

Examples

```
## Basics:
# (1) Using global prob and freq values:
plot_prism()  # default prism plot,
# same as:
# plot_prism(by = "cddc", area = "no", scale = "p",
#            f_lbl = "num", f_lwd = 0, cex_lbl = .90,
#            p_lbl = "mix", arr_c = -2, cex_p_lbl = NA)

# (2) Providing values:
plot_prism(N = 10, prev = 1/3, sens = 3/5, spec = 4/5, area = "hr")
plot_prism(N = 10, prev = 1/4, sens = 3/5, spec = 2/5, area = "sq", mar_notes = TRUE)

# (3) Rounding and sampling:
plot_prism(N = 100, prev = 1/3, sens = 2/3, spec = 6/7, area = "hr", round = FALSE)
plot_prism(N = 100, prev = 1/3, sens = 2/3, spec = 6/7, area = "hr", sample = TRUE, scale = "freq")

# (4) Custom colors and text:
plot_prism(col_pal = pal_bw, f_lwd = .5, p_lwd = .5, lty = 2, # custom fbox color, prob links,
           font = 3, cex_p_lbl = .75)                          # and text labels

my_txt <- init_txt(cond_lbl = "The Truth", cond_true_lbl = "so true", cond_false_lbl = "so false",
                   hi_lbl = "TP", mi_lbl = "FN", fa_lbl = "FP", cr_lbl = "TN")
my_col <- init_pal(N_col = rgb(0, 169, 224, max = 255),  # seeblau
             hi_col = "gold", mi_col = "firebrick1", fa_col = "firebrick2", cr_col = "orange")
plot_prism(f_lbl = "nam", lbl_txt = my_txt,
           col_pal = my_col, f_lwd = .5)

## Local values and custom color/txt settings:
plot_prism(N = 7, prev = 1/2, sens = 3/5, spec = 4/5, round = FALSE,
           by = "cdac", lbl_txt = txt_org, f_lbl = "namnum", f_lbl_sep = ":\n",
           f_lwd = 1, col_pal = pal_rgb)  # custom colors

plot_prism(N = 5, prev = 1/2, sens = .8, spec = .5, scale = "p",  # note scale!
           by = "cddc", area = "hr", col_pal = pal_bw, f_lwd = 1) # custom colors

plot_prism(N = 3, prev = .50, sens = .50, spec = .50, scale = "p",            # note scale!
         area = "sq", lbl_txt = txt_org, f_lbl = "namnum", f_lbl_sep = ":\n", # custom text
          col_pal = pal_kn, f_lwd = .5)                                      # custom colors

## Plot versions:
# (A) tree/single tree (nchar(by) == 2):
#     3 versions:
plot_prism(by = "cd", f_lbl = "def", col_pal = pal_mod) # by condition (freq boxes: hi mi fa cr)
plot_prism(by = "dc", f_lbl = "def", col_pal = pal_mod) # by decision  (freq boxes: hi fa mi cr)
plot_prism(by = "ac", f_lbl = "def", col_pal = pal_mod) # by accuracy  (freq boxes: hi cr mi fa)

# (B) prism/double tree (nchar(by) == 4):
#     6 (3 x 2) versions (+ 3 redundant ones):
plot_prism(by = "cddc")    # v01 (default)
plot_prism(by = "cdac")    # v02
# plot_prism(by = "cdcd")  # (+) Message
```

```
plot_prism(by = "dccd")    # v03
plot_prism(by = "dcac")    # v04
# plot_prism(by = "dcdc")  # (+) Message

plot_prism(by = "accd")    # v05
plot_prism(by = "acdc")    # v06
# plot_prism(by = "acac")  # (+) Message

## Other options:

# area:
# plot_prism(area = "no")  # rectangular boxes (default): (same if area = NA/NULL)
plot_prism(area = "hr")  # horizontal rectangles (widths on each level sum to N)
plot_prism(area = "sq")  # squares (areas on each level sum to N)

# scale (matters for scaled areas and small N):
plot_prism(N = 5, prev = .3, sens = .8, spec = .6,
           area = "hr", scale = "p")  # widths scaled by prob
plot_prism(N = 5, prev = .3, sens = .8, spec = .6,
           area = "hr", scale = "f")  # widths scaled by (rounded or non-rounded) freq
plot_prism(N = 4, prev = .2, sens = .7, spec = .8,
           area = "sq", scale = "p")  # areas scaled by prob
plot_prism(N = 4, prev = .2, sens = .7, spec = .8,
           area = "sq", scale = "f")  # areas scaled by (rounded or non-rounded) freq

## Frequency boxes:
# f_lbl:
plot_prism(f_lbl = "abb")      # abbreviated freq names (variable names)
plot_prism(f_lbl = "nam")      # only freq names
plot_prism(f_lbl = "num")      # only numeric freq values (default)
plot_prism(f_lbl = "namnum")   # names and numeric freq values
# plot_prism(f_lbl = "namnum", cex_lbl = .75)  # smaller freq labels
# plot_prism(f_lbl = NA)       # no freq labels
# plot_prism(f_lbl = "def")   # informative default: short name and numeric value (abb = num)

# f_lwd:
# plot_prism(f_lwd =  0)  # no lines (default), set to tiny_lwd = .001, lty = 0 (same if NA/NULL)
plot_prism(f_lwd =  1)  # basic lines
plot_prism(f_lwd =  3)  # thicker lines
# plot_prism(f_lwd = .5)  # thinner lines

## Probability links:
# Scale link widths (p_lwd & p_scale):
plot_prism(p_lwd = 6, p_scale = TRUE)
plot_prism(area = "sq", f_lbl = "num", p_lbl = NA, col_pal = pal_bw, p_lwd = 6, p_scale = TRUE)

# p_lbl:
plot_prism(p_lbl = "mix")     # abbreviated names with numeric values (abb = num)
plot_prism(p_lbl = "min")     # minimal names (of key probabilities)
# plot_prism(p_lbl = NA)      # no prob labels (NA/NULL/"none")
plot_prism(p_lbl = "nam")     # only prob names
plot_prism(p_lbl = "num")     # only numeric prob values
```

```
plot_prism(p_lbl = "namnum")  # names and numeric prob values
# plot_prism(p_lbl = "namnum", cex_p_lbl = .70)  # smaller prob labels
# plot_prism(by = "cddc", p_lbl = "min")  # minimal labels
# plot_prism(by = "cdac", p_lbl = "min")
# plot_prism(by = "cddc", p_lbl = "mix")  # mix abbreviated names and numeric values
# plot_prism(by = "cdac", p_lbl = "mix")
# plot_prism(by = "cddc", p_lbl = "abb")  # abbreviated names
# plot_prism(by = "cdac", p_lbl = "abb")
# plot_prism(p_lbl = "any") # short name and value (abb = num)

# arr_c:
plot_prism(arr_c =  0)  # acc_c = 0: no arrows
plot_prism(arr_c = -3)  # arr_c = -1 to -3: points at both ends
plot_prism(arr_c = -2)  # point at far end
plot_prism(arr_c = +2)  # crr_c = 1-3: V-shape arrows at far end
# plot_prism(arr_c = +3)  # V-shape arrows at both ends
# plot_prism(arr_c = +6)  # arr_c = 4-6: T-shape arrows

## Plain plot versions:
plot_prism(area = "no", f_lbl = "def", p_lbl = "num", col_pal = pal_mod, f_lwd = 1,
           main = NA, sub = NA, mar_notes = FALSE)  # remove titles and margin notes
plot_prism(area = "no", f_lbl = "nam", p_lbl = "min",
           main = NA, sub = "My subtitle", col_pal = pal_rgb)  # only subtitle
plot_prism(area = "no", f_lbl = "num", p_lbl = "num", col_pal = pal_kn) # default title & subtitle

plot_prism(area = "hr", f_lbl = "nam", f_lwd = .5, p_lwd = .5, col_pal = pal_bwp)
plot_prism(area = "hr", f_lbl = "nam", f_lwd = .5, p_lbl = "num", main = NA, sub = NA)

# plot_prism(area = "sq", f_lbl = "nam", p_lbl = NA, col_pal = pal_rgb)
plot_prism(area = "sq", f_lbl = "def", f_lbl_sep = ":\n", p_lbl = NA, f_lwd = 1, col_pal = pal_kn)

## Suggested combinations:
plot_prism(f_lbl = "nam", p_lbl = "mix", col_pal = pal_mod)  # basic plot
plot_prism(f_lbl = "namnum", p_lbl = "num", cex_lbl = .80, cex_p_lbl = .75)
# plot_prism(area = "no", f_lbl = "def", p_lbl = "abb",           # def/abb labels
#            f_lwd = .8, p_lwd = .8, lty = 3, col_pal = pal_bwp)  # black-&-white

plot_prism(area = "hr", f_lbl = "num", p_lbl = "mix", f_lwd = 1, cex_p_lbl = .75)
plot_prism(area = "hr", f_lbl = "nam", p_lbl = "num", p_lwd = 6, p_scale = TRUE)
plot_prism(area = "hr", f_lbl = "abb", p_lbl = "abb", f_lwd = 1, col_pal = pal_kn)

# plot_prism(area = "sq", f_lbl = "nam", p_lbl = "abb", lbl_txt = txt_TF)
plot_prism(area = "sq", f_lbl = "num", p_lbl = "num", f_lwd = 1, col_pal = pal_rgb)
plot_prism(area = "sq", f_lbl = "namnum", p_lbl = "mix", f_lwd = .5, col_pal = pal_kn)
```

---

plot_tab — *Plot a 2 x 2 contingency table of population frequencies.*

Description

plot_tab plots a 2 x 2 contingency table (aka. confusion table) of 4 classification cases (hi, mi, fa, cr) and corresponding row and column sums.

Usage

```
plot_tab(
  prev = num$prev,
  sens = num$sens,
  mirt = NA,
  spec = num$spec,
  fart = NA,
  N = num$N,
  by = "cddc",
  p_split = "v",
  area = "no",
  scale = "p",
  round = TRUE,
  sample = FALSE,
  f_lbl = "num",
  f_lbl_sep = NA,
  f_lbl_sum = f_lbl,
  f_lbl_hd = "nam",
  f_lwd = 0,
  gaps = c(NA, NA),
  brd_w = 0.1,
  p_lbl = NA,
  arr_c = -3,
  col_p = c(grey(0.15, 0.99), "yellow", "yellow"),
  brd_dis = 0.3,
  lbl_txt = txt,
  main = txt$scen_lbl,
  sub = "type",
  title_lbl = NULL,
  cex_lbl = 0.9,
  cex_p_lbl = NA,
  col_pal = pal,
  mar_notes = FALSE,
  ...
)
```

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. Note: `N` is not represented in the plot, but used for computing frequency information `freq` from current probabilities `prob`.

`by` A character code specifying 2 perspectives that split the population into subsets, with 6 options:

1. `"cddc"`: by condition (cd) and by decision (dc) (default);
2. `"cdac"`: by condition (cd) and by accuracy (ac);
3. `"dccd"`: by decision (dc) and by condition (cd);
4. `"dcac"`: by decision (dc) and by accuracy (ac);
5. `"accd"`: by accuracy (ac) and by condition (cd);
6. `"acdc"`: by accuracy (ac) and by decision (dc).

`p_split` Primary perspective for population split, with 2 options:

1. `"v"`: vertical (default);
2. `"h"`: horizontal.

Note: In contrast to `plot_area`, this setting only determines which 3 probability links are shown (e.g., when `p_link = "def"`).

`area` A character code specifying the shape of the main area, with 4 options:

1. `"sq"`: main area is scaled to square;
2. `"no"`: no scaling (rectangular area fills plot size; default).

`scale` Scale probabilities (but not table cell dimensions) either by exact probability or by (rounded or non-rounded) frequency, with 2 options:

1. `"p"`: scale main area dimensions by exact probability (default);
2. `"f"`: re-compute probabilities from (rounded or non-rounded) frequencies and scale main area dimensions by their frequency.

Note: `scale` setting matters for the display of probability values and for area plots with small population sizes `N` when `round = TRUE`.

`round` A Boolean option specifying whether computed frequencies are rounded to integers. Default: `round = TRUE`.

`sample` Boolean value that determines whether frequency values are sampled from `N`, given the probability values of `prev`, `sens`, and `spec`. Default: `sample = FALSE`.

`f_lbl` Type of label for showing frequency values in 4 main areas, with 6 options:

1. `"def"`: abbreviated names and frequency values (default);
2. `"abb"`: abbreviated frequency names only (as specified in code);

3. "nam": names only (as specified in lbl_txt = txt);
4. "num": numeric frequency values only;
5. "namnum": names (as specified in lbl_txt = txt) and numeric values;
6. "no": no frequency labels (same for f_lbl = NA or NULL).

f_lbl_sep
: Label separator for main frequencies (used for f_lbl = "def" OR "namnum"). Use f_lbl_sep = ":\n" to add a line break between name and numeric value. Default: f_lbl_sep = NA (set to " = " or ":\n" based on f_lbl).

f_lbl_sum
: Type of label for showing frequency values in summary cells, with same 6 options as f_lbl (above). Default: f_lbl_sum = "def": abbreviated names and numeric values.

f_lbl_hd
: Type of label for showing frequency values in header, with same 6 options as f_lbl (above). Default: f_lbl_hd = "nam": names only (as specified in lbl_txt = txt).

f_lwd
: Line width of areas. Default: f_lwd = 1.

gaps
: Size of gaps (as binary numeric vector) specifying the widths of vertical and horizontal gaps between 2 x 2 table and sums (in bottom row and right column). Default: gaps = c(.05, .06).

brd_w
: Border width for showing 2 perspective summaries on top and left borders of main area (as a proportion of area size) in a range 0 <= brd_w <= 1. Default: brd_w = .10.

p_lbl
: Type of label for showing 3 key probability links and values, with 7 options:

1. "def": show links and abbreviated names and probability values;
2. "abb": show links and abbreviated probability names;
3. "nam": show links and probability names (as specified in code);
4. "num": show links and numeric probability values;
5. "namnum": show links with names and numeric probability values;
6. "no": show links with no labels;
7. NA: no link (same for p_lbl = NULL, default).

arr_c
: Arrow code for symbols at ends of probability links (as a numeric value -3 <= arr_c <= +6), with the following options:

- -1 to -3: points at one/other/both end/s;
- 0: no symbols;
- +1 to +3: V-arrow at one/other/both end/s;
- +4 to +6: T-arrow at one/other/both end/s.

Default: arr_c = -3 (points at both ends).

col_p
: Colors of probability links (as vector of 3 colors). Default: col_p = c(grey(.15, .99), "yellow", "yellow").

brd_dis
: Distance of probability links from cell center (as a constant). Default: brd_dis = .30. Note: Adjust to avoid overlapping labels.

lbl_txt
: Default label set for text elements. Default: lbl_txt = txt.

main
: Text label for main plot title. Default: main = txt$scen_lbl.

sub	Text label for the subtitle of the plot (shown below the main title). Default: sub = "type" shows information on current plot type.
title_lbl	**Deprecated** text label for current plot title. Replaced by main.
cex_lbl	Scaling factor for text labels (frequencies and headers). Default: cex_lbl = .90.
cex_p_lbl	Scaling factor for text labels (probabilities). Default: cex_p_lbl = cex_lbl - .05.
col_pal	Color palette. Default: col_pal = pal.
mar_notes	Boolean option for showing margin notes. Default: mar_notes = FALSE.
...	Other (graphical) parameters.

Details

plot_tab computes its frequencies freq 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.

plot_tab is derived from plot_area, but does not scale the dimensions of table cells.

Value

Nothing (NULL).

See Also

plot_area for plotting mosaic plot (scaling area dimensions); pal contains current color settings; txt contains current text settings.

Other visualization functions: plot.riskyr(), plot_area(), plot_bar(), plot_crisk(), plot_curve(), plot_fnet(), plot_icons(), plot_mosaic(), plot_plane(), plot_prism(), plot_tree()

Examples

```
## Basics:
# (1) Plotting global freq and prob values:
plot_tab()
plot_tab(area = "sq", f_lwd = 3, col_pal = pal_rgb)
plot_tab(f_lbl = "namnum", f_lbl_sep = " = ", brd_w = .10, f_lwd = .5)

# (2) Computing local freq and prob values:
plot_tab(prev = .5, sens = 4/5, spec = 3/5, N = 10, f_lwd = 1)

# (3) Rounding and sampling:
plot_tab(N = 100, prev = 1/3, sens = 2/3, spec = 6/7, round = FALSE)
plot_tab(N = 100, prev = 1/3, sens = 2/3, spec = 6/7, sample = TRUE)

## Plot versions:
# by x p_split [yields (3 x 2) x 2] = 12 versions]:
plot_tab(by = "cddc", p_split = "v", p_lbl = "def")  # v01 (see v07)
plot_tab(by = "cdac", p_split = "v", p_lbl = "def")  # v02 (see v11)
```

```
plot_tab(by = "cddc", p_split = "h", p_lbl = "def")  # v03 (see v05)
plot_tab(by = "cdac", p_split = "h", p_lbl = "def")  # v04 (see v09)

# plot_tab(by = "dccd", p_split = "h", p_lbl = "def")  # v07 (v01 rotated)
# plot_tab(by = "dccd", p_split = "v", p_lbl = "def")  # v05 (v03 rotated)
plot_tab(by = "dcac", p_split = "v", p_lbl = "def")    # v06 (see v12)
plot_tab(by = "dcac", p_split = "h", p_lbl = "def")    # v08 (see v10)

# plot_tab(by = "accd", p_split = "v", p_lbl = "def")  # v09 (v04 rotated)
# plot_tab(by = "acdc", p_split = "v", p_lbl = "def")  # v10 (v08 rotated)
# plot_tab(by = "accd", p_split = "h", p_lbl = "def")  # v11 (v02 rotated)
# plot_tab(by = "acdc", p_split = "h", p_lbl = "def")  # v12 (v06 rotated)

## Explore labels and links:
# plot_tab(f_lbl = "abb", p_lbl = NA)  # abbr. labels, no probability links
# plot_tab(f_lbl = "num", f_lbl_sum = "abb", p_lbl = "num", f_lbl_hd = "abb")
plot_tab(f_lbl = "def", f_lbl_sum = "def", p_lbl = "def", f_lbl_hd = "nam")
plot_tab(f_lbl = "namnum", f_lbl_sep = " = ",
         f_lbl_sum = "namnum", f_lbl_hd = "num", p_lbl = "namnum")

## Misc. options:
plot_tab(area = "sq")        # area: square
# plot_tab(main = "")     # no titles
# plot_tab(mar_notes = TRUE)   # show margin notes
plot_tab(by = "cddc", gaps = c(.08, .00), area = "sq")      # gaps
# plot_tab(by = "cddc", gaps = c(.02, .08), p_split = "h")  # gaps

# Showing prob as lines:
plot_tab(prev = 1/4, sens = 6/7, spec = 3/5, N = 100,
         by = "cddc", p_split = "v", col_pal = pal_rgb,
         p_lbl = "def", brd_dis = .25, arr_c = +3, lwd = 2)

# Custom text labels and colors:
plot_tab(prev = .5, sens = 4/5, spec = 3/5, N = 10,
         by = "cddc", p_split = "v", area = "no",
         main = "Main title", sub = "The subtitle", lbl_txt = txt_TF,  # custom text
         f_lbl = "namnum", f_lbl_sep = ":\n", f_lbl_sum = "num", f_lbl_hd  = "nam",
         col_pal = pal_vir, f_lwd = 3)  # custom colors
plot_tab(prev = .5, sens = 3/5, spec = 4/5, N = 10,
         by = "cddc", p_split = "h", area = "sq",
         main = NA, sub = NA, lbl_txt = txt_org,  # custom text
         f_lbl = "namnum", f_lbl_sep = ":\n", f_lbl_sum = "num", f_lbl_hd  = "nam",
         col_pal = pal_kn, f_lwd = 1)  # custom colors

## Note some differences to plot_area (i.e., area/mosaic plot):
# In plot_tab:
# (1) p_split does not matter (except for selecting different prob links):
plot_tab(by = "cddc", p_split = "v")  # v01 (see v07)
plot_tab(by = "cddc", p_split = "h")  # v03 (see v05)

# (2) scale does not matter for dimensions (which are constant),
#     BUT matters for values shown in prob links and on margins:
plot_tab(N = 5, prev = .3, sens = .9, spec = .5,
```

```
        by = "cddc", scale = "p", p_lbl = "def", round = TRUE)  # (a) exact prob values
plot_tab(N = 5, prev = .3, sens = .9, spec = .5,
      by = "cddc", scale = "f", p_lbl = "def", round = TRUE)  # (b) prob from rounded freq!
plot_tab(N = 5, prev = .3, sens = .9, spec = .5,
        by = "cddc", scale = "f", p_lbl = "def", round = FALSE) # (c) same values as (a)
```

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

Description

`plot_tree` drew a tree diagram of frequencies (as nodes) and probabilities (as edges).

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 = "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),
  cex_lbl = 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.
sens	The decision's sensitivity sens.
mirt	The decision's miss rate mirt.
spec	The decision's specificity value spec.
fart	The decision's false alarm rate fart.
N	The number of individuals in the population.
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 3 options: 1. "cd" ... by condition; 2. "dc" ... by decision; 3. "ac" ... by accuracy.
area	A character code specifying the area of the boxes (or their relative sizes) with 3 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.
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 on the margin of the plot.
w_acc	Weighting parameter w used to compute weighted accuracy w_acc in comp_accu_freq.
	Various other options allow the customization of text labels and colors:
title_lbl	Text label for current plot title.
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).
cex_lbl	Scaling factor for text labels (in boxes and on arrows).
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_tree is deprecated – please use plot_prism instead.

Value

Nothing (NULL).

See Also

plot_prism is the new version of this function.

Other visualization functions: plot.riskyr(), plot_area(), plot_bar(), plot_crisk(), plot_curve(), plot_fnet(), plot_icons(), plot_mosaic(), plot_plane(), plot_prism(), plot_tab()

Examples

```
plot_tree()  # frequency tree with current default options (by = "cd")

# alternative perspectives:
plot_tree(by = "dc")  # tree by decision
plot_tree(by = "ac")  # tree by accuracy

# See plot_prism for details and additional options.
```

---

popu — *A table of cases (in the current population).*

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 (i.e., its case or cell combination, in SDT terms), as true positive (hit hi), false negative (miss mi), false positive (false alarm fa), or true negative (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.

By default, `comp_popu` uses the current information contained in `txt` to define text labels.

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`; `read_popu` interprets a data frame as a riskyr scenario; `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 (aka. bias).*

Description

ppod defines the proportion (baseline probability or rate) of a decision being `positive` (but not necessarily accurate/correct).

Usage

```
ppod
```

Format

An object of class `numeric` of length 1.

Details

ppod is also known as bias, though the latter term is also used to describe a systematic tendency to deviate in any — rather than just positive — direction.

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 accurate/correct).

- 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 probabilities.

Other probabilities: FDR, FOR, NPV, PPV, acc, err, 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
```

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 probabilities.

Other probabilities: FDR, FOR, NPV, acc, err, 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
```

---

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 or truth.
- 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 (when adopting a "by decision" perspective) and to acc (when adopting a "by accuracy" 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 of the condition, but independent of the decision process or diagnostic procedure.

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

References

Consult Wikipedia for additional information.

See Also

prob contains current probability information; num contains basic numeric variables; init_num initializes basic numeric variables; comp_prob computes derived probabilities; comp_freq computes natural frequencies from probabilities; is_prob verifies probabilities.

Other probabilities: FDR, FOR, NPV, PPV, acc, err, 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
```

---

print.summary.riskyr *Print summary information of a riskyr scenario.*

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 generic print function).

Format

Printed output of a "summary.riskyr" object.

See Also

riskyr initializes a riskyr scenario.

Examples

```
summary(scenarios$n4)
```

---

prob   *List current probability information.*

Description

prob is a list of named numeric variables containing 3 essential (1 non-conditional prev and 2 conditional sens and spec) probabilities and 8 derived (ppod and acc, as well as 6 conditional) probabilities:

Usage

```
prob
```

Format

An object of class list of length 13.

Details

prob currently contains the following probabilities:

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).
11. the accuracy acc (i.e., probability of correct decisions dec_cor or correspondence of decisions to conditions).
12. the conditional probability p_acc_hi (i.e., the probability of hi given that the decision is correct dec_cor).
13. the conditional probability p_err_fa (i.e., the probability of fa given that the decision is erroneous dec_err).

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).

Key relationships between frequencies and probabilities (see documentation of comp_freq or comp_prob for details):

- Three perspectives on a population:
  by condition / by decision / by accuracy.
- Defining probabilities in terms of frequencies:
  Probabilities can be computed as ratios between frequencies, but beware of rounding issues.

Functions translating between representational formats: comp_prob_prob, comp_prob_freq, comp_freq_prob, comp_freq_freq (see documentation of comp_prob_prob for details).

Visualizations of current probability information are provided by plot_area, plot_prism, and plot_curve.

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; accu contains current accuracy information.

Other lists containing current scenario information: accu, freq, num, pal_bwp, pal_bw, pal_kn, pal_mbw, pal_mod, pal_org, pal_rgb, pal_unikn, pal_vir, pal, txt_TF, txt_org, txt

Examples

```
prob <- comp_prob()  # initialize prob to default parameters
prob                 # show current values
length(prob)         # 13 key probabilities (and their values)
```

read_popu — *Read population data (from df) into a riskyr scenario (description).*

Description

read_popu reads a data frame df (containing observations of some population that are cross-classified on two binary variables) and returns a scenario of class "riskyr" (i.e., a description of the data).

Usage

```
read_popu(
  df = popu,
  ix_by_top = 1,
  ix_by_bot = 2,
  ix_sdt = 3,
  hi_lbl = txt$hi_lbl,
  mi_lbl = txt$mi_lbl,
  fa_lbl = txt$fa_lbl,
  cr_lbl = txt$cr_lbl,
  ...
)
```

Arguments

df	A data frame providing a population popu of individuals, which are identified on at least 2 binary variables and cross-classified into 4 cases in a 3rd variable. Default: df = popu (as data frame).
ix_by_top	Index of variable (column) providing the 1st (X/top) perspective (in df). Default: ix_by_top = 1 (1st column).
ix_by_bot	Index of variable (column) providing the 2nd (Y/bot) perspective (in df). Default: ix_by_bot = 2 (2nd column).
ix_sdt	Index of variable (column) providing a cross-classification into 4 cases (in df). Default: ix_by_bot = 3 (3rd column).
hi_lbl	Label of cases classified as hi (TP).
mi_lbl	Label of cases classified as mi (FN).
fa_lbl	Label of cases classified as fa (FP).
cr_lbl	Label of cases classified as cr (TN).
...	Additional parameters (passed to riskyr).

Details

Note that df needs to be structured (cross-classified) according to the data frame popu, created by comp_popu.

Value

An object of class "riskyr" describing a risk-related scenario.

See Also

`comp_popu` creates data (as df) from description (frequencies); `write_popu` creates data (as df) from a riskyr scenario (description); `popu` for data format; `riskyr` initializes a `riskyr` scenario.

Other functions converting data/descriptions: `comp_popu()`, `write_popu()`

Examples

```
# Generating and interpreting different scenario types:

# (A) Diagnostic/screening scenario (using default labels): ------
popu_diag <- comp_popu(hi = 4, mi = 1, fa = 2, cr = 3)
# popu_diag
scen_diag <- read_popu(popu_diag, scen_lbl = "Diagnostics", popu_lbl = "Population tested")
plot(scen_diag, type = "prism", area = "no", f_lbl = "namnum")

# (B) Intervention/treatment scenario: ------
popu_treat <- comp_popu(hi = 80, mi = 20, fa = 45, cr = 55,
                cond_lbl = "Treatment", cond_true_lbl = "pill", cond_false_lbl = "placebo",
                 dec_lbl = "Health status", dec_pos_lbl = "healthy", dec_neg_lbl = "sick")
# popu_treat
s_treat <- read_popu(popu_treat, scen_lbl = "Treatment", popu_lbl = "Population treated")
plot(s_treat, type = "prism", area = "sq", f_lbl = "namnum", p_lbl = "num")
plot(s_treat, type = "icon", lbl_txt = txt_org, col_pal = pal_org)

# (C) Prevention scenario (e.g., vaccination): ------
popu_vacc <- comp_popu(hi = 960, mi = 40, fa = 880, cr = 120,
                   cond_lbl = "Vaccination", cond_true_lbl = "yes", cond_false_lbl = "no",
                       dec_lbl = "Disease", dec_pos_lbl = "no flu", dec_neg_lbl = "flu")
# popu_vacc
s_vacc <- read_popu(popu_vacc, scen_lbl = "Vaccination effects", popu_lbl = "RCT population")
plot(s_vacc, type = "prism", area = "sq", f_lbl = "namnum", col_pal = pal_rgb, p_lbl = "num")
```

`riskyr` *Create a riskyr scenario.*

Description

`riskyr` creates a scenario of class "riskyr", which can be visualized by the `plot` method `plot.riskyr` and summarized by the `summary` method `summary.riskyr`.

Usage

```
riskyr(
  scen_lbl = txt$scen_lbl,
  popu_lbl = txt$popu_lbl,
  N_lbl = txt$N_lbl,
  cond_lbl = txt$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$dec_neg_lbl,
  acc_lbl = txt$acc_lbl,
  dec_cor_lbl = txt$dec_cor_lbl,
  dec_err_lbl = txt$dec_err_lbl,
  sdt_lbl = txt$sdt_lbl,
  hi_lbl = txt$hi_lbl,
  mi_lbl = txt$mi_lbl,
  fa_lbl = txt$fa_lbl,
  cr_lbl = txt$cr_lbl,
  prev = NA,
  sens = NA,
  spec = NA,
  fart = NA,
  N = NA,
  hi = NA,
  mi = NA,
  fa = NA,
  cr = NA,
  scen_lng = txt$scen_lng,
  scen_txt = txt$scen_txt,
  scen_src = txt$scen_src,
  scen_apa = txt$scen_apa,
  round = TRUE,
  sample = FALSE
)
```

Arguments

`scen_lbl`	The current scenario title (sometimes in Title Caps).
`popu_lbl`	A brief description of the current population or sample.
`N_lbl`	A label for the current population `popu` or sample.
`cond_lbl`	A label 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 label 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).

acc_lbl — A label for *accuracy* (i.e., correspondence between condition and decision or judgment).

dec_cor_lbl — A label for *correct* (or accurate) decisions or judgments.

dec_err_lbl — A label for *incorrect* (or erroneous) decisions or judgments.

sdt_lbl — A label for the combination of *condition* and *decision* currently made.

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).

Essential probabilities:

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.

Essential frequencies:

N — The number of individuals in the scenario's population. A suitable value of N is computed, if not provided.

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 and source information:

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).

scen_src — Source information for the current scenario.

`scen_apa`	Source information for the current scenario according to the American Psychological Association (APA style).
`round`	Boolean value that determines whether frequency values are rounded to the nearest integer. Default: `round = TRUE`. Note: Only rounding when using `comp_freq_prob` (i.e., computing `freq` from `prob` description).
`sample`	Boolean value that determines whether frequency values are sampled from `N`, given the probability values of `prev`, `sens`, and `spec`. Default: `sample = FALSE`. Note: Only sampling when using `comp_freq_prob` (i.e., computing `freq` from `prob` description).

Format

An object of class "riskyr" with textual and numeric information describing a risk-related scenario.

Details

Beyond basic scenario information (i.e., text elements describing a scenario) only the population size `N` and the essential probabilities `prev`, `sens`, `spec`, and `fart` are used and returned.

Note:

- Basic text information and some numeric parameters (see `num` and `init_num`) are integral parts of a `riskyr` scenario.
- By contrast, basic *color* information (see `pal` and `init_pal`) is not an integral part, but independently defined.
- The names of *probabilities* (see `prob`) are currently not an integral part of `txt` and `riskyr` scenarios (but defined in `prob_lbl_def` and `label_prob`).

Value

An object of class "riskyr" describing a risk-related scenario.

Scenario-specific titles and text labels (see `txt`).

See Also

`init_num` and `num` for basic numeric parameters; `init_txt` and `txt` for current text settings; `init_pal` and `pal` for current color settings.

Other riskyr scenario functions: `plot.riskyr()`, `summary.riskyr()`

Other functions initializing scenario information: `init_num()`, `init_pal()`, `init_txt()`

Examples

```
# Defining scenarios: -----
# (a) minimal information:
hustosis <- riskyr(scen_lbl = "Screening for hustosis",
                   N = 1000, prev = .04, sens = .80, spec = .95)
```

```
# (2) detailed information:
scen_reoffend <- 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",
                        sdt_lbl = "combination",
                        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,  # (provide 1 of 2)
                        N = 753,
                        scen_src = "Example scenario")

# Using scenarios: -----
summary(hustosis)
plot(hustosis)

summary(scen_reoffend)
plot(scen_reoffend)

# 2 ways of defining the same scenario:
s1 <- riskyr(prev = .5, sens = .5, spec = .5, N = 100)  # s1: define by 3 prob & N
s2 <- riskyr(hi = 25, mi = 25, fa = 25, cr = 25)        # s2: same scenario by 4 freq
all.equal(s1, s2)  # should be TRUE

# Rounding and sampling:
s3 <- riskyr(prev = 1/3, sens = 2/3, spec = 6/7, N = 100, round = FALSE)  # s3: w/o rounding
s4 <- riskyr(prev = 1/3, sens = 2/3, spec = 6/7, N = 100, sample = TRUE)  # s4: with sampling

# Note:
riskyr(prev = .5, sens = .5, spec = .5, hi = 25, mi = 25, fa = 25, cr = 25)  # works (consistent)
riskyr(prev = .5, sens = .5, spec = .5, hi = 25, mi = 25, fa = 25)        # works (ignores freq)

## Watch out for:
# riskyr(hi = 25, mi = 25, fa = 25, cr = 25, N = 101)  # warns, uses actual sum of freq
# riskyr(prev = .4, sens = .5, spec = .5, hi = 25, mi = 25, fa = 25, cr = 25)  # warns, uses freq
```

`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 (as list).*

Description

`scenarios` is a list of scenarios of class `riskyr` collected from the scientific literature and other sources and to be used by visualization and summary functions.

Usage

```
scenarios
```

Format

A list with currently 25 scenarios of class `riskyr` which are each described by 21 variables.

Details

`scenarios` currently contains the following scenarios (n1 to n12 in English language, n13 to n25 in German language):

1. Bowel cancer screening
2. Cab problem
3. Hemoccult test
4. Mammography screening
5. Mammography (freq)
6. Mammography (prob)
7. Mushrooms
8. Musical town
9. PSA test (baseline)
10. PSA test (patients)
11. Psylicraptis screening
12. Sepsis
13. Amniozentese (in German language)
14. HIV-Test 1
15. HIV-Test 2
16. HIV-Test 3
17. HIV-Test 4
18. Mammografie 1

19. Mammografie 2
20. Mammografie 3
21. Mammografie 4
22. Nackenfaltentest (NFT) 1
23. Nackenfaltentest (NFT) 2
24. Sigmoidoskopie 1
25. Sigmoidoskopie 2

Variables describing a scenario:

1. `scen_lbl`: Text label for current scenario.
2. `scen_lng`: Language of current scenario (en/de).
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 a subset of `init_txt` (to initialize `txt`) and `init_num` (to initialize `num`).

The variables `scen_src` and `scen_apa` provide 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/`).

See Also

`riskyr` initializes a `riskyr` scenario.

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 probabilities.

Other probabilities: FDR, FOR, NPV, PPV, acc, err, 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
```

---

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 probabilities.

Other probabilities: FDR, FOR, NPV, PPV, acc, err, 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
```

---

summary.riskyr *Summarize 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 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.

See Also

`riskyr` initializes a `riskyr` scenario.

Other riskyr scenario functions: `plot.riskyr()`, `riskyr()`

Examples

```
summary(scenarios$n4)
```

`txt` *Basic text elements.*

Description

`txt` is initialized to a list of named elements to define basic scenario titles and labels.

Usage

```
txt
```

Format

An object of class `list` of length 21.

Details

All textual elements that specify generic labels and titles of `riskyr` scenarios 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.

The list `txt` is used throughout the `riskyr` package unless a scenario defines scenario-specific text labels (when using the `riskyr` function).

Note:

- Basic text information and some numeric parameters (see `num` and `init_num`) are integral parts of a `riskyr` scenario.
- By contrast, basic *color* information (see `pal` and `init_pal`) is not an integral part, but independently defined.

- The names of *probabilities* (see prob) are currently not an integral part of txt and riskyr scenarios (but defined in prob_lbl_def and label_prob).

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 general name describing the current *population*.
7. N_lbl A short label for the current population popu or sample.
8. cond_lbl A general name for the *condition* dimension, or the feature (e.g., some disease) currently considered.
9. cond_true_lbl A short label for the *presence* of the current condition or cond_true cases (the condition's true state of being TRUE).
10. cond_false_lbl A short label for the *absence* of the current condition or cond_false cases (the condition's true state of being FALSE).
11. dec_lbl A general name for the *decision* dimension, or the judgment (e.g., some diagnostic test) currently made.
12. dec_pos_lbl A short label for *positive* decisions or dec_pos cases (e.g., predicting the presence of the condition).
13. dec_neg_lbl A short label for *negative* decisions or dec_neg cases (e.g., predicting the absence of the condition).
14. acc_lbl A general name for the *accuracy* dimension, or the correspondence between the condition currently considered and the decision judgment currently made.
15. dec_cor_lbl A short label for *correct* and *accurate* decisions or dec_cor cases (accurate predictions).
16. dec_err_lbl A short label for *incorrect* decisions or dec_err cases (erroneous predictions).
17. sdt_lbl A general name for all 4 *cases/categories/cells* of the 2x2 contingency table (e.g., condition x decision, using SDT).
18. hi_lbl A short label for *hits* or *true positives* hi/TP cases (i.e., correct decisions of the presence of the condition, when the condition is actually present).
19. mi_lbl A short label for *misses* or *false negatives* mi/FN cases (i.e., incorrect decisions of the absence of the condition when the condition is actually present).
20. fa_lbl A short label for *false alarms* or *false positives* fa/FP cases (i.e., incorrect decisions of the presence of the condition when the condition is actually absent).
21. cr_lbl A short label for *correct rejections* or *true negatives* cr/TN cases (i.e., a correct decision of the absence of the condition, when the condition is actually absent).

See Also

init_txt initializes text information; riskyr initializes a riskyr scenario; 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_bwp, pal_bw, pal_kn, pal_mbw, pal_mod, pal_org, pal_rgb, pal_unikn, pal_vir, pal, prob, txt_TF, txt_org

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 example"  # Set a new scenario title
```

---

txt_org *List of original values of text elements.*

Description

txt_org is a copy of the initial list of text elements to define all scenario titles and labels.

Usage

```
txt_org
```

Format

An object of class list of length 21.

Details

See txt for details and default text information.

Assign txt <- txt_org to re-set default text labels.

See Also

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, freq, num, pal_bwp, pal_bw, pal_kn, pal_mbw, pal_mod, pal_org, pal_rgb, pal_unikn, pal_vir, pal, prob, txt_TF, txt

Examples

```
txt_org        # shows original text labels
txt_org["hi"]  # shows the original label for hits ("hi")
txt_org["hi"] <- "TP" # defines a new label for hits (true positives, TP)
```

txt_TF — *Alternative text labels (TP, FN, FP, TN).*

Description

`txt_TF` is initialized to alternative text labels to define a frequency naming scheme in which (hi, mi, fa, cr) are called (TP, FN, FP, TN).

Usage

```
txt_TF
```

Format

An object of class `list` of length 21.

Details

See `txt` for details and default text information.

Assign `txt <- txt_TF` to use as default text labels.

See Also

`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`, `freq`, `num`, `pal_bwp`, `pal_bw`, `pal_kn`, `pal_mbw`, `pal_mod`, `pal_org`, `pal_rgb`, `pal_unikn`, `pal_vir`, `pal`, `prob`, `txt_org`, `txt`

Examples

```
txt_TF        # shows text labels of txt_TF
txt_TF["hi"]  # shows the current label for hits ("TP")
txt_TF["hi"] <- "hit" # defines a new label for hits (true positives, TP)
```

t_A — *Cumulative risk curve A (main/transfer task A).*

Description

`t_A` provides the cumulative risk of some genetic risk factor for developing disease A in some target population as a function of age.

Usage

```
t_A
```

Format

A data frame (17 x 2).

age: age (in years).

crisk_A: cumulative risk of developing some disease A in the target population.

See Also

plot_crisk plots cumulative risk curves.

Other datasets: BRCA1_mam, BRCA1_ova, BRCA1, BRCA2_mam, BRCA2_ova, BRCA2, df_scenarios, t_B, t_I

---

t_B *Cumulative risk curve B (main/transfer task B).*

Description

t_B provides the cumulative risk of some genetic risk factor for developing disease B in some target population as a function of age.

Usage

```
t_B
```

Format

A data frame (17 x 2).

age: age (in years).

crisk_B: cumulative risk of developing some disease B in the target population.

See Also

plot_crisk plots cumulative risk curves.

Other datasets: BRCA1_mam, BRCA1_ova, BRCA1, BRCA2_mam, BRCA2_ova, BRCA2, df_scenarios, t_A, t_I t_I — *Cumulative risk curve I (introductory task).*

Description

t_I provides the cumulative risk of some genetic risk factor for developing a disease in some target population as a function of age.

Usage

```
t_I
```

Format

A data frame (17 x 2).

age: age (in years).

crisk_I: cumulative risk of developing some disease in the target population.

See Also

plot_crisk plots cumulative risk curves.

Other datasets: BRCA1_mam, BRCA1_ova, BRCA1, BRCA2_mam, BRCA2_ova, BRCA2, df_scenarios, t_A, t_B

---

write_popu — *Write a population table (data) from a riskyr scenario (description).*

Description

write_popu computes (or expands) a table popu (as an R data frame) from a riskyr scenario (description), using its 4 essential frequencies.

Usage

```
write_popu(x = NULL, ...)
```

Arguments

x	A riskyr scenario (description).
...	Additional parameters (text labels, passed to comp_popu).

Format

An object of class data.frame with N rows and 3 columns (e.g., "X/truth/cd", "Y/test/dc", "SDT/cell/class").

Details

write_popu expects a riskyr scenario as input and passes its 4 essential frequencies (rounded to integers) to comp_popu.

By default, write_popu uses the text settings contained in txt, but labels can be changed by passing arguments to comp_popu (via ...).

Value

A data frame popu containing N rows (individual cases) and 3 columns (e.g., "X/truth/cd", "Y/test/dc", "SDT/cell/class"). encoded as ordered factors (with 2, 2, and 4 levels, respectively).

See Also

comp_popu creates data (as df) from description (frequencies); read_popu creates a scenario (description) from data (as df); popu for data format; txt for current text settings; riskyr initializes a riskyr scenario.

Other functions converting data/descriptions: comp_popu(), read_popu()

Examples

```
# Define scenarios (by description):
s1 <- riskyr(prev = .5, sens = .5, spec = .5, N = 10)  # s1: define by 3 prob & N
s2 <- riskyr(hi = 2, mi = 3, fa = 2, cr = 3)           # s2: same scenario by 4 freq

# Create data (from descriptions):
write_popu(s1)  # data from (prob) description
write_popu(s2,  # data from (freq) description & change labels:
           cond_lbl = "Disease (X)",
           cond_true_lbl = "sick", cond_false_lbl = "healthy",
           dec_lbl = "Test (Y)")

# Rounding:
s3 <- riskyr(prev = 1/3, sens = 2/3, spec = 6/7, N = 10, round = FALSE)  # s3: w/o rounding
write_popu(s3, cond_lbl = "X", dec_lbl = "Y", sdt_lbl = "class")  # rounded to nearest integers

# Sampling:
s4 <- riskyr(prev = 1/3, sens = 2/3, spec = 6/7, N = 10, sample = TRUE)  # s4: with sampling
write_popu(s4, cond_lbl = "X", dec_lbl = "Y", sdt_lbl = "class")  # data from sampling
```

Index