Package: r4lineups (via r-universe)

November 2, 2026

Title Statistical Inference on Lineup Fairness

Version 2.3.0

URL https://github.com/CGTZA2/r4lineups, https://cgtza2.github.io/r4lineups/

BugReports https://github.com/CGTZA2/r4lineups/issues

Description Computes statistical measures for laboratory and police lineups in forensic psychology, including lineup bias, functional and effective size, diagnosticity, confidence-accuracy curves, signal detection models, and simulation tools. Optional face-similarity functions use numeric face representations computed by Python's 'deepface' library; optional lineup-memory model fitting uses a separately installed, pinned 'pyWitness' engine.

Depends R (>= 4.1.0)

Imports boot, dplyr, ggplot2, ggrepel, gridExtra, magick, magrittr, purrr, reticulate (>= 1.26), stats, tibble, utils

Suggests bslib, chromote, here, knitr, lme4, pander, patchwork, DT, rmarkdown, Rtsne, shiny, shinytest2, splines, testthat (>= 3.0.0), umap

License MIT + file LICENSE

Encoding UTF-8

Language en-US

LazyData true

VignetteBuilder knitr

Maintainer Colin Tredoux <colin.tredoux@uct.ac.za>

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libmagick++-dev gsfonts libpng-dev libssl-dev python3

Repository https://cgtza2.r-universe.dev

Date/Publication 2026-11-02 11:22:56 UTC

RemoteUrl https://github.com/CGTZA2/r4lineups

RemoteRef main

RemoteSha 18f4853c99a3becb98ec8993f7158e6127d3d193

Contents

allfoil_cihigh . . . 6
allfoilbias . . . 7
allprop . . . 7
available_detectors . . . 8
available_models . . . 9
batch_embeddings . . . 10
boot_winter_2ht . . . 11
bootstrap_anri . . . 13
calibration_bayes . . . 14
check_python_deps . . . 17
check_pywitness_deps . . . 18
chi_diag . . . 18
classify_lineup_choices . . . 20
compare_anri . . . 22
compare_dpp . . . 24
compare_eff_sizes.boot . . . 25
compare_max_sdt . . . 27
compare_models . . . 28
compare_multid . . . 30
compare_pauc . . . 31
compare_sdt_summary . . . 33
compare_utility . . . 35
compute_anri . . . 36
compute_dpp . . . 38
compute_eig . . . 39
cosine_to_similarity . . . 41
create_example_lineup_data . . . 42
d_bar . . . 43
d_weights . . . 44
datacheck1 . . . 46
datacheck2 . . . 46
datacheck3 . . . 47
detect_faces . . . 48
diag_param . . . 49
diag_ratio_T . . . 51
diag_ratio_T_bayes . . . 52
diag_ratio_T_bayes_compare . . . 54
diag_ratio_W . . . 55
display_lineup . . . 56
dmax_filler . . . 57
eff_size_per_foils . . . 58
effsize_compare . . . 59

embedding_distance . . . 60
esize_boot_dist . . . 61
esize_m . . . 63
esize_m_boot . . . 64
esize_T . . . 65
esize_T_bayes . . . 67
esize_T_bayes_compare . . . 69
esize_T_boot . . . 71
esize_T_ci_n . . . 72
estimate_msdt_params . . . 73
extract_sdt_metrics . . . 74
face_similarity . . . 75
fit_lineup_models . . . 76
fit_max_sdt . . . 78
fit_sdt_glm . . . 82
fit_sdt_glmm . . . 83
fit_sdt_roc . . . 84
fit_winter_2ht . . . 87
format_comparison_table . . . 89
func_size . . . 90
func_size.boot . . . 91
func_size_bayes . . . 92
func_size_report . . . 94
gen_boot_propci . . . 95
gen_boot_propmean_se . . . 96
gen_boot_samples . . . 97
gen_boot_samples_list . . . 98
gen_esize_m . . . 98
gen_esize_m_ci . . . 99
gen_lineup_prop . . . 100
gen_linevec . . . 101
get_embedding . . . 102
get_face_landmarks . . . 103
get_key_landmarks . . . 105
has_valid_face . . . 106
homog_diag . . . 106
homog_diag_boot . . . 108
i_esize_T . . . 110
innocent_id_rate_effective . . . 111
innocent_id_rate_nominal . . . 112
innocent_id_rate_uncorrected . . . 113
install_pywitness . . . 114
install_r4lineups_python . . . 115
line73 . . . 116
lineup_bias_boot_dist . . . 117
lineup_boot_allprop . . . 118
lineup_example . . . 119
lineup_pairwise_matrix . . . 120

lineup_prop_boot . . . 121
lineup_prop_tab . . . 122
lineup_prop_vec . . . 123
lineup_similarity . . . 123
lineup_table . . . 125
ln_diag_ratio . . . 126
make_bayes_curves . . . 128
make_bree_curve . . . 130
make_cac . . . 132
make_cac_gg . . . 133
make_cacdata . . . 134
make_calibration . . . 136
make_calibration_by_condition . . . 137
make_calibration_by_condition_gg . . . 139
make_calibration_data . . . 140
make_calibration_gg . . . 142
make_dpp . . . 143
make_eig . . . 145
make_eig_data . . . 147
make_fullroc . . . 148
make_fullroc_data . . . 150
make_multid . . . 152
make_ppv_range . . . 155
make_rac . . . 156
make_rac_gg . . . 157
make_racdata . . . 158
make_roc . . . 160
make_roc_gg . . . 161
make_rocdata . . . 162
make_utility_curves . . . 164
make_utility_difference . . . 166
makevec_prop . . . 168
max_filler_moments . . . 169
mickwick . . . 170
mockdata . . . 170
nortje2012 . . . 171
plot.esize_T_bayes . . . 171
plot.esize_T_bayes_compare . . . 172
plot.lineup_model_comparison . . . 172
plot.model_comparison . . . 173
plot.multid . . . 173
plot.pauc_comparison . . . 174
plot.power_analysis . . . 175
plot.sdt_roc_fit . . . 175
plot.winter_2ht . . . 176
plot.winter_2ht_boot . . . 176
plot_2ht_fit . . . 177
plot_2ht_parameters . . . 177

plot_anri_comparison . . . 178
plot_anri_difference_distribution . . . 179
plot_bayes_information_gain . . . 180
plot_bayes_prior_posterior . . . 181
plot_bree . . . 182
plot_dpp . . . 183
plot_dpp_comparison . . . 184
plot_effective_size_conf . . . 184
plot_eig . . . 185
plot_eig_posteriors . . . 186
plot_embedding_space . . . 187
plot_error_rate_conf . . . 188
plot_esize_distribution . . . 189
plot_fullroc . . . 190
plot_lineup_bias_distribution . . . 191
plot_lineup_similarity . . . 192
plot_msdt_joint . . . 193
plot_ppv_range . . . 194
plot_roc_cac . . . 195
plot_similarity_distribution . . . 196
plot_utility_curves . . . 197
plot_utility_difference . . . 197
ppv_by_confidence . . . 198
ppv_range_by_confidence . . . 201
print.esize_T_bayes . . . 202
print.esize_T_bayes_compare . . . 203
print.lineup_anri . . . 203
print.lineup_anri_comparison . . . 204
print.lineup_bayes_curves . . . 204
print.lineup_cac . . . 205
print.lineup_calibration . . . 205
print.lineup_data . . . 206
print.lineup_dpp . . . 206
print.lineup_dpp_comparison . . . 207
print.lineup_eig . . . 207
print.lineup_fullroc . . . 208
print.lineup_ppv . . . 208
print.lineup_ppv_range . . . 209
print.lineup_rac . . . 209
print.lineup_roc . . . 210
print.lineup_similarity . . . 210
print.lineup_table . . . 211
print.lineup_utility_comparison . . . 211
print.model_comparison . . . 212
print.pauc_comparison . . . 212
print.sdt_roc_fit . . . 213
print.simulated_lineup_data . . . 213
print.winter_2ht . . . 214

print.winter_2ht_boot . . . 214
rep_index . . . 215
rot_vector . . . 215
run_r4lineups_app . . . 216
sdt_compare . . . 216
sdt_summary_from_counts . . . 218
sdt_summary_variance . . . 219
simulate_lineup_data . . . 220
simulate_power_analysis . . . 223
standardize_lineup_data . . . 226
summary.lineup_eig . . . 228
summary.lineup_similarity . . . 228
summary.model_comparison . . . 229
summary.pauc_comparison . . . 229
summary.sdt_roc_fit . . . 230
summary.winter_2ht . . . 230
summary.winter_2ht_boot . . . 231
validate_lineup_data . . . 231
var_diag_ratio . . . 233
var_lnd . . . 234

Index **236**

`allfoil_cihigh` *Confidence Intervals for Proportion*

Description

Function to compute ci high for each foil in a lineup

Usage

```
allfoil_cihigh(linetabprops, sumlineup)
```

Arguments

`linetabprops`	A dataframe of bootstrapped lineup proportions
`sumlineup`	Number of mock-witness observations used to estimate each proportion.

References

Malpass, R. S. (1981). Effective size and defendant bias in eyewitness identification lineups. *Law and Human Behavior, 5*(4), 299-309.

Examples

```
# Upper CI bounds for the choice proportions of three lineup members,
# based on 20 mock witnesses
allfoil_cihigh(c(0.5, 0.3, 0.2), 20)
```

allfoilbias *Bias for each lineup member*

Description

Function to compute bias for each lineup member (assuming foil is suspect, from Malpass, 1981)

Usage

```
allfoilbias(lineup_tab, target_pos, k)
```

Arguments

lineup_tab	A table of lineup choices
target_pos	A scalar, representing target position in lineup. Must be declared by user
k	Nominal size (i.e., total number of members in lineup). Must be specified by user (scalar).

References

Malpass, R. S. (1981). Effective size and defendant bias in eyewitness identification lineups. *Law and Human Behavior, 5*(4), 299-309.

Examples

```
lineup_vec <- round(runif(100, 1, 6))
lineup_tab <- table(lineup_vec)
x <- allfoilbias(lineup_tab, 5, 6)
```

allprop *Lineup proportion for all lineup members*

Description

Computes lineup proportion for each member in a lineup

Usage

```
allprop(lineup_vec, k)
```

Arguments

lineup_vec	A numeric vector of lineup choices
k	Number of members in lineup. Must be specified by user (scalar).

Value

Returns a vector containing lineup proportion for each lineup member

References

Wells, G. L., Leippe, M. R., & Ostrom, T. M. (1979). Guidelines for assessing the fairness of a lineup. *Law and Human Behavior, 3*(4), 285-293.

Examples

```
#Data:
lineup_vec <- round(runif(100, 1, 6))

#Call:
x <- allprop(lineup_vec, k = 6)
```

---

available_detectors *List Available Face Detectors*

Description

Returns a character vector of available face detection backends.

Usage

```
available_detectors()
```

Details

Available detectors (in rough order of accuracy/speed tradeoff):

- retinaface - Most accurate, recommended for research
- mtcnn - Good accuracy, moderate speed
- mediapipe - Fast, good for real-time
- opencv - Fastest, less accurate
- ssd - Fast CNN-based
- dlib - Classic HOG-based
- yolov8 - YOLO-based detection
- centerface - Anchor-free detection
- skip - Skip detection (use full image)

Value

Character vector of detector names.

Examples

```
available_detectors()
```

`available_models` *List Available Face Recognition Models*

Description

Returns a character vector of available face recognition models that can be used for embedding extraction.

Usage

```
available_models()
```

Details

Available models with their embedding dimensions:

- VGG-Face (4096-d) - Legacy model
- Facenet (128-d) - Google's FaceNet
- Facenet512 (512-d) - FaceNet with larger embeddings
- OpenFace (128-d) - Open source
- DeepFace (4096-d) - Facebook's model
- DeepID (160-d) - Compact model
- ArcFace (512-d) - State-of-the-art, recommended
- Dlib (128-d) - Lightweight
- SFace (128-d) - Lightweight
- GhostFaceNet (512-d) - Efficient

Value

Character vector of model names.

Examples

```
available_models()
```

batch_embeddings *Extract Embeddings for Multiple Images*

Description

Efficiently extracts face embeddings from multiple images.

Usage

```
batch_embeddings(
  img_paths,
  model = "ArcFace",
  detector = "retinaface",
  align = TRUE,
  enforce_detection = TRUE,
  progress = TRUE
)
```

Arguments

img_paths	Character vector of paths to image files.
model	Character. The face recognition model to use. Default is "ArcFace".
detector	Character. Face detection backend. Default is "retinaface".
align	Logical. Whether to align faces. Default is TRUE.
enforce_detection	Logical. Error if face not detected. Default is TRUE.
progress	Logical. Show progress bar. Default is TRUE.

Value

A tibble with columns:

- img_path - Path to the image
- embedding - List column containing embedding vectors
- facial_area - List column with face bounding box
- face_confidence - Detection confidence
- success - Logical indicating if extraction succeeded
- error - Error message if extraction failed

Examples

```
## Not run:
# Get embeddings for all images in a folder
img_files <- list.files("faces/", pattern = "\\.jpg$", full.names = TRUE)
embeddings <- batch_embeddings(img_files)

# Access individual embeddings
embeddings$embedding[[1]]

## End(Not run)
```

`boot_winter_2ht` *Bootstrap Confidence Intervals for 2-HT Model Parameters*

Description

Computes bootstrap confidence intervals for the parameters of the 2-HT model by resampling the observed data with replacement.

Usage

```
boot_winter_2ht(
  x,
  nboot = 1000,
  conf_level = 0.95,
  lineup_size = 6,
  method = c("percentile", "bca"),
  parallel = FALSE,
  ncpus = 2,
  seed = NULL,
  ...
)
```

Arguments

`x`	A winter_2ht object from fit_winter_2ht(), OR count data/data frame as in fit_winter_2ht().
`nboot`	Number of bootstrap samples. Default is 1000.
`conf_level`	Confidence level for intervals. Default is 0.95.
`lineup_size`	Lineup size (only needed if x is count data, not a winter_2ht object).
`method`	Type of bootstrap CI: "percentile" (default) or "bca" (bias-corrected and accelerated).
`parallel`	Logical indicating whether to use parallel processing. Default is FALSE.
`ncpus`	Number of CPUs to use if parallel = TRUE. Default is 2.
`seed`	Random seed for reproducibility. Default is NULL.
`...`	Additional arguments passed to fit_winter_2ht().

Details

The bootstrap procedure resamples within TP and TA strata with replacement to keep both experimental totals fixed. For each bootstrap sample, the 2-HT model is re-fitted, and the distribution of parameter estimates across bootstrap samples is used to construct confidence intervals.

Two types of confidence intervals are available:

- **percentile**: Uses quantiles of the bootstrap distribution
- **bca**: Bias-corrected and accelerated intervals (Efron, 1987). The bias correction counts bootstrap ties with the estimate as half, and the acceleration comes from the jackknife over individual observations, centered and scaled within TP/TA strata, using at most six distinct leave-one-out response-count fits.

Value

An object of class "winter_2ht_boot" containing:

`boot_estimates`	Matrix of bootstrap parameter estimates (nboot x 4)
`ci`	Matrix of confidence intervals for each parameter
`original_fit`	The original model fit
`nboot`	Number of bootstrap samples
`conf_level`	Confidence level used

References

Efron, B. (1987). Better bootstrap confidence intervals. *Journal of the American Statistical Association, 82*(397), 171-185.

Examples

```
counts <- c(
  n_tp_suspect = 147, n_tp_filler = 94, n_tp_reject = 141,
  n_ta_suspect = 38, n_ta_filler = 138, n_ta_reject = 206
)
fit <- fit_winter_2ht(counts, lineup_size = 6)

# Bootstrap CIs
boot_fit <- boot_winter_2ht(fit, nboot = 200, seed = 123)
print(boot_fit)
plot(boot_fit)
```

bootstrap_anri *Bootstrap Confidence Intervals for ANRI*

Description

Computes bootstrap confidence intervals around ANRI estimates using percentile method.

Usage

```
bootstrap_anri(
  data,
  confidence_bins,
  choosers_only = TRUE,
  lineup_size = 6,
  n_bootstrap = 1000,
  conf_level = 0.95,
  seed = NULL,
  confidence_scale = c("auto", "0-1", "0-100"),
  innocent_suspect = "auto"
)
```

Arguments

data	Dataframe with standard lineup format
confidence_bins	Numeric vector of bin edges
choosers_only	Logical. Whether to analyze only suspect IDs (default = TRUE)
lineup_size	Integer. Lineup size (default = 6)
n_bootstrap	Integer. Number of bootstrap replications (default = 1000)
conf_level	Numeric. Confidence level for CIs (default = 0.95)
seed	Integer. Random seed for reproducibility (default = NULL)
confidence_scale	How the confidence scale is interpreted: "auto" (default), "0-1", or "0-100". See make_calibration_data.
innocent_suspect	How innocent-suspect identifications in target-absent lineups are obtained. "designated": the lineups had a designated innocent suspect, so target-absent "suspect" rows are counted directly, and zero such rows means a false-ID rate of zero. "estimated": there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by lineup_size. "auto" (the default) chooses "designated" when any target-absent row is a suspect ID or the data have a non-missing suspect_pos column, and "estimated" otherwise, with a message.

Details

Bootstrap resampling procedure: 1. Resample observations with replacement 2. Compute ANRI for each bootstrap sample 3. Construct percentile-based confidence interval

The percentile method uses the empirical quantiles of the bootstrap distribution. For a 95

Value

A list containing:

- anri: Point estimate of ANRI
- nri: Point estimate of NRI
- ci_lower: Lower bound of bootstrap CI
- ci_upper: Upper bound of bootstrap CI
- conf_level: Confidence level used
- n_bootstrap: Number of bootstrap replications
- bootstrap_distribution: Vector of bootstrap ANRI values

References

Efron, B., & Tibshirani, R. J. (1994). *An Introduction to the Bootstrap*. Chapman & Hall/CRC.

Examples

```
set.seed(123)
n <- 200
data <- data.frame(
  target_present = rep(c(TRUE, FALSE), each = n / 2),
  identification = sample(c("suspect", "filler", "reject"), n,
                          replace = TRUE, prob = c(0.5, 0.25, 0.25)),
  confidence = round(runif(n, 0, 100))
)
boot_result <- bootstrap_anri(data, confidence_bins = seq(0, 100, 20),
                              n_bootstrap = 200, seed = 1)
c(boot_result$ci_lower, boot_result$ci_upper)
```

---

calibration_bayes *Bayesian Calibration Analysis (Beta-Binomial Model)*

Description

Computes Bayesian posterior distributions for the calibration statistic C, over/underconfidence (O/U), and the Normalized Resolution Index (NRI) by placing independent Beta priors on the accuracy rate in each confidence bin.

Usage

```
calibration_bayes(
  data,
  confidence_bins = NULL,
  choosers_only = TRUE,
  lineup_size = 6,
  alpha = 0.5,
  S = 10000,
  credible_mass = 0.95,
  confidence_scale = c("auto", "0-1", "0-100"),
  innocent_suspect = "auto"
)
```

Arguments

`data`	A dataframe with columns `target_present` (logical), `identification` (character: "suspect", "filler", "reject"), and `confidence` (numeric).
`confidence_bins`	Numeric vector of bin edges for confidence (e.g., `c(0, 60, 80, 100)`). If `NULL` (default), treats each unique confidence level as its own bin.
`choosers_only`	Logical. If `TRUE` (default), only suspect identifications are included (standard eyewitness calibration practice).
`lineup_size`	Integer. Number of lineup members (default 6).
`alpha`	Prior concentration for the Beta prior on per-bin accuracy. Default 0.5 (Jeffreys). Named shortcuts: `"jeffreys"`, `"uniform"`, `"weak"`.
`S`	Number of posterior draws per bin. Default 10000.
`credible_mass`	Width of equal-tailed credible intervals. Default 0.95.
`confidence_scale`	How the confidence scale is interpreted: "auto" (default), "0-1", or "0-100". See `make_calibration_data`.
`innocent_suspect`	How innocent-suspect identifications in target-absent lineups are obtained. `"designated"`: the lineups had a designated innocent suspect, so target-absent `"suspect"` rows are counted directly, and zero such rows means a false-ID rate of zero. `"estimated"`: there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by `lineup_size`. `"auto"` (the default) chooses `"designated"` when any target-absent row is a suspect ID or the data have a non-missing `suspect_pos` column, and `"estimated"` otherwise, with a message.

Details

For each confidence bin j with n_j responses and k_j correct identifications, the posterior accuracy is:

$$a_j \mid \mathbf{n} \sim \text{Beta}(k_j + \alpha,\ n_j - k_j + \alpha).$$

For each posterior draw s, the calibration statistics are computed:

$$C^{(s)} = \sum_j \frac{n_j}{N}(c_j - a_j^{(s)})^2$$

$$OU^{(s)} = \bar{c} - \sum_j \frac{n_j}{N} a_j^{(s)}$$

$$NRI^{(s)} = \frac{\sum_j \frac{n_j}{N}(a_j^{(s)} - \bar{a}^{(s)})^2}{\bar{a}^{(s)}(1 - \bar{a}^{(s)})}$$

where c_j is the mean confidence (proportion scale) in bin j and $\bar{c}$ is the overall mean confidence. This model assumes exchangeable Bernoulli correctness within each bin. In experiments with fixed TP and TA totals, conditioning on the number of identifications in a bin does not generally give a binomial correct count. These intervals are therefore conditional working-model summaries, rather than automatically exact posterior inference for that fixed-arm design. If innocent IDs are estimated from fillers, the fractional effective counts produce a working Beta posterior, not the exact likelihood for observed designated-suspect counts. Its intervals do not propagate all uncertainty in that estimation. Bin frequencies and mean confidence are conditioned on.

Value

An object of class `"calibration_bayes"` containing:

C_draws, OU_draws, NRI_draws Posterior draws of each statistic.

C_mean, C_median, C_ci Posterior mean, median, and credible interval for C.

OU_mean, OU_median, OU_ci Same for O/U.

NRI_mean, NRI_median, NRI_ci Same for NRI.

point_estimates Frequentist C, OU, NRI from `make_calibration_data()`.

bin_data Per-bin summary (n, n_correct, mean_confidence, accuracy).

prior_alpha, S, credible_mass, n_total Metadata.

References

Juslin, P., Olsson, N., & Winman, A. (1996). Calibration and diagnosticity of confidence in eyewitness identification. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 22*(5), 1304-1316.

Brewer, N., & Wells, G. L. (2006). The confidence-accuracy relationship in eyewitness identification. *Journal of Experimental Psychology: Applied, 12*(1), 11-30.

See Also

`make_calibration_data`, `esize_T_bayes`

Examples

```
set.seed(42)
n <- 200
conf  <- sample(c(50, 70, 90), n, replace = TRUE)
tp    <- sample(c(TRUE, FALSE), n, replace = TRUE)
ident <- ifelse(tp, "suspect", sample(c("suspect","filler","reject"), n, replace=TRUE))
df    <- data.frame(target_present = tp, identification = ident, confidence = conf)
res   <- calibration_bayes(df, confidence_bins = c(0, 60, 80, 100))
print(res)
plot(res)
```

---

check_python_deps *Check Python Dependencies*

Description

Verifies that all required Python packages are installed and reports their versions.

Usage

```
check_python_deps(verbose = TRUE)
```

Arguments

verbose	Logical. If TRUE, prints detailed information.

Value

A tibble with package names, installed status, and versions. Its "face_mesh" attribute says whether get_face_landmarks() can run: "solutions", "tasks" or "unavailable".

Examples

```
## Not run:
check_python_deps()

## End(Not run)
```

check_pywitness_deps *Check Availability of the pyWitness Engine*

Description

Reports whether Python and pyWitness are available without initializing Python by default.

Usage

```
check_pywitness_deps(
  envname = .pywitness_default_env,
  initialize = FALSE,
  verbose = TRUE
)
```

Arguments

envname	Name or path of the intended virtual environment.
initialize	Logical. If TRUE, select envname, initialize Python, and import pyWitness. The default performs no initialization.
verbose	Logical. Print a concise status report.

Value

A one-row tibble with environment and engine status, including whether the imported version encodes the audited revision when initialize = TRUE.

Examples

```
check_pywitness_deps()
## Not run:
check_pywitness_deps(initialize = TRUE)

## End(Not run)
```

chi_diag *Chi-squared estimate of homogeneity of diagnosticity ratio*

Description

Function for getting chi-squared value for homogeneity of diagnosticity ratios

Usage

```
chi_diag(df)
```

Arguments

df	A dataframe containing: ln(d), variance of ln(d), d weights

Details

- To compute linedf, use the diag_param helper function.
- To compute df, apply ln_diag_ratio, var_lnd & d_weights functions to linedf, then bind results into one dataframe (see **Examples**)
- The order in which the estimates are bound together (i.e., their position in the dataframe) is important, and should always be as follows: row 1: *var*, row 2: *lnd*, row 3: *wi*.

Value

Chi squared estimate of homogeneity of diagnosticity ratios for k independent lineups

References

Malpass, R. S. (1981). Effective size and defendant bias in eyewitness identification lineups. *Law and Human Behavior, 5*(4), 299-309.

Malpass, R. S., Tredoux, C., & McQuiston-Surrett, D. (2007). Lineup construction and lineup fairness. In R. Lindsay, D. F. Ross, J. D. Read, & M. P. Toglia (Eds.), *Handbook of Eyewitness Psychology, Vol. 2: Memory for people* (pp. 155-178). Mahwah, NJ: Lawrence Erlbaum Associates.

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22*(2), 217-237.

Tredoux, C. (1999). Statistical considerations when determining measures of lineup size and lineup bias. *Applied Cognitive Psychology, 13*, S9-S26.

Wells, G. L., Leippe, M. R., & Ostrom, T. M. (1979). Guidelines for empirically assessing the fairness of a lineup. *Law and Human Behavior, 3*(4), 285-293.

Examples

```
#Target present data:
A <- rep(1:6, length.out = 100)
B <- rep(1:5, length.out = 70)
C <- rep(1:4, length.out = 20)
lineup_pres_list <- list(A, B, C)
rm(A, B, C)


#Target absent data:
A <- rep(6:1, length.out = 100)
B <- rep(5:1, length.out = 70)
C <- rep(4:1, length.out = 20)
lineup_abs_list <- list(A, B, C)
rm(A, B, C)

# Suspect position for each TP/TA pair
pos_list <- c(3, 2, 1)
```

```
#Nominal size:
k <- c(6, 5, 4)

#Use diag param helper function to get data (n11, n21, n12, n22):
linedf <- diag_param(lineup_pres_list, lineup_abs_list, pos_list, k)

#Get ln(d), variance of ln(d) & d weights:
ratio <- ln_diag_ratio(linedf, correction = FALSE)
var <- var_lnd(linedf)
wi <- d_weights(linedf)

#Bind estimates into one df of 3 rows & x observations
#(see Details above)
df <- t(cbind(var, ratio, wi))

#Call:
chi_diag(df)
```

`classify_lineup_choices` *Classify Raw Lineup Choices into Identification Outcomes*

Description

Derives an identification outcome — suspect identification, foil identification, or rejection — from the raw record of which lineup member each witness actually chose.

Usage

```
classify_lineup_choices(
  data,
  member_col = "lineup_member",
  suspect_col = "suspect_pos",
  into = "identification",
  k = NULL,
  overwrite = FALSE
)
```

Arguments

`data`	A data frame of lineup trials, one row per witness.
`member_col`	Name of the column holding the position of the lineup member chosen, with `0` (or `NA`) denoting a rejection. Defaults to `"lineup_member"`.
`suspect_col`	Either the name of a column holding the suspect's position for each trial, or a single number giving a suspect position common to all trials. Defaults to `"suspect_pos"`.

into	Name of the column to write the derived outcome into. Defaults to "identification".
k	Optional lineup size. When supplied, choices are checked to lie in 0:k and the positions of any null foils are attached to the result as the "null_foils" attribute.
overwrite	Logical. Whether to overwrite into if it already exists. Defaults to FALSE, which raises an error instead.

Details

Lineup data should record *which member* a witness chose, not merely whether the choice happened to fall on the suspect. Foils are not interchangeable: they differ in how plausible they are, and a foil that no witness ever chooses is not a functioning alternative to the suspect. Storing only a collapsed "suspect / filler / reject" label discards the information needed to see that.

This function treats the collapsed label as *derived*. Raw data keeps the chosen position; classify_lineup_choices() produces the label on demand, so both are available and the two cannot drift apart.

The outcome is "suspect" when the chosen member is the suspect, "reject" when no member was chosen, and "filler" otherwise. These are the labels used throughout the package, so the result can be passed straight to make_roc, make_cac and the other analysis functions.

Value

data with the derived outcome column added. If k was supplied, the "null_foils" attribute gives the positions that no witness chose.

References

Malpass, R. S. (1981). Effective size and defendant bias in eyewitness identification lineups. *Law and Human Behavior, 5*(4), 299-309.

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22*(2), 217-237.

See Also

lineup_table for a tabulation that shows null foils, and simulate_lineup_data, which records lineup_member directly.

Examples

```
# Raw choices from an 8-person lineup in which the suspect stands at
# position 3; 0 records a rejection
trials <- data.frame(
  lineup_member = c(3, 0, 5, 3, 1, 0, 3, 7),
  suspect_pos   = 3
)

classify_lineup_choices(trials, k = 8)

# A suspect position common to every trial can be given as a number
classify_lineup_choices(
```

```
  data.frame(lineup_member = c(2, 0, 4)),
  suspect_col = 2
)
```

`compare_anri` *Compare ANRI Between Groups with Bootstrap*

Description

Computes ANRI for two groups and tests whether they differ using bootstrap confidence intervals for the difference.

Usage

```
compare_anri(
  data,
  group_var,
  confidence_bins,
  choosers_only = TRUE,
  lineup_size = 6,
  n_bootstrap = 1000,
  conf_level = 0.95,
  seed = NULL,
  confidence_scale = c("auto", "0-1", "0-100"),
  innocent_suspect = "auto"
)
```

Arguments

`data`	Dataframe with standard lineup format plus a grouping variable
`group_var`	Character. Name of grouping variable (must have exactly 2 levels)
`confidence_bins`	Numeric vector of bin edges
`choosers_only`	Logical. Whether to analyze only suspect IDs (default = TRUE)
`lineup_size`	Integer. Lineup size (default = 6)
`n_bootstrap`	Integer. Number of bootstrap replications (default = 1000)
`conf_level`	Numeric. Confidence level for CIs (default = 0.95)
`seed`	Integer. Random seed for reproducibility (default = NULL)
`confidence_scale`	How the confidence scale is interpreted: "auto" (default), "0-1", or "0-100". See `make_calibration_data`.

`innocent_suspect`
: How innocent-suspect identifications in target-absent lineups are obtained. `"designated"`: the lineups had a designated innocent suspect, so target-absent `"suspect"` rows are counted directly, and zero such rows means a false-ID rate of zero. `"estimated"`: there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by `lineup_size`. `"auto"` (the default) chooses `"designated"` when any target-absent row is a suspect ID or the data have a non-missing `suspect_pos` column, and `"estimated"` otherwise, with a message.

Details

This function: 1. Computes ANRI separately for each group 2. Bootstraps each group independently 3. Computes bootstrap distribution of the difference 4. Tests H0: ANRI_1 = ANRI_2 using CI for difference

Interpretation:

- Positive difference: Group 1 has higher resolution
- Negative difference: Group 2 has higher resolution
- CI excludes 0: Statistically significant difference

Value

A list containing:

- anri_group1: ANRI for first group
- anri_group2: ANRI for second group
- difference: Point estimate of difference (group1 - group2)
- ci_lower: Lower bound of CI for difference
- ci_upper: Upper bound of CI for difference
- significant: Whether difference is significant (CI excludes 0)
- group_names: Names of the two groups
- bootstrap_results_group1: Bootstrap object for group 1
- bootstrap_results_group2: Bootstrap object for group 2

Examples

```
set.seed(123)
n <- 300
data <- data.frame(
  target_present = rep(c(TRUE, FALSE), each = n / 2),
  identification = sample(c("suspect", "filler", "reject"), n,
                          replace = TRUE, prob = c(0.5, 0.25, 0.25)),
  confidence = round(runif(n, 0, 100)),
  condition = rep(c("sequential", "simultaneous"), times = n / 2)
)
cmp <- compare_anri(data, group_var = "condition",
                    confidence_bins = seq(0, 100, 20),
```

```
                    n_bootstrap = 100, seed = 1)
cmp$difference
c(cmp$ci_lower, cmp$ci_upper)
```

---

compare_dpp *Compare DPP Between Two Procedures*

Description

Computes average DPP for two lineup procedures and tests their difference with a nonparametric bootstrap, as Smith et al. (2019) did.

Usage

```
compare_dpp(
  data_proc_a,
  data_proc_b,
  lineup_size = 6,
  alpha = NULL,
  beta = NULL,
  n_bootstrap = 1000,
  conf_level = 0.95,
  seed = NULL,
  innocent_suspect = "auto"
)
```

Arguments

data_proc_a	Dataframe for procedure A (standard lineup format)
data_proc_b	Dataframe for procedure B (standard lineup format)
lineup_size	Integer. Lineup size (default = 6)
alpha, beta	Optional Equation 3 weights; see make_dpp.
n_bootstrap	Integer. Number of bootstrap samples (default = 1000). Use 0 to skip the bootstrap.
conf_level	Numeric. Confidence level for the interval (default = 0.95)
seed	Integer. Random seed for reproducibility (default = NULL)
innocent_suspect	How innocent-suspect identifications in target-absent lineups are obtained. "designated": the lineups had a designated innocent suspect, so target-absent "suspect" rows are counted directly, and zero such rows means a false-ID rate of zero. "estimated": there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by lineup_size. "auto" (the default) chooses "designated" when any target-absent row is a suspect ID or the data have a non-missing suspect_pos column, and "estimated" otherwise, with a message.

Details

Lower DPP indicates better performance, so a negative difference means procedure A is better. Each bootstrap replicate resamples target-present and target-absent trials separately within each procedure, so every replicate keeps the original design, and the two procedures are treated as independent samples. The innocent-suspect design and the confidence criteria are determined once on each full dataset and reused in every replicate, so every replicate averages over the same criteria.

Value

An object of class `"lineup_dpp_comparison"`, a list containing:

- dpp_a: Average DPP for procedure A
- dpp_b: Average DPP for procedure B
- dpp_difference: DPP_A - DPP_B (negative = A is better)
- se_difference: Bootstrap standard error of the difference
- ci_difference: Percentile bootstrap interval for the difference
- boot_differences: Bootstrap distribution of the difference
- dpp_obj_a, dpp_obj_b: Full `make_dpp()` objects
- n_bootstrap, conf_level: Settings used

References

Smith, A. M., Lampinen, J. M., Wells, G. L., Smalarz, L., & Mackovichova, S. (2019). Deviation from perfect performance measures the diagnostic utility of eyewitness lineups but partial area under the ROC curve does not. *Journal of Applied Research in Memory and Cognition, 8*(1), 50-59. doi:10.1016/j.jarmac.2018.09.003

Examples

```
data(lineup_example)
# Compare two halves of the example data (odd vs even rows)
odd <- seq(1, nrow(lineup_example), by = 2)
# 50 resamples keep the example quick; use 1000 or more in an analysis
cmp <- compare_dpp(lineup_example[odd, ], lineup_example[-odd, ],
                   n_bootstrap = 50, seed = 1)
cmp
```

`compare_eff_sizes.boot` *Comparing Effective Size: Base function for bootstrapping*

Description

A row-resampling statistic for two columns of lineup choices.

Usage

```
compare_eff_sizes.boot(linedf, d)
```

Arguments

linedf	A dataframe of lineup data. Must consist of 2 columns, each containing data for 2 independent lineups
d	Indices for bootstrap sample. Argument used by boot function to select samples for bootstrapping

Details

This statistic applies the same row indices to both columns. Using it with an ordinary row bootstrap therefore resamples paired observations. For independent lineup samples use `effsize_compare`, which resamples each sample independently. Bootstrap intervals remain approximations; inclusion or exclusion of zero does not establish calibration in small or nearly uniform samples.

References

Davison, A.C. & Hinkley, D.V. (1997). *Bootstrap methods and their application*. Cambridge University Press.

Malpass, R. S. (1981). Effective size and defendant bias in eyewitness identification lineups. *Law and Human Behavior, 5*(4), 299-309.

Malpass, R. S., Tredoux, C., & McQuiston-Surrett, D. (2007). Lineup construction and lineup fairness. In R. Lindsay, D. F. Ross, J. D. Read, & M. P. Toglia (Eds.), *Handbook of Eyewitness Psychology, Vol. 2: Memory for people* (pp. 155-178). Mahwah, NJ: Lawrence Erlbaum Associates.

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22*(2), 217-237.

Tredoux, C. (1999). Statistical considerations when determining measures of lineup size and lineup bias. *Applied Cognitive Psychology, 13*, S9-S26.

Wells, G. L.,Leippe, M. R., & Ostrom, T. M. (1979). Guidelines for empirically assessing the fairness of a lineup. *Law and Human Behavior, 3*(4), 285-293.

Examples

```
# Two independent lineups of 6 members, 50 mock witnesses each
set.seed(1)
linedf <- data.frame(lineup1 = sample(1:6, 50, replace = TRUE),
                     lineup2 = sample(1:6, 50, replace = TRUE))
# Difference in effective size for the observed data
compare_eff_sizes.boot(linedf, seq_len(nrow(linedf)))
# Resample the independent lineups separately
effsize_compare(linedf, R = 100)
```

compare_max_sdt *Compare Two Nested MAX SDT Models*

Description

Performs a chi-squared difference test comparing a less constrained (free) MAX SDT model to a more constrained (nested) model. Both models must be fitted to the same data.

Usage

```
compare_max_sdt(free, constrained)
```

Arguments

free	A "max_sdt_fit" object: the less constrained model.
constrained	A "max_sdt_fit" object: the more constrained model. Must have fewer free parameters than free.

Details

The chi-squared difference statistic is:

$$\Delta\chi^2 = \chi^2_{\text{constrained}} - \chi^2_{\text{free}}$$

with degrees of freedom equal to the difference in the number of free parameters. A significant result indicates that the constraint worsens fit. The fits must use identical counts and lineup sizes, must have converged, and the constrained model must impose every constraint in the free model plus at least one additional constraint.

Value

An object of class "max_sdt_compare" with the test result.

See Also

fit_max_sdt

Examples

```
# Each two-condition fit takes a few seconds
fit_free <- fit_max_sdt(
  n_hit=69, n_tp_choose=82, n_fa=64, N_tp=96, N_ta=106,
  n_hit_2=68, n_tp_choose_2=78, n_fa_2=42, N_tp_2=96, N_ta_2=96
)
fit_eqd <- fit_max_sdt(
  n_hit=69, n_tp_choose=82, n_fa=64, N_tp=96, N_ta=106,
  n_hit_2=68, n_tp_choose_2=78, n_fa_2=42, N_tp_2=96, N_ta_2=96,
  constrain_d = TRUE
)
```

```
compare_max_sdt(fit_free, fit_eqd)
```

`compare_models` *Compare Multiple Models for Lineup Identification Data*

Description

Fits and compares multiple models (2-HT, EIG, Full ROC) to the same lineup identification dataset, providing a side-by-side descriptive table.

Usage

```
compare_models(
  data,
  models = c("2ht", "eig", "fullroc"),
  lineup_size = 6,
  prior_guilt = 0.5,
  confidence_bins = NULL,
  show_warnings = FALSE,
  ...
)
```

Arguments

`data`	A dataframe with columns: target_present, identification, confidence
`models`	Character vector of models to fit. Options: • "2ht" or "winter": Winter et al. (2022) Two-High Threshold MPT model • "eig": Expected Information Gain (Starns et al., 2023) • "fullroc": Full ROC curve (Smith & Yang, 2020) Default = c("2ht", "eig", "fullroc") fits all models.
`lineup_size`	Integer. Number of people in the lineup (default = 6)
`prior_guilt`	Numeric. Prior probability of guilt for EIG (default = 0.5)
`confidence_bins`	Numeric vector of confidence bin edges (optional). Used for EIG and Full ROC if specified.
`show_warnings`	Logical. Whether to show model fitting warnings (default = FALSE)
`...`	Additional arguments passed to individual model functions

Details

This function provides a unified interface for fitting and comparing multiple eyewitness identification models. It automatically handles different data requirements and output formats across models.

Models Compared:

- **2-HT (Winter et al., 2022)**: Multinomial processing tree model with parameters for detection (dP, dA), bias (b), and guessing (g). Provides AIC/BIC for comparison only with other likelihood models fit to the same observations and outcome representation.
- **EIG (Starns et al., 2023)**: Information-theoretic measure of evidentiary value. Higher values indicate more diagnostic procedures.
- **Full ROC (Smith & Yang, 2020)**: Uses ALL responses to compute investigator discriminability. AUC ranges from 0 to 1; 0.5 represents chance under a fixed ordering.

Interpretation:

- AIC/BIC describe the 2-HT likelihood fit but cannot be compared with EIG or full-ROC AUC
- Use EIG for comparing procedure diagnosticity (higher is better)
- Use Full ROC AUC for investigator discriminability (higher is better)

Value

An object of class "model_comparison" containing:

- comparison_table: Dataframe comparing model fits
- fitted_models: List of fitted model objects
- best_model: Always NULL. The included methods estimate different quantities and cannot be ranked by a common information criterion.
- data: The input data
- models_fit: Character vector of models successfully fit

References

Winter, K., Menne, N. M., Bell, R., & Buchner, A. (2022). Experimental validation of a multinomial processing tree model for analyzing eyewitness identification decisions. *Scientific Reports, 12*, 15571.

Starns, J. J., Cohen, A. L., & Rotello, C. M. (2023). A complete method for assessing the effectiveness of eyewitness identification procedures: Expected information gain. *Psychological Review, 130*(3), 677–719.

Smith, A. M., Yang, Y., & Wells, G. L. (2020). Distinguishing between investigator discriminability and eyewitness discriminability. *Perspectives on Psychological Science, 15*(3), 589-607.

Examples

```
data(lineup_example)
# Fit all models
comparison <- compare_models(lineup_example)
print(comparison)
summary(comparison)

# Fit specific models
comparison <- compare_models(lineup_example, models = c("2ht", "eig"))

# Access individual fitted models
comparison$fitted_models$`2ht`
comparison$fitted_models$eig
```

---

compare_multid *Compare Multi-d' Indices Between Two Conditions*

Description

Computes the difference (x minus y) on every Multi-d' index, for example simultaneous versus sequential lineups, or high- versus low-similarity lineups. When both objects carry bootstrap draws, standard errors, percentile intervals and normal-approximation p-values are given for each difference. The two conditions are treated as independent samples: the standard error is the square root of the sum of the two bootstrap variances, and the interval is taken from differences between randomly re-paired draws, so the result does not depend on whether the two fits used the same seed.

Usage

```
compare_multid(x, y, labels = c("x", "y"), seed = NULL)
```

Arguments

x, y	Objects of class "multid" from make_multid, fitted with the same filler_rate.
labels	Names for the two conditions.
seed	Optional random seed for re-pairing the bootstrap draws.

Value

A data frame of class "multid_comparison".

Examples

```
# Simultaneous vs sequential lineups, Lee & Penrod's (2019) Table 3
sim <- make_multid(counts = c(G = 52, Fp = 25, Rp = 24, I = 28, Fa = 26, Ra = 46),
                   correction = "none")
sqn <- make_multid(counts = c(G = 44, Fp = 19, Rp = 39, I = 15, Fa = 17, Ra = 68),
                   correction = "none")
compare_multid(sim, sqn, labels = c("simultaneous", "sequential"))
```

compare_pauc *Compare pAUC Between Two ROC Curves*

Description

Statistically compares partial Area Under the Curve (pAUC) values between two lineup conditions using bootstrap-based standard errors and z-tests. Follows the methodology from pyWitness (Mickes et al., 2024).

Usage

```
compare_pauc(
  data1,
  data2,
  lineup_size = 6,
  max_false_id_rate = NULL,
  n_bootstrap = 2000,
  conf_level = 0.95,
  seed = NULL,
  label1 = "Condition 1",
  label2 = "Condition 2",
  innocent_suspect = "auto"
)
```

Arguments

data1	First dataset (dataframe with target_present, identification, confidence)
data2	Second dataset (same format as data1)
lineup_size	Integer. Number of people in lineup (default = 6)
max_false_id_rate	Numeric. Upper limit of the false-ID rate region over which partial AUC is computed. Default NULL uses the largest region *both* conditions actually reach, i.e. the smaller of their two maximum observed false-ID rates. Supplying a larger value means at least one condition is integrated only as far as its own data extend, so the two areas cover different intervals; a warning is issued in that case.
n_bootstrap	Integer. Number of bootstrap samples (default = 2000)
conf_level	Numeric. Confidence level for intervals (default = 0.95)
seed	Integer. Random seed for reproducibility (default = NULL)
label1	Character. Label for first condition (default = "Condition 1")
label2	Character. Label for second condition (default = "Condition 2")
innocent_suspect	How innocent-suspect identifications in target-absent lineups are obtained. "designated": the lineups had a designated innocent suspect, so target-absent "suspect" rows

are counted directly, and zero such rows means a false-ID rate of zero. `"estimated"`: there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by `lineup_size`. `"auto"` (the default) chooses `"designated"` when any target-absent row is a suspect ID or the data have a non-missing `suspect_pos` column, and `"estimated"` otherwise, with a message.

Details

This function implements an approximate statistical test for comparing ROC curves between two conditions (e.g., simultaneous vs. sequential lineups, different retention intervals, etc.).

Method:

1. Computes pAUC for each condition (optionally up to a maximum false ID rate)
2. Uses bootstrap resampling to estimate standard errors
3. Computes Z-statistic: Z = (pAUC1 - pAUC2) / SE(pAUC1 - pAUC2)
4. Calculates p-value from standard normal distribution

The two datasets are treated as independent conditions. This function does not implement a paired or clustered bootstrap.

Interpretation:

- Positive pAUC difference: Condition 1 has better discriminability
- Negative pAUC difference: Condition 2 has better discriminability
- $p < 0.05$: Significant difference between conditions

The independent-sample bootstrap estimates uncertainty in an empirical partial-area functional; the final comparison uses a normal/Wald interval. It is approximate and can be liberal in finite samples. A data-selected cutoff, sparse high-confidence cells, or resamples that do not reach the cutoff require additional caution. It is not an exact discrimination test.

Value

An object of class "pauc_comparison" containing:

- pauc1: pAUC for condition 1
- pauc2: pAUC for condition 2
- pauc_diff: Difference in pAUC (pauc1 - pauc2)
- se_pauc1: Bootstrap standard error for pAUC1
- se_pauc2: Bootstrap standard error for pAUC2
- se_diff: Standard error of the difference
- z_score: Z-statistic for difference test
- p_value: Two-tailed p-value
- ci_diff: Confidence interval for difference
- roc1: ROC object for condition 1

- roc2: ROC object for condition 2
- max_false_id_rate: Cutoff used for pAUC
- label1, label2: Condition labels
- n_bootstrap: Number of bootstrap samples used

References

Mickes, L., Seale-Carlisle, T. M., Chen, X., & Boogert, S. (2024). pyWitness 1.0: A python eyewitness identification analysis toolkit. *Behavior Research Methods, 56*, 1533-1550.

Wixted, J. T., & Mickes, L. (2012). The field of eyewitness memory should abandon probative value and embrace receiver operating characteristic analysis. *Perspectives on Psychological Science, 7*(3), 275-278.

Examples

```
# Compare two lineup procedures (halves of the example data)
data(lineup_example)
odd <- seq(1, nrow(lineup_example), by = 2)
comparison <- compare_pauc(
  lineup_example[odd, ],
  lineup_example[-odd, ],
  label1 = "Procedure A",
  label2 = "Procedure B",
  n_bootstrap = 200,
  seed = 123
)

print(comparison)
plot(comparison)
```

`compare_sdt_summary` *Compare SDT Summary Metrics Between Two Conditions*

Description

Performs a large-sample z-test comparing d' (or criterion or ln beta) between two independent conditions using summary counts only.

Usage

```
compare_sdt_summary(
  hits_a,
  fas_a,
  misses_a,
  cr_a,
```

```
  hits_b,
  fas_b,
  misses_b,
  cr_b,
  metric = c("dprime", "c", "ln_beta"),
  method = c("miller", "gourevitch", "bootstrap"),
  correction = c("loglinear", "half", "none"),
  nboot = 1000,
  seed = NULL
)
```

Arguments

hits_a	Hits in condition A.
fas_a	False alarms in condition A.
misses_a	Misses in condition A.
cr_a	Correct rejections in condition A.
hits_b	Hits in condition B.
fas_b	False alarms in condition B.
misses_b	Misses in condition B.
cr_b	Correct rejections in condition B.
metric	Which metric to compare: "dprime", "c", or "ln_beta".
method	Variance method: "miller", "gourevitch", or "bootstrap".
correction	Correction for extreme rates ("loglinear", "half", or "none").
nboot	Number of bootstrap samples when method = "bootstrap".
seed	Optional random seed for bootstrap variance.

Value

A list with estimates, standard error, z, and p-value.

Examples

```
cmp <- compare_sdt_summary(hits_a = 70, fas_a = 20, misses_a = 30, cr_a = 80,
                           hits_b = 60, fas_b = 35, misses_b = 40, cr_b = 65,
                           metric = "dprime")
cmp$z
cmp$p_value
```

`compare_utility` *Compare Utilities for Two Procedures*

Description

Main wrapper function to compute and plot utility comparison between two lineup procedures.

Usage

```
compare_utility(
  data_proc_a,
  data_proc_b,
  base_rate_grid = seq(0.01, 0.99, 0.01),
  utility_matrix = c(tp = 1, fn = -0.5, fp = -2, tn = 0.5),
  utility_type = c("max", "avg", "all"),
  show_plot = TRUE,
  innocent_suspect = "auto",
  ...
)
```

Arguments

`data_proc_a`	Dataframe for procedure A
`data_proc_b`	Dataframe for procedure B
`base_rate_grid`	Numeric vector of base rates (default: seq(0.01, 0.99, 0.01))
`utility_matrix`	Named vector of utilities (default: c(tp=1, fn=-0.5, fp=-2, tn=0.5))
`utility_type`	Character. "max", "avg", or "all" (default = "max")
`show_plot`	Logical. Whether to display plot (default = TRUE)
`innocent_suspect`	How innocent-suspect identifications in target-absent lineups are obtained. `"designated"`: the lineups had a designated innocent suspect, so target-absent `"suspect"` rows are counted directly, and zero such rows means a false-ID rate of zero. `"estimated"`: there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by `lineup_size`. `"auto"` (the default) chooses `"designated"` when any target-absent row is a suspect ID or the data have a non-missing `suspect_pos` column, and `"estimated"` otherwise, with a message.
`...`	Additional arguments passed to plot_utility_difference()

Value

A list containing difference curve data and plot

Examples

```
data(lineup_example)
odd <- seq(1, nrow(lineup_example), by = 2)
comparison <- compare_utility(lineup_example[odd, ],
                              lineup_example[-odd, ],
                              base_rate_grid = seq(0.1, 0.9, 0.1),
                              utility_type = "max")
print(comparison)
```

`compute_anri` *Compute ANRI (Adjusted Normalized Resolution Index)*

Description

Computes ANRI, a bias-corrected version of NRI following Yaniv et al. (1991). ANRI adjusts NRI for the number of confidence bins and sample size, providing more accurate estimates especially when J (bins) is small relative to N.

Usage

```
compute_anri(
  data,
  confidence_bins,
  choosers_only = TRUE,
  lineup_size = 6,
  confidence_scale = c("auto", "0-1", "0-100"),
  innocent_suspect = "auto"
)
```

Arguments

`data`	A dataframe with the following columns: • target_present: Logical. TRUE if guilty suspect in lineup • identification: Character. "suspect", "filler", or "reject" • confidence: Numeric. Confidence rating
`confidence_bins`	Numeric vector of bin edges (required for ANRI)
`choosers_only`	Logical. Whether to analyze only suspect IDs (default = TRUE)
`lineup_size`	Integer. Lineup size (default = 6)
`confidence_scale`	How the confidence scale is interpreted: "auto" (default), "0-1", or "0-100". See `make_calibration_data`.

`innocent_suspect`
: How innocent-suspect identifications in target-absent lineups are obtained. `"designated"`: the lineups had a designated innocent suspect, so target-absent `"suspect"` rows are counted directly, and zero such rows means a false-ID rate of zero. `"estimated"`: there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by `lineup_size`. `"auto"` (the default) chooses `"designated"` when any target-absent row is a suspect ID or the data have a non-missing `suspect_pos` column, and `"estimated"` otherwise, with a message.

Details

ANRI corrects NRI for small-sample and small-bin-count bias:

$$NRI = \frac{\frac{1}{N}\sum_{j=1}^{J} n_j(a_j - \bar{a})^2}{\bar{a}(1-\bar{a})}$$

$$ANRI = \frac{N \cdot NRI - J + 1}{N - J + 1}$$

Where:

- N = total sample size
- J = number of bins
- n_j = sample size in bin j
- a_j = accuracy (proportion correct) in bin j
- a_bar = overall accuracy

When to use ANRI vs NRI:

- Use ANRI when J is small (e.g., 3-5 bins)
- Use ANRI when N is modest (e.g., < 200)
- Use ANRI when comparing across different numbers of bins
- NRI is asymptotically unbiased as N approaches infinity

ANRI provides a less biased estimate of the population resolution, making it more appropriate for hypothesis testing and group comparisons. When target-absent suspect identifications are estimated from filler choices, `N` and the bin counts are effective (possibly fractional) counts from `make_calibration_data()`, so the correction is an approximation. See `make_calibration_data` for the designated-suspect distinction.

Value

A list containing:

- anri: Adjusted Normalized Resolution Index
- nri: Original Normalized Resolution Index (unadjusted)
- n_total: Total sample size

- n_bins: Number of confidence bins
- calibration_data: Per-bin accuracy and sample sizes
- overall_accuracy: Mean accuracy

References

Yaniv, I., Yates, J. F., & Smith, J. E. K. (1991). Measures of discrimination skill in probabilistic judgment. *Psychological Bulletin, 110*(3), 611-617.

Juslin, P., Olsson, N., & Winman, A. (1996). Calibration and diagnosticity of confidence in eyewitness identification. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 22*(5), 1304-1316.

Examples

```
set.seed(123)
n <- 200
data <- data.frame(
  target_present = rep(c(TRUE, FALSE), each = n / 2),
  identification = sample(c("suspect", "filler", "reject"), n,
                          replace = TRUE, prob = c(0.5, 0.25, 0.25)),
  confidence = round(runif(n, 0, 100))
)
result <- compute_anri(data, confidence_bins = seq(0, 100, 20))
result$anri
```

---

compute_dpp *Compute and Plot DPP*

Description

Main wrapper function to compute and plot DPP analysis.

Usage

```
compute_dpp(
  data,
  lineup_size = 6,
  show_plot = TRUE,
  alpha = NULL,
  beta = NULL,
  innocent_suspect = "auto",
  ...
)
```

Arguments

`data`	Dataframe with standard lineup format
`lineup_size`	Integer. Lineup size (default = 6)
`show_plot`	Logical. Whether to display plot (default = TRUE)
`alpha, beta`	Optional Equation 3 weights; see `make_dpp`.
`innocent_suspect`	How innocent-suspect identifications in target-absent lineups are obtained. `"designated"`: the lineups had a designated innocent suspect, so target-absent `"suspect"` rows are counted directly, and zero such rows means a false-ID rate of zero. `"estimated"`: there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by `lineup_size`. `"auto"` (the default) chooses `"designated"` when any target-absent row is a suspect ID or the data have a non-missing `suspect_pos` column, and `"estimated"` otherwise, with a message.
`...`	Additional arguments passed to plot_dpp()

Value

An object of class `"lineup_dpp"`: the `make_dpp()` result plus the plot.

Examples

```
data(lineup_example)
dpp_result <- compute_dpp(lineup_example, show_plot = FALSE)
print(dpp_result)
```

---

compute_eig *Compute Expected Information Gain (EIG)*

Description

Computes Expected Information Gain for lineup identification procedures following Starns et al. (2023). EIG quantifies the evidentiary value of witness responses using information theory.

Usage

```
compute_eig(
  eig_data,
  prior_guilt = 0.5,
  confidence_bins = NULL,
  lineup_size = 6,
  innocent_suspect = "auto"
)
```

Arguments

`eig_data`	List output from make_eig_data(), or a dataframe with columns: target_present, identification, confidence
`prior_guilt`	Numeric. Prior probability that suspect is guilty (default = 0.5)
`confidence_bins`	Numeric vector of bin edges (only used if eig_data is a dataframe)
`lineup_size`	Nominal lineup size used when raw target-absent data have no designated innocent suspect.
`innocent_suspect`	How innocent-suspect identifications in target-absent lineups are obtained. `"designated"`: the lineups had a designated innocent suspect, so target-absent `"suspect"` rows are counted directly, and zero such rows means a false-ID rate of zero. `"estimated"`: there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by `lineup_size`. `"auto"` (the default) chooses `"designated"` when any target-absent row is a suspect ID or the data have a non-missing `suspect_pos` column, and `"estimated"` otherwise, with a message.

Details

EIG measures how much information (in bits) a witness response provides about guilt vs. innocence. It is computed as:

$$EIG = \sum_x p(x) \times [H(prior) - H(p(guilty|x))]$$

where:

- x = response category (e.g., "suspect_high_confidence")
- p(x) = probability of that response
- H() = Shannon entropy
- p(guilty|x) = posterior probability of guilt given response x

Higher EIG values indicate more diagnostic procedures. EIG = 0 means no information gain (responses don't distinguish guilty from innocent). The maximum EIG is the prior entropy; it is 1 bit only at prior guilt 0.5.

Value

A list of class "lineup_eig" containing:

- eig: Expected Information Gain in bits
- response_data: Dataframe with information gain for each response category
- prior_guilt: Prior probability used
- prior_entropy: Entropy of prior (baseline uncertainty)
- n_guilty: Number of target-present lineups
- n_innocent: Number of target-absent lineups

References

Starns, J. J., Cohen, A. L., & Rotello, C. M. (2023). A complete method for assessing the effectiveness of eyewitness identification procedures: Expected information gain. *Psychological Review, 130*(3), 677–719.

Examples

```
# Compute EIG with binned confidence
data(lineup_example)
eig_result <- compute_eig(lineup_example,
                          prior_guilt = 0.5,
                          confidence_bins = c(0, 60, 80, 100))
print(eig_result)
```

---

cosine_to_similarity *Convert Cosine Distance to Similarity Score*

Description

Converts cosine distance to cosine similarity in [-1, 1].

Usage

```
cosine_to_similarity(distance)
```

Arguments

distance	Numeric. Cosine distance value(s).

Value

Numeric cosine similarity score(s), where 1 means the same direction, 0 means orthogonal, and -1 means opposite directions.

Examples

```
# Distance of 0 -> similarity of 1
cosine_to_similarity(0)  # Returns 1

# Distance of 0.5 -> similarity of 0.5
cosine_to_similarity(0.5)  # Returns 0.5
```

`create_example_lineup_data` *Create Example Lineup Data*

Description

Generate example lineup identification data in standardized format. Useful for testing, demonstrations, and learning r4lineups functions.

Usage

```
create_example_lineup_data(
  n_trials = 100,
  prop_target_present = 0.5,
  include_confidence = TRUE,
  include_response_time = FALSE,
  seed = NULL
)
```

Arguments

`n_trials`	Integer. Number of trials to generate (default = 100)
`prop_target_present`	Numeric. Proportion of target-present lineups (default = 0.5)
`include_confidence`	Logical. Include confidence ratings (default = TRUE)
`include_response_time`	Logical. Include response times (default = FALSE)
`seed`	Integer. Random seed for reproducibility (default = NULL)

Value

A lineup_data object with realistic example data

Examples

```
# Basic example data
example_data <- create_example_lineup_data(n_trials = 200, seed = 123)
print(example_data)

# Use in analysis
roc <- make_roc(example_data, lineup_size = 6)
```

d_bar *Mean diagnosticity ratio for k lineup pairs*

Description

Function for computing pooled estimator from a set of k diagnosticity ratios

Usage

```
d_bar(df)
```

Arguments

df	A dataframe containing rows: ln(d), variance of ln(d), d weights

Details

. The order in which the estimates are bound together (i.e., their position in the dataframe) is important, and should always be as follows:

- row 1: var
- row 2: lnd
- row 3: wi

Value

Mean diagnosticity ratio for k independent lineups

References

Malpass, R. S. (1981). Effective size and defendant bias in eyewitness identification lineups. *Law and Human Behavior, 5*(4), 299-309.

Malpass, R. S., Tredoux, C., & McQuiston-Surrett, D. (2007). Lineup construction and lineup fairness. In R. Lindsay, D. F. Ross, J. D. Read, & M. P. Toglia (Eds.), *Handbook of Eyewitness Psychology, Vol. 2: Memory for people* (pp. 155-178). Mahwah, NJ: Lawrence Erlbaum Associates.

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22*(2), 217-237.

Tredoux, C. (1999). Statistical considerations when determining measures of lineup size and lineup bias. *Applied Cognitive Psychology, 13*, S9-S26.

Wells, G. L.,Leippe, M. R., & Ostrom, T. M. (1979). Guidelines for empirically assessing the fairness of a lineup. *Law and Human Behavior, 3*(4), 285-293.

Examples

```
#Target present data:
A <- rep(1:6, length.out = 100)
B <- rep(1:5, length.out = 70)
C <- rep(1:4, length.out = 20)
lineup_pres_list <- list(A, B, C)
rm(A, B, C)


#Target absent data:
A <- rep(6:1, length.out = 100)
B <- rep(5:1, length.out = 70)
C <- rep(4:1, length.out = 20)
lineup_abs_list <- list(A, B, C)
rm(A, B, C)

# Suspect position for each TP/TA pair
pos_list <- c(3, 2, 1)

#Nominal size:
k <- c(6, 5, 4)

#Use diag param helper function to get data (n11, n21, n12, n22):
linedf <- diag_param(lineup_pres_list, lineup_abs_list, pos_list, k)

#Compute ln(d):
ratio <- ln_diag_ratio(linedf, correction = FALSE)

#Compute variance of ln(d):
var <- var_lnd(linedf)

#Compute weights for pooled estimator:
wi <- d_weights(linedf)

#Bind ln(d), variance of ln(d) and weights into one df (of 3 rows & x observations)
#(see Details above):
df <- t(cbind(var, ratio, wi))

#Call:
d_bar(df)
```

`d_weights` *Diagnosticity ratio weights*

Description

Function to compute weights of each diagnosticity ratio for k lineup pairs

Usage

```
d_weights(linedf)
```

Arguments

linedf	A dataframe of parameters for computing diagnosticity ratio

Details

In order to obtain a pooled estimate of a set of diagnosticity ratios, we use a weight for each ratio that is equal to the inverse of its variance.

To get linedf, use the diag_param helper function

diag_param returns a dataframe containing the following:

- *n11*: Number of mock witnesses who identified the suspect in the target present condition
- *n21*: Number of mock witnesses who did not identify the suspect in the target present condition
- *n12*: Number of mock witnesses who identified the suspect in the target absent condition
- *n22*: Number of mock witnesses who did not identify the suspect in the target absent condition

Value

A dataframe with one column containing weights for each pair of lineups for which diagnosticity ratio is being calculated.

References

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22*(2), 217-237.

Examples

```
#Target present data:
A <- rep(1:6, length.out = 100)
B <- rep(1:5, length.out = 70)
C <- rep(1:4, length.out = 20)
lineup_pres_list <- list(A, B, C)
rm(A, B, C)


#Target absent data:
A <- rep(6:1, length.out = 100)
B <- rep(5:1, length.out = 70)
C <- rep(4:1, length.out = 20)
lineup_abs_list <- list(A, B, C)
rm(A, B, C)

# Suspect position for each TP/TA pair
pos_list <- c(3, 2, 1)

#Nominal size:
k <- c(6, 5, 4)
```

```
#Use diag param helper function to get data (n11, n21, n12, n22):
linedf <- diag_param(lineup_pres_list, lineup_abs_list, pos_list, k)

#Call:
wi <- d_weights(linedf)
```

`datacheck1` *Helper function*

Description

Checks that number of lineup choices contained in a vector is accurate

Usage

```
datacheck1(lineup_vec, k)
```

Arguments

`lineup_vec`	A numeric vector of lineup choices
`k`	Number of members in lineup

Details

This function ensures that a non-selected lineup member is not accidentally omitted from the dataframe due to lack of selection by all mock witnesses. It functions as a check that the total number of lineup members is accurate.

Examples

```
# Choices of 50 mock witnesses to a 6-member lineup: passes silently
set.seed(1)
lineup_vec <- sample(1:6, 50, replace = TRUE)
datacheck1(lineup_vec, 6)
```

`datacheck2` *Helper function*

Description

Checks that number of lineup choices contained in a vector is accurate

Usage

```
datacheck2(lineup_pres, lineup_abs, k1, k2)
```

Arguments

lineup_pres	A numeric vector of lineup choices for a lineup in which the target was present
lineup_abs	A numeric vector of lineup choices for a lineup in which the target was absent
k1	Number of targets in TP lineup
k2	Number of targets in TA lineup

Details

This function ensures that a non-selected lineup member is not accidentally omitted from the dataframe due to lack of selection by all mock witnesses. It functions as a check that the total number of lineup members is accurate

Examples

```
# Target-present and target-absent lineups of 6 members each
set.seed(1)
lineup_pres <- sample(1:6, 50, replace = TRUE)
lineup_abs <- sample(1:6, 50, replace = TRUE)
datacheck2(lineup_pres, lineup_abs, 6, 6)
```

---

datacheck3 *Helper function*

Description

Checks that number of lineup choices contained in a data table is accurate

Usage

```
datacheck3(lineup_tab, k)
```

Arguments

lineup_tab	A table of lineup choices
k	Nominal size (i.e., total number of members in lineup)

Details

This function ensures that a non-selected lineup member is not accidentally omitted from the data due to lack of selection by all mock witnesses. It functions as a check that the total number of lineup members is accurate.

Examples

```
# Table of choices from 50 mock witnesses to a 6-member lineup
set.seed(1)
lineup_tab <- table(sample(1:6, 50, replace = TRUE))
datacheck3(lineup_tab, 6)
```

`detect_faces` *Detect Faces in an Image*

Description

Detects all faces in an image and returns their bounding boxes and facial landmarks.

Usage

```
detect_faces(img_path, detector = "retinaface")
```

Arguments

`img_path`	Path to the image file.
`detector`	Character. Face detection backend to use. Options: "retinaface" (default, most accurate), "mtcnn", "mediapipe", "opencv", "ssd", "dlib", "yolov8", "centerface".

Value

A list where each element represents a detected face with:

- facial_area - List with x, y, w, h coordinates
- landmarks - List with eye, nose, mouth positions (if available)
- confidence - Detection confidence score

Returns an empty list if no faces are detected.

Examples

```
## Not run:
# Detect faces in an image
faces <- detect_faces("group_photo.jpg")

# Number of faces found
length(faces)

# Get bounding box of first face
faces[[1]]$facial_area

## End(Not run)
```

diag_param *Parameters for diagnosticity ratio*

Description

This function calculates the parameters needed to calculate the diagnosticity ratio for several lineup pairs.

Usage

```
diag_param(lineup_pres_list, lineup_abs_list, pos_list, k)
```

Arguments

lineup_pres_list	A list containing k vectors of lineup choices for k lineups, in which the target was present
lineup_abs_list	A list containing k vectors of lineup choices for k lineups, in which the target was absent
pos_list	Suspect positions for each lineup pair. Supply a numeric vector with one position per pair when TP and TA suspect positions match, or a list whose elements contain either one shared position or two positions in TP, TA order.
k	A vector indexing number of members in each lineup pair. Must be specified by user (scalar).

Details

- Lineup pairs consist of one lineup in which the target was present (TP) and one lineup in which the target was absent (TA).
- Each lineup pair must occupy corresponding positions in the TA and TP lists.

  Example:

  For a lineup pair A that consists of (1)TP lineup and (2)TA lineup: A(1) is the first vector in the TP list A(2) is the first vector in the TA list
- The order in which nominal size for each lineup pair is listed must also correspond with the positions of each respective lineup in the lineup lists (i.e., if lineup 1 has k = 6, then the first element of vector 'k' = 6)
- Data must be in a list format. This allows the function to compare lineups in which the number of choices and number of lineup members differs.
- TP and TA vectors may have different sample sizes; they are counted independently and are never compared element by element.

Value

Returns a dataframe containing:

- *n11*: Number of mock witnesses who identified the suspect in the target present condition
- *n21*: Number of mock witnesses who did not identify the suspect in the target present condition
- *n12*: Number of mock witnesses who identified the suspect in the target absent condition
- *n22*: Number of mock witnesses who did not identify the suspect in the target absent condition

References

Malpass, R. S. (1981). Effective size and defendant bias in eyewitness identification lineups. *Law and Human Behavior, 5*(4), 299-309.

Malpass, R. S., Tredoux, C., & McQuiston-Surrett, D. (2007). Lineup construction and lineup fairness. In R. Lindsay, D. F. Ross, J. D. Read, & M. P. Toglia (Eds.), *Handbook of Eyewitness Psychology, Vol. 2: Memory for people* (pp. 155-178). Mahwah, NJ: Lawrence Erlbaum Associates.

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22*(2), 217-237.

Tredoux, C. (1999). Statistical considerations when determining measures of lineup size and lineup bias. *Applied Cognitive Psychology*, 13, S9-S26.

Wells, G. L.,Leippe, M. R., & Ostrom, T. M. (1979). Guidelines for empirically assessing the fairness of a lineup. *Law and Human Behavior, 3*(4), 285-293.

Examples

```
#Target present data:
A <- rep(1:6, length.out = 100)
B <- rep(1:5, length.out = 70)
C <- rep(1:4, length.out = 20)
lineup_pres_list <- list(A, B, C)
rm(A, B, C)


#Target absent data:
A <- rep(6:1, length.out = 100)
B <- rep(5:1, length.out = 70)
C <- rep(4:1, length.out = 20)
lineup_abs_list <- list(A, B, C)
rm(A, B, C)

# Suspect position for each TP/TA pair
pos_list <- c(3, 2, 1)

#Nominal size:
k <- c(6, 5, 4)

#Call:
linedf <- diag_param(lineup_pres_list, lineup_abs_list, pos_list, k)
```

diag_ratio_T *Diagnosticity Ratio (Tredoux, 1998)*

Description

Computes Tredoux's small-cell corrected diagnosticity ratio for one target-present/target-absent lineup pair.

Usage

```
diag_ratio_T(lineup_pres, lineup_abs, pos_pres, pos_abs, k1, k2)
```

Arguments

lineup_pres	A numeric vector of lineup choices for a lineup in which the target was present
lineup_abs	A numeric vector of lineup choices for a lineup in which the target was absent
pos_pres	A scalar, representing target position in TP lineup. Must be declared by user
pos_abs	A scalar, representing target position in TA lineup. Must be declared by user
k1	Number of targets in TP lineup. Must be specified by user (scalar).
k2	Number of targets in TA lineup. Must be specified by user (scalar).

Details

The corrected estimator is:

$$DR_T = \frac{(n_{TP,s} + 0.5)/(N_{TP} + 0.5)}{(n_{TA,s} + 0.5)/(N_{TA} + 0.5)},$$

where $n_{TP,s}$ and $n_{TA,s}$ are suspect-selection counts in target-present and target-absent lineups, respectively, and N_{TP} and N_{TA} are the corresponding sample sizes.

References

Malpass, R. S. (1981). Effective size and defendant bias in eyewitness identification lineups. *Law and Human Behavior, 5*(4), 299-309.

Malpass, R. S., Tredoux, C., & McQuiston-Surrett, D. (2007). Lineup construction and lineup fairness. In R. Lindsay, D. F. Ross, J. D. Read, & M. P. Toglia (Eds.), *Handbook of Eyewitness Psychology, Vol. 2: Memory for people* (pp. 155-178). Mahwah, NJ: Lawrence Erlbaum Associates.

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22*(2), 217-237.

Tredoux, C. (1999). Statistical considerations when determining measures of lineup size and lineup bias. *Applied Cognitive Psychology*, 13, S9-S26.

Wells, G. L.,Leippe, M. R., & Ostrom, T. M. (1979). Guidelines for empirically assessing the fairness of a lineup. *Law and Human Behavior, 3*(4), 285-293.

Examples

```
#Data:
lineup_pres <- round(runif(100, 1, 6))
lineup_abs <- round(runif(70, 1, 5))
pos_pres <- 3
pos_abs <- 5

#Call:
diag_ratio_T(lineup_pres, lineup_abs, pos_pres, pos_abs, 6, 5)
diag_ratio_T(lineup_pres, lineup_abs, 3, 5, 6, 5)
```

---

diag_ratio_T_bayes *Bayesian Diagnosticity Ratio (Tredoux, Beta-Binomial Model)*

Description

Computes a Bayesian posterior distribution for the Tredoux diagnosticity ratio using independent Beta-Binomial models for the target-present and target-absent suspect-selection rates.

Usage

```
diag_ratio_T_bayes(
  lineup_pres,
  lineup_abs,
  pos_pres,
  pos_abs,
  k1,
  k2,
  alpha = 0.5,
  S = 10000,
  credible_mass = 0.95,
  threshold = NULL
)
```

Arguments

lineup_pres	A numeric vector of lineup choices for a target-present lineup.
lineup_abs	A numeric vector of lineup choices for a target-absent lineup.
pos_pres	Scalar; suspect position in the target-present lineup.
pos_abs	Scalar; suspect position in the target-absent lineup.
k1	Number of members in the target-present lineup (for input validation).
k2	Number of members in the target-absent lineup (for input validation).
alpha	Dirichlet concentration (prior strength). A scalar $\alpha > 0$ applied to both cells. Named shortcuts: "jeffreys" (0.5, default), "uniform" (1), "weak" (0.1).

S	Number of posterior draws. Default 10000.
credible_mass	Width of the equal-tailed credible interval. Default 0.95.
threshold	Optional numeric. If supplied, reports $P(DR < t)$ and $P(DR > t\|\mathbf{n})$.

Details

The target-present suspect-selection rate p_{TP} and the target-absent rate p_{TA} are modelled independently with Beta posteriors:

$$p_{TP} \mid \mathbf{n} \sim \text{Beta}(n_{TP,s} + \alpha,\ N_{TP} - n_{TP,s} + \alpha)$$

$$p_{TA} \mid \mathbf{n} \sim \text{Beta}(n_{TA,s} + \alpha,\ N_{TA} - n_{TA,s} + \alpha)$$

The posterior of the diagnosticity ratio is obtained by drawing $DR^{(s)} = p_{TP}^{(s)}/p_{TA}^{(s)}$ for each sample s. The default Jeffreys prior ($\alpha = 0.5$) is recommended for small samples. Unlike the continuity-corrected point estimate, the posterior handles zero cell counts naturally and provides direct probability statements such as $P(DR > 1 \mid \mathbf{n})$.

The posterior mean is computed exactly, not from the draws. Because the two rates are independent, $E[DR] = E[p_{TP}]\,E[1/p_{TA}]$, with $E[1/p_{TA}] = (a + b - 1)/(a - 1)$ for $p_{TA} \sim \text{Beta}(a, b)$ when $a > 1$. When $a \leq 1$ (for example, no target-absent suspect choices under the Jeffreys prior) the mean is infinite, and an average of the draws would not settle on any value as S grew. Report the median and interval instead. In `diag_ratio_T_bayes_compare()`, the mean difference is `NA` when both means are infinite.

Value

An object of class `"diag_ratio_T_bayes"` containing:

DR_draws Numeric vector of length S: posterior draws of DR.

lnDR_draws Log-transformed posterior draws.

posterior_mean, posterior_median Posterior point estimates. The mean is exact, and is `Inf` when it does not exist (see Details).

credible_interval Named two-element vector (lower, upper).

prior_alpha Alpha value used.

n_tp, n_ta, S, credible_mass Input metadata.

threshold, threshold_probs Threshold and associated probabilities (or NULL).

References

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22*(2), 217-237.

Wells, G. L., & Turtle, J. W. (1986). Eyewitness identification: The importance of lineup models. *Psychological Bulletin, 99*(3), 320-329.

See Also

diag_ratio_T, diag_ratio_T_bayes_compare, esize_T_bayes

Examples

```
set.seed(1)
lineup_pres <- round(runif(100, 1, 6))
lineup_abs  <- round(runif(80, 1, 6))
res <- diag_ratio_T_bayes(lineup_pres, lineup_abs, pos_pres = 3, pos_abs = 3,
                          k1 = 6, k2 = 6)
print(res)
plot(res)
```

---

diag_ratio_T_bayes_compare

Bayesian Comparison of Two Diagnosticity Ratios

Description

Fits independent Beta-Binomial models to two lineup pairs and returns the posterior distribution of the difference $\Delta = DR_A - DR_B$, together with $P(DR_A > DR_B \mid \mathbf{n})$.

Usage

```
diag_ratio_T_bayes_compare(
  pres_A,
  abs_A,
  pos_pres_A,
  pos_abs_A,
  k1_A,
  k2_A,
  pres_B,
  abs_B,
  pos_pres_B,
  pos_abs_B,
  k1_B,
  k2_B,
  alpha = 0.5,
  S = 10000,
  credible_mass = 0.95
)
```

Arguments

pres_A, abs_A	Numeric vectors of lineup choices for lineup pair A (target-present and target-absent, respectively).
pos_pres_A, pos_abs_A	Suspect positions for pair A.
k1_A, k2_A	Lineup sizes for pair A (for validation).

pres_B, abs_B	Numeric vectors of lineup choices for lineup pair B.
pos_pres_B, pos_abs_B	Suspect positions for pair B.
k1_B, k2_B	Lineup sizes for pair B.
alpha	Prior concentration (shared). Default 0.5 (Jeffreys).
S	Number of posterior draws. Default 10000.
credible_mass	Width of credible interval. Default 0.95.

Value

An object of class "diag_ratio_T_bayes_compare".

See Also

diag_ratio_T_bayes

Examples

```
set.seed(1)
pres_A <- round(runif(100, 1, 6)); abs_A <- round(runif(80, 1, 6))
pres_B <- round(runif(100, 1, 6)); abs_B <- round(runif(80, 1, 6))
res <- diag_ratio_T_bayes_compare(pres_A, abs_A, 3, 3, 6, 6,
                                  pres_B, abs_B, 3, 3, 6, 6)
print(res)
```

---

diag_ratio_W *Diagnosticity Ratio (Wells & Lindsay, 1980; Wells & Turtle, 1986)*

Description

Computes Wells' diagnosticity ratio for one lineup pair

Usage

```
diag_ratio_W(lineup_pres, lineup_abs, pos_pres, pos_abs, k1, k2)
```

Arguments

lineup_pres	A numeric vector of lineup choices for a lineup in which the target was present
lineup_abs	A numeric vector of lineup choices for a lineup in which the target was absent
pos_pres	A scalar, representing target position in TP lineup. Must be declared by user
pos_abs	A scalar, representing target position in TA lineup. Must be declared by user
k1	Number of targets in TP lineup. Must be specified by user (scalar).
k2	Number of targets in TA lineup. Must be specified by user (scalar).

References

Wells, G. L., & Lindsay, R. C. L. (1980).On estimating the diagnosticity of eyewitness nonidentifications.*Psychological Bulletin, 88*, 776-784.

Wells, G. L., & Turtle, J. W. (1986). Eyewitness identification: The importance of lineup models. *Psychological Bulletin, 99*, 320-329.

Examples

```
#Data:
lineup_pres <- round(runif(100, 1, 6))
lineup_abs <- round(runif(70, 1, 5))
pos_pres <- 3
pos_abs <- 5

#Call:
diag_ratio_W(lineup_pres, lineup_abs, pos_pres, pos_abs, 6, 5)
diag_ratio_W(lineup_pres, lineup_abs, 3, 5, 6, 5)
```

---

display_lineup *Display Lineup as Image Grid*

Description

Creates a visual display of the lineup faces arranged in a grid.

Usage

```
display_lineup(
  target_path,
  foil_paths,
  ncol = 3,
  scale = 150,
  show_labels = TRUE,
  highlight_target = TRUE
)
```

Arguments

target_path	Path to target image.
foil_paths	Paths to foil images.
ncol	Number of columns in the grid.
scale	Scale factor for images.
show_labels	Logical. Add labels under each face.
highlight_target	Logical. Add border around target.

Value

A magick image object that can be displayed or saved.

Examples

```
## Not run:
# Display lineup
img <- display_lineup("target.jpg", paste0("foil", 1:5, ".jpg"))
print(img)

# Save to file
magick::image_write(img, "lineup_display.png")

## End(Not run)
```

---

dmax_filler *mSDT: Max Filler Distribution (IID Normal Fillers)*

Description

Distribution functions for the maximum of (m-1) independent standard normal filler signals, where m is the lineup size (suspect + fillers).

Usage

```
dmax_filler(x, lineup_size, log = FALSE)

pmax_filler(q, lineup_size, lower.tail = TRUE, log.p = FALSE)

qmax_filler(p, lineup_size, lower.tail = TRUE, log.p = FALSE)

rmax_filler(n, lineup_size)
```

Arguments

x, q	Numeric vector of quantiles.
lineup_size	Lineup size (suspect + fillers), must be >= 2.
log, log.p	Logical. Return log-density / log-probability.
lower.tail	Logical. If TRUE (default), returns P[X <= q].
p	Numeric vector of probabilities.
n	Number of random draws.

Value

Density, distribution, quantile, or random draws.

Examples

```
dmax_filler(0, lineup_size = 6)
pmax_filler(0, lineup_size = 6)
qmax_filler(0.5, lineup_size = 6)
rmax_filler(5, lineup_size = 6)
```

`eff_size_per_foils` *Effective Size per Foils*

Description

A package-specific count of members whose choice-proportion intervals include chance guessing. This is distinct from Malpass' continuous effective size.

Usage

```
eff_size_per_foils(lineup_vec, target_pos, k, conf = 0.95, R = 1000)
```

Arguments

`lineup_vec`	A numeric vector of lineup choices
`target_pos`	A numeric vector indexing all lineup members
`k`	Number of members in lineup. Must be specified by user (scalar).
`conf`	Confidence level for the intervals, e.g. 0.95 for 95 percent intervals. Defaults to 0.95. May be specified by user (scalar).
`R`	Number of bootstrap replications. Defaults to 1000.

Details

Because the result is a count of members rather than a continuous quantity, it changes only in whole steps as `conf` varies, and it is flat across ranges of `conf` in which no member's interval crosses $1/k$. A value that does not move when `conf` is changed from, say, 0.95 to 0.99 is therefore expected behaviour and not a sign that the argument is being ignored. Widening the interval can only ever increase the count.

Intervals are computed by `lineup_boot_allprop`, and are not corrected for the fact that `k` of them are examined at once.

Value

A single integer: the number of lineup members whose bootstrapped confidence interval for the proportion choosing them includes the chance value $1/k$. These are the members that a mock witness chose at a rate not distinguishable from guessing.

References

Malpass, R S. (1981). Effective size and defendant bias in eyewitness identification lineups. *Law and Human Behavior, 5*, 299-309.

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22*(2), 217-237.

Examples

```
#Data:
lineup_vec <- round(runif(100, 1, 6))
target_pos <- c(1, 2, 3, 4, 5, 6)

#Call:
eff_size_per_foils(lineup_vec, target_pos, 6)
eff_size_per_foils(lineup_vec, target_pos, conf = 0.95, 6)
```

---

effsize_compare *Master Function: Comparing Effective Size*

Description

Function for comparing effective size of two independent lineups (Tredoux, 1998)

Usage

```
effsize_compare(linedf, R = 1000)
```

Arguments

linedf	A dataframe of lineup data. Contains 2 columns, each of which hold data for 2 independent lineups
R	Number of bootstrap replications.

Details

This function is a master function, calling other functions it needs, and reporting results in some detail Each lineup is resampled independently; row alignment has no sampling meaning. Confidence intervals are bootstrap approximations and can be unreliable for small or nearly uniform empirical choice distributions. Normal and BCa intervals can under-cover in finite concentrated samples; percentile intervals can be conservative near uniformity. None provides a distribution-free level-.05 test of equal population effective sizes.

Value

Effective size, significance level, and confidence intervals (95 normal theory, percentile & bias-corrected)

References

Davison, A.C. & Hinkley, D.V. (1997). *Bootstrap methods and their application*. Cambridge University Press.

Malpass, R. S. (1981). Effective size and defendant bias in eyewitness identification lineups. *Law and Human Behavior, 5*(4), 299-309.

Malpass, R. S., Tredoux, C., & McQuiston-Surrett, D. (2007). Lineup construction and lineup fairness. In R. Lindsay, D. F. Ross, J. D. Read, & M. P. Toglia (Eds.), *Handbook of Eyewitness Psychology, Vol. 2: Memory for people* (pp. 155-178). Mahwah, NJ: Lawrence Erlbaum Associates.

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22*(2), 217-237.

Tredoux, C. (1999). Statistical considerations when determining measures of lineup size and lineup bias. *Applied Cognitive Psychology, 13*, S9-S26.

Wells, G. L.,Leippe, M. R., & Ostrom, T. M. (1979). Guidelines for empirically assessing the fairness of a lineup. *Law and Human Behavior, 3*(4), 285-293.

See Also

boot: https://cran.r-project.org/web/packages/boot/boot.pdf

Examples

```
#Data:

lineup_vec1 <- round(runif(100, 1, 6))
lineup_vec2 <- round(runif(100, 1, 6))
linedf <- as.data.frame(cbind(lineup_vec1, lineup_vec2))

#Call:
x <- effsize_compare(linedf, R = 100)
```

---

embedding_distance *Compute Distance Between Two Embeddings*

Description

Calculates the distance or similarity between two face embedding vectors.

Usage

```
embedding_distance(
  emb1,
  emb2,
  metric = c("cosine", "euclidean", "euclidean_l2")
)
```

Arguments

emb1	Numeric vector. First embedding.
emb2	Numeric vector. Second embedding.
metric	Character. Distance metric to use: • "cosine" - Cosine distance (1 - cosine similarity). Recommended for ArcFace. • "euclidean" - Euclidean (L2) distance. • "euclidean_l2" - Euclidean distance on L2-normalized vectors.

Details

For ArcFace and other angular-margin trained models, cosine distance is recommended as it aligns with the training objective.

Cosine distance ranges from 0 (identical) to 2 (opposite). Typical thresholds for same-person verification:

- ArcFace + cosine: ~0.68
- Facenet512 + cosine: ~0.30

Value

Numeric distance value. Lower values indicate more similar faces.

Examples

```
## Not run:
emb1 <- get_embedding("face1.jpg")$embedding
emb2 <- get_embedding("face2.jpg")$embedding

# Cosine distance (recommended)
embedding_distance(emb1, emb2, metric = "cosine")

# Euclidean distance
embedding_distance(emb1, emb2, metric = "euclidean")

## End(Not run)
```

---

esize_boot_dist *Bootstrap Distribution of Effective Size*

Description

Computes a bootstrap distribution for effective size using either Tredoux's E' or Malpass's adjusted E.

Usage

```
esize_boot_dist(
  lineup_input,
  k,
  metric = c("tredoux", "malpass"),
  input_type = c("vector", "table"),
  R = 1000,
  unbiased = FALSE
)
```

Arguments

lineup_input	A numeric lineup vector or a lineup table.
k	Nominal lineup size (suspect + fillers).
metric	Either "tredoux" or "malpass".
input_type	Either "vector" or "table".
R	Number of bootstrap resamples.
unbiased	Logical. For `metric = "tredoux"`, bootstrap the small-sample corrected E' of `esize_T(unbiased = TRUE)` instead of the plug-in estimate. Default `FALSE`.

Details

Percentile or BCa intervals from the plug-in E' bootstrap are not valid tests of exact uniform fairness. The population maximum E'=k is a nonregular boundary. If the sample size is not a multiple of k, even the largest attainable empirical E' is below k, so those intervals have zero coverage at exact fairness. More bootstrap replicates cannot fix this.

Value

A numeric vector of bootstrap estimates, carrying an `"n"` attribute equal to the number of lineup choices resampled. With `unbiased = TRUE`, resamples in which every choice fell on a different member have an unbounded corrected estimate; they are retained as `Inf`, with a warning, and their number is recorded in an `"n_undefined"` attribute.

Examples

```
vec <- round(runif(200, 1, 6))
dist <- esize_boot_dist(vec, k = 6, metric = "tredoux", R = 500)
dist_corrected <- esize_boot_dist(vec, k = 6, R = 500, unbiased = TRUE)
```

`esize_m` *Effective Size*

Description

Function for computing Effective Size

Usage

```
esize_m(lineup_tab, k, both = FALSE)
```

Arguments

`lineup_tab`	A table of lineup choices.
`k`	Number of members in the lineup. Must be specified by the user (scalar).
`both`	Logical. Defaults to FALSE, returning Tredoux's adjusted effective size estimate only. If TRUE, both Malpass's (1981) original formulation and Malpass's adjusted version (Tredoux, 1998) are printed to the console.

Details

Reduces the size of a lineup from a (corrected) nominal starting value by the degree to which members are, in sum, chosen below the level of chance expectation.

Value

The Tredoux-adjusted Malpass effective-size estimate. With `both = TRUE`, the original and adjusted estimates are also printed.

References

Malpass, R. S. (1981). Effective size and defendant bias in eyewitness identification lineups. *Law and Human Behavior, 5*(4), 299-309.

Malpass, R. S., Tredoux, C., & McQuiston-Surrett, D. (2007). Lineup construction and lineup fairness. In R. Lindsay, D. F. Ross, J. D. Read, & M. P. Toglia (Eds.), *Handbook of Eyewitness Psychology, Vol. 2: Memory for people* (pp. 155-178). Mahwah, NJ: Lawrence Erlbaum Associates.

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22*(2), 217-237.

Tredoux, C. (1999). Statistical considerations when determining measures of lineup size and lineup bias. *Applied Cognitive Psychology*, 13, S9-S26.

Wells, G. L.,Leippe, M. R., & Ostrom, T. M. (1979). Guidelines for empirically assessing the fairness of a lineup. *Law and Human Behavior, 3*(4), 285-293.

See Also

`esize_T`, whose "Small-sample bias" section describes a downward bias affecting plug-in effective-size estimators at small numbers of choices, and an optional correction.

Examples

```
#Data:
lineup_vec <- round(runif(100, 1, 6))

#Call:
esize_m(lineup_vec, 6, both = TRUE)
esize_m(lineup_vec, 6)
```

`esize_m_boot` *Bootstrapped Effective Size*

Description

Base function for computing bootstrapped effective size

Usage

```
esize_m_boot(lineup_vec, d, k)
```

Arguments

`lineup_vec`	A vector of lineup choices
`d`	Indices for bootstrap resampling
`k`	A vector indexing number of members in each lineup pair. Must be specified by user (scalar).

Details

Function to call when bootstrap resampling using boot function (in package 'boot')

Value

The Tredoux-adjusted Malpass effective-size estimate for the resample.

References

Davison, A.C. & Hinkley, D.V. (1997). *Bootstrap methods and their application*. Cambridge University Press.

Malpass, R. S. (1981). Effective size and defendant bias in eyewitness identification lineups. *Law and Human Behavior, 5*(4), 299-309.

Malpass, R. S., Tredoux, C., & McQuiston-Surrett, D. (2007). Lineup construction and lineup fairness. In R. Lindsay, D. F. Ross, J. D. Read, & M. P. Toglia (Eds.), *Handbook of Eyewitness Psychology, Vol. 2: Memory for people* (pp. 155-178). Mahwah, NJ: Lawrence Erlbaum Associates.

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22*(2), 217-237.

Tredoux, C. (1999). Statistical considerations when determining measures of lineup size and lineup bias. *Applied Cognitive Psychology*, 13, S9-S26.

Wells, G. L.,Leippe, M. R., & Ostrom, T. M. (1979). Guidelines for empirically assessing the fairness of a lineup. *Law and Human Behavior, 3*(4), 285-293.

See Also

boot: https://cran.r-project.org/web/packages/boot/boot.pdf

Examples

```
#Data:
lineup_vec <- round(runif(100, 1, 6))

#Get boot object:
bootobject <- boot::boot(lineup_vec, esize_m_boot, k = 6, R=1000)
bootobject

#To get confidence intervals:
cis <- boot::boot.ci(bootobject, conf = 0.95, type = "all", k = 6)
```

---

esize_T *Effective Size (Tredoux, 1998)*

Description

Function for generating Tredoux's transformed index-of-diversity effective size.

Usage

```
esize_T(lineup_tab, unbiased = FALSE)
```

Arguments

lineup_tab	A table of lineup choices
unbiased	Logical. If FALSE (the default) the plug-in estimator is returned, as in Tredoux (1998). If TRUE, the small-sample bias correction described below is applied. The default is unchanged so that existing analyses and published values are unaffected.

Details

Tredoux (1998) proposed using Agresti and Agresti's (1978) index of diversity for lineup choices,

$$I = 1 - \frac{\sum_i o_i^2}{N^2},$$

where o_i is the observed count for lineup member i and N is the total number of choices. The index is then transformed back to an effective-size scale:

$$E' = \frac{1}{1 - I} = \frac{N^2}{\sum_i o_i^2} = \frac{1}{\sum_i p_i^2}.$$

This formula is algebraically equivalent to a reciprocal concentration form.

Small-sample bias

The expression above is a *plug-in* estimator: sample proportions are substituted for the unknown p_i. The plug-in concentration $\sum_i \hat{p}_i^2$ is biased upward. Its reciprocal E' is biased **downward** for fair and near-fair lineups, but the direction is not guaranteed: when one member draws most choices the bias can be upward (two members, $p = (.9, .1)$, $N = 10$: mean 1.230 against a true 1.220, by exact enumeration). For N choices spread evenly over k members the bias is approximately

$$E[\hat{E}'] \approx \frac{Nk}{N + k - 1}, \qquad \text{bias} = -\frac{k(k-1)}{N + k - 1}.$$

A perfectly fair six-person lineup assessed with 50 choices therefore returns about 5.47 rather than 6, and with 20 choices about 4.87. The bias is depends on both nominal size and the choice distribution; no single sample size makes it negligible for every lineup.

With `unbiased = TRUE` the function substitutes the unbiased estimator of the concentration, $\sum_i o_i(o_i - 1)/(N(N-1))$, and returns its reciprocal,

$$\hat{E}'_{\text{corr}} = \frac{N(N-1)}{\sum_i o_i(o_i - 1)}.$$

The reciprocal of an unbiased estimate is not itself unbiased. For near-fair lineups this can reduce bias, with increased variability; in concentrated distributions it can overshoot (mean 1.269 in the two-member example). When fewer than two choices are available the correction is undefined and returns `NA` with a warning. When every choice is on a different member, the corrected reciprocal is `Inf`, retained with a warning. If $N \leq k$, that event can have positive probability, so the corrected estimator can have an infinite sampling mean. It is not uniformly superior.

Which to use depends on what is being reported. For a description of the choice distribution actually observed, the plug-in value is the quantity of interest. For inference about the lineup that produced those choices, the framing of Tredoux (1998), the corrected form usually has less bias when choices are spread across members. Neither is uniformly better. Note that the other functions in this family (`esize_T_boot`, `esize_T_ci_n`, `i_esize_T`) use the plug-in form.

References

Malpass, R. S. (1981). Effective size and defendant bias in eyewitness identification lineups. *Law and Human Behavior, 5*(4), 299-309.

Agresti, A., & Agresti, B. F. (1978). Statistical analysis of qualitative variation. *Sociological Methodology, 9*, 204-237.

Malpass, R. S., Tredoux, C., & McQuiston-Surrett, D. (2007). Lineup construction and lineup fairness. In R. Lindsay, D. F. Ross, J. D. Read, & M. P. Toglia (Eds.), Handbook of Eyewitness Psychology, Vol. 2: Memory for people (pp. 155-178). Mahwah, NJ: Lawrence Erlbaum Associates.

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22(2)*, 217-237.

Tredoux, C. (1999). Statistical considerations when determining measures of lineup size and lineup bias. *Applied Cognitive Psychology, 13*, S9-S26.

Wells, G. L., Leippe, M. R., & Ostrom, T. M. (1979). Guidelines for empirically assessing the fairness of a lineup. *Law and Human Behavior, 3*(4), 285-293.

Examples

```
#Data:
lineup_vec <- round(runif(100, 1, 6))
lineup_tab <- table(lineup_vec)

#Call:
e <- esize_T(lineup_tab)
```

---

esize_T_bayes *Bayesian Posterior Distribution of Effective Size (Tredoux, 1998)*

Description

Fits a Dirichlet-Multinomial conjugate model to mock-witness choice counts and returns the posterior distribution of effective size E' (Tredoux, 1998).

Usage

```
esize_T_bayes(
  lineup_tab,
  alpha = 0.5,
  S = 10000,
  credible_mass = 0.95,
  threshold = NULL
)
```

Arguments

lineup_tab	A table of lineup choice counts (one entry per position).
alpha	Dirichlet concentration parameter(s). A scalar (applied to all k positions) or a length-k numeric vector. Typical choices: 0.5 (Jeffreys prior, default), 1 (uniform over the simplex), 0.1 (weakly informative). Smaller values allow the posterior to concentrate more sharply around the observed proportions.
S	Number of posterior draws. Default 10000.
credible_mass	Width of the equal-tailed credible interval. Default 0.95.

`threshold` Optional finite scalar. When supplied, the function also reports P(E' < threshold | data) and P(E' > threshold | data), enabling direct probability statements about whether the lineup is functioning below or above a fairness benchmark. Because E' <= k for every choice-probability vector, with equality only when all k probabilities are exactly equal, P(E' < k | data) is 1 under any continuous posterior, however uniform the data. A threshold of k or more therefore carries no information, and the function warns. Use a tolerance below k that you choose and justify in advance (for example 0.8 k), or report the credible interval for E' or for the deficit k - E'.

Details

Effective size E' = 1/sum(p_i^2) (Tredoux, 1998) is mathematically identical to the reciprocal of the Herfindahl-Hirschman concentration index (Herfindahl, 1950; Hirschman, 1945) and to the reciprocal of Simpson's (1949) concentration statistic lambda. The Bayesian approach estimates the latent choice-probability vector p rather than E' directly.

With mock-witness counts n = (n_1, ..., n_k) and a Dirichlet prior:

$$p \sim \mathrm{Dirichlet}(\alpha_1, \ldots, \alpha_k),$$

the posterior is conjugate:

$$p \mid n \sim \mathrm{Dirichlet}(n_1 + \alpha_1, \ldots, n_k + \alpha_k).$$

Each posterior draw of p is mapped to a draw of E' via E'(p) = 1/sum(p_i^2). The resulting posterior distribution of E' summarises uncertainty about effective lineup size given the observed choices.

Dirichlet samples are generated by normalising independent Gamma draws (base R only; no additional package dependencies).

For publication, a prior sensitivity analysis using at least alpha = 0.5, 1, and 0.1 is recommended. If the substantive conclusion changes across these priors, the mock-witness sample is likely too small to support a confident fairness conclusion.

Value

An object of class `"esize_T_bayes"` containing:

E_draws Numeric vector of S posterior draws of E'.

posterior_mean Posterior mean of E'.

posterior_median Posterior median of E'.

credible_interval Named 2-vector (lower, upper) giving the equal-tailed credible interval.

prior_alpha The alpha vector used.

n Total number of mock-witness choices.

k Number of lineup positions.

S Number of posterior draws.

credible_mass Credible interval width.

threshold Threshold value supplied, or NULL.

threshold_probs List with `P_below` and `P_above`, or NULL when no threshold was supplied.

References

Herfindahl, O. C. (1950). *Concentration in the US steel industry*. Doctoral dissertation, Columbia University.

Hirschman, A. O. (1945). *National power and the structure of foreign trade*. University of California Press.

Simpson, E. H. (1949). Measurement of diversity. *Nature, 163*, 688.

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22*(2), 217-237.

See Also

esize_T, esize_T_ci_n, esize_boot_dist, esize_T_bayes_compare

Examples

```
counts <- c(18, 7, 5, 4, 3, 3)
lineup_tab <- as.table(setNames(counts, 1:6))

# Default Jeffreys prior
result <- esize_T_bayes(lineup_tab)
print(result)

# With a fairness threshold
result2 <- esize_T_bayes(lineup_tab, threshold = 4)
print(result2)

# Plot the posterior
plot(result)
```

---

esize_T_bayes_compare *Bayesian Comparison of Effective Size (Tredoux, 1998) Across Two Lineups*

Description

Fits independent Dirichlet-Multinomial models to two lineup tables and returns the posterior distribution of the difference in Tredoux E'.

Usage

```
esize_T_bayes_compare(
  table_a,
  table_b,
  alpha = 0.5,
  S = 10000,
  credible_mass = 0.95
)
```

Arguments

table_a	A table of choice counts for lineup A.
table_b	A table of choice counts for lineup B.
alpha	Dirichlet concentration parameter(s). Applied to both lineups. A scalar or a length-k vector (both lineups must have the same k). Default 0.5 (Jeffreys).
S	Number of posterior draws per lineup. Default 10000.
credible_mass	Width of the equal-tailed credible interval for the difference. Default 0.95.

Details

Independent Dirichlet-Multinomial posteriors are fitted for each lineup. The posterior of Delta = E'_A - E'_B is obtained by subtracting corresponding draws. P(E'_A > E'_B | data) is reported directly.

When the two lineups have different nominal sizes (different k), alpha must be a scalar; it is expanded independently for each lineup.

Value

An object of class "esize_T_bayes_compare" containing:

delta Posterior draws of Delta = E'_A - E'_B.

posterior_mean Posterior mean of Delta.

posterior_median Posterior median of Delta.

credible_interval Named 2-vector (lower, upper) for Delta.

P_A_greater Posterior probability that E'_A > E'_B.

P_B_greater Posterior probability that E'_B > E'_A.

result_a The esize_T_bayes object for lineup A.

result_b The esize_T_bayes object for lineup B.

credible_mass Credible interval width.

S Number of posterior draws.

References

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22*(2), 217-237.

See Also

esize_T_bayes, compare_eff_sizes.boot

Examples

```
counts_a <- c(20, 5, 5, 5, 5, 5)
counts_b <- c(10, 8, 8, 8, 8, 8)
tab_a <- as.table(setNames(counts_a, 1:6))
tab_b <- as.table(setNames(counts_b, 1:6))

result <- esize_T_bayes_compare(tab_a, tab_b)
print(result)
plot(result)
```

`esize_T_boot` *Bootstrapped Effective Size (Tredoux, 1998)*

Description

Base function for generating bootstrapped Effective Size (Tredoux, 1998)

Usage

```
esize_T_boot(lineup_tab, d)
```

Arguments

`lineup_tab`	A vector of individual lineup choices. The historical argument name is retained for compatibility.
`d`	Indices for bootstrap resampling

Details

This is the statistic function passed to `boot::boot()` for bootstrapping Tredoux's E'. The data argument must contain individual lineup choices; `d` resamples mock witnesses rather than count-table cells. For a higher-level interface that accepts raw lineup vectors, see `esize_boot_dist`. Percentile and BCa intervals from this plug-in statistic cannot be used to test exact fairness E'=k; see the boundary warning in `esize_boot_dist`.

References

Davison, A.C. & Hinkley, D.V. (1997). *Bootstrap methods and their application*. Cambridge University Press.

Malpass, R. S. (1981). Effective size and defendant bias in eyewitness identification lineups. *Law and Human Behavior, 5*(4), 299-309.

Malpass, R. S., Tredoux, C., & McQuiston-Surrett, D. (2007). Lineup construction and lineup fairness. In R. Lindsay, D. F. Ross, J. D. Read, & M. P. Toglia (Eds.), Handbook of Eyewitness Psychology, Vol. 2: Memory for people (pp. 155-178). Mahwah, NJ: Lawrence Erlbaum Associates.

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22(2)*, 217-237.

Tredoux, C. (1999). Statistical considerations when determining measures of lineup size and lineup bias. *Applied Cognitive Psychology, 13*, S9-S26.

Wells, G. L., Leippe, M. R., & Ostrom, T. M. (1979). Guidelines for empirically assessing the fairness of a lineup. *Law and Human Behavior, 3*(4), 285-293.

See Also

boot: https://cran.r-project.org/web/packages/boot/boot.pdf

Examples

```
# Choices from 50 mock witnesses to a 6-member lineup
set.seed(1)
lineup_tab <- sample(1:6, 50, replace = TRUE)
# E' for the observed data
esize_T_boot(lineup_tab, seq_along(lineup_tab))
```

---

esize_T_ci_n *Effective Size with Confidence Intervals from Normal Theory (Tredoux, 1998)*

Description

Computes effective size E' (Tredoux, 1998) together with a confidence interval derived from the normal-theory variance formula in Tredoux (1998).

Usage

```
esize_T_ci_n(lineup_tab, alpha)
```

Arguments

lineup_tab	A table of lineup choices.
alpha	Confidence level as a decimal (e.g., 0.95 for a 95-percent CI).

Details

The variance of the diversity index I is estimated via Tredoux's (1998) closed-form expression:

$$\mathrm{Var}(I) \approx \frac{4}{N}\left(\sum_i p_i^3 - \left(\sum_i p_i^2\right)^2\right).$$

Normal quantiles are applied to I and the resulting bounds are back-transformed to the E' scale via $E' = 1/(1 - I)$. For small samples or skewed distributions, all interval methods require qualification. In particular, plug-in bootstrap intervals also fail at exact uniform fairness; see esize_boot_dist. Bayesian credible intervals describe a stated posterior rather than guaranteeing fixed-parameter frequentist coverage.

Note that this concerns the *interval*. The point estimate itself is also biased downward near uniform choice distributions at small samples, although the direction is not universal, since E' is a plug-in estimator; see the "Small-sample bias" section of esize_T. This function uses the plug-in form. This first-order interval is unreliable in small samples and at uniform or concentrated boundary distributions. A zero estimated variance yields a degenerate interval, not evidence of zero population uncertainty. An ordinary bootstrap is also not a general solution to these boundary problems.

References

Malpass, R. S. (1981). Effective size and defendant bias in eyewitness identification lineups. *Law and Human Behavior, 5*(4), 299-309.

Malpass, R. S., Tredoux, C., & McQuiston-Surrett, D. (2007). Lineup construction and lineup fairness. In R. Lindsay, D. F. Ross, J. D. Read, & M. P. Toglia (Eds.), *Handbook of Eyewitness Psychology, Vol. 2: Memory for people* (pp. 155-178). Mahwah, NJ: Lawrence Erlbaum Associates.

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22*(2), 217-237.

Tredoux, C. (1999). Statistical considerations when determining measures of lineup size and lineup bias. *Applied Cognitive Psychology*, 13, S9-S26.

Wells, G. L.,Leippe, M. R., & Ostrom, T. M. (1979). Guidelines for empirically assessing the fairness of a lineup. *Law and Human Behavior, 3*(4), 285-293.

Examples

```
#Data:
lineup_vec <- round(runif(100, 1, 6))
lineup_tab <- table(lineup_vec)

#Call:
e_ci <- esize_T_ci_n(lineup_tab, .95)
```

---

estimate_msdt_params *mSDT Parameter Estimation from Rejection Rates*

Description

Estimate decision criterion (gamma) and discriminability (d') from rejection rates in culprit-absent and culprit-present lineups.

Usage

```
estimate_msdt_params(pr_rej_I, pr_rej_G, lineup_size, eps = 1e-06)

msdt_gamma_from_rej(pr_rej_I, lineup_size, eps = 1e-06)

msdt_dprime_from_rej(pr_rej_G, gamma, lineup_size, eps = 1e-06)
```

Arguments

pr_rej_I	Rejection rate in culprit-absent lineups.
pr_rej_G	Rejection rate in culprit-present lineups.
lineup_size	Lineup size (suspect + fillers), must be >= 2.
eps	Small value used to avoid probabilities of 0 or 1.
gamma	Decision criterion estimated from culprit-absent rejection rates, typically returned by msdt_gamma_from_rej().

Details

This is a **method-of-moments estimator** that is exactly identified from the two rejection rates alone: Pr(REJ|I) fixes gamma, and the gap between culprit-absent and culprit-present rejection rates fixes d'. It is **information-lossy**: it ignores the composition of choices (suspect-ID vs filler-ID) and the spread of filler choices, which are the most diagnostic outcomes. If you have the full response breakdown (suspect/filler/reject counts for both lineup types), prefer the minimum-Pearson fitter fit_max_sdt or the 2-HT model (fit_winter_2ht), which use all outcome counts.

Value

A list with gamma and dprime.

See Also

fit_max_sdt for a full-outcome, minimum-Pearson fit; fit_winter_2ht for the 2-HT MPT model.

Examples

```
estimate_msdt_params(pr_rej_I = 0.25, pr_rej_G = 0.15, lineup_size = 6)
```

---

extract_sdt_metrics *Extract SDT Metrics from a GLM/GLMM*

Description

Extracts d' (probit) or ln(OR) (logit) from SDT GLM/GLMM fits. Criterion is reported only when is_old was centered to -0.5/+0.5. With covariates these quantities are evaluated at zero/reference covariates; interactions can change them at other covariate values. GLMM coefficients are conditional latent-scale effects, not marginal population SDT measures. A logit coefficient is log odds ratio rather than normal-theory d-prime.

Usage

```
extract_sdt_metrics(model, link = NULL)
```

Arguments

model	A glm or glmerMod object produced by fit_sdt_glm/fit_sdt_glmm.
link	Optional. If NULL, inferred from the model family.

Value

A list containing d' or ln(OR), criterion (if available), and metadata.

Examples

```
set.seed(123)
n <- 200
is_old <- rep(c(0, 1), each = n / 2)
said_old <- rbinom(n, 1, ifelse(is_old == 1, 0.7, 0.3))
df <- data.frame(is_old = is_old, said_old = said_old)
fit <- fit_sdt_glm(df, is_old = "is_old", said_old = "said_old")
metrics <- extract_sdt_metrics(fit)
metrics$estimate  # d' under the probit link
```

---

face_similarity *Compute Face Similarity Between Two Images*

Description

Computes the similarity between two face images using deep learning embeddings. This is the core function for comparing a target face to a foil face.

Usage

```
face_similarity(
  img1_path,
  img2_path,
  model = "ArcFace",
  metric = "cosine",
  detector = "retinaface",
  return_embeddings = FALSE
)
```

Arguments

img1_path	Path to the first (target) image.
img2_path	Path to the second (foil) image.
model	Character. Face recognition model. Default is "ArcFace".
metric	Character. Distance metric: "cosine" (recommended), "euclidean", or "euclidean_l2".
detector	Character. Face detection backend. Default is "retinaface".
return_embeddings	Logical. If TRUE, also returns the embedding vectors.

Details

This function wraps DeepFace's verify function but is designed for similarity measurement rather than identity verification. In lineup research, we typically want to measure how similar foils are to the target, not whether they are the same person.

The 'verified' field is the model's identity decision: whether the distance falls inside the model's threshold for "same person". In a lineup it is a prompt to inspect the two photographs, not a finding. Face recognition systems make false matches, and a match does not show that a foil is nonfunctioning for witnesses.

Value

A list containing:

- distance - The computed distance (lower = more similar)
- similarity - Similarity score in [-1,1] for cosine metric
- verified - Logical, TRUE if distance is below the model's threshold
- threshold - The verification threshold for this model/metric
- model - Model used
- metric - Metric used
- embeddings - (if return_embeddings=TRUE) List of both embeddings

Examples

```
## Not run:
# Compare target face to a foil
result <- face_similarity("target.jpg", "foil1.jpg")

# Check similarity score
result$similarity  # 0.85 would mean very similar

# Inspect the model's same-identity flag; this is not a fairness judgment.
result$verified  # TRUE would indicate problematic similarity

## End(Not run)
```

---

fit_lineup_models *Fit Competing Wixted Lineup-Memory Models*

Description

Fits confidence-binned Independent Observations, Ensemble/BEST-Rest, and Integration models through an explicitly installed pyWitness engine.

Usage

```
fit_lineup_models(
  data,
  models = c("independent", "ensemble", "integration"),
  lineup_size = 6L,
  confidence_bins = NULL,
  variance = c("equal", "unequal"),
  shared_variance = c("estimated", "zero"),
  starts = NULL,
  control = NULL,
  envname = .pywitness_default_env
)
```

Arguments

data	A data frame with target_present, identification, and numeric confidence columns.
models	Character vector containing "independent", "ensemble", "integration", and optionally "best_rest". "max" aliases "independent".
lineup_size	Integer lineup size.
confidence_bins	Optional strictly increasing numeric bin edges. When NULL, each observed confidence value is an ordered bin.
variance	Either "equal" or "unequal".
shared_variance	Either "estimated" or "zero". Shared variance cancels from Ensemble and BEST-Rest and is fixed at zero there.
starts	Optional named numeric starting vector applied to all models, or a named list of such vectors indexed by model.
control	Named list controlling maxiter, optimizer method, numerical integration_sigma, chi2_variance, objective, and verbose. objective = "corrected" (the default) applies the per-fit correction described in Details; objective = "pywitness" leaves the engine's own objective and optimizer untouched, which reproduces results from pyWitness itself, including published fits.
envname	Name or path of the isolated pyWitness virtual environment.

Details

This interface is restricted to fair simultaneous lineups. Explicit target-absent suspect IDs are combined with other target-absent choices because all innocent members share one lure distribution. The upstream processor requires at least one observation in each aggregate TP/TA response category; zero cells within particular confidence bins remain supported. If the input includes a condition or procedure column, it must contain only one non-missing value; otherwise the data must be subset before fitting.

By default the bridge corrects the pinned engine's reversed observed/expected rejection-cell variance denominators in memory for each fit, without changing its installed environment; control

= list(objective = "pywitness") turns the correction off. With the default chi2_variance = "expected", every cell therefore uses the Pearson denominator. With "observed", every cell uses the observed denominator (the Neyman discrepancy), requiring positive observed cells. Thus, loglik_at_estimate is a multinomial likelihood evaluated at the minimum-Pearson estimate, not a maximized likelihood. AIC and BIC are returned as NA; computing them from a non-likelihood optimum would be invalid.

BEST-Rest and Ensemble obey

$$DV_{ensemble} = (k-1)DV_{best-rest}/k.$$

They fit identically after rescaling criteria and must not be counted as independent model-selection evidence. Integration is included as a historically important comparator, not a recommended default.

Value

An R-native "lineup_model_comparison" object containing comparison statistics, parameters, observed and expected cells, diagnostics, specifications, and engine metadata. It contains no live Python objects.

References

Wixted, J. T., Vul, E., Mickes, L., & Wilson, B. M. (2018). Models of lineup memory. *Cognitive Psychology, 105*, 81–114. doi:10.1016/j.cogpsych.2018.06.001

Mickes, L., Seale-Carlisle, T. M., Chen, X., & Boogert, S. (2024). pyWitness 1.0: A Python eyewitness identification analysis toolkit. *Behavior Research Methods, 56*, 1533–1550.

Examples

```
## Not run:
install_pywitness()
fits <- fit_lineup_models(
  lineup_example,
  confidence_bins = c(0, 60, 80, 100)
)
print(fits)
plot(fits, type = "fit")

## End(Not run)
```

---

fit_max_sdt *Restricted Independent-Observations/MAX Model for Lineup Counts*

Description

Fits the equal-variance, independent-signal, single-criterion MAX model to aggregate simultaneous-lineup counts using minimum-Pearson chi-squared estimation. Estimates d' (sensitivity) and lambda (criterion) simultaneously, supporting one or two conditions and optional cross-condition constraints.

Usage

```
fit_max_sdt(
  n_hit,
  n_tp_choose,
  n_fa,
  N_tp,
  N_ta,
  n_hit_2 = NULL,
  n_tp_choose_2 = NULL,
  n_fa_2 = NULL,
  N_tp_2 = NULL,
  N_ta_2 = NULL,
  n = 6,
  constrain_d = FALSE,
  constrain_c = FALSE,
  start = NULL,
  nboot = 0,
  ci_level = 0.95,
  seed = NULL
)
```

Arguments

n_hit	Number of correct suspect identifications in target-present (TP) lineups (condition 1).
n_tp_choose	Total number of participants who chose any lineup member in TP lineups (condition 1). Must be >= n_hit.
n_fa	Number of participants who chose any lineup member in target-absent (TA) lineups (condition 1).
N_tp	Total number of participants in TP lineups (condition 1).
N_ta	Total number of participants in TA lineups (condition 1).
n_hit_2, n_tp_choose_2, n_fa_2, N_tp_2, N_ta_2	Corresponding counts for an optional second condition. All five must be supplied together or not at all.
n	Lineup size (number of members). Default 6.
constrain_d	Logical. If TRUE and two conditions are supplied, constrains d' to be equal across conditions. Default FALSE.
constrain_c	Logical. If TRUE and two conditions are supplied, constrains lambda to be equal across conditions. Default FALSE.
start	Named numeric vector of starting values for the optimizer. If NULL (default), reasonable starting values are chosen automatically. Names should be "d1", "lambda1" (and "d2", "lambda2" for two conditions).
nboot	Number of parametric bootstrap replicates for confidence intervals. Default 0 (no bootstrap). Set to e.g. 1000 to obtain bootstrap CIs.
ci_level	Confidence level for bootstrap CIs. Default 0.95.
seed	Optional integer seed for reproducible bootstrap.

Details

This is a restricted Independent Observations/MAX model for simultaneous, fair lineups. Fillers are IID $N(0,1)$, the culprit signal is $N(d',1)$, memory signals are independent, and a single criterion is applied to the largest signal. Target-absent `n_fa` therefore means a choice of any lineup member, not only a designated innocent suspect.

For a lineup of n members with culprit signal strength d' and response criterion λ:

$$P(\text{correct ID}) = \int_{\lambda}^{\infty} \phi(x - d')\Phi(x)^{n-1}\, dx$$

$$P(\text{choose anyone} \mid \text{TP}) = 1 - \Phi(\lambda - d')\Phi(\lambda)^{n-1}$$

$$P(\text{choose anyone} \mid \text{TA}) = 1 - \Phi(\lambda)^{n}$$

Parameters are estimated by minimizing the Pearson chi-squared discrepancy between the mutually exclusive response cells: target-present correct ID, filler ID and rejection; and target-absent choice and rejection. This is not maximum-likelihood estimation and is not the multi-criterion likelihood model of Wixted et al. (2018). Nested comparisons (equal d' or equal lambda across conditions) are supported via `compare_max_sdt`.

Estimates are conditional on the model assumptions. Correlated signals, unequal variances, confidence criteria, sequential presentation, unfair lineups, and the Ensemble and Integration decision variables are outside the scope of this function.

Parametric bootstrap CIs are obtained by simulating data from the fitted model and re-fitting. Replicates that fail to converge are dropped and counted in `boot_n_failed`.

Value

An object of class `"max_sdt_fit"` containing:

dprime_1, lambda_1 Estimated d' and criterion for condition 1.

dprime_2, lambda_2 Estimated d' and criterion for condition 2 (if supplied).

chisq Chi-squared goodness-of-fit statistic.

df Degrees of freedom for the GoF test.

p_value P-value for the GoF test.

observed Named vector of observed counts.

predicted Named vector of predicted counts.

convergence Convergence code from `optim()` (0 = success).

constrain_d, constrain_c, n_conditions, n Input metadata.

boot_ci Bootstrap confidence intervals (if nboot > 0).

boot_params Full bootstrap parameter matrix (if nboot > 0). Rows for replicates that failed to converge are `NA`, so summarise with `na.rm = TRUE`.

boot_n_used, boot_n_failed Number of bootstrap replicates that converged and that did not (if nboot > 0).

Provenance

The implementation follows R scripts written by Andrew M. Smith (Iowa State University) for the analyses in Reisberg and Pezdek (2024), which fit what he describes as "the MAX SDT model (sometimes called the compound-decision model)". The default example counts are the Profile-lineup condition from that work. The constrained fits (`constrain_d`, `constrain_c`) and the nested chi-square comparison in `compare_max_sdt` follow the procedure described there.

Relation to Smith's scripts

Estimates from this function will **not** match those scripts exactly, and the reason is the set of cells entering the discrepancy function.

Smith's scripts fit culprit identifications, *total* target-present choices and *total* target-absent choices, together with the rejection counts. Culprit identifications therefore enter twice — once on their own and again inside total target-present choices — so the cells do not partition the sample, and the filler count is never fitted directly.

This function uses five mutually exclusive cells per condition: correct identification, filler identification, target-present rejection, target-absent choice, target-absent rejection. A Pearson chi-square is defined for a partition, so this form keeps the statistic and its degrees of freedom interpretable, and it fits the observed counts better on that criterion.

On the Profile-lineup counts used in the examples below, the two give d' = 1.94 (this function) against d' = 1.78 (Smith's script) for the first condition. Neither is a coding error; they minimise different discrepancies.

References

Duncan, M. (2006). *A signal detection model of compound decision tasks*. DRDC Toronto TR 2006-256.

Reisberg, D., & Pezdek, K. (2024). Data analysed with the MAX SDT model in scripts supplied by A. M. Smith (personal communication, 4 April 2024).

Kaesler, M., Dunn, J. C., Ransom, K., & Semmler, C. (2020). Do sequential lineups impair underlying discriminability? *Cognitive Research: Principles and Implications, 5*, 35. doi:10.1186/s41235020002345

Wixted, J. T., Vul, E., Mickes, L., & Wilson, B. M. (2018). Models of lineup memory. *Cognitive Psychology, 105*, 81–114. doi:10.1016/j.cogpsych.2018.06.001

See Also

`compare_max_sdt`, `estimate_msdt_params`, `sdt_compare`

Examples

```
# Single condition
fit1 <- fit_max_sdt(
  n_hit = 69, n_tp_choose = 82, n_fa = 64,
  N_tp = 96, N_ta = 106, n = 6
)
print(fit1)
```

```
# Two conditions, free parameters (each two-condition fit takes a few seconds)
fit_free <- fit_max_sdt(
  n_hit = 69, n_tp_choose = 82, n_fa = 64,
  N_tp = 96, N_ta = 106,
  n_hit_2 = 68, n_tp_choose_2 = 78, n_fa_2 = 42,
  N_tp_2 = 96, N_ta_2 = 96, n = 6
)

# Two conditions, equal d' (constrained)
fit_eqd <- fit_max_sdt(
  n_hit = 69, n_tp_choose = 82, n_fa = 64,
  N_tp = 96, N_ta = 106,
  n_hit_2 = 68, n_tp_choose_2 = 78, n_fa_2 = 42,
  N_tp_2 = 96, N_ta_2 = 96, n = 6,
  constrain_d = TRUE
)

# Compare models
cmp <- compare_max_sdt(fit_free, fit_eqd)
print(cmp)
```

`fit_sdt_glm` *Fit an SDT GLM for Old/New Recognition*

Description

Fits a single-level GLM (binomial link) to estimate SDT metrics from old/new recognition responses. With centered item coding, the coefficient on item status corresponds to d' (probit) or ln(OR) (logit).

Usage

```
fit_sdt_glm(
  data,
  is_old,
  said_old,
  covariates = NULL,
  link = "probit",
  center_isold = TRUE,
  interactions = FALSE
)
```

Arguments

`data`	Data frame containing trial-level responses.

is_old	Column name indicating whether the item is old (1) or new (0).
said_old	Column name indicating whether the participant said "old" (1) or "new" (0).
covariates	Optional character vector of covariate column names.
link	Link function for the binomial model ("probit" or "logit").
center_isold	Logical. If TRUE, recodes is_old to -0.5 / +0.5.
interactions	Logical. If TRUE, include is_old interactions with covariates.

Value

A glm object with SDT attributes.

Examples

```
set.seed(123)
n <- 200
is_old <- rep(c(0, 1), each = n / 2)
# d' of about 1: higher hit rate for old items
said_old <- rbinom(n, 1, ifelse(is_old == 1, 0.7, 0.3))
df <- data.frame(is_old = is_old, said_old = said_old)
fit <- fit_sdt_glm(df, is_old = "is_old", said_old = "said_old")
extract_sdt_metrics(fit)
```

---

fit_sdt_glmm *Fit an SDT GLMM for Old/New Recognition*

Description

Fits a multilevel GLMM (binomial link) to estimate SDT metrics from old/new recognition responses, using random effects for participants and optionally items.

Usage

```
fit_sdt_glmm(
  data,
  is_old,
  said_old,
  subject_id,
  covariates = NULL,
  item_id = NULL,
  link = "probit",
  center_isold = TRUE,
  random_slope = TRUE,
  interactions = FALSE
)
```

Arguments

data	Data frame containing trial-level responses.
is_old	Column name indicating whether the item is old (1) or new (0).
said_old	Column name indicating whether the participant said "old" (1) or "new" (0).
subject_id	Column name identifying participants.
covariates	Optional character vector of covariate column names.
item_id	Optional column name for item-level random intercepts.
link	Link function for the binomial model ("probit" or "logit").
center_isold	Logical. If TRUE, recodes is_old to -0.5 / +0.5.
random_slope	Logical. If TRUE, include random slope for is_old by subject.
interactions	Logical. If TRUE, include is_old interactions with covariates.

Value

A glmerMod object with SDT attributes.

Examples

```
if (requireNamespace("lme4", quietly = TRUE)) {
  set.seed(123)
  n_subj <- 20
  n_trial <- 20
  df <- expand.grid(subject = seq_len(n_subj), trial = seq_len(n_trial))
  df$is_old <- rep(c(0, 1), length.out = nrow(df))
  df$said_old <- rbinom(nrow(df), 1, ifelse(df$is_old == 1, 0.7, 0.3))
  fit <- fit_sdt_glmm(df, is_old = "is_old", said_old = "said_old",
                      subject_id = "subject", random_slope = FALSE)
  extract_sdt_metrics(fit)
}
```

---

fit_sdt_roc *Fit Signal Detection Theory Model to ROC Data*

Description

Fits an equal-variance or unequal-variance SDT model to ROC data using z-ROC analysis. Extracts discriminability (d'), decision criteria (c), and optionally the target/lure variance ratio.

Usage

```
fit_sdt_roc(
  data,
  lineup_size = NULL,
  model = c("equal_variance", "unequal_variance"),
  bootstrap = TRUE,
  n_bootstrap = 1000,
  conf_level = 0.95,
  seed = NULL,
  innocent_suspect = "auto"
)
```

Arguments

`data`	A dataframe with columns: target_present, identification, confidence OR a roc-data object from make_rocdata()
`lineup_size`	Integer. Number of people in lineup (required if data is dataframe)
`model`	Character. "equal_variance" or "unequal_variance" (default = "equal_variance")
`bootstrap`	Logical. Compute bootstrap confidence intervals? (default = TRUE)
`n_bootstrap`	Integer. Number of bootstrap samples (default = 1000)
`conf_level`	Numeric. Confidence level for intervals (default = 0.95)
`seed`	Integer. Random seed for reproducibility (default = NULL)
`innocent_suspect`	How innocent-suspect identifications in target-absent lineups are obtained. `"designated"`: the lineups had a designated innocent suspect, so target-absent `"suspect"` rows are counted directly, and zero such rows means a false-ID rate of zero. `"estimated"`: there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by `lineup_size`. `"auto"` (the default) chooses `"designated"` when any target-absent row is a suspect ID or the data have a non-missing `suspect_pos` column, and `"estimated"` otherwise, with a message.

Details

z-ROC Analysis:

The z-ROC is created by plotting z-transformed hit rates against z-transformed false alarm rates for each confidence level.

Under equal-variance SDT: - z-ROC should be linear with slope = 1 - d' = z(HR) - z(FAR) for any criterion - Intercept = d' in the line z(HR) = z(FAR) + d'

Under unequal-variance SDT: - Slope = SD_lure / SD_target - Slope > 1 indicates greater variability in lure distribution

Parameter Estimation: - d' = average distance between distributions in standard deviation units - c = decision criterion (0 = unbiased, positive = conservative, negative = liberal) - Higher d' = better discriminability

Bootstrap Confidence Intervals: If bootstrap = TRUE, resamples data with replacement to estimate standard errors and confidence intervals for all parameters.

Value

An S3 object of class "sdt_roc_fit" containing:

- dprime: Overall discriminability (d')
- criteria: Decision criterion for each confidence level (c values)
- variance_ratio: Ratio of lure SD to target SD, the z-ROC slope (if unequal_variance)
- slope: Slope of z-ROC
- intercept: Intercept of z-ROC
- roc_data: Original ROC data with z-transforms
- model_type: Type of model fit
- bootstrap_ci: Confidence intervals (if bootstrap = TRUE)
- fit_diagnostics: R-squared, residuals, etc.

References

Macmillan, N. A., & Creelman, C. D. (2005). Detection theory: A user's guide (2nd ed.).

Mickes, L. (2015). Receiver operating characteristic analysis and confidence-accuracy characteristic analysis in investigations of system variables. Journal of Applied Research in Memory and Cognition, 4(2), 93-102.

Examples

```
# Simulate data
sim_data <- simulate_lineup_data(
  n_tp = 200, n_ta = 200,
  d_prime = 1.5,
  conf_levels = 5,
  seed = 123
)

# Fit equal-variance SDT model
sdt_fit <- fit_sdt_roc(sim_data, lineup_size = 6, bootstrap = FALSE)
print(sdt_fit)
plot(sdt_fit)

# Get d' estimate
sdt_fit$dprime

# Fit unequal-variance model
sdt_uv <- fit_sdt_roc(
  sim_data, lineup_size = 6, model = "unequal_variance", bootstrap = FALSE
)
sdt_uv$variance_ratio


# Use more replicates for publication analyses.
sdt_boot <- fit_sdt_roc(
  sim_data, lineup_size = 6, n_bootstrap = 200, seed = 123
```

```
)
```

`fit_winter_2ht` *Winter et al. (2022) Two-High Threshold (2-HT) MPT Model for Eyewitness Identification*

Description

Fits the 2-HT multinomial processing tree (MPT) model to lineup identification data. This model separates detection-based processes (dP, dA) from non-detection-based processes (biased selection b, guessing-based selection g) using the full 2x3 outcome structure of lineup procedures.

Usage

```
fit_winter_2ht(
  data,
  lineup_size = 6,
  target_present = NULL,
  identification = NULL,
  start_params = NULL,
  method = "L-BFGS-B",
  ...
)
```

Arguments

`data`	A data frame with lineup identification data, OR a named vector/list of counts with the following elements: - n_tp_suspect: Number of suspect IDs in target-present (TP) lineups - n_tp_filler: Number of filler IDs in TP lineups - n_tp_reject: Number of rejections in TP lineups - n_ta_suspect: Number of suspect IDs in target-absent (TA) lineups - n_ta_filler: Number of filler IDs in TA lineups - n_ta_reject: Number of rejections in TA lineups
`lineup_size`	Size of the lineup (L). Default is 6.
`target_present`	Column name indicating target presence (TRUE/FALSE) if data is a data frame.
`identification`	Column name for identification decision if data is a data frame. Should contain "suspect", "filler", or "reject".
`start_params`	Optional named vector of starting parameter values (dP, dA, b, g). If NULL, uses reasonable defaults.
`method`	Optimization method. Default is "L-BFGS-B" for bounded optimization.
`...`	Additional arguments passed to optim().

Details

The 2-HT model (Winter, Menne, Bell, & Buchner, 2022) uses four parameters:

- **dP**: Probability of detecting culprit presence (0 to 1)
- **dA**: Probability of detecting culprit absence (0 to 1)
- **b**: Probability of biased suspect selection (0 to 1)
- **g**: Probability of guessing-based selection (0 to 1)

Model equations for culprit-present lineups:

- P(suspect ID) = dP + (1-dP) * [b + (1-b) * g * (1/L)]
- P(filler ID) = (1-dP) * (1-b) * g * ((L-1)/L)
- P(reject) = (1-dP) * (1-b) * (1-g)

Model equations for culprit-absent lineups:

- P(suspect ID) = (1-dA) * [b + (1-b) * g * (1/L)]
- P(filler ID) = (1-dA) * (1-b) * g * ((L-1)/L)
- P(reject) = dA + (1-dA) * (1-b) * (1-g)

With one aggregate 2 by 3 outcome table, the four free parameters fit the four independent cell proportions, so the model is saturated and has no residual goodness-of-fit degrees of freedom. Inference about model fit or restrictions requires additional conditions, parameter constraints, or participant/item-level replication; the reported AIC/BIC describe this likelihood fit but do not by themselves validate the process interpretation.

A feasible inverse of the published tree supplies the saturated interior solution; otherwise bounded optimization uses multiple starts. Regular Hessian SEs are returned only for interior fits. Boundary fits retain their estimates but have unavailable regular SEs. A low rejection percentage alone is not used as a criterion for refusing a fit.

Value

An object of class "winter_2ht" containing:

`parameters`	Estimated parameters (dP, dA, b, g)
`se`	Standard errors of parameter estimates
`loglik`	Log-likelihood of the fitted model
`aic`	Akaike Information Criterion
`bic`	Bayesian Information Criterion
`fitted_probs`	Model-predicted probabilities for each outcome
`observed_counts`	Observed counts
`expected_counts`	Expected counts under the model
`lineup_size`	Lineup size used
`convergence`	Convergence code from optim
`n_total`	Total sample size

References

Winter, K., Menne, N. M., Bell, R., & Buchner, A. (2022). Experimental validation of a multinomial processing tree model for analyzing eyewitness identification decisions. Scientific Reports, 12, 15571. https://doi.org/10.1038/s41598-022-19513-w

Examples

```
# Example 1: Using count data directly
counts <- c(
  n_tp_suspect = 147, n_tp_filler = 94, n_tp_reject = 141,
  n_ta_suspect = 38, n_ta_filler = 138, n_ta_reject = 206
)
fit <- fit_winter_2ht(counts, lineup_size = 6)
print(fit)
summary(fit)

# Example 2: Using a data frame
data(lineup_example)
fit <- fit_winter_2ht(
  lineup_example,
  lineup_size = 6,
  target_present = "target_present",
  identification = "identification"
)
```

---

format_comparison_table

Create Comprehensive Model Comparison Table

Description

Creates a formatted table comparing model fits for publication or reporting.

Usage

```
format_comparison_table(comparison_obj, format = "console", digits = 3)
```

Arguments

comparison_obj	A model_comparison object from compare_models()
format	Character. Output format: "console" (default), "markdown", "latex"
digits	Integer. Number of decimal places (default = 3)

Details

Creates a publication-ready table with key statistics for each model:

- 2-HT: Parameters (dP, dA, b, g), AIC, BIC
- EIG: Information gain, efficiency
- Full ROC: AUC

Value

For `format = "console"`, the comparison dataframe. For `"markdown"` and `"latex"`, a `knitr_kable` object, which is a character vector with **one element per line**. Print it with `cat(x, sep = "\n")` inside a chunk using `results = "asis"`; plain `cat(x)` joins the lines with spaces and the table collapses into a single paragraph.

Examples

```
data(lineup_example)
comparison <- compare_models(lineup_example)
format_comparison_table(comparison)
```

`func_size` *Functional Size*

Description

This function computes functional size (N/D), where N = number of mock witnesses, & D = number of mock witnesses choosing the target

Usage

```
func_size(lineup_vec, target_pos)
```

Arguments

`lineup_vec`	A numeric vector of lineup choices
`target_pos`	A scalar, representing target position in lineup. Must be declared by user

Details

An estimation of the number of plausible lineup members in a given lineup.

Value

Functional size of lineup

References

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22*(2), 217-237.

Tredoux, C. (1999). Statistical considerations when determining measures of lineup size and lineup bias. *Applied Cognitive Psychology, 13*, S9-S26.

Wells, G. L.,Leippe, M. R., & Ostrom, T. M. (1979). Guidelines for empirically assessing the fairness of a lineup. *Law and Human Behavior, 3*(4), 285-293.

Examples

```
#Data:
lineup_vec <- round(runif(100, 1, 6))
target_pos <- 3

#Call:
x <- func_size(lineup_vec, target_pos)
x <- func_size(lineup_vec, 3)
```

`func_size.boot` *Bootstrapped Functional Size*

Description

This function is a base function for the bootstrapping that ensues to compute bootstrapped confidence intervals for the estimate of functional size

Usage

```
func_size.boot(lineup_vec, d = d, target_pos)
```

Arguments

`lineup_vec`	A numeric vector of lineup choices
`d`	Indices for bootstrap resampling
`target_pos`	A scalar, representing position of target in lineup. Must be declared by user

Details

Function is called bootstrap resampling using boot function (in package 'boot') This function is never called by the user - it is called in the functional size master function

References

Davison, A.C. & Hinkley, D.V. (1997). *Bootstrap methods and their application*. Cambridge University Press.

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22*(2), 217-237.

Tredoux, C. (1999). Statistical considerations when determining measures of lineup size and lineup bias. *Applied Cognitive Psychology*, 13, S9-S26.

Wells, G. L.,Leippe, M. R., & Ostrom, T. M. (1979). Guidelines for empirically assessing the fairness of a lineup. *Law and Human Behavior, 3*(4), 285-293.

See Also

boot: https://cran.r-project.org/web/packages/boot/boot.pdf

Examples

```
# Lineup choices of 50 mock witnesses; target in position 3
set.seed(1)
lineup_vec <- sample(1:6, 50, replace = TRUE)
# Functional size for the observed data
func_size.boot(lineup_vec, seq_along(lineup_vec), target_pos = 3)
# Bootstrap functional size
boot_fs <- boot::boot(lineup_vec, func_size.boot, R = 100, target_pos = 3)
boot::boot.ci(boot_fs, type = "perc")
```

---

func_size_bayes *Bayesian Functional Size (Beta-Binomial Model)*

Description

Computes a Bayesian posterior distribution for the functional size of a lineup using a Beta-Binomial conjugate model for the suspect-selection rate.

Usage

```
func_size_bayes(
  lineup_vec,
  target_pos,
  alpha = 0.5,
  S = 10000,
  credible_mass = 0.95,
  threshold = NULL
)
```

Arguments

`lineup_vec`	A numeric vector of lineup choices.
`target_pos`	A scalar; the position of the suspect in the lineup.
`alpha`	Dirichlet concentration (prior strength). A positive scalar. Named shortcuts: `"jeffreys"` (0.5, default), `"uniform"` (1), `"weak"` (0.1).
`S`	Number of posterior draws. Default 10000.
`credible_mass`	Width of the equal-tailed credible interval. Default 0.95.
`threshold`	Optional numeric. If supplied, reports $P(F < t)$ and $P(F > t \mid \mathbf{n})$.

Details

Functional size $F = N/n_s$ is the reciprocal of the suspect-selection rate $p = n_s/N$. The Beta-Binomial conjugate model gives:

$$p \mid \mathbf{n} \sim \text{Beta}(n_s + \alpha,\ N - n_s + \alpha)$$

Posterior draws of $p^{(s)}$ are transformed via $F^{(s)} = 1/p^{(s)}$. The default Jeffreys prior ($\alpha = 0.5$) is recommended. The posterior gives direct probability statements such as $P(F < k \mid \mathbf{n})$, where k is the nominal lineup size.

The posterior mean of a reciprocal is computed exactly, not from the draws: with $a = n_s + \alpha$ and $b = N - n_s + \alpha$, $E[F \mid \mathbf{n}] = (a + b - 1)/(a - 1)$ when $a > 1$. When $a \leq 1$ (for example, no suspect choices under the Jeffreys prior) the mean is infinite, and the Monte Carlo average of the draws would not settle on any value as `S` grew. Report the median and interval instead.

Value

An object of class `"func_size_bayes"` containing:

F_draws Numeric vector of length `S`: posterior draws of functional size.

posterior_mean, posterior_median Posterior point estimates. The mean is exact, and is `Inf` when it does not exist (see Details).

credible_interval Named two-element vector (lower, upper).

prior_alpha, n, n_suspect, S, credible_mass Input metadata.

threshold, threshold_probs Threshold and probabilities (or NULL).

References

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22*(2), 217-237.

See Also

`func_size`, `esize_T_bayes`

Examples

```
set.seed(1)
lineup_vec <- round(runif(100, 1, 6))
res <- func_size_bayes(lineup_vec, target_pos = 3)
print(res)
plot(res)

# Threshold: P(F < 6 | data)
res_t <- func_size_bayes(lineup_vec, target_pos = 3, threshold = 6)
print(res_t)
```

`func_size_report` *Functional Size with Bootstrapped Confidence Intervals*

Description

This function is a master function, calling other functions it needs, and reporting results in some detail

Usage

```
func_size_report(lineup_vec, target_pos, k, R = 1000)
```

Arguments

`lineup_vec`	A numeric vector of lineup choices
`target_pos`	A scalar, representing target position in lineup. Must be declared by user
`k`	Number of members in lineup. Must be specified by user (scalar).
`R`	Number of bootstrap samples. Defaults to 1000

Details

Functional size is the reciprocal of the proportion of mock witnesses choosing the target, p. The bootstrap resamples p, which stays finite when a resample contains no target choices, and the interval limits for p are inverted. Percentile and BCa intervals respect monotone transformations, so they are the intervals for functional size itself, computed from the whole bootstrap distribution; a lower limit of 0 for p gives an unbounded (`Inf`) upper limit. The normal interval is the normal-theory interval for p, inverted. The exact interval inverts the Clopper-Pearson interval for p (`binom.test`). Near zero or one, empirical bootstrap intervals can have severe undercoverage, including a degenerate interval when the suspect was never chosen. Prefer the exact interval in these cases; retaining infinite limits alone does not guarantee nominal bootstrap coverage.

Function depends on functions from package 'boot'

Value

Invisibly returns the functional-size estimate; the normal, percentile and BCa bootstrap intervals; the exact interval; the number of replications; the number of replications in which nobody chose the target (`n_infinite`); and the bootstrap object, whose statistic is the proportion choosing the target.

References

Davison, A.C. & Hinkley, D.V. (1997). *Bootstrap methods and their application*. Cambridge University Press.

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22*(2), 217-237.

Tredoux, C. (1999). Statistical considerations when determining measures of lineup size and lineup bias. *Applied Cognitive Psychology*, 13, S9-S26.

Wells, G. L.,Leippe, M. R., & Ostrom, T. M. (1979). Guidelines for empirically assessing the fairness of a lineup. *Law and Human Behavior, 3*(4), 285-293.

See Also

`boot`: https://cran.r-project.org/web/packages/boot/boot.pdf

Examples

```
#Data:
lineup_vec <- round(runif(100, 1, 6))
target_pos <- 3

#Call:
x <- func_size_report(lineup_vec, target_pos, 6)
x <- func_size_report(lineup_vec, 3, 6)
```

---

gen_boot_propci *Percentile of Bootstrapped Lineup Proportion*

Description

Function for computing arbitrary percentile of bootstrapped lineup proportion

Usage

```
gen_boot_propci(lineuprops, perc = 0.05)
```

Arguments

`lineuprops`	A vector of bootstrapped lineup proportions
`perc`	Percentile to be computed. Must be declared by user (scalar) Defaults to .05

Details

Can be used to calculate confidence intervals at desired level of alpha

Value

Arbitrary percentile of bootstrapped lineup proportion

Examples

```
#Data:
lineup_vec <- round(runif(100, 1, 6))
target_pos <- 3
k <- 6

#Bootstrap data:
lineup_boot_df <- gen_boot_samples(lineup_vec, 1000)

#Compute proportion for bootstrap samples:
lineuprops <- gen_lineup_prop(lineup_boot_df, target_pos, 6)

#Get boot CIs:
prop_bootci_lower <- gen_boot_propci(lineuprops, perc = .025)
prop_bootci_upper <- gen_boot_propci(lineuprops, perc = .975)
```

---

gen_boot_propmean_se *Descriptive statistics for bootstrapped lineup proportion*

Description

Function for computing the mean, median, and bootstrap standard error. The standard deviation of the bootstrap replicates is the estimated standard error; it is not divided by the square root of the number of replicates.

Usage

```
gen_boot_propmean_se(lineuprops)
```

Arguments

lineuprops	A dataframe of bootstrapped lineup proportions

Value

Mean, median, standard deviation, standard error & 95 CIs of lineup proportion across a bootstrapped dataframe

Examples

```
#Data:
lineup_vec <- round(runif(100, 1, 6))
target_pos <- 3

#Bootstrap data:
lineup_boot_df <- gen_boot_samples(lineup_vec, 1000)

#Compute proportion for bootstrap samples:
lineuprops <- gen_lineup_prop(lineup_boot_df, target_pos = 3, k = 6)

#Call:
gen_boot_propmean_se(lineuprops)

#OR:

lineuprops <- boot::boot(lineup_vec, lineup_prop_boot, target_pos = 3, R = 1000)
gen_boot_propmean_se(lineuprops$t)
```

---

gen_boot_samples *Bootstrap resampling*

Description

Function for generating bootstrapped samples from 1 vector of lineup data

Usage

```
gen_boot_samples(lineup_vec, bootno)
```

Arguments

lineup_vec	A numeric vectors of lineup choices
bootno	Number of bootstrap samples

Value

A dataframe of bootstrapped lineup data

Examples

```
#Data:
lineup_vec <- round(runif(100,1,6))
bootno <- 1000

#Call:
bootdf <- gen_boot_samples(lineup_vec, bootno)
```

gen_boot_samples_list *Bootstrapped resampling*

Description

Function for generating bootstrapped samples for of k vectors of lineup choices

Usage

```
gen_boot_samples_list(lineup_list, bootno)
```

Arguments

lineup_list	A list containing k vectors of lineup choices for k lineups, in which the target was either absent or present
bootno	Number of bootstrap samples

Value

A list of bootstrapped lineup data Length of list = number of bootstrap sample draws

Examples

```
#Data:
A <-  round(runif(100,1,6))
B <-  round(runif(70,1,5))
C <-  round(runif(20,1,4))
linelist <- list(A, B, C)
rm(A, B, C)

bootno <- 1000

#Call:
bootdata <- gen_boot_samples_list(linelist, bootno)
```

gen_esize_m *Effective Size (across a dataframe)*

Description

Function for computing Effective Size (Tredoux, 1998) on lineups contained as columns in a df, usually from a bootstrapped sample

Usage

```
gen_esize_m(lineup_boot_df, k)
```

Arguments

lineup_boot_df	A dataframe containing bootstrapped samples of lineup data
k	Number of members in lineup. Must be specified by user (scalar).

Details

This function computes effective size for k lineups simultaneously.

Value

A vector of effective size calculations for each lineup in bootstrapped df

References

Malpass, R. S. (1981). Effective size and defendant bias in eyewitness identification lineups. *Law and Human Behavior, 5*(4), 299-309.

Malpass, R. S., Tredoux, C., & McQuiston-Surrett, D. (2007). Lineup construction and lineup fairness. In R. Lindsay, D. F. Ross, J. D. Read, & M. P. Toglia (Eds.), *Handbook of Eyewitness Psychology, Vol. 2: Memory for people* (pp. 155-178). Mahwah, NJ: Lawrence Erlbaum Associates.

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22*(2), 217-237.

Tredoux, C. (1999). Statistical considerations when determining measures of lineup size and lineup bias. *Applied Cognitive Psychology*, 13, S9-S26.

Wells, G. L.,Leippe, M. R., & Ostrom, T. M. (1979). Guidelines for empirically assessing the fairness of a lineup. *Law and Human Behavior, 3*(4), 285-293.

Examples

```
#Data:
lineup_vec <- round(runif(100,1,6))
bootdf <- gen_boot_samples(lineup_vec, 1000)

#Call:
esize_vec <- gen_esize_m(bootdf, 6)
```

---

gen_esize_m_ci *Bootstrap Quantile for Effective Size*

Description

Computes one requested quantile from a vector of bootstrapped Malpass effective sizes. Call the function twice (for example, at 0.025 and 0.975) to obtain both endpoints of a percentile interval.

Usage

```
gen_esize_m_ci(lineupsizes, perc = 0.05)
```

Arguments

lineupsizes	A non-empty numeric vector of bootstrapped effective sizes.
perc	A single quantile probability between 0 and 1. Defaults to 0.05.

Value

A single named bootstrap quantile.

Examples

```
#Data:
lineup_vec <- rep(1:6, length.out = 100)
k <- 6

#Use gen_boot_samples to get bootstrapped data:
bootdata <- gen_boot_samples(lineup_vec, 1000)

#Compute effective size over df of bootstrapped data:
lineupsizes <- gen_esize_m(bootdata, 6)

#Call:
gen_esize_m_ci(lineupsizes)
gen_esize_m_ci(lineupsizes, perc = .025)
gen_esize_m_ci(lineupsizes, perc = .975)
```

---

gen_lineup_prop *Lineup proportion over dataframe*

Description

Function for computing lineup proportion over a dataframe

Usage

```
gen_lineup_prop(lineup_boot_df, target_pos, k)
```

Arguments

lineup_boot_df	Dataframe of lineup choices (usually a bootstrapped set)
target_pos	A scalar, representing target position in lineup. Must be declared by user.
k	A vector indexing number of members in each lineup pair (nominal size). Must be specified by user (scalar).

Value

A vector of bootstrapped proportions, indicating the frequency with which a target was identified in a lineup Length of vector = number of bootstrap sample draws

Examples

```
#Data
lineup_vec <- round(runif(100, 1, 6))
target_pos <- 3

#Get bootstrapped data:
lineup_boot_df <- gen_boot_samples(lineup_vec, 1000)

#Call:
lineuprops <- gen_lineup_prop(lineup_boot_df, 3, 6)
```

---

gen_linevec *Lineup vector*

Description

A function that takes a table of lineup choices and generates a vector of raw lineup data

Usage

```
gen_linevec(lineup_tab, k)
```

Arguments

lineup_tab	A table of lineup choices
k	Nominal size (i.e.,total number of lineup members). Must be declared by user

Value

Returns a vector of lineup choices

Examples

```
#Data:
lineup_vec <- round(runif(100, 1, 6))
lineup_tab <- table(lineup_vec)

#Call:
lineup_vec <- gen_linevec(lineup_tab, 6)
```

`get_embedding` *Extract Face Embedding from an Image*

Description

Extracts a deep learning embedding vector from a face image using a specified model. The embedding can be used to compute face similarity.

Usage

```
get_embedding(
  img_path,
  model = "ArcFace",
  detector = "retinaface",
  align = TRUE,
  enforce_detection = TRUE,
  expand_percentage = 0
)
```

Arguments

`img_path`	Path to the image file containing a face.
`model`	Character. The face recognition model to use. Default is "ArcFace" (recommended). See `available_models`.
`detector`	Character. Face detection backend. Default is "retinaface" (most accurate). See `available_detectors`.
`align`	Logical. Whether to align the face before embedding. Default is TRUE (recommended).
`enforce_detection`	Logical. If TRUE, raises an error if no face is detected. If FALSE, attempts to use the entire image. Default is TRUE.
`expand_percentage`	Numeric. Percentage to expand the detected face region. Default is 0.

Details

The embedding dimension depends on the model:

- ArcFace: 512 dimensions
- Facenet512: 512 dimensions
- Facenet: 128 dimensions
- VGG-Face: 4096 dimensions

One example configuration used in the package is:

- model = "ArcFace"

- detector = "retinaface"
- align = TRUE

These defaults do not establish superior accuracy for every population, image source or research question, or validate lineup fairness.

Value

A named list containing:

- embedding - Numeric vector of face embeddings
- facial_area - List with x, y, w, h of detected face region
- face_confidence - Detection confidence score

Examples

```
## Not run:
# Get embedding for a single face
emb <- get_embedding("path/to/face.jpg")

# Access the embedding vector
emb$embedding

# Use a different model
emb_facenet <- get_embedding("path/to/face.jpg", model = "Facenet512")

## End(Not run)
```

---

get_face_landmarks *Get 478-Point Face Mesh Landmarks*

Description

Extracts dense facial landmarks using MediaPipe Face Mesh, which provides 478 3D landmark points covering the entire face, including the irises.

Usage

```
get_face_landmarks(img_path, return_3d = TRUE)
```

Arguments

img_path	Path to the image file.
return_3d	Logical. If TRUE, returns 3D coordinates (x, y, z). If FALSE, returns only 2D coordinates (x, y). Default is TRUE.

Details

MediaPipe Face Mesh provides detailed landmarks covering:

- Face oval/contour
- Left and right eyebrows
- Left and right eyes, with 10 iris points (468 to 477)
- Nose
- Lips (inner and outer contours)
- Face interior regions (cheeks, forehead)

The x and y coordinates are normalized to [0, 1] relative to image dimensions. The z coordinate represents depth relative to the face center.

Older MediaPipe releases expose the legacy `mediapipe.solutions.face_mesh` interface, which is used when present. Current releases have only the Tasks API (`FaceLandmarker`), which needs a model file. It is downloaded once, on first use, to `tools::R_user_dir("r4lineups", "cache")`. On macOS the Tasks API runs on the GPU delegate, because the CPU delegate in current macOS wheels aborts the process.

Value

A list containing:

- landmarks - Matrix of landmark coordinates (478 x 2 or 478 x 3)
- face_detected - Logical indicating if a face was found
- image_width - Width of the input image
- image_height - Height of the input image
- n_landmarks - Number of landmarks (when a face was found)

Note

Requires MediaPipe to be installed (installed by default with install_r4lineups_python()).

Examples

```
## Not run:
# Get face mesh landmarks
mesh <- get_face_landmarks("face.jpg")

# Plot the landmarks
plot(mesh$landmarks[, 1], mesh$landmarks[, 2], pch = ".")

## End(Not run)
```

`get_key_landmarks` *Get Key Facial Landmarks (5-Point)*

Description

Extracts the 5 key facial landmarks commonly used for face alignment: left eye, right eye, nose tip, left mouth corner, right mouth corner.

Usage

```
get_key_landmarks(img_path, detector = "retinaface")
```

Arguments

`img_path`	Path to the image file.
`detector`	Detector backend. "retinaface" provides these landmarks directly.

Details

These 5 landmarks are sufficient for basic face alignment and are commonly used in face recognition pipelines. For more detailed landmarks (478 points), use `get_face_landmarks`.

Value

A list with named landmark coordinates:

- left_eye - (x, y) coordinates
- right_eye - (x, y) coordinates
- nose - (x, y) coordinates
- mouth_left - (x, y) coordinates
- mouth_right - (x, y) coordinates

Examples

```
## Not run:
# Get key landmarks
lm <- get_key_landmarks("face.jpg")

# Calculate inter-pupillary distance
ipd <- sqrt(sum((lm$left_eye - lm$right_eye)^2))

## End(Not run)
```

has_valid_face *Check if Image Contains a Valid Face*

Description

Quick check to determine if an image contains a detectable face that can be used for embedding extraction.

Usage

```
has_valid_face(img_path, detector = "retinaface", min_confidence = 0.5)
```

Arguments

img_path	Path to the image file.
detector	Detector to use.
min_confidence	Minimum detection confidence (0-1).

Value

Logical. TRUE if a valid face is detected.

Examples

```
## Not run:
if (has_valid_face("image.jpg")) {
  emb <- get_embedding("image.jpg")
}

## End(Not run)
```

homog_diag *Master function: Homogeneity of diagnosticity ratio*

Description

This function provides assesses the homogeneity of the diagnosticity ratio of k lineup pairs.

Usage

```
homog_diag(lineup_pres_list, lineup_abs_list, pos_list, k, conf.level = 0.95)
```

Arguments

`lineup_pres_list`	A list containing k vectors of lineup choices for k lineups, in which the target was present
`lineup_abs_list`	A list containing k vectors of lineup choices for k lineups, in which the target was absent
`pos_list`	Suspect positions for each lineup pair. See `diag_param()`.
`k`	A vector indexing number of members in each lineup pair (nominal size). Must be specified by user (scalar).
`conf.level`	Confidence level for the per-pair intervals, and the complement of the level at which heterogeneity is declared. Defaults to 0.95.

Details

Master function for assessing homogeneity of diagnosticity ratio for k independent lineups.

The chi-square is Cochran's Q, $\sum w_i(\ln d_i - \ln \bar{d})^2$, so it measures the *spread* of the k ratios around their pooled value. It answers a different question from a confidence interval on the pooled ratio, which asks whether the average ratio differs from 1. The two can disagree: ratios that differ in direction (say 6, 1 and 0.44) give a very large Q while averaging to roughly 1. When Q is significant the ratios do not share a common value, the pooled estimate should not be reported as a single summary, and the pairs should be interpreted separately. This function prints a note to that effect and records it in the `heterogeneous` element.

Value

Invisibly, a list with the pooled (precision-weighted) diagnosticity ratio, the chi-square statistic and its degrees of freedom and p-value, a `heterogeneous` flag, and a `parameters` data frame giving the cell counts plus d, $\ln d$, its variance, its weight and a confidence interval for each lineup pair. The same information is printed.

References

Malpass, R. S. (1981). Effective size and defendant bias in eyewitness identification lineups. *Law and Human Behavior, 5*(4), 299-309.

Malpass, R. S., Tredoux, C., & McQuiston-Surrett, D. (2007). Lineup construction and lineup fairness. In R. Lindsay, D. F. Ross, J. D. Read, & M. P. Toglia (Eds.), *Handbook of Eyewitness Psychology, Vol. 2: Memory for people* (pp. 155-178). Mahwah, NJ: Lawrence Erlbaum Associates.

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22*(2), 217-237.

Tredoux, C. (1999). Statistical considerations when determining measures of lineup size and lineup bias. *Applied Cognitive Psychology*, 13, S9-S26.

Wells, G. L.,Leippe, M. R., & Ostrom, T. M. (1979). Guidelines for empirically assessing the fairness of a lineup. *Law and Human Behavior, 3*(4), 285-293.

Examples

```
#Target present data:
A <- rep(1:6, length.out = 100)
B <- rep(1:5, length.out = 70)
C <- rep(1:4, length.out = 20)
lineup_pres_list <- list(A, B, C)
rm(A, B, C)

#Target absent data:
A <- rep(6:1, length.out = 100)
B <- rep(5:1, length.out = 70)
C <- rep(4:1, length.out = 20)
lineup_abs_list <- list(A, B, C)
rm(A, B, C)

# Suspect position for each TP/TA pair
pos_list <- c(3, 2, 1)

#Nominal size:
k <- c(6, 5, 4)

#Call:
homog_diag(lineup_pres_list, lineup_abs_list, pos_list, k)
```

---

homog_diag_boot *Homogeneity of diagnosticity ratio with bootstrapped CIs*

Description

Function for computing bootstrapped estimates of homogeneity of diagnosticity ratio

Usage

```
homog_diag_boot(
  lineup_pres_list,
  lineup_abs_list,
  k,
  R = 100,
  pos_list = NULL,
  seed = NULL,
  conf.level = 0.95
)
```

Arguments

`lineup_pres_list`
: A list containing k vectors of lineup choices for k lineups, in which the target was present

`lineup_abs_list`
: A list containing k vectors of lineup choices for k lineups, in which the target was absent

`k`
: Number of members in lineup. Must be specified by user (scalar).

`R`
: Number of bootstrap replications. Defaults to R = 100.

`pos_list`
: Suspect positions for each lineup pair, in the same format as `diag_param()`. This is required; earlier releases attempted to infer suspect positions from observed choices, which cannot be done validly.

`seed`
: Optional integer seed for reproducible resampling.

`conf.level`
: Confidence level for the percentile intervals and the per-pair intervals, and the complement of the level at which heterogeneity is declared. Defaults to 0.95.

Details

Computes bootstrapped diagnosticity ratio with chi-squared estimate, significance level and confidence intervals for k lineup pairs.

The interval on the pooled ratio and the chi-square answer different questions, and can disagree. See `homog_diag` for the distinction; as there, a note is printed whenever the k ratios are heterogeneous and the pooled estimate should not be reported on its own.

Value

Invisibly returns a list containing the observed mean diagnosticity, chi-square statistic with degrees of freedom and asymptotic p-value, percentile intervals, a `heterogeneous` flag, a `parameters` data frame of per-pair ratios and intervals, all attempted bootstrap draws (failed rows retained as NA), `n_successful`, `n_failed`, and `R`. Intervals are conditional on successful draws when failures occur.

References

Malpass, R. S. (1981). Effective size and defendant bias in eyewitness identification lineups. *Law and Human Behavior, 5*(4), 299-309.

Malpass, R. S., Tredoux, C., & McQuiston-Surrett, D. (2007). Lineup construction and lineup fairness. In R. Lindsay, D. F. Ross, J. D. Read, & M. P. Toglia (Eds.), *Handbook of Eyewitness Psychology, Vol. 2: Memory for people* (pp. 155-178). Mahwah, NJ: Lawrence Erlbaum Associates.

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22*(2), 217-237.

Tredoux, C. (1999). Statistical considerations when determining measures of lineup size and lineup bias. *Applied Cognitive Psychology*, 13, S9-S26.

Wells, G. L.,Leippe, M. R., & Ostrom, T. M. (1979). Guidelines for empirically assessing the fairness of a lineup. *Law and Human Behavior, 3*(4), 285-293.

Examples

```
#Target present data:
A <- rep(1:6, length.out = 100)
B <- rep(1:5, length.out = 70)
C <- rep(1:4, length.out = 20)
```

```
lineup_pres_list <- list(A, B, C)
rm(A, B, C)

#Target absent data:
A <- rep(6:1, length.out = 100)
B <- rep(5:1, length.out = 70)
C <- rep(4:1, length.out = 20)
lineup_abs_list <- list(A, B, C)
rm(A, B, C)

pos_list <- c(3, 2, 1)
k <- c(6, 5, 4)
homog_diag_boot(lineup_pres_list, lineup_abs_list, k, R = 20,
                pos_list = pos_list, seed = 1)
```

`i_esize_T` *I Component of Effective Size (Tredoux, 1998)*

Description

Computes the Agresti-Agresti index of diversity I, the intermediate quantity used to construct effective size E' = 1/(1 - I) (Tredoux, 1998).

Usage

```
i_esize_T(lineup_tab)
```

Arguments

`lineup_tab`	A table of lineup choices.

Details

Tredoux (1998) used Agresti and Agresti's (1978) index of diversity:

$$I = 1 - \frac{\sum_i o_i^2}{N^2},$$

where o_i is the observed count for lineup member i and N is the total number of choices. This equals $1 - \sum_i p_i^2$, where $p_i = o_i/N$. The effective size is the reciprocal transform $E' = 1/(1 - I)$, computed by `esize_T`.

Value

A scalar: the diversity index I.

References

Agresti, A., & Agresti, B. F. (1978). Statistical analysis of qualitative variation. *Sociological Methodology, 9*, 204-237.

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22*(2), 217-237.

See Also

esize_T

Examples

```
lineup_vec <- round(runif(100, 1, 6))
lineup_tab <- table(lineup_vec)
i <- i_esize_T(lineup_tab)
```

---

innocent_id_rate_effective

Compute Innocent Suspect ID Rate with Effective Size Correction

---

Description

Estimates innocent-suspect identification rate using effective lineup size, accounting for implausible fillers and lineup bias.

Usage

```
innocent_id_rate_effective(error_rate, effective_size)
```

Arguments

error_rate	Numeric. Overall mistaken identification rate from culprit-absent lineups.
effective_size	Numeric. Effective size computed from the distribution of mistaken IDs (typically via esize_T).

Details

This represents a more realistic estimate when lineup fairness is uncertain. Uses the distribution of mistaken IDs to estimate the number of plausible lineup members. The formula is: innocent_id_rate = error_rate / effective_size

When effective size < nominal size, this indicates lineup bias and increases the estimated innocent-suspect risk.

Value

Numeric. Estimated innocent-suspect identification rate.

References

Fitzgerald, R. J., Tredoux, C. G., & Juncu, S. (2023). Estimation of eyewitness error rates in fair and biased lineups. *Law and Human Behavior*.

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22*(2), 217-237.

Examples

```
# 30% mistaken ID rate with an effective size of 4.2
innocent_id_rate_effective(0.30, 4.2)
```

`innocent_id_rate_nominal` *Compute Innocent Suspect ID Rate with Nominal Size Correction*

Description

Estimates innocent-suspect identification rate using nominal lineup size, assuming all lineup members are equally plausible (perfectly fair lineup).

Usage

```
innocent_id_rate_nominal(error_rate, lineup_size)
```

Arguments

`error_rate`	Numeric. Overall mistaken identification rate from culprit-absent lineups.
`lineup_size`	Integer. Nominal number of people in the lineup.

Details

This represents the "best case" estimate assuming perfect lineup fairness. The formula is: innocent_id_rate = error_rate / lineup_size

Value

Numeric. Estimated innocent-suspect identification rate.

References

Fitzgerald, R. J., Tredoux, C. G., & Juncu, S. (2023). Estimation of eyewitness error rates in fair and biased lineups. *Law and Human Behavior*.

Examples

```
# 30% mistaken ID rate in a fair 6-person lineup
innocent_id_rate_nominal(0.30, 6)
```

`innocent_id_rate_uncorrected` *Compute Innocent Suspect ID Rate with No Correction*

Description

Estimates innocent-suspect identification rate assuming all mistaken IDs are innocent-suspect IDs (no correction for lineup size).

Usage

```
innocent_id_rate_uncorrected(error_rate)
```

Arguments

`error_rate`	Numeric. Overall mistaken identification rate from culprit-absent lineups.

Details

This represents the "worst case" estimate, treating every mistaken ID as an innocent-suspect identification. Provides an upper bound on innocent-suspect risk. The formula is simply: innocent_id_rate = error_rate

Value

Numeric. Estimated innocent-suspect identification rate (equal to error_rate).

References

Fitzgerald, R. J., Tredoux, C. G., & Juncu, S. (2023). Estimation of eyewitness error rates in fair and biased lineups. *Law and Human Behavior*.

Examples

```
# Worst case: all mistaken IDs treated as innocent-suspect IDs
innocent_id_rate_uncorrected(0.30)
```

`install_pywitness` *Install the Audited pyWitness Engine*

Description

Explicitly creates or reuses an isolated Python virtual environment and installs the audited pyWitness Git revision used by `fit_lineup_models`. This function is never called when the package is loaded or checked.

Usage

```
install_pywitness(
  envname = .pywitness_default_env,
  python = NULL,
  pip_options = NULL,
  force = FALSE
)
```

Arguments

`envname`	Name or path of the virtual environment.
`python`	Optional path to the Python executable used to create a new virtual environment.
`pip_options`	Optional character vector passed to pip.
`force`	Logical. Reinstall the pinned revision if already present.

Details

The installation is pinned to Git revision `e726dcfc09423d0e0ff7f46c8e3a711040293eba`. pyWitness is GPL-3 software maintained separately from r4lineups; no pyWitness source or documentation is bundled in this package. The immutable commit archive is used so installation does not require a local Git executable.

Value

Invisibly returns the environment name.

References

Mickes, L., Seale-Carlisle, T. M., Chen, X., & Boogert, S. (2024). pyWitness 1.0: A Python eyewitness identification analysis toolkit. *Behavior Research Methods, 56*, 1533–1550.

Examples

```
## Not run:
install_pywitness()
check_pywitness_deps(initialize = TRUE)

## End(Not run)
```

`install_r4lineups_python` *Install Python Dependencies for r4lineups*

Description

Installs the required Python packages using pip. This function should be run once after installing the r4lineups package.

Usage

```
install_r4lineups_python(
  method = "auto",
  envname = NULL,
  pip_options = NULL,
  gpu = FALSE,
  python_version = NULL
)
```

Arguments

`method`	Installation method. Default is "auto" which uses pip.
`envname`	Name of virtual environment to use (optional). If NULL, installs to the system Python or currently active environment.
`pip_options`	Additional options to pass to pip (e.g., "–user").
`gpu`	Logical. If TRUE, installs tensorflow with GPU support. Default is FALSE.
`python_version`	Python version to build the virtual environment with, e.g. `"3.11"`. Defaults to `NULL`, which leaves the choice to reticulate but still checks the result against the supported range.

Details

This function installs:

- deepface - Main face analysis library
- opencv-python - Image processing
- numpy - Numerical computing

- tf-keras - TensorFlow/Keras backend
- mediapipe - For 478-point face mesh (optional but recommended)
- retinaface - Best face detector

Python version. TensorFlow publishes wheels for a limited range of Python versions (currently 3.9 to 3.12). On a machine whose default interpreter is newer than that, pip finds no TensorFlow distribution at all and the install fails partway through with `No matching distribution found for tensorflow`, after downloading the other packages. This function therefore checks the interpreter before installing anything, and stops with an explanatory message if it is outside the supported range. Pass `python_version` to select a suitable interpreter, which requires `envname` so that a virtual environment can be built from it.

Value

`TRUE`, invisibly.

Examples

```
## Not run:
# Basic installation
install_r4lineups_python()

# Install to a virtual environment
install_r4lineups_python(envname = "r4lineups-face-env")

# Build the environment with a TensorFlow-supported interpreter
install_r4lineups_python(envname = "r4lineups-face-env",
                         python_version = "3.11")

# Install with GPU support
install_r4lineups_python(gpu = TRUE)

## End(Not run)
```

`line73` *line73*

Description

A dataframe of lineup choices for 1 trial; real data from a Doob & Kirshenbaum (1973) article. This article was the first to discuss lineup size 1 column x 42 observations

42 mock witnesses Nominal size is 6 Target position is 4

Usage

```
data(line73)
```

Format

An object of class `data.frame` with 42 rows and 1 columns.

References

Doob, A. N., & Kirshenbaum, H. M. (1973). Bias in police lineups-Partial remembering. *Journal of Police Science and Administration, 1*, 287-293.

`lineup_bias_boot_dist` *Bootstrap Distribution of Lineup Bias (Target Proportion)*

Description

Computes a bootstrap distribution for the target-position proportion.

Usage

```
lineup_bias_boot_dist(
  lineup_input,
  target_pos,
  k,
  input_type = c("vector", "table"),
  R = 1000
)
```

Arguments

`lineup_input`	A numeric lineup vector or a lineup table.
`target_pos`	Target position in the lineup (scalar).
`k`	Nominal lineup size (suspect + fillers).
`input_type`	Either "vector" or "table".
`R`	Number of bootstrap resamples.

Value

A numeric vector of bootstrap estimates, carrying an `"n"` attribute equal to the number of lineup choices resampled. Because each estimate is a count divided by `n`, the estimates fall on a discrete lattice with spacing `1/n`; `plot_lineup_bias_distribution` uses the attribute to bin at that resolution.

Examples

```
vec <- round(runif(200, 1, 6))
dist <- lineup_bias_boot_dist(vec, target_pos = 3, k = 6, R = 500)
```

`lineup_boot_allprop` *Confidence intervals for lineup proportion*

Description

Computes bootstrapped confidence intervals for lineup proportion

Usage

```
lineup_boot_allprop(lineup_vec, k, conf = 0.95, R = 1000)
```

Arguments

`lineup_vec`	A numeric vector of lineup choices
`k`	Number of targets in lineup. Must be specified by user (scalar).
`conf`	Confidence level for the intervals, e.g. 0.95 for 95 percent intervals. Defaults to 0.95. May be specified by user (scalar).
`R`	Number of bootstrap replications. Defaults to 1000.

Details

Computes a bootstrapped lineup proportion for each member in turn by calling the `boot` function in the `boot` package. Where a BCa interval cannot be computed, the percentile interval is used instead.

Note that the `k` intervals are each computed at the nominal `conf` level, with **no** adjustment for the fact that `k` of them are inspected together. Interpreting all of them at once inflates the chance that at least one excludes the chance value by more than the nominal amount. If a family-wise statement is wanted, raise the level for each interval accordingly, e.g. `conf = 1 - 0.05 / k` for a Bonferroni-style correction.

Value

A dataframe of `k` rows and two columns, `ci_low` and `ci_high`, holding the bias-corrected and accelerated (BCa) confidence interval for the proportion choosing each lineup member. Rows are named by lineup position.

References

Davison, A.C. & Hinkley, D.V. (1997). *Bootstrap methods and their application*. Cambridge University Press.

Wells, G. L., Leippe, M. R., & Ostrom, T. M. (1979). Guidelines for assessing the fairness of a lineup. *Law and Human Behavior, 3*(4), 285-293.

See Also

`boot`: https://cran.r-project.org/web/packages/boot/boot.pdf

Examples

```
#Data:
lineup_vec <- round(runif(100, 1, 6))

#Call:
lineuprops_ci <- lineup_boot_allprop(lineup_vec, k= 6)
lineuprops_ci <- lineup_boot_allprop(lineup_vec, k= 6, conf = 0.975)
```

---

lineup_example *Example Lineup Identification Data*

Description

A dataset containing simulated eyewitness lineup identification data with target-present and target-absent lineups, including confidence ratings.

Usage

```
lineup_example
```

Format

A data frame with 1400 rows and 5 variables:

target_present Logical indicating whether the guilty suspect was in the lineup

lineup_member Integer position of the lineup member chosen, with 0 for a rejection

suspect_pos Integer position occupied by the suspect (1 throughout)

identification Character indicating the witness decision: "suspect", "filler", or "reject"

confidence Numeric confidence rating (0-100 scale)

Details

This dataset is used for demonstrating ROC, CAC, and EIG analyses in the r4lineups package. It contains 700 target-present and 700 target-absent lineups of size 6.

`lineup_member` records *which* member each witness chose, so foils keep their separate identities and are not treated as interchangeable. They are not merely simulated separately: each foil is given its own mean memory strength, so some are strong alternatives to the suspect and others are chosen only rarely. In this dataset the five foils attract roughly 248, 203, 115, 53 and 10 identifications respectively.

`identification` is derived from `lineup_member` and `suspect_pos`, and can be recovered exactly with `classify_lineup_choices`; it is retained because the analysis functions in this package take it directly.

High confidence is strongly diagnostic here but deliberately *not* perfect: a small number of innocent suspects are identified at the highest confidence levels, as happens in real cases. Data in which no innocent suspect is ever identified with high confidence produces an infinite likelihood ratio and degenerate prior-posterior curves.

Source

Simulated with simulate_lineup_data. The generating script is data-raw/lineup_example.R, which carries the acceptance criteria the dataset must satisfy as explicit checks.

See Also

classify_lineup_choices, lineup_table

Examples

```
data(lineup_example)
str(lineup_example)
table(lineup_example$identification, lineup_example$target_present)

# Which lineup members were chosen, and were any never chosen?
lineup_table(lineup_example$lineup_member[lineup_example$lineup_member > 0],
             k = 6, suspect_pos = 1)
```

---

lineup_pairwise_matrix

Compute Pairwise Similarity Matrix for All Lineup Members

Description

Computes a full pairwise similarity matrix between all faces in a lineup, including the target. Useful for understanding the overall structure of face similarities in the lineup.

Usage

```
lineup_pairwise_matrix(
  target_path,
  foil_paths,
  model = "ArcFace",
  metric = "cosine",
  detector = "retinaface"
)
```

Arguments

target_path	Path to target face.
foil_paths	Paths to foil faces.
model	Face recognition model.
metric	Distance metric.
detector	Face detector.

Value

A square matrix of pairwise distances, with row/column names.

Examples

```
## Not run:
mat <- lineup_pairwise_matrix("target.jpg", c("foil1.jpg", "foil2.jpg"))
# Visualize with heatmap
heatmap(mat)

## End(Not run)
```

---

lineup_prop_boot *Bootstrapped lineup proportion*

Description

Base function for computing bootstrapped lineup proportion for a lineup member

Usage

```
lineup_prop_boot(lineup_vec, d, target_pos)
```

Arguments

lineup_vec	A numeric vector of lineup choices
d	Indices for bootstrap sample. Argument used by boot function to select samples for bootstrapping
target_pos	A scalar, representing target position in lineup. Must be declared by user

Details

Function to call when bootstrap resampling using boot function

References

Davison, A.C. & Hinkley, D.V. (1997). *Bootstrap methods and their application*. Cambridge University Press.

Wells, G. L.,Leippe, M. R., & Ostrom, T. M. (1979). Guidelines for empirically assessing the fairness of a lineup. *Law and Human Behavior, 3*(4), 285-293.

See Also

boot: https://cran.r-project.org/web/packages/boot/boot.pdf

Examples

```
lineup_vec <- round(runif(100, 1, 6))

bootobject <- boot::boot(lineup_vec, lineup_prop_boot, target_pos = 3, R = 1000)
cis <- boot::boot.ci(bootobject, conf = 0.95, type = "all", target_pos = 3)
```

---

lineup_prop_tab *Lineup proportion*

Description

Computes the proportion of mock witnesses identifying a particular lineup member

Usage

```
lineup_prop_tab(lineup_tab, target_pos)
```

Arguments

lineup_tab	A table of lineup choices
target_pos	A scalar, representing target position in lineup. Must be declared by user

Value

Returns a proportion indicating the frequency with which a lineup member was selected

References

Wells, G. L.,Leippe, M. R., & Ostrom, T. M. (1979). Guidelines for empirically assessing the fairness of a lineup. *Law and Human Behavior, 3*(4), 285-293.

Examples

```
#Data:
lineup_vec <- round(runif(100, 1))
lineup_tab <- table(lineup_vec)

#Call:
lineup_prop_tab(lineup_tab, 3)
lineup_prop_tab(table(lineup_vec), 2)
```

lineup_prop_vec *Lineup proportion*

Description

Computes the proportion of mock witnesses identifying a particular lineup member

Usage

```
lineup_prop_vec(lineup_vec, target_pos, k)
```

Arguments

lineup_vec	A numeric vector of lineup choices
target_pos	Suspect/lineup member position. Must be declared by user (scalar).
k	Nominal size (i.e., total number of members in lineup). Must be specified by user (scalar).

Value

Returns a proportion indicating the frequency with which a target was identified in a lineup

References

Wells, G. L.,Leippe, M. R., & Ostrom, T. M. (1979). Guidelines for empirically assessing the fairness of a lineup. *Law and Human Behavior, 3*(4), 285-293.

Examples

```
#Data:
lineup_vec <- round(runif(100, 1, 6))

#Call:
lineup_prop_vec(lineup_vec, 3, 6)
```

lineup_similarity *Compute Lineup Similarity Matrix*

Description

Computes similarity between a target face and all foils in a lineup, returning a comprehensive similarity analysis useful for lineup fairness assessment.

Usage

```
lineup_similarity(
  target_path,
  foil_paths,
  model = "ArcFace",
  metric = "cosine",
  detector = "retinaface",
  foil_names = NULL
)
```

Arguments

target_path	Path to the target (suspect) face image.
foil_paths	Character vector of paths to foil face images.
model	Character. Face recognition model. Default is "ArcFace".
metric	Character. Distance metric. Default is "cosine".
detector	Character. Face detection backend. Default is "retinaface".
foil_names	Optional character vector of names/labels for foils. If NULL, uses file names.

Details

This function describes how similar each foil is to the target in a face recognition model's embedding space. It does not assess fairness: no validated similarity range defines a fair lineup, and embedding similarity is not the same as resemblance judged by people or as mock-witness choice. Foils very dissimilar to the target may make the target stand out; foils the model judges to be the same identity (verified = TRUE) deserve inspection. Rows are ordered by rank (most similar first), and the model's distance threshold is stored in `attr(x, "threshold")`. For cosine, similarity = 1 - distance, which ranges from -1 to 1.

Value

A tibble with columns:

- foil_id - Numeric ID (1, 2, 3, ...)
- foil_name - Name/label for the foil
- foil_path - Path to foil image
- distance - Distance from target (lower = more similar)
- similarity - Cosine similarity score [-1,1] for cosine metric
- verified - Would be identified as same person
- rank - Rank by similarity (1 = most similar to target)

Also includes attributes:

- target_path - Path to target image
- model - Model used
- metric - Metric used
- threshold - Verification threshold

Examples

```
## Not run:
# Analyze a 6-person lineup
foils <- paste0("foil", 1:5, ".jpg")
lineup <- lineup_similarity("target.jpg", foils)

# View results sorted by similarity
print(lineup)

# Check which foils might be too similar
lineup[lineup$verified, ]

# Get summary statistics
summary(lineup)

## End(Not run)
```

`lineup_table` *Tabulate Lineup Choices, Including Null Foils*

Description

Builds a table of lineup choices with one entry for every lineup member, including members that no mock witness chose.

Usage

```
lineup_table(lineup_vec, k, suspect_pos = NULL)
```

Arguments

`lineup_vec`	A numeric vector of lineup choices.
`k`	Number of members in the lineup (suspect plus foils). Must be specified by the user (scalar).
`suspect_pos`	Optional position of the nominated suspect, marked in the printed output. Defaults to `NULL`, for lineups with no pre-specified suspect.

Details

Calling `table()` directly on a vector of lineup choices tabulates only the positions that were actually chosen. A lineup member that no witness selected — a *null foil* — is silently dropped, so the resulting table has fewer than `k` entries and nothing in the output reveals that a member is missing. This is important because a foil nobody would pick is not a functioning alternative to the suspect, one of several things that can make a lineup unfair. A member with no choices in a small sample is not necessarily such a foil.

`lineup_table()` tabulates over all `k` positions, so unchosen members appear explicitly as zero counts, and its print method names them.

The measures in this package already handle null foils correctly when given a complete table. `esize_m()`, for instance, reindexes counts by position and deducts unchosen members from the adjusted estimate. What `lineup_table()` adds is visibility, not a change in the arithmetic.

Value

An object of class `lineup_table`, inheriting from `table`, with `k` named entries. Because it inherits from `table` it can be passed to `esize_m`, `esize_T` and the other table-taking functions in this package unchanged.

References

Malpass, R. S. (1981). Effective size and defendant bias in eyewitness identification lineups. *Law and Human Behavior, 5*(4), 299-309.

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22*(2), 217-237.

See Also

`esize_m`, `esize_T`, `allprop`

Examples

```
# A lineup in which position 4 was never chosen
lineup_vec <- rep(c(5, 3, 2, 1, 8, 6, 7), each = 4)

# table() drops the unchosen member: only 7 entries for an 8-person lineup
table(lineup_vec)

# lineup_table() keeps it, and says so
lineup_table(lineup_vec, k = 8, suspect_pos = 3)

# Accepted by the effective size functions unchanged
esize_m(lineup_table(lineup_vec, k = 8), k = 8)
```

`ln_diag_ratio` *Ln of Diagnosticity Ratio*

Description

Computes ln of diagnosticity ratio: ln(d)

Usage

```
ln_diag_ratio(linedf, correction = TRUE)
```

Arguments

`linedf`	A dataframe of parameters for computing diagnosticity ratio
`correction`	Logical. Apply Tredoux's small-cell correction (default `TRUE`). Set to `FALSE` for the uncorrected log relative risk used by the homogeneity procedure, which requires positive suspect-ID counts.

Details

The correction adds 0.5 to each suspect-identification count and to its corresponding sample total before computing the ratio:

$$d = \frac{(n_{11} + 0.5)/(n_{11} + n_{21} + 0.5)}{(n_{12} + 0.5)/(n_{12} + n_{22} + 0.5)}$$

$\ln(d)$ is then returned. The correction avoids undefined log ratios when a cell count is zero and stabilises variance estimates in small samples.

To get linedf, use the diag_param helper function

diag_param returns a dataframe containing the following:

- *n11*: Number of mock witnesses who identified the suspect in the target present condition
- *n21*: Number of mock witnesses who did not identify the suspect in the target present condition
- *n12*: Number of mock witnesses who identified the suspect in the target absent condition
- *n22*: Number of mock witnesses who did not identify the suspect in the target absent condition

References

Malpass, R. S. (1981). Effective size and defendant bias in eyewitness identification lineups. *Law and Human Behavior, 5*(4), 299-309.

Malpass, R. S., Tredoux, C., & McQuiston-Surrett, D. (2007). Lineup construction and lineup fairness. In R. Lindsay, D. F. Ross, J. D. Read, & M. P. Toglia (Eds.), *Handbook of Eyewitness Psychology, Vol. 2: Memory for people* (pp. 155-178). Mahwah, NJ: Lawrence Erlbaum Associates.

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22*(2), 217-237.

Tredoux, C. (1999). Statistical considerations when determining measures of lineup size and lineup bias. *Applied Cognitive Psychology, 13*, S9-S26.

Wells, G. L.,Leippe, M. R., & Ostrom, T. M. (1979). Guidelines for empirically assessing the fairness of a lineup. *Law and Human Behavior, 3*(4), 285-293.

Examples

```
#Target present data:
A <- rep(1:6, length.out = 100)
B <- rep(1:5, length.out = 70)
C <- rep(1:4, length.out = 20)
lineup_pres_list <- list(A, B, C)
rm(A, B, C)
```

```
#Target absent data:
A <- rep(6:1, length.out = 100)
B <- rep(5:1, length.out = 70)
C <- rep(4:1, length.out = 20)
lineup_abs_list <- list(A, B, C)
rm(A, B, C)

# Suspect position for each TP/TA pair
pos_list <- c(3, 2, 1)

#Nominal size:
k <- c(6, 5, 4)

#Use diag param helper function to get data (n11, n21, n12, n22):
linedf <- diag_param(lineup_pres_list, lineup_abs_list, pos_list, k)

#Call:
lnd <- ln_diag_ratio(linedf)
```

`make_bayes_curves` *Compute Bayesian Prior-Posterior and Information Gain Curves*

Description

Computes Bayesian updating curves showing how different lineup responses update beliefs about suspect guilt, following Wells, Yang, & Smalarz (2015).

Usage

```
make_bayes_curves(
  data,
  response_categories = c("simple", "confidence"),
  confidence_bins = NULL,
  prior_grid = seq(0.01, 0.99, 0.01),
  lineup_size = 6,
  innocent_suspect = "auto"
)
```

Arguments

`data` A dataframe with the following columns:

- target_present: Logical. TRUE if guilty suspect in lineup
- identification: Character. "suspect", "filler", or "reject"
- confidence: Numeric. Confidence rating (optional for response categories)

`response_categories`
: Character vector specifying how to categorize responses. Options:
  - "simple": Three categories (suspect, filler, reject)
  - "confidence": Combines identification with confidence bins

`confidence_bins`
: Numeric vector of bin edges if using confidence categories (e.g., c(0, 60, 80, 100)). Ignored if response_categories = "simple".

`prior_grid`
: Numeric vector of prior probabilities to evaluate (default: seq(0.01, 0.99, 0.01))

`lineup_size`
: Nominal lineup size, used when target-absent data have no designated innocent suspect.

`innocent_suspect`
: How innocent-suspect identifications in target-absent lineups are obtained. `"designated"`: the lineups had a designated innocent suspect, so target-absent `"suspect"` rows are counted directly, and zero such rows means a false-ID rate of zero. `"estimated"`: there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by `lineup_size`. `"auto"` (the default) chooses `"designated"` when any target-absent row is a suspect ID or the data have a non-missing `suspect_pos` column, and `"estimated"` otherwise, with a message.

Details

This function treats lineup outcomes as evidence for Bayesian updating. For each response type x (e.g., high-confidence suspect ID, rejection, etc.), it computes:

Likelihoods from observed data:

$$p(x|guilty) = \frac{count(x|target_present)}{N_{guilty}}$$

$$p(x|innocent) = \frac{count(x|target_absent)}{N_{innocent}}$$

For each prior probability p(guilty):

$$p(x) = p(guilty) \cdot p(x|guilty) + (1 - p(guilty)) \cdot p(x|innocent)$$

$$p(guilty|x) = \frac{p(guilty) \cdot p(x|guilty)}{p(x)}$$

Information gain:

$$IG(x) = H(prior) - H(posterior)$$

where H(p) is binary entropy.

The information gain quantifies how much uncertainty is reduced by observing response x. It is negative when the response moves belief towards 0.5, as a suspect identification does at a low prior; that is a correct update that leaves more uncertainty, not misleading evidence.

Value

A list containing:

- curves: Dataframe with prior, posterior, and information gain for each response
- likelihoods: Dataframe with p(x|guilty) and p(x|innocent) for each response
- response_counts: Count of each response type by target presence
- n_guilty: Number of target-present lineups
- n_innocent: Number of target-absent lineups

References

Wells, G. L., Yang, Y., & Smalarz, L. (2015). Eyewitness identification: Bayesian information gain, base-rate effect-equivalency curves, and reasonable suspicion. *Law and Human Behavior, 39*(2), 99-122.

Examples

```
data(lineup_example)
bayes <- make_bayes_curves(lineup_example)
bayes$likelihoods
```

`make_bree_curve` *Compute Base-Rate Effect-Equivalency (BREE) Curves*

Description

Computes BREE curves showing the base-rate shift required for one procedure to produce the same posterior probability as another procedure, following Wells, Yang, & Smalarz (2015).

Usage

```
make_bree_curve(
  data_proc_a,
  data_proc_b,
  reference_response = "suspect",
  confidence_bins = NULL,
  prior_grid = seq(0.01, 0.99, 0.01),
  lineup_size = 6,
  innocent_suspect = "auto"
)
```

Arguments

`data_proc_a`	A dataframe for procedure A with columns: target_present, identification, confidence (standard lineup data format)
`data_proc_b`	A dataframe for procedure B (same format as proc_a)
`reference_response`	Character string specifying which response to compare (e.g., "suspect" for suspect IDs). Default = "suspect".
`confidence_bins`	Optional numeric vector of bin edges if using confidence-based response (e.g., c(0, 60, 80, 100)). If provided, reference_response should specify both identification and bin (e.g., "suspect_(80,100]")
`prior_grid`	Numeric vector of prior probabilities for procedure A (default: seq(0.01, 0.99, 0.01))
`lineup_size`	Nominal lineup size for procedures without a designated innocent suspect.
`innocent_suspect`	How innocent-suspect identifications in target-absent lineups are obtained. `"designated"`: the lineups had a designated innocent suspect, so target-absent `"suspect"` rows are counted directly, and zero such rows means a false-ID rate of zero. `"estimated"`: there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by `lineup_size`. `"auto"` (the default) chooses `"designated"` when any target-absent row is a suspect ID or the data have a non-missing `suspect_pos` column, and `"estimated"` otherwise, with a message.

Details

BREE curves answer the question: "How much would the base rate need to change for Procedure B to yield the same posterior probability as Procedure A?"

For each prior probability in procedure A: 1. Compute the posterior p(guilty|x) for the reference response in procedure A 2. Find the prior probability in procedure B that yields the same posterior 3. Delta = prior_B - prior_A

Interpretation:

- Delta > 0: Procedure B requires a higher base rate to match A's posterior (A is more diagnostic)
- Delta < 0: Procedure B requires a lower base rate (B is more diagnostic)
- Delta = 0: Procedures are equally diagnostic

Value

A list containing:

- bree_curve: Dataframe with prior_a, posterior_a, prior_b, and delta
- proc_a_likelihoods: Likelihoods for procedure A
- proc_b_likelihoods: Likelihoods for procedure B
- reference_response: The response category used for comparison

References

Wells, G. L., Yang, Y., & Smalarz, L. (2015). Eyewitness identification: Bayesian information gain, base-rate effect-equivalency curves, and reasonable suspicion. *Law and Human Behavior, 39*(2), 99-122.

Examples

```
data(lineup_example)
odd <- seq(1, nrow(lineup_example), by = 2)
bree <- make_bree_curve(lineup_example[odd, ], lineup_example[-odd, ])
head(bree$bree_curve)
```

`make_cac` *Compute and Plot CAC for Lineup Identification*

Description

Main function to compute and plot CAC (Confidence-Accuracy Characteristic) for eyewitness lineup data.

Usage

```
make_cac(
  data,
  lineup_size = 6,
  confidence_bins = NULL,
  show_plot = TRUE,
  innocent_suspect = "auto",
  base_rate = NULL,
  ...
)
```

Arguments

`data`	A dataframe with columns: target_present, identification, confidence
`lineup_size`	Integer. Number of people in lineup (default = 6)
`confidence_bins`	Numeric vector of bin edges (optional)
`show_plot`	Logical. Whether to display the plot (default = TRUE)
`innocent_suspect`	How innocent-suspect identifications in target-absent lineups are obtained. `"designated"`: the lineups had a designated innocent suspect, so target-absent `"suspect"` rows are counted directly, and zero such rows means a false-ID rate of zero. `"estimated"`: there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by `lineup_size`. `"auto"` (the default) chooses

	"designated" when any target-absent row is a suspect ID or the data have a non-missing suspect_pos column, and "estimated" otherwise, with a message.
base_rate	See make_cacdata.
...	Additional arguments passed to make_cac_gg()

Value

A list containing CAC data and plot

Examples

```
data(lineup_example)
cac_result <- make_cac(lineup_example, confidence_bins = c(0, 60, 80, 100))
print(cac_result)
```

---

make_cac_gg *Plot CAC Curve*

Description

Creates a ggplot2 visualization of CAC data showing the relationship between confidence and accuracy.

Usage

```
make_cac_gg(cacobj_list, show_errorbars = TRUE, show_n = TRUE)
```

Arguments

cacobj_list	List output from make_cacdata()
show_errorbars	Logical. Whether to show error bars (default = TRUE)
show_n	Logical. Whether to show sample sizes (default = TRUE)

Value

A ggplot2 object

Examples

```
data(lineup_example)
cac <- make_cacdata(lineup_example, confidence_bins = c(0, 60, 80, 100))
make_cac_gg(cac)
```

`make_cacdata` *Compute CAC (Confidence-Accuracy Characteristic) Data*

Description

Computes confidence-accuracy data for lineup identifications following Mickes (2015). CAC analysis shows the relationship between confidence and accuracy for suspect identifications.

Usage

```
make_cacdata(
  data,
  lineup_size = 6,
  confidence_bins = NULL,
  innocent_suspect = "auto",
  base_rate = NULL
)
```

Arguments

`data` A dataframe with the following columns:

- target_present: Logical. TRUE if guilty suspect in lineup
- identification: Character. "suspect", "filler", or "reject"
- confidence: Numeric. Confidence rating

`lineup_size` Integer. Number of people in lineup (default = 6)

`confidence_bins`
Numeric vector of bin edges for grouping confidence (e.g., c(0, 60, 80, 100) creates bins 0-60, 61-80, 81-100). If NULL, uses individual confidence levels.

`innocent_suspect`
How innocent-suspect identifications in target-absent lineups are obtained. `"designated"`: the lineups had a designated innocent suspect, so target-absent `"suspect"` rows are counted directly, and zero such rows means a false-ID rate of zero. `"estimated"`: there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by `lineup_size`. `"auto"` (the default) chooses `"designated"` when any target-absent row is a suspect ID or the data have a non-missing `suspect_pos` column, and `"estimated"` otherwise, with a message.

`base_rate` Prior probability that the suspect is guilty, used to compute accuracy from rates rather than counts. `NULL` (default) keeps the count definition of Mickes (2015). A number strictly between 0 and 1 gives $\pi HR(c)/(\pi HR(c)+(1-\pi)FAR(c))$; see Details.

Details

CAC analysis computes, for each confidence level:

$$Accuracy = \frac{CorrectSuspectIDs}{CorrectSuspectIDs + IncorrectSuspectIDs}$$

Explicit target-absent suspect IDs are used when present. Otherwise, target-absent filler IDs are divided by lineup size to estimate incorrect suspect IDs. The two estimators are never added together.

The count ratio equals the probability of guilt given a suspect ID only when the target-present and target-absent samples are the same size, which Mickes (2015) assumes; she notes that unequal base rates make the calculation more complicated. With unequal samples the count-based accuracy shifts with the TP:TA split even when hit and false-ID rates are unchanged, and the function says so in a message. Setting `base_rate` computes accuracy from the hit rate HR(c) = correct IDs / n(TP) and the false-ID rate FAR(c) = incorrect IDs / n(TA) at the stated prior, so that `base_rate = 0.5` gives the balanced-design value whatever the sample sizes. Its standard error is then a delta-method approximation from the two independent binomial rates; the count-based standard error is binomial over the bin's suspect IDs.

According to Mickes (2015), CAC analysis is most relevant for triers of fact (judges/jurors) evaluating estimator variables (e.g., exposure duration, retention interval). It shows how trustworthy an ID is at different confidence levels.

Value

A list containing:

- cac_data: Dataframe with confidence levels and accuracy
- overall_accuracy: Overall accuracy across all suspect IDs
- n_total_suspect_ids: Correct plus incorrect suspect IDs
- base_rate: The base rate used, or NULL for count-based accuracy
- n_target_present, n_target_absent: Sample sizes of the two arms

References

Mickes, L. (2015). Receiver operating characteristic analysis and confidence-accuracy characteristic analysis in investigations of system variables and estimator variables that affect eyewitness memory. *Journal of Applied Research in Memory and Cognition, 4*(2), 93-102.

Juslin, P., Olsson, N., & Winman, A. (1996). Calibration and diagnosticity of confidence in eyewitness identification: Comments on what can be inferred from the low confidence-accuracy correlation. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 22*(5), 1304-1316.

Examples

```
data(lineup_example)
cac <- make_cacdata(lineup_example, confidence_bins = c(0, 60, 80, 100))
cac$cac_data
```

`make_calibration` *Compute and Plot Calibration for Lineup Identification*

Description

Main function to compute and plot calibration analysis for eyewitness lineup data.

Usage

```
make_calibration(
  data,
  confidence_bins = NULL,
  choosers_only = TRUE,
  lineup_size = 6,
  show_plot = TRUE,
  confidence_scale = c("auto", "0-1", "0-100"),
  innocent_suspect = "auto",
  ...
)
```

Arguments

`data`	A dataframe with columns: target_present, identification, confidence
`confidence_bins`	Numeric vector of bin edges (optional)
`choosers_only`	Logical. Whether to analyze only suspect IDs (default = TRUE)
`lineup_size`	Integer. Number of people in lineup (default = 6)
`show_plot`	Logical. Whether to display the plot (default = TRUE)
`confidence_scale`	How the confidence scale is interpreted: "auto" (default), "0-1", or "0-100". See `make_calibration_data`.
`innocent_suspect`	How innocent-suspect identifications in target-absent lineups are obtained. `"designated"`: the lineups had a designated innocent suspect, so target-absent `"suspect"` rows are counted directly, and zero such rows means a false-ID rate of zero. `"estimated"`: there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by `lineup_size`. `"auto"` (the default) chooses `"designated"` when any target-absent row is a suspect ID or the data have a non-missing `suspect_pos` column, and `"estimated"` otherwise, with a message.
`...`	Additional arguments passed to make_calibration_gg()

Value

A list containing calibration data, statistics, and plot

Examples

```
set.seed(123)
n <- 200
lineup_data <- data.frame(
  target_present = rep(c(TRUE, FALSE), each = n / 2),
  identification = sample(c("suspect", "filler", "reject"), n,
                          replace = TRUE, prob = c(0.5, 0.25, 0.25)),
  confidence = round(runif(n, 0, 100))
)
cal_result <- make_calibration(lineup_data,
                               confidence_bins = c(0, 60, 80, 100))
print(cal_result)
```

---

make_calibration_by_condition

Compute Calibration Statistics by Condition

Description

Computes calibration analysis separately for different experimental conditions. Useful for examining how calibration varies across lineup instructions, foil similarity, or other system/estimator variables.

Usage

```
make_calibration_by_condition(
  data,
  condition_vars,
  confidence_bins = NULL,
  choosers_only = TRUE,
  lineup_size = 6,
  confidence_scale = c("auto", "0-1", "0-100"),
  innocent_suspect = "auto"
)
```

Arguments

data	A dataframe with columns: target_present, identification, confidence, plus one or more condition variables
condition_vars	Character vector of column names defining conditions (e.g., c("instruction_type", "foil_similarity"))
confidence_bins	Numeric vector of bin edges (optional)
choosers_only	Logical. Whether to analyze only suspect IDs (default = TRUE)
lineup_size	Integer. Number of people in lineup (default = 6)

`confidence_scale`
: How the confidence scale is interpreted: "auto" (default), "0-1", or "0-100". See `make_calibration_data`.

`innocent_suspect`
: How innocent-suspect identifications in target-absent lineups are obtained. `"designated"`: the lineups had a designated innocent suspect, so target-absent `"suspect"` rows are counted directly, and zero such rows means a false-ID rate of zero. `"estimated"`: there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by `lineup_size`. `"auto"` (the default) chooses `"designated"` when any target-absent row is a suspect ID or the data have a non-missing `suspect_pos` column, and `"estimated"` otherwise, with a message.

Details

This function splits the data by the specified condition variables and computes calibration statistics separately for each combination. Useful for examining questions like "Does calibration improve with biased vs unbiased instructions?" or "How does foil similarity affect over/underconfidence?"

Value

A list containing:

- by_condition: List of calibration results for each condition combination
- condition_summary: Dataframe summarizing C, O/U, NRI across conditions

References

Brewer, N., & Wells, G. L. (2006). The confidence-accuracy relationship in eyewitness identification: Effects of lineup instructions, foil similarity, and target-absent base rates. *Journal of Experimental Psychology: Applied, 12*(1), 11-30.

Examples

```
set.seed(123)
n <- 300
data <- data.frame(
  target_present = rep(c(TRUE, FALSE), each = n / 2),
  identification = sample(c("suspect", "filler", "reject"), n,
                          replace = TRUE, prob = c(0.5, 0.25, 0.25)),
  confidence = round(runif(n, 0, 100)),
  instruction = rep(c("biased", "unbiased"), times = n / 2)
)
by_cond <- make_calibration_by_condition(data, condition_vars = "instruction",
                                         confidence_bins = c(0, 60, 80, 100))
by_cond$condition_summary
```

make_calibration_by_condition_gg *Plot Calibration Curves by Condition*

Description

Creates a faceted or overlaid plot comparing calibration across conditions.

Usage

```
make_calibration_by_condition_gg(
  cal_by_cond_obj,
  facet = TRUE,
  show_stats = TRUE
)
```

Arguments

cal_by_cond_obj	List output from make_calibration_by_condition()
facet	Logical. If TRUE (default), creates faceted plot. If FALSE, overlays curves.
show_stats	Logical. Whether to display statistics (default = TRUE when faceted)

Value

A ggplot2 object

Examples

```
set.seed(123)
n <- 300
data <- data.frame(
  target_present = rep(c(TRUE, FALSE), each = n / 2),
  identification = sample(c("suspect", "filler", "reject"), n,
                          replace = TRUE, prob = c(0.5, 0.25, 0.25)),
  confidence = round(runif(n, 0, 100)),
  instruction = rep(c("biased", "unbiased"), times = n / 2)
)
by_cond <- make_calibration_by_condition(data, condition_vars = "instruction",
                                         confidence_bins = c(0, 60, 80, 100))
make_calibration_by_condition_gg(by_cond)
```

`make_calibration_data` *Compute Calibration Statistics for Eyewitness Identification*

Description

Computes calibration analysis metrics following Juslin, Olsson, & Winman (1996). Calibration analysis assesses the match between confidence and accuracy, providing a more appropriate measure than simple confidence-accuracy correlation.

Usage

```
make_calibration_data(
  data,
  confidence_bins = NULL,
  choosers_only = TRUE,
  lineup_size = 6,
  confidence_scale = c("auto", "0-1", "0-100"),
  innocent_suspect = "auto"
)
```

Arguments

`data` A dataframe with the following columns:

- target_present: Logical. TRUE if guilty suspect in lineup
- identification: Character. "suspect", "filler", or "reject"
- confidence: Numeric. Confidence rating

`confidence_bins` Numeric vector of bin edges for grouping confidence (e.g., c(0, 60, 80, 100) creates bins 0-60, 60-80, 80-100). If NULL, uses individual confidence levels.

`choosers_only` Logical. If TRUE (default), only analyze suspect identifications. If FALSE, analyze all responses (including fillers and rejections).

`lineup_size` Integer. Number of people in lineup (default = 6). Used for estimating incorrect suspect IDs from filler choices in target-absent lineups.

`confidence_scale` How the confidence scale is interpreted: "auto" (default; max <= 1 treated as 0-1, otherwise as 0-100, with a warning for ambiguous Likert/0-10 scales), "0-1", or "0-100". C and O/U require a 0-1/0-100 probability scale.

`innocent_suspect` How innocent-suspect identifications in target-absent lineups are obtained. `"designated"`: the lineups had a designated innocent suspect, so target-absent `"suspect"` rows are counted directly, and zero such rows means a false-ID rate of zero. `"estimated"`: there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by `lineup_size`. `"auto"` (the default) chooses `"designated"` when any target-absent row is a suspect ID or the data have a non-missing `suspect_pos` column, and `"estimated"` otherwise, with a message.

Details

Calibration analysis distinguishes between:

- **Calibration (C)**: How well confidence matches accuracy (perfect = 0)
- **Over/underconfidence (O/U)**: Overall bias in confidence judgments (0 = perfectly calibrated)
- **Resolution (NRI)**: Ability to discriminate correct from incorrect responses

The calibration statistic C measures the weighted mean squared deviation between mean confidence and accuracy in each bin:

$$C = \sum_{j=1}^{J} \frac{n_j}{N}(c_j - a_j)^2$$

The Normalized Resolution Index (NRI) captures how well confidence discriminates correct from incorrect responses:

$$NRI = \frac{\frac{1}{N}\sum_{j=1}^{J} n_j(a_j - \bar{a})^2}{\bar{a}(1-\bar{a})}$$

When `choosers_only = TRUE`, only suspect identifications are included (standard for eyewitness calibration analysis). When `choosers_only = FALSE`, all responses are included with fillers and rejections counted as incorrect. Consequently that option measures correct-suspect identification across all responses, not general decision accuracy: a correct TA rejection still scores zero. Do not interpret it as calibration of the correctness of every decision.

If target-absent data contain designated innocent-suspect identifications, those observations are used directly. Otherwise, each target-absent filler identification contributes weight `1 / lineup_size` as an estimate of innocent-suspect identifications. In that latter case the per-bin counts and effective total used by C, O/U, NRI, and downstream ANRI can be fractional; the results are lineup-size approximations rather than statistics from an observed designated-suspect outcome.

Value

A list containing:

- calibration_data: Dataframe with per-bin confidence, accuracy, and counts
- C: Calibration statistic (weighted mean squared difference between confidence and accuracy)
- OU: Over/underconfidence (mean confidence minus mean accuracy)
- NRI: Normalized Resolution Index (standardized within-person variance)
- overall_accuracy: Mean accuracy across all responses
- overall_confidence: Mean confidence across all responses
- n_total: Total number of responses analyzed

References

Juslin, P., Olsson, N., & Winman, A. (1996). Calibration and diagnosticity of confidence in eyewitness identification: Comments on what can be inferred from the low confidence-accuracy correlation. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 22*(5), 1304-1316.

Brewer, N., & Wells, G. L. (2006). The confidence-accuracy relationship in eyewitness identification: Effects of lineup instructions, foil similarity, and target-absent base rates. *Journal of Experimental Psychology: Applied, 12*(1), 11-30.

Examples

```
set.seed(123)
n <- 200
data <- data.frame(
  target_present = rep(c(TRUE, FALSE), each = n / 2),
  identification = sample(c("suspect", "filler", "reject"), n,
                          replace = TRUE, prob = c(0.5, 0.25, 0.25)),
  confidence = round(runif(n, 0, 100))
)
cal <- make_calibration_data(data, confidence_bins = c(0, 60, 80, 100))
cal$C
cal$calibration_data
```

---

make_calibration_gg *Plot Calibration Curve*

Description

Creates a calibration curve plot showing the relationship between confidence and accuracy. A perfectly calibrated witness would fall on the diagonal line.

Usage

```
make_calibration_gg(
  cal_obj,
  show_stats = TRUE,
  show_n = TRUE,
  show_diagonal = TRUE
)
```

Arguments

cal_obj	List output from make_calibration_data()
show_stats	Logical. Whether to display calibration statistics on plot (default = TRUE)
show_n	Logical. Whether to show sample sizes per bin (default = TRUE)
show_diagonal	Logical. Whether to show perfect calibration line (default = TRUE)

Details

The calibration curve plots mean confidence (x-axis) against accuracy (y-axis) for each confidence bin. Points on the diagonal indicate perfect calibration. Points above the diagonal indicate underconfidence, while points below indicate overconfidence.

Value

A ggplot2 object

Examples

```
set.seed(123)
n <- 200
data <- data.frame(
  target_present = rep(c(TRUE, FALSE), each = n / 2),
  identification = sample(c("suspect", "filler", "reject"), n,
                          replace = TRUE, prob = c(0.5, 0.25, 0.25)),
  confidence = round(runif(n, 0, 100))
)
cal <- make_calibration_data(data, confidence_bins = c(0, 60, 80, 100))
make_calibration_gg(cal)
```

---

make_dpp *Compute Deviation from Perfect Performance (DPP)*

Description

Computes the average deviation from perfect performance of Smith, Lampinen, Wells, Smalarz and Mackovichova (2019). DPP is calculated at every operating point (cumulative confidence threshold) of the ROC and averaged over them, so that, unlike pAUC, it is not tied to a region of false-ID rates that two differently truncated ROC curves may not share.

Usage

```
make_dpp(
  data,
  lineup_size = 6,
  use_roc_obj = FALSE,
  alpha = NULL,
  beta = NULL,
  innocent_suspect = "auto",
  thresholds = NULL
)
```

Arguments

`data`	A dataframe with the following columns: • target_present: Logical. TRUE if guilty suspect in lineup • identification: Character. "suspect", "filler", or "reject" • confidence: Numeric. Confidence rating Alternatively, with `use_roc_obj = TRUE`, the output of `make_rocdata`.
`lineup_size`	Integer. Number of people in lineup (default = 6)
`use_roc_obj`	Logical. If FALSE (default), computes ROC from data. If TRUE, data should be the output from make_rocdata(), whose first `roc_data` row is the origin and is not an operating point.

`alpha`, `beta` Optional weights for Smith et al.'s Equation 3. `alpha` is the cost of an innocent-suspect identification relative to a missed culprit identification, and `beta` is the base rate of culprit presence. Supply both or neither. When both are `NULL` (default), the unweighted Equation 2 is used.

`innocent_suspect`
How innocent-suspect identifications in target-absent lineups are obtained. `"designated"`: the lineups had a designated innocent suspect, so target-absent `"suspect"` rows are counted directly, and zero such rows means a false-ID rate of zero. `"estimated"`: there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by `lineup_size`. `"auto"` (the default) chooses `"designated"` when any target-absent row is a suspect ID or the data have a non-missing `suspect_pos` column, and `"estimated"` otherwise, with a message.

`thresholds` Optional numeric vector of confidence criteria at which to evaluate the ROC. The default uses the confidence levels of the responses that can move an operating point; see Details. Supplying the criteria of a full sample keeps them fixed across bootstrap resamples.

Details

At an operating point c, Smith et al.'s (2019) Equation 2 is

$$DPP(c) = 1 - Culprit(c) + Innocent(c)$$

where $Culprit(c)$ is the culprit (correct) identification rate and $Innocent(c)$ the innocent-suspect identification rate at that point. DPP(c) ranges from 0 (perfect performance: every culprit identified, no innocent suspect identified) to 2 (the reverse). Any point on the chance diagonal has DPP(c) = 1. When the base rate of culprit presence is .5 and the costs of the two errors are equal, DPP(c) = 1 - U(c), where $U(c) = Culprit(c) - Innocent(c)$ is the expected utility.

The reported `dpp` is the mean of DPP(c) over the operating points, one for each confidence criterion of the ROC, from the highest to the lowest (which includes every identification). Lower values are better. The criteria are the confidence levels of the responses that enter the ROC (see `make_rocdata`), so the confidence given to rejections does not add points or change the average.

Equation 3 weights the two errors:

$$DPP_{\alpha\beta}(c) = \beta[1 - Culprit(c)] + \alpha(1 - \beta)Innocent(c)$$

With `alpha = 1` and `beta = 0.5` this is half of Equation 2. Weighted values no longer range from 0 to 2.

Value

A list containing:

- dpp: Average DPP over the operating points
- dpp_by_point: Tibble with confidence threshold, culprit (correct-ID) rate, innocent-suspect ID rate and DPP at each operating point

- best_point: The operating point with the lowest DPP, which is the point of maximum utility
- roc_data: ROC curve data from `make_rocdata()`
- alpha, beta: The weights used (`NULL` for Equation 2)
- n_target_present, n_target_absent: Sample sizes
- innocent_suspect_method: How innocent-suspect IDs were obtained
- thresholds: The confidence criteria of the operating points

References

Smith, A. M., Lampinen, J. M., Wells, G. L., Smalarz, L., & Mackovichova, S. (2019). Deviation from perfect performance measures the diagnostic utility of eyewitness lineups but partial area under the ROC curve does not. *Journal of Applied Research in Memory and Cognition, 8*(1), 50-59. doi:10.1016/j.jarmac.2018.09.003

Examples

```
data(lineup_example)
dpp <- make_dpp(lineup_example)
dpp$dpp
dpp$dpp_by_point

# Innocent-suspect IDs twice as costly as misses, culprit base rate .5
make_dpp(lineup_example, alpha = 2, beta = 0.5)$dpp
```

---

make_eig *Main Function to Compute and Visualize EIG*

Description

Convenience wrapper that computes EIG and creates visualizations.

Usage

```
make_eig(
  data,
  prior_guilt = 0.5,
  confidence_bins = NULL,
  show_plot = TRUE,
  plot_type = "both",
  lineup_size = 6,
  innocent_suspect = "auto"
)
```

Arguments

`data`	A dataframe with columns: target_present, identification, confidence
`prior_guilt`	Numeric. Prior probability that suspect is guilty (default = 0.5)
`confidence_bins`	Numeric vector of bin edges (optional)
`show_plot`	Logical. Whether to create plots (default = TRUE)
`plot_type`	Character. Which plot to create: "ig" (information gain), "posteriors" (posterior probabilities), or "both" (default)
`lineup_size`	Nominal lineup size used when no designated innocent suspect is present.
`innocent_suspect`	How innocent-suspect identifications in target-absent lineups are obtained. `"designated"`: the lineups had a designated innocent suspect, so target-absent `"suspect"` rows are counted directly, and zero such rows means a false-ID rate of zero. `"estimated"`: there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by `lineup_size`. `"auto"` (the default) chooses `"designated"` when any target-absent row is a suspect ID or the data have a non-missing `suspect_pos` column, and `"estimated"` otherwise, with a message.

Value

A list of class "lineup_eig" containing:

- eig: Expected Information Gain value
- response_data: Full response category data
- plot_ig: ggplot of information gain (if requested)
- plot_posteriors: ggplot of posteriors (if requested)
- ... (additional components from compute_eig)

Examples

```
data(lineup_example)
# With confidence binning
result <- make_eig(lineup_example, confidence_bins = c(0, 60, 80, 100))

# Access plots
result$plot_ig
result$plot_posteriors
```

`make_eig_data` *Prepare EIG Data from Lineup Identification*

Description

Prepares response category data for Expected Information Gain (EIG) analysis following Starns et al. (2023). Creates response categories from identification decisions and confidence levels.

Usage

```
make_eig_data(
  data,
  confidence_bins = NULL,
  lineup_size = 6,
  innocent_suspect = "auto"
)
```

Arguments

`data`	A dataframe with the following columns: • target_present: Logical. TRUE if guilty suspect in lineup • identification: Character. "suspect", "filler", or "reject" • confidence: Numeric. Confidence rating
`confidence_bins`	Numeric vector of bin edges for grouping confidence (e.g., c(0, 60, 80, 100) creates bins 0-60, 61-80, 81-100). If NULL, uses individual confidence levels.
`lineup_size`	Nominal lineup size. Used to split target-absent filler identifications when no designated innocent suspect is present.
`innocent_suspect`	How innocent-suspect identifications in target-absent lineups are obtained. `"designated"`: the lineups had a designated innocent suspect, so target-absent `"suspect"` rows are counted directly, and zero such rows means a false-ID rate of zero. `"estimated"`: there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by `lineup_size`. `"auto"` (the default) chooses `"designated"` when any target-absent row is a suspect ID or the data have a non-missing `suspect_pos` column, and `"estimated"` otherwise, with a message.

Details

Response categories are defined as combinations of identification decision (suspect/filler/reject) and confidence level or bin. This function counts the frequency of each response category separately for target-present (guilty) and target-absent (innocent) lineups.

Value

A list containing:

- response_data: Dataframe with response categories and frequencies
- n_guilty: Total number of target-present lineups
- n_innocent: Total number of target-absent lineups
- confidence_bins: The binning specification used (if any)

References

Starns, J. J., Cohen, A. L., & Rotello, C. M. (2023). A complete method for assessing the effectiveness of eyewitness identification procedures: Expected information gain. *Psychological Review, 130*(3), 677–719.

Examples

```
data(lineup_example)
eig_data <- make_eig_data(lineup_example, confidence_bins = c(0, 60, 80, 100))
eig_data$response_data
```

---

make_fullroc *Compute and Plot Full ROC Curve (Smith & Yang, 2020)*

Description

Main function to compute and plot a full ROC curve using all eyewitness responses (suspect ID, filler ID, rejection) following Smith & Yang (2020).

Usage

```
make_fullroc(
  data,
  conf_bins = NULL,
  order = c("diagnosticity", "apriori"),
  lineup_size = 6,
  show_plot = TRUE,
  epsilon = 0.001,
  innocent_suspect = "auto",
  ...
)
```

Arguments

`data`	A dataframe with columns: target_present, identification, confidence
`conf_bins`	Numeric vector for confidence bin boundaries (default = NULL)
`order`	Character. "diagnosticity" or "apriori" (default = "diagnosticity")
`lineup_size`	Integer. Number of people in lineup (default = 6)
`show_plot`	Logical. Whether to display the plot (default = TRUE)
`epsilon`	Numeric. Small value for FAR = 0 cases (default = 0.001)
`innocent_suspect`	How innocent-suspect identifications in target-absent lineups are obtained. `"designated"`: the lineups had a designated innocent suspect, so target-absent `"suspect"` rows are counted directly, and zero such rows means a false-ID rate of zero. `"estimated"`: there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by `lineup_size`. `"auto"` (the default) chooses `"designated"` when any target-absent row is a suspect ID or the data have a non-missing `suspect_pos` column, and `"estimated"` otherwise, with a message.
`...`	Additional arguments passed to plot_fullroc()

Details

This is the main user-facing function for computing full ROC curves. It calls make_fullroc_data() to compute the ROC, then plot_fullroc() to visualize it.

Value

A list containing:

- plot: ggplot2 object. Always returned, whatever show_plot is set to; show_plot controls only whether it is displayed immediately
- roc_data: Dataframe with ROC curve points
- auc: Full area under the curve
- diagnosticity_table: Table with diagnosticity ratios
- summary: Summary statistics

References

Smith, A. M., Yang, Y., & Wells, G. L. (2020). Distinguishing between investigator discriminability and eyewitness discriminability: A method for creating full receiver operating characteristic curves of lineup identification performance. *Perspectives on Psychological Science, 15*(3), 589-607.

Examples

```
data(lineup_example)
# Compute and plot full ROC
result <- make_fullroc(lineup_example)
print(result$auc)
```

```
# With custom confidence bins
result2 <- make_fullroc(lineup_example, conf_bins = c(0, 60, 80, 100))
```

`make_fullroc_data` *Compute Full ROC Data for Lineup Identification (Smith & Yang, 2020)*

Description

Computes full Receiver Operating Characteristic (ROC) data using ALL response categories (suspect ID, filler ID, rejection) following Smith & Yang (2020). This method distinguishes between investigator discriminability and eyewitness discriminability.

Usage

```
make_fullroc_data(
  data,
  conf_bins = NULL,
  order = c("diagnosticity", "apriori"),
  lineup_size = 6,
  epsilon = 0.001,
  innocent_suspect = "auto"
)
```

Arguments

`data`	A dataframe with the following columns: • target_present: Logical. TRUE if guilty suspect in lineup, FALSE if innocent • identification: Character. "suspect", "filler", or "reject" • confidence: Numeric. Confidence rating (higher = more confident)
`conf_bins`	Numeric vector specifying confidence bin boundaries (default = NULL, which uses unique confidence values). Can specify bins like c(0, 60, 80, 100) to create low (0-60), medium (60-80), and high (80-100) confidence categories.
`order`	Character. Method for ordering response categories: • "diagnosticity": Order by diagnosticity ratio (DR = HR/FAR) (default) • "apriori": Use a-priori ordering (suspect high conf → filler/reject low conf)
`lineup_size`	Integer. Number of people in the lineup (default = 6)
`epsilon`	Numeric. Small value to add when FAR = 0 to avoid infinite DR (default = 0.001)

`innocent_suspect`
: How innocent-suspect identifications in target-absent lineups are obtained. `"designated"`: the lineups had a designated innocent suspect, so target-absent `"suspect"` rows are counted directly, and zero such rows means a false-ID rate of zero. `"estimated"`: there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by `lineup_size`. `"auto"` (the default) chooses `"designated"` when any target-absent row is a suspect ID or the data have a non-missing `suspect_pos` column, and `"estimated"` otherwise, with a message.

Details

This function implements the "full ROC" method described in Smith & Yang (2020). Unlike traditional partial ROC curves that only use suspect identifications, this method uses ALL eyewitness responses:

- Suspect identifications (evidence of guilt)
- Filler identifications (evidence of innocence)
- Lineup rejections (evidence of innocence)

Each response type is crossed with confidence levels to create decision criteria. These are ordered by their diagnosticity ratio (HR/FAR) or by a-priori ordering, then cumulative hit and false alarm rates are computed to form the full ROC curve.

The full ROC provides a threshold-free measure of investigator discriminability— the ability to distinguish guilty from innocent suspects using ALL available eyewitness evidence.

Note on ordering and bias: ordering cells by the *sample* diagnosticity ratio (`order = "diagnosticity"`) bows the curve upward from noise alone, so the resulting AUC is optimistically biased — identical target-present and target-absent response distributions yield AUC slightly above 0.5, with the bias growing with the number of decision-by-confidence cells and shrinking with sample size. For inference or comparisons across conditions, prefer `order = "apriori"` (the theoretically fixed ordering suggested by Smith & Yang, 2020) or compare against a permutation baseline.

Value

A list containing:

- roc_data: Dataframe with cumulative hit rates and false alarm rates
- auc: Full area under the curve
- order_used: The ordering method used
- diagnosticity_table: Table showing non-cumulative rates and DR for each category
- n_target_present: Number of target-present lineups
- n_target_absent: Number of target-absent lineups

References

Smith, A. M., Yang, Y., & Wells, G. L. (2020). Distinguishing between investigator discriminability and eyewitness discriminability: A method for creating full receiver operating characteristic curves of lineup identification performance. *Perspectives on Psychological Science, 15*(3), 589-607.

Examples

```
data(lineup_example)
# Compute full ROC with diagnosticity ordering
fullroc_result <- make_fullroc_data(lineup_example)
print(fullroc_result$auc)

# Compute full ROC with specified confidence bins
fullroc_binned <- make_fullroc_data(lineup_example, conf_bins = c(0, 60, 80, 100))
```

---

make_multid *Multi-d' Model of Lineup Performance (Lee & Penrod, 2019)*

Description

Computes the multiple empirical discriminability (d') indices of Lee and Penrod's (2019) Multi-d' model from target-present (TP) and target-absent (TA) lineup data. Unlike the conventional 2 x 2 signal detection treatment, which folds filler identifications into "negative" responses, the Multi-d' model uses the full 2 x 3 response table (suspect ID, filler ID, rejection) in each lineup type.

Usage

```
make_multid(
  data = NULL,
  counts = NULL,
  lineup_size = 6,
  filler_rate = c("total", "per_filler"),
  correction = c("loglinear", "half", "none"),
  by_confidence = FALSE,
  conf_thresholds = NULL,
  n_bootstrap = 0,
  conf_level = 0.95,
  seed = NULL,
  innocent_suspect = "auto"
)
```

Arguments

data	A data frame with columns `target_present` (logical) and `identification` (`"suspect"`, `"filler"` or `"reject"`). A `confidence` column is needed only when `by_confidence = TRUE`. TA rows with `identification == "suspect"` mark a designated innocent suspect; if there are none, the innocent suspect is estimated from filler choices (see Details). Supply either `data` or `counts`.
counts	A named numeric vector of response frequencies with elements `G` (guilty-suspect IDs), `Fp` (TP filler IDs), `Rp` (TP rejections), `I` (innocent-suspect IDs), `Fa` (TA filler IDs) and `Ra` (TA rejections). Omit `I` (or set it to `NA`) when the TA lineups have no designated innocent suspect; `Fa` is then the total number of TA filler

IDs and is split as described in Details. Proportions may be supplied in place of frequencies, provided `n_bootstrap = 0`.

`lineup_size` Nominal lineup size (suspect plus fillers). Default 6.

`filler_rate` Either `"total"` (default), which uses the total filler ID rate as Lee and Penrod (2019) do, or `"per_filler"`, which divides each filler ID rate by the number of fillers (`lineup_size - 1`) so that the filler indices refer to an average single filler.

`correction` Correction for rates of 0 or 1: `"loglinear"` (default), `"half"` or `"none"`. See `sdt_summary_from_counts`. Use `"none"` to reproduce published values computed from raw proportions.

`by_confidence` Logical. If `TRUE`, the indices are also computed cumulatively by confidence, as in Lee and Penrod's (2019) Table 9. Requires `data` with a `confidence` column.

`conf_thresholds`
Optional numeric vector of confidence thresholds for `by_confidence`. Defaults to every observed confidence value.

`n_bootstrap` Number of bootstrap resamples for percentile confidence intervals (default 0, no bootstrap). Resampling is done within the TP and TA arms separately.

`conf_level` Confidence level for bootstrap intervals. Default 0.95.

`seed` Optional random seed for the bootstrap.

`innocent_suspect`
For trial-level `data`, how innocent-suspect identifications are obtained; see `make_rocdata`. Ignored for `counts`, where the presence of `I` decides.

Details

With each d' computed as the difference between two z-transformed (inverse-normal) response rates, the model defines:

- $d'(GI) = z(G) - z(I)$: guilty versus innocent suspect, the usual empirical discriminability;
- $d'(GFp) = z(G) - z(Fp)$: guilty suspect versus TP fillers;
- $d'(IFa) = z(I) - z(Fa)$: innocent suspect versus TA fillers, which Lee and Penrod treat as TA lineup bias;
- $d'(FaFp) = z(Fa) - z(Fp)$: the greater appeal of TA than TP fillers (differential filler siphoning);
- $d'(RaRp) = z(Ra) - z(Rp)$: discrimination of culprit presence from culprit absence (detection).

Lee and Penrod also propose $d'(GFp) - d'(RaRp)$ as an index of TP lineup bias net of memory strength, and $d'(GFp) - d'(IFa)$ as an index of differential filler siphoning. The decomposition

$$d'(GI) = d'(GFp) - d'(IFa) - d'(FaFp)$$

holds exactly for any data, because the filler terms cancel. It is an accounting identity that shows where d'(GI) comes from, not an empirical result.

When TA lineups have no designated innocent suspect, the innocent-suspect count is estimated as the TA filler count divided by `lineup_size`, and the remaining $(k-1)/k$ of filler choices are kept

as TA filler IDs. Lee and Penrod call this the "average-filler" practice. It forces the TA lineup to be fair, so d'(IFa) then reflects the rejection rate rather than lineup bias.

With `filler_rate = "total"`, a perfectly fair TA lineup does not give $d'(IFa) = 0$: the innocent suspect is compared with all fillers together. The returned `fair_reference` is the d'(IFa) a perfectly fair lineup would give at the observed TA rejection rate (for example, about -1.93 when no one rejects a 6-person lineup), and `ifa_vs_fair` is the observed d'(IFa) minus that reference. These two quantities are not part of Lee and Penrod's model; they are provided to help interpret d'(IFa). With `filler_rate = "per_filler"`, the fair reference is 0, and the average-filler practice gives d'(IFa) = 0 exactly when `correction = "none"` (a rate correction shifts the two rates slightly differently).

The criterion `c` is $-(z(G) + z(I))/2$ and `dr` is the diagnosticity ratio G/I, both as reported by Lee and Penrod (2019).

Value

An object of class `"multid"`, a list with:

indices Data frame of indices with estimates and, if bootstrapped, standard errors and percentile confidence limits.

rates Named vector of the six response rates used (after correction and, if requested, per-filler scaling).

counts Named vector of the six response counts (after any average-filler split).

n_tp, n_ta Number of TP and TA lineups.

innocent_method `"designated"` or `"estimated_from_fillers"`.

by_confidence Data frame of cumulative indices by confidence threshold, or `NULL`.

boot Matrix of bootstrap draws, or `NULL`.

plus the settings used.

References

Lee, J., & Penrod, S. D. (2019). New signal detection theory-based framework for eyewitness performance in lineups. *Law and Human Behavior*. doi:10.1037/lhb0000343

Lee, J., Mansour, J. K., & Penrod, S. D. (2022). Validity of mock-witness measures for assessing lineup fairness. *Psychology, Crime & Law*. doi:10.1080/1068316X.2021.1905811

Lee, J., Mansour, J. K., & Penrod, S. D. (2024). How to measure lineup fairness: Concurrent and predictive validity of lineup-fairness measures. *Psychology, Crime & Law*. doi:10.1080/1068316X.2024.2307358

See Also

`compare_multid`, `plot.multid`, `sdt_summary_from_counts`

Examples

```
# Lineup column of Lee & Penrod's (2019) Table 4 (Steblay et al., 2003)
md <- make_multid(counts = c(G = 45, Fp = 24, Rp = 31, I = 17, Fa = 26, Ra = 57),
                  correction = "none")
md

# From trial-level data, with bootstrap CIs and confidence breakdown
data(lineup_example)
md2 <- make_multid(lineup_example, by_confidence = TRUE,
                   n_bootstrap = 200, seed = 1)
summary(md2)
plot(md2)
```

---

make_ppv_range *Main Function to Compute and Visualize PPV Range*

Description

Convenience wrapper that computes PPV range and creates visualizations.

Usage

```
make_ppv_range(
  data,
  lineup_size = 6,
  confidence_bins = NULL,
  show_plots = TRUE,
  plot_type = "all",
  innocent_suspect = "auto",
  base_rate = NULL
)
```

Arguments

`data`	A dataframe with columns: target_present, identification, confidence
`lineup_size`	Integer. Nominal number of people in lineup (default = 6)
`confidence_bins`	Numeric vector of bin edges (optional)
`show_plots`	Logical. Whether to create plots (default = TRUE)
`plot_type`	Character. Which plots to create: "ppv" (PPV range only), "effective" (effective size only), "all" (all plots, default)
`innocent_suspect`	How innocent-suspect identifications in target-absent lineups are obtained. `"designated"`: the lineups had a designated innocent suspect, so target-absent `"suspect"` rows are counted directly, and zero such rows means a false-ID rate of zero. `"estimated"`:

there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by `lineup_size`. `"auto"` (the default) chooses `"designated"` when any target-absent row is a suspect ID or the data have a non-missing `suspect_pos` column, and `"estimated"` otherwise, with a message.

`base_rate` Prior probability that the suspect is guilty. `NULL` (default) computes PPV from counts, as below. A number strictly between 0 and 1 computes it from the guilty-ID rate (per target-present trial) and the innocent-suspect-ID rate (per target-absent trial) at that prior, so the result does not depend on the TP:TA split.

Value

A list of class "lineup_ppv_range" containing:

- ppv_range_data: Dataframe with all PPV estimates
- plot_ppv_range: ggplot of PPV range (if requested)
- plot_effective_size: ggplot of effective size (if requested)
- plot_error_rate: ggplot of error rates (if requested)
- ... (additional components from ppv_range_by_confidence)

Examples

```
data(lineup_example)
# With confidence binning
result <- make_ppv_range(lineup_example, confidence_bins = c(0, 60, 80, 100))

# Access plots
result$plot_ppv_range
result$plot_effective_size
```

---

make_rac *Compute and Plot RAC for Lineup Identification*

Description

Main function to compute and plot RAC (Response Time-Accuracy Characteristic) for eyewitness lineup data.

Usage

```
make_rac(
  data,
  lineup_size = 6,
  time_bins = NULL,
  show_plot = TRUE,
  innocent_suspect = "auto",
  ...
)
```

Arguments

data	A dataframe with columns: target_present, identification, response_time
lineup_size	Integer. Number of people in lineup (default = 6)
time_bins	Numeric vector of bin edges (recommended for continuous RT data)
show_plot	Logical. Whether to display the plot (default = TRUE)
innocent_suspect	How innocent-suspect identifications in target-absent lineups are obtained. "designated": the lineups had a designated innocent suspect, so target-absent "suspect" rows are counted directly, and zero such rows means a false-ID rate of zero. "estimated": there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by lineup_size. "auto" (the default) chooses "designated" when any target-absent row is a suspect ID or the data have a non-missing suspect_pos column, and "estimated" otherwise, with a message.
...	Additional arguments passed to make_rac_gg()

Value

A list containing RAC data and plot

Examples

```
# Example with binned response times (in milliseconds)
lineup_data <- create_example_lineup_data(n_trials = 200,
                                          include_response_time = TRUE,
                                          seed = 123)
rac_result <- make_rac(lineup_data,
                       time_bins = c(0, 5000, 10000, 15000, 20000))
print(rac_result)
rac_result$plot
```

---

make_rac_gg *Plot RAC Curve*

Description

Creates a ggplot2 visualization of RAC data showing the relationship between response time and accuracy.

Usage

```
make_rac_gg(
  racobj_list,
  show_errorbars = TRUE,
  show_n = TRUE,
  time_units = "ms"
)
```

Arguments

racobj_list	List output from make_racdata()
show_errorbars	Logical. Whether to show error bars (default = TRUE)
show_n	Logical. Whether to show sample sizes (default = TRUE)
time_units	Character. Label for time units (e.g., "ms", "seconds"). Default = "ms"

Value

A ggplot2 object

Examples

```
data <- create_example_lineup_data(n_trials = 200,
                                   include_response_time = TRUE,
                                   seed = 123)
rac <- make_racdata(data, time_bins = c(0, 4000, 8000, 12000, 20000))
make_rac_gg(rac)
```

---

make_racdata *Compute RAC (Response Time-Accuracy Characteristic) Data*

Description

Computes response time-accuracy data for lineup identifications following the approach described in Seale-Carlisle et al. (2019) and implemented in pyWitness (Mickes et al., 2024). RAC analysis shows the relationship between response time and accuracy for suspect identifications.

Usage

```
make_racdata(
  data,
  lineup_size = 6,
  time_bins = NULL,
  innocent_suspect = "auto"
)
```

Arguments

data	A dataframe with the following columns: • target_present: Logical. TRUE if guilty suspect in lineup • identification: Character. "suspect", "filler", or "reject" • response_time: Numeric. Response time in milliseconds (or seconds)
lineup_size	Integer. Number of people in lineup (default = 6)

`time_bins` Numeric vector of bin edges for grouping response times (e.g., c(0, 5000, 10000, 15000, 20000) creates bins in milliseconds). If NULL, uses individual response time values (not recommended for continuous data).

`innocent_suspect` How innocent-suspect identifications in target-absent lineups are obtained. `"designated"`: the lineups had a designated innocent suspect, so target-absent `"suspect"` rows are counted directly, and zero such rows means a false-ID rate of zero. `"estimated"`: there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by `lineup_size`. `"auto"` (the default) chooses `"designated"` when any target-absent row is a suspect ID or the data have a non-missing `suspect_pos` column, and `"estimated"` otherwise, with a message.

Details

RAC analysis computes, for each response time bin:

$$Accuracy = \frac{CorrectSuspectIDs}{CorrectSuspectIDs + IncorrectSuspectIDs}$$

Explicit target-absent suspect IDs are used when present. Otherwise, target-absent filler IDs are divided by lineup size to estimate incorrect suspect IDs. The two estimators are never added together. The bin mean time uses the same suspect-ID observations as accuracy; estimated-mode TA filler times receive weight $1/k$. Bin edges must cover all supplied observations. Count accuracy depends on TP/TA sampling fractions and does not adjust to a different population base rate. The displayed binomial SE is approximate, especially for estimated counts.

RAC analysis is useful for examining the speed-accuracy tradeoff in eyewitness identifications. Faster responses typically indicate stronger memory, and RAC curves often show higher accuracy for faster response times.

Value

A list containing:

- rac_data: Dataframe with response time bins and accuracy
- n_ids_per_bin: Number of suspect IDs in each time bin
- overall_accuracy: Overall accuracy across all suspect IDs

References

Seale-Carlisle, T. M., Colloff, M. F., Flowe, H. D., Wells, W., Wixted, J. T., & Mickes, L. (2019). Confidence and response time as indicators of eyewitness identification accuracy in the lab and in the real world. *Journal of Applied Research in Memory and Cognition, 8*(4), 420-428.

Mickes, L., Seale-Carlisle, T. M., Chen, X., & Boogert, S. (2024). pyWitness 1.0: A python eyewitness identification analysis toolkit. *Behavior Research Methods, 56*, 1533-1550.

Examples

```
data <- create_example_lineup_data(n_trials = 200,
                                   include_response_time = TRUE,
                                   seed = 123)
rac <- make_racdata(data, time_bins = c(0, 4000, 8000, 12000, 20000))
rac$rac_data
```

`make_roc` *Compute and Plot ROC Curve for Lineup Identification*

Description

Main function to compute and plot an ROC curve for eyewitness lineup data. This follows the methodology of Wixted & Mickes (2012) and Mickes (2015).

Usage

```
make_roc(
  data,
  lineup_size = 6,
  show_plot = TRUE,
  innocent_suspect = "auto",
  ...
)
```

Arguments

`data`	A dataframe with the following columns: • target_present: Logical. TRUE if guilty suspect in lineup • identification: Character. "suspect", "filler", or "reject" • confidence: Numeric. Confidence rating
`lineup_size`	Integer. Number of people in lineup (default = 6)
`show_plot`	Logical. Whether to display the plot (default = TRUE)
`innocent_suspect`	How innocent-suspect identifications in target-absent lineups are obtained. `"designated"`: the lineups had a designated innocent suspect, so target-absent `"suspect"` rows are counted directly, and zero such rows means a false-ID rate of zero. `"estimated"`: there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by `lineup_size`. `"auto"` (the default) chooses `"designated"` when any target-absent row is a suspect ID or the data have a non-missing `suspect_pos` column, and `"estimated"` otherwise, with a message.
`...`	Additional arguments passed to make_roc_gg()

Details

ROC analysis measures discriminability - the ability to distinguish innocent from guilty suspects. According to Mickes (2015), ROC analysis is most relevant for policymakers deciding on system variables (e.g., simultaneous vs. sequential lineups, lineup size, etc.).

For estimator variables (e.g., exposure duration, retention interval), use CAC analysis instead (see `make_cac`).

Value

A list containing:

- plot: ggplot2 object. Always returned, whatever show_plot is set to; show_plot controls only whether it is displayed immediately
- roc_data: Dataframe with ROC curve points
- pauc: Partial area under the curve
- summary: Summary statistics

References

Wixted, J. T., & Mickes, L. (2012). The field of eyewitness memory should abandon probative value and embrace receiver operating characteristic analysis. *Perspectives on Psychological Science, 7*(3), 275-278.

Mickes, L. (2015). Receiver operating characteristic analysis and confidence-accuracy characteristic analysis in investigations of system variables and estimator variables that affect eyewitness memory. *Journal of Applied Research in Memory and Cognition, 4*(2), 93-102.

Examples

```
data(lineup_example)
roc_result <- make_roc(lineup_example)
print(roc_result$pauc)
```

---

make_roc_gg *Plot ROC Curve for Lineup Identification*

Description

Creates a ggplot2 visualization of ROC data from lineup identification experiments.

Usage

```
make_roc_gg(
  rocobj_list,
  show_pauc = TRUE,
  point_labels = TRUE,
  title = "Partial ROC Curve (suspect IDs only)",
  data_source = NULL
)
```

Arguments

rocobj_list	List output from make_rocdata(), or the lineup_roc object from make_roc()
show_pauc	Logical. Whether to display pAUC value on plot (default = TRUE)
point_labels	Logical. Whether to label points with confidence levels (default = TRUE)
title	Character. Plot title. The default names the ROC variant, so that a partial ROC is not mistaken for a full ROC (see plot_fullroc).
data_source	Character. Name of the dataset the curve was computed from, appended to the caption. Defaults to the "data_source" attribute recorded by make_rocdata, and is omitted when unavailable.

Value

A ggplot2 object. Being an ordinary ggplot, it can be modified with the usual + syntax, e.g. p + ggplot2::theme_minimal().

Examples

```
data(lineup_example)
roc <- make_rocdata(lineup_example)
make_roc_gg(roc)
```

---

make_rocdata *Compute ROC Data for Lineup Identification*

Description

Computes Receiver Operating Characteristic (ROC) data from lineup identification experiments following the methodology of Mickes, Wixted, and Gronlund.

Usage

```
make_rocdata(
  data,
  lineup_size = 6,
  innocent_suspect = "auto",
  thresholds = NULL
)
```

Arguments

`data`	A dataframe with the following columns: • target_present: Logical. TRUE if guilty suspect in lineup, FALSE if innocent • identification: Character. "suspect", "filler", or "reject" • confidence: Numeric. Confidence rating (higher = more confident)
`lineup_size`	Integer. Number of people in the lineup (default = 6). Used to estimate false ID rate when fillers are chosen from target-absent lineups.
`innocent_suspect`	How innocent-suspect identifications in target-absent lineups are obtained. `"designated"`: the lineups had a designated innocent suspect, so target-absent `"suspect"` rows are counted directly, and zero such rows means a false-ID rate of zero. `"estimated"`: there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by `lineup_size`. `"auto"` (the default) chooses `"designated"` when any target-absent row is a suspect ID or the data have a non-missing `suspect_pos` column, and `"estimated"` otherwise, with a message.
`thresholds`	Optional numeric vector of confidence criteria at which to evaluate the ROC. The default uses the confidence levels of the responses that can move an operating point; see Details. Supplying the criteria of a full sample keeps them fixed across bootstrap resamples.

Details

This function computes ROC curves following Wixted & Mickes (2012) and Mickes (2015). For each confidence criterion:

- Correct ID rate = proportion of target-present lineups where suspect identified
- False ID rate = proportion of target-absent lineups where suspect identified

The criteria are the confidence levels of the responses that enter the ROC: target-present suspect IDs, and target-absent suspect IDs (designated suspect) or filler IDs (estimated). The confidence given to rejections, or to filler IDs in target-present lineups, does not define a criterion, because it cannot move an operating point. Each criterion therefore gives a distinct point, and averages over points (DPP, z-ROC d') do not depend on how rejection confidence was coded.

With a designated innocent suspect, target-absent `"suspect"` responses are used directly. Without one, filler IDs are divided by lineup size. The two estimators are never added together. See `innocent_suspect` for how the design is determined.

Value

A list containing:

- roc_data: Dataframe with correct_id_rate, false_id_rate, and confidence
- pauc: Partial area under the curve
- n_target_present: Number of target-present lineups
- n_target_absent: Number of target-absent lineups
- thresholds: The confidence criteria used, highest first

References

Wixted, J. T., & Mickes, L. (2012). The field of eyewitness memory should abandon probative value and embrace receiver operating characteristic analysis. *Perspectives on Psychological Science, 7*(3), 275-278.

Mickes, L. (2015). Receiver operating characteristic analysis and confidence-accuracy characteristic analysis in investigations of system variables and estimator variables that affect eyewitness memory. *Journal of Applied Research in Memory and Cognition, 4*(2), 93-102.

Gronlund, S. D., Wixted, J. T., & Mickes, L. (2014). Evaluating eyewitness identification procedures using receiver operating characteristic analysis. *Current Directions in Psychological Science, 23*(1), 3-10.

Examples

```
data(lineup_example)
roc <- make_rocdata(lineup_example)
roc$roc_data
roc$pauc
```

`make_utility_curves` *Compute Expected Utility Curves for Lineup Identification*

Description

Computes expected utility at different confidence criteria following Lampinen, Smith, & Wells (2019). Utility analysis accounts for the costs and benefits of different identification outcomes.

Usage

```
make_utility_curves(
  data,
  base_rate = 0.5,
  utility_matrix = c(tp = 1, fn = -0.5, fp = -2, tn = 0.5),
  lineup_size = 6,
  criteria = c("confidence", "all"),
  innocent_suspect = "auto",
  thresholds = NULL
)
```

Arguments

`data`	A dataframe with the following columns: • target_present: Logical. TRUE if guilty suspect in lineup • identification: Character. "suspect", "filler", or "reject" • confidence: Numeric. Confidence rating (higher = more confident)
`base_rate`	Numeric. Prior probability that suspect is guilty (0 to 1). Default = 0.5.

`utility_matrix` Named numeric vector with utilities/costs for:

- tp: True positive (correct suspect ID) - typically positive
- fn: False negative (miss/rejection of guilty) - typically negative
- fp: False positive (innocent suspect ID) - typically very negative
- tn: True negative (correct rejection of innocent) - typically positive

Default: c(tp=1, fn=-0.5, fp=-2, tn=0.5)

`lineup_size` Integer. Number of people in lineup (default = 6). Used to estimate false IDs from filler choices.

`criteria` Character. How to define decision criteria:

- "confidence": Use confidence thresholds (cumulative from high to low)
- "all": Include all identifications regardless of confidence

Default = "confidence".

`innocent_suspect` How innocent-suspect identifications in target-absent lineups are obtained. `"designated"`: the lineups had a designated innocent suspect, so target-absent `"suspect"` rows are counted directly, and zero such rows means a false-ID rate of zero. `"estimated"`: there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by `lineup_size`. `"auto"` (the default) chooses `"designated"` when any target-absent row is a suspect ID or the data have a non-missing `suspect_pos` column, and `"estimated"` otherwise, with a message.

`thresholds` Optional numeric vector of confidence criteria. The default uses the confidence levels of the responses that can change the hit or false-alarm rate, as in `make_rocdata`, so the confidence given to rejections does not add criteria or change the average utility.

Details

Expected utility combines hit rates and false alarm rates with the costs/benefits of different outcomes:

$$EU = base_rate \cdot [hit \cdot U_{TP} + (1 - hit) \cdot U_{FN}] +$$
$$(1 - base_rate) \cdot [fa \cdot U_{FP} + (1 - fa) \cdot U_{TN}]$$

Where:

- hit = p(suspect ID | guilty) at criterion
- fa = p(suspect ID | innocent) at criterion
- U_TP, U_FN, U_FP, U_TN = utilities for each outcome

The utility matrix should reflect the relative value/cost of outcomes. For example:

- tp = 1: Correctly identifying the perpetrator
- fn = -0.5: Missing the perpetrator (they remain at large)
- fp = -2: Wrongly convicting an innocent person (severe injustice)

- tn = 0.5: Correctly rejecting when innocent

Lampinen et al. (2019) show that comparing procedures by ROC curves alone can be misleading. Utility analysis incorporates base rates and the relative costs of errors, providing a more complete evaluation.

Note on the false-alarm estimate: explicit target-absent suspect IDs are used when present. Otherwise, filler IDs are divided by lineup size. Direct and filler-derived estimates are never added together.

Value

A list containing:

- utility_data: Dataframe with criterion, strategy, hit rate, false alarm rate, and expected utility. Each row is the rule "accept suspect IDs with confidence >= criterion". The `"reject_all"` row has a criterion above the observed identification confidences and supplied grid (possibly `Inf`), so that applying it accepts no identification.
- max_utility: Maximum expected utility, with the corresponding criterion and strategy
- avg_utility: Average expected utility across all criteria
- utility_all_ids: Expected utility if all IDs are accepted
- base_rate: Base rate used
- utility_matrix: Utility matrix used

References

Lampinen, J. M., Smith, A. M., & Wells, G. L. (2019). Four utilities in eyewitness identification practice: Dissociations between receiver operating characteristic analysis and expected utility analysis. *Law and Human Behavior, 43*(1), 26-44.

Examples

```
data(lineup_example)
util <- make_utility_curves(lineup_example)
util$max_utility
```

`make_utility_difference` *Compute Utility Difference Curves Comparing Two Procedures*

Description

Computes the difference in expected utility between two lineup procedures across a range of base rates and/or utility matrices, following Lampinen, Smith, & Wells (2019).

Usage

```
make_utility_difference(
  data_proc_a,
  data_proc_b,
  base_rate_grid = seq(0.01, 0.99, 0.01),
  utility_matrix = c(tp = 1, fn = -0.5, fp = -2, tn = 0.5),
  utility_type = c("max", "avg", "all"),
  lineup_size = 6,
  innocent_suspect = "auto"
)
```

Arguments

`data_proc_a`	A dataframe for procedure A (standard lineup format)
`data_proc_b`	A dataframe for procedure B (standard lineup format)
`base_rate_grid`	Numeric vector of base rates to evaluate. Default: seq(0.01, 0.99, 0.01)
`utility_matrix`	Named numeric vector with utilities (tp, fn, fp, tn). Default: c(tp=1, fn=-0.5, fp=-2, tn=0.5)
`utility_type`	Character. Which utility to compare: • "max": Maximum utility (optimal criterion) • "avg": Average utility across all criteria • "all": Utility of accepting all identifications Default = "max".
`lineup_size`	Integer. Lineup size (default = 6)
`innocent_suspect`	How innocent-suspect identifications in target-absent lineups are obtained. `"designated"`: the lineups had a designated innocent suspect, so target-absent `"suspect"` rows are counted directly, and zero such rows means a false-ID rate of zero. `"estimated"`: there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by `lineup_size`. `"auto"` (the default) chooses `"designated"` when any target-absent row is a suspect ID or the data have a non-missing `suspect_pos` column, and `"estimated"` otherwise, with a message.

Details

Utility difference curves show how the relative value of two procedures changes with the base rate (prevalence) of guilty suspects. This addresses the question: "Which procedure is better, and does it depend on the base rate?"

$$UtilityDifference = EU_A - EU_B$$

Interpretation:

- Positive difference: Procedure A has higher utility (preferred)
- Negative difference: Procedure B has higher utility (preferred)

- Zero difference: Procedures equivalent at that base rate

Lampinen et al. (2019) show that ROC dominance does not guarantee utility dominance. A procedure with a lower pAUC might have higher utility at certain base rates and cost structures.

Value

A list containing:

- difference_curve: Dataframe with base_rate, eu_a, eu_b, and difference
- utility_type: Type of utility compared
- utility_matrix: Utility matrix used
- crossover_points: Base rates where procedures have equal utility (if any)

References

Lampinen, J. M., Smith, A. M., & Wells, G. L. (2019). Four utilities in eyewitness identification practice: Dissociations between receiver operating characteristic analysis and expected utility analysis. *Law and Human Behavior, 43*(1), 26-44.

Examples

```
data(lineup_example)
odd <- seq(1, nrow(lineup_example), by = 2)
util_diff <- make_utility_difference(lineup_example[odd, ],
                                     lineup_example[-odd, ],
                                     base_rate_grid = seq(0.1, 0.9, 0.1))
util_diff$difference_curve
```

---

makevec_prop *Helper functions*

Description

Several helper functions that compute bootstrap confidence limits from proportions.

Usage

```
makevec_prop(prop, n)
```

Arguments

prop	A single finite proportion between 0 and 1.
n	A positive whole-number sample size. `n * prop` must be a whole number.

Examples

```
# Expand a choice proportion of 0.25 among 20 mock witnesses into a
# binary choice vector
makevec_prop(0.25, 20)
```

`max_filler_moments` *Max Filler Distribution Moments*

Description

Numeric moments of the max filler distribution using integration.

Usage

```
max_filler_moments(
  lineup_size,
  moments = c("mean", "var", "skewness"),
  rel.tol = 1e-08,
  subdivisions = 2000
)
```

Arguments

`lineup_size`	Lineup size (suspect + fillers), must be >= 2.
`moments`	Character vector specifying moments to return. Supported: "mean", "var", "skewness".
`rel.tol`	Relative tolerance passed to integrate().
`subdivisions`	Maximum subdivisions for integrate().

Value

Named list of moments.

Examples

```
max_filler_moments(6)
```

mickwick *Confidence & Accuracy data (Mickes & Wixted)*

Description

Data capturing confidence and accuracy for a lineup trial

Usage

```
data(mickwick)
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 100 rows and 2 columns.

Details

mickwick

mockdata *mockdata*

Description

A dataframe containing real lineup data.

3 columns x 94 observations.

The first column simply provides the participant number, whilst the second and third contain data for each participant's choice (i.e., the position of the lineup member that they identified) and a confidence score for each choice.

In the field of eyewitness testimony, it is common for participants to rate how confident they are that they have accurately identified the perpetrator. Here, confidence is recorded as a percentage.

This lineup has the following parameters:

- 94 mock witness choices
- Target position is 7
- Nominal size is 9

Usage

```
data(mockdata)
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 94 rows and 3 columns.

nortje2012 *nortje2012*

Description

A data file containing lineup data for 3 independent lineups. 133 rows x 3 columns. Each lineup trial consisted of 133 mock witnesses & 8 lineup members.

- : Target position for lineup 1 is 3
- : Target position for lineup 2 is 6
- : Target position for lineup 3 is 2

Usage

```
data(nortje2012)
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 133 rows and 3 columns.

plot.esize_T_bayes *Plot method for esize_T_bayes objects*

Description

Plots the posterior distribution of Tredoux's E' as a histogram overlaid with a kernel density estimate. Vertical lines mark the posterior median and credible interval bounds.

Usage

```
## S3 method for class 'esize_T_bayes'
plot(x, ...)
```

Arguments

`x`	An object of class `"esize_T_bayes"`.
`...`	Further arguments (unused).

Value

A ggplot object.

plot.esize_T_bayes_compare

Plot method for esize_T_bayes_compare objects

Description

Plots the posterior distribution of Delta = E'_A - E'_B as a histogram with density overlay. A vertical line at zero and shaded probability mass above/below zero are shown.

Usage

```
## S3 method for class 'esize_T_bayes_compare'
plot(x, ...)
```

Arguments

x	An object of class "esize_T_bayes_compare".
...	Further arguments (unused).

Value

A ggplot object.

plot.lineup_model_comparison

Plot a Wixted Lineup-Memory Model Comparison

Description

Plot a Wixted Lineup-Memory Model Comparison

Usage

```
## S3 method for class 'lineup_model_comparison'
plot(x, type = c("comparison", "fit", "roc"), ...)
```

Arguments

x	A "lineup_model_comparison" object.
type	Plot type: Pearson comparison, fitted response cells, or ROC.
...	Additional arguments (currently ignored).

Value

A ggplot object.

plot.model_comparison *Plot Side-by-Side Model Comparisons*

Description

Creates a multi-panel visualization comparing fitted models.

Usage

```
## S3 method for class 'model_comparison'
plot(x, which = "all", ncol = 2, ...)
```

Arguments

x	A model_comparison object from compare_models()
which	Character vector specifying which plots to create. Options: "2ht", "eig_ig", "eig_posteriors", "fullroc" Default = "all" creates all available plots.
ncol	Integer. Number of columns for plot layout (default = 2)
...	Additional arguments passed to individual plotting functions

Details

This function creates side-by-side visualizations of all fitted models for easy comparison. The specific plots depend on which models were fit:

- **2-HT**: Parameter estimates with confidence intervals
- **EIG**: Information gain by response category
- **Full ROC**: ROC curve with AUC

Value

A combined ggplot object (via patchwork or cowplot)

plot.multid *Plot a Multi-d' Analysis*

Description

Plot a Multi-d' Analysis

Usage

```
## S3 method for class 'multid'
plot(x, type = c("decomposition", "indices", "confidence"), ...)
```

Arguments

x	An object of class "multid" from make_multid.
type	"decomposition" (default) shows d'(GFp), -d'(IFa) and -d'(FaFp) as a waterfall summing to d'(GI). "indices" plots each index, with bootstrap intervals when available. "confidence" plots the cumulative indices against confidence threshold (requires by_confidence = TRUE).
...	Unused.

Value

A ggplot object.

Examples

```
md <- make_multid(counts = c(G = 45, Fp = 24, Rp = 31, I = 17, Fa = 26, Ra = 57))
plot(md)
plot(md, type = "indices")
```

---

plot.pauc_comparison *Plot pAUC Comparison*

Description

Creates a side-by-side visualization of ROC curves for two conditions with shaded pAUC regions and statistical test results.

Usage

```
## S3 method for class 'pauc_comparison'
plot(x, show_cutoff = TRUE, show_test_results = TRUE, ...)
```

Arguments

x	A pauc_comparison object from compare_pauc()
show_cutoff	Logical. Whether to show false ID rate cutoff line (default = TRUE)
show_test_results	Logical. Whether to show test results on plot (default = TRUE)
...	Additional arguments (ignored)

Details

Creates a comparison plot showing:

- ROC curves for both conditions
- Shaded pAUC regions (up to cutoff)
- Vertical line at false ID rate cutoff
- Test statistics (Z-score, p-value)

Value

A ggplot2 object

`plot.power_analysis` *Plot Power Analysis Results*

Description

Plot Power Analysis Results

Usage

```
## S3 method for class 'power_analysis'
plot(x, ...)
```

Arguments

`x`	A power_analysis object
`...`	Additional arguments (ignored)

Value

A ggplot object, returned invisibly. Because it is an ordinary ggplot, further layers and themes can be added to it.

`plot.sdt_roc_fit` *Plot Method for SDT ROC Fit*

Description

Plot Method for SDT ROC Fit

Usage

```
## S3 method for class 'sdt_roc_fit'
plot(x, ...)
```

Arguments

`x`	An sdt_roc_fit object
`...`	Additional arguments passed to plot

plot.winter_2ht *Plot Method for winter_2ht Objects*

Description

Creates a default plot for winter_2ht objects showing parameter estimates.

Usage

```
## S3 method for class 'winter_2ht'
plot(x, ...)
```

Arguments

x	A winter_2ht object
...	Additional arguments passed to plot_2ht_parameters()

plot.winter_2ht_boot *Plot Bootstrap Distributions*

Description

Creates histograms showing the bootstrap distributions of parameter estimates.

Usage

```
## S3 method for class 'winter_2ht_boot'
plot(x, ...)
```

Arguments

x	A winter_2ht_boot object
...	Additional arguments (not used)

plot_2ht_fit *Plot Observed vs. Expected Counts*

Description

Creates a comparison plot showing observed counts vs. model-predicted expected counts for both target-present and target-absent lineups.

Usage

```
plot_2ht_fit(x, ...)
```

Arguments

x	A winter_2ht object from fit_winter_2ht()
...	Additional arguments (not used)

Value

A ggplot object

Examples

```
counts <- c(
  n_tp_suspect = 147, n_tp_filler = 94, n_tp_reject = 141,
  n_ta_suspect = 38, n_ta_filler = 138, n_ta_reject = 206
)
fit <- fit_winter_2ht(counts, lineup_size = 6)
plot_2ht_fit(fit)
```

plot_2ht_parameters *Plot Parameter Estimates from 2-HT Model*

Description

Creates a visual display of the parameter estimates from a fitted 2-HT model, with error bars showing the supplied bootstrap intervals, or regular 95% normal intervals where available.

Usage

```
plot_2ht_parameters(x, ..., which = c("dP", "dA", "b", "g"), show_ci = TRUE)
```

Arguments

x	A winter_2ht object, or a winter_2ht_boot object from boot_winter_2ht. Given the latter, the plotted intervals are the bootstrap percentile intervals; given the former they are normal approximations clamped to [0, 1]. from fit_winter_2ht()
...	Additional arguments passed to plotting functions
which	Character vector specifying which parameters to plot. Default is c("dP", "dA", "b", "g") (all parameters).
show_ci	Logical indicating whether to show 95% confidence intervals. Default is TRUE.

Value

A ggplot object

Examples

```
counts <- c(
  n_tp_suspect = 147, n_tp_filler = 94, n_tp_reject = 141,
  n_ta_suspect = 38, n_ta_filler = 138, n_ta_reject = 206
)
fit <- fit_winter_2ht(counts, lineup_size = 6)
plot_2ht_parameters(fit)
```

---

plot_anri_comparison *Plot ANRI Comparison Between Groups*

Description

Creates a visualization comparing ANRI estimates between two groups with bootstrap confidence intervals.

Usage

```
plot_anri_comparison(compare_obj)
```

Arguments

compare_obj	List output from compare_anri()

Details

Creates a point-and-interval plot showing ANRI estimates and bootstrap CIs for each group. Helps visualize the magnitude and uncertainty of group differences.

Value

A ggplot2 object

Examples

```
set.seed(123)
n <- 300
data <- data.frame(
  target_present = rep(c(TRUE, FALSE), each = n / 2),
  identification = sample(c("suspect", "filler", "reject"), n,
                          replace = TRUE, prob = c(0.5, 0.25, 0.25)),
  confidence = round(runif(n, 0, 100)),
  condition = rep(c("sequential", "simultaneous"), times = n / 2)
)
cmp <- compare_anri(data, group_var = "condition",
                    confidence_bins = seq(0, 100, 20),
                    n_bootstrap = 100, seed = 1)
plot_anri_comparison(cmp)
```

---

plot_anri_difference_distribution

Plot Bootstrap Distribution of ANRI Difference

Description

Creates a histogram of the bootstrap distribution of the difference between two groups.

Usage

```
plot_anri_difference_distribution(compare_obj)
```

Arguments

compare_obj	List output from compare_anri()

Value

A ggplot2 object

Examples

```
set.seed(123)
n <- 300
data <- data.frame(
  target_present = rep(c(TRUE, FALSE), each = n / 2),
  identification = sample(c("suspect", "filler", "reject"), n,
                          replace = TRUE, prob = c(0.5, 0.25, 0.25)),
  confidence = round(runif(n, 0, 100)),
  condition = rep(c("sequential", "simultaneous"), times = n / 2)
)
cmp <- compare_anri(data, group_var = "condition",
```

```
                    confidence_bins = seq(0, 100, 20),
                    n_bootstrap = 100, seed = 1)
plot_anri_difference_distribution(cmp)
```

`plot_bayes_information_gain` *Plot Information Gain Curves*

Description

Creates a plot showing how much uncertainty is reduced by different lineup responses.

Usage

```
plot_bayes_information_gain(bayes_obj, selected_responses = NULL)
```

Arguments

`bayes_obj`	List output from make_bayes_curves()
`selected_responses`	Character vector of response categories to plot. If NULL (default), plots all responses.

Details

Information gain curves show the reduction in uncertainty (measured in bits) for each response type across different prior probabilities. A response that points to guilt gains most at priors above 0.5 and one that points to innocence at priors below 0.5; either can be negative where it moves belief towards 0.5. The legend lists responses from the most to the least indicative of guilt.

Value

A ggplot2 object

Examples

```
data(lineup_example)
bayes <- make_bayes_curves(lineup_example)
plot_bayes_information_gain(bayes)
```

plot_bayes_prior_posterior *Plot Prior-Posterior Curves*

Description

Creates a plot showing how different lineup responses update prior beliefs about suspect guilt to posterior beliefs.

Usage

```
plot_bayes_prior_posterior(
  bayes_obj,
  selected_responses = NULL,
  show_diagonal = TRUE
)
```

Arguments

bayes_obj	List output from make_bayes_curves()
selected_responses	Character vector of response categories to plot. If NULL (default), plots all responses.
show_diagonal	Logical. Whether to show diagonal line (no update). Default = TRUE.

Details

Prior-posterior curves show Bayesian updating for different responses. The diagonal line represents no update (posterior = prior). Curves above the diagonal indicate responses that increase belief in guilt, while curves below decrease belief in guilt.

Value

A ggplot2 object

Examples

```
data(lineup_example)
bayes <- make_bayes_curves(lineup_example)
plot_bayes_prior_posterior(bayes)
```

plot_bree *Plot BREE Curve*

Description

Creates a plot showing the base-rate shift required for one procedure to match another's diagnostic value.

Usage

```
plot_bree(bree_obj, show_reference = TRUE)
```

Arguments

bree_obj	List output from make_bree_curve()
show_reference	Logical. Whether to show reference line at delta = 0. Default = TRUE.

Details

The BREE curve shows how much the base rate (prior probability) would need to shift for Procedure B to yield the same posterior as Procedure A.

Positive delta: Procedure A is more diagnostic (B needs higher base rate to match) Negative delta: Procedure B is more diagnostic (B needs lower base rate to match) Zero delta: Procedures are equally diagnostic

Value

A ggplot2 object

Examples

```
data(lineup_example)
odd <- seq(1, nrow(lineup_example), by = 2)
bree <- make_bree_curve(lineup_example[odd, ], lineup_example[-odd, ])
plot_bree(bree)
```

plot_dpp *Plot DPP Across Operating Points*

Description

Plots DPP at each operating point of the ROC against the confidence threshold that defines it, with the average marked.

Usage

```
plot_dpp(dpp_obj, show_mean = TRUE, show_chance = TRUE, show_dpp = TRUE)
```

Arguments

dpp_obj	List output from make_dpp() or compute_dpp()
show_mean	Logical. Draw the average DPP as a dashed line (default = TRUE)
show_chance	Logical. Draw the chance reference at DPP = 1, which applies to the unweighted measure only (default = TRUE)
show_dpp	Logical. Display the average DPP value on the plot (default = TRUE)

Details

Points are ordered from the strictest threshold (highest confidence, left) to the most lenient (right), as in Smith et al.'s (2019) Table 4. The operating point with the lowest DPP, the point of maximum utility, is highlighted.

Value

A ggplot2 object

Examples

```
data(lineup_example)
dpp <- make_dpp(lineup_example)
plot_dpp(dpp)
```

plot_dpp_comparison *Plot DPP Comparison Between Two Procedures*

Description

Overlays DPP at each operating point for two procedures.

Usage

```
plot_dpp_comparison(
  compare_obj,
  proc_a_label = "Procedure A",
  proc_b_label = "Procedure B"
)
```

Arguments

compare_obj	List output from compare_dpp()
proc_a_label	Character. Label for procedure A (default = "Procedure A")
proc_b_label	Character. Label for procedure B (default = "Procedure B")

Value

A ggplot2 object

Examples

```
data(lineup_example)
odd <- seq(1, nrow(lineup_example), by = 2)
cmp <- compare_dpp(lineup_example[odd, ], lineup_example[-odd, ],
                   n_bootstrap = 0)
plot_dpp_comparison(cmp)
```

plot_effective_size_conf *Plot Effective Size by Confidence Level*

Description

Creates a ggplot visualization showing how effective lineup size varies across confidence levels.

Usage

```
plot_effective_size_conf(
  ppv_range_obj,
  show_nominal = TRUE,
  show_points = TRUE
)
```

Arguments

ppv_range_obj	A lineup_ppv_range object from ppv_range_by_confidence(), or a dataframe with columns: confidence, effective_size
show_nominal	Logical. Whether to show nominal lineup size as reference line (default = TRUE)
show_points	Logical. Whether to show points at each confidence level (default = TRUE)

Details

Effective size < nominal size indicates lineup bias, where some members are implausible and rarely chosen. This can vary by confidence level if witnesses at different confidence levels have different lineup viewing strategies.

A horizontal line at the nominal lineup size provides a reference. Values below this line indicate bias; values at or above indicate fair lineups.

Value

A ggplot2 object

Examples

```
data(lineup_example)
ppv_range <- ppv_range_by_confidence(lineup_example,
                                     confidence_bins = c(0, 60, 80, 100))
plot_effective_size_conf(ppv_range)
```

---

plot_eig *Plot Information Gain by Response Category*

Description

Creates a ggplot2 visualization showing information gain for each response category in an EIG analysis.

Usage

```
plot_eig(eig_obj, max_responses = 15, color_by = "identification")
```

Arguments

eig_obj	A lineup_eig object from compute_eig()
max_responses	Maximum number of response categories to display (default = 15). Categories are sorted by information gain (highest first).
color_by	Character. Color bars by "identification" decision or "ig" (information gain). Default = "identification".

Details

This function creates a bar plot showing information gain for each response category. Response categories are ordered by IG (descending), with the most informative responses shown first. Colors distinguish different identification decisions (suspect/filler/reject).

Value

A ggplot2 object

Examples

```
data(lineup_example)
eig_result <- compute_eig(lineup_example, confidence_bins = c(0, 60, 80, 100))
plot_eig(eig_result)
```

---

plot_eig_posteriors *Plot Posterior Probabilities for Response Categories*

Description

Creates a ggplot2 visualization showing how each response category updates beliefs about guilt (posterior probabilities).

Usage

```
plot_eig_posteriors(eig_obj, max_responses = 15, show_prior = TRUE)
```

Arguments

eig_obj	A lineup_eig object from compute_eig()
max_responses	Maximum number of response categories to display (default = 15)
show_prior	Logical. Whether to show prior probability line (default = TRUE)

Details

This function creates a bar plot showing the posterior probability of guilt for each response category. The prior probability is shown as a dashed line for reference. Response categories that push beliefs toward guilt (posterior > prior) are colored red, while those pushing toward innocence are colored blue.

Value

A ggplot2 object

Examples

```
data(lineup_example)
eig_result <- compute_eig(lineup_example, confidence_bins = c(0, 60, 80, 100))
plot_eig_posteriors(eig_result)
```

`plot_embedding_space` *Plot Embedding Space*

Description

Creates a 2D visualization of face embeddings using dimensionality reduction, showing how faces cluster in the embedding space.

Usage

```
plot_embedding_space(
  embeddings_df,
  method = c("umap", "pca", "tsne"),
  labels = NULL,
  highlight = NULL,
  n_neighbors = 15,
  perplexity = 30
)
```

Arguments

`embeddings_df`	A tibble from `batch_embeddings` with an 'embedding' list column.
`method`	Dimensionality reduction method: "umap" (default), "pca", or "tsne".
`labels`	Optional character vector of labels for each point.
`highlight`	Optional indices of points to highlight (e.g., target position).
`n_neighbors`	For UMAP, number of neighbors. Default is 15.
`perplexity`	For t-SNE, perplexity parameter. Default is 30.

Details

This visualization helps understand:

- How faces cluster by identity
- Which foils are closest to the target in embedding space
- Overall structure of similarity in a lineup

Value

A ggplot2 object.

Examples

```
## Not run:
# Get embeddings for all lineup members
all_faces <- c("target.jpg", paste0("foil", 1:5, ".jpg"))
embs <- batch_embeddings(all_faces)

# Plot with target highlighted
plot_embedding_space(embs, labels = c("Target", paste0("F", 1:5)), highlight = 1)

## End(Not run)
```

---

plot_error_rate_conf *Plot Error Rate by Confidence Level*

Description

Creates a ggplot visualization showing the mistaken identification rate (from target-absent lineups) across confidence levels.

Usage

```
plot_error_rate_conf(ppv_range_obj, show_points = TRUE)
```

Arguments

ppv_range_obj	A lineup_ppv_range object from ppv_range_by_confidence(), or a dataframe with columns: confidence, error_rate
show_points	Logical. Whether to show points at each confidence level (default = TRUE)

Details

The error rate is computed from target-absent (culprit-absent) lineups as: (suspect IDs + filler IDs) / total target-absent trials

Higher confidence should generally be associated with lower error rates (better calibration).

Value

A ggplot2 object

Examples

```
data(lineup_example)
ppv_range <- ppv_range_by_confidence(lineup_example,
                                     confidence_bins = c(0, 60, 80, 100))
plot_error_rate_conf(ppv_range)
```

`plot_esize_distribution` *Plot Bootstrap Distribution of Effective Size*

Description

Plot Bootstrap Distribution of Effective Size

Usage

```
plot_esize_distribution(
  boot_values,
  metric = "Effective size",
  bins = NULL,
  binwidth = NULL
)
```

Arguments

`boot_values`	Numeric vector of bootstrap estimates, as returned by `esize_boot_dist`.
`metric`	Label for the effective size statistic.
`bins`	Number of histogram bins. Defaults to `NULL`, which selects the bin width automatically.
`binwidth`	Histogram bin width. Defaults to `NULL`, which selects the bin width automatically. Takes precedence over `bins`.

Details

Effective size is a function of the whole table of choices, so its bootstrap distribution is usually fine-grained enough to bin as a continuous quantity, and the default is then 30 bins. When the number of lineup choices is small the estimates become visibly discrete, and the automatic rule bins at the resolution of the attainable values instead. See `plot_lineup_bias_distribution` for the rationale.

Value

A ggplot object.

Examples

```
vec <- round(runif(200, 1, 6))
dist <- esize_boot_dist(vec, k = 6, metric = "tredoux", R = 500)
plot_esize_distribution(dist, metric = "E (Tredoux, 1998)")
```

`plot_fullroc` *Plot Full ROC Curve*

Description

Creates a ggplot2 visualization of full ROC data from lineup identification experiments using the Smith & Yang (2020) method.

Usage

```
plot_fullroc(
  fullroc_obj,
  show_auc = TRUE,
  point_labels = FALSE,
  title = "Full ROC Curve (Smith & Yang, 2020)"
)
```

Arguments

`fullroc_obj`	Either the list returned by `make_fullroc_data()` or the `lineup_fullroc` object returned by `make_fullroc()`. The two record the ordering method in different places, and both are accepted.
`show_auc`	Logical. Whether to display AUC value on plot (default = TRUE)
`point_labels`	Logical. Whether to label points (default = FALSE, can be crowded)
`title`	Character. Plot title (default = "Full ROC Curve (Smith & Yang, 2020)")

Value

A ggplot2 object. Being an ordinary ggplot, it can be modified with the usual + syntax, e.g. `p + ggplot2::theme_minimal()`.

Examples

```
data(lineup_example)
fullroc_result <- make_fullroc_data(lineup_example)
plot_fullroc(fullroc_result)
```

plot_lineup_bias_distribution
Plot Bootstrap Distribution of Lineup Bias

Description

Plot Bootstrap Distribution of Lineup Bias

Usage

```
plot_lineup_bias_distribution(
  boot_values,
  target_pos = NULL,
  bins = NULL,
  binwidth = NULL
)
```

Arguments

boot_values	Numeric vector of bootstrap estimates, as returned by lineup_bias_boot_dist.
target_pos	Optional target position label for title.
bins	Number of histogram bins. Defaults to NULL, which selects the bin width automatically (see Details).
binwidth	Histogram bin width. Defaults to NULL, which selects the bin width automatically (see Details). Takes precedence over bins.

Details

Bootstrap estimates of a lineup proportion are counts divided by the number of lineup choices, so they fall on a discrete lattice with spacing 1/n rather than on a continuum. Binning such values at an arbitrary resolution produces a misleading histogram: bins that contain no attainable value are empty regardless of the data, and bins that happen to straddle two attainable values are twice as tall as their neighbours. When neither bins nor binwidth is supplied, the lattice spacing is recovered (from the "n" attribute when present, otherwise from the values themselves) and used as the bin width, so each bar covers exactly one attainable proportion. Estimates that are effectively continuous fall back to 30 bins.

Value

A ggplot object.

Examples

```
vec <- round(runif(200, 1, 6))
dist <- lineup_bias_boot_dist(vec, target_pos = 3, k = 6, R = 500)
plot_lineup_bias_distribution(dist, target_pos = 3)
```

plot_lineup_similarity *Plot Lineup Similarity Results*

Description

Creates a bar plot showing the similarity of each foil to the target face, describing scores in a recognition model's embedding space, not measuring or certifying lineup fairness.

Usage

```
plot_lineup_similarity(
  lineup,
  show_threshold = TRUE,
  use_similarity = TRUE,
  title = NULL,
  colors = NULL
)
```

Arguments

lineup	A lineup_similarity object from lineup_similarity.
show_threshold	Logical. If TRUE, adds a horizontal line at the verification threshold. Default is TRUE.
use_similarity	Logical. If TRUE, plots similarity scores (higher = more similar). If FALSE, plots distance (lower = more similar). Default is TRUE.
title	Optional plot title.
colors	Color palette for bars. Default uses a gradient from blue (dissimilar) to red (similar).

Value

A ggplot2 object.

Examples

```
## Not run:
lineup <- lineup_similarity("target.jpg", foil_paths)
plot_lineup_similarity(lineup)

# Show distance instead of similarity
plot_lineup_similarity(lineup, use_similarity = FALSE)

## End(Not run)
```

`plot_msdt_joint` *Plot mSDT Joint Distributions*

Description

Contour plot of the joint distribution of suspect signal and max filler signal.

Usage

```
plot_msdt_joint(
  dprime,
  gamma,
  lineup_size,
  lineup_type = c("culprit_present", "culprit_absent"),
  n = 120,
  xlim = NULL,
  ylim = NULL,
  show_decision = TRUE
)
```

Arguments

`dprime`	Discriminability between guilty and innocent suspects.
`gamma`	Decision criterion.
`lineup_size`	Lineup size (suspect + fillers), must be >= 2.
`lineup_type`	Either "culprit_present" or "culprit_absent".
`n`	Grid resolution per axis.
`xlim`, `ylim`	Optional numeric ranges for axes.
`show_decision`	Logical. If TRUE, overlays MAX rule boundaries.

Value

A ggplot object.

Examples

```
# Joint distribution for a culprit-present 6-person lineup
plot_msdt_joint(dprime = 1.5, gamma = 0.8, lineup_size = 6)
```

`plot_ppv_range` *Plot PPV Range Across Confidence Levels*

Description

Creates a ggplot visualization showing the range of PPV estimates across the three correction methods (nominal, effective, none) as recommended by Fitzgerald et al. (2023).

Usage

```
plot_ppv_range(ppv_range_obj, show_band = TRUE, show_points = TRUE)
```

Arguments

`ppv_range_obj`	A lineup_ppv_range object from ppv_range_by_confidence()
`show_band`	Logical. Whether to show shaded uncertainty band between nominal and none estimates (default = TRUE)
`show_points`	Logical. Whether to show points at each confidence level (default = TRUE)

Details

The plot shows three curves:

- **Nominal** (blue, dashed): Best-case estimate assuming fair lineup
- **Effective** (red, solid): Size-based approximation accounting for uneven choices
- **None** (gray, dashed): Worst-case estimate (no correction)

The shaded band between nominal and none represents the PPV uncertainty range due to unknown lineup fairness conditions. The effective size estimate typically falls within this range and is a size-based approximation, not a validated best estimate.

Value

A ggplot2 object

References

Fitzgerald, R. J., Tredoux, C. G., & Juncu, S. (2023). Estimation of eyewitness error rates in fair and biased lineups. *Law and Human Behavior*.

Examples

```
data(lineup_example)
ppv_range <- ppv_range_by_confidence(lineup_example,
                                     confidence_bins = c(0, 60, 80, 100))
plot_ppv_range(ppv_range)
```

plot_roc_cac *Plot Methods for ROC and CAC Objects*

Description

`plot()` methods for the objects returned by the ROC and CAC functions, so that `plot(x)` works directly rather than requiring the user to know which plot builder matches which object.

Usage

```
## S3 method for class 'lineup_roc_data'
plot(x, ...)

## S3 method for class 'lineup_roc'
plot(x, ...)

## S3 method for class 'lineup_cac_data'
plot(x, ...)

## S3 method for class 'lineup_cac'
plot(x, ...)

## S3 method for class 'lineup_fullroc_data'
plot(x, ...)

## S3 method for class 'lineup_fullroc'
plot(x, ...)
```

Arguments

`x`	An object returned by one of the ROC or CAC functions listed above.
`...`	Additional arguments passed to the underlying plot builder.

Details

Each ROC/CAC analysis comes in two forms. The `*_data()` function computes and returns the data only; the function without the suffix wraps it, builds the plot, and returns a classed object with the plot stored in `$plot`:

- `make_rocdata` / `make_roc`
- `make_cacdata` / `make_cac`
- `make_fullroc_data` / `make_fullroc`

These methods accept either form. When a plot has already been built it is returned as-is; otherwise it is constructed with the matching builder (`make_roc_gg`, `make_cac_gg` or `plot_fullroc`).

The value is an ordinary ggplot object, so it can be modified with the usual `+` syntax, e.g. `plot(x) + ggplot2::theme_minimal()`.

Value

A ggplot2 object.

Examples

```
data(lineup_example)
plot(make_rocdata(lineup_example))
plot(make_cacdata(lineup_example))
```

`plot_similarity_distribution` *Plot Similarity Distribution*

Description

Creates a histogram or density plot showing the distribution of face similarities, useful for understanding the range of similarities in a lineup or database.

Usage

```
plot_similarity_distribution(
  similarities,
  type = c("histogram", "density"),
  threshold = NULL,
  xlab = "Similarity",
  title = "Distribution of Face Similarities"
)
```

Arguments

`similarities`	Numeric vector of similarity scores or distances.
`type`	Plot type: "histogram" or "density".
`threshold`	Optional threshold value to show as vertical line.
`xlab`	X-axis label.
`title`	Plot title.

Value

A ggplot2 object.

Examples

```
set.seed(123)
similarities <- rbeta(100, 2, 5)
plot_similarity_distribution(similarities, threshold = 0.6)
```

plot_utility_curves *Plot Expected Utility Curves*

Description

Creates a plot showing expected utility across confidence criteria.

Usage

```
plot_utility_curves(utility_obj, show_max = TRUE, show_reject = TRUE)
```

Arguments

utility_obj	List output from make_utility_curves()
show_max	Logical. Whether to highlight maximum utility point. Default = TRUE.
show_reject	Logical. Whether to show the reject-all utility, drawn as a dashed horizontal reference line. Default = TRUE.

Details

The utility curve shows the expected utility at each confidence criterion. Higher confidence criteria have lower hit rates but also lower false alarm rates. The optimal criterion maximizes expected utility given the base rate and cost structure.

Value

A ggplot2 object

Examples

```
data(lineup_example)
util <- make_utility_curves(lineup_example)
plot_utility_curves(util)
```

plot_utility_difference *Plot Utility Difference Curve*

Description

Creates a plot showing the difference in expected utility between two procedures across base rates.

Usage

```
plot_utility_difference(util_diff_obj, show_crossover = TRUE)
```

Arguments

util_diff_obj	List output from make_utility_difference()
show_crossover	Logical. Whether to mark crossover points. Default = TRUE.

Details

Utility difference curves reveal which procedure is superior at different base rates. Regions above zero favor Procedure A, while regions below zero favor Procedure B. Crossover points indicate base rates where procedures are equivalent.

Value

A ggplot2 object

Examples

```
data(lineup_example)
odd <- seq(1, nrow(lineup_example), by = 2)
util_diff <- make_utility_difference(lineup_example[odd, ],
                                     lineup_example[-odd, ],
                                     base_rate_grid = seq(0.1, 0.9, 0.1))
plot_utility_difference(util_diff)
```

---

ppv_by_confidence *Compute PPV by Confidence with Lineup Size Correction*

Description

Calculates Positive Predictive Value (PPV) for each confidence level, using a specified method to correct for lineup size when estimating innocent-suspect identification rates.

Usage

```
ppv_by_confidence(
  data,
  lineup_size = 6,
  confidence_bins = NULL,
  correction = c("nominal", "effective", "none"),
  effective_size_data = NULL,
  quiet = FALSE,
  innocent_suspect = "auto",
  base_rate = NULL
)
```

Arguments

`data`	A dataframe with columns: • target_present: Logical. TRUE if guilty suspect in lineup • identification: Character. "suspect", "filler", or "reject" • confidence: Numeric. Confidence rating
`lineup_size`	Integer. Nominal number of people in lineup (default = 6)
`confidence_bins`	Numeric vector of bin edges for grouping confidence (e.g., c(0, 60, 80, 100)). If NULL, uses individual confidence levels.
`correction`	Character. Method for estimating innocent-suspect IDs: "nominal" (best-case, assumes fair lineup), "effective" (accounts for bias), "none" (worst-case, no correction). Default = "nominal".
`effective_size_data`	Optional dataframe with pre-computed effective sizes per confidence bin (must have columns: conf_level, effective_size). If NULL and correction="effective", computes from data.
`quiet`	Logical. Suppress the note issued when `correction` is not applied because the data designate an innocent suspect, and the note issued when the two arms differ in size. Default `FALSE`.
`innocent_suspect`	How innocent-suspect identifications in target-absent lineups are obtained. `"designated"`: the lineups had a designated innocent suspect, so target-absent `"suspect"` rows are counted directly, and zero such rows means a false-ID rate of zero. `"estimated"`: there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by `lineup_size`. `"auto"` (the default) chooses `"designated"` when any target-absent row is a suspect ID or the data have a non-missing `suspect_pos` column, and `"estimated"` otherwise, with a message.
`base_rate`	Prior probability that the suspect is guilty. `NULL` (default) computes PPV from counts, as below. A number strictly between 0 and 1 computes it from the guilty-ID rate (per target-present trial) and the innocent-suspect-ID rate (per target-absent trial) at that prior, so the result does not depend on the TP:TA split.

Details

PPV (Positive Predictive Value) is the probability that a suspect is guilty given an identification. The formula is:

PPV = guilty_suspect_IDs / (guilty_suspect_IDs + innocent_suspect_IDs)

Different correction methods estimate innocent_suspect_IDs differently:

- "nominal": assumes perfect lineup fairness (error_rate / lineup_size)
- "effective": uses effective size to account for bias (error_rate / effective_size)
- "none": assumes all errors are innocent-suspect IDs (error_rate)

Note on the base rate: by default PPV is computed from raw target-present and target-absent counts, so the prior probability of guilt is implicitly the experiment's TP:TA ratio (e.g., 0.5 for the usual balanced design), and a message is issued when the two arms differ in size. Set `base_rate` to compute $\pi g(c)/(\pi g(c) + (1-\pi)i(c))$ from the guilty-ID rate $g(c)$ and innocent-suspect-ID rate $i(c)$ instead; `base_rate = 0.5` reproduces the balanced-design value at any sample sizes. PPV is *not* base-rate-free: at different real-world base rates the PPV of the same procedure will differ. When target-absent lineups have no designated innocent suspect, the correction also relies on the pseudo-distribution assumption of spreading filler choices over lineup members, which is a modeling approximation.

Value

A list containing:

- ppv_data: Dataframe with PPV by confidence level
- overall_ppv: Overall PPV across all confidence levels
- correction_method: The correction method requested
- innocent_suspect_method: "designated" when target-absent trials identify an innocent suspect directly, otherwise the estimation route
- lineup_size: Nominal lineup size

When the correction is ignored

Lineup-size corrections exist to *estimate* how many of the observed target-absent identifications fell on an innocent suspect, in data where no innocent suspect was designated. When the data do designate one, that count is observed rather than estimated, and `correction` has no work to do: all three methods return the same PPV. This is correct, not a fault, but it is easy to mistake for one, so a message is issued unless `quiet = TRUE`. `lineup_example` is data of this kind.

References

Fitzgerald, R. J., Tredoux, C. G., & Juncu, S. (2023). Estimation of eyewitness error rates in fair and biased lineups. *Law and Human Behavior*.

Examples

```
data(lineup_example)
# Nominal correction (assumes fair lineup)
ppv_result <- ppv_by_confidence(lineup_example,
                                correction = "nominal",
                                confidence_bins = c(0, 60, 80, 100))

# Effective size correction (accounts for bias)
ppv_result <- ppv_by_confidence(lineup_example,
                                correction = "effective",
                                confidence_bins = c(0, 60, 80, 100))
```

ppv_range_by_confidence *Compute PPV Range by Confidence (All Three Corrections)*

Description

Calculates PPV for each confidence level using all three correction methods (nominal, effective, none) to show the range of plausible PPV estimates.

Usage

```
ppv_range_by_confidence(
  data,
  lineup_size = 6,
  confidence_bins = NULL,
  effective_size_data = NULL,
  innocent_suspect = "auto",
  base_rate = NULL
)
```

Arguments

`data`	A dataframe with columns: target_present, identification, confidence
`lineup_size`	Integer. Nominal number of people in lineup (default = 6)
`confidence_bins`	Numeric vector of bin edges (optional)
`effective_size_data`	Optional pre-computed effective sizes
`innocent_suspect`	How innocent-suspect identifications in target-absent lineups are obtained. `"designated"`: the lineups had a designated innocent suspect, so target-absent `"suspect"` rows are counted directly, and zero such rows means a false-ID rate of zero. `"estimated"`: there was no designated innocent suspect, so the innocent-suspect count is estimated as filler IDs divided by `lineup_size`. `"auto"` (the default) chooses `"designated"` when any target-absent row is a suspect ID or the data have a non-missing `suspect_pos` column, and `"estimated"` otherwise, with a message.
`base_rate`	Prior probability that the suspect is guilty. `NULL` (default) computes PPV from counts, as below. A number strictly between 0 and 1 computes it from the guilty-ID rate (per target-present trial) and the innocent-suspect-ID rate (per target-absent trial) at that prior, so the result does not depend on the TP:TA split.

Details

This function provides the full PPV range recommended by Fitzgerald et al. (2023):

- **Nominal** (upper bound): Assumes perfectly fair lineup
- **Effective**: Accounts for lineup bias via effective size
- **None** (lower bound): Assumes worst-case (all errors are innocent-suspect)

The range between nominal and none represents the uncertainty in PPV due to unknown lineup fairness conditions.

Value

A list of class "lineup_ppv_range" containing:

- ppv_range_data: Dataframe with all three PPV estimates per confidence level
- ppv_nominal: Nominal correction results
- ppv_effective: Effective size correction results
- ppv_none: No correction results
- lineup_size: Nominal lineup size

References

Fitzgerald, R. J., Tredoux, C. G., & Juncu, S. (2023). Estimation of eyewitness error rates in fair and biased lineups. *Law and Human Behavior*.

Examples

```
data(lineup_example)
# Compute full PPV range
ppv_range <- ppv_range_by_confidence(lineup_example,
                                     confidence_bins = c(0, 60, 80, 100))
print(ppv_range)

# Access individual corrections
ppv_range$ppv_nominal$overall_ppv
ppv_range$ppv_effective$overall_ppv
ppv_range$ppv_none$overall_ppv
```

---

print.esize_T_bayes *Print method for esize_T_bayes objects*

Description

Print method for esize_T_bayes objects

Usage

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

Arguments

x	An object of class "esize_T_bayes".
...	Further arguments (unused).

---

print.esize_T_bayes_compare *Print method for esize_T_bayes_compare objects*

Description

Print method for esize_T_bayes_compare objects

Usage

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

Arguments

x	An object of class "esize_T_bayes_compare".
...	Further arguments (unused).

---

print.lineup_anri *Print Method for ANRI Objects*

Description

Print Method for ANRI Objects

Usage

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

Arguments

x	An anri object from compute_anri()
...	Additional arguments (ignored)

print.lineup_anri_comparison

Print Method for ANRI Comparison Objects

Description

Print Method for ANRI Comparison Objects

Usage

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

Arguments

x	A comparison object from compare_anri()
...	Additional arguments (ignored)

print.lineup_bayes_curves

Print Method for Bayes Curves Objects

Description

Print Method for Bayes Curves Objects

Usage

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

Arguments

x	A bayes_curves object from make_bayes_curves()
...	Additional arguments (ignored)

print.lineup_cac *Print Method for lineup_cac Objects*

Description

Print Method for lineup_cac Objects

Usage

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

Arguments

x	A lineup_cac object
...	Additional arguments (ignored)

print.lineup_calibration *Print Method for lineup_calibration Objects*

Description

Print Method for lineup_calibration Objects

Usage

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

Arguments

x	A lineup_calibration object
...	Additional arguments (ignored)

print.lineup_data *Print Method for Lineup Data*

Description

Print Method for Lineup Data

Usage

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

Arguments

x	A lineup_data object
...	Additional arguments (ignored)

print.lineup_dpp *Print Method for lineup_dpp Objects*

Description

Print Method for lineup_dpp Objects

Usage

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

Arguments

x	A lineup_dpp object
...	Additional arguments (ignored)

print.lineup_dpp_comparison *Print Method for lineup_dpp_comparison Objects*

Description

Print Method for lineup_dpp_comparison Objects

Usage

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

Arguments

x	A lineup_dpp_comparison object from compare_dpp()
...	Additional arguments (ignored)

print.lineup_eig *Print Method for lineup_eig Objects*

Description

Print Method for lineup_eig Objects

Usage

```
## S3 method for class 'lineup_eig'
print(x, n_responses = 10, ...)
```

Arguments

x	A lineup_eig object from compute_eig()
n_responses	Number of response categories to display (default = 10)
...	Additional arguments (ignored)

print.lineup_fullroc *Print Method for lineup_fullroc Objects*

Description

Print Method for lineup_fullroc Objects

Usage

```
## S3 method for class 'lineup_fullroc'
print(x, show_diagnosticity = TRUE, ...)
```

Arguments

x	A lineup_fullroc object
show_diagnosticity	Logical. Whether to print diagnosticity table (default = TRUE)
...	Additional arguments (ignored)

print.lineup_ppv *Print Method for lineup_ppv Objects*

Description

Print Method for lineup_ppv Objects

Usage

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

Arguments

x	A lineup_ppv object
...	Additional arguments (ignored)

print.lineup_ppv_range *Print Method for lineup_ppv_range Objects*

Description

Print Method for lineup_ppv_range Objects

Usage

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

Arguments

x	A lineup_ppv_range object
...	Additional arguments (ignored)

print.lineup_rac *Print Method for lineup_rac Objects*

Description

Print Method for lineup_rac Objects

Usage

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

Arguments

x	A lineup_rac object
...	Additional arguments (ignored)

print.lineup_roc *Print Method for lineup_roc Objects*

Description

Print Method for lineup_roc Objects

Usage

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

Arguments

x	A lineup_roc object
...	Additional arguments (ignored)

print.lineup_similarity *Print Method for lineup_similarity*

Description

Print Method for lineup_similarity

Usage

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

Arguments

x	A lineup_similarity object
...	Additional arguments (ignored)

print.lineup_table *Print Method for lineup_table Objects*

Description

Print Method for lineup_table Objects

Usage

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

Arguments

x	A lineup_table object.
...	Additional arguments (ignored).

Value

x, invisibly.

Examples

```
print(lineup_table(rep(c(5, 3, 2, 1, 8, 6, 7), each = 4), k = 8))
```

print.lineup_utility_comparison *Print Method for lineup_utility_comparison Objects*

Description

Print Method for lineup_utility_comparison Objects

Usage

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

Arguments

x	A lineup_utility_comparison object
...	Additional arguments (ignored)

print.model_comparison *Print Method for model_comparison Objects*

Description

Print Method for model_comparison Objects

Usage

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

Arguments

x	A model_comparison object
...	Additional arguments (ignored)

print.pauc_comparison *Print Method for pauc_comparison Objects*

Description

Print Method for pauc_comparison Objects

Usage

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

Arguments

x	A pauc_comparison object
...	Additional arguments (ignored)

print.sdt_roc_fit *Print Method for SDT ROC Fit*

Description

Print Method for SDT ROC Fit

Usage

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

Arguments

x	An sdt_roc_fit object
...	Additional arguments (ignored)

print.simulated_lineup_data *Print Method for Simulated Lineup Data*

Description

Print Method for Simulated Lineup Data

Usage

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

Arguments

x	A simulated_lineup_data object
...	Additional arguments (ignored)

print.winter_2ht *Print Method for winter_2ht Objects*

Description

Print Method for winter_2ht Objects

Usage

```
## S3 method for class 'winter_2ht'
print(x, ...)
```

Arguments

x	A winter_2ht object
...	Additional arguments (not used)

print.winter_2ht_boot *Print Method for winter_2ht_boot Objects*

Description

Print Method for winter_2ht_boot Objects

Usage

```
## S3 method for class 'winter_2ht_boot'
print(x, ...)
```

Arguments

x	A winter_2ht_boot object
...	Additional arguments (not used)

rep_index *Rep index*

Description

Helper function that repeats a number k times

Usage

```
rep_index(index, num)
```

Arguments

index	Number to be repeated
num	Number of times to repeat index

Details

Can be used to create vectors of lineup data

Examples

```
rep_index(1, 10)
```

rot_vector *Rotate vector*

Description

Helper function to rotate vector

Usage

```
rot_vector(lineup_vec)
```

Arguments

lineup_vec	A vector of lineup data

Examples

```
# Rotate lineup positions by one place
rot_vector(1:6)
```

run_r4lineups_app *Run the r4lineups Shiny app*

Description

Launches the bundled Shiny app. It covers lineup fairness from mock-witness choices (bias, effective and functional size, diagnosticity), Bayesian posteriors, Lee and Penrod's Multi-d' model, ROC/CAC/full ROC with pAUC comparison, calibration and decision analyses (ANRI, DPP, expected utility, Bayesian and BREE curves, RAC), expected information gain and PPV ranges, and, when Python is available, face similarity. Every result is computed by an exported r4lineups function.

Usage

```
run_r4lineups_app()
```

Details

The app needs the suggested packages **shiny**, **bslib** and **DT**.

Value

None. Launches a Shiny app.

Examples

```
## Not run:
run_r4lineups_app()

## End(Not run)
```

sdt_compare *Bayesian Comparison of SDT Sensitivity (d') Between Two Conditions*

Description

Computes a Bayesian posterior distribution for the difference $\Delta d' = d'_A - d'_B$ using independent Jeffreys Beta posteriors for the hit and false-alarm rates in each condition.

Usage

```
sdt_compare(
  hits_A,
  misses_A,
  fas_A,
  crs_A,
  hits_B,
  misses_B,
  fas_B,
  crs_B,
  label_A = "A",
  label_B = "B",
  alpha = 0.5,
  S = 10000,
  credible_mass = 0.95
)
```

Arguments

`hits_A`, `misses_A`, `fas_A`, `crs_A`	Integer counts for condition A.
`hits_B`, `misses_B`, `fas_B`, `crs_B`	Integer counts for condition B.
`label_A`, `label_B`	Character labels for conditions (default "A", "B").
`alpha`	Prior concentration for the Beta prior on each rate. Default 0.5 (Jeffreys). Named shortcuts: `"jeffreys"`, `"uniform"`, `"weak"`.
`S`	Number of posterior draws. Default 10000.
`credible_mass`	Width of the equal-tailed credible interval. Default 0.95.

Details

Independent Jeffreys Beta posteriors are placed on the hit and false-alarm rates in each condition:

$$HR_A \mid \mathbf{n} \sim \text{Beta}(\text{hits}_A + \alpha,\ \text{misses}_A + \alpha)$$

$$FAR_A \mid \mathbf{n} \sim \text{Beta}(\text{fas}_A + \alpha,\ \text{crs}_A + \alpha)$$

The posterior d' draws are:

$$d'^{(s)}_A = \Phi^{-1}(HR^{(s)}_A) - \Phi^{-1}(FAR^{(s)}_A)$$

and analogously for B. The posterior of the difference is $\Delta d'^{(s)} = d'^{(s)}_A - d'^{(s)}_B$, from which $P(d'_A > d'_B \mid \mathbf{n})$ is directly read off.

The Jeffreys prior ($\alpha = 0.5$) handles zero cell counts gracefully and provides direct probability statements - unlike the asymptotic z-test in `compare_sdt_summary`.

Value

An object of class "sdt_compare" containing:

delta_dprime_draws Posterior draws of $\Delta d'$.

dprime_A_draws, dprime_B_draws Per-condition d' draws.

posterior_mean, posterior_median Posterior summaries of $\Delta d'$.

credible_interval Named lower/upper CI on $\Delta d'$.

P_A_greater Posterior probability that $d'_A > d'_B$.

P_B_greater Posterior probability that $d'_B > d'_A$.

dprime_A_mean, dprime_B_mean Posterior means of each d'.

frequentist Output of `compare_sdt_summary()` for comparison.

prior_alpha, S, credible_mass Metadata.

References

Macmillan, N. A., & Creelman, C. D. (2005). *Detection Theory: A User's Guide* (2nd ed.). Lawrence Erlbaum.

Wixted, J. T., & Mickes, L. (2014). A signal-detection-based diagnostic-feature detection model of eyewitness identification. *Psychological Review, 121*(2), 262-276.

See Also

compare_sdt_summary, sdt_summary_from_counts, diag_ratio_T_bayes

Examples

```
# Simultaneous vs. sequential lineup comparison
res <- sdt_compare(
  hits_A = 45, misses_A = 55, fas_A = 12, crs_A = 88,
  hits_B = 38, misses_B = 62, fas_B = 10, crs_B = 90,
  label_A = "Simultaneous", label_B = "Sequential"
)
print(res)
plot(res)
```

---

sdt_summary_from_counts

SDT Summary Statistics from Counts

Description

Compute d', criterion, and beta from 2x2 SDT counts (hits, false alarms, misses, correct rejections).

Usage

```
sdt_summary_from_counts(
  hits,
  fas,
  misses,
  cr,
  correction = c("loglinear", "half", "none")
)
```

Arguments

hits	Number of hits.
fas	Number of false alarms.
misses	Number of misses.
cr	Number of correct rejections.
correction	Correction for extreme rates ("loglinear", "half", or "none").

Value

A list with rates and SDT metrics.

Examples

```
# 70 hits, 30 misses; 20 false alarms, 80 correct rejections
summ <- sdt_summary_from_counts(hits = 70, fas = 20, misses = 30, cr = 80)
summ$dprime
summ$c
```

---

sdt_summary_variance *Variance Estimates for SDT Summary Metrics*

Description

Estimates the variance of d' (and criterion) from summary SDT counts using the Gourevitch & Galanter (1967) delta method, the Miller (1996) exact-binomial method, or a parametric bootstrap.

Usage

```
sdt_summary_variance(
  hits,
  fas,
  misses,
  cr,
  method = c("miller", "gourevitch", "bootstrap"),
  correction = c("loglinear", "half", "none"),
  nboot = 1000,
  seed = NULL
)
```

Arguments

hits	Number of hits.
fas	Number of false alarms.
misses	Number of misses.
cr	Number of correct rejections.
method	Variance method: "gourevitch", "miller", or "bootstrap".
correction	Correction for extreme rates ("loglinear", "half", or "none").
nboot	Number of bootstrap samples when method = "bootstrap".
seed	Optional random seed for bootstrap variance.

Details

Miller enumeration is exact for a binomial distribution at its supplied probability. Here that probability is estimated from the observed counts. The resulting variance and downstream Wald test are plug-in approximations, not exact finite-sample inference.

Value

A list with variance estimates for zH, zF, d', and c.

Examples

```
v <- sdt_summary_variance(hits = 70, fas = 20, misses = 30, cr = 80,
                          method = "miller")
v$var_dprime
```

---

simulate_lineup_data *Simulate Lineup Identification Data*

Description

Generates simulated eyewitness lineup data based on signal detection theory parameters. Useful for power analysis, method validation, and teaching.

Usage

```
simulate_lineup_data(
  n_tp = 100,
  n_ta = 100,
  d_prime = 1.5,
  c_criterion = 0,
  lineup_size = 6,
  conf_levels = 5,
  decision_rule = c("max", "best_rest", "ensemble", "integration"),
  foil_bias = NULL,
  include_response_time = FALSE,
  seed = NULL
)
```

Arguments

`n_tp`	Integer. Number of target-present lineups (default = 100)
`n_ta`	Integer. Number of target-absent lineups (default = 100)
`d_prime`	Numeric. Discriminability (d') between target and lure distributions. Higher values = better memory. Typical range: 0.5 to 3.0 (default = 1.5)
`c_criterion`	Numeric scalar or vector of ordered decision criteria. The first value is always the identification/rejection threshold. With confidence ratings, a scalar generates `conf_levels` thresholds from `c_criterion` through `c_criterion + 2`; alternatively, supply exactly one strictly increasing threshold per confidence level. Lower values are more liberal (default = 0).
`lineup_size`	Integer. Number of lineup members (default = 6)
`conf_levels`	Integer. Number of confidence levels to simulate. If NULL, returns binary decision only (default = 5)
`decision_rule`	Character. Decision strategy for lineup choices: • "max" - Independent observations, choose highest strength (default) • "best_rest" - Best strength minus the mean of the other members • "ensemble" - Best strength minus the mean of all lineup members • "integration" - Sum memory strengths across all members
`foil_bias`	Numeric vector of length `lineup_size - 1`, giving the mean memory strength of each foil. Defaults to `NULL`, which places every foil at `N(0, 1)`. Supplying a vector makes the foils differ in how plausible they are, so that they are no longer interchangeable. Values are means on the same scale as `d_prime`: a foil at `0.5` is chosen more often than one at `-0.5`, and a strongly negative value produces a foil that is rarely or never chosen — a null foil.
`include_response_time`	Logical. Whether to simulate response times correlated with memory strength (default = FALSE)
`seed`	Integer. Random seed for reproducibility (default = NULL)

Details

This function implements signal detection models with multiple decision rules:

- Target distribution: Normal(d_prime, 1)
- Lure distribution: Normal(0, 1)
- Decision: Depends on decision_rule parameter

Decision Rules:

- **MAX** (default): Independent observations model. Choose lineup member with highest memory strength (Clark, 2003).
- **BEST-REST**: Best strength minus the average of the remaining members (Clark, 2003).
- **Ensemble**: Best strength minus the average of all lineup members (Wixted et al., 2018). For lineup size k, this is $(k-1)/k$ times BEST-REST, so the two are equivalent model parameterizations when their criteria are rescaled by the same factor.

- **Integration**: Sum memory strengths across all lineup members. Represents complete integration of evidence (Wixted et al., 2018).

All rules use independent signals with target distribution $N(d', 1)$ and filler distribution $N(0, 1)$. Thus, this simulator represents the equal-variance, zero-correlation special case of the lineup models. It does not implement the shared-variance or unequal-variance likelihood models fitted by Wixted et al. (2018).

The simulation assumes:

- Fair lineups (all fillers equally similar to description)
- Perfect attention (no guessing without memory)
- Normal distributions for memory strength

Response times (if simulated) are a heuristic convenience, not a fitted reaction-time or drift-diffusion model:

- Faster for stronger memory signals
- A bounded noisy function of decision strength

Value

A dataframe with columns:

- participant_id: Unique ID for each trial
- target_present: Logical. TRUE if target in lineup
- lineup_member: Integer. Position of the lineup member actually chosen, or 0 for a rejection. Each foil is simulated separately and retains its own identity; foils are not treated as interchangeable
- suspect_pos: Integer. Position occupied by the suspect (1 by construction; the target in target-present lineups, the designated innocent suspect in target-absent lineups)
- identification: Character. "suspect", "filler", or "reject", derived from lineup_member and suspect_pos. See `classify_lineup_choices`
- confidence: Numeric (if conf_levels specified). Confidence rating
- response_time: Numeric (if include_response_time = TRUE). RT in ms

Foil identity

Memory strengths are drawn independently for every lineup member, so the foils are distinguishable and are not exchangeable. `lineup_member` preserves which member was chosen; `identification` is a derived summary of it. Analyses that need member-level information (for example the effective-size correction in `make_ppv_range`) can therefore use `lineup_member` directly rather than working from the collapsed labels.

References

Wixted, J. T., Vul, E., Mickes, L., & Wilson, B. M. (2018). Models of lineup memory. *Cognitive Psychology, 105*, 81–114. doi:10.1016/j.cogpsych.2018.06.001

Mickes, L., et al. (2024). pyWitness 1.0: A python eyewitness identification analysis toolkit. *Behavior Research Methods, 56*, 1533-1550.

Examples

```
# Basic simulation: strong memory with MAX rule (default)
strong_memory <- simulate_lineup_data(
  n_tp = 200, n_ta = 200,
  d_prime = 2.0,
  conf_levels = 3,
  seed = 42
)

# Ensemble decision rule
ensemble_data <- simulate_lineup_data(
  n_tp = 200, n_ta = 200,
  d_prime = 1.5,
  decision_rule = "ensemble",
  conf_levels = 5,
  seed = 42
)

# Integration decision rule
integration_data <- simulate_lineup_data(
  n_tp = 200, n_ta = 200,
  d_prime = 1.5,
  decision_rule = "integration",
  conf_levels = 5,
  seed = 42
)

# Compare ROC curves across decision rules
roc_max <- make_roc(strong_memory, lineup_size = 6)
roc_ensemble <- make_roc(ensemble_data, lineup_size = 6)


# Power analysis: vary sample size (few simulations, to keep this quick;
# use hundreds in practice)
power_results <- simulate_power_analysis(
  sample_sizes = c(50, 100),
  d_prime = 1.5,
  n_simulations = 5,
  n_bootstrap = 50,
  seed = 1
)
```

`simulate_power_analysis` *Simulate Power Analysis for Lineup Studies*

Description

Estimates statistical power to detect a difference in discriminability between two lineup conditions, by simulating both conditions at a range of sample sizes and recording how often the difference is detected.

Usage

```
simulate_power_analysis(
  sample_sizes,
  d_prime,
  d_prime_null = 0,
  n_simulations = 1000,
  test_function = NULL,
  alpha = 0.05,
  lineup_size = 6,
  n_bootstrap = 200,
  seed = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

`sample_sizes`	Integer vector of sizes to test (e.g., c(50, 100, 200)). A size n means n target-present *and* n target-absent witnesses in each of the two simulated conditions: 2n witnesses per condition and 4n in the whole simulated experiment.
`d_prime`	Numeric. Discriminability of the condition of interest.
`d_prime_null`	Numeric. Discriminability of the comparison condition. Defaults to 0, which asks how often the procedure can be shown to discriminate better than chance. Set it to a non-zero value to ask the more usual question of how often two working procedures can be told apart.
`n_simulations`	Integer. Number of simulated studies per sample size (default = 1000). Power is a proportion estimated from these, so its own standard error is roughly `sqrt(0.25 / n_simulations)`.
`test_function`	Function of two dataframes returning a p-value. Defaults to a bootstrap comparison of partial AUC via `compare_pauc`.
`alpha`	Numeric. Significance level (default = 0.05)
`lineup_size`	Integer. Lineup size for both conditions (default = 6)
`n_bootstrap`	Integer. Bootstrap resamples used by the default test (default = 200). This is the dominant cost: the default test is run `n_simulations` times for every sample size.
`seed`	Optional integer seed for reproducibility.
`verbose`	Logical. Report progress (default = TRUE).
`...`	Additional arguments passed to simulate_lineup_data(), for example `conf_levels` or `foil_bias`.

Details

Power is the probability that a study of a given size detects an effect that is really there. Estimating it by simulation means specifying the effect, generating many studies of that size, applying the analysis you intend to use, and counting how often it reaches significance.

That requires a null to test against, which is what `d_prime_null` supplies. Simulating a single condition and computing a statistic from it is not a power analysis, because no hypothesis is being tested.

Value

A dataframe of class `power_analysis` with columns:

- sample_size: n, the number of target-present (and, separately, target-absent) witnesses per condition
- n_total: Witnesses in the whole simulated experiment, 4n
- power: Proportion of simulated studies reaching significance
- power_se: Standard error of that proportion
- mean_effect: Mean observed difference in pAUC
- sd_effect: Standard deviation of the observed difference
- n_valid: Simulations that produced a usable test
- n_failed: Attempted simulations without a usable test
- n_attempted: All attempted simulations

Power and its Monte Carlo standard error are conditional on usable tests. Failures are retained in the attempted count and trigger a warning; compare failure rates as well as conditional power when assessing a design.

See Also

`compare_pauc`, `simulate_lineup_data`

Examples

```
# Power to show discriminability better than chance
power_res <- simulate_power_analysis(
  sample_sizes = c(50, 100),
  d_prime = 1.5,
  n_simulations = 8,
  n_bootstrap = 50,
  conf_levels = 5,
  seed = 123
)
print(power_res)
```

`standardize_lineup_data` *Standardize Lineup Identification Data*

Description

Converts lineup data from various formats into the standardized r4lineups format. Handles common data issues and ensures compatibility with all confidence-based analyses.

Usage

```
standardize_lineup_data(
  data,
  target_present_col = "target_present",
  identification_col = "identification",
  confidence_col = "confidence",
  response_time_col = NULL,
  participant_id_col = NULL,
  recode_identification = NULL,
  validate = TRUE
)
```

Arguments

`data`	A data frame with lineup identification data
`target_present_col`	Name of column indicating target presence (default = "target_present"). Can also be "tp", "culprit_present", etc.
`identification_col`	Name of column with identification responses (default = "identification"). Can also be "response", "choice", etc.
`confidence_col`	Name of column with confidence ratings (default = "confidence"). Can also be "conf", "certainty", etc.
`response_time_col`	Name of column with response times (optional)
`participant_id_col`	Name of column with participant IDs (optional)
`recode_identification`	Named character vector for recoding identification values. E.g., c("target" = "suspect", "foil" = "filler")
`validate`	Logical. Whether to validate the result (default = TRUE)

Details

This function:

- Renames columns to standard names
- Converts data types as needed
- Recodes identification values to standard terms
- Adds participant IDs if missing
- Validates the result (if validate = TRUE)
- Adds S3 class "lineup_data"

Common recoding patterns:

- "target" → "suspect"
- "foil" / "distractor" → "filler"
- "none" / "not present" / "no choice" → "reject"

Value

A standardized data frame with class "lineup_data" containing:

- target_present: Logical
- identification: Character ("suspect", "filler", "reject")
- confidence: Numeric (if present)
- participant_id: Original or generated (if present)
- response_time: Numeric (if present)

followed by every other column of `data` unchanged, including `suspect_pos` and `lineup_member`, which some analyses use.

Examples

```
# Data with non-standard column names
raw_data <- data.frame(
  tp = c(1, 1, 0, 0),
  response = c("target", "foil", "none", "target"),
  conf = c(5, 3, 2, 4)
)

# Standardize
std_data <- standardize_lineup_data(
  raw_data,
  target_present_col = "tp",
  identification_col = "response",
  confidence_col = "conf",
  recode_identification = c(
    "target" = "suspect",
    "foil" = "filler",
    "none" = "reject"
```

```
  )
)
print(std_data)
```

`summary.lineup_eig` *Summary Method for lineup_eig Objects*

Description

Summary Method for lineup_eig Objects

Usage

```
## S3 method for class 'lineup_eig'
summary(object, ...)
```

Arguments

`object`	A lineup_eig object from compute_eig()
`...`	Additional arguments (ignored)

`summary.lineup_similarity`
Summary Method for lineup_similarity

Description

Summary Method for lineup_similarity

Usage

```
## S3 method for class 'lineup_similarity'
summary(object, ...)
```

Arguments

`object`	A lineup_similarity object
`...`	Additional arguments (ignored)

summary.model_comparison *Summary Method for model_comparison Objects*

Description

Summary Method for model_comparison Objects

Usage

```
## S3 method for class 'model_comparison'
summary(object, ...)
```

Arguments

object	A model_comparison object
...	Additional arguments (ignored)

summary.pauc_comparison *Summary Method for pauc_comparison Objects*

Description

Summary Method for pauc_comparison Objects

Usage

```
## S3 method for class 'pauc_comparison'
summary(object, ...)
```

Arguments

object	A pauc_comparison object
...	Additional arguments (ignored)

summary.sdt_roc_fit *Summary Method for SDT ROC Fit*

Description

Summary Method for SDT ROC Fit

Usage

```
## S3 method for class 'sdt_roc_fit'
summary(object, ...)
```

Arguments

object	An sdt_roc_fit object
...	Additional arguments (ignored)

summary.winter_2ht *Summary Method for winter_2ht Objects*

Description

Summary Method for winter_2ht Objects

Usage

```
## S3 method for class 'winter_2ht'
summary(object, ...)
```

Arguments

object	A winter_2ht object
...	Additional arguments (not used)

summary.winter_2ht_boot *Summary Method for winter_2ht_boot Objects*

Description

Summary Method for winter_2ht_boot Objects

Usage

```
## S3 method for class 'winter_2ht_boot'
summary(object, ...)
```

Arguments

object	A winter_2ht_boot object
...	Additional arguments (not used)

validate_lineup_data *Validate Lineup Identification Data*

Description

Checks whether a data frame meets the requirements for r4lineups confidence-based analyses. Ensures required columns exist with correct types and valid values.

Usage

```
validate_lineup_data(
  data,
  require_confidence = TRUE,
  require_response_time = FALSE,
  strict = FALSE
)
```

Arguments

data	A data frame to validate
require_confidence	Logical. Whether confidence column is required (default = TRUE). Set to FALSE for analyses that don't need confidence.
require_response_time	Logical. Whether response_time is required (default = FALSE)
strict	Logical. If TRUE, stops with error on validation failure. If FALSE, returns validation result with messages (default = FALSE)

Details

Required columns:

- ‘target_present‘: Logical (TRUE/FALSE) indicating if target in lineup
- ‘identification‘: Character or factor with values "suspect", "filler", or "reject"
- ‘confidence‘: Numeric (if require_confidence = TRUE). Higher = more confident

Optional columns:

- ‘participant_id‘: Unique identifier for each trial/participant
- ‘response_time‘: Numeric response time in milliseconds

Validation checks:

- All required columns present
- Correct data types
- No missing values in required columns
- Valid identification values
- Positive confidence values
- At least some target-present and target-absent trials

Value

If strict = FALSE, returns a list with:

- valid: Logical indicating if data passes validation
- messages: Character vector of validation messages/errors
- warnings: Character vector of warnings (non-fatal issues)

If strict = TRUE, either returns TRUE invisibly or stops with error.

Examples

```
# Valid data
valid_data <- data.frame(
  target_present = c(TRUE, TRUE, FALSE, FALSE),
  identification = c("suspect", "filler", "reject", "suspect"),
  confidence = c(5, 3, 2, 4)
)
validate_lineup_data(valid_data)

# Invalid data (missing column)
invalid_data <- data.frame(
  target_present = c(TRUE, FALSE),
  identification = c("suspect", "filler")
)
result <- validate_lineup_data(invalid_data, strict = FALSE)
print(result$messages)
```

```
# Strict validation (errors on failure)
validate_lineup_data(valid_data, strict = TRUE)
```

`var_diag_ratio` *Asymptotic Variance of Log Diagnosticity*

Description

Computes the large-sample variance of the log diagnosticity ratio for a lineup pair.

Usage

```
var_diag_ratio(lineup_pres, lineup_abs, pos_pres, pos_abs, k1, k2)
```

Arguments

`lineup_pres`	A numeric vector of lineup choices for a lineup in which the target was present
`lineup_abs`	A numeric vector of lineup choices for a lineup in which the target was absent
`pos_pres`	A scalar, representing target position in TP lineup. Must be declared by user
`pos_abs`	A scalar, representing target position in TA lineup. Must be declared by user
`k1`	Number of targets in TP lineup. Must be specified by user (scalar).
`k2`	Number of targets in TA lineup. Must be specified by user (scalar).

Details

Returns 1/h - 1/N_TP + 1/f - 1/N_TA for positive suspect-ID counts h and f. This is on the log-ratio scale, as in Tredoux (1998), not the raw ratio scale. A delta-method variance for the raw ratio is the squared diagnosticity ratio times this value.

References

Malpass, R. S. (1981). Effective size and defendant bias in eyewitness identification lineups. *Law and Human Behavior, 5*(4), 299-309.

Malpass, R. S., Tredoux, C., & McQuiston-Surrett, D. (2007). Lineup construction and lineup fairness. In R. Lindsay, D. F. Ross, J. D. Read, & M. P. Toglia (Eds.), *Handbook of Eyewitness Psychology, Vol. 2: Memory for people* (pp. 155-178). Mahwah, NJ: Lawrence Erlbaum Associates.

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22*(2), 217-237.

Tredoux, C. (1999). Statistical considerations when determining measures of lineup size and lineup bias. *Applied Cognitive Psychology*, 13, S9-S26.

Wells, G. L.,Leippe, M. R., & Ostrom, T. M. (1979). Guidelines for empirically assessing the fairness of a lineup. *Law and Human Behavior, 3*(4), 285-293.

Examples

```
#Data:
lineup_pres <- round(runif(100, 1, 6))
lineup_abs <- round(runif(100, 1, 6))
pos_pres <- 3
pos_abs <- 4

#Call:
var_d <- var_diag_ratio(lineup_pres, lineup_abs, pos_pres, pos_abs, 6, 6)
var_d <- var_diag_ratio(lineup_pres, lineup_abs, 3, 4, 6, 6)
```

---

var_lnd *Variance of ln of diagnosticity ratio*

Description

Function to compute variance of ln(d) for k lineup pairs

Usage

```
var_lnd(linedf)
```

Arguments

linedf	A dataframe of parameters for computing diagnosticity ratio

Details

To get linedf, use the diag_param helper function

diag_param returns a dataframe containing the following:

- *n11*: Number of mock witnesses who identified the suspect in the target present condition
- *n21*: Number of mock witnesses who did not identify the suspect in the target present condition
- *n12*: Number of mock witnesses who identified the suspect in the target absent condition
- *n22*: Number of mock witnesses who did not identify the suspect in the target absent condition

Value

A dataframe containing the estimated variance of the log diagnosticity ratio for each lineup.

References

Malpass, R. S. (1981). Effective size and defendant bias in eyewitness identification lineups. *Law and Human Behavior, 5*(4), 299-309.

Malpass, R. S., Tredoux, C., & McQuiston-Surrett, D. (2007). Lineup construction and lineup fairness. In R. Lindsay, D. F. Ross, J. D. Read, & M. P. Toglia (Eds.), *Handbook of Eyewitness Psychology, Vol. 2: Memory for people* (pp. 155-178). Mahwah, NJ: Lawrence Erlbaum Associates.

Tredoux, C. G. (1998). Statistical inference on measures of lineup fairness. *Law and Human Behavior, 22*(2), 217-237.

Tredoux, C. (1999). Statistical considerations when determining measures of lineup size and lineup bias. *Applied Cognitive Psychology*, 13, S9-S26.

Wells, G. L.,Leippe, M. R., & Ostrom, T. M. (1979). Guidelines for empirically assessing the fairness of a lineup. *Law and Human Behavior, 3*(4), 285-293.

Examples

```
#Target present data:
A <- rep(1:6, length.out = 100)
B <- rep(1:5, length.out = 70)
C <- rep(1:4, length.out = 20)
lineup_pres_list <- list(A, B, C)
rm(A, B, C)


#Target absent data:
A <- rep(6:1, length.out = 100)
B <- rep(5:1, length.out = 70)
C <- rep(4:1, length.out = 20)
lineup_abs_list <- list(A, B, C)
rm(A, B, C)

# Suspect position for each TP/TA pair
pos_list <- c(3, 2, 1)

#Nominal size:
k <- c(6, 5, 4)

#Use diag param helper function to get data (n11, n21, n12, n22):
linedf <- diag_param(lineup_pres_list, lineup_abs_list, pos_list, k)
#Call:
var <- var_lnd(linedf)
```

Index

∗ **datasets**
line73, 116
lineup_example, 119
mickwick, 170
mockdata, 170
nortje2012, 171

allfoil_cihigh, 6
allfoilbias, 7
allprop, 7, *126*
available_detectors, 8, *102*
available_models, 9, *102*

batch_embeddings, 10, *187*
binom.test, *94*
boot, *60*, *65*, *72*, *92*, *95*, *118*, *121*
boot_winter_2ht, 11, *178*
bootstrap_anri, 13

calibration_bayes, 14
check_python_deps, 17
check_pywitness_deps, 18
chi_diag, 18
classify_lineup_choices, 20, *119, 120*, *222*
compare_anri, 22
compare_dpp, 24
compare_eff_sizes.boot, 25, *70*
compare_max_sdt, 27, *80, 81*
compare_models, 28
compare_multid, 30, *154*
compare_pauc, 31, *224, 225*
compare_sdt_summary, 33, *217, 218*
compare_utility, 35
compute_anri, 36
compute_dpp, 38
compute_eig, 39
cosine_to_similarity, 41
create_example_lineup_data, 42

d_bar, 43
d_weights, 44
datacheck1, 46
datacheck2, 46
datacheck3, 47
detect_faces, 48
diag_param, 49
diag_ratio_T, 51, *53*
diag_ratio_T_bayes, 52, *55*, *218*
diag_ratio_T_bayes_compare, *53*, 54
diag_ratio_W, 55
display_lineup, 56
dmax_filler, 57

eff_size_per_foils, 58
effsize_compare, *26*, 59
embedding_distance, 60
esize_boot_dist, 61, *69*, *71, 72*, *189*
esize_m, 63, *126*
esize_m_boot, 64
esize_T, *62, 63*, 65, *69*, *73*, *110, 111*, *126*
esize_T_bayes, *16*, *53*, 67, *70*, *93*
esize_T_bayes_compare, 69, *69*
esize_T_boot, *66*, 71
esize_T_ci_n, *66*, *69*, 72
estimate_msdt_params, 73, *81*
extract_sdt_metrics, 74

face_similarity, 75
fit_lineup_models, 76, *114*
fit_max_sdt, *27*, *74*, 78
fit_sdt_glm, 82
fit_sdt_glmm, 83
fit_sdt_roc, 84
fit_winter_2ht, *74*, 87
format_comparison_table, 89
func_size, 90, *93*
func_size.boot, 91
func_size_bayes, 92
func_size_report, 94

gen_boot_propci, 95
gen_boot_propmean_se, 96
gen_boot_samples, 97
gen_boot_samples_list, 98
gen_esize_m, 98
gen_esize_m_ci, 99
gen_lineup_prop, 100
gen_linevec, 101
get_embedding, 102
get_face_landmarks, 103, *105*
get_key_landmarks, 105

has_valid_face, 106
homog_diag, 106, *109*
homog_diag_boot, 108

i_esize_T, *66*, 110
innocent_id_rate_effective, 111
innocent_id_rate_nominal, 112
innocent_id_rate_uncorrected, 113
install_pywitness, 114
install_r4lineups_python, 115

line73, 116
lineup_bias_boot_dist, 117, *191*
lineup_boot_allprop, *58*, 118
lineup_example, 119
lineup_pairwise_matrix, 120
lineup_prop_boot, 121
lineup_prop_tab, 122
lineup_prop_vec, 123
lineup_similarity, 123, *192*
lineup_table, *21*, *120*, 125
ln_diag_ratio, 126

make_bayes_curves, 128
make_bree_curve, 130
make_cac, *21*, 132, *161*, *195*
make_cac_gg, 133, *195*
make_cacdata, *133*, 134, *195*
make_calibration, 136
make_calibration_by_condition, 137
make_calibration_by_condition_gg, 139
make_calibration_data, *13*, *15*, *16*, *22*, *36*, *37*, *136*, *138*, 140
make_calibration_gg, 142
make_dpp, *24*, *39*, 143
make_eig, 145
make_eig_data, 147
make_fullroc, 148, *195*
make_fullroc_data, 150, *195*
make_multid, *30*, 152, *174*
make_ppv_range, 155, *222*
make_rac, 156
make_rac_gg, 157
make_racdata, 158
make_roc, *21*, 160, *195*
make_roc_gg, 161, *195*
make_rocdata, *143*, *144*, *153*, 162, *162*, *165*, *195*
make_utility_curves, 164
make_utility_difference, 166
makevec_prop, 168
max_filler_moments, 169
mickwick, 170
mockdata, 170
msdt_dprime_from_rej *(*estimate_msdt_params*)*, 73
msdt_gamma_from_rej *(*estimate_msdt_params*)*, 73

nortje2012, 171

plot.esize_T_bayes, 171
plot.esize_T_bayes_compare, 172
plot.lineup_cac *(*plot_roc_cac*)*, 195
plot.lineup_cac_data *(*plot_roc_cac*)*, 195
plot.lineup_fullroc *(*plot_roc_cac*)*, 195
plot.lineup_fullroc_data *(*plot_roc_cac*)*, 195
plot.lineup_model_comparison, 172
plot.lineup_roc *(*plot_roc_cac*)*, 195
plot.lineup_roc_data *(*plot_roc_cac*)*, 195
plot.model_comparison, 173
plot.multid, *154*, 173
plot.pauc_comparison, 174
plot.power_analysis, 175
plot.sdt_roc_fit, 175
plot.winter_2ht, 176
plot.winter_2ht_boot, 176
plot_2ht_fit, 177
plot_2ht_parameters, 177
plot_anri_comparison, 178
plot_anri_difference_distribution, 179
plot_bayes_information_gain, 180
plot_bayes_prior_posterior, 181
plot_bree, 182
plot_dpp, 183

plot_dpp_comparison, 184
plot_effective_size_conf, 184
plot_eig, 185
plot_eig_posteriors, 186
plot_embedding_space, 187
plot_error_rate_conf, 188
plot_esize_distribution, 189
plot_fullroc, *162*, 190, *195*
plot_lineup_bias_distribution, *117*, *189*, 191
plot_lineup_similarity, 192
plot_msdt_joint, 193
plot_ppv_range, 194
plot_roc_cac, 195
plot_similarity_distribution, 196
plot_utility_curves, 197
plot_utility_difference, 197
pmax_filler *(*dmax_filler*)*, 57
ppv_by_confidence, 198
ppv_range_by_confidence, 201
print.esize_T_bayes, 202
print.esize_T_bayes_compare, 203
print.lineup_anri, 203
print.lineup_anri_comparison, 204
print.lineup_bayes_curves, 204
print.lineup_cac, 205
print.lineup_calibration, 205
print.lineup_data, 206
print.lineup_dpp, 206
print.lineup_dpp_comparison, 207
print.lineup_eig, 207
print.lineup_fullroc, 208
print.lineup_ppv, 208
print.lineup_ppv_range, 209
print.lineup_rac, 209
print.lineup_roc, 210
print.lineup_similarity, 210
print.lineup_table, 211
print.lineup_utility_comparison, 211
print.model_comparison, 212
print.pauc_comparison, 212
print.sdt_roc_fit, 213
print.simulated_lineup_data, 213
print.winter_2ht, 214
print.winter_2ht_boot, 214

qmax_filler *(*dmax_filler*)*, 57

rep_index, 215
rmax_filler *(*dmax_filler*)*, 57
rot_vector, 215
run_r4lineups_app, 216

sdt_compare, *81*, 216
sdt_summary_from_counts, *153, 154*, 218, *218*
sdt_summary_variance, 219
simulate_lineup_data, *21*, *120*, 220, *225*
simulate_power_analysis, 223
standardize_lineup_data, 226
summary.lineup_eig, 228
summary.lineup_similarity, 228
summary.model_comparison, 229
summary.pauc_comparison, 229
summary.sdt_roc_fit, 230
summary.winter_2ht, 230
summary.winter_2ht_boot, 231

validate_lineup_data, 231
var_diag_ratio, 233
var_lnd, 234