

Package ‘phontrast’

October 7, 2026

Title Contrast and Separation Metrics for Phonological Categories

Version 2.5.0

Author Grant M. Berry [aut, cre]

Maintainer Grant M. Berry <berry.grant@gmail.com>

Description Computes and compares multiple measures of separation and overlap between phonological categories (for example vowels or consonants) in arbitrary multi-dimensional acoustic spaces such as formant values, mel-frequency cepstral coefficients (MFCCs), duration, or learned embeddings. The main entry point, `phontrast()`, reports several contrast metrics in one call -- Jensen-Shannon divergence and distance (Lin, 1991) <doi:10.1109/18.61115>, the Pillai-Bartlett trace, Bhattacharyya distance and affinity, Mahalanobis distance, and proportional overlap -- globally or by group on a common separation-oriented scale, with bootstrap confidence intervals. `rank_contrasts()` implements a measurement protocol for ranking speakers' contrasts by Jensen-Shannon distance and checking the ranking against Pillai, with sample-size licensing and a bandwidth-sensitivity check. Also provides utilities for preparing estimates for downstream modelling such as generalized additive models and mixed-effects models. Formerly released as 'phonJSD'.

License MIT + file LICENSE

URL <https://github.com/berrygrant/phontrast>

BugReports <https://github.com/berrygrant/phontrast/issues>

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports ks, dplyr, purrr, tibble, rlang, stats, grDevices, utils

Suggests ggplot2, mgcv, mvtnorm, tuneR, testthat (>= 3.0.0), knitr, rmarkdown

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2026-10-07 14:40:02 UTC

Contents

bhattacharyya_mvnorm	3
boot_jsd	3
estimate_bhatt	5
estimate_jsd	5
estimate_overlap	8
estimate_pillai	9
extract_mfcc	10
global_boot_jsd	11
global_pillai	13
hier_boot_jsd_model	14
inspect_contrast	15
jsd	17
jsd_kde_nd	17
jsd_summary	20
kl_div	21
percentile_rank	22
percent_overlap_kde	23
phontrast	24
phontrast_palette	28
pillai_overlap	28
plot.phontrast_contrast	30
plot.phontrast_ranking	31
plot_category_pca	31
plot_category_space	33
plot_contrast	34
plot_overlap_metrics	36
plot_rank_agreement	38
prepare_jsd_beta	39
protocol_floors	40
rank_contrasts	41
recommended_estimator	43
scale_colour_phontrast	44
speaker_bhatt	45
speaker_jsd	46
speaker_pillai	47
theme_phontrast	47
vowel_cohort	48

Index

50

bhattacharyya_mvnorm	<i>Bhattacharyya distance and affinity under multivariate normality</i>
----------------------	---

Description

Estimates means and covariances per category and computes Bhattacharyya distance and affinity ($\exp(-\text{distance})$) under the assumption of multivariate normality.

Usage

```
bhattacharyya_mvnorm(data, features, category_col, eps = 1e-06)
```

Arguments

data	Data frame.
features	Character vector of numeric feature columns.
category_col	String; column with exactly two categories.
eps	Small ridge constant added to covariance matrices to improve numerical stability.

Value

A list with distance and affinity ($\exp(-\text{distance})$).

boot_jsd	<i>Bootstrap JSD for each group</i>
----------	-------------------------------------

Description

Computes bootstrap mean, SD, and confidence interval for JSD within each group (e.g., speaker), using resampling with replacement.

Usage

```
boot_jsd(
  data,
  group_col,
  category_col,
  features,
  n_boot = 1000,
  min_tokens = 20,
  est_distance = FALSE,
  conf_level = 0.95,
  bw = c("Hpi", "Hscv", "Hpi.diag", "scott.diag"),
  eval_on = c("pooled", "group1", "group2", "pooled_sample"),
```

```

    eval_n = NULL,
    eval_seed = NULL,
    engine = c("ks", "fast_diag", "fast_diagonal"),
    chunk_size = 1000L,
    ...
  )

```

Arguments

<code>data</code>	Data frame containing acoustic measurements.
<code>group_col</code>	Character vector giving one or more grouping columns (e.g., "speaker" or c("Sex", "Style")).
<code>category_col</code>	String: name of column giving the category to compare (e.g., "vowel"). Each group must have exactly two categories.
<code>features</code>	Character vector of column names giving the acoustic space.
<code>n_boot</code>	Number of bootstrap resamples per group.
<code>min_tokens</code>	Minimum number of tokens per group required to compute JSD. Groups with fewer tokens are dropped.
<code>est_distance</code>	Logical; if TRUE, return Jensen-Shannon distance (sqrt of divergence) instead of divergence.
<code>conf_level</code>	Confidence level for bootstrap intervals.
<code>bw</code>	Bandwidth selection method passed to <code>jsd_kde_nd()</code> .
<code>eval_on</code>	KDE evaluation points passed to <code>jsd_kde_nd()</code> .
<code>eval_n</code>	Optional maximum number of KDE evaluation points.
<code>eval_seed</code>	Optional integer seed for KDE evaluation-point subsampling.
<code>engine</code>	KDE evaluation engine passed to <code>jsd_kde_nd()</code> . "fast_diagonal" is accepted as an alias for "fast_diag".
<code>chunk_size</code>	Chunk size for engine = "fast_diag".
<code>...</code>	Additional arguments passed to <code>jsd_kde_nd()</code> .

Value

A tibble with one row per group and columns: `group`, `n_tokens`, `n_boot`, `conf_level`, `jsd_mean`, `jsd_sd`, `ci_lower`, `ci_upper`, `jsd_low`, and `jsd_high`. `jsd_low` and `jsd_high` are retained as legacy aliases for `ci_lower` and `ci_upper`.

estimate_bhatt	<i>Estimate Bhattacharyya distance, globally or by group</i>
----------------	--

Description

Unified front-end for Bhattacharyya distance (and affinity) under a multivariate normal approximation.

Usage

```
estimate_bhatt(
  data,
  features,
  category_col,
  group_col = NULL,
  min_tokens = 20,
  eps = 1e-06
)
```

Arguments

data	Data frame.
features	Character vector of numeric feature columns.
category_col	String; category column name (exactly two levels globally).
group_col	Optional character vector of one or more grouping columns. If NULL, a single global Bhattacharyya distance is returned.
min_tokens	Minimum tokens (globally or per group).
eps	Small ridge constant passed to <code>bhattacharyya_mvnorm()</code> .

Value

Data frame with either one global row or one row per group.

estimate_jsd	<i>Estimate Jensen-Shannon divergence or distance between two categories</i>
--------------	--

Description

Use this function when Jensen-Shannon divergence (JSD) or Jensen-Shannon distance is the primary outcome. If you want to compare JSD with Pillai, Bhattacharyya, Mahalanobis, and percent-overlap metrics, start with `phontrast()` instead.

Usage

```

estimate_jsd(
  data,
  features,
  category_col,
  group_col = NULL,
  do_boot = FALSE,
  n_boot = 1000,
  min_tokens = 20,
  est_distance = FALSE,
  conf_level = 0.95,
  bw = c("Hpi", "Hscv", "Hpi.diag", "scott.diag"),
  eval_on = c("pooled", "group1", "group2", "pooled_sample"),
  eval_n = NULL,
  eval_seed = NULL,
  engine = c("ks", "fast_diag", "fast_diagonal"),
  chunk_size = 1000L,
  method = c("mc", "legacy"),
  density = c("kde", "mvnrm"),
  mc_n = 10000L,
  bw_scale = 1,
  ...
)

```

Arguments

<code>data</code>	Data frame with at least <code>category_col</code> and <code>features</code> .
<code>features</code>	Character vector of feature column names (e.g., <code>c("F1", "F2")</code>).
<code>category_col</code>	Name of the column giving the two-way category factor.
<code>group_col</code>	Optional character vector of one or more grouping columns. If provided, returns per-group JSD. Multiple grouping columns are combined into a labeled group value such as <code>"Sex=F Style=read"</code> .
<code>do_boot</code>	Logical; if TRUE, run nonparametric bootstrap.
<code>n_boot</code>	Number of bootstrap resamples.
<code>min_tokens</code>	Minimum total tokens required (globally or per group).
<code>est_distance</code>	Logical; if TRUE, return Jensen-Shannon <i>distance</i> (sqrt of divergence).
<code>conf_level</code>	Confidence level for bootstrap interval.
<code>bw</code>	Bandwidth selection method passed to <code>jsd_kde_nd()</code> .
<code>eval_on</code>	KDE evaluation points passed to <code>jsd_kde_nd()</code> .
<code>eval_n</code>	Optional maximum number of KDE evaluation points.
<code>eval_seed</code>	Optional integer seed for KDE evaluation-point subsampling.
<code>engine</code>	KDE evaluation engine passed to <code>jsd_kde_nd()</code> . <code>"fast_diagonal"</code> is accepted as an alias for <code>"fast_diag"</code> .
<code>chunk_size</code>	Chunk size for <code>engine = "fast_diag"</code> .

method	Estimator passed to <code>jsd_kde_nd()</code> : "mc" (default) for the Monte-Carlo plug-in estimate of the continuous JSD, or "legacy" to reproduce the pre-1.2.0 self-normalized sample-point estimate. Ignored when <code>density = "mvnorm"</code> .
density	Density model behind the estimate, passed to <code>jsd_kde_nd()</code> : "kde" (default) estimates each category's density by kernel density estimation; "mvnorm" fits one multivariate normal per category and estimates the continuous JSD between the two Gaussians by Monte-Carlo. Under "mvnorm" the KDE-specific arguments do not apply and the Monte-Carlo sample size is set by <code>mc_n</code> .
mc_n	Positive integer; number of Monte-Carlo samples drawn from each fitted Gaussian when <code>density = "mvnorm"</code> (default 10000). Ignored when <code>density = "kde"</code> .
bw_scale	Positive number multiplying the selected kernel bandwidth on the standard-deviation scale (default 1); 0.5 and 2 give the halved and doubled bandwidths of the smoothing-sensitivity check. Passed to <code>jsd_kde_nd()</code> ; ignored when <code>density = "mvnorm"</code> .
...	Additional arguments passed to <code>jsd_kde_nd()</code> (e.g., <code>loo</code>).

Details

JSD is bounded from 0 to 1 for two equally weighted distributions. Larger values indicate greater category separation. Jensen-Shannon distance is $\sqrt{\text{JSD}}$ and has the same direction of interpretation.

Value

A tibble. Global: one row with columns `scope`, `n_tokens`, `n_boot`, `conf_level`, `jsd_point`, `jsd_mean`, `jsd_sd`, `ci_lower`, `ci_upper`, `jsd_low`, `jsd_high`. Grouped: one row per group with columns `scope`, `group`, `n_tokens`, `n_boot`, `conf_level`, `jsd_point`, `jsd_mean`, `jsd_sd`, `ci_lower`, `ci_upper`, `jsd_low`, `jsd_high`. `ci_lower` and `ci_upper` are the preferred confidence interval columns; `jsd_low` and `jsd_high` are retained as legacy aliases.

Examples

```
set.seed(2026)
vowels <- data.frame(
  speaker = rep(c("s01", "s02"), each = 60),
  vowel = rep(rep(c("ih", "eh"), each = 30), 2),
  f1 = c(
    rnorm(30, 500, 55), rnorm(30, 560, 60),
    rnorm(30, 510, 60), rnorm(30, 575, 65)
  ),
  f2 = c(
    rnorm(30, 1980, 150), rnorm(30, 1880, 155),
    rnorm(30, 1960, 160), rnorm(30, 1840, 165)
  )
)

# Point estimate of JSD (fast), globally and by speaker.
estimate_jsd(
  data = vowels,
```

```

    features = c("f1", "f2"),
    category_col = "vowel"
  )
  estimate_jsd(vowels, c("f1", "f2"), "vowel", group_col = "speaker")

# Bootstrap confidence intervals, shown on a single feature: multivariate
# bootstraps work the same way but repeat multivariate bandwidth selection
# on every resample, so they take correspondingly longer. Increase n_boot
# for real analyses.
estimate_jsd(vowels, "f1", "vowel", do_boot = TRUE, n_boot = 20)

```

estimate_overlap	<i>Estimate proportional overlap globally or by group</i>
------------------	---

Description

Unified front-end for KDE-based proportional overlap between two categories. The returned overlap column is a 0–1 proportion, not a 0–100 percentage.

Usage

```

estimate_overlap(
  data,
  features,
  category_col,
  group_col = NULL,
  min_tokens = 20,
  bw = c("Hpi", "Hscv", "Hpi.diag", "scott.diag"),
  eval_on = c("pooled", "group1", "group2", "pooled_sample"),
  eval_n = NULL,
  eval_seed = NULL,
  engine = c("ks", "fast_diag", "fast_diagonal"),
  chunk_size = 1000L,
  method = c("mc", "legacy"),
  density = c("kde", "mvnorm"),
  mc_n = 10000L,
  bw_scale = 1,
  ...
)

```

Arguments

<code>data</code>	Data frame with at least <code>category_col</code> and <code>features</code> .
<code>features</code>	Character vector of feature column names (e.g., <code>c("F1", "F2")</code>).
<code>category_col</code>	Name of the column giving the two-way category factor.
<code>group_col</code>	Optional character vector of one or more grouping columns. If provided, returns per-group overlap. Multiple grouping columns are combined into a labeled group value such as <code>"Sex=F Style=read"</code> .

min_tokens	Minimum total tokens required (globally or per group).
bw	Bandwidth selection method passed to percent_overlap_kde().
eval_on	KDE evaluation points passed to percent_overlap_kde().
eval_n	Optional maximum number of KDE evaluation points.
eval_seed	Optional integer seed for KDE evaluation-point subsampling.
engine	KDE evaluation engine passed to percent_overlap_kde(). "fast_diagonal" is accepted as an alias for "fast_diag".
chunk_size	Chunk size for engine = "fast_diag".
method	Estimator passed to percent_overlap_kde(): "mc" (default) or "legacy" (pre-1.2.0 self-normalized estimate). Ignored when density = "mvnorm".
density	Density model passed to percent_overlap_kde(): "kde" (default) or "mvnorm" (fit one multivariate normal per category and estimate the overlapping coefficient between the two Gaussians by Monte-Carlo).
mc_n	Positive integer; number of Monte-Carlo samples drawn from each fitted Gaussian when density = "mvnorm" (default 10000). Ignored when density = "kde".
bw_scale	Positive bandwidth multiplier passed to percent_overlap_kde() (default 1); see jsd_kde_nd().
...	Additional arguments passed to percent_overlap_kde().

Value

A tibble (global = one row; grouped = one per group) with overlap as a 0–1 proportion.

estimate_pillai	<i>Estimate Pillai trace, globally or by group</i>
-----------------	--

Description

Unified front-end for Pillai-Bartlett trace. If group_col is NULL, computes a single global Pillai trace for the full dataset. If group_col is provided, computes Pillai per group (e.g., per speaker).

Usage

```
estimate_pillai(
  data,
  features,
  category_col,
  group_col = NULL,
  min_tokens = 20
)
```

Arguments

data	Data frame.
features	Character vector of numeric feature columns.
category_col	String; category column name.
group_col	Optional character vector of one or more grouping columns. If NULL, a global Pillai value is returned.
min_tokens	Minimum tokens (globally or per group).

Value

A data frame with either one global row or one row per group.

extract_mfcc	<i>Extract MFCCs for vowel segments</i>
--------------	---

Description

Adds MFCC feature columns (e.g., mfcc1..mfcc13) to a data frame that contains audio file paths and (optionally) segment boundaries.

Usage

```
extract_mfcc(
  data,
  file_col,
  start_col = NULL,
  end_col = NULL,
  fs = NULL,
  numcep = 13,
  prefix = "mfcc",
  strict = FALSE,
  warn = TRUE,
  ...
)
```

Arguments

data	Data frame containing audio paths and segment boundaries.
file_col	String; column name containing WAV file paths.
start_col	Optional string; column name containing segment start time (in seconds). If NULL, the full file is used.
end_col	Optional string; column name containing segment end time (in seconds). If NULL, the full file is used.
fs	Optional numeric; override sampling rate (Hz). If NULL, uses the file's sampling rate.

numcep	Integer; number of MFCC coefficients to return.
prefix	String; prefix for output columns (default "mfcc").
strict	Logical; if TRUE, stop on the first row that cannot be processed. If FALSE (default), leave that row's MFCC values as NA.
warn	Logical; if TRUE (default), warn when one or more rows cannot be processed and strict = FALSE.
...	Additional arguments passed to tuneR::melfcc().

Details

This function uses **tuneR** to read WAV files and compute MFCCs. The package is optional; if it is not installed, an informative error is raised.

Value

The input data frame with added MFCC columns.

Examples

```
if (requireNamespace("tuneR", quietly = TRUE)) {
  wav_path <- tempfile(fileext = ".wav")
  wave <- tuneR::sine(
    freq = 440, duration = 0.25, samp.rate = 16000, xunit = "time"
  )
  tuneR::writeWave(wave, wav_path)

  segments <- data.frame(wav_path = wav_path)
  mfcc <- extract_mfcc(segments, file_col = "wav_path", numcep = 3)
  unlink(wav_path)
  mfcc[, c("mfcc1", "mfcc2", "mfcc3")]
}
```

global_boot_jsd

Global JSD with bootstrap confidence interval

Description

Computes a single Jensen-Shannon divergence (JSD) value for two categories in an n-dimensional acoustic space, together with bootstrap-based confidence intervals obtained by resampling tokens with replacement.

Usage

```
global_boot_jsd(
  data,
  features,
  category_col,
```

```

n_boot = 1000,
min_tokens = 20,
est_distance = FALSE,
conf_level = 0.95,
bw = c("Hpi", "Hscv", "Hpi.diag", "scott.diag"),
eval_on = c("pooled", "group1", "group2", "pooled_sample"),
eval_n = NULL,
eval_seed = NULL,
engine = c("ks", "fast_diag", "fast_diagonal"),
chunk_size = 1000L,
method = c("mc", "legacy"),
density = c("kde", "mvnorm"),
mc_n = 10000L,
...
)

```

Arguments

<code>data</code>	Data frame containing at least the category column and the feature columns.
<code>features</code>	Character vector of column names giving the acoustic dimensions (e.g., <code>c("f1", "f2")</code>) or <code>paste0("mfcc", 1:13)</code>).
<code>category_col</code>	String; name of the column giving the two categories to compare (e.g., "vowel"). Must have exactly two unique values.
<code>n_boot</code>	Integer; number of bootstrap resamples.
<code>min_tokens</code>	Minimum total number of non-missing tokens required.
<code>est_distance</code>	Logical; if TRUE, return Jensen-Shannon distance (sqrt of divergence) instead of divergence.
<code>conf_level</code>	Confidence level for bootstrap intervals.
<code>bw</code>	Bandwidth selection method passed to <code>jsd_kde_nd()</code> .
<code>eval_on</code>	KDE evaluation points passed to <code>jsd_kde_nd()</code> .
<code>eval_n</code>	Optional maximum number of KDE evaluation points.
<code>eval_seed</code>	Optional integer seed for KDE evaluation-point subsampling.
<code>engine</code>	KDE evaluation engine passed to <code>jsd_kde_nd()</code> . "fast_diagonal" is accepted as an alias for "fast_diag".
<code>chunk_size</code>	Chunk size for engine = "fast_diag".
<code>method</code>	Estimator passed to <code>jsd_kde_nd()</code> : "mc" (default) or "legacy" (pre-1.2.0 self-normalized estimate). Ignored when density = "mvnorm".
<code>density</code>	Density model passed to <code>estimate_jsd()</code> : "kde" (default) or "mvnorm" (fit one multivariate normal per category and estimate JSD between the two Gaussians by Monte-Carlo).
<code>mc_n</code>	Positive integer; number of Monte-Carlo samples drawn from each fitted Gaussian when density = "mvnorm" (default 10000). Ignored when density = "kde".
<code>...</code>	Additional arguments passed to <code>jsd_kde_nd()</code> .

Details

This is the "group-wise" version of JSD: it ignores speakers and treats all tokens as coming from a single population for each category.

Value

A one-row data frame with columns:

- `n_tokens` - total number of tokens used
- `n_boot` - number of successful bootstrap samples
- `conf_level` - confidence level used for the interval
- `jsd_point` - JSD on the full dataset
- `jsd_mean` - mean JSD across bootstrap samples
- `jsd_sd` - standard deviation of bootstrap JSD
- `ci_lower`, `ci_upper` - bootstrap confidence interval
- `jsd_low`, `jsd_high` - legacy aliases for `ci_lower` and `ci_upper`

`global_pillai`

Global Pillai trace (point estimate)

Description

Computes a single Pillai-Bartlett trace for the full dataset in the specified feature space, returning a one-row data frame that includes the total number of tokens used.

Usage

```
global_pillai(data, features, category_col, min_tokens = 20)
```

Arguments

<code>data</code>	Data frame.
<code>features</code>	Character vector of numeric feature columns.
<code>category_col</code>	String; column with category labels (≥ 2 levels).
<code>min_tokens</code>	Minimum total tokens required after removing missing values.

Value

A one-row data frame with columns: `n_tokens`, `pillai`, and `p_value`.

hier_boot_jsd_model *Hierarchical bootstrap for JSD-based models*

Description

Performs a hierarchical bootstrap: resample groups with replacement, resample tokens within each sampled group, compute JSD per group, fit a model to the bootstrap JSD values, and repeat.

Usage

```
hier_boot_jsd_model(
  data,
  group_col,
  category_col,
  features,
  formula,
  fit_fun = NULL,
  n_outer = 200,
  min_tokens = 20,
  eps = 1e-06,
  progress = TRUE,
  ...
)
```

Arguments

data	Data frame with at least: group_col, category_col, features, and any predictors used in the model.
group_col	String: grouping variable (e.g., "speaker").
category_col	String: category variable with 2 levels (e.g., "vowel").
features	Character vector of acoustic feature columns.
formula	Model formula to pass to fit_fun (e.g., jsd_beta ~ s(age) + s(region, bs = "re")).
fit_fun	A function that takes (formula, data, ...) and returns a fitted model. Defaults to mgcv::gam if available, otherwise stats::lm.
n_outer	Number of hierarchical bootstrap replicates.
min_tokens	Minimum within-group tokens required.
eps	Small epsilon for bounding JSD in (0, 1) if using Beta family.
progress	Logical; if TRUE, prints progress every 10 replicates.
...	Additional arguments passed to fit_fun.

Details

This lets you propagate measurement uncertainty in JSD into model parameters (e.g., GAM/LMM coefficients).

Value

A tibble with columns:

- `boot_id` - bootstrap replicate index
- `term` - model term
- `estimate` - estimate for that term in that replicate

Examples

```
set.seed(2026)
speakers <- paste0("s", 1:4)
dat <- data.frame(
  speaker = rep(speakers, each = 60),
  age = rep(c(22, 35, 48, 61), each = 60),
  vowel = rep(rep(c("ih", "eh"), each = 30), 4)
)
dat$f1 <- rnorm(
  nrow(dat),
  mean = ifelse(dat$vowel == "ih", 500, 560) + dat$age * 0.3,
  sd = 55
)

hier_boot_jsd_model(
  data = dat,
  group_col = "speaker",
  category_col = "vowel",
  features = "f1",
  formula = jsd_beta ~ age,
  fit_fun = stats::lm,
  n_outer = 3,
  min_tokens = 20,
  progress = FALSE
)
```

inspect_contrast

Inspect one speaker's contrast across bandwidths

Description

Redraws a single speaker from a `rank_contrasts()` ranking with `plot_contrast()`'s distribution-aware layers, one panel per kernel bandwidth, so a flagged or set-aside speaker can be inspected the way the protocol asks: at half, at the selected, and at twice the diagonal Scott bandwidth (the smoothing used by the bandwidth check). Each panel is labelled with the Jensen-Shannon distance and shared mass at that bandwidth and with the speaker's Pillai trace, which takes no bandwidth; the subtitle restates the reported ranks, the rank difference, and the outcome of the bandwidth check. The tokens come from the ranking itself (`attr(ranking, "protocol")$data`), so no further data is needed.

Usage

```
inspect_contrast(
  ranking,
  group,
  features = NULL,
  bw_scales = c(0.5, 1, 2),
  levels = c(0.5, 0.8, 0.95),
  points = TRUE,
  overlap = TRUE,
  grid_n = NULL,
  point_alpha = 0.55,
  point_size = 1.6,
  reverse_x = FALSE,
  reverse_y = FALSE
)
```

Arguments

ranking	A phontrast_ranking object from rank_contrasts().
group	String; the speaker (a value of ranking\$group) to inspect.
features	Features to display: by default the ranking's own features, which must number one or two. For a ranking in more than two dimensions, name two of its features here; the annotated metrics are still computed on all of the ranking's features.
bw_scales	Positive bandwidth multipliers, one panel each (default c(0.5, 1, 2)).
levels, points, overlap, grid_n, point_alpha, point_size, reverse_x, reverse_y	As in plot_contrast().

Value

A **ggplot2** plot object. The per-panel metrics are attached as attr(p, "inspect_metrics").

See Also

rank_contrasts(), plot_rank_agreement(), plot_contrast().

Examples

```
set.seed(2026)
gaps <- c(30, 60, 90, 120, 150, 180, 210, 240)
cohort <- do.call(rbind, lapply(seq_along(gaps), function(i) {
  data.frame(
    speaker = sprintf("s%02d", i),
    vowel = rep(c("ih", "eh"), each = 50),
    f1 = c(rnorm(50, 500, 55), rnorm(50, 500 + gaps[i], 55)),
    f2 = c(rnorm(50, 1900, 120), rnorm(50, 1900 - gaps[i], 120))
  )
}))
ranking <- rank_contrasts(cohort, c("f1", "f2"), "vowel", "speaker",
```

```

      bw_check = FALSE)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  inspect_contrast(ranking, "s04")
}

```

jsd

*Jensen-Shannon divergence for discrete distributions***Description**

Computes JSD(p, q) in bits for discrete probability vectors. The value is bounded in $[0, 1]$ for equally weighted mixtures.

Usage

```
jsd(p, q)
```

Arguments

p, q Numeric probability vectors of the same length.

Value

A single numeric value: the Jensen-Shannon divergence in bits.

jsd_kde_nd

*n-dimensional JSD via multivariate kernel density estimation***Description**

Computes Jensen-Shannon divergence between two categories in an arbitrary n-dimensional acoustic space using multivariate KDE. The default engine uses the **ks** package; a faster diagonal-Gaussian engine is available for diagonal bandwidths.

Usage

```

jsd_kde_nd(
  data,
  features,
  group = "category",
  bw = c("Hpi", "Hscv", "Hpi.diag", "scott.diag"),
  eval_on = c("pooled", "group1", "group2", "pooled_sample"),
  eval_n = NULL,
  eval_seed = NULL,
  engine = c("ks", "fast_diag", "fast_diagonal"),
  chunk_size = 1000L,

```

```

method = c("mc", "legacy"),
density = c("kde", "mvnorm"),
mc_n = 10000L,
loo = TRUE,
bw_scale = 1
)

```

Arguments

<code>data</code>	A data frame containing observations from exactly two categories.
<code>features</code>	Character vector of column names giving the acoustic dimensions (e.g., MFCC1..MFCC13, F1/F2/duration).
<code>group</code>	String: name of the column giving the category labels (e.g., "vowel", "segment"). Must have exactly two unique values in data.
<code>bw</code>	Bandwidth selection method. One of "Hpi", "Hscv", "Hpi.diag", or "scott.diag". The first three are passed to <code>ks::Hpi()</code> , <code>ks::Hscv()</code> , or <code>ks::Hpi.diag()</code> for multivariate inputs. "scott.diag" uses a diagonal Scott rule-of-thumb bandwidth matrix. For one-dimensional inputs, these map to <code>stats::bw.SJ()</code> , <code>stats::bw.ucv()</code> , <code>stats::bw.nrd0()</code> , and Scott's rule, respectively, with a robust fallback for constant samples.
<code>eval_on</code>	Where to evaluate the KDEs (method = "legacy" only). "pooled" (default) evaluates on all observations from both categories; "group1" or "group2" evaluate on the respective group only. "pooled_sample" evaluates on a sampled subset of pooled observations and requires <code>eval_n</code> . Ignored when method = "mc" (which always evaluates each category at its own observations).
<code>eval_n</code>	Optional positive integer giving the maximum number of evaluation points to use. If supplied, evaluation points are sampled from the set chosen by <code>eval_on</code> .
<code>eval_seed</code>	Optional integer seed used only when <code>eval_n</code> causes evaluation-point subsampling. If NULL, the current R random-number state is used.
<code>engine</code>	KDE evaluation engine. "ks" uses <code>ks::kde()</code> . "fast_diag" uses a chunked diagonal-Gaussian evaluator and requires <code>bw = "scott.diag"</code> or <code>bw = "Hpi.diag"</code> for multivariate KDE. "fast_diagonal" is accepted as an alias for "fast_diag".
<code>chunk_size</code>	Positive integer controlling the number of evaluation points processed per chunk by engine = "fast_diag".
<code>method</code>	Estimator: "mc" (default) for the Monte-Carlo plug-in estimate of the continuous JSD, or "legacy" for the pre-1.2.0 self-normalized sample-point index. Ignored when density = "mvnorm".
<code>density</code>	Density model behind the estimate: "kde" (default) estimates each category's density by kernel density estimation; "mvnorm" fits one multivariate normal per category and estimates the continuous JSD between the two Gaussians by Monte-Carlo (no closed form exists). Under "mvnorm" the KDE-specific arguments (<code>bw</code> , <code>engine</code> , <code>eval_on</code> , <code>chunk_size</code> , <code>method</code> , <code>eval_n</code> , <code>loo</code>) do not apply; the Monte-Carlo sample size is set by <code>mc_n</code> and <code>eval_seed</code> makes the draw reproducible.

mc_n	Positive integer; number of Monte-Carlo samples drawn from each fitted Gaussian when density = "mvnrm" (default 10000). The estimator draws mc_n fresh points from each category's fitted Gaussian and averages the log density ratio, so it targets the JSD between the two fitted Gaussians rather than a re-substitution estimate at the observed points. Larger values reduce Monte-Carlo variance. Ignored when density = "kde".
loo	Logical; if TRUE (default) the Monte-Carlo estimator uses a partial leave-one-out correction on each category's self-density to reduce resubstitution bias. The correction removes a sample-size-scaled fraction $n / (n + 20)$ of each point's own kernel: half at 20 tokens per category (the min_tokens default), approaching the full leave-one-out correction as the category grows. Removing only part of the self-kernel keeps the corrected density strictly positive at isolated points, so small but real divergences remain small positive values rather than being floored to exactly 0 (as the full leave-one-out correction did through phontrast 2.0.2). Ignored when method = "legacy".
bw_scale	Positive number multiplying the selected kernel bandwidth on the standard-deviation scale: univariate bandwidths are multiplied by bw_scale and bandwidth matrices by bw_scale^2. The default 1 uses the selected bandwidth unchanged; 0.5 and 2 give the halved and doubled bandwidths of the smoothing-sensitivity check in rank_contrasts(). Ignored when density = "mvnrm".

Details

By default (method = "mc") JSD is estimated with a Monte-Carlo plug-in: each category's KDE is evaluated at that category's own observations and the log density ratio against the mixture is averaged. This is a consistent estimator of the continuous JSD in any dimension. method = "legacy" reproduces the pre-1.2.0 self-normalized sample-point estimate (a bounded relative separation index rather than the continuous JSD); use it only to reproduce results from phonJSD 1.0.0.

Value

A single numeric JSD value in bits, bounded in $[0, 1]$.

Examples

```
set.seed(2026)
vowels <- data.frame(
  vowel = rep(c("ih", "eh"), each = 40),
  f1 = c(rnorm(40, 500, 55), rnorm(40, 565, 60)),
  f2 = c(rnorm(40, 1980, 150), rnorm(40, 1870, 155))
)

# One-dimensional JSD, for example a single formant or duration.
jsd_kde_nd(vowels, features = "f1", group = "vowel")

# Two-dimensional JSD in F1/F2 space.
jsd_kde_nd(vowels, features = c("f1", "f2"), group = "vowel")

# Faster high-dimensional path: diagonal Scott bandwidth and sampled
# pooled evaluation points.
```

```
jsd_kde_nd(
  vowels,
  features = c("f1", "f2"),
  group = "vowel",
  bw = "scott.diag",
  eval_n = 40,
  eval_seed = 2026,
  engine = "fast_diag"
)
```

jsd_summary

JSD summary: point estimate and optional bootstrap per group

Description

Convenience wrapper that returns both the point-estimate JSD and, optionally, bootstrap-based uncertainty (mean, SD, and CI) for each group.

Usage

```
jsd_summary(
  data,
  group_col,
  category_col,
  features,
  do_boot = TRUE,
  n_boot = 1000,
  min_tokens = 20,
  conf_level = 0.95,
  bw = c("Hpi", "Hscv", "Hpi.diag", "scott.diag"),
  eval_on = c("pooled", "group1", "group2", "pooled_sample"),
  eval_n = NULL,
  eval_seed = NULL,
  engine = c("ks", "fast_diag", "fast_diagonal"),
  chunk_size = 1000L,
  method = c("mc", "legacy"),
  density = c("kde", "mvnorm"),
  mc_n = 10000L,
  ...
)
```

Arguments

data	Data frame containing acoustic measurements.
group_col	Character vector giving one or more grouping columns (e.g., "speaker" or c("Sex", "Style")).
category_col	String: name of column giving the category to compare (e.g., "vowel"). Each group must have exactly two categories.

features	Character vector of column names giving the acoustic space.
do_boot	Logical; if TRUE (default), perform bootstrap via <code>boot_jsd()</code> .
n_boot	Integer; number of bootstrap resamples per group if <code>do_boot = TRUE</code> .
min_tokens	Minimum number of tokens per group required to compute JSD. Groups with fewer tokens are dropped.
conf_level	Confidence level for bootstrap intervals.
bw	Bandwidth selection method passed to <code>jsd_kde_nd()</code> .
eval_on	KDE evaluation points passed to <code>jsd_kde_nd()</code> .
eval_n	Optional maximum number of KDE evaluation points.
eval_seed	Optional integer seed for KDE evaluation-point subsampling.
engine	KDE evaluation engine passed to <code>jsd_kde_nd()</code> . "fast_diagonal" is accepted as an alias for "fast_diag".
chunk_size	Chunk size for engine = "fast_diag".
method	Estimator passed to <code>jsd_kde_nd()</code> : "mc" (default) or "legacy" (pre-1.2.0 self-normalized estimate). Ignored when <code>density = "mvnorm"</code> .
density	Density model passed to <code>jsd_kde_nd()</code> : "kde" (default) or "mvnorm" (fit one multivariate normal per category and estimate JSD between the two Gaussians by Monte-Carlo).
mc_n	Positive integer; number of Monte-Carlo samples drawn from each fitted Gaussian when <code>density = "mvnorm"</code> (default 10000). Ignored when <code>density = "kde"</code> .
...	Additional arguments passed to <code>jsd_kde_nd()</code> .

Value

A tibble with one row per group and columns:

- group - group ID (e.g., speaker)
- n_tokens - number of tokens for that group
- jsd_point - single JSD point estimate
- n_boot, conf_level, jsd_mean, jsd_sd, ci_lower, ci_upper, jsd_low, jsd_high - bootstrap summary columns. These are NA (or 0 for n_boot) if `do_boot = FALSE`.

kl_div

Kullback-Leibler divergence for discrete distributions

Description

Computes $KL(p \parallel q)$ in bits for discrete probability vectors. Zero-probability events in p contribute zero; positive mass in p where q is zero returns Inf.

Usage

```
kl_div(p, q)
```

Arguments

p, q Numeric probability vectors of the same length.

Value

A single numeric value: the KL divergence in bits.

percentile_rank	<i>Percentile ranks with averaged ties</i>
-----------------	--

Description

Converts a vector of measurements into percentile ranks, $(\bar{r} - 1/2)/k$, where $\bar{r}$ is the average rank (ties share the mean of the ranks they span) and k is the number of measurements ranked. This is the rank scale on which the agreement check of `rank_contrasts()` compares Jensen-Shannon distance with Pillai: a difference of 0.25 means the two measures place a speaker a quarter of the ordering apart. Ranks are ascending, so larger values of x receive larger percentile ranks. Measurements that are NA or flagged in `exclude` are left out of the ranking (they do not count toward k) and come back as NA.

Usage

```
percentile_rank(x, exclude = NULL)
```

Arguments

x Numeric vector of measurements to rank.

exclude Optional logical vector the same length as x: TRUE removes a measurement from the ranking before the ranks are computed. For example, `exclude = x >= 0.99` drops Jensen-Shannon distances at the ceiling. NA entries are treated as FALSE.

Value

Numeric vector the same length as x with percentile ranks in $(0, 1)$ for the ranked measurements and NA elsewhere.

Examples

```
percentile_rank(c(0.2, 0.5, 0.5, 0.9))
percentile_rank(c(0.2, 0.5, 0.5, 0.995), exclude = c(0.2, 0.5, 0.5, 0.995) >= 0.99)
```

percent_overlap_kde *Proportional overlap between two distributions via KDE*

Description

Computes the proportional overlap (shared area) between two categories in an n-dimensional acoustic space using multivariate kernel density estimation. Despite the historical function name, the return value is a 0–1 proportion: 0 = no overlap, 1 = identical.

Usage

```
percent_overlap_kde(
  data,
  features,
  category_col,
  bw = c("Hpi", "Hscv", "Hpi.diag", "scott.diag"),
  eval_on = c("pooled", "group1", "group2", "pooled_sample"),
  eval_n = NULL,
  eval_seed = NULL,
  engine = c("ks", "fast_diag", "fast_diagonal"),
  chunk_size = 1000L,
  method = c("mc", "legacy"),
  density = c("kde", "mvnorm"),
  mc_n = 10000L,
  bw_scale = 1,
  ...
)
```

Arguments

data	Data frame.
features	Character vector of numeric feature columns.
category_col	String; exactly two categories.
bw	Bandwidth selection method. Uses the same options as <code>jsd_kde_nd()</code> : "Hpi", "Hscv", "Hpi.diag", or "scott.diag".
eval_on	Where to evaluate the KDEs. Uses the same options as <code>jsd_kde_nd()</code> : "pooled", "group1", "group2", or "pooled_sample".
eval_n	Optional positive integer giving the maximum number of evaluation points to use.
eval_seed	Optional integer seed used only when <code>eval_n</code> causes evaluation-point subsampling.
engine	KDE evaluation engine. Uses the same options as <code>jsd_kde_nd()</code> : "ks", "fast_diag", or "fast_diagonal".
chunk_size	Positive integer controlling the number of evaluation points processed per chunk by <code>engine = "fast_diag"</code> .

method	Estimator: "mc" (default) for the Monte-Carlo plug-in estimate of the overlapping coefficient, or "legacy" for the pre-1.2.0 self-normalized sample-point estimate. eval_on applies to "legacy" only. Ignored when density = "mvnorm".
density	Density model behind the estimate: "kde" (default) estimates each category's density by kernel density estimation; "mvnorm" fits one multivariate normal per category and estimates the overlapping coefficient between the two Gaussians by Monte-Carlo. Under "mvnorm" the KDE-specific arguments (bw, engine, eval_on, chunk_size, method, eval_n) do not apply; the Monte-Carlo sample size is set by mc_n and eval_seed makes the draw reproducible.
mc_n	Positive integer; number of Monte-Carlo samples drawn from each fitted Gaussian when density = "mvnorm" (default 10000). The estimator draws mc_n fresh points from each category's fitted Gaussian to estimate the overlapping coefficient between the two Gaussians. Larger values reduce Monte-Carlo variance. Ignored when density = "kde".
bw_scale	Positive number multiplying the selected kernel bandwidth on the standard-deviation scale (univariate bandwidths by bw_scale, bandwidth matrices by bw_scale^2); see jsd_kde_nd(). Ignored when density = "mvnorm".
...	Reserved for future extensions; currently unused.

Value

Numeric scalar proportion in $[0, 1]$.

phontrast

Compute and compare phonological contrast metrics

Description

phontrast() is the package's main entry point. It computes one or more category separation and overlap metrics for a two-category phonological contrast in a single call: Jensen-Shannon divergence and distance, the Pillai-Bartlett trace, Bhattacharyya distance and affinity, Mahalanobis distance, and proportional overlap. Choose the metrics you want with metrics; the default computes all of them. Results are returned globally or by group, in a wide format (one column per metric, the default) or a tidy long format (one row per metric per comparison). The percent_overlap values are 0–1 proportions, not 0–100 percentages.

Usage

```
phontrast(
  data,
  features,
  category_col,
  group_col = NULL,
  metrics = c("jsd", "js_distance", "pillai", "bhattacharyya", "mahalanobis", "overlap"),
  min_tokens = 20,
  bw = c("Hpi", "Hscv", "Hpi.diag", "scott.diag"),
```

```

eval_on = c("pooled", "group1", "group2", "pooled_sample"),
eval_n = NULL,
eval_seed = NULL,
engine = c("ks", "fast_diag", "fast_diagonal"),
chunk_size = 1000L,
eps = 1e-06,
output = c("wide", "long"),
do_boot = FALSE,
n_boot = 1000,
conf_level = 0.95,
progress = TRUE,
method = c("mc", "legacy"),
density = c("kde", "mvnorm"),
mc_n = 10000L,
bw_scale = 1
)

```

Arguments

<code>data</code>	Data frame containing category labels and acoustic features.
<code>features</code>	Character vector of numeric feature columns.
<code>category_col</code>	String; column giving the two categories to compare.
<code>group_col</code>	Optional character vector of one or more grouping columns. If NULL, metrics are computed globally. Multiple grouping columns are combined into a labeled group value such as "Sex=F Style=read".
<code>metrics</code>	Character vector selecting which contrast metrics to compute. Any of "jsd", "js_distance", "pillai", "battacharyya", "mahalanobis", "overlap", "tv", "battacharyya_kde", and "euclidean". Defaults to the first six, the metric set of earlier releases. "battacharyya" returns both the Bhattacharyya distance and affinity under a closed-form multivariate-normal fit. The remaining three are opt-in: "tv" is total variation, 1 – proportional overlap (<code>total_variation</code>); "battacharyya_kde" returns the Bhattacharyya distance and affinity and the Hellinger distance (<code>bhatt_kde_dist</code> , <code>bhatt_kde_affinity</code> , <code>hellinger</code>) read off the <i>same</i> kernel densities as the Jensen-Shannon and overlap columns – a matched-kernel estimator of the quantity the closed-form column estimates parametrically, so measure and estimator can be told apart; and "euclidean" is the Euclidean distance between the two category means after dividing each feature by the standard deviation of the pooled two-category sample (<code>euclidean_dist</code> , bounded by $2\sqrt{d}$ for equal category sizes). All kernel-family columns (Jensen-Shannon, overlap, total variation, kernel Bhattacharyya, Hellinger) come from one shared density estimate per comparison.
<code>min_tokens</code>	Minimum tokens required globally or per group.
<code>bw</code>	Bandwidth selection method passed to <code>jsd_kde_nd()</code> and <code>percent_overlap_kde()</code> .
<code>eval_on</code>	KDE evaluation points passed to <code>jsd_kde_nd()</code> and <code>percent_overlap_kde()</code> .
<code>eval_n</code>	Optional maximum number of KDE evaluation points passed to <code>jsd_kde_nd()</code> and <code>percent_overlap_kde()</code> .

<code>eval_seed</code>	Optional integer seed for KDE evaluation-point subsampling.
<code>engine</code>	KDE evaluation engine passed to <code>jsd_kde_nd()</code> and <code>percent_overlap_kde()</code> . "fast_diagonal" is accepted as an alias for "fast_diag".
<code>chunk_size</code>	Chunk size for engine = "fast_diag".
<code>eps</code>	Small ridge constant for covariance-based metrics.
<code>output</code>	Output format: "wide" returns one row per global/group comparison; "long" returns one row per metric per comparison.
<code>do_boot</code>	Logical; if TRUE, compute bootstrap means, standard deviations, and confidence intervals for each reported metric.
<code>n_boot</code>	Number of bootstrap resamples if <code>do_boot</code> = TRUE.
<code>conf_level</code>	Confidence level for bootstrap intervals.
<code>progress</code>	Logical; if TRUE, print progress messages while bootstrap resamples are running.
<code>method</code>	KDE estimator for the JSD and percent-overlap columns, passed to <code>jsd_kde_nd()</code> / <code>percent_overlap_kde()</code> . "mc" (default) for the Monte-Carlo plug-in, or "legacy" for the pre-1.2.0 self-normalized estimate. Ignored when <code>density</code> = "mvnorm".
<code>density</code>	Density model behind the two distributional metrics (Jensen-Shannon and proportional overlap): "kde" (default) estimates each category's density by kernel density estimation; "mvnorm" fits one multivariate normal per category and estimates those two metrics between the fitted Gaussians by Monte-Carlo. This lets the density estimator be matched to the same multivariate-normal assumptions the Pillai, Bhattacharyya, and Mahalanobis columns already make. The Pillai, Bhattacharyya, and Mahalanobis columns are parametric by construction and are unaffected by this argument.
<code>mc_n</code>	Positive integer; number of Monte-Carlo samples drawn from each fitted Gaussian for the Jensen-Shannon and overlap columns when <code>density</code> = "mvnorm" (default 10000). Ignored when <code>density</code> = "kde".
<code>bw_scale</code>	Positive number multiplying the selected kernel bandwidth on the standard-deviation scale for the Jensen-Shannon and overlap columns (default 1); 0.5 and 2 give the halved and doubled bandwidths of the smoothing-sensitivity check in <code>rank_contrasts()</code> . Ignored when <code>density</code> = "mvnorm".

Details

Use `estimate_jsd()` when Jensen-Shannon divergence is the only outcome of interest, and the lower-level metric helpers when you need direct control over one estimator.

Metric directions differ. JSD, Jensen-Shannon distance, Pillai trace, Bhattacharyya distance, and Mahalanobis distance increase as categories become more separated. Percent overlap and Bhattacharyya affinity increase as categories overlap more. Long output includes `orientation`, `separation_value`, and `separation_rank` columns so all metrics can be read on a separation-oriented scale.

If `do_boot` = TRUE, each metric is recomputed on `n_boot` nonparametric bootstrap resamples to estimate uncertainty. This can take substantial time because every resample recomputes KDE, MANOVA, and covariance-based metrics. Progress messages are printed by default while bootstrapping is running; set `progress` = FALSE to suppress them.

Value

A data frame containing only the requested metrics. Wide output (the default) contains one column per requested metric plus `pillai_p_value` when Pillai is requested; with `do_boot = TRUE` it also includes metric-specific `*_mean`, `*_sd`, `*_ci_lower`, `*_ci_upper`, and `*_n_boot` columns. Long output contains `metric`, `estimate`, `orientation`, `bounded_0_1`, `separation_value`, `separation_rank`, and `p_value` (populated for the Pillai row, NA otherwise) columns; with `do_boot = TRUE` it also includes `boot_mean`, `boot_sd`, `ci_lower`, `ci_upper`, `n_boot`, and `conf_level`. The result carries class `"phontrast_contrast"`, so `plot()` and `ggplot2::autoplot()` draw it directly via `plot_overlap_metrics()`.

Examples

```
set.seed(2026)
vowels <- data.frame(
  speaker = rep(c("s01", "s02"), each = 60),
  vowel = rep(rep(c("ih", "eh"), each = 30), 2),
  f1 = c(
    rnorm(30, 500, 55), rnorm(30, 560, 60),
    rnorm(30, 510, 60), rnorm(30, 575, 65)
  ),
  f2 = c(
    rnorm(30, 1980, 150), rnorm(30, 1880, 155),
    rnorm(30, 1960, 160), rnorm(30, 1840, 165)
  )
)

# All metrics in one wide comparison table (the default), by speaker.
phontrast(
  data = vowels,
  features = c("f1", "f2"),
  category_col = "vowel",
  group_col = "speaker"
)

# A single metric in wide format.
phontrast(
  data = vowels,
  features = c("f1", "f2"),
  category_col = "vowel",
  group_col = "speaker",
  metrics = "pillai",
  output = "wide"
)

# Bootstrapping is useful but slower because every requested metric is
# recomputed on every resample. Use a larger n_boot for real analyses.
phontrast(
  data = vowels,
  features = "f1",
  category_col = "vowel",
  group_col = "speaker",
```

```

metrics = c("jsd", "pillai"),
do_boot = TRUE,
n_boot = 5,
progress = FALSE
)

```

phontrast_palette	<i>Okabe-Ito color palette used by phontrast plots</i>
-------------------	--

Description

Returns n colors from the Okabe-Ito palette, a qualitative palette designed to be distinguishable under the common forms of color-vision deficiency. The palette is reordered so the first two colors (blue and vermillion) form the highest-contrast pair for two-category contrasts. For more than eight groups the palette is interpolated, with a warning, since interpolated qualitative colors lose their guarantees.

Usage

```
phontrast_palette(n = NULL)
```

Arguments

n Number of colors to return. Defaults to the full palette.

Value

A character vector of n hex colors, named for $n \leq 8$.

Examples

```

phontrast_palette()
phontrast_palette(2)

```

pillai_overlap	<i>Pillai trace for multivariate overlap</i>
----------------	--

Description

Computes the Pillai-Bartlett trace from a MANOVA of features ~ category. Optionally adds separation estimates and a proportion-standardized Pillai score for an exactly two-category, no-covariate design.

Usage

```
pillai_overlap(data, features, category_col, proportion_standardized = FALSE)
```

Arguments

data	Data frame.
features	Character vector of numeric feature columns.
category_col	String; column giving exactly two categories.
proportion_standardized	Logical; if TRUE, append the plug-in and unbiased squared Mahalanobis separation estimates and the proportion-standardized Pillai score. The default, FALSE, preserves the original two-field return value.

Details

With `proportion_standardized = FALSE`, the function returns the ordinary Pillai trace and its p-value exactly as before. Raw Pillai describes the categories in the realized data; the proportion-standardized score targets the balanced-design score for the same estimated underlying separation.

For two categories with realized counts n_1 and n_2 , total N , p features, error degrees of freedom $\nu_e = N - 2$, and $H = 2n_1n_2/N$, the optional estimator chain is

$$\hat{D}^2 = 2\nu_e[V/(1 - V)]/H,$$

$$\tilde{D}^2 = [(\nu_e - p - 1)/\nu_e]\hat{D}^2 - 2p/H,$$

followed, when $\tilde{D}^2 \geq 0$, by

$$V_{eq} = \tilde{D}^2/(4 + \tilde{D}^2).$$

The unbiased estimator is as given by Lachenbruch and Mickey (1968), and V_{eq} implements Becker's (1986) correction and his two-group multivariate generalization.

If $\tilde{D}^2 < 0$, `pillai_eq` is NA, `pillai_eq_fallback` is TRUE, and `d2_fallback` contains only the multiplicatively corrected first term. This common near-merger outcome is labelled separately because the fallback retains the split-dependent bias `bias_2p_over_H` and is not a balanced-design equivalent.

The estimator chain assumes multivariate normality within each category and a common within-category covariance. It supports no covariates. Both categories must contain at least two complete tokens, and the within-class error SSCP must be nonsingular. When $\nu_e - p - 1 \leq 0$, all optional fields are returned as typed NA values with a warning. A minority category with fewer than $p + 1$ tokens remains computable but is flagged by `fragile_minority`.

Because $x/(4 + x)$ is strictly concave, `pillai_eq` is slightly downward biased for the balanced-design target. This bias grows with the sampling variance of $\tilde{D}^2$ and is largest for small, imbalanced, near-merged samples.

Value

A list with elements `pillai` and `p_value`. When `proportion_standardized = TRUE`, the list additionally contains `n1`, `n2`, `H`, `d2_plugin`, `d2_unbiased`, `pillai_eq`, `pillai_eq_fallback`, `d2_fallback`, `bias_2p_over_H`, and `fragile_minority`. Counts follow the category factor-level order used by the model. On the fallback path, `pillai_eq` is NA and `d2_fallback` is labelled separately; off that path, `d2_fallback` is NA.

References

- Becker, G. (1986). Correcting the point-biserial correlation for attenuation owing to unequal sample size. *Journal of Experimental Education*, 55(1), 5-8.
- Lachenbruch, P. A., & Mickey, M. R. (1968). Estimation of error rates in discriminant analysis. *Technometrics*, 10(1), 1-11.
- Berry, G. M. (2026). Beyond the null: Calibration, balance, and the interpretation of Pillai scores. Preprint.

plot.phontrast_contrast

Plot a phontrast() result directly

Description

phontrast() results carry the class "phontrast_contrast", so they can be plotted without an explicit call to plot_overlap_metrics(): plot(phontrast(...)) draws the metric comparison, and ggplot2::autoplot(phontrast(...)) returns the same plot unprinted for further composition.

Usage

```
## S3 method for class 'phontrast_contrast'
plot(x, ...)

## S3 method for class 'phontrast_contrast'
autoplot(object, ...)
```

Arguments

x, object	A phontrast_contrast object returned by phontrast().
...	Passed on to plot_overlap_metrics().

Value

plot() draws the plot and returns it invisibly; autoplot() returns the **ggplot2** object unprinted.

Examples

```
set.seed(2026)
vowels <- data.frame(
  vowel = rep(c("ih", "eh"), each = 40),
  f1 = c(rnorm(40, 500, 55), rnorm(40, 565, 60)),
  f2 = c(rnorm(40, 1980, 150), rnorm(40, 1870, 155))
)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  plot(phontrast(vowels, c("f1", "f2"), "vowel", output = "long"))
}
```

plot.phontrast_ranking

Plot a rank_contrasts() result directly

Description

rank_contrasts() results carry the class "phontrast_ranking", so plot(ranking) draws the rank-agreement plot of plot_rank_agreement() and ggplot2::autoplot(ranking) returns it unprinted.

Usage

```
## S3 method for class 'phontrast_ranking'
plot(x, ...)
```

```
## S3 method for class 'phontrast_ranking'
autoplot(object, ...)
```

Arguments

x, object A phontrast_ranking object.
 ... Passed on to plot_rank_agreement().

Value

plot() draws the plot and returns it invisibly; autoplot() returns the **ggplot2** object unprinted.

plot_category_pca

Plot a PCA projection of multidimensional category space

Description

Projects an arbitrary multidimensional acoustic feature set onto two principal components and visualizes the result with **ggplot2**. This is a diagnostic plot for high-dimensional workflows: metric estimates should still be computed on the full feature set when that is the intended analysis.

Usage

```
plot_category_pca(
  data,
  features,
  category_col,
  group_col = NULL,
  components = c(1L, 2L),
  center = TRUE,
```

```

    scale. = TRUE,
    points = TRUE,
    ellipses = TRUE,
    point_alpha = 0.65,
    point_size = 1.8,
    equal_axes = TRUE,
    facet_scales = c("fixed", "free", "free_x", "free_y")
  )

```

Arguments

<code>data</code>	Data frame containing category labels and acoustic features.
<code>features</code>	Character vector of two or more numeric feature columns used for PCA.
<code>category_col</code>	String; category column.
<code>group_col</code>	Optional grouping column used for facets.
<code>components</code>	Two positive integers giving principal components to plot.
<code>center, scale.</code>	Passed to <code>stats::prcomp()</code> .
<code>points</code>	Logical; if TRUE, show projected observations.
<code>ellipses</code>	Logical; if TRUE, add normal ellipses when enough observations are available.
<code>point_alpha</code>	Point transparency.
<code>point_size</code>	Point size.
<code>equal_axes</code>	Logical; if TRUE, use a fixed coordinate ratio.
<code>facet_scales</code>	Scales passed to <code>ggplot2::facet_wrap()</code> when <code>group_col</code> is supplied.

Value

A **ggplot2** plot object. The fitted `prcomp` object and variance-explained table are stored as `"pca"` and `"variance_explained"` attributes.

Examples

```

set.seed(2026)
features <- paste0("mfcc", 1:5)
vowels <- data.frame(
  vowel = rep(c("ih", "eh"), each = 50),
  matrix(rnorm(100 * length(features)), ncol = length(features))
)
names(vowels)[-1] <- features
vowels[vowels$vowel == "eh", features[1:2]] <- vowels[vowels$vowel == "eh", features[1:2]] + 0.8

if (requireNamespace("ggplot2", quietly = TRUE)) {
  plot_category_pca(vowels, features = features, category_col = "vowel")
}

```

plot_category_space	<i>Plot phonological categories in acoustic space</i>
---------------------	---

Description

Creates a **ggplot2** visualization of one or two acoustic dimensions from a token-level table. One-dimensional inputs are plotted as density curves; two-dimensional inputs are plotted as category-colored scatterplots with optional normal ellipses.

Usage

```
plot_category_space(
  data,
  features,
  category_col,
  group_col = NULL,
  points = TRUE,
  ellipses = TRUE,
  point_alpha = 0.65,
  point_size = 1.8,
  reverse_x = FALSE,
  reverse_y = FALSE,
  equal_axes = FALSE,
  facet_scales = c("fixed", "free", "free_x", "free_y")
)
```

Arguments

data	Data frame containing category labels and acoustic features.
features	One or two numeric feature columns to plot.
category_col	String; category column.
group_col	Optional grouping column used for facets.
points	Logical; if TRUE, show observed tokens.
ellipses	Logical; if TRUE, add normal ellipses for two-feature plots when enough observations are available.
point_alpha	Point transparency.
point_size	Point size.
reverse_x	Logical; if TRUE, reverse the x-axis.
reverse_y	Logical; if TRUE, reverse the y-axis.
equal_axes	Logical; if TRUE, use a fixed coordinate ratio for two-feature plots.
facet_scales	Scales passed to <code>ggplot2::facet_wrap()</code> when <code>group_col</code> is supplied.

Value

A **ggplot2** plot object.

Examples

```
set.seed(2026)
vowels <- data.frame(
  vowel = rep(c("ih", "eh"), each = 40),
  f1 = c(rnorm(40, 500, 55), rnorm(40, 565, 60)),
  f2 = c(rnorm(40, 1980, 150), rnorm(40, 1870, 155))
)

if (requireNamespace("ggplot2", quietly = TRUE)) {
  plot_category_space(vowels, features = "f1", category_col = "vowel")
  plot_category_space(
    vowels,
    features = c("f2", "f1"),
    category_col = "vowel",
    reverse_x = TRUE,
    reverse_y = TRUE
  )
}
```

plot_contrast

Distribution-aware contrast plot for two categories

Description

The flagship visualization for a two-category contrast. Unlike a decorative scatterplot, `plot_contrast()` draws the *same density model the distributional metrics are computed from*: under `density = "kde"` it shows highest-density regions of each category's kernel density estimate (same bandwidth selection as `jsd_kde_nd()`); under `density = "mvnorm"` it shows coverage ellipses of the fitted multivariate normals used by the parametric backend. The pointwise minimum of the two densities – the mass that the proportional-overlap metric integrates – is shaded, so the overlap itself is visible rather than implied.

Usage

```
plot_contrast(
  data,
  features,
  category_col,
  group_col = NULL,
  density = c("kde", "mvnorm"),
  bw = c("Hpi", "Hscv", "Hpi.diag", "scott.diag"),
  bw_scale = 1,
  levels = c(0.5, 0.8, 0.95),
  points = TRUE,
  overlap = TRUE,
  annotate = TRUE,
  n_boot = 0,
  conf_level = 0.95,
```

```

min_tokens = 20,
mc_n = 10000L,
eval_seed = NULL,
grid_n = NULL,
point_alpha = 0.55,
point_size = 1.6,
reverse_x = FALSE,
reverse_y = FALSE,
facet_scales = c("fixed", "free", "free_x", "free_y")
)

```

Arguments

data	Data frame with category labels and one or two numeric features.
features	One or two numeric feature columns. One feature gives density curves; two give a feature-space plot with density regions. For higher-dimensional spaces, plot a projection with <code>plot_category_pca()</code> (metrics should still be computed on the full space).
category_col	String; category column with exactly two observed categories.
group_col	Optional character vector of grouping columns; one panel per group, with per-group densities and annotations.
density	Density model to draw and to use for annotations: "kde" (default) or "mvnorm". Matches the density argument of the metric functions.
bw	Bandwidth selection method for density = "kde"; same options as <code>jsd_kde_nd()</code> .
bw_scale	Positive multiplier on the selected kernel bandwidth for density = "kde" (default 1; see <code>jsd_kde_nd()</code>). The drawn regions, the shaded overlap, and the annotations all use the same scaled bandwidth, so <code>bw_scale = 0.5</code> and <code>2</code> show the halved and doubled smoothing of the bandwidth check in <code>rank_contrasts()</code> .
levels	Numeric vector of probability levels in (0, 1) for the drawn regions: highest-density regions under "kde", coverage ellipses under "mvnorm".
points	Logical; show observed tokens (2D points, 1D rug).
overlap	Logical; shade the pointwise minimum of the two category densities (a ribbon in 1D, a soft raster in 2D). Shading strength is normalized across panels, so lighter panels genuinely overlap less.
annotate	Logical; label each panel with Jensen-Shannon divergence and proportional overlap computed under the plotted density model.
n_boot	Number of bootstrap resamples for annotation confidence intervals; 0 (default) annotates point estimates only.
conf_level	Confidence level for bootstrap intervals.
min_tokens	Minimum tokens per group; smaller groups are dropped with a warning (same convention as the metric functions).
mc_n	Monte-Carlo sample size for density = "mvnorm" annotations; passed to the metric functions.
eval_seed	Optional integer seed passed to the metric functions so annotated values are reproducible.

grid_n	Grid resolution for density evaluation: points per axis. Default 512 for one feature, 151 for two.
point_alpha	Point (or rug) transparency.
point_size	Point size for two-feature plots.
reverse_x, reverse_y	Logical; reverse an axis (e.g. F2 by F1 vowel space convention).
facet_scales	Scales passed to <code>ggplot2::facet_wrap()</code> when <code>group_col</code> is supplied.

Details

With `annotate = TRUE` (default) the panel is labelled with the Jensen-Shannon divergence and proportional overlap computed by `phontrast()` under the same density, `bw`, `mc_n`, and `eval_seed` settings, and the caption records the estimator configuration. The full annotation table is attached to the returned plot as `attr(p, "contrast_metrics")`.

Value

A **ggplot2** plot object. When `annotate = TRUE`, the `phontrast()` table behind the labels is attached as `attr(p, "contrast_metrics")`.

Examples

```
set.seed(2026)
vowels <- data.frame(
  vowel = rep(c("ih", "eh"), each = 60),
  f1 = c(rnorm(60, 500, 55), rnorm(60, 565, 60)),
  f2 = c(rnorm(60, 1980, 150), rnorm(60, 1870, 155))
)

if (requireNamespace("ggplot2", quietly = TRUE)) {
  # Two-feature contrast in vowel-space orientation, KDE regions.
  plot_contrast(vowels, c("f2", "f1"), "vowel",
    reverse_x = TRUE, reverse_y = TRUE)

  # One-feature contrast with the overlap ribbon.
  plot_contrast(vowels, "f1", "vowel")

  # The same contrast under the multivariate-normal backend.
  plot_contrast(vowels, c("f2", "f1"), "vowel", density = "mvnorm",
    eval_seed = 2026, reverse_x = TRUE, reverse_y = TRUE)
}
```

Description

Visualizes the output of `phontrast()` with **ggplot2**. The input may be either wide or long output. By default, values are plotted on a separation-oriented scale so overlap-oriented metrics are transformed as $1 - \text{estimate}$.

Usage

```
plot_overlap_metrics(
  metrics,
  value = c("separation", "estimate"),
  metric = NULL,
  group_col = NULL,
  show_ci = TRUE,
  facet = TRUE,
  sort = TRUE
)
```

Arguments

<code>metrics</code>	Data frame returned by <code>phontrast()</code> .
<code>value</code>	Scale to plot: "separation" plots <code>separation_value</code> ; "estimate" plots raw metric estimates.
<code>metric</code>	Optional character vector of metric display names to include.
<code>group_col</code>	Optional column to use on the x-axis. Defaults to "group" when present, then "scope".
<code>show_ci</code>	Logical; if TRUE, draw confidence intervals when <code>ci_lower/ci_upper</code> columns are present.
<code>facet</code>	Logical; if TRUE, facet by metric.
<code>sort</code>	Logical; if TRUE, order comparisons by their mean plotted value.

Value

A **ggplot2** plot object.

Examples

```
set.seed(2026)
vowels <- data.frame(
  speaker = rep(c("s01", "s02"), each = 60),
  vowel = rep(rep(c("ih", "eh"), each = 30), 2),
  f1 = c(rnorm(30, 500, 55), rnorm(30, 560, 60),
        rnorm(30, 510, 60), rnorm(30, 575, 65)),
  f2 = c(rnorm(30, 1980, 150), rnorm(30, 1880, 155),
        rnorm(30, 1960, 160), rnorm(30, 1840, 165))
)

metrics <- phontrast(
  vowels,
```

```

    features = c("f1", "f2"),
    category_col = "vowel",
    group_col = "speaker",
    output = "long"
  )

  if (requireNamespace("ggplot2", quietly = TRUE)) {
    plot_overlap_metrics(metrics)
  }

```

plot_rank_agreement *Rank-agreement plot for a speaker ranking*

Description

Draws step 3 of the ranking protocol behind `rank_contrasts()`: each speaker's Pillai percentile rank against its Jensen-Shannon distance percentile rank, the identity line on which the two measures agree, and the inspection band of margin either side of it. Speakers outside the band are flagged; speakers the bandwidth check set aside are crossed; speakers whose flag may not be read at this sample size and dimensionality are hollow. Speakers at the $\sqrt{JSD}$ ceiling have no rank and are listed in the caption instead of drawn.

Usage

```

plot_rank_agreement(
  ranking,
  label = c("flagged", "all", "none"),
  base_size = 12
)

```

Arguments

ranking	A <code>phontrast_ranking</code> object from <code>rank_contrasts()</code> .
label	Which speakers to name on the plot: "flagged" (default: flagged or set-aside speakers), "all", or "none".
base_size	Base font size passed to <code>theme_phontrast()</code> .

Value

A `ggplot2` plot object.

See Also

`rank_contrasts()`, `inspect_contrast()`.

Examples

```

set.seed(2026)
gaps <- c(30, 60, 90, 120, 150, 180, 210, 240)
cohort <- do.call(rbind, lapply(seq_along(gaps), function(i) {
  data.frame(
    speaker = sprintf("s%02d", i),
    vowel = rep(c("ih", "eh"), each = 50),
    f1 = c(rnorm(50, 500, 55), rnorm(50, 500 + gaps[i], 55)),
    f2 = c(rnorm(50, 1900, 120), rnorm(50, 1900 - gaps[i], 120))
  )
}))
ranking <- rank_contrasts(cohort, c("f1", "f2"), "vowel", "speaker",
  bw_check = FALSE)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  plot_rank_agreement(ranking, label = "all")
}

```

prepare_jsd_beta

*Prepare JSD estimates for beta regression / GAMs***Description**

JSD lives in $[0, 1]$. This helper adds a `jsd_beta` column bounded in $(0, 1)$ so it can be used with the Beta family (e.g., in **mgcv**).

Usage

```
prepare_jsd_beta(jsd_df, jsd_col = "jsd_mean", eps = 1e-06)
```

Arguments

<code>jsd_df</code>	Data frame containing a JSD column.
<code>jsd_col</code>	String: name of the JSD column (default "jsd_mean").
<code>eps</code>	Small constant used to bound JSD away from 0 and 1.

Value

A modified data frame with an added `jsd_beta` column.

Examples

```

jsd_by_speaker <- data.frame(
  speaker = paste0("s", 1:8),
  age = c(18, 22, 27, 31, 38, 45, 52, 60),
  jsd_mean = c(0.02, 0.05, 0.08, 0.13, 0.18, 0.24, 0.31, 0.39)
)

model_data <- prepare_jsd_beta(jsd_by_speaker)

```

```

model_data

if (requireNamespace("mgcv", quietly = TRUE)) {
  fit <- mgcv::gam(
    jsd_beta ~ age,
    data = model_data,
    family = mgcv::betar(),
    method = "REML"
  )
  stats::predict(fit, type = "response")
}

```

protocol_floors

Sample-size floors of the ranking protocol

Description

The token floors per category at which the simulation study behind `rank_contrasts()` licenses a $\sqrt{JSD}$ ranking and the Pillai- $\sqrt{JSD}$ agreement flag, by number of features d (Berry, under review, Secs. V.D and VII.A). Dimensionalities the study did not calibrate take the floors of the next higher calibrated dimensionality.

Usage

```
protocol_floors(d)
```

Arguments

`d` Positive integer; the number of features.

Value

A list with d ; `rank_floor`, the tokens per category from which ranking by $\sqrt{JSD}$ is licensed (50 at $d \leq 2$, 200 at $d = 3, 4$, 500 at $d = 5$ to 8, Inf above eight dimensions, where the study gives ordering evidence only); `flag_floor`, the tokens per category from which the agreement flag may be read (100 at $d \leq 2$, 200 at $d = 3, 4$, Inf from $d = 5$, since from eight dimensions the test returns agreement whatever the data contain); `calibrated_at`, the calibrated dimensionality the floors come from; and `ordering_only`, TRUE above eight dimensions.

Examples

```

protocol_floors(2)
protocol_floors(4)$flag_floor

```

rank_contrasts	<i>Rank speakers' contrasts by Jensen-Shannon distance and check them against Pillai</i>
----------------	--

Description

One-call implementation of the measurement protocol recommended by the simulation study behind phontrast (Berry, under review, Sec. VII.A). For each speaker (or other unit named by group_col) it

1. computes Jensen-Shannon distance ($\sqrt{JSD}$) and the Pillai trace on the same tokens, from one shared kernel density estimate, and reports the estimated shared probability mass beside them;
2. ranks the speakers by $\sqrt{JSD}$ on the percentile-rank scale of percentile_rank();
3. flags speakers whose Pillai percentile rank differs from their $\sqrt{JSD}$ percentile rank by margin (0.25 of the ordering) or more, for inspection with inspect_contrast() or plot_contrast().

The conditions the study attaches to those steps are applied and reported rather than left to the user: measurements at the $\sqrt{JSD}$ ceiling are set apart, sample-size floors decide whether a rank or a flag may be read, and a bandwidth check marks measurements whose rank depends on the smoothing.

Usage

```
rank_contrasts(
  data,
  features,
  category_col,
  group_col,
  margin = 0.25,
  ceiling = 0.99,
  bw_check = TRUE,
  estimator = recommended_estimator(length(features)),
  min_tokens = 10,
  eval_seed = NULL,
  chunk_size = 1000L
)
```

Arguments

data	Data frame with the speaker, category, and feature columns.
features	Character vector of numeric feature columns.
category_col	String; column giving the two categories of the contrast (for example the two vowels).
group_col	Character vector of one or more columns identifying the speakers (or other units) to rank. Required: one speaker measured once on one contrast gives nothing to rank.

margin	Inspection margin on the percentile-rank scale: the flag is raised when $\text{abs}(\text{rank_diff}) \geq \text{margin}$ and a measurement is set aside by the bandwidth check when its $\sqrt{JSD}$ rank moves by margin or more. Default 0.25 of the ordering, the value calibrated in the study.
ceiling	$\sqrt{JSD}$ at or above which a measurement counts as at the ceiling (default 0.99).
bw_check	Logical; run the bandwidth check (default TRUE).
estimator	Kernel estimator settings: a list with elements bw, engine, eval_n, and loo as returned by <code>recommended_estimator()</code> . Defaults to the settings the study used at this dimensionality, <code>recommended_estimator(length(features))</code> .
min_tokens	Minimum tokens in the smaller category for a speaker to be measured at all (default 10). Speakers below it, or without exactly two observed categories, are left out with a message; the licensing floors above are applied to the speakers that remain.
eval_seed	Optional integer seed used when <code>estimator\$eval_n</code> subsamples the evaluation points; makes the estimates reproducible.
chunk_size	Chunk size for engine = "fast_diag".

Value

A tibble of class "phontrast_ranking", one row per measured speaker, sorted by `sqrt_jsd` (largest first), with columns `group`, `n_tokens`, `n_min`, `sqrt_jsd`, `pillai`, `shared_mass`, `at_ceiling`, `pr_jsd`, `pr_pillai`, `rank_diff` (`pr_jsd - pr_pillai`), `flag`, `rank_licensed`, `flag_licensed`, `rank_basis`, and, with `bw_check = TRUE`, `sqrt_jsd_half`, `sqrt_jsd_double`, `bw_shift`, `sign_change`, and `set_aside`. The protocol settings, the floors, and the cleaned tokens the ranking was computed from are attached as `attr(x, "protocol")` (so `inspect_contrast()` can redraw a speaker without the original data); `print()` summarizes them. `plot()` and `ggplot2::autoplot()` draw the rank-agreement plot.

Ceiling

A measurement with $\sqrt{JSD} \geq \text{ceiling}$ (default 0.99) is at the ceiling: the kernel estimate can no longer separate degrees of separation. Its shared mass and Pillai value are reported, but it is excluded from both percentile rankings (`at_ceiling = TRUE`, `pr_jsd` and `pr_pillai` are NA), and the ranks of the remaining k speakers are computed over those k alone.

Sample-size floors

Floors are read from the smaller category's token count per speaker (`n_min`) and from the number of features d (see `protocol_floors()`). Ranking by $\sqrt{JSD}$ is licensed from 50 tokens per category at $d = 2$, 200 at $d = 3$ or 4, and 500 at $d = 5$ to 8; above eight dimensions the study gives ordering evidence only, so no rank is licensed. Below the floor `rank_basis` is "pillai": order that speaker by Pillai. The flag has floors of its own: 100 tokens per category at $d = 2$ and 200 at $d = 3$ or 4; from eight dimensions up the agreement test returns agreement whatever the data contain, and it is undefined when $d \geq 2 * n_{\min}$. Where the flag may not be read, `flag` is NA and `rank_diff` is still reported. The floors are where the simulation recovers the average ordering of its separation levels; they do not guarantee an accurate ranking of an individual speaker. Eight speakers is the smallest useful set (the calibration used sets of 32); fewer draws a warning.

Bandwidth check

With `bw_check = TRUE` (the default) $\sqrt{JSD}$ is recomputed at half and at twice the diagonal Scott bandwidth (`bw = "scott.diag"`, `bw_scale = 0.5` and `2`), at two dimensions as well, where the reported estimate uses the plug-in rule. The two re-estimates are ranked over the same speakers and `bw_shift` is the percentile-rank change between them. A measurement is set aside (`set_aside = TRUE`) when `abs(bw_shift) >= margin`, or when a flagged Pillai- $\sqrt{JSD}$ rank difference changes sign between the halved and the doubled bandwidth (`sign_change`). The check tests sensitivity to smoothing, not the estimate's accuracy.

See Also

`percentile_rank()`, `protocol_floors()`, `recommended_estimator()`, `inspect_contrast()`, `plot_contrast()`, `phontrast()`.

Examples

```
set.seed(2026)
gaps <- c(30, 60, 90, 120, 150, 180, 210, 240)
cohort <- do.call(rbind, lapply(seq_along(gaps), function(i) {
  data.frame(
    speaker = sprintf("s%02d", i),
    vowel = rep(c("ih", "eh"), each = 50),
    f1 = c(rnorm(50, 500, 55), rnorm(50, 500 + gaps[i], 55)),
    f2 = c(rnorm(50, 1900, 120), rnorm(50, 1900 - gaps[i], 120))
  )
}))
ranking <- rank_contrasts(cohort, c("f1", "f2"), "vowel", "speaker",
                          bw_check = FALSE)
ranking
```

`recommended_estimator` *Kernel estimator settings by dimensionality*

Description

Returns the kernel-density estimator configuration that the simulation study behind `rank_contrasts()` used at each dimensionality (Berry, under review, Table III), so that a contrast measured with `phontrast` is scored with the settings the safe-use envelope was calibrated on. Three settings switch together with the number of features `d`:

Usage

```
recommended_estimator(d)
```

Arguments

`d` Positive integer; the number of acoustic features (dimensions).

Details

- $d \leq 4$ Plug-in bandwidth ($bw = "Hpi"$), the "ks" engine, densities evaluated at every token, and the partial leave-one-out correction on. Calibrated at $d = 2$ and 4 .
- $5 \leq d \leq 13$ Diagonal Scott bandwidth ($bw = "scott.diag"$), the "fast_diag" engine, densities evaluated at 200 subsampled tokens per category, leave-one-out on. Calibrated at $d = 8$ and 13 ; dimensionalities between the calibrated ones take the settings of the next higher calibrated dimensionality.
- $d \geq 14$ As above but with no leave-one-out correction, reproducing the log-space path the study used at $d = 32$ and 64 . phontrast's "fast_diag" engine evaluates kernels in log space, so this tier runs natively. Above eight dimensions the study gives ordering evidence only.

Value

A list with elements `d`, `tier` (a label for the row of Table III applied), `calibrated_at` (the dimensionalities the row was calibrated on), `bw`, `engine`, `eval_n`, `loo`, and `note`. The `bw`, `engine`, `eval_n`, and `loo` elements can be passed straight to `jsd_kde_nd()`, `estimate_jsd()`, or `phontrast()`.

Examples

```
recommended_estimator(2)
recommended_estimator(13)$engine

set.seed(2026)
vowels <- data.frame(
  vowel = rep(c("ih", "eh"), each = 40),
  f1 = c(rnorm(40, 500, 55), rnorm(40, 565, 60)),
  f2 = c(rnorm(40, 1980, 150), rnorm(40, 1870, 155))
)
est <- recommended_estimator(2)
jsd_kde_nd(vowels, c("f1", "f2"), "vowel",
  bw = est$bw, engine = est$engine, loo = est$loo)
```

scale_colour_phontrast

Okabe-Ito discrete color and fill scales

Description

Discrete **ggplot2** scales built on `phontrast_palette()`. These are the default scales for all phontrast plotting functions and can be added to any ggplot to match the package's visual identity.

Usage

```
scale_colour_phontrast(...)

scale_color_phontrast(...)

scale_fill_phontrast(...)
```

Arguments

... Passed to `ggplot2::discrete_scale()` (e.g. name, labels, guide).

Value

A `ggplot2` scale object.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  ggplot2::ggplot(iris, ggplot2::aes(Sepal.Length, Sepal.Width,
                                     color = Species)) +
    ggplot2::geom_point() +
    scale_color_phontrast() +
    theme_phontrast()
}
```

speaker_bhatt	<i>Bhattacharyya distance by group</i>
---------------	--

Description

Computes Bhattacharyya distance and affinity for each group, under a multivariate normal approximation.

Usage

```
speaker_bhatt(
  data,
  group_col,
  category_col,
  features,
  min_tokens = 20,
  eps = 1e-06
)
```

Arguments

data	Data frame.
group_col	Character vector of one or more grouping columns (e.g., "speaker" or <code>c("Sex", "Style")</code>).
category_col	String; category column with exactly two levels per group.
features	Character vector of numeric feature columns.
min_tokens	Minimum tokens per group.
eps	Small ridge constant passed to <code>bhattacharyya_mvnorm()</code> .

Value

Data frame with columns: group, n_tokens, bhatt_dist, bhatt_affinity.

speaker_jsd	<i>Group-level JSD point estimates</i>
-------------	--

Description

Computes JSD for each group (e.g., speaker) comparing two categories (e.g., vowels) in an n-dimensional acoustic space.

Usage

```
speaker_jsd(
  data,
  group_col,
  category_col,
  features,
  min_tokens = 20,
  bw = c("Hpi", "Hscv", "Hpi.diag", "scott.diag"),
  eval_on = c("pooled", "group1", "group2", "pooled_sample"),
  eval_n = NULL,
  eval_seed = NULL,
  engine = c("ks", "fast_diag", "fast_diagonal"),
  chunk_size = 1000L,
  ...
)
```

Arguments

data	Data frame containing acoustic measurements.
group_col	Character vector giving one or more grouping columns (e.g., "speaker" or c("Sex", "Style")).
category_col	String: name of column giving the category to compare (e.g., "vowel"). Each group must have exactly two categories.
features	Character vector of column names giving the acoustic space.
min_tokens	Minimum number of tokens per group required to compute JSD. Groups with fewer tokens are dropped.
bw	Bandwidth selection method passed to jsd_kde_nd().
eval_on	KDE evaluation points passed to jsd_kde_nd().
eval_n	Optional maximum number of KDE evaluation points.
eval_seed	Optional integer seed for KDE evaluation-point subsampling.
engine	KDE evaluation engine passed to jsd_kde_nd(). "fast_diagonal" is accepted as an alias for "fast_diag".
chunk_size	Chunk size for engine = "fast_diag".
...	Additional arguments passed to jsd_kde_nd().

Value

A tibble with one row per group and columns: group, n_tokens, and jsd.

speaker_pillai	<i>Group-level Pillai scores</i>
----------------	----------------------------------

Description

Computes Pillai scores and associated p-values per group (e.g., per speaker).

Usage

```
speaker_pillai(data, group_col, category_col, features, min_tokens = 20)
```

Arguments

data	Data frame.
group_col	Character vector of one or more grouping columns (e.g., "speaker" or c("Sex", "Style")).
category_col	String; category column (e.g., "vowel").
features	Character vector of numeric feature columns.
min_tokens	Minimum tokens per group.

Value

A tibble with columns: group, n_tokens, pillai, p_value.

theme_phontrast	<i>Publication theme for phontrast plots</i>
-----------------	--

Description

A minimal, publication-oriented **ggplot2** theme: quiet major grid, no minor grid, bold plot-aligned title, muted subtitle, left-aligned caption for estimator provenance, and a top-aligned legend. Applied by default in all phontrast plotting functions.

Usage

```
theme_phontrast(base_size = 12, base_family = "")
```

Arguments

base_size	Base font size in points.
base_family	Base font family.

Value

A **ggplot2** theme object.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  ggplot2::ggplot(iris, ggplot2::aes(Sepal.Length, Sepal.Width)) +
    ggplot2::geom_point() +
    theme_phontrast()
}
```

vowel_cohort

Simulated twelve-speaker vowel cohort

Description

A simulated cohort for demonstrating the ranking protocol of `rank_contrasts()`: twelve speakers, two vowel categories ("ih" and "eh"), and two acoustic features (F1 and F2, in Hz). Each speaker was built to exercise one part of the protocol:

spk01–spk08 Gaussian categories whose centroid gap grows from 25 to 200 Hz, 100 tokens per vowel: a graded ordering that both Jensen-Shannon distance and Pillai recover.

spk09 The same centroids for both vowels, but a bimodal "eh" (two variants 190 Hz apart in F1 and 440 Hz apart in F2), 100 tokens per vowel. A mean-based measure sees no contrast while the distributions barely overlap: the planted Pillai / $\sqrt{JSD}$ disagreement the agreement flag should catch.

spk10 Fully separated categories, 100 tokens per vowel: $\sqrt{JSD}$ at the ceiling.

spk11 60 tokens per vowel: ranking by $\sqrt{JSD}$ is licensed at two dimensions (floor 50) but the flag is not readable (floor 100).

spk12 40 tokens per vowel: below the two-dimensional rank floor, so the speaker is ordered by Pillai.

Usage

```
vowel_cohort
```

Format

A data frame with 2200 rows and 4 columns:

speaker Character; speaker identifier "spk01" to "spk12".

vowel Character; vowel category, "ih" or "eh".

f1 Numeric; first formant frequency in Hz.

f2 Numeric; second formant frequency in Hz.

Source

Simulated with a fixed seed; the generating script is `data-raw/vowel_cohort.R` in the package repository.

See Also

`rank_contrasts()`, `inspect_contrast()`.

Examples

```
head(vowel_cohort)
table(vowel_cohort$speaker, vowel_cohort$vowel)
```

Index

- * **datasets**
 - vowel_cohort, [48](#)
- autoplot.phontrast_contrast
 - (plot.phontrast_contrast), [30](#)
- autoplot.phontrast_ranking
 - (plot.phontrast_ranking), [31](#)
- bhattacharyya_mvnorm, [3](#)
- boot_jsd, [3](#)
- estimate_bhatt, [5](#)
- estimate_jsd, [5](#)
- estimate_overlap, [8](#)
- estimate_pillai, [9](#)
- extract_mfcc, [10](#)
- global_boot_jsd, [11](#)
- global_pillai, [13](#)
- hier_boot_jsd_model, [14](#)
- inspect_contrast, [15](#)
- jsd, [17](#)
- jsd_kde_nd, [17](#)
- jsd_summary, [20](#)
- kl_div, [21](#)
- percent_overlap_kde, [23](#)
- percentile_rank, [22](#)
- phontrast, [24](#)
- phontrast_palette, [28](#)
- pillai_overlap, [28](#)
- plot.phontrast_contrast, [30](#)
- plot.phontrast_ranking, [31](#)
- plot_category_pca, [31](#)
- plot_category_space, [33](#)
- plot_contrast, [34](#)
- plot_overlap_metrics, [36](#)
- plot_rank_agreement, [38](#)
- prepare_jsd_beta, [39](#)
- protocol_floors, [40](#)
- rank_contrasts, [41](#)
- recommended_estimator, [43](#)
- scale_color_phontrast
 - (scale_colour_phontrast), [44](#)
- scale_colour_phontrast, [44](#)
- scale_fill_phontrast
 - (scale_colour_phontrast), [44](#)
- speaker_bhatt, [45](#)
- speaker_jsd, [46](#)
- speaker_pillai, [47](#)
- theme_phontrast, [47](#)
- vowel_cohort, [48](#)