

Package: janusplot (via r-universe)

July 2, 2026

Type Package

Title Asymmetric Smoothed-Association Matrices via GAM Fits

Version 0.1.1

Description Render a pairwise, asymmetric smoothed-association matrix of continuous variables. Each cell shows the fitted spline from an 'mgcv' generalised additive model, with the upper triangle displaying 'gam(x_j ~ s(x_i))' and the lower triangle 'gam(x_i ~ s(x_j))'. Unlike Pearson's correlation matrix, the visualisation is intentionally asymmetric, revealing heteroscedasticity, leverage, and directional non-linearity that a single scalar correlation hides. An asymmetry index and a 24-category shape taxonomy quantify the directional difference and qualitative form of each fitted smooth.

License GPL (>= 3)

URL <https://github.com/max578/janusplot>,
<https://max578.github.io/janusplot/>

BugReports <https://github.com/max578/janusplot/issues>

Encoding UTF-8

Language en-AU

Depends R (>= 4.3.0)

Imports mgcv (>= 1.9.0), ggplot2 (>= 3.5.0), patchwork (>= 1.1.0),
grid, stats, cli (>= 3.6.0), lifecycle, rlang (>= 1.1.0)

Suggests agridat, future, future.apply, knitr, MASS, palmerpenguins,
rmarkdown, testthat (>= 3.0.0), vdiffR (>= 1.0.0), withr

VignetteBuilder knitr

RoxygenNote 7.3.3

Config/testthat/edition 3

Config/Needs/website pkgdown

LazyData true

NeedsCompilation no

Author Max Moldovan [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0001-9680-8474>)

Maintainer Max Moldovan <max.moldovan@adelaide.edu.au>

Repository <https://cran.r-universe.dev>

Date/Publication 2026-07-02 07:00:29 UTC

RemoteUrl <https://github.com/cran/janusplot>

RemoteRef HEAD

RemoteSha 4591e5294b401812a773a2f521d1cd843df508bf

Contents

janusplot	2
janusplot_data	10
janusplot_shape_cutoffs	14
janusplot_shape_hierarchy	15
janusplot_shape_metrics	16
janusplot_shape_sensitivity	18
janusplot_shape_sensitivity_plot	20
janusplot_shape_sensitivity_shapes	21
janusplot_shape_sensitivity_summary	21
shape_sensitivity_demo	22

Index 24

janusplot	<i>Asymmetric smoothed-association matrix</i>
-----------	---

Description

[Experimental]

Render a pairwise, asymmetric matrix of smoothed associations between numeric variables. Each cell $[i, j]$ where $i \neq j$ shows the fitted spline from `mgcv::gam()`:

- Upper triangle ($i < j$): `gam(x_j ~ s(x_i) + <adjust>)`.
- Lower triangle ($i > j$): `gam(x_i ~ s(x_j) + <adjust>)`.
- Diagonal: blank panel when labels live on the border (default), or a variable-name label when `labels = "diagonal"`.

The two triangles intentionally differ — the asymmetry reveals heteroscedasticity, leverage, and directional non-linearity that a single scalar correlation hides.

Usage

```
janusplot(  
  data,  
  vars = NULL,  
  adjust = NULL,  
  method = NULL,  
  k = -1L,  
  bs = "tp",  
  engine = c("bam", "gam"),  
  discrete = FALSE,  
  nthreads = 1L,  
  order = c("original", "hclust", "alphabetical"),  
  show_data = TRUE,  
  show_ci = TRUE,  
  display = c("fit", "d1", "d2"),  
  derivative_ci = c("none", "pointwise", "simultaneous"),  
  derivative_ci_nsim = 1000L,  
  n_grid = NULL,  
  colour_by = c("pearson", "spearman", "kendall", "edf", "deviance_gap", "none"),  
  fill_by = NULL,  
  palette = NULL,  
  annotations = c("edf", "A"),  
  shape_cutoffs = janusplot_shape_cutoffs(),  
  show_shape_legend = TRUE,  
  glyph_style = c("ascii", "unicode"),  
  labels = c("border", "diagonal", "none"),  
  diagonal = c("auto", "blank", "name", "density"),  
  label_srt = 45,  
  label_cex = 1,  
  signif_glyph = TRUE,  
  show_asymmetry = NULL,  
  na_action = c("pairwise", "complete"),  
  parallel = FALSE,  
  with_data = FALSE,  
  text_scale_diag = 1,  
  text_scale_off_diag = 1,  
  show_glossary = TRUE,  
  glossary_scale = 1,  
  k_check_thresholds = NULL,  
  auto_refit_k = FALSE,  
  k_max_iter = 2L,  
  compact = c("auto", "always", "never"),  
  compact_threshold = 12L,  
  compact_levels = NULL,  
  focus_by = NA_character_,  
  focus_threshold = "q90",  
  focus_dim_alpha = 0.25,  
  axes = c("original", "standardised", "centred", "rank"),
```

```

    save_as = NULL,
    save_width = NULL,
    save_height = NULL,
    save_dpi = 300,
    ...
)

```

Arguments

<code>data</code>	A data frame with numeric columns to include.
<code>vars</code>	Character vector of column names to use. <code>NULL</code> (default) uses all numeric columns in <code>data</code> . Non-numeric columns trigger an error listing offenders.
<code>adjust</code>	A one-sided formula RHS giving additional covariates and/or random effects to include in every pairwise GAM. For example, <code>adjust = ~ s(age) + s(site, bs = "re")</code> fits <code>gam(y ~ s(x) + s(age) + s(site, bs = "re"))</code> for each pair. Default <code>NULL</code> fits unadjusted pairwise smooths.
<code>method</code>	Smoothing-parameter estimation method passed to the chosen fitting backend. Default <code>NULL</code> resolves per-engine: <code>"fREML"</code> for engine = <code>"bam"</code> (mgcv's recommended at scale), <code>"REML"</code> for engine = <code>"gam"</code> (the v0.1.0 behaviour). Pass any valid mgcv method string to override.
<code>k</code>	Integer, or named list mapping variable names to integers. Basis dimension for <code>s()</code> . Default <code>-1L</code> (mgcv's automatic choice).
<code>bs</code>	Basis type for <code>s()</code> . Default <code>"tp"</code> (thin plate).
<code>engine</code>	One of <code>"bam"</code> (default, new in v0.1.1) or <code>"gam"</code> . Selects mgcv's fitting backend: <ul style="list-style-type: none"> <code>"bam"</code> — <code>mgcv::bam()</code>. Block-Lanczos solve + fREML estimation + lower memory. ~3-10x speedup at janusplot's scale (<code>k = 15-25</code> vars, 600+ pairwise fits per call). The default, and the one non-byte-identical change in v0.1.1: fREML differs from REML by ~1-3% in EDF on identical data, so the asymmetry index may shift by similar amounts vs v0.1.0 output. Recoverable verbatim via engine = <code>"gam"</code>. <code>"gam"</code> — <code>mgcv::gam()</code>. The v0.1.0 backend. Use for backward-compat reproduction, very small <code>n</code> (<code>< 200</code>) where <code>bam</code>'s setup overhead exceeds its solve gain, or methodologically sensitive contexts that require REML rather than fREML.
<code>discrete</code>	Logical. <code>bam</code> -only opt-in to mgcv's covariate-discretisation optimisation. Further ~2-5x speedup at the cost of small (sub-pixel at typical janusplot resolution) prediction-shift. Default <code>FALSE</code> . Ignored when engine = <code>"gam"</code> .
<code>nthreads</code>	Integer. <code>bam</code> -only intra-fit threading. Default <code>1L</code> to avoid oversubscription when combined with <code>parallel = TRUE</code> (future.apply fans out pair fits across cores; <code>nthreads > 1</code> within each pair would double-book CPUs). Raise above 1 only when <code>parallel = FALSE</code> . Ignored when engine = <code>"gam"</code> .
<code>order</code>	One of <code>"original"</code> (default), <code>"hclust"</code> (reorder by hierarchical clustering of Pearson correlations), or <code>"alphabetical"</code> .
<code>show_data</code>	Logical. If <code>TRUE</code> (default), overlay raw data points (low alpha) behind each spline. Only applies when <code>display = "fit"</code> ; derivative panels never overlay raw data.

show_ci	Logical. If TRUE (default), overlay the 95% confidence envelope from <code>predict(gam, se.fit = TRUE)</code> on the fit panel (i.e. when <code>display = "fit"</code>). CI rendering on derivative panels is controlled separately by <code>derivative_ci</code> .
display	<p>One of "fit" (default), "d1", or "d2". Selects which single quantity is rendered in every off-diagonal cell of the matrix.</p> <ul style="list-style-type: none"> • "fit" — the fitted smooth $\hat{f}(x)$; default, behaviour identical to the pre-derivative release. • "d1" — the first derivative $\hat{f}'(x)$ of the fitted smooth. Zero crossings localise turning points of \hat{f}. • "d2" — the second derivative $\hat{f}''(x)$. Zero crossings localise inflection points of \hat{f}. <p>A single matrix shows a single quantity by design: stacked multi-panel cells crowd the matrix at any realistic variable count. To compare fit against derivative, render two or three <code>janusplot()</code> calls side-by-side; each call keeps its own <code>with_data = TRUE</code> summary table tagged with the <code>display</code> column.</p> <p>Orders $k \geq 3$ are not exposed — higher-order derivatives of penalised regression splines amplify noise and rarely carry usable signal at realistic sample sizes. See <code>vignette("janusplot")</code> for the theoretical justification and applied use-cases.</p>
derivative_ci	<p>One of "none" (default), "pointwise", or "simultaneous". Controls whether — and how — a 95% confidence ribbon is drawn underneath the derivative curve when <code>display %in% c("d1", "d2")</code>. Ignored when <code>display = "fit"</code>.</p> <ul style="list-style-type: none"> • "none" — no ribbon. The curve and the zero reference line are all you see. Default, because pointwise ribbons overshoot nominal coverage as a joint region and can invite over-reading of local features. • "pointwise" — 95% pointwise ribbon from $\sqrt{\text{diag}(DV_p D^T)}$ (Wood 2017 §7.2.4). Valid marginally; not a simultaneous statement. • "simultaneous" — 95% simultaneous band via the Monte Carlo construction of Ruppert, Wand & Carroll (2003) popularised for GAMs by Simpson (2018, <i>Frontiers Ecol. Evol.</i> 6:149): draw B samples $\tilde{\beta} \sim \mathcal{N}(\hat{\beta}, V_p)$, compute $\max_x D_i(\tilde{\beta} - \hat{\beta}) /se_i$, and use the $(1 - \alpha)$ quantile as a critical multiplier on the pointwise SE. Valid for feature localisation ("where is $\hat{f}'(x)$ significantly non-zero").
derivative_ci_nsim	<p>Integer. Number of Monte Carlo samples used when <code>derivative_ci = "simultaneous"</code>. Default 1000L — a compromise between coverage accuracy (Simpson 2018 uses 10000) and CPU budget across every pair in a medium-sized matrix. Ignored for any other <code>derivative_ci</code>.</p>
n_grid	<p>Integer or NULL. Number of equally-spaced points used to evaluate each fitted smooth (and its derivatives). Default NULL resolves to 100 when <code>display = "fit"</code> and 200 otherwise, because finite-difference second derivatives visibly degrade below ~ 150 points on moderate-k smooths. Supplying <code>n_grid</code> directly overrides both defaults. Larger grids shift the numerical shape-metric values (M, C, turning / inflection counts) slightly because they are computed on this same grid. Shapes and asymmetry are the primary reading; M, C and the counts are secondary diagnostics and the grid-induced drift is tolerable.</p>

colour_by	One of "pearson" (default), "spearman", "kendall", "edf", "deviance_gap", or "none". Encodes the per-cell fill colour by the chosen scalar. Correlation choices use a diverging palette with limits $c(-1, 1)$ and a shared corr colour-bar title; "edf" and "deviance_gap" use a sequential palette labelled by the metric.
fill_by	Deprecated alias for colour_by. If supplied emits a single soft deprecation warning and is forwarded to colour_by.
palette	Character. Colour palette for the cell fill scale. Defaults to "RdBu" when colour_by is a correlation and "viridis" otherwise. Sequential choices: "viridis", "magma", "inferno", "plasma", "cividis", "mako", "rocket", "turbo" (not CB-safe), "YlOrRd", "YlGnBu", "Blues", "Greens". Diverging choices: "RdYlBu", "RdBu", "PuOr", "Spectral" (not CB-safe). Passing a sequential palette while colour_by is a correlation silently upgrades to the default diverging palette.
annotations	Character vector, a subset of $c("edf", "A", "shape", "code")$. Controls which corner annotations appear on each off-diagonal cell: <ul style="list-style-type: none"> "code" — 2-letter ASCII shape code, top-left corner. "A" and "edf" — asymmetry index and effective degrees of freedom, stacked bottom-left. "shape" — shape glyph (Unicode or ASCII per glyph_style), bottom-right corner. <p>Default $c("edf", "A")$. "code" and "shape" occupy distinct corners so both can be requested together. See janusplot_shape_hierarchy() for the full code list.</p>
shape_cutoffs	Named list of classification thresholds used to map the continuous shape indices into discrete shape_category labels; see janusplot_shape_cutoffs() .
show_shape_legend	Logical. If TRUE (default), attach a standing shape-types legend plate below the matrix that illustrates every category in the taxonomy as a canonical thumbnail spline. Independent of annotations.
glyph_style	One of "ascii" (default) or "unicode". Controls how cell shape glyphs render when "shape" is included in annotations. Default is "ascii" for maximum portability across typesetting pipelines; switch to "unicode" only when the target font is known to cover the curve glyph set.
labels	One of "border" (default), "diagonal", or "none". Controls where variable names are rendered: <ul style="list-style-type: none"> "border" — names along the top (rotated per label_srt) and left margins of the matrix; diagonal cells are left blank. Mirrors corrrplot's <code>tl.pos = "lt"</code> convention. "diagonal" — names centred on the diagonal cells (the pre-0.1 layout). "none" — labels suppressed entirely; diagonal cells blank.
diagonal	One of "auto" (default), "blank", "name", or "density". Controls what is rendered in the diagonal cells of the matrix. <ul style="list-style-type: none"> "auto" — preserves the historical behaviour: variable name when labels = "diagonal", blank otherwise.

- "blank" — empty bordered panel (uniform grid reading).
- "name" — variable name centred in the cell, bold.
- "density" — kernel density of the variable filled in translucent grey, with a rug of raw values along the bottom edge. Mirrors the `GGally::ggpairs` convention; surfaces tail weight, bimodality, and support clipping that the pairwise smooths alone cannot reveal. Variable names should come from the border (`labels = "border"`, the default) when this mode is on.

<code>label_srt</code>	Numeric. Rotation (degrees) of top labels when <code>labels = "border"</code> . Default 45; set to 0 for horizontal or 90 for vertical. Ignored when <code>labels != "border"</code> .
<code>label_cex</code>	Positive numeric multiplier on the border-label font size. Default 1. Ignored when <code>labels = "none"</code> .
<code>signif_glyph</code>	Logical. If TRUE (default), annotate cells with <code>· / * / **</code> reflecting the smooth's F-test p-value.
<code>show_asymmetry</code>	Deprecated. Use annotations instead (" <code>A</code> " <code>%in%</code> annotations). When supplied, a soft deprecation warning fires and the argument is merged into annotations.
<code>na_action</code>	One of "pairwise" (default; per-cell complete observations) or "complete" (listwise; all cells use the same rows).
<code>parallel</code>	Logical. If TRUE, use <code>future.apply::future_mapply()</code> to fit pairs in parallel. Requires the <code>future.apply</code> package and a user-configured <code>future::plan()</code> . Default FALSE.
<code>with_data</code>	Logical. If TRUE, return a two-element list <code>list(plot, data)</code> where <code>data</code> is a flat per-cell summary (one row per off-diagonal cell) of everything the plot displays. The <code>data</code> element is always a plain <code>data.frame</code> (base R — no <code>data.table</code> dependency). Default FALSE — in which case only the <code>ggplot</code> is returned.
<code>text_scale_diag</code>	Positive numeric multiplier applied to the diagonal variable-name labels. Default 1. Diagonal labels additionally auto-shrink for long variable names (<code>nchar(var) > 10</code>) so they fit the cell regardless of this value.
<code>text_scale_off_diag</code>	Positive numeric multiplier applied to all off-diagonal annotations (n / EDF readouts, significance glyphs, asymmetry-index labels). Default 1. Use <code>< 1</code> when cells are small and the annotations crowd the fit line; use <code>> 1</code> for presentation plots.
<code>show_glossary</code>	Logical. If TRUE (default), attach a multi-line caption below the matrix describing the on-plot abbreviations (n, EDF, A, fill encoding, significance glyphs). Only keys actually displayed are listed.
<code>glossary_scale</code>	Positive numeric multiplier on the glossary caption font size. Default 1.
<code>k_check_thresholds</code>	Named list giving the three flag-criterion thresholds used by <code>mgcv::k.check()</code> -style basis-dimension diagnostics. Required entries: <code>edf_ratio</code> (Wood's \widehat{edf}/k' ratio above which the smooth is too close to its basis cap), <code>k_index</code> (residual-difference variance ratio below which the basis appears underspecified), and <code>p</code> (the simulation p-value below which the basis-deficiency signal is significant). Defaults — <code>edf_ratio = 0.9</code> , <code>k_index = 1.0</code> , <code>p = 0.05</code> — track <code>mgcv::gam.check()</code> and Wood (2017) §5.9.

auto_refit_k	Logical. If TRUE, every cell whose Wood trifacta flags an underfit is refit with a doubling-k loop until either the flag clears, the per-cell unique-x cap is reached, or k_max_iter iterations have passed. Default FALSE — the diagnostic (k_check_status, k_flag, k_prime, k_index, k_p) is always computed and surfaced regardless of this flag, but the refit is opt-in because it can multiply wall time on pathological data.
k_max_iter	Integer. Maximum number of doublings allowed per cell when auto_refit_k = TRUE. Default 2L (so a cell that starts at the mgcv default k = 10 will visit at most k = 20 and then k = 40, capped by the per-cell unique-x limit). Ignored when auto_refit_k = FALSE.
compact	One of "auto" (default), "always", or "never". Controls scale-aware content suppression per cell: <ul style="list-style-type: none"> "auto" — tier 0 at n_var < compact_threshold (the v0.1.0 behaviour); progressively suppresses scatter, then CI, then annotations, then the spline itself as n_var crosses the compact_levels ladder. The matrix remains readable as k grows toward 25–30 by trading detail for legibility. "always" — force at least tier 1 regardless of n_var. Useful for very dense fixed-size renders. "never" — force tier 0 regardless of n_var. Useful for reproducing v0.1.0 figures on large matrices.
compact_threshold	Integer. The n_var value at which tier 1 (drop scatter) auto-activates under compact = "auto". Default 12L, anchored on the 150 × 150 px-per-cell pixel budget at typical 6"×6" 300 DPI R Journal figures.
compact_levels	Optional named list with entries t1, t2, t3 overriding the auto-tier ladder. Defaults derive from compact_threshold: t1 = compact_threshold, t2 = compact_threshold + 6, t3 = compact_threshold + 13. NULL (default) uses these derived defaults.
focus_by	One of NA (default — no filter), "asymmetry", "edf", "k_flag", or "non_linearity" (defined as edf - 1). When set, cells whose chosen metric falls below focus_threshold are rendered in grey85 at alpha focus_dim_alpha; the matrix shape is preserved so attention drains visually to high-metric cells. This is a visual filter, not a statistical one — the underlying fits are unchanged and the with_data table carries every cell.
focus_threshold	Either a quantile-string like "q90" (default) or a numeric cutoff. The quantile is taken over the non-NA distribution of focus_by across all off-diagonal cells in the matrix. Ignored when focus_by = NA.
focus_dim_alpha	Numeric in [0, 1]. Alpha applied to the grey85 wash on unfocused cells. Default 0.25. Ignored when focus_by = NA.
axes	One of "original" (default), "standardised", "centred", or "rank". Rendering-only knob — the underlying mgcv::gam fits are byte-identical across all four modes (verifiable via digest::digest() on the fit list); the transformation lives entirely inside the cell renderer and propagates to (a) the raw scatter, (b) the spline prediction grid, (c) the CI ribbon, and (d) the variable label on the matrix border. Use:

- "original" — raw units. Maximum interpretability per cell. v0.1.0 behaviour.
- "standardised" — $(x - \text{mean}(x)) / \text{sd}(x)$ per variable. Border label becomes e.g. "mpg (z)". Pairs scaled into a comparable visual range; useful at $k \geq 15$ when raw-unit panels look disparate.
- "centred" — $x - \text{mean}(x)$ per variable. Border label becomes e.g. "mpg (centred)". Preserves units while anchoring the origin.
- "rank" — empirical-CDF-based rank, scaled to $[\emptyset, n]$ per variable. Border label becomes "rank(mpg)". Sanity-check view: collapses outliers; if the smooth changes shape vs "original" the relationship is monotone-but-not-linear.

At compact tier 3 ($n_{\text{var}} \geq 25$ under `compact = "auto"`), the cells render only colour fill + shape-class glyph — no curve, no scatter — so axes becomes a **documented no-op** (the border labels still pick up the mode suffix).

<code>save_as</code>	Optional file path with extension. When set, the final assembled plot is written to this path via <code>ggplot2::ggsave()</code> ; the device is inferred from the extension. Supported extensions: <code>.png</code> , <code>.pdf</code> , <code>.svg</code> , <code>.jpg</code> / <code>.jpeg</code> , <code>.tif</code> / <code>.tiff</code> , <code>.eps</code> , <code>.ps</code> , <code>.bmp</code> . Default NULL (no file written; <code>janusplot()</code> still returns the <code>ggplot</code>). Width / height default to <code>pmax(6, 0.65 * k_n)</code> inches each — square aspect, scaling with matrix dimension.
<code>save_width</code>	Numeric. Override width (inches) for <code>save_as</code> . Default NULL uses the auto-resolved square.
<code>save_height</code>	Numeric. Override height (inches) for <code>save_as</code> . Default NULL uses the auto-resolved square.
<code>save_dpi</code>	Integer. Raster DPI for <code>save_as</code> when the inferred device is bitmap (<code>png/jpg/tiff/bmp</code>). Default 300, matching the R Journal target.
<code>...</code>	Additional arguments passed to <code>mgcv::gam()</code> .

Value

If `with_data = FALSE` (default), a `ggplot2::ggplot` object (via `patchwork::wrap_plots()`) carrying a top-of-matrix title that names the displayed quantity ("Direct fit", "First derivative f' ", or "Second derivative f'' "). If `with_data = TRUE`, a list with two elements: `plot` (the `ggplot`) and `data` (a tidy table with columns `var_x`, `var_y`, `position`, `n_used`, `edf`, `pvalue`, `signif`, `dev_exp`, `asymmetry_index`, `cor_pearson`, `cor_spearman`, `cor_kendall`, `tie_ratio`, `monotonicity_index`, `convexity_index`, `n_turning_points`, `n_inflections`, `flat_range_ratio`, `shape_category`, `colour_value`, `display`, one row per off-diagonal cell). The display column tags which quantity the call rendered, so separate calls for `fit` / `d1` / `d2` yield comparable, stackable tables. Derivative *curves* themselves (grid of x , fitted $\hat{f}^{(k)}$, SE) live on `janusplot_data()` — see there.

See Also

[janusplot_data\(\)](#) for the raw per-cell fits + metrics.

Other smooth-associations: [janusplot_data\(\)](#)

Examples

```
# Minimal runnable example – 3 variables, 6 asymmetric pairwise GAM fits.
janusplot(mtcars[, c("mpg", "hp", "wt")])

# Heteroscedastic DGP: Pearson r is ~ 0.9 but the inverse fit is
# clearly non-linear, yielding asymmetry index > 0.5.
set.seed(2026L)
n <- 200L
x1 <- stats::runif(n, 0, 10)
x2 <- x1 + stats::rnorm(n, sd = 0.2 * x1)
janusplot(data.frame(x1 = x1, x2 = x2, x3 = stats::rnorm(n)))

# A single matrix renders a single quantity. To compare the fit
# against its derivatives, render three calls and place them
# side-by-side; each call's title makes the quantity explicit.
set.seed(2026L)
xs <- stats::runif(300L, -3, 3)
df <- data.frame(
  x = xs,
  y1 = sin(xs) + stats::rnorm(300L, sd = 0.3),
  y2 = xs^2 + stats::rnorm(300L, sd = 0.6)
)
janusplot(df, display = "fit")
janusplot(df, display = "d1")
janusplot(df, display = "d2")

# Simultaneous CI bands on a derivative panel, per Simpson (2018).
janusplot(df, display = "d1", derivative_ci = "simultaneous")
```

janusplot_data

Raw GAM fits and per-cell metrics for a smoothed-association matrix

Description

[Experimental]

Companion to `janusplot()` returning the raw list of GAM fits plus per-cell metrics (EDF, F-test p-value, deviance explained, asymmetry index, pairwise correlations, shape descriptors) without constructing the `ggplot`. Useful for custom rendering or downstream analysis.

Usage

```
janusplot_data(
  data,
  vars = NULL,
  adjust = NULL,
  method = NULL,
  k = -1L,
```

```

bs = "tp",
na_action = c("pairwise", "complete"),
parallel = FALSE,
keep_fits = FALSE,
derivatives = integer(),
derivative_ci = c("pointwise", "none", "simultaneous"),
derivative_ci_nsim = 1000L,
n_grid = NULL,
shape_cutoffs = janusplot_shape_cutoffs(),
k_check_thresholds = NULL,
auto_refit_k = FALSE,
k_max_iter = 2L,
engine = c("bam", "gam"),
discrete = FALSE,
nthreads = 1L,
...
)

```

Arguments

<code>data</code>	A data frame with numeric columns to include.
<code>vars</code>	Character vector of column names to use. <code>NULL</code> (default) uses all numeric columns in data. Non-numeric columns trigger an error listing offenders.
<code>adjust</code>	A one-sided formula RHS giving additional covariates and/or random effects to include in every pairwise GAM. For example, <code>adjust = ~ s(age) + s(site, bs = "re")</code> fits <code>gam(y ~ s(x) + s(age) + s(site, bs = "re"))</code> for each pair. Default <code>NULL</code> fits unadjusted pairwise smooths.
<code>method</code>	Smoothing-parameter estimation method passed to the chosen fitting backend. Default <code>NULL</code> resolves per-engine: <code>"fREML"</code> for engine = <code>"bam"</code> (mgcv's recommended at scale), <code>"REML"</code> for engine = <code>"gam"</code> (the v0.1.0 behaviour). Pass any valid mgcv method string to override.
<code>k</code>	Integer, or named list mapping variable names to integers. Basis dimension for <code>s()</code> . Default <code>-1L</code> (mgcv's automatic choice).
<code>bs</code>	Basis type for <code>s()</code> . Default <code>"tp"</code> (thin plate).
<code>na_action</code>	One of <code>"pairwise"</code> (default; per-cell complete observations) or <code>"complete"</code> (listwise; all cells use the same rows).
<code>parallel</code>	Logical. If <code>TRUE</code> , use <code>future.apply::future_mapply()</code> to fit pairs in parallel. Requires the <code>future.apply</code> package and a user-configured <code>future::plan()</code> . Default <code>FALSE</code> .
<code>keep_fits</code>	Logical. If <code>TRUE</code> , retain full <code>mgcv::gam()</code> model objects in the return (large memory footprint for <code>k</code> above <code>~15</code>). Default <code>FALSE</code> — retains summary metrics and prediction grids only.
<code>derivatives</code>	Integer vector of derivative orders to compute on every pair (subset of <code>1:2</code>). Default <code>integer()</code> — no derivatives. Unlike <code>janusplot()</code> , the data companion can return multiple orders from a single call for programmatic analysis; pass <code>c(1L, 2L)</code> to surface both.

derivative_ci	<p>One of "none" (default), "pointwise", or "simultaneous". Controls whether — and how — a 95% confidence ribbon is drawn underneath the derivative curve when <code>display %in% c("d1", "d2")</code>. Ignored when <code>display = "fit"</code>.</p> <ul style="list-style-type: none"> • "none" — no ribbon. The curve and the zero reference line are all you see. Default, because pointwise ribbons overshoot nominal coverage as a joint region and can invite over-reading of local features. • "pointwise" — 95% pointwise ribbon from $\sqrt{\text{diag}(DV_p D^T)}$ (Wood 2017 §7.2.4). Valid marginally; not a simultaneous statement. • "simultaneous" — 95% simultaneous band via the Monte Carlo construction of Ruppert, Wand & Carroll (2003) popularised for GAMs by Simpson (2018, <i>Frontiers Ecol. Evol.</i> 6:149): draw B samples $\tilde{\beta} \sim \mathcal{N}(\hat{\beta}, V_p)$, compute $\max_x D_i(\tilde{\beta} - \hat{\beta}) /se_i$, and use the $(1 - \alpha)$ quantile as a critical multiplier on the pointwise SE. Valid for feature localisation ("where is $\hat{f}'(x)$ significantly non-zero").
derivative_ci_nsim	<p>Integer. Number of Monte Carlo samples used when <code>derivative_ci = "simultaneous"</code>. Default 1000L — a compromise between coverage accuracy (Simpson 2018 uses 10000) and CPU budget across every pair in a medium-sized matrix. Ignored for any other <code>derivative_ci</code>.</p>
n_grid	<p>Integer or NULL. Number of equally-spaced points used to evaluate each fitted smooth (and its derivatives). Default NULL resolves to 100 when <code>display = "fit"</code> and 200 otherwise, because finite-difference second derivatives visibly degrade below ~ 150 points on moderate-k smooths. Supplying <code>n_grid</code> directly overrides both defaults. Larger grids shift the numerical shape-metric values (M, C, turning / inflection counts) slightly because they are computed on this same grid. Shapes and asymmetry are the primary reading; M, C and the counts are secondary diagnostics and the grid-induced drift is tolerable.</p>
shape_cutoffs	<p>Named list of classification thresholds used to map the continuous shape indices (<code>monotonicity_index</code>, <code>convexity_index</code>) into discrete <code>shape_category</code> labels. Defaults from <code>janusplot_shape_cutoffs()</code>.</p>
k_check_thresholds	<p>Named list giving the three flag-criterion thresholds used by <code>mgcv::k.check()</code>-style basis-dimension diagnostics. Required entries: <code>edf_ratio</code> (Wood's $\widehat{\text{edf}}/k'$ ratio above which the smooth is too close to its basis cap), <code>k_index</code> (residual-difference variance ratio below which the basis appears underspecified), and <code>p</code> (the simulation p-value below which the basis-deficiency signal is significant). Defaults — <code>edf_ratio = 0.9</code>, <code>k_index = 1.0</code>, <code>p = 0.05</code> — track <code>mgcv::gam.check()</code> and Wood (2017) §5.9.</p>
auto_refit_k	<p>Logical. If TRUE, every cell whose Wood trifecta flags an underfit is refit with a doubling-k loop until either the flag clears, the per-cell unique-x cap is reached, or <code>k_max_iter</code> iterations have passed. Default FALSE — the diagnostic (<code>k_check_status</code>, <code>k_flag</code>, <code>k_prime</code>, <code>k_index</code>, <code>k_p</code>) is always computed and surfaced regardless of this flag, but the refit is opt-in because it can multiply wall time on pathological data.</p>
k_max_iter	<p>Integer. Maximum number of doublings allowed per cell when <code>auto_refit_k = TRUE</code>. Default 2L (so a cell that starts at the <code>mgcv</code> default <code>k = 10</code> will visit at most</p>

	k = 20 and then k = 40, capped by the per-cell unique-x limit). Ignored when <code>auto_refit_k = FALSE</code> .
engine	One of "bam" (default, new in v0.1.1) or "gam". Selects mgcv's fitting backend: <ul style="list-style-type: none"> "bam" — <code>mgcv::bam()</code>. Block-Lanczos solve + fREML estimation + lower memory. ~3-10x speedup at janusplot's scale (k = 15-25 vars, 600+ pairwise fits per call). The default, and the one non-byte-identical change in v0.1.1: fREML differs from REML by ~1-3% in EDF on identical data, so the asymmetry index may shift by similar amounts vs v0.1.0 output. Recoverable verbatim via <code>engine = "gam"</code>. "gam" — <code>mgcv::gam()</code>. The v0.1.0 backend. Use for backward-compat reproduction, very small n (< 200) where bam's setup overhead exceeds its solve gain, or methodologically sensitive contexts that require REML rather than fREML.
discrete	Logical. bam-only opt-in to mgcv's covariate-discretisation optimisation. Further ~2-5x speedup at the cost of small (sub-pixel at typical janusplot resolution) prediction-shift. Default FALSE. Ignored when <code>engine = "gam"</code> .
nthreads	Integer. bam-only intra-fit threading. Default 1L to avoid oversubscription when combined with <code>parallel = TRUE</code> (<code>future.apply</code> fans out pair fits across cores; <code>nthreads > 1</code> within each pair would double-book CPUs). Raise above 1 only when <code>parallel = FALSE</code> . Ignored when <code>engine = "gam"</code> .
...	Additional arguments passed to <code>mgcv::gam()</code> .

Value

A list with components:

`vars` Character vector of variables used, in plotted order.

`pairs` List of per-pair results. Each element has `i`, `j`, `var_i`, `var_j`, `fit_yx`, `fit_xy` (NULL if `keep_fits = FALSE`), `pred_yx`, `pred_xy` (data frames with `x`, `fit`, `se`, `lo`, `hi`), `edf_yx`, `edf_xy`, `pvalue_yx`, `pvalue_xy`, `dev_exp_yx`, `dev_exp_xy`, `n_used`, `asymmetry_index`, plus Pearson / Spearman / Kendall correlations (`cor_pearson`, `cor_spearman`, `cor_kendall`), the maximum tie ratio across x and y (`tie_ratio`), and per-direction shape descriptors (`monotonicity_index_yx`, `convexity_index_yx`, `monotonicity_index_xy`, `convexity_index_xy`, `n_turning_yx`, `n_inflect_yx`, `n_turning_xy`, `n_inflect_xy`, `shape_yx`, `shape_xy`). When `derivatives` is non-empty, each pair additionally carries `deriv_yx` and `deriv_xy`, each a named list keyed by order ("1", "2") whose entries are data frames with columns `x`, `fit`, `se`, `lo`, `hi`, `ci_type` matching the schema of `pred_yx` / `pred_xy`. The `ci_type` column records whether the `lo` / `hi` columns are "pointwise" (default), "simultaneous" (Ruppert–Wand–Carroll / Simpson 2018 critical-multiplier bands), or "none". When `derivative_ci = "simultaneous"`, each derivative frame also carries a "crit_multiplier" attribute giving the MC-derived critical multiplier used. See [janusplot_shape_metrics\(\)](#) for the definition of the monotonicity and convexity indices.

`call` Match call.

See Also

[janusplot\(\)](#) for the ggplot front-end, [janusplot_shape_metrics\(\)](#) for the shape-metric primitives.

Other smooth-associations: [janusplot\(\)](#)

Examples

```
# Per-pair fits + metrics on a small mtcars slice
out <- janusplot_data(mtcars[, c("mpg", "hp", "wt")])
out$pairs[[1L]]$asymmetry_index
out$pairs[[1L]]$cor_spearman
out$pairs[[1L]]$shape_yx
```

`janusplot_shape_cutoffs`

Default cutoff thresholds for shape_category classification

Description

[Experimental]

Returns the named list of thresholds used to map the continuous monotonicity (M) and convexity (C) indices (plus inflection counts) into a discrete `shape_category`. Expose so callers can override individual thresholds or pass a fully custom list to [janusplot\(\)](#) / [janusplot_shape_metrics\(\)](#).

Usage

```
janusplot_shape_cutoffs(...)
```

Arguments

... Optional named overrides to merge into the defaults.

Value

A named list with numeric thresholds:

- `mono_strong` |M| threshold for a strictly monotone smooth (default 0.9).
- `mono_mod` |M| threshold for a curved-but-monotone smooth (default 0.5).
- `mono_nonmono` |M| below this is considered non-monotone (default 0.3).
- `mono_s` |M| threshold for labelling an S-shape (default 0.5).
- `curv_low` |C| below this is considered near-linear curvature (default 0.2).
- `curv_mod` |C| threshold for a clearly curved monotone (default 0.5).
- `curv_strong` |C| threshold for a U-shape / inverted-U shape (default 0.5).
- `flat` $\text{range}(\text{fit}) / \text{sd}(y)$ below this is called flat (default 0.05).

See Also

Other shape-metrics: [janusplot_shape_hierarchy\(\)](#), [janusplot_shape_metrics\(\)](#)

Examples

```
janusplot_shape_cutoffs()
janusplot_shape_cutoffs(curv_mod = 0.6, flat = 0.02)
```

```
janusplot_shape_hierarchy
      Shape-category taxonomy table
```

Description**[Experimental]**

Return the full janusplot shape taxonomy as a data frame with four hierarchy columns plus presentation fields. The taxonomy is the single source of truth consumed by the classifier, the cell renderer, the legend plate, and the `janusplot_data()` output.

Hierarchy columns (finest → coarsest):

`category` 24-way fine label (`linear_up`, `skewed_peak`, `bimodal`, ...). Computed per cell by `janusplot()`.

`code` Unique two-letter ASCII shorthand (safe on any font or typesetting pipeline) — e.g. `lu` for `linear_up`.

`archetype` Seven-family grouping: `monotone_linear`, `monotone_curved`, `unimodal`, `wave`, `multimodal`, `chaotic`, `degenerate`.

`monotonic` Three-way coarse classification: `monotone` / `non_monotone` / `degenerate`.

`linear` Binary: `linear` / `non_linear` / `degenerate`.

The broader tiers (`linear/non-linear`, `monotone/non-monotone`) are textbook calculus; the archetype layer maps cleanly to shape-constrained regression vocabulary (Pya & Wood 2015; Meyer 2008) and to dose-response shape categories (Calabrese 2008; Calabrese & Baldwin 2001). The (T, I) dispatch underlying each fine category is a coarsened Morse-theoretic critical-point classification (Milnor 1963).

Usage

```
janusplot_shape_hierarchy()
```

Value

A data frame with 24 rows and columns `category`, `code`, `archetype`, `monotonic`, `linear`, `glyph`, `ascii`, `label`, `gloss`.

References

- Calabrese, E. J. (2008). Hormesis: why it is important to toxicology and toxicologists. *Environmental Toxicology and Chemistry*, **27**(7), 1451–1474.
- Meyer, M. C. (2008). Inference using shape-restricted regression splines. *Annals of Applied Statistics*, **2**(3), 1013–1033.
- Milnor, J. (1963). *Morse Theory*. Princeton University Press.
- Pyra, N., & Wood, S. N. (2015). Shape constrained additive models. *Statistics and Computing*, **25**(3), 543–559.

See Also

Other shape-metrics: [janusplot_shape_cutoffs\(\)](#), [janusplot_shape_metrics\(\)](#)

Examples

```
tax <- janusplot_shape_hierarchy()
head(tax[, c("category", "code", "archetype", "monotonic", "linear")])
# Count how many categories live in each archetype
table(tax$archetype)
```

janusplot_shape_metrics

Shape metrics for a fitted univariate smooth

Description

[Experimental]

Compute the continuous monotonicity and convexity indices, inflection and turning-point counts, and rule-based shape category for a fitted univariate smooth. Works on either a per-pair fit object returned from the janusplot internal machinery or a freshly fitted `mgcv::gam()` with a single `s()` term.

Both indices are bounded in $[-1, 1]$ and weighted by the empirical density of the predictor:

- `monotonicity_index` (paper symbol M). Let f be the fitted smooth evaluated on a dense grid of `n_grid` equally-spaced points across the predictor range, f' its numerical first derivative, and w the empirical density of the predictor on the same grid with $\text{sum}(w) = 1$. Then $\text{monotonicity_index} = \text{sum}(w * f') / \text{sum}(w * |f'|)$ in $[-1, 1]$. +1 is strictly increasing, -1 strictly decreasing, 0 non-monotone.
- `convexity_index` (paper symbol C). With f'' the numerical second derivative on the same grid, $\text{convexity_index} = \text{sum}(w * f'') / \text{sum}(w * |f''|)$ in $[-1, 1]$. +1 is globally convex (bowl-up), -1 globally concave (bowl-down), 0 inflection-dominated (S-curve, sine, flat).

Both indices are scale-invariant (replacing $y \rightarrow a*y + b$ leaves them unchanged) and density-weighted so they describe the smooth *where the data actually live*, not extrapolated tails.

Usage

```
janusplot_shape_metrics(
  fit,
  x_name = NULL,
  newdata = NULL,
  n_grid = 200L,
  cutoffs = janusplot_shape_cutoffs()
)
```

Arguments

<code>fit</code>	Either a list returned by a janusplot pair-fit helper (must contain <code>pred</code> and <code>raw</code>), or a fitted <code>mgcv::gam()</code> with a single $s(x)$ term.
<code>x_name</code>	Character. Column name of the predictor when <code>fit</code> is a <code>mgcv::gam()</code> object. Ignored for pair-fit lists.
<code>newdata</code>	Optional data frame supplying the raw predictor values used for density weighting when <code>fit</code> is a <code>mgcv::gam()</code> object. If <code>NULL</code> , the model frame is used.
<code>n_grid</code>	Integer. Prediction grid length when <code>fit</code> is a <code>mgcv::gam()</code> object. Default 200L.
<code>cutoffs</code>	Named list of classification thresholds; see <code>janusplot_shape_cutoffs()</code> . Default uses package defaults.

Value

A named list with components:

`monotonicity_index` M in $[-1, 1]$. See Description.

`convexity_index` C in $[-1, 1]$. See Description.

`n_turning_points` Integer count of lobe-mass-weighted sign changes of f' . Equals the number of interior extrema.

`n_inflections` Integer count of lobe-mass-weighted sign changes of f'' .

`flat_range_ratio` $\text{range}(f) / \text{sd}(y)$ — small values indicate a degenerate flat smooth.

`shape_category` One of 24 labels from `janusplot_shape_hierarchy()` dispatched on `(n_turning_points, n_inflect)` with `(monotonicity_index, convexity_index)` disambiguation for the monotone case.

See Also

[janusplot_shape_cutoffs\(\)](#), [janusplot\(\)](#), [janusplot_data\(\)](#).

Other shape-metrics: [janusplot_shape_cutoffs\(\)](#), [janusplot_shape_hierarchy\(\)](#)

Examples

```
# On a fitted gam
set.seed(2026L)
n <- 200L
x <- stats::runif(n, 0, 10)
y <- log1p(x) + stats::rnorm(n, sd = 0.3)
d <- data.frame(x = x, y = y)
```

```
fit <- mgcv::gam(y ~ s(x), data = d, method = "REML")
janusplot_shape_metrics(fit, x_name = "x", newdata = d)
```

```
janusplot_shape_sensitivity
```

Shape-recognition sensitivity study

Description

[Experimental]

Run a full-factorial sensitivity sweep for the janusplot 24-category shape classifier. For each combination of ground-truth shape, sample size n , noise level σ , and replicate, the sweep:

1. Generates n points from the noiseless canonical curve on $[0, 1]$ + Gaussian noise with SD = σ (fraction of the y-range, so signal-to-noise is comparable across shapes).
2. Fits `mgcv::gam(y ~ s(x), method = "REML")`.
3. Runs `janusplot_shape_metrics()` to classify the fitted smooth.
4. Records correctness at both the fine (24-category) and archetype (7-family) levels.

The function is the package-native implementation of `simulation/scripts/scenario_4_shape_recognition.R`. A small precomputed dataset is shipped as `shape_sensitivity_demo` for downstream examples without requiring users to re-run the sweep.

Usage

```
janusplot_shape_sensitivity(
  shapes = NULL,
  n_grid = c(50L, 100L, 200L, 500L),
  sigma_grid = c(0.02, 0.05, 0.1, 0.2, 0.4),
  n_rep = 200L,
  cutoffs = janusplot_shape_cutoffs(),
  parallel = FALSE,
  seed = 2026L,
  verbose = interactive()
)
```

Arguments

<code>shapes</code>	Character vector of ground-truth names from <code>janusplot_shape_sensitivity_shapes()</code> . Default NULL \rightarrow all 14.
<code>n_grid</code>	Integer vector of sample sizes. Default <code>c(50L, 100L, 200L, 500L)</code> .
<code>sigma_grid</code>	Numeric vector of noise levels (fraction of the y-range). Default <code>c(0.02, 0.05, 0.10, 0.20, 0.40)</code> .
<code>n_rep</code>	Integer. Replicates per cell. Default 200L.
<code>cutoffs</code>	Named list of classification thresholds; see <code>janusplot_shape_cutoffs()</code> .

parallel	Logical. If TRUE and <code>future.apply</code> is installed, <code>dispatch</code> replicates in parallel. The caller is responsible for configuring <code>future::plan()</code> (e.g. <code>future::plan(future::multisession)</code>).
seed	Integer. Base seed — each fit uses <code>seed + row_index</code> so results are reproducible and cell-permutation-invariant.
verbose	Logical. Print progress messages to the console. Default is <code>interactive()</code> .

Value

A data frame with one row per fit. Columns:

`truth` Ground-truth shape name.

`n` Sample size for this fit.

`sigma` Noise level for this fit.

`seed` RNG seed used.

`predicted` Classifier output at the fine (24-category) level.

`correct` Logical — does `predicted == truth`?

`archetype_truth` Expected archetype for truth.

`archetype_pred` Archetype of predicted.

`archetype_correct` Logical — archetype-level correctness.

`monotonicity_index` Monotonicity index M (see [janusplot_shape_metrics\(\)](#)).

`convexity_index` Convexity index C (see [janusplot_shape_metrics\(\)](#)).

`n_turn, n_inflect` Recovered turning-point and inflection counts.

`error` "gam_fit_failed" when `mgcv::gam()` errored; NA otherwise.

See Also

[janusplot_shape_sensitivity_summary\(\)](#), [janusplot_shape_sensitivity_plot\(\)](#), [janusplot_shape_sensitivity_shape_sensitivity_demo](#).

Other shape-sensitivity: [janusplot_shape_sensitivity_plot\(\)](#), [janusplot_shape_sensitivity_shapes\(\)](#), [janusplot_shape_sensitivity_summary\(\)](#)

Examples

```
# Tiny-run smoke test (< 2 seconds): 3 shapes x 2 n x 2 sigma x 5 reps.
res <- janusplot_shape_sensitivity(
  shapes = c("linear_up", "u_shape", "wave"),
  n_grid = c(100L, 200L),
  sigma_grid = c(0.05, 0.20),
  n_rep = 5L,
  verbose = FALSE
)
head(res)
janusplot_shape_sensitivity_summary(res, level = "archetype")
```

`janusplot_shape_sensitivity_plot`*Visualise a shape-sensitivity sweep*

Description

[Experimental]

Produce one of four diagnostic plots from the raw data frame returned by `janusplot_shape_sensitivity()`:

"confusion_fine" 24 x (lshapes) confusion matrix at the fine category level — rows = ground truth, columns = predicted, cells coloured by $P(\text{pred} \mid \text{truth})$.

"confusion_archetype" 7 x 7 confusion matrix at the archetype level.

"accuracy_grid" per-shape heatmap of archetype-level accuracy across the (n, sigma) design.

"recovery_curves" accuracy as a function of sigma, one line per sample size, faceted by shape.

Usage

```
janusplot_shape_sensitivity_plot(  
  results,  
  type = c("confusion_fine", "confusion_archetype", "accuracy_grid", "recovery_curves")  
)
```

Arguments

<code>results</code>	Data frame from <code>janusplot_shape_sensitivity()</code> or the precomputed <code>shape_sensitivity_demo</code> .
<code>type</code>	One of "confusion_fine", "confusion_archetype", "accuracy_grid", or "recovery_curves".

Value

A `ggplot2::ggplot` object.

See Also

Other shape-sensitivity: `janusplot_shape_sensitivity()`, `janusplot_shape_sensitivity_shapes()`, `janusplot_shape_sensitivity_summary()`

Examples

```
data("shape_sensitivity_demo", package = "janusplot")  
janusplot_shape_sensitivity_plot(shape_sensitivity_demo,  
  "recovery_curves")
```

`janusplot_shape_sensitivity_shapes`*Canonical ground-truth shapes for the sensitivity study*

Description**[Experimental]**

Return the names of every canonical ground-truth shape that `janusplot_shape_sensitivity()` can simulate from. Fourteen shapes spanning five archetypes (`monotone_linear`, `monotone_curved`, `unimodal`, `wave`, `multimodal`). The chaotic and degenerate archetypes are out of scope (no realistic deterministic generator).

Usage

```
janusplot_shape_sensitivity_shapes()
```

Value

Character vector of length 14 — the generator names.

See Also

[janusplot_shape_sensitivity\(\)](#), [janusplot_shape_hierarchy\(\)](#).

Other shape-sensitivity: [janusplot_shape_sensitivity\(\)](#), [janusplot_shape_sensitivity_plot\(\)](#), [janusplot_shape_sensitivity_summary\(\)](#)

Examples

```
janusplot_shape_sensitivity_shapes()
```

`janusplot_shape_sensitivity_summary`*Summarise a shape-sensitivity sweep*

Description**[Experimental]**

Aggregate the raw output of `janusplot_shape_sensitivity()` into a per-cell mean-accuracy table at either the fine (24-category) or archetype (7-family) level.

Usage

```
janusplot_shape_sensitivity_summary(results, level = c("fine", "archetype"))
```

Arguments

`results` Data frame returned by `janusplot_shape_sensitivity()`.
`level` One of "fine" (default) or "archetype".

Value

A data frame with columns `truth`, `n`, `sigma`, `accuracy`.

See Also

Other shape-sensitivity: `janusplot_shape_sensitivity()`, `janusplot_shape_sensitivity_plot()`, `janusplot_shape_sensitivity_shapes()`

Examples

```
data("shape_sensitivity_demo", package = "janusplot")
head(janusplot_shape_sensitivity_summary(shape_sensitivity_demo,
                                       level = "archetype"))
```

shape_sensitivity_demo

Precomputed shape-recognition sensitivity results (demo)

Description

Raw output from a small-footprint invocation of `janusplot_shape_sensitivity()`. Shipped so users can explore the sensitivity API and regenerate every figure in the `shape-recognition-sensitivity` vignette without having to re-run the sweep themselves. Regenerated via `data-raw/shape_sensitivity_demo.R`.

Design:

- **Shapes** (6, one per non-degenerate archetype): `linear_up`, `concave_up`, `u_shape`, `inverted_u`, `wave`, `bimodal`.
- **Sample sizes** (3): `c(100, 200, 500)`.
- **Noise levels** (4): `c(0.05, 0.10, 0.20, 0.40)` fraction of y-range.
- **Replicates**: 30.
- **Total fits**: 2160.
- **Seed**: 2026.

Usage

```
shape_sensitivity_demo
```

Format

A data frame with 2160 rows and 14 columns — see the "Value" section of `janusplot_shape_sensitivity()` for the column schema.

See Also

[janusplot_shape_sensitivity\(\)](#), [janusplot_shape_sensitivity_plot\(\)](#), [janusplot_shape_sensitivity_summary\(\)](#)

Examples

```
data("shape_sensitivity_demo", package = "janusplot")
head(shape_sensitivity_demo)
janusplot_shape_sensitivity_plot(shape_sensitivity_demo,
                                "recovery_curves")
```

Index

- * **datasets**
 - shape_sensitivity_demo, [22](#)
 - * **shape-metrics**
 - janusplot_shape_cutoffs, [14](#)
 - janusplot_shape_hierarchy, [15](#)
 - janusplot_shape_metrics, [16](#)
 - * **shape-sensitivity**
 - janusplot_shape_sensitivity, [18](#)
 - janusplot_shape_sensitivity_plot, [20](#)
 - janusplot_shape_sensitivity_shapes, [21](#)
 - janusplot_shape_sensitivity_summary, [21](#)
 - * **smooth-associations**
 - janusplot, [2](#)
 - janusplot_data, [10](#)
- ggplot2::ggplot, [9, 20](#)
ggplot2::ggsave(), [9](#)
- janusplot, [2, 14](#)
janusplot(), [10, 13–15, 17](#)
janusplot_data, [9, 10](#)
janusplot_data(), [9, 17](#)
janusplot_shape_cutoffs, [14, 16, 17](#)
janusplot_shape_cutoffs(), [6, 12, 17, 18](#)
janusplot_shape_hierarchy, [14, 15, 17](#)
janusplot_shape_hierarchy(), [6, 17, 21](#)
janusplot_shape_metrics, [14, 16, 16](#)
janusplot_shape_metrics(), [13, 14, 18, 19](#)
janusplot_shape_sensitivity, [18, 20–22](#)
janusplot_shape_sensitivity(), [20–23](#)
janusplot_shape_sensitivity_plot, [19, 20, 21, 22](#)
janusplot_shape_sensitivity_plot(), [19, 23](#)
janusplot_shape_sensitivity_shapes, [19, 20, 21, 22](#)
janusplot_shape_sensitivity_shapes(), [18, 19](#)
janusplot_shape_sensitivity_summary, [19, 20, 21, 21](#)
janusplot_shape_sensitivity_summary(), [19, 23](#)
mgcv::bam(), [4, 13](#)
mgcv::gam(), [2, 4, 9, 11, 13, 16, 17](#)
patchwork::wrap_plots(), [9](#)
shape_sensitivity_demo, [18–20, 22](#)