Package: gatekeepR (via r-universe)

November 3, 2026

Title Reviewable Marker Gating and Phenotyping for Multiplexed Imaging

Version 1.1.0

Description Turns per-cell marker measurements from multiplexed tissue images into reviewed cell types and state calls. Declares a marker panel and an ordered phenotype hierarchy as data, decides per image whether each marker can be called at all, and estimates thresholds with documented methods (manual, density valley, density tail, two-component Gaussian mixture, population density crossing on an embedding and parent-specific crossing). A single classifier is shared by batch code and the review application. Reviewer decisions are kept in append-only ledgers, and a review is locked into a content-addressed export whose labels, decisions and deletion histories can be replayed and verified.

License MIT + file LICENSE

URL https://github.com/CTTIR/gatekeepR

BugReports https://github.com/CTTIR/gatekeepR/issues

Depends R (>= 4.1.0)

Imports cli, data.table, digest, fs, filelock, jsonlite, lifecycle, MASS, rlang, stats, utils

Suggests callr, cellspecR (>= 1.0.0), EBImage, dbscan, ggplot2, jsonvalidate, knitr, patchwork, pkgload, rmarkdown, shiny, testthat (>= 3.0.0), uwot, withr

Additional_repositories https://cttir.r-universe.dev

Config/testthat/edition 3

VignetteBuilder knitr

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

Config/Needs/website CTTIR/themakR

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake make libuv1-dev

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

Date/Publication 2026-11-03 11:21:03 UTC

RemoteUrl https://github.com/CTTIR/gatekeepR

RemoteRef HEAD

RemoteSha 4ecea65d4dd5104ef3f6b2b62457977e6191cee6

Contents

gk_app . . . 3
gk_apply_candidate_rules . . . 4
gk_callability . . . 5
gk_cellspec . . . 6
gk_classify . . . 7
gk_config . . . 8
gk_config_hash . . . 9
gk_correct . . . 10
gk_decide_structure . . . 11
gk_dispose . . . 12
gk_embed . . . 13
gk_example_config . . . 14
gk_example_path . . . 15
gk_export . . . 16
gk_fit_residual_model . . . 16
gk_fixed_signal_calls . . . 18
gk_fixed_state_diagnostics . . . 19
gk_fixed_threshold_calls . . . 20
gk_hierarchy . . . 21
gk_import_cuts . . . 22
gk_import_residual_model . . . 23
gk_ledgers . . . 24
gk_load_review . . . 25
gk_lock . . . 25
gk_min_support . . . 26
gk_panel . . . 27
gk_plot_density . . . 29
gk_plot_dotplot . . . 29
gk_plot_embedding . . . 30
gk_plot_map . . . 31
gk_plot_overview . . . 31
gk_plot_states . . . 32
gk_plot_thresholds . . . 32
gk_prepare . . . 33
gk_read_export . . . 34
gk_replace_review . . . 35

gk_replay 36
gk_revert_last 36
gk_review 37
gk_save_review 38
gk_set_cut 39
gk_set_state 40
gk_signal_policy 41
gk_simulate 42
gk_state_calls 43
gk_structures 44
gk_threshold_methods 45
gk_thresholds 46
gk_validate_config 47
gk_verify_export 48
gk_write_cellspec 49
gk_write_config 50
hierarchy-elements 51

Index **54**

---

gk_app *Launch the gatekeepR review application*

Description

Review threshold, structure, disposition and state decisions in a local Shiny session. Every change requires a reviewer and reason and appends a ledger entry. A locked download uses one verified snapshot; subsequent edits invalidate that download until a new snapshot is locked. Choosing `REVIEWED` is an explicit human assessment, not automatic panel validation.

Usage

```
gk_app(
  review = NULL,
  cellspec = NULL,
  config = NULL,
  workdir = NULL,
  export = NULL,
  reviewer = NULL,
  max_upload_mb = 2048,
  allow_local_paths = FALSE,
  max_plot_points = 50000L,
  ...
)
```

Arguments

review	An existing gk_review() object.
cellspec	A cellspec object or path used to build a review.
config	A gk_config() used with cellspec.
workdir	A saved review directory to resume, or a new checkpoint directory when supplying review or cellspec. Checkpoints are written only when Save checkpoint is clicked. Existing directories must be resumed with workdir alone, preventing replacement with an unrelated review.
export	A verified export directory to inspect read-only.
reviewer	Initial reviewer identifier; it can be changed in the app.
max_upload_mb	Reserved compatibility option. There is no upload view.
allow_local_paths	Allow cellspec to be a local path. Explicit workdir and export arguments are trusted host configuration; there are no browser-supplied local path inputs.
max_plot_points	Maximum cells displayed in the map. The cell table shows at most 500 cells, or this limit if it is smaller. Display limits never change analysis, ledgers or exported rows.
...	Reserved application options.

Details

Review state is private to each browser session. When a checkpoint directory is configured, only the first active browser session can save it. Separate R processes are additionally protected by the checkpoint ownership rules in gk_save_review(). The download is a gzip-compressed tar archive containing a review directory that can be checked with gk_verify_export().

Value

A shiny.appobj.

---

gk_apply_candidate_rules

Evaluate declarative rules on ternary candidate calls

Description

[Experimental]

Each rule has unique id, label, and optional named all and any lists. List entries specify allowed states for a named marker. All all conditions and at least one any condition must match; absent groups impose no condition. Later matching rules overwrite earlier labels. No expression is evaluated, and unavailable never implicitly means LOW. Labels are caller descriptions, not scientific approvals; authorization flags are always false.

Usage

```
gk_apply_candidate_rules(calls, rules, default_label)
```

Arguments

`calls`	Keyed character matrix containing HIGH, LOW or UNAVAILABLE.
`rules`	Ordered list of declarative rules as described above.
`default_label`	Nonempty label for rows matching no rule.

Value

Data frame with cell_id, label, rule_id, review_authorized and production_eligible. Both authorization fields are always FALSE.

See Also

Other classification: `gk_classify()`, `gk_fixed_signal_calls()`, `gk_state_calls()`, `gk_structures()`

Examples

```
calls <- matrix(c("HIGH", "UNAVAILABLE"), 2, 1,
  dimnames = list(c("one", "two"), "A"))
rules <- list(list(id = "signal", label = "candidate", all = list(A = "HIGH")))
gk_apply_candidate_rules(calls, rules, "unresolved")
```

`gk_callability` *Callability of markers per image*

Description

[Experimental]

Reports, for every image and marker, whether the marker can be called and why not. Checks on the raw selected signal run in `gk_prepare()` before any correction or threshold; threshold estimation adds its own outcomes.

Status	Meaning
`NOT_CALLABLE_ABSENT`	constant or near-constant signal
`NOT_CALLABLE_UNAVAILABLE`	too many cells without usable signal
`NOT_CALLABLE_UNDERPOWERED`	too few cells, or too few in a pool
`NOT_CALLABLE_NO_POPULATION`	the method finds no positive population
`NOT_CALLABLE_NO_SEPARATION`	positive and negative are not separated
`PENDING_THRESHOLD`	supported; threshold not yet estimated
`NOT_GATED`	context marker with usable signal
`CALLABLE`	finite threshold, all checks passed
`MANUAL`	cut set by a reviewer or the configuration

A marker that is not callable never produces positive calls, and rules requiring it are disabled and listed.

Usage

```
gk_callability(x, ...)
```

Arguments

x	A gk_prepared object, or a later result carrying callability (thresholds, classification, review).
...	Unused.

Value

A data frame with image_id, marker, role, status, details and support statistics (n_cells, n_available, fraction_unavailable, sd, mad, fraction_mode, q01, q50, q99).

See Also

Other preparation: gk_prepare()

Examples

```
prep <- gk_prepare(gk_example_path("example-slide"), gk_example_config(), quiet = TRUE)
cb <- gk_callability(prep)
cb[cb$status != "PENDING_THRESHOLD", c("image_id", "marker", "status", "details")]
```

---

gk_cellspec *Build a cell table in the cellspec 1.0 format*

Description

[Experimental]

Assembles a cellspec object from a cell table and a measurement matrix. This is an interim constructor that follows the cellspec 1.0 specification; once cellspecR is released, its readers produce the same object from tool exports.

Usage

```
gk_cellspec(
  cells,
  measurements,
  dictionary = NULL,
  images = NULL,
  channels = NULL,
  provenance = NULL
)
```

Arguments

cells	Data frame with one row per cell and at least the columns cell_id, image_id, x and y (centroids in um). sample_id defaults to image_id. Identifiers are coerced to character.
measurements	Numeric matrix, cells x features, rows in the order of cells. Column names are feature IDs of the form "<compartment>:<marker>:<statistic>", e.g. "nucleus:FOXP3:mean".
dictionary	Optional feature dictionary (columns feature_id, kind, marker, compartment, statistic, unit, source_name). Derived from the column names of measurements when NULL.
images	Optional image table (image_id, sample_id, pixel_size). Derived from cells when NULL.
channels	Optional channel table (image_id, channel_index, channel_name, marker). Derived from the dictionary when NULL.
provenance	Optional list describing the origin of the data.

Value

An object of class cellspec: a list with spec_version, cells, measurements, dictionary, images, channels, provenance and adjacency (NULL).

See Also

Other cell tables: gk_write_cellspec()

Examples

```
cells <- data.frame(
  cell_id = c("c1", "c2", "c3"), image_id = "img1",
  x = c(10, 20, 30), y = c(5, 5, 5)
)
m <- cbind("cell:CD3e:mean" = c(1, 50, 3), "nucleus:FOXP3:mean" = c(0, 12, 1))
x <- gk_cellspec(cells, m)
x
```

---

gk_classify *Classify cells with the configured ordered hierarchy*

Description

[Experimental]

Evaluates flags and ordered hierarchy rules against per-image thresholds. This is the only phenotype evaluator in gatekeepR: the review application and batch code both call it. A rule that depends on a non-callable required marker is disabled and recorded; no missing marker is silently treated as positive. Structure-review rules receive their configured pending label until a later review decision is replayed.

Usage

```
gk_classify(prepared, thresholds, config, correction = NULL)
```

Arguments

prepared	A gk_prepare() result.
thresholds	A gk_thresholds() result.
config	The configuration used to prepare the data.
correction	NULL or a gk_correct() result.

Value

An object of class gk_classification with cells, calls, disabled_rules, the configuration and threshold hashes, and the source hashes needed for review and export.

See Also

Other classification: gk_apply_candidate_rules(), gk_fixed_signal_calls(), gk_state_calls(), gk_structures()

Examples

```
prep <- gk_prepare(gk_example_path("example-slide"), gk_example_config(), quiet = TRUE)
th <- gk_thresholds(prep, gk_example_config(), seed = 1L)
cl <- gk_classify(prep, th, gk_example_config())
summary(cl)
```

---

gk_config *Assemble a gating configuration*

Description

[Experimental]

A configuration is the single source of truth for gating a study: which measurement represents each marker, the marker panel, the named parent sets used by state markers, the phenotype hierarchy, per-image threshold overrides and state-based refinements. It is validated as a whole and hashed canonically (see gk_config_hash()); the hash travels with every result produced from it.

Usage

```
gk_config(
  signal_policy,
  panel,
  parent_sets = list(),
  hierarchy,
```

```
  overrides = NULL,
  refinements = NULL
)
```

Arguments

signal_policy	A gk_signal_policy() (or cellspecR signal policy) data frame.
panel	A gk_panel().
parent_sets	Named list of character vectors of cell-type labels, e.g. list("T cells" = c("CD4 T cells", "CD8 T cells")).
hierarchy	A gk_hierarchy().
overrides	NULL or a list of gk_override() objects.
refinements	NULL or a list of gk_refinement() objects.

Value

An object of class gk_config. Invalid combinations abort with a gatekeepr_error_config listing every failed check.

See Also

gk_validate_config() for a check table without aborting; vignette("configuration", package = "gatekeepR").

Other configuration: gk_config_hash(), gk_hierarchy(), gk_min_support(), gk_panel(), gk_signal_policy(), gk_validate_config(), gk_write_config(), hierarchy-elements

Examples

```
cfg <- gk_example_config()
cfg
gk_config_hash(cfg)
```

---

gk_config_hash *Canonical hash of a configuration*

Description

[Experimental]

SHA-256 of the canonical JSON form of a configuration: object keys sorted, no insignificant whitespace, numbers in shortest round-trip form. The hash is therefore identical for configurations that differ only in key order, whitespace or numeric spelling in their JSON files, and changes whenever a rule, threshold setting, flag policy, override or refinement changes.

Usage

```
gk_config_hash(config)
```

Arguments

config	A gk_config().

Value

A 64-character lowercase hexadecimal string.

See Also

Other configuration: gk_config(), gk_hierarchy(), gk_min_support(), gk_panel(), gk_signal_policy(), gk_validate_config(), gk_write_config(), hierarchy-elements

Examples

```
gk_config_hash(gk_example_config())
```

---

gk_correct *Correct prepared marker scores*

Description

[Experimental]

Applies a deterministic correction to transformed marker scores. Markers on images that fail the raw support checks are excluded from fitting and are NA in the corrected matrix. Missing measurements remain NA. Fitting uses all supported images together; residual methods use median imputation internally without exposing imputed values as measurements. The fitted parameters are returned in a gk_correction_model and can be applied again to the same prepared data. Stored correction models from older data-hash versions must be regenerated from their original source and configuration. For migration of the separate frozen residual parameter models, see gk_apply_residual_model().

Usage

```
gk_correct(
  prepared,
  method = c("none", "robust_z", "common_score_residual", "pc1_panel_residual"),
  markers = NULL,
  row_center = FALSE
)
```

Arguments

prepared	A gk_prepare() result.
method	One of "none", "robust_z", "common_score_residual" or "pc1_panel_residual".
markers	Character vector of markers to correct. NULL selects identity and conditional identity markers.
row_center	Whether to subtract each cell's residual mean. This changes the meaning of per-cell values and is FALSE by default.

Value

A gk_corrected list with matrix, model, config_sha256 and source_sha256. The model has class gk_correction_model and stores all fitted centres, scales, rotations and coefficients.

See Also

Other corrections: gk_fit_residual_model(), gk_import_residual_model()

Examples

```
prep <- gk_prepare(gk_example_path("example-slide"), gk_example_config(), quiet = TRUE)
corrected <- gk_correct(prep, method = "robust_z")
corrected$model
```

---

gk_decide_structure *Decide one structure or bounding box*

Description

Decide one structure or bounding box

Usage

```
gk_decide_structure(
  review,
  image_id,
  structure_id = NULL,
  bbox = NULL,
  decision = c("Tumor", "Benign", "Unresolved"),
  reviewer,
  reason
)
```

Arguments

review	A gk_review() object.
image_id	Image to change.
structure_id	A structure ID, or NULL.
bbox	A c(xmin, xmax, ymin, ymax) selection, or NULL.
decision	One of "Tumor", "Benign" or "Unresolved".
reviewer	Reviewer identifier.
reason	Required reason recorded in the ledger.

Value

A new gk_review object.

See Also

Other review: gk_dispose(), gk_import_cuts(), gk_ledgers(), gk_load_review(), gk_replace_review(), gk_replay(), gk_revert_last(), gk_review(), gk_save_review(), gk_set_cut(), gk_set_state()

---

gk_dispose *Set a cell disposition*

Description

Set a cell disposition

Usage

```
gk_dispose(
  review,
  image_id,
  cell_id = NULL,
  bbox = NULL,
  disposition = c("EXCLUDE", "DELETE", "RESTORE"),
  reviewer,
  reason
)
```

Arguments

review	A gk_review() object.
image_id	Image to change.
cell_id	A cell ID, or NULL when using bbox.
bbox	A c(xmin, xmax, ymin, ymax) selection, or NULL.
disposition	"EXCLUDE", "DELETE" or "RESTORE".
reviewer	Reviewer identifier.
reason	Required reason recorded in the ledger.

Value

A new gk_review object.

See Also

Other review: gk_decide_structure(), gk_import_cuts(), gk_ledgers(), gk_load_review(), gk_replace_review(), gk_replay(), gk_revert_last(), gk_review(), gk_save_review(), gk_set_cut(), gk_set_state()

gk_embed *Compute a reproducible marker embedding and density populations*

Description

[Experimental]

Computes a sign-canonicalised PCA, a deterministic two-dimensional UMAP, and density basins used by `population_crossing` thresholds. The default basin implementation is pure R. `small_basins = "uncertain"` retains cells while assigning their population as `NA`; `"merge"` assigns them to the nearest retained basin.

Usage

```
gk_embed(
  prepared,
  correction = NULL,
  n_pcs = 10L,
  n_neighbors = 20L,
  min_dist = 0.15,
  grid = 400L,
  bg = 0.03,
  tolerance = 0.03,
  basin_method = c("hillclimb", "ebimage"),
  small_basins = c("uncertain", "merge"),
  min_abs = 30L,
  min_frac = 0.002,
  seed = 42L
)
```

Arguments

`prepared`	A `gk_prepare()` result.
`correction`	`NULL` or a `gk_correct()` result.
`n_pcs`	Number of principal components to retain.
`n_neighbors`	UMAP neighbourhood size.
`min_dist`	UMAP minimum distance.
`grid`	Number of KDE grid points in each dimension.
`bg`	Relative density below which KDE cells are background.
`tolerance`	Basin merge or watershed tolerance.
`basin_method`	`"hillclimb"` or optional `"ebimage"`.
`small_basins`	Whether small basins become uncertain or merge.
`min_abs`	Minimum cells for a retained basin.
`min_frac`	Minimum fraction for a retained basin.
`seed`	Random seed used by UMAP.

Value

An object of class gk_embedding with pca, umap, population, KDE data, basin information, parameters and provenance hashes.

See Also

Other thresholds: gk_threshold_methods(), gk_thresholds()

Examples

```
if (requireNamespace("uwot", quietly = TRUE)) {
  prep <- gk_prepare(gk_simulate(n_cells = 300, seed = 7),
    gk_example_config(), quiet = TRUE)
  emb <- gk_embed(prep, n_pcs = 3L, grid = 40L, seed = 7L)
  emb
}
```

---

gk_example_config *Example configuration for a generic tumour-immune panel*

Description

[Experimental]

A study-neutral configuration used by examples, tests, vignettes and the app, matching the data produced by gk_simulate(). It declares twelve markers:

- identity: PanCK, CD45, CD3e, CD4, CD8, CD20, CD68, CD31;
- conditional identity: CD21 (optional; absent in the simulated slide);
- context: SMA (displayed only);
- state: FOXP3 (nuclear, within CD4 T cells and tumour cells) and Ki67 (nuclear, within tumour and immune cells), both with a nucleus/cytoplasm localisation rule.

The hierarchy contains a structure-review rule for PanCK-positive cells (they stay "Unreviewed epithelial cells" until a tumour/benign decision), a mixed-contact flag with policy unresolved, and two FOXP3 refinements. It is not a validated panel for any study.

Usage

```
gk_example_config()
```

Value

A gk_config() object.

See Also

Other example data: gk_example_path(), gk_simulate()

Examples

```
cfg <- gk_example_config()
cfg$panel
cfg$hierarchy
```

`gk_example_path` *Paths to bundled example data*

Description

[Experimental]

Returns the path of a file or directory shipped in `inst/extdata`:

- `"example-slide"`: a cellspec directory from `gk_simulate(seed = 1)`;
- `"example-config.json"`: `gk_example_config()` written as JSON.

Review checkpoints and exports are created by the reviewer rather than bundled as pre-reviewed decisions. Use `gk_review()` and `gk_save_review()` to create a checkpoint, or `gk_lock()` and `gk_export()` for an export.

Usage

```
gk_example_path(name = c("example-slide", "example-config.json"))
```

Arguments

`name`	Which example to locate.

Value

A single path.

See Also

Other example data: `gk_example_config()`, `gk_simulate()`

Examples

```
gk_example_path("example-config.json")
```

gk_export *Write a locked review export*

Description

Modified snapshots are rejected before writing. Staged files must pass content and replay verification before they replace the destination.

Usage

```
gk_export(snapshot, dir, overwrite = FALSE)
```

Arguments

snapshot	A gk_lock() result.
dir	Destination directory.
overwrite	Replace an existing export directory.

Value

The destination directory invisibly.

See Also

Other export: gk_lock(), gk_read_export(), gk_verify_export()

gk_fit_residual_model *Freeze and apply a fitted residual score space*

Description

[Experimental]

Fits two numerical spaces from already transformed marker scores: an identity common-score residual and a panel residual against the first identity principal component. Robust centring uses medians and MAD with constant 1.4826, with nonpositive MAD replaced by 1; standardized inputs are clipped to [-5, 5]. Identity residuals are row-centred, then centred and scaled per marker. Panel residuals subtract the row mean of identity residuals before final per-marker centring and scaling. The first principal component has a deterministic sign (largest absolute loading positive).

At least 20 training rows are required. Panel regressions require 20 finite observations per marker; insufficient optional markers remain unavailable. Final nonpositive or nonfinite standard deviations use 1. Missing identity inputs and rank-deficient regression fits fail; they are never imputed.

Application uses only stored parameters, so deleting, reordering or changing another row cannot refit the scores of an unchanged row. The exact ordered marker dictionary is required. Optional marker NA values remain NA. The model contains no fitted per-cell matrix and can be saved with

base::saveRDS(). Its content hash binds parameters, dimensions, names, provenance and bindings. Schema gk_frozen_residual_2.0.0 uses corrected round-trip numeric formatting for content hashes. Legacy schema 1 models must not be applied or relabelled as schema 2. Preserve their files and verify the original parameter records; gk_import_residual_model() can import those records with an authenticated source-file hash without refitting. Preserve prior provenance and bindings, then requalify the resulting score space before downstream use.

This is a numerical fit/apply contract, not raw-channel callability, gate estimation or biological validation. The caller owns raw measurement selection, any asinh cofactor, image-specific training scope and downstream decisions. Constant raw channels may retain historical residual artifacts; downstream callers must enforce raw support independently. Reusing a fitted model on a different image or cohort requires separate scientific qualification.

Usage

```
gk_fit_residual_model(
  scores,
  identity_markers,
  provenance = list(),
  bindings = list()
)

gk_apply_residual_model(scores, model)
```

Arguments

scores	Numeric matrix of already transformed scores with unique, ordered marker column names. Identity columns must be finite. Other columns may contain NA, but neither NaN nor infinity is accepted.
identity_markers	Unique ordered identity-marker names in scores.
provenance	Named list recording the caller's training source, transformation and fit scope. Stored without interpretation.
bindings	Named list of immutable configuration or threshold records to bind to this score space. No gate interpretation or transfer is performed.
model	A gk_frozen_residual_model from gk_fit_residual_model().

Value

Fit returns a gk_frozen_residual_model with schema version, parameters, source-score SHA-256, provenance, bindings and model SHA-256. Apply returns a list with identity and gate matrices and model_sha256.

See Also

Other corrections: gk_correct(), gk_import_residual_model()

Examples

```
scores <- cbind(A = seq_len(40), B = sin(seq_len(40)), C = cos(seq_len(40)))
model <- gk_fit_residual_model(scores, c("A", "B"),
  provenance = list(fit_scope = "synthetic input image", transform = "none"))
result <- gk_apply_residual_model(scores, model)
dim(result$gate)
```

---

gk_fixed_signal_calls *Apply fixed cutoffs with explicit raw-support availability*

Description

[Experimental]

Selected signals and underlying features are assessed over the full supplied image scope. A score can be HIGH or LOW only when enabled, finite, supported by at least `min_finite` nonnegative selected values with distinct values, and its selected source feature has at least `min_finite` finite values with distinct values. The named source value must also be finite at that cell; finite selected values cannot substitute for missing source measurements. All other calls remain UNAVAILABLE. No threshold is fitted.

Usage

```
gk_fixed_signal_calls(
  scores,
  selected,
  selected_features,
  raw_features,
  cutoffs,
  enabled,
  min_finite = 20L
)
```

Arguments

`scores`	Keyed numeric score matrix; its rows may be a subset of selected.
`selected`	Keyed numeric selected-signal matrix for the full image scope.
`selected_features`	Character matrix matching selected, naming each underlying raw feature. NA denotes an unavailable selected source.
`raw_features`	Keyed numeric raw-feature matrix with the same row order as selected. Source-feature support counts all finite values.
`cutoffs`	Named numeric vector in exact score-column order.
`enabled`	Named logical vector in exact score-column order.
`min_finite`	Minimum finite count, an integer of at least two.

Value

List with ternary `calls`, selected-marker and source-feature support.

See Also

Other classification: `gk_apply_candidate_rules()`, `gk_classify()`, `gk_state_calls()`, `gk_structures()`

Examples

```
x <- matrix(1:4, 4, 1, dimnames = list(letters[1:4], "A"))
f <- x
colnames(f) <- "raw_a"
sources <- matrix("raw_a", 4, 1, dimnames = dimnames(x))
gk_fixed_signal_calls(x, x, sources, f, c(A = 2), c(A = TRUE), 2)$calls
```

---

gk_fixed_state_diagnostics

Diagnose fixed state thresholds with explicit parent and localization scope

---

Description

[Experimental]

Each column is an independent state test. Parent eligibility is supplied by the caller and is not scientific authorization. Optional localization uses `(numerator + offset) / (denominator + offset) >= minimum`; both measurements and their offset-adjusted operands must be finite. Nonfinite ratios cannot be positive. `audit_missing` explicitly selects whether nonfinite or unavailable measurements produce FALSE or NA in the legacy audit flag. The separate nullable positive flag is NA whenever the measurement, threshold or parent is unavailable or ineligible. Missingness status takes precedence over threshold availability, then parent eligibility. No model, threshold or authorization is inferred.

Usage

```
gk_fixed_state_diagnostics(
  values,
  cutoffs,
  parent,
  localization_numerator = NULL,
  localization_denominator = NULL,
  localization_minimum = NULL,
  localization_offset = 1e-09,
  audit_missing
)
```

Arguments

values	Keyed numeric matrix of state measurements in threshold units.
cutoffs	Named numeric vector in exact state-column order.
parent	Keyed logical matrix matching values; NA means unknown eligibility.
localization_numerator, localization_denominator	Optional keyed numeric matrices matching values, in caller-specified common units.
localization_minimum	Named numeric vector matching columns. NA disables localization for that column; finite values enable it.
localization_offset	Finite nonnegative scalar added to both operands.
audit_missing	Named character vector of false or propagate per state.

Value

Long data frame with cell_id, state, audit_positive, measurement_evaluable, parent_eligible, threshold_available, diagnostic_status, nullable_positive, review_authorized and production_eligible. Authorization is always FALSE.

Examples

```
x <- matrix(c(1, NA), 2, 1, dimnames = list(c("a", "b"), "state"))
parent <- matrix(TRUE, 2, 1, dimnames = dimnames(x))
gk_fixed_state_diagnostics(x, c(state = 1), parent,
  audit_missing = c(state = "propagate"))
```

---

gk_fixed_threshold_calls

Apply fixed score thresholds without raw-support assessment

Description

[Experimental]

This numerical diagnostic does not establish raw measurement support or scientific eligibility. Disabled, missing or nonfinite scores and thresholds produce UNAVAILABLE. Equality to a finite cutoff produces HIGH.

Usage

```
gk_fixed_threshold_calls(scores, cutoffs, enabled)
```

Arguments

scores	Keyed numeric score matrix.
cutoffs	Named numeric vector in exact column order.
enabled	Named logical vector in exact column order, without missingness.

Value

Keyed character matrix containing HIGH, LOW or UNAVAILABLE.

Examples

```
x <- matrix(c(1, NA), 2, 1, dimnames = list(c("a", "b"), "signal"))
gk_fixed_threshold_calls(x, c(signal = 1), c(signal = TRUE))
```

---

gk_hierarchy *Declare a phenotype hierarchy*

Description

[Experimental]

An ordered set of gk_rule()s (first match wins), flags evaluated before the rules with an explicit policy each, and the labels used for cells that match no rule, that are excluded by a flag, or that are unresolved.

Flag policies:

- `exclude`: the cell is labelled `excluded_label` and not analysed.
- `unresolved`: the cell is labelled `unresolved_label`.
- `flag_only`: the flag is recorded and the cell continues through the rules.

The chosen policy is part of the configuration hash and applies identically in batch code and in the review app.

Usage

```
gk_hierarchy(
  rules,
  flags = list(),
  flag_policy = list(),
  undefined_label = "Undefined",
  excluded_label = "Excluded",
  unresolved_label = "Unresolved",
  version,
  rationale = ""
)
```

Arguments

`rules`	List of gk_rule() objects, in evaluation order.
`flags`	Named list of gk_flag() objects.
`flag_policy`	Named list or character vector giving the policy of every flag.
`undefined_label, excluded_label, unresolved_label`	Labels for cells matching no rule, excluded by a flag, or unresolved by a flag.
`version`	Version string of the hierarchy (required).
`rationale`	Free text recorded with the hierarchy.

Value

An object of class gk_hierarchy.

See Also

Other configuration: gk_config(), gk_config_hash(), gk_min_support(), gk_panel(), gk_signal_policy(), gk_validate_config(), gk_write_config(), hierarchy-elements

Examples

```
gk_hierarchy(
  rules = list(
    gk_rule("R010", "T cells", all_of = c("CD45", "CD3e")),
    gk_rule("R020", "Other immune cells", all_of = "CD45")
  ),
  flags = list(mixed = gk_flag(all_of = "PanCK", any_of = "CD45")),
  flag_policy = c(mixed = "unresolved"),
  version = "1.0"
)
```

---

gk_import_cuts *Import identity-marker cuts without replacing other ledgers*

Description

Import identity-marker cuts without replacing other ledgers

Usage

```
gk_import_cuts(review, path, reviewer, reason)
```

Arguments

review	A gk_review() object.
path	A TSV file with image_id, marker, estimate and optional parent columns.
reviewer	Reviewer identifier.
reason	Required reason recorded for every imported cut.

Value

A new gk_review object.

See Also

Other review: gk_decide_structure(), gk_dispose(), gk_ledgers(), gk_load_review(), gk_replace_review(), gk_replay(), gk_revert_last(), gk_review(), gk_save_review(), gk_set_cut(), gk_set_state()

gk_import_residual_model *Import externally frozen residual parameters without fitting*

Description

[Experimental]

Validates a numerical parameter bundle and authenticates its external source file. The caller maps the external format into the documented parameter fields; the source hash authenticates bytes, not mapping correctness or scientific approval. No coefficient is estimated or changed. Import also supports an explicitly verified legacy parameter record. Keep its original file, provenance and bindings; import creates a schema 2 model without modifying that file or refitting any parameter. Existing provenance fields are preserved, and `import_source_sha256` on the model authenticates the file supplied to this import. Downstream use requires requalification.

Usage

```
gk_import_residual_model(
  parameters,
  source_file,
  source_sha256,
  provenance = list(),
  bindings = list()
)
```

Arguments

`parameters`	Named list containing `panel`, `identity_markers`, `identity_robust`, `identity_regression`, `identity_center`, `identity_scale`, `panel_robust`, `pc_center`, `pc1_rotation`, `panel_regression`, `panel_center`, `panel_scale` and `n_training`, with the same shapes as a fitted model.
`source_file`	Existing file from which the parameters were obtained.
`source_sha256`	Expected lowercase SHA-256 of that file.
`provenance`	Named list of caller source and transformation metadata.
`bindings`	Named list of immutable caller policies or thresholds.

Value

A hash-bound `gk_frozen_residual_model`. The training-score hash is unavailable; the authenticated source-file hash is stored in `import_source_sha256` and, when absent, provenance's `external_source_sha256` field.

See Also

Other corrections: `gk_correct()`, `gk_fit_residual_model()`

Examples

```
x <- cbind(A = seq_len(30), B = sin(seq_len(30)))
fitted <- gk_fit_residual_model(x, c("A", "B"))
fields <- c("panel", "identity_markers", "identity_robust",
  "identity_regression", "identity_center", "identity_scale", "panel_robust",
  "pc_center", "pc1_rotation", "panel_regression", "panel_center",
  "panel_scale", "n_training")
path <- tempfile(fileext = ".rds")
saveRDS(fitted, path)
imported <- gk_import_residual_model(unclass(fitted)[fields], path,
  digest::digest(file = path, algo = "sha256"))
gk_apply_residual_model(x, imported)$identity
unlink(path)
```

---

gk_ledgers *Return the four review ledgers*

Description

Return the four review ledgers

Usage

```
gk_ledgers(review)
```

Arguments

review	A gk_review() object.

Value

A named list of append-only data frames.

See Also

Other review: gk_decide_structure(), gk_dispose(), gk_import_cuts(), gk_load_review(), gk_replace_review(), gk_replay(), gk_revert_last(), gk_review(), gk_save_review(), gk_set_cut(), gk_set_state()

gk_load_review *Load a review checkpoint and acquire its single-writer lock*

Description

[Experimental]

Usage

```
gk_load_review(dir, takeover = FALSE)
```

Arguments

dir	Directory written by gk_save_review().
takeover	Replace a lock left by a stale session.

Details

Loading is serialized with saving by the sibling operation lock described in gk_save_review(). Explicit takeover applies to session ownership; it cannot interrupt a save or load currently holding the operating-system lock.

Value

A gk_review object.

See Also

Other review: gk_decide_structure(), gk_dispose(), gk_import_cuts(), gk_ledgers(), gk_replace_review(), gk_replay(), gk_revert_last(), gk_review(), gk_save_review(), gk_set_cut(), gk_set_state()

gk_lock *Lock a review into an immutable snapshot*

Description

The version 2 review ID binds source identity, typed replay inputs, effective outputs, ledgers, correction model, reviewer and status. It is a content hash, not a signature or proof of a reviewer's identity. REVIEWED records the human reviewer's assertion; it does not certify scientific or clinical validity.

Usage

```
gk_lock(review, reviewer, status = c("PENDING", "REVIEWED"))
```

Arguments

review	A gk_review() object.
reviewer	Reviewer identifier.
status	"PENDING" or "REVIEWED".

Value

An object of class gk_snapshot.

See Also

Other export: gk_export(), gk_read_export(), gk_verify_export()

---

gk_min_support *Callability rules for a marker*

Description

[Experimental]

Sets the checks that decide whether a marker can be called on an image before and after threshold estimation (see gk_callability()).

Usage

```
gk_min_support(
  min_cells = 20L,
  max_unavailable = 0.2,
  min_auc = 0.65,
  near_constant_mode_fraction = 0.99
)
```

Arguments

min_cells	Minimum number of available cells, and minimum cells per positive and negative pool.
max_unavailable	Largest fraction of cells whose signal is unavailable (missing or below the policy's min_value) before the marker is NOT_CALLABLE_UNAVAILABLE.
min_auc	Minimum AUC between positive and negative pools for the crossing methods. The AUC describes internally selected pools and is not a validated accuracy.
near_constant_mode_fraction	A marker whose median absolute deviation is zero and whose most frequent value covers at least this fraction of available cells is NOT_CALLABLE_ABSENT.

Value

A list of class `gk_min_support`.

See Also

Other configuration: `gk_config()`, `gk_config_hash()`, `gk_hierarchy()`, `gk_panel()`, `gk_signal_policy()`, `gk_validate_config()`, `gk_write_config()`, `hierarchy-elements`

Examples

```
gk_min_support(min_auc = 0.6)
```

---

gk_panel *Declare a marker panel*

Description

[Experimental]

A panel lists the markers used for gating, their role and how their signal is transformed and thresholded. Arguments are vectorised over `marker`; scalar arguments are recycled.

Roles:

- `identity`: used in hierarchy rules.
- `conditional_identity`: used in rules only on images where it is callable; when it is not callable, rules requiring it are disabled and it is treated as negative where rules exclude it.
- `state`: called within parent cell types (see `gk_state_calls()`).
- `context`: measured and displayed, never used in rules.

Usage

```
gk_panel(
  marker,
  role,
  transform = "asinh",
  cofactor = 5,
  threshold_method = "mixture",
  threshold_args = list(),
  parents = list(),
  localization = NULL,
  optional = FALSE,
  min_support = gk_min_support()
)
```

Arguments

marker	Character vector of unique marker names; each must appear in the signal policy of the configuration.
role	Role per marker (see above).
transform	"asinh" (asinh(x / cofactor)), "log1p" or "none".
cofactor	Positive cofactor for asinh.
threshold_method	Method per marker, see gk_threshold_methods().
threshold_args	Method arguments: either one list used for every marker, or a list named by marker whose elements are lists.
parents	State markers only: the parent sets (names from parent_sets in gk_config()) within which the marker is estimated and called. Either a character vector used for every state marker, or a list named by marker.
localization	NULL, or a list named by marker whose elements are lists with numerator and denominator compartments, min_ratio and optionally statistic (default "mean"), e.g. list(FOXP3 = list(numerator = "nucleus", denominator = "cytoplasm", min_ratio = 1.5)).
optional	Logical per marker. A non-callable optional marker only disables its own rules; a non-callable required marker additionally raises a gatekeepr_warning_uncallable warning. Neither blocks an image.
min_support	A gk_min_support() object used for every marker, or a list of them named by marker.

Value

An object of class gk_panel: a list with markers (a data frame with marker, role, transform, cofactor, threshold_method, optional) and per-marker named lists threshold_args, parents, localization and min_support.

See Also

Other configuration: gk_config(), gk_config_hash(), gk_hierarchy(), gk_min_support(), gk_signal_policy(), gk_validate_config(), gk_write_config(), hierarchy-elements

Examples

```
gk_panel(
  marker = c("CD45", "CD3e", "FOXP3"),
  role = c("identity", "identity", "state"),
  threshold_method = c("mixture", "mixture", "parent_crossing"),
  parents = list(FOXP3 = "T cells"),
  localization = list(
    FOXP3 = list(numerator = "nucleus", denominator = "cytoplasm", min_ratio = 1.5)
  )
)
```

gk_plot_density *Plot a threshold density and its evidence pools*

Description

Plot a threshold density and its evidence pools

Usage

```
gk_plot_density(thresholds, marker, image_id, parent = NA_character_)
```

Arguments

thresholds	A gk_thresholds() result.
marker	Marker to plot.
image_id	Image to plot.
parent	Parent set for a state threshold; NA for identity markers.

Value

A ggplot object.

See Also

Other plots: gk_plot_dotplot(), gk_plot_embedding(), gk_plot_map(), gk_plot_overview(), gk_plot_states(), gk_plot_thresholds()

gk_plot_dotplot *Plot marker means by classified cell type*

Description

One point represents the mean of observed finite scores for one image, marker and cell type. Point size is the number of contributing cells. Scores use the same correction as the classification. Objects with mismatched source data, row order or score space are rejected.

Usage

```
gk_plot_dotplot(prepared, classification, thresholds)
```

Arguments

prepared	A gk_prepare() result.
classification	A gk_classify() result.
thresholds	A gk_thresholds() result.

Value

A ggplot object.

See Also

Other plots: gk_plot_density(), gk_plot_embedding(), gk_plot_map(), gk_plot_overview(), gk_plot_states(), gk_plot_thresholds()

`gk_plot_embedding` *Plot an embedding coloured by population or classification*

Description

Plot an embedding coloured by population or classification

Usage

```
gk_plot_embedding(embedding, classification = NULL, colour_by = "population")
```

Arguments

`embedding`	A gk_embed() result.
`classification`	Optional gk_classify() result.
`colour_by`	Column in classification cells, or population.

Value

A ggplot object.

See Also

Other plots: gk_plot_density(), gk_plot_dotplot(), gk_plot_map(), gk_plot_overview(), gk_plot_states(), gk_plot_thresholds()

gk_plot_map *Plot cell centroids on an image map*

Description

Plot cell centroids on an image map

Usage

```
gk_plot_map(x, image_id, colour_by = "cell_type", layers = NULL)
```

Arguments

x	A gk_prepare(), gk_classify() or gk_replayed() result.
image_id	Image to plot.
colour_by	Cell column used for colour.
layers	Reserved layer configuration.

Value

A ggplot object.

See Also

Other plots: gk_plot_density(), gk_plot_dotplot(), gk_plot_embedding(), gk_plot_overview(), gk_plot_states(), gk_plot_thresholds()

gk_plot_overview *Make a compact overview*

Description

Make a compact overview

Usage

```
gk_plot_overview(x, image_id)
```

Arguments

x	A prepared, classified or replayed object.
image_id	Image to plot.

Value

A patchwork object when patchwork is installed, otherwise the map.

See Also

Other plots: gk_plot_density(), gk_plot_dotplot(), gk_plot_embedding(), gk_plot_map(), gk_plot_states(), gk_plot_thresholds()

`gk_plot_states` *Plot state calls for one image*

Description

Plot state calls for one image

Usage

```
gk_plot_states(state_calls, image_id)
```

Arguments

`state_calls`	A gk_state_calls() result.
`image_id`	Image to plot.

Value

A ggplot object.

See Also

Other plots: gk_plot_density(), gk_plot_dotplot(), gk_plot_embedding(), gk_plot_map(), gk_plot_overview(), gk_plot_thresholds()

`gk_plot_thresholds` *Plot thresholds for one image*

Description

Plot thresholds for one image

Usage

```
gk_plot_thresholds(thresholds, image_id)
```

Arguments

`thresholds`	A gk_thresholds() result.
`image_id`	Image to plot.

Value

A ggplot object.

See Also

Other plots: gk_plot_density(), gk_plot_dotplot(), gk_plot_embedding(), gk_plot_map(), gk_plot_overview(), gk_plot_states()

---

gk_prepare *Prepare marker signals for gating*

Description

[Experimental]

Selects one signal per cell and marker according to the configuration's signal policy, applies the per-marker transform, resolves localisation measurements, and computes callability checks that run before any correction or threshold (see gk_callability()).

A panel marker whose measurement is missing from the cell table is a structural failure (gatekeepr_error_structure). A marker that is present but carries no usable signal is not an error: it is reported as not callable, and every rule depending on it is disabled later.

Usage

```
gk_prepare(x, config, image_id = NULL, quiet = FALSE)
```

Arguments

x	A cellspec object (see gk_cellspec()) or the path of a cellspec directory.
config	A gk_config().
image_id	NULL for all images, or the images to keep.
quiet	Suppress the summary message?

Value

An object of class gk_prepared, a list with:

- cells: data frame image_id, cell_id, sample_id, x, y;
- signal_raw: cells x panel markers, selected raw signal (NA where unavailable);
- score: transformed signal;
- source: integer matrix, 1 = preferred compartment, 2 = fallback, 0 = unavailable; source_labels describes the features;
- localization: named list of per-marker data frames with numerator, denominator, ratio and evaluable;
- callability: see gk_callability();
- transforms: marker, transform and cofactor;
- config, config_sha256, source_sha256 and source_manifest_sha256.

See Also

Other preparation: gk_callability()

Examples

```
x <- gk_read_cellspec(gk_example_path("example-slide"))
prep <- gk_prepare(x, gk_example_config())
prep
gk_callability(prep)
```

---

gk_read_export *Read a verified export*

Description

Read a verified export

Usage

```
gk_read_export(dir, verify = TRUE, expected_review_id = NULL)
```

Arguments

dir	Export directory.
verify	Verify before reading? FALSE permits inspection of legacy or untrusted exports and makes no integrity claim.
expected_review_id	Optional full ID passed to gk_verify_export().

Value

A gk_export list containing snapshot, configuration and tables.

See Also

Other export: gk_export(), gk_lock(), gk_verify_export()

gk_replace_review *Replace selected review ledgers from a saved checkpoint*

Description

Replace selected review ledgers from a saved checkpoint

Usage

```
gk_replace_review(
  review,
  snapshot_dir,
  scope = c("cuts", "structures", "dispositions", "states", "all"),
  confirm = FALSE,
  backup_dir = NULL
)
```

Arguments

review	A gk_review() object.
snapshot_dir	A directory written by gk_save_review().
scope	Ledgers to replace.
confirm	Must be TRUE, because history is being replaced.
backup_dir	Directory in which to save the current review first, or NULL for a temporary directory.

Details

Replacement requires the same analytical base. Partial replacement must preserve globally unique action orders and valid undo references. If histories conflict, use gk_import_cuts() to append new cuts, or replace the complete history with scope = "all".

Value

A new gk_review object with backup_path attribute.

See Also

Other review: gk_decide_structure(), gk_dispose(), gk_import_cuts(), gk_ledgers(), gk_load_review(), gk_replay(), gk_revert_last(), gk_review(), gk_save_review(), gk_set_cut(), gk_set_state()

gk_replay *Replay all effective review decisions*

Description

Replay all effective review decisions

Usage

```
gk_replay(review)
```

Arguments

review	A gk_review() object.

Value

A gk_replayed object with effective cells, thresholds and state calls.

See Also

Other review: gk_decide_structure(), gk_dispose(), gk_import_cuts(), gk_ledgers(), gk_load_review(), gk_replace_review(), gk_revert_last(), gk_review(), gk_save_review(), gk_set_cut(), gk_set_state()

gk_revert_last *Undo the most recent review action without deleting history*

Description

Undo the most recent review action without deleting history

Usage

```
gk_revert_last(review, reviewer, reason)
```

Arguments

review	A gk_review() object.
reviewer	Reviewer identifier.
reason	Required reason recorded in the ledger.

Details

The appended row references the target action. Replay restores the state before that action, including earlier overlapping selections. Undoing an undo reactivates its target. No existing ledger row is removed.

Value

A new gk_review object.

See Also

Other review: gk_decide_structure(), gk_dispose(), gk_import_cuts(), gk_ledgers(), gk_load_review(), gk_replace_review(), gk_replay(), gk_review(), gk_save_review(), gk_set_cut(), gk_set_state()

---

gk_review *Create an editable review state*

Description

[Experimental]

Creates an immutable base review and four empty append-only ledgers. Every subsequent review function returns a new object; effective labels and dispositions are calculated by gk_replay().

Usage

```
gk_review(classification, state_calls, structures, thresholds, prepared)
```

Arguments

classification	A gk_classify() result.
state_calls	A gk_state_calls() result.
structures	A gk_structures() result.
thresholds	A gk_thresholds() result.
prepared	The matching gk_prepare() result.

Value

An object of class gk_review.

See Also

Other review: gk_decide_structure(), gk_dispose(), gk_import_cuts(), gk_ledgers(), gk_load_review(), gk_replace_review(), gk_replay(), gk_revert_last(), gk_save_review(), gk_set_cut(), gk_set_state()

gk_save_review *Save a review checkpoint atomically*

Description

[Experimental]

Usage

```
gk_save_review(review, dir, overwrite = FALSE)
```

Arguments

review	A gk_review() object.
dir	Directory to create.
overwrite	Replace an existing directory.

Details

Saving over a checkpoint requires ownership of any existing session lock. Both save and load use an operating-system lock on a separate hidden sibling file named .DIRECTORY.gatekeepR.lock. This empty file remains after use; do not remove it while sessions may be accessing the review. The filesystem must support advisory file locks, and its parent directory must be writable. Session ownership is preserved when replacing the checkpoint. Reviews returned by gk_load_review() carry a checkpoint revision. Saving that review back to its working directory rejects intervening writes, even from another browser session in the same R process. After saving, reload the checkpoint before further editing, or copy the returned path's checkpoint_revision attribute to review$checkpoint_revision. Explicit overwrites with reviews that were never loaded remain supported.

Value

The saved directory invisibly, with a checkpoint_revision attribute.

See Also

Other review: gk_decide_structure(), gk_dispose(), gk_import_cuts(), gk_ledgers(), gk_load_review(), gk_replace_review(), gk_replay(), gk_revert_last(), gk_review(), gk_set_cut(), gk_set_state()

gk_set_cut *Set an identity or state threshold in the review ledger*

Description

Set an identity or state threshold in the review ledger

Usage

```
gk_set_cut(
  review,
  image_id,
  marker,
  parent = NA_character_,
  estimate,
  reviewer,
  reason
)
```

Arguments

review	A gk_review() object.
image_id	Image to change.
marker	Marker to change.
parent	Parent set for a state marker; NA for identity markers.
estimate	New finite threshold.
reviewer	Reviewer identifier.
reason	Required reason recorded in the ledger.

Value

A new gk_review object.

See Also

Other review: gk_decide_structure(), gk_dispose(), gk_import_cuts(), gk_ledgers(), gk_load_review(), gk_replace_review(), gk_replay(), gk_revert_last(), gk_review(), gk_save_review(), gk_set_state()

gk_set_state *Enable, disable or set a parent-specific state call*

Description

Enable, disable or set a parent-specific state call

Usage

```
gk_set_state(
  review,
  image_id,
  marker,
  parent,
  action = c("ENABLE", "DISABLE", "SET_CUT"),
  estimate = NULL,
  reviewer,
  reason
)
```

Arguments

review	A gk_review() object.
image_id	Image to change.
marker	State marker.
parent	Configured parent set.
action	"ENABLE", "DISABLE" or "SET_CUT".
estimate	New threshold for SET_CUT.
reviewer	Reviewer identifier.
reason	Required reason recorded in the ledger.

Value

A new gk_review object.

See Also

Other review: gk_decide_structure(), gk_dispose(), gk_import_cuts(), gk_ledgers(), gk_load_review(), gk_replace_review(), gk_replay(), gk_revert_last(), gk_review(), gk_save_review(), gk_set_cut()

gk_signal_policy *Declare which measurement represents each marker*

Description

[Experimental]

A signal policy chooses, per marker, the compartment and statistic whose value represents the marker, with an optional fallback compartment used per cell when the preferred value is missing or below `min_value`. The result has the same structure as `cellspecR::cs_signal_policy()`, so either can be used in `gk_config()`.

Usage

```
gk_signal_policy(
  marker,
  compartment = "cell",
  statistic = "mean",
  fallback_compartment = NA_character_,
  min_value = 0
)
```

Arguments

`marker`	Character vector of unique marker names.
`compartment`	Preferred compartment per marker: `"cell"`, `"nucleus"`, `"cytoplasm"` or `"membrane"`. Recycled.
`statistic`	Statistic per marker, e.g. `"mean"`. Recycled.
`fallback_compartment`	Compartment used when the preferred value is unavailable, or `NA` for none. Recycled.
`min_value`	Values below this are treated as unavailable. Recycled.

Value

A data frame of class `gk_signal_policy` (also `cs_signal_policy`) with columns `marker`, `compartment`, `statistic`, `fallback_compartment` and `min_value`.

See Also

Other configuration: `gk_config()`, `gk_config_hash()`, `gk_hierarchy()`, `gk_min_support()`, `gk_panel()`, `gk_validate_config()`, `gk_write_config()`, `hierarchy-elements`

Examples

```
gk_signal_policy(
  marker = c("CD3e", "FOXP3", "PanCK"),
  compartment = c("cell", "nucleus", "cytoplasm"),
  fallback_compartment = c(NA, NA, "cell")
)
```

`gk_simulate` *Simulate a multiplexed tissue slide*

Description

[Experimental]

Generates a synthetic slide matching `gk_example_config()`, with known ground truth, for examples, tests and method checks. The slide contains:

- lineage populations (epithelial nests, CD8 and CD4 T cells including FOXP3-positive regulatory cells, B cells, macrophages, other immune cells, endothelial and stromal cells);
- spatial epithelial nests (tumour-like and small benign-like ducts) and scattered single epithelial cells;
- epithelial-immune contact objects co-positive for PanCK and CD45;
- a rare CD4/CD8 double-positive T-cell population;
- nuclear state markers (FOXP3, Ki67) with nuclear and cytoplasmic measurements, including cytoplasmic FOXP3 artefacts that fail the localisation rule;
- a shared per-cell brightness component;
- an absent channel (all zeros) and a shuffled marker.

Intensities are raw arbitrary units rounded to two decimals, as tool exports typically are. Ground truth is stored in the extra cell columns `sim_population` and `sim_structure`.

Usage

```
gk_simulate(
  n_cells = 8000L,
  markers = NULL,
  absent = "CD21",
  shuffled = "SMA",
  rare_fraction = 0.003,
  separation = 1,
  image_id = "sim-01",
  seed = 1L
)
```

Arguments

n_cells	Number of cells.
markers	NULL for all twelve example markers, or a subset.
absent	Markers whose channel is entirely zero.
shuffled	Markers whose values are permuted across cells, destroying any relation to cell identity.
rare_fraction	Fraction of cells in the rare double-positive population.
separation	Multiplier of the positive-population signal on the log scale; values below 1 make gating harder.
image_id	Image identifier.
seed	Random seed. The caller's random number state is restored.

Value

A cellspec object (see gk_cellspec()).

See Also

Other example data: gk_example_config(), gk_example_path()

Examples

```
x <- gk_simulate(n_cells = 2000, seed = 3)
x
table(x$cells$sim_population)
```

---

gk_state_calls *Call state markers within configured parent sets*

Description

[Experimental]

Estimates a separate state-marker threshold for every image and parent set, then returns a long table containing every cell by marker by parent combination. Localisation requirements are evaluated on the raw selected measurements. State calls are disabled for analysis by default and become effective only after a review ledger enables them. parent_crossing compares the parent with its reference; valley, tail and mixture estimate within the parent (or pooled parent union). Raw support is required even for manual cuts. Estimated positive and negative pools must each meet the marker's min_cells requirement.

Usage

```
gk_state_calls(prepared, thresholds, classification, config)
```

Arguments

prepared	A gk_prepare() result.
thresholds	A gk_thresholds() result for identity markers.
classification	A gk_classify() result.
config	The configuration used to prepare the data.

Value

An object of class gk_state_calls with long calls, per-parent state_thresholds, and provenance hashes.

See Also

Other classification: gk_apply_candidate_rules(), gk_classify(), gk_fixed_signal_calls(), gk_structures()

Examples

```
prep <- gk_prepare(gk_example_path("example-slide"), gk_example_config(), quiet = TRUE)
th <- gk_thresholds(prep, gk_example_config(), seed = 1L)
cl <- gk_classify(prep, th, gk_example_config())
states <- gk_state_calls(prep, th, cl, gk_example_config())
head(states$calls)
```

---

gk_structures *Find epithelial structures for review*

Description

[Experimental]

Runs deterministic centroid-based DBSCAN on cells awaiting a configured structure decision. Epsilon is the 95th percentile of the (min_pts - 1)th nearest-neighbour distance, bounded by eps_min and eps_max. Retained structures are ordered by decreasing size and then centroid coordinates, making IDs stable under input row permutation. Cells enter DBSCAN in coordinate and composite image/cell-key order so ambiguous border points receive the same assignment when input rows are reordered.

Usage

```
gk_structures(
  classification,
  prepared,
  eps_min = 15,
  eps_max = 30,
  min_pts = 10L,
  min_cells = 20L
)
```

Arguments

classification	A gk_classify() result.
prepared	The matching gk_prepare() result.
eps_min, eps_max	Lower and upper bounds for DBSCAN epsilon in the coordinate units of the cellspec.
min_pts	DBSCAN minimum points.
min_cells	Minimum cells for an exported structure.

Value

An object of class gk_structures with structures, cell_structure, parameters and provenance hashes.

See Also

Other classification: gk_apply_candidate_rules(), gk_classify(), gk_fixed_signal_calls(), gk_state_calls()

Examples

```
prep <- gk_prepare(gk_example_path("example-slide"), gk_example_config(), quiet = TRUE)
th <- gk_thresholds(prep, gk_example_config(), seed = 1L)
cl <- gk_classify(prep, th, gk_example_config())
if (requireNamespace("dbscan", quietly = TRUE)) gk_structures(cl, prep)
```

---

gk_threshold_methods *Threshold methods available in gatekeepR*

Description

[Experimental]

Lists the threshold estimation methods, the marker roles allowed to use them, their arguments with defaults, and a description. Set a marker's method with threshold_method and its arguments with threshold_args in gk_panel().

State markers also accept pooled (default FALSE): with pooled = TRUE one estimate is made over the union of the marker's parent sets and used for every parent, and the threshold table records this. The valley, tail and mixture methods use the scores within that parent or union; only parent_crossing compares it with an external reference. Estimated positive and negative pools must each satisfy min_cells from gk_min_support(). Manual cuts still require the raw support checks.

Usage

```
gk_threshold_methods()
```

Value

A data frame with columns `method`, `role_allowed` (roles joined by `", "`), `arguments` (`name = default` pairs) and `description`.

See Also

Other thresholds: `gk_embed()`, `gk_thresholds()`

Examples

```
gk_threshold_methods()[, c("method", "role_allowed")]
```

`gk_thresholds` *Estimate per-image marker thresholds*

Description

[Experimental]

Estimates thresholds for identity and conditional identity markers. Support is checked before estimation, and every method records a status rather than inventing a cut when its input is absent, underpowered or not separated. State-marker thresholds are estimated by `gk_state_calls()` after parent labels are available.

Usage

```
gk_thresholds(
  prepared,
  config,
  embedding = NULL,
  correction = NULL,
  seed = 5420L
)
```

Arguments

`prepared`	A `gk_prepare()` result.
`config`	The configuration used to prepare the data.
`embedding`	An optional `gk_embed()` result for `population_crossing`.
`correction`	`NULL` or a `gk_correct()` result.
`seed`	Seed used when sampling embedding populations.

Value

An object of class `gk_thresholds`, a data frame with one row per image and marker, the threshold result columns, and an `evidence` attribute containing density and pool data for plotting.

See Also

Other thresholds: gk_embed(), gk_threshold_methods()

Examples

```
prep <- gk_prepare(gk_example_path("example-slide"), gk_example_config(), quiet = TRUE)
thresholds <- gk_thresholds(prep, gk_example_config(), seed = 1L)
thresholds[, c("marker", "status", "estimate")]
```

---

gk_validate_config *Validate a gating configuration*

Description

[Experimental]

Runs every configuration check and returns them as a table instead of aborting. Accepts a gk_config() object, a path to a configuration JSON file, or a list as parsed from such a file. Construction errors (for example a duplicated rule_id) are reported as a failed construction check.

Usage

```
gk_validate_config(config)
```

Arguments

config	A gk_config, a path to a JSON file, or a parsed list.

Value

A data frame of class gk_config_checks with columns check, status ("pass" or "fail") and message.

See Also

Other configuration: gk_config(), gk_config_hash(), gk_hierarchy(), gk_min_support(), gk_panel(), gk_signal_policy(), gk_write_config(), hierarchy-elements

Examples

```
gk_validate_config(gk_example_config())
```

`gk_verify_export` *Verify content identity, replay and reconciliation invariants*

Description

Version 2 checks typed contents, source identity, correction model and every ledger, then reconstructs the base review and compares effective cells, scores and state calls. Legacy exports can be inspected explicitly with `gk_read_export(verify = FALSE)`, but require a new review and lock to obtain this verification contract. No export files are modified.

Usage

```
gk_verify_export(
  dir,
  cellspec_dir = NULL,
  strict = TRUE,
  expected_review_id = NULL
)
```

Arguments

`dir`	Export directory written by `gk_export()`.
`cellspec_dir`	Optional source cellspec directory, checked against the source and prepared score contents.
`strict`	Abort on failed verification?
`expected_review_id`	Optional independently retained full SHA-256 review ID (the snapshot's `review_id_full`).

Details

Hashes are not signatures. Keep the full review ID independently and supply `expected_review_id` when verification must detect a replaced whole export.

Value

A check data frame with `check`, `ok` and `details`.

See Also

Other export: `gk_export()`, `gk_lock()`, `gk_read_export()`

gk_write_cellspec *Read and write cellspec directories*

Description

[Experimental]

`gk_write_cellspec()` writes a `cellspec` object to a selected directory layout. The legacy layout uses `cells.tsv.gz` (cell columns then measurement columns), `cellspec.json` (spec version, images, channels, dictionary, provenance), `MANIFEST.sha256` and `DONE`. The directory is staged and renamed atomically. `gk_read_cellspec()` reads it back and, with `verify = TRUE`, refuses a directory whose manifest does not match.

The default `format = "legacy"` retains the original gatekeepR layout for compatibility. It is not the canonical cellspecR disk schema and is planned for retirement after migration is qualified. It does not serialize adjacency or typed sidecar metadata. Select `"parquet"` or `"tsv.gz"` explicitly for canonical storage owned by cellspecR. These formats require cellspecR and a valid canonical object, including declared positive pixel calibration; missing calibration is never guessed. Canonical storage preserves adjacency, typed metadata and additional columns.

Reading dispatches by the declared sidecar schema. Malformed or unsupported canonical data never fall back to the legacy reader. Canonical validation failures retain their `cellspec_error` condition classes.

Usage

```
gk_write_cellspec(
  x,
  dir,
  overwrite = FALSE,
  format = c("legacy", "parquet", "tsv.gz")
)

gk_read_cellspec(dir, verify = TRUE)
```

Arguments

`x`	A `cellspec` object.
`dir`	Directory path.
`overwrite`	Replace an existing directory?
`format`	Output layout: `"legacy"` for compatibility, or canonical `"parquet"` or `"tsv.gz"` through cellspecR.
`verify`	Check `DONE` and every manifest hash before reading?

Value

`gk_write_cellspec()` returns `x` invisibly. `gk_read_cellspec()` returns a `cellspec` object whose `provenance$manifest_sha256` holds the SHA-256 of `MANIFEST.sha256`.

See Also

Other cell tables: gk_cellspec()

Examples

```
x <- gk_read_cellspec(gk_example_path("example-slide"))
x
out <- file.path(tempdir(), "copy-of-example")
gk_write_cellspec(x, out, overwrite = TRUE)
unlink(out, recursive = TRUE)
```

---

gk_write_config *Read and write configuration files*

Description

[Experimental]

gk_write_config() writes a configuration as pretty-printed canonical JSON (sorted keys) following inst/schema/gatekeepr-config-1.0.0.schema.json. gk_read_config() parses a file with jsonlite (no code is evaluated), rebuilds the configuration through the constructors and aborts with a gatekeepr_error_config listing every failed check.

Usage

```
gk_write_config(config, path)

gk_read_config(path)
```

Arguments

config	A gk_config().
path	File path.

Value

gk_write_config() returns config invisibly; gk_read_config() returns a gk_config.

See Also

Other configuration: gk_config(), gk_config_hash(), gk_hierarchy(), gk_min_support(), gk_panel(), gk_signal_policy(), gk_validate_config(), hierarchy-elements

Examples

```
path <- tempfile(fileext = ".json")
gk_write_config(gk_example_config(), path)
cfg <- gk_read_config(path)
identical(gk_config_hash(cfg), gk_config_hash(gk_example_config()))
unlink(path)
```

`hierarchy-elements` *Hierarchy rules, flags, refinements and overrides*

Description

[Experimental]

Building blocks of a phenotype hierarchy:

- `gk_rule()` assigns `label` to cells positive for every marker in `all_of`, for at least one marker in `any_of` (ignored when empty) and negative for every marker in `none_of`. Rules are evaluated in order and the first match wins.
- `gk_flag()` is a named Boolean expression evaluated before the rules; its policy (`exclude`, `flag_only` or `unresolved`) is set in `gk_hierarchy()`.
- `gk_refinement()` relabels cells of `from_label` that are positive for an enabled state call (`marker` within `parent`), e.g. CD4 T cells that are FOXP3-positive become Treg cells.
- `gk_override()` replaces the threshold method or arguments of one marker on one image, with a mandatory reason. Overrides are part of the configuration and its hash; they never live in code.

Usage

```
gk_rule(
  rule_id,
  label,
  all_of = character(),
  any_of = character(),
  none_of = character(),
  on_uncallable = "disable_rule",
  review = NULL,
  pending_label = NULL,
  decisions = NULL
)

gk_flag(all_of = character(), any_of = character(), none_of = character())

gk_refinement(rule_id, marker, parent, from_label, to_label)

gk_override(
```

```
  image_id,
  marker,
  parent = NA_character_,
  threshold_method = NULL,
  threshold_args = NULL,
  reason
)
```

Arguments

rule_id	Stable identifier, e.g. "R010".
label	Cell type assigned by the rule.
all_of, any_of, none_of	Character vectors of marker names.
on_uncallable	What to do with markers in none_of that are not callable on an image: "disable_rule" (default) or "treat_as_negative". Markers in all_of or any_of that are not callable always disable the rule.
review	NULL, or "structure": matching cells start as pending_label until a structure decision exists (see gk_decide_structure()).
pending_label	Label of cells awaiting a structure decision.
decisions	Named list or character vector mapping each decision (Tumor, Benign, Unresolved) to a label.
marker, parent	State marker and parent set of a refinement or override.
from_label, to_label	Labels before and after refinement.
image_id	Image the override applies to.
threshold_method, threshold_args	Replacement method and arguments; NULL keeps the panel's.
reason	Why the override exists (required, recorded).

Value

Objects of class gk_rule, gk_flag, gk_refinement or gk_override.

See Also

Other configuration: gk_config(), gk_config_hash(), gk_hierarchy(), gk_min_support(), gk_panel(), gk_signal_policy(), gk_validate_config(), gk_write_config()

Examples

```
gk_rule("R020", "CD8 T cells", all_of = c("CD45", "CD3e", "CD8"), none_of = "CD4")
gk_rule(
  "R010", "Epithelial cells",
  all_of = "PanCK", review = "structure",
  pending_label = "Unreviewed epithelial cells",
  decisions = c(
```

```
    Tumor = "Tumor cells", Benign = "Benign epithelial cells",
    Unresolved = "Unresolved epithelial cells"
  )
)
gk_flag(all_of = "PanCK", any_of = c("CD45", "CD3e"))
gk_refinement("RF10", "FOXP3", "CD4 T cells", "CD4 T cells", "Treg cells")
gk_override("slide-07", "CD3e",
  threshold_args = list(rule = "crossing"),
  reason = "Dim staining batch; q99 rule over-conservative (review 2026-03)"
)
```

Index

∗ **app**
gk_app, 3
∗ **cell tables**
gk_cellspec, 6
gk_write_cellspec, 49
∗ **classification**
gk_apply_candidate_rules, 4
gk_classify, 7
gk_fixed_signal_calls, 18
gk_state_calls, 43
gk_structures, 44
∗ **configuration**
gk_config, 8
gk_config_hash, 9
gk_hierarchy, 21
gk_min_support, 26
gk_panel, 27
gk_signal_policy, 41
gk_validate_config, 47
gk_write_config, 50
hierarchy-elements, 51
∗ **corrections**
gk_correct, 10
gk_fit_residual_model, 16
gk_import_residual_model, 23
∗ **example data**
gk_example_config, 14
gk_example_path, 15
gk_simulate, 42
∗ **export**
gk_export, 16
gk_lock, 25
gk_read_export, 34
gk_verify_export, 48
∗ **plots**
gk_plot_density, 29
gk_plot_dotplot, 29
gk_plot_embedding, 30
gk_plot_map, 31
gk_plot_overview, 31
gk_plot_states, 32
gk_plot_thresholds, 32
∗ **preparation**
gk_callability, 5
gk_prepare, 33
∗ **review**
gk_decide_structure, 11
gk_dispose, 12
gk_import_cuts, 22
gk_ledgers, 24
gk_load_review, 25
gk_replace_review, 35
gk_replay, 36
gk_revert_last, 36
gk_review, 37
gk_save_review, 38
gk_set_cut, 39
gk_set_state, 40
∗ **thresholds**
gk_embed, 13
gk_threshold_methods, 45
gk_thresholds, 46

base::saveRDS(), *17*

gk_app, 3
gk_apply_candidate_rules, 4
gk_apply_candidate_rules(), *8*, *19*, *44, 45*
gk_apply_residual_model
*(*gk_fit_residual_model*)*, 16
gk_apply_residual_model(), *10*
gk_callability, 5
gk_callability(), *26*, *33, 34*
gk_cellspec, 6
gk_cellspec(), *33*, *43*, *50*
gk_classify, 7
gk_classify(), *5*, *19*, *37*, *44, 45*
gk_config, 8

gk_config(), *4*, *10*, *14*, *22*, *27, 28*, *33*, *41*, *47*, *50*, *52*
gk_config_hash, 9
gk_config_hash(), *8, 9*, *22*, *27, 28*, *41*, *47*, *50*, *52*
gk_correct, 10
gk_correct(), *8*, *13*, *17*, *23*, *46*
gk_decide_structure, 11
gk_decide_structure(), *12*, *22*, *24, 25*, *35–40*, *52*
gk_dispose, 12
gk_dispose(), *12*, *22*, *24, 25*, *35–40*
gk_embed, 13
gk_embed(), *46, 47*
gk_example_config, 14
gk_example_config(), *15*, *42, 43*
gk_example_path, 15
gk_example_path(), *14*, *43*
gk_export, 16
gk_export(), *15*, *26*, *34*, *48*
gk_fit_residual_model, 16
gk_fit_residual_model(), *11*, *23*
gk_fixed_signal_calls, 18
gk_fixed_signal_calls(), *5*, *8*, *44, 45*
gk_fixed_state_diagnostics, 19
gk_fixed_threshold_calls, 20
gk_flag *(*hierarchy-elements*)*, 51
gk_flag(), *21*
gk_hierarchy, 21
gk_hierarchy(), *9, 10*, *27, 28*, *41*, *47*, *50–52*
gk_import_cuts, 22
gk_import_cuts(), *12*, *24, 25*, *35–40*
gk_import_residual_model, 23
gk_import_residual_model(), *11*, *17*
gk_ledgers, 24
gk_ledgers(), *12*, *22*, *25*, *35–40*
gk_load_review, 25
gk_load_review(), *12*, *22*, *24*, *35–40*
gk_lock, 25
gk_lock(), *15, 16*, *34*, *48*
gk_min_support, 26
gk_min_support(), *9, 10*, *22*, *28*, *41*, *45*, *47*, *50*, *52*
gk_override *(*hierarchy-elements*)*, 51
gk_override(), *9*
gk_panel, 27
gk_panel(), *9, 10*, *22*, *27*, *41*, *45*, *47*, *50*, *52*
gk_plot_density, 29
gk_plot_density(), *30–33*
gk_plot_dotplot, 29
gk_plot_dotplot(), *29–33*
gk_plot_embedding, 30
gk_plot_embedding(), *29–33*
gk_plot_map, 31
gk_plot_map(), *29, 30*, *32, 33*
gk_plot_overview, 31
gk_plot_overview(), *29–33*
gk_plot_states, 32
gk_plot_states(), *29–33*
gk_plot_thresholds, 32
gk_plot_thresholds(), *29–32*
gk_prepare, 33
gk_prepare(), *5, 6*, *8*, *10*, *13*, *37*, *44–46*
gk_read_cellspec *(*gk_write_cellspec*)*, 49
gk_read_config *(*gk_write_config*)*, 50
gk_read_export, 34
gk_read_export(), *16*, *26*, *48*
gk_refinement *(*hierarchy-elements*)*, 51
gk_refinement(), *9*
gk_replace_review, 35
gk_replace_review(), *12*, *22*, *24, 25*, *36–40*
gk_replay, 36
gk_replay(), *12*, *22*, *24, 25*, *35*, *37–40*
gk_revert_last, 36
gk_revert_last(), *12*, *22*, *24, 25*, *35–40*
gk_review, 37
gk_review(), *4*, *11, 12*, *15*, *22*, *24–26*, *35–40*
gk_rule *(*hierarchy-elements*)*, 51
gk_rule(), *21*
gk_save_review, 38
gk_save_review(), *4*, *12*, *15*, *22*, *24, 25*, *35–37*, *39, 40*
gk_set_cut, 39
gk_set_cut(), *12*, *22*, *24, 25*, *35–38*, *40*
gk_set_state, 40
gk_set_state(), *12*, *22*, *24, 25*, *35–39*
gk_signal_policy, 41
gk_signal_policy(), *9, 10*, *22*, *27, 28*, *47*, *50*, *52*
gk_simulate, 42
gk_simulate(), *14, 15*
gk_state_calls, 43
gk_state_calls(), *5*, *8*, *19*, *27*, *37*, *45, 46*
gk_structures, 44
gk_structures(), *5*, *8*, *19*, *37*, *44*
gk_threshold_methods, 45

gk_threshold_methods(), *14*, *28*, *47*
gk_thresholds, 46
gk_thresholds(), *8*, *14*, *37*, *44*, *46*
gk_validate_config, 47
gk_validate_config(), *9, 10*, *22*, *27, 28*, *41*, *50*, *52*
gk_verify_export, 48
gk_verify_export(), *4*, *16*, *26*, *34*
gk_write_cellspec, 49
gk_write_cellspec(), *7*
gk_write_config, 50
gk_write_config(), *9, 10*, *22*, *27, 28*, *41*, *47*, *52*

hierarchy-elements, 51