Package 'HiCPotts'

November 13, 2026

Title Hierarchical Modeling to Identify and Correct Genomic Biases in Hi-C

Version 1.3.1

Date 2026-09-06

Description Bayesian analysis of Hi-C interaction counts using a three-state hierarchical mixture model with Potts spatial dependence and genomic distance, GC-content, transposable-element and accessibility covariates. The three biological components are low-mean noise, true signal with a distinct covariate-response pattern, and elevated false signal whose covariate-response slopes resemble the noise component. Robust regression fitting uses a multi-chain soft empirical-Bayes pilot to construct one shared prior that is frozen for all production chains, dispersion uses component-group-specific Gamma priors,
zero inflation uses a conjugate augmented Gibbs step, and spatial coupling uses retained-state approximate Bayesian computation. The official classification pools post-burn-in latent-state frequencies from the fitted model; parameter-plus-Potts allocation is retained as a separate sensitivity analysis. Parameter summaries include posterior intervals, effective sample sizes and split-chain R-hat, with configurable diagnostic criteria for reporting. Cached native likelihood calculations, direct checkerboard allocation, reproducible cross-platform parallel chains, fit provenance and stage timings improve computational efficiency and auditability without changing the model target or official classification rule.

License GPL-3 | file LICENSE

Encoding UTF-8

URL https://github.com/igosungithub/HiCPotts

BugReports https://github.com/igosungithub/HiCPotts/issues

biocViews StatisticalMethod, FunctionalGenomics, GenomeAnnotation, GenomeWideAssociation, PeakDetection, DataImport, Spatial, Bayesian, Classification, HiddenMarkovModel, Regression

Depends R (>= 4.5)

Imports Rcpp (>= 0.11.0), Biostrings, GenomicRanges, IRanges, S4Vectors, ggnewscale, parallel, rhdf5, rlang, rtracklayer, stats, withr

Suggests BSgenome, BSgenome.Dmelanogaster.UCSC.dm6, BiocManager, BiocStyle, ggplot2 (>= 3.5.0), knitr (>= 1.30), reshape2 (>= 1.4.4), rmarkdown (>= 2.10), testthat (>= 3.0.0)

LinkingTo Rcpp, RcppArmadillo

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation yes

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

git_url https://git.bioconductor.org/packages/HiCPotts

git_branch devel

git_last_commit 5d3f5d0

git_last_commit_date 2026-09-06

Repository Bioconductor 3.24

Date/Publication 2026-11-12

Author Itunu. Godwin Osuntoki [aut, cre] (ORCID: <https://orcid.org/0009-0005-1037-9346>), Nicolae. Radu Zabet [aut]

Maintainer Itunu. Godwin Osuntoki <hitunes4@gmail.com>

Contents

HiCPotts-package 3
allocation_diagnostics 3
classify_hicpotts 5
combine_hicpotts_blocks 7
compute_HMRFHiC_probabilities 8
diagnose_hicpotts_fit 12
get_data 14
plot_hicpotts_mcmc_by_component 16
plot_upper_prob_lower_count 17
posterior_predictive_hicpotts 19
process_data 20
relabel_hicpotts 22
run_chain_betas 23
summarise_hicpotts_parameters 26
summarise_hicpotts_probabilities 28

Index **30**

HiCPotts-package *HiCPotts: Hierarchical Modeling to Identify and Correct Genomic Biases in Hi-C*

Description

Bayesian analysis of Hi-C interaction counts using a three-state hierarchical mixture model with Potts spatial dependence and genomic distance, GC-content, transposable-element and accessibility covariates. The three biological components are low-mean noise, true signal with a distinct covariate-response pattern, and elevated false signal whose covariate-response slopes resemble the noise component. Robust regression fitting uses a multi-chain soft empirical-Bayes pilot to construct one shared prior that is frozen for all production chains, dispersion uses component-group-specific Gamma priors, zero inflation uses a conjugate augmented Gibbs step, and spatial coupling uses retained-state approximate Bayesian computation. The official classification pools post-burn-in latent-state frequencies from the fitted model; parameter-plus-Potts allocation is retained as a separate sensitivity analysis. Parameter summaries include posterior intervals, effective sample sizes and split-chain R-hat, with configurable diagnostic criteria for reporting. Cached native likelihood calculations, direct checkerboard allocation, reproducible cross-platform parallel chains, fit provenance and stage timings improve computational efficiency and auditability without changing the model target or official classification rule.

Author(s)

Maintainer: Itunu. Godwin Osuntoki <hitunes4@gmail.com> (ORCID)

Authors:

- Nicolae. Radu Zabet <r.zabet@qmul.ac.uk>

See Also

Useful links:

- https://github.com/igosungithub/HiCPotts
- Report bugs at https://github.com/igosungithub/HiCPotts/issues

allocation_diagnostics *Monte-Carlo reliability of the per-cell latent-state probabilities*

Description

Quantifies how precisely the sampler has estimated each cell's membership probabilities, using the batched post-burn-in latent-state frequencies recorded by `run_chain_betas()`.

Allocation draws are serially correlated, so the naive binomial standard error $\sqrt{p(1-p)/n}$ understates the Monte-Carlo error of a membership probability. Batch means give an autocorrelation-aware estimate: the post-burn-in draws are split into contiguous batches and the spread between batch means carries the dependence.

This is a diagnostic layer only. It reports how reliable the probabilities are; it does not modify them, and the three-way MAP label is unchanged.

Usage

```
allocation_diagnostics(
  fit,
  component_names = c("noise", "signal", "false signal")
)
```

Arguments

`fit`	A raw chain, list of chains, robust result returned by `run_chain_betas()`, or block-aware result from `combine_hicpotts_blocks()`.
`component_names`	Character vector of length three naming components 1, 2 and 3.

Value

A list with:

`mcse` Three $N \times N$ matrices of per-cell Monte-Carlo standard errors, one per component.

`ess` Three $N \times N$ matrices of per-cell allocation effective sample sizes.

`worst_cell_ess` The smallest effective sample size over all cells and components – the binding constraint on the classification.

`max_mcse` The largest per-cell Monte-Carlo standard error.

`between_chain_disagreement` `NULL` for a single chain; otherwise the mean and maximum per-cell total-variation distance between chains' biologically labelled membership probabilities.

See Also

`classify_hicpotts`

Examples

```
batch_means <- array(
    c(
        rep(0.70, 4), rep(0.20, 4), rep(0.10, 4),
        rep(0.65, 4), rep(0.25, 4), rep(0.10, 4)
    ),
    dim = c(2, 2, 3, 2)
)
fit <- list(
    chains = lapply(seq_len(3L), function(k) matrix(k, 2, 1)),
    z_probability_batches = list(
        batch_means = batch_means,
        batch_draws = c(10, 10)
    )
)
diagnostic <- allocation_diagnostics(fit)
diagnostic$worst_cell_ess
diagnostic$max_mcse
```

`classify_hicpotts` *Officially classify interactions from sampled HiCPotts latent states*

Description

Combines the post-burn-in latent-state membership frequencies stored by the HiCPotts sampler across one or more relabelled chains. This is the package's official three-state classification and a classification-layer summary of the fitted model: it does not refit the model, change an MCMC transition, or apply additional spatial smoothing.

Usage

```
classify_hicpotts(
  fit,
  data = NULL,
  component_names = hicpotts_component_definition()$label,
  use = c("selected", "all"),
  relabel = TRUE,
  min_draws = 100L,
  reflect = c("auto", "always", "never")
)
```

Arguments

`fit`	A raw chain, list of chains, `hicpotts_robust_fit`, or block-aware result from `combine_hicpotts_blocks()`.
`data`	Optional data frame in the same column-major lattice order as `process_data()`. It must contain one row per lattice cell. When omitted, row and column indices are returned.
`component_names`	Character vector of length three giving the semantic names of components 1, 2 and 3, in that order. The default is the package's canonical naming, `c("noise", "signal", "false signal")`, where component 1 is low-mean noise, component 2 is a true interaction with an unrestricted covariate-response pattern, and component 3 is elevated noise whose covariate-response pattern resembles component 1. Supply your own wording to rename the reported classes; this relabels the output only and does not change which component a cell is assigned to. The default is produced internally, so it is shown in the usage above as a function call rather than as a literal vector.
`use`	For a robust fit, use the mode-consistent chains selected for parameter summaries (`"selected"`, the default) or every retained chain (`"all"`). The selected result is conditional on that posterior mode; a warning is emitted when another replicated mode exists.
`relabel`	Whether to apply the package's component-2/3 relabelling and cross-chain allocation alignment before pooling. The default is `TRUE`.
`min_draws`	Minimum number of post-burn-in latent-state draws, pooled across chains, required before a classification is returned. Classifying from very few draws produces near-degenerate probabilities whose margin and entropy diagnostics are not meaningful, so this is an error rather than a warning. There is no fallback to the single final latent state. Defaults to `100`; lower it deliberately if a coarser classification is acceptable.

`reflect` Reflection averaging for mirrored contact matrices. On a symmetric input, cells `(i,j)` and `(j,i)` are two stored copies of one measured contact and must receive the same label; averaging their pooled probabilities guarantees this. `"auto"` (the default) applies it when every fit records a confirmed symmetric input and the lattice is square, `"always"` forces it, `"never"` disables it.

Details

Unlike a classification based on the single final state or on posterior-mean parameters, the returned probabilities average the latent allocations actually sampled under the full count and Potts model. Each cell is assigned to its maximum-posterior component; probability margin and entropy are returned as diagnostics.

Value

A data frame containing `prob1`, `prob2`, `prob3`, the maximum-posterior component and label, the winning probability, probability margin, normalized entropy, and the classification. Attributes record the number of chains and latent-state draws pooled and the classification scope.

Working with sub-blocks of a large map

This function summarises one contact map. It cannot span sub-blocks: chains fitted to different blocks have different data and different frozen priors, so they are refused rather than pooled, and passing a whole-map `data` frame to a single block's fit fails the one-row-per-lattice-cell check.

Run it per block and stitch the results, which lets every block keep the chains its own mode screen accepted – block 1 may be summarised from chains 1 and 3 while block 2 uses chains 1 and 2:

```
parts <- list()
for (j in seq_along(fits)) {
    parts[[j]] <- classify_hicpotts(fits[[j]], data = blocks[[j]])
    parts[[j]]$block <- names(fits)[j]
}
combined <- do.call(rbind, parts)
```

`combine_hicpotts_blocks()` does this with the checks that matter (matching lengths, one row per cell, non-overlapping blocks) and records the chains each block used. Note that component labels are only aligned within a block, not across blocks.

See Also

`compute_HMRFHiC_probabilities`, `relabel_hicpotts`

Examples

```
N <- 3L
probs <- list(
    component1 = matrix(0.8, N, N),
    component2 = matrix(0.15, N, N),
    component3 = matrix(0.05, N, N)
)
fake_fit <- list(
    chains = lapply(1:3, function(k) matrix(k, 4, 5)),
    size = matrix(1, 3, 4),
    z_final = matrix(1, N, N),
```

```
    z_checkpoints = list(),
    z_probabilities = probs,
    z_probability_draws = 200L
)
classify_hicpotts(fake_fit, relabel = FALSE)
```

`combine_hicpotts_blocks` *Combine independently fitted HiCPotts matrix blocks*

Description

Create one block-aware HiCPotts object from fits to submatrices of a larger contact map. Each block retains the chains selected by its own mode screen. Downstream functions analyse every block independently and combine only compatible cell-level results.

Usage

```
combine_hicpotts_blocks(
  fits,
  data = NULL,
  processed = NULL,
  what = c("fit", "classification", "probabilities"),
  ...
)
```

Arguments

`fits`	A fitted result, or a list containing one fitted result per matrix block. Names, when present, label the blocks.
`data`	Optional original data frame, or list of one data frame per block. For multiple blocks, a whole-map data frame is split sequentially using the fitted lattice sizes.
`processed`	Optional structured result from `process_data`. For convenience, it may be supplied as the second positional argument.
`what`	Return a reusable block-aware `'fit'` (the default), or immediately return combined `'classification'` or `'probabilities'` output.
`...`	Arguments passed to `classify_hicpotts` or `compute_HMRFHiC_probabilities` when requested by `what`.

Details

MCMC draws from different matrix blocks target different posteriors and are retained separately. For a robust fit, `mode_selection$selected` is reapplied to `all_fits`; consequently block 1 can use chains 1 and 3 while block 2 uses chain 2. The selected chains are recorded in `block_table`. `summarise_hicpotts_parameters(pool_blocks = 'posterior_weighted')` can additionally calculate a descriptive, occupancy-aware average from independent posterior draws. It is explicitly labelled as an aggregate and is not a substitute for a joint shared-parameter fit.

A single fit is returned unchanged when `what = 'fit'`. Thus callers do not need a special branch for an analysis containing only one matrix.

The structured output of process_data may be supplied through processed. It is sufficient for classification, allocation diagnostics, parameter summaries and posterior predictive checks. The parameter-based probability sensitivity function additionally needs the original data frame(s), because genomic coordinates and untransformed covariates cannot be reconstructed from processed matrices.

Value

For multiple blocks and what = 'fit', an object of class hicpotts_block_fit. For one block, the original fit is returned. The other values of what return a combined data frame with block provenance.

See Also

classify_hicpotts, compute_HMRFHiC_probabilities, summarise_hicpotts_parameters

Examples

```
make_example_fit <- function(component, N = 2L) {
    probabilities <- lapply(1:3, function(k) {
        matrix(if (k == component) 0.9 else 0.05, N, N)
    })
    list(
        chains = lapply(1:3, function(k) matrix(k, 4L, 5L)),
        z_final = matrix(component, N, N),
        z_probabilities = probabilities,
        z_probability_draws = 20L
    )
}
block_fit <- combine_hicpotts_blocks(list(
    left = make_example_fit(1L), right = make_example_fit(2L)
))
classified <- classify_hicpotts(
    block_fit, relabel = FALSE, min_draws = 1L
)
table(classified$block, classified$map_component)
```

---

compute_HMRFHiC_probabilities

Compute HiCPotts Probabilities of Assigning an Interaction to Each Component

---

Description

Computes a secondary parameter-based allocation of genomic interactions to each of three HiCPotts components. The official classification is returned by classify_hicpotts() from sampled latent-state frequencies. This function is retained for parameter-plus-Potts sensitivity analysis and backwards compatibility.

Posterior mean regression parameters are extracted from MCMC output and used to evaluate the component-wise likelihood of each observed interaction count. The resulting probabilities are normalized to sum to 1 for each interaction.

Optionally, the MCMC output can be relabeled before posterior summaries are computed. This helps reduce label-switching effects across MCMC draws and chains.

Usage

```
compute_HMRFHiC_probabilities(
  data, chain_betas, iterations,
  dist = "ZINB",
  max_interactions = NA_integer_,
  consistent_dist = FALSE,
  relabel = FALSE,
  N = NULL,
  potts_iterations = 5L,
  method = c("integrated", "plugin"),
  n_draws = 200L,
  component_definition = hicpotts_component_definition()
)
```

Arguments

data	A data.frame, or block data accepted by combine_hicpotts_blocks(), with required columns: start, end, interactions, GC, TES, and ACC.
chain_betas	A list of MCMC chain results, robust fit, or block-aware result from combine_hicpotts_blocks().
iterations	Total number of MCMC iterations.
dist	Distribution used for count modeling. One of "Poisson", "NB", "ZIP", or "ZINB".
max_interactions	Optional cap on interactions before density evaluation. Defaults to NA, meaning no cap.
consistent_dist	Logical; controls the family used for components 2 and 3. Defaults to FALSE, which matches the fitted model: the sampler applies structural zero inflation to component 1 only, so components 2 and 3 use "Poisson" or "NB" for their emission density, matching the likelihood the sampler evaluates. TRUE gives all three components the zero-inflated family and therefore does NOT correspond to any model this package fits; with equal means it drives every observed zero to a flat 1/3, 1/3, 1/3 instead of the decisive component-1 call the fitted likelihood implies. It is retained only for backwards comparison.
relabel	Logical; if TRUE, relabel each fitted chain result before posterior means are computed.
N	Optional integer lattice dimension. If data represents an $N \times N$ lattice in the same row-order convention process_data() uses (i.e. nrow(data) == N^2), supplying N enables a Potts/neighbour spatial smoothing term in the returned probabilities, matching the neighbour coupling the sampler applies when it draws the latent states during MCMC. If omitted, the function returns per-cell-likelihood-only probabilities and emits a warning (see Details).
potts_iterations	Number of mean-field Potts smoothing sweeps to run when N is supplied. Default 5.
method	Either "integrated" (default) or "plugin". "integrated" computes normalised component probabilities for each retained posterior draw and averages them, which is the integrated posterior membership probability. "plugin" reproduces the legacy behaviour: average the parameters first, then evaluate a

single density. Since $p(z \mid y, E[\theta]) \neq E[p(z \mid y, \theta)]$, the plug-in values understate uncertainty and should be described as approximate scores, not posterior probabilities.

`n_draws` Integer; number of posterior draws to thin to when `method = "integrated"`. Cost is linear in this value.

`component_definition`

Three-row data frame used to label the returned probability columns, with columns `component` (1, 2, 3), `label` and `definition`. The default is the package's canonical definition, equivalent to

```
data.frame(
  component  = 1:3,
  label      = c("noise", "signal", "false signal"),
  definition = c("low-mean noise",
                 "true interaction with an unrestricted
                  covariate-response pattern",
                "elevated noise with a covariate-response pattern
                  resembling component 1")
)
```

It is built internally, which is why the usage above shows a function call. Supplying your own wording renames the reported classes only; it does not change the fitted component identities.

Details

When `N` is supplied and `nrow(data) == N^2`, the per-cell log-densities are reshaped into $N \times N$ matrices (the same column-major row-order convention as `process_data()`) and a small number of mean-field Potts smoothing sweeps are run: each cell's smoothed log-probability is its own per-cell log-density plus the fitted `gamma` (posterior mean, extracted the same way as theta/size) times the current-estimate same-label probability mass among its four lattice neighbours, iterated `potts_iterations` times so the spatial term propagates a short distance – the mean-field (deterministic, iterated) analogue of the neighbour-agreement term the sampler applies by drawing latent states during MCMC. If `N` is not supplied (or does not satisfy `N^2 == nrow(data)`), the function returns the spatial-free probabilities and emits an explicit warning rather than silently guessing. The function:

1. checks the required input columns,
2. optionally relabels the fitted MCMC output,
3. discards the first half of iterations as burn-in,
4. extracts posterior means for each component,
5. evaluates the component-wise count density,
6. normalizes these values into posterior probabilities.

No hard probability threshold is applied. The returned `prob1`, `prob2`, and `prob3` columns are intended for users to threshold or otherwise interpret according to their scientific application.

Value

A `data.frame` containing the original columns plus:

- `prob1`
- `prob2`
- `prob3`

Working with sub-blocks of a large map

This function summarises one contact map. It cannot span sub-blocks: chains fitted to different blocks have different data and different frozen priors, so they are refused rather than pooled, and passing a whole-map `data` frame to a single block's fit fails the one-row-per-lattice-cell check.

Run it per block and stitch the results, which lets every block keep the chains its own mode screen accepted – block 1 may be summarised from chains 1 and 3 while block 2 uses chains 1 and 2:

```
parts <- list()
for (j in seq_along(fits)) {
    parts[[j]] <- compute_HMRFHiC_probabilities(
        blocks[[j]], fits[[j]], iterations)
    parts[[j]]$block <- names(fits)[j]
}
combined <- do.call(rbind, parts)
```

`combine_hicpotts_blocks()` does this with the checks that matter (matching lengths, one row per cell, non-overlapping blocks) and records the chains each block used. Note that component labels are only aligned within a block, not across blocks.

See Also

`relabel_hicpotts`, `dpois`, `dnbinom`

Examples

```
set.seed(4921)

sim_data <- data.frame(
    start = c(1e6, 2e6, 3e6, 4e6),
    end = c(2e6, 3e6, 4e6, 5e6),
    interactions = c(0, 5, 10, 20),
    GC = c(0.40, 0.45, 0.50, 0.55),
    TES = c(0.10, 0.20, 0.30, 0.40),
    ACC = c(0.30, 0.40, 0.50, 0.60)
)

iterations <- 10L
n_draw <- iterations + 1L

make_chain <- function(intercept) {
    cbind(
        rnorm(n_draw, intercept, 0.05),
        rnorm(n_draw, -0.01, 0.01),
        rnorm(n_draw, 0.01, 0.01),
        rnorm(n_draw, 0.01, 0.01),
        rnorm(n_draw, 0.01, 0.01)
    )
}

chain_betas <- list(
    list(
        chains = list(
            make_chain(-1),
            make_chain(0),
            make_chain(1)
```

```
        ),
        theta = runif(n_draw, 0.05, 0.20),
        size = matrix(10, nrow = 3, ncol = n_draw),
        gamma = runif(n_draw, 0.5, 0.8)
    )
)

prob_res <- compute_HMRFHiC_probabilities(
    data = sim_data,
    chain_betas = chain_betas,
    iterations = iterations,
    dist = "ZINB",
    relabel = TRUE
)

head(prob_res)
```

`diagnose_hicpotts_fit` *Diagnose HiCPotts parameter estimation*

Description

Reports posterior estimates, intervals, sign probabilities, ESS and R-hat, plus internal parameter occupancy and reported classification occupancy. Optional probability output adds mean classification entropy.

Usage

```
diagnose_hicpotts_fit(
  fit,
  burnin = NULL,
  ci_level = 0.95,
  prob_result = NULL,
  minimum_component_cells = 0L,
  minimum_ess = 200,
  maximum_rhat = 1.01,
  minimum_chains = 4L,
  gamma_boundary_tolerance = 0.01,
  maximum_gamma_boundary_fraction = 0.95,
  minimum_gamma_unique = 20L,
  covariate_diagnostics = NULL,
  relabel = TRUE
)
```

Arguments

`fit`	A fit, list of independently seeded fits of the same data, or block-aware result from `combine_hicpotts_blocks()`.
`burnin`	Initial rows to discard; defaults to half.
`ci_level`	Posterior interval level.
`prob_result`	Optional output from `compute_HMRFHiC_probabilities()`.

`minimum_component_cells`
: Non-negative occupancy threshold. The default zero permits empty components and retains occupancy counts for reporting.

`minimum_ess`
: Minimum recommended effective sample size.

`maximum_rhat`
: Maximum recommended split-chain R-hat.

`minimum_chains`
: Minimum recommended number of independent chains.

`gamma_boundary_tolerance`
: Distance from zero or one counted as the boundary region for the bounded Potts parameter.

`maximum_gamma_boundary_fraction`
: Largest allowed fraction of retained gamma draws in the boundary region in every chain.

`minimum_gamma_unique`
: Minimum number of distinct retained gamma draws required in every chain. A frozen gamma chain is rejected explicitly even when another diagnostic is unavailable.

`covariate_diagnostics`
: Optional pre-computed covariate conditioning report, used to fill the `covariate_conditioning` reliability gate. It is normally left as `NULL`. `summarise_hicpotts_parameters()` builds one internally and passes it through when it is given `x_vars` and `diagnose_covariates = TRUE` (the default); for a `hicpotts_robust_fit` without `x_vars` it reuses the report already stored on the fit. Supply a value here only to reuse a report you computed yourself. When `NULL` the covariate gate is recorded as passing, because no conditioning problem has been demonstrated – it is not evidence that the covariates are well conditioned.

`relabel`
: Whether to relabel components 2 and 3 before pooling chains.

Details

The returned `reliability_flags` table holds one row per diagnostic criterion, with its configured threshold and observed result. Parameter summaries and cell-level classifications have separate diagnostic outputs. Inspect the criteria relevant to the quantity being reported, together with its posterior interval and Monte Carlo precision.

`mcse_precision` The batch-means Monte Carlo standard error reached its relative threshold before the requested iterations were exhausted.

`beta_movement` Every component accepted at least one ordinary retained regression-coefficient proposal. Branch-label swaps are counted separately and cannot satisfy this gate.

`common_posterior_target` All production chains use the same frozen regression-prior hyperparameters, as required for interpretable multi-chain R-hat and ESS diagnostics.

`gamma_not_boundary` In every chain, less than `maximum_gamma_boundary_fraction` of the retained gamma draws lie within `gamma_boundary_tolerance` of zero or one. This records posterior concentration near the limits of the support.

`gamma_movement` Every chain retained at least `minimum_gamma_unique` distinct gamma values. This catches a frozen gamma chain even when the boundary test passes.

`component_occupancy` Every component was allocated at least `minimum_component_cells` lattice cells internally. Occupancy describes the data contributing to each component.

`effective_sample_size` Bulk and tail ESS both reach `minimum_ess` for every parameter. A non-finite ESS marks a stuck chain and fails the gate rather than being dropped. The tail is checked separately because a chain can mix well in the bulk with different precision in its outer quantiles.

`split_rhat` Split-chain R-hat is at most `maximum_rhat` for every parameter. This needs at least two chains; with a single chain the statistic is undefined and the gate fails.

`independent_chains` At least `minimum_chains` independently seeded chains were supplied.

`relabelled` Every fit carries a relabelling rule, so components 2 and 3 mean the same thing in each chain before chains are pooled.

`coefficient_scale` All chains report coefficients on the original log1p-covariate scale, so estimates from different chains are comparable.

`covariate_conditioning` Carried through from `covariate_diagnostics` when supplied; it flags collinear or ill-conditioned covariates.

`summarise_hicpotts_parameters()` applies the subset of criteria described on its help page. With `require_reliable = FALSE`, it displays estimates together with their diagnostic indicators. `classify_hicpotts()` provides the cell-level classification.

Value

A list of diagnostics: `parameters` (estimates, intervals, ESS and R-hat), `reliability_flags` (one row per gate, with its threshold), `component_occupancy`, `classification_entropy`, `mcse_status`, `gamma_diagnostics`, `beta_mixing_status`, `noise_relationship_probabilities`, the `fits` used and any `warnings`.

Examples

```
N <- 5
y <- matrix(rpois(N * N, 4), N, N)
mk <- function() list(matrix(runif(N * N), N, N))
x_vars <- list(distance = mk(), GC = mk(), TES = mk(), ACC = mk())
fits <- run_chain_betas(
    N = N, iterations = 5, x_vars = x_vars, y = y,
    dist = "ZINB", theta_start = 0.5, size_start = c(1, 1, 1),
    n_chains = 2, seeds = 1:2, robust = FALSE
)
d <- diagnose_hicpotts_fit(
    fits,
    minimum_component_cells = 1, minimum_ess = 1
)
d$reliability_flags[, c("criterion", "passed")]
d$classification_entropy
```

`get_data` *Extract Hi-C Bin Interactions with GC, Accessibility, and TE Counts*

Description

Reads a Hi-C contact matrix from a `.cool` or `.h5` (HDF5) file, subsets to a specified genomic region, rebins to a desired resolution (merging or slicing as needed), and returns a complete grid of bin-pair interactions. For each bin-pair it computes:

- GC content from a BSgenome
- DNase-I accessibility (if provided, lifted over)
- Transposable element (TE) overlap counts
- Interaction counts

Usage

```
get_data(
  file_path, chr, start, end, resolution,
  genome_package = NULL, acc_wig = NULL,
  chain_file = NULL, te_granges = NULL
)
```

Arguments

file_path	Path to a .cool file or a supported HDF5 schema. Binary .hic files are not read directly; convert them to .cool or extract counts with strawr::straw() first.
chr	Chromosome name (e.g. "chr2L").
start	1-based start coordinate of the region.
end	End coordinate of the region.
resolution	Target bin size (bp); smaller native bins are merged, larger ones are sliced.
genome_package	Optional name of a BSgenome package (e.g. "BSgenome.Dmelanogaster.UCSC.dm6") for GC calculations. When NULL, the returned GC column is NA.
acc_wig	(Optional) Path to a DNase-I wig (bedGraph) file. Requires chain_file to lift over.
chain_file	(Optional) LiftOver chain file for mapping acc_wig.
te_granges	(Optional) Path to a BED/GTF of TE annotations or a GRanges object. Only TEs on chr are counted.

Value

A data.frame with one row per bin-pair across [start,end] at resolution, containing:

start Bin1 start.

end.i Bin1 end.

start.j Bin2 start.

end Bin2 end.

chrom Chromosome name.

GC Combined GC fraction of the two bins.

ACC Mean DNase-I score per bin or NA.

TES Sum of TE overlaps in both bins or NA.

interactions Observed contact count (0 if absent).

See Also

h5read, liftOver

Examples

```
## A 500 kb window of the bundled chr4 Hi-C matrix at 100 kb resolution.
## GC, accessibility and TE columns stay NA unless the optional
## annotation arguments are supplied.
cool_file <- system.file(
    "extdata", "BG3_WT_merged_hic_matrix_chr4_100Kb.cool",
    package = "HiCPotts"
```

```
)
bins <- get_data(
    file_path  = cool_file,
    chr        = "chr4",
    start      = 1,
    end        = 500000,
    resolution = 100000
)
head(bins)
```

`plot_hicpotts_mcmc_by_component` *Plot HiCPotts MCMC traces by component*

Description

Produces trace plots for the component-specific regression coefficients and, when available, the size parameters. Optionally also plots the global parameters `gamma` and `theta`.

This function is intended for raw fitted HiCPotts objects returned by the MCMC fitting functions, not for the data frame returned by `compute_HMRFHiC_probabilities()`.

Usage

```
plot_hicpotts_mcmc_by_component(
  fit,
  index = 1L,
  burnin = NULL,
  component_names = c("Noise", "Signal", "False signal"),
  beta_names = c("Intercept", "Distance", "GC", "TES", "ACC"),
  plot_globals = TRUE,
  plot_size = TRUE,
  ask = interactive()
)
```

Arguments

`fit`	A single HiCPotts fit object or a list of such objects.
`index`	Integer index used when `fit` is a list.
`burnin`	Optional burn-in iteration. Defaults to half the iterations.
`component_names`	Character vector of length 3 giving component labels.
`beta_names`	Character vector giving the regression coefficient names.
`plot_globals`	Logical; if `TRUE`, also plot `gamma` and `theta` when present.
`plot_size`	Logical; if `TRUE`, plot the component-wise size parameters when present.
`ask`	Logical; if `TRUE` in an interactive session, pause between pages.

Value

Invisibly returns the extracted HiCPotts fit object used for plotting.

Examples

```
set.seed(4921)
n_draw <- 12L

make_chain <- function(intercept) {
    cbind(
        rnorm(n_draw, intercept, 0.05),
        rnorm(n_draw),
        rnorm(n_draw),
        rnorm(n_draw),
        rnorm(n_draw)
    )
}

fit <- list(
    chains = list(
        make_chain(-1),
        make_chain(0),
        make_chain(1)
    ),
    size = matrix(5, nrow = 3, ncol = n_draw),
    theta = runif(n_draw, 0.1, 0.2),
    gamma = runif(n_draw, 0.5, 0.8)
)

plot_hicpotts_mcmc_by_component(
    fit,
    burnin = 5,
    ask = FALSE
)
```

`plot_upper_prob_lower_count` *Plot a dual-triangle Hi-C heatmap*

Description

Creates a Hi-C style heatmap where the visually upper triangle shows posterior probabilities and the visually lower triangle shows observed interaction counts. Symmetric and ordered input matrices are both supported. This is designed for HiCPotts-style long-format output where each row represents an interaction pair. By default it uses columns commonly found in package outputs: `start`, `end`, `prob2`, and `interactions`.

Usage

```
plot_upper_prob_lower_count(
  results,
  bin1_col = "start",
  bin2_col = "end",
  prob_col = "prob2",
  count_col = "interactions",
  chr_label = "2L",
```

```
  title = "Significant interactions detected by HiCPotts",
  prob_agg = max,
  count_agg = max,
  use_log_count = TRUE,
  symmetric_matrix = TRUE
)
```

Arguments

results	A data.frame containing interaction results.
bin1_col	Column name for the first bin coordinate.
bin2_col	Column name for the second bin coordinate.
prob_col	Column name for the posterior probability to display in the upper triangle.
count_col	Column name for the observed interaction counts to display in the lower triangle.
chr_label	Chromosome label for the axes.
title	Plot title.
prob_agg	Aggregation function for duplicate probability rows. When symmetric_matrix = TRUE, it also combines the two orientations of an unordered pair. Defaults to max.
count_agg	Aggregation function for duplicate count rows. When symmetric_matrix = TRUE, it also combines the two orientations of an unordered pair. Defaults to max.
use_log_count	Logical; if TRUE, plot log1p(count) in the lower triangle. If FALSE, plot raw counts.
symmetric_matrix	Logical; if TRUE (the default), treat (i, j) and (j, i) as two orientations of the same Hi-C contact. A single complete triangle is reflected, while a full matrix is checked for mirrored counts before its duplicate orientations are aggregated. If FALSE, preserve ordered pairs and use the supplied upper and lower triangles without reflection or cross-triangle aggregation.

Value

A ggplot object.

Examples

```
sim_res <- data.frame(
    start = c(1e6, 1e6, 2e6, 2e6, 3e6),
    end = c(1e6, 2e6, 2e6, 3e6, 3e6),
    prob2 = c(0.10, 0.80, 0.20, 0.70, 0.30),
    interactions = c(5, 20, 8, 15, 6)
)

p <- plot_upper_prob_lower_count(sim_res)
print(p)
```

posterior_predictive_hicpotts *Conditional posterior-predictive diagnostics for HiCPotts*

Description

Generates replicated count matrices from posterior parameter draws, conditional on each chain's final internal (unforced) latent allocation. It reports cellwise log-count error, neighbour-correlation error, diagonal-decay error, overall and near-diagonal zero-fraction error, and variance-to-mean error. The reported forced-zero `z_final` is deliberately not used for these parameter diagnostics.

Usage

```
posterior_predictive_hicpotts(
  fit,
  x_vars = NULL,
  y = NULL,
  dist = "ZINB",
  burnin = NULL,
  n_rep = 100L,
  seed = 1L
)
```

Arguments

`fit`	One fit, a list of independently seeded fits, or block-aware result from `combine_hicpotts_blocks()`.
`x_vars`	Named HiCPotts covariates. May be omitted when a block-aware fit stored the structured output of `process_data()`.
`y`	Observed count matrix. May be omitted when a block-aware fit stored the structured output of `process_data()`.
`dist`	One of `Poisson`, `NB`, `ZIP` or `ZINB`.
`burnin`	Initial draws to discard; defaults to half of each chain.
`n_rep`	Number of replicated matrices.
`seed`	Integer simulation seed.

Value

A list with replicate discrepancies, interval summaries and observed matrix summaries.

Examples

```
N <- 6
y <- matrix(rpois(N * N, 4), N, N)
mk <- function() list(matrix(runif(N * N), N, N))
x_vars <- list(distance = mk(), GC = mk(), TES = mk(), ACC = mk())
fits <- run_chain_betas(
    N = N, iterations = 5, x_vars = x_vars, y = y,
    dist = "ZINB", theta_start = 0.5, size_start = c(1, 1, 1),
    n_chains = 2, seeds = 1:2, robust = FALSE
)
```

```
pp <- posterior_predictive_hicpotts(
    fits, x_vars, y,
    dist = "ZINB", n_rep = 5
)
pp$summary
```

`process_data` *CSV file Data Processing for Hi-C Interaction Matrices and Covariates*

Description

The `process_data` function takes a data frame with interaction information and associated covariates (genomic bins, GC content, transposable elements(TES), and accessibility) and converts it into a structured list of $N \times N$ matrices suitable for modeling. It also provides options for scaling interaction counts and checking for required columns and missing values.

Usage

```
process_data(data, N, scale_max = NA_real_,
     standardization_y = FALSE, pad_with_zero = FALSE,
     mirror = FALSE)
```

Arguments

`data`	A `data.frame` containing the following required columns: • `start`: Numeric start coordinates for locus i. • `end`: Numeric end coordinates for locus j. • `interactions`: Numeric vector of observed interaction counts between loci i and j. • `GC`: GC content measure for the given loci. • `ACC`: Accessibility score for the given loci. • `TES`: A measure related to transposable elements for the given loci.
`N`	An integer specifying the dimension of the resulting $N \times N$ matrices. The data provided should correspond to N^2 interactions or multiples thereof, as it will be reshaped into one or more $N \times N$ matrices.
`scale_max`	A numeric value indicating the maximum scaling factor for interaction counts when `standardization_y = TRUE`.
`standardization_y`	A logical value. If `TRUE`, interaction counts are scaled by applying a min-max normalization and rounding. If `FALSE`, the raw interaction counts are used as provided.
`pad_with_zero`	Logical; if TRUE, zero-pads the last block when nrow(data) is not a multiple of N^2.
`mirror`	Logical; if `TRUE`, treat `data` as an upper- or lower-triangle half matrix and fill the opposite triangle by reflection before reshaping. Hi-C contacts are symmetric, so cell (i, j) and cell (j, i) are two records of one measured contact; reflecting therefore adds no fabricated observations, unlike `pad_with_zero`. Rows are also returned in the column-major order the rest of the package assumes. A half matrix over N bins holds $N(N+1)/2$ rows with the diagonal or $N(N-1)/2$ without it, neither of which is a multiple of N^2, which is why such input otherwise fails the block check.

Details

This function processes a long-format data frame where each row represents an interaction between two loci (denoted by `start` and `end`) and associated covariate values. Key steps include:

1. Validating that required columns are present and checking for missing values.
2. Optionally scaling the interaction counts to a fixed range to reduce skew or prepare data for models sensitive to the magnitude of counts.
3. Computing the genomic distance between the two bins of each pair. When the `start.j.` column is present (as in `get_data()` output) the distance is $|start.j. - start|$, the separation between the two bin STARTS, which is zero on the diagonal and equal for both mirrored copies of a contact. Designs carrying only `start`/`end` fall back to $|end - start|$; note that `end` is bin j's END, so that legacy form spans both bins. The definition actually used is reported in a message.
4. Reshaping the vectorized interaction and covariate data into $N \times N$ matrices. If the number of rows in `data` is greater than N^2, it splits them into multiple $N \times N$ matrices (one for each segment of length N^2).
5. Storing the resulting data in a list structure suitable for downstream analyses.

The returned `x_vars` list contains multiple $N \times N$ matrices for each covariate:

- `x_vars$distance`: A list of one or more $N \times N$ matrices containing genomic distances.
- `x_vars$GC`: A list of $N \times N$ matrices for GC content.
- `x_vars$TES`: A list of $N \times N$ matrices for TES values.
- `x_vars$ACC`: A list of $N \times N$ matrices for accessibility scores.

The `y` element in the returned list contains the scaled (or raw) interaction counts structured as a list of $N \times N$ matrices.

Value

A list with two elements:

`x_vars` A named list containing lists of $N \times N$ matrices for each covariate: `distance`, `GC`, `TES`, `ACC`.

`y` A list of $N \times N$ matrices representing the (scaled) interaction counts corresponding to the covariates in `x_vars`.

See Also

`run_chain_betas` for downstream MCMC inference and modeling steps once data is processed into matrix form.

Examples

```
set.seed(123)
df <- data.frame(
    start = rep(1:10, each = 10),
    end = rep(11:20, times = 10),
    interactions = rpois(100, 5),
    GC = runif(100, 0, 1),
    TES = runif(100, 0, 1),
    ACC = runif(100, 0, 1)
)
```

```
processed <- process_data(df,
    N = 10, scale_max = NA_real_,
    standardization_y = FALSE, pad_with_zero = FALSE
)
# x_vars <- processed$x_vars
# y_matrices <- processed$y
# str(x_vars) # Show structure of covariates
# str(y_matrices) # Show structure of interaction matrices


# Extended example with larger dataset
# Suppose we have a data frame 'large_df' corresponding to a
# 20x20 interaction matrix
# large_df <- data.frame(
#  start = rep(1:20, each = 20),
#  end = rep(21:40, times = 20),
#  interactions = rpois(400, 5),
#  GC = runif(400, 0, 1),
#  TES = runif(400, 0, 1),
#  ACC = runif(400, 0, 1)
# )
# processed <- process_data(large_df, N = 20, scale_max = NA_real_,
#   standardization_y = FALSE, pad_with_zero = FALSE)
# x_vars <- processed[[1]]
# y_matrices <- processed[[2]]
# str(x_vars)
# str(y_matrices)
# See vignette("HiCPotts_vignette") for detailed examples with
# real Hi-C data.
#
```

`relabel_hicpotts` *Relabel HiCPotts output*

Description

Applies the component-1/3 biological noise relationship to either one HiCPotts result or a list of independent results. Every chain is labelled by the same scientific rule; no allocation-agreement or population-size tie-break can override it.

Usage

```
relabel_hicpotts(x)
```

Arguments

`x`	A single HiCPotts result or a list of HiCPotts results.

Value

A relabelled object with the same outer structure as `x`.

Examples

```
make_chain <- function(intercept, slopes) {
    cbind(
        intercept = rep(intercept, 4L),
        matrix(rep(slopes, each = 4L), nrow = 4L)
    )
}
fit <- list(
    chains = list(
        make_chain(0, rep(0, 4L)),
        make_chain(2, rep(1, 4L)),
        make_chain(3, rep(0, 4L))
    ),
    size = matrix(1, 3, 4),
    z_final = matrix(c(1, 2, 3, 1), 2, 2),
    z_checkpoints = list(),
    z_probabilities = lapply(seq_len(3L), function(k) matrix(1 / 3, 2, 2))
)
relabelled <- relabel_hicpotts(fit)
relabelled$component_definition
```

`run_chain_betas` *Fit one or more Hi-C datasets with HiCPotts*

Description

`run_chain_betas()` is the single public fitting entry point. It can run a user-selected number of independently seeded chains for one matrix or for each matrix in a list. With `robust = TRUE`, it also constructs priors, checks covariates, screens replicated allocation modes and applies the parameter-reliability gates. With `robust = FALSE`, it returns the raw relabelled chains. These choices change orchestration and reporting, not the fitted likelihood or sampler target.

Usage

```
run_chain_betas(
  N,
  x_vars,
  y,
  dist = "ZIP",
  gamma_start = 0.3,
  iterations = 20000L,
  burnin = NULL,
  n_chains = 1L,
  seeds = NULL,
  robust = FALSE,
  initialization = "auto",
  theta_start = NULL,
  size_start = NULL,
  use_data_priors = TRUE,
  user_fixed_priors = NULL,
  gamma_prior_shape = c(1, 1),
  epsilon = NULL,
```

```
  abc_potts_sweeps = 0L,
  abc_sim_reps = 4L,
  distance_metric = "manhattan",
  relabel = TRUE,
  mc_cores = 1L,
  verbose = FALSE,
  progress_interval = 50L,
  slope_sd_standardized = 0.5,
  abc_epsilon_quantile = 0.1,
  gamma_update_interval = 5L,
  z_probability_burnin = NULL,
  mcse_stop = TRUE,
  mcse_min_iterations = 10000L,
  mcse_check_interval = 500L,
  mcse_relative_threshold = 0.05,
  diagnostic_control = list(),
  gamma_prior = NULL
)
```

Arguments

N	Number of rows and columns in every square interaction matrix.
x_vars	Named list containing distance, GC, TES and ACC. Entries may be matrices for one dataset or lists of matrices matching y.
y	One numeric count matrix, or a list of independently fitted count matrices.
dist	One of "Poisson", "NB", "ZIP" or "ZINB".
gamma_start	Initial value of the Potts spatial parameter. This is a starting value, not the parameters of its Beta prior.
iterations	Positive maximum MCMC updates per chain.
burnin	Initial iterations excluded by robust diagnostics. NULL uses half the requested iterations, capped at 5,000.
n_chains	Positive number of independently seeded chains per dataset. The default of one preserves the historical run_chain_betas() result; four or more chains are recommended when convergence will be assessed.
seeds	Optional integer vector with exactly n_chains values. When omitted, seq_len(n_chains) is used; when n_chains is omitted, the length of seeds determines it.
robust	Logical. If TRUE, return the diagnostic and mode-screened robust workflow; if FALSE, return raw fitted chains.
initialization	Starting allocation strategy. "auto" uses two likelihood-informed starts followed by distance-adjusted and noise-anchored-random starts, recycled over the requested chains. A single method is used for every chain; alternatively supply exactly one method per chain. Available methods are "likelihood_informed", "count_quantile", "distance_adjusted", "noise_anchored_random" and "random".
theta_start	Optional starting zero-inflation probability.
size_start	Optional three-element vector of starting NB2 dispersions.
use_data_priors	Whether to estimate empirical-Bayes regression-prior parameters. With robust = TRUE, a multi-chain pilot constructs one prior that is shared and frozen across

all production chains; without the robust workflow each chain performs its historical warm-up update.

`user_fixed_priors`
: Optional fixed regression priors used when `use_data_priors = FALSE`.

`gamma_prior_shape`
: Positive two-element numeric vector containing the shape parameters of the Beta prior on gamma. The default `c(1, 1)` is uniform on `(0, 1)`.

`epsilon`
: Optional fixed positive ABC bandwidth. `NULL` calibrates one shared bandwidth for all chains of a dataset.

`abc_potts_sweeps`
: Non-negative number of Gibbs sweeps used to equilibrate each simulated auxiliary Potts field. Zero selects the lattice-dependent default, `max(100, 100 * N^2 / 400)`, which keeps the auxiliary fields equally well equilibrated as `N` grows. That default makes the ABC step scale as `N^4` and dominates run time on large lattices: at `N = 100` it is 2500 sweeps. A smaller value is much faster but under-equilibrates the auxiliary fields, which biases the estimated spatial coupling upward.

`abc_sim_reps`
: Positive number of auxiliary fields averaged per ABC proposal. Cost is linear in this value; fewer replicates give a noisier acceptance decision for the spatial coupling.

`distance_metric`
: Retained for source compatibility and ignored.

`relabel`
: Whether to apply the model-based component relabelling rule.

`mc_cores`
: Positive number of chains to execute concurrently.

`verbose`
: Logical. Print sampler progress when `TRUE`.

`progress_interval`
: Positive number of iterations between progress messages when `verbose = TRUE`.

`slope_sd_standardized`
: Standardized slope prior SD used when fixed priors are constructed automatically by the robust workflow.

`abc_epsilon_quantile`
: Quantile in `(0, 1)` used to calibrate the Gaussian ABC bandwidth when `epsilon = NULL`.

`gamma_update_interval`
: Positive number of iterations between ABC gamma updates. Larger values reduce computation but produce fewer gamma moves.

`z_probability_burnin`
: Iteration after which latent-state membership frequencies are accumulated. `NULL` uses `burnin`.

`mcse_stop`
: Whether the sampler may stop early after satisfying the relative Monte Carlo standard-error rule for every monitored parameter.

`mcse_min_iterations`
: Minimum completed iterations before MCSE stopping is considered.

`mcse_check_interval`
: Positive number of iterations between MCSE checks.

`mcse_relative_threshold`
: Maximum batch-means MCSE divided by posterior standard deviation required for early stopping.

`diagnostic_control`
: Named list overriding robust diagnostic defaults. Recognised entries are `minimum_component_cells`, `minimum_ess`, `maximum_rhat`, `gamma_boundary_tolerance`, `maximum_gamma_boundary_fraction`, `minimum_gamma_unique`, `mode_agreement_threshold` and `minimum_mode_chains`. These settings affect reliability reporting and mode selection, not the sampler target.

`gamma_prior`
: Deprecated compatibility alias for `gamma_start`. New code should use `gamma_start`.

Details

Initialisation, progress reporting, MCSE stopping, gamma-update frequency, ABC accuracy/workload and diagnostic thresholds are public controls. The warm-up tempering schedule, component-defining relationship prior, component-2/3 barrier and reversible allocation-move settings remain fixed to preserve the package's component definitions and default sampler.

`abc_potts_sweeps`, `abc_sim_reps` and `gamma_update_interval` are the main ABC cost controls. Their defaults reproduce the package workflow; reducing sweeps or replicates trades auxiliary-simulation fidelity for speed, while increasing the update interval reduces the number of gamma transitions.

Value

For one matrix, a `hicpotts_robust_fit` when `robust = TRUE` or a list of fitted chains otherwise. For a list of matrices, a named list with one such result per dataset. For backward compatibility, a non-robust fit of a list with `n_chains = 1` is returned as one flat fit per dataset.

Examples

```
N <- 5
y <- matrix(rpois(N * N, 4), N, N)
mk <- function() matrix(runif(N * N), N, N)
x_vars <- list(distance = mk(), GC = mk(), TES = mk(), ACC = mk())
fits <- run_chain_betas(
    N = N, x_vars = x_vars, y = y, dist = "Poisson",
    iterations = 5, n_chains = 2, robust = FALSE
)
length(fits)
```

---

summarise_hicpotts_parameters

Report HiCPotts parameter summaries and diagnostic criteria

Description

Extracts posterior estimates, intervals and Monte Carlo diagnostics. The reporting criteria check comparable posterior targets, coefficient movement, component alignment, coefficient scales, covariate conditioning and parameter-specific ESS and R-hat. Other recorded criteria provide context for interpreting the fit. Classification is available separately through `classify_hicpotts()`.

Usage

```
summarise_hicpotts_parameters(
  fit,
  require_reliable = TRUE,
  x_vars = NULL,
  diagnose_covariates = TRUE,
  covariate_diagnostic_args = list(),
  pool_blocks = c("none", "posterior_weighted"),
  pooling_draws = 100000L,
  pooling_seed = 1L,
  pooling_min_expected_cells = 0,
  ...
)
```

Arguments

`fit`	A HiCPotts fit, list of fits, `hicpotts_robust_fit`, block-aware result from `combine_hicpotts_blocks()`, or an object returned by `diagnose_hicpotts_fit()`.
`require_reliable`	Logical. With the default `TRUE`, require a common frozen regression prior, beta movement, aligned components, compatible coefficient scales, covariate conditioning and the requested parameter-specific ESS and R-hat thresholds. Gamma boundary fraction, occupancy and recommended chain count are recorded separately in `reliability_flags`. With `FALSE`, return the table together with a `resolved` column indicating each row's ESS/R-hat result.
`x_vars`	Optional named covariate list. When supplied and `diagnose_covariates = TRUE`, covariate conditioning is checked inside this summary call.
`diagnose_covariates`	Logical. Run the internal covariate diagnostic when `x_vars` is supplied; otherwise reuse diagnostics stored in a robust fit when available.
`covariate_diagnostic_args`	Optional named list of threshold arguments passed to the internal covariate diagnostic.
`pool_blocks`	For a block-aware fit, `"none"` retains one row per block and parameter. `"posterior_weighted"` returns an overall descriptive posterior summary. Component-specific coefficients and dispersions are weighted by posterior expected component occupancy; `gamma` and `theta` are weighted by analysed cells. This does not turn independently fitted blocks into a joint shared-parameter model.
`pooling_draws`	Number of independent posterior-draw combinations used to form empirical credible intervals for a pooled block summary.
`pooling_seed`	Integer seed for reproducible posterior-draw pooling.
`pooling_min_expected_cells`	Non-negative minimum posterior expected component occupancy required for a block to contribute to a component-specific pooled parameter. A zero-occupancy component therefore contributes zero weight without invalidating the fitted block.
`...`	Arguments passed to `diagnose_hicpotts_fit()` when `fit` is a raw fit or fit list.

Value

The parameter data frame with diagnostic metadata attached. For a block fit with `pool_blocks = "posterior_weighted"`, the returned `hicpotts_pooled_parameter_summary` contains overall rows and retains the per-block table in the `"block_summaries"` attribute.

See Also

`diagnose_hicpotts_fit`, `classify_hicpotts`

Examples

```
diagnostic <- list(
    parameters = data.frame(parameter = "gamma", estimate = 0.30),
    reliability_flags = data.frame(
        criterion = "example", passed = TRUE, threshold = "illustrative"
    ),
    classification_entropy = NA_real_,
    warnings = character()
)
summarise_hicpotts_parameters(diagnostic)
```

---

summarise_hicpotts_probabilities

Summarise HiCPotts posterior probabilities

Description

Summarises the data frame returned by `compute_HMRFHiC_probabilities()`. For each component, this function reports the mean probability, median probability, quantile-based interval across interactions, and optional hard-assignment counts based on the maximum posterior probability.

This replaces the older chain-based posterior summary table when the main downstream object is the probability output from `compute_HMRFHiC_probabilities()`.

Usage

```
summarise_hicpotts_probabilities(prob_result, ci_level = 0.95,
component_names = c("noise", "signal", "false signal"),
include_hard_calls = TRUE, include_interaction_summary = TRUE)
```

Arguments

`prob_result`	A data.frame returned by `compute_HMRFHiC_probabilities()`.
`ci_level`	Numeric confidence level in `(0, 1)` used for the quantile interval across interactions.
`component_names`	Character vector of length 3 naming the components.
`include_hard_calls`	Logical; if `TRUE`, include maximum-probability assignment counts and proportions.
`include_interaction_summary`	Logical; if `TRUE` and the `interactions` column is present, include mean and median interaction counts among rows hard-assigned to each component.

Value

A data.frame with one row per component.

Examples

```
prob_res <- data.frame(
    start = c(1e6, 2e6, 3e6),
    end = c(2e6, 3e6, 4e6),
    interactions = c(0, 5, 12),
    prob1 = c(0.80, 0.20, 0.10),
    prob2 = c(0.15, 0.60, 0.25),
    prob3 = c(0.05, 0.20, 0.65)
)

summarise_hicpotts_probabilities(
    prob_result = prob_res, ci_level = 0.95,
    component_names = c("noise", "signal", "false signal"),
    include_hard_calls = TRUE, include_interaction_summary = TRUE
)
```

Index

allocation_diagnostics, 3

classify_hicpotts, *4*, 5, *7, 8*, *28*
combine_hicpotts_blocks, *6*, 7, *11*
compute_HMRFHiC_probabilities, *6–8*, 8

diagnose_hicpotts_fit, 12, *28*
dnbinom, *11*
dpois, *11*

get_data, 14

h5read, *15*
HiCPotts *(*HiCPotts-package*)*, 3
HiCPotts-package, 3

liftOver, *15*

plot_hicpotts_mcmc_by_component, 16
plot_upper_prob_lower_count, 17
posterior_predictive_hicpotts, 19
process_data, *7, 8*, 20

relabel_hicpotts, *6*, *11*, 22
run_chain_betas, *21*, 23

summarise_hicpotts_parameters, *8*, *13*, 26
summarise_hicpotts_probabilities, 28