Package 'HiCPotts'

January 20, 2026

Type Package

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

Version 1.0.0

Description The HiCPotts package provides a comprehensive Bayesian framework for analyzing Hi-C interaction data, integrating both spatial and genomic biases within a probabilistic modeling framework. At its core, HiCPotts leverages the Potts model (Wu, 1982)—a well-established graphical model—to capture and quantify spatial dependencies across interaction loci arranged on a genomic lattice. By treating each interaction as a spatially correlated random variable, the Potts model enables robust segmentation of the genomic landscape into meaningful components, such as noise, true signals, and false signals. To model the influence of various genomic biases, HiCPotts employs a regression-based approach incorporating multiple covariates: Genomic distance (D): The distance between interacting loci, recognized as a fundamental driver of contact frequency. GC-content (GC): The local GC composition around the interacting loci, which can influence chromatin structure and interaction patterns. Transposable elements (TEs): The presence and abundance of repetitive elements that may shape contact probability through chromatin organization. Accessibility score (Acc): A measure of chromatin openness, informing how accessible certain genomic regions are to interaction. By embedding these covariates into a hierarchical mixture model, HiCPotts characterizes each interaction's probability of belonging to one of several latent components. The model parameters, including regression coefficients, zero-inflation parameters (for ZIP/ZINB distributions), and dispersion terms (for NB/ZINB distributions), are inferred via a MCMC sampler. This algorithm draws samples from the joint posterior distribution, allowing for flexible posterior inference on model parameters and hidden states. From these posterior samples, HiCPotts computes posterior means of regression parameters and other quantities of interest. These posterior estimates are then used to calculate the posterior probabilities that assign each interaction to a specific component. The resulting classification sheds light on the underlying structure: distinguishing genuine high-confidence interactions (signal) from background noise and potential false signals, while simultaneously quantifying the impact of genomic biases on observed interaction frequencies. In summary, HiCPotts seamlessly integrates spatial modeling, bias correction, and probabilistic classification into a unified Bayesian inference framework. It provides rich posterior summaries and interpretable, model-based assignments of interaction states, enabling researchers to better understand the interplay between genomic organization, biases, and spatial correlation in Hi-C data.

License GPL-3

Encoding UTF-8

Imports Rcpp(>= 0.11.0), parallel, stats, Biostrings, GenomicRanges, rtracklayer, strawr, rhdf5, BSgenome,IRanges, S4Vectors, BSgenome.Dmelanogaster.UCSC.dm6

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.3.2

NeedsCompilation yes

Suggests BiocStyle, knitr (>= 1.30), rmarkdown (>= 2.10), ggplot2 (>= 3.5.0), reshape2 (>= 1.4.4), testthat (>= 3.0.0), BiocManager

Config/testthat/edition 3

Depends R (>= 4.5)

LazyData false

VignetteBuilder knitr

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

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

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

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

git_branch RELEASE_3_22

git_last_commit d5f20b8

git_last_commit_date 2025-10-29

Repository Bioconductor 3.22

Date/Publication 2026-01-19

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

`compute_HMRFHiC_probabilities` *Compute HMRFHiC Probabilities of Assigning an Interaction to Each Component*

Description

This function computes the posterior probabilities of assigning genomic interactions to each of three mixture components (or states) in a Hidden Markov Random Field (HMRF) model. It uses the posterior means of regression parameters obtained from MCMC simulations and combines these with user-specified distributions (zero-inflated or standard) to produce probabilities for each observed interaction.

Usage

```
compute_HMRFHiC_probabilities(data, chain_betas, iterations, dist = "ZINB")
```

Arguments

`data` A `data.frame` containing the following required columns:

- `start`: Genomic start position for locus i.
- `end`: Genomic end position for locus j.
- `interactions`: Observed interaction counts between loci i and j.
- `GC`: GC content or related genomic feature for the interaction.
- `TES`: Transposable Elements data or similar regulatory measure.
- `ACC`: Accessibility measure or another continuous genomic covariate.

The `start` and `end` columns must correspond to the indices of interacting loci (i and j).

`chain_betas` A list of MCMC chain results generated by the HMRFHiC model-fitting procedure. Each element should correspond to one chain and must include:

- `chains`: A list of 3 MCMC chains (one for each component), each containing posterior samples of regression coefficients.
- `theta`: (If applicable) Posterior samples of the zero-inflation parameter θ.
- `size`: (If applicable) Posterior samples for the overdispersion parameter used in Negative Binomial or Zero-Inflated Negative Binomial models.

`iterations` An integer specifying the number of total MCMC iterations. The function uses half of these as burn-in (i.e., `iterations/2`).

`dist` A character string indicating the chosen distribution for modeling interaction counts. One of:

- `"ZIP"`: Zero-Inflated Poisson
- `"NB"`: Negative Binomial
- `"Poisson"`: Poisson
- `"ZINB"`: Zero-Inflated Negative Binomial

The default is `"ZINB"`.

Details

This function is part of the HMRFHiC pipeline that models genomic interactions (e.g., Hi-C interaction counts) using a mixture model approach. The model typically considers three components (or latent states), each characterized by a distinct mean-structure and potentially different zero-inflation or overdispersion parameters, depending on the chosen distribution.

The function:

1. Extracts posterior means of regression parameters from MCMC chains, discarding the initial half of the samples as burn-in.
2. Estimates mean interaction intensities (λ) for each component using log-linear models with covariates: distance (log of |end-start|),GC, TES, and ACC (each transformed by a $\log(x+1)$ operation).
3. Given the specified distribution (`dist`), calculates the probability (on the natural scale) of observing the recorded interaction count for each component.
4. Normalizes these probabilities so that each interaction is assigned a set of three probabilities summing to 1.

For zero-inflated distributions (`"ZIP"`, `"ZINB"`), a θ parameter captures the probability of an excess zero. For negative binomial distributions (`"NB"`, `"ZINB"`), an overdispersion parameter is included.

The computed probabilities can be used for downstream analysis, such as segmenting interactions into classes or modeling spatial dependence in a hidden Markov field.

Value

A `data.frame` containing the original input columns plus three new columns:

- `prob1`: The posterior probability that the interaction belongs to component 1.
- `prob2`: The posterior probability that the interaction belongs to component 2.
- `prob3`: The posterior probability that the interaction belongs to component 3.

The returned data frame thus provides a probabilistic classification of each observed interaction into one of the three modeled components.

See Also

`dpois`, `dnbinom`, for probability calculations.

Examples

```
#
#
# Synthetic data
set.seed(123)


large_data <- data.frame(
  start = c(1, 10, 20),
  end = c(5, 15, 30),
  interactions = c(10, 20, 30),
  GC = c(0.5, 0.8, 0.3),
  TES = c(0.2, 0.5, 0.7),
  ACC = c(0.9, 0.4, 0.6)
)
```

```
chain_betas <- list(
  list(
    chains = list(
      matrix(runif(25, 0.1, 1), ncol = 5),
      matrix(runif(25, 0.1, 1), ncol = 5),
      matrix(runif(25, 0.1, 1), ncol = 5)
    ),
    theta = runif(5, 0.1, 0.9),
    size = matrix(runif(15, 1, 10), nrow = 3)
  )
)

result <- compute_HMRFHiC_probabilities(
  data = large_data,
  chain_betas = chain_betas,
  iterations = 5,
  dist = "Poisson"
)
print(result)
# See vignette("HMRFHiC_vignette") for detailed examples with real Hi-C data.
#
#
```

`gamma_prior_value` *Prior Value for the Potts Model Interaction Parameter*

Description

This function generates a prior value for the interaction parameter in a Potts model from a Beta distribution. The Potts model is a spatial statistical model where the interaction parameter influences the tendency of neighboring sites on a lattice to take on similar states. By drawing the interaction parameter from a Beta distribution, we impose a prior belief on the range and likely values of this parameter.

Usage

```
gamma_prior_value()
```

Details

The function samples a single random value from a `Beta(10, 5)` distribution. This distribution places more mass towards the higher end of the \[0, 1\] interval (since Beta(10, 5) is skewed towards 1), indicating a prior belief that the interaction parameter is likely to encourage some degree of spatial clustering.

Value

A numeric value between \[0,1\] representing the prior draw for the interaction parameter. This is a random draw from the specified Beta distribution.

Examples

```
#
# Generate a single prior value for the Potts model interaction parameter
prior_val <- gamma_prior_value()
#prior_val
```

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

Description

Reads a Hi-C contact matrix from a `.hic`, `.cool`, or `.h5` 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, acc_wig = NULL,
  chain_file = NULL, te_granges = NULL
)
```

Arguments

`file_path`	Path to the Hi-C file (`.hic`, `.cool`, or `.h5`).
`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`	Name of a BSgenome package (e.g. `"BSgenome.Dmelanogaster.UCSC.dm6"`) for GC calculations.
`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

`straw`, `h5read`, `liftOver`

Examples

```
wig <- system.file("extdata", "DNaseI_BG3_gr_chr4.bedGraph", package = "HiCPotts")
chain <- system.file("extdata", "dm3ToDm6_chr4_only.chain", package = "HiCPotts")
te <- system.file("extdata", "dm6_TEs_chr4.gtf", package = "HiCPotts")
hic <- system.file("extdata", "BG3_WT_merged_hic_matrix_chr4_100Kb.cool", package = "HiCPotts")

bb <- get_data(
  file_path      = hic,
  chr            = "chr4",
  start          = 1,
  end            = 100000,
  resolution     = 10000,
  genome_package = "BSgenome.Dmelanogaster.UCSC.dm6",
  acc_wig        = wig,
  chain_file     = chain,
  te_granges     = te
)
```

---

likelihood_combined *Likelihood Function for a Specified Model Component and Distribution*

Description

This function computes the likelihood contribution of a specific model component using a chosen distribution. It is designed to work within a mixture-modeling or hierarchical modeling framework, where each interaction (such as genomic interactions in a Hi-C experiment) can be modeled by a combination of regression parameters and potentially zero-inflation or overdispersion parameters.

Usage

```
likelihood_combined(pred_combined, params, z, y, x_vars, component, theta, size, N, dist)
```

Arguments

`pred_combined`	A numeric vector of predictor values obtained from a prediction function. This typically represents the linear predictor λ for the given component.
`params`	A numeric vector of parameters associated with the model's linear predictors. Typically includes intercepts and regression coefficients related to the covariates in `x_vars`.
`z`	A matrix or array representing the latent state indicators or other structural variables that influence the model. `z` helps map each observation to a particular mixture component.
`y`	A numeric vector of observed interaction counts (the response variable). Each element corresponds to one observation (e.g., interaction count between a pair of genomic loci).
`x_vars`	A list of covariates used as predictors in the linear model. Each element in the list corresponds to a covariate vector, and all must be of length `N`, the number of observations.
`component`	An integer or factor specifying which component of the mixture model is currently being evaluated. In the 3-component mixture, this is `1`, `2`, or `3`.
`theta`	(Optional) A numeric value for the zero-inflation parameter. Required for ZIP and ZINB models. This parameter controls the probability of excess zeros not explained by the Poisson or NB component.
`size`	(Optional) A numeric value for the size (overdispersion) parameter. Required for NB and ZINB models. This parameter captures variance that exceeds that of a Poisson distribution.
`N`	An integer specifying the number of observations. This should match the length of the response variable `y` and the covariates in `x_vars`.
`dist`	A character string specifying the distribution to be used for modeling the interaction counts. Options may include: • `"Poisson"`: Poisson distribution • `"NB"`: Negative Binomial distribution • `"ZIP"`: Zero-Inflated Poisson distribution • `"ZINB"`: Zero-Inflated Negative Binomial distribution

Details

This function calculates the likelihood for a single component given a set of parameters and covariates. The steps typically involved are:

1. Compute the linear predictor λ from the supplied parameters and covariates. The `pred_combined` represents $\log(\lambda)$.
2. Depending on `dist`, convert the linear predictor into a mean parameter (e.g., λ for Poisson or NB).
3. Compute the likelihood of each observed count `y[i]` under the chosen distribution with the given parameters (λ, θ, $size$, etc.).
4. For zero-inflated models (ZIP, ZINB), the likelihood incorporates the probability of an extra zero.

5. Return the computed likelihood values. This is used internally to evaluate and update model parameters during the MCMC steps.

The function is a building block in a larger modeling framework (MCMC inference for mixture models). While end-users might not call it directly, it enables flexible specification of distributions and model components for complex hierarchical models.

Value

The function returns the computed likelihood under the specified model setup. It returns a numeric vector of likelihood contributions for each observation.

See Also

dpois, dnbinom for related probability mass functions.

Examples

```
# Example setup
N <- 3
z <- matrix(c(1, 1, 2, 3, 1, 2, 3, 1, 2), nrow = 3, byrow = TRUE)
y <- matrix(c(0, 3, 5, 1, 0, 4, 6, 2, 0), nrow = 3, byrow = TRUE)
params <- c(0.1, 0.2, 0.3, 0.4, 0.5)
x_vars <- x_vars <- list(
  list(matrix(runif(9, 1, 10), nrow = N)), # first covariate
  list(matrix(runif(9, 1, 10), nrow = N)), # second covariate
  list(matrix(runif(9, 1, 10), nrow = N)), # third covariate
  list(matrix(runif(9, 1, 10), nrow = N)) # fourth covariate
)

theta <- 0.2
size <- 10
preds_comp1 <- pred_combined(params, z, x_vars, component = 1, N = N)
# Compute likelihood under a Poisson model, component 1
ll_values <- likelihood_combined(
  pred_combined = preds_comp1,
  params = params,
  z = z,
  y = y,
  x_vars = x_vars,
  component = 1,
  theta,
  size,
  N = N,
  dist = "Poisson"
)

#print(sum(ll_values)) # sum of likelihood contributions
```

`likelihood_gamma` *Likelihood Function for the Interaction Parameter in a Potts Model*

Description

This function computes the matrix of probabilities (or likelihood values) associated with the interaction parameter in the Potts model. The Potts model is a generalization of the Ising model and is commonly used in spatial statistics, image analysis, and other fields where data can be represented on a lattice. The interaction parameter controls how neighboring sites on the lattice influence each other.

Usage

```
likelihood_gamma(x, pair_neighbours_DA_x1, N)
```

Arguments

`x`	A numeric vector representing the interaction parameter(s) for the Potts model. This could be a single parameter repeated for each pair of neighbors.
`pair_neighbours_DA_x1`	A numeric vector of the same length as `x`, representing the pairwise interaction structure between neighbors in the Potts model. Each element indicates whether a pair of sites is assigned as neighbors.
`N`	An integer specifying the dimension of the Potts lattice. The returned matrix will be `N` by `N`, representing the spatial layout of the Potts model.

Details

The Potts model describes configurations of states on a lattice, where the interaction parameter `x` influences how often neighboring sites are in the same state. The likelihood or probability of a configuration is determined by exponentials of the interaction terms between neighboring sites.

This function:

1. Multiplies the interaction parameter vector `x` by the vector `pair_neighbours_DA_x1`, which encodes neighbor relationships or their contributions.
2. Applies a stability adjustment by subtracting the maximum value (`max_val`) from each term $x * pair\_neighbours\_DA\_x1$ to avoid numerical overflow.
3. Exponentiates these adjusted values and normalizes them to obtain a set of values that sum to 1, interpreting them as probabilities or likelihood contributions.
4. Reshapes these normalized values into an `N` x `N` matrix, `potts_DA`, which represents the spatial layout of likelihoods (or probabilities) under the Potts model.

Value

A numeric `N` x `N` matrix of normalized probabilities corresponding to the Potts model likelihood for the given interaction parameter. Each element of the matrix represents the likelihood contribution of that site/pair under the model parameterized by `x`.

Examples

```
# Suppose we have an N=4 lattice and a vector x and pair_neighbours_DA_x1 of length 16
N <- 4
x <- rep(0.5, N * N) # interaction parameter repeated for each pair
pair_neighbours_DA_x1 <- rnorm(N * N) # random neighbor influences

# Compute the Potts DA matrix
potts_matrix <- likelihood_gamma(x, pair_neighbours_DA_x1, N)
#print(potts_matrix)
```

Neighbours_combined *Neighbours Function for the Potts Model*

Description

Computes the number of agreeing neighbors for each site in a Potts model lattice. The Potts model is a generalization of the Ising model, where each site on a lattice can take on multiple states. By counting how many of a site's four neighbors share the same state, this function provides a measure of clustering within the lattice.

Usage

```
Neighbours_combined(potts_data, N, proposed_value = NULL)
```

Arguments

potts_data	A numeric $N \times N$ matrix representing the current configuration of the Potts model. Each element corresponds to the state of a particular site.
N	An integer specifying the dimension of the potts_data lattice (i.e., the lattice has N rows and N columns).
proposed_value	(Optional) A numeric $N \times N$ matrix representing a proposed configuration of the Potts model. If provided, the function computes neighbor relationships relative to this proposed configuration; otherwise, it uses a shifted version of potts_data to determine neighbors.

Details

The function checks each site and compares it with its four directional neighbors (up, down, left, right). For each neighbor that has the same state, a value of 1 is assigned. Summing these values for each site results in a measure of how "similar" the immediate neighborhood is to that site.

Internally, the function:

1. Uses shift_matrix to create versions of the lattice shifted in each of the four directions.
2. Uses compute_neighbours to determine if neighboring positions match in state.
3. Sums the contributions from all four directions to form the Neighbours_total matrix.

By providing an optional proposed_value matrix, this function can be integrated into an MCMC algorithm where new configurations are proposed and evaluated based on their local coherence.

Value

A numeric $N \times N$ matrix, `Neighbours_total`, where each element is the count of how many of the four neighbors match the state of that site.

Examples

```
# Suppose we have a 5x5 Potts model lattice:
potts_data <- matrix(sample(1:3, 25, replace = TRUE), ncol = 5)
N <- 5

# Compute the neighbors without a proposed configuration:
neigh <- Neighbours_combined(potts_data, N)
neigh

# Suppose we propose a new configuration:
proposed <- matrix(sample(1:3, 25, replace = TRUE), ncol = 5)

# Evaluate neighbors for the proposed configuration:
neigh_proposed <- Neighbours_combined(potts_data, N, proposed_value = proposed)
neigh_proposed
```

---

posterior_combined *Compute the Posterior for the Model Components*

Description

This function computes the combined posterior value for a specified component in the model utilizing a variety of distributions (Poisson, Negative Binomial, Zero-Inflated Poisson, Zero-Inflated Negative Binomial). It incorporates both the likelihood of the observed data under the given model parameters and the prior distributions for those parameters. When applicable, it also includes a prior on the dispersion parameter (size) used in Negative Binomial or Zero-Inflated Negative Binomial models.

Usage

```
posterior_combined(pred_combined, params, z, y, x_vars, component, theta, N,
                   use_data_priors, user_fixed_priors, dist, size)
```

Arguments

`pred_combined`	A numeric vector of predictor values generated from a prediction function. Typically represents the linear predictors on a log scale.
`params`	A numeric vector of parameters for the model. This includes the regression coefficients.
`z`	A matrix or data structure representing latent variables, components, or states in the model. For example, in the mixture model, `z` it indicates component membership of observations.
`y`	A numeric vector of observed counts or interaction values. Each element corresponds to an observation (e.g., interaction counts between genomic loci).

`x_vars` A list of covariates used in the model. Each element corresponds to a predictor variable, and all should be of length `N`, the number of observations.

`component` An integer or factor indicating the component (e.g., mixture component) for which the posterior is being computed. Different components might have different parameter sets or priors.

`theta` (Optional) A numeric value representing the zero-inflation parameter for Zero-Inflated models. θ governs the probability of extra zeros beyond what the Poisson or Negative Binomial distribution would predict.

`N` An integer specifying the number of observations. This should match the length of `y`.

`use_data_priors` A logical value indicating whether to use data-driven priors for each component. If `TRUE`, the prior distributions may be estimated from the data.

`user_fixed_priors` A logical value indicating whether to use user-specified priors for each component. If `TRUE`, a user-supplied prior distribution is applied instead of a data-driven one.

`dist` A character string specifying the distribution family. One of:

- `"Poisson"`: Poisson distribution
- `"NB"`: Negative Binomial distribution
- `"ZIP"`: Zero-Inflated Poisson distribution
- `"ZINB"`: Zero-Inflated Negative Binomial distribution

`size` (Optional) A numeric value for the dispersion parameter in the NB or ZINB distribution. This parameter models overdispersion, allowing the variance to exceed the mean.

Details

The posterior is computed as:

$$\text{Posterior} = \text{Likelihood} + \text{Prior}(+\text{Size Prior if NB or ZINB})$$

Steps involved:

1. Compute the likelihood of the data (`y`) given the parameters, latent structure (`z`), covariates (`x_vars`), and chosen distribution (`dist`) using the provided `pred_combined` values. This is done via the `likelihood_combined` function.
2. Compute the prior distribution over the parameters (`params`). Depending on `use_data_priors` and `user_fixed_priors`, these priors might be estimated from data or supplied by the user. This step uses the `prior_combined` function.
3. If the distribution is `"NB"` or `"ZINB"`, incorporate a separate prior on the dispersion parameter (`size`). This step uses the `size_prior` function.

By combining the likelihood and priors, the function returns the posterior value, which is used in the Bayesian inference and MCMC steps to update parameters and perform parameters estimations.

Value

A numeric value representing the posterior log-probability for the given component under the specified distribution and modeling assumptions.

Examples

```
# Example usage of posterior:
N <- 5
y <- rpois(N, lambda = 5)
x_vars <- list(
  list(matrix(runif(25), nrow = 5)),
  list(matrix(runif(25), nrow = 5)),
  list(matrix(runif(25), nrow = 5)),
  list(matrix(runif(25), nrow = 5))
)
params <- c(1, 2, 3, 4, 5)
z <- matrix(sample(1:3, N * 5, replace = TRUE), nrow = N, ncol = 5)
preds_comp1 <- pred_combined(params, z, x_vars, component = 1, N = N)

# Compute posterior for component 1 using a Poisson distribution
post_val <- posterior_combined(
  pred_combined = preds_comp1,
  params = params,
  z = z,
  y = y,
  x_vars = x_vars,
  component = 1,
  theta = NULL,
  N = N,
  use_data_priors = TRUE,
  user_fixed_priors = NULL,
  dist = "Poisson",
  size = NULL
)
#print(post_val) # Display result
```

---

pred_combined *Prediction Function for Combined Model*

Description

This function computes the predicted linear predictor (λ) values for a specified model component. It is designed to correct for various sources of bias by incorporating multiple covariates through a log-transformed relationship. The resulting predictions serves as inputs to downstream likelihood or posterior calculations in the Bayesian framework.

Usage

```
pred_combined(params, z, x_vars, component, N)
```

Arguments

params	A numeric vector of parameters for the model. Typically, params = (a, b, c, d, e), where a is the intercept and b, c, d, e are coefficients for the log-transformed covariates.

`z` A matrix used to indicate component membership of each observation. The function uses `z == component` to select the subset of observations belonging to the specified component.

`x_vars` A list of covariates, each element itself a list or vector of length `N`, corresponding to different sources of bias or predictors. For example:

- `x_vars[[1]]`: The first covariate (wrapped in a list), accessible via `x_vars[[1]][[1]]`.
- `x_vars[[2]]`, `x_vars[[3]]`, `x_vars[[4]]`: Additional covariates in similar nested-list structures.

Each covariate vector must be of length `N`, and the values are expected to be non-negative since a $\log(x + 1)$ transformation is applied.

`component` An integer specifying which component of the mixture or hierarchical model to use when subsetting the data. For the three components, `component = 1` will extract the subset of data where `z == 1`.

`N` An integer specifying the total number of observations. This should match the length of the `y` vector and each covariate vector.

Details

The predicted values (λ) are computed as:

$$\lambda_i = a + b \cdot \log(x1_i + 1) + c \cdot \log(x2_i + 1) + d \cdot \log(x3_i + 1) + e \cdot \log(x4_i + 1)$$

for each observation i in the specified component. This log-transform often stabilizes variance and normalizes skewed distributions of covariates.

The resulting λ values serves as linear predictors (in a Poisson, NB, ZIP, or ZINB model) and as inputs into the likelihood and posterior computation steps.

Value

A numeric vector containing the predicted values (λ) for observations belonging to the specified component.

Examples

```
#\donttest{
# Assume we have:
# N = 3 observations
# z indicates component membership for each observation
# x_vars is a list of four covariates, each nested in a list: x_vars[[i]][[1]]
N <- 3
params <- c(1, 2, 3, 4, 5) # Example parameter values
z <- matrix(c(1, 1, 2, 3, 1, 3, 2, 3, 1), nrow = 3) # Example component matrix
x_vars <- list(
  list(matrix(runif(9, 1, 10), nrow = N)), # first covariate
  list(matrix(runif(9, 1, 10), nrow = N)), # second covariate
  list(matrix(runif(9, 1, 10), nrow = N)), # third covariate
  list(matrix(runif(9, 1, 10), nrow = N)) # fourth covariate
)


# Get predicted values for component 1:
preds_comp1 <- pred_combined(params, z, x_vars, component = 1, N = N)
print(preds_comp1)
```

```
#}
```

`prior_combined` *Prior Function for Model Components*

Description

This function computes the prior contribution to the log-posterior for the parameters associated with a specified model component. It can use either data-driven priors (estimated from the subset of data corresponding to that component) or user-specified fixed priors. By integrating these priors into the Bayesian inference framework, the function helps ensure that parameter estimates remain grounded in reasonable values and incorporate domain knowledge or data-derived distributions.

Usage

```
prior_combined(params, component, y, x_vars, z, use_data_priors = TRUE, user_fixed_priors)
```

Arguments

`params`	A numeric vector of parameters for the model. Typically, `params` will include a set of parameters associated with the regression coefficients for the model. For example, `params` might be (a, b, c, d, e) representing coefficients for the covariates in `x_vars`.
`component`	An integer specifying which component of the model to consider. The model is a mixture with multiple components, this argument indicates which component's parameters and priors are being evaluated.
`y`	A numeric vector of observed responses (e.g., interaction counts). This is used when estimating data-driven priors. The vector should be of length `N`, where `N` is the size of the system.
`x_vars`	A list of covariate information. Each element of `x_vars` is typically another list or vector representing a single covariate. These covariates are used to compute data-driven priors by extracting summary statistics from the data corresponding to the specified component.
`z`	A matrix or similar structure used to assign observations to components. If `z == component`, that observation belongs to the given component. The function uses this to subset `y` and `x_vars` for calculating data-driven priors.
`use_data_priors`	A logical value indicating whether to use data-driven priors (`TRUE`) or user-specified fixed priors (`FALSE`). When `TRUE`, the function computes means and standard deviations from the subset of the data assigned to the specified component.
`user_fixed_priors`	A list of user-defined priors for each component. This argument is only used when `use_data_priors = FALSE`. The list should contain entries named `component1`, `component2`, and `component3`, each defining `meany`, `meanx1`, `meanx2`, `meanx3`, `meanx4`, and corresponding `sdy`, `sdx1`, `sdx2`, `sdx3`, `sdx4`.

Details

This function supports two modes:

1. **Data-Driven Priors:** When `use_data_priors = TRUE`, the function subsets the data for the specified `component`, calculates summary statistics, and then draws from those statistics to form priors for each parameter. This approach allows the priors to adapt to the characteristics of the data associated with that component.
2. **User-Specified Fixed Priors:** When `use_data_priors = FALSE`, the function uses priors provided by the user through `user_fixed_priors`, bypassing any data-driven estimation.

In both modes, the priors are assumed to be normal distributions for each parameter. The function returns the sum of the log of these priors.

Value

A single numeric value representing the sum of the log-priors for all parameters in `params`. This value can be integrated into a Bayesian inference framework to update parameter estimates based on the chosen priors.

Examples

```
# Example:
N <- 100
y <- rpois(N, lambda = 5)
x_vars <- list(
  list(runif(N)), # x1
  list(rnorm(N)), # x2
  list(rnorm(N)), # x3
  list(rnorm(N)) # x4
)
z <- sample(1:3, N, replace = TRUE)
params <- c(a = 1, b = 0.5, c = -0.2, d = 2, e = 0.3)

# Using data-driven priors for component 1
prior_val_data <- prior_combined(
  params = params,
  component = 1,
  y = y,
  x_vars = x_vars,
  z = z,
  use_data_priors = TRUE,
  user_fixed_priors = NULL
)

# Using user-specified fixed priors for component 2
#user_fixed_priors <- list(
#  component1 = list(
#    meany = 5, meanx1 = 0, meanx2 = 0, meanx3 = 0, meanx4 = 0,
#    sdy = 1, sdx1 = 1, sdx2 = 1, sdx3 = 1, sdx4 = 1
#  ),
#  component2 = list(
#    meany = 10, meanx1 = 1, meanx2 = 1, meanx3 = 1, meanx4 = 1,
#    sdy = 2, sdx1 = 2, sdx2 = 2, sdx3 = 2, sdx4 = 2
#  ),
#  component3 = list(
#    meany = 2, meanx1 = 2, meanx2 = 2, meanx3 = 2, meanx4 = 2,
```

```
#    sdy = 1, sdx1 = 1, sdx2 = 1, sdx3 = 1, sdx4 = 1
#  )
#)
#
#prior_val_fixed <- prior_combined(
#  params = params,
#  component = 2,
#  y = y,
#  x_vars = x_vars,
#  z = z,
#  use_data_priors = FALSE,
#  user_fixed_priors = user_fixed_priors
#)
```

---

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 = 500, standardization_y = TRUE)
```

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. Defaults to 500. After scaling, interaction counts range between [1, scale_max].
standardization_y	A logical value. If TRUE, interaction counts are scaled to the range [1, scale_max] by applying a min-max normalization and rounding. If FALSE, the raw interaction counts are used as provided.

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 genomic distances as $|end - start|$.
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 = 500, standardization_y = TRUE)
#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 = 500, standardization_y = TRUE)
#x_vars <- processed[[1]]
#y_matrices <- processed[[2]]
#str(x_vars)
#str(y_matrices)
# See vignette("HMRFHiC_vignette") for detailed examples with real Hi-C data.
#
```

`proposaldensity_combined` *Proposal Density Function for Model Components*

Description

This function computes the proposal density under a given proposal distribution for a specified model component. In a Bayesian MCMC framework, a proposal distribution is used to generate candidate parameter values in each iteration. By evaluating the log-density of the proposed parameters under this distribution, the MCMC algorithm can determine the acceptance or rejection of the new draw.

Usage

```
proposaldensity_combined(params, component)
```

Arguments

`params`	A numeric vector of parameters for which the proposal density is evaluated. The length and meaning of these parameters are determined by the model and the selected component.
`component`	An integer or identifier specifying which component's proposal distribution should be used. Each component has its own set of means and standard deviations defining the proposal distribution. Valid components should match those defined within the function (e.g., 1, 2, or 3).

Details

The function defines three components, each with its own proposal distribution parameters (means and standard deviations) for a set of parameters. It then evaluates the log-density of the `params` vector under a multivariate normal distribution (assuming independence between parameters), using the component-specific means and standard deviations.

Steps performed by the function:

1. Verifies that the specified `component` is valid and corresponds to one of the defined distributions.
2. Extracts the mean and standard deviation vectors associated with that component.
3. Checks that the length of `params` matches the expected length of the means and standard deviations.
4. Evaluates the log of the normal probability density function (PDF) for each parameter using the corresponding mean and standard deviation.
5. Sums these log-densities to produce the overall log-density of the proposal distribution for the parameter vector.

This function is used internally in our MCMC algorithms where proposals are drawn from component-specific Normal distributions. By returning a log-probability, it simplifies calculations involved in Metropolis-Hastings updates and related MCMC procedures.

Value

A single numeric value representing the sum of the log proposal densities of all parameters for the specified component.

Examples

```
# Example usage:
# Suppose we have a parameter vector and want to compute its proposal density under component 2:
params <- c(0, 3, 5, 6, 1) # parameter vector
component <- 2
log_proposal <- proposaldensity_combined(params, component)
#print(log_proposal)
```

`proposalfunction` *Proposal Function for the Potts Model Interaction Parameter*

Description

This function generates a proposal value for the interaction parameter in the Potts model, drawn from a specified Beta distribution. It is typically used within the Markov Chain Monte Carlo (MCMC) framework to propose new candidate values for the interaction parameter at each iteration.

Usage

```
proposalfunction()
```

Details

The Potts model describes configurations of states on a lattice, and the interaction parameter influences how neighboring sites align. An MCMC algorithm typically requires a proposal distribution to generate new candidate values for parameters. Here, the function draws from a `Beta(10,5)` distribution, which, similar to the prior example, draws the proposals towards the higher end of the \[0,1\] interval.

Value

A numeric value between 0 and 1, drawn from a Beta(10, 5) distribution, serving as a proposal value for the interaction parameter in the Potts model.

Examples

```
# Generate a proposal value for the Potts model interaction parameter
proposed_val <- proposalfunction()
#proposed_val
```

---

pz_123 *Posterior Function for the Potts Model*

Description

Computes log-posterior probabilities for each site in an $N \times N$ lattice under the Potts model, combined with an interaction count model (e.g., genomic interactions). Incorporates mixture components, neighbor relationships, and distributional assumptions (Poisson, Negative Binomial, Zero-Inflated Poisson, Zero-Inflated Negative Binomial).

Usage

```
pz_123(z, sum_neighbours, y, pred_combined, chains, chain_gamma,
       x_vars, theta, size_chain, N, iter, dist)
```

Arguments

`z`	A numeric $N \times N$ matrix representing the component assignments for each site on the lattice. Each element is a component label (e.g., 1, 2, or 3), indicating which mixture component that interaction currently belongs to.
`sum_neighbours`	An $N \times N$ numeric matrix representing the number of neighboring sites (up, down, left, right) that agree in state with each site. Typically computed by the function `Neighbours_combined()`.
`y`	An $N \times N$ numeric matrix of observed interaction counts (e.g., genomic interaction data). Each element corresponds to the count observed at that site.
`pred_combined`	An R function (passed from R to C++) that computes predicted values (linear predictors) for the specified component. This is often a reference to the `pred_combined()` function in R.
`chains`	A `list` of parameter vectors (MCMC chains) for the model. Each element corresponds to a component and contains the parameters associated with that component.
`chain_gamma`	A numeric vector (using ABC algorithm) for the interaction parameter γ that governs neighbor effects in the Potts model. The `iter`-th element of this vector is used in the calculation.
`x_vars`	A list of covariates (sources of biases), each a nested structure as used by `pred_combined()`. These covariates are used to compute component-specific predicted values (λ).

theta	A numeric scalar representing the zero-inflation parameter. Used when the distribution is Zero-Inflated Poisson (ZIP) or Zero-Inflated Negative Binomial (ZINB), typically for the first component.
size_chain	An $3 \times M$ matrix representing the MCMC draws of the size (overdispersion) parameter for Negative Binomial or Zero-Inflated Negative Binomial distributions. One row per component, and columns correspond to iterations. The value at `size_chain[comp - 1, iter]` is used for the specified iteration.
N	An integer specifying the dimension of the lattice ($N \times N$).
iter	An integer indicating which iteration of the MCMC chains (parameters, γ, size) to use in the calculation.
dist	A character string specifying the distribution: • `"Poisson"`: Poisson distribution • `"NB"`: Negative Binomial distribution • `"ZIP"`: Zero-Inflated Poisson distribution • `"ZINB"`: Zero-Inflated Negative Binomial distribution

Details

This function interfaces with C++ code to calculate log-posterior probabilities for an $N \times N$ lattice under the Potts model. The model accounts for:

1. Component assignments (`z`) for each site.
2. Observed interaction counts (`y`) from genomic data.
3. Neighbor effects (`sum_neighbours`) weighted by γ (`chain_gamma`).
4. Covariates (`x_vars`) used to compute predicted values (λ) via `pred_combined`.
5. Distributional assumptions (`dist`: Poisson, NB, ZIP, or ZINB), with zero-inflation (`theta`) for ZIP/ZINB (first component only) and overdispersion (`size_chain`) for NB/ZINB.

The `params` list bundles all necessary parameters, which are passed to the C++ function `_HMRFHiC_potts_posterior`. Probabilities are computed for each site, adjusted by neighbor effects using $\exp(\gamma \times$ (`sum_neighbours`)), and returned as log-posteriors.

Value

A numeric $N \times N$ matrix of log-posterior probabilities for each site for the specified component.

Examples

```
#\donttest{
 z <- matrix(sample(1:3, 25, replace = TRUE), nrow = 5)
 sum_neighbours <- matrix(runif(25, 0, 5), nrow = 5)
 y <- matrix(rpois(25, lambda = 5), nrow = 5)
 pred_combined <- function(params, z, x_vars, component, N) {
   return(runif(sum(z == component)))
 }
 chains <- list(
   matrix(rnorm(25), nrow = 5, ncol = 5),
   matrix(rnorm(25), nrow = 5, ncol = 5),
   matrix(rnorm(25), nrow = 5, ncol = 5)
 )
 chain_gamma <- rnorm(5)
 x_vars <- list(
```

```
  distance = list(matrix(runif(25, 0, 1), nrow = 5)),
  GC = list(matrix(runif(25, 0, 1), nrow = 5)),
  TES = list(matrix(runif(25, 0, 1), nrow = 5)),
  ACC = list(matrix(runif(25, 0, 1), nrow = 5))
)
theta <- 0.5
size_chain <- matrix(runif(25, 0.5, 1.5), nrow = 5, ncol = 5)
N <- nrow(z)
iter <- 1
dist <- "ZIP"

# Run the function
pz_vals <- pz_123(
  z = z,
  sum_neighbours = sum_neighbours,
  y = y,
  pred_combined = pred_combined,
  chains = chains,
  chain_gamma = chain_gamma,
  x_vars = x_vars,
  theta = theta,
  size_chain = size_chain,
  N = N,
  iter = iter,
  dist = dist
)

#print(pz_vals)
#}
```

`run_chain_betas` *MCMC Chain Execution for HiCPotts Simulations*

Description

The run_metropolis_MCMC_betas function serves as a high-level wrapper to execute the MCMC sampling process on Hi-C interaction data. It leverages the run_metropolis_MCMC_betas function to perform Bayesian inference, including updating model parameters (betas, gamma, size, and theta) and mixture component assignments. It can also run in parallel. By running multiple simulations simultaneously, it streamlines and expedites large-scale analyses.

Usage

```
run_chain_betas(
  N,
  gamma_prior=0.3,
  iterations,
  x_vars,
  y,
  theta_start=NULL,
  size_start=NULL,
  use_data_priors=TRUE,
```

```
  user_fixed_priors=NULL,
  dist="ZIP",
  epsilon=NULL,
  distance_metric="manhattan",
  mc_cores = 1
)
```

Arguments

N	An integer specifying the dimension of each interaction matrix ($N \times N$). All interaction matrices in y must have this dimension.
gamma_prior	A numeric value specifying the initial (prior) value for the Potts model interaction parameter γ. This parameter influences the spatial dependency structure in the model.
iterations	An integer indicating the number of MCMC iterations to run for each dataset. During these iterations, parameters (betas, gamma, theta, size) and component (z) are updated repeatedly.
x_vars	A list of covariates (e.g., distance, GC, TES, ACC) used in the regression model for the Hi-C interactions. Each element of x_vars is expected to be a list or matrix of appropriate dimensions matching the $N \times N$ lattice. These covariates are used to adjust for various genomic biases in interaction frequencies.
y	A list of Hi-C interaction count matrices. Each element of the list should be an $N \times N$ numeric matrix representing interaction counts between genomic loci. These data are independently modeled, allowing the function to run separate MCMC chains for each dataset.
theta_start	A numeric value for the initial theta parameter (zero-inflation probability) for ZIP or ZINB distributions. This parameter controls the degree of zero-inflation in the model. If the chosen distribution does not involve zero-inflation, theta_start may be ignored.
size_start	A numeric vector of length 3 providing initial values for the size (overdispersion) parameter, required when dist is Negative Binomial (NB) or Zero-Inflated Negative Binomial (ZINB). Each of the three values corresponds to one mixture component.
use_data_priors	data driven prior.
user_fixed_priors	user-specified prior.
dist	A character string specifying the distribution family to use for modeling interaction counts. Options include: • "Poisson": Poisson distribution • "NB": Negative Binomial distribution • "ZIP": Zero-Inflated Poisson distribution • "ZINB": Zero-Inflated Negative Binomial distribution
epsilon	(Optional) A numeric value for the ABC threshold ϵ. If NULL, the threshold is determined dynamically within the MCMC. This threshold is used to accept or reject proposed γ values based on their ability to produce synthetic data that match observed data according to a given distance metric.
distance_metric	A character string specifying the distance measure used in the ABC step. Supported values include:

- "manhattan": Absolute differences between means
- "euclidean": Squared differences between means

mc_cores — An integer specifying the number of CPU cores to use for parallel processing. If mc_cores > 1, the function runs MCMC chains for multiple datasets concurrently, utilizing multi-core capabilities to reduce total runtime. Defaults to 1 (no parallelization).

Details

The run_chain_betas function provides a convenient interface for running MCMC-based inference on multiple Hi-C datasets simultaneously. It calls run_metropolis_MCMC_betas which performs:

1. Estimation of mixture model parameters (betas) that adjust for biases like genomic distance, GC content, TES, and accessibility.
2. Updating the Potts model interaction parameter γ via Approximate Bayesian Computation (ABC).
3. Handling of zero-inflation (θ) if using ZIP or ZINB distributions.
4. Updating the dispersion parameter (size) for NB or ZINB distributions.

By passing a list of count matrices through y, users can run multiple independent MCMC chains concurrently, facilitating simulation studies, sensitivity analyses, or other large-scale inference tasks.

Value

A list of length equal to the length of y. Each element of the returned list is itself a list (as returned by run_metropolis_MCMC_betas) containing:

- chains: MCMC samples of the regression parameters for each mixture component.
- gamma: Samples of the Potts model interaction parameter.
- theta: Samples of the zero-inflation parameter (if applicable).
- size: Samples of the dispersion parameter (if using NB or ZINB).

See Also

[run_metropolis_MCMC_betas()]

Examples

```
#
# Suppose we have multiple simulated Hi-C data sets:
N <- 5
y <- list(
  matrix(rpois((N*N), lambda = 5), nrow = N)
)


gamma_prior <- 0.3
iterations <- 10
x_vars <- list(
  distance = list(matrix(runif(N * N, 0, 1), nrow = N)),
  GC = list(matrix(runif(N * N, 0, 1), nrow = N)),
```

```
  TES = list(matrix(runif(N * N, 0, 1), nrow = N)),
  ACC = list(matrix(runif(N * N, 0, 1), nrow = N))
)
theta_start <- 0.5
size_start <- c(1, 1, 1)

# Run the MCMC chains in parallel using 2 cores
results <- run_chain_betas(
  N = N,
  gamma_prior = gamma_prior,
  iterations = iterations,
  x_vars = x_vars,
  y = y,
  theta_start = theta_start,
  use_data_priors = TRUE,
  user_fixed_priors = FALSE,
  epsilon = NULL,
  distance_metric = "manhattan",
  dist = "ZINB",
  size_start = size_start,
  mc_cores = 1
)
print(results)
# Each element of 'results' corresponds to the output of run_metropolis_MCMC_betas for each dataset
#

#/donttest{

user_fixed_priors <- list(
 component1 = list(
   meany = 3, meanx1 = 5, meanx2 = 0.2, meanx3 = 0, meanx4 = 4,
   sdy = 1, sdx1 = 0.01, sdx2 = 0.01, sdx3 = 0.002, sdx4 = 0.005
 ),
 component2 = list(
   meany = 300, meanx1 = 8, meanx2 = 0.3, meanx3 = 3, meanx4 = 8,
   sdy = 0.1, sdx1 = 0.1, sdx2 = 0.01, sdx3 = 0.02, sdx4 = 0.02
 ),
 component3 = list(
   meany = 200, meanx1 = 10, meanx2 = 0.3, meanx3 = 1.6, meanx4 = 4.5,
   sdy = 0.1, sdx1 = 0.6, sdx2 = 0.1, sdx3 = 0.2, sdx4 = 0.2
 )
 )

#}
```

`run_metropolis_MCMC_betas` *Run Metropolis-Hastings MCMC for HMRFHiC Model*

Description

Runs a Markov Chain Monte Carlo (MCMC) algorithm using Metropolis-Hastings to sample parameters for the Hidden Markov Random Field (HMRF) model in HMRFHiC. Integrates proposal

distributions, acceptance/rejection steps based on posterior calculations, and Approximate Bayesian Computation (ABC) updates for the γ parameter. Supports data-driven or user-defined priors for covariates.

Usage

```
run_metropolis_MCMC_betas(
  N,
  gamma_prior,
  iterations,
  x_vars,
  y,
  use_data_priors,
  user_fixed_priors = NULL,
  dist = "ZIP",
  epsilon = NULL,
  distance_metric = "manhattan",
  size_start = NULL,
  theta_start = NULL
)
```

Arguments

N	Integer specifying the dimension of the lattice ($N \times N$).
gamma_prior	Numeric. Initial (prior) value for the interaction parameter γ in the Potts model. Set at 0.3 as default.
iterations	Integer. Number of MCMC iterations to run.
x_vars	A list of covariates used as predictors. Must contain named elements "distance", "GC", "TES", and "ACC", each a list of $N \times N$ matrices.
y	An $N \times N$ numeric matrix of observed interaction counts, with elements corresponding to locus pairs (i,j).
use_data_priors	Logical. If TRUE, uses data-driven priors for each component. If FALSE, user_fixed_priors must be provided.
user_fixed_priors	(Optional) A list of user-specified priors for model components if use_data_priors = FALSE. Each component's priors should include means and standard deviations.
dist	Character string specifying the distribution family: "Poisson", "NB" (Negative Binomial), "ZIP" (Zero-Inflated Poisson, default), or "ZINB" (Zero-Inflated Negative Binomial).
epsilon	(Optional) Numeric tolerance for the ABC update of γ. If NULL, computed dynamically from data.
distance_metric	Character string specifying the distance metric for ABC: "manhattan" (default) or "euclidean".
size_start	(Optional) Numeric vector of length 3 specifying initial size (overdispersion) parameters for each component if dist is "NB" or "ZINB".
theta_start	(Optional) Numeric initial value for θ, the zero-inflation parameter in ZIP/ZINB models. Defaults to 0.5 if NULL and dist is ZIP/ZINB.

Details

The algorithm proceeds as follows:

1. **Initialization:** Initializes the component assignment matrix z based on `y` and sets starting values for parameters (γ, θ, and size if applicable).
2. **MCMC Updates per Iteration:**
   - Updates latent assignments z by proposing new states and computing posterior probabilities using internal functions.
   - Updates model parameters (betas) for each component via Metropolis-Hastings:
     - Proposes new values from normal distributions.
     - Computes posterior probabilities and accepts/rejects proposals.
   - Updates overdispersion `size` or zero-inflation θ parameters if applicable.
   - Updates γ via ABC:
     - Proposes a new γ and simulates synthetic data.
     - Compares to observed data using the specified `distance_metric` and ϵ.
     - Accepts/rejects based on whether the distance is below ϵ.
3. **Adaptive Tuning:** Tunes proposal standard deviations to achieve a target acceptance rate.

Value

A list containing:

- `chains`: List of three $(iterations + 1) \times 5$ matrices, each storing parameter chains for one of the three components.
- `gamma`: Numeric vector of length $iterations + 1$, the chain of γ values.
- `theta`: Numeric vector of length $iterations + 1$, the chain of θ values (unchanged if not ZIP/ZINB).
- `size`: $3 \times (iterations + 1)$ matrix of size (overdispersion) parameters if `dist` is NB or ZINB; otherwise unused.

Examples

```
N <- 10
gamma_prior <- 0.5
iterations <- 10
x_vars <- list(
  distance = list(matrix(runif(N * N, 0, 10), nrow = N)),
  GC = list(matrix(runif(N * N, 0, 1), nrow = N)),
  TES = list(matrix(runif(N * N, 0, 2), nrow = N)),
  ACC = list(matrix(runif(N * N, 0, 5), nrow = N))
)
y <- matrix(rpois(N * N, lambda = 5), nrow = N)
results <- run_metropolis_MCMC_betas(
  N = N,
  gamma_prior = gamma_prior,
  iterations = iterations,
  x_vars = x_vars,
  y = y,
  use_data_priors = TRUE,
  dist = "ZIP"
)
plot(results$gamma, type = "l")
```

size_prior *Size Prior for the Negative Binomial-Type Distributions*

Description

This function computes the prior contribution of the size parameter (also known as the dispersion parameter) for a specified component in models using a Negative Binomial (NB) or Zero-Inflated Negative Binomial (ZINB) distribution. The prior is assumed to follow a Gamma distribution.

Usage

```
size_prior(size_value, component)
```

Arguments

size_value	A numeric value representing the size (dispersion) parameter of the NB or ZINB distribution. This parameter controls the variance of the distribution, with larger values implying less overdispersion.
component	An integer specifying which component of the mixture model to consider (e.g., 1, 2, or 3). Different components can have different prior distributions for the size parameter.

Details

In NB and ZINB distributions, the size parameter controls the variance relative to the mean. Bayesian inference often places a prior on this parameter to regularize its estimation. By assigning a Gamma prior, we leverage its conjugacy properties and flexibility in capturing a range of plausible dispersion values.

This function assigns a component-specific Gamma prior. Each component can have distinct shape (and possibly rate) parameters, allowing the model to accommodate different dispersion characteristics across mixture components.

The returned value is the log-density of the Gamma prior at the given size_value.

Value

A numeric value representing the log of the prior density for the size_value parameter under the Gamma distribution defined for the specified component.

Examples

```
# Example: Compute the size prior for size_value = 2.5 in component 1
log_prior_comp1 <- size_prior(size_value = 2.5, component = 1)
#log_prior_comp1
```

Index