Package: consensusTADs (via r-universe)

June 23, 2026

Title Generate Consensus Topologically Associating Domains from Multiple Prediction Tools

Version 0.99.1

Description Integrates Topologically Associating Domains (TADs) predictions from multiple computational tools to generate high-confidence consensus TAD sets. The package implements the Measure of Concordance (MoC) metric to quantify agreement between different TAD predictions and uses dynamic programming algorithms to select optimal non-overlapping TAD boundaries. This approach helps resolve inconsistencies between TAD calling methods and produces more reliable chromatin domain annotations for downstream genomic analyses.

License MIT + file LICENSE

Encoding UTF-8

biocViews Software, HiC

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports dplyr, furrr, future, GenomicRanges, IRanges, magrittr, purrr, stringr, tibble, tidyr

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/CSOgroup/consensusTADs

BugReports https://github.com/CSOgroup/consensusTADs/issues

Config/pak/sysreqs libicu-dev

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

Date/Publication 2026-06-23 11:04:38 UTC

RemoteUrl https://github.com/BiocStaging/consensusTADs

RemoteRef HEAD

RemoteSha 7731a201ac07926c7ae2edbea57eaa61a2d0221b

Contents

`generate_tad_consensus` *Generate consensus TADs from multiple tool predictions*

Description

This function generates consensus TADs from predictions made by multiple tools. It applies an iterative threshold approach to select optimal non-overlapping TADs that represent the consensus across different prediction methods.

Parallel processing is controlled by the future framework. Configure it before calling: future::plan(future::multisession(worke = 4))

Usage

```
generate_tad_consensus(
  df_tools,
  threshold = 0,
  step = -0.05,
  split_vars = c("chr"),
  include_isolated = FALSE,
  consider_level = FALSE
)
```

Arguments

`df_tools`	Data frame containing TAD information with columns: chr, start, end, meta.tool where meta.tool identifies the prediction tool source
`threshold`	Numeric value, the minimum threshold for MoC filtering, default is 0
`step`	Numeric vector, sequence of threshold values to use in the iterative selection process, default is -0.05
`split_vars`	Character vector, variables to split data by for parallel processing, default is c("chr")
`include_isolated`	Logical, whether to include isolated TADs (with no overlaps) when threshold is 0, default is FALSE
`consider_level`	Logical, whether to consider meta.tool_level when filtering overlaps, default is FALSE

Value

Data frame containing both the original tool TADs and consensus TADs with additional columns: score_source (metadata about contributing tools), threshold (the MoC threshold at which each consensus TAD was selected)

Examples

```
tad_data <- data.frame(
  chr = rep("chr1", 6),
  start = c(10000, 20000, 50000, 12000, 22000, 48000),
  end = c(30000, 45000, 65000, 32000, 43000, 67000),
  meta.tool = c(rep("tool1", 3), rep("tool2", 3))
)

# Sequential (default)
consensus_results <- generate_tad_consensus(tad_data)

# Generate consensus TADs with custom threshold values
consensus_results <- generate_tad_consensus(
  tad_data,
  threshold = 0.8,
  step = -0.05
)

# Parallel controlled by environment
options(future.globals.maxSize = 100 * 1024^3)
future::plan(future::multisession(workers = 4))
consensus_results <- generate_tad_consensus(tad_data)
future::plan(future::sequential)
```

---

generate_tad_consensus_hierarchy

Generate hierarchical consensus TADs through iterative rounds

---

Description

This function generates consensus TADs through multiple rounds of iteration. In each round, it identifies consensus TADs and removes partially overlapping regions from the input data for the next round. This hierarchical approach helps identify TADs at different levels of consensus strength.

Parallel processing is controlled by the future framework. Configure it before calling: future::plan(future::multisession(worke = 4))

Usage

```
generate_tad_consensus_hierarchy(
  df_tools,
  threshold = 0,
```

```
  step = -0.05,
  split_vars = c("chr"),
  max_round = 10,
  include_isolated = FALSE,
  consider_level = FALSE
)
```

Arguments

df_tools	Data frame containing TAD information with columns: chr, start, end, meta.tool
threshold	Numeric value, the minimum threshold for MoC filtering, default is 0
step	Numeric vector, sequence of threshold values to use in the iterative selection process, default is -0.05
split_vars	Character vector, variables to split data by for parallel processing, default is c("chr")
max_round	Integer, maximum number of rounds to perform. If NULL, continues until no more TADs remain in the input data. Default is 10
include_isolated	Logical, whether to include isolated TADs (with no overlaps) when threshold is 0, default is FALSE
consider_level	Logical, whether to consider meta.tool_level when filtering overlaps, default is FALSE

Value

Data frame containing all consensus TADs with round information

Examples

```
tad_data <- data.frame(
  chr = rep("chr1", 6),
  start = c(10000, 20000, 50000, 12000, 22000, 48000),
  end = c(30000, 45000, 65000, 32000, 43000, 67000),
  meta.tool = c(rep("tool1", 3), rep("tool2", 3))
)

# Basic usage
result <- generate_tad_consensus_hierarchy(tad_data, max_round = 3)

# Parallel processing
options(future.globals.maxSize = 100 * 1024^3)
future::plan(future::multisession(workers = 4))
result <- generate_tad_consensus_hierarchy(tad_data, max_round = 5)
future::plan(future::sequential)

# With tool levels
tad_data_with_level <- data.frame(
  chr = rep("chr1", 8),
  start = c(10000, 15000, 20000, 50000, 55000, 15000, 50000, 80000),
```

```
  end = c(30000, 35000, 45000, 70000, 75000, 35000, 70000, 100000),
  meta.tool = c("tool1", "tool1", "tool2", "tool3", "tool3", "tool2", "tool1", "tool4"),
  meta.tool_level = c("L1", "L2", NA, "L1", "L2", NA, "L2", NA)
)

result_hierarchy <- generate_tad_consensus_hierarchy(
  tad_data_with_level,
  max_round = 3,
  consider_level = TRUE
)
```

`moc_score_filter` *Calculate and filter Measure of Concordance (MoC) between TADs*

Description

This function calculates the Measure of Concordance (MoC) between TADs in the input data frame and filters significant overlaps based on a threshold. The MoC is calculated as: intersect.width^2 / (width1 * width2), where intersect.width is the overlap length between two regions, and width1 and width2 are the lengths of the two regions.

Usage

```
moc_score_filter(
  tb_tool_sel,
  moc_cut,
  include_moc_cut = TRUE,
  include_isolated = FALSE,
  consider_level = FALSE
)
```

Arguments

`tb_tool_sel`	Data frame containing TAD coordinates. Must include columns: chr, start, end, meta.tool. Optionally can include meta.tool_level for finer tool classification
`moc_cut`	Numeric value, threshold for MoC
`include_moc_cut`	Logical, whether to include results equal to MoC threshold, default is TRUE
`include_isolated`	Logical, whether to include isolated TADs (with no overlaps) when moc_cut is 0. These TADs will have moc_score = 0. Default is FALSE
`consider_level`	Logical, whether to consider meta.tool_level when filtering overlaps. If TRUE and meta.tool_level exists, different levels of the same tool are treated as different tools. Default is FALSE

Value

Data frame containing merged TAD information with calculated MoC scores and the following columns:

`chr`	Character, the chromosome name where the TAD is located
`start`	Integer, the start coordinate of the TAD
`end`	Integer, the end coordinate of the TAD
`moc_score`	Numeric, the Measure of Concordance (MoC) score calculated for the TAD, representing the level of agreement between different TADs
`score_source`	Character, a string containing information about the tools that contributed to this TAD and their individual MoC scores

Examples

```
# Prepare input data
tad_data <- data.frame(
  chr = rep("chr1", 3),
  start = c(10000, 20000, 50000),
  end = c(30000, 45000, 65000),
  meta.tool = c("tool1", "tool2", "tool3")
)

# Calculate MoC
results <- moc_score_filter(tad_data, moc_cut = 0.1)

# Include isolated TADs when moc_cut is 0
results_with_isolated <- moc_score_filter(tad_data, moc_cut = 0, include_isolated = TRUE)

# With tool levels
tad_data_with_level <- data.frame(
  chr = rep("chr1", 8),
  start = c(10000, 15000, 20000, 50000, 55000, 15000, 50000, 50000),
  end = c(30000, 35000, 45000, 70000, 75000, 35000, 70000, 70000),
  meta.tool = c("tool1", "tool1", "tool2", "tool3", "tool3", "tool2", "tool1", "tool4"),
  meta.tool_level = c("L1", "L2", NA, "L1", "L2", NA, "L2", NA)
)

# Without considering levels - tool1(L1) and tool1(L2) are treated as same tool
results_no_level <- moc_score_filter(tad_data_with_level, moc_cut = 0.1, consider_level = FALSE)
# Output shows overlaps between tool1, tool2, tool3

# With considering levels - tool1(L1) and tool1(L2) are treated as different tools
results_with_level <- moc_score_filter(tad_data_with_level, moc_cut = 0.1, consider_level = TRUE)
# Output shows overlaps between tool1(L1), tool1(L2), tool2, tool3(L1), tool3(L2)
# score_source will show format like: tool1(L1)_0.5; tool2_0.3
```

`select_global_optimal_tads` *Select globally optimal non-overlapping TADs using dynamic programming*

Description

This function implements a dynamic programming algorithm to select a set of non-overlapping TADs that maximize the total MoC score. The algorithm sorts TADs by their end coordinates and builds an optimal solution by either including or excluding each TAD based on which choice yields the highest total score.

Usage

```
select_global_optimal_tads(tad_all)
```

Arguments

`tad_all`	Data frame containing TAD information with columns: chr, start, end, moc_score, score_source

Value

Data frame containing the selected non-overlapping TADs that maximize total score

Examples

```
# Prepare input data
tad_data <- data.frame(
  chr = rep("chr1", 4),
  start = c(10000, 20000, 50000, 70000),
  end = c(30000, 45000, 65000, 90000),
  moc_score = c(2.5, 3.2, 1.8, 4.1),
  score_source = c("tool1, tool2", "tool1, tool3", "tool2, tool3", "tool1, tool4")
)

# Select optimal TADs
optimal_tads <- select_global_optimal_tads(tad_data)
```

`select_tads_by_threshold` *Select optimal non-overlapping TADs using a single threshold*

Description

This function selects a set of optimal non-overlapping TADs by first filtering TADs based on the provided MoC threshold, then applying a global optimization algorithm to select TADs that maximize the total score without overlaps.

Usage

```
select_tads_by_threshold(
  tb_tool_sel,
  threshold,
  include_threshold = TRUE,
  considering_width = TRUE,
  include_isolated = FALSE,
  consider_level = FALSE
)
```

Arguments

`tb_tool_sel`	Data frame containing TAD information with columns: chr, start, end, meta.tool
`threshold`	Numeric value, threshold for filtering TADs based on MoC score
`include_threshold`	Logical, whether to include TADs equal to the threshold, default is TRUE
`considering_width`	Logical, whether to adjust scores by TAD width, default is TRUE
`include_isolated`	Logical, whether to include isolated TADs (with no overlaps) when threshold is 0, default is FALSE
`consider_level`	Logical, whether to consider meta.tool_level when filtering overlaps, default is FALSE

Value

Data frame containing the selected optimal non-overlapping TADs

Examples

```
# Prepare input data
tad_data <- data.frame(
  chr = rep("chr1", 4),
  start = c(10000, 20000, 50000, 70000),
  end = c(30000, 45000, 65000, 90000),
```

```
  meta.tool = c("tool1", "tool2", "tool3", "tool4")
)

# Select TADs with threshold 0.2
selected_tads <- select_tads_by_threshold(tad_data, threshold = 0.2)
```

`select_tads_by_threshold_series` *Select TADs using a series of decreasing thresholds*

Description

This function iteratively selects TADs using a series of decreasing MoC thresholds. It starts with the highest threshold and gradually processes the remaining unselected TADs with lower thresholds. For each iteration, it removes previously selected TADs from consideration to avoid redundancy.

Usage

```
select_tads_by_threshold_series(
  tb_tool_sel,
  threshold_c,
  include_threshold = TRUE,
  considering_width = TRUE,
  include_isolated = FALSE,
  consider_level = FALSE
)
```

Arguments

`tb_tool_sel`	Data frame containing TAD information with columns: chr, start, end, meta.tool
`threshold_c`	Numeric vector, series of decreasing MoC thresholds
`include_threshold`	Logical, whether to include TADs equal to the threshold, default is TRUE
`considering_width`	Logical, whether to adjust scores by TAD width, default is TRUE
`include_isolated`	Logical, whether to include isolated TADs (with no overlaps) when threshold is 0, default is FALSE
`consider_level`	Logical, whether to consider meta.tool_level when filtering overlaps, default is FALSE

Value

Data frame containing the selected optimal non-overlapping TADs

Examples

```
# Prepare input data
tad_data <- data.frame(
  chr = rep("chr1", 5),
  start = c(10000, 20000, 50000, 70000, 90000),
  end = c(30000, 45000, 65000, 85000, 110000),
  meta.tool = c("tool1", "tool2", "tool3", "tool1", "tool2")
)

# Select TADs using a series of thresholds
selected_tads <- select_tads_by_threshold_series(
  tad_data,
  threshold_c = round(seq(1, 0.2, -0.05), 2)
)
```

Index