Package 'TmCalculator'

June 25, 2026

Type Package

Title Genome-Wide Nucleic Acid Melting Temperature Profiling and Multi-Omics Integration

Version 1.0.7

Date 2026-06-25

Description Accurate calculation of nucleic acid melting temperature (Tm) is fundamental to many molecular biology applications, and this software scales Tm analysis from individual sequences to genome-wide thermodynamic profiling. This package extends Tm analysis from simple sequence level computation to comprehensive genome-wide thermodynamic profiling. It takes multiple input formats including sequence strings, FASTA files, genomic coordinates. The implementation provides three Tm calculation methods: the Wallace rule (Thein & Wallace, 1986), empirical GC-content formulas (Marmur, 1962; Schildkraut, 2010; Wetmur, 1991; Untergasser, 2012; von Ahsen, 2001), and nearest-neighbor thermodynamics (Breslauer, 1986; Sugimoto, 1996; Allawi, 1998; SantaLucia, 2004; Freier, 1986; Xia, 1998; Chen, 2012; Bommarito, 2000; Turner, 2010; Sugimoto, 1995; Allawi, 1997; SantaLucia, 2005). Corrections are supported for salt ions (SantaLucia, 1996, 1998; Owczarzy, 2004, 2008) and for chemical conditions such as dimethyl sulfoxide and formamide. This package returns result as a GRanges object for interoperability with Bioconductor workflows and downstream multi-omics analyses. Data-level integration reconciles Tm windows with external multi-omics GRanges objects through overlap, nearest-feature, windowed-count, and binned-average strategies, returning a single unified GRanges object ready for downstream analysis. Visualization-level integration renders multiple feature layers as independent concentric tracks on a shared genomic axis, each retaining its native coordinate resolution. Group comparison supports Wilcoxon rank-sum and Student's t-tests with multiple available correction methods for contrasting Tm and other features across region classes.

BugReports https://github.com/JunhuiLi1017/TmCalculator/issues

License MIT + file LICENSE

Depends R (>= 3.5)

VignetteBuilder knitr

Imports BSgenome, BiocGenerics, Biostrings, GenomeInfoDb, GenomicRanges, IRanges, S4Vectors, graphics, grDevices, methods

Suggests testthat (>= 3.0.0), knitr, utils, rmarkdown, remotes, BiocManager, BSgenomeForge, BSgenome.Hsapiens.UCSC.hg38, karyoploteR, ggplot2, ggridges, ggforce, rlang, seqinr

NeedsCompilation no

Repository CRAN

RoxygenNote 7.3.2

LazyData true

LazyDataCompression xz

Encoding UTF-8

Author Junhui Li [cre, aut] (ORCID: <https://orcid.org/0000-0003-3973-1700>),
Lihua Julie Zhu [aut] (ORCID: <https://orcid.org/0000-0001-7416-0590>)

Maintainer Junhui Li <ljh.biostat@gmail.com>

Date/Publication 2026-06-25 11:10:17 UTC

Contents

check_filter_seq *Filter invalid bases in nucleotide sequences*

Description

This function processes nucleotide sequences by converting characters to uppercase and replacing invalid bases with "". based on the specified method. The function preserves the sequence length and attributes (name and Tm) of each sequence.

Usage

```
check_filter_seq(seq_list, method)
```

Arguments

seq_list	Input sequence in 5' to 3' direction. Must be provided as: - A list of sequences with attributes (name and Tm)
method	Method to determine valid bases:
	TM_Wallace: Valid bases are "A","B","C","D","G","H","I","K","M","N","R","S","T","V","W" and "Y"
	TM_GC: Valid bases are "A","B","C","D","G","H","I","K","M","N","R","S","T","V","W", "X" and "Y"
	TM_NN: Valid bases are "A","C","G","I" and "T"

Value

Returns a list of sequences with the same structure as input, where invalid bases are replaced with ""

Author(s)

Junhui Li

References

```
citation("TmCalculator")
```

chem_correct *Corrections of melting temperature with chemical substances*

Description

Apply corrections to melting temperature calculations based on the presence of DMSO and formamide. These corrections are rough approximations and should be used with caution.

Usage

```
chem_correct(
  DMSO = 0,
  formamide_unit = list(value = 0, unit = "percent"),
  dmso_factor = 0.75,
  formamide_factor = 0.65,
  pt_gc = NULL
)
```

Arguments

DMSO	Percent DMSO concentration in the reaction mixture. Default: 0 DMSO can lower the melting temperature of nucleic acid duplexes.
formamide_unit	A list containing formamide concentration information: - value: numeric value of formamide concentration - unit: character string specifying the unit ("percent" or "molar") Default: list(value=0, unit="percent")
dmso_factor	Coefficient of melting temperature (Tm) decrease per percent DMSO. Default: 0.75 (von Ahsen N, 2001, PMID:11673362) Other published values: 0.5, 0.6, 0.675
formamide_factor	Coefficient of melting temperature (Tm) decrease per percent formamide. Default: 0.65 Literature reports values ranging from 0.6 to 0.72
pt_gc	Percentage of GC content in the sequence (0-100 This is used in molar formamide corrections.

Details

When formamide_unit$unit = "percent": Correction = - factor * percentage_of_formamide

When formamide_unit$unit = "molar": Correction = (0.453 * GC/100 - 2.88) * formamide

Author(s)

Junhui Li

References

von Ahsen N, Wittwer CT, Schutz E, et al. Oligonucleotide melting temperatures under PCR conditions: deoxynucleotide Triphosphate and Dimethyl sulfoxide concentrations with comparison to alternative empirical formulas. Clin Chem 2001, 47:1956-1961.

Examples

```
# DMSO correction
chem_correct(DMSO = 3)

# Formamide correction (percent)
chem_correct(formamide_unit = list(value = 1.25, unit = "percent"), pt_gc = 50)

# Formamide correction (molar)
chem_correct(formamide_unit = list(value = 1.25, unit = "molar"), pt_gc = 50)
```

`compare_groups` *Compare numeric GRanges metadata across groups*

Description

Test whether one or more numeric metadata columns (e.g. `Tm`, `GC`) differ between levels of a categorical grouping column in a `GRanges` object.

Usage

```
compare_groups(
  gr,
  target = "Tm",
  method = c("wilcoxon", "t.test"),
  group,
  min_n_per_group = 2L,
  paired = FALSE,
  alternative = c("two.sided", "less", "greater"),
  posthoc = TRUE,
 p.adjust.method = c("holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr",
    "none")
)
```

Arguments

`gr`	A `GRanges` object.
`target`	Character vector of metadata column names to test (e.g. `c("Tm", "GC")`). All must be numeric.
`method`	Statistical test family. One of: • `"wilcoxon"`: rank-based tests (Wilcoxon / Kruskal-Wallis). • `"t.test"`: Gaussian-based tests (t-test / ANOVA).
`group`	Character. Name of a metadata column in `gr` defining groups.
`min_n_per_group`	Minimum number of non-missing observations required per group. Default: `2`.

`paired` Logical. Only for exactly two groups. Default: `FALSE`.

`alternative` Character. Alternative hypothesis for two-group tests. One of `"two.sided"` (default), `"less"`, or `"greater"`.

`posthoc` Logical. For ≥ 3 groups, run pairwise follow-up tests with multiple-testing correction. Default: `TRUE`.

`p.adjust.method`
Multiple-testing correction for post-hoc pairwise comparisons. Passed to `pairwise.wilcox.test` / `pairwise.t.test`. One of: `"holm"` (default), `"hochberg"`, `"hommel"`, `"bonferroni"`, `"BH"` (Benjamini-Hochberg FDR), `"BY"` (Benjamini-Yekutieli), `"fdr"` (alias of `"BH"`), or `"none"`.

Details

The test used depends on `method` and the number of group levels:

Groups	`method = "wilcoxon"`	`method = "t.test"`
2	Wilcoxon rank-sum (or signed-rank if `paired = TRUE`)	Welch two-sample t-test (or paired t-test)
≥ 3	Kruskal-Wallis	One-way ANOVA (Welch)

When there are three or more groups and `posthoc = TRUE`, pairwise follow-up tests are run (Wilcoxon or Welch t-test) with `p.adjust` multiple-testing correction.

Value

A list with:

`results` Data frame with one row per `target`: omnibus test name, statistic, `p.value`, and group information.

`summary` Per-group descriptive statistics (`n`, `mean`, `sd`, `median`).

`pairwise` Pairwise comparisons when `posthoc = TRUE` and there are ≥ 3 groups; otherwise `NULL`.

Author(s)

Junhui Li

Examples

```
## Not run:
library(GenomicRanges)
set.seed(1)
gr <- GRanges(
  seqnames = rep("chr1", 90),
  ranges = IRanges(start = sort(sample(1e6, 90)), width = 50),
  Tm = c(rnorm(30, 74, 2), rnorm(30, 70, 2), rnorm(30, 66, 2)),
  GC = runif(90, 40, 60)
)
gr$group <- rep(c("high", "mid", "low"), each = 30)
```

```
# Two groups
gr2 <- gr[gr$group %in% c("high", "low")]
compare_groups(gr2, target = "Tm", group = "group", posthoc = FALSE)

# Three or more groups (Kruskal-Wallis + pairwise Wilcoxon)
compare_groups(gr, target = c("Tm", "GC"), group = "group",
                    method = "wilcoxon")

# Three or more groups (one-way ANOVA + pairwise t-tests)
compare_groups(gr, target = "Tm", group = "group", method = "t.test")

# Post-hoc with Benjamini-Hochberg FDR control
compare_groups(gr, target = "Tm", group = "group",
                    p.adjust.method = "BH")

## End(Not run)
```

`complement_fast` *Fast complement and reverse complement*

Description

Uses chartr() instead of character-by-character translation. ~10x faster than the seqinr-style implementation for long sequences.

Usage

```
complement_fast(seq_str, rev = FALSE)
```

Arguments

`seq_str`	Character string or vector.
`rev`	Logical. Return reverse complement? Default FALSE.

Value

Character string(s) with complement.

`coor_to_genomic_ranges` *Convert genomic coordinate strings to a GRanges object*

Description

Fast conversion of genomic coordinate strings to a `GRanges` object with reference sequences fetched from installed `BSgenome.*` packages. Designed for large sliding-window inputs: genome packages are loaded once, coordinate strings are parsed in one pass, and sequences are extracted with vectorized `getSeq` (or optional chromosome preloading).

Usage

```
coor_to_genomic_ranges(
  input,
  complement_seq = NULL,
  method = c("vectorized", "preload_chr")
)
```

Arguments

`input`	Coordinate input. Either: • A list with `pkg_name` (BSgenome package name) and `seq` (character vector of coordinate strings). Preferred for many windows on the same genome. • A plain character vector of coordinate strings (legacy format; genome package name is read from field 4 when present). Supported colon-separated formats: • `chr:start-end:strand:region_id` - requires `pkg_name` in the input list. • `chr:start-end:strand:pkg_name:region_id` - as produced by `make_genomiccoord`.
`complement_seq`	Optional complement coordinates in the same format as `input$seq`. When `NULL`, complements are generated automatically from `sequence`.
`method`	Sequence extraction strategy: `"vectorized"` One `getSeq()` call per genome package (default). `"preload_chr"` Load each chromosome once and extract windows with `subseq()`. Faster for dense whole-chromosome tiling; uses more memory.

Details

For genome-wide tiling with thousands of windows, pass coordinates as `list(pkg_name = "BSgenome.Hsapiens.UCSC.hg3`
`seq = ...)` so the genome package is loaded once instead of per interval.

Value

A GRanges object with metadata columns:

sequence Reference sequence for each interval.

complement Complementary sequence.

GC GC fraction (0-1) per interval.

region_id Region identifier from the coordinate string.

genome_pkg BSgenome package name used.

Author(s)

Junhui Li

See Also

to_genomic_ranges, to_genomic_ranges_fast

Examples

```
## Not run:
coords <- c(
  "chr1:1000-1199:+:win1",
  "chr1:1200-1399:+:win2"
)
gr <- coor_to_genomic_ranges(
  list(pkg_name = "BSgenome.Hsapiens.UCSC.hg38", seq = coords)
)
gr

## End(Not run)
```

---

ecoli_rep_hotspots *E. coli K-12 MG1655 replication-associated hotspot annotations*

Description

A named list of genomic feature tables for *Escherichia coli* K-12 MG1655 (NCBI assembly GCF_000005845.2 / ASM584v2, chromosome U00096.3). The data support the genome-wide Tm vignette and plot_genome_track examples by providing independent multi-omics layers that can be overlaid on the same genomic axis alongside Tm profiles computed with tm_calculate.

Usage

```
ecoli_rep_hotspots
```

Format

A named list with five elements:

`all_peaks_IP_mutH` A data frame (38 rows) of MutL protein ChIP-seq peaks marking mismatch-repair-associated regions (MutL-AR). Columns: `chr`, `start`, `end`, `Sample`, `name`, `Size`.

`bins_rep` A data frame (4,642 rows) of tandem-repeat (microsatellite) counts in non-overlapping 1 kb bins. Columns: `chr`, `start`, `end`, `count`.

`bins_cru` A data frame (4,642 rows) of cruciform-forming sequence counts in non-overlapping 1 kb bins. Columns: `chr`, `start`, `end`, `count`.

`ssdna` A data frame (2,636 rows) of single-stranded DNA regions. Columns: `chr`, `start`, `end`, `Cells..strand.`, `Region`.

`bins_gatc` A data frame (4,642 rows) of GATC methylation-site counts in non-overlapping 1 kb bins. Columns: `chr`, `start`, `end`, `count`.

All coordinate-based tables use `chr = "U00096.3"` and are compatible with `plot_genome_track` and `compare_groups`.

Source

Curated from published *E. coli* K-12 MG1655 multi-omics datasets used in the package vignette `vignette("genome_wide_tm_ecoli", package = "TmCalculator")`.

Examples

```
data(ecoli_rep_hotspots)
names(ecoli_rep_hotspots)

# MutL-AR peak coordinates
head(ecoli_rep_hotspots$all_peaks_IP_mutH)

# Microsatellite density in 1 kb bins
summary(ecoli_rep_hotspots$bins_rep$count)
```

`fa_to_genomic_ranges` *Convert FASTA file to GenomicRanges object*

Description

This function reads sequences from a FASTA file and converts them to a GenomicRanges object. If named with format ">chr2:1-10:[+|-]:[seq_name]", the name will be parsed into GRanges components.

Usage

```
fa_to_genomic_ranges(input_seq)
```

Arguments

input_seq	Path to the input FASTA file

Value

A GenomicRanges object containing: - GRanges information (seqnames, ranges, strand) - sequence data from FASTA file - Complementary sequences (if provided) - Names from FASTA headers

Examples

```
# Example with single FASTA file
input_seq <- system.file("extdata", "example1.fasta", package = "TmCalculator")
gr <- fa_to_genomic_ranges(input_seq)
```

---

gc *Calculate G and C content of nucleotide sequences*

Description

Calculate G and C content of nucleotide sequences. The function calculates the percentage of G and C bases relative to the total number of A, T, G, and C bases in the sequence.

Usage

```
gc(input_seq, ambiguous = FALSE)
```

Arguments

input_seq	Sequence (5' to 3') of one strand of the nucleic acid duplex. Can be provided as either: - A character string (e.g., "ATGCG") - A path to a FASTA file containing the sequence(s)
ambiguous	Logical. If TRUE, ambiguous bases are taken into account when computing the G and C content. The function handles various ambiguous bases (S, W, M, K, R, Y, V, H, D, B) by proportionally distributing their contribution to GC content based on their possible nucleotide compositions. For example: - S (G or C) contributes fully to GC content - W (A or T) contributes fully to AT content - M (A or C) contributes proportionally based on the ratio of A to C in the sequence - And so on for other ambiguous bases

Value

Content of G and C as a percentage (range from 0 to 100

Author(s)

Junhui Li

Examples

```
# Calculate GC content of a DNA sequence
gc(c("a","t","c","t","g","g","g","c","c","a","g","t","a"))  # 53.85%

# Calculate GC content including ambiguous bases
gc("GCATSWSYK", ambiguous = TRUE)  # 55.56%
```

`generate_complement` *Generate complementary sequence*

Description

Generate the complementary sequence of a nucleic acid sequence, with an option to reverse it.

Usage

```
generate_complement(input_seq, reverse = FALSE)
```

Arguments

`input_seq`	Input sequence(s) in 5' to 3' direction. Must be provided as either: - A character string (e.g., c("ATGCG", "GCTAG"))
`reverse`	Logical. If TRUE, the complementary sequence is reversed (3' to 5'). If FALSE (default), the complementary sequence is in the same direction (5' to 3').

Value

Returns the complementary sequence(s) in the specified direction.

Author(s)

Junhui Li

References

```
citation("TmCalculator")
```

Examples

```
# Generate complementary sequence in same direction (5' to 3')
generate_complement("ATGCG", reverse = FALSE)

# Generate complementary sequence in reverse direction (3' to 5')
generate_complement("ATGCG", reverse = TRUE)
```

integrate_granges *Integrate a Tm GRanges with multi-omic feature ranges*

Description

Combines the output of tm_calculate (a GRanges object with Tm and GC columns) with a second GRanges carrying arbitrary multi-omic metadata (ChIP-seq peaks, ATAC-seq signal, methylation sites, gene annotations, etc.) using one of four positional strategies:

"overlap" Each tm range is annotated with the aggregated metadata of all feature ranges it directly overlaps.

"nearest" Each tm range is annotated with the metadata of its single closest feature range, plus an added distance column.

"window" Each tm range is expanded symmetrically by window_size bp and annotated with aggregated metadata from all features that fall within the expanded window.

"bin" The genomic space covered by the data is tiled into equal-width bins. Each bin is annotated with the mean tm / GC of overlapping tm ranges *and* the aggregated feature values - suitable for joint heatmaps and genome-wide correlation analyses.

For strategies "overlap" and "window", when a single Tm range matches multiple features the default behaviour is to *summarise*: numeric columns are aggregated via agg_fun (default mean), and categorical columns are collapsed to a comma-separated string of unique values.

Usage

```
integrate_granges(
  gr_tm,
  gr_features,
  strategy = c("overlap", "nearest", "window", "bin"),
  feature_cols = NULL,
  prefix = "",
  window_size = 1000L,
  bin_size = 1e+06,
  agg_fun = mean,
  min_overlap = 1L,
  ignore_strand = TRUE,
  keep_unmatched = TRUE,
  distance_col = "distance_to_feature"
)
```

Arguments

gr_tm	A GRanges object produced by tm_calculate() (or tm_calculate()$gr). Must contain at least a Tm metadata column. A gc column is used automatically when present.

gr_features	A GRanges object with multi-omic feature ranges. All (or a subset of) its metadata columns are transferred / aggregated.
strategy	Character. Integration strategy. One of "overlap" (default), "nearest", "window", or "bin".
feature_cols	Character vector. Names of metadata columns in gr_features to transfer. NULL (default) transfers all metadata columns.
prefix	Character. Prefix prepended to transferred column names to avoid clashes with existing columns in gr_tm. Default: "". Use e.g. "feat_" if there are naming conflicts.
window_size	Integer. Half-width (bp) of the symmetric window added around each Tm range in "window" mode. Default: 1000.
bin_size	Integer. Width (bp) of genomic bins in "bin" mode. Default: 1e6 (1 Mb). Smaller values give finer resolution but sparser coverage.
agg_fun	Function. Applied to numeric feature values when multiple features map to the same Tm range / bin. Must accept a numeric vector and an na.rm argument (e.g. mean, median, sum, max). Default: mean.
min_overlap	Integer. Minimum overlap in base pairs required between a Tm range and a feature range in "overlap" mode. Default: 1.
ignore_strand	Logical. If TRUE (default), strand is ignored when finding overlaps / nearest neighbours.
keep_unmatched	Logical. In "overlap" mode only: if TRUE (default) Tm ranges with no overlapping feature are retained with NA in the transferred columns. If FALSE, unmatched Tm ranges are dropped.
distance_col	Character. Name of the distance column added in "nearest" mode. Default: "distance_to_feature".

Value

- "overlap", "nearest", "window": A GRanges object with the same ranges as gr_tm (minus unmatched ranges if keep_unmatched = FALSE), with additional metadata columns from gr_features.
- "bin": A new GRanges of genomic bins. Each bin carries Tm_mean, GC_mean (if available), n_tm_ranges, n_features, and one aggregated column per requested feature column.

Author(s)

Junhui Li

Examples

```
## Not run:
library(GenomicRanges)

# -- Sample data ------------------------------------------------------------
set.seed(42)
gr_tm <- GRanges(
```

```
  seqnames = c(rep("chr1", 60), rep("chr2", 30)),
  ranges   = IRanges(
    start = c(sort(sample(1:249e6, 60)),
              sort(sample(1:243e6, 30))),
    width = sample(50:200, 90, replace = TRUE)
  ),
  Tm = runif(90, 55, 85),
  GC = runif(90, 30, 70)
)

gr_features <- GRanges(
  seqnames = c(rep("chr1", 40), rep("chr2", 20)),
  ranges   = IRanges(
    start = c(sort(sample(1:249e6, 40)),
              sort(sample(1:243e6, 20))),
    width = sample(500:5000, 60, replace = TRUE)
  ),
  score      = runif(60, 0, 100),
  peak_type  = sample(c("narrow", "broad"), 60, replace = TRUE),
  signal     = rnorm(60, 5, 2)
)

# Strategy 1: overlap - annotate Tm ranges with overlapping peak features
res_overlap <- integrate_granges(gr_tm, gr_features,
                                 strategy = "overlap")

# Strategy 2: nearest - every Tm range gets its closest peak + distance
res_nearest <- integrate_granges(gr_tm, gr_features,
                                 strategy = "nearest")
head(res_nearest$distance_to_feature)

# Strategy 3: window - 5 kb window around each probe
res_window <- integrate_granges(gr_tm, gr_features,
                                strategy = "window", window_size = 5000)

# Strategy 4: bin - 500 kb genome bins with mean Tm and aggregated signal
res_bin <- integrate_granges(gr_tm, gr_features,
                             strategy = "bin", bin_size = 5e5)
as.data.frame(res_bin) |> head()

# Use a subset of feature columns and add a prefix
integrate_granges(gr_tm, gr_features,
                  strategy    = "overlap",
                  feature_cols = c("score", "peak_type"),
                  prefix       = "chip_")

## End(Not run)
```

---

make_genomiccoord *Generate sliding-window genomic coordinate strings for Tm calculation*

Description

Tiles one or more chromosomes from a **BSgenome** object into overlapping windows and returns a named character vector of coordinate strings in the format `"chr:start-end:strand:genome_pkg:region_id"`. This vector is the primary input for downstream Tm calculation functions (`tm_nn`, `tm_gc`, `tm_wallace`) that accept genomic coordinate strings.

Usage

```
make_genomiccoord(
  bsgenome,
  chromosomes = NULL,
  window = 200L,
  slide = 50L,
  start = NULL,
  end = NULL,
  strand = "+",
  trim_N = c("ends", "filter", "none"),
  max_N_frac = 0.1,
  N_scan_block = window,
  region_prefix = "region",
  genome_pkg_name = NULL,
  as_vector = TRUE,
  verbose = TRUE
)
```

Arguments

`bsgenome`	A `BSgenome` object (e.g. `BSgenome.Hsapiens.UCSC.hg38`) **or** a character string with the BSgenome package name (loaded automatically).
`chromosomes`	Character vector of chromosome names to tile. Must be present in `seqlevels(bsgenome)`. Default: the 24 standard human chromosomes `paste0("chr", c(1:22, "X", "Y"))`.
`window`	Integer. Width of each sliding window in base pairs. Default `200L`.
`slide`	Integer. Step size between consecutive window starts in base pairs. `slide == window` gives non-overlapping tiling; `slide < window` gives overlapping windows. Default `50L`.
`start`	Integer **or** `NULL`. Override the start position for every chromosome. When `NULL` (default) the start is chromosome position 1 (adjusted for N-trimming if `trim_N != "none"`). Can be a named integer vector to set different starts per chromosome.
`end`	Integer **or** `NULL`. Override the end position for every chromosome. When `NULL` (default) the end is the chromosome length (adjusted for N-trimming). Can be a named integer vector.
`strand`	Character. Strand label embedded in the coordinate string. One of `"+"` (default) or `"-"`.
`trim_N`	Character. N-base handling strategy. One of `"ends"` (default), `"filter"`, or `"none"`. See section *N-base trimming* above.

max_N_frac	Numeric in $[0, 1]$. Maximum fraction of N bases tolerated per window. Windows exceeding this threshold are dropped. Only used when trim_N = "filter". Default 0.1.
N_scan_block	Integer. Block size (bp) for the coarse N-end detection scan used by trim_N = "ends". Larger values are faster but less precise. Default 10000L.
region_prefix	Character. Prefix for region IDs embedded in the coordinate string. Default "region" (producing region1, region2, ...).
genome_pkg_name	Character **or** NULL. The genome package name to embed in the coordinate string (4th field). When NULL (default), the name is extracted automatically from the bsgenome object via S4Vectors::metadata(bsgenome). Supply explicitly when using a custom BSgenome object whose metadata name differs from the canonical package name.
as_vector	Logical. When TRUE (default), return a plain named character vector. When FALSE, return a data.frame with columns coord, chr, start, end, strand, genome, region_id, chr_start_used, chr_end_used – useful for downstream GRanges construction.
verbose	Logical. Print per-chromosome progress messages. Default TRUE.

Value

When as_vector = TRUE (default): a named character vector of coordinate strings, one element per window. Names are the region IDs (region1, region2, ...).

When as_vector = FALSE: a data.frame with columns coord, chr, win_start, win_end, strand, genome, region_id, chr_start_used, chr_end_used.

Coordinate string format

Each element of the returned vector follows the pattern:

```
chr1:10001-10200:+:BSgenome.Hsapiens.UCSC.hg38:region1
chr1:10051-10250:+:BSgenome.Hsapiens.UCSC.hg38:region2
...
```

Fields (colon-separated):

1. **chromosome** – e.g. chr1
2. **start-end** – 1-based, inclusive coordinates
3. **strand** – + or -
4. **genome** – BSgenome package name (character)
5. **region_id** – unique label regionN

N-base trimming

Human (and most mammalian) chromosomes begin and end with long stretches of N bases that represent assembly gaps. Windows that consist entirely or predominantly of Ns produce meaningless Tm values. The function offers two trimming strategies controlled by trim_N:

`"none"` No trimming. Windows start at `start` and end at `end` (or chromosome length).

`"ends"` Detect the first and last positions of non-N bases on each chromosome using `Biostrings::letterFrequency()` on a coarse block scan, then trim `start` and `end` to those positions. Efficient: reads the chromosome sequence only once in blocks.

`"filter"` Generate windows across the full `start`-`end` range but then remove any window whose N-base fraction exceeds `max_N_frac` (default 0.1, i.e. 10%). More granular than `"ends"` but slower because it reads each window sequence.

Author(s)

Junhui Li

See Also

`tm_nn`, `tm_gc`, `tm_wallace`, `BSgenome`, `letterFrequency`

Examples

```
## Not run:
library(BSgenome.Hsapiens.UCSC.hg38)

## -- Basic usage: tile chr1 with 200 bp windows, 50 bp slide ----------
coords <- make_genomiccoord(
  bsgenome    = BSgenome.Hsapiens.UCSC.hg38,
  chromosomes = "chr1",
  window      = 200L,
  slide       = 50L
)
length(coords)      # number of windows on chr1
head(coords, 3)
# region1 "chr1:10001-10200:+:BSgenome.Hsapiens.UCSC.hg38:region1"
# region2 "chr1:10051-10250:+:BSgenome.Hsapiens.UCSC.hg38:region2"
# region3 "chr1:10101-10300:+:BSgenome.Hsapiens.UCSC.hg38:region3"

## -- Non-overlapping tiling (slide == window) -------------------------
coords_nonoverlap <- make_genomiccoord(
  bsgenome    = BSgenome.Hsapiens.UCSC.hg38,
  chromosomes = paste0("chr", 1:22),
  window      = 200L,
  slide       = 200L    # no overlap
)
length(coords_nonoverlap)   # ~15 million windows across autosomes

## -- Custom start/end (e.g. a specific sub-region) --------------------
coords_sub <- make_genomiccoord(
  bsgenome    = BSgenome.Hsapiens.UCSC.hg38,
  chromosomes = "chr1",
  window      = 200L,
  slide       = 50L,
  start       = 1000000L,
  end         = 2000000L
```

```
)
length(coords_sub)   # 19,981 windows in 1 Mb region

## -- No N-trimming (use full chromosome length) ------------------------
coords_noN <- make_genomiccoord(
  bsgenome    = BSgenome.Hsapiens.UCSC.hg38,
  chromosomes = "chr1",
  window      = 200L,
  slide       = 50L,
  trim_N      = "none"
)

## -- Per-window N-filtering (removes windows with >10% N) -------------
coords_filt <- make_genomiccoord(
  bsgenome    = BSgenome.Hsapiens.UCSC.hg38,
  chromosomes = "chr1",
  window      = 200L,
  slide       = 50L,
  trim_N      = "filter",
  max_N_frac  = 0.10
)

## -- Get data.frame output for GRanges construction -------------------
df <- make_genomiccoord(
  bsgenome    = BSgenome.Hsapiens.UCSC.hg38,
  chromosomes = "chr1",
  window      = 200L,
  slide       = 50L,
  as_vector   = FALSE
)
gr <- GenomicRanges::GRanges(
  seqnames = df$chr,
  ranges   = IRanges::IRanges(start = df$win_start, end = df$win_end),
  strand   = df$strand
)

## -- Pass directly to tm_nn -------------------------------------------
coords <- make_genomiccoord(
  bsgenome    = BSgenome.Hsapiens.UCSC.hg38,
  chromosomes = "chr1",
  window      = 200L,
  slide       = 200L,
  start       = 1000000L,
  end         = 1010000L
)
tm_results <- tm_nn(coords, Na = 50)

## End(Not run)
```

---

plot_genome_track *Plot genome tracks in linear or circular layout*

Description

Visualise genomic tracks as either a linear karyoploteR plot or a circular plot (base R graphics). Set `circular = TRUE` to switch to the circular layout; the same `track_list` works for both views without modification.

Supported track types in each list element:

- type = "rect" : data as GRanges/data.frame, drawn as rectangles
- type = "line" : data as GRanges/data.frame with value_col, drawn as lines
- type = "area" : same as line but filled (linear only)
- type = "region": piled-up regions (linear only)
- type = "coverage": coverage area plot (linear only)

A track with `ideogram = TRUE` is drawn inside the chromosome bar (linear) or as the outermost ring with a grey background (circular). Only `"rect"` type is supported for ideogram tracks.

A per-track `height` field (numeric, default 1) controls relative sizing. In linear mode heights are proportional fractions of the data panel; in circular mode they scale the radial thickness of each ring.

A per-track `highlight` field (list or list-of-lists with data/col/alpha/border) draws highlight bands within that track only.

Entries with `type = "highlight"` draw translucent bands spanning *all* real tracks. These accept: data, col (default "#F1C40F"), alpha (default 0.18), border (default NA), and min.degree (circular only, default 0.4).

Each list element may also contain: name, col, bg.col, border, ylim, lwd, alpha (0–1 transparency), legend_font_col, ideogram, height, highlight.

Usage

```
plot_genome_track(
  genome_name,
  genome_size = NULL,
  genome = NULL,
  track_list,
  circular = FALSE,
  label = NULL,
  chromosomes = NULL,
  zoom = NULL,
  plot.type = 1,
  track.gap = 0.01,
  legend.show = TRUE,
  legend.position = NULL,
  legend.cex = 0.6,
  legend.bty = "n",
  legend.border = NA,
  legend.font.col = "black",
  legend.lwd = 1,
  legend.seg.len = -0.5,
  legend.box.lwd = 0.25,
```

```
  legend.x.intersp = 1,
  legend.lty = 1,
  axis.cex = NULL,
  title.cex = NULL,
  base.tick.dist = NULL,
  start.degree = 34.47,
  track.height = 0.05,
  gap.after = 0,
  cell.padding = c(0, 0, 0, 0),
  track.margin = c(0.005, 0.005),
  circle.margin = c(0.001, 0.001),
  canvas.xlim = c(-1, 1),
  canvas.ylim = c(-1, 1),
  axis.unit = "Mb",
  axis.step = NULL,
  axis.show.unit = FALSE,
  label.column = 4,
  label.niceFacing = TRUE,
  label.cex = 0.6,
  label.side = "inside",
  label.labels_height = 0.01,
  label.connection_height = 0.03,
  label.line_lwd = 0.25
)
```

Arguments

genome_name	Character. Used for the title.
genome_size	Numeric. Total genome length (single-chromosome genomes).
genome	Optional GRanges, or a karyoploteR genome string (e.g. "hg38"). Ignored in circular mode.
track_list	List of track specs (see above).
circular	Logical. If TRUE, render as a circular plot using base R graphics instead of a karyoploteR linear plot.
label	Optional data.frame/GRanges with labels.
chromosomes	Character vector to restrict/reorder chromosomes (linear only).
zoom	A GRanges object, a character string, or a character vector specifying one or more regions to zoom into (e.g. "chr1:1e6-2e6" or c("chr1:1e6-1.2e6", "chr1:3e6-3.2e6")). In linear mode each region is drawn as a separate stacked panel. In circular mode the regions are concatenated around the circle with small gaps between them. NULL (default) shows the full genome.
plot.type	karyoploteR plot.type (linear only).
track.gap	Relative gap between tracks (linear only, 0 to ~0.05).
legend.show	Logical.

`legend.position`
: Legend position. Character for linear (e.g. "topright"), numeric vector of length 2 for circular (e.g. c(0.75, 1)).

`legend.cex`, `legend.bty`, `legend.border`
: See `legend`.

`legend.font.col`
: Character. Default legend text colour.

`legend.lwd`, `legend.seg.len`, `legend.box.lwd`, `legend.x.intersp`, `legend.lty`
: Additional legend parameters.

`axis.cex`
: Axis label size.

`title.cex`
: Title size.

`base.tick.dist`
: Numeric or NULL (linear only).

`start.degree`
: Numeric. Starting angle in degrees for the circular layout (default 34.47, matching the circlize convention).

`track.height`, `gap.after`, `cell.padding`, `track.margin`
: Kept for backward compatibility; ignored in the base R circular layout.

`circle.margin`
: Numeric vector (length 2 or 4) controlling plot margins as fractions of the device size. Length 2 is recycled as `c(bottom, left, bottom, left)`. Default `c(0.001, 0.001)`.

`canvas.xlim`, `canvas.ylim`
: Numeric length-2 vectors controlling the plotting window (circular only). Adjust to pan or zoom into a portion of the circle.

`axis.unit`, `axis.step`, `axis.show.unit`
: Circular axis parameters.

`label.column`, `label.niceFacing`, `label.cex`, `label.side`, `label.labels_height`, `label.connection_height`, `label.line_lwd`
: Circular label parameters.

Value

Invisibly returns the KaryoPlot object (linear) or `NULL` (circular).

Examples

```
## Not run:
data(ecoli_rep_hotspots)

library("BSgenome.Ecoli.NCBI.ASM584v2")
genome_name <- "BSgenome.Ecoli.NCBI.ASM584v2"
chr_name    <- "U00096.3"
genome <- get(genome_name, envir = asNamespace(genome_name))
chr_length  <- length(genome[[chr_name]])
genome_name="BSgenome.Ecoli.NCBI.ASM584v2"
bins_gc <- make_genomiccoord(
  bsgenome    = genome_name,
  chromosomes = chr_name,
```

```
  window      = 200L,
  slide       = 200L,
  start       = 1,
  end         = chr_length,
  strand      = "+"
)
input_new <- list(pkg_name = genome_name, seq = bins_gc)
gr_batch <- to_genomic_ranges_fast(input_new)
tm_ASM584v2 <- tm_calculate(
  gr_batch,
  method   = "tm_nn"
)
Tm <- as.data.frame(tm_ASM584v2$gr[, c("Tm", "GC")])

tracks <- list(
  list(type = "rect", data = ecoli_rep_hotspots$all_peaks_IP_mutH,
       col = "#2C3E50", bg.col = "grey", name = "MutL-AR",
       legend_font_col = "#2C3E50", ideogram = TRUE, height = 0.5),
  list(type = "line", data = Tm, value_col = "GC",
       name = "GC content", col = "#4A90E2",
       legend_font_col = "#4A90E2"),
  list(type = "line", data = Tm, value_col = "Tm",
       name = "Melting temp", col = "#E06666",
       legend_font_col = "#E06666", height = 2),
  list(type = "line", data = ecoli_rep_hotspots$bins_rep,
       value_col = "count", name = "Microsatellites", col = "#2ECC71",
       legend_font_col = "#2ECC71"),
  list(type = "line", data = ecoli_rep_hotspots$bins_cru,
       value_col = "count", name = "Cruciform", col = "#3B3E6B",
       legend_font_col = "#3B3E6B"),
  list(data = ecoli_rep_hotspots$ssdna, name = "ssDNA",
       col = "#8E44AD", legend_font_col = "#8E44AD"),
  list(type = "line", data = ecoli_rep_hotspots$bins_gatc,
       value_col = "count", name = "GATC sites", col = "#D35400",
       legend_font_col = "#D35400"),
  list(type = "highlight", data = ecoli_rep_hotspots$all_peaks_IP_mutH,
       col = "#F1C40F", alpha = 0.18)
)

# Circular
plot_genome_track("E. coli", genome_size = 4641652,
                  track_list = tracks, circular = TRUE)

# Linear
plot_genome_track("E. coli", genome_size = 4641652,
                  track_list = tracks)

# Linear zoom (single region)
plot_genome_track("E. coli", genome_size = 4641652,
                  track_list = tracks,
                  zoom = "U00096.3:1000000-2000000")

# Linear zoom (multiple regions — stacked panels)
```

```
plot_genome_track("E. coli", genome_size = 4641652,
                  track_list = tracks,
                  zoom = c("U00096.3:1000000-1200000",
                           "U00096.3:3000000-3200000"))

# Circular zoom (multiple regions — concatenated arcs)
plot_genome_track("E. coli", genome_size = 4641652,
                  track_list = tracks, circular = TRUE,
                  zoom = c("U00096.3:1000000-1200000",
                           "U00096.3:3000000-3200000"))

# Circular with canvas panning
plot_genome_track("E. coli", genome_size = 4641652,
                  track_list = tracks, circular = TRUE,
                  canvas.xlim = c(0.5, 1), canvas.ylim = c(0, 1),
                  circle.margin = c(0.05, 0.05))

## End(Not run)
```

---

plot_tm *Compare Tm distributions across groups*

Description

Visualise how melting temperature (and optionally other numeric metadata) differs between region classes such as peak vs. non-peak, mutant vs. wild-type, or any categorical annotation stored in a GRanges object. Designed as the visual companion to compare_groups().

Usage

```
plot_tm(
  gr,
  group,
  value = "Tm",
  plot_type = c("box", "violin", "rank", "density", "ridgeline", "ecdf", "sina"),
  color_palette = c("viridis", "magma", "plasma", "inferno", "cividis"),
  show_points = FALSE,
  point_size = 1,
  point_alpha = 0.3,
  notch = FALSE,
  add_mean = FALSE,
  facet_by = NULL,
  show_pvalue = FALSE,
  p_adjust_method = "BH",
  title = NULL,
  ylab = "Tm (°C)",
  xlab = NULL,
  show_legend = TRUE
)
```

Arguments

gr	A GRanges object containing at least a numeric Tm metadata column and a categorical grouping column.
group	Character. Name of the metadata column that defines the groups to compare (e.g. "in_mutH").
value	Character. Numeric metadata column to plot on the y-axis. Default: "Tm". Can be any numeric column such as "GC".
plot_type	Character. Visualisation style: "box" Box plot (default). Shows median, quartiles, and outliers per group. "violin" Violin plot with embedded box plot. Shows the full density shape of each group. "rank" Rank plot. All values sorted ascending on the x-axis, coloured by group. Visually demonstrates whether one group clusters at higher or lower values — a graphical Wilcoxon test. "density" Overlaid density curves per group. "ridgeline" Stacked density ridges per group (requires **ggridges**). "ecdf" Empirical cumulative distribution function per group. Useful for visualising distributional shifts. "sina" Sina (strip) plot — jittered points shaped by the local density, combining the resolution of a strip chart with the density information of a violin plot.
color_palette	Character. Viridis palette for group colours: "viridis" (default), "magma", "plasma", "inferno", or "cividis".
show_points	Logical. Overlay jittered points on box / violin plots. Default: FALSE. Ignored for rank, density, ridgeline, ecdf.
point_size	Numeric. Point size. Default: 1.
point_alpha	Numeric in [0, 1]. Point transparency. Default: 0.3.
notch	Logical. Draw notched box plots (approximate 95% CI for median). Default: FALSE.
add_mean	Logical. Add a diamond marker at the group mean on box / violin plots. Default: FALSE.
facet_by	Character or NULL. Optional metadata column to facet the plot by (e.g. "chromosome"). Default: NULL.
show_pvalue	Logical. Annotate the plot with Wilcoxon rank-sum test p-values. For two groups a single p-value is shown; for three or more groups all pairwise comparisons are displayed. P-values are placed as bracket annotations on box / violin / sina plots, or as a subtitle on density / ecdf / rank / ridgeline plots. Default: FALSE.
p_adjust_method	Character. Method for p-value adjustment when there are more than two groups (passed to p.adjust). Default: "BH" (Benjamini-Hochberg).
title	Character or NULL. Plot title. A sensible default is generated when NULL.
ylab	Character. Y-axis label. Default: "Tm (\u00B0C)".

xlab	Character. X-axis label. Default: group column name for box/violin/sina, "Rank" for rank, value for density/ecdf.
show_legend	Logical. Show colour legend. Default: TRUE.

Details

The function pairs naturally with integrate_granges() and compare_groups():

1. Use integrate_granges() to annotate Tm windows with feature overlaps (e.g. ChIP-seq peaks, repeat classes).
2. Use compare_groups() for the statistical test.
3. Use plot_tm() to visualise the distributional difference.

Value

A ggplot object.

Examples

```
## Not run:
library(GenomicRanges)
data(ecoli_rep_hotspots)

library("BSgenome.Ecoli.NCBI.ASM584v2")
genome_name <- "BSgenome.Ecoli.NCBI.ASM584v2"
chr_name    <- "U00096.3"
genome <- get(genome_name, envir = asNamespace(genome_name))
chr_length  <- length(genome[[chr_name]])
genome_name="BSgenome.Ecoli.NCBI.ASM584v2"
bins_gc <- make_genomiccoord(
  bsgenome    = genome_name,
  chromosomes = chr_name,
  window      = 200L,
  slide       = 200L,
  start       = 1,
  end         = chr_length,
  strand      = "+"
)
input_new <- list(pkg_name = genome_name, seq = bins_gc)
gr_batch <- to_genomic_ranges_fast(input_new)
tm_ASM584v2 <- tm_calculate(
  gr_batch,
  method   = "tm_nn"
)
gr_tm <- tm_ASM584v2$gr

# Annotate with MutL-AR peak membership
mutH_peaks <- GRanges(
  seqnames = ecoli_rep_hotspots$all_peaks_IP_mutH$chr,
  ranges   = IRanges(start = ecoli_rep_hotspots$all_peaks_IP_mutH$start,
                     end   = ecoli_rep_hotspots$all_peaks_IP_mutH$end)
```

```
)
mutH_peaks$peak_id <- paste0("mutH_", seq_along(mutH_peaks))
gr_annot <- integrate_granges(
  gr_tm = gr_tm, gr_features = mutH_peaks,
  strategy = "overlap", feature_cols = "peak_id",
  keep_unmatched = TRUE
)
gr_annot$in_mutH <- ifelse(is.na(gr_annot$peak_id), "non_peak", "peak")

# Box plot — peak vs non-peak Tm
plot_tm(gr_annot, group = "in_mutH")

# Box plot with Wilcoxon p-value bracket
plot_tm(gr_annot, group = "in_mutH", show_pvalue = TRUE)

# Violin plot with p-value and jittered points
plot_tm(gr_annot, group = "in_mutH", plot_type = "violin",
        show_points = TRUE, show_pvalue = TRUE)

# Rank plot — visual Wilcoxon test with p-value subtitle
plot_tm(gr_annot, group = "in_mutH", plot_type = "rank",
        show_pvalue = TRUE)

# Density overlay with p-value
plot_tm(gr_annot, group = "in_mutH", plot_type = "density",
        show_pvalue = TRUE)

# ECDF — cumulative distribution comparison
plot_tm(gr_annot, group = "in_mutH", plot_type = "ecdf",
        show_pvalue = TRUE)

# Compare GC content instead of Tm
plot_tm(gr_annot, group = "in_mutH", value = "GC", ylab = "GC content",
        show_pvalue = TRUE)

# Notched box plot with mean markers and p-value
plot_tm(gr_annot, group = "in_mutH", notch = TRUE, add_mean = TRUE,
        show_pvalue = TRUE)

## End(Not run)
```

`print.TmCalculator` *Prints melting temperature from a* `TmCalculator` *object*

Description

`print.TmCalculator` prints to console the melting temperature value from an object of class `TmCalculator`.

Usage

```
## S3 method for class 'TmCalculator'
print(x, ...)
```

Arguments

x	An object of class TmCalculator.
...	Unused

Value

The melting temperature value.

---

salt_correct *Corrections of melting temperature with salt concentration*

Description

Apply corrections to melting temperature calculations based on salt concentrations. Different correction methods are available for various experimental conditions.

Usage

```
salt_correct(
  Na = 0,
  K = 0,
  Tris = 0,
  Mg = 0,
  dNTPs = 0,
 method = c("Schildkraut2010", "Wetmur1991", "SantaLucia1996", "SantaLucia1998-1",
    "SantaLucia1998-2", "Owczarzy2004", "Owczarzy2008"),
  input_seq,
  ambiguous = FALSE
)
```

Arguments

Na	Millimolar concentration of sodium ions. Default: 0
K	Millimolar concentration of potassium ions. Default: 0
Tris	Millimolar concentration of Tris buffer. Default: 0
Mg	Millimolar concentration of magnesium ions. Default: 0
dNTPs	Millimolar concentration of deoxynucleotide triphosphates. Default: 0

method	Method for calculating salt concentration corrections to the melting temperature. Available options: - "Schildkraut2010": Updated salt correction method - "Wetmur1991": Classic salt correction method - "SantaLucia1996": DNA-specific salt correction - "SantaLucia1998-1": Improved DNA salt correction - "SantaLucia1998-2": Alternative DNA salt correction (requires input_seq) - "Owczarzy2004": Comprehensive salt correction (requires input_seq) - "Owczarzy2008": Updated comprehensive salt correction (requires input_seq) Note: Setting to NA disables salt correction
input_seq	Sequence (5' to 3') of one strand of the nucleic acid duplex. Can be provided as either: - A character string (e.g., "ATGCG") - A path to a FASTA file containing the sequence(s) Required for methods: "SantaLucia1998-2", "Owczarzy2004", and "Owczarzy2008"
ambiguous	Logical. If TRUE, ambiguous bases are taken into account when computing the G and C content. The function handles various ambiguous bases (S, W, M, K, R, Y, V, H, D, B) by proportionally distributing their contribution to GC content based on their possible nucleotide compositions.

Details

Different correction methods are available for various experimental conditions:

- Schildkraut2010: Updated salt correction method that accounts for monovalent and divalent cations - Wetmur1991: Classic salt correction method for monovalent cations - SantaLucia1996: DNA-specific salt correction - SantaLucia1998-1: Improved DNA salt correction - SantaLucia1998-2: Alternative DNA salt correction (requires sequence information) - Owczarzy2004: Comprehensive salt correction including effects of divalent cations (requires sequence information) - Owczarzy2008: Updated comprehensive salt correction (requires sequence information)

Author(s)

Junhui Li

References

Schildkraut C, Lifson S. Dependence of the melting temperature of DNA on salt concentration. Biopolymers. 1965;3(2):195-208.

Wetmur JG. DNA Probes: Applications of the Principles of Nucleic Acid Hybridization. Critical Reviews in Biochemistry and Molecular Biology. 1991;26(3-4):227-259.

SantaLucia J. A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics. Proceedings of the National Academy of Sciences. 1998;95(4):1460-1465.

Owczarzy R, Moreira BG, Manthey JA, et al. Predicting stability of DNA duplexes in solutions containing magnesium and monovalent cations. Biochemistry. 2008;47(19):5336-5353.

Examples

```
salt_correct(Na = 50, Mg = 1.5, method = "Owczarzy2008",
               input_seq = "ATGCGATGCG")
```

thermodynamic_gc_params

Thermodynamic parameters for GC-based Tm calculation methods

Description

A data frame containing coefficients and parameters for different GC-based Tm calculation methods. Each row represents a different method with its specific coefficients (A, B, C, D) and salt correction method.

Usage

```
thermodynamic_gc_params
```

Format

A data frame with 8 rows and 5 columns:

A Intercept coefficient

B GC content coefficient

C Length correction coefficient

D Mismatch coefficient

salt_correct Associated salt correction method

Details

The methods included are: - Chester1993: Tm = 69.3 + 0.41(Percentage_GC) - 650/N - QuikChange: Tm = 81.5 + 0.41(Percentage_GC) - 675/N - Percentage_mismatch - Schildkraut1965: Tm = 81.5 + 0.41(Percentage_GC) - 675/N + 16.6 x log10[Na+] - Wetmur1991_MELTING: Tm = 81.5 + 0.41(Percentage_GC) - 500/N + Wetmur salt - - Wetmur1991_RNA: Tm = 78 + 0.7(Percentage_GC) - 500/N + Wetmur salt - - Wetmur1991_RNA/DNA: Tm = 67 + 0.8(Percentage_GC) - 500/N + Wetmur salt - - Primer3Plus: Tm = 81.5 + 0.41(Percentage_GC) - 600/N + 16.6 x log10[Na+] - vonAhsen2001: Tm = 77.1 + 0.41(Percentage_GC) - 528/N + 11.7 x log10[Na+]

thermodynamic_nn_params

Thermodynamic Tables for Nucleic Acid Hybridization

Description

A comprehensive collection of thermodynamic parameters used for calculating melting temperatures of nucleic acid duplexes. The dataset includes parameters for DNA/DNA, RNA/RNA, RNA/DNA and 2'-O-methylRNA/RNA hybridizations, plus parameters for mismatches, dangling ends, internal / bulge loops, GU wobble, CNG repeats, inosine, hydroxyadenine, azobenzene and locked nucleic acids (LNA).

Usage

```
thermodynamic_nn_params
```

Format

A named list of two-column matrices with `colnames = c("left", "right")` (left = ΔH, right = ΔS). Rownames are dinucleotide step keys such as `"AT/TA"`, mismatch / dangling-end keys, or feature-specific keys (e.g. `"CAG_4"` for the (CAG)4 CNG entry).

Populated tables

DNA_NN_Breslauer_1986 DNA/DNA NN, Breslauer et al. (1986)

DNA_NN_Sugimoto_1996 DNA/DNA NN, Sugimoto et al. (1996)

DNA_NN_Allawi_1998 DNA/DNA NN, Allawi (1998)

DNA_NN_SantaLucia_2004 DNA/DNA NN, SantaLucia & Hicks (2004)

RNA_NN_Freier_1986 RNA/RNA NN, Freier (1986)

RNA_NN_Xia_1998 RNA/RNA NN, Xia (1998)

RNA_NN_Chen_2012 RNA/RNA NN with GU, Chen / Serra (2012)

RNA_DNA_NN_Sugimoto_1995 RNA/DNA hybrid NN, Sugimoto (1995)

DNA_IMM_Peyret_1999 DNA single internal mismatch, Peyret (1999)

DNA_TMM_Bommarito_2000 DNA terminal mismatch, Bommarito (2000)

DNA_DE_Bommarito_2000 DNA single dangling end, Bommarito (2000)

RNA_DE_Turner_2010 RNA single dangling end, Turner (2010)

Registered placeholders (values TBD - populate from primary literature)

DNA_NN_AllSan_1997 Allawi & SantaLucia (1997) Biochemistry 36:10581

DNA_NN_SantaLucia_1996 SantaLucia et al. (1996) Biochemistry 35:3555

DNA_NN_Tanaka_2004 Tanaka et al. (2004) Biochemistry 43:7143

MeRNA_RNA_NN_Kierzek_2006 Kierzek et al. (2006) Biochemistry 45:581

DNA_RNA_IMM_Watkins_2011 Watkins et al. (2011) Nucleic Acids Res 39:1894

RNA_IMM_Lu_2006 Lu et al. (2006) Nucleic Acids Res 34:4912

RNA_IMM_Davis_Znosko_2007 Davis & Znosko (2007) Biochemistry 46:13425

RNA_IMM_Davis_Znosko_2008 Davis & Znosko (2008) Biochemistry 47:10178

DNA_TandemMM_AllSanPey Allawi & SantaLucia (1997/1998); Peyret (1999)

RNA_TandemMM_Mathews_1999 Mathews et al. (1999) JMB 288:911

DNA_DE_Ohmichi_2002 Ohmichi et al. (2002) JACS 124:10367

RNA_DE_Ohmichi_2002 Ohmichi et al. (2002) JACS 124:10367

RNA_DE_Serra_2008 O'Toole / Serra (2006, 2008)

DNA_DDE_Ohmichi_2002 Ohmichi et al. (2002) JACS 124:10367

RNA_DDE_Ohmichi_2002 Ohmichi et al. (2002) JACS 124:10367

RNA_DDE_OToole_2005 O'Toole et al. (2005) Biochemistry 44:14914

RNA_DDE_OToole_2006 O'Toole et al. (2006) NAR 34:3338

DNA_LDE_Ohmichi_2002 Ohmichi et al. (2002) JACS 124:10367

RNA_LDE_Ohmichi_2002 Ohmichi et al. (2002) JACS 124:10367

DNA_IL_SantaLucia_2004 SantaLucia & Hicks (2004)

RNA_IL_Lu_2006 Lu et al. (2006) NAR 34:4912

RNA_IL_Badhwar_2007 Badhwar et al. (2007) Biochemistry 46:14715

DNA_SBL_Tanaka_2007 Tan & Chen (2007) Biophys J 92:3615

DNA_SBL_SantaLucia_2004 SantaLucia & Hicks (2004)

RNA_SBL_Lu_2006 Lu et al. (2006) NAR 34:4912

RNA_SBL_Blose_2007 Blose et al. (2007) Biochemistry 46:15123

DNA_LBL_SantaLucia_2004 SantaLucia & Hicks (2004)

RNA_LBL_Lu_2006 Mathews (1999); Lu et al. (2006)

RNA_GU_Mathews_1999 Mathews & Turner (1999) JMB 288:911

RNA_GU_Chen_2012 Chen / Serra (2012)

DNA_CNG_Broda_2005 Broda et al. (2005) Biochemistry 44:13851 - rownames use `paste0("C", N, "G_", n)`, e.g. `"CAG_4"`.

DNA_INO_SantaLucia_2005 Watkins & SantaLucia (2005) NAR 33:6258

RNA_INO_Wright_2007 Wright et al. (2007) Biochemistry 46:4625

DNA_HA_Kawakami_2001 Kawakami / Sugimoto (2001) Biochemistry 40:14040

DNA_AZB_Asanuma_2005 Asanuma et al. (2005) Angew Chem Int Ed 44:2747

DNA_LNA_Owczarzy_2011 Owczarzy et al. (2011) Biochemistry 50:9352

DNA_LNA_McTigue_2004 McTigue et al. (2004) Biochemistry 43:5388

DNA_cLNA_Owczarzy_2011 Owczarzy et al. (2011) Biochemistry 50:9352

DNA_cLNA_MM_Owczarzy_2011 Owczarzy et al. (2011) Biochemistry 50:9352

To add the numeric values for any placeholder table, construct a matrix with two columns (ΔH kcal/mol, ΔS cal/(mol*K)) and the dinucleotide-step rownames described above, then assign it:

```
thermodynamic_nn_params$DNA_CNG_Broda_2005 <- new_table
```

Details

Coverage mirrors the rmelting (Bioconductor) vignette sections 4.2.1 - 4.4. Tables whose numeric values are not yet ported from primary literature are included in the list as `NULL` placeholders so that `tm_nn` can still dispatch on the table name; the corresponding contribution is silently skipped after a one-time package-startup notice.

Source

Various publications as cited above.

References

Breslauer K J (1986) <doi:10.1073/pnas.83.11.3746> Sugimoto N (1996) <doi:10.1093/nar/24.22.4501> Allawi H (1998) <doi:10.1093/nar/26.11.2694> SantaLucia J (2004) <doi:10.1146/annurev.biophys.32.110601.141800> Freier S (1986) <doi:10.1073/pnas.83.24.9373> Xia T (1998) <doi:10.1021/bi9809425> Chen JL (2012) <doi:10.1021/bi3002709> Sugimoto N (1995) <doi:10.1016/S0048-9697(98)00088-6> Bommarito S (2000) <doi:10.1093/nar/28.9.1929> Peyret N (1999) <doi:10.1021/bi9825091> Allawi H T & SantaLucia J (1997) <doi:10.1021/bi962590c> SantaLucia J (2005) <doi:10.1093/nar/gki918> Turner D H (2010) <doi:10.1093/nar/gkp892> Tanaka F (2004) Biochemistry 43:7143 Kierzek E (2006) Biochemistry 45:581 Watkins N E (2011) Nucleic Acids Res 39:1894 Lu Z J (2006) Nucleic Acids Res 34:4912 Davis A R & Znosko B M (2007, 2008) Biochemistry Mathews D H (1999) <doi:10.1006/jmbi.1999.2700> Ohmichi T (2002) J Am Chem Soc 124:10367 O'Toole A S (2005, 2006) Biochemistry / Nucleic Acids Res Badhwar J (2007) Biochemistry 46:14715 Tan Z J & Chen S J (2007) Biophys J 92:3615 Blose J M (2007) Biochemistry 46:15123 Broda M (2005) <doi:10.1021/bi0501447> Wright D J (2007) Biochemistry 46:4625 Kawakami J / Sugimoto N (2001) <doi:10.1021/bi010918b> Asanuma H (2005) Angew Chem Int Ed 44:2747 McTigue P M (2004) Biochemistry 43:5388 Owczarzy R (2011) Biochemistry 50:9352

Examples

```
# Access DNA/DNA nearest neighbor parameters
thermodynamic_nn_params$DNA_NN_SantaLucia_2004

# Access DNA internal mismatch parameters
thermodynamic_nn_params$DNA_IMM_Peyret_1999

# See which tables are still placeholders (NULL)
names(Filter(is.null, thermodynamic_nn_params))
```

`tm_calculate` *Calculate melting temperature using multiple methods*

Description

Calculates melting temperature using multiple methods: - Nearest Neighbor thermodynamics (tm_nn) - GC content-based method (tm_gc) - Wallace rule (tm_wallace)

Usage

```
tm_calculate(
  input_seq,
  method = c("tm_nn", "tm_gc", "tm_wallace"),
  complement_seq = NULL,
  ambiguous = FALSE,
  shift = 0,
 nn_table = c("DNA_NN_SantaLucia_2004", "DNA_NN_Breslauer_1986", "DNA_NN_Sugimoto_1996",
  "DNA_NN_Allawi_1998", "RNA_NN_Freier_1986", "RNA_NN_Xia_1998", "RNA_NN_Chen_2012",
   "RNA_DNA_NN_Sugimoto_1995"),
```

```
  tmm_table = "DNA_TMM_Bommarito_2000",
  imm_table = "DNA_IMM_Peyret_1999",
  de_table = c("DNA_DE_Bommarito_2000", "RNA_DE_Turner_2010"),
  dnac_high = 25,
  dnac_low = 25,
  self_comp = FALSE,
  variant = c("Primer3Plus", "Chester1993", "QuikChange", "Schildkraut1965",
   "Wetmur1991_MELTING", "Wetmur1991_RNA", "Wetmur1991_RNA/DNA", "vonAhsen2001"),
  userset = NULL,
  Na = 50,
  K = 0,
  Tris = 0,
  Mg = 0,
  dNTPs = 0,
 salt_method = c("Schildkraut2010", "Wetmur1991", "SantaLucia1996", "SantaLucia1998-1",
    "Owczarzy2004", "Owczarzy2008"),
  DMSO = 0,
  formamide_unit = list(value = 0, unit = "percent"),
  dmso_factor = 0.75,
  formamide_factor = 0.65,
  mismatch = TRUE
)
```

Arguments

input_seq	Input sequence(s) in 5' to 3' direction. Can be provided as either: - A character string (e.g., "ATGCG") - A path to a FASTA file containing the sequence(s) - A GRanges object with sequence and complement metadata should be provided if mismatch is TRUE - A character vector where each element is a string in the format "chr:start-end:strand:species" (e.g., "chr1:100-200:+:BSgenome.Hsapiens.UCSC.hg38"). Strand is "+" for positive (default if not provided) or "-" for negative. - chr: Chromosome ID - start: Start position - end: End position - strand: positive or negtive strand - species: Species name for reference genome (e.g., "BSgenome.Hsapiens.UCSC.hg38"), see `BSgenome::available.genomes()` for all available genomes. please make sure the genome package is installed, otherwise the function will stop.
method	Method(s) to use for Tm calculation. Can be one or more of: - "tm_nn": Nearest Neighbor thermodynamics (default) - "tm_gc": GC content-based method - "tm_wallace": Wallace rule Default: c("tm_nn", "tm_gc", "tm_wallace")
complement_seq	Complementary sequence(s) in 3' to 5' direction. If not provided, the function will automatically generate it from input_seq. This is the template/target sequence that the input sequence will hybridize with. Can be provided as input_seq format besides A NULL value(default)
ambiguous	Logical. If TRUE, ambiguous bases are taken into account when computing the G and C content. The function handles various ambiguous bases (S, W, M, K, R, Y, V, H, D, B) by proportionally distributing their contribution to GC content based on their possible nucleotide compositions. Default: FALSE
shift	Integer value controlling the alignment offset between primer and template sequences. Only applicable for the NN method. Default: 0

`nn_table` Thermodynamic nearest-neighbor parameters for different nucleic acid hybridizations. Only applicable for the NN method. Default: "DNA_NN_SantaLucia_2004"

`tmm_table` Thermodynamic parameters for terminal mismatches. Only applicable for the NN method. Default: "DNA_TMM_Bommarito_2000"

`imm_table` Thermodynamic parameters for internal mismatches. Only applicable for the NN method. Default: "DNA_IMM_Peyret_1999"

`de_table` Thermodynamic parameters for dangling ends. Only applicable for the NN method. Default: "DNA_DE_Bommarito_2000"

`dnac_high` Concentration of the higher concentrated strand in nM. Only applicable for the NN method. Default: 25

`dnac_low` Concentration of the lower concentrated strand in nM. Only applicable for the NN method. Default: 25

`self_comp` Logical value indicating if the sequence is self-complementary. Only applicable for the NN method. Default: FALSE

`variant` Empirical constants coefficient for GC method. Only applicable for the GC method. Default: "Primer3Plus"

`userset` A vector of four coefficient values for GC method. Only applicable for the GC method. Usersets override value sets. Default: NULL

`Na` Millimolar concentration of sodium ions. Default: 50

`K` Millimolar concentration of potassium ions. Default: 0

`Tris` Millimolar concentration of Tris buffer. Default: 0

`Mg` Millimolar concentration of magnesium ions. Default: 0

`dNTPs` Millimolar concentration of deoxynucleotide triphosphates. Default: 0

`salt_method` Salt correction method for Tm. Default: "Schildkraut2010" Available options: - "Schildkraut2010": Updated salt correction method - "Wetmur1991": Classic salt correction method - "SantaLucia1996": DNA-specific salt correction - "SantaLucia1998-1": Improved DNA salt correction - "SantaLucia1998-2": Alternative DNA salt correction - "Owczarzy2004": Comprehensive salt correction - "Owczarzy2008": Updated comprehensive salt correction Default: "Schildkraut2010"

`DMSO` Percent DMSO concentration in the reaction mixture. Default: 0

`formamide_unit` Formamide concentration as ‘list(value, unit)‘. Default: list(value = 0, unit = "percent") - value: Numeric value of formamide concentration - unit: Either "percent" or "molar"

`dmso_factor` Coefficient of Tm decreases per percent DMSO. Default: 0.75 Other published values are 0.5, 0.6 and 0.675.

`formamide_factor` Tm decrease per percent formamide. Default: 0.65 Several papers report factors between 0.6 and 0.72.

`mismatch` Logical. If TRUE, every '.' in the sequence is counted as a mismatch. Only applicable for the GC method. Default: TRUE

Details

The function calculates melting temperature using the specified method(s). For each method: - NN: Uses nearest neighbor thermodynamics with detailed sequence analysis - GC: Uses GC content-based calculation with various empirical formulas - Wallace: Uses the simple Wallace rule (2deg C per A/T, 4deg C per G/C)

The function processes the input sequence once and applies it to all selected methods, making it more efficient than calling each method separately.

Value

A `TmCalculator` list with:

`gr`	The input `GRanges` with metadata columns `Tm` and `GC` (melting temperature in °C and GC percent).
`options`	Calculation parameters and method information.

Available Options

Method Selection:

- `method`: c("tm_nn", "tm_gc", "tm_wallace")

Nearest Neighbor (NN) Method Options:

- `nn_table`:
  - "DNA_NN_Breslauer_1986"
  - "DNA_NN_Sugimoto_1996"
  - "DNA_NN_Allawi_1998"
  - "DNA_NN_SantaLucia_2004" (default)
  - "RNA_NN_Freier_1986"
  - "RNA_NN_Xia_1998"
  - "RNA_NN_Chen_2012"
  - "RNA_DNA_NN_Sugimoto_1995"
- `tmm_table` (Terminal Mismatches):
  - "DNA_TMM_Bommarito_2000" (default)
- `imm_table` (Internal Mismatches):
  - "DNA_IMM_Peyret_1999" (default)
- `de_table` (Dangling Ends):
  - "DNA_DE_Bommarito_2000" (default)
  - "RNA_DE_Turner_2010"

GC Method Options:

- `variant`:
  - "Primer3Plus" (default)
  - "Chester1993"

  - "QuikChange"
  - "Schildkraut1965"
  - "Wetmur1991_MELTING"
  - "Wetmur1991_RNA"
  - "Wetmur1991_RNA/DNA"
  - "vonAhsen2001"

Salt Correction Options:

- `salt_method`:
  - "Schildkraut2010" (default)
  - "Wetmur1991"
  - "SantaLucia1996"
  - "SantaLucia1998-1"
  - "SantaLucia1998-2"
  - "Owczarzy2004"
  - "Owczarzy2008"

Formamide Unit Options:

- `formamide_unit$unit`:
  - "percent" (default)
  - "molar"

Other Parameters:

- `ambiguous`: TRUE/FALSE (default: FALSE)
- `shift`: Integer value (default: 0)
- `dnac_high`: Numeric value in nM (default: 25)
- `dnac_low`: Numeric value in nM (default: 25)
- `self_comp`: TRUE/FALSE (default: FALSE)
- `Na`: Millimolar concentration (default: 50)
- `K`: Millimolar concentration (default: 0)
- `Tris`: Millimolar concentration (default: 0)
- `Mg`: Millimolar concentration (default: 0)
- `dNTPs`: Millimolar concentration (default: 0)
- `DMSO`: Percent concentration (default: 0)
- `dmso_factor`: Numeric value (default: 0.75)
- `formamide_factor`: Numeric value (default: 0.65)
- `mismatch`: TRUE/FALSE (default: TRUE)

Author(s)

Junhui Li

Examples

```
## Not run:
input_seq <- c("chr1:1000100-1000150:+:BSgenome.Hsapiens.UCSC.hg38")
result <- tm_calculate(
  input_seq,
  method = "tm_nn",
  nn_table = "DNA_NN_SantaLucia_2004",
  salt_method = "Owczarzy2008"
)

## End(Not run)
```

`tm_gc` *Calculate the melting temperature using empirical formulas based on GC content*

Description

Calculate the melting temperature using empirical formulas based on GC content with different options. The function returns a list of sequences with updated Tm attributes and calculation options.

Usage

```
tm_gc(
  gr_seq,
  ambiguous = FALSE,
  userset = NULL,
  variant = c("Primer3Plus", "Chester1993", "QuikChange", "Schildkraut1965",
   "Wetmur1991_MELTING", "Wetmur1991_RNA", "Wetmur1991_RNA/DNA", "vonAhsen2001"),
  Na = 50,
  K = 0,
  Tris = 0,
  Mg = 0,
  dNTPs = 0,
 salt_method = c("Schildkraut2010", "Wetmur1991", "SantaLucia1996", "SantaLucia1998-1",
    "Owczarzy2004", "Owczarzy2008"),
  mismatch = TRUE,
  DMSO = 0,
  formamide_unit = list(value = 0, unit = "percent"),
  dmso_factor = 0.75,
  formamide_factor = 0.65
)
```

Arguments

`gr_seq`	Pre-processed sequence(s) in 5' to 3' direction. This should be the output from to_genomic_ranges() function.

ambiguous	Logical. If TRUE, ambiguous bases are taken into account when computing the G and C content. The function handles various ambiguous bases (S, W, M, K, R, Y, V, H, D, B) by proportionally distributing their contribution to GC content based on their possible nucleotide compositions.
userset	A vector of four coefficient values. Usersets override value sets.
variant	Empirical constants coefficient with 8 variants: - Chester1993: Tm = 69.3 + 0.41(Percentage_GC) - 650/N - QuikChange: Tm = 81.5 + 0.41(Percentage_GC) - 675/N - Percentage_mismatch - Schildkraut1965: Tm = 81.5 + 0.41(- Wetmur1991_MELTING: Tm = 81.5 + 0.41(- Wetmur1991_RNA: Tm = 78 + 0.7(- Wetmur1991_RNA/DNA: Tm = 67 + 0.8(- Primer3Plus: Tm = 81.5 + 0.41(- vonAhsen2001: Tm = 77.1 + 0.41(Salt correction is applied only for variants that include it in the formula (via `salt_correct()`). Chester1993 and QuikChange use no salt term. D is the mismatch penalty (typically 1): Tm decreases by D x (Use `X` (or `.`) in the sequence to mark mismatch positions.
Na	Millimolar concentration of sodium ions. Default: 50
K	Millimolar concentration of potassium ions. Default: 0
Tris	Millimolar concentration of Tris buffer. Default: 0
Mg	Millimolar concentration of magnesium ions. Default: 0
dNTPs	Millimolar concentration of deoxynucleotide triphosphates. Default: 0
salt_method	Salt correction method. `NULL` (default) uses the method associated with `variant`. Set to `NA` to disable salt correction. Options: - "Schildkraut2010": Schildkraut & Lifson 1965 - "Wetmur1991": Wetmur 1991 - "SantaLucia1996": SantaLucia 1996 - "SantaLucia1998-1": SantaLucia 1998 (Method 1) - "Owczarzy2004": Owczarzy 2004 - "Owczarzy2008": Owczarzy 2008 Note: "SantaLucia1998-2" is not available for this function.
mismatch	Logical. If TRUE (default), every 'X' in the sequence is counted as a mismatch
DMSO	Percent DMSO concentration in the reaction mixture. Default: 0
formamide_unit	Formamide concentration as 'list(value, unit)'. Default: list(value = 0, unit = "percent") - value: Numeric value of formamide concentration - unit: Either "percent" or "molar"
dmso_factor	Coefficient of Tm decreases per percent DMSO. Default: 0.75 (von Ahsen et al. 2001) Other published values are 0.5, 0.6 and 0.675.
formamide_factor	Coefficient of Tm decrease per percent formamide. Default: 0.65 Several papers report factors between 0.6 and 0.72.

Value

Returns a list with two components: - Tm: A list of sequences with updated Tm attributes - Options: A list containing calculation parameters and method information

Author(s)

Junhui Li

References

Marmur J, Doty P. Determination of the base composition of deoxyribonucleic acid from its thermal denaturation temperature. Journal of Molecular Biology, 1962, 5(1):109-118.

Schildkraut C. Dependence of the melting temperature of DNA on salt concentration. Biopolymers, 2010, 3(2):195-208.

Wetmur JG. DNA Probes: Applications of the Principles of Nucleic Acid Hybridization. CRC Critical Reviews in Biochemistry, 1991, 26(3-4):33.

Untergasser A, Cutcutache I, Koressaar T, et al. Primer3–new capabilities and interfaces. Nucleic Acids Research, 2012, 40(15):e115-e115.

von Ahsen N, Wittwer CT, Schutz E, et al. Oligonucleotide melting temperatures under PCR conditions: deoxynucleotide Triphosphate and Dimethyl sulfoxide concentrations with comparison to alternative empirical formulas. Clin Chem 2001, 47:1956-1961.

Examples

```
# Example with multiple sequences
input_seq <- c("ATCGTGCGTAGCAGTACGATCAGTAG", "ATCGTGCGTAGCAGTACGATCAGTAG")
gr_seq <- to_genomic_ranges(input_seq)
out <- tm_gc(gr_seq, ambiguous = TRUE, variant = "Primer3Plus", Na = 50, mismatch = TRUE)
out
out$Options
```

`tm_nn` *Calculate melting temperature using nearest neighbor thermodynamics*

Description

Calculate melting temperature using nearest neighbor thermodynamics. The function checks if all sequence combinations in the input sequence are present in the thermodynamic parameter tables before performing calculations.

Usage

```
tm_nn(
  gr_seq,
  ambiguous = FALSE,
  shift = 0,
 nn_table = c("DNA_NN_SantaLucia_2004", "DNA_NN_Breslauer_1986", "DNA_NN_Sugimoto_1996",
   "DNA_NN_Allawi_1998", "RNA_NN_Freier_1986", "RNA_NN_Xia_1998", "RNA_NN_Chen_2012",
     "RNA_DNA_NN_Sugimoto_1995"),
  tmm_table = "DNA_TMM_Bommarito_2000",
  imm_table = "DNA_IMM_Peyret_1999",
  de_table = c("DNA_DE_Bommarito_2000", "RNA_DE_Turner_2010"),
  dnac_high = 25,
```

```
  dnac_low = 25,
  self_comp = FALSE,
  Na = 50,
  K = 0,
  Tris = 0,
  Mg = 0,
  dNTPs = 0,
 salt_method = c("Schildkraut2010", "Wetmur1991", "SantaLucia1996", "SantaLucia1998-1",
    "Owczarzy2004", "Owczarzy2008"),
  DMSO = 0,
  formamide_unit = list(value = 0, unit = "percent"),
  dmso_factor = 0.75,
  formamide_factor = 0.65
)
```

Arguments

gr_seq	Pre-processed sequence(s) in 5' to 3' direction. This should be the output from to_genomic_ranges() function.
ambiguous	Logical value controlling how ambiguous bases are handled: - TRUE: Ambiguous bases (e.g., N, R, Y) are included in calculations - FALSE (default): Ambiguous bases are excluded from calculations
shift	Integer value controlling the alignment offset between primer and template sequences. Visual representation of different shift values: shift = 0 (default): Primer: 5' ATGCG 3' Template: 3' TACGC 5' shift = -1: Primer: 5' ATGCG 3' Template: 3' TACGC 5' ^ shift = 1: Primer: 5' ATGCG 3' Template: 3' TACGC 5' ^ The shift parameter is necessary when: - Sequences have different lengths - Dangling ends are required - Specific alignment positions are needed
nn_table	Thermodynamic nearest-neighbor parameters for different nucleic acid hybridizations. Eight parameter sets are available, organized by hybridization type: DNA/DNA hybridizations: - "DNA_NN_Breslauer_1986": Original DNA/DNA parameters - "DNA_NN_Sugimoto_1996": Improved DNA/DNA parameters - "DNA_NN_Allawi_1998": DNA/DNA parameters with internal mismatch corrections - "DNA_NN_SantaLucia_2004": Updated DNA/DNA parameters RNA/RNA hybridizations: - "RNA_NN_Freier_1986": Original RNA/RNA parameters - "RNA_NN_Xia_1998": Improved RNA/RNA parameters - "RNA_NN_Chen_2012": Updated RNA/RNA parameters with GU pair corrections RNA/DNA hybridizations: - "RNA_DNA_NN_Sugimoto_1995": RNA/DNA hybridization parameters
tmm_table	Thermodynamic parameters for terminal mismatches. Default: "DNA_TMM_Bommarito_2000" These parameters account for mismatches at the ends of the duplex.
imm_table	Thermodynamic parameters for internal mismatches. Default: "DNA_IMM_Peyret_1999" These parameters account for mismatches within the duplex, including inosine mismatches.

de_table	Thermodynamic parameters for dangling ends. Default: "DNA_DE_Bommarito_2000" Available options: - "DNA_DE_Bommarito_2000": Parameters for DNA dangling ends - "RNA_DE_Turner_2010": Parameters for RNA dangling ends
dnac_high	Concentration of the higher concentrated strand in nM. Default: 25 Typically this is the primer (for PCR) or the probe concentration.
dnac_low	Concentration of the lower concentrated strand in nM. Default: 25 This is typically the template concentration.
self_comp	Logical value indicating if the sequence is self-complementary: - TRUE: Sequence can bind to itself, dnac_low is ignored - FALSE (default): Sequence binds to a different complementary sequence
Na	Millimolar concentration of sodium ions. Default: 50
K	Millimolar concentration of potassium ions. Default: 0
Tris	Millimolar concentration of Tris buffer. Default: 0
Mg	Millimolar concentration of magnesium ions. Default: 0
dNTPs	Millimolar concentration of deoxynucleotide triphosphates. Default: 0
salt_method	Salt correction method. Options are: Available options: - "Schildkraut2010": Updated salt correction method - "Wetmur1991": Classic salt correction method - "SantaLucia1996": DNA-specific salt correction - "SantaLucia1998-1": Improved DNA salt correction - "SantaLucia1998-2": Alternative DNA salt correction - "Owczarzy2004": Comprehensive salt correction - "Owczarzy2008": Updated comprehensive salt correction Note: Setting to NA disables salt correction
DMSO	Percent DMSO concentration in the reaction mixture. Default: 0 DMSO can lower the melting temperature of nucleic acid duplexes.
formamide_unit	Formamide concentration as 'list(value, unit)'. Default: list(value = 0, unit = "percent") - value: numeric value of formamide concentration - unit: character string specifying the unit ("percent" or "molar") Default: list(value=0, unit="percent")
dmso_factor	Coefficient of melting temperature (Tm) decrease per percent DMSO. Default: 0.75 (von Ahsen N, 2001, PMID:11673362) Other published values: 0.5, 0.6, 0.675
formamide_factor	Coefficient of melting temperature (Tm) decrease per percent formamide. Default: 0.65 Literature reports values ranging from 0.6 to 0.72

Details

DNA_NN_Breslauer_1986: Breslauer K J (1986) <doi:10.1073/pnas.83.11.3746>

DNA_NN_Sugimoto_1996: Sugimoto N (1996) <doi:10.1093/nar/24.22.4501>

DNA_NN_Allawi_1998: Allawi H (1998) <doi:10.1093/nar/26.11.2694>

DNA_NN_SantaLucia_2004: SantaLucia J (2004) <doi:10.1146/annurev.biophys.32.110601.141800>

RNA_NN_Freier_1986: Freier S (1986) <doi:10.1073/pnas.83.24.9373>

RNA_NN_Xia_1998: Xia T (1998) <doi:10.1021/bi9809425>

RNA_NN_Chen_2012: Chen JL (2012) <doi:10.1021/bi3002709>

RNA_DNA_NN_Sugimoto_1995: Sugimoto N (1995)<doi:10.1016/S0048-9697(98)00088-6>

DNA_TMM_Bommarito_2000: Bommarito S (2000) <doi:10.1093/nar/28.9.1929>

DNA_IMM_Peyret_1999: Peyret N (1999) <doi:10.1021/bi9825091> & Allawi H T (1997) <doi:10.1021/bi962590c> & Santalucia N (2005) <doi:10.1093/nar/gki918>

DNA_DE_Bommarito_2000: Bommarito S (2000) <doi:10.1093/nar/28.9.1929>

RNA_DE_Turner_2010: Turner D H (2010) <doi:10.1093/nar/gkp892>

Author(s)

Junhui Li

References

Breslauer K J , Frank R , Blocker H , et al. Predicting DNA duplex stability from the base sequence.[J]. Proceedings of the National Academy of Sciences, 1986, 83(11):3746-3750.

Sugimoto N , Nakano S , Yoneyama M , et al. Improved Thermodynamic Parameters and Helix Initiation Factor to Predict Stability of DNA Duplexes[J]. Nucleic Acids Research, 1996, 24(22):4501-5.

Allawi, H. Thermodynamics of internal C.T mismatches in DNA[J]. Nucleic Acids Research, 1998, 26(11):2694-2701.

Hicks L D , Santalucia J . The thermodynamics of DNA structural motifs.[J]. Annual Review of Biophysics & Biomolecular Structure, 2004, 33(1):415-440.

Freier S M , Kierzek R , Jaeger J A , et al. Improved free-energy parameters for predictions of RNA duplex stability.[J]. Proceedings of the National Academy of Sciences, 1986, 83(24):9373-9377.

Xia T , Santalucia , J , Burkard M E , et al. Thermodynamic Parameters for an Expanded Nearest-Neighbor Model for Formation of RNA Duplexes with Watson-Crick Base Pairs,[J]. Biochemistry, 1998, 37(42):14719-14735.

Chen J L , Dishler A L , Kennedy S D , et al. Testing the Nearest Neighbor Model for Canonical RNA Base Pairs: Revision of GU Parameters[J]. Biochemistry, 2012, 51(16):3508-3522.

Bommarito S, Peyret N, Jr S L. Thermodynamic parameters for DNA sequences with dangling ends[J]. Nucleic Acids Research, 2000, 28(9):1929-1934.

Turner D H , Mathews D H . NNDB: the nearest neighbor parameter database for predicting stability of nucleic acid secondary structure[J]. Nucleic Acids Research, 2010, 38(Database issue):D280-D282.

Sugimoto N , Nakano S I , Katoh M , et al. Thermodynamic Parameters To Predict Stability of RNA/DNA Hybrid Duplexes[J]. Biochemistry, 1995, 34(35):11211-11216.

Allawi H, SantaLucia J: Thermodynamics and NMR of internal G-T mismatches in DNA. Biochemistry 1997, 36:10581-10594.

Santalucia N E W J . Nearest-neighbor thermodynamics of deoxyinosine pairs in DNA duplexes[J]. Nucleic Acids Research, 2005, 33(19):6258-67.

Peyret N , Seneviratne P A , Allawi H T , et al. Nearest-Neighbor Thermodynamics and NMR of DNA Sequences with Internal A-A, C-C, G-G, and T-T Mismatches, [J]. Biochemistry, 1999, 38(12):3468-3477.

Examples

```
input_seq <- c("AAAATTTTTTTCCCCCCCCCCCCCCGGGGGGGGGGGGTGTGCGCTGC",
"AAAATTTTTTTCCCCCCCCCCCCCCGGGGGGGGGGGGTGTGCGCTGC")
seqs <- to_genomic_ranges(input_seq)
out <- tm_nn(seqs, Na=50)
out
```

`tm_wallace` *Calculate the melting temperature using the 'Wallace rule'*

Description

The Wallace rule is often used as rule of thumb for approximate melting temperature calculations for primers with 14 to 20 nt length.

Usage

```
tm_wallace(gr_seq, ambiguous = FALSE)
```

Arguments

`gr_seq`	Pre-processed sequence(s) in 5' to 3' direction. This should be the output from to_genomic_ranges() function.
`ambiguous`	Ambiguous bases are taken into account to compute the G and C content when ambiguous is TRUE.

Value

Returns a list of sequences with updated Tm attributes

Author(s)

Junhui Li

References

Thein S L , Lynch J R , Weatherall D J , et al. DIRECT DETECTION OF HAEMOGLOBIN E WITH SYNTHETIC OLIGONUCLEOTIDES[J]. The Lancet, 1986, 327(8472):93.

Examples

```
input_seq = c('acgtTGCAATGCCGTAWSDBSY','acgtTGCCCCGGCCGCGCCGTAWSDBSY') #for wallace rule
gr_seq <- to_genomic_ranges(input_seq)
out <- tm_wallace(gr_seq, ambiguous = TRUE)
out
out$Options
```

`to_genomic_ranges_fast` *Convert input sequences to a GRanges object (fast backend)*

Description

Drop-in replacement for `to_genomic_ranges` that uses the fast coordinate backend in `coor_to_genomic_ranges` for genomic coordinate input. Accepts the same `input_seq` and `complement_seq` arguments as `to_genomic_ranges`, plus a list input `list(pkg_name = ..., seq = ...)` for large tiling jobs.

This function processes a vector of sequences string, a FASTA file, or a character vector with genomic coordinates into a GenomicRanges object, optionally including complementary sequences. sequence names are parsed based on their format: - If names have this pattern "chr:start-end:strand:species[:name]" (e.g., "chr1:1-5:+:seq_1"), parse components into seqnames, ranges, strand, and name. - If names have this pattern "chr:start-end:strand" (e.g., "chr1:1-5:+"), parse components into seqnames, ranges, and strand. - If names have this pattern "chr:start-end" (e.g., "chr1:1-5"), parse components into seqnames and ranges. - If no names are provided, use default values: seqnames = "chr1", start = 1, width = sequence length, strand = "*", name = "1", etc. Complementary sequences are either provided or automatically generated.

Usage

```
to_genomic_ranges_fast(
  input_seq,
  complement_seq = NULL,
  method = c("vectorized", "preload_chr")
)

to_genomic_ranges(input_seq, complement_seq = NULL)
```

Arguments

`input_seq`	Input sequence(s) in 5' to 3' direction. Can be provided as either: - A character string (e.g., c("ATGCG", "GCTAG")) - A path to a FASTA file containing the sequence(s) - A character vector where each element is a string in the format "chr:start-end:strand:species" #' (e.g., "chr1:100-200:+:BSgenome.Hsapiens.UCSC.hg38"). Strand is "+" for positive or "-" for negative. - chr: Chromosome ID - start: Start position - end: End position - strand: positive or negative strand - species: Species name for reference genome (e.g., "BSgenome.Hsapiens.UCSC.hg38"), see `BSgenome::available.genomes()` for all available genomes. please make sure the genome package is installed, otherwise the function will stop.
`complement_seq`	Optional complementary sequences. If NULL, complementary sequences will be auto-generated. otherwise, the complementary sequences will be used as metadata. Can be provided as format of input_seq.
`method`	Sequence extraction method passed to `coor_to_genomic_ranges`. One of `"vectorized"` (default) or `"preload_chr"`.

Value

A GRanges object. See coor_to_genomic_ranges for metadata columns when coordinate input is used.

A GenomicRanges object with seqnames, ranges, strand, name, sequence, Complement, and Tm as metadata.

Author(s)

Junhui Li

Examples

```
## Not run:
gr <- to_genomic_ranges_fast(
  list(
    pkg_name = "BSgenome.Hsapiens.UCSC.hg38",
    seq = c("chr1:1000-1199:+:win1", "chr1:1200-1399:+:win2")
  )
)

## End(Not run)

# Using a character vector with auto-generated complementary sequences
seqs <- c("ATGCG", "GCTAG")
names(seqs) <- c("chr1:1-5:+:seq_1", "chr2:1-5:+")
gr <- to_genomic_ranges(seqs)
gr

# Using a character vector with provided complementary sequences
seqs <- c("ATGCG", "GCTAG")
comp_seqs <- c("TACGC", "CGTA")
gr <- to_genomic_ranges(seqs, comp_seqs)
gr

# Using a FASTA file
gr <- to_genomic_ranges(system.file("extdata", "example1.fasta", package = "TmCalculator"))
## Not run:
# Using a character vector with genomic coordinates
seqs <- c(
  "chr1:1898000-1898050:+:BSgenome.Hsapiens.UCSC.hg38",
  "chr2:2563000-2563050:-:BSgenome.Hsapiens.UCSC.hg38"
)
gr <- to_genomic_ranges(seqs)
gr

## End(Not run)
```

vec_to_genomic_ranges *Convert sequence strings to GenomicRanges object*

Description

This function converts sequence strings to a GenomicRanges object, handling both named and unnamed sequences. It can also process complementary sequences if provided. sequence names can be in the format ">chr2:1-10:+:seq2" which will be parsed into chromosome, position, strand, and name components.

Usage

```
vec_to_genomic_ranges(input_seq)
```

Arguments

input_seq	A character vector of sequences. If named with format "chr2:1-10:[+|-]:[seq_name]" the name will be parsed into GRanges components.

Value

A GenomicRanges object containing: - GRanges information (seqnames, ranges, strand) - sequence data - Complementary sequences - Names from input or auto-generated

Examples

```
# Example with named sequences in GRanges format
seqs <- c("ATGCG", "GCTAG")
names(seqs) <- c("chr1:1111-1115:+:seq1", "chr2:1221-1225:+")
gr <- vec_to_genomic_ranges(seqs)

# Example with unnamed sequences
seqs <- c("ATGCG", "GCTAG")
gr <- vec_to_genomic_ranges(seqs)
```

Index