Package 'TmCalculator'

October 21, 2026

Type Package

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

Version 1.1.3

Date 2026-10-20

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 four input sources: sequence strings, a FASTA file, an installed 'BSgenome' package named by string, or a 'GRanges' carrying sequences. A 'regions' argument selects what to cover and 'window' and 'slide' set the resolution at which it is tiled. The implementation provides three Tm calculation methods: the Wallace rule (Thein & Wallace, 1986), empirical GC-content formulas (Marmur, 1962; Schildkraut, 1965; Wetmur, 1991; Untergasser, 2012; von Ahsen, 2001), and nearest-neighbor thermodynamics (Breslauer, 1986; Sugimoto, 1996; Allawi, 1997, 1998; SantaLucia, 2004; Freier, 1986; Xia, 1998; Chen, 2012; Bommarito, 2000; Turner, 2010; Sugimoto, 1995; Peyret, 1999; SantaLucia & Peyret, 2001; Watkins & SantaLucia, 2005; Zuber, 2022; Ghosh, 2020, 2023). Nearest-neighbor parameter sets are provided for DNA, RNA and RNA/DNA hybrid duplexes. These include sets obtained by melting-temperature optimization that are fitted directly at a stated sodium concentration (Weber, 2015; Ferreira, 2019; Basilio Barbosa, 2019; Banerjee, 2020), which replace salt correction rather than being corrected; salt correction is skipped automatically when the requested condition matches the one a set was fitted at. The Zuber (2022) set additionally replaces the single terminal-AU penalty with end terms that depend on the penultimate base pair, applied automatically at both duplex ends. Parameter sets measured under molecular crowding (Ghosh, 2020, 2023) are also provided for DNA and RNA duplexes, so that duplex stability can be evaluated under cell-like rather than dilute-solution conditions. Salt corrections are otherwise applied to the nearest-neighbor model (SantaLucia, 1996, 1998; Owczarzy, 2004, 2008), while each empirical GC-content formula carries the salt term it was published with; corrections for chemical conditions such as dimethyl sulfoxide and formamide apply to both. A compiled C++ core, and task partitioning by region across 'BiocParallel' workers through a 'BPPARAM' argument, profile the human genome in 3 minutes on a six-core laptop. This package returns result as a GRanges object for interoperability with Bioconductor workflows and downstream multi-omics analy-

ses. 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.

URL https://github.com/JunhuiLi1017/TmCalculator, https://doi.org/10.3390/app162010400

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

License MIT + file LICENSE

Depends R (>= 3.5)

VignetteBuilder knitr

Imports BiocGenerics, Biostrings, GenomeInfoDb, GenomicRanges, BiocParallel, IRanges, Rcpp, S4Vectors, graphics, grDevices, methods

LinkingTo Rcpp

Suggests BSgenome, testthat (>= 3.0.0), knitr, rmarkdown, remotes, BiocManager, BSgenomeForge, karyoploteR, ggplot2, ggridges, ggforce, ps

NeedsCompilation yes

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 2026-10-21 17:10:02 UTC

Contents

check_filter_seq 3
chem_correct 4
compare_groups 5
complement_fast 7
coor_to_genomic_ranges 8
ecoli_rep_hotspots 9
fa_to_genomic_ranges 10
gc_content 11

generate_complement 12
integrate_granges 13
make_genomiccoord 18
plot_genome_track 22
plot_tm 27
print.TmCalculator 30
salt_correct 31
thermodynamic_gc_params 33
thermodynamic_nn_params 34
tm_calculate 37
tm_gc 45
tm_nn 48
tm_wallace 56
to_genomic_ranges_fast 58
vec_to_genomic_ranges 60
$.TmCalculator 61

Index **62**

`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 accepted empirical coefficients: 0.5, 0.6, 0.65, 0.675
`formamide_factor`	Coefficient of melting temperature (Tm) decrease per percent formamide. Default: 0.65 (empirical convention); accepted alternatives: 0.6, 0.72. Hutton (1977) reports 0.60, not the default 0.65.
`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, from Blake & Delcourt (1996).

Author(s)

Junhui Li

References

von Ahsen N, Wittwer CT, Schutz E. Oligonucleotide melting temperatures under PCR conditions: nearest-neighbor corrections for Mg2+, deoxynucleotide triphosphate, and dimethyl sulfoxide concentrations with comparison to alternative empirical formulas. Clin Chem 2001, 47:1956-1961.

Blake RD, Delcourt SG (1996). Thermodynamic effects of formamide on DNA stability. NAR 24:2095-2103. <doi:10.1093/nar/24.11.2095>

Hutton JR (1977). Renaturation kinetics and thermal stability of DNA in aqueous solutions of formamide and urea. <doi:10.1093/nar/4.10.3537>

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). Best when windows are scattered across many chromosomes. "preload_chr" Load each chromosome once and extract all of its windows in a single extractAt() call. Recommended for dense tiling of one or a few chromosomes (e.g. genome-wide sliding windows), where it is several times faster than getSeq(); holds one chromosome in memory at a time. A BSgenome caches the sequences it has loaded, so a second call for the same chromosome does not read it again.

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 percentage (0-100) per interval, computed as 100 * (G+C)/(A+C+G+T) so that N is excluded from the denominator.

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_content *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_content(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_content(c("a","t","c","t","g","g","g","c","c","a","g","t","a"))  # 53.85%

# Calculate GC content including ambiguous bases
gc_content("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, controlling which of the two ways of writing the opposite strand is returned. `FALSE` (default) gives the plain complement: base `i` of the result pairs with base `i` of the input, so written underneath the input it runs **3' to 5'**. This is the form `to_genomic_ranges` expects for `complement_seq`, and the form its auto-generated complements take. `TRUE` gives the reverse complement: the same strand written the conventional way round, **5' to 3'**. It is what you would order from a supplier, and it is *not* what `complement_seq` wants – passing it there pairs every position against the wrong base.

```
 input                5'-A T G C G-3'
reverse = FALSE      T A C G C    (3' to 5', pairs position by position)
reverse = TRUE     5'-C G C A T-3'  (the same strand, written 5' to 3')
```

Value

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

Author(s)

Junhui Li

References

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

Examples

```
# Plain complement: pairs position by position, reads 3' to 5'
generate_complement("ATGCG", reverse = FALSE)

# Reverse complement: the same strand written 5' to 3'
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,
  weight = c("none", "overlap"),
```

```
  report_coverage = FALSE,
  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. It is called as agg_fun(values, na.rm = TRUE), so it must accept an na.rm argument; mean, median, sum and max all qualify, whereas function(x) x[1] does not. Ignored for character columns, which are always joined as a comma-separated list of their unique values. Default: mean.
weight	Character. How features are combined within a range. "none" (default) gives every feature that passes the overlap test the same weight, whatever the length of its overlap. "overlap" computes a mean weighted by the number of overlapping base pairs, $\sum_j w_{ij} v_j / \sum_j w_{ij}$ with w_{ij} the width of the intersection of range i and feature j. The distinction matters wherever a signal changes sharply. A 200 bp window whose first 190 bp are covered at depth 5 and whose last 10 bp are covered at depth 200 has a true mean depth of 14.75; unweighted aggregation returns 102.5, because the 10 bp feature counts as much as the 190 bp one. Raising min_overlap does not fix this, it only reverses the sign of the bias by discarding the short feature entirely. "overlap" requires agg_fun = mean: an overlap-weighted maximum has no accepted definition, and quietly ignoring agg_fun would return an unweighted value that looks weighted. It does not apply to strategy = "nearest", which performs no aggregation.

report_coverage
: Logical. Add a covered_frac column giving the fraction of each range covered by at least one feature, computed after reducing the features so that overlapping ones are not counted twice. A weighted mean normalises by the covered bases, so a value derived from a quarter of a window is indistinguishable from one derived from all of it; this column is what makes the difference visible. Produced by the "overlap" and "window" strategies. Default: FALSE.

min_overlap
: Integer. Minimum overlap in base pairs required between a Tm range and a feature range. It is a filter and not a weight: a feature either qualifies or does not, and a qualifying feature counts in full. Applies to the "overlap" strategy only; "window" and "bin" require a single overlapping base pair. 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.

Aggregating continuous signals

When several features map to the same range, numeric columns are summarised by agg_fun and character columns are joined as their unique values. Which features take part is decided by min_overlap, and how much each one counts is decided by weight. The two are easy to confuse, and the distinction is what determines whether a coverage-like signal is summarised correctly at a boundary.

Let R_i be the i-th range of gr_tm, F_j the j-th feature, x_j its value, and w_{ij} the number of base pairs R_i and F_j share. Write m for min_overlap and f for agg_fun. The features entering the summary of R_i are those with $w_{ij} \geq m$, and

$$v_i = f(\{x_j : w_{ij} \geq m\})$$

with weight = "none", or

$$v_i = \frac{\sum_j w_{ij} x_j}{\sum_j w_{ij}}$$

taken over the same features, with weight = "overlap".

The unweighted form gives a feature that overlaps by one base pair the same influence as one that spans the whole range. This is harmless where a signal is flat and wrong where it steps, which is to say at exon boundaries, peak edges and promoters. Raising `min_overlap` does not repair it: the threshold is a filter, so a short feature is either counted in full or discarded in full, and the bias changes sign rather than disappearing. The example below shows both failures against a case with a known answer.

The weighted mean normalises by the covered base pairs, not by the width of the range, so a value derived from a quarter of a window is indistinguishable from one derived from all of it. `report_coverage = TRUE` adds a `covered_frac` column giving the fraction of each range covered by at least one feature, computed after `reduce`-ing the features so that overlapping ones are not double counted. Multiply by it to convert a mean over covered bases into a mean over the range.

Weighting is not the default. Enabling it changes numeric output, and existing analyses should stay reproducible unless their author decides otherwise.

Author(s)

Junhui Li

Examples

```
## Aggregation: a coverage track with a known answer -----------------------
## Two 200 bp windows over a signal that steps sharply inside the first.
##
##   window 1  [  1 .. 200]        window 2  [401 .. 600]
##   depth     [  1 .. 190] = 5
##             [191 .. 400] = 200
##                                          [401 .. 450] = 60
library(GenomicRanges)

win <- GRanges("chr1", IRanges(start = c(1, 401), width = 200),
               Tm = c(70, 72))
cov <- GRanges("chr1", IRanges(start = c(1, 191, 401),
                               end   = c(190, 400, 450)),
               cov = c(5, 200, 60))

## Window 1 truly averages (5 * 190 + 200 * 10) / 200 = 14.75.
integrate_granges(win, cov, strategy = "overlap")$cov
## 102.5  60   the 10 bp feature counts as much as the 190 bp one

integrate_granges(win, cov, strategy = "overlap",
                  weight = "overlap")$cov
## 14.75  60   overlap-weighted mean recovers the true depth

integrate_granges(win, cov, strategy = "overlap", min_overlap = 20L)$cov
## 5      60   the threshold discards the short feature; bias reverses

## Window 2 is only a quarter covered. Weighting cannot show that, because
## it normalises by the covered bases; the coverage column can.
res <- integrate_granges(win, cov, strategy = "overlap",
                         weight = "overlap", report_coverage = TRUE)
res$cov                     # 14.75  60
```

```
res$covered_frac             # 1.00   0.25
res$cov * res$covered_frac   # 14.75  15    mean over the whole window

## 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 in-memory N-end detection scan used by `trim_N = "ends"`. The chromosome range is loaded once and scanned block by block at C speed; values below 1 Mb are raised to 1 Mb. The detected positions are exact regardless of block size. Default: `window`.

`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 `N`s 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,
  plot.params = NULL,
  main = NULL,
  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,
  base.tick.units = TRUE,
  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).
plot.params	Optional named list overriding entries of karyoploteR::getDefaultPlotParams() (linear only). The defaults size the margins for a single track, so several tracks in one panel leave conspicuous white space above and below; e.g. plot.params = list(topmargin = 15, bottommargin = 15,data1outmargin = 8). Unknown names are ignored with a warning.
main	Panel title. NULL (default) uses the genome name, with the interval appended when several zoom regions are drawn. Supply a character string to replace it, or NA / "" to omit it: a manuscript figure normally carries the genome and interval in its caption instead, and the title otherwise takes vertical space from the tracks. A vector of the same length as zoom titles each panel separately.
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).

`base.tick.units`
: Logical (linear only). Append the unit to the base position labels, so that an axis reads `100 kb` rather than `100`, which on its own could be bases, kb or Mb. Default `TRUE`.

`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": Schildkraut & Lifson (1965); historical identifier - "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: Schildkraut & Lifson (1965) monovalent correction; the name is historical - 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: Sodium correction from Owczarzy et al. (2004) (requires sequence information) - Owczarzy2008: Updated comprehensive salt correction (requires sequence information)

For methods other than Owczarzy2008, the sodium-equivalent magnesium adjustment follows von Ahsen et al. (2001); it is separate from the original monovalent formulas.

Owczarzy2008 is piecewise. Which of its three forms applies is decided by the ratio $R = \sqrt{[Mg^{2+}]_{free}}/[Mon]$, where $[Mon]$ is the total monovalent concentration in molar and the free magnesium is what is left after chelation by dNTPs ($K_a = 3 \times 10^4\ M^{-1}$). Below 0.22 the monovalent term alone applies; between 0.22 and 6 a competing form whose coefficients are themselves functions of $[Mon]$; at 6 and above the divalent-dominated form, in which $[Mon]$ drops out of the expression entirely. A solution with magnesium and no monovalent cation at all therefore has a defined correction rather than none.

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.

SantaLucia J Jr, Allawi HT, Seneviratne PA (1996). Improved nearest-neighbor parameters for predicting DNA duplex stability. <doi:10.1021/bi951907q>

Owczarzy R et al. (2004). Effects of sodium ions on DNA duplex oligomers: improved predictions of melting temperatures. <doi:10.1021/bi034621r>

von Ahsen N, Wittwer CT, Schutz E (2001). Oligonucleotide melting temperatures under PCR conditions. <doi:10.1093/clinchem/47.11.1956>

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_1997 DNA/DNA Watson-Crick NN, Allawi & SantaLucia (1997), Table 1; legacy identifier

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 internal mismatches, six sources (1997-2005); see tm_nn

DNA_TMM_SantaLucia_2001 DNA terminal mismatch, SantaLucia & Peyret (2001), WO2001094611A2, Tables 2-3; legacy identifier

DNA_DE_Bommarito_2000 DNA single dangling end, Bommarito (2000)

RNA_DE_Turner_2010 RNA single dangling end, NNDB Turner 2004 compilation; database described in 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. See the installed `extdata/tm_nn_reference_audit.md` for the audit of runtime tables.

References

Breslauer K J (1986) <doi:10.1073/pnas.83.11.3746> Sugimoto N (1996) <doi:10.1093/nar/24.22.4501> Allawi H T & SantaLucia J Jr (1997), Table 1 <doi:10.1021/bi962590c>; also SantaLucia (1998), Table 2 <doi:10.1073/pnas.95.4.1460> 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:10.1021/bi00035a029> 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> Watkins N E Jr & SantaLucia J Jr (2005) <doi:10.1093/nar/gki918> Allawi H T & SantaLucia J Jr (1998, G.A) <doi:10.1021/bi9724873> Allawi H T & SantaLucia J Jr (1998, A.C) <doi:10.1021/bi9803729> Allawi H T & SantaLucia J Jr (1998, C.T) <doi:10.1093/nar/26.11.2694> SantaLucia J Jr & Peyret N (2001), WO2001094611A2, Tables 2-3. https://patents.google.com/patent/WO2001094611A2/en Turner D H & Mathews 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 nucleic acid melting temperature (Tm) by one of three methods, and returns the result as a `GRanges` object so that Tm can be used directly as a quantitative genomic feature alongside other assays:

- **Nearest neighbor** (`tm_nn`) sums the stacking free energies of adjacent base-pair steps using experimentally derived enthalpy and entropy parameters, and so resolves sequences of identical base composition but different order. It supports salt and chemical corrections, mismatches and dangling ends, and is the default.
- **GC content** (`tm_gc`) computes Tm as an empirical function of GC percentage with corrections for length and ionic strength. Cheaper than NN, but blind to sequence order.
- **Wallace rule** (`tm_wallace`) assigns a fixed contribution per base. It is calibrated for short oligonucleotides, typically 14 to 20 bp, and ignores sequence context and reaction conditions, so it is not appropriate for long sequences or for genome-wide windows.

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_Ghosh_2020_PEG200",
    "DNA_NN_Breslauer_1986", "DNA_NN_Sugimoto_1996", "DNA_NN_Allawi_1997",
   "RNA_NN_Freier_1986", "RNA_NN_Xia_1998", "RNA_NN_Chen_2012", "RNA_NN_Zuber_2022",
    "RNA_NN_Ghosh_2023_PEG200", "RNA_DNA_NN_Sugimoto_1995", "DNA_NN_Weber_2015",
    "DNA_NN_Weber_OW04_69", "DNA_NN_Weber_OW04_119", "DNA_NN_Weber_OW04_220",
    "DNA_NN_Weber_OW04_621", "DNA_NN_Weber_OW04_1020", "RNA_NN_Weber_VIF_71",
    "RNA_NN_Weber_VIF_121", "RNA_NN_Weber_VIF_221", "RNA_NN_Weber_VIF_621",

    "RNA_NN_Weber_VIF_1021", "RNA_NN_Weber_FIF_71", "RNA_NN_Weber_FIF_121",
    "RNA_NN_Weber_FIF_221", "RNA_NN_Weber_FIF_621", "RNA_NN_Weber_FIF_1021",
   "RNA_DNA_NN_Weber_2019_FT", "RNA_DNA_NN_Weber_2019_VH", "RNA_DNA_NN_Weber_2019_LS",
    "RNA_DNA_NN_Banerjee_2020"),
  tmm_table = "DNA_TMM_SantaLucia_2001",
  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",
   "SantaLucia1998-2", "Owczarzy2004", "Owczarzy2008", "none"),
 DMSO = 0,
 formamide_unit = list(value = 0, unit = "percent"),
 dmso_factor = 0.75,
 formamide_factor = 0.65,
 mismatch = TRUE,
 regions = NULL,
 window = NULL,
 slide = window,
 unit = c("segment", "region"),
 segment_size = 5e+07,
 BPPARAM = NULL,
 keep_sequence = NULL,
 tmpdir = tempdir(),
 verbose = FALSE
)
```

Arguments

`input_seq` Where the sequence comes from. One of:

- **Sequences**, as a character vector in 5' to 3' direction, e.g. `c("ATGCG", "GGCCA")`. Names, when present, become the `seqnames` of the result; unnamed sequences are keyed by their position in the vector.
- **An installed BSgenome package**, by name, e.g. `"BSgenome.Hsapiens.UCSC.hg38"`. It is named rather than passed as a loaded object because each worker opens the genome for itself, so no sequence crosses between processes. See `BSgenome::available.genomes()` for what exists, and install the package before calling.
- **A FASTA file**, by path; gzipped files are read directly.
- **A** `GRanges` carrying a `sequence` metadata column. The complement is derived from it when absent.

`regions` selects from any of the four, and means the same thing in each: the identifier before the colon is resolved against whatever names the source itself offers, and falls back to position. A BSgenome or a FASTA file with no `regions` is taken whole, which for a genome means its standard chromosomes.

Also accepted, and unchanged from earlier versions: a character vector of coordinate strings `"chr:start-end:strand:species"`, for example `"chr1:100-200:+:BSgenome.Hsapien` where strand defaults to `"+"`. This form carries its own genome in every element, so it is read directly and ignores `regions`; to profile coordinates against a genome, prefer passing the genome here and the coordinates as `regions`.

`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. Sets whose name encodes a sodium concentration were fitted at that condition and are not salt-corrected again. See `tm_nn` for the full list and guidance on choosing between them. Default: "DNA_NN_SantaLucia_2004"

Alternatively, supply a matrix or data.frame of parameters directly. This is the route for parameter sets the package does not ship, in particular sets covering modified bases such as 5-methylcytosine. Requirements:

- numeric, with columns 1 and 2 read as delta H (kcal/mol) and delta S (cal/mol/K); further columns are ignored;
- row names giving the parameter keys, e.g. `"AA/TT"`, `"init"`, `"init_A/T"`, `"sym"`;
- every key of a built-in reference set must be present. The reference is named by `attr(x, "reference")`, or defaults to the first built-in listed for the argument, which is a DNA/DNA set; RNA and hybrid tables should therefore set the attribute. Extra keys beyond the reference are kept, which is how a modified-base set adds stacks rather than replacing them.

The supplied table is reordered to the reference key order before use, so that two tables differing only in row order give identical results. A missing key would otherwise contribute zero to the calculation instead of raising an error, which is why the full key set is required. Keys that disagree with their reverse complement produce a warning: expected for modified bases, a transposition error otherwise.

Two optional attributes are honoured. `attr(x, "salt_mM")` marks a set as fitted at a stated sodium concentration, which suppresses the salt correction at that concentration exactly as for the built-in sets fitted this way; without it the table is treated as a reference-condition set and `salt_method` is applied. `attr(x, "end_table")` supplies a companion penultimate-pair end-effect table.

`tmm_table` Thermodynamic parameters for terminal mismatches. Only applicable for the NN method. Default: "DNA_TMM_SantaLucia_2001"

`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: - "none": Disables salt correction. Also selected automatically when the chosen `nn_table` was fitted at the requested `Na`. - "Schildkraut2010": Schildkraut & Lifson (1965); historical identifier - "Wetmur1991": Classic salt correction method - "SantaLucia1996": DNA-specific salt correction - "SantaLucia1998-1": Improved DNA salt correction, applied to Tm - "SantaLucia1998-2": the same correction applied to the entropy of the nearest-neighbor model rather than to Tm - "Owczarzy2004": Comprehensive salt correction - "Owczarzy2008": Updated comprehensive salt correction Default: "Schildkraut2010"

With `method = "tm_gc"` the salt term is part of the published formula selected by `variant`, so naming a *different* one is ignored there, with a warning, unless `userset` is supplied. `"none"` (or `NA`) is not a substitution but a request to drop the correction, and is honoured on either path without a warning. The two Owczarzy corrections are not available to `"tm_gc"` at all: they apply to the reciprocal of the melting temperature in kelvin, referenced to the same duplex in 1 M Na+, and carry a duplex-length term the GC-content formulas already have.

`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

regions — What to take from input_seq. NULL, the default, means all of it, except for a **BSgenome**, where it means GenomeInfoDb::standardChromosomes() of that genome, which for GRCh38 includes chrM. Otherwise: names or numbers (c("chr1", "chr2"), 1:2), coordinate strings "name:start-end" with commas and scientific notation accepted ("chr1:5,000,000-6e6"), a mixture of the two, or a GRanges.

The identifier before the colon is resolved against whatever names the source itself offers, and falls back to position. So "chr1" is a chromosome in a **BSgenome**, a record in a FASTA file and a seqname in a GRanges, and "1:1-200" is the first 200 bases of the first sequence in an unnamed character vector. On a **BSgenome** the chr prefix is added or removed as the genome requires, since that is a convention rather than information; elsewhere names are matched exactly.

window — Window width in base pairs. NULL, the default, gives one melting temperature per region, which is what short records such as probes, primers and oligonucleotides call for; a region longer than 1 Mb with window = NULL is an error rather than one meaningless Tm. Regions shorter than window are returned whole.

slide — Step between window starts, defaulting to window for a non-overlapping tiling. Ignored when window is NULL.

unit — How regions become tasks: "segment" cuts them into pieces of about segment_size bp, "region" makes one task per region. Segments are faster and need less memory per worker, because no worker then holds a whole large chromosome.

segment_size — Task size in base pairs when unit = "segment", rounded down to a multiple of slide so that the window grid is the one an unsegmented run would produce. Default 50 Mb.

BPPARAM — A BiocParallelParam from **BiocParallel**, e.g. SnowParam(workers = 5), to spread the tasks over processes. NULL, the default, runs them here, with no dependency on **BiocParallel**. Parallelism divides the work by region, never the sequences of one region: each task opens the source itself, so only a name and a coordinate pair cross between processes.

keep_sequence — Keep the sequence and complement columns. The default keeps them for sequences the caller supplied and drops them for a genome or a file, where they run to roughly 500 MB per large chromosome.

tmpdir — Directory for the temporary FASTA file written when sequences are supplied directly and there is tiling or parallelism to do. Worth setting on a cluster, where tempdir() is often a small partition.

verbose — Report the task and window counts.

Details

The input sequence is processed once and passed to the selected method, which is faster than calling the individual functions separately.

The three methods differ in resolution and in the range of sequence lengths over which they are calibrated, so they are not interchangeable.

tm_nn is the appropriate default. Because it sums sequence-dependent stacking terms, it distinguishes sequences of identical GC content but different base order, which the other two cannot. Its

parameters were derived from short duplexes under a two-state assumption; when applied to long sequences or to fixed-width genomic windows the resulting value is best read as a relative measure of local thermodynamic stability for comparison across windows, rather than as an absolute experimental melting temperature.

`tm_gc` computes Tm from GC percentage using one of several published empirical formulas selected by `variant`, with corrections for length and ionic strength. It extends to longer sequences at low computational cost but cannot resolve base order.

`tm_wallace` applies the 2 + 4 rule, adding 2 °C per A or T and 4 °C per G or C. It is calibrated for short oligonucleotides, typically 14 to 20 bp, and takes no account of sequence context, salt or chemical additives. Accuracy degrades quickly with length, so it is retained for compatibility rather than recommended for genome-scale work.

Salt and chemical corrections apply to `tm_nn` and `tm_gc` only. The input sequence is parsed and validated once and reused by the selected method, which is faster than calling the individual functions directly.

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. For the nearest-neighbor method this includes `Salt correction applied` (logical) and `Parameter set fitted at [Na+] (mM)`, which record whether a salt correction was actually performed.

Salt handling

Most nearest-neighbor parameter sets were fitted at a single reference sodium concentration, and other conditions are reached through the `salt_method` correction formulas. The Weber/VarGibbs sets were instead fitted directly at a stated sodium concentration and are meant to replace salt correction. When such a set is selected and `Na` matches the concentration it was fitted at, correction is skipped automatically; when it does not, correction is applied with a warning. See `tm_nn` for details.

Available Options

Method Selection:

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

Nearest Neighbor (NN) Method Options:

- `nn_table`:
  - DNA/DNA: "DNA_NN_Breslauer_1986", "DNA_NN_Sugimoto_1996", "DNA_NN_Allawi_1997", "DNA_NN_SantaLucia_2004" (default)
  - DNA/DNA, molecular crowding: "DNA_NN_Ghosh_2020_PEG200" (40 wt
  - DNA/DNA, salt-optimized: "DNA_NN_Weber_2015" (1020 mM), "DNA_NN_Weber_OW04_69", "..._119", "..._220", "..._621", "..._1020" (fitted at 69 to 1020 mM sodium)

  - RNA/RNA: "RNA_NN_Freier_1986", "RNA_NN_Xia_1998", "RNA_NN_Chen_2012", "RNA_NN_Zuber_2022" (improved end effects), "RNA_NN_Ghosh_2023_PEG200" (molecular crowding, cell-like)
  - RNA/RNA, salt-optimized: "RNA_NN_Weber_VIF_71", "..._121", "..._221", "..._621", "..._1021" and the corresponding "RNA_NN_Weber_FIF_*" sets
  - RNA/DNA: "RNA_DNA_NN_Sugimoto_1995", "RNA_DNA_NN_Weber_2019_FT", "RNA_DNA_NN_Weber_ (1000 mM), "RNA_DNA_NN_Weber_2019_LS" (100 mM), "RNA_DNA_NN_Banerjee_2020" (100 mM)

- `tmm_table` (Terminal Mismatches):
  - "DNA_TMM_SantaLucia_2001" (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" (`method = "tm_nn"` only)
  - "Owczarzy2004" (`method = "tm_nn"` only)
  - "Owczarzy2008" (`method = "tm_nn"` only)
  - "none" (also selected automatically when `nn_table` was fitted at the requested `Na`)

With `method = "tm_gc"` the salt term belongs to the published formula selected by `variant`, so naming a different one is ignored there, with a warning, unless `userset` is supplied. `"none"` and `NA` drop the correction and are honoured on either path.

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

See Also

`tm_nn` for the nearest-neighbor method and the full list of thermodynamic parameter sets.

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"
)

# A hybrid parameter set fitted at 100 mM sodium. Salt correction is
# skipped automatically because Na matches the fitted condition.
result_ls <- tm_calculate(
  input_seq,
  method = "tm_nn",
  nn_table = "RNA_DNA_NN_Weber_2019_LS",
  Na = 100
```

```
)

# Genome scale. The source is named rather than loaded, because each
# worker opens it for itself; `regions` says what to cover and `window`
# says at what resolution.
hg38 <- "BSgenome.Hsapiens.UCSC.hg38"
tm_calculate(hg38, regions = "chr21:10e6-20e6", window = 200, slide = 200)

# Five processes. Tasks are divided by region, never by splitting the
# sequences of one region, so only coordinates cross between them.
library(BiocParallel)
whole <- tm_calculate(hg38, window = 200, slide = 200,
                      BPPARAM = SnowParam(workers = 5))
whole$gr

# The same, for a FASTA file and for sequences already in R. With
# window = NULL, the default, each record returns a single Tm, which is
# what a file of probes or primers calls for.
tm_calculate("probes.fa", BPPARAM = SnowParam(workers = 4))
tm_calculate(c("ACGTGCTAGCTAGCTAGC", "GGCCATATATGCGC"))

## 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 = NULL,
  mismatch = TRUE,
  DMSO = 0,
```

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

Arguments

gr_seq	Sequence(s) in 5' to 3' direction, as the GRanges that to_genomic_ranges returns. A character vector of sequences, a path to a FASTA file, or genomic coordinate strings are also accepted and converted for you. To pair a sequence with a complement of your own, build the object explicitly: to_genomic_ranges(seq, complement_seq = cmp).
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): the correction that belongs to the formula, i.e. the one published with variant, or "Schildkraut2010" when userset is supplied. - NA or "none": no salt correction at all. - "Schildkraut2010": Schildkraut & Lifson 1965 - "Wetmur1991": Wetmur 1991 - "SantaLucia1996": SantaLucia 1996 - "SantaLucia1998-1": SantaLucia 1998 (Method 1) With a built-in variant the salt term is part of the published formula rather than a free choice, so naming a *different* one is ignored with a warning: the result would otherwise be labelled with one method and computed with another. Supply userset to choose the correction yourself. Dropping it with NA is not a substitution and is honoured on either path. "SantaLucia1998-2", "Owczarzy2004" and "Owczarzy2008" are not available for this function. The first corrects the entropy of a nearest-neighbor model, which a GC-content formula does not have. The other two correct the reciprocal

of the melting temperature in kelvin, referenced to the same duplex in 1 M Na+, and carry a duplex-length term of their own, which these formulas already have. All three are available in tm_nn.

`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, 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. 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

Calculates melting temperature (Tm) from nearest-neighbor (NN) thermodynamics, summing the stacking enthalpies and entropies of adjacent base-pair steps and applying initiation, symmetry, salt and chemical corrections. Terminal mismatches, internal mismatches and dangling ends are supported through dedicated parameter tables. The function verifies that every dinucleotide step in the input sequence is present in the selected tables before calculating.

Usage

```
tm_nn(
  gr_seq,
  ambiguous = FALSE,
  shift = 0,
  nn_table = c("DNA_NN_SantaLucia_2004", "DNA_NN_Ghosh_2020_PEG200",
    "DNA_NN_Breslauer_1986", "DNA_NN_Sugimoto_1996", "DNA_NN_Allawi_1997",
   "RNA_NN_Freier_1986", "RNA_NN_Xia_1998", "RNA_NN_Chen_2012", "RNA_NN_Zuber_2022",
    "RNA_NN_Ghosh_2023_PEG200", "RNA_DNA_NN_Sugimoto_1995", "DNA_NN_Weber_2015",
    "DNA_NN_Weber_OW04_69", "DNA_NN_Weber_OW04_119", "DNA_NN_Weber_OW04_220",
    "DNA_NN_Weber_OW04_621", "DNA_NN_Weber_OW04_1020", "RNA_NN_Weber_VIF_71",
    "RNA_NN_Weber_VIF_121", "RNA_NN_Weber_VIF_221", "RNA_NN_Weber_VIF_621",

    "RNA_NN_Weber_VIF_1021", "RNA_NN_Weber_FIF_71", "RNA_NN_Weber_FIF_121",
    "RNA_NN_Weber_FIF_221", "RNA_NN_Weber_FIF_621", "RNA_NN_Weber_FIF_1021",
   "RNA_DNA_NN_Weber_2019_FT", "RNA_DNA_NN_Weber_2019_VH", "RNA_DNA_NN_Weber_2019_LS",
    "RNA_DNA_NN_Banerjee_2020"),
  tmm_table = "DNA_TMM_SantaLucia_2001",
  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",
    "SantaLucia1998-2", "Owczarzy2004", "Owczarzy2008", "none"),
  DMSO = 0,
  formamide_unit = list(value = 0, unit = "percent"),
  dmso_factor = 0.75,
  formamide_factor = 0.65
```

)

Arguments

gr_seq	Sequence(s) in 5' to 3' direction, as the GRanges that to_genomic_ranges returns. A character vector of sequences, a path to a FASTA file, or genomic coordinate strings are also accepted and converted for you. To pair a sequence with a complement of your own, build the object explicitly: to_genomic_ranges(seq, complement_seq = cmp).
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. Parameter sets are listed below by hybridization type. Sets marked with a sodium concentration were fitted at that condition and are not salt-corrected further (see the "Choosing a parameter set" section). DNA/DNA hybridizations, reference salt: - "DNA_NN_Breslauer_1986": Original DNA/DNA parameters - "DNA_NN_Sugimoto_1996": Improved DNA/DNA parameters - "DNA_NN_Allawi_1997": Watson-Crick parameters from Allawi & SantaLucia (1997), Table 1; the historical identifier is retained for compatibility - "DNA_NN_SantaLucia_2004": Unified DNA/DNA parameters (default) DNA/DNA hybridizations, melting-temperature optimized (Weber 2015): - "DNA_NN_Weber_2015": Combined dataset, 1020 mM. Recommended when a salt-optimized DNA set is wanted at high salt - "DNA_NN_Weber_OW04_69", "...119", "...220", "...621", "...1020": fitted independently at 69, 119, 220, 621 and 1020 mM sodium DNA/DNA under molecular crowding (cell-like rather than dilute solution): - "DNA_NN_Ghosh_2020_PEG200": fitted in 40 wt RNA/RNA hybridizations, reference salt: - "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_NN_Zuber_2022": Successor to Xia 1998 with improved end effects. The terminal-AU penalty is replaced by end terms that depend on the penultimate base pair, applied automatically from a companion table RNA/RNA under molecular crowding (cell-like rather than dilute solution): - "RNA_NN_Ghosh_2023_PEG200": fitted in 40 wt and shown to describe duplexes in an intracellular cation composition RNA/RNA hybridizations, salt-optimized (Ferreira 2019). VIF (variable initiation factors) gave better cross-validation than FIF (fixed): - "RNA_NN_Weber_VIF_71", "...121", "...221", "...621", "...1021" - "RNA_NN_Weber_FIF_71", "...121", "...221", "...621", "...1021"

RNA/DNA hybridizations: - "RNA_DNA_NN_Sugimoto_1995": RNA/DNA hybridization parameters - "RNA_DNA_NN_Weber_2019_FT": curve-fitting derived, 1000 mM. Best performing high-salt hybrid set in Basilio Barbosa (2019) - "RNA_DNA_NN_Weber_2019_VH": van't Hoff derived, 1000 mM - "RNA_DNA_NN_Weber_2019_LS": low salt, 100 mM - "RNA_DNA_NN_Banerjee_2020": improved hybrid parameters fitted at a physiological condition (100 mM NaCl), Banerjee et al. (2020)

For every hybrid set the sequence you supply must be the **RNA** strand, written 5' to 3' and spelled with T in place of U; its complement is then the DNA strand, read 3' to 5'. The published keys are indexed the same way, RNA on top: `"AG/TC"` means 5'-rAG-3' paired with 3'-dTC-5'. Unlike DNA/DNA and RNA/RNA parameters, hybrid parameters are *not* invariant under reversing a key, because reversing it swaps which strand carries the ribose. Supplying the DNA strand therefore does not give the same answer: it silently reports the melting temperature of the opposite hybrid. The asymmetry is large, since a purine-rich RNA strand binds DNA more tightly than a pyrimidine-rich one (for example 5'-rAA-3'/3'-dTT-5' and 5'-rUU-3'/3'-dAA-5' differ by about 0.8 kcal/mol in delta G at 37 degrees C).

Alternatively, supply a matrix or data.frame of parameters directly. This is the route for parameter sets the package does not ship, in particular sets covering modified bases such as 5-methylcytosine. Requirements:

- numeric, with columns 1 and 2 read as delta H (kcal/mol) and delta S (cal/mol/K); further columns are ignored;
- row names giving the parameter keys, e.g. `"AA/TT"`, `"init"`, `"init_A/T"`, `"sym"`;
- every key of a built-in reference set must be present. The reference is named by `attr(x, "reference")`, or defaults to the first built-in listed for the argument, which is a DNA/DNA set; RNA and hybrid tables should therefore set the attribute. Extra keys beyond the reference are kept, which is how a modified-base set adds stacks rather than replacing them.

The supplied table is reordered to the reference key order before use, so that two tables differing only in row order give identical results. A missing key would otherwise contribute zero to the calculation instead of raising an error, which is why the full key set is required. Keys that disagree with their reverse complement produce a warning: expected for modified bases, a transposition error otherwise.

Two optional attributes are honoured. `attr(x, "salt_mM")` marks a set as fitted at a stated sodium concentration, which suppresses the salt correction at that concentration exactly as for the built-in sets fitted this way; without it the table is treated as a reference-condition set and `salt_method` is applied. `attr(x, "end_table")` supplies a companion penultimate-pair end-effect table.

`tmm_table` Thermodynamic parameters for terminal mismatches. Default: "DNA_TMM_SantaLucia_2001" These 48 parameters come from SantaLucia & Peyret (2001), patent WO2001094611A2, Tables 2-3. The historical identifier is retained; Bommarito (2000) is the source of DNA dangling ends, not this table.

`imm_table` Thermodynamic parameters for internal mismatches. Default: "DNA_IMM_Peyret_1999" This is a composite of six publications (1997-2005), not a single Peyret (1999)

table: 11 G.T, 8 G.A, 8 A.C, 8 C.T, 16 like-with-like mismatches and 36 inosine entries. The A.C parameters are for pH 7.

`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": NNDB Turner 2004 RNA dangling-end compilation; 2010 is the database publication year. Entropies are derived from tabulated enthalpies and free energies at 37 degrees C

`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": Schildkraut & Lifson (1965); historical identifier - "Wetmur1991": Classic salt correction method - "SantaLucia1996": DNA-specific salt correction - "SantaLucia1998-1": Improved DNA salt correction, applied to Tm - "SantaLucia1998-2": the same correction applied to the entropy of the nearest-neighbor model rather than to Tm, which is why it is available here and not in `tm_gc` - "Owczarzy2004": Comprehensive salt correction - "Owczarzy2008": Updated comprehensive salt correction - "none": Disables salt correction entirely Note: Parameter sets fitted at a specific sodium concentration (those carrying a "salt_mM" attribute, e.g. the Weber/VarGibbs series and Banerjee 2020) already account for salt. When the requested `Na` matches the concentration such a set was fitted at, salt correction is skipped automatically; when it does not, a warning is issued.

`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 accepted empirical coefficients: 0.5, 0.6, 0.65, 0.675

`formamide_factor`

Coefficient of melting temperature (Tm) decrease per percent formamide. Default: 0.65, an empirical convention. Accepted alternatives are 0.6 and 0.72; these are not all estimates from the same publication. The molar formula instead follows Blake & Delcourt (1996).

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_1997: Allawi H T & SantaLucia J Jr (1997), Table 1 <doi:10.1021/bi962590c>; also SantaLucia (1998), Table 2 <doi:10.1073/pnas.95.4.1460>

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:10.1021/bi00035a029>

The following sets were derived by melting-temperature optimization and are fitted at the sodium concentration given in parentheses. They are not salt-corrected further; see the "Choosing a parameter set" section.

DNA_NN_Weber_2015 (1020 mM): Weber G (2015) <doi:10.1093/bioinformatics/btu751>

DNA_NN_Weber_OW04_69 (69 mM), DNA_NN_Weber_OW04_119 (119 mM), DNA_NN_Weber_OW04_220 (220 mM), DNA_NN_Weber_OW04_621 (621 mM), DNA_NN_Weber_OW04_1020 (1020 mM): Weber G (2015) <doi:10.1093/bioinformatics/btu751>

RNA_NN_Weber_VIF_71 (71 mM), RNA_NN_Weber_VIF_121 (121 mM), RNA_NN_Weber_VIF_221 (221 mM), RNA_NN_Weber_VIF_621 (621 mM), RNA_NN_Weber_VIF_1021 (1021 mM): Ferreira I (2019) <doi:10.1016/j.chemphys.2019.01.016>, variable initiation factors

RNA_NN_Weber_FIF_71 (71 mM), RNA_NN_Weber_FIF_121 (121 mM), RNA_NN_Weber_FIF_221 (221 mM), RNA_NN_Weber_FIF_621 (621 mM), RNA_NN_Weber_FIF_1021 (1021 mM): Ferreira I (2019) <doi:10.1016/j.chemphys.2019.01.016>, fixed initiation factors

RNA_DNA_NN_Weber_2019_FT (1000 mM), RNA_DNA_NN_Weber_2019_VH (1000 mM), RNA_DNA_NN_Weber_20
(100 mM): Basilio Barbosa V (2019) <doi:10.1016/j.bpc.2019.106189>

RNA_DNA_NN_Banerjee_2020 (100 mM): Banerjee D (2020) <doi:10.1093/nar/gkaa572>

RNA_NN_Zuber_2022: Zuber J (2022) <doi:10.1093/nar/gkac261>

RNA_NN_Ghosh_2023_PEG200 (100 mM, 40 wt

DNA_NN_Ghosh_2020_PEG200 (100 mM, 40 wt

DNA_TMM_SantaLucia_2001: SantaLucia J Jr & Peyret N (2001), WO2001094611A2, Tables 2-3 (https://patents.google.com/patent/WO2001094611A2/en)

DNA_IMM_Peyret_1999: Allawi H T & SantaLucia J Jr (1997, G.T) <doi:10.1021/bi962590c>; (1998, G.A) <doi:10.1021/bi9724873>; (1998, A.C) <doi:10.1021/bi9803729>; (1998, C.T) <doi:10.1093/nar/26.11.2694>; Peyret N et al. (1999, A.A/C.C/G.G/T.T) <doi:10.1021/bi9825091>; Watkins N E Jr & SantaLucia J Jr (2005, inosine) <doi:10.1093/nar/gki918>

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

RNA_DE_Turner_2010: NNDB Turner 2004 dangling-end compilation; Turner D H & Mathews D H (2010, database description) <doi:10.1093/nar/gkp892>

Value

A TmCalculator list with:

gr	The input GRanges with metadata columns Tm and GC (melting temperature in °C and GC percent). Sequences that could not be evaluated receive NA.
options	The calculation parameters actually used, including the parameter tables and their citations, ion and additive concentrations, Salt correction method, and two fields recording how salt was handled: Salt correction applied (logical) and Parameter set fitted at [Na+] (mM), which is NA for reference-salt sets. Regions skipped for containing N are listed in Skipped regions containing N.

Choosing a parameter set

Parameter sets fall into two families that are handled differently.

The two families are distinguished by one mechanical criterion: a set is condition-specific if and only if it carries a salt_mM attribute.

Reference-salt sets (no salt_mM; Breslauer 1986, Sugimoto 1996, Allawi 1997 (legacy name: Allawi_1998), SantaLucia 2004, Freier 1986, Xia 1998, Chen 2012, Zuber 2022, Sugimoto 1995) were fitted at a single reference sodium concentration, and other conditions are reached by applying one of the salt_method correction formulas.

Condition-specific sets (the Weber/VarGibbs series, Banerjee 2020, and the molecular-crowding sets of Ghosh 2020 and Ghosh 2023) were instead fitted directly at a stated sodium concentration and are intended to *replace* salt correction rather than be corrected. When the requested Na matches the set's salt_mM value, salt correction is skipped automatically; when it does not, the correction is still applied but a warning is issued, since correcting an already condition-specific set double-counts the ionic effect. Whether a correction was applied is recorded in the returned options.

As a rule of thumb, pick the set whose fitted salt is closest to your experimental condition rather than correcting a distant one.

What the initiation terms are charged on

A dangling end or a terminal mismatch is accounted for by its own parameter, from de_table or tmm_table, and the position it covers is then consumed. The initiation terms – init, init_A/T, init_G/C, init_5T/A and the init_allA/T / init_oneG/C pair – are charged on what remains, that is, on the duplex that actually closes.

This matters because the terminal AT penalty is a property of the closing base pair. SantaLucia and Hicks (2004) define it as applied for each end of a duplex that has a terminal AT, so the question it asks is whether the pair holding that end shut is AT or GC. A mismatched terminus is neither an AT nor a GC pair, and a dangling residue is not in a base pair at all, so neither one can carry the penalty. Charging it there would also double-count the end, since the terminal-mismatch and dangling-end parameters were measured as terminal and already carry that end's contribution.

Some implementations index these terms on the sequence as supplied rather than on the trimmed duplex. That makes the melting temperature depend on which of the two strands is handed over as gr_seq: the same duplex, written from the other strand, can then come back with a different answer. The convention here is invariant under that swap by construction. See the 1.1.2 entry in NEWS for a worked example.

Parameter provenance and limitations

Table identifiers are retained for compatibility and are not always literal author-year citations. The installed files `extdata/tm_nn_reference_audit.md` and `extdata/tm_nn_parameter_sources.tsv` record the source and verification status of every built-in parameter row.

The audit identified unresolved numerical/model issues: the patent's `GG/CG` terminal-mismatch enthalpy is inconsistent with its printed free energy; `RNA_NN_Chen_2012` combines refitted Watson-Crick terms with GU terms fitted against Xia (1998); and Banerjee (2020) initiation conditions in Table 2 footnotes are not reproduced by the current per-end application. The audit corrects citations without changing these numerical results. Consult the audit before using these affected models.

Author(s)

Junhui Li

References

Breslauer KJ, Frank R, Blocker H, Marky LA (1986). Predicting DNA duplex stability from the base sequence. PNAS 83:3746-3750. <doi:10.1073/pnas.83.11.3746>

Sugimoto N, Nakano S, Yoneyama M, Honda K (1996). Improved thermodynamic parameters and helix initiation factor to predict stability of DNA duplexes. NAR 24:4501-4505. <doi:10.1093/nar/24.22.4501>

Allawi HT, SantaLucia J Jr (1997). Thermodynamics and NMR of internal G.T mismatches in DNA. Biochemistry 36:10581-10594. Table 1: Watson-Crick parameters; Table 5: G.T mismatches. <doi:10.1021/bi962590c>

SantaLucia J Jr (1998). A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics. PNAS 95:1460-1465. <doi:10.1073/pnas.95.4.1460>

SantaLucia J Jr, Hicks D (2004). The thermodynamics of DNA structural motifs. Annual Review of Biophysics and Biomolecular Structure 33:415-440. <doi:10.1146/annurev.biophys.32.110601.141800>

Freier SM et al. (1986). Improved free-energy parameters for predictions of RNA duplex stability. PNAS 83:9373-9377. <doi:10.1073/pnas.83.24.9373>

Xia T et al. (1998). Thermodynamic parameters for an expanded nearest-neighbor model for formation of RNA duplexes with Watson-Crick base pairs. Biochemistry 37:14719-14735. <doi:10.1021/bi9809425>

Chen JL et al. (2012). Testing the nearest neighbor model for canonical RNA base pairs: revision of GU parameters. Biochemistry 51:3508-3522. <doi:10.1021/bi3002709>

Sugimoto N et al. (1995). Thermodynamic parameters to predict stability of RNA/DNA hybrid duplexes. Biochemistry 34:11211-11216. <doi:10.1021/bi00035a029>

Allawi HT, SantaLucia J Jr (1998). Nearest-neighbor thermodynamics of internal A.C mismatches in DNA: sequence dependence and pH effects. Biochemistry 37:9435-9444. <doi:10.1021/bi9803729>

Allawi HT, SantaLucia J Jr (1998). Nearest neighbor thermodynamic parameters for internal G.A mismatches in DNA. Biochemistry 37:2170-2179. <doi:10.1021/bi9724873>

Allawi HT, SantaLucia J Jr (1998). Thermodynamics of internal C.T mismatches in DNA. NAR 26:2694-2701. <doi:10.1093/nar/26.11.2694>

Peyret N, Seneviratne PA, Allawi HT, SantaLucia J Jr (1999). Nearest-neighbor thermodynamics and NMR of DNA sequences with internal A.A, C.C, G.G, and T.T mismatches. Biochemistry 38:3468-3477. <doi:10.1021/bi9825091>

Watkins NE Jr, SantaLucia J Jr (2005). Nearest-neighbor thermodynamics of deoxyinosine pairs in DNA duplexes. NAR 33:6258-6267. <doi:10.1093/nar/gki918>

SantaLucia J Jr, Peyret N (2001). Method and system for predicting nucleic acid hybridization thermodynamics and computer-readable storage medium for use therein. Patent WO2001094611A2, published December 13, 2001, Tables 2-3. https://patents.google.com/patent/WO2001094611A2/en

Bommarito S, Peyret N, SantaLucia J Jr (2000). Thermodynamic parameters for DNA sequences with dangling ends. NAR 28:1929-1934. <doi:10.1093/nar/28.9.1929>

Turner DH, Mathews DH (2010). NNDB: the nearest neighbor parameter database for predicting stability of nucleic acid secondary structure. NAR 38:D280-D282. <doi:10.1093/nar/gkp892> The RNA dangling-end values are the Turner 2004 compilation: https://rna.urmc.rochester.edu/NNDB/rna_2004/rna_2004_dangling_ends.html

Weber G (2015). Optimization method for obtaining nearest-neighbour DNA entropies and enthalpies directly from melting temperatures. Bioinformatics 31:871-877. <doi:10.1093/bioinformatics/btu751>

Ferreira I, Jolley EA, Znosko BM, Weber G (2019). Replacing salt correction factors with optimized RNA nearest-neighbour enthalpy and entropy parameters. Chemical Physics 521:69-76. <doi:10.1016/j.chemphys.2019.01.016>

Basilio Barbosa V, de Oliveira Martins E, Weber G (2019). Nearest-neighbour parameters optimized for melting temperature prediction of DNA/RNA hybrids at high and low salt concentrations. Biophysical Chemistry 251:106189. <doi:10.1016/j.bpc.2019.106189>

Banerjee D et al. (2020). Improved nearest-neighbor parameters for the stability of RNA/DNA hybrids under a physiological condition. NAR 48:12042-12054. <doi:10.1093/nar/gkaa572>

Zuber J, Schroeder SJ, Sun H, Turner DH, Mathews DH (2022). Nearest neighbor rules for RNA helix folding thermodynamics: improved end effects. NAR 50:5251-5262. <doi:10.1093/nar/gkac261>

Ghosh S et al. (2020). Nearest-neighbor parameters for predicting DNA duplex stability in diverse molecular crowding conditions. PNAS 117:14194-14201. <doi:10.1073/pnas.1920886117>

Ghosh S et al. (2023). Nearest-neighbor parameters for the prediction of RNA duplex stability in diverse in vitro and cellular-like crowding conditions. NAR 51:4101-4111. <doi:10.1093/nar/gkad020>

Blake RD, Delcourt SG (1996). Thermodynamic effects of formamide on DNA stability. NAR 24:2095-2103. <doi:10.1093/nar/24.11.2095>

See Also

tm_calculate for a single entry point to the nearest-neighbor, GC-content and Wallace methods.

Examples

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

# A parameter set fitted at a stated sodium concentration. Because Na
# matches the concentration the set was fitted at, salt correction is
# skipped automatically rather than applied on top of it.
```

```
out_ls <- tm_nn(seqs, nn_table = "RNA_DNA_NN_Weber_2019_LS", Na = 100)
out_ls$options[["Salt correction applied"]]

# -- A parameter table supplied by the user --------------------------------
# Any of nn_table, tmm_table, imm_table and de_table also accepts a matrix,
# which is how a set the package does not ship, most obviously one covering
# modified bases, is used without waiting for a new release.
#
# Start from a built-in set to get the required keys, then add a stack. The
# key naming follows the built-in convention: top strand, "/", bottom
# strand, so "MG/CG" would be a 5-methylcytosine followed by G, paired with
# CG. The values here are illustrative and are NOT measured parameters.
tbl <- TmCalculator:::get_table("DNA_NN_SantaLucia_2004")
tbl <- rbind(tbl, "MG/CG" = c(-9.1, -24.0))

# Optional attributes. "reference" names the built-in whose key set must be
# present, which matters for RNA and hybrid tables because the default
# reference is a DNA/DNA set. "salt_mM" marks the table as fitted at a
# stated sodium concentration, so that it suppresses the salt correction at
# that concentration exactly as the built-in condition-specific sets do.
attr(tbl, "reference") <- "DNA_NN_SantaLucia_2004"

out_user <- tm_nn(seqs, nn_table = tbl, Na = 50)
out_user$options[["Thermodynamic NN values"]]
# "user-supplied (reference: DNA_NN_SantaLucia_2004)"

# Passing a built-in table back in through this route changes nothing: the
# supplied table is reordered to the reference key order before use, so row
# order carries no information.
identical(tm_nn(seqs, nn_table = "DNA_NN_SantaLucia_2004")$gr$Tm,
          tm_nn(seqs,
                nn_table = TmCalculator:::get_table("DNA_NN_SantaLucia_2004")
                )$gr$Tm)

# A table missing a key is refused rather than tolerated. The compiled core
# resolves each stack by name, so an absent key would contribute zero
# enthalpy and entropy to every sequence containing that step, silently.
try(tm_nn(seqs, nn_table = tbl[setdiff(rownames(tbl), "AA/TT"), ]))
out_ls$options[["Parameter set fitted at [Na+] (mM)"]]
```

`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	Sequence(s) in 5' to 3' direction, as the GRanges that to_genomic_ranges returns. A character vector of sequences, a path to a FASTA file, or genomic coordinate strings are also accepted and converted for you. To pair a sequence with a complement of your own, build the object explicitly: to_genomic_ranges(seq, complement_seq = cmp).
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

Length of validity

The 2 + 4 rule was calibrated on 14 to 20 nt hybridisation probes and carries no length-dependent, salt-dependent or concentration-dependent term. Its error therefore grows without bound as sequences lengthen, and on genome-scale windows it returns a number that is not a melting temperature in any useful sense: a 200 bp window is reported at several hundred degrees Celsius.

Sequences longer than 30 nt raise a warning that names tm_nn as the appropriate alternative. A warning rather than an error, because the rule stays a legitimate rule of thumb and users who knowingly apply it outside its calibrated range should not be blocked; wrap the call in suppressWarnings() in that case.

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, in the same formats `input_seq` accepts. `NULL` (default) generates them. Supply the **plain complement**, aligned base for base with `input_seq`: position `i` of `complement_seq` is the base paired with position `i` of `input_seq`, so written underneath it the complement runs **3' to 5'**. Same length, same order, no reversal.

```
input_seq       5'-G C A T C G-3'
complement_seq  3'-C G T A G C-5'
```

The *reverse* complement ("CGATGC" here) is the same strand written 5' to 3', which is what `Biostrings::reverseComplement()` and a supplier's order form give you. Passing it pairs every position against the wrong base; the call warns when it detects this.

A complement that differs from the plain one at some positions is a mismatched duplex and is handled as such.

`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)
```

`$.TmCalculator` *Lazy* `$df` *accessor for* `TmCalculator` *objects*

Description

`result$df` is computed on demand from `result$gr` instead of being stored eagerly, so genome-scale results are not converted to a data.frame unless actually requested. All other elements behave as plain list access.

Usage

```
## S3 method for class 'TmCalculator'
x$name
```

Arguments

`x`	A `TmCalculator` object.
`name`	Element name.

Value

The element, or a data.frame built from `x$gr` when `name == "df"` and no stored `df` exists.

Index

∗ **datasets**
    ecoli_rep_hotspots, 9
    thermodynamic_gc_params, 33
    thermodynamic_nn_params, 34
$.TmCalculator, 61

BSgenome, *20*

check_filter_seq, 3
chem_correct, 4
compare_groups, 5, *10*, *27, 28*
complement_fast, 7
coor_to_genomic_ranges, 8, *58, 59*

ecoli_rep_hotspots, 9

fa_to_genomic_ranges, 10

gc_content, 11
generate_complement, 12
getSeq, *8*

integrate_granges, 13, *28*

legend, *24*
letterFrequency, *20*

make_genomiccoord, *8*, 18

p.adjust, *28*
pairwise.t.test, *6*
pairwise.wilcox.test, *6*
plot_genome_track, *9, 10*, 22
plot_tm, 27
print.TmCalculator, 30

reduce, *16*

salt_correct, 31

thermodynamic_gc_params, 33
thermodynamic_nn_params, 34
tm_calculate, *9*, *13*, 37, *55*
tm_gc, *20*, 45, *51*
tm_nn, *20*, *36*, *39*, *42*, *44*, *47*, 48, *57*
tm_wallace, *20*, 56
to_genomic_ranges, *9*, *12*, *46*, *49*, *57, 58*
to_genomic_ranges
        *(*to_genomic_ranges_fast*)*, 58
to_genomic_ranges_fast, *9*, 58

vec_to_genomic_ranges, 60