# Package 'TmCalculator'

June 4, 2026

**Type** Package

**Title** Extending Nucleic Acid Melting Temperature Analysis from Sequence-Level Computation to Genome-Wide Thermodynamic Profiling

**Version** 1.0.5

**Date** 2026-05-29

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

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

**License** MIT + file LICENSE

**Depends** R (>= 3.5)

**VignetteBuilder** knitr

**Imports** seqinr, methods, BSgenome, Biostrings, GenomicRanges, IRanges, S4Vectors, GenomeInfoDb, Gviz, dplyr, ggbio, ggplot2, karyoploteR, circlize, graphics, grDevices, viridis, rlang, plotly

**Suggests** testthat (>= 3.0.0), knitr, rmarkdown, devtools, roxygen2, remotes, BiocManager, BSgenomeForge, BSgenome.Hsapiens.UCSC.hg38

**NeedsCompilation** no

**Repository** CRAN

**RoxygenNote** 7.3.2

**LazyData** true

**Encoding** UTF-8

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

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

**Date/Publication** 2026-06-04 15:00:11 UTC

# Contents

check_filter_seq *Filter invalid bases in nucleotide sequences*

## Description

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

## Usage

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

## Arguments

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

## Value

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

## Author(s)

Junhui Li

## References

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

chem_correct *Corrections of melting temperature with chemical substances*

## Description

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

## Usage

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

## Arguments

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

## Details

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

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

## Author(s)

Junhui Li

## References

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

## Examples

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

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

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

---

`compare_groups` *Compare numeric GRanges metadata across groups*

---

## Description

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

## Usage

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

## Arguments

| | |
|---|---|
| `gr` | A `GRanges` object. |
| `target` | Character vector of metadata column names to test (e.g. `c("Tm", "GC")`). All must be numeric. |
| `method` | Statistical test family. One of:<br>• `"wilcoxon"`: rank-based tests (Wilcoxon / Kruskal-Wallis).<br>• `"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 $\geq 3$ 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) |
| $\geq 3$ | 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 $\geq 3$ 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:<br>• A list with `pkg_name` (BSgenome package name) and `seq` (character vector of coordinate strings). Preferred for many windows on the same genome.<br>• A plain character vector of coordinate strings (legacy format; genome package name is read from field 4 when present).<br>Supported colon-separated formats:<br>• `chr:start-end:strand:region_id` - requires `pkg_name` in the input list.<br>• `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:<br>`"vectorized"` One `getSeq()` call per genome package (default).<br>`"preload_chr"` Load each chromosome once and extract windows with `subseq()`. Faster for dense whole-chromosome tiling; uses more memory. |

## Details

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

## Value

A GRanges object with metadata columns:

sequence Reference sequence for each interval.

complement Complementary sequence.

GC GC fraction (0-1) per interval.

region_id Region identifier from the coordinate string.

genome_pkg BSgenome package name used.

## Author(s)

Junhui Li

## See Also

to_genomic_ranges, to_genomic_ranges_fast

## Examples

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

## End(Not run)
```

---

# ecoli_rep_hotspots *Example *E. coli* replication-hotspot dataset*

---

## Description

A named list of data.frame objects used in bacterial genome examples.

## Usage

```
data(ecoli_rep_hotspots)
```

## Format

A named list of data.frames (peaks, binned tracks, and ssDNA regions).

## fa_to_genomic_ranges *Convert FASTA file to GenomicRanges object*

### Description

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

### Usage

```
fa_to_genomic_ranges(input_seq)
```

### Arguments

| | |
|---|---|
| input_seq | Path to the input FASTA file |

### Value

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

### Examples

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

## gc *Calculate G and C content of nucleotide sequences*

### Description

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

### Usage

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

### Arguments

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

### Value

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

### Author(s)

Junhui Li

### Examples

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

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

---

## generate_complement *Generate complementary sequence*

---

### Description

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

### Usage

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

### Arguments

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

## Value

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

## Author(s)

Junhui Li

## References

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

## Examples

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

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

---

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

---

## Description

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

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

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

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

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

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

## Usage

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

## Arguments

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

## Value

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

## Author(s)

Junhui Li

## Examples

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

# -- Sample data ------------------------------------------------------------
set.seed(42)
gr_tm <- GRanges(
  seqnames = c(rep("chr1", 60), rep("chr2", 30)),
  ranges   = IRanges(
    start = c(sort(sample(1:249e6, 60)),
              sort(sample(1:243e6, 30))),
    width = sample(50:200, 90, replace = TRUE)
  ),
  tm = runif(90, 55, 85),
  gc = runif(90, 30, 70)
)

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

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

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

# Strategy 3: window - 5 kb window around each probe
```

```
res_window <- integrate_granges(gr_tm, gr_features,
                                strategy = "window", window_size = 5000)

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

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

## End(Not run)
```

---

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

---

## Description

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

## Usage

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

## Arguments

| | |
|---|---|
| bsgenome | A BSgenome object (e.g. BSgenome.Hsapiens.UCSC.hg38) **or** a character string with the BSgenome package name (loaded automatically). |
| chromosomes | Character vector of chromosome names to tile. Must be present in seqlevels(bsgenome). Default: the 24 standard human chromosomes paste0("chr", c(1:22, "X", "Y")). |
| window | Integer. Width of each sliding window in base pairs. Default 200L. |
| slide | Integer. Step size between consecutive window starts in base pairs. slide == window gives non-overlapping tiling; slide < window gives overlapping windows. Default 50L. |
| start | Integer **or** NULL. Override the start position for every chromosome. When NULL (default) the start is chromosome position 1 (adjusted for N-trimming if trim_N != "none"). Can be a named integer vector to set different starts per chromosome. |
| end | Integer **or** NULL. Override the end position for every chromosome. When NULL (default) the end is the chromosome length (adjusted for N-trimming). Can be a named integer vector. |
| strand | Character. Strand label embedded in the coordinate string. One of "+" (default) or "-". |
| trim_N | Character. N-base handling strategy. One of "ends" (default), "filter", or "none". See section *N-base trimming* above. |
| max_N_frac | Numeric in $[0, 1]$. Maximum fraction of N bases tolerated per window. Windows exceeding this threshold are dropped. Only used when trim_N = "filter". Default 0.1. |
| N_scan_block | Integer. Block size (bp) for the coarse N-end detection scan used by trim_N = "ends". Larger values are faster but less precise. Default 10000L. |
| region_prefix | Character. Prefix for region IDs embedded in the coordinate string. Default "region" (producing region1, region2, ...). |
| genome_pkg_name | Character **or** NULL. The genome package name to embed in the coordinate string (4th field). When NULL (default), the name is extracted automatically from the bsgenome object via S4Vectors::metadata(bsgenome). Supply explicitly when using a custom BSgenome object whose metadata name differs from the canonical package name. |
| as_vector | Logical. When TRUE (default), return a plain named character vector. When FALSE, return a data.frame with columns coord, chr, start, end, strand, genome, region_id, chr_start_used, chr_end_used – useful for downstream GRanges construction. |
| verbose | Logical. Print per-chromosome progress messages. Default TRUE. |

## Value

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

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

### Coordinate string format

Each element of the returned vector follows the pattern:

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

Fields (colon-separated):

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

### N-base trimming

Human (and most mammalian) chromosomes begin and end with long stretches of `N` bases that represent assembly gaps. Windows that consist entirely or predominantly of `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_circos_genome` *Plot circular genome tracks using circlize*

---

## Description

Generate a circular genome plot with flexible track definitions using the circlize package. Each track is defined in a list, allowing dynamic visualization of genomic data such as GC content, Tm, and peak regions.

## Usage

```
plot_circos_genome(
  genome_name,
  genome_size,
  track_list,
  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.cex = 0.6,
  axis.show.unit = FALSE,
  legend.show = TRUE,
```

```
  legend.position = c(0.75, 1),
  legend.cex = 0.6,
  legend.lwd = 1,
  legend.seg.len = -0.5,
  legend.box.lwd = 0.25,
  legend.x.intersp = 1,
  legend.bty = "n",
  legend.border = NA,
  legend.lty = 1,
  label = NULL,
  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,
  title.cex = 0.7
)
```

## Arguments

| | |
|---|---|
| genome_name | Character. Genome name. |
| genome_size | Numeric. Genome size. |
| track_list | List. Each element defines a track with fields such as:<br>• type: "line" or "rect"<br>• data: data.frame or GRanges<br>• ylim: numeric vector of length 2<br>• value_col: column name (for line tracks)<br>• col: color<br>• bg.border, bg.col, bg.lwd: optional track background settings |
| start.degree | Numeric. Starting angle of the plot. |
| track.height | Numeric. Height of each track. |
| gap.after | Numeric. Gap between sectors. |
| cell.padding | Numeric vector of length 4. |
| track.margin | Numeric vector of length 2. |
| circle.margin | Numeric vector of length 2. |
| canvas.xlim | Numeric vector. |
| canvas.ylim | Numeric vector. |
| axis.unit | Character. Axis unit. |
| axis.step | Numeric. Axis step. |
| axis.cex | Numeric. Axis label size. |
| axis.show.unit | Logical. Whether to show axis unit. |

`legend.show` Logical. Whether to show legend.

`legend.position`
Numeric vector. Legend position.

`legend.cex` Numeric. Legend label size.

`legend.lwd` Numeric. Legend line width.

`legend.seg.len` Numeric. Legend segment length.

`legend.box.lwd` Numeric. Legend box line width.

`legend.x.intersp`
Numeric. Legend x interspacing.

`legend.bty` Character. Legend box type.

`legend.border` Character. Legend box border color.

`legend.lty` Numeric. Legend line type.

`label` Optional GRanges/data.frame for labels.

`label.column` Numeric. Label column.

`label.niceFacing`
Logical. Whether to face labels.

`label.cex` Numeric. Label size.

`label.side` Character. Label side.

`label.labels_height`
Numeric. Label labels height.

`label.connection_height`
Numeric. Label connection height.

`label.line_lwd` Numeric. Label line width.

`title.cex` Numeric. Title size.

## Value

A circular plot.

## Author(s)

Junhui Li

---

`plot_karyotype_genome` *Plot multi-track karyotype genome view using karyoploteR*

---

## Description

Generate a linear genome plot with flexible track definitions using the karyoploteR package. Each track is defined in a list, allowing dynamic visualization of genomic data such as GC content, Tm, peak regions, and other multi-omics signals. The input format mirrors `plot_circos_genome` for consistency across visualization modes.

## Usage

```
plot_karyotype_genome(
  genome_assembly,
  track_list,
  chromosomes = "auto",
  plot_type = 1,
  track.margin = 0.05,
  zoom = NULL,
  tick_dist = 1e+07,
  axis_cex = 0.6,
  chr_cex = 0.8,
  legend.show = TRUE,
  legend.position = "topright",
  legend.cex = 0.7,
  legend.bty = "n",
  label = NULL,
  label.column = NULL,
  label.cex = 0.6,
  title_name = NULL,
  title.cex = 0.8
)
```

## Arguments

`genome_assembly`
: Character string specifying the genome assembly (e.g., "hg19", "mm10"), or any genome specification accepted by `plotKaryotype` (GRanges, Seqinfo, etc.).

`track_list`
: List. Each element defines one track. Supported fields (identical to `plot_circos_genome` where applicable):

- **type**: Character. Drawing type - one of `"points"`, `"line"`, `"bars"`, or `"rect"`. Default: `"points"`.
- **data**: GRanges or data.frame with columns seqnames/chr, start, end, and the value column.
- **min_bar_width**: Numeric (0-1). For `bars` type only. Minimum bar width as a fraction of chromosome length. Needed when sequences are short (e.g. primers / probes) whose actual genomic footprint is sub-pixel at whole-chromosome scale. Default: `0.002` (0.2% of chr length, ~500 kb on chr1). Set to `0` to use actual range widths only.
- **value_col**: Character. Metadata column containing numeric values (required for `points`, `line`, `bars`).
- **ylim**: Numeric vector of length 2. Y-axis limits. Inferred from data if `NULL`.
- **col**: Color for data and legend symbol. Auto-assigned if `NULL`.
- **name**: Character. Track label shown in legend.
- **r0**, **r1**: Numeric (0-1). Vertical position of the track within the data panel. Auto-computed if `NULL`.
- **bg.col**: Track background fill color. Default: `"white"`.

- **bg.border**: Track background border color. Default: NA.
- **bg.lwd**: Track background border line width. Default: 0.5.
- **lwd**: Line width (for line type). Default: 1.
- **cex**: Point size (for points type). Default: 0.5.
- **pch**: Point shape (for points type). Default: 16.
- **ytop**, **ybottom**: Top/bottom y values for rect tracks within ylim range. Defaults: 0.98 / 0.02.
- **border**: Border color for rect and bars tracks. Default: NA.

chromosomes
: Character vector of chromosomes to plot, or "auto" to derive from the data, or any value accepted by karyoploteR (e.g., "canonical", "autosomal"). Default: "auto".

plot_type
: Integer. karyoploteR plot type (1 = horizontal chromosomes stacked vertically, 2 = two panels, etc.). Default: 1.

track.margin
: Numeric. Fractional gap between adjacent tracks. Default: 0.05.

zoom
: A GRanges or character string (e.g., "chr1:1e6-2e6") specifying a region to zoom into. NULL shows full chromosomes.

tick_dist
: Numeric. Distance between x-axis tick marks in base pairs. Default: 10000000.

axis_cex
: Numeric. Axis label text size. Default: 0.6.

chr_cex
: Numeric. Chromosome name text size. Default: 0.8.

legend.show
: Logical. Whether to display the legend. Default: TRUE.

legend.position
: Character. Legend position passed to legend (e.g., "topright", "topleft"). Default: "topright".

legend.cex
: Numeric. Legend label text size. Default: 0.7.

legend.bty
: Character. Legend box type ("n" for no box). Default: "n".

label
: Optional GRanges whose metadata column label.column contains text labels to plot at each range.

label.column
: Character or integer. The metadata column in label that holds label text.

label.cex
: Numeric. Label text size. Default: 0.6.

title_name
: Character. Main plot title. NULL omits the title.

title.cex
: Numeric. Title text size. Default: 0.8.

## Details

Track vertical positions (r0/r1) are auto-computed by dividing the data panel equally among tracks, separated by track.margin. Supply explicit r0/r1 values in individual track list entries to override the automatic layout.

The track_list format is intentionally identical to plot_circos_genome so that the same track definitions can be reused across circular and linear genome representations.

## Value

Invisibly returns the karyoploteR KaryoPlot object, enabling further customisation with karyoploteR functions.

## Author(s)

Junhui Li

## Examples

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

# -- Sample data ------------------------------------------------------------
# -- Typical workflow: output from tm_calculate() already has Tm AND GC ------
# tm_calculate() stores both columns in result$gr, so one GRanges object
# is sufficient for all tracks.
#
# result <- tm_calculate(my_seqs, method = "tm_nn")
# gr_tm  <- result$gr   # has $Tm (deg C) and $GC (%) columns

# -- Example 1: Tm points + GC bars from the same GRanges -------------------
set.seed(42)
gr_tm <- GRanges(
  seqnames = c(rep("chr1", 50), rep("chr2", 30)),
  ranges = IRanges(
    start = c(sort(sample(1:249e6, 50)), sort(sample(1:243e6, 30))),
    width = sample(50:200, 80, replace = TRUE)
  ),
  Tm = runif(80, 55, 85),
  GC = runif(80, 30, 70)   # gc() returns percentages (0-100)
)
gr_peaks <- GRanges(
  seqnames = c("chr1", "chr1", "chr2"),
  ranges = IRanges(start = c(10e6, 100e6, 50e6), width = c(1e6, 2e6, 500e3))
)

track_list <- list(
  list(type = "points", data = gr_tm, value_col = "Tm",
       col = "#e41a1c", name = "Tm (deg C)"),
  list(type = "bars",   data = gr_tm, value_col = "GC",
       col = "#377eb8", name = "GC (%)"),
  list(type = "rect",   data = gr_peaks,
       col = "#4daf4a", name = "Peaks")
)
plot_karyotype_genome(
  genome_assembly = "hg19",
  track_list      = track_list,
  title_name      = "Multi-omics - hg19"
)

# -- Example 2: line track, zoom --------------------------------------------
plot_karyotype_genome(
  genome_assembly = "hg19",
  track_list      = list(
    list(type = "line", data = gr_tm, value_col = "Tm", col = "#984ea3", name = "GC (%)")
```

```
  ),
  zoom           = "chr1:1000000-50000000",
  title_name     = "Tm - chr1 zoom"
)

## End(Not run)
```

---

`plot_linear_genome` *Plot linear genome tracks using karyoploteR*

---

## Description

Linear equivalent of plot_circos_genome(). Takes the same track_list format so the same input can produce either view.

## Usage

```
plot_linear_genome(
  genome_name,
  genome_size = NULL,
  genome = NULL,
  track_list,
  label = NULL,
  chromosomes = NULL,
  plot.type = 1,
  track.gap = 0.01,
  legend.show = TRUE,
  legend.position = "topright",
  legend.cex = 0.6,
  legend.bty = "n",
  legend.border = NA,
  axis.cex = 0.5,
  title.cex = 0.9,
  base.tick.dist = NULL
)
```

## Arguments

| | |
|---|---|
| `genome_name` | Character. Used for the title. |
| `genome_size` | Numeric. Total length (single-chromosome genomes like E. coli). Alternatively pass ‘genome‘ directly (GRanges) for multi-chromosome plots. |
| `genome` | Optional GRanges describing the genome. If NULL, built from genome_name + genome_size as a single contig. |
| `track_list` | List of track specs (see above). |
| `label` | data.frame/GRanges with seqnames/start/end/label (like circos version). |

| | |
|---|---|
| chromosomes | Optional character vector to restrict/reorder chromosomes. |
| plot.type | karyoploteR plot.type (1 = tracks above only, 2 = above+below). |
| track.gap | Relative gap between tracks (0 to ~0.05). |
| legend.show | Logical. |
| legend.position | Passed to graphics::legend (e.g. "topright", or x/y). |
| legend.cex, legend.bty, legend.border | See graphics::legend. |
| axis.cex | Base numbers size. |
| title.cex | Title size. |
| base.tick.dist | Numeric or NULL. Distance between base-number ticks passed to karyoploteR::kpAddBaseNumbers. When NULL, a default is chosen based on total genome size. |

## Details

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 (ribbon from 0 to value) - type = "region": piled-up regions via kpPlotRegions (good for "ssDNA"-style tracks) - type = "coverage": coverage area plot from a set of regions

Each list element may also contain: name, col, bg.col, border, ylim, lwd

## Value

Invisibly returns the KaryoPlot object.

---

# plot_tm_genome_tracks *Plot Tm values as Genome Browser Tracks using Gviz*

---

## Description

This function generates Gviz plots displaying Tm values as DataTracks alongside genome axes and ideograms for specified chromosomes. The display type (gradient lines, points, connected line, or bars) is configurable, and coloring can follow a continuous Tm gradient or any categorical metadata column (e.g., a "group" annotation). Multiple zoom windows on the same chromosome can be compared side by side.

## Usage

```
plot_tm_genome_tracks(
  gr,
  chromosome_to_plot,
  genome_assembly = NULL,
  tm_track_title = "Melting Temperature (°C)",
  color_palette = c("viridis", "magma", "plasma", "inferno", "cividis"),
```

```
  show_ideogram = TRUE,
  zoom = NULL,
  x_axis = c("genome", "regions"),
  track_type = c("gradient", "points", "line", "bars"),
  color_by = "Tm",
  facet_by_zoom = FALSE,
  regions_point_size = 2,
  regions_show_bars = TRUE
)
```

## Arguments

| | |
|---|---|
| gr | A GRanges object. Must contain a metadata column `Tm` with numeric melting temperature values. |
| chromosome_to_plot | Character string. Chromosome to visualise; must exist in `gr`. |
| genome_assembly | Character string specifying the genome assembly (e.g., `"hg19"`, `"mm10"`). |
| tm_track_title | Character. Track / axis title. Default: `"Melting Temperature (\u00B0C)"`. |
| color_palette | Character. Viridis palette: `"viridis"` (default), `"magma"`, `"plasma"`, `"inferno"`, or `"cividis"`. |
| show_ideogram | Logical. Display chromosome ideogram. Default: `TRUE`. |
| zoom | Character string or character vector specifying genomic region(s) to zoom into, e.g. `"chr1:10e6-20e6"` or `c(Mut = "chr1:100-1000", WT = "chr1:5e6-5.001e6")`. When the vector is *named* the names become region labels. Multiple entries produce side-by-side panels in `"genome"` mode and filtered, labelled data in `"regions"` mode. `NULL` (default) shows the full chromosome extent. |
| x_axis | Character. X-axis mode:<br>• `"genome"` (default): Standard Gviz genome-browser view.<br>• `"regions"`: ggplot2 view with region index on the x-axis, suitable for sparse data spread across a large chromosome. |
| track_type | Character. How Tm values are drawn:<br>• `"gradient"` (default): Vertical lines coloured by a continuous Tm gradient (lollipop view). Not compatible with discrete `color_by`; automatically switches to `"points"` when a categorical `color_by` is requested.<br>• `"points"`: Scatter points, coloured per-point by Tm gradient or by category.<br>• `"line"`: Connected line through all regions.<br>• `"bars"`: Bar / histogram display. |
| color_by | Character. Variable mapped to colour. Options:<br>• `"Tm"` (default): continuous viridis gradient over Tm values.<br>• `"zoom_region"`: one colour per zoom window – requires `zoom` to have two or more entries.<br>• Any metadata column name present in `gr` (e.g. `"group"`): treated as a discrete categorical variable. |

`facet_by_zoom` Logical. In `"regions"` mode with multiple zoom windows, split into one ggplot2 facet per window. Default: `FALSE`.

`regions_point_size` Numeric. Point size in `"regions"` mode. Default: `2`.

`regions_show_bars` Logical. Deprecated; superseded by `track_type`. Retained for backward compatibility.

## Value

In `"regions"` mode: a `ggplot` object. In `"genome"` mode: `invisible(NULL)` (Gviz renders directly).

## Examples

```
## Not run:
library(GenomicRanges)
set.seed(123)
gr <- GRanges(
  seqnames = rep("chr1", 100),
  ranges   = IRanges(start = sort(sample(1:249250621, 100)),
                     width = sample(50:200, 100, replace = TRUE)),
  Tm    = runif(100, 60, 80),
  group = sample(c("mutation", "wildtype"), 100, replace = TRUE)
)
# Standard gradient view
plot_tm_genome_tracks(gr, "chr1", "hg19",
                      zoom = "chr1:10062800-20000000")

# Points colored by Tm gradient (fixed)
plot_tm_genome_tracks(gr, "chr1", "hg19",
                      track_type = "points",
                      zoom = "chr1:10062800-20000000")

# Compare two regions side by side, colored by region
plot_tm_genome_tracks(gr, "chr1", "hg19",
  zoom      = c(Region1 = "chr1:7634457-133482943",
                Region2 = "chr1:135756721-248931747"),
  track_type = "points",
  color_by   = "zoom_region")

# Color by metadata group column
plot_tm_genome_tracks(gr, "chr1", "hg19",
  track_type = "points",
  color_by   = "group")

# Regions mode - index on x-axis, multiple zoom windows
p <- plot_tm_genome_tracks(gr, "chr1", "hg19",
  x_axis   = "regions",
  zoom     = c(Region1 = "chr1:100-200000000",
               Region2 = "chr1:200000000-249250621"),
  color_by = "zoom_region")
```

```
print(p)

## End(Not run)
```

---

plot_tm_heatmap *Plot Tm values as a heatmap using ggplot2*

---

## Description

This function generates a heatmap visualization of Tm values across chromosomes using ggplot2. It supports both a genome-coordinate view (faceted by chromosome, x-axis = genomic position) and a region-indexed view (x-axis = region index, y-axis = Tm value) that makes sparse data legible at any genomic scale.

## Usage

```
plot_tm_heatmap(
  gr,
  genome_assembly = NULL,
  chromosome_to_plot = NULL,
  plot_type = c("karyogram", "faceted"),
  color_palette = c("viridis", "magma", "plasma", "inferno", "cividis"),
  title_name = NULL,
  zoom = NULL,
  x_axis = c("genome", "regions"),
  regions_facet = FALSE,
  regions_color_by = c("Tm", "chromosome")
)
```

## Arguments

| | |
|---|---|
| gr | A GRanges object. It MUST contain a metadata column named 'Tm' with numeric melting temperature values. |
| genome_assembly | A character string indicating the genome assembly (e.g., "hg19", "mm10"). This is used for setting chromosome lengths when needed. |
| chromosome_to_plot | A character vector specifying which chromosomes to visualize. These chromosomes must exist in your GRanges object. |
| plot_type | A character string specifying the type of plot to generate:<br>• "karyogram" (default): Chromosomes arranged as rows, each region shown as a colored rectangle at its genomic position.<br>• "faceted": Same as "karyogram" but with more whitespace between chromosome panels.<br>Ignored when `x_axis = "regions"`. |

color_palette A character string specifying the viridis color palette to use. Available options are:

- "viridis" (default)
- "magma"
- "plasma"
- "inferno"
- "cividis"

title_name A character string for the plot title.

zoom A character string specifying the genomic region to zoom into. If NULL (default), the entire range of each chromosome will be shown. Example: c("chr1:1000000-2000000", "chr2:1000000-2000000"). Only used when `x_axis = "genome"`.

x_axis Character. Controls how the x-axis is constructed:

- `"genome"` (default): x-axis represents genomic position; regions are drawn as colored rectangles in chromosomal space.
- `"regions"`: x-axis represents a sequential region index (ordered by chromosome then start position). y-axis shows Tm value. This view makes it easy to compare Tm values across all input regions regardless of how far apart they are in the genome.

regions_facet Logical. When `x_axis = "regions"`, facet the plot by chromosome so each panel has its own x index. Default: `FALSE`.

regions_color_by Character. When `x_axis = "regions"`, map colour to `"Tm"` (default, continuous gradient) or `"chromosome"` (discrete per-chromosome colour).

## Value

A plotly object (interactive).

## Examples

```
## Not run:
# Create example GRanges object
gr_tm <- GenomicRanges::GRanges(
  seqnames = c("chr1", "chr2", "chr1", "chr2", "chr1"),
  ranges = IRanges::IRanges(
    start = c(100, 200, 300, 400, 150),
    end = c(150, 250, 350, 450, 200)
  ),
  Tm = c(65.5, 68.2, 70.1, 63.8, 72.0)
)

# Genome-coordinate view (default)
plot_tm_heatmap(gr_tm, genome_assembly = "hg19", plot_type = "karyogram")

# Region-indexed view: compare Tm values across all regions
plot_tm_heatmap(gr_tm, genome_assembly = "hg19", x_axis = "regions")

# Region view, faceted and coloured by chromosome
```

```
plot_tm_heatmap(gr_tm, genome_assembly = "hg19", x_axis = "regions",
                regions_facet = TRUE, regions_color_by = "chromosome")

## End(Not run)
```

---

`plot_tm_heatmap_interactive` *Convert Tm plots to interactive plotly versions*

---

## Description

These functions convert the standard Tm plots to interactive plotly versions that can be used in Shiny applications or R Markdown documents. Each function mirrors the parameters of its non-interactive counterpart and adds an `x_axis` argument that switches between genome-coordinate and region-indexed views.

## Usage

```
plot_tm_heatmap_interactive(
  gr,
  genome_assembly = NULL,
  chromosome_to_plot = NULL,
  plot_type = c("karyogram", "faceted"),
  color_palette = c("viridis", "magma", "plasma", "inferno", "cividis"),
  title_name = NULL,
  zoom = NULL,
  x_axis = c("genome", "regions"),
  regions_facet = FALSE,
  regions_color_by = c("Tm", "chromosome")
)

plot_tm_karyotype_interactive(
  gr,
  chromosomes = NULL,
  genome_assembly = NULL,
  colors = NULL,
  shapes = NULL,
  plot_type = 1,
  point_cex = 1.5,
  xaxis_cex = 0.7,
  yaxis_cex = 0.8,
  chr_cex = 1,
  tick_dist = 1e+07,
  zoom = NULL,
  x_axis = c("genome", "regions")
)
```

```
plot_tm_genome_tracks_interactive(
  gr,
  chromosome_to_plot,
  genome_assembly = NULL,
  tm_track_title = "Melting Temperature (°C)",
  color_palette = c("viridis", "magma", "plasma", "inferno", "cividis"),
  show_ideogram = TRUE,
  zoom = NULL,
  x_axis = c("genome", "regions"),
  regions_show_bars = TRUE
)
```

## Arguments

| | |
|---|---|
| gr | A GRanges object containing the Tm values. |
| genome_assembly | A string specifying the genome assembly. |
| chromosome_to_plot | A string specifying the chromosome to plot. |
| plot_type | Integer. karyoploteR plot type (ignored in regions mode). |
| color_palette | A string specifying the viridis palette. |
| title_name | A string specifying the plot title. |
| zoom | A string specifying the zoom region (genome mode only). |
| x_axis | Character. "genome" for chromosomal coordinates; "regions" for region-indexed view. |
| regions_facet | Logical. (heatmap only) Facet the regions view by chromosome. Default: FALSE. |
| regions_color_by | Character. (heatmap only) Colour regions by "Tm" or "chromosome". Default: "Tm". |
| chromosomes | A character vector specifying chromosomes to plot. |
| colors | A named vector of colours per chromosome. |
| shapes | A named integer vector of pch values per chromosome. |
| point_cex | Numeric. Point size (genome mode) or point size multiplier (regions mode). Default: 1.5. |
| xaxis_cex | Numeric. X-axis label size. Default: 0.7. |
| yaxis_cex | Numeric. Y-axis label size. Default: 0.8. |
| chr_cex | Numeric. Chromosome label size. Default: 1. |
| tick_dist | Numeric. X-axis tick spacing in bp. Default: 10000000. |
| tm_track_title | A string specifying the track title. |
| show_ideogram | Logical. Show ideogram (genome mode only). |
| regions_show_bars | Logical. Add lollipop bars in regions mode. Default: TRUE. |

## Value

A plotly object.

A plotly object.

---

`plot_tm_linear` *Plot Tm values distributed across GRanges regions on the x-axis*

---

## Description

Creates a ggplot2-based linear plot where the x-axis represents individual GRanges regions rather than full chromosomal coordinates. This addresses the sparsity problem of whole-genome views: when probe or primer sequences are scattered across a large genome, plotting against chromosomal position compresses all data into a tiny fraction of the x-axis. `plot_tm_linear` instead places each region sequentially on the x-axis so that Tm values are visible at the scale of the input data.

## Usage

```
plot_tm_linear(
  gr,
  x_axis = c("index", "label", "position"),
  color_by = "chromosome",
  color_palette = c("viridis", "magma", "plasma", "inferno", "cividis"),
  facet_by_chr = FALSE,
  sort_by = c("position", "Tm"),
  add_line = FALSE,
  point_size = 2,
  x_label_angle = 45,
  title_name = NULL,
  ylab = "Tm (°C)",
  show_legend = TRUE
)
```

## Arguments

| | |
|---|---|
| `gr` | A `GRanges` object containing a `Tm` metadata column with numeric melting temperature values. |
| `x_axis` | Character. How to represent regions on the x-axis:<br>• `"index"` (default): Regions are numbered 1, 2, ..., N in chromosomal order (chr then start). Clean and compact.<br>• `"label"`: Each tick is labelled `"seqname:start-end"`. Informative but verbose; use `x_label_angle` to rotate labels.<br>• `"position"`: The x-axis shows genomic midpoints; a `facet_wrap` by chromosome is applied automatically so that each panel is zoomed to its data range. |
| `color_by` | Character. Variable mapped to point colour: |

- "chromosome" (default): Each chromosome gets a distinct colour from the chosen viridis palette.
- "Tm": A continuous viridis colour gradient over Tm values.
- Any other metadata column name present in gr (e.g., "group"): treated as a discrete categorical variable - each unique level gets a distinct colour from the chosen viridis palette. Useful for comparing categories such as "mutation regions" vs. "non-mutation".

color_palette — Character. Viridis palette to use for colouring. One of "viridis" (default), "magma", "plasma", "inferno", or "cividis".

facet_by_chr — Logical. Split the plot into one facet per chromosome. Forced to TRUE when x_axis = "position". Default: FALSE.

sort_by — Character. Order in which regions appear on the x-axis (x_axis = "index" or "label" only):

- "position" (default): chromosomal order (seqname then start).
- "Tm": ascending Tm value.

add_line — Logical. Connect points with a thin line (useful for showing trends). Default: FALSE.

point_size — Numeric. Size passed to geom_point. Default: 2.

x_label_angle — Numeric. Rotation angle for x-axis text. Most useful when x_axis = "label". Default: 45.

title_name — Character. Plot title. Defaults to a generic string when NULL.

ylab — Character. Y-axis label. Default: "Tm (deg C)".

show_legend — Logical. Display the colour legend. Default: TRUE.

## Details

Regions are sorted by chromosome (natural order) then start position before indexing unless sort_by = "Tm". The returned ggplot uses theme_bw() and can be further customised with standard ggplot2 layers.

## Value

A ggplot object. Print it to render, or convert to an interactive view with plotly::ggplotly().

## Author(s)

Junhui Li

## Examples

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

# -- Sample data ------------------------------------------------------------
set.seed(1)
gr <- GRanges(
```

```
  seqnames = c(rep("chr1", 40), rep("chr2", 30), rep("chr3", 20)),
  ranges = IRanges(
    start = c(sort(sample(1:249e6, 40)),
              sort(sample(1:243e6, 30)),
              sort(sample(1:198e6, 20))),
    width = sample(50:150, 90, replace = TRUE)
  ),
  Tm = runif(90, 55, 85)
)

# -- Example 1: Default - index on x, colour by chromosome -----------------
plot_tm_linear(gr)

# -- Example 2: Region labels on x-axis -------------------------------------
plotly::ggplotly(plot_tm_linear(gr, x_axis = "label", x_label_angle = 60))

# -- Example 3: Colour by Tm, sorted by Tm ----------------------------------
plotly::ggplotly(plot_tm_linear(gr, color_by = "chromosome", sort_by = "Tm", add_line = TRUE))

# -- Example 4: Per-chromosome positional view ------------------------------
plot_tm_linear(gr, x_axis = "position", color_by = "Tm",
               color_palette = "magma")

# -- Example 5: Faceted by chromosome, index x-axis -------------------------
plot_tm_linear(gr, facet_by_chr = TRUE, color_by = "Tm",
               color_palette = "plasma")

# -- Example 6: Colour by a categorical metadata column ("group") ----------
gr$group <- sample(c("mutation regions", "non-mutation"), 90, replace = TRUE)
plot_tm_linear(gr, color_by = "group")
plotly::ggplotly(plot_tm_linear(gr, color_by = "group", add_line = FALSE))


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

### Arguments

| | |
|---|---|
| Na | Millimolar concentration of sodium ions. Default: 0 |
| K | Millimolar concentration of potassium ions. Default: 0 |
| Tris | Millimolar concentration of Tris buffer. Default: 0 |
| Mg | Millimolar concentration of magnesium ions. Default: 0 |
| dNTPs | Millimolar concentration of deoxynucleotide triphosphates. Default: 0 |
| method | Method for calculating salt concentration corrections to the melting temperature. Available options: - "Schildkraut2010": Updated salt correction method - "Wetmur1991": Classic salt correction method - "SantaLucia1996": DNA-specific salt correction - "SantaLucia1998-1": Improved DNA salt correction - "SantaLucia1998-2": Alternative DNA salt correction (requires input_seq) - "Owczarzy2004": Comprehensive salt correction (requires input_seq) - "Owczarzy2008": Updated comprehensive salt correction (requires input_seq) Note: Setting to NA disables salt correction |

`input_seq` Sequence (5' to 3') of one strand of the nucleic acid duplex. Can be provided as either: - A character string (e.g., "ATGCG") - A path to a FASTA file containing the sequence(s) Required for methods: "SantaLucia1998-2", "Owczarzy2004", and "Owczarzy2008"

`ambiguous` Logical. If TRUE, ambiguous bases are taken into account when computing the G and C content. The function handles various ambiguous bases (S, W, M, K, R, Y, V, H, D, B) by proportionally distributing their contribution to GC content based on their possible nucleotide compositions.

## Details

Different correction methods are available for various experimental conditions:

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

## Author(s)

Junhui Li

## References

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

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

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

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

## Examples

```
salt_correction(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_correction** 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 = $\Delta H$, right = $\Delta S$). Rownames are dinucleotide step keys such as `"AT/TA"`, mismatch / dangling-end keys, or feature-specific keys (e.g. `"CAG_4"` for the (CAG)4 CNG entry).

### Populated tables

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

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

**DNA_NN_Allawi_1998** DNA/DNA NN, Allawi (1998)

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

**RNA_NN_Freier_1986** RNA/RNA NN, Freier (1986)

**RNA_NN_Xia_1998** RNA/RNA NN, Xia (1998)

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

**RNA_DNA_NN_Sugimoto_1995** RNA/DNA hybrid NN, Sugimoto (1995)

**DNA_IMM_Peyret_1999** DNA single internal mismatch, Peyret (1999)

**DNA_TMM_Bommarito_2000** DNA terminal mismatch, Bommarito (2000)

**DNA_DE_Bommarito_2000** DNA single dangling end, Bommarito (2000)

**RNA_DE_Turner_2010** RNA single dangling end, Turner (2010)

### Registered placeholders (values TBD - populate from primary literature)

**DNA_NN_AllSan_1997** Allawi & SantaLucia (1997) Biochemistry 36:10581

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

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

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

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

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

**RNA_IMM_Davis_Znosko_2007** Davis & Znosko (2007) Biochemistry 46:13425

**RNA_IMM_Davis_Znosko_2008** Davis & Znosko (2008) Biochemistry 47:10178

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

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

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

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

**RNA_DE_Serra_2008** O'Toole / Serra (2006, 2008)

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

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

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

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

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

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

**DNA_IL_SantaLucia_2004** SantaLucia & Hicks (2004)

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

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

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

**DNA_SBL_SantaLucia_2004** SantaLucia & Hicks (2004)

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

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

**DNA_LBL_SantaLucia_2004** SantaLucia & Hicks (2004)

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

**RNA_GU_Mathews_1999** Mathews & Turner (1999) JMB 288:911

**RNA_GU_Chen_2012** Chen / Serra (2012)

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

**DNA_INO_SantaLucia_2005** Watkins & SantaLucia (2005) NAR 33:6258

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

**DNA_HA_Kawakami_2001** Kawakami / Sugimoto (2001) Biochemistry 40:14040

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

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

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

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

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

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

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

## Details

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

## Source

Various publications as cited above.

## References

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

## Examples

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

# Access DNA internal mismatch parameters
thermodynamic_nn_params$DNA_IMM_Peyret_1999

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

---

# tm_calculate *Calculate melting temperature using multiple methods*

---

## Description

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

## Usage

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

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

## Arguments

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

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

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

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

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

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

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

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

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

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

Na — Millimolar concentration of sodium ions. Default: 50

K — Millimolar concentration of potassium ions. Default: 0

Tris — Millimolar concentration of Tris buffer. Default: 0

Mg — Millimolar concentration of magnesium ions. Default: 0

dNTPs — Millimolar concentration of deoxynucleotide triphosphates. Default: 0

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

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

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

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

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

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

## Details

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

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

## Value

A `TmCalculator` list with:

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

## Available Options

### Method Selection:

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

### Nearest Neighbor (NN) Method Options:

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

### GC Method Options:

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

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

**Salt Correction Options:**

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

**Formamide Unit Options:**

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

**Other Parameters:**

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

## Author(s)

Junhui Li

## Examples

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

## End(Not run)
```

---

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

---

## Description

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

## Usage

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

## Arguments

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

`ambiguous` Logical. If TRUE, ambiguous bases are taken into account when computing the G and C content. The function handles various ambiguous bases (S, W, M, K, R, Y, V, H, D, B) by proportionally distributing their contribution to GC content based on their possible nucleotide compositions.

`userset` A vector of four coefficient values. Usersets override value sets.

`variant` Empirical constants coefficient with 8 variants: - Chester1993: Tm = 69.3 + 0.41(Percentage_GC) - 650/N - QuikChange: Tm = 81.5 + 0.41(Percentage_GC) - 675/N - Percentage_mismatch - Schildkraut1965: Tm = 81.5 + 0.41( - Wetmur1991_MELTING: Tm = 81.5 + 0.41( - Wetmur1991_RNA: Tm = 78 + 0.7( - Wetmur1991_RNA/DNA: Tm = 67 + 0.8( - Primer3Plus: Tm = 81.5 + 0.41( - vonAhsen2001: Tm = 77.1 + 0.41(

Salt correction is applied only for variants that include it in the formula (via `salt_correction()`). Chester1993 and QuikChange use no salt term. D is the mismatch penalty (typically 1): Tm decreases by D x ( Use `X` (or `.`) in the sequence to mark mismatch positions.

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

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

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

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

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

`salt_method` Salt correction method. `NULL` (default) uses the method associated with `variant`. Set to `NA` to disable salt correction. Options: - "Schildkraut2010": Schildkraut & Lifson 1965 - "Wetmur1991": Wetmur 1991 - "SantaLucia1996": SantaLucia 1996 - "SantaLucia1998-1": SantaLucia 1998 (Method 1) - "Owczarzy2004": Owczarzy 2004 - "Owczarzy2008": Owczarzy 2008 Note: "SantaLucia1998-2" is not available for this function.

`mismatch` Logical. If TRUE (default), every 'X' in the sequence is counted as a mismatch

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

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

`dmso_factor` Coefficient of Tm decreases per percent DMSO. Default: 0.75 (von Ahsen et al. 2001) Other published values are 0.5, 0.6 and 0.675.

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

## Value

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

## Author(s)

Junhui Li

## References

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

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

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

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

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

## Examples

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

---

# tm_nn *Calculate melting temperature using nearest neighbor thermodynamics*

---

## Description

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

## Usage

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

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

## Arguments

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

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

## Details

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

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

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

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

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

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

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

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

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

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

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

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

## Author(s)

Junhui Li

## References

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

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

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

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

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

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

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

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

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

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

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

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

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

## Examples

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

---

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

---

## Description

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

## Usage

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

## Arguments

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

## Value

Returns a list of sequences with updated Tm attributes

## Author(s)

Junhui Li

## References

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

## Examples

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

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

## Description

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

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

## Usage

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

to_genomic_ranges(input_seq, complement_seq = NULL)
```

## Arguments

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

## Value

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

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

## Author(s)

Junhui Li

## Examples

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

## End(Not run)

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

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

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

## End(Not run)
```

vec_to_genomic_ranges *Convert sequence strings to GenomicRanges object*

## Description

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

## Usage

```
vec_to_genomic_ranges(input_seq)
```

## Arguments

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

## Value

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

## Examples

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

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

# Index