# Genome-analysis-tools manual
# version 1.0.0

December XX, 2016
Koji Masuda

## Contents



## 1 Overview

These tools are designed for analyzing the peaks and/or read distributions derived from next-generation sequencing such as ChIP-seq, MNase-seq or RNA-seq. For peaks, BED format as well as any tab-delimited table with chromosome name, start position and end position are available. For reads, compressed WIG format (.wig.gz,

either one file for whole chromosomes or each file for each chromosome is OK) and bedGraph (.bedgraph) are supported. To make comparison between samples possible, these reads should be normalized by using softwares such as DROMPA (Nakato et al., 2013) or BEDTools (Quinlan and Hall, 2010). Each tool and option are explained in detail with sample files in the following section.

## 2 Requirement

Genome-analysis-tools are written in ANSI C and can be executed on Linux OS. Genome-analysis-tools requires the following libraries:

- GCC compiler (http://gcc.gnu.org/)
- GNU C Library(glibc) (http://www.gnu.org/software/libc/)
- zlib (http://www.zlib.net/)
- (optional) R (http://www.r-project.org/)

## 3 Installation

Install through git:

```
$ git clone git@github.com:KojiMasuda/genome-analysis-tools.git
$ cd genome-analysis-tools
$ make
```

Install through source file:

```
$ wget https://github.com/KojiMasuda/genome-analysis-tools/archive/master.zip
$ unzip master.zip
$ cd genome-analysis-tools-master
$ make
```

If you get an installation error, make sure that all required libraries are installed.

Add the directory you installed the tools to your PATH environment variable. If you installed it in "$HOME/genome-analysis-tools" directory, add the following line to your Bash startup file such as $HOME/.bashrc;

```
$ export PATH=$PATH:HOME/genome-analysis-tools
```

# 4 Genome-analysis-tools

In general, <command> without any argument or <command> -h/--help shows command usage. Also, <command> -v shows version information. In most tools, the first line of input file is considered as header if --header option (default:off) is specified. To support as many format as possible, you can specify the number of column. For example, if you have tab-delimited file of which start position is on 5th column, set --col_start 4 (note that column number option is zero-based).

## 4.1. ga_overlap

**ga_overlap** reports the overlaps of two peaks/summits.
Among output files;

```
peak1_over_peak2.txt          is peak1 which overlap peak2,
peak1_nonover_peak2.txt       is peak1 which doesn't overlap
peak2,
peak1_vs_peak2.txt            is peak1 with over/nonover flags,
peak1_vs_peak2_summary.txt    is overlapping summary.
```

If you need overlapping of peak1 to peak2 with header:

```
$ ga_overlap -1 sample/peak1.txt -2 sample/peak2.txt --header
```

If the input files don't have headers, just remove --header option.

In addition, if you need *the number* of overlapping peak2, specify --count option:

```
$ ga_overlap -1 sample/peak1.txt -2 sample/peak2.txt --header --count
```

You may want information of overlapping peak2, specify --preserve2 option:

```
$ ga_overlap -1 sample/peak1.txt -2 sample/peak2.txt --header --preserve2
```

Or you may want to compare *the summit* of peak1, specify the start and the end column to the summit (summit is on 4th column in this case):

```
$ ga_overlap -1 sample/peak1.txt -2 sample/peak2.txt --header --col_start1 3
  --col_end1 3
```

Note that the column number is zero-based.

In some case, you may want to use the fixed length as a peak width. In that case, specify --hw <int> option as a half-width (total width is 2xhw):

```
$ ga_overlap -1 sample/peak1.txt -2 sample/peak2.txt --header --col_start1 3
  --col_end1 3 --hw 250 ##total width is 500 bp
```

Type ga_overlap -h for the detail explanation of options.

## 4.2. ga_reads_summit

**ga_reads_summit** reports the average distribution of reads around summits or specific positions such as transcription start sites (TSS) or transcription end sites (TES).
Output file is made in the directory of --sig file. In the output file;

```
1st column <relative_pos>: distance from peak summit
2nd column <smt_mean>: average read values around summit
3rd column <CI95.00percent_U>: Upper 95% confidence interval
4th column <CI95.00percent_L>: Lower 95% confidence interval
5th column <smtNb>: summit number
6th column <Centered>: centered peak name
7th column <Signal>: signal(read) name
```

If you have peak sets and want to calculate the average ChIP read distribution;

```
$ ga_reads_summit --smt sample/peak1.txt --sig sample/chip1/chip1 --sigfmt sepwiggz --col_start 3 --col_end 3 --header

$ Rscript sample/R/script1.r
```

When you specify --sigfmt sepwiggz (separated WIG.GZ file for each chromosome), set file name before "_chr*.wig.gz" as read file. In this case it's "sample/chip1/chip1". In the above example, set --col_start 3 and --col_end 3 to calculate the read distribution around summit positions, which is stored on 4th column in peak1.txt file. The second command creates png file of the output (Figure 1)

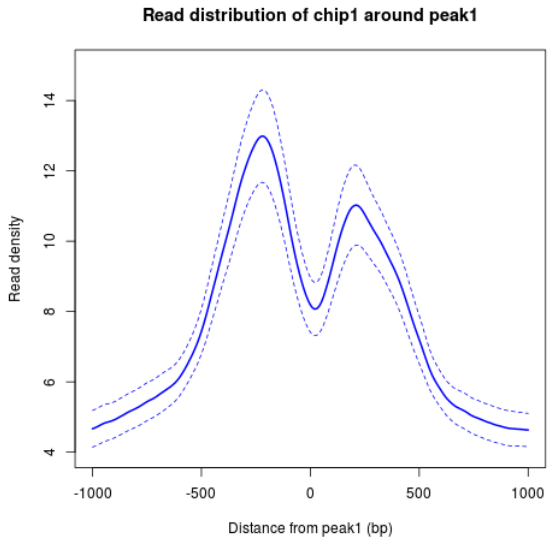


Figure 1. Read distribution of chip1 around peak1. Dashed lines are upper and lower 95 % confidence intervals.

If you have input file as a denominator, signals from --sig file are divided by signals from --sig_d file by specifying --sig_d option;

```
$ ga_reads_summit --smt sample/peak1.txt --sig sample/chip1/chip1 --sigfmt
```

```
    sepwiggz --col_start 3 --col_end 3 --header --sig_d sample/input1/input1
$   Rscript sample/R/script2.r
```

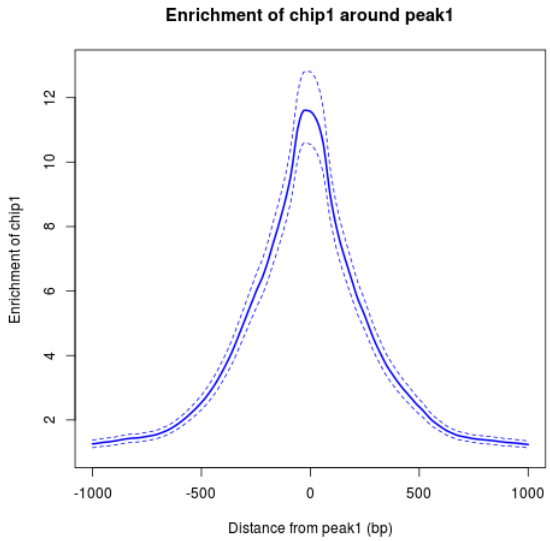


Figure 2. Distribution of chip1 enrichment around peak1. Dashed lines are upper and lower 95 % confidence intervals.

The second command creates png file of the output file (Figure 2).

In some case, you may want to know the read distribution throughout the genome. If you specify --rand <int> option, the positions from 1 to the genome length is randomly picked up (the number of the position is same as the peak numbers), then read density is calculated around the picked-up positions. This process is iterated for <int> times then the mean values are reported;

```
$   ga_reads_summit --smt sample/peak1.txt --sig sample/chip1/chip1 --sigfmt
    sepwiggz --col_start 3 --col_end 3 --header --sig_d sample/input1/input1 --rand
    1000 --gt sample/genome_table.txt
$   Rscript sample/R/script3.r
```

Don't forget setting genome table by --gt option for random simulation because the tool would know the genome length from this file. The second command creates png file of the output file (Figure 3).

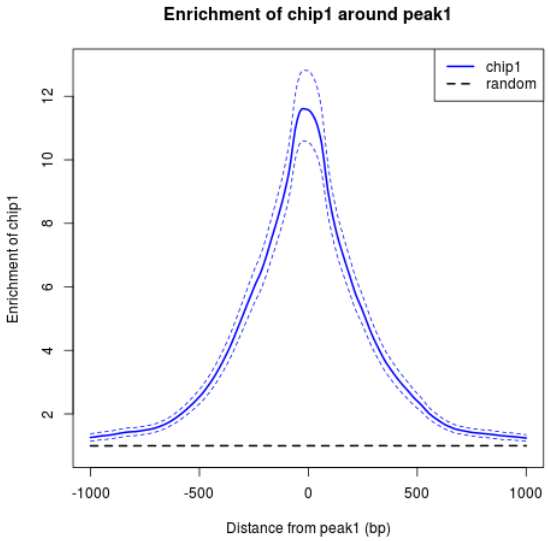


Figure 3. Distribution of chip1 enrichment (blue) around peak1 with enrichment from randomly picked up positions throughout genome (dashed black line). Dashed blue lines are upper and lower 95 % confidence intervals.

If you want to know the read distribution around TSS, specify proper start and end positions by --col_start and --col_end options, respectively. Generally speaking, each gene has its strandness. Thus let the tool know the column for the strandness of annotation file by --col_strand option;

```
$ ga_reads_summit --smt sample/anno.txt --sig sample/nuc.wig.gz --sigfmt
  onewiggz --col_chr 2 --col_start 4 --col_end 5 --col_strand 3

$ Rscript sample/R/script4.r
```

Note that “onewiggz” is specified for --sigfmt argument because “sample/nuc.wig.gz” is WIG.GZ file which contains data for whole chromosome. When you specify --col_strand, the tool checks whether the gene has '+' or '-'. The second command creates png file of the output file (Figure 4).

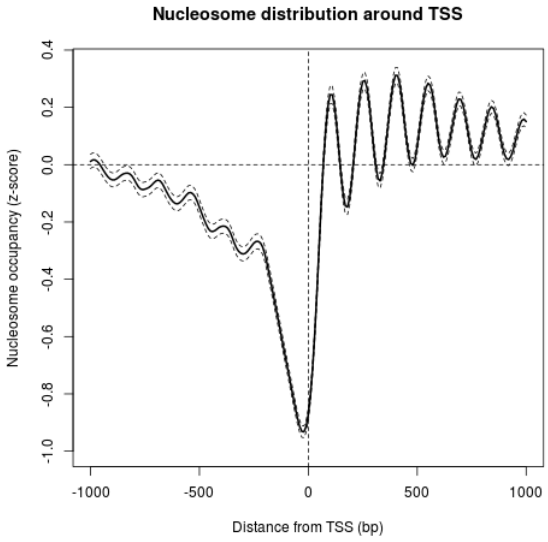

Figure 4. Nucleosome occupancy around TSS. Dashed lines are upper and lower 95 % confidence intervals.

Set appropriate half-window size (--hw), step size (--step) and window size for read calculation (--win) for your sample (default is suitable for yeast sample). Type ga_reads_summit -h for the detail explanation of options.

### 4.3. ga_reads_summit_all

**ga_reads_summit_all** reports the read distributions around ALL summits/peaks. The usage is quite similar to **ga_reads_summit** tool (see section 4.2).
If you have mammalian ChIP reads, specifying only --sig argument is recommended because of sparse input reads;

```
$ ga_reads_summit_all --smt sample/anno_mm10.bed --sig sample/chip2/chip2
  --sigfmt sepwiggz --col_strand 5 --hw 5000 --step 100 --win 500

$ Rscript sample/R/script5.r
```

Note that --col_strand argument is specified for strandness of annotation genes. Half-window size (--hw), step size (--step) and window size for read calculation (--win) are set for this sample. The row number and column number of output file is the number of peaks (+header) and the number of windows (2xhalf-window/step + 1), respectively. The second command creates png file of the ouput file (Figure 5).

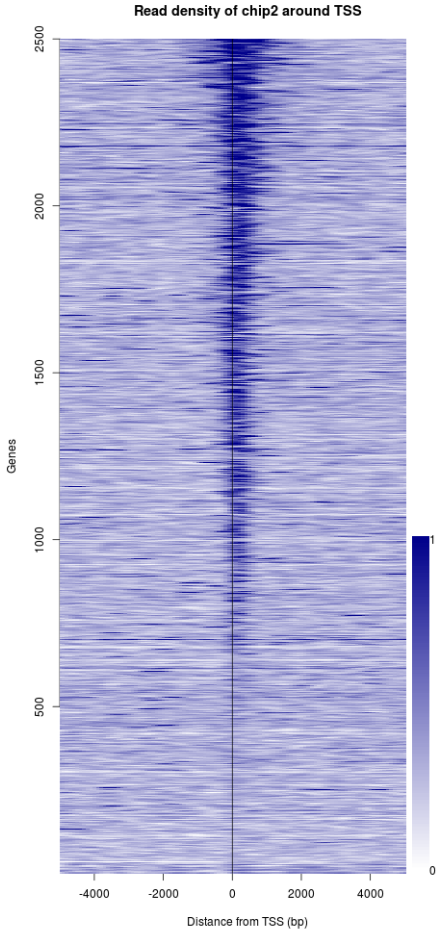


Figure 5. Heat map of chip2 reads around TSS.

If you have yeast ChIP reads, you can specify --sig_d argument as an input as well as --sig argument as a ChIP;

```
$ ga_reads_summit_all --smt sample/peak1.txt --sig sample/chip1/chip1 --sigfmt sepwiggz --col_start 3 --col_end 3 --header --sig_d sample/input1/input1 --win 300

$ Rscript sample/R/script6.r
```

The second command creates png file of the output file (Figure 6).

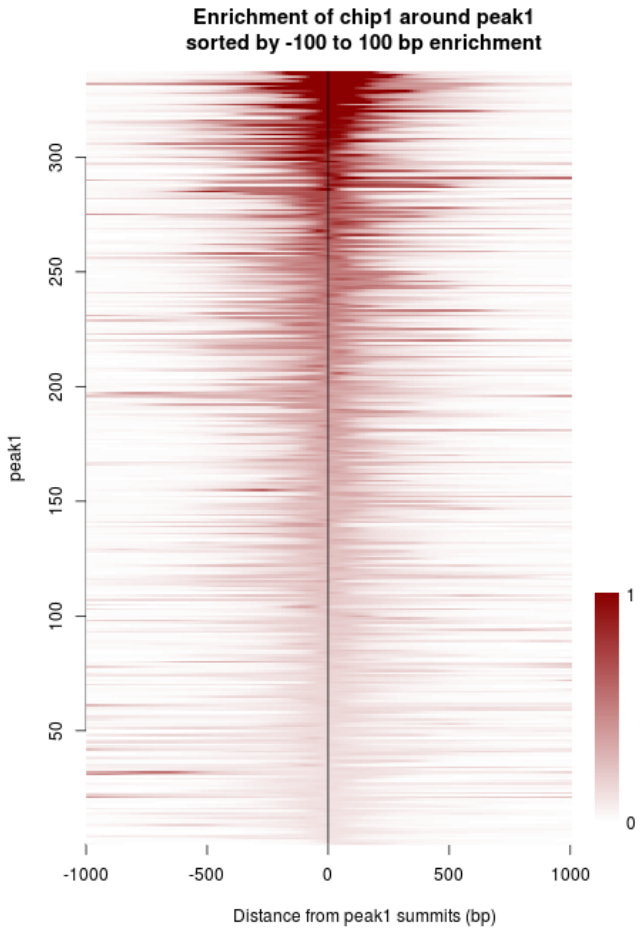


Figure 6. Enrichment of chip1 around peak1.

### 4.4. ga_calc_dist

**ga_calc_dist** reports the distance of two peaks(mode:two) or inter-summit distance(mode:isd). If you have one peak set and want the inter-summit distance;

```
$ ga_calc_dist -1 sample/peak1.txt -mode isd --header
```

"isd" is set as -mode argument for inter-summit distance calculation.

In some case, you may want to know the distance between two peak sets. When you set "two" mode, **ga_calc_dist** calculate the distance from each peak1 to *the closest peak2*;

```
$ ga_calc_dist -1 sample/peak1.txt -2 sample/peak2.txt -mode two --header
```

The distance from each peak1 to the closest peak2 is added to the right most column.

### 4.5. ga_reads_region

**ga_reads_region** reports the amount of reads inside regions.
You may be interested in read amounts inside gene, up-stream of TSS, down-stream of TES, or around summits. With specific "region" or "smt" modes, you can calculate the read amount inside regions you're interested in. The basic usages are;

```
$ ga_reads_region [options] --smt <file_summit> --sig <file_signal> --sigfmt
  <sig format:bedgraph | sepwiggz | onewiggz> --mode smt --col_smt <column
  of summit>
```

or

```
$ ga_reads_region [options] --smt <file_summit> --sig <file_signal> --sigfmt
  <sig format:bedgraph | sepwiggz | onewiggz> --mode <region mode: region |
  up-tss | tss-dw | up-tss-dw | up-tes | tes-dw | up-tes-dw>
```

Set --smt, --sig, --sigfmt arguments as explained in section 4.2. You can set suitable --mode argument for your purpose. See scheme in Figure 7 for detail. By specifying --col_strand option, the tool consider strandness of region such as gene.

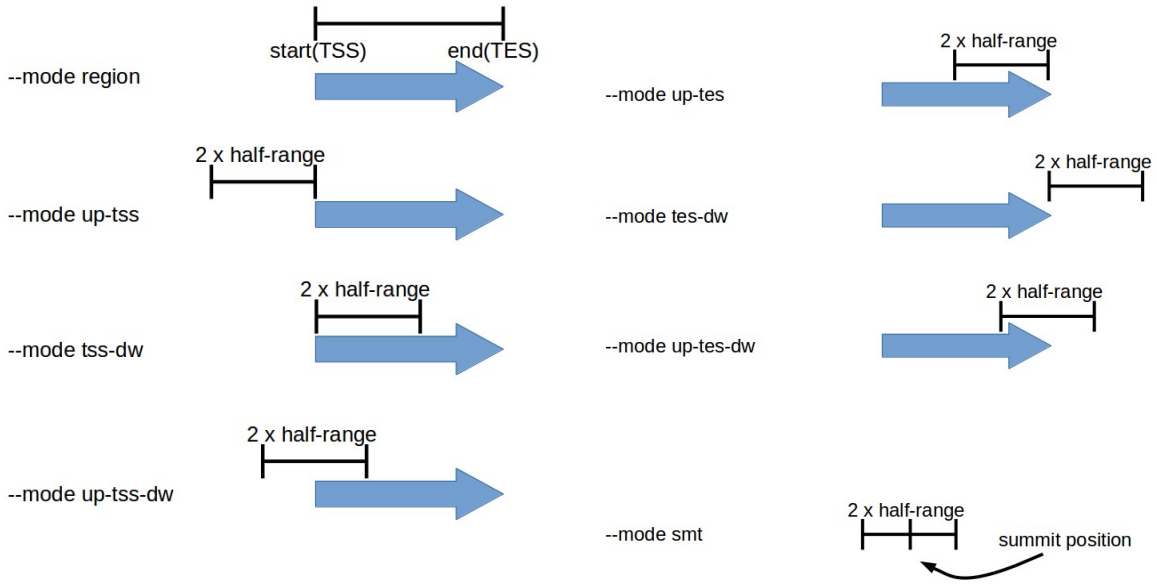


Figure 7: The graphical explanation of mode options. When “region” is specified as --mode argument, start and end position is used and half-range (--hw) is ignored. On the other hand, fixed region length is specified by half-range as --hw argument when other mode is specified.

Note that the amount of reads inside regions is normalized by region length.
The amount of reads are added to the right most column.

### 4.6. ga_nuc_region

**ga_nuc_region** reports the nucleotide composition(number of ATCG) and AT content. The tool is useful if you want the nucleotide composition of regions you're interested in;

```
$ ga_nuc_region -fa sample/fasta1.fa -region sample/peak1.txt -gt
  sample/genome_table.txt --header
```

The nucleotide composition is added to the right most column.

### 4.7. ga_nuc_summit

**ga_nuc_summit** reports the nucleotide composition(number of ATCG) or AT content around summit. The row and column of output file represents each peak and window such as output file of **ga_reads_summit_all** (see section 4.3). The usage is also similar to **ga_reads_summit_all**, and set --smt, --col_chr, --col_start, --col_end, --col_strand arguments as **ga_reads_summit_all**. If you want AT contents around peak1;

```
$ ga_nuc_summit -fa sample/fasta1.fa -smt sample/peak1.txt -gt
  sample/genome_table.txt -n_flag AT --col_start 3 --col_end 3 --header

$ Rscript sample/R/script7.r
```

The second command creates png file of average AT content of output file (Figure 8).

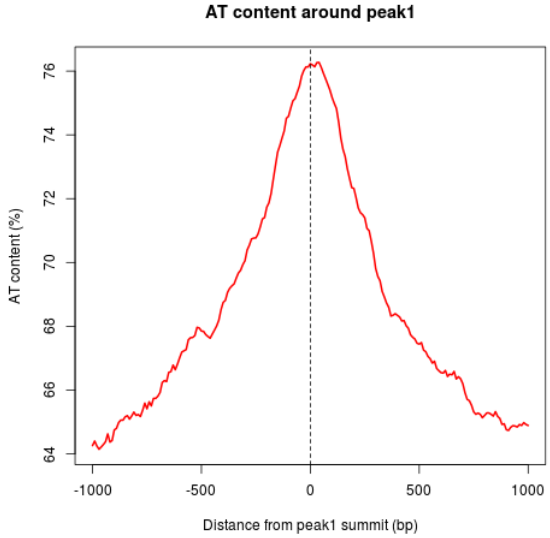


Figure 8. Average AT content around peak1.

## 4.8. ga_deltaG

**ga_deltaG** makes the wiggle file of the free energy(delta-G) difference between the duplex and single-strand states from fasta file. With default setting, delta-G is calculated as a sum of experimentally determined DNA stability of nearest-neighbor values within the window. For the nearest-neighbor values, two libraries are available for now (Breslauer et al., 1986; SantaLucia et al., 1996). For example;

```
$  ga_deltaG -fa sample/fasta1.fa -gt sample/genome_table.txt -lib
   SantaLucia_1996 -win 200 -step 10
```

If --gradient option is specified, coefficients are used to consider the effect of position within the window for the energy calculation. The middle of the window is coeffient of 1, and it linearly decreases to 0 for both left and right side;

```
$  ga_deltaG -fa sample/fasta1.fa -gt sample/genome_table.txt -lib
   SantaLucia_1996 -win 200 -step 10 --gradient
```

### 4.9. ga_RPKM

**ga_RPKM** reports the expression level as reads per kilobase per million mapped reads (RPKM). Only bedgraph format is available as an expression file for now. Note that before using the tool, the expression file MUST BE normalized by total mapped million read number, i.e. mapped reads should be divided by 10 if mapped read number is 10 M. This can be achieved by "genomecov" command in BEDTools (Quinlan and Hall, 2010). The most simple example is;

```
$ ga_RPKM -exp sample/RNA1.bedgraph -ref sample/anno.txt -readlen 90
```

The tool has additional --consid_ov option. This makes some difference if two or more genes overlap *and* your sample is *not* strand-specific. Actually, you can't tell from which gene the reads come. In other wards, the --consid_ov option shouldn't be used if you have strand-specific reads. With this option and --thresh option (default 0.50), if the proportion of overlapped exon is less than (1-thresh) * 100 %, RPKM is calculated from the exons which are not overlapped to improve the accuracy of expression levels. For genes which aren't overlapped, or the proportion of overlapped exon is more than (1-thresh) * 100 %, RPKM is normally calculated;

```
$ ga_RPKM -exp sample/RNA1.bedgraph -ref sample/anno.txt -readlen 90
  --consid_ov --thresh 0.7
```

In the output file;

```
Ref + 1 column <RPKM>: expression levels in RPKM
Ref + 2 column <sub_exons_st>: newly defined exon start positions
(if --consid_ov)
Ref + 3 column <sub_exons_ed>: newly defined exon end positions
(if --consid_ov)
Ref + 4 column <ov_genes>: overlapping genes (if exist and
--consid_ov)
Ref + 5 column <ov_prop>: proportion of overlapped exon
```

# 5 References


Breslauer, K.J., Frank, R., Blöcker, H., and Marky, L.A. (1986). Predicting DNA duplex stability from the base sequence. Proc. Natl. Acad. Sci. U. S. A. *83*, 3746–3750.

Nakato, R., Itoh, T., and Shirahige, K. (2013). DROMPA: Easy-to-handle peak calling and visualization software for the computational analysis and validation of ChIP-seq data. Genes to Cells *18*, 589–601.

Quinlan, A.R., and Hall, I.M. (2010). BEDTools: A flexible suite of utilities for comparing genomic features. Bioinformatics *26*, 841–842.

SantaLucia, J., Allawi, H.T., and Seneviratne, P.A. (1996). Improved nearest-neighbor parameters for predicting DNA duplex stability. Biochemistry *35*, 3555–3562.