RC25125 (W1103-072) March 10, 2011
Other

IBM Research Report

Genomic Regions Tools for High-throughput Analytics in Genomics

A. Tsirigos, N. Haiminen, E. Bilal
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Genomic regions tools for high-throughput analytics in genomics

A. Tsirigos*, N. Haiminen and E. Bilal

IBM Thomas J. Watson Research Center, PO Box 218, Yorktown Heights, NY 10598, U.S.A.

E-mail: atsirigo@us.ibm.com, nhaimin@us.ibm.com, ebilal@us.ibm.com

* Correspondence should be addressed to A.T. (atsirigo@us.ibm.com)

Abstract

Following the dramatic reduction of sequencing cost, research laboratories have been producing huge amounts of data, measuring DNA variations, RNA abundances, protein-DNA interactions, DNA methylation levels, and even chromosomal conformations. Making sense of terabytes of data requires reliable data management, computational resources, and, eventually, efficient computational methods for preprocessing, quality control, analysis and meta-analysis. In this work, we present a flexible computational platform for accomplishing such computational tasks. Genomic data is represented in our platform as genomic regions, i.e. sets of intervals, effectively covering most popular types of genome-wide data, such as transcripts/genes, exons/introns, promoter sites, sequences, multiple sequence alignments, transcription factor binding sites, intergenic regions, repeat elements, microarray probes (expression, SNP, CNV, etc), sequencing data (RNA-seq, ChIP-seq, DNA-seq, etc), chromosomal conformations (3C-seq, 4C-seq, etc), or inter-chromosomal associations. Our computational platform implements a variety of elementary and composite mathematical operations between sets of regions, so as to enable the prototyping of computational pipelines that can address a wide spectrum of computational tasks, from preprocessing and quality control to meta-analyses. More specifically, the user can easily create average read profiles across transcriptional start sites or enhancer sites, quickly prototype customized peak discovery methods for ChIP-seq experiments, perform genome-wide statistical tests such as enrichment analyses, design controls via user-designed randomization schemes, among other applications.

Introduction

Existing computational tools for large-scale data analytics include Bioconductor, Galaxy, the USCS genome browser, GREAT, and EMBOSS.

Bioconductor [1] provides tools for the analysis and comprehension of high-throughput genomic data. Bioconductor uses the R statistical programming language, and is open source and open development. The functional scope of Bioconductor packages includes the analysis of DNA microarray, sequence, flow, and SNP data.

Galaxy [2] is an open web-based platform for genomic research. Galaxy is based around workflows: reusable template analyses that a user can run repeatedly on different data. An analysis tool can be written in any programming language. Galaxy has been used for different types of genomic research, including investigations of epigenomics, chromatin profiling, transcriptional enhancers, and genome-environment interactions.

The UCSC genome browser [3] is a web-based visualization platform for certain sequenced genomes that incorporates data from several public databases. The browser can be used to visualize user data, and the associated databases can be queried for genomic features of interest for a given region.

The Genomic Regions Enrichment of Annotations Tool (GREAT) [4] was designed to analyze the functional significance of cis-regulatory regions identified by localized measurements of DNA binding events across an entire genome. GREAT incorporates several types of data to perform genome-wide analyses, and is available as a web application.

The European Molecular Biology Open Software Suite (EMBOSS) [5] is a free open source package for molecular biology analyses. EMBOSS integrates a range of currently available packages and tools for sequence analysis. The suite includes tools for sequence alignment, nucleotide sequence pattern analyses, and protein motif identification, among others.

Our genomic regions tools allow for more flexibility compared to the existing ones: different types of data are stored in the same format and can easily be analyzed and compared against each other. The low-level operations we provide can be combined in ways that allow for great complexity in the analyses that can be performed. The same operations apply to various types of input data and allow designing data type independent workflows. Examples of analyses that can be performed with the genomic regions tools are given in the Applications section.

Motivation

Genomic regions nicely complement existing tools such as Bioconductor by enabling large-scale computational analyses of hundreds of sequencing datasets. This is achieved by three key design features of our platform: (a) “command-line” version for batch computations, (b) single-pass processing with minimal memory requirements, and (c) pipeline design to minimize the requirement of intermediate files.

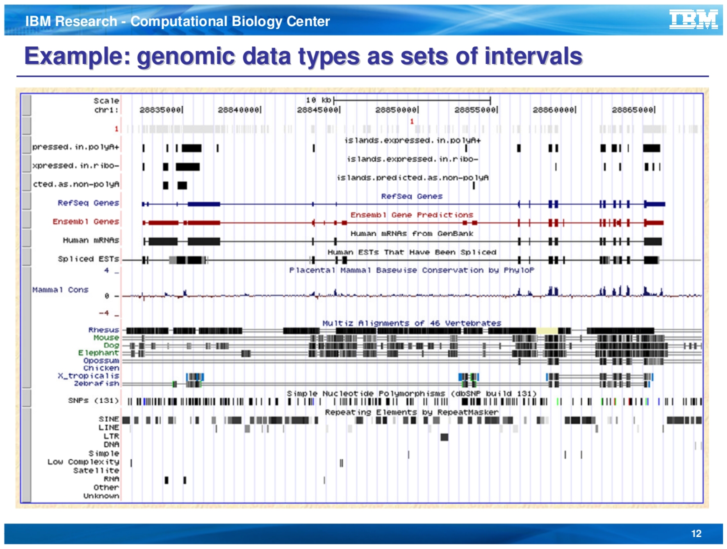
Our platform is compatible with the popular BED format used by the UCSC genome browser and can also convert to WIGGLE format for visualization in the browser.

Our computational platform enables the development of complex analytics for efficient batch processing of hundreds of datasets with minimal memory requirements. The results can be easily converted to UCSC genome browser formats for visualization or used as input to other computational platforms such as the Bioconductor for further analyses.

The various types of input genomic data are represented as sets of intervals stored in files that are processed by the various tools. Many of the tools rely on sorting the input into a specific order that enables scanning and comparing various input files without the need for intermediate files. The interval-based file format is intuitive and simple, yet containing all the necessary information for running complex analyses.

The genomic tools include C++ classes that facilitate the development of further analytics based on the existing code. Complex statistical test can be designed to add functionality to the current tools.

Genome-wide data, such as transcripts/genes, exons/introns, promoter sites, sequences, multiple sequence alignments, transcription factor binding sites, intergenic regions, repeat elements, microarray probes (expression, SNP, CNV, etc), sequencing data (RNA-seq, ChIP-seq, DNA-seq, etc), chromosomal conformations (3C-seq, 4C-seq, etc), or inter-chromosomal associations can easily be represented as sets of genomic intervals (see Figure 1).

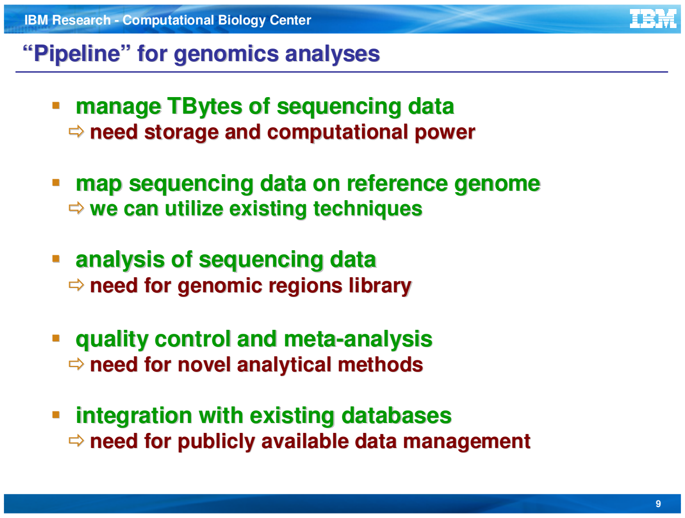


IBM Research - Computational Biology Center

Example: genomic data types as sets of intervals

Figure 1: Diverse types of genomic data

Eventually, we want to use our genomic tools as building blocks for a flexible pipeline (see Figure 2).



IBM Research - Computational Biology Center

"Pipeline" for genomics analyses

- manage TBytes of sequencing data
  ⇨ need storage and computational power
- map sequencing data on reference genome
  ⇨ we can utilize existing techniques
- analysis of sequencing data
  ⇨ need for genomic regions library
- quality control and meta-analysis
  ⇨ need for novel analytical methods
- integration with existing databases
  ⇨ need for publicly available data management

9

Figure 2: Genomic data analyses pipeline.

Input format

Formally, a genomic region (equivalently, an interval) is an entity with the following five fields in this order: <label, chromosome, strand, start position, end position>. The start position is always smaller or equal to the end position. The first two fields (label, chromosome) are tab-separated, the remaining fields are separated by a space. Names of files in this format are usually given the ending ".reg" indicating they are genomic region files (reg files).

Example set of genomic regions:

Read1<TAB>1 + 100 110
Gene1<TAB>1 + 105 857
Read2<TAB>1 – 95 105

In the above example, Read1 overlaps Gene1 at positions 105—110 on chromosome 1 "+" strand. Read2 does not overlap Gene1 since it is located on the opposite strand. However, the reverse complement of Read2 does overlap Gene1 at location 105 on chromosome 1 "+" strand.

Files in the popular BED used by the USCS genome browser can be given as input to the genomic_regions and genomic_overlaps operations (described in the Appendix). However, there are two limitations regarding the format in which the BED file must be: i) no header lines are allowed (e.g. track name), and ii) no tabs are allowed: the columns should be separated by spaces.

For running the overlap operations described later, the regions need to be sorted first by chromosome, then by strand, then by start position, and finally by end position (the example set of regions above follows this order).

The input can be sorted by one of the following commands, depending on the type of the input file (reg or BED):

```
$ sort +1 -3 +3 -4n +4 -5n file.reg
$ sort +0 -1 +5 -6 +1 -2n +2 -3n file.bed
```

Testing whether a file is sorted can be done by the following command ("file" is either file.reg of file.bed):

```
$ genomic_regions -test file
```

Applications

In this section, we describe some practical applications of our genomics tools: (a) *genomic_regions*, a tool for general mathematical operations on genomic sets of intervals, (b) *genomic_overlaps*, a tool for computing overlaps in different formats, and (c) auxiliary *vectors* and *matrix* tools for basic mathematical operations on vectors and matrices respectively. For a listing of all operations the reader is referred to the Appenix.

1. Sliding window computations

Our first example shows how to analyze RNA-seq data by computing RPKM values across sliding windows on the genome and converting those to the WIG format for visualization. RPKM values are useful for comparing expression of genes or genomic regions within or between experiments. Files in the WIG format can be uploaded as tracks to the UCSC browser for convenient visualization.

The end result is shown in Figure 3 for a segment of the human chromosome 1.

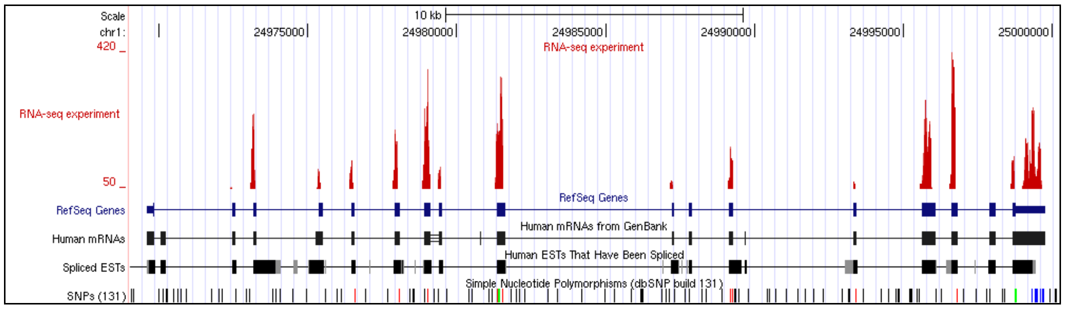


Figure 3: Example WIG file from RNA-seq experiment.

First, RNA-seq reads that have been mapped on the reference genome need to be converted to the reg format. The details on this conversion depend on the output format of the mapping program, and are not discussed here. The first field, label, in the resulting reg file can be set to e.g. "_" for all lines if one does not need to track the read identities. If the chromosome strand (+ or -) of the reads is not known, or if one does not wish to differentiate the reads according to the strand, all reads can be given a strand value "+".

Let the reads be stored in a file called "reads.reg". We need to sort this file with the command:

```
$ sort +1 -3 +3 -4n +4 -5n reads.reg > reads.sorted.reg
```

Next, we need to decide the size of the windows and the sliding step, say we will study windows of size 100 nt with a sliding step of 20 nt (adjacent windows will thus overlap by 80 nt). Scanning the reads and computing the read densities per window can be done with the "-scanc" operation of the genomic_regions tool:

```
$ genomic_regions -scanc –h
```

```
USAGE:
 genomic_regions -scanc [OPTIONS] <REGION-SET>

-v      verbose mode                                          [false]
-help   help                                                  [true]
-h      help                                                  [true]
-g      genome region file                                    [genome.reg+]
-r      reference region file                                 []
-w      window size (must be a multiple of window step)       [500]
-d      window step                                           [25]
-n      use region label as read count                        [false]
```

```
-op     preprocess operator (1=start, c=center, p=all points)     [c]
-min    minimum required reads for output window                  [0]
```

In this example case, read density per window is generated by the following command:

```
$ genomic_regions –scanc –g human.reg –w 100 –d 20 –min 10 reads.sorted.reg > reads.win
```

Here "human.reg" is a reg file that defines the start and end coordinates for each chromosome of the human genome, sorted with the command given earlier for sorting reg files. The first two lines of the file could look like this:

```
chromosome_1<TAB>1 + 1 249250621
chromosome_1<TAB>1 - 1 249250621
```

The option "-min 10" states that the output will only contain those windows with 10 or more reads. The first two lines of the "reads.win" output file could look like this:

```
45<TAB>1 + 569821 569920
60<TAB>1 + 569841 569940
```

The first field states how many reads were encountered in this window (read count R) and the coordinates give the location of the window on the reference genome. To convert these to RPKM values, one simply applies this equation to the first field:

$$RPKM = ((R/L)*1000) / M \qquad \text{(equation 1)}$$

Here R is the read count in the window, L is the window length, and M is the number of mapped reads in the experiment in millions. Let us assume this equation is applied to the first field on all lines of the file "reads.win", and let us call this modified file "reads.rpkm".

Next, we will convert the RPKM values to the WIG format with the "-wig" option of the genomic_regions tool:

```
$ genomic_regions –wig –h
```

```
USAGE:
 genomic_regions -wig [OPTIONS] <REGION-SET>

-v      verbose mode                                     [false]
-help    help                                            [true]
-h      help                                             [true]
-t      title                                            []
-c      color                                            [200,0,0]
-s      span                                             [1]
-p      browser position                                 []
-o      track type options                               []
-chr    convert chromosome names from ENSEMBL to UCSC    [false]
```

Since we are going to visualize the results in the UCSC genome browser we need to use the option "-chr", and we use the option "-s 20" to make each window appear 20 nt wide on the browser. The appropriate command in this case is the following:

```
$ genomic_regions –wig –t "RNA-seq experiment" –chr –s 20 reads.rpkm > reads.wig
```

Assuming we have 5 million reads in our experiment, the RPKM values are 2*R in each window (according to equation 1). Therefore the first rows in the WIG file will look like this:

```
track type=wiggle_0 name='RNA-seq experiment' color=200,0,0
variableStep chrom=chr1 span=20
569821 90
569841 120
```

Now this WIG file can be uploaded as a user track in the UCSC genome browser. The result is illustrated in Figure 3 after setting the user track visibility to "full" in the genome browser.

2. ChIP-seq read profiles

Next, we will demonstrate how you can create read profiles around gene TSSs. This is particularly useful for validating ChIP-seq experiments. For example, histone modification H3K9ac peaks near the TSSs activated genes. More specifically, H3K9ac has a peak a few base pairs upstream of the TSSs and a bigger peak a few base pairs downstream of the TSSs (see Figure 4).

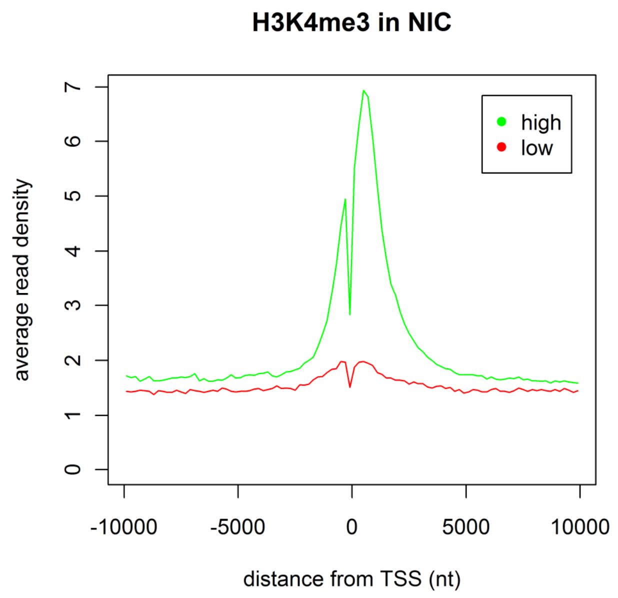


Figure 4: read profile around gene TSSs.

We can use genomic region tools to produce such profiles as follows: suppose the H3K9ac sequenced reads have been mapped to the reference genome and are stored as a BED file "H3K9ac.bed", and that we also have a file containing all the gene transcript genomic regions (from transcript start to transcript end, both exons and introns) also in a BED file named “transcript.bed”. All input files need to be sorted by chromosome, strand and start position.

First, we need to find the TSSs of the transcripts and “flank” them by say 10kb upstream and downstream. This can be accomplished by applying the “-pos” operation of genomic_regions tool:

```
$ genomic_regions -pos -h
```

```
USAGE:
 genomic_regions -pos [OPTIONS] <REGION-SET>

-v      verbose mode                                                            [false]
-h      help                                                                    [true]
-op     position operation (1=start, 2=stop, 5p=5'-end, 3p=3'-end, c=center)    [1]
-c      position shift                                                          [0]
```

Simply create the following pipeline:

```
$ cat transcript.bed | genomic_regions -pos -op 5p | genomic_regions -shift -1 -10000 -2
10000 > TSS.reg
```

This pipeline first selects the 5’-end of every transcript and the used the “-shift” operation to shift the first coordinate by -10kb and the second one by +10kb. The result is a reg file containing all TSSs flanked by 10kb.

Then, to create the profiles, we use the “-offset” operation of the genomic_overlaps tool:

```
$ genomic_overlaps -offset –h
```

```
USAGE:
 genomic_overlaps -offset [OPTIONS] QUERY-REG-FILE <INDEX-REG-FILE>

-v      verbose mode                                                            [false]
-h      help                                                                    [true]
-val    use values contained in the labels of index intervals                   [false]
-op     position operation (1=start, 2=stop, 5p=5'-end, 3p=3'-end, c=center)    [1]
-a      print distances as a fraction of total size                             [false]
-c      print center of interval only                                           [false]
```

The genomic_overlaps tool computes overlaps between two genomic region files, in this example, between the H3K9ac reads and the TSSs. The offset operation computes the distances of these overlapping reads from the reference TSSs regions. Remember that we need to compute the distances from the 5’-end of the TSS to correctly interpret upstream (i.e. negative) versus downstream (i.e. positive) distances. Option “-c” ensures that we only obtain the center of the offsets (by default, the program output the offsets of both the start and the end coordinates). Also, we enable the option “-a” which normalizes the distance (i.e. offset) by dividing by the TSS size, so the distance is a number between 0 and 1, and the offsets are summarized as a histogram using the “-hist” operation of the vectors tool (here we are using 100 bins, option “-b”):

```
$ cat H3K9ac.bed | genomic_overlaps -offset -v -op 5p -a –c TSS.bed | vectors -hist -b 100 -n
6 > H3K9ac.histogram.txt
```

Then, we can simply plot the histogram using our favorite program (e.g. gnuplot, R, etc). In order to create two histograms like in Figure 4, we need to create one TSS file for the genes of high expression and one for the genes of low expression and run the same pipeline separately.

A slightly more sophisticated pipeline can take into account the fragmentation length (if known). Suppose the fragmentation length is 200nt, then the following pipeline can be used:

```
$ cat H3K9ac.bed | genomic_regions –pos -op 5p -c 200 | genomic_overlaps -offset -v -op 5p
-a –c TSS.bed | vectors -hist -b 100 -n 6 > H3K9ac.histogram.txt
```

Basically, the "-pos" operation is used to extend each sequenced read by 200nt downstream from its 5'-end, while the rest of the pipeline remains unchanged.

References

[1] R. Gentleman et al., "Bioconductor: open software development for computational biology and bioinformatics," Genome Biology, vol. 5, no. 10, p. R80, 2004.

[2] J. Goecks, A. Nekrutenko, J. Taylor, and T. Galaxy Team, "Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences," Genome Biology, vol. 11, no. 8, p. R86, 2010.

[3] W. J. Kent et al., "The Human Genome Browser at UCSC," Genome Research, vol. 12, no. 6, pp. 996 -1006, Jun. 2002.

[4] C. Y. McLean et al., "GREAT improves functional interpretation of cis-regulatory regions," Nat Biotech, vol. 28, no. 5, pp. 495-501, May. 2010.

[5] P. Rice, I. Longden, and A. Bleasby, "EMBOSS: The European Molecular Biology Open Software Suite," Trends in Genetics, vol. 16, no. 6, pp. 276-277, Jun. 2000.

Appendix

USAGE:

genomic_regions OPERATION [OPTIONS] <REGION-SET>

DESCRIPTION:

This is a collection of line/file based operations for manipulating genomic regions.

OPTIONS:

Line-based operations for reg, bed and fasta (when applicable) input formats:

-bed	convert input reg to UCSC bed format
-bounds	check interval start/stop positions against genome chromosome regions
-center	print center interval
-connect	connect intervals from minimum start to maximum stop
-diff	compute the difference between successive intervals
-dist	compute distances (for pairs of intervals only)
-divide	divide intervals in the middle
-fix	remove rogue intervals
-int	compute the interval intersection
-n	compute total interval length (including possible overlaps)
-pos	modify interval start/stop positions
-print	format intervals
-rnd	randomize intervals across entire genome
-shift	shift interval start/stop positions
-shuffle	shuffle intervals within given reference region
-sort	sort intervals
-strand	modify interval strand information
-union	compute the interval union
-wig	convert input reg to UCSC wiggle format
-win	create new invervals by sliding windows
-x	extract corresponding sequences from DNA files

File-based operations for reg, bed and fasta (when applicable) input formats:

-cluster	cluster regions based on overlaps
-gdist	compute distances between successive regions
-inv	invert regions given the genome chromosomal boundaries
-link	link overlapping regions thus producing a non-overlapping set
-merge	merge consecutive non-overlapping regions
-scanc	scan input intervals in sliding windows and print read count
-test	test whether genomic regions are sorted and non-overlapping

USAGE:

genomic_regions -bed [OPTIONS] <REGION-SET>

DESCRIPTION:
Converts input REG file to UCSC BED format.
OPTIONS:

-v	verbose mode	[false]
-help	help	[true]
-h	help	[true]
-t	title	[]
-c	color	[]
-p	browser position	[]
-chr	convert chromosome names from ENSEMBL to UCSC	[false]

USAGE:
genomic_regions -bounds [OPTIONS] <REGION-SET>
DESCRIPTION:
Checks interval start/stop positions against genome chromosome regions.
OPTIONS:

-v	verbose mode	[false]
-help	help	[true]
-h	help	[true]
-g	genome region file	[]

USAGE:
genomic_regions -center [OPTIONS] <REGION-SET>
DESCRIPTION:
Prints center interval.
OPTIONS:

-v	verbose mode	[false]
-help	help	[true]
-h	help	[true]

USAGE:
genomic_regions -connect [OPTIONS] <REGION-SET>
DESCRIPTION:
Connects intervals from minimum start to maximum stop.
OPTIONS:

-v	verbose mode	[false]
-help	help	[true]
-h	help	[true]

USAGE:
genomic_regions -diff [OPTIONS] <REGION-SET>
DESCRIPTION:
Computes the difference between successive intervals.
OPTIONS:

-v	verbose mode	[false]
-help	help	[true]
-h	help	[true]

USAGE:
genomic_regions -dist [OPTIONS] <REGION-SET>
DESCRIPTION:
Computes distances (for pairs of intervals only).
OPTIONS:

-v	verbose mode	[false]
-help	help	[true]
-h	help	[true]
-3p	distance of the 3' ends	[false]
-5p	distance of the 5' ends	[false]

USAGE:
genomic_regions -divide [OPTIONS] <REGION-SET>
DESCRIPTION:
Divides intervals in the middle.
OPTIONS:

-v	verbose mode	[false]
-help	help	[true]
-h	help	[true]

USAGE:
genomic_regions -fix [OPTIONS] <REGION-SET>
DESCRIPTION:
Removes rogue intervals.
OPTIONS:

-v	verbose mode	[false]
-help	help	[true]
-h	help	[true]

USAGE:

genomic_regions -n [OPTIONS] <REGION-SET>

DESCRIPTION:

Calculates total interval length (including possible overlaps).

OPTIONS:

-v	verbose mode	[false]
-help	help	[true]
-h	help	[true]

USAGE:

genomic_regions -pos [OPTIONS] <REGION-SET>

DESCRIPTION:

Modify interval start/stop positions.

OPTIONS:

-v	verbose mode	[false]
-help	help	[true]
-h	help	[true]
-op	position operation (1=start, 2=stop, 5p=5'-end, 3p=3'-end, c=center)	[1]
-c	position shift	[0]

EXAMPLE: Shift all regions 10 positions from 3’-end and save result to output.reg

```
genomic_regions –pos –op c –3p 10 sample.reg > output.reg
```

USAGE:

genomic_regions -print [OPTIONS] <REGION-SET>

DESCRIPTION:

Formats intervals by compacting in start/end format or by printing every point.

OPTIONS:

-v	verbose mode	[false]
-help	help	[true]
-h	help	[true]
-1	print one region per line	[false]
-c	print in compact starts/ends format	[false]
-p	print regions as points	[false]

USAGE:

genomic_regions -rnd [OPTIONS] <REGION-SET>

DESCRIPTION:

Randomizes intervals across entire genome. Useful for nonparametric tests.
OPTIONS:

```
 -v           verbose mode                                  [false]
 -help        help                                          [true]
 -h           help                                          [true]
 -g           genome region file
        [genome.reg]
```

USAGE:
 genomic_regions -shift [OPTIONS] <REGION-SET>
DESCRIPTION:
Shifts interval start/stop positions.
OPTIONS:

```
 -v           verbose mode                                  [false]
 -help        help                                          [true]
 -h           help                                          [true]
 -1           start shift offset                            [1]
 -2           stop shift offset                             [1]
 -3p          shift the 3' end                              [false]
 -5p          shift the 5' end                              [false]
```

USAGE:
 genomic_regions -shuffle [OPTIONS] <REGION-SET>
DESCRIPTION:
Shuffle intervals within given reference region.
OPTIONS:

```
 -v           verbose mode                                  [false]
 -help        help                                          [true]
 -h           help                                          [true]
 -r           reference region file                         []
```

USAGE:
 genomic_regions -sort [OPTIONS] <REGION-SET>
DESCRIPTION:
Sorts intervals by chromosome, start and then stop positions.
OPTIONS:

```
 -v           verbose mode                                  [false]
 -help        help                                          [true]
 -h           help                                          [true]
```

USAGE:

genomic_regions -strand [OPTIONS] <REGION-SET>

DESCRIPTION:

Modifies interval strand information.

OPTIONS:

-v	verbose mode	[false]
-help	help	[true]
-h	help	[true]
-op	strand operation (b=both, +=positive, -=negative, r=reverse)	[b]

USAGE:

genomic_regions -union [OPTIONS] <REGION-SET>

DESCRIPTION:

Computes the union of intervals.

OPTIONS:

-v	verbose mode	[false]
-help	help	[true]
-h	help	[true]

USAGE:

genomic_regions -wig [OPTIONS] <REGION-SET>

DESCRIPTION:

Converts input REG to UCSC WIGGLE format.

OPTIONS:

-v	verbose mode	[false]
-help	help	[true]
-h	help	[true]
-t	title	[]
-c	color	[200,0,0]
-s	span	[1]
-p	browser position	[]
-o	track type options	[]
-chr	convert chromosome names from ENSEMBL to UCSC	[false]

USAGE:

genomic_regions -win [OPTIONS] <REGION-SET>

DESCRIPTION:

Create new intervals by sliding windows.
OPTIONS:

-v	verbose mode	[false]
-help	help	[true]
-h	help	[true]
-d	window distance	[1]
-s	window size	[1]

EXAMPLE: Create 100 nt size windows with 50 nt sliding step.

```
 genomic_regions –win –d 50 –s 100 sample.reg > output.reg
```

USAGE:
 genomic_regions -x [OPTIONS] <REGION-SET>
DESCRIPTION:
Extract corresponding sequences from DNA files.
OPTIONS:

-v	verbose mode	[false]
-help	help	[true]
-h	help	[true]
-D	chromosome DNA file and map directory	[.]
-q	sequence map file [chromosome.map]	
-r	replace 'N' characters with 'a'	[false]
-i	ignore boundary errors	[false]

USAGE:
 genomic_regions -cluster [OPTIONS] <REGION-SET>
DESCRIPTION:
Cluster regions based on overlaps.
OPTIONS:

-v	verbose mode	[false]
-help	help	[true]
-h	help	[true]
-m	merge regions in each cluster	[false]

USAGE:
 genomic_regions -gdist [OPTIONS] <REGION-SET>
DESCRIPTION:
Computes distances between successive regions. Distances are reported in the first column as Read1|Read2|dist.
OPTIONS:

```
 -v              verbose mode                                          [false]
 -help           help                                                  [true]
 -h              help                                                  [true]
```

USAGE:
 genomic_regions -inv [OPTIONS] <REGION-SET>
DESCRIPTION:
Gives the complementary regions with respect to the chromosomal boundaries.
OPTIONS:

```
 -v              verbose mode                                          [false]
 -help           help                                                  [true]
 -h              help                                                  [true]
 -g              genome region file
         [genome.reg]
```

USAGE:
 genomic_regions -link [OPTIONS] <REGION-SET>
DESCRIPTION:
Links overlapping regions thus producing a non-overlapping set.
OPTIONS:

```
 -v              verbose mode                                          [false]
 -help           help                                                  [true]
 -h              help                                                  [true]
```

USAGE:
 genomic_regions -merge [OPTIONS] <REGION-SET>
DESCRIPTION:
Merges consecutive non-overlapping regions.
OPTIONS:

```
 -v              verbose mode                                          [false]
 -help           help                                                  [true]
 -h              help                                                  [true]
 -w              window length (nt)                                    [100]
```

EXAMPLE: Merge consecutive regions that are at most 50 nt apart

```
 genomic_regions –merge –w 50 sample.reg > output.reg
```

USAGE:
 genomic_regions -scanc [OPTIONS] <REGION-SET>

DESCRIPTION:
Scan input intervals in sliding windows and print read count.
OPTIONS:

-v	verbose mode	[false]
-help	help	[true]
-h	help	[true]
-g	genome region file	[genome.reg]
-r	reference region file	[]
-w	window size (must be a multiple of window step)	[500]
-d	window step	[25]
-n	use region label as read count	[false]
-i	ignore negative strand	[false]
-op	preprocess operator (1=start, c=center, p=all points)	[c]
-min	minimum required reads for output window	[0]

EXAMPLE: Scan the input REG file with 100 nt non-overlapping windows and report the read count.

```
  genomic_regions –scanc –w 100 –d 100 –g genome.reg sample.reg > output.reg
```

USAGE:
genomic_regions -test [OPTIONS] <REGION-SET>
DESCRIPTION:
Tests whether genomic regions are sorted and non-overlapping.
OPTIONS:

-v	verbose mode	[false]
-help	help	[true]
-h	help	[true]

USAGE:
genomic_overlaps OPERATION [OPTIONS] QUERY-REG-FILE <INDEX-REG-FILE>
DESCRIPTION:
A tool for computing overlaps in different formats. Query and index regions must be single intervals and sorted.
OPTIONS:

-count	count matches per query region
-density	compute density of matches per query region
-offset	find overlaps and compute offsets
-overlap	find overlaps with index regions

USAGE:

genomic_overlaps -count [OPTIONS] QUERY-REG-FILE <INDEX-REG-FILE>

DESCRIPTION:

Counts total number of nucleotide matches per query region.

OPTIONS:

-v	verbose mode	[false]
-help	help	[true]
-h	help	[true]
-val	use values contained in the labels of index intervals	[false]
-min	minimum count	[0]

EXAMPLE: Find out how many nucleotides match regions in index.reg. Report only intervals that match at least 50 points.

```
genomic_regions –count –min 50 sample.reg index.reg > output.reg
```

USAGE:

genomic_overlaps -density [OPTIONS] QUERY-REG-FILE <INDEX-REG-FILE>

DESCRIPTION:

Compute density of matches per query region.

OPTIONS:

-v	verbose mode	[false]
-help	help	[true]
-h	help	[true]
-val	use values contained in the labels of index intervals	[false]
-min	minimum density [0.000000]	

EXAMPLE: Calculate density of matches of regions in sample.reg with respect to index.reg. Report only intervals with density > 1

```
genomic_regions –density –min 1 sample.reg index.reg > output.reg
```

USAGE:

genomic_overlaps -offset [OPTIONS] QUERY-REG-FILE <INDEX-REG-FILE>

DESCRIPTION:

Find overlaps and compute offsets with reference to start, stop, 3’ or 5’ position.

OPTIONS:

-v	verbose mode	[false]
-help	help	[true]
-h	help	[true]
-val	use values contained in the labels of index intervals	[false]
-op	reference point (1=start, 2=stop, 5p=5'-end, 3p=3'-end)	[1]
-a	print distances as a fraction of total size	[false]

-c	print center of interval only	[false]

USAGE:

genomic_overlaps -overlap [OPTIONS] QUERY-REG-FILE <INDEX-REG-FILE>

DESCRIPTION:

Find overlaps with index regions.

OPTIONS:

-v	verbose mode	[false]
-help	help	[true]
-h	help	[true]
-val	use values contained in the labels of index intervals	[false]
-x	compute exact overlap boundaries	[false]