

User's guide to ChIP-Seq applications: command-line usage and option summary

1. Basics about the ChIP-Seq Tools

The ChIP-Seq software provides a set of tools performing common genome-wide ChIP-Seq analysis tasks, including positional correlation analysis, peak detection, and genome partitioning into signal-rich and signal-poor regions. These tools exist as stand-alone C programs and perform the following tasks:

- I. Positional correlation analysis (`chipcor`, `chipextract`),
- II. Tag centering (`chipcenter`),
- III. Signal peaks detection (`chippeak`),
- IV. Partitioning of the genome into signal-rich and poor regions (`chippart`),
- V. Feature extraction tool (`chipscore`)
- VI. Filtering tool (`counts_filter`)

Because the ChIP-Seq tools are primarily optimized for speed, they use their own compact format for ChIP-Seq data representation called SGA (Simplified Genome Annotation). SGA is a line-oriented, tab-delimited plain text format with the following five obligatory fields per line:

- I. Sequence ID (char string)
- II. Feature (char string)
- III. Sequence Position (integer)
- IV. Strand (+/- or 0)
- V. Tag Counts (integer)

An additional field may be added containing application-specific information used by other programs. In the case of ChIP-Seq data, an SGA file represents the genome-wide tag count distributions from one or several experiments. However, the format can also represent a large variety of other types of genomic data, including derived features such as peaks extracted from ChIP-Seq data, or genome annotations such as promoters. Any type of genomic feature that can be projected to a single base on a chromosome can be represented in SGA format.

The *sequence ID* field typically identifies a chromosome. The software does not impose any naming convention. However, if you merge data from different experiments, the same names must be used for the same chromosomes. The public data files access by the ChIP-Seq server menu use complete (i.e. including the version) NCBI/RefSeq accession numbers as sequence/chromosome identifiers. Chromosome names as used by the UCSC genome browser constitute another *de facto* standard and are readily converted into NCBI/RefSeq accession numbers. However, keep in mind that chromosome names are ambiguous unless the corresponding assembly is indicated. Do not mix chromosome names from different assemblies otherwise you will get wrong results.

The *feature* field contains a short code which identifies an experiment. It often corresponds to the name of the molecular target of a ChIP-Seq experiment. Its function is to distinguish data lines relating to different experiments that were merged into a single file. The *position* field contains the position within the sequence. The *strand* field indicates the strand to which the feature has been mapped. The SGA format distinguishes between “oriented” features that occur either on the plus or on the minus strand of the chromosome sequence, and “unoriented” features which cannot be assigned to one or the other strand. Peaks from a ChIP-Seq experiment, for instance, constitute an example of an unoriented feature. Unoriented features are identified by a 0 (zero) in field 4. The *counts* field contains the number of reads that have been mapped to a specific base position on the chromosome. It can also be used for other purposes as well, such as representing conservation scores, SNP frequencies or any type of genome annotation that can be mapped to a single base on a chromosome.

An example of an SGA-formatted file is shown here below:

NC_000001.9	H3K4me3	4794	+	1
NC_000001.9	H3K4me3	6090	+	1
NC_000001.9	H3K4me3	6099	+	3
NC_000001.9	H3K4me3	6655	+	1
NC_000001.9	H3K4me3	18453	-	1
NC_000001.9	H3K4me3	19285	+	1
NC_000001.9	H3K4me3	44529	+	4
NC_000001.9	H3K4me3	46333	+	1
NC_000001.9	H3K4me3	46349	-	1

ChIP-Seq programs require SGA files to be sorted by sequence name, position, and strand. In a UNIX environment, the command to properly sort SGA files is the following:

```
sort -s -k1,1 -k3,3n -k4,4 <SGA file>
```

Duplicated lines are allowed, even though we provide a program called *compactsga* to merge repeated lines into a single line adjusting the count field accordingly.

2. ChIP-Seq programs: command-line usage and examples of use

1. Chipcor

```
chipcor [OPTIONS] -A <feature A> -B <feature B> -b <from> -e <to> -w <window> [<] SGAfile
```

The *chipcor* program reads a data file (or from *<stdin>*) in SGA format (*<SGAfile>*) and generates a count correlation histogram, which is a text file that indicates the frequency of the target feature (*<feature B>*) as a function of the relative distance to the reference feature (*<feature A>*). The output of *chipcor* can be easily imported into any statistical or graphical tool, such as R or gnuplot, to generate the correlation histogram.

Mandatory parameters are:

Parameter	Description
-A <feature>	Feature field name selecting reference tags in the SGA file. The strand can also be included as a feature specification (<feature> = <name> [+ -]).
-B <feature>	Feature field name selecting target tags in the SGA file. The strand can also be included as a feature specification (<feature> = <name> [+ -]).
-b <from>	Beginning position of the correlation analysis range (relative distance in bp) considered in the output histogram.
-e <to>	End position of the correlation analysis range (relative distance in bp).
-w <window>	The window defines the histogram step size or bin.

Options are:

Option	Description
-h	Show the program usage.
-d	Produce debugging output and check whether the input SGA is properly sorted.
-o	Oriented strand processing. It means reverting the chromosome axis when the reference feature is on the – strand.
-c	Count cut-off value for all input tags (def=10).
-n	Histogram normalization. By default is 0, meaning that histograms entries display raw tag counts. If –n 1, histograms entries display the count density (#counts/bp) of the target feature. If –n 2, histograms entries represent target feature abundance as a fold change relative to a genome average.

The exact position of the window along the range is defined such that the window is always centered at position 0. As a result, the range might be slightly reduced.

Here is an example of using the ChIP-Seq correlation program *chipcor* to estimate the average fragment size of CTCF tags:

```
chipcor -A "CTCF +" -B "CTCF -" -b -1000 -e 1000 -w 1 -c 1 -n 1 CTCF.sga > CTCF.out
```

where:

<i>CTCF.sga</i>	is the input file containing the list of mapped CTCF tags
-A "CTCF +"	is the reference feature (CTCF plus strand)
-B "CTCF -"	is the target feature (CTCF minus strand)
-b	is beginning of the range considered in the output histogram
-e	is the end of the range
-w	is the window width
-c	is the count cut-off value
-n	is the normalization mode (1 means “count density”)
<i>CTCF.out</i>	is the output file containing the histogram values in text format

Documentation for most ChIP-Seq programs is provided via UNIX- style man page, type *e.g.*:

man chipcor

Short usage instructions can also be obtained by simply typing the name of the program without any options or arguments.

In the above example, the reference and target features were contained in the same input file. This is not always the case. In the following example, we will analyze the distribution of CTCF peaks in mouse embryonic stem cells relative to transcription start sites from the Eukaryotic Promoter Database EPD. The input data for this analysis are provided in two separate files: *ES_CTCF_peaks.sga* for CTCF peaks in mouse ES cells, and *Mm_EPDnew_001_mm9.sga* for TSSs in mouse.

These files need first to be merged before they can be processed by *chipcor*.

```
sort -s -m -k1,1 -k3,3n -k4,4 Mm_EPDnew_001_mm9.sga ES_CTCF_peaks.sga |  
> merged.sga  
chipcor -A "TSS" -B "CTCF_P" -b -2500 -e 2500 -w 50 -c 1 -n 1 -o merged.sga |  
> CTCF_peaks.out
```

When the ‘oriented’ option (-o) is set, the strand of the reference tag is taken into account in order to calculate the relative distance between the reference tag and all the target tags that fall within the correlation range. In other words, the chromosome axis is reverted each time the reference feature, that is the TSS in this example, is on the minus strand.

2. Chipcenter

```
chipcenter [OPTIONS] [-f <feature>] -s <shift> [<] SGAfile
```

The *chipcenter* program reads a data file (or from <stdin>) in SGA format (<SGAfile>), and shifts (by <shift> bases) tag positions corresponding to a specific feature (<feature>) to the estimated center-positions of DNA fragments. If no feature specification is set, the program accepts all oriented lines of the input SGA. Strandless features are ignored. The default output of *chipcenter* is a list of shifted oriented positions in SGA format. If necessary, one can set the output strand to zero by means of the ‘-z’ option. The ‘-r’ option is used to replace the feature field with a new string.

The main parameters are:

Parameter	Description
-f <feature>	It is used to select all or a sub-set of input tags. The feature name is specified in the second field of the SGA-formatted input. This parameter is not mandatory, in the sense that if no feature is given, then all tags are selected.
-s <shift>	It defines the relative shift (in bp) of tag positions to estimated center-positions of DNA fragments.

Options are:

Option	Description
-h	Show the program usage.
-d	Produce debugging output and check whether the input SGA is properly sorted.
-z	Set output strand to zero.
-c	Count cut-off value for all input tags (def=10).
-r	New feature name (for feature replacement)

Chipcenter is often used as a data preprocessing step in most applications.

Here is an example of using *chipcenter* in order to center mapped CTCF tags from human CD4+ cells:

```
chipcenter -f "CTCF" -s 40 CTCF.sga > CTCF_centered.out
```

where:

CTCF.sga is the input file containing the list of mapped CTCF tags
-f "CTCF" is the feature name
-s 40 is the tag position shift

For ChIP-seq data, the shift value corresponds to half the average fragment size of the tag reads as estimated by the 5'end-3'end correlation diagram produced by *chipcor*.

3. Chipextract

```
chipextract [OPTIONS] -A <feature A> -B <feature B> -b <from> -e <to> -w <bin> [<] SGAfile
```

The *chipextract* program reads a data file (or from *<stdin>*) in SGA format (*<SGAfile>*) containing two features, a reference feature (*<feature A>*) and a target feature (*<feature B>*), and, for each reference feature, it extracts target feature tags that fall into a distance range (*[<from>, <to>]*) relative to the reference feature. The output of *chipextract* is a table in text format consisting of all reference features (rows) with relative target tag counts in bins of a given *<bin>* size (columns). For visualization purposes, the *chipextract* table can be easily converted into a heat map, using R or similar tools.

Mandatory parameters are:

Parameter	Description
-A <feature>	Feature field name selecting reference tags in the SGA file. The strand can also be included as a feature specification, the format being the following: <i><feature> = <name> [+ - 0(strandless) a(any) o(oriented)]</i> . If the strand is not specified, it is set to a(any) by default.
-B <feature>	Feature field name selecting target tags in the SGA file. The strand can also be included as a feature specification in the same way as for the

	reference feature.
-b <from>	It defines the beginning position of the correlation analysis range (relative distance in bp).
-e <to>	End position of the correlation analysis range (relative distance in bp).
-w <bin>	Columns width (in bp) of the heat map table.

Options are:

Option	Description
-h	Show the program usage.
-d	Produce debugging output and check whether the input SGA is properly sorted.
-c	Count cut-off value for all input tags (def=1).

The exact position of the sliding window along the range is defined such that the window is always centered at position 0. As a result, the range might be slightly reduced.

Here is an example of using *chipextract* to analyze H3K4 histone modifications in mouse embryonic stem cells relative to transcription start sites from the Eukaryotic Promoter Database EPD. We would like to extract H3K4 tags around transcription initiation sites (TSS). The input data for this analysis are provided in two separate files: *ES_K4.sga* for H3K4me3 histone marks in mouse ES cells, and *Mm_EPDnew_001_mm9.sga* for TSSs in mouse.

We first need to center the histone tags to estimated middle-positions of DNA fragments. We shift the tags of 75 bp according to the 5'end-3'end correlation analysis:

```
chipcenter -f "H3K4me3" -s 75 ES.K4.sga > ES.K4_centered.sga
```

As for *chipcor*, the input files need first to be merged before they can be processed by *chipextract*:

```
sort -s -m -k1,1 -k3,3n -k4,4 Mm_EPDnew_001_mm9.sga ES.K4_centered.sga |> ES.TSS.K4.merged.sga

chipextract -A "TSS o" -B "H3K4me3 a" -b-1000 -e1000 -w20 ES.TSS.K4_merged.sga > ES.TSS.K4_heatmap.out
```

where:

ES.TSS.K4_merged.sga is the input file containing the list of centered histone marks tags together with the TSSs

-A "TSS o" is the reference feature (oriented TSS)

-B "H3K4me3 a" is the target feature (H3K4me3 any, i.e. on both strands)

-b is beginning of the range considered in the output table

-e is the end of the range

-w is the bin size (in bp)

ES.TSS.K4_heatmap.out is the heat map table in text format consisting of target tag counts in bins of 20bp relative to each TSS

4. Chippeak

```
chippeak [OPTIONS] [-f <feature>] -t <threshold> -w <window> -v <vicinity> [<] SGAfile
```

The *chippeak* program reads a data file (or from *<stdin>*) in SGA format (*<SGAfile>*), and detects signal peaks for tag positions corresponding to a specific feature (*<feature>*) when given. If no feature is specified, all input tags or positions are equally processed. The output is a list of peak center positions in SGA format. If tags or positions on both strands are selected, the peak list is strandless, whereas if features on only one strand are selected, the resulting peak list is oriented. The peak counts are reported in the 5th field. The main parameters are:

Parameter	Description
-f <feature>	It is used to select all or a sub-set of input tags. The feature name is specified in the second field of the SGA-formatted input. This parameter is not mandatory, in the sense that if no feature is given, then all tags are selected.
-t <threshold>	Peak threshold. Cumulative tag counts within a range <i><window></i> should be bigger or equal to threshold <i><threshold></i> . The default value is 50.
-w <window>	It defines the integration range (in bp) of tag counts around each tag position across the whole tag distribution.
-v <vicinity>	It defines the minimal distance (in bp) amongst a group of local peak maxima (high count values).

Chippeak implements a very simple method which works as follows. The number of tags is counted in a sliding window of fixed width. Speed is gained by considering only those windows which have at least one tag at the center position. At the end, all windows which have tag numbers greater or equal to threshold value *<t>*, and in addition are locally maximal within a so-called “vicinity range” *<v>*, are reported as peaks. Note that only the peak mid-point positions are included in the output. If “peak refinement” is selected, a post-processing step is turned on so as to improve peak location. For initially selected peaks, the position is recomputed as the center of gravity of the counts in the region defined by the window *<w>* parameter.

Chippeak works at its best with centered tag distributions.

Options are:

Option	Description
-h	Show the program usage.
-d	Produce debugging output and check whether the input SGA is properly sorted.
-r	Refine peak positions.
-c	Count cut-off value for all input tags (def=10).

Here is an example of using *chippeak* on the CTCF centered tags:

```
chippeak -f "CTCF" -t 15 -w 200 -v 200 CTCF_centered.sga > CTCF_peaks.out
```

where:

<i>CTCF_centered.sga</i>	is the input file containing the list of centered CTCF tags
<i>-f "CTCF"</i>	is the feature name
<i>-t 15</i>	is the peak threshold (in tag counts)
<i>-w 200</i>	is the window width
<i>-v 200</i>	is the vicinity range

With these parameters, the program detects 15'600 peaks.

5. Chippart

```
chippart [OPTIONS] [-f <feature>] -p <transition penalty> -s <density threshold> [<] SGAfile
```

The *chippart* program reads a data file (or from *<stdin>*) in SGA format (*<SGAfile>*), and finds signal-enriched regions for tag positions corresponding to a given feature *<feature>*. If no feature is specified, all input tags are equally processed. It outputs a list of signal-enriched regions in SGA format in two lines (beginning of the region, end of the region). For *chippart* the strand field of the output has a different meaning: the '+' character indicates the beginning of the region, whereas the '-' character indicates the end of the region.

The main parameters are:

Parameter	Description
-f <feature>	It is used to select all or a sub-set of input tags. The feature name is specified in the second field of the SGA-formatted input. This parameter is not mandatory, in the sense that if no feature is given, then all tags are selected.
-p <transition penalty>	It assigns a negative score to a transition between signal-enriched and signal-poor regions. The parameter needs to be negative. It controls the fragment length, i.e. high penalty -> long fragments.
-s <density threshold>	It is a count density threshold. A region must have a count density higher than the density threshold to be considered as a signal-enriched DNA stretch.

The program also generates (to *<stderr>*) a statistical report with the following information:

- I. Total number of processed sequences, total DNA length, and total number of fragments;
- II. Total fragment length, average fragment length, and percentage of DNA length;
- III. Percentage of total tag counts, average number of counts, and count density.

The two parameters of the programs are used to optimize a partitioning scoring function by means of a fast dynamic programming algorithm. The scoring function is defined as a sum of scores of:

1. Transition penalties (**penalty**)
2. Signal-enriched scores : $length * (\text{local count-density} - \text{threshold})$
3. Signal-poor regions : $length * (\text{threshold} - \text{local count-density})$

Options are:

Option	Description
-h	Show the program usage.
-d	Produce debugging output and check whether the input SGA is properly sorted.
-c	Count cut-off value for all input tags (def=10).

The *chippart* program is most suited to find signal regions such as histone modifications that spread over large DNA regions.

Here is an example of using *chippart* to analyze H3K4 histone modifications in mouse embryonic stem cells. As we did previously, we first need to center the histone tags to estimated middle-positions of DNA fragments. We shift the tags of 75 bp according to the 5'end-3'end correlation analysis:

```
chipcenter -f "H3K4me3" -s 75 ES.K4.sga > ES.K4_centered.sga
```

```
chippart -f "H3K4me3" -p-10 -s0.014 ES.K4_centered.sga > ES.K4_partit.out
```

where:

<i>ES.K4_centered.sga</i>	is the input file containing the list of centered histone marks tags
<i>-f "H3K4me3"</i>	is the feature name
<i>-p-10</i>	is the transition penalty
<i>-s 0.014</i>	is the count density threshold

The output of *chippart* is a SGA-like format in which each region of interest is split into two lines, the beginning and the end, respectively. This SGA-like format can be easily converted into BED format, using the conversion tool *partit2bed.pl*.

It is important to note that all ChIP-Seq programs send their output to standard output (*<stdout>*) and can therefore be used in a pipeline.

Here is an example of such pipeline using *chipcenter* and *chippart*:

```
chipcenter -f "H3K4me3" -s 75 ES.K4.sga | chippart -f "H3K4me3" -p-10 -s0.014 > ES.K4_partit.out
```

Another example using *chipcenter* and *chippeak* for peak calling is the following:

```
chipcenter -f "CTCF" -s 40 CTCF.sga | chippeak -f "CTCF" -t 15 -w 200 -v 200 > CTCF_peaks.out
```

6. Chipscore

```
chipscore [OPTIONS] -A <feature A> -B <feature B> -b <from> -e <to> [-t <thres>] [<] SGAfile
```

The *chipscore* program reads a data file (or from *<stdin>*) in SGA format (*<SGAfile>*) and extracts all reference sites (*<feature A>*) that are enriched or depleted in target feature (*<feature B>*) sites according to a given count threshold or score (*<thres>*). Output SGA lines are those reference features that are enriched or depleted in target feature. The counts corresponding to the target feature are reported in the 6th field. The main parameters are:

Parameter	Description
-A <feature>	Feature field name selecting reference tags in the SGA file. The strand can also be included as a feature specification (<i><feature> = <name> [+ -]</i>).
-B <feature>	Feature field name selecting target tags in the SGA file. The strand can also be included as a feature specification (<i><feature> = <name> [+ -]</i>).
-b <from>	Beginning position of the correlation analysis range (relative distance in bp) considered in the output histogram.
-e <to>	End position of the correlation analysis range (relative distance in bp).
-t <thres>	Output threshold or score. The program extracts reference tags that are enriched (\geq) or depleted in target tags according to a given threshold <i><thres></i> . This is not a mandatory parameter and it is set to 0 by default.

Options are:

Option	Description
-h	Show the program usage.
-d	Produce debugging output and check whether the input SGA is properly sorted.
-c	Count cut-off value for all input tags (def=10).
-o	Oriented strand processing. It means reverting the chromosome axis when the reference feature is on the – strand.
-r	Reverse extraction process. The program extracts reference sites that are depleted in target sites.
-q	Report feature B tag counts as 'feature name =<int>' (in the 6 th field).

Here is an example of using the program *chipscore* to extract transcription start sites from the Eukaryotic Promoter Database EPD that are enriched in CTCF peaks. The input data for this analysis are provided in two separate files: *ES_CTCF_peaks.sga* for CTCF peaks in mouse ES cells, and *Mm_EPDbnew_001_mm9.sga* for TSS's in mouse.

These files need first to be merged before they can be processed by *chipscore*.

```
sort -s -m -k1,1 -k3,3n -k4,4 Mm_EPDbnew_001_mm9.sga ES_CTCF_peaks.sga |> merged.sga
```

```
chipscore -A "TSS" -B "CTCF_P" -b -300 -e 10 -t 1 -c 1 -o merged.sga |> TSS_CTCF-enriched.out
```

where:

<i>merged.sga</i>	is the input file containing the list of mapped TSSs and CTCF peaks
<i>-A "TSS"</i>	is the reference feature
<i>-B "CTCF"</i>	is the target feature (CTCF peaks)
<i>-b</i>	is beginning of the range considered in the analysis
<i>-e</i>	is the end of the range
<i>-c</i>	is the count cut-off value
<i>-t</i>	is the count threshold
<i>-o</i>	is the oriented option (TSSs are oriented features)
<i>TSS_CTCF-enriched.out</i>	is the output file containing those TSS's that contain a CTCF peak.

Alternatively, one can run the following pipeline:

```
sort -s -m -k1,1 -k3,3n -k4,4 Mm EPDnew_001_mm9.sga | chipscore -A "TSS" -B "CTCF_P" |  
-b -300 -e 10 -t 1 -c 1 -o > TSS_CTCF-enriched.out
```

3. Auxiliary Tools for data reformatting

We also provide a series of auxiliary tools (most of them perl scripts) that can be used to perform format conversion tasks.

Most of the ChIP-seq data sets come in BAM or BED formats. If you have BAM files, and you need to convert them to SGA format, the steps to follow are:

1. Find out which genome assembly has been used to generate the BAM files, and make sure that the chromosome names agree with the naming scheme of the UCSC genome browser.
2. Then use the following type of command.

```
bamToBed -i reads.bam | bed2sga.pl -f<feature> -s <species> | sort -s -k1,1 -k3,3n -k4,4 |  
compactsga > reads.sga
```

The program *bamToBed* belongs to the **BEDTools** package, a suite of utilities for comparing genomic features that can be installed from:

<https://github.com/arq5x/bedtools2>

bamToBed converts BAM alignments to BED format. The '*bed2sga.pl*' tool is a perl script that converts BED files to SGA format. To find instructions how to use it, type:

```
bed2sga.pl -h
```

SGA format requires a *feature* field (the second field) containing a character string that defines the genomic feature represented by the file. The BED file does not have an equivalent field. You therefore need to supply a feature for the conversion via the *-f* option. The *species* is the name used by UCSC for the genome assembly to which the

reads have been mapped (e.g. hg18, mm9, dm3, etc.). *bed2sga.pl* will automatically convert UCSC chromosome names into corresponding NCBI/RefSeq accession numbers. Currently we provide chromosome ID conversion for the following assemblies: hg19, *hg18*, *mm8*, *mm9*, *mm10*, *ce4*, *ce6*, *dm3*, *dm6*, *danRer4*, *danRer5*, *danRer7*, *rn4*, *rn5*, *panTro2*, *sacCer3*, *araTha1*, *spo2* (*S.pombe*), and *plasFalc1* (*P.falcariipum*). The *species* is only required if one wants to translate UCSC-based chromosome names into NCBI RefSeq identifiers for comparison with data that use NCBI identifiers.

The information required to map NCBI RefSeq identifiers into chromosome numbers is provided by a binary table called *chro_idx.nstorage*. The binary file *chro_idx.nstorage* includes a hash table in Perl that, for each assembly, stores chromosome number-NCBI identifier pairs as well as chromosome lengths indexed by chromosome NCBI identifiers.

The reasons why sorting is required have been explained before. The program '*compactsga*' is a C program which merge lines corresponding to the same genome position (identical sequence name, position and strand). For instance, the two input lines

NC_000001.9	H3K4me3	5011	+	1
NC_000001.9	H3K4me3	5011	+	1

would be replaced by the following single line:

NC_000001.9	H3K4me3	5011	+	2
-------------	---------	------	---	---

Compacting sga files saves space and reduces program execution time, but unlike sorting is not formally required by the ChIP-Seq tools.

The *bed2sga.pl* has two basic modes of operations, *centered* and *oriented*. In the *centered* mode, the midpoint between the *start* and *end* position from the BED line (2nd and 3rd field) will be used as position. In the *oriented* mode, the conversion depends on the *strand* indicated in the BED file (6th field). If the strand is + then the value of the *start* field will be incremented by one and used as position in the SGA file. If the strand is -, the value of the *end* field will be used as position in the SGA file. Incrementing the start position in *oriented* mode is necessary because the BED format has a “zero-based” numbering system for chromosomal regions whereas SGA has a 1-based numbering system. The behavior of the two conversion modes is illustrated by the following example:

BED input:

chr1	100000266	100000291	.	0	+
chr1	100000383	100000408	.	0	-

SGA output, centered mode:

NC_000001.9	CHIPSEQ	100000278	+	1
NC_000001.9	CHIPSEQ	100000395	-	1

SGA output, oriented mode:

```
NC_000001.9 CHIPSEQ 100000267 + 1
NC_000001.9 CHIPSEQ 100000408 - 1
```

By the default, the conversion mode depends on the contents of the BED file. If the strand field (which is optional in BED) is presented, the conversion will be done in *oriented* mode. Otherwise, the conversion will be done in centered mode, and the strand field will be set to zero. However, centered mode can be forced by the command line option *-c*.

The most common command pipeline to perform BED-to-SGA conversion is the following:

```
bed2sga.pl -f <feature> -s <species> reads.bed | sort -s -k1,1 -k3,3n -k4,4 | compactsga >
reads.sga
```

There are several additional reformatting programs that may be useful in certain situations. The C program *featreplace* replaces all feature names in an SGA file by a new name:

```
featreplace -f <feature> old.sga > new.sga
```

The formatting programs are typically used together with other formatting tools belonging to publicly available software packages.

One example of such software is the liftOver program from UCSC that can be used to convert chromosomal coordinates in a BED file from one genome assembly to another, e.g.:

```
liftOver input.bed /db/liftOver/mm8ToMm9.over.chain.gz output.bed trash
```

Note that liftOver chain files can be downloaded from the UCSC site.

Additional conversion scripts include:

<i>sga2bed.pl</i>	SGA to BED
<i>partit2bed.pl</i>	Output of Partitioning tool (special SGA) to BED
<i>fps2sga.pl</i>	FPS to SGA
<i>sga2fps.pl</i>	SGA to FPS
<i>gff2sga.pl</i>	GFF to SGA
<i>sga2gff.pl</i>	SGA to GFF
<i>sga2wigFS.pl</i>	SGA to WIG (wiggle) fixedStep format
<i>sga2wigVS.pl</i>	SGA to WIG variableStep format

The last two programs produce custom track files that can be uploaded to the UCSC genome browser for visualization. The FPS (Functional Position Set) format is the specific format used by the Signal Search Analysis (SSA) tools, a set of programs developed by our group for motif analysis (<http://seqanswers.com/wiki/FPS>).

4. Appendix: ChIP-Seq tools summary

Program Name	Description	Input format	Output Format	Language
Correlation Tools				
chipcor	Correlates two features corresponding to two ChIP-seq tag (or read) distributions	SGA	Text (histogram)	C
chipextract	Extracts target features tags that fall into a distance range relative to a reference feature	SGA	Text (heatmap)	C
chipscore	Extracts ChIP-seq tags from feature A that are enriched (or depleted) in feature B tags	SGA	SGA	C
Peak finding Tools				
chippeak	Locates ChIP-seq signal peaks	SGA	SGA	C
chippart	Partitions the genome into signal-rich and signal-poor segments	SGA	Special SGA	C
Pre-processing Tools				
chipcenter	Shifts ChIP-seq tags to estimated center positions of DNA fragments	SGA	SGA	C
counts_filter	Filters out or select all ChIP-seq tags that occur within a set of DNA regions (e.g. repeat mask)	SGA	SGA	C
Reformatting Tools				
compactsga	Merges equal ChIP-seq tag positions into a single line adjusting the count field	SGA	SGA	C
featreplace	Changes the name of the <feature> field	SGA	SGA	C
Perl conversion Tools				
bed2sga.pl	Converts BED format into SGA format	BED	SGA	Perl
fps2sga.pl	Converts FPS format into SGA format	FPS	SGA	Perl
gff2sga.pl	Converts GFF format into SGA format	GFF	SGA	Perl
partit2sga.pl	Converts the output from the chippart tool into centered SGA format	Special SGA	BED	Perl
partit2bed.pl	Converts the output from the chippart tool into BED format	Special SGA	BED	Perl
partit2gff.pl	Converts the output from the chippart tool into GFF format	Special SGA	GFF	Perl
wigVS2sga.pl	Converts Wig Variable Step format into SGA format	Wig	SGA	
sga2bed.pl	Converts SGA format into BED format	SGA	BED	Perl
sga2gff.pl	Converts SGA format into GFF format	SGA	GFF	Perl
sga2fps.pl	Converts SGA format into FPS format	SGA	FPS	Perl
sga2wigVS.pl	Converts SGA format into Wig Variable Step format	SGA	Wig	Perl
sga2wigFS.pl	Converts SGA format into Wig Fixed Step format	SGA	Wig	Perl