HOMER

Software for motif discovery and next-gen sequencing analysis

---

Next-Generation Sequencing Analysis

ChIP-Seq is the best thing that happened to ChIP since the antibody. It is 100x better than ChIP-Chip since it escapes most of the problems of microarray probe hybridization. Plus it is cheaper, and genome wide. But ChIP-Seq is only the tip of the iceberg - there are many inventive ways to use a sequencer. Below are a list of the the more popular methods that will be covered below:

ChIP-Seq: Isolation and sequencing of genomic DNA "bound" by a specific transcription factor, covalently modified histone, or other nuclear protein. This methodology provides genome-wide maps of factor binding. Most of HOMER's routines cater to the analysis of ChIP-Seq data.

DNase-Seq: Treatment of nuclei with a restriction enzyme such as DNase I will result in cleavage of DNA at accessible regions. Isolation of these regions and their detection by sequencing allows the creation of DNase hypersensitivity maps, providing information about which regulatory elements are accessible in the genome.

MNase-Seq: Micrococcal Nuclease (MNase) is a restriction enzyme that degrades genomic DNA not wrapped around histones. The remaining DNA represents nucleosomal DNA, and can be sequencing to reveal nucleosome positions along the genome. This method can also be combined with ChIP to map nucleosomes that contain specific histone modifications.

RNA-Seq: Extraction, fragmentation, and sequencing of RNA populations within a sample. The replacement for gene expression measurements by microarray. There are many variants on this, such as Ribo-Seq (isolation of ribosomes translating RNA), small RNA-Seq (to identify miRNAs), etc.

GRO-Seq: RNA-Seq of nascent RNA. Transcription is halted, nuclei are isolated, labeled nucleotides are added back, and transcription briefly restarted resulting in labeled RNA molecules. These newly created, nascent RNAs are isolated and sequenced to reveal "rates of transcription" as opposed to the total number of stable transcripts measured by normal RNA-seq.

Unsolicited advice: If you are going to perform RNA-Seq, use a protocol for **STRAND-SPECIFIC** RNA-Seq. Why would you throw away the strand information?
More Unsolicited advice: Run controls!!!
Yet More Unsolicited advice: Be careful about GC-bias!!

(mini tutorial for each of these data types are on their way...)

HOMER offers solid tools and methods for interpreting Next-gen-Seq experiments. In addition to UCSC visualization support and peak finding [and motif finding of course], HOMER can help assemble data across multiple experiments and look at positional specific relationships between sequencing tags, motifs, and other features. You do not need to use the peak finding methods in this package to use motif finding.

Basic Analysis can be separated into the following steps for each experiment type:

1. Mapping to the genome (NOT performed by HOMER, but important to understand)
2. Creation Tag directories, quality control, and normalization. (**makeTagDirectory**)
3. UCSC visualization (**makeUCSCfile**, **makeBigWig.pl**)
4. Peak finding / Transcript detection / Feature identification (**findPeaks**)
5. Motif analysis (**findMotifsGenome.pl**)
6. Annotation of Peaks (**annotatePeaks.pl**)
7. Quantification of Data at Peaks/Regions in the Genome/Histograms and Heatmaps (**annotatePeaks.pl**)
8. Quantification of Transcripts (**analyzeRNA.pl**)

Additional analysis strategies:

- General sequence manipulation tools (**homerTools**)
- Miscellaneous Tools for Sharing Data between programs, etc. (**tagDir2bed.pl, bed2pos.pl, pos2bed.pl** ...)
- Finding overlapping or differentially bound peaks (**mergePeaks**, **getDifferentialPeaks**)
- ChIP-Seq analysis automation (**analyzeChIP-Seq.pl**)
- Description of file formats

NOTE: The current implementation is geared for single tag sequencing. **Most** of the types of experiments above don't necessarily gain much from paired-end sequencing (in terms of information/per bp).

Can't figure something out? Questions, comments, concerns, or other feedback: cbenner@ucsd.edu

HOMER

Software for motif discovery and next-gen sequencing analysis

---

Alignment of High-throughput Sequencing Data

Homer does not perform alignment - this is something that must be done before running homer. Several quality tools are available for alignment of short reads to large genomes. Check out this link for a list of programs that do short read alignment. BLAST, BLAT, and other traditional alignment programs, while great at what they do, are not practical for alignment of these types of data.

If you need help deciding on a program to use, I'll recommend Bowtie (it's nice and fast).

If you have a core that maps your data for you, don't worry about this step. However, in many cases there is public data available that hasn't been mapped to the genome or mapped to a different version of the genome or mapped with different parameters. In these cases it is nice to be able to map data yourself to keep a nice, consistent set of data for analysis.

Most types of ChIP-Seq/DNase-Seq/MNase-Seq and GRO-Seq simply need to be mapped to the genome, as they represent the sequencing of genomic DNA (or nascent RNA, which should not be spliced yet). If analyzing RNA-Seq, you may be throwing away interesting information about splicing if you simply align the data to the genome. If aligning RNA, I'd recommend sticking to the formal wear and trying Tophat, which does a good job of identifying splice junctions in your data.

Which reference genome (version) should I map my reads to?

Both the organism and the exact *version* (i.e. hg18, hg19) are very important when mapping sequencing reads. Reads mapped to one version are NOT interchangeable with reads mapped to a different version. I would follow this recommendation list when choosing a genome (Obviously try to match species or sub species when selecting a genome):

1. Do you have a favorite genome in the lab that already has a bunch of experiments mapped to it? Use that one.
2. Do any of your collaborators have a favorite genome?
3. Use the latest stable release - I would recommend using genomes curated at UCSC so that you can easily visualize your data later using the UCSC Genome Browser. (i.e. mm9, hg18)

Q: I'm changing genome versions, can I just "liftover" my data using UCSC liftover tool, or do I need to remap it to the new genome version?

If you want to do it right, you need to remap it. This is because some regions of the genome that are considered "unique" in one version may suddenly be found multiple times in the new version and vice versa, so using the liftover tool will yield different results from remapping. However, liftover is fine if you're looking for a quick and dirty solution. If you fell like cheating, as Chuck often does, try **convertCoordinates.pl**. - it's a wrapper that uses the "liftOver" program to migrate peak files and whole Tag Directores.

Should I trim my reads when mapping to the genome?

Depends. In the old days, the read quality dropped off quite a bit past ~30 bp, but these days even the end of sequencing reads are pretty high quality. In the end, I would recommend mapping ~32 bp reads with up the 3 mismatches, using only the uniquely alignable reads for downstream analysis. That will give you access to probably 80-90% of what is interesting in your data set.

I have barcodes and/or adpater sequences in my reads. Should I remove them first or just map them?

You should definitely remove the adapter sequences or other "non-biological" sequences before mapping. Various tools can accomplish this. You can check out homerTools for trimming sequences and dealing with adapters. Galaxy also has a nice variety of tools for accomplishing this type of stuff.

Example - Alignment with bowtie:

Step 1 - Build Index (takes a while, but only do this once):

After installing bowtie, the reference genome must first be "indexed" so that reads may be quickly aligned. You can download pre-made indices from the bowtie website (check for those here first). Otherwise, to perform make your own from FASTA files, do the following:

1. Download FASTA files for the unmasked genome of interest if you haven't already (i.e. from UCSC)
2. From the directory containing the FASTA files, run the "bowtie-build" command. For example, for hg18:
   - **/path-to-bowtie-programs/bowtie-build chr1.fa,chr2.fa,chr3.fa,...chrY.fa,chrM.fa hg18**
   - Where ... are the rest of the *.fa files. This command will take a long time to run, but will produce several files named **hg18.*.ebwt**
3. Copy the *.ebwt files to the bowtie indexes directory so that bowtie knows where to find them later:

- **cp *.ebwt /path-to-bowtie-programs/indexes/**

Step 2 - Align sequences with bowtie (perform for each experiment):

The most common output format for high-throughput sequencing is FASTQ format, which contains information about the sequence (A,C,G,Ts) and quality information which describes how certain the sequencer is of the base calls that were made. In the case of Illumina sequencing, the output is usually a "**s_1_sequence.txt**" file. In addition, much of the data available in the SRA, the primary archive of high-throughput sequencing data, is in this format. To map this data, run the following command:

/path-to-bowtie-programs/bowtie -q --best -m 1 -p <# cpu> <genome> <fastq file> <output filename>
Where <genome> would be hg18 from the index made above, <fastq file> could be "s_1_sequence.txt", and <output filename> something like "s_1_sequence.hg18.alignment.txt"

The parameters "--best" and "-m 1" are needed to make sure bowtie outputs only unique alignments. There are many options and many different ways to perform alignments, with different trade-offs for different types of projects - well beyond the scope of what I am describing here.

NOTE: HOMER contains automated parsing for uniquely aligned reads from output files generated with bowtie in this fashion. Homer also accepts *eland_result.txt and *_export.txt formats from the Illumina pipeline. If different programs are used, or special parsing of output files are needed, please parse/reformat alignment files to general BED format, which is also accepted by HOMER. HOMER also accepts SAM formatted file. If using BAM files, use "samtools view input.bam > output.sam" to convert to a SAM file.

Can't figure something out? Questions, comments, concerns, or other feedback: cbenner@ucsd.edu

HOMER

Software for motif discovery and ChIP-Seq analysis

Next-gen Sequencing Analysis: Creating a "Tag Directory" with makeTagDirectory

To facilitate the analysis of ChIP-Seq (or any other type of short read re-sequencing data), it is useful to first transform the sequence alignment into platform independent data structure representing the experiment, analogous to loading the data into a database. HOMER does this by placing all relevant information about the experiment into a "Tag Directory", which is essentially a directory on your computer that contains several files describing your experiment.

During the creation of tag directories, several quality control routines are run to help provide information and feedback about the quality of the experiment. During this phase several important parameters are estimated that are later used for downstream analysis, such as the estimated length of ChIP-Seq fragments.

Input Alignment Files

During this part of the analysis, any type of sequencing data can be use (ChIP-Seq/RNA-Seq/DNase-Seq, etc.). If these files are zipped (*.gz, *.zip, *.bz2), HOMER will automatically unzip, process, and re-zip them (if applicable) so you don't have to worry about this step.

To create a "Tag Directory", you must have alignment files in one of the following formats:

- BED format
- SAM format
- BAM format (HOMER will use "samtools view file.BAM > file.SAM" to covert to a SAM file, so "samtools" must be available)
- default bowtie output format
- *.eland_result.txt or *_export.txt format from the Illumina pipeline

If your alignment is in a different format, it is recommended that you convert it into a BED file format:

Column1: chromosome
Column2: start position
Column3: end position
Column4: Name (or strand +/-)
Column5: Number of reads at this position
Column6: Strand +/-

**Unfortunately, BED files are used in different ways by different groups. By default, HOMER assumes the 5th column of a BED files is the number of reads at that position. If this is NOT the case, add the option "-forceBED" in your command to tell HOMER to ignore this value.

Alternatively (or in combination), you can make tag directories from existing tag directories or from tag files (explained below).

If your input files are named "s_1_sequence.txt", or have a suffix such as *.fq or *.fastq, then you probably have raw sequence files without raw sequence alignment information. You need to align these sequences to the genome first. Learn about aligning data to the genome here.

Paired-End Reads

Unfortunately, HOMER does NOT explicitely support paired-end reads... yet. This is primarily due to the fact that our group hasn't used paired-end sequencing for any of these applications (nor is there much data in the literature using paired-end reads for applications like ChIP-Seq, with the exception of RNA-Seq), but this is changing and support for them will be available in the near future.

For now, the best option is to separate the paired-end reads and treat them as separate single-end runs (or just use one of the two reads).

Creating Tag Directories

To make a tag directory, run the following command:

makeTagDirectory <Output Directory Name> [options] <alignment file1> [alignment file 2] ...

Where the first argument must be the output directory (required). If it does not exist, it will be created. If it does exist, it will be overwritten.
An example:

makeTagDirectory Macrophage-PU.1-ChIP-Seq/ pu1.lane1.bed pu1.lane2.bed pu1.lane3.bed

Several additional options exist for **makeTagDirectory**. The program attempts to guess the format of your alignment files, but if it is unsuccessful, you can force the format with "**-format <X>**".

makeTagDirectory Macrophage-PU.1-ChIP-Seq/ pu1.lane1.bed pu1.lane2.bed pu1.lane3.bed -format bed
makeTagDirectory Macrophage-H3K4me1-ChIP-Seq/ s_1_sequence.align.sam -format sam

Sometimes BED file alignments contain stupid values in the 5th column, such as

quality information etc. HOMER will treat this value as the number of reads aligning to the same location. If this is not how the value is used, add "-forceBED" to ignore the value found int the 5th column of a BED file.

makeTagDirectory Macrophage-PU.1-ChIP-Seq/ pu1.lane1.bed pu1.lane2.bed pu1.lane3.bed -format bed -forceBED

To combine tag directories, for example when combining two separate experiments into one, do the following:

makeTagDirectory Combined-PU.1-ChIP-Seq/ -d Exp1-ChIP-Seq/ Exp2-ChIP-Seq/ Exp3-ChIP-Seq/

What does makeTagDirectory do?

makeTagDirectory basically parses through the alignment file and splits the tags into separate files based on their chromosome. As a result, several *.tags.tsv files are created in the output directory. These are made to very efficiently return to the data during downstream analysis. This also helps speed up the analysis of very large data sets without running out of memory.

In the end, your output directory will contain several *.tags.tsv files, as well as a file named "**tagInfo.txt**". This file contains information about your sequencing run, including the total number of tags considered. This file is used by later programs to quickly reference information about the experiment, and can be manually modified to set certain parameters for analysis.

makeTagDirectory also performs several quality control steps shown below.

Basic Quality Control Analysis:

The following 4 basic quality control results are produced by default and are relatively inexpensive in terms of processing power to create while paring alignment files into Tag Directories, and do not require any extra information to produce. These files are meant to be opened with a text editor or graphed using EXCEL or similar program. More detailed descriptions of these files and how to interpret them are found in the sequencing technique-specific tutorials.

Basic Tag information

tagInfo.txt - Contains basic configuration information, such as the total number of reads, the total number of unique positions with aligned reads (by genome and chromosome), and various other statistics. One of the more important parameters is "fragmentLengthEstimate=##", which provides an estimate of the length of fragments used for sequencing.

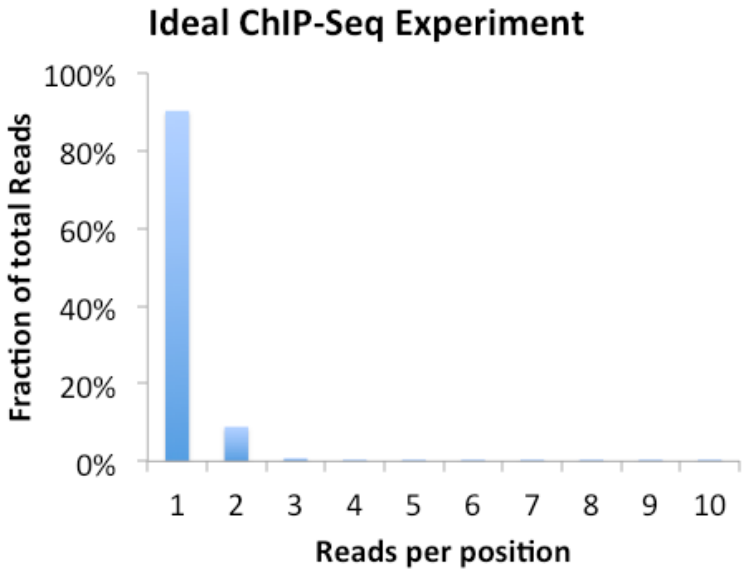
Read Length Distribution

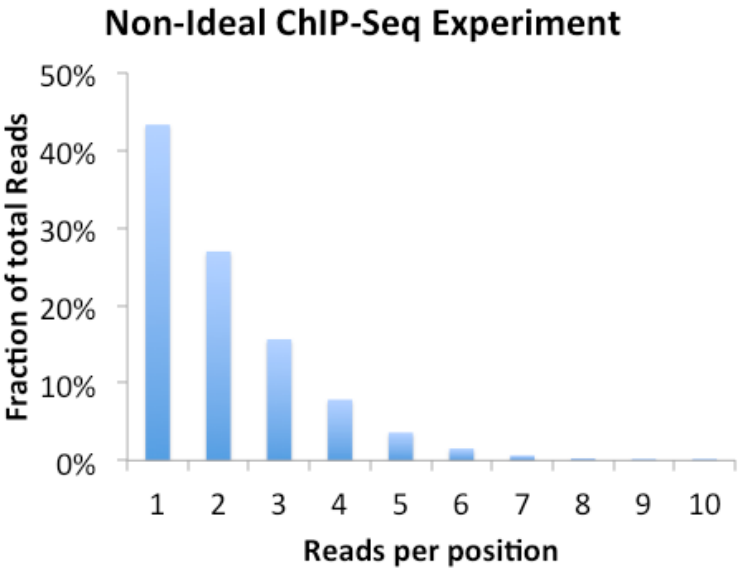
tagLengthDistribution.txt - File contains a histogram of read lengths used

for alignment.

Clonal Tag Distribution

tagCountDistribution.txt - File contains a histogram of clonal read depth, showing the number of reads per unique position. If an experiment is "over-sequenced", you start seeing the same reads over and over instead of unique reads. Sometimes this is a sign there was not enough starting material for sequencing library preparation. Below are examples of ideal and non-ideal results - in the case of the non-ideal experiment, you probably don't want to sequence that library anymore.





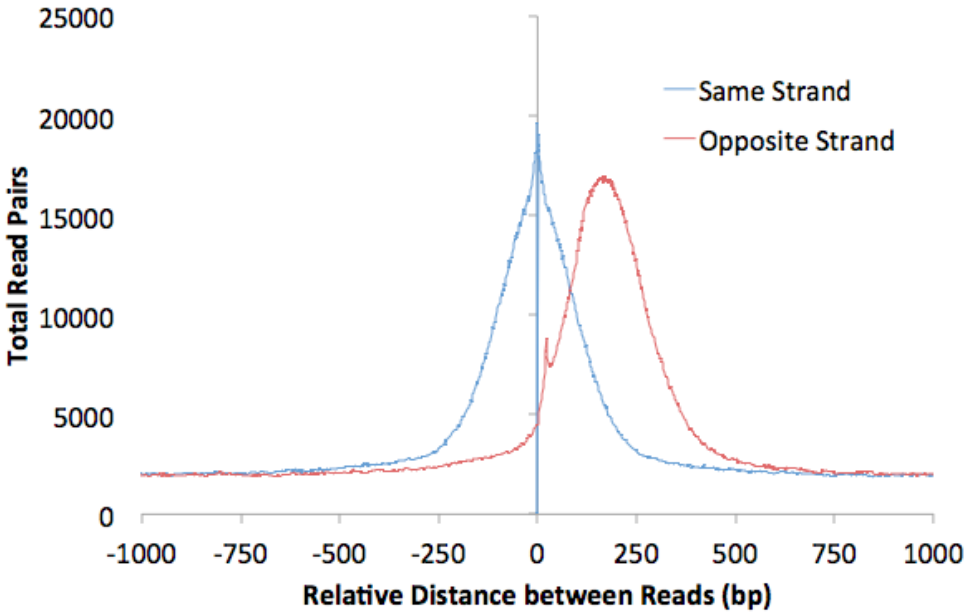
If the experiment is highly clonal and not expected to be, it might help to clean up the downstream analysis by forcing tag counts at each position to be no greater than *x*, where *x* is usually 1. To do this rerun the makeTagDirectory command and add "-tbp <#>" where # is the maxium tags per bp.

Autocorrelation Analysis

tagAutocorrelation.txt - The autocorrelation routine creates a distribution of

distances between adjacent reads in the genome. If reads are mapped to the same strand, they are added to the first column. If adjacent reads map to different strands, they are added to the 2nd column. The results from autocorrelation analysis are very useful for troubleshooting problems with the experiment, and are used to estimate the fragment length for ChIP-Seq and MNase-Seq. The fragment length is estimated by finding the position were the "opposite strand" distribution is maximum. HOMER will use this value as the fragment length unless overridden with the option "**-fragLength <#>**".

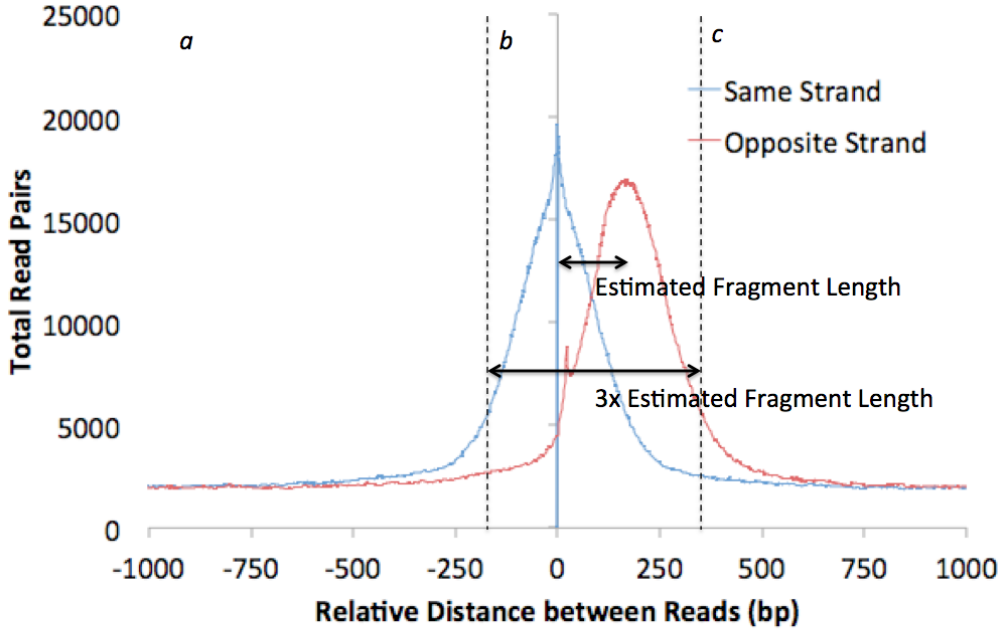
Different types of experiments (i.e. ChIP-Seq vs. DNase-Seq) produce different looking autocorrelation plots, and more detailed discussion of these differences can be found in the individual tutorials. Below is an example from a successful ChIP-Seq experiment:



HOMER also uses the autocorrelation results to guess what type of experiment you conducted. It computes 3 statistics:

- Same strand fold enrichment: Enrichment of reads on the same strand within 3x the estimated fragment length
- Diff strand fold enrichment: Enrichment of reads on different strands within 3x the estimated fragment length
- Same / Diff fold enrichment: Difference between enrichment of reads on the same strand or different strands

Below is a schematic to visualize how these are calculated (keep in mind that the background is calculated out to +/- 2kb):



Same strand and Diff Strand Fold Enrichment: $\frac{\text{Density } b}{\text{Density } a \text{ \& } c}$

Same/Diff Fold Enrichment: $\frac{\text{Density } b \text{ (same strand)}}{\text{Density } b \text{ (Opposite Strand)}}$

Depending on the value of the autocorrelation quality statistics, HOMER will guess what your experiment is:

- If the Same/Diff Fold Enrichment is > 8 fold, it's a great chance the sample is strand-specific RNA. If HOMER decides the sample is probably RNA due to the difference between same strand and different strand numbers, it will automatically set the estimated fragment length to 75 bp. This is because it is difficult to estimate the fragment length for RNA/GRO-seq. To manually set the fragment length, use "**-fragLength <#>**"
- If both the "Same Strand Fold Enrichment" and "Diff Strand Fold Enrichment" are both greater than 1.5 fold, there is good chance you're looking at a working ChIP-Seq experiment.

This is meant as immediate feedback, and not a definitive declaration of the quality or type of your data. We've found it useful to help identify wrong annotations in sample IDs, for example, and helped give us an idea about how well an experiment worked. Good ChIP-Seq antibodies give "Same Strand Fold Enrichment" values well above 5.0 for transcription factors. However, in may cases good results can be extracted out of experiments with lower enrichments, even those that yield a "Guessing sample is ChIP-Seq - may have low enrichment with lots of background" message.

Sequence Bias Analysis:

Invaluable information about a sequencing experiment can be found by examining the relationship between sequencing reads and the genomic sequence the came from. Sequence bias analysis is not performed by default. To perform this analysis, you must provide the *genome* and the "-checkGC" option.

makeTagDirectory <Output Directory Name> [options] -genome <genome> -checkGC <alignment file1> ...
i.e. **makeTagDirectory Macrophage-PU.1-ChIP-Seq -genome mm9 -checkGC pu1.alignment.bed**

This analysis will produce several output files in addition to the basic quality control analysis described above.

An important prerequisite for analyzing sequence bias is that the appropriate *genome* must by configured for use with HOMER. In version v3.1, HOMER now handles custom/arbitrary genomes. Instead of intalling/configuring a genome, you can specify the path to a file or directory containing the genomic sequence in FASTA format. The genome can be in a single FASTA file, or you specify a directory where where each chromosome is in a separate file (named chr*XXX*.fa or chr*XXX*.fa.masked). In either case, the FASTA headers must contain the chromosome names followed by white space, i.e. ">chr blahblahblah", not ">chr1-blahblahblah", or prefereably only ">chr1".

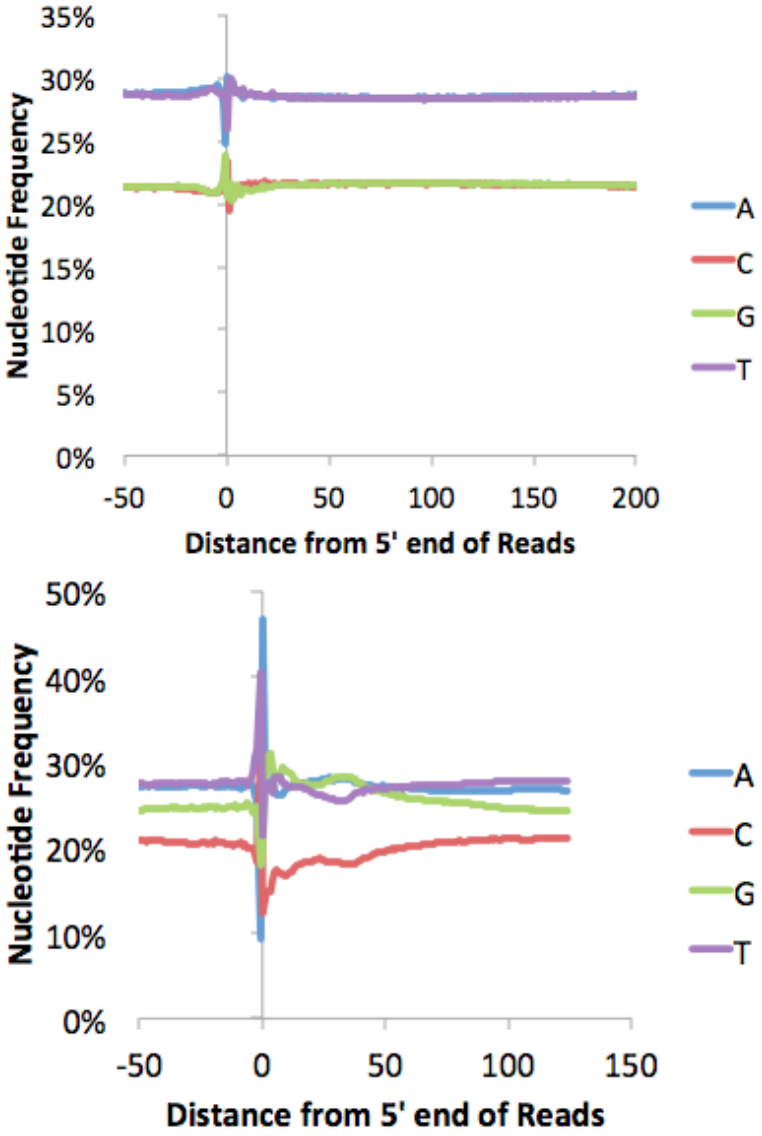
NOTE: If using a sequencing type other than ChIP-Seq or MNase-Seq, you may want to add "-fragLength <#>" since it may be difficult for HOMER to automatically determine the size of fragments used for sequencing.

Genomic Nucleotide Frequency relative to read positions

tagFreq.txt - Calculates the nucleotide and dinucleotide frequencies as a function of distance from the 5' end of all reads.
tagFreqUniq.txt - Same as tagFreq.txt, however individual genomic positions are only counted once. If 10 reads mapped to the same position, those nucleotide counts would be added 10 times for "tagFreq.txt", but only once for "tagFreqUniq.txt".

By default, HOMER calculates this frequency from -50 bp to +50 bp relative to the end of the estimated fragment (e.g. not +50bp relative to the 36 bp read, but +50 bp relative to the 200 bp ChIP-fragment). This can be changed using the options "**-freqStart <#>**" and "**-freqEnd <#>**"

Below are some examples of this file graphed with EXCEL. One is a pretty typical ChIP-Seq file, the other Chuck wouldn't say where it came from, but he's pretty sure there might be a problem with how they performed their experiment... Quite a bit can be learned from looking at these types of plots (see individual tutorials of additional details)

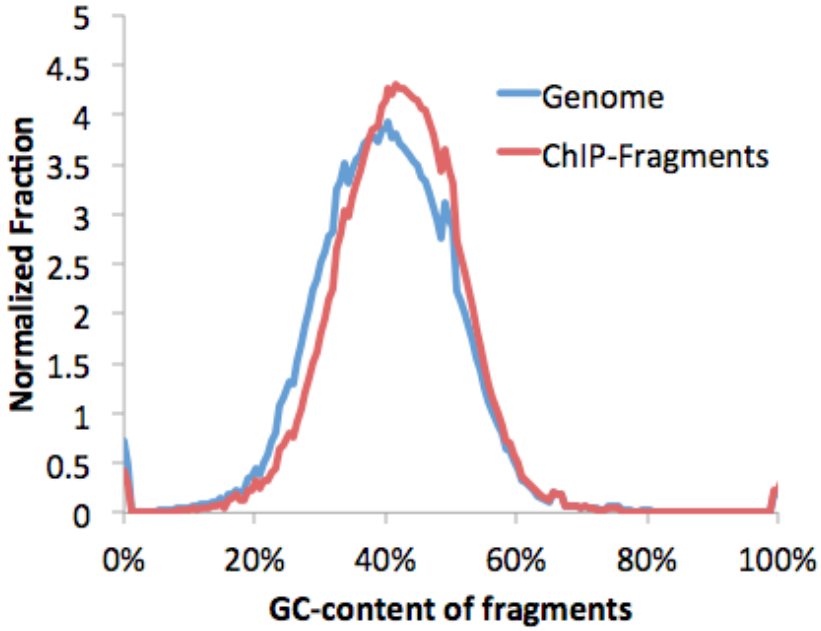


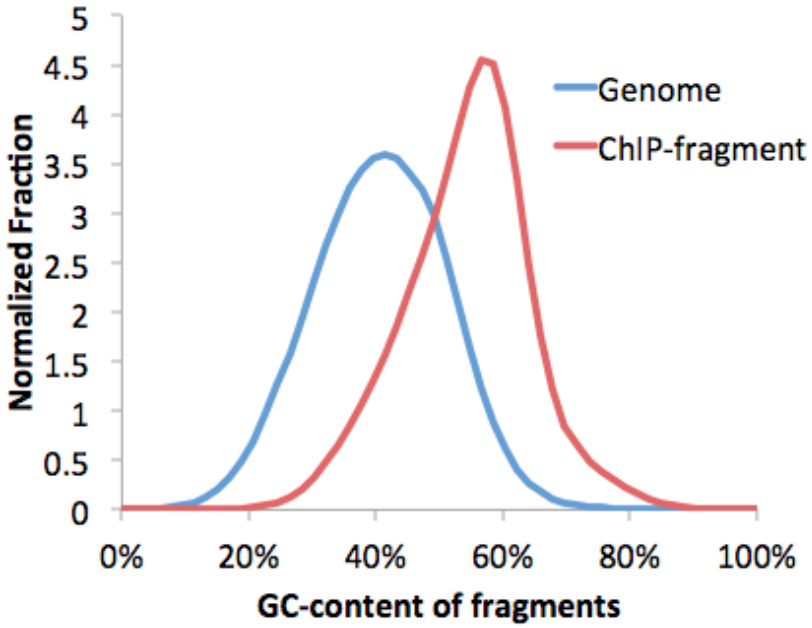
Fragment GC% Distribution

tagGCcontent.txt - Distribution of fragment GC%-content. GC% is calculated from the 5' end position of the read to end of the "fragment", not the end of the read. For ChIP-Seq and MNase-Seq, the fragment length is automatically estimated with pretty high confidence. If you are using another type of sequencing or want to set this manually, be sure to specify the desired fragment length manually ("**-fragLength <#>**").

genomeGCcontent.txt - Distribution of fragment GC%-content at each location in the genome (i.e. expected distribution)

These are very important files to check for any sequencing experiment. Due to the numerous steps of library preparation involving amplification, size selection and gel extraction, it is very easy for the average GC%-content of the sequencing library to "shift". This can be a disastrous problem. Consider the following:





The problem with a GC% shifted sample is that even if the sample is random sequence, you will start to show "enrichment" at places with high GC-content in the genome, such as at CpG Islands. This is unfortunate because most GC-rich areas are at transcription start sites, which might make you think the experiment worked, when in reality the sample was boiled instead of placed in the freezer. See below to learn about GC-normalization.

Sequence Bias GC normalization:

HOMER has built in routines for normalizing GC-bias in a sequencing experiment. In general, normalizing for GC-content is tricky. First, you need to decide if it is worth normalizing at all.

When should I normalize for GC-content in my sequencing experiment?

First you must be able to identify that there is a GC-bias problem. If the average GC% is within 5% of the expected genomic average, you're probably ok. If you are comparing several experiments of similar type, and they all have the same approximate GC-bias, you're also probably better off just comparing the experiments as they are. However, if one of your replicates is much more GC-rich than another, similar sample, you may want to consider normalizing it.

For some experiments, it tough to know what the expected GC-content should be. For example, if sequencing H3K4me3 ChIP-Seq data, which is typically found near CpG Islands, you may expect the GC% to be higher. However, even with H3K4me3 ChIP-Seq, most ChIP experiments are not very efficient, and most of the DNA being sequenced is background. As a result, the average GC content should not be too far off the expected genomic average. You can always "try" normalizing.

If the GC-bias is severe, you might be better off repeating the experiment. Normalization essentially destroys information, so repeating the experiment (or at least the sequencing library preparation) may be a good move. When re-preping the sample, try amplifying less and when extracting DNA from gels, dissolve the gels at room temperature.

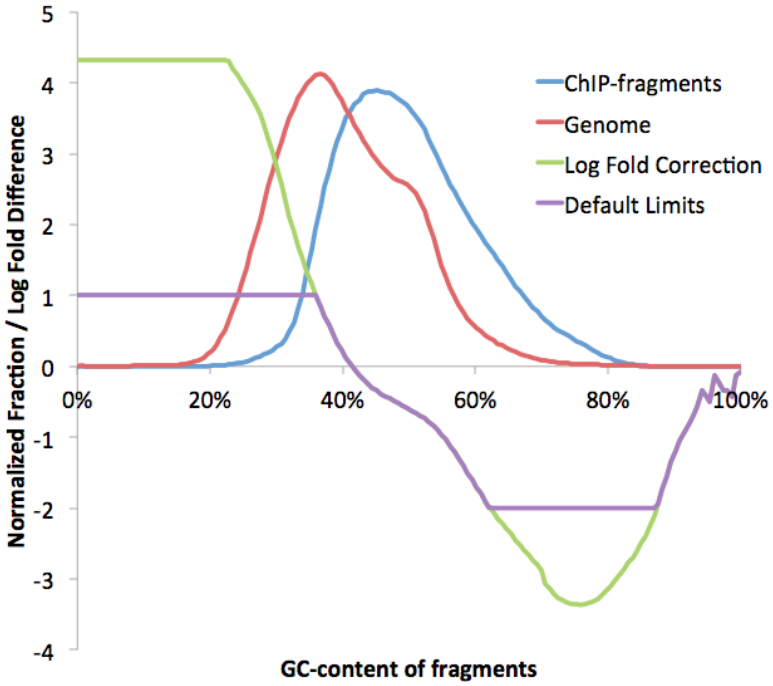
Normalizing tags for GC-bias

To normalize for GC-bias, run the same **makeTagDirectory** command, but you must specify the genome (i.e. "**-genome hg18**") and a target normalization file ("**-normGC <filename>**"). To normalize to the expected genomic average, use "**-normGC default**".

makeTagDirectory <Tag Directory> -genome <genome> -normGC <GC profile file profile> <alignment file 1> [alignment file 2] ...

i.e. makeTagDirectory Macrophage-PU.1-GC -genome mm9 -normGC default aligned.reads.bed

The normalization procedure is actually very simple. HOMER calculates the distribution of fragment GC%-contents, then for each range of GC%, normalizes the tag counts to the expected distribution. If your sample is more GC-rich that expected, reads in GC-rich regions will be reduced to a fractional value (limited by "**-minNormRatio <#>**"). Likewise, if reads are in AT-rich regions, their values will be increased (limited by "**-maxNormRatio <#>**"). HOMER is happy to deal with fractional tag values - there is no reason why a sequencing read can't be counted as a "half-a-read", for example. The biggest problem problem with **increasing** the value of a read is that it is akin to creating information that doesn't exist, so HOMER puts a tight cap on that adjustment to only 2-fold (change with "**-maxNormRatio <#>**"). The resulting normalization information is recorded in the **tagGCnormalization.txt** file.



Often, the expected genome GC-content is not appropriate. To normalize to a custom GC-profile, provide a file after **-normGC** option ("**-normGC <filename>**"). This file should be formatted just like the **tagGCcontent.txt** files found in the directories. In fact, the idea is that you can normalize one experiment to another by providing the experiments **tagGCcontent.txt** file as the argument for **-normGC**. One potential strategy is to combine several experimental files across different experiments into a single tag directory, and then use the tagGCcontent.txt file from this directory to normalize each of the other experiment, normalizing each experiment to the average from all of the experiments.

Command line options of makeTagDirectory command:

Usage: parseAlignment <directory> <alignment file 1> [file 2] ... [options]

Creates a platform-independent 'tag directory' for later analysis.
Currently BED, eland, bowtie, and sam files are accepted. The program will try to automatically detect the alignment format if not specified.
Existing tag directories can be added or combined to make a new one using -d/-t
If more than one format is needed and the program cannot auto-detect it properly, make separate tag directories by running the program separately, then combine them.

Options:
-fragLength <# | given> (Set estimated fragment length - given: use read lengths)
By default treats the sample as a single read ChIP-Seq experiment
-format <X> where X can be: (with column specifications underneath)

bed - BED format files:
(1:chr,2:start,3:end,4:+/- or read name,5:# tags,6:+/-)
bowtie - output from bowtie (run with --best -k 2 options)
(1:read name,2:+/-,3:chr,4:position,5:seq,6:quality,
7:NA,8:mismatch info)
eland_result - output from basic eland
(1:read name,2:seq,3:code,4:#zeroMM,5:#oneMM,6:#twoMM,7:chr,
8:position,9:F/R,10-:mismatches
eland_export - output from illumina pipeline (22 columns total)
(1-5:read name
info,9:sequence,10:quality,11:chr,13:position,14:strand)
eland_extended - output from illumina pipeline (4 columns total)
(1:read name,2:sequence,3:match stats,4:positions[,])
sam - SAM formatted files (use samTools to covert BAMs into SAM if you have BAM)
-keep (keep one mapping of each read regardless if multiple equal mappings exist)
-forceBED (if 5th column of BED file contains stupid values, like mapping quality
instead of number of tags, then ignore this column)
-d <tag directory> [tag directory 2] ... (add Tag directory to new tag directory)
-t <tag file> [tag file 2] ... (add tag file i.e. *.tags.tsv to new tag directory)
-single (Create a single tags.tsv file for all "chromosomes" - i.e. if >100 chromosomes)
-tbp <#> (Maximum tags per bp, default: no maximum)

GC-bias options:
-genome <genome version> (To see available genomes, use "-genome list")
-checkGC (check Sequence bias, requires "-genome")
-freqStart <#> (offset to start calculating frequency, default: -50)
-freqEnd <#> (distance past fragment length to calculate frequency, default: +50)
-normGC <target GC profile file> (i.e. tagGCcontent.txt file from control experiment)
Use "-normGC default" to match the genomic GC distribution
-minNormRatio <#> (Minimum deflation ratio of tag counts, default: 0.25)
-maxNormRatio <#> (Maximum inflation ratio of tag counts, default: 2.0)

Next: [Creating UCSC Genome Browser visualization files]

Can't figure something out? Questions, comments, concerns, or other feedback: cbenner@ucsd.edu

HOMER

Software for motif discovery and next-gen sequencing analysis

---

Visualizing Experiments with the UCSC Genome Browser

The UCSC Genome Browser is quite possibly one of the best computational tools ever developed. Not only does it contain an incredible amount of data in a single application, it allows users to upload custom information such as data from their ChIP-Seq experiments so that they can be easily visualized and compared to other information.

Making Genome Browser Files

The basic strategy HOMER uses is to create a bedGraph formatted file that can then be uploaded as a custom track to the genome browser. This is accomplished using the **makeUCSCfile** program. To make a ucsc visualization file, type the following:

makeUCSCfile <tag directory> -o auto

i.e. **makeUCSCfile PU.1-ChIP-Seq/ -o auto**
(output file will be in the PU.1-ChIP-Seq/ folder named PU.1-ChIP-Seq.ucsc.bedGraph.gz)

The "-o auto" with make the program automatically generate an output file name (i.e. TagDirectory.ucsc.bedGraph.gz) and place it in the tag directory which helps with the organization of all these files. The output file can be named differently by specifying "-o outputfilename" or by simply omitting "-o", which will send the output of the program to *stdout* (i.e. add " > outputfile" to capture it in the file outputfile). It is recommended that you zip the file using **gzip** and directly upload the *zipped* file when loading custom tracks at UCSC.

To visualize the experiment in the UCSC Genome Browser, go to Genome Browser page and select the appropriate genome (i.e. the genome that the sequencing tags were mapped to). Then click on the "add custom tracks" button (this will read "manage custom tracks" once at least one custom track is loaded). Enter the file created earlier in the "Paste URLs or data" section and click "Submit".

Problems Loading UCSC Files

The most common problem encountered while loading UCSC files is to see "position exceeds chromosome length" or something to that effect. This is usually caused by one of two problems:

1. You are trying to load the file to the wrong genome assembly. Make sure the assembly is correct!

2. Some of your tags are mapping outside the reference chromosome - this can be caused by mapping to non-standard assemblies or by some alignment programs. To remove all reads outside of the UCSC chromosome lengths, you can run the program **removeOutOfBoundsReads.pl**.

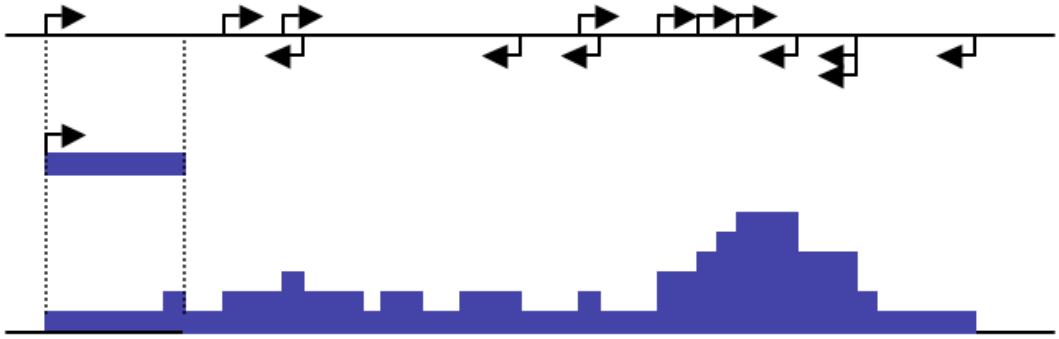
removeOutOfBoundsReads.pl <tag directory> <genome>
i.e. removeOutOfBoundsReads.pl PU.1-ChIP-Seq/ mm9

After running the program, you can rerun **makeUCSCfile**.

What does makeUCSCfile do?

The program works by approximating the ChIP-fragment density at each position in the genome. This is

done by starting with each tag and extending it by the estimated fragment length (determined by autocorrelation, or it can be manually specified using "-fragLength <#>"). The ChIP-fragment density is then defined as the total number of overlapping fragments at each position in the genome. Below is a diagram that depicts how this works:



As great as the UCSC Genome Browser is, the large size of recent ChIP-Seq experiments results in custom track files that are *very* large. In addition to taking a long time to upload, the genome browser has trouble loading excessively large files. To help cope with this, the **makeUCSCfile** program works by specifying a target file size when zipped (default 50MB). In order to meet the specified target file size, makeUCSCfile merges adjacent regions of tag density levels by their weighted average to reduce the total number lines in the final bedGraph file. If you have trouble loading getting your file to load, try reducing the size of the file using the "**-fsize <#>**" option (i.e. "-fsize 2e7"). To force the creation of larger files, use a very large file size (i.e. "-fsize 1e50") - this will create a file that does not merge any regions and displays a "native" view of the data.

Tags can be visualized separately for each strand using the "-strand separate" option.

Changing the Resolution

By default, makeUCSCfile uses the "**-fsize <#>**" option to determine how man reads to essentially "skip" when making the output file. You can also manually set the resolution.

In an effort to reduce the size of large UCSC files, one attractive option is to reduce the overall resolution of the file. By default, **makeUCSCfile** will make full resolution (i.e. 1 bp) files, but this can be changed by specifying the "**-res <#>**" option. For example, "-res 10" will cause changes in ChIP-fragment density to be reported only every 10 bp.

Normalization of UCSC files

In order to easily compare ChIP-fragment densities between different experiments, **makeUCSCfile** will normalize density profiles based on the total number of mapped tags for each experiment. As with other programs apart of HOMER, the total number of tags is normalized to 10 million. This means that tags from an experiment with only 5 million mapped tags will count for 2 tags apiece. The total of tags to normalize to can be changed using the "**-norm <#>**" option.

Separating data from different strands / RNA-Seq

You can specify that HOMER separate the data based on the strand by using the "**-strand <...>**" option. This is useful when looking at strand-specific RNA-Seq/GRO-Seq experiments. The following options are available:

-strand both : default behavior, for ChIP-Seq/MNase-Seq etc.
-strand separate : separate data by strand, for RNA-Seq/GRO-Seq
-strand + : only show the positive strand (i.e. Watson strand) data
-strand - : only show the negative strand (i.e. crick strand) data

Creating bigWig files with HOMER

Some data sets of very large, but you still want to see all of the details from your sequencing in the UCSC Genome Browser. HOMER can produce bigWig files by running the conversion program for you (**bedGraphToBigWig**). The only catch is that **you must have access to a webserver** where you can post the resulting bigWig file - this is because instead of uploading the whole file to UCSC, the browser actually looks for the data file on YOUR webserver and grabs only the parts it needs. Slick, eh. Chuck uses this all the time for big experiments.

Before even trying to make bigWigs, you must download the **bedGraphToBigWig** program from UCSC and place it somewhere in your executable path (i.e. the /path-to-homer/bin/ folder). This called directly by HOMER to create the BigWig files.

Using the makeBigWig.pl Script

To make bigWig files easier to generate, HOMER includes a program creatively named "**makeBigWig.pl**" that automates all of the steps below.

makeBigWig.pl <tag directory> <genome> [special options] [makeUCSC file options] -webDir /path-to-web-fold/ -url http://webserverURL/bigwigFold/
i.e. **makeBigWig.pl PU.1-ChIP-Seq/ mm9 -webDir /var/www/bigWigs/ -url http://ChuckNorrisU.edu/bigWigs/**

If you are visualizing strand specific data (i.e. RNA-Seq), specify "**-strand**". The **-url** and **-webDir** are the directories are the web URL directory and file system directory where the bigWigs will be stored, respectively. Recent changes to UCSC require that the chromosome sizes be specified exactly. If having trouble, the current version of HOMER has the option "**-chromSizes <filename>**" so that you can specify the sizes explicitly.

Making bigWigs from scratch

This is a quick description of what HOMER is doing. To make a bigWig, add the "**-bigWig <chrom.sizes file> -fsize 1e20**" parameters to your makeUCSCfile command. When making a bigWig, you usually want to see all of the tag information, so make sure the "-fsize" options is large. You also need to specify an output file using "**-o <bigwigfilename>**" and also capture the stdout stream using "**> trackfileoutput.txt**". You can also use "**-o auto**". The "trackfileoutput.txt" will contain the header information that is uploaded as a custom track to UCSC. Recently, changes to UCSC require that HOMER know the exact size of the chromosomes when making the file - these should be placed in a file (<chrom.sizes> file). **makeBigWig.pl** and **makeMultiWigHub.pl** will generate these files automatically by analyzing the sequences in the genome directory.

After running the makeUCSCfile program with the bigWig options, you need to do the following:

1. Copy the *.bigWig file to your webserver location and make sure it is viewable over the internet.
2. Need to edit the "trackfileoutput.txt" file and enter the URL of your bigWig file (... bigDataUrl=http://server/path/bigWigFilename ...)
3. Upload the "trackfileoutput.txt" file to UCSC as a custom track to view your data.

For example:

makeUCSCfile <tag directory> -o auto -bigWig <chrom.sizes file> -fsize 1e20 > trackInfo.txt

i.e.
makeUCSCfile PU.1-ChIP-Seq/ -o auto -bigWig chrom.sizes -fsize 1e20 > PU.1-bigWig.trackInfo.txt
cp PU.1-ChIP-Seq/PU.1-ChIP-Seq.ucsc.bigWig /Web/Server/Root/Path/
** edit PU.1-bigWig.trackInfo.txt to have the right URL **

NOTE: As of now, a bigWig file can only be composed of a single track - if you want to separate the data by strands, do the following:

makeUCSCfile PU.1-ChIP-Seq/ -o PU.1.positiveStrand.bigWig -bigWig chrom.sizes -fsize 1e20 -strand + > PU.1-bigWig.trackInfo.positiveStrand.txt
makeUCSCfile PU.1-ChIP-Seq/ -o PU.1.negativeStrand.bigWig -bigWig chrom.sizes -fsize 1e20 -strand - > PU.1-bigWig.trackInfo.negativeStrand.txt
cp PU.1.positiveStrand.bigWig PU.1.negativeStrand.bigWig /Web/Server/Root/Path/
cat PU.1-bigWig.trackInfo.positiveStrand.txt PU.1-bigWig.trackInfo.negativeStrand.txt > PU.1-bigWig.trackInfo.both.txt
** edit PU.1-bigWig.trackInfo.both.txt to have the right URLs for both the negative and positive strands **

Creating Multi-Experiment Overlay Tracks

UCSC has recently added the option to create overlay tracks, where several bigWig files can be viewed in the same space with the help of transparent colors. The first example of this was the Encode Regulation Track, which showed H3K4me1/3 data from several cell types at the same time. This is very useful for large-scale data sets will many different experiments. In these cases it is just about impossible to get them on the screen together.

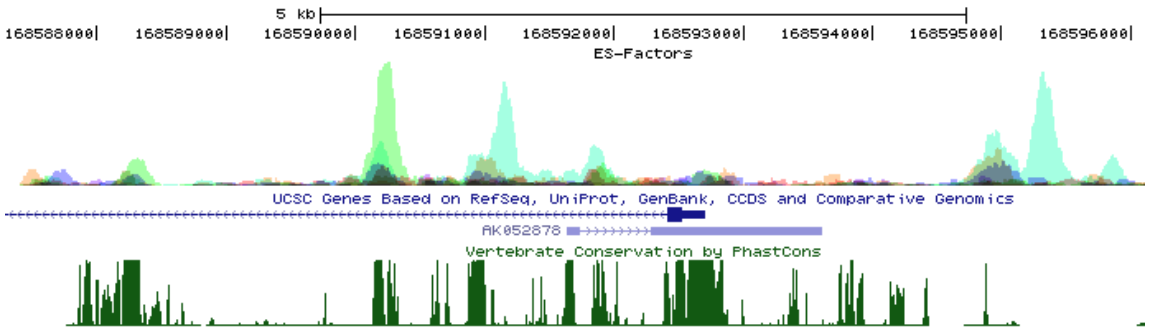
To make a "multi-wig hub", as we will refer to them, you need to make sure you have the **bedGraphToBigWig** program from UCSC, and a working webserver to host your files. If you can handle bigWigs in the section above, you can make multi-wig hubs.

The HOMER program to handle multi-wig hubs is called **makeMultiWigHub.pl**. It works essentially the same way as the makeBigWig.pl script, however, the syntax is a little different. The basic usage is:

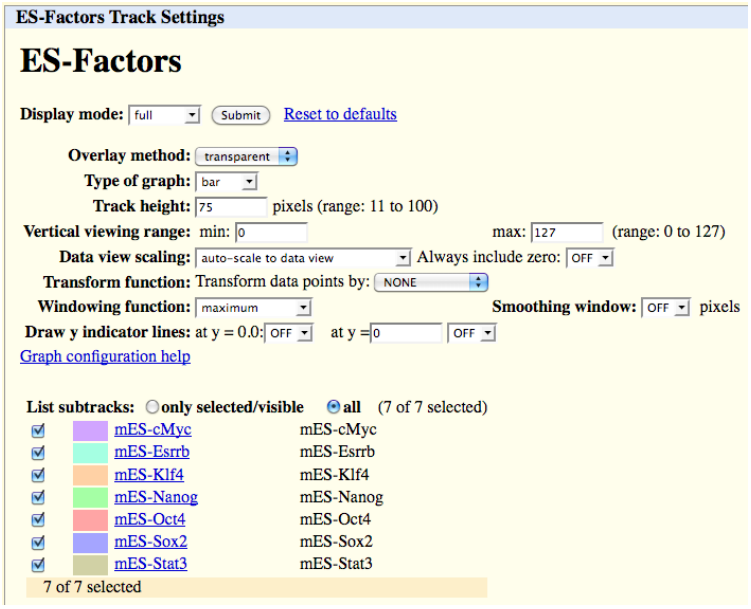
> **makeMultiWigHub.pl <hub name> <genome> [options] -d <tag directory1> <tag directory2> ...**
> i.e. **makeMultiWigHub.pl ES-Factors mm9 -d mES-Oct4/ mES-Sox2/ mES-Nanog/ mES-Klf4/ mES-Esrrb/ mES-cMyc/ mES-Stat3/**

NOTE: make sure you use the UCSC genome (e.g. mm9) and not the masked, bastardized HOMER version (mm9r).

The above example will produce a hub called "ES-Factors", composed of configuration files and bigWig files, and place it on your server in the directory specified by "**-webDir <directory>**". It will also provide you with a URL to the hub (dependent on the value of **-url <base url>**"). To load the Hub, click on "Track Hubs" on the UCSC browser (next to custom tracks button), and paste the URL in to the dialog box. The example above will look something like this:

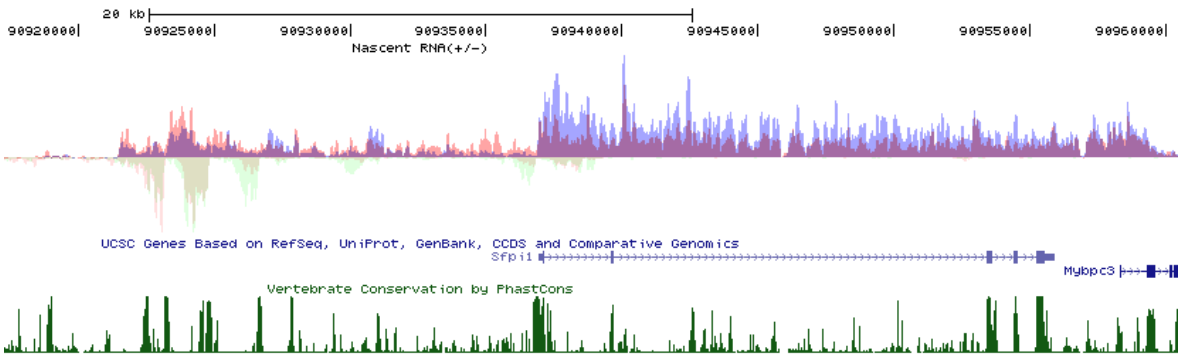


To figure out which factors correspond to which colors, click on the Blue Heading for the Hub in the settings area below the UCSC picture. Something like this should pop up:



Unfortunately, as of now editing hub information can only be done by directly modifying the hub files on the server. For example, to edit to colors, you must edit the "/webserver/directory/hubName/genome/trackDB.txt" file.

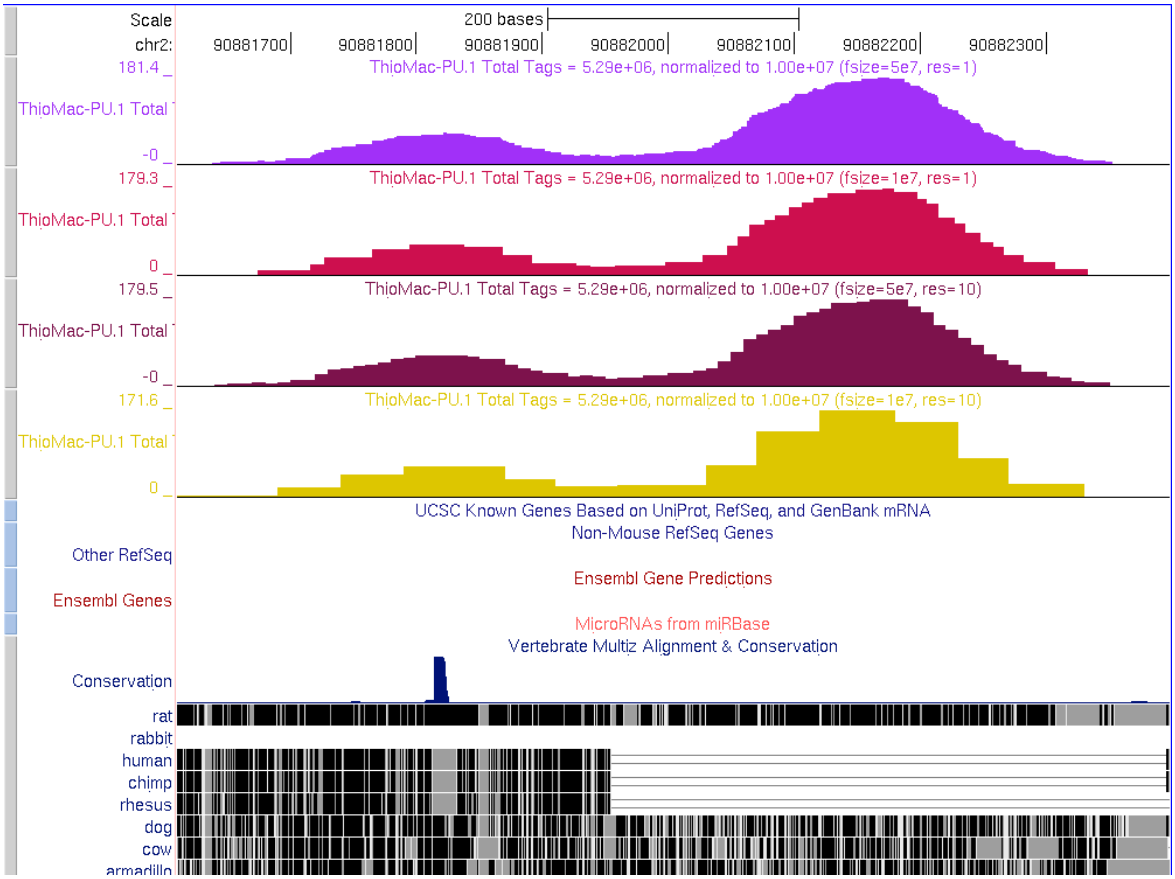
Because Hubs are so cool, HOMER will also do +/- strand RNA data right. Unfortunately, for now you can't mix stranded and non-stranded data in the same hub with the **makeMultiWigHub.pl** program. To visualize stranded information, add "**-strand**". Below is an example:



Examples of UCSC bedGraph files

The following shows what the same data set looks like changing options for file size (-fsize) and resolution (-res). Usually it's best to use one or the other.

1. -fsize 5e7 -res 1
2. -fsize 1e7 -res 1
3. -fsize 5e7 -res 10
4. -fsize 1e7 -res 10



Command line options for makeUCSCfile

Usage: makeUCSCfile <tag directory> [options]

Creates a bedgraph file for visualization using the UCSC Genome Browser

General Options:

-fsize <#> (Size of file, when gzipped, default: 5e7)
-strand <both|separate|+|-> (control if reads are separated by strand, default: both)
-fragLength <# | auto | given> (Approximate fragment length, default: auto)
-adjust <#> (Adjust edge of tag 3' by # bp, negative for 5', default: none[good for dnase])
-tbp <#> (Maximum tags per bp to count, default: no limit)
-res <#> (Resolution, in bp, of file, default: 1)
-lastTag (To keep ucsc happy, last mapped tag is NOT extended by default
    Using this option will allow extending of data past the last tag position)
-norm <#> (Total number of tags to normalize experiment to, default: 1e7)
-noadj (Do not normalize tag counts)
-neg (plot negative values, i.e. for - strand transcription)
-CpG (Show unmethylated CpG ratios)
-color <(0-255),(0-255),(0-255)> (no spaces, rgb color for UCSC track, default: random)
-bigWig <chrom.sizes> (creates a full resolution bigWig file and track line file)
    This requires bedGraphToBigWig to be available in your executable path
    Also, because how how bigWig files work, use "-strand -" and "-strand +"
    in separate runs to make strand specific files: "-strand separate" will not work
-o <filename|auto> (send output to this file - will be gzipped, default: prints to stdout)
    auto: this will place an appropriately named file in the tag directory
-name <...> (Name of UCSC track, default: auto generated)
-style <option> (See options below:)
    chipseq (standard, default)
    rnaseq (strand specific)
    tss (strand specific, single bp fragment length)

dnase (fragments centered on tag position instead of downstream)
methylated (single bp resolution of cytosine methylation)
unmethylated (single bp resolution of unmethylated cytosines)
-circos <chrN:XXX-YYY|genome> (output only a specific region for circos[no header])

Command line options for makeBigWig.pl

Script for automating the process of creating bigWigs

Usage: makeBigWig.pl <tag directory> <genome> [special options] [options]

Special Options for bigWigs [choose one, don't combine]:
-normal (ChIP-Seq style, default)
-strand (Strand specific, for RNA-Seq and GRO-Seq)
-dnase (Special options for Crawford-lab style DNase-Seq)
-cage (Special options for CAGE/TSS-Seq)
-cpg (Special options for mCpG/CpG)

Other options:
Whatever options you want to pass to makeUCSCfile
!!Warning!!: do not try to specify "-strand separate" - use the special option above.

File options:
-fsize <#> (Use to limit the size of the bigwig files)
-url <URL> (URL directory -no filename- to tell UCSC where to look)
-webdir <directory> (name of directory to place resulting bigWig file)
-update (overwrite bigwigs in the webDir directory, otherwise random numbers are added to make the file unique.

Current url target (-url): http://biowhat.ucsd.edu/bigWig/
Current web directory (-webDir): /data/www/bigWig/

You're going to want to modify the $wwwDir and $httpDir variables at the top of the makeBigWig.pl program file to accomidate your system so you don't have to specify -url and -webdir all the time.

Command line options for makeMultiWigHub.pl

Script for automating the process of creating multiWig tracks

Usage: makeMultiWigHub.pl <hubname> <genome> [options] -d <tag directory1> [tag directory2]...

Special Options for bigWigs [choose one, don't combine]:
-normal (ChIP-Seq style, default)
-strand (Strand specific, for RNA-Seq and GRO-Seq)
-dnase (Special options for Crawford-lab style DNase-Seq)
-cage (Special options for CAGE/TSS-Seq)
-cpg (Special options for mCpG/CpG)

Other options:
Whatever options you want to pass to makeUCSCfile
!!Warning!!: do not try to specify "-strand separate" - use the special option above.
Also, for the genome, do NOT use repeat version (mm9r) - use mm9 instead

File options:
-force (overwrite existing hub)
-fsize <#> (limit the file size of the bigwig files to this value)

-url <URL> (URL directory -no filename- to tell UCSC where to look)
-webdir <directory> (name of directory to place resulting hub directory)

Current url target (-url): http://biowhat.ucsd.edu/hubs/
Current web directory (-webDir): /data/www/hubs/

You're going to want to modify the $wwwDir and $httpDir variables at the top of the makeMultiWigHub.pl program file to accomidate your system so you don't have to specify -url and -webdir all the time.

Next: [Finding Peaks (ChIP-enriched regions) in the genome]

Can't figure something out? Questions, comments, concerns, or other feedback:
cbenner@ucsd.edu

HOMER

Software for motif discovery and next-gen sequencing analysis

---

Finding Enriched Peaks, Regions, and Transcripts

HOMER contains a program called **findPeaks** that performs all of the peak calling and transcript identification analysis. (Not to be confused with another peak finding program called FindPeaks, which was also very creatively named). Finding peaks is one of the central goals of any ChIP-Seq experiment, and the same basic principles apply to other types of sequencing such as DNase-Seq. The basic idea is to identify regions in the genome where we find more sequencing reads than we would expect to see by chance. There are number of different approaches one can use to find peaks, and correspondingly there are many different methods for identifying peaks from ChIP-Seq experiments. It is not required that you use HOMER for peak finding to use the rest of the tools included in HOMER (see below).

findPeaks has 3 basic modes of operation:

factor
: Peak finding for single contact or focal ChIP-Seq experiments or DNase-Seq. This type of analysis is useful for transcription factors, and aims to identify the precise location of DNA-protein contact. This type of peak finding uses a FIXED width peak size, which is automatically estimated from the Tag Autocorrelation.

histone
: Peak finding for broad regions of enrichment found in ChIP-Seq experiments for various histone marks. This analysis finds variable-width peaks.

groseq
: De novo transcript identification from strand specific GRO-Seq. This attempts to identify transcripts from nascent RNA sequencing reads.

HOMER does not perform de novo transcript isoform detection from spliced RNA-Seq. As we have just started analyzing RNA-Seq, and there is already a bunch of great work on this topic, there's no need to reinvent the wheel for this type of analysis. We recommend the Tophat/Cufflinks family of programs for RNA-Seq isoform detection.

Using findPeaks

To run **findPeaks**, you will normally type:

findPeaks <tag directory> -style <factor|histone|groseq> -o auto -i <control tag directory>

i.e. **findPeaks ERalpha-ChIP-Seq/ -style factor -o auto -i Control-ChIP-Seq/**

Where the first argument must be the tag directory (required). The other options are not required. The "**-style <...>**" option can be either "**factor**", "**histone**", or "**groseq**". Use the "**-i**" option to specify a control experiment tag directory (good idea when doing ChIP-Seq).

Output files

Use the "**-o <filename>**" to specify where to send the resulting peak file. If "**-o**" is not specified, the peak file will be written to ***stdout***.

If "**-o auto**" is specified, the peaks will be written to:
"**<tag directory>/peaks.txt**" (-style factor)
"**<tag directory>/regions.txt**" (-style histone)
"**<tag directory>/transcripts.txt**" and "**<tag directory>/transcripts.gtf**" (-style groseq)

The top portion of the peak file will contain parameters and various analysis information. This output differs somewhat for GRO-Seq analysis, and is explained in more detail later. Some of the values are self explanatory. Others are explained below:

```
# HOMER Peaks
# Peak finding parameters:
# tag directory = Sox2-ChIP-Seq
#
# total peaks = 10280
# peak size = 137
```

```
# peaks found using tags on both strands
# minimum distance between peaks = 342
# fragment length = 132
# genome size = 4000000000
# Total tags = 9908245.0
# Total tags in peaks = 156820.0
# Approximate IP efficiency = 1.58%
# tags per bp = 0.001907
# expected tags per peak = 0.523
# maximum tags considered per bp = 1.0
# effective number of tags used for normalization = 10000000.0
# Peaks have been centered at maximum tag pile-up
# FDR rate threshold = 0.001000
# FDR effective poisson threshold = 0.000000
# FDR tag threshold = 8.0
# number of putative peaks = 10800
#
# size of region used for local filtering = 10000
# Fold over local region required = 4.00
# Poisson p-value over local region required = 1.00e-04
# Putative peaks filtered by local signal = 484
#
# Maximum fold under expected unique positions for tags = 2.00
# Putative peaks filtered for being too clonal = 36
#
# cmd = findPeaks Sox2-ChIP-Seq -style factor -o auto
#
# Column Headers:
```

Genome size represents the total effective number of mappable bases in the genome (remember each base could be mapped in each direction)

Approximate IP effeciency describes the fraction of tags found in peaks versus. genomic background. This provides an estimate of how well the ChIP worked. Certain antibodies like H3K4me3, ERa, or PU.1 will yield very high IP efficiencies (>20%), while most rand in the 1-20% range. Once this number dips below 1% it's a good sign the ChIP didn't work very well and should probably be optimized.

Below the header information are the peaks, listed in each row. Columns contain information about each peak:

- Column 1: PeakID - a unique name for each peak (very important that peaks have unique names...)
- Column 2: chr - chromosome where peak is located
- Column 3: starting position of peak
- Column 4: ending position of peak
- Column 5: Strand (+/-)
- Column 6: Normalized Tag Counts - number of tags found at the peak, normalized to 10 million total mapped tags (or defined by the user)
- Column 7: (-style factor): Focus Ratio - fraction of tags found appropriately upstream and downstream of the peak center. (see below)
- (-style histone/-style groseq): Region Size - length of enriched region
- Column 8: Peak score (position adjusted reads from initial peak region - reads per position may be limited)
- Columns 9+: Statistics and Data from filtering

Two generic tools are available as part of HOMER to convert peak files to BED files and back. This will allow you to upload your peak files to the UCSC Genome Browser, or convert peak files in BED format from another program into a peak file that can be used by HOMER. These programs are named **pos2bed.pl** and **bed2pos.pl**, which can be used the following way:

pos2bed.pl peakfile.txt > peakfile.bed
bed2pos.pl peakfile.bed > peakfile.txt

Finding Transcription Factor Peaks with HOMER

To find peaks for a transcription factor use the **findPeaks** command:

findPeaks <tag directory> -style factor -o auto -i <input tag directory>

i.e. **findPeaks ERalpha-ChIP-Seq/ -style factor -o auto -i MCF7-input-ChIP-Seq**

Identification of Putative Peaks

If **findPeaks** is run in "**factor**" mode, a fixed peak size is selected based on estimates from the autocorrelation analysis performed during the **makeTagDirectory** command. This type of analysis maximizes sensitivity for identifying locations where the factor makes a single contact with the DNA. Peak size can be set manually with "**-size <#>**".

findPeaks loads tags from each chromosome, adjusting them to the center of their fragments, or by half of the estimated fragment length in the 3' direction (this value is also automatically estimated from the autocorrelation analysis). The fragment length can be specified manually using the "**-fragLength <#>**" option. It then scans the entire genome looking for fixed width clusters with the highest density of tags. As clusters are found, the regions immediately adjacent are excluded to ensure there are no "piggyback peaks" feed off the signal of large peaks. By default, peaks must be greater than 2x the peak width apart from on another (set manually with "**-minDist <#>**"). This continues until all tags have been assigned to clusters.

After all clusters have been found, a tag threshold is established to correct for the fact that we may expect to see clusters simply by random chance. Previously, to estimate the expected number of peaks for each tag threshold, HOMER would randomly assign tag positions and repeat the peak finding procedure. HOMER now assumes the local density of tags follows a Poisson distribution, and uses this to estimate the expected peak numbers given the input parameters much more quickly. Using the expected distribution of peaks, HOMER calculates the expected number of false positives in the data set for each tag threshold, setting the threshold that beats the desired False Discovery Rate specified by the user (default: 0.001, "**-fdr <#>**").

HOMER assumes the total number of mappable base pairs in the genome is 2,000,000,000 bp (** change from previous version. here 2e9 assumes the actual number of mappable positions is actually 2x [think + and - strand]) , which is "close enough" for human and mouse calculations. You can specify a different genome size using "**-gsize <#>**". HOMER also uses the reads themselves to estimate the size of the genome (i.e. that highest tag position on each chromosome). If this estimate is lower than the default, it will use that value to avoid using too large of a number on smaller genomes (For example, if you used findPeaks on *drosophila* data without specifying **"-gsize**").

It is important to note that this false discovery rate controls for the random distribution of tags along the genome, and not any other sources of experimental variation. Alternatively, users can specify the threshold using "**-poisson <#>**" to calculate the tag threshold that yields a cumulative poisson p-value less than provided or "**-tagThreshold <#>**" to specify a specific number tags to use as the threshold.

Filtering Peaks

The initial step of peak finding is to find non-random clusters of tags, but in many cases these clusters may not be representative of true transcription factor binding events. To increase the overall quality of peaks identified by HOMER, 3 separate filtering steps can be applied to the initial, putative peaks identified:

Using Input/IgG Sequencing as a Control

To use an Input or IgG sequencing run as a control (**HIGHLY RECOMMENDED**), you must first create a separate tag directory for the input experiment (see here). Additionally, you can use other cleaver experiments as a control, such as a ChIP-Seq experiment for the same factor in another cell or in a knockout. To find peaks using a control, type:

findPeaks <tag directory> -style factor -i <control tag directory> -o auto

i.e. **findPeaks ERalpha-ChIP-Seq/ -style factor -i Input-ChIP-Seq/ -o auto**

HOMER uses two parameters to filter peaks against a control experiment. First, it uses a fold change (which is sequencing depth-independent), requiring each putative peak to have 4-fold more normalized tags in the target experiment than the control (or specify a different fold change with "**-F <#>**"). In the case where there are no input tags near the putative peak, HOMER automatically sets these regions to be set to the average input tag coverage to avoid dividing by zero. HOMER also uses the poisson distribution to determine the chance that the differences in tag counts are statistically significant (sequencing-depth dependent), requiring a cumulative poisson p-value of 0.0001 (change with "**-P <#>**"). This effectively removes peaks with low tag counts for which there is a chance the differential enrichment is found simply due to sampling error.

One modification in recent versions of HOMER is the in the size of region used to compare experiment with control tags. Since control experiments are not always performed the same way, e.g. different fragment lengths, it helps to enlarge the peak size for the purposes of comparing experiments to ensure control reads found immediately outside of the peak region are still considered. By default, HOMER enlarges peaks by 2x to search the control experiment (change with "**-inputSize <#>**"). This may reduce specificity when trying to identify certain types of peaks.

Filtering Based on Local Signal

Our experience with peak finding is that often putative peaks are identified in regions of genomic duplication, or in regions where the reference genome likely differs from that of the genome being sequenced. This produces large regions of high tag counts, and if no Input/IgG sample is available, it can be hard to exclude these regions. Also, it may be advantageous to remove putative peaks that a spread out over larger regions as it may be difficult to pin-

point the important regulatory regions within them.

To deal with this, HOMER will filter peaks based on the local tag counts (similar in principle to MACS). Be default, HOMER requires the tag density at peaks to be 4-fold greater than in the surrounding 10 kb region. This can be modified using "**-L <#>**" and "**-localSize <#>**" to change the fold threshold and size of the local region, respectively. As with input filtering, the comparison must also pass a poisson p-value threshold of 0.0001, which can be set using "**-LP <#>**" option.

Filtering Based on Clonal Signal

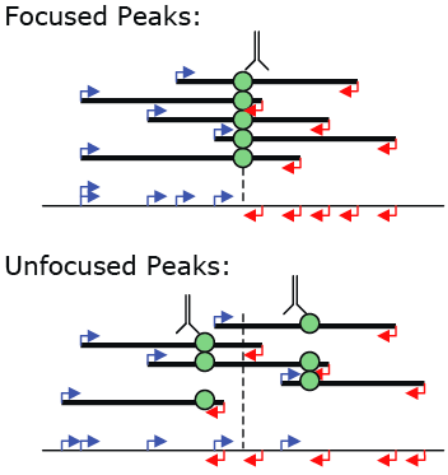
When we first sifted through peaks identified in ChIP-Seq experiments we noticed there are many peaks near repeat elements that contain odd tag distributions. These appear to arise from expanded repeats that result in peaks with high numbers of tags from only a small number of unique positions, even when many of the other positions withing the region may be "mappable". To help remove these peaks, HOMER will compare the number of unique positions containing tags in a peak relative to the expected number of unique positions given the total number of tags in the peak. If the ratio between the later and the former number gets to high, the peak is discarded. The fold threshold can be set with the "**-C <#>**" option (default: "-C 2"). Homer uses the **averageTagsPerPosition** parameter in the **tagInfo.txt** file adjust this calculation as to not over-penalize ChIP-Seq experiments that are already highly "clonal". If analyzing MNase or other restriction enzyme digestion experiments turn this option off ("**-C 0**");

Disabling Filtering

To disable Input, Local, or Clonal filtering set any combination of "**-F 0 -L 0 -C 0**".

Peak Centering and Focus Ratios

If the option "**-style factor**" or "**-center**" is specified, **findPeaks** will calculate the position within the peak with the maximum ChIP-fragment overlap and calculate a **focusRatio** for the peak. This is not always desired (such as with histone modifications). The focus ratio is defined as the ratio of tags located 5' of the peak center on either strand relative to the total number of tags in the peak. Peaks that contain tags in the ideal positions are more likely to be centered on a single binding site, and these peaks can be used to help determine what sequences are directly bound by a transcription factor. Unfocused peaks, or peaks with low (i.e <80%) focusRatios may be the result of several closely spaced binding sites or large complexes that cross-link to multiple positions along the DNA. Sometimes this means you have more than one binding site in close proximity (example below), but other times it means you cross-linked the $#@& out of your cells, or you have background in your sample.



To isolate focused peaks, you can use the getFocalPeaks.pl tool:

getFocalPeaks.pl <peak file> <focus % threshold> > focalPeaksOutput.txt

i.e. **getFocalPeaks.pl** ERpeaks.txt 0.90 > ERfocalPeaks.txt

Finding Enriched Regions of Variable Length

To find variable length peaks for histone marks, use the **findPeaks** command:

findPeaks <tag directory> -style histone -o auto -i <input tag directory>

i.e. **findPeaks H3K4me1-ChIP-Seq/ -style histone -o auto -i Input-ChIP-Seq**

If the option "**-style histone**" or "**-region**" is specified, **findPeaks** will stitch together enriched peaks into regions. Note that local filtering is turned off when finding regions. The most important parameters for region finding are the "**-size**" and "**-minDist**" (and of course the fragment length). First of all, "**-size**" specifies the width of peaks that will form the basic building blocks for extending peaks into regions. Smaller peak sizes offer better resolution, but larger peak sizes are usually more sensitive. By default, "**-style histone**" evokes a peak size of 500.

The second parameter, "**-minDist**", is usually used to specify the minimum distance between adjacent peaks. If "**-region**" is used, this parameter then specifies the maximum distance between putative peaks that is allowed if they are to be stitched together to form a region. By default this is 2x the peak size. If you think about histone modifications, the signal is never continuous in enriched regions, with reduced signal due to non-unique sequences (that can't be mapped to) and nucleosome depleted regions. "-minDist" informs **findPeaks** how big of a gap in the signal will be tolerated for adjacent peaks to be considered part of the same region. (by default "-style histone" sets this to 1000).

One thing to note is that you may have to play around with these parameters to get the results you want. If you look at the examples below, you could make arguments for using each of the tracks given what you're interested in and how you would define a "region".

For example: (in the example below, the default size comes from the autocorrelation estimate for the Macrophage-H3K4me1 dataset)

Default Parameters:
findPeaks Macrophage-H3K4me1/ -i Macrophage-IgG -size 150 -minDist 370 > output.txt (i.e. defaults)

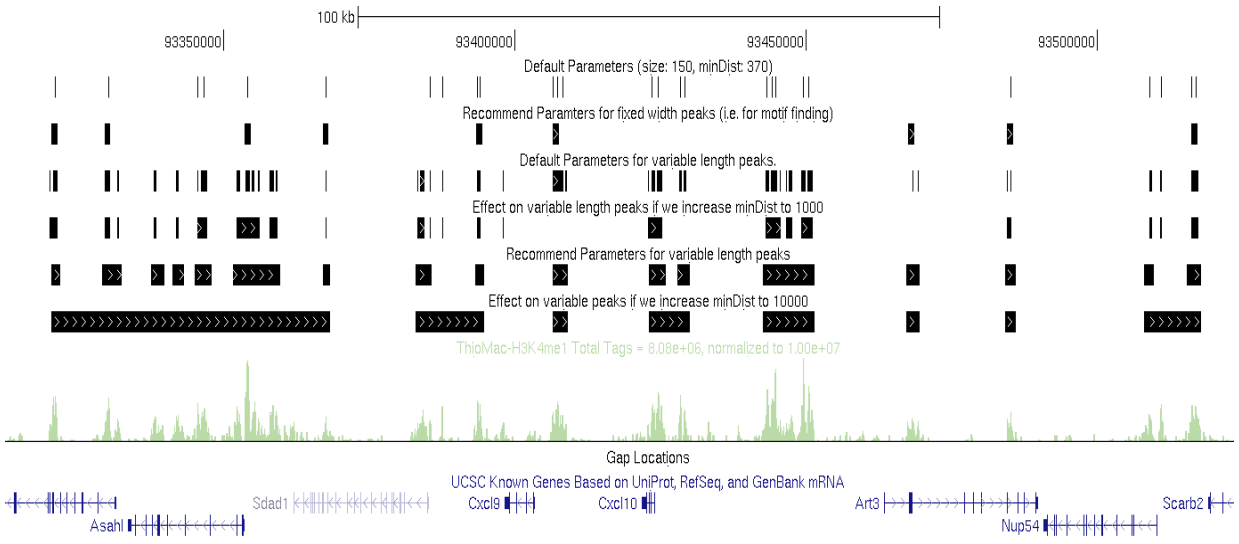
Recommend Parameters for fixed width peaks (i.e. for motif finding):
findPeaks Macrophage-H3K4me1/ -i Macrophage-IgG -size 1000 -minDist 2500 > output.txt

Default Parameters for variable length peaks.
findPeaks Macrophage-H3K4me1/ -i Macrophage-IgG -region -size 150 -minDist 370 > output.txt

Effect on variable length peaks if we increase minDist to 1000.
findPeaks Macrophage-H3K4me1/ -i Macrophage-IgG -region -size 150 -minDist 1000 > output.txt

Recommend Parameters for variable length peaks (H3K4me1 at least).
findPeaks Macrophage-H3K4me1/ -i Macrophage-IgG -region -size 1000 -minDist 2500 > output.txt

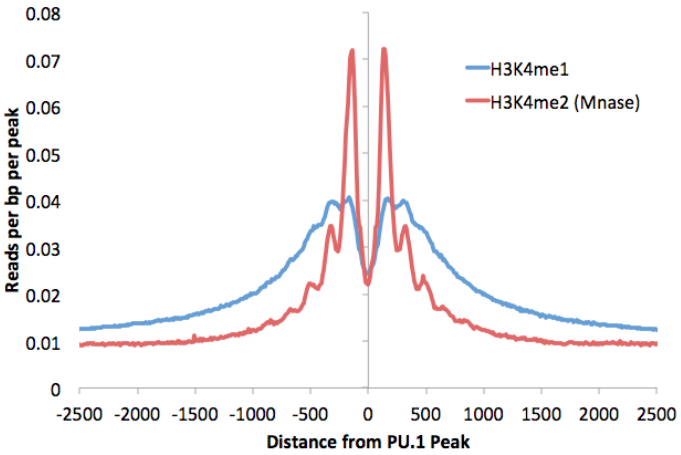
Effect on variable peaks if we increase minDist to 10000 (H3K4me1 at least).
findPeaks Macrophage-H3K4me1/ -i Macrophage-IgG -region -size 1000 -minDist 10000 > output.txt



Finding Histone Modification Peaks and Motif Finding

Finding peaks using histone modification data can be a little tricky - largely because we have very little idea what the histone marks actually do. If you want to find peaks in histone modification data with the purpose of analyzing them for enriched motifs, read this section. The problem with histone modification data (and some other types) is that the signal can spread over large distances. Trying to analyze large, variable length regions for motif enrichment is very difficult and not recommended. As such it is recommended sometimes a better idea to use fixed-size peak finding on histone marks (i.e. H3K4me1 enhancer mark) to improve motif analysis results. Using fixed size peaks helps make sure the assumptions needed for Motif Finding are, let's say, *less violated*, when dealing with regions of constant size.

There are two important differences between finding fixed width peaks for transcription factors and histone modifications. The first is the size of the peaks: Most histone modifications should be analyzed using a peak size in the range of 500-2000 usually (i.e. "-size 1000"). Below is an example of the histone mark distribution around transcription factor peaks (i.e. the things you're hoping Motif Finding will identify), which can be used to help estimate the parameters:



The other is that you should omit the "-center" option (i.e. do not specify "-style factor"). Since you are looking at a region, you do not necessarily want to center the peak on the specific position with the highest tag density, which may be at the edge of the region. Besides, in the case of histone modifications at enhancers, the highest signal will usually be found on nucleosomes surrounding the center of the enhancer, which is where the functional sequences and transcription factor binding sites reside. Consider H3K4me marks surrounding distal PU.1 transcription factor peaks. Typically, adding the -center option moves peaks further away from the functional sequence in these scenarios. An example for finding peaks:

findPeaks H3K4me1-ChIP-Seq/ -i Input-ChIP-Seq -size 1000 -o H3K4me1-ChIP-Seq/peaks.1kb.txt

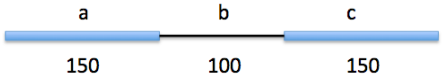
One issue with finding histone modification peaks using the defaults in HOMER is that the Local filtering step removes several of the peaks due to the "spreading" nature of many histone modifications. This can be good and bad. If you are looking for nice concentrated regions of modified histones, the resulting peaks will be a nice set for further analysis such as motif finding. However, if you are looking to identify every region in the cell that has an appreciable amount of modified histone, you may want to disable local filtering, or consider using the "**-region**" option below e.g.:

findPeaks H3K4me1-ChIP-Seq/ -i Input-ChIP-Seq -size 1000 -L 0 -o H3K4me1-ChIP-Seq/peaks.1kb.txt

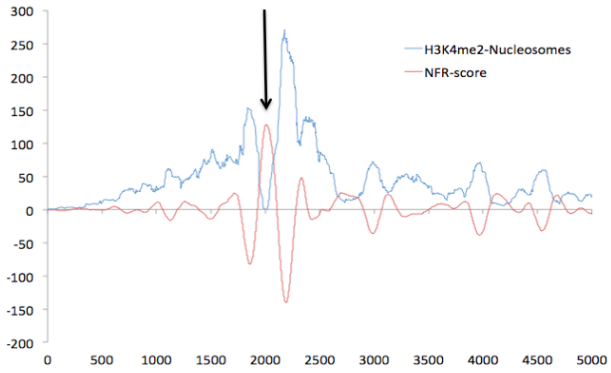
Also, if using MNase-treated chromatin (i.e. nucleosome mapping), you may want to use "**-C 0**" to avoid filtering peaks composed of highly clonal tag positions. These can arise from well positioned nucleosomes (or sites that are nicely digested by MNase at least).

Nucleosome Free Regions (NFR) -nfr

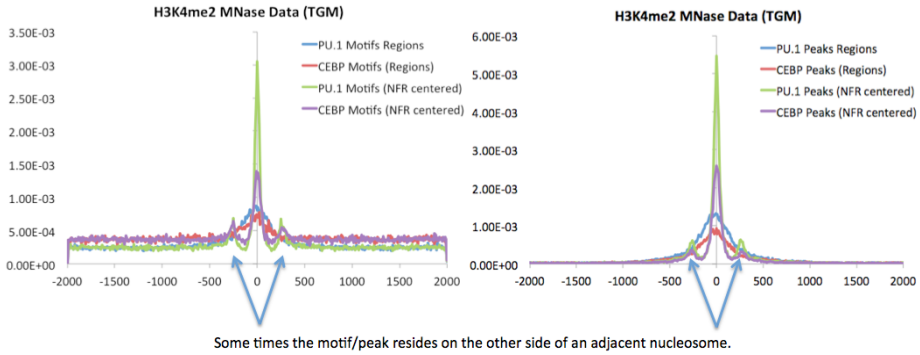
Just like peak centering for transcription factors, **findPeaks** has an option to look for large "dips" in the histone modification signal to infer where the primary nucleosome free region (NFR) is in the region. By adding "**-nfr**" to the command, HOMER will search for the location within the region that has the greatest differential in ChIP-signal and assign that location as the peak center.



NFR score = (a+c) – b
Where a, b, and c are normalized for their length



Works much better with MNase treated ChIP-Seq samples. The output file will then be centered on the NFR - this is useful for motif finding. By using NFRs instead of whole regions, you can narrow the search for regulatory elements down the +/- 100 bp instead of +/-500 or 1000 bp. Below is an example of ChIP-Seq peak locations with respect to center of H3K4me2-ChIP-Seq regions generated with and without the "**-nfr**" flag in Macrophages.



Some times the motif/peak resides on the other side of an adjacent nucleosome.

Peak finding and Sequencing Saturation

HOMER does not try to estimate sequencing saturation, which is the estimate of whether or not you have sequenced enough tags to identify all the peaks in a given experiment. Generally speaking, if you sequence more, you will get more peaks since your sensitivity will increase. The only real way to assess this is if you can somehow show that all of the "functional" or "real" peaks have high tag counts (i.e. are well above the threshold for identifying a peak), meaning that sequencing more is not likely to identify more "real" peaks. This generally cannot be determined by simply re-sampling the data and repeating the peak finding procedure - you need some sort of outside information to assess peak quality, such as motif enrichment or something else. Simply re-sampling will likely only tell you if you've gotten to the point where you are simply re-sequencing the same fragments again - i.e. becoming clonal.

Analyzing GRO-Seq: *de novo* transcript identification

To find transcripts directly from GRO-Seq, use the **findPeaks** command:

findPeaks <tag directory> -style groseq -o auto

i.e. **findPeaks Macrophage-GroSeq -style groseq -o auto**

GRO-Seq analysis does not make use of an control tag directory

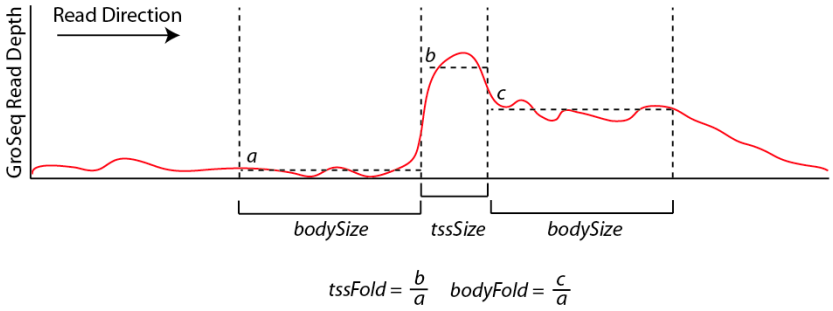
Basic Idea behind GRO-Seq Transcript identification

Finding transcripts using strand-specific GRO-Seq data is not trivial. GRO-Seq measures the production of nascent RNA, and is capable of revealing the loction of protein coding transcripts, promoter anti-sense transcripts, enhancer templated transcripts, long and short functional non-coding and miRNA transcripts, Pol III and Pol I transcripts, and whatever else is being transcribed in the cell nucleus. Identification and quantifiction of these transcripts is important for downstream analysis. Traditional RNA-Seq tools mainly focus on mRNA, which has different features than GRO-Seq, and are generally not useful for identifying GRO-Seq transcripts.

Important NOTE: Just as with ChIP-Seq, not all GRO-Seq data was created equally. Data created by different labs can

have features that make it difficult to have an single analysis technique that works perfectly for each one. As such, there are many parameters to play with the help get the desired results.

A large number of assumptions go into the analysis and are covered in more detail in the GRO-Seq tutorial (coming soon). In a nutshell, findPeaks tracks along each strand of each chromosome, searching for regions of continous GRO-Seq signal. Once it encounters high numbers of GRO-Seq reads, it starts a transcript. If the signal decreases significantly or disappears, the putative transcript is stopped. If the signal increases significantly (and sustainably), then a new transcript is considered from that point on. If the signal spikes, but overall does not increase over a large distance, it is considered an artifact or pause site and not considered in the analysis. Below is a chart that helps explain how the transcript detection works:



$$tssFold = \frac{b}{a} \quad bodyFold = \frac{c}{a}$$

By default, new transcripts are created when the tssFold exceeds 4 and bodyFold exceed 3 ("**-tssFold <#>**", "**-bodyFold <#>**"). A small pseudo-count is added to the tag count from region *a* above to avoid dividing by zero and helps serve to set a minimum threshold for transcript detection (**"-pseudoCount <#>"**, default: 1). Most transcripts show robust signal at the start of the transcript, and the *tssFold* helps select for these regions with high accuracy. The *bodyFold* is important for distinguishing between "spikes" in signal and real start sites; if a transcript is real, it's likely that increased levels of transcription follow behind the putative TSS. If the signal is roughly equal before and after the putative TSS, it is more likely to be an artifact.

To increase senstivity, HOMER tries to adjust the size of the *bodySize* parameter above since it essentially defines the resolution of the detected transcript. If there are a large number of GRO-Seq tags in a region, the *bodySize* can be small since there is adequate data to estimate the location of the transcript. However, if the data is relatively sparse, the bodySize needs to be large to get a reliable estimate of the level of the transcript. The minimum and maximum *bodySizes* are 600 and 10000 bp ("**-minBodySize <#>**", "**-maxBodySize <#>**"). HOMER uses the smallest *bodySize* that contains at least *x* number of tags, where *x* is determined as the number of tags where the chance of detecting a *bodyFold* change is less than 0.00001 assuming the read depth varies according to the poisson distribution (adjustable with "**-confPvalue <#>**", or directly with "**-minReadDepth <#>**"). The basic idea is that the threshold for tag counts must be high enough that we don't expect it to vary too much by chance.

Using uniquely mappable regions to improve results

Since some transcripts cover very large regions, there are many places where genomic repeats interrupt the GRO-Seq signal of continous transcripts. To help deal with this problem, HOMER can take advantage of mappability information to help estimate transcript levels where uniquely mapping sequencing reads is not possible. In general this information is not really that helpful for ChIP-Seq analysis, but in this case it can make an important difference. For now, HOMER only take specially formatted binary files available below. To use them, download the appropriate version and unzip the archive:

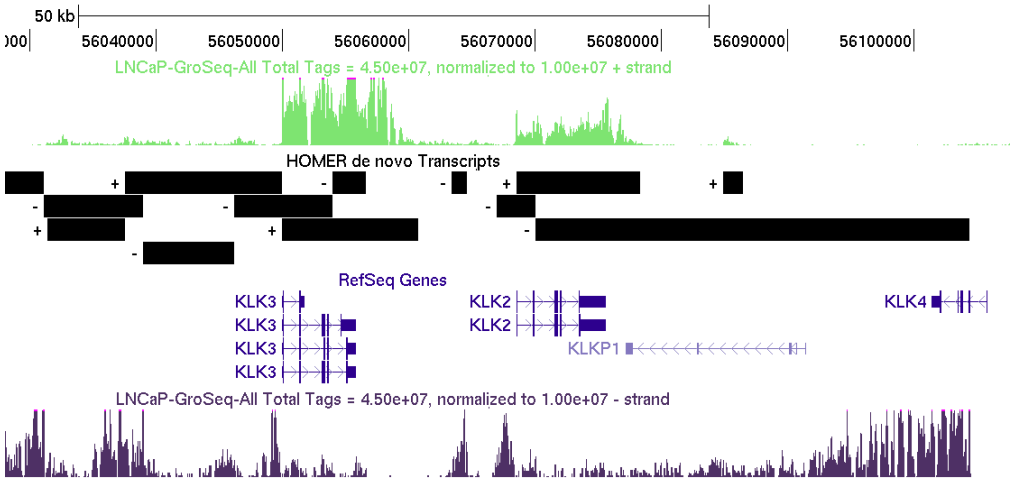
Human: hg18 hg19
Mouse: mm8 mm9
Fly: dm3

To use the uniq-map information, specify the location of the unzipped directory on the command line with "**-uniqmap <directory>**":

findPeak Macrophage-GroSeq/ -style groseq -o auto -uniqmap mm9-uniqmap/

GRO-Seq analysis output

Running findPeaks in groseq mode will produce a file much like the one produced for traditional peak finding, complete with a header section listing the parameters and statistics from the analysis. HOMER can also produce a GTF (gene transfer format) file for use with various programs. If "**-o auto**" is used to specify output, a "transcripts.gtf" file will be created in the tag directory. Otherwise, you can specify the name of the GTF output file by use "**-gtf <filename>**". The GTF file can also be easily uploaded to the UCSC Genome Browser to visualize your transcripts.



The GRO-Seq transcript detection works pretty well, but is likely to get some face-lifts in the near future.

Command line options for findPeaks

Usage: findPeaks <tag directory> [options]

Finds peaks in the provided tag directory. By default, peak list printed to stdout

General analysis options:
-o <filename|auto> (file name for to output peaks, default: stdout)
"-o auto" will send output to "<tag directory>/peaks.txt", ".../regions.txt",
or ".../transcripts.txt" depending on the "-style" option
-style <option> (Specialized options for specific analysis strategies)
factor (transcription factor ChIP-Seq, uses -center, default)
histone (histone modification ChIP-Seq, region based, uses -region -size 500 -L 0)
groseq (de novo transcript identification from GroSeq data)

chipseq/histone options:
-i <input tag directory> (Experiment to use as IgG/Input/Control)
-size <#> (Peak size, default: auto)
-minDist <#> (minimum distance between peaks, default: peak size x2)
-gsize <#> (Set effective mappable genome size, default: 4e9 [Think double stranded!])
-fragLength <#|auto> (Approximate fragment length, default: auto)
-inputFragLength <#|auto> (Approximate fragment length of input tags, default: auto)
-tbp <#> (Maximum tags per bp to count, 0 = no limit, default: auto)
-inputtbp <#> (Maximum tags per bp to count in input, 0 = no limit, default: auto)
-strand <both|separate> (find peaks using tags on both strands or separate, default:both)
-norm # (Tag count to normalize to, default 10000000)
-center (Centers peaks on maximum tag overlap and calculates focus ratios)
-region (extends start/stop coordinates to cover full region considered "enriched")

Peak Filtering options: (set -F/-L/-C to 0 to skip)
-F <#> (fold enrichment over input tag count, default: 4.0)
-P <#> (poisson p-value threshold relative to input tag count, default: 0.0001)
-L <#> (fold enrichment over local tag count, default: 4.0)
-LP <#> (poisson p-value threshold relative to local tag count, default: 0.0001)
-C <#> (fold enrichment limit of expected unique tag positions, default: 2.0)
-localSize <#> (region to check for local tag enrichment, default: 10000)
-inputSize <#> (Size of region to search for control tags, default: 2x peak size)
-fdr <#> (False discovery rate, default = 0.001)
-poisson <#> (Set poisson p-value cutoff, default: uses fdr)
-tagThreshold <#> (Set # of tags to define a peak, default: uses fdr)

GroSeq Options: (Need to specify "-style groseq"):
-tssSize <#> (size of region for initiation detection/artifact size, default: 300)
-minBodySize <#> (size of regoin for transcript body detection, default: 600)
-maxBodySize <#> (size of regoin for transcript body detection, default: 10000)
-tssFold <#> (fold enrichment for new initiation dectection, default: 4.0)
-bodyFold <#> (fold enrichment for new transcript dectection, default: 3.0)
-endFold <#> (end transcript when levels are this much less than the start, default: 25.0)

-fragLength <#> (Approximate fragment length, default: 150)
-uniqmap <directory> (directory of binary files specifying uniquely mappable locations)
Download from http://biowhat.ucsd.edu/homer/groseq/
-confPvalue <#> (confidence p-value: 0.00001)
-minReadDepth <#> (Minimum initial read depth for transcripts, default: auto)
-gtf <filename> (Output de novo transcripts in GTF format)
"-o auto" will produce <dir>/transcripts.txt and <dir>/transcripts.gtf

Links to alternative peak finding software

Lots of quality programs exist for finding peaks in ChIP-Seq data. Most use slightly different assumptions and peak definitions that result in slightly different sets of peaks. Many of these programs output peak files in BED format. To covert these to HOMER peak file format, use the bed2pos.pl program to convert the file (as of now, most HOMER programs work with BED files, so this isn't really necessary):

bed2pos.pl peaks.bed > peaks.txt

Peak finding software links:

- MACS (Liu Lab)
- ChIPSeq Peak Finder (Wold Lab)
- CCAT (Good for histone modifications)
- FindPeaks
- Probably hundreds of others: Google it!

Can't figure something out? Questions, comments, concerns, or other feedback:
cbenner@ucsd.edu

HOMER

Software for motif discovery and next-gen sequencing analysis

---

Finding Enriched Motifs in Genomic Regions (*findMotifsGenome.pl*)

HOMER was initially developed to automate the process of finding enriched motifs in ChIP-Seq peaks. More generally, HOMER analyzes genomic positions, not limited to only ChIP-Seq peaks, for enriched motifs. The main idea is that all the user really needs is a file containing genomic coordinates (i.e. a HOMER peak file or BED file), and HOMER will generally take care of the rest. To analyze a peak file for motifs, run the following command:

> **findMotifsGenome.pl <peak/BED file> <genome> <output directory> [options]**
>
> i.e. **findMotifsGenome.pl ERpeaks.txt hg18r ER_MotifOutput/**

A variety of output files will be placed in the <output directory>, including html pages showing the results.

The findMotifsGenome.pl program is a wrapper that helps set up the data for analysis using the HOMER motif discovery algorithm. By default this will perform *de novo* motif discovery as well as check the enrichment of known motifs. If you have not done so already, please look over this page describing how HOMER analyzes sequences for enriched motifs.

An important prerequisite for analyzing genomic motifs is that the appropriate genome must by configured for use with HOMER. In version v3.1, HOMER now handles custom/arbitrary genomes. Instead of intalling/configuring a genome, you can specify the path to a file or directory containing the genomic sequence in FASTA format. The genome can be in a single FASTA file, or you specify a directory where where each chromosome can be in a separate file (named chr*XXX*.fa or chr*XXX*.fa.masked). In either case, the FASTA headers must contain the chromosome names followed by white space, i.e. ">chr blahblahblah", not ">chr1-blahblahblah", or prefereably only ">chr1". (also note that homer will create a "preparsed/" directory where the genome is, so make sure you have write permissions in the genomic directory.

Acceptable Input files

> **findMotifsGenome.pl** accepts HOMER peak files or BED files:
>
> HOMER peak files should have at minimum 5 columns (separated by TABs, additional columns will be ignored):
>
> - Column1: Unique Peak ID
> - Column2: chromosome
> - Column3: starting position
> - Column4: ending position
> - Column5: Strand (+/- or 0/1, where 0="+", 1="-")
>
> BED files should have at minimum 6 columns (separated by TABs, additional columns will be

ignored)

- Column1: chromosome
- Column2: starting position
- Column3: ending position
- Column4: Unique Peak ID
- Column5: not used
- Column6: Strand (+/- or 0/1, where 0="+", 1="-")

In theory, HOMER will accept BED files with only 4 columns (+/- in the 4th column), and files without unique IDs, but this is NOT recommended. For one, if you don't have unique IDs for your regions, it's hard to go back and figure out which region contains which peak.

Mac Users: If using a EXCEL to prepare input files, make sure to save files as a "Text (Windows)" if running MacOS - saving as "Tab delimited text" in Mac produces problems for the software. Otherwise, you can run the script "**changeNewLine.pl <filename>**" to convert the Mac-formatted text file to a Windows/Dos/Unix formatted text file.

If errors occur, it is likely that the file is not in the correct format, or the first column is not actually populated with unique identifiers.

Custom Background Regions

Since HOMER uses a differential motif discovery algorithm, different types of background sequences can be chosen to produce different results. For example, you may want to compare the ChIP-Seq peaks specific in one cell type versus the peaks that are specific to another. To do this, create a second peak/BED file and use it with the argument "**-bg <peak/BED file>**". HOMER will still try to normalize the background to remove GC-bias and will also perform autonormalization (see below). You can turn off the normalization with ("**-noweight**" and/or "**-nlen 0**").

How findMotifsGenome.pl works

There are a series of steps that the program goes through to find quality motifs:

1. Verify peak/BED file

   HOMER makes sure you have valid peaks, and checks to make sure you have unique peak identifiers. If there are replicates, it will inform you, and will add numbers to peak names to ensure they are unique for downstream analysis.

2. Extract sequences from the genome corresponding to the regions in the input file, filtering sequences that are >70% "N"

   This step is pretty self explanatory. If you wish to extract sequences from a genome for any reason, check out **homerTools**. HOMER will also trash sequences that are predominately "N". If you feel you are throwing away too many sequences, try running **findMotifsGenome.pl** on an unmasked genome.

3. Calculate GC/CpG content of peak sequences.

   CpG Islands are the single biggest source of sequence content bias in mammalian genomes, and are unfortunately found near transcription start sites, where all the action is! By default, HOMER tracks GC% (use "**-cpg**" to use CpG%).

4. Preparse the genomic sequences of the selected size to serve as background sequences.

   This step is only done the first time you find motifs from regions of a given size ("**-size <#>**"). HOMER takes regions near the TSS of genes (+/- 50kb) and splits them into regions of the indicated size. It then calculates their GC/CpG% and stores them for later use to speed up execution the next time you search for motifs from similar sized regions.

5. Randomly select background regions for motif discovery.

   Since HOMER is a differential motif discovery algorithm, it must use background sequence regions as a control. By default, HOMER selects enough random background regions such that the total number of regions is 50000 or 2x the total number of peaks, which ever is larger (to change use "**-N <#>**"). The more total sequence that is used, the slower the program will run, but you want to make sure there is enough background regions to reliably estimate motif frequency. HOMER attempts to select background regions that match the GC-content distribution of the input sequences (in 5% increments). For example, if your input regions are extremely GC-rich, HOMER will select random regions from GC-rich regions of the genome as a control.

   If custom background regions are provided ("**-bg <peak/BED file>**"), HOMER will automatically ensure that these regions do NOT overlap with the target regions (using **mergePeaks**). Custom regions will still be normalized for GC-content.

6. Autonormalization of sequence bias.

   Autonormalization is a unique procedure provided by HOMER that attempts to remove bias introduced by lower-order oligo sequences. It works by assuming your targets regions and background regions should not have an imbalance in 1-mers, 2-mers, 3-mers, etc. The maximum length of oligo that is autonormalized is specified by "**-nlen <#>**" (default is 3, to disable use "**-nlen 0**"). For example, there should not be significantly more A's in the target sequences that in the background. After calculating the imbalances for each oligo, it adjusts the weights of each background sequence by a small amount to help normalize any imbalance. If target sequences are rich in A, then background sequences that contain many A's will be assigned higher weights while those with very few A's will be assigned lower weights. The weights are incremented by only small amounts and the procedure repeated many times in a hill climbing optimization. This procedure helps remove some of the sequence bias associated with certain genomic regions, or bias that may have been introduced by biased experimental results such as biased sequencing.

7. Check enrichment of known motifs

   HOMER screens it's library of reliable motifs against the target and background sequences for enrichment, returning motifs enriched with a p-value less than 0.05. The known motif enrichment is performed first since it is usually faster, and gives a faster look at what's enriched in your target regions. Know motif enrichment will be reported to the "knownResults.html" file in the output directory.

8. *de novo* motif finding

   Best saved for last. By default, HOMER will search for motifs of len 8, 10, and 12 bp (change using **-len <#,#,#>** with no spaces between the numbers, i.e. **"-len 6,10,15,20"**). For a more detail description of the motif discovery algorithm, see here.

Output from the de novo motif finding will be displayed in the "homerResults.html" file.

findMotifsGenome.pl Output

A full description of motif finding output and the output can be found here.

Several files are produced in the output directory:

homerMotifs.motifs<#> : these are the output files from the de novo motif finding, separated by motif length, and represent separate runs of the algorithm.

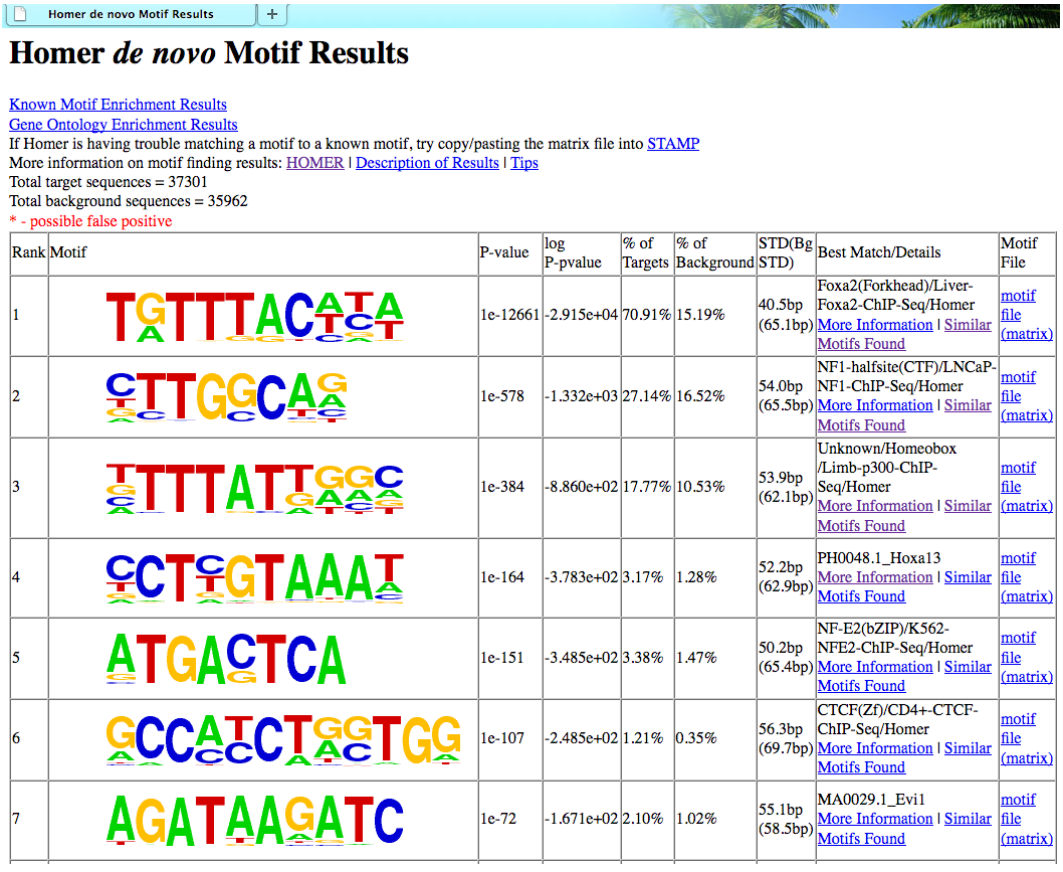
homerMotifs.all.motifs : Simply the concatenated file composed of all the homerMotifs.motifs<#> files.

motifFindingParameters.txt : command used to execute findMotifsGenome.pl

knownResults.txt : text file containing statistics about known motif enrichment (open in EXCEL).

seq.autonorm.tsv : autonormalization statistics for lower-order oligo normalization.

homerResults.html : formatted output of *de novo* motif finding.



Homer de novo Motif Results

Homer *de novo* Motif Results

Known Motif Enrichment Results
Gene Ontology Enrichment Results
If Homer is having trouble matching a motif to a known motif, try copy/pasting the matrix file into STAMP
More information on motif finding results: HOMER | Description of Results | Tips
Total target sequences = 37301
Total background sequences = 35962
* - possible false positive

Rank	Motif	P-value	log P-pvalue	% of Targets	% of Background	STD(Bg STD)	Best Match/Details	Motif File
1		1e-12661	-2.915e+04	70.91%	15.19%	40.5bp (65.1bp)	Foxa2(Forkhead)/Liver-Foxa2-ChIP-Seq/Homer More Information | Similar Motifs Found	motif file (matrix)
2		1e-578	-1.332e+03	27.14%	16.52%	54.0bp (65.5bp)	NF1-halfsite(CTF)/LNCaP-NF1-ChIP-Seq/Homer More Information | Similar Motifs Found	motif file (matrix)
3		1e-384	-8.860e+02	17.77%	10.53%	53.9bp (62.1bp)	Unknown/Homeobox /Limb-p300-ChIP-Seq/Homer More Information | Similar Motifs Found	motif file (matrix)
4		1e-164	-3.783e+02	3.17%	1.28%	52.2bp (62.9bp)	PH0048.1_Hoxa13 More Information | Similar Motifs Found	motif file (matrix)
5		1e-151	-3.485e+02	3.38%	1.47%	50.2bp (65.4bp)	NF-E2(bZIP)/K562-NFE2-ChIP-Seq/Homer More Information | Similar Motifs Found	motif file (matrix)
6		1e-107	-2.485e+02	1.21%	0.35%	56.3bp (69.7bp)	CTCF(Zf)/CD4+-CTCF-ChIP-Seq/Homer More Information | Similar Motifs Found	motif file (matrix)
7		1e-72	-1.671e+02	2.10%	1.02%	55.1bp (58.5bp)	MA0029.1_Evi1 More Information | Similar Motifs Found	motif file (matrix)

homerResults/ directory: contains files for the homerResults.html webpage, including motif<#>.motif files for use in finding specific instance of each motif.

knownResults.html : formatted output of known motif finding.

knownResults/ directory: contains files for the knownResults.html webpage, including known<#>.motif files for use in finding specific instance of each motif.

Interpreting motif finding results

The format of the output files generated by **findMotifsGenome.pl** are identical to those generated by the promoter-based version **findMotifs.pl** (description).

In general, when analyzing ChIP-Seq / ChIP-Chip peaks you should expect to see strong enrichment for a motif resembling the site recognized by the DNA binding domain of the factor you are studying. Enrichment p-values reported by HOMER should be very very significant (i.e. << 1e-50). If this is not the case, there is a strong possibility that the experiment may have failed in one way or another. For example, the peaks could be of low quality because the factor is not expressed very high.

Practical Tips for Motif finding

Important motif finding parameters

Masked vs. Unmasked Genome (i.e. hg18 vs. hg18r)

Actually, this usually doesn't matter *that* much. Since HOMER is a differential motif discovery algorithm, common repeats are usually in both the target and background sequences. However, it is not uncommon that a transcription factor binds to a certain class of repeats, which may cause several large stretches of similar sequence to be processed, biasing the results. Usually it's safer to go with the masked version.

Region Size ("**-size <#>**", default: 200)

The size of the region used for motif finding is important. If analyzing ChIP-Seq peaks from a transcription factor, Chuck would recommend 50 bp for establishing the primary motif bound by a given transcription factor and 200 bp for finding both primary and "co-enriched" motifs for a transcription factor. When looking at histone marked regions, 500-1000 bp is probably a good idea (i.e. H3K4me or H3/H4 acetylated regions). In theory, HOMER can work with very large regions (i.e. 10kb), but with the larger the regions comes more sequence and longer execution time.

Motif length ("**-len <#>**" or "**-len <#>,<#>,...**", default 8,10,12)

Specifies the length of motifs to be found. HOMER will find motifs of each size separately and then combine the results at the end. The length of time it takes to find motifs increases greatly with increasing size. In general, it's best to try out enrichment with shorter lengths (i.e. less than 15) before trying longer lengths. Much longer motifs can be found with HOMER, but it's best to use smaller sets of sequence when trying to find long motifs (i.e. use "-len 20 -size 50"), otherwise it may take way too long (or take too much memory). The other trick to reduce the total resource consumption is to reduce the number of background sequences (-N <#>).

Mismatches allowed in global optimization phase ("**-mis <#>**", default: 2)

HOMER looks for promising candidates by initially checking ordinary oligos for enrichment, allowing mismatches. The more mismatches you allow, the more sensitive the algorithm, particularly for longer motifs. However, this also slows down the algorithm a bit. If searching for motifs longer than 12-15 bp, it's best to increase this value to at least 3 or even 4.

Number of CPUs to use ("**-p <#>**", default 1)

HOMER is now multicore compliant. It's not perfectly parallelized, however, certain types of analysis can benefit. In general, the longer the length of the motif, the better the speed-up you'll see.

Number of motifs to find ("**-S <#>**", default 25)

Specifies the number of motifs of each length to find. 25 is already quite a bit. If anything, I'd recommend reducing this number, particularly for long motifs to reduce the total execution time.

Normalize CpG% content instead of GC% content ("**-cpg**")

Consider tying if HOMER is stuck finding "CGCGCGCG"-like motifs. You can also play around with disabling GC/CpG normalization ("**-noweight**").

Region level autonormalization (**"-nlen <#>**", default 3, "-nlen 0" to disable)
Motif level autonormalization (**-olen <#>**, default 0 i.e. disabled)

Autonormalization attempts to remove sequence bias from lower order oligos (1-mers, 2-mers ... up to <#>). Region level autonormalization, which is for 1/2/3 mers by default, attempts to normalize background regions by adjusting their weights. If this isn't getting the job done (autonormalization is not guaranteed to remove all sequence bias), you can try the more aggressive motif level autonormalization (**-olen <#>**). This performs the autonormalization routine on the oligo table during de novo motif discovery. (see here for more info)

User defined background regions ("**-bg <peak file of background regions>**")

Why let HOMER randomly pick you background regions when you can choose them yourself!! These will still be normalized for CpG% or GC% content just like randomly chosen sequences and autonormalized unless these options are turned off (i.e. "-nlen 0 -noweight"). This can be very useful since HOMER is a differential motif discovery algorithm. For example, you can give HOMER a set of peaks co-bound by another factor and compare them to the rest of the peaks. HOMER will automatically check if the background peaks overlap with the target peaks using **mergePeaks**, and discard overlapping regions.

Hypergeometric enrichment scoring ("**-h**")

By default, **findMotifsGenome.pl** uses the binomial distribution to score motifs. This works well when the number of background sequences greatly out number the target sequences - however, if you are using "**-bg**" option above, and the number of background sequences is smaller than target sequences, it is a good idea to use the hypergeometric distribution instead ("**-h**"). FYI - The binomial is faster to compute, hence it's use for motif finding in large numbers of regions.

Find enrichment of individual oligos ("**-oligo**").

This creates output files in the output directory named oligo.*length*.txt.

Force **findMotifsGenome.pl** to re-preparse genome for the given region size ("**-preparse**").

In case there is a problem with the existing preparsed files, force them to be remade with "-preparse".

Only search for motifs on + strand ("**-norevopp**")

By default, HOMER looks for transcription factor-like motifs on both strands. This will force it to only look at the + strand (relative to the peak, so - strand if the peak is on the - strand).

Search for RNA motifs ("**-rna**")

If looking at RNA data (i.e. Clip-Seq or similar), this option will restrict HOMER to only search the + strand (relative to the peak), and will output RNA motif logos (i.e. U instead of T). It will also try to compare found motifs to an RNA motif database, which sadly, only contains miRNAs right now... I guess chuck roundhouse kicked all of the splicing and other RNA motifs into hard to find databases.

Mask motifs ("**-mask <motif file>**")

Mask the motif(s) in the supplied motif file before starting motif finding. Multiple motifs can be in the motif file.

Optimize motifs ("**-opt <motif file>**")

Instead of looking for novel de novo motifs, HOMER will instead try to optimize the motif supplied. This is cool when trying to change the length of a motif, or find a very long version of a given motif. For example, if you specify "-opt <file>" and "-len 50", it will try to expand the motif to 50bp and optimize it.

Dump FASTA files ("**-dumpFasta**")

Like the fact that HOMER organizes and extracts your sequence files, but don't care for HOMER as a motif finding algorithm? That's cool, just specify "-dumpFasta" and the files "target.fa" and "background.fa" will show up in your output directory. You can then use them with MEME or whatever. Just remember, Chuck knows where you live...

Finding Instance of Specific Motifs

By default, HOMER does not return the locations of each motif found in the motif discovery process. To recover the motif locations, you must first select the motifs you're interested in by getting the "motif file" output by HOMER. You can combine multiple motifs in single file if you like to form a "motif library". To identify motif locations, you have two options:

1. Run **findMotifsGenome.pl** with the "**-find <motif file>**" option. This will output a tab-delimited text file with each line containing an instance of the motif in the target peaks. The output is sent to *stdout*.

For example: **findMotifsGenome.pl ERalpha.peaks hg18 MotifOutputDirectory/ -find motif1.motif > outputfile.txt**

The output file will contain the columns:

1. Peak/Region ID
2. Offset from the center of the region
3. Sequence of the site
4. Name of the Motif
5. Strand
6. Motif Score (log odds score of the motif matrix, higher scores are better matches)

2. Run **annotatePeaks.pl** with the "**-m <motif file>**" option (see the annotation section for more info). Chuck prefers doing it this way. This will output a tab-delimited text file with each line containing a peak/region and a column containing instance of each motif separated by commas to stdout

For example: **annotatePeaks.pl ERalpha.peaks hg18 -m motif1.motif > outputfile.txt**

The output file will contain columns:

1. Peak/Region ID
2. Chromosome
3. Start

4. End
5. Strand of Peaks

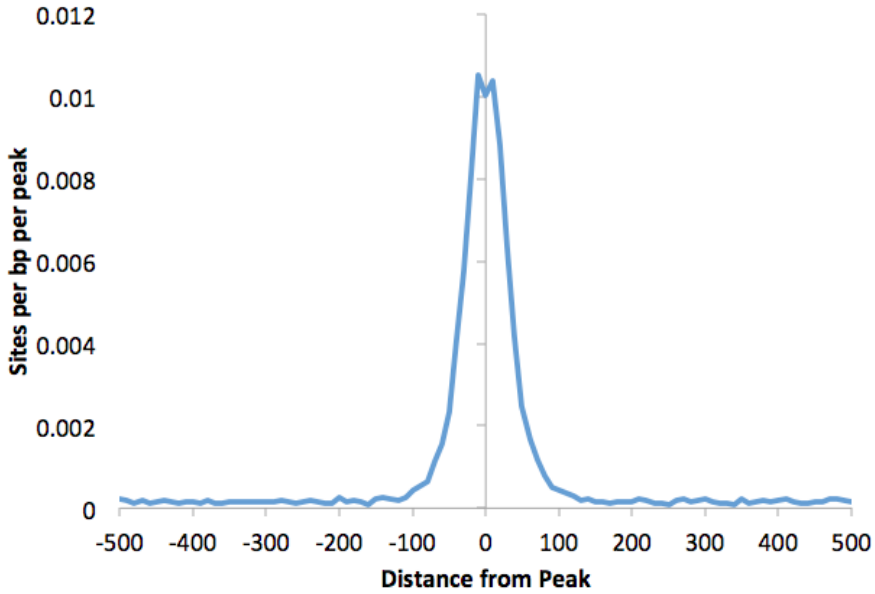
6-18: annotation information
19. CpG%
20. GC%
21. Motif Instances
...

Motif Instances have the following format:
<distance from center of region>(<sequence>,<strand>,<conservation>)
i.e -29(TAAATCAACA,+,0.00)

You can also find histogram of motif density this way by adding "-hist <#>" to the command. For example:

annotatePeaks.pl ERalpha.peaks hg18 -m ere.motif foxa1.motif -size 1000 -hist 10 > outputfile.txt

Graphing the output with EXCEL:



Command-line options for findMotifsGenome.pl

Program will find de novo and known motifs in regions in the genome

Usage: findMotifsGenome.pl <pos file> <genome> <output directory> [additional options]
Example: findMotifsGenome.pl peaks.txt mm8r peakAnalysis -size 200 -len 8

Possible Genomes:
...
Custom: provide the path to genome FASTA files (directory or single file)
Heads up: will create the directory "preparsed/" in same location.

Basic options:

-bg <background position file> (genomic positions to be used as background, default=automatic)
removes background positions overlapping with target positions
-chopify (chop up large background regions to the avg size of target regions)
-len <#>[,<#>,<#>...] (motif length, default=8,10,12) [NOTE: values greater 12 may cause the program
to run out of memory - in these cases decrease the number of sequences analyzed (-N), or try analyzing shorter sequence regions (i.e. -size 100)]
-size <#> (fragment size to use for motif finding, default=200)
-size <#,#> (i.e. -size -100,50 will get sequences from -100 to +50 relative from center)
-size given (uses the exact regions you give it)
-S <#> (Number of motifs to optimize, default: 25)
-mis <#> (global optimization: searches for strings with # mismatches, default: 2)
-norevopp (don't search reverse strand for motifs)
-nomotif (don't search for de novo motif enrichment)
-rna (output RNA motif logos and compare to RNA motif database, automatically sets -norevopp)

Scanning sequence for motifs
-find <motif file> (This will cause the program to only scan for motifs)

Known Motif Options/Visualization
-bits (scale sequence logos by information content, default: doesn't scale)
-nocheck (don't search for de novo vs. known motif similarity)
-mcheck <motif file> (known motifs to check against de novo motifs,
default: /bioinformatics/homer/data/knownTFs/all.motifs
-float (allow adjustment of the degeneracy threshold for known motifs to improve p-value[dangerous])
-noknown (don't search for known motif enrichment, default: -known)
-mknown <motif file> (known motifs to check for enrichment,
default: /bioinformatics/homer/data/knownTFs/known.motifs

Sequence normalization options:
-gc (use GC% for sequence content normalization, now the default)
-cpg (use CpG% instead of GC% for sequence content normalization)
-noweight (no CG correction)

Advanced options:
-h (use hypergeometric for p-values, binomial is default)
-N <#> (Number of sequences to use for motif finding, default=max(50k, 2x input)
-noforce (will attempt to reuse sequence files etc. that are already in output directory)
-local <#> (use local background, # of equal size regions around peaks to use i.e. 2)
-redundant <#> (Remove redundant sequences matching greater than # percent, i.e. -redundant 0.5)
-mask <motif file1> [motif file 2]... (motifs to mask before motif finding)
-opt <motif file1> [motif file 2]... (motifs to optimize or change length of)
-refine <motif file1> (motif to optimize)
-rand (randomize target and background sequences labels)
-ref <peak file> (use file for target and background - first argument is list of peak ids for targets)
-oligo (perform analysis of individual oligo enrichment)
-dumpFasta (Dump fasta files for target and background sequences for use with other programs)
-preparse (force new background files to be created)
-keepFiles (keep temporary files)

homer2 specific options:
-homer2 (use homer2 instead of original homer, default)
-nlen <#> (length of lower-order oligos to normalize in background, default: -nlen 3)
-nmax <#> (Max normalization iterations, default: 160)
-olen <#> (lower-order oligo normalization for oligo table, use if -nlen isn't working well)
-p <#> (Number of processors to use, default: 1)
-e <#> (Maximum expected motif instance per bp in random sequence, default: 0.01)
-cache <#> (size in MB for statistics cache, default: 500)
-quickMask (skip full masking after finding motifs, similar to original homer)

Original homer specific options:
-homer1 (to force the use of the original homer)
-depth [low|med|high|allnight] (time spent on local optimization default: med)

Can't figure something out? Questions, comments, concerns, or other feedback:
cbenner@ucsd.edu

HOMER

Software for motif discovery and next-gen sequencing analysis

---

Annotating Regions in the Genome (*annotatePeaks.pl*)

Homer contains a useful, all-in-one program for performing peak annotation called **annotatePeaks.pl**. In addition to associating peaks with nearby genes, **annotatePeaks.pl** can perform Gene Ontology Analysis, genomic feature association analysis (Genome Ontology), associate peaks with gene expression data, calculate ChIP-Seq Tag densities from different experiments, and find motif occurrences in peaks. **annotatePeaks.pl** can also be used to create histograms and heatmaps. Description of the annotation functions are covered below, while quantification of tags, motifs, histograms, etc. are covered here.

NOTE: If you're running annotatePeaks.pl on your laptop, you may want to use "**-noann**" to skip the full annotation routines, which use a bit of memory (up to 4Gb)

Basic usage:

annotatePeaks.pl <peak/BED file> <genome> > <output file>

i.e. **annotatePeaks.pl ERpeaks.txt hg18 > outputfile.txt**

The first two arguments, the <peak file> and <genome>, are required, and must be the first two arguments. Other optional command line arguments can be placed in any order after the first two. By default, **annotatePeaks.pl** prints the program output to *stdout*, which can be captured in a file by appending " > filename" to the command. With most uses of **annotatePeaks.pl**, the output is a data table that is meant to be opened with EXCEL or similar program. An example of the output can been seen below:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	PeakID	Chr	Start	End	Strand	Peak Sco	Focus Ra	Annotation	Detailed Anno	Distance to T	Nearest Pror	PromoterID	Nearest Unig	Nearest Refs	Nearest Ense	Gene Name	Gene Alias	Gene Descrip
2	chr18-1	chr18	69007968	69008268	+	593	0.939	intron (NR_03	intron (NR_03	74595	NR_034133	400655	Hs.579378	NR_034133		LOC400655	-	hypothetical
3	chr9-1	chr9	88209966	88210266	+	531.9	0.946	Intergenic	Intergenic	-50894	NM_001185	79670	Hs.597057	NM_001185	ENSG000000	ZCCHC6	DKFZp666B1	zinc finger, C
4	chr14-1	chr14	62337073	62337373	+	505.4	0.918	intron (NM_17	intron (NM_17	244485	NM_172375	27133	Hs.27043	NM_139318	ENSG000001	KCNH5	EAG2|H-EAG	potassium vo
5	chr17-1	chr17	5076243	5076543	+	492.1	0.936	intron (NR_03	intron (NR_03	2414	NM_207103	388325	Hs.462080	NM_207103	ENSG000001	C17orf87	FLJ32580|M	chromosome
6	chr17-2	chr17	47851714	47852014	+	476.2	0.824	Intergenic	Intergenic	-259488	NM_001082	56934	Hs.463466	NM_001082	ENSG000001	CA10	CA-RPX|CAR	carbonic anh
7	chr10-1	chr10	98420680	98420980	+	474.9	0.967	intron (NM_15	intron (NM_15	49439	NM_152309	118788	Hs.310456	NM_152309	ENSG000001	PIK3AP1	BCAP|RP11-	phosphoinos
8	chr9-2	chr9	81294389	81294689	+	456.3	0.957	Intergenic	Intergenic	-82159	NM_007005	7091	Hs.444213	NM_007005	ENSG000001	TLE4	BCE-1|BCE1	transducin-li
9	chr14-2	chr14	36817736	36818036	+	452.3	0.757	intron (NM_13	intron (NM_13	81017	NM_001195	145282	Hs.660396	NM_001195	ENSG000001	MIPOL1	DKFZp313M	mirror-image
10	chr18-2	chr18	20049825	20050125	+	449.7	0.853	intron (NM_08	intron (NM_08	56219	NM_018030	114876	Hs.370725	NM_018030	ENSG000001	OSBPL1A	FLJ10217|OF	oxysterol bin
11	chr7-1	chr7	12226829	12227129	+	445.7	0.901	intron (NM_01	intron (NM_01	9606	NM_001134	54664	Hs.396358	NM_001134	ENSG000001	TMEM106B	FLJ11273|M	transmembr
12	chr14-3	chr14	88712188	88712488	+	443.1	0.844	intron (NM_00	intron (NM_00	240869	NM_005197	1112	Hs.621371	NM_001085	ENSG000000	FOXN3	C14orf116|C	forkhead box
13	chr18-3	chr18	62951924	62952224	+	443.1	0.947	Intergenic	Intergenic	-382689	NR_033921	643542	Hs.652901	NR_033921		LOC643542	-	hypothetical
14	chr3-1	chr3	32196769	32197069	+	443.1	0.87	Intergenic	Intergenic	-58256	NM_178868	152189	Hs.154986	NM_178868	ENSG000001	CMTM8	CKLFSF8|CKL	CKLF-like MA
15	chr11-1	chr11	110685448	110685748	+	425.8	0.907	Intergenic	Intergenic	-9849	NR_034154	399948	Hs.729225	NR_034154		C11orf92	DKFZp781P1	chromosome
16	chr4-1	chr4	81755366	81755666	+	423.2	0.908	intron (NM_15	intron (NM_15	279618	NM_152770	255119	Hs.527104	NM_152770	ENSG000001	C4orf22	MGC35043	chromosome

Acceptable Input files

annotatePeaks.pl accepts HOMER peak files or BED files:

HOMER peak files should have at minimum 5 columns (separated by TABs, additional columns will be ignored):

- Column1: Unique Peak ID
- Column2: chromosome
- Column3: starting position
- Column4: ending position
- Column5: Strand (+/- or 0/1, where 0="+", 1="-")

BED files should have at minimum 6 columns (separated by TABs, additional columns will be ignored)

- Column1: chromosome
- Column2: starting position
- Column3: ending position
- Column4: Unique Peak ID
- Column5: not used

- Column6: Strand (+/- or 0/1, where 0="+", 1="-")

In theory, HOMER will accept BED files with only 4 columns (+/- in the 4th column), and files without unique IDs, but this is NOT recommended.

Mac Users: If using a EXCEL to prepare input files, make sure to save files as a "Text (Windows)" if running MacOS - saving as "Tab delimited text" in Mac produces problems for the software. Otherwise, you can run the script "**changeNewLine.pl <filename>**" to convert the Mac-formatted text file to a Windows/Dos/Unix formatted text file.

If errors occur, it is likely that the file is not in the correct format, or the first column is not actually populated with unique identifiers.

How Basic Annotation Works

The process of annotating peaks/regions is divided into two primary parts. The first determines the distance to the nearest TSS and assigns the peak to that gene. The second determines the genomic annotation of the region occupied by the center of the peak/region.

Distance to the nearest TSS

By default, annotatePeaks.pl loads a file in the "/*path-to-homer*/data/genomes/<*genome*>/<*genome*>.tss" that contains the positions of RefSeq transcription start sites. It uses these positions to determine the closest TSS, reporting the distance (negative values mean upstream of the TSS, positive values mean downstream), and various annotation information linked to locus including alternative identifiers (unigene, entrez gene, ensembl, gene symbol etc.). This information is also used to link gene-specific information (see below) to a peak/region, such as gene expression.

Genomic Annotation

To annotate the location of a given peak in terms of important genomic features, **annotatePeaks.pl** calls a separate program (**assignGenomeAnnotation**) to efficiently assign peaks to one of millions of possible annotations genome wide. Two types of output are provided. The first is "Basic Annotation" that includes whether a peak is in the TSS (transcription start site), TTS (transcription termination site), Exon (Coding), 5' UTR Exon, 3' UTR Exon, Intronic, or Intergenic, which are common annotations that many researchers are interested in. A second round of "Detailed Annotation" also includes more detailed annotation, also considering repeat elements and CpG islands. Since some annotation overlap, a priority is assign based on the following (in case of ties it's random [i.e. if there are two overlapping repeat element annotations]):

1. TSS (by default defined from -1kb to +100bp)
2. TTS (by default defined from -100 bp to +1kb)
3. CDS Exons
4. 5' UTR Exons
5. 3' UTR Exons
6. **CpG Islands
7. **Repeats
8. Introns
9. Intergenic

** Only applicable for the "Detailed Annotation".

Although HOMER doesn't allow you to explicitly change the definition of the region that is the TSS (-1kb to +100bp), you can "do it yourself" by sorting the annotation output in EXCEL by the "Distance to nearest TSS" column, and selecting those within the range you are interested in.

Using Custom Annotations

Custom Gene Annotation Definition using GTF files

RefSeq doesn't do it for everyone. There are many other quality gene annotations out there, including UCSC genes, Ensembl, and Gencode to name a couple. Even more important, as RNA-Seq methods develop, the locations of exons etc. can be defined based on your own experimental RNA data rather than using a static database to define transcripts. You can download GTF files for various gene definitions from the UCSC Genome Browser in the "Table Browser" section. Custom GTF files can be created from RNA-Seq data using

tools like Cufflinks.

HOMER can process GTF (Gene Transfer Format) files and use them for annotation purposes ("**-gtf <gtf filename>**"). If a GTF file is specified, HOMER will parse it and use the TSS from the GTF file for determining the distance to the nearest TSS. It will also use the GTF file's definition of TSS/TTS/exons/Introns for Basic Genome Annotation. The original HOMER annotation files are still used for the "Detailed Annotation" since the repeats will not be define in the GTF files.

i.e. **annotatePeaks.pl ERpeaks.txt hg18 -gtf gencode.gtf > outputfile.txt**

HOMER will try it's best to take the "transcript_id" from the GTF definition and translate it into a known gene identifier. If it can't match it to a known gene, many of the annotation columns corresponding to Unigene etc. will be empty.

HOMER can also work with GFF and GFF3 files - to a degree. The problem with these is that the formats do not have the same strict naming convention enforced in GTF files. To use them, substitute "**-gtf <GTF file>**" with "**-gff <GFF file>**" or "**-gff3 <GFF3 file>**"

Custom Promoter Locations

By default, **annotatePeaks.pl** assigns peaks to the nearest TSS. If you have a custom peak/pos file of these locations, you can supply that with the option "**-cTSS <peak/pos file>**". This will override the default, which is RefSeq TSS.

Using Custom Genomes/Annotations

For organisms with relatively incomplete genomes, annotatePeaks.pl can still provide some functionality. If the genome is not available as a pre-configured genome in HOMER, then you can supply the path to the full genome FASTA file or path to directory containing chromsome FASTA files as the 2nd argument. For example, lets say you were sequencing the banana slug genome, and had downloaded the current draft of the genome into the file "bananaSlug.fa". You could then run annotatePeaks.pl like this:

annotatePeaks.pl chip-seq-peaks.txt bananaSlug.fa > output.txt

If no genome sequence is available, you can also specify "**none**":

annotatePeaks.pl chip-seq-peaks.txt none > output.txt

You may also find a custom annotation file for the organism, such as banana_slug_genes.gtf, or banana_slug_genes.gff from the community website. This can then be used to help annotate your data:

annotatePeaks.pl chip-seq-peaks.txt bananaSlug.fa -gtf banana_slug_genes.gtf > output.txt
or
annotatePeaks.pl chip-seq-peaks.txt bananaSlug.fa -gff banana_slug_genes.gff > output.txt

In the case of custom genomes, HOMER will not be able to convert gene IDs to names - you may have to do this yourself, but many of the other features, including motif finding etc., are available as long as you have the sequence.

Basic Annotation File Output

Description of Columns:

1. Peak ID
2. Chromosome
3. Peak start position
4. Peak end position
5. Strand
6. Peak Score
7. FDR/Peak Focus Ratio/Region Size
8. Annotation (i.e. Exon, Intron, ...)
9. Detailed Annotation (Exon, Intron etc. + CpG Islands, repeats, etc.)

10. Distance to nearest RefSeq TSS
11. Nearest TSS: Native ID of annotation file
12. Nearest TSS: Entrez Gene ID
13. Nearest TSS: Unigene ID
14. Nearest TSS: RefSeq ID
15. Nearest TSS: Ensembl ID
16. Nearest TSS: Gene Symbol
17. Nearest TSS: Gene Aliases
18. Nearest TSS: Gene description
19. *Additional columns depend on options selected when running the program.*

As of now, basic annotation is based on alignments of RefSeq transcripts to the UCSC hosted genomes.

Adding Gene Expression Data

annotatePeaks.pl can add gene-specific information to peaks based on each peak's nearest annotated TSS. To add gene expression or other data types, first create a gene data file (tab delimited text file) where the first column contains gene identifiers, and the first row is a header describing the contents of each column. In principle, the contents of these columns doesn't mater. To add this information to the annotation result, use the "**-gene <gene data file>**".

annotation.pl <peak file> <genome> -gene <gene data file> > output.txt

For peaks that are near genes with associated data in the "gene data file", this data will be appended to the end of the row for each peak.

Peak Annotation Enrichment

Gene Ontology Analysis of Associated Genes

annotatePeaks.pl offers two types of annotation enrichment analysis. The first is based on Gene Ontology classifications for genes. The idea is that your regions (i.e. ChIP-Seq peaks) might be preferentially found near genes with specific biological functions. By specifying "**-go <GO output directory>**", HOMER will take the list of genes associated with your regions and search for enriched functional categories, placing the output in the indicated directory. This is just like looking for GO enrichment in a set of regulated genes (covered in greater detail here).

There are some caveats to this type of analysis - for example, some genes are simply more likely to be considered in the analysis if there are isolated along the genome where they are likely to be the "closest gene" for peaks in the region. But... since gene density roughly correlates with transcription factor density, it works itself out to some degree. Other tools, such as GREAT attempt to deal with this problem to some degree, and are worth trying. My experience is that the results are not that different once you factor in the fact that most other tools use "GO slims".

Genome Ontology: Looking for Enriched Genomic Annotations

The **Genome Ontology** looks for enrichment of various genomic annotations in your list of peaks/regions. Just as the Gene Ontology contains groups of genes sharing a biological function, the Genome Ontology contains groups of genomic regions sharing an annotation, such as TSS, or LINE repeats, coding exons, Transcription factor peaks, etc. To run the Genome Ontology, add the option "**-genomeOntology <output directory>**". A much more detailed description of the Genome Ontology will be available here (coming soon).

Command line options for annotatePeaks.pl

Usage: annotatePeaks.pl <peak file | tss> <genome version> [additional options...]

Available Genomes (required argument): (name,org,directory,default promoter set)
-- or --

```
   Custom: provide the path to genome FASTA files (directory or single file)

User defined annotation files (default is UCSC refGene annotation):
   annotatePeaks.pl accepts GTF (gene transfer formatted) files to annotate positions relative
   to custom annotations, such as those from de novo transcript discovery or Gencode.
   -gtf <gtf format file> (-gff and -gff3 can work for those files, but GTF is better)

Peak vs. tss/tts/rna mode (works with custom GTF file):
   If the first argument is "tss" (i.e. annotatePeaks.pl tss hg18 ...) then a TSS centric
   analysis will be carried out.  Tag counts and motifs will be found relative to the TSS.
   (no position file needed) ["tts" now works too - e.g. 3' end of gene]
   ["rna" specifies gene bodies, will automaticall set "-size given"]
   NOTE: The default TSS peak size is 4000 bp, i.e. +/- 2kb (change with -size option)
   -list <gene id list> (subset of genes to perform analysis [unigene, gene id, accession,
      probe, etc.], default = all promoters)
   -cTSS <promoter position file i.e. peak file> (should be centered on TSS)

Primary Annotation Options:
   -m <motif file 1> [motif file 2] ... (list of motifs to find in peaks)
      -mscore (reports the highest log-odds score within the peak)
      -nmotifs (reports the number of motifs per peak)
      -mdist (reports distance to closest motif)
      -mfasta <filename> (reports sites in a fasta file - for building new motifs)
      -fm <motif file 1> [motif file 2] (list of motifs to filter from above)
      -rmrevopp <#> (only count sites found within <#> on both strands once, i.e. palindromic)
      -matrix <prefix> (outputs a motif co-occurrence files:
         prefix.count.matrix.txt - number of peaks with motif co-occurrence
         prefix.ratio.matrix.txt - ratio of observed vs. expected  co-occurrence
         prefix.logPvalue.matrix.txt - co-occurrence enrichment
         prefix.stats.txt - table of pair-wise motif co-occurrence statistics
         additional options:
         -matrixMinDist <#> (minimum distance between motif pairs - to avoid overlap)
         -matrixMaxDist <#> (maximum distance between motif pairs)
      -mbed <filename> (Output motif positions to a BED file to load at UCSC (or -mpeak))
   -d <tag directory 1> [tag directory 2] ... (list of experiment directories to show
      tag counts for) NOTE: -dfile <file> where file is a list of directories in first column
   -bedGraph <bedGraph file 1> [bedGraph file 2] ... (read coverage counts from bedGraph files)
   -p <peak file> [peak file 2] ... (to find nearest peaks)
      -pdist to report only distance (-pdist2 gives directional distance)
      -pcount to report number of peaks within region
   -vcf <VCF file> (annotate peaks with genetic variation infomation, one col per individual)
      -editDistance (Computes the # bp changes relative to reference)
      -individuals <name1> [name2] ... (restrict analysis to these individuals)
   -gene <data file> ... (Adds additional data to result based on the closest gene.
      This is useful for adding gene expression data.  The file must have a header,
      and the first column must be a GeneID, Accession number, etc.  If the peak
      cannot be mapped to data in the file then the entry will be left empty.
   -go <output directory> (perform GO analysis using genes near peaks)
   -genomeOntology <output directory> (perform genomeOntology analysis on peaks)
      -gsize <#> (Genome size for genomeOntology analysis, default: 2e9)

Annotation vs. Histogram mode:
   -hist <bin size in bp> (i.e 1, 2, 5, 10, 20, 50, 100 etc.)
   The -hist option can be used to generate histograms of position dependent features relative
   to the center of peaks.  This is primarily meant to be used with -d and -m options to map
   distribution of motifs and ChIP-Seq tags.  For ChIP-Seq peaks for a Transcription factor
   you might want to use the -center option (below) to center peaks on the known motif
   ** If using "-size given", histogram will be scaled to each region (i.e. 0-100%), with
   the -hist parameter being the number of bins to divide each region into.
      Histogram Mode specific Options:
      -nuc (calculated mononucleotide frequencies at each position,
         Will report by default if extracting sequence for other purposes like motifs)
      -di (calculated dinucleotide frequencies at each position)
      -histNorm <#> (normalize the total tag count for each region to 1, where <#> is the
         minimum tag total per region - use to avoid tag spikes from low coverage
      -ghist (outputs profiles for each gene, for peak shape clustering)
```

-rm <#> (remove occurrences of same motif that occur within # bp)

Peak Centering: (other options are ignored)
-center <motif file> (This will re-center peaks on the specified motif, or remove peak if there is no motif in the peak. ONLY recentering will be performed, and all other options will be ignored. This will output a new peak file that can then be reanalyzed to reveal fine-grain structure in peaks (It is advised to use -size < 200) with this to keep peaks from moving too far (-mirror flips the position)
-multi (returns genomic positions of all sites instead of just the closest to center)

Advanced Options:
-len <#> / -fragLength <#> (Fragment length, default=auto, might want to set to 0 for RNA)
-size <#> (Peak size[from center of peak], default=inferred from peak file)
-size #,# (i.e. -size -10,50 count tags from -10 bp to +50 bp from center)
-size "given" (count tags etc. using the actual regions - for variable length regions)
-log (output tag counts as log2(x+1+rand) values - for scatter plots)
-sqrt (output tag counts as sqrt(x+rand) values - for scatter plots)
-strand <+|-|both> (Count tags on specific strands relative to peak, default: both)
-pc <#> (maximum number of tags to count per bp, default=0 [no maximum])
-cons (Retrieve conservation information for peaks/sites)
-CpG (Calculate CpG/GC content)
-ratio (process tag values as ratios - i.e. chip-seq, or mCpG/CpG)
-nfr (report nucleosome free region scores instead of tag counts, also -nfrSize <#>)
-norevopp (do not search for motifs on the opposite strand [works with -center too])
-noadj (do not adjust the tag counts based on total tags sequenced)
-norm <#> (normalize tags to this tag count, default=1e7, 0=average tag count in all directories)
-pdist (only report distance to nearest peak using -p, not peak name)
-map <mapping file> (mapping between peak IDs and promoter IDs, overrides closest assignment)
-noann, -nogene (skip genome annotation step, skip TSS annotation)
-homer1/-homer2 (by default, the new version of homer [-homer2] is used for finding motifs)

Can't figure something out? Questions, comments, concerns, or other feedback:
cbenner@ucsd.edu

HOMER

Software for motif discovery and next-gen sequencing analysis

---

Quantifying Data and Motifs and Comparing Peaks/Regions in the Genome

Homer contains a useful, all-in-one program for performing peak annotation called **annotatePeaks.pl**. In addition to associating peaks with nearby genes, **annotatePeaks.pl** can perform Gene Ontology Analysis, genomic feature association analysis (Genome Ontology), associate peaks with gene expression data, calculate ChIP-Seq Tag densities from different experiments, and find motif occurrences in peaks. **annotatePeaks.pl** can also be used to create histograms and heatmaps. Description of the annotation functions are covered here, while quantification of tags, motifs, histograms, etc. are covered below.

Basic usage (see Annotation):

annotatePeaks.pl <peak/BED file> <genome> [options] > <output file>

i.e. **annotatePeaks.pl ERpeaks.txt hg18 > outputfile.txt**

Everything packed into one program

The great thing about annotatePeaks.pl is that it combines many features into a single location. It is the primary program to investigate how sequencing reads, sequence motifs, and other information and annotations interact.

Three primary options are available to specify types of data that can be processed by annotatePeaks.pl:

-d <tag directory1> [tag directory 2] ...
-m <motif file 1> [motif file 2] ...
-p <peak/BED file 1> [peak/BED file 2] ...

By default, these data types are processed relative to each peak/region provided in the primary input file. There are a bunch of options that help fine tune how each type of data is considered by the program covered below.

However, annotatePeaks.pl can take the same input data and do other things, such as make histograms and heatmaps, allowing you to explore the data in a different way.

-hist <# bin size>

Acceptable Input files

annotatePeaks.pl accepts HOMER peak files or BED files:

HOMER peak files should have at minimum 5 columns (separated by TABs, additional columns will be ignored):

- Column1: Unique Peak ID
- Column2: chromosome
- Column3: starting position
- Column4: ending position
- Column5: Strand (+/- or 0/1, where 0="+", 1="-")

BED files should have at minimum 6 columns (separated by TABs, additional columns will be ignored)

- Column1: chromosome
- Column2: starting position
- Column3: ending position
- Column4: Unique Peak ID
- Column5: not used
- Column6: Strand (+/- or 0/1, where 0="+", 1="-")

In theory, HOMER will accept BED files with only 4 columns (+/- in the 4th column), and files without unique IDs, but this is NOT recommended. For one, if you don't have unique IDs for your regions, it's hard to go back and figure out which region contains which peak.

Mac Users: If using a EXCEL to prepare input files, make sure to save files as a "Text (Windows)" if running MacOS - saving as "Tab delimited text" in Mac produces problems for the software. Otherwise, you can run the script "**changeNewLine.pl <filename>**" to convert the Mac-formatted text file to a Windows/Dos/Unix formatted text file.

If errors occur, it is likely that the file is not in the correct format, or the first column is not actually populated with unique identifiers.

TSS Mode

Instead of supplying a peak file, HOMER makes it easy to do analysis of features near TSS. To analyze transcription start sites as if they were peaks, use "**tss**" as the first argument.

annotatePeaks.pl tss hg18 ...

To restrict the analysis to a subset of TSS promoters, add the option "**-list <file>**" where the file is a Tab-delimited text file with the first column containing gene Identifiers. TSS mode also works with custom gene definitions specified with "**-gtf <GTF file>**".

Specifying the Peak Size - the most important parameter

Pretty much everything explained in the following sections depends heavily on the "**-size <#>**" parameter. A couple of quick notes:

-size <#> : will perform analysis on the # bp surrounding the peak centers [example: -size 1000]
-size <#,#> : will perform analysis from # to # relative to peak center [example: -size -200,50]
-size given : will perform analysis on different sized peaks - size given by actual coordinates in peak/BED file [example: -size given]

For example, if you peaks are actually transcription start sites, you might want to specify "-size -500,100" to perform the analysis upstream -500 bp to +100 bp downstream. If your peaks/regions are actually "transcript" regions, specifying "**-size given**" will count reads along the entire transcript. If it doesn't make sense, watch Delta Force I and II back to back. That should numb the brain enough to get it.

Annotating Individual Peaks

Calculating ChIP-Seq Tag Densities across different experiments

annotatePeaks.pl is useful program for cross-referencing data from multiple experiments. In order to count the number of tags from different sequencing experiments, you must first create tag directories for each of these experiments. Once created, tag counts from these directories in the vicinity of your peaks can be added by specifying "**-d <tag directory 1> <tag directory 2> ...**". You can specify as many tag directories as you like. Tag totals for each directory will be placed in new columns starting on column 18. For example:

annotatePeaks.pl pu1peaks.txt mm8 -size 400 -d Macrophage-PU.1/ Bcell-PU.1/ > output.txt

output.txt, when opened in EXCEL, will look like this:

N	O	P	Q	R	S	T	U	V	W	X
est Refs	Nearest Ense	Gene Name	Gene Alias	Gene Descrip	Macrophage-PU.1/ Tag C	Bcell-PU.1/ Tag Count in 400 bp (8861792 Total, normalization factor = 1.13, effe				
)010133	ENSMUSG000	Gm11487	OTTMUSG00(	predicted ger	3268.14	8209.4				
)010455	ENSMUSG000	Zfp868	AI449175|AV	zinc finger pr	4919.21	11197.51				
)25423	ENSMUSG000	1110059E24F	-	RIKEN cDNA 1	3.78	1005.44				
)10924	ENSMUSG000	Nnmt	-	nicotinamide	935.1	1751.34				
)11412	ENSMUSG000	Slit3	Slil2	slit homolog	3.78	804.58				
.77662	ENSMUSG000	Ctso	A330105D01	cathepsin O	5.67	920.81				
)28810	ENSMUSG000	Rnd3	2610017M01	Rho family G	846.32	1663.32				
)11518	ENSMUSG000	Syk	-	spleen tyrosi	627.18	1649.78				

HOMER automatically normalizes each directory by the total number of mapped tags such that each directory contains **10 million tags**. This total can be changed by specifying "**-norm <#>**" or by specifying "**-noadj**", which will skip this normalization step.

The other important parameter when counting tags is to specify the size of the region you would like to count tags in with "**-size <#>**". For example, "**-size 1000**" will count tags in the 1kb region centered on each peak, while "**-size 50**" will count tags in the 50 bp region centered on the peak (default is 200). The number of tags is not normalized by the size of the region.

One last thing to keep in mind is that in order to fairly count tags, HOMER will automatically center tags based on their estimated ChIP-fragment lengths. This is can be overridden by specifying a fixed ChIP-fragment length using "**-len <#>**" or "**-fragLength <#>**". This is important to consider when trying to count RNA tags, or things such as 5' RNA CAGE/TSS-Seq, where you may want to specify "**-len 0**" so that HOMER doesn't try to move the tags before counting them.

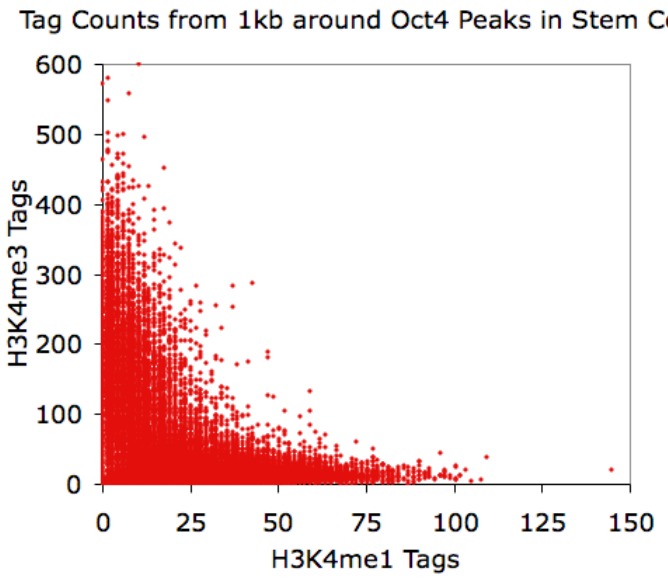
Making Scatter Plots

X-Y scatter plots are a great way to present information, and by counting tag densities from different tag directories, you can visualize the relative levels of different sequencing experiments. Because of the large range of values, it can be difficult to appreciate the relationship between data sets without log transforming the data (or sqrt to stay Poisson friendly). Also, due to the digital nature of tag counting, it can be hard to properly assess the data from a X-Y scatter plot since may of the data points will have the same values and overlap. To assist with these issues, you can specify "**-log**" or "**-sqrt**" to transform the data. These functions will actually report "log(value+1+rand)" and "sqrt(value+rand)", respectively, where rand is a random

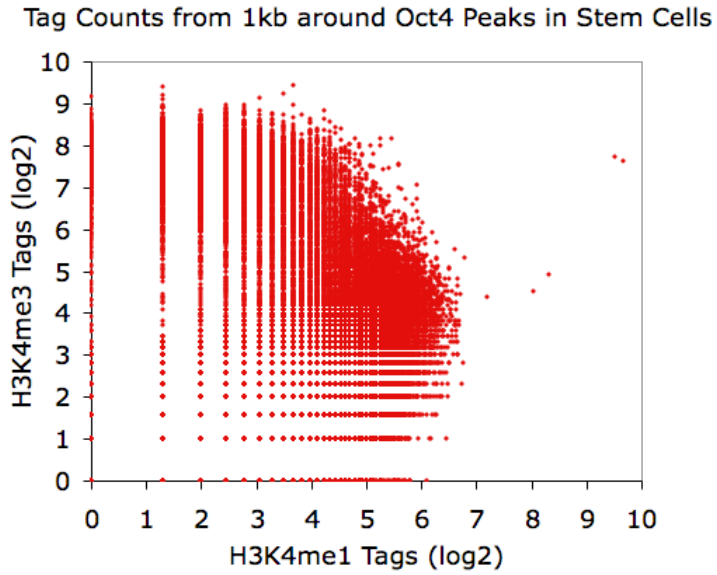
"fraction of a tag" that adds *jitter* to your data so that data points with low tag counts will not have exactly the same value. For example, lets look at the distribution of H3K4me1 and H3K4me3 near Oct4 peaks in mouse embryonic stem cells:

annotatePeaks.pl Oct4.peaks.txt mm8 -size 1000 -d H3K4me1-ChIP-Seq/ H3K4me3-ChIP-Seq/ > output.txt

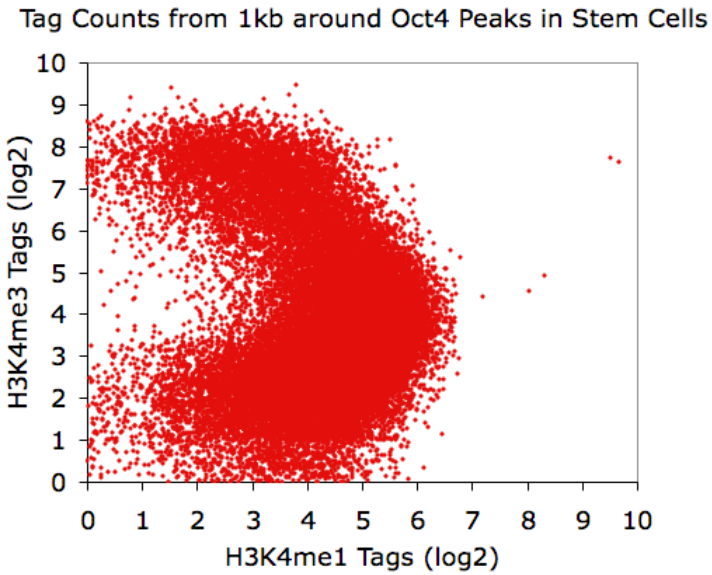
Opening output.txt with EXCEL and plotting the last two columns:



Using EXCEL to take the log(base 2) of the data:



Now using the "-log" option:
annotatePeaks.pl Oct4.peaks.txt mm8 -size 1000 **-log** -d H3K4me1-ChIP-Seq/ H3K4me3-ChIP-Seq/ > output.txt



Believe it or not, all of these X-Y plots show the same data. Interesting, eh?

Finding instances of motifs near peaks

Figuring out which peaks have instances of motifs found with findMotifsGenome.pl is very easy. Simply use "**-m <motif file1> <motif file 2>...**" with **annotatePeaks.pl**. (Motif files can be concatenated into a single file for ease of use) This will search for each of these motifs near each peak in your peak file. Use "**-size <#>**" to specify the size of the region around the peak center you wish to search. Found instances of each motif will be reported in additional columns of the output file. For example:

annotatePeaks.pl pu1peaks.txt mm8 -size 200 -m pu1.motif cebp.motif > output.txt

Opening output.txt with EXCEL:

O	P	Q	R	S	T	U	V
arest Ense	Gene Name	Gene Alias	Gene Descrip	CpG%	GC%	PU.1/ThioMac-PU.1-ChIP-Seq/Homer Distance From Peak(sequence,strand,conservation)	CEBP/CEBPb-ChIP-Seq/Homer Dista
MUSG000	Dsn1	1700022L09F	DSN1, MIND I	0.015	0.621891	-55(GGAGGAAGTG,+,0.00),-26(TGGGGAAGTG,+,0.00),35(GCAGGAAGTG,+,0.00)	
MUSG000	Cd53	AI323659|Ox	CD53 antiger	0.01	0.432836	34(TGAGGAAGTG,+,0.00)	
MUSG000	Atg4c	Apg4-C|Apg4	autophagy-re	0.01	0.378109	8(AGAGGAACTG,+,0.00),54(CACTTCCCTT,-,0.00)	-28(ATTTCATAAC,+,0.00)
MUSG000	Tax1bp1	1200003J11F	Tax1 (humar	0.019608	0.307692		
MUSG000	Plekho2	AI840980|M(	pleckstrin ho	0.005	0.532338	-11(TGGGGAAGTG,+,0.00),33(TGAGGAAGTG,+,0.00)	
MUSG000	Cttnbp2nl	AA552995|A4	CTTNBP2 N-t	0.01	0.462687	-97(GGAGGAAGTG,+,0.00),48(ACAGGAACTG,+,0.00)	
MUSG000	Pltp	OD107	phospholipid	0.005	0.467662	29(CAGTTCCTTT,-,0.00)	
MUSG000	Ifngr2	Ifgr2|Ifgt	interferon ga	0.01	0.462687	-24(CACTTCCTCA,-,0.00),49(CACTTCCTCA,-,0.00)	
MUSG000	Aff2	Fmr2|Ox19|(	AF4/FMR2 fa	0	0.512438		
MUSG000	Syk	-	spleen tyrosi	0.01	0.497512	40(AGGGGAACTG,+,0.00),60(GGAGGAACTG,+,0.00)	
MUSG000	Exoc6	4833405E05F	exocyst comp	0.015	0.432836	-41(CAGTTCCTCT,-,0.00)	
MUSG000	Ccdc109b	9030408N13	coiled-coil do	0.005	0.482587	71(GGAGGAAGTG,+,0.00)	53(GTTGTGTAAG,-,0.00)
MUSG000	Ctnnd2	Catnd2|Npraj	catenin (cadh	0.021858	0.472826	-54(AGCGGAAGTG,+,0.00),18(CAGTTCCTGT,-,0.00),34(CACTTCCTCT,-,0.00)	64(GTTTCATAAT,-,0.00)
MUSG000	Slc2a1	Glut-1|Glut1	solute carrier	0.02	0.547264	-28(AGAGGAAGTG,+,0.00),-16(CACTTCCTCT,-,0.00)	

Each instance of the motif is specified in the following format (separated by commas):

Distance from Peak Center(*Sequence Matching Motif*,*Strand*,*Average Conservation*)

The average conservation will not be reported unless you specify "**-cons**". Also, when finding motifs, the average CpG/GC content will automatically be reported since it has to extract peak sequences from the genome anyway.

There are also a bunch of motif specific options for specialized analysis:

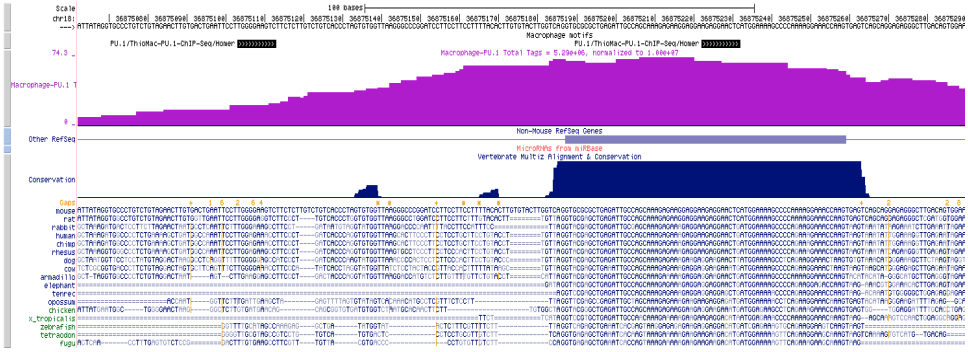
"**-norevopp**" (only search + strand relative to peak strand for motifs)
"**-nmotifs**" (just report the total number of motifs per peak)
"**-mscore**" (report the maximum log-odds score of the motif in each peak)
"**-rmrevopp <#>**" (tries to avoid double counting reverse opposites within # bp)
"**-mdist**" (reports distance to closest motif)
"**-fm <motif file 1> [motif file 2]**" (list of motifs to filter out of results if found)
"**-mfasta <filename>**" (reports sites in a fasta file - for building new motifs)
"**-mbed <filename>**" (Output motif positions to a BED file to load at UCSC - see below)
"**-matrix <filename>**" (outputs a motif co-occurrence matrix with the p-value of co-occurrence assuming instance of each motif are independently distributed amongst the peaks)

Visualizing Motif positions in the UCSC Genome Browser

This feature may seem slightly out of place, but since **annotatePeaks.pl** is the workhorse of HOMER, you can add "**-mbed <filename>**" in conjunction with "**-m <motif file 1> [motif file 2] ...**" to produce a BED file describing motif positions near

peaks that can be loaded as a custom track in the UCSC genome browser. In the example below, you would load motif.bed as a custom track:

annotatePeaks.pl pu1peaks.txt mm8 -size 200 -m pu1.motif cebp.motif -mbed motif.bed > output.txt



Finding the distance to other sets of Peaks

In order to find the nearest peak from another set of peaks, use "**-p <peak file 1> [peak file 2] ...**". This will add columns to the output spreadsheet that will specify the nearest peak ID and the distance to that peak. If all you want is the distance (so you can sort this column), add the option "**-pdist**" to the command. Otherwise, if you prefer to count the number of peaks in the peak file found within the indicated regions (i.e. with "**-size <#>**"), add "**-pcount**".

Creating Histograms from High-throughput Sequencing data and Motifs

HOMER can be used to make histograms that document sequencing library and motif densities relative to specific positions in the genome. This can be done near peaks, subsets of peaks, or near promoters, exon junctions or anywhere else you find interesting. To make histograms, use the **annotatePeaks.pl** program but add the parameters "**-hist <#>**" to produce a tab delimited text file that can then be visualized using EXCEL or other data visualization software.

Basic usage:

annotatePeaks.pl <peak file> <genome> -size <#> -hist <#> -d <tag directory 1> [tag directory2] ... -m <motif 1> <motif 2> ... > <output matrix file>

i.e. **annotatePeaks.pl ERpeaks.txt hg18 -size 6000 -hist 25 -d MCF7-H3K4me1/ MCF7-H3K4me2/ MCF7-H3K4me3/ > outputfile.txt**

Running this command is very similar to creating annotated peak files - in fact, most of the data can be used to make both types of files - hence the reason for combining this functionality in the same command. Be default, HOMER normalizes the output histogram such that the resulting units are **per bp per peak**, on top of the standard total mapped tag normalization of **10 million tags**.

Histograms of Tag Directories:

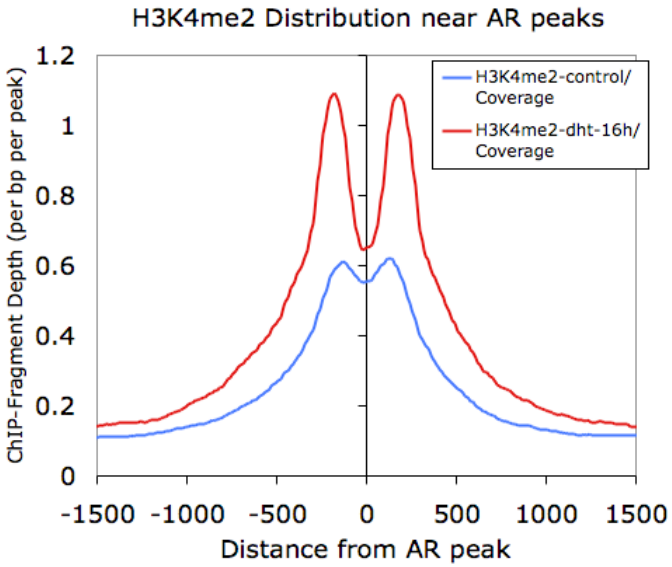
For each tag directory or motif, HOMER will output 3 columns in the histogram. In the case of tag directories, the first column will indicate **ChIP-Fragment Coverage**, which is calculated by extending tags by their estimated ChIP-fragment length, and is analogous to the profiles made for the UCSC Genome Browser. The 2nd and 3rd columns report the density of 5' and 3' aligned tags, and are independent of fragment length. For example, lets look at H3K4me2 distribution near Androgen Receptor (AR) peaks before and after 16 hours of treatment with testosterone (dht):

annotatePeaks.pl ARpeaks.txt hg18 -size 4000 -hist 10 -d H3K4me2-control/ H3K4me2-dht-16h/ > outputfile.txt

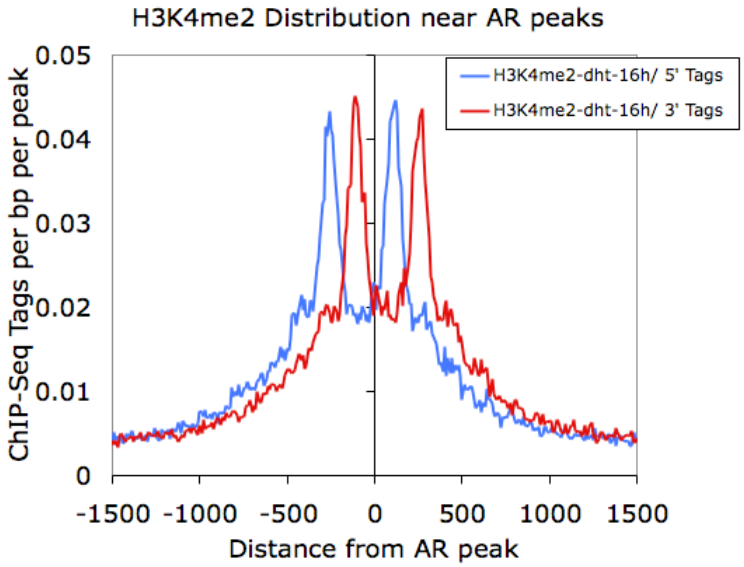
Opening outputfile.txt with EXCEL, we see:

	A	B	C	D	E	F	G
1	Distance from Center	H3K4me2-control/ Coverage	H3K4me2-control/ 5' Tags	H3K4me2-control/ 3' Tags	H3K4me2-dht-16h/ Coverage	H3K4me2-dht-16h/ 5' Tags	H3K4me2-dht-16h/ 3' Tags
2	-2000	0.046971	0.001564	0.001734	0.059736	0.001824	0.002136
3	-1990	0.052394	0.003621	0.003179	0.06552	0.004344	0.002928
4	-1980	0.05661	0.003689	0.003298	0.070872	0.004008	0.003528
5	-1970	0.059721	0.003026	0.003128	0.075048	0.003816	0.003912
6	-1960	0.063614	0.003638	0.002975	0.079128	0.00396	0.00372
7	-1950	0.066742	0.002958	0.003298	0.084048	0.004392	0.00372
8	-1940	0.069751	0.003315	0.003145	0.087792	0.004128	0.003432
9	-1930	0.072403	0.003281	0.00323	0.091416	0.003312	0.004632
10	-1920	0.075259	0.002924	0.002958	0.0936	0.003504	0.004224

Graphing columns B and E while using column A for the x-coordinates, we get the following:



However, if we graph only the 5' and 3' tags that come from the H3K4me2-dht-16h directory (columns F and G):



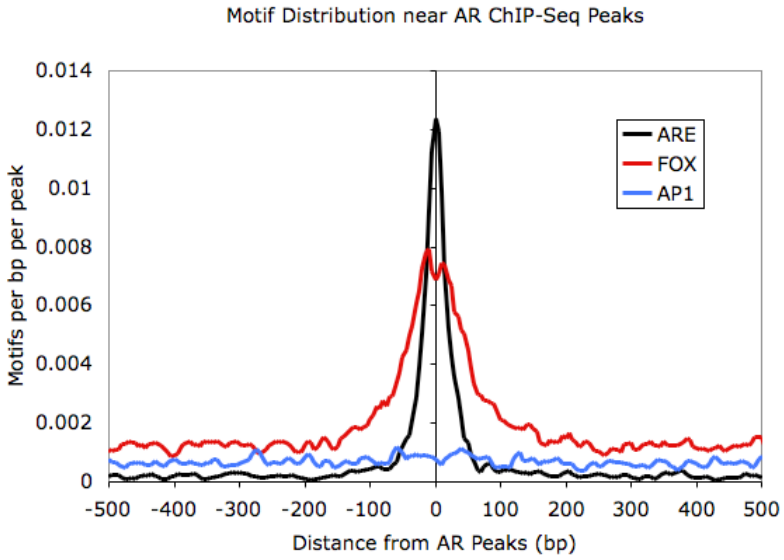
Here we can see how the 5' and 3' reads from the H3K4me2 marked nucleosomes are distributed near the AR sites.

Histograms of Motif Densities:

Making histograms out of motif occurrences is very similar to sequencing tag distributions. Run the annotatePeaks.pl program with "**-hist <#>**" and "**-m <motif file>**" (you can also find motif densities and tag densities at the same time):

annotatePeaks.pl ARpeaks.txt hg18 -size 1000 -hist 5 -m are.motif fox.motif ap1.motif > outputfile.txt

Graphing outputfile.txt with EXCEL:



Centering Peaks on Motifs

One cool analysis strategy is to center peaks on a specific motif. For example, by centering peak for the Androgen Receptor Transcription Factor on the ARE motif (GNACANNNTGTNC), you can map the spacial relationship between the motif and other sequence features and sequencing reads.

To center peaks on a motif, run **annotatePeaks.pl** with the following options:

annotatePeaks.pl <peak file> <genome> -size <#> -center <motif file> > newpeakfile.txt

i.e. **annotatePeaks.pl ARpeaks.txt hg18r -size 200 -center are.motif > areCenteredPeaks.txt**

Now the idea is to use the new peak file to perform analysis:

annotatePeaks.pl areCenteredPeaks.txt hg18 -size 6000 -hist 10 -d H3K4me2-notx/ ... > output.txt

Creating Heatmaps from High-throughput Sequencing data

HOMER is not capable of generating actual Heatmaps *per se*, but it will generate the data matrix (similar to a gene expression matrix) than can then be visualized using standard gene expression heatmap tools. For example, I will generate a heatmap data matrix file using HOMER, and then open it with Cluster 3.0 (Micheal Eisen/de Hoon) to cluster it and/or visualize it with Java Tree View (by Alok J. Saldanha). In reality, you can use any clustering and/or heatmap visualization software (i.e. R).

Basic usage (add "**-ghist**" when making a histogram):

annotatePeaks.pl <peak file> <genome> -size <#> -hist <#> -ghist -d <tag directory 1> [tag directory2] ... > <output matrix file>

i.e. **annotatePeaks.pl ARpeaks.txt hg18 -size 6000 -hist 25 -ghist -d H3K4me2-control/ H3K4me2-dht-16h/ > outputfile.txt**

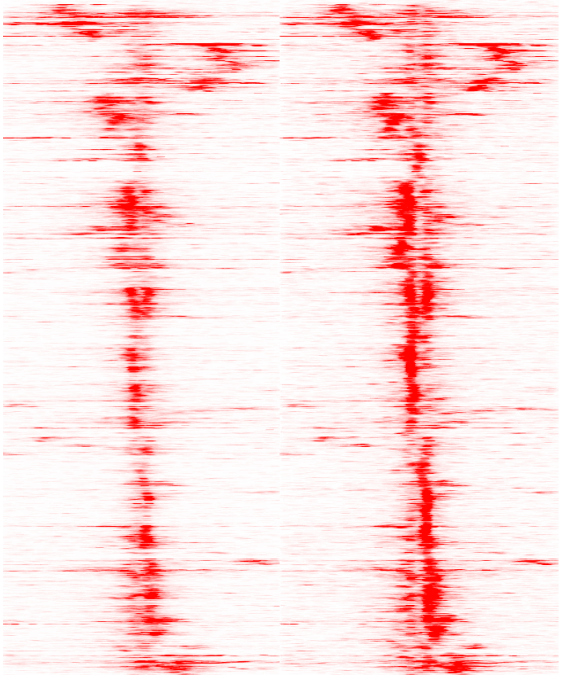
Running this command is very similar to making histograms with **annotatePeaks.pl**. In fact, a heatmap isn't really all that different from a histogram - basically, instead of averaging all of the data from each peak, we keep data from each peak separate and visualize it all together in a heatmap. The key difference when making a heat map or a histogram is that you must add "**-ghist**" when making a heatmap.

Format of Data Matrix Output File

The resulting file is a tab-delimited text file where the first row is a header file and the remaining rows represent each peak from the input peak file. The first set of columns will describe the read densities from the **first tag directory** as a function of distance from the center of the peak, with bin sizes corresponding to the parameter used with "**-hist <#>**". After the first block of columns, a second block will start over with read densities from the **2nd tag directory**, and so on. If you would like to cluster this file to help organize the patterns, make sure you only cluster the "genes" (i.e. rows).

Example: **annotatePeaks.pl ARpeaks.txt hg18 -size 6000 -hist 25 -ghist -d H3K4me2-control/ H3K4me2-dht-16h/ > outputfile.txt**

After creating the file, I loaded into Cluster 3.0 to cluster and clustered the genes using "centered correlation" as the distance metric, then loaded that output into Java Tree View.



General Options to Control Data Analysis Behavior

-strand <+|-|both> (only look for motif etc. on + or - strand, default both)
-fragLength <#> (Fragment length, default=auto, might want to set to 0 for RNA)
-pc <#> (maximum number of tags to count per bp, default=0 [no maximum])
-cons (Retrieve conservation information for peaks/sites - creates new column for this information)
-CpG (Calculate CpG/GC content)
-norevopp (do not search for motifs on the opposite strand [works with -center too])
-norm <#> (normalize tags to this tag count, default=1e7, 0=average tag count in all directories)
Use **-noadj** to disable tag normalization for sequencing depth

Command line options for annotatePeaks.pl

Usage: annotatePeaks.pl <peak file | tss> <genome version> [additional options...]

Available Genomes (required argument): (name,org,directory,default promoter set)

User defined annotation files (default is UCSC refGene annotation):
annotatePeaks.pl accepts GTF (gene transfer formatted) files to annotate positions relative
to custom annotations, such as those from de novo transcript discovery or Gencode.
-gtf <gtf format file>

Peak vs. tss / tts mode (works with custom GTF file):
If the first argument is "tss" (i.e. annotatePeaks.pl tss hg18 ...) then a TSS centric
analysis will be carried out. Tag counts and motifs will be found relative to the TSS.
(no position file needed) ["tts" now works too - e.g. 3' end of gene]
NOTE: The default TSS peak size is 4000 bp, i.e. +/- 2kb (change with -size option)
-list <gene id list> (subset of genes to perform analysis [unigene, gene id, accession,
probe, etc.], default = all promoters)

Primary Annotation Options:
-m <motif file 1> [motif file 2] ... (list of motifs to find in peaks)
-mscore (reports the highest log-odds score within the peak)
-nmotifs (reports the number of motifs per peak)
-mdist (reports distance to closest motif)
-mfasta <filename> (reports sites in a fasta file - for building new motifs)
-fm <motif file 1> [motif file 2] (list of motifs to filter from above)
-rmrevopp <#> (only count sites found within <#> on both strands once, i.e. palindromic)
-matrix <filename> (outputs a motif co-occurrence matrix)
-mbed <filename> (Output motif positions to a BED file to load at UCSC (or -mpeak)

```
   -d <tag directory 1> [tag directory 2] ... (list of experiment directories to show
      tag counts for) NOTE: -dfile <file> where file is a list of directories in first column
   -p <peak file> [peak file 2] ... (to find nearest peaks)
      -pdist to report only distance (-pdist2 gives directional distance)
      -pcount to report number of peaks within region
   -gene <data file> ... (Adds additional data to result based on the closest gene.
      This is useful for adding gene expression data.  The file must have a header,
      and the first column must be a GeneID, Accession number, etc.  If the peak
      cannot be mapped to data in the file then the entry will be left empty.
   -go <output directory> (perform GO analysis using genes near peaks)
   -genomeOntology <output directory> (perform genomeOntology analysis on peaks)
      -gsize <#> (Genome size for genomeOntology analysis, default: 2e9)

Annotation vs. Histogram mode:
   -hist <bin size in bp> (i.e 1, 2, 5, 10, 20, 50, 100 etc.)
   The -hist option can be used to generate histograms of position dependent features relative
   to the center of peaks.  This is primarily meant to be used with -d and -m options to map
   distribution of motifs and ChIP-Seq tags.  For ChIP-Seq peaks for a Transcription factor
   you might want to use the -center option (below) to center peaks on the known motif
   ** If using "-size given", histogram will be scaled to each region (i.e. 0-100%), with
   the -hist parameter being the number of bins to divide each region into.
      Histogram Mode specific Options:
      -nuc (calculated mononucleotide frequencies at each position,
         Will report by default if extracting sequence for other purposes like motifs)
      -di (calculated dinucleotide frequencies at each position)
      -histNorm <#> (normalize the total tag count for each region to 1, where <#> is the
         minimum tag total per region - use to avoid tag spikes from low coverage
      -ghist (outputs profiles for each gene, for peak shape clustering)
      -rm <#> (remove occurrences of same motif that occur within # bp)

Peak Centering: (other options are ignored)
   -center <motif file> (This will re-center peaks on the specified motif, or remove peak
      if there is no motif in the peak.  ONLY recentering will be performed, and all other
      options will be ignored.  This will output a new peak file that can then be reanalyzed
      to reveal fine-grain structure in peaks (It is advised to use -size < 200) with this
      to keep peaks from moving too far (-mirror flips the position)
   -multi (returns genomic positions of all sites instead of just the closest to center)

Advanced Options:
   -len <#> / -fragLength <#> (Fragment length, default=auto, might want to set to 0 for RNA)
   -size <#> (Peak size[from center of peak], default=inferred from peak file)
      -size #,# (i.e. -size -10,50 count tags from -10 bp to +50 bp from center)
      -size "given" (count tags etc. using the actual regions - for variable length regions)
   -log (output tag counts as log2(x+1+rand) values - for scatter plots)
   -sqrt (output tag counts as sqrt(x+rand) values - for scatter plots)
   -strand <+|-|both> (Count tags on specific strands relative to peak, default: both)
   -pc <#> (maximum number of tags to count per bp, default=0 [no maximum])
   -cons (Retrieve conservation information for peaks/sites)
   -CpG (Calculate CpG/GC content)
   -ratio (process tag values as ratios - i.e. chip-seq, or mCpG/CpG)
   -norevopp (do not search for motifs on the opposite strand [works with -center too])
   -noadj (do not adjust the tag counts based on total tags sequenced)
   -norm <#> (normalize tags to this tag count, default=1e7, 0=average tag count in all directories)
   -pdist (only report distance to nearest peak using -p, not peak name)
   -noann, -nogene (skip genome annotation step, skip TSS annotation)
   -homer1/-homer2 (by default, the new version of homer [-homer2] is used for finding motifs)
```

Can't figure something out? Questions, comments, concerns, or other feedback:
cbenner@ucsd.edu

HOMER

Software for motif discovery and next-gen sequencing analysis

---

Quantifying RNA with analyzeRNA.pl

Homer contains a program (**analyzeRNA.pl**) to quantify RNA reads in genes. Of all the tools in HOMER, this one is probably the least efficient and resource hungry program in terms of memory. It needs to be rewritten but I have not found the time or pressing need to do this yet. However, it does perform useful calculations, particularly for GRO-Seq applications, that are not available or not automated for other programs.

analyzeRNA.pl DOES produces a gene expression matrix from "**Tag Directories**", and works in a similar manner to how annotatedPeaks.pl works for **ChIP-Seq** data. It has options to count reads in "genic" vs. "exon" regions, and attempts to automate the analysis of promoter proximal pausing seen in GRO-Seq data.

analyzeRNA.pl DOES NOT produce differential expression or differential splicing calls. It would be great if Chuck had the time to help me with that, but for now I recommend sending the output from analyzeRNA.pl to other utilities such as edgeR or DEseq or whatever your favorite next-gen differential signal finder is.

Basic usage:

analyzeRNA.pl <rna|repeats|gtf file> <genome> [options] -count [genes|exons|introns] -d <Tag Directory> [Tag Directory 2] ... > <output file>

i.e. **analyzeRNA.pl rna hg18 -count genes -d IMR90-GroSeq > outputfile.txt**

Specifying the Genes/Transcripts to Analyze

The first argument to analyzeRNA.pl specifies which "gene definition" to use for analysis. There are three options:

1. **rna** - This will direct HOMER to analyze the default RefSeq annotation for that genome.
2. **repeats** - HOMER will load repeat definitions from UCSC and assess the expression levels of different repeat classes. This can be resource hungry and is not recommended unless your computer as a bunch of memory.
3. **<GTF file>** - Specify your own custom genes. These can an alternative annotation (i.e. Ensembl genes, UCSC genes) or it can be custom, de novo genes you found analyzing GRO-Seq or mRNAs defined using "cufflinks". If you are looking for a GTF file for you favorite annotation, check out the UCSC Table Browser. You can use this tool to download GTF formmated files (with the correct genome version) for many popular annotations.

Examples include "analyzeRNA.pl rna mm9 ..." or "analyzeRNA.pl myGenes.gtf mm9 ...".

Choosing Experiments to Analyze

As with all HOMER programs, you need to create "Tag Directories" out of each experiment you want to quantify first. To quantify them with **analyzeRNA.pl** simply add one or more directory after "**-d**". For example:

analyzeRNA.pl rna hg18 -d LNCaP-RNA-Seq-notx/ LNCaP-RNA-Seq-Dht/ > outputfile.txt

This will produce an output file containing genes expression values for two experiments, "LNCaP-RNA-Seq-notx" and "LNCaP-RNA-Seq-Dht".

Measuring Gene Expression in Exons vs. Gene Bodies.

Depending on the type of sequencing you are analyzing, you will want to quantify RNA from different parts of the gene. The "-count [...]" option controls which regions of the gene are used for analysis (use like "**-count exons**" or "**-count genes**")

- **exons** (default) Counts tags in exons only. Use this for most applications of RNA-Seq, such as polyA-RNA-seq or other techniques that aim to measure mRNA.
- **cds** - Counts tags in coding regions only. This could be useful for quantifying ribosome coverage on coding sequences with techniques such as Ribo-Seq
- **introns** - Counts tags on introns only.
- **5utr** or **3utr** - Count tags on 5' UTR and 3' UTR regions, respectively
- **genes** - Counts tags on the full gene body (TSS to TTS). This is useful for GRO-Seq where we expect coverage across the entire transcript. Can also be used to quantify H3K36me3 or PolII ChIP-Seq.

By default, analyzeRNA.pl assumes your RNA is strand-specific. If it is not, or you are using analyzeRNA.pl for other types of data such as ChIP-Seq, you can specify:

"**-strand +**" - default, only measure + strand (relative to gene orientation), for strand specific RNA
"**-strand -**" - only measure - strand (relative to gene orientation)
"**-strand both**" - count reads on both strands, for non-strand specific RNA or ChIP-Seq

Also, there are a couple options to slightly modify how tags are counted to avoid TSS/TTS features. By specifying "**-start <#>**", you can adjust where (relative to the TSS) HOMER starts counting reads. This is useful for GRO-Seq when you might want to avoid the promoter-proximal paused region by add "**-start 500**" to start counting 500 bp downstream from the TSS. Negative values would start counting 500 bp upstream. The optoin "**-end <#>**" is similar, but applies to the TTS. Negative values will stop the tag counting upstream of the TTS ("**-end -500**"), while positive values will count tags further downstream.

Normalization of Gene Expression Values

Normalization is probably the trickiest part about RNA-Seq. There are lots of papers on it. HOMER offers a couple different normalization options depending on what the experiment is and what your needs are.

By default, HOMER normalizes each experiment to 10 million mapped reads, which is the same normalization strategy used in **annotatePeaks.pl** for ChIP-Seq data. However, RNA has the potential to contain much more "contaminates" than ChIP-Seq. In ChIP-Seq, the background is the genome, with random fragments coming down in the immunoprecipitation step. With RNA, instead of a two copies of the genome per cell, you have 99% ribosomal RNA, along with a host of other very very common short RNA species such as tRNAs, snoRNAs, and other problamatic things, and then a small fraction of what you're actually interested in. Most RNA-Seq protocols contain enrichment steps, such as polyA selection, to isolate mRNAs from the rest of the crap (my appoligies to those studying rRNA and tRNA). These enrichment steps can have different efficiencies from sample to sample, with some samples containing more rRNA, tRNA etc. than others. When mapping to the genome, many of these RNA species will be discarded if you only keep reads that map uniquely since the genomic elements that create them are often repeated in the genome. However, you're bound to still map many of these reads. Differential contamination of common RNA species can throw off the default normalization - if 50% of your "mapped" reads are rRNA in one sample and only 25% in another, you're likely going to have problems.

There are several things you can do to combat these problems. One is to "pre-clear" common RNA species computationally, i.e. map your reads to rRNA first to remove them, then map the rest to the genome. Another strategy is to normalize strictly with mRNA species only instead of considering the total number of mapped reads. In general, I would recommend normalizing in this strategy ("**-normMatrix <#>**") - it's probably the safest in most situations due to the contamination problem. Below are the various normalization options in **analyzeRNA.pl**.

- **-norm <#>** - Normalize the total number of mapped reads per experiment to #, this is the default "**-norm 1e7**".
- **-normMatrix <#>** - Normalize the total of number of reads found in the gene expression matrix to # (i.e. normalize total reads in mRNAs)
- **-noadj** - Don't perform any normalization, just report the exact number of reads found. This is useful when sending the output of HOMER to another differential expression program such as edgeR or similar that requires the raw read counts.

The options above perform the actual normalization. "Dress up" normalization/transformations options follow:

- **-rpkm** - Report normalized values as reads per kilobase per million mapped reads

- **-log** - Report log normalized values with jitter =log2(x+1+rand(0-1))
- **-sqrt** - Report square root of the normalized value with jitter =sqrt(x+rand(0-1))

Limiting the number of Tags per Position ("-pc <#>")

Another important parameter (that I almost made mandatory) is "**-pc <#>**", which controls the maximum number of reads to consider per position. In the case of some genes, there may be high-abundance RNA species in introns, where large spikes in RNA reads may occur. If quantifying expression from gene bodies (i.e. "-count genes", GRO-Seq), these reads will be counted toward the gene's total abundance. Ideally these are removed. However, as a quick and dirty solution to this problem, you can limit the number of reads that HOMER counts from each position. In many cases rRNA loci will have tens of thousands of reads per bp. On the flip side, limiting the number of reads per bp can kill your dynamic range. Usually the middle gound such as "**-pc 3**" is a good way to go - you effectively remove the super spikes but still allow genes with high RPKM to "express themselves" in your analysis.

How analyzeRNA.pl Works

The following are the steps HOMER takes to produce a Gene Expression Matrix:

1. Parses the RNA definition file to figure out where all of the genes are in the genome.
2. Queries each of the "Tag Directories" and determine where each of the tags are within gene bodies.
3. At that point it will rummage through the tag positions and count only the tags found within the desired regions (i.e. exons). Incidently, this is also the part that makes it less efficient at the momment - it's done with perl and keeps way too much extra information around.
4. Expression values are calculated and normalized
5. Results are formatted and set to *stdout*

Descrition of Output Files

The output gene expression matrix with the following columns:

1. Transcript ID (RefSeq accession is the default with "rna" option, custom name is used otherwise)
2. chromosome
3. start
4. end
5. strand
6. mRNA length (e.g. exons only)
7. gene length (TSS to TTS)
8. Copies in genome (some genes map to the genome more than once...)
9. Symbol (e.g Gene Name)
10. Alias (alternative gene names)
11. Description
12. Unigene
13. Entrez Gene ID
14. Ensembl
15. Data: First Tag Directory Gene Expression information
16. [Data: Second Tag Directory Gene Expression information]
17. ...

A separate column is generated for each tag directory. The head of these columns contains the name, the region of the gene used for expression (i.e. gene, exon, 5utr..), the total number of mapped tags in the directory, and the normalization factor.

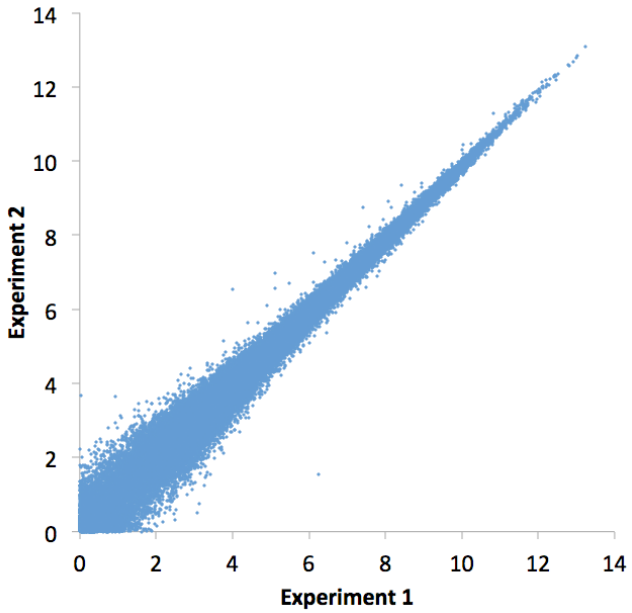
Due to the fact that some genes map to multiple places in the genome (<1% of RefSeq), only one of their locations will be reported. However, the gene length and mRNA lengths will be averaged, and their read totals will be averaged in the output.

If custom IDs are used, HOMER will attempt to link them to known gene identifiers (As of now it does not annotate their position, just treats their ID as an accession number and tries to match it wil known genes). If you give analyzeRNA.pl a custom GTF file from Cufflinks, where the IDs are all "CUFF12345.1", most of the annotation columns will be blank.

Below is an example of the output opened with EXCEL:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Genes	chr	start	end	stran	mRNA Length	Gene Length	Copies in	Symbol	Alias	Description	Unigene	GeneID	Ensembl	Macrophage-RNA-nc	Macrophage-RN
2	NM_001001130	chr13	67848736	67856071	-	2218	7335	1	Zfp85-rs1	KRAB19|Rslc	zinc finger pr	Mm.288396	22746	ENSMUSG	25.49128477	26.38801643
3	NM_001001144	chr9	110235796	110287450	+	4286	51654	1	Scap	9530044G19	SREBF chape	Mm.288741	235623	ENSMUSG	82.8466755	160.357946
4	NM_001001152	chr13	67355853	67370004	-	3488	14151	1	Zfp458	BC062958|R	zinc finger pr	Mm.306358	238690	ENSMUSG	11.15243709	12.85570031
5	NM_001001160	chr6	85419571	85452880	-	6562	33309	1	Fbxo41	9630017H13	F-box protein	Mm.38777	330369	ENSMUSG	0	0
6	NM_001001176	chrX	103402212	103415181	-	2583	12969	1	Taf9b	BC066223|Ta	TAF9B RNA p	Mm.19440	407786	ENSMUSG	0	0
7	NM_001001177	chr17	34535764	34597679	+	1900	61915	1	BC051142	NG8|TSBP|T	cDNA sequer	Mm.73205	407788	ENSMUSG	0.796602649	0
8	NM_001001178	chr2	58674108	58998684	-	3920	324576	1	Ccdc148	5830402J09	coiled-coil dc	Mm.432280	227933	ENSMUSG	0	0
9	NM_001001179	chr6	128489839	128531624	-	4698	41785	1	BC048546	MGC58520|(	cDNA sequer	Mm.259234	232400	ENSMUSG	0	0
10	NM_001001180	chr7	147995575	148008077	-	3909	12502	1	Zfp941	-	zinc finger pr	Mm.359154	407812	ENSMUSG	0	0.676615806
11	NM_001001181	chr18	75165553	75169587	+	1047	4034	1	BC031181	-	cDNA sequer	Mm.29866	407819	ENSMUSG	157.7273245	150.2087089
12	NM_001001182	chr2	59737419	59963797	-	7980	226378	1	Baz2b	5830435C13	bromodomai	Mm.436730	407823	ENSMUSG	215.8793179	158.3280986
13	NM_001001183	chr17	25194646	25218059	-	1779	23413	1	Tmem204	-	transmembr	Mm.34379	407831	ENSMUSG	1.593205298	0.676615806
14	NM_001001184	chr8	47660947	47702554	-	3797	41607	1	Ccdc111	BC065112|M	coiled-coil dc	Mm.217385	408022	ENSMUSG	145.7782848	63.60188576
15	NM_001001185	chr13	67964273	67964854	+	581	581	1	BC048507	-	cDNA sequer	Mm.177840	408058	ENSMUSG	24.69468212	33.8307903
16	NM_001001186	chr13	67464519	67476699	-	4079	12180	1	Zfp456	BC065418|M	zinc finger pr	Mm.461583	408065	ENSMUSG	47.79615894	33.15417449
17	NM_001001187	chr13	67768377	67784449	-	4504	16072	1	Zfp738	3830402I07R	zinc finger pr	Mm.435551	408068	ENSMUSG	218.2691258	174.5668779
18	NM_001001295	chr9	64154562	64189064	-	3575	34502	1	Dis3l	AV340375	DIS3 mitotic	Mm.268341	213550	ENSMUSG	45.40635099	64.27850156
19	NM_001001297	chr2	21127350	21136636	+	2327	9286	1	Thnsl1	AW413632	threonine sy	Mm.268841	208967	ENSMUSG	23.10147682	12.17908451
20	NM_001001309	chr2	12028286	12223547	-	5782	195261	1	Itga8	AI447669	integrin alph	Mm.329997	241226	ENSMUSG	0	0
21	NM_001001319	chr4	143649028	143659221	+	2399	10193	1	Pramel4	C79430|D4E	preferentiall	Mm.271697	347710	ENSMUSG	0	0
22	NM_001001320	chr19	3992751	3999512	+	1778	6761	1	Tbx10	Dc|MGC129	T-box 10	Mm.246555	109575	ENSMUSG	0	0
23	NM_001001321	chr13	64197617	64230638	-	2276	33021	1	Slc35d2	5730408I21R	solute carrie	Mm.133731	70484	ENSMUSG	42.2199404	37.21386932
24	NM_001001322	chr2	26828935	26865145	+	4580	36210	1	Adamts13	Gm710|vWF	a disintegrin-	Mm.330084	279028	ENSMUSG	0	1.353231612
25	NM_001001326	chr7	116667424	116760661	-	4255	93237	1	St5	2010004M01	suppression	Mm.252009	76954	ENSMUSG	140.9986689	59.54219092

If you specify the "**-log**" or "**-sqrt**" options, you can make a nice X-Y scatter plot of the data columns. Here is an example:



Quantifying Promoter-proximal RNA Polymerase Pausing

One of the neat observations from GRO-Seq experiments is that an actively engadged RNA Polymerases are present at the promoters of many genes, but they do not seem to elongate down the body of the gene. The idea is that they are "paused", waiting for elongation signals to give them the green light to make the gene.

HOMER can calculate the "pausing" ratio, which is the ratio of tag density at the promoter relative to the body of the gene. To use this option, specify "**-pausing <#>**", where # the is the distance downstream of the TSS to stop counting promoter tags and start counting gene body tags. A safe value is probably 250-500.

This will que the program to produce 3 columns of output per experiment. First, it will report the pausing ratio, and then it will report the promoter and gene body read densities.

Analyzing Repeat and non-mRNA Expression

mRNA is not the only RNA species present in the cell. In fact, it makes up a very small fraction of the total. **analyzeRNA.pl** offers an option to help quantify repeat RNAs, rRNA, etc. By specifying "**repeats**", HOMER will load the definition of all repeats in the genome and quantify tags on them. Within the repeat definition are tRNAs, rRNAs, etc. so you get more than just transposable elements. Use it like so:

analyzeRNA.pl repeats mm9 -d Macrophage-RNAseq > outputfile.txt

By default, all the repeats are treated like "exons". Repeats from similar classes are combined to give a single expression value. Only one of the poistions are reported (randomly). Be carful! It uses a lot of memory! If you're running out of memory, try using fewer tag directories (i.e. analyze one at a time).

To do repeat expression justice, you should carefully consider how the data is mapped to the genome. Normally for ChIP-Seq (or even RNA-Seq), you do not want to consider reads that map to multiple locations in the genome. However, in the case of RNA repeats, this means that you will be discarding many of the reads mapping to repeat regions. One trick that works fairly well is to keep one random position from the mapping, regardless if it maps uniquely to the genome (but only do it for this type of analysis). Typically, if a read maps to multiple locations, those multiple locations are probably all the same type of repeat element, so it will be added to the expression of that repeat class regardless of where it is specifically placed. This is "approximate", so use with care.

The thing to note is that repeat/rRNA/tRNA expression is compounded by the fact that most protocols try to get rid of it, so depending on the efficiency of these clearing steps experiment-to-experiment, you may be measuring the difference in clearing efficiency rather than the actual expression of the RNA.

Command line options for analyzeRNA.pl

Usage: analyzeRNA.pl <rna | repeats | custom RNA/GTF file> <genome version> [additional options...]

Program for quantifying RNA tag counts. The first argument can be "rna" (refseq genes), "repeats" (repeat classes), or a custom RNA definition file. (see website for format)
!!! Right now "repeats" is not memory efficient, need to optimize - i.e. 20Gb !!!
!!! Only run "repeats" with a single tag directory for now !!!

Available Genomes (required argument): (name,org,directory,default promoter set)

Primary Annotation Options:
-d <tag directory 1> [tag directory 2] ... (list of experiment directories to show tag counts for) NOTE: -dfile <file> where file is a list of directories in first column
-rpkm (Report results as reads per kb per million mapped)
-norm <#> (Normalize to total mapped tags: default 1e7)
-normMatrix <#> (Normalize to total tags in gene expression matrix: not used)
-noadj (Don't normalize)
-count <exons|introns|genes|5utr|3utr|cds> (Count tags in introns, exons, etc., default: exons")
-noCondensing (do not condense counts from entries will same ID, default: do condense)
-pc <#> (maximum tags to count per position, default: 0=no limit)
-strand <+|-|both> (count tags on indicated strand, default: +)
-gene <data file> ... (Adds additional data to result based on the closest gene. This is useful for adding gene expression data. The file must have a header, and the first column must be a GeneID, Accession number, etc. If the peak cannot be mapped to data in the file then the entry will be left empty.
-log (output tag counts as randomized log2 values - for scatter plots)
-sqrt (output tag counts as randomized sqrt values - for scatter plots)
-start <#> (start counting tags relative # offset of beginning of gene)
-end <#> (finish counting tags relative # offset to end of the gene)
-pausing <#> (calculate ratio of pausing first [# bp of transcript] to gene body)
Produces 3 columns - promoter rpk, body rpk, and ratio (add -log for log versions)
Also sets "-count genes". Use "-strand both" when analyzing Pol II ChIP-Seq

rpk is reads per kb - set -norm 1e6 or -normMatrix 1e6 to get rpkm

Can't figure something out? Questions, comments, concerns, or other feedback:
cbenner@ucsd.edu

HOMER

Software for motif discovery and next-gen sequencing analysis

---

homerTools - General sequence manipulation

homerTools is a utility program Chuck uses for basic sequence manipulation of FASTQ files, extracting sequences from genome FASTA files, and calculating nucleotide frequencies. To run **homerTools** do the following:

> **homerTools [command] [command specific options]**
>
> i.e. **homerTools trim -3 AAAAAAAA s_1_sequence.txt**

The following commands are available in **homerTools**:

> **barcodes** - for separating and removing 5' barcodes from FASTQ/FASTA files
> **trim** - for trimming by adapter sequence, specific lengths, etc. from FASTQ/FASTA files
> **freq** - for calculating nucleotide frequencies in FASTQ/FASTA/txt sequence files
> **extract** - for extracting specific regions of seqeuence from genomic FASTA files

Separating 5' Barcodes:

To separate and remove 5' barcodes from sequencing data (where the first "x" base pairs of the read are the barcode):

> **homerTools barcodes <# length of barcode> [options] <sequence file1> [sequence file2] ...**
>
> i.e. **homerTools barcodes 3 s_1_sequence.txt** (removes first 3bp as the barcode and sorts the reads by barcode)

The 3rd argument must be the length of the 5' barcode, which will be the first base pairs in the sequence. By default, this command creates files named "filename.barcode", such as s_1_sequence.txt.AAA, s_1_sequence.txt.AAC, s_1_sequence.txt.AAG etc. The parameter "**-min <#>**" specifies the minimum barcode frequence to keep (default is 0.02 [2%]). The frequency of each barcode is recorded in the output file "filename.freq.txt". If important barcodes were deleted, rerun the command with a smaller value for "**-min <#>**".

Trimming Sequence Files

With all the fancy types of sequencing being done, it is getting common to find adapters as part of the sequences that are analyzed. The trim command allows users to trim sequences from the 3' and 5' ends by either a specific number of nucleotides or remove a specific adapter sequence. The basic command is executed like this:

homerTools trim [options] <sequence file1> [sequence file2]

The output will be placed in files "filename.trimmed" and the distribution of sequence lengths after trimming will be in "filename.lengths" for each of the input files. The following options control how homerTools trims the sequences:

-len <#> (trim sequences to this length)
-min <#> (remove sequence that are shorter than this after timming)
-3 <#> (trim this many bp off the 3' end of the sequence)
-5 <#> (trim this many bp off the 5' end of the sequence)
-3 <ACGT> (trim adapter sequence (i.e. "-3 GGAGGATTT") from the 3' end of the sequence)
-5 <ACGT> (trim adapter sequence (i.e. "-5 GGAGGATTT") from the 5' end of the sequence)

For adapter sequence trimming, it will search for the first full match to the sequence and delete the rest of the sequence. For example if you specify "-3 AA", it will search for the first instance of "AA" and delete everything after it. It will also delete partial matches if they are at the end of the sequence (or beginning for 5'). As another example, our lab uses an amplification strategy for RNA that results in the ligation of a polyA tail to the RNA sequence. If the reads are long enough, the read will be just As.

i.e. GAGATTATCTACGTACCGAAAAAAAAAAAAAAAA

Trimming with "**-3 AAAAAAAAA**" will cleave the complete polyA stretch.

In this example: GAGATTATCTACGTACCGTACTGCATGACGGGAAAA, only the final 4 As would be trimmed.

Extracting Genomic Sequences From FASTA Files

The **extract** command can be used to extract large numbers of specific genomic sequence. The first input file you need is a HOMER style peak file or a BED file with genomic locations. Next, you must have the genomic DNA sequences in one of two formats: (1) a directory of chr1.fa, chr2.fa FASTA files (can be masked file like *.fa.masked), or (2) a single file FASTA file with all of the chromosomes concatonated in one file. The sequences are sent to *stdout* as a tab-delimited file, or as a FASTA formatted file if "**-fa**" is added to the end of the command. Save the output to a file by adding " **> outputfile.txt**" to the end of the command. The program is run like this:

homerTools extract <peak/BED file> <FASTA directory or file location> [-fa]

i.e. **homerTools extract peaks.bed /home/chucknorris/homer/data/genomes/mm9/ > outputSequences.txt**
Or, to get FASTA files back, i.e. **homerTools extract peaks.bed /home/chucknorris/homer/data/genomes/mm9/ -fa > outputSequence.fa**

Calculating Nucleotide Frequencies

The **freq** command will calculate nucleotide frequencies from FASTQ, FASTA, or tab-delimited text sequence files. The program tries to auto detect the format, but it may help to specify the format directly ("**-format fastq**", "**-format fasta**", "**-format tsv**"). The program outputs a position-dependent nucleotide/dinucleotide frequency file as a function of the distance from the start of the sequencing reads. The output is sent to *stdout*, unless you specify "**-o <outputfile.txt>**". If you specify "**-gc <outpufile2.txt>**", the program will also create a file that specifies the cumulative frequency of CpG, total G+C, total A+G, and total A+C in each individual sequence.

homerTools freq -format fastq s_1_sequence.txt > s_1.frequency.txt
homerTools freq -format fastq s_1_sequence.txt -gc GCdistribution.txt -o positionFrequency.txt

homerTools Command Line options:

Usage: homerTools <command> [--help | options]

Collection of tools for sequence manipulation

Commands: [type "homerTools <command>" to see individual command options]
barcodes - separate FASTQ file by barcodes
trim - trim adapter sequences or fixed sizes from FASTQ files(also splits)
freq - calculate position-dependent nucleotide/dinucleotide frequencies
extract - extract specific sequences from FASTA file(s)
decontaminate - remove bad tags from a contaminated tag directory
cluster - hierarchical clustering of a NxN distance matrix
special - specialized routines (i.e. only really useful for chuck)

Options for command: barcode
-min <#> (Minimum frequency of barcodes to keep: default=0.020
-freq <filename> (output file for barcode frequencies, default=file.freq.txt)
-qual <#> (Minimum quality score for barcode nucleotides, default=not used)
-qualBase <character> (Minimum quality character in FASTQ file, default=B)

Options for command: trim

-3 <#|[ACGT]> (trim # bp or adapter sequence from 3' end of sequences)
-5 <#|[ACGT]> (trim # bp or adapter sequence from 5' end of sequences)
-mis <#> (Maximum allowed mismatches in adapter sequence, default: 0)
-minMatchLength <#> (minimum adapter sequence at edge to match, default: half adapter length)
-len <#> (Keep first # bp of sequence - i.e. make them the same length)
-stats <filename> (Output trimming statistics to filename, default: sent to stdout)
-min <#> (Minimum size of trimmed sequence to keep, default: 1)
-max <#> (Maximum read length, default: 100000)
-suffix <filename suffix> (output is sent to InuptFileName.suffix, default: trimmed)
-lenSuffix <filename suffix> (length distribution is sent to InuptFileName.suffix, default: lengths)
-split <#> (Split reads into two reads at bp #, output to trimmed1 and trimmed2)
-revopp <#> (Return reverse opposite of read [if used with -split, only the 2nd half of the read will be retuned as reverse opposite])

Options for command: freq
-format <tsv|fasta|fastq> (sequence file format, default: auto detect)
-offset <#> (offset of first base in output file, default: 0)
-maxlen <#> (Maximum length of sequences to consider, default: length of 1st seq)
-o <filename> (Output filename, default: output sent to stdout)
-gc <filename> (calculate CpG/GC content per sequence output to "filename")
OutputFormat: name<tab>CpG<tab>GC<tab>AG<tab>AC<tab>Length

Options for command: extract
-fa (output sequences in FASTA format - default is tab-delimited format)

Alternate Usage: homerTools extract stats <Directory of FASTA files>
Displays stats about the genome files (such as length)

Options for command: decontaminate
-frac <#> (Estimate fraction of sample that is contaminated, default: auto)
-estimateOnly (Only estimate the contamination, do not decontaminate)
-o <output tag directory> (default: overrites contaminated tag directory)
-size <#> (Peak size for estimating contamination/Max distance from contaminant
reads to remove contaminated reads, default: 250)
-min <#> (Minimum tag count to consider when estimating contamination, default: 20)

Can't figure something out? Questions, comments, concerns, or other feedback: cbenner@ucsd.edu

HOMER

Software for motif discovery and next-gen sequencing analysis

---

Miscellaneous Tools for Sharing Data

HOMER contains several utility programs for changing file formats and performing tasks that you may find useful from time to time. Below are a list of programs that may come in handy.

Outputing Tag Directory as a BED file (tagDir2bed.pl):

This is useful when you want to share your sequencing with others as BED format is about the most general format there is out there.

tagDir2bed.pl <tag directory> > output.alignment.file.bed

i.e. **tagDir2bed.pl Macrophage-PU.1-ChipSeq/ > mac.pu1.bed**

This will produce a large BED file that can be used to import the data to other programs.

Coverting between HOMER peak and BED file formats (pos2bed.pl / bed2pos.pl):

Want to load HOMER peaks into the genome browser? Or use them with other software?

Covert a HOMER peak/position file to a BED file:

pos2bed.pl <peak file> > output.bed

Covert a BED file to a HOMER peak/position file:

bed2pos.pl <BED file> > output.peakfile.txt

Can't figure something out? Questions, comments, concerns, or other feedback: cbenner@ucsd.edu

HOMER

Software for motif discovery and ChIP-Seq analysis

Finding Overlapping and Differentially Bound Peaks

HOMER provides a utility for comparing sets of peaks called **mergePeaks**. It's default behavior is to take two or more peak files and return a single peak file containing the unique peak positions from the original files. For example:

mergePeaks -d <maximum distance to merge> <peak file1> <peak file2> [peak file3] ... > newPeakFile.txt

The program will output a new peak file containing the merged peaks to *stdout*. Peaks within the distance in bp specified by "**-d <#>**" will be reported as the average position between the peaks found within the common region (default=100 bp, good for transcription factors). The origin of the peaks is specified in the 7th column of the new peak file. Alternatively you can specify "**-d given**" to require a specific overlap between the start and end coordinates of the peaks. This is more useful if comparing large regions as opposed to peaks. The program will also output the numbers for creating a venn diagram, and these can be directed to a specific file by specifying "**-venn <filename>**".

Separating Peaks into Unique and Overlapping sets

Merging peaks together into a single file is very useful for certain types of analysis, such as making scatter plots that compare the tag-densities between peaks from separate experiments - in this case you want to count tags at specific and common regions. Alternatively, you may be interested in separating the peaks into common and specific sets for focused analysis. To do this use the "**-prefix <filename>**" option - this will create separate files based on overlapping peaks for each set of peaks. For example:

mergePeaks -d 100 pu1.peaks cebp.peaks -prefix mmm

This will create files named "**mmm_pu1.peaks**", "**mmm_cebp.peaks**", and "**mmm_pu1.peaks_cebp.peaks**".

The output file will contain the following columns:

1. Merged Peak name (will start with "Merged-")
2. chromosome
3. start (average from merged peaks)
4. end (average from merged peaks)

5. strand
6. Average peak score (actually, the average of the original values in column 6 of the peak files - or column 5 of BED files)
7. Original peak files contributing to the merged peak
8. Total number of peaks merged (occasionally more than one peak from a single file will be merged if the peaks are within the specify distance or two or more peaks from one file overlap with the same single peak(s) from another file)

Peak Co-Occurrence Statistics

The **mergePeaks** program will also find calculate the statistics of co-occurrence between peaks in a pairwise fashion. If "**-matrix <filename>**" is specified, HOMER will calculate statistics about the pairwise overlap of peaks. Three separate pairwise matrix files will be produced using the supplied <filename> as a prefix:

filename.logPvalue.matrix.txt (natural log p-values for overlap using the hypergeometric distribution, positive values signify divergence)
filename.logRatio.matrix.txt (natural log of the ratio of observed overlapping peaks to the expected number of overlapping peaks)
filename.count.matrix.txt (raw counts of overlapping peaks)

The statistics are dependent on the effective size of the genome, which can be specified using "**-gsize <#>**" (default: 2,000,000,000)

Co-Bound Peaks

Sometimes you just want to know how many other peaks bind a set of reference peaks. If "**-cobound <#>**", mergePeaks counts how many of the other peak files contain overlapping peaks with the peaks found in the first peak file. It then outputs peak files named "coboundBy0.txt", "coboundBy1.txt", etc. up to the number specified.

Differentially Bound Peaks

To find peaks that are differentially enriched between two experiments, there are two basic options. First, you could run **findPeaks** (info here) using the 2nd experiment as the control sample. Alternatively, you can use **getDifferentialPeaks**, which will take a given list of peaks and quickly identify which peaks contain significantly more tags in the target experiment relative to the background experiment. To use it, follow this syntax:

getDifferentialPeaks <peak/BED file> <target Tag directory> <background Tag directory> [options]

By default it looks for peaks that have 4-fold more tags (sequencing-depth

independent) and a cumulative Poisson p-value less than 0.0001 (sequencing-depth dependent). These parameters are adjustable with ("**-F <#>**", and "**-P <#>**"). By specifying "**-same**", peaks that are similar between the two tag directories will be returned instead of differential peaks. One caveat is that it is a good idea to set the size of the region used to search for reads to be larger than the actual peaks (i.e. +100 bp relative to the peak size) to avoid problems that arise from experiments with different fragment lengths, etc.

Command Line options for mergePeaks

Usage: mergePeaks [options] <primary peak file> [additional peak/annotation files...]

Merges and/or compares peak/position files (peak files listed twice are only considered once)

General Options:
-strand (Only merge/consider peaks on the same strand, default: either strand)
-d <#|given> (Maximum distance between peak centers to merge, default: 100)
Using "-d given" looks for literal overlaps in peak regions
Use "-d given" when features have vastly different sizes (i.e. peaks vs. introns)
-file <filename> (file listing peak files to compare - for lots of peak files)
-gsize <#> (Genome size for significance calculations, default: 2e9)

Merging Peaks Options (default):
-prefix <filename> (Generates separate files for overlapping and unique peaks)
By default all peaks are sent to stdout
-matrix <filename> (Generates files with pairwise comparison statistics)
filename.logPvalue.matrix.txt - ln p-values for overlap, +values for divergence
filename.logRatio.matrix.txt - ln ratio of observed/expected overlaps
filename.count.matrix.txt - peak overlap counts
-venn <filename> (output venn diagram numbers to file, default: to stderr)
-code (report peak membership as binary instead of by file names)

Classify peaks by how many are co-bound by other peak files vs. reference(1st file)
-cobound <#> (Maximum number of co-bound peaks to consider)
Will output sets of peaks that are co-bound by various numbers of factors
to files coBoundBy0.txt, coBoundBy1.txt, coboundBy2.txt, ...
Or <prefix>.coBoundBy0.txt, <prefix>.coBoundBy1.txt, ...
-matrix <filename> (generates similar files to above with pairwise overlap statistics)

Single peak file:
(If a single peak file is given, peaks within the maximum distance will be merged)
-filter chrN:XXX-YYY (only analyze peaks within range)
-coverage <output file> (returns the total bp covered by each peak file - use "-d given"

Command Line options for getDifferentialPeaks

Usage: getDifferentialPeaks <peak file> <target tag directory> <background tag directory> [options]

Extracts tags near each peak from the tag directories and counts them,
outputting peaks with significantly different tag densities

General Options:
-F <#> (fold enrichment over background tag count, default: 4.0)
-P <#> (poisson enrichment p-value over background tag count, default: 0.0001)
-same (return similar peaks instead of different peaks)
-rev (return peaks with higher tag counts in background instead of target library)
-size <#> (size of region around peak to count tags, default: -fixed)
-fixed (Count tags relative to actual peak start and stop, default)

Output Options:
-strand <both|+|-> (Strand [relative to peak] to count tags from, default:both)
-tagAdjust <#> (bp to shift tag positions to estimate fragment centers, default: auto)
'-tagAdjust auto' uses half of the estimated tag fragment length
-tagAdjustBg <#> (bp to shift background tag positions to estimate fragment centers, default: auto)
'-tagAdjustBg auto' uses half of the estimated tag fragment length
-tbp <#> (Maximum tags per bp to count, 0 = no limit, default: 0)
-tbpBg <#> (Maximum background tags per bp to count, 0 = no limit, default: 0)

Can't figure something out? Questions, comments, concerns, or other feedback: cbenner@ucsd.edu

HOMER

Software for motif discovery and next-gen sequecing analysis

ChIP-Seq Analysis: Analyzing a ChIP-Seq experiment with one command

Even I don't like typing the same commands over and over again. The following command performs the standard set of analysis commands so that you can do better things while your data is processed.

> **analyzeChIP-Seq.pl <Tag Directory> <genome> [general options] [-A | B | C | D sub-program options]**
>
> i.e. a common use: **analyzeChIP-Seq.pl Factor-ChIP-Seq/ hg18r -i Input-ChIP-Seq -focus -A factor_alignment_file.bed factor_alignment_file2.bed**

This command performs 4 separate tasks labeled as A,B,C & D:

> A. Runs **makeTagDirectory** to parse alignment files, set up the tag directory, and performs basic QC such as tag auto correlation and checks for sequence bias.
> B. Runs make **makeUCSCfile** and **findPeaks** to generate UCSC Genome Browser files and peak files for the experiment.
> C. Runs **findMotifsGenome.pl** to determine enriched motifs in your ChIP-Seq peaks.
> D. Runs **annotatePeaks.pl** to generate an annotated peak file and performs GO analysis on genes found near the peaks.

As output, this program will create standard files in the "Tag Directory" including an "index.html" file that links you to each of the output files.

There are a couple of general options that can be used with **analyzeChIP-Seq.pl**:

> -i <input tag directory> : "Tag directory" to use as a control for peak finding.
> -size <#> : Force this peak size for analysis (default is "auto" for peak finding, 200 bp for motif analysis and 50 bp for focused peak analysis)
> -focus : This will find enriched motifs in 85% focused peaks using only +/- 25 bp of sequence, useful for identifying the primary motif bound by the factor.
> -enhancer : This will set the peak size to "-size 1000" and only perform motif analysis on peaks > 3kb from the TSS.

To specifically tailor the options used by the sub-programs, first enter the "general options" you want above, then enter "**-A**" followed by the options you want passed to the sub-programs. For example, the most used sub-program option I use is the

following:

> "-A s_1_eland_result.txt" - this passes the alignment file to the **makeTagDirectory** program so that it will be used to make the Tag Directory.

If you've already made a tag directory, no need to use the "-A blah blah" option. In similar fashion, to tell the motif finding to check motifs of length 10, 11, and 12, add "**-C -len 10,11,12**" to the end of the command.

Back to ChIP-Seq Analysis

Can't figure something out? Questions, comments, concerns, or other feedback: cbenner@ucsd.edu

HOMER

Software for motif discovery and ChIP-Seq analysis

File Formats

List of files used by HOMER - might be helpful when encountering problems.
Another good resource on file formats: UCSC Genome Browser File Formats

Peak/Positions files

These files specify genomic locations similar to BED files. They are tab-delimited text files with a minimum of 5 columns (additional columns are ignored). They are 1-indexed and inclusive, meaning the first nucleotide of a chromosome is referenced as position 1. They are inclusive in the sense that a line with a start of 100 and end of 200 indicates of region of size 101. Columns are as followed:

1. peak name (should be unique)
2. chromsome
3. starting position [integer] (1-indexed)
4. end position [integer]
5. strand [either 0/1 or +/-] (in HOMER strand of 0 is +, 1 is -)
6. Optional/ignored ...
...

Peak/Position files are very similar to BED files - to convert them use **pos2bed.pl** or **bed2pos.pl**.

BED files

These are essentially the same as Peak/Position files, except that they have a stricter definition but greater portability. They are also tab-delimited text files - the important difference is that they are 0-indexed, meaning the first nucleotide of the chromosome is referenced as position 0.

1. chromosome
2. starting position [integer] (0-indexed)
3. ending position [integer]
4. peak name
5. value (usually ignored)
6. strand [+/-]

BED files also come in a short form:

1. chromosome

2. starting position [integer] (0-indexed)
3. ending position [integer]
4. strand [+/-]

Peak/Position files are very similar to BED files - to convert them use **pos2bed.pl** or **bed2pos.pl**.

Motif files

These are files for specifying motifs, and are created by HOMER during motif discovery. They are tab-delimited text files. A more elaborate description of the format and how to tinker with it is here. Basically, each motif within the file contains a header row starting with a ">", followed by several rows with 4 columns, specifying the probabilities of each nucleotide at each position.

```
>ASTTCCTCTT    1-ASTTCCTCTT    8.059752      -23791.535714   0
T:17311.0(44 ...
0.726   0.002   0.170   0.103
0.002   0.494   0.354   0.151
0.016   0.017   0.014   0.954
0.005   0.006   0.027   0.963
0.002   0.995   0.002   0.002
0.002   0.989   0.008   0.002
0.004   0.311   0.148   0.538
0.002   0.757   0.233   0.009
0.276   0.153   0.030   0.542
0.189   0.214   0.055   0.543
```

The first row starts with a ">" followed by various information, and the other rows are the positions specific probabilities for each nucleotide (A/C/G/T). These values do not need to be between 0-1. HOMER will automatically normalize whatever values are there, so interger counts are ok. The header row is actually TAB delimited, and contains the following information:

1. **">" + Consensus sequence (not actually used for anything, can be blank) example: >ASTTCCTCTT**
2. **Motif name (should be unique if several motifs are in the same file) example: 1-ASTTCCTCTT or NFkB**
3. **Log odds detection threshold, used to determine bound vs. unbound sites (mandatory) example: 8.059752**
4. (optional) log P-value of enrichment, example: -23791.535714
5. (optional) 0 (A place holder for backward compatibility, used to describe "gapped" motifs in old version, turns out it wasn't very useful :)
6. (optional) Occurence Information separated by commas, example: T:17311.0(44.36%),B:2181.5(5.80%),P:1e-10317
    1. T:#(%) - number of target sequences with motif, % of total of total targets
    2. B:#(%) - number of background sequences with motif, % of total background
    3. P:# - final enrichment p-value
7. (optional) Motif statistics separated by commas, example:

Tpos:100.7,Tstd:32.6,Bpos:100.1,Bstd:64.6,StrandBias:0.0,Multiplicity:1.13

1. Tpos: average position of motif in target sequences (0 = start of sequences)
2. Tstd: standard deviation of position in target sequences
3. Bpos: average position of motif in background sequences (0 = start of sequences)
4. Bstd: standard deviation of position in background sequences
5. StrandBias: log ratio of + strand occurrences to - strand occurrences.
6. Multiplicity: The averge number of occurrences per sequence in sequences with 1 or more binding site.

Only the first 3 columns are needed. In fact, the rest of the columns are really just statistics from motif finding and aren't important when searching for instances of a motif.

The MOST IMPORTANT value is the 3rd column - this sets the detection threshold, which specifies whether a given sequence is enough of a "match" to be considered recognized by the motif. More on that below.

Internal File Formats:

These are files that you normally won't modify or play with, but in case your interested...

*.tags.tsv files

These are files used to store sequencing data in HOMER tag directories. They are tab-delimited text files that are sorted to allow for relatively quick access and processing.

1. blank (can be used for a name)
2. chromsome
3. position (1-indexed)
4. strand (0 or 1, +/- not allowed here)
5. Number of reads (can be fractional)
6. length of the read (optional)

Can't figure something out? Questions, comments, concerns, or other feedback: cbenner@ucsd.edu