

BEDTools– Genome Arithmetic

Biol4230 Thurs, March 22, 2018
Bill Pearson wrp@virginia.edu 4-2818 Pinn 6-057

- Borrowed from Aaron Quinlan, BEDTools author
- Genome file types/formats
 - format types
 - BED files
- Overview of genome features
- Genome arithmetic
 - approaches
 - UCSC "binning" algorithm
- Introduction to BEDtools
 - intersection / window / many more

fasta.bioch.virginia.edu/biol4230

1

To learn more:

File formats:

1. <http://genome.ucsc.edu/FAQ/FAQformat>
2. Mills (2014) *Curr. Protoc. Bioinform.* 45:A.1B.1-A.1B.18

BEDTools:

1. <http://bedtools.readthedocs.org>
2. Quinlan, A. R. BEDTools: The Swiss-Army Tool for Genome Feature Analysis. *Curr Protoc Bioinformatics* **47**, 11.12.1–11.12.34 (2014).
3. Galaxy Screencasts: <http://main.g2.bx.psu.edu/>
4. Download genome features from UCSC and use BEDTools on the command line

fasta.bioch.virginia.edu/biol4230

2

Analysis of transcription regulation

1. Affy-chip/RNA-seq for RNA expression levels
2. Identify differentially expressed genes (edgeR, DESeq2)
 - (possibly) identify subsets of genes associated with different pathways
3. Isolate regions of DNA around (pathway-specific) coordinately expressed (induced) genes
 - BEDtools
4. Take collections of candidate regulatory regions and look for common DNA motif
 - meme, HOMER

fasta.bioch.virginia.edu/biol4230

3

Genome file formats:

Data	Alignment	Feature
GenBank	SAM/BAM	GFF2/GTF
EMBL	Axt	GFF3
FASTA	MAF	VCF
FASTQ	Chain	BED/bigBED
	Stockholm	bedGraph
		WIG/bigWIG

Mills (2014) Current Prot. in Bioinfo.
Appendix A.1B.1-A.1B.18

fasta.bioch.virginia.edu/biol4230

4

Genome data file formats:

- **FASTA:**

```
>sp|P09488|GSTM1_HUMAN  
MPMILGYWDIRGLAHAIRLLEYTDSSYYEKKYTMGDAPDYDRSQWLNEKFKLGLDFPNLPYLI  
DGAHKITQSNAILCYIARKHNLCGETEEEKIRVDILENQTMDNHMQLGMICYNPEF
```

- **FASTQ (FASTA with quality scores)**

```
@UNC10-SN254_238:4:1101:1351:51174/1  
CTTCGCGGTAGCTGGACCGCCGTTAGTCGCNAATANGCNGCTCTNTGT  
+  
B@CCCCFFDFHHHJJJJJJJJ@FHJGJJ#####  
@UNC10-SN254_238:4:1101:1351:7840/1  
ATTTTTTACATGGAGCAGGAACTGGAGTAAATGCAANACNGTGTNTAA  
+  
CCCCFFFFHHHHHJIIIIJIJIIGIIJHIGGHIIIHIG#####
```

Mills (2014) Current Prot. in Bioinfo.
Appendix A.1B.1-A.1B.18

fasta.bioch.virginia.edu/biol4230

5

Genome feature file formats:

- **GFF/GTF/GFF2:**

```
##gff-version 2  
##type Protein  
  
##sequence-region P09488 1 218  
P09488 UniProtKB mature_protein_region 2 218 . . . Note "Glutathione S-trans. Mu 1"  
P09488 UniProtKB polypeptide_domain 2 88 . . . Note "GST N-terminal"  
P09488 UniProtKB polypeptide_domain 90 208 . . . Note "GST C-terminal"  
P09488 UniProtKB polypeptide_region 7 8 . . . Note "Glutathione binding"  
P09488 UniProtKB polypeptide_region 46 50 . . . Note "Glutathione binding"  
P09488 UniProtKB polypeptide_region 59 60 . . . Note "Glutathione binding"  
P09488 UniProtKB polypeptide_region 72 73 . . . Note "Glutathione binding"  
P09488 UniProtKB binding_motif 116 116 . . . Note "Substrate"  
P09488 UniProtKB alt_seq_site 153 189 . . . Note "UniProtKB FT ID: VSP_036618"  
P09488 UniProtKB nat_variant_site 173 173 . . . Note "K -> N"  
P09488 UniProtKB nat_variant_site 210 210 . . . Note "S -> T" ; ...  
  
Note field[8] is can be very long, separated by ';' ;  
Note "S -> T" ; Note "UniProtKB FT ID: VAR_014497" ; Note "dbSNP:rs449856" ; Link  
"http://www.ncbi.nlm.nih.gov/SNP/snp_ref.cgi?type=rs&rs=449856" ; Link  
"http://www.ensembl.org/Homo_sapiens/Variation/Explore?v=rs449856"
```

Mills (2014) Current Prot. in Bioinfo.
Appendix A.1B.1-A.1B.18

fasta.bioch.virginia.edu/biol4230

6

Genome feature file formats: (BED)

chr	start 0-based	stop 1-based	name (len = stop - start)	score	strand
-----	------------------	-----------------	------------------------------	-------	--------

BED files:

```

chr1    110230417 110230531 NM_146421_exon_0_0_chr1_110230418_f    0      +
chr1    110230791 110230867 NM_146421_exon_1_0_chr1_110230792_f    0      +
chr1    110231294 110231359 NM_146421_exon_2_0_chr1_110231295_f    0      +
chr1    110231669 110231751 NM_146421_exon_3_0_chr1_110231670_f    0      +
chr1    110231846 110231947 NM_146421_exon_4_0_chr1_110231847_f    0      +
chr1    110232892 110232988 NM_146421_exon_5_0_chr1_110232893_f    0      +
chr1    110235827 110236367 NM_146421_exon_6_0_chr1_110235828_f    0      +
chr1    110230417 110230531 NM_000561_exon_0_0_chr1_110230418_f    0      +
chr1    110230791 110230867 NM_000561_exon_1_0_chr1_110230792_f    0      +
chr1    110231294 110231359 NM_000561_exon_2_0_chr1_110231295_f    0      +
chr1    110231669 110231751 NM_000561_exon_3_0_chr1_110231670_f    0      +
chr1    110231846 110231947 NM_000561_exon_4_0_chr1_110231847_f    0      +
chr1    110232892 110232988 NM_000561_exon_5_0_chr1_110232893_f    0      +
chr1    110233075 110233186 NM_000561_exon_6_0_chr1_110233076_f    0      +
chr1    110235827 110236367 NM_000561_exon_7_0_chr1_110235828_f    0      +

```

Mills (2014) Current Prot. in Bioinfo.
Appendix A.1B.1-A.1B.18

fasta.bioch.virginia.edu/biol4230

7

BED (Browser Extensible Data) format:

Column	Field	Description
1	Chromosome	Chr1, chr2... chrX or scaffold
2	Start	Zero based start of feature
3	Stop	One based end of a feature, not inclusive
4	Name	Name displayed in the genome browser
5	Score	0-100
6	Strand	“+” or “-”
7	thickStrand	Beginning of thick band in browser
8	thickEnd	End of thick band in browser
9	itemRGB	Colors the feature using RGB codes
10	blockCount	Number of block in the BED line
11	blockSizes	Comma separated list of block sizes
12	blockStarts	Comma separated list of block positions relative to the Start

Mills (2014) Current Prot. in Bioinfo.
Appendix A.1B.1-A.1B.18

fasta.bioch.virginia.edu/biol4230

8

Genome analysis pipelines



Initial read mapping

Feature (peak) finding

Mills (2014) Current Prot. in Bioinfo.
Appendix A.1B.1-A.1B.18

fasta.bioch.virginia.edu/biol4230

9

Genome data (UCSC table browser)

The screenshot shows the UCSC Table Browser interface. At the top, there is a navigation bar with links for Genomes, Genome Browser, Tools, Mirrors, Downloads, My Data, Help, and About Us. Below the navigation bar is a section titled "Table Browser". This section contains various input fields and controls:

- clade: Mammal
- genome: Human
- assembly: Dec. 2013 (GRCh38/hg38)
- group: Genes and Gene Predictions
- track: UCSC Genes
- table: knownGene
- region: chr1:109631271-109750270
- identifiers (names/acccessions): paste list, upload list
- filter: create
- intersection: create
- correlation: create
- output format: BED - browser extensible data
- Send output to: Galaxy, GREAT, GenomeSpace
- output file: (leave blank to keep output in browser)
- file type returned: plain text, gzip compressed
- get output, summary/statistics

To reset all user cart settings (including custom tracks), [click here](#).

fasta.bioch.virginia.edu/biol4230

10

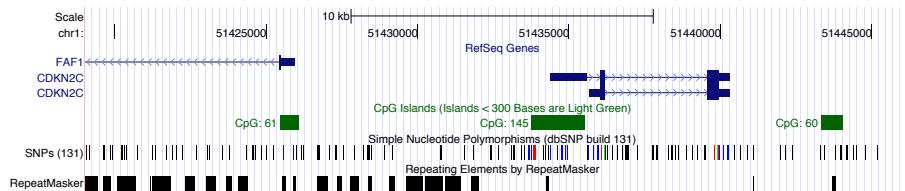
What is a genome “feature”?

- Genes: exons, introns, UTRs, promoters
- Conservation
- Genetic variation
- Transposons
- Origins of replication
- TF binding sites
- CpG islands
- Segmental duplications
- Sequence alignments
- Chromatin annotations
- Gene expression data
- ...
- Your own observations: put them in context

Genome “features”

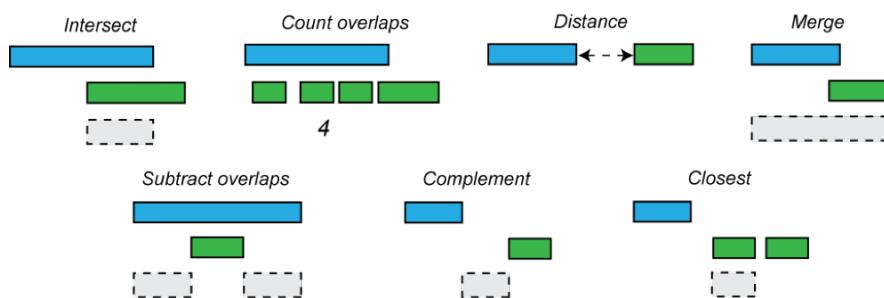


Overlaps, closest, merge...



What is genome arithmetic?

"Set theory on the genome"



Answerable questions

- Closest gene to a ChIP-seq peak.
- Is my latest discovery novel?
- Is there strand bias in my data?
- How many genes does this mutation affect?
- Where did I fail to collect sequence coverage?
- Is my favorite feature significantly correlated with some other feature?

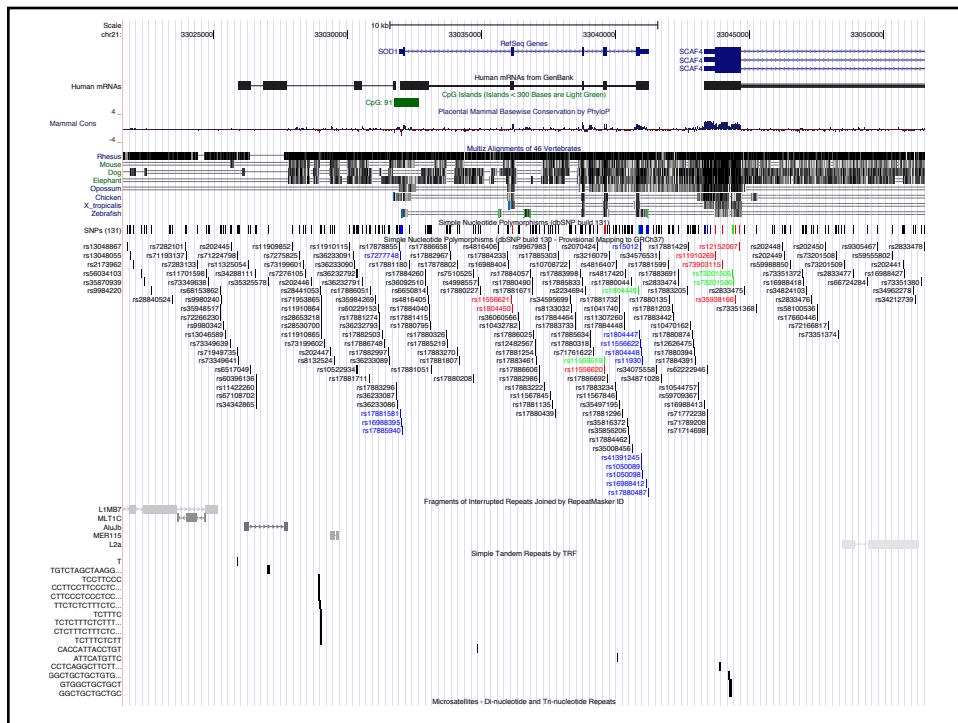
An annoying wealth of formats

- BED
- BEDGRAPH
- WIG
- GFF
- VCF
- SAM/BAM
- ...

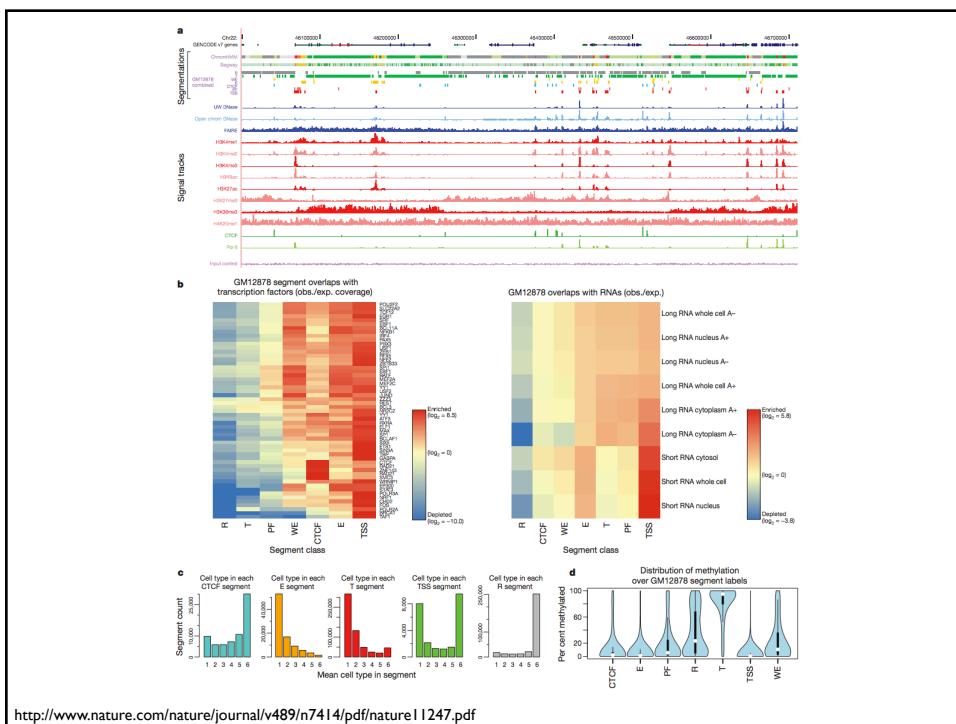
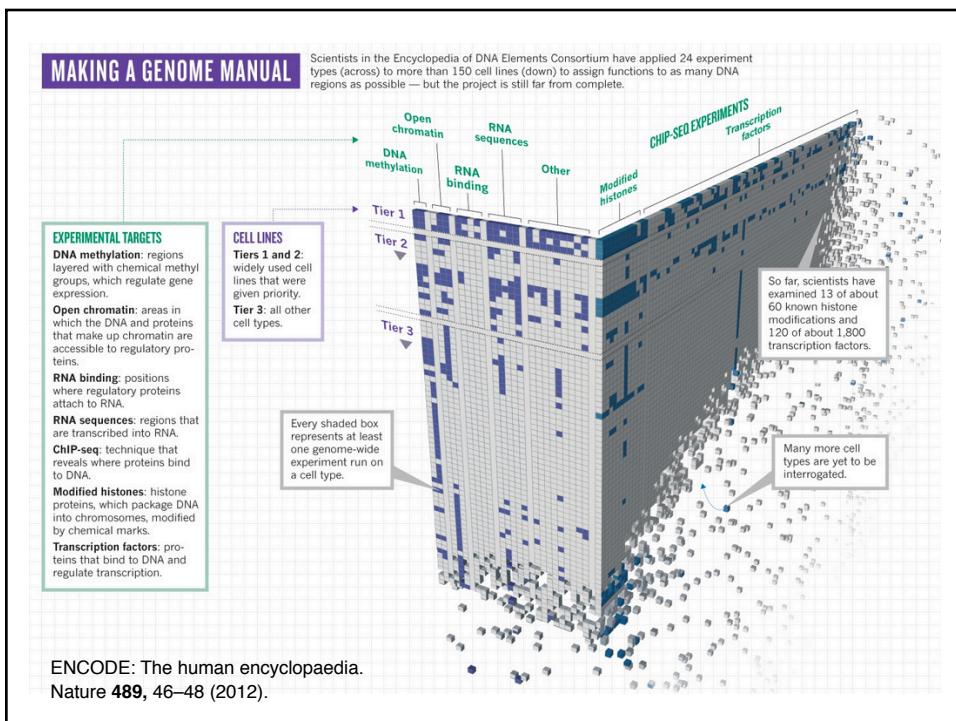
Standard attributes

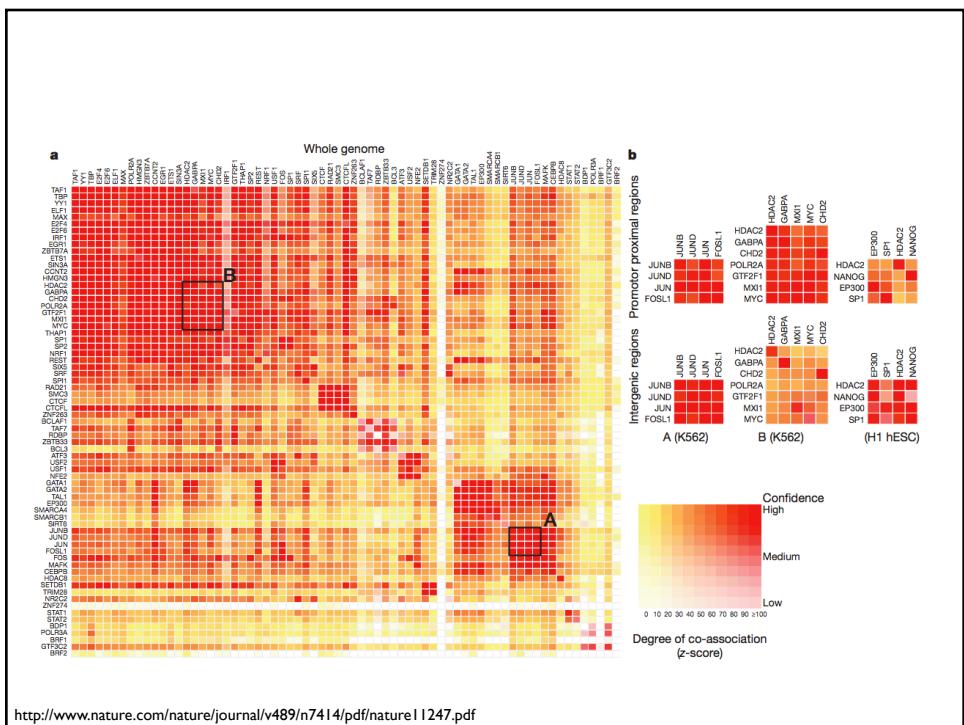
- chrom
- start position
- end position
- name / label
- score / value
- strand
- other

No standards: Some formats use 1-based coordinates, while others use 0-based



Why?





How?

Searching for overlaps

- Brute force: loop over all N features
 - Can be tragically slow when N is large
- Build a “tree”
 - R-tree
 - NCList
 - UCSC “binning” algorithm
- Extend the sort & “merge” algorithm
 - Widely used to “join” database table

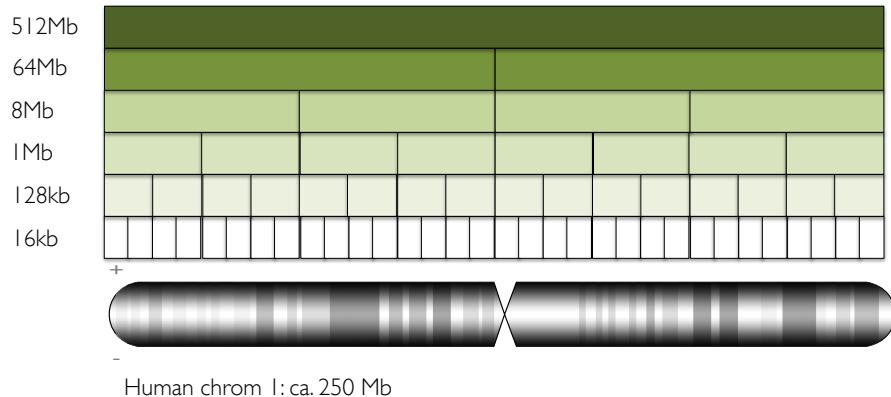
Brute force: sloooooow

```
foreach fA in file A
{
    foreach fB in file B
    {
        does fA overlap fB?*
    }
}
```

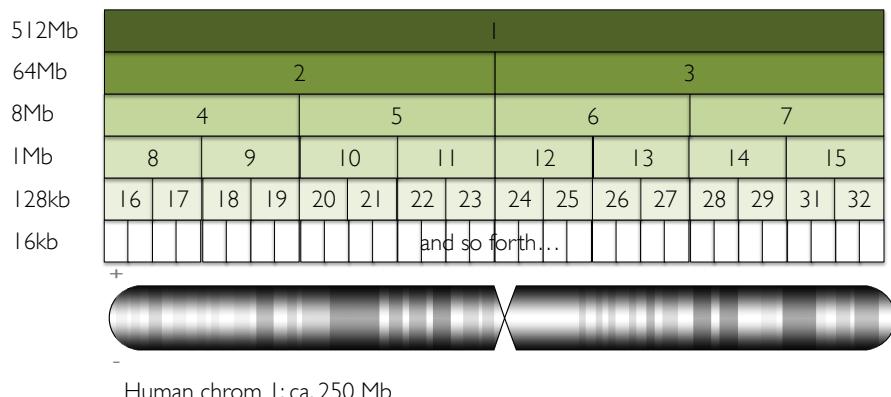
```
overlap.pl
1 #!/usr/bin/perl -w
2 use strict;
3
4 my $fileA;
5 my $fileB;
6 my $fileC = shift;
7
8 # attempt to open the files, will if trouble
9 open A, "<$fileA" or die $!;
10 open B, "<$fileB" or die $!;
11
12 my @files = files into arrays
13 my @A_features;
14 my @B_features = @B;
15
16 foreach my $a (@A_features)
17 {
18     my ($a_chrom, $a_start, $a_end) = split(/\t/, $a);
19     foreach my $b (@B_features)
20     {
21         my ($b_chrom, $b_start, $b_end) = split(/\t/, $b);
22         next if $a_chrom ne $b_chrom;
23         if ($a_start >= $b_start & $a_start <= $b_end > 0) {
24             chomp $a;
25             chomp $b;
26             print $a, "\t", $b, "\n";
27         }
28     }
29 }
30
31 sub overlap {
32     my ($id1, $id2, $id3, $id4) = @_;
33     my $min = min($id1, $id2, $id3, $id4);
34     my $max = max($id1, $id2, $id3, $id4);
35     if ($min == $max) {
36         return $min;
37     } else {
38         my $mid = int((($min + $max) / 2));
39         if ($mid == $min) {
40             return $mid;
41         } else {
42             my $mid2 = int((($mid + $max) / 2));
43             if ($mid2 == $mid) {
44                 return $mid;
45             } else {
46                 my $mid3 = int((($mid + $mid2) / 2));
47                 if ($mid3 == $mid2) {
48                     return $mid2;
49                 } else {
50                     my $mid4 = int((($mid3 + $max) / 2));
51                     if ($mid4 == $mid3) {
52                         return $mid3;
53                     } else {
54                         return $mid4;
55                     }
56                 }
57             }
58         }
59     }
60 }
```

* When A and B contain many features, the answer will be NO the vast majority of the time.
This means most of the processing is spent NOT finding features that overlap.

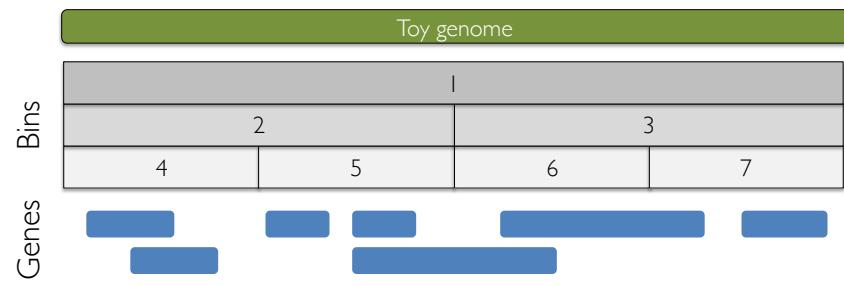
Narrow the search space w/ “binning” *the UCSC genome browser approach*



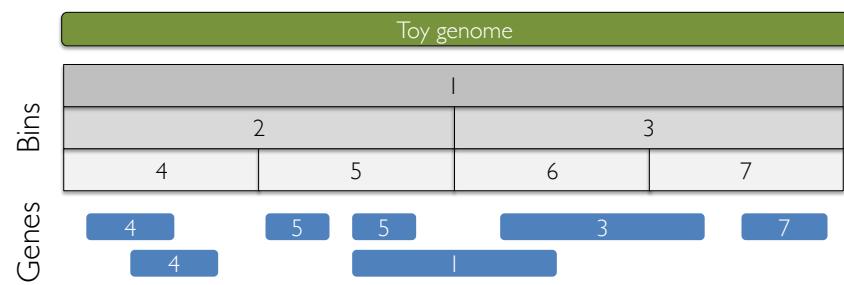
Narrow the search space w/ “binning” *the UCSC genome browser approach*



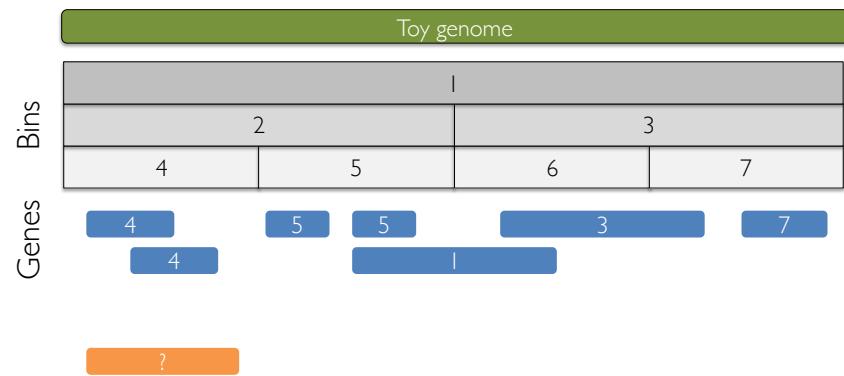
A toy example



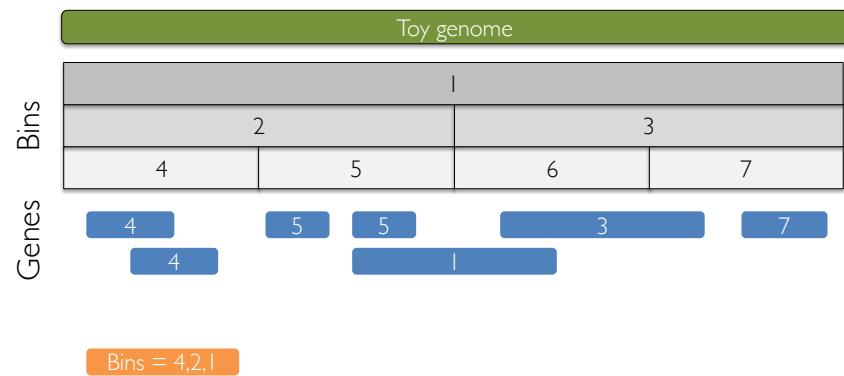
A toy example



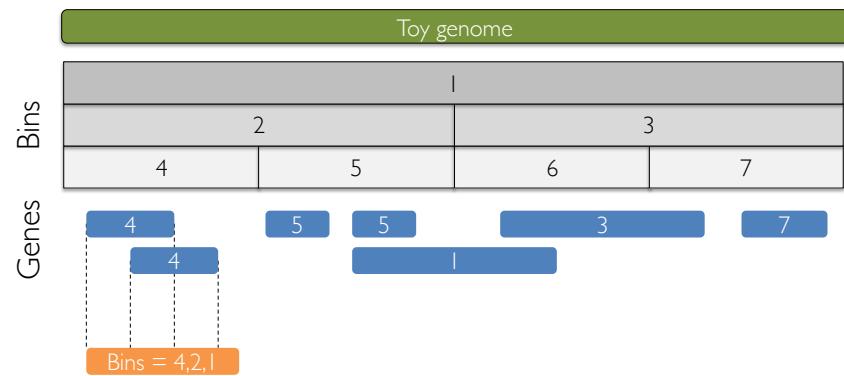
A toy example



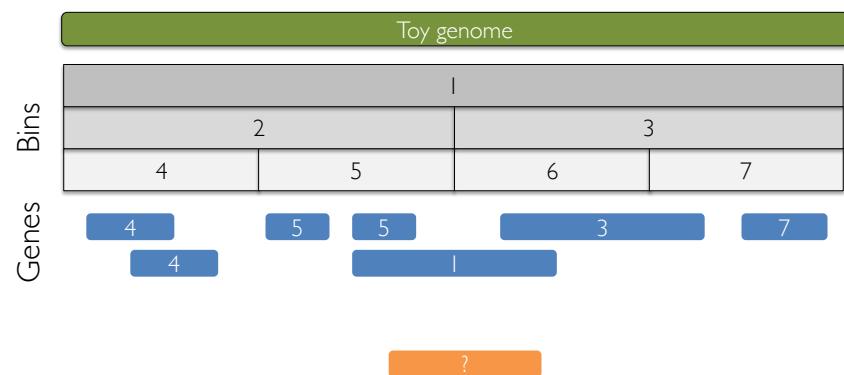
A toy example



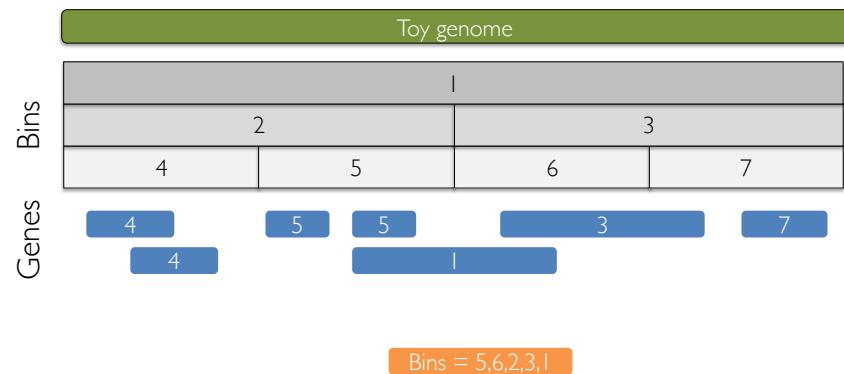
A toy example



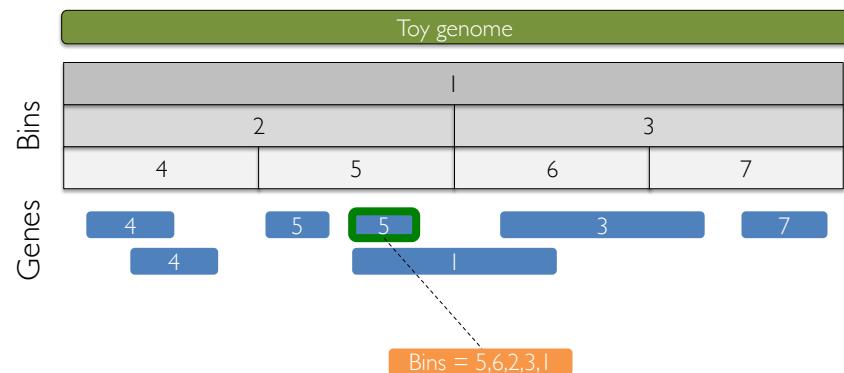
A toy example



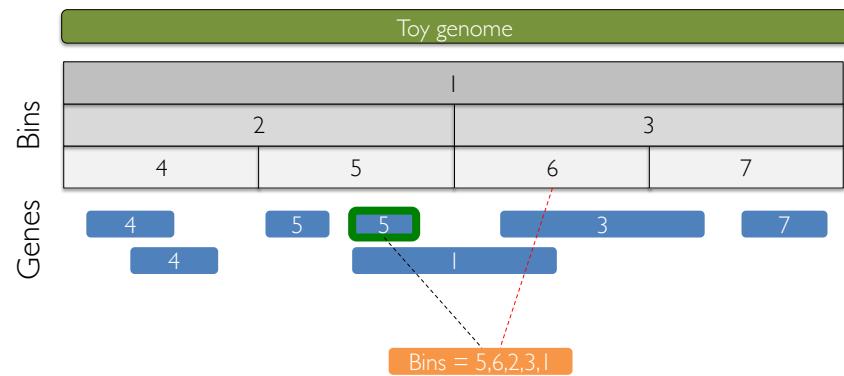
A toy example



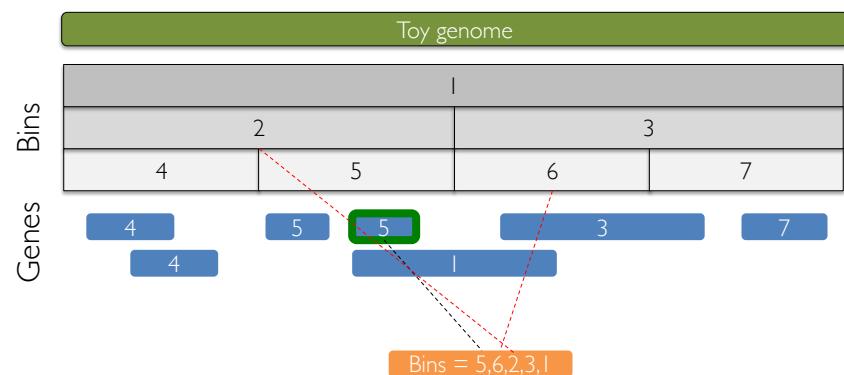
A toy example



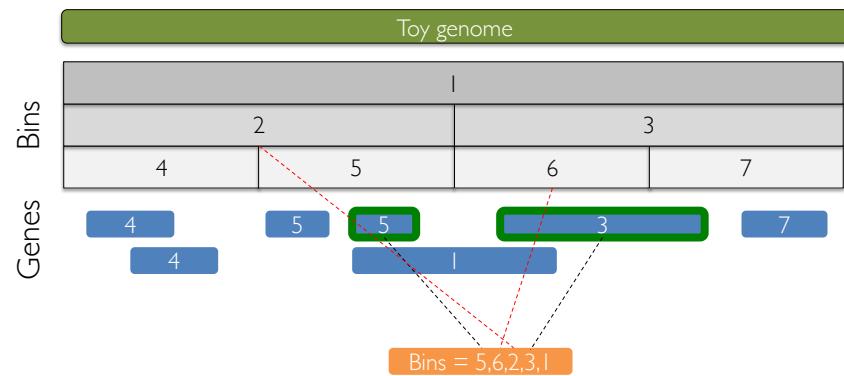
A toy example



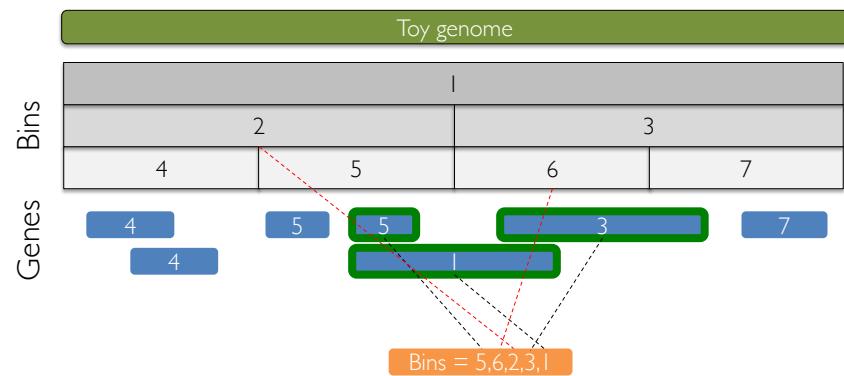
A toy example



A toy example



A toy example

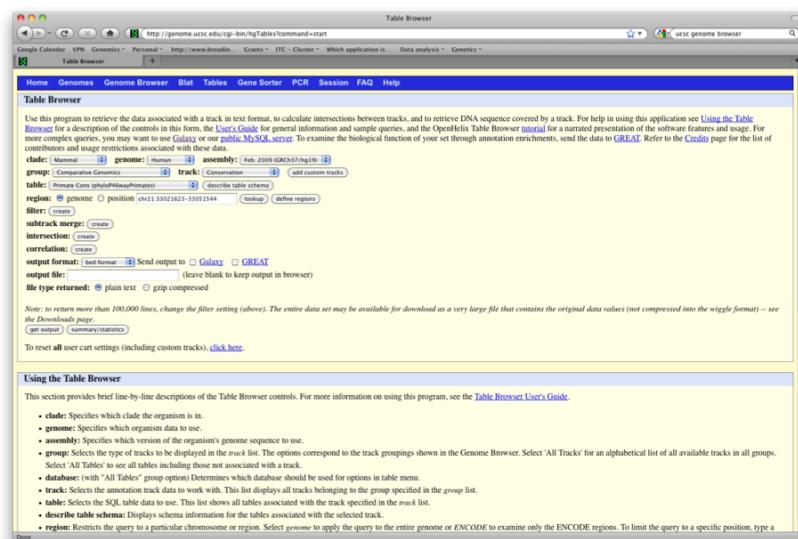


Available tools for GA

[UCSC Genome Browser](#)
[Galaxy](#)
[BEDTools](#)

UCSC Genome Browser

<http://genome.ucsc.edu/cgi-bin/hgTables?command=start>

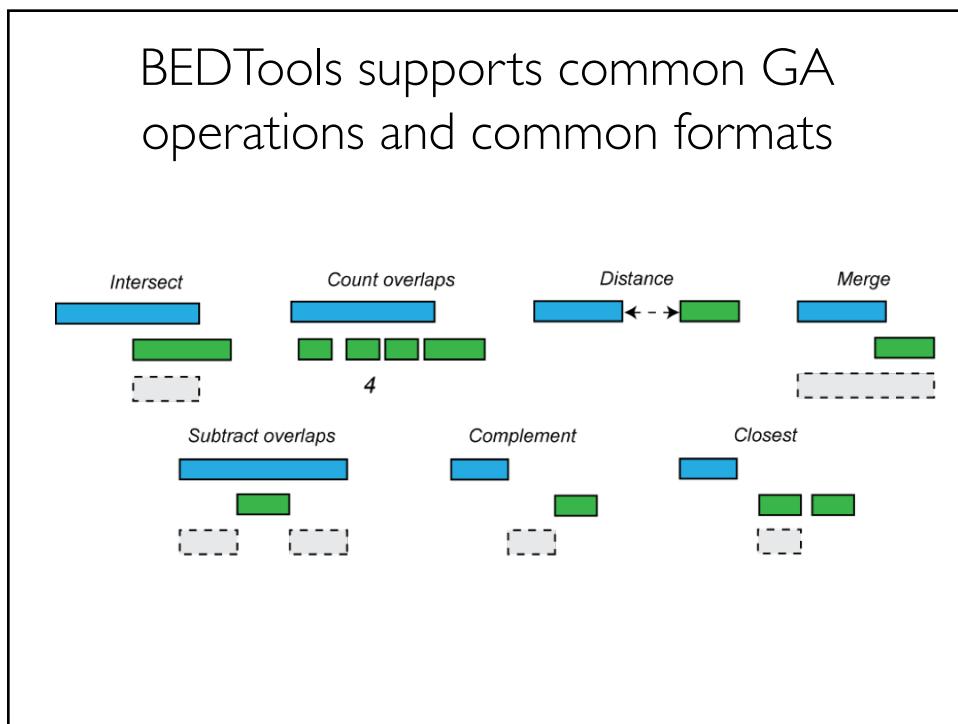


The screenshot shows the UCSC Genome Browser Table Browser interface. At the top, there's a navigation bar with links for Home, Genomes, Genome Browser, Blat, Tables, Gene Sorter, PCR, Session, FAQ, and Help. Below the navigation bar, the main form has fields for 'clade' (set to Mammal), 'genome' (set to Human), and 'assembly' (set to Feb 2009 GRCh37/hg19). Other fields include 'group' (set to Comparative Genomics), 'track' (set to Conservation), and 'table' (set to PrimeCons (phiHd4PmPrimates)). There are also fields for 'region' (set to genome position chr21:33021623-33051544), 'strand' (set to +), 'subtrack merge' (set to Create), 'intersection' (set to Create), 'correlation' (set to Create), and 'output format' (set to bed format). A checkbox for 'Send output to' is checked for Galaxy and GREAT. The 'output file' field is empty. The 'file type returned' dropdown is set to plain text. Below the form, there's a note about returning more than 100,000 lines and a link to click here. At the bottom, there's a section titled 'Using the Table Browser' with detailed descriptions of the controls.

BEDTools

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

The screenshot shows the GitHub repository page for BEDTools. The title "BEDTools" is at the top, followed by the URL "https://github.com/arq5x/bedtools2". Below the URL is a heading "bedtools - a swiss army knife for genome arithmetic". Underneath this, it says "Current version: 2.23.0". A "Note" section follows, stating that stable releases were formerly archived on Google Code but are now maintained via this GitHub repository. It also links to "Full documentation: http://bedtools.readthedocs.org". The "Summary" section describes BEDTools as a "swiss-army knife" of tools for genomic analysis, mentioning its ability to perform arithmetic operations like intersect, merge, and count overlaps on genomic intervals.



BEDTools + UNIX

Real world usage

Find SNPs that have the potential to alter gene expression regulation by affecting methylation at CpG islands.

(restrict to chrom 1)

Step 1: Get annotations

- Single-nucleotide polymorphisms (SNPs)
- CpG islands
- Genes

SNPs from dbSNP via UCSC

The terminal window shows the command:

```
cdn:~:10-5080 arq5$ wc -l snps.bed
```

The output of the command is:

```
204877 snps.bed
```

The 'head' command on the 'snps.bed' file shows the following data:

chrom	chromStart	chromEnd	name	score	strand	observed	func
chr1	10453	10452	r595260509	0	+	C/G	near-gene-5
chr1	10454	10453	r595260509	0	+	C/G	near-gene-5
chr1	10518	10519	r562630508	0	+	C/G	near-gene-5
chr1	10519	10520	r562630508	0	+	A/G	near-gene-5
chr1	10527	10528	r562630508	0	+	A/G	near-gene-5
chr1	10903	10904	rs102184933	0	+	A/G	near-gene-5
chr1	10904	10903	rs102184933	0	+	A/G	near-gene-5
chr1	10937	10938	rs102184937	0	+	A/G	near-gene-5
chr1	11001	11002	rs79537094	0	+	A/C	near-gene-5

Below the terminal window, the text "snps.bed" and "N = 2,064,872" is displayed.

CpG islands via UCSC

```
dm@B10-5080:~$ wc -l cpg.bed
2463 cpg.bed
dm@B10-5080:~$ head cpg.bed
#chrom chromStart chromEnd length cgNum
chr1 28735 29810 1075 116
chr1 32750 32823 73 10
chr1 32750 35829 439 29
chr1 437151 438164 1013 84
chr1 438164 439177 1034 84
chr1 53319 534114 995 94
chr1 544738 546649 1911 171
chr1 546649 548560 1871 66
chr1 763416 763445 1829 115
dm@B10-5080:~$
```

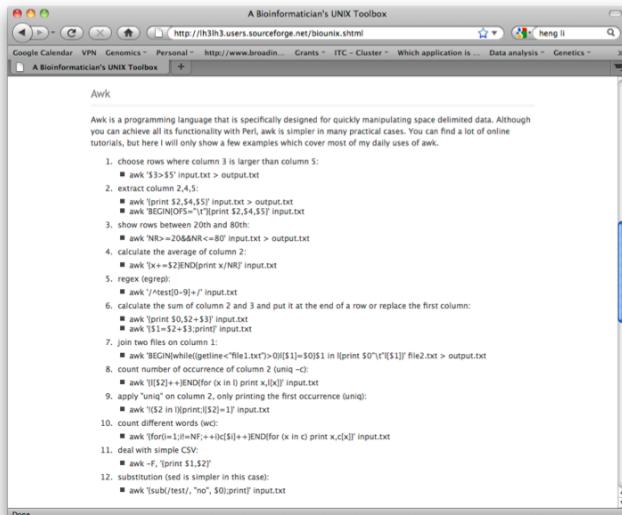
“cpg.bed”
N = 2,463

Problem: No name and no strand in the output. Let's fix it...

The awesome power of *awk*

- written in 1977 by Alfred Aho, Peter Weinberger, and Brian Kernighan
- Similar constructs as Perl (\$1, \$2, \$3, etc.)
- Typically used for filtering, summarizing, or reorganizing files.
- “One liners”, used on files or “streams”
- `awk '{print "Hello World!\n"}'` ([Print "HelloWorld!"](#))
- `awk '$2 >= 30' foo.txt` ([Get lines in foo where 2nd col is g.e. 30](#))
- `awk '{print $3,$2,$7,$1}' foo.txt > foo.reordered.txt`

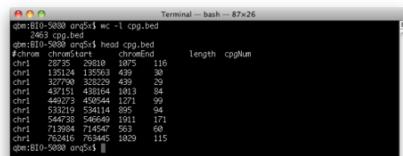
The awesome power of *awk*



A screenshot of a web browser window titled "A Bioinformatician's UNIX Toolbox". The page is titled "Awk" and contains a list of 12 numbered awk command examples. The examples demonstrate various operations such as filtering rows, extracting columns, calculating averages, and performing regex operations. The browser interface includes a back button, forward button, and a search bar at the top.

1. choose rows where column 3 is larger than column 5:
■ awk '\$3>\$5' input.txt > output.txt
2. extract column 2,4,5:
■ awk '{print \$2,\$4,\$5}' input.txt > output.txt
■ awk BEGIN{NR=1}{print \$2,\$4,\$5} input.txt
3. show rows between 20th and 80th:
■ awk 'NR==20&NR<=80' input.txt > output.txt
4. calculate the average of column 2:
■ awk '{x+= \$2}END{print x/NR}' input.txt
5. regex (egrep):
■ awk '/^test[0-9]+/' input.txt
6. calculate the sum of columns 2 and 3 and put it at the end of a row or replace the first column:
■ awk '{a=\$2+\$3}1' input.txt
■ awk '\$1=\$2+\$3' input.txt
7. join two files on column 1:
■ awk 'BEGIN{while((getline <"file1.txt")>0){\$0=\$1 in !(print \$0"\t"\$1)" file2.txt > output.txt}
8. count number of occurrence of column 2 (uniq -c):
■ awk '{if(\$2>1){END{for (x in c){print x,c[x]}}}' input.txt
9. apply "uniq" on column 2, only printing the first occurrence (uniq):
■ awk '{if(\$2 in !(print \$2=1'))}' input.txt
10. count different words (wc):
■ awk '{for(i=1;i<NF;i++)c[\$i]++}'\$1|END{for (x in c){print x,c[x]}}' input.txt
11. deal with simple CSV:
■ awk -F, '{print \$1,\$2}'
12. substitution (sed is simpler in this case):
■ awk '{sub(/test/, "no", \$0)};print' input.txt

Use awk to fix cpg.bed



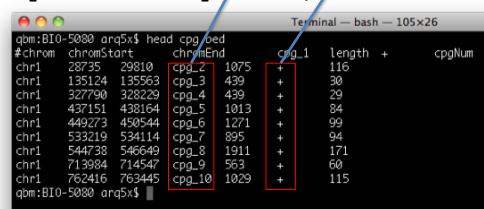
A screenshot of a terminal window titled "Terminal — bash — 87x26". The command "wc -l cpg.bed" is run, showing the file has 2659 lines. The command "head cpg.bed" is run, displaying the first few lines of the file. The file contains genomic data with columns for chromosome, start position, end position, strand, length, and CpG number.

```
qpm:BI0-5080 arq5x$ wc -l cpg.bed
2659 cpg.bed
qpm:BI0-5080 arq5x$ head cpg.bed
#chrom chromStart chromEnd      length  cpgNum
chr1  28735  29810  1075    116
chr1  135124 135563 439      +      30
chr1  327790 328229 439      +      29
chr1  437151 438164 1013    84
chr1  449273 450544 1271    99
chr1  533219 534114 895     94
chr1  544738 546649 1911    171
chr1  713984 714547 563     60
chr1  762416 763445 1029    115
qpm:BI0-5080 arq5x$
```

Problem: No name and no strand in the output. Let's fix it..

awk '{OFS="\t"; print \$1,\$2,\$3,"cpg_"NR,\$4,"+",\$5}' cpg.bed > cpg.full.bed
mv cpg.full.bed cpg.bed

OFS — output field separator; NR — record number

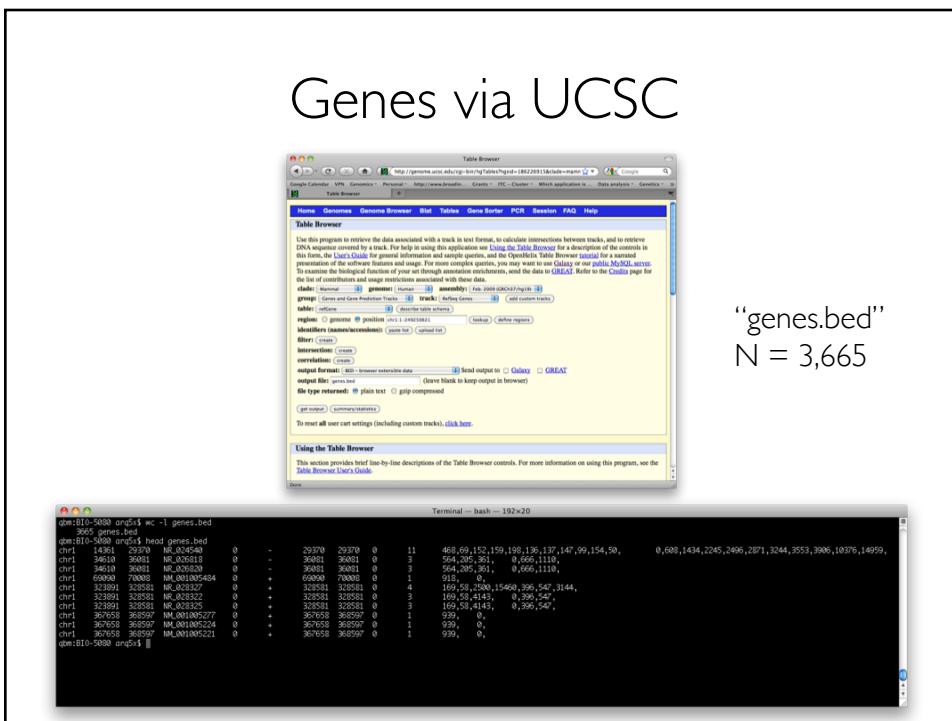


A screenshot of a terminal window titled "Terminal — bash — 105x26". The command "head cpg.full.bed" is run, showing the first few lines of the file. The file now includes a header row and uses tabs as field separators. The columns are labeled: #chrom, chromStart, chromEnd, cpg_1, length, +, and cpgNum.

```
qpm:BI0-5080 arq5x$ head cpg.full.bed
#chrom chromStart chromEnd      cpg_1  length  +      cpgNum
chr1  28735  29810  1075    +      116
chr1  135124 135563 439      +      30
chr1  327790 328229 439      +      29
chr1  437151 438164 1013    +      84
chr1  449273 450544 1271    +      99
chr1  533219 534114 895     +      94
chr1  544738 546649 1911    +      171
chr1  713984 714547 563     +      60
chr1  762416 763445 1029    +      115
qpm:BI0-5080 arq5x$
```

Genes via UCSC

“genes.bed”
N = 3,665



Find SNPs that have the potential to alter gene expression regulation by affecting methylation at CpG islands.

Let's put it all together by using BEDTools

Step 1: Find SNPs in CpG islands

```
# How many SNPs are there on chrom 1?  
wc -l snps.bed  
2064872  
# Find SNPs that overlap CpG islands using intersectBed  
bedtools intersect -a snps.bed -b cpg.bed -wa >  
snps.cpg.bed  
# How many SNPs overlapped a CpG island?  
wc -l snps.cpg.bed  
14974
```

BEDTools typically puts the second file in memory, and reads the first file line-by-line, so put the smaller file second.

-wa report the intersecting –a region,
normally, only the intersection is reported

So, 0.7% of the SNPs were in CpG islands. Is this sensible?

Step 1: Find SNPs in CpG islands

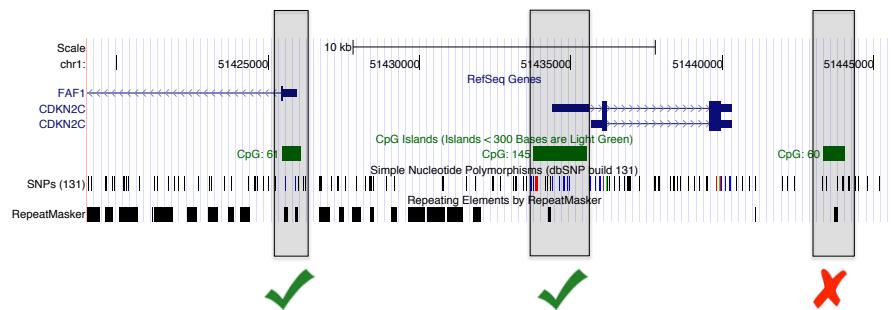
```
# How many SNPs are there on chrom 1?  
wc -l snps.bed  
2064872  
# Find SNPs that overlap CpG islands using intersectBed  
bedtools intersect -a snps.bed -b cpg.bed -wa > snps.cpg.bed  
# How many SNPs overlapped a CpG island?  
wc -l snps.cpg.bed  
14974
```

So, 0.7% of the SNPs were in CpG islands. Is this sensible?

```
# How much of chrom1 do the CpG islands occupy?  
awk '{sum+=$7} END{print sum}' cpg.bed  
1881629
```

$$1,881,629 / 249,250,621 = 0.75\%$$

Step 2: Find TSS/promoter CpG islands



We want to make sure the CpG is near the start of a gene

Step 2: Find TSS/promoter CpG islands

use "window" from BEDTools

By default, bedtools window adds 1000 bp upstream and downstream of each A feature and searches for features in B that overlap this "window". If an overlap is found in B, both the *original* A feature and the *original* B feature are reported (expands window for intersection, produces more columns):

```
$ cat A.bed
chr1 100 200
$ cat B.bed
chr1 500 1000
chr1 1300 2000
$ bedtools window -a A.bed -b B.bed
chr1 100 200 chr1 500 1000
```

```
bedtools window -l 10000 -r 0 -sw -a genes.bed -b cpg.bed | \
cut -f 13-19 | \
sort | \
uniq > cpg.gene_impact.bed
```

-l (add to left coordinate)
-r (add to right coordinate)
-sw (add/subtract based on strand)

Step 3: Find SNPs in TSS/promoter CpG islands
use intersectBed from BEDTools

```
bedtools intersect -a snps.cpg.bed -b  
cpg.gene_impact.bed > snps.cpg.gene_impact.bed
```

Step 3: Find SNPs in TSS/promoter CpG islands
use intersectBed from BEDTools

```
bedtools intersect -a snps.cpg.bed -b  
cpg.gene_impact.bed > snps.cpg.gene_impact.bed
```

What types of SNPs are they? Eg, A→G, G→T

```
grep -v "\-" snps.cpg.gene_impact.bed | \  
cut -f 7 | \  
sort | \  
uniq -c
```

BEDTools

- Genome file types/formats
 - format types
 - BED files
- Overview of genome features
- Genome arithmetic
 - approaches
 - UCSC "binning" algorithm
- Introduction to BEDtools
 - intersection / window / many more

Analysis of transcription regulation

1. Affy-chip/RNA-seq for RNA expression levels
2. Identify differentially expressed genes (edgeR, DESeq2)
 - (possibly) identify subsets of genes associated with different pathways
3. Isolate regions of DNA around (pathway-specific) coordinately expressed (induced) genes
 - BEDtools
4. Take collections of candidate regulatory regions and look for common DNA motif
 - meme, HOMER