BEDTools– Genome Arithmetic

Biol4230 Thurs, April 13, 2017
Bill Pearson wrp@virginia.edu 4-2818 Jordan 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

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

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

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

Genome data file formats:

- FASTA:

```
>sp|P09488|GSTM1_HUMAN
MPMILGYWDIRGLAHAIRLLLEYTDSSYEEKKYTMGDAPDYDRSQWLNEKFKLGLDFPNLPYLI
DGAHKITQSNAILCYIARKHNLCGETEEEKIRVDILENQTMDNHMQLGMICYNPEF
```

- FASTQ (FASTA with quality scores)

```
@UNC10-SN254_238:4:1101:1351:51174/1
CTTCGCGGTAGCTGGGACCGCCGTTCAGTCGCNAATANGCNGCTCTNTGT
+
B@CFFFFFDFHHHJJJJJJIIJJ@FHHJGIJJ##################
@UNC10-SN254_238:4:1101:1351:7840/1
ATTTTTTCACATGGAGCAGGAACTGGAGTAAATGCAANACNGTGTTNTAA
+
CCCFFFFFHHHHHJIIIJIJIIGIIJHIGGHIIIHIG#############
```

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

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 "Glutationne 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

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    +
```

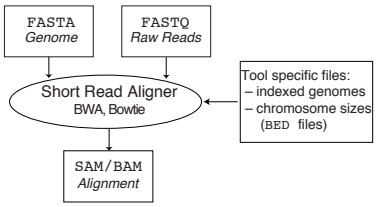
Mills (2014) Current Prot. in Bioinfo. Appendix A.1B.1-A.1B.18

BED (Browser Extensible Data) format:

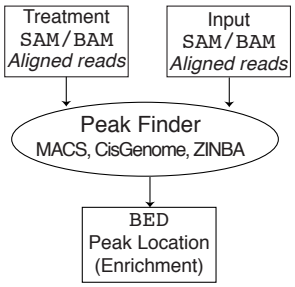
Column	Field	Description
1	Chromosome	Chr1, chr2… chrX or scaffold
2	Start	Zero based start of feature
3	Stop	One based end of a feature, not inclusive
4	Name	Name displayed in the genome browser
5	Score	0-100
6	Strand	“+” or “-“
7	thickStrand	Beginning of thick band in browser
8	thickEnd	End of thick 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

Genome analysis pipelines



Initial read mapping



Feature (peak) finding

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

Genome data (UCSC table browser)

Genomes | Genome Browser | Tools | Mirrors | Downloads | My Data | Help | About Us

Table Browser

Use this program to retrieve the data associated with a track in text format, to calculate intersections between tracks, and to retrieve DNA sequence covered by a track. For help in using this application see Using the Table Browser for a description of the controls in this form, the User's Guide for general information and sample queries, and the OpenHelix Table Browser tutorial for a narrated presentation of the software features and usage. For more complex queries, you may want to use Galaxy or our public MySQL server. To examine the biological function of your set through annotation enrichments, send the data to GREAT. Send data to GenomeSpace for use with diverse computational tools. Refer to the Credits page for the list of contributors and usage restrictions associated with these data. All tables can be downloaded in their entirety from the Sequence and Annotation Downloads page.

clade: Mammal **genome:** Human **assembly:** Dec. 2013 (GRCh38/hg38)
group: Genes and Gene Predictions **track:** UCSC Genes add custom tracks track hubs
table: knownGene describe table schema
region: genome position chr1:109631271-109750270 lookup define regions
identifiers (names/accessions): 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

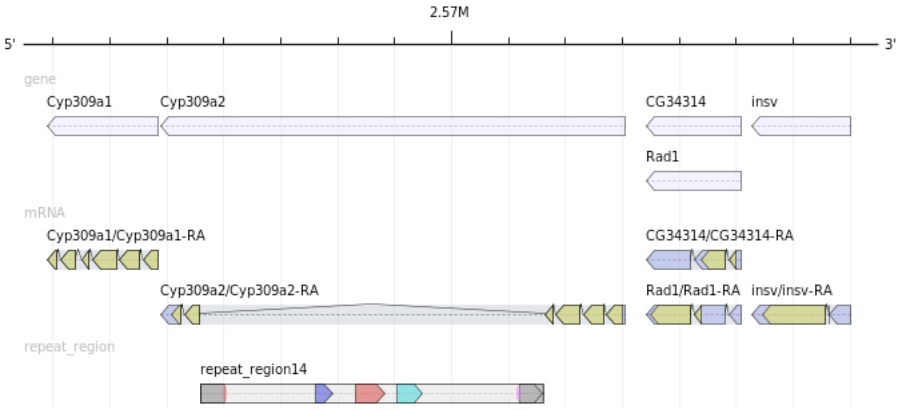
get output summary/statistics

To reset **all** user cart settings (including custom tracks), click here.

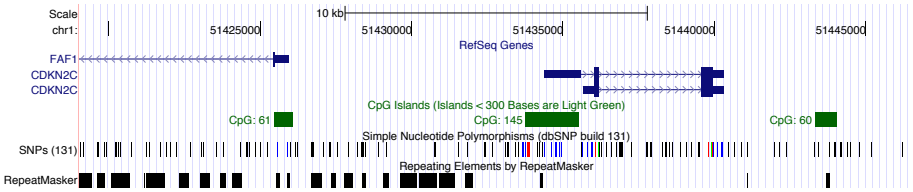
What is a genome "feature"?

- 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"

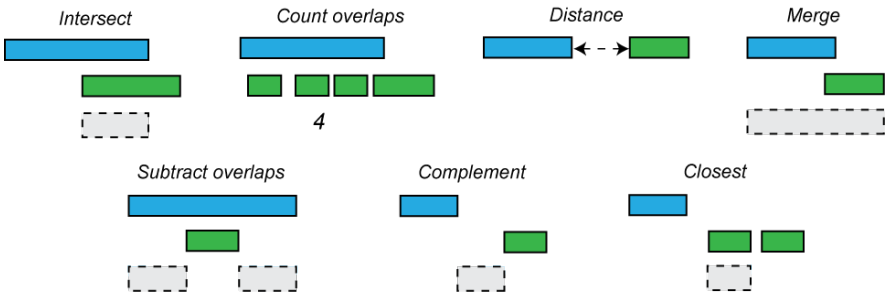


Overlaps, closest, merge…



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?

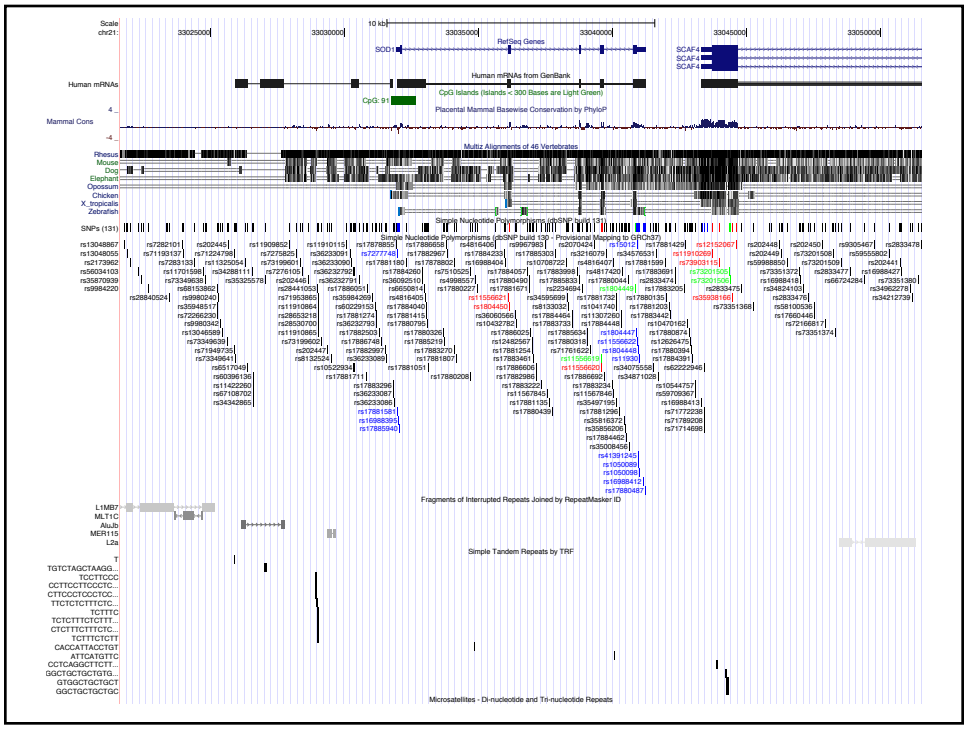
An annoying wealth of formats

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

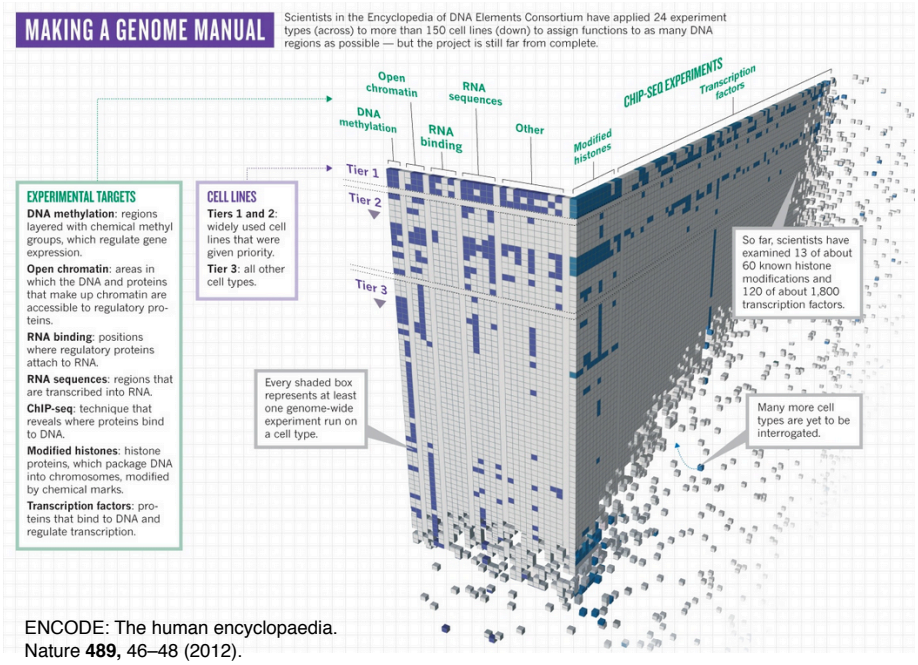
Standard attributes

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

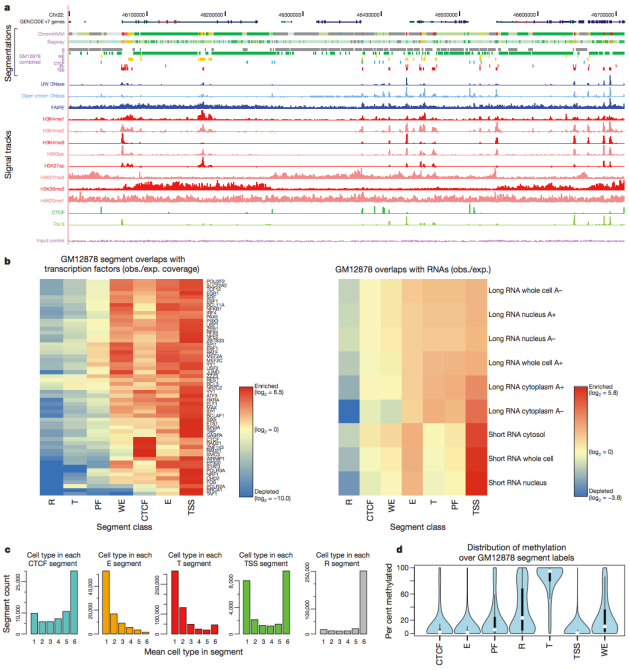
No standards: Some formats use 1-based coordinates, while others use 0-based


Scale
chr21:
10 kb
33025000
33030000
33035000
33040000
33045000
33050000
RefSeq Genes
SOD1
SCAF4
SCAF4
SCAF4
Human mRNAs
Human mRNAs from GenBank
CpG Islands (Islands < 300 Bases are Light Green)
CpG: 91
Placental Mammal Basewise Conservation by PhyloP
Mammal Cons
4
-4
Multiz Alignments of 46 Vertebrates
Rhesus
Mouse
Dog
Elephant
Opossum
Chicken
X_tropicalis
Zebrafish
SNPs (131)
Simple Nucleotide Polymorphisms (dbSNP build 130 - Provisional Mapping to GRCh37)
Fragments of Interrupted Repeats Joined by RepeatMasker ID
L1MB7
MLT1C
AluJb
MER11B
L2a
Simple Tandem Repeats by TRF
T
TGTCTAGCTAAGG...
TCCTTCCC
CCTTCCTTCCCTC...
CTTCCCTCCCTCC...
TTCTCTCTTTCTC...
TCTTTC
TCTCTTTCTCTTT...
CTCTTTCTTTCTC...
TCTTTCTCTT
CACCATTACCTGT
ATTCATGTTC
CCTCAGGCTTCTT...
GGCTGCTGCTGTG...
GTGGCTGCTGCT
GGCTGCTGCTGC
Microsatellites - Di-nucleotide and Tri-nucleotide Repeats

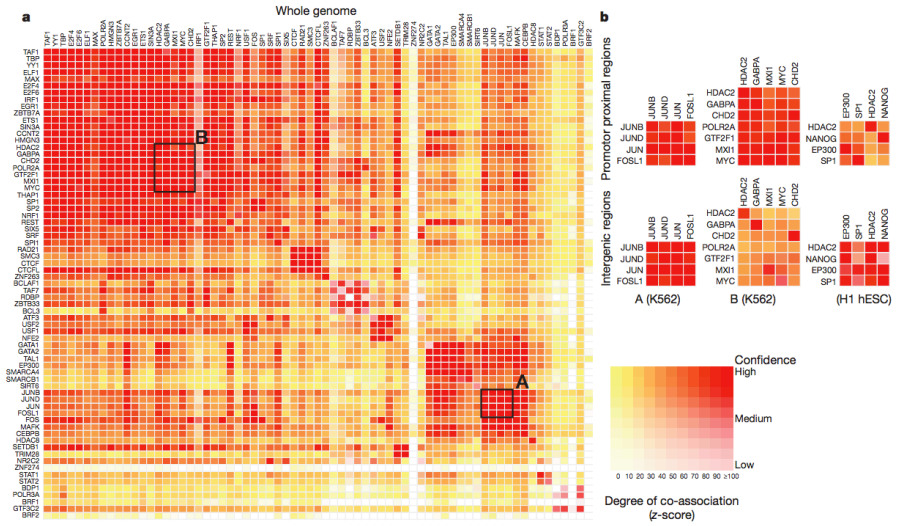
Why?



ENCODE: The human encyclopaedia. Nature **489,** 46–48 (2012).



http://www.nature.com/nature/journal/v489/n7414/pdf/nature11247.pdf



http://www.nature.com/nature/journal/v489/n7414/pdf/nature11247.pdf

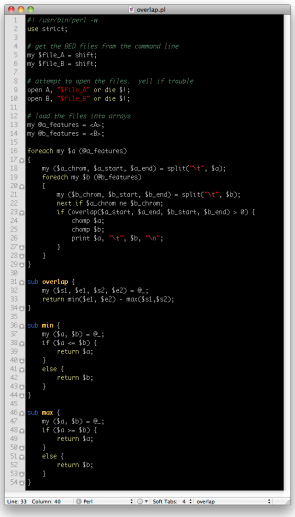
How?

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: slooooow

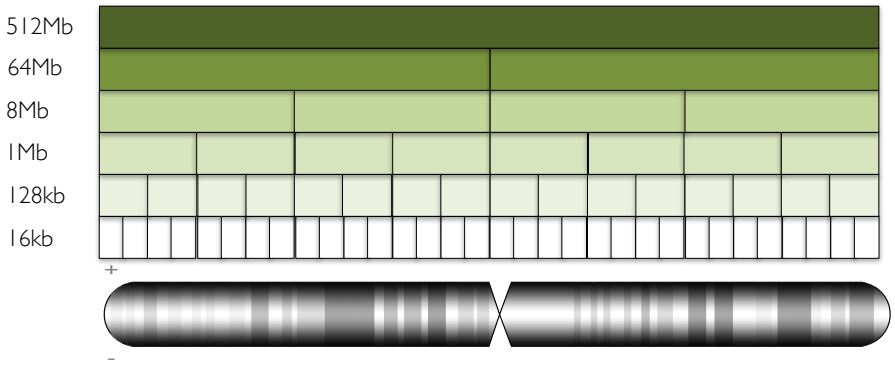
```
foreach fA in file A
{
   foreach fB in file B
   {
      does fA overlap fB?*
   }
}
```



* 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"

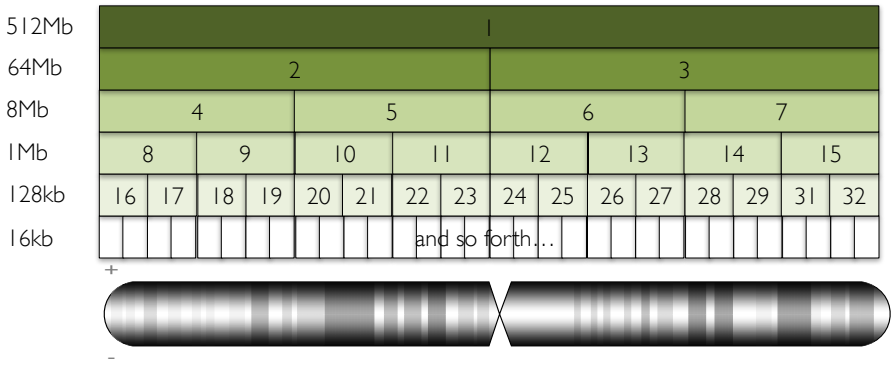
the UCSC genome browser approach



Human chrom 1: ca. 250 Mb

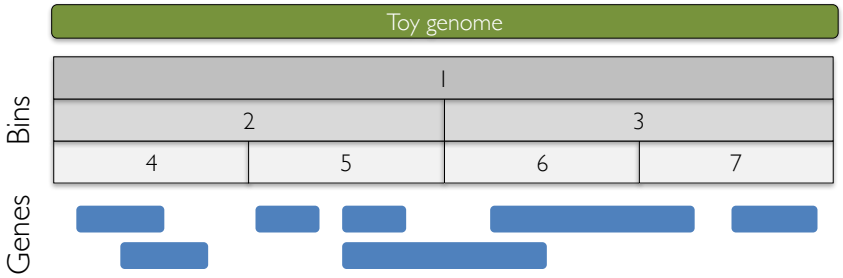
Narrow the search space w/ "binning"

the UCSC genome browser approach

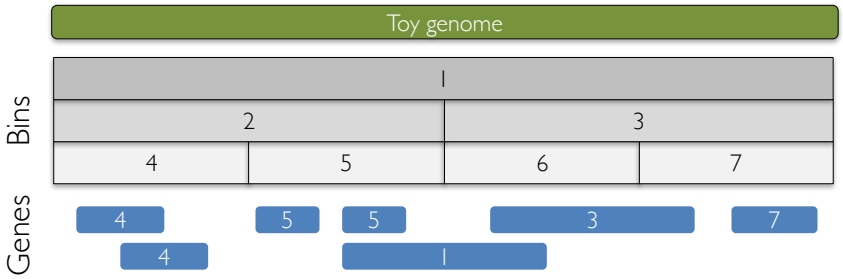


Human chrom 1: ca. 250 Mb

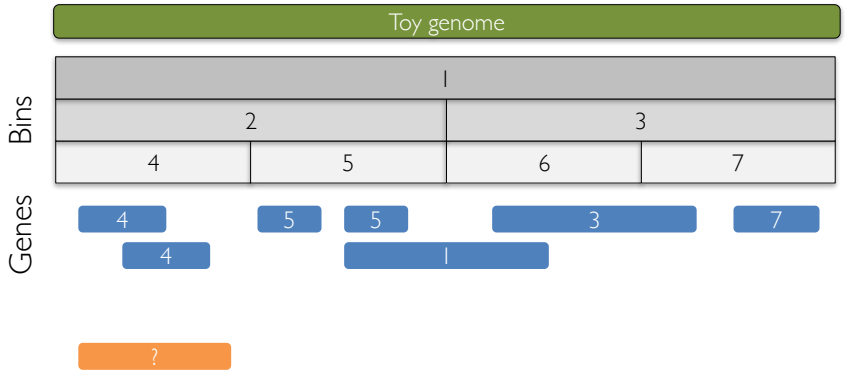
A toy example



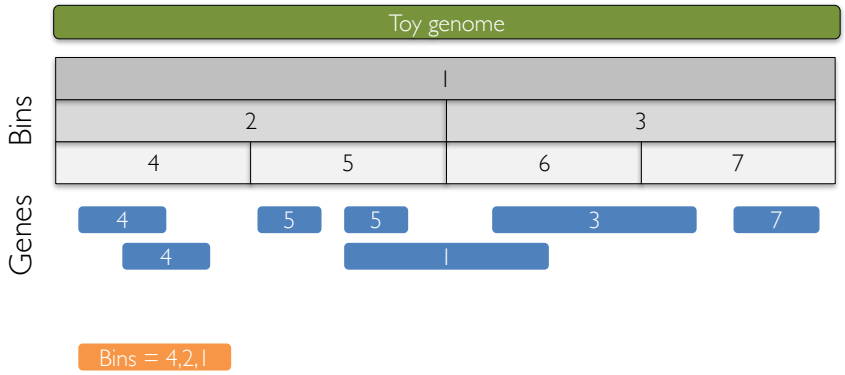
A toy example



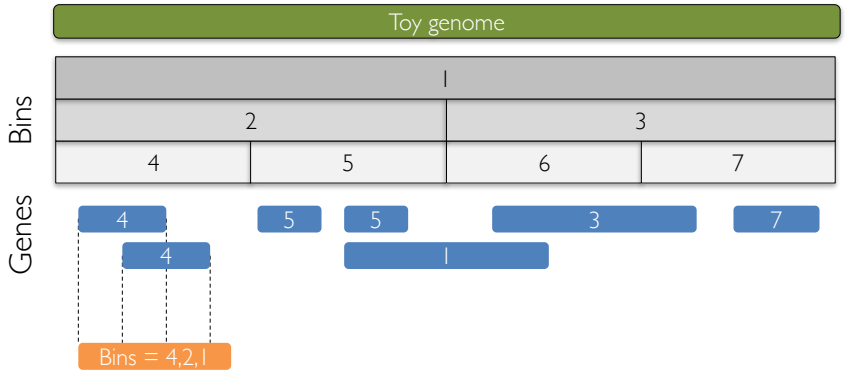
A toy example



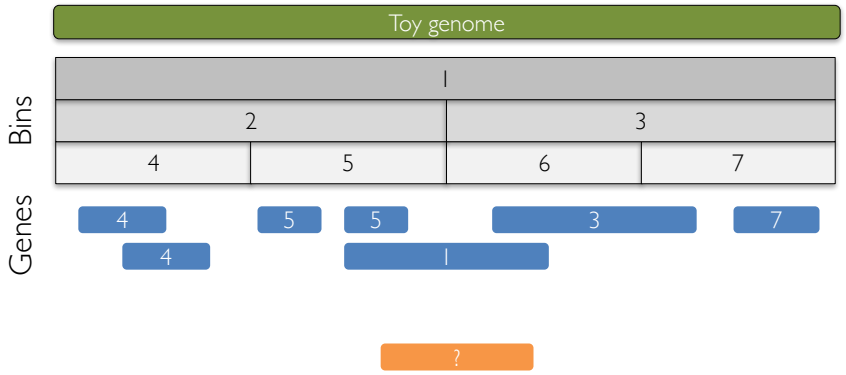
A toy example



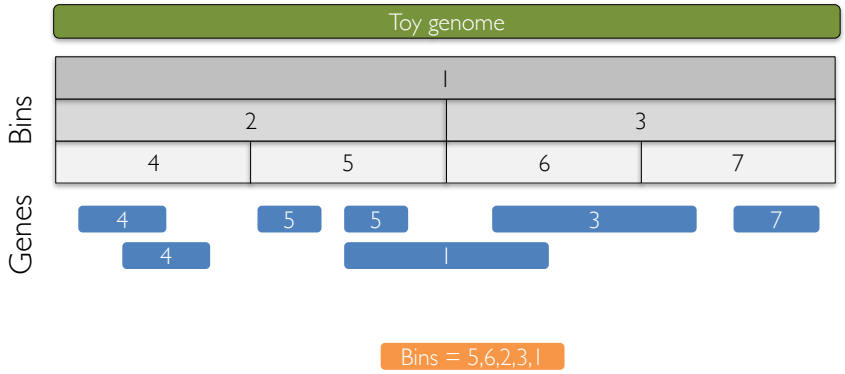
A toy example



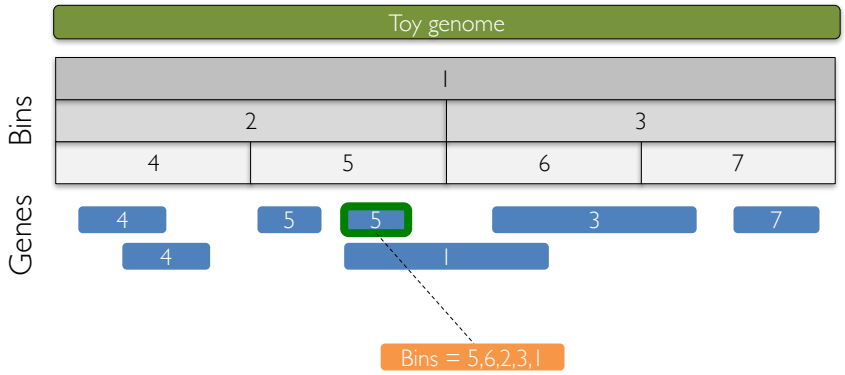
A toy example



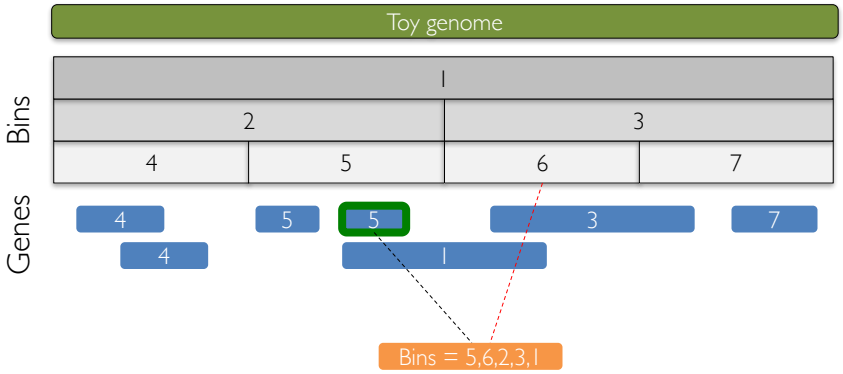
A toy example



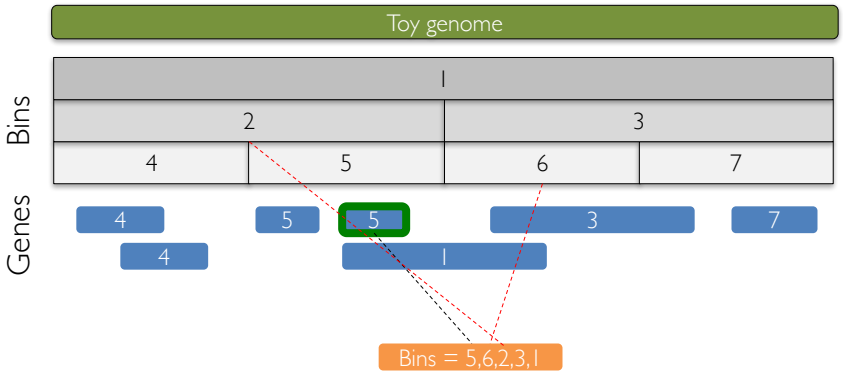
A toy example



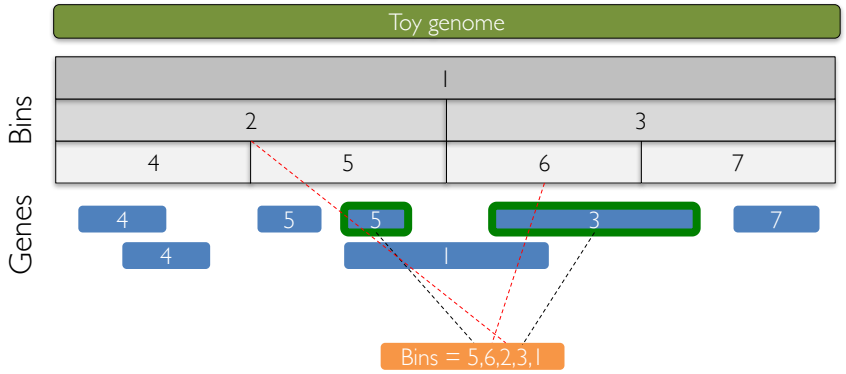
A toy example



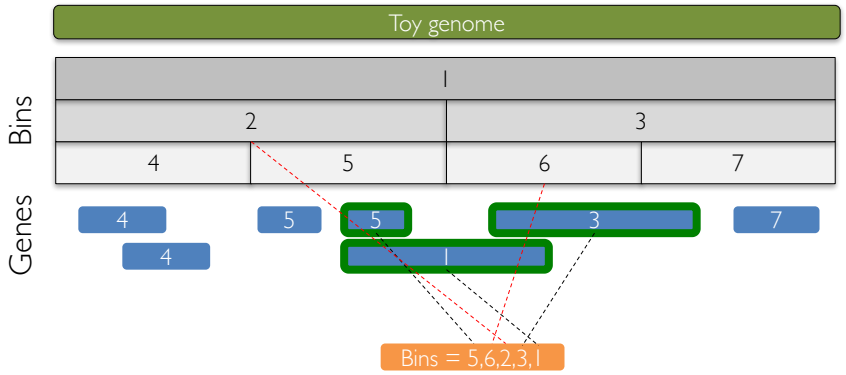
A toy example



A toy example



A toy example

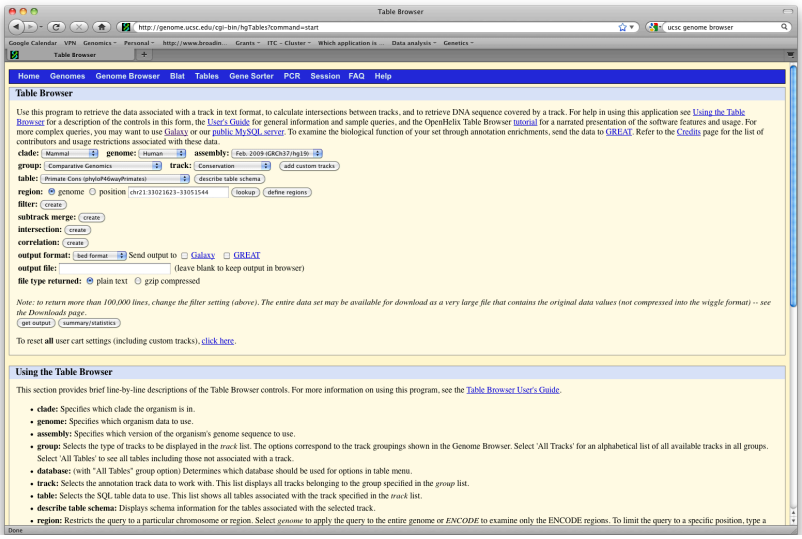


Available tools for GA

UCSC Genome Browser
Galaxy
BEDTools

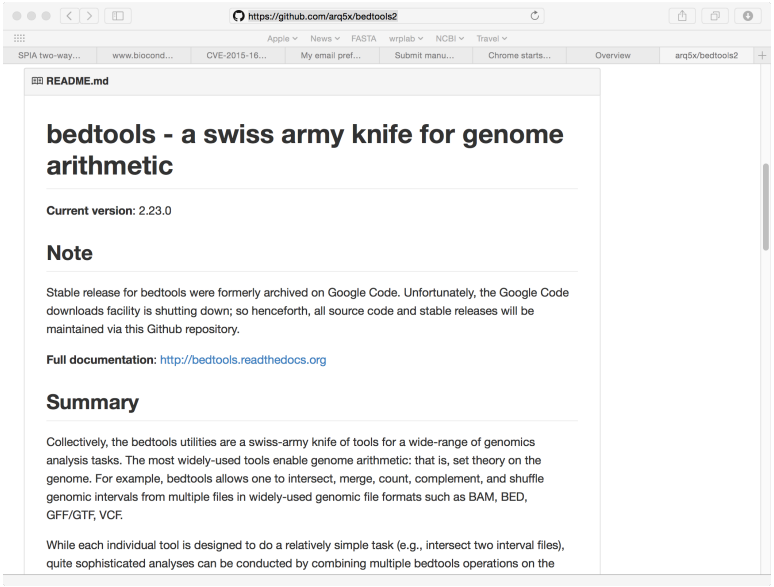
UCSC Genome Browser

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

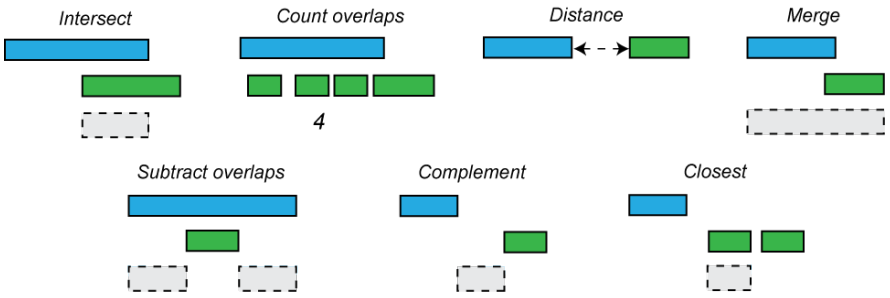


BEDTools

https://github.com/arq5x/bedtools2



BEDTools supports common GA operations and common formats



BEDTools + UNIX

Real world usage

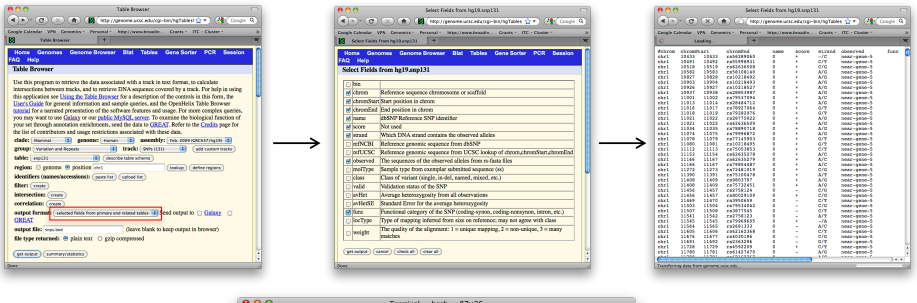
Find SNPs that have the potential to alter gene expression regulation by affecting methylation at CpG islands.

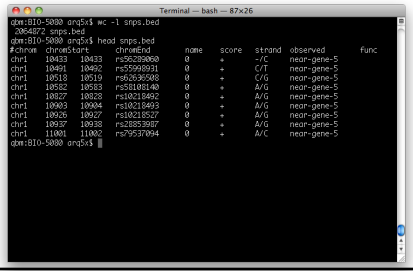
(restrict to chrom 1)

Step 1: Get annotations

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

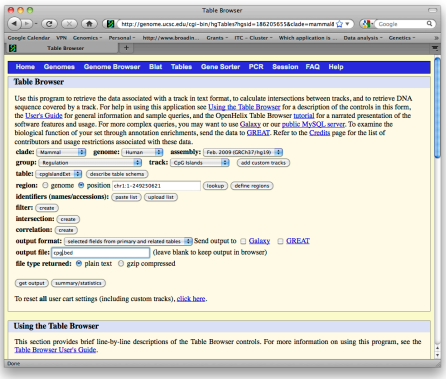
SNPs from dbSNP via UCSC

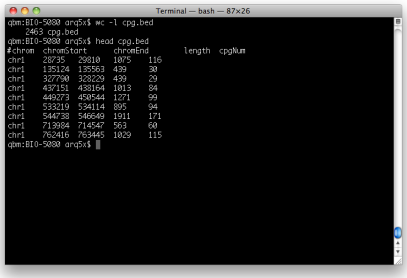




"snps.bed"
N = 2,064,872

CpG islands via UCSC





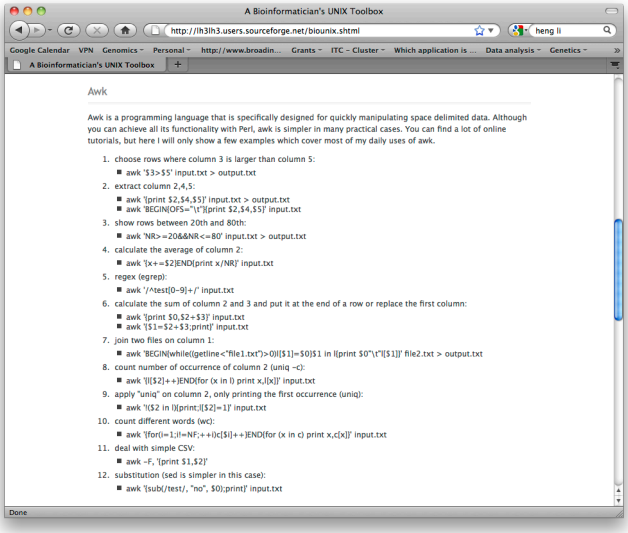
"cpg.bed"
N = 2,463

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

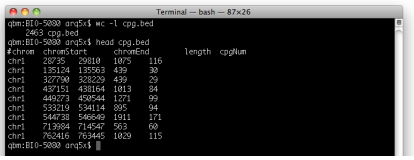
The awesome power of ***awk***

- written in 1977 by Alfred **A**ho, Peter **W**einberger, and Brian **K**ernighan
- 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 "Hello World!")
- `awk '$2 >= 30' foo.txt` (Get lines in foo where 2nd col. is g.t.e. 30)
- `awk '{print $3,$2,$7,$1}' foo.txt > foo.reordered.txt`

The awesome power of ***awk***



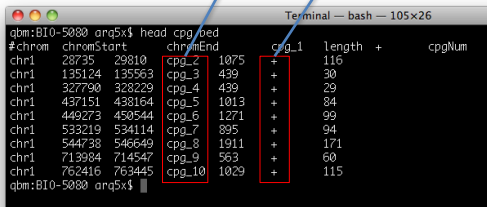
Use awk to fix cpg.bed



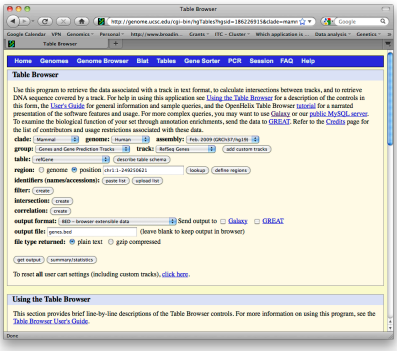
Problem: No name and no strand in the output. Let's fix it...

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

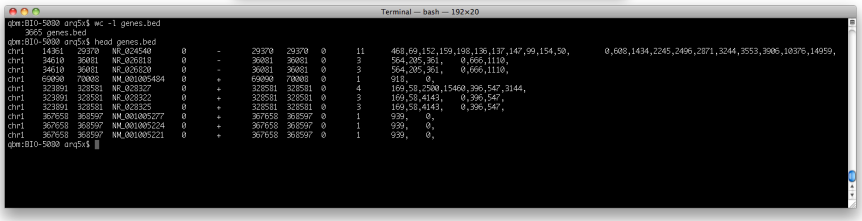
OFS – output field separator; NR – record number



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

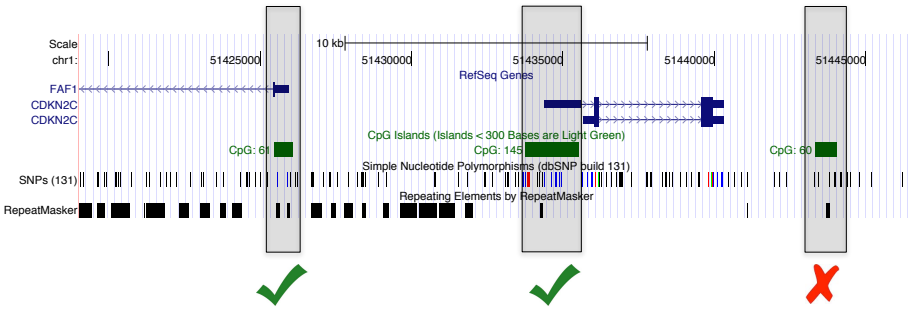
```
# How many SNPs are there on chrom 1?
wc -l snps.bed
2064872
# Find SNPs that overlap CpG islands using intersectBed
bedtools intersect -a snps.bed -b cpg.bed -wa > snps.cpg.bed
# How many SNPs overlapped a CpG island?
wc -l snps.cpg.bed
14974
```

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/substract 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? E.g., 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