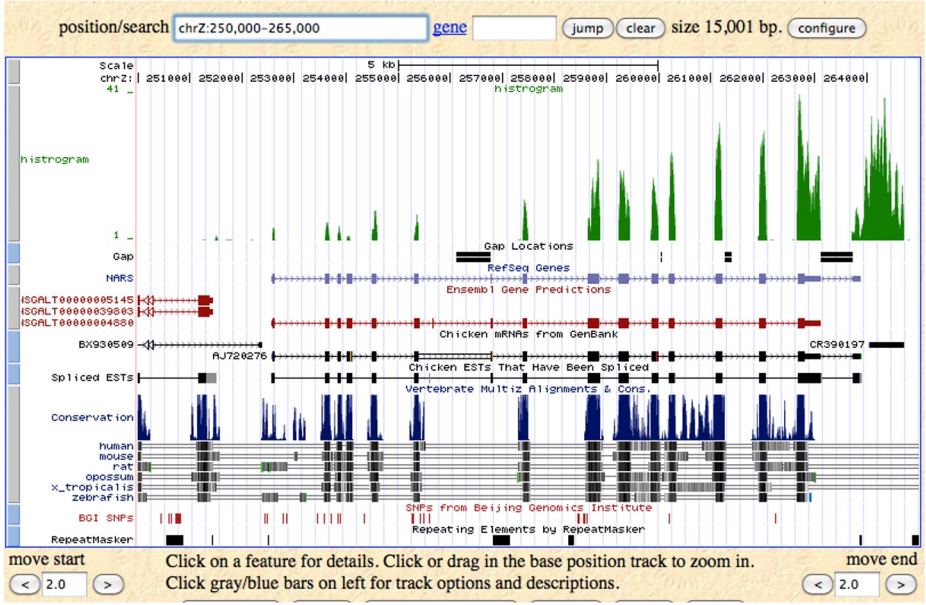
Using bedtools (and awk) to perform genome arithmetic

Adapted from Aaron Quinlan's
Applied Computational Genomics, Lecture 16 - 18

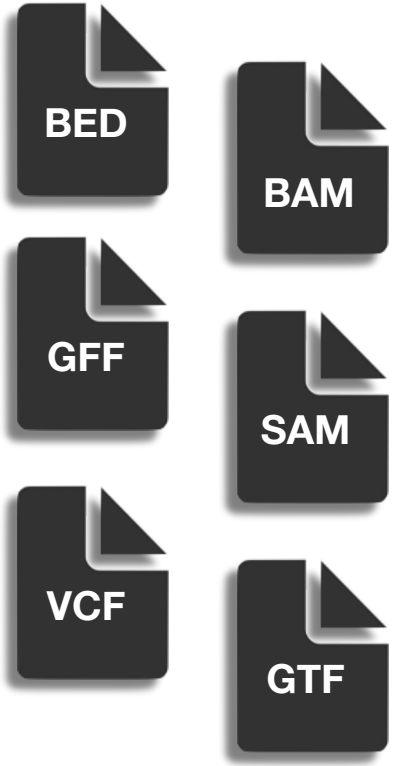
Jason Kunisaki
Quinlan Lab
University of Utah

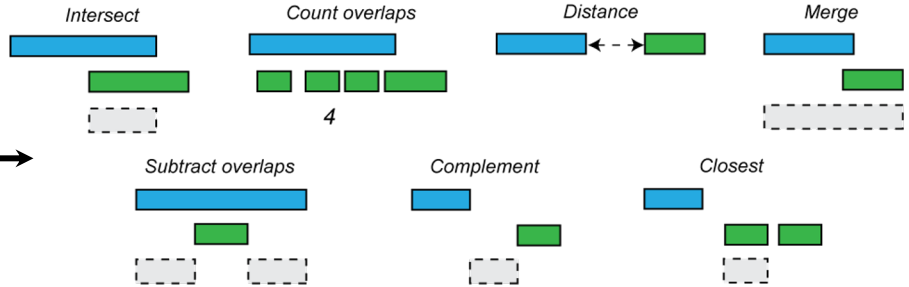
What is a genome interval?

- Genes: exons, introns, UTRs, promoters (BED, GFF, GTF
- Conservation (BEDGRAPH)
- Genetic variation (VCF)
- Sequence alignments (BAM)
- Transcription factor binding sites (BED, BEDGRAPH)
- CpG islands (BED)
- Segmental duplications (BED)
- Chromatin annotations (BED)
- Gene expression data (WIG, BIGWIG, BEDGRAPH)
- <u>Your own observations: put them in context</u>



Supports most interval formats & handles diff. coordinate systems





Sort chromosomes lexicographically.

Then sort numerically by start coordinate

```
sort -k1,1 -k2,2n interval_1.bed > interval_1.sorted.bed
```

Bedtools: example analyses

- Closest gene to a ChIP-seq peak.
- 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?
- What is the density of variants in "windows" along the genome?

http://bedtools.readthedocs.io/en/latest/

Download the data

mkdir ~/workspace/bedtools_tutorial

cd ~/workspace/bedtools_tutorial

```
curl -O https://s3.amazonaws.com/bedtools-tutorials/web/cpg.bed
curl -O https://s3.amazonaws.com/bedtools-tutorials/web/exons.bed
curl -O https://s3.amazonaws.com/bedtools-tutorials/web/gwas.bed
curl -O https://s3.amazonaws.com/bedtools-tutorials/web/genome.txt
curl -O https://s3.amazonaws.com/bedtools-tutorials/web/hesc.chromHmm.bed
```

How do we further convince ourselves we downloaded these 6 files?

These files are already sorted

Description of the files

cpg.bed --> Genome coordinates + annotations for CpG islands or genomic intervals enriched for C and G nucleotides

Description of the files

cpg.bed --> Genome coordinates + annotations for CpG islands or genomic intervals enriched for C and G nucleotides

exons.bed --> genome coordinates + transcript + strand information for exons in the human

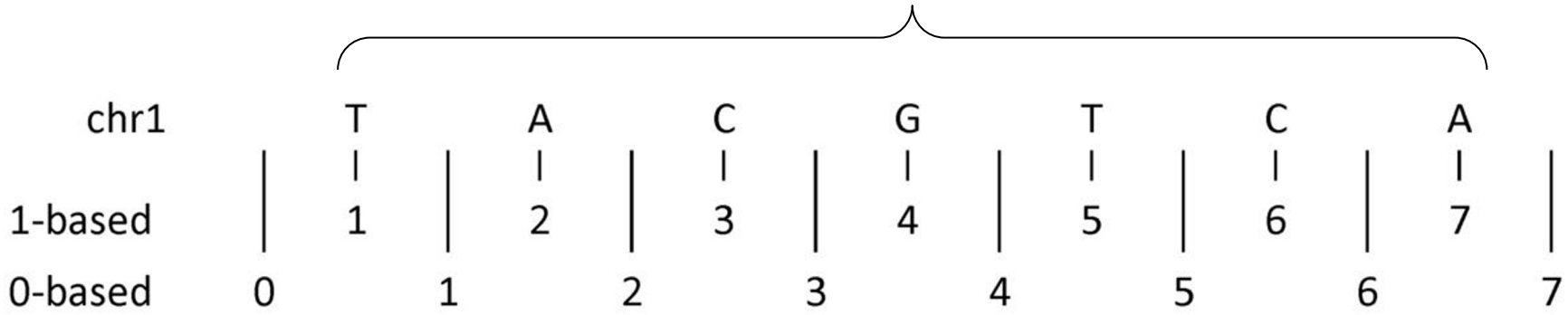
genome.txt --> lengths of all chromosomes in the human genome

gwas.bed --> genome coordinates + ID for disease-associated single nucleotide polymorphisms

hesc.chromHmm.bed --> genome coordinates + function in the genome (e.g., promoter, enhancer, etc.)

Using “basic” **awk** to process intervals from the command line

Questions we can ask with awk



Questions we can ask:

- How many nucleotides are in this interval?
- What is the sum of all intervals on chromosome 1?

https://www.biostars.org/p/84686/

Now to learn some more UNIX. **awk** is your dear friend.

AWK

A W K

Alfred Aho

Peter Weinberger

Brian Kernighan

awk dates from, I think, 1977. It's by far the biggest software project that I have been involved with. There were three of us in that, and that's completely unworkable. Somehow, it's much easier working with two rather than three. It's harder to split things. There's more divergence of opinion, sometimes that's good because it means that the more things there but sometimes it means that it's not as cohesive as it might be. On the other hand it was very very nice to work with Al Aho and Peter Weinberger so I had no problem with that.

NLLGG

Now to learn some more UNIX. **awk** is your dear friend.

Awk is a programming language that is specifically designed for quickly manipulating space delimited data.
-Heng Li (http://lh3lh3.users.sourceforge.net/biounix.shtml)

Each column is referred to by number.

Only report annotations in cpg.bed that are for chr1

```
awk '$1 == "chr1"' cpg.bed
```

Every awk program begins and ends with single quotes.

Now to learn some more UNIX. **awk** is your dear friend.

```
awk '$1 == "chr1"' cpg.bed
```

$0 refers to the entire input line

```
awk '{ if ($1 == "chr1") print $0 }' cpg.bed
```

If using a print statement, you must add curly brackets between the single quotes describing the program.

Now to learn some more UNIX. **awk** is your dear friend.

```
awk '$1 == "chr1"' cpg.bed
```

```
awk '{ if ($1 == "chr1") print $0 }' cpg.bed
```

```
awk '{ if ($1 == "chr1" || $1 == “chr22”) print $0 }' cpg.bed
```

What happens if you replace || with &&?

We can also compare values across columns using some critera

Sanity check your bed file.

What's another way to do this?

```
awk '$3 <= $2' cpg.bed
```

Only report annotations in cpg.bed where the end coordinate is less than or equal to the start coordinate. How many such records do we expect?

chr1		T		A		C		G		T		C		A	
1-based		1		2		3		4		5		6		7	
0-based	0		1		2		3		4		5		6		7

0-based BED file

chr1	2	3
chr1	4	7

What would these coordinates indicate in 1-base?

chr1	**C**	2	3
chr1	**TCA**	4	7

Do some computation and report the results

```
chr1            2
        3       C
chr1            4
        7       TCA
```

$0 refers to the entire input line

Print the BED record followed by the length (end - start) of the record

```
awk '{print $0, $3-$2}' cpg.bed
```

If using a print statement, you must add curly brackets between the single quotes describing the program.

However, there is something wrong with this file...

```
## Store stdout in a file
awk '{print $0, $3-$2}' cpg.bed > cpg_length.bed

## Look at hidden characters
cat -t -e cpg_length.bed
```

What do you see?

How to resolve the lack of a tab-delimited file

There are many approaches to resolve this issue, but this is probably the most intuitive one:

```
awk -v OFS="\t" '{print $0, $3-$2}' cpg.bed
```

OFS stands for **o**utput **f**ield **s**eparator

- Default behavior is to use a space
- We can specify how our output should be separated

How to use these values to quantify the total length of all intervals

Compute the total number of base pairs represented by CpG islands

Create a variable named "sum" whose value starts at 0, but is increased by the length ($3-$2) of each CpG island.

END: after all the processing of each line in the file occurs, print the final value of sum.

```
awk '{sum += $3-$2} END{print sum}' cpg.bed
```

However, this is "lazy"

- We are not initializing the variable sum

Or more formally…

```
awk 'BEGIN {sum=0} {sum += $3-$2} END  {print sum}' cpg.bed
```

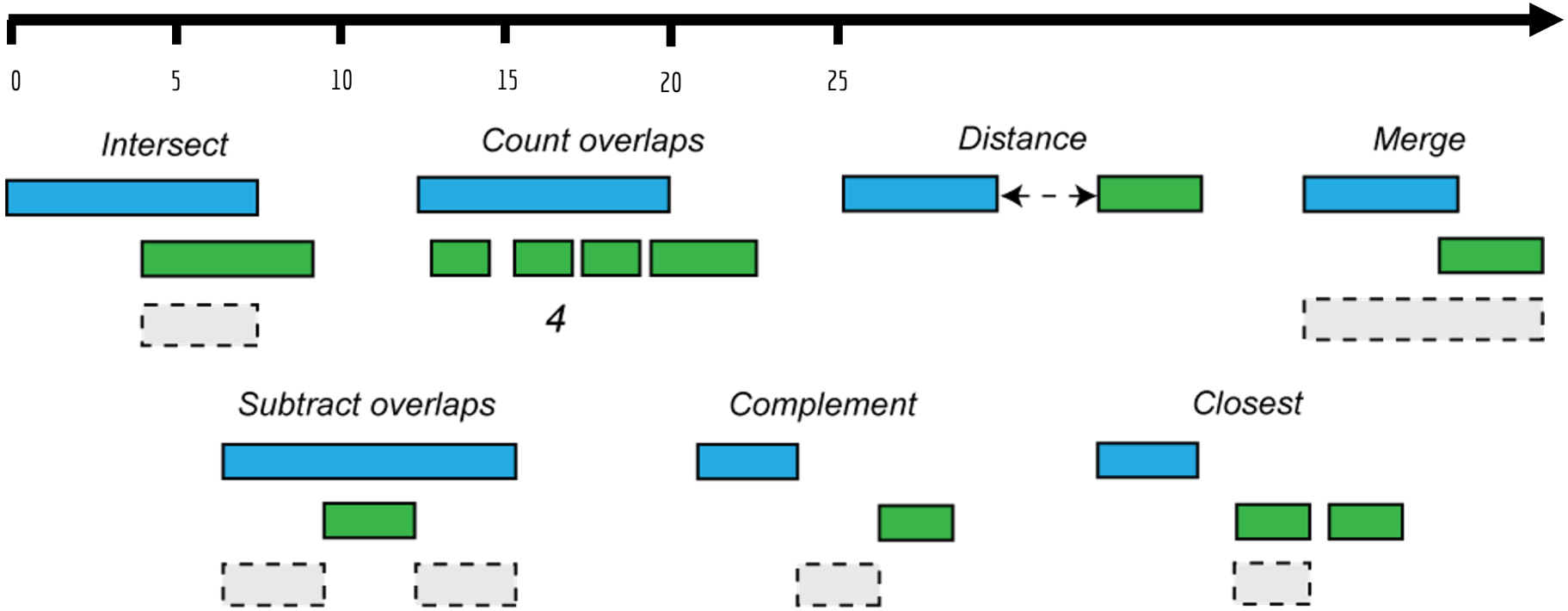
```
## Combining above with an if statement
awk 'BEGIN {sum=0} { if ($1 == "chr1") sum += $3-$2} END
{print sum}' cpg.bed
```

How many (whitespace-separated) columns are on each line?

NF: The number of "fields" (that is, the number of whitespace-separated values) detected for the line

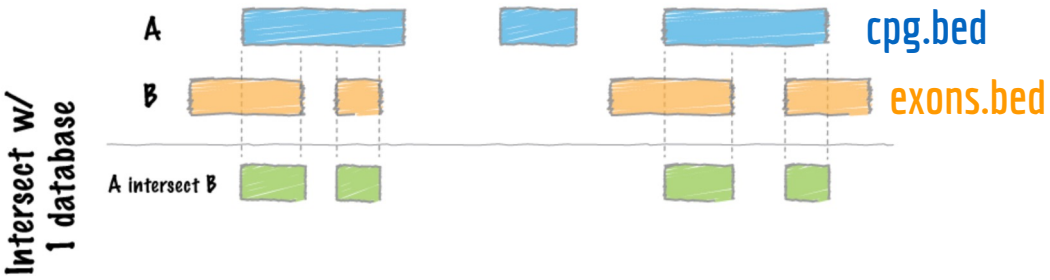
```
awk '{print NF}' cpg.bed
```

Using awk to perform genome arithmetic could get complicated



Let's work through the bedtools tutorial

bedtools intersect




```
bedtools intersect -a cpg.bed -b exons.bed | head -5
```

What does this output?
The interval corresponding to the regions of overlap between the two bed files. AKA “Where the intersections occurred”

Let’s step through that output

```
bedtools intersect -a cpg.bed -b exons.bed | head -5

chr1    29320   29370   CpG:_116
chr1    135124  135563  CpG:_30
chr1    327790  328229  CpG:_29
chr1    327790  328229  CpG:_29
chr1    327790  328229  CpG:_29
```

What’s up with this?
This entry only occurs once in the cpg.bed file…

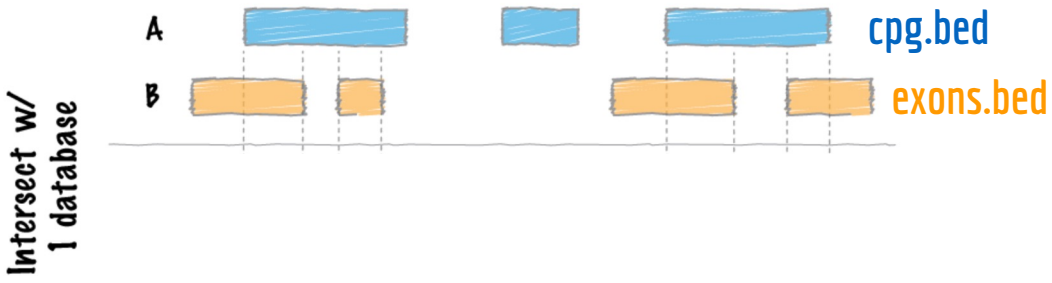
Any ideas?

bedtools intersect -wa -wb -u


```
bedtools intersect -a cpg.bed -b exons.bed -wa
bedtools intersect -a cpg.bed -b exons.bed -wb
bedtools intersect -a cpg.bed -b exons.bed -wa -wb
bedtools intersect -a cpg.bed -b exons.bed -wa -wb -u ## what?
```

What does this output? “What intersected”

bedtools intersect -wo (**w**rite **o**verlap)




```
bedtools intersect -a cpg.bed -b exons.bed -wo
bedtools intersect -a cpg.bed -b exons.bed -wo -u ## why does this fail?
```

What does this output?
"Amount of overlap between intersecting features"

Awk practice question:

Output of write overlap

chr1	28735	29810	CpG:_116	chr1	29320	29370
	NR_024540_exon_10_0_chr1_29321_r			0	-	50
chr1	135124	135563	CpG:_30	chr1	134772	139696
	NR_039983_exon_0_0_chr1_134773_r			0	-	439
chr1	327790	328229	CpG:_29	chr1	324438	328581
	NR_028322_exon_2_0_chr1_324439_f			0	+	439
chr1	327790	328229	CpG:_29	chr1	324438	328581
	NR_028325_exon_2_0_chr1_324439_f			0	+	439
chr1	327[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	328581
	NR_[illegible]					439

Interval A

Interval B

50 bp

Output of write overlap

chr1	28735	29810	CpG:_116	chr1	29320	29370
	NR_024540_exon_10_0_chr1_29321_r			0	-	50
chr1	135124	135563	CpG:_30	chr1	134772	139696
	NR_039983_exon_0_0_chr1_134773_r			0	-	439
chr1	327790	328229	CpG:_29	chr1	324438	328581
	NR_028322_exon_2_0_chr1_324439_f			0	+	439
chr1	327790	328229	CpG:_29	chr1	324438	328581
	NR_028325_exon_2_0_chr1_324439_f			0	+	439
chr1	327[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	328581
	NR_[illegible]					439

Interval A

Interval B?

Output of write overlap

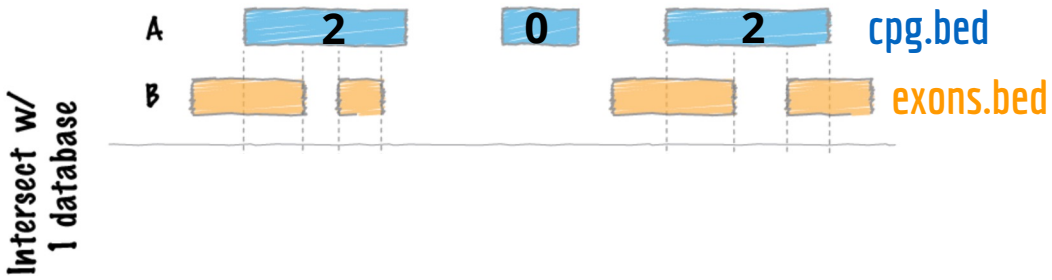
chr1	28735	29810	CpG:_116	chr1	29320	29370
	NR_024540_exon_10_0_chr1_29321_r			0	-	50
chr1	135124	135563	CpG:_30	chr1	134772	139696
	NR_039983_exon_0_0_chr1_134773_r			0	-	439
chr1	327790	328229	CpG:_29	chr1	324438	328581
	NR_028322_exon_2_0_chr1_324439_f			0	+	439
chr1	327790	328229	CpG:_29	chr1	324438	328581
	NR_028325_exon_2_0_chr1_324439_f			0	+	439
chr1	327[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	328581
	NR_[illegible]					439

Interval A

Interval B

439 bp

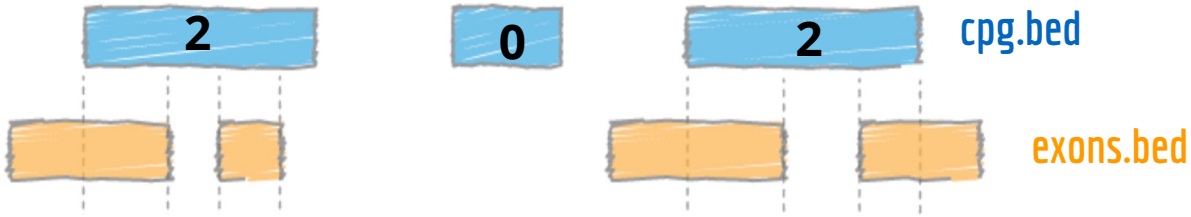
bedtools intersect -c (count)



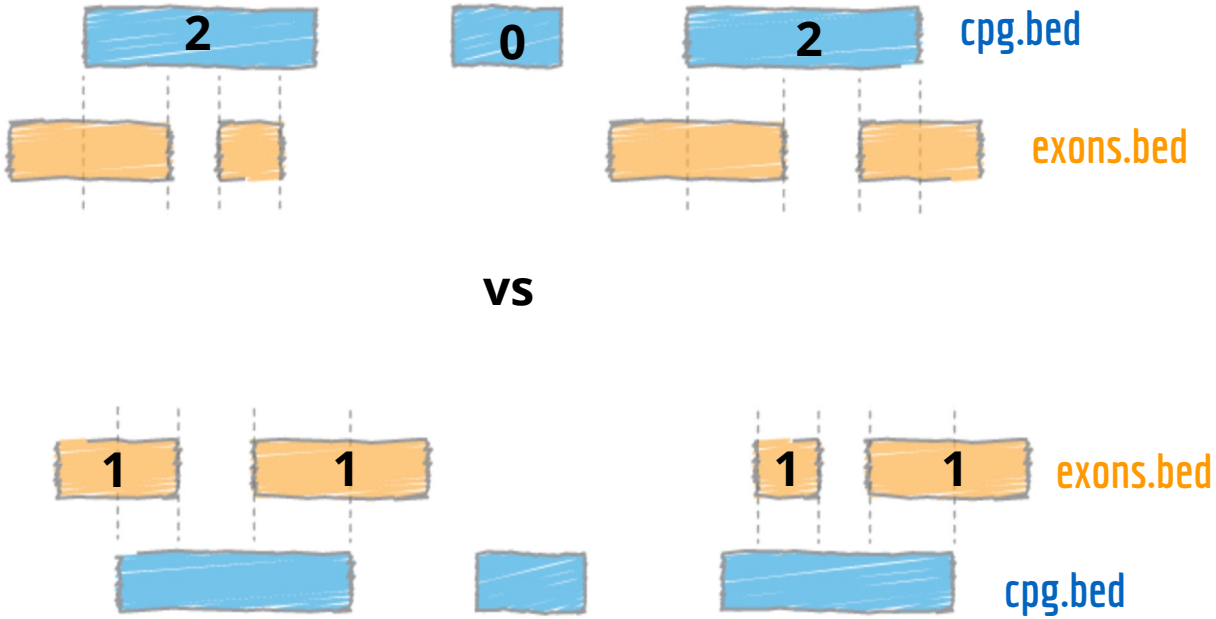

```
bedtools intersect -a cpg.bed -b exons.bed -c | head
```

What does this output?
"The number of exons (B) found in each CpG Island interval (A)"

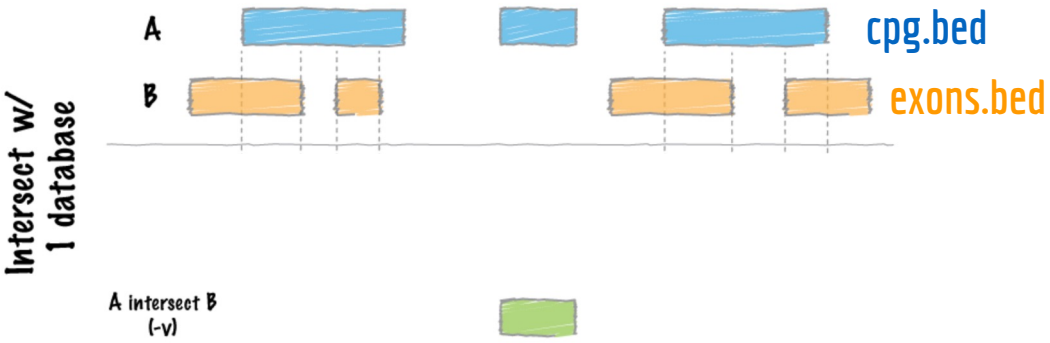
What happens if we swap the files that go into a/b?



What happens if we swap the files that go into a/b?



bedtools intersect -v




```
bedtools intersect -a cpg.bed -b exons.bed -v | head
```

What does this output?
"CpG islands (A) that do not overlap any exons (B)"

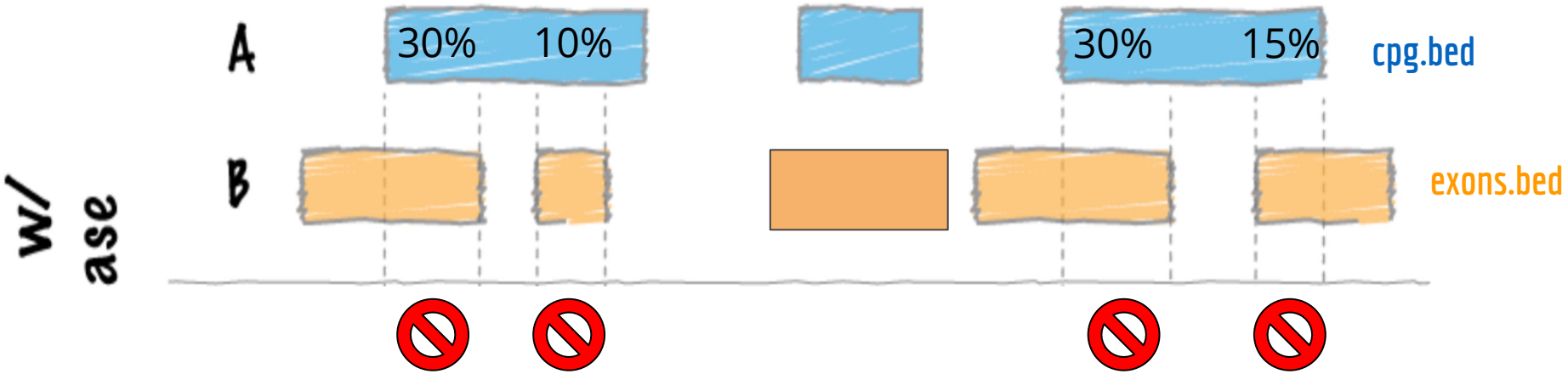
Question: what happens if you flipped it where -a exons.bed and -b cpg.bed

bedtools intersect -wo -f


```
-f requires sufficient fractional overlap between intervals.
bedtools intersect -a cpg.bed -b exons.bed -wo -f 0.50 | head
```

Requires 50% of interval A to be overlapped by interval B

bedtools intersect -wo -f

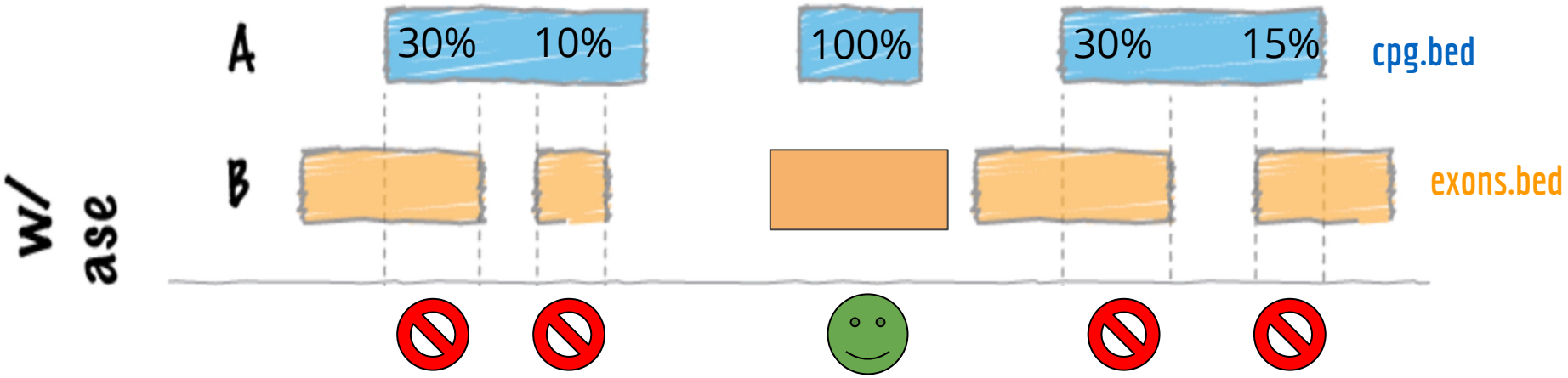


-f requires sufficient fractional overlap between intervals.

```
bedtools intersect -a cpg.bed -b exons.bed -wo -f 0.50 | head
```

Requires 50% of interval A to be overlapped by interval B

bedtools intersect -wo -f

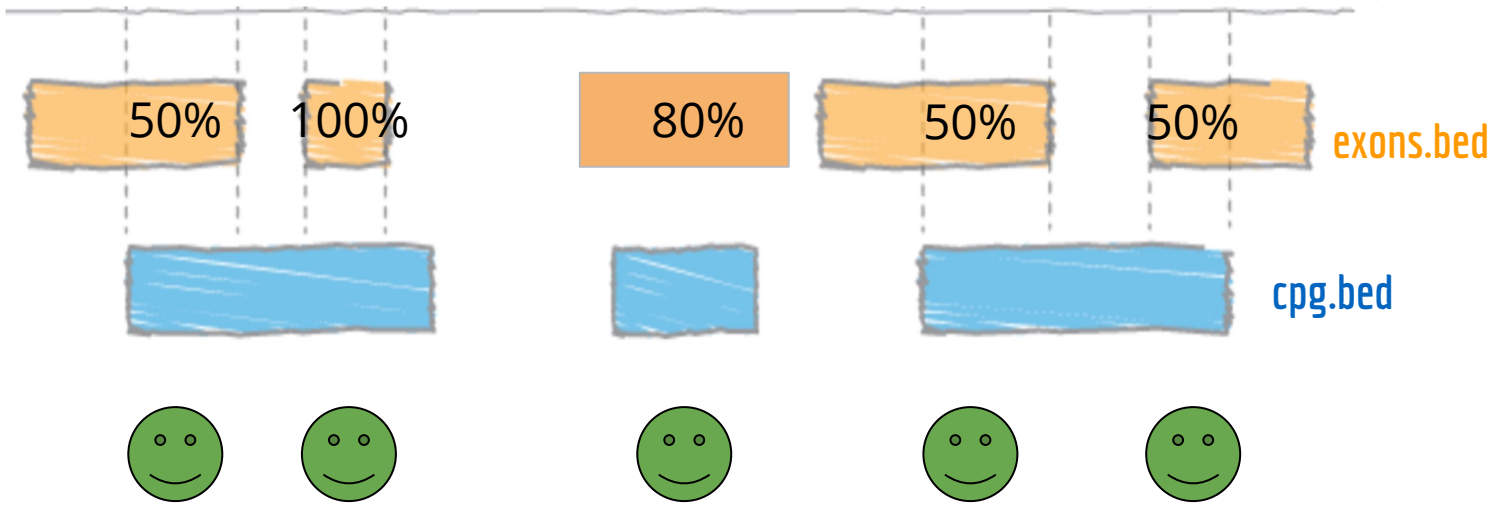


-f requires sufficient fractional overlap between intervals.

```
bedtools intersect -a cpg.bed -b exons.bed -wo -f 0.50 | head
```

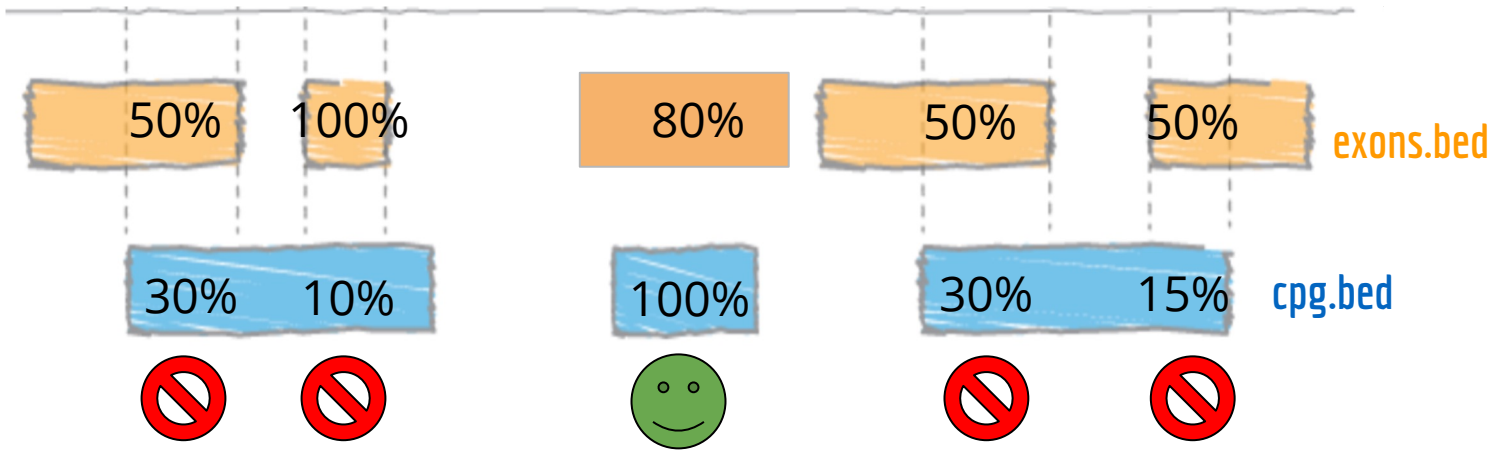
Requires 50% of interval A to be overlapped by interval B

What happens if we swap -a and -b?




```
bedtools intersect -a exons.bed -b cpg.bed -wo -f 0.5 | head -n 5
```

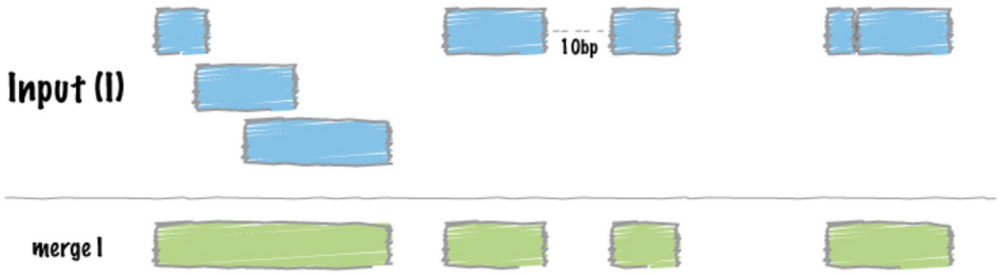
bedtools intersect -wo -f -r




```
bedtools intersect -a cpg.bed -b exons.bed -wo -f 0.5 -r | head
```

`-r` --› requires reciprocal overlap at the value defined by f

bedtools merge

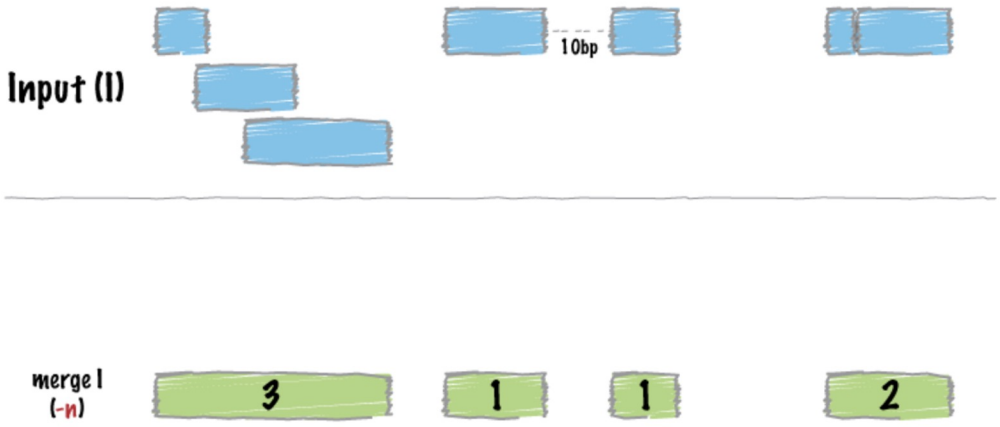


We must sort the file we want to merge (it is required)

How to sort exons.bed by chr and then start?

```
bedtools merge -i exons.bed | head -n 10
```

bedtools merge -c 1 -o count (operation)



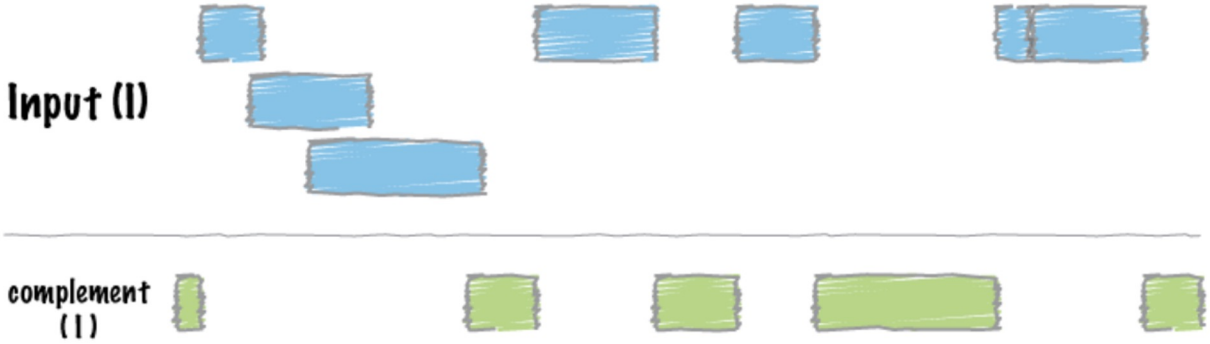
We must sort the file we want to merge (it is required)

How to sort exons.bed by chr and then start?

```
bedtools merge -i exons.bed -c 1 -o count | head -n 5
```

What does this do? Counts the number of overlapping intervals that were merged together

bedtools complement




```
bedtools complement -i exons.bed -g genome.txt > nonexonic.bed
```

What does this do? Identifies intervals of a genome that are not covered.
Why do we need the genome.txt file which contains chromosome lengths?

Bedtools exercises

1. Create a BED file representing all of the intervals in the genome that are NOT exonic and are not Promoters (based on the promoters in the hESC file).
2. What is the average distance from GWAS SNPs to the closest exon? (Hint - have a look at the closest tool.)
3. Count how many exons occur in each 500kb interval ("window") in the human genome. (Hint - have a look at the makewindows tool.)
4. Are there any exons that are completely overlapped by an enhancer? If so, how many?
5. What fraction of the GWAS SNPs are exonic? Hint: should you worry about double counting?