# Bioinformatics Analysis Tools for NGS Data

## Primer on down-stream analysis

Philipp Rescheneder, Moritz Smolka

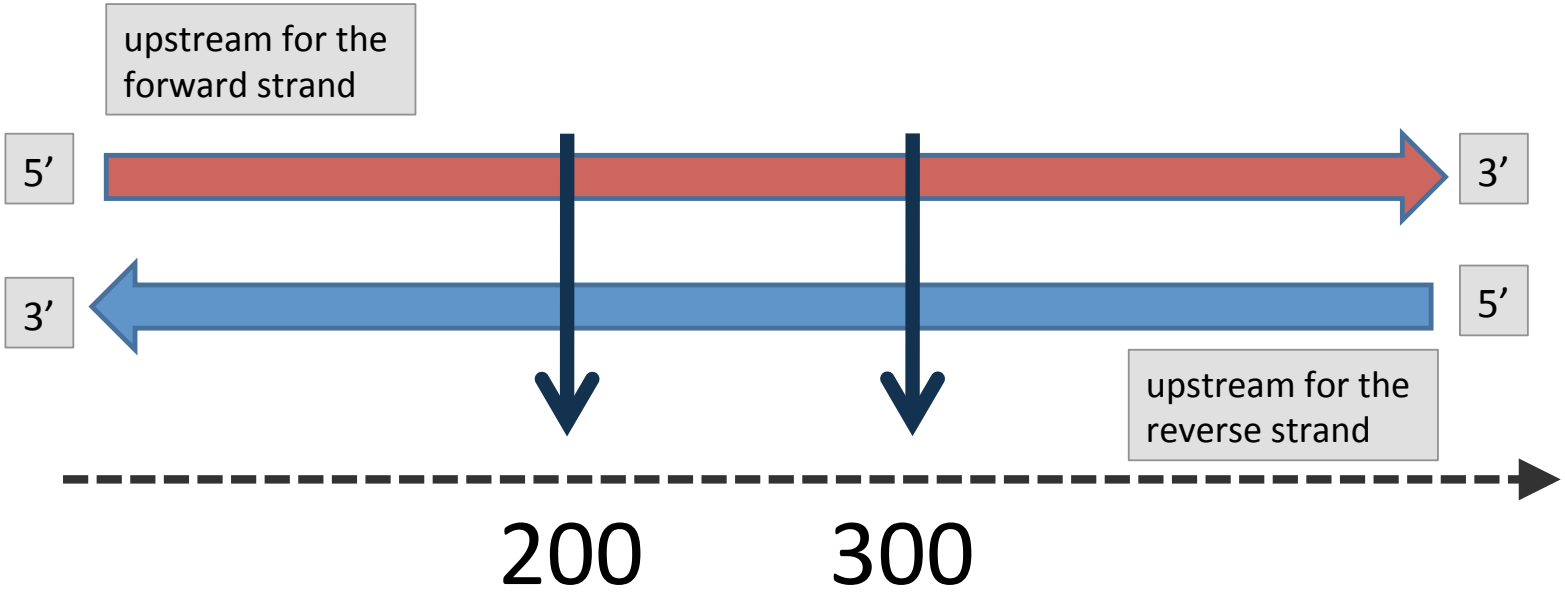
April 29, 2016

Center for Integrative Bioinformatics Vienna
Max F. Perutz Laboratories

CIBIV
Center for Integrative Bioinformatics Vienna

# Genomic coordinates

DNA two stranded and direc9onal
But there is only one coordinate system




Standard formats use **start < end** even for the reverse strand

The **upstream region** – before the 5' end rela9ve to the direc9on of transcrip9on

# Coordinate systems

- 0 based → 0, 1, 2, … 9
- 1 based → 1, 2, 3, … 10

**Typically**

- 0 based are non-inclusive **10:20 → [ 10, 20 )**
- 1 based include both ends **10:20 → [ 10, 20 ]**

# What is a genomic feature?

- Feature: a genomic region (interval) associated with a certain annotation (description).

Typical attributes to describe a feature

1. chromosome
2. start
3. end
4. strand
5. name

Why do we have so many variants? There is no good rational reason … history I guess

# Data format fruit salat

UCSC Genome Bioinformatics: FAQ - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://genome.ucsc.edu/FAQ/FAQformat.html

Gmail Google iGoogle RSeek LionDB Silva Local BCC Th Calendar Mend Biostar Unfuddle

Gmail - Pugh La... | Welcome! — Bio... | samtools.1 | Course: Practical ... | UCSC Genome ...

UCSC Genome Bioinformatics

Home - Genomes - Blat - Tables - Gene Sorter - PCR - Proteome - Help

**Frequently Asked Questions: Data File Formats**

- BED format
- bigBed format
- BED detail format
- PSL format
- GFF format
- GTF format
- MAF format
- BAM format
- WIG format
- bigWig format
- Microarray format
- Chain format
- Net format
- Axt format
- .2bit format
- .nib format
- GenePred table format
- Personal Genome SNP format

Done

# Most commonly used formats

- **BED** – UCSC genome browser → 0 based non inclusive → also used to display tracks in the genome browser (US "standard")
  (variants: **bigBed**, **bedgraph**)

- **GFF** – Sanger institute in Great Britain → 1 based inclusive indexing system ("European standard"),
  (variants: **GTF**, **GFF 2.0**)

# BED FORMAT

Search for BED format

Tab separated 3 required and 9 optional columns. Lower numbered filed must be filled.

1. **chrom** (name of the chromosome, sequence id)
2. **chromStart** (starting position on the chromosome)
3. **chromEnd** (end position of the chromosome, **note** this base is not included!)
4. **name** (feature name)
5. **score** (between 0 and 1000)
6. **strand** (+ or -)
7. **thickStart** (the starting position at which the feature is drawn thickly)
8. **thickEnd** (the ending position at which the feature is drawn thickly)
9. **itemRGB** (RGB color → 255, 0, 0 display color of the data contained)
10. **blockCount** (the number of blocks (exons) in the BED line.)
11. **blockSizes** (a comma-separated list of the block sizes)
12. **blockStarts** (a comma-separated list of the block starts)

These files may also take a track definition line that is visualization specific

# BedGraph format

Tab separated 4 required columns.

1. **chrom** (name of the chromosome, sequence id)
2. **chromStart** (starting position on the chromosome)
3. **chromEnd** (end position of the chromosome, **note** this base is not included!)
4. **dataValue** (value of the data for that region)

# GFF FORMAT

Search for GFF3 → http://www.sequenceontology.org/gff3.shtml

Tab separated with 9 columns. Missing attributes may be replaced with a dot → .

1. **Seqid** (usually chromosome)
2. **Source** (where is the data coming from)
3. **Type** (usually a term from the sequence ontology)
4. **Start** (interval start relative to the seqid)
5. **End** (interval end relative to the seqid)
6. **Score** (the score of the feature, a floating point number)
7. **Strand** (+ or –)
8. **Phase** (used to indicate reading frame for coding sequences)
9. **Attributes** (semicolon separated attributes → Name=ABC;ID=1)

people like to stuff a lot of information here

# Wiggle format

- two versions → fixed step and variable step each trying to optimize the amount of data storage

```
fixedStep chrom=chr1 start=100 step=1
10
15
11
22
… … …
```

```
variableStep chrom=chr1
100 10
101 15
102 11
103 22
variableStep chrom=chr2
2000 23
2005 40
… … …
```

Wiggle is an nasty format – it looks simpler than it is – please avoid

# Convert coordinates between formats

Being "**one off**" is one of the most common errors in bioinformatics.

Conversion from GFF to BED

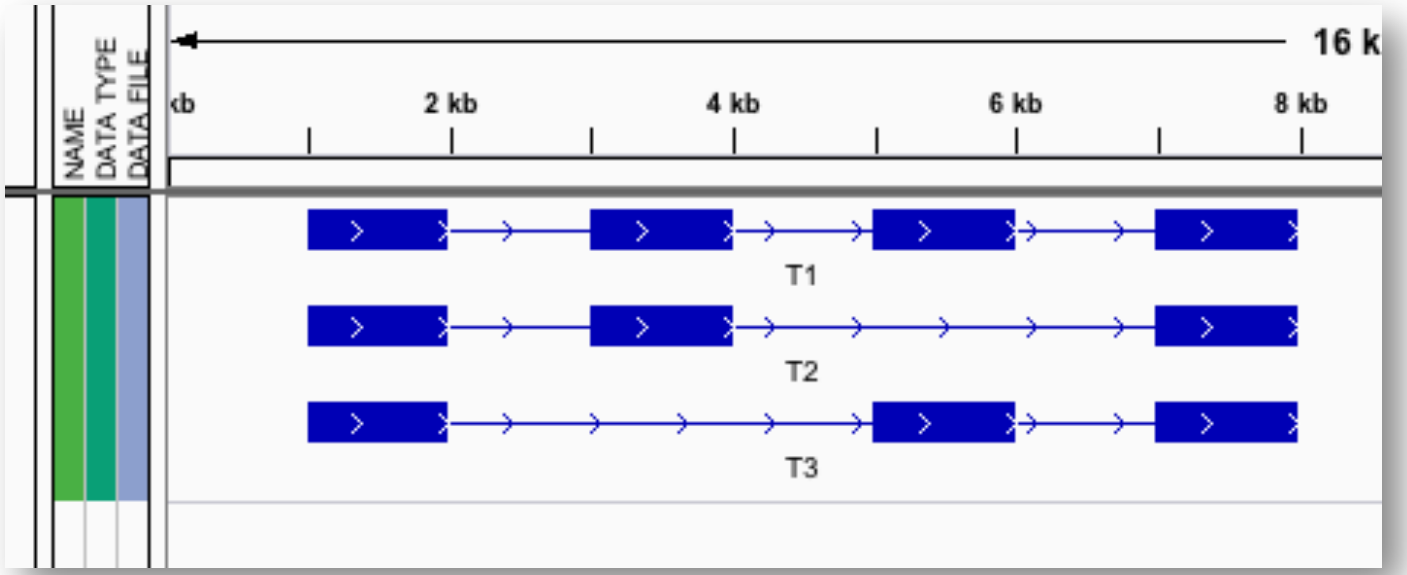
**(start, end) → (start – 1, end)**

Conversion from BED to GFF

**(start, end) → (start + 1, end)**

**Not that there will be differences when selecting positions that depend on the END coordinate!**

- We have a gene with three splicing variants




Start at 1000 ends at 8000, each exon is 1kb and is separated by 1kb

**How to represent this in a data format?**

# GFF

example.gtf (~/work/lec26)

Start Page | example.gff | example.gtf | Text-1.txt *

```
chrI	demo	exon	1000	2000	.	+	.	gene_id "G1"; transcript_id "T1";
chrI	demo	exon	3000	4000	.	+	.	gene_id "G1"; transcript_id "T1";
chrI	demo	exon	5000	6000	.	+	.	gene_id "G1"; transcript_id "T1";
chrI	demo	exon	7000	8000	.	+	.	gene_id "G1"; transcript_id "T1";
chrI	demo	exon	1000	2000	.	+	.	gene_id "G1"; transcript_id "T2";
chrI	demo	exon	3000	4000	.	+	.	gene_id "G1"; transcript_id "T2";
chrI	demo	exon	7000	8000	.	+	.	gene_id "G1"; transcript_id "T2";
chrI	demo	exon	1000	2000	.	+	.	gene_id "G1"; transcript_id "T3";
chrI	demo	exon	5000	6000	.	+	.	gene_id "G1"; transcript_id "T3";
chrI	demo	exon	7000	8000	.	+	.	gene_id "G1"; transcript_id "T3";
```

Ready | Mac-Roman | Ln: 1 Col: 81 | Text

A distinct line is entered for each exon, repeated for each transcript

# Example BED

example.bed (~/web/pyblue-sites/ialbert.web/courses/files/2013/code)

Start Page | example.bed | example.gtf

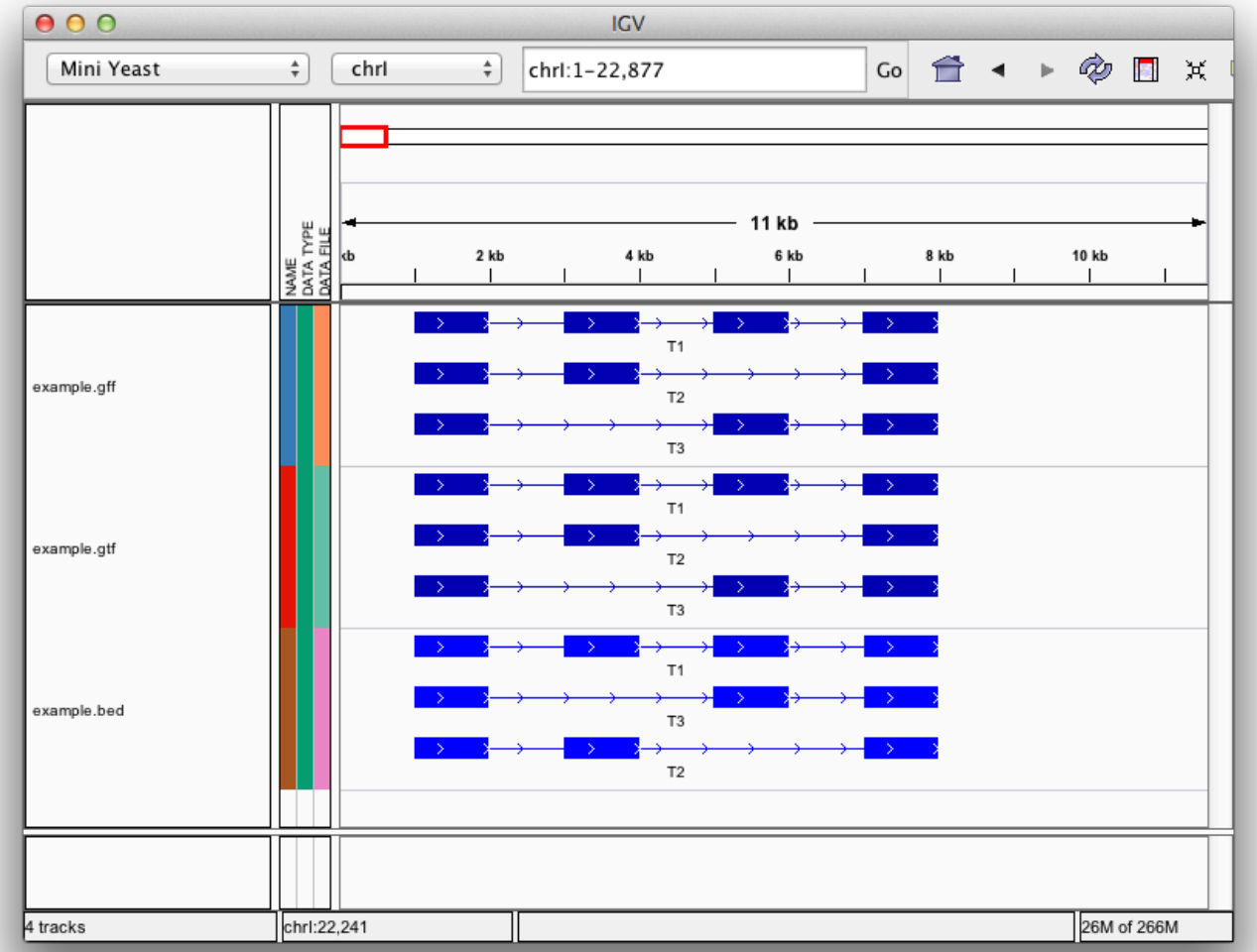
```
chrI	999	8000	T1	0	+	999	8000	0,0,255	4	1000,1000,1000,1000	0,2000,4000,6000
chrI	999	8000	T2	0	+	999	8000	0,0,255	3	1000,1000,1000	0,2000,6000
chrI	999	8000	T3	0	+	999	8000	0,0,255	3	1000,1000,1000	0,4000,6000
```

Ready | Mac-Roman | Text | Ln: 1 Col: 26

6. **strand** - Defines the strand - either '+' or '-'.
7. **thickStart** - The starting position at which the feature is drawn thickly (for example, the start codon in gene displays).
8. **thickEnd** - The ending position at which the feature is drawn thickly (for example, the stop codon in gene displays).
9. **itemRgb** - An RGB value of the form R,G,B (e.g. 255,0,0). If the track line *itemRgb* attribute is set to "On", this RBG value will determine the display color of the data contained in this BED line. NOTE: It is recommended that a simple color scheme (eight colors or less) be used with this attribute to avoid overwhelming the color resources of the Genome Browser and your Internet browser.
10. **blockCount** - The number of blocks (exons) in the BED line.
11. **blockSizes** - A comma-separated list of the block sizes. The number of items in this list should correspond to *blockCount*.
12. **blockStarts** - A comma-separated list of block starts. All of the *blockStart* positions should be calculated relative to *chromStart*. The number of items in this list should correspond to *blockCount*.

From the BED format specification

# Visualising in IGV



# BEDTOOLS

- High performance software package that operates on multiple interval oriented data formats: BED, GFF, SAM, BAM and VCF

- Download and install bedtools

**http://bedtools.readthedocs.org/en/latest/**

Quinlan AR and Hall IM,
*BEDTools: a flexible suite of utilities for comparing genomic features*.
**Bioinformatics**. 26, 6, (2010)

# Excellent documentation

bedtools v2.22.0 »  next  index

Bedtools is a fast, flexible toolset for genome arithmetic.

**Bedtools links**

Issue Tracker
Source @ GitHub
Old Releases @ Google Code
Mailing list @ Google Groups
Queries @ Biostar
Quinlan lab @ UVa

**Sources**

Browse source @ GitHub .

## bedtools: *a powerful toolset for genome arithmetic*

Collectively, the **bedtools** utilities are a swiss-army knife of tools for a wide-range of genomics analysis tasks. The most widely-used tools enable *genome arithmetic*: that is, set theory on the genome. For example, **bedtools** allows one to *intersect*, *merge*, *count*, *complement*, and *shuffle* genomic intervals from multiple files in widely-used genomic file formats such as BAM, BED, GFF/GTF, VCF. While each individual tool is designed to do a relatively simple task (e.g., *intersect* two interval files), quite sophisticated analyses can be conducted by combining multiple bedtools operations on the UNIX command line.

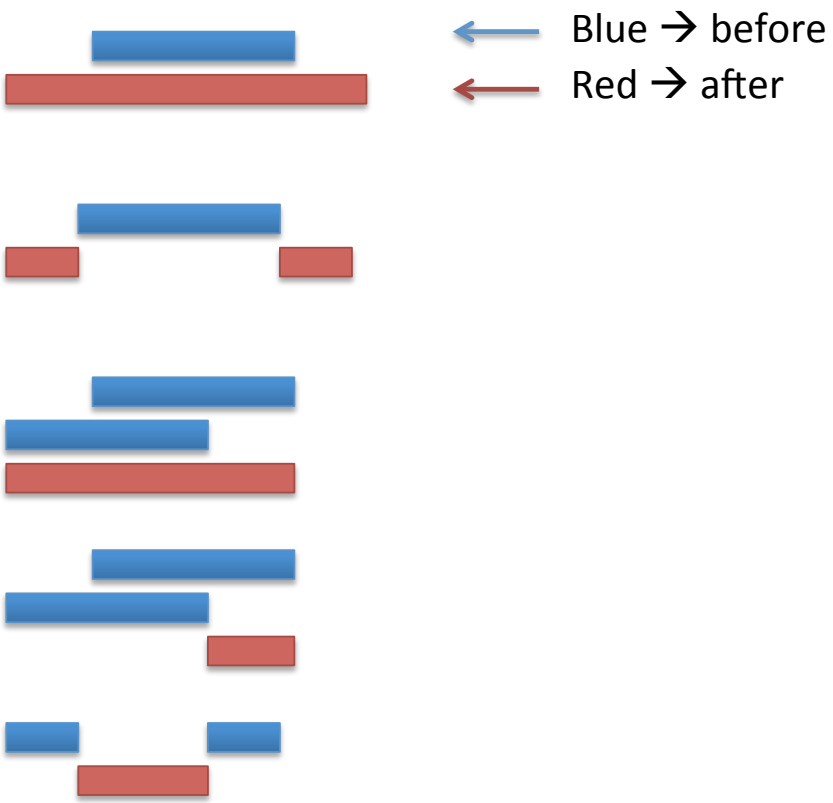
## Interesting usage examples

To whet your appetite, here are a few examples of ways in which bedtools has been used for genome research. If you have interesteding examples, please send them our way and we will add them to the list.

- Coverage analysis for targeted DNA capture. Thanks to Stephen Turner.
- Measuring similarity of DNase hypersensitivity among many cell types
- Extracting promoter sequences from a genome
- Comparing intersections among many genome interval files
- RNA-seq coverage analysis. Thanks to Erik Minikel.
- Identifying targeted regions that lack coverage. Thanks to Brent Pedersen.
- Calculating GC content for CCDS exons.

# A few BedTools operators

- slop (extend)
- flank
- merge
- subtract
- complement



# Strand Awareness

- Some tools take a **–l (left)**, **-r (right)** parameter that will have a different effect if the "stranded" mode is turned on

1. **default mode**: left, right are interpreted on the forward strand's coordinate system

2. **stranded mode**: left, right are interpreted in the transcriptional direction 5' to 3'

# How to learn Bedtoosl

Some tools may require a **genome file**, a tab delimited list of chromosome sizes

```
lec25 — ~/edu/lec25 — bash — 55×17
(env)
ialbert@grit ~/edu/lec25
$ cat demo.bed
KM034562        100     200     one     0       +
KM034562        400     500     two     0       -
(env)
ialbert@grit ~/edu/lec25
$ cat genome.txt
KM034562        18959(env)
ialbert@grit ~/edu/lec25
$ bedtools slop -i demo.bed -g genome.txt -b 50
KM034562        50      250     one     0       +
KM034562        350     550     two     0       -
(env)
ialbert@grit ~/edu/lec25
$
```

Input file

genome file

results

# Stranded mode

```
lec25 — ~/edu/lec25 — bash — 56×14
$ cat demo.bed
KM034562        100     200     one     0       +
KM034562        400     500     two     0       -
$
$ bedtools slop -i demo.bed -g genome.txt -l 10 -r 0
KM034562        90      200     one     0       +
KM034562        390     500     two     0       -
$
$ bedtools slop -i demo.bed -g genome.txt -l 10 -r 0 -s
KM034562        90      200     one     0       +
KM034562        400     510     two     0       -
$
```

It is very important to understand what happens here.
It can be occasionally feel counterintuitive

# BedTools is format aware for input

```
lec25 — ~/edu/lec25 — bash — 69×10
$ cat demo.gff
KM034562        .       .       101     200     0       +       .
KM034562        .       .       401     500     0       -       .
$ bedtools slop -i demo.gff -g genome.txt -l 10 -r 0 -s
KM034562        .       .       91      200     0       +       .
KM034562        .       .       401     510     0       -       .
$
```

It tries to keep the format the same as when  it is possible.

When it creates a new dataset with less information it may produce a different format.

# Slop vs. Flank

```
lec25 — ~/edu/lec25 — bash — 77×13
$ cat demo.gff
KM034562        .       .       101     200     0       +       .       .
KM034562        .       .       401     500     0       -       .       .
$
$ bedtools slop -i demo.gff -g genome.txt -l 10 -r 0 -s
KM034562        .       .       91      200     0       +       .       .
KM034562        .       .       401     510     0       -       .       .
$
$ bedtools flank -i demo.gff -g genome.txt -l 10 -r 0 -s
KM034562        .       .       91      100     0       +       .       .
KM034562        .       .       501     510     0       -       .       .
$
```

The best is to draw the intervals and track what each tool does

# Visualize you intervals

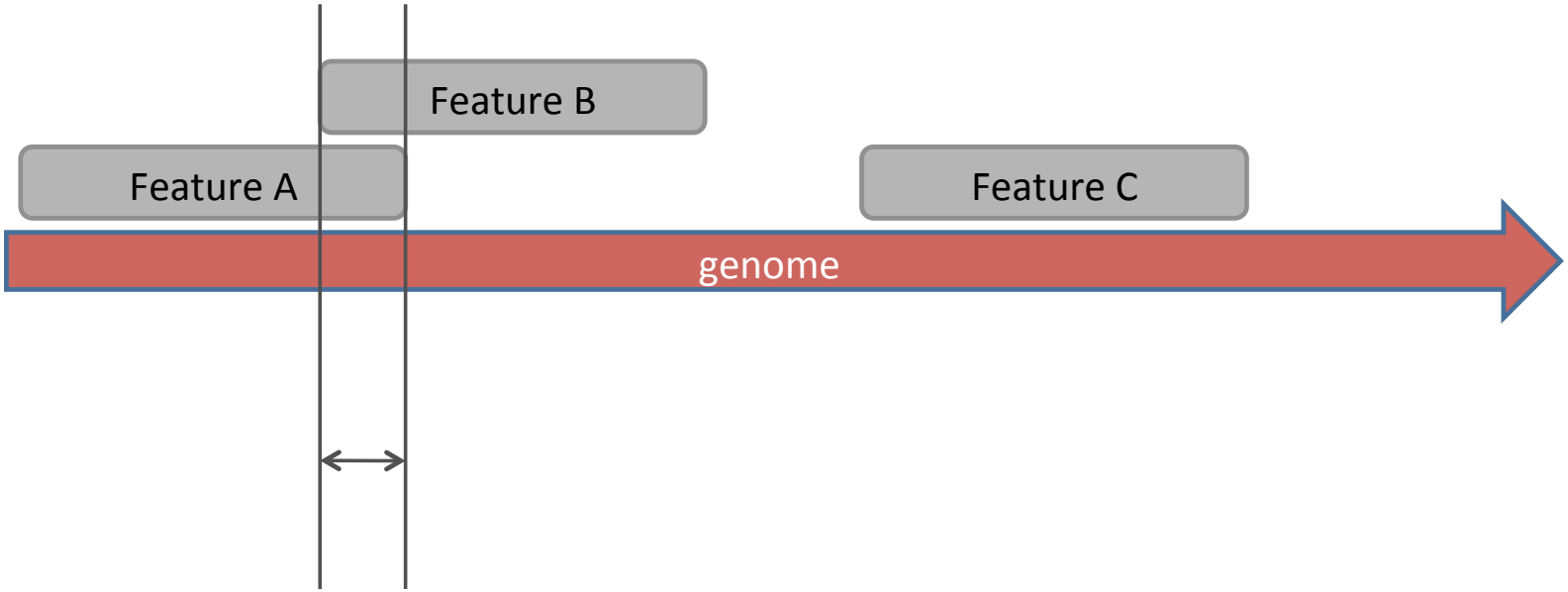

Prepare simple examples and explore what the tool does.

Pay close attention to the directionality

Think in terms of “interval operations” as they were “mathematical operations”

# Overlap/intersect two intervals

- Two features are said to overlap or intersect if they share at least one base in common.



# Basic concepts

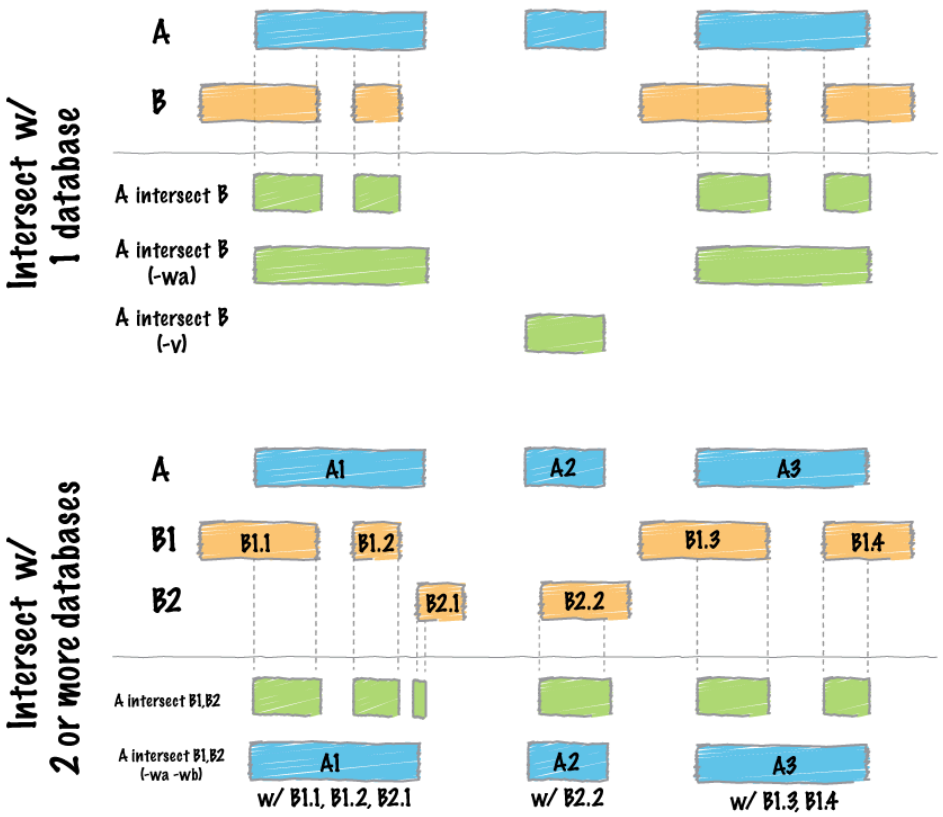
- For any operation that requires **two files** the tools will require a file **A** and file **B**
- **Each element in file A** is matched against **each element in file B**
- **File B is loaded into memory** – try to make that the **smaller** file

(for example the A file contains the the reads –B file contains the features)

# INTERSECT: TUTORIAL

## bedtools "intersect"

The `intersect` command is the workhorse of the `bedtools` suite. It compares two or more BED/BAM/VCF/GFF files and identifies all the regions in the gemome where the features in the two files overlap (that is, share at least one base pair in common).

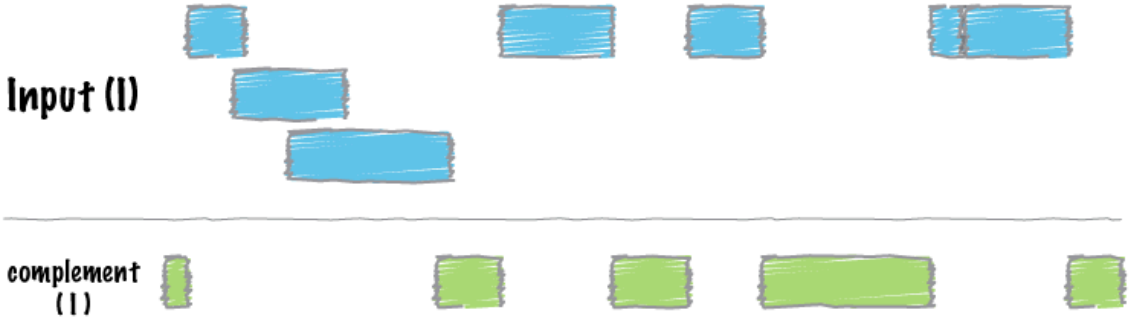



Material taught at Cold Spring Harbor summer workshops
http://quinlanlab.org/tutorials/cshl2014/bedtools.html

# Regions not covered by intervals

## bedtools "complement"

We often want to know which intervals of the genome are **NOT** "covered" by intervals in a given feature file. For example, if you have a set of ChIP-seq peaks, you may also want to know which regions of the genome are not bound by the factor you assayed. The `complement` addresses this task.



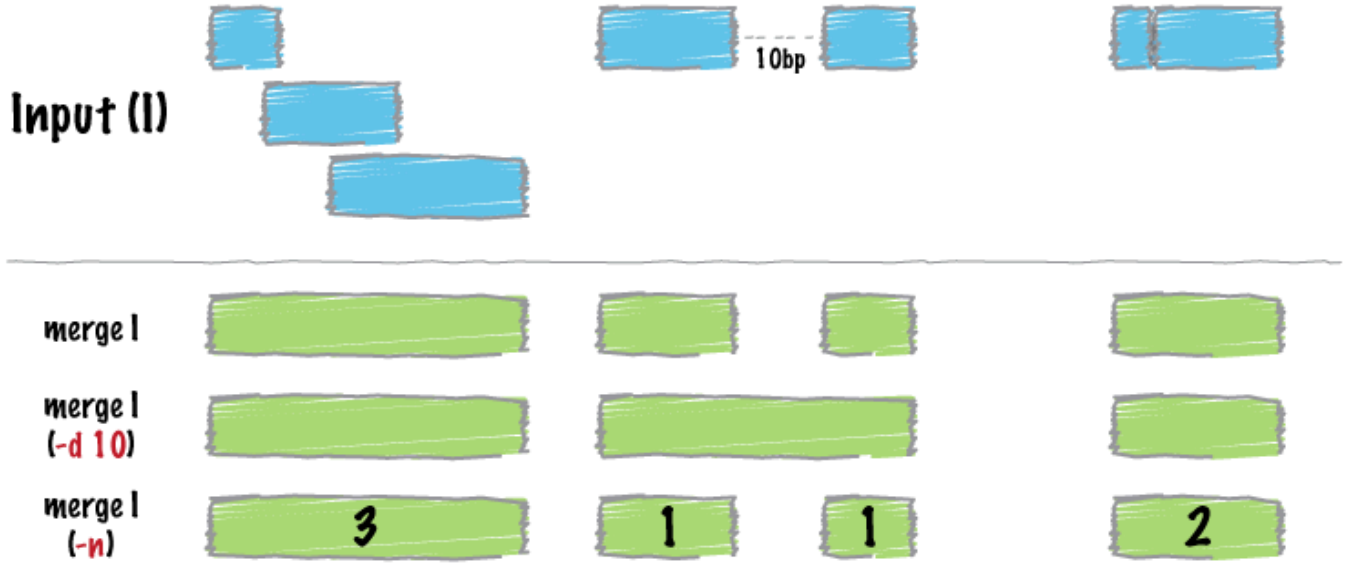

As an example, let's find all of the non-exonic (i.e., intronic or intergenic) regions of the genome. Note, to do this you need a "genome" file, which tells `bedtools` the length of each chromosome in your file. *Consider why the tool would need this information...*

```
bedtools complement -i exons.bed -g genome.txt \
> non-exonic.bed
head non-exonic.bed
chr1    0    11873
chr1    12227    12612
chr1    12721    13220
chr1    14829    14969
chr1    15038    15795
chr1    15947    16606
chr1    16765    16857
chr1    17055    17232
chr1    17368    17605
chr1    17742    17914
```

# Merging overlapping intervals

## bedtools "merge"

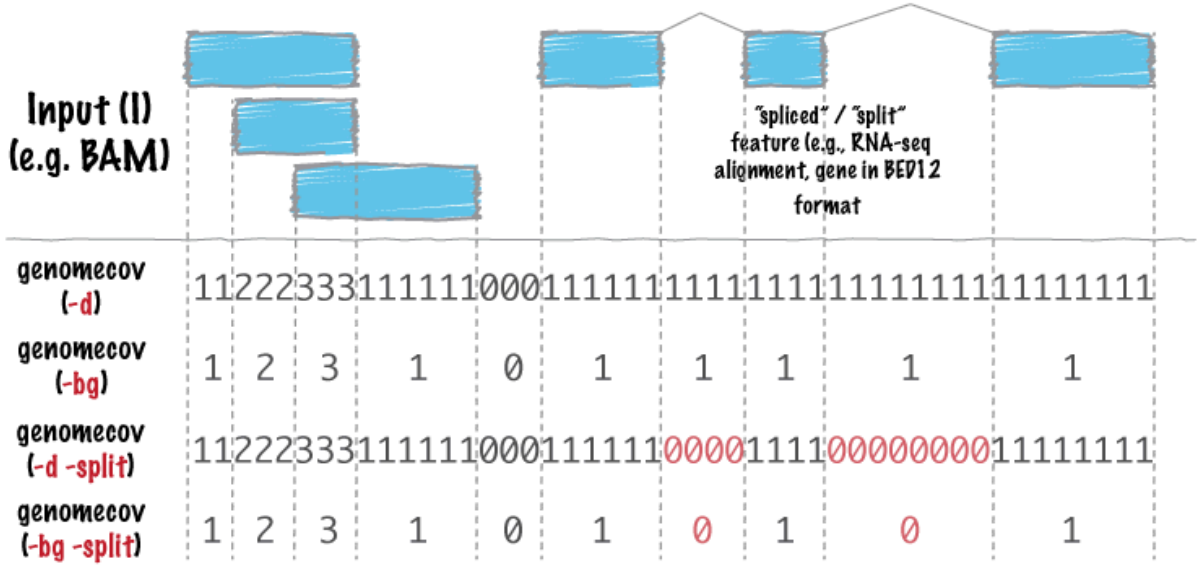
Many datasets of genomic features have many individual features that overlap one another (e.g. aligments from a ChiP seq experiment). It is often useful to just cobine the overlapping into a single, contiguous interval. The bedtools `merge` command will do this for you.



# Genome wide coverage

## bedtools "genomecov"

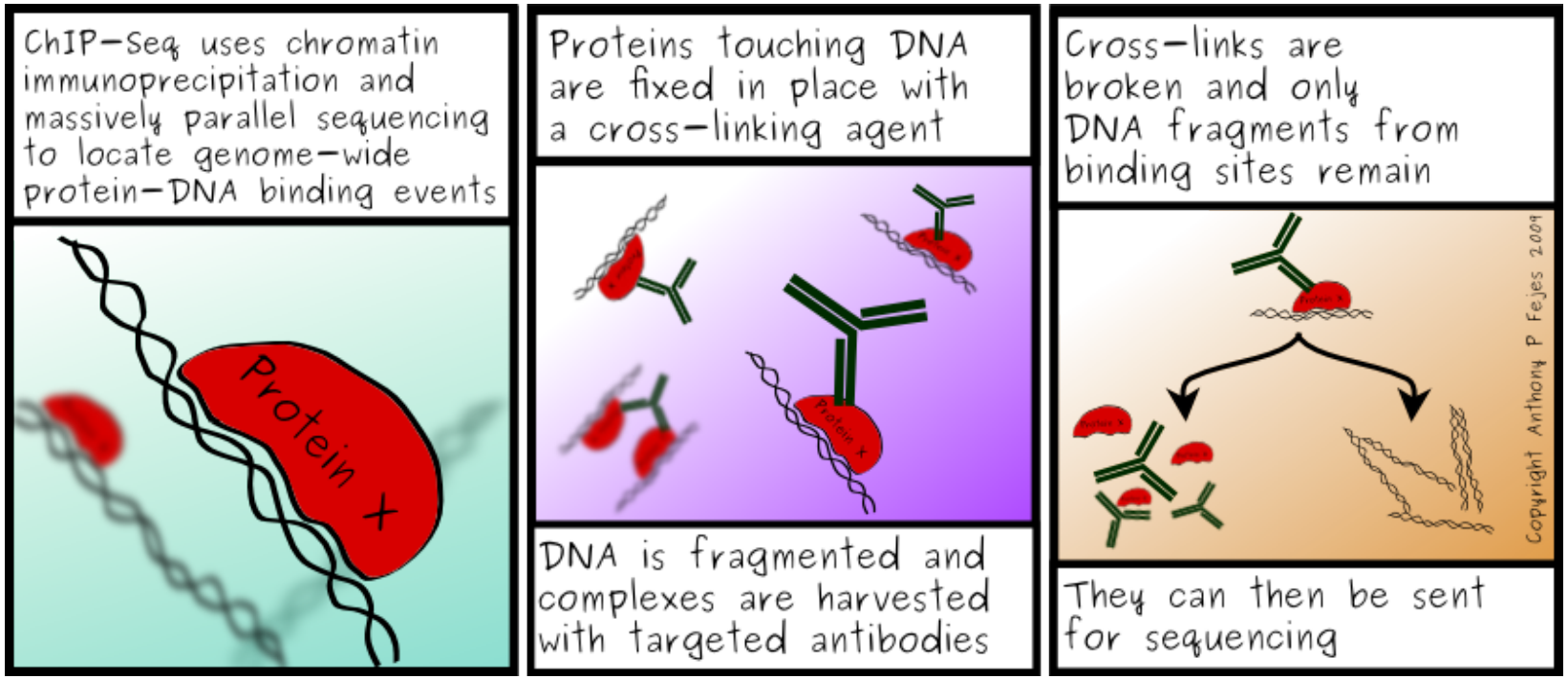
For many analyses, one wants to measure the genome wide coverage of a feature file. For example, we often want to know what fraction of the genome is covered by 1 feature, 2 features, 3 features, etc. This is frequently crucial when assessing the "uniformity" of coverage from whole-genome sequencing. This is done with the versatile `genomecov` tool.




As an example, let's produce a histogram of coverage of the exons throughout the genome. Like the `merge` tool, `genomecov` requires pre-sorted data. It also needs a genome file as above.

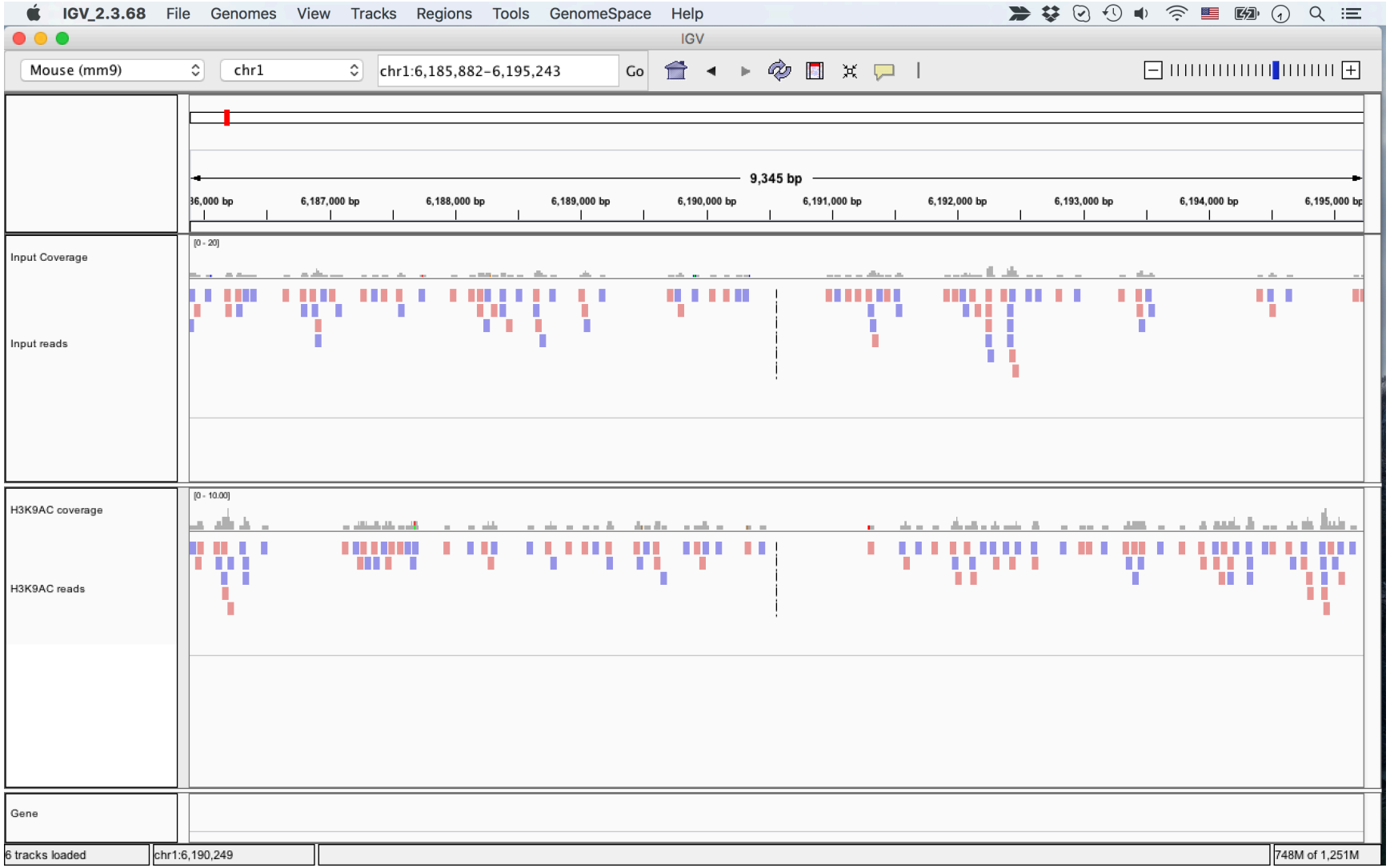
- Creates BedGraph file which is too large for bigger genomes to load into memory
- Convert to BigWig using bedGraphToBigWig (UCSC toolkit). Only parts of the file that are required are loaded

# Example: ChIP-Seq



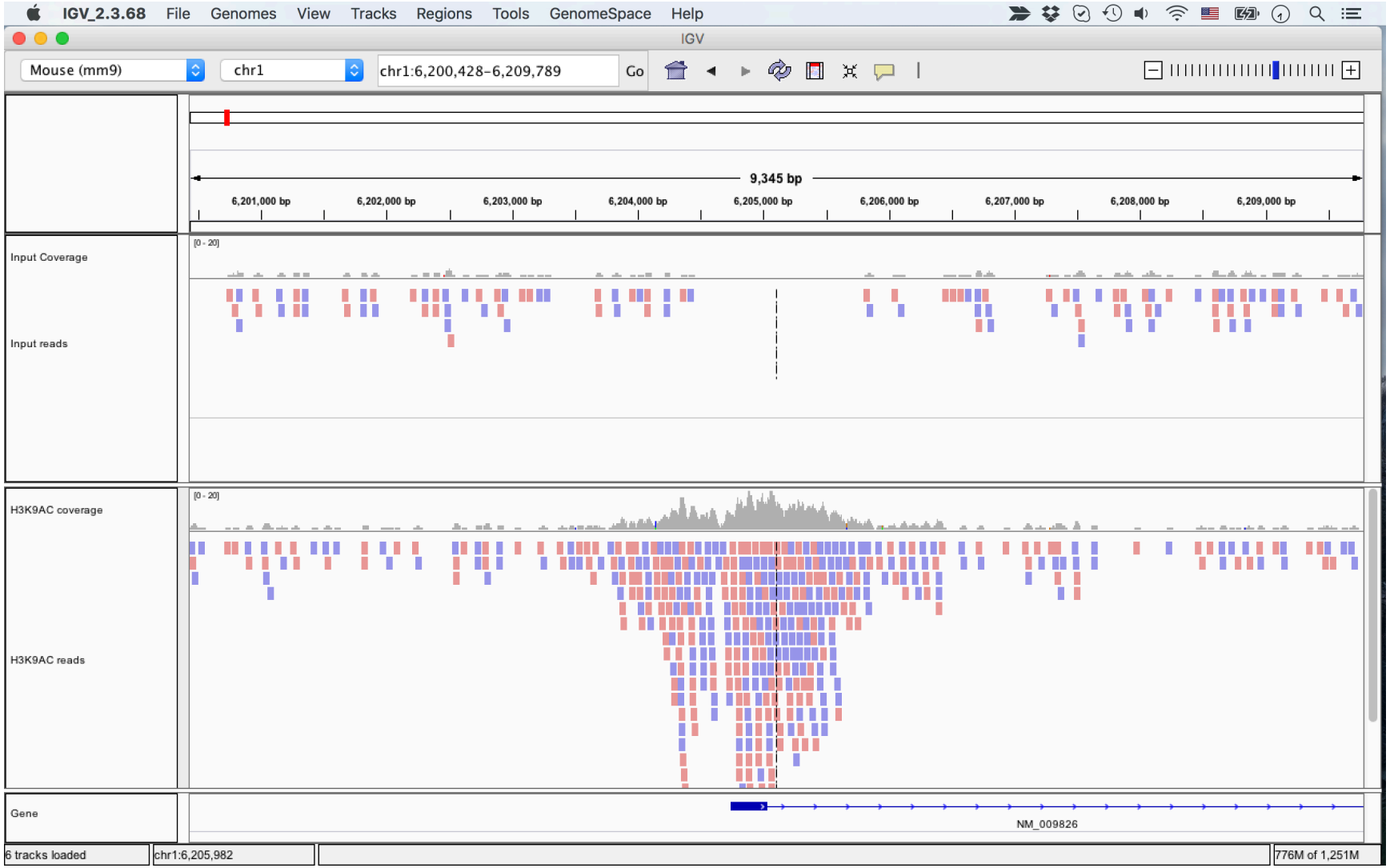

- Reference genome: mouse (mm9)
- Sequencing: input and H3K9Ac ChIP

# ChIP-Seq: input data




- No enrichment in ChiP-Seq reads compared to input

# ChIP-Seq: input data




- Clear enrichment. Is this significant, where does the enrichment start and end

# ChIP-Seq: calling peaks with MACS

## Usage of MACS2

```
macs2 [-h] [--version]
          {callpeak,filterdup,bdgpeakcall,bdgcmp,randsample,bdgdiff,bdgbroadcall}
```

| | |
|---|---|
| **Example for regular peak calling:** | `macs2 callpeak -t ChIP.bam -c Control.bam -f BAM -g hs -n test -B -q 0.01` |
| **Example for broad peak calling:** | `macs2 callpeak -t ChIP.bam -c Control.bam --broad -g hs --broad-cutoff 0.1` |

There are seven major functions available in MACS serving as sub-commands.

| | |
|---|---|
| **callpeak:** | Main MACS2 Function to Call peaks from alignment results. |
| **bdgpeakcall:** | Call peaks from bedGraph output. |
| **bdgbroadcall:** | Call broad peaks from bedGraph output. |
| **bdgcmp:** | Deduct noise by comparing two signal tracks in bedGraph. |
| **bdgdiff:** | Differential peak detection based on paired four bedgraph files. |
| **filterdup:** | Remove duplicate reads at the same position, then convert acceptable format to BED format. |
| **predictd:** | Predict d or fragment size from alignment results. |
| **pileup:** | Pileup aligned reads with a given extension size (fragment size or d in MACS language). Note there will be no step for duplicate reads filtering or sequencing depth scaling, so you may need to do certain post- processing. |
| **randsample:** | Randomly sample number/percentage of total reads. |
| **refinepeak:** | (Experimental) Take raw reads alignment, refine peak summits and give scores measuring balance of forward- backward tags. Inspired by SPP. |

- https://github.com/taoliu/MACS

# ChIP-Seq: visualising peaks in IGV


- MACs outputs peaks as BED file

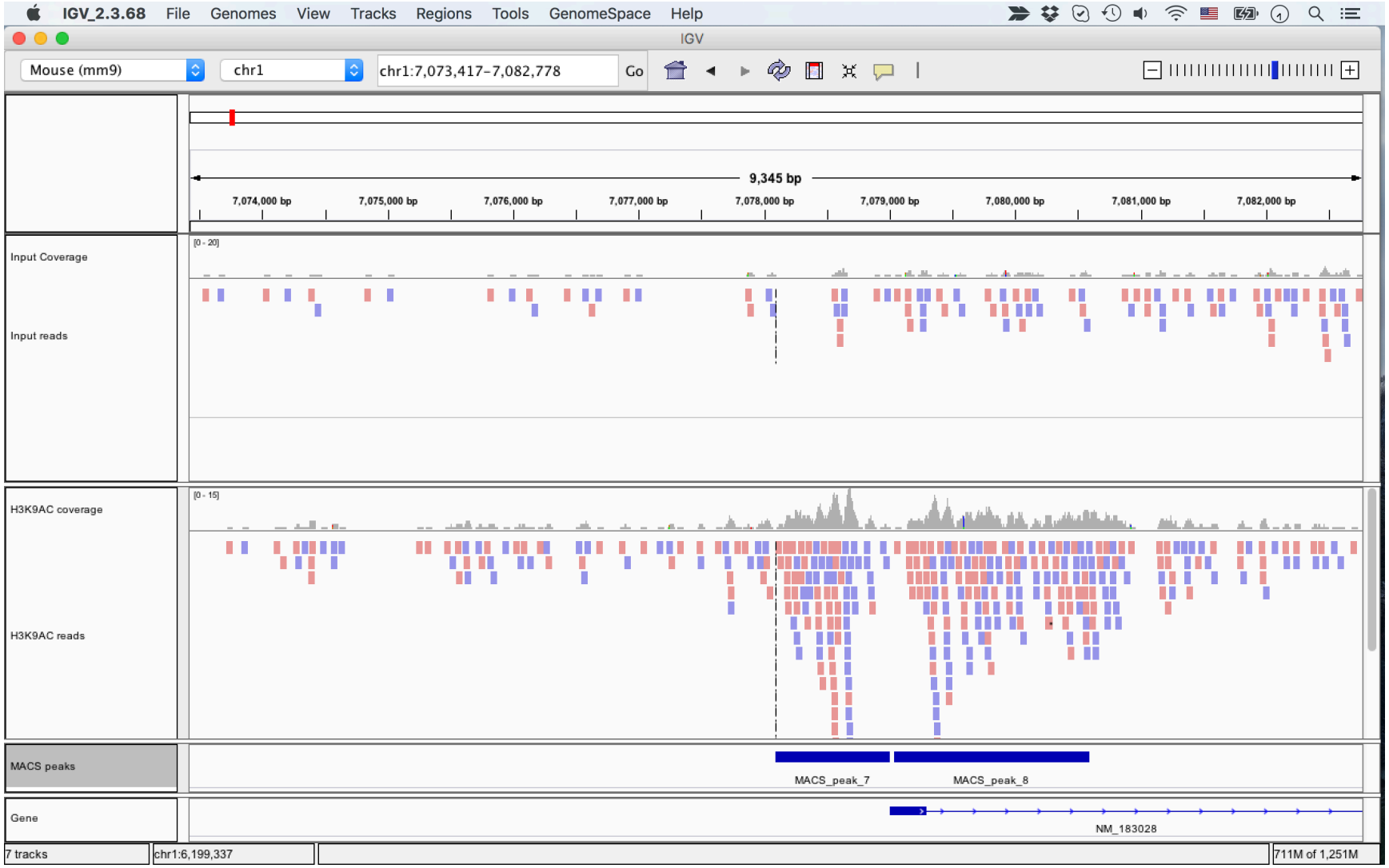
# Navigating intervals in IGV


- How to quickly navigate to next peak?

# Navigating intervals in IGV

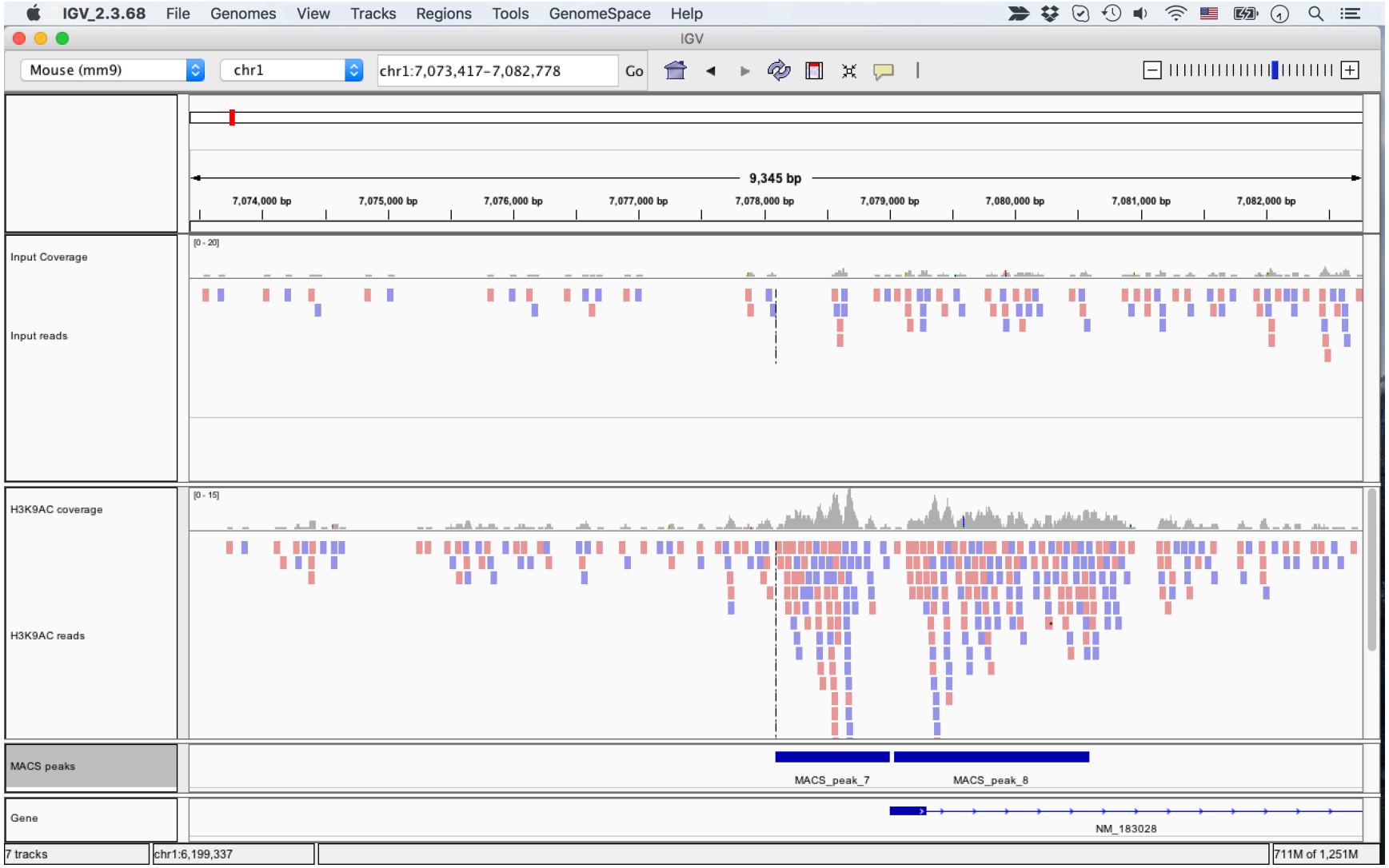

- Select BED track and pres Ctrl + F (forward)

# Navigating intervals in IGV




- Select BED track and pres Ctrl + F (forward)

# Navigating intervals in IGV

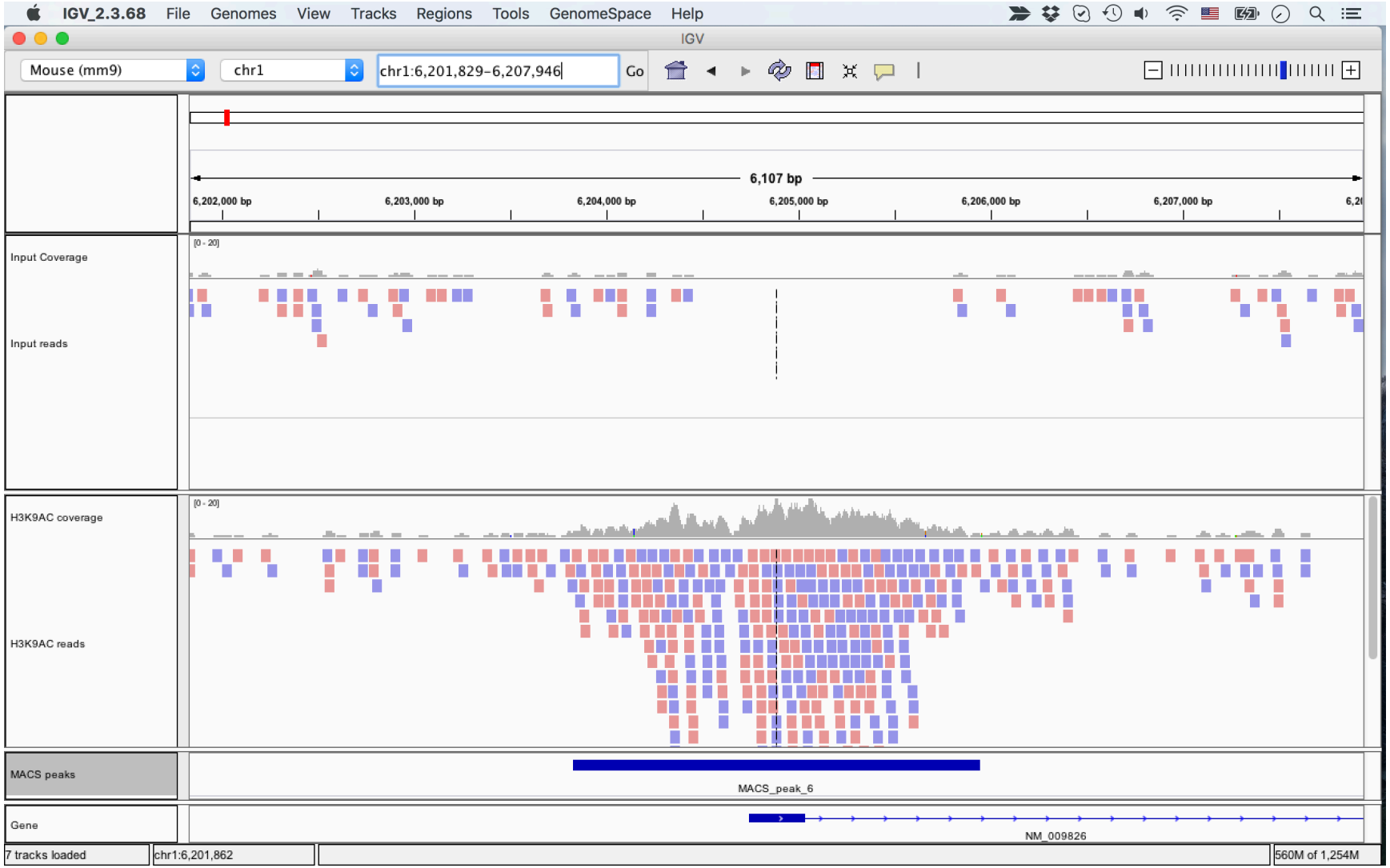



- Select BED track and pres Ctrl + B (backward) to go to back

# Navigating intervals in IGV


- Select BED track and pres Ctrl + B (backward) to go to back

# Navigating using feature/gene names

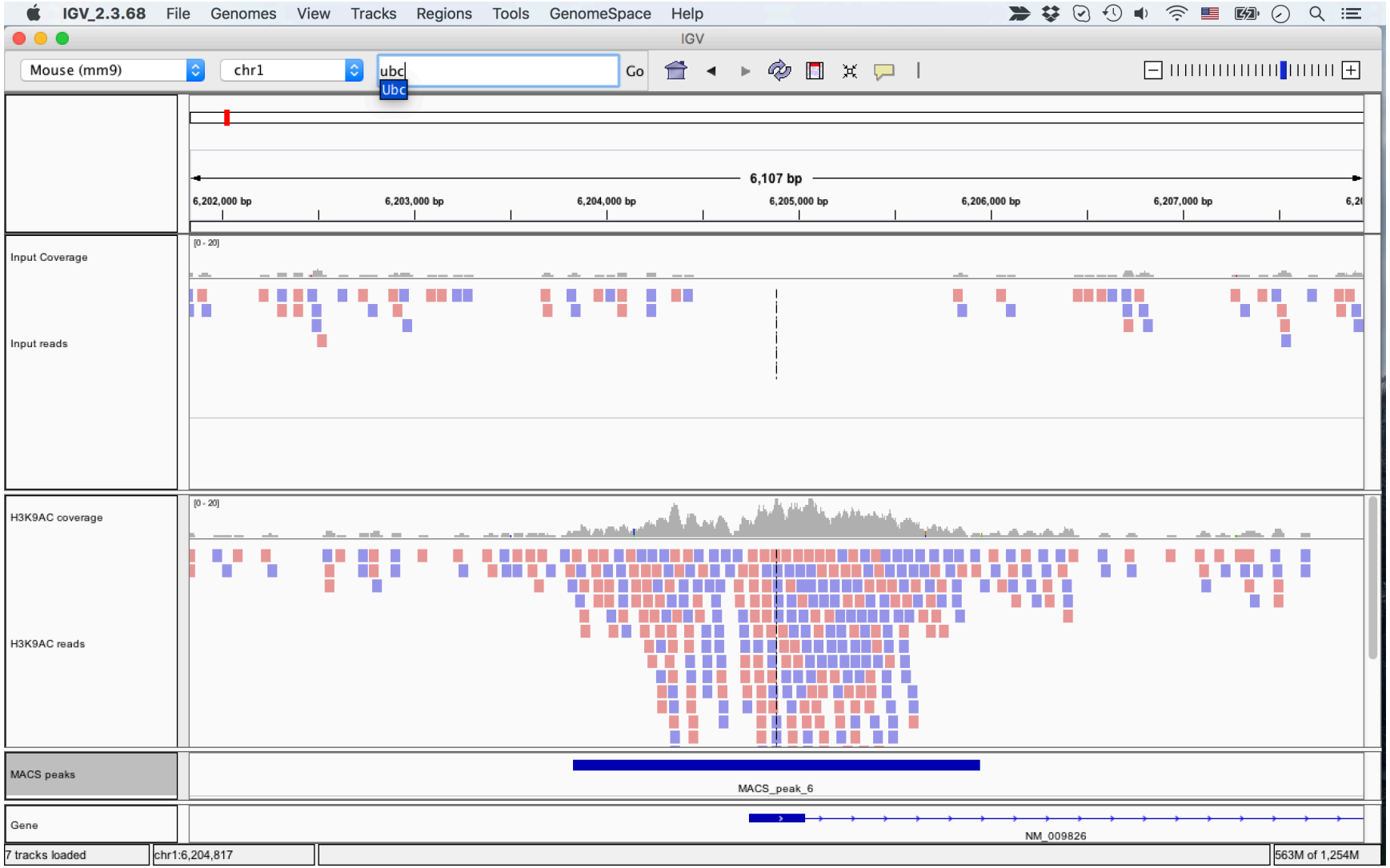



- Navigate to the gene "UBC"

# Navigating using feature/gene names

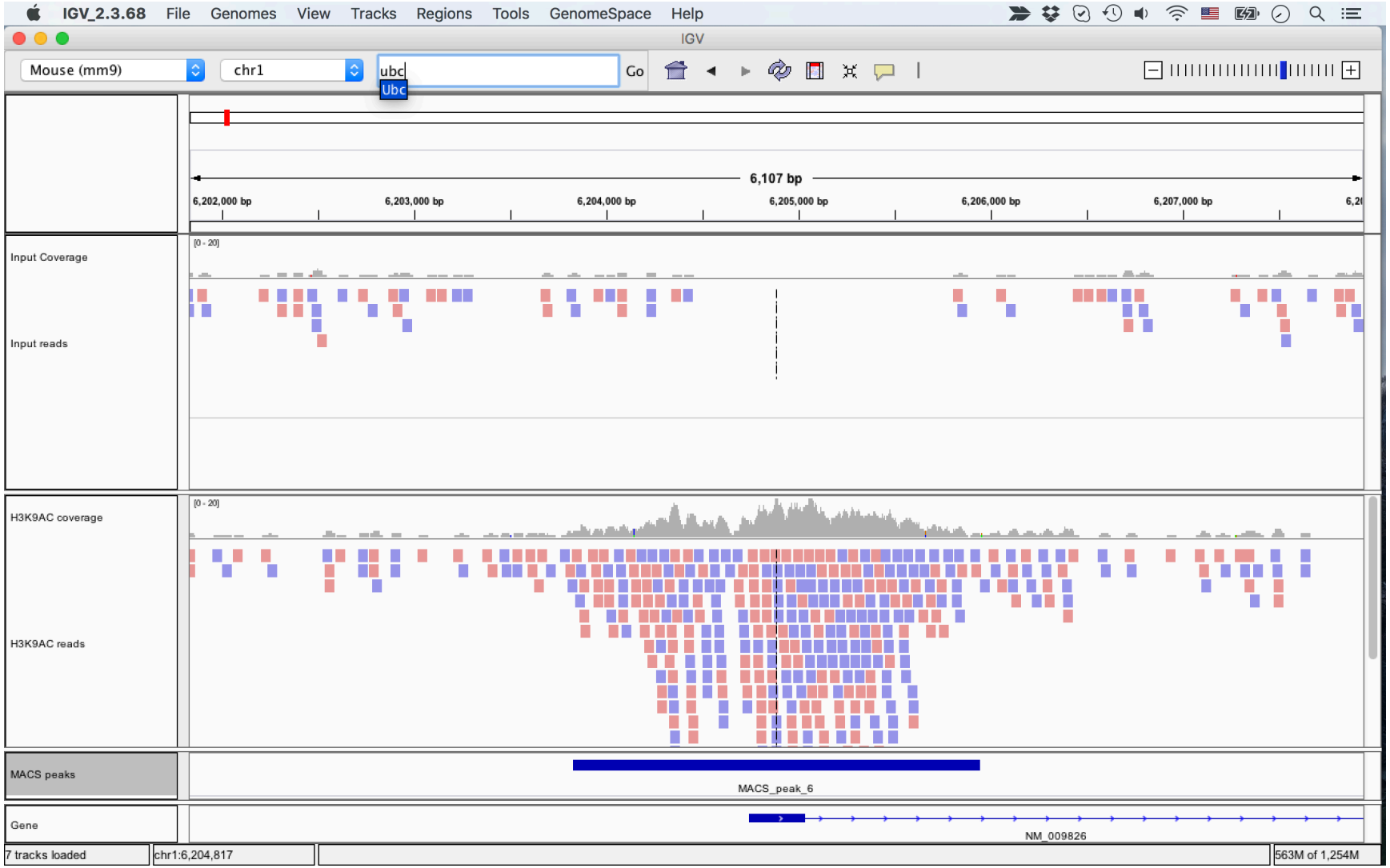

- Navigate to the gene "UBC"

# Navigating using feature/gene names



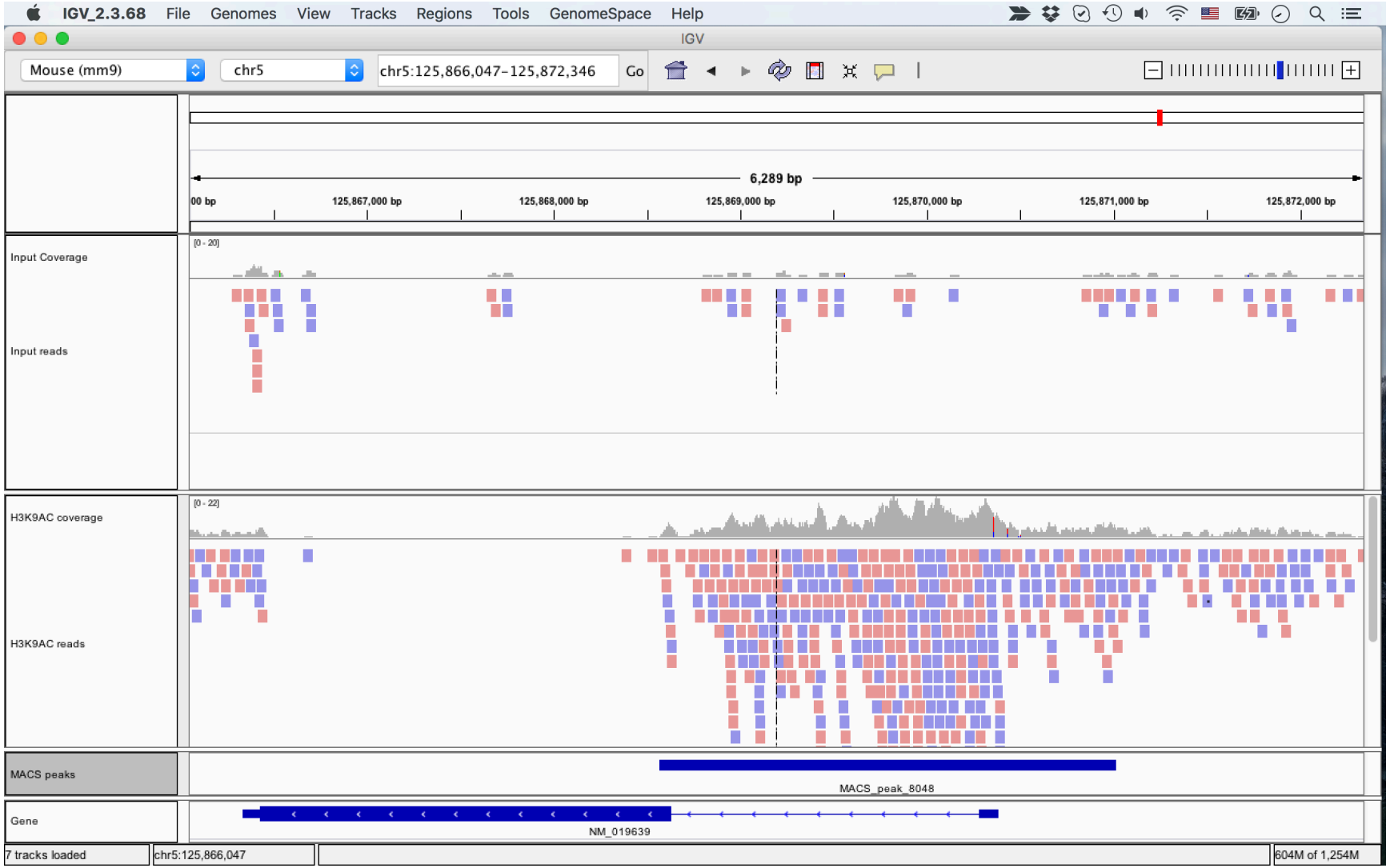

- Navigate to the gene "UBC"

# Navigating using feature/gene names




- Navigate to the gene "UBC"

# Navigating using feature/gene names




- Navigate to the gene "UBC"

# ChIP-Seq: bedtools use cases

- Find all genes/promoters that overlap with a peak
  - bedtools intersect with -wa or -u option
- Merge peaks in close proximity
  - bedtools merge with -d option
- For each peak, find genes that has the closest 5' end
  - bedtools closest

# From Sequence data to genotypes

- A common sequencing workflow

| Sequencing reads → | Alignments → | Variant calls |
|---|---|---|
| FASTQ | SAM/BAM | VCF |
| a list of short sequences | a list of short sequences and where they are in the genome | a list of locations in the genome and what the base is at each |

# How to call variants?

- Naive variant calling
  - Check all the reads that cover base chr1:291
  - Add up the bases at chr1:291
  - e.g. 10 A's, 2 G's
    - Is this an A/G heterozygous site or two sequencing errors?
- Actual variant callers
  - Estimate likelihood of a variant site vs a sequencing error
    - Sequencing error rate
    - Quality scores

Note: it is not always obvious what the underlying assumptions of a snp caller are. Especially when used for genomes other than human/mouse. These are by far the most studied and customized for.

# Variant callers

- samtools/bcftools
- GATK: The Genome Analysis Toolkit
- VarScan
  - samtools mpileup -f reference.fasta myData.bam | java -jar VarScan.v2.2.jar pileup2snp –strand-filter 0 –output-vcf –min-var-freq 0.8 –min-coverage 10 –variants 1
- Many more

# VCF: Variant Call Format

- Represent a list of locations and the variant call at each
  - Simple, right?
- Yes and no.
  - Simple foundation
    - Location and base
  - Complex “bonus features”
    - Indels, structural variants, etc.
    - Multiple samples
    - Haplotype phasing

# VCF: Variant Call Format

## NAME

vcf – Variant Call Format

## DESCRIPTION

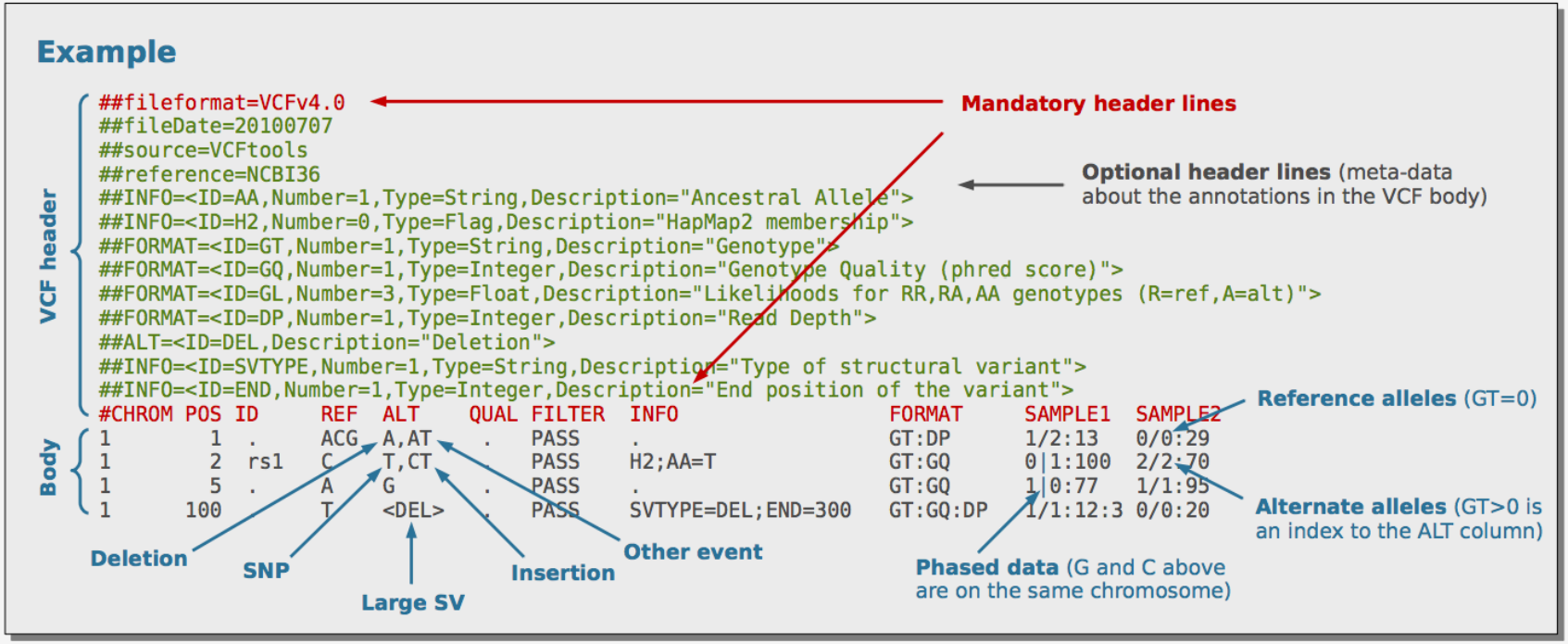
The Variant Call Format (VCF) is a TAB-delimited format with each data line consisting of the following fields:

| | | |
|---|---|---|
| 1 | CHROM | CHROMosome name |
| 2 | POS | the left-most POSition of the variant |
| 3 | ID | unique variant IDentifier |
| 4 | REF | the REFerence allele |
| 5 | ALT | the ALTernate allele(s) (comma-separated) |
| 6 | QUAL | variant/reference QUALity |
| 7 | FILTER | FILTERs applied |
| 8 | INFO | INFOrmation related to the variant (semicolon-separated) |
| 9 | FORMAT | FORMAT of the genotype fields (optional; colon-separated) |
| 10+ | SAMPLE | SAMPLE genotypes and per-sample information (optional) |

The following table gives the **INFO** tags used by samtools and bcftools.

| | |
|---|---|
| **AF1** | Max-likelihood estimate of the site allele frequency (AF) of the first ALT allele (double) |
| **DP** | Raw read depth (without quality filtering) (int) |
| **DP4** | # high-quality reference forward bases, ref reverse, alternate for and alt rev bases (int[4]) |
| **FQ** | Consensus quality. Positive: sample genotypes different; negative: otherwise (int) |
| **MQ** | Root-Mean-Square mapping quality of covering reads (int) |
| **PC2** | Phred probability of AF in group1 samples being larger (,smaller) than in group2 (int[2]) |
| **PCHI2** | Posterior weighted chi^2 P-value between group1 and group2 samples (double) |
| **PV4** | P-value for strand bias, baseQ bias, mapQ bias and tail distance bias (double[4]) |
| **QCHI2** | Phred-scaled PCHI2 (int) |
| **RP** | # permutations yielding a smaller PCHI2 (int) |
| **CLR** | Phred log ratio of genotype likelihoods with and without the trio/pair constraint (int) |
| **UGT** | Most probable genotype configuration without the trio constraint (string) |
| **CGT** | Most probable configuration with the trio constraint (string) |
| **VDB** | Tests variant positions within reads. Intended for filtering RNA-seq artifacts around splice sites (float) |
| **RPB** | Mann-Whitney rank-sum test for tail distance bias (float) |
| **HWE** | Hardy-Weinberg equilibrium test (Wigginton et al) (float) |

# Variant Call Format

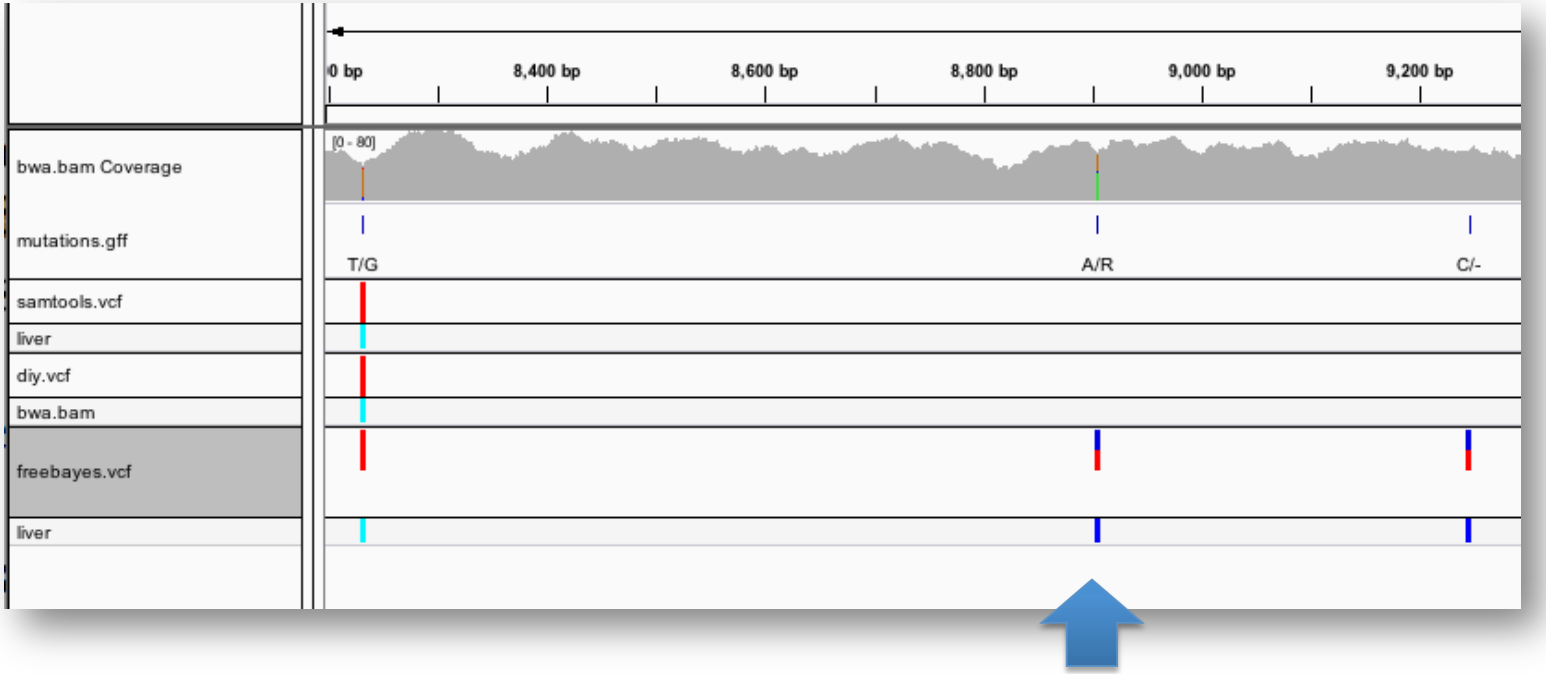


# VCF and BCF

- The same relationship as SAM and BAM formats

- BCF – binary, compressed VCF – much smaller but need to be operated on with **bcftools**

# How to characterize the effect of mutations?




What effect could it have?

# SnpEff

SNPEFF
- Home
- SnpEff description
- Download
- Features
- SnpSift
- Paper & Citing
- Who uses SnpEff?
- Integration

HELP
- Bug reports
- Feature requests
- Asking for help
- About

DOCUMENTATION
- SnpEff manual
- SnpSift manual
- Usage Examples

## SnpEff

Genetic variant annotation and effect prediction toolbox.

Download SnpEff

Latest version 4.0 E (2014-09-13)

Requires Java 1.7

### SnpEff

Genetic variant annotation and effect prediction toolbox. It annotates and predicts the effects of variants on genes (such as amino acid changes). Features:

- Supports over **20,000 genomes**.
- **Cancer variants analysis**
- **GATK** compatible ( `-o gatk` )
- **HGVS** notation
- **Sequence Ontology** standardized terms

View details »

### Version 4.0

Major improvements and support for standards:

- HGVS notations
- Sequence Ontology terms
- Easier to use
- SnpEff downloads databases automatically
- Automatic third party databases downloads
- Support for GRCh38
- Support for Ebola Zaire Virus (2014 West Africa outbreak)

View details »

### SnpSift

SnpSift helps filtering and manipulating genomic annotated files (VCF). Once you annotated your files using SnpEff, you can use SnpSift to help you filter large genomic datasets in order to find the most significant variants

View details »

# SnpEff report

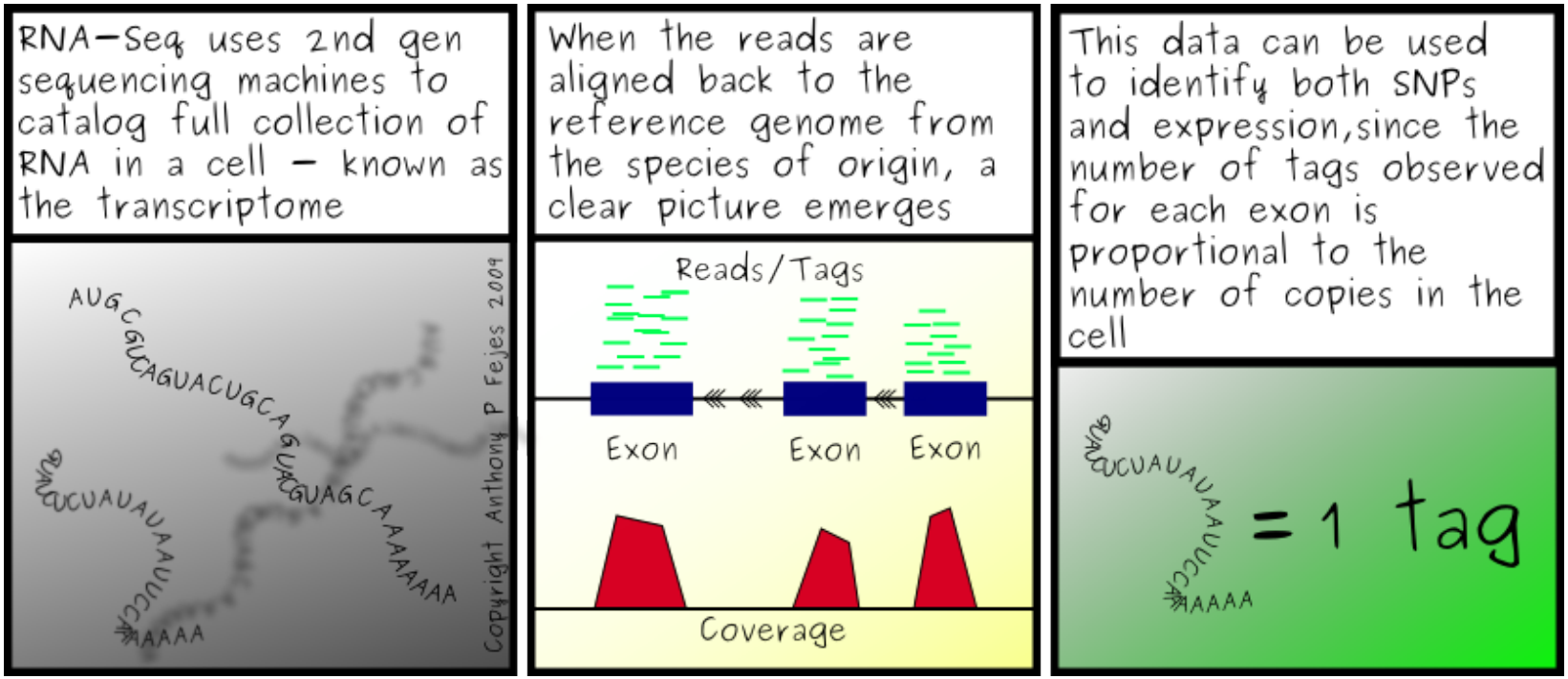
**Type**

| Type (alphabetical order) | Count | Percent |
|---|---|---|
| DOWNSTREAM | 2,093 | 1.766% |
| INTERGENIC | 26,314 | 22.204% |
| INTRAGENIC | 78 | 0.066% |
| INTRON | 54,238 | 45.767% |
| NON_SYNONYMOUS_CODING | 237 | 0.2% |
| NON_SYNONYMOUS_START | 1 | 0.001% |
| SPLICE_SITE_DONOR | 4 | 0.003% |
| START_GAINED | 57 | 0.048% |
| STOP_GAINED | 3 | 0.003% |
| STOP_LOST | 1 | 0.001% |
| SYNONYMOUS_CODING | 378 | 0.319% |
| TRANSCRIPT | 32,163 | 27.14% |
| UPSTREAM | 2,102 | 1.774% |
| UTR_3_PRIME | 690 | 0.582% |
| UTR_5_PRIME | 149 | 0.126% |

**Region**

| Type (alphabetical order) | Count | Percent |
|---|---|---|
| DOWNSTREAM | 2,093 | 1.766% |
| EXON | 620 | 0.523% |
| INTERGENIC | 26,314 | 22.204% |
| INTRON | 54,238 | 45.767% |
| NONE | 32,241 | 27.206% |
| SPLICE_SITE_DONOR | 4 | 0.003% |
| UPSTREAM | 2,102 | 1.774% |
| UTR_3_PRIME | 690 | 0.582% |
| UTR_5_PRIME | 206 | 0.174% |

# RNA-Seq


RNA-Seq uses 2nd gen sequencing machines to catalog full collection of RNA in a cell – known as the transcriptome
AUGCGUCAGUACUGCAGUAGUAGCAAAAAAA
UAGUCUAUAUAAUUCCAAAAA
Copyright Anthony P Fejes 2009
When the reads are aligned back to the reference genome from the species of origin, a clear picture emerges
Reads/Tags
Exon
Exon
Exon
Coverage
This data can be used to identify both SNPs and expression, since the number of tags observed for each exon is proportional to the number of copies in the cell
UAGUCUAUAUAAUUCCAAAAA
= 1 tag

# RNA-Seq pipeline

There are three major, quite independent steps

1. **Alignment** → produces a BAM file

2. **Quantitation (abundance estimation)** starts with a BAM file produces tabular files

3. **Differential Expression Analysis (statistical inference)** → statistically significant changes between samples

4. **Result Annotation** → what is the biological significance of results

# RNA-Seq considerations

- Annotation information lacking:
  - Missing altogether → transcriptome assembly
  - Incomplete → new transcripts based on known exons
- Transcript level versus exon level analysis

# RNA-Seq alignment

1. Align against a known transcriptome:

   – **good**: efficient, well defined answers
   – **bad**: unable to discover novel transcripts, may align reads that would map better in noncoding regions

2. Align against genome:

   – **good:** discover novel transcripts
   – **bad**: more false positives, more uncertainty

   Some (most) methods try to make use of a combination of both

# Align to transcripts

Simplest approach:

- Produce  transcripts sequences and treat them as the “reference”
- Use an aligner to align against this reference
- Post-process the results

# Align to the genome

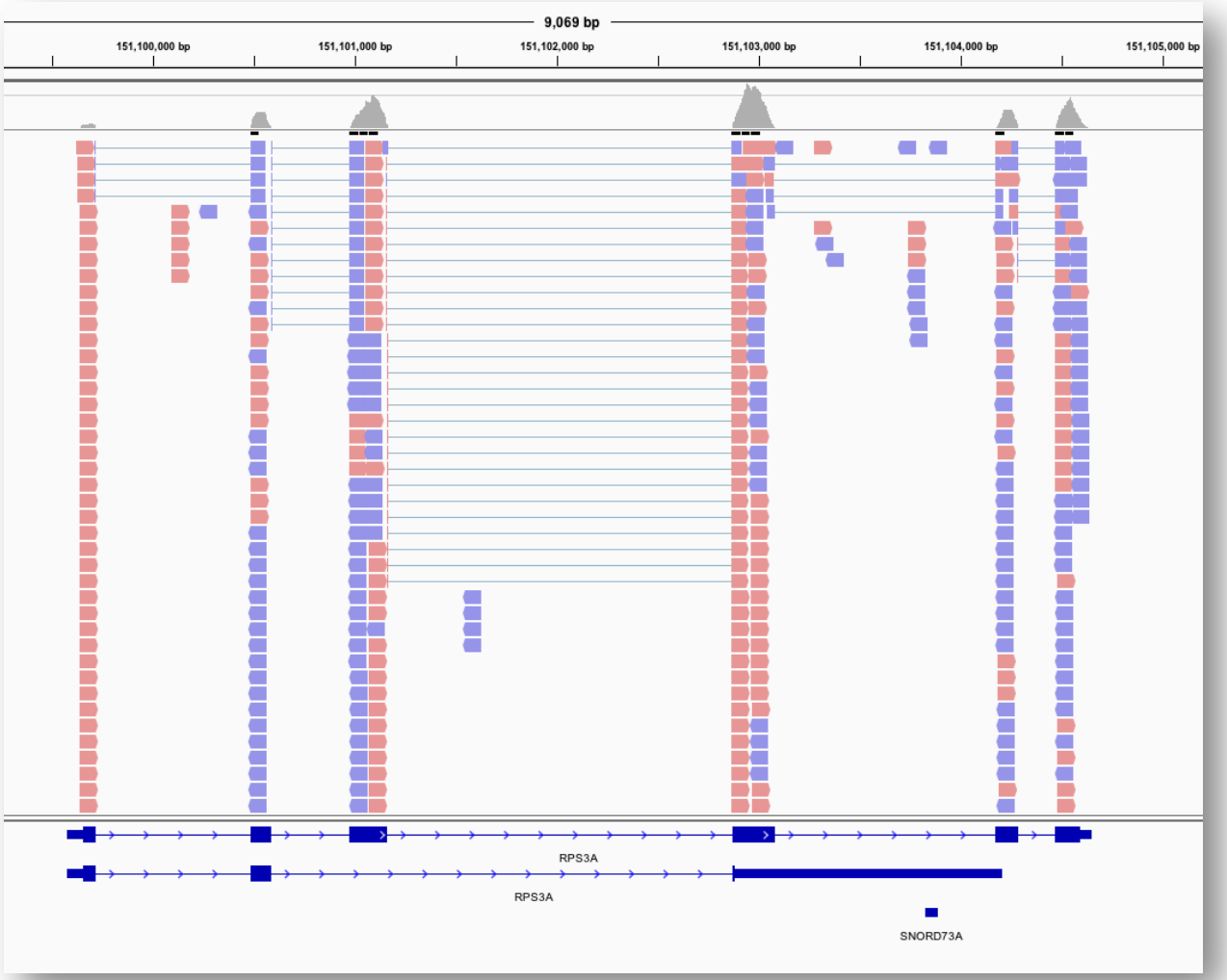
Typical methodology implemented internally by tools

1. Separate reads (read-pairs) that map "correctly" to exonic locations

2. Re-align reads that did not map in step 1 to "potential" junction sites

   – create a putative transcriptome by fusing sequences at the border of mapped reads
   – identify intron splicing indictor base pairs: GT --- AG, etc.
   – train machine learning algorithms to predict junction sites

# Spliced alignments in IGV



## Alignment

- Tophat, STAR, SubRead, RSEM

## Feature Counting

- Cuffdiff, htseq, featureCount, eXpress, corset,

**NOTE:** most read counters need data **sorted by read na**me (!) not position. Most aligners produce data in this format by default. But remember that **bam** files sorted by read name cannot be visualized in IGV! You may need two bam files for each dataset.

## Differential Expression Analysis (statistics)

- Cuffdiff, DESeq, DESeq2, edgeR, EBSeq

PROTOCOL

# Count-based differential expression analysis of RNA sequencing data using R and Bioconductor

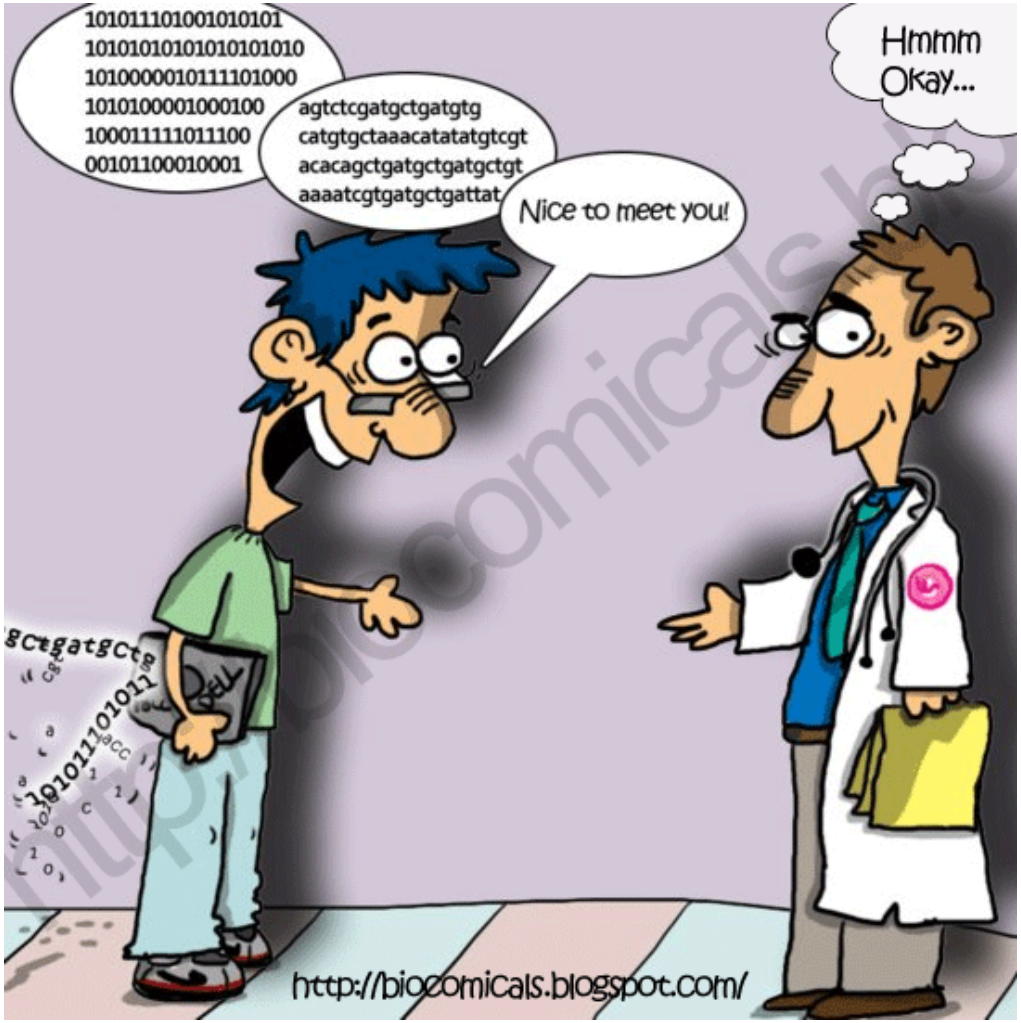
Simon Anders[1], Davis J McCarthy[2,3], Yunshun Chen[4,5], Michal Okoniewski[6], Gordon K Smyth[4,7], Wolfgang Huber[1] & Mark D Robinson[8,9]

[1]Genome Biology Unit, European Molecular Biology Laboratory, Heidelberg, Germany. [2]Department of Statistics, University of Oxford, Oxford, UK. [3]Wellcome Trust Centre for Human Genetics, University of Oxford, Oxford, UK. [4]Bioinformatics Division, Walter and Eliza Hall Institute, Parkville, Victoria, Australia. [5]Department of Medical Biology, University of Melbourne, Melbourne, Victoria, Australia. [6]Functional Genomics Center UNI ETH, Zurich, Switzerland. [7]Department of Mathematics and Statistics, University of Melbourne, Melbourne, Victoria, Australia. [8]Institute of Molecular Life Sciences, University of Zurich, Zurich, Switzerland. [9]SIB Swiss Institute of Bioinformatics, University of Zurich, Zurich, Switzerland. Correspondence should be addressed to M.D.R. (mark.robinson@imls.uzh.ch) or W.H. (whuber@embl.de).

Published online 22 August 2013; doi:10.1038/nprot.2013.099

**RNA sequencing (RNA-seq) has been rapidly adopted for the profiling of transcriptomes in many areas of biology, including studies into gene regulation, development and disease. Of particular interest is the discovery of differentially expressed genes across different conditions (e.g., tissues, perturbations) while optionally adjusting for other systematic factors that affect the data-collection process. There are a number of subtle yet crucial aspects of these analyses, such as read counting, appropriate treatment of biological variability, quality control checks and appropriate setup of statistical modeling. Several variations have been presented in the literature, and there is a need for guidance on current best practices. This protocol presents a state-of-the-art computational and statistical RNA-seq differential expression analysis workflow largely based on the free open-source R language and Bioconductor software and, in particular, on two widely used tools, DESeq and edgeR. Hands-on time for typical small experiments (e.g., 4–10 samples) can be <1 h, with computation time <1 d using a standard desktop PC.**

# Thanks


10101110100101010101
10101010101010101010
10100000101111101000
10101000010001000100
10001111101110 0
00101100010001
agtctcgatgctgatgtg
catgtgctaaacatatatgtcgt
acacagctgatgctgatgctgt
aaaatcgtgatgctgattat
Nice to meet you!
Hmmm
Okay...
http://biocomicals.blogspot.com/