

# Genome arithmetic

what, why, how, and how not

Aaron Quinlan  
BIOCH 5080  
Mar. 20, 2013  
[arq5x@virginia.edu](mailto:arq5x@virginia.edu)  
[quinlanlab.org](http://quinlanlab.org)

# Goals of today's lecture

- Overview of genome features
- What is genome arithmetic?
- Genome arithmetic approaches
- The UCSC “binning” algorithm
- An introduction to BEDTools
- Some real world examples

# What should you do to reinforce the lecture material?

- Recreate examples in lecture
- Read BEDTools docs [1]
- Watch Galaxy screencasts [2]
- Download genome features from UCSC and use BEDTools on the command line
- Review common file formats [3]

[1] <http://bedtools.readthedocs.org>

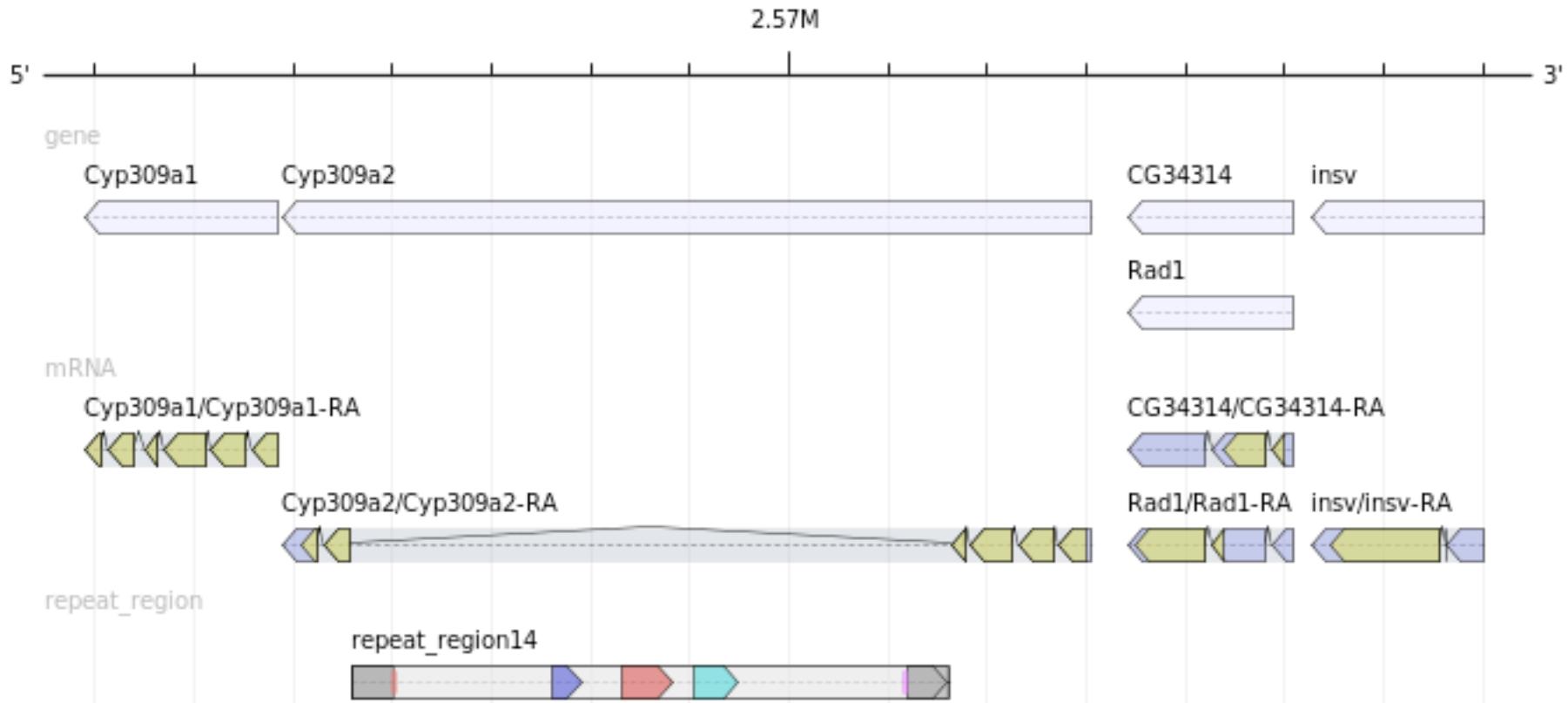
[2] <http://main.g2.bx.psu.edu/>

[3] <http://genome.ucsc.edu/FAQ/FAQformat>

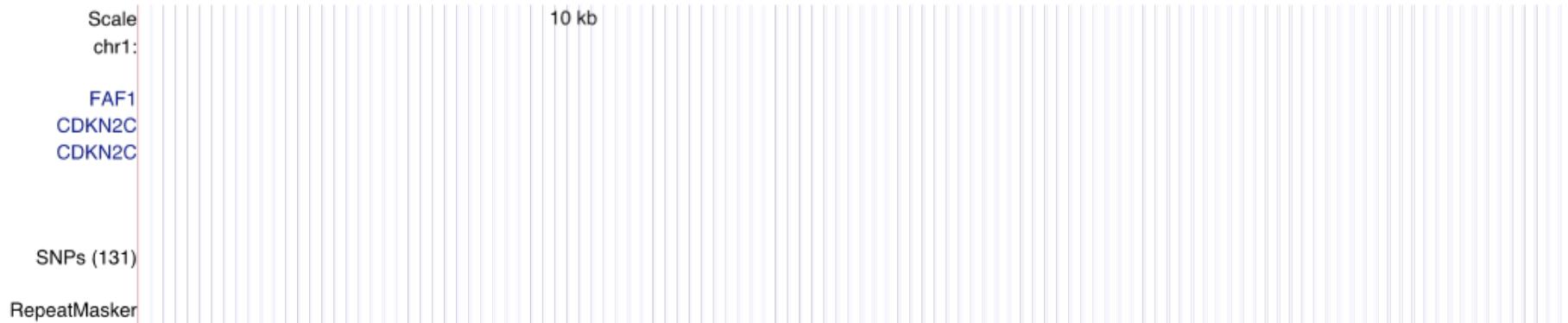
# What is a genome “feature”?

- 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”

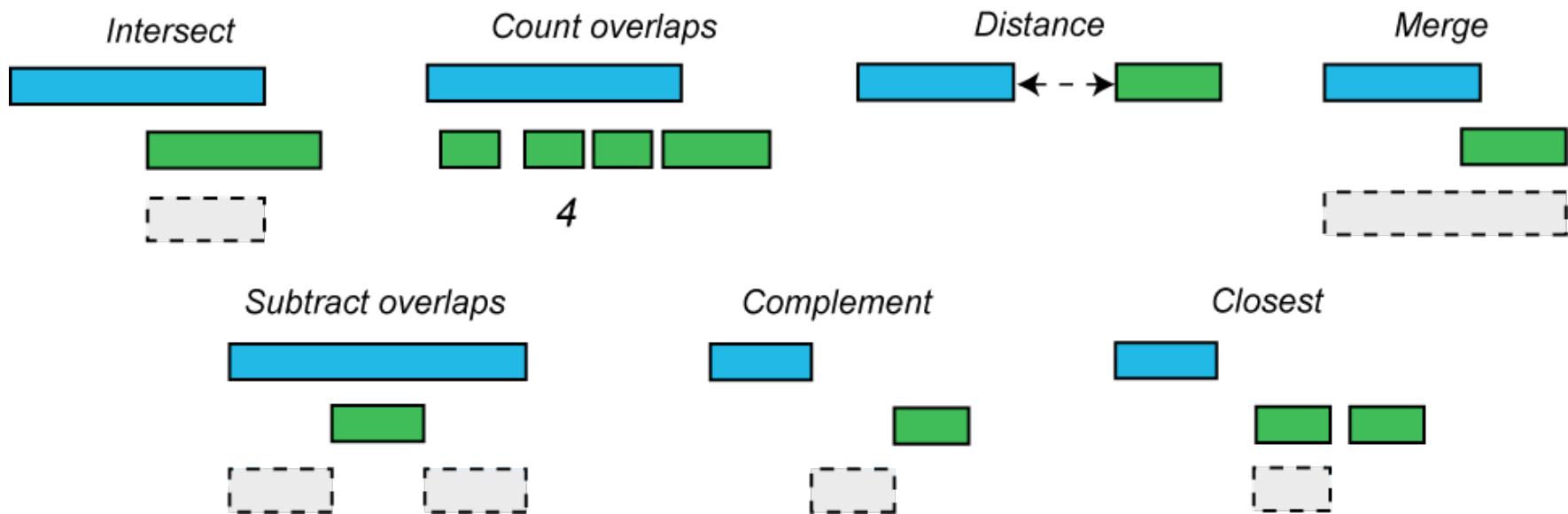


# Overlaps, closest, merge...



# 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
- ...

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

# 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

Human mRNAs

Rhesus  
Mouse  
Dog  
Elephant  
Opossum  
Chicken  
*X. tropicalis*  
Zebrafish

SNPs (131)

L1MB7  
MLT1C  
Alu**b**  
MER115  
L2a

T

TGTCTAGCTAAGG...  
TCCTTCCC  
CCTTCCTTCCCTC...  
CTTCCTCCCTCC...  
TTCTCTCTTCTC...  
TCCTTC  
TCTCTTCCTTCTT...  
CTCTTCTTCTC...  
TCTTCTTCTT  
CACCAATTACCTGT  
ATTCATGTT  
CCTCAGGGCTTCTT...  
GGCTGCTGCTGTG...  
GTGGCTGCTGCT  
GGCTGCTGCTGC

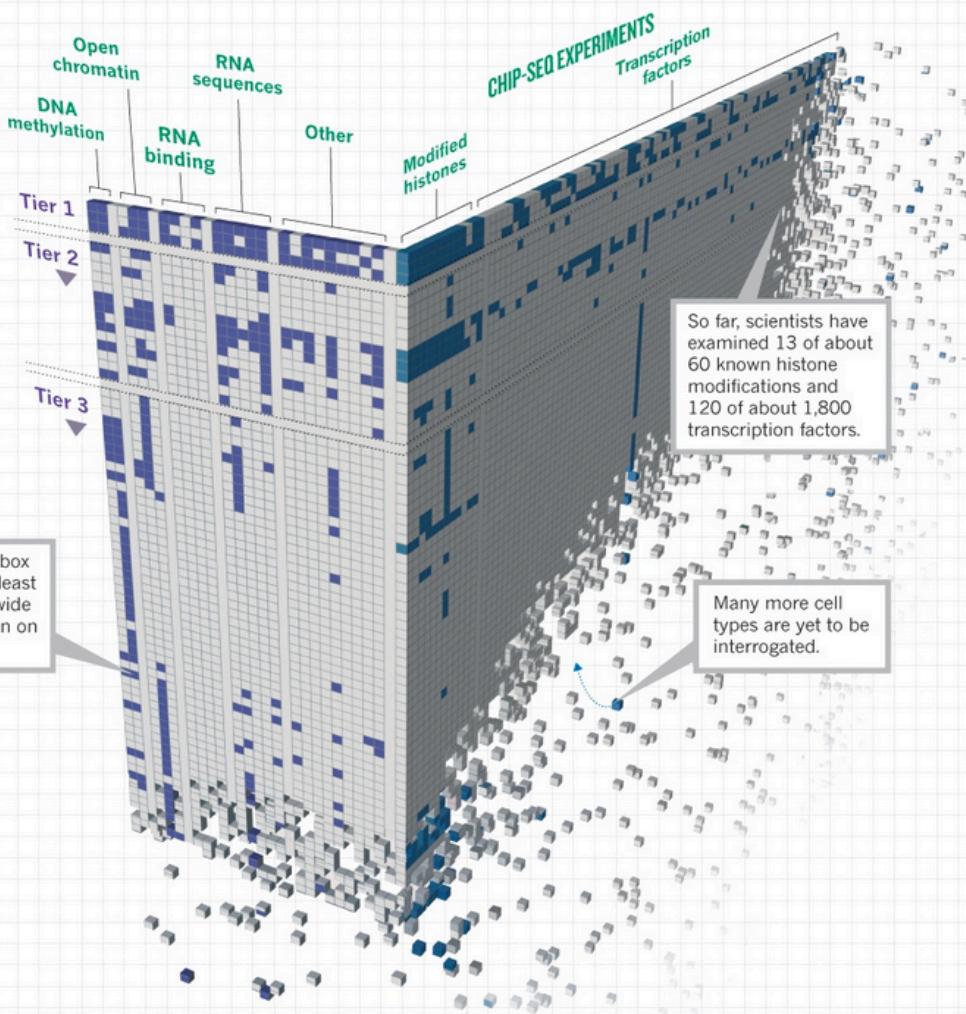
Why?

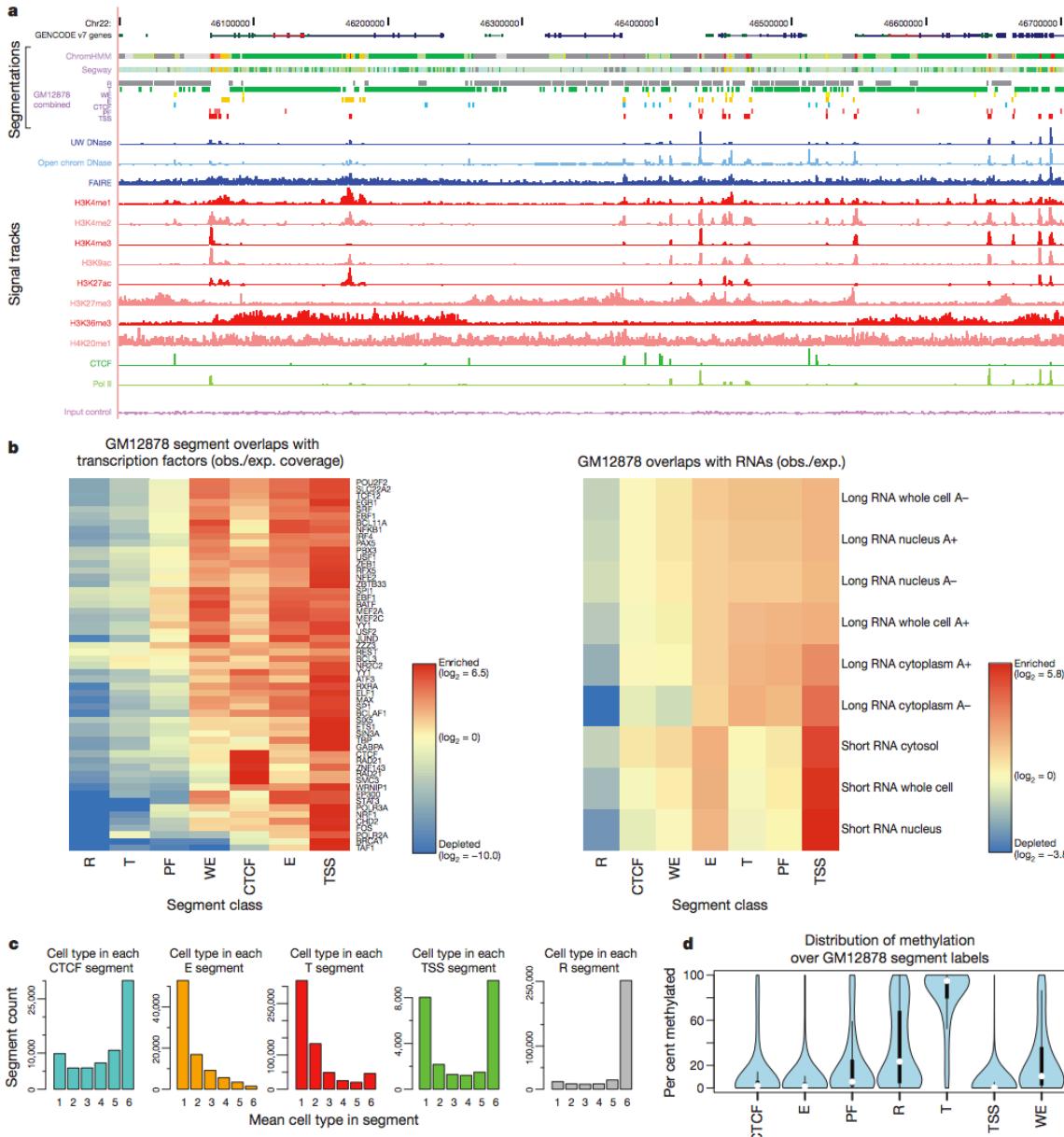
## MAKING A GENOME MANUAL

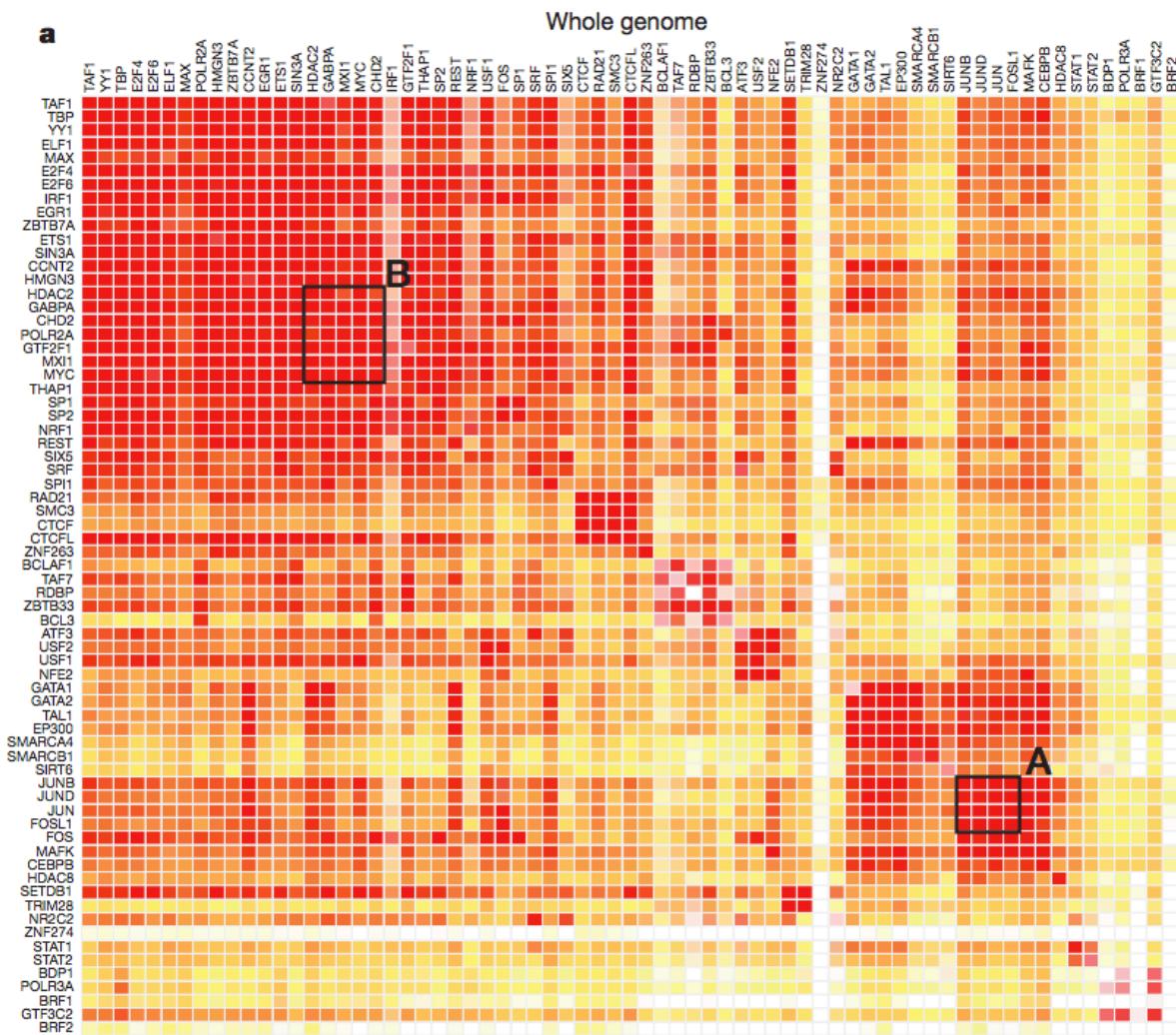
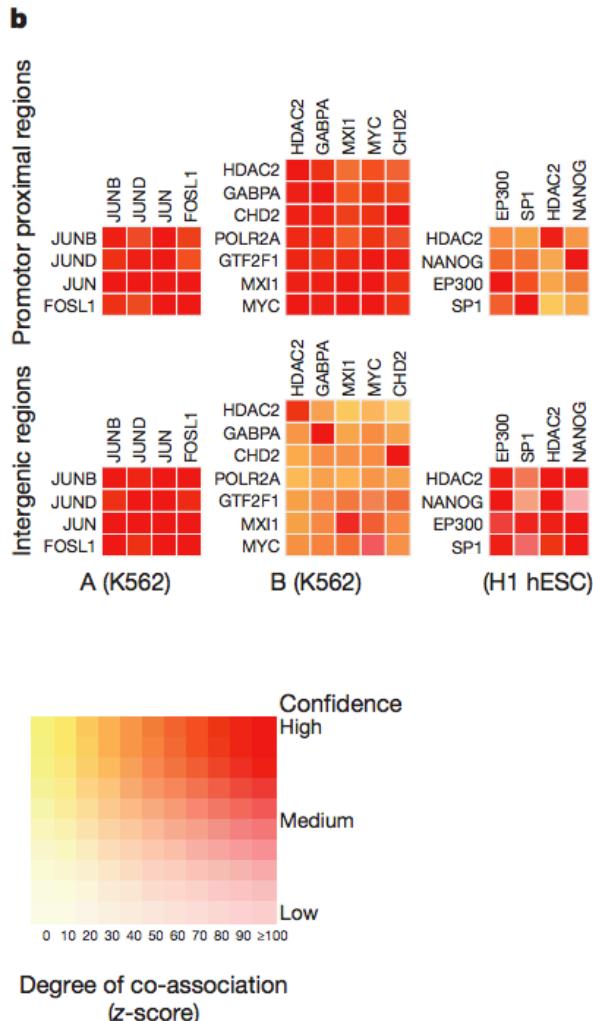
Scientists in the Encyclopedia of DNA Elements Consortium have applied 24 experiment types (across) to more than 150 cell lines (down) to assign functions to as many DNA regions as possible — but the project is still far from complete.

EXPERIMENTAL TARGETS
<b>DNA methylation:</b> regions layered with chemical methyl groups, which regulate gene expression.
<b>Open chromatin:</b> areas in which the DNA and proteins that make up chromatin are accessible to regulatory proteins.
<b>RNA binding:</b> positions where regulatory proteins attach to RNA.
<b>RNA sequences:</b> regions that are transcribed into RNA.
<b>ChIP-seq:</b> technique that reveals where proteins bind to DNA.
<b>Modified histones:</b> histone proteins, which package DNA into chromosomes, modified by chemical marks.
<b>Transcription factors:</b> proteins that bind to DNA and regulate transcription.

**CELL LINES**  
Tiers 1 and 2: widely used cell lines that were given priority.  
Tier 3: all other cell types.





**a****b**

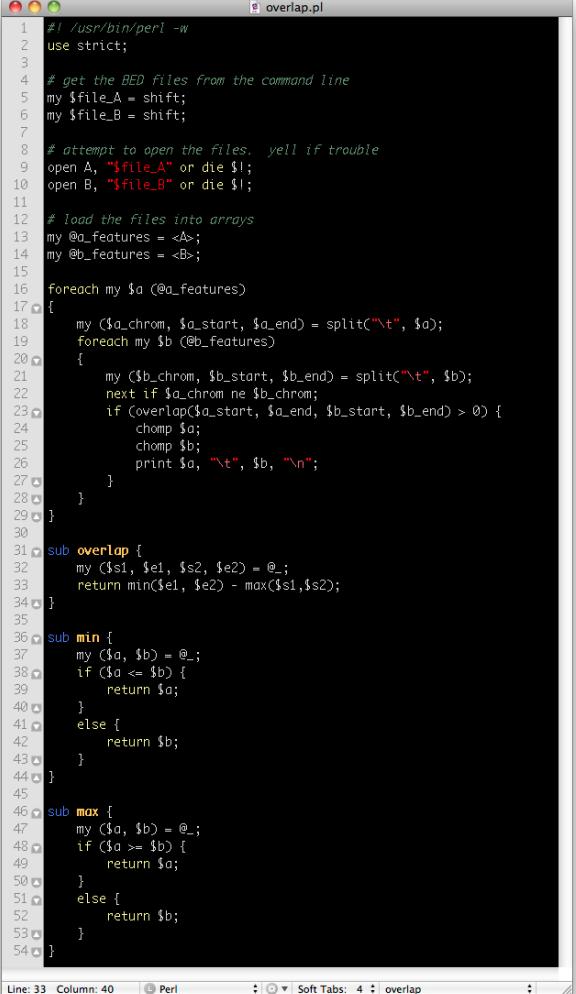
How?

# 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?*
    }
}
```



The screenshot shows a terminal window with the title bar "overlap.pl". The window displays a Perl script with line numbers from 1 to 54. The script starts with shebang "#! /usr/bin/perl -w" and uses strictures. It reads two BED files from command-line arguments (\$file\_A and \$file\_B). It loads the features into arrays (@a\_features and @b\_features) and then iterates through each feature in @a\_features. For each feature \$a, it splits its coordinates into chrom, start, and end. Then, it iterates through each feature in @b\_features. For each feature \$b, it splits its coordinates into chrom, start, and end. It checks if the overlap between \$a and \$b is greater than 0. If so, it prints the coordinates of \$a and \$b separated by a tab. The script includes three helper subroutines: overlap, min, and max.

```
1 #! /usr/bin/perl -w
2 use strict;
3
4 # get the BED files from the command line
5 my $file_A = shift;
6 my $file_B = shift;
7
8 # attempt to open the files, yell if trouble
9 open A, "$file_A" or die $!;
10 open B, "$file_B" or die $!;
11
12 # load the files into arrays
13 my @a_features = <A>;
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 $b_chrom;
23         if (Overlap($a_start, $a_end, $b_start, $b_end) > 0) {
24             chomp $a;
25             chomp $b;
26             print $a, "\t", $b, "\n";
27         }
28     }
29 }
30
31 sub overlap {
32     my ($s1, $e1, $s2, $e2) = @_;
33     return min($e1, $e2) - max($s1,$s2);
34 }
35
36 sub min {
37     my ($a, $b) = @_;
38     if ($a <= $b) {
39         return $a;
40     } else {
41         return $b;
42     }
43 }
44
45 sub max {
46     my ($a, $b) = @_;
47     if ($a >= $b) {
48         return $a;
49     } else {
50         return $b;
51     }
52 }
53 }
```

\* 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*

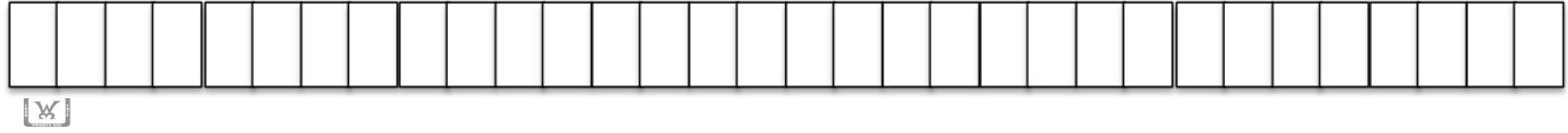


Human chrom 1: ca. 250 Mb

# Narrow the search space w/ “binning”

*the UCSC genome browser approach*

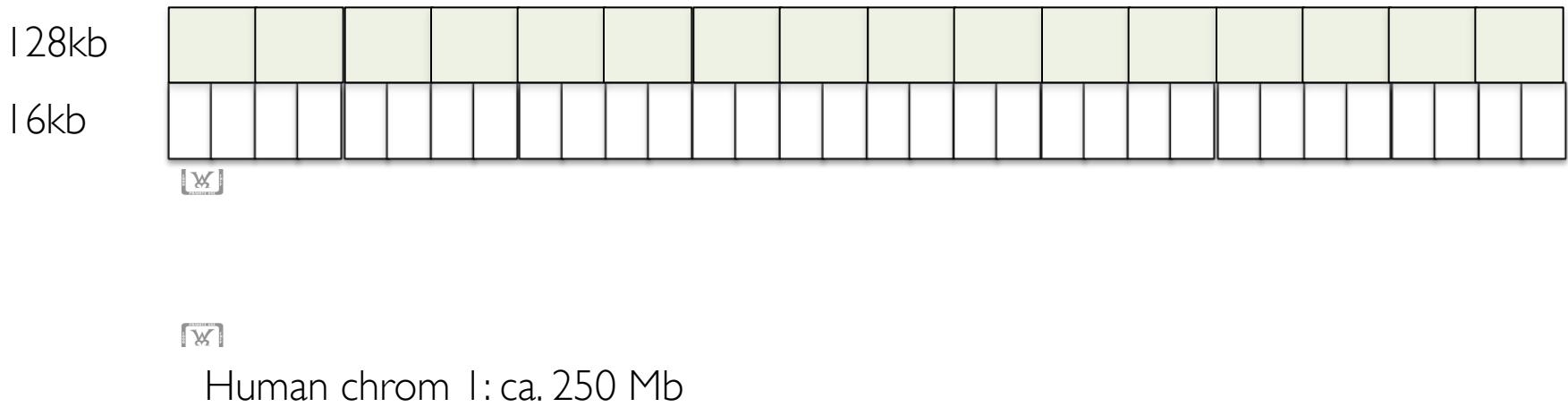
16kb



Human chrom 1: ca. 250 Mb

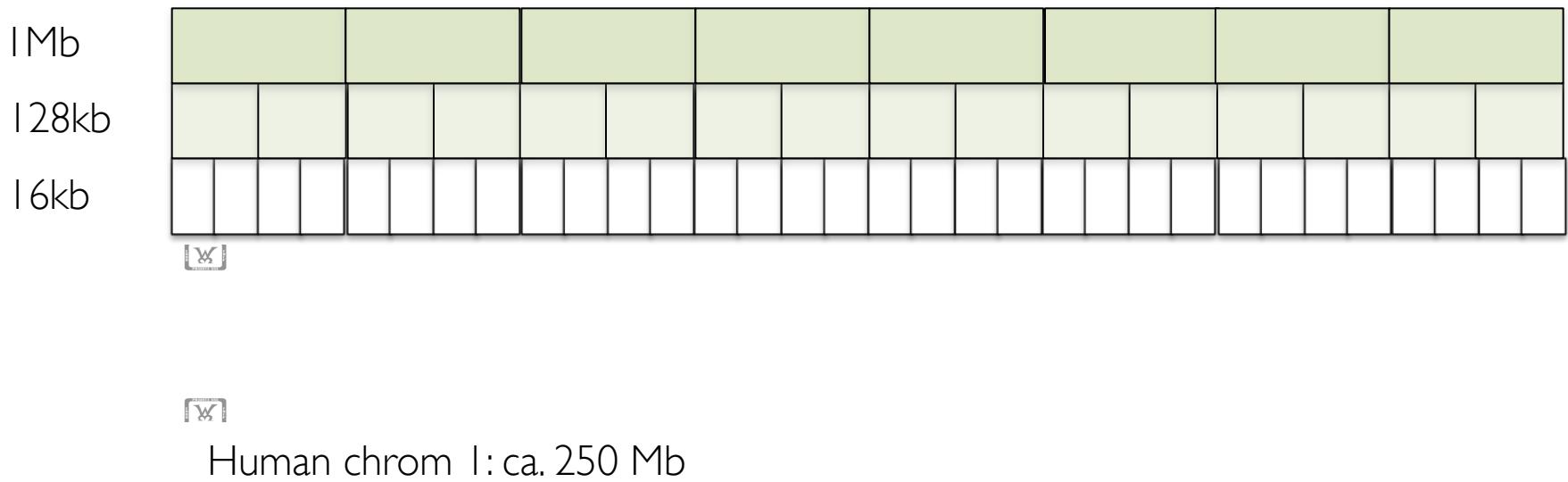
# Narrow the search space w/ “binning”

*the UCSC genome browser approach*



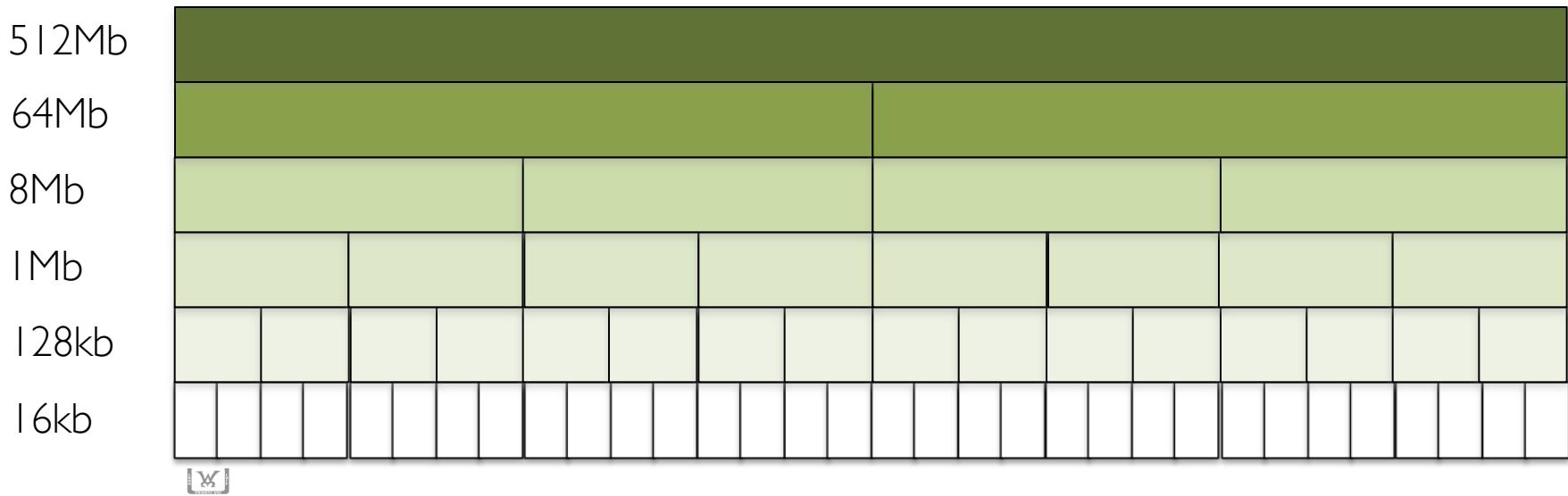
# Narrow the search space w/ “binning”

*the UCSC genome browser approach*



# Narrow the search space w/ “binning”

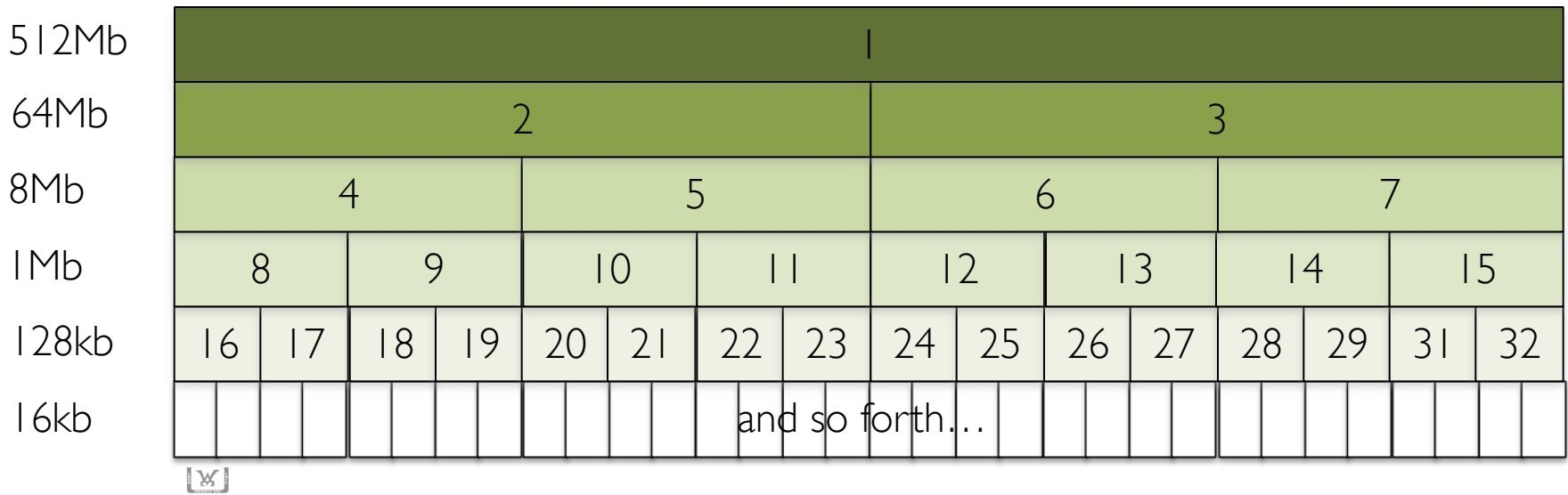
*the UCSC genome browser approach*



Human chrom 1: ca. 250 Mb

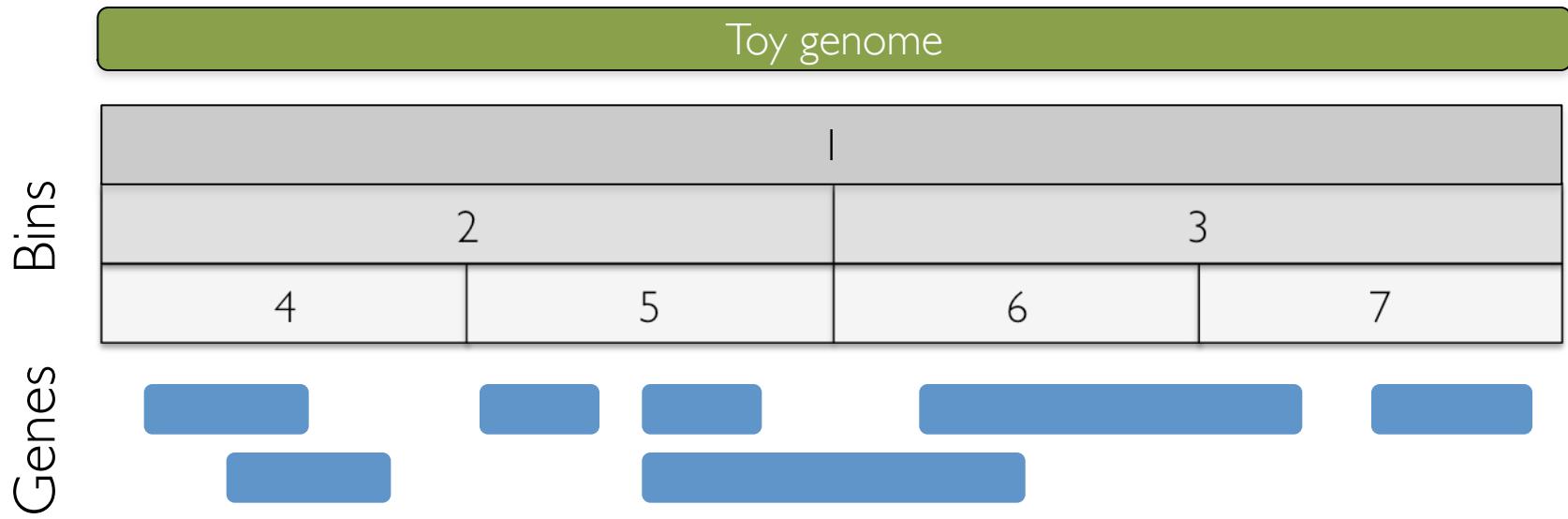
# Narrow the search space w/ “binning”

*the UCSC genome browser approach*

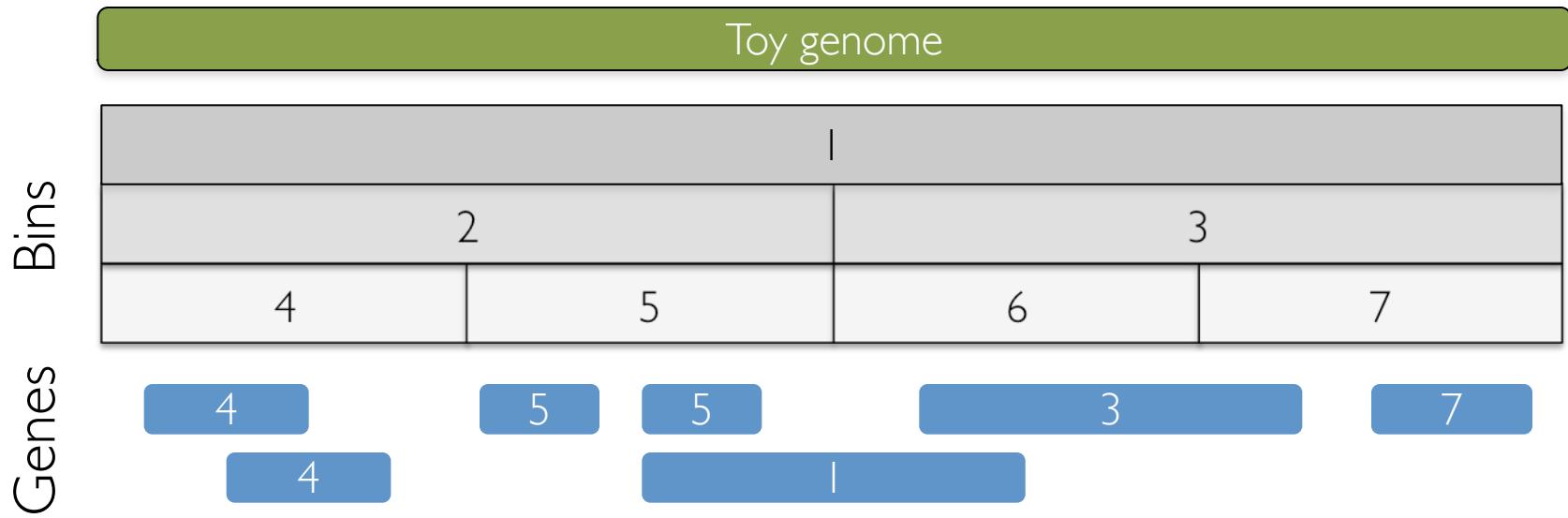


Human chrom 1: ca. 250 Mb

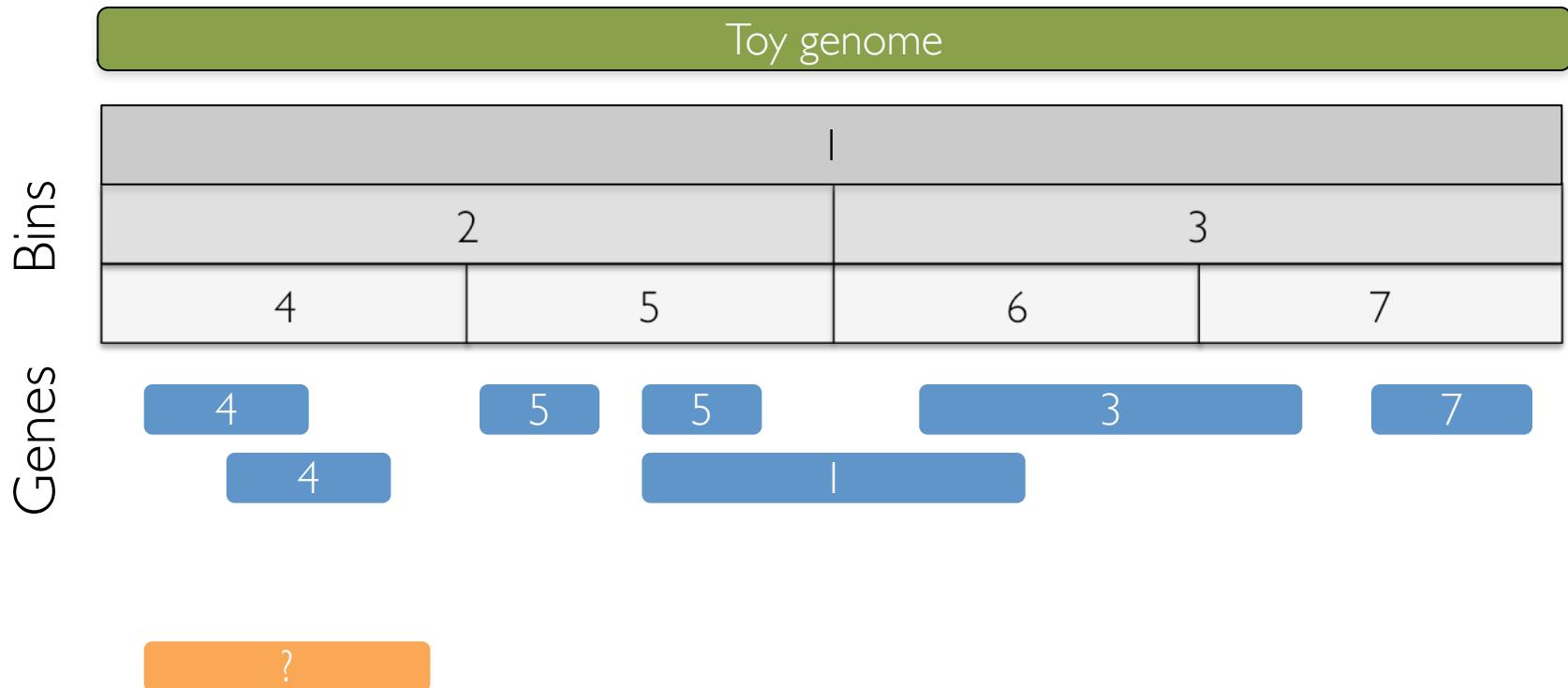
# A toy example



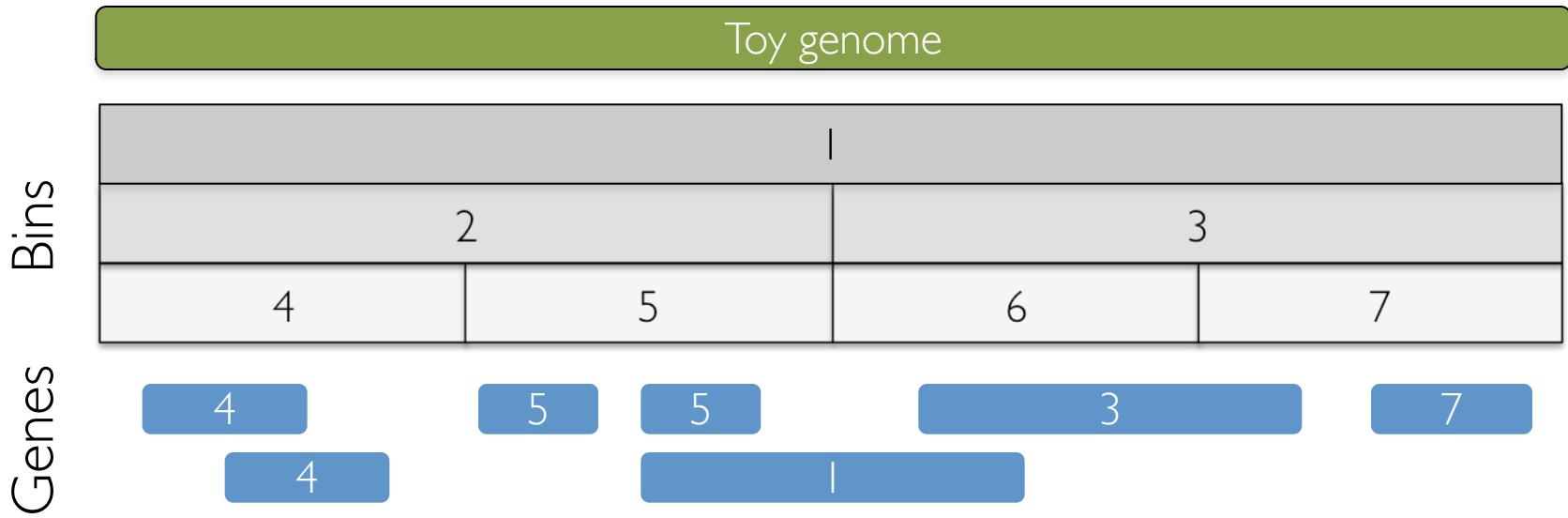
# A toy example



# A toy example

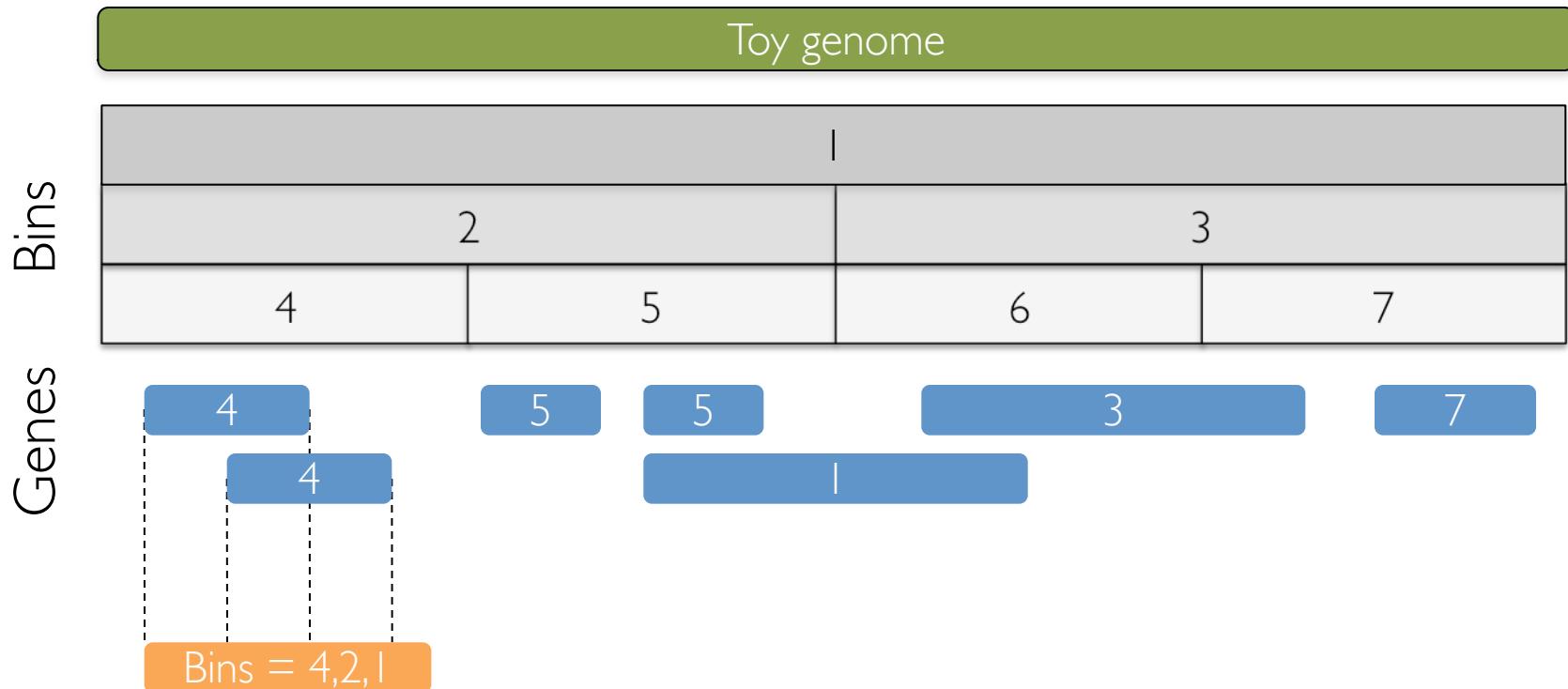


# A toy example

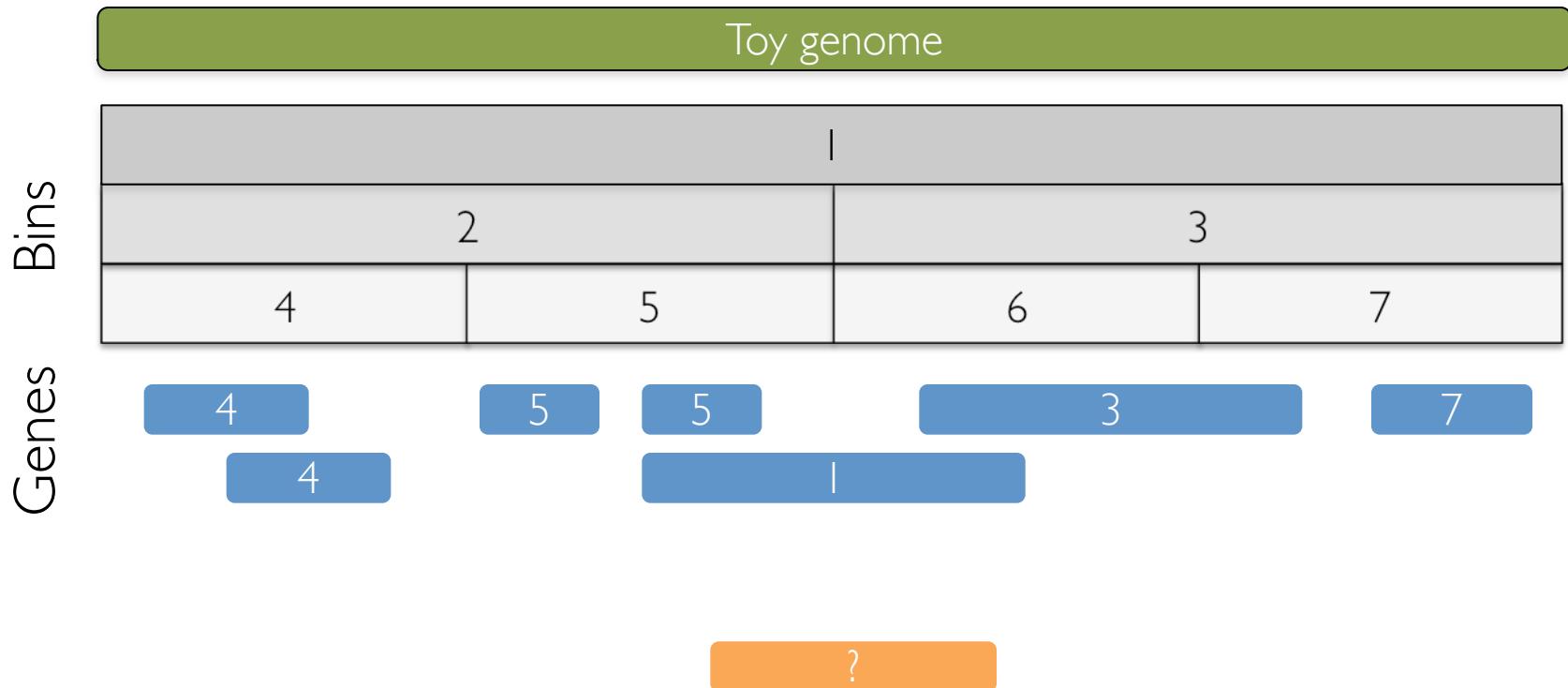


Bins = 4,2,|

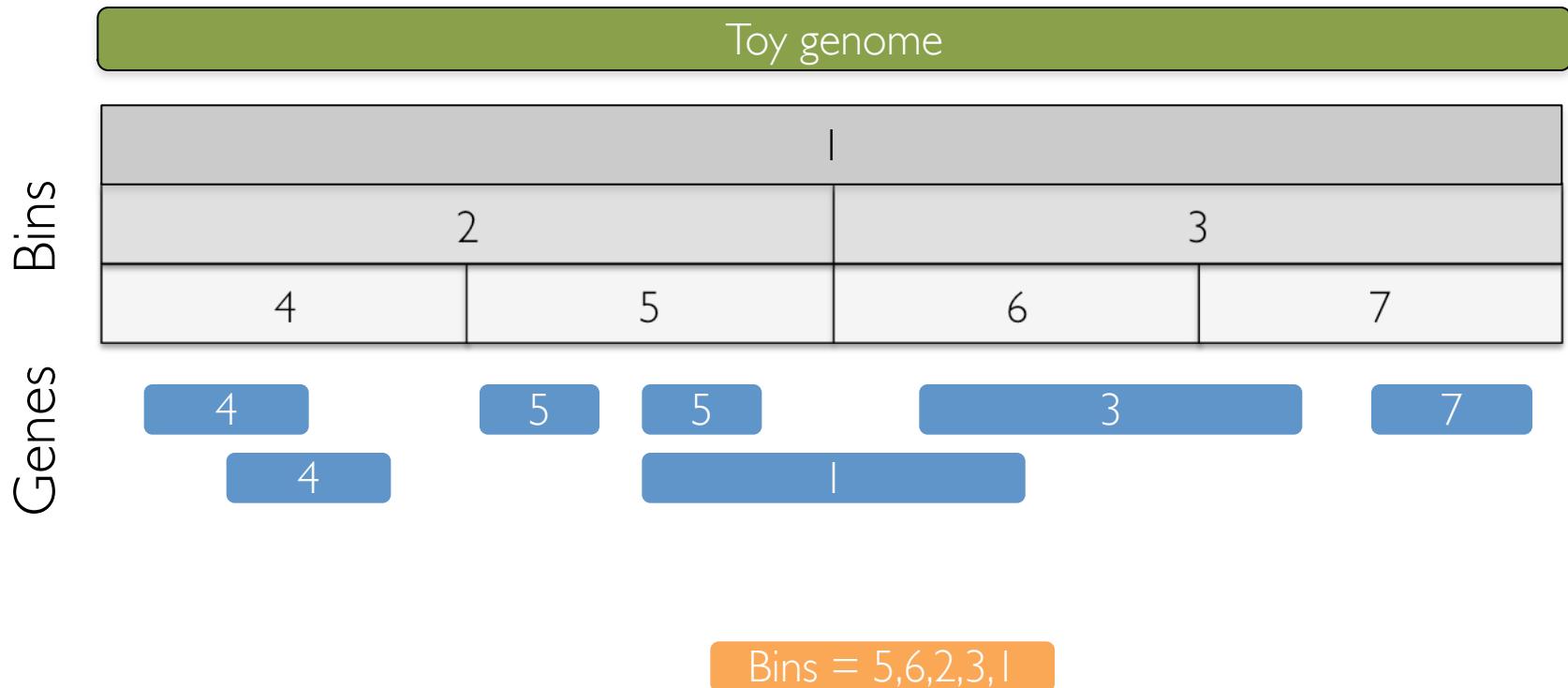
# A toy example



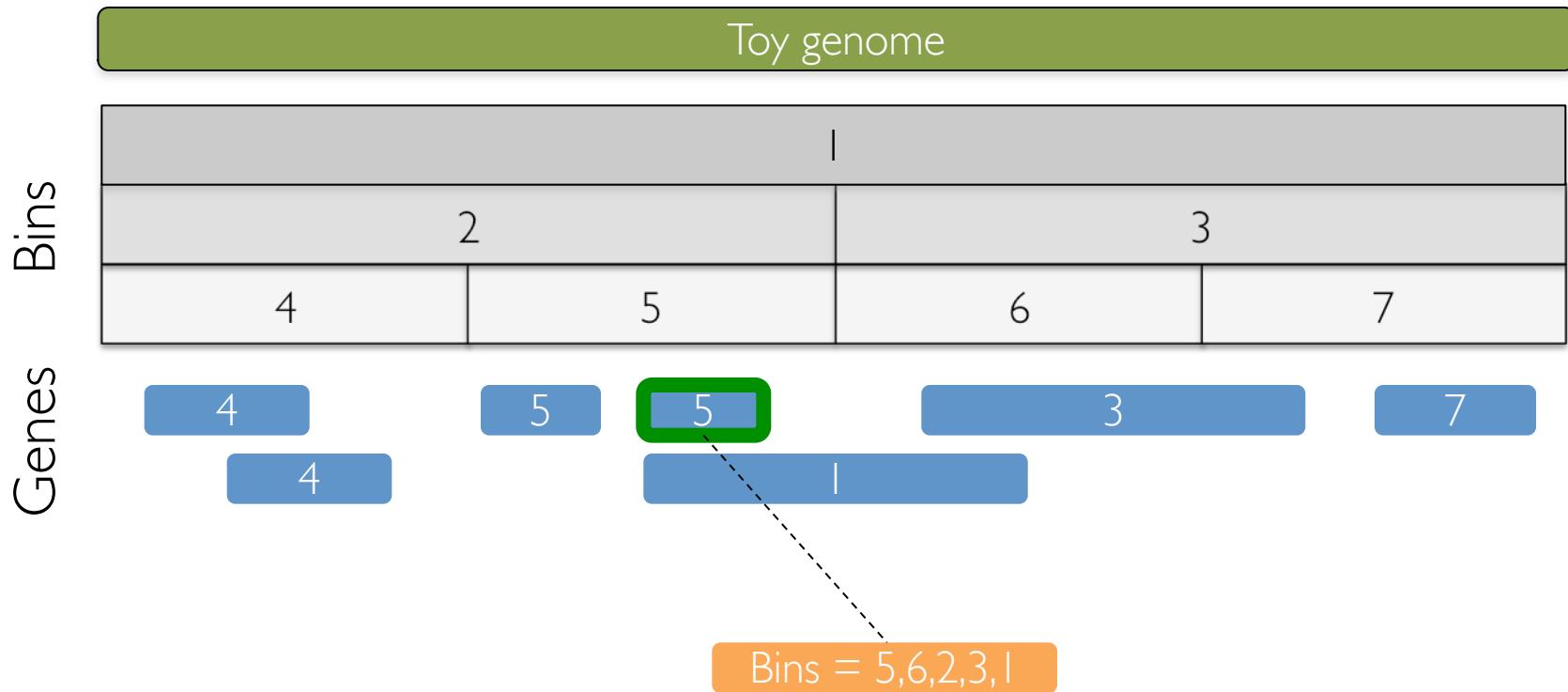
# A toy example



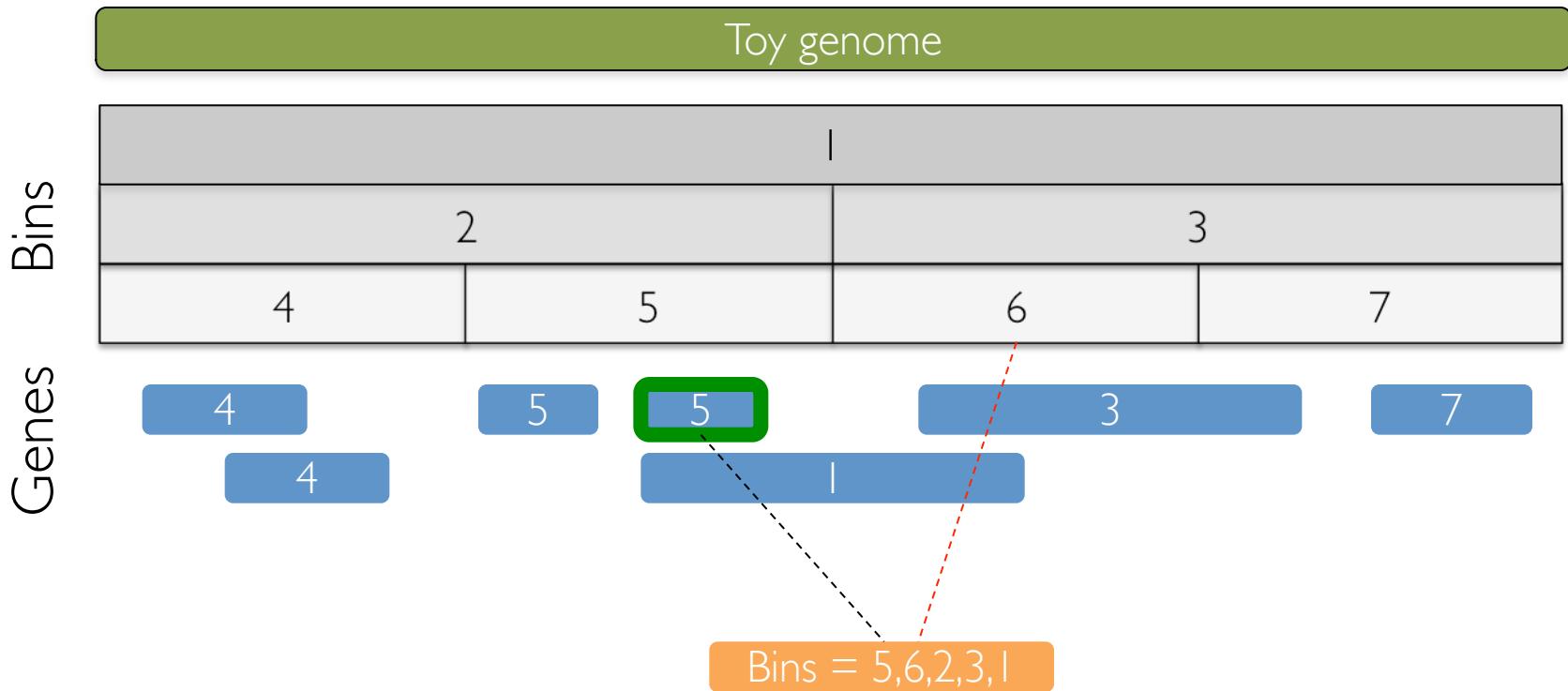
# A toy example



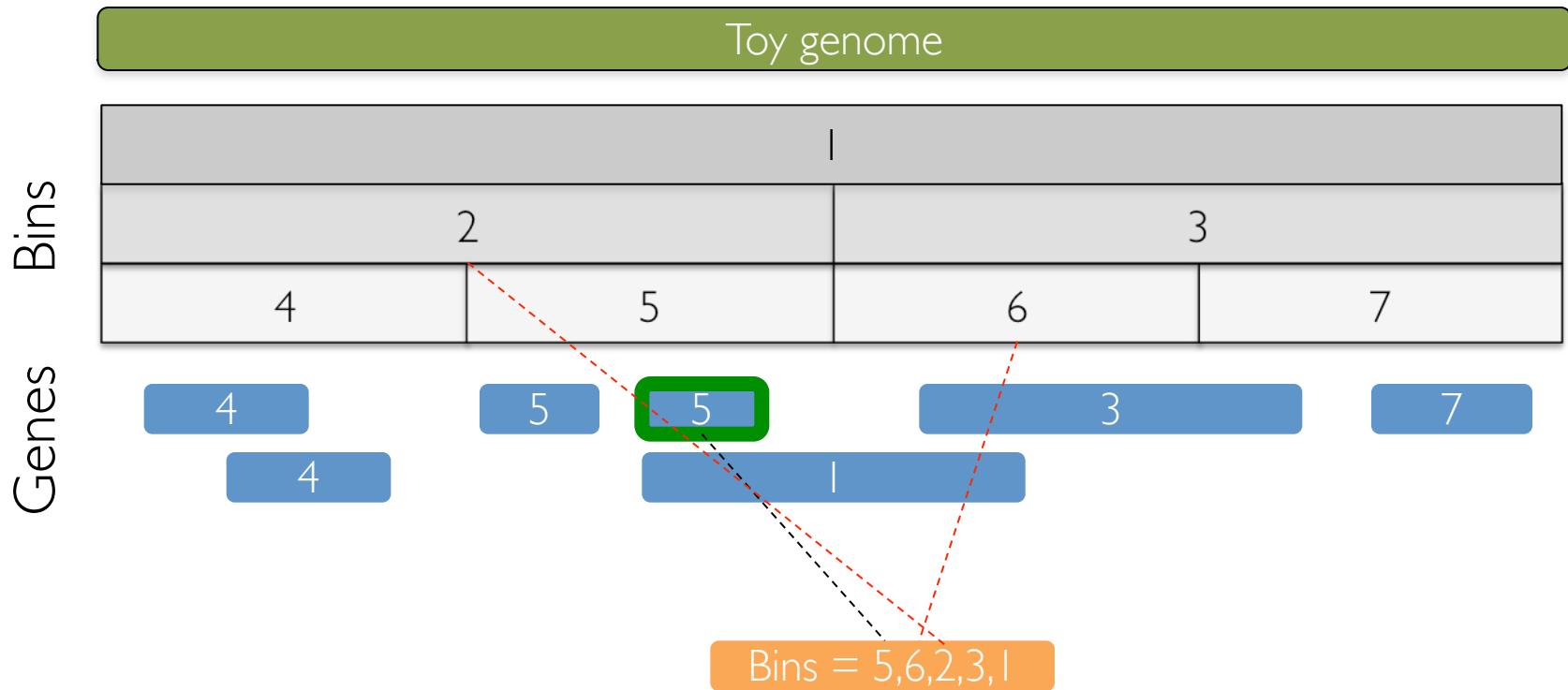
# A toy example



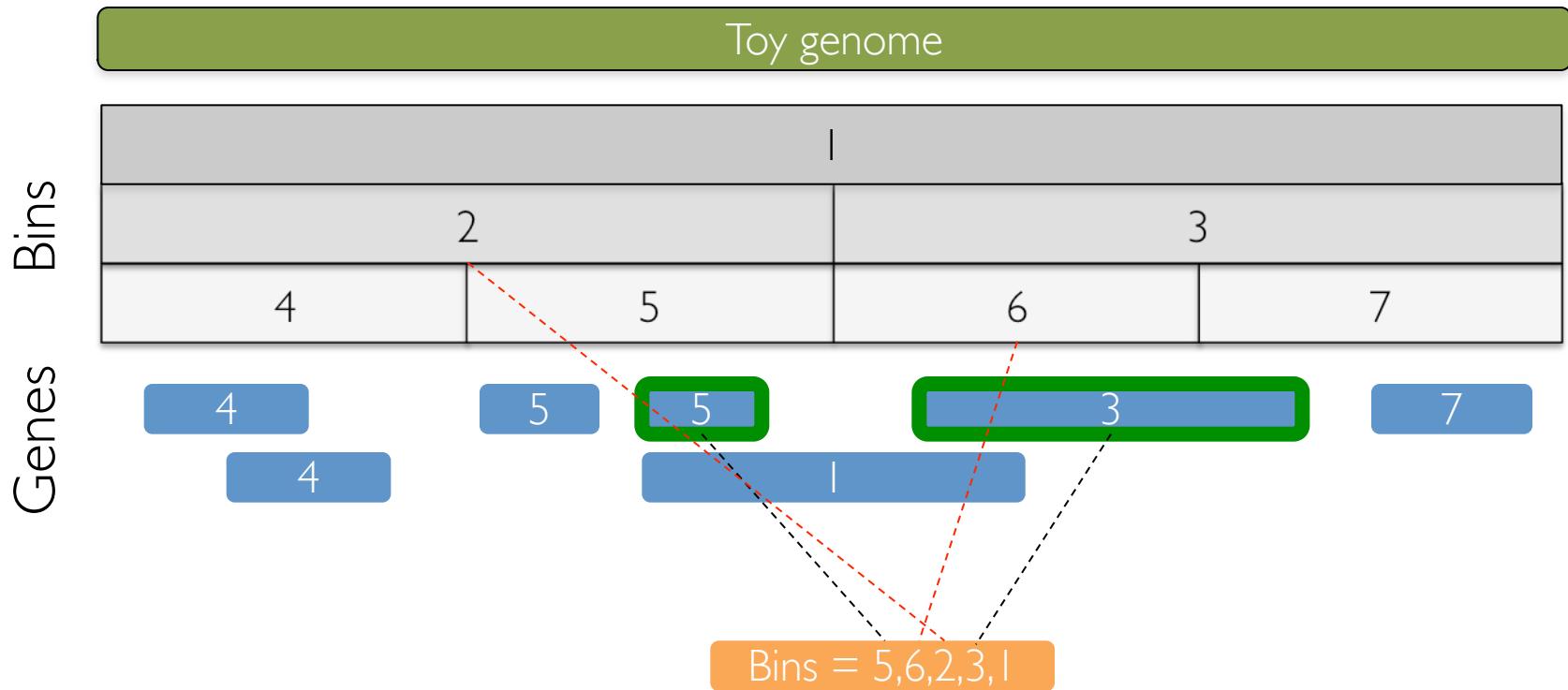
# A toy example



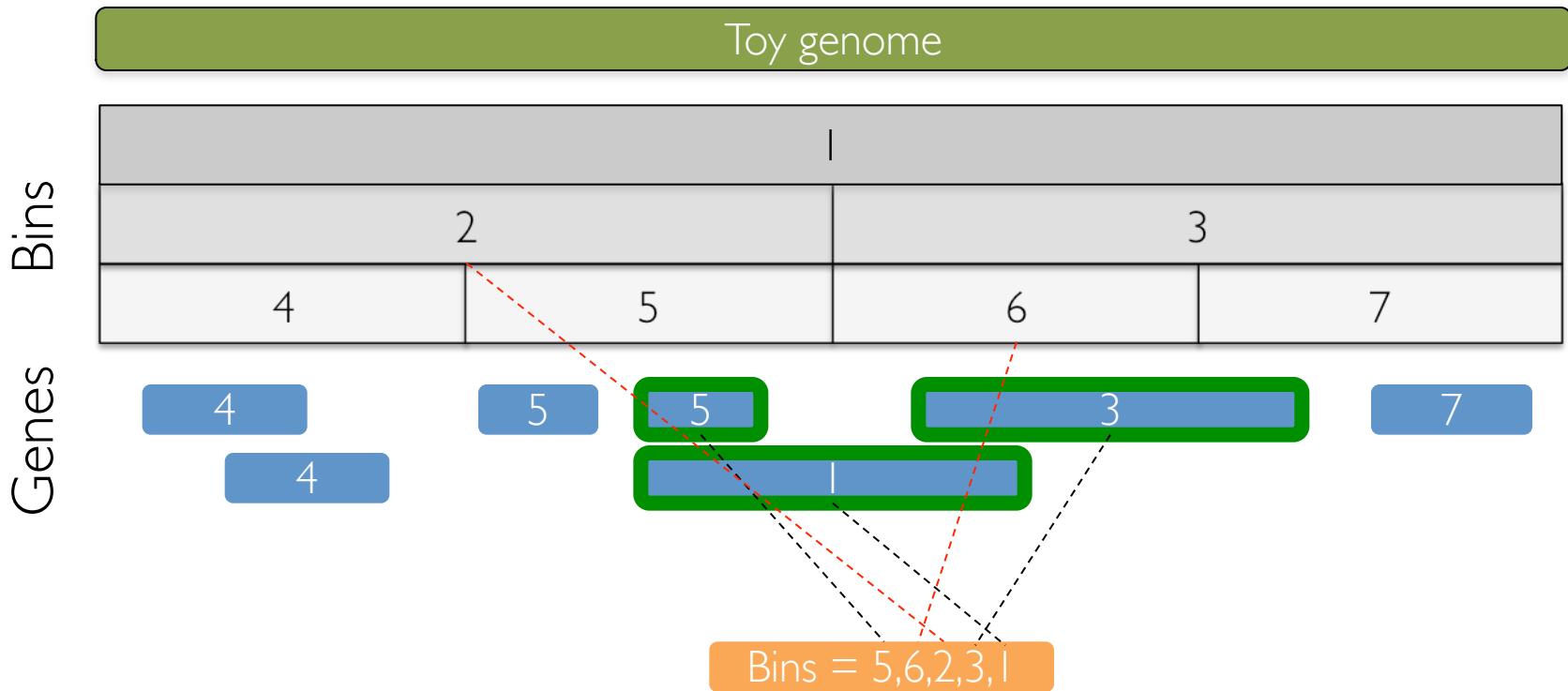
# A toy example



# A toy example



# A toy example



# Available tools for GA

UCSC Genome Browser

# UCSC Genome Browser

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

The screenshot shows the UCSC Table Browser interface running in a Mac OS X window. The title bar reads "Table Browser". The address bar shows the URL "http://genome.ucsc.edu/cgi-bin/hgTables?command=start". The menu bar includes "Google Calendar", "VPN", "Genomics", "Personal", "http://www.broadin...", "Grants", "ITC - Cluster", "Which application is ...", "Data analysis", "Genetics", and "Table Browser". The main content area has a blue header bar with links for "Home", "Genomes", "Genome Browser", "Blat", "Tables", "Gene Sorter", "PCR", "Session", "FAQ", and "Help". Below this is a section titled "Table Browser" with descriptive text about the program's functions. It contains several dropdown menus and input fields: "clade: Mammal", "genome: Human", "assembly: Feb. 2009 (GRCh37/hg19)", "group: Comparative Genomics", "track: Conservation", "table: Primate Cons (phyloP4GwayPrimates)", "region: genome (chr21:33021623-33051544)", "filter: create", "subtrack merge: create", "intersection: create", "correlation: create", "output format: bed format", "Send output to: Galaxy, GREAT", "output file: (leave blank to keep output in browser)", "file type returned: plain text, gzip compressed". There are also buttons for "get output" and "summary/statistics". A note at the bottom says "Note: to return more than 100,000 lines, change the filter setting (above). The entire data set may be available for download as a very large file that contains the original data values (not compressed into the wiggle format) -- see the Downloads page." Below this is a link "To reset all user cart settings (including custom tracks), click here.". At the bottom of the main content area is a section titled "Using the Table Browser" with a list of controls and their descriptions.

## Using the Table Browser

This section provides brief line-by-line descriptions of the Table Browser controls. For more information on using this program, see the [Table Browser User's Guide](#).

- **clade:** Specifies which clade the organism is in.
- **genome:** Specifies which organism data to use.
- **assembly:** Specifies which version of the organism's genome sequence to use.
- **group:** Selects the type of tracks to be displayed in the *track* list. The options correspond to the track groupings shown in the Genome Browser. Select 'All Tracks' for an alphabetical list of all available tracks in all groups. Select 'All Tables' to see all tables including those not associated with a track.
- **database:** (with "All Tables" group option) Determines which database should be used for options in table menu.
- **track:** Selects the annotation track data to work with. This list displays all tracks belonging to the group specified in the *group* list.
- **table:** Selects the SQL table data to use. This list shows all tables associated with the track specified in the *track* list.
- **describe table schema:** Displays schema information for the tables associated with the selected track.
- **region:** Restricts the query to a particular chromosome or region. Select *genome* to apply the query to the entire genome or *ENCODE* to examine only the ENCODE regions. To limit the query to a specific position, type a

# Galaxy

<http://main.g2.bx.psu.edu/root>

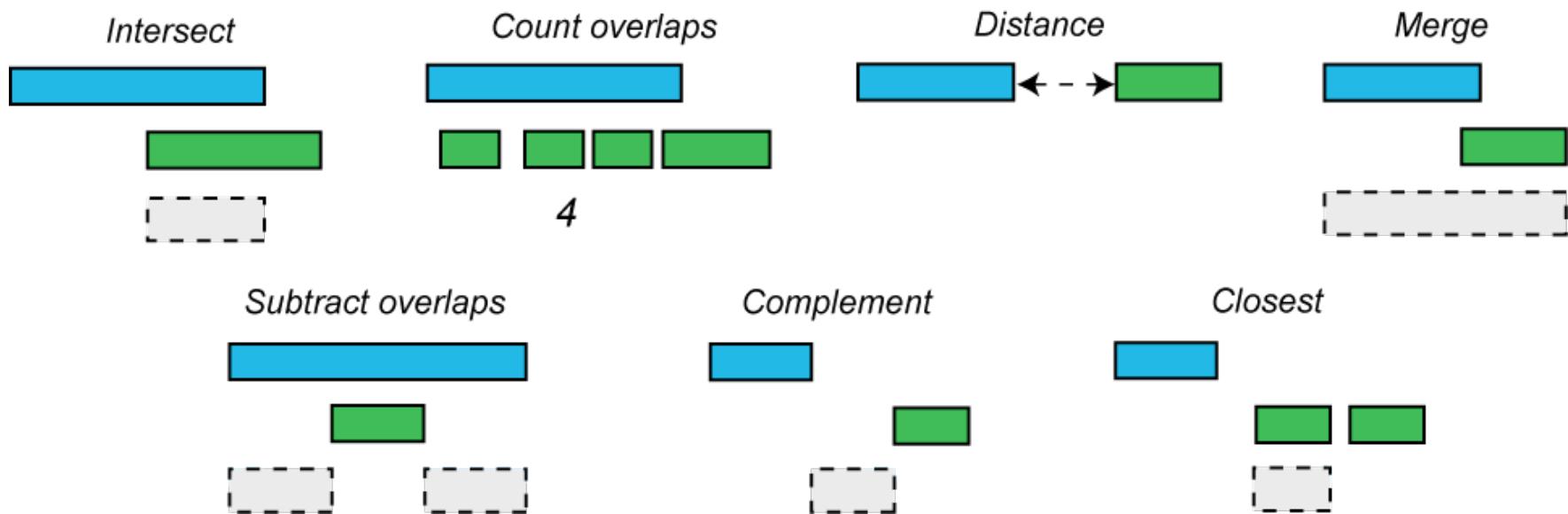
The screenshot shows the Galaxy web interface running in a Mac OS X browser window. The title bar reads "Galaxy". The address bar shows the URL "http://main.g2.bx.psu.edu/root". The top menu bar includes "Google Calendar", "VPN", "Genomics", "Personal", "ITC - Cluster", "Which application is ...", "Data analysis", and "Genetics". The main content area has a dark header with "Galaxy" and a "Tools" dropdown. The left sidebar contains a "Get Data" section with links to various servers like UCSC Main, BX main, BioMart, GrameneMart, Flymine, modENCODE fly, modENCODE modMine, Ratmine, modENCODE worm, Wormbase, EuPathDB, EncodeDB, and EnGRAPH. Below this are sections for "Send Data", "ENCODE Tools", "Lift-Over", "Text Manipulation", "Convert Formats", "FASTA manipulation", "Filter and Sort", "Join, Subtract and Group", "Extract Features", "Fetch Sequences", "Fetch Alignments", "Get Genomic Scores", "Operate on Genomic Intervals", "Statistics", "Graph/Display Data", "Regional Variation", "Multiple regression", "Multivariate Analysis", "Evolution", "Metagenomic analyses", "Human Genome Variation", and "EMBOSS". A "History" panel on the right says "Your history is empty. Click 'Get Data' on the left pane to start". The central area features a "Here is what's happening..." box with a "Galaxy 2011 Community Conference" banner for May 25-26 in Lunteren, The Netherlands, with a "Register now!" button. Below it is a "Live Quickies" section with six cards: "Mapping against custom genome" (Galactic quickie #10), "Illumina mapping: Single Ends" (Galactic quickie #11), "Illumina mapping: Paired Ends" (Galactic quickie #12), "Basic fastQ manipulation" (Galactic quickie #13), "Advanced fastQ manipulation" (Galactic quickie #14), and "454 Mapping: Single End" (Galactic quickie #15). At the bottom, it says "The Galaxy team is a part of BX at Penn State. This project is supported in part by NSF, NHGRI, The Huck Institutes of the Life Sciences, and The Institute for CyberScience at Penn State. Galaxy build: \$Rev 4919:95d65755ac69\$". The footer shows the URL "http://main.g2.bx.psu.edu/root/tool\_menu#".

# BEDTools

<http://code.google.com/p/bedtools/>

The screenshot shows a Mac OS X browser window displaying the bedtools project page on Google Code. The title bar reads "bedtools - Project Hosting on Google Code". The main content area shows the project's logo, a colorful DNA helix icon, and the text "bedtools: a flexible suite of utilities for comparing genomic features". A navigation menu at the top includes links for Project Home, Downloads, Wiki, Issues, Source, and Administer. The "Project Home" tab is selected. On the left, there's a sidebar with sections for Project Information (Starred project, Activity, Code license), Labels (bioinformatics, genomics, bed, sam, bam, overlap, features, sequencing, intersect, coverage, gff, vcf, bedgraph), Feeds (Project feeds), Owners (aaronquinlan), Committers (0 committers), Contributors (0 contributors), and People details. The main content area has tabs for Summary, Updates, and People. The "Summary" tab is active. It contains sections for "Project Information" (Source Repository, Citation, Latest news (Version 2.11.2, 31-January-2011), BEDTools Summary, Brief example, Table of supported utilities, Documentation, Notes regarding usage, Installation, Citation, Contact), "Source Repository" (explains GitHub integration), "Citation" (with a note to cite the article by Quinlan and Hall, 2010), and "Latest news (Version 2.11.2, 31-January-2011)" (listing 11 improvements). To the right, there are "Featured" links for Downloads (BEDTools-User-Manual.pdf, BEDTools v2.11.2.tar.gz) and Wiki pages (Contributors, History, Usage, UsageAdvanced). A "Links" sidebar lists External links (BEDTools manuscript, BED format, GFF/GTF format, VCF format (v4.0), BamTools, SAM/BAM format, FASTX Toolkit) and Groups (BEDTools Discussion).

# 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

# Step I: Get annotations

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

# SNPs from dbSNP via UCSC

The figure illustrates the process of retrieving SNPs from dbSNP via the UCSC Genome Browser Table Browser.

**Table Browser (Left):**

- Assembly: Feb. 2009 (GRCh37/hg19)
- Group: Variation and Repeats
- Region: chr1
- Identifiers: snp131
- Output format: selected fields from primary and related tables
- File type returned: plain text

**Select Fields from hg19.snp131 (Middle):**

Fields selected:

- bin
- chrom
- chromStart
- chromEnd
- name
- score
- strand
- observed
- func

**Terminal (Bottom Left):**

```
qbm:BI0-5080 arq5x$ wc -l snps.bed
2064872 snps.bed
qbm:BI0-5080 arq5x$ head snps.bed
#chrom chromStart chromEnd name score strand observed func
chr1 10433 10433 rs56289060 0 + -/C near-gene-5
chr1 10491 10492 rs55998931 0 + C/T near-gene-5
chr1 10518 10519 rs62635508 0 + C/G near-gene-5
chr1 10582 10583 rs58108140 0 + A/G near-gene-5
chr1 10827 10828 rs10218492 0 + A/G near-gene-5
chr1 10903 10904 rs10218493 0 + A/G near-gene-5
chr1 10926 10927 rs10218527 0 + A/G near-gene-5
chr1 10937 10938 rs28853987 0 + A/G near-gene-5
chr1 11001 11002 rs79537094 0 + A/C near-gene-5
```

**Select Fields from hg19.snp131 (Right):**

Data table showing selected fields for SNPs in hg19.snp131. The columns include chrom, chromStart, chromEnd, name, score, strand, observed, and func.

**Text Output (Bottom Right):**

“snps.bed”  
N = 2,064,872

# CpG islands via UCSC

Table Browser

http://genome.ucsc.edu/cgi-bin/hgTables?hgSID=186205655&clade=mammal

Home Genomes Genome Browser Blat Tables Gene Sorter PCR Session FAQ Help

**Table Browser**

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](#). Refer to the [Credits](#) page for the list of contributors and usage restrictions associated with these data.

clade: Mammal genome: Human assembly: Feb. 2009 (GRCh37/hg19)

group: Regulation track: CpG Islands add custom tracks

table: cgislandExt describe table schema

region:  genome  position chr1:1-249250621 lookup define regions

identifiers (names/accessions): paste list upload list

filter: create

intersection: create

correlation: create

output format: selected fields from primary and related tables Send output to  Galaxy  GREAT

output file: cgislandExt (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](#).

**Using the Table Browser**

This section provides brief line-by-line descriptions of the Table Browser controls. For more information on using this program, see the [Table Browser User's Guide](#).

Done

```
Terminal — bash — 87x26
qbm:BI0-5080 arq5x$ wc -l cgislandExt
2463 cgislandExt
qbm:BI0-5080 arq5x$ head cgislandExt
#chrom chromStart chromEnd length cgislandExtNum
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
qbm:BI0-5080 arq5x$
```

“cgislandExt”  
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 2<sup>nd</sup> col. is g.t.e. 30)
- `awk '{print $3,$2,$7,$1}' foo.txt > foo.reordered.txt`

# The awesome power of **awk**

A Bioinformatician's UNIX Toolbox

http://lh3lh3.users.sourceforge.net/biounix.shtml

heng li

Google Calendar VPN Genomics Personal http://www.broadin... Grants ITC - Cluster Which application is ... Data analysis Genetics

A Bioinformatician's UNIX Toolbox

## Awk

Awk is a programming language that is specifically designed for quickly manipulating space delimited data. Although you can achieve all its functionality with Perl, awk is simpler in many practical cases. You can find a lot of online tutorials, but here I will only show a few examples which cover most of my daily uses of awk.

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

# Use awk to fix cpg.bed

```
Terminal — bash — 87x26
qbm:BI0-5080 arq5x$ wc -l cpg.bed
    2463 cpg.bed
qbm: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
qbm: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
```

```
Terminal — bash — 105x26
qbm:BI0-5080 arq5x$ head cpg.bed
#chrom  chromStart  chromEnd      length  cpgNum
chr1    28735     29810     1075     116
chr1    135124    135563    cpg_2      +
chr1    327790    328229    cpg_3      +
chr1    437151    438164    cpg_4      +
chr1    449273    450544    cpg_5      +
chr1    533219    534114    cpg_6      +
chr1    544738    546649    cpg_7      +
chr1    713984    714547    cpg_8      +
chr1    762416    763445    cpg_9      +
chr1    762416    763445    cpg_10     +
qbm:BI0-5080 arq5x$
```

# Genes via UCSC

The screenshot shows the UCSC Table Browser interface. At the top, there are dropdown menus for 'clade' (Mammal), 'genome' (Human), and 'assembly' (Feb. 2009 GRCh37/hg19). Below these are fields for 'group' (Genes and Gene Prediction Tracks) and 'track' (RefSeq Genes). The 'region' section includes a radio button for 'position' (chr1:1-249250621) and a 'lookup' button. The 'identifiers' section has buttons for 'paste list' and 'upload list'. A 'filter' section contains a 'create' button. Below these are buttons for 'intersection' and 'correlation', both with 'create' options. The 'output format' is set to 'BED – browser extensible data'. There are checkboxes for 'Send output to Galaxy' and 'GREAT'. The 'output file' field contains 'genes.bed'. The 'file type returned' section has radio buttons for 'plain text' (selected) and 'gzip compressed'. At the bottom, there are 'get output' and 'summary/statistics' buttons, and a link to reset user cart settings.

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

**Using the Table Browser**

This section provides brief line-by-line descriptions of the Table Browser controls. For more information on using this program, see the [Table Browser User's Guide](#).

The terminal window shows the command 'wc -l genes.bed' being run, resulting in the output '3665 genes.bed'. Then, the command 'head genes.bed' is run, displaying the first few lines of the BED file. The file contains genomic coordinates and other data for 3665 genes. The terminal window title is 'Terminal — bash — 192x20'.

```
qbm:BI0-5080 arq5x$ wc -l genes.bed
3665 genes.bed
qbm:BI0-5080 arq5x$ head genes.bed
chr1 14361 29370 NR_024540 0 - 29370 29370 0 11 468,69,152,159,198,136,137,147,99,154,50, 0,608,1434,2245,2496,2871,3244,3553,3906,10376,14959,
chr1 34610 36081 NR_026818 0 - 36081 36081 0 3 564,205,361, 0,666,1110,
chr1 34610 36081 NR_026820 0 - 36081 36081 0 3 564,205,361, 0,666,1110,
chr1 69090 70008 NM_001005484 0 + 69090 70008 0 1 918, 0,
chr1 323891 328581 NR_028327 0 + 328581 328581 0 4 169,58,2500,15460,396,547,3144,
chr1 323891 328581 NR_028322 0 + 328581 328581 0 3 169,58,4143, 0,396,547,
chr1 323891 328581 NR_028325 0 + 328581 328581 0 3 169,58,4143, 0,396,547,
chr1 367658 368597 NM_001005277 0 + 367658 368597 0 1 939, 0,
chr1 367658 368597 NM_001005224 0 + 367658 368597 0 1 939, 0,
chr1 367658 368597 NM_001005221 0 + 367658 368597 0 1 939, 0,
qbm:BI0-5080 arq5x$
```

“genes.bed”  
N = 3,665

Find SNPs that have the potential to alter gene expression regulation by

# 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?

# Step 1: Find SNPs in CpG islands

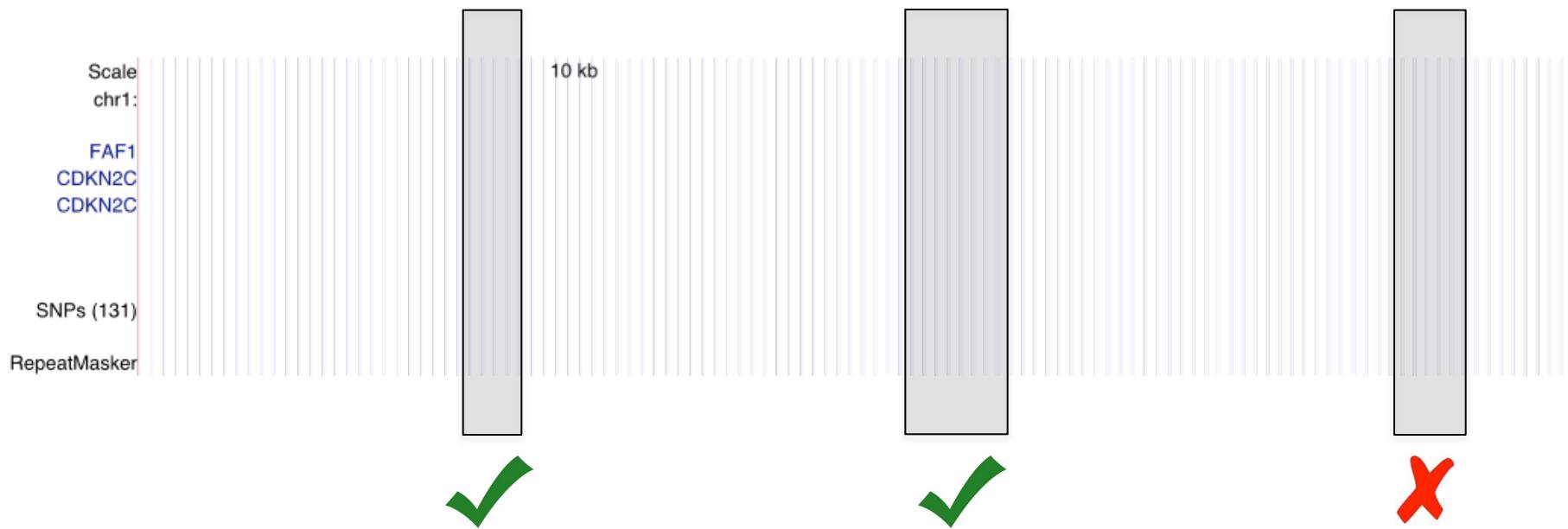
```
# How many SNPs are there on chrom 1?  
wc -l snps.bed  
2064872  
# Find SNPs that overlap CpG islands using intersectBed  
bedtools intersect -a snps.bed -b cpg.bed -wa >  
snps.cpg.bed  
# How many SNPs overlapped a CpG island?  
wc -l snps.cpg.bed  
14974
```

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

```
# How much of chrom1 do the CpG islands occupy?  
awk '{sum+=$5} END{print sum}' cpg.bed  
1881629
```

$$1881629 / 249250621 = 0.75\%$$

## 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

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

# 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
```

How do we know if it worked?