Intervals and Ranges for genome-scale data analysis

CSAMA 2012

© 2012 VJ Carey, PhD

Harvard Medical School

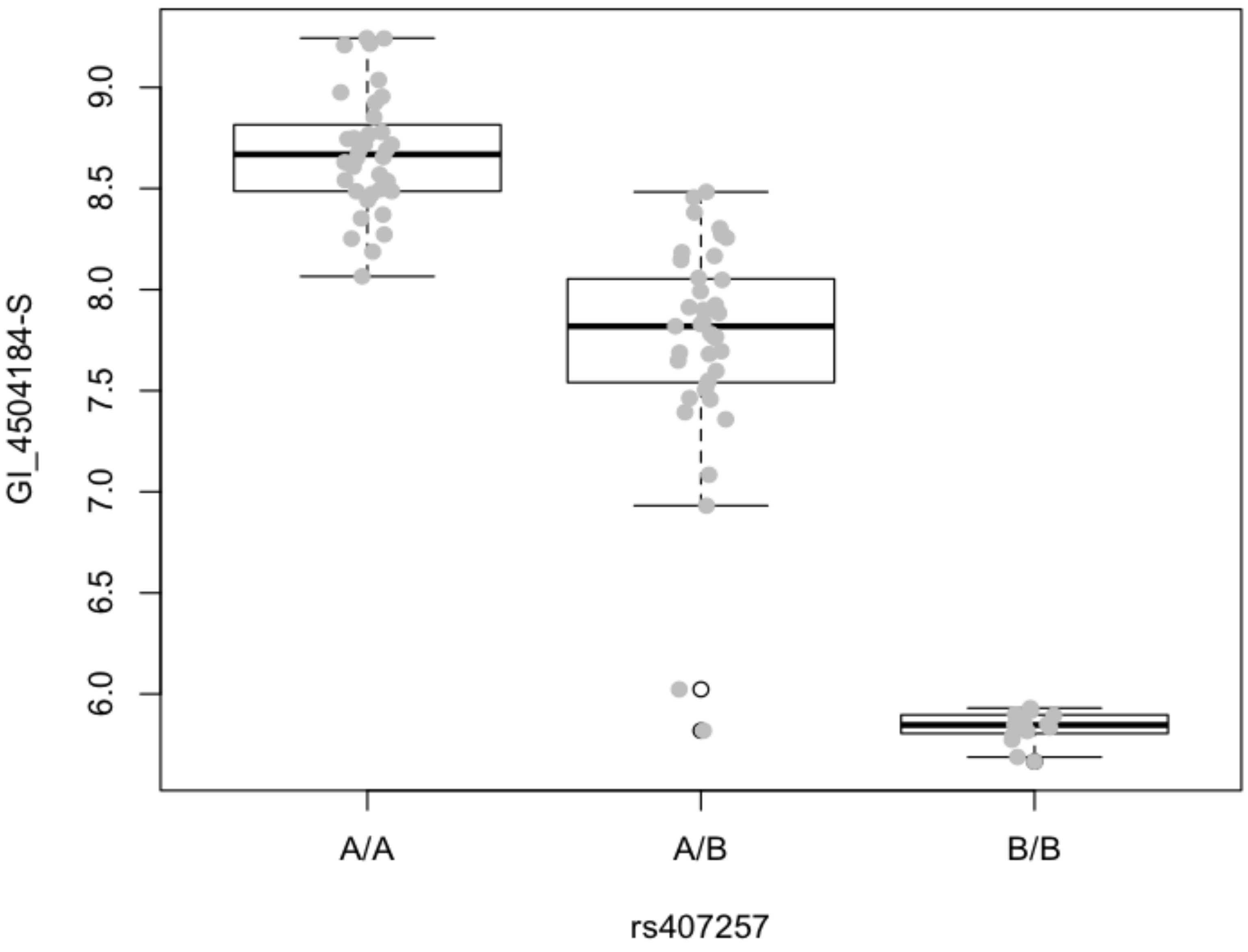
Road map

- Finishing up the reproducibility lecture with an illustration of the SummarizedExperiment container
  - X[G, S] is an idiom for selecting results from genome-scale experiment X
  - When features are indexed by genomic coordinates, what should X be/do?
- Bioconductor resources for sequences, gene models and efficient interval algebra

What is an eQTL?

- “expression” quantitative trait locus
- What are the ingredients, criteria?
- What are the roles of genomic coordinates and ranges?

Average expression varies by genotype – why?



Basic schematic

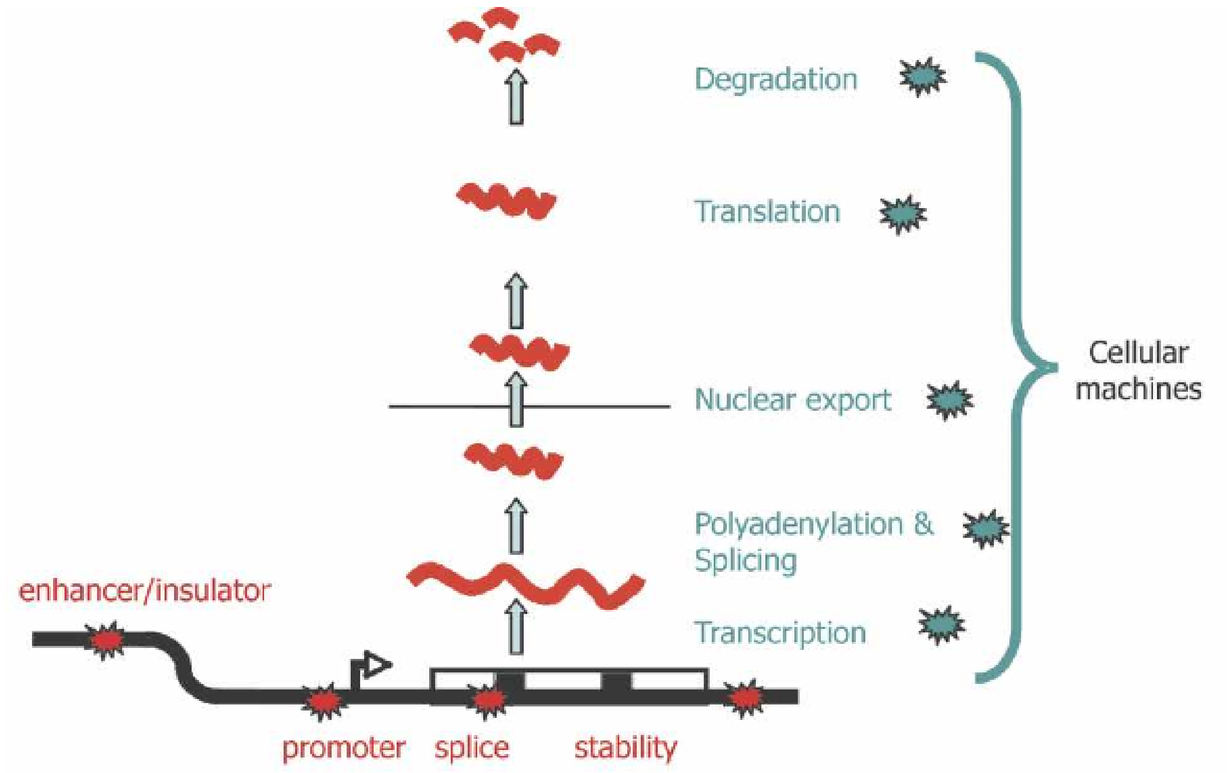
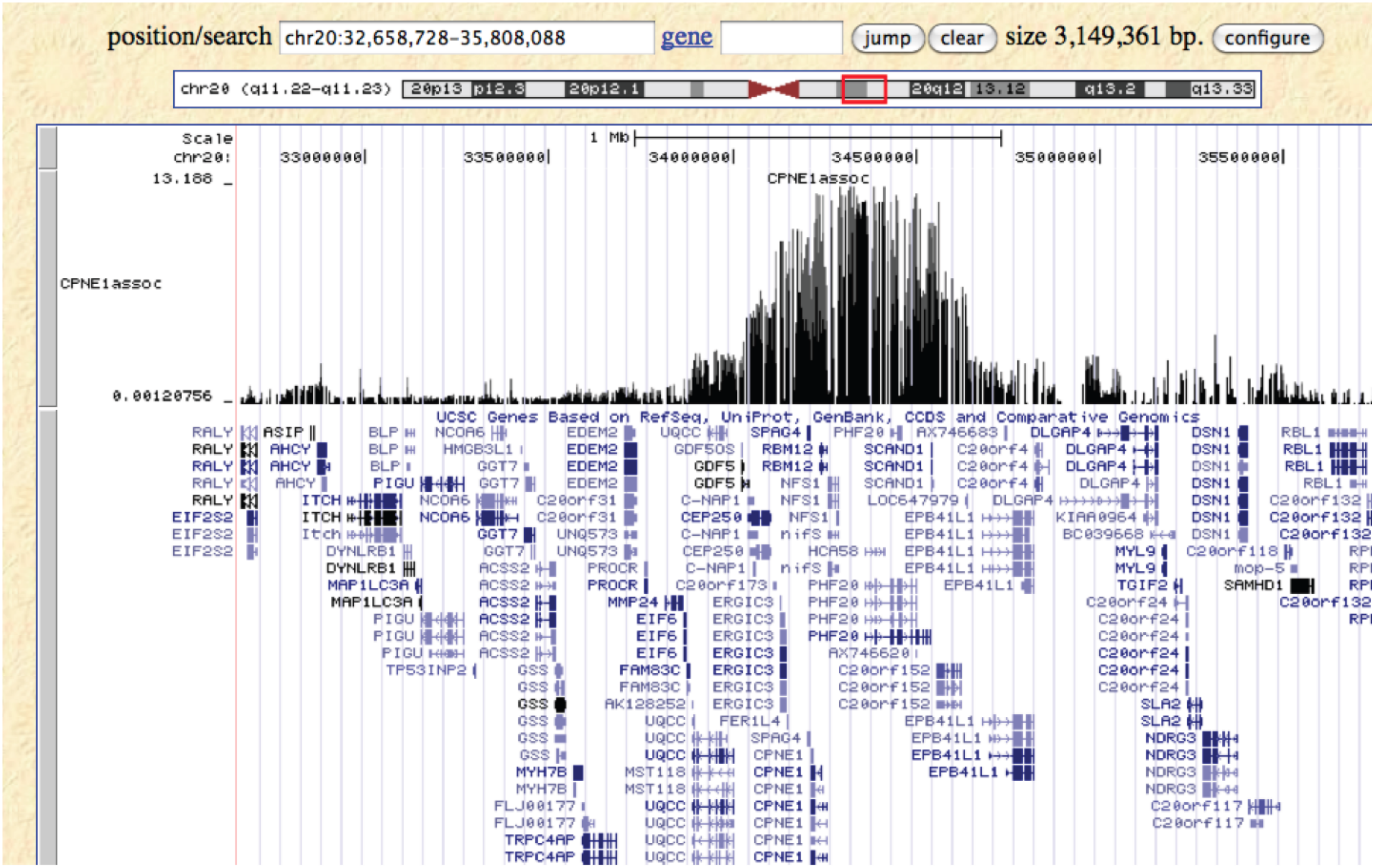


Figure 1. Plausible sites of action for genetic determinants of mRNA levels. Genetic variations influencing gene expression may reside within the regulatory sequences, promoters, enhancers, splice sites, and secondary structure motifs of the target gene and so be genetically in *cis* (red stars), or there may be variations in the molecular machinery that interact with *cis*-regulatory sequences and so act genetically in *trans* (blue stars).

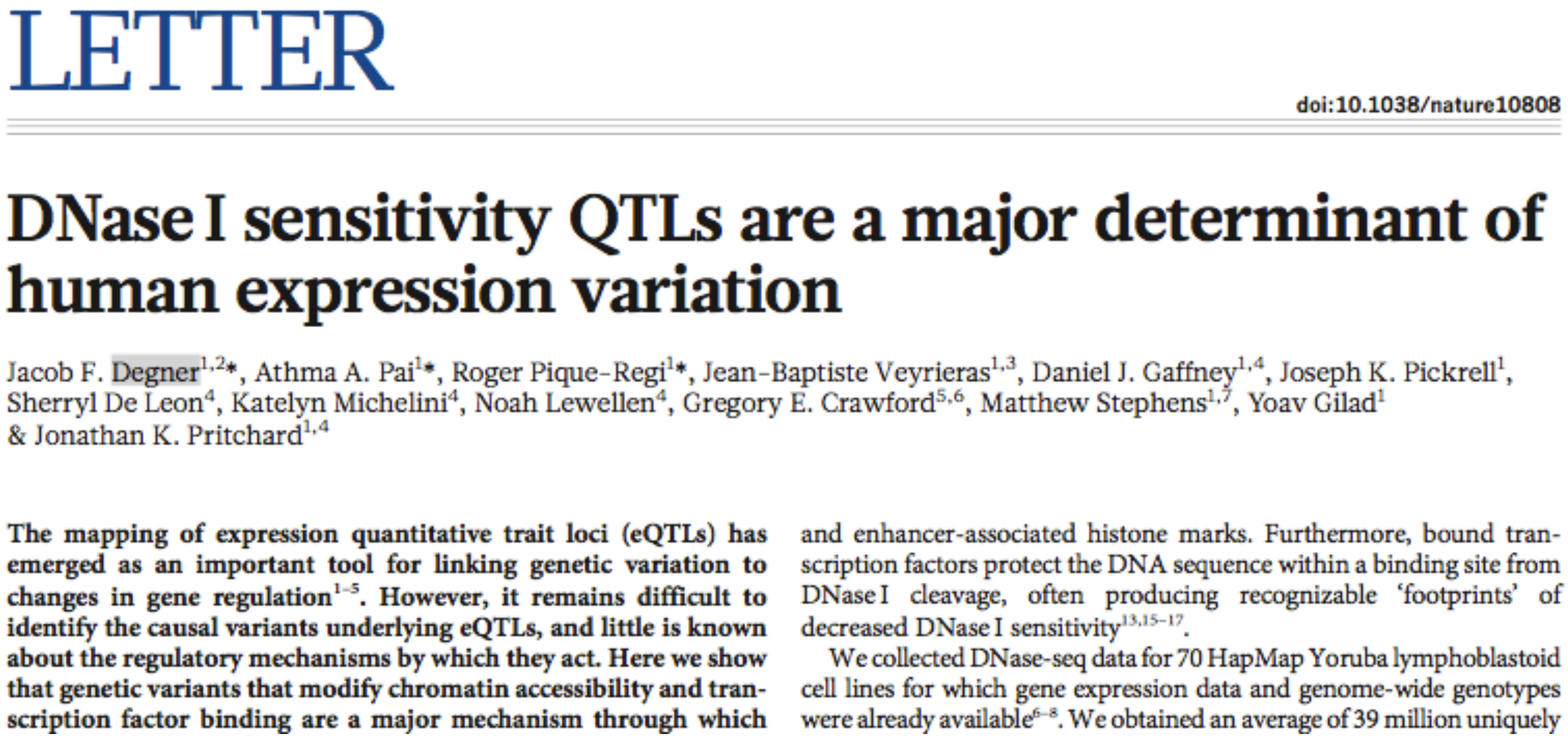
Range-related concepts

- Where is the DNA variant?
- Where is the transcript that covaries with genotype? (cis-radius, trans ….)
- What are the functional attributes of the region in which the variant lies?



position/search chr20:32,658,728-35,808,088 gene jump clear size 3,149,361 bp. configure

A high profile paper and some reproduction/extensibility exercises



LETTER

doi:10.1038/nature10808

DNase I sensitivity QTLs are a major determinant of human expression variation

Jacob F. Degner[1,2]*, Athma A. Pai[1]*, Roger Pique-Regi[1]*, Jean-Baptiste Veyrieras[1,3], Daniel J. Gaffney[1,4], Joseph K. Pickrell[1], Sherryl De Leon[4], Katelyn Michelini[4], Noah Lewellen[4], Gregory E. Crawford[5,6], Matthew Stephens[1,7], Yoav Gilad[1] & Jonathan K. Pritchard[1,4]

The mapping of expression quantitative trait loci (eQTLs) has emerged as an important tool for linking genetic variation to changes in gene regulation[1–5]. However, it remains difficult to identify the causal variants underlying eQTLs, and little is known about the regulatory mechanisms by which they act. Here we show that genetic variants that modify chromatin accessibility and transcription factor binding are a major mechanism through which

and enhancer-associated histone marks. Furthermore, bound transcription factors protect the DNA sequence within a binding site from DNase I cleavage, often producing recognizable 'footprints' of decreased DNase I sensitivity[13,15–17].

We collected DNase-seq data for 70 HapMap Yoruba lymphoblastoid cell lines for which gene expression data and genome-wide genotypes were already available[6–8]. We obtained an average of 39 million uniquely

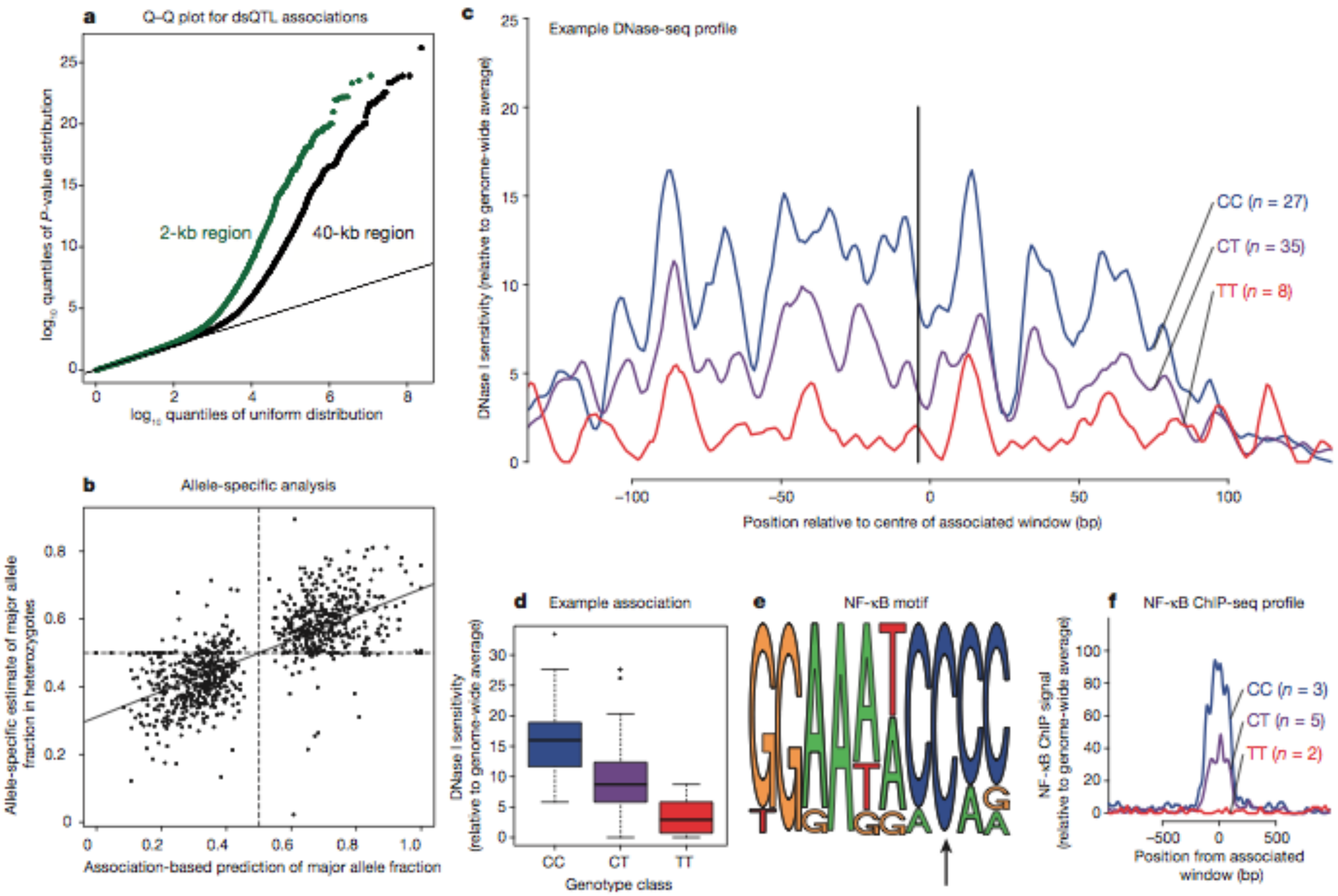


Figure 1 | Genome-wide identification of dsQTLs and a typical example. **a**, Q–Q plots for all tests of association between DNase I cut rates in 100-bp windows, and variants within 2-kb (green) and 40-kb (black) regions centred dsQTL (rs4953223). The black line indicates the position of the associated SNP. **d**, Box plot showing that rs4953223 is strongly associated with local chromatin accessibility ($P = 3 \times 10^{-13}$). **e**, The T allele, which is associated with low

a

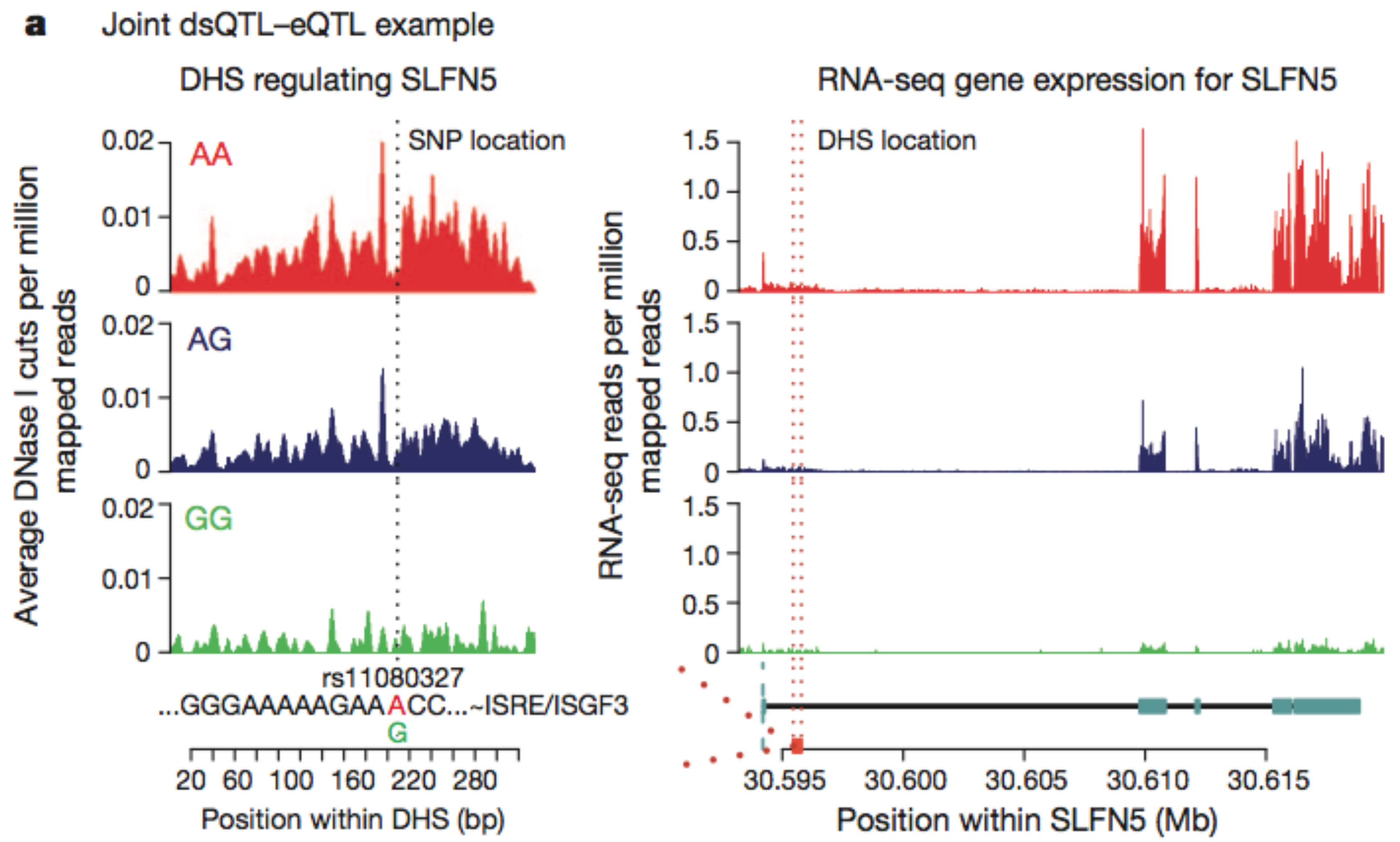
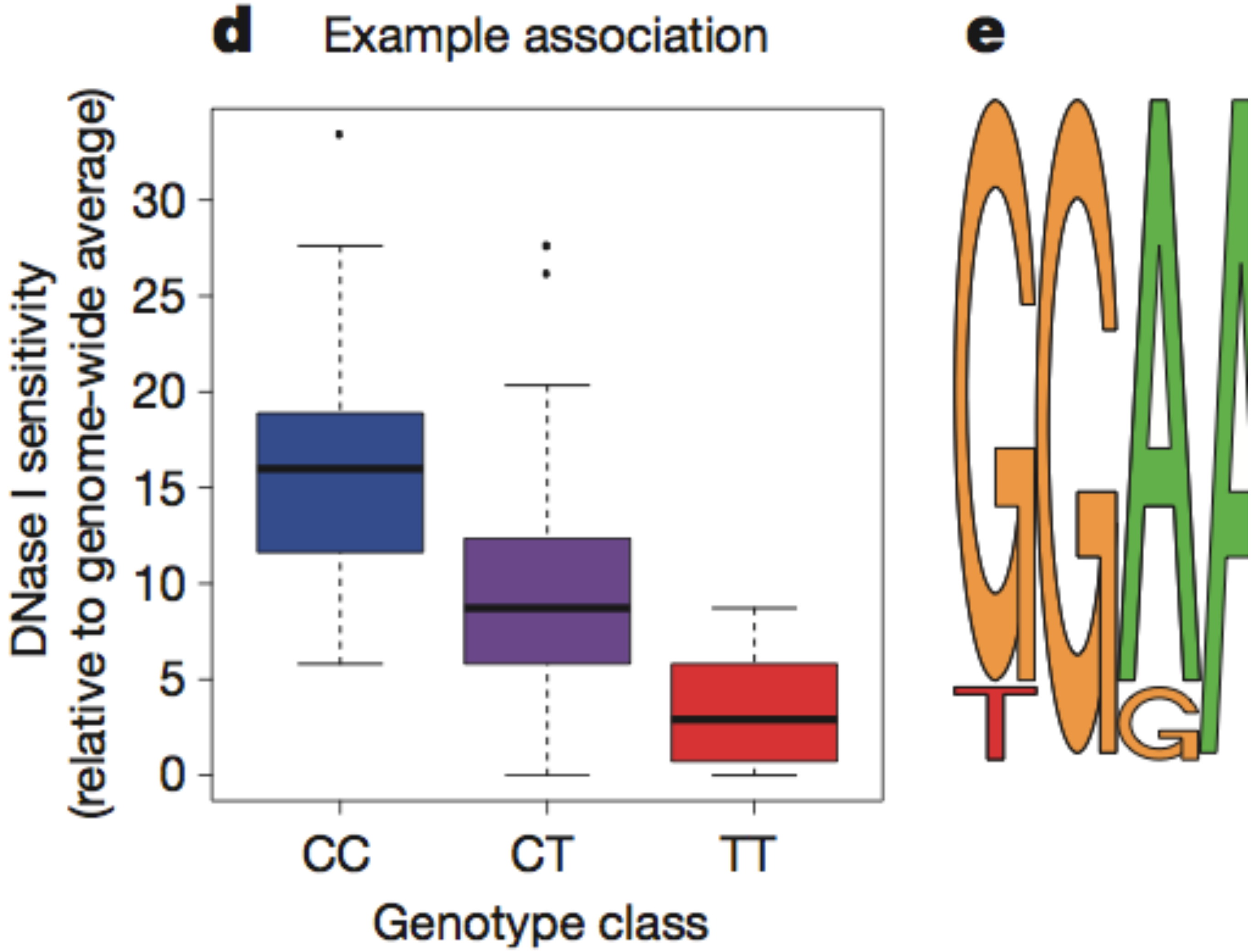


Figure 3 | Relationship between dsQTLs and eQTLs. a, Example of a dsQTL SNP that is also an eQTL for the gene *SLFN5*. The SNP disrupts an interferon- (right) measurer genotype at the p

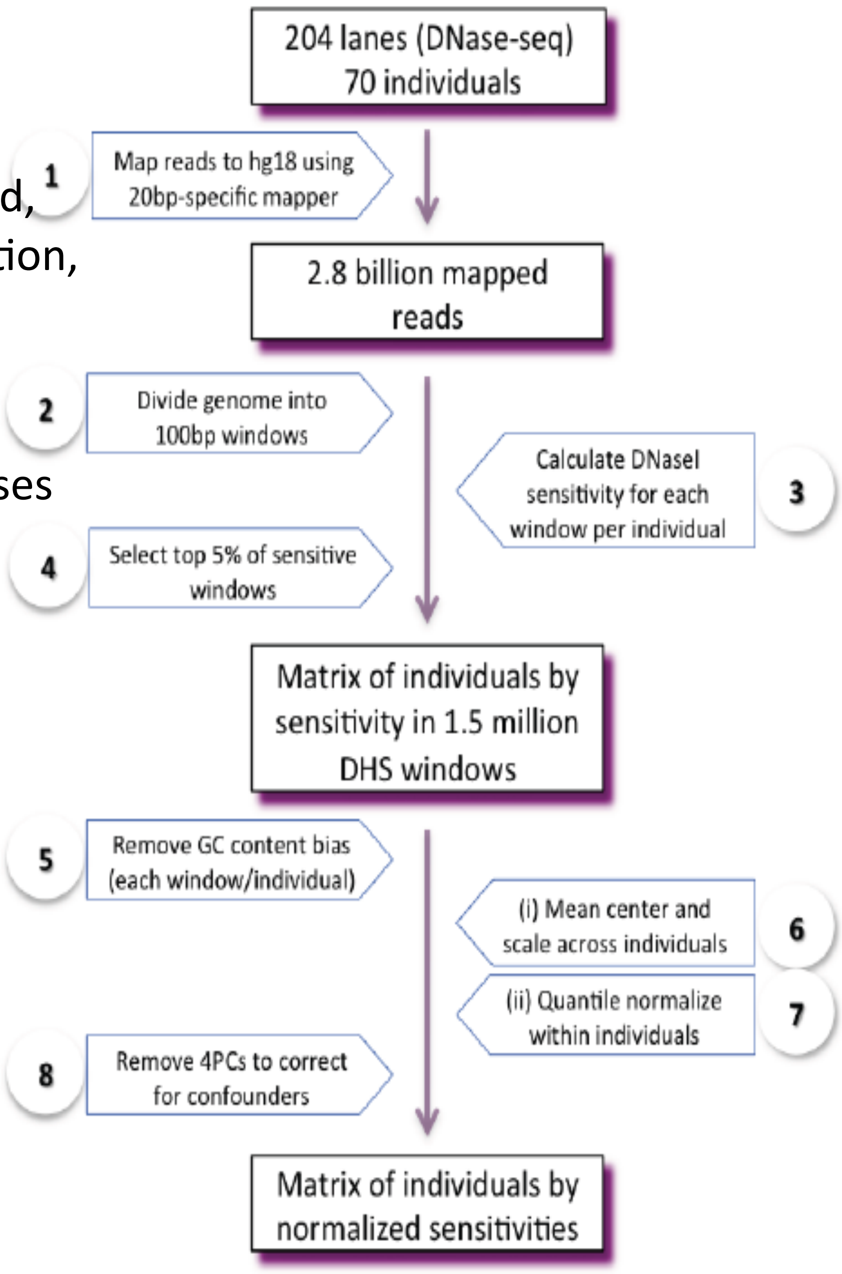
What can we do to make this finding concretely reproducible? Extensible?



and a typical example. dsQTL (rs4953223).

Vulnerabilities: hg18,
100bp window,
5% sensitivity threshold,
GC content bias reduction,
4 PC (SVA-like),
cis radius

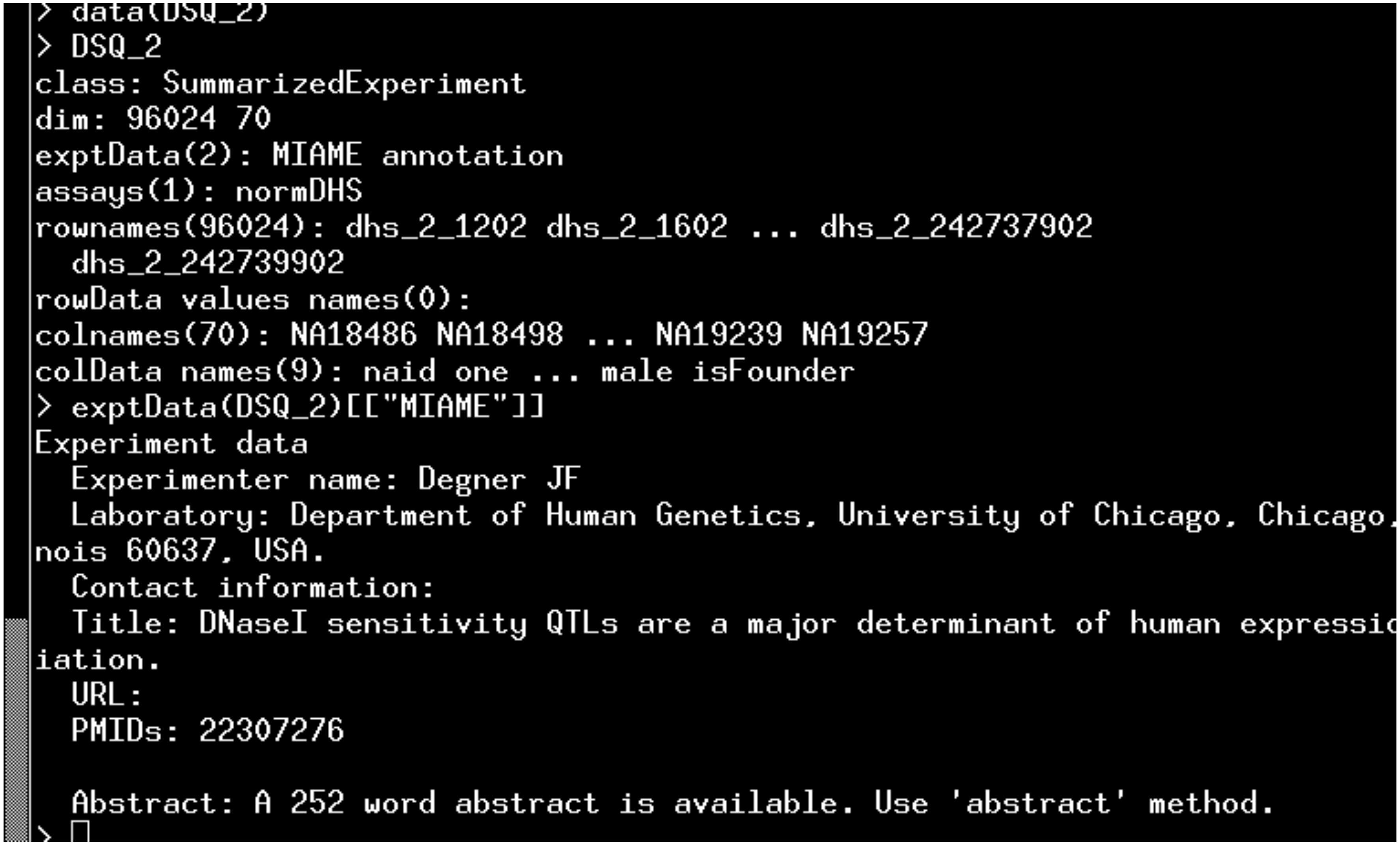
Some sensitivity analyses described


204 lanes (DNase-seq)
70 individuals
1
Map reads to hg18 using 20bp-specific mapper
2.8 billion mapped reads
2
Divide genome into 100bp windows
Calculate DNaseI sensitivity for each window per individual
3
4
Select top 5% of sensitive windows
Matrix of individuals by sensitivity in 1.5 million DHS windows
5
Remove GC content bias (each window/individual)
(i) Mean center and scale across individuals
6
(ii) Quantile normalize within individuals
7
8
Remove 4PCs to correct for confounders
Matrix of individuals by normalized sensitivities

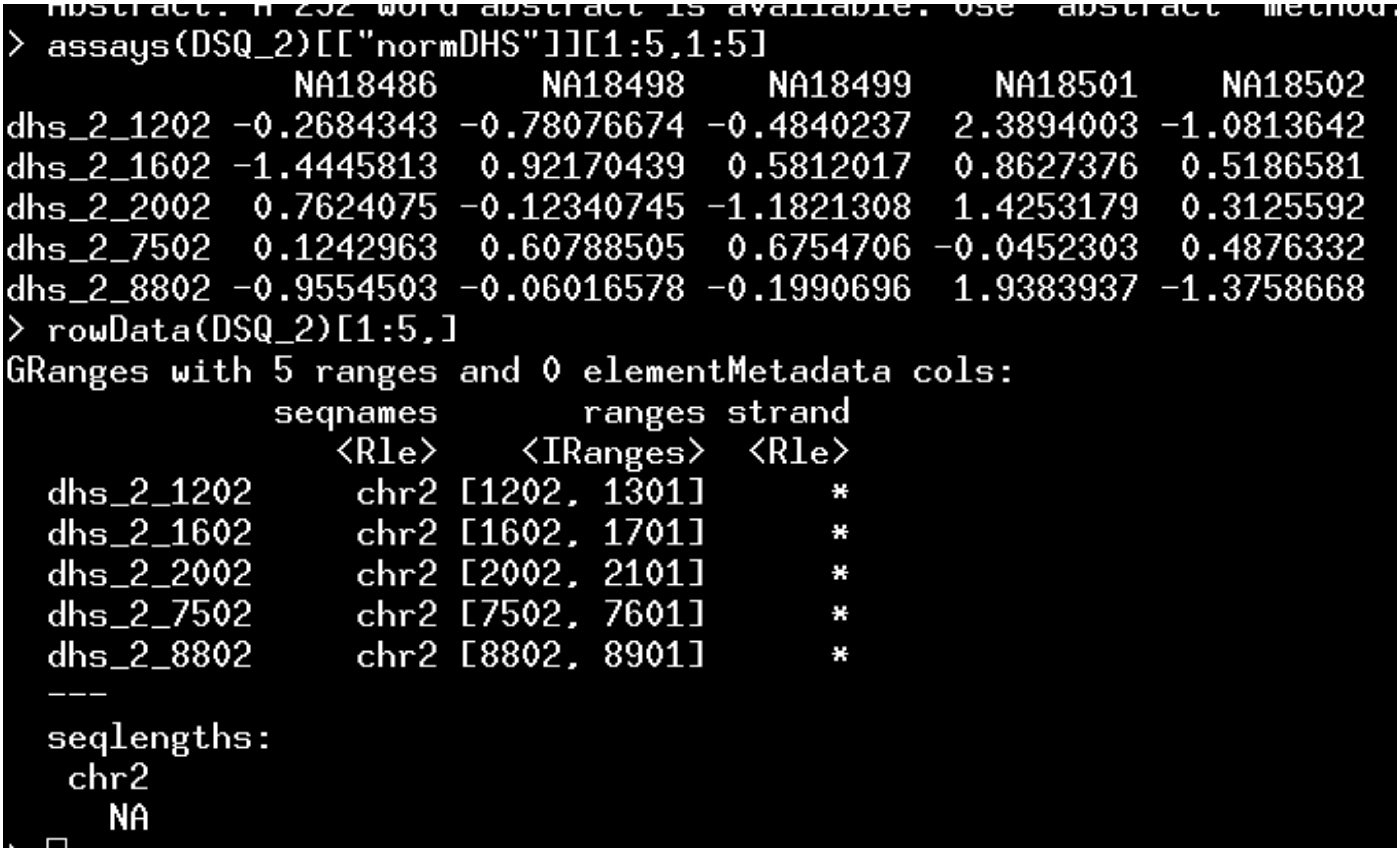
New directions in feature/test volume?

- DNase-seq read-counts were assembled in a 100bp tiling of the genome, so 30 million scores per individual
- dsQTL analysis involves associating ~30 million imputed SNP with each of these scores; cis filtering reduces volume considerably
- How should we manage the basic quantities?
  - Stage 1: SummarizedExperiment demo
  - Stage 2: integrative DHS+genotype container permitting very high-volume testing with small footprint

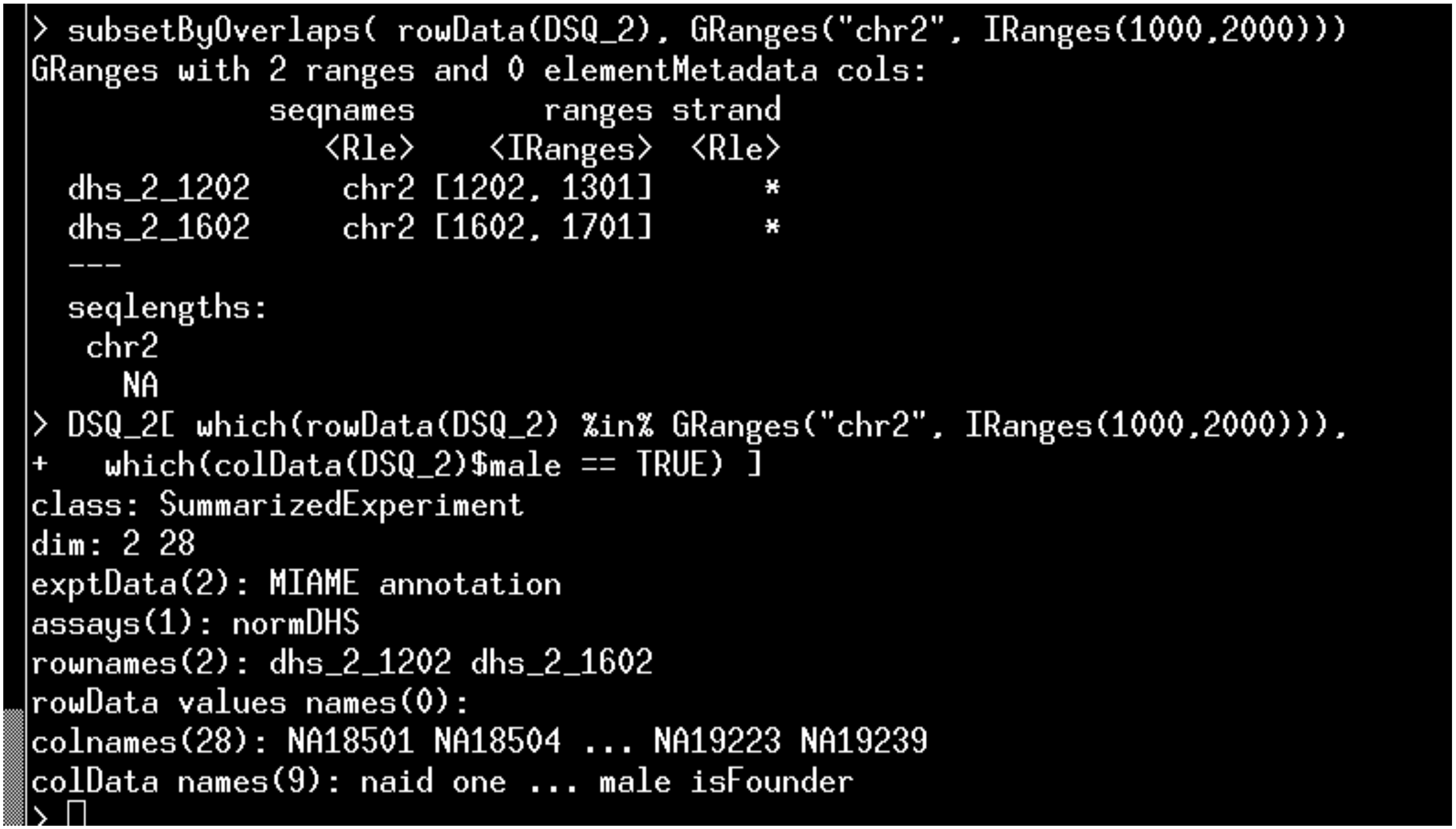
The dsQTL experimental data package



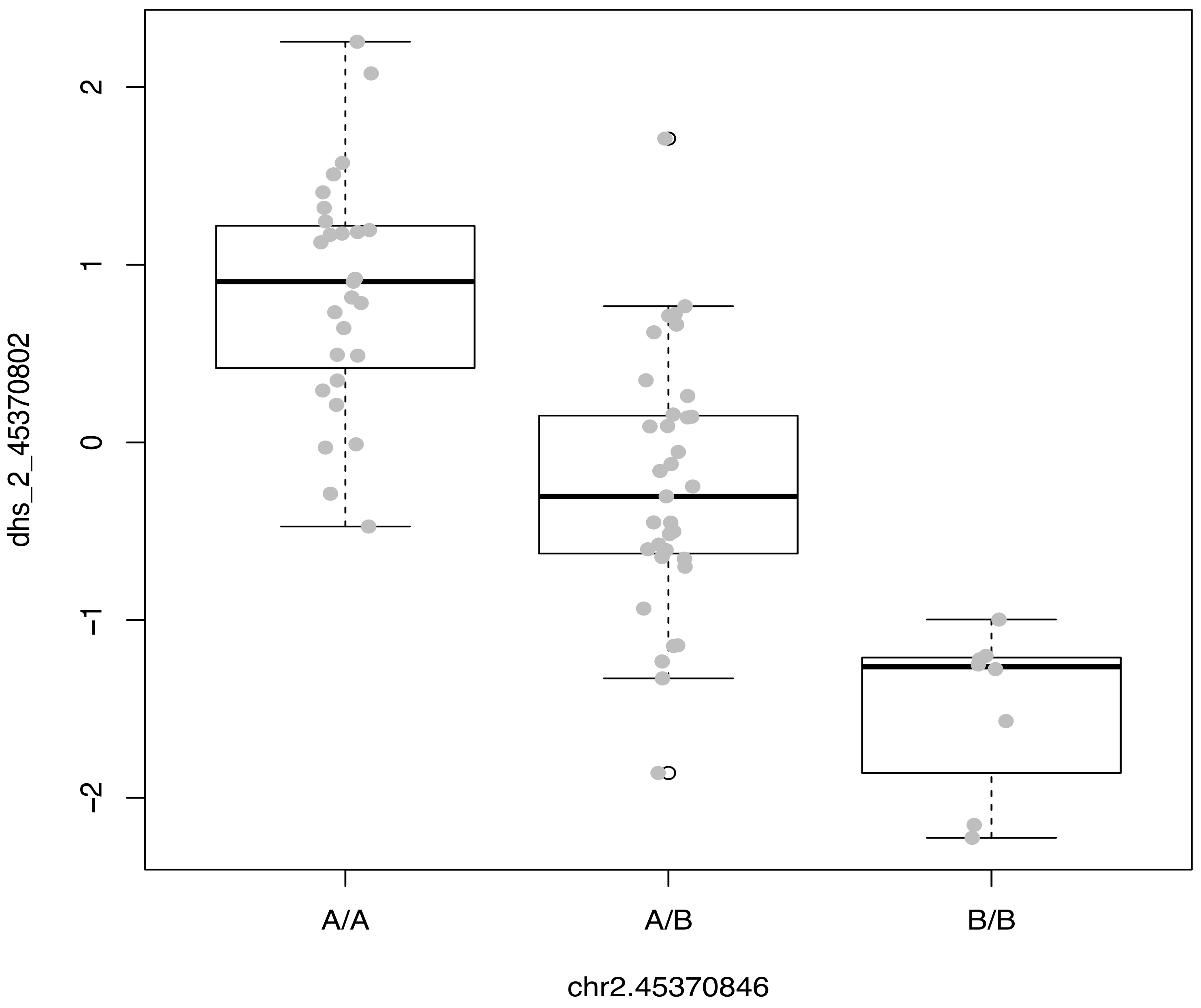
```
> data(DSQ_2)
> DSQ_2
class: SummarizedExperiment
dim: 96024 70
exptData(2): MIAME annotation
assays(1): normDHS
rownames(96024): dhs_2_1202 dhs_2_1602 ... dhs_2_242737902
  dhs_2_242739902
rowData values names(0):
colnames(70): NA18486 NA18498 ... NA19239 NA19257
colData names(9): naid one ... male isFounder
> exptData(DSQ_2)[["MIAME"]]
Experiment data
  Experimenter name: Degner JF
  Laboratory: Department of Human Genetics, University of Chicago, Chicago,
nois 60637, USA.
  Contact information:
  Title: DNaseI sensitivity QTLs are a major determinant of human expressio
iation.
  URL:
  PMIDs: 22307276

  Abstract: A 252 word abstract is available. Use 'abstract' method.
> 
```



```
 Abstract: A 252 word abstract is available. Use 'abstract' method.
> assays(DSQ_2)[["normDHS"]][1:5,1:5]
              NA18486     NA18498    NA18499    NA18501    NA18502
dhs_2_1202 -0.2684343 -0.78076674 -0.4840237  2.3894003 -1.0813642
dhs_2_1602 -1.4445813  0.92170439  0.5812017  0.8627376  0.5186581
dhs_2_2002  0.7624075 -0.12340745 -1.1821308  1.4253179  0.3125592
dhs_2_7502  0.1242963  0.60788505  0.6754706 -0.0452303  0.4876332
dhs_2_8802 -0.9554503 -0.06016578 -0.1990696  1.9383937 -1.3758668
> rowData(DSQ_2)[1:5,]
GRanges with 5 ranges and 0 elementMetadata cols:
             seqnames       ranges strand
                <Rle>    <IRanges>  <Rle>
  dhs_2_1202     chr2 [1202, 1301]      *
  dhs_2_1602     chr2 [1602, 1701]      *
  dhs_2_2002     chr2 [2002, 2101]      *
  dhs_2_7502     chr2 [7502, 7601]      *
  dhs_2_8802     chr2 [8802, 8901]      *
  ---
  seqlengths:
   chr2
     NA
> 
```



```
> subsetByOverlaps( rowData(DSQ_2), GRanges("chr2", IRanges(1000,2000)))
GRanges with 2 ranges and 0 elementMetadata cols:
             seqnames       ranges strand
                <Rle>    <IRanges>  <Rle>
  dhs_2_1202     chr2 [1202, 1301]      *
  dhs_2_1602     chr2 [1602, 1701]      *
  ---
  seqlengths:
   chr2
     NA
> DSQ_2[ which(rowData(DSQ_2) %in% GRanges("chr2", IRanges(1000,2000))),
+   which(colData(DSQ_2)$male == TRUE) ]
class: SummarizedExperiment
dim: 2 28
exptData(2): MIAME annotation
assays(1): normDHS
rownames(2): dhs_2_1202 dhs_2_1602
rowData values names(0):
colnames(28): NA18501 NA18504 ... NA19223 NA19239
colData names(9): naid one ... male isFounder
> 
```



dhs_2_45370802
2
1
0
−1
−2
A/A
A/B
B/B
chr2.45370846

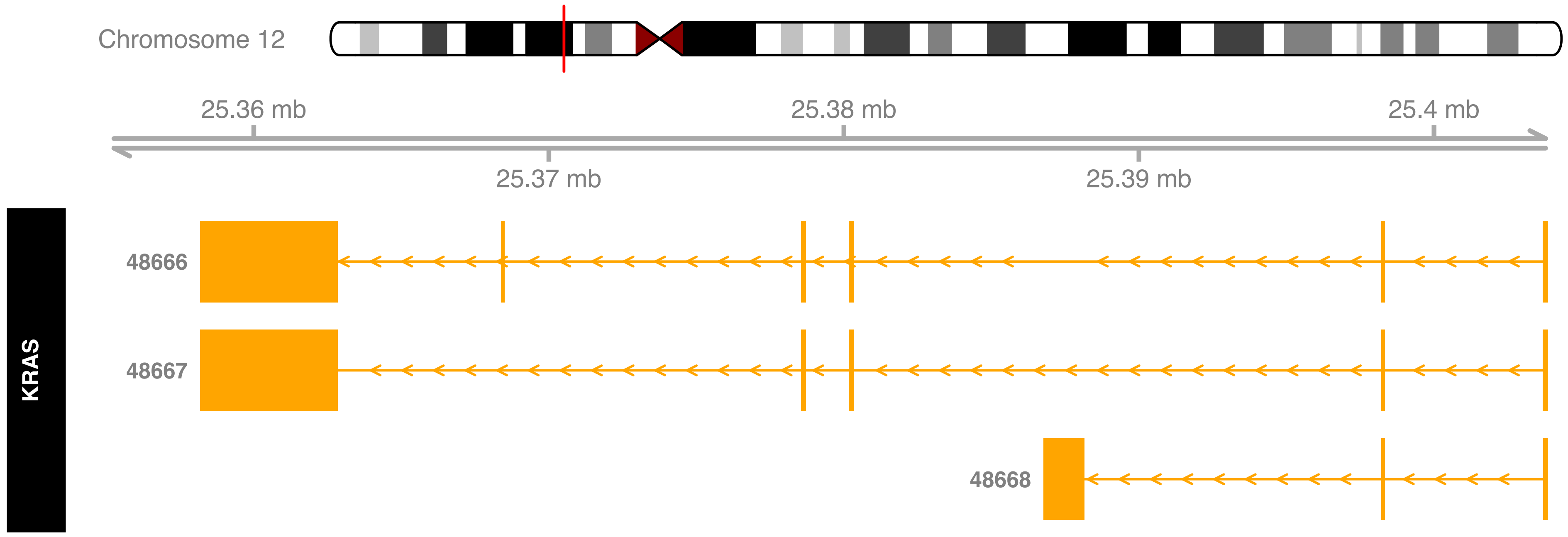
Recap

- Tight binding of metadata to assay data for many millions of features per sample
- Fast, idiomatic query resolution using genomic coordinates
- X[G, S] has values for selected features and samples, responds to any method on X
- Relax restrictions on the “back end” when the resources are really massive
- Often the cooked resources are manageable and can reside in such containers, facilitating easy distribution and uptake: extensibility

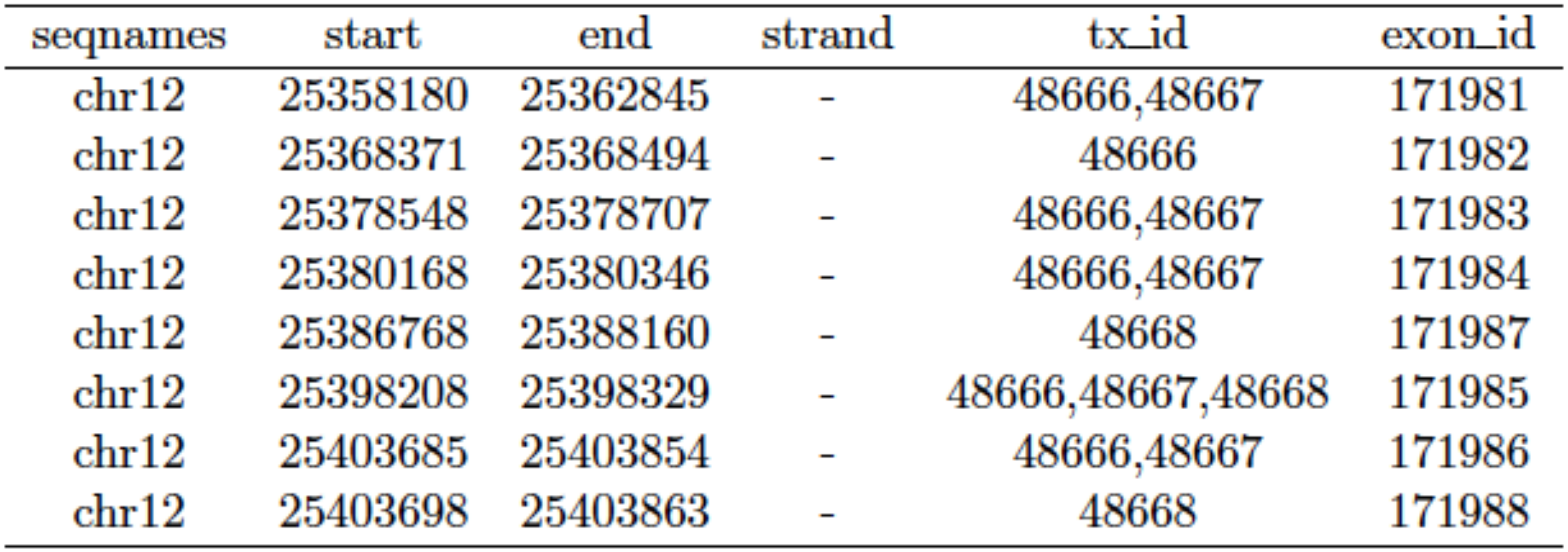
Ranges computations in detail

- Acquiring and manipulating a gene model
- The TxDb transcript range databases
- GRanges instances; DataFrame, elementMetadata
- The IRanges API
- Example: testing for reduced frequency of SNP incidence in regions confidently scored as TFBS

Visualizing a simple model for KRAS

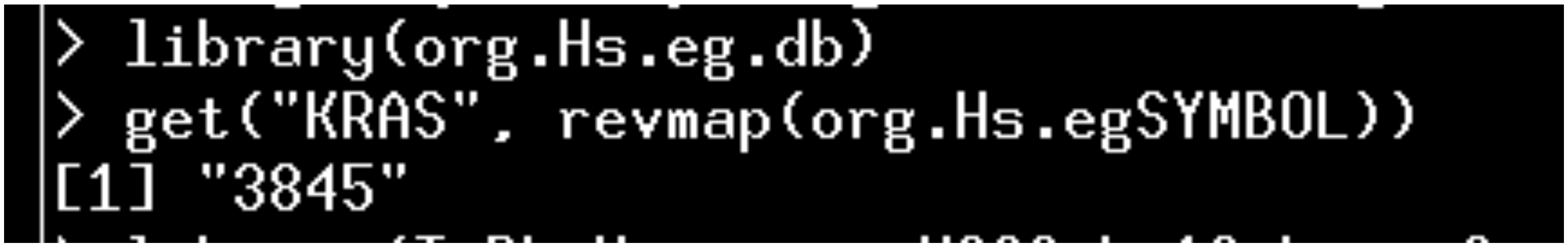


A textual view of the model



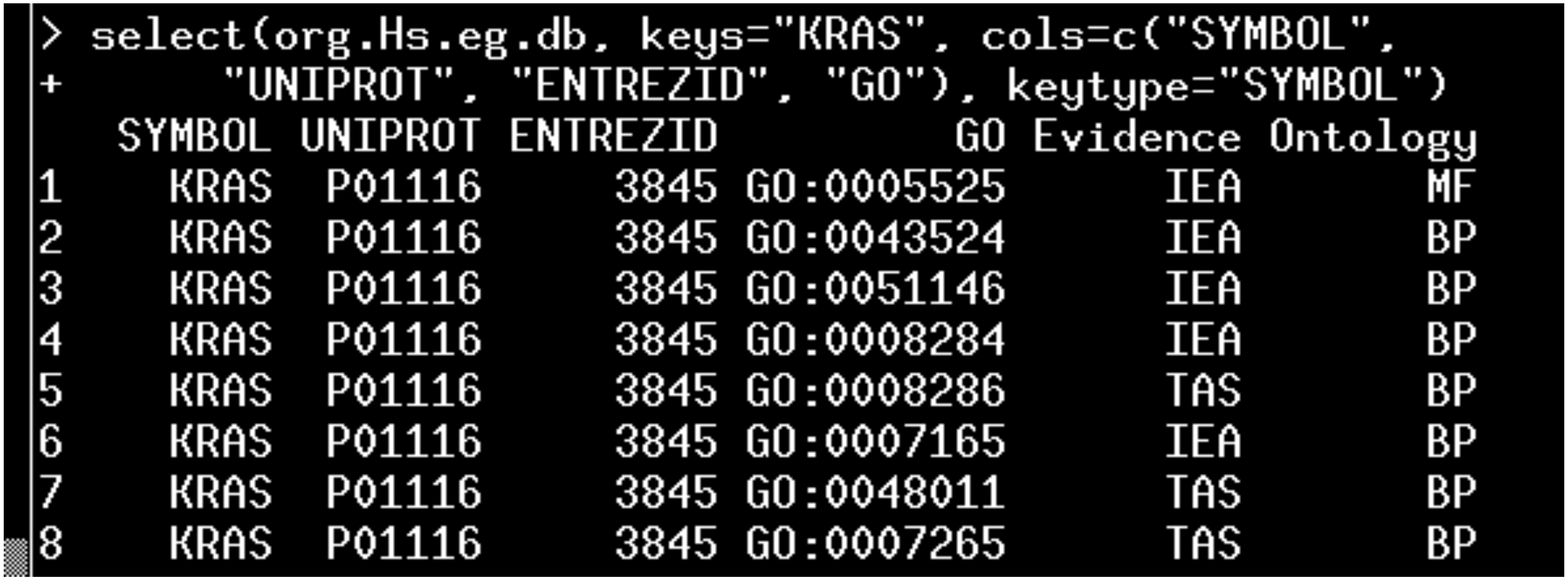
seqnames	start	end	strand	tx_id	exon_id
chr12	25358180	25362845	-	48666,48667	171981
chr12	25368371	25368494	-	48666	171982
chr12	25378548	25378707	-	48666,48667	171983
chr12	25380168	25380346	-	48666,48667	171984
chr12	25386768	25388160	-	48668	171987
chr12	25398208	25398329	-	48666,48667,48668	171985
chr12	25403685	25403854	-	48666,48667	171986
chr12	25403698	25403863	-	48668	171988

Finding an institutional identifier



```
> library(org.Hs.eg.db)
> get("KRAS", revmap(org.Hs.egSYMBOL))
[1] "3845"
```

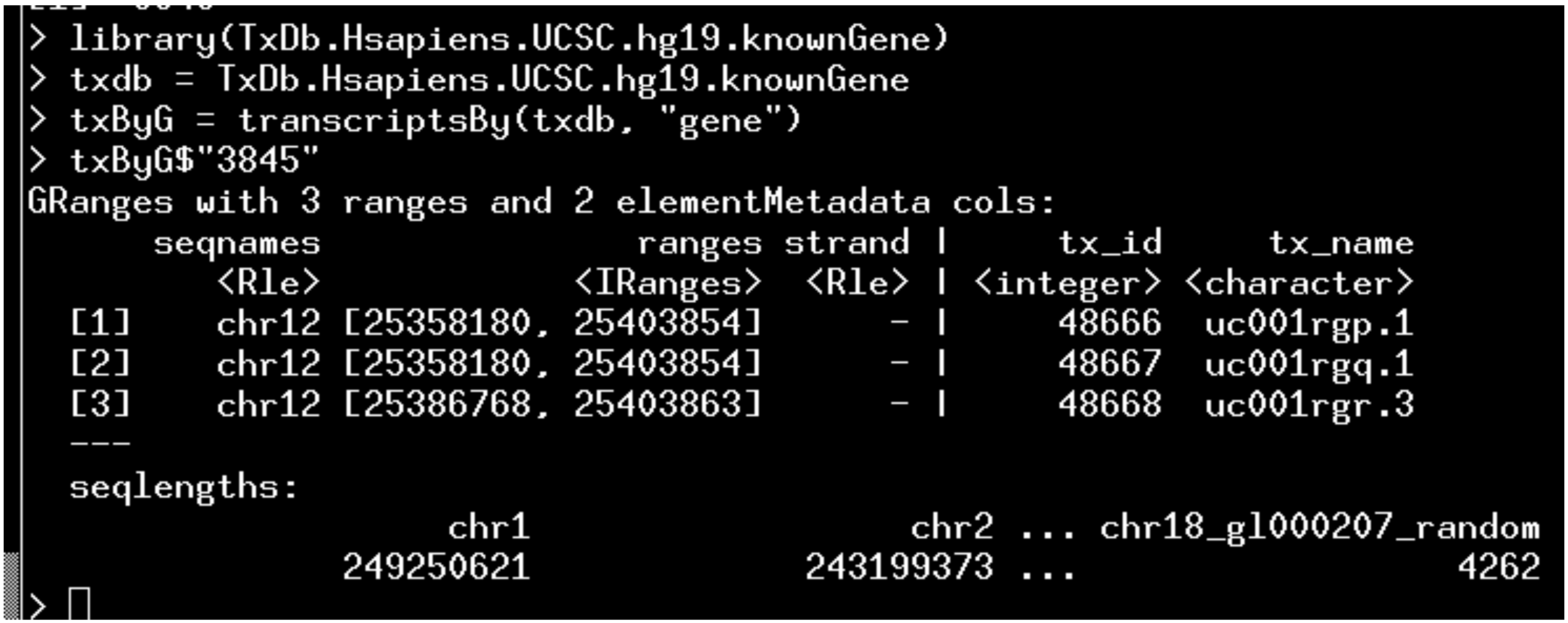
An up-to-date resolution approach



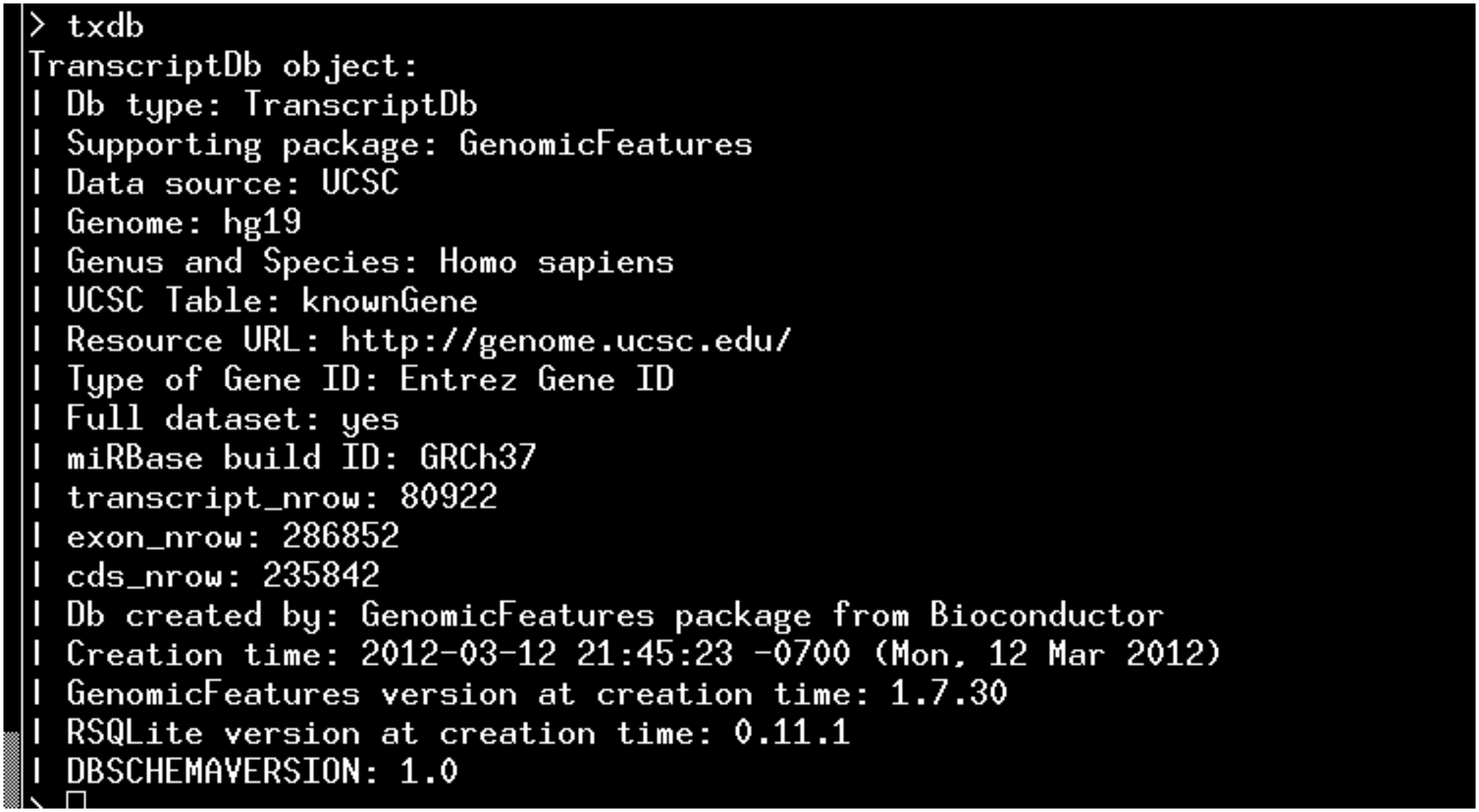
```
> select(org.Hs.eg.db, keys="KRAS", cols=c("SYMBOL",
+       "UNIPROT", "ENTREZID", "GO"), keytype="SYMBOL")
  SYMBOL UNIPROT ENTREZID         GO Evidence Ontology
1   KRAS  P01116     3845 GO:0005525      IEA       MF
2   KRAS  P01116     3845 GO:0043524      IEA       BP
3   KRAS  P01116     3845 GO:0051146      IEA       BP
4   KRAS  P01116     3845 GO:0008284      IEA       BP
5   KRAS  P01116     3845 GO:0008286      TAS       BP
6   KRAS  P01116     3845 GO:0007165      IEA       BP
7   KRAS  P01116     3845 GO:0048011      TAS       BP
8   KRAS  P01116     3845 GO:0007265      TAS       BP
```

…

Once we know the ENTREZID, we can get "the UCSC transcript model"

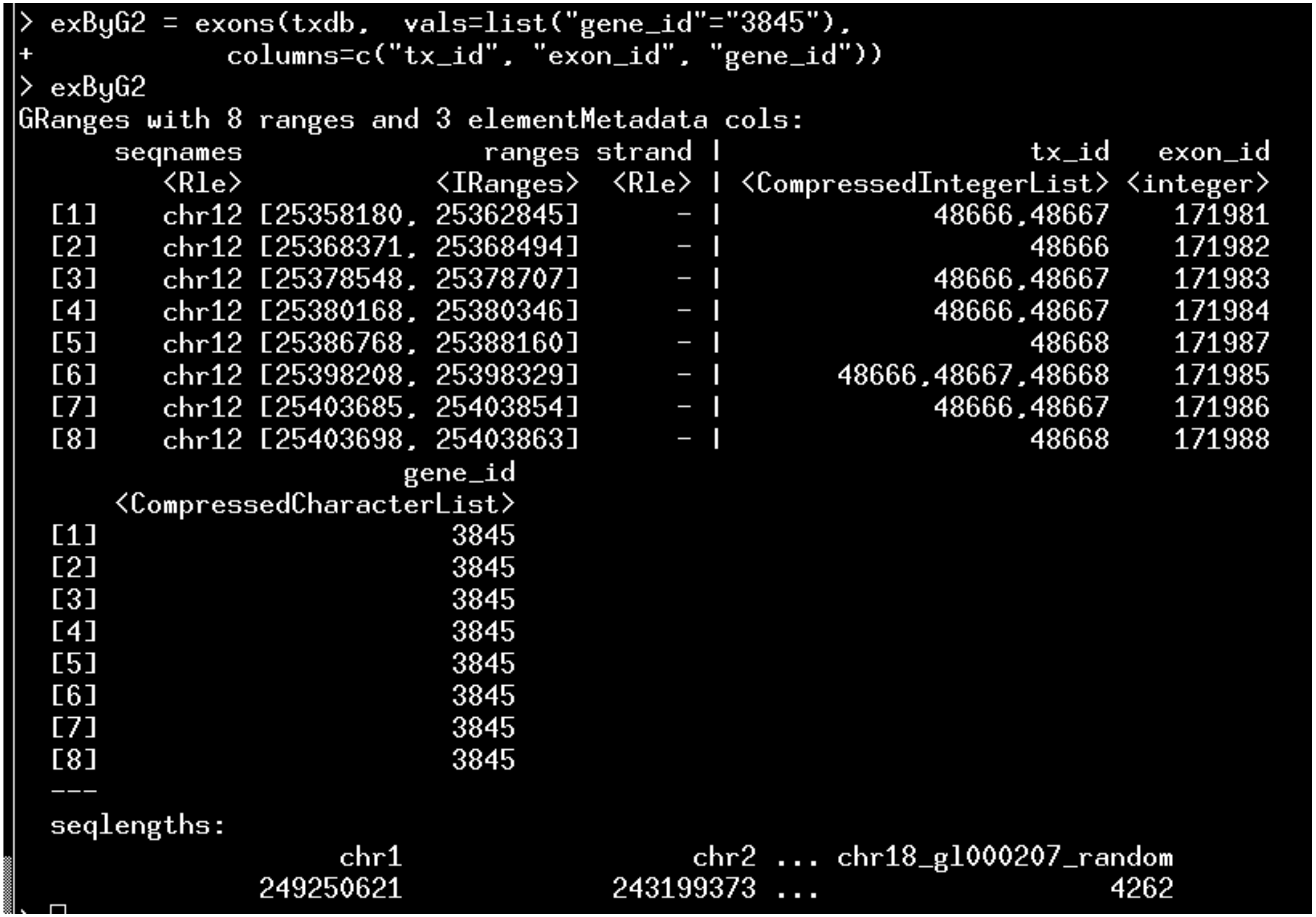


```
> library(TxDb.Hsapiens.UCSC.hg19.knownGene)
> txdb = TxDb.Hsapiens.UCSC.hg19.knownGene
> txByG = transcriptsBy(txdb, "gene")
> txByG$"3845"
GRanges with 3 ranges and 2 elementMetadata cols:
      seqnames               ranges strand |     tx_id     tx_name
         <Rle>            <IRanges>  <Rle> | <integer> <character>
  [1]    chr12 [25358180, 25403854]      - |     48666  uc001rgp.1
  [2]    chr12 [25358180, 25403854]      - |     48667  uc001rgq.1
  [3]    chr12 [25386768, 25403863]      - |     48668  uc001rgr.3
  ---
  seqlengths:
                  chr1                  chr2 ... chr18_gl000207_random
             249250621             243199373 ...                  4262
> 
```

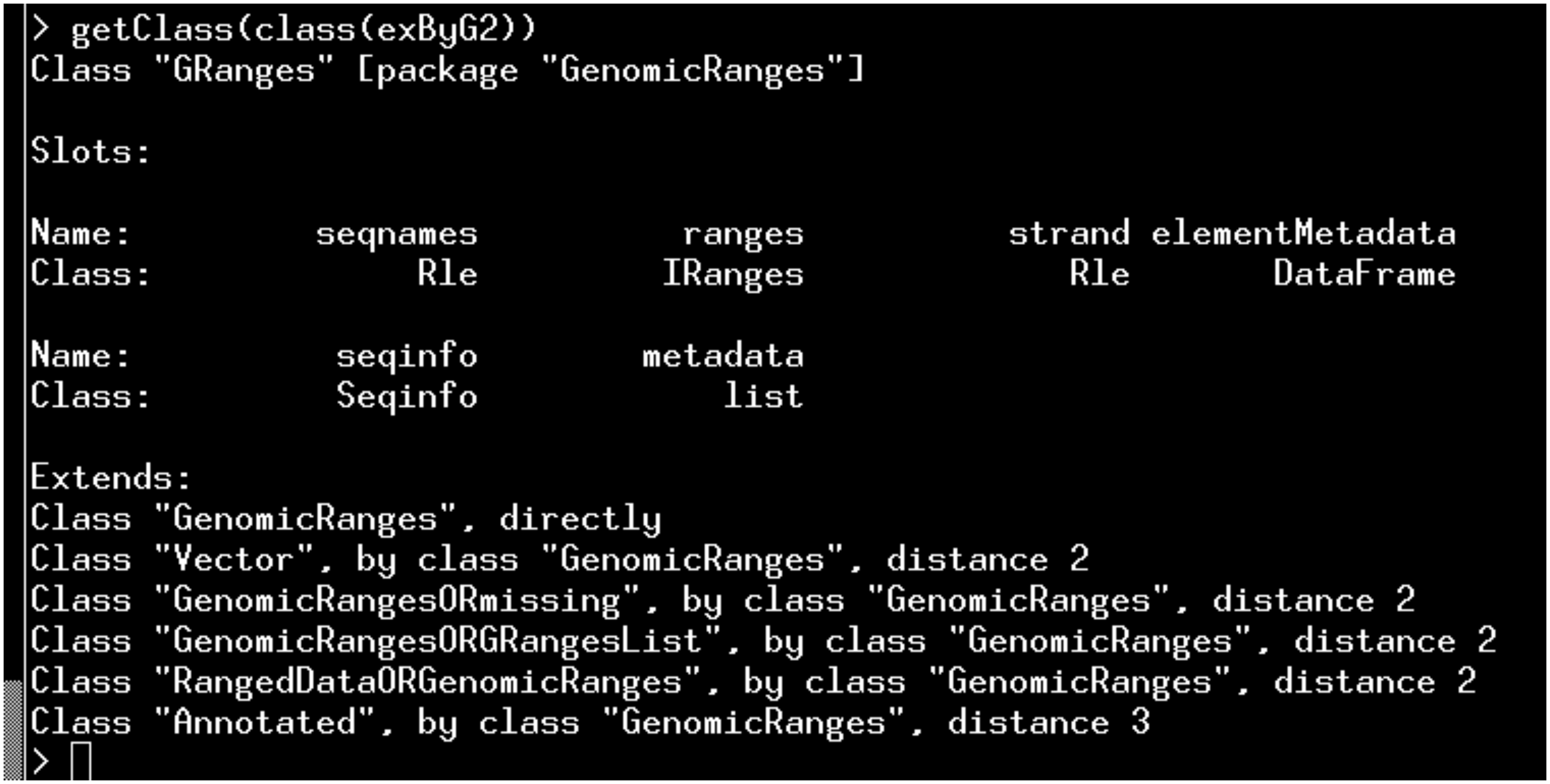


```
> txdb
TranscriptDb object:
| Db type: TranscriptDb
| Supporting package: GenomicFeatures
| Data source: UCSC
| Genome: hg19
| Genus and Species: Homo sapiens
| UCSC Table: knownGene
| Resource URL: http://genome.ucsc.edu/
| Type of Gene ID: Entrez Gene ID
| Full dataset: yes
| miRBase build ID: GRCh37
| transcript_nrow: 80922
| exon_nrow: 286852
| cds_nrow: 235842
| Db created by: GenomicFeatures package from Bioconductor
| Creation time: 2012-03-12 21:45:23 -0700 (Mon, 12 Mar 2012)
| GenomicFeatures version at creation time: 1.7.30
| RSQLite version at creation time: 0.11.1
| DBSCHEMAVERSION: 1.0
>
```

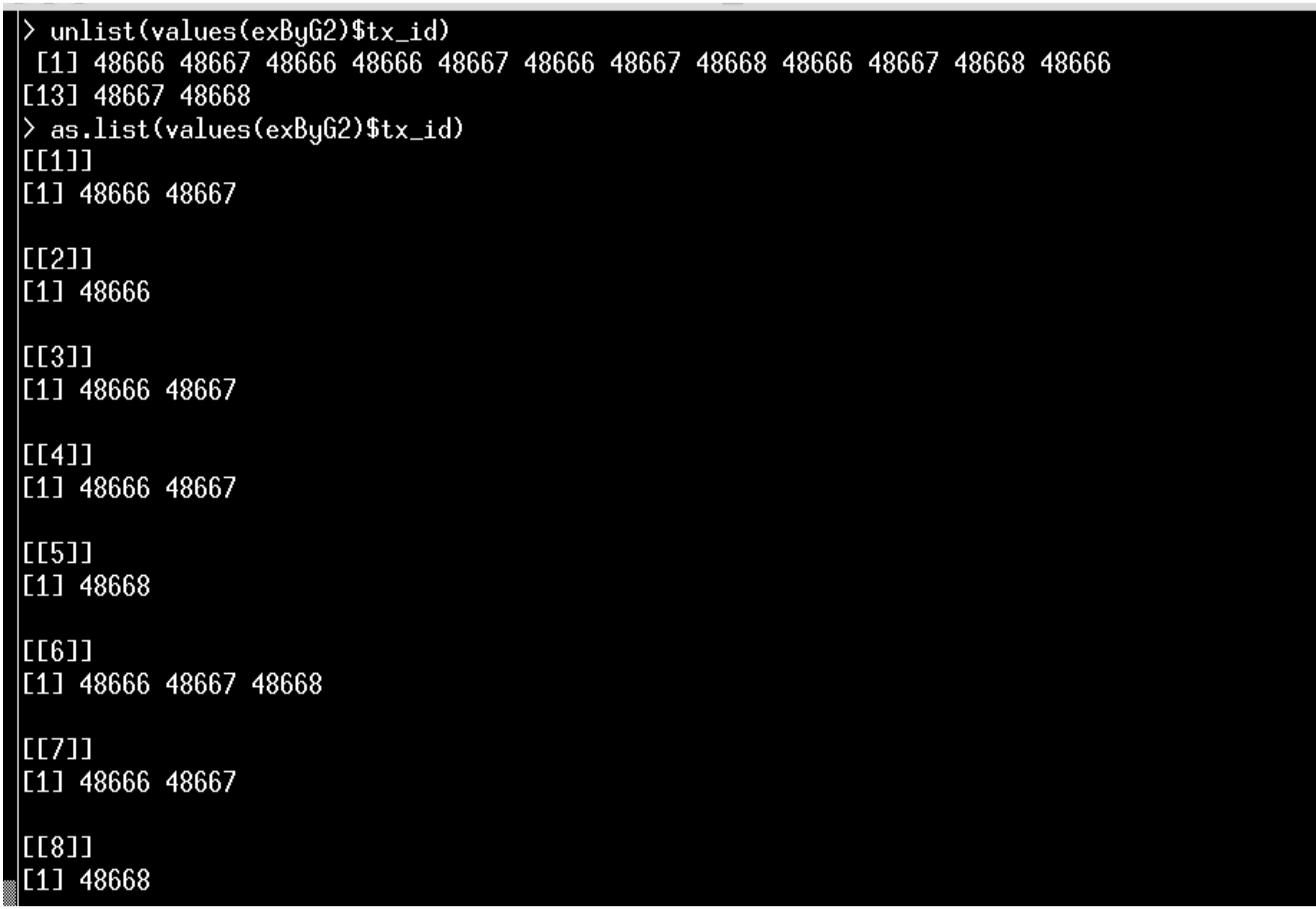
Retrieving the “UCSC exon model” for KRAS: 8 exons, 3 tx



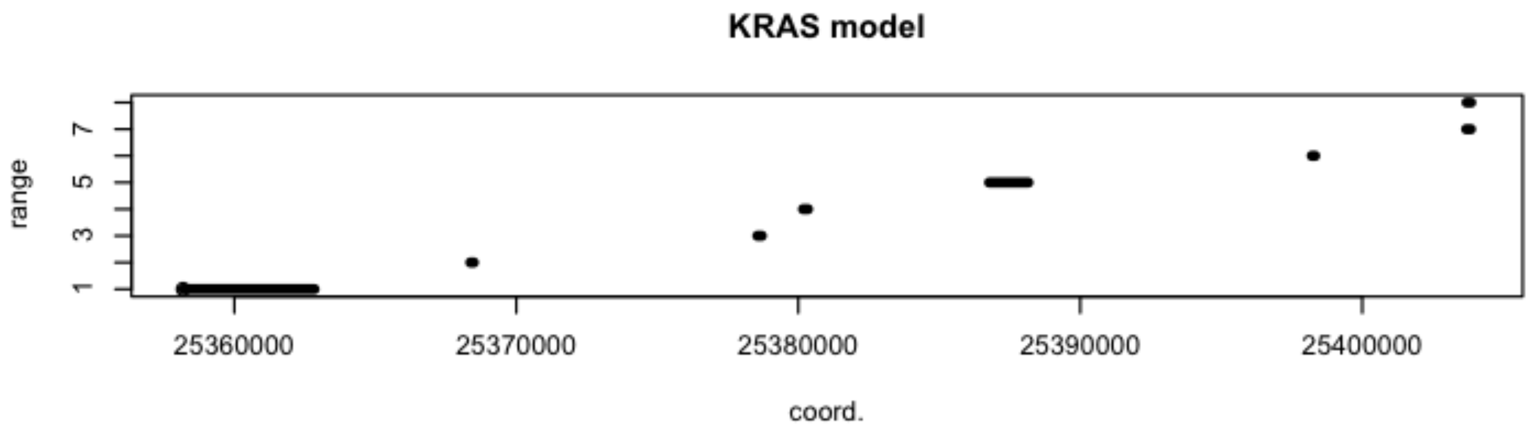
```
> exByG2 = exons(txdb,  vals=list("gene_id"="3845"),
+             columns=c("tx_id", "exon_id", "gene_id"))
> exByG2
GRanges with 8 ranges and 3 elementMetadata cols:
      seqnames                 ranges strand |                    tx_id   exon_id
         <Rle>              <IRanges>  <Rle> | <CompressedIntegerList> <integer>
  [1]    chr12 [25358180, 25362845]      - |             48666,48667    171981
  [2]    chr12 [25368371, 25368494]      - |                   48666    171982
  [3]    chr12 [25378548, 25378707]      - |             48666,48667    171983
  [4]    chr12 [25380168, 25380346]      - |             48666,48667    171984
  [5]    chr12 [25386768, 25388160]      - |                   48668    171987
  [6]    chr12 [25398208, 25398329]      - |       48666,48667,48668    171985
  [7]    chr12 [25403685, 25403854]      - |             48666,48667    171986
  [8]    chr12 [25403698, 25403863]      - |                   48668    171988
                       gene_id
      <CompressedCharacterList>
  [1]                     3845
  [2]                     3845
  [3]                     3845
  [4]                     3845
  [5]                     3845
  [6]                     3845
  [7]                     3845
  [8]                     3845
  ---
  seqlengths:
                chr1                  chr2 ... chr18_gl000207_random
           249250621             243199373 ...                  4262
```

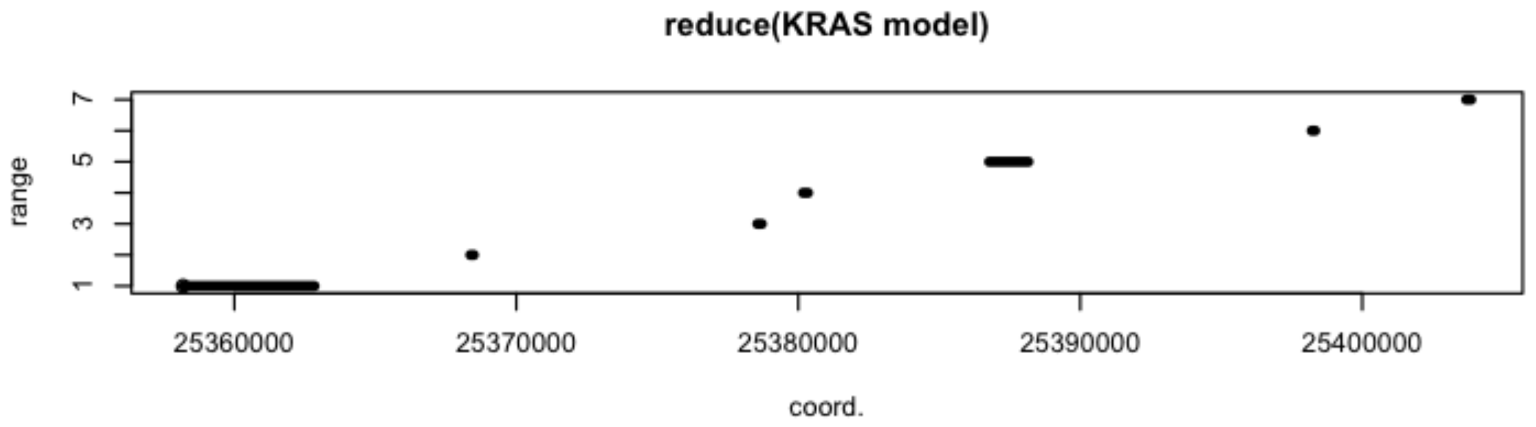


```
> getClass(class(exByG2))
Class "GRanges" [package "GenomicRanges"]

Slots:

Name:          seqnames          ranges          strand elementMetadata
Class:              Rle         IRanges             Rle       DataFrame

Name:           seqinfo        metadata
Class:          Seqinfo            list

Extends:
Class "GenomicRanges", directly
Class "Vector", by class "GenomicRanges", distance 2
Class "GenomicRangesORmissing", by class "GenomicRanges", distance 2
Class "GenomicRangesORGRangesList", by class "GenomicRanges", distance 2
Class "RangedDataORGenomicRanges", by class "GenomicRanges", distance 2
Class "Annotated", by class "GenomicRanges", distance 3
> 
```



```
> unlist(values(exByG2)$tx_id)
 [1] 48666 48667 48666 48666 48667 48666 48667 48668 48666 48667 48668 48666
[13] 48667 48668
> as.list(values(exByG2)$tx_id)
[[1]]
[1] 48666 48667

[[2]]
[1] 48666

[[3]]
[1] 48666 48667

[[4]]
[1] 48666 48667

[[5]]
[1] 48668

[[6]]
[1] 48666 48667 48668

[[7]]
[1] 48666 48667

[[8]]
[1] 48668
```


KRAS model
range
coord.
25360000
25370000
25380000
25390000
25400000
1
3
5
7


reduce(KRAS model)
range
coord.
25360000
25370000
25380000
25390000
25400000
1
3
5
7

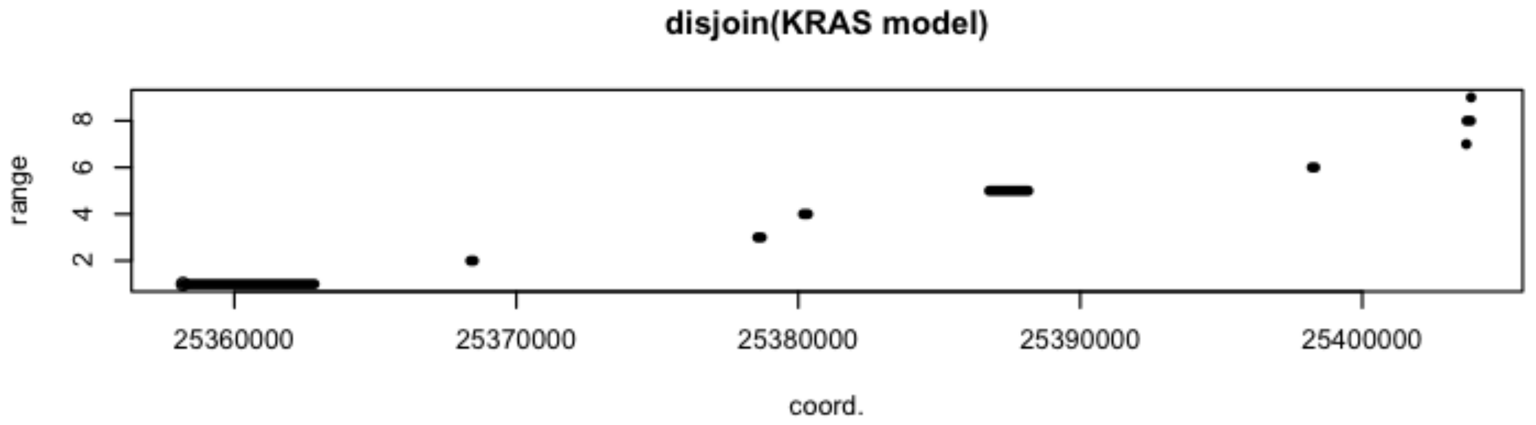

disjoin(KRAS model)
range
coord.
25360000
25370000
25380000
25390000
25400000
2
4
6
8

Table 3: Summary of the API on range-like objects such as *IRanges*, *RangesList*, *GRanges* and *GRangesList*.

Category	Function	Description
Accessors	`start`, `end`, `width`	Get or s et the starts, ends and widths
	`names`	Get or set the names
	`elementMetadata`, `metadata`	Get or set metadata on elements or object
	`length`	Number of ranges in the vector
	`range`	Range formed from min start and max end
Ordering	`<`, `<=`, `>`, `>=`, `==`, `!=`	Compare ranges, ordering by start then width
	`sort`, `order`, `rank`	Sort by the ordering
	`duplicated`	Find ranges with multiple instances
	`unique`	Find unique instances, removing duplicates
Arithmetic	$r + x$, r - x, r * x	Shrink or expand ranges `r` by number `x`
	`shift`	Move the ranges by specified amount
	`resize`	Change width, ancoring on start, end or mid
	`distance`	Separation between ranges (closest endpoints)
	`restrict`	Clamp ranges to within some start and end
	`flank`	Generate adjacent regions on start or end

Set operations	`reduce`	Merge overlapping and adjacent ranges
	`intersect`, `union`, `setdiff`	Set operations on reduced ranges
	`pintersect`, `punion`, `psetdiff`	Parallel set operations, on each *x[i]*, *y[i]*
	`gaps`, `pgap`	Find regions not covered by reduced ranges
	`disjoin`	Ranges formed from union of endpoints
Overlaps	`findOverlaps`	Find all overlaps for each `x` in `y`
	`countOverlaps`	Count overlaps of each `x` range in `y`
	`nearest`	Find nearest neighbors (closest endpoints)
	`precede`, `follow`	Find nearest `y` that `x` precedes or follows
	x %in% y	Find ranges in `x` that overlap range in `y`
Coverage	`coverage`	Count ranges covering each position
Extraction	*r[i]*	Get or set by logical or numeric index
	r[[i]]	Get integer sequence from *start[i]* to *end[i]*
	`subsetByOverlaps`	Subset `x` for those that overlap in `y`
	`head`, `tail`, `rev`, `rep`	Conventional R semantics
Split, combine	`split`	Split ranges by a factor into a *RangesList*
	`c`	Concatenate two or more range objects

Software tools that use IRanges+

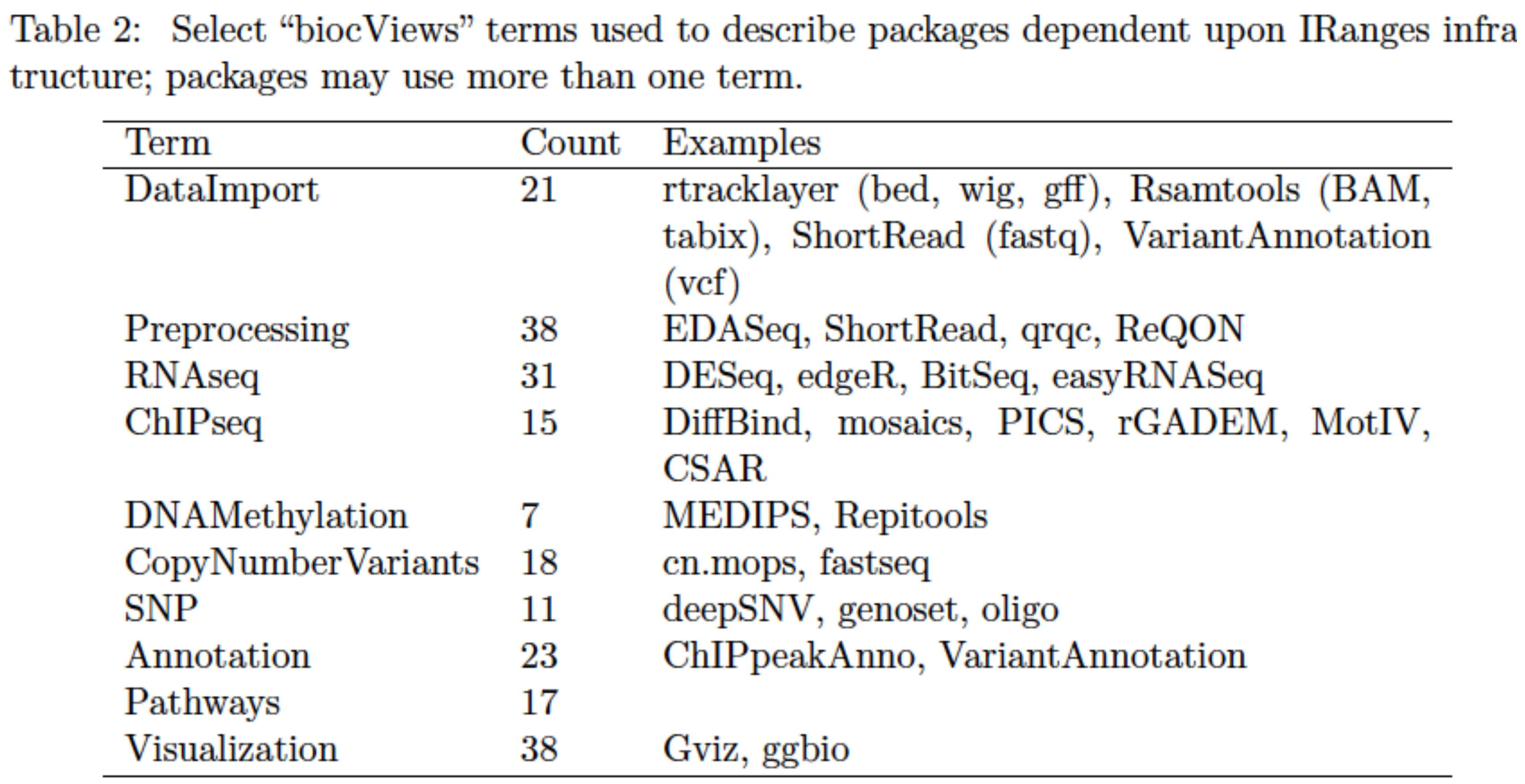
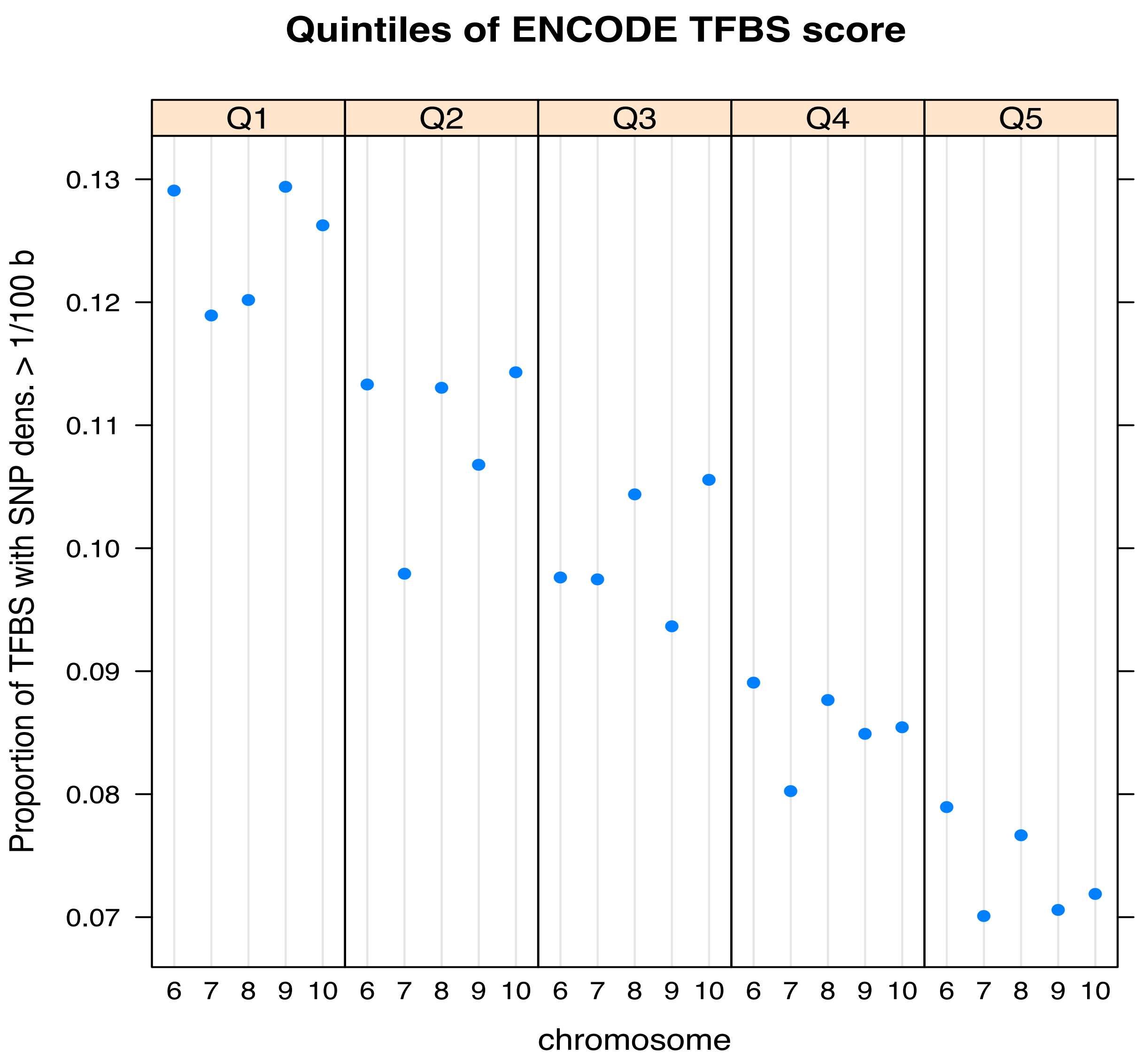
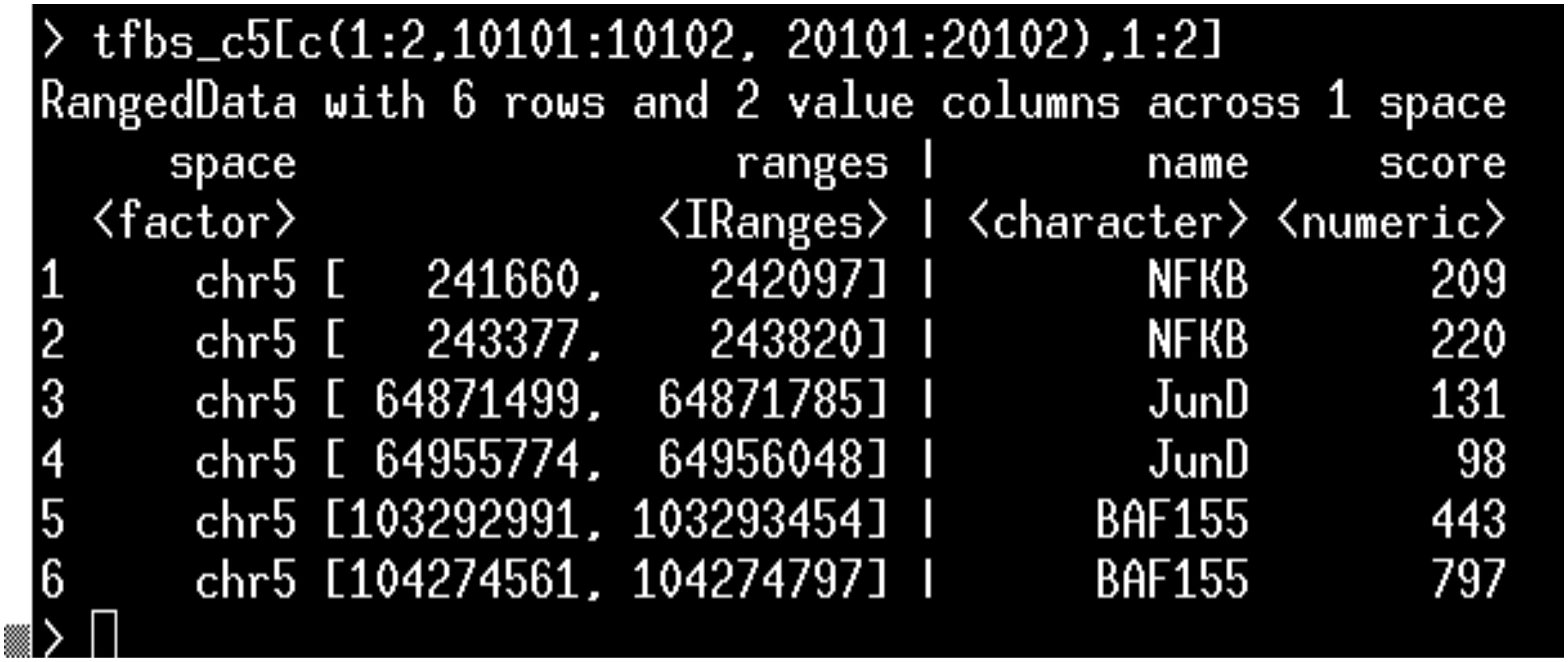


Table 2: Select "biocViews" terms used to describe packages dependent upon IRanges infra tructure; packages may use more than one term.

Term	Count	Examples
DataImport	21	rtracklayer (bed, wig, gff), Rsamtools (BAM, tabix), ShortRead (fastq), VariantAnnotation (vcf)
Preprocessing	38	EDASeq, ShortRead, qrqc, ReQON
RNAseq	31	DESeq, edgeR, BitSeq, easyRNASeq
ChIPseq	15	DiffBind, mosaics, PICS, rGADEM, MotIV, CSAR
DNAMethylation	7	MEDIPS, Repitools
CopyNumberVariants	18	cn.mops, fastseq
SNP	11	deepSNV, genoset, oligo
Annotation	23	ChIPpeakAnno, VariantAnnotation
Pathways	17	
Visualization	38	Gviz, ggbio

Exercise: assess relative frequency of SNP in regions with different TFBS scores





```
> tfbs_c5[c(1:2,10101:10102, 20101:20102),1:2]
RangedData with 6 rows and 2 value columns across 1 space
     space                  ranges |        name     score
  <factor>               <IRanges> | <character> <numeric>
1     chr5 [   241660,    242097] |        NFKB       209
2     chr5 [   243377,    243820] |        NFKB       220
3     chr5 [ 64871499,  64871785] |        JunD       131
4     chr5 [ 64955774,  64956048] |        JunD        98
5     chr5 [103292991, 103293454] |      BAF155       443
6     chr5 [104274561, 104274797] |      BAF155       797
> 
```

```
> ri = reduce(tfbs_c5)    # merge overlapping intervals
> dim(ri)
```

```
[1] 27194     0
```

We will measure the confidence that each of the merged intervals is in fact a TFBS by associating with it the maximum confidence reported for any of its constituent unmerged intervals. To do this, we compute the partition of the original ranges dictated by the merge. This takes several steps. First, the overlaps of merged with original intervals are determined.

```
> ov = findOverlaps(ri, tfbs_c5)
```

The resulting overlap structure is used to create a numeric vector `scores` consisting of the TFBS confidence scores reordered to correspond to intervals as merged in `ri`.

```
> scores <- tfbs_c5$score[subjectHits(ov)] # order as merged
> partitioning <- PartitioningByWidth(as.integer(as.table(ov)))
> scoreViews <- Views(scores, partitioning)
> scoreViews[1:5]

Views on a 86651-double XDouble subject
subject:   72  718   93  346  710 1000 ...  858 1000  187  328   64  500  241
views:
    start end width
[1]     1   3     3 [ 72 718  93]
[2]     4   4     1 [346]
[3]     5   7     3 [ 710 1000  281]
[4]     8   9     2 [215 470]
[5]    10  14     5 [1000  381 1000  188 1000]
```

Finally, the maximum in each of the merged ranges is obtained and loaded back into the merged structure.

```
> ri$maxscores <- viewMaxs(scoreViews)  # obtain max score in merged region
> ri[1:3,]

RangedData with 3 rows and 1 value column across 1 space
     space         ranges | maxscores
  <factor>      <IRanges> | <numeric>
1     chr5 [66079, 66353] |       718
2     chr5 [70326, 70605] |       346
3     chr5 [74647, 74998] |      1000
```

Conclusions

- Efficient programming with entities and features anchored to genomic coordinates is well-supported in Bioconductor
- Underlying infrastructure based on interval trees, RLE, Allen's interval algebra
- Rapid conversion between standard genomic record formats and *Ranges = rtracklayer
- Visualizations: Gviz, ggbio take advantage of the infrastructure