Simple RNA-seq Expression Measures

Patrick Aboyoun

Fred Hutchinson Cancer Research Center

January 29, 2010

Outline

Motivation

RNA-seq is now a standard assay technology for measuring gene expression. This lab will show how to create simple measures of gene expression for RNA-seq experiments.

Expression Measurement Categories

The lab develops two approaches for aggregating alignments in RNA-seq experiments:

- Summarizing coverage values within gene (or transcript) regions.
- Counting the number of alignments that fall in or near gene (or transcript) regions.

For alternative approaches visit `http://bioconductor.org/packages/release/HighThroughputSequencing.html`.

Functions Used in Lab

Sequence Views : `Views`, `viewMax`, `viewMean`

Alignment : `chromosome`, `position`, `strand`, `width`

Interval Ops. : `IRanges`, `resize`, `findOverlaps`, `subjectHits`

Library/File : `library`, `data`

Vector Ops. : `is.na`, `sort`, `table`

Matrix Ops. : `cbind`

Integer Vec. : `L` (e.g. `1L`), `as.integer`, `round`

String Ops. : `paste`, `as.roman`

Logical Ops. : `!`, `==`, `!=`

Object Reshape : `split`, `unlist`

Subscripting : `[`, `[[`, `head`, `tail`

Summary : `mean`, `summary`, `pmin`

Metadata : `levels`, `names`

Data Classes Used in Lab

AlignedRead : imported alignments (verbose)

RleList : genome coverage vectors

RleViewsList : genome coverage vectors combined with intervals of interest, e.g. genes

RangedData : genomic features represented as a data table

RangesList : intervals across a genome

Outline

Loading Saved Work

The previous three labs added alignment, coverage, and gene annotation objects to the *day3* package that we need for this lab.

Smoothed Alignment Coverage

```
> library(day3)
> data(aln)                    # alignments
> data(combSmoothCover)  # coverage
> yeastGenes <- extractYeastGenesAsRangedData()
```

Fixing Chromosome Name Mismatches

Using Roman numerals in chromosome names

```
> head(levels(chromosome(aln)), 4)

[1] "chrI"   "chrII"  "chrIII" "chrIV"

> head(names(combSmoothCover), 4)

[1] "chrI"   "chrII"  "chrIII" "chrIV"

> head(names(yeastGenes), 4)

[1] "1"  "10" "11" "12"

> names(yeastGenes) <-
+   paste("chr",as.roman(names(yeastGenes)),sep="")
> head(names(yeastGenes), 4)

[1] "chrI"   "chrX"   "chrXI"  "chrXII"
```

Reordering the Chromosomes

Coordinating element order in the objects

```
> head(names(combSmoothCover), 4)

[1] "chrI"   "chrII"  "chrIII" "chrIV"

> head(names(yeastGenes), 4)

[1] "chrI"   "chrX"   "chrXI"  "chrXII"

> yeastGenes <- yeastGenes[names(combSmoothCover)]
> head(names(yeastGenes), 4)

[1] "chrI"   "chrII"  "chrIII" "chrIV"

> geneNames <- yeastGenes[["systematic_name"]]
```

Outline

Summarizing Coverage Vectors

- This approach involves summarizing coverage vectors within regions of interest (e.g. genes/transcripts) so each region is assigned 1 number.
- Common statistical summaries are maximum, mean, and sum.

Views on Coverage

Constructing views

```
> geneViews <- Views(combSmoothCover, ranges(yeastGenes))
> geneViews

SimpleRleViewsList of length 16
$chrI
Views on a 230208-length Rle subject

views:
       start    end width
  [1] 151467 151584   118 [0 0 0 0 0 0 0 0 0 0 0 0 0 ...]
  [2]  99306  99869   564 [0 0 0 0 1 1 1 1 1 1 1 1 1 ...]
  [3] 147596 151168  3573 [0 0 0 0 0 0 0 0 0 0 0 0 0 ...]
  [4] 143709 147533  3825 [1 1 1 1 1 1 1 1 1 1 1 1 1 ...]
  [5] 142176 143162   987 [6 6 6 6 6 6 6 7 7 7 7 7 7 ...]
  [6] 139505 141433  1929 [36 36 36 36 36 37 37 37 37 ...]
  [7] 137700 138347   648 [0 0 0 0 0 0 0 0 0 0 0 0 0 ...]
  [8] 136916 137512   597 [0 0 0 0 0 0 0 0 0 0 0 0 0 ...]
```

Maximum Coverage Within Genes

viewMaxs

```
> maxCover <- viewMaxs(geneViews)
> maxCover <- unlist(maxCover, use.names=FALSE)
> names(maxCover) <- geneNames
> tail(sort(maxCover), 4)

RDN25-1 RDN37-1 RDN25-2 RDN37-2
   8200    8200    8230    8230

> mean(maxCover == 0)

[1] 0.18

> summary(maxCover)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      0       1       2      14       3    8230
```

Mean Coverage Within Genes

viewMeans

```
> meanCover <- round(viewMeans(geneViews))
> meanCover <- unlist(meanCover, use.names=FALSE)
> names(meanCover) <- geneNames
> tail(sort(meanCover), 4)

YLR154C-G   RDN25-2   RDN25-1 YLR154W-A
     4373      4474      4485      5965

> summary(meanCover)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0     0.0     0.0     6.6     1.0  5960.0
```

Outline

Counting Alignments That Overlap Features

- This approach involves creating a tally of the number of alignments that overlap each genomic feature of interest.
- As with the coverage calculations we will perform this tally on each strand separately and then reconcile the differences.

Generate Alignment Ranges

Once again we will extend the alignments to a fixed fragment length of 150 bp.

Construction of stranded alignments

```
> posStr <- strand(aln) == "+"
> alnRanges <- IRanges(position(aln), width = width(aln))
> posRanges <- split(alnRanges[posStr],
+                    chromosome(aln)[posStr])
> posRanges <- resize(posRanges, width = 150L)
> negRanges <- split(alnRanges[!posStr],
+                    chromosome(aln)[!posStr])
> negRanges <- resize(negRanges, width = 150L, start=FALSE)
```

Positive Strand Alignment Overlaps

Count along the positive strand

```
> posCounts <-
+   table(subjectHits(findOverlaps(posRanges, yeastGenes)))
> i <- as.integer(names(posCounts))
> names(posCounts) <- geneNames[i]
> posCounts <- posCounts[geneNames]
> names(posCounts) <- geneNames
> posCounts[is.na(posCounts)] <- 0L
```

Negative Strand Alignment Overlaps

Count along the negative strand

```
> negCounts <-
+   table(subjectHits(findOverlaps(negRanges, yeastGenes)))
> i <- as.integer(names(negCounts))
> names(negCounts) <- geneNames[i]
> negCounts <- negCounts[geneNames]
> names(negCounts) <- geneNames
> negCounts[is.na(negCounts)] <- 0L
```

Parallel Minimum Combined Overlaps

Creating the combined overlaps

```
> combOverlaps <- cbind(pos = posCounts, neg = negCounts)
> head(combOverlaps, 2)

     pos neg
CEN1   0   0
HRA1   7   7

> overlapCounts <- pmin(combOverlaps[,1], combOverlaps[,2])
> tail(sort(overlapCounts), 4)

RDN25-2 RDN25-1 RDN37-2 RDN37-1
 106369  107531  118972  120266

> summary(overlapCounts)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      0       2       6      88      15  120000
```

Exercises

1. Get the gene names for the top 50 largest in each of the three measures (gene maximums, gene averages, overlap counts).
2. How many of the genes are in all three top 50 lists?

Answers

```
> topMaxs <-
+   head(names(sort(maxCover, decreasing=TRUE)), 50)
> topMeans <-
+   head(names(sort(meanCover, decreasing=TRUE)), 50)
> topOverlaps <-
+   head(names(sort(overlapCounts, decreasing=TRUE)), 50)
> length(intersect(topMaxs, intersect(topMeans, topOverlaps)))
```

```
[1] 29
```

Outline

Session Information

- R version 2.10.1 Patched (2010-01-28 r51060), `x86_64-unknown-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=en_US.UTF-8`, `LC_MONETARY=C`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Base packages: base, datasets, graphics, grDevices, methods, stats, tools, utils
- Other packages: AnnotationDbi 1.8.1, Biobase 2.6.1, biomaRt 2.2.0, Biostrings 2.14.8, bitops 1.0-4.1, BSgenome 1.14.2, BSgenome.Scerevisiae.UCSC.sacCer2 1.3.16, day3 0.0.3, DBI 0.2-4, IRanges 1.4.9, lattice 0.17-26, org.Sc.sgd.db 2.3.5, RCurl 1.3-0, RSQLite 0.7-3, rtracklayer 1.6.0, ShortRead 1.4.0
- Loaded via a namespace (and not attached): grid 2.10.1, [illegible]