Exploratory Data Analysis of Coverage

Patrick Aboyoun

Fred Hutchinson Cancer Research Center

January 29, 2010

Outline

Motivation

Alignment coverage is commonly used as a proxy for the prevalence of ChIP pull-downs (ChIP-seq) or cDNA production (RNA-seq) on a reference genome. This lab will demonstrate how to examine coverage prior to a formal analysis.

Experiment Description

Experiment README file

```
> file.show(system.file("extdata", "README.txt",
+                       package = "day3"))
```

```
BYe9.head.map contains aligned reads from an RNA-seq

This file gives some details on how BYe9.head.map was

Experimental data
-----------------

- Type of experiment: RNA-seq

- Organism: Yeast
```

Hardest Concept in Lab

Number puzzle

```
> 1 == 1L

[1] TRUE

> identical(1, 1L)

[1] FALSE
```

Why? Extra credit if answer in the form of a haiku.

Functions Used in Lab

Library/File	: `library`, `data`, `search`, `system.file`, `file.show`
Vector Ops.	: `as.vector`, `c`, `length`
Integer Vec.	: `:` (e.g. `1:10`), `L` (e.g. `1L`), `-`, `round`
Logical Ops.	: `!`, `==`, `!=`
Object Ops.	: `class`, `identical`
Subscripting	: `[`, `[[`, `head`, `window`
Summary	: `max`, `pmin`
Metadata	: `levels`, `names`
Alignment	: `chromosome`, `strand`, `width`
Coverage	: `coverage`
Smoothing	: `runmean`
Looping	: `mendoapply`
Plotting	: `plot`, `polygon`

Flash Quiz

How can I find out if there is a function similar to `head` that displays the last part of an object?

Run-Length Encoding (RLE)

- Chromosomes can be hundreds of million of base pairs long, making them hard to manage in computer memory.
- Fortunately, coverage vectors tend to follow an integer step function.
- Run-length encoding (RLE) is a common compression technique for storing long sequences with lengthy repeats.
- An RLE couples values with run lengths, e.g. the vector 0, 0, 0, 1, 1, 2 would be represented as (3) 0's, (2) 1's, and (1) 2.
- The *IRanges* package uses the *Rle* and *RleList* classes to house coverage vectors.

Outline

Loading Packages

The *ShortRead* package contains the *AlignedRead* class definition. It will also load the *IRanges* package that we will use.

ShortRead package

```
> library(ShortRead)

> head(search())

[1] ".GlobalEnv"          "package:rtracklayer"
[3] "package:RCurl"       "package:bitops"
[5] "package:ShortRead"   "package:lattice"
```

Loading Alignments

The *day3* package has a saved version of an *AlignedRead* object containing the yeast RNA-seq alignments that was created during the *ShortRead* lab.

Loading a saved AlignedRead object

```
> library(day3)
> data(aln)
```

Chromosome Lengths (Annotation Teaser)

Coverage computation requires chromosome lengths, which are not included in the alignment object.

Yeast genome package

```
> library(org.Sc.sgd.db)
> chrlen <- org.Sc.sgdCHRLENGTHS
> head(names(chrlen), 4)

[1] "chrIV"  "chrXV"  "chrVII" "chrXII"

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

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

Outline

Coverage on Both Strands

In this RNA-seq experiment, the cDNA were sonicated to fragments of length 100 - 200 bp. To simplify the analysis, we will use a fixed fragment length of 150 bp in the coverage calculation.

Create coverage on each strand

```
> posStr <- strand(aln) == "+"
> posCover <- coverage(aln[posStr], width = chrlen,
+                      extend = 150L - width(aln)[posStr])
> negCover <- coverage(aln[!posStr], width = chrlen,
+                      extend = 150L - width(aln)[!posStr])
```

Genome Browsing in Bioconductor

- R has no standard graphics device that is ideally suited for genome browsing.
- The *rtracklayer* package in Bioconductor serves as a bridge to external genome browsers (e.g. UCSC).
- In this lab we will use functions in the *day3* package to create simple graphics in R.

Plotting Coverage

The *day3* package contains a function to plot coverage vectors. It is defined as:

A day3 package coverage plot function

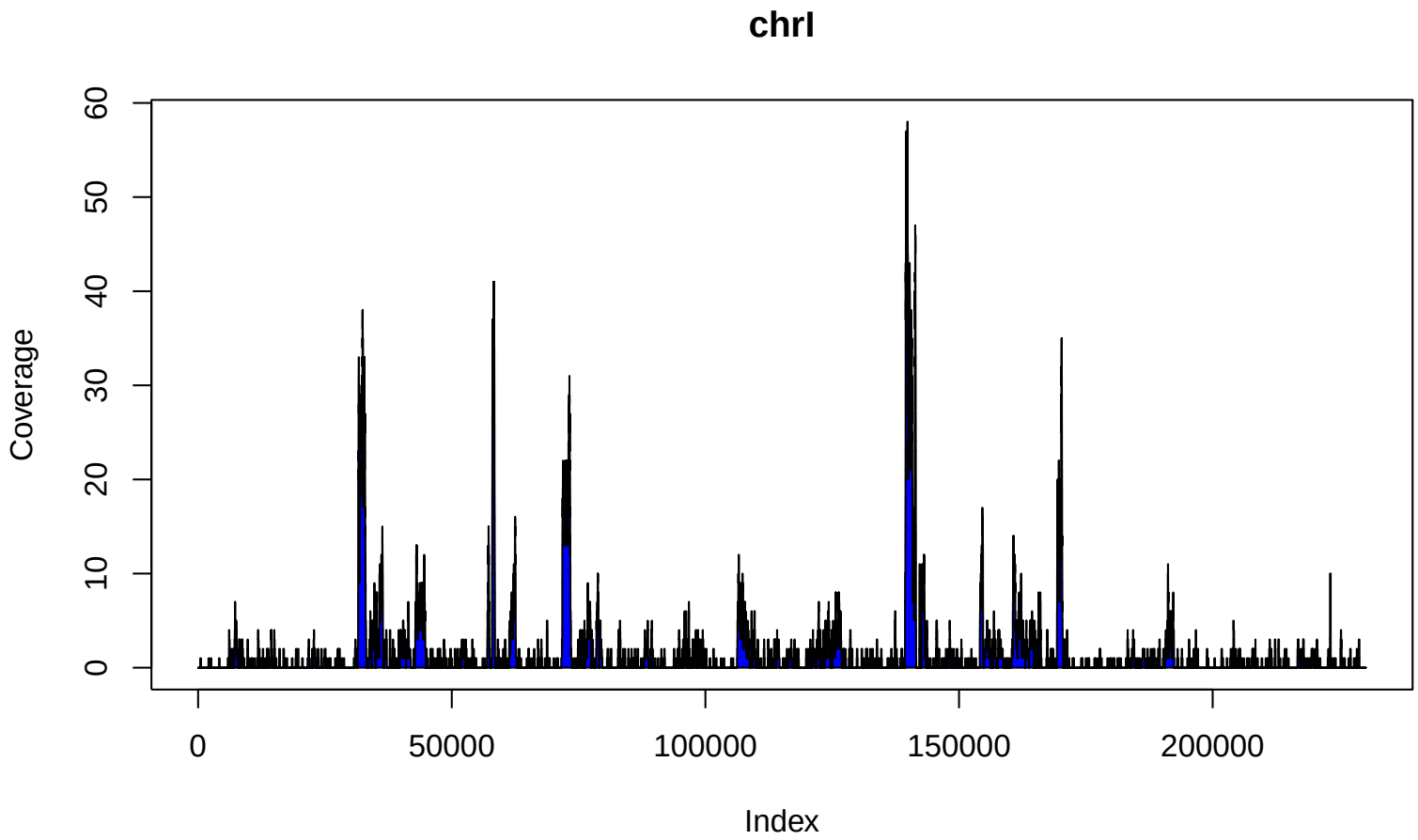
```
> library(day3)
> plotCoverage

function (x, chrom, start = 1, end = length(x[[chrom]]), col = "blue",
    xlab = "Index", ylab = "Coverage", main = chrom)
{
    xWindow <- as.vector(window(x[[chrom]], start, end))
    x <- start:end
    xlim <- c(start, end)
    ylim <- c(0, max(xWindow))
    plot(x = start, y = 0, xlim = xlim, ylim = ylim, xlab = xlab,
        ylab = ylab, main = main, type = "n")
    polygon(c(start, x, end), c(0, xWindow, 0), col = col)
}
```

Plotting Coverage on One Strand

Plotting chr1+ coverage

```
> plotCoverage(posCover, "chrI")
```



Plotting Stranded Coverage

The *day3* package also contains a function to plot coverage vectors for both strands of a chromosome. It is defined as:

A day3 package stranded coverage plot function

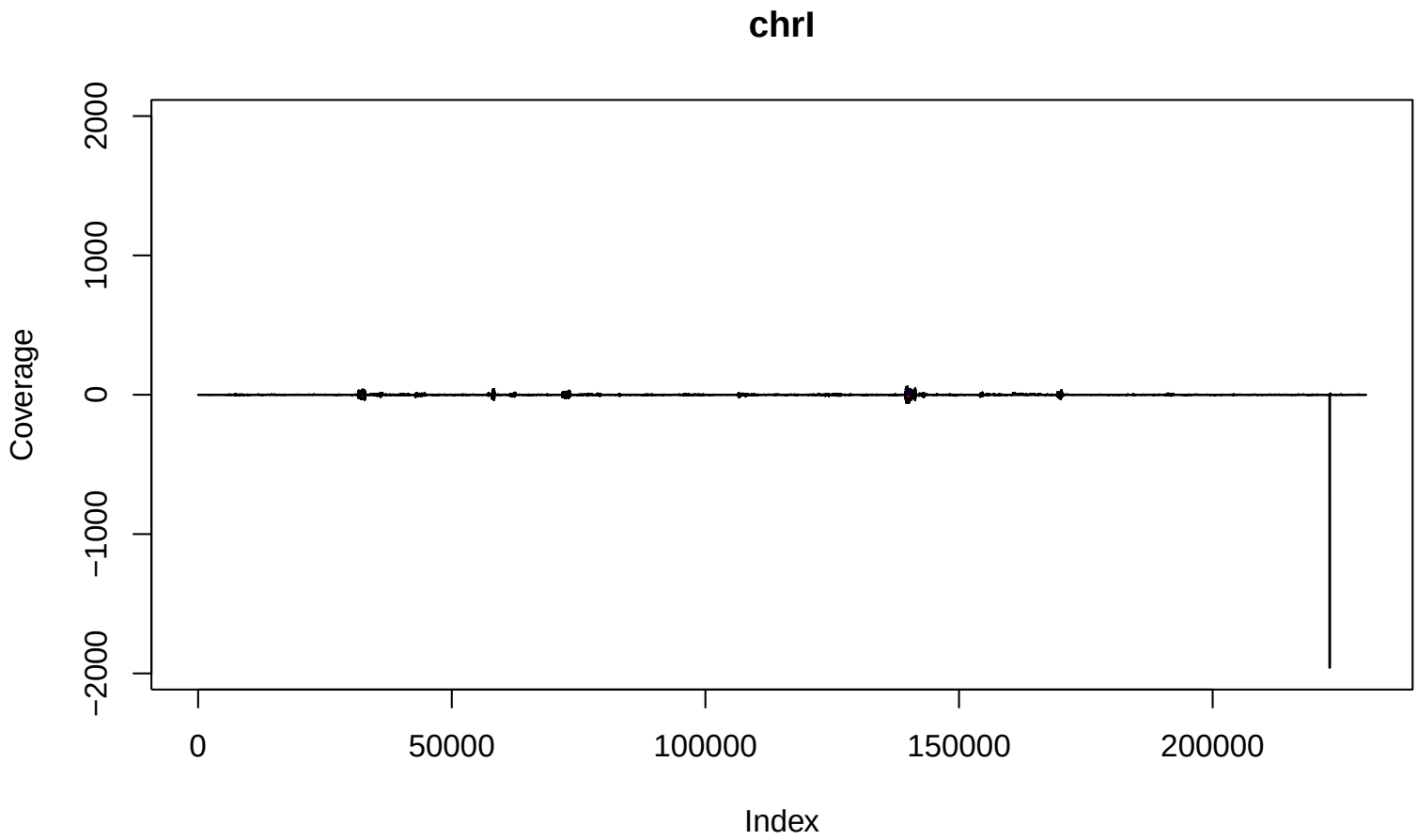
```
> plotCoverageStrands

function (pos, neg, chrom, start = 1, end = length(pos[[chrom]]),
    pos.col = "blue", neg.col = "red", xlab = "Index", ylab = "Coverage",
    main = chrom)
{
    posWindow <- as.vector(window(pos[[chrom]], start, end))
    negWindow <- as.vector(window(neg[[chrom]], start, end))
    x <- start:end
    xlim <- c(start, end)
    ylim <- c(-1, 1) * max(posWindow, negWindow)
    plot(x = start, y = 0, xlim = xlim, ylim = ylim, xlab = xlab,
        ylab = ylab, main = main, type = "n")
    polygon(c(start, x, end), c(0, posWindow, 0), col = pos.col)
    polygon(c(start, x, end), c(0, -negWindow, 0), col = neg.col)
}
```

Plotting Coverage on Both Strands

Plotting chr1 coverage, both strands

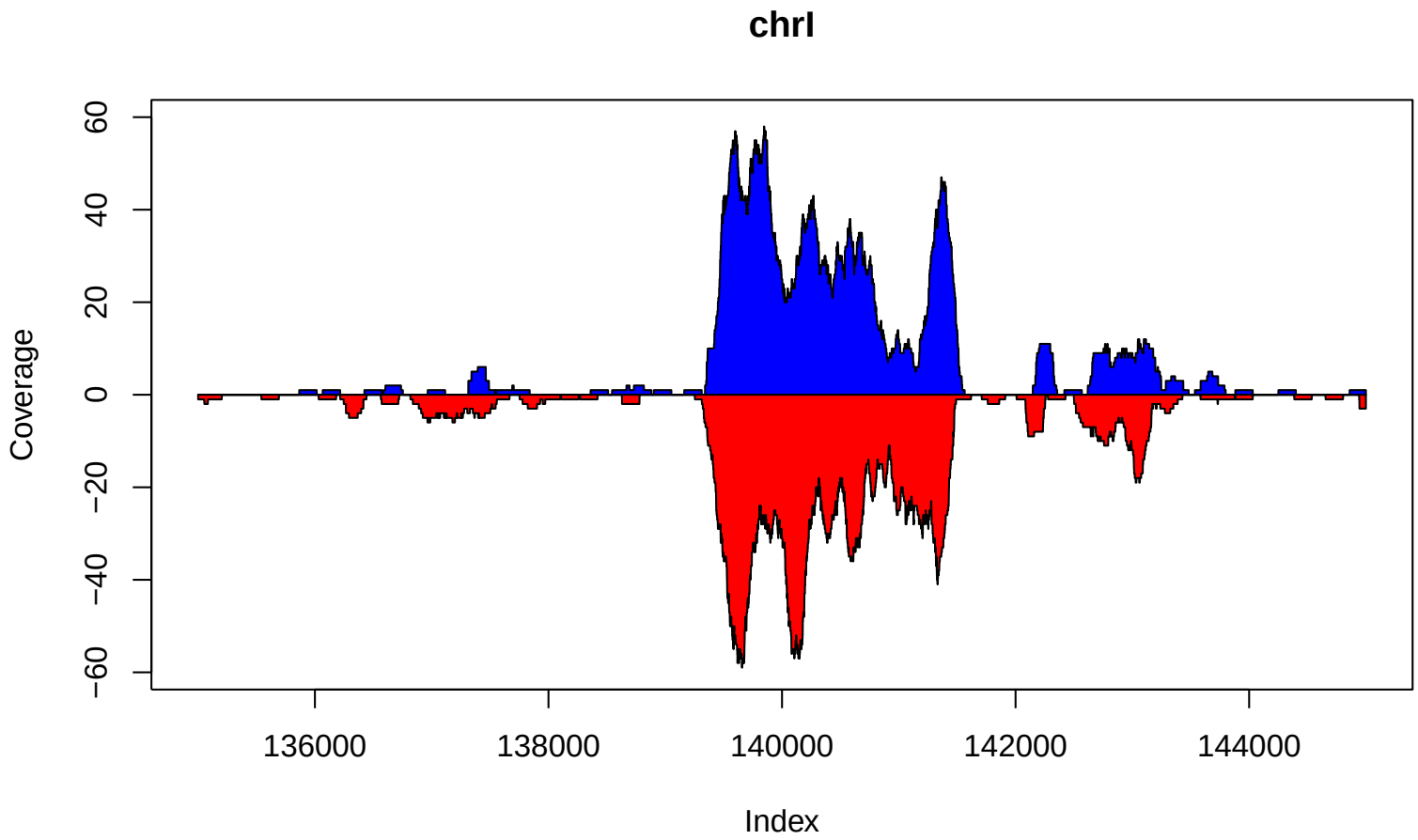
```
> plotCoverageStrands(posCover, negCover, "chrI")
```



Plotting Coverage on Both Strands

Plotting chr1 coverage, both strands

```
> plotCoverageStrands(posCover, negCover, "chrI", 135000, 145000)
```



Exercise

- Use `plotCoverage` and `plotCoverageStrands` to explore the RNA-seq alignment coverage across the yeast genome.
- Record the locations with interesting coverage characteristics so you can examine them again after we perform some analyses in the afternoon.

Outline

Removing Coverage Noise

- As you have discovered, these coverage vectors are very noisy.
- The noise can be removed using running window smoothers.
- A running window smoother uses a fixed length "window" that slides across the chromosome and replaces the value at the center of the window with a statistic calculated using the values within the window.
- We will perform a running window mean where the window is half the length of the estimated fragment length, i.e. 75.
- Since these means are real valued, we will round the results to the nearest integer.

Smoothing Coverage

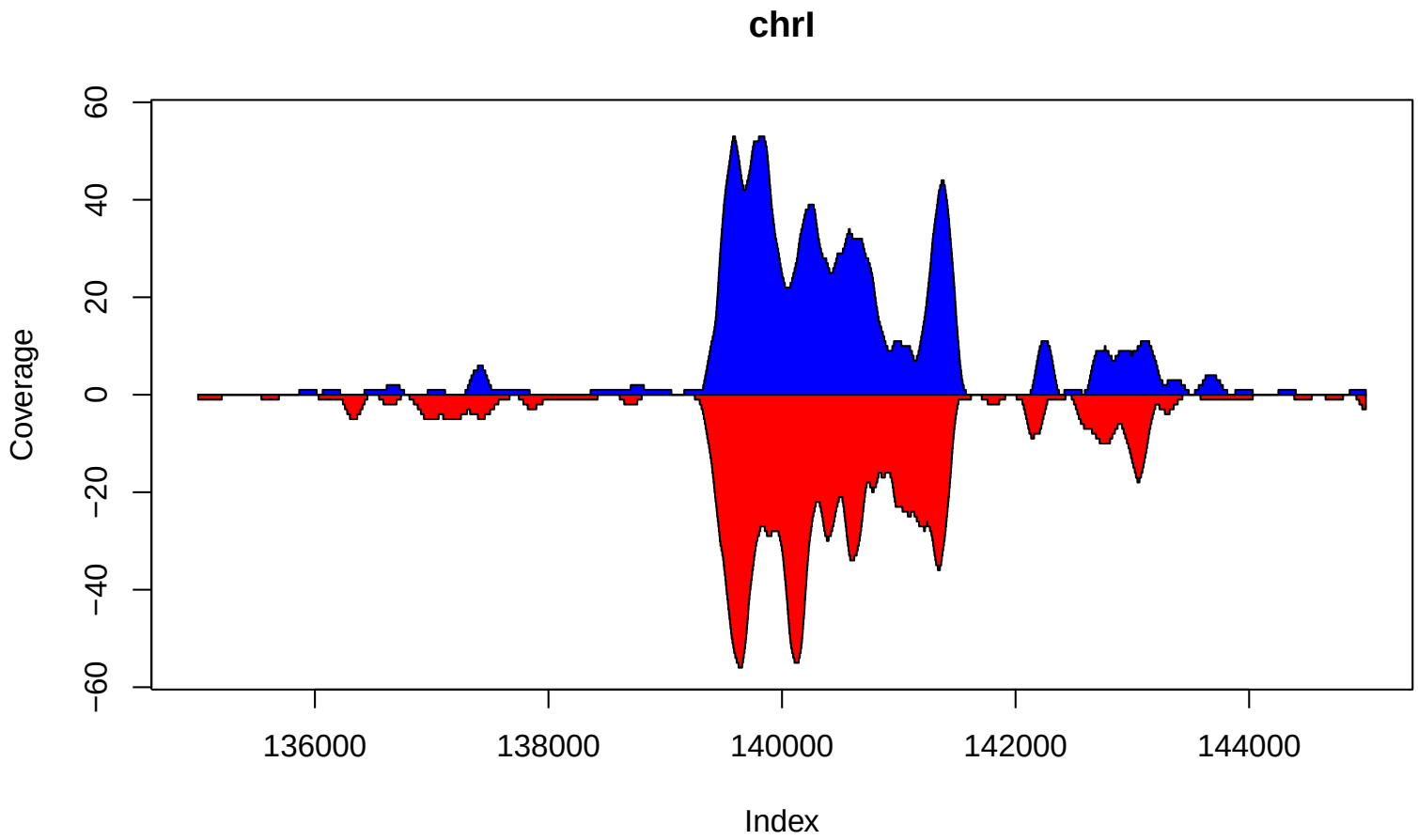
Running window mean

```
> posSmoothCover <- runmean(posCover, 75,
+                           endrule = "constant")
> posSmoothCover <- round(posSmoothCover)
> negSmoothCover <- runmean(negCover, 75,
+                           endrule = "constant")
> negSmoothCover <- round(negSmoothCover)
```

Plotting Coverage on Smoothed Strands

Plotting chr1, smoothed strands

```
> plotCoverageStrands(posSmoothCover,negSmoothCover,"chrI",135000,145000)
```



Combining Coverage

- Now that we have less noisy coverage vectors for the strands, we can combine them to determine the "hot spots".
- A conservative measure is to use the minimum coverage value on either strand at each position on the genome.
- This can be computed using the `pmin`.

Parallel Minimums

- The `mendoapply` function is a convenient looping function in the `apply` family.
- It performs elementwise operations across multiple inputs of the same type.
- It returns an object of the same type as the inputs.

Combine using "parallel" minimums

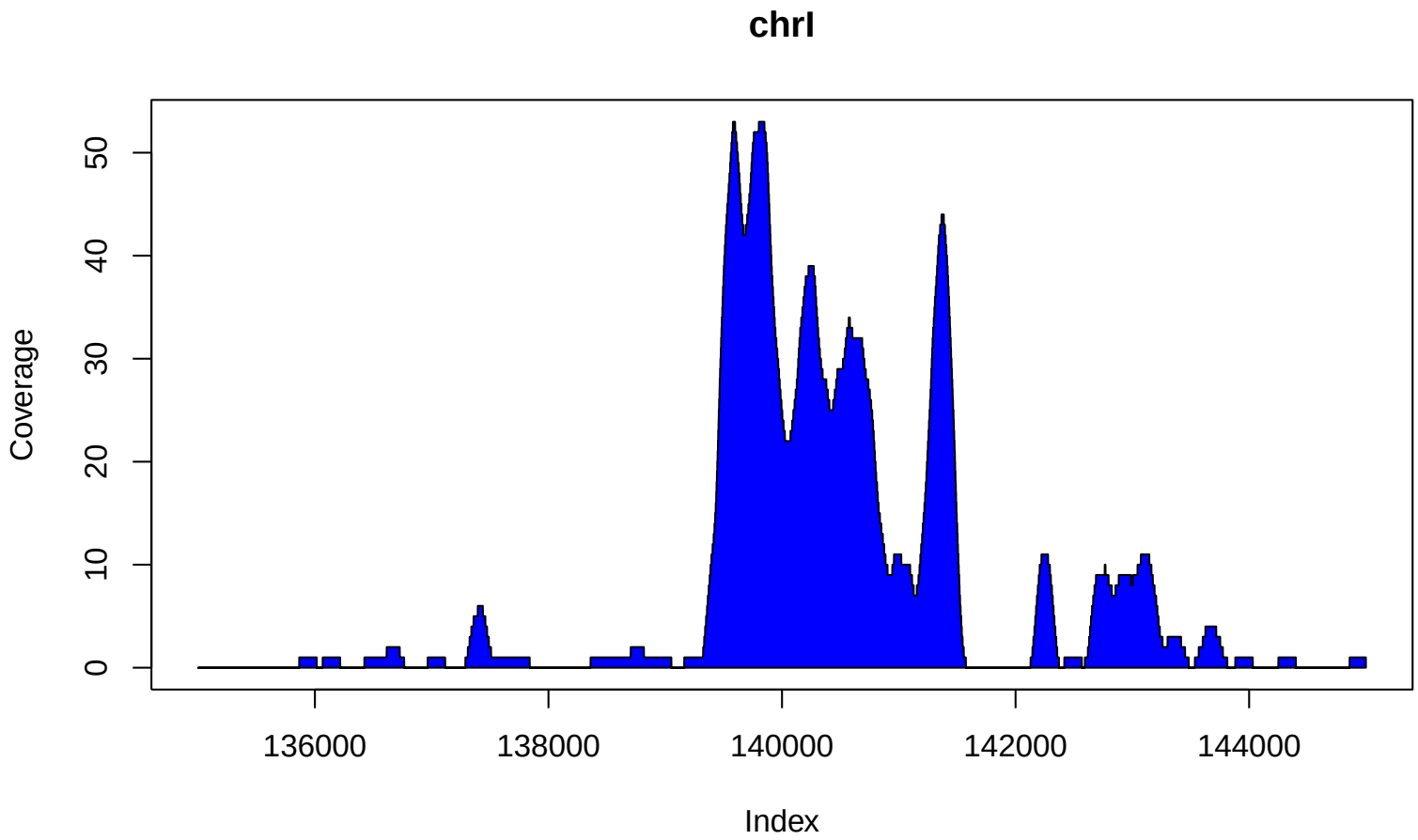
```
> combSmoothCover <- mendoapply(pmin,
+                               posSmoothCover,
+                               negSmoothCover)
> identical(class(posSmoothCover), class(combSmoothCover))

[1] TRUE
```

Plotting Combined Coverage

Plotting chr1, combined strands

```
> plotCoverage(posSmoothCover, "chrI", 135000, 145000)
```



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]