# Introduction to GenomicFiles

Valerie Obenchain, Michael Love, Martin Morgan

Last modified: April 2014; Compiled: April 15, 2014

## Contents

1 Introduction 1

2 Constructing a *FileViews class 1

3 Division of labor 2
- 3.1 reduceByRange . . . 2
- 3.2 reduceByFile . . . 2

4 Summary statistics with bam2R 3
- 4.1 reduceByRange . . . 3
- 4.2 reduceByFile . . . 4

5 coverage and summary for BigWigFileViews 5

6 Case / control example (file grouping) 6

7 sessionInfo() 7

## 1 Introduction

This vignette illustrates how to use the *GenomicFiles* package for distributed computations across files. The package introduces the `GenomicFileViews` class infrastructure for defining and executing genomic queries. `reduceByFile` and `reduceByRange` functions distribute computations in parallel by file or by range. MAP and REDUCE functions, similar in spirit to `Map` and `Reduce` in *base R*, provide a flexible interface to manipulate and combine data across workers.

We assume the reader has some previous experience with *R* and with basic manipulation of ranges objects such as `GRanges` and `GAlignments` and file classes such as `BamFile` and `BigWigFile`. See the vignettes and documentation in *GenomicRanges*, *GenomicAlignments*, *Rsamtools* and *rtracklayer* for an introduction to these classes.

The *GenomicFiles* package is available at bioconductor.org and can be downloaded via `biocLite`:

```
> source("http://bioconductor.org/biocLite.R")
> biocLite("GenomicFiles")
```

## 2 Constructing a *FileViews class

```
> library(GenomicFiles)
```

Each file type (e.g., Bam, BigWig, Fasta) has a corresponding *FileViews class. Here we construct a *BamFileViews* with Bam files from a transcription profiling experiment available in the *RNAseqData.HNRNPC.bam.chr14* package.

```
> library(RNAseqData.HNRNPC.bam.chr14)
> fls <- RNAseqData.HNRNPC.bam.chr14_BAMFILES
> bfv <- BamFileViews(fls)
> bfv
```

```
BamFileViews dim: 0 ranges x 8 samples
names: ERR127306_chr14.bam ERR127307_chr14.bam ... ERR127304_chr14.bam ERR127305_chr14.bam
detail: use fileList(), fileSample(), fileRange(), ...
```

The class can hold files, ranges, a list for experimental data and a yield size parameter.

```
> getSlots("BamFileViews")
```

```
      fileList     fileSample      fileRange fileExperiment      yieldSize .views_on_file
        "List"    "DataFrame"      "GRanges"         "list"      "integer"  "environment"
```

Data are added to slots with accessors of the same name.

```
> fileRange(bfv) <- GRanges("chr14", IRanges(c(1.9e7, 2e7), width = 900000))
```

# 3 Division of labor

Computations with *GenomicFiles* are distributed in parallel by file or by range. The files in `fileList` or ranges in `fileRange` are sent to workers along with functions to perform map and reduce steps. The map step directives are specified in a `MAP` function and the reduce directives in a `REDUCE` function.

The map and reduce steps used by *GenomicFiles* follow the list-processing combinators from functional programming (i.e., `Map` and `Reduce` in *base R*). We use the all-caps designation of `MAP` and `REDUCE` to represent the functions in *GenomicFiles*.

Note that the map and reduce directives both occur within the distributed step, which is slightly different than the typical map-reduce framework, where the map directive occurs in parallel, while the reduce step gathers and summarizes the results from the distributed workers. In *GenomicFileview*, if the computations are distributed in parallel over ranges, for a given range, the map function will apply to each file and the reduce function will combine results across files *within that range*. If the computations are distributed in parallel over files, then for a given file, the map function will apply to each range and the reduce function will combine results across the ranges *within that file*.

## 3.1 reduceByRange

Returns a list the same length as the number of ranges.

- ranges in `fileRange` are sent to workers
- `MAP` function is applied all files, single range
- `REDUCE` function is applied to the output of `MAP`

This approach is best suited for applications that need the same range of data from many files together for an operation.

## 3.2 reduceByFile

Returns a list the same length as the number of files.

- files in `fileList` are sent to workers
- `MAP` function is applied to all ranges, single file

- REDUCE function is applied to the output of MAP

This approach is best suited for applications that need ranges from the same file together for an operation.

# 4 Summary statistics with bam2R

In this example we compute statistics across Bam files and summarize by file and by range. This simple use case introduces the use of MAP and REDUCE in the conjunction with reduceByFile and reduceByRange.

We use bam2R from the *deepSNV* package to compute the statistics.

## 4.1 reduceByRange

The motivating example is to summarize nucleotide counts (pileups) by range for a group of files.

Create a *BamFileViews* object:

```
> ranges1 <- GRanges("chr14", IRanges(c(19411677, 19659063, 105421963,
+                    105613740), width = 20))
> bfv1 <- BamFileViews(fls, fileRange = ranges1)
```

The map step invokes bam2R and retains only the nucleotide counts (see ?bam2R for other output fields). Counts from the reference strand are uppercase and counts from the complement are lowercase.

```
> MAP1 <- function(FILE, RANGE, ...) {
+     require(deepSNV)
+     ct <- bam2R(path(FILE), seqlevels(RANGE), start(RANGE), end(RANGE), q=0)
+     ct[, c("A", "T", "C", "G", "a", "t", "c", "g")]
+ }
```

It is necessary to require(deepSNV) to allow the MAP1 function to be run on nodes or in processes that have not yet loaded the package. The reduce step sums the counts by position.

```
> REDUCE1 <- function(MAPPED, ...) {
+     Reduce("+", MAPPED)
+ }
```

reduceByRange returns a list the same length as the number of ranges.

```
> res1 <- reduceByRange(bfv1, MAP1, REDUCE1)
> length(res1)
```

```
[1] 4
```

Each element is a matrix of counts (position by nucleotide) summed over the 8 files.

```
> res1[[1]]
```

```
      A  T  C  G  a  t  c  g
 [1,] 14  0  0  0 21  0  0  0
 [2,] 16  0  0  0 21  0  0  0
 [3,] 16  0  0  0 20  0  0  0
 [4,]  0  0  0 16  0  0  0 20
 [5,]  0  0 20  0  0  0 19  0
 [6,] 19  0  0  0 19  0  0  0
 [7,]  0  0  0 19  0  0  0 19
 [8,]  0 19  0  0  0 18  0  0
 [9,]  0 19  0  0  0 18  0  0
```

```
[10,]  0  0 19  0  0  0 18  0
[11,]  0 18  0  0  0 18  0  0
[12,]  0 18  0  0  0 18  0  0
[13,]  0 18  0  0  0 17  0  0
[14,] 18  0  0  0 15  0  0  0
[15,]  0  0 18  0  0  0 14  0
[16,]  0 18  0  0  0 14  0  0
[17,] 18  0  0  0 13  0  0  0
[18,] 17  0  0  0 13  0  0  0
[19,]  0 17  0  0  0 12  0  0
[20,]  0 17  0  0  0 12  0  0
```

## 4.2 reduceByFile

To demonstrate a 'by file' operation we look for the presence of insertions / deletions in a group of ranges in each file.

Create a new object with expanded ranges to cover more interesting regions containing insertions and deletions:

```
> ranges2 <- GRanges("chr14",
+                 IRanges(c(19411677, 19659063, 105421963, 105613740),
+                         c(19411747, 19659135, 105422990, 105614142)))
> bfv2 <- BamFileViews(fls, fileRange = ranges2)
```

The map step computes the number of insertions and deletions.

```
> MAP2 <- function(FILE, RANGE, ...) {
+     require(deepSNV)
+     ct <- bam2R(path(FILE), seqlevels(RANGE), start(RANGE), end(RANGE), q=0)
+     ct[, c("INS", "DEL", "ins", "del")]
+ }
```

The reducer sums counts across all ranges in a single file.

```
> REDUCE2 <- function(MAPPED, ...) {
+     sapply(MAPPED, colSums)
+ }
```

reduceByFile computes the number of insertions and deletions present in all ranges for each file. The return value is a list the same length as the number of files.

```
> res2 <- reduceByFile(bfv2, MAP2, REDUCE2)
> length(res2)
```

```
[1] 8
```

The first element of 'res2' contains insertion / deletion counts for the first file summarized across all ranges.

```
> res2[[1]]
```

```
    1 2 3 4
INS 0 0 0 0
DEL 0 1 0 0
ins 1 0 0 1
del 0 0 1 0
```

## 5 coverage and summary for BigWigFileViews

The coverage and summary methods defined for *BigWigFileViews* objects use functions from *rtracklayer* to efficiently query a *BigWigFile*. Internally, these functions are fairly simple calls using map and reduce directives, as described in the previous sections. Here we will use a BigWig and build a *BigWigFileViews* object over 6 ranges and pointing to the same file twice to test iteration over multiple files.

```
> fl <- system.file("tests", "test.bw", package = "rtracklayer")
> gr <- GRanges(Rle(c("chr2", "chr19"), c(4, 2)),
+               IRanges(1 + c(200, 250, 500, 550, 1450, 1750), width=100))
> bwfv <- BigWigFileViews(c(fl, fl), fileRange=gr)
```

The coverage method imports the BigWig coverage over the given ranges as an *RleList* and, if the argument summarize=TRUE (the default), packages these *RleList* objects into a *SummarizedExperiment* object, which is the same dimension as the *BigWigFileViews*. The individual elements of the *matrix* returned by assay are *RleList* objects of the coverage of a given range for a given file.

```
> covSE <- coverage(bwfv)
```

```
> covSE
class: SummarizedExperiment
dim: 6 2
exptData(0):
assays(1): ''
rownames: NULL
rowData metadata column names(0):
colnames: NULL
colData names(1): filePath

> assay(covSE)[6,1]
[[1]]
RleList of length 1
$chr19
numeric-Rle of length 100 with 2 runs
  Lengths:   50   50
  Values : 0.25  0.5
```

Likewise, the summary function produces a *SummarizedExperiment* by default, which contains the specific summary statistic as queried by the summary method for *BigWigFile* objects in the *rtracklayer* packages. These are specified by the type argument which is passed through. Supported types are described in the manual page for *BigWigFile*.

```
> sumSE <- summary(bwfv, type="mean")
```

```
> sumSE
class: SummarizedExperiment
dim: 6 2
exptData(0):
assays(1): ''
rownames: NULL
rowData metadata column names(0):
colnames: NULL
colData names(1): filePath

> assay(sumSE)
       [,1]   [,2]
[1,] -1.000 -1.000
[2,] -0.875 -0.875
```

```
[3,] -0.750 -0.750
[4,] -0.625 -0.625
[5,]  0.250  0.250
[6,]  0.375  0.375
```

# 6 Case / control example (file grouping)

In Section 4.1, an example of streaming along the genome was shown in which the nucleotide counts for a range were summed over a series of files. In this section we show a slightly more complicated example of using *GenomicFiles* to stream along the genome and process a number of files, here passing along a variable which specifies which files belong to which experimental group. This allows for operations such as a basepair-level $t$-test on coverage.

First we define arbitrary groups for the files in `bfv1`:

```
> grp <- factor(rep(c("A","B"), each=ncol(bfv1)/2))
```

The map function will read in alignments from each BAM file and convert into a numeric vector of coverage. Note: this is not the most efficient way to import coverage, however the code is just for demontration of the principle of grouping files.

```
> MAP <- function(RANGE, FILE, ...) {
+     require(GenomicAlignments)
+     stopifnot(length(RANGE) == 1)
+     param <- ScanBamParam(which=RANGE)
+     algns <- readGAlignments(path(FILE), param=param)
+     as.numeric(coverage(algns)[RANGE][[1]])
+ }
```

The reduce function then combines the numeric coverage vector into a matrix, identifies those rows which have all zeros, and then performs row-wise $t$-testing using the *genefilter* package. The index of which rows correspond to which basepair of the original range is stored as a column `offset`.

```
> REDUCE <- function(MAPPED, ..., grp) {
+     m <- simplify2array(MAPPED)
+     idx <- which(rowSums(m) != 0)
+     df <- genefilter::rowttests(m[idx,], grp)
+     cbind(offset=idx - 1, df)
+ }
```

The call to `reduceByRange` produces a list as long as the number of ranges of the *GenomicFileViews* object. We can assign this result as a metadata column on the `fileRange`. Each element is then a table of basepair-level $t$-test results.

```
> fileRange(bfv1)$result <- reduceByRange(bfv1, MAP, REDUCE, grp=grp)
> fileRange(bfv1)
```

```
GRanges with 4 ranges and 1 metadata column:
      seqnames               ranges strand |   result
         <Rle>            <IRanges>  <Rle> |   <list>
  [1]    chr14 [ 19411677,  19411696]      * | ########
  [2]    chr14 [ 19659063,  19659082]      * | ########
  [3]    chr14 [105421963, 105421982]      * | ########
  [4]    chr14 [105613740, 105613759]      * | ########
  ---
  seqlengths:
   chr14
      NA
```

```
> head(fileRange(bfv1)$result[[1]])
```

```
  offset statistic   dm   p.value
1      0 1.1489125 2.75 0.2943227
2      1 0.9761871 2.25 0.3666718
3      2 0.8320503 1.50 0.4372365
4      3 0.8320503 1.50 0.4372365
5      4 0.5222330 1.00 0.6202200
6      5 0.2927700 0.50 0.7795590
```

# 7 sessionInfo()

```
> toLatex(sessionInfo())
```

- R version 3.1.0 (2014-04-10), x86_64-unknown-linux-gnu
- Locale: LC_CTYPE=en_US.UTF-8, LC_NUMERIC=C, LC_TIME=en_US.UTF-8, LC_COLLATE=C, LC_MONETARY=en_US.UTF-8, 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, grDevices, graphics, methods, parallel, stats, utils
- Other packages: BiocGenerics 0.10.0, BiocParallel 0.6.0, Biostrings 2.32.0, GenomeInfoDb 1.0.2, GenomicFiles 1.0.1, GenomicRanges 1.16.1, IRanges 1.22.2, RNAseqData.HNRNPC.bam.chr14 0.2.0, Rsamtools 1.16.0, XVector 0.4.0, rtracklayer 1.24.0
- Loaded via a namespace (and not attached): BBmisc 1.5, BSgenome 1.32.0, BatchJobs 1.2, BiocStyle 1.2.0, DBI 0.2-7, GenomicAlignments 1.0.0, RCurl 1.95-4.1, RSQLite 0.11.4, Rcpp 0.11.1, XML 3.98-1.1, bitops 1.0-6, brew 1.0-6, codetools 0.2-8, digest 0.6.4, fail 1.2, foreach 1.4.2, iterators 1.0.7, plyr 1.8.1, sendmailR 1.1-2, stats4 3.1.0, stringr 0.6.2, tools 3.1.0, zlibbioc 1.10.0