Segvis: A package for visualization of high throughput sequencing data along genomic segments

Rene Welch (welch@stat.wisc.edu) and Sündüz Keleş (keles@stat.wisc.edu)
Department of Statistics, University of Wisconsin - Madison
Madison, WI

April 2015

Contents

1 Overview **1**

2 How to use Segvis? **1**
2.1 Building a set of regions for *Segvis* 2
2.2 Creating a `segvis` object 3
2.3 Creating `segvis_block` object 4

3 SessionInfo **4**

1 Overview

This vignette provides an introduction to the visualization of sequencing data by using the *Segvis* package. The minimum input to the package includes:

1. Coordinates for regions of interest.
2. One or more bam files of aligned read data (e.g. from ChIP-seq experiments).

Segvis provides different tools to summarize and visualize these data, including but not limited to the following tasks:

- Extract read data of specified input regions.
- Plot data from different files (conditions) accross the same set of regions, e.g. peak plots for (SET or PET) ChIP-seq.
- Calculate and plot statistics(e.g. mean, median, variace, etc.) over a window around biologically meaningful coordinates (TSS, TFBS, etc.)
- Subset this regions according to user defined annotations.
- Plot the heatmap of signal curves accross regions separated by annotation.

2 How to use Segvis?

The package can be loaded with the command:

```
library(Segvis)
```

Different visualization of the data is done by the use of three following classes `segvis`, `segvis_block` and `segvis_block_list`. The first one is used to store the reads for a given bam file, the second is the one used to interact with the data and the third one is simply a list made exclusively of `segvis_block` objects.

2.1 Building a set of regions for Segvis

The minimum input for the package includes:

1. Coordinates for regions of interest.
2. One or more bam files of aligned read data (e.g. from ChIP-seq experiments).

The coordinates may be obtained by several means: Visual exploration in the genome browser, calling peaks from a ChIP-seq experiment, etc. To use *Segvis* it is necessary to load the regions of interest into a `segvis` object by formatting them as a *GRanges* object.

For example, if the peaks are saved in a **narrowPeak** file format[1], then we can load it into *R* by using:

```
  peaks_file = "../inst/extdata/peaks/encode_K562_Ctcf_peaks_first3chr.narrowPeak"
  ctcf_peaks = read.table(peaks_file)
  head(ctcf_peaks,15)
```

```
##      V1        V2        V3 V4 V5 V6       V7 V8      V9 V10
## 1  chr1 114889057 114889538  .  0  . 671.4930 -1 4.57207 240
## 2  chr1 225662556 225663044  .  0  . 632.4595 -1 4.57207 249
## 3  chr1 150951878 150952345  .  0  . 601.5148 -1 4.57207 259
## 4  chr1  17036198  17036690  .  0  . 585.6260 -1 4.57207 255
## 5  chr1  35318207  35318710  .  0  . 520.4329 -1 4.57207 262
## 6  chr1 204776085 204776784  .  0  . 519.1454 -1 4.57207 376
## 7  chr1  33177715  33178175  .  0  . 514.7651 -1 4.57207 239
## 8  chr1  19239498  19239983  .  0  . 501.9843 -1 4.57207 230
## 9  chr1  38455665  38456081  .  0  . 496.4812 -1 4.57207 185
## 10 chr1 154989953 154990397  .  0  . 495.8747 -1 4.57207 221
## 11 chr1   9686983   9687432  .  0  . 489.9299 -1 4.57207 216
## 12 chr1 186344435 186344962  .  0  . 485.4386 -1 4.57207 239
## 13 chr1  26221873  26222297  .  0  . 481.1880 -1 4.57207 187
## 14 chr1  47902101  47902527  .  0  . 476.5053 -1 4.57207 185
## 15 chr1 109806165 109806687  .  0  . 476.2783 -1 4.57207 236
```

Then to convert it into a *GRanges* object we can use:

```
  K = 2000
  ctcf_gr = GRanges(seqnames = ctcf_peaks$V1,
    ranges = IRanges(start = ctcf_peaks$V2,
      end = ctcf_peaks$V3),strand = "*")
  ctcf_gr = ctcf_gr[order(ctcf_peaks$V7,decreasing=TRUE)[1:K]]
  ctcf_gr
```

```
## GRanges object with 2000 ranges and 0 metadata columns:
##          seqnames               ranges strand
##             <Rle>            <IRanges>  <Rle>
##      [1]     chr1 [114889057, 114889538]      *
##      [2]     chr1 [225662556, 225663044]      *
##      [3]     chr3 [195577700, 195578158]      *
##      [4]     chr1 [150951878, 150952345]      *
##      [5]     chr2 [ 91814978,  91815458]      *
##      ...      ...                 ...    ...
```

[1] A description of several common file formats is given in https://genome.ucsc.edu/FAQ/FAQformat.html

```
##   [1996]     chr1 [ 43823756,  43824034]      *
##   [1997]     chr1 [205323464, 205323745]      *
##   [1998]     chr2 [ 56410741,  56410988]      *
##   [1999]     chr1 [152430955, 152431197]      *
##   [2000]     chr1 [226033541, 226033846]      *
##   -------
##   seqinfo: 3 sequences from an unspecified genome; no seqlengths
```

A complete description of the meaning of each column in the peaks file is given in https://genome.ucsc.edu/FAQ/FAQformat.html#format12. Using the signal values (on the 7th column), we are going to consider the top 2000 peaks:

2.2 Creating a `segvis` object

To create a *segvis* object, it is necessary to specify the following parameters:

- `name` - The name of the `segvis` object.
- `regions` - The regions to be loaded, in our case those are `ctcf_gr`.
- `file` - The file were the reads of the experiment are stored.
- `maxBandwidth` - The upper bound of all the possible bandwidths used to smooth the coverage plots when creating a `segvis_block` object.
- `fragLen` - The fragment length used to extend the fragment reads. If it is defined as zero, then it would use the original read widths.
- `chr` - The chromosomes for which the `segvis` object is defined. There are a couple of predefined cases as 'human' or 'mouse' to automatically consider all chromosomes in those genomes.
- `isPET` - A logical indicator if the reads of the experiment are paired-ended. In this case, the `fragLen` parameter is ignored.

```
  ctcf = Segvis(name = "ctcf_peaks",
    file = "../inst/extdata/reads/encode_K562_Ctcf_first3chr_Rep1.sort.bam",
    maxBandwidth = 101,fragLen = 200,isPET = FALSE,
    chr = c("chr1","chr2","chr3"))
  regions(ctcf) = ctcf_gr
  ctcf
```

```
## Segvis for ctcf_peaks regions
## Paired-end Tags:  FALSE
## Fragment length: 200
## Max Bandwidth: 101
## Using reads file:
## ../inst/extdata/reads/encode_K562_Ctcf_first3chr_Rep1.sort.bam
## Using regions for 3 chromosomes
## GRanges object with 2000 ranges and 0 metadata columns:
##          seqnames               ranges strand
##             <Rle>            <IRanges>  <Rle>
##      [1]     chr1 [114889057, 114889538]      *
##      [2]     chr1 [225662556, 225663044]      *
##      [3]     chr1 [150951878, 150952345]      *
##      [4]     chr1 [ 17036198,  17036690]      *
##      [5]     chr1 [ 35318207,  35318710]      *
##      ...      ...                  ...    ...
##   [1996]     chr3 [171171060, 171171364]      *
##   [1997]     chr3 [150863855, 150864128]      *
##   [1998]     chr3 [196756655, 196756932]      *
##   [1999]     chr3 [118603610, 118603955]      *
```

```
##   [2000]     chr3 [  9768737,   9769156]      *
##   -------
##   seqinfo: 3 sequences from an unspecified genome; no seqlengths
```

2.3 Creating `segvis_block` object

Segvis allows the use of several cores by using the parameter `mc`, which specifies the number of cores used by parallel processing. To create a `segvis_block` object it is necessary to follow a series of steps:

```
ctcf = loadReads(ctcf, mc = 24)
ctcf = matchReads(ctcf,mc = 24)
ctcf = getCoverage(ctcf,mc = 24)
ctcf_block = Segvis_block(ctcf,bw = 1,mc = 24)
```

To obtain the number of reads considered in an experiment:

```
ctcf_reads = countReads(ctcf)
ctcf_reads
```

```
## [1] 6649370
```

In order to visually compare the enrichment level of several experiments it is necessary to normalize the experiment, which *Segvis* allows to do, by using:

```
normConst(ctcf_block) = ctcf_reads
ctcf_block = normalize(ctcf_block)
```

3 SessionInfo

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

- R version 3.1.1 (2014-07-10), `x86_64-redhat-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=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, graphics, grDevices, methods, parallel, stats, stats4, utils
- Other packages: BiocGenerics 0.12.1, Biostrings 2.34.1, devtools 1.7.0, GenomeInfoDb 1.2.4, GenomicAlignments 1.2.2, GenomicRanges 1.18.4, ggplot2 1.0.1, IRanges 2.0.1, knitr 1.9, rbamtools 2.10.0, Rsamtools 1.18.3, S4Vectors 0.4.0, Segvis 2.0, XVector 0.6.0
- Loaded via a namespace (and not attached): base64enc 0.1-2, BatchJobs 1.6, BBmisc 1.9, BiocParallel 1.0.3, BiocStyle 1.4.1, bitops 1.0-6, brew 1.0-6, checkmate 1.5.1, chron 2.3-45, codetools 0.2-11, colorspace 1.2-6, data.table 1.9.4, DBI 0.3.1, digest 0.6.8, evaluate 0.5.5, fail 1.2, foreach 1.4.2, formatR 1.0, grid 3.1.1, gtable 0.1.2, highr 0.4, iterators 1.0.7, MASS 7.3-39, munsell 0.4.2, plyr 1.8.1, proto 0.3-10, Rcpp 0.11.5, reshape2 1.4.1, roxygen2 4.1.0, RSQLite 1.0.0, scales 0.2.4, sendmailR 1.2-1, stringr 0.6.2, tools 3.1.1, zlibbioc 1.12.0