String Basics

Hervé Pagès

29 January, 2010

Contents

1 Lab overview

In this lab we will:

- Use the BSgenome package for Yeast to extract the GC content for each chromosome as well as for the gene regions of each chromosome.
- Use `coverage` on the aligned reads in `BYe9.head.map` and find whether or not there is a correlation between the coverage of the Yeast genome and its GC content.

2 BSgenome basics

The Bioconductor project provides *BSgenome data packages* for the commonly studied organism. Use the `available.genomes()` function from the BSgenome *software* package to see all the packages available.

The name of a *BSgenome data package* is made of 4 parts separated by a dot (e.g. BSgenome.Celegans.UCSC.ce2):

- The 1st part is always `BSgenome`.

- The 2nd part is the name of the organism (abbreviated).
- The 3rd part is the name of the organisation who assembled the genome.
- The 4th part is the release string or number used by this organisation for this assembly of the genome.

All *BSgenome data package* contain a single top level object whose name matches the second part of the package name.

Exercise 1

1. *Get the list of all available BSgenome data packages.*
2. *Get the list of all installed BSgenome data packages.*
3. *After you've loaded a BSgenome data package, use* `?<name-of-the-package>` *to see useful information about the package and some examples on how to use it.*
4. *What's the quick and easy way to get the lengths of all the sequences stored in a BSgenome data package?*
5. *Load Yeast chrI.*

3 GC content of the chromosomes

The Biostrings package contains many tools to operate on the DNA sequences stored in a *BSgenome data package*.

For example `alphabetFrequency` will compute the number of occurences for each letter in the DNA alphabet.

Exercise 2

1. *Use* `alphabetFrequency` *on Yeast chrI.*
2. *Try again with* `as.prob=TRUE` *and* `baseOnly=TRUE`*.*
3. *Extract the GC content of Yeast chrI.*
4. *Write the* `getChromGCcontent` *function that takes 1 argument (*`chrom`*, must be a valid chromosome name), and returns the GC content for this chromosome. Note that, for convenience, the* `day3` *package provides the* `NORMCHROM2UCSCCHROM` *named vector that you can use to translate normalized chromosome names to UCSC chromosome names. Use it in* `getChromGCcontent` *to make the function work with normalized chromosome names.*
5. *Get the GC content of all chromosomes in an* `sapply` *loop.*

4 GC content of the gene regions for each chromosome

In this section we want to extract the GC content of the regions in the chromosome where the genes are. We will use the gene starts and ends from the org.Sc.sgd.db package to define the regions on the chromosome where we want to restrict the computation of the GC content. For convenience, the day3 package provides the `extractYeastGenesAsRangesList` function that can be called with no argument and will return a *RangesList* object with 1 element per chromosome.

Exercise 3

1. *Call* `extractYeastGenesAsRangesList` *and store the result in* `genes`*.*
2. *For convenience, the day3 package provides the* `normalizeGeneChromosome` *function. Pass* `genes` *thru it in order to normalize its names.*

The IRanges package provides the `Views` constructor to create a set of views on a sequence. Each view is just a start and an end position on the sequence (always 1-based). A simple way to call `Views` is to pass it the sequence and the *Ranges* object containing the starts/ends of the views to create.

Exercise 4

1. *Create the views on Yeast chrI that correspond to the gene locations.*
2. *Use* `alphabetFrequency` *on the Views object with* `collapse=TRUE`*.*
3. *Extract the GC content of the gene regions in chrI.*
4. *Write the* `getGenesGCcontent` *function that takes 2 arguments (*`chrom`*,* `genes`*), and returns the gene GC content for this chromosome.*
5. *Get the gene GC content of all chromosomes in an* `sapply` *loop.*
6. *Compare with the GC content of the chromosomes.*

5 How deep `BYe9.head.map` is covering the Yeast genome and the underlying GC content

For this section we load our aligned reads again and compute their coverage of the Yeast genome:

```
> aln_file <- system.file("extdata", "BYe9.head.map", package="day3")
> aln <- readAligned(aln_file, type="Bowtie")
> cvg <- coverage(aln)
```

For convenience, the day3 package provides the `normalizeAlignedReadChromosome` function that we can use to normalize the names of `cvg`:

```
> cvg <- normalizeAlignedReadChromosome(cvg)
> plotCoverage(cvg, "chr7")
```

The coverage of a given chromosome is a numeric sequence (or signal). The IRanges package provides the `slice` function to create views on such a numeric sequence based on the values in the sequence. The user needs to specify a min and/or a max value and `slice` will create the views that correspond to the regions in the sequence where the signal is within these bounds.

Exercise 5

1. *slice the coverage of chr7 so that the views are the regions where this coverage is at least 20.*
2. *Call `viewMeans` on this slicing. What does it do exactly?*
3. *Create the views on the chr7 DNA sequence that correspond to this slicing.*
4. *Create a plot where each view is represented by a dot. The x of the dot is the mean coverage for that view and its y is the GC content for that view.*

In the last exercise, we want to write a function that will produce this plot for the chromosome specified by the user.

Exercise 6

Write the `plotGCcontentVersusCoverageMean` function that takes 3 arguments (`chrom`, `cvg`, `min.cvg`), and plots the GC content versus the mean coverage for the regions where the coverage is at least `min.cvg`. `cvg` is the RleList object containing the coverage vector for each chromosome.

6 Session information

- R version 2.10.1 Patched (2010-01-28 r51060), `x86_64-unknown-linux-gnu`
- Locale: `LC_CTYPE=C`, `LC_NUMERIC=C`, `LC_TIME=C`, `LC_COLLATE=C`, `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, grDevices, graphics, methods, stats, tools, utils
- Other packages: AnnotationDbi 1.8.1, BSgenome 1.14.2, BSgenome.Scerevisiae.UCSC.sacCer2 1.3.16, Biobase 2.6.1, Biostrings 2.14.8, DBI 0.2-4, IRanges 1.4.9, RCurl 1.3-0, RSQLite 0.7-3, ShortRead 1.4.0, biomaRt 2.2.0, bitops 1.0-4.1, day3 0.0.3, lattice 0.17-26, org.Sc.sgd.db 2.3.5, rtracklayer 1.6.0
- Loaded via a namespace (and not attached): XML 2.6-0, grid 2.10.1, hwriter 1.1