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.*

```
> library(BSgenome)
> available.genomes()

 [1] "BSgenome.Amellifera.BeeBase.assembly4"
 [2] "BSgenome.Amellifera.UCSC.apiMel2"
 [3] "BSgenome.Athaliana.TAIR.01222004"
 [4] "BSgenome.Athaliana.TAIR.04232008"
 [5] "BSgenome.Btaurus.UCSC.bosTau3"
 [6] "BSgenome.Btaurus.UCSC.bosTau4"
 [7] "BSgenome.Celegans.UCSC.ce2"
 [8] "BSgenome.Cfamiliaris.UCSC.canFam2"
 [9] "BSgenome.Dmelanogaster.UCSC.dm2"
[10] "BSgenome.Dmelanogaster.UCSC.dm3"
[11] "BSgenome.Drerio.UCSC.danRer5"
[12] "BSgenome.Ecoli.NCBI.20080805"
[13] "BSgenome.Ggallus.UCSC.galGal3"
[14] "BSgenome.Hsapiens.UCSC.hg17"
[15] "BSgenome.Hsapiens.UCSC.hg18"
[16] "BSgenome.Hsapiens.UCSC.hg19"
[17] "BSgenome.Mmusculus.UCSC.mm8"
[18] "BSgenome.Mmusculus.UCSC.mm9"
[19] "BSgenome.Ptroglodytes.UCSC.panTro2"
[20] "BSgenome.Rnorvegicus.UCSC.rn4"
[21] "BSgenome.Scerevisiae.UCSC.sacCer1"
[22] "BSgenome.Scerevisiae.UCSC.sacCer2"
```

```
> installed.genomes()

[1] "BSgenome.Mmusculus.UCSC.mm9"
[2] "BSgenome.Scerevisiae.UCSC.sacCer2"

> library(BSgenome.Scerevisiae.UCSC.sacCer2)
> Scerevisiae

Yeast genome
|
| organism: Saccharomyces cerevisiae (Yeast)
| provider: UCSC
| provider version: sacCer2
| release date: June 2008
| release name: SGD June 2008 sequence
|
| sequences (see '?seqnames'):
|   chrI     chrII    chrIII   chrIV    chrV     chrVI
|   chrVII   chrVIII  chrIX    chrX     chrXI    chrXII
|   chrXIII  chrXIV   chrXV    chrXVI   chrM     2micron
|
| (use the '$' or '[[' operator to access a given sequence)

> seqlengths(Scerevisiae)

   chrI   chrII  chrIII   chrIV    chrV   chrVI  chrVII
 230208  813178  316617 1531919  576869  270148 1090947
chrVIII   chrIX    chrX   chrXI  chrXII chrXIII  chrXIV
 562643  439885  745742  666454 1078175  924429  784333
  chrXV  chrXVI    chrM 2micron
1091289  948062   85779    6318

> Scerevisiae$chrI

  230208-letter "DNAString" instance
seq: CCACACCACACCCACACACCCACACA...TGTGGTGTGGGTGTGGTGTGTGTGGG

> Scerevisiae[["chrI"]]

  230208-letter "DNAString" instance
seq: CCACACCACACCCACACACCCACACA...TGTGGTGTGGGTGTGGTGTGTGTGGG
```

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.*

```
> alphabetFrequency(Scerevisiae$chrI)

    A     C     G     T     M     R     W     S     Y     K
69826 44646 45765 69971     0     0     0     0     0     0
    V     H     D     B     N     -     +
    0     0     0     0     0     0     0

> af <- alphabetFrequency(Scerevisiae$chrI, as.prob=TRUE, baseOnly=TRUE)
> sum(af[2:3])  # GC content of Yeast chrI

[1] 0.39

> library(day3)  # to get NORMCHROM2UCSCCHROM
> NORMCHROM2UCSCCHROM

     chr1      chr2      chr3      chr4      chr5      chr6
   "chrI"   "chrII"  "chrIII"   "chrIV"    "chrV"   "chrVI"
     chr7      chr8      chr9     chr10     chr11     chr12
 "chrVII" "chrVIII"   "chrIX"    "chrX"   "chrXI"  "chrXII"
    chr13     chr14     chr15     chr16     chrmt
"chrXIII"  "chrXIV"   "chrXV"  "chrXVI"    "chrM"

> getChromGCcontent <- function(chrom)
+ {
+     library(BSgenome.Scerevisiae.UCSC.sacCer2)
+     chrom_seq <- Scerevisiae[[NORMCHROM2UCSCCHROM[[chrom]]]]
+     af <- alphabetFrequency(chrom_seq, as.prob=TRUE, baseOnly=TRUE)
+     sum(af[2:3])
+ }
> getChromGCcontent("chr1")

[1] 0.39
```

```
> chromGCcontent <- sapply(names(NORMCHROM2UCSCCHROM),
+                          function(chrom) getChromGCcontent(chrom))
```

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.*

```
> genes <- normalizeGeneChromosome(extractYeastGenesAsRangesList())
```

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.*

```
> v <- Views(Scerevisiae$chrI, genes$chr1)
> v
```

```
  Views on a 230208-letter DNAString subject
subject: CCACACCACACCCACACACCCACA...TGGTGTGGGTGTGGTGTGTGTGGG
views:
       start    end width
  [1] 151467 151584   118 [GTCACATGACATAAT...TCCGAAGCAGTCAA]
  [2]  99306  99869   564 [GGGCCCTTTCTTCCG...TTTTCTGGCAAAAA]
  [3] 147596 151168  3573 [TTATGTAGATTCATA...ATCGTCAGTACCAT]
  [4] 143709 147533  3825 [ATGGAGCAAAATGGC...CAAAAATAGTATAA]
  [5] 142176 143162   987 [ATGGCATCCACCGAT...TGCAAAAATTATAA]
  [6] 139505 141433  1929 [TTAATCAACTTCTTC...ACAGCTTTTGACAT]
  [7] 137700 138347   648 [TTAAACGTAATTTTT...GTAGATTTGATCAT]
  [8] 136916 137512   597 [ATGACTTTGGCTTTT...ATGCTAACAAATGA]
  [9] 135856 136635   780 [ATGGAGCCAGAGAGC...CTAAATCAGAATGA]
  ...    ...    ...   ... ...
[137] 160106 160238   133 [TATATTATCGATCCT...AAGATAGTTGGTTC]
[138] 160239 160575   337 [TGAGAAATGGGTGAA...TTTCTATTCCAACA]
[139] 165827 166163   337 [TGAGAAATGGGTGAA...TTTCTATTCCAACA]
[140] 209439 209769   331 [TGGCGAATGTTTTGA...GTCTTATCCCAACA]
[141] 183136 183468   333 [TGTTGGAATAAAAAT...TCCCCCAATTCTCA]
[142] 189420 189751   332 [TGTTGGAATAAAAAT...TCCCCCAATTCTCA]
[143] 182614 182953   340 [TGTTGTATCTCAAAA...CCCGTAATACAACA]
[144] 166268 166340    73 [GGGCACATGGCGCAG...TCCGGTTGCGTCCA]
[145] 182516 182597    82 [CGACAACTGCAGGAC...TCGGCCAAGTTGCC]

> af <- alphabetFrequency(v, as.prob=TRUE, baseOnly=TRUE, collapse=TRUE)
> sum(af[2:3])

[1] 0.41

> getGenesGCcontent <- function(chrom, genes)
+ {
+     library(BSgenome.Scerevisiae.UCSC.sacCer2)
+     chrom_seq <- Scerevisiae[[NORMCHROM2UCSCCHROM[[chrom]]]]
+     v <- Views(chrom_seq, genes[[chrom]])
+     af <- alphabetFrequency(v, as.prob=TRUE, baseOnly=TRUE, collapse=TRUE)
+     sum(af[2:3])
+ }
> getGenesGCcontent("chr1", genes)

[1] 0.41

> genesGCcontent <- sapply(names(NORMCHROM2UCSCCHROM),
+                          function(chrom) getGenesGCcontent(chrom, genes))
> all(genesGCcontent >= chromGCcontent)

[1] TRUE
```

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.*

```
> s <- slice(cvg$chr7, lower=20)
> s

Views on a 1090607-length Rle subject

views:
       start    end width
  [1]  80607  80609     3 [20 21 21]
  [2]  80612  80632    21 [20 20 20 20 20 20 20 20 20 ...]
  [3] 144831 144831     1 [21]
  [4] 144835 144847    13 [20 20 20 21 21 22 21 21 21 ...]
  [5] 144896 144898     3 [20 20 20]
  [6] 145068 145068     1 [20]
  [7] 145070 145071     2 [20 20]
  [8] 145073 145080     8 [20 20 22 21 21 20 20 20]
```

```
  [9] 145120 145145    26 [20 20 21 21 21 22 22 22 23 ...]
  ...    ...    ...   ... ...
[190] 939503 939535    33 [42 52 54 56 56 56 56 56 56 ...]
[191] 939543 939545     3 [21 20 21]
[192] 939547 939549     3 [20 20 21]
[193] 939554 939562     9 [21 22 21 22 22 21 21 20 20]
[194] 939566 939571     6 [20 21 20 20 20 20]
[195] 939577 939580     4 [20 21 20 20]
[196] 939618 939654    37 [20 20 21 23 25 30 40 40 39 ...]
[197] 978084 978085     2 [20 20]
[198] 978087 978099    13 [20 20 20 20 20 20 20 21 21 ...]

> viewMeans(s)

  [1] 21 20 21 21 20 20 20 20 22 20 21 29 20 23 20 22 20 21
 [19] 20 20 20 21 22 20 20 20 22 20 21 20 20 21 21 21 33 24
 [37] 35 34 69 68 38 20 20 20 20 20 20 20 20 20 25 23 21 21
 [55] 20 22 23 20 24 20 27 23 20 26 20 25 24 26 21 20 23 20
 [73] 21 20 21 20 27 26 22 20 20 25 21 23 28 27 21 28 30 24
 [91] 20 25 30 20 24 20 21 21 29 22 22 27 24 20 20 20 20 22
[109] 20 20 20 25 25 20 22 20 22 22 21 20 20 20 20 20 20 21
[127] 23 20 20 24 20 20 21 23 23 20 21 21 21 23 21 20 20 20
[145] 20 21 22 21 22 21 22 20 20 21 25 23 28 26 25 90 58 30
[163] 25 20 20 27 21 29 24 20 20 23 27 24 27 25 20 23 22 24
[181] 20 23 25 20 20 20 22 23 22 57 21 20 21 20 20 33 20 20

> v <- Views(Scerevisiae$chrVII, s)
> v

  Views on a 1090947-letter DNAString subject
subject: CCACACCCACACACACCACACCCA...GGTTTCATTTTCATTTTTTTTTTT
views:
       start    end width
  [1]  80607  80609     3 [AAA]
  [2]  80612  80632    21 [AAAAAAAAAAAAAAAAAAAAA]
  [3] 144831 144831     1 [T]
  [4] 144835 144847    13 [AGATATGCATCTT]
  [5] 144896 144898     3 [TGG]
  [6] 145068 145068     1 [A]
  [7] 145070 145071     2 [AT]
  [8] 145073 145080     8 [TGCCAGAC]
  [9] 145120 145145    26 [TCTAAGCCTTTTTTCTGGGGTGACGG]
  ...    ...    ...   ... ...
[190] 939503 939535    33 [AAAAAATATGGCAAG...CAGTAACGGACAGC]
[191] 939543 939545     3 [GTT]
[192] 939547 939549     3 [CTA]
[193] 939554 939562     9 [GACAACACC]
```

```
[194] 939566 939571     6 [ATGGTT]
[195] 939577 939580     4 [AAAA]
[196] 939618 939654    37 [AAAAGGCCCACAGTT...TGACCTCCCTCCGC]
[197] 978084 978085     2 [AA]
[198] 978087 978099    13 [TCATCAAACGAAT]
```

```
> af <- alphabetFrequency(v, as.prob=TRUE, baseOnly=TRUE)
> GCcontent <- rowSums(af[ , 2:3])
> plot(viewMeans(s), GCcontent, log="x")
```

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.*

```
> plotGCcontentVersusCoverageMean <- function(chrom, cvg, min.cvg=8)
+ {
+     library(BSgenome.Scerevisiae.UCSC.sacCer2)
+     chrom_seq <- Scerevisiae[[NORMCHROM2UCSCCHROM[[chrom]]]]
+     chrom_cvg <- cvg[[chrom]]
+     s <- slice(chrom_cvg, lower=min.cvg)
+     v <- Views(chrom_seq, s)
+     af <- alphabetFrequency(v, as.prob=TRUE, baseOnly=TRUE)
+     GCcontent <- rowSums(af[ , 2:3])
+     plot(viewMeans(s), GCcontent, log="x")
+     ## we add an horizontal line to show the chromosome GC content
+     chrom_GCcontent <- getChromGCcontent(chrom)
+     lines(x=c(1, max(viewMeans(s))), y=c(chrom_GCcontent, chrom_GCcontent))
+ }
> plotGCcontentVersusCoverageMean("chr7", cvg)
```

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