Managing sequence and annotation data using Biostrings and BSgenome

Patrick Aboyoun
Fred Hutchinson Cancer Research Center
Seattle, WA 98008

22 January 2009

Contents

1 Lab Overview

This lab is designed to teach the basics of Biostrings and *BSgenome data packages*. For this lab you need:

- A laptop with a recent build of R-devel (R 2.9 series).
- The Biostrings, BSgenome and BSgenome.Mmusculus.UCSC.mm9 packages.
- `topReads.rda`: serialized object containing the top 1000 reads for all 8 Solexa lanes of 2 ChIP-seq experiments.

2 Setup

Exercise 1
Start an R session and use the `library` *function to load the BSgenome.Mmusculus.UCSC.mm9 genome package along with its dependencies using the following commands:*

```
> library("BSgenome.Mmusculus.UCSC.mm9")
```

This lab also requires you have access to sample data set `topReads.rda`.

Exercise 2
Copy the data from the distribution media to your local hard drive. Change the working directory in R to point to the data location.

```
> setwd(file.path("path", "to", "data"))
```

Exercise 3
Use the `load` *function to load the pre-processed top short reads object from the data directory into your R session.*

```
> load(file.path("data", "topReads.rda"))
```

3 Basic containers

3.1 DNAString objects

The *DNAString* class is the basic container for storing a large nucleotide sequence. Unlike a standard character vector in R that can store an arbitrary number of strings, a *DNAString* object can only contain one sequence.

Exercise 4

1. *Create an object* `r1` *by using the* `[[` *operator to extract the first read from experiment 2, lane 1 to obtain a DNAString object.*
2. *Use the* `nchar` *and* `alphabetFrequency` *function to obtain the number of characters and alphabet frequency.*
3. *Get its reverse complement.*
4. *Extract an arbitrary substring with* `subseq`*.*

```
> r1 <- topReads[["experiment2"]][["lane1"]][, "read"][[1]]
> nchar(r1)

[1] 36

> alphabetFrequency(r1)

 A  C  G  T  M  R  W  S  Y  K  V  H  D  B  N  -  +
 8  8 10 10  0  0  0  0  0  0  0  0  0  0  0  0  0

> reverseComplement(r1)

  36-letter "DNAString" instance
seq: TTTCAAGCAGAAGACGGCATACGAGCTCTTCCGATC
```

```
> subseq(r1, start = 5, end = 15)

  11-letter "DNAString" instance
seq: GGAAGAGCTCG

> subseq(r1, end = 15)

  15-letter "DNAString" instance
seq: GATCGGAAGAGCTCG

> subseq(r1, start = -5)

  5-letter "DNAString" instance
seq: TGAAA
```

3.2 DNAStringSet objects

The *DNAStringSet* class is the basic container for storing an arbitrary number of nucleotide sequences. As with R character vectors (and any vector-like object in general), the `length` function returns the number of elements (sequences) stored in a *DNAStringSet* object and the `[` operator to subset it. In addition, subsetting operator `[[` can be used to extract an arbitrary element as a *DNAString* object.

Exercise 5

1. *Use the* `DNAStringSet` *constructor to store the 1000 reads from experiment 2 / lane 1 into a DNAStringSet object. Let's call this instance* `dict0`*.*
2. *Use* `length` *and* `width` *on* `dict0`*.*
3. *Use subsetting operator* `[` *to remove its 2nd element.*
4. *Use the* `rev` *to invert the order of its elements.*
5. *Use subsetting operator* `[[` *to extract its 1st element as a DNAString object.*
6. *Use the* `DNAStringSet` *constructor (i) to remove the last 2 nucleotides of each element, then (ii) to keep only the last 10 nucleotides.*
7. *Call* `alphabetFrequency` *on* `dict0` *and on its reverse complement. Try again with* `collapse=TRUE`*.*
8. *Remove reads with Ns (put the "clean" dictionary in* `dict0` *again).*

```
> dict0 <- topReads[["experiment2"]][["lane1"]][, "read"]
> length(dict0)

[1] 1000

> table(width(dict0))

  36
1000

> dict0[-2]
```

```
  A DNAStringSet instance of length 999
      width seq
  [1]    36 GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTGAAA
  [2]    36 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
  [3]    36 ANNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
  [4]    36 GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTGGAT
  [5]    36 GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTTGAT
  [6]    36 GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTATAT
  [7]    36 GNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
  [8]    36 CNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
  [9]    36 TNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
  ...   ... ...
[991]    36 TGTCCACTGTAGGACGTGGAATATGGCAAGAAAACT
[992]    36 ATTCCTCCCGACACATAATAATCAGAACAACAAATG
[993]    36 ATTGATATACACTGTTCTACAAATCCCGTTTCCAAC
[994]    36 ANNNNNNNNNAAAAAANNNNANNAAAAAAAAAAAAA
[995]    36 ANNNNNNNNNNNNNNNNNNNNNNNNAANNNANNNNNN
[996]    36 CATATTCCAGGTCCTACAGTGTGCATTTCTCATTTT
[997]    36 CNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNTN
[998]    36 GATCGGAAGAGCTCGTATGCCGCCTTCTGCTTGGAT
[999]    36 GATCGGAAGAGCTCGTATGCCGGTCTTCTGTTTAGA

> rev(dict0)

  A DNAStringSet instance of length 1000
       width seq
   [1]    36 GATCGGAAGAGCTCGTATGCCGGTCTTCTGTTTAGA
   [2]    36 GATCGGAAGAGCTCGTATGCCGCCTTCTGCTTGGAT
   [3]    36 CNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNTN
   [4]    36 CATATTCCAGGTCCTACAGTGTGCATTTCTCATTTT
   [5]    36 ANNNNNNNNNNNNNNNNNNNNNNNNAANNNANNNNNN
   [6]    36 ANNNNNNNNNAAAAAANNNNANNAAAAAAAAAAAAA
   [7]    36 ATTGATATACACTGTTCTACAAATCCCGTTTCCAAC
   [8]    36 ATTCCTCCCGACACATAATAATCAGAACAACAAATG
   [9]    36 TGTCCACTGTAGGACGTGGAATATGGCAAGAAAACT
   ...   ... ...
 [992]    36 CNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
 [993]    36 GNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
 [994]    36 GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTATAT
 [995]    36 GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTTGAT
 [996]    36 GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTGGAT
 [997]    36 ANNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
 [998]    36 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
 [999]    36 GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTAGAT
[1000]    36 GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTGAAA

> dict0[[1]]

  36-letter "DNAString" instance
seq: GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTGAAA

> DNAStringSet(dict0, end = -3)
```

```
  A DNAStringSet instance of length 1000
       width seq
   [1]    34 GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTGA
   [2]    34 GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTAG
   [3]    34 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
   [4]    34 ANNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
   [5]    34 GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTGG
   [6]    34 GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTTG
   [7]    34 GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTAT
   [8]    34 GNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
   [9]    34 CNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
   ...   ... ...
 [992]    34 TGTCCACTGTAGGACGTGGAATATGGCAAGAAAA
 [993]    34 ATTCCTCCCGACACATAATAATCAGAACAACAAA
 [994]    34 ATTGATATACACTGTTCTACAAATCCCGTTTCCA
 [995]    34 ANNNNNNNNNAAAAANNNNANNAAAAAAAAAAAA
 [996]    34 ANNNNNNNNNNNNNNNNNNNNNNNNAANNNANNNN
 [997]    34 CATATTCCAGGTCCTACAGTGTGCATTTCTCATT
 [998]    34 CNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
 [999]    34 GATCGGAAGAGCTCGTATGCCGCCTTCTGCTTGG
[1000]    34 GATCGGAAGAGCTCGTATGCCGGTCTTCTGTTTA

> DNAStringSet(dict0, start = -10)

  A DNAStringSet instance of length 1000
       width seq
   [1]    10 CTGCTTGAAA
   [2]    10 CTGCTTAGAT
   [3]    10 AAAAAAAAAA
   [4]    10 NNNNNNNNNN
   [5]    10 CTGCTTGGAT
   [6]    10 CTGCTTTGAT
   [7]    10 CTGCTTATAT
   [8]    10 NNNNNNNNNN
   [9]    10 NNNNNNNNNN
   ...   ... ...
 [992]    10 CAAGAAAACT
 [993]    10 ACAACAAATG
 [994]    10 CGTTTCCAAC
 [995]    10 AAAAAAAAAA
 [996]    10 NNNANNNNNN
 [997]    10 TTCTCATTTT
 [998]    10 NNNNNNNNTN
 [999]    10 CTGCTTGGAT
[1000]    10 TCTGTTTAGA

> head(alphabetFrequency(dict0))

      A C  G  T M R W S Y K V H D B  N - +
[1,]  8 8 10 10 0 0 0 0 0 0 0 0 0 0  0 0 0
[2,]  7 8 10 11 0 0 0 0 0 0 0 0 0 0  0 0 0
[3,] 36 0  0  0 0 0 0 0 0 0 0 0 0 0  0 0 0
```

```
[4,]  1 0  0  0 0 0 0 0 0 0 0 0 0 0 35 0 0
[5,]  6 8 11 11 0 0 0 0 0 0 0 0 0 0  0 0 0
[6,]  6 8 10 12 0 0 0 0 0 0 0 0 0 0  0 0 0

> reverseComplement(dict0)

  A DNAStringSet instance of length 1000
       width seq
   [1]    36 TTTCAAGCAGAAGACGGCATACGAGCTCTTCCGATC
   [2]    36 ATCTAAGCAGAAGACGGCATACGAGCTCTTCCGATC
   [3]    36 TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
   [4]    36 NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNT
   [5]    36 ATCCAAGCAGAAGACGGCATACGAGCTCTTCCGATC
   [6]    36 ATCAAAGCAGAAGACGGCATACGAGCTCTTCCGATC
   [7]    36 ATATAAGCAGAAGACGGCATACGAGCTCTTCCGATC
   [8]    36 NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNC
   [9]    36 NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNG
   ...   ... ...
 [992]    36 AGTTTTCTTGCCATATTCCACGTCCTACAGTGGACA
 [993]    36 CATTTGTTGTTCTGATTATTATGTGTCGGGAGGAAT
 [994]    36 GTTGGAAACGGGATTTGTAGAACAGTGTATATCAAT
 [995]    36 TTTTTTTTTTTTTTNNTNNNNTTTTTNNNNNNNNNT
 [996]    36 NNNNNNTNNNTTNNNNNNNNNNNNNNNNNNNNNNNT
 [997]    36 AAAATGAGAAATGCACACTGTAGGACCTGGAATATG
 [998]    36 NANNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNG
 [999]    36 ATCCAAGCAGAAGGCGGCATACGAGCTCTTCCGATC
[1000]    36 TCTAAACAGAAGACCGGCATACGAGCTCTTCCGATC

> alphabetFrequency(dict0, collapse = TRUE)

   A    C    G    T    M    R    W    S    Y    K    V    H    D    B
9713 5970 6197 8955    0    0    0    0    0    0    0    0    0    0
   N    -    +
5165    0    0

> alphabetFrequency(reverseComplement(dict0), collapse = TRUE)

   A    C    G    T    M    R    W    S    Y    K    V    H    D    B
8955 6197 5970 9713    0    0    0    0    0    0    0    0    0    0
   N    -    +
5165    0    0

> dict0 <- dict0[alphabetFrequency(dict0)[, "N"] == 0]
```

3.3 XStringViews objects

An *XStringViews* object contains a set of views on the same sequence called *the subject* (for example this can be a *DNAString* object). Each view is defined by its start and end locations: both are integers such that start <= end. The `Views` function can be used to create an *XStringViews* object given a subject and a set of start and end locations. Like for *DNAStringSet* objects, `length`, `width`, `[` and `[[` are supported for *XStringViews* objects. Additional `subject`, `start`, `end` and `gaps` methods are also provided.

Exercise 6

1. *Use the `Views` function to create an XStringViews object with a DNAString subject. Make it such that some views are overlapping but also that the set of views don't cover the subject entirely.*

2. *Try* `subject`, `start`, `end` *and* `gaps` *on this XStringViews object.*

3. *Try* `alphabetFrequency` *on it.*

4. *Turn it into a DNAStringSet object with the* `DNAStringSet` *constructor.*

```
> v3 <- Views(dict0[[1]], start = c(2, 12, 20), end = c(5,
+     26, 27))
> subject(v3)

  36-letter "DNAString" instance
seq: GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTGAAA

> start(v3)

[1]  2 12 20

> end(v3)

[1]  5 26 27

> gaps(v3)

  Views on a 36-letter DNAString subject
subject: GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTGAAA
views:
    start end width
[1]     1   1     1 [G]
[2]     6  11     6 [GAAGAG]
[3]    28  36     9 [TGCTTGAAA]

> alphabetFrequency(v3)

     A C G T M R W S Y K V H D B N - +
[1,] 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
[2,] 1 5 3 6 0 0 0 0 0 0 0 0 0 0 0 0 0
[3,] 0 4 1 3 0 0 0 0 0 0 0 0 0 0 0 0 0

> DNAStringSet(v3)

  A DNAStringSet instance of length 3
    width seq
[1]     4 ATCG
[2]    15 CTCGTATGCCGTCTT
[3]     8 CCGTCTTC
```

4 BSgenome data packages

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 7

1. *Load* `BSgenome.Mmusculus.UCSC.mm9` *and display its top level object. Note that this doesn't load any sequence in memory yet.*
2. *Use* `seqlengths` *on it to get the lengths of the single sequences (this doesn't load any sequence either).*
3. *Display some of the chromosomes. Some information about the built-in masks is displayed. Let's drop the masks for now by accessing the sequences with e.g.* `unmasked(Mmusculus$chrM)`*. Note that a sequence is not loaded until it is accessed.*
4. *Do the chromosomes contain IUPAC extended letters?*
5. *Use* `chartr` *to simulate a bisulfite transformation of chromosome 1 (see* `?chartr`*).*

```
> library("BSgenome.Mmusculus.UCSC.mm9")
> Mmusculus

Mouse genome
|
| organism: Mus musculus
| provider: UCSC
| provider version: mm9
| release date: Jul. 2007
| release name: NCBI Build 37
|
| single sequences (see '?seqnames'):
|   chr1          chr2          chr3          chr4          chr5
|   chr6          chr7          chr8          chr9          chr10
|   chr11         chr12         chr13         chr14         chr15
|   chr16         chr17         chr18         chr19         chrX
|   chrY          chrM          chr1_random   chr3_random   chr4_random
|   chr5_random   chr7_random   chr8_random   chr9_random   chr13_random
|   chr16_random  chr17_random  chrX_random   chrY_random   chrUn_random
|
| multiple sequences (see '?mseqnames'):
|   upstream1000  upstream2000  upstream5000
|
| (use the '$' or '[[' operator to access a given sequence)

> seqlengths(Mmusculus)

        chr1        chr2        chr3        chr4        chr5
   197195432   181748087   159599783   155630120   152537259
        chr6        chr7        chr8        chr9       chr10
   149517037   152524553   131738871   124076172   129993255
       chr11       chr12       chr13       chr14       chr15
   121843856   121257530   120284312   125194864   103494974
       chr16       chr17       chr18       chr19        chrX
    98319150    95272651    90772031    61342430   166650296
```

```
        chrY          chrM  chr1_random  chr3_random  chr4_random
    15902555         16299      1231697        41899       160594
 chr5_random   chr7_random  chr8_random  chr9_random chr13_random
      357350        362490       849593       449403       400311
chr16_random chr17_random  chrX_random  chrY_random chrUn_random
        3994       628739      1785075     58682461      5900358

> Mmusculus$chrM

  16299-letter "MaskedDNAString" instance (# for masking)
seq: GTTAATGTAGCTTAATAACAAAGCAAAGCACT...ATCATACTCTATTACGCAATAAACATTAACAA
masks:
  maskedwidth maskedratio active names
1           0  0.00000000   TRUE AGAPS
2           0  0.00000000   TRUE   AMB
3         414  0.02540033  FALSE    RM
4           0  0.00000000  FALSE   TRF
                                     desc
1                    assembly gaps (empty)
2           intra-contig ambiguities (empty)
3                             RepeatMasker
4 Tandem Repeats Finder [period<=12] (empty)
all masks together:
  maskedwidth maskedratio
          414  0.02540033
all active masks together:
  maskedwidth maskedratio
            0           0

> unmasked(Mmusculus$chrM)

  16299-letter "DNAString" instance
seq: GTTAATGTAGCTTAATAACAAAGCAAAGCACT...ATCATACTCTATTACGCAATAAACATTAACAA

> af <- sapply(seqnames(Mmusculus), function(name) alphabetFrequency(unmasked(Mmusculus[[name]]),
+     baseOnly = TRUE))
> af

          chr1     chr2     chr3     chr4     chr5     chr6     chr7
A     56406566 51614200 46502602 43772553 42432935 42837793 40278036
C     39397656 37519269 31602626 32114659 31415883 30312298 30547699
G     39371416 37553084 31658981 32132726 31417037 30286505 30516113
T     56301791 51705519 46629703 43866906 42455330 42880438 40536362
other  5718003  3356015  3205871  3743276  4816074  3200003 10646343
          chr8     chr9    chr10    chr11    chr12    chr13    chr14
A     35996782 34590008 37115263 33400682 34037210 34040590 35745762
C     26445298 25793837 26219914 26022132 24492820 24216626 25018093
G     26428227 25770013 26296202 26004005 24552042 24209942 25044786
T     35926464 34566364 37216470 33316737 34377238 33903746 35826668
other  6942100  3355950  3145406  3100300  3798220  3913408  3559555
         chr15    chr16    chr17    chr18    chr19     chrX     chrY
A     29111909 28031741 26303792 25659082 16713262 49240037   845394
```

```
C      21078483 19435740 19626270 18124262 12437065 31796795   524639
G      21068688 19457453 19607250 18174526 12408688 31812478   528217
T      29180894 28079995 26360890 25642221 16583215 49231582   804305
other   3055000  3314221  3374449  3171940  3200200  4569404 13200000
       chrM chr1_random chr3_random chr4_random chr5_random chr7_random
A      5629      282444       11331       31914       93565       90912
C      3976      226177        9084       19002       62811       62686
G      2013      221341        9782       19834       63289       64308
T      4681      278840       11702       36872       87685       94584
other     0      222895           0       52972       50000       50000
       chr8_random chr9_random chr13_random chr16_random chr17_random
A           203014      116012        95682         1601       144546
C           153618       75853        71613          832       114307
G           152837       77592        72357          719       117771
T           190124      113206        96213          842       152115
other       150000       66740        64446            0       100000
       chrX_random chrY_random chrUn_random
A           375899    16395141       926119
C           249457    10418246       679674
G           249384    10418764       679821
T           387384    16400310       910977
other       522951     5050000      2703767
```

```
> plus_strand <- chartr("C", "T", unmasked(Mmusculus$chr1))
> alphabetFrequency(plus_strand)

       A        C        G        T        M        R        W        S
56406566        0 39371416 95699447        0        0        0        0
       Y        K        V        H        D        B        N        -
       0        0        0        0        0        0  5718003        0
       +
       0

> minus_strand <- chartr("G", "A", unmasked(Mmusculus$chr1))
> alphabetFrequency(minus_strand)

       A        C        G        T        M        R        W        S
95777982 39397656        0 56301791        0        0        0        0
       Y        K        V        H        D        B        N        -
       0        0        0        0        0        0  5718003        0
       +
       0
```

5 String matching

5.1 The matchPattern function

This function finds all the occurences (aka *matches* or *hits*) of a given pattern in a reference sequence called *the subject*.

Exercise 8

1. *Find all the matches of a short pattern (invent one) in mouse chromosome 1. Don't choose the pattern too short or too long.*

2. *In fact, if we don't take any special action, we only get the hits in the plus strand of the chromosome. Find the matches in the minus strand too. (Note: the cost of taking the reverse complement of an entire chromosome sequence can be high in terms of memory usage. Try to do something better.)*

3. `matchPattern` *now support indels (recent improvement) via the* `with.indels` *argument. Use the same pattern to find all the matches in chromosome 1 that are at an edit distance <= 2 from it.*

```
> pattern <- DNAString("ACCGGTTATC")
> matchPattern(pattern, Mmusculus$chr1)

  Views on a 197195432-letter DNAString subject
subject: NNNNNNNNNNNNNNNNNNNNNNNNNNNNNN...AATTTGGTATTAAACTTAAAACTGGAATTC
views:
        start       end width
[1]   8156832   8156841    10 [ACCGGTTATC]
[2]  12001296  12001305    10 [ACCGGTTATC]
[3]  75279793  75279802    10 [ACCGGTTATC]
[4]  82285523  82285532    10 [ACCGGTTATC]
[5]  88424005  88424014    10 [ACCGGTTATC]
[6] 126585955 126585964    10 [ACCGGTTATC]
[7] 138355255 138355264    10 [ACCGGTTATC]
[8] 161627898 161627907    10 [ACCGGTTATC]
[9] 193744211 193744220    10 [ACCGGTTATC]

> matchPattern(reverseComplement(pattern), Mmusculus$chr1)

  Views on a 197195432-letter DNAString subject
subject: NNNNNNNNNNNNNNNNNNNNNNNNNNNNNN...AATTTGGTATTAAACTTAAAACTGGAATTC
views:
         start       end width
 [1]  17987412  17987421    10 [GATAACCGGT]
 [2]  43662665  43662674    10 [GATAACCGGT]
 [3]  63457603  63457612    10 [GATAACCGGT]
 [4]  72305737  72305746    10 [GATAACCGGT]
 [5]  98629056  98629065    10 [GATAACCGGT]
 [6] 140427341 140427350    10 [GATAACCGGT]
 [7] 151232858 151232867    10 [GATAACCGGT]
 [8] 155572774 155572783    10 [GATAACCGGT]
 [9] 172823371 172823380    10 [GATAACCGGT]
[10] 182880551 182880560    10 [GATAACCGGT]
[11] 187652744 187652753    10 [GATAACCGGT]
[12] 194050463 194050472    10 [GATAACCGGT]

> matchPattern(pattern, Mmusculus$chr1, max.mismatch = 2,
+     with.indels = TRUE)

  Views on a 197195432-letter DNAString subject
subject: NNNNNNNNNNNNNNNNNNNNNNNNNNNNNN...AATTTGGTATTAAACTTAAAACTGGAATTC
views:
            start       end width
    [1]   3000946   3000954     9 [ACCTGTTAT]
    [2]   3007605   3007613     9 [ACCTGTATC]
    [3]   3008007   3008016    10 [ACCTGGTATC]
    [4]   3010957   3010965     9 [CCGGTTGTC]
    [5]   3011092   3011100     9 [CCGGTTGTC]
    [6]   3011156   3011164     9 [ACCGCTTTC]
    [7]   3017808   3017817    10 [ACAGTTTATC]
    [8]   3025912   3025919     8 [ACCGGTTT]
    [9]   3027212   3027220     9 [CTGGTTATC]
    ...       ...       ...   ... ...
[54196] 197169777 197169785     9 [ACCAGTTAC]
[54197] 197173103 197173112    10 [ACAGGTTATC]
[54198] 197173437 197173445     9 [ACAGGTATC]
[54199] 197173488 197173497    10 [AGCTGTTATC]
[54200] 197177320 197177329    10 [ACGGGTTCTC]
[54201] 197177466 197177473     8 [ACCGGTAT]
[54202] 197180598 197180607    10 [ACCTGTTGTC]
[54203] 197188088 197188097    10 [ACAGGTTATC]
[54204] 197194661 197194670    10 [ACAAGTTATC]
```

5.2 The vmatchPattern function

This function finds all the matches of a given pattern in a set of reference sequences.

Exercise 9

1. *Load the* `upstream5000` *object from* `Mmusculus` *and find all the matches of a short arbitrary pattern in it.*
2. *The value returned by* `vmatchPattern` *is an MIndex object containing the match coordinates for each reference sequence. You can use the* `startIndex` *and* `endIndex` *accessors on it to extract the match starting and ending positions as lists (one list element per reference sequence).* `[[` *extracts the matches of a given reference sequence as an MIndex object.* `coundIndex` *extract the match counts as an integer vector (one element per reference sequence).*

```
> Mmusculus$upstream5000

  A DNAStringSet instance of length 18429
        width seq                                    names
    [1]  5000 AGGAAGAACATATTCTC...GAACGCGGGGCTTTCTA NM_028778_up_5000...
    [2]  5000 ATCCCAAAAGTCCCCCA...TCTTCAGCTGGAGCTGG NM_027671_up_5000...
    [3]  5000 TTCTTTACTTAGAAAGT...ACTTGGATAAGGCGCAA NM_175642_up_5000...
    [4]  5000 TGGGTCAAGCATACAAA...CTCCCGCCACTGGGAGA NM_008922_up_5000...
    [5]  5000 GTAGCCCAAGTGCTCAG...CCATCCTGGGGCACAAG NM_175370_up_5000...
    [6]  5000 ATGAAACCACTATGATA...CGCGAGCCTGACGTTGC NM_178884_up_5000...
    [7]  5000 TTGTGTGCATCATTTCA...CTGCTAACTTCTGCCTT NM_009126_up_5000...
    [8]  5000 ATTAACCTGATCCTGAT...GCCACACACAGGCTTCT NM_198680_up_5000...
    [9]  5000 AGCAGAGAGACTCTTTC...GCTTTTCTCTTCCGCCA NM_199021_up_5000...
    ...   ... ...
[18421]  5000 TTAAGAACTTTCACGCT...TTTTTTTTTTTGCCATT NM_001037748_up_5...
[18422]  5000 GCCATTCCAAAAAAGTT...GGACTTGAAGGTGGAGG NM_011667_up_5000...
[18423]  5000 TGCATTAGGCACACATA...TTCAAGGTGAGTTCACT NM_001017393_up_5...
[18424]  5000 AAGAGAAATAATTGATC...TTTTTTTTTTTGCCATT NM_001037748_up_5...
[18425]  5000 GTGGGTGTTAGAAATTG...GCGCATCTATTCCACTT NM_001025241_up_5...
[18426]  5000 ACTATTGATCCTTAGGC...ACTTAGAGACACTAGAA NM_009220_up_5000...
[18427]  5000 TTGATCCTCACTAAAAT...TTTTTTTTTTTGCCATT NM_001037748_up_5...
```

```
[18428]  5000 TGATCCTCACTAAAATT...TTTTTTTTTTTGCCATT NM_001037748_up_5...
[18429]  5000 CCATGTGGGTGTTAGAA...GCGCATCTATTCCACTT NM_001025241_up_5...
```

```
> m <- vmatchPattern(pattern, Mmusculus$upstream5000)
> which(countIndex(m) != 0)
```

```
[1]  2956  7540 10701 11387
```

```
> m[[2956]]
```

```
IRanges object:
  start  end width
1  3682 3691    10
```

5.3 Ambiguities

IUPAC extended letters can be used to express ambiguities in the pattern or in the subject of a search with `matchPattern`. This is controlled via the `fixed` argument of the function. If `fixed` is `TRUE` (the default), all letters in the pattern and the subject are interpreted litterally. If `fixed` is `FALSE`, IUPAC extended letters in the pattern and in the subject are interpreted as ambiguities e.g. `M` will match `A` or `C` and `N` will match any letter (the `IUPAC_CODE_MAP` named character vector gives the mapping between IUPAC letters and the set of nucleotides that they stand for). The most common use of this feature is to introduce wildcards in the pattern by replacing some of its letters with `N`s.

Exercise 10

1. *Search pattern* `GAACTTTGCCACTC` *in Mouse chromosome 1.*
2. *Repeat but this time allow the 2nd* `T` *in the pattern (6th letter) to match anything. Anything wrong?*
3. *Call* `matchPattern` *with* `fixed="subject"` *to work around this problem.*

```
> matchPattern("GAACTTTGCCACTC", Mmusculus$chr1)
```

```
  Views on a 197195432-letter DNAString subject
subject: NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN...AATTTGGTATTAAACTTAAAACTGGAATTC
views: NONE
```

```
> matchPattern("GAACTNTGCCACTC", Mmusculus$chr1)
```

```
  Views on a 197195432-letter DNAString subject
subject: NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN...AATTTGGTATTAAACTTAAAACTGGAATTC
views: NONE
```

```
> matchPattern("GAACTNTGCCACTC", Mmusculus$chr1, fixed = FALSE)
```

```
  Views on a 197195432-letter DNAString subject
subject: NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN...AATTTGGTATTAAACTTAAAACTGGAATTC
views:
        start       end width
[1] 180842072 180842085    14 [GAACTGTGCCACTC]
```

5.4 Masking

The *MaskedDNAString* container is dedicated to the storage of masked DNA sequences. As mentioned previously, you can use the unmasked accessor to turn a *MaskedDNAString* object into a *DNAString* object (the masks will be lost), or use the masks accessor to extract the masks (the sequence that is masked will be lost).

Each mask on a sequence can be active or not. Masks can be activated individually with:

```
> chr1 <- Mmusculus$chr1
> active(masks(chr1))["TRF"] <- TRUE
```

or all together with:

```
> active(masks(chr1)) <- TRUE
```

Some functions in Biostrings like alphabetFrequency or the string matching functions will skip the masked region when walking along a sequence with active masks.

Exercise 11

1. *What percentage of Mouse chromosome 1 is made of assembly gaps?*
2. *Check the alphabet frequency of Mouse chromosome 1 when only the AGAPS mask is active, when only the AGAPS and AMB masks are active. Compare with unmasked chromosome 1.*
3. *Try* `as(chr1 , "XStringViews")` *and* `gaps(as(chr1 , "XStringViews"))` *with different sets of active masks. How do you use this to display the contigs as views?*
4. *Activate all masks and find the occurences of an arbitrary DNA pattern in it. Compare to what you get with unmasked chromosome 1.*

```
> maskedratio(masks(Mmusculus$chr1)["AGAPS"])

[1] 0.02899639

> chr1 <- Mmusculus$chr1
> active(masks(chr1)) <- FALSE
> active(masks(chr1))["AGAPS"] <- TRUE
> chr1

  197195432-letter "MaskedDNAString" instance (# for masking)
seq: #################################...AGAATTTGGTATTAAACTTAAAACTGGAATTC
masks:
  maskedwidth  maskedratio active names
1     5717956 2.899639e-02   TRUE AGAPS
2          47 2.383422e-07  FALSE   AMB
3    84650265 4.292709e-01  FALSE    RM
4     4014755 2.035927e-02  FALSE   TRF
                              desc
1                    assembly gaps
2          intra-contig ambiguities
3                     RepeatMasker
4 Tandem Repeats Finder [period<=12]
all masks together:
  maskedwidth maskedratio
     90481616   0.4588424
all active masks together:
  maskedwidth maskedratio
      5717956  0.02899639
```

```
> alphabetFrequency(chr1)

       A        C        G        T        M        R        W        S 
56406566 39397656 39371416 56301791        0        0        0        0 
       Y        K        V        H        D        B        N        - 
       0        0        0        0        0        0       47        0 
       + 
       0 

> active(masks(chr1))["AMB"] <- TRUE
> alphabetFrequency(chr1)

       A        C        G        T        M        R        W        S 
56406566 39397656 39371416 56301791        0        0        0        0 
       Y        K        V        H        D        B        N        - 
       0        0        0        0        0        0        0        0 
       + 
       0 

> alphabetFrequency(unmasked(chr1))

       A        C        G        T        M        R        W        S 
56406566 39397656 39371416 56301791        0        0        0        0 
       Y        K        V        H        D        B        N        - 
       0        0        0        0        0        0  5718003        0 
       + 
       0 

> chr1 <- Mmusculus$chr1
> active(masks(chr1)) <- FALSE
> active(masks(chr1))["AGAPS"] <- TRUE
> as(chr1, "XStringViews")

  Views on a 197195432-letter DNAString subject
subject: NNNNNNNNNNNNNNNNNNNNNNNNNNNNNN...AATTTGGTATTAAACTTAAAACTGGAATTC
views:
         start       end    width
 [1]   3000001  22414948 19414948 [GAATTCTTTTCTATGAT...ATTTCCTTGTTATTTT]
 [2]  22415049  22423349     8301 [GGAAGCAGCAAATTCTG...AATATTAATTGTGGGG]
 [3]  22473350  24686638  2213289 [AGAGTGCTGTATCTGAA...TACTAGGAGAGAATTC]
 [4]  24736639  75102130 50365492 [GAATTCACTGGCTTTCC...ACCAGTGAAGAACTAG]
 [5]  75118131  78603540  3485410 [AGGCAGGACATTCAAAT...CTCTAGAAATCAAAGG]
 [6]  78603641  78604724     1084 [GTCTCTATGTGTGCGTG...GCTGGGATTAAAGGTG]
 [7]  78605670  78606725     1056 [AGGGTAAGGCACCCCCC...TAAATACTGAATTTTG]
 [8]  78607361  78610454     3094 [AGTTGAGTTGGGGAGGG...ATTCTCCTCTTGGGAC]
 [9]  78610738  85343678  6732941 [TCGTTCTCAGCTCTTCC...GGCAACAGTAGAATTC]
 ...       ...       ...      ... ...
[16] 185327811 193781121  8453311 [GTGTGTGTGTGTATGTC...ATGTGTGTGTAGTATG]
[17] 193781222 193785973     4752 [TTTTTTTTTTTTTTTTT...ACACACCACACACACC]
[18] 193786082 193825657    39576 [CACACACACACACACAC...CATTTAGAGGAAAGTC]
[19] 193875658 193877920     2263 [GGGCTCTACATGATCTG...AGTGCAATGCTCTGAC]
[20] 193878021 193883483     5463 [TGCAGGGGGAGGGAATG...GGAGGGAGGGAGGGAG]
[21] 193883584 193976498    92915 [AGGGAGGGAGGGAGGGA...GGGTGTCGCTTCCTGG]
```

```
[22] 193976831 193980538     3708 [AAAAAATCTACAACCCA...GCAGTGCGCGAGAAGA]
[23] 193987696 194007972    20277 [CTTACCTGTGGTTAAAT...CAAGAGGAGGAGGAGC]
[24] 194008894 197195432  3186539 [GAATTCTTTATGTATAC...CTTAAAACTGGAATTC]

> active(masks(chr1)) <- TRUE
> chr1

  197195432-letter "MaskedDNAString" instance (# for masking)
seq: ################################...################################
masks:
  maskedwidth  maskedratio active names
1     5717956 2.899639e-02   TRUE AGAPS
2          47 2.383422e-07   TRUE   AMB
3    84650265 4.292709e-01   TRUE    RM
4     4014755 2.035927e-02   TRUE   TRF
                               desc
1                     assembly gaps
2          intra-contig ambiguities
3                      RepeatMasker
4 Tandem Repeats Finder [period<=12]
all masks together:
  maskedwidth maskedratio
     90481616   0.4588424

> matchPattern("ACACACACACACACACACAC", chr1)

  Views on a 197195432-letter DNAString subject
subject: NNNNNNNNNNNNNNNNNNNNNNNNNNNNNN...AATTTGGTATTAAACTTAAAACTGGAATTC
views:
        start       end width
[1]  48952246  48952265    20 [ACACACACACACACACACAC]
[2] 100889001 100889020    20 [ACACACACACACACACACAC]
[3] 164163938 164163957    20 [ACACACACACACACACACAC]
[4] 176883480 176883499    20 [ACACACACACACACACACAC]

> matchPattern("ACACACACACACACACACAC", unmasked(chr1))

  Views on a 197195432-letter DNAString subject
subject: NNNNNNNNNNNNNNNNNNNNNNNNNNNNNN...AATTTGGTATTAAACTTAAAACTGGAATTC
views:
            start       end width
    [1]   3035551   3035570    20 [ACACACACACACACACACAC]
    [2]   3035553   3035572    20 [ACACACACACACACACACAC]
    [3]   3035555   3035574    20 [ACACACACACACACACACAC]
    [4]   3035557   3035576    20 [ACACACACACACACACACAC]
    [5]   3035559   3035578    20 [ACACACACACACACACACAC]
    [6]   3041036   3041055    20 [ACACACACACACACACACAC]
    [7]   3041038   3041057    20 [ACACACACACACACACACAC]
    [8]   3041040   3041059    20 [ACACACACACACACACACAC]
    [9]   3041042   3041061    20 [ACACACACACACACACACAC]
    ...       ...       ...   ... ...
[91680] 197189111 197189130    20 [ACACACACACACACACACAC]
```

```
[91681] 197189113 197189132    20 [ACACACACACACACACACAC]
[91682] 197189115 197189134    20 [ACACACACACACACACACAC]
[91683] 197189117 197189136    20 [ACACACACACACACACACAC]
[91684] 197189119 197189138    20 [ACACACACACACACACACAC]
[91685] 197189121 197189140    20 [ACACACACACACACACACAC]
[91686] 197189123 197189142    20 [ACACACACACACACACACAC]
[91687] 197189125 197189144    20 [ACACACACACACACACACAC]
[91688] 197189127 197189146    20 [ACACACACACACACACACAC]
```

In addition to the built-in masks, the user can put its own mask on a sequence. Two types of user-controlled masking are supported: *by content* or *by position*. The `maskMotif` function will mask the regions of a sequence that contain a motif specified by the user. The `Mask` constructor will return the mask made of the regions defined by the start and end locations specified by the user (like with the `Views` function).

5.5 Finding the hits of a large set of short motifs

Our own competitor to other fast alignment tools like MAQ or bowtie is the `matchPDict` function. Its speed is comparable to the speed of MAQ but it uses more memory than MAQ to align the same set of reads against the same genome. Here are some important differences between `matchPDict` and MAQ (or bowtie):

1. `matchPDict` ignores the quality scores,
2. it finds all the matches,
3. it fully supports 2 or 3 (or more) mismatching nucleotides anywhere in the reads (performance will decrease significantly though if the reads are not long enough),
4. it supports masking (masked regions are skipped),
5. it supports IUPAC ambiguities in the subject (useful for SNP detection).

The workflow with `matchPDict` is the following:

1. Preprocess the set of short reads with the `PDict` constructor.
2. Call `matchPDict` on it.
3. Query the *MIndex* object returned by `matchPDict`.

Exercise 12

1. *Preprocess* `dict0` *(obtained earlier from* `topReads.rda`*) with the* `PDict` *constructor.*
2. *Use this PDict object to find the (exact) hits of* `dict0` *in Mouse chromosome 1.*
3. *Use* `countIndex` *on the MIndex object returned by* `matchPDict` *to extract the nb of hits per read.*
4. *Which read has the highest number of hits? Display those hits as an XStringViews object. Check this result with a call to* `matchPattern`*.*
5. *You only got the hits that belong to the + strand. How would you get the hits that belong to the - strand?*
6. *Redo this analysis for inexact matches with at most 2 mismatches per read in the last 20 nucleotides.*

```
> pdict0 <- PDict(dict0)
> m <- matchPDict(pdict0, Mmusculus$chr1)
> Rle(countIndex(m))
```

```
  'integer' Rle instance of length 824 with 147 runs
  Lengths:  2 1 20 1 1 1 3 1 1 1 ...
  Values :  0 1523 0 52 0 54 0 50 0 51 ...

> which(countIndex(m) == max(countIndex(m)))

[1] 46

> pdict0[[46]]

  36-letter "DNAString" instance
seq: ACACACACACACACACACACACACACACACACACAC

> Views(unmasked(Mmusculus$chr1), start = start(m[[46]]),
+     end = end(m[[46]]))

  Views on a 197195432-letter DNAString subject
subject: NNNNNNNNNNNNNNNNNNNNNNNNNNNNNN...AATTTGGTATTAAACTTAAAACTGGAATTC
views:
            start       end width
    [1]   3041036   3041071    36 [ACACACACACACACACACACACACACACACACACAC]
    [2]   3041038   3041073    36 [ACACACACACACACACACACACACACACACACACAC]
    [3]   3041040   3041075    36 [ACACACACACACACACACACACACACACACACACAC]
    [4]   3041042   3041077    36 [ACACACACACACACACACACACACACACACACACAC]
    [5]   3041044   3041079    36 [ACACACACACACACACACACACACACACACACACAC]
    [6]   3041046   3041081    36 [ACACACACACACACACACACACACACACACACACAC]
    [7]   3041048   3041083    36 [ACACACACACACACACACACACACACACACACACAC]
    [8]   3220742   3220777    36 [ACACACACACACACACACACACACACACACACACAC]
    [9]   3220744   3220779    36 [ACACACACACACACACACACACACACACACACACAC]
    ...       ...       ...   ... ...
[25729] 197055223 197055258    36 [ACACACACACACACACACACACACACACACACACAC]
[25730] 197059731 197059766    36 [ACACACACACACACACACACACACACACACACACAC]
[25731] 197059733 197059768    36 [ACACACACACACACACACACACACACACACACACAC]
[25732] 197059735 197059770    36 [ACACACACACACACACACACACACACACACACACAC]
[25733] 197059737 197059772    36 [ACACACACACACACACACACACACACACACACACAC]
[25734] 197059739 197059774    36 [ACACACACACACACACACACACACACACACACACAC]
[25735] 197059741 197059776    36 [ACACACACACACACACACACACACACACACACACAC]
[25736] 197189109 197189144    36 [ACACACACACACACACACACACACACACACACACAC]
[25737] 197189111 197189146    36 [ACACACACACACACACACACACACACACACACACAC]

> matchPattern(pdict0[[46]], Mmusculus$chr1)

  Views on a 197195432-letter DNAString subject
subject: NNNNNNNNNNNNNNNNNNNNNNNNNNNNNN...AATTTGGTATTAAACTTAAAACTGGAATTC
views:
            start       end width
    [1]   3041036   3041071    36 [ACACACACACACACACACACACACACACACACACAC]
    [2]   3041038   3041073    36 [ACACACACACACACACACACACACACACACACACAC]
    [3]   3041040   3041075    36 [ACACACACACACACACACACACACACACACACACAC]
    [4]   3041042   3041077    36 [ACACACACACACACACACACACACACACACACACAC]
    [5]   3041044   3041079    36 [ACACACACACACACACACACACACACACACACACAC]
    [6]   3041046   3041081    36 [ACACACACACACACACACACACACACACACACACAC]
    [7]   3041048   3041083    36 [ACACACACACACACACACACACACACACACACACAC]
    [8]   3220742   3220777    36 [ACACACACACACACACACACACACACACACACACAC]
    [9]   3220744   3220779    36 [ACACACACACACACACACACACACACACACACACAC]
    ...       ...       ...   ... ...
[25729] 197055223 197055258    36 [ACACACACACACACACACACACACACACACACACAC]
[25730] 197059731 197059766    36 [ACACACACACACACACACACACACACACACACACAC]
[25731] 197059733 197059768    36 [ACACACACACACACACACACACACACACACACACAC]
[25732] 197059735 197059770    36 [ACACACACACACACACACACACACACACACACACAC]
[25733] 197059737 197059772    36 [ACACACACACACACACACACACACACACACACACAC]
[25734] 197059739 197059774    36 [ACACACACACACACACACACACACACACACACACAC]
[25735] 197059741 197059776    36 [ACACACACACACACACACACACACACACACACACAC]
[25736] 197189109 197189144    36 [ACACACACACACACACACACACACACACACACACAC]
[25737] 197189111 197189146    36 [ACACACACACACACACACACACACACACACACACAC]
```

```
> pdict1 <- PDict(reverseComplement(dict0))
> m1 <- matchPDict(pdict1, Mmusculus$chr1)
> Rle(countIndex(m1))

  'integer' Rle instance of length 824 with 152 runs
  Lengths:  2 1 20 1 1 1 3 1 1 1 ...
  Values :  0 1429 0 34 0 35 0 33 0 35 ...

> which(countIndex(m1) == max(countIndex(m1)))

[1] 433

> reverseComplement(pdict1[[433]])

  36-letter "DNAString" instance
seq: GTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGT

> pdict2 <- PDict(dict0, tb.end = 16)
> m2 <- matchPDict(pdict2, Mmusculus$chr1, max.mismatch = 2)
> all(countIndex(m2) >= countIndex(m))

[1] TRUE

> which(countIndex(m2) == max(countIndex(m2)))

[1] 90

> pdict0[[90]]

  36-letter "DNAString" instance
seq: TGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTG
```

6 Session Information

```
> toLatex(sessionInfo())

\begin{itemize}
  \item R version 2.9.0 Under development (unstable) (2009-01-19 r47650), \verb|i386-apple-darwin9.6.0|
  \item Locale: \verb|C/C/C/C/C/en_US.UTF-8|
  \item Base packages: base, datasets, graphics, grDevices,
    methods, stats, utils
  \item Other packages: Biostrings~2.11.26, BSgenome~1.11.9,
    BSgenome.Mmusculus.UCSC.mm9~1.3.11, IRanges~1.1.34
  \item Loaded via a namespace (and not attached): grid~2.9.0,
    lattice~0.17-20, Matrix~0.999375-18
\end{itemize}
```