An overview of the Biostrings/BSgenome framework

Hervé Pagès
Computational Biology Program
Fred Hutchinson Cancer Research Center
Biostrings

- Containers for representing large biological sequences (DNA/RNA/amino acids)
- Utilities for basic computations on sequences
- Tools for sequence matching and pairwise alignments

BSgenome data packages

- Full genomes stored in Biostrings containers
- Currently 16 organisms supported (Human, Mouse, Worm, Yeast, etc...)
- Facilities for supporting new genomes (BSgenomeForge)
Biostrings

Basic string containers

- Single sequence: XString (virtual class) and its direct extensions BString, DNAString, RNAString and AAString

- Set of sequences: XStringSet (virtual class) and its direct extensions BStringSet, DNAStringSet, RNAStringSet and AAStringSet

- Set of views on a sequence: XStringViews

- Masked sequence: MaskedXString (virtual class) and its direct extensions MaskedBString, MaskedDNAString, MaskedRNAString and MaskedAAString
Biostrings
Basic utilities

Extracting a subsequence: subseq()

```r
> library(BSgenome.Celegans.UCSC.ce2)

> chrI <- Celegans$chrI

> chrI
15080483-letter "DNAString" instance
seq: GCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCC...GGCTTAGGCTTAGGCTTAGGTTTAGGCTTAGGC

> subseq(chrI, start=1000, end=-1000)
15078485-letter "DNAString" instance
seq: ATTTTTCGGGTTTTTTGAAATGAATATCGTAGC...TTTAAACTCGTATCGGTTAACCAACCCTTTGGAT
```

IO: read.DNAStringSet(), write.XStringSet()

```r
> orfs <- read.DNAStringSet(file, "fasta") # loading some Yeast Open Reading Frames
Read 450 items

> orfs
A DNAStringSet instance of length 7

<table>
<thead>
<tr>
<th>width</th>
<th>seq</th>
<th>names</th>
</tr>
</thead>
<tbody>
<tr>
<td>5573</td>
<td>ACTTGTAAATATATCTTTTTA...ATCGACCTTTATTGGTAGAT</td>
<td>YAL001C TFC3 SGDI...</td>
</tr>
<tr>
<td>5825</td>
<td>TTCCAAGGCCGATGAATTCG...AAATTTTTTTCTATTCTTTT</td>
<td>YAL002W VPS8 SGDI...</td>
</tr>
<tr>
<td>2987</td>
<td>CTTCATGTAGCTGCCTCATTT...TACTCATGTAGCTGCCTCAT</td>
<td>YAL003W EFB1 SGDI...</td>
</tr>
<tr>
<td>3929</td>
<td>CACTCATATCGGGGGGTCTTTA...CCCAGAAAACGAAAAAGTAC</td>
<td>YAL005C SSA1 SGDI...</td>
</tr>
<tr>
<td>2648</td>
<td>AGAGAAAGAGTTTTCACTTCTTT...TAATTTATGTGTAACATAG</td>
<td>YAL007C ERP2 SGDI...</td>
</tr>
<tr>
<td>2597</td>
<td>GTGTCGGGGCCTCGCAGGCG...TTTGGCAAGATGTACTTTTT</td>
<td>YAL008W FUN14 SGD...</td>
</tr>
<tr>
<td>2780</td>
<td>CAAGATAATGTCAAGGTTTAGA...AAGGAAGAAAATACAC</td>
<td>YAL009W SP07 SGDI...</td>
</tr>
</tbody>
</table>
```
BString(), DNAString(), RNAString(), AAString()

```r
> dna <- DNAString("actttGtaa-NNYaA")
> dna
  15-letter "DNAString" instance
  seq: ACTTTGTAA-NNYAA

> RNAString(dna)
  15-letter "RNASTring" instance
  seq: ACUUUGUAA-NNYAA
```
**Biostrings**

**XStringSet constructors**

`BStringSet()`, `DNAStringSet()`, `RNAStringSet()`, `AAStringSet()`

```
> RNAStringSet(orfs)
  A RNAStringSet instance of length 7
  width seq                          names
[1]  5573 ACUUGUAAAAAUACUUUUA...AUCGACCUUUAUUGUUGAUAU YAL001C TFC3 SGDI...
[2]  5825 UUCCAAGGCCGAUGAAUUCG...AAAAUUAAUUUCUAUUUCUCUU YAL002W VPS8 SGDI...
[3]  2987 CUUCAUGUCAGCCUCGACUUU...UACUCAUGUAGCUGCCUCAAU YAL003W EFB1 SGDI...
[4]  3929 CACUCAUAUCGGGGGUUCUU...UACGAAACACGAAAAGUAC YAL005C SSA1 SGDI...
[5]  2648 AGAGAAAGAGUUCUCACUUCUU...UAAAUUUAGUGUGAACAUAG YAL007C ERP2 SGDI...
[6]  2597 GUGUCCGGGCCUCGCAGCAGCG...UUUGGCGAGAAUGUACUUUU YAL008W FUN14 SGD...
[7]  2780 CAAGAUAAGUCAAAUGUUAAG...AAGGAAAGAAAAAAUCAC YAL009W SP07 SGDI...
```

```
> RNAStringSet(orfs, end=12)
  A RNAStringSet instance of length 7
  width seq                          names
[1]    12 ACUUGUAAAAAUUUUUAAUUGUUGUUGAUAU YAL001C TFC3 SGDI...
[2]    12 UUCCAAGGCCGAUGAAUUCG...AAAAUUAAUUUCUAUUUCUCUU YAL002W VPS8 SGDI...
[3]    12 CUUCAUGUCAGCCUCGACUUU...UACUCAUGUAGCUGCCUCAAU YAL003W EFB1 SGDI...
[4]    12 CACUCAUAUCGGGGGUUCUU...UACGAAACACGAAAAGUAC YAL005C SSA1 SGDI...
[5]    12 AGAGAAAGAGUUCUCACUUCUU...UAAAUUUAGUGUGAACAUAG YAL007C ERP2 SGDI...
[6]    12 GUGUCCGGGCCUCGCAGCAGCG...UUUGGCGAGAAUGUACUUUU YAL008W FUN14 SGD...
[7]    12 CAAGAUAAGUCAAAUGUUAAG...AAGGAAAGAAAAAAUCAC YAL009W SP07 SGDI...
```

```
> AAStringSet(orfs)
Error in getXStringSubtypeConversionLookup(from_baseClass, baseClass) :
  incompatible XString/XStringSet subtypes
```
Biostrings

Basic transformations

**reverse()**, **complement()**, **reverseComplement()**, **translate()**

```
> x
 21-letter "DNAString" instance
seq: TCAACGTTGAATAGCGTACCG
> reverseComplement(x)
 21-letter "DNAString" instance
seq: CGGTACGCTATTCAACGTTGA
```

```
> translate(x)
 7-letter "AAString" instance
seq: STLNSVP
> translate(reverseComplement(x))
 7-letter "AAString" instance
seq: RYAIQR*
```

**Character translation: chartr()**

```
> library(BSgenome.Celegans.UCSC.ce2)
> chrII <- Celegans$chrII
> alphabetFrequency(chrII, baseOnly=TRUE)
   A   C   G   T   other
 4878194 2769208 2762193 4869710       3

> chrIIbis <- chartr("C", "T", chrII)
> chrIIbis
15279308-letter "DNAString" instance
seq: TTTAAGTTTAAGGTTTAAGTTTAAGGTTT...AGGTGAGATTTAGGTGTAGTTTAGTTAGT
> alphabetFrequency(chrIIbis, baseOnly=TRUE)
   A   C   G   T   other
 4878194  0 2762193  7638918       3
```
Biostrings

Counting letter occurrences

alphabetFrequency(), uniqueLetters()

```r
> yeast1
  230208-letter "DNAString" instance
  seq: CCACACCACACCCACACACCCACACACCACACC...GTGTGGGTGTGGTGTGGGTGTGGTGTGGTGG

> alphabetFrequency(yeast1)
  A   C   G   T   M   R   W   S   Y   K   V   H
  69830 44643 45765 69970     0     0     0     0     0     0     0
  D   B   N   -   +
    0     0     0     0

> alphabetFrequency(yeast1, baseOnly=TRUE)
  A   C   G   T other
  69830 44643 45765 69970

> uniqueLetters(yeast1)
[1] "A" "C" "G" "T"
```

dinucleotideFrequency(), trinucleotideFrequency(), oligonucleotideFrequency()

```r
> dinucleotideFrequency(yeast1)
   AA  AC  AG  AT  CA  CC  CG  CT  GA  GC  GG  GT
  23947 12493 13621 19769 15224  9218  7089 13112 14478  8910  9438 12938
   TA  TC  TG  TT
  16181 14021 15617 24151

> head(trinucleotideFrequency(yeast1))
   AAA  AAC  AAG  AAT  ACA  ACC
   8576  4105  4960  6306  3924  2849
```
Some useful predefined constants

> DNA_BASES
[1] "A" "C" "G" "T"

> DNA_ALPHABET
[1] "A" "C" "G" "T" "M" "R" "W" "S" "Y" "K" "V" "H" "D" "B" "N" "-" "+"

> IUPAC_CODE_MAP
       A  C  G  T  M  R  W  S  Y  K  V
   "A" "C" "G" "T" "AC" "AG" "AT" "CG" "CT" "GT" "ACG"
   "H" "D" "B" "N"
   "ACT" "AGT" "CGT" "ACGT"
Biostrings
Creating views on a DNAString object

With the Views() constructor

```r
> data(yeastSEQCHR1)
> dna <- DNAString(yeastSEQCHR1)
> dna
230208-letter "DNAString" instance
seq: CCACACCACACCACACCACACCACACCACACCACAC...GGTGTGTGGGTGTGGTGTGGGTGTGGTGTGTGTGGG
> Views(dna, start=c(1, 35, 777, 770), width=20)
Views on a 230208-letter DNAString subject
subject: CCACACCACACCACACCACACCACACCACACCACAC...TGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGGG
views:
<table>
<thead>
<tr>
<th>start</th>
<th>end</th>
<th>width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>35</td>
<td>54</td>
<td>20</td>
</tr>
<tr>
<td>777</td>
<td>796</td>
<td>20</td>
</tr>
<tr>
<td>770</td>
<td>789</td>
<td>20</td>
</tr>
</tbody>
</table>
```

With the successiveViews() constructor

```r
> successiveViews(dna, width=rep(20, 1+length(dna)/20))
Views on a 230208-letter DNAString subject
subject: CCACACCACACCACACCACACCACACCACACCACACAC...TGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGGG
views:
<table>
<thead>
<tr>
<th>start</th>
<th>end</th>
<th>width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>21</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>41</td>
<td>60</td>
<td>20</td>
</tr>
<tr>
<td>61</td>
<td>80</td>
<td>20</td>
</tr>
<tr>
<td>230141</td>
<td>230160</td>
<td>20</td>
</tr>
<tr>
<td>230161</td>
<td>230180</td>
<td>20</td>
</tr>
<tr>
<td>230181</td>
<td>230200</td>
<td>20</td>
</tr>
<tr>
<td>230201</td>
<td>230220</td>
<td>20</td>
</tr>
</tbody>
</table>
```
By turning the non-masked regions of a MaskedDNAString object into views
Biostrings

Creating views on a DNAString object

By turning the non-masked regions of a MaskedDNAString object into views

```r
> as(chr2, "Views")[1:10]
Views on a 243199373-letter DNAString subject
subject: NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN...NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
views:
  start  end    width
[1]  10001 3529312 3519312 [CGTATCCACACACCACACCA...GGAAATAGAGAAAACAGTAAGG]
[2] 3579313 5018788 1439476 [GATCATCACATTAAAAAGCTGTGG...TCTGGATTTATCCACCACTATA]
[3] 5118789 16279724 11160936 [TTTGGTGACGAGAAAGATCTTTG...CAATCAGAAACCGATGCGGACC]
[4] 16329725 21153113 4823389 [TTATATTGTATATATATGTTA...TTCTGGATGTTTATCCCACCATA]
[5] 21178114 31705550 10527437 [AGTTGGTATATATTGGAAGGGA...ATGCTATGCTGAGATGTATGT]
[6] 31705552 31725939 20388 [TTGATATGATATTGGTCTTTTTGG...TAAACATTGTTCTAATAAA]
[7] 31726791 31816827 90037 [CCTCTTCTCCTCCACATCAG...ACATGGAATAACTACGCGCCAT]
[8] 31816829 31816854 26 [AAAAATGATGAGTTCATGTCCTTTGT]
[9] 31816856 31816858 3 [AGG]
[10] 31816860 33092197 1275338 [CATGGATGACGTGACAGCATT...TTTTTTTTTTTTTTTTC]
```

```r
> subseq(chr2, start=31816827)
211382547-letter "MaskedDNAString" instance (# for masking)
seq: T#AAAAATGATGAGTTCATGTCCTTTGT#AGG#CATG...#AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

```r
> subseq(chr2, start=31816827)
211382547-letter "MaskedDNAString" instance (# for masking)
```

```r
> subseq(chr2, start=31816827)
211382547-letter "MaskedDNAString" instance (# for masking)
```
Biostrings

Inverting the views

With gaps()

```r
> v <- Views(dna, start=c(1, 35, 777), width=20)

> v
Views on a 230208-letter DNAString subject
subject: CCACACCACACCCACACACACACACACACACCACACCAC...TGTGTGGGTGTG
views:
    start end width
[1]   1   20   20 [CCACACCACACCCACACACC]
[2]  35   54   20 [CACACCACACACACCACAC]
[3] 777  796   20 [CAACAAATATACATAACAT]

> gaps(v)
Views on a 230208-letter DNAString subject
subject: CCACACCACACCCACACACACACACACACACCACACCAC...TGTGTGGGTGTG
views:
    start end width
[1]   21   34   14 [CACACACCACAC]
[2]  55   776  722 [CACACACATCTAACAATACCTAACCCTACTAC...CCTAAACATAAAATATTCTACTTTT]
[3] 797 230208 229412 [ATTGGCTTGTGGTAGCAACACTATCA...TGTG]
```
String matching

- A common problem: find all the occurrences (aka matches or hits) of a given pattern (typically short) in a (typically long) reference sequence (aka the subject)

  pattern: ATGAT

  subject: . . . ACGAGATTTATGATGATCGGATTATACGACACCAGATCGGCCATATGATTAC . . .

- **Exact** match = the sequence in the match is identical to the pattern

- **Inexact** match = some differences are allowed
  - Allow a maximum number of mismatching letters per match (max.mismatch argument)
  - Allow indels i.e. the “edit distance” between a match and the pattern must be < arbitrary value
  - IUPAC ambiguity letters in the pattern: interpret them literally or allow them to match the base letters they stand for?
Biostrings

String matching

- `matchPattern()`: 1 pattern, 1 reference sequence in the subject
  - pattern: ATGAT
  - subject: TTTACGAGATTTATGATGATCGGATTATAACAAG

- `vmatchPattern()`: 1 pattern, N reference sequences in the subject
  - pattern: ATGAT
  - subject:
    - ACGGAATTAGACCAT
    - TTTACGAGATTTATGATGATCGGATTATAACAAG
    - ...
    - TGGACAGGTACGGATGCGGT
**Biostrings**

**String matching**

- **matchPDict():** \( N \) patterns, 1 reference sequence in the subject

  - Patterns:
    - ATGAT
    - AGTTC
    - TTCAC
    - TGCTA
    - ...
    - GATGC

  - Subject:
    - TTTACGAGATTATGATGATCGGATTATACAAG

- **vmatchPDict():** \( N \) patterns, \( N \) reference sequences in the subject

  - Patterns:
    - ATGAT
    - AGTTC
    - TTCAC
    - TGCTA
    - ...
    - GATGC
    - ACGGAATTAGACCAT
    - TTTACGAGATTATGATGATCGGATTATACAAG
    - ...
    - TGGACAGGTAAGGATGCGGTTA

  - Subject:
    - ACGGAATTAGACCAT
    - TTTACGAGATTATGATGATCGGATTATACAAG
    - ...
    - TGGACAGGTAAGGATGCGGTTA

- The set of patterns (aka the *dictionary*) needs to be preprocessed first (stored in a PDict object)

- Preprocessing is fast but requires a lot of memory (e.g. < 40 sec. and 1-2 GB for 5-10 millions 36-mers)

- Some of the search parameters (e.g. exact/inexact matching) must be decided at preprocessing time

- [In the TODO pipe]
Biostrings

String matching examples

- **EXAMPLE 1: Using `matchPattern()`**

```r
> matchPattern("CAACTCCGATCG", chrII)
Views on a 15279308-letter DNAString subject
subject: CCTAAGCCTAAGCCTAAGC...GCTTAGGCTTAGGCTTAGT
views:

<table>
<thead>
<tr>
<th>start</th>
<th>end</th>
<th>width</th>
<th>pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>13490043</td>
<td>13490054</td>
<td>12</td>
<td>[CAACTCCGATCG]</td>
</tr>
</tbody>
</table>
```

```r
> matchPattern("CAACTCCGATCG", chrII, max.mismatch=1)
Views on a 15279308-letter DNAString subject
subject: CCTAAGCCTAAGCCTAAGC...GCTTAGGCTTAGGCTTAGT
views:

<table>
<thead>
<tr>
<th>start</th>
<th>end</th>
<th>width</th>
<th>pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>448786</td>
<td>448797</td>
<td>12</td>
<td>[CAAATCCGATCG]</td>
</tr>
<tr>
<td>1258669</td>
<td>1258680</td>
<td>12</td>
<td>[CAACTCCGATGG]</td>
</tr>
<tr>
<td>3340998</td>
<td>3341009</td>
<td>12</td>
<td>[CAGCTCCGATCG]</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>13490043</td>
<td>13490054</td>
<td>12</td>
<td>[CAACTCCGATCG]</td>
</tr>
<tr>
<td>13760610</td>
<td>13760621</td>
<td>12</td>
<td>[CAACTCCGATTG]</td>
</tr>
<tr>
<td>15213851</td>
<td>15213862</td>
<td>12</td>
<td>[CAACTCCGATCT]</td>
</tr>
</tbody>
</table>
```

```r
> matchPattern("CAACTCCGATCG", chrII, max.mismatch=1, with.indels=TRUE)
Views on a 15279308-letter DNAString subject
subject: CCTAAGCCTAAGCCTAAGC...GCTTAGGCTTAGGCTTAGT
views:

<table>
<thead>
<tr>
<th>start</th>
<th>end</th>
<th>width</th>
<th>pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>448786</td>
<td>448797</td>
<td>12</td>
<td>[CAAATCCGATCG]</td>
</tr>
<tr>
<td>861918</td>
<td>861928</td>
<td>11</td>
<td>[CAACTCCGATG]</td>
</tr>
<tr>
<td>1258669</td>
<td>1258679</td>
<td>11</td>
<td>[CAACTCCGATG]</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>13490043</td>
<td>13490054</td>
<td>12</td>
<td>[CAACTCCGATCG]</td>
</tr>
<tr>
<td>13760610</td>
<td>13760621</td>
<td>12</td>
<td>[CAACTCCGATTG]</td>
</tr>
<tr>
<td>15213851</td>
<td>15213861</td>
<td>11</td>
<td>[CAACTCCGATC]</td>
</tr>
</tbody>
</table>
```
String matching examples

- EXAMPLE 2: Using `matchPDict()` for exact matching of a constant-width dictionary

```r
library(hgu95av2probe)
dict0 <- DNAStringSet(hgu95av2probe)
dict0
```

```
A DNAStringSet instance of length 201800

<table>
<thead>
<tr>
<th>width</th>
<th>seq</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>TGGCTCCTGCTGAGGTCCCCTTTCC</td>
</tr>
<tr>
<td>25</td>
<td>GGCTGTGAATTCTGTACATATTTC</td>
</tr>
<tr>
<td>25</td>
<td>GCTTCAATTCCATTATGTTTTAATG</td>
</tr>
<tr>
<td>25</td>
<td>GCCGTTTGACAGAGCATGCTCTGCG</td>
</tr>
<tr>
<td>25</td>
<td>TGACAGAGCATGCTCTGCGTTGTTG</td>
</tr>
<tr>
<td>25</td>
<td>CTCTGCGTTGTTTACCAGCTTCTGCG</td>
</tr>
<tr>
<td>25</td>
<td>GGTTCACCAGCTTCTGCCCTCACA</td>
</tr>
<tr>
<td>25</td>
<td>TTCTGCCCCTCACATGCAGGGATT</td>
</tr>
<tr>
<td>25</td>
<td>CCTCACATGCAGGGATTAAACAA</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>25</td>
<td>GAGTGCCAATTCATGATGAGTCAG</td>
</tr>
<tr>
<td>25</td>
<td>ACACTGACACTTGTGCTCCTTGTCA</td>
</tr>
<tr>
<td>25</td>
<td>GATGGATAGTGAATGGATAGCCAG</td>
</tr>
<tr>
<td>25</td>
<td>CTCTGTTTCTCTGCAATCATCTCATCT</td>
</tr>
<tr>
<td>25</td>
<td>TTCTGCAAAGATGATGATGATCC</td>
</tr>
<tr>
<td>25</td>
<td>CAAAGATGATGATGATGATGATCC</td>
</tr>
<tr>
<td>25</td>
<td>GATGATGATGATGATGATGATCC</td>
</tr>
<tr>
<td>25</td>
<td>ATAGCTTTGTCAAAGATGATGATGATC</td>
</tr>
<tr>
<td>25</td>
<td>TTCTGCTTTGTCAAAGATGATGATGATC</td>
</tr>
<tr>
<td>25</td>
<td>CAAAGATGATGATGATGATGATCC</td>
</tr>
<tr>
<td>25</td>
<td>GTGCTCCTTGTCAAAGATGATGATC</td>
</tr>
</tbody>
</table>
```

1. Load the dictionary
**Biostrings**

**String matching examples**

- **EXAMPLE 2:** Using `matchPDict()` for **exact** matching of a **constant-width** dictionary

```
2. Preprocess the dictionary

   > pdict <- PDict(dict0)  # takes < 5 sec.
   > pdict
   TB_PDict object of length 201800 and width 25 (preprocessing algo="ACtree2")

3. Load the subject

   > library(BSgenome.Hsapiens.UCSC.hg18)
   > chr1 <- unmasked(Hsapiens$chr1)
   > chr1
   247249719-letter "DNAString" instance
   seq: TAACCCTAACCCTAACCCTAACCCTAACCC...NNNNNNNNNNNNNNNNNNNNNNNNNNNN

4. Call `matchPDict()`

   > m <- matchPDict(pdict, chr1)  # takes < 30 sec.
   > m
   201800-pattern MIndex object

5. Query the MIndex object

   > m[[700]]  # extract hits for pattern 700
   IRanges object:
   start       end width
   1 58787625  58787649    25
   2 110757441 110757465    25
   3 195332894 195332918    25
```
EXAMPLE 3: Allow a **small** number of mismatches **anywhere** in the patterns of a **constant-width** dictionary with PDict()/matchPDict()

---

**Preprocess the dictionary**

```r
> pdict <- PDict(dict0, max.mismatch=1)
```

---

**Call matchPDict()**

```r
> m <- matchPDict(pdict, chr1, max.mismatch=1)  # takes < 2 min.
> m
201800-pattern MIndex object
```

---

**Query the MIndex object**

```r
> endIndex(m)[[700]]
[1]  24302718  58787649 110757465 195332918
```
Biostrings

More string matching functions

- `countPattern()`, `vcountPattern()`, `countPDict()`, `vcountPDict()`): like the `match*()` functions but return the match count only (less memory needed)

Specialized string matching functions:
- `trimLRPatterns()`: trims left and/or right flanking patterns from sequences
- `matchLRPatterns()`: the matches are specified by a left pattern, a right pattern and a maximum distance between them
- `matchProbePair()`: finds amplicons given by a pair of primers (simulate PCR)
- `matchPWM()`: finds motifs described by a Position Weight Matrix (PWM)
- `findPalindromes()` / `findComplementedPalindromes()`
- `pairwiseAlignment()`: solves (Needleman-Wunsch) global alignment, (Smith-Waterman) local alignment, and (ends-free) overlap alignment problems

Support indels:
- pairwiseAlignment(), matchPattern()/countPattern(), vcountPattern()
- On the TODO list: vmatchPattern(), matchPDict(), countPDict(), ...
Biostrings

More string matching examples

• EXAMPLE 4: Using trimLRpatterns()

```r
> subject <- DNAStringSet(c("TGCTTGACGCAAAGA", "TTCTGCTTGGATCGG"))

> subject
A DNAStringSet instance of length 2
   width seq
[1]  15 TGCTTGACGCAAAGA
[2]  15 TTCTGCTTGGATCGG

> trimLRPatterns(Lpattern="TTCTGCTT", Rpattern="ATCGGAAG", subject)
A DNAStringSet instance of length 2
   width seq
[1]   9 GACGCAAAG
[2]   2 GG
```
Biostrings

More string matching examples

- EXAMPLE 5: Using `pairwiseAlignment()`

```R
> pairwiseAlignment("TTGCACCC", "TTGGATTGACCCA")
Global PairwiseAlignedFixedSubject (1 of 1)
pattern: [1] TTGCA-----CCC
subject: [1] TTGGATTGACCCA
score: -29.90804

> pairwiseAlignment("TTGCACCC", "TTGGATTGACCCA", type="global-local")
Global-Local PairwiseAlignedFixedSubject (1 of 1)
pattern: [1] TTGCACCC
subject: [6] TTG-ACCC
score: -0.1277071

> pairwiseAlignment("TTC", "ATTATTA", type="global-local")
Global-Local PairwiseAlignedFixedSubject (1 of 1)
pattern: [1] TTC
subject: [5] TTA
score: -2.596666
```
BSgenome data packages

- Full genome sequences stored in Biostrings containers / 1 genome per package
- 16 organisms / 27 packages in the current release (BioC 2.8)
- Most (but not all) packages contain sequences with built-in masks
- Use `available.genomes()` (from the BSgenome software package) to get the list

```r
> library(BSgenome)
> available.genomes()
[1] "BSgenome.Alyrata.JGI.v1"
[3] "BSgenome.Amellifera.UCSC.apiMel2"
[4] "BSgenome.Athaliana.TAIR.04232008"
[5] "BSgenome.Athaliana.TAIR.TAIR9"
[7] "BSgenome.Btaurus.UCSC.bosTau4"
[8] "BSgenome.Celegans.UCSC.ce2"
[9] "BSgenome.Celegans.UCSC.ce6"
[10] "BSgenome.Cfamiliaris.UCSC.canFam2"
[12] "BSgenome.Dmelanogaster.UCSC.dm3"
[13] "BSgenome.Drerio.UCSC.danRer5"
[14] "BSgenome.Drerio.UCSC.danRer6"
[15] "BSgenome.Drerio.UCSC.danRer7"
[16] "BSgenome.Ecoli.NCBI.20080805"
[17] "BSgenome.Gaculeatus.UCSC.gasAcu1"
[18] "BSgenome.Ggallus.UCSC.galGal3"
[19] "BSgenome.Hsapiens.UCSC.hg17"
[20] "BSgenome.Hsapiens.UCSC.hg18"
[21] "BSgenome.Hsapiens.UCSC.hg19"
[22] "BSgenome.Mmusculus.UCSC.mm8"
[23] "BSgenome.Mmusculus.UCSC.mm9"
[24] "BSgenome.Ptroglodytes.UCSC.panTro2"
[25] "BSgenome.Rnorvegicus.UCSC.rn4"
[26] "BSgenome.Scerevisiae.UCSC.sacCer1"
[27] "BSgenome.Scerevisiae.UCSC.sacCer2"
```
A BSgenome data package with no built-in masks

> library(BSgenome.Celegans.UCSC.ce2)

> Celegans
Worm genome

organism: Caenorhabditis elegans
provider: UCSC
provider version: ce2
release date: Mar. 2004
release name: WormBase v. WS120

single sequences (see '?seqnames'):
- chrI  chrII  chrIII  chrIV  chrV  chrX  chrM

multiple sequences (see '?mseqnames'):
- upstream1000  upstream2000  upstream5000

(use the '$' or '[[ operator to access a given sequence)

> Celegans$chrI
15080483-letter "DNAString" instance
seq: GCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAA...TTAGGCTTAGGCTTAGGCTTAGGTTTAGGCTTAGGC

> class(Celegans$chrI)
[1] "DNAString"
attr("package")
[1] "Biostrings"
BSgenome data packages

- `seqnames()`, `seqlengths()` (do not load the sequences)

```r
> seqnames(Celegans)
[1] "chrI"  "chrII"  "chrIII"  "chrIV"  "chrV"  "chrX"  "chrM"
> seqlengths(Celegans)
  chrI    chrII    chrIII    chrIV    chrV    chrX    chrM
  15080483  15279308  13783313  17493791  20922231  17718849    13794
```

- Use standard `lapply()`/`sapply()` to apply the same function to all the chromosomes

```r
> sapply(seqnames(Celegans),
          function(seqname)
            alphabetFrequency(Celegans[[seqname]], baseOnly=TRUE))
  chrI  chrII  chrIII  chrIV  chrV  chrX  chrM
     A  4838561  4878194  4444527  5711041  6749806  5746418  4335
     C  2697177  2769208  2449074  3034771  3711722  3119282  1225
     G  2693544  2762193  2466260  3017009  3700959  3118284  2055
     T  4851201  4869710  4423447  5730970  6759744  5734865  6179
     other  0       3       5       0       0       0     0
```

- Here all the sequences were loaded
- Applying `colSums()` to this matrix would give the same result as `seqlengths(Celegans)`
BSgenome data packages

A BSgenome data package with built-in masks

> library(BSgenome.Btaurus.UCSC.bosTau4)

> Btaurus
Cow genome

organism: Bos taurus
provider: UCSC
provider version: bosTau4
release date: Oct. 2007
release name: Baylor College of Medicine HGSC Btau_4.0

single sequences (see '?seqnames'):
  chr1  chr2  chr3  chr4  chr5  chr6  chr7  chr8  chr9  chr10 chr11
  chr12 chr13 chr14 chr15 chr16 chr17 chr18 chr19 chr20 chr21 chr22
  chr23 chr24 chr25 chr26 chr27 chr28 chr29 chrX  chrM

multiple sequences (see '?mseqnames'):
  chrUn.scaffolds  upstream1000  upstream2000  upstream5000

(use the '$' or '[[]' operator to access a given sequence)
ABased on the information provided, a BSgenome data package with built-in masks is demonstrated using the Btaurus package. The code snippet below illustrates how to access a sequence named Btaurus$chr1, showing the seq portion of the DNA string. The seq contains the characters TACCCCACTCACACTTATGGATAGATCAACTAAACA...TCCATTTTAGTTTATTTTTTTGTATGGTTAGAATACT.

The masks are presented as follows:

<table>
<thead>
<tr>
<th>Mask ID</th>
<th>Masked Width</th>
<th>Masked Ratio</th>
<th>Active Status</th>
<th>Mask Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11225171</td>
<td>0.00069676</td>
<td>TRUE</td>
<td>AGAPS assembly gaps</td>
</tr>
<tr>
<td>2</td>
<td>1098</td>
<td>0.00006815</td>
<td>TRUE</td>
<td>AMB intra-contig ambiguities</td>
</tr>
<tr>
<td>3</td>
<td>72217189</td>
<td>0.00044826</td>
<td>FALSE</td>
<td>RM RepeatMasker</td>
</tr>
<tr>
<td>4</td>
<td>314874</td>
<td>0.00001954</td>
<td>FALSE</td>
<td>TRF Tandem Repeats Finder [period&lt;=12]</td>
</tr>
</tbody>
</table>

All masks together:

- Masked Width: 83443591
- Masked Ratio: 0.5179414

All active masks together:

- Masked Width: 11226269
- Masked Ratio: 0.0696824

The class of Btaurus$chr1 is a MaskedDNAString, and the package version is Biostrings.

The masks can be either active or inactive. A mask can be active (the masked regions will be skipped during most computations) or inactive (the mask will be ignored). The user can toggle this.

The set of built-in masks is guaranteed to be the same for all the single sequences in a given package (same order and same active masks too).
**BSgenome data packages**

**Built-in mask names**

1. AGAPS: mask of assembly gaps
2. AMB: mask of intra-contig ambiguities
3. RM: mask of repeat regions as determined by the RepeatMasker software
4. TRF: mask of repeat regions as determined by the Tandem Repeats Finder software
   (where only repeats with period less than or equal to 12 were kept)

*Note that masks 2, 3 and 4 should never overlap with mask 1.*

```R
> unmasked(chr1)
  161106243-letter "DNAS" instance
  seq: TACCCACA...TCCATTTTAGTTTTTTGTATGGTAGAATACT

> uniqueLetters(unmasked(chr1))
[1] "A" "C" "G" "T" "N"

> uniqueLetters(chr1)
[1] "A" "C" "G" "T"

> uniqueLetters(gaps(chr1))
[1] "N"

> active(masks(chr1))[, "AMB"] <- FALSE

> uniqueLetters(chr1)
[1] "A" "C" "G" "T" "N"
```
SNP injection – part 1/3

> library(BSgenome.Hsapiens.UCSC.hg19)
> Hsapiens

Human genome

- organism: Homo sapiens (Human)
- provider: UCSC
- provider version: hg19
- release date: Feb. 2009
- release name: Genome Reference Consortium GRCh37

> available.SNPs()
[1] "SNPlocs.Hsapiens.dbSNP.20080617"
[2] "SNPlocs.Hsapiens.dbSNP.20090506"
[3] "SNPlocs.Hsapiens.dbSNP.20100427"
[4] "SNPlocs.Hsapiens.dbSNP.20101109"

> installed.SNPs()
[1] "SNPlocs.Hsapiens.dbSNP.20100427"

> SnpHsapiens <- injectSNPs(Hsapiens, "SNPlocs.Hsapiens.dbSNP.20100427")
BSgenome data packages

SNP injection – part 2/3

> SnpHsapiens
Human genome
|
| organism: Homo sapiens (Human)
| provider: UCSC
| provider version: hg19
| release date: Feb. 2009
| release name: Genome Reference Consortium GRCh37
| with SNPs injected from package: SNPlocs.Hsapiens.dbSNP.20100427
| ...

> SNPcount(SnpHsapiens)

<table>
<thead>
<tr>
<th>chr1</th>
<th>chr2</th>
<th>chr3</th>
<th>chr4</th>
<th>chr5</th>
<th>chr6</th>
<th>chr7</th>
<th>chr8</th>
<th>chr9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1369185</td>
<td>1422439</td>
<td>1186807</td>
<td>1191984</td>
<td>1064540</td>
<td>1040203</td>
<td>977275</td>
<td>919207</td>
<td>740516</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>859433</td>
<td>855225</td>
<td>822825</td>
<td>615382</td>
<td>543041</td>
<td>499101</td>
<td>556394</td>
<td>464686</td>
<td>482359</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>375259</td>
<td>450685</td>
<td>253480</td>
<td>244482</td>
<td>445596</td>
<td>53908</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
BSgenome data packages

SNP injection – part 3/3

> head(SNPlocs(SnpHsapiens, "chr1"))

<table>
<thead>
<tr>
<th>RefSNP_id</th>
<th>alleles_as_ambig</th>
<th>loc</th>
</tr>
</thead>
<tbody>
<tr>
<td>55998931</td>
<td>Y</td>
<td>10492</td>
</tr>
<tr>
<td>62636508</td>
<td>S</td>
<td>10519</td>
</tr>
<tr>
<td>58108140</td>
<td>R</td>
<td>10583</td>
</tr>
<tr>
<td>10218492</td>
<td>R</td>
<td>10828</td>
</tr>
<tr>
<td>10218493</td>
<td>R</td>
<td>10904</td>
</tr>
<tr>
<td>10218527</td>
<td>R</td>
<td>10927</td>
</tr>
</tbody>
</table>

> alphabetFrequency(Hsapiens$chr1)

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
<th>M</th>
<th>R</th>
<th>W</th>
<th>S</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>65570891</td>
<td>47024412</td>
<td>47016562</td>
<td>65668756</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>K</td>
<td>V</td>
<td>H</td>
<td>D</td>
<td>B</td>
<td>N</td>
<td>-</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

> alphabetFrequency(SnpHsapiens$chr1)

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
<th>M</th>
<th>R</th>
<th>W</th>
<th>S</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>65262079</td>
<td>46649521</td>
<td>46640527</td>
<td>65359967</td>
<td>144252</td>
<td>425323</td>
<td>112365</td>
<td>107535</td>
<td>423260</td>
</tr>
<tr>
<td>K</td>
<td>V</td>
<td>H</td>
<td>D</td>
<td>B</td>
<td>N</td>
<td>-</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>146506</td>
<td>1885</td>
<td>1915</td>
<td>1974</td>
<td>1859</td>
<td>1653</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Resources

- Man pages:

  > ?IUPAC_CODE_MAP
  > ?trimLRpatterns
  > ?injectSNPs
  > class?MaskedDNAString

  > ?inject
  No documentation for 'inject' in specified packages and libraries: you could try '??inject'
  > ??inject

  > chartr
  standardGeneric for "chartr" defined from package "base"

  function (old, new, x)
  standardGeneric("chartr")
  <environment: 0x28b84c8>
  Methods may be defined for arguments: old, new, x
  Use showMethods("chartr") for currently available ones.

  > showMethods("chartr")
  Function: chartr (package base)
  old="ANY", new="ANY", x="ANY"
  old="ANY", new="ANY", x="CompressedCharacterList"
  old="ANY", new="ANY", x="CompressedRleList"
  old="ANY", new="ANY", x="MaskedXString"
  old="ANY", new="ANY", x="Rle"
  old="ANY", new="ANY", x="SimpleCharacterList"
  old="ANY", new="ANY", x="SimpleRleList"
  old="ANY", new="ANY", x="XString"
  old="ANY", new="ANY", x="XStringSet"
  old="ANY", new="ANY", x="XStringViews"

  > `chartr,ANY,ANY,XString-method`
More resources

- *Pairwise Sequence Alignments* vignette in the Biostrings package
- *BSgenomeForge* vignette in the BSgenome software package to forge your own BSgenome data package
- Bioconductor mailing lists: http://bioconductor.org/help/mailing-list/