...ACGAGATTT**ATGATGAT**CGGATTATACGACACCGATCGGCCAT**ATGAT**TAC...

An overview of the Biostrings/BSgenome framework

Hervé Pagès
Computational Biology Program
Fred Hutchinson Cancer Research Center

...ACGAGATTT**ATGATGAT**CGGATTATACGACACCGATCGGCCAT**ATGAT**TAC...

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: Masked`XString` (virtual class) and its direct extensions Masked`BString`, Masked`DNAString`, Masked`RNAString` and Masked`AAString`

Biostrings

Basic utilities

Extracting a subsequence: `subseq()`

```
> 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...TTTAAACTCGTATCGGTTAACCAACCCTTGGAT
```

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

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

> orfs
  A DNAStringSet instance of length 7
    width seq                                               names
[1]  5573 ACTTGTAAATATATCTTTTA...ATCGACCTTATTGTTGATAT YAL001C TFC3 SGDI...
[2]  5825 TTCCAAGGCCGATGAATTCG...AAATTTTTTTCTATTCTCTT YAL002W VPS8 SGDI...
[3]  2987 CTTCATGTCAGCCTGCACTT...TACTCATGTAGCTGCCTCAT YAL003W EFB1 SGDI...
[4]  3929 CACTCATATCGGGGGTCTTA...CCCGAAACACGAAAAAGTAC YAL005C SSA1 SGDI...
[5]  2648 AGAGAAAGAGTTTCACTTCT...TAATTTATGTGTGAACATAG YAL007C ERP2 SGDI...
[6]  2597 GTGTCCGGGCCTCGCAGGCG...TTTTGGCAGAATGTACTTTT YAL008W FUN14 SGD...
[7]  2780 CAAGATAATGTCAAAGTTAG...AAGGAAGAAAAAAAAATCAC YAL009W SPO7 SGDI...
```

Biostrings

XString constructors

BString(), DNAString(), RNAString(), AAString()

```
> 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 ACUUGUAAAUAUAUCUUUUA...AUCGACCUUAUUGUUGAUAU YAL001C TFC3 SGDI...
[2]  5825 UUCCAAGGCCGAUGAAUUCG...AAAUUUUUUUCUAUUCUCUU YAL002W VPS8 SGDI...
[3]  2987 CUUCAUGUCAGCCUGCACUU...UACUCAUGUAGCUGCCUCAU YAL003W EFB1 SGDI...
[4]  3929 CACUCAUAUCGGGGGUCUUA...CCCGAAACACGAAAAAGUAC YAL005C SSA1 SGDI...
[5]  2648 AGAGAAAGAGUUUCACUUCU...UAAUUUAUGUGUGAACAUAG YAL007C ERP2 SGDI...
[6]  2597 GUGUCCGGGCCUCGCAGGCG...UUUUGGCAGAAUGUACUUUU YAL008W FUN14 SGD...
[7]  2780 CAAGAUAAUGUCAAAGUUAG...AAGGAAGAAAAAAAAAAUCAC YAL009W SPO7 SGDI...
```

```
> RNAStringSet(orfs, end=12)
  A RNAStringSet instance of length 7
    width seq                                               names
[1]    12 ACUUGUAAAUAU                                      YAL001C TFC3 SGDI...
[2]    12 UUCCAAGGCCGA                                      YAL002W VPS8 SGDI...
[3]    12 CUUCAUGUCAGC                                      YAL003W EFB1 SGDI...
[4]    12 CACUCAUAUCGG                                      YAL005C SSA1 SGDI...
[5]    12 AGAGAAAGAGUU                                      YAL007C ERP2 SGDI...
[6]    12 GUGUCCGGGCCU                                      YAL008W FUN14 SGD...
[7]    12 CAAGAUAAUGUC                                      YAL009W SPO7 SGDI...
```

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

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: TTTAAGTTTAAGTTTAAGTTTAAGTTTAAGTTT...AGGTTGAGATTTAGGTTTAGGTTTAGGTTTAGT
> alphabetFrequency(chrIIbis, baseOnly=TRUE)
      A       C       G       T   other
4878194       0 2762193 7638918       3
```

Biostrings

Counting letter occurences

alphabetFrequency(), uniqueLetters()

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

> 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     0
    D     B     N     -     +
    0     0     0     0     0

> alphabetFrequency(yeast1, baseOnly=TRUE)
    A     C     G     T other
69830 44643 45765 69970     0

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

dinucleotideFrequency(), trinucleotideFrequency(), oligonucleotideFrequency()

```
> 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
```

Biostrings

Predefined constants

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"
```

Creating views on a DNAString object

With the `Views()` constructor

```
> data(yeastSEQCHR1)
> dna <- DNAString(yeastSEQCHR1)
> dna
  230208-letter "DNAString" instance
seq: CCACACCACACCCACACACCCACACACCACACCACAC...GGTGTGTGGGTGTGGTGTGGGTGTGGTGTGTGTGGG
> Views(dna, start=c(1, 35, 777, 770), width=20)
  Views on a 230208-letter DNAString subject
subject: CCACACCACACCCACACACCCACACACCACACCAC...TGTGTGGGTGTGGTGTGGGTGTGGTGTGTGTGGG
views:
    start end width
[1]     1  20    20 [CCACACCACACCCACACACC]
[2]    35  54    20 [CACACCACACCACACCCACA]
[3]   777 796    20 [CAACAATAATACATAAACAT]
[4]   770 789    20 [TACTTTTCAACAATAATACA]
```

With the successive`Views()` constructor

```
> successiveViews(dna, width=rep(20, 1+length(dna)/20))
  Views on a 230208-letter DNAString subject
subject: CCACACCACACCCACACACCCACACACCACACCAC...TGTGTGGGTGTGGTGTGGGTGTGGTGTGTGTGGG
views:
         start    end width
    [1]      1     20    20 [CCACACCACACCCACACACC]
    [2]     21     40    20 [CACACACCACACCACACACC]
    [3]     41     60    20 [ACACCACACCCACACACACA]
    [4]     61     80    20 [CATCCTAACACTACCCTAAC]
    ...    ...    ...   ... ...
[11508] 230141 230160    20 [TGTGGGTGTGGGTGTGGGTG]
[11509] 230161 230180    20 [TGGGTGTGGTGTGGTGTGTG]
[11510] 230181 230200    20 [GGTGTGGTGTGGGTGTGGTG]
[11511] 230201 230220    20 [TGTGTGGG            ]
```

Biostrings

Creating views on a DNAString object

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

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

> chr2 <- Hsapiens$chr2

> chr2
  243199373-letter "MaskedDNAString" instance (# for masking)
seq: ####################################...####################################
masks:
  maskedwidth  maskedratio active names                                desc
1     4992000 2.052637e-02   TRUE AGAPS                        assembly gaps
2        2855 1.173934e-05   TRUE   AMB             intra-contig ambiguities
3   114946036 4.726412e-01  FALSE    RM                          RepeatMasker
4     1738758 7.149517e-03  FALSE   TRF Tandem Repeats Finder [period<=12]
all masks together:
  maskedwidth maskedratio
    120030016   0.4935457
all active masks together:
  maskedwidth maskedratio
      4994855  0.02053811
```

Biostrings

Creating views on a DNAString object

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

```
> as(chr2, "Views")[1:10]
  Views on a 243199373-letter DNAString subject
subject: NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN...NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
views:
        start      end    width
 [1]    10001  3529312  3519312 [CGTATCCCACACACCACACCCA...GGAAATAGAGAAAACAGTAAGG]
 [2]  3579313  5018788  1439476 [GATCATCATTAAAAAGCTGTGG...TCTGGATGTTTATCCCACCATA]
 [3]  5118789 16279724 11160936 [TTTGTTGAGCGAAAGATCTTGT...CAATCAGAAACCCGATGGGACC]
 [4] 16329725 21153113  4823389 [TTATATTGTATATAATATGATA...GGAAAGCCCCGTGAGGGAAGCG]
 [5] 21178114 31705550 10527437 [AGTTGTGATATATTGGAAGGGA...ATGCTATGGTCTGAGAGTATGT]
 [6] 31705552 31725939    20388 [TGGTATGATTTGGTTCTTTTGG...TAAACATTGTTCTAATAATAAA]
 [7] 31726791 31816827    90037 [CCTCTTCTCCTCCACACATCAC...ACATGGAATAACTACGCGCCAT]
 [8] 31816829 31816854       26 [AAAAATGATGAGTTCATGTCCTTTGT]
 [9] 31816856 31816858        3 [AGG]
[10] 31816860 33092197  1275338 [CATGGATGAAGCTGGAAACCAT...TTTTTTTTTTTTTTTTTTTCC]
```

```
> subseq(chr2, start=31816827)
  211382547-letter "MaskedDNAString" instance (# for masking)
seq: T#AAAAATGATGAGTTCATGTCCTTTGT#AGG#CATG...####################################
masks:
  maskedwidth  maskedratio active names                                desc
1     4757000 2.250422e-02   TRUE AGAPS                       assembly gaps
2        2003 9.475711e-06   TRUE   AMB           intra-contig ambiguities
3   100276450 4.743838e-01  FALSE    RM                        RepeatMasker
4     1493678 7.066231e-03  FALSE   TRF Tandem Repeats Finder [period<=12]
all masks together:
  maskedwidth maskedratio
    105110386    0.497252
all active masks together:
  maskedwidth maskedratio
      4759003   0.0225137
```

Biostrings

Inverting the views

With gaps()

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

> v
  Views on a 230208-letter DNAString subject
subject: CCACACCACACCCACACACCCACACACCACACCAC...TGTGTGGGTGTGGTGTGGGTGTGGTGTGTGTGGG
views:
    start end width
[1]     1  20    20 [CCACACCACACCCACACACC]
[2]    35  54    20 [CACACCACACCACACCCACA]
[3]   777 796    20 [CAACAATAATACATAAACAT]

> gaps(v)
  Views on a 230208-letter DNAString subject
subject: CCACACCACACCCACACACCCACACACCACACCAC...TGTGTGGGTGTGGTGTGGGTGTGGTGTGTGTGGG
views:
    start    end  width
[1]    21     34     14 [CACACACCACACCA]
[2]    55    776    722 [CACACACATCCTAACACTACCCTAAC...CCTAAACATAAAAATATTCTACTTTT]
[3]   797 230208 229412 [ATTGGCTTGTGGTAGCAACACTATCA...TGTGGTGTGGGTGTGGTGTGTGTGGG]
```

Biostrings

String matching

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

pattern: ATGAT

subject:

```
...ACGAGATTTATGATGATCGGATTATACGACACCGATCGGCCATATGATTAC...
               -------                      -----
```

- *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: `TTTACGAGATTTATGATGATCGGATTATACAAG`

- **`vmatchPattern()`**: **1** pattern, **N** reference sequences in the subject

  pattern: `ATGAT`

  subject:

  ```
  ACGGAATTAGACCAT
  TTTACGAGATTTATGATGATCGGATTATACAAG
  ...
  TGGACAGGTACGTAGGATGCGGTTA
  ```

Biostrings

String matching

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

patterns:

```
ATGAT
AGTTC
TTCAC
TGCTA
...
GATGC
```

subject:

```
TTTACGAGATTTATGATGATCGGATTATACAAG
```

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

patterns:

```
ATGAT
AGTTC
TTCAC
TGCTA
...
GATGC
```

subject:

```
ACGGAATTAGACCAT
TTTACGAGATTTATGATGATCGGATTATACAAG
...
TGGACAGGTACGTAGGATGCGGTTA
```

Biostrings

String matching examples

- EXAMPLE 1: Using `matchPattern()`

```
> matchPattern("CAACTCCGATCG", chrII)
  Views on a 15279308-letter DNAString subject
subject: CCTAAGCCTAAGCCTAAGC...GCTTAGGCTTAGGCTTAGT
views:
       start      end width
[1] 13490043 13490054    12 [CAACTCCGATCG]
> matchPattern("CAACTCCGATCG", chrII, max.mismatch=1)
  Views on a 15279308-letter DNAString subject
subject: CCTAAGCCTAAGCCTAAGC...GCTTAGGCTTAGGCTTAGT
views:
        start      end width
 [1]   448786   448797    12 [CAAATCCGATCG]
 [2]  1258669  1258680    12 [CAACTCCGATGG]
 [3]  3340998  3341009    12 [CAGCTCCGATCG]
 ...      ...      ...   ... ...
[11] 13490043 13490054    12 [CAACTCCGATCG]
[12] 13760610 13760621    12 [CAACTCCGATTG]
[13] 15213851 15213862    12 [CAACTCCGATCT]
```

```
> matchPattern("CAACTCCGATCG", chrII, max.mismatch=1, with.indels=TRUE)
  Views on a 15279308-letter DNAString subject
subject: CCTAAGCCTAAGCCTAAGC...GCTTAGGCTTAGGCTTAGT
views:
        start      end width
 [1]   448786   448797    12 [CAAATCCGATCG]
 [2]   861918   861928    11 [CAACTCCGATG]
 [3]  1258669  1258679    11 [CAACTCCGATG]
 ...      ...      ...   ... ...
[36] 13490043 13490054    12 [CAACTCCGATCG]
[37] 13760610 13760621    12 [CAACTCCGATTG]
[38] 15213851 15213861    11 [CAACTCCGATC]
```

Biostrings

String matching examples

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

1. Load the dictionary

```
> library(hgu95av2probe)

> dict0 <- DNAStringSet(hgu95av2probe)

> dict0
  A DNAStringSet instance of length 201800
         width seq
     [1]    25 TGGCTCCTGCTGAGGTCCCCTTTCC
     [2]    25 GGCTGTGAATTCCTGTACATATTTC
     [3]    25 GCTTCAATTCCATTATGTTTTAATG
     [4]    25 GCCGTTTGACAGAGCATGCTCTGCG
     [5]    25 TGACAGAGCATGCTCTGCGTTGTTG
     [6]    25 CTCTGCGTTGTTGGTTTCACCAGCT
     [7]    25 GGTTTCACCAGCTTCTGCCCTCACA
     [8]    25 TTCTGCCCTCACATGCACAGGGATT
     [9]    25 CCTCACATGCACAGGGATTTAACAA
     ...   ... ...
[201792]    25 GAGTGCCAATTCGATGATGAGTCAG
[201793]    25 ACACTGACACTTGTGCTCCTTGTCA
[201794]    25 CAATTCGATGATGAGTCAGCAACTG
[201795]    25 GACTTTCTGAGGAGATGGATAGCCT
[201796]    25 AGATGGATAGCCTTCTGTCAAAGCA
[201797]    25 ATAGCCTTCTGTCAAAGCATCATCT
[201798]    25 TTCTGTCAAAGCATCATCTCAACAA
[201799]    25 CAAAGCATCATCTCAACAAGCCCTC
[201800]    25 GTGCTCCTTGTCAACAGCGCACCCA
```

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...NNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
```

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
```

Biostrings

String matching examples

- EXAMPLE 3: Allow a **small** number of mismatches **anywhere** in the patterns of a **constant-width** dictionary with `PDict()/matchPDict()`

Preprocess the dictionary

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

Call matchPDict()

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

Query the MIndex object

```
> 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()`

```
> 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 `pairwiseAligment()`

```
> 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
- Naming convention: `BSgenome.`*Organism*`.`*Provider*`.`*BuildVersion*
- Use **`available.genomes()`** (from the BSgenome software package) to get the list

```
> library(BSgenome)
> available.genomes()
 [1] "BSgenome.Alyrata.JGI.v1"
 [2] "BSgenome.Amellifera.BeeBase.assembly4"
 [3] "BSgenome.Amellifera.UCSC.apiMel2"
 [4] "BSgenome.Athaliana.TAIR.04232008"
 [5] "BSgenome.Athaliana.TAIR.TAIR9"
 [6] "BSgenome.Btaurus.UCSC.bosTau3"
 [7] "BSgenome.Btaurus.UCSC.bosTau4"
 [8] "BSgenome.Celegans.UCSC.ce2"
 [9] "BSgenome.Celegans.UCSC.ce6"
[10] "BSgenome.Cfamiliaris.UCSC.canFam2"
[11] "BSgenome.Dmelanogaster.UCSC.dm2"
[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"
```

BSgenome data packages

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)

```
> 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

```
> 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)
```

BSgenome data packages

A BSgenome data package with built-in masks

```
> Btaurus$chr1
  161106243-letter "MaskedDNAString" instance (# for masking)
seq: TACCCCACTCACACTTATGGATAGATCAACTAAACA...TCCATTTTAGTTTATTTTTTTGTATGGTAGAATACT
masks:
  maskedwidth  maskedratio active names                                desc
1    11225171 6.967558e-02   TRUE AGAPS                    assembly gaps
2        1098 6.815378e-06   TRUE   AMB          intra-contig ambiguities
3    72217189 4.482582e-01  FALSE    RM                        RepeatMasker
4      314874 1.954449e-03  FALSE   TRF Tandem Repeats Finder [period<=12]
all masks together:
  maskedwidth maskedratio
     83443591   0.5179414
all active masks together:
  maskedwidth maskedratio
     11226269   0.0696824

> class(Btaurus$chr1)
[1] "MaskedDNAString"
attr(,"package")
[1] "Biostrings"

> masknames(Btaurus)
[1] "AGAPS" "AMB"   "RM"    "TRF"
```

- 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.

```
> unmasked(chr1)
  161106243-letter "DNAString" instance
seq: TACCCCACTCACACTTATGGATAGATCAACTAAACA...TCCATTTTAGTTTATTTTTTTGTATGGTAGAATACT

> 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"
```

BSgenome data packages

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)
   chr1    chr2    chr3    chr4    chr5    chr6    chr7    chr8    chr9
1369185 1422439 1186807 1191984 1064540 1040203  977275  919207  740516
  chr10   chr11   chr12   chr13   chr14   chr15   chr16   chr17   chr18
 859433  855225  822825  615382  543041  499101  556394  464686  482359
  chr19   chr20   chr21   chr22    chrX    chrY
 375259  450685  253480  244482  445596   53908
```

BSgenome data packages

SNP injection – part 3/3

```
> head(SNPlocs(SnpHsapiens, "chr1"))
  RefSNP_id alleles_as_ambig   loc
1  55998931                Y 10492
2  62636508                S 10519
3  58108140                R 10583
4  10218492                R 10828
5  10218493                R 10904
6  10218527                R 10927

> alphabetFrequency(Hsapiens$chr1)
       A        C        G        T        M        R        W        S        Y
65570891 47024412 47016562 65668756        0        0        0        0        0
       K        V        H        D        B        N        -        +
       0        0        0        0        0        0        0        0

> alphabetFrequency(SnpHsapiens$chr1)
       A        C        G        T        M        R        W        S        Y
65262079 46649521 46640527 65359967   144252   425323   112365   107535   423260
       K        V        H        D        B        N        -        +
  146506     1885     1915     1974     1859     1653        0        0
```

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`
```

Documentation for a particular method can be hard to find! :-/

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/`