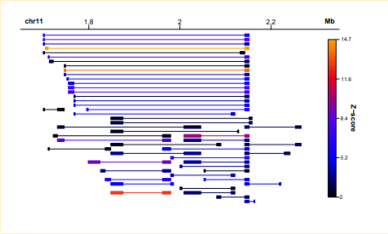
roxygen2

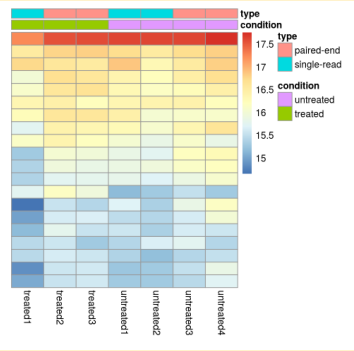
- Might be useful for final project
- Automatically generates NAMESPACE and/or Rd files for your packages
- You document your functions using comments in your code
- Be sure to still document your functions thoroughly if you use it

Introduction to

Lecture 12, CPSC 499 Fall 2018



What is Bioconductor?



- A collection of R packages for bioinformatics, all designed to work together
- The biggest R package repository outside of CRAN
- Started in 2001 for microarray data, but has expanded to use for sequencing and other types
- New versions for all packages released every six months, to coincide with releases of new R versions
- Also has packages containing genome sequences and annotations (not for many crops though)

Installing Bioconductor packages

- If you don't already have `BiocManager` installed, you can get it from CRAN with `install.packages`
- In the `BiocManager` package, there is a function called `install` that you can then use to install Bioconductor packages
- `BiocManager::install` in case you have other packages with an `install` function
- If it asks to update packages, usually best to update all (`a`)
- If some packages can't be updated, restart R and install just those packages first
- Ignore any references to `biocLite` in old tutorials (this is the older installer)
- Load with `library` like a normal package

Working with sequences: the `Biostrings` package

- `DNAString`, `RNAString`, and `AAString` classes can store a single DNA, RNA, or amino acid sequence
- `DNAStringSet`, `RNAStringSet`, and `AAStringSet` if you want to have a vector of sequences
- Constructor functions accept character vectors of your sequences
- `DNAStringSetList` etc. if you need a collection of collections of sequences

```
> test <- DNAStringSet(c("ATG", "CAG", "TAG"))
> test
  A DNAStringSet instance of length 3
    width seq
[1]     3 ATG
[2]     3 CAG
[3]     3 TAG
```

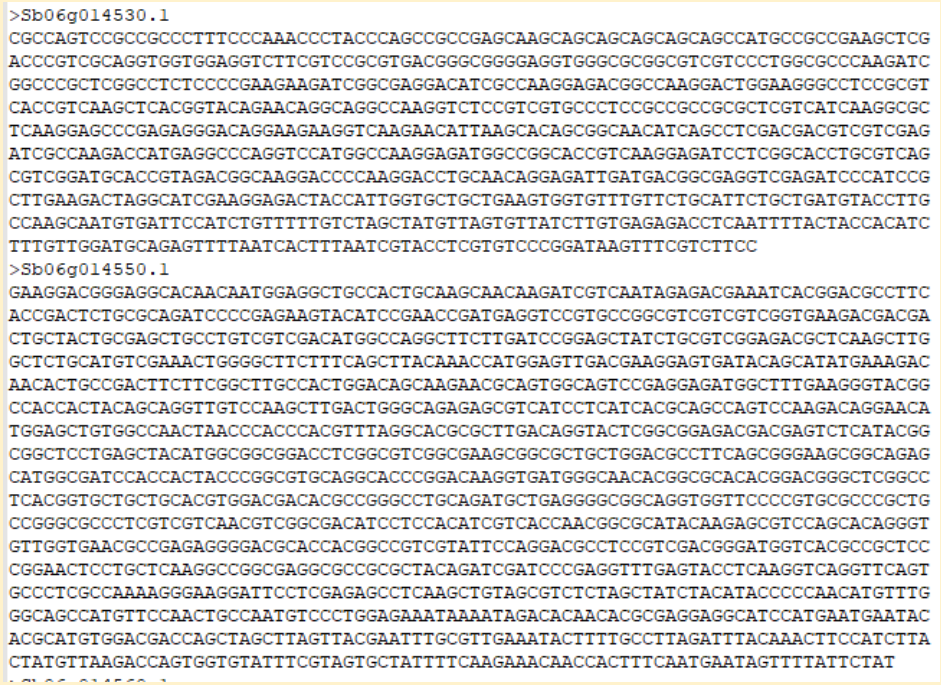
Importing sequences into Biostrings

- `readDNAStringSet`, `readRNAStringSet`, `readAAStringSet`
- These can read files in FASTA or FASTQ format

- To export to same file format, use `writeXStringSet`

FASTA format

- Can store one or multiple sequences
- DNA, RNA, or protein
- Every sequence has a comment line starting with > , usually with a name of the sequence



- The sequence itself can be on one or multiple lines after that
- IUPAC codes for nucleotides and amino acids

Typical uses for FASTA files

- Whole genome sequence; one entry per chromosome or contig
- Collection of genes, transcripts, or proteins
- Sequences downloaded from genome browsers
- Sequence alignments

```
>aligned seq 1
ATG---CCCGATTAG
>aligned seq 2
ATGATTCTCGATTGG
```

FASTQ format

- Stores DNA sequences with a quality score for each nucleotide (probability that it is incorrect)
- Typically output from NGS (next-generation sequencing) technology
- Four lines per sequencing read

Comment line starting with @ containing technical info

Read 1

```
@K00317:101:HLHL5BBXX:6:1101:12215:1384 1:N:0:16
TCCGAGAACCAACATCAACAACAACCAAAGTGTCTCCGAGTAGTAGAAGCAGCACCGTCGCCAGGACCAAGTGACAGCATGCTCCCTC
+
JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ
```

Read 2

```
@K00317:101:HLHL5BBXX:6:1101:12905:1384 1:N:0:16
TAATGTGTATCAATACCTTCTTGTAGATATTTCAACATGTGCTTATTGATTTGCCTGGCCTAGCATTCTTCACAACCTACGCTCTGCTA
+
JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJFJJJJJJJFJJJJJJJJJJJJJJJJJJFJJJJJJJFFF
```

Read 3

```
@K00317:101:HLHL5BBXX:6:1101:13291:1384 1:N:0:16
CCCGTAGCGCGGAAAAGCGGATTATTCCTCTCCCCCTAGCTAGCCCCTGCGGGCATCGGCGAAGCTCGTCCATCCAAAGTGCTCGGC
+
JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJFJJFJJJJJJJJJJJFAJJJJJJJJJJJJJFJJJJJ
```

Sequence

Quality scores

Working with XStringSet objects

- See https://bioconductor.org/packages/release/bioc/vignettes/Biostrings/inst/doc/BiostringsQuickOverview.pdf for quick reference
- Can do indexing, `length`, `nchar`, `match`, `==`, etc. like it was a character vector
- `reverseComplement`, `translate` functions for DNAStringSet objects

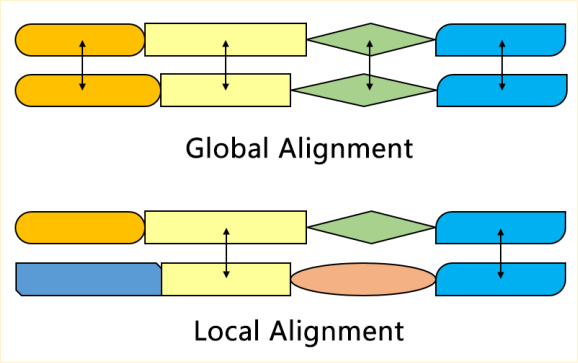
Pattern matching with Biostrings

- `grep` works like it does with character vectors, if you want to use regular expressions to search for a sequence
- `matchPattern` if you want to use IUPAC ambiguity codes, allow for mismatches and indels
- To search `XString`, use `matchPattern`. To search `XStringSet`, use `vmatchPattern`.
- `countPattern` and `vcountPattern` also available
- `matchPDict` to match many patterns against a reference sequence

Mini exercise

- Import the FASTA file from Lab 4 using `readAAStringSet`
- Search for the amino acid pattern “HSF” using `matchPattern`

Pairwise sequence alignment



- Needleman-Wunsch global alignment
  - Assumes two sequences should align along their entirety (like protein sequences of two closely-related genes)
  - Processing time proportional to length of seq 1 * length of seq 2
- Smith-Waterman local alignment
  - Just looks for local aligning regions within two sequences (like two related genes with divergent introns, or primer location with a gene)
  - Needs more processing time

Image: Yz cs5160 on Wikimedia Commons

Pairwise sequence alignment cont'd

- Overlap alignments
  - For finding joins between sequences, like when doing sequence assembly

- All three are adjustable in terms of penalties
  - How much should gaps be avoided
  - Are some letters considered more similar to each other than others (e.g. hydrophobic vs. polar and charged amino acids)

Performing pairwise alignments in Bioconductor

- `pairwiseAlignment` function
- One or more sequences for "pattern"
- One sequence for "subject"
- Returns an object of `PairwiseAlignments` class
- Can export with `writePairwiseAlignments`
- Functions like `indel` to extract particular information

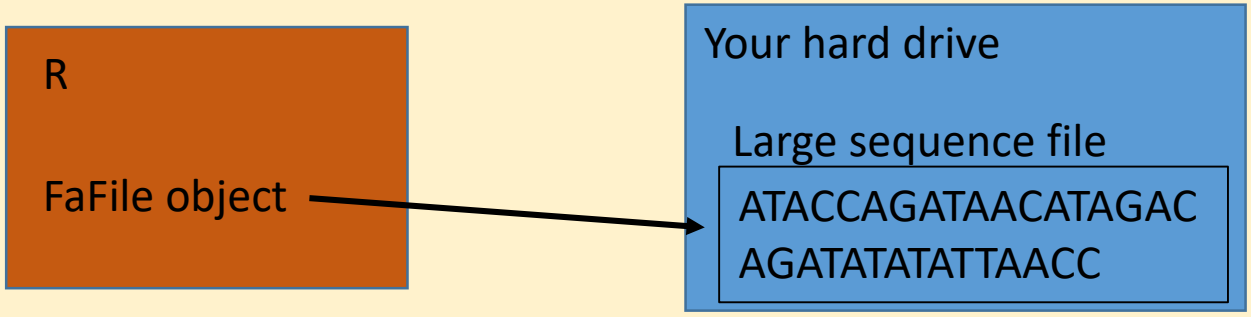
Mini exercise

- For one of the two TFL1 sequences, extract a smaller sequence using `subseq`
- Perform local and global pairwise alignment and compare the results

Working with whole genome sequences and annotations

`FaFile` objects

- From package Rsamtools
- For working with FASTA file without loading the whole thing into memory
- Build index with `indexFa` – this makes .fai file that speeds up access to the FASTA file
- Generally how you want to work with whole genome sequences



Working with `FaFile` objects

- `seqinfo` – get a summary of sequence names and lengths (`SeqInfo` S4 class)
  - `seqnames` and `seqlengths` to extract just that info
- `countFa` – how many sequences are in file
- `getSeq` or `scanFa` – import specific sequences as `XStrings`
  - Specify sequences with `GRanges` object (see next slide...)

GRanges

- From GenomicRanges package
- Used for specifying any location in the genome
  - Whole chromosome
  - Gene location
  - Mutation
  - Etc.
-
```
GRanges(c("Chr01", "Chr02", "Chr02"),
        IRanges(c(200, 110046, 3005),
                c(340, 115077, 4200)))
```

seqnames

Start positions

End positions

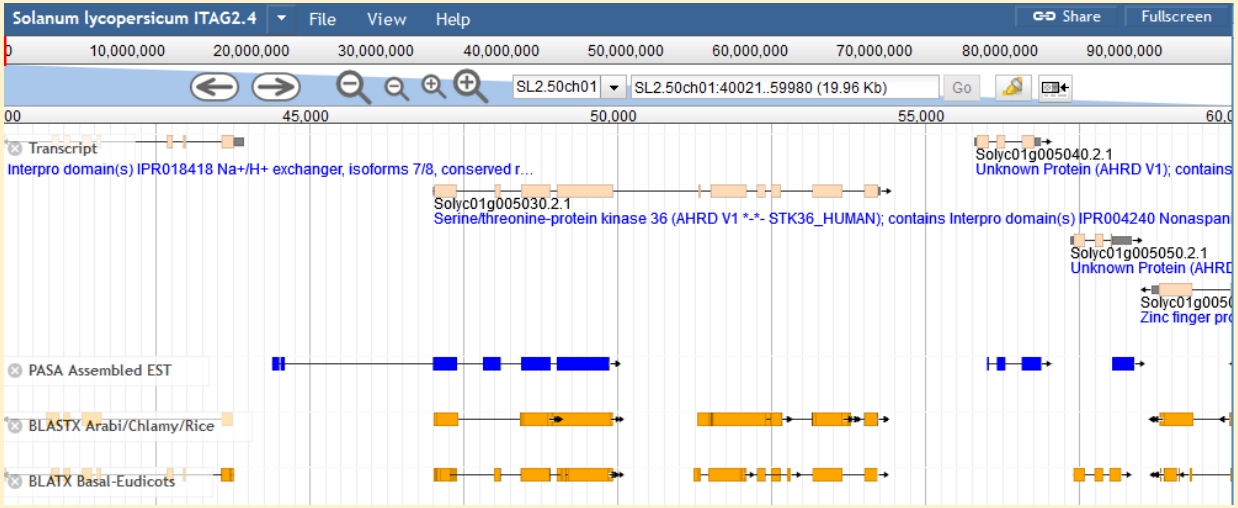
More on `GRanges` objects

- Can have their own `SeqInfo` object attached to them
- Can add or retrieve a data frame containing any additional info (one row per genomic range) using `mcols`
- Lots of other useful accessors (see help page)

```
GRanges object with 3 ranges and 1 metadata column:
                seqnames        ranges strand | When_I_decided_it_was_cool
                   <Rle>     <IRanges>  <Rle> |                      <Date>
  favorite gene    Chr01       200-340      * |                  2018-09-05
      okay gene    Chr02 110046-115077      * |                  2018-09-07
       meh gene    Chr02     3005-4200      * |                  2018-10-20
  -------
  seqinfo: 2 sequences from an unspecified genome; no seqlengths
```

What is genome annotation?

- Indicating where genes and coding regions are located in the genome
- Generally done by sequencing transcriptome (RNA) and aligning to whole genome sequence
- Similarity to genes from model species adds additional evidence
- Predicted gene functions listed based on protein similarities to known genes



Genome annotation file formats

- GFF3 – tab-delimited listing of features

```
##gff-version 3
ctg123 . mRNA            1300  9000  .  +  .  ID=mrna0001;Name=sonichedgehog
ctg123 . exon            1300  1500  .  +  .  ID=exon00001;Parent=mrna0001
ctg123 . exon            1050  1500  .  +  .  ID=exon00002;Parent=mrna0001
ctg123 . exon            3000  3902  .  +  .  ID=exon00003;Parent=mrna0001
ctg123 . exon            5000  5500  .  +  .  ID=exon00004;Parent=mrna0001
ctg123 . exon            7000  9000  .  +  .  ID=exon00005;Parent=mrna0001
```

start end strand

- GFF/GTF is similar tab-delimited format, different column order

TxDb objects

- From GenomicFeatures package
- Contain locations of genes, exons, and CDS for entire genome
- Do not contain sequence
- Generally for crops, we want to download a GFF3, GFF, or GTF file and import with `makeTxDbFromGFF`
- Can use `saveDb` to save it to a file, to quickly reload later with `loadDb`

Previewing `TxDb` objects

- `seqlevels` to view chromosome names
- `columns` to view column names
- `keytypes` to see columns that can be used for looking up features
- `keys` to get a vector of gene/transcript/exon IDs for the specified keytype

Extracting features from `TxDb` objects

- `genes`, `transcripts`, `exons`, `cds`, and `promoters` functions all return `GRanges` objects
- Use `columns` argument to specify what metadata columns you want
- Use the `filter` argument to filter elements by key

- Alternatively, `transcriptsByOverlaps`, `exonsByOverlaps`, and `cdsByOverlaps` will return `GRanges` for all elements within a range specified with another `GRanges` object

Mini exercise

- Pick some 50kb region in the tomato genome at random
- Use `transcriptsByOverlaps` to get transcript locations in that region
- Use `getSeq` to extract the sequences of those transcripts

Extracting features grouped by gene

- `transcriptsBy`, `exonsBy`, `cdsBy` functions
- `intronsByTranscript`, `threeUTRsByTranscript`, `fiveUTRsByTranscript`
- Give you a `GRangesList` object, one item per gene or transcript, listing all elements for that gene or transcript
- Usually run on whole `TxDb` at once

Thursday's lab

- Using primer sequences to get predicted PCR products