Laboratory 9: sequence assembly and remapping

In this laboratory exercise, you will simulate 'next generation' high–throughput (Illumina) sequencing reactions and then assemble the resulting reads into a consensus that will hopefully be similar to (if not the same as) the sequence input into the simulation. You will also map the simulated reads onto portions of the reference genome in order to determine if the two occurrences of the 23S gene is an assembly (simulated) artifact. To accomplish this, you will use the *rbcL* gene as a reference since there is only one copy in the finished sequence. This exercise should help you to understand the power and limitations of the sequencing technologies as well as the programs used to process the data.

Tasks

(1) Retrieve the *Ginkgo biloba* (Lin et al. 2012) whole plastid genome sequence by typing `esearch -db nuccore -query 'NC_016986[Accession]' | efetch -format fasta > Ginkgo.fasta` in the terminal.

(2) Install the ART 'next generation' sequence read simulation tools (Huang et al. 2012) by typing `sudo apt install art-nextgen-simulation-tools` in the terminal.

(3) Simulate a paired–end Illumina sequencing reaction by typing `art_illumina -i Ginkgo.fasta -p -l 150 -f 100 -m 400 -s 10 -na -rs 5 -ss MSv3 -o paired` in the terminal. Answer question (1).

(4) Install the SSAKE prefix–tree sequence assembler (Warren et al. 2007; 'Enjoy SSAKE responsibly!'):

   (a) Start by typing `sudo apt install ssake` in the terminal.

   (b) Read the documentation in the man page as well as the files '/usr/share/doc/ssake/TRIMMING_PAIRED_READS.README' and '/usr/share/doc/ssake/TQS.readme'.

   (c) Type `echo 'export PATH=$PATH:/usr/share/ssake/' >> .bashrc` in the terminal.

   (d) Close the terminal and open a new terminal window.

   (e) The current version of SSAKE's TQSfastq.py includes an indentation error. To fix it, first type `sudo apt install python3-autopep8` in the terminal to install a useful python tool. Agree to the install and type your password when requested.

   (f) To fix the indentation error, type `sudo autopep8 -i /usr/share/ssake/TQSfastq.py` in the terminal (type your password if requested).

(5) Begin to convert the paired–end FASTQ files output by ART into the FASTA–like format used by SSAKE for paired–end sequences by typing `perl -lane 'BEGIN{$x=0}{if($x==0){$_=~s/^@/>/;print($_)}elsif($x==1){print($_)}elsif($x==3){$x=-1};$x++}' paired1.fq > x.fasta` in the terminal.

(6) Finish the conversion by typing `perl -lane 'BEGIN{$x=0}{if($x==0){print(400)}elsif($x==1){print($_)}elsif($x==3){$x=-1};$x++}' paired2.fq | paste -d: x.fasta - > paired.fasta` in the terminal. Answer question (2).

(7) Assemble the raw paired–end read data by typing `ssake -f paired.fasta -w 2 -m 16 -o 2 -r 0.6 -t 5 -h 1 -p 1 -b paired-raw` in the terminal. Answer question (3).

(8) Quality trim the sequence data by typing `TQSfastq.py -f paired1.fq -t 10 -c 20 -e 33` followed by `TQSfastq.py -f paired2.fq -t 10 -c 20 -e 33` in the terminal. Answer question (4).

(9) Combine the two trimmed files by typing `makePairedOutput2UNEQUALfiles.pl paired1.fq_T10C20E33.trim.fa paired2.fq_T10C20E33.trim.fa 400` in the terminal. This should create two files: 'paired.fa' and 'unpaired.fa'. Answer question (5).

(10) Assemble the trimmed paired–end read data by typing `ssake -f paired.fa -w 2 -m 16 -o 2 -r 0.6 -t 5 -h 1 -p 1 -g unpaired.fa -b paired-trimmed` in the terminal.

(11) To calculate the median and maximum contig length, type `grep '>' paired-trimmed_contigs.fa | tr -d 'A-z' | datamash -t '|' median 2 max 2` in the terminal. Answer question (6).

(12) BLAST the combined contigs (scaffolds) against the plastid database created for Laboratory 7 by typing `blastn -query paired-trimmed_scaffolds.fa -task blastn -db plastid -outfmt '6 sscinames sseqid evalue bitscore score length pident qstart qend sstart send' -num_threads $(nproc) -max_target_seqs 5000 -out paired-trimmed.txt` in the terminal. Answer question (7).

(13) Read the ABySS (Jackman et al. 2017) documentation from the repository web page (https://github.com/bcgsc/abyss/blob/master/README.md). ABySS should have been installed during Laboratory 2.

(14) Create a trimmed input file in ABySS format by typing `perl -pe 's/^>@/>/' paired*.fq_T10C20E33.trim.fa > all-trim.fasta` in the terminal.

(15) Assemble the ssequences by typing `abyss-pe name=all-abyss24 B=250M k=24 q=20 in=all-trim.fasta` in the terminal to make an assembly using a *k* value of 24. Also try *k* values of 26, 28, and 30. Answer question (8).

assembler	*k*	number of contigs	maximum contig size (bp)	median contig size (bp)	median contig coverage
SSAKE	—				
ABySS	24				
ABySS	26				
ABySS	28				
ABySS	30				

(16) BLAST the best ABySS contigs against the plastid database created for Laboratory 7. Answer question (9).

(17) Retrieve 23S and *rbcL* sequences using re-PCR and BLAST.

(a) Create a re-PCR database by typing `famap -b Ginkgo.mmap -t N Ginkgo.fasta` followed by `fahash -b Ginkgo.hash -w 3 -f 2 Ginkgo.mmap` in the terminal.

(b) Create a primers file by typing `echo -e '23S\tGAGTGAAATAGAACATGAAACCGTAAG\tCTATTACG CACTCTTTCAAGGATGG\t600-650' > regions.primers` in the terminal. These primers correspond to a portion of 23S.

(c) Add *rbcL* primers (Poinar et al. 1998; Little 2014) by typing `echo -e 'rbcL\tATGTCACCACAA ACAGAGACTAAAGCAAGT\tCTGRGAGTTMACGTTTTCATCATC\t600-650' >> regions.primers` in the terminal.

(d) Use re-PCR to find the locations of the sequences by typing `re-PCR -S Ginkgo.hash -n 5 -g 0 -o regions.rePCR regions.primers` in the terminal. Answer question (10).

(e) Extract the sequences by typing `grep -v '^#' regions.rePCR | perl -F'\t' -lane '{ $F[2]=~s/\-/minus/;$F[2]=~s/\+/plus/;print("blastdbcmd -db plastid -dbtype nucl -entry ".$F[1]." -strand ".$F[2]." -range ".$F[3]."-".$F[4])}' | bash > regions.fasta` in the terminal.

(f) Rename the extracts to their gene names by typing `perl -pe '{s/ Ginkgo biloba chloroplast, complete genome//; s/111841-112463/23S/; s/c144404-143782/23S/; s/59807-60433/rbcL/}' regions.fasta > regions-named.fasta` in the terminal. Answer question (11).

(18) Build and install STAR (Dobin et al. 2013):

(a) Download STAR by typing `wget https://github.com/alexdobin/STAR/archive/refs/tags/2.7.10b.tar.gz` in the terminal.

(b) Extract the archive by typing `tar xvzf 2.7.10b.tar.gz` in the terminal.

(c) Change to the source directory by typing `cd STAR-2.7.10b/source/` in the terminal.

(d) Install the dependencies by typing `sudo apt install zlib1g-dev` in the terminal. Agree to the install and type your password if requested.

(e) Build by typing `make STAR -j$(nproc) CFLAGS="-O2 -march=native"` in the terminal.

(f) Install STAR by typing `cp STAR $HOME/scripts/; cd` in the terminal.

(g) Read the STAR manual (https://github.com/alexdobin/STAR/blob/master/doc/STARmanual.pdf).

(19) Create a STAR database by typing `mkdir genomes` followed by `STAR --runThreadN $(nproc) --runMode genomeGenerate --genomeSAindexNbases $(grep -v '^>' regions-named.fasta | tr -d '\n' | perl -lane '{$x=int((log(length($F[0]))/log(2))/2-1); if($x<14){print($x)}else{print(14)}}') --genomeDir genomes --genomeFastaFiles regions-named.fasta` in the terminal. Answer question (12).

(20) Run STAR by typing `STAR --runThreadN $(nproc) --genomeDir genomes --readFilesIn paired1.fq paired2.fq --outFileNamePrefix Ginkgo-raw-` in the terminal. Answer question (13).

(21) Check the output by typing `grep -v '^@' Ginkgo-raw-Aligned.out.sam | perl -F'\t' -lane '{if(!(($F[1]&0x4)||($F[1]&0x200))){print($F[2])}}' | sort | uniq -c` in the terminal. Answer question (14). Enter the output data in the table below.

(22) Build and install karect (Allam et al. 2015).

(a) Download karect by typing `wget https://github.com/aminallam/karect/archive/v1.0.tar.gz` in the terminal.

(b) Extract the archive by typing `tar xvzf v1.0.tar.gz` in the terminal.

(c) Change to the source directory by typing `cd karect-1.0/` in the terminal.

(d) Build by typing `make -j$(nproc) CFLAGS="-O2 -march=native"` in the terminal.

(e) Install and karect by typing `cp karect $HOME/scripts/; cd` in the terminal.

(23) Error correct the simulated reads by typing `karect -correct -inputfile=paired1.fq -inputfile=paired2.fq -celltype=haploid -matchtype=hamming -threads=$(nproc)` in the terminal. Answer question (15).

(24) Map the error corrected reads, for both species, using STAR as in step (20) replacing the raw files with the karect ouput.

(25) Check the output using the command in step (21) replacing the file names as appropriate. Enter the output data in the table below. Answer question (16).

read type	reference	mapped reads	reads per reference base
raw	23S		
raw	*rbcL*		
karect	23S		
karect	*rbcL*		

Questions (https://forms.gle/s38Z2dq3g6p7tNPW7)

(1) For task (3):

  (a) What does each of the art_illumina options do?
  (b) Why would one specify a random seed rather than let the program use the default value?
  (c) How many reads were output?
  (d) Is this the expected number?

(2) For task (6), what does each part of the command do?

(3) For task (7):

  (a) What does each of the SSAKE options do?
  (b) How many contigs did the assembly create?
  (c) Why does the SSAKE documentation warn against assembling data that has not been quality trimmed?

(4) For task (8):

  (a) What does each of the TQS options do?
  (b) How many sequences were retained?
  (c) What is their median size?

(5) For task (9):

  (a) What does the script (and options) do?
  (b) How many sequences were retained?
  (c) What is their median size?

(6) For task (11):

  (a) How many contigs were created?
  (b) What is their median size?
  (c) Their median coverage?

(7) For task (12):

(a) What do the contigs BLAST to?
(b) Do they appear to be good assemblies?
(c) How can you tell?
(d) If you wanted a complete plastid genome, how could you further assemble the data?

(8) For task (15):

(a) What do each of the ABySS options do?
(b) Which *k* value worked best?
(c) What statistics support your choice?
(d) How many contigs were created?
(e) What is their median size?
(f) Their median coverage?

(9) For task (16):

(a) What do the contigs BLAST to?
(b) Do they appear to be good assemblies?
(c) Overall, which assembly program worked best? Why?

(10) For task (17)(d):

(a) How many 23S and *rbcL* regions are found?
(b) Do different re-PCR settings change this?

(11) For task (17)(f), explain what each step of the command does.

(12) For task (19), explain what each step of the command does.

(13) For task (20), explain what each of the STAR options does.

(14) For task (21), explain what each step of the command does.

(15) For task (23), explain what each of the karect options does.

(16) For task (25):

(a) What conclusions can you draw from the output?
(b) Does it appear that *Ginkgo biloba* should have two copies of 23S?

Literature cited

Allam, A., P. Kalnis & V. Solovyev. 2015. Karect: accurate correction of substitution, insertion and deletion errors for next–generation sequencing data. Bioinformatics 31: 3421–3428.

Dobin, A., C. A. Davis, F. Schlesinger, J. Drenkow, C. Zaleski, S. Jha, P. Batut, M. Chaisson & T. R. Gingeras. 2013. STAR: ultrafast universal RNA–seq aligner. Bioinformatics 29: 15–21.

Huang, W., L. Li, J. R. Myers & G. T. Marth. 2012. ART: a next–generation sequencing read simulator. Bioinformatics 28: 593–594.

Jackman, S. D., B. P. Vandervalk, H. Mohamadi, J. Chu, S. Yeo, S. A. Hammond, G. Jahesh, H. Khan, L. Coombe, R. L. Warren & I. Birol. 2017. ABySS 2.0: resource–efficient assembly of large genomes using a Bloom filter. Genome Research 27: 768–777.

Lin, C.-P., C.-S. Wu, Y.-Y. Huang & S.-M. Chaw. 2012. The complete chloroplast genome of *Ginkgo biloba* reveals the mechanism of inverted repeat contraction. Genome Biology and Evolution 4: 374–381.

Little, D. P. 2014. A DNA mini–barcode for land plants. Molecular Ecology Resources 14: 437–446.

Poinar, H. N., M. Hofreiter, W. G. Spaulding, P. S. Martin, B. A. Stankiewicz, H. Bland, R. P. Evershed, G. Possnert & S. Pääbo. 1998. Molecular coproscopy: dung and diet of the extinct ground sloth *Nothrotheriops shastensis*. Science 281: 402–406.

Warren, R. L., G. G. Sutton, S. J. M. Jones & R. A. Holt. 2007. Assembling millions of short DNA sequences using SSAKE. Bioinformatics 23: 500–501.

Due at the start of class March 28.