Analysis of high-throughput sequence data with the R language – a compilation of useful commands

Daniel Berner, Zoological Institute, University of Basel

January 2016

The commands include both **standard R commands** you might already know from analyses with R in non-genetic contexts, and **commands specific to R/Bioconductor packages** tailored to high-throughput DNA sequence data analysis. In addition, you can find **SAMtools commands** at the end of the file that will allow you to convert alignments in SAM format (as produced by aligner software like Novoalign, Bowtie etc.) to BAM format, which can be imported into R.

General info:

Download R: http://www.r-project.org/

Download Bioconductor packages: http://www.bioconductor.org/

Mailing lists for discussing problems: http://www.bioconductor.org/help/mailing-list/

A tutorial giving an overview of various Bioconductor applications: http://manuals.bioinformatics.ucr.edu/home/ht-seq

SAMtools: http://samtools.sourceforge.net/

Standard R commands (alphabetically):

`#`	The hash says that what follows is not interpreted by R; useful for annotation
`/`	Make sure you use forward slashes when specifying file paths
`<-`	R's assignment symbol for creating objects
`[]`	Square brackets immediately following an expression allow accessing specific elements of an object (e.g. `a<-c(5,7,1); a[2]` is 7)
`$`	Extract a part of an object, matching by name (e.g. a column from a data frame)
`:`	Operator generating sequences (e.g. `5:8` is 5, 6, 7, 8)

`agrep(pattern, X, max.distance=list(all=1, insertions=0, deletions=0, substitutions=1))` Approximate pattern search. returns the indices of the elements of X that match the search pattern while allowing for a level of mismatch defined by the `max.distance=` argument. See also `grep()` below

`as.character()` Converts on object to character string(s). useful to access DNAString or DNAStringSet objects, see below. `as.numeric()` and `as.integer()` work analogously

`c()` Combines elements to form a vector (e.g. `a<-c(1,4,12)`)

`cbind(X,Y)` combines vectors and/or matrices by column (see `rbind()`)

`dim()` Returns the dimension (number of rows and columns) of an object

`for(i in 1:N){}` Construct a control-flow that loops from 1 to N

`grep(pattern, X, invert=FALSE)` Pattern search. returns the indices of the elements of X that match the search pattern. `invert=TRUE` reports the elements NOT matching the pattern. note that '^' and '$' should be used if the match should be exact at the start and at the end. See also `agrep()` above

`help("subset")` If the syntax of a command (here `subset()`) is unclear, use `help()` to call the html description file

`if(condition) {action}` Control-flow operator specifying an action conditional on a test criterion

`ifelse(condition, action, alternative)` Control-flow operator specifying an action and an alternative, conditional on a test criterion

`length()` Returns the number of elements in an object. e.g. `a<-c("C","y");` `length(a)` is 2

`library(Biostrings)` Load an R package (here the Bioconductor package Biostrings)

`merge(X, Y, by=)` Join two data frames by common column or row names

`names()` Returns the names of an object (e.g. the names of the columns of a data frame)

`paste(X,Y,sep="")` Concatenates characters or vectors to character strings, with a specific separator (including none). e.g. `paste("/path/", "filename", ".fastq", sep="")`

`plot()` R's general graphing command

`read.table("/path/file.txt", header=T, sep="\t")` Read a file into R

`rbind()` Combines vectors and/or matrices by row (see `cbind()`)

`rm()` Remove an R object; `rm(list=ls())` completely empties your work space

`sapply(X,fun)` Applies a function to each element of vector X (e.g. `fun<-function(z) {z+5}; a<-c(1,4,12); sapply(a,fun)` is 6,9,17)

`sort(X, decreasing=TRUE)` Sort elements of a vector (see also `order()`)

`subset(X, X[,1]=="blabla", select=)` Extracts a row-subset of a matrix according to the specified level of a factor column of the matrix. `select=` allows specifying the columns to extract

`table()` Generates a contingency table with counts for factor levels

`unique()` Returns the unique elements of an object. e.g. `a<-c(2,6,2,1,1,6)`; `unique(a)` is 2,6,1

`which(X==pattern)` Returns the indices of the elements of X exactly matching the search pattern. Much faster than `grep()`

`write.table(X, file="/path/file.txt")` Write R object to text file

Commands specific to the packages ShortRead, Biostrings, and Rsamtools (alphabetically):

`clean()` Removes all those reads from a ShortRead object that contain one or more non-A,T,G,C bases

`complement()` Complements a sequence (e.g. `a<-DNAString("AAGGC"); complement(a)` is TTCCG)

`DNAString("ATCG")` Allows efficient storage and manipulation of a DNA sequence; accepts only IUPAC genetic alphabet entries, and the gap symbol '-'

`DNAStringSet()` Efficient storage of MULTIPLE sequences (e.g. the sequences of a ShortRead object) in a single object

`id()` Calls the IDs of sequences in a ShortRead object

`narrow(X, start=, end=, width=)` Trims an object (e.g. ShortRead object) to the span specified (only two arguments needed, e.g. start & end)

`PhredQuality("IIIHHHHHHH@@@")` Turns character string into standard Sanger encoded ASCII characters; convert to integer with `as.integer()`

`quality()` Calls the quality string of sequences in a ShortRead object

`read.DNAStringSet("/path/file","fasta")` Import sequences stored in fasta format

`readFastq(dirPath="/path", pattern="X.fastq", withIds=T)`

Imports information in fastq format as ShortRead object. the sequences can be of different length, but sequence and quality need to match

`replaceLetterAt(X, at, letter)` Makes a copy of a DNAString(Set) object (=sequence, or a set of sequences) and then replaces some of the original letters by new letters at the specified locations

`reverse()` Reverse a sequence or quality string (e.g. `q<-PhredQuality("HHI##"); reverse(q)` is ##IHH)

`reverseComplement(X)` Reverse-complement a sequence. e.g. `a<-DNAString("AAGGC"); reverseComplement(a)` is GCCTT

`ScanBamParam(flag=scanBamFlag(isUnmappedQuery=FALSE), what=c("qname", "rname", "pos", "seq", "qual", "strand"), reverseComplement=TRUE)` Specify settings for importing an alignment in BAM format into R. see `ScanBam()`

`scanBam("/path/file.bam", param=ScanBamParam())[[1]]` Uploads an alignment in BAM format according to the parameters specified using `ScanBamParam()`. Components can be accessed using $. e.g. `X<-scanBam(infile, param= ScanBamParam())[[1]]; X$seq` calls the sequences

`sread()` Calls the sequences of a ShortRead object; output is DNAStringSet object

`subseq(X, start=, end=, width=)` Extract the subsequence from a DNAStringSet specified by start, end, and/or width, just like `narrow()`

`trimLRPatterns(Lpattern="AGCCD", subject=X, max.Lmismatch=0)` Trims a certain pattern from the start `(Lpattern)` or end `(Rpattern)` of ShortRead object X

`writeFastq(X, file="X.fastq", mode="w")` Writes ShortRead object X to a fastq file. `mode="w"` creates a new file, `"a"` appends to an already existing file X.fastq

`write.XStringSet(X, file="path/name.txt", format="fasta", width=80)` Writes out a string set object; here a DNAStringSet to fasta

---

SAM to BAM conversion of alignments by using SAMtools:

`cd/user/local` Go to directory where SAMtools is stored

```
./samtools view -b -S /path/file.sam >/path/file.bam
```