Tutorial: BASH one-liners for subsampling reads

Umer Zeeshan Ijaz

In the following one-liners, k=10000 subsamples the files to 10000 reads using reservoir sampling:

Subsampling paired-end reads (FASTQ):

```
paste forward.fastq reverse.fastq | awk '{ printf("%s",$0); n++;
if(n%4==0) { printf("\n");} else { printf("\t");} }' |
awk -v k=10000 'BEGIN{srand(systime() + PROCINFO["pid"]);}{s=x++<k?x-
1:int(rand()*x);if(s<k)R[s]=$0}END{for(i in R)print R[i]}' |
awk -F"\t" '{print $1"\n"$3"\n"$5"\n"$7 > "forward_sub.fastq";print
$2"\n"$4"\n"$6"\n"$8 > "reverse_sub.fastq"}'
```

Subsampling paired-end reads (FASTA):

```
paste <(awk '/^>/ {printf("\n%s\n",$0);next; }
{printf("%s",$0);}  END {printf("\n");}' < forward.fasta) <(awk '/^>/
{printf("\n%s\n",$0);next; } { printf("%s",$0);}  END
{printf("\n");}' < reverse.fasta) | awk 'NR>1{ printf("%s",$0); n++;
if(n%2==0) { printf("\n");} else { printf("\t");} }' |
awk -v k=10000 'BEGIN{srand(systime() + PROCINFO["pid"]);}{s=x++<k?x-
1:int(rand()*x);if(s<k)R[s]=$0}END{for(i in R)print R[i]}' |
awk -F"\t" '{print $1"\n"$3 > "forward_sub.fasta";print $2"\n"$4 >
"reverse_sub.fasta"}'
```

Subsampling single reads (FASTQ):

```
cat single.fastq | awk '{ printf("%s",$0); n++; if(n%4==0) {
printf("\n");} else { printf("\t");} }' |
awk -v k=10000 'BEGIN{srand(systime() + PROCINFO["pid"]);}{s=x++<k?x-
1:int(rand()*x);if(s<k)R[s]=$0}END{for(i in R)print R[i]}' |
awk -F"\t" '{print $1"\n"$2"\n"$3"\n"$4 > "single_sub.fastq"}'
```

Subsampling single reads (FASTA):

```
awk '/^>/ {printf("\n%s\n",$0);next; } { printf("%s",$0);}  END
{printf("\n");}' < single.fasta | awk 'NR>1{ printf("%s",$0); n++;
if(n%2==0) { printf("\n");} else { printf("\t");} }' |
awk -v k=10000 'BEGIN{srand(systime() + PROCINFO["pid"]);}{s=x++<k?x-
1:int(rand()*x);if(s<k)R[s]=$0}END{for(i in R)print R[i]}' |
awk -F"\t" '{print $1"\n"$2 > "single_sub.fasta"}'
```

Reservoir sampling:

A simple random sampling strategy to produce a sample without replacement from a stream of data - that is, in one pass: O(N)

Want to sample s instances - uniformly at random without replacement - from a population size of n records, where n is not known.

Figuring out n would require 2 passes. Reservoir sampling achieves this in 1 pass.
A reservoir R here is simply an array of size s. Let D be data stream of size n

Algorithm:

```
Store first $s$ elements into $R$.
for each element in position $k = s+1$ to $n$ ,
    accept it with probability $s/k$
    if accepted, choose a random element from $R$ to replace.
```

Partial analysis:

Base case is trivial. For the $k+1$st case, the probability a given element i with position $<= k$ is in R is s/k. The prob. i is replaced is the probability $k+1$st element is chosen multiplied by i being chosen to be replaced, which is: $s/(k+1) * 1/s = 1/(k+1)$, and prob that i is not replaced is $k/k+1$.

So any given element's probability of lasting after $k+1$ rounds is: (chosen in k steps, and not removed in k steps)

$= s/k * k/(k+1)$, which is $s/(k+1)$.

So, when $k+1 = n$, any element is present with probability s/n.

Reference: http://blogs.msdn.com/b/spt/archive/2008/02/05/reservoir-sampling.aspx