Bioinformatics one-liners

Useful bash one-liners for useful for bioinformatics.

https://github.com/stephenturner/oneliners

Sources:

- http://sed.sourceforge.net/sed1line.txt
- https://github.com/lh3/seqtk
- http://lh3lh3.users.sourceforge.net/biounix.shtml
- http://genomespot.blogspot.com/2013/08/a-selection-of-useful-bash-one-liners.html
- http://biowize.wordpress.com/2012/06/15/command-line-magic-for-your-gene-annotations/
- http://genomics-array.blogspot.com/2010/11/some-unixperl-oneliners-for.html
- http://gettinggeneticsdone.blogspot.com/2013/10/useful-linux-oneliners-for-bioinformatics.html#comments

awk, sed

Sum column 1 of file.txt:

```
awk '{sum+=$1} END {print sum}' file.txt
```

Number each line in file.txt:

```
sed = file.txt | sed 'N;s/\n/ /'
```

Get unique entries in file.txt based on column 1 (takes only the first instance):

```
awk '!arr[$2]++' file.txt
```

Print each line where the 5th field is equal to ‘abc123’:

```
awk '$5 == "abc123"' file.txt
```

Print each line where the 5th field is *not* equal to ‘abc123’:

```
awk '$5 == "abc123"' file.txt
```

Print each line whose 7th field matches the regular expression:

```
awk '$7  ~ /^[a-f]/' file.txt
```

Print each line whose 7th field *does not* match the regular expression:

```
awk '$7  ~ /^[a-f]/' file.txt
```

Replace all occurances of `foo` with `bar` in file.txt:

```
sed 's/foo/bar/g' file.txt
```

Convert a FASTQ file to FASTA:

```
sed -n '1~4s/^@/>/p;2~4p' file.fq > file.fa
```

Extract every 4th line starting at the second line (extract the sequence from FASTQ file):

```
sed -n '2~4p' file.fq
```

Basic sequence statistics. Print total number of reads, total number unique reads, percentage of unique reads, most abundant sequence, its frequency, and percentage of total in file.fq:

```
cat myfile.fq | awk '((NR-2)%4==0){read=$1;total++;count[read]++}END{for(read in
count){if(!max||count[read]>max)
{max=count[read];maxRead=read};if(count[read]==1){unique++}};print
total,unique,unique*100/total,maxRead,count[maxRead],count[maxRead]*100/total}'
```

Convert .bam back to .fastq:

```
samtools view file.bam | awk 'BEGIN {FS="\t"} {print "@" $1 "\n" $10 "\n+\n"
$11}' > file.fq
```

Keep only top bit scores in blast hits (best bit score only):

```
awk '{ if(!x[$1]++) {print $0; bitscore=($14-1)} else { if($14>bitscore) print
$0} }' blastout.txt
```

Keep only top bit scores in blast hits (5 less than the top):

```
awk '{ if(!x[$1]++) {print $0; bitscore=($14-6)} else { if($14>bitscore) print
$0} }' blastout.txt
```

Trim leading whitespace in file.txt:

```
sed 's/^[ \t]*//' file.txt
```

Trim trailing whitespace in file.txt:

```
sed 's/[ \t]*$//' file.txt
```

Trim leading and trailing whitespace in file.txt:

```
sed 's/^[ \t]*//;s/[ \t]*$//' file.txt
```

Delete blank lines in file.txt:

```
sed '/^$/d' file.txt
```

sort, uniq, cut, etc.

Count the number of unique lines in file.txt

```
cat file.txt | sort | uniq | wc -l
```

Find number of lines shared by 2 files:

```
sort file1 file2 | uniq -d
```

Find the most common strings in column 2:

```
cut -f2 file.txt | sort | uniq -c | sort -k1nr | head
```

Pick 10 random lines from a file:

```
shuf file.txt | head -n 10
```

Print rows where column 3 is larger than column 5 in file.txt:

```
awk '$3>$5' file.txt
```

Compute the mean of column 2:

```
awk '{x+=$2}END{print x/NR}' file.txt
```

Extract fields 2, 4, and 5 from file.txt:

```
awk '{print $2,$4,$5}' input.txt
```

Print all possible 3mer DNA sequence combinations:

```
echo {A,C,T,G}{A,C,T,G}{A,C,T,G}
```

Untangle an interleaved paired-end FASTQ file. If a FASTQ file has paired-end reads intermingled, and you want to separate them into separate /1 and /2 files, and assuming the /1 reads precede the /2 reads:

```
cat interleaved.fq |paste - - - - - - - - | tee >(cut -f 1-4 | tr "\t" "\n" >
deinterleaved_1.fq) | cut -f 5-8 | tr "\t" "\n" > deinterleaved_2.fq
```

find, xargs, and GNU parallel

Download GNU parallel at https://www.gnu.org/software/parallel/.

Search for .bam files anywhere in the current directory recursively:

```
find . -name "*.bam"
```

Delete all .bam files:

```
find . -name "*.bam" | xargs rm
```

Rename all .txt files to .bak (backup *.txt before doing something else to them, for example):

```
find . -name "*.txt" | sed "s/\.txt$//" | xargs -i echo mv {}.txt {}.bak | sh
```

Chastity filter raw Illumina data (grep reads containing `:N:`, append (-A) the three lines after the match containing the sequence and quality info, and write a new filtered fastq file):

```
find *fq | parallel "cat {} | grep -A 3 '^@.*[^:]*:N:[^:]*:' | grep -v '^\-\-$'
> {}.filt.fq"
```

Run FASTQC in parallel 12 jobs at a time:

```
find *.fq | parallel  -j 12 "fastqc {} --outdir ."
```

Index your bam files in parallel, but only echo the commands (`--dry-run`) rather than actually running them:

```
find *.bam | parallel --dry-run 'samtools index {}'
```

seqtk

Download seqtk at https://github.com/lh3/seqtk. Seqtk is a fast and lightweight tool for processing sequences in the FASTA or FASTQ format. It seamlessly parses both FASTA and FASTQ files which can also be optionally compressed by gzip.

Convert FASTQ to FASTA:

```
seqtk seq -a in.fq.gz > out.fa
```

Convert ILLUMINA 1.3+ FASTQ to FASTA and mask bases with quality lower than 20 to lowercases (the 1st command line) or to `N` (the 2nd):

```
seqtk seq -aQ64 -q20 in.fq > out.fa
seqtk seq -aQ64 -q20 -n N in.fq > out.fa
```

Fold long FASTA/Q lines and remove FASTA/Q comments:

```
seqtk seq -Cl60 in.fa > out.fa
```

Convert multi-line FASTQ to 4-line FASTQ:

```
seqtk seq -l0 in.fq > out.fq
```

Reverse complement FASTA/Q:

```
seqtk seq -r in.fq > out.fq
```

Extract sequences with names in file `name.lst`, one sequence name per line:

```
seqtk subseq in.fq name.lst > out.fq
```

Extract sequences in regions contained in file `reg.bed`:

```
seqtk subseq in.fa reg.bed > out.fa
```

Mask regions in `reg.bed` to lowercases:

```
seqtk seq -M reg.bed in.fa > out.fa
```

Subsample 10000 read pairs from two large paired FASTQ files (remember to use the same random seed to keep pairing):

```
seqtk sample -s100 read1.fq 10000 > sub1.fq
seqtk sample -s100 read2.fq 10000 > sub2.fq
```

Trim low-quality bases from both ends using the Phred algorithm:

```
seqtk trimfq in.fq > out.fq
```

Trim 5bp from the left end of each read and 10bp from the right end:

```
seqtk trimfq -b 5 -e 10 in.fa > out.fa
```

Untangle an interleaved paired-end FASTQ file. If a FASTQ file has paired-end reads intermingled, and you want to separate them into separate /1 and /2 files, and assuming the /1 reads precede the /2 reads:

```
seqtk seq -l0 interleaved.fq | awk '{if ((NR-1) % 8 < 4) print >>
"deinterleaved_1.fq"; else print >> "deinterleaved_2.fq"}'
```

GFF3 Annotations

Print all sequences annotated in a GFF3 file.

```
cut -s -f 1,9 yourannots.gff3 | grep $'\t' | cut -f 1 | sort | uniq
```

Determine all feature types annotated in a GFF3 file.

```
grep -v '^#' yourannots.gff3 | cut -s -f 3 | sort | uniq
```

Determine the number of genes annotated in a GFF3 file.

```
grep -c $'\tgene\t' yourannots.gff3
```

Extract all gene IDs from a GFF3 file.

```
grep $'\tgene\t' yourannots.gff3 | perl -ne '/ID=([^;]+)/ and printf("%s\n",
$1)'
```

Print length of each gene in a GFF3 file.

```
grep $'\tgene\t' yourannots.gff3 | cut -s -f 4,5 | perl -ne '@v = split(/\t/);
printf("%d\n", $v[1] - $v[0] + 1)'
```

FASTA header lines to GFF format (assuming the length is in the header as an appended "_length" as in Velvet assembled transcripts):

```
grep '>' file.fasta | awk -F "_" 'BEGIN{i=1; print "##gff-version 3"}{ print
$0"\t
BLAT\tEXON\t1\t"$10"\t95\t+\t.\tgene_id="$0";transcript_id=Transcript_"i;i++ }'
> file.gff
```

Other generally useful aliases for your .bashrc

Get a prompt that looks like `user@hostname:/full/path/cwd/:$`

```
export PS1="\u@\h:\w\\$ "
```

Never type `cd ../../..` again:

```
alias ..='cd ..'
alias ...='cd ../../'
alias ....='cd ../../../'
alias .....='cd ../../../../'
alias ......='cd ../../../../../'
```

Ask before removing or overwriting files:

```
alias mv="mv -i"
alias cp="cp -i"
alias rm="rm -i"
```

My favorite `ls` aliases:

```
alias ls="ls -1p --color=auto"
alias l="ls -lhGgo"
alias ll="ls -lh"
alias la="ls -lhGgoA"
alias lt="ls -lhGgotr"
alias lS="ls -lhGgoSr"
alias l.="ls -lhGgod .*"
alias lhead="ls -lhGgo | head"
alias ltail="ls -lhGgo | tail"
alias lmore='ls -lhGgo | more'
```

Use `cut` on space- or comma- delimited files:

```
alias cuts="cut -d \" \""
alias cutc="cut -d \",\""
```

Pack and unpack tar.gz files:

```
alias tarup="tar -zcf"
alias tardown="tar -zxf"
```

Use `mcd` to create a directory and `cd` to it simultaneously:

```
function mcd { mkdir -p "$1" && cd "$1";}
```

Go up to the parent directory and list it's contents:

```
alias u="cd ..;ls"
```

Make grep pretty:

```
alias grep="grep --color=auto"
```

Refresh your `.bashrc`:

```
alias refresh="source ~/.bashrc"
```

Common typos:

```
alias mf="mv -i"
alias mroe="more"
```