Genomic alignment, ancestral alleles, lift-over of coordinates

Ancestral allele inference and lift-over of genomic coordinates require a pairwise alignment of genome sequences. We align the nine-spined stickleback sequence with that of three-spined stickleback using the alignment program "last" (http://last.cbrc.jp/). Three-spined stickleback (*Gasterosteus aculeatus*) is a model species for studying morphological variation, evolution and population genetics, and is included in Ensembl (http://www.ensembl.org/Gasterosteus_aculeatus/Info/Index).

Genomic alignment

Make a new directory for the alignment and get the genome data there:

```
mkdir ~/session_7
cd ~/session_7
mkdir align

ln -s ../session_3/reference .
```

This file is big (448MB) and you don't need to download it if you skip the alignment step below (the alignment may take hours to complete).

```
# wget http://wasabiapp.org/vbox/data/session_7/threespine.fa -P reference
```

You can find the instructions for using "last" at http://last.cbrc.jp/doc/. We follow Examples 6 and 8 in the tutorial http://last.cbrc.jp/doc/last-tutorial.html. First we need to index (http://last.cbrc.jp/doc/lastdb.html) one of the sequences, here the one for nine-spined:

```
lastdb -uNEAR -cR11 reference/ninespinedb reference/ninespine.fa
```

We then align (http://last.cbrc.jp/doc/lastal.html) the second sequence (here three-spined) against that. We combine two commands to consider split alignments crossing rearrangement breakpoints (http://last.cbrc.jp/doc/last-split.html):

```
# lastal -m100 -E0.05 reference/ninespinedb reference/threespine.fa \
| last-split -m1  > align/out.maf
```

However, genome alignment takes easily a couple of hours and there's no point waiting for that. Instead we download the alignment and continue with that:

```
wget http://wasabiapp.org/vbox/data/session_7/out.maf -P align
```

As explained in Example 8 of the last tutorial, the commands above guarantee 1-to-1 relationship from three-spined to nine-spined but not necessarily for the opposite directions (can be 1-to-many). We fix that with commands:

```
maf-swap align/out.maf | last-split -m1 > align/out2.maf
```

We swap the sequences one more time to have nine-spined first (use `less -S out2.maf` and `less -S out3.maf` to see the difference):

```
maf-swap align/out2.maf > align/out3.maf
```

We then convert (http://last.cbrc.jp/doc/maf-convert.html) the maf alignment to sam format, sort that and conver to bam. This will be used for the AA reconstruction.

```
maf-convert sam -r 'ID:1 PL:ILLUMINA SM:threespine' align/out3.maf \
| samtools view -bt reference/ninespine.fa.fai -o align/out3.bam

samtools sort -T out3_sort align/out3.bam -Obam -o align/ThreeSpine.bam

samtools index align/ThreeSpine.bam
```

You can see the alignment with standard tools:

```
samtools view align/ThreeSpine.bam | less -S

samtools tview align/ThreeSpine.bam reference/ninespine.fa \
-p ctg7180000005484:19000
```

Ancestral alleles

We use the bam file to add ancestral alleles into our vcf file. First copy the vcf file from the previous time:

```
ln -s ../session_3/data101_good.vcf.gz .
```

We then use `samtools mpileup` and `bcftools call` to call the positions defined by `data101_good.vcf.gz`:

```
samtools mpileup -ugf reference/ninespine.fa -l data101_good.vcf.gz \
align/ThreeSpine.bam  | bcftools call -m \
| bcftools view -Oz -o ThreeSpine_good.vcf.gz

bcftools index ThreeSpine_good.vcf.gz
```

We merge the old vcf file and the new three-spined vcf file (did we forget something?):

```
bcftools merge -Oz -o merged.vcf.gz data101_good.vcf.gz ThreeSpine_good.vcf.gz
```

We then use `bcftools` to output specific fields and an awk command to pick either the reference or the variant allele as the fifth field:

```
bcftools view -m2 -M2 -v snps -s threespine merged.vcf.gz \
| bcftools query -f '%CHROM\t%POS\t%REF\t%ALT[\t%GT]\n' \
| awk '{OFS="\t";if($5=="0/0"){print $1,$2,$3,$4,$3} \
                 if($5=="0/1"){print $1,$2,$3,$4,$4}}' > data101_aa.tab
```

What are the five fields that we have picked?

This tab-separated file is compressed and indexed:

```
bgzip data101_aa.tab
tabix -s1 -b2 -e2 data101_aa.tab.gz
```

We create an Info line to be added to the vcf header section and then use `bcftools annotate` to add those fields to the vcf file (In fact, the first four fields are used to match the lines and only the fifth INFO/AA field is added as new data):

```
echo '##INFO=<ID=AA,Number=1,Type=Character,Description="Ancestral allele">' >
hdr.txt

bcftools annotate -a data101_aa.tab.gz -c CHROM,POS,REF,ALT,INFO/AA -h hdr.txt -Oz
-o data101_goodAA.vcf.gz merged.vcf.gz
```

We can check that everything looks OK:

```
bcftools view -m2 -M2 -v snps data101_goodAA.vcf.gz \
| bcftools query -f '%CHROM\t%POS\t%REF\t%ALT\t%INFO/AA\n' | less

bcftools view -m2 -M2 -v snps data101_goodAA.vcf.gz -H | wc -l

bcftools view -m2 -M2 -v snps -e 'INFO/AA="."' data101_goodAA.vcf.gz -H | wc -l
```

For what proportion of SNPs did we get an ancestral allele?

Creation of lift-over chain

We create the lift-over chain following the UCSC approach and using the Kent utilities:
http://genomewiki.ucsc.edu/index.php/LiftOver_Howto
https://genome.ucsc.edu/FAQ/FAQformat.html

```
mkdir align/chainMerge
mkdir align/net
```

Kent tools use the 2bit sequence format:

```
faToTwoBit reference/ninespine.fa reference/ninespine.2bit
#faToTwoBit reference/threespine.fa reference/threespine.2bit
```

If you don't have the three-spined genome as a Fasta file, you can download the 2bit version:

```
wget http://wasabiapp.org/vbox/data/session_7/threespine.2bit -P reference

twoBitInfo reference/ninespine.2bit reference/ninespine.chromInfo
```

```
twoBitInfo reference/threespine.2bit reference/threespine.chromInfo
```

First we convert the maf file to the psl format:

```
maf-convert psl align/out3.maf > align/out3.psl
```

Compare the two files:

```
ls -lh align/out3.maf
ls -lh align/out3.psl

less -S align/out3.maf
less -S align/out3.psl
```

We next chain the alignment hits and then sort them and split by the target:

```
axtChain -linearGap=medium -psl align/out3.psl reference/ninespine.2bit
reference/threespine.2bit align/out3.chain 2> align/axtChain.log

chainMergeSort align/out3.chain | chainSplit align/chainMerge stdin -lump=50
```

The different chains are catenated into one file and then sorted:

```
cat align/chainMerge/*.chain > align/all.chain

chainSort align/all.chain align/all.sorted.chain
```

We then make alignment nets out of chains:

```
chainNet align/all.sorted.chain reference/ninespine.chromInfo
reference/threespine.chromInfo align/net/all.net /dev/null
```

Finally we create a chain file with a subset of chains that appear in the net:

```
netChainSubset align/net/all.net align/all.chain reference/nineToThree.liftOver
```

This gives us a nine-spined-to-three-spined chain file. We swap that to get a three-spined-to-nine-spined chain file:

```
chainSwap reference/nineToThree.liftOver reference/threeToNine.liftOver
```

We have a look on one of the chain files (it looks boring) and compress them to make their use faster:

```
less -S reference/nineToThree.liftOver

gzip reference/nineToThree.liftOver
gzip reference/threeToNine.liftOver
```

We leave the actual use of the liftover chains for the next time.