Ensembl REST API, variant annotation and four-fold degenerate sites

Let's continue with the old data:

```
cd ~/session_7
mkdir annotation
cd annotation
```

Ensembl REST API

Much of Ensembl core data is available through Ensembl BioMart at http://www.ensembl.org/biomart. BioMart is useful for downloading large amounts of data for one organism. However, the data are mainly covering the organism's genes and BioMart contains very little comparative data (it contains homology information, though). BioMart can be accessed programmatically but its Perl API is not easy.

The new Ensembl REST API (http://rest.ensembl.org) provides the easiest access to many types of data. The API can be accessed using different programming languages or, more simply, using wget or curl.

See this region in Ensembl:
http://www.ensembl.org/Gasterosteus_aculeatus/Location/View?db=core;r=MT:1-15742

Get familiar with the API and, using wget or curl, download data for that region

- species is **stickleback**
- features **overlap** in any position in **MT** chromosome
- features are: **gene**, **transcript**, **cds**, **exon**
- output content type is: **text/x-gff3**
- output file name is **threespine_mt_genes.gff**

The output should match this:

```
wc threespine_mt_genes.gff
126  1203 25210 threespine_mt_genes.gff
```

Using these gene coordinates our aim is to annotate the mt genes in our ninespine data. We first need to transfer them to the ninespine genome:

```
CrossMap.py gff ../reference/threeToNine.liftOver.gz threespine_mt_genes.gff \
ninespine_mt_genes.gff
```

We have a problem, however. Can you spot what?

```
less -S ninespine_mt_genes.gff
awk '{print $4}' ninespine_mt_genes.gff

bcftools query -r deg7180000006464 -f '%CHROM\t%POS\n' \
../../session_3/data/data100.vcf.gz
```

We use for another chain file that maps the genes differently:

```
wget http://wasabiapp.org/vbox/data/session_11/threeToNine.liftOver.gz

CrossMap.py gff threeToNine.liftOver.gz threespine_mt_genes.gff \
ninespine_mt_genes.gff

less -S ninespine_mt_genes.gff
```

We then extract the variants for the mtDNA only, giving a smaller file that is easier to handle:

```
bcftools view -r deg7180000006464 ../../session_3/data101_clean.vcf.gz -Oz -o
data101_mt.vcf.gz

bcftools view -H data101_mt.vcf.gz | wc
```

What does this show?

R has a package called VariantAnnotation that takes a gff file, a vcf file and a reference genome. From these it finds the variants that overlap with gene annotations and computes the effect of the variant:

```
library(VariantAnnotation)
library(GenomicFeatures)

mttx <- makeTxDbFromGFF("ninespine_mt_genes.gff",
     format=c("gff3"),
     dataSource="rest.ensembl.org",
     dbxrefTag=c("gene_id"))

mtvcf = readVcf("data101_mt.vcf.gz",c("stickleback"))

fas=FaFile("../reference/ninespine.fa")

vareff = predictCoding(mtvcf,mttx,fas)

vareff

names(vareff)
start(vareff)

vareff$REF

vareff[1,]
vareff[2,]
```

Now open the data in IGV: "data101_mt.vcf.gz" and "ninespine_mt_genes.gff". Zoom into a variant region. Right-click left panel "Sequence" and select "Show translation".

What is a potential problem? A hint can be found from the same left panel settings box.

Extracting four-fold degenerate sites

Four-fold degenerate sites are those positions in a protein coding sequence where none of the possible base substitutions changes the amino acid that the codon codes for. Because of this, these sites are considered a good proxy of neutral substitution process.

To test the analysis of four-fold degenerate sites we need to get some alignment data. Find coordinates of ENSGACT00000027727 and extract multiple alignment for different fish species:

```
wget -q --header='Content-type:application/json'
'http://rest.ensembl.org/alignment/region/stickleback/<fill in something
here>?species_set_group=fish'  -O - > mt_fish.tmp
```

The output is in json format. A not-so-elegant way to parse the interesting fields is this:

```
sed 's/strand/\n/g' mt_fish.tmp | grep MT | cut -d\" -f8-9,17 \
| sed 's/:"/>/;s/"/\n/' | grep . > mt_fish.fas

less -S mt_fish.tmp
less -S mt_fish.fas
```

This data can now be analysed with R (after installing the missing packages):

```
library(rphast)

msa <- read.msa("mt_fish.fas")

whole = feat(seq="gasterosteus_aculeatus", src=".", feature="CDS",start=1,
end=ncol.msa(msa))

ffd <- get4d.msa(msa,feature=whole)
write.msa(ffd,"fish_ffd.fas","FASTA")
write.msa(ffd,"fish_ffd.phy","PHYLIP")

write.msa(msa,"fish_all.fas","FASTA")
write.msa(msa,"fish_all.phy","PHYLIP")
```

We then return to bash and (1) fix the fasta file, remove the newlines from phyip file and run RAxML:

```
sed -i 's/ //' fish_ffd.fas
perl -pe 's/(\w{70})\n/$1/' fish_ffd.phy > fish_ffd.raxml
raxmlHPC -m GTRGAMMA -p 11 -s fish_ffd.raxml -n fish_ffd

sed -i 's/ //' fish_all.fas
perl -pe 's/(\w{70})\n/$1/' fish_all.phy > fish_all.raxml
raxmlHPC -m GTRGAMMA -p 11 -s fish_all.raxml -n fish_all
```

On my VM, the RAxML analysis takes too long (it should be a second or two, only). If that's the case, stop it and download the files:

```
wget http://wasabiapp.org/vbox/data/session_11/RAxML_bestTree.fish_ffd
wget http://wasabiapp.org/vbox/data/session_11/RAxML_bestTree.fish_all
```

With these we can continue in R:

```
library(ape)
msa.ffd=read.FASTA("fish_ffd.fas")
round(dist.dna(msa.ffd,model = "T92"),4)

msa.all=read.FASTA("fish_all.fas")
round(dist.dna(msa.all,model = "T92"),4)

tree.ffd = read.tree("RAxML_bestTree.fish_ffd")
tree.ffd2=root(tree.ffd,1)

plot(tree.ffd2)
edgelabels(text=round(tree.ffd2$edge.length,4),
adj = c(0.5, -.95),frame="n",cex=0.85)

tree.all = read.tree("RAxML_bestTree.fish_all")
tree.all2=root(tree.all,1)

plot(tree.all2)
edgelabels(text=round(tree.all2$edge.length,4),
adj = c(0.5, -.95),frame="n",cex=0.85)
```

What do you notice? Is the analysis of ffd sites appropriate for these species?