```
#!/usr/bin/perl -w
# get a list of overlapping genes

use strict;
use Bio::Tools::GFF;

my $debug = 0;
my $check_mRNAs = 0;
my $prog_report = 1;
my $usage = "$0 <ath GFF file>";
die $usage unless (@ARGV > 0);
my $gffFile = $ARGV[0];
my $gffio = Bio::Tools::GFF->new(-file=>$gffFile, -gff_version=>3);


##genes and other features
my %ignoredTypes;
my %chrs; ## chromosome => chromosome feature
my %genes; ## chromosome->gene id = gene feature
my %mRNAs; ## gene id->mRNA id->mRNA feature
my %CDSs; ## mRNA id->an array of features (UTRs and CDSs)
while(my $feature = $gffio->next_feature){
        if($debug){
                print STDERR "Feature: seqid=", $feature->seq_id,
                " start=", $feature->start, " end=",
                      $feature->end, " primary_tag=", $feature->primary_tag,
                          " sorce_tag=", $feature->source_tag, " strand=",
                      $feature->strand, "\n";
                print STDERR "Tag values:";
                foreach my $tag($feature->get_all_tags){
                        print STDERR " tag=", $tag, " values=", join(" ",
                                    $feature->get_tag_values($tag)), "\n";
                }
        }

    my $type = $feature->primary_tag;
        if($type eq "chromosome"){
                my ($chr_id, $rest) = $feature->get_tag_values("ID");
                $chrs{$chr_id} = $feature;
        }elsif($type eq "gene"){
                my ($gene_id, $rest) = $feature->get_tag_values("ID");
                $genes{$feature->seq_id}->{$gene_id} = $feature;
                if($debug){
                        print STDERR "find gene ", $gene_id, " on chromosome ",
                         $feature->seq_id, "\n";
                }
        }elsif($type eq "mRNA"){
                my ($gene_id, $rest) = $feature->get_tag_values("Parent");

                my ($mRNA_id, $rest_2) = $feature->get_tag_values("ID");
                $mRNAs{$gene_id}->{$mRNA_id} = $feature;
        }elsif(($type eq "five_prime_UTR") ||
               ($type eq "three_prime_UTR") ||
                   ($type eq "CDS")){
                my ($mRNA_id, $rest) = $feature->get_tag_values("Parent");
                my @mult_ids = split /\,/, $mRNA_id; # for CDS, exclude protein id
                if(@mult_ids > 1){
                        $mRNA_id = $mult_ids[0];
                }
                push @{$CDSs{$mRNA_id}}, $feature;
```

```
        }else{
                $ignoredTypes{$type}++;
        }
}

# check if coordinates of gene cover all mRNAs at the same locus
if($check_mRNAs){
foreach my $chr(keys %chrs){
        if($debug){
                #print STDERR "Chromosome is ", $chr, "\n";
        }
        next unless(defined $genes{$chr});
        my %genesHere = %{$genes{$chr}};
        foreach my $gene_id(keys %genesHere){
                my $geneFeat = $genes{$chr}->{$gene_id};
                if(!defined $mRNAs{$gene_id}){
                        print STDERR "Gene ", $gene_id, " has no mRNAs\n";
                        next;
                }
                my %mRNAsHere = %{$mRNAs{$gene_id}};
                foreach my $mRNA_id(keys %mRNAsHere){
                        my $mRNAFeat = $mRNAs{$gene_id}->{$mRNA_id};
                        if($geneFeat->contains($mRNAFeat)){
                                print "Gene ", $gene_id, " contains mRNA ", $mRNA_id, "\n";
                        }else{
                                die "Gene ", $gene_id, " does not contain mRNA ", $mRNA_id,
                                " gene_start=", $geneFeat->start, " gene_end=", $geneFeat->end,
                                " mRNA_start=", $mRNAFeat->start, " mRNA_end=", $mRNAFeat->end,
                                "\n";
                        }
                }
        }
}
}

## find overlapping genes
## first check if two genes overlap, then find the mRNAs that overlap the most
## print header
print join("\t", ("Chromosome", "GeneID1", "GeneID2",
                  "mRNAID1", "mRNAID2", "Strand1", "Strand2", "OverlapLen",
                                "WhetherContains", "mRNAStart1", "mRNAEnd1", "mRNAStart2",
                                "mRNAEnd2")),"\n";
my %pairs;
foreach my $chr(keys %chrs){
        if($debug){
                #print STDERR "Chromosome is ", $chr, "\n";
                #next unless($chr eq "Chr1");
        }
        next unless(defined $genes{$chr});
        if($prog_report){
                print STDERR "Doing chromosome $chr\n";
        }
        my @genesHere = sort keys %{$genes{$chr}};
        foreach my $i(0..$#genesHere){
                my $gene_id1 = $genesHere[$i];
                my $gene_feat1 = $genes{$chr}->{$gene_id1};
                if($prog_report){
                        print STDERR "Doing Gene 1 $gene_id1 (i=$i)\n";
                }
                foreach my $j(($i+1)..$#genesHere){
                        my $gene_id2 = $genesHere[$j];
```

```
                        next if ($gene_id1 eq $gene_id2);
                        next if(defined $pairs{$gene_id1}->{$gene_id2} ||
                                defined $pairs{$gene_id2}->{$gene_id1});
                    my $gene_feat2 = $genes{$chr}->{$gene_id2};
                        if($gene_feat1->overlaps($gene_feat2)){

                            $pairs{$gene_id1}->{$gene_id2} = 1;
                #determine if one gene contains the other
                                my $contains = 0;
                                if($gene_feat1->contains($gene_feat2) ||
                                   $gene_feat2->contains($gene_feat1)){
                                        $contains = 1;
                                }
                                #find two mRNAs that overlap the most
                                my ($max_overlap, $max_mRNA_feat1, $max_mRNA_feat2,
                                $max_mRNA_id1, $max_mRNA_id2) = (0, "NONE", "NONE",
                                "NONE", "NONE");
                                foreach my $mRNA_id1(sort keys %{$mRNAs{$gene_id1}}){
                                        my $mRNA_feat1 = $mRNAs{$gene_id1}->{$mRNA_id1};
                                        my @CDSs1 = @{$CDSs{$mRNA_id1}};
                                        foreach my $mRNA_id2(sort keys %{$mRNAs{$gene_id2}}){
                                                my $mRNA_feat2 = $mRNAs{$gene_id2}->{$mRNA_id2};
                                                my @CDSs2 = @{$CDSs{$mRNA_id2}};
                                                my $overlap = 0;
                                                foreach my $c1(@CDSs1){
                                                        foreach my $c2(@CDSs2){
                                                                my $inter = $c1->intersection($c2);
                                                                if(defined $inter){
                                                                        $overlap += $inter->length;
                                                                }
                                                        }
                                                }
                                                if($overlap > $max_overlap){
                                                        $max_overlap = $overlap;
                                                        $max_mRNA_feat1 = $mRNA_feat1;
                                                        $max_mRNA_feat2 = $mRNA_feat2;
                                                        $max_mRNA_id1 = $mRNA_id1;
                                                        $max_mRNA_id2 = $mRNA_id2;
                                                }
                                        }
                                }

                                if($max_overlap > 0){
                                if($max_mRNA_feat1->start <= $max_mRNA_feat2->start){
                                print join("\t", ($chr, $gene_id1, $gene_id2, $max_mRNA_id1,
$max_mRNA_id2,
                                           $gene_feat1->strand, $gene_feat2->strand, $max_overlap,
                                           $contains, $max_mRNA_feat1->start, $max_mRNA_feat1->end,
                                           $max_mRNA_feat2->start, $max_mRNA_feat2->end)), "\n";
                                }else{
                                        print join("\t", ($chr, $gene_id2, $gene_id1, $max_mRNA_id2,
$max_mRNA_id1,
                                        $gene_feat2->strand, $gene_feat1->strand, $max_overlap,
                                        $contains, $max_mRNA_feat2->start, $max_mRNA_feat2->end,
                                        $max_mRNA_feat1->start, $max_mRNA_feat1->end)), "\n";
                                }
                                }else{
                                        if($gene_feat1->start <= $gene_feat2->start){
                                print join("\t", ($chr, $gene_id1, $gene_id2, $max_mRNA_id1,
$max_mRNA_id2,
                                           $gene_feat1->strand, $gene_feat2->strand, $max_overlap,
```

```
                                        $contains, $gene_feat1->start, $gene_feat1->end,
                                        $gene_feat2->start, $gene_feat2->end)), "\n";
                                }else{
                                    print join("\t", ($chr, $gene_id2, $gene_id1, $max_mRNA_id2,
$max_mRNA_id1,
                                    $gene_feat2->strand, $gene_feat1->strand, $max_overlap,
                                    $contains, $gene_feat2->start, $gene_feat2->end,
                                    $gene_feat1->start, $gene_feat1->end)), "\n";
                                }
                                }

                        }
                }
        }
}

print STDERR "Ignored features:\n";
foreach my $t(sort keys %ignoredTypes){
        print STDERR join("\t", ($t, $ignoredTypes{$t})), "\n";
}
```