script: blastThemAll.pl

```
#! /usr/bin/perl -w
#
#
#
#
$dir1= $ARGV[0];#human or mouse
$dir2= $ARGV[1];# zebrafish


while (defined ($filename = glob ("$dir1\/*.fasta"))) {
      print $filename."\n";
      while (defined ($filename2 = glob ("$dir2\/*.nsi"))) {
            #print $filename2."\n";
            if ($filename2 =~ /$dir2\/(E[0-9a-z]+)_(E[a-z0-9]+)\./i) {
                  $aux1= $1;
                  $aux2= $2;
                  $filename2= $aux1."_".$aux2;
                  $aux3= $aux2."_".$aux1;
                  #print $aux3."| <-- aux\n";
            }
            if ($filename =~ /$dir1\/(\S+)\./) {
                  $aux= $1;
                  #print $aux."| <-- aux\n"
            }
            if ($aux eq $aux3) {
                  #print "entrou!!\n";
                  $res = $aux.".Mblt";
                  if (-e $res) {print "ja esta \n";} else {
                        system ("megablast -q -1 -W 12 -D 3 -F F -v 10 -b 10 -d
".$dir2."/".$filename2." -i ".$dir1."/".$aux.".fasta -o ".$res);
                        print("megablast -q -1 -W 12 -D 3 -F F -v 10 -b 10 -d
".$dir2."/".$filename2." -i ".$dir2."/".$aux.".fasta -o ".$res."\n");
                  }
            }
      }
}
```

script: blastZThemAll.pl

```
#! /usr/bin/perl -w
#
#
#
#
$pairs= $ARGV[0];
$dir1= $ARGV[1];
$dir2= $ARGV[2];


$res= "not_done.txt";
if (-e $res) {
} else {
      $res= "not_done2.txt";
}
open (RES, ">$res");
open (FIL,$pairs);
while (<FIL>) {
      $linha= $_;
      chomp $linha;
```

```
      @data= split (/\t/, $linha);
      if (@data == 2) {
      $zf_gene= $data[0];
      $mm_gene= $data[1];
      #do t he blast for this pair
      $aux2= $dir2."/".$mm_gene.".fasta";
      $aux= $dir1."/".$zf_gene.".fasta";
      $aux3= $zf_gene."_".$mm_gene.".BlZ";
      print $aux."\t".$aux2."\n";
      if (-e $aux3) {
            print "ja esta!\n";
      } else {
            if ( (-e $aux) && (-e $aux2) ) {
                  system ("blastz ".$aux." ".$aux2." H=2200 T=0 W=6 K=2200 >
".$aux3."");
                  print("blastz ".$aux." ".$aux2." H=2200 T=0 W=6 K=2200 >
".$aux3."\n");
            } else {
                  print RES $aux3."\n";
            }
      }
      }
}
close (FIL);
close (RES);
```

script: clean_empty_lines.pl

```
#! /usr/bin/perl -w


$file = $ARGV[0];
$res = $ARGV[1];
open (FIL, $file);
open (RES, ">$res");
while (<FIL>) {
      $linha = $_;
      if ($linha =~ /\S/) {
            print RES $linha;
      }

}
close (FIL);
close (RES);
```

script: compile_unig_expr_ForEnsemblIds.pl

```
#! /usr/bin/perl -w

$file1 = $ARGV[0];
$file2 = $ARGV[1];
$res = $ARGV[2];
$count = 0;
open (FIL, $file1);
open (RES, ">$res");
while (<FIL>) {
      $linha = $_;
      chomp $linha;
      @data1 = split (/\t/, $linha);
      $geneId = $data1[0];
      #print "\t".$geneId."\n";
```

```
	if (@data1 > 1) {
		$uniId = $data1[1];
		print"\t".$geneId."\t".$uniId."\n";
		$expr = "";
		open (FIL2, $file2);
		while (<FIL2>) {
			$linha2 = $_;
			chomp $linha2;
			@data2 = split (/\t/, $linha2);
			if ($data2[0] eq $uniId) {
				print $linha2."\n";
				$expr = $expr.$data2[1];
				print "\t".$expr."\n";
			}
		}
		close (FIL2);
		$presente = 0;
		for ($i = 1; $i <= $count; $i++) {
			if ($geneId eq $geneId[$i]) {
				$presente = 1;
				$expr[$i] = $expr[$i]."|".$expr;
			}
		}
		if ($presente == 0) {
			$count++;
			$geneId[$count] = $geneId;
			$expr[$count] = $expr;
		}
	} else {
		$presente = 0;
		for ($i = 1; $i <= $count; $i++) {
			$presente = 0;
			if ($geneId eq $geneId[$i]) {
				$presente = 1;
			}
		}
		if ($presente == 0) {
			$count++;
			$geneId[$count] = $geneId;
			$expr[$count] = "";
		}
	}

}
close (FIL);
for ($i = 1; $i <= $count; $i++) {
	print RES $geneId[$i]."\t".$expr[$i]."\n";
	print $geneId[$i]."\t".$expr[$i]."\n";
}
close (RES);
```

script: concatenate_bds.pl

```
#! /usr/bin/perl -w

while (defined($file = glob ("*.fa"))) {
	if ($file =~ /(E[0-9a-z]+_E[0-9a-z]+)_/i) {
		$res =$1.".fasta";
	}
	open (FIL, $file);
	open (RES, ">>$res");
```

```
      while (<FIL>) {
            $linha = $_;
            chomp $linha;
            if ($linha =~ /^>E[0-9a-z]+_E[0-9a-z]+_(.*)/) {
                  print RES ">".$1;
            } else {
                  print RES $linha."\n";
            }
      }
      close (FIL);
      close (RES);
}
```

script: extract_ECRs_locally.pl

```
#! /usr/bin/perl -w
#
#
#
#
$dirtxt= $ARGV[0];
$FA1= $ARGV[1];
$FA2= $ARGV[2];
$ECR1= $ARGV[3];
$ECR2= $ARGV[4];

#mkdir $ECR1;
#mkdir $ECR2;
$flag= 0;

while (defined ($filename = glob ("$dirtxt/*.txt"))) {
      print $filename."\n";
      $count_seq= 0;
      open (FIL, $filename);
      while (<FIL>) {
            $count_seq++;
            $linha = $_;
            chomp $linha;
            @data = split (/\t/, $linha);
            $fa1= $FA1."/".$data[0].".fasta";
            $fa2= $FA2."/".$data[1].".fasta";
            #print $fa1."\t".$fa2."\n";
            $res1=$ECR1."/".$data[0]."_".$data[1].".ECRS.fa";
            $res2=$ECR2."/".$data[1]."_".$data[0].".ECRS.fa";
            #print $res1."\t".$res2."\n";
            $ini= $data[3];
            $fim= $data[4];
            $flag= $data[2];
            $ini2= $data[6];
            $fim2= $data[7];
            $flag2= $data[5];
            $count= 0;
            open (RES1, ">>$res1");
            open (FA1, $fa1);
            RETORNOA:while (<FA1>) {
                  $linha1= $_;
                  #Get position (Human, they are always in the right
orientation!)
                  if ($linha1 =~ /^>/) {
                        if ($linha1 =~
/^>chromosome\:\S+\:(\S+)\:(\d+)\:(\d+)\:\d+/) {
```

```
                              $chr= $1;
                              $begin= $2;
                              $end= $3;
                        } elsif ($linha1 =~ /^>(.*).fa/) {
                              @coordenates = split (/_/, $1);
                              if ( @coordenates == 4) {
                                    $chr= $coordenates[@coordenates - 3];
                              } else {
                                    $chr= $coordenates[@coordenates -
4]."_".$coordenates[@coordenates - 3];
                              }
                              $begin= $coordenates[@coordenates - 2];
                              $end= $coordenates[@coordenates - 1];
                        }
                        $iniF= $begin + $ini - 1;
                        $fimF= $begin + $fim - 1;
                        print RES1
"\n>".$count_seq."_".$chr."_".$iniF."_".$fimF."_".$flag."\n";
                  }
                  #Get the seq ECR
                  if ($linha1 !~ /^>/) {
                        $aux= $linha1;
                        chomp $aux;
                        while ($aux ne "") {
                              #print $aux." < aux\n";
                              $car= substr ($aux, 0, 1);
                              $length= length ($aux);
                              $length= $length - 1;
                              $aux= substr ($aux, 1, $length);
                              $count++;
                              #print $count." < count\n";
                              if (($count <= $fim) && ($count >= $ini)) {
                                    print RES1 $car;
                                    #print  $car." < car\n";
                              }
                              if ($count > $fim) {
                                    last RETORNOA;
                              }
                        }
                  }
            }
            close (FA1);
            close (RES1);


            $count= 0;
            open (RES2, ">>$res2");
            open (FA2, $fa2);
            RETORNOB:while (<FA2>) {
                  $linha2= $_;
                  #get the name of the ECR for Chick
                  if ($linha2 =~ /^>/) {
                        if ($linha2 =~
/^>chromosome\:\S+\:(\S+)\:(\d+)\:(\d+)\:\d+/) {
                              $chr= $1;
                              $begin= $2;
                              $end= $3;
                        } elsif ($linha2 =~ /^>(.*).fa/) {
                              @coordenates = split (/_/, $1);
                              if ( @coordenates == 4) {
                                    $chr= $coordenates[@coordenates - 3];
```

```
                              } else {
                                    $chr= $coordenates[@coordenates -
4]."_".$coordenates[@coordenates - 3];
                              }
                              $begin= $coordenates[@coordenates - 2];
                              $end= $coordenates[@coordenates - 1];
                        }
                        if ($flag2 == 1) {
                              $iniF= $end - $fim2 + 1;
                              $fimF= $end - $ini2 + 1;
                              $ini2 = $iniF - $begin + 1;
                              $fim2 = $fimF - $begin + 1;
                        } else {
                              $iniF= $begin + $ini2 - 1;
                              $fimF= $begin + $fim2 - 1;
                        }
                        print RES2
"\n>".$count_seq."_".$chr."_".$iniF."_".$fimF."_".$flag2."\n";
                  }
                  #Get the seq for the ECR Chick
                  if ($linha2 !~ /^>/) {
                        $aux= $linha2;
                        chomp $aux;
                        while ($aux ne "") {
                              #print $aux." < aux\n";
                              $car= substr ($aux, 0, 1);
                              $length= length ($aux);
                              $length= $length - 1;
                              $aux= substr ($aux, 1, $length);
                              $count++;
                              #print $count." < count\n";
                              if (($count <= $fim2) && ($count >= $ini2)) {
                                    print RES2 $car;
                                    #print  $car." < car\n";
                              }
                              if ($count > $fim2) {
                                    last RETORNOB;
                              }
                        }
                  }
            }
            close (FA2);
            close (RES2);
      }
      close (FIL);
}
```

script: extract_fastaBDS_locally.pl

```
#! /usr/bin/perl -w
#
#
#
#
$nonexonic= $ARGV[0];
$FA= $ARGV[1];
#$BDS= $ARGV[2];

#mkdir $BDS;

while (defined ($filename = glob ("$nonexonic/*.NonExonic"))) {
```

```
	print "\n".$filename."<- filename\n";
	if ($filename =~ /_(E[a-z0-9]+)_(E[a-z0-9]+)\.ECRS\./i) {
		$aux1= $1;
		$aux2= $2;
		$gene= $aux1;
		$name= $aux1."_".$aux2;
	}
	$count_seq= 0;
	open (FIL, $filename);
	while (<FIL>) {
		$linha = $_;
		if ($linha !~ /\S/) {} else {
			$count_seq++;
			print $linha;
			chomp $linha;
			@data = split (/\s+/, $linha);
			@data2 = split (/_/, $data[0]);
			$chr = "";
			if (@data2 > 5) {
					$chr = $data2[@data2 - 5]."_";
			 }
				$chr = $chr.$data2[@data2 - 4];
				$ini = $data2[@data2 - 3];
				$fim = $data2[@data2 - 2];
				print $chr."|".$ini."|".$fim."\n";
				$fim = $ini + $data[3];
				$ini = $ini + $data[2];
				print $chr."|".$ini."|".$fim."\n";
			$ini = $ini - 40;
			$fim = $fim + 40;
				print $chr."|".$ini."|".$fim."\n";
			$fa= $FA."/".$gene.".fasta";
			$res= $name."_".$chr."_".$ini."_".$fim.".fa";
			print $res."\n";
			if (-e $res) {
				print $res." existe!\n";
			} else {
				#print $res."\n";
				$count= 0;
				open (RES, ">$res");
				open (FA, $fa);
				RETORNOA:while (<FA>) {
					$linha2= $_;
					#Get position
					#if ($linha2 =~
/^>chromosome\:\S+\:(\S+)\:(\d+)\:(\d+)\:\d+/) {
					if ($linha2 =~ /^>ENS[a-z0-
9]+_(\S+)_(\d+)_\d+.fa/i) {
						$chr= $1;
						$begin= $2;
						#print $linha."  < fa\n";
						$iniF= $ini - $begin + 1;
						$fimF= $fim - $begin + 1;
						print  RES
">".$name."_".$chr."_".$ini."_".$fim."\n";
					} elsif ($linha2 =~ /^>/) {
						print "\n".$filename."<- filename ERROR\n";
					}
					#Get the seq ECR
					if ($linha2 !~ /^>/) {
						$aux= $linha2;
```

```
                                    chomp $aux;
                                    while ($aux ne "") {
                                          #print $aux." < aux\n";
                                          $car= substr ($aux, 0, 1);
                                          $length= length ($aux);
                                          $length= $length - 1;
                                          $aux= substr ($aux, 1, $length);
                                          $count++;
                                          #print $count." < count\n";
                                          if (($count <= $fimF) && ($count >=
$iniF)) {
                                                print RES $car;
                                                #print  $car." < car\n";
                                          }
                                          if ($count > $fimF) {
                                                last RETORNOA;
                                          }
                                    }
                              }
                        }
                        close (FA);
                        close (RES);
                  }
            }
      }
      close (FIL);
}
```

script: extract_local_fast.pl

```
#! /usr/bin/perl -w
```

script: extract_ort_genes.pl

```
#! /usr/bin/perl -w

$file = $ARGV[0];
$res = $ARGV[1];

open (FIL, $file);
open (RES, ">$res");
while (<FIL>) {
      $linha = $_;
      chomp $linha;
      @data = split (/\t/, $linha);
      if (@data > 1) {
            print RES $data[1]."\n";
      }
}
close (FIL);
close (RES);
```

script: find_nice_all.pl

```
#! /usr/bin/perl -w
```

```
$res = $ARGV[0];
$tissue = $ARGV[1];

while (defined ($file = glob ("expr*.*"))) {
      print $file."\n";
      system ("perl
/Users/pedro/Desktop/Pax6_paper/Data1/Ensembl_181208/SCRIPTS_screen1/find_nice_g
enes.pl $file $res $tissue");
}
```

script: find_nice_genes.pl

```
#! /usr/bin/perl -w


$file = $ARGV[0];
$res = $ARGV[1];
$tissue = $ARGV[2];

$count = 0;
open (FIL, $file);
open (RES, ">>$res");
while (<FIL>) {
      $linha = $_;
      if ($linha =~ /$tissue/) {
            $count++;
      }
}
close (FIL);
print RES $file."\t".$count."\n";
close (RES);
```

script: find_nonOverlpBDS.pl

```
#! /usr/bin/perl -w

$file = $ARGV[0];
$res = $ARGV[1];
$count = 0;

open (RES, ">$res");
open (FIL, $file);
while (<FIL>) {
      $linha = $_;
      chomp $linha;
      @data = split (/\t/, $linha);
      $chr = $data[1];
      $ini = $data[2];
      $fim = $data[3];
      $new = 1;
      for ($i = 1; $i<= $count; $i++) {
            if ( (($fim > $ini[$i]) && ($ini < $ini[$i])) || (($fim[$i] > $ini)
&& ($ini[$i] < $ini)) ) {
                  $new = 0;
                  print $chr[$i]."   ".$ini[$i]."    ".$fim[$i]."
".$linha."\n";
            }
      }
      if ($new == 1) {
            $count++;
```

```
            $chr[$count] = $chr;
            $ini[$count] = $ini;
            $fim[$count] = $fim;
            print RES $linha."\n";
      }
}
close (FIL);
close (RES);
```

script: fix_enhancersV2.pl

```
#! /usr/bin/perl -w
#
#
while (defined ($filename = glob ("*.fa"))) {
      $res= $filename."F";
      open (FIL, $filename);
      open (RES, ">$res");
      while (<FIL>) {
            $linha = $_;
            if ($linha =~ /\S/) {
                  print RES $linha;
            }
      }
      close (FIL);
      close (RES);
}
```

script: fix_initial_gene_file_name.pl

```
#! /usr/bin/perl -w

while (defined ($file = glob ("*.fa"))) {
      if ($file =~ /(ENS[a-z0-9]+)/i) {
            $aux = $1.".fasta";
            system ("mv -f $file $aux");
            print $aux."\n";
      }
}
```

script: get_annotationForNames.pl

```
#! /usr/bin/perl -w

$file = $ARGV[0];
$annot = $ARGV[1];
$res = $ARGV[2];

open (FIL, $file);
open (RES, ">$res");
while (<FIL>) {
      $linha = $_;
      chomp $linha;
      open (FIL2, $annot);
      while (<FIL2>) {
            $linha2 = $_;
            if ($linha2 =~ /$linha/) {
                  print RES $linha2;
            }
      }
      close (FIL2);
```

```
}
close (FIL);
close (RES);
```

script: get_names.pl

```
#! /usr/bin/perl -w

$vari = $ARGV[0];
$res = $ARGV[1];

open (RES, ">$res");
while (defined ($file = glob ("*.csv"))) {
      print $file."\n";
      if ($file =~ /(E\S+)_(E\S+)/) {
            $gene1 = $1;
            $gene2 = $2;
            if ($vari ==1) {
                  print RES $gene1."\n";
            } else {
                  print RES $gene2."\n";
            }
      }
}
close (RES);
```

script: get_proteins2.pl

```
#! /usr/bin/perl -w

$file = $ARGV[0];
$rel = $ARGV[1];
$res = $ARGV[2];
open (FIL, $file);
open (RES, ">$res");
while (<FIL>) {
      $linha = $_;
      chomp $linha;
      open (REL, $rel);
      RETORNO:while (<REL>) {
            $linha2 = $_;
            chomp $linha2;
            @data2 = split (/\t/,$linha2);
            if (@data2 > 1) {
                  if ($data2[0] eq $linha) {
                        print RES $data2[2]."\n";
                        last RETORNO;
                  }
            }
      }
      close (REL);
}
close (FIL);
close (RES);
```

script: get_proteins3.pl

```
#! /usr/bin/perl -w

$file = $ARGV[0];
$rel = $ARGV[1];
```

```
$res = $ARGV[2];
open (FIL, $file);
open (RES, ">$res");
while (<FIL>) {
      $linha = $_;
      chomp $linha;
      @data = split (/\t/, $linha);
      $gene = $data[0];
      open (REL, $rel);
      RETORNO:while (<REL>) {
            $linha2 = $_;
            chomp $linha2;
            @data2 = split (/\t/,$linha2);
            if (@data2 > 1) {
                  if ($data2[0] eq $gene) {
                        print RES $linha."\t".$data2[2]."\n";
                        last RETORNO;
                  }
            }
      }
      close (REL);
}
close (FIL);
close (RES);
```

script: hmmsearchManyAgainstMAny.pl

```
#! /usr/bin/perl -w
#
#
#
$note= "usage: hmmsearchManyAgainstMAny <dir hmm> <dir dna>";
die "$note\n" if (@ARGV!=2);

$dir_hmm= $ARGV[0];
$dir_dna= $ARGV[1];

$count_hmm= 0;
$count_dna= 0;

while (defined ($filename = glob ("$dir_hmm/*.hmm"))) {
      $count_hmm++;
      $hmm[$count_hmm]= $filename;
      #print $filename."\n";
}

while (defined ($filename = glob ("$dir_dna/*.faF"))) {
      $count_dna++;
      if ($filename =~ /$dir_dna\/(\S+.faF)/) {
            $dna[$count_dna]= $1;
      }
      #print $filename."\n";
}

for ($i = 1; $i <= $count_hmm; $i++) {
      if ($hmm[$i] =~ /([a-z0-9_\-\.]+)\.hmm/i) {
            $hmm= $1;
      }
      for ($j = 1; $j <= $count_dna; $j++) {
            if ($dna[$j] =~ /([a-z0-9_\-\.]+).faF/i) {
                  $dna= $1;
```

```
            }
            $aux1= $hmm."_".$dna.".out";
            $aux2= $hmm."_".$dna.".hmmsearch_prs";
            if ( (-e $aux1) || (-e $aux2) ) {
                  print $aux1." or ".$aux2."  ja existe\n";
            } else {
                  system ("hmmsearch -E 1000 $hmm[$i] $dir_dna/$dna[$j] >
$aux1");
                  print ("hmmsearch -E 1000 $hmm[$i] $dir_dna/$dna[$j] >
$aux1\n");
            }
      }
}
```

script: intersectAndExclude.pl

```
#! /usr/bin/perl -w
#
#
#
$file1= $ARGV[0];
$file2= $ARGV[1];

$res1= $file1."_INTERSECT_".$file2;
$res2= $file1."_MINUS_".$file2;
$res3= $file2."_MINUS_".$file1;

#upload the file1
#
$count1= 0;
open (FIL, $file1);
while (<FIL>) {
      $linha = $_;
      $count1++;
      $ele1[$count1]= $linha;
}
close (FIL);
#upload the file2
#
$count2= 0;
open (FIL, $file2);
while (<FIL>) {
      $linha = $_;
      $count2++;
      $ele2[$count2]= $linha;
}
close (FIL);
#get the res
#
open (RES1, ">$res1");
open (RES2, ">$res2");
open (RES3, ">$res3");
for ($i = 1; $i <= $count1; $i++) {
      $presente = 0;
      RETORNOA: for ($j = 1; $j <= $count2; $j++) {
            if ($ele1[$i] eq $ele2[$j]) {
                  print RES1 $ele1[$i];
                  $presente = 1;
                  last RETORNOA;
            }
      }
```

```
      if ($presente == 0) {
            print RES2 $ele1[$i];
      }
}
for ($i = 1; $i <= $count2; $i++) {
      $presente = 0;
      RETORNOB: for ($j = 1; $j <= $count1; $j++) {
            if ($ele2[$i] eq $ele1[$j]) {
                  $presente = 1;
                  last RETORNOB;
            }
      }
      if ($presente == 0) {
            print RES3 $ele2[$i];
      }
}
close (RES1);
close (RES2);
close (RES3);
```

script: make_bootS_files.pl

```
#! /usr/bin/perl -w

use warnings;


$file = $ARGV[0];
$numberF = $ARGV[1];
$numberL = $ARGV[2];
$range = $ARGV[3];

srand (time ^ $$ ^ unpack "%L*", `ps axww | gzip`);

for ($i = 1; $i <= $numberF; $i++) {
      for ($l = 1; $l <= $numberL; $l++) {
            $num[$l] = 0;
      }
      $res = $i."_".$file;
      if (-e $res) {} else {
            $count = 0;
            while ($count <= $numberL) {
                  $random_number = int(rand($range)) + 1;
                  #print $random_number."\n";
                  $presente = 0;
                  for ($k = 1; $k <= $count; $k++) {
                        if ($random_number == $num[$k]) {
                              $presente = 1;
                        }
                  }
                  if ($presente == 0) {
                        $count++;
                        $num[$count] = $random_number;
                  }
            }
            for ($k = 1 ; $k <= $numberL; $k++) {
                  #print $num[$k]."\t".$k."\n";
            }
            print $res."\n";
            open (RES, ">$res");
            for ($k = 1 ; $k <= $numberL; $k++) {
```

```
                $aux = 0;
                open (FIL, $file);
                while (<FIL>) {
                        $linha = $_;
                        $aux++;
                        if ($aux == $num[$k]) {
                                print RES $linha;
                        }
                }
                close (FIL);
        }
        close (RES);
    }
}
```

script: make_distrib_perGene.pl

```
#! /usr/bin/perl -w

$file = $ARGV[0];
$res = $ARGV[1];
$max = 200;

for ($i = 0; $i <= $max; $i++) {
    $countGenes[$i]= 0;
}

$count = 0;
$ultimo = "";
open (RES, ">$res");
open (FIL, $file);
while (<FIL>) {
    $linha = $_;
    chomp $linha;
    @data = split (/\t/, $linha);
    $gene = $data[0];
    if ( $gene eq $ultimo) {
        $count++;
    } else {
        $countGenes[$count] = $countGenes[$count] + 1;
        $count= 1;
        $ultimo = $gene;
    }

}
close (FIL);
$countGenes[$count] = $countGenes[$count] + 1;
for ($i = 1; $i <= $max; $i++) {
    print RES $i."\t".$countGenes[$i]."\n";
}
close (RES);
```

script: make_distrib_perGeneNames.pl

```
#! /usr/bin/perl -w

$file = $ARGV[0];
$res = $ARGV[1];
$count = 0;
$ultimo = "";
open (RES, ">$res");
```

```
open (FIL, $file);
while (<FIL>) {
      $linha = $_;
      chomp $linha;
      @data = split (/\t/, $linha);
      $gene = $data[0];
      if ($gene eq $ultimo) {
            $count++;
      } else {
            print RES $ultimo."\t".$count."\n";
            $count= 1;
            $ultimo = $gene;
      }

}
close (FIL);
print RES $ultimo."\t".$count."\n";
close (RES);
```

script: make_pair_files.pl

```
#! /usr/bin/perl -w

$file= $ARGV[0];
$res1= $ARGV[1];
$res2= $ARGV[2];

open (FIL, $file);
open (RES1, ">$res1");
open (RES2, ">$res2");
while (<FIL>) {
      $linha = $_;
      @data = split (/\t/, $linha);
      if ($linha =~ /ENSG/) {
            print RES1 $data[0];
            print RES1 "\t";
            print RES1 $data[8];
            print RES1 "\n";
      }
      if ($linha =~ /ENSMUSG/) {
            print RES2 $data[0];
            print RES2 "\t";
            print RES2 $data[12];
            print RES2 "\n";
      }
}
close (FIL);
close (RES1);
close (RES2);
```

script: make_pos_bds_put.pl

```
#! /usr/bin/perl -w

while (defined ($file = glob ("*.Mblt_res_70In40"))) {
      print $file."\n";
      if ($file =~ /(\S+)\.Mblt_res_70In40/) {
            $aux = $1;
            $res = $aux."_pos_put_bds.csv";
            if ($aux =~ /(E[0-9a-z]+)_(E[0-9a-z]+)/i) {
                  $gene1= $1;
```

```
                $gene2= $2;
                print $gene1."\t".$gene2."\n";
        }
    }
    open (FIL, $file);
    open (RES, ">$res");
    while (<FIL>) {
        $linha = $_;
        @data = split (/\s+/, $linha);
        @data1 = split (/_/, $data[0]);
        @data2 = split (/_/, $data[1]);
        if (@data1 > 5) {
                $chr1 = $data1[@data1 - 4]."_";
            }
            $chr1 = $data1[@data1 - 3];
            $ini1 = $data1[@data1 - 2];
        $fim1 = $ini1 + $data[7];
        $ini1 = $ini1 + $data[6];
        print $chr1."<--chr1\t".$ini1."<--ini1\t".$fim1."<--fim1\n";
        if (@data2 > 5) {
                $chr2 = $data2[@data2 - 4]."_";
            }
            $chr2 = $data2[@data2 - 3];
            $ini2 = $data2[@data2 - 2];
        $fim2 = $ini2 + $data[9];
        $ini2 = $ini2 + $data[8];
        print RES
$gene1."\t".$chr1."\t".$ini1."\t".$fim1."\t".$gene2."\t".$chr2."\t".$ini2."\t".$
fim2."\n";
    }
    close (RES);
    close (FIL);
}
```

script: make_pos_hmm_out.pl

```
#! /usr/bin/perl -w
$ending= $ARGV[0];
while (defined ($file = glob ("*.$ending"))) {
    print $file."\n";
    if ($file =~ /(\S+)\.$ending/) {
        $aux = $1;
        $res = $aux."_pos_put_bds.csv";
        if ($aux =~ /(E[0-9a-z]+)_(E[0-9a-z]+)/i) {
            $gene1= $1;
            $gene2= $2;
            print $gene1."\t".$gene2."\n";
        }
    }
    open (FIL, $file);
    open (RES, ">>$res");
    while (<FIL>) {
        $linha = $_;
        if ($linha =~ /\S/) {
            @data = split (/\s+/, $linha);
            @data1 = split (/_/, $data[0]);
            if (@data1 > 5) {
                    $chr1 = $data1[@data1 - 5]."_";
            }
            $chr1 = $data1[@data1 - 4];
            $ini1 = $data1[@data1 - 3];
```

```
                  $fim1 = $ini1 + $data[3];
                  $ini1 = $ini1 + $data[2];
                  print $chr1."<--chr1\t".$ini1."<--ini1\t".$fim1."<--fim1\n";
                  print RES $gene1."\t".$chr1."\t".$ini1."\t".$fim1."\n";
            }
      }
      close (RES);
      close (FIL);
}
```

script: make_relat_unigExprFiles.pl

```
#! /usr/bin/perl -w

$file = $ARGV[0];
$unig = $ARGV[1];
$res = $ARGV[2];

open (FIL, $file);
open (RES, ">$res");
while (<FIL>) {
      $linha = $_;
      if ($linha =~ /^(\S+)/) {
            #print $1."\n";
            $geneId = $1;
            $expr= "NULL";
            open (REL, $unig);
            while (<REL>) {
                  $linhaR = $_;
                  chomp $linhaR;
                  @dataR = split (/\t/, $linhaR);
                  if ($dataR[0] eq $geneId) {
                        if (@dataR > 1) {
                              $expr = $dataR[1];
                        }
                  }
            }
            close (REL);
            print RES $geneId."\t".$expr."\n";
      }
}
close (RES);
close (FIL);
```

script: make_relative_all.pl

```
#! /usr/bin/perl -w



$unig = $ARGV[0];

while (defined ($file = glob ("*.txt"))) {
      $res = "expr_".$file;
      if ((-e $res) || ($file=~ /^expr_/)) {} else {
            system ("perl
/Users/pedro/Desktop/Pax6_paper/Data1/Ensembl_181208/SCRIPTS_screen1/make_relat_
unigExprFiles.pl $file $unig $res");
      }
}
```

script: make_uscs_tables.pl

```
#! /usr/bin/perl -w

$gene_annot= $ARGV[0];
#read the pos files
while (defined ($pos_file = glob("*.csv"))) {
      if ($pos_file =~ /(\S+)\.csv/) {
            #$res = $1.".drer_uscs_table";
            $res = $1.".mmus_uscs_table";
      }
      print $pos_file."\n";
      $s_temp= "s_temp";
      system ("sort -g $pos_file > $s_temp");
      open (FIL, $s_temp);
      $count= 0;
      while (<FIL>) {
            $linha_pos_file = $_;
            print $linha_pos_file;
            chomp $linha_pos_file;
            @data_pos_file = split (/\t/, $linha_pos_file);
            $gene_name= $data_pos_file[0];
            $count++;
            $chr = $data_pos_file[1];
            $ini[$count]= $data_pos_file[2];
            $fim[$count]= $data_pos_file[3];
            if ($fim[$count] < $ini[$count]) {
                  $aux = $ini[$count];
                  $ini[$count] = $fim[$count];
                  $fim[$count] = $aux;
            }
      }
      print $gene_name."\n";
      print $count."\n";
      close (FIL);
      system ("rm -f $s_temp");
      #get the position of the locus
      open (GENE, $gene_annot);
      while (<GENE>) {
            $linha_gene_annot = $_;
            chomp $linha_gene_annot;
            @data_gene_annot = split (/\t/, $linha_gene_annot);
            if ($data_gene_annot[0] eq $gene_name) {
                  $gene_ini = $data_gene_annot[2];
                  $locus_ini = $gene_ini - 20000;
                  $gene_fim = $data_gene_annot[3];
                  $locus_fim = $gene_fim + 20000;
                  #$strand = $data_gene_annot[4];
            }
      }
      close (GENE);
      #calculate the relative positions for the pos seqs
      open (RES, ">$res");
      print RES $chr."\t".$locus_ini."\t".$locus_fim."\n";
      for ($i = 1; $i <= $count; $i++) {
            $rel_ini[$i] = $ini[$i] - $locus_ini + 1;
            $rel_fim[$i] = $fim[$i] - $locus_ini + 1;
            $size[$i] = $rel_fim[$i] - $rel_ini[$i] + 1;
            print RES
$gene_name."\t".$rel_ini[$i]."\t".$rel_fim[$i]."\t".$size[$i]."\n";
            # make the table
```

```
	}
	close (RES);
}

script: parse_BlZ.pl

#! /usr/bin/perl -w
#
#
#
$dir1= $ARGV[0];


while (defined ($file1 = glob ("$dir1/*.BlZ"))) {
	if ($file1 =~ /$dir1\/(\S+)_(\S+)\.BlZ/) {
		$name1= $1;
		$name2= $2;
		print $name1."\t";
		print $name2."\n";
		$res = $name1."_".$name2.".txt";
	}
	$count1= 0;
	open (FIL1, $file1);
	open (RES, ">$res");
	while (<FIL1>) {
		$linha1 = $_;
		if ($linha1 =~ /".*$name1.*"\s+\d+\s+\d+\s+(\d+)\s+\d+/) {
			#print $linha1."  1\n";
			$flag1= $1;
			#print $flag1[$count1]."<- flag1 \n";
		}
		if ($linha1 =~ /".*$name2.*"\s+\d+\s+\d+\s+(\d+)\s+\d+/) {
			#print $linha1."  2\n";
			$flag2= $1;
			#print $flag2[$count1]."<- flag2\n";
	}
		if ($linha1 =~ /\s+b\s(\d+)\s(\d+)/) {
			$count1++;
			$flag1[$count1]= $flag1;
			$flag2[$count1]= $flag2;
			$begin1[1][$count1]= $1;
			$begin1[2][$count1]= $2;
		}
		if ($linha1 =~ /\s+e\s(\d+)\s(\d+)/) {
			$end1[1][$count1]= $1;
			$end1[2][$count1]= $2;
			if ($begin1[1][$count1] > $end1[1][$count1]) {
				print "entrou!!\n";
				$aux= $end1[1][$count1];
				$end1[1][$count1]= $begin1[1][$count1];
				$begin1[1][$count1]= $aux;
			}
			if ($begin1[2][$count1] > $end1[2][$count1]) {
				$aux= $end1[2][$count1];
				print "entrou!!\n";
				$end1[2][$count1]= $begin1[2][$count1];
				$begin1[2][$count1]= $aux;
			}
		}
	}
	close (FIL1);
```

```
      for ($i = 1; $i <= $count1; $i++) {
            print RES
$name1."\t".$name2."\t".$flag1[$i]."\t".$begin1[1][$i]."\t".$end1[1][$i]."\t".$f
lag2[$i]."\t".$begin1[2][$i]."\t".$end1[2][$i]."\n";
      }
      close (RES);

}
```

script: parse_fileWgeneDistPerOrgan.pl

```
#! /usr/bin/perl -w

$file = $ARGV[0];
$expres = $ARGV[1];
$tissue = $ARGV[2];
$res = $ARGV[3];

open (FIL, $file);
open (RES, ">$res");
while (<FIL>) {
      $linha = $_;
      @data = split (/\t/, $linha);
      $gene = $data[0];
      open (EXPRE, $expres);
      while (<EXPRE>) {
            $linha2 = $_;
            if (($linha2 =~ /$gene/) && ($linha2 =~ /$tissue/)) {
                  print RES $linha;
            }
      }
      close (EXPRE);
}
close (FIL);
close (RES);
```

script: parse_hits_threshold2.pl

```
#! /usr/bin/perl -w
#
#
#
#
$file= $ARGV[0];
$thres= $ARGV[1];
$res1= $ARGV[2];
$res2= $ARGV[3];


open (FIL, $file);
open (RES1, ">$res1");
open (RES2, ">$res2");
while (<FIL>) {
      $linha= $_;
      if ($linha =~ /\S/) {
            @data = split (/\s+/, $linha);
            if ($data[@data - 2] > $thres) {
                  print RES1 $linha;
            } else {
                  print RES2 $linha;
            }
```

```
      }
}
close (RES1);
close (RES2);
close (FIL);
```

script: parse_megablast.pl

```
#! /usr/bin/perl -w
#
#
#
$alignment_size= 40;
$similarity= 70;
while (defined ($filename = glob ("*.Mblt"))) {
      $res= $filename."_res_70In40";
      print $res."\n";
      #$res= $filename."_res_75In60";
      #$res= $filename."_res_70In60";
      open (RES, ">$res");
      open (FIL, $filename);
      print $filename."\n";
      $count= 0;
      while (<FIL>) {
            $linha= $_;
            if ($linha =~ /^#/) {} else {
                  @data = split (/\t/, $linha);
                  if ( ($data[3] >= $alignment_size) && ($data[2] >=
$similarity) ) {
                        print RES $linha;
                  }
            }
      }
      close (FIL);
      close (RES);
}
```

script: parse_unigenes_expPats.pl

```
#! /usr/bin/perl -w

$file = $ARGV[0];
$res = $ARGV[1];

open (FIL, $file);
open (RES, ">$res");
while (<FIL>) {
      $linha = $_;
      if ($linha =~/^ID\s+(\S+)/) {
            #print $1."\n";
            $uniId = $1;
      }
      if ($linha =~ /EXPRESS\s+(.*)/) {
            $expr = $1;
            print RES $uniId."\t".$expr."\n";
            print $uniId."\t".$expr."\n";
      }

}
close (FIL);
close (RES);
```

script: parseHmmpsearchRes.pl

```
#! /usr/bin/perl -w
#
#
#
while (defined ($filename = glob("*.out"))) {
      if ($filename =~ /(\S+)\.out/) {
            $res= $1.".hmmsearch_prs";
      }
      open (RES, ">$res") || die "can't opne $res: $!";
      open (FIL, $filename) || die "can't opne $filename: $!";
      $escreve= 0;
      $pode= 0;
      RETORNO: while (<FIL>) {
            $linha = $_;
            if ($linha =~ /Alignments of top-scoring domains/) {
                  last RETORNO;
            }
            if ($linha =~ /^Parsed for domains/) {
                  $pode= 1;
                  next;
            }
            if ( ($pode == 1) && ($linha =~ /^--------/) ) {
                  $escreve= 1;
                  next;
            }
            if ($escreve == 1) {
                  print RES $linha;
            }
      }
      close (FIL) || die "can't close $filename: $!";
      close (RES) || die "can't close $res: $!";
      #system ("rm -f $filename");
}
```

script: prepareForblast.pl

```
#! /usr/bin/perl -w
#
#
#
$dir = $ARGV[0];
while (defined ($filename = glob ("$dir/*.fasta"))) {
      if ($filename =~ /$dir\/([a-z0-9\._]+).fasta/i) {
            $res= $1;
      }
      #PRepare for blast
      system ("formatdb -i $filename -p F -o T -n $res");
      print ("formatdb -i $filename -p F -o T -n $res\n");
}
```

script: remove_empty.pl

```
#! /usr/bin/perl -w
#
#
while (defined ($filename = glob ("*"))) {
      $count= 0;
      open (FIL, $filename);
```

```
      RETORNO:while (<FIL>) {
            $count++;
            if ($count > 0) {
                  last RETORNO;
            }
      }
      close (FIL);
      if ($count == 0) {
            unlink($filename);
      }
}
```

script: remove_exonic.pl

```
#! /usr/bin/perl -w


$exon= $ARGV[0];
$aboves= $ARGV[1];

#ler os exons
$count= 0;
open (EXON, $exon);
while (<EXON>) {
      $linha = $_;
      chomp $linha;
      @data = split (/\t/, $linha);
      $count++;
      $chr[$count]= $data[1];
      $ini[$count]= $data[5];
      $fim[$count]= $data[6];
      print $count." <-- count\n";
}
close (EXON);


#parse the binding sites
while (defined ($filename = glob ("$aboves/*.above"))) {
      if ($filename =~ /$aboves\/(\S+.above)/) {
            $res = $1.".NonExonic";
      }
      print $res."\n";
      if (-e $res) {} else {
            open (FIL, $filename);
            open (RES, ">$res");
            while (<FIL>) {
                  $linha = $_;
                  print $linha;
                  $escreve = 1;
                  $chr= "";
                  @data_aux = split (/\s+/,$linha);
                  $pos= $data_aux[0];
                  @data = split (/_/, $pos);
                  if (@data > 5) {
                                    $chr = $data[@data - 5]."_";
                        }
                  $chr = $chr.$data[@data - 4];
                  $ini = $data[@data - 3];
                  $fim = $data[@data - 2];
                  print $chr."|".$ini."|".$fim."\n";
                  $fim = $ini + $data_aux[3];
```

```
                  $ini = $ini + $data_aux[2];
                  print $chr."|".$ini."|".$fim."\n";
                  # check it
                  RETORNO: for ($i = 1; $i <= $count; $i++) {
                        if ($chr eq $chr[$i]) {
                              if ( ( ($ini <= $fim[$i]) && ($ini[$i] <= $ini) )
||
                                    ( ($ini[$i] <= $fim) && ($ini <= $ini[$i]) )
||
                                    ( ($fim <= $fim[$i]) && ($ini[$i] <= $fim) )
||
                                    ( ($fim[$i] <= $fim) && ($ini <= $fim[$i]) ) )
{
                                    $escreve= 0;
                                    #print
$linha."\t".$ini[$i]."\t".$fim[$i]."\t".$chr[$i]."\n";
                                    last RETORNO;
                              }
                        }
                  }
                  if ($escreve == 1) {
                        print RES $linha."\n";
                  }
            }
            close (RES);
            close (FIL);
      }
}
```

script: remove_repetitionFromSorted.pl

```
#! /usr/bin/perl -w
#
#
#
#
$ultimo= "";
while (<STDIN>) {
      $linha= $_;
      if ($linha eq $ultimo) {

      } else {
            print $linha;
            $ultimo= $linha;
      }
}
```

script: repeat_mask3V2.pl

```
#! /usr/bin/perl -w
#
#
$species= $ARGV[0];
#while (defined ($filename = glob ("*.fasta"))) {
while (defined ($filename = glob ("*.fa"))) {
      $res= $filename.".masked";
      if (-e $res) {

      } else {
            system ("RepeatMasker -species $species $filename");
            if (-e $res) {
```

```
                system ("rm -f *cat *out *tbl");
            } else {
                system ("cp $filename $res");
            }
            sleep 300;
        }
}
```