Multigene families from blast hits

Scott Edwards

making-multigene-family-maps-from-long-read-genomes

These scripts can help you get from a set of long-read contigs and blast hits to a nice figure of your multigene family

Introduction

This module will help you go from a set of long-read genome contigs to a simple representation of a comple multigene family like the major histocompatibility complex. This is not a substitute for full-on annotatio but it may be useful if you want a quick overview of a multigene family across several haplotypes of the sam species. We will use the scrub jay MHC as an example and we'll assume that you have already generate formatted tables of blast hits using the command-line blast. For this you will need to setup a blast databas of your genome to be blasted, as well as find the appropriate query sequences of genes you are interested i representing. I have found these resources to be helpful in doing this: https://www.ncbi.nlm.nih.gov/books NBK569856/ and https://www.metagenomics.wiki/tools/blast/blastn-output-format-6. For example, let imagine I have several genomes (fasta files) ending with the suffice "ren". On the Cannon cluster, you ca run the command:

```
module load blast/2.6.0+-fasrc01
for file in *ren*;
  do makeblastdb -in "$file" -dbtype nucl -out ${file%.*};done
```

You can then make an 'uber' database of all your genomes (in this case, the "MCZ" files) with this command:

```
blastdb_aliastool -dblist "MCZ_orn_365326.hap1.fa_ren MCZ_orn_365326.hap2.fa_ren MCZ_orn_365338.hap1.fa
MCZ_orn_366490.hap1.fa_ren MCZ_orn_366490.hap2.fa_ren MCZ_orn_366493.hap1.fa_ren MCZ_orn_366493.hap2.fa
-dbtype nucl -out scrubjay_9_haps -title "10 scrub jay renamed genomes combined"
```

Getting your blast output together

Now by blasting to this library (called "scrubjay_9_haps") you can blast to all the individual genome databases simultaneously. You will want to set your output format when you blast so that it generates a table that is easily manipulated. I did this by using the -outfmt "6" command, further specifying the precise columns to generate using the codes available here. So the commmand to generate a table for mhc class II exon 3 using part of this scrub jay sequence as a probe might look like this:

```
blastn -db scrubjay_26_haps -query Mhc_U23958.1_exon3.fa -out sj_mhc2_exon3_blast.out -task megablast -
```

Your output should be a 6 column table of blast hits and target sequence ids. You may want to add additional columns but we will mainly use qseqid, sseqid, sstart and send. Let's retrieve that file, as well as other gene blast tables we might use as anchors in our map, and look at them:

Preparing the blast outputs in R

```
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.1.0     v forcats 0.5.1
## v purrr   0.3.4

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggfittext)
library(gggenes)
library(RColorBrewer)
setwd("/Users/scott/OneDrive\ -\ Harvard\ University/Research/Scrub\ Jays/26_haps/blasts/gggenes\ layou
mhc3<-read.table("sj_mhc2_exon3_blast.out")
brd2<-read.table("sj_zfbrd2_26_blast.out")
slc39<-read.table("sj_slc39A7_26_blast.out")
dim(mhc3)
```

```
## [1] 285   6
```

```
head(mhc3)
```

```
##           V1                                V2     V3     V4     V5    V6
## 1 Aphelocoma MCZ_orn_366494.hap2.h2tg0011751 99.647  19876  19594 minus
## 2 Aphelocoma MCZ_orn_366494.hap2.h2tg0005601 99.301   5453   5738  plus
## 3 Aphelocoma MCZ_orn_366494.hap2.h2tg0005601 99.301  18085  18370  plus
## 4 Aphelocoma MCZ_orn_366494.hap2.h2tg0005601 99.301  38462  38747  plus
## 5 Aphelocoma MCZ_orn_366494.hap2.h2tg0005601 99.301  69228  69513  plus
## 6 Aphelocoma MCZ_orn_366494.hap2.h2tg0005601 99.301 112022 112307  plus
```

Let's put the gene (query) name in the first column of each table for clarity:

```
mhc3[,1]<-"MHC_classIIB_exon3"
brd2[,1]<-"BRD2"
slc39[,1]<-"SLC39A7"
```

Subsetting the scaffolds

Now let's find out which scaffolds have all three of the genes we are interested in, join the tables together and re-name the tables. The second column (V2) of each table has the scaffold hits:

```
corescaffs<-intersect(mhc3$V2,union(brd2$V2,slc39$V2))
corescaffs
```

```
##  [1] "MCZ_orn_366494.hap2.h2tg0011751" "MCZ_orn_366494.hap1.h1tg0012501"
##  [3] "MCZ_orn_366494.hap1.h1tg0008271" "MCZ_orn_366493.hap2.h2tg0004711"
##  [5] "MCZ_orn_366493.hap1.h1tg0000741" "MCZ_orn_366490.hap2.h2tg0008701"
##  [7] "MCZ_orn_366490.hap1.h1tg0004701" "MCZ_orn_365338.hap2.h2tg0006831"
##  [9] "MCZ_orn_365338.hap1.h1tg0007951" "MCZ_orn_365326.hap2.h2tg0004201"
## [11] "MCZ_orn_365326.hap1.h1tg0006791" "MCZ_Orn_366499.hap2.h2tg0007161"
## [13] "MCZ_Orn_366498.hap2.h2tg0000861" "MCZ_Orn_366498.hap1.h1tg0006591"
## [15] "MCZ_Orn_366488.hap2.h2tg0007901" "MCZ_Orn_366488.hap2.h2tg0007021"
## [17] "MCZ_Orn_366488.hap1.h1tg0009751" "MCZ_Orn_366488.hap1.h1tg0008431"
## [19] "MCZ_Orn_366488.hap1.h1tg0004271" "MCZ_Orn_366487.hap2.h2tg0006481"
## [21] "MCZ_Orn_365337.hap1.h1tg0008021" "MCZ_Orn_365336.hap2.h2tg0004551"
## [23] "MCZ_Orn_365336.hap1.h1tg0006321" "MCZ_Orn_365335.hap2.h2tg0004591"
## [25] "MCZ_Orn_365335.hap1.h1tg0007101" "MCZ_Orn_365327.hap2.h2tg0003731"
```

```
fulltab<-rbind(mhc3,brd2,slc39)
head(fulltab)
```

```
##                   V1                                V2     V3     V4     V5    V6
## 1 MHC_classIIB_exon3 MCZ_orn_366494.hap2.h2tg0011751 99.647  19876  19594 minus
## 2 MHC_classIIB_exon3 MCZ_orn_366494.hap2.h2tg0005601 99.301   5453   5738  plus
## 3 MHC_classIIB_exon3 MCZ_orn_366494.hap2.h2tg0005601 99.301  18085  18370  plus
## 4 MHC_classIIB_exon3 MCZ_orn_366494.hap2.h2tg0005601 99.301  38462  38747  plus
## 5 MHC_classIIB_exon3 MCZ_orn_366494.hap2.h2tg0005601 99.301  69228  69513  plus
## 6 MHC_classIIB_exon3 MCZ_orn_366494.hap2.h2tg0005601 99.301 112022 112307  plus
```

```
colnames(fulltab)<-c("gene","molecule","perc_ident","start","end","strand")
intertab<-fulltab %>% filter(molecule %in% corescaffs)
```

Now we have a table of scaffolds that at least have an MHC hit as well as a BRD2 or SLC39 hit. There are a total of 26 unique contigs/scaffolds in this table. The single-copy gene BRD2 in particular is a good anchor gene. SLC39 and RXRB are also good anchor genes, even if SLC39 and RXRB are small multigene families. Exon 4 of RXRB is a good low-copy conserved probe to delimit the passerine MHC class II region.

Putting contigs/scaffolds in a common orientation

I find it helpful now to determine the orientation of genes on each scaffold, since we want to display them all in the same orientation. I do this by making a table of hits ordered by position in scaffold, and then asking if the BRD2 is at the beginning or end of the scaffold (beginning or end is arbitrary). I want to identify scaffolds with BRD2 at the end, so I can reverse them, because I want BRD2 to be on the left end of each of my maps.

```
scaffs<-unique(intertab$molecule)
for(i in 1:length(scaffs)){
  list<-intertab %>% filter(molecule == scaffs[i]) %>% arrange(start)
  write.table(list,"scaff_arrange_tab.txt",sep = "\t",append = T,col.names = F,quote = F,row.names = F)
}
```

By inspecting the file scaff_arrange_tab.txt I found that 15 of the 26 scaffolds needed reversing. I put the names of these scaffolds in a list called strands.

```
strands<-c("MCZ_Orn_366498.hap2.h2tg000086l","MCZ_Orn_366498.hap1.h1tg000659l","MCZ_Orn_366487.hap2.h2tg
           "MCZ_Orn_365336.hap1.h1tg000632l","MCZ_Orn_365335.hap2.h2tg000459l","MCZ_orn_365326.hap2.h2tg
           "MCZ_orn_365326.hap1.h1tg000679l","MCZ_orn_366494.hap1.h1tg000827l","MCZ_orn_366493.hap2.h2tg
           "MCZ_orn_366493.hap1.h1tg000074l","MCZ_orn_366490.hap1.h1tg000470l","MCZ_Orn_366499.hap2.h2tg
           "MCZ_Orn_366488.hap2.h2tg000702l","MCZ_Orn_365337.hap1.h1tg000802l","MCZ_Orn_365327.hap2.h2tg
```

Now let's create new columns to either reverse or preserve the orientation of the start and end of blast hits, as appropriate:

```
`%notin%` <- Negate(`%in%`)
negall<-intertab[which(intertab$molecule %in% strands),]
notreverse<-intertab %>% filter(molecule %notin% strands)
reverseall<-mutate(negall,negstart = -start,negend = -end)
notreverse<-mutate(notreverse,negstart = start,negend = end)
reversetab2<-bind_rows(reverseall,notreverse)
```

We can inspect the order of genes on individual scaffolds now. For example:

```
reversetab2 %>% filter(molecule == "MCZ_orn_366494.hap1.h1tg000827l") %>% arrange(start)
```

```
##                 gene                          molecule perc_ident  start    end
## 1 MHC_classIIB_exon3 MCZ_orn_366494.hap1.h1tg000827l     99.301   9541   9826
## 2 MHC_classIIB_exon3 MCZ_orn_366494.hap1.h1tg000827l     99.301  22592  22877
## 3 MHC_classIIB_exon3 MCZ_orn_366494.hap1.h1tg000827l     99.647  41490  41208
## 4 MHC_classIIB_exon3 MCZ_orn_366494.hap1.h1tg000827l     99.647 105020 104738
## 5 MHC_classIIB_exon3 MCZ_orn_366494.hap1.h1tg000827l     99.647 124122 123840
## 6 MHC_classIIB_exon3 MCZ_orn_366494.hap1.h1tg000827l     99.647 145595 145313
## 7 MHC_classIIB_exon3 MCZ_orn_366494.hap1.h1tg000827l     99.647 164740 164458
## 8              BRD2 MCZ_orn_366494.hap1.h1tg000827l     85.099 218964 218665
##   strand negstart  negend
## 1   plus    -9541   -9826
## 2   plus   -22592  -22877
## 3  minus   -41490  -41208
## 4  minus  -105020 -104738
## 5  minus  -124122 -123840
## 6  minus  -145595 -145313
## 7  minus  -164740 -164458
## 8  minus  -218964 -218665
```

Some of the genes have negative positions, but that's ok. The fact that BRD2 has the lowest position (most negative) assures me that it will be on the left of the map when I plot it.

Making small genes on large contigs a bit easier to see

Now we need to find out which direction each gene is in so that we can add a bit more length to it in the appropriate orientation so that we can see individual genes better. This example involves contigs that are about 400 kb in length; by adding 3kb to the beginning and end of each gene we can see it better but we must remember that our diagram is now not strictly to scale.

```
norm2<-which(reversetab2$negstart < reversetab2$negend)
reversetab3<-mutate(reversetab2[norm2,],negstart03=negstart-3000,negend03=negend+3000)
reversetab4<-mutate(reversetab2[-norm2,],negstart03=negstart+3000,negend03=negend-3000)
reversetab5<-bind_rows(reversetab3,reversetab4)
```

Anchoring your scaffolds on a gene and common scale

Now let's decide to anchor our diagram on the BRD2 gene. We do this by constructing a dummy alignment using functions in gggenes and the columns indicating our expanded gene hit sizes. When we do this, we then need to focus only on those scaffolds that actually have the BRD2 gene. This will lower our scaffold number from 26 to 19.

```
dummies <- make_alignment_dummies(
  reversetab5,
  aes(xmin = negstart03, xmax = negend03, y = molecule, id = gene),
  on = "BRD2"
)
```

Now, in order to plot each haplotype on a common axis, we need to do some magic that I learned from David Wilcox, author of gggenes, who kindly answered some questions for me on his github site. Basically we are re-setting each gene on a common scale.

```
reversetab5 <- reversetab5 %>%
  group_by(molecule) %>%
  mutate(genes_min = min(c(negstart03, negend03))) %>%
  mutate(negstart03 = negstart03 - genes_min) %>%
  mutate(negend03 = negend03 - genes_min) %>%
  ungroup()
```

Plotting!

Finally, let's compile all the scaffolds that have the BRD2 gene and then plot just those.

```
brd2scaffs<-(reversetab5 %>% filter(gene == "BRD2"))$molecule
```

It will work better to make your R studio plot window large before running this chunk. If you get an error like "Error in grid.Call. . . " it probably means your R studio window is too small. Better to pass the plot to a variable and then use ggsave with a large output to save it. You can vary the thickness of the gene maps, the colors of the genes, and many other aspects using parameters in this scipt.

or

```
p<-reversetab5 %>% filter(molecule %in% brd2scaffs) %>% ggplot(aes(xmin = negstart03, xmax = negend03, 
  geom_gene_arrow(arrowhead_height = unit(3, "mm"), arrowhead_width = unit(1, "mm"),
                  colour = "blue", size = 0) +
  facet_wrap(~ molecule, scales = "free", ncol = 1) +
  scale_fill_brewer(palette = "Set3") +
  scale_x_continuous(labels = scales::comma, limits = c(0, 420000)) +
  theme_genes() %+replace% theme(axis.text = element_text(size = 10))

ggsave("Mhc_classII_gene_plot.pdf",p, height = 20, width = 10,limitsize = FALSE)
```

There are also ways to remove all but the top or bottom scale, although at this stage I just remove the unwanted scales in Illustrator.