Hudson redbreast landscape genetics using microsats

Import packages, data, and helper functions

Packages, helpers, and most of the data

```
library(tidyverse) # also imports readxl, so don't need to bring that in
library(cowplot)
library(adegenet) # for the pop gen
library(poppr) # assist with pop gen, particularly with piping
library(hierfstat) # run many pop gen stats
library(sf) # import spatial data
library(igraph) # for graph plots like genetic distance and physical distance
library(MuMIn) # for AICc
library(mmod) # for Jost D
library(nlme) # For linear mixed models
library(corMLPE) # for the correlation structure to account for pairwise comparisons

`%nin%` = Negate(`%in%`)

load('/workdir/smallmouth/hudson/redbreast/publication_files/hudson_data_import.rda')

# Helper function
  # Tibble should have "ind" as first column, then all subsequent columns are the ones to genera
te pairwise distances from. Give the name ("Type") of relationship being examined. this is a str
ing to name the output column (i.e. physical_distance, genetic_distance, adaptive_difference, ec
t)
  diffs<-function(tib_in, type){

    tib_in %>%
       column_to_rownames('ind') %>%
       na.omit() %>%
       ecodist::distance() %>%
       as.matrix() %>%
       as_tibble(rownames = 'ind1') %>%
       pivot_longer(-ind1, names_to = 'ind2', values_to = type) %>%
       rowwise() %>%
       mutate(ind12=paste0(sort(c(ind1,ind2)),collapse='---'))  %>%
       ungroup() %>%
       filter(!duplicated(ind12), ind1 != ind2) %>%
       dplyr::select(-c(ind1, ind2))
  }
```

Set up IBD dataframe

Genetic

```
# Genetic distance
    Fst_matrix<-as.matrix(genet.dist(obj,method="Dch")) # The recommended (Takezaki & Nei (1996) default, Cavalli-Sforza and Edwards Chord distance
    Fst_matrix[lower.tri(Fst_matrix, diag = T)]<-NA # set the lower diagonal as NA

    D_matrix<-as.matrix(pairwise_D(obj))
    D_matrix[lower.tri(D_matrix, diag = T)]<-NA

    Fst_tibble<-rownames_to_column(as.data.frame(Fst_matrix), var = 'pop') %>%
      pivot_longer(-pop, names_to = 'pop2', values_to = 'Fst')

    D_tibble<-rownames_to_column(as.data.frame(D_matrix), var = 'pop') %>%
      pivot_longer(-pop, names_to = 'pop2', values_to = 'D')

    Fst_D_tibble<-left_join(Fst_tibble, D_tibble, by = c('pop', 'pop2')) %>%
      filter(!is.na(D)) %>%
      rowwise() %>%
      mutate(pop12=paste0(sort(c(pop,pop2)),collapse='/')) %>%
      ungroup() %>%
      dplyr::select(-c(pop,pop2))
```

Physical distance and barriers between sites

```
# get cumulative number of barriers, and barrier types
    distance_between_sites_joiner<-distance_between_sites %>%
      mutate(barrier_index = as.character(barrier_index),
             up_down = paste0(upstream, '---', downstream),
             distance_corrected = if_else(order == 3, distance * 12.6,
                                          if_else(order == 4, distance * 2, distance)),
             distance_corrected = distance_corrected/1000)

    distance_between_sites_expanded<-distance_between_sites_joiner %>%
      separate(barrier_index, into = paste0('barrier_index-', 1:3), sep = ';') %>%
      dplyr::select(up_down, contains('barrier_index')) %>%
      pivot_longer(-up_down, values_to = 'barrier_index') %>%
      left_join(barriers, by = 'barrier_index') %>%
      filter(!is.na(barrier_index)) %>%
      group_by(up_down) %>%
      summarise(barrier_index = paste0(barrier_index, collapse = ';'),
                full_description = paste0(full_description, collapse = ';'),
                max_height = max(height_ft),
                max_gen_impounded = max(generations_impounded)) %>%
      ungroup() %>%
      rowwise() %>%
      mutate(n_intact = sum(str_count(full_description, 'intact dam')),
             n_waterfall = sum(str_count(full_description, 'waterfall')),
             n_beaver = sum(str_count(full_description, 'beaver')),
             n_removed = sum(str_count(full_description, 'removed')),
             n_intact_order3 = sum(str_count(full_description, 'intact dam-order3')),
             n_intact_order4 = sum(str_count(full_description, 'intact dam-order4')),
             n_intact_order5 = sum(str_count(full_description, 'intact dam-order5')),
             max_height = replace_na(max_height, 0),
             max_gen_impounded = replace_na(max_gen_impounded, 0)) %>%
      ungroup() %>%
      left_join(dplyr::select(distance_between_sites_joiner,
                              up_down,
                              distance,
                              distance_corrected,
                              confluence,
                              hudson_confluence,
                              barrier_n,
                              alt_change,
                              order), by = 'up_down') %>%
      mutate(river_distance_3 = if_else(order == 3, distance, 0),
             river_distance_4 = if_else(order == 4, distance, 0),
             river_distance_5 = if_else(order == 5, distance, 0)) %>%
      dplyr::select(-c(barrier_index, full_description, order)) %>%
      separate(up_down, into = c('upstream','downstream'), sep = '---')

# Run a pairwise generator to get all the different isolation by... measurements
    loop_out<-tibble()
    for(i in colnames(dplyr::select(distance_between_sites_expanded, -contains('stream')))){

      #i<-'barrier_n'
```

```
        all_edges<-distance_between_sites_expanded %>%
          dplyr::rename(x='upstream', y='downstream', weight=i) %>%
          dplyr::select(x, y, weight)


        all_nodes <- distance_between_sites_expanded %>%
          pivot_longer(cols = c(upstream, downstream), names_to = 'name', values_to = 'pop') %>%
          filter(!duplicated(pop)) %>%
          dplyr::select(pop)

        dist_igraph<-igraph::graph.data.frame(all_edges, vertices = all_nodes, directed = F)

        # print(dist_igraph, e=T, v=T)
        # jpeg(file="/workdir/smallmouth/hudson/redbreast/results/igraph-moodna.jpeg", width =
8, height = 9, units = 'in', res = 600)
        # plot.igraph(dist_igraph,
        #             mark.border = NA,
        #             vertex.label.cex = 0.4,
        #             vertex.size = 0.5,
        #             edge.label = all_edges$weight)
        # dev.off()

        # Just grab the upper half of the triangle
            mat<-igraph::shortest.paths(dist_igraph) %>%
              as.matrix()

            mat[lower.tri(mat, diag = T)]<-NA

            dist_export<-mat %>%
              as.data.frame() %>%
              rownames_to_column(var = 'pop') %>%
              pivot_longer(!pop, names_to='pop2', values_to = 'dist') %>%
              mutate(dist_type=i)

        loop_out<-bind_rows(loop_out, dist_export)
    }

# Bind together genetic and physical distance
    pairwise_physical_genet_dist<-loop_out %>%
      filter(!is.na(dist)) %>%
      pivot_wider(names_from = dist_type, values_from = dist) %>%
      write_csv('/workdir/smallmouth/hudson/redbreast/results/pairwise_physical_dist.csv') %>%
      rowwise() %>%
      mutate(pop12=paste0(sort(c(pop,pop2)),collapse='/')) %>%
      ungroup() %>%
      dplyr::select(-c(pop,pop2)) %>%
      left_join(Fst_D_tibble, by = 'pop12') %>%
      filter(!is.na(D))
```

build dataset

```
# create dataset
    order_joiner <- tibble(pop1_basin=c('Furnace','Cromline','Moodna','Satterly','upperMoodna'),
                           order = c(3,4,5,3,4))

# we're going to include cDG
  # cdg<-  GD.pop$cGD.gstudio %>%
  #     as.matrix() %>%
  #     #as.data.frame() %>%
  #     as_tibble(rownames = 'pop1') %>%
  #     pivot_longer(-pop1) %>%
  #     rowwise() %>%
  #     mutate(pop12 = paste0(sort(c(pop1, name)), collapse = '/')) %>%
  #     ungroup() %>%
  #     filter(!duplicated(pop12))

# Create overall ibd dataset
    ibd_all<-pairwise_physical_genet_dist %>%
      separate(pop12, into = c('pop1','pop2'), sep = '/', remove = F) %>%
      separate(pop1, into = c('pop1_river','pop1_basin','pop1_site'), sep = '-') %>%
      separate(pop2, into = c('pop2_river','pop2_basin','pop2_site'), sep = '-') %>%
      left_join(order_joiner) %>%
      mutate(fst_linear = Fst/(1-Fst),
           d_linear = D/(1-D),
           `Dam present` = if_else(n_intact>0, 'Yes', 'No'),
           `Stream order` = as.character(order),
           dist_km = distance/1000) %>%
      mutate(riv12 = paste0(pop1_river, pop2_river))

    ibd_in<-ibd_all %>%
      filter(riv12 != 'Furnace_brookMoodna')  # exclude the cross-boundary ones
```

Prepare data for export and review (don’t run unless adding new data)

- Note to reviewer: This code block does not need to be run, but organizes and cleans all the data used in the Rdata file imported above in the .rda file

panel design

Download sequence data from bioHPC

```
cd /workdir/smallmouth/hudson/redbreast/redbreast_msatlibrary_sequences

# the following is from cornell biohpc
wget -q -c -O 13191_2229_160172_GBC72_Laur_msats_ATTACTCG_TATAGCCT_R1.fastq.gz "http://cbsuapps.
biohpc.cornell.edu/Sequencing/showseqfile.aspx?mode=http&cntrl=1669757443&refid=942290"
wget -q -c -O 13191_2229_160172_GBC72_Laur_msats_ATTACTCG_TATAGCCT_R2.fastq.gz "http://cbsuapps.
biohpc.cornell.edu/Sequencing/showseqfile.aspx?mode=http&cntrl=2081976143&refid=942291"
```

Run contig assembly / alignment

Note - each contig is a separate microsat locus. Can visually scan the sequence in the final haplotype file to confirm that each actually has an msat

Steve uses SeqMan NGen assembly tool - 2 week trial or paid version

- I've emailed Qi Sun to see if there's an open-source command line substitute - STACKS or SoapDenovo?
- Use transcriptome option, which is better set up to deal with the uneven sequencing across the genome
- de novo option, no reference genome
- Set mer (size of fragment used for assembly) to 99, the highst level possible. This reduces the chance that one locus will get split into two
- Reverse transcribe the fastq files and submit these too
- Remove contigs below 100bph
- Reduce the gap penalty and increase the max gap size so that both alleles from a locus get captured

Some general primer notes (https://support.illumina.com/bulletins/2016/04/adapter-trimming-why-are-adapter-sequences-trimmed-from-only-the--ends-of-reads.html (https://support.illumina.com/bulletins/2016/04/adapter-trimming-why-are-adapter-sequences-trimmed-from-only-the--ends-of-reads.html))

- the illumina adapter is annealed to both ends, followed by barcodes (outside the adapters), and sequencing starts right inside of the adapter
- If the fragment is too short, the adapter on the 3' end will get read - hence we need adapter clipping
- goal is for fragment size to be short enough to get some overlap in the middle, but long enough to not have the sequencer spill over to adapter
- i5/i7 and adapter are not read in the sequencer, so when we say we are sequencing 250pe (for example), that is all primer and insert
- Avoid ordering primers for the same locus, this is only an issue for the same motifs unless there are two msats in the same fragment

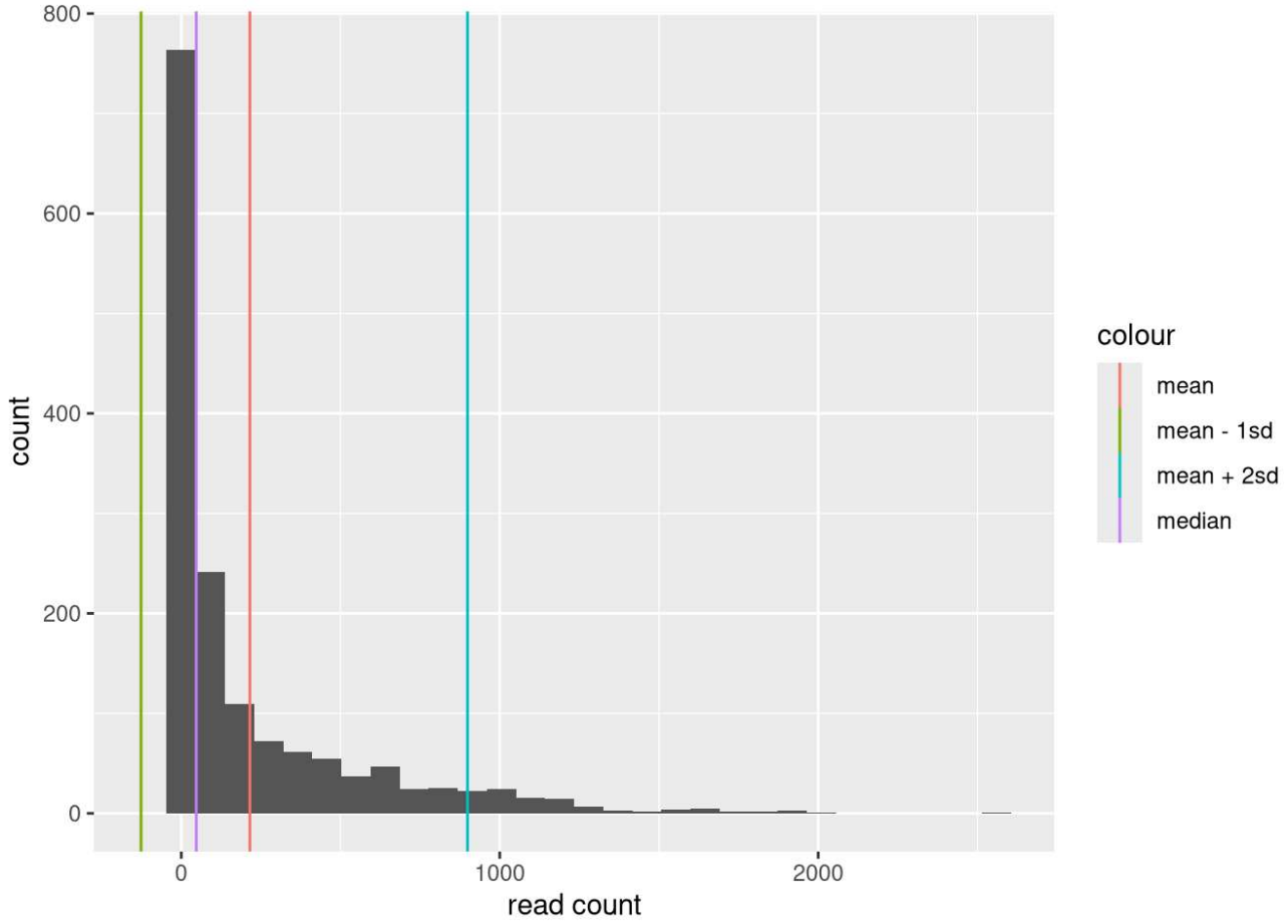
Identify microsats

Notes about microsat filtering process, implemented in below code chunk

- Steve uses msatcommander, which unfortunately is just for mac - I'll try krait
- Target 190-210bp (150pe sequencing) or 390-410bp (250pe sequencing) fragments for msats
- 250pe is only available on MiSeq, 150pe is available on all platforms - only downside is we may lose data on some very long alleles, but this is unlikely. Going 150pe allows us to scale up to a NextSeq

- Goal is to get around 500x coverage per locus and individual, assume half of this will get rejected, hopefully end up with 200x coverage in genotype table
- Design primers to melt at 60C
- Shoot for 25% to 50% GC content in the motif
- Shoot for intermediate read count (coverage depth), I can do 1sd or 2sd from the mean
- Repeat count from 10-13 is a good range
- the first section in the output from msatcommander is unique regions, the next section is “potentially duplicated”, don’t use this
- maximize motif diversity

```
# import msatcommander output spreadsheet, select 25% to 50% GC content
msat<- msat_commander_output %>%   # going shorter fragments for 150pe
  slice_head(n=1672) %>%  # this gets rid of the bottom part of the spreadsheet, which is "poten
tially duplicated primers"
  filter(between(pair_product_size,190,210)) %>% # This is just a double-check that all products
are in our ideal size range
  mutate(gc_content = str_count(motif, "G|C")/4) %>%  # Make a new column that detects proportio
n of G's and C's
  filter(between(gc_content,0.25,0.5)) # Exclude anything besides GC content of 25% or 50%

# Generate a histogram of read count so we can exclude outliers
  msat %>%
    ggplot(aes(`read count`)) +
    geom_histogram() +
    geom_vline(aes(xintercept = median(msat$`read count`), color='median')) +
    geom_vline(aes(xintercept = mean(msat$`read count`), color='mean')) +
    geom_vline(aes(xintercept = (mean(msat$`read count`)+sd(msat$`read count`)*2), color='mean +
2sd')) +
    geom_vline(aes(xintercept = (mean(msat$`read count`)-sd(msat$`read count`)), color='mean - 1
sd'))
```




```
## [1] "Well that's gross - super inflated low-end distribution. Lets just shoot for read depth
between the median (47) and 2sd above the mean (899)"
```

```
  msat_readCount<-msat %>%
    filter(between(`read count`, median(msat$`read count`), (mean(msat$`read count`)+sd(msat$`re
ad count`)*2)))

# Next, shoot for repeat count between 10 and 13
  msat_readCount_repeatCount<-msat_readCount %>%
    filter(between(`repeat count`, 10, 13))

# Finally, we can see how much motif diversity we have
  msat_readCount_repeatCount %>%
    group_by(motif) %>%
    summarise(n=n())
```

```
## # A tibble: 8 × 2
##   motif     n
##   <chr> <int>
## 1 AAAG     12
## 2 AACT      1
## 3 AATC      5
## 4 AATG      2
## 5 ACAG      6
## 6 ACTC      5
## 7 AGAT     16
## 8 ATCC     21
```

```
# As I have already ordered one set, exclude these primers from the set, and order another 15 ra
ndom ones
  # Import previous batch and set the column Contig to be in the same format as the working file
currently. Then exclude
  previous_batch<-msat_batch_1 %>%
    mutate(Contig=str_replace(Contig, 'Laur_f_',''),
           Contig=str_replace(Contig, 'Laur_r_',''),
           Contig=paste0('Contig',Contig))

  msat_readCount_repeatCount<-anti_join(msat_readCount_repeatCount, previous_batch, by='Contig')

  # Randomly select two occurences out of each motif. If there's just one occurence (else statem
ent), then keep that one occurence of the motif
      motif_subset_out<-tibble()
      for(i in unique(msat_readCount_repeatCount$motif)){
        print(paste0("motif ",i," has ", nrow(filter(msat_readCount_repeatCount, motif==i)), " o
ccurence"))
        if(nrow(filter(msat_readCount_repeatCount, motif==i))>1){
            motif_subset_in<-msat_readCount_repeatCount %>%
                            filter(motif==i) %>%
                            slice_sample(n=3, replace = F)
            motif_subset_out <-bind_rows(motif_subset_in, motif_subset_out)
        } else{
          motif_subset_in<-msat_readCount_repeatCount %>%
                            filter(motif==i)
            motif_subset_out <-bind_rows(motif_subset_in, motif_subset_out)
        }

      }
```

```
## [1] "motif AAAG has 12 occurence"
## [1] "motif AGAT has 16 occurence"
## [1] "motif AATC has 5 occurence"
## [1] "motif ATCC has 21 occurence"
## [1] "motif ACAG has 6 occurence"
## [1] "motif ACTC has 5 occurence"
## [1] "motif AATG has 2 occurence"
## [1] "motif AACT has 1 occurence"
```

```
    # Open this here and make sure to filter this list to the ones I actually ordered - batch2 h
ad two extra loci that I didn't end up ordering, Contig92 and Contig 4481
```

How much have we filtered our microsats?

```
## [1] "After excluding the potentially duplicated primers and filtering for 25% to 50% GC conte
nt, we have 1542 loci."
```

```
## [1] "After cutting out read depths below 47 and above 899 we have 676 loci"
```

```
## [1] "After keeping only repeat counts of 10-13 we have 68 loci"
```

```
## [1] "We randomly subset the 8 motifs to end with 21 loci"
```

```
## [1] "The final dataset is exported as csv, Laur_Primers_msats_190_210_3May22_filtered.csv, wi
th batch number listed"
```

If this doens't give us enought regions, expand repeat count to 8-15. If that doesn't work, decrease depth coverage (from 40s to 20s and 30s). don't want to go up, this could lead to microsats that show up in more than one placein the genome!

double check that the selected microsats aren't the same contig

Steve did this, but I could take the contig that each of the selected microsats are on and blast them against all other contigs of selected microsats. Nothing should be above ~80% similarity

Desiging primers

- Put a Nextera tag on the 5' end of all primers (forward and reverse). Barcodes will attach upstream from this
- Order the primers with the nextera tags attached, and then we have the barcodes already in the fridge to attach to each individual
- See resources in /workdir/smallmouth/hudson/redbreast/primer_ordering_resources
- Generate a .txt file with two columns (tab-delimited): primer name [species]*[F or R]*[ID number], and sequence [nextera tag, primer]. Remember that the tags are different for forward and reverse
- "Left sequence" is the forward, "Right sequence" is the reverse. Don't need to be reverse complemented

```
forward_nextera_tag<-'TCGTCGGCAGCGTCAGATGTGTATAAGAGACAG'
reverse_nextera_tag<-'GTCTCGTGGGCTCGGAGATGTGTATAAGAGACAG'

msat_batch_2_order<-motif_subset_out %>%
  mutate(`F`=paste0(forward_nextera_tag,left_sequence),
         R=paste0(reverse_nextera_tag, right_sequence),
         Contig=str_replace(Contig, 'Contig','')) %>%
  dplyr::select(Contig, `F`,R) %>%
  pivot_longer(!Contig, names_to = 'forward_reverse', values_to = 'seq') %>%
  mutate(label=paste0('Laur_',forward_reverse,'_',Contig)) %>%
  dplyr::select(label, seq) %>%
  write_tsv('/workdir/smallmouth/hudson/redbreast/primer_design/Laur_Primers_msats_190_210_3May2
2_filtered_idt_batch2.tsv', col_names = F) # export here to place order for IDT
```

Test if the primer pairs have any hairpins or primerdimers

autodimer

Starting with the software AutoDimer (https://www-s.nist.gov/dnaAnalysis/primerToolsPage.do (https://www-s.nist.gov/dnaAnalysis/primerToolsPage.do)) This is a website GUI which can handle up to 100 sequences and up to 75 nucleotides The input file needs to be designed as a multifasta

```
batches<-msat_batch_2_order %>%
  dplyr::rename(Contig=label, sequence=seq) %>%
  bind_rows(msat_batch_1)

seqinr::write.fasta(sequences = as.list(batches$sequence), names = batches$Contig, file.out = '/
workdir/smallmouth/hudson/redbreast/primer_design/batch12_primers_adapters.fasta') # upload to A
utoDimer to check
```

When uploaded to AutoDimer website, we didn't get any hairpins, but we did get 5 primerdimers. However, the Tm was low (max 30C) so I'm not worried about it.

Steve found that there are actually many primerdimers with the thermofisher primerdimer detection website, so I ended up decrasing the threshold from 7 to 4, and ended up finding quite a few on the AutoDimer website as well.

primerpooler

Testing PrimerPooler (https://biohpc.cornell.edu/lab/userguide.aspx?a=software&i=918#c (https://biohpc.cornell.edu/lab/userguide.aspx?a=software&i=918#c))

first I need to re-format the contig names

```
batches2<-batches %>%
  separate(Contig, into = c('spp', 'direciton', 'code'), sep = '_') %>%
  mutate(Contig = paste0(spp,'_',code,'_',direciton))

seqinr::write.fasta(sequences = as.list(batches2$sequence), names = batches2$Contig, file.out =
'/workdir/smallmouth/hudson/redbreast/primer_design/batch12_primerpooler.fasta')
```

```
/programs/PrimerPooler/pooler/pooler
# /workdir/smallmouth/hudson/redbreast/primer_design/batch12_primerpooler.fasta
# deltaG temperature is 37C (not sure why), 50C is less conservative (I get better deltaG number
s)
# saved file of dimers worse than -5 are /workdir/smallmouth/hudson/redbreast/primer_design/prim
erpooler_50C.txt

# This section didn't work, but maybe it will with the SMB
# amplicon overlaps /workdir/smallmouth/hudson/redbreast/redbreast_msatlibrary_sequences/Steves_
assembly_Laur_msats_2May22_contigs.fasta
# Max amplicon length 600
# write file of location of all amplicons /workdir/smallmouth/hudson/redbreast/primer_design/pri
merpooler_locations.txt
```

parse the output - get the problem primerpairs and the deltaG

```
primerpooler<-primerpooler[seq(1,nrow(primerpooler),by=7),] %>%
  separate(input, into = c('first','second'), sep = ' versus ')

bind_rows(tibble(input = primerpooler$first), tibble(input = primerpooler$second)) %>%
  group_by(input) %>%
  summarise(n=n())
```

```
## # A tibble: 9 × 2
##   input            n
##   <chr>        <int>
## 1 Laur_F_948       8
## 2 Laur_R_11108     1
## 3 Laur_R_2045      1
## 4 Laur_R_248       1
## 5 Laur_R_2546      1
## 6 Laur_R_3437      1
## 7 Laur_R_5428      1
## 8 Laur_R_6012      1
## 9 Laur_R_948       1
```

```
# looks like we will have problems with 948
```

Some notes on ordering primers

- Note - for batch 2, I had 18 primer pairs, but only wanted 15. I deleted the last three primer pairs.

- In eSHOP, search for “Integrated DNA technologies”, open the IDT portal, under “Custom synthesis”, choose “custom DNA oligos,” chose “bulk input”, then ***this is important, as importing a tsv, csv, or txt file doesn’t work*** copy/paste your tab delimited primers file into the bulk input window. Click submit to place the order. Don’t change scale, purification, these stay as defaults. If needed I could make new columns for scale and purification - but for now I am ordering them dry, not wet
- When the primers arrive, they are dry and stable at room temp, although Steve stores them in the fridge until ready to dilute them in buffer. After that, they live at -20
- In the future I should order them wet, not dry. # Getting multiplexed sequence back and calculating genotypes
- Reviewer note: For ease of reviewing, I have supplied the data on genotype codes, rather than raw reads, thus this section will not run. I am happy to provide raw reads if desired.

download sequence from the BRC

Save download script in the directory where the files are to be located, cd to this directory, run script

```
cd /workdir/backup/redbreast/fastq
sh ./download.sh
```

See if we have any problem primerdimers with nate’s primerdimer check

combine reads

```
cd /workdir/backup/redbreast/
zcat *R1.fastq > all_samples_R1.fastq
zcat *R2.fastq.gz > all_samples_R2.fastq
mkdir primer_interaction_test
mv all_samples* primer_interaction_test

# trim adapter sequences
java -jar /programs/trimmomatic/trimmomatic-0.39.jar PE -phred33 -threads 20 /workdir/backup/red
breast/primer-interaction-test/all_samples_R1.fastq /workdir/backup/redbreast/primer-interaction
-test/all_samples_R2.fastq /workdir/backup/redbreast/primer-interaction-test/all_samples_R1_adap
ter_clipped_paired.fastq /workdir/backup/redbreast/primer-interaction-test/all_samples_R1_adapte
r_clipped_unpaired.fastq /workdir/backup/redbreast/primer-interaction-test/all_samples_R2_adapte
r_clipped_paired.fastq /workdir/backup/redbreast/primer-interaction-test/all_samples_R2_adapter_
clipped_unpaired.fastq 'ILLUMINACLIP:'/programs/trimmomatic/adapters/NexteraPE-PE.fa':2:30:10' L
EADING:3 TRAILING:3 SLIDINGWINDOW:4:15 MINLEN:36

# combine the forwrad and reverse reads
cd /workdir/backup/redbreast/primer_interaction_test/
perl /workdir/smallmouth/gtseq/GTseek_utils-Main/paste_fastq.pl all_samples_R1_adapter_clipped_p
aired.fastq all_samples_R2_adapter_clipped_paired.fastq > all_samples_R1_R2_adapter_clipped_pair
ed.fastq
```

hash seqs

```
cd /workdir/backup/redbreast/primer_interaction_test/
perl /workdir/smallmouth/gtseq/GTseek_utils-Main/HashSeqs.pl all_samples_R1_R2_adapter_clipped_p
aired.fastq > all_samples_R1_R2_adapter_clipped_paired.fastq.hash
```

Here I would normally generate a tsv of LocusName FWD-Primer-sequence REV-Primer-sequence However, this is already calculated as an input of the amplicon.py protocol below, so I can just use it However, I do need to make sure that it has headers, so change a bit

```
read_tsv('/workdir/smallmouth/hudson/redbreast/multiplex_sequence/amplicon_py/key_rs.tsv', col_n
ames = c('LocusName', 'FWD-primer-sequence', 'RVS-Primer-sequence')) %>%
  write_tsv('/workdir/smallmouth/hudson/redbreast/multiplex_sequence/amplicon_py/key_rs_primer-i
nteraction-test.tsv')
```

Run the primer interaction test

```
cd /workdir/backup/redbreast/primer_interaction_test/
perl /workdir/smallmouth/gtseq/GTseek_utils-Main/GTseq_Primer-Interaction-Test_v3.pl /workdir/sm
allmouth/hudson/redbreast/multiplex_sequence/amplicon_py/key_rs_primer-interaction-test.tsv all_
samples_R1_R2_adapter_clipped_paired.fastq.hash > all_samples_R1_R2_adapter_clipped_paired.fast
q.hash_primer_interaction.txt

# then, filter it for the important stuff
grep -A20 '[Pp]rimer\|[Bb]lack' all_samples_R1_R2_adapter_clipped_paired.fastq.hash_primer_inter
action.txt > all_samples_R1_R2_adapter_clipped_paired.fastq.hash_primer_interaction_truncated.tx
t
```

5096 OR 1990 5096 OR 1953 Contig2526 Contig2171 Contig1953 Contig5096 Contig1990 Contig5096

Looking at this, I should exclude 5096 (shows up in multiple dimers), but keep 1990 2171, but keep 2526

Also, based on the read counts of the bad loci, most of them didn't amplify, so exclude all of the loci on the bad list except Contig516 and Contig8425, which got a little bit of amplification:

Contig7786 Contig4719 Contig3437 Contig836 Contig948 Contig11108

Prep directory and input files

key file is a tsv file, with 1)locus 2)F_primer 3)R_primer without nextera adapters in order to run amplicon.py (BRC software)

```
# here, filter this for the ones I actually ordered then export
ordered12 %>%
  separate(Contig, into  = c('spp','direction','number'), sep = '_') %>%
  mutate(Contig = paste0('Contig',number)) %>%
  dplyr::select(Contig) %>%
  filter(duplicated(Contig)) %>%
  left_join(batch12, by = 'Contig') %>%
  write_tsv('/workdir/smallmouth/hudson/redbreast/multiplex_sequence/amplicon_py/key_rs.tsv', co
l_names = F)
```

Move fastq files to amplicon_py

```
cd /workdir/backup/redbreast/fastq
cp *.gz /workdir/smallmouth/hudson/redbreast/multiplex_sequence/amplicon_py

# NOTE - I RENAMED SOME FILES AFTER TRANSFERRING OVER LANE 2 - CHANGING THE CONTROL AND BG SO TH
AT THE LENGTH MATCHES THE RS
```

sample table tsv: 1)Sample Name; 2) another unique string 3) Paired-end sequence file 1 (fastq or fastq.gz); 4) Paired-end sequence file 2

```
# add the new plates to the below sheet in lab notebook, and change (for example) N701 to nucleo
tides
sample_table<-demultiplex_codes %>%
  mutate(unique_id = paste0(LabCode,N7, N5),
         unique_id = str_replace_all(unique_id, '_', '')) %>%
  dplyr::select(unique_id, LabCode, Plate)

as_tibble(list.files(path = '/workdir/smallmouth/hudson/redbreast/multiplex_sequence/amplicon_p
y', pattern = '.gz')) %>%
  # filter(str_detect(value, 'CT_0')) # I can run this just to see if the negative controls got
any reads. If they show up here they still may be zero - if they aren't they certainly are zero
  mutate(unique_id = str_sub(value, 26,51), # first call in unique ID string, but with the rando
m a or b
         unique_id = str_replace_all(unique_id, '_', ''),
         end=str_sub(value,-11,-10)) %>%
  pivot_wider(names_from=end, values_from=value)  %>%
  left_join(sample_table, by = 'unique_id') %>%
  mutate(unique_id = paste0(unique_id, '-plate', Plate),
         sample_name = paste0(LabCode, '-plate', Plate)) %>%
  dplyr::select(sample_name, unique_id, R1, R2) %>%
  write_tsv('/workdir/smallmouth/hudson/redbreast/multiplex_sequence/amplicon_py/sample_table_r
s.tsv',col_names = F)
```

run amplicon.py

The goal of this first pass is to ID the most common haplotypes for each primer pair, check that each is on the right contig, and edit the reference fasta (if needed) to subset for just the haplotypes we want

info here https://bitbucket.org/cornell_bioinformatics/amplicon/src/master/ (https://bitbucket.org/cornell_bioinformatics/amplicon/src/master/)

To get amplicon.py working, I needed to install Bio with the following command (conda install didn't work) python3 -m pip install Bio

may also need to install cutadapt python3 -m pip install cutadapt

inputs -c # nuber of individuals an allele has to occur for it to be considered to be a real allele. if -r is 1, every genotye error will be included as an allele (steve changes to 2) -r is the ratio of reads to be considred heterozygous (default of 20 is way too conservative, steve does 5-9, Diana does 5) -l 50 (low size filter, can filter out dimers) -j is number of cores -a minimum maf (default is 0.01, but steve recommends no higher than 1/2n. we have 620 inds, so set at 8e-4 (0.0008))

```
cd /workdir/smallmouth/hudson/redbreast/multiplex_sequence/amplicon_py/

# remove the old amp_py_unfiltered
rm -r amp_py_unfiltered/

# Then run amplicon.py
# git clone https://bitbucket.org/cornell_bioinformatics/amplicon.git
export PATH=/programs/cutadapt-4.1/bin:/programs/muscle:/programs/bbmap-39.01:$PATH
export PYTHONPATH=/programs/cutadapt-4.1/lib/python3.9/site-packages:/programs/cutadapt-4.1/lib6
4/python3.9/site-packages
amplicon/amplicon.py -s sample_table_rs.tsv -k key_rs.tsv -o amp_py_output -a 0.0008 -c 2 -l 50
-r 5 -j 40

# for more informatoin on sequences, try amplicon/amplicon.py -h
# If for some reason this isn't working (ie the topallelefasta isnt getting generated), check th
e output and make sure that the software is finding all packages (ie cutadapt)
# can now delete fastas in multiplex_sequence

# copy this into a new folder so that we can look at the haplotypes for other species (hybridiza
ion)
cp -R amp_py_output amp_py_unfiltered
```

Filter out off-target reads with amplicon.py filter

the first step is to limit the microsat reference to just the sequences I'm interested in. The TopHaplotypeAllele is the sequence between the two primers, so the overall idea is to subset the sequence between each of the ordered primers after removing the nextera tags

```
# read in reference and ordered congits
full_reference<-Biostrings::readDNAStringSet('/workdir/smallmouth/hudson/redbreast/genome/Steves_assembly_Laur_msats_2May22_contigs.fasta')

successful_seuqnces<-Biostrings::readDNAStringSet('/workdir/smallmouth/hudson/redbreast/multiplex_sequence/amplicon_py/amp_py_unfiltered/topHaplotypeAllele.fasta')

ord_prim<-Biostrings::readDNAStringSet('/workdir/smallmouth/hudson/redbreast/primer_design/batch12_primerpooler.fasta')

ord_prim_tib<-tibble(names=names(ord_prim), seq = as.character(ord_prim)) %>%
  filter(!duplicated(names)) %>%
  separate(names, into = c('Laur','number','direction'), sep = '_') %>%
  mutate(name=paste0('Contig',number)) %>%
    dplyr::select(name, seq, direction) %>%
  pivot_wider(names_from=direction, values_from=seq) %>%
  filter(name %in% names(successful_seuqnces))

seq_between_primers<-as_tibble(names(full_reference)) %>%
  mutate(seq = as.character(full_reference)) %>%
  separate(value, into = c('name','reads','xx','length'), sep = ' ', remove = F) %>%
  dplyr::select(name, seq) %>%
  right_join(ord_prim_tib, by = 'name') %>%
  mutate(`F` = str_replace(`F`, 'TCGTCGGCAGCGTCAGATGTGTATAAGAGACAG',''),
         R = str_replace(R, 'GTCTCGTGGGCTCGGAGATGTGTATAAGAGACAG',''),
         R = as.character(Biostrings::reverseComplement(Biostrings::DNAStringSet(R)))) %>%
  rowwise() %>%
  mutate(seq = str_replace(seq, as.character(`F`), '--'),
                       seq = str_replace(seq, as.character(R), '--')) %>%
  ungroup() %>%
  separate(seq, into = c('before','between_primers','after'), sep = '--') %>%
  dplyr::select(-c(before, after))

seqinr::write.fasta(names = as.list(seq_between_primers$name), sequences = as.list(seq_between_primers$between_primers), file.out = '/workdir/smallmouth/hudson/redbreast/results/reference_between_primers.fasta')

write_csv(seq_between_primers, '/workdir/smallmouth/hudson/redbreast/results/sequence_between_primers.csv')
```

Then, actually filter the haplotype calls, removing ones that don't match the reference sequence -r can provide the reference fasta (/workdir/smallmouth/hudson/redbreast/genome/Steves_assembly_Laur_msats_2May22_contigs.fasta). If this is not given, it will use the top allele (topHaplotypeAllele.fasta)

```
python3 -m pip install swalign
# for more information on arguments, try amplicon/amplicon_filter.py -h
cd amp_py_output

amplicon_filter.py -i HaplotypeAllele.fasta  -f 2 -r /workdir/smallmouth/hudson/redbreast/results/reference_between_primers.fasta
```

Finally, use the filtered sequences to re-call genotypes, excluding the removed alleles (I think this just records the ind/locus as NA?)

```
cd /workdir/smallmouth/hudson/redbreast/multiplex_sequence/amplicon_py/
amplicon/amplicon.py -s sample_table_rs.tsv -k key_rs.tsv -a 0.0008 -c 2 -l 50 -r 5 -j 40 -i 1 -
g amp_py_output/amplicon.kept.fasta -o amp_py_output
```

Quality control

Contamination check: look at how many alleles pass allelic ratio and depth filter, shouldn’t be more than 2 /

locus / ind

```
ar_thresh <- 0.2 # allelic ratio
read_depth_thresh <- 20 # read depth
paralog_thresh = 0.20 # xxx % of genotypes within a loci must be below this threshold to not be
considered a paralog
sum_contam_loci_thresh <- 2 # Individuals which have more than xxx 3+ allele loci are excluded

# tbs_out<-tibble()
#
# for(i in list.files(path='/workdir/smallmouth/hudson/redbreast/multiplex_sequence/amplicon_py/
amp_py_output/tagBySampleDir/', pattern='tbs', full.names = T)){
#   try(
#     tbs_out<-read_tsv(i, col_names = c('ind','contig','seq','reads')) %>%
#     group_by(contig) %>% mutate(max_read=max(reads)) %>% ungroup() %>%
#     mutate(AR = reads/max_read) %>%
#     dplyr::select(-seq) %>%
#     bind_rows(tbs_out)
#   )
# }
#
# write_csv(tbs_out, '/workdir/smallmouth/hudson/redbreast/results/reads_per_allele_ind_contamin
ation.csv')

tbs_out<-reads_per_allele_ind %>%
  filter(str_sub(ind, 1, 7) %in% phenos$ind) %>% # Just grab the 2022 fish
  filter(reads>read_depth_thresh,
         AR > ar_thresh) %>%
  group_by(ind, contig) %>%
  tally() %>%
  ungroup() %>%
  filter(str_detect(ind, 'RS')) %>%
  mutate(contam_paralog=if_else(n<3, 0, 1))

# Here, make a list of the loci that exceed out threshold
  paralogs<-group_by(tbs_out, contig) %>%
    summarise(perc_paralog = mean(contam_paralog)) %>%
    filter(perc_paralog > paralog_thresh)

  write_csv(paralogs, '/workdir/smallmouth/hudson/redbreast/results/paralogs.csv')

# Next, remove tnose loci, and exclude individuals which show contamination
  filter(tbs_out, contig %nin% paralogs$contig) %>%
  group_by(ind) %>%
    summarise(sum_contam_loci = sum(contam_paralog)) %>%
    filter(sum_contam_loci >= sum_contam_loci_thresh) %>%
    write_csv('/workdir/smallmouth/hudson/redbreast/results/contam_ind.csv')
```

check if the duplicated individuals genotype similarly, then remove duplicated individuals

```
duplicated_inds<-paste0(c('RS_0347', 'RS_0561', 'RS_0635'), collapse = '|') # here's our list of
duplicated individuals

acc_check<-hap_genotype_noParalog_noContam %>%
  filter(str_detect(ind, duplicated_inds)) %>% # Here are the list of duplicated individuals
  mutate(dup = str_sub(ind, 1,7)) %>%
  arrange(dup) %>%
  group_by(dup) %>%
  mutate(n=row_number()) %>%
  ungroup() %>%
  dplyr::select(c(ind, locNames(obj), dup, n)) # just filter the the loci we feel good about

acc_output<-tibble()
for(i in unique(acc_check$dup)){
  dat_in<-acc_check %>%
    filter(dup == i)

  pairwise_combos<-combn(dat_in$ind, 2) # generate all pairwise combinations of the duplicated i
ndividuals

  for(ii in 1:ncol(pairwise_combos)){
    ind1<-pairwise_combos[1,ii] # grab the first indivdiual in the comparison...
    ind2<-pairwise_combos[2,ii] # and the second individual

    acc_in<-dat_in %>%
      filter(ind == ind1 | ind == ind2) %>%
      dplyr::select(-c(dup, n)) %>%
      pivot_longer(-ind) %>%
      pivot_wider(names_from=ind, values_from=value) %>%
      mutate(same = if_else(.[[2]] == .[[3]], 1, 0)) %>% # The 2nd and 3rd column need to match,
or else the call is wrong
      filter(!is.na(same)) %>%
      group_by(same) %>%
      tally() %>%
      mutate(ind1 = ind1, ind2 = ind2)

    acc_output<-bind_rows(acc_output, acc_in)

  }
}

acc_output %>%
  group_by(same) %>%
  summarise(n = sum(n)) %>%
  pivot_wider(names_from=same, values_from=n) #%>%  # The below doesn't run because we have 100%
accuracy
```

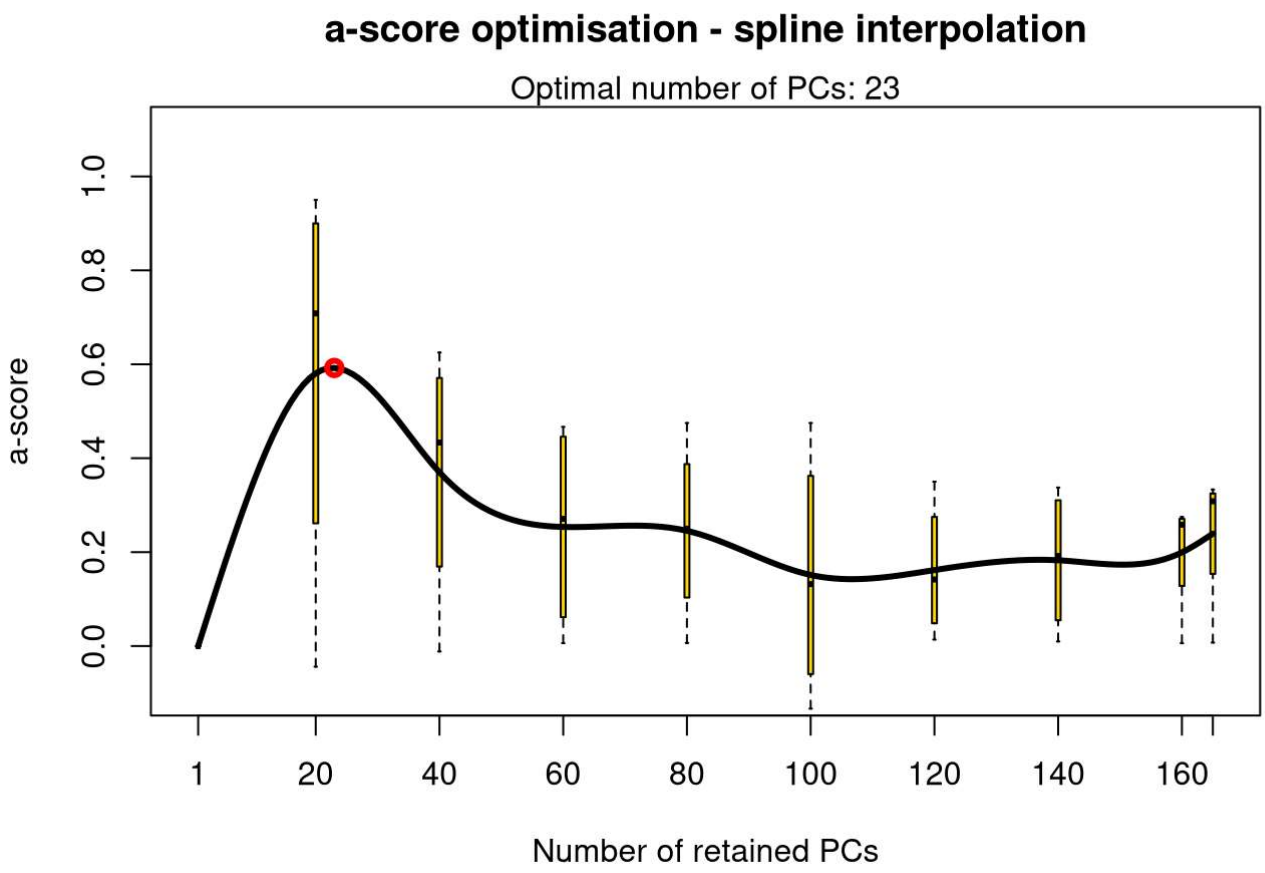
```
## # A tibble: 1 × 1
##     `1`
##   <int>
## 1    85
```

```
  # summarise(genotyping_accuracy = `1` / (`1` + `0`))

# save the duplicated individuals to discard
acc_check %>%
  filter(n!= 1) %>%
  write_csv('/workdir/smallmouth/hudson/redbreast/results/duplicated_individuals.csv')
```

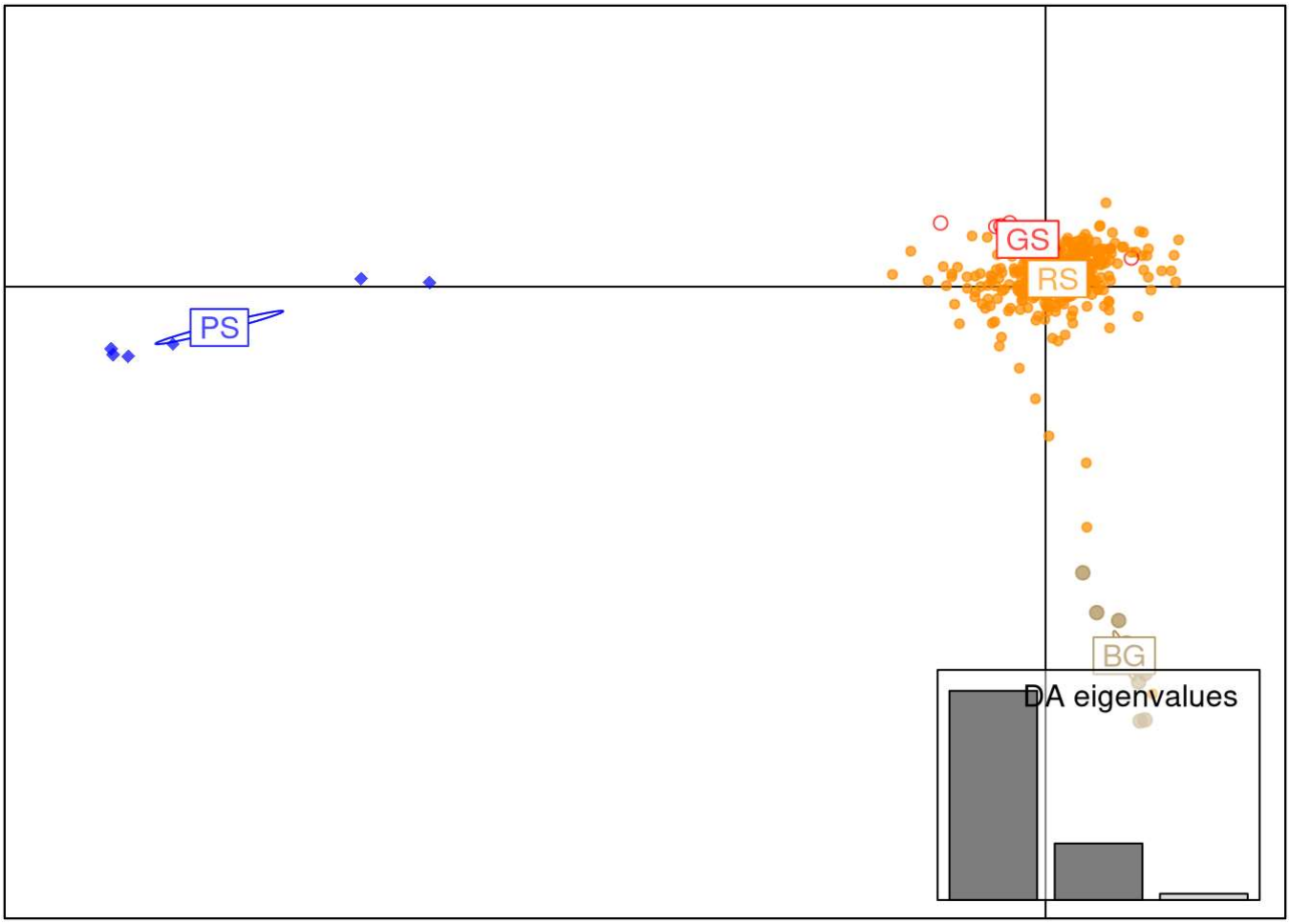
identify and remove hybrids - DAPC polarizing with known spp

tutorial: http://adegenet.r-forge.r-project.org/files/tutorial-dapc.pdf (http://adegenet.r-forge.r-project.org/files/tutorial-dapc.pdf) First round

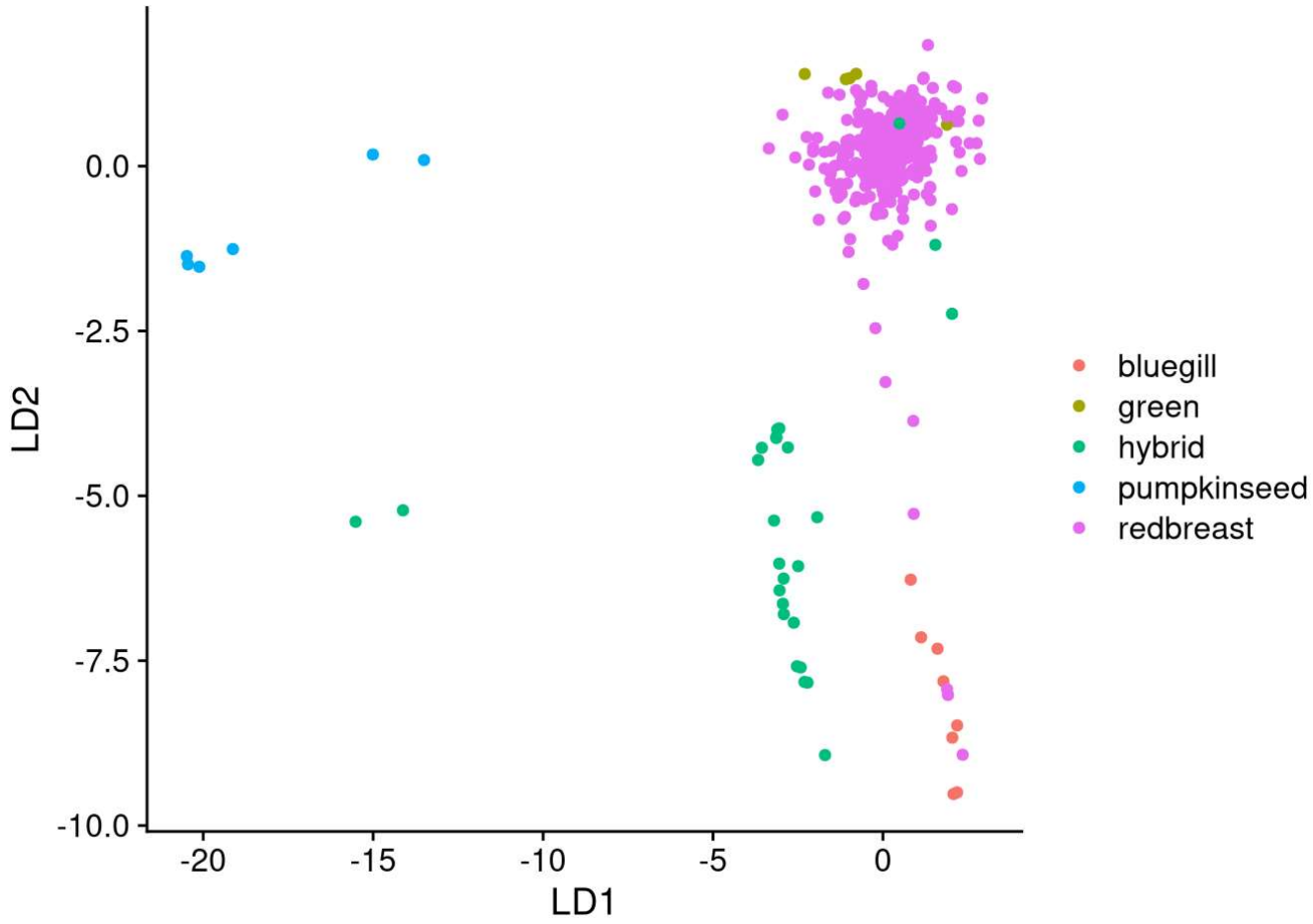
```
# upload the most recent version of the lab notebook, and import sample info here
        transect_info_hyb<-lab_notebook_samples %>%
          mutate(Species = if_else(is.na(Species), 'RS',
                           if_else(Species=='0', 'RS', Species)),
                 Field_code = LabCode) %>%
          dplyr::select('ind' = 'LabCode', 'pop' = 'Species')

# read in genotype data and import populations as well
        hyb<-pre_genos %>%
          left_join(transect_info_hyb, by='ind') %>%
          filter(!duplicated(ind))

# create new genind, split into known and hybrids
        objH <- df2genind(dplyr::select(hyb, -c(ind,pop)), sep = '/', ploidy = 2, ncode = 2, pop
= as.character(hyb$pop), ind.names = as.character(hyb$ind))
        objH_model<-objH[filter(hyb, !str_detect(pop,'/') & !str_detect(pop, 'hyb'))$ind]
        objH_hyb<-objH[filter(hyb, str_detect(pop,'/') | str_detect(pop, 'hyb'))$ind]

# Run DAPc, polarizing using known species
    dapc.objH.optimal<-optim.a.score(dapc(objH_model, n.pca = 200, n.da = nPop(objH_model) - 1))
```




```
    dapc.objH.model<-dapc(objH_model, var.contrib = T, n.pca = dapc.objH.optimal$best, n.da = nP
op(objH_model) - 1)
    scatter(dapc.objH.model, cellipse = .5, pch = 18:23, cex=1,cstar = 0,scree.da = T, clabel =
1, label.inds = T) # visualize
```



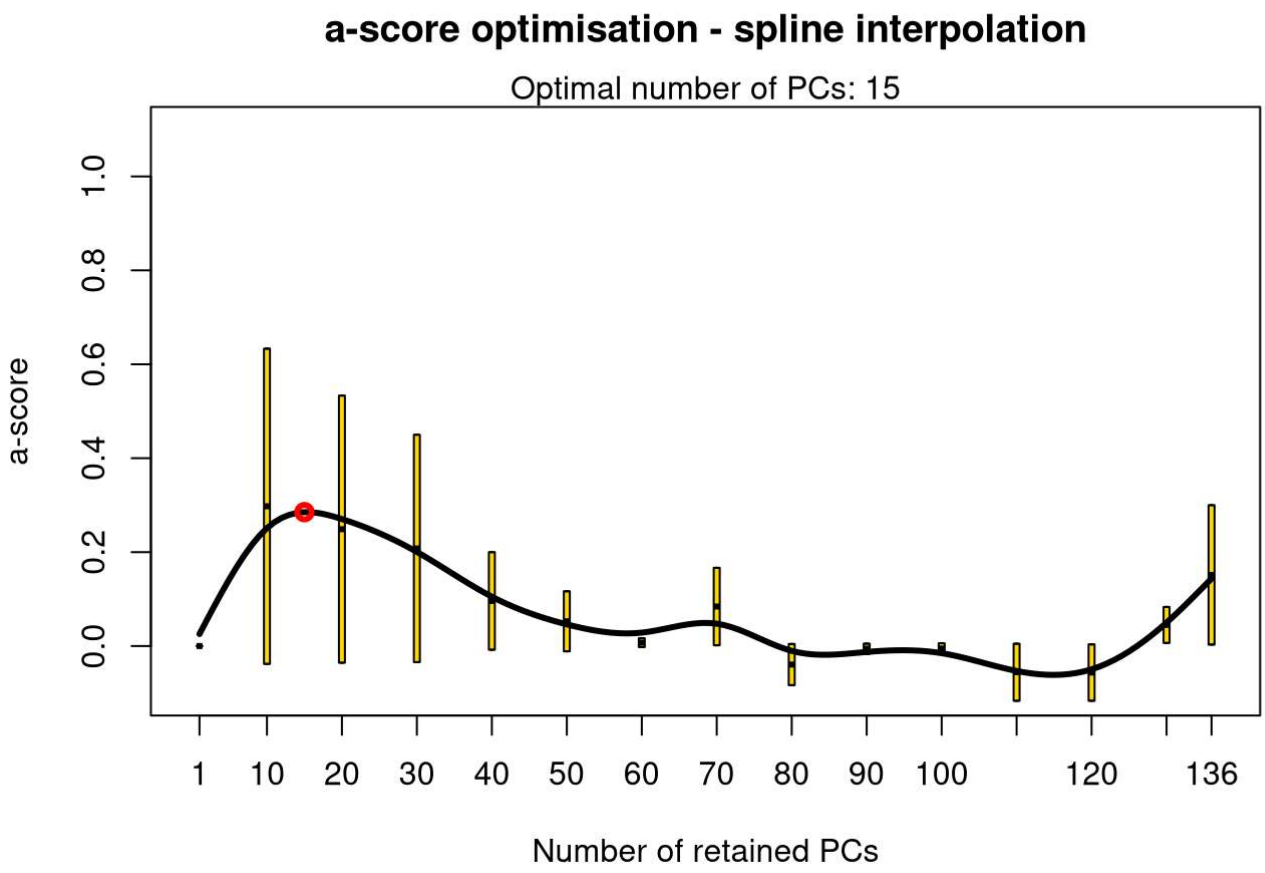

```
# Make a species joiner
    spp_code<-tibble(pop3=c('PS','BG','RS','GS','hyb'),
                     spp_code=c('pumpkinseed','bluegill','redbreast','green','hybrid'))

# with the polarized DAPC, bring in the new species
    objH_dapc1<-
      bind_rows(as.data.frame(dapc.objH.model$ind.coord), # species the model was polarized on
                as.data.frame(predict.dapc(dapc.objH.model, # bring in the hybrids
                                           newdata=objH_hyb)$ind.scores)) %>%
      rownames_to_column(var = 'ind') %>%
      left_join(hyb, by = 'ind') %>%
      left_join(dplyr::select(phenos, ind, river), by = 'ind') %>%
      mutate(pop2 = paste0(pop, "-", river),
             pop2 = if_else(str_detect(pop2, '/'), paste0('hyb',"-",river), pop2),
             pop3 = if_else(str_detect(pop, '/'), 'hyb', pop)) %>%
      left_join(spp_code, by = 'pop3')

    objH_dapc1 %>%
      ggplot(aes(LD1, LD2, color = spp_code)) +
      geom_point() +
      labs(color = '') +
      theme_cowplot()
```



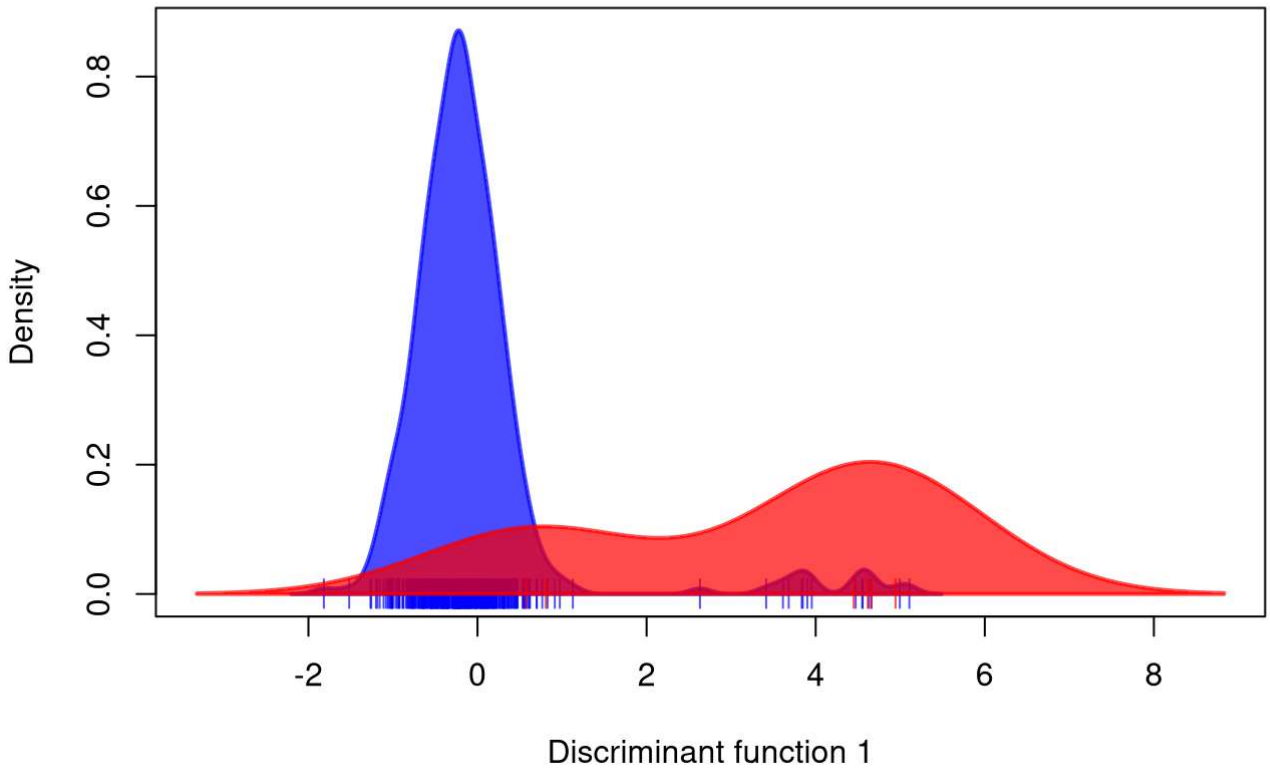

```
    ggsave('/workdir/smallmouth/hudson/redbreast/results/Fig_s1.svg', height = 5, width = 7)

    objH_dapc1<-objH_dapc1 %>%
      mutate(pop4 = if_else(LD1 < -17, "pumkinseed",
                    if_else(LD2 < -6 & LD1 > 0, 'bluegill',
                    if_else(pop3 == 'GS', 'green',
                    if_else(LD2 > -2 & LD1 > -5, 'redbreast','hybrid')))))
```

Second round - can we pull out GS?

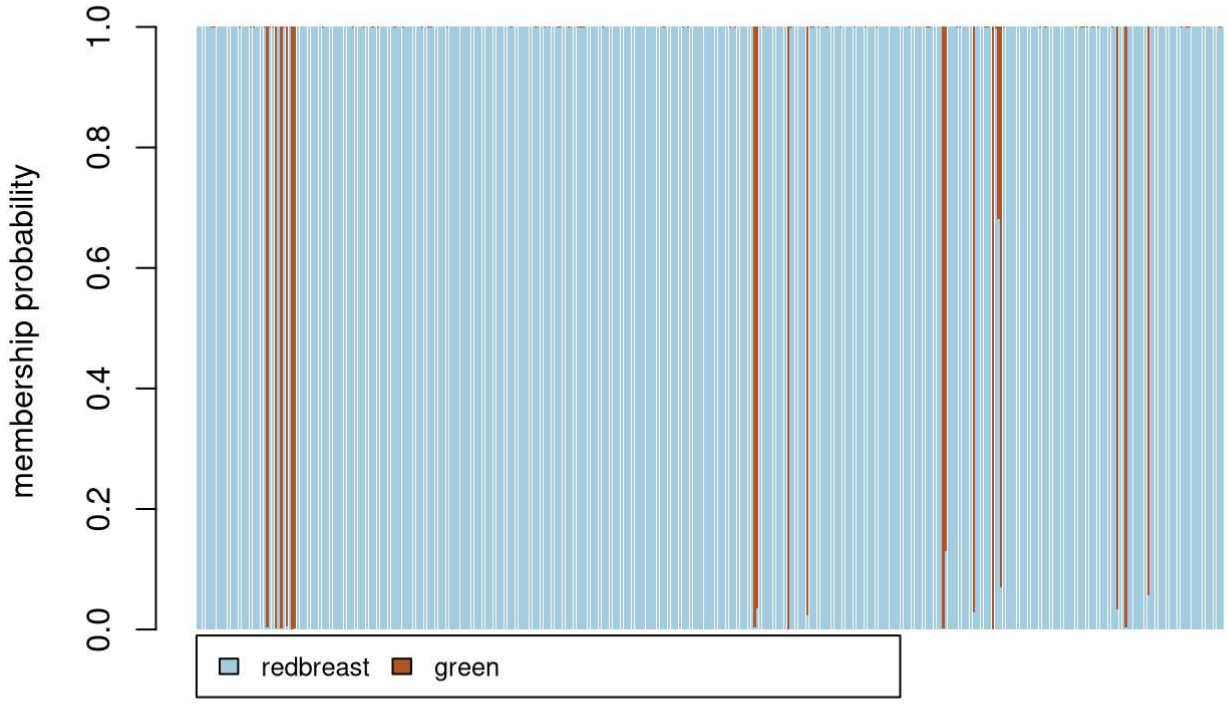
```
# Build each of the tibble going in  - here we are just trying to resolve green and red
    objH_dapc1_in<-objH_dapc1 %>%
      dplyr::select(-c(contains('LD'), pop, river, pop2, pop3, spp_code)) %>%
      filter(str_detect(pop4, 'redbreast|green'))

       objH2_model <- df2genind(dplyr::select(objH_dapc1_in, -c(ind,pop4)), sep = '/', ploidy =
2, ncode = 2, pop = as.character(objH_dapc1_in$pop4), ind.names = as.character(objH_dapc1_in$in
d))

# build DAPC, which should be better at resolving introgression
    dapc.objH2.optimal<-optim.a.score(dapc(objH2_model, n.pca = 200, n.da = nPop(objH2_model) -
1))
```



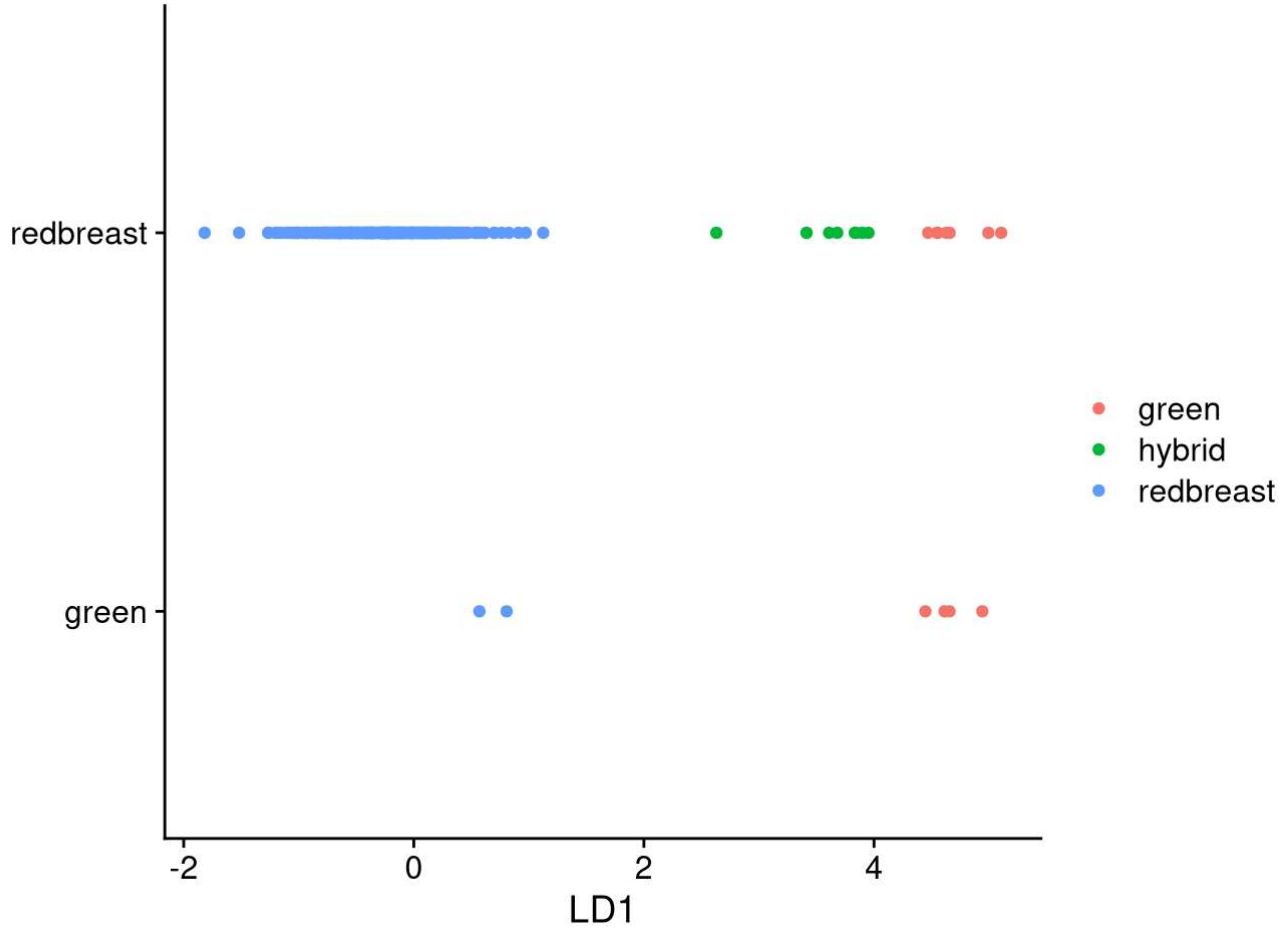

```
    dapc.objH2.model<-dapc(objH2_model, var.contrib = T, n.pca = dapc.objH2.optimal$best, n.da =
nPop(objH2_model) - 1)
    scatter(dapc.objH2.model, cellipse = .5, pch = 18:23, cex=1,cstar = 0,scree.da = T, clabel =
1, label.inds = T) # visualize
```




```
dd<-compoplot(dapc.objH2.model)
```




```
objh_dapc_2<- as.data.frame(dapc.objH2.model$ind.coord) %>%
  rownames_to_column(var = 'ind') %>%
  left_join(objH_dapc1_in, by = 'ind') %>%
  mutate(pop5 = if_else(LD1 > 4, 'green',
                        if_else(LD1<2, 'redbreast','hybrid')))

objh_dapc_2 %>%
  ggplot(aes(LD1, pop4, color = pop5)) +
  geom_point() +
  theme_cowplot() +
  labs(y='',color='')
```




```
    ggsave('/workdir/smallmouth/hudson/redbreast/results/Fig_s2.svg', height = 4, width = 7)

    objh_dapc_2 %>%
    bind_rows(anti_join(objH_dapc1, objH_dapc1_in, by = 'ind')) %>% # join to the spp designatio
ns we set in dapc1
      mutate(pop6 = coalesce(pop5, pop4)) %>%
      dplyr::select(ind, pop_final=pop6) %>%
      write_csv('/workdir/smallmouth/hudson/redbreast/results/hybrid_results.csv')
```

Aggegate data for hybrids

```
hyb_final_data<-genind2df(objH) %>%  # do this because its excluded blank loci
  rownames_to_column(var = 'ind') %>%
  left_join(read_csv('/workdir/smallmouth/hudson/redbreast/results/hybrid_results.csv'), by = 'i
nd') # bind to results

# How many of each species did we end with?
hyb_final_data %>%
  group_by(pop_final) %>%
  tally()
```

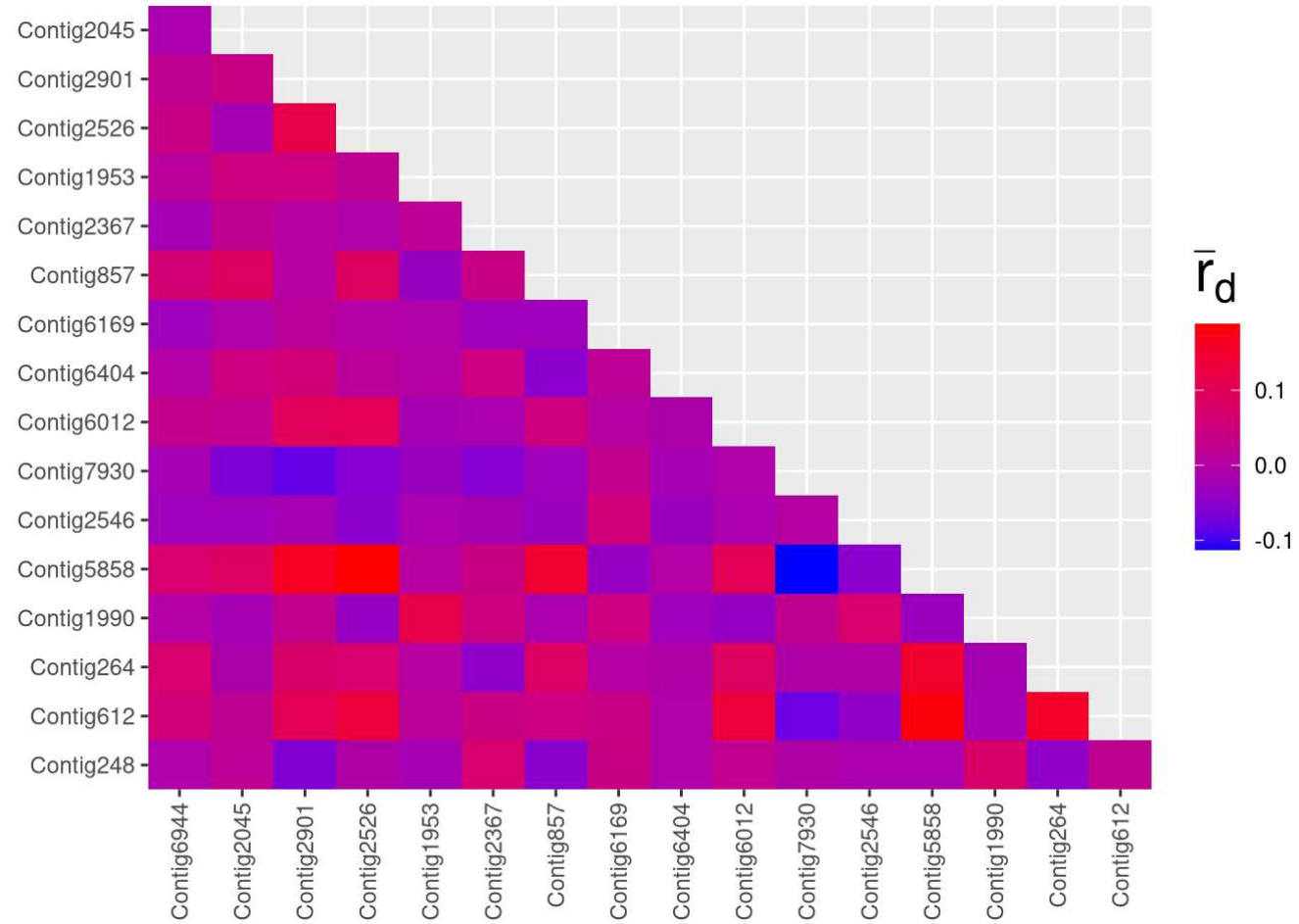
```
## # A tibble: 5 × 2
##   pop_final      n
##   <chr>      <int>
## 1 bluegill      11
## 2 green         11
## 3 hybrid        38
## 4 pumkinseed     4
## 5 redbreast    352
```

```
# How well did the panel work for other species?
    hyb_final_data %>%
      dplyr::select(contains('Contig'), pop_final, ind) %>%
      pivot_longer(-c(pop_final, ind)) %>%
      group_by(ind, pop_final) %>%
      summarise(proportion_amplified = sum(!is.na(value))/n()) %>%
      ungroup() %>%
      group_by(pop_final) %>%
      summarise(mean_amplified = round(mean(proportion_amplified),2),
                sd_amplified = round(sd(proportion_amplified),2))
```

```
## # A tibble: 5 × 3
##   pop_final  mean_amplified sd_amplified
##   <chr>               <dbl>        <dbl>
## 1 bluegill             0.34         0.06
## 2 green                0.32         0.05
## 3 hybrid               0.35         0.13
## 4 pumkinseed           0.51         0.02
## 5 redbreast            0.78         0.13
```

Linkage disequilibrium, HWE

```
# Check for any loci out of LD
    tt<-pair.ia(obj)
```



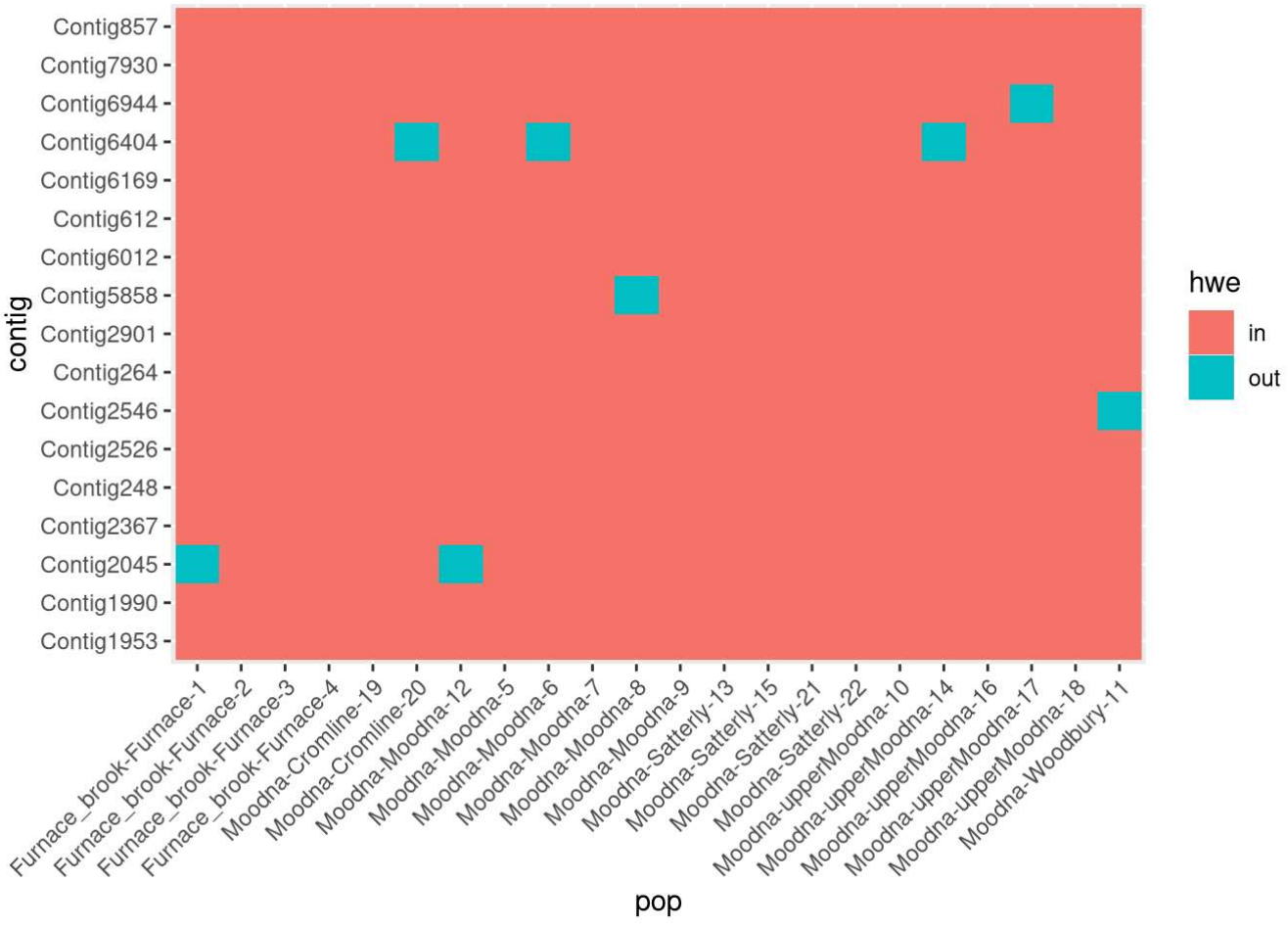

```
    paste0('max observed LD ',max(tt[,2])) # maximum observed LD
```

```
## [1] "max observed LD 0.188262508599301"
```

```
    paste0('mean observed LD ',mean(tt[,2])) # average observed LD
```

```
## [1] "mean observed LD 0.017343949065883"
```

```
# exclude loci out of HWE
    hwe_alpha<-0.01
    obj_hwe<-seppop(obj) %>% lapply(pegas::hw.test, B = 0)
    obj_hwe_pval<-sapply(obj_hwe, "[", i = TRUE, j = 3)
    obj_hwe_pval[obj_hwe_pval > hwe_alpha] <- 1
    hwe_plot<-obj_hwe_pval %>%
      as_tibble() %>%
      mutate(contig = row.names(obj_hwe_pval)) %>%
      pivot_longer(-contig, names_to = 'pop', values_to = 'hwe') %>%
      mutate(hwe = if_else(hwe<0.05, 'out', 'in'))

# Visualize
    hwe_plot %>%
      ggplot(aes(pop, contig, fill = hwe)) +
      geom_tile() +
      theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



```
# any loci out of HWE in greater than 50% of populations?
    hwe_summary<-hwe_plot %>%
      group_by(contig, hwe) %>%
      tally() %>%
      filter(!is.na(hwe)) %>%
      pivot_wider(names_from=hwe, values_from=n) %>%
      mutate(out = if_else(is.na(out), as.integer(0), out),
             percent_out_of_hwe = out / (`in`+out))

    hwe_summary
```

```
## # A tibble: 17 × 4
## # Groups:   contig [17]
##    contig      `in`   out percent_out_of_hwe
##    <chr>      <int> <int>              <dbl>
##  1 Contig1953    22     0             0
##  2 Contig1990    22     0             0
##  3 Contig2045    20     2             0.0909
##  4 Contig2367    22     0             0
##  5 Contig248     22     0             0
##  6 Contig2526    22     0             0
##  7 Contig2546    21     1             0.0455
##  8 Contig264     22     0             0
##  9 Contig2901    22     0             0
## 10 Contig5858    21     1             0.0455
## 11 Contig6012    22     0             0
## 12 Contig612     22     0             0
## 13 Contig6169    22     0             0
## 14 Contig6404    19     3             0.136
## 15 Contig6944    21     1             0.0455
## 16 Contig7930    22     0             0
## 17 Contig857     22     0             0
```

```
    paste0( 'mean observed HWE ', mean(hwe_summary$percent_out_of_hwe))
```

```
## [1] "mean observed HWE 0.0213903743315508"
```

```
   paste0 ('max observed HWE ', max(hwe_summary$percent_out_of_hwe))
```

```
## [1] "max observed HWE 0.136363636363636"
```

Test for null alleles

Tutorial: https://bookdown.org/hhwagner1/LandGenCourse_book/WE_3.html (https://bookdown.org/hhwagner1/LandGenCourse_book/WE_3.html)

all pops

Run this R script on unix, it takes a long time

```
nohup Rscript --vanilla /workdir/smallmouth/hudson/redbreast/publication_files/run_popgenreport_
nulls.R /workdir/smallmouth/hudson/redbreast/results/genos_clean.csv /workdir/smallmouth/hudson/
redbreast/results/popgenreport_nulls.csv > /workdir/smallmouth/nohups/run_popgenreport_hudson.no
hup &
```

Check the inputs - don't worry about anything less than, say 20%. Exclude anything above 50%. 20-50%, check pop-specific

```
  read_csv('/workdir/smallmouth/hudson/redbreast/results/popgenreport_nulls.csv') %>%
  filter(`2.5th percentile`>0)
```

```
## # A tibble: 0 × 6
## # ℹ 6 variables: name <chr>, Observed frequency <dbl>, Median frequency <dbl>,
## #   2.5th percentile <dbl>, 97.5th percentile <dbl>, test <chr>
```

pop-specific

Run the pop-specific popgenreport

```
nohup Rscript /workdir/smallmouth/hudson/redbreast/publication_files/run_popgenreport_nulls_bypo
p.R  > /workdir/smallmouth/nohups/run_popgenreport_nulls_bypop.nohup &
```

Read out pop-specific results - lots of pops show very high rates, not just one, so let's kick this locus out

```
read_csv('/workdir/smallmouth/hudson/redbreast/results/null_alleles_summ12_popSpecific.csv') %>%
  pivot_wider(names_from='quantile') %>%
  separate(pop, into = c('river','basin','transect'), sep = '-') %>%
  ggplot(aes(as.numeric(transect), `Observed frequency`)) +
  geom_point() +
  geom_errorbar(aes(ymin=`2.5th percentile`, ymax=`97.5th percentile`)) +
  facet_wrap(~summary, nrow=1)
```

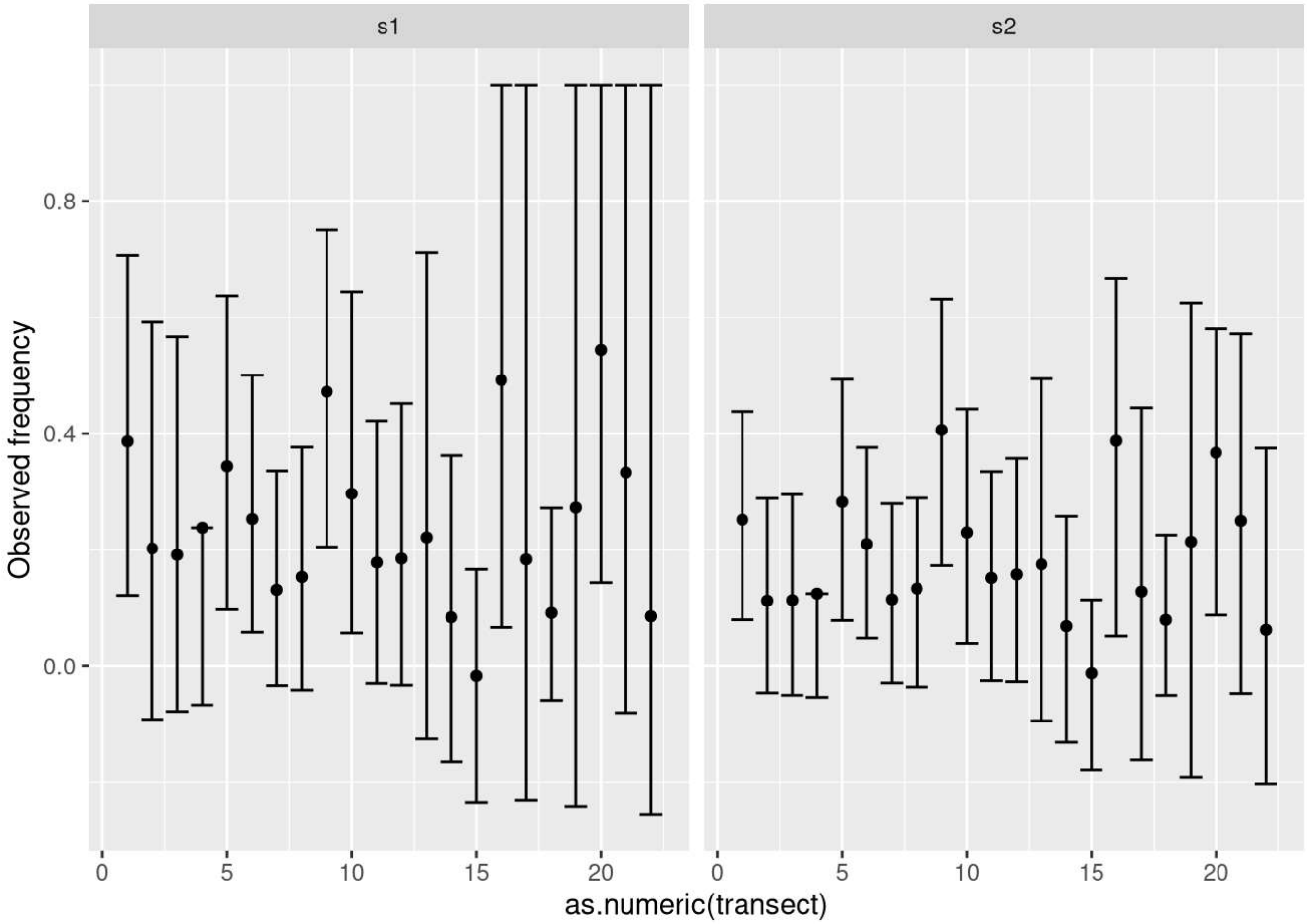


Table 1: Population-specific metrics

% related with sibship/parent/offspring pairs

some useful packages

```
long2freqs <- function(L) {
  L %>%
    group_by(Locus, Allele) %>%
    tally() %>%
    filter(!is.na(Allele)) %>%
    mutate(Freq = n / sum(n)) %>%
    arrange(Locus, desc(Freq)) %>%
    dplyr::select(-n) %>%
    group_by(Locus) %>%
    filter(n() > 1) %>% # this removes monomorphic loci
    mutate(AlleIdx = 1:n()) %>%
    ungroup() %>%
    mutate(LocIdx = as.integer(factor(Locus, levels = unique(Locus))),
           Chrom = "Unk",  # Ian probably knows where these are, but I don't have that info at t
he moment.
           Pos = LocIdx
           ) %>%
    dplyr::select(Chrom, Pos, Locus, Allele, Freq, LocIdx, AlleIdx) %>%
    reindex_markers()  # this is a CKMRsim function
}
      ngs_msat_create_ckmr <- function(Fr) {
        create_ckmr(D = Fr,
                    kappa_matrix = kappas[c("PO", "FS", "HS", "U"), ],
                    ge_mod_assumed = ge_model_microsat1,
                    ge_mod_true = ge_model_microsat1,
                    ge_mod_assumed_pars_list = list(miscall_rate = 0.003, dropout_rate = 0.004),
                    ge_mod_true_pars_list = list(miscall_rate = 0.003, dropout_rate = 0.004)
                   )
      }
      simQs <- function(CK) {
        simulate_Qij(C = CK, reps = 2e04,
                    sim_relats = c("PO", "FS", "HS", "U"),
                    calc_relats = c("PO", "FS", "U"))
      }

           mc_samp <- function(Q) {
        mc_sample_simple(
          Q,
          nu = c("PO", "FS"),
          FNRs = c(seq(0.001, 0.20, by = 0.001))
        )
            }
```

find fs/po with CKMRsim doing this for separate genetic clusters, at least the obvious ones

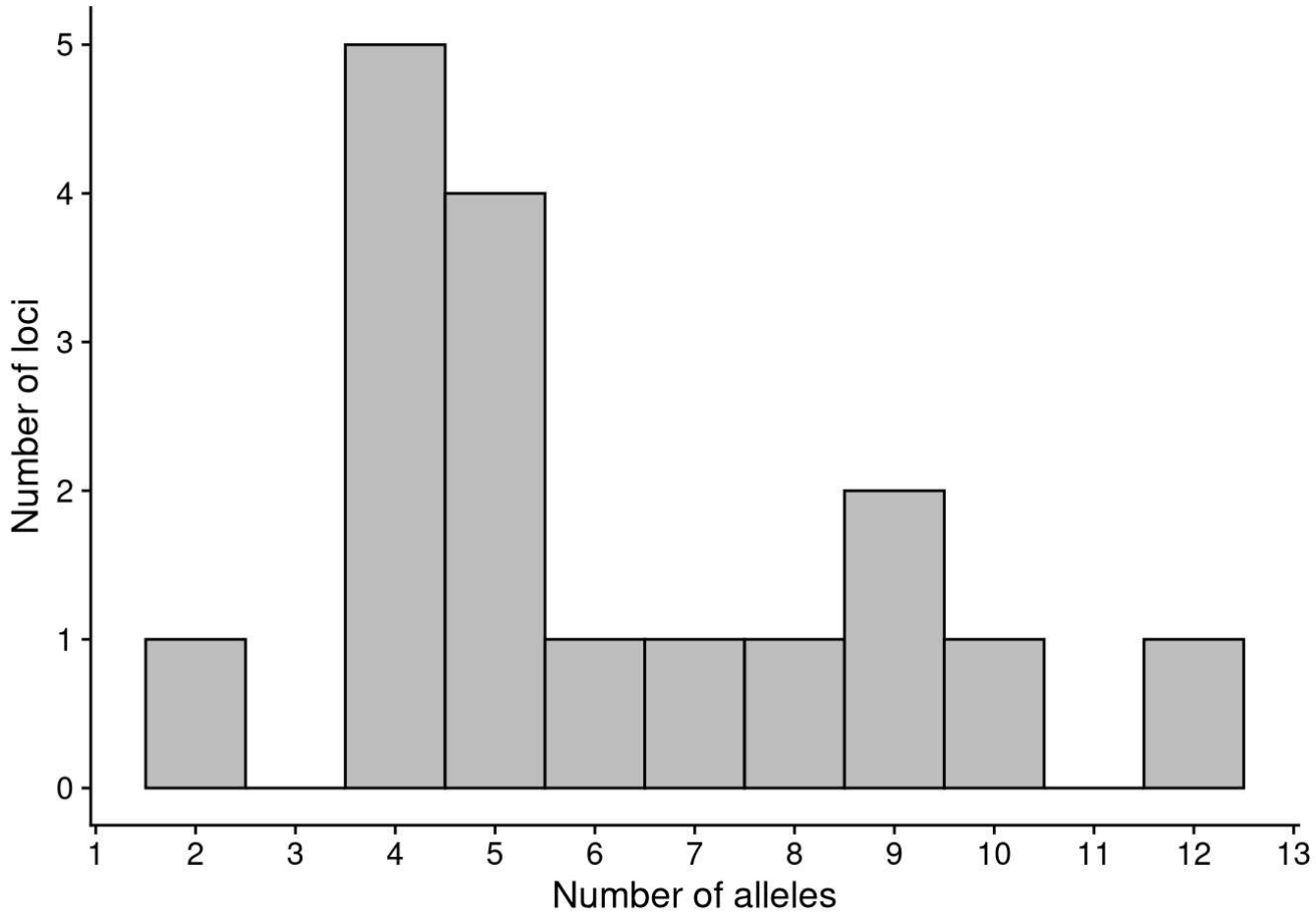
```
library(CKMRsim)
logl_cutoff <- 0.1
pop_comparisons<-relatedness_groups # furnace, moodna, and pop 18

po_fs_out<-tibble()
for(pop_in in unique(pop_comparisons$pop)){ # unique(pop_comparisons$pop) c('Furnace','Moodna')

    # Prep the genos going in into a long format
          # need to reorder so that individual is first and we have the separate alleles coded a
s gene_copy
          ckmr_input<-
            rownames_to_column(genind2df(obj, sep = '_'),var = 'Indiv') %>%
            separate(pop, into = c('River','Basin','Site'), sep = '-') %>%
            filter(
              Site %in% filter(pop_comparisons, pop == pop_in)$paper_pop,
                   Indiv %nin% read_csv('/workdir/smallmouth/hudson/redbreast/results/matchers.c
sv')$indiv_1) %>% # reviewer note - create an empty file for now called matchers.csv and point t
o it
            dplyr::select(-c(River, Basin, Site)) %>%
            pivot_longer(-Indiv, names_to = 'Locus', values_to = 'allele_diploid') %>%
            separate(allele_diploid, into = c('1','2'), sep = '_') %>%
            pivot_longer(-c(Indiv, Locus), names_to = 'gene_copy', values_to = 'Allele') %>%
            filter(!is.na(Allele), Allele != '') # this step is crucial - got to remove both NA
and ''.

    # estimate frequences for each allele in the locus
          ckmr_freqs <- long2freqs(ckmr_input)

    # make the ckmr obj_noFillect
          Lp <- semi_join(ckmr_input, ckmr_freqs, by = "Locus")
          RS_in<-list(Genos = Lp, Freqs = ckmr_freqs)
          CK_out<-create_ckmr(D = RS_in$Freqs,
                              kappa_matrix = kappas[c("PO", "FS", "HS", "U"), ],
                              ge_mod_assumed = ge_model_microsat1,
                              ge_mod_true = ge_model_microsat1,
                              ge_mod_assumed_pars_list = list(miscall_rate = 0.003, dropout_rate = 0.0
04),
                              ge_mod_true_pars_list = list(miscall_rate = 0.003, dropout_rate = 0.00
4))

    # find individuals which might be actually the same, with all but 2 loci mismatched
          matchers<-find_close_matching_genotypes(
            LG = RS_in$Genos,
            CK = CK_out,
            max_mismatch = 2)

          read_csv('/workdir/smallmouth/hudson/redbreast/results/matchers.csv') %>%
            bind_rows(matchers) %>%
            write_csv('/workdir/smallmouth/hudson/redbreast/results/matchers.csv')

    # assess our power for POP and FS, using the conservative cutoff Eric suggests
          Qs <- simQs(CK_out)
```

```
          logls <- bind_rows(
            extract_logls(Qs, numer = c(PO = 1), denom = c(U = 1)),
            extract_logls(Qs, numer = c(FS = 1), denom = c(U = 1)),
            extract_logls(Qs, numer = c(PO = 1), denom = c(FS = 1))
          )
          logls %>%
            filter(denom_wts == "U=1") %>%
            ggplot(., aes(x = logl_ratio, fill = true_relat)) +
              geom_density(alpha = 0.35) +
              facet_wrap(~ numer_wts, ncol = 1)

    # get pairwise relationships for FSP
      # get logL ratio needed for low FPR
          FS_candidates <- pairwise_kin_logl_ratios(D1 = ckmr_input, D2 = ckmr_input, CK = CK_ou
t, numer = "FS", denom = "U", keep_top = 10)

          FPRs <- mc_sample_simple(Qs, nu = "FS",de = "U",lambda_stars = seq(1,20, 0.1))
          cutoff <-logl_cutoff * (nrow(FS_candidates) ^ -1)
          FSP_logL<-filter(FPRs, FPR < cutoff) %>%
            slice_head(n=1)

          FS_confirmed <- filter(FS_candidates, logl_ratio > FSP_logL$Lambda_star) %>% mutate(re
lationship = 'FSP')

    # get pairwise POP relationship
        # size at age 2, 75mm+ (https://link.springer.com/article/10.1007/s11252-017-0696-8)
        # mature at age 2, age 2 is 80mm + (https://scholarship.richmond.edu/cgi/viewcontent.cg
i?article=1990&context=masters-theses)
          ckmr_input_juvs <- filter(ckmr_input, Indiv %in% filter(phenos, Length <= 80)$ind)
          ckmr_input_adult <- filter(ckmr_input, Indiv %in% filter(phenos, Length > 80)$ind)

        try({
          PO_candidates <- pairwise_kin_logl_ratios(D1 = ckmr_input_adult, D2 = ckmr_input_juvs,
CK = CK_out, numer = "PO", denom = "U", keep_top = 10)

          cutoff <-logl_cutoff * (nrow(PO_candidates) ^ -1)
          FPRs <- mc_sample_simple(Qs, nu = "PO",de = "U",lambda_stars = seq(1,20, 0.1))
          PO_lambda<-filter(FPRs, FPR < cutoff) %>%
            slice_head(n=1)

          PO_confirmed <- filter(PO_candidates, logl_ratio > PO_lambda$Lambda_star) %>%
            mutate(relationship = 'POP')
        })


    # export, removing duplicates, then make list of family groups
          po_fs<-bind_rows(FS_confirmed, PO_confirmed) %>%
            rowwise()  %>%
            mutate(d12 = paste0(sort(c(D2_indiv, D1_indiv)), collapse = '-')) %>%
            ungroup() %>%
            filter(!duplicated(d12)) %>%
            dplyr::select(D1=D1_indiv, D2=D2_indiv)
```

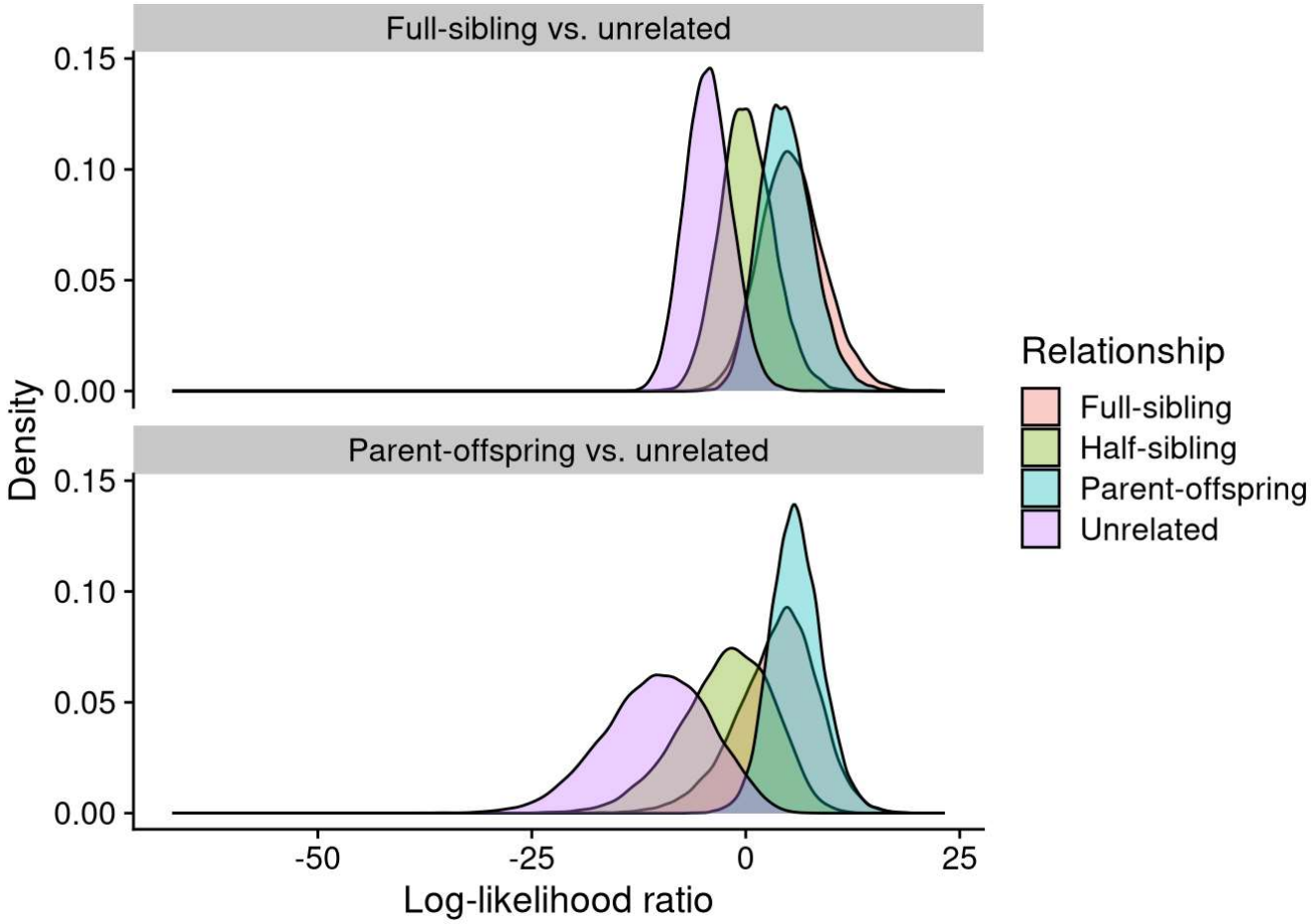
```
          po_fs_out<-bind_rows(po_fs, po_fs_out)
}

      g3<-simplify(graph.data.frame(po_fs_out[order(po_fs_out[[1]]),], directed = F))

      single_fams<-tibble(ind = indNames(obj)) %>%
        filter(ind %nin% po_fs_out$D1, ind %nin% po_fs_out$D2)

      components(g3)$membership %>%
        as_tibble(rownames = 'ind') %>%
        bind_rows(single_fams)  %>%
        arrange(value) %>%
        mutate(fam_index = if_else(is.na(value), as.double(row_number()), value), # add sequenti
al numbers
               fam_index = as.double(as.factor(fam_index))) %>%
        dplyr::select(-value) %>%
        write_csv('/workdir/smallmouth/hudson/redbreast/ckmrsim/redbreast_relatedness.csv')

      write_csv(po_fs_out, '/workdir/smallmouth/hudson/redbreast/ckmrsim/redbreast_pairwise_rela
tedness.csv')
```

Export figures of power - run this for all populations

```
    # Prep the genos going in into a long format
          # need to reorder so that individual is first and we have the separate alleles coded a
s gene_copy
          ckmr_input<-
            rownames_to_column(genind2df(obj, sep = '_'),var = 'Indiv') %>%
            separate(pop, into = c('River','Basin','Site'), sep = '-') %>%
            filter(Indiv %nin% read_csv('/workdir/smallmouth/hudson/redbreast/results/matchers.c
sv')$indiv_1) %>%
            dplyr::select(-c(River, Basin, Site)) %>%
            pivot_longer(-Indiv, names_to = 'Locus', values_to = 'allele_diploid') %>%
            separate(allele_diploid, into = c('1','2'), sep = '_') %>%
            pivot_longer(-c(Indiv, Locus), names_to = 'gene_copy', values_to = 'Allele') %>%
            filter(!is.na(Allele), Allele != '') # this step is crucial - got to remove both NA
and ''.

    # estimate frequences for each allele in the locus
          ckmr_freqs <- long2freqs(ckmr_input)

    # make the ckmr obj_noFillect
          Lp <- semi_join(ckmr_input, ckmr_freqs, by = "Locus")
          RS_in<-list(Genos = Lp, Freqs = ckmr_freqs)
          CK_out<-create_ckmr(D = RS_in$Freqs,
                              kappa_matrix = kappas[c("PO", "FS", "HS", "U"), ],
                              ge_mod_assumed = ge_model_microsat1,
                              ge_mod_true = ge_model_microsat1,
                              ge_mod_assumed_pars_list = list(miscall_rate = 0.003, dropout_rate = 0.0
04),
                              ge_mod_true_pars_list = list(miscall_rate = 0.003, dropout_rate = 0.00
4))

    # Plot of alleles per locus
          ckmr_freqs %>%
            group_by(Locus) %>%
            tally() %>%
            ggplot(aes(n)) +
            geom_histogram(binwidth = 1, color = 'black', fill = 'grey') +
            theme_cowplot() +
            scale_x_continuous(breaks = 0:15) +
            labs(x = 'Number of alleles', y = 'Number of loci')
```




```
          ggsave('/workdir/smallmouth/hudson/redbreast/results/ckmr_alleles.png', height = 3, wi
dth = 5)

    # assess our power for POP and FS, using the conservative cutoff Eric suggests
          Qs <- simQs(CK_out)
          logls <- bind_rows(
            extract_logls(Qs, numer = c(PO = 1), denom = c(U = 1)),
            extract_logls(Qs, numer = c(FS = 1), denom = c(U = 1)),
            extract_logls(Qs, numer = c(PO = 1), denom = c(FS = 1))
          )

          lookup<-tibble(true_relat = c('FS','PO','HS','U'),
                         long_relat = c('Full-sibling','Parent-offspring','Half-sibling','Unrela
ted'))

          logls %>%
            filter(denom_wts == "U=1") %>%
            mutate(comp = if_else(numer_wts == 'FS=1', 'Full-sibling vs. unrelated', 'Parent-off
spring vs. unrelated')) %>%
            left_join(lookup) %>%
            ggplot(., aes(x = logl_ratio, fill = long_relat)) +
              geom_density(alpha = 0.35) +
              facet_wrap(~ comp, ncol = 1) +
            theme_cowplot() +
            labs(x = 'Log-likelihood ratio', y = 'Density', fill = 'Relationship')
```




```
        ggsave('/workdir/smallmouth/hudson/redbreast/results/ckmr_power.png', height = 5, widt
h = 6)
```

Ne (Vareff)

Install from github

```
cd /workdir/smallmouth/hudson/redbreast/VarEff/
git clone https://github.com/cran/VarEff.git
tar -czvf VarEff.tar.gz VarEff
# Go to packages > install > select VarEff

mkdir input_data
```

Need to manually recode our alleles as the actual number of repeats, instead of using the flanking regions like I am

```
# Make a list of all of the alleles we kept for each locus
    overall_tib<-genind2df(obj, sep = '-') %>%
      rownames_to_column(var='ind') %>%
      group_by(pop) %>%
      mutate(n=n()) %>%
      ungroup() %>%
      # filter(n>4) %>%  # exclude poulations with 4 or fewer individuals
      dplyr::select(-n) %>%
      pivot_longer(-c(ind,pop), names_to='locus') %>%
      separate(value, into = c('al1','al2'), sep = '-') %>%
      pivot_longer(-c(ind,pop,locus), names_to='copy', values_to='allele')

    kept_alleles<-overall_tib %>%
      mutate(locus_allele=paste0(locus, '#',allele)) %>%
      filter(!duplicated(locus_allele)) %>%
      dplyr::select(locus_allele) %>%
      separate(locus_allele, into = c('locus','allele'), sep = '#', remove = F) %>%
      arrange(locus, allele) %>%
      filter(allele != 'NA')

# Read in each unique haplotype and filter for the ones that we are using
    commonAlleles_filt<-commonAlleles[names(commonAlleles) %in% kept_alleles$locus_allele]

    DECIPHER::BrowseSeqs(commonAlleles) # if desired, take a look at the sequences

# Here, let's score the sequences based on
    allele_converter<-tibble(locus_allele=names(commonAlleles_filt), length = Biostrings::width
(commonAlleles_filt)) %>%
      separate(locus_allele, into = c('locus','allele'), sep = '#') %>%
      group_by(locus) %>%
      mutate(msat_repeat=1+(length-min(length))/4) %>% # take the minimum length as the baseline
(+1) then the number of 4-repeats on top of that
      ungroup() %>%
      filter(msat_repeat %% 1==0)  # we're going to have to exclude any that don't show 4-repeat
s

    nrow(kept_alleles)-nrow(allele_converter) # how many alleles are we losing? can look at them
if desired, two lines above
```

```
## [1] 2
```

```
# Convert genotype calls to repeats
    repeat_tib<-overall_tib %>%
      left_join(allele_converter, by = c('locus','allele')) %>%
      filter(!is.na(msat_repeat)) %>%
      write_csv('/workdir/smallmouth/hudson/redbreast/results/genos_msat_repeat.csv') %>%
      group_by(pop, locus, msat_repeat) %>%
      summarise(n_alleles_per_locus_per_pop=n()) %>%
      ungroup()

# for loop where we build the input file for each population separately this take ~2 minutes for
all 18 pops
    for(pop_in in unique(repeat_tib$pop)){
      pop_filtered<-filter(repeat_tib, pop==pop_in)

      for(loc_in in unique(pop_filtered$locus)){
        pop_loc_filtered<-filter(pop_filtered, locus==loc_in)

        n_alleles<-max(filter(repeat_tib, locus==loc_in)$msat_repeat) # for this locus, grab the
number of alleles

        cat(paste0(n_alleles), file= paste0("/workdir/smallmouth/hudson/redbreast/VarEff/input_d
ata/",pop_in,".txt"), append=TRUE,sep="\n") #write n_allele to file

        allele_counts<-tibble(msat_repeat=1:n_alleles) %>%
          left_join(dplyr::select(pop_loc_filtered, msat_repeat, n_alleles_per_locus_per_pop), b
y = 'msat_repeat') %>%
          mutate(n_alleles_per_locus_per_pop = replace_na(n_alleles_per_locus_per_pop, 0))

        allele_counts_string<-paste(unlist(allele_counts$n_alleles_per_locus_per_pop), collapse
='\t')

        cat(paste0(allele_counts_string), file= paste0("/workdir/smallmouth/hudson/redbreast/Var
Eff/input_data/",pop_in,".txt"), append=TRUE,sep="\n") #write n_allele to file
      }
    }

# Write these to file so that the R script running on unix can find them
input<- write_csv(tibble(pop=(popNames(obj)), nloc=nLoc(obj)), '/workdir/smallmouth/hudson/redbr
east/results/obj_popnames.csv') # write the popnames to file so can run it independently
```

I put the R code in the terminal, because it will take a while

```
nohup Rscript /workdir/smallmouth/hudson/redbreast/publication_files/run_vareff.R > /workdir/sma
llmouth/nohups/run_vareff.nohup &
```

Export the results

```
read_csv('/workdir/smallmouth/hudson/redbreast/results/vareff_output.csv') %>%
  separate(pop, into  =c('river','basin','transect'), sep = '-', remove = F) %>%
  left_join(phenos_subset, by = 'pop') %>%
  filter(n>4) %>%
  write_csv('/workdir/smallmouth/hudson/redbreast/results/vareff_output_pops_filtered.csv')
```

private alleles, rarefied

private allele rarefaction

```
df_in<-genind2df(obj) %>%
  rownames_to_column(var = 'ind')

pop_n<-df_in %>%
  group_by(pop) %>%
  tally()

boot_out<-tibble()
for(boot in 1:100){
    priv_genind_in<-df_in %>%
      group_by(pop) %>%
      slice_sample(n = (min(pop_n$n)), replace = F) %>%
      ungroup()

    boot_in<-df2genind(dplyr::select(priv_genind_in, -c(ind,pop)), sep = '/', ploidy = 2, ncode
= 2, pop = as.character(priv_genind_in$pop), ind.names = as.character(priv_genind_in$ind)) %>%
      private_alleles()  %>%
      as_tibble(rownames = 'pop') %>%
      pivot_longer(-pop, names_to = 'locus', values_to = 'priv') %>%
      group_by(pop) %>%
      summarise(priv = sum(priv)) %>%
      mutate(iter = boot)

    boot_out<-bind_rows(boot_out, boot_in)
}

boot_out %>%
  group_by(pop) %>%
  summarise(`Private alleles` = round(mean(priv),2)) %>%
  write_csv('/workdir/smallmouth/hudson/redbreast/results/private_allele_rarefied_per_basin.cs
v')
```

All the other ones (easy) then bind it all together

```
# Number of redbreast
    n_RS<-strata(obj) %>%
      mutate(pop = paste0(River, '-', Basin, '-', Transect)) %>%
      group_by(pop) %>%
      summarise(`Number of redbreast` = n()) %>%
      ungroup()

# Prop related per site
    related<-relats %>%
      left_join(phenos) %>%
      group_by(pop) %>%
      summarise(n_sampled = n(),
                n_fams = length(unique(fam_index)),
                Relatedness = 1-round(n_fams/n_sampled, 2)) %>%
      dplyr::select(pop, Relatedness)

# Number of hybrids per site
    n_hyb<-read_csv('/workdir/smallmouth/hudson/redbreast/results/hybrid_results.csv') %>%
      left_join(phenos_all_fish, by = 'ind') %>%
      filter(pop_final == 'hybrid') %>%
      group_by(old_pop = pop) %>%
      summarise(n_hybrid = n()) %>%
      left_join(site_joiner, by = 'old_pop') %>%
      dplyr::select(-old_pop)

    sum(n_hyb$n_hybrid) # How many hybrids detected total?
```

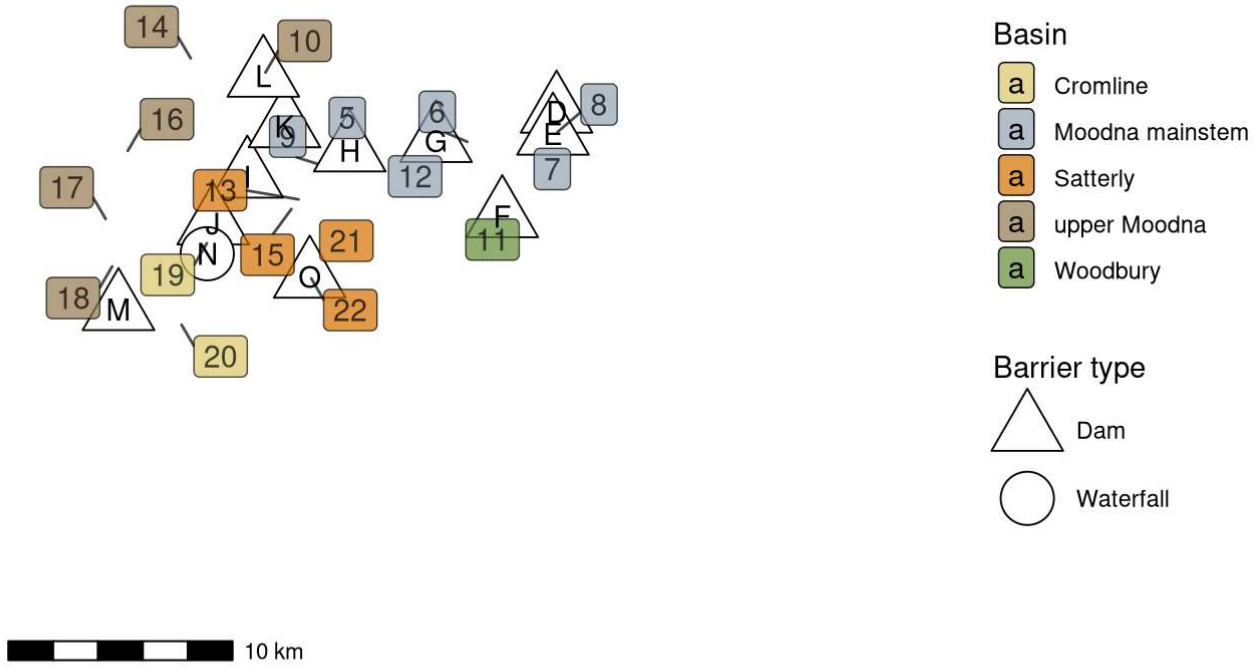
```
## [1] 38
```

```
# Allelic richness
    Ar_mean<-as_tibble(rownames_to_column(allelic.richness(genind2hierfstat(obj), diploid = T, m
in.n = min_n_per_pop*2)$Ar, var = 'locus')) %>%
      pivot_longer(-locus, names_to = 'pop', values_to = 'Ar') %>%
      write_csv('/workdir/smallmouth/hudson/redbreast/results/Ar_full.csv') %>% # export the ful
l AR table
      group_by(pop) %>%
      summarise(`Rarefied allelic richness` = round(mean(Ar),2))

# Inbreeding coefficient. Nan means its fixed at that allele, so not informative
    Fis_mean<-as_tibble(rownames_to_column(data.frame(basic.stats(obj, diploid = TRUE)$Fis), var
= 'locus')) %>%
      pivot_longer(-locus, names_to = 'pop', values_to = 'Fis') %>%
      mutate(pop = str_replace_all(pop, '\\.', '-')) %>%
      write_csv('/workdir/smallmouth/hudson/redbreast/results/Fis_full.csv') %>%
      group_by(pop) %>%
      summarise(`Inbreeding coefficient (Fis)` = round(mean(Fis, na.rm = T), 2))

# Observed heterozygosity
    Ho_mean<-as_tibble(rownames_to_column(data.frame(basic.stats(obj, diploid = TRUE)$Ho), var =
'locus')) %>%
      pivot_longer(-locus, names_to = 'pop', values_to = 'Ho') %>%
      mutate(pop = str_replace_all(pop, '\\.', '-')) %>%
      write_csv('/workdir/smallmouth/hudson/redbreast/results/Ho_full.csv') %>%
      group_by(pop) %>%
      summarise(`Observed heterozygosity (Ho)` = round(mean(Ho, na.rm = T), 2))

# Gene diversity (Expected heterozygosity)
    He_mean<-as_tibble(rownames_to_column(data.frame(basic.stats(obj, diploid = TRUE)$Hs), var =
'locus')) %>%
      pivot_longer(-locus, names_to = 'pop', values_to = 'He') %>%
      mutate(pop = str_replace_all(pop, '\\.', '-')) %>%
      write_csv('/workdir/smallmouth/hudson/redbreast/results/He_full.csv') %>%
      group_by(pop) %>%
      summarise(`Expected heterozygosity (He)` = round(mean(He, na.rm = T), 2))

# population-specific Fst (figures foloowing Kitada 2021 G3 Figure 2)
    fst_mean<-betas(obj)$betaiovl %>%
      as.data.frame() %>%
      rownames_to_column(var = 'pop') %>%
      dplyr::rename(`Fixation index (Fst)`='.') %>%
      mutate(`Fixation index (Fst)` = round(`Fixation index (Fst)`, 2))

# Ne
    ne_in<-read_csv('/workdir/smallmouth/hudson/redbreast/results/vareff_output_pops_filtered.cs
v') %>%
      filter(generations==0) %>%
      mutate(Ne = paste0(round(Ne_harmonic_mean, 0), ' (',round(Ne_5th_quantile, 0),'-',round(Ne
_95th_quantile, 0), ')')) %>%
      dplyr::select(pop, `Effective population size (Ne)`=Ne)

# VeRatio
```

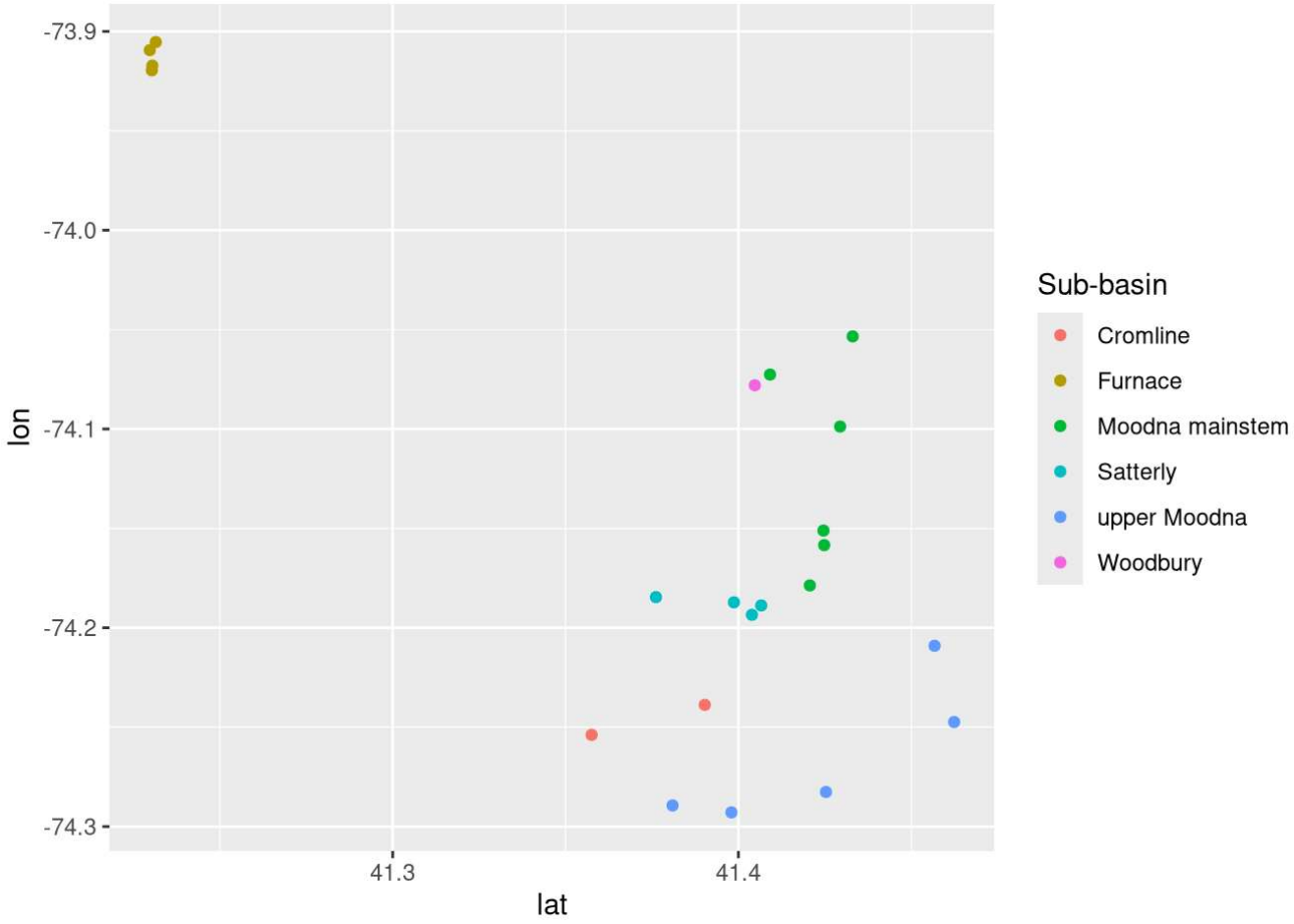
```
    ve_in<-read_csv('/workdir/smallmouth/hudson/redbreast/results/vareff_output_pops_filtered.csv') %>%
      filter(generations==0 | generations==100) %>%
      dplyr::select(pop, generations, Ne_harmonic_mean) %>%
      pivot_wider(names_from = generations, values_from=Ne_harmonic_mean) %>%
      mutate(VarEff=round(`100`/`0`,2)) %>%
      dplyr::select(c(pop, `Variance Ne`=VarEff))

# cpue * river order
  phenos_sec<-phenos %>%
    group_by(pop,river_order, altitude_m) %>%
    summarise(Sec_shocked = unique(Sec_shocked))

# Make final export table
    left_join(n_RS, n_hyb, by = 'pop') %>%
      mutate(`Number of hybrids` = replace_na(n_hybrid, 0)) %>%
      left_join(phenos_sec, by = 'pop') %>%
      mutate(`Redbreast catch/minute` = round((`Number of redbreast`/Sec_shocked)*60, 2)) %>%
      relocate(pop, `River order` = river_order, `Altitude (m)` = altitude_m, `Number of redbreast`, `Redbreast catch/minute`) %>%
      left_join(related, by = 'pop') %>%
      #left_join(Ar_mean, by = 'pop') %>%
      left_join(Fis_mean, by = 'pop') %>%
      #left_join(Ho_mean, by = 'pop') %>%
      #left_join(He_mean, by = 'pop') %>%
      #left_join(read_csv('/workdir/smallmouth/hudson/redbreast/results/private_allele_rarefied_per_basin.csv'), by = 'pop') %>%
      #left_join(fst_mean, by = 'pop') %>%
      left_join(ne_in, by = 'pop') %>%
      #left_join(ve_in, by = 'pop') %>%
      separate(pop, into = c('Basin','Sub-basin','Site'), sep = '-') %>%
      dplyr::select(-c(Sec_shocked, n_hybrid, Basin)) %>%
      arrange(as.integer(Site)) %>%
      write_csv('/workdir/smallmouth/hudson/redbreast/results/summary_stats_per_pop.csv')
```

Figure 1: Study site map

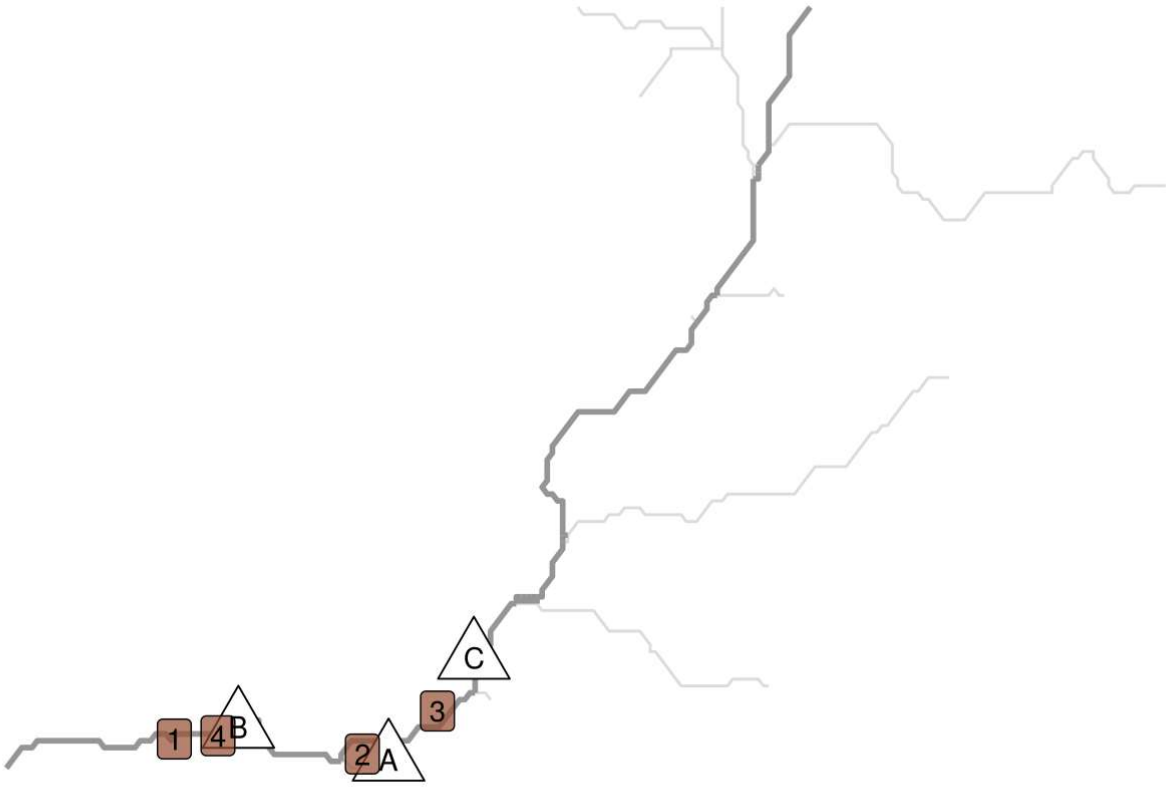
```
barriers_no_beaver<-filter(barriers, description != 'beaver meadow') %>%
  mutate(`Barrier type` = if_else(`barrier type` == 'dam', 'Dam', 'Waterfall'))

barriers_no_beaver<-bind_cols(barriers_no_beaver, tibble(barrier_letter = LETTERS[1:nrow(barrier
s_no_beaver)]))

phenos_subset %>%
  filter(Basin != 'Furnace') %>%
  ggplot() +
  #geom_path(data = hudson_map, aes(long, lat), linewidth = 12, color = 'grey') +
  #geom_sf(data = maps, aes(color = as.character(GRID_CODE_factor), linewidth = GRID_CODE_facto
r)) + # If I'm using my map
  geom_sf(data = maps, color = 'white', alpha = 0) +  # to use Felipe's map
  scale_linewidth_manual(values = c(0.5, 0.5, 1, 2, 3)) +
   scale_color_manual(values = c('#dddddd','#dddddd','#999999','#555555','#1B1B1B')) +
  scale_linewidth_identity() +
  geom_sf(data = filter(barriers_no_beaver, Basin == 'Moodna'), aes(shape=`Barrier type`), size
= 9, fill = 'white') +
  scale_shape_manual(name = 'Barrier type', labels = c('Dam', 'Waterfall'), values = c(24, 21))
+
  ggrepel::geom_label_repel(aes(lon, lat, label = paper_pop, fill = Basin), box.padding = 0.3, a
lpha = 0.7, size = 4) + # don't buffer too much, sites will disappear
  geom_sf_text(data = filter(barriers_no_beaver, Basin == 'Moodna'), aes(label = barrier_lette
r)) +
  labs(x='',y='', fill='') +
  theme(axis.text = element_blank(),
        plot.title = element_text(hjust = 0.5),
        axis.line = element_blank(),
        axis.ticks = element_blank(),
        panel.background = element_blank(),
        plot.background = element_blank(),
        panel.grid = element_blank()) +
  colScale_sites_fill +
  colScale_river_color +
  guides(linewidth = guide_legend('Stream order'),
      color = guide_legend('Stream order')) +
  ggspatial::annotation_scale()
```




```
ggsave('/workdir/smallmouth/hudson/redbreast/results/fig1.png',height=8, width = 13.3)

phenos_subset %>%
  dplyr::rename(`Sub-basin` = Basin) %>%
  ggplot(aes(lat, lon, color = `Sub-basin`)) +
  geom_point()
```


-73.9
-74.0
-74.1
-74.2
-74.3
lon
41.3
41.4
lat
Sub-basin
Cromline
Furnace
Moodna mainstem
Satterly
upper Moodna
Woodbury

```
ggsave('/workdir/smallmouth/hudson/redbreast/results/fig1_need_subbasin.png',height=5, width =
7)

phenos_subset %>%
  filter(watershed == 'Furnace_brook') %>%
  ggplot() +
  geom_sf(data = fb, aes(color = as.character(GRID_CODE_factor), linewidth = GRID_CODE_factor))
+
  scale_linewidth_manual(values = c(0.5, 0.5, 1, 2, 3)) +
  geom_sf(data = filter(barriers_no_beaver, Basin == 'Furnace_brook'), aes(shape=`Barrier type
`), size = 9, fill = 'white') +
  scale_shape_manual(name = 'Barrier type', labels = c('Dam', 'Natural'), values = c(24, 21)) +
  geom_label(aes(lon, lat, label = paper_pop, fill = Basin), alpha = 0.7) + # don't buffer too m
uch, sites will disappear
  geom_sf_text(data = filter(barriers_no_beaver, Basin == 'Furnace_brook'), aes(label = barrier_
letter)) +
  labs(x='',y='', fill='') +
  theme(axis.text.x = element_blank(),
        axis.text.y = element_blank(),
        plot.title = element_text(hjust = 0.5),
        legend.position="none",
        axis.line = element_blank(),
        axis.ticks = element_blank(),
        panel.background = element_rect(fill = 'white', color = 'white')) +
  colScale_sites_fill +
  colScale_river_color
```




```
ggsave('/workdir/smallmouth/hudson/redbreast/results/fig1_fb.png',height=5, width = 7)
```

Table 2: Barrier-specific metrics

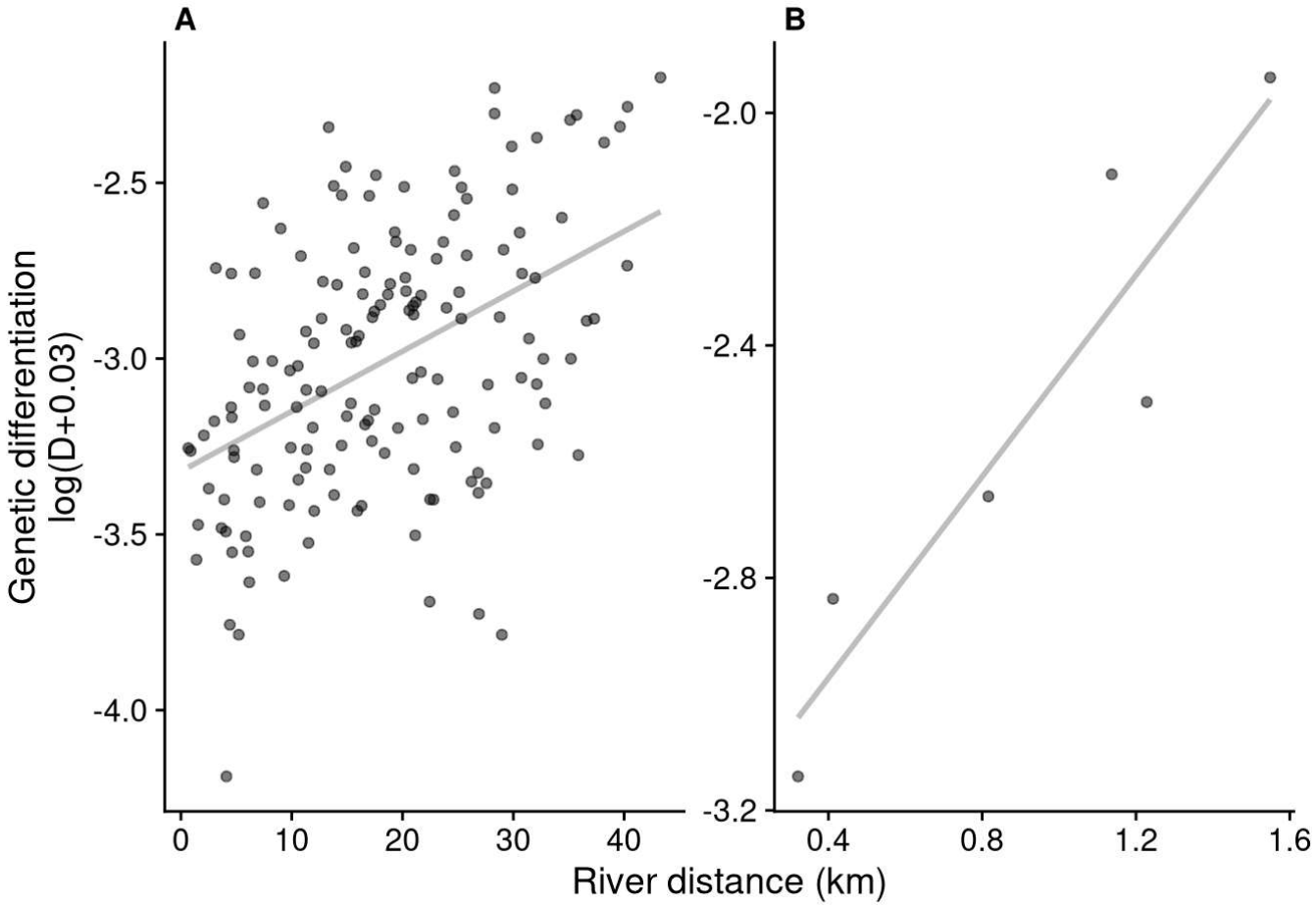
```
barriers_no_beaver %>%
  mutate(height_m = round(height_ft /3.28,1)) %>%
  dplyr::select(Barrier = barrier_letter, Name=name, `State ID` = NYS_ID, Type=`Barrier type`, `
Year built` = year_built, Height = height_m) %>%
  write_csv('/workdir/smallmouth/hudson/redbreast/results/table_barriers.csv')
```

prediction 1 (Figure 2-4): sunfish show low connectivity across the landscape

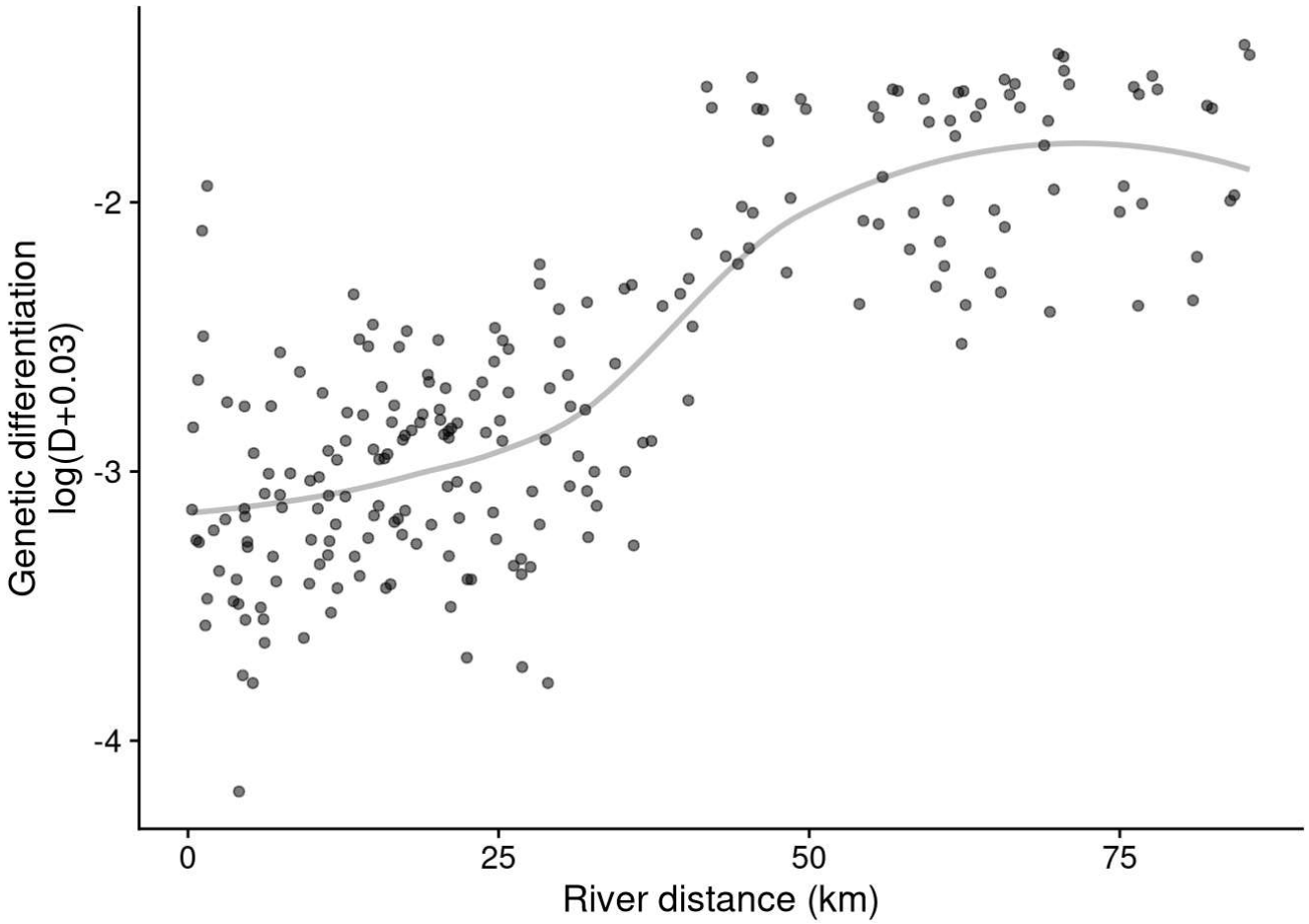
Figure 2: Isolation by distance for undammed

comparisons

```
# Plot  IBD by river basin
    ibd_in %>%
      mutate(D_trans = log(D+0.03),
             riv = if_else(pop1_river == 'Moodna', 'A', 'B')) %>%
      ggplot(aes(dist_km, D_trans)) + # , color = `Dam present`
      geom_smooth(method = 'lm', color = 'grey', se = F) +
      geom_point(alpha = 0.5) +
      labs(x = 'River distance (km)', y = 'Genetic differentiation\nlog(D+0.03)') +
      theme_cowplot() +
      facet_wrap(~riv, scales = 'free') +
      theme(strip.background = element_blank(),
            strip.text = element_text(hjust=0,
                                      face='bold'))
```



```
    ggsave('/workdir/smallmouth/hudson/redbreast/results/ibd_basin.svg', height = 4, width = 7)

# Plot IBD for both basins combined
    ibd_all %>%
      mutate(D_trans = log(D+0.03)) %>%
      ggplot(aes(dist_km, D_trans)) + # , color = `Dam present`
      geom_smooth(color = 'grey', se = F) +
      geom_point(alpha = 0.5) +
      labs(x = 'River distance (km)', y = 'Genetic differentiation\nlog(D+0.03)') +
      theme_cowplot()
```



```
    ggsave('/workdir/smallmouth/hudson/redbreast/results/ibd_all.png', height = 4, width = 6)

# Statistical comparison
   ibd_trans<-ibd_in %>%
     mutate(D_trans = log(D+0.03))

   for(riv_in in c('Moodna','Furnace_brook')){
     ibd_trans_filt<-filter(ibd_trans, pop1_river == riv_in)

     gg<-lme(D_trans ~ dist_km,
              random=~1|pop1_river, # for some reason it doesn't work without a random effect
              correlation=corMLPE(form=~pop1_site+pop2_site|pop1_river),
              data=  ibd_trans_filt,
              method="REML")
     print(paste0(riv_in,
                  ' marginal r(',
                  summary(gg)$tTable[2,3],
                  ')=',
                  round(MuMIn::r.squaredGLMM(gg)[1,1],2),
                  ', p=',
                  round(summary(gg)$tTable[2,5],3),
                  ', n=',
                  nrow(ibd_trans_filt)))
   }
```

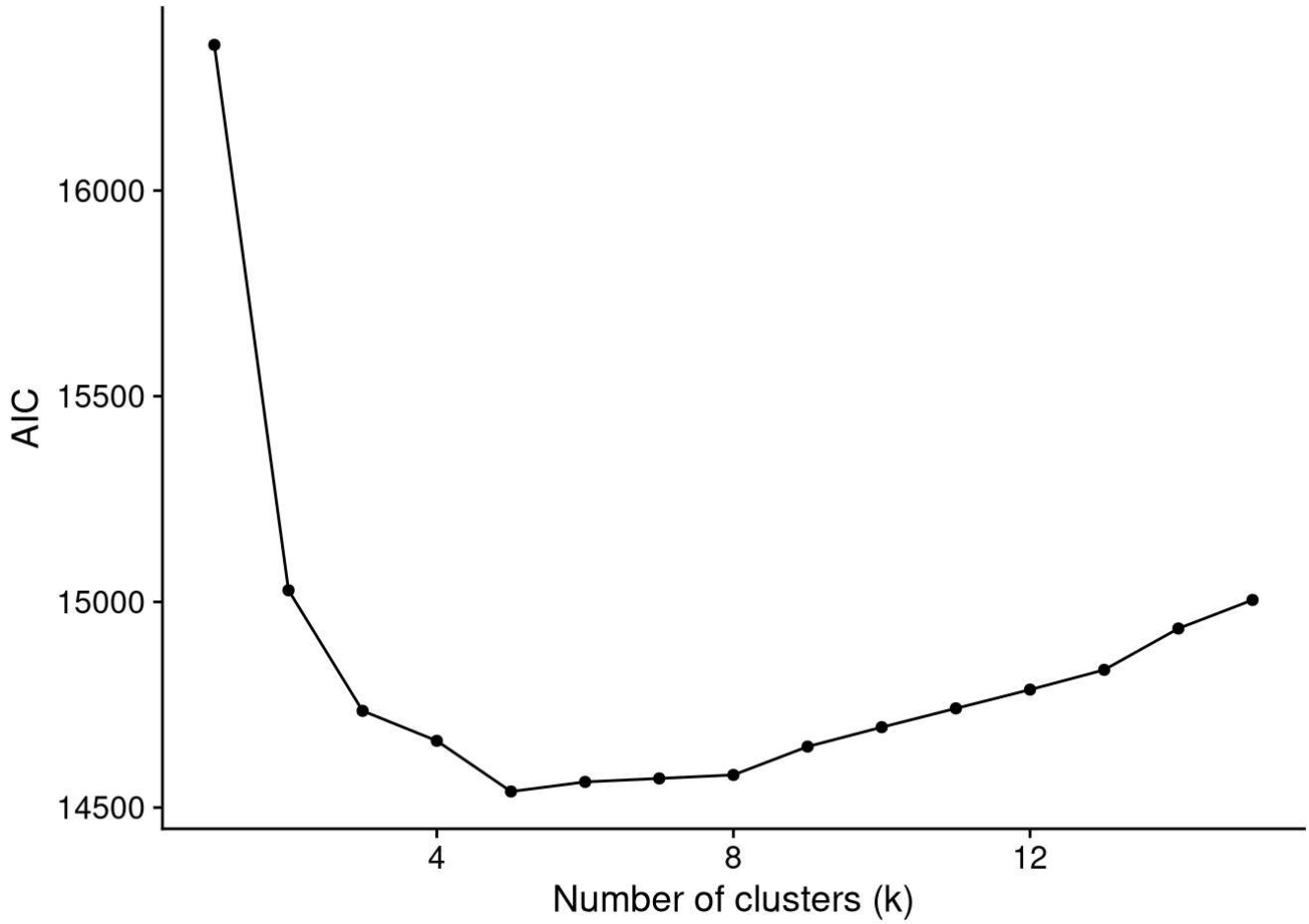
```
## [1] "Moodna marginal r(151)=0.17, p=0, n=153"
## [1] "Furnace_brook marginal r(4)=0.62, p=0, n=6"
```

Figure 3: snapclust and mds

genetic clustering with snapclust

Find optimal k

```
# optimal number of k
find_k<-tibble()
    for(criterion in c('AIC','BIC')){
      find_k<-snapclust.choose.k(obj_noRelats, max = 15, IC = criterion) %>%
        as_tibble(rownames = 'k') %>%
        mutate(k = as.integer(k),
               test = criterion) %>%
        bind_rows(find_k)
    }

find_k %>%
  filter(test == 'AIC') %>%
  ggplot(aes(k, value)) +
  geom_point() +
  geom_line() +
  theme_cowplot() +
  labs(x = 'Number of clusters (k)', y = 'AIC')
```

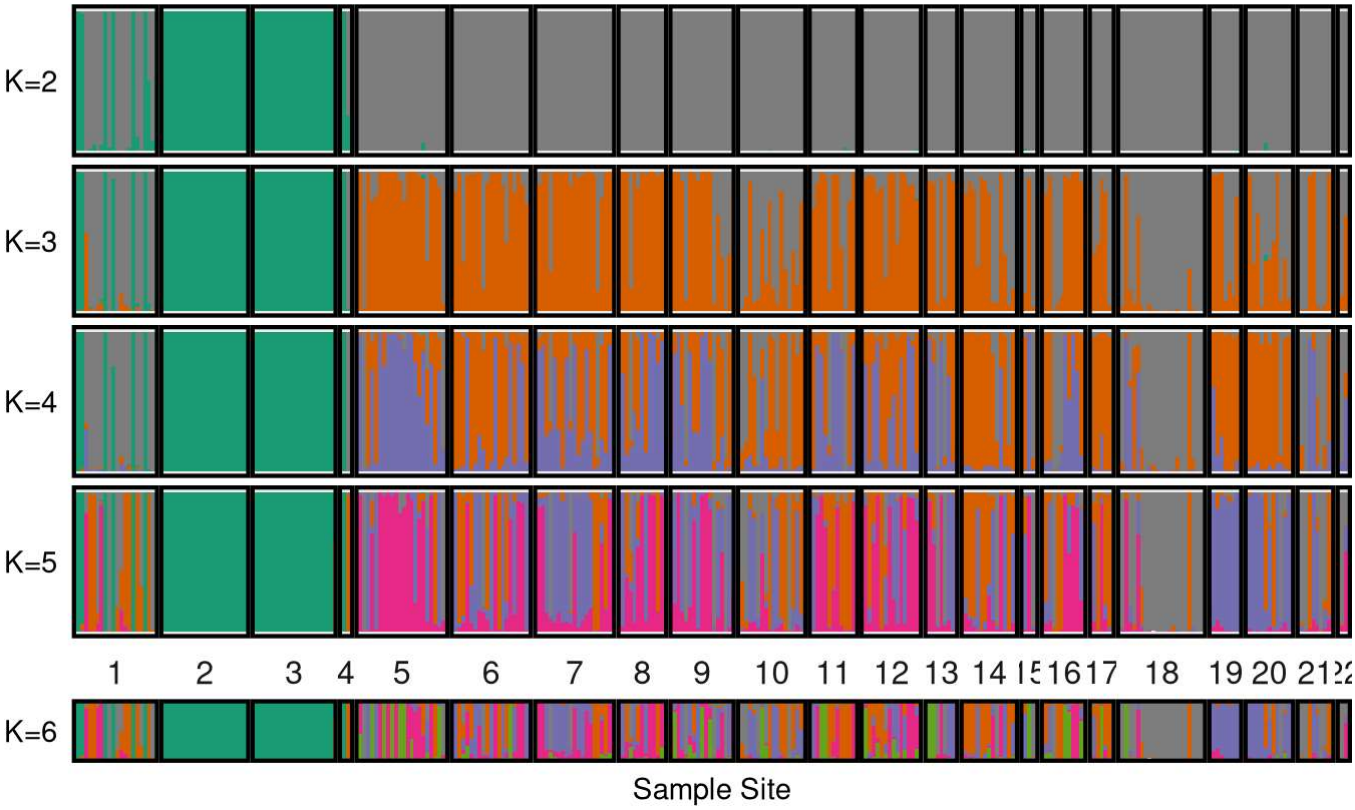



```
ggsave('/workdir/smallmouth/hudson/redbreast/results/find_k_allpops.png', height = 4, width = 4)
```

plot structure

```
map_out<-list()
k_range<-2:5 # leave this and crop off 2-4. For 2-8, change this to 2:8

# set colors
rc<-RColorBrewer::brewer.pal(length(k_range)+1, name = 'Dark2')
cols <- c('2' = rc[1], '3' = rc[2], '4' = rc[3], '5' = rc[4], '6' = rc[5], '7' = rc[6], '8' = rc
[7], '9' = rc[8])

for(k_in in c(k_range, max(k_range)+1)){
    struct_in<-snapclust(obj, k = k_in)$proba  %>%
      as_tibble(rownames = 'ind') %>%
      pivot_longer(-c(ind)) %>%
      left_join(dplyr::select(phenos, ind, paper_pop))

    if(k_in == max(k_range)+1){
      map_in<-ggplot(struct_in, aes(ind, value, fill = as.character(name))) +
      geom_bar(stat = 'identity', width = 1) +
      scale_fill_manual(values = cols) +
      facet_grid(~paper_pop, scales = "free_x", space = 'free_x') +
      labs(x = 'Sample Site', y = paste0('K=', k_in)) +
      theme(panel.spacing = unit(0, "lines"),
          axis.text = element_blank(),
          axis.ticks = element_blank(),
          axis.line = element_blank(),
          plot.margin = unit(c(0,0,0,0), "cm"),
          panel.border = element_rect(color = "black", fill = NA, size = 1.5),
          axis.title.y = element_text(angle = 0, vjust = 0.5),
          strip.text = element_text(size = 12),
          strip.background = element_blank(),
          plot.background = element_blank(),
          panel.grid = element_blank(),
          legend.position = 'none')

    } else{

      map_in<-ggplot(struct_in, aes(ind, value, fill = as.character(name))) +
      geom_bar(stat = 'identity', width = 1) +
      scale_fill_manual(values = cols) +
      facet_grid(~paper_pop, scales = "free_x", space = 'free_x') +
      labs(x = NULL, y = paste0('K=', k_in)) +
      theme(panel.spacing = unit(0, "lines"),
          axis.text = element_blank(),
          axis.ticks = element_blank(),
          strip.background = element_blank(),
          plot.background = element_blank(),
          strip.text.x = element_blank(),
          axis.line = element_blank(),
          panel.grid = element_blank(),
          plot.margin = unit(c(0,0,0,0), "cm"),
          panel.border = element_rect(color = "black", fill = NA, size = 1.5),
          axis.title.y = element_text(angle = 0, vjust = 0.5),
          legend.position = 'none')
```

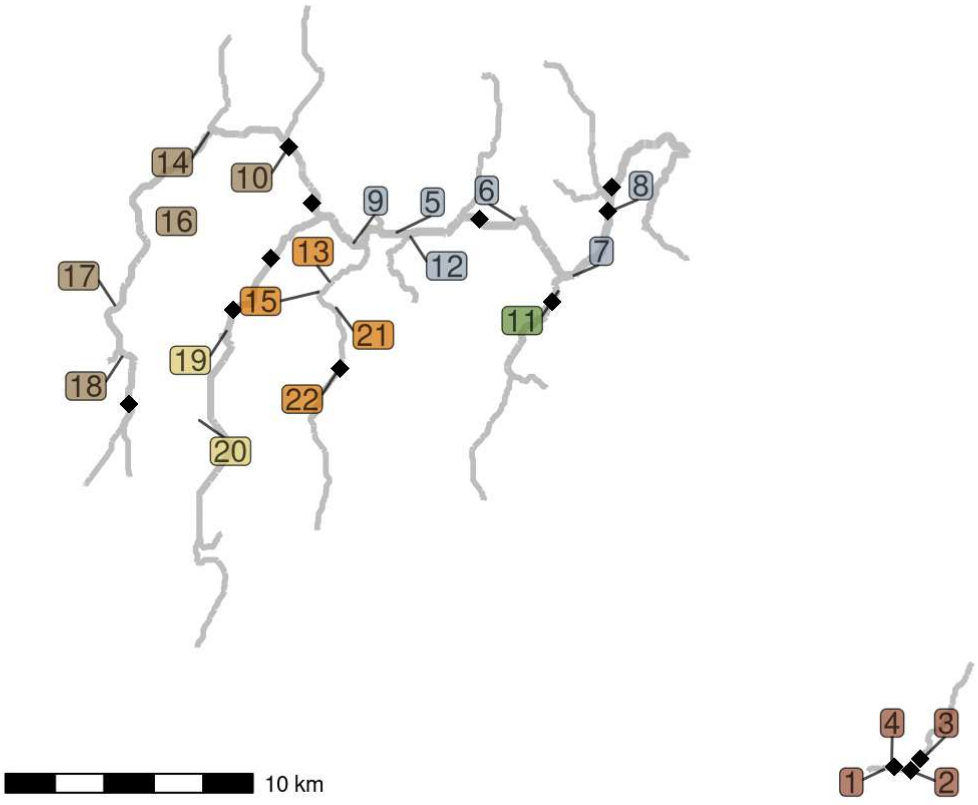
```
    }

    map_out[[k_in]]<-map_in
}

plot_grid(plotlist =  map_out, ncol = 1, scale = 1)
```




```
ggsave('/workdir/smallmouth/hudson/redbreast/results/structure_allpops.png', height = 8, width = 15) # leave height at 8 for both
```

Make a little reference map

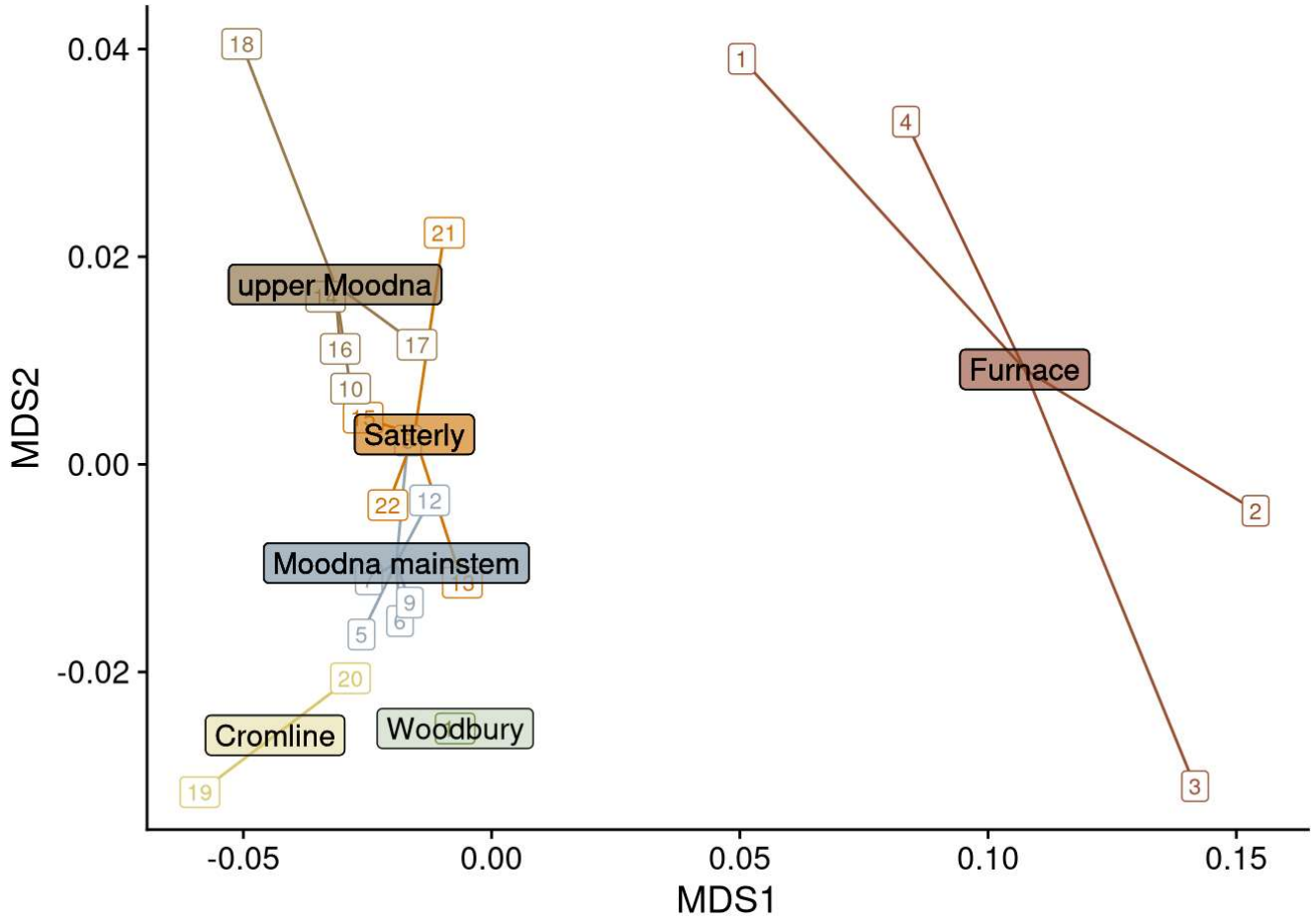
```
phenos_subset %>%
  ggplot() +
  geom_sf(data = maps, aes(linewidth = GRID_CODE_factor), color = 'grey') +
  geom_sf(data = filter(barriers, description == 'intact dam'), size = 3, shape = 18,  fill = 'b
lack') +
  scale_linewidth_manual(values = c(0, 0, 1, 1.25, 1.5)) +
  scale_shape_manual(name = 'Barrier type', labels = c('Dam', 'Natural'), values = c(24, 21)) +
  ggrepel::geom_label_repel(aes(lon, lat, label = paper_pop, fill = Basin), box.padding = 0.4, l
abel.padding = .08, size = 4, alpha = 0.7) +
  colScale_sites_fill +
  labs(x='',y='', fill='') +
  theme(axis.text.x = element_blank(),
        axis.text.y = element_blank(),
        plot.title = element_text(hjust = 0.5),
        legend.position = 'none',
        axis.line = element_blank(),
        axis.ticks = element_blank(),
        panel.background = element_rect(fill = 'transparent', color = 'transparent')) +
        guides(linewidth = guide_legend('Stream order'),
        color = guide_legend('Stream order')) +
       ggspatial::annotation_scale()
```




```
ggsave('/workdir/smallmouth/hudson/redbreast/results/small_map.png', height = 5, width = 5)
```

mds of Jost D

```
# For each transect
  mds_D<-cmdscale(as.matrix(pairwise_D(obj)))

  mds_D %>%
    as.data.frame() %>%
    rownames_to_column(var='pop') %>%
    separate(pop, into = c('xx','river','Transect'), sep = '-') %>%
    left_join(cols_in, by = 'river') %>%
    group_by(Basin) %>%
    mutate(mV1 = mean(V1),
           mV2 = mean(V2)) %>%
    ggplot() +
    geom_segment(mapping = aes(V1, V2, color = Basin, xend = mV1, yend = mV2)) +
    geom_label(aes(V1, V2, color = Basin, label = Transect), size = 3) +
    geom_label(aes(mV1, mV2, fill = Basin, label = Basin), size = 4, alpha = 0.2) +
    theme_cowplot() +
    colScale_sites_fill +
    colScale_sites_color +
    labs(x='MDS1',y='MDS2') +
    theme(legend.position="none")
```




```
  ggsave('/workdir/smallmouth/hudson/redbreast/results/mds_allBasins.png',height=4,width=6)
```

Figure 4: Infer recent migration magnitude and direction, and get migrant proportion (BayesAss)

If we do this with SNPs in the future, we can auto-tune the mixing parameters with https://github.com/stevemussmann/BA3-SNPS-autotune (https://github.com/stevemussmann/BA3-SNPS-autotune)

Set up input files. Use the version that includes related individuals…we are looking for migrants from one patch to another!

```
genind2df(obj, sep = '/') %>%
  rownames_to_column(var = 'ind') %>%
  mutate(pop = if_else(pop == 'Furnace_brook-Furnace-2','Furnace_brook-Furnace-3',pop)) %>%
  # filter(pop %in% filter(phenos_subset, n > 9)$pop) %>% # If we want to run with 10+ inds
  pivot_longer(-c(ind, pop)) %>%
  mutate(value = replace_na(value, '0/0')) %>% # as requested by the documentation
  separate(value, into = c('al1','al2'), sep = '/') %>%
  write_tsv('/workdir/smallmouth/hudson/redbreast/BayesAss/input.txt', col_names = F)
```

run on CMD (this took 2 hours on cluster) -i is number of MCMC iterations -b is the burn-in to discard at the beginning -n is the thinning number -t says to generate a trace so we can check the mcmc fitting -0 is output -s is seed (123) -v is verbose, print results to screen (recommended for the beginning where we are seting the parameters) -g gives us an output of detailed genotype information (recommended) -u writes the input parameters to the beginning of the output file (recommended) -m gives the mixing parameter for migration rates (output 1), default 0.1 -a gives the mixing parameter for allele frequencies (output 3), default 0.1 -f gives the mixing parameter for inbreeding coefficients (output 4), default 0.1

Produces an output with % acceptance of 5 parameters: (1) migration rates, (2) individual migrant ancestries, (3) allele frequencies, (4) inbreeding coefficients, (5) missing genotypes

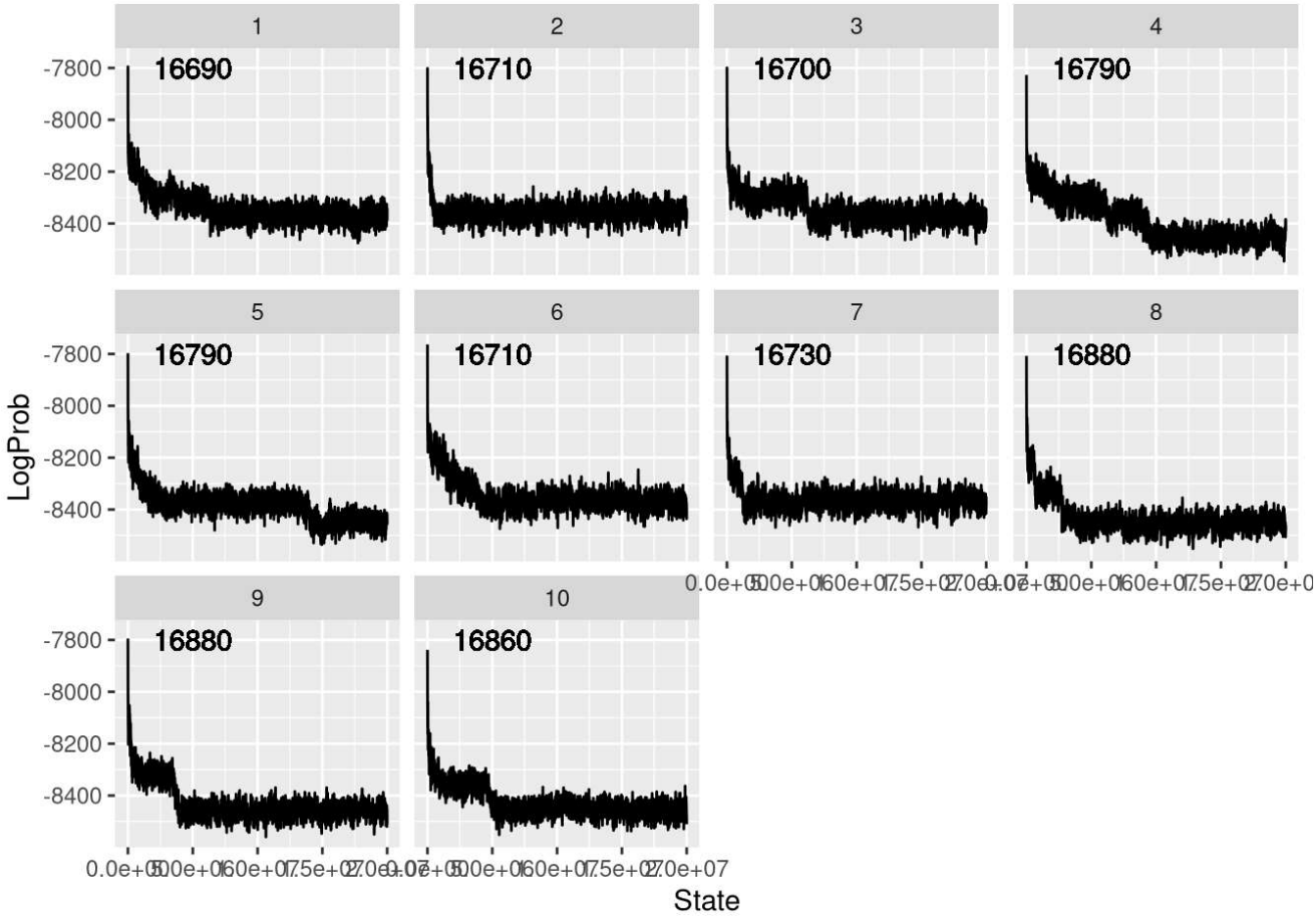
To start, run it without nohup, and once its around 7-8% done, check the % acceptance of values 1, 3, and 4 at the beginning. They should be ~20-60% First run, they were at 0.42 (just right), 0.76 (too high), and 0.83 (too high) so we increased the proposal step length for -a and -f Second run, after increasing -a and -f to 0.3, all were still too high, including migration rates, so increased all I can't increase past 1, but I brought them all up to 1. In the manual, they say that "If the likelihood surface is very flat, for example, as may occur with weakly informative data, acceptance rates above 0.6 may occur even with a proposal step length of 1." Kill program with [CTRL]+C

The recommendation is to run several different tests, each with a different starting seed, then seeing what results are consistent between all of them

```
cd /workdir/smallmouth/hudson/redbreast/BayesAss

# testing
/programs/BA3-3.0.5/BA3MSAT -i10000000 -b1000000 -n1000 -t -v -g -u -s1 -m1 -a1 -f1 -o output_1.
txt input.txt

# production

for run in `seq 1 10`; do

cd /workdir/smallmouth/hudson/redbreast/BayesAss
rm -r seed$run
mkdir seed$run
cd seed$run

nohup /programs/BA3-3.0.5/BA3MSAT -i20000000 -b1000000 -n1000 -t -v -g -u -s$run -m1 -a1 -f1 -o
output.txt /workdir/smallmouth/hudson/redbreast/BayesAss/input.txt > BayesAss_run.nohup &

done
```

Check convergence and output mixing parameters. Trace wasn’t converging, so I boosted the number of chains to 20 mil from 10 mil

```
# input values of burnin and sampling interval to calculate the Bayesian deviance (Meirmans 201
3)
    burnin<-1000000
    samp<-1000

trace_out<-tibble()
D_out<-tibble()
acc_out<-tibble()
for(seed in 1:10){
     # Diagnose convergence
        trace_in<-read_tsv(paste0('/workdir/smallmouth/hudson/redbreast/BayesAss/seed',seed,'/BA
3trace.txt')) %>%
          mutate(seed = seed)

     # Calculate bayesian deviance
        range <- (trace_in$State > burnin & trace_in$State %% samp == 0)
        D <- -2*mean(trace_in$LogProb[range])

     # Check the finishing acceptance %
        acc<-tibble(accept = tail(readLines(paste0('/workdir/smallmouth/hudson/redbreast/BayesAs
s/seed',seed,'/BayesAss_run.nohup')))[2]) %>%
          mutate(accept = str_sub(accept, -30, -1),
                 seed = seed)

     # export all
        trace_out<-bind_rows(trace_out, trace_in)
        D_out<-bind_rows(D_out, tibble(seed = seed, D = D))
        acc_out<-bind_rows(acc_out, acc)
}

# Visualize trace
    trace_out %>%
      left_join(D_out, by = 'seed') %>%
      ggplot(aes(State, LogProb)) +
      geom_line() +
      geom_text(aes(x = 5e6, y = -7800, label = round(D, -1))) +
      facet_wrap(~seed)
```




```
# check acceptance rates
    acc_out
```

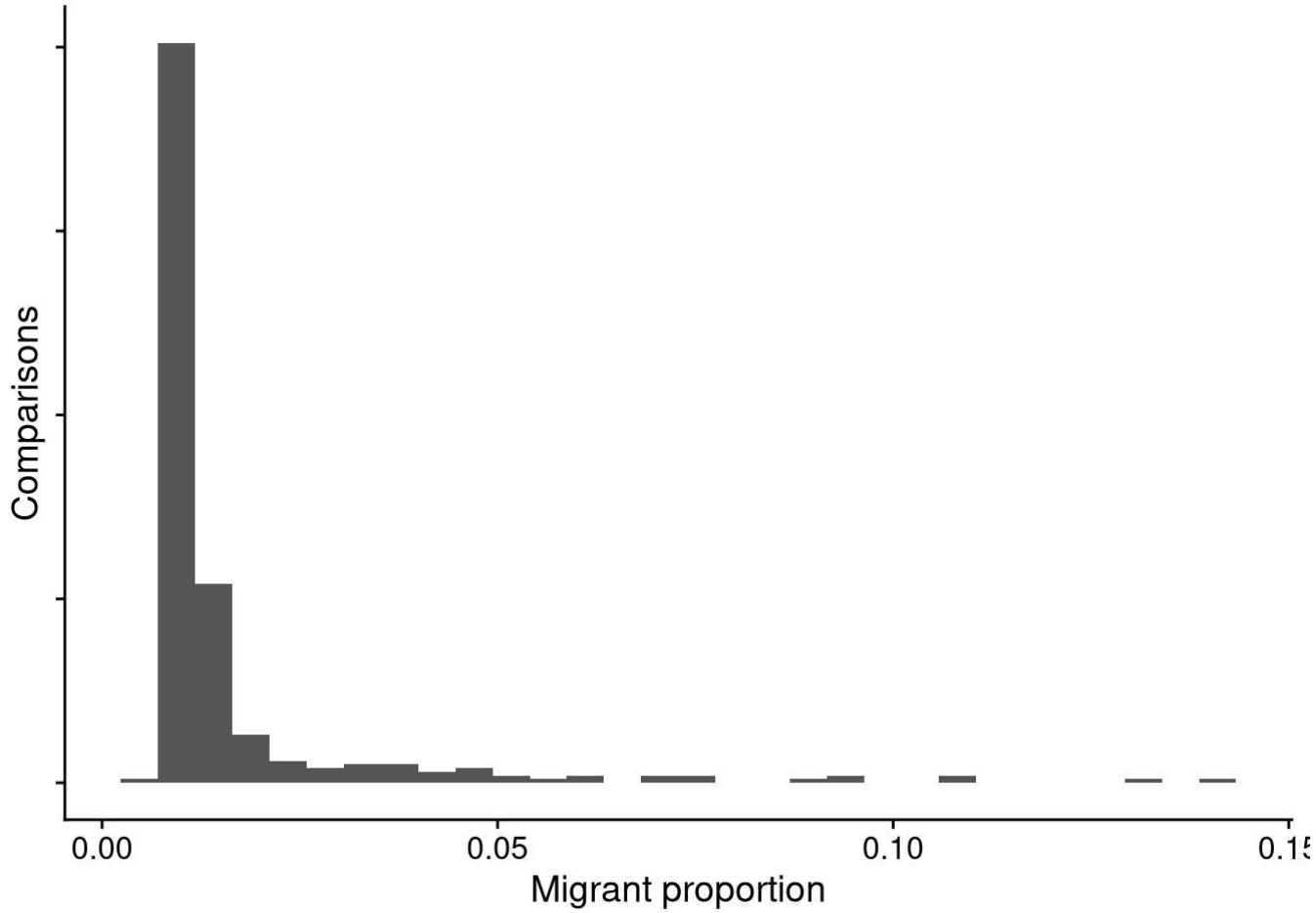
```
## # A tibble: 10 × 2
##    accept                          seed
##    <chr>                          <int>
##  1 (0.71, 0.01, 0.73, 0.75, 0.25)     1
##  2 (0.72, 0.01, 0.74, 0.76, 0.24)     2
##  3 (0.71, 0.01, 0.73, 0.75, 0.25)     3
##  4 (0.73, 0.01, 0.75, 0.77, 0.24)     4
##  5 (0.71, 0.01, 0.75, 0.77, 0.24)     5
##  6 (0.71, 0.01, 0.73, 0.75, 0.25)     6
##  7 (0.71, 0.01, 0.74, 0.76, 0.24)     7
##  8 (0.73, 0.01, 0.77, 0.79, 0.24)     8
##  9 (0.75, 0.01, 0.77, 0.79, 0.24)     9
## 10 (0.74, 0.01, 0.77, 0.79, 0.24)    10
```

```
# Choose the seed that has the most stable trace, lowest bayesian deviance score. It also has th
e lowest acceptance rates, although still high
# individual migrant ancestries are really low.
```

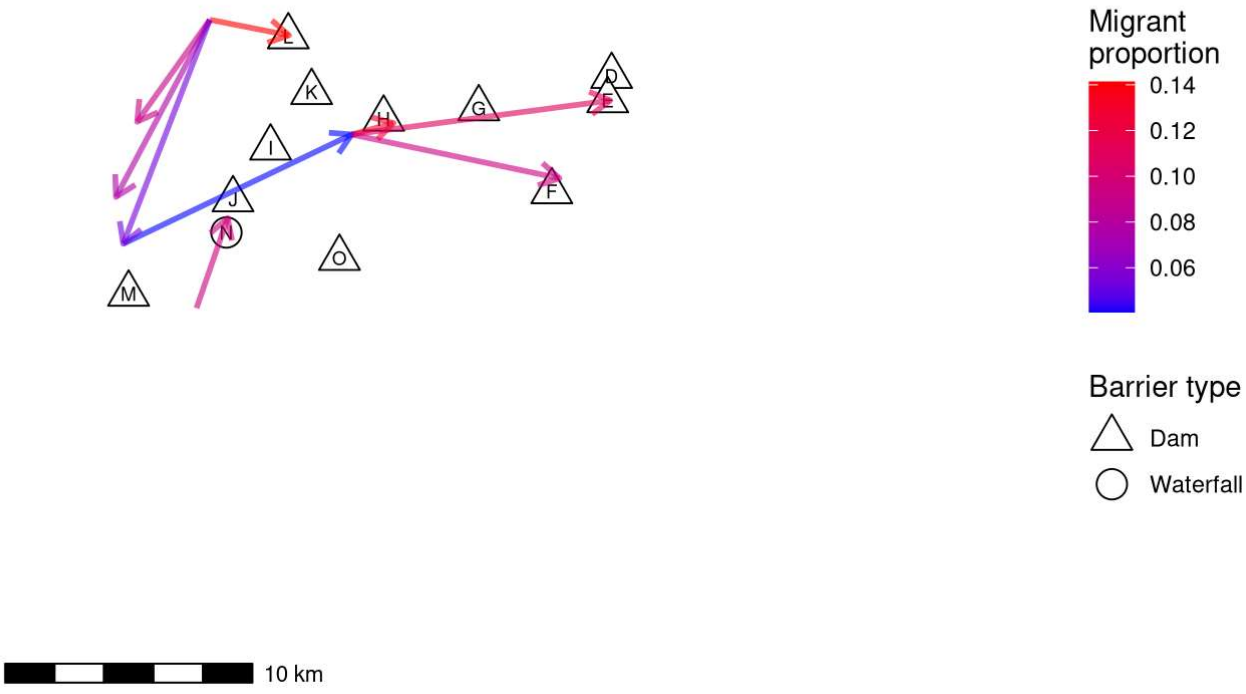
Read in output and make a map

```
seed<-1 # This is set in the previous step

# Read in results on direction and magnitude of migration
    output<-readLines(paste0('/workdir/smallmouth/hudson/redbreast/BayesAss/seed',seed,'/output.
txt')) # This gives direction of effective migration

  # make population labels
      ass_joiner<-output[grep('Population Index -> Population Label:', output)+2] %>%
        as.tibble() %>%
        separate(value, into = paste0('temp',1:(nPop(obj)+2)), sep = ' ') %>%
        mutate(temp1 = 1) %>%
        pivot_longer(-temp1) %>%
        separate(value, into = c('value','pop'), sep = '->') %>%
        dplyr::select(c('value','pop'))

  # Put together a list of all pairwise comparisons
     migrants_in<-output[(grep('Migration Rates:', output)+2):(grep('Inbreeding Coefficients:',
output)-2)] %>%
        as.tibble() %>%
        separate(value, into = paste0('temp',1:(nPop(obj)+1)), sep = ' m') %>%
        dplyr::select(-temp1) %>%
        mutate(row_n = row_number()) %>%
        pivot_longer(-row_n) %>%
        separate(value, into = c('pops','estimates'), sep = ':') %>%
        separate(pops, into = c('sink_pop', 'source_pop'), sep = '\\]\\[') %>%
        separate(estimates, into = c('migrant_prop', 'sd'), sep = '\\(') %>%
        mutate(sink_pop = str_replace(sink_pop, '\\[',''),
               source_pop = str_replace(source_pop, '\\]',''),
               sd = as.numeric(str_replace(sd, '\\)','')),
               seed = seed) %>%
        filter(sink_pop != source_pop) %>%
        mutate(CI = sd*1.96,
          migrant_prop = as.numeric(migrant_prop),
          migrant_ci_0 = if_else((migrant_prop-CI)<0, 0, migrant_prop),
          row_n = row_number()) %>%
        dplyr::select(row_n, sink=sink_pop, source=source_pop, migrant_prop, migrant_ci_0, seed)
%>%
        pivot_longer(-c(row_n, migrant_prop, migrant_ci_0, seed)) %>%
        left_join(ass_joiner, by = 'value') %>%
        left_join(phenos_subset, by = 'pop') %>%
        dplyr::select(c(row_n, migrant_prop, migrant_ci_0, name, pop, lat, lon, seed)) %>%
        pivot_wider(names_from = name, values_from = c(pop, lat,lon)) %>%
       mutate(sinksource = paste0(pop_sink, pop_source))

  # Overall histogram of migrant proportion of each population
      migrants_in %>%
        mutate(across = if_else(str_detect(sinksource, 'Moodna') & str_detect(sinksource, 'Furna
ce'), 1, 0)) %>%
        filter(across == 0) %>%
        ggplot(aes(migrant_prop)) +
        geom_histogram() +
        theme_cowplot() +
```

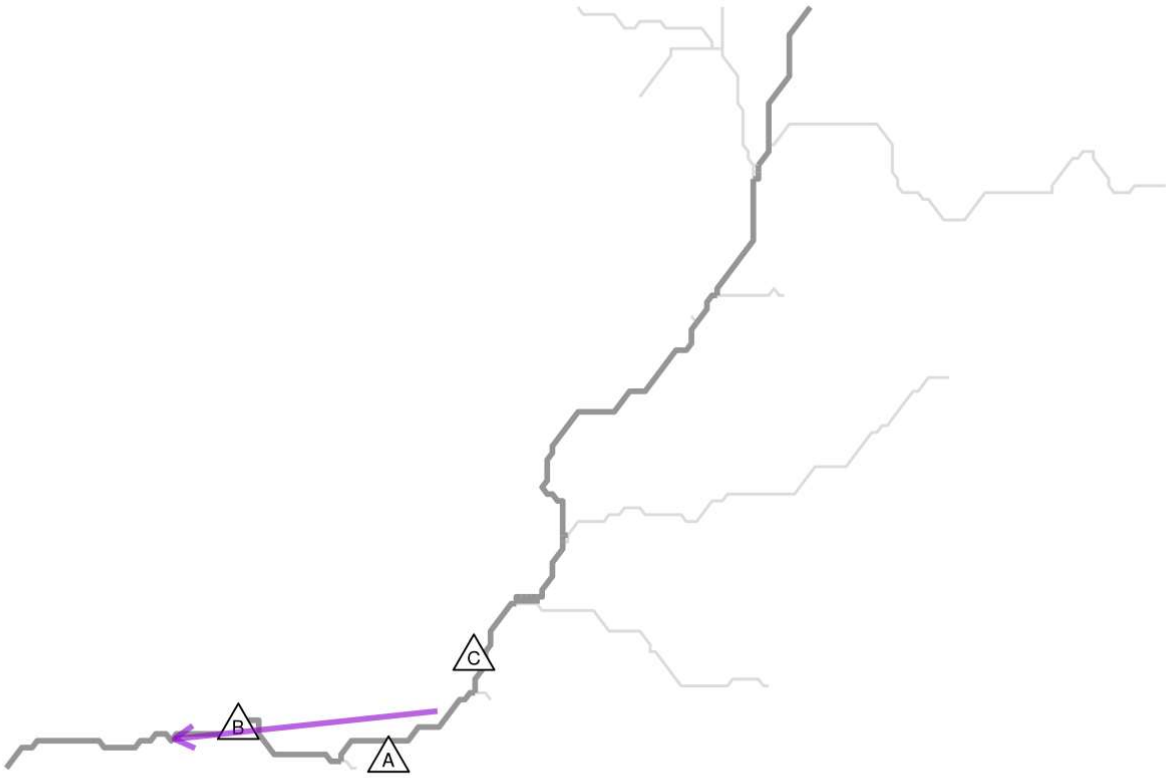
```
        labs(x = 'Migrant proportion', y = 'Comparisons') +
        theme(axis.text.y = element_blank())
```



```
      ggsave('/workdir/smallmouth/hudson/redbreast/results/ba3_migrant_proportion.png', height =
2, width = 3)

  # Only grab the links that have CI greater than zero
      migrants_confirmed<-migrants_in %>%
        filter(migrant_ci_0 > 0)

  # Plot Moodna
      migrants_confirmed %>%
       filter(str_detect(sinksource, 'Moodna')) %>%
       ggplot() +
       # geom_sf(data = maps, aes(linewidth = GRID_CODE_factor, color = as.character(GRID_CODE_f
actor))) +
       geom_sf(data = maps, color = 'white', alpha = 0) +  # to use Felipe's map
       scale_linewidth_manual(values = c(0.5, 0.5, 1, 2, 3)) +
       colScale_river_color +
       guides(linewidth = guide_legend('Stream order'),
           color = guide_legend('Stream order')) +
       geom_sf(data = filter(barriers_no_beaver, Basin == 'Moodna'), aes(shape=`barrier type`),
size = 5, fill = 'white') +
       scale_shape_manual(name = 'Barrier type', labels = c('Dam', 'Waterfall'), values = c(24,
21)) +
       geom_sf_text(data = filter(barriers_no_beaver, Basin == 'Moodna'), aes(label = barrier_le
tter), size = 2.5) +
       ggnewscale::new_scale_color() +
       geom_segment(aes(x = lon_source, xend = lon_sink, y = lat_source, yend = lat_sink, color
= migrant_prop), # , color = mean_migrant_prop
                    arrow = arrow(length = unit(0.3,"cm")), size = 1, alpha = 0.6) +
       scale_color_gradient(low = 'blue',
                   high = 'red',
                   limits = c(min(migrants_confirmed$migrant_prop),
                              max(migrants_confirmed$migrant_prop))) +
        labs(x='',y='', color='Migrant\nproportion') +
       ggspatial::annotation_scale() +
       theme(axis.text = element_blank(),
             plot.title = element_text(hjust = 0.5),
             axis.line = element_blank(),
             axis.ticks = element_blank(),
             panel.background = element_blank(),
             plot.background = element_blank(),
             panel.grid = element_blank())
```


L
K
H
G
D
E
I
F
J
N
O
M
Migrant proportion
0.14
0.12
0.10
0.08
0.06
Barrier type
Dam
Waterfall
10 km

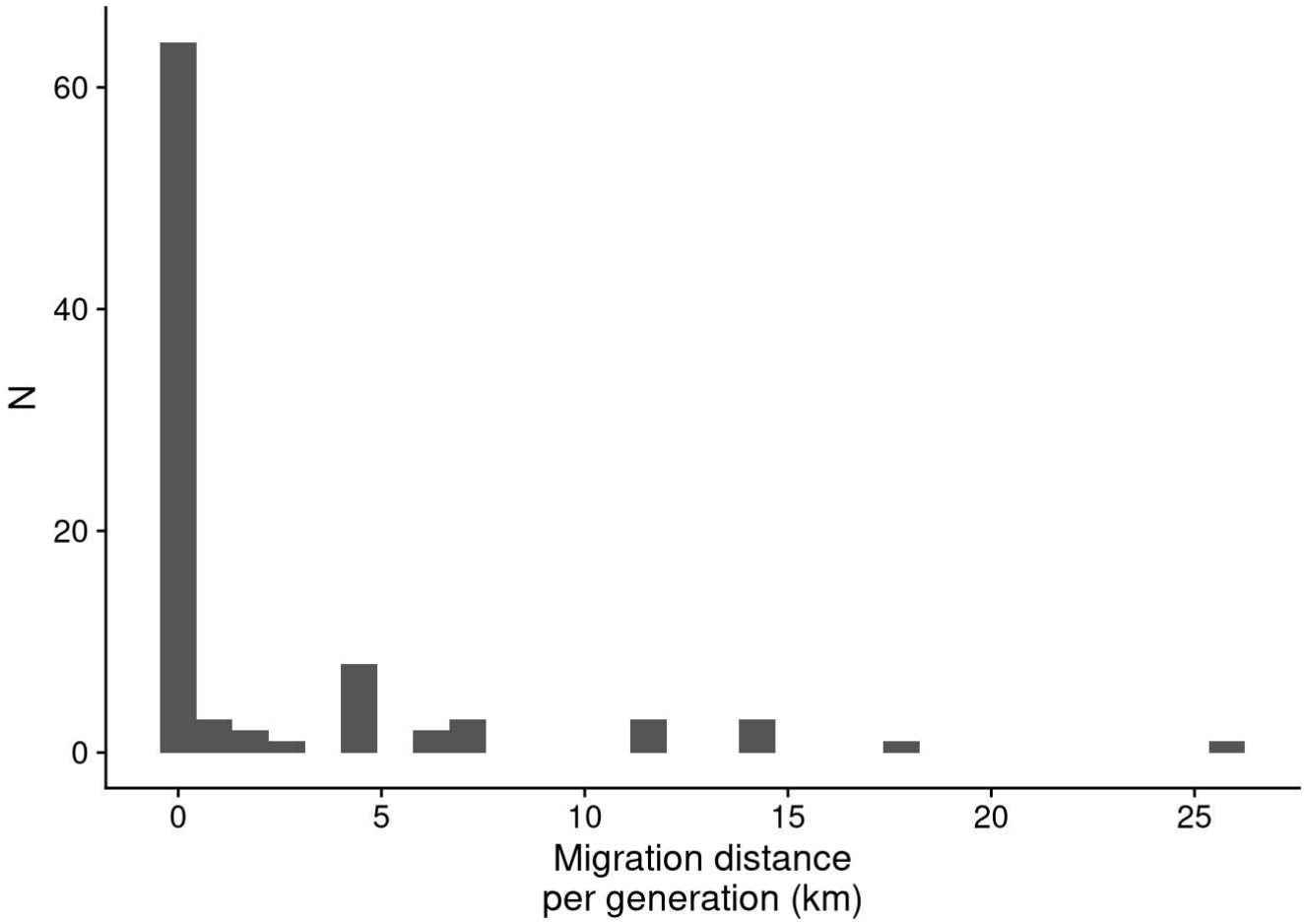
```
      ggsave('/workdir/smallmouth/hudson/redbreast/results/ba3_overview.png',height=5, width =
8)

  # Plot Furnace
     migrants_confirmed %>%
       filter(str_detect(sinksource, 'Furnace')) %>%
       ggplot() +
       geom_sf(data = fb, aes(linewidth = GRID_CODE_factor, color = as.character(GRID_CODE_facto
r))) +
       scale_linewidth_manual(values = c(0.5, 0.5, 1, 2, 3)) +
       colScale_river_color +
       guides(linewidth = guide_legend('Stream order'),
           color = guide_legend('Stream order')) +
       geom_sf(data = filter(barriers_no_beaver, Basin == 'Furnace_brook'), aes(shape=`barrier t
ype`), size = 5, fill = 'white') +
       scale_shape_manual(name = 'Barrier type', labels = c('Dam', 'Waterfall'), values = c(24,
21)) +
       geom_sf_text(data = filter(barriers_no_beaver, Basin == 'Furnace_brook'), aes(label = bar
rier_letter), size = 2.5) +
       ggnewscale::new_scale_color() +
       geom_segment(aes(x = lon_source, xend = lon_sink, y = lat_source, yend = lat_sink, color
= migrant_prop),
                    arrow = arrow(length = unit(0.3,"cm")), size = 1, alpha = 0.6) +
       scale_color_gradient(low = 'blue',
                   high = 'red',
                   limits = c(min(migrants_confirmed$migrant_prop),
                              max(migrants_confirmed$migrant_prop))) +
      labs(x='',y='') +
       theme(axis.text.x = element_blank(),
          axis.text.y = element_blank(),
          plot.title = element_text(hjust = 0.5),
          legend.position="none",
          axis.line = element_blank(),
          axis.ticks = element_blank(),
          panel.background = element_rect(fill = 'white', color = 'white'))
```


C
B
A

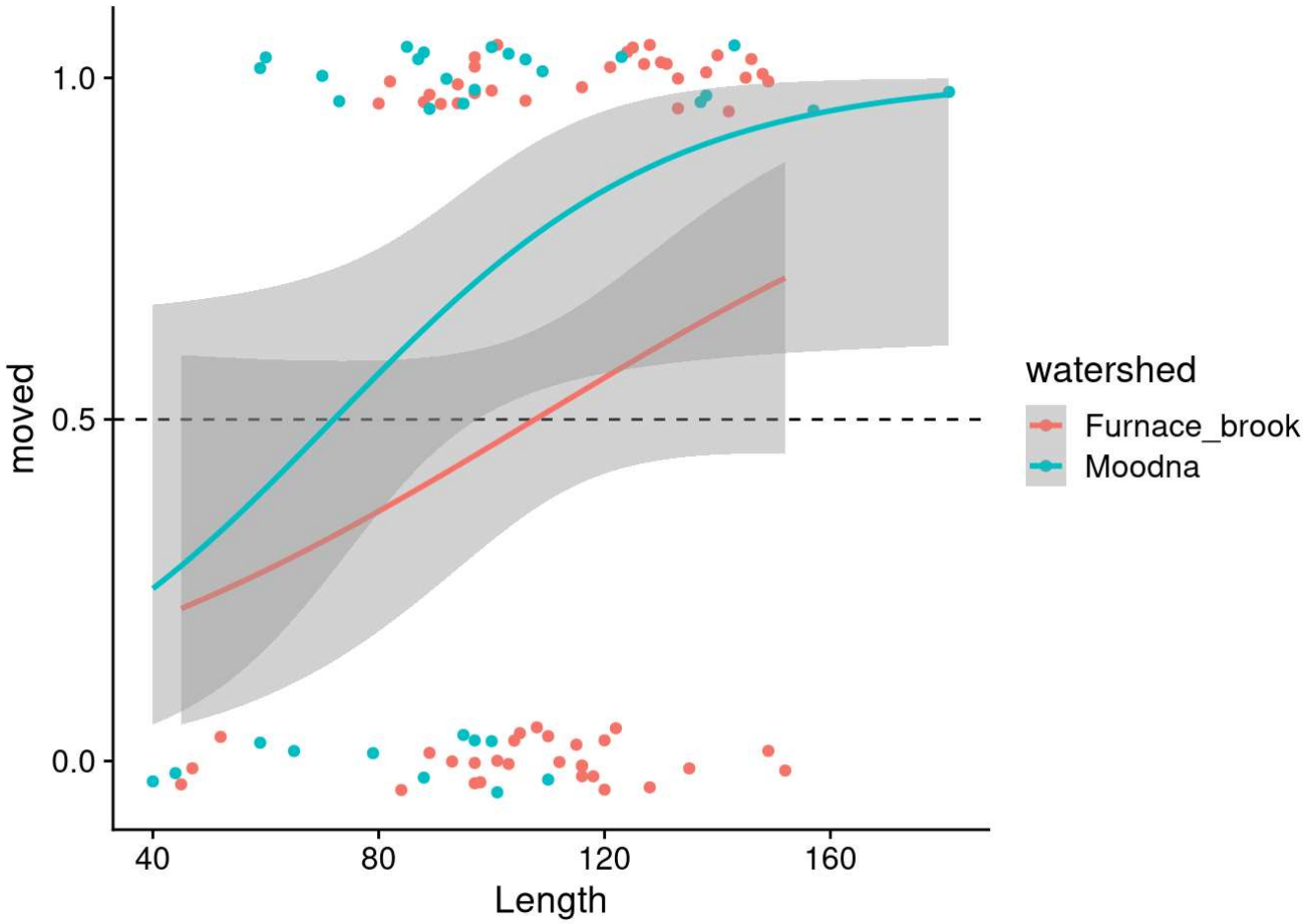
```
     ggsave('/workdir/smallmouth/hudson/redbreast/results/ba3_fb.png',height=5, width = 8)

# read in individual ancestry
    ances<-readLines(paste0('/workdir/smallmouth/hudson/redbreast/BayesAss/seed',seed,'/BA3indi
v.txt'))

  ancest_out<-tibble() # grab ancestry info from each individual
  for(line_in in grep('Individual', ances)){
    ancest_in<-tibble(ind = as.character(str_sub(ances[line_in], 14, 20)),
           migrants = ances[(line_in+5):(line_in+7)]) %>%
      separate(migrants, into = as.character(1:100), sep = ' ') %>%
      discard(~all(is.na(.) | . =="")) %>% # remove empty columns
      pivot_longer(-ind) %>%
      separate(value, into = c('x','prob'), sep = ':') %>%
      mutate(prob = as.double(prob)) %>%
      filter(prob > 0) %>%
      mutate(x = str_sub(x, 2, -2)) %>%
      separate(x, into = c('source','migrant_gen'), sep = ',') %>%
      left_join(dplyr::rename(ass_joiner, source=value, SOURCE = pop)) %>%
      dplyr::select(-c(name, source))

    ancest_out<-bind_rows(ancest_out, ancest_in)
  }

 ancest_data<- ancest_out %>% # bind this to pairwise distance and get the median distance moved
per generation
    left_join(dplyr::select(phenos, ind, pop, Length, watershed)) %>%
    filter(prob>0.95) %>%
    rowwise() %>%
    mutate(pop12 = paste0(sort(c(SOURCE, pop)), collapse = '/')) %>%
    ungroup() %>%
    left_join(pairwise_physical_genet_dist, by = 'pop12') %>%
    mutate(distance = replace_na(distance, 0)/1000,
           distance_gen = if_else(migrant_gen == 2, distance/2, distance),
           mean_d = mean(distance_gen),
           med_d = median(distance_gen),
           n = n(),
           prop_zero = sum(distance == 0)/n,
           moved = if_else(distance == 0, 0, 1))

 # How many fish moved?
 nrow(filter(ancest_data, distance_gen < 1)) / nrow(ancest_data)
```

```
## [1] 0.7142857
```

```
ancest_data %>%
  ggplot(aes(distance_gen)) +
   geom_histogram(bins = 30) +
   #geom_vline(aes(xintercept = mean_d), color = 'red', lty = 'dashed', size = 1) +
   #geom_vline(aes(xintercept = med_d), color = 'blue', lty = 'dashed', size = 1) +
   theme_cowplot() +
   scale_x_continuous(breaks = seq(0, 100, 5)) +
   labs(x= 'Migration distance\nper generation (km)', y = 'N')
```




```
  ggsave('/workdir/smallmouth/hudson/redbreast/results/ba3_migration.png', height = 3, width =
3)

# Smaller fish were more likely to be found in their natal population
  ancest_data %>%
    ggplot(aes(Length, moved, color = watershed)) +
    geom_hline(yintercept = 0.5, lty = 'dashed') +
    geom_jitter(width = 0, height = 0.05) +
    geom_smooth(method = "glm", method.args = list(family = "binomial")) +
    theme_cowplot() +
    scale_y_continuous(breaks = c(0, 0.5, 1))
```




```
glm_model<-glm(moved ~ Length + watershed,
               data = ancest_data,
               family = binomial(link = "logit"))

summary(glm_model)
```

```
##
## Call:
## glm(formula = moved ~ Length + watershed, family = binomial(link = "logit"),
##     data = ancest_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6617  -1.1586   0.7144   0.9843   1.4893
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -2.702986   1.106079  -2.444   0.0145 *
## Length           0.024929   0.009612   2.593   0.0095 **
## watershedMoodna  1.051768   0.524712   2.004   0.0450 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 123.16  on 89  degrees of freedom
## Residual deviance: 113.63  on 87  degrees of freedom
##   (1 observation deleted due to missingness)
## AIC: 119.63
##
## Number of Fisher Scoring iterations: 3
```

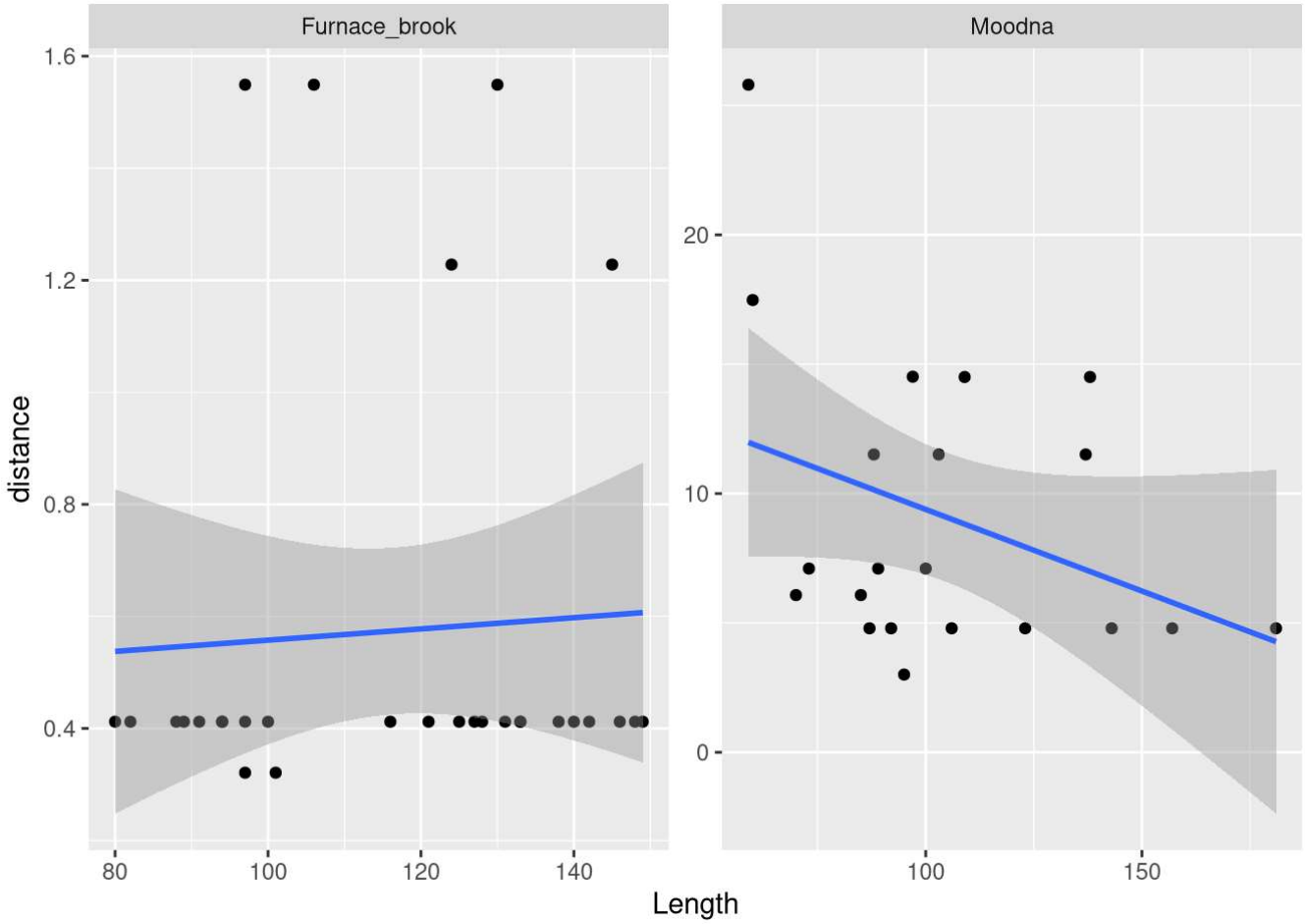
```
  with(summary(glm_model), 1 - deviance/null.deviance) # McFadden's R2
```

```
## [1] 0.0773694
```

```
  MASS::dose.p(glm(moved ~ Length + watershed,
               data = ancest_data,
               family = binomial(link = "logit")), p = c(0.25, 0.5, 0.75))
```

```
##               Dose       SE
## p = 0.25:  64.3577 21.22959
## p = 0.50: 108.4273 11.03200
## p = 0.75: 152.4970 19.24087
```

```
# Of the fish that moved, do larger fish move farther? Seems to be no difference
    ancest_data %>%
      filter(distance > 0) %>%
      ggplot(aes(Length, distance)) +
      geom_point() +
      geom_smooth(method = 'lm') +
      facet_wrap(~watershed, scales = 'free')
```




```
# How far did the fish who moved go?
    ancest_data %>%
      filter(distance > 0) %>%
      summarise(meanDist = mean(distance),
                medDist = median(distance))
```

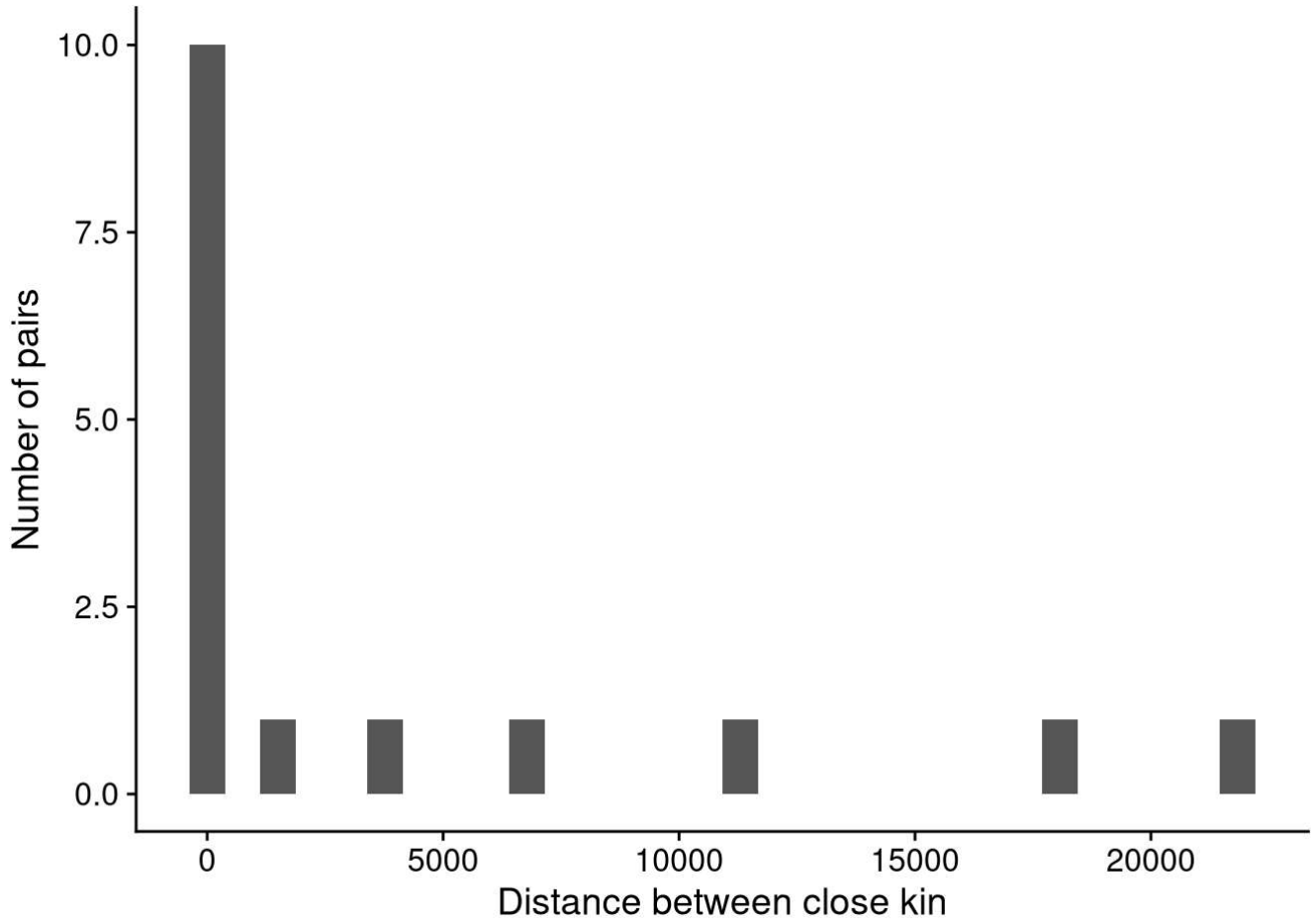
```
## # A tibble: 1 × 2
##   meanDist medDist
##      <dbl>   <dbl>
## 1     4.10    1.23
```

```
# Maximum distance
    max(ancest_data$distance)
```

```
## [1] 25.801
```

Compare the results to close-kin

```
# wrangle dataset of distance separating close-kin
  distance_between_kin<-kins_detected %>%
    mutate(pair = row_number()) %>%
    pivot_longer(-pair, names_to = 'xx', values_to = 'ind') %>%
    left_join(phenos) %>%
    dplyr::select(pair, xx, ind, pop) %>%
    pivot_wider(names_from = xx, values_from = c(ind, pop)) %>%
    rowwise() %>%
    mutate(pop12=paste0(sort(c(pop_D1,pop_D2)),collapse='/'))  %>%
    ungroup() %>%
    left_join(ibd_in, by = 'pop12') %>%
    mutate(distance = replace_na(distance, 0))

# plot histogram
  distance_between_kin %>%
    ggplot(aes(distance)) +
    geom_histogram() +
    theme_cowplot() +
    labs(x = 'Distance between close kin', y = 'Number of pairs')
```



```
  ggsave('/workdir/smallmouth/hudson/redbreast/results/close_kin_distance.png', height = 4, widt
h = 6)

# what percent of fish moved
  distance_between_kin %>%
    mutate(stayed_home = if_else(distance>1, 0, 1)) %>%
    summarise(mean(stayed_home))
```

```
## # A tibble: 1 × 1
##   `mean(stayed_home)`
##                 <dbl>
## 1               0.562
```

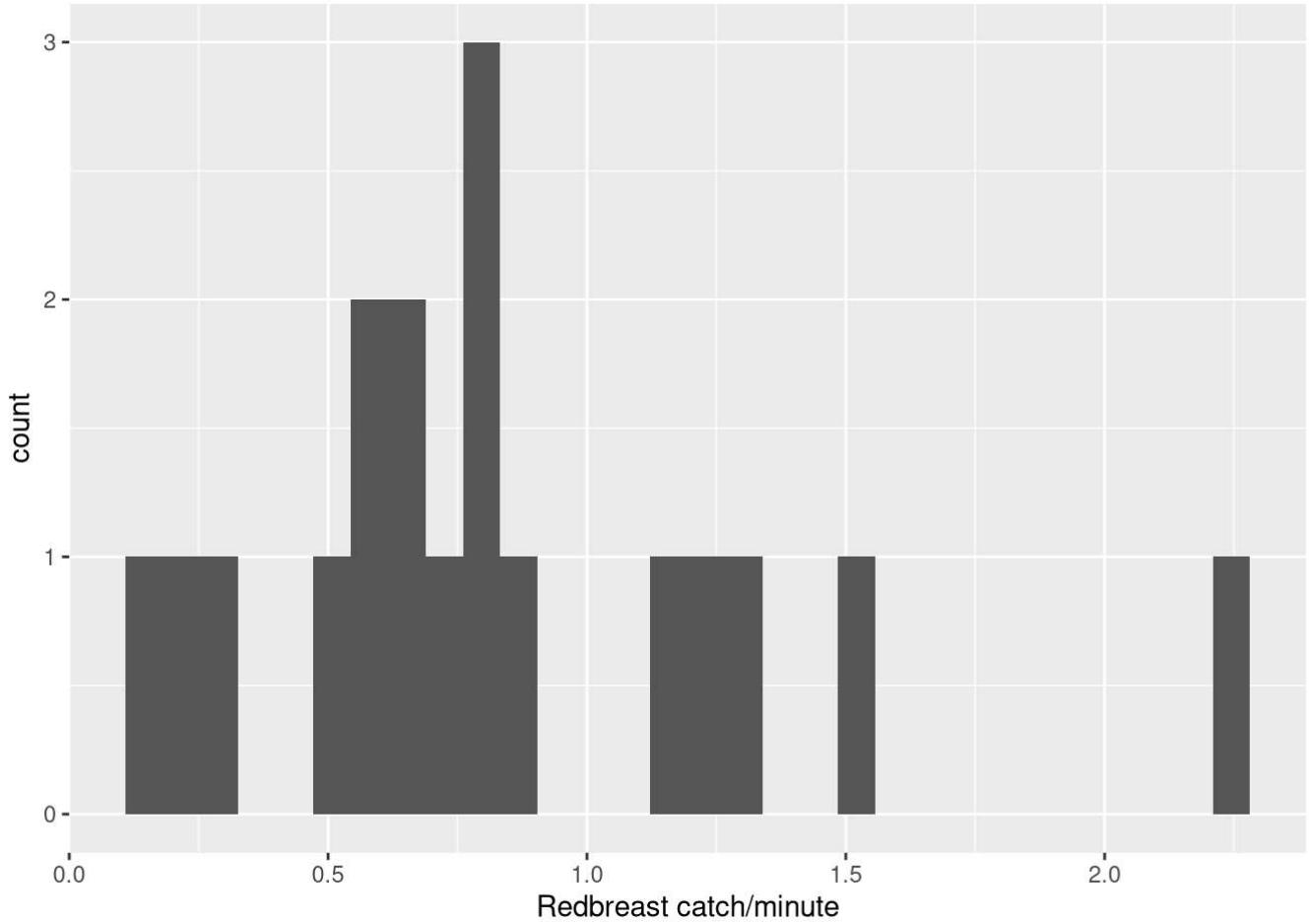
```
# What was the mean and median distance traveled by all fish, homebodies included?
  distance_between_kin %>%
    summarise(median_distance=median(distance),
              mean_distance=mean(distance))
```

```
## # A tibble: 1 × 2
##   median_distance mean_distance
##             <dbl>         <dbl>
## 1               0         3938.
```

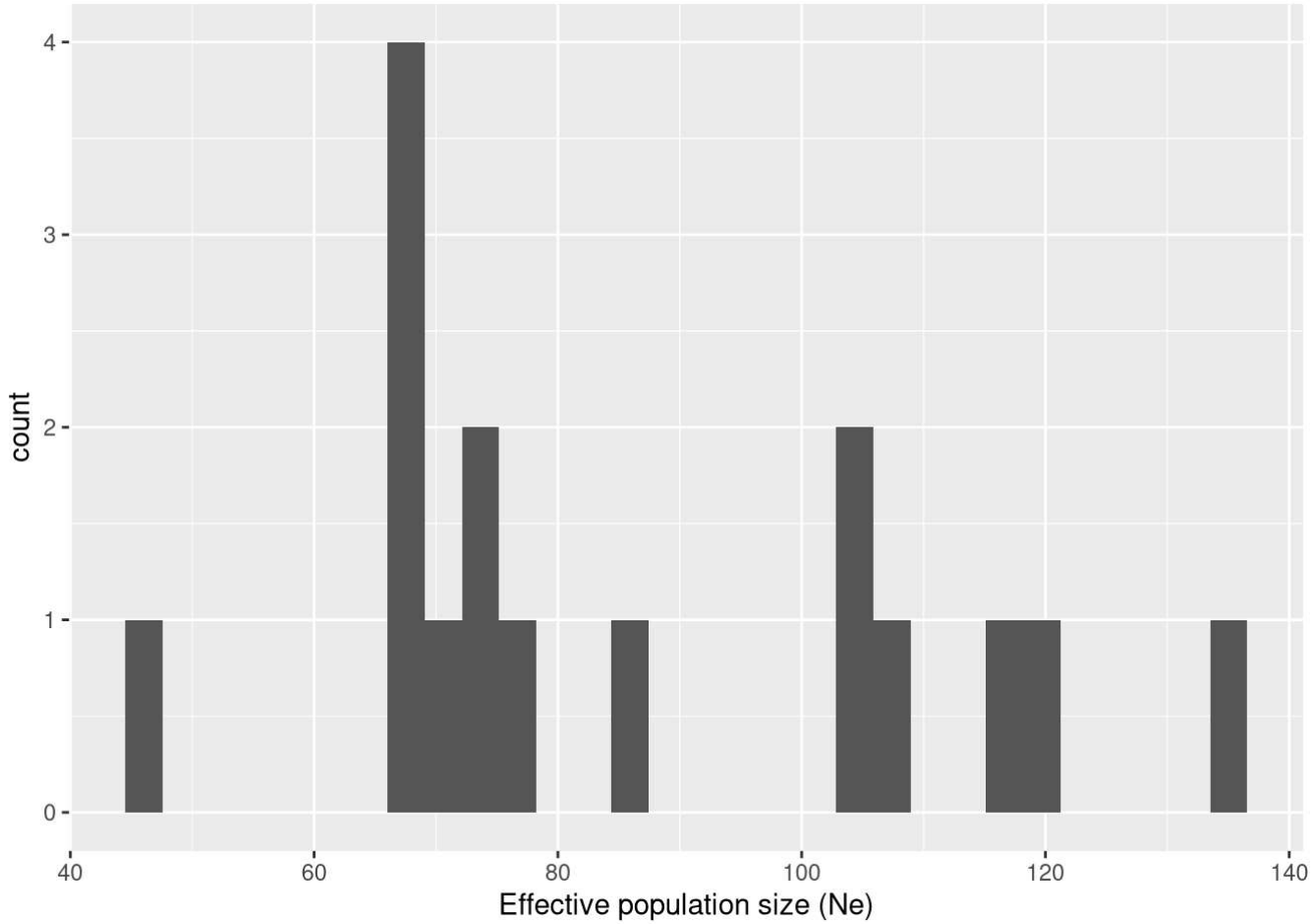
prediction 2 (Figure 5): density and Ne decreases in upper reaches of watersheds

Data are normally distributed, so I can use linear model Here, also test for spatial autocorrelation

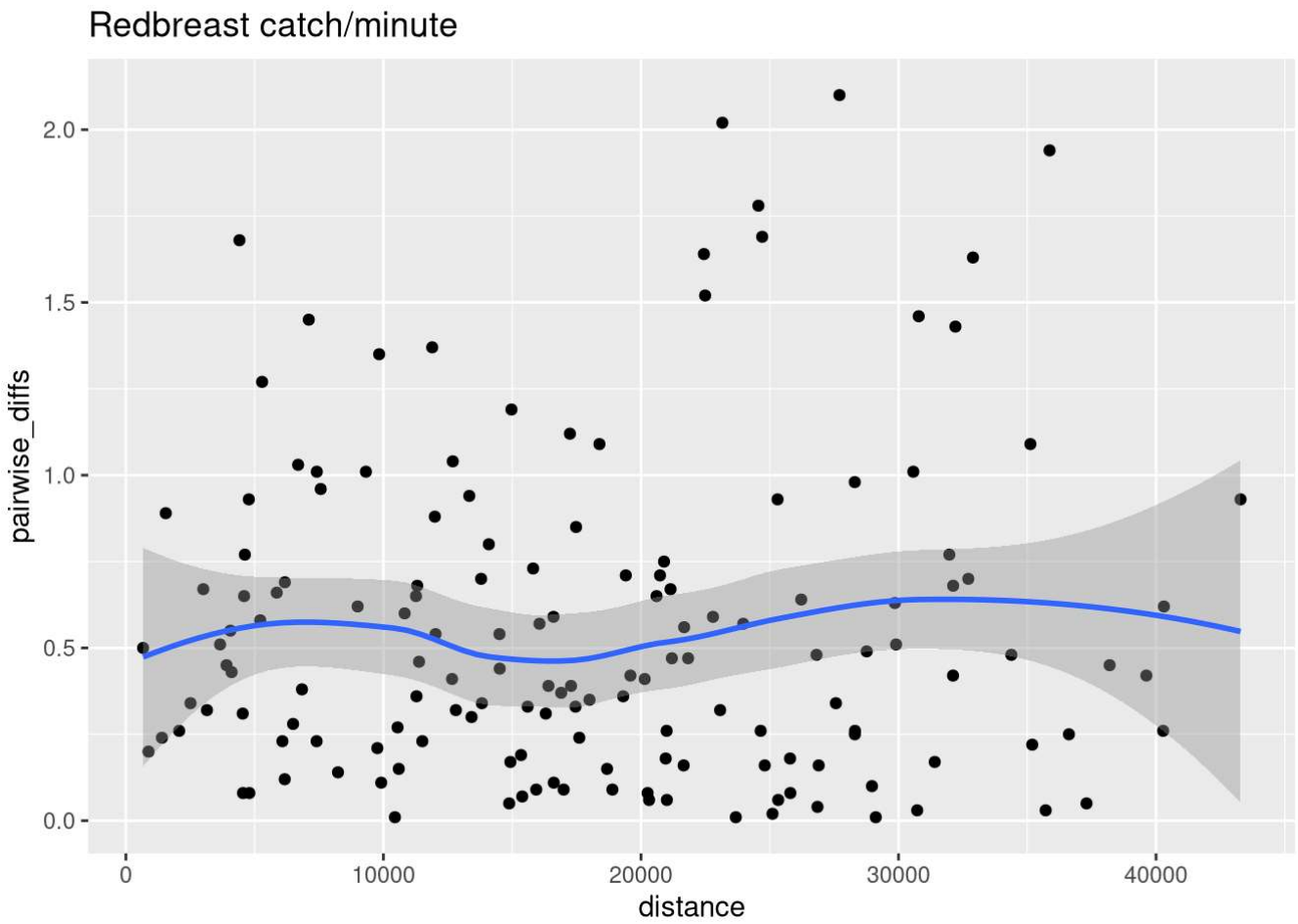
```
pre_stats_box<-read_csv('/workdir/smallmouth/hudson/redbreast/results/summary_stats_per_pop.cs
v') %>%
  separate(`Effective population size (Ne)`, into = c('Ne','sd'), sep = ' \\(') %>%
  mutate(Ne = as.double(Ne)) %>%
  dplyr::select(c(Site, `Sub-basin`, `River order`, `Redbreast catch/minute`, `Effective populat
ion size (Ne)`=Ne)) %>%
  filter(`Sub-basin` != 'Furnace')

# Test for normality - not too bad
  pre_stats_box %>%
    ggplot(aes(`Redbreast catch/minute`)) +
    geom_histogram()
```



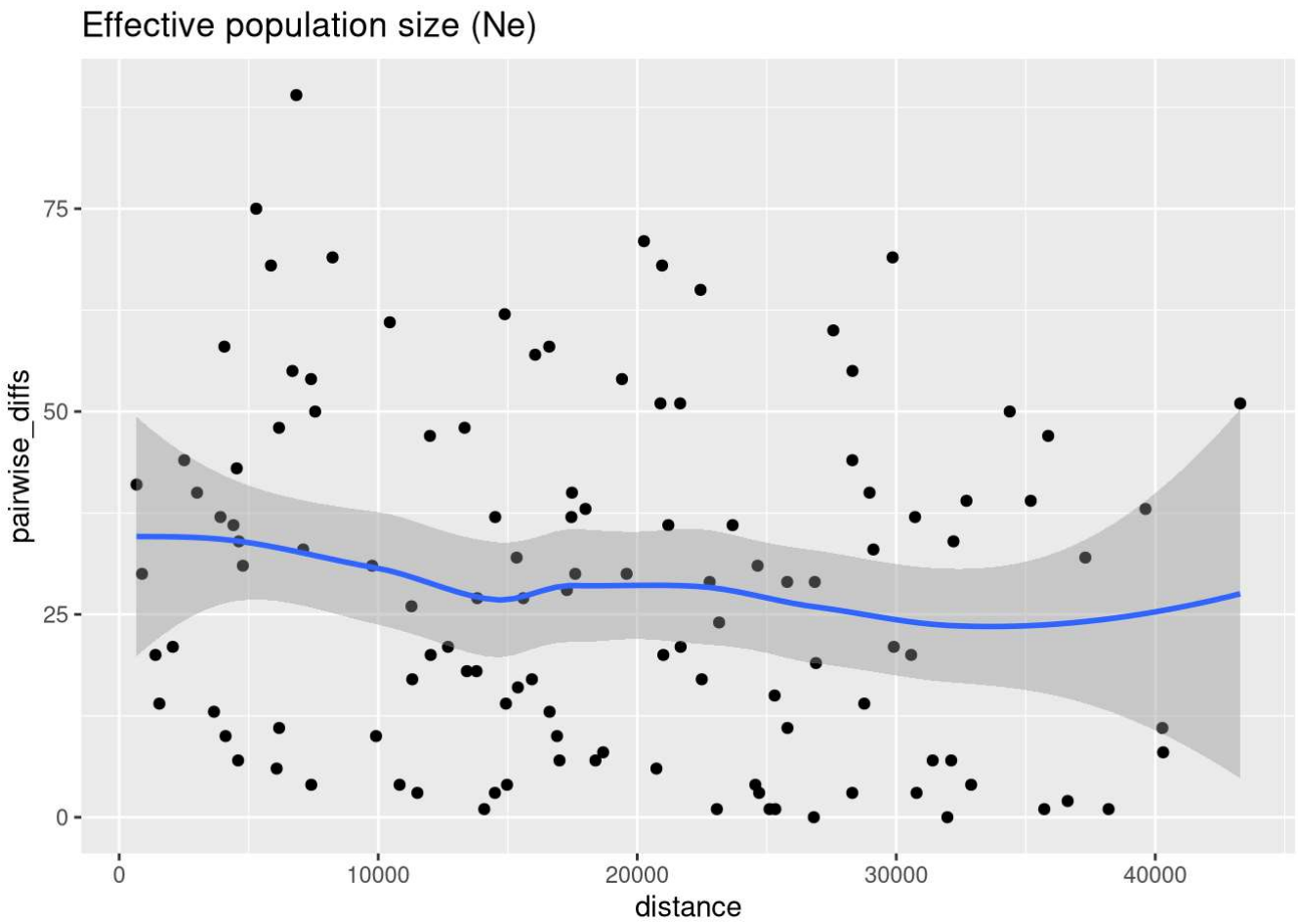

```
pre_stats_box %>%
  ggplot(aes(`Effective population size (Ne)`)) +
  geom_histogram()
```


count
4
3
2
1
0
40
60
80
100
120
140
Effective population size (Ne)

```
# Test for SA
  # First clean up the distance measurements
    ibd_sa<-ibd_in %>%
      rowwise() %>%
      mutate(ind12=paste0(sort(c(pop1_site,pop2_site)),collapse='---'))  %>%
      ungroup() %>%
      dplyr::select(ind12, distance)

  # Generate pairwise differences in measurements and test for correlation
    pval_out<-tibble()
    for(test_in in c('Redbreast catch/minute', 'Effective population size (Ne)')){
      # generate pairwise differences
      diffs_in<-pre_stats_box %>%
        dplyr::select(ind=Site, test_in) %>%
        diffs('pairwise_diffs') %>%
        full_join(ibd_sa, by ='ind12') %>%
        filter(!is.na(distance), !is.na(pairwise_diffs))

      # Export a plot
      print(ggplot(diffs_in, aes(distance, pairwise_diffs)) +
        geom_point() +
        geom_smooth() +
        ggtitle(test_in))

      # Test sequentially increasing distances
      for(dist_step in seq(0,20000,1000)){
        diffs_out<-diffs_in %>%
          filter(distance < dist_step) %>%
          separate(ind12, into = c('pop1','pop2'), sep = '---') %>%
          mutate(river = 'Moodna') # add river for all, lme corMLPE doesn't work wo/random effec
t

        try({
        gg<-summary(lme(pairwise_diffs ~ distance,
                        random=~1|river, # for some reason it doesn't work without a random effe
ct
                        correlation=corMLPE(form=~pop1+pop2|river),
                        data=  diffs_out,
                        method="REML"))

        pval_out<-bind_rows(pval_out, tibble(test = test_in,
                                             dist_km = dist_step/1000,
                                             p = round(gg$tTable[2,5],2)))
        })
      }
    }
```




```
## Error in h(simpleError(msg, call)) :
##   error in evaluating the argument 'object' in selecting a method for function 'summary': len
gth of 'dimnames' [2] not equal to array extent
## Error in h(simpleError(msg, call)) :
##   error in evaluating the argument 'object' in selecting a method for function 'summary': the
leading minor of order 1 is not positive definite
```




```
## Error in h(simpleError(msg, call)) :
##   error in evaluating the argument 'object' in selecting a method for function 'summary': len
gth of 'dimnames' [2] not equal to array extent
## Error in h(simpleError(msg, call)) :
##   error in evaluating the argument 'object' in selecting a method for function 'summary': the
leading minor of order 1 is not positive definite
```

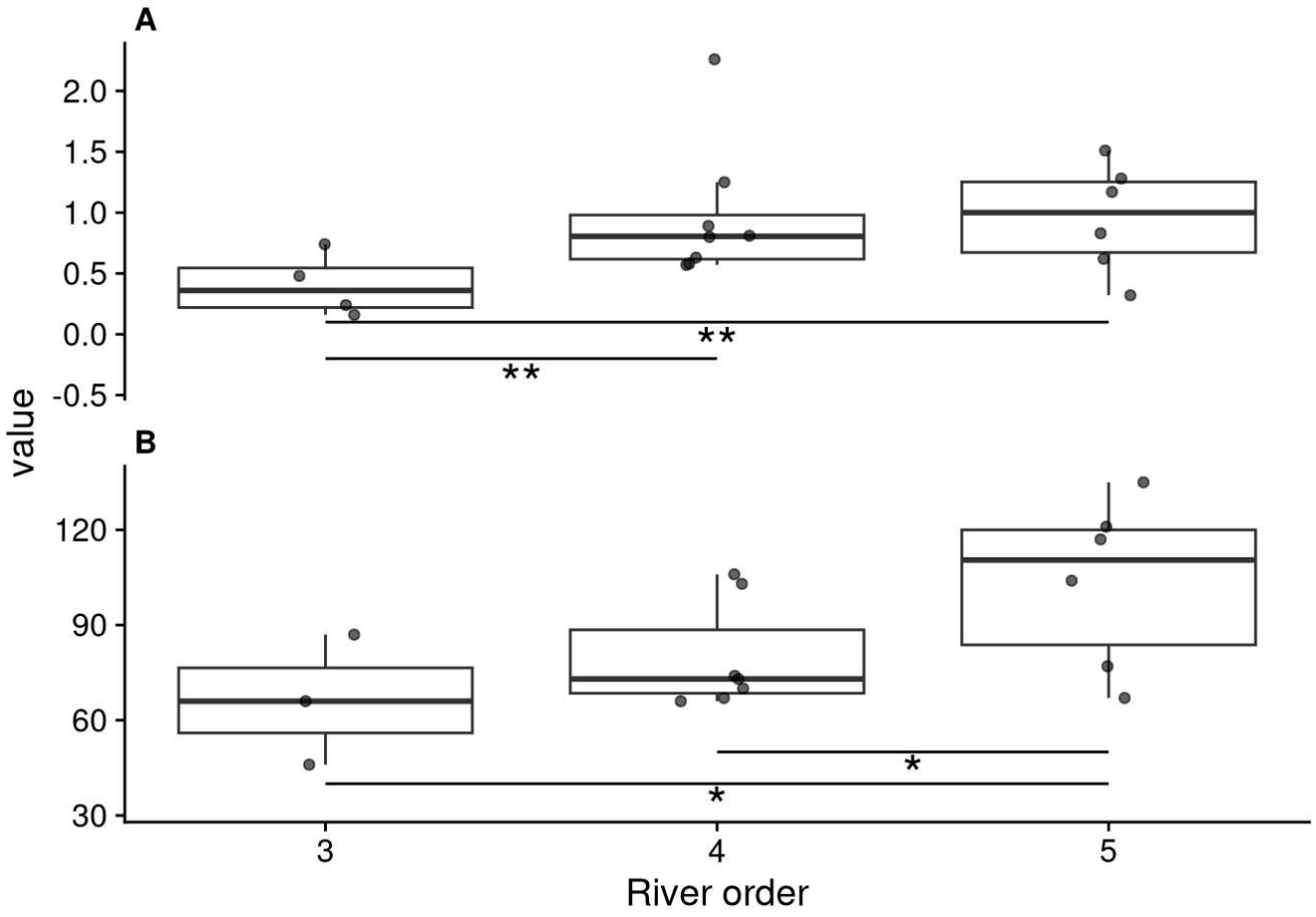
```
  # we only get a moderately significant p-value at 9km for Ne.
    pval_out %>%
      filter(p<0.1)
```

```
## # A tibble: 3 × 3
##   test                           dist_km     p
##   <chr>                            <dbl> <dbl>
## 1 Effective population size (Ne)       2  0.01
## 2 Effective population size (Ne)       7  0.09
## 3 Effective population size (Ne)       9  0.07
```

```
# Export results
  stats_box<-pre_stats_box %>%
    pivot_longer(-c(Site, `Sub-basin`, `River order`) )

  stats_box %>%
    mutate(re = as.factor(`River order`)) %>%
    group_by(name) %>%
    rstatix::t_test(value ~ re) %>%
    filter(p<=0.1) %>%
    dplyr::select(name, group1, group2, p)
```

```
## # A tibble: 4 × 4
##   name                           group1 group2     p
##   <chr>                          <chr>  <chr>  <dbl>
## 1 Effective population size (Ne) 3      5      0.066
## 2 Effective population size (Ne) 4      5      0.096
## 3 Redbreast catch/minute         3      4      0.038
## 4 Redbreast catch/minute         3      5      0.04
```

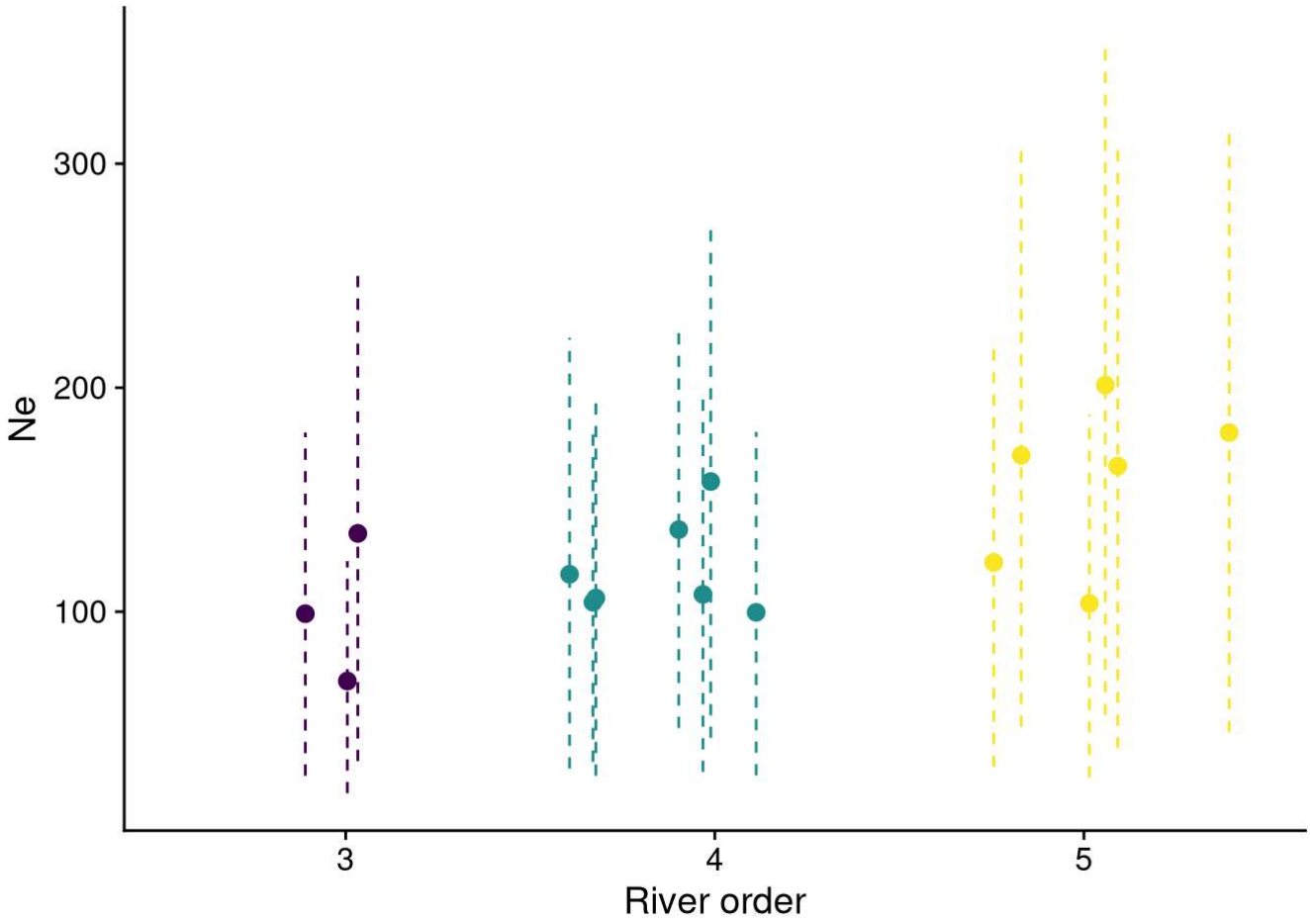
```
  stats_box %>%
    bind_rows(tibble(`River order` = c(3, 3, 4, 3),
                     name = c('Redbreast catch/minute', 'Effective population size (Ne)', 'Effec
tive population size (Ne)','Redbreast catch/minute'),
                     yval = c(.1,40, 50, -.2),
                     size = 20,
                     ytext = c(-.1,33, 43, -.4),
                     xtext = c(4, 4, 4.5, 3.5),
                     xend = c(5,5, 5, 4),
                     ylab = c('**', '*', '*', '**'))) %>% # had to fine-tune this to get it in t
he right place
    mutate(name_short=if_else(str_detect(name, 'Effective'), 'B', 'A')) %>%
    ggplot(aes(`River order`, value, group = `River order`)) +
    geom_boxplot(outlier.shape = NA) +
    geom_jitter(height = 0, width = 0.1, alpha = 0.6) +
    facet_wrap(~name_short, ncol=1, scale = 'free_y') +
    theme_cowplot() +
    scale_x_continuous(breaks = 1:5) +
    geom_segment(aes(y = yval, xend = xend, yend = yval)) +
    geom_text(aes(x = xtext, y = ytext, label = ylab), size = 7) +
    theme(strip.background = element_blank(),
      strip.text = element_text(hjust=0,
                                face='bold'))
```


A
2.0
1.5
1.0
0.5
0.0
-0.5
**
**
B
120
90
60
30
*
*
value
3
4
5
River order

```
  ggsave('/workdir/smallmouth/hudson/redbreast/results/basic_stats_by_river_order.png', height =
6, width = 5)

# Also make a plot with 95% confidence intervals on Ne
  Ne_data<-read_csv('/workdir/smallmouth/hudson/redbreast/results/vareff_output_pops_filtered.cs
v') %>%
    filter(generations==0, !str_detect(pop, 'Furnace')) %>%
    dplyr::select(pop, river_order, Ne = Ne_arithmetic_mean, Ne_5 = Ne_5th_quantile, Ne_95 = Ne_
95th_quantile) %>%
    mutate(width_90 = Ne_95-Ne_5,
           standard_error = width_90/3.29,
           variance = standard_error^2)

  # use weighted regression, with 1 / variance as the weight for each point (inverse variance we
ighting)
    # It turns out that lm doesn't do the weighting in the best way for this test: https://www.m
etafor-project.org/doku.php/tips:rma_vs_lm_lme_lmer
    # summary(lm(Ne ~ river_order, data = Ne_data, weights = 1/variance))

  # Run this with metafor, for a more robust test
      m<-metafor::rma(yi=Ne, # dependent
                  mods=river_order, # independent
                  data = Ne_data,
                  method = 'FE', # fixed-effects: IE no random effects
                  sei = standard_error) # include the standard error of each point
      summary(m)
```

```
##
## Fixed-Effects with Moderators Model (k = 16)
##
##   logLik  deviance       AIC       BIC      AICc
## -81.2772    3.2391  166.5544  168.0996  167.4775
##
## I^2 (residual heterogeneity / unaccounted variability): 0.00%
## H^2 (unaccounted variability / sampling variability):   0.23
## R^2 (amount of heterogeneity accounted for):            36.79%
##
## Test for Residual Heterogeneity:
## QE(df = 14) = 3.2391, p-val = 0.9986
##
## Test of Moderators (coefficient 2):
## QM(df = 1) = 2.2517, p-val = 0.1335
##
## Model Results:
##
##          estimate       se    zval    pval      ci.lb     ci.ub
## intrcpt    3.5630  73.4511  0.0485  0.9613  -140.3985  147.5245
## mods      27.6034  18.3954  1.5006  0.1335    -8.4510   63.6578
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Plot
  Ne_data %>%
  ggplot(aes(as.character(river_order), Ne, color = as.character(river_order))) +
    geom_pointrange(aes(ymin = Ne_5, ymax = Ne_95),
                    position=position_jitter(width=0.4),
                    lty = 'dashed') +
    theme_cowplot() +
    labs(x = 'River order', y = 'Ne') +
    scale_color_viridis_d() +
    theme(legend.position = 'none')
```

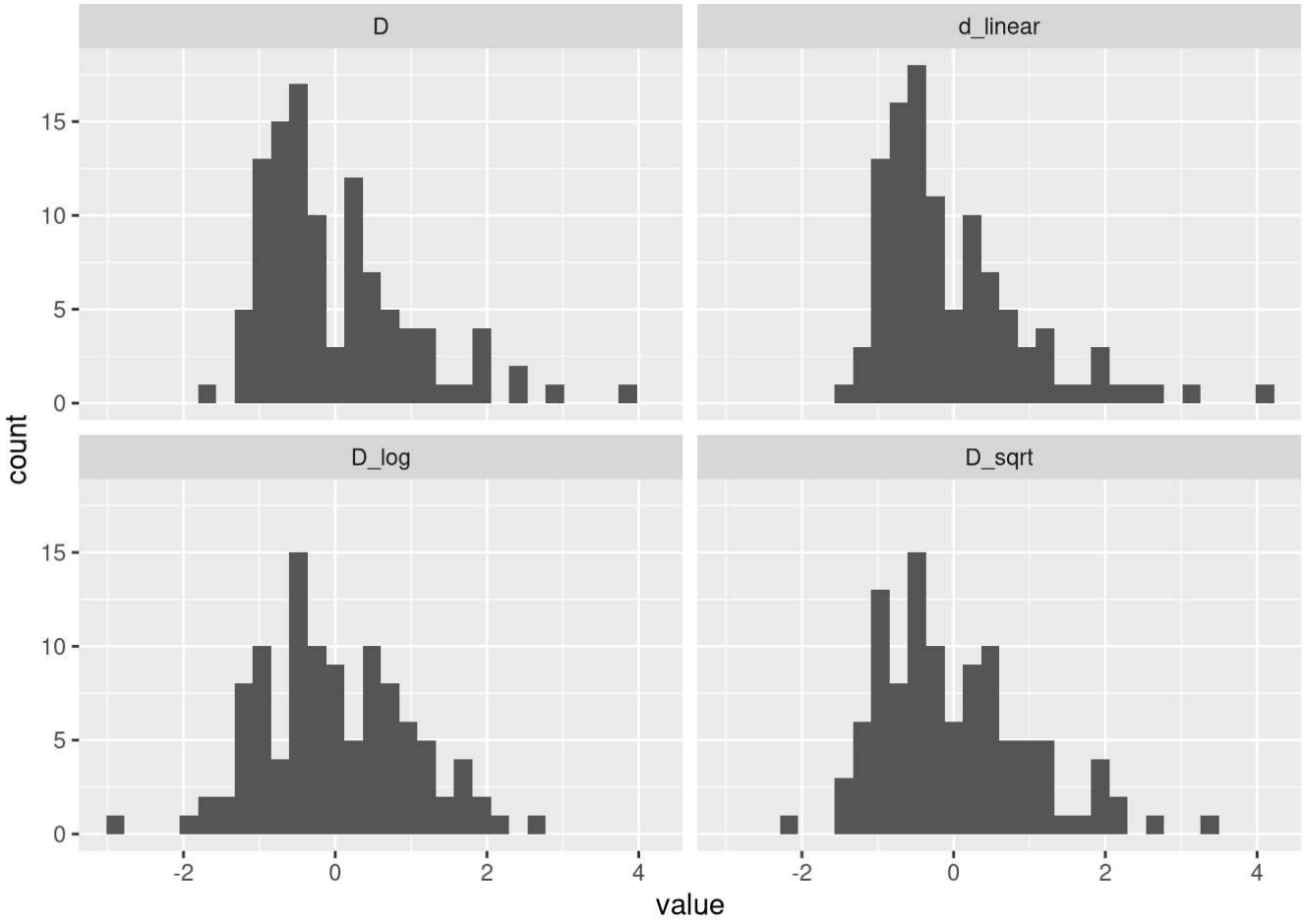



```
    ggsave('/workdir/smallmouth/hudson/redbreast/results/Ne_confidence_intervals.svg', height =
4, width = 6)
```

prediction 3 (Table 3/4): dams in upper tribs show greater genetic differentiation

Detrended mixed effects modeling approach to test for impact of dams and river order , suggested by Ryan from Jaffe 2019

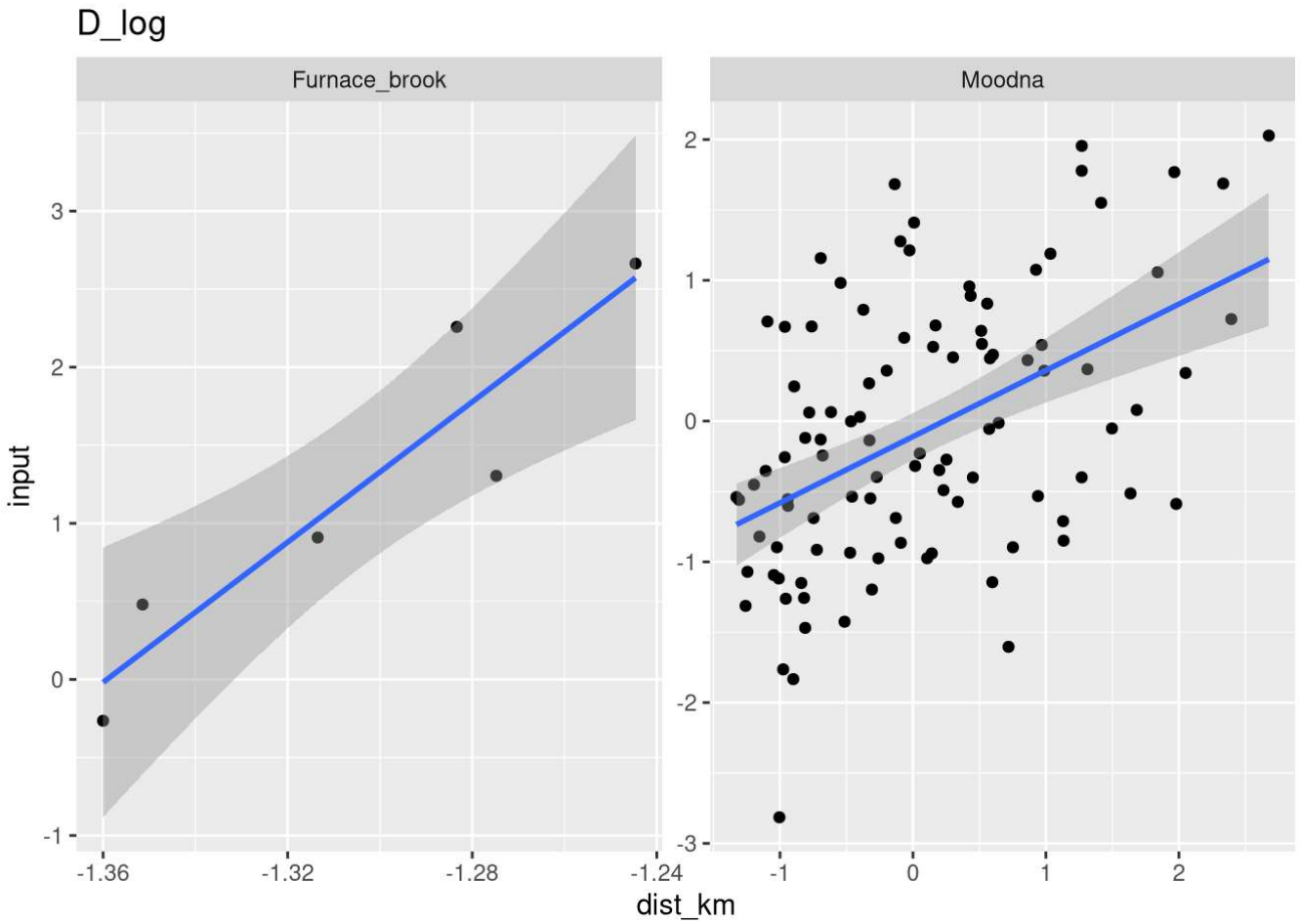
```
# Build dataset
  reml_in<-ibd_in %>%
    filter(pop12 %in% upstream_downstream_by_basin$Pop12_name) %>% # limit to just within basin,
and connection
    mutate(D_sqrt = sqrt(D+0.03),# +0.03 bcs that brings the most negative to (+), then sqrt for
right-skew
           D_log = log(D+0.03)) %>%
    separate(pop12, into = c('pop1','pop2'), sep = '/', remove = F) %>%
    mutate(average_river_order = (river_distance_3*3+river_distance_4*4+river_distance_5*5)/(dis
t_km*1000),
           average_river_order = if_else(average_river_order < 3 & str_detect(pop2, '-Moodna-|up
perMoodna'), 5,
                                         if_else(average_river_order < 3 & str_detect(pop2, 'Satterly'),
3, average_river_order)), # Have to do a correction here because some 5th order streams get igno
red
           across(where(is.numeric), scale)) %>%  # scale all numeric values
    dplyr::select(pop12, river=pop1_river, D, contains('D_'), dist_km, average_river_order, n_in
tact, n_intact_order3, n_intact_order4, n_intact_order5, max_gen_impounded, max_height, alt_chan
ge, pop1, pop2)

# Test for normality in different genetic differentiation correction methods
  normal_test<-reml_in %>%
    dplyr::select(pop12, D, d_linear, contains('D_')) %>%
    pivot_longer(-pop12)

  normal_test %>%
    ggplot(aes(value)) +
    geom_histogram() +
    facet_wrap(~name)
```



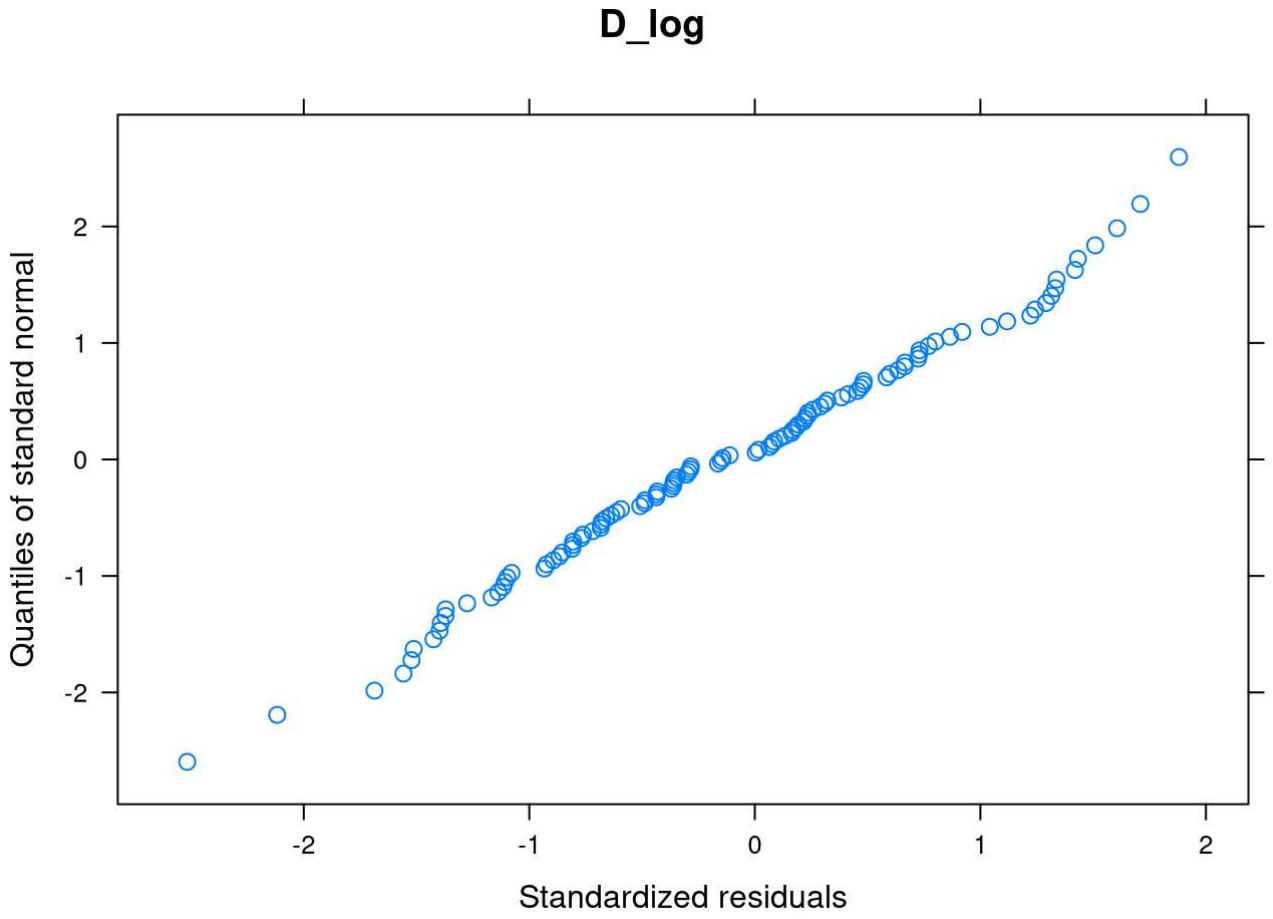

```
normal_test %>%
  group_by(name) %>%
  rstatix::shapiro_test(value)
```

```
## # A tibble: 4 × 4
##   name     variable statistic           p
##   <chr>    <chr>        <dbl>       <dbl>
## 1 D        value        0.899 0.000000708
## 2 D_log    value        0.989 0.543
## 3 D_sqrt   value        0.958 0.00218
## 4 d_linear value        0.880 0.0000000959
```

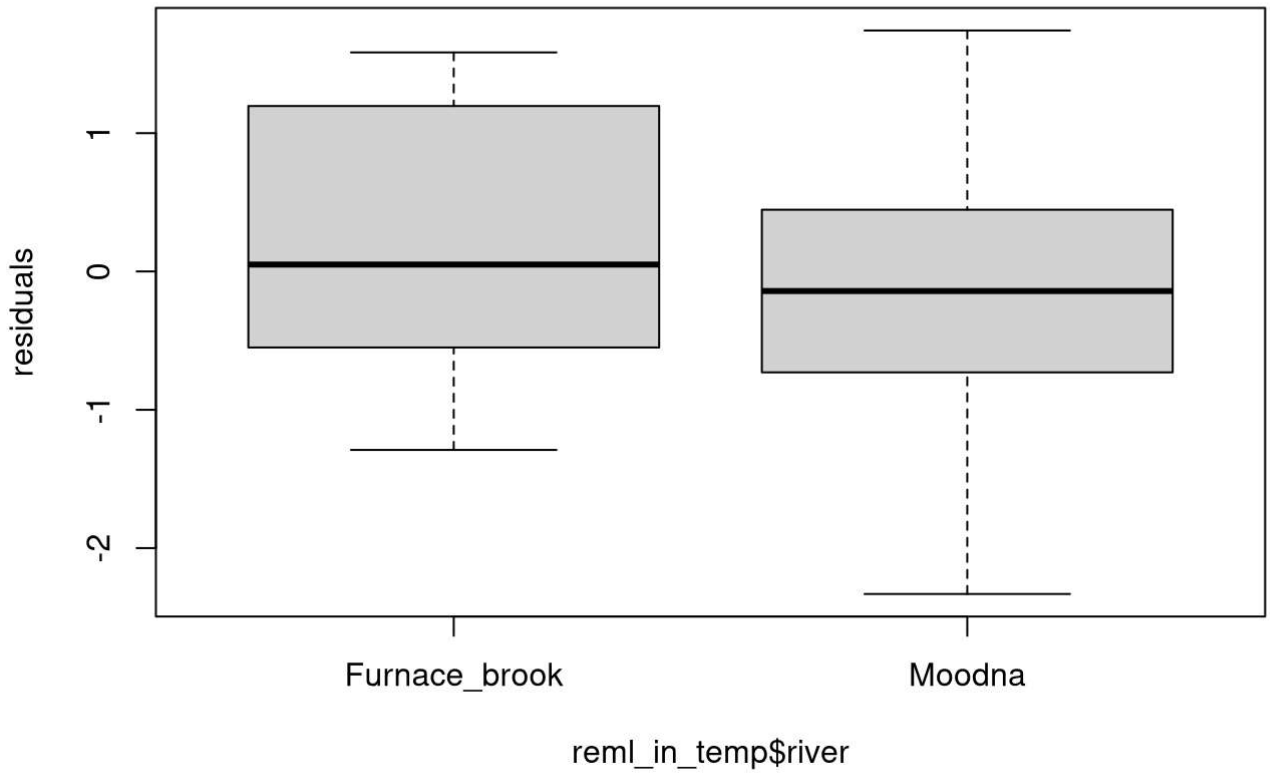
```
# Detrend all fixed effects to account for the contaminating effect of distance
  # Use the older optimizer for lme (see https://stats.stackexchange.com/questions/40647/lme-err
or-iteration-limit-reached)
    ctrl <- lmeControl(opt='optim', msMaxIter=100)

  # Detrend each fixed effect
    detrend_summary<-tibble()
    detrend_resids<-dplyr::select(reml_in, pop12, pop1, pop2, river)
    for(fixed in c('average_river_order','alt_change','n_intact','max_gen_impounded','max_heigh
t','n_intact_order3','n_intact_order4','n_intact_order5','D_log')){

      #fixed<-'D_trans'

      reml_in_temp<-reml_in %>%
        dplyr::rename(input=fixed)

      model_resid<-lme(input ~ dist_km,
                random=~dist_km|river, # ~dist_km|river would be random slope and intercept, ~1|
river is random intercept
                correlation=corMLPE(form=~pop1+pop2|river),
                data=  reml_in_temp,
                method="REML",
                control = ctrl)

      detrend_summary<-tibble(fixed = fixed) %>%
        bind_cols(as_tibble(summary(model_resid)$tTable)[2,]) %>%
        bind_rows(detrend_summary)

      detrend_resids<-tibble(!!fixed := residuals(model_resid)) %>% # dynamically set the column
name
        bind_cols(detrend_resids)
    }

    # Purposefully did genetic differentiation last so that I could show residuals
      plot(ggplot(reml_in_temp, aes(dist_km, input)) +
             geom_point() +
             geom_smooth(method = 'lm') +
             facet_wrap(~river, scales = 'free') +
             ggtitle(fixed))
```




```
plot(qqnorm(model_resid, main = fixed))
```




```
boxplot(residuals(model_resid)~reml_in_temp$river, ylab = "residuals")
```




```
    # All fixed effects are contaminated by distance except for order 3 dams and average river o
rder
      detrend_summary %>%
        mutate(sig = if_else(`p-value` < 0.05, 1,0))
```

```
## # A tibble: 9 × 7
##   fixed                Value Std.Error    DF `t-value` `p-value`    sig
##   <chr>                <dbl>     <dbl> <dbl>     <dbl>     <dbl> <dbl>
## 1 D_log              0.483      0.0966   103    5.00    2.41e- 6     1
## 2 n_intact_order5    0.363      0.0529   103    6.86    5.28e-10     1
## 3 n_intact_order4    0.859      0.0617   103   13.9     2.08e-25     1
## 4 n_intact_order3    0.00583    0.0571   103    0.102   9.19e- 1     0
## 5 max_height         0.860      0.0462   103   18.6     1.06e-34     1
## 6 max_gen_impounded  0.999      0.0359   103   27.8     1.08e-49     1
## 7 n_intact           0.909      0.0431   103   21.1     3.45e-39     1
## 8 alt_change         0.604      0.167    103    3.61    4.69e- 4     1
## 9 average_river_order 0.00769   0.469    103    0.0164  9.87e- 1     0
```

```
# Now make the full model to test fixed effects
  model_full<-lme(D_log ~ average_river_order + alt_change + n_intact + max_gen_impounded + max_
height + n_intact_order3 + n_intact_order4 + n_intact_order5,
        random=~1|river,
        correlation=corMLPE(form=~pop1+pop2|river),
        data=detrend_resids,
        method="REML",
        control = ctrl)

  car::vif(model_full)
```

```
## average_river_order          alt_change            n_intact   max_gen_impounded
##            1.216135            1.334997          468.806759            1.535929
##          max_height     n_intact_order3     n_intact_order4     n_intact_order5
##            7.286749           60.125751          604.187324          265.968246
```

```
  Null <- update(model_full, ~ 1, method = 'ML')
  `Natural: River order` <- update(model_full, ~ average_river_order, method = 'ML')
  `Natural: Altitude change` <- update(model_full, ~ alt_change, method = 'ML')
  `Dams: Count` <- update(model_full, ~ n_intact, method = 'ML')
  `Dams: Count and maximum generations of impoundment` <- update(model_full, ~ n_intact + max_ge
n_impounded, method = 'ML')
  `Dams: Count and maximum height` <- update(model_full, ~ n_intact + max_height, method = 'ML')
  `Dams: Count on river order 3, 4, and 5` <- update(model_full, ~ n_intact_order3 + n_intact_or
der4 + n_intact_order5, method = 'ML')

  model.sel(Null,
            `Natural: River order`,
            `Natural: Altitude change`,
            `Dams: Count`,
            `Dams: Count and maximum generations of impoundment`,
            `Dams: Count and maximum height`,
            `Dams: Count on river order 3, 4, and 5`) %>%
    as.data.frame() %>%
    rownames_to_column(var = 'Model') %>%
    dplyr::select(-c(2:10)) %>%
    mutate(across(where(is.numeric), round, 1)) %>%
    write_tsv('/workdir/smallmouth/hudson/redbreast/results/aic_model_selection_table.tsv',col_n
ames = T)

  summary(`Dams: Count on river order 3, 4, and 5`)$tTable  %>%
    as.data.frame() %>%
    rownames_to_column(var = 'Parameter') %>%
    mutate(across(where(is.numeric), round, 2))  %>%
    write_tsv('/workdir/smallmouth/hudson/redbreast/results/top_model_parameters.tsv')
```

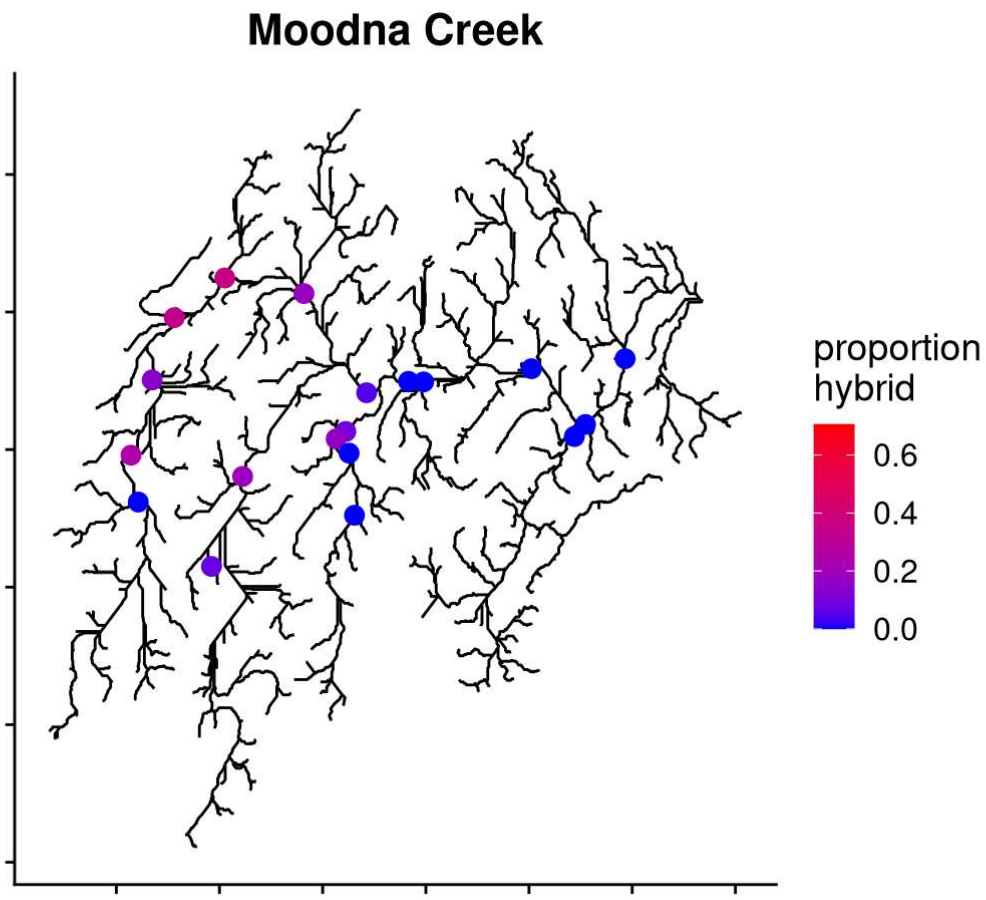
Supplementary information

Export locus tables (incude average read depth)

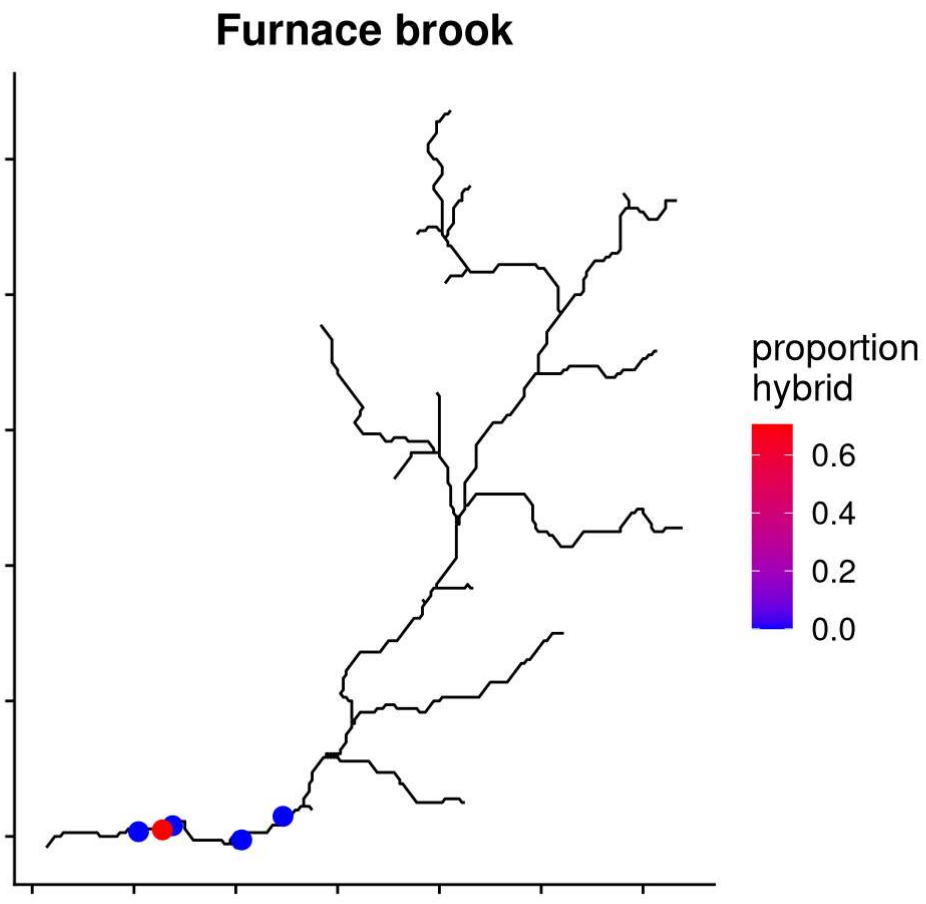
```
# export important tables
    locus_table<-rownames_to_column(as.data.frame(locus_table(obj)), var = 'locus') %>%
      arrange(locus) %>%
      rename(Simpsons = `1-D`) %>%
      mutate(allele = as.integer(allele),
             Simpsons = round(Simpsons,2),
             Hexp = round(Hexp,2),
             Evenness = round(Evenness, 2))

    # generate average read depth per locus
        ind_list<-rownames_to_column(genind2df(obj), var = 'ind') # create a list of inds that p
assed all filters

        read_tsv('/workdir/smallmouth/hudson/redbreast/multiplex_sequence/amplicon_py/amp_py_out
put/hap_genotype_readDepth') %>%
          separate(ind, into = c('ind', 'plate'), sep = '-plate') %>%
          filter(ind %in% ind_list$ind,
                 Locus %in% locus_table$locus) %>%
          group_by(locus = Locus) %>%
          summarise(mean_depth = round(mean(read_depth),0)) %>%
          left_join(locus_table, by = 'locus') %>%
          write_csv('/workdir/smallmouth/hudson/redbreast/results/locus_table.csv')  # Don't thi
nk I need to exclude anything
```

Where do we find hybrids?

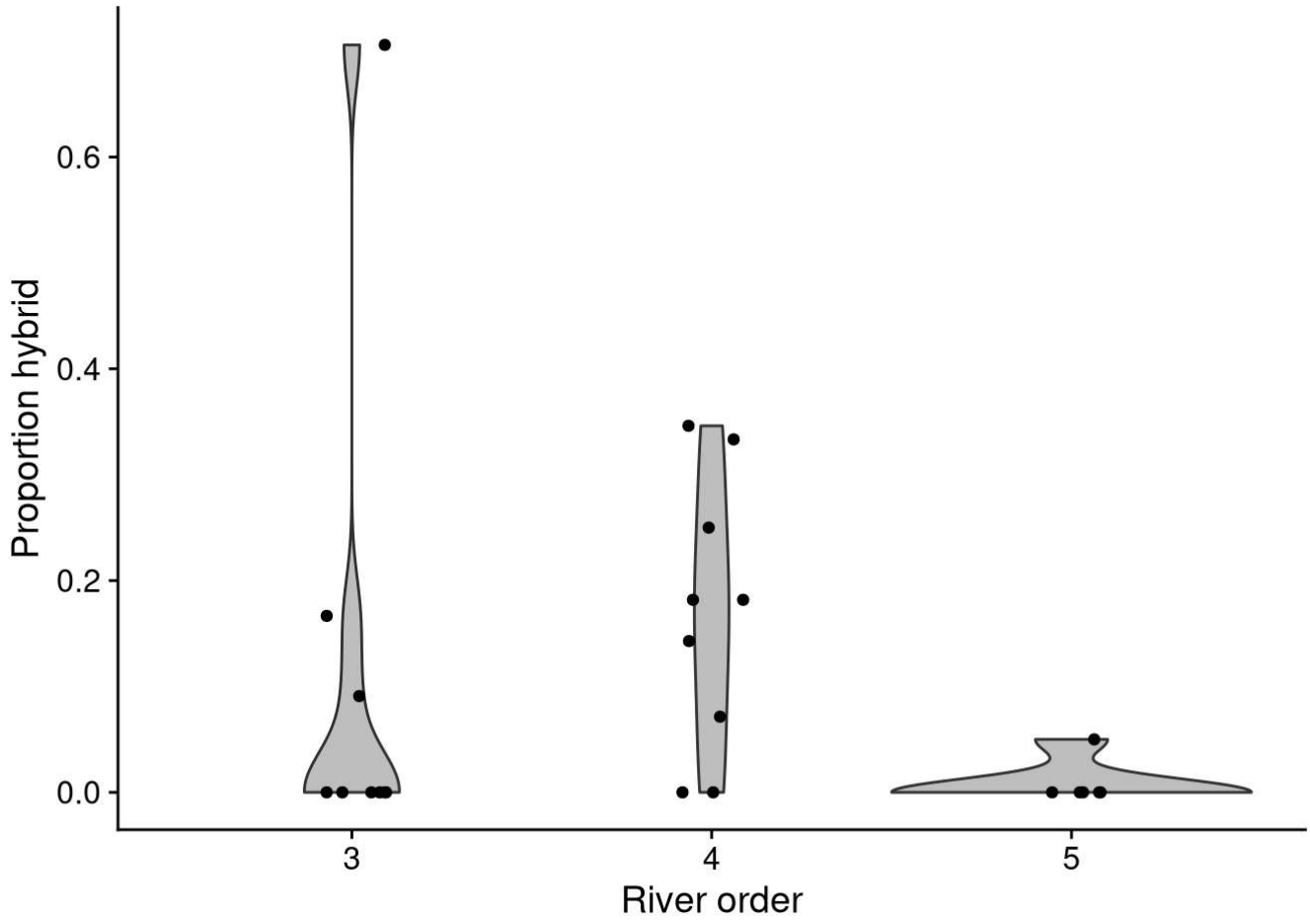
```
input<-pre_genos %>%
        left_join(read_csv('/workdir/smallmouth/hudson/redbreast/results/hybrid_results.csv'), b
y = 'ind') %>%
  left_join(phenos_all_fish, by = 'ind') %>%
  group_by(pop, lat, lon, pop_final, river_order) %>%
  tally() %>%
  ungroup() %>%
  filter(str_detect(pop_final, 'redbreast|hybrid')) %>%
  pivot_wider(names_from=pop_final, values_from=n) %>%
  mutate(prop_hybrid = if_else(is.na(hybrid), 0, hybrid/(hybrid+redbreast))) %>%
  separate(pop, into = c('River', 'Transect'), sep = '-')

  ggplot() +
    geom_sf(data = moodna_map) +
    geom_point(data = filter(input, River == 'Moodna'),
               aes(lon, lat, color = prop_hybrid), size = 3)  +
    scale_color_gradient(low = 'blue',
                         high = 'red',
                         limits = c(0,max(input$prop_hybrid))) +
    theme_cowplot() +
    labs(x='',y='', color='proportion\nhybrid') +
    theme(axis.text.x = element_blank(),
          axis.text.y = element_blank(),
          plot.title = element_text(hjust = 0.5)) +
    ggtitle('Moodna Creek')
```




```
ggplot() +
  geom_sf(data = fb_map) +
  geom_point(data = filter(input, River == 'Furnace_brook'),
             aes(lon, lat, color = prop_hybrid), size = 3)  +
  scale_color_gradient(low = 'blue',
                       high = 'red',
                       limits = c(0,max(input$prop_hybrid))) +
  theme_cowplot() +
  labs(x='',y='', color='proportion\nhybrid') +
  theme(axis.text.x = element_blank(),
        axis.text.y = element_blank(),
        plot.title = element_text(hjust = 0.5)) +
  ggtitle('Furnace brook')
```




```
# proportion hybrid per populatoin
  input %>%
    mutate(order = replace_na(river_order, 3)) %>%
    ggplot(aes(order, prop_hybrid, group = order)) +
    geom_violin(width = 1,fill = 'grey') +
    geom_jitter(height = 0, width = 0.1) +
    theme_cowplot() +
    labs(x = 'River order', y = 'Proportion hybrid')
```



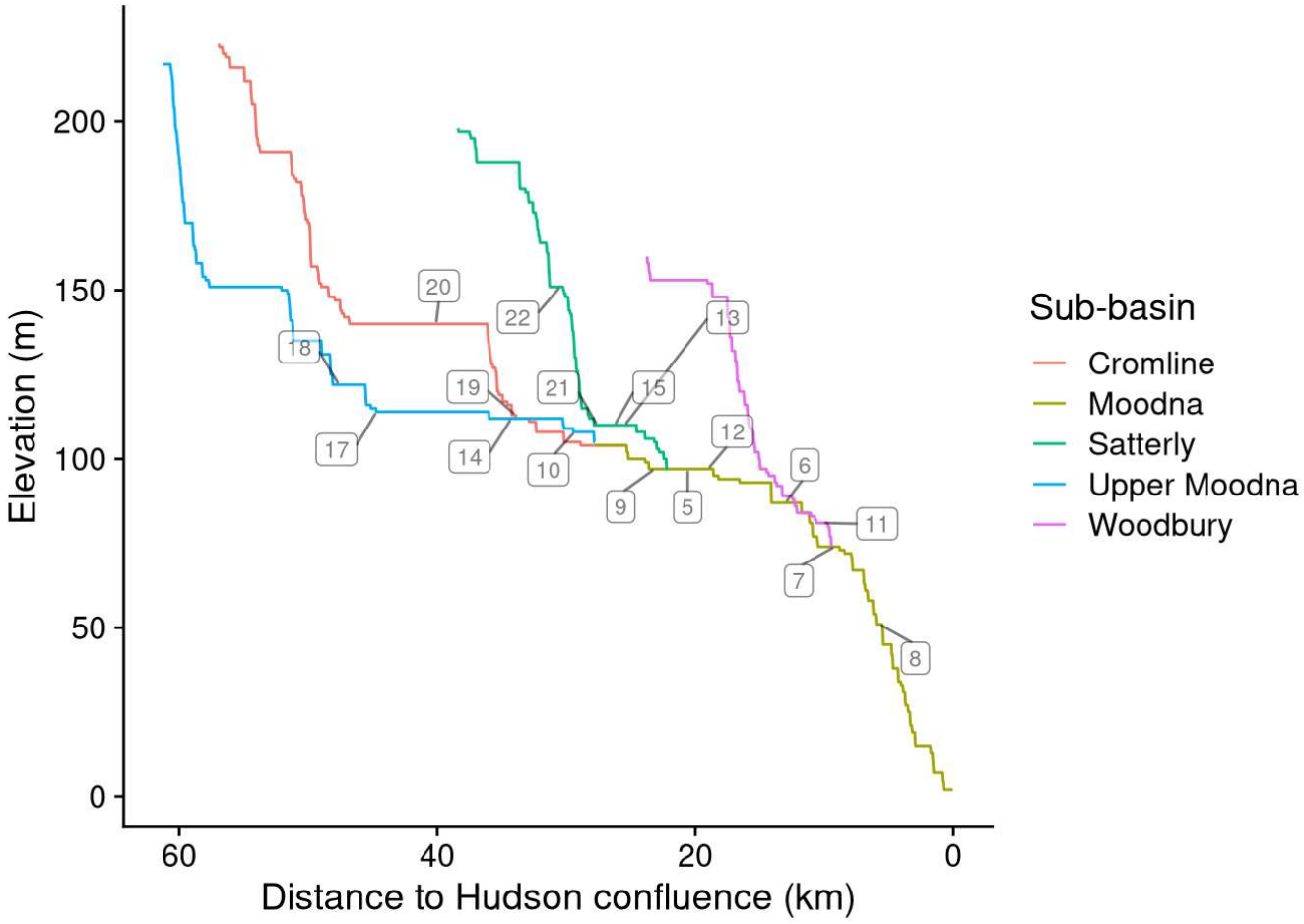

```
# statistical test - binomial
  hyb_test<-pre_genos %>%
    left_join(read_csv('/workdir/smallmouth/hudson/redbreast/results/hybrid_results.csv'), by =
'ind') %>%
    left_join(phenos_all_fish, by = 'ind') %>%
    filter(str_detect(pop_final, 'redbreast|hybrid'), !is.na(river_order)) %>%
    mutate(hybrid = if_else(str_detect(pop_final, 'hybrid'), 1, 0),
           river_order = as.character(river_order)) %>%
    dplyr::select(ind, pop, river_order, hybrid)

  m<-lme4::glmer(hybrid ~ river_order + (1|pop),
              family = binomial,
              data = hyb_test)
  summary(m)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: hybrid ~ river_order + (1 | pop)
##    Data: hyb_test
##
##      AIC      BIC   logLik deviance df.resid
##    162.2    177.9    -77.1    154.2      369
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -0.6315 -0.2891 -0.1155 -0.0765  9.8723
##
## Random effects:
##  Groups Name        Variance Std.Dev.
##  pop    (Intercept) 0.6036   0.7769
## Number of obs: 373, groups:  pop, 23
##
## Fixed effects:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -4.1261     0.8049  -5.126 2.95e-07 ***
## river_order4   2.2489     0.8699   2.585  0.00973 **
## river_order5  -0.9344     1.3260  -0.705  0.48103
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) rvr_r4
## river_ordr4 -0.891
## river_ordr5 -0.566  0.534
```

recreating Felipe’s elevation and distance to river

mouth map

```
all_tribs<-felipe_trib %>%
  mutate(Basin = if_else(Basin == 'Moodna' & Elevation >= 105, 'Upper Moodna', Basin))

names<-strata(obj) %>%
  distinct() %>%
  mutate(Transect = as.double(as.character(Transect)))

site_data<-felipe_site %>%
  left_join(names)

ggplot() +
  geom_path(data = all_tribs, aes(distance_to_hudson, Elevation, color = Basin)) +
  ggrepel::geom_label_repel(data = site_data, aes(distance_to_hudson, Elevation, label = Transec
t), label.padding = 0.2, size = 3, box.padding = 0.5, alpha = 0.5) +
  scale_x_reverse() +
  theme_cowplot() +
  labs(x = 'Distance to Hudson confluence (km)', y = 'Elevation (m)', color = 'Sub-basin')
```




```
ggsave('/workdir/smallmouth/hudson/redbreast/results/elevatoin_distance_to_hudson.png', height =
4, width = 6)
```

supplementary bash scripts

run_popgenreport_nulls_bypop.R

```
library(tidyverse)
library(adegenet)
library(poppr)

genos_clean_in<-read_csv('/workdir/smallmouth/hudson/redbreast/results/genos_clean.csv')

obj<-df2genind(dplyr::select(genos_clean_in, -c(ind,pop)), sep = '_', ploidy = 2, ncode = 2, pop
= as.character(genos_clean_in$pop), ind.names = as.character(genos_clean_in$ind))

null_out<-tibble()

problem_loci<-c("Contig5428")

for(pop_in in popNames(obj)){
  for(locus_in in problem_loci){

    nulls_bypop<-obj[loc=locus_in] %>%
      popsub(sublist=pop_in) %>%
      PopGenReport::null.all()

    s1<-nulls_bypop$null.allele.freq$summary1 %>%
      as.data.frame() %>%
      rownames_to_column(var = 'quantile') %>%
      mutate(summary='s1')

    s2<-nulls_bypop$null.allele.freq$summary2 %>%
      as.data.frame() %>%
      rownames_to_column(var = 'quantile') %>%
      mutate(summary = 's2')

    null_in<-bind_rows(s1, s2) %>%
      mutate(pop=pop_in) %>%
      pivot_longer(-c(quantile, summary,pop))

    null_out<-bind_rows(null_in, null_out)
  }
}

# Finally, figure out if its just a few populations causing issues, or all of them
null_out %>%
  write_csv('/workdir/smallmouth/hudson/redbreast/results/null_alleles_summ12_popSpecific.csv')
```

run_popgenreport_nulls.R

```
#!/usr/bin/env Rscript
args = commandArgs(trailingOnly=TRUE)

# install.packages('PopGenReport')
library(tidyverse)
library(adegenet)

genos_clean_in<-read_csv(args[1]) # the input genotype file, ready to be converted to genind, wi
th ind, pop, and loci (No .'s)
output_file_path<-args[2] # where the results should be printed to

print(output_file_path)

# This runs really quickly on the cluster! just a few minutes, but hours on R
obj<-df2genind(dplyr::select(genos_clean_in, -c(ind,pop)), sep = '_', ploidy = 2, ncode = 2, pop
= as.character(genos_clean_in$pop), ind.names = as.character(genos_clean_in$ind))

# Test all populations
nulls<-PopGenReport::null.all(obj)

test_1<-nulls$null.allele.freq$summary1 %>%  # Chakraborty et al. (1994)
  as.data.frame() %>%
  rownames_to_column(var = 'quantile') %>%
  pivot_longer(-quantile) %>%
  pivot_wider(names_from=quantile, values_from=value) %>%
  mutate(test = 'Chakraborty et al. (1994)')

test_2<-nulls$null.allele.freq$summary2 %>%  # Brookfield 1996
  as.data.frame() %>%
  rownames_to_column(var = 'quantile') %>%
  pivot_longer(-quantile) %>%
  pivot_wider(names_from=quantile, values_from=value) %>%
  mutate(test = 'Brookfield (1996)')

bind_rows(test_1, test_2) %>%
  write_csv(output_file_path)
```

run_vareff.R

```
# install.packages('mcmc')
# install.packages("/workdir/smallmouth/hudson/redbreast/VarEff/VarEff.tar.gz",
#                  repos = NULL,
#                  type = "source")
# install.packages('adegenet')
library(mcmc)
library(VarEff)
library(tidyverse)
library(adegenet)

wd<-'/workdir/smallmouth/hudson/redbreast/VarEff/input_data/' # set this to where input and outp
ut files are

# When I'm setting up this script, write these to file so that
#input<- write_csv(tibble(pop=(popNames(obj)), nloc=nLoc(obj)), '/workdir/smallmouth/hudson/redb
reast/results/obj_popnames.csv') # write the popnames to file so can run it independently

input<-read_csv('/workdir/smallmouth/hudson/redbreast/results/obj_popnames.csv')

Ne_vareff_out<-tibble()
for(job_name in input$pop){
  setwd(wd)

  # Some testing here to see if I can set priors correctly, but model i

  #  job_name<-  "Moodna-Moodna-6"

  #   Theta(infile=paste0(wd, job_name, '.txt'), # where is the input file stored?
  #          parafile = job_name, # name the job
  #          NBLOC=unique(input$nloc), # number of loci
  #          MUTAT=0.01, # the mutation rate
  #          DMAX = 8, # start with intermediate value , ~8 (needs to be between 1 and 14)
  #          JMAX=3, # how many times has Ne changed?
  #          MODEL = 'S',  # the mutation model
  #          NBAR=500,  # global prior mean of Ne, start with intermediate (between current and
ancestral, ~500)
  #          VARP1=2,  # start with intermediate value, 2
  #          RHOCORN=0, # keep set to 0
  #          GBAR=5000, # start with large value, 5000
  #          VARP2=3, # start large
  #          Diagonale=0.5, # keep to 0.5
  #          NumberBatch = 1000,
  #          LengthBatch = 1,
  #          SpaceBatch = 10)
  #
  # source(paste0(wd, job_name, '.R'))

  VarEff(infile=paste0(wd, job_name, '.txt'), # where is the input file stored?
         parafile = job_name, # name the job
         NBLOC=unique(input$nloc), # number of loci
         JMAX=2, # # of times we think Ne has changed
         MODEL = 'S',  # the mutation model
```

```
          MUTAT=0.0005, # the mutation rate - this is standard for for fish
          NBAR=1000,  # global prior mean of Ne
          VARP1=3,  # and from here, see the manual
          RHOCORN=0,
          GBAR=5000,
          VARP2=3,
          DMAXPLUS=12,
          Diagonale=0.5,
          NumberBatch = 10000,
          LengthBatch = 10,
          SpaceBatch = 10)

  NatSizeDist(NameBATCH = paste0(wd, job_name, '.Batch'),
              MUTAT = 0.0005, # mutation rate - used standard for fish
              TMAX = 500,  # How many generations back to analyze?
              NBT = 50) # How many time steps to use (this will calculate every TMAX/NBT years)

  ne_vareff_in<-read_delim(paste0(wd, job_name, '.Nstat'),
                           col_names = c('generations','Ne_arithmetic_mean','Ne_harmonic_mea
n','Ne_mode','Ne_median','Ne_5th_quantile','Ne_95th_quantile','Ne_sd'),
                           delim=' ') %>%
    mutate(pop=job_name)

  Ne_vareff_out<-bind_rows(Ne_vareff_out, ne_vareff_in)
}

write_csv(Ne_vareff_out, '/workdir/smallmouth/hudson/redbreast/results/vareff_output.csv')
```

Archiving data

Redbreast microsatellite reference

RS_0018 from below Maiden Lane Dam on Furnace Brook, collected 2020

```
# grab lat/long
phenos_all_fish %>%
  filter(str_detect(pop, '778')) # Not the exact location, but nearby
```

```
## # A tibble: 25 × 31
##    ind     pop    plate GPS   Length Weight Died  Species Notes_data Sec_shocked
##    <chr>   <chr>  <dbl> <chr>  <dbl>  <dbl> <lgl> <chr>   <chr>            <dbl>
##  1 RS_0170 Furna…     5 778      118     20 NA    <NA>    <NA>              2711
##  2 RS_0171 Furna…     5 778       97     12 NA    <NA>    <NA>              2711
##  3 RS_0172 Furna…     5 778      151     54 NA    <NA>    <NA>              2711
##  4 RS_0173 Furna…     5 778       91     14 NA    <NA>    <NA>              2711
##  5 RS_0174 Furna…     5 778       99     17 NA    <NA>    <NA>              2711
##  6 RS_0175 Furna…     5 778       97     15 NA    <NA>    <NA>              2711
##  7 RS_0176 Furna…     5 778      104     17 NA    <NA>    <NA>              2711
##  8 RS_0177 Furna…     5 778      115     26 NA    <NA>    <NA>              2711
##  9 RS_0178 Furna…     5 778      106     18 NA    <NA>    <NA>              2711
## 10 RS_0179 Furna…     5 778      106     20 NA    <NA>    <NA>              2711
## # ℹ 15 more rows
## # ℹ 21 more variables: Shocker <chr>, Location <chr>, Date <dbl>,
## #   fish_sampled <dbl>, Spp_detected <chr>, Notes_meta <chr>, lat <dbl>,
## #   lon <dbl>, n <dbl>, K <dbl>, site <chr>, latitude <dbl>, longitude <dbl>,
## #   cpue <dbl>, river_order <dbl>, altitude_m <dbl>, river <chr>,
## #   river_rank <dbl>, richness <dbl>, patch_size_m <dbl>, distance_to_res <dbl>
```

```
# ftp upload
cd /workdir/backup/redbreast/genome
# make sure that the files I want to upload are the only ones in here
ftp ftp-private.ncbi.nlm.nih.gov
subftp
iodrac9octOckEpp
cd uploads/zarriliam_gmail.com_ghUQIc1h
mkdir redbreast_reference
cd redbreast_reference
put *
```

Microsatellite reads

Info for each biosample list of fastq's to move to their own folder

```
genbank<-phenos %>%
  mutate(lat_lon = paste0(latitude, ', ', longitude),
         date_slash = paste0(substr(Date, 1,2), '/', substr(Date, 3, 4), '/', substr(Date, 5,
8))) %>%
  dplyr::select(ind, date_slash, lat_lon) %>%
  write_tsv('/workdir/smallmouth/hudson/redbreast/results/genbank_submission_ids.tsv')

genbank
```

```
## # A tibble: 321 × 3
##    ind     date_slash lat_lon
##    <chr>   <chr>      <chr>
##  1 RS_0170 10/08/22   41.230348, -73.919553
##  2 RS_0171 10/08/22   41.230348, -73.919553
##  3 RS_0172 10/08/22   41.230348, -73.919553
##  4 RS_0173 10/08/22   41.230348, -73.919553
##  5 RS_0174 10/08/22   41.230348, -73.919553
##  6 RS_0175 10/08/22   41.230348, -73.919553
##  7 RS_0176 10/08/22   41.230348, -73.919553
##  8 RS_0177 10/08/22   41.230348, -73.919553
##  9 RS_0179 10/08/22   41.230348, -73.919553
## 10 RS_0180 10/08/22   41.230348, -73.919553
## # ℹ 311 more rows
```

```
# Next, upload genotype calls
genind2df(obj, sep = '/') %>%
  rownames_to_column(var = 'ind') %>%
  arrange(ind) %>%
  write_csv('/workdir/smallmouth/hudson/redbreast/results/genotypes_exported.csv')
```

metadata

```
phenos %>%
  dplyr::select(basin=watershed, pop, Length, Weight, Date, lat, lon, river_order, altitude_m) %
>%
  write_csv('/workdir/smallmouth/hudson/redbreast/results/metadata_exported.csv')
```