Population Genomics Analysis of Chinook Salmon: Investigating Genetic Relationships and Body Size Determinants

Biology 470 – Computational Genomics
Jaspreet Sidhu
2025-04-11

Abstract

In this study, I conducted a population genomics analysis of Chinook salmon to investigate genetic structure among populations and identify genomic regions associated with body size. I performed quality filtering and trimming on sequencing reads, aligned them to a truncated reference genome, followed by variant calling. Genetic structure was assessed using ADMIXTURE and principal component analysis (PCA), revealing moderate differentiation with evidence of shared ancestry among several populations. Population 7 was found to be genetically distinct, while others clustered into broader genetic groups, likely reflecting geographic proximity. Genome-wide FST analyses confirmed these patterns and highlighted regions of high differentiation, particularly on chromosome 2. A genome-wide association study (GWAS) for body size identified significant SNPs also located on chromosome 2 within the gene *LOC112247479*, suggesting possible genes that are evolving differently between populations. The findings contribute to a better understanding of genetic diversity in Chinook salmon and offer some clues about the genetic factors that may influence their body size.

Methods and Results

1. Filtering reads for adapter sequences and base quality

To examine the quality of the data, we'll be comparing the forward and reverse reads directly using `fastqc/0.12.1` as it can handle uncompressed FASTQ format.

```
for f in fastq/*_R1.fastq.gz; do
    fastqc "$f" "${f/_R1.fastq.gz/_R2.fastq.gz}";
done
```

On average, the forward reads were of slightly higher quality since the quality of reverse reads degraded as the sequencing runs progressed.
The next step is to trim the reads using `trimmomatic-0.39`. This will do 3 things:

1) Remove adapter sequences, which were added to the DNA but aren't from the target sample's genome.
2) Remove low quality reads.
3) Trim poor quality bases off the ends of reads.

```
# I decided to remove bases from the beginning or end of reads if their quality scores were below
3 and also remove any reads shorter than 36 bases.
for f1 in fastq/*_R1.fastq.gz; do
    f2="${f1/_R1.fastq.gz/_R2.fastq.gz}"
    java -jar $EBROOTTRIMMOMATIC/trimmomatic-0.39.jar PE "$f1" "$f2" \
        "${f1/_R1.fastq.gz/_R1.trim.fastq}" "${f1/_R1.fastq.gz/_R1.unpaired.fastq}" \
        "${f2/_R2.fastq.gz/_R2.trim.fastq}" "${f2/_R2.fastq.gz/_R2.unpaired.fastq}" \
        ILLUMINACLIP:$EBROOTTRIMMOMATIC/adapters/TruSeq3-PE.fa:2:30:10:2:True \
        LEADING:3 TRAILING:3 MINLEN:36
done
```

2. Alignment to the Chinook salmon reference genome

Our samples will be aligned to a truncated version of the Chinook genome (GCA_002872995.1), which first needs to be indexed using `samtools/1.18`:

```
samtools faidx ref/SalmonReference.fasta
```

Then the reference genome must also be indexed for the Burrows-Wheeler Aligner using `bwa-mem2/2.2.1`:

```
bwa-mem2 index ref/SalmonReference.fasta
```

Finally, the samples can be aligned using BWA. To align all the population samples we will run the following loop:

```
for R1 in fastq/*_R1.fastq.gz; do
    R2="${R1/_R1.fastq.gz/_R2.fastq.gz}"
    OUT="${R1/_R1.fastq.gz/.sam}"
    echo "Aligning $OUT..."
    bwa-mem2 mem -t 2 "ref/SalmonReference.fasta" "$R1" "$R2" > "$OUT"
done
```

The “sam” files need to be sorted by read position and converted to their binary, “bam”, format.

```
for SAM in fastq/*.sam; do
    BAM="${SAM/.sam/.sort.bam}"
    echo "Sorting $BAM..."
    samtools sort "$SAM" > "$BAM"
done
```

3. Marking or removal of duplicate reads

The next step in processing the alignment files is to mark duplicates using `picard/3.1.0`. These are cases where we have sequenced the same molecule more than once. Since they don’t represent independent observations of the genome, we need to mark them so that we only use one copy in our calculations.

```
for BAM in fastq/*.sort.bam; do
    MARKDUP_BAM="${BAM/.sort.bam/.sort.markdup.bam}"
    METRICS="${BAM/.sort.bam/.dupmetrics.txt}"
    echo "Marking duplicates in $BAM..."
    java -jar $EBROOTPICARD/picard.jar MarkDuplicates \
        I="$BAM" O="$MARKDUP_BAM" M="$METRICS"
done
```

Lastly, we need to index the bam files:

```
for BAM in fastq/*.sort.markdup.bam; do
    echo "Indexing $BAM..."
    samtools index "$BAM"
done
```

Stats of any sample file can be checked by running `samtools coverage "$SAMPLE".sort.markdup.bam`. This shows a summary of the depth and coverage of the sample. Depth indicates how many reads are on each base (on average) and coverage tells you how much of the reference sequence is covered by at least one base.
We can also look at the alignment itself by using `samtools tview "$SAMPLE".sort.markdup.bam SalmonReference.fasta`.
An example alignment for the sample `Biol470.p8.i9.500000.sort.markdup.bam` looks like this:

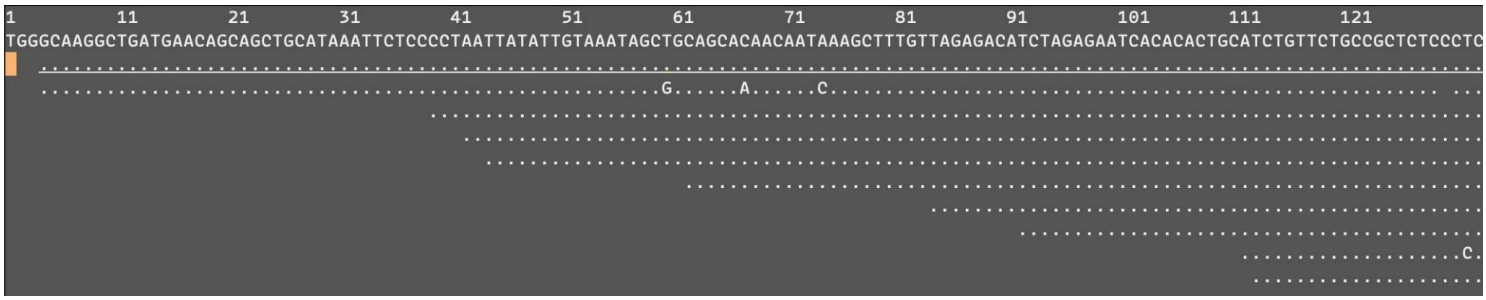

Figure 1. Example output of samtools version 1.21 tview of bam file showing alignment of contigs to chinook salmon reference genome

4. Variant calling

The goal is to detect where our samples differ from the reference genome and how multiple samples differ from each other. To do this, we will use the aligned sequence data to call variants.
First, we need to add a "read group" sample to each sorted and duplicate-marked BAM file so that the sample names are formatted correctly instead of being the file paths. This will be done using a script called `add_rg.sh`:

```
#!/bin/bash

for bam in fastq/*.sort.markdup.bam; do
    # get sample name from filename (remove path and extension)
    filename=$(basename "$bam")
    sample=${filename%%.sort.markdup.bam}
    # define output BAM with read groups
    out="fastq/${sample}.sort.markdup.rg.bam"

    # add or replace read groups
    echo "Processing $bam -> $out"
    java -jar $EBROOTPICARD/picard.jar AddOrReplaceReadGroups \
        I="$bam" \
        O="$out" \
        RGID="$sample" \
        RGSM="$sample" \
        RGLB="$sample" \
        RGPL=Illumina \
        RGPU=NULL

    # index the new BAM
    samtools index "$out"
done
```

Next, we create a file to tell `bcftools/1.18` where all the sorted and duplicate-marked BAM files are and their names:

```
ls fastq/*.sort.markdup.rg.bam > bamlist.rg.txt
```

Now, we're going to use `bcftools mpileup` to call genotypes:

```
# use -q 20 option to require that reads have mapq >= 20.
bcftools mpileup -q 20 -f ref/SalmonReference.fasta -b bamlist.rg.txt > genotypes.rg.g.vcf
```

The next step is to use this `.g.vcf` to call genotypes and create a vcf file:

```
# -m tag allows for multiple alternate alleles when variant calling.
# -v outputs only variant sites (no monomorphic positions).
```

```
# 'bcftools +fill-tags' to fill in some extra information about our variants.
bcftools call -mv genotypes.g.vcf | bcftools +fill-tags > genotypes.bcftools.vcf
```

5. Filtering of the VCF file

If we look at this vcf file, many sites have an AF of 1, meaning that all the samples have the alternate allele. This is partially an artifact of how the data was simulated. Since these sites are monomorphic within our dataset, they're not particularly useful. A common way of filtering a vcf is using a minor (less common) allele frequency cut off.

```
bcftools view -q 0.1:minor genotypes.bcftools.vcf > genotypes.bcftools.maf10.vcf
```

To visualize our vcf directly, we're using R. This lets us see overall patterns in missing data or genotypes and can spot problems that we wouldn't notice otherwise

```
library(tidyr)
library(dplyr)
library(vcfR)
library(ggplot2)

vcf_file <- "/Biol470/project/genotypes.bcftools.maf10.vcf"
vcf <- read.vcfR(vcf_file)

# extract genotypes
gt_data <- vcfR2tidy(vcf, format_fields = 'GT')
gt <- gt_data$gt
names(gt) <- c('Chr', 'Position', 'ID', 'Genotype', 'Allele')
gt$ID <- gsub("Biol470\\.", "", gt$ID)    # remove "Biol470." from ID
gt$ID <- gsub("\\.500000", "", gt$ID)     # remove ".500000" from ID

# genotype heatmap for Chromosome 1
p <- gt %>%
  filter(Chr == 1) %>%
  ggplot(aes(y = ID, x = as.factor(Position), fill = Genotype)) +
  geom_tile() +
  xlab("Position") +
  ylab("Sample") +
  scale_fill_brewer("Genotype", palette = "Set1") +
  labs(
    title = "Figure: Genotype Heatmap for Chromosome 1",
    subtitle = "Each tile represents a genotype call for a sample at a given position",
    fill = "Genotype"
  ) +
  theme_minimal() +
  theme(
    axis.text.x = element_text(size = 4, angle = 90, vjust = 0.5, hjust = 1),
    panel.border = element_rect(colour = "black", fill = NA, size = 1),
    strip.text.y.right = element_text(angle = 0),
    plot.title = element_text(face = "bold"),
    plot.background = element_rect(fill = "white", color = NA)
  )

ggsave("genotype_plot_1.png", plot = p, width = 12, height = 8, units = "in", dpi = 300)
```

Figure 2. Genotype heatmaps for chromosome 1 and 2 of Chinook samples. Heatmaps plot genotypes across chromosomes of individual samples. Graph generated with RStudio's ggplot2 version 3.5.2 library, data processed with bcftools version 1.21

6. Population Structure Analysis

We want to group these samples into populations, so we're going to use the program ADMIXTURE version 1.3.0. This program takes a set of genotypes and attempts to put samples into "groups" or "populations". To decide on the correct number of populations to divide the data into, admixture uses cross-validation. This is a statistical technique that repeatedly builds a model with some of the data and then tests it with the rest. If the test data fits the model well, then it suggests the assumptions (i.e. the number of populations) are probably correct.
Before starting with admixture, we have to pre-process our vcf file. Our chromosomes have non-numeric names but admixture only supports integer chromosome codes, so it will crash if these aren't mapped to integers first. This can be accomplished using `bcftools annotate` and a `mapping.txt` file.
The mapping.txt file will have the following content:

```
chr_1 1
chr_2 2
```

And the command to rename the chromosomes in the vcf file is:

```
bcftools annotate --rename-chrs mapping.txt \
  genotypes.bcftools.maf10.vcf -Ov -o genotypes.bcftools.maf10.nchrs.vcf
```

After moving the input file and the output file with renamed chromosome numbers to a new `/vcf/` directory, we're going to use PLINK version 1.9 to convert our `.vcf` to `.bed`. We need our data in `.bed` format, this is the binary version of `.ped` format, which basically organizes the data into a table with genotype calls. Bed format also has accessory files `.fam` and `.bim` which have information about family structure and SNP identity.

```
plink --vcf genotypes.bcftools.maf10.nchrs.vcf \
  --out genotypes.bcftools.maf10.nchrs \
  --make-bed \
  --allow-extra-chr \
  --double-id \
  --autosome-num 95
```

The next step is to run admixture. We give it our bed file and a k value which is the number of groups that admixture will try to place our individuals into. We can start with k=1, as a baseline.

```
admixture genotypes.bcftools.maf10.nchrs.bed 1
```

We now have two files: `genotypes.bcftools.maf10.nchrs.1.P` and `genotypes.bcftools.maf10.nchrs.1.Q`. The Q file includes the ancestry of each sample for each group. In this case, there's only one group, but this will be more useful with higher K values. The P file estimates allele frequencies for each SNP in each population, which isn't useful for us now. Ultimately for this type of algorithm, we want to try multiple K values and find which one is best. To do that we use a cross-validation method where they mask some genotype values, and try to predict what they are, based on the groupings. Lower cross-validation error means the model is a better fit. We can run this for K from 1 to 10.

```
for K in `seq 10`
  do
  admixture --cv genotypes.bcftools.maf10.nchrs.bed $K -j3 | tee log${K}.out;
done
```

The CV scores produced are as follows:

```
CV error (K=10): 0.94745
CV error (K=1): 0.93352
CV error (K=2): 0.81352
CV error (K=3): 0.78560
CV error (K=4): 0.80519
CV error (K=5): 0.84844
CV error (K=6): 0.85224
```

```
CV error (K=7): 0.81936
CV error (K=8): 0.86640
CV error (K=9): 0.91629
```

From this we can see that the lowest CV is with K=3, and K=4 is only slightly worse. This suggests that there isn't much population grouping in our dataset.
To look at how the population groups are distributed amongst each sample and see if it makes sense biologically, we will plot the Q values for each K.

```
library(ggplot2)
library(dplyr)
library(readr)
library(tidyr)
library(forcats)

# get sample names in order from the .fam
samples <- read_table("/Biol470/project/vcf/genotypes.bcftools.maf10.nchrs.fam",col_names = F) %>%
  select(X1) %>%
  rename(sample.id=X1)
all_data <- tibble()

for (k in 1:10){
  data <- read_delim(paste0("/Biol470/project/vcf/genotypes.bcftools.maf10.nchrs.",k,".Q"),
                     col_names = paste0("Q",seq(1:k)),
                     delim=" ")
  # add sample names
  data$sample.id <- samples$sample.id
  # add the K value
  data$k <- k

  # convert the wide table to a long table, which is easier to plot
  data %>% gather(Q, value, -sample.id,-k) -> data
  # bind together all the outputs of each K value
  all_data <- rbind(all_data,data)
}

# create columns based on sample name
all_data <- all_data %>%
  separate(sample.id, c("species","pop","year","spacer"), ".", remove=F ) %>%
  separate(spacer, c("sample_n"),remove=T)

# make plot
all_data %>%
  filter(k == 10) %>%
  ggplot(aes(x = fct_reorder(sample.id, pop), y = value, fill = factor(Q))) +
  geom_bar(stat = "identity", position = "stack") +
  xlab("Sample ID") +
  ylab("Estimated Ancestry Proportion") +
  labs(
    title = "Figure: Ancestry Proportions for K = 10",
    subtitle = "Each bar represents a sample, with colors indicating inferred ancestry components
(Q1-Q10)",
```

```
    fill = "Ancestry Cluster"
  ) +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 60, hjust = 1),
    plot.title = element_text(face = "bold"),
    plot.background = element_rect(fill = "white", color = NA)
  )

ggsave("admixture_plot_k10.png", width = 12, height = 6, units = "in", dpi = 300)
```

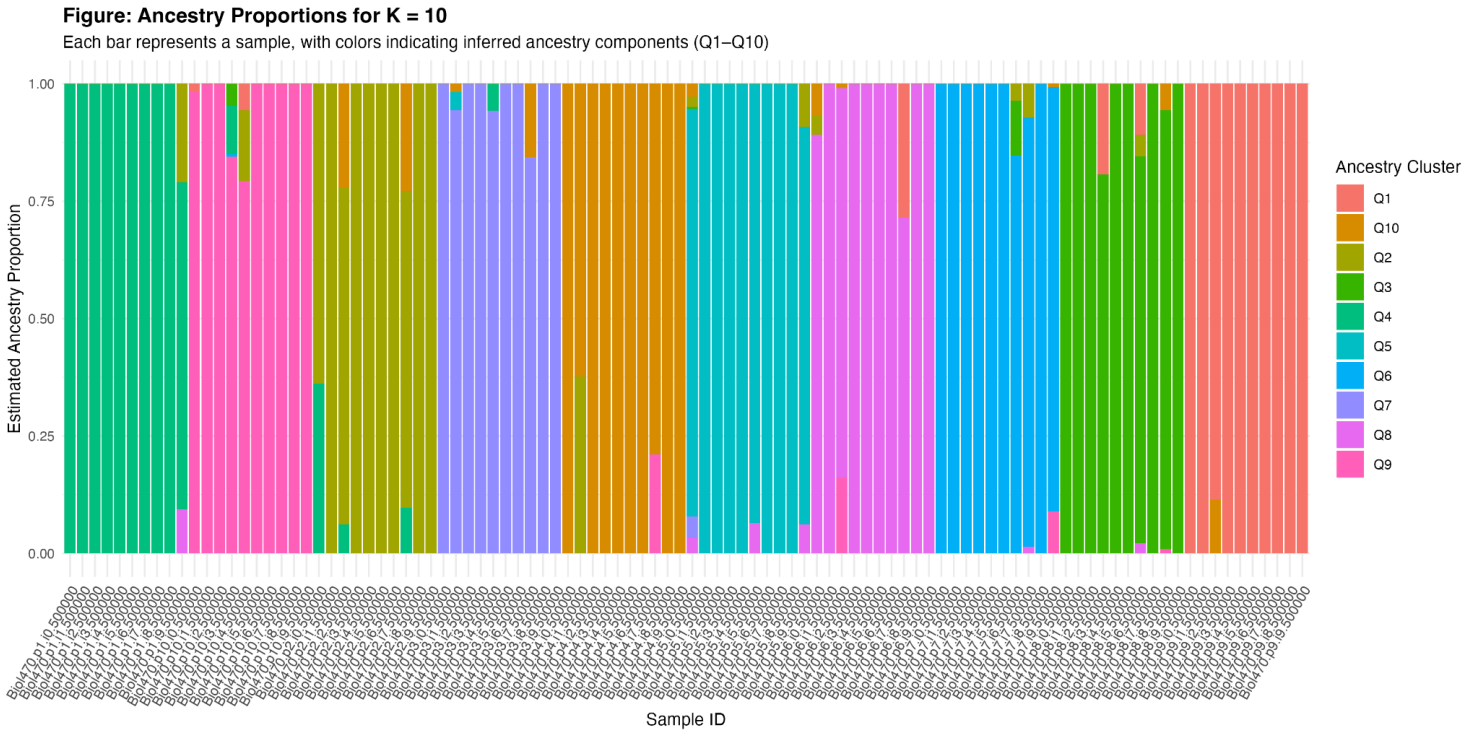

Figure 3. Population structure analysis plot. Sample ID's are ordered by population and colour represents ancestry clusters. Data was processed using admixture version 1.3.0 and plotted with ggplot2 version 3.5.2

7. Principal Component Analysis

A Principal Component Analysis on the samples can help see how they are related. PCA is a statistical technique used to visualize genetic relationships among samples by reducing the dimensionality of genetic data. It transforms the original variables (i.e. genotype counts) into a new set of uncorrelated variables called principal components, which capture the maximum variance in the data.
To do this we're going to use plink again:

```
plink --vcf genotypes.bcftools.maf10.nchrs.vcf \
  --out genotypes.bcftools.maf10.nchrs \
  --pca --allow-extra-chr --double-id --autosome-num 95
```

This outputs a `.eigenvec` and a `.eigenval` file. The `.eigenvec` file has the scores for the first 20 Principal Components, which we can plot.

```
library(readr)
```

```
library(dplyr)
library(ggplot2)

pca_data <- read_table("/Biol470/project/vcf/genotypes.bcftools.maf10.nchrs.eigenvec",
                       col_names = c("sample.id","spacer",paste0("PC",1:20)))

# extract population group (p1, p2, ..., p10) from the sample.id
pca_data <- pca_data %>%
  mutate(group = sub(".*\\.(p[0-9]+)\\..*", "\\1", sample.id))

# plot PC1 vs PC2
ggplot(pca_data, aes(x = PC1, y = PC2, color = group)) +
  geom_point(size = 3, alpha = 0.9) +
  labs(
    title = "PCA of Genotype Data",
    subtitle = "PC1 vs PC2 colored by population group",
    x = "Principal Component 1 (PC1)",
    y = "Principal Component 2 (PC2)",
    color = "Population"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold"),
    plot.background = element_rect(fill = "white", color = NA),
    panel.background = element_rect(fill = "white", color = NA)
  )

ggsave("pca_plot.png", width = 12, height = 8, units = "in", dpi = 300)
```

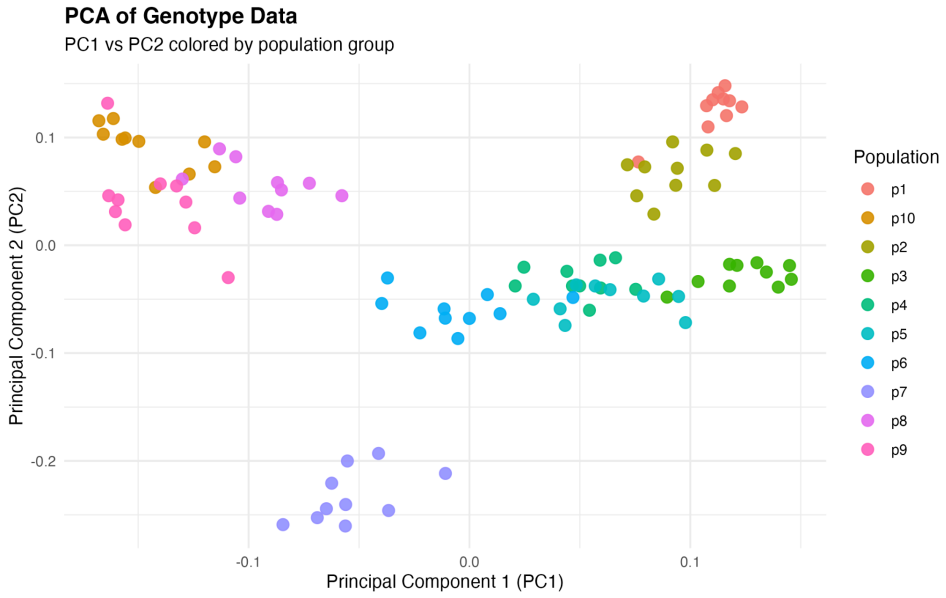

Figure 4. Principal component analysis graph plotting principal component 1 against principal component 2. Colours indicate the sample's population. Data processed using plink version 1.9 and graph generated using ggplot2 version 3.5.2

In the context of population genetics, PCA helps identify patterns of genetic similarity and differentiation among individuals, revealing clusters that may correspond to distinct populations or subpopulations. So, it would be helpful to look at all 20 principle components on a single plot to see if anything stands out.

```
library(readr)
library(dplyr)
library(ggplot2)

pca_data <- read_table("/Biol470/project/vcf/genotypes.bcftools.maf10.nchrs.eigenvec",
                       col_names = c("sample.id", "spacer", paste0("PC", 1:20))) %>%
  mutate(group = sub(".*\\.(p[0-9]+)\\..*", "\\1", sample.id))

# create plot data for PC pairs (PC1 vs PC2, ..., PC19 vs PC20)
plot_data <- lapply(seq(1, 19, by = 2), function(i) {
  pca_data %>%
    select(group, x = paste0("PC", i), y = paste0("PC", i + 1)) %>%
    mutate(pair_label = paste0("PC", i, " vs PC", i + 1))
}) %>%
  bind_rows() %>%
  mutate(pair_label = factor(pair_label, levels = unique(pair_label)))

# create faceted scatter plots
ggplot(plot_data, aes(x = x, y = y, color = group)) +
  geom_point(size = 2, alpha = 0.85) +
  facet_wrap(~ pair_label, scales = "free", ncol = 2) +
  labs(
    title = "Faceted PCA Plots (PC1 to PC20)",
    subtitle = "Each panel shows two consecutive principal components",
    x = "Principal Component (X)",
    y = "Principal Component (Y)",
    color = "Population"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold"),
    strip.text = element_text(face = "bold"),
    plot.background = element_rect(fill = "white", color = NA)
  )

ggsave("faceted_pca_plot.png", width = 12, height = 12, units = "in", dpi = 300)
```

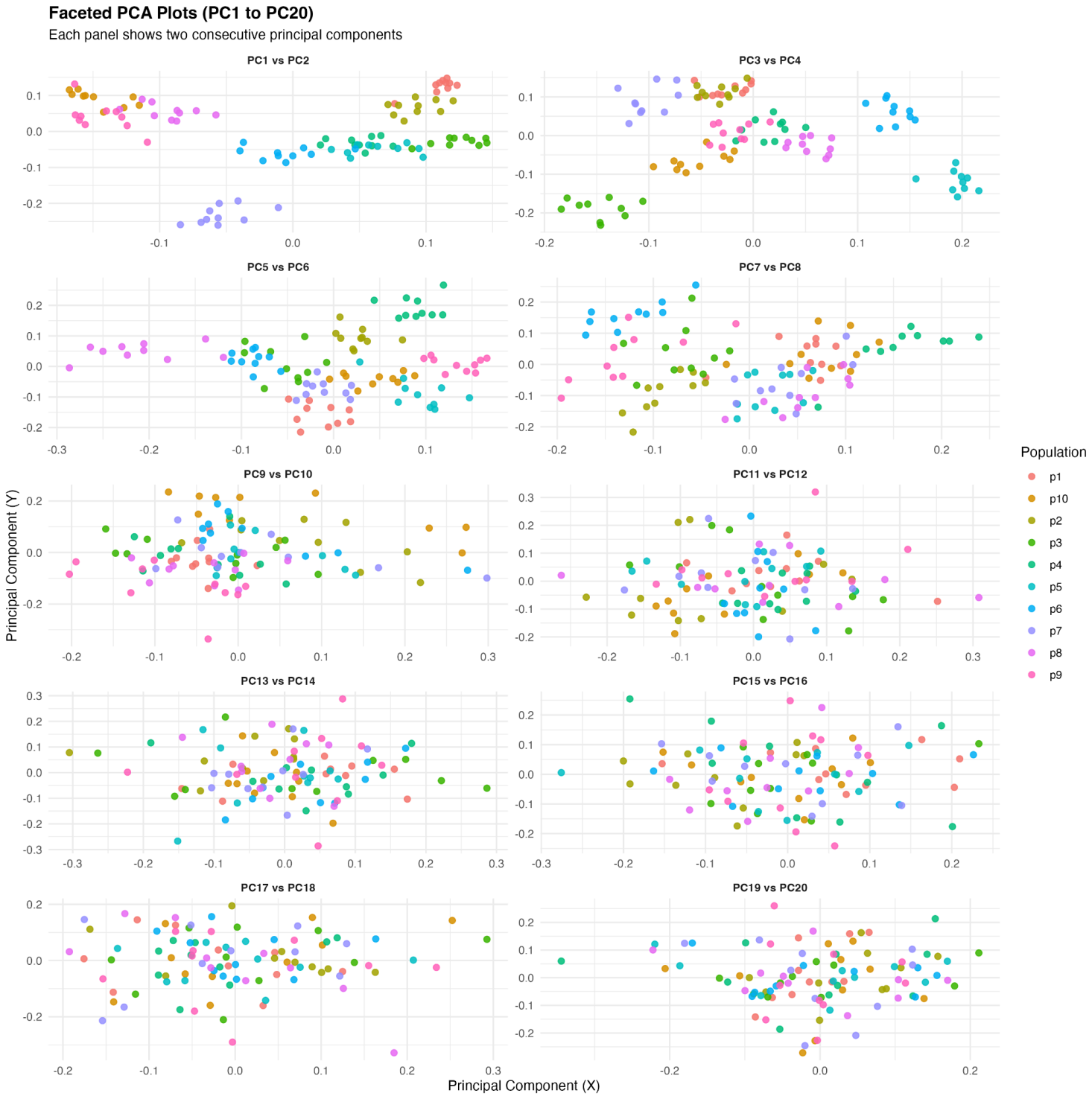

Figure 5. A series of principal component plots of the principal components with the highest variance (PC1 v.s.PC2) to the lowest variance principal components (PC19 v.s. PC20). Colours highlight population origin. Data processed with plink version 1.9 and plotted with ggplot2 version 3.5.2

Lastly, we want to calculate FST between our populations, this is a measure of genetic differentiation. Two populations with FST=0 means that they have the same allele frequencies, FST=1 means that they are fixed for different alleles, and higher values means that populations are more different.
To do this, we need to first make a list of all the samples:

```
bcftools query -l genotypes.bcftools.maf10.vcf > genotypes.bcftools.maf10.samplenames.txt
```

Next, we need to get lists of samples for each population:

```
for pop in p1 p2 p3 p4 p5 p6 p7 p8 p9 p10; do
  grep "$pop" genotypes.bcftools.maf10.samplenames.txt > pop_$pop.txt
done
```

Now we can calculate Weir and Cockerhams FST from our `.vcf` using a script, `pairwise_fst.sh`. This will run 45 pairwise FST calculations across the 10 populations.

```
#!/bin/bash

# create an array of population labels p1 to p10
pops=(p1 p2 p3 p4 p5 p6 p7 p8 p9 p10)

# loop through all unique population pairs and run FST
for (( i=0; i<${#pops[@]}; i++ )); do
  for (( j=i+1; j<${#pops[@]}; j++ )); do
    pop1=${pops[i]}
    pop2=${pops[j]}

    echo "running FST for $pop1 vs $pop2..."
    vcftools --vcf "genotypes.bcftools.maf10.vcf" \
      --weir-fst-pop "pop_${pop1}.txt" \
      --weir-fst-pop "pop_${pop2}.txt" \
      --out "genotypes.bcftools.maf10.${pop1}_vs_${pop2}"
  done
done
```

This gives us a per-site FST value that we can visualize across the genome using Manhattan plots to identify genome-wide and region-specific differentiation patterns across all the populations.

```
library(readr)
library(dplyr)
library(stringr)
library(purrr)
library(ggplot2)

# read and combine all FST files, extracting pairwise comparison labels
files <- list.files(path = "/Biol470/project/vcf/", pattern = "*.weir.fst$", full.names = TRUE)
fst_all <- map_dfr(files, ~ read_tsv(.x, comment = "#", na = ".") %>%
                     mutate(comparison = str_extract(.x, "p[0-9]+_vs_p[0-9]+")))

# order the comparison factor to keep plots consistent
fst_all <- fst_all %>%
  mutate(comparison = factor(comparison, levels = sort(unique(comparison))))

# calculate chromosome lengths and cumulative offsets for manhattan plots
chr_lengths <- fst_all %>%
  group_by(CHROM) %>%
  summarize(length = max(POS, na.rm = TRUE), .groups = "drop") %>%
  mutate(total = cumsum(length) - length)
```

```r
# add cumulative position for each SNP
fst_cumulative <- fst_all %>%
  left_join(chr_lengths %>% select(CHROM, total), by = "CHROM") %>%
  mutate(cumulative_pos = POS + total) %>%
  arrange(comparison, CHROM, POS)

# create chromosome center positions for x-axis labels
axisdf <- fst_cumulative %>%
  group_by(CHROM) %>%
  summarize(center = (min(cumulative_pos, na.rm = TRUE) + max(cumulative_pos, na.rm = TRUE)) / 2,
            .groups = "drop")

# create manhattan plots
ggplot(fst_cumulative, aes(x = cumulative_pos, y = WEIR_AND_COCKERHAM_FST, color = CHROM)) +
  geom_point(alpha = 0.5, size = 0.7) +
  scale_color_manual(values = rep(c("skyblue", "grey50"), length(unique(fst_cumulative$CHROM)))) +
  scale_x_continuous(labels = axisdf$CHROM, breaks = axisdf$center) +
  facet_wrap(~comparison, scales = "free_y", ncol = 5) +
  theme_minimal(base_size = 10) +
  theme(
    legend.position = "none",
    panel.grid.major.x = element_blank(),
    panel.grid.minor = element_blank(),
    strip.text = element_text(face = "bold", size = 10),
    axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1),
    plot.title = element_text(face = "bold"),
    plot.background = element_rect(fill = "white", color = NA)
  ) +
  labs(
    x = "Chromosome",
    y = "Weir and Cockerham FST",
    title = "Pairwise FST Manhattan Plots"
  )

ggsave("all_populations_fst_plot.png", width = 12, height = 16, units = "in", dpi = 300)
```

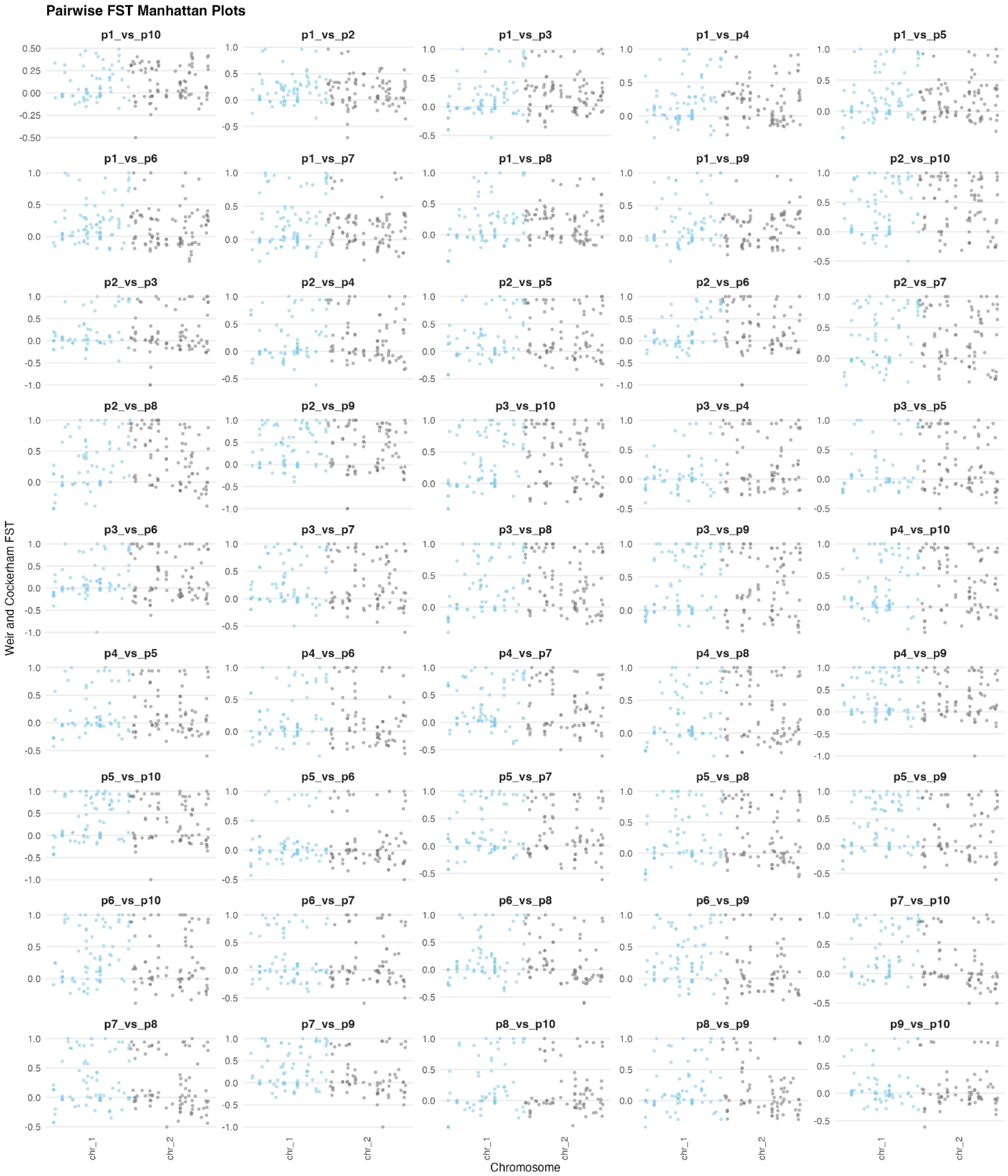
Pairwise FST Manhattan Plots
p1_vs_p10
p1_vs_p2
p1_vs_p3
p1_vs_p4
p1_vs_p5
p1_vs_p6
p1_vs_p7
p1_vs_p8
p1_vs_p9
p2_vs_p10
p2_vs_p3
p2_vs_p4
p2_vs_p5
p2_vs_p6
p2_vs_p7
p2_vs_p8
p2_vs_p9
p3_vs_p10
p3_vs_p4
p3_vs_p5
p3_vs_p6
p3_vs_p7
p3_vs_p8
p3_vs_p9
p4_vs_p10
p4_vs_p5
p4_vs_p6
p4_vs_p7
p4_vs_p8
p4_vs_p9
p5_vs_p10
p5_vs_p6
p5_vs_p7
p5_vs_p8
p5_vs_p9
p6_vs_p10
p6_vs_p7
p6_vs_p8
p6_vs_p9
p7_vs_p10
p7_vs_p8
p7_vs_p9
p8_vs_p10
p8_vs_p9
p9_vs_p10
Weir and Cockerham FST
chr_1
chr_2
Chromosome

Figure 6. Series of plots of pairwise comparisons of each population from population 1 to 10. Blue represents SNP's from chromosome 1 whilst black represents chromosome 2. Data processed using vfctools version 1.13 and plotted with ggplot2 3.5.2

8. Genome-wide association for the trait

Before running a genome-wide association, we have to generate the covariant and phenotype files. This will be done using the PCA values as covariates to control for population structure. We also added the body sizes of the individuals and locations (longitude and latitude) of the populations as additional covariates to explore environmental or spatial effects on population structure. We will be running the GWAS using `body_size` as the phenotype.

```
library(dplyr)
library(readr)
library(readxl)

# get data and extract population and individual (as integers)
gwa_data <- read_table("/Biol470/project/vcf/genotypes.bcftools.maf10.nchrs.eigenvec",
                       col_names = c("sample.id", "spacer", paste0("PC", 1:20))) %>%
  mutate(
    population = as.integer(sub(".*\\.p([0-9]+)\\..*", "\\1", sample.id)),
    individual = as.integer(sub(".*\\.i([0-9]+)\\..*", "\\1", sample.id))
  )

# get body size data
body_size_data <- read_tsv("/Biol470/project/info/body_size.txt")

# get location data
location_data <- read_excel("/Biol470/project/info/Final_project_locations.xlsx") %>%
  mutate(population = as.integer(sub("pop", "", Population))) %>%
  select(population, Latitude, Longitude)

# merge body size and location data into the genetic data
gwa_data <- left_join(gwa_data, body_size_data, by = c("population" = "pop", "individual"))
gwa_data <- left_join(gwa_data, location_data, by = "population")

# generate covariate file for GWAS
gwa_data %>%
  mutate(FID = sample.id, IID = sample.id) %>%
  select(FID, IID, starts_with("PC"), Latitude, Longitude) %>%
  write_tsv("/Biol470/project/info/gwas.pca.txt", col_names = TRUE)

# generate phenotype file for GWAS
gwa_data %>%
  mutate(FID = sample.id, IID = sample.id) %>%
  select(FID, IID, body_size) %>%
  write_tsv("/Biol470/project/info/gwas.pheno.txt", col_names = TRUE)
```

Now we're ready to run GWAS using a linear model to test the association between the `body_size` trait and the genotype. We're using all 20 principle components as covariates to control for population stratification, and we're passing the phenotype values in as a separate file with the `-pheno` option.

```
plink --vcf vcf/genotypes.bcftools.maf10.nchrs.vcf \
    --out gwas/gwas \
```

```
    --linear \
    --allow-extra-chr \
    --double-id \
    --pheno info/gwas.pheno.txt \
    --all-pheno \
    --allow-no-sex \
    --set-missing-var-ids @:#
```

Now we can plot this information using R

```
library(ggplot2)
library(dplyr)
library(readr)

gwas <- read_table("Biol470/project/gwas/gwas.body_size.assoc.linear")

# add -log10(p) values
gwas <- gwas %>%
  mutate(log_p = -log10(P))

gwas_plot <- gwas %>%
  filter(log_p > 1) %>%
  ggplot(aes(x = BP, y = log_p)) +
  geom_point(alpha = 0.6, color = "darkgreen") +
  ylab("Association Strength (-log10(p-value))") +
  xlab("Base Pair Position (BP)") +
  ggtitle("Genome-Wide Association Analysis of Body Size") +
  theme_minimal() +
  theme(
    strip.text = element_text(face = "bold", size = 10),
    axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1),
    plot.title = element_text(face = "bold"),
    plot.background = element_rect(fill = "white", color = NA)
  )

ggsave(filename = "body_size_gwas.png", plot = gwas_plot, width = 7, height = 4, dpi = 300)
```

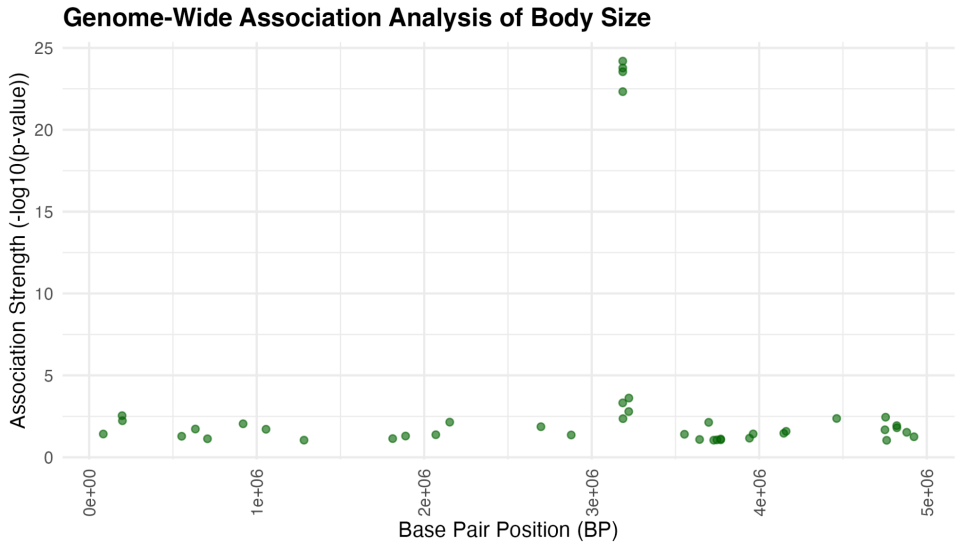

Figure 7. A plot showing a genome-wide association analysis of Chinook body size. Each point represents a SNP tested for association with body size. The y-axis shows -log10(p-value), indicating strength of association. Only SNPs with -log10(p) > 1 are shown.

Discussion

The goal of this study was to analyse genomic reads from Chinook salmon to find associated single nucleotide polymorphisms (SNPs) with the phenotype of body size as well as to better understand the genetic relationship between our sample populations. Our analysis included population structure analysis, principal component analysis, and genome-wide association study. Through our population structure analysis, the program admixture found 10 distinct groups, sorted by colour, when the plot was organized by population. This indicated that the populations where each sample had come from were genetically distinct. However, some samples were expressing multiple Q values, indicating shared genetics and potentially some hybridization between populations.

Afterwards, principal component analysis was done to further analyse the relationship between populations. PC1 was plotted against PC2 and distinct clustering was observed. Overlap between populations was seen creating four larger multi population clusters except for population 7 which was isolated. The multi population clusters were grouped as; [population 10, population 9, population 8], [population 6, population 5, population 4, population 3], and [population 2, population 1]. This suggests that populations within these multi population clusters are more genetically similar with one another compared to other populations. This could be attributed to their proximity to each other as populations that were found to be in a multi population cluster were also physically closer to each other. Clustering was distinct in PC3 vs PC4, but as the plots continued, they became noisier and clustering was no longer visible. Four SNP's at positions *3185216*, *3185230*, *3185308*, and *3185255* all on chromosome 2 were found to have the highest -log(p-value). Using the National Library of Medicine's Genome Data Viewer on Chinook salmon, these SNP's were found to reside on the gene *LOC112247479*. Google Scholar yielded no results when referencing this gene and Chinook salmon body size, so further study into this gene is required. Limitations imposed on the study include unaccounted environmental factors which are known to have an effect on body size (Crozier et al. 2010).

The pairwise FST Manhattan plots highlighted both broad and localized genetic differentiation among populations. Comparisons involving population 10 showed higher FST values, while closely related groups like populations 5 and 6 had lower differentiation. Population 7 consistently displayed elevated FST, reinforcing its distinct genetic profile. Several comparisons also revealed strong FST peaks on chromosome 2, overlapping with SNPs identified in the GWAS for body size. These SNPs fall within the gene *LOC112247479*, suggesting possible divergent selection at this locus linked to phenotypic variation.

References

01. The reference genome is a truncated version of the Chinook genome (GCA_002872995.1)
02. Andrews, S. (2010). FastQC: A Quality Control Tool for High Throughput Sequence Data [Online]. Available online at: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
03. Bolger, A. M., Lohse, M., & Usadel, B. (2014). Trimmomatic: A flexible trimmer for Illumina Sequence Data. Bioinformatics, btu170. Available online at: http://www.usadellab.org/cms/?page=trimmomatic
04. Crozier, L. G., Zabel, R. W., Hockersmith, E. E., & Achord, S. (2010). Interacting effects of density and temperature on body size in multiple populations of Chinook salmon. *Journal of Animal Ecology*, 79(2), 342-349. doi:10.1111/j.1365-2656.2009.01641.x
05. Danecek P, Auton A, Abecasis G, Albers CA, Banks E, DePristo MA, Handsaker RE, Lunter G, Marth GT, Sherry ST, McVean G, Durbin R. The variant call format and VCFtools. Bioinformatics. 2011 Aug 1;27(15):2156–8. Available online at: https://vcftools.github.io/index.html
06. Danecek P, Bonfield JK, Liddle J, Marshall J, Ohan V, Pollard MO, Whitwham A, Keane T, McCarthy SA, Davies RM, Li H. Twelve years of SAMtools and BCFtools. GigaScience. 2021. Available online at: https://doi.org/10.1093/gigascience/giab008.
07. D.H. Alexander, J. Novembre, and K. Lange. Fast model-based estimation of ancestry in unrelated individuals. Genome Research, 19:1655–1664, 2009. Available online at: https://dalexander.github.io/admixture/index.html
08. Garrison E, Marth G. Haplotype-based variant detection from short-read sequencing. arXiv preprint arXiv:1207.3907 [q-bio.GN] 2012. Available online at: https://github.com/freebayes/freebayes
09. Knaus BJ, Grünwald NJ (2017). “VCFR: a package to manipulate and visualize variant call format data in R.” Molecular Ecology Resources, 17(1), 44–53. ISSN 757, https://dx.doi.org/10.1111/1755-0998.12549.
10. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R, and 1000 Genome Project Data Processing Subgroup, The Sequence alignment/map (SAM) format and SAMtools, Bioinformatics (2009) 25(16) 2078-9. Available online at: http://www.htslib.org
11. Picard toolkit. Broad Institute, GitHub repository. (2019). Available online at: https://broadinstitute.github.io/picard/
12. Purcell, S., Neale, B., Todd-Brown, K., Thomas, L., Ferreira, M. A., Bender, D., Maller, J., Sklar, P., de Bakker, P. I., Daly, M. J., & Sham, P. C. (2007). PLINK: a tool set for whole-genome association and population-based linkage analyses. American journal of human genetics, 81(3), 559–575. https://doi.org/10.1086/519795
13. RStudio Team (2020). RStudio: Integrated Development for R. RStudio, PBC, Boston, MA. Available online at: http://www.rstudio.com/
14. U.S. National Library of Medicine. (n.d.). *Genome data viewer - NCBI*. National Center for Biotechnology Information. https://www.ncbi.nlm.nih.gov/gdv/browser/genome/?id=GCF_018296145.1
15. Vasimuddin Md, Sanchit Misra, Heng Li, Srinivas Aluru. Efficient Architecture-Aware Acceleration of BWA-MEM for Multicore Systems. IEEE Parallel and Distributed Processing Symposium (IPDPS), 2019. Available online at: https://github.com/bwa-mem2/bwa-mem2
16. Wickham H, Averick M, Bryan J, Chang W, McGowan LD, François R, Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen TL, Miller E, Bache SM, Müller K, Ooms J, Robinson D, Seidel DP, Spinu V, Takahashi K, Vaughan D, Wilke C, Woo K, Yutani H (2019). “Welcome to the tidyverse.” Journal of Open Source Software, 4(43), 1686. doi:10.21105/joss.01686.