Below is the R and SAS code for differentiation analysis and permutations. These are written such that one should be able to copy and paste the syntax into the proper program and run it with minor adjustments. The most common adjustment being any directories. R syntax is written for RStudio and thus those pasting into the base R should only copy the text between ```{r, [chunk\_name], warning=FALSE} and ```, for each chunk.

The analyses require occasional movement between SAS and R code. In this file, the R code is listed before the SAS code. Running all of these analyses can take a great deal of time on a personal computer (totaling in weeks of runtime, mostly due to SNP analyses and permutations). The WGS SNP analyses has a chunk at the start for extracting 100 loci for analyses.

Workflow for WGS SNP analyses [R code] [SAS Code]

# 1 Subset Allelic Data for Simple Analyses (only for 100 loci sample)

# 2 Prep Allelic Data for Mixed Model Analyses

# 3 Create Template Files

WGS SAS Code

# 4 Merge Results

# 5 AIC Model Selection

# 6 Identify P-Value

# 7 No Within-Line Variance

# 8 Write Results

# 9 Group SNPs

# 10 Select Maxima from Groups

Haplotype

# H11 Recode Alleles

# 3 Create Template Files (Not necessary if files still remain from SNP analyses)

Haplotype SAS Code

# H12 Functions for Combining Haplotype Results

# H13 Read Haplotype Results

# H14 Run Haplotype Functions

# H15 Combine into One File

# 1 Subset Allelic Data for Simple Analyses

Creates a new csv file called "File\_S2\_sample.csv" which contains only the first 100 rows of "File S2.csv".

```{r, subset, warning=FALSE}

setwd("C:\\Users\\david\\Desktop\\Garland\\Manuscript1\\Sample\_Analyses") # Directory

data <- read.csv("File S2.csv", stringsAsFactors = FALSE, nrows = 100) # Reads 1000 rows of "File S2.csv"

write.csv(data, "File\_S2\_sample.csv", row.names = FALSE) # Writes out "File\_S2\_sample.csv"

```

# 2 Prep Allelic Data for Mixed Model Analyses

This step uses the sample data file produced by the previous chunk of code but could easily use the full data set by changing the file name.

### NOTE: The "ready" file is referred to in later analyses and organization of results after SAS code is run

```{r, WGS\_prep, warning=FALSE}

setwd("C:\\Users\\david\\Desktop\\Garland\\Manuscript1\\Sample\_Analyses") # Directory

data <- read.csv("File\_S2\_sample.csv", stringsAsFactors = FALSE) # Reads raw data file

new\_marker <- as.data.frame(c(1:nrow(data))) # Makes a new marker column numbered 1 through the total number of rows in the data file

data\_out <- cbind(data[,c(1,2)], new\_marker, data[,c(4:161)])

colnames(data\_out)[1] <- "chromosome" # Renames column 1 to be "chromosome"

colnames(data\_out)[3] <- "marker" # Renames column 3 to be "marker"

write.csv(data\_out, "File\_S2\_ready.csv", row.names = FALSE, na = ".") # Writes out file swapping "NA" for "." for SAS recognition

```

# 3 Create Template Files

The template files will be loaded into SAS to base output files.

These only need to be created once regardless of how many times the SAS code is run.

These tables are populated with arbitrary data, similar to what will be produced in SAS

```{r, templates, warning=FALSE}

setwd("C:\\Users\\david\\Desktop\\Garland\\Manuscript1\\Sample\_Analyses") # Directory

AIC <- as.data.frame(matrix(c("-2 Res Log Likelihood", "AIC (Smaller is Better)", 55.7, 61.7), 2, 2))

colnames(AIC) <- c("Descr", "Value")

write.csv(AIC, "AIC.csv", row.names = FALSE, quote = FALSE)

Tests <- as.data.frame(matrix(c("pop", "pop", 1, 1, 5.54, 5.54, 65.96, 65.96, 0.0003, 0.0003), 2, 5))

colnames(Tests) <- c("Effect", "NumDF", "DenDF", "FValue", "ProbF")

write.csv(Tests, "Tests.csv", row.names = FALSE, quote = FALSE)

```

### NOTE: "WGS SAS Code" must be run before proceeding to the next step

# 4 Merge Results

The following code reads in and merges the AICc and Test scores

```{r, merge\_results, warning=FALSE}

dir <- "C:\\Users\\david\\Desktop\\Garland\\Manuscript1\\Sample\_Analyses" # Set dir for the next few chunks of code

setwd(dir)

pfull <- read.csv("Results\_full\_tests.csv", stringsAsFactors = FALSE)

pnogroup <- read.csv("Results\_nogroup\_tests.csv", stringsAsFactors = FALSE)

pnogroupLine <- read.csv("Results\_nogroupLine\_tests.csv", stringsAsFactors = FALSE)

pnogroupMouse <- read.csv("Results\_nogroupMouse\_tests.csv", stringsAsFactors = FALSE)

pval <- as.data.frame(cbind(pfull[,5], pnogroup[,5], pnogroupLine[,5], pnogroupMouse[,5])) # Merges all p-values together

colnames(pval) <- c("P\_full", "P\_nogroup", "P\_nogroupLine", "P\_nogroupMouse")

fstat <- as.data.frame(cbind(pfull[,4], pnogroup[,4], pnogroupLine[,4], pnogroupMouse[,4])) # Merges all f-statistics together

colnames(fstat) <- c("F\_full", "F\_nogroup", "F\_nogroupLine", "F\_nogroupMouse")

logfull <- read.csv("Results\_full\_log.csv", stringsAsFactors = FALSE)

lognogroup <- read.csv("Results\_nogroup\_log.csv", stringsAsFactors = FALSE)

lognogroupLine <- read.csv("Results\_nogroupLine\_log.csv", stringsAsFactors = FALSE)

lognogroupMouse <- read.csv("Results\_nogroupMouse\_log.csv", stringsAsFactors = FALSE)

aic <- as.data.frame(cbind(logfull[,3], lognogroup[,3], lognogroupLine[,3], lognogroupMouse[,3])) # Merges all AICc scores together

colnames(aic) <- c("AICC\_f", "AICC\_ng", "AICC\_ngL", "AICC\_ngM")

```

# 5 AIC Model Selection

Identifies the model with the lowest AICc and reports number of ties for the lowest.

Ties are resolved by prioritizing the model with fewer parameters (Simple, SepVarLines, SepVarInd, Full)

```{r, AICselect}

time1 <- Sys.time()

loci <- c(1:nrow(aic))

AIC\_min <- c()

AIC\_name <- c()

AIC\_ties <- c()

for (i in loci) {

m <- min(aic[i,])

AIC\_min <- c(AIC\_min, m) # Populates list with minimum AICc values for each loci, only used for error check later in this chunk

if (sum(aic[i,]==m) > 1) {

AIC\_ties <- c(AIC\_ties, i) # Populates list with loci which contain ties in AICc, not incorporated into final table

}

if (aic[i,2]==m & !is.na(aic[i,2])) {AIC\_name <- c(AIC\_name, "nogroup") # Populates list with model whose AICc matches the minimum

} else if (aic[i,4]==m & !is.na(aic[i,4])) {AIC\_name <- c(AIC\_name, "nogroupMouse") # Populates list with model whose AICc matches the minimum

} else if (aic[i,3]==m & !is.na(aic[i,3])) {AIC\_name <- c(AIC\_name, "nogroupLine") # Populates list with model whose AICc matches the minimum

} else if (aic[i,1]==m & !is.na(aic[i,1])) {AIC\_name <- c(AIC\_name, "full") # Populates list with model whose AICc matches the minimum

} else {AIC\_name <- c(AIC\_name, NA)}

}

AIC\_name <- as.data.frame(AIC\_name)

AIC\_min <- as.data.frame(AIC\_min)

nrow(AIC\_min)==nrow(aic) #Returns TRUE if no rows were omitted from "AIC\_min"

nrow(AIC\_min)==nrow(AIC\_name) #Returns TRUE if no rows were omitted from "AIC\_name"

print(paste("Number of AICc ties = ", length(AIC\_ties), sep = "" )) #Returns number of AICc ties

time2 <- Sys.time()

time2-time1

```

# 6 Identify P-Value

This will use the lowest AICC to select the best P-Value create its own data frame for that list of values.

This will also combine the data for output so that "no variance" chunks can implement changes before writing the csv.

```{r, bestP, warning=FALSE}

time1 <- Sys.time()

bestp <- c()

bestf <- c()

for (i in loci) {

if (AIC\_name[i,1]=="nogroup") {

bestp <- c(bestp, pval[i,2]) # Populates list with p-value associated with model which had lowest AICc

bestf <- c(bestf, fstat[i,2]) # Populates list with f-statistic associated with model which had lowest AICc

} else if (AIC\_name[i,1]=="nogroupMouse") {

bestp <- c(bestp, pval[i,4])

bestf <- c(bestf, fstat[i,4])

} else if (AIC\_name[i,1]=="nogroupLine") {

bestp <- c(bestp, pval[i,3])

bestf <- c(bestf, fstat[i,3])

} else if (AIC\_name[i,1]=="full") {

bestp <- c(bestp, pval[i,1])

bestf <- c(bestf, fstat[i,1])

} else {

bestp <- c(bestp, NA)

bestf <- c(bestf, NA)

}

}

bestp <- as.data.frame(bestp)

bestf <- as.data.frame(bestf)

count <- as.data.frame(c(1:nrow(aic)))

colnames(count) <- c("Loci")

summ <- cbind(count, aicc, AIC\_name, bestf, bestp, pval, fstat) # Combines "best" values and model with all p-values, f-statistics, and AICc scores

colnames(summ)[6] <- "Best\_model"

# This may appear redundant, but it is the most effective way I know to remove factors

setwd(dir)

write.csv(summ, "summ\_int.csv", row.names = FALSE)

summ <- read.csv("summ\_int.csv", stringsAsFactors = FALSE)

time2 <- Sys.time()

time2-time1

```

# 7 No Within-Line Variance

This swaps the "best model" for loci where every line is fixed for one allele or another with "no variance"

This is necessary because the mixed model ANOVA cannot properly analyze these loci.

```{r, lineVar, warning=FALSE}

time1 <- Sys.time()

setwd(dir)

allele\_data <- read.csv("File\_S2\_ready.csv", stringsAsFactors = FALSE, na.strings = ".")

lvar <- allele\_data[,c(1:3)] # Makes new table with SNP location

lvar$AF\_C1 <- rowMeans(allele\_data[,c(4:23)], na.rm = TRUE) # Adds line 1 Allele frequencies

lvar$AF\_C2 <- rowMeans(allele\_data[,c(24:43)], na.rm = TRUE) # Adds line 2 Allele frequencies

lvar$AF\_C4 <- rowMeans(allele\_data[,c(44:63)], na.rm = TRUE) # Adds line 4 Allele frequencies

lvar$AF\_C5 <- rowMeans(allele\_data[,c(64:83)], na.rm = TRUE) # Adds line 5 Allele frequencies

lvar$AF\_HR3 <- rowMeans(allele\_data[,c(84:101)], na.rm = TRUE) # Adds line 3 Allele frequencies

lvar$AF\_HR6 <- rowMeans(allele\_data[,c(102:121)], na.rm = TRUE) # Adds line 6 Allele frequencies

lvar$AF\_HR7 <- rowMeans(allele\_data[,c(122:141)], na.rm = TRUE) # Adds line 7 Allele frequencies

lvar$AF\_HR8 <- rowMeans(allele\_data[,c(142:161)], na.rm = TRUE) # Adds line 8 Allele frequencies

# The allele frequencies file does not have to be saved out for later analyses but may be worth keeping for reference

write.csv(lvar, "File\_S2\_allele\_frequencies\_per\_line.csv", row.names = FALSE)

lvar$fixed <- 0

for (i in 1:nrow(lvar)) {lvar[i,12] <- sum(lvar[i,c(4:11)]==0 | lvar[i,c(4:11)]==1)} # Adds to 12th column (fixed) total number of fixed lines

print(paste("NAs in fixed loci counts = ", sum(is.na(lvar[,12])), ". This should be 0.", sep = "")) # If not 0, this indicates an error

novar <- which(lvar[,12]==8) # Any loci with 8 fixed lines has no within-line variance

for (i in novar) {

summ[i,6] <- "No Variance" # Best\_model replaced

summ[i,7] <- NA # bestf replaced

summ[i,8] <- NA # bestp replaced

}

time2 <- Sys.time()

time2-time1

```

# 8 Write Results

This writes two files.

The "summary" file includes AICc, F-statistic, and P-value for all loci.

The "simplified" file only includes the best model (lowest AICc), and corresponding F-statistic and P-value.

```{r, write, warning=FALSE}

setwd(dir)

summ <- cbind(allele\_data[,c(1,2)], summ)

write.csv(summ, "Complete\_Results\_Summary.csv", row.names = FALSE)

short <- summ[,c(1:3, 8:10)]

write.csv(short, "Complete\_Results\_Simplified.csv", row.names = FALSE)

```

# 9 Group SNPs

The purpose of the code below is to take the clusters of suggestive loci (default p <0.001).

NOTE: This will not produce results with the sample 1000 loci (no p <0.001), however, changing cutoff to <0.1 will produce one local max.

```{r, LMax, warning=FALSE}

dir <- "C:\\Users\\david\\Desktop\\Garland\\Manuscript1\\Sample\_Analyses"

setwd(dir)

short <- read.csv("Complete\_Results\_Simplified.csv", stringsAsFactors = FALSE)

clust <- short[which(short[,6]<0.001), c(3,1,2,6)] #Removes "Best\_model" and "bestf" columns and any loci with p >= 0.001, change here for sample loci

clust$logP <- -log10(clust[,4]) # Adds -logP column

c.count <- c() # List that will be populated with a group number for those suggested loci within 1 million bp of each other

group <- 0 # Tracks the group number

for (i in unique(clust[,2])) {

current <- clust[clust[,2]==i,] # Breaks clust down to individual chromosome

loci <- c(1:nrow(current))

size <- 1 # Denotes the determined size of the current group

for (k in loci) {

if (k==nrow(current)){ # Automatically adds last loci of chromsome into c.count

group <- group + 1

c.count <- c(c.count, c(rep(group, size)))

size <- 1}

else if (current[k+1,3]-current[k,3]<=1000000) {size <- size + 1} # Increase size of group when gap between suggestive loci is <1 million bp

else if (current[k+1,3]-current[k,3]>1000000) { # Stops adding to group size when gap >1 million bp and appends c.count with completed group

group <- group + 1

c.count <- c(c.count, c(rep(group, size)))

size <- 1}

}

}

c.count <- as.data.frame(c.count)

chrom.group <- cbind(clust, c.count) # combines c.count with clust to be written to file

setwd(dir)

write.csv(chrom.group, "grouped\_suggestive\_E-03.csv", row.names = FALSE)

```

# 10 Select Maxima from Groups

This code will take the loci from the previous chunk and find local maxima (most significant) among them

Default one local max for every 500,000 bp in group

```{r, Maxima, warning=FALSE}

setwd(dir)

data <- read.csv("grouped\_suggestive\_E-03.csv", stringsAsFactors = FALSE)

count <- c(1:max(data[,6]))

final <- NULL # Will be populated with all local maxima

for (i in count) { # Determines local maxima one group at a time

current <- data[data[,6]==i,]

if (nrow(current)==1) {final <- rbind(final, current)} # Automatically identifies the loci as the one local max if group has only 1 loci

else {

new <- NULL # Will be populated with local maxima for group

curlen <- c(1:nrow(current))

med <- median(current[,5]) # Identifies median p-value in the group

sec <- ceiling((max(current[,3])-min(current[,3]))/500000) # Identifies total number of local max for group

cursort <- current[order(current[,5], decreasing = TRUE),] # Sorts loci so that most significant is at top

done <- 1 # This is a counter which will stop the following loop when total local max is reached

for (k in curlen) {

if (k==1) {new <- rbind(new, cursort[1,])} # First row is always local max

else if (done<sec) {

gap <- 0

for (j in c(1:done)) {

tempmin <- min(c(cursort[k,1], new[j,1])) # Identifies lowest bp position of current prospective and previously determined local max

tempmax <- max(c(cursort[k,1], new[j,1])) # Identifies highest bp position of current prospective and previously determined local max

curtemp <- current[current[,1]>=tempmin & current[,1]<=tempmax,] # Extracts all suggestive loci between two positions

if (min(curtemp[,5])<=med) {} # Checks that these loci contain at least one loci with p-value <= median

else {gap <- gap + 1} } # Gap ensures that prospective local max fails if a median p-value does not separate it from EVERY previous local max

if (gap==0) { # If prospective local max succeeds, it is added to list and "done" counter is increased

new <- rbind(new, cursort[k,])

done <- done + 1}

else {} }

else if (done>=sec) break # Stops loop when sufficient local max identified

}

final <- rbind(final, new)

}

}

write.csv(final, "local\_max\_500kbp.csv", row.names = FALSE)

```

# H11 Recode Alleles

This will create three files:

A file for 2-allele blocks where the most common allele is denoted with "0" and the alternate is denoted with "1"

A "primary" file where "0" denotes the most common allele and "1" represents both others.

A "secondary" file where "0" denotes the second most common allele and "1" represents both others.

```{r, Recode, warning=FALSE}

setwd("C:\\Users\\david\\Desktop\\Garland\\Manuscript1\\Sample\_Analyses")

FINAL <- read.csv("File S5.csv", stringsAsFactors = FALSE)

time1 <- Sys.time()

Allele2 <- FINAL[which(FINAL$nhap\_g61==2),] # Extracts all loci with only two alleles

Alle.count <- c(1:158)

Alle.len <- c(1:nrow(Allele2))

for (i in Alle.len) { # Performs this loop for each loci in data set

a <- as.numeric(substr(as.character(Allele2[i,8]), 1, 1)) # Identifies first listed allele as 'a'

b <- as.numeric(substr(as.character(Allele2[i,8]), 3, 3)) # Identifies second listed allele as 'b'

if (sum(Allele2[i,9:ncol(Allele2)]==a, na.rm = TRUE) < sum(Allele2[i,9:ncol(Allele2)]==b, na.rm = TRUE)) {

n <- 1 # Uses n as an indicator for which is the most common allele (n=1 for a < b)

} else {n <- 0}

for (j in Alle.count) { # Goes through each mouse and replaces most common allele with 0 and least common with 1

if (is.na(Allele2[i,j+8])) {}

else if (Allele2[i,j+8]==a & n==0) {Allele2[i,j+8] <- 0}

else if (Allele2[i,j+8]==a & n==1) {Allele2[i,j+8] <- 1}

else if (Allele2[i,j+8]==b & n==0) {Allele2[i,j+8] <- 1}

else if (Allele2[i,j+8]==b & n==1) {Allele2[i,j+8] <- 0}

}

}

Allele3 <- FINAL[which(FINAL$nhap\_g61==3),] # Extracts all loci with three alleles

Alle.count <- c(1:158)

Alle.len <- c(1:nrow(Allele3))

for (i in Alle.len) {

a <- sum(Allele3[i,9:ncol(Allele3)]==1, na.rm = TRUE) # Identifies first listed allele as 'a'

b <- sum(Allele3[i,9:ncol(Allele3)]==2, na.rm = TRUE) # Identifies second listed allele as 'b'

c <- sum(Allele3[i,9:ncol(Allele3)]==3, na.rm = TRUE) # Identifies third listed allele as 'c'

if ((a>=b & a<=c) | (a<=b & a>=c)) { # If 'a' is second most common allele

for (j in Alle.count) {

if (is.na(Allele3[i,j+8])) {}

else if (Allele3[i,j+8]==1 ) {Allele3[i,j+8] <- 1} # Change 'a' allele to 1

else if (Allele3[i,j+8]==2 | Allele3[i,j+8]==3) {Allele3[i,j+8] <- 0} # Change other alleles to 0

} }

else if ((b>=a & b<=c) | (b<=a & b>=c)) { # If 'b' is second most common allele

for (j in Alle.count) {

if (is.na(Allele3[i,j+8])) {}

else if (Allele3[i,j+8]==2 ) {Allele3[i,j+8] <- 1} # Change 'b' allele to 1

else if (Allele3[i,j+8]==1 | Allele3[i,j+8]==3) {Allele3[i,j+8] <- 0} # Change other alleles to 0

} }

else if ((c>=a & c<=b) | (c<=a & c>=b)) { # If 'c' is second most common allele

for (j in Alle.count) {

if (is.na(Allele3[i,j+8])) {}

else if (Allele3[i,j+8]==3 ) {Allele3[i,j+8] <- 1} # Change 'c' allele to 1

else if (Allele3[i,j+8]==1 | Allele3[i,j+8]==2) {Allele3[i,j+8] <- 0} # Change other alleles to 0

} }

}

second <- Allele3 # Most common and least common alleles coded as 0, second most common allele coded as 1

Allele3 <- FINAL[which(FINAL$nhap\_g61==3),]

for (i in Alle.len) {

a <- sum(Allele3[i,9:ncol(Allele3)]==1, na.rm = TRUE)

b <- sum(Allele3[i,9:ncol(Allele3)]==2, na.rm = TRUE)

c <- sum(Allele3[i,9:ncol(Allele3)]==3, na.rm = TRUE)

if (c<=a & c<=b) { # If 'c' is least common allele

for (j in Alle.count) {

if (is.na(Allele3[i,j+8])) {}

else if (Allele3[i,j+8]==3 ) {Allele3[i,j+8] <- 1} # Change 'c' allele to 1

else if (Allele3[i,j+8]==1 | Allele3[i,j+8]==2) {Allele3[i,j+8] <- 0} # Change other alleles to 0

} }

else if (b<=a & b<=c) { # If 'b' is least common allele

for (j in Alle.count) {

if (is.na(Allele3[i,j+8])) {}

else if (Allele3[i,j+8]==2 ) {Allele3[i,j+8] <- 1} # Change 'b' allele to 1

else if (Allele3[i,j+8]==1 | Allele3[i,j+8]==3) {Allele3[i,j+8] <- 0} # Change other alleles to 0

} }

else if (a<=b & a<=c) { # If 'a' is least common allele

for (j in Alle.count) {

if (is.na(Allele3[i,j+8])) {}

else if (Allele3[i,j+8]==1 ) {Allele3[i,j+8] <- 1} # Change 'a' allele to 1

else if (Allele3[i,j+8]==2 | Allele3[i,j+8]==3) {Allele3[i,j+8] <- 0} # Change other alleles to 0

} }

}

third <- Allele3 # Most common and second most common alleles coded as 0, least common allele coded as 1

# Add "marker" column for SAS Analyses

marker <- as.data.frame(c(1:nrow(Allele2)))

Allele2 <- cbind(marker, Allele2)

colnames(Allele2)[1] <- "marker"

marker <- as.data.frame(c(1:nrow(second)))

second <- cbind(marker, second)

colnames(second)[1] <- "marker"

third <- cbind(marker, third)

colnames(third)[1] <- "marker"

write.csv(Allele2, "haplotype\_2alleles.csv", row.names = FALSE, na = ".")

write.csv(second, "haplotype\_3alleles\_2nd.csv", row.names = FALSE, na = ".")

write.csv(third, "haplotype\_3alleles\_3rd.csv", row.names = FALSE, na = ".")

time2 <- Sys.time()

time2-time1

# NOTE: Chunk "Create Template Files" must be run before SAS can be run with this data

```

### NOTE: "Haplotype SAS Code" must be run before proceeding to the next step

# H12 Functions for Combining Haplotype Results

These are functions that perform much of the same steps as WGS chunks 5-7

Function: model\_select(aic) [Returns: best\_model], This will select the model with the lowest AICc

Function: identify\_p(pval, fstat, best\_model, aic) [Returns: summ], This combines p-values, f-statistics, best\_model, and aic

Function: hap\_freq(allele\_data) [Returns: lvar], This takes the data used in SAS analyses and computes allele frequency by line

Function: no\_variance(lvar, summ) [Returns: summ], This removes the model and p-value for those loci with no within-line variance (Failed ANOVA assumptions)

For line by line details of the code, please see the WGS version in previous chunks

```{r, HapFunctions, warning=FALSE}

# model\_select()

model\_select <- function(aic) {

loci <- c(1:nrow(aic))

AIC\_min <- c()

AIC\_name <- c()

AIC\_ties <- c()

for (i in loci) {

m <- min(aic[i,])

AIC\_min <- c(AIC\_min, m)

if (sum(aic[i,]==m) > 1) {

AIC\_ties <- c(AIC\_ties, i)

}

if (aic[i,2]==m & !is.na(aic[i,2])) {AIC\_name <- c(AIC\_name, "nogroup")

} else if (aic[i,4]==m & !is.na(aic[i,4])) {AIC\_name <- c(AIC\_name, "nogroupMouse")

} else if (aic[i,3]==m & !is.na(aic[i,3])) {AIC\_name <- c(AIC\_name, "nogroupLine")

} else if (aic[i,1]==m & !is.na(aic[i,1])) {AIC\_name <- c(AIC\_name, "full")

} else {AIC\_name <- c(AIC\_name, NA)}

}

AIC\_name <- as.data.frame(AIC\_name)

AIC\_min <- as.data.frame(AIC\_min)

print(paste("Diagnostics for: ", deparse(substitute(aic)), sep = ""))

print(nrow(AIC\_min)==nrow(aic)) #Returns TRUE if no rows were omitted from "AIC\_min"

print(nrow(AIC\_min)==nrow(AIC\_name)) #Returns TRUE if no rows were omitted from "AIC\_name"

print(paste("Number of AICc ties = ", length(AIC\_ties), sep = "" )) #Returns number of AICc ties

best\_model <- cbind(AIC\_name, AIC\_min)

return(best\_model)

}

# identify\_p()

identify\_p <- function(pval, fstat, best\_model, aic) {

loci <- c(1:nrow(pval))

bestp <- c()

bestf <- c()

for (i in loci) {

if (best\_model[i,1]=="nogroup") {

bestp <- c(bestp, pval[i,2])

bestf <- c(bestf, fstat[i,2])

} else if (best\_model[i,1]=="nogroupMouse") {

bestp <- c(bestp, pval[i,4])

bestf <- c(bestf, fstat[i,4])

} else if (best\_model[i,1]=="nogroupLine") {

bestp <- c(bestp, pval[i,3])

bestf <- c(bestf, fstat[i,3])

} else if (best\_model[i,1]=="full") {

bestp <- c(bestp, pval[i,1])

bestf <- c(bestf, fstat[i,1])

} else {

bestp <- c(bestp, NA)

bestf <- c(bestf, NA)

}

}

bestp <- as.data.frame(bestp)

bestf <- as.data.frame(bestf)

count <- as.data.frame(c(1:nrow(aic)))

colnames(count) <- c("Loci")

summ <- cbind(count, aic, best\_model[,1], bestf, bestp, pval, fstat)

colnames(summ)[6] <- "Best\_model"

write.csv(summ, "summ\_int.csv", row.names = FALSE)

summ <- read.csv("summ\_int.csv", stringsAsFactors = FALSE)

return(summ)

}

# hap\_freq()

hap\_freq <- function(allele\_data) {

lvar <- allele\_data[,c(1:9)]

lvar$AF\_C1 <- rowMeans(allele\_data[,c(10:29)], na.rm = TRUE)

lvar$AF\_C2 <- rowMeans(allele\_data[,c(30:49)], na.rm = TRUE)

lvar$AF\_C4 <- rowMeans(allele\_data[,c(50:69)], na.rm = TRUE)

lvar$AF\_C5 <- rowMeans(allele\_data[,c(70:89)], na.rm = TRUE)

lvar$AF\_HR3 <- rowMeans(allele\_data[,c(90:107)], na.rm = TRUE)

lvar$AF\_HR6 <- rowMeans(allele\_data[,c(108:127)], na.rm = TRUE)

lvar$AF\_HR7 <- rowMeans(allele\_data[,c(128:147)], na.rm = TRUE)

lvar$AF\_HR8 <- rowMeans(allele\_data[,c(148:167)], na.rm = TRUE)

return(lvar)

}

# no\_variance()

no\_variance <- function(lvar, summ) {

lvar$fixed <- 0

for (i in 1:nrow(lvar)) {lvar[i,18] <- sum(lvar[i,c(10:17)]==0 | lvar[i,c(10:17)]==1)}

novar <- which(lvar[,18]==8)

for (i in novar) {

summ[i,6] <- "No Variance"

summ[i,7] <- NA

summ[i,8] <- NA

}

return(summ)

}

```

# H13 Read Haplotype Results

```{r, HapResults, warning=FALSE}

setwd("C:\\Users\\david\\Desktop\\Garland\\Manuscript1\\Sample\_Analyses") # Set dir to match SAS input directory

allele\_data2 <- read.csv("haplotype\_2alleles.csv", stringsAsFactors = FALSE, na.strings = ".")

allele\_data32 <- read.csv("haplotype\_3alleles\_2nd.csv", stringsAsFactors = FALSE, na.strings = ".")

allele\_data33 <- read.csv("haplotype\_3alleles\_3rd.csv", stringsAsFactors = FALSE, na.strings = ".")

setwd("C:\\Users\\david\\Desktop\\Garland\\Manuscript1\\Sample\_Analyses\\haplotype") # Set dir to match SAS output directory

# Load 2-Allele Results

pfull <- read.csv("haplotype\_2alleles\_full\_tests.csv", stringsAsFactors = FALSE)

pnogroup <- read.csv("haplotype\_2alleles\_nogroup\_tests.csv", stringsAsFactors = FALSE)

pnogroupLine <- read.csv("haplotype\_2alleles\_nogroupLine\_tests.csv", stringsAsFactors = FALSE)

pnogroupMouse <- read.csv("haplotype\_2alleles\_nogroupMouse\_tests.csv", stringsAsFactors = FALSE)

pval2 <- as.data.frame(cbind(pfull[,5], pnogroup[,5], pnogroupLine[,5], pnogroupMouse[,5]))

colnames(pval2) <- c("P\_full", "P\_nogroup", "P\_nogroupLine", "P\_nogroupMouse")

fstat2 <- as.data.frame(cbind(pfull[,4], pnogroup[,4], pnogroupLine[,4], pnogroupMouse[,4]))

colnames(fstat2) <- c("F\_full", "F\_nogroup", "F\_nogroupLine", "F\_nogroupMouse")

logfull <- read.csv("haplotype\_2alleles\_full\_log.csv", stringsAsFactors = FALSE)

lognogroup <- read.csv("haplotype\_2alleles\_nogroup\_log.csv", stringsAsFactors = FALSE)

lognogroupLine <- read.csv("haplotype\_2alleles\_nogroupLine\_log.csv", stringsAsFactors = FALSE)

lognogroupMouse <- read.csv("haplotype\_2alleles\_nogroupMouse\_log.csv", stringsAsFactors = FALSE)

aic2 <- as.data.frame(cbind(logfull[,3], lognogroup[,3], lognogroupLine[,3], lognogroupMouse[,3]))

colnames(aic2) <- c("AICC\_f", "AICC\_ng", "AICC\_ngL", "AICC\_ngM")

# Load 3-Allele (2nd) Results

pfull <- read.csv("haplotype\_3alleles\_2nd\_full\_tests.csv", stringsAsFactors = FALSE)

pnogroup <- read.csv("haplotype\_3alleles\_2nd\_nogroup\_tests.csv", stringsAsFactors = FALSE)

pnogroupLine <- read.csv("haplotype\_3alleles\_2nd\_nogroupLine\_tests.csv", stringsAsFactors = FALSE)

pnogroupMouse <- read.csv("haplotype\_3alleles\_2nd\_nogroupMouse\_tests.csv", stringsAsFactors = FALSE)

pval32 <- as.data.frame(cbind(pfull[,5], pnogroup[,5], pnogroupLine[,5], pnogroupMouse[,5]))

colnames(pval32) <- c("P\_full", "P\_nogroup", "P\_nogroupLine", "P\_nogroupMouse")

fstat32 <- as.data.frame(cbind(pfull[,4], pnogroup[,4], pnogroupLine[,4], pnogroupMouse[,4]))

colnames(fstat32) <- c("F\_full", "F\_nogroup", "F\_nogroupLine", "F\_nogroupMouse")

logfull <- read.csv("haplotype\_3alleles\_2nd\_full\_log.csv", stringsAsFactors = FALSE)

lognogroup <- read.csv("haplotype\_3alleles\_2nd\_nogroup\_log.csv", stringsAsFactors = FALSE)

lognogroupLine <- read.csv("haplotype\_3alleles\_2nd\_nogroupLine\_log.csv", stringsAsFactors = FALSE)

lognogroupMouse <- read.csv("haplotype\_3alleles\_2nd\_nogroupMouse\_log.csv", stringsAsFactors = FALSE)

aic32 <- as.data.frame(cbind(logfull[,3], lognogroup[,3], lognogroupLine[,3], lognogroupMouse[,3]))

colnames(aic32) <- c("AICC\_f", "AICC\_ng", "AICC\_ngL", "AICC\_ngM")

# Load 3-Allele (3rd) Results

pfull <- read.csv("haplotype\_3alleles\_3rd\_full\_tests.csv", stringsAsFactors = FALSE)

pnogroup <- read.csv("haplotype\_3alleles\_3rd\_nogroup\_tests.csv", stringsAsFactors = FALSE)

pnogroupLine <- read.csv("haplotype\_3alleles\_3rd\_nogroupLine\_tests.csv", stringsAsFactors = FALSE)

pnogroupMouse <- read.csv("haplotype\_3alleles\_3rd\_nogroupMouse\_tests.csv", stringsAsFactors = FALSE)

pval33 <- as.data.frame(cbind(pfull[,5], pnogroup[,5], pnogroupLine[,5], pnogroupMouse[,5]))

colnames(pval33) <- c("P\_full", "P\_nogroup", "P\_nogroupLine", "P\_nogroupMouse")

fstat33 <- as.data.frame(cbind(pfull[,4], pnogroup[,4], pnogroupLine[,4], pnogroupMouse[,4]))

colnames(fstat33) <- c("F\_full", "F\_nogroup", "F\_nogroupLine", "F\_nogroupMouse")

logfull <- read.csv("haplotype\_3alleles\_3rd\_full\_log.csv", stringsAsFactors = FALSE)

lognogroup <- read.csv("haplotype\_3alleles\_3rd\_nogroup\_log.csv", stringsAsFactors = FALSE)

lognogroupLine <- read.csv("haplotype\_3alleles\_3rd\_nogroupLine\_log.csv", stringsAsFactors = FALSE)

lognogroupMouse <- read.csv("haplotype\_3alleles\_3rd\_nogroupMouse\_log.csv", stringsAsFactors = FALSE)

aic33 <- as.data.frame(cbind(logfull[,3], lognogroup[,3], lognogroupLine[,3], lognogroupMouse[,3]))

colnames(aic33) <- c("AICC\_f", "AICC\_ng", "AICC\_ngL", "AICC\_ngM")

```

# H14 Run Haplotype Functions

```{r, runFunctions, warning=FALSE}

setwd("C:\\Users\\david\\Desktop\\Garland\\Manuscript1\\Sample\_Analyses\\haplotype\_results") # Set dir to match desired R output directory

# 2-Allele Combination

bestmodel2 <- model\_select(aic2)

summ2 <- identify\_p(pval2, fstat2, bestmodel2, aic2)

lvar2 <- hap\_freq(allele\_data2)

write.csv(lvar2, "Haplotype\_2alleles\_frequencies\_per\_line.csv", row.names = FALSE)

summ\_2 <- no\_variance(lvar2, summ2)

summ\_2 <- cbind(allele\_data2[,c(2:9)], summ\_2)

write.csv(summ\_2, "Haplotype\_2alleles\_Results\_Summary.csv", row.names = FALSE)

short2 <- summ\_2[,c(1:9, 14:16)]

write.csv(short2, "Haplotype\_2alleles\_Results\_Simplified.csv", row.names = FALSE)

# 3-Allele (2nd) Combination

bestmodel32 <- model\_select(aic32)

summ32 <- identify\_p(pval32, fstat32, bestmodel32, aic32)

lvar32 <- hap\_freq(allele\_data32)

write.csv(lvar32, "Haplotype\_3alleles\_2nd\_frequencies\_per\_line.csv", row.names = FALSE)

summ\_32 <- no\_variance(lvar32, summ32)

summ\_32 <- cbind(allele\_data32[,c(2:9)], summ\_32)

write.csv(summ\_32, "Haplotype\_3alleles\_2nd\_Results\_Summary.csv", row.names = FALSE)

short32 <- summ\_32[,c(1:9, 14:16)]

write.csv(short32, "Haplotype\_3alleles\_2nd\_Results\_Simplified.csv", row.names = FALSE)

# 3-Allele (3rd) Combination

bestmodel33 <- model\_select(aic33)

summ33 <- identify\_p(pval33, fstat33, bestmodel33, aic33)

lvar33 <- hap\_freq(allele\_data33)

write.csv(lvar33, "Haplotype\_3alleles\_3rd\_frequencies\_per\_line.csv", row.names = FALSE)

summ\_33 <- no\_variance(lvar33, summ33)

summ\_33 <- cbind(allele\_data33[,c(2:9)], summ\_33)

write.csv(summ\_33, "Haplotype\_3alleles\_3rd\_Results\_Summary.csv", row.names = FALSE)

short33 <- summ\_33[,c(1:9, 14:16)]

write.csv(short33, "Haplotype\_3alleles\_3rd\_Results\_Simplified.csv", row.names = FALSE)

```

# H15 Combine into One File

```{r, CombineP, warning=FALSE}

setwd("C:\\Users\\david\\Desktop\\Garland\\Manuscript1\\Sample\_Analyses\\haplotype\_results") # Set dir to match haplotype output directory

short2 <- read.csv("Haplotype\_2alleles\_Results\_Simplified.csv", stringsAsFactors = FALSE)

short32 <- read.csv("Haplotype\_3alleles\_2nd\_Results\_Simplified.csv", stringsAsFactors = FALSE)

short33 <- read.csv("Haplotype\_3alleles\_3rd\_Results\_Simplified.csv", stringsAsFactors = FALSE)

# Combine 3-allele p-values with Fisher's method

mark <- c(1:nrow(short32))

p.meta <- c()

logp.meta <- c()

for (i in mark) {

chi <- -2 \* (log(short32[i,12]) + log(short33[i,12]))

pvalue <- 1 - pchisq(chi, df=4)

logp <- -1 \* log10(pvalue)

p.meta <- c(p.meta, pvalue)

logp.meta <- c(logp.meta, logp)

}

p.meta <- as.data.frame(p.meta)

logp.meta <- as.data.frame(logp.meta)

result <- cbind(short32[,c(1:9)], p.meta, logp.meta, short32[,12], short33[,12])

colnames(result) <- c("chr", "Block", "n.hap", "Nblock\_g61", "MinPOS", "MaxPOS", "nhap\_g61", "haps\_g61", "Loci", "Combined\_P", "Combined\_logP",

"2nd\_P", "3rd\_P")

write.csv(result, "Haplotype\_3alleles\_combinedP.csv", row.names = FALSE)

# Combine 2-allele and 3-allele results

all2 <- short2[,c(1:8, 12)]

all3 <- result[,c(1:8, 10)]

colnames(all3)[9] <- "P"

colnames(all2) <- colnames(all3)

all\_hap <- rbind(all2, all3)

all\_hap <- all\_hap[order(all\_hap[,1], all\_hap[,2]),]

write.csv(all\_hap, "Complete\_Haplotype\_Results.csv", row.names = FALSE)

```

**SAS Code**

WGS SAS Code

/\*The purpose of this file is to perform the mivque0 analysis with PROC MIXED using the full genome data.\*/

%let \_timer\_start = %sysfunc(datetime()); /\* Start timer, for total runtime of program \*/

/\* The following 4 lines change options in SAS to not save unnecessary data\*/

options nonotes; /\* Turns off blue notes in Log so that it doesn't need to be cleared periodically\*/

ods noresults; /\* SAS Support - no results accumulating in the results tree\*/

ods \_all\_ close; /\* SAS Support - no output\*/

ods graphics off; /\* SAS Support - no ods graphics\*/

%let dir\_in=C:\Users\david\Desktop\Garland\Manuscript1\Sample\_Analyses; /\*Set working directory for files being imported\*/

%let dir\_out=C:\Users\david\Desktop\Garland\Manuscript1\Sample\_Analyses; /\*Set working directory for files being exported\*/

filename ina "&dir\_in\File S3.csv"; /\*Has mouse data\*/

filename inb "&dir\_in\File\_S2\_ready.csv"; /\*Has Genotypes and marker columns\*/

filename inc "&dir\_in\Tests.csv"; /\*Used to make empty "tests" tables\*/

filename ind "&dir\_in\AIC.csv"; /\*Used to make empty "log" tables\*/

**proc** **import** datafile=ina out=meta dbms=csv replace; /\* Loads mouse data in SAS as "meta"\*/

**proc** **import** datafile=inb out=geno dbms=csv replace; /\* Loads genetic data in SAS as "geno"\*/

**proc** **import** datafile=inc out=test dbms=csv replace; /\* Loads test results in SAS as "test"\*/

**proc** **import** datafile=ind out=aic dbms=csv replace; /\* Loads test results in SAS as "aic"\*/

**run**;

/\* That should take care of loading and initial prep of data\*/

/\* Formating Columns\*/

/\* NOTE: These may have to be changed depending on defaults of user's computer\*/

**data** test1; length Effect $**9**; set test; format DenDF BEST4. ProbF PVALUE12.;

**data** aic1; length Descr $**25**; set aic;

**run**;

**proc** **sql**; /\* These are empty tables to be populated in the macro\*/

create table full\_tests like test1;

create table nogroup\_tests like test1;

create table nogroupLine\_tests like test1;

create table nogroupMouse\_tests like test1;

create table noLineWgroup\_tests like test1;

create table noLine\_tests like test1;

create table full\_log like aic1;

create table nogroup\_log like aic1;

create table nogroupLine\_log like aic1;

create table nogroupMouse\_log like aic1;

create table noLineWgroup\_log like aic1;

create table noLine\_log like aic1;

**quit**;

**data** geno1 (drop=Chromosome POS); set geno; **run**; /\* Drops the unnecessary columns\*/

**proc** **transpose** data=geno1 out=geno2; **run**; /\* Transposes the newly merge dataset\*/

%let n=0; **run**;

sasfile geno1 open;

sasfile meta open;

/\* This is the macro which is necessary for running procedures in do loops\*/

/\* NOTE: 100 will have to be changed to run additional loci (5932124 for whole genome)\*/

**%macro** ***allModels***; /\* "allModels" is the name of the macro\*/

%do k = **1** %to **100**; /\* This should always start on 1 and end on however many loci are in the data\*/

%let n=%eval(&n+1);

data allele; /\* This identifies the correct locus and drop marker\*/

set geno1;

where marker=&n;

data alleleNoM (drop=marker);

set allele;

proc transpose data=alleleNoM out=allele1; /\* This transposes the one locus to be merged with the mouse data\*/

data locus; merge meta allele1;

proc mixed data=locus method=mivque0; /\* Full model\*/

class pop sub mouse; /\* Identifies population and subpopulation in meta data and is shuffled with mouseID\*/

model COL1=pop/solution; /\* Describes model\*/

random sub(pop) /group=pop; /\* Sets random effects, nesting, and groups for analysis parameters\*/

random mouse(sub pop) /group = pop; /\* Sets random effects, nesting, and groups for analysis parameters\*/

ods output Tests3=type3 FitStatistics=log; /\* Identifies output table names\*/

proc datasets;

append base=full\_tests data=type3 force; /\* Appends the p-values\*/

append base=full\_log data=log force; /\* Appends the AIC\*/

proc mixed data=locus method=mivque0; /\* nogroup model\*/

class pop sub mouse; /\* Identifies population and subpopulation in meta data and is shuffled with mouseID\*/

model COL1=pop/solution; /\* Describes model\*/

random sub(pop); /\* Sets random effects, nesting, and groups for analysis parameters\*/

random mouse(sub pop); /\* Sets random effects, nesting, and groups for analysis parameters\*/

ods output Tests3=type3 FitStatistics=log; /\* Identifies output table names\*/

proc datasets;

append base=nogroup\_tests data=type3 force; /\* Appends the p-values\*/

append base=nogroup\_log data=log force; /\* Appends the AIC\*/

proc mixed data=locus method=mivque0; /\* nogroupLine model\*/

class pop sub mouse; /\* Identifies population and subpopulation in meta data and is shuffled with mouseID\*/

model COL1=pop/solution; /\* Describes model\*/

random sub(pop); /\* Sets random effects, nesting, and groups for analysis parameters\*/

random mouse(sub pop) /group = pop; /\* Sets random effects, nesting, and groups for analysis parameters\*/

ods output Tests3=type3 FitStatistics=log; /\* Identifies output table names\*/

proc datasets;

append base=nogroupLine\_tests data=type3 force; /\* Appends the p-values\*/

append base=nogroupLine\_log data=log force; /\* Appends the AIC\*/

proc mixed data=locus method=mivque0; /\* nogroupMouse model\*/

class pop sub mouse; /\* Identifies population and subpopulation in meta data and is shuffled with mouseID\*/

model COL1=pop/solution; /\* Describes model\*/

random sub(pop) /group = pop; /\* Sets random effects, nesting, and groups for analysis parameters\*/

random mouse(sub pop); /\* Sets random effects, nesting, and groups for analysis parameters\*/

ods output Tests3=type3 FitStatistics=log; /\* Identifies output table names\*/

proc datasets;

append base=nogroupMouse\_tests data=type3 force; /\* Appends the p-values\*/

append base=nogroupMouse\_log data=log force; /\* Appends the AIC\*/

data \_null\_; /\* Counts off every 50 loci analyzed\*/

y = &n;

z=mod(y,**50**);

if z=**0** then do;

put y;

end;

%end;

**%mend** allModels; /\* mend ends the macro, differing notes on whether the name needs to be included\*/

/\* This actually runs the analysis\*/

%***allModels***;

**run**;

/\* This changes all p-values to 10 decimal places until stated otherwise\*/

**data** full\_tests1; set full\_tests; format ProbF pvalue12.10; **run**;

**data** nogroup\_tests1; set nogroup\_tests; format ProbF pvalue12.10; **run**;

**data** nogroupLine\_tests1; set nogroupLine\_tests; format ProbF pvalue12.10; **run**;

**data** nogroupMouse\_tests1; set nogroupMouse\_tests; format ProbF pvalue12.10; **run**;

/\* Organize log\*/

/\*NOTE: Values must be changed according to the number of loci analyzed\*/

/\* Values for 100 loci sample: 100, 200, 101, 300, 201, 301\*/

/\* Values for whole genome: 5932124, 11864248, 5932125, 17796372, 11864249, 17796373\*/

**data** log; set full\_log; format Value BEST12.; **run**; /\* Changes sig figs to 4 digits before decimal and 8 after\*/

**proc** **sort** data=log; by Descr; **run**;

**data** log2 (rename=(Value=Log2) drop=descr); set log; if \_n\_ gt **100** then delete; **run**;

**data** logAIC (rename=(Value=AIC) drop=descr); set log; if \_n\_ gt **200** then delete; if \_n\_ lt **101** then delete; **run**;

**data** logAICC (rename=(Value=AICC) drop=descr); set log; if \_n\_ gt **300** then delete; if \_n\_ lt **201** then delete; **run**;

**data** logBIC (rename=(Value=BIC) drop=descr); set log; if \_n\_ lt **301** then delete; **run**;

**data** full\_log\_final; merge log2 logAIC logAICC logBIC; **run**;

**data** log; set nogroup\_log; format Value BEST12.; **run**; /\* Changes sig figs to 4 digits before decimal and 8 after\*/

**proc** **sort** data=log; by Descr; **run**;

**data** log2 (rename=(Value=Log2) drop=descr); set log; if \_n\_ gt **100** then delete; **run**;

**data** logAIC (rename=(Value=AIC) drop=descr); set log; if \_n\_ gt **200** then delete; if \_n\_ lt **101** then delete; **run**;

**data** logAICC (rename=(Value=AICC) drop=descr); set log; if \_n\_ gt **300** then delete; if \_n\_ lt **201** then delete; **run**;

**data** logBIC (rename=(Value=BIC) drop=descr); set log; if \_n\_ lt **301** then delete; **run**;

**data** nogroup\_log\_final; merge log2 logAIC logAICC logBIC; **run**;

**data** log; set nogroupLine\_log; format Value BEST12. run; /\* Changes sig figs to 4 digits before decimal and 8 after\*/

**proc** **sort** data=log; by Descr; **run**;

**data** log2 (rename=(Value=Log2) drop=descr); set log; if \_n\_ gt **100** then delete; **run**;

**data** logAIC (rename=(Value=AIC) drop=descr); set log; if \_n\_ gt **200** then delete; if \_n\_ lt **101** then delete; **run**;

**data** logAICC (rename=(Value=AICC) drop=descr); set log; if \_n\_ gt **300** then delete; if \_n\_ lt **201** then delete; **run**;

**data** logBIC (rename=(Value=BIC) drop=descr); set log; if \_n\_ lt **301** then delete; **run**;

**data** nogroupLine\_log\_final; merge log2 logAIC logAICC logBIC; **run**;

**data** log; set nogroupMouse\_log; format Value BEST12.; **run**; /\* Changes sig figs to 4 digits before decimal and 8 after\*/

**proc** **sort** data=log; by Descr; **run**;

**data** log2 (rename=(Value=Log2) drop=descr); set log; if \_n\_ gt **100** then delete; **run**;

**data** logAIC (rename=(Value=AIC) drop=descr); set log; if \_n\_ gt **200** then delete; if \_n\_ lt **101** then delete; **run**;

**data** logAICC (rename=(Value=AICC) drop=descr); set log; if \_n\_ gt **300** then delete; if \_n\_ lt **201** then delete; **run**;

**data** logBIC (rename=(Value=BIC) drop=descr); set log; if \_n\_ lt **301** then delete; **run**;

**data** nogroupMouse\_log\_final; merge log2 logAIC logAICC logBIC; **run**;

/\* Identify file names\*/

filename outa "&dir\_out\Results\_full\_tests.csv";

filename outb "&dir\_out\Results\_nogroup\_tests.csv";

filename outc "&dir\_out\Results\_nogroupLine\_tests.csv";

filename outd "&dir\_out\Results\_nogroupMouse\_tests.csv";

filename oute "&dir\_out\Results\_full\_log.csv";

filename outf "&dir\_out\Results\_nogroup\_log.csv";

filename outg "&dir\_out\Results\_nogroupLine\_log.csv";

filename outh "&dir\_out\Results\_nogroupMouse\_log.csv";

/\* Write out files\*/

**proc** **export** data=full\_tests1 outfile=outa dbms=csv replace;

**proc** **export** data=nogroup\_tests1 outfile=outb dbms=csv replace;

**proc** **export** data=nogroupLine\_tests1 outfile=outc dbms=csv replace;

**proc** **export** data=nogroupMouse\_tests1 outfile=outd dbms=csv replace;

**proc** **export** data=full\_log\_final outfile=oute dbms=csv replace;

**proc** **export** data=nogroup\_log\_final outfile=outf dbms=csv replace;

**proc** **export** data=nogroupLine\_log\_final outfile=outg dbms=csv replace;

**proc** **export** data=nogroupMouse\_log\_final outfile=outh dbms=csv replace;

**Run**;

/\* Stop timer\*/

**data** \_null\_;

dur = datetime() - &\_timer\_start;

put **30**\*'-' / ' TOTAL DURATION:' dur time13.2 / **30**\*'-';

**run**;

Haplotype SAS Code

/\* This code is for haplotype analyses and requires that the R code for recoding the haplotype file (File S5.csv) be run first\*/

/\* This is not the most efficient way to analyze these data, but prioritizes being similar to WGS analyses code\*/

%let \_timer\_start = %sysfunc(datetime()); /\* Start timer, for total runtime of program \*/

/\* The following 4 lines change options in SAS to not save unnecessary data\*/

options nonotes; /\* Turns off blue notes in Log so that it doesn't need to be cleared periodically\*/

ods noresults; /\* SAS Support - no results accumulating in the results tree\*/

ods \_all\_ close; /\* SAS Support - no output\*/

ods graphics off; /\* SAS Support - no ods graphics\*/

%let dir\_in=C:\Users\david\Desktop\Garland\Manuscript1\Sample\_Analyses; /\*Set working directory for files being imported\*/

%let dir\_out=C:\Users\david\Desktop\Garland\Manuscript1\Sample\_Analyses\haplotype; /\*Set working directory for files being exported\*/

filename ina "&dir\_in\File S3.csv"; /\*Has mouse data, same as WGS analyses\*/

filename inb "&dir\_in\haplotype\_2alleles.csv"; /\*2-allele data\*/

filename inc "&dir\_in\haplotype\_3alleles\_2nd.csv"; /\*3-allele data\*/

filename ind "&dir\_in\haplotype\_3alleles\_3rd.csv"; /\*3-allele data\*/

filename ine "&dir\_in\Tests.csv"; /\*Used to make empty "tests" tables\*/

filename inf "&dir\_in\AIC.csv"; /\*Used to make empty "log" tables\*/

**proc** **import** datafile=ina out=meta dbms=csv replace; /\* Loads mouse data in SAS as "meta"\*/

**proc** **import** datafile=inb out=geno\_2allele dbms=csv replace; /\* Loads 2-allele data in SAS\*/

**proc** **import** datafile=inc out=geno\_3allele\_2nd dbms=csv replace; /\* Loads 3-allele data in SAS\*/

**proc** **import** datafile=ind out=geno\_3allele\_3rd dbms=csv replace; /\* Loads 3-allele data in SAS\*/

**proc** **import** datafile=ine out=test dbms=csv replace; /\* Loads test results in SAS as "test"\*/

**proc** **import** datafile=inf out=aic dbms=csv replace; /\* Loads test results in SAS as "aic"\*/

**run**;

/\* That should take care of loading and initial prep of data\*/

/\* Formating Stuff\*/

**data** test1; length Effect $**9**; set test; format DenDF BEST4. ProbF PVALUE12.;

**data** aic1; length Descr $**25**; set aic;

**run**;

/\* Start Analyses for 2-allele\*/

**proc** **sql**; /\* These are empty tables to be populated in the macro\*/

create table full\_tests like test1;

create table nogroup\_tests like test1;

create table nogroupLine\_tests like test1;

create table nogroupMouse\_tests like test1;

create table full\_log like aic1;

create table nogroup\_log like aic1;

create table nogroupLine\_log like aic1;

create table nogroupMouse\_log like aic1;

**quit**;

**data** geno1 (drop=chr block n\_hap Nblock\_g61 MinPOS MaxPOS nhap\_g61 haps\_g61); set geno\_2allele; **run**; /\* Drops the unnecessary columns\*/

**proc** **transpose** data=geno1 out=geno2; **run**; /\* Transposes the newly merge dataset\*/

%let n=0; **run**;

/\* This is the macro which is necessary for running procedures in do loops\*/

/\* Still work in progress\*/

**%macro** ***allModels***; /\* "allModels" is the name of the macro\*/

%do k = **1** %to **11032**; /\* This should always start on 1 and end on however many loci are in the chromosome\*/

%let n=%eval(&n+1); /\* This will cause the program to count from one marker to the next\*/

data allele; /\* This identifies the correct locus and drop marker\*/

set geno1;

where marker=&n;

data alleleNoM (drop=marker);

set allele;

proc transpose data=alleleNoM out=allele1; /\* This transposes the one locus to be merged with the mouse data\*/

data locus; merge meta allele1;

proc mixed data=locus method=mivque0; /\* Full model\*/

class pop sub mouse; /\* Identifies population and subpopulation in meta data and is shuffled with mouseID\*/

model COL1=pop/solution; /\* Describes model\*/

random sub(pop) /group=pop; /\* Sets random effects, nesting, and groups for analysis parameters\*/

random mouse(sub pop) /group = pop; /\* Sets random effects, nesting, and groups for analysis parameters\*/

ods output CovParms=parms Tests3=type3 FitStatistics=log; /\* Identifies output table names\*/

proc datasets;

append base=full\_tests data=type3 force; /\* Appends the p-values\*/

append base=full\_log data=log force; /\* Appends the AIC\*/

append base=full\_parms data=parms force; /\* Appends the CovParms\*/

proc mixed data=locus method=mivque0; /\* nogroup model\*/

class pop sub mouse; /\* Identifies population and subpopulation in meta data and is shuffled with mouseID\*/

model COL1=pop/solution; /\* Describes model\*/

random sub(pop); /\* Sets random effects, nesting, and groups for analysis parameters\*/

random mouse(sub pop); /\* Sets random effects, nesting, and groups for analysis parameters\*/

ods output CovParms=parms Tests3=type3 FitStatistics=log; /\* Identifies output table names\*/

proc datasets;

append base=nogroup\_tests data=type3 force; /\* Appends the p-values\*/

append base=nogroup\_log data=log force; /\* Appends the AIC\*/

append base=nogroup\_parms data=parms force; /\* Appends the CovParms\*/

proc mixed data=locus method=mivque0; /\* nogroupLine model\*/

class pop sub mouse; /\* Identifies population and subpopulation in meta data and is shuffled with mouseID\*/

model COL1=pop/solution; /\* Describes model\*/

random sub(pop); /\* Sets random effects, nesting, and groups for analysis parameters\*/

random mouse(sub pop) /group = pop; /\* Sets random effects, nesting, and groups for analysis parameters\*/

ods output CovParms=parms Tests3=type3 FitStatistics=log; /\* Identifies output table names\*/

proc datasets;

append base=nogroupLine\_tests data=type3 force; /\* Appends the p-values\*/

append base=nogroupLine\_log data=log force; /\* Appends the AIC\*/

append base=nogroupLine\_parms data=parms force; /\* Appends the CovParms\*/

proc mixed data=locus method=mivque0; /\* nogroupMouse model\*/

class pop sub mouse; /\* Identifies population and subpopulation in meta data and is shuffled with mouseID\*/

model COL1=pop/solution; /\* Describes model\*/

random sub(pop) /group = pop; /\* Sets random effects, nesting, and groups for analysis parameters\*/

random mouse(sub pop); /\* Sets random effects, nesting, and groups for analysis parameters\*/

ods output CovParms=parms Tests3=type3 FitStatistics=log; /\* Identifies output table names\*/

proc datasets;

append base=nogroupMouse\_tests data=type3 force; /\* Appends the p-values\*/

append base=nogroupMouse\_log data=log force; /\* Appends the AIC\*/

append base=nogroupMouse\_parms data=parms force; /\* Appends the CovParms\*/

data \_null\_; /\* Counts off every 50 loci analyzed\*/

y = &n;

z=mod(y,**50**);

if z=**0** then do;

put y;

end;

%end;

**%mend** allModels; /\* mend ends the macro, differing notes on whether the name needs to be included\*/

/\* This actually runs the analysis\*/

%***allModels***;

**run**;

/\* Organize parms has been removed, it would require modified code for each model and we are currenly not using it

These files can be modified in R after running SAS, if needed\*/

/\* This changes all p-values to 10 decimal places until stated otherwise\*/

**data** full\_tests1; set full\_tests; format ProbF pvalue12.10; **run**;

**data** nogroup\_tests1; set nogroup\_tests; format ProbF pvalue12.10; **run**;

**data** nogroupLine\_tests1; set nogroupLine\_tests; format ProbF pvalue12.10; **run**;

**data** nogroupMouse\_tests1; set nogroupMouse\_tests; format ProbF pvalue12.10; **run**;

/\* Organize log\*/

**data** log; set full\_log; format Value BEST12.; **run**; /\* Changes sig figs to most informative 12 digits\*/

**proc** **sort** data=log; by Descr; **run**;

**data** log2 (rename=(Value=Log2) drop=descr); set log; if \_n\_ gt **11032** then delete; **run**;

**data** logAIC (rename=(Value=AIC) drop=marker descr); set log; if \_n\_ gt **22064** then delete; if \_n\_ lt **11033** then delete; **run**;

**data** logAICC (rename=(Value=AICC) drop=marker descr); set log; if \_n\_ gt **33096** then delete; if \_n\_ lt **22065** then delete; **run**;

**data** logBIC (rename=(Value=BIC) drop=marker descr); set log; if \_n\_ lt **33097** then delete; **run**;

**data** full\_log\_final; merge log2 logAIC logAICC logBIC; **run**;

**data** log; set nogroup\_log; format Value BEST12.; **run**; /\* Changes sig figs to most informative 12 digits\*/

**proc** **sort** data=log; by Descr; **run**;

**data** log2 (rename=(Value=Log2) drop=descr); set log; if \_n\_ gt **11032** then delete; **run**;

**data** logAIC (rename=(Value=AIC) drop=marker descr); set log; if \_n\_ gt **22064** then delete; if \_n\_ lt **11033** then delete; **run**;

**data** logAICC (rename=(Value=AICC) drop=marker descr); set log; if \_n\_ gt **33096** then delete; if \_n\_ lt **22065** then delete; **run**;

**data** logBIC (rename=(Value=BIC) drop=marker descr); set log; if \_n\_ lt **33097** then delete; **run**;

**data** nogroup\_log\_final; merge log2 logAIC logAICC logBIC; **run**;

**data** log; set nogroupLine\_log; format Value BEST12. run; /\* Changes sig figs to most informative 12 digits\*/

**proc** **sort** data=log; by Descr; **run**;

**data** log2 (rename=(Value=Log2) drop=descr); set log; if \_n\_ gt **11032** then delete; **run**;

**data** logAIC (rename=(Value=AIC) drop=marker descr); set log; if \_n\_ gt **22064** then delete; if \_n\_ lt **11033** then delete; **run**;

**data** logAICC (rename=(Value=AICC) drop=marker descr); set log; if \_n\_ gt **33096** then delete; if \_n\_ lt **22065** then delete; **run**;

**data** logBIC (rename=(Value=BIC) drop=marker descr); set log; if \_n\_ lt **33097** then delete; **run**;

**data** nogroupLine\_log\_final; merge log2 logAIC logAICC logBIC; **run**;

**data** log; set nogroupMouse\_log; format Value BEST12.; **run**; /\* Changes sig figs to most informative 12 digits\*/

**proc** **sort** data=log; by Descr; **run**;

**data** log2 (rename=(Value=Log2) drop=descr); set log; if \_n\_ gt **11032** then delete; **run**;

**data** logAIC (rename=(Value=AIC) drop=marker descr); set log; if \_n\_ gt **22064** then delete; if \_n\_ lt **11033** then delete; **run**;

**data** logAICC (rename=(Value=AICC) drop=marker descr); set log; if \_n\_ gt **33096** then delete; if \_n\_ lt **22065** then delete; **run**;

**data** logBIC (rename=(Value=BIC) drop=marker descr); set log; if \_n\_ lt **33097** then delete; **run**;

**data** nogroupMouse\_log\_final; merge log2 logAIC logAICC logBIC; **run**;

filename outa "&dir\_out\haplotype\_2alleles\_full\_tests.csv";

filename outb "&dir\_out\haplotype\_2alleles\_nogroup\_tests.csv";

filename outc "&dir\_out\haplotype\_2alleles\_nogroupLine\_tests.csv";

filename outd "&dir\_out\haplotype\_2alleles\_nogroupMouse\_tests.csv";

filename oute "&dir\_out\haplotype\_2alleles\_full\_log.csv";

filename outf "&dir\_out\haplotype\_2alleles\_nogroup\_log.csv";

filename outg "&dir\_out\haplotype\_2alleles\_nogroupLine\_log.csv";

filename outh "&dir\_out\haplotype\_2alleles\_nogroupMouse\_log.csv";

**proc** **export** data=full\_tests1 outfile=outa dbms=csv replace;

**proc** **export** data=nogroup\_tests1 outfile=outb dbms=csv replace;

**proc** **export** data=nogroupLine\_tests1 outfile=outc dbms=csv replace;

**proc** **export** data=nogroupMouse\_tests1 outfile=outd dbms=csv replace;

**proc** **export** data=full\_log\_final outfile=oute dbms=csv replace;

**proc** **export** data=nogroup\_log\_final outfile=outf dbms=csv replace;

**proc** **export** data=nogroupLine\_log\_final outfile=outg dbms=csv replace;

**proc** **export** data=nogroupMouse\_log\_final outfile=outh dbms=csv replace;

**Run**;

/\* Start Analyses for 3-allele (2nd)\*/

**proc** **sql**; /\* These are empty tables to be populated in the macro\*/

create table full\_tests like test1;

create table nogroup\_tests like test1;

create table nogroupLine\_tests like test1;

create table nogroupMouse\_tests like test1;

create table full\_log like aic1;

create table nogroup\_log like aic1;

create table nogroupLine\_log like aic1;

create table nogroupMouse\_log like aic1;

**quit**;

**data** geno1 (drop=chr block n\_hap Nblock\_g61 MinPOS MaxPOS nhap\_g61 haps\_g61); set geno\_3allele\_2nd; **run**; /\* Drops the unnecessary columns\*/

**proc** **transpose** data=geno1 out=geno2; **run**; /\* Transposes the newly merge dataset\*/

%let n=0; **run**;

/\* This is the macro which is necessary for running procedures in do loops\*/

/\* Still work in progress\*/

**%macro** ***allModels***; /\* "allModels" is the name of the macro\*/

%do k = **1** %to **5869**; /\* This should always start on 1 and end on however many loci are in the chromosome\*/

%let n=%eval(&n+1); /\* This will cause the program to count from one marker to the next\*/

data allele; /\* This identifies the correct locus and drop marker\*/

set geno1;

where marker=&n;

data alleleNoM (drop=marker);

set allele;

proc transpose data=alleleNoM out=allele1; /\* This transposes the one locus to be merged with the mouse data\*/

data locus; merge meta allele1;

proc mixed data=locus method=mivque0; /\* Full model\*/

class pop sub mouse; /\* Identifies population and subpopulation in meta data and is shuffled with mouseID\*/

model COL1=pop/solution; /\* Describes model\*/

random sub(pop) /group=pop; /\* Sets random effects, nesting, and groups for analysis parameters\*/

random mouse(sub pop) /group = pop; /\* Sets random effects, nesting, and groups for analysis parameters\*/

ods output CovParms=parms Tests3=type3 FitStatistics=log; /\* Identifies output table names\*/

proc datasets;

append base=full\_tests data=type3 force; /\* Appends the p-values\*/

append base=full\_log data=log force; /\* Appends the AIC\*/

append base=full\_parms data=parms force; /\* Appends the CovParms\*/

proc mixed data=locus method=mivque0; /\* nogroup model\*/

class pop sub mouse; /\* Identifies population and subpopulation in meta data and is shuffled with mouseID\*/

model COL1=pop/solution; /\* Describes model\*/

random sub(pop); /\* Sets random effects, nesting, and groups for analysis parameters\*/

random mouse(sub pop); /\* Sets random effects, nesting, and groups for analysis parameters\*/

ods output CovParms=parms Tests3=type3 FitStatistics=log; /\* Identifies output table names\*/

proc datasets;

append base=nogroup\_tests data=type3 force; /\* Appends the p-values\*/

append base=nogroup\_log data=log force; /\* Appends the AIC\*/

append base=nogroup\_parms data=parms force; /\* Appends the CovParms\*/

proc mixed data=locus method=mivque0; /\* nogroupLine model\*/

class pop sub mouse; /\* Identifies population and subpopulation in meta data and is shuffled with mouseID\*/

model COL1=pop/solution; /\* Describes model\*/

random sub(pop); /\* Sets random effects, nesting, and groups for analysis parameters\*/

random mouse(sub pop) /group = pop; /\* Sets random effects, nesting, and groups for analysis parameters\*/

ods output CovParms=parms Tests3=type3 FitStatistics=log; /\* Identifies output table names\*/

proc datasets;

append base=nogroupLine\_tests data=type3 force; /\* Appends the p-values\*/

append base=nogroupLine\_log data=log force; /\* Appends the AIC\*/

append base=nogroupLine\_parms data=parms force; /\* Appends the CovParms\*/

proc mixed data=locus method=mivque0; /\* nogroupMouse model\*/

class pop sub mouse; /\* Identifies population and subpopulation in meta data and is shuffled with mouseID\*/

model COL1=pop/solution; /\* Describes model\*/

random sub(pop) /group = pop; /\* Sets random effects, nesting, and groups for analysis parameters\*/

random mouse(sub pop); /\* Sets random effects, nesting, and groups for analysis parameters\*/

ods output CovParms=parms Tests3=type3 FitStatistics=log; /\* Identifies output table names\*/

proc datasets;

append base=nogroupMouse\_tests data=type3 force; /\* Appends the p-values\*/

append base=nogroupMouse\_log data=log force; /\* Appends the AIC\*/

append base=nogroupMouse\_parms data=parms force; /\* Appends the CovParms\*/

data \_null\_; /\* Counts off every 50 loci analyzed\*/

y = &n;

z=mod(y,**50**);

if z=**0** then do;

put y;

end;

%end;

**%mend** allModels; /\* mend ends the macro, differing notes on whether the name needs to be included\*/

/\* This actually runs the analysis\*/

%***allModels***;

**run**;

/\* Organize parms has been removed, it would require modified code for each model and we are currenly not using it

These files can be modified in R after running SAS, if needed\*/

/\* This changes all p-values to 10 decimal places until stated otherwise\*/

**data** full\_tests1; set full\_tests; format ProbF pvalue12.10; **run**;

**data** nogroup\_tests1; set nogroup\_tests; format ProbF pvalue12.10; **run**;

**data** nogroupLine\_tests1; set nogroupLine\_tests; format ProbF pvalue12.10; **run**;

**data** nogroupMouse\_tests1; set nogroupMouse\_tests; format ProbF pvalue12.10; **run**;

/\* Organize log\*/

**data** log; set full\_log; format Value BEST12.; **run**; /\* Changes sig figs to most informative 12 digits\*/

**proc** **sort** data=log; by Descr; **run**;

**data** log2 (rename=(Value=Log2) drop=descr); set log; if \_n\_ gt **5869** then delete; **run**;

**data** logAIC (rename=(Value=AIC) drop=marker descr); set log; if \_n\_ gt **11738** then delete; if \_n\_ lt **5870** then delete; **run**;

**data** logAICC (rename=(Value=AICC) drop=marker descr); set log; if \_n\_ gt **17607** then delete; if \_n\_ lt **11739** then delete; **run**;

**data** logBIC (rename=(Value=BIC) drop=marker descr); set log; if \_n\_ lt **17608** then delete; **run**;

**data** full\_log\_final; merge log2 logAIC logAICC logBIC; **run**;

**data** log; set nogroup\_log; format Value BEST12.; **run**; /\* Changes sig figs to most informative 12 digits\*/

**proc** **sort** data=log; by Descr; **run**;

**data** log2 (rename=(Value=Log2) drop=descr); set log; if \_n\_ gt **5869** then delete; **run**;

**data** logAIC (rename=(Value=AIC) drop=marker descr); set log; if \_n\_ gt **11738** then delete; if \_n\_ lt **5870** then delete; **run**;

**data** logAICC (rename=(Value=AICC) drop=marker descr); set log; if \_n\_ gt **17607** then delete; if \_n\_ lt **11739** then delete; **run**;

**data** logBIC (rename=(Value=BIC) drop=marker descr); set log; if \_n\_ lt **17608** then delete; **run**;

**data** nogroup\_log\_final; merge log2 logAIC logAICC logBIC; **run**;

**data** log; set nogroupLine\_log; format Value BEST12. run; /\* Changes sig figs to most informative 12 digits\*/

**proc** **sort** data=log; by Descr; **run**;

**data** log2 (rename=(Value=Log2) drop=descr); set log; if \_n\_ gt **5869** then delete; **run**;

**data** logAIC (rename=(Value=AIC) drop=marker descr); set log; if \_n\_ gt **11738** then delete; if \_n\_ lt **5870** then delete; **run**;

**data** logAICC (rename=(Value=AICC) drop=marker descr); set log; if \_n\_ gt **17607** then delete; if \_n\_ lt **11739** then delete; **run**;

**data** logBIC (rename=(Value=BIC) drop=marker descr); set log; if \_n\_ lt **17608** then delete; **run**;

**data** nogroupLine\_log\_final; merge log2 logAIC logAICC logBIC; **run**;

**data** log; set nogroupMouse\_log; format Value BEST12.; **run**; /\* Changes sig figs to most informative 12 digits\*/

**proc** **sort** data=log; by Descr; **run**;

**data** log2 (rename=(Value=Log2) drop=descr); set log; if \_n\_ gt **5869** then delete; **run**;

**data** logAIC (rename=(Value=AIC) drop=marker descr); set log; if \_n\_ gt **11738** then delete; if \_n\_ lt **5870** then delete; **run**;

**data** logAICC (rename=(Value=AICC) drop=marker descr); set log; if \_n\_ gt **17607** then delete; if \_n\_ lt **11739** then delete; **run**;

**data** logBIC (rename=(Value=BIC) drop=marker descr); set log; if \_n\_ lt **17608** then delete; **run**;

**data** nogroupMouse\_log\_final; merge log2 logAIC logAICC logBIC; **run**;

filename outa "&dir\_out\haplotype\_3alleles\_2nd\_full\_tests.csv";

filename outb "&dir\_out\haplotype\_3alleles\_2nd\_nogroup\_tests.csv";

filename outc "&dir\_out\haplotype\_3alleles\_2nd\_nogroupLine\_tests.csv";

filename outd "&dir\_out\haplotype\_3alleles\_2nd\_nogroupMouse\_tests.csv";

filename oute "&dir\_out\haplotype\_3alleles\_2nd\_full\_log.csv";

filename outf "&dir\_out\haplotype\_3alleles\_2nd\_nogroup\_log.csv";

filename outg "&dir\_out\haplotype\_3alleles\_2nd\_nogroupLine\_log.csv";

filename outh "&dir\_out\haplotype\_3alleles\_2nd\_nogroupMouse\_log.csv";

**proc** **export** data=full\_tests1 outfile=outa dbms=csv replace;

**proc** **export** data=nogroup\_tests1 outfile=outb dbms=csv replace;

**proc** **export** data=nogroupLine\_tests1 outfile=outc dbms=csv replace;

**proc** **export** data=nogroupMouse\_tests1 outfile=outd dbms=csv replace;

**proc** **export** data=full\_log\_final outfile=oute dbms=csv replace;

**proc** **export** data=nogroup\_log\_final outfile=outf dbms=csv replace;

**proc** **export** data=nogroupLine\_log\_final outfile=outg dbms=csv replace;

**proc** **export** data=nogroupMouse\_log\_final outfile=outh dbms=csv replace;

**Run**;

/\* Start Analyses for 3-allele (3rd)\*/

**proc** **sql**; /\* These are empty tables to be populated in the macro\*/

create table full\_tests like test1;

create table nogroup\_tests like test1;

create table nogroupLine\_tests like test1;

create table nogroupMouse\_tests like test1;

create table full\_log like aic1;

create table nogroup\_log like aic1;

create table nogroupLine\_log like aic1;

create table nogroupMouse\_log like aic1;

**quit**;

**data** geno1 (drop=chr block n\_hap Nblock\_g61 MinPOS MaxPOS nhap\_g61 haps\_g61); set geno\_3allele\_3rd; **run**; /\* Drops the unnecessary columns\*/

**proc** **transpose** data=geno1 out=geno2; **run**; /\* Transposes the newly merge dataset\*/

%let n=0; **run**;

/\* This is the macro which is necessary for running procedures in do loops\*/

/\* Still work in progress\*/

**%macro** ***allModels***; /\* "allModels" is the name of the macro\*/

%do k = **1** %to **5869**; /\* This should always start on 1 and end on however many loci are in the chromosome\*/

%let n=%eval(&n+1); /\* This will cause the program to count from one marker to the next\*/

data allele; /\* This identifies the correct locus and drop marker\*/

set geno1;

where marker=&n;

data alleleNoM (drop=marker);

set allele;

proc transpose data=alleleNoM out=allele1; /\* This transposes the one locus to be merged with the mouse data\*/

data locus; merge meta allele1;

proc mixed data=locus method=mivque0; /\* Full model\*/

class pop sub mouse; /\* Identifies population and subpopulation in meta data and is shuffled with mouseID\*/

model COL1=pop/solution; /\* Describes model\*/

random sub(pop) /group=pop; /\* Sets random effects, nesting, and groups for analysis parameters\*/

random mouse(sub pop) /group = pop; /\* Sets random effects, nesting, and groups for analysis parameters\*/

ods output CovParms=parms Tests3=type3 FitStatistics=log; /\* Identifies output table names\*/

proc datasets;

append base=full\_tests data=type3 force; /\* Appends the p-values\*/

append base=full\_log data=log force; /\* Appends the AIC\*/

append base=full\_parms data=parms force; /\* Appends the CovParms\*/

proc mixed data=locus method=mivque0; /\* nogroup model\*/

class pop sub mouse; /\* Identifies population and subpopulation in meta data and is shuffled with mouseID\*/

model COL1=pop/solution; /\* Describes model\*/

random sub(pop); /\* Sets random effects, nesting, and groups for analysis parameters\*/

random mouse(sub pop); /\* Sets random effects, nesting, and groups for analysis parameters\*/

ods output CovParms=parms Tests3=type3 FitStatistics=log; /\* Identifies output table names\*/

proc datasets;

append base=nogroup\_tests data=type3 force; /\* Appends the p-values\*/

append base=nogroup\_log data=log force; /\* Appends the AIC\*/

append base=nogroup\_parms data=parms force; /\* Appends the CovParms\*/

proc mixed data=locus method=mivque0; /\* nogroupLine model\*/

class pop sub mouse; /\* Identifies population and subpopulation in meta data and is shuffled with mouseID\*/

model COL1=pop/solution; /\* Describes model\*/

random sub(pop); /\* Sets random effects, nesting, and groups for analysis parameters\*/

random mouse(sub pop) /group = pop; /\* Sets random effects, nesting, and groups for analysis parameters\*/

ods output CovParms=parms Tests3=type3 FitStatistics=log; /\* Identifies output table names\*/

proc datasets;

append base=nogroupLine\_tests data=type3 force; /\* Appends the p-values\*/

append base=nogroupLine\_log data=log force; /\* Appends the AIC\*/

append base=nogroupLine\_parms data=parms force; /\* Appends the CovParms\*/

proc mixed data=locus method=mivque0; /\* nogroupMouse model\*/

class pop sub mouse; /\* Identifies population and subpopulation in meta data and is shuffled with mouseID\*/

model COL1=pop/solution; /\* Describes model\*/

random sub(pop) /group = pop; /\* Sets random effects, nesting, and groups for analysis parameters\*/

random mouse(sub pop); /\* Sets random effects, nesting, and groups for analysis parameters\*/

ods output CovParms=parms Tests3=type3 FitStatistics=log; /\* Identifies output table names\*/

proc datasets;

append base=nogroupMouse\_tests data=type3 force; /\* Appends the p-values\*/

append base=nogroupMouse\_log data=log force; /\* Appends the AIC\*/

append base=nogroupMouse\_parms data=parms force; /\* Appends the CovParms\*/

data \_null\_; /\* Counts off every 50 loci analyzed\*/

y = &n;

z=mod(y,**50**);

if z=**0** then do;

put y;

end;

%end;

**%mend** allModels; /\* mend ends the macro, differing notes on whether the name needs to be included\*/

/\* This actually runs the analysis\*/

%***allModels***;

**run**;

/\* Organize parms has been removed, it would require modified code for each model and we are currenly not using it

These files can be modified in R after running SAS, if needed\*/

/\* This changes all p-values to 10 decimal places until stated otherwise\*/

**data** full\_tests1; set full\_tests; format ProbF pvalue12.10; **run**;

**data** nogroup\_tests1; set nogroup\_tests; format ProbF pvalue12.10; **run**;

**data** nogroupLine\_tests1; set nogroupLine\_tests; format ProbF pvalue12.10; **run**;

**data** nogroupMouse\_tests1; set nogroupMouse\_tests; format ProbF pvalue12.10; **run**;

/\* Organize log\*/

**data** log; set full\_log; format Value BEST12.; **run**; /\* Changes sig figs to most informative 12 digits\*/

**proc** **sort** data=log; by Descr; **run**;

**data** log2 (rename=(Value=Log2) drop=descr); set log; if \_n\_ gt **5869** then delete; **run**;

**data** logAIC (rename=(Value=AIC) drop=marker descr); set log; if \_n\_ gt **11738** then delete; if \_n\_ lt **5870** then delete; **run**;

**data** logAICC (rename=(Value=AICC) drop=marker descr); set log; if \_n\_ gt **17607** then delete; if \_n\_ lt **11739** then delete; **run**;

**data** logBIC (rename=(Value=BIC) drop=marker descr); set log; if \_n\_ lt **17608** then delete; **run**;

**data** full\_log\_final; merge log2 logAIC logAICC logBIC; **run**;

**data** log; set nogroup\_log; format Value BEST12.; **run**; /\* Changes sig figs to most informative 12 digits\*/

**proc** **sort** data=log; by Descr; **run**;

**data** log2 (rename=(Value=Log2) drop=descr); set log; if \_n\_ gt **5869** then delete; **run**;

**data** logAIC (rename=(Value=AIC) drop=marker descr); set log; if \_n\_ gt **11738** then delete; if \_n\_ lt **5870** then delete; **run**;

**data** logAICC (rename=(Value=AICC) drop=marker descr); set log; if \_n\_ gt **17607** then delete; if \_n\_ lt **11739** then delete; **run**;

**data** logBIC (rename=(Value=BIC) drop=marker descr); set log; if \_n\_ lt **17608** then delete; **run**;

**data** nogroup\_log\_final; merge log2 logAIC logAICC logBIC; **run**;

**data** log; set nogroupLine\_log; format Value BEST12. run; /\* Changes sig figs to most informative 12 digits\*/

**proc** **sort** data=log; by Descr; **run**;

**data** log2 (rename=(Value=Log2) drop=descr); set log; if \_n\_ gt **5869** then delete; **run**;

**data** logAIC (rename=(Value=AIC) drop=marker descr); set log; if \_n\_ gt **11738** then delete; if \_n\_ lt **5870** then delete; **run**;

**data** logAICC (rename=(Value=AICC) drop=marker descr); set log; if \_n\_ gt **17607** then delete; if \_n\_ lt **11739** then delete; **run**;

**data** logBIC (rename=(Value=BIC) drop=marker descr); set log; if \_n\_ lt **17608** then delete; **run**;

**data** nogroupLine\_log\_final; merge log2 logAIC logAICC logBIC; **run**;

**data** log; set nogroupMouse\_log; format Value BEST12.; **run**; /\* Changes sig figs to most informative 12 digits\*/

**proc** **sort** data=log; by Descr; **run**;

**data** log2 (rename=(Value=Log2) drop=descr); set log; if \_n\_ gt **5869** then delete; **run**;

**data** logAIC (rename=(Value=AIC) drop=marker descr); set log; if \_n\_ gt **11738** then delete; if \_n\_ lt **5870** then delete; **run**;

**data** logAICC (rename=(Value=AICC) drop=marker descr); set log; if \_n\_ gt **17607** then delete; if \_n\_ lt **11739** then delete; **run**;

**data** logBIC (rename=(Value=BIC) drop=marker descr); set log; if \_n\_ lt **17608** then delete; **run**;

**data** nogroupMouse\_log\_final; merge log2 logAIC logAICC logBIC; **run**;

filename outa "&dir\_out\haplotype\_3alleles\_3rd\_full\_tests.csv";

filename outb "&dir\_out\haplotype\_3alleles\_3rd\_nogroup\_tests.csv";

filename outc "&dir\_out\haplotype\_3alleles\_3rd\_nogroupLine\_tests.csv";

filename outd "&dir\_out\haplotype\_3alleles\_3rd\_nogroupMouse\_tests.csv";

filename oute "&dir\_out\haplotype\_3alleles\_3rd\_full\_log.csv";

filename outf "&dir\_out\haplotype\_3alleles\_3rd\_nogroup\_log.csv";

filename outg "&dir\_out\haplotype\_3alleles\_3rd\_nogroupLine\_log.csv";

filename outh "&dir\_out\haplotype\_3alleles\_3rd\_nogroupMouse\_log.csv";

**proc** **export** data=full\_tests1 outfile=outa dbms=csv replace;

**proc** **export** data=nogroup\_tests1 outfile=outb dbms=csv replace;

**proc** **export** data=nogroupLine\_tests1 outfile=outc dbms=csv replace;

**proc** **export** data=nogroupMouse\_tests1 outfile=outd dbms=csv replace;

**proc** **export** data=full\_log\_final outfile=oute dbms=csv replace;

**proc** **export** data=nogroup\_log\_final outfile=outf dbms=csv replace;

**proc** **export** data=nogroupLine\_log\_final outfile=outg dbms=csv replace;

**proc** **export** data=nogroupMouse\_log\_final outfile=outh dbms=csv replace;

**Run**;

/\* Stop timer\*/

**data** \_null\_;

dur = datetime() - &\_timer\_start;

put **30**\*'-' / ' TOTAL DURATION:' dur time13.2 / **30**\*'-';

**run**;