UM Black Bear Pedigree Inference Guide

2024-08-06

by Dr. Maya Weissman

Setup

Load all packages used:

Input data files:

21/22 File Name	Description
GT_BB2122_nuc.csv	CSV with nDNA snps. Columns correspond to SNPs. 0 = no copies of variant, 1 = heterozygous, 2 = homozygous with variant, -9 = missing read.
2021_CKMR_Sample_Data_age.xlsx	Excel sheet with life history data for samples collected in 2021
2022_CKMR_Sample_Data_age.xlsx	Excel sheet with life history data for samples collected in 2022
BB2122_compiled_R1_genotypes.csv	CSV of raw nDNA. Includes meta data (i.e. IFI) and sequencing of 5 genetic sex markers
BB_2022_001_calls_annotated_maf0.1_sequoia.vcf	VCF (variant call format) file with mitochondrial DNA for all individuals

R files:

R File	Description
bb_functions_datafiltering.R	Custom built data filtering functions
bb_nDNA_qcfilter.R	Filters samples based on nuclear DNA
bb_mtDNA_qcfilter.R	Filters and analyzes mitochondrial DNA
bb_functions_popgen.R	Custom built functions for popgen analyses
bb_popgen_nuc.R	Performs popgen analyses on nuclear DNA
bb_mtDNA_popgen.R	Performs popgen analyses on mitochondrial DNA
bb_mDNA_haplotypes.R.	Assigns mitochondrial haplotypes and analyzes haplotype network. *not working right now*
bb_pedigree_diagnostics.R	Uses CKMRsim package to test inference power of nuclear DNA in resolving pedigree
bb_functions_pedigree_pairs.R	Custom built functions for analyzing sequoia pedigrees
bb_sequoia_pedigree_inference.R	Creates pedigree using sequoia package

Output Files:

Output File	Description
QCBears_fulldata_final.csv	Pruned QC bears, snps, and life history data
AllBears_fulldata_final.csv	All bears, with SNPS and life history
nuclear_popgen_clusters.csv	QC bear dataframe, plus columns for PC and k-means cluster data based on nuclear DNA
nuc_gdist_fst.csv	CSV of genetic distance (Fst) between Harvest BMUs
nuc_popgen_ibd.csv	CSV of pairwise genetic distances and geographic distances for all individuals.
QCBears_mDNA.csv	Pruned mitochondrial DNA snps for QC bears
mito_popgen_clusters.csv	QC bear dataframe, plus columns for PC and k-means cluster data based on mitochondrial DNA
CKMRsim_error_rates.csv	False positive and false negative error rates found using simulated genotypes to test inference power of nuclear DNA
seqped_ap_po.rds	R Data object of full pedigree, with age priors and PO pairs only
seqped_ap_full.rds	R Data object of full pedigree, with age priors and full pedigree
seqped_noap_po.rds	R Data object of full pedigree, with no age priors and PO pairs only
seqped_noap_full.rds	R Data object of full pedigree, with no age priors and full pedigree
BB_POpairs_final.csv	CSV of all PO pairs, with LH data. **File used in CKMR model.**

Step 1: Data filtering

1.1 Nuclear DNA

1.1.2 Filtering Run the bb_nDNA_qcfilter.R file. In summary, this file does several quality control steps:

1. Per individual nuclear SNP coverage: Individuals must have reads at over half of SNPs.
2. Genetic sex: We need to be confident in the genetic sex of individuals to construct a pedigree. Therefore, at least 3/5 genetic sex markers must agree on the sex.
3. IFI: IFI is a measure of noisiness indicative of mixed samples. The lower the number the better. We include only individuals with an IFI < 2.
4. Loci must have reads for more than half of individuals

You will get two output files:

- QCBears_fulldata_final.csv, which contains only bears and SNPs that passed all quality control steps.
- AllBears_fulldata_final.csv, which includes the full combined dataframe of all individuals.

Let's first look at a summary of the Full Data file, to see how filtering steps went.

```r
#Load libraries
library(ggplot2)
library(tidyverse)
```

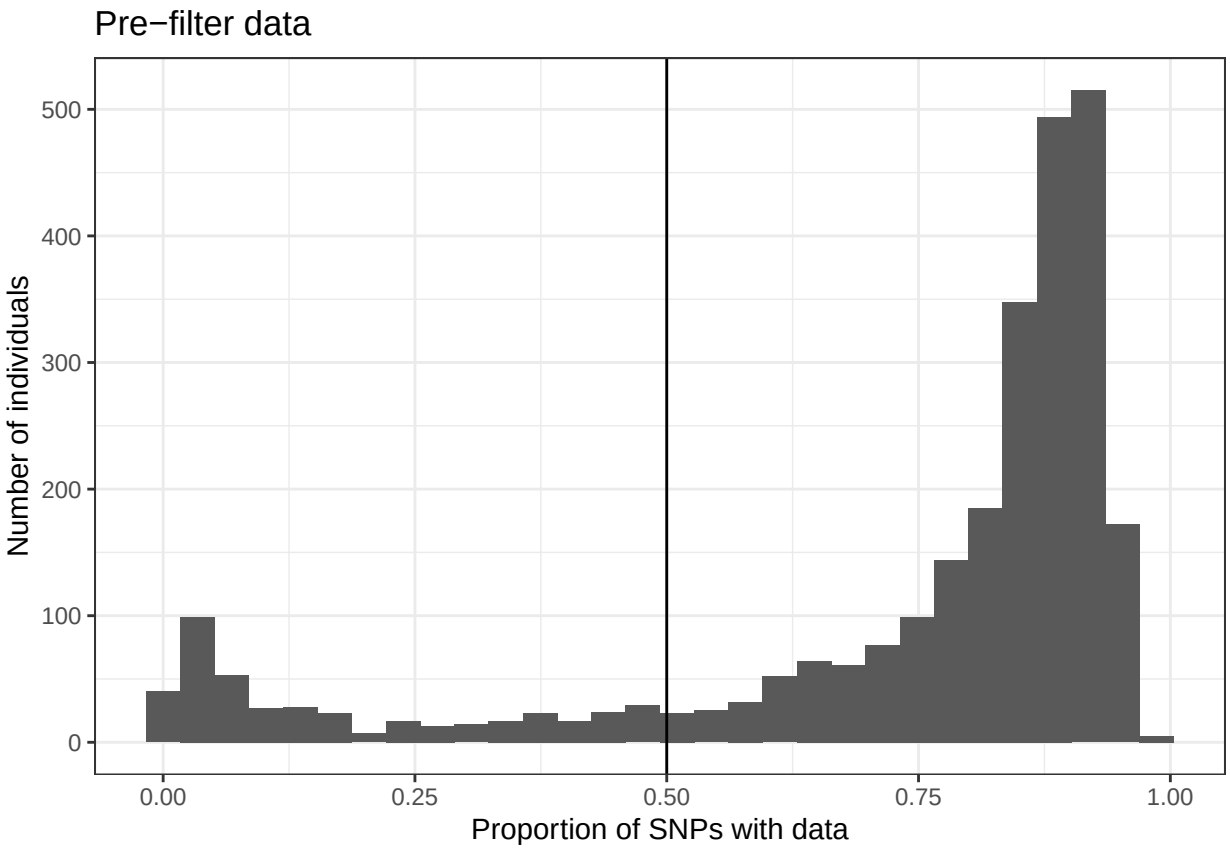
```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.0
## v lubridate 1.9.2     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(viridis)
```

```
## Loading required package: viridisLite
```

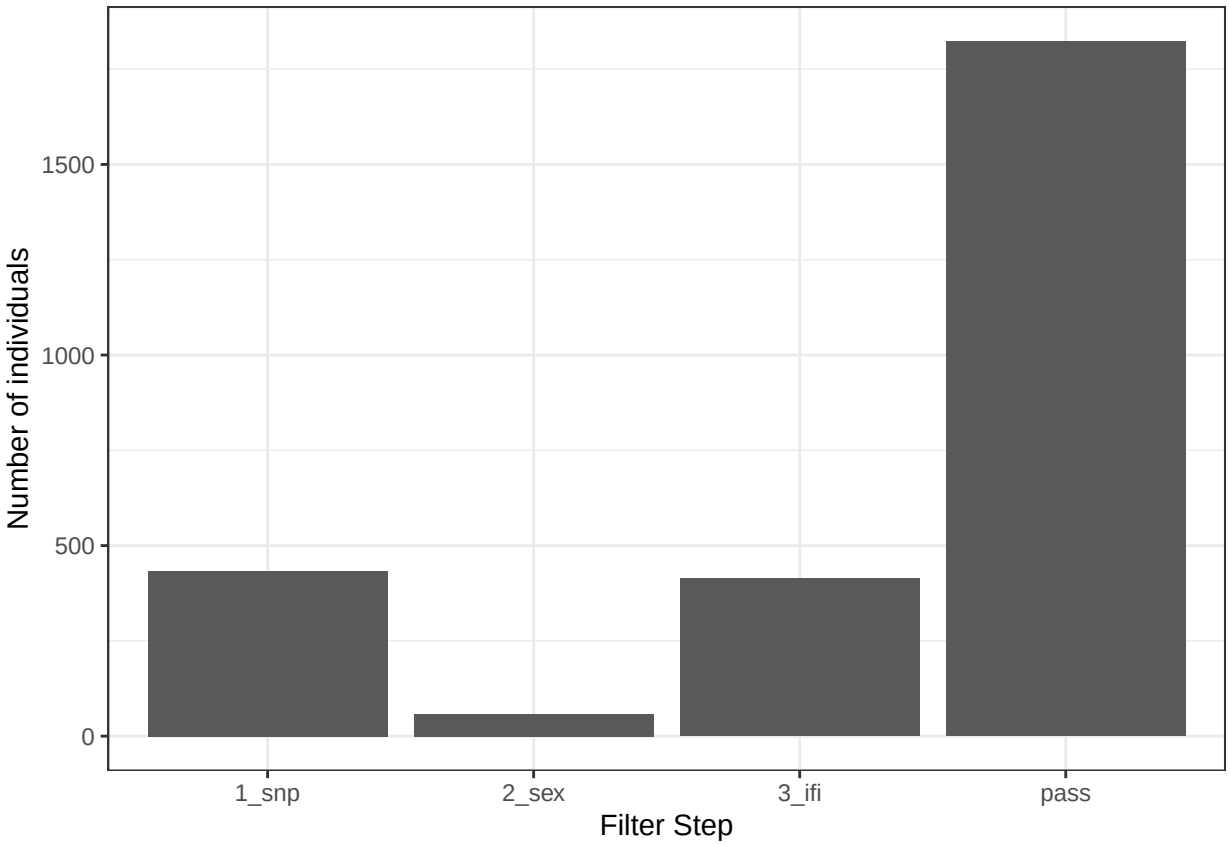
```r
#Load dataframe
full_data <- read.csv("AllBears_fulldata_final.csv")

ggplot(full_data) + geom_histogram(aes(x = prop_SNPs)) +
  theme_bw() +
  xlab("Proportion of SNPs with data") +
  ylab("Number of individuals") +
  ggtitle("Pre-filter data") +
  geom_vline(xintercept = c(0.5))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



The majority of SNPs have coverage for over half of individuals, which means that the DNA quality is really good. Next, let's look at how many individuals we lose at each stage:

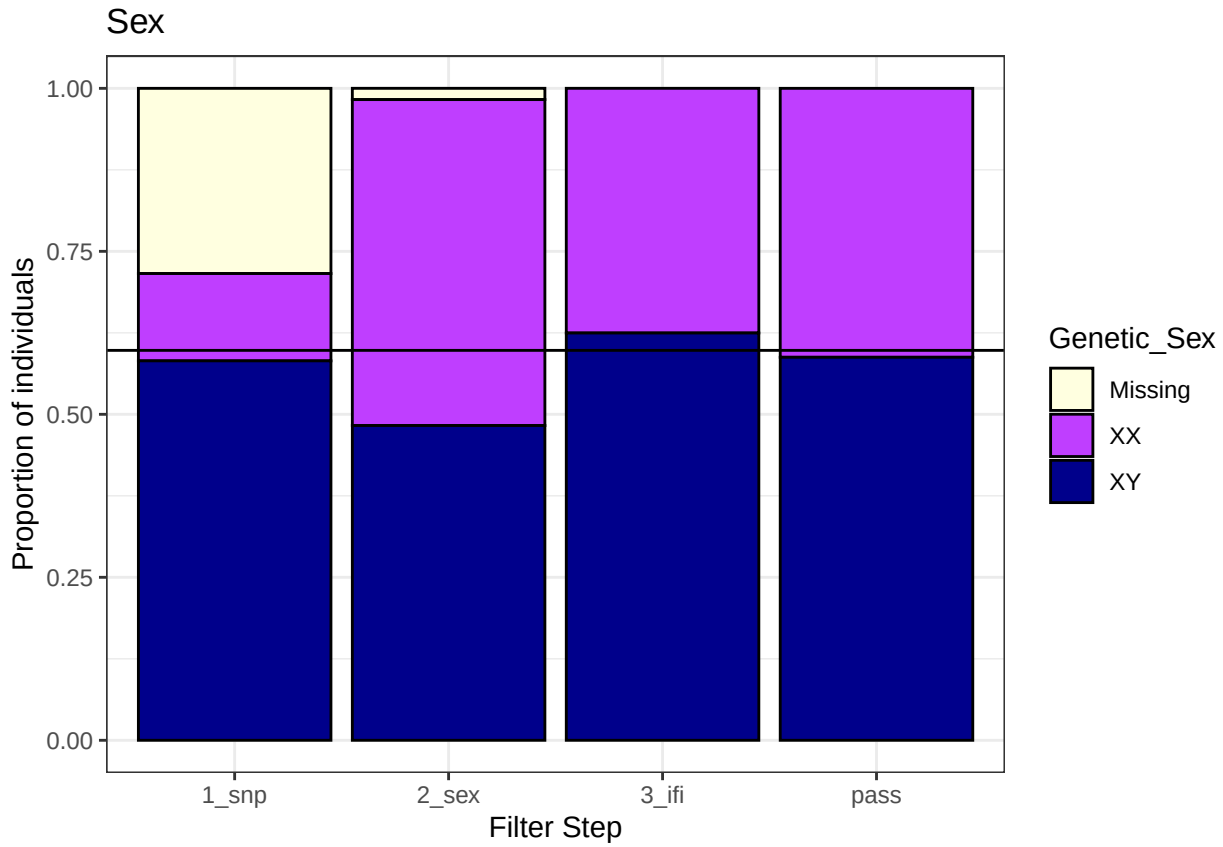
```
ggplot(data = full_data) + geom_bar(aes(x =filter_remove)) + theme_bw() +
  ylab("Number of individuals") +
  xlab("Filter Step")
```



We also need to make sure that filtering steps didn't introduce biases in any of our life history traits, like:

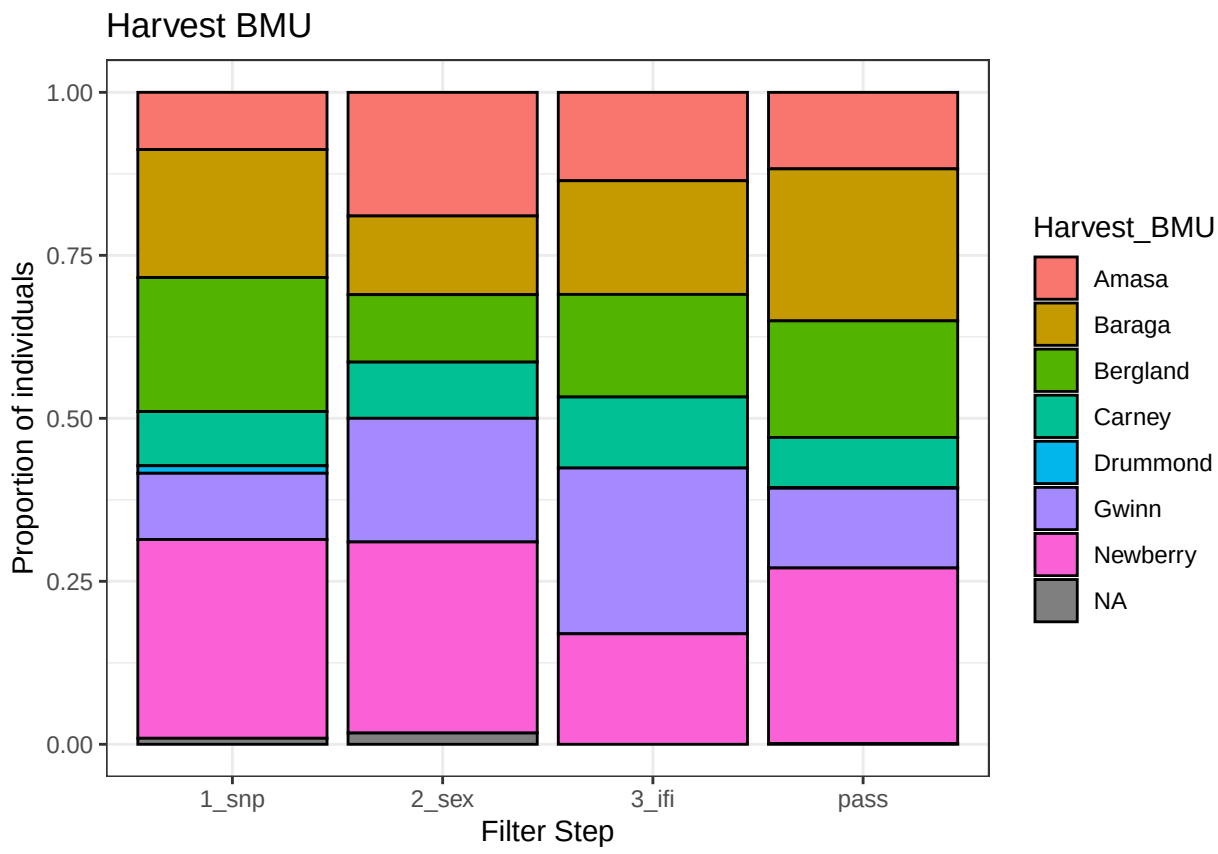
- Sex

```
ggplot(data = full_data) +
  geom_bar(aes(x =filter_remove, fill = Genetic_Sex), color = "black", position = position_fill()) +
  geom_hline(yintercept = 1631/nrow(full_data)) +
  scale_fill_manual(values = c("XX" = "darkorchid1", "XY" = "darkblue", "Missing" = "lightyellow")) +
  ylab("Proportion of individuals") +
  xlab("Filter Step") +
  ggtitle("Sex") +
  theme_bw()
```



- Geographic location

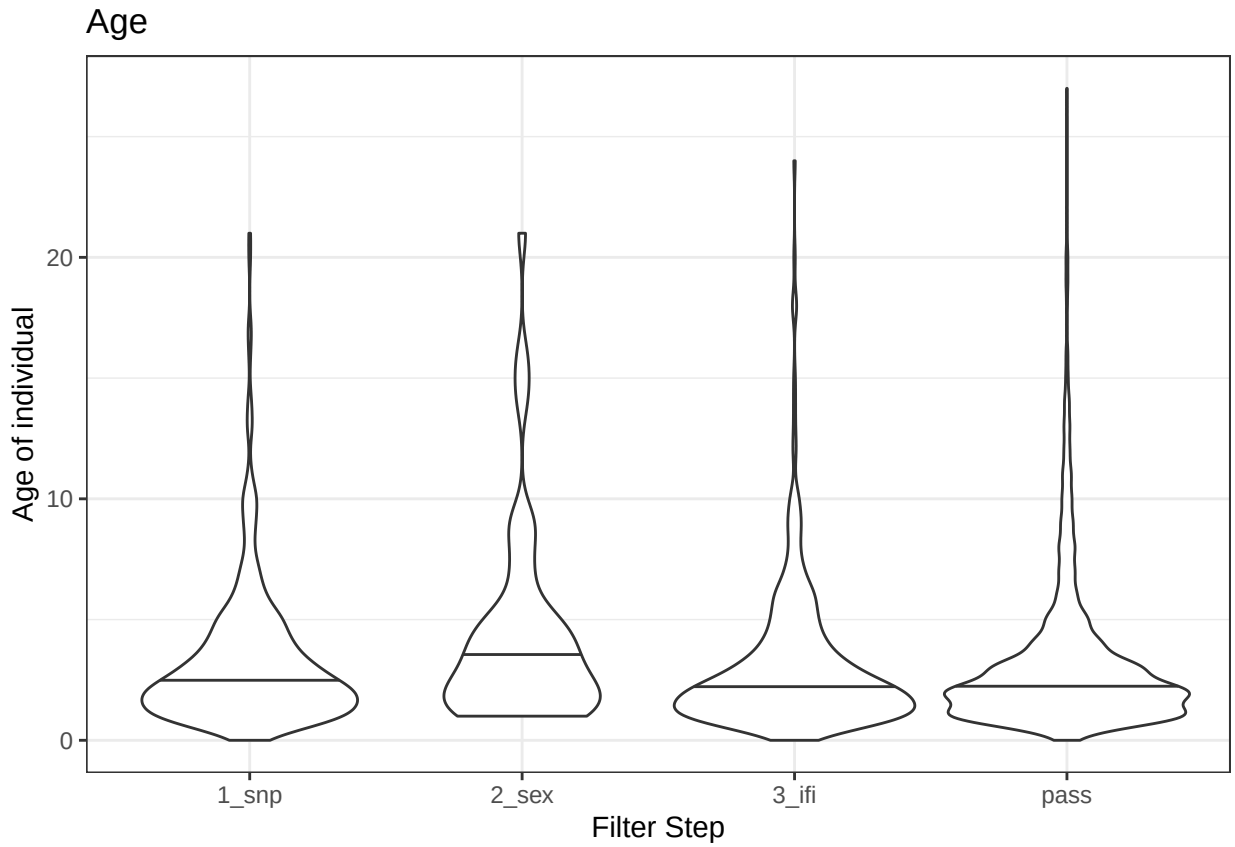
```
ggplot(data = full_data) +
  geom_bar(aes(x =filter_remove, fill = Harvest_BMU), color = "black", position = position_fill()) +
  ggtitle("Harvest BMU") +
  ylab("Proportion of individuals") +
  xlab("Filter Step") +
  theme_bw()
```



- Age

```
ggplot(data = full_data) +
  geom_violin(aes(x =filter_remove, y = Age), draw_quantiles = c(0.5)) +
  ggtitle("Age") +
  ylab("Age of individual") +
  xlab("Filter Step") +
  theme_bw()
```

```
## Warning: Removed 88 rows containing non-finite outside the scale range
## (`stat_ydensity()`).
```

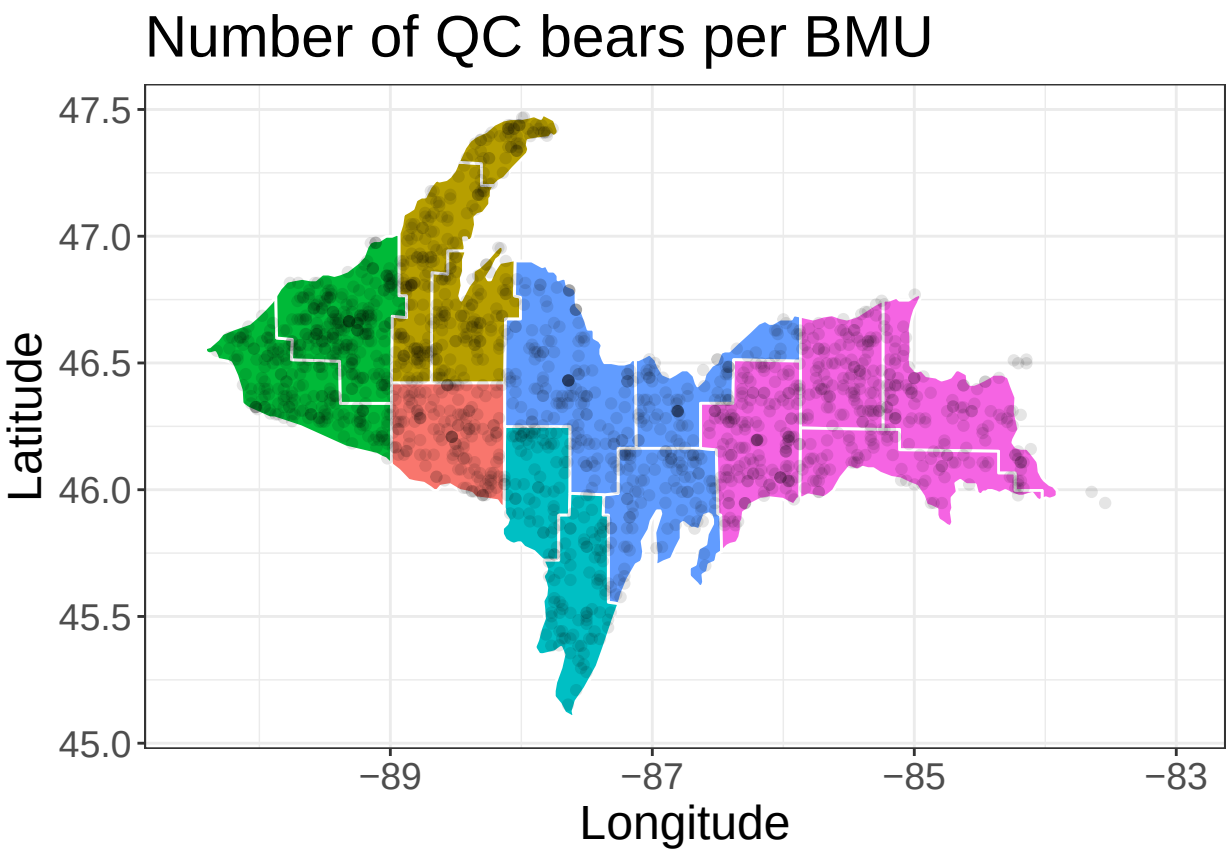


Finally, we can make a map that summarizes how many individuals per BMU

```
full_data <- read.csv("AllBears_fulldata_final.csv")
quality_bears <- read.csv("QCBears_fulldata_v2.csv")

# Determine the number of samples per BMU
all_county_count <- as.data.frame(table(full_data$Harvest_BMU)) # Number of samples prefilter
colnames(all_county_count) <- c("Harvest_BMU", "Pre_Filter_Count")
post_filter_count <- as.data.frame(table(subset(full_data, filter_remove == "pass")$Harvest_BMU)) # Num
colnames(post_filter_count) <- c("Harvest_BMU", "Post_Filter_Count")

# Merge pre and post filter counts to
all_county_count <- merge(all_county_count, post_filter_count, by = "Harvest_BMU")

# Determine fraction that passed filter
all_county_count$Prop <- all_county_count$Post_Filter_Count/all_county_count$Pre_Filter_Count

# Add central lat long coordinates so labels will go in the right place
all_county_count$long <- c(-88.640276, -88.772050, -89.571814, -87.551147, -83.8, -87.439331, -84.5)
all_county_count$lat <- c(46.08292, 46.846302, 46.602538, 45.601479, 45.992422, 46.4, 46.340274)

# Create a label
all_county_count$label <- paste(all_county_count$Harvest_BMU, ": ", all_county_count$Post_Filter_Count,

# Load michigan county map data and assign loose BMU's to each county
mi_counties <- map_data("county", "michigan")
```

```
mi_counties$apprx_BMU[mi_counties$subregion == "gogebic" | mi_counties$subregion == "ontonagon"] <- "Be
mi_counties$apprx_BMU[mi_counties$subregion == "houghton" | mi_counties$subregion == "keweenaw" | mi_cou
mi_counties$apprx_BMU[mi_counties$subregion == "iron"] <- "Amasa"
mi_counties$apprx_BMU[mi_counties$subregion == "marquette" | mi_counties$subregion == "alger" | mi_coun
mi_counties$apprx_BMU[mi_counties$subregion == "dickinson" | mi_counties$subregion == "menominee"] <- "
mi_counties$apprx_BMU[mi_counties$subregion == "schoolcraft" | mi_counties$subregion == "luce" | mi_cou
mi_counties <- subset(mi_counties, !is.na(apprx_BMU))

# Plot
ggplot() +
  geom_polygon(mi_counties, mapping = aes(x=long, y=lat, group = group, fill = apprx_BMU), colour = "wh
  geom_point(data = quality_bears, mapping = aes(x = Longitude, y = Latitude), alpha = 0.1) +
  xlim(-90.5, -83) +
  #ylim(45, 47.6) +
  xlab("Longitude") + ylab("Latitude") +
  ggtitle("Number of QC bears per BMU") +
  theme_bw()+
  theme(text = element_text(size = 18)) +
  theme(legend.position = "none")
```

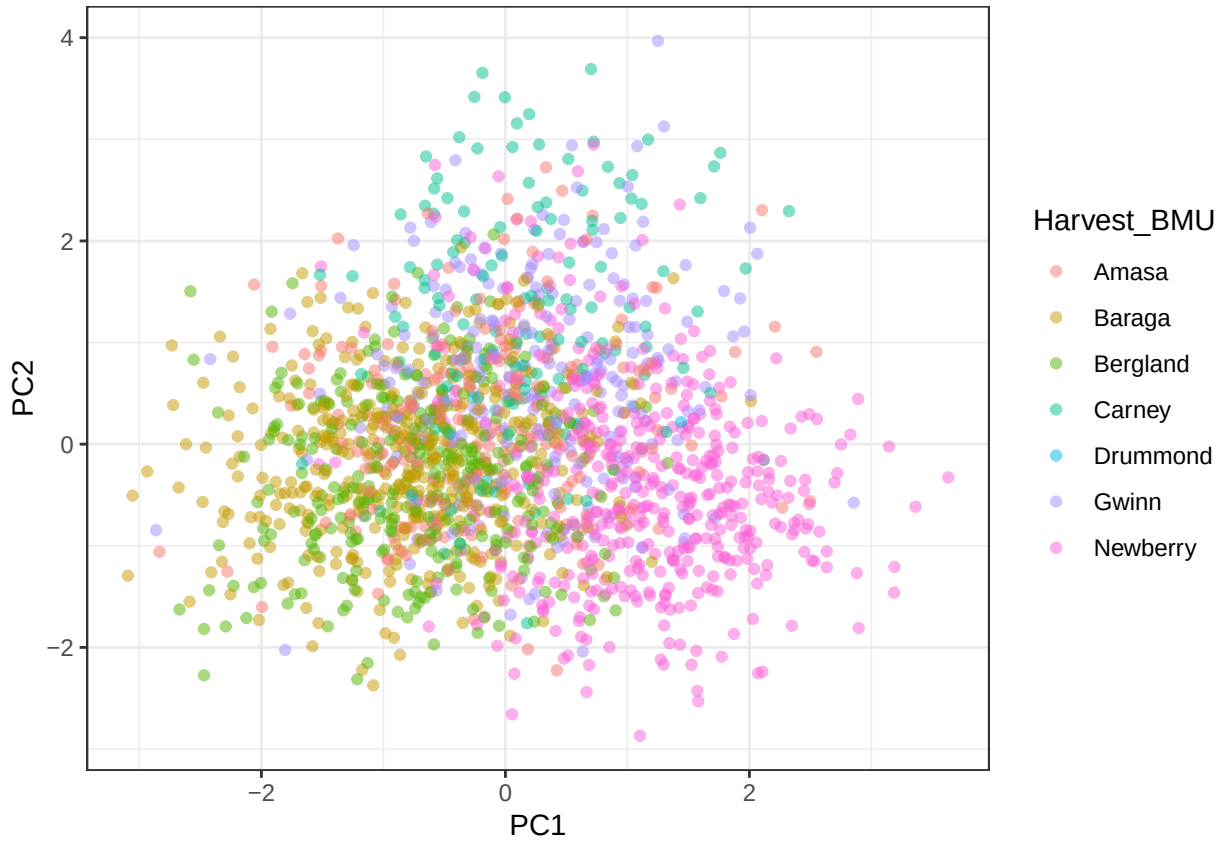
```
## Warning: Removed 3 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



1.1.2 Population genomics analyses Run bb_popgen_nuc.R. This file does the following analyses: PCA, k-means clustering, Fst, and isolation by distance.

1.1.2.1 PCA We performed a PCA on the nuclear genomic data. The first question is, how well do these principal components explain variation in the data? The answer is not super well. PC1 and PC2 only explain ~3% of the variation. Additionally, when we plot the PCA, we see little to no separation between clusters.

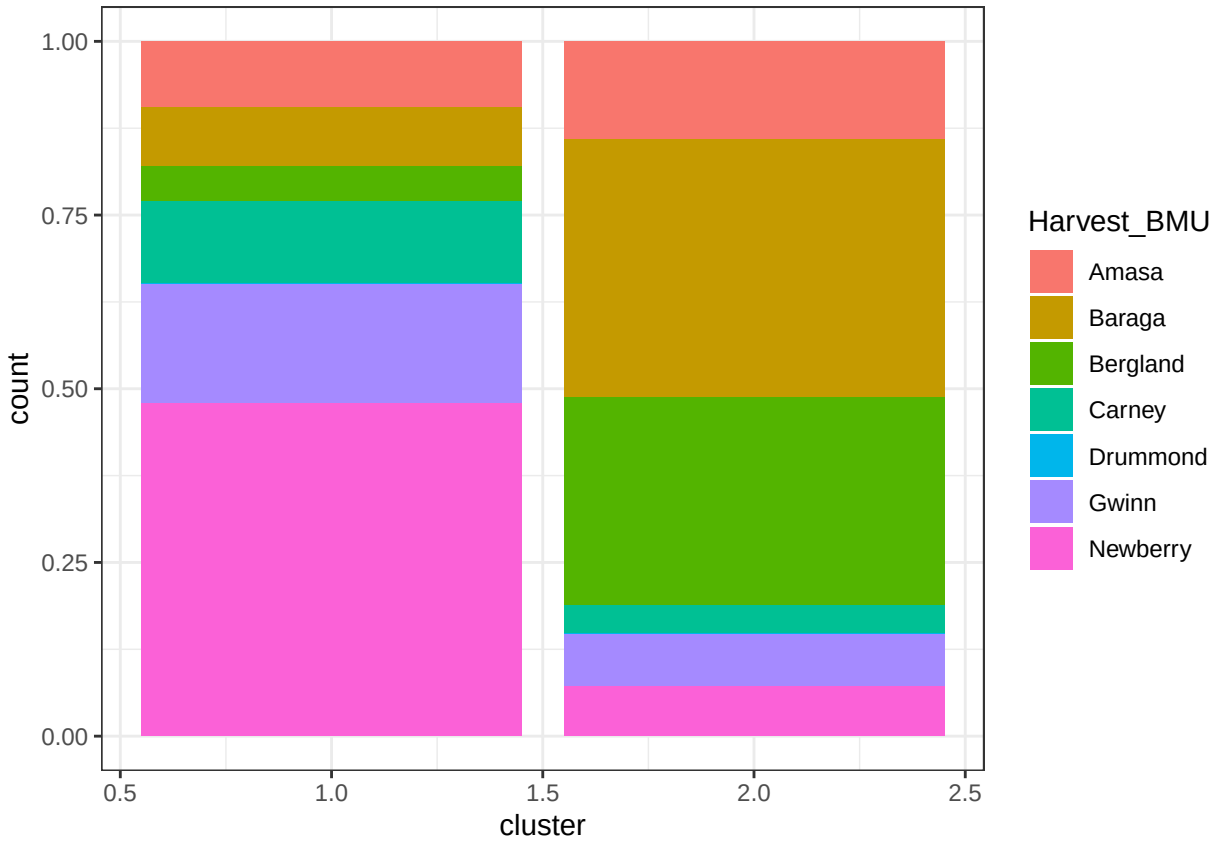
```
popgen_full <- read.csv("nuclear_popgen_clusters.csv")
ggplot(data = popgen_full) +
  geom_point(aes(x = PC1, y = PC2, color = Harvest_BMU), alpha = 0.5) +
  theme_bw()
```



We see a small degree of East / West clustering, driven by Bergland + Baraga vs. Newberry, which are the three BMUs with the most samples.

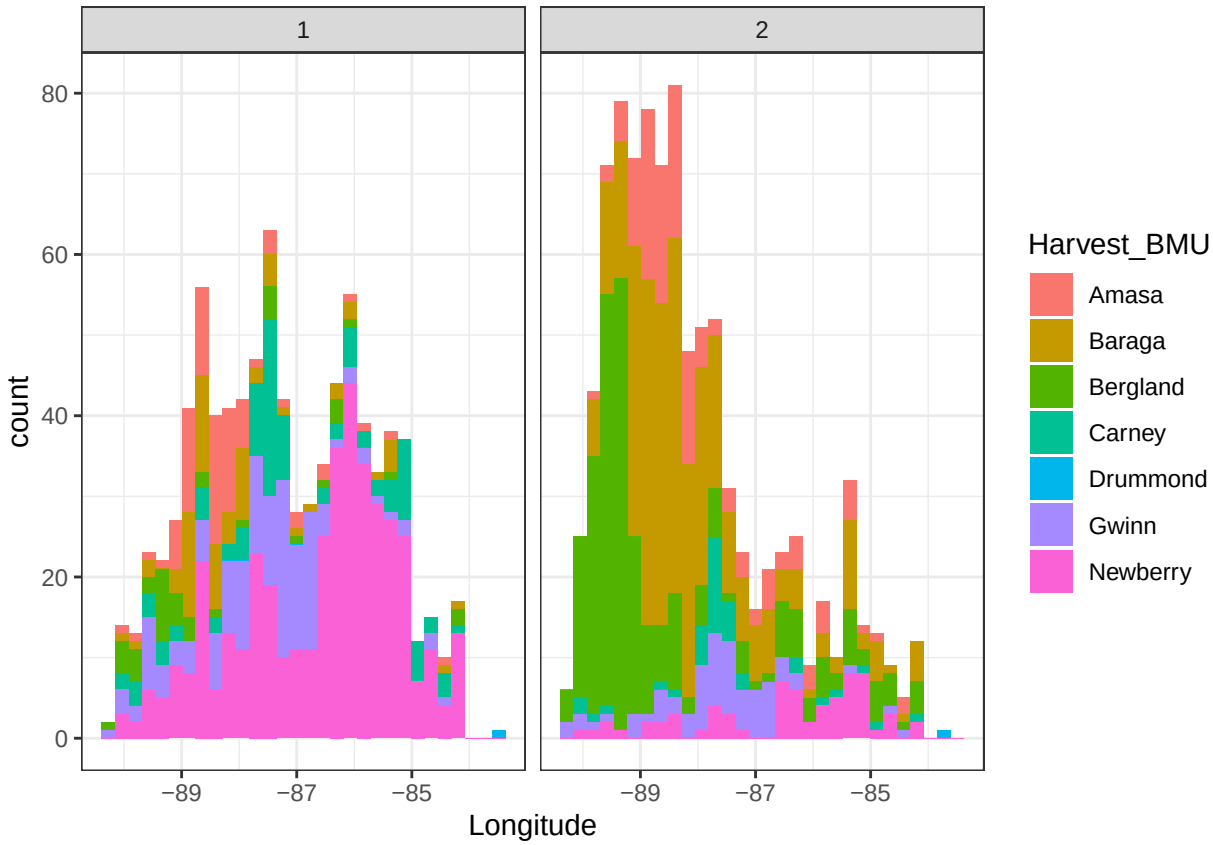
1.1.2.2 K-means clustering We first determined that the optimal number of clusters is k=2. We can now assign all individuals to one of these two clusters, and see if the clusters are correlated with Harvest BMU.

```
ggplot(data = subset(popgen_full, !is.na(Harvest_BMU))) + geom_bar(aes(x = cluster, fill = Harvest_BMU)
  theme_bw()
```




```
ggplot(data = popgen_full) + geom_histogram(aes(x = Longitude, fill = Harvest_BMU)) +
  facet_wrap(~cluster) +
  theme_bw()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

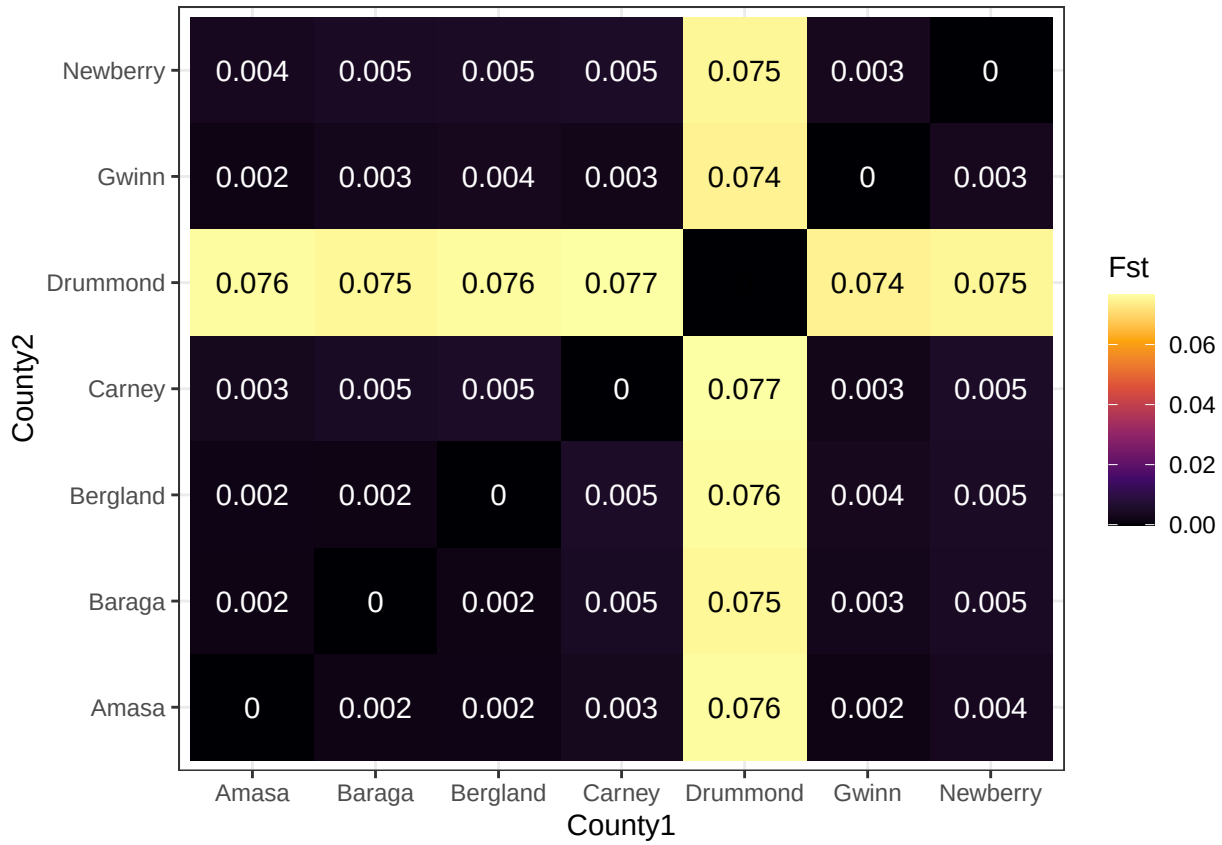


We again see that same minor East / West clustering.

1.1.2.3 Populutation genetics (Fst, IBD)

One of the primary popgen stats we care about is Fst.

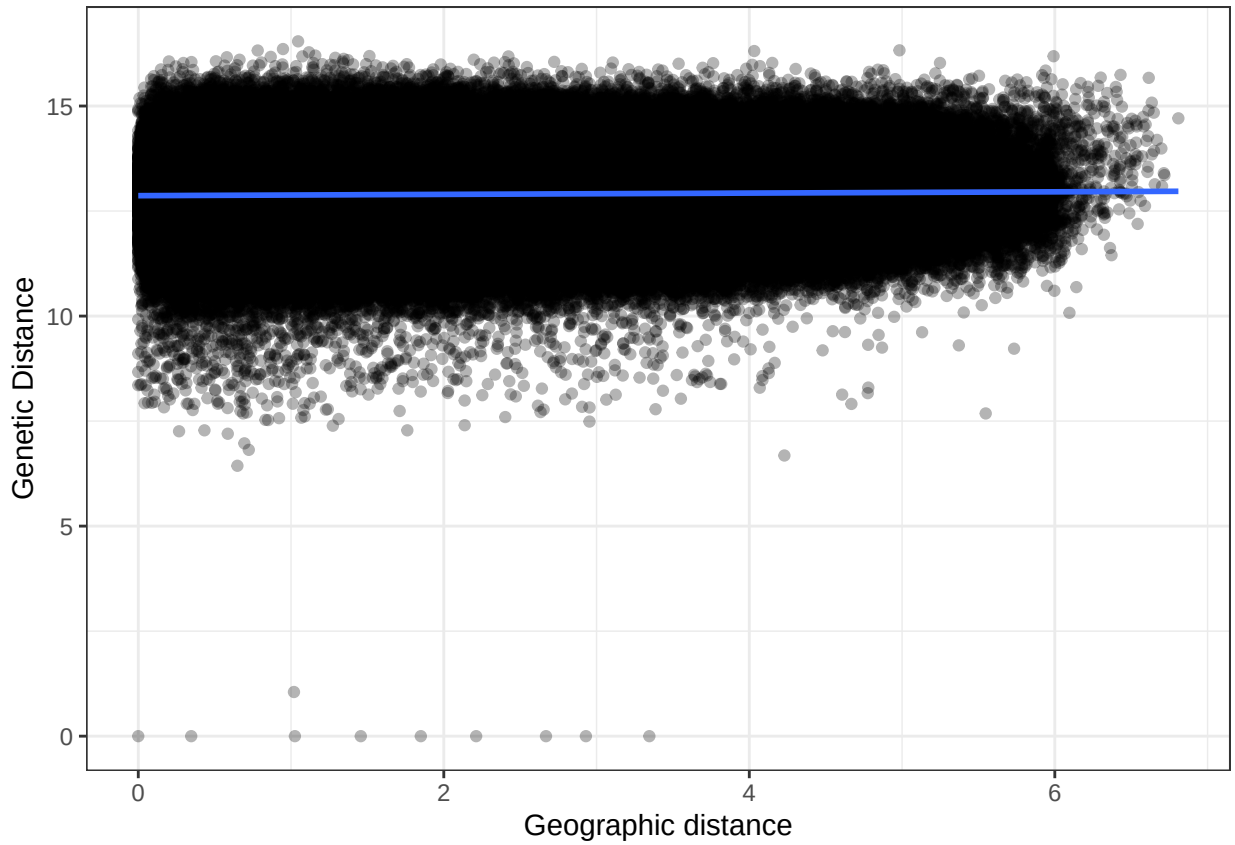
```
gdist_long <- read.csv("nuc_gdist_fst.csv")

# Plot as heat map
ggplot(data = gdist_long, aes(x = County1, y = County2)) + geom_tile(aes(fill = Fst)) +
  geom_text(aes(label = round(Fst, 3), color = color)) +
  scale_color_manual(values = c("black" = "black", "white" = "white"), guide = "none") +
  theme_bw() +
  scale_fill_viridis_c(option = "B")
```



Finally, let's look at isolation by distance by plotting geographic distance vs. genetic difference.

```
dist_long_df <- read.csv("nuc_popgen_ibd.csv")

ggplot(data = dist_long_df, aes(x = geo_dist, y = gen_dist)) + geom_point(alpha = 0.3) +
  theme_bw() +
  geom_smooth(method = "lm") +
  xlab("Geographic distance") +
  ylab("Genetic Distance")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Combined, these analyses all indicate the population is panmictic.

1.2 Mitochondrial DNA

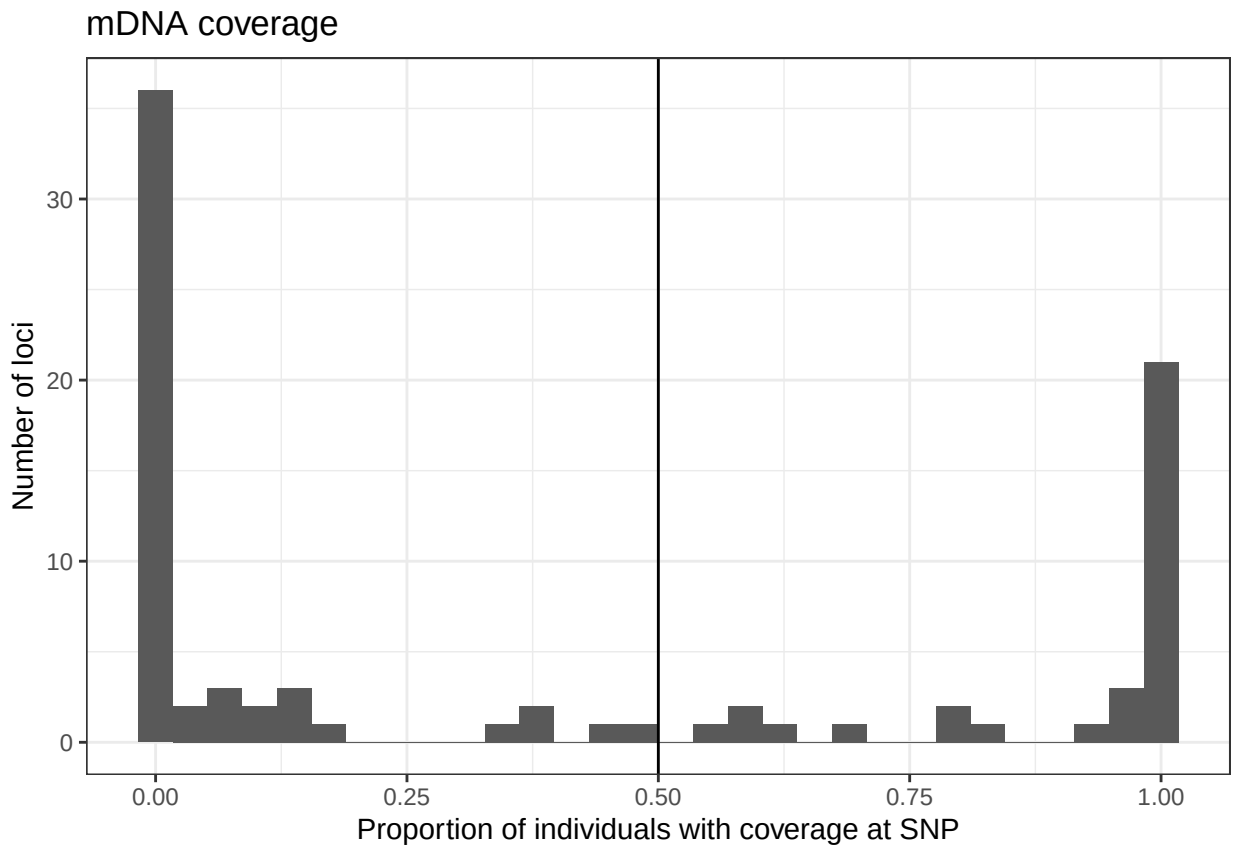
NOTE: Mitochondrial DNA was not used in downstream analyses. However, for completeness sake, I am including the code here.

1.2.1 Filtering Run bb_mtDNA_qcfilter.R. First, this removes low coverage SNPs and individuals. Second, it performs a few preliminary population genomics analyses to evaluate the diversity of mDNA.

Let's first look at how many SNPs and individuals we will be losing.

```
perloc_propna <- read.csv("mtDNA_perlocna.csv")

ggplot() + geom_histogram(data = perloc_propna, aes(x = 1-prop_na)) +
  theme_bw() +
  xlab("Proportion of individuals with coverage at SNP") +
  geom_vline(xintercept = 0.5) +
  ylab("Number of loci") +
  ggtitle("mDNA coverage")
```

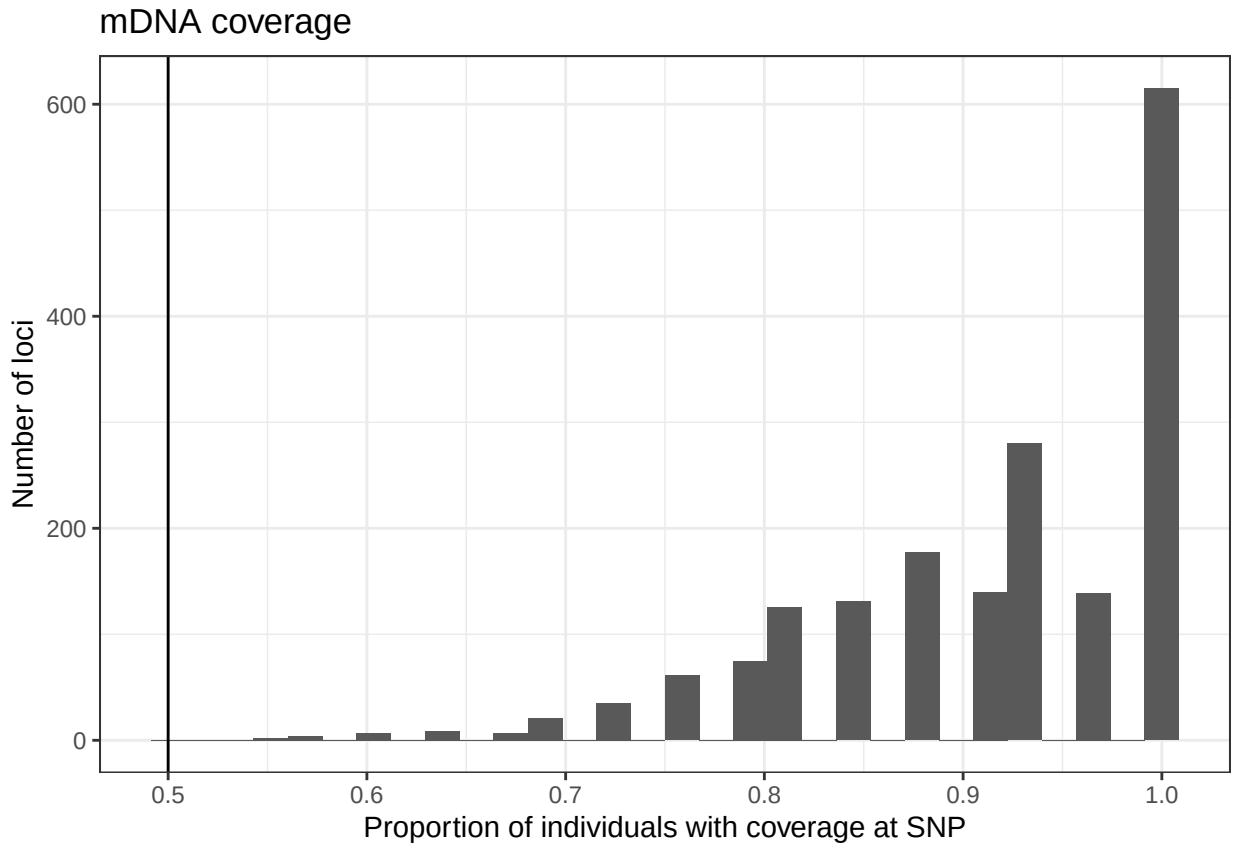
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



We're left with only 33 loci.

```
perbear_propna <- read.csv("mtDNA_perbearna.csv")

ggplot() + geom_histogram(data = perbear_propna, aes(x = 1-prop_na)) +
  theme_bw() +
  xlab("Proportion of individuals with coverage at SNP") +
  geom_vline(xintercept = 0.5) +
  ylab("Number of loci") +
  ggtitle("mDNA coverage")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

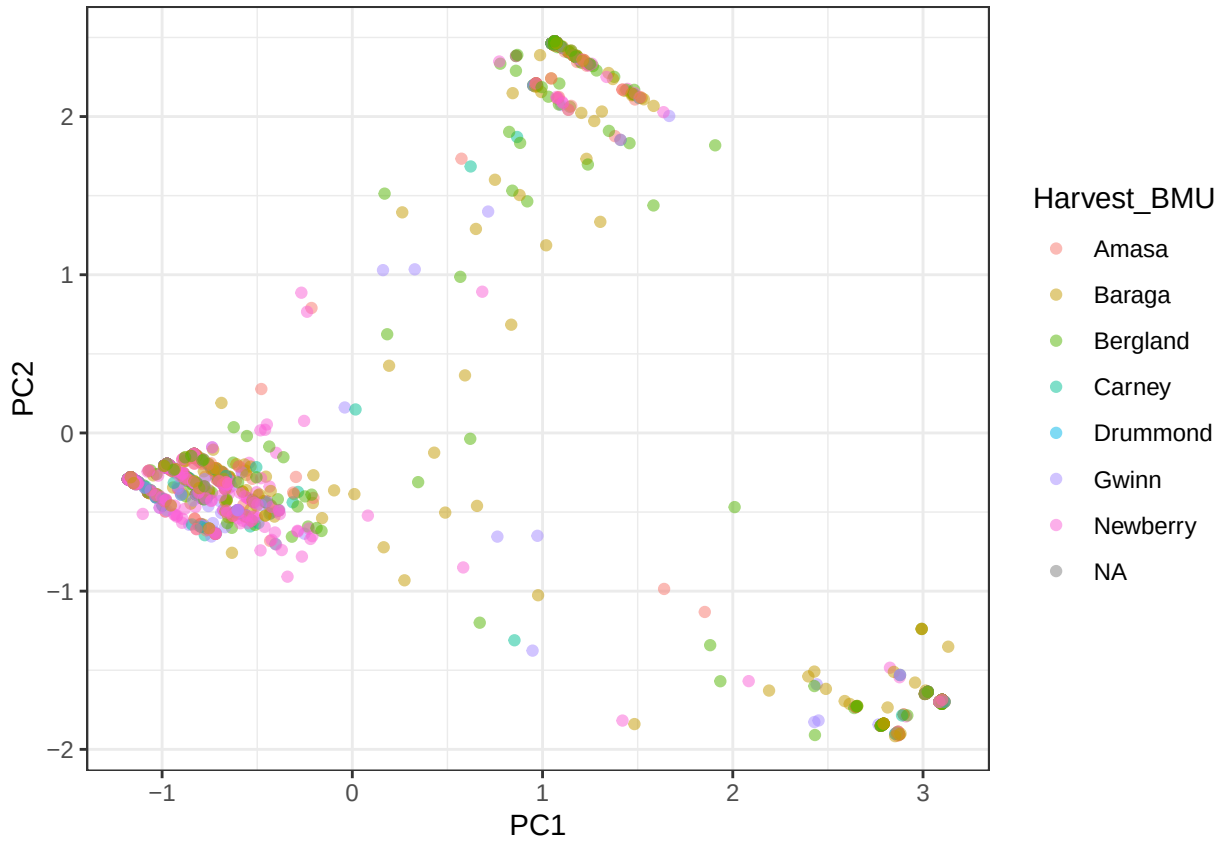


But no individuals need to be removed.

1.2.2 Population genomics analyses (mDNA) Run bb_mtDNA_popgen.R. This file is an updated version of the nuclear popgen file (bb_popgen_nuc.R). This file does the following analyses: PCA, k-means clustering, and genetic distance.

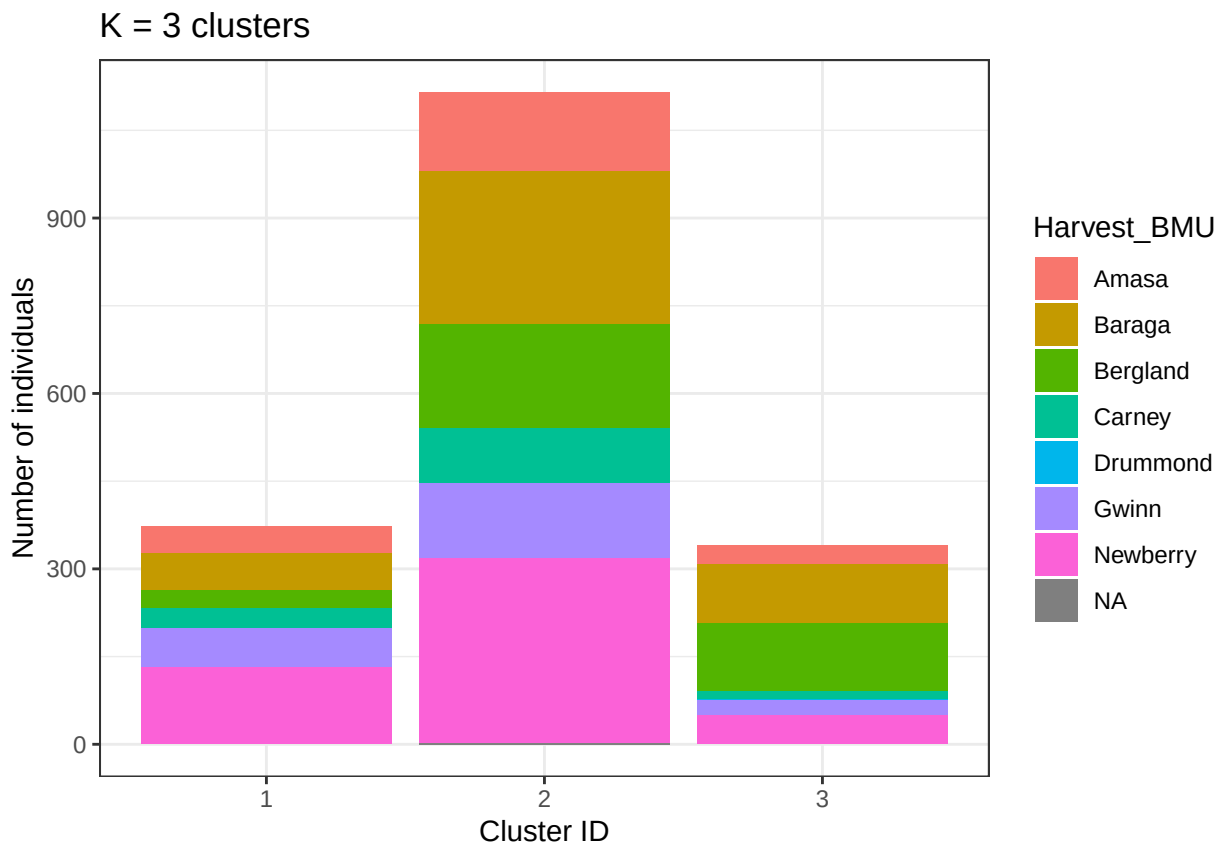
1.2.2.1 PCA The mitochondrial PCA actually seems to do a pretty good job of explaining variation in the data. PC 1 and 2 explain 70.6% of the variation in our data. However, when we plot the PCA, we see no signal for Harvest BMU. There seem to be three major cluster, with some points scattered between them. Of note, the majority of points belong to the same cluster.

```
popgen_full_mDNA <- read.csv("mito_popgen_clusters.csv")

ggplot(data = popgen_full_mDNA) +
  geom_point(aes(x = PC1, y = PC2, color = Harvest_BMU), alpha = 0.5) +
  theme_bw()
```

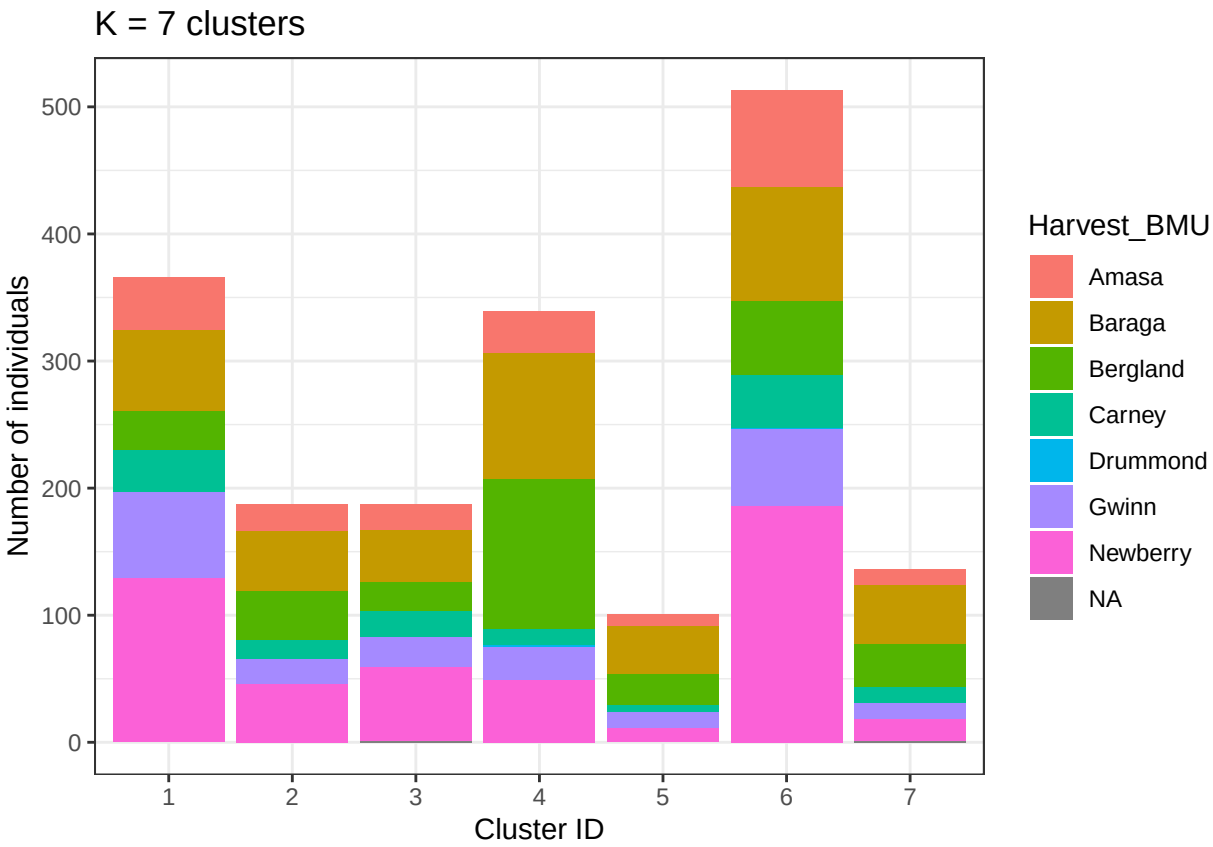


1.1.2.2 K-means clustering First, we had to determine the optimal number of clusters. The silhouette width method predicts K = 3, while the gap statistic methods predicts K = 7. We'll try both and see how Harvest BMU compares.

```
ggplot(data = popgen_full_mDNA) + geom_bar(aes(x = as.factor(cluster_3), fill = Harvest_BMU)) +
  theme_bw() +
  xlab("Cluster ID") +
  ylab("Number of individuals") +
  ggtitle("K = 3 clusters")
```




```
ggplot(data = popgen_full_mDNA) + geom_bar(aes(x = as.factor(cluster_7), fill = Harvest_BMU)) +
  theme_bw() +
  xlab("Cluster ID") +
  ylab("Number of individuals") +
  ggtitle("K = 7 clusters")
```



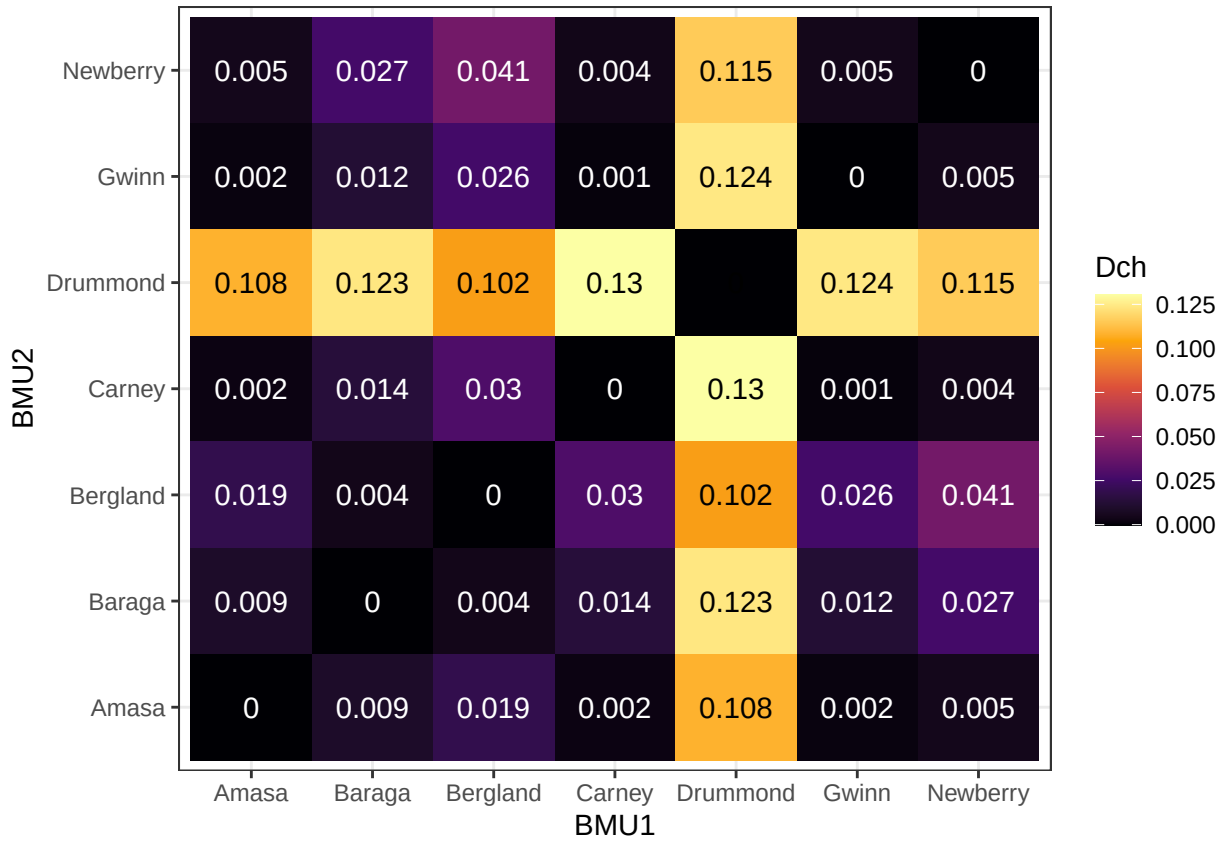
Neither k cluster is able to align with Harvest BMU. Also, when k = 3, we see that over 2/3 of samples belong to the same cluster, which is similar to what we saw in the PCA.

One of our primary conclusions at this point was that the lack of diversity means that mitochondrial DNA won't be useful for resolving relationships.

1.1.2.3 Genetic distance

Let's look at genetic distance again.

```
gdist_long_mtDNA <- read.csv("mtDNA_gdist_long.csv")

# Plot as a heatmap
ggplot(data = gdist_long_mtDNA, aes(x = BMU1, y = BMU2)) + geom_tile(aes(fill = Dch)) +
  geom_text(aes(label = round(Dch, 3), color = color)) +
  scale_color_manual(values = c("black" = "black", "white" = "white"), guide = "none") +
  theme_bw() +
  scale_fill_viridis_c(option = "B")
```



Again Drummond is the most differentiated, but the populations appear panmictic.

1.3 Mitochondrial Haplotypes

From our PCA and k-means clustering, it seems like there are ~3 haplotypes, with over two thirds of individuals belonging to one of those haplotypes. In order to investigate this in a more rigorous way, we should run code specifically designed to assign and analyze haplotype networks. To do this, run bb_mDNA_haplotypes.R. *However, this is still in progress and not working great right now.*

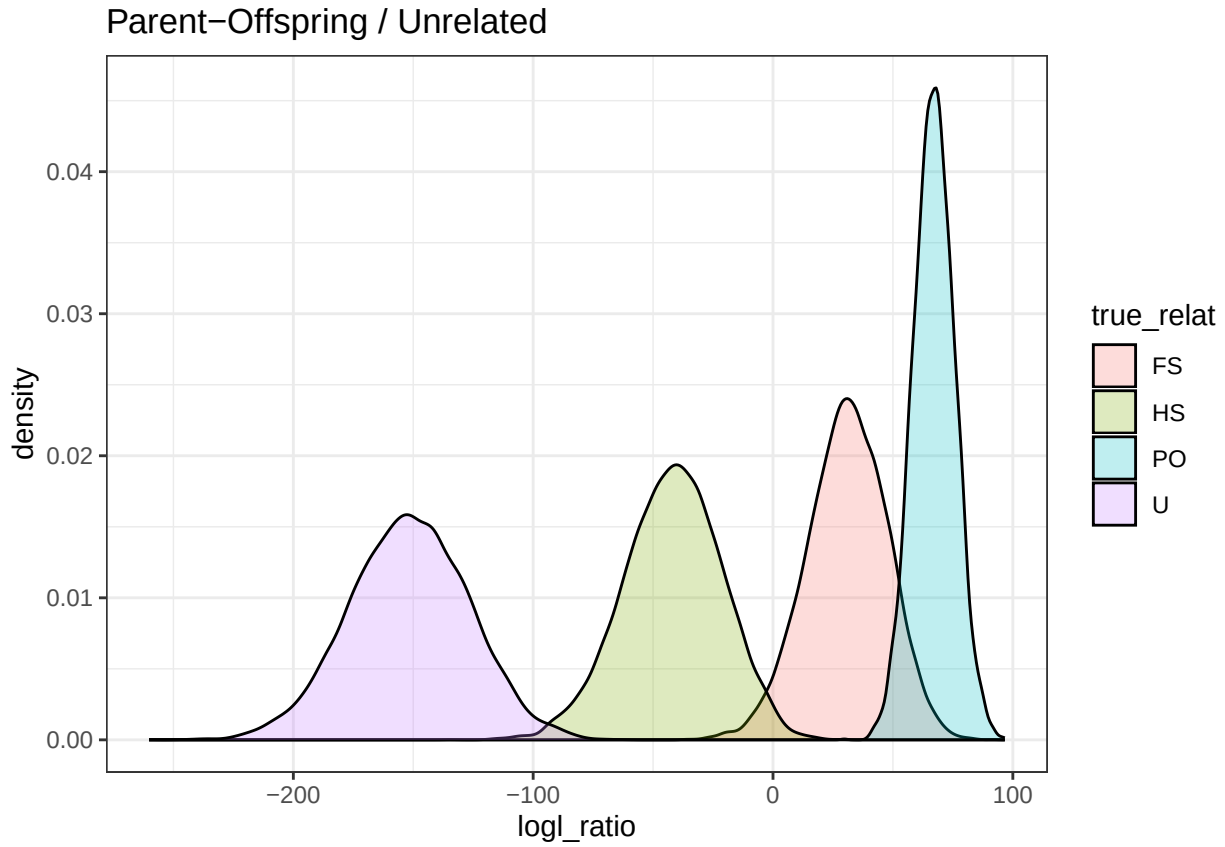
Altogether, we concluded that lack of diversity in mitochondrial DNA makes it not useful to us. However, the nuclear DNA seems to be sufficient to resolve relationships.

Step 2: Testing inference power using CKMRsim

Run bb_pedigree_diagnostics.R file. This files uses the CKMRsim package to test the inference power of our data by comparing false positive and false negative rates in assigning relationships. I highly recommend looking at {this CKMRsim package vignette}(https://eriqande.github.io/CKMRsim/articles/CKMRsim-example-1.html).

This file first finds allele frequencies. Then, it uses these allele frequencies to simulate genomes and uses the simulated data to test the ability to infer relationships. Using this simulated data, we can look at the power to discern relationship types. We'll focus on three focal relationship types: parent offspring (PO), full sibling (FS), and half-sibling (HS). Using simulated genotypes for which the "true relationship" is known, we can visualize the log likelihood ratio: or the log of the probability that the pair is assigned the focal relationship divided by the probability they are assigned as unrelated (U). For example, we can look at the log likelihood density plot for PO vs. Unrelated individuals:

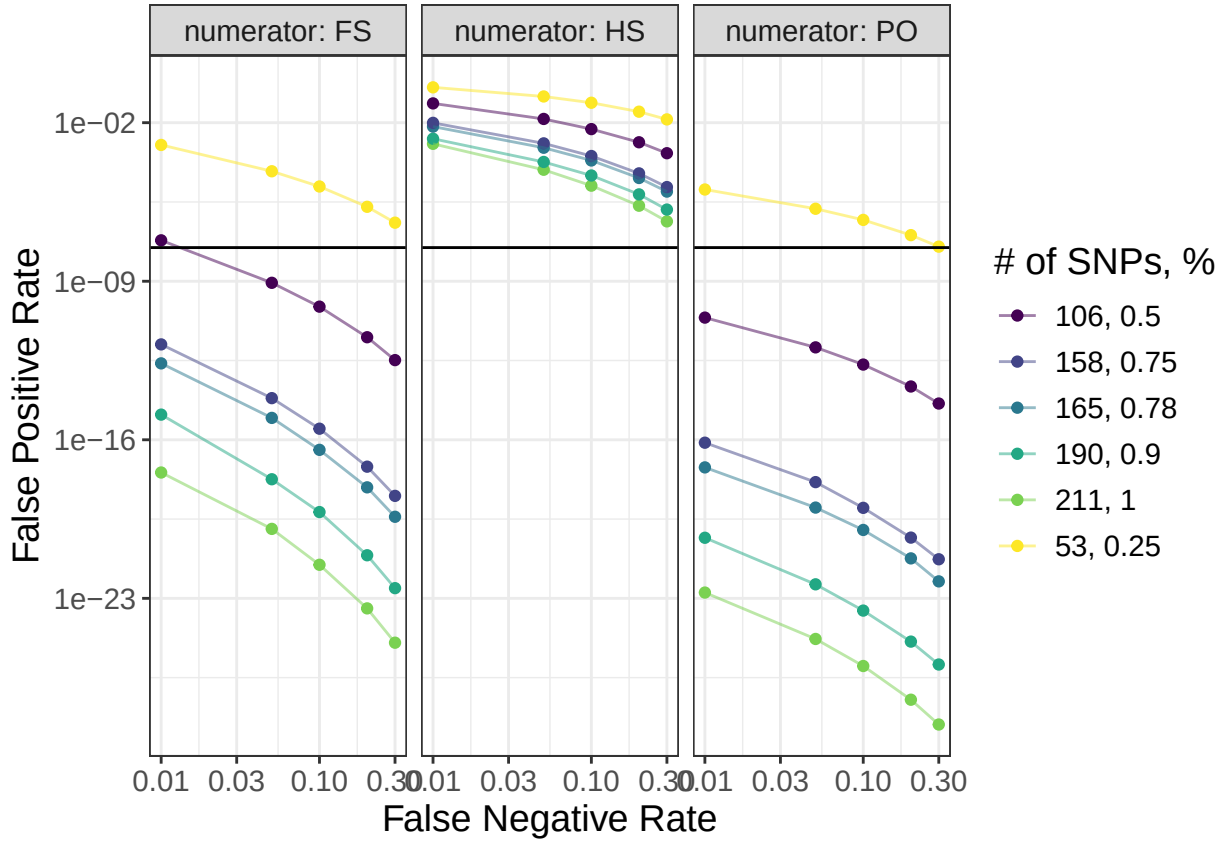
```
PO_U_logls <- read.csv("PO_U_logls.csv")

ggplot(PO_U_logls, aes(x = logl_ratio, fill = true_relat)) +
  geom_density(alpha = 0.25) +
  ggtitle("Parent-Offspring / Unrelated") +
  theme_bw()
```



Another important metric is false positive vs. false negative rates. Here, we also want to check to see the minimum number of loci needed to maintain an acceptable error rate. We want the false positive rate (FPR) to be at least 10 times smaller than the reciprocal of the number of comparisons.

```
error_rate_df <- read.csv("CKMRsim_error_rates.csv")

# Calculate the threshold of acceptable false positive rates
logls_thresh <- 3.059167e-08

# Plot error rate
ggplot(subset(error_rate_df, denominator == "U")) +
  geom_point(aes(x = FNR, y = FPR, color = nloc_lab)) +
  geom_line(aes(x = FNR, y = FPR, color = nloc_lab), alpha = 0.5) +
  facet_wrap(~ numerator, labeller = label_both) +
  scale_y_log10() + scale_x_log10() +
  ylab("False Positive Rate") +
  xlab("False Negative Rate") +
  scale_color_viridis(discrete = T, name = "# of SNPs, %") +
  geom_hline(yintercept = logls_thresh) +
```

```
  theme_bw() +
  theme(text = element_text(size = 14))
```



We have pretty high confidence in our ability to assign FS relationships. Even with using as few as 75% of available SNPs, we still maintain an acceptable FPR. For HS relationships, we do not have the confidence to resolve relationships, even when using 100% of available SNPs. Finally, we have very high confidence in our ability to resolve PO relationships, even when using as few as 50% of available SNPs.

Step 3: Constructing pedigree in Sequoia

Run the bb_sequoia_pedigree_inference.R file to construct a pedigree and find all PO pairs.

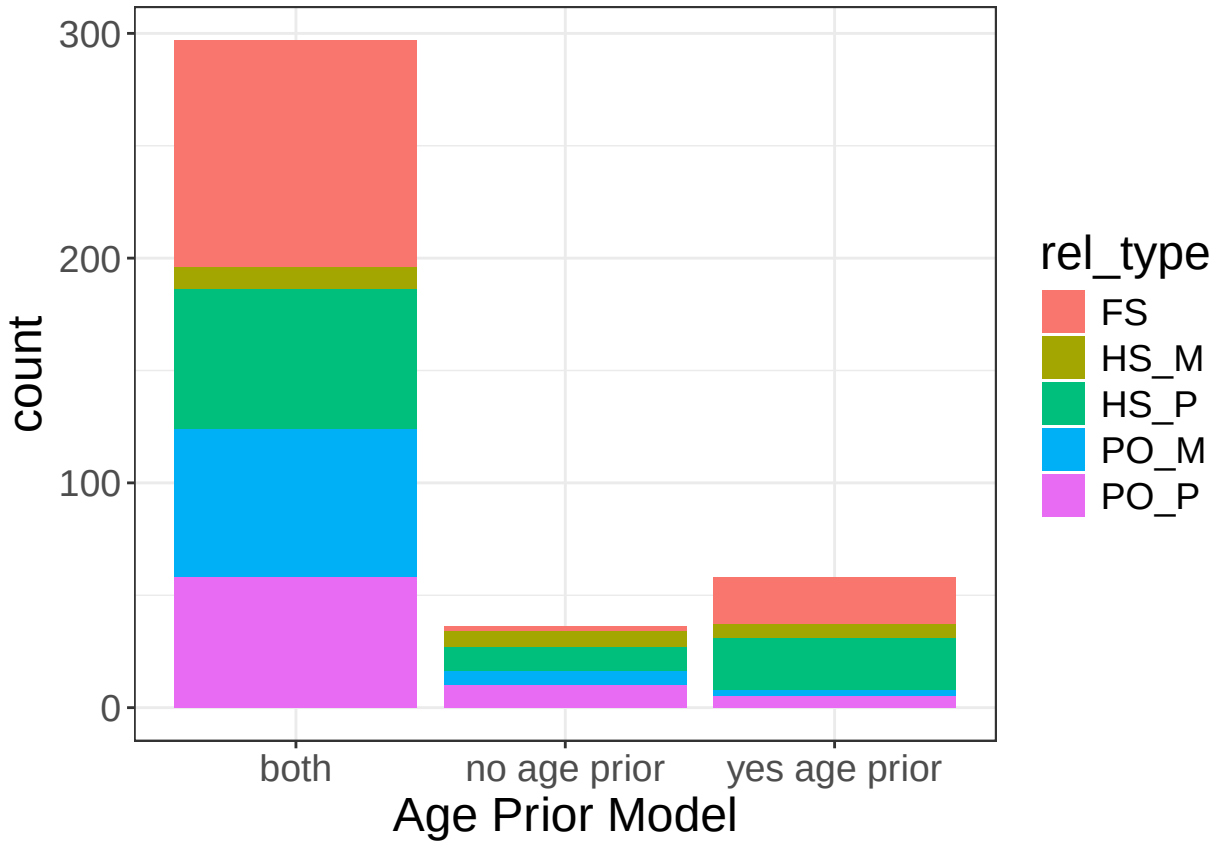
3.1 Pairs are robust across model runs

This code creates four versions of the pedigree: with and without age priors, as well as both full and PO pedigrees. We did this to make sure that the pairs identified were robust across slightly different model runs with slightly different inputs. We expect to find (for the most part) the same pairs regardless of what type of model we use.

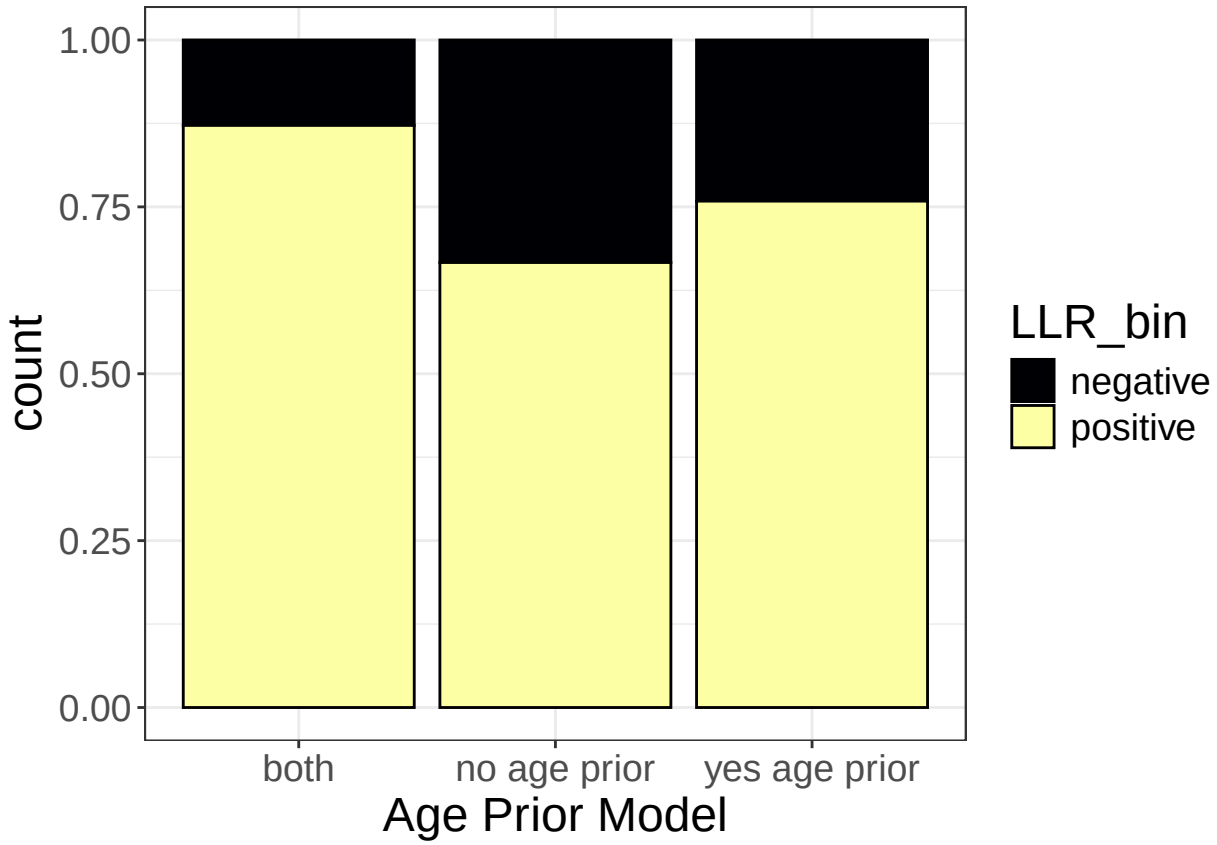
First, let's look at the effect of turning on and off age priors. Age priors limit the possible age differences between parents and offspring.

```
seqped_all <- read.csv("seq_allpeds.csv")
```

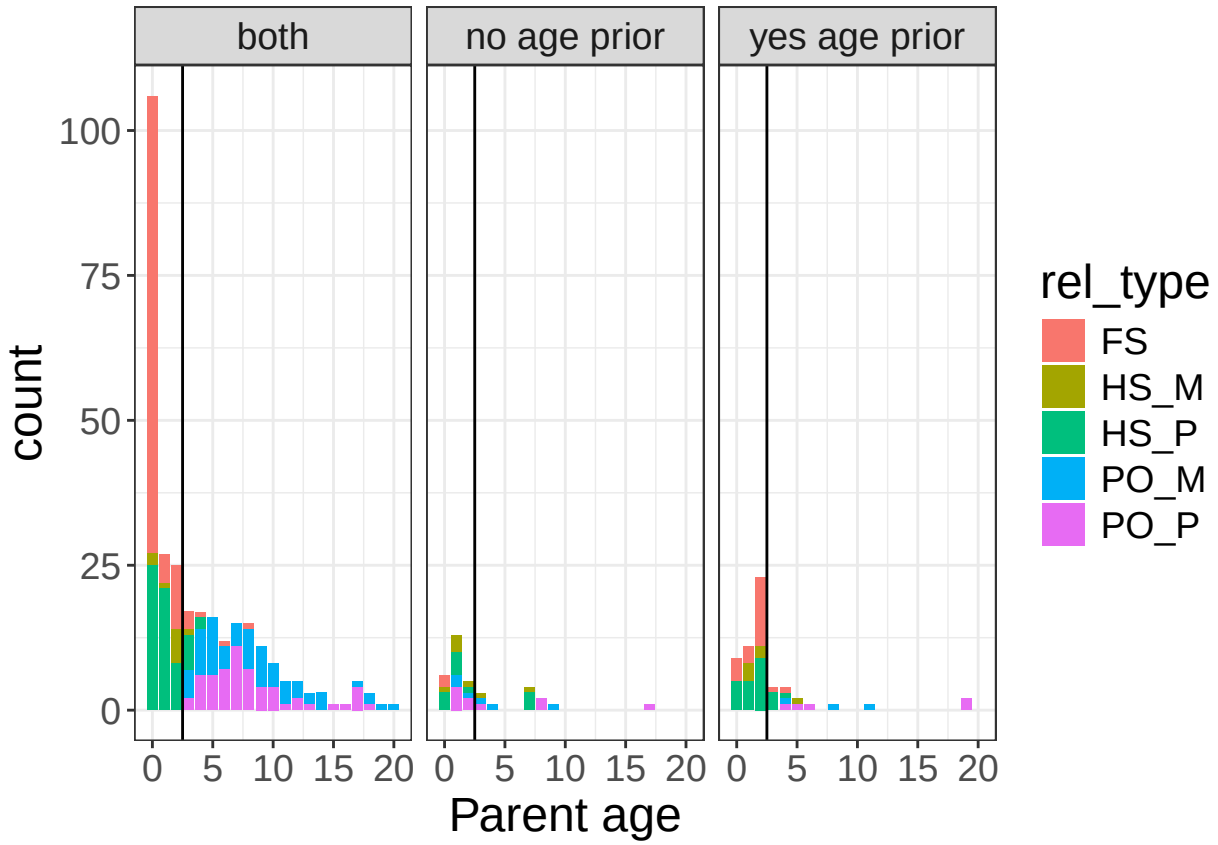
```r
# Construct pedigree overlap dataframe
ap_overlap <- subset(seqped_all, model == "full ped, none" | model == "full ped, yes ap") %>%
  group_by(pair_id) %>%
  mutate(n_model = n(),
         which_models = ifelse(n_model == 2, "both", model)) %>%
  ungroup() %>%
  distinct(pair_id, .keep_all = TRUE)
ap_overlap$which_models[ap_overlap$which_models == "full ped, none"] <- "no age prior"
ap_overlap$which_models[ap_overlap$which_models == "full ped, yes ap"] <- "yes age prior"
ap_overlap$LLR_bin <- ifelse(ap_overlap$LLR >= 0, "positive", "negative")

# Plot results: number of pairs that are in both models?
ggplot(data = ap_overlap) + geom_bar(aes(x = which_models, fill = rel_type)) +
  theme_bw() +
  theme(text = element_text(size = 18)) +
  xlab("Age Prior Model")
```




```r
# How does the LLR break down across these three groups?
ggplot(data = ap_overlap) +
  geom_bar(aes(x = which_models, fill = LLR_bin), color = "black", position = position_fill()) +
  geom_vline(xintercept = -0.25) +
  scale_fill_viridis(discrete = T, option = "B") +
  theme_bw() +
  theme(text = element_text(size = 18)) +
  xlab("Age Prior Model")
```



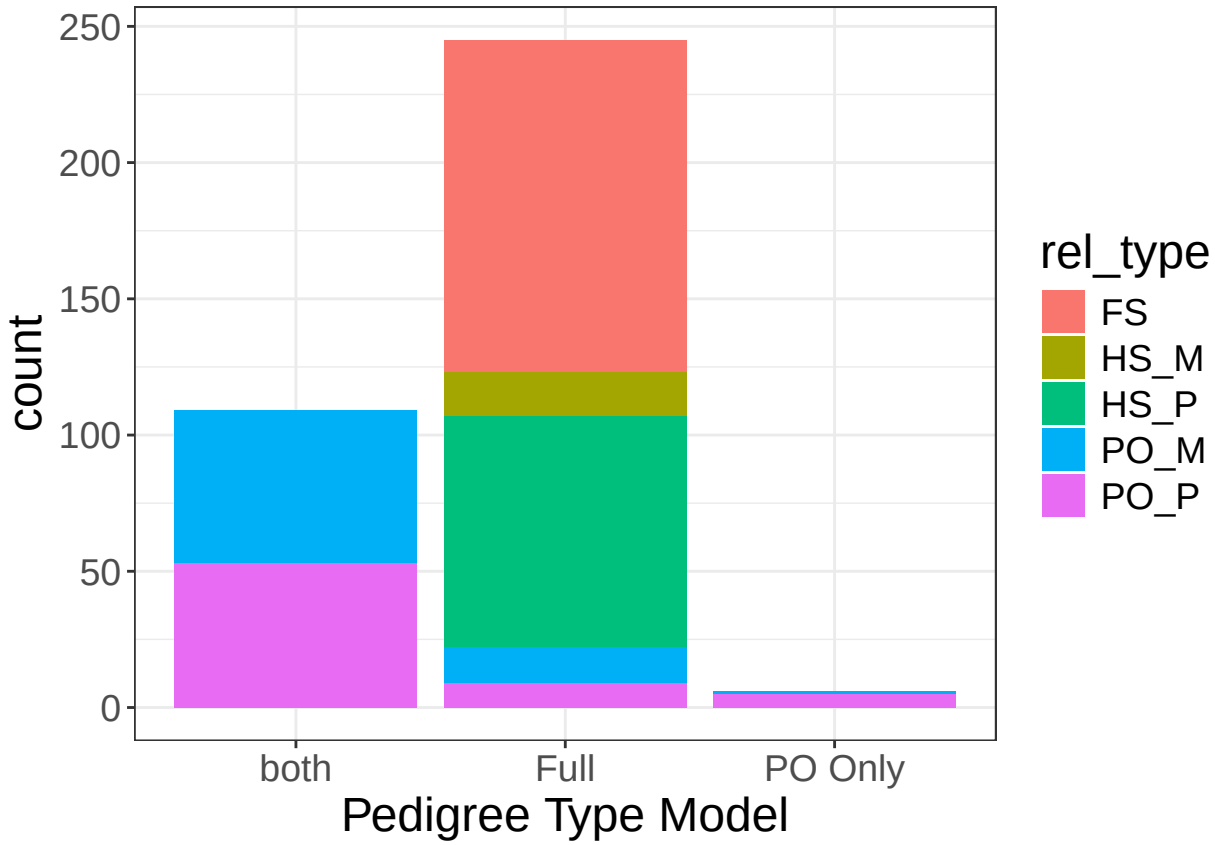

```
# How does the age difference between parents and offspring compare across models?
ggplot(data = ap_overlap) + geom_bar(aes(x = age_diff, fill = rel_type)) +
  geom_vline(xintercept = 2.5) +
  theme_bw() +
  theme(text = element_text(size = 18)) +
  facet_wrap(~ which_models) +
  xlab("Parent age")
```



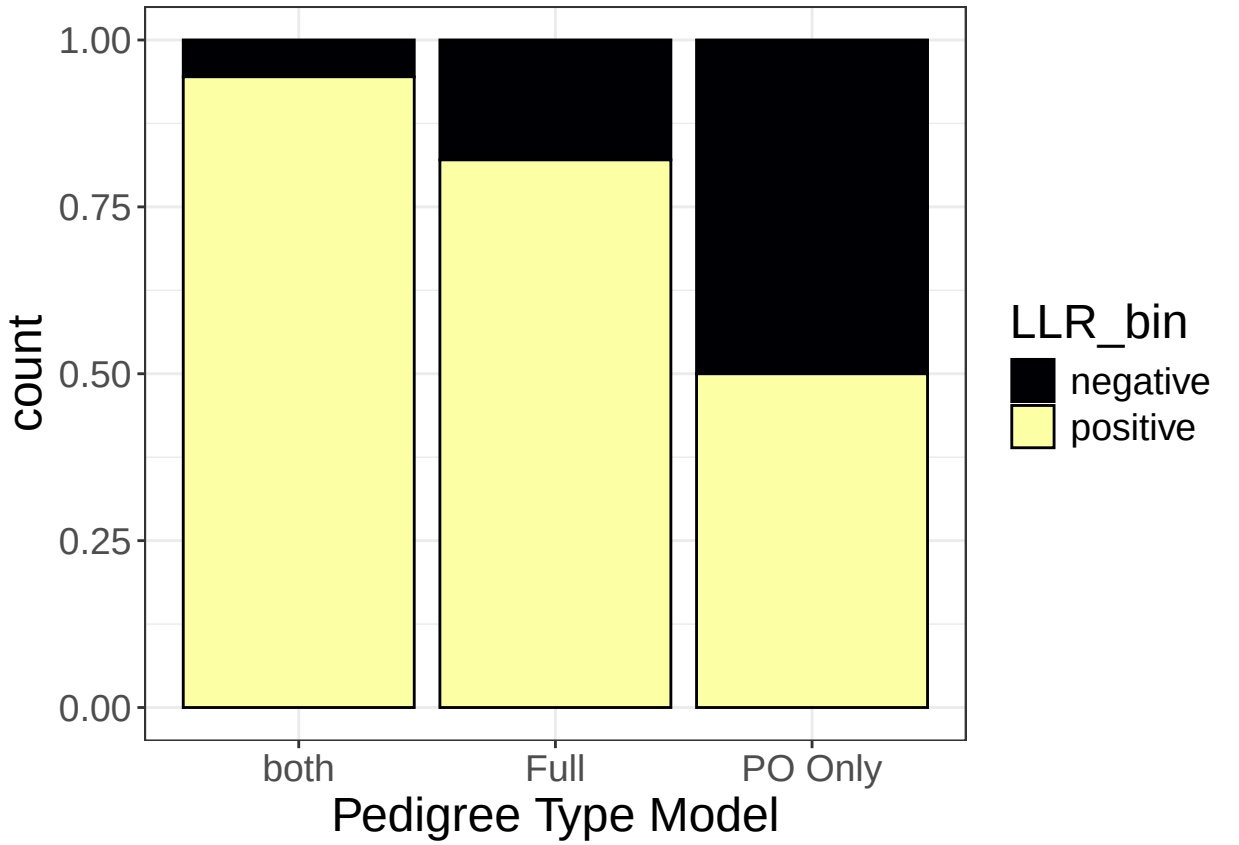
The majority of pairs are found in both models. Also, pairs in only one of the two models were more likely to have a LLR < 0, which means we would have lower confidence in those pairs. Unsurprisingly, most pairs found in only the pedigree with no age priors were those in which the parent's age was less than 3, which are explicitly excluded from the model that uses our age prior.

Nex, let's look at: does constructing a full vs. PO pedigree change assigned pairs?

```
# Construct pedigree overlap dataframe
ped_overlap <- subset(seqped_all, model == "full ped, yes ap" | model == "PO only, yes ap") %>%
  group_by(pair_id) %>%
  mutate(n_model = n(),
         which_models = ifelse(n_model == 2, "both", model)) %>%
  ungroup() %>%
  distinct(pair_id, .keep_all = TRUE)
ped_overlap$which_models[ped_overlap$which_models == "full ped, yes ap"] <- "Full"
ped_overlap$which_models[ped_overlap$which_models == "PO only, yes ap"] <- "PO Only"
ped_overlap$LLR_bin <- ifelse(ped_overlap$LLR >= 0, "positive", "negative")

# Plot results: number of pairs that are in both models?
ggplot(data = ped_overlap) + geom_bar(aes(x = which_models, fill = rel_type)) +
  theme_bw() +
  theme(text = element_text(size = 18)) +
  xlab("Pedigree Type Model")
```




```
# How does the LLR break down across these three groups?
ggplot(data = ped_overlap) + geom_bar(aes(x = which_models, fill = LLR_bin), color = "black", position =
  geom_vline(xintercept = -0.25) +
  scale_fill_viridis(discrete = T, option = "B") +
  theme_bw() +
  theme(text = element_text(size = 18)) +
  xlab("Pedigree Type Model")
```



Again, the majority of PO pairs were found in both the full and PO only pedigrees. Also, pairs in only one of the two models were more likely to have a LLR < 0.

Now that we've demonstrated our results across iterations, we can feel pretty confident in the assigned pears. Because constructing pedigrees can take a bit of time and we know that we don't need to run multiple models, in the future one could only construct "seqped_ap_full" and comment out the other versions.

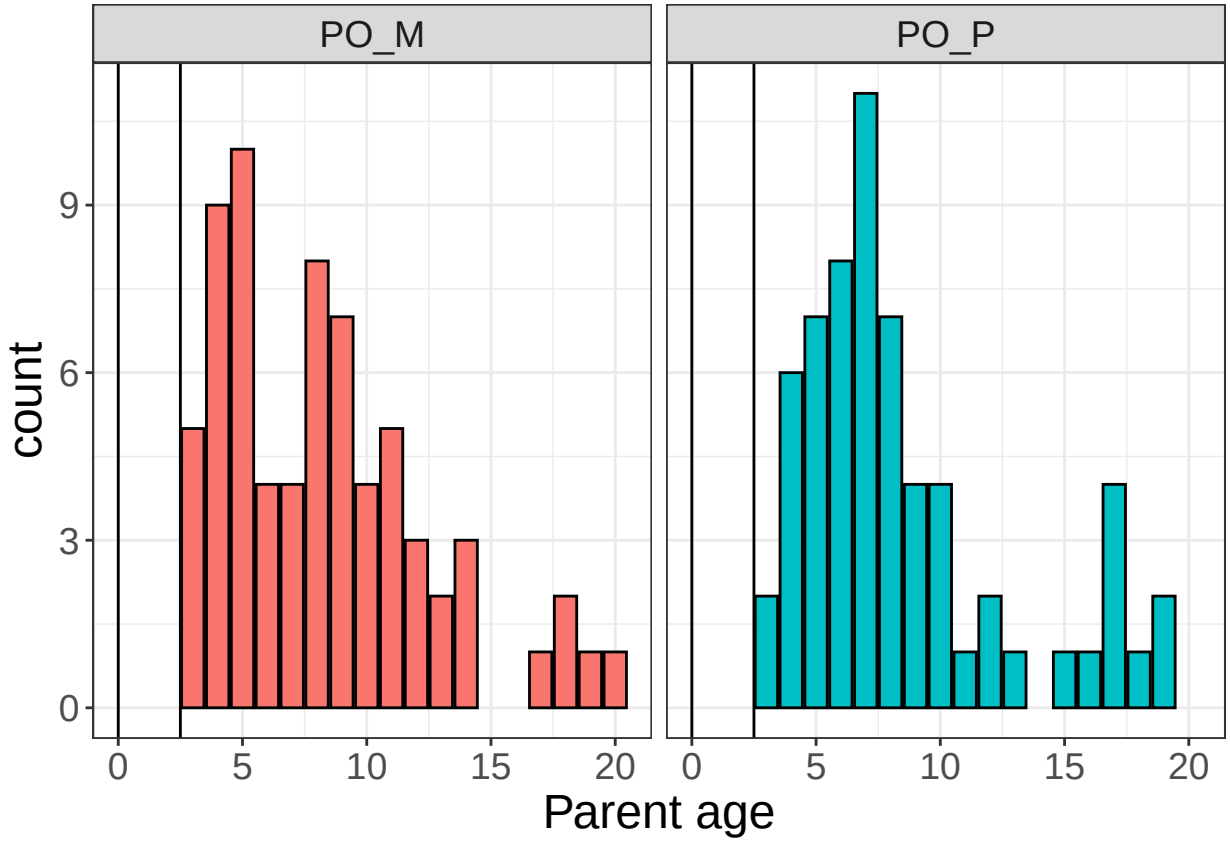
3.2 Analyzing PO pairs

In the file, I provide code to visualize various life history traits across pairs: age, geography, and number of connections per bear.

3.3.1 Age To start with, we can look at age. The primary aspect of age we will want to analyze is parent age. Parent age is simply the age of the parent minus the age of the offspring. Because of the age prior, we won't have any parents who are less than 3 years old.

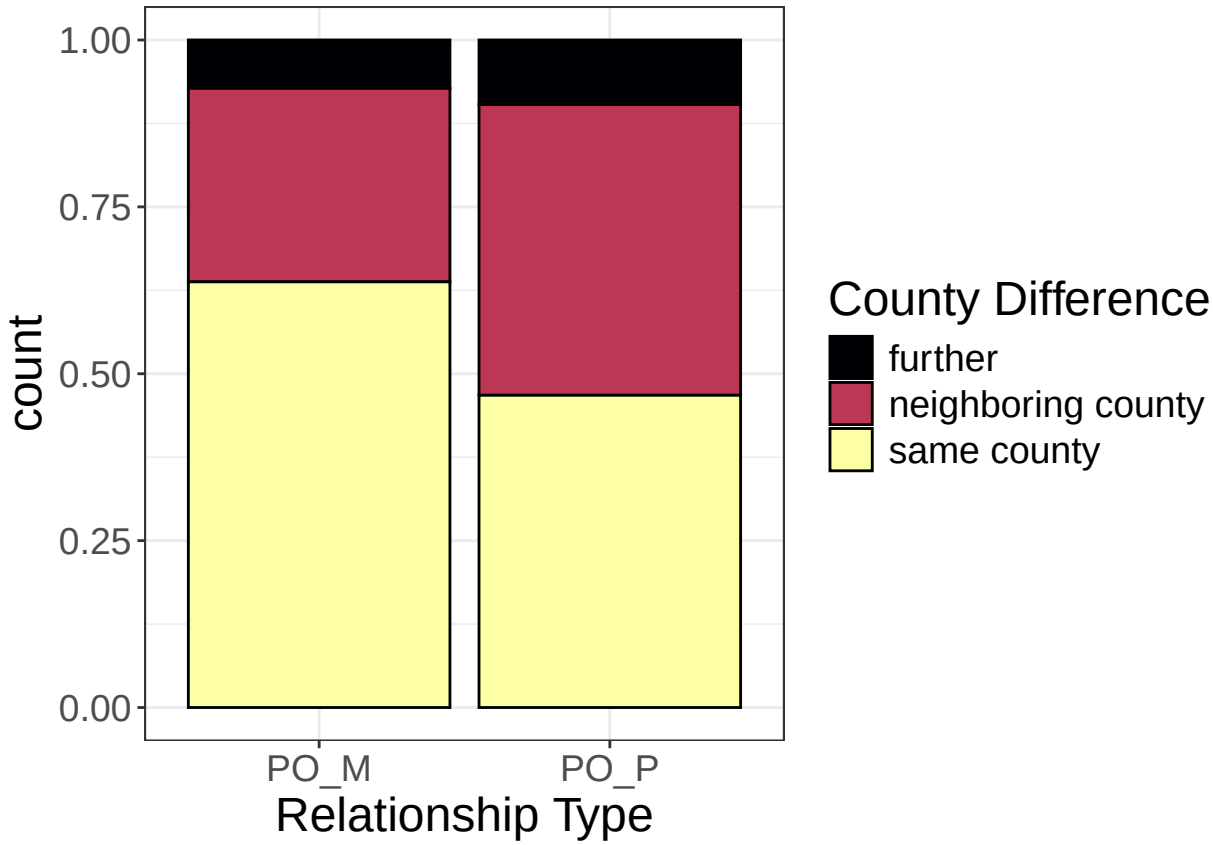
```
final_pair_df <- read.csv("BB_POpairs_final.csv")

ggplot(data = final_pair_df) +
  geom_bar(aes(x = age_diff, fill = rel_type), color = "black") +
  geom_vline(xintercept = 0) +
  geom_vline(xintercept = 2.5) +
  theme_bw() +
  theme(legend.position = "none") +
  theme(text = element_text(size = 18)) +
```

```
  facet_wrap( ~rel_type) +
  xlab("Parent age")
```



3.3.2 Geography The primary sanity check with respect to geography is how far apart parent and offspring were harvested. For a few reasons, I don't feel super confident in the latitude and longitude coordinates provided. Instead, I use county as a metric for location. I checked to see if parent and offspring were harvested in the same county. If they were harvested from different counties, I noted if they were found in counties directly bordering each other (neighboring county) or further.

```
ggplot(data = final_pair_df) +
  geom_bar(aes(x = rel_type, fill = county_difference), position = position_fill(), color = "black")+
  scale_fill_viridis(discrete = T, option = "B", name = "County Difference") +
  theme_bw() +
  theme(text = element_text(size = 18)) +
  xlab("Relationship Type")
```

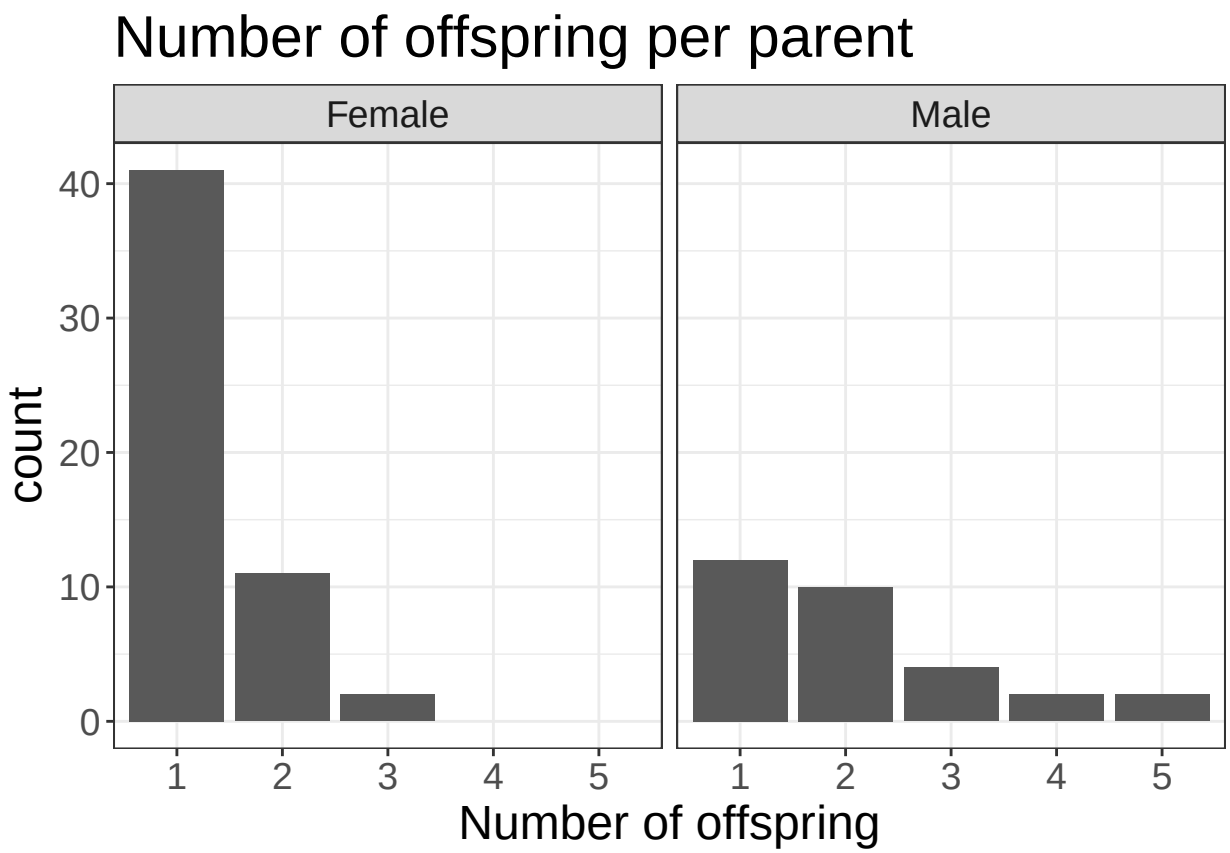


The vast majority of pairs were found in the same county or neighboring counties, which aligns with our expectation that bears do not travel that far.

3.3.4 Number of connections per bear The final sanity check is to look at the number of connections per bear. Given the size of our dataset, it is unlikely that multiple connections (i.e. mutliple offsprings per parent, or multiple parents per offspring) will be super common.

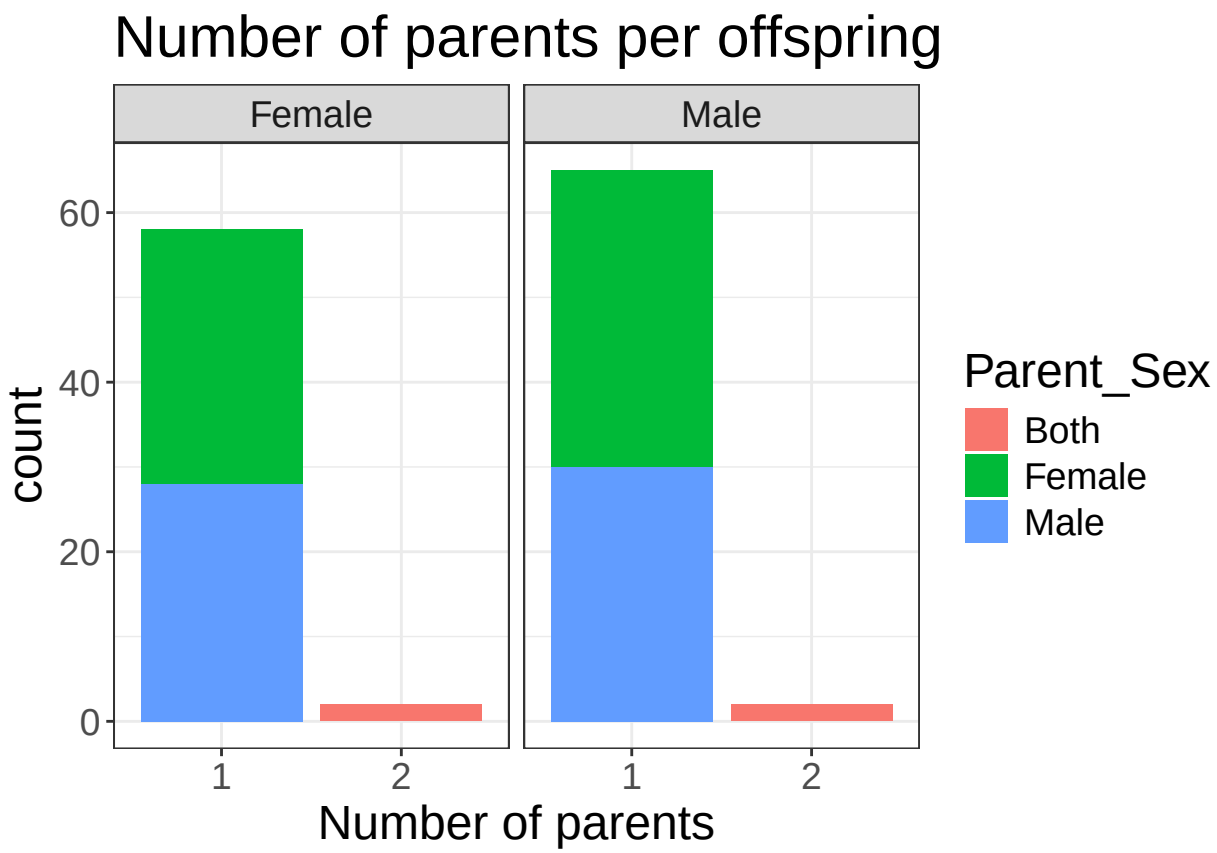
Let's first look at number of offspring per parent. Multiple offspring are not impossible; fathers especially are likely to have multiple offspring. But if there is an outlier (i.e. one bear who sired over half of all offspring), that would skew the CKMR model.

```
# Construct connections per bear dataframe
ap_df_parents <- final_pair_df %>%
  group_by(bear2) %>%
  summarize(n_offs = n(),
            bear2_sex = max(bear2_sex))

# Plot number of offspring per parent
ggplot(data = ap_df_parents) + geom_bar(aes(x = as.factor(n_offs))) +
  facet_grid(~ bear2_sex) +
  xlab("Number of offspring") +
  theme_bw() +
  ggtitle("Number of offspring per parent") +
  theme(text = element_text(size = 18))
```



These distributions look pretty good, so let's move on to look at offspring.

```
ap_df_off <- final_pair_df %>%
  group_by(bear1) %>%
  summarize(n_pars = n(),
            bear1_sex = max(bear1_sex),
            bear2_sex = max(bear2_sex))
colnames(ap_df_off) <- c("bear1", "n_pars", "Offspring_Sex", "Parent_Sex")
ap_df_off$Parent_Sex[ap_df_off$n_pars == 2] <- "Both"

# Visualize number of parents per offspring
ggplot(data = ap_df_off) + geom_bar(aes(x = as.factor(n_pars), fill = Parent_Sex)) +
  facet_grid(~ Offspring_Sex) +
  xlab("Number of parents") +
  theme_bw() +
  ggtitle("Number of parents per offspring") +
  theme(text = element_text(size = 18))
```



No offspring have more than 2 parents, which would be biologically impossible. Also, offspring for which both parents were harvested are very rare, which makes sense.