KW Population Analysis

Caila Kucheravy

2024-12-18

Examine population structure using PCA. Script from E. de Greef, help from the PCAdapt vignette: https://bcm-uga.github.io/pcadapt/articles/pcadapt.html.

Prep the environment:

```r
setwd("~/Dropbox/killer_whale_genomics/snps3/chp_2_gen_snps")

library(pcadapt)
library(ggplot2)
library(ggrepel)
library(patchwork)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(StAMPP)
```

```
## Loading required package: pegas

## Loading required package: ape

##
## Attaching package: 'ape'

## The following object is masked from 'package:dplyr':
##
##     where

##
## Attaching package: 'pegas'

## The following object is masked from 'package:ape':
##
##     mst

## Registered S3 method overwritten by 'ade4':
##   method      from
##   print.amova pegas
```

```r
library(vcfR)
```

```
##
##    *****       ***   vcfR   ***       *****
##    This is vcfR 1.15.0
##      browseVignettes('vcfR') # Documentation
##      citation('vcfR') # Citation
##    *****       *****      *****       *****

##
## Attaching package: 'vcfR'

## The following objects are masked from 'package:pegas':
##
##     getINFO, write.vcf
```

Load data and sample info:

```r
sample_info <- read.csv("chp2_killerwhale_genomics_sample_info_round3_kinship_removed.csv", header=T)
```

Remove duplicates & close kin (based on kinship file):

```r
sample_info <- sample_info %>%
  filter(!remove_duplicates == "x") %>%
  filter(!remove_closekin == "x")
```

Verify that snp file IDs are the in the same order as the metadata file:

```r
snp_IDs <- read.table("killerwhale3_snps.ID.filter1.miss.biallel.min100kb.autosomes.hwe.maf.LDprunedr08
snp_IDs$vcf_ID <- sample_info$genome_sample_ID
snp_IDs$all_equal <- snp_IDs$V1==snp_IDs$vcf_ID #column should all say "TRUE"
snp_IDs$all_equal
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [16] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [31] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [46] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [61] TRUE TRUE
```

PCA

Load SNP data with pcadapt:

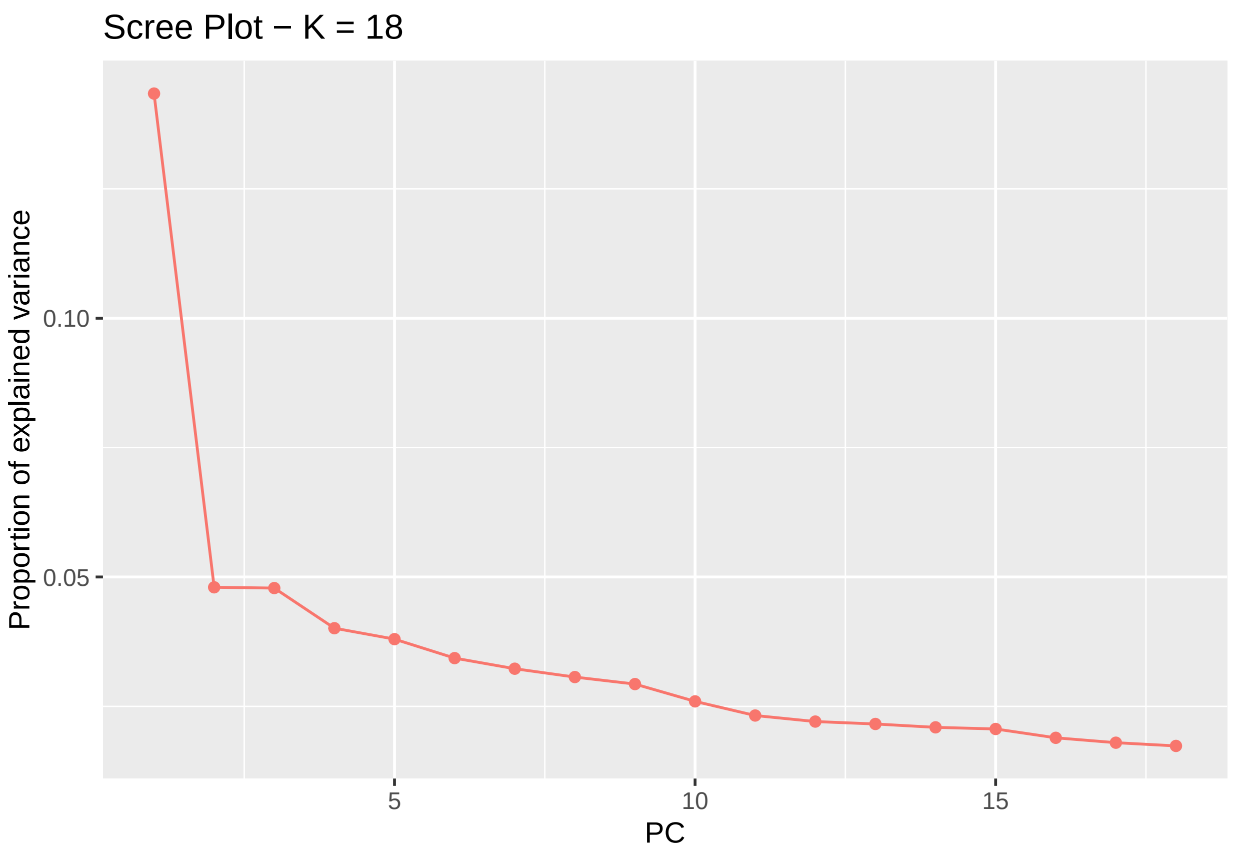
```r
snp_data_pca <- read.pcadapt("killerwhale3_snps.ID.filter1.miss.biallel.min100kb.autosomes.hwe.maf.LDpru
```

Run pcadapt, setting k-value to the desired number of eigenvectors to be produced:

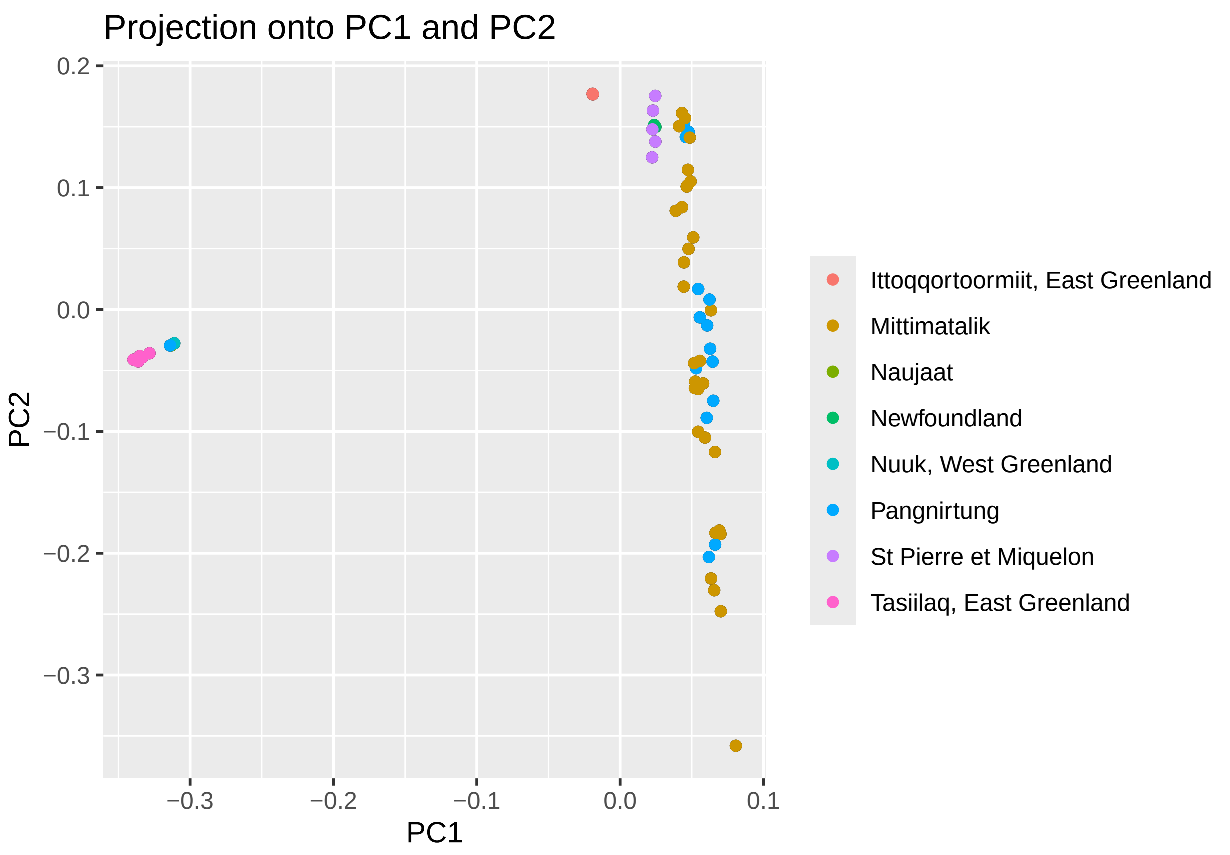
```r
pca <- pcadapt(input = snp_data_pca, K = 18)
```

Plot screeplot and PCA:

```r
# Quick Screeplot:
plot(pca, option = "screeplot")
```



```
# Quick biplot:
plot(pca, option = "scores", pop = sample_info$location_name, labels = sample_info$genome_sample_ID)
```

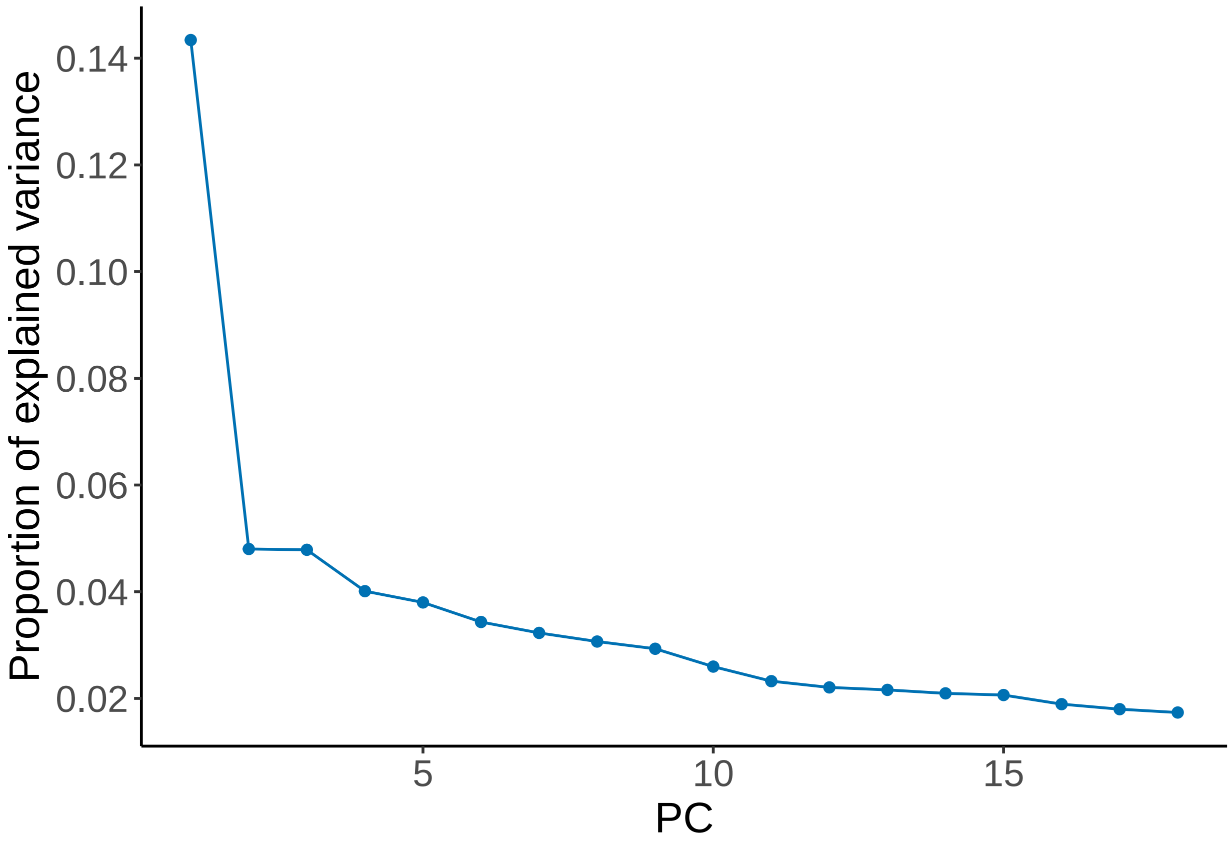


Examine PCA scores, loadings, and z-scores, and calculate proportion variance for first few eigenvectors:

```
# scores:
scores <- as.data.frame(pca$scores)

# loadings:
loadings <- as.data.frame(pca$loadings)

# z-scores:
z_scores <- as.data.frame(pca$zscores)

# proportion variance
proportion <- as.data.frame(pca$singular.values)
proportion$squared <- proportion$`pca$singular.values`* proportion$`pca$singular.values`
prop_var <- as.data.frame(proportion$squared)
PC1_proportion <- (round(prop_var[1,], digits=4))*100
PC2_proportion <- (round(prop_var[2,], digits=4))*100
```

Make screeplot nicer:

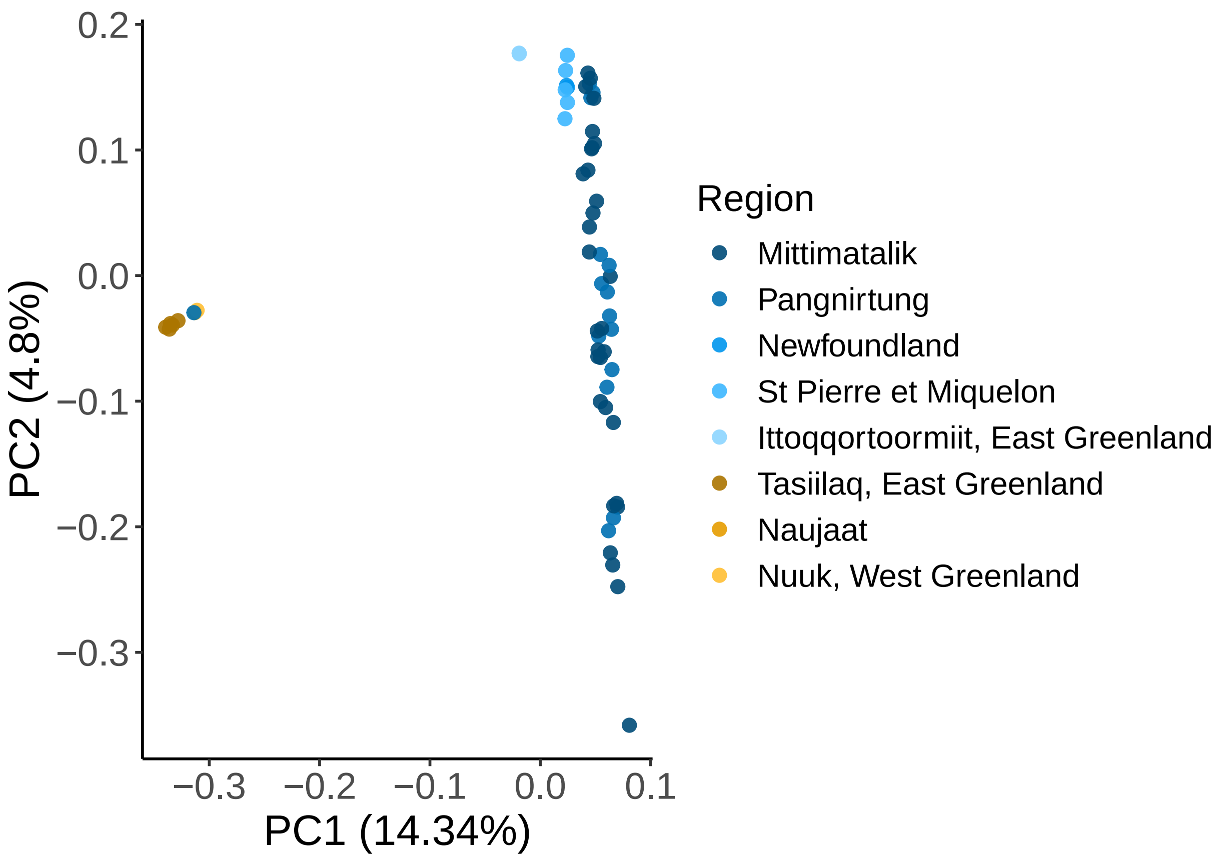
```
prop_var$num <- 1:nrow(prop_var)

scree <- ggplot(data=prop_var, aes(x=num, y=prop_var$`proportion$squared`)) +
  geom_point(col = "#0071b3") +
  geom_line(col = "#0071b3") +
  scale_y_continuous(breaks = seq(0,0.16,0.02)) +
  ylab("Proportion of explained variance") +
  xlab("PC") +
  theme_classic() +
  theme(axis.title = element_text(size = 16),
        axis.text = element_text(size = 14))

scree
```



```
#ggsave("scree_plot_pca_allSamples.png", plot = scree, width=6, height=4.5, dpi=300)
```

Make nice PCA:

```
# Order locations for plotting:
sample_info$location_name <- factor(sample_info$location_name, levels = c("Mittimatalik", "Pangnirtung"

# Set cols:
#           Pond        Pang        Nfld        SP&M        SCores     Tasiilaq   Naujaat    Nuuk
cols <- c("#004c78", "#0071b3", "#0096ee", "#3db8ff", "#8cd5ff","#ab7500", "#e69d00", "#ffbf35")

evec <- cbind(sample_info$genome_sample_ID, scores)
colnames(evec)[1] <- "sample"

ggplot(data=evec, aes(x=V1,y=V2))+
  geom_point(aes(color=sample_info$location_name),size=2, alpha=0.9)+
  theme_classic()+
  xlab(paste("PC1 (", PC1_proportion, "%)", sep=""))+
  ylab(paste("PC2 (", PC2_proportion, "%)", sep=""))+
  #geom_text_repel(aes(label=sample_info$genome_sample_ID), size=2)+
  scale_color_manual(values = cols, name = "Region") +
  theme(axis.title = element_text(size = 16),
        axis.text = element_text(size = 14),
        legend.text = element_text(size = 12),
        legend.title = element_text(size = 14))
```



```
#ggsave("pca_closekinremoved_Dec2024.png", width=8, height=4.5, dpi=300)
```

We see the two populations as before, though appears to be some grouping of the SPM & Newfoundland whales, and then the Scoresbysund whales.

Assign individuals to populations based on the PCA:

```
pc1_for_grouping <- evec %>%
  dplyr::select(sample, V1) %>%
  mutate(genetic_group = if_else(V1 > -0.1, "ECAG1", "ECAG2")) %>%
  rename("genome_sample_ID" = "sample")

sample_info_genetic_group <- sample_info %>%
  left_join(pc1_for_grouping, by = "genome_sample_ID")
# saveRDS(sample_info_genetic_group, "sample_info_genetic_groups.rds")
```

FST

This section run on bio server.

Load snps with vcfR:

```
# # Populate the ID column of VCF data:
# snps <- read.vcfR("killerwhale3_snps.ID.filter1.miss.biallel.min100kb.autosomes.hwe.maf.LDprunedr08.k
#
```

```
# # add IDs:
# snps <- addID(snps, sep = "_")
#
# # Convert vcfR objects to objects supported by other R packages (such as StAMPP)
# snp_data_fst <- vcfR2genlight(snps)
```

Add pop info to the snp data:

```
# sample_info <- readRDS("sample_info_genetic_groups.rds")
# snp_data_fst@pop <- as.factor(sample_info$genetic_group)
# fst_snps <- snp_data_fst
```

Calculate fst:

```
# Calculate Fst - run on server
# kws_fst <- stamppFst(fst_snps, nboots = 100, percent = 95, nclusters = 45)
# saveRDS(kws_fst, "kw_fst.rds")
# write.csv(kws_fst, "kw_fst.csv")

kws_fst <- readRDS("kw_fst.rds")
kws_fst
```

```
## $Fsts
##           ECAG2 ECAG1
## ECAG2        NA    NA
## ECAG1 0.2255297    NA
##
## $Pvalues
##       ECAG2 ECAG1
## ECAG2    NA    NA
## ECAG1     0    NA
##
## $Bootstraps
##   Population1 Population2         1         2         3         4         5
## 1       ECAG2       ECAG1 0.2249108 0.2249225 0.2249378 0.2249728 0.2249873
##           6         7         8         9        10       11        12
## 1 0.2250544 0.2250547 0.2250622 0.2250625 0.2251055 0.225135 0.2251435
##          13        14        15        16        17        18        19
## 1 0.2251633 0.2251911 0.2251953 0.2252007 0.2252157 0.2252212 0.2252425
##          20        21        22        23        24        25        26
## 1 0.2252695 0.2252794 0.2252827 0.2253015 0.2253109 0.2253123 0.2253195
##          27       28        29        30        31        32        33       34
## 1 0.2253262 0.225332 0.2253355 0.2253469 0.2253516 0.2253525 0.2253653 0.225374
##          35        36        37        38        39        40        41
## 1 0.2253964 0.2254032 0.2254043 0.2254053 0.2254091 0.2254107 0.2254128
##          42        43        44        45        46        47        48
## 1 0.2254176 0.2254581 0.2254594 0.2254665 0.2254749 0.2254831 0.2254854
##          49        50        51        52        53       54        55
## 1 0.2254893 0.2254928 0.2254929 0.2254953 0.2254985 0.225502 0.2255037
##          56        57        58        59        60        61        62
## 1 0.2255071 0.2255124 0.2255187 0.2255278 0.2255402 0.2255443 0.2255579
##          63        64        65        66        67        68        69
## 1 0.2255658 0.2255664 0.2255731 0.2255744 0.2255771 0.2256102 0.2256142
##          70        71        72        73        74        75        76
## 1 0.2256181 0.2256228 0.2256254 0.2256282 0.2256462 0.2256651 0.2256673
##          77        78        79        80        81        82        83
## 1 0.2256823 0.225684 0.2256923 0.2256955 0.2257016 0.2257129 0.2257152
##          84        85        86        87        88        89        90
## 1 0.2257301 0.2257338 0.2257382 0.2257623 0.2257863 0.2258061 0.2258139
##          91        92        93        94        95        96        97
## 1 0.2258703 0.2258711 0.2258745 0.2259091 0.2259774 0.2259897 0.2260102
##          98        99       100 Lower bound CI limit Upper bound CI limit
## 1 0.2260897 0.2261964 0.2262284            0.2249378            0.2260102
##   p-value       Fst
## 1       0 0.2255297
```