unknown-species-id-template

June 2023 Anita Wray Code originally written by Diana B

Walkthrough of species id for unknown samples:

Using the rds file output from microhaplot: 1. read in rds files 2. apply read depth filters 3. apply allele balance filter

```
knitr::opts_knit$set(root.dir = "~/Desktop/RE_BS/stock review 2025 work/")
```

```
source("R/rockfish-funcs2.R")

# get the names of the files
fdf <- read.table("rds-file-list.txt", stringsAsFactors = FALSE, header = TRUE) %>%
  tibble::as_tibble()

dir <- "microhaplot/"



# cycle over them, read them and add the gtseq_run column on each.
# at the end, bind them together.
genos_long <- lapply(1:nrow(fdf), function(i) {
  message("Working on ", fdf$file[i])
  call_genos_from_haplotRDS(path = file.path(dir, fdf$file[i])) %>%
    mutate(gtseq_run = fdf$gtseq_run[i]) %>%
    select(gtseq_run, everything())
}) %>%
  bind_rows()
```

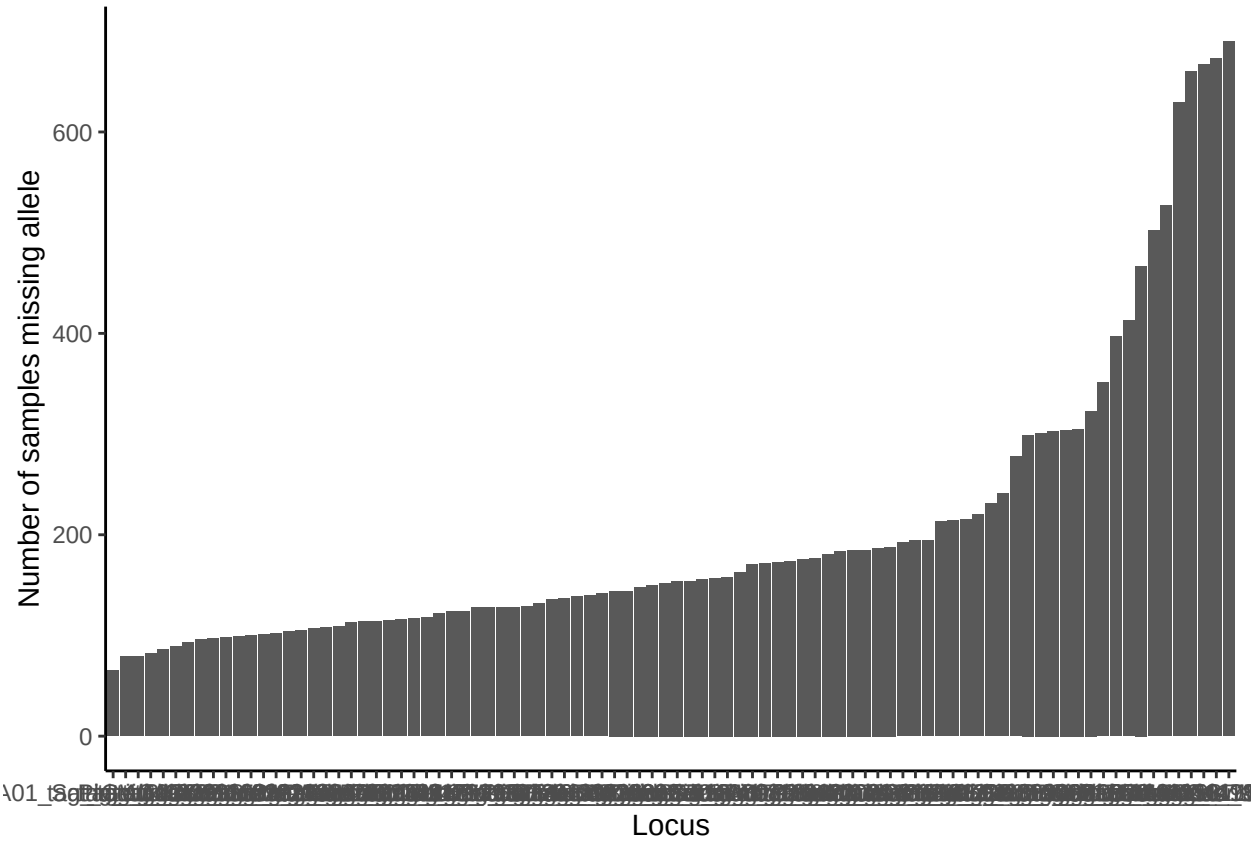
```
## Working on DIANA--target_fastas--diana-fasta--snps4test.rds

## Joining with `by = join_by(id, locus, rank)`
```

```
#genos_long$id <- gsub('-','',genos_long$id)

# we go ahead and save it in data/processed, with xz compression
saveRDS(genos_long, file = "processed/called_genos_anita.rds", compress = "xz")


#### In the end, let us get a data frame that includes genotypes for all the individuals  ####
# and which explicitly has NAs in places where data are missing, and also
# has the NMFS_DNA_ID on there
genos_long_explicit_NAs <- genos_long %>%
  select(gtseq_run, id) %>%
  unique() %>%
```

```
  unite(col = gid, sep = "_", gtseq_run, id) %>%
  select(gid) %>%
  unlist() %>%
  unname() %>%
  expand.grid(gid = ., locus = unique(genos_long$locus), gene_copy = 1:2, stringsAsFactors = FALSE) %>%
  tibble::as_tibble() %>%
  separate(gid, into = c("gtseq_run", "id"), convert = TRUE, sep = '_') %>%
  left_join(., genos_long) %>%
  arrange(gtseq_run, id, locus, gene_copy)
```

```
## Joining with `by = join_by(gtseq_run, id, locus, gene_copy)`
```

```
# and then save that
saveRDS(genos_long_explicit_NAs, file = "processed/called_genos_na_explicit_anita.rds", compress = "xz")

final_df <- data.frame(matrix(NA, nrow = 90, ncol = 2))
colnames(final_df) <- c("locus_name", 'count of missing alleles')

unique_locus_names <- unique(genos_long$locus)
j <- 1
for (i in unique_locus_names){
  temp_df <- subset(genos_long, genos_long$locus == i)
  final_df[j,1] <- i
  final_df[j,2] <- sum(is.na(temp_df$allele))
  j <- j +1
}

ggplot(data = final_df, aes(x = fct_reorder(locus_name, `count of missing alleles`), y = `count of missi
  geom_col() +
  ylab('Number of samples missing allele')+
  xlab('Locus')+
  theme_classic()
```



Using those genotypes...

```
genos_long_explicit_NAs %>%
  group_by(gtseq_run, id) %>%
  tally()
```

```
## # A tibble: 721 x 3
## # Groups:   gtseq_run [1]
##    gtseq_run id                       n
##    <chr>     <chr>                <int>
##  1 gtseq3    BREVISPINIS-UW114059   180
##  2 gtseq3    BREVISPINIS-UW119935   180
##  3 gtseq3    BREVISPINIS-UW153443   180
##  4 gtseq3    BREVISPINIS-UW153444   180
##  5 gtseq3    BREVISPINIS-UW157098   180
##  6 gtseq3    EOS-UW114068           180
##  7 gtseq3    GILLI-UW202792         180
##  8 gtseq3    GILLI-UW202802         180
##  9 gtseq3    GILLI-UW202913         180
## 10 gtseq3    GILLI-UW202915         180
## # i 711 more rows
```

Look at missing data: 180 gene copies total (90 loci x2)

```
ind_to_toss <- genos_long_explicit_NAs %>%
  group_by(gtseq_run, id) %>%
  filter(is.na(allele)) %>% # missing data
  tally() %>%
  arrange(desc(n)) %>% # remove samples with >30% missing data
  filter(n > 54)

##write those removed samples for the PCA
write.csv(ind_to_toss, file = "~/Desktop/RE_BS/stock review 2025 work/removed_samples_rubias_04_11_25.cs

# remove those from the df
genos_ind_filtered <- genos_long_explicit_NAs %>%
  anti_join(., ind_to_toss)
```

```
## Joining with `by = join_by(gtseq_run, id)`
```

```
table(length(unique(genos_ind_filtered$id)))
```

```
##
## 524
##   1
```

Load baseline data

```
# baseline data - curated, 997 indivs
baseline <- readRDS("new_baseline_data/processed/sebastes_spp_id_baseline_haplotypes_04_17.rds")

# remove the 6 loci that had HWE and other issues
to_remove <- read_csv("data/loci_to_remove.csv")
```

```
## Rows: 6 Columns: 1
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): locus
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
baseline90 <- baseline %>%
 anti_join(., to_remove)
```

```
## Joining with `by = join_by(locus)`
```

```
# remind myself which species are in the baseline:
baseline90 %>%
  select(collection) %>%
  unique() %>%
  arrange()
```

```
## # A tibble: 60 x 1
##    collection
##    <chr>
##  1 BREVISPINIS
##  2 GILLI
##  3 crocotulus
##  4 HELVOMACULATUS
##  5 LENTIGINOSUS
##  6 MACDONALDI
##  7 aleutianus
##  8 alutus
##  9 auriculatus
## 10 aurora
## # i 50 more rows
```

```
tossers <- baseline90 %>%
  select(indiv, gtseq_run, id) %>%
  unique() %>%
  group_by(indiv) %>%
  tally() %>%
  filter(n >1)

baseline90_one_each <- baseline90 %>%
  anti_join(., tossers)
```

```
## Joining with `by = join_by(indiv)`
```

```
# baseline data - curated, 1024 indivs
baseline_spp_info <- baseline90_one_each %>%
  select(sample_type, repunit, collection, indiv, gtseq_run, id, species) %>%
  unique()
baseline_spp_info$gtseq_run <- as.character(baseline_spp_info$gtseq_run)
```

```
# slim that down to just the matching field with the unknowns
for_alleidx <- baseline90_one_each %>%
 select(-indiv, -c(1:3, 12:13), -species)

for_alleidx$gtseq_run <- as.character(for_alleidx$gtseq_run)
```

```
genos_ind_filtered <- genos_ind_filtered %>%
  subset(!id %in% for_alleidx$id)


# merge the two dataframes
merged_df <- bind_rows(for_alleidx, genos_ind_filtered)

# first make integers of the alleles
alle_idxs <- merged_df %>%
  dplyr::select(gtseq_run, id, locus, gene_copy, allele) %>%
  group_by(locus) %>%
  mutate(alleidx = as.integer(factor(allele, levels = unique(allele)))) %>%
  ungroup() %>%
```

```
  arrange(gtseq_run, id, locus, alleidx) # rubias can handle NA's, so no need to change them to 0's


# and spread the alleles
two_col <- alle_idxs %>%
  group_by(id, locus) %>%
  unite(loc, locus, gene_copy, sep = ".") %>%
  ungroup() %>%
  select(-allele) %>%
  pivot_wider(names_from = loc, values_from = alleidx)
```

add back on info for reference and make two-column format for rubias

```
# baseline
reference <- two_col %>%
  left_join(., baseline_spp_info) %>%
  filter(!is.na(species)) %>%
  select(-gtseq_run, -id, -species) %>%
  select(sample_type, repunit, collection, indiv, everything())
```

```
## Joining with `by = join_by(gtseq_run, id)`
```

```
# mixture
rubias_mix <- two_col %>%
  anti_join(., baseline_spp_info) %>%
  mutate(sample_type = "mixture", collection = "krista", repunit = NA) %>%
  select(sample_type, repunit, collection, everything()) %>%
  unite(gtseq_run, id, col = "indiv", sep = "_") #removed gtseq_run, id
```

```
## Joining with `by = join_by(gtseq_run, id)`
```

```
#Pull out two known sunset individual
croc_known <- c('gtseq3_H-14-MI-V0270','gtseq3_H-14-MI-V0272')

croc <- subset(rubias_mix, rubias_mix$indiv %in% croc_known)
croc$sample_type <- 'reference'
croc$repunit <- 'crocotulus'
croc$collection <- 'crocotulus'

#Add it to the reference
reference <- rbind(reference, croc)

#Remove it from the mixture
rubias_mix <- subset(rubias_mix, !rubias_mix$indiv %in% croc_known)
```

Mixture assignment with rubias

```
rubias_output <- infer_mixture(reference = reference, mixture = rubias_mix, gen_start_col = 5)
```

```
## Collating data; compiling reference allele frequencies, etc.   time: 0.57 seconds
## Computing reference locus specific means and variances for computing mixture z-scores   time: 0.20 se
## Working on mixture collection: krista with 505 individuals
##   calculating log-likelihoods of the mixture individuals.   time: 0.10 seconds
##   performing 2000 total sweeps, 100 of which are burn-in and will not be used in computing averages
##   tidying output into a tibble.   time: 0.12 seconds
```

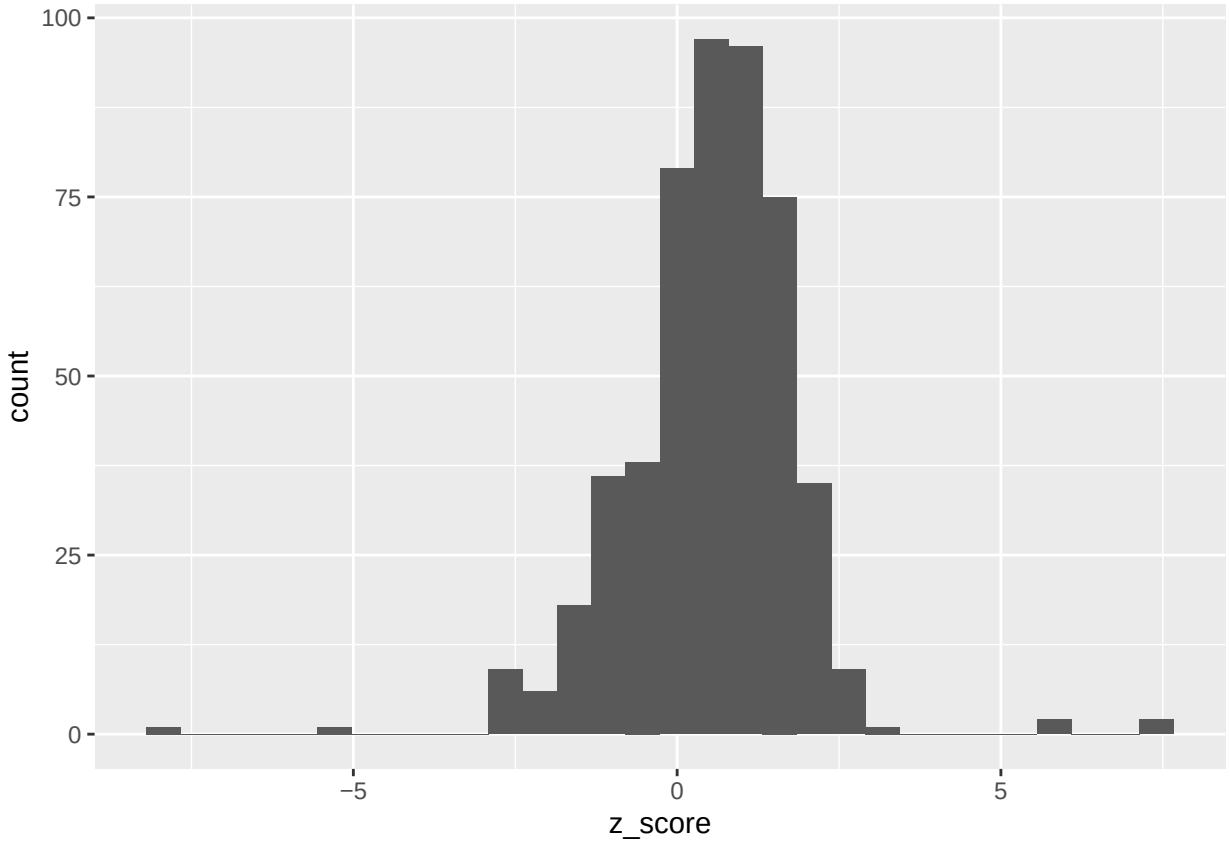
```
# take the top output for each sample
top_assign <- rubias_output$indiv_posteriors %>%
  group_by(indiv) %>%
  slice_max(., order_by = PofZ)

nonREBS <- top_assign %>%
  subset(repunit != 'melanostictus') %>%
  subset(repunit != 'aleutianus')

table(nonREBS$repunit)
```

```
##
##    auriculatus       borealis    BREVISPINIS  chlorostictus       diaconus
##              1              1              1              5              1
##      diploproa      elongatus        ensifer HELVOMACULATUS     MACDONALDI
##              1              1              7              3              1
##       rosaceus     ruberrimus      rufinanus       umbrosus      zacentrus
##              9              1              1              1              1
```

```
df <- apply(top_assign,2,as.character)
write.csv(df, file = '~/Desktop/RE_BS/stock review 2025 work/rubias_output_04_17_2025.csv')
```

Check on z-scores:

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




```
## Saving 6.5 x 4.5 in image
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```