```
# Recode a fasta alignment for PCA analysis, collapsing any
redundant columns (@col_unique)
#
# Note regarding input file conventions:
#
# There is assumed to be no missing data, no ambiguous calls, and
no genotypes beyond diploid.
# Uppercase "A, C, G, T" represent a homozygote of two
nucleotides
# Uppercase "K, Y, W, S, R, M" represent a heterozygote with two
nucleotides
# Lowercase "a, c, g, t" represent a heterozygote with one gap
allele and one nucleotide
# "-" represents a homozygote of two gaps
#
#
# usage:
# perl AlleleCoder.pl infile.fasta > 123.txt
#
# AMB 12/13/07
#
# This program is free software. It comes without any warranty,
to
# the extent permitted by applicable law. You can redistribute it
# and/or modify it under the terms of the WTF Public License,
# Version 2, as published by Sam Hocevar. See
# http://sam.zoy.org/wtfpl/COPYING for more details.

use strict;
use warnings;
use Bio::Align::AlignI;
use Bio::AlignIO;
use Bio::SeqIO;

use Bio::LocatableSeq;
use Tie::IxHash; # Make the hash print out in the order it was
filled

my $fna = $ARGV[0]; # input fasta format file.  See note above
for conventions regarding heterozygous genotypes.

my %columns=();
my $firstone=1;
my $numseq;
my $numcol;
my %matrix;
tie %matrix, "Tie::IxHash";
my %out;
tie %out, "Tie::IxHash";


my $in  = Bio::AlignIO->new(-file   => $fna ,
```

```
                    -format => 'fasta');
my $aln = $in->next_aln();

$numseq = $aln->no_sequences;
$numcol = $aln->length;

for (my $ctr = 1; $ctr < $numcol+1; $ctr++) {

  my $A = 0;
  my $C = 0;
  my $G = 0;
  my $T = 0;
  my $GAP = 0;
  my $HET=0;

  my $AA=0;
  my $AC=0;
  my $AG=0;
  my $AT=0;
  my $CC=0;
  my $CG=0;
  my $CT=0;
  my $GG=0;
  my $GT=0;
  my $TT=0;
  my $AGAP=0;
  my $CGAP=0;
  my $GGAP=0;
  my $TGAP=0;
  my $GAPGAP=0;

  # count up all the residues in this column to find the most
common allele
  foreach my $seq ($aln->each_seq) {
    my $res = $seq->subseq($ctr, $ctr);
    if ($res eq "A") {
      $A = $A +2;
      $AA++;
    } elsif ($res eq "C") {
      $C = $C +2;
      $CC++;
    } elsif ($res eq "G") {
      $G = $G +2;
      $GG++;
    } elsif ($res eq "T") {
      $T = $T +2;
      $TT++;
    } elsif ($res eq "-") {
      $GAP = $GAP +2;
      $GAPGAP++;
    } elsif ($res eq "a") {
      $HET++;
```

```
    $A++;
    $GAP++ ;
    $AGAP++;
  } elsif ($res eq "c") {
    $HET++;
    $C++;
    $GAP++;
    $CGAP++;
  } elsif ($res eq "g") {
    $HET++;
    $G++;
    $GAP++;
    $GGAP++;
  } elsif ($res eq "t") {
    $HET++;
    $T++;
    $GAP++;
    $TGAP++;
  } elsif ($res eq "K") {
    $HET++;
    $G++;
    $T++;
    $GT++;
  } elsif ($res eq "Y") {
    $HET++;
    $C++;
    $T++;
    $CT++;
  } elsif ($res eq "W") {
    $HET++;
    $T++;
    $A++;
    $AT++;
  } elsif ($res eq "S") {
    $HET++;
    $C++;
    $G++;
    $CG++;
  } elsif ($res eq "R") {
    $HET++;
    $A++;
    $G++;
    $AG++;
  } elsif ($res eq "M") {
    $HET++;
    $A++;
    $C++;
    $AC++;
  }
}

# calculate the most common allele among A C G T GAP and set
```

```
the hash of heterozygote characters accordingly
  my $mostcommon = "uninitialized";
  my %hets;

  if (
      ($A >= $C) &&
      ($A >= $G) &&
      ($A >= $T) &&
      ($A >= $GAP)
     ) {
    $mostcommon = 'A';
    $hets{a}=1;
    $hets{M}=1;
    $hets{R}=1;
    $hets{W}=1;
  } elsif (
         ($C >= $A) &&
         ($C >= $G) &&
         ($C >= $T) &&
         ($C >= $GAP)
        ) {
    $mostcommon = 'C';
    $hets{c}=1;
    $hets{M}=1;
    $hets{S}=1;
    $hets{Y}=1;
  } elsif (
         ($G >= $A) &&
         ($G >= $C) &&
         ($G >= $T) &&
         ($G >= $GAP)
        ) {
    $mostcommon = 'G';
    $hets{g}=1;
    $hets{R}=1;
    $hets{S}=1;
    $hets{K}=1;
  } elsif (
         ($T >= $A) &&
         ($T >= $C) &&
         ($T >= $G) &&
         ($T >= $GAP)
        ) {
    $mostcommon = 'T';
    $hets{t}=1;
    $hets{W}=1;
    $hets{Y}=1;
    $hets{K}=1;
  } elsif (
         ($GAP >= $A) &&
         ($GAP >= $C) &&
         ($GAP >= $G) &&
```

```
        ($GAP >= $T)
       ) {
    $mostcommon = '-';
    $hets{a}=1;
    $hets{c}=1;
    $hets{g}=1;
    $hets{t}=1;
  }

  my $code;

  # recode each residue into a numeric value and save in a hash
of strings representing the alignment
  foreach my $seq ($aln->each_seq) {
    my $res = $seq->subseq($ctr, $ctr);
    if ($res eq $mostcommon) {
      $code = 1;
    } elsif (exists $hets{$res}) { # is this one of the
designated heterozygotes of the most common allele?
      $code = 2;
    } else {
      $code = 3;
    }
    if (exists $matrix{$seq->display_id}) {
      $matrix{$seq->display_id} = $matrix{$seq->display_id}.
$code;
    } else {
      $matrix{$seq->display_id} = $code;
    }
  }
}                     # for $ctr;

# collapse invariant and non-unique contiguous columns to
decrease PCA computing time
my $column="";
my %seen;
my @col_unique;
my %identical;


for (my $ctr = 0; $ctr < $numcol; $ctr++) {

  foreach my $key (keys %matrix) {
    $column = $column.substr($matrix{$key},$ctr,1);
  }
  if (not exists ($seen{$column}) && ($column !~ /^[1]$/)) {
    $seen{$column}=$ctr;
    push (@col_unique,$ctr);
  } else {
    $identical{$ctr}=$seen{$column};
  }
  $column="";
```

```
}



# store the collapsed matrix in a R-compatible format
$out{taxon} = "";
foreach my $column (@col_unique) {
  $out{taxon} = $out{taxon}." ".$column;
  foreach my $key (keys %matrix) {
    if (exists $out{$key}) {
      $out{$key} = $out{$key}." ".substr($matrix{$key},
$column,1);
    } else {
      $out{$key} = substr($matrix{$key},$column,1);
    }
  }
}

# print the output
foreach my $key (keys %out) {
  print $key,"\t",$out{$key},"\n";
}
```