Analysis of Human Origins (plus Pacific) data

Alejandro Ochoa and John D. Storey

2019-05-01

Introduction

In this R markdown file we demonstrate the basic analysis of Human Origins dataset presented in our paper "New kinship and FST estimates reveal higher levels of differentiation in the global human population" by Ochoa and Storey. We rely exclusively on the data files provided in this repository (https://github.com/StoreyLab/human_differentiation_analysis), which are the public subset of Human Origins and the Pacific datasets.

R package dependencies

To run this code, you will need the following packages:

```
library(BEDMatrix)    # to load genotype matrices
library(popkin)       # to estimate "population kinship" matrices
library(genio)        # to parse FAM file
library(readr)        # to read additional tables
library(RColorBrewer) # for colors
```

Estimate the kinship matrix

Load the genotype matrix (smartly, actually doesn't load into memory):

```
X <- BEDMatrix('data/human_origins_and_pacific_public', simple_names = TRUE)
```

```
## Extracting number of samples and rownames from human_origins_and_pacific_public.fam...
```

```
## Extracting number of variants and colnames from human_origins_and_pacific_public.bim...
```

This function from `genio` loads individual annotations (in plink FAM format), including subpopulations.

```
fam <- read_fam('data/human_origins_and_pacific_public')
```

```
## Reading: data/human_origins_and_pacific_public.fam
```

Now estimate the kinship matrix. We rely on the subpopulation labels to estimate the minimum kinship value to be treated as unrelated (zero). This step takes about 7 minutes.

```
kinship <- popkin(X, subpops = fam$fam)
```

Sub-subpopulation annotations

Before plotting, we need to reorder individuals so things look reasonable. To achieve this and other tasks, we load an additional table with subpopulation annotations.

```
info <- read_tsv(
    'data/human_origins_and_pacific_public_subpops.txt',
    col_types = 'ccddc'
)
# inspect
info
```

```
## # A tibble: 248 x 5
##    subsubpop     subpop      x     y country
##    <chr>         <chr>   <dbl> <dbl> <chr>
##  1 Ju_hoan_South SAfrica  20.7 -21.2 Botswana
##  2 Ju_hoan_North SAfrica  21.5 -18.9 Namibia
##  3 Taa_West      SAfrica  20.3 -23.6 Botswana
##  4 Taa_East      SAfrica  22.8 -24.2 Botswana
##  5 Taa_North     SAfrica  22.4 -23   Botswana
##  6 Naro          SAfrica  21.6 -22   Botswana
##  7 Gui           SAfrica  23.3 -21.5 Botswana
##  8 Hoan          SAfrica  23.4 -24   Botswana
##  9 Xuun          SAfrica  19.7 -18.7 Namibia
## 10 Gana          SAfrica  23.4 -21.7 Botswana
## # ... with 238 more rows
```

This table has one row per sub-subpopulation, for each of the 248 sub-subpopulations in the full dataset (including singleton and other sub-subpopulations removed in our publication). The table contains geographical coordinates (`x` and `y`) and country annotations, which we won't use here. These sub-subpopulations are grouped into 11 subpopulations (second column) that represent continental-level regions. We will use the `subsubpop` and `subpop` labels, and their order in this table, to sort individuals and create plots. The order and subpopulation assignments were defined manually, informed by geography and refined iteratively using the kinship plots to keep the most similar sub-subpopulations together.

Reorder individuals using sub-subpopulation annotations

Let's reorder individuals. First we ensure that every sub-subpopulation in the genotype table is present in the annotations table:

```
stopifnot(
    fam$fam %in% info$subsubpop
)
```

Now we reorder individuals so their sub-subpopulations appear in the desired order

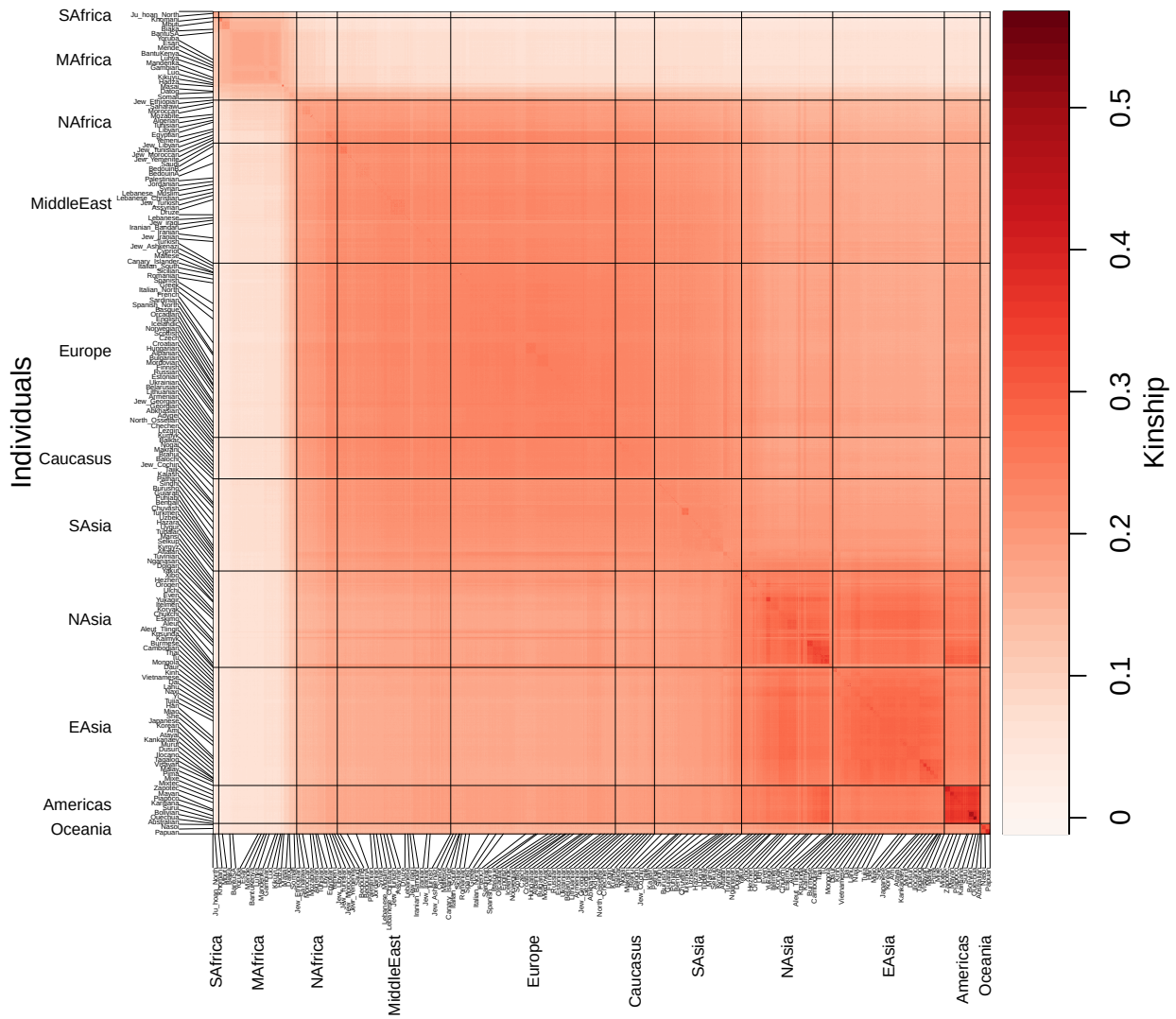
```
# the rank ties every individual in the same sub-subpopulation
fam_rank <- match(fam$fam, info$subsubpop)
# this order breaks ties by order of appearance
indexes <- order(fam_rank)
# before reordering, let's map subpopulations onto the FAM table
fam$subpop <- info$subpop[fam_rank]
# apply order to kinship matrix (both dimensions)
kinship <- kinship[indexes, indexes]
# and to individual annotations (rows only)
fam <- fam[indexes, ]
```

Kinship matrix plot

We are ready to visualize the estimated kinship matrix:

```
# line position for outer labels
line <- 3
# inner margins
par(mar = c(0, 0, 0, 0) + 0.2)
# outer margins
par(oma = c(line + 2.2, line + 3.2, 0, 3))

# main plot with all labeling bells and whistles
plot_popkin(
    # plot inbreeding along diagonal
    kinship = inbr_diag( kinship ),
    # leave a lot of space for labels
    ylab_line = line + 2,
    # two levels of labels
    labs = cbind(fam$fam, fam$subpop),
    labs_cex = c(0.25, 0.6),
    labs_las = 2,
    labs_line = c(1, line),
    labs_lwd = c(0.1, 0.5),
    labs_sep = c(FALSE, TRUE),
    labs_even = c(TRUE, FALSE)
)
```



Compared to the full dataset, this version limited to the public data has very few samples from `SAfrica` and `Oceania`.

Weights and FST

This function calculates weights for individuals such that:

- every subpopulation has equal weight, and
- every sub-subpopulation has equal weight within its subpopulation.

```
# wrap code around function
get_weights_human_origins <- function(fam, info) {
    # this processing will be wrong if info contains more subpops than are present in fam,
    # so let's make them agree internally
    info <- info[ info$subsubpop %in% fam$fam, ]

    # get counts for sub-subpops and subpops
    # number of individuals in each sub-subpopulation
    subsubpop_to_counts <- table( fam$fam )
```

```
    # number of sub-subpopulations in each subpopulation
    subpop_to_counts <- table( info$subpop )
    # number of unique subpopulations
    K <- length( subpop_to_counts )

    # construct weights!
    weights <- 1 / (K * subsubpop_to_counts[fam$fam] * subpop_to_counts[fam$subpop] )
}
# apply function
weights <- get_weights_human_origins(fam, info)
```

This is the kinship estimate we obtain:

```
fst_estimate <- fst( kinship, weights )
# inspect
fst_estimate
```

```
## [1] 0.2622629
```

Nice! The value we estimated in the publication, using the full dataset, was 0.260.

Density of inbreeding coefficients

TODO: code needs to be cleaned up!

```
fstTitle <- expression(bold(paste(F[ST], ' estimates')))
fstVals <- fst_estimate
fstMethods <- "New"
fstLtys <- 1
fstCols <- 'black'

subpops <- unique(info$subpop)
i2subpop <- fam$subpop # subpopulation assignments
cols <- rev(brewer.pal(length(subpops), 'Spectral')) # nice palette for subpopulations # matches pallet

# figure out figure ranges
inbrs <- inbr( kinship )
myXlim <- range(0, inbrs)
myYlim <- 0 # will grow below

# collect densities of interest
sp2d <- list() # maps each subpopulation to its inbreeding density
for (sp in subpops) {
    is <- i2subpop == sp # filter of data
    inbri <- inbrs[is] # data to make density of
    wi <- weights[is] # get subset weights too
    wi <- wi/sum(wi) # renormalize when filters are done!
    # estimate density using these weights
    di <- density( inbri, weights = wi )
    sp2d[[sp]] <- di
    myYlim <- range(myYlim, di$y) # grow if needed
}

lwdBgd <- 3
```

```r
par(mar = c(2.8, 2.7, 0, 0) )
par(lab = c(5, 3, 7))
plot(NA, xlim=myXlim, ylim=myYlim, type='n', xlab='Inbreeding Coefficient', ylab='Density')
# add each subpopulation density
for (sp in subpops) {
    lines(sp2d[[sp]], lwd=lwdBgd) # first pass: black and thicker
}
for (sp in rev(subpops)) {
    lines(sp2d[[sp]], col=cols[match(sp, subpops)])
}
# add each Fst estimate
for (i in 1:length(fstVals)) {
    abline(v=fstVals[i], lty=fstLtys[i], col=fstCols[i])
}
# legends
cexLeg <- 0.7
myLegW <- max(strwidth(subpops, font=2, cex=cexLeg)) # ensure that things work when text is bold!
legend('topright', subpops, title='', lty=1, col='black', cex=cexLeg, text.width=myLegW, bty='n', lwd=lw
legend('topright', subpops, title='Subpopulations', lty=1, col=cols, cex=cexLeg, text.width=myLegW, bty=
myLegW2 <- max(strwidth(fstMethods, font=2, cex=cexLeg)) # ensure that things work when text is bold!
legend('topright', fstMethods, title=fstTitle, lty=fstLtys, col=fstCols, cex=cexLeg, text.width=(2*myLeg
```