Estimate Kinship and F_{ST} under Arbitrary Population Structure with `popkin`

Alejandro Ochoa and John D. Storey

2018-01-26

1 Introduction

The `popkin` ("population kinship") package estimates the kinship matrix of individuals and F_{ST} from their biallelic genotypes. Our estimation framework is the first to be practically unbiased under arbitrary population structures (Ochoa and Storey 2016a; Ochoa and Storey 2016b). Here we briefly summarize the notation and intuition behind the key parameters.

1.1 Kinship and inbreeding coefficients

Kinship and inbreeding coefficients are probabilities of "identity by descent" (IBD) carefully defined elsewhere (Ochoa and Storey 2016a; Ochoa and Storey 2016b). The reference ancestral population T sets the level of relatedness treated as zero (as demonstrated in the sample usage section below). f_j^T is the inbreeding coefficient of individual j when T is the ancestral population, and φ_{jk}^T is the kinship coefficient of the pair individuals j, k when T is the ancestral population. In a structured population we expect most $f_j^T, \varphi_{jk}^T > 0$. If j, k are the parents of l then $f_l^T = \varphi_{jk}^T$, so within a panmictic subpopulation we expect $f_j^T \approx \varphi_{jk}^T$ for $j \neq k$. The "self-kinship" $j = k$ case equals $\varphi_{jj}^T = \frac{1}{2}\left(1 + f_j^T\right)$ rather than f_j^T.

Let $\Phi^T = (\varphi_{jk}^T)$ be the $n \times n$ matrix that contains all kinship coefficients of all individuals in a dataset. The ancestral population T is the most recent common ancestor (MRCA) population if and only if $\min \varphi_{jk}^T = 0$, assuming such unrelated pairs of individuals exist in the dataset. Thus, the only role T plays in our estimates is determining the level of relatedness that is treated as zero.

Note that the diagonal of our estimated Φ^T contains φ_{jj}^T values rather than f_j^T, which is required for statistical modeling applications; however, φ_{jj}^T tends to take on much greater values than φ_{jk}^T for $j \neq k$, while $f_j^T \approx \varphi_{jk}^T$ for $j \neq k$ within panmictic subpopulations (see above), so for visualization we strongly recommend replacing the diagonal of Φ^T with f_j^T values.

1.2 The generalized F_{ST}

F_{ST} is also an IBD probability that equals the mean inbreeding coefficients in a population partitioned into homogeneous subpopulations. We recently generalized the F_{ST} definition to arbitrary population structures—dropping the need for subpopulations—and generalized the partition of "total" inbreeding into "local" inbreeding (due to having unusually closely related parents) and "structural" inbreeding (due to the population structure) (Ochoa and Storey 2016a). The current `popkin` estimates the total kinship matrix Φ^T only; in the future, `popkin` will also extract the structural kinship matrix. However, when all individuals are "locally outbred"—the most common case in population data—F_{ST} is simply the weighted mean inbreeding coefficient:

$$F_{\text{ST}} = \sum_{j=1}^{n} w_j f_j^T,$$

where $0 < w_j < 1, \sum_{j=1}^{n} w_j = 1$ are weights for individuals intended to help users balance skewed samples (i.e. if there are subpopulations with much greater sample sizes than others). The current `popkin` version assumes all individuals are locally outbred in estimating F_{ST}.

1.3 The individual-level pairwise F_{ST}

Another quantity of interest is the individual-level pairwise F_{ST}, which generalize the F_{ST} between two populations to pairs of individuals. Here each comparison between two individuals has a different ancestral population, namely the MRCA population of the two individuals. When individuals are again locally outbred and also locally unrelated, the pairwise F_{ST} is given in terms of the inbreeding and kinship coefficients (Ochoa and Storey 2016a):

$$F_{jk} = \frac{\frac{f_j^T + f_k^T}{2} - \varphi_{jk}^T}{1 - \varphi_{jk}^T}.$$

The `popkin` package also provides an estimator of the pairwise F_{ST} matrix (containing F_{jk} estimates between every pair of individuals).

2 Sample usage

2.1 Input genotype data

The `popkin` function accepts biallelic genotype matrices in three forms:

1. A genotype matrix `X` with values in `c(0,1,2,NA)` only. It is preferable, though not necessary, for `X` to be an integer matrix (with values in `c(0L,1L,2L,NA)` only, see `?storage.mode`). This standard encoding for biallelic SNPs counts reference alleles: 2 is homozygous for the reference allele, 0 is homozygous for the alternative allele, 1 is heterozygous, and NA is missing data. Which allele is the reference does not matter: `popkin` estimates the same kinship and F_{ST} for `X` and `2-X`. By default `popkin` expects loci along rows and individuals along columns (an $m \times n$ matrix); a transposed `X` is handled best by also setting `lociOnCols=TRUE`.

2. BED-formatted data loaded with the `BEDMatrix` package, which `popkin` uses to keep memory usage low. For example, load `myData.bed`, `myData.bim`, `myData.fam` using:

   ```
   library(BEDMatrix)
   X <- BEDMatrix('myData') # note: excluding extension is ok
   ```

   This `BEDMatrix` object is not a regular matrix but `popkin` handles it correctly. Other genotype formats can be converted into BED using plink2 or other software.

3. A function `X(m)` that when called loads the next m SNPs of the data, returning an $m \times n$ matrix in the format of Case 1 above. This option allows direct and memory-efficient processing of large non-BED data, but should be the last resort since users must write their own functions `X(m)` for their custom formats. Try first converting your data to BED and loading with `BEDMatrix`.

2.2 Load and clean sample data

For illustration, let's load the real human data worldwide sample ("HGDP subset") contained in the `lfa` package:

```
library(popkin)
library(lfa) # for hgdp_subset sample data only
X <- hgdp_subset # rename for simplicity
dim(X)
```

```
## [1] 5000  159
```

This data has $m = 5000$ loci and $n = 159$ individuals, and is oriented as popkin expects by default. These samples have labels grouping them by continental subpopulation in colnames(X). To make visualizations easier later on, let’s shorten these labels and reorder to have nice blocks:

```
# shorten subpopulation labels
colnames(X)[colnames(X)==’AFRICA’] <- ’AFR’
colnames(X)[colnames(X)==’MIDDLE_EAST’] <- ’MDE’
colnames(X)[colnames(X)==’EUROPE’] <- ’EUR’
colnames(X)[colnames(X)==’CENTRAL_SOUTH_ASIA’] <- ’SAS’
colnames(X)[colnames(X)==’EAST_ASIA’] <- ’EAS’
colnames(X)[colnames(X)==’OCEANIA’] <- ’OCE’
colnames(X)[colnames(X)==’AMERICA’] <- ’AMR’
# order roughly by distance from Africa
popOrder <- c(’AFR’, ’MDE’, ’EUR’, ’SAS’, ’EAS’, ’OCE’, ’AMR’)
# applies reordering
X <- X[,order(match(colnames(X), popOrder))]
subpops <- colnames(X) # extract subpopulations vector
```

Here’s a quick view of the top left corner of the matrix X with values in 0, 1, 2, and NA (this example has no missing values, but popkin handles them too). This matrix does not preserve the identity of the reference or alternative alleles, but this distinction does not matter for estimating kinship and F_{ST}.

```
X[1:10,1:15]
```

```
##             AFR AFR AFR AFR AFR AFR AFR AFR AFR AFR AFR AFR AFR AFR AFR
## rs4050954     2   2   1   2   1   1   1   2   2   0   0   2   0   2   0
## rs302665      0   0   0   0   0   0   2   0   0   1   0   0   0   0   0
## rs2899446     2   1   2   2   1   2   1   2   2   2   2   1   2   2   2
## rs10069940    2   1   2   2   0   2   0   0   1   0   0   0   0   0   1
## rs6114921     1   0   0   1   2   1   2   2   1   0   1   1   0   0   1
## rs201718      2   2   2   2   2   2   2   1   2   1   2   1   2   2   2
## rs1335715     0   1   0   1   1   0   1   1   2   0   1   1   0   0   0
## rs17006588    2   0   0   1   0   0   1   0   0   1   1   2   0   0   0
## rs3885909     0   0   0   0   0   0   0   1   2   0   0   0   1   0   0
## rs986457      0   1   0   1   1   0   2   1   1   1   1   1   1   1   0
```

Now we’re ready to analyze this data with popkin!

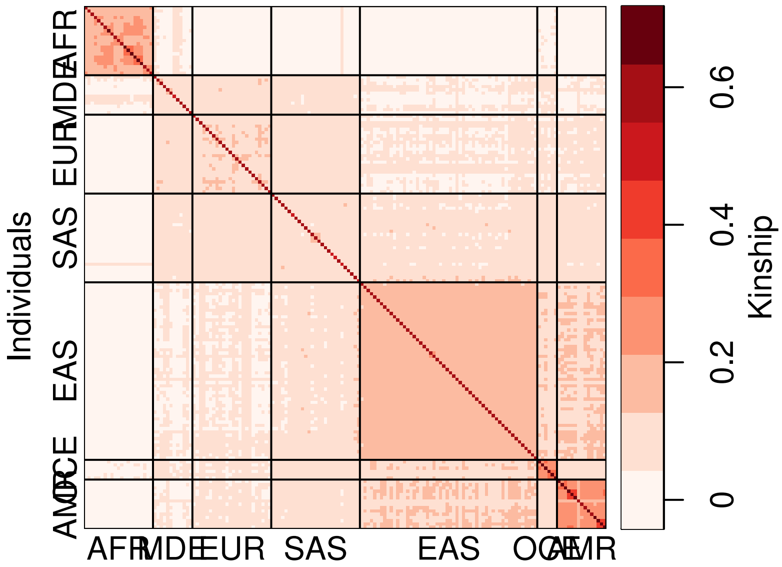
2.3 Estimate and visualize kinship using genotypes and subpopulations

Estimating a kinship matrix requires the genotype matrix X and subpopulation levels used only to estimate the minimum level of kinship. Using the lfa sample data we cleaned in the last subsection, obtaining the estimate is simple:

```
Phi <- popkin(X, subpops)
```

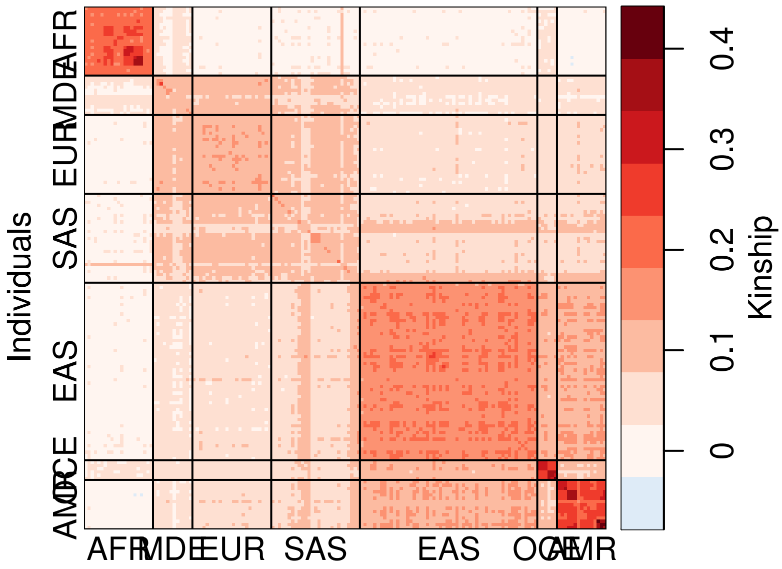
Now let’s visualize the raw kinship matrix estimate:

```
# set outer margin for axis labels (left and right are non-zero)
par(oma=c(0,1.5,0,3))
# set inner margin for subpopulation labels (bottom and left are non-zero), add padding
par(mar=c(1,1,0,0)+0.2)
# now plot!
plotPopkin(Phi, labs=subpops)
```



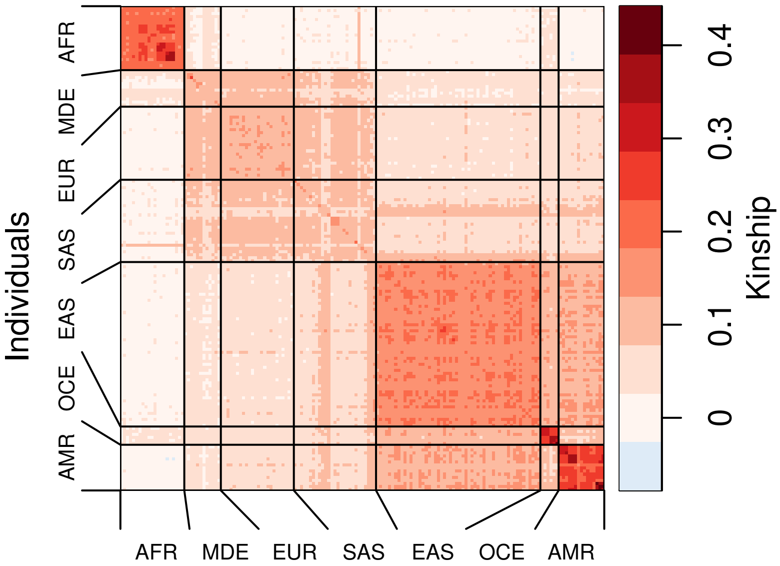
Ignoring the overlapping labels for a moment, this plot shows that self-kinship (the diagonal) is much greater than kinship between different individuals (min $\varphi^T_{jj} \geq 0.5$). It makes more sense to plot inbreeding (f^T_j) values on the diagonal (they are on the same scale as φ^T_{jk} for $j \neq k$), which is achieved using `inbrDiag`:

```
par(oma=c(0,1.5,0,3))
par(mar=c(1,1,0,0)+0.2)
plotPopkin(inbrDiag(Phi), labs=subpops)
```



Now let's tweak the plot. We improve the labeling by setting `labsEven=TRUE`, which arranges the subpopulation labels with equal spacing and adds lines that map to their blocks. To see these new lines, we must move these labels further from the heatmap by setting `labsLine=1`. We shrink the labels with `labsCex=0.7`.

```
par(oma=c(0,1.5,0,3))
# increase margins because labels go farther out
par(mar=c(2,2,0,0)+0.2)
plotPopkin(inbrDiag(Phi), labs=subpops, labsEven=TRUE, labsLine=1, labsCex=0.7)
```



This figure clearly shows the population structure of these worldwide samples, with block patterns that are coherent with serial founder effects in the dispersal of humans out of Africa. Since only $m = 5000$ SNPs are included in this sample, the estimates are noisier than in more complete data (datasets routinely have over 300K SNPs).

This figure also illustrates how subpopulations are used to estimate kinship by `popkin`: they only set the zero kinship as the mean kinship between the two most distant populations, which in this case are AFR and AMR.

2.4 Estimate F_{ST} and individual inbreeding from a kinship matrix

Since F_{ST} is the weighted mean of the inbreeding coefficients, and since some subpopulations are overrepresented in this data (EAS is much larger than the rest), it makes sense to use weights that balance these subpopulations:

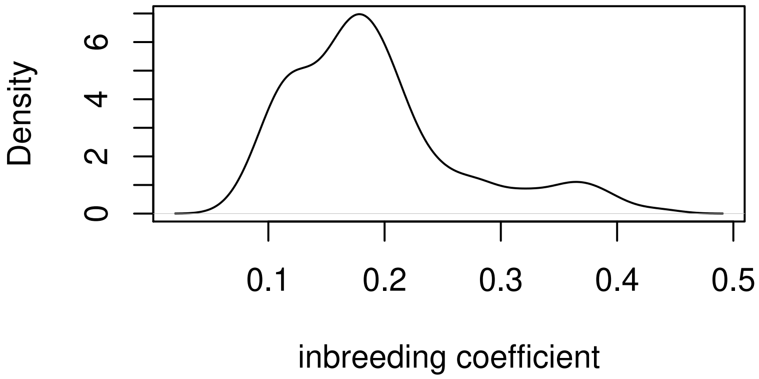
```r
# get weights
w <- weightsSubpops(subpops)
# compute FST!
# Note: don't use the output to inbrDiag(Phi) or FST will be wrong!
fst(Phi, w)
```

```
## [1] 0.2126638
```

If you compare these estimates to those we obtained for Human Origins (Ochoa and Storey 2016a), you'll notice things look a bit different: here F_{ST} is smaller and the kinship within AFR is relatively much higher than within EUR or EAS. Besides containing many fewer SNPs, the SNPs in this HGDP sample were likely biased for common variants in Europeans, which might explain the difference.

We can also extract the vector of inbreeding coefficients from the kinship matrix using `inbr`:

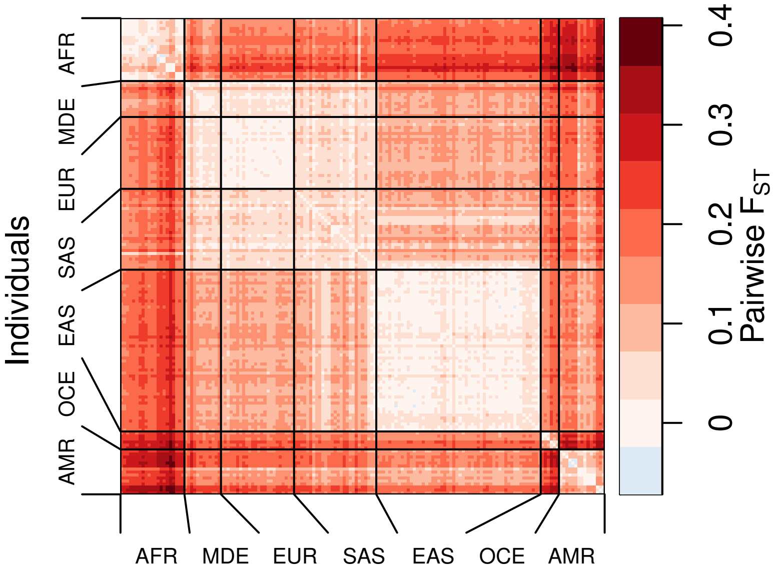
```r
inbrs <- inbr(Phi) # vector of inbreeding coefficients
# quick plot
par(mar=c(4, 4, 0, 0.2) + 0.2) # adjust margins
plot(density(inbrs), xlab='inbreeding coefficient', main='') # see their distribution
```



2.5 Estimate individual-level pairwise F_{ST} matrix from a kinship matrix

We calculate individual-level pairwise F_{ST} estimates from the previous kinship estimates using `pwfst`. Note that the pairwise F_{ST} is a distance between pairs of individuals: approximately zero for individuals in the same population, and increasing for more distant pairs of individuals.

```
pwF <- pwfst(Phi) # compute pairwise FST matrix from kinship matrix
legTitle <- expression(paste('Pairwise ', F[ST])) # fancy legend label
par(oma=c(0,1.5,0,3))
par(mar=c(2,2,0.2,0)+0.2)
# NOTE no need for inbrDiag() here!
plotPopkin(pwF, labs=subpops, labsEven=TRUE, labsLine=1, labsCex=0.7, legTitle=legTitle)
```

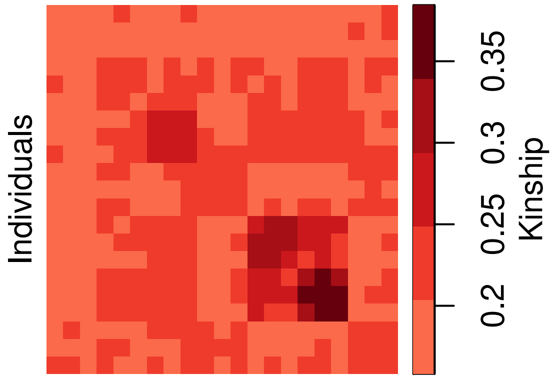


2.6 Rescale kinship matrix in a subset of the data

Suppose now you're interested in one subpopulation, say AFR:

```
indexesAfr <- subpops == 'AFR'
# AFR subset of the kinship matrix
PhiAfr <- Phi[indexesAfr,indexesAfr]
```

```
# kinship matrix plot
par(oma=c(0,1.5,0,3))
par(mar=c(0,0,0,0)+0.2) # zero margins for no labels
plotPopkin(inbrDiag(PhiAfr))
```



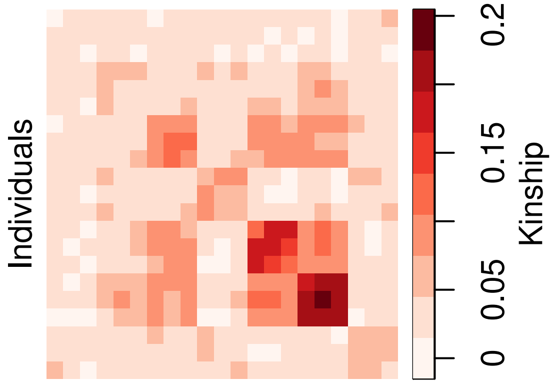
```
# estimate FST before rescaling (this value will be wrong, too high!)
fst(PhiAfr)
```

```
## [1] 0.2474861
```

Removing populations changes the MRCA population T, drastically in this case (the reason the minimum kinship is so large and the within-AFR F_{ST} above is wrong). To ensure the minimum kinship is zero, instead of re-estimate the kinship matrix from the subset genotypes, it suffices to rescale the given kinship matrix with `rescalePopkin`!

```
# rescale PhiAfr
# since subpops is missing, minimum Phi value is set to zero
# (no averaging between subpopulations)
PhiAfr <- rescalePopkin(PhiAfr)

# kinship matrix plot
par(oma=c(0,1.5,0,3))
par(mar=c(0,0,0,0)+0.2) # zero margins for no labels
plotPopkin(inbrDiag(PhiAfr))
```



```
# FST is now correct, relative to the MRCA of AFR individuals
fst(PhiAfr)
```

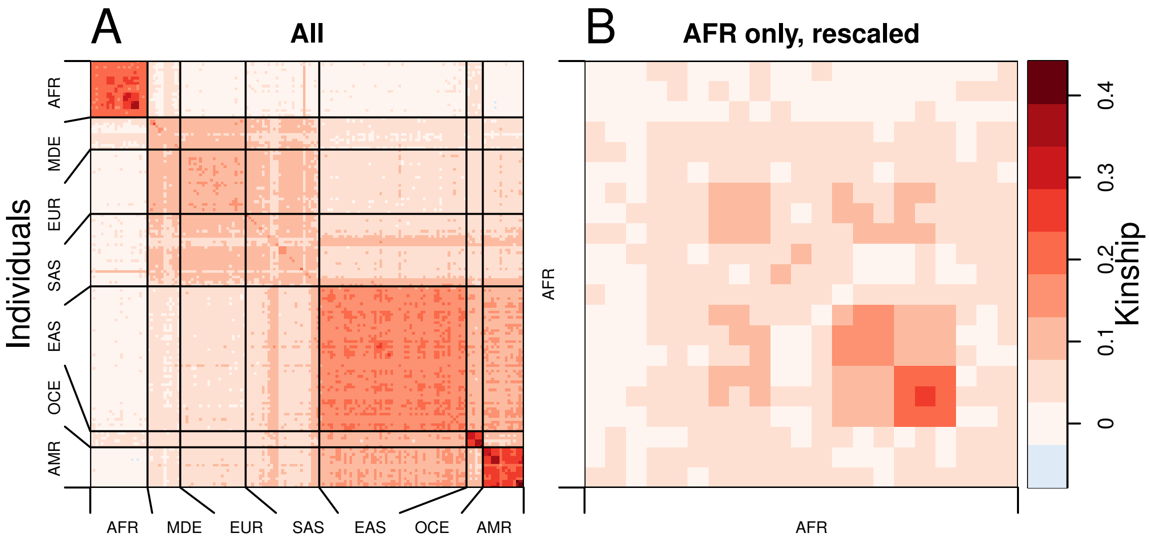
```
## [1] 0.08879701
```

There is clear substructure within Sub-Saharan Africa, but this sample data does not have more detailed labels that could help us interpret further.

2.7 Plot multiple kinship matrices together

As a final example, we plot the global `Phi` and the rescaled AFR subset `PhiAfr` side-by-side, illustrating how more than one kinship matrix can be plotted with a shared legend.

```
par(oma=c(0,1.5,0,3))
# increase top margin for titles
par(mar=c(2,2,2,0)+0.2)
# dummy labels to have lines in second panel
subpopsAfr <- subpops[indexesAfr]
plotPopkin(
    list(inbrDiag(Phi), inbrDiag(PhiAfr)), # list of matrices
    titles=c('All', 'AFR only, rescaled'), # title of each panel
    labs=list(subpops, subpopsAfr), # pass per-panel labels using a list
    labsEven=TRUE, # scalar options are shared across panels
    labsLine=1,
    labsCex=0.5
    )
```



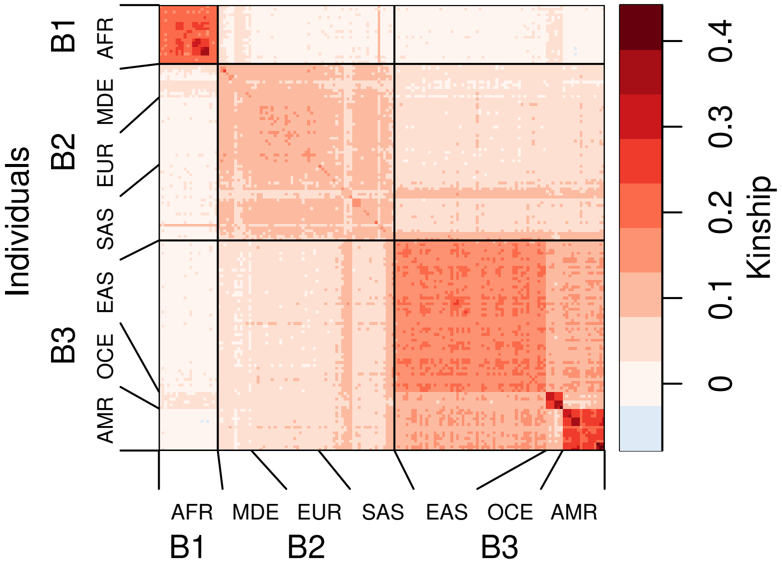
2.8 Plot kinship matrices with multiple levels of labels

The `plotPopkin` function has advanced options for plotting more than one level of labels. For this example, we will highlight the three "blocks" that represent the first two splits in the human migration out of Africa:

```
# create second level of labels
# first copy first-level labels
blocks <- subpops
# first block is AFR
blocks[blocks=='AFR'] <- 'B1'
# second block is West Eurasians, broadly defined
blocks[blocks=='MDE'] <- 'B2'
blocks[blocks=='EUR'] <- 'B2'
blocks[blocks=='SAS'] <- 'B2'
# third block is East Eurasians, broadly defined
```

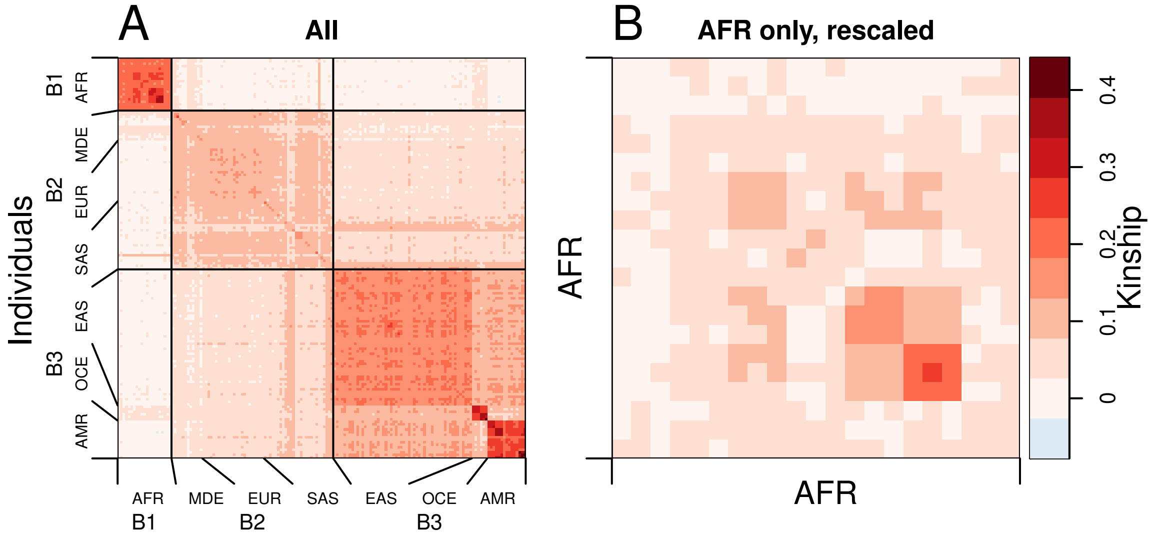
```
blocks[blocks=='EAS'] <- 'B3'
blocks[blocks=='OCE'] <- 'B3'
blocks[blocks=='AMR'] <- 'B3'

par(oma=c(0,1.5,0,3))
# increase margins again
par(mar=c(3,3,0,0)+0.2)
# plotting with different options per level is more complicated...
plotPopkin(
    inbrDiag(Phi),
    labs=cbind(subpops,blocks),    # ... labs is now a matrix with levels on columns
    labsEven=c(TRUE, FALSE),       # ... even spacing for first level only
    labsLine=c(1,2),               # ... put second level further out
    labsCex=c(0.7, 1),             # ... don't shrink second level
    labsSkipLines=c(TRUE, FALSE), # ... draw lines inside heatmap for second level only
    ylabAdj=0.65                   # push up outer margin ylab "Individuals"
    )
```



The final example adds a second panel to what we have above, showing how options must be passed when labels differ per panel and there are multiple levels:

```
par(oma=c(0,1.5,0,3))
par(mar=c(3,3,2,0)+0.2)
plotPopkin(
    list(inbrDiag(Phi), inbrDiag(PhiAfr)),
    titles=c('All', 'AFR only, rescaled'),
    labs=list(cbind(subpops,blocks), subpopsAfr), # list of matrices
    labsEven=c(TRUE, FALSE), # non-list: values are reused for both panels
    labsLine=c(1,2),
    # make label bigger in second panel (custom per-panel values)
    labsCex=list(c(0.5, 0.7), 1), # list of vectors
    # add lines for first level of second panel (custom per-panel values)
    labsSkipLines=list(c(TRUE, FALSE), FALSE) # list of vectors
    )
```



References

Ochoa, Alejandro, and John D. Storey. 2016a. "F_{ST} And Kinship for Arbitrary Population Structures I: Generalized Definitions." *bioRxiv* doi:10.1101/083915. Cold Spring Harbor Labs Journals. doi:10.1101/083915.

———. 2016b. "F_{ST} And Kinship for Arbitrary Population Structures II: Method of Moments Estimators." *bioRxiv* doi:10.1101/083923. Cold Spring Harbor Labs Journals. doi:10.1101/083923.