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

Alejandro Ochoa and John D. Storey

2017-05-08

Introduction

The `popkin` ("population kinship") package lets users estimate the kinship matrix of all individuals in a dataset from their genotypes, which yields individual kinship and inbreeding coefficients as well as F_{ST} for arbitrarily structured populations. We recently introduced a new framework for generalizing and estimating F_{ST} and kinship under arbitrary population structures (Ochoa and Storey 2016a; Ochoa and Storey 2016b). Here we briefly summarize the notation and intuition behind the key parameters.

Kinship and inbreeding coefficients

Let T be the reference ancestral population, which sets the level of relatedness treated as zero as explained shortly. 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. Both f_j^T, φ_{jk}^T are probabilities of "identity by descent" (IBD) that are more carefully defined elsewhere (Ochoa and Storey 2016a; Ochoa and Storey 2016b). In a structured population we expect most if not all $f_j^T, \varphi_{jk}^T > 0$. 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 kinship definition also applies to $j = k$ (the "self-kinship"), which counterintuitively 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 must be so for statistical modeling applications; however, for visualization purposes 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 replacing the diagonal of Φ^T with f_j^T values results in more aesthetically-pleasing figures.

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 introduced a generalized F_{ST} definition that applies to arbitrary population structures—dropping the need for subpopulations—and partitions "total" inbreeding into "local" inbreeding (due to having unusually closely related parents) and "structural" inbreeding (due to the population structure) (Ochoa and Storey 2016a). Future work will focus on extracting structural from total kinship, but in the meantime only the total kinship matrix Φ^T is estimated by `popkin`. 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 `popkin` package assumes all individuals are locally outbred in estimating F_{ST}.

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).

Sample usage

Loading data

The `popkin` package does not come with its own data parser, but accepts inputs in three forms.

1. The simplest case assumes your genotypes are in an R matrix `X` that contains values only in `c(0L,1L,2L,NA)` (recall in R `0` is internally as double but `0L` is an integer; `popkin` handles both but providing only integers will be most efficient). This is a standard encoding for biallelic SNPs that 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` gives the same kinship and F_{ST} estimates if you input `X` or `2L-X`. By default `popkin` expects loci along rows and individuals along columns (an $m \times n$ matrix); you may input a transposed `X` matrix to the `popkin` function along with `lociOnCols=TRUE`.

2. If your data is in BED format, we recommend using the `BEDMatrix` package to load this data. For example, if you have the three files `myData.bed`, `myData.bim`, `myData.fam`, you can load the BEDMatrix object using:

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

   Note this `X` is not exactly like the regular matrix `X` in the first case because the data is not loaded onto memory until you access it, but `popkin` accepts either as input and just works. `popkin` is designed to work with `BEDMatrix` to use memory as efficiently as possible. The plink2 software can be used to convert many common formats into BED.

3. The last option is to provide a function `X(m)` that when called loads the next m SNPs of the data, returning an $m \times n$ matrix in the same format as `X` in the first case above. This option is provided for non-BED data that is too large to load into an R matrix. Since this requires the most work from users (who must write their own functions `X(m)` for their custom formats), we recommend converting the data into BED format and using `BEDMatrix` to load it.

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
```

Therefore, 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
```

Now we're ready to analyze this data with `popkin`!

Estimate the kinship matrix and F_{ST} using subpopulations

Estimating a kinship matrix requires the genotype matrix `X` and subpopulation levels used only to estimate the minimum level of kinship. Given the previous data, obtaining the estimate is simple:

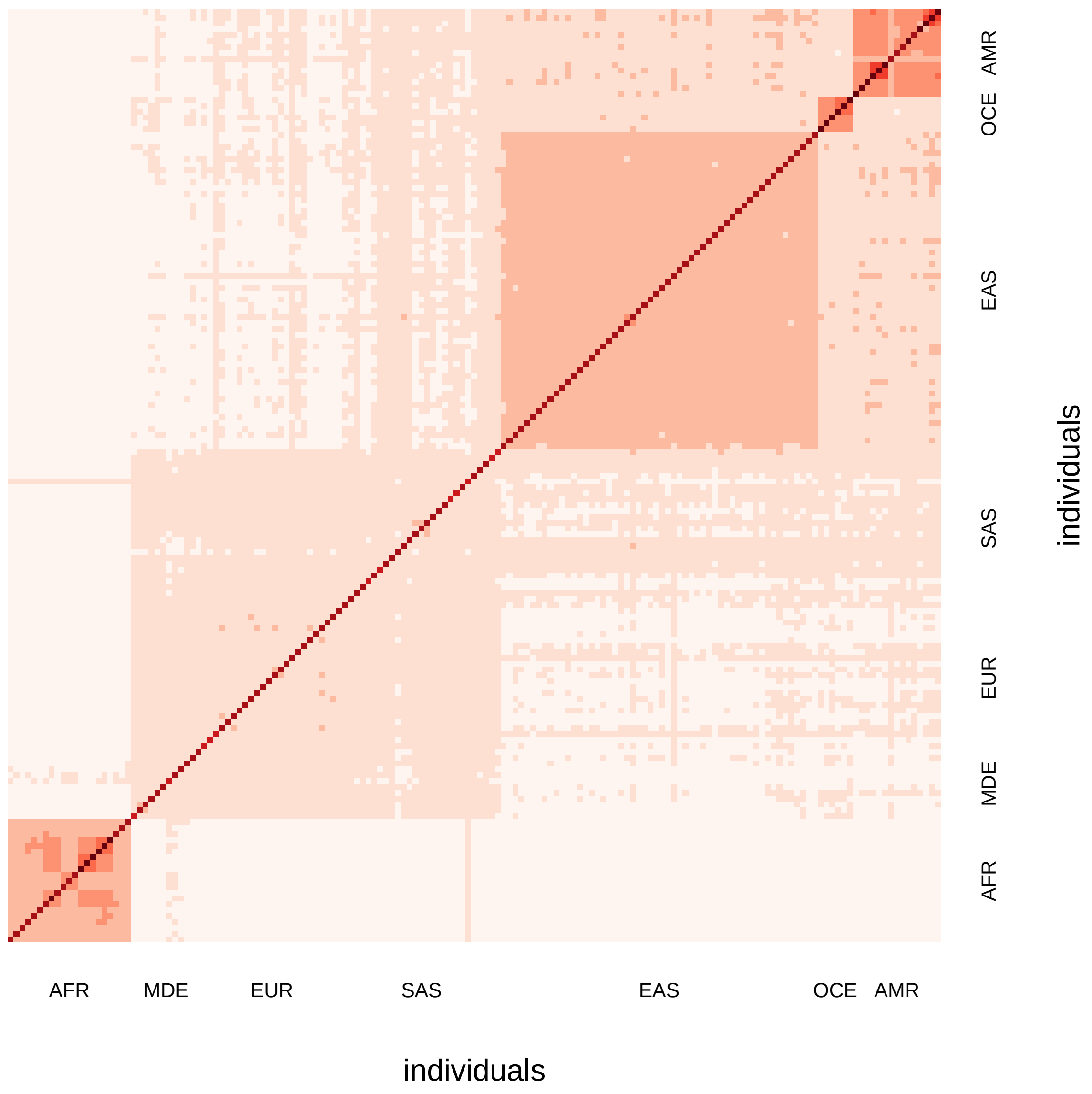
```
Phi <- popkin(X, subpops)
```

Let's visualize the data. First let's setup a simple heatmap function:

```
# quick and dirty heatmap of the kinship matrix
library(RColorBrewer)
myHeatmap <- function(Phi, subpops=NULL) {
    # place subpop labels in middle of their ranges
    subpopLabs <- NULL
    if (!is.null(subpops)) {
        # mean index per subpop
        subpopLabs <- aggregate(1:length(subpops), list(subpop=subpops), mean)
    }
    colHM <- brewer.pal(9, 'Reds') # heatmap colors
    par(xaxt='n', yaxt='n') # heatmap doesn't omit axes correctly, force here
    # plot as image, without reordering, dendrograms, etc
    heatmap(Phi, Rowv=NA, Colv=NA, symm=TRUE, col=colHM,
        xlab='individuals', ylab='individuals',
        add.expr={
            # add population labels
        if (!is.null(subpopLabs)) {
            mtext(subpopLabs$subpop, 1, at=subpopLabs$x, line=1, cex=0.7)
            mtext(subpopLabs$subpop, 4, at=subpopLabs$x, line=1, cex=0.7)
        }
        })
    par(xaxt='s', yaxt='s') # reset to other plots aren't affected

}
```

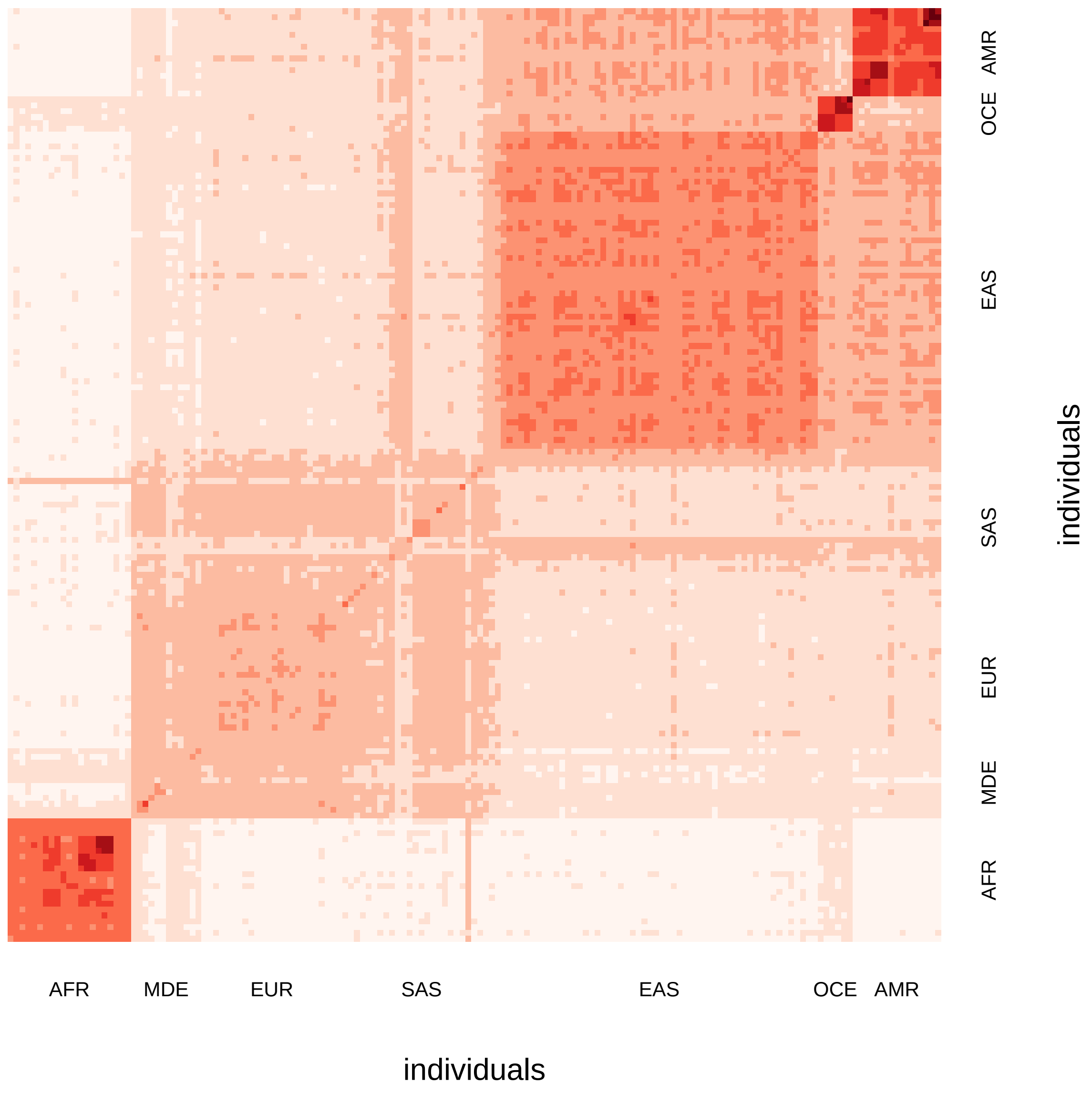
Now let's visualize the raw kinship matrix estimate:

```
myHeatmap(Phi, subpops)
```



In the previous plot it's clear that the self-kinship estimates (the diagonal) are much greater than the rest of the kinship values (the minimum along the diagonal is 0.5). It makes more sense to plot the inbreeding coefficients along the diagonal, so `popkin` includes the `inbrDiag` function that rescales the diagonal appropriately:

```r
myHeatmap(inbrDiag(Phi), subpops)
```



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 somewhat noisier than they would be in full data (datasets routinely have over 300 thousand SNPs).

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

F_{ST} is then estimated from the 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:

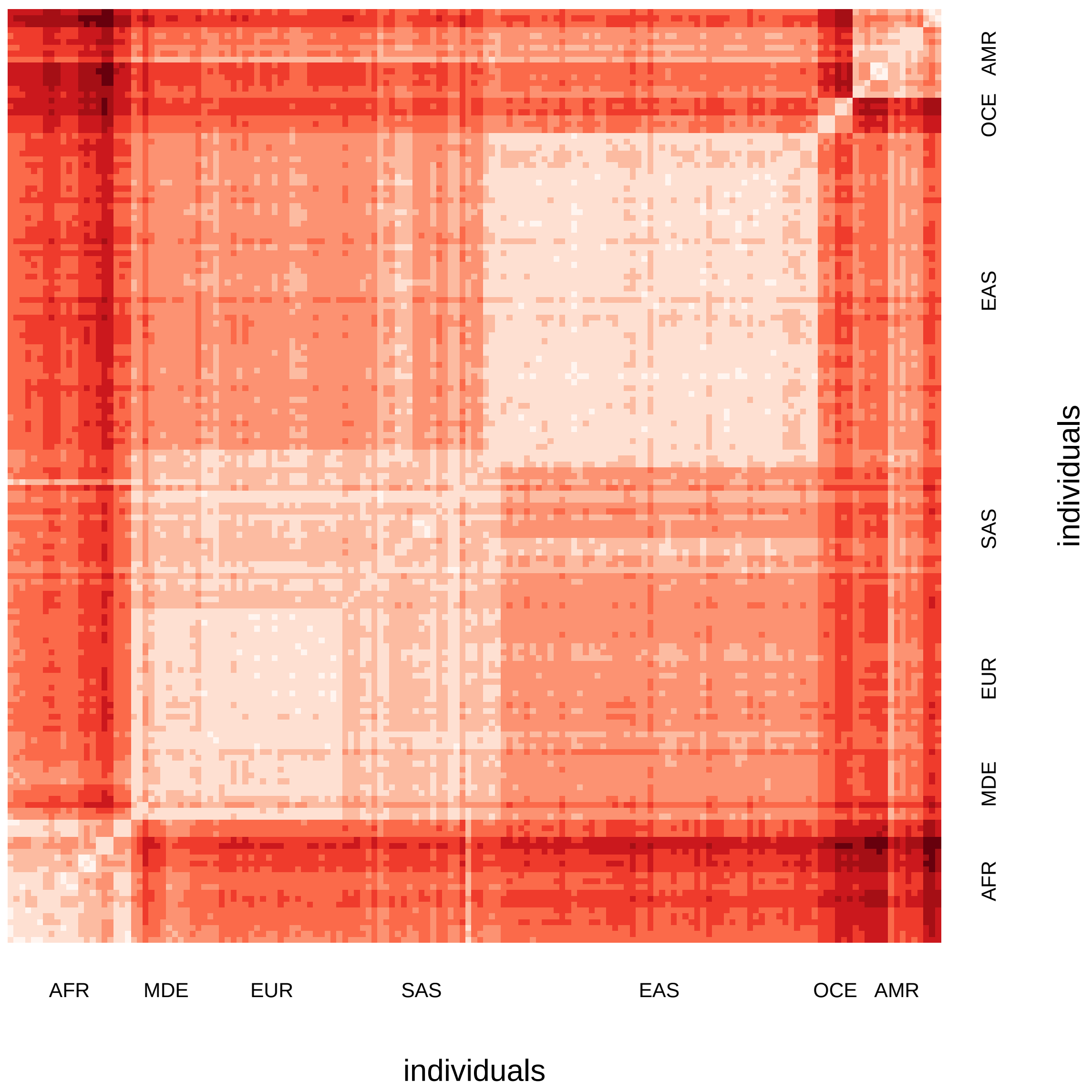
```r
# get weights
w <- weightsSubpops(subpops)
# compute FST!
# Note: don't use the output to inbrDiag(Phi) or FST will be wrong!
fst(Phi, w)
```

```
## [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, this HGDP sample is older and likely suffered from SNP ascertainment issues, which might explain the difference.

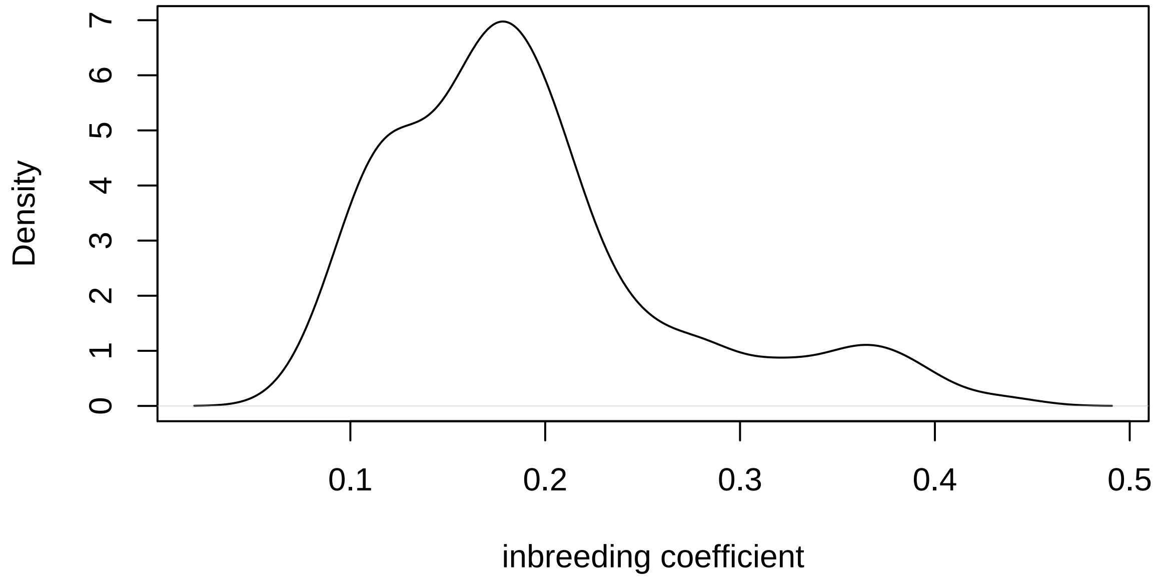
We calculate individual-level pairwise F_{ST} estimates from the previous kinship estimates using `pwfst`. Note that the pairwise F_{ST} is a distance between pairs of individuals: approximately zero for individuals in the same population, and greater than zero and increasing for more distant pairs of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

```
pwF <- pwfst(Phi) # compute pairwise FST matrix from kinship matrix
myHeatmap(pwF, subpops) # NOTE no need for inbrDiag() here
```



Lastly, we can extract the vector of inbreeding coefficients from the kinship matrix using `inbr`:

```r
inbrs <- inbr(Phi) # vector of inbreeding coefficients
par(mar=c(4, 4, 0, 0) + 0.1) # reduce margins
plot(density(inbrs), xlab='inbreeding coefficient', main='') # see their distribution
```



Rescale kinship matrix in a subset of the data

Suppose now you're interested in one subpopulation, say AFR. Removing other populations changes the MRCA population T, so the appropriate action is to re-estimate the kinship matrix in this subset only, although this could be slow if you have a very large dataset. Fortunately, you can take the kinship matrix you already had and manipulate it to get the same answer!

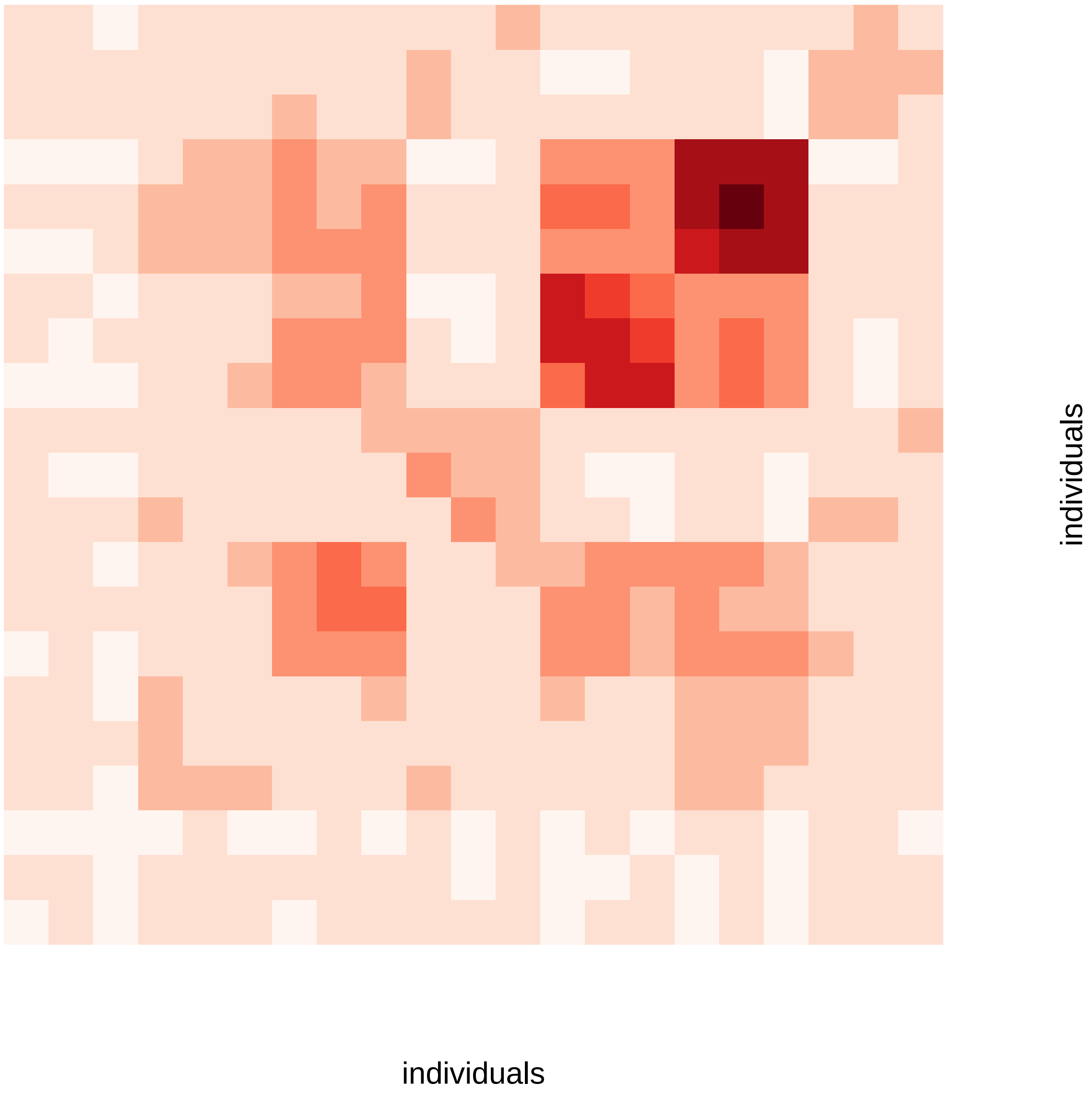
```r
# filter to only keep individuals within AFR
indexesAfr <- subpops == 'AFR'
PhiAfr <- Phi[indexesAfr,indexesAfr]

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

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

```r
# now rescale
# since subpops is missing, minimum Phi value is set to zero
# (no averaging between subpopulations)
PhiAfr <- rescalePopkin(PhiAfr)
# FST is now correct, relative to the MRCA of AFR individuals
fst(PhiAfr)
```

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

```r
# kinship matrix visualization
myHeatmap(inbrDiag(PhiAfr))
```



In this sample dataset there are no labels for individuals or sub-subpopulations, so unfortunately we can't interpret this matrix further, beyond that there is clear substructure within Sub-Saharan Africa.

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.