iMKT using PopFly or PopHuman data

Brief intro about lots of data right now. Then, PopFly and PopHuman as great databases.

Therefore, iMKT package includes some functions which allow an easy retrieval and analysis of population genetics information stored in these genome browsers. Specifically, the functions permit the download of population genetics parameters computed for every gene annotation in several populations for both model species *Drosophila melanogaster* and *Homo sapiens*.

The examples in this vignette focus only on PopFly data and analysis, but the same process could be performed using human data and funcions (**loadPopHumandata()**, **PopHumanAnalysis()**, and **PopHumanData**), following the same steps described here. Recombination rate values of human data are retrieved from Bheller et al. and correspond to the sex average estimates.

Loading the row data

The first step is to load the information into your working enviroment. This would allow a manual examination of the data before starting any alayses. However, this step can be skipped, and the main data object is loaded into the workspace the first time that the analysis function is called. To download PopFly data use the **loadPopFly()** function, without any argument.

Keep in mind that you are downloading the complete gene information of several populations (16 *D. melanogaster* in this case and 26 in the case of *H. sapiens*), and this procces could take a while.

Once the function finishes,the **PopFlyData** object is loaded into the workspace. This object can be manually examined or used when performing the main iMKT analyses.

```
library(iMKT)
loadPopFly()
#> Loading PopFly data into your workspace.
#> This process may take some seconds to complete, please be patient.
ls() ## new object created
#> [1] "PopFlyData"
names(PopFlyData)
#>  [1] "Pop"         "Name"        "Start"       "End"         "Chr"
#>  [6] "p0"          "pi"          "d0"          "di"          "m0"
#> [11] "mi"          "alpha"       "fisher_pval" "DoS"         "KaKs"
#> [16] "DAF0f"       "DAF4f"       "cM_Mb"
```

Each row of the **PopFlyData** dataframe contains information regarding one gene annotation in one single population. In total, 16 populations and 13745 genes are included. Metrics for each gene contain information about segrating, divergent and analyzed sites and the Derived Allele Frequency (DAF) for neutral (4fold) and putatively selected (0fold) sites; together with some neutrality tests statistics (Standard MKT, Direction of Selection, Ka/Ks) and gene-associated recombination rate estimates (from Comeron et al.).

Once the data is loaded into the workspace, analyses can be performed using the function **PopFlyAnalysis()**.

Performing PopFly Analyses

The **PopFlyAnalisys()** function allows performing any MK test using a subset of PopFly data defined by custom genes and populations lists. It uses the previously loaded dataframe (PopFlyData). In addition to the **genes** and **populations** lists, the function also has the following parameters:

- **recomb**: group genes according to recombination values (must specify number of bins). Options are: TRUE/FALSE. Recombination values (cM/Mb) from Comeron et al. 2012.
- **bins**: number of recombination bins to compute (mandatory if recomb = TRUE)
- **test**: which test to perform. Options include: standard (default), DGRP, FWW, asymptotic, iMK
- **xlow**: lower limit for asymptotic alpha fit (default=0)
- **xhigh**: higher limit for asymptotic alpha fit (default=1)

Custom genes must be listed using FlyBase IDs (FBgn...), and the available populations from PopFly are: AM, AUS, CHB, EA, EF, EG, ENA, EQA, FR, RAL, SA, SD, SP, USI, USW, ZI.

Hence, using the parameters recomb and bins, the function allows deciding whether to analyze genes groupped by recombination bins or not.

Example 1

In this first example, the analysis is focused in two genes and 2 populations, without considering gene's recombination context, and using DGRP methodology.

The function groups polymorphism and divergence values of the custom genes, creating a new "concatenated gene" for each population of interest and performs the test defined. Then, it returns a list of lists with the default test output (DGRP in this case) for each population (RAL and ZI).

Explain output of DGRP function.

```
PopFlyAnalysis(genes=c("FBgn0000055","FBgn0003016"), pops=c("RAL","ZI"), recomb=F, test="DGRP")
#> [1] "Population = RAL"
#> Warning in check_input(daf, divergence, 0, 1): Input daf file contains P0 values = 0.
#> This can bias the function fitting and the estimation of alpha.
#> [1] "Population = ZI"
#> $`Population =  RAL`
#> $`Population =  RAL`$Results
#>                alpha.symbol Fishers exact test P-value
#> Cutoff =  0     -1.19780220                 0.03712979
#> Cutoff =  0.05  -0.04395604                 1.00000000
#> Cutoff =  0.2   -0.37362637                 0.44760661
#>
#> $`Population =  RAL`$`Divergence metrics`
#> $`Population =  RAL`$`Divergence metrics`$`Global metrics`
#>            Ka         Ks      omega
#> 1 0.003767024 0.05226481 0.072075733
#>
#> $`Population =  RAL`$`Divergence metrics`$`Estimates by cutoff`
#>                omegaA.symbol omegaD.symbol
#> Cutoff =  0     -0.086332464    0.15840819
#> Cutoff =  0.05  -0.003168164    0.07524389
#> Cutoff =  0.2   -0.026929392    0.09900512
#>
```

```
#>
#> $`Population =  RAL`$`MKT tables`
#> $`Population =  RAL`$`MKT tables`$`Number of segregating sites by DAF category - Cutoff =  0`
#>
#>
#> Table: cutoff
#>
#>                  DAF.below.cutoff   DAF.above.cutoff
#> ---------------  -----------------  -----------------
#> Neutral class                    0                 63
#> Selected class                   0                 40
#>
#> $`Population =  RAL`$`MKT tables`$`Number of segregating sites by DAF category - Cutoff =  0.05`
#>
#>
#> Table: cutoff
#>
#>                  DAF.below.cutoff   DAF.above.cutoff
#> ---------------  -----------------  -----------------
#> Neutral class                   20                 43
#> Selected class                  34                  6
#>
#> $`Population =  RAL`$`MKT tables`$`Number of segregating sites by DAF category - Cutoff =  0.2`
#>
#>
#> Table: cutoff
#>
#>                  DAF.below.cutoff   DAF.above.cutoff
#> ---------------  -----------------  -----------------
#> Neutral class                   30                 33
#> Selected class                  34                  6
#>
#> $`Population =  RAL`$`MKT tables`$`MKT standard table`
#>
#>
#>                  Polymorphism   Divergence
#> ---------------  -------------  -----------
#> Neutral class               63           45
#> Selected class              40           13
#>
#>
#> $`Population =  RAL`$Fractions
#>           0       0.05        0.2
#> d 0.8415918 0.84039746 0.84178039
#> f 0.1584082 0.07524389 0.09900512
#> b 0.0000000 0.08435865 0.05921449
#>
#>
#> $`Population =  ZI`
#> $`Population =  ZI`$Results
#>                alpha.symbol Fishers exact test P-value
#> Cutoff =  0      -0.5279503                  0.2545795
#> Cutoff =  0.05   -0.1180124                  0.8623250
#> Cutoff =  0.2    -0.2670807                  0.6114433
```

```
#>
#> $`Population =  ZI`$`Divergence metrics`
#> $`Population =  ZI`$`Divergence metrics`$`Global metrics`
#>            Ka         Ks      omega
#> 1 0.003765933 0.04524362 0.08323677
#>
#> $`Population =  ZI`$`Divergence metrics`$`Estimates by cutoff`
#>                omegaA.symbol omegaD.symbol
#> Cutoff =  0     -0.043944879    0.12718165
#> Cutoff =  0.05  -0.009822973    0.09305974
#> Cutoff =  0.2   -0.022230939    0.10546771
#>
#>
#> $`Population =  ZI`$`MKT tables`
#> $`Population =  ZI`$`MKT tables`$`Number of segregating sites by DAF category - Cutoff =  0`
#>
#>
#> Table: cutoff
#>
#>                   DAF.below.cutoff   DAF.above.cutoff
#> ----------------  -----------------  -----------------
#> Neutral class                     0                161
#> Selected class                    0                 82
#>
#> $`Population =  ZI`$`MKT tables`$`Number of segregating sites by DAF category - Cutoff =  0.05`
#>
#>
#> Table: cutoff
#>
#>                   DAF.below.cutoff   DAF.above.cutoff
#> ----------------  -----------------  -----------------
#> Neutral class                   108                 53
#> Selected class                   77                  5
#>
#> $`Population =  ZI`$`MKT tables`$`Number of segregating sites by DAF category - Cutoff =  0.2`
#>
#>
#> Table: cutoff
#>
#>                   DAF.below.cutoff   DAF.above.cutoff
#> ----------------  -----------------  -----------------
#> Neutral class                   125                 36
#> Selected class                   78                  4
#>
#> $`Population =  ZI`$`MKT tables`$`MKT standard table`
#>
#>
#>                   Polymorphism   Divergence
#> ----------------  -------------  -----------
#> Neutral class              161           39
#> Selected class              82           13
#>
#>
#> $`Population =  ZI`$Fractions
```

```
#>           0       0.05        0.2
#> d 0.8728183 0.87282798 0.87229814
#> f 0.1271817 0.09305974 0.10546771
#> b 0.0000000 0.03411227 0.02223415
```

Example 2

In this second example, genes from RAL population are groupped in 2 recombination bins, using recombination values from Comeron et al (ref). The test used is iMK, with xlow and xhigh values set to 0 and 0.9, respectively.

In this case, the function creates two different "concatenated" genes containing the same number of genes each (6 in this case), grouping again polymorphism and divergence values. These aggrupations are made according to the gene associated recombination rate estimates. The function returns a list of lists with the default test output (iMKT in this case) for each population (RAL) and recombination bin (1 and 2).

Explain iMK function output.

```
geneList <- c("FBgn0053196","FBgn0086906","FBgn0261836","FBgn0031617","FBgn0260965",
              "FBgn0028899","FBgn0052580","FBgn0036181","FBgn0263077","FBgn0013733",
              "FBgn0031857","FBgn0037836")

PopFlyAnalysis(genes=geneList , pops=c("RAL"), recomb=T, bins=2, test="iMK", xlow=0, xhigh=0.9)
#> [1] "Population = RAL"
#> [1] "Recombination bin = 1"
#> [1] "Recombination bin = 2"
#> $`Population =  RAL`
#> $`Population =  RAL`$`Recombination bin =  1`
#> $`Population =  RAL`$`Recombination bin =  1`$`Asymptotic MK table`
#>         model      a       b       c alpha_asymptotic  CI_low CI_high
#> 1 exponential 0.4626 -0.8511 14.8205           0.4626 -0.9147 27.7653
#>   alpha_original
#> 1         0.2489
#>
#> $`Population =  RAL`$`Recombination bin =  1`$`Fractions of sites`
#>   Type  Fraction
#> 1    d 0.6515607
#> 2    f 0.3484393
#> 3    b 0.0000000
#>
#> $`Population =  RAL`$`Recombination bin =  1`$`Recombination bin Summary`
#>   numGenes minRecomb medianRecomb meanRecomb maxRecomb
#> 1        6 0.6230327     1.216535    1.20733  1.759126
#>
#>
#> $`Population =  RAL`$`Recombination bin =  2`
#> $`Population =  RAL`$`Recombination bin =  2`$`Asymptotic MK table`
#>         model      a       b      c alpha_asymptotic  CI_low CI_high
#> 1 exponential 0.6632 -0.5395 7.3407           0.6628 -0.3102  0.7312
#>   alpha_original
#> 1          0.426
#>
```

```
#> $`Population =  RAL`$`Recombination bin =  2`$`Fractions of sites`
#>   Type Fraction
#> 1    d 0.727223
#> 2    f 0.272777
#> 3    b 0.000000
#>
#> $`Population =  RAL`$`Recombination bin =  2`$`Recombination bin Summary`
#>   numGenes minRecomb medianRecomb meanRecomb maxRecomb
#> 1        6  2.277748     3.518252   3.398377  4.104627
```