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*.

This vignette is divided in two main sections:

- Loading the row data
- Performing iMKT analyses

Although the examples presented here focus only on PopFly data and analyses, the same process could be performed using human data and functions (**PopHumanData**, **loadPopHuman()**, **PopHumanAnalysis()**), following the same steps described in this document. Recombination rate values included in human data were retrieved from Bhérer et al. 2017 Nature Commun. 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 alaysis. However, to speed analyses, this step can be skipped and the main data object is loaded into the workspace the first time that the corresponding analysis function is called. To download PopFly data use the **loadPopFly()** function, without any argument.

Keep in mind that you are downloading full-genome information of several populations (13745 gene annotations for 16 populations in *D.melanogaster* and 18145 gene annotations for 26 populationsin *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 iMKT analyses.

```
## Load the iMKT library
library(iMKT)

## Load PopFlyData
loadPopFly()
#> Loading PopFly data into your workspace.
#> This process may take several seconds to complete, please be patient.

## Check data object
ls()
#> [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. Metrics for each gene contain information about the number of segrating, divergent and analyzed

positions, and the Derived Allele Frequency distribution (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 (retrieved from Comeron et al. 2012 Plos Genetics).

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

Performing iMKT 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 (TRUE/FALSE)
- **bins**: number of recombination bins to compute (mandatory if recomb=TRUE)
- **test**: which test to perform. Options include: standardMK (default), DGRP, FWW, asymptoticMK, iMK
- **xlow**: lower limit for asymptotic alpha fit (default=0)
- **xhigh**: higher limit for asymptotic alpha fit (default=1)
- **plot**: report plot (optional, default=FALSE)

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.

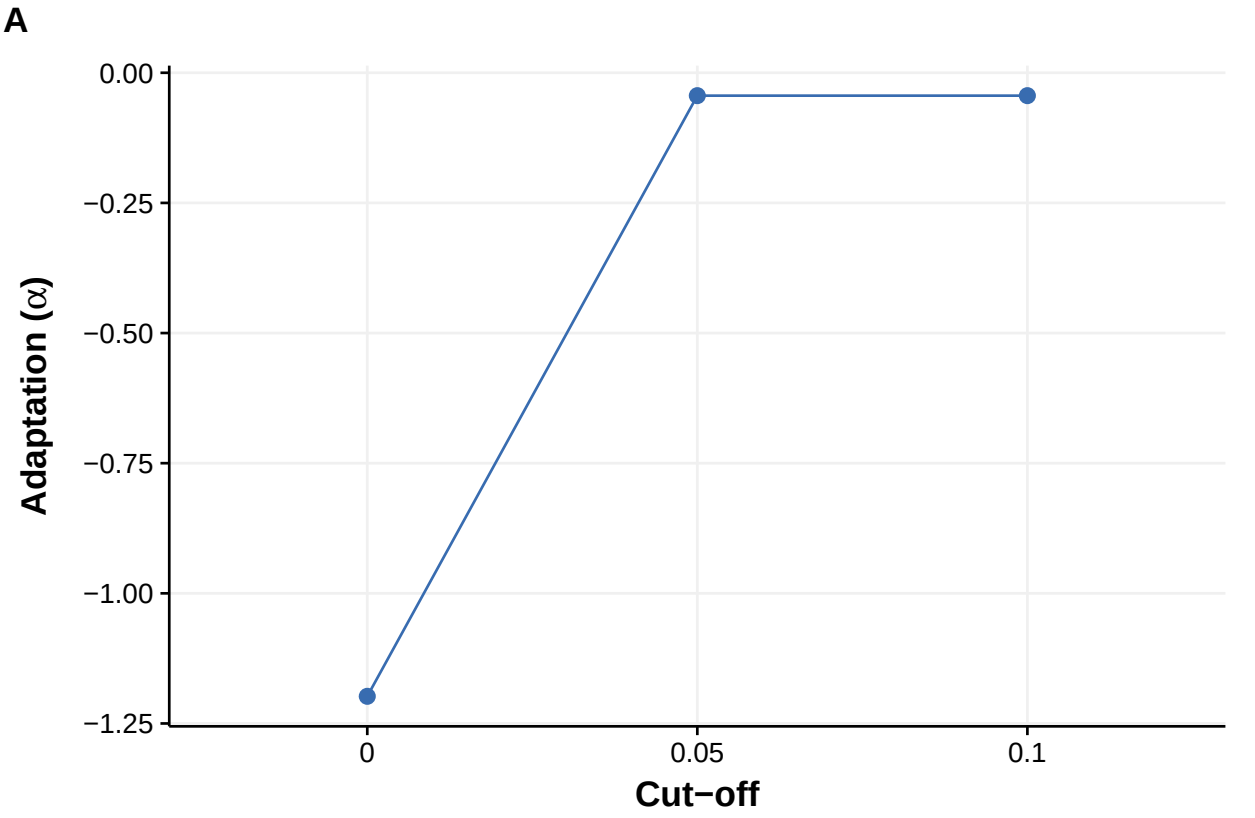
Example 1. DGRP analysis without recombination

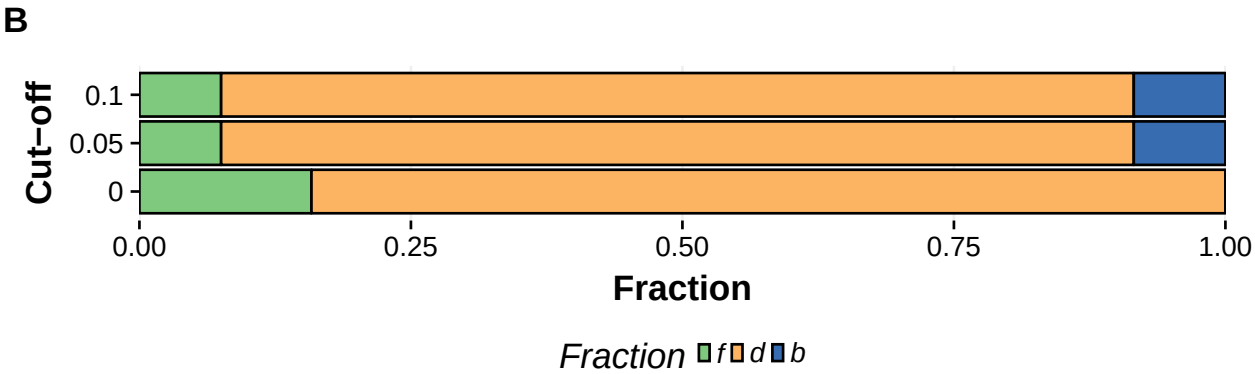
In this first example, the analysis is focused in two genes and 2 populations, without considering gene's recombination context, and using DGRP methodology.

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.

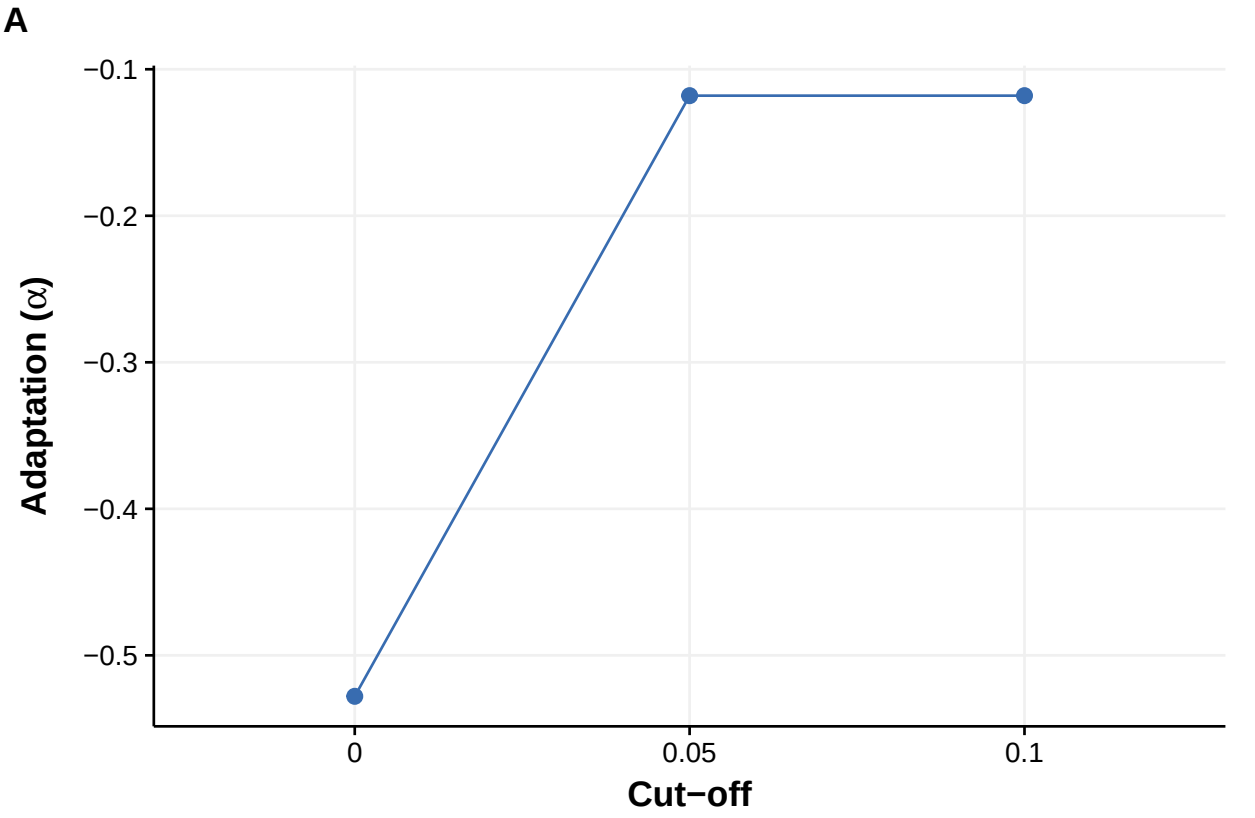
```
PopFlyAnalysis(genes=c("FBgn0000055","FBgn0003016"), pops=c("RAL","ZI"), recomb=F, test="DGRP", plot=TRU
#> [1] "Population = RAL"
#> [1] "Population = ZI"
#> $`Population =  RAL`
#> $`Population =  RAL`$Results
#>                alpha.symbol Fishers exact test P-value
#> Cutoff =  0     -1.19780220                 0.037712979
#> Cutoff =  0.05  -0.04395604                 1.00000000
#> Cutoff =  0.1   -0.04395604                 1.00000000
#>
#> $`Population =  RAL`$Graph
```

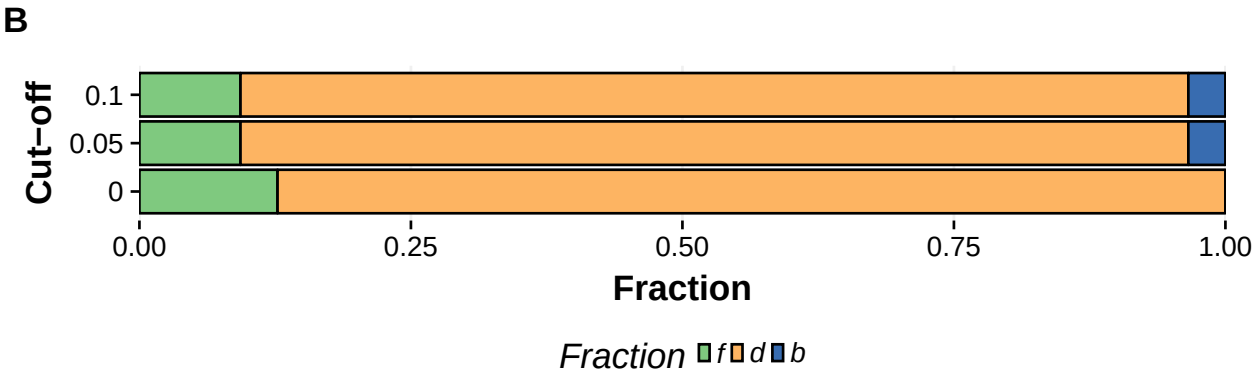





```
#>
#> $`Population =  RAL`$`Divergence metrics`
#> $`Population =  RAL`$`Divergence metrics`$`Global metrics`
#>            Ka         Ks      omega
#> 1 0.003767024 0.05226481 0.07207573
#>
#> $`Population =  RAL`$`Divergence metrics`$`Estimates by cutoff`
#>                omegaA.symbol omegaD.symbol
#> Cutoff =  0     -0.086332464    0.15840819
#> Cutoff =  0.05  -0.003168164    0.07524389
#> Cutoff =  0.1   -0.003168164    0.07524389
#>
#>
#> $`Population =  RAL`$`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 =  0.1`
#>
#>
#> Table: cutoff
#>
#>                    DAF.below.cutoff   DAF.above.cutoff
#> ----------------  ------------------  ------------------
#> Neutral class                     20                  43
#> 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        0.1
#> d 0.8415918 0.84039746 0.84039746
#> f 0.1584082 0.07524389 0.07524389
#> b 0.0000000 0.08435865 0.08435865
#>
#>
#> $`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 =  0.1    -0.1180124                  0.8623250
#>
#> $`Population =  ZI`$Graph
```






```
#>
#> $`Population =  ZI`$`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 =  0.1   -0.009822973    0.09305974
#>
#>
#> $`Population =  ZI`$`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 =  0.1`
#>
#>
#> Table: cutoff
#>
#>                   DAF.below.cutoff   DAF.above.cutoff
#> ----------------  -----------------  -----------------
#> Neutral class                   108                 53
#> Selected class                   77                  5
#>
#> $`Population =  ZI`$`MKT tables`$`MKT standard table`
#>
#>
#>                   Polymorphism   Divergence
#> ----------------  -------------  -----------
#> Neutral class              161           39
#> Selected class              82           13
#>
#>
#> $`Population =  ZI`$Fractions
#>           0       0.05        0.1
#> d 0.8728183 0.87282798 0.87282798
#> f 0.1271817 0.09305974 0.09305974
#> b 0.0000000 0.03411227 0.03411227
```

Example 2. iMK analysis using recombination bins

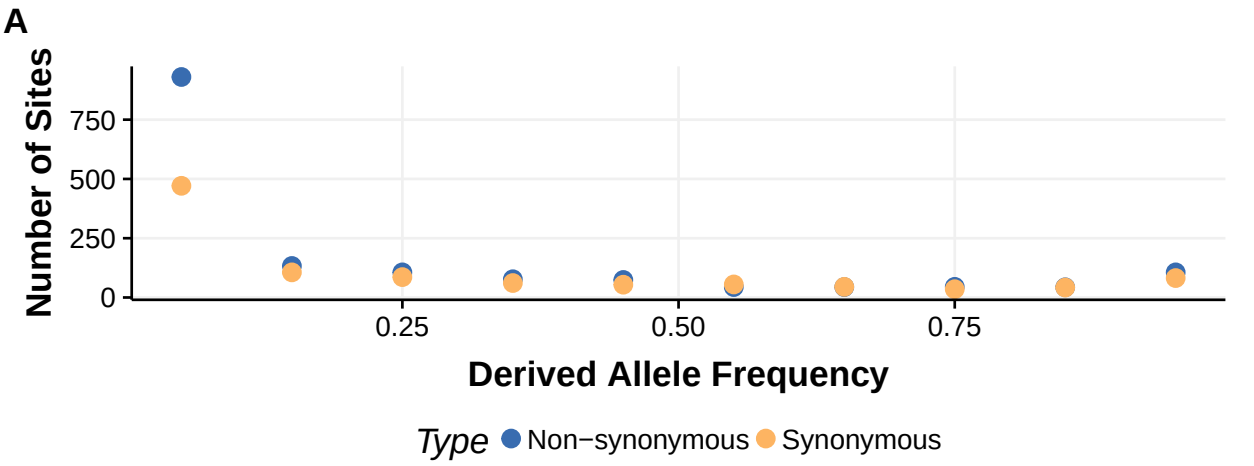
In this second example, genes from RAL population are groupped in 2 recombination bins, using recombination values from Comeron et al. 2012 Plos Genetics. The test used is iMK, with xlow and xhigh values set to 0 and 0.9, respectively.

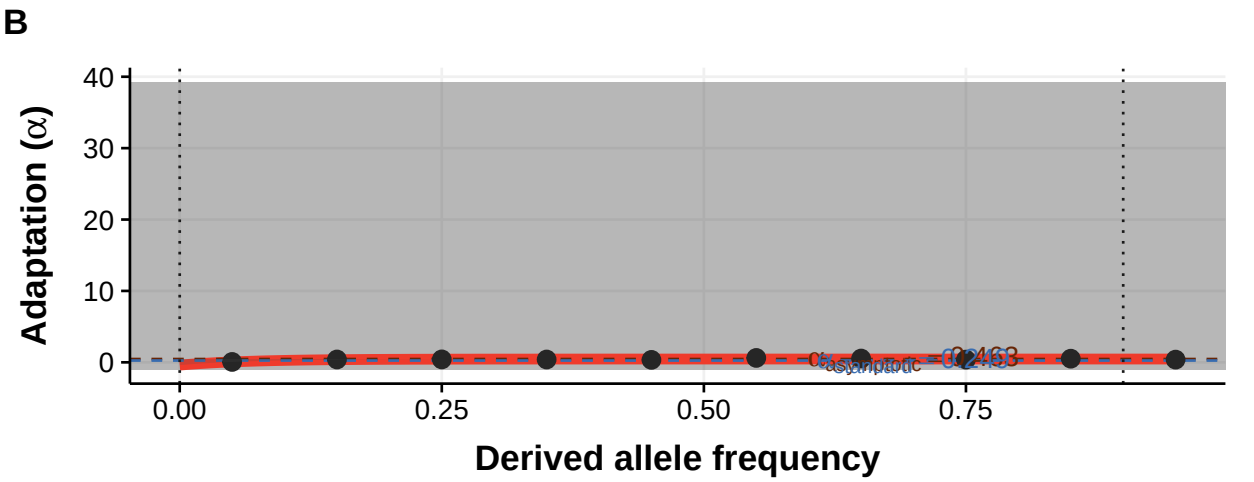
In this case, the function creates two different "concatenated" genes containing the same number of genes each (6 in this case), grouping again polymorphism and divergence values. These aggrupations are made

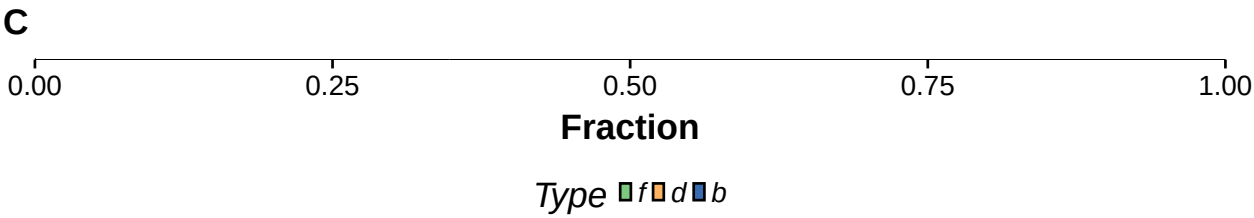
according to the gene associated recombination rate estimates. The function returns a list of lists with the default test output (iMKT in this case) for each population (RAL) and recombination bin (1 and 2).

Explain iMK function output.

```
geneList <- c("FBgn0053196","FBgn0086906","FBgn0261836","FBgn0031617","FBgn0260965",
              "FBgn0028899","FBgn0052580","FBgn0036181","FBgn0263077","FBgn0013733",
              "FBgn0031857","FBgn0037836")
PopFlyAnalysis(genes=geneList , pops="RAL", recomb=T, bins=2, test="iMK", xlow=0, xhigh=0.9, plot=TRUE)
#> [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 -0.9728 39.2591
#>   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`$Graphs
```

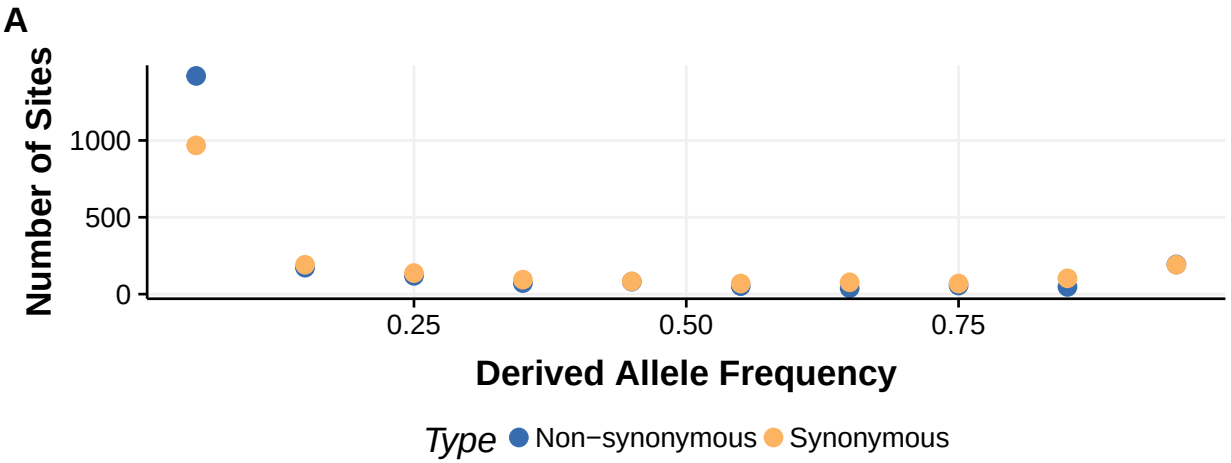


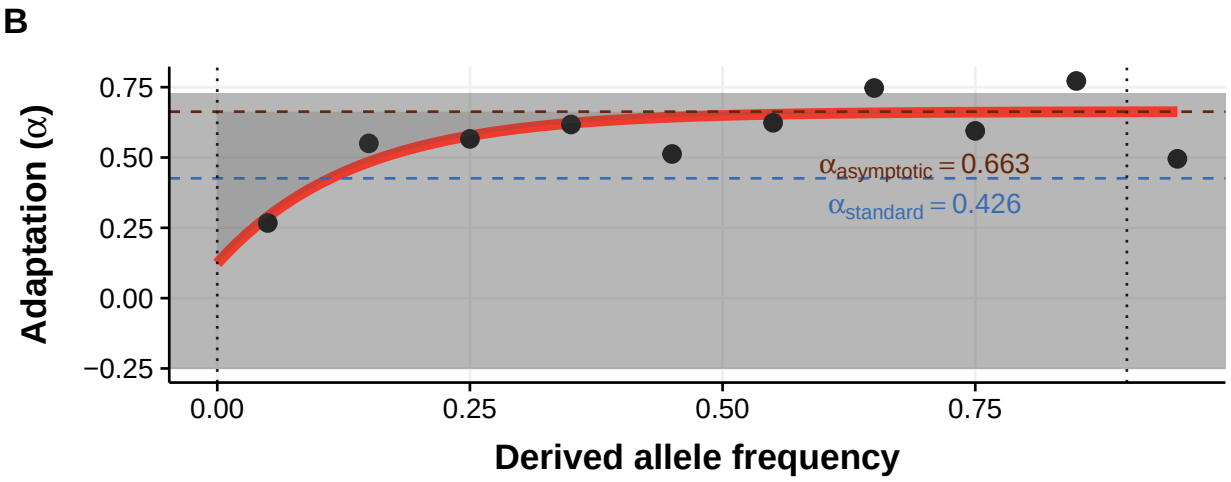


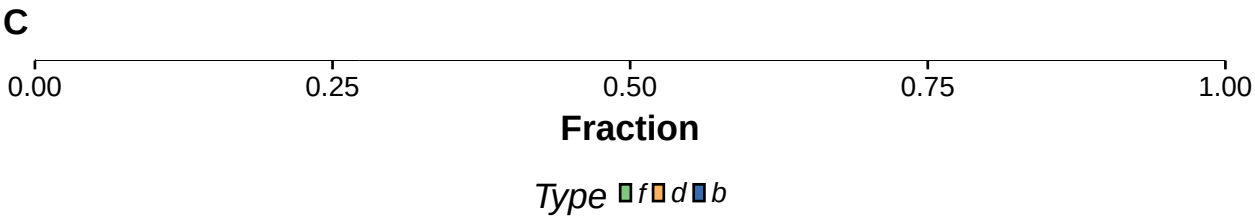



```
#>
#> $`Population =  RAL`$`Recombination bin =  1`$`Recombination bin Summary`
#>   numGenes minRecomb medianRecomb meanRecomb maxRecomb
#> 1        6 0.6230327     1.216535    1.20733  1.759126
#>
#>
#> $`Population =  RAL`$`Recombination bin =  2`
#> $`Population =  RAL`$`Recombination bin =  2`$`Asymptotic MK table`
#>         model      a       b      c alpha_asymptotic CI_low CI_high
#> 1 exponential 0.6632 -0.5395 7.3407           0.6628 -0.249  0.7298
#>   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`$Graphs
```








```
#>
#> $`Population =  RAL`$`Recombination bin =  2`$`Recombination bin Summary`
#>   numGenes minRecomb medianRecomb meanRecomb maxRecomb
#> 1        6  2.277748     3.518252   3.398377  4.104627
rm(geneList)
```