PopGenome Session

Bastian Pfeifer

June 20, 2013

1 Reading data

Loading the PopGenome package

```
> library(PopGenome)
```

Reading three alignments in FASTA-format stored in the folder "FASTA".

```
> GENOME.class <- readData("FASTA")
```

```
|            :            |            :            | 100 %
|====================================================
```

`GENOME.class` is an object of class `GENOME`.

```
> GENOME.class
```

```
-----
Modules:
-----
                Calculation             Description        Get.the.Result
1                  readData            Reading data          get.sum.data
2          neutrality.stats        Neutrality tests        get.neutrality
3             linkage.stats  Linkage disequilibrium           get.linkage
4              recomb.stats           Recombination            get.recomb
5                F_ST.stats         Fixation index get.F_ST,get.diversity
6                       MKT  McDonald-Kreitman test               get.MKT
7              detail.stats       Mixed statistics            get.detail
8                        MS   Coalescent simulation                     @
9            --------------             -----------         -------------
10          set.populations Defines the populations
11 sliding.window.transform          Sliding window
12            splitting.data         Splits the data
13               show.slots       ?provided slots?
14               get.status  Status of calculations
```

The class `GENOME` contains all observed data and statistic values which are presentable in a multi-locus-scale. Use the function `show.slots(GENOME.class)` to get an overview or check out the manual. To access those values we use the @-operator.

How many sites were analyzed in each alignment ?

```
> GENOME.class@n.sites

4CL1tl.fas  C4Htl.fas  CADtl.fas
      2979       2620       2930

> GENOME.class@region.names

[1] "4CL1tl.fas" "C4Htl.fas"  "CADtl.fas"
```

To get some summary information from the alignments use the `get.sum.data` function. This function extracts the values from the class `GENOME` and puts them into a matrix. You can also look at those values seperately with the @-operator (`GENOME.class@n.biallelic.sites`).

```
> get.sum.data(GENOME.class)

           n.sites n.biallelic.sites n.gaps n.unknowns n.valid.sites
4CL1tl.fas    2979               176    617          0          2362
C4Htl.fas     2620                84   1454          0          1161
CADtl.fas     2930               197    740          0          2189
           n.polyallelic.sites trans.transv.ratio
4CL1tl.fas                   0           1.120482
C4Htl.fas                    5           1.470588
CADtl.fas                    1           0.970000
```

The Slot `region.data` contains some detail (site specific) informations, which are not presentable in a multi-locus-scale. `region.data` is another class and its slots are accessable with the @ operator.

```
> GENOME.class@region.data

-----
SLOTS:
-----
                      Slots                          Description
1               populations   Samples of each population (rows)
2              populations2  Samples of each population (names)
3                  outgroup                  Samples of outgroup
4               transitions          Biallelic site transitions
5          biallelic.matrix                     Biallelic matrix
6              n.singletons                 Number of singletons
7           biallelic.sites           Position of biallelic sites
8                 reference                        SNP reference
9              n.nucleotides  Number of nucleotides per sequence
10  biallelic.compositions Nucleotides per sequence (biallelic)
11               synonymous           Synonymous biallelic sites
12 biallelic.substitutions             Biallelic substitutions
13         polyallelic.sites             Sites with >2 nucleotides
14           sites.with.gaps              Sites with gap positions
15       sites.with.unknowns          Sites with unknown positions
16             minor.alleles                        Minor alleles
17                    codons    Codons of biallelic substitutions
```

```
18               IntronSNPS                SNPs in intron region
19                 UTRSNPS                    SNPs in UTR region
20               CodingSNPS                SNPs in coding region
21                 ExonSNPS                  SNPs in exon region
22                 GeneSNPS                  SNPs in gene region

---------------
These are the Slots (class region.data)
```

The first 10 biallelic positions of the first alignment:

```
> GENOME.class@region.data@biallelic.sites[[1]][1:10]
```

```
 [1]  12  13  31  44  59 101 121 154 165 202
```

Which of those biallelic sites are transitions ?

```
> GENOME.class@region.data@transitions[[1]][1:10]
```

```
 [1]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE FALSE FALSE
```

2 Reading data with GFF/GTF information

The GFF folder contains GFF-files for each alignment. The GFF-files have the same names as the corresponding alignments

```
> GENOME.class <- readData("FASTA",gffpath="GFF")
```

```
|             :             |             :             | 100 %
|=====================================================
```

Which of the first 10 SNPs of the second `[[2]]` alignment are part of an synonymous mutation ?

```
> GENOME.class@region.data@synonymous[[2]][1:10]
```

```
 [1] TRUE TRUE TRUE TRUE TRUE TRUE   NA   NA   NA   NA
```

`NA` values indicate that the sites are not in a coding region

```
> GENOME.class@region.data@CodingSNPS[[2]][1:10]
```

```
 [1] 1413 1428 1446 1455 1482 1488 1744 1756 1798 1802
```

2.1 Splitting the data in subsites

If the number of individuals are identical, you can use the `splitting.data` function to split the data in subsites. In this example we are splitting into `coding` (CDS) regions. The returned value is again an object of class `GENOME`.

```
> GENOME.class.split <- splitting.data(GENOME.class, subsites="coding")
```

```
|             :             |             :             | 100 %
|=====================================================|             :             |
|=====================================================
```

Each region contains now the SNP-informations of each coding region defined in the gff-files. In case of whole-genome SNP data this mechanism can be very useful. (manual:`readSNP,readVCF`)

```
> GENOME.class.split@n.sites
```

```
 [1] 1056  413  103   96  785  132  595   92  112  226  438  220
```

```
> GENOME.class.split <- neutrality.stats(GENOME.class.split)
```

Apply the neutrality module to all synonymous SNPs in the coding regions.

```
> GENOME.class.split <- neutrality.stats(GENOME.class.split, subsites="syn")
```

```
> GENOME.class.split@Tajima.D
```

3 Define populations

Define two poulations as a list.

```
> GENOME.class <- set.populations(GENOME.class,list(
+                  c("CON","KAS-1","RUB-1","PER-1","RI-0","MR-0","TUL-0"),
+                  c("MH-0","YO-0","ITA-0","CVI-0","COL-2","LA-0","NC-1")
+                 ))
```

```
|            :            |            :            | 100 %
|===================================================
```

4 Statistics

4.1 Neutrality statistics

```
> GENOME.class  <- neutrality.stats(GENOME.class)
```

```
|            :            |            :            | 100 %
|===================================================
```

Getting the result from the object of class `GENOME`.

```
> get.neutrality(GENOME.class)
```

```
      neutrality stats
pop 1 Numeric,27
pop 2 Numeric,27
```

Lets look at the first population `[[1]]`.

```
> get.neutrality(GENOME.class)[[1]]
```

```
             Tajima.D n.segregating.sites Rozas.R_2    Fu.Li.F    Fu.Li.D
4CL1tl.fas -1.1791799                  16        NA -0.9247377 -1.1331823
C4Htl.fas   0.6987394                  17        NA  0.6742517  0.4167836
CADtl.fas   0.5503743                  14        NA  0.4458431  0.1590690
           Fu.F_S Fay.Wu.H Zeng.E Strobeck.S
4CL1tl.fas     NA      NaN    NaN         NA
C4Htl.fas      NA      NaN    NaN         NA
CADtl.fas      NA      NaN    NaN         NA
```

The `NA` values indicates that the statistics could not be calculated. This can have several reasons.

- the statistic needs an outgroup
- the statistic was not switched on
- there are no SNPs in the entire region

In each module you can switch on/off statistics and define an outgroup. (check the manual !). PopGenome also provides a population specific view of each statistic value.

```
> GENOME.class@Tajima.D
```

```
                pop 1      pop 2
4CL1tl.fas -1.1791799 -0.0702101
C4Htl.fas   0.6987394  1.1819777
CADtl.fas   0.5503743  0.2682897
```

If there there was a GFF/GTF file specified, you can also analyse subsites like SNPs `exon,coding,utr or intron regions`.

```
> GENOME.class  <- neutrality.stats(GENOME.class, subsites="coding")
```

```
|            :            |            :            | 100 %
|====================================================
```

```
> GENOME.class@Tajima.D
```

```
               pop 1     pop 2
4CL1tl.fas -1.023785 0.2626617
C4Htl.fas   1.013372 1.9121846
CADtl.fas   1.981520 1.5191652
```

Or each subsite-region seperately by splitting the data as described in section 2.1.

```
> GENOME.class.split <- splitting.data(GENOME.class, subsites="coding")
```

```
|            :            |            :            | 100 %
|====================================================|            :            |
|====================================================
```

```
> GENOME.class.split  <- neutrality.stats(GENOME.class.split)
```

```
|            :            |            :            | 100 %
|===================================================

> GENOME.class.split@Tajima.D

                 pop 1      pop 2
240 - 1295  -0.2749244 -0.3186974
1890 - 2302 -1.0062306  0.7546749
2679 - 2781 -1.0062306  0.5590170
2884 - 2979 -1.0062306        NaN
3465 - 4249         NA         NA
4337 - 4468        NaN        NaN
4696 - 5290 -1.6097384  2.1259529
6181 - 6272        NaN        NaN
6412 - 6523        NaN        NaN
7320 - 7545  0.2390231  1.8112198
7643 - 8080 -0.3018700  1.1684289
8176 - 8395        NaN        NaN
```

The PopGenome framework provides several modules to calculate statistics. All methods will work as the neutrality.stats() function described above. Please read the user manual.

4.2 The slot region.stats

The slot `region.stats` includes some site-specific statistics or values that can not be shown in a multi-locus-scale.

```
> GENOME.class@region.stats

-----
SLOTS:
-----
                   Slots                    Description  Module
1   nucleotide.diversity            Nucleotide diversity     FST
2    haplotype.diversity             Haplotype diversity     FST
3       haplotype.counts          Haplotype distribution     FST
4     minor.allele.freqs        Minor allele frequencies  Detail
5 linkage.disequilibrium         Linkage disequilibrium Linkage
6    biallelic.structure Shared and fixed polymorphisms  Detail

---------------
These are the Slots (class region.data)

> GENOME.class  <- F_ST.stats(GENOME.class)

|            :            |            :            | 100 %
|===================================================

> GENOME.class@region.stats@nucleotide.diversity
```

```
[[1]]
         pop 1    pop 2
pop 1 5.142857       NA
pop 2 6.163265 5.238095

[[2]]
         pop 1 pop 2
pop 1 7.809524    NA
pop 2 8.816327     4

[[3]]
         pop 1    pop 2
pop 1 6.285714       NA
pop 2 5.836735 4.285714
```

5 Sliding Window Analysis

The `sliding.window.transform()` transforms an object of class `GENOME` in another object of class `GENOME`. This mechanism enables the user to apply all methods existing in the PopGenome framework.

PopGenome tries to concatenate the data if the parameter `whole.data=TRUE`. This mechanism is useful to handle chunks in the PopGenome framework. Otherwise the regions are scanned seperately.

`type=1`: Scanning the SNPs
`type=2`: Scanning the wohle data

5.1 Scanning whole data

```
> GENOME.class.slide  <- sliding.window.transform(GENOME.class,width=50,
+                        jump=50,type=1,whole.data=TRUE)

|            :            |            :            | 100 %
|===================================================|            :            |
|===================================================

> GENOME.class.slide@region.names

[1] "1 - 50 :"    "51 - 100 :"  "101 - 150 :" "151 - 200 :" "201 - 250 :"
[6] "251 - 300 :" "301 - 350 :" "351 - 400 :" "401 - 450 :"

> GENOME.class.slide  <- linkage.stats(GENOME.class.slide)

|            :            |            :            | 100 %
|===================================================

> get.linkage(GENOME.class.slide)[[1]]

               Wall.B    Wall.Q   Rozas.ZA    Rozas.ZZ  Kelly.Z_nS
1 - 50 :    0.6666667 0.7500000 0.66666667  0.29166667 0.375000000
```

```
51 - 100 :       NaN       NaN 0.00000000  0.00000000 0.000000000
101 - 150 : 0.0000000 0.0000000 0.01851852 -0.05266204 0.071180556
151 - 200 : 0.6250000 0.6666667 0.37847222  0.10206619 0.276406036
201 - 250 : 0.5833333 0.6923077 5.40972222  1.05354208 4.356180145
251 - 300 : 0.0000000 0.0000000 0.01388889 -0.17860000 0.192488889
301 - 350 : 0.0000000 0.0000000 0.01388889  0.00462963 0.009259259
351 - 400 : 0.4000000 0.5000000 3.95688889  2.19704321 1.759845679
401 - 450 : 0.5000000 0.6000000 1.81250000  1.31916667 0.493333333
```

5.2 Scanning the regions seperately

```
> GENOME.class.slide  <- sliding.window.transform(GENOME.class,width=50,
+                         jump=50,type=1,whole.data=FALSE)

|             :             |             :             | 100 %
|=====================================================

> GENOME.class.slide@region.names

[1] "1:4CL1tl.fas" "2:4CL1tl.fas" "3:4CL1tl.fas" "4:C4Htl.fas"  "5:CADtl.fas"
[6] "6:CADtl.fas"  "7:CADtl.fas"

> GENOME.class.slide  <- linkage.stats(GENOME.class.slide)

|             :             |             :             | 100 %
|=====================================================

> get.linkage(GENOME.class.slide)[[1]]

                Wall.B Wall.Q   Rozas.ZA    Rozas.ZZ Kelly.Z_nS
1:4CL1tl.fas 0.6666667   0.75 0.66666667  0.29166667 0.37500000
2:4CL1tl.fas       NaN    NaN 0.00000000  0.00000000 0.00000000
3:4CL1tl.fas 0.0000000   0.00 0.01851852 -0.05266204 0.07118056
4:C4Htl.fas  0.6666667   0.80 0.54086420 -0.09315802 0.63402222
5:CADtl.fas  0.0000000   0.00 2.09259259 -0.04456019 2.13715278
6:CADtl.fas  0.0000000   0.00 0.01388889 -1.37808642 1.39197531
7:CADtl.fas  0.5000000   0.60 0.88888889 -0.27527778 1.16416667
```

6 Coalescent simulation

PopGenome supports the Coalescent simulation program `MS` from Hudson as well as the `MSMS` simulation tool from Greg Ewing. The obsereved statistics are tested against the simulated values. You have to specify the θ value and the module you want to apply to the simulated data. An new object of class `cs.stats` will be created. The main input is an object of class `GENOME`

```
> MS.class <- MS(GENOME.class,thetaID="Tajima",neutrality=TRUE)

|             :             |             :             | 100 %
|=====================================================

> MS.class
```

```
-----
SLOTS:
-----
       Slots                                       Description
1  prob.less           Prob. that sim.val <=  obs.val P(sim <= obs)
2 prob.equal              Prob. that sim.val = obs.val P(sim = obs)
3 valid.iter           number of valid iter. for each test and loci
4    obs.val                               obs.values for each test
5     n.loci                              number of loci considered
6     n.iter                     number of iterations for each loci
7    average      average values of each statistic  (across all loci)
8   variance      variance values of each statistic (across all loci)
9      locus list of loc.stats objects, (detail stats for each locus)

---------------
```

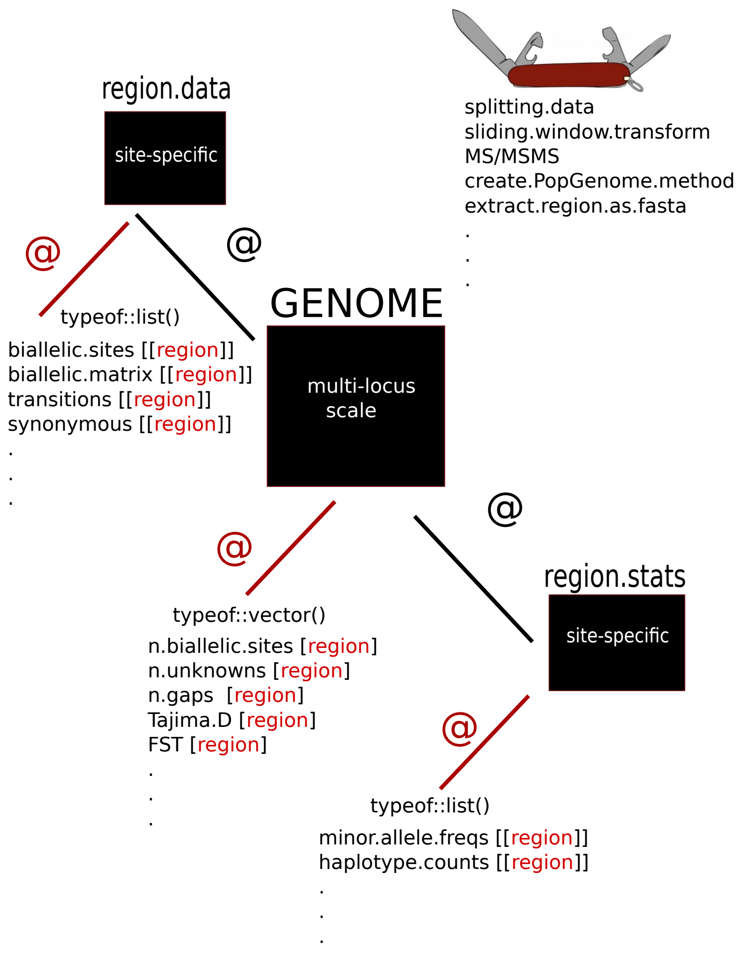
Lets look at the data of the first region

```
> MS.class@locus[[1]]

   Length     Class      Mode
        1 loc.stats        S4
-----
SLOTS:
-----
            Slots                                        Description
1           n.sam                 number of samples for each iteration
2          n.iter                                  number of iteration
3           theta                                   mutation parameter
4         obs.val                vector with observed values for each test
5       positions                      position of each polymorphic site
6           trees              if printtree=1, gene tree in Newick format
7           seeds                 random numbers used to generate samples
8     halplotypes                           haplotypes in each iteration
9           stats                 variety of test stats compiled a matrix
10  loc.prob.less Prob. that simulated val. <= to observed val. P(Sim <= Obs)
11 loc.prob.equal   Prob. that simulated val = to  observed val. P(Sim = Obs)
12 loc.valid.iter                 number of valid iteration for each test
13      quantiles                              13 quantiles for each test

---------------
[1] "These are the Slots"
```

7 PopGenome classes



splitting.data
sliding.window.transform
MS/MSMS
create.PopGenome.method
extract.region.as.fasta
.
.
.
region.data
site-specific
@
@
typeof::list()
biallelic.sites [[region]]
biallelic.matrix [[region]]
transitions [[region]]
synonymous [[region]]
.
.
.
GENOME
multi-locus scale
@
@
typeof::vector()
n.biallelic.sites [region]
n.unknowns [region]
n.gaps [region]
Tajima.D [region]
FST [region]
.
.
.
region.stats
site-specific
@
typeof::list()
minor.allele.freqs [[region]]
haplotype.counts [[region]]
.
.
.