AutoPoly Introduction

David Field

2017-07-06

1 What is AutoPoly?

AutoPoly is an R package that performs population assignment in autotetraploids and autohexaploids using genetic marker information. The analysis uses allele frequency information from reference genotypes sampled from candidate (source) populations, and then assigns a set of genotyped (or phenotyped) individuals of unknown origin (i.e. offspring) to their most likely:

- joint maternal and paternal population, when maternal genotype is unknown (e.g. seed dispersal), or
- paternal population, given the known maternal genotype of each offspring (e.g. pollen dispersal).

This is achieved using co-dominant markers (microsatellites or single nucleotide polymorphisms) with a frequency-based approach that incorporates polysomic inheritance and accounts for double reduction. The package also allows both genotype (allele copy number known) and phenotype allelic marker data (allele copy number unknown). Note that although either SSR or SNPs can be used, the package is not optimised for large number of SNP loci that may be required to acheive sufficient performance in population assignment.

This short guide documents the important functions, the required data input format and a typical workflow for **AutoPoly**, outlining the steps required to import data, run analyses and examine the output using example data sets.

In summary, a typical workflow for **AutoPoly** would involve to following steps:

1. Preparation of genotype data file in Comma Separated Value (CSV). The required format is documented in section **Data format**
2. Starting R and installing and loading **AutoPoly** package and setting the working directory to the location of the data file.
3. Importing the data and processing with **AutoPoly**.
4. Running the required functions to run the population assignment with **AutoPoly**
5. Export the output to CSV and examine the assignment likelihoods

2 Important functions

- `inputData()`: imports the data from CSV into R and performs various checks and tidying of the data set.
- `DRRsetup()`: makes a dataframe containing randomly generated Double Reduction Rates (DRR) at individual loci.
- `makeGameteSegTable()`: makes a dataframe with the segregation ratios given the DRR defined with `DRRsetup()`.
- `alleleFreqGeno()` and `alleleFreqPheno()`: calculates the allele frequencies in each population for genotype or phenotype data, respectively.
- `gameteFreq()`: calculate expected gamete frequencies at random mating equilibrium for all populations in a dataset
- `simAssignmentMums()`and `simAssignment()`: performs simulations of population assignment for data sets with or without mother information, respectively.
- `assignmentPheno()` and `assignmentGeno()`: performs population assignment on phenotype data and genotype data, respectively.
- `summaryAssignment()`: final posthoc and summary statistics for population assignment
- `convert.Structure()`: convert the data set to the Structure format.

3 Data file format

AutoPoly requires data on the multilocus genotypes or phenotypes of reference individuals sampled from a set of candidate populations, and of another set of individuals to be assigned.

An example of the first few lines from a data file, for adult samples from a population (GA):

Table 1: Microsattelite data set provided with **AutoPoly** for the autohexaploid plant, Eremophila glabra. Consists of phenotypes (allele dosage unknown) for six loci. Here we show an example for one locus for a sample of adults (9 shown) and offspring (5 shown from mother - GA1)

ID	pop	mother	locus223a	locus223b	locus223c	locus223d	locus223e	locus223f
GA1	GA		214	220	*	*	*	*
GA10	GA		212	*	*	*	*	*
GA11	GA		214	216	218	*	*	*
GA12	GA		218	220	*	*	*	*
GE5	GE		218	220	*	*	*	*
GE6	GE		218	220	*	*	*	*
GE7	GE		216	220	*	*	*	*
GE8	GE		218	220	*	*	*	*
GE9	GE		216	*	*	*	*	*
GA1-1	GA	GA1	212	214	216	*	*	*
GA1-10	GA	GA1	214	218	220	222	*	*
GA1-2	GA	GA1	212	214	218	220	*	*
GA1-3	GA	GA1	214	216	220	*	*	*
GA1-4	GA	GA1	214	218	220	*	*	*

Some important rules to the data file format:

- single row per individual.
- individuals have strictly a maximum of four or six alleles for any given analysis (i.e. no mixture of tetraploid and hexaploid individuals).
- the first column, contains the individual's unique identification label (ID)
- the second column, the population location (pop)
- the third column the maternal parent (mother) for offspring in the data (if known). If unknown, leave a 'u' in this column.
- the remaining columns consist of the genotype or phenotype where the number of columns per locus must equal the ploidy level (e.g. tetraploid; locus1a, locus1b, locus1c, locus1d, locus2a, locus2b, locus2c, locus2d, . . .).
- alleles can be listed as alphanumeric characters or numbers (excluding -9, NA, *, which are reserved for missing data).
- adults (including mothers) contain no information in the mother column.
- when mother information is being used, the labels identifying the mother of offspring must match adult individual labels (ID)
- only complete maternal information or no maternal information allowed in any given data set for all offspring (i.e. all offspring are either listed with a known mother or all are left unknown).
- for genotype markers, all cells must be filled with alleles.
- for phenotype markers, only the unique alleles per locus present, with the remaining cells left with the missing data symbol ('*')

4 Example workflow

If you are running **AutoPoly** for the first time you will have to install and load into R:

```
install.packages("AutoPoly")
library(AutoPoly)
```

AutoPoly is dependent on the package 'gtools' which should be installed automatically. If not, you will have to install and load using:

Alternatively, if you have downloaded a copy of the program separately as **AutoPoly_0.1.0.tar.gz** you could also install from the terminal window with:

```
R CMD INSTALL AutoPoly_0.1.0.tar.gz
```

and then start R from the terminal and load the package:

```
library(AutoPoly)
```

AutoPoly includes two example microsatellite (SSR) data sets in the required format: (i) six SSR loci from an autohexaploid plant (Eremophila), and (ii) 24 SSR loci from a simulated autotetraploid data set (x4_Fst0_13_Sim).

4.1. Eremophila data (mother known, phenotypes)

The empirical data set is from the autohexaploid hermaphrodite plant Eremophila glabra ssp. glabra from central Australia (details see Field et al., in review, Elliot 2010). The data set includes six SSR loci for a set of adult plants (n = 667) sampled from 15 discrete populations and a set of offspring (n = 467) collected from a subset of plants which are known (mother is among the adults, yet the location of the father is unknown). The data consists of phenotypes (allele dosage unknown), typical of most polyploid codominant marker data.

Assuming the working directory has been set, load the empirical data from the package into R and write as a new CSV:

```
data(Eremophila)
write.csv(Eremophila, "Eremophila_mumKnown.csv", quote = F, row.names = F)
```

Before importing the data from file, we make tables with the double reduction rates (DRR) for each locus. In this example, we make two tables (i) DRRtable_1 contains the true underlying DRR to simulate, (ii) DRRtable_2 are the estimated DRR we will use to calculate assignment likelihoods. Using different tables allows us to examine departures from the true simulated DRR and how they affect assignment accuracy in simulation step. We can use the same tables (see below) to examine the alternative case where the DRR was somehow known with precision. To do this, first we randomly generate DRR rates using:

```
DRRtable_1 <- DRRsetup(numLoci = 6, ploidy = 6, propDRR = 1)
```

and then assume an intermediate DRR (for a hexaploid) at all loci for the actual assignment estimatations:

```
DRRtable_2 <- DRRtable_1
DRRtable_2[, 1] <- 0.13635
```

Similarly, we make two sets of the expected segregation ratios at each locus using the function `segregationRatios()`:

```
segTable_1 <- makeGameteSegTable(ploidy = 6, DRRtable_1)
segTable_2 <- makeGameteSegTable(ploidy = 6, DRRtable_2)
```

Next the data file is imported into R using `inputData()`. This function imports an external data file and performs various checks that the formatting is correct for the main assignment functions. These checks and tidying to the data include:

- the number of columns match the number of loci and ploidy level
- all individual IDs (labels) are unique
- adults and progeny correct identified
- no progeny are also mothers
- the expected number of alleles matches the marker type
- removes individuals with less than the specified number of loci (lociMin)

We must specify a number of features about the data set, including: `numLoci` = the number of loci, `lociMin` = the minimum number of loci to import an individual (less than this number are removed), `ploidy` = copy number of homologous chromosomes, `marker` = "phenotype" (allele copy number known) or "genotype" (allele copy known), `cohorts` = TRUE (adults and progeny) or FALSE (only adults).

```
EremophilaData <- inputData("Eremophila_mumKnown.csv", numLoci = 6,
    lociMin = 5, ploidy = 6, marker = "phenotype", cohorts = T,
    motherknown = T, DRRtable_1)
```

Now we are ready to use the functions `AlleleFreqPheno()` or `AlleleFreqGeno()` to calculate the allele frequencies in each population. If we are confident that the alleles have been sufficiently sampled in the candidate populations then set `replacement = FALSE` and run as:

```
alleleFreq <- alleleFreqPheno(EremophilaData, numLoci = 6, ploidy = 6,
    replacement = FALSE)
```

Alternatively, if there is concern over the possibility of missing alleles and its effects on the multilocus likelihoods, we can use the parameter `replacement = TRUE`. This lets the frequency of the absent allele be proportional to the inverse of the number of gene copies at the locus, adjusted by the number of observed alleles as, $p_{kji} = (1/K_j)/(N_{ij} * Y)$, where K_j is the total number of alleles detected across all populations for the jth locus, N_{i_j} are the total number of individuals sampled in the ith candidate population and Y is the ploidy level (e.g. $Y = 6$ for hexaploid). Given this allele frequency **AutoPoly** re-calculates the probability of the missing alleles at the given locus. Note that this approach is considerably slower to run due to the combinatorial burden of calculating all possible gamete and genotype probabilities.

Next we calculate the expected gamete frequencies at random mating equilibrium using `gameteFreq()`:

```
gameteFreq_1 <- gameteFreq(EremophilaData, alleleFrequencies = alleleFreq,
    numLoci = 6, ploidy = 6, DRRtable = DRRtable_1, DRRtype = "general")
```

Here, we have to define how to calculate the gamete frequencies with `DRRtype` which can be either "general" (formulas for any given DRR defined in DRRtable), "min" (DRR=0 all loci) or "max" (DRR = maximum theoretical value, depending on ploidy). We then make another version of the gamete frequencies using our second set of DRR:

```
gameteFreq_2 <- gameteFreq(EremophilaData, alleleFrequencies = alleleFreq,
    numLoci = 6, ploidy = 6, DRRtable = DRRtable_2, DRRtype = "general")
```

Next, we run simulations of the population assignment using `simAssignmentMums()` to asses the power of the data set and generate confidence intervals:

```
simulationEremophila <- simAssignmentMums(inData = EremophilaData,
    alleleFrequencies = alleleFreq, replacement = T, gameteFreq_actual = gameteFreq_1,
    gameteFreq_est = gameteFreq_2, ploidy = 6, marker = "phenotype",
    numLoci = 6, selfing = 0.1, DRRtable_actual = DRRtable_1,
    DRRtable_est = DRRtable_2, segTable_actual = segTable_1,
    segTable_est = segTable_2, DRRtype = "general", simMissLoci = 0,
    lociMin = 1, epsilon = 0.1, errorRate1 = 0.01, errorRate2 = 0.01,
    output.txt = F, numAdultsSim = 50, numMothersSim = 10, numOffspringSim = 5000,
    fixMothersSim = F, immRateSim = 0.3)
```

The important parameters to note here include: `numAdultsSim` = the number of adults to simulate per population, `numMothersSim` = number of mothers to simulate per population, `numOffspringSim`= the total number of offspring to simulate (distributed across mothers and populations), `immRateSim`= the proportion of interpopulation immigrants to simulate. The details of the other parameters can be found by using the inhelp file `?simAssignmentMums`. The full output of the simulations and their likelihoods are written to a file called `simsAssignmentFull.csv` in your working directory.

Now we run the population assignment with the function `assignmentPheno()`:

```
EremophilaAssignment <- assignmentPheno(inData = EremophilaData,
    alleleFrequencies = alleleFreq, gameteFrequencies = gameteFreq_2,
    numLoci = 6, ploidy = 6, motherknown = T, DRRtable_2, DRRtype = "general",
    segregationRatios = segTable_2)
```

Finally, we can perform posthoc tests and generate a summary of the assignment analysis using the function `summaryAssignment()`:

Here we must include the original data (`inData`), the population assignment output (`results`), and the simulation output (`simulationResults`).

Once this function is complete, two files will be written to the working directory. (i) `assignmentLikelihoods.csv`: contains the full set of assignment likelihoods at all candidate populations, and (ii) `assignmentSummary.csv`: contains the two most likely populations with confidence intervals.

In the `assignmentSummary.csv` you will find the following output:

Table 2: Summary of population assignment of Eremophila data with **AutoPoly** (first 4 individuals)

offspring	home	mother	1	2	LogL1	LogL2	Delta1	critical	loci	miss	M_O_misM	M_O_misM_
GA1-1	GA	GA1	GE	TR	-68.582	-75.021	6.439	***	6	0	0	0
GA1-10	GA	GA1	HP7	GT	-61.488	-64.909	3.421	***	5	0	1	locus.6
GA1-2	GA	GA1	GA	RE	-56.516	-57.278	0.762	*	6	0	0	0
GA1-3	GA	GA1	HeP	GN	-61.073	-61.229	0.155	-	5	0	1	locus.6

The column headers specify:

- offspring = ID name of the offspring
- home = home population from where the offspring was located

- mother = ID name of the mother
- pop1 = most likely population of origin for the offspring
- pop2 = second most likely population of origin for the offspring
- LogL1 = log likelihood for the most likely population of origin for the offspring
- LogL2 = log likelihood for the second most likely population of origin for the offspring
- Delta1 = difference in the log likelihoods between the first and second most likely populations of origin
- critical = significance for delta1, where P = 0.05 ($***$), P = 0.1 ($**$), P = 0.2 ($*$)
- loci = the number of loci with data for the offspring
- miss = names of the loci missing (if applicable)
- M_O_misM = the number of loci mismatching between offspring and mother (for mother known model)
- M_O_misM_L = names of the loci mismatching between offspring and mother (for mother known model)

4.2. Simulated tetraploid data (mother known/unknown, phenotypes/genotypes)

The second data set consists of simulated individuals (adults and offspring) from one realization from the power simulations. This is of the replicates from a metapopulation of 10 populations with a mean FST = 0.09 with 24 SSR loci from Field et al., (in review B). Given this is simulated data, it contains full information (mother known and genotypes) for a larger number of loci (24). However, the data can be reduced in power by using functions available with *AutoPoly* to reduce the number of loci, remove parent information and convert to phenotypes.

details to be completed

References

Field DL, Broadhurst LM, Elliot C, Young AG (in review A). Population assignment in autopolyploids.
Field DL, Broadhurst LM, Young AG (in review B). AutoPoly: An R package for population assignment in autopolyploids.