Using the PolyPatEx package

Alexander Zwart

April 11, 2016

1 What is PolyPatEx?

PolyPatEx is an R package for performing paternity exclusion in autopolyploid, monoecious or diocecious species having ploidy $p = 4n$, $6n$ or $8n$, using allele matching on codominant marker data such as microsatellite data. **PolyPatEx** can also perform exclusion on diploid data, provided the data is *genotypic* (see below for an explanation of *genotypic* vs *phenotypic* data). Note that **PolyPatEx** is suited to low density, high information markers, but is not suited to high density markers such as SNPs.

The analysis is based on direct comparison of the alleles at a given locus (referred to as an 'allele set') to determine whether there is a possible match between a candidate father's allele set and the allele set of the progeny, given the allele set observed in the progeny's known mother. Results of such per-locus matches are then collated across all loci to provide multi-locus progeny-father matches for each progeny in the dataset.

'Candidate fathers' may be referred to simply as 'candidates' in this document and the **PolyPatEx** reference documentation.

PolyPatEx makes several key assumptions. In an autopolyploid species of ploidy p, at a given locus,

- $p/2$ of the p alleles in the progeny are assumed to have been selected, without replacement, from the mother's p alleles (the 'maternal gamete');
- the remaining $p/2$ of the progeny's alleles are assumed to have been selected, without replacement, from the father's p alleles (the 'paternal gamete');
- The relationship between mother and progeny is known—the aim of the analysis is to determine which of one or more candidate fathers is capable of producing a paternal gamete compatible with the alleles in the progeny-mother pair;
- No account is taken of mutation or other events (such as double reduction) which can violate the assumptions of 'selection, without replacement, of $p/2$ alleles from p alleles'.

PolyPatEx does not provide a probabilistic paternity exclusion analysis which would yield information on likelihood of paternity and more elegantly handle issues such as missing data. Consequently no assumptions are made concerning possible correlations between loci (linkage), nor is *random* selection of gametic alleles a necessary assumption. Appropriate caution in interpreting results from this software is advised, in light of the assumptions made by the package.

In collating per-locus comparisons results across multiple loci, the relevant **PolyPatEx** functions *do* provide one simple option to address potential effects of mutation or missing data. In assessing whether a candidate should be flagged as a potential father, the option exists to allow one or more mismatching loci in a progeny-candidate pair that is flagged as a 'match'.

PolyPatEx contains routines to perform paternity exclusion on datasets where marker dosages are known (genotype data) and where marker dosages are unknown ('phenotype' data). In the latter case, the software allows for the fact that comparisons made between allele sets must take into account all of the possible genotypic allele sets that can arise from a given 'phenotypic' allele set.

A typical **PolyPatEx** workflow:

1. Prepare an allele data set file in Comma Separated Value (CSV) format. The required data layout is described in Section 4, **Input data format** on page 3.
2. Start up R, load the **PolyPatEx** package and set the R working directory to the location of the data file.
3. Load the data file into R and perform a number of checks and necessary preprocessing steps on the data using functions provided by the **PolyPatEx** package.
4. Perform the exclusion analysis.
5. Query, and/or output to CSV file, the results of the exclusion analysis.

The workflow is described in more detail in Section 5, **PolyPatEx example workflow** on page 9.

2 Typesetting conventions and notation

In this document, displayed R code and references to R functions, function arguments or data structures are displayed in `typewriter-style` font. References to columns and content of the input allele data file are similarly typeset.

References to R function names are followed by an empty pair of braces, as in `inputData()`. The '`()`' suffix is an informal notation useful for clarifying that the name refers specifically to an R *function*. Similarly, references to function *argument* names are immediately followed by an 'equals' sign, as in `file=`. I hope these conventions will aid (rather than confuse) those users not familiar with R programming.

As they are not formal R conventions, I do not use these notations in the R reference documentation.

3 Key PolyPatEx functions

The functions in **PolyPatEx** that are of interest to the user are (in approximate order of possible use):

- `fixCSV()`—'Tidies' a CSV file prior to input into R.
- `inputData()`—Loads an allele dataset (from a CSV file) into R.
- `preprocessData()`—Performs a number of essential checks and preprocessing steps on the dataset, prior to further analysis by other **PolyPatEx** functions. This function is *automatically called for you* by `inputData()`, but can also be called directly should you load your dataset into R by other means.
- `phenotPPE()`, `genotPPE()`—perform the paternity exclusion analyses on 'phenotypic' or genotypic allele data sets respectively.
- `potentialFatherCounts()`—tabulates the number of potential fathers identified by `phenotPPE()` or `genotPPE()` for each offspring.
- `potentialFatherIDs()`—displays the potential fathers that were identified by `phenotPPE()` or `genotPPE()` for each offspring.

Three other functions of possible interest are:

- `multilocusTypes()`—summarise the unique 'phenotypes' or genotypes in an allele data set.
- `foreignAlleles()`—identify the alleles present in the progeny that are not present in any of the adults in a dataset.
- `convertToPhenot()`—convert a genotype dataset to a 'phenotype' dataset by removing all repeated alleles in each allele set.

Reference documentation for these functions can be found via the R HTML package documentation, or by typing, e.g.:

```
?fixCSV
```

at the R command prompt.

4 Input data format

4.1 Data table schematic

PolyPatEx requires the allele data to be presented as a Comma Separated Value (CSV) file in a specific format, which we now describe.

The first three columns of the data set are named `id`, `popn` and `mother`, in that order. The **PolyPatEx** scripts are case-sensitive, so ensure the column names are in lower case as shown.

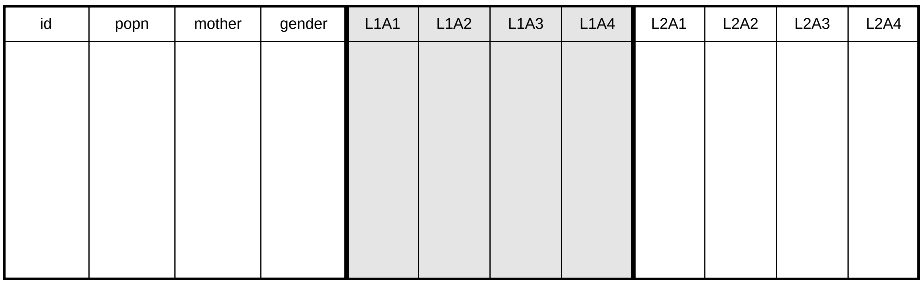
If dealing with data from a dioecious species, the fourth column of the data set is named `gender`.

These first three (or four) columns are immediately followed by $p \times k$ further columns, where p is the ploidy (4, 6 or 8) of the data set, and k is the number of loci being analysed. These columns contain the marker allele information for the first locus in the first p columns, information for the next locus in the next p columns, and so on. Each cell in this part of the dataset contains either an allele label, or is left empty or contains an appropriate missing data symbol (see below).

The $p \times k$ marker allele columns should all be named. Choose whatever names you like, provided they are unique and contain no commas (to avoid confusion with the CSV format). It is recommended to keep column names brief, and avoid using spaces or symbols other than periods or underscores in column names - otherwise, R may change column names when loading the data, to suit its own restrictions on valid variable names.

Note that in **PolyPatEx** output, the k loci will be referred to simply as `Locus 1` up to `Locus k`, in the order that the groups of p columns occur in the data file.

Figure 1 is a schematic of the required data layout, for a dioecious data set with ploidy $p = 4$, and $k = 2$ loci. A dataset for a monoecious species would neglect the gender column.



id	popn	mother	gender	L1A1	L1A2	L1A3	L1A4	L2A1	L2A2	L2A3	L2A4

Figure 1: Schematic of the input data layout required by **PolyPatEx**, for a dioecious species - for a monoecious species, neglect the `gender` column.

4.2 Data column contents

PolyPatEx assumes a set of marker data from a number of subjects (individual plants or animals), which include progeny, their mothers, and other adults—the dataset should contain a row for each subject. All included progeny should have their (known) mother present in the data set.

Since the data is to be loaded from a Comma Separated Variable (CSV) format, do not use commas in any id or allele labels or column names, to avoid conflict with the use of the comma as a separator by the CSV format.

Certain cells of the data set should be left empty in the CSV file as described below. When the dataset is loaded into R using `inputData()`, these blank cells are automatically converted to contain `NA`, which is R's symbol for a missing datum.[1] In the CSV file, you may also use * or NA (note: uppercase) for these 'blank' cells if you prefer. R users who create or load their data by means other than inputData() should ensure that all 'blank' cells contain NA prior to calling preprocessData()).

- Column id contains a unique identifying label for each individual in the dataset.
- Column popn contains a label for the population each subject comes from. Note that although the popn must be present in the dataset, it is not currently used by PolyPatEx—all analyses are applied ignoring population distinctions within a given allele dataset.
- Column mother contains, for each offspring, the id label of the mother of that offspring. For all non-progeny, entries in this column should be left blank.
- For dioecious data, column gender contains M for male adults, or F for female adults. Entries in this column for progeny should be left blank.
- The remaining $p \times k$ columns contain the allele labels, one label per cell. Where fewer than p alleles were observed at a locus, the remaining cells for that locus-subject combination should be left blank.
- *Do not use spaces in allele labels.* **PolyPatEx** functions use spaces as delimiters between allele labels in their processing of the data, so labels that already contain spaces will cause errors to occur.

Table 1 is an example of a (very small) 'phenotype' data set for a monoecious species with ploidy $6n$ having just two observed loci.

In Table 1, note that:

- GF1 and GF13 are mothers of progeny in the dataset. Their progeny are indicated by the presence of GF1 or GF13 in the mother column.
- GF14 up to GF17 are other adults.
- Blank cells in each block of six columns indicate where fewer than six alleles were observed. Locus L2 appears to have fewer unique alleles overall (with heavier multiplicity) than Locus L1.
- The order of the rows is unimportant, but the order of the columns *is* important!

Table 2 is an example of a genotype data set for a dioecious species having a ploidy $4n$ and three observed loci. Subjects 1913 and 1914 are mothers; their progeny are again indicated by the mother id's in the mother column. Two of the subject-locus combinations failed to record any alleles and have therefore been left blank.

[1] NA is a special symbol recognised by R, and is not to be confused with "NA" or 'NA', which are each character strings containing the letters 'N' and 'A'. Also, R is case sensitive, so use NA, not na or Na.

id	popn	mother	L1a	L1b	L1c	L1d	L1e	L1f	L2a	L2b	L2c	L2d	L2e	L2f
GF1	GF		252	249	255	267	268	279	367					
GF13	GF		249	250	255	264	272	277	367	369	374			
GF14	GF		249	252	255				367					
GF15	GF		246	249	250	277	264	278	367	369				
GF16	GF		249	250	255	264	252	282	367	368	374			
GF17	GF		249	252	255	257	282		367	369	375			
GF1-2310	GF	GF1	249	255	267	272	282		367					
GF1-2311	GF	GF1	244	249	255	268	279							
GF1-2315	GF	GF1	249	252	255	264	266		367	374				
GF1-2316	GF	GF1	252		268	279	267		367					
GF13-2480	GF	GF13	249	252	255	264	277		367	369	374			
GF13-2481	GF	GF13	249	250	264	277	252		368	369				
GF13-2482	GF	GF13	249	250	255	252			369	374				
GF13-2485	GF	GF13	250	264	272	277			367	369				
GF13-2487	GF	GF13	249	264	277	252			367	369	368			
GF13-2491	GF	GF13	250	253	264	272	277	252	368	374				
GF13-2492	GF	GF13	249	250	264	277			367	369	374			
GF13-2493	GF	GF13	250	255	264	268	277		369	374	368			
GF13-2495	GF	GF13	249	252	264	267	277		367	374				
GF13-2496	GF	GF13	249	250	252	255	266		368	374				

Table 1: Small example ‘phenotype’ data set for two observed loci in a monoecious species having ploidy $6n$.

id	popn	mother	gender	SB85a	SB85b	SB85c	SB85d	SB101a	SB101b	SB101c	SB101d	SB243a	SB243b	SB243c	SB243d
2089	FR		M	103	103	103	103	179	179	179	179	126	120	120	120
2090	FR		M	103	103	103	106	179	179		179	120	120	120	123
2091	FR		M					179	179	179	179	117	120	120	126
2092	FR		M	100	103	103	106	179	179	179	187	117	120	120	126
2208	FR		M	100	100	103	103	179	179	179	179	117	120	120	126
1913	FR		F	103	103	103	103	179	179	179	179	117	120	120	123
2800	FR	1913		103	103	103	103	179	179	179	179	117	120	120	123
2809	FR	1913		103	103	103	103	179	179	179	179	120	120	123	123
2810	FR	1913		103	103	103	103	179	179	179	179	117	120	123	126
1914	FR		F	103	103	103	103	179	179	179	179	117	117	117	123
2820	FR	1914		103	103	103	103	179	179	179	179	117	120	120	123
2829	FR	1914		103	103	103	103	179	179	186	186	117	117	117	123
2824	FR	1914		103	103	103	106					117	117	117	126
2825	FR	1914		103	103	103	103	179	179	179	186	120	120	120	123

Table 2: Small example genotype data set for three observed loci in a dioecious species having ploidy $4n$. The gender column contains codes M for Male, and F for Female.

4.3 Spreadsheets, CSV format, and the `fixCSV` function

Most users will prepare their data in a commercial spreadsheet application, then export or 'Save As...' the data to a CSV formatted file. In some spreadsheets applications, when 'Save As...' is used, the process can be a little confusing, so we strongly recommend you save your spreadsheet file and *make a backup of it before you attempt to create the CSV file.*

A further complication that can arise relates to the cells of the data table that are represented in the CSV file when it is created. The CSV format is a plain text format, and when viewed as a plain text file, the CSV file for a rectangular table should ideally contain one fewer commas (column separators) per row as there are columns in the table. The data set of Table 1 contains 15 columns, so ideally there should be exactly 14 commas in each row of the resulting CSV file, as in Table 3 below.

```
id,popn,mother,L1a,L1b,L1c,L1d,L1e,L1f,L2a,L2b,L2c,L2d,L2e,L2f
GF1,GF,,252,249,255,267,268,279,367,,,,,
GF13,GF,,249,250,255,264,272,277,367,369,374,,,
GF14,GF,,249,252,255,,,,367,,,,,
GF15,GF,,246,249,250,277,264,278,367,369,,,,
GF16,GF,,249,250,255,264,252,282,367,368,374,,,
GF17,GF,,249,252,255,257,282,,367,369,375,,,
GF1-2310,GF,GF1,249,255,267,272,282,,367,,,,,
GF1-2311,GF,GF1,244,249,255,268,279,,,,,,,
GF1-2315,GF,GF1,249,252,255,264,266,,367,374,,,,
GF1-2316,GF,GF1,252,,268,279,267,,367,,,,,
GF13-2480,GF,GF13,249,252,255,264,277,,367,369,374,,,
GF13-2481,GF,GF13,249,250,264,277,252,,368,369,,,,
GF13-2482,GF,GF13,249,250,255,252,,,369,374,,,,
GF13-2485,GF,GF13,250,264,272,277,,,367,369,,,,
GF13-2487,GF,GF13,249,264,277,252,,,367,369,368,,,
GF13-2491,GF,GF13,250,253,264,272,277,252,368,374,,,,
GF13-2492,GF,GF13,249,250,264,277,,,367,369,374,,,
GF13-2493,GF,GF13,250,255,264,268,277,,369,374,368,,,
GF13-2495,GF,GF13,249,252,264,267,277,,367,374,,,,
GF13-2496,GF,GF13,249,250,252,255,266,,368,374,,,,
```

Table 3: The data of Table 1 in CSV format, viewed as plain (ascii) text.

Note in Table 3 that the empty cells at the right hand side of the data table (due to the sparsity of distinct alleles in the phenotypic data at locus `L2`) are still represented in the CSV file, hence each row contains exactly 14 commas (even though the trailing commas may be thought of as somewhat redundant).

The **PolyPatEx** function `inputData()` is pedantic on this point—it will complain if a column contains too many or too few commas in any of the rows. Also, any empty rows appearing below the data table in the CSV file will cause errors to occur when the data is checked by `preprocessData()`. When a CSV file is created by exporting/saving from a spreadsheet program:

- The spreadsheet may drop trailing commas that it sees as redundant in a row, particularly if there would be a large number of such commas (this can happen if there are lots of empty cells at the right hand side of a data table, as in the example of Table 1);
- The spreadsheet may think that the data area is larger than it is meant to be, perhaps due to edits or formatting changes made in cells beyond the rows and columns of the data table (even if the content that was previously there has since been deleted). This can result in excess commas in a row, or commas appearing below the data table.

Both possibilities arise because spreadsheet applications typically provide a working space that is larger (in terms of number of rows and columns) than the final data table to be saved—so the spreadsheet has to make decisions as to which of the 'blank' cells in the spreadsheet should be represented in the CSV file.

There are two steps to ensure these problems do not occur:

1. To avoid problems due to previous edits outside the data table, it is a good idea to copy the completed data table from its original spreadsheet into a newly created spreadsheet or spreadsheet tab, before exporting to CSV format. If you copy *only the rectangle containing the data and column headers*, you should leave behind any edits outside the data region which could otherwise cause problems when exporting to CSV format.
2. In addition, **PolyPatEx** provides the function `fixCSV()`, which can be used to 'tidy up' a CSV file before you load it into R with `inputData()`. Function `fixCSV()` will check the header row of the dataset in the CSV file and will add or remove trailing commas in each subsequent row as needed, to obtain the correct number of commas in each row. `fixCSV()` can also recognise and remove redundant empty rows below the main body of data should they occur (if you have carried out step 1 above, this should not occur). By default `fixCSV()` will write the result to a new CSV file having the word 'FIXED' appended to the original filename. An option exists to overwrite the original file, but if you wish to use this option *make sure you have backed up your data file first*!

5 PolyPatEx example workflow

Prior to using **PolyPatEx** for the first time, you should ensure that you have installed package **gtools** from the **Comprehensive R Archive Network** (CRAN). If not already installed, and assuming your computer is connected to the internet, the following R command can be used to download and install **gtools**:

```
install.packages('gtools')
```

The **PolyPatEx** R package contains two example microsatellite data sets in the required input file format. Once **PolyPatEx** is installed, the following R command will print out the location of these files:

```
system.file("extdata", package = "PolyPatEx")
```

These data sets may also be found on the **PolyPatEx** website:

```
http://www.anbg.gov.au/cpbr/tools/polypatex/
```

The file `GF_Phenotype.csv` contains data from seven loci in a hexaploid, monoecious species, *Eremophila glabra ssp. glabra* (Elliott 2010). The file `FR_Genotype.csv` contains data from seven loci in a tetraploid, dioecious species, *Salix cinerea* (Hopley 2011).

Let us assume that the file `GF_Phenotype.csv` has been copied to a suitable directory, and the R working directory has been set to this location (via the R file menu, or see the R function `setwd()`). Table 4 shows example R code to load the data set into R, perform the exclusion analysis, and output results from the summary functions `potentialFatherCounts()` and `potentialFatherIDs()` to CSV files for scrutiny in a spreadsheet application. Note that R code is case sensitive.

In Table 4, the `require()` function is used to load the **PolyPatEx** package.[2] The data file is then checked for possible CSV format issues discussed in Section 4.3, using `fixCSV()`—the option `overwrite=TRUE` has been specified to overwrite the original file with the 'corrected' version should any issues be found - always ensure you have packed up the original versions of the your datasets before using this option!

Function `inputData()` is then used to input the allele data from the CSV file. The data set is immediately passed by `inputData()` to `preprocessData()` which performs a number of checks and preprocessing steps on the dataset before the result is returned in the form of an R data frame - this is stored as `adata` in the example of Table 4. Users should not make changes to this data frame before analysing it with other **PolyPatEx** functions, unless they re-run `preprocessData()` on the data frame again. Users who wish to load or create the allele data set in R without using `inputData()` must run `preprocessData()` on the data frame prior to using further **PolyPatEx** analysis functions.

Further information on the key checks and preprocessing performed by `preprocessData()` is provided in Section 6.

Arguments to `inputData()` (or when called directly, `preprocessData()`) should specify the details of the data to be loaded:

- `numLoci=` The number of loci k, in the dataset.
- `ploidy=` The ploidy p (4, 6, or 8) of the autopolyploid species being analysed. The ploidy can also be 2, provided `dataType="genotype"`.
- `dataType=` Either `"genotype"` (allele copy numbers known) or `"phenotype"` (allele copy numbers unknown).

[2] For newcomers to R: When an R package is *installed*, its content is placed where R can access it on your computer's hard drive—but in an R session, R does not try to access that content until the package is *loaded* via the `require()` or `library()` functions. Loading the **PolyPatEx** package is required for each new R session, before you can access **PolyPatEx** functions. Provided you have installed package **gtools**, you need not `require()` it before using **PolyPatEx**—**PolyPatEx** functions know how to access **gtools** content directly.

```
require(PolyPatEx)

fixCSV("GF_Phenotype.csv",overwrite=TRUE)

adata <- inputData("GF_Phenotype.csv",
                   numLoci=7,
                   ploidy=6,
                   dataType="phenotype",
                   dioecious=FALSE,
                   selfCompatible=TRUE)

ppe1 <- phenotPPE(adata)

write.csv(potentialFatherCounts(ppe1),file="pFatherCounts.csv")

write.csv(potentialFatherIDs(ppe1),file="pFatherIDs.csv")
```

Table 4: Example code to perform a paternity exclusion analysis on a phenotype dataset.

- `dioecious=` Is the species dioecious (`dioecious=TRUE`) or monoecious (`dioecious=FALSE`)?
- `selfCompatible=` If the species is monoecious, should an offspring's mother be regarded also as a candidate father (`selfCompatible=TRUE`) or not (`selfCompatible=FALSE`)? If `dioecious=TRUE`, you need not specify this argument.
- `mothersOnly=` If the species is dioecious, should female adults that are not mothers be removed from the dataset (`mothersOnly=TRUE`) or not (`mothersOnly=FALSE`)? Female adults that are not mothers do not play a part in the exclusion analysis, but this option does affect results from the functions `multilocusTypes()` and `foreignAlleles()`.
- `lociMin=` The minimum required number of loci that must have alleles present for the individual to be retained in the dataset. This argument has a default value of `lociMin=1`. For more on this argument, see Section 6.
- `matMismatches` The minimum allowed number of loci in a given offspring that are allowed to mismatch between mother and offspring before the offspring is removed from the dataset. For more on this argument, see Section 6.

Once the data has been loaded into R, checked by `preprocessData()` and stored as an R data frame object (here called `adata`), the data frame is passed to one of the exclusion routines, in this case `phenotPPE`, which performs the paternity exclusion analysis.

The result of the exclusion analysis is an R 'list' structure, which is stored here as `ppe1`. This list will generally be very large, and its contents are explained in more detail in the reference information for functions `phenotPPE` and `genotPPE`.

However, the contents of the list `ppe1` will usually not be the most useful output for interpretation. Functions `potentialFatherCounts()` and `potentialFatherIDs()` are provided to summarise the results of the analysis in a more useable form - these functions return R data frames summarising the number of non-excluded candidates (i.e., potential fathers) for each offspring in the dataset, and the ID's of potential fathers of each offspring.

Two optional arguments to these functions (not shown in the example) should be noted. Missing allele sets in progeny, mother or candidate father and issues such as mismatching allele sets between progeny and mother, can result in loci at which exclusion comparisons with a given candidate cannot be made. Argument `VLTMin=` can be used to set the minimum number of 'valid' loci (at which valid exclusion comparisons could be made) that are required for a candidate to be assessed as a potential father to a given offspring. Candidates with fewer than this number of 'valid' loci are ignored. The default value is `VLTMin=1`.

Argument `mismatches=` can be used to specify a maximum number of *mismatching* loci that are allowed in a candidate that is still flagged as a potential father. This option can provide some allowance for the possibility of mismatches occurring due to mutations or genotyping errors. The default value is `mismatches=0`, hence all valid loci must provide an offspring-candidate match before the candidate is flagged as a potential father.

In the example code, the results of calls to `potentialFatherCounts()` and `potentialFatherIDs()` have been saved directly to CSV files via R's `write.csv()`, rather than being saved as R objects.

6 More on function `preprocessData()`

As mentioned, function `preprocessData()` performs a number of checks and preprocessing steps on the input allele dataset, and is a prerequisite to using any of **PolyPatEx**'s analysis functions. `preprocessData()` may stop with an error message if certain errors in the data layout or content are found, requiring the user to correct the dataset before calling `fixCSV()`, `inputData()` and/or `preprocessData()` again as appropriate.

Certain other issues result in affected allele sets being reset to have no alleles. In genotype datasets, allele sets that do not contain p alleles are reset to contain no alleles (a 'missing' allele set), and subsequent analyses will ignore this locus in this individual - hence, for example, reducing the number of 'valid' loci remaining from the exclusion analysis. (In allelic phenotype datasets, there is no requirement to observe exactly p unique alleles, so this adjustment is only relevant to genotypic data.) Such changes to the allele dataset only affects the data frame in R - `preprocessData` does not change the input CSV file.

As an example, offspring-mother pairs are checked by `preprocessData()` for loci in which the allele sets are incompatible (a mother-offspring 'mismatch'). In genotype datasets, the condition for compatibility between mother and offspring is simply that the mother's allele set must be able to provide at least $p/2$ of the alleles present in the offspring's allele set. In phenotype datasets, where allele copy numbers are not known, the comparison is more complex—the software must search through the possible genotypes arising from the observed phenotypes before it can confirm whether a mother's observed alleles compatible with the offsprings observed alleles.

An argument, `matMismatches` is provided that allows the user to specify the number of mismatching loci between mother and offspring that are allowed before the offspring is removed from the dataset. `matMismatches` may be between `0` and `numLoci - 1`. If a given offspring has `matMismatches` or fewer loci mismatching with its mother, the affected loci in the *offspring* are set to be missing (i.e. to contain no non-missing alleles). If an offspring has greater than `matMismatches` missing loci, its data will be removed from the dataset returned by `preprocessData`. The default value for `matMismatches` is `0`. Details of the affected individuals and loci involved are reported to the user, so that they may, if they wish, check for and correct possible data errors in the input CSV file.

7 The exclusion comparisons

At each locus, the exclusion routines `genotPPE` and `phenotPPE` look for candidate fathers whose allele set is compatible with the alleles in the offspring, given the alleles present in the mother's allele set. Comparisons are made neglecting the possibility of mutation or other mechanisms that may violate the assumptions described in Section 1.

In genotype datasets, this means that the candidate must be able to account for any alleles in the offpsring that cannot be provided by that offspring's mother, plus as many more of the offspring's alleles (also found in the mother) as are needed to make up a full gamete's component of $p/2$ alleles.

In phenotype datasets, where allele copy numbers are unknown, the situation is more complex, since a proper comparison between offspring and candidate father (given the offspring's mother) must take into account all of the possible genotypes (totalling p alleles per allele set) that could have arisen from the 1 to p alleles observed at the locus in each of the paired individuals. In `phenotPPE`, the search through all possible comparisons is implemented via previously calculated lookup tables included in the package. Currently, lookup tables are provided for ploidies $4n$, $6n$ and $8n$.

8 PolyPatEx - License

CSIRO Open Source Software License Agreement (GPLv3)

Copyright (c) 2014, Commonwealth Scientific and Industrial Research Organisation (CSIRO) ABN 41 687 119 230.

All rights reserved. CSIRO is willing to grant you a license to the PolyPatEx R package on the terms of the GNU General Public License version 3 as published by the Free Software Foundation (http://www.gnu.org/licenses/gpl.html), except where otherwise indicated for third party material.

The following additional terms apply under clause 7 of that license:

EXCEPT AS EXPRESSLY STATED IN THIS AGREEMENT AND TO THE FULL EXTENT PERMITTED BY APPLICABLE LAW, THE SOFTWARE IS PROVIDED ”AS-IS”. CSIRO MAKES NO REPRESENTATIONS, WARRANTIES OR CONDITIONS OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY REPRESENTATIONS, WARRANTIES OR CONDITIONS REGARDING THE CONTENTS OR ACCURACY OF THE SOFTWARE, OR OF TITLE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, THE ABSENCE OF LATENT OR OTHER DEFECTS, OR THE PRESENCE OR ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE.

TO THE FULL EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL CSIRO BE LIABLE ON ANY LEGAL THEORY (INCLUDING, WITHOUT LIMITATION, IN AN ACTION FOR BREACH OF CONTRACT, NEGLIGENCE OR OTHERWISE) FOR ANY CLAIM, LOSS, DAMAGES OR OTHER LIABILITY HOWSOEVER INCURRED. WITHOUT LIMITING THE SCOPE OF THE PREVIOUS SENTENCE THE EXCLUSION OF LIABILITY SHALL INCLUDE: LOSS OF PRODUCTION OR OPERATION TIME, LOSS, DAMAGE OR CORRUPTION OF DATA OR RECORDS; OR LOSS OF ANTICIPATED SAVINGS, OPPORTUNITY, REVENUE, PROFIT OR GOODWILL, OR OTHER ECONOMIC LOSS; OR ANY SPECIAL, INCIDENTAL, INDIRECT, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES, ARISING OUT OF OR IN CONNECTION WITH THIS AGREEMENT, ACCESS OF THE SOFTWARE OR ANY OTHER DEALINGS WITH THE SOFTWARE, EVEN IF CSIRO HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH CLAIM, LOSS, DAMAGES OR OTHER LIABILITY.

APPLICABLE LEGISLATION SUCH AS THE AUSTRALIAN CONSUMER LAW MAY APPLY REPRESENTATIONS, WARRANTIES, OR CONDITIONS, OR IMPOSES OBLIGATIONS OR LIABILITY ON CSIRO THAT CANNOT BE EXCLUDED, RESTRICTED OR MODIFIED TO THE FULL EXTENT SET OUT IN THE EXPRESS TERMS OF THIS CLAUSE ABOVE ”CONSUMER GUARANTEES”. TO THE EXTENT THAT SUCH CONSUMER GUARANTEES CONTINUE TO APPLY, THEN TO THE FULL EXTENT PERMITTED BY THE APPLICABLE LEGISLATION, THE LIABILITY OF CSIRO UNDER THE RELEVANT CONSUMER GUARANTEE IS LIMITED (WHERE PERMITTED AT CSIRO’S OPTION) TO ONE OF FOLLOWING REMEDIES OR SUBSTANTIALLY EQUIVALENT REMEDIES:

(a) THE REPLACEMENT OF THE SOFTWARE, THE SUPPLY OF EQUIVALENT SOFTWARE, OR SUPPLYING RELEVANT SERVICES AGAIN;

(b) THE REPAIR OF THE SOFTWARE; (c) THE PAYMENT OF THE COST OF REPLACING THE SOFTWARE, OF ACQUIRING EQUIVALENT SOFTWARE, HAVING THE RELEVANT SERVICES SUPPLIED AGAIN, OR HAVING THE SOFTWARE REPAIRED.

Note: The GNU General Public License version 3 can also be viewed at `http://www.r-project.org/licenses/` or in the file `share/licenses/GPL-3` in the R (source or binary) distribution of the PolyPatEx package