MQ^2 Documentation

Release 1.1.0

Pierre-Yves Chibon

July 01, 2013

CONTENTS

A simple python module to process output from QTL mapping tool including MapQTL and R/qtl.

Assuming one QTL per linkage group and using the LOD threshold set by the user. This application extracts all the QTLs detected by the QTL mapping tool, it finds the closest marker and finally put the number of QTLs found for each marker on the genetic map.

This approach quickly allows you to find potential QTL hotspot in your dataset. This is particularly usefull for large QTL analysis on a large number of traits.

CHAPTER

ONE

DOCUMENTATION:

1.1 Install

1.1.1 Dependencies

MQ² requires python 2 (version 2.6 and above) and is compatible with Python 3.

In addition to python MQ² has the following dependencies:

- straight.plugin
- xlrd (only required by the Excel plugin, if not present only this plugin will be affected)

1.1.2 Install methods

Install release

MQ² releases are uploaded on pypi and you can find them on the pypi MQ2 page.

Having MQ² on pypi allows `easy_install`. You can therefore install the latest MQ² release by running:

```
easy_install MQ2
```

Note: This is the recommanded method for users running Windows.

Install from sources

On the pypi MQ² page you can also download the latest version of the MQ². Once downloaded, you can extract its content and install it via the `setup.py`.

The steps are then:

- Download the latest release from MQ2 pypi page
- Extract the files somewhere on your system
- Install MQ² using the command:

  ```
  python setup.py install
  ```

Install from git

To install the development version, you need to have git installed on your system.

Retrieve the sources from the git using the command:

```
git clone https://github.com/PBR/MQ2.git
```

Then you can either

- run MQ² from the cloned repository using:

  ```
  python MQ2/mq2.py --help
  ```

- install MQ² on your system via the command

  ```
  python setup.py install
  ```

1.2 Prepare your data

In order to run correctly MQ², you may have to prepare a little bit you data.

1.2.1 Running from MapQTL output

For each project, MapQTL generates a `.mqp` file containing information about the project. Within the same folder, MapQTL generates a folder that contains all QTL mapping results. The results folder has the same name as the project and ends with the extension: `.mpd`.

To prepare the QTL mapping data for MQ: - locate the `.mpd` folder - create a zip archive of this folder

Only the files with the extension *.mqo* are required, however, the presence of other files does not affect the results.

1.2.2 Running from R/qtl output

MQ² can be used on data generated using R/qtl. This data has to be provided in a CSV file or an Excel document.

Example R code to generate a MQ² compatible output file from R/qtl

```
library(qtl)
data(fake.bc)

# Perform the QTL mapping analysis
fake.bc <- calc.genoprob(fake.bc, step=2.5)
qtl_out <- scanone(fake.bc, pheno.col=c(1:length(fake.bc$pheno)))

write.csv(qtl_out, file='rqtl_out.csv')
```

See *Running from other QTL mapping tools* for more explanations about the format.

1.2.3 Running from other QTL mapping tools

MQ² can be run on data from any QTL mapping tool according the data can be transformed to a Comma Separated Value (CSV) file (using commas `,` as delimiter) or in an Excel document.

The format of the input is very important for MQ² to work. The CSV or Excel sheet should be formatted as follow:

- 1st column contains the markers
- 2nd column contains the linkage groups
- 3rd column contains the position of these markers
- 4th and following columns contain the trait data: the LOD value associated to each marker for this trait.

The first row of the document contains the headers (Markers, Linkage Group, position, Trait name1, Trait name2, etc).

The screenshot below presents how the data should be formated:

	A	B	C	D	E	F	G
1		chr	pos	pheno1	pheno2	Trait 3	Trait 4
2	D1M430	1	0	0.6617472931	0.8342067823	0.8834137855	0.0825279093
3	c1.loc2.5	1	2.5	0.7269621239	0.8504986784	0.9065070363	0.1279394177
4	c1.loc5	1	5	0.7790906581	0.859657896	0.9115121893	0.178422737
5	c1.loc7.5	1	7.5	0.8190823064	0.8628030898	0.9014637989	0.2313521931
6	D1M318	1	9.8	0.8466105122	0.8613983719	0.8821573276	0.2803080129
7	c1.loc10	1	10	0.8444246561	0.8564908068	0.8828229651	0.277641572
8	c1.loc12.5	1	12.5	0.8124159915	0.7835078808	0.8750017498	0.2440978501
9	D1M212	1	13.11	0.8034223114	0.7624578401	0.8690215932	0.2359481028
10	c1.loc15	1	15	0.7251542941	0.6865992366	0.8189291853	0.203756772
11	c1.loc17.5	1	17.5	0.6126848308	0.5792159868	0.7430835657	0.1598913733
12	c1.loc20	1	20	0.4932165979	0.4668625949	0.657288466	0.1165394042
13	c1.loc22.5	1	22.5	0.3722689578	0.354412031	0.5638913636	0.0764208998
14	c1.loc25	1	25	0.2573679502	0.2481773121	0.4666323232	0.0425310026
15	c1.loc27.5	1	27.5	0.1570299716	0.1551245542	0.3702857753	0.0175724016
16	c1.loc30	1	30	0.0788310461	0.0814817524	0.2799152349	0.0033303283
17	c1.loc32.5	1	32.5	0.0273121669	0.0311989858	0.199946517	0.0002533718
18	c1.loc35	1	35	0.0029071875	0.0050429948	0.1333982027	0.0074296781
19	D1M437	1	37.11	0.0011884124	0.0001815142	0.0886619641	0.0200728301
20	c1.loc37.5	1	37.5	0.0012947144	0.000561299	0.085346705	0.0211487634
21	c1.loc40	1	40	0.002080597	0.0080894819	0.0648385644	0.0286251425
22	c1.loc42.5	1	42.5	0.0030256602	0.0243719726	0.0463018429	0.0368911452
23	c1.loc45	1	45	0.0040882172	0.0485601416	0.0305365977	0.0455907583
24	c1.loc47.5	1	47.5	0.0052174641	0.0790110201	0.0180528314	0.0543396188
25	c1.loc50	1	50	0.006361623	0.1136544689	0.0090192889	0.0627832925

Figure 1.1: Structure the CSV file and Excel documents should have to be processed by MQ².

The CSV or Excel document should be compressed into a .zip archive to be uploaded on the web-interface.

Note: MQ² can only analyze one CSV file or one Excel document at a time, however, the Excel document may contain multiple sheets.

1.3 Usage

The command line version of MQ² is ran via the `MQ2` command.

1.3.1 Options

All the options can be listed via `MQ2 --help` or `MQ2 -h`.

These options are:

- `-h` / `--help`, this will simply present you the list of all the options available and what they represent, a little bit like the present document but shorter.

- `-z` / `--zipfile`, this is one of the two options to specify to MQ² your input files. With this option you are providing to MQ² a zip archive containing the files generated by your QTL mapping tool.
- `-d` / `--dir`, this is the second option to specify to MQ² your input files. With this option, you are pointing MQ² to a directory containing your input files.
- `-f` / `--file`, this is the third option to specify to MQ² an input file. With this option, you are pointing MQ² to a single input file.
- `--lod`, this is the option to specify which LOD value from which a QTL is considered to be significant. You can run a permutation test in MapQTL to determine the optimal LOD threshold to use.
- `--session`, this is the option that allows you to specify which MapQTL session or which sheet of an Excel document to analyse in your project. Within one MapQTL project, one can analyse the same data with different approaches or different parameters creating a different session each time. Within a single Excel document, one may have several sheets each containing the results of different analysis (methods, parameters, data). The session number you provide to this option is the session or name of the sheet that you would like to analyse.
- `--verbose`, this option is mostly of interest to have a more verbose output when running MQ².
- `--debug`, this option is mostly of interest if you are facing a problem with MQ². Turn in the option and send us the output in order to improve MQ².

1.3.2 Example commands

A basic command of MQ² will look like:

```
MQ2 --folder /path/to/folder --lod <lod_threshold> --session <mapqtl_session>
MQ2 --zipfile /path/to/folder/archive.zip --lod <lod_threshold> --session <ExcelSheetName>
MQ2 --file /path/to/folder/output.csv --lod <lod_threshold>
```

For example:

```
MQ2 --folder c:\Documents\mapqtl\AnalysesB\ --lod 3.2 --session 3
MQ2 --zipfile c:\Documents\rqtl\Excel\excel_output.zip --lod 3.2 --session="Sheet2"
MQ2 --file c:\Documents\rqtl\csv\rqtl_out.csv --lod 3.2
```

Note: MQ2 will generate its output in the current working directory. Be aware of this when you run it several time on different dataset or with different parameters.

1.4 Output

MQ², command line or via its web interface, will generate a number of output files for each steps of the procedure, allowing to follow how the final results have been generated.

1.4.1 qtls.csv

The `qtls.csv` file list all the QTLs found while parsing the MapQTL output files. MQ² only extracts the strongest QTL above the LOD threshold per linkage group for each trait. Each row of this file corresponds to the peak of one of the QTL found.

When running a MapQTL, one can ask MapQTL to impute putative markers in between the markers given in the genetic map. As a results, in the *Locus* column of the `qtls.csv` file, some rows might have no values. This means that there are no marker from the genetic map at this position.

All columns in the `qtls.csv` file are directly coming from the MapQTL output. They correspond to the values that are displayed in the *Result* tab within MapQTL.

1.4.2 qtls_with_mk.csv

The `qtls_with_mk.csv` correspond to the same file as the *qtls.csv* with the addition of an extra column *Closest marker* which correspond to the name of the marker closest to the position of this QTL peak.

1.4.3 map.csv

The `map.csv` file corresponds to the genetic map provided by the user for the MapQTL analysis and that MQ² retrieves from the MapQTL output.

1.4.4 map_with_qtl.csv

The `map_with_qtl.csv` file corresponds to the same file as the *map.csv* with the addition of an extra column *# QTLs* corresponding to the number of QTLs found at this specific marker.

The number of QTLs is compiled from the *qtls_with_mk.csv* file using the added column *Closest marker*.

1.4.5 qtls_matrix.csv

The `qtls_matrix.csv` file is a matrix providing for each trait analysed and for each marker on the genetic map the LOD value found by MapQTL.

An extra column *# QTLs* is added at the end of the file providing the number traits having a LOD value above the LOD threshold for that specific marker.

This matrix gives the possibility to have an overview of the QTL interval for each trait.

1.4.6 MapChart.map

New in version 0.2. MapChart is a window-specific (freely available) program to visualize QTLs on a genetic map.

MQ² provides a `MapChart.map` output file which can be loaded directly into MapChart and will allow a more detail visualisation of the QTL intervals on the genetic map.

More information about MapChart can also be found in:

```
Voorrips, R.E., 2002. MapChart: Software for the graphical presentation of
linkage maps and QTLs. The Journal of Heredity 93 (1): 77-78.
```

1.5 Contributing

MQ² is licensed under the GPLv3 or any later version. This means that you are free to use the tool, observe how it works, change it and redistribute it.

We also welcome patches and request for enhancement.

If you're submitting patches to MQ², please observe the following:

- Check that your python code is PEP8-compliant. There is a pep8 tool that can automatically check your source.

- Check your code quality usint pylint.
- Check that your code doesn't break the test suite. The test suite can be run using `nosetest`.
- If you are adding new code, please write tests for them in `test/`,
- If your change warrants a modification to the docs in `doc/` or any docstrings in `MQ2/` please make that modification.

Note: You have a doubt, you don't know how to do something, you have an idea but don't know how to implement it, you just have something bugging you?

Contact us by email or just open a ticket on github.

1.6 Extend MQ², write your own plugin

If you the QTL mapping tool you use is not currently supported by MQ², it might be easily added by adding a plugin specific to this format/tool.

To create a new plugin, you will need to create a new python file, place it under `MQ2/plugins/`. This python file should contain its own class which inherits and implements the method defined in `PluginInterface`.

1.6.1 Plugin interface:

class `MQ2.plugin_interface.`**`PluginInterface`**

The interface that each plugin should extends to support their file format and tool.

Each plugin should be able to detect if it can be run or not automatically. In case the plugin cannot be used, it should not prevent the other plugins from running.

Each plugin should be able to detect in a folder (which may contain subfolders) if there are files they are compatible with.

For some plugin, multiple analyzes can be submitted at once and the plugin needs to know which one should be analyzed. This will be the role of the `session` argument.

classmethod `convert_inputfiles` (*folder=None*, *inputfile=None*, *session=None*, *lod_threshold=None*, *qtls_file='qtls.csv'*, *matrix_file='qtls_matrix.csv'*, *map_file='map.csv'*)

Convert the input files present in the given folder or inputfile. This method creates the matrix representation of the QTLs results providing for each marker position the LOD value found for each trait as well as a list of all the significant QTLs found in the results and a representation of the genetic map used in the experiment. The genetic map should be cleared of any markers added by the QTL mapping software.

Parameters

- **folder** – string of the path to the folder containing the files to check. This folder may contain sub-folders.
- **inputfile** – string of the path to the input file to use
- **session** – the session identifier used to identify which session to process
- **lod_threshold** – the LOD threshold to apply to determine if a QTL is significant or not
- **qtls_file** – a csv file containing the list of all the significant QTLs found in the analysis. The matrix is of type: `trait, linkage group, position, Marker, LOD other columns`

- **matrix_file** – a csv file containing a matrix representation of the QTL data. This matrix is of type: `marker, linkage group, position, trait1 lod, trait2, lod`
- **map_file** – a csv file containing the genetic map used in this experiment. The map is of structure: `marker, linkage group, position`

classmethod **`get_files`** (*folder*)

Retrieve the list of files the plugin can work on. Find this list based on the files name, files extension or even actually by reading in the file.

The method `get_files` will browse the provided path for any file in the specified folder or sub-folder that the plugin can handle.

Note: `get_files` should be able to handle the case where the provided path points to a file rather than a folder, in which case the plugin should return an empty list.

Parameters **folder** – string of the path to the folder containing the files to check. This folder may contain sub-folders.

classmethod **`get_session_identifiers`** (*folder=None*, *inputfile=None*)

Retrieve the list of session identifiers contained in the data on the folder or the inputfile.

The method `get_session_identifiers` is used by the web-interface to present to the user a list of sessions they can choose from and by the command line interface, when the user did not specified a session or specified an invalid session.

The `get_session_identifiers` method receive either a folder or an inputfile argument (and should raise and `MQ2Exception` if both are provided). The method extract the session identifiers from this input and return them as a list. If both `folder` and `inputfile` are `None`, it may return an empty list.

Parameters

- **folder** – string of the path to the folder containing the files to check. This folder may contain sub-folders.
- **inputfile** – string of the path to the input file to use

classmethod **`is_applicable`** ()

Functions used to check whether the plugin can be used or not. This is the function that would check the import and that should make sure the rest of the plugin will run smoothly.

The method `is_applicable` will be called by the program and should return a simple boolean telling if the plugin can be run or not. It is this method that should check that all the potential dependencies are met for the plugin to run.

classmethod **`valid_file`** (*filename*)

Check if the provided file is a valid file for this plugin.

Since MQ² can also be used on a single file via the command-line, the `valid_file` will then be used to check if the provided file can be handled by the plugin.

Note: `valid_file` should be able to handle the case where the provided path points to a director rather than a file, in which case the plugin should return a *False* boolean.

Parameters **filename** – string of the path to the file to check.

INDEX