Purpose of scaffhunter

Scaffhunter regroups several programs which principal aims are to work and visualize genetic mapping data. In addition to 2D visualization of linkage between markers, these tools can be used to order scaffolds without passing by the tedious step of genetic map construction and reconciliation between marker order in scaffold and marker order in the genetic map.

Installation

All proposed tools described here are written in python and work on linux system
To install the tools:
1- unzip the tar.gz file with the following command line : tar -xvzf <archive-file>
2- open the loca_programs.conf file
3- set the path to each programs required

To run fully all programs are needed:
1- blast
2- bowtie2 can be found at http://bowtie-bio.sourceforge.net/bowtie2/index.shtml
3- bwa can be found at http://bio-bwa.sourceforge.net
4- orthodotter is provided in the bin directory with these tools
perl, python and java are required. Biopython is also required.

Two of these tools use **orthodotter** program provided in this archive. This program has been developed by Jean-Marc Aury (Institut de Genomique / DSV / CEA, FRANCE) and is deposited under CeCILL license http://www.cecill.info.

How to cite

Please cite : Martin G, Baurens F-C, Droc G, Rouard M, Cenci A, Kilian A, Hastie A, Doležel J, Aury J-M, Alberti A, et al. 2016. Improvement of the banana “Musa acuminata” reference sequence using NGS data and semi-automated bioinformatics methods. BMC Genomics 17:1–12.

Scaffremodler descriptions

The package provided comprise 10 programs listed here:

GBS_corrector.py
JMpwd2matrix.py
locOnRef.py
mat2submat.py
matrix2ortho.py
pwd2figure.py
pwd2matrix.py
reorderient.py
scaff2chrom.py
UPGMA.py

All 10 programs run using the following command: python program-name <--options-name value>

Tools descriptions

JMpwd2matrix.py
This program takes a pairwise file generated by JoinMap (.pwd file in the folder created by joinmap) and returns two matrixes:

- a matrix containing pairwise LOD score
- a matrix containing pairwise recombination frequencies

Options:
--pwd: The pairwise file calculated by joinmap
--lod: The LOD matrix output file name. (Default: lod_matrix.tab)
--rec: The REC matrix output file name. (Default: rec_matrix.tab)

pwd2matrix.py
This program takes a file containing pairwise statistics and returns a matrix containing the same information. Missing data in the matrix will be filled with 999999999 value.

Options:
--pwd: Pairwise file. Column 1: id1, column 2: id2, column 3: statistics
--out: Output file name

locOnRef.py
This programs locates fasta sequences onto a reference sequence. This program take in input two multifasta files: a file containing short sequence (markers file --fasta) to align against the other fasta file (reference sequence file --ref).
Several tools can be used together. This program can use bowtie2 (local and end to end modes) with --very-sensitive mode, bwa mem and blast. Only unique hits are conserved for each tools used. Only identical hits between all tools used are conserved. Identical hits between tools are identified if all hits from different tools co-locate in the same region in a region of 2 fold --margin option. Otherwise, they are discarded. If a tool, does not report a hit for a sequence (due to multi hit or no hit), this tools is not used for this sequence.

If blast is selected, this program perform for each sequence a blast (e-value = 1e-10, -F F).
Blast output is used to reconstruct the alignment from different HSP and calculate an identity of the hit (This step is performed by blat_results_analyzer_v3.pl, script written by Philippe FRANCOIS and modified by Guillaume MARTIN). For each identity percentage threshold (98%, 95%, 90%, 80 and 80% identity) valid hits are counted and hit is reported only if it is unique for at least one of these identity thresholds.

Options:
--ref: The reference fasta file
--blast: Use blast:
y: use blast

n: do not use blast (Default)

--bwa_mem: Use bwa_mem

y: use bwa_mem

n: do not use bwa_mem (Default)

--bow: Use bowtie2

y: use bowtie2

n: do not use bowtie2 (Default)

--bow_loc: Use bowtie2 with local mode

y: use bowtie2 with local mode

n: do not use bowtie2 with local mode (Default)

--fasta: The fasta file containing markers

--out: Output file name

--margin: The margin to consider hit as similar (integer). (Default: 2500)

--index: Build reference index:

y: build index (default)

n: do not build index

--rmindex: Remove reference index at the end of calculation

y: remove index

n: do not remove index (default)

--thread: The thread number used for mapping (integer). For --blast no more than 5 is needed, for --bow 1, for --bow_loc 1, for --bwa_mem 1 (Default: 1)

UPGMA.py

This program takes in input a matrix containing maker linkage or divergence and a tabulated file locating these markers on scaffolds (or sequence) and try to group and order them based on an UPGMA like approach.

This program work as followed:

1- an mean linkage/divergence is calculated between scaffolds

2- scaffolds are then grouped using an UPGMA like approach

3- scaffolds are orientated and positioned (at the beginning or the end of a precedent group) trying to optimize a score calculated for each position and orientation.

- In case of maker linkage information (LOD score) in the matrix, the score is calculated as followed:

$$\text{score} = \sum_{i=1, j=1, x_i<x_j}^{n} \left(1 - \frac{(x_j - x_i)}{\text{n}}\right) \text{LOD}_{\text{ij}}$$

- In case of maker divergence information (recombination rate) in the matrix the score is calculated as followed:

$$\text{score} = \sum_{i=1, j=1, x_i<x_j}^{n} \left(\frac{(x_j - x_i)}{\text{n}}\right) \text{LOD}_{\text{ij}}$$

with n the number of markers in the LG to order, x_i and x_j are the position of markers i and j in the tested order, and LOD_{ij} the LOD score between markers i and j. To optimize computation time and as order is not tested within scaffolds, i and j are markers from different scaffolds.

This program output a file containing ordered scaffolds in a table format and a table file containing ordered markers relative to the scaffold order.

Options:
--scaff: A table file containing in column 1: markers name, column 2: scaffold name. Columns should be ordered first on scaffold name and second on marker position on scaffold
--mat: Matrix file with pairwise statistics between markers.
--out1: Output file name of ordered scaffold. (Default: UPGMA_ordered_scaff.txt)
--out2: Output file name of ordered markers. (Default: UPGMA_ordered_mark.txt)
--type: Data type: IDENT (marker linkage) or DIF (recombination rate). (Default: IDENT)

reorderient.py
This program take in input a matrix containing maker linkage or divergence and the two output files of UPGMA.py and try to optimize the scaffold order and orientation proposed by UPGMA.py.
The optimization is performed by calculating a score for the scaffold order, trying rearrangements (scaffold or scaffold group permutations and/or inversion) followed by score re-calculation. If the new score is better than the previous, the new order is conserved; else, the previous order is conserved. The program stops when more than a defined number of consecutive rearrangement (passed in --iter argument) do not improve the ordering.

The score value calculation is different depending on the statistics found in the matrix:

- In case of maker linkage information (LOD score) in the matrix, the score is calculated as followed:

$$\text{score} = \sum_{i=1, j=1,\, x_i<x_j}^{n} \left(1 - \frac{(x_j - x_i)}{\text{n}}\right) \text{LOD}_{\text{ij}}$$

- In case of maker divergence information (recombination rate) in the matrix the score is calculated as followed:

$$\text{score} = \sum_{i=1, j=1,\, x_i<x_j}^{n} \left(\frac{(x_j - x_i)}{\text{n}}\right) \text{LOD}_{\text{ij}}$$

with n the number of markers in the LG to order, x_i and x_j are the position of markers i and j in the tested order, and LOD_{ij} the LOD score between markers i and j. To optimize computation time and as order is not tested within scaffolds, i and j are markers from different scaffolds.

This program output a file containing ordered scaffolds in a table format and a table file containing ordered markers relative to the scaffold order.

Options:
--mat: Matrix file with pairwise statistics between markers.
--scaff: The output file passed in --out1 argument of UPGMA.py. This file contains ordered and orientated scaffolds ate the end of UPGMA.py program.
--mark: A table file containing in column 1: markers name, column2: scaffold name. This is the file generated by UPGMA.py in --out2 argument
--iter: Number of consecutive rearrangement tried without improvement to stop the improvement of the ordering and orientation (integer or auto).

If “auto” is passed to the file, this number is estimated to the total number of possible iterations in case of one scaffold repositioned: $(n-1)^2 * 2 + n$ with n=scaffold number.
--type: Data type: IDENT (marker linkage) or DIF (recombination rate). (Default: IDENT)
--out1: Output file name of ordered scaffold. (Default: UPGMA_ordered_opt_scaf.txt)
--out2: Output file name of ordered markers. (Default: UPGMA_ordered_opt_mark.txt)

matrix2ortho.py
This program takes a matrix containing pairwise statistics between markers and a file containing markers order and plot pairwise marker statistics in a dot-plot like picture.

Options:
--mat: Matrix file containing the pairwise information.
--order: Table file containing marker ordered. column 1 : marker name, column 2 : identifier (ex: chr id). Markers should be grouped by identifier and ordered based on their position in the group. The file can contain additional columns (example position on identifier) that will not be used.
--type: Type of statistics in the matrix: LOD (odd ratio), REC (recombination), COR (correlation). (Default: LOD)
--png: Output file name. (Default: dot-plot.png)

scaff2chrom.py
This program takes in input files a multifasta containing scaffolds and a table file containing scaffolds order calculated by reorderient.py and output a multifasta file containing reconstructed chromosomes and an agp file locating scaffolds into chromosomes.

Options:
--table: Table file containing in column 1: chromosome name, column 2: scaffold name, column 3: expected orientation (FWD or REV).
--seq: Multifasta sequence file containing scaffolds.
--unknown: Build an unknown chromosome with the remaining sequences. (Default: yes)
--out: Fasta output file name.
--agp: agp output file name.

pwd2figure.py
This program take a pairwise file generated by JoinMap and return a dot-plot representing pairwise marker linkage of markers along chromosomes. Two additional file are generated that can be used by Darwin software to construct tree. These files contain pairwise recombination rate converted into mapping distance using the Kosambi mapping function

Options:

--pwd: Pairwise file calculated by Joinmap.
--type: Type of pairwise statistics: LOD or REC.

--order: Tabulated file containing in column 1: markers names and column 2: chromosome name. Markers should be grouped by identifier and ordered based on their position in the group. The file can contain additional columns (example position on identifier) that will not be used.
--png: Output name for the dot-plot (png file).
--dis: Output name for the .dis file (for Darwin).
--don: Output name for the .don file (for Darwin).

mat2submat.py
This program takes a square matrix (--mat) file and create de sub-matrix containing all IDs provided in another file (--mark).

Options:
--mark: A tabulated file containing in column 1 the IDs that will be contained in the sub-matrix
--mat: Square matrix file.
--out: Output file name. (Default: sub_matrix.txt)

GBS_corrector.py
This program identifies markers genotyping errors recorded in a table file based on their order (obtained from genetic map or reference sequence) provided in a table file. This program is based on the principle that no more than one recombination event can occur in a window of x around the observed marker (x is passed in --fen argument). If more than one recombination event is observed, a genotyping error is identified. This program output three files:

1) A file containing corrected markers (--nosu argument)
2) A file containing corrected markers where redundancy is removed (--nr argument)
3) A file where markers with more than X errors are identified are removed. (--Nosu argument)

This program works on phased markers of same type. It is based on the principle that no more than one recombination can be observed in a window around each markers.

Options:
--table: The table file containing phased data. col 1 : markers name, columns 2, 3, 4 necessary but not used by the program, columns 5 to end : individual genotypes. First line contain header. Redundant names (markers and individuals) are not allowed.
--fen: Minimal marker number in a linkage group/scaffold/reference sequence to apply correction followed by half window size. Both values should be separated by "=". (Default:10=3)
Example: if --fen 5=1, only scaffold/chromosome/linkage group containing at least 5 will be treated and if a marker contain more than one recombination in a window of [marker -1; marker+1], an error is detected.
If different classes, couple can be repeated and separated by "=".
Example: 10=3=5=1, in this case if a group contain at least 10 markers, a window size of ±3 will be applied and in groups comprising between 5 and 9 markers, a window size of ±1 will be applied.
--order: A table file with markers names in column 1 and scaffold they match with in column2. Markers should be ordered first on linkage group/scaffold/reference sequence and second on markers position on linkage group/scaffold/reference sequence

--suspect: The minimal number of correction in a marker to consider this marker as supect. (Default: 10)
--nr: The name of the corrected non redundant output file. (Default: corrected_non_redunddant_mark.tab)
--cor: The name of the corrected output file. (Default: corrected_mark.tab)
--Nosu: The name of the output file containing the non-suspect markers. (Default: filtered_mark.tab)