Synteny Analysis Pipeline

Overview

The pipeline uses the genomic positions of orthologous proteins to fit monotonic polynomial curves describing the synteny relationship between two genomes, which can then be used to estimate the position of an unplaced gene from the position of its ortholog in the other genome. Orthologs are identified using a simple blastp reciprocal best evalue approach. Curve fitting is semi-automated utilising a dot plot showing ortholog positions in the two genomes on the x and y axes: first regions containing a below average density of orthologs are automatically filtered out, then some manual pruning of orthologs is allowed, then individual regions with clear synteny are manual selected by clicking on the first and last gene of the region, finally polynomials are fitted automatically to each region. The polynomial coefficients can then be used to calculate the predicted position of unplaced genes provided a placed ortholog can be identified in the other genome within a region covered by a fitted curve.

Scripts are numbered in the order they need to be run, with the exception that there are alternative versions for some steps where only one needs to be run. This pipeline was created using diploid oat as one genome, and another grass genome as the other (either barley, brachypodium, rice or sorghum). In the scripts spp is usually the variable refering to the other genome. The scripts are not a well polished, reusable analysis pipeline, but are made available to document how the oat genome was analysed and in the hope they may prove useful as a basis for making your own pipeline. You are likely to need to know some python and R to make use of the pipeline. For further information read the comments in the script files. Do not run any script before you understand which files will be deleted and overwritten! Most editable parameters appear near the top of the scripts. Some scripts require my python utilities library 'rjv', also available on github.

(1) BLASTP

Scripts 010_blastp_fwd.sh and 020_blastp_rev.sh are simply Sun Grid Engine wrapper scripts for blastp. Choose an appropriate evalue (I used 1e-30) and output using the -outfmt 6 flag (or equivalent if using legacy blast) to output tab separated blast hits without header information. If doing a parallelised search, simply concatenate all the output files into one before proceeding to step (2). The fwd and rev scripts are the same except that the species acting as the query and subject are reversed.

Requirements: some version of blast, one FASTA for each of the two species containing their proteins, blast database of both FASTAs.

To run the scripts edit the filenames and SUN Grid Engine settings, or simply run the blasts using your own scripts or system and generate the two output tab separated value files for the forward and the reverse blasts.

(2) Find Reciprocal Best Hits

Edit the script 030_find_rbh2.py and set the names of the two files containing the forward and reverse blastp hits, and the name of the output file you want to contain the reciprocal best hits.

inpfwd = [forward blastp hits]
inprev = [reverse blastp hits]
out = [output file]

Leave maxhit equal to 1. This script is pure python and should not require any external dependencies or use a large amount of resources. Run the script and produce the output file. The output file should contain tab separated data with a gene id from the first genome, second genome and the evalue of the forward hit on each line.

(3) Filter out low density ortholog hits

Script 050_sliding_window.py scans along one of the two genomes and removes ortholog hits from regions with a below average number of orthologs. Scripts 055_sliding_window_oat.py does the same thing for the second genome. The script refers to one genome as 'oat' and the other as 'spp'. Edit the script and set the names of the input and output files. Required inputs are the reciprocal best hits file 'hits' and two files for each genome giving the position of each gene ('oatpos' / 'spppos') and the lengths of the chromosomes / linkage groups ('oatsize'/'sppsize'). The distances can be in any units, and can be floating point values, but the same units must be used in the size and pos file for the same genome. (Script 040_generate_gene_positions.py is part of the system I used to create the position information from gff files, but is not a complete solution.) Script 050 outputs two files: 'out', the filtered ortholog list and 'outplot', coordinates for a dot plot showing ortholog positions *before* filtering. Script 055 needs 'out' from 050 as its input file 'hits' and creates a new file 'out' now with the orthologs filtered for low density in both genomes.

The format of the 'pos' files in a comma separated value file containing three columns: geneid, chromosome/linkage group (1-based integer), position (integer or floating point). There is no header.

The format of the 'size' file is a single line of comma separated values listing the sizes of all chromosomes in numerical order (integer or floating point).

The window size used to measure the average number of orthologs is the total genome size divided by the number of bins. To first calculate the average number of genes per bin the genome is sampled at 'samples' regularly placed locations per *chromosome*. This will work fine for genomes with roughly equally sized chromosomes, but the script will need to be modified if sizes vary considerably.

(4) Manual pruning step one

Edit script 060_make_synteny_plot_data_filtered.py and set the names of the input and output files. 'hits' is the name of the 'out' file from script 055 and 'out' is the name of the output file that will be created.

070_prune_points_stage1.R is the first manual pruning stage, allowing to delete all orthologs matching to a given chromosome-chromosome pair. The dot plot is displayed with lines separating each chromosome, so the plot appears as a grid of squares. To manually delete all orthologs click on any dot within a square and all the dots will be deleted. The script is currently set up to use a variable 'spp' to simplify comparisons between oat and multiple other genomes, so you may have to edit the script a little. Requires R and the R library plotrix.

Edit the script to set the correct input and output file names:
coords=[file output from script 060]
sizes=[first genome chromosome/linkage groups sizes]
chrm_sizes=[second genome chromosome/linkage group sizes]

outfile=[file to output filtered hits to]

On linux the script can be run as follows: from the command line launch R, then run using source('070_prune...'). Now left clicking on any dot within a square will delete all donts within that square. Right click when you are done. To run with the plot window maximised, maximise the window then right click on any dot to cancel, then rerun without closing the plot window. If you accidentally delete the wrong orthologs, use a right click to quit, then rerun the script again by typing the source command again. These details may vary if you are using R on a different platform. All the R scripts from the pipeline should generally be run in the same R session without closing it.

(5) Manual pruning step two

Script 080_prune_points_stage2.R is very similar to script 070, but allows pruning of individual points rather than whole squares. Edit the script to set the correct input and output filenames. The inpfile should be the name of the outfile from script 070. Run the script from within R using the source('080_prun...') command as before. Left click a point to delete it, right click to finish. If you delete something accidentally, start again by right clicking then run the script again immediately.

Script 085_prune_points_stage2_zoomed.R is the same as 080 except that it displays each square individually to allow the plot to be zoomed-in more.

(6) Marking syntenic region end points

Edit script 090_mark_synteny_end_points.R to set the correct input and output file names. inpfile should be the outfile from script 080/085. This script allows the selection of the first and last gene in a syntenic region (in terms of x axis position only). The script waits for pairs of left clicks, the first click marks the gene at one end of the region and the second marks the one at the other end (the order should not matter). Right click when done. Script 095 is the zoomed version of script 090.

(7) Fit monotonic third degree polynomials

Edit the file names in script 100_dump_selected_end_points.R then run using the R 'source' command. Then also edit and run the filenames in script 110_fit_curves_monotonic.R. This script requires R packages MASS, MonoPoly and plotrix. 'outfile' will receive the polynomial coefficients and 'outplot' will receive a PNG image showing the fitted curves. 'endpts' receives a list of the syntenic regions.

See script 120_assign_gene_positions_polynomial.py for an example of how to calculate the location of an unplaced gene from the position of a placed ortholog in the other genome. Line 164 shows the polynomial formula (new_pos = c[0] + c[1]*x + c[2]*x**2 + c[3]*x**3) where x is the chromosomal position of the placed gene and c[0] to c[3] are the four coefficients. The conditionals on lines 136 and 137 (if spplg != reg[0][4]: continue; if spppos < reg[0][5] or spppos > reg[1][5]: continue) have first been used to determine which syntenic region the placed gene falls within so that the correct set of coefficients can be used.