Identify Blueberry to *Arabidopsis* Syntelogs and Homologs

Scott J. Teresi *Edger Lab - Michigan State University*

This document contains my workflow, scripts and notes on generating *Vaccinium corymbosum* to *Arabidopsis thaliana* homologs

---

Approach:

To identify homologs, I am going to first identify orthologous genes using synteny. I will then follow up by using BLAST to identify any homologs that we would've missed using the synteny-based approach (orthologous genes that may have translocated elsewhere, thus breaking synteny).

Rough Outline

1. Use **SynMap** on CoGe to compare syntenic blocks between *Arabidopsis thaliana* and *Vaccinium corymbosum*.
2. Supplement results with BLAST search to catch any non-syntenic homologs (single-gene transpositions).
3. Clean up ortholog file, parse out gene-pairs that do not match our significance threshold. Finalize ortholog output.

Data Input and Genome Versions:

This section catalogs the versions of the genomes I used for this analysis and contains the methods I used to generate, missing/supplementary files.

Genome Versions Used:

Regular CoGE ID	Masked ID
Arabidopsis thaliana Col-0 (id1)	CNS PL2.0 Masked Repeats 50X (v10, id 16746)
Vaccinium corymbosum (id39928)	mask w/ RepeatMasker (v3, id 58746)

Running SynMap

This section describes the methods to run SynMap on CoGe. I will describe some of the options I have used here and why (in the following sections), but it would benefit the reader to go over SynMap's documentation and read more about E-values.

SynMap Analysis Options:

We are going to run SynMap with default options. Here is a link to the documentation. The default options at the time of writing are:

- DAGChainer
  - Relative gene order
  - Maximum distance between two matches: *20* genes
  - Minimum number of aligned pairs: *5* genes
- Merge Syntenic Blocks
  - Algorithm: Quota Align Merge
  - Maximum distance between two blocks: *4:1*
- Syntenic Depth
  - Algorithm ?????
- Fractionation Bias
  - Test???
- CodeML
- Advanced Options

A Word on E-Values:

Briefly, E-values, which stands for expectation value, is a correction of the p-value for multiple testing (we are multiple testing when we search each gene for a match in the other genome, and by chance we could generate a significant p-value, so we must enforce a mathematical correction). In the context of database searches the E-value is the number of distinct alignments with a score equivalent to or better than S[1], that are expected to occur in the database search by chance.

SynMap Output:

It outputs a tab-separated text file. I would encourage the reader to check out the summary of the output format from this link under *Results* and *DAGChainer Output*. Here is the link to regenerate the analysis on CoGe.

[1]S: A score is the numeral value that describes the overall waulity of an alignment, higher numbers means higher similarity.

Using BLAST to Supplement the Syntelog Search:

We are doing this step to identify homologs that may have been missed using a synteny-based approach. Genes that could have been missed by the synteny search include single-gene transpositions (and others). We are going to use a BLAST database of protein predictions.

Setting up a BLAST Database:

Please refer to the script at `BLAST_Orthologs/blastall.sb` for more information. First we generate a BLAST database and prepare it for protein indices. Then we can run the BLAST algorithm on the two sequence files, this may take awhile. For notes on the options for `blastall`, please refer to the documentation.

Filtering and Combining our Data:

We have two data files, one from Synmap (synteny) and one from BLAST (homology). Now we must filter both outputs and merge the files. I orchestrate all of the running of the code from `generate_pairs.py` but compartmentalize each function/class object as necessary into their own files. Please refer to `generate_pairs.py` for the general control flow of the project from this point on out.

Set up Python Environment:

Please refer to the project README.md to ensure you have the correct Python packages. It may help to create a virtual environment.

Filter the SynMap Output File:

The output file has a lot of extraneous information. We are going to distill it down into a 2-column tab-separated values (tsv) file. To do this we will use the Pandas module in Python to maniuplate the dataframe. If you have an aversion to Python you can always accomplish this task in R. Here I accomplish this with the use of the `import_syntelogs.py` file.

Filter the BLAST Output File:

This output file also has a lot of extraneous information for our purposes. We distill this file down as well. Please examine `import_homologs.py` for this portion.

Merging Syntelogs and Homologs:

Here I use a Python program to merge the files, making sure that I don't put the same gene twice, as the BLAST search could turn up some of the same gene pairs we found in the SynMap search. We defer to taking the SynMap result (Arabidopsis - Blueberry gene-pair) over the BLAST pair result. Please examine `merge_homo_synt.py` for this portion.

Merging Genes From Expression Analyses with the Merged Data:

Earlier in the year I worked on calculating differential expression data from galling wasp treatments on two blueberry cultivars (Draper and Liberty, shortened to DRA and LIB in the data files). There were several timepoints and a treatments, for each comparison group I create a file in the output data folder.

We want to examine our merged data file, and take the subset of Arabidopsis-Blueberry gene pairs that match with the blueberry gene given from the differential expression data set. We perform this subsetting in two ways. First we take the Arabidopsis gene that belongs to every Arabidopsis-Blueberry pair with a matching differential expression blueberry gene. However some of these differential expression blueberry genes may match with the same Arabidopsis gene, so we run the analysis again separately and remove any matches in which the Arabidopsis would be repeated twice.

Interpretation of Data:

Use StringDB