Using Random Forest for Sexually Antagonistic Selection Detection

Sam Su
University of Texas at Austin
CNS, Department of Integrative Biology
sam.su@utexas.edu

1 Introduction

Sexually Antagonistic Selection (SAS) emerges when natural selection is biased towards one sex, where an allele (and its corresponding expressed trait or combination of traits) beneficial to one sex may be harmful to the other. Here, we are interested in detecting signals of SAS in the recombining pseudo-autosomal regions (PAR, combining) of sex chromosomes, which is known to generate more conspicuous patterns of variation (peaks of F_{ST}) compared to the sex determining regions (SDR, non-combining) of sex chromosomes or the autosomes. Our data comes from Japan Sea sticklebacks (JSP, Gasterosteus nipponicus), which has a expanded PAR suitable for identifying targets of SAS. According to Andrius' manuscript, we found four potential SAS sites with high peaks of divergence (F_{ST}) between the X and Y chromosomes of JSP. However, random genetic drift (stochastic changes in allele frequencies as a result from population size limits) can cause larger differences than Fst than expected. Hence, the employment of machine learning was considered to exclude that possibility.

2 Training Data

We imitated the evolution of genetic sequences on the X and Y chromosomes across 14 samples of JSP using coalescent simulations. The simulation takes four major parameters: the overall recombination rate, the frequency of SA allele on X and Y chromosomes, and then the SAS window's location (in ρ, 10kb) with respect to the SDR. Please see the README for this repository for must specifics on input formats. By tuning around the SA allele frequencies, we can generate labeled samples with and without SAS for use in building a classification model, effectively framing this as a supervised learning problem.

Around 5000 samples with high strength (high difference in SA allele frequency between X and Y) SAS and without SAS (both SA allele frequencies are 0) were generated for training/cross validation, totaling 10000 training samples. 500 samples with high strength SAS and without SAS (1000 total) were generated for testing purposes (not used in model building), totaling 1000 test samples. The location of the SAS window is chosen randomly from the range 75ρ to 400ρ to allow space far enough away from the SDR to avoid effects of linkage evolution. The results from the simulation are then processed in several ways to obtain the final set of attributes that we would feed into a predictive model to detect signals of SAS. Please see the methods section for specifics on what we tried.

3 Methods

Overall, we think random forest is a great model that harvests benefits both from descriptive modeling (explaining the data) and predictive modeling. Specifically, not only do we want to effectively identify the presence of SAS but also understand why, by looking at attributes that are critical (most often used by RF) when making that decision (see results). In addition, trees are computationally inexpensive, even for large training sets, not to say classification for new records are to simply 'walk' down the tree's decision rules. Decision trees alone to robust to presence of noise (especially after pruning), and random forest further enhances this fact by aggregating multiple trees. Trees are also indifferent to redundant/irrelevant/correlated features, since if all attributes produces similar gain, than a feature will be picked arbitrarily without bias (though redundancy can affect tree depth). There's a slight issue of data fragmentation, where at some leaves, the number of records may be too small to make a statistically significant decision about class representation. However, this is a hyperparameter can that be tuned during training. We used Scitkit-Learn's implementation of random forest, such that the model is trained (with hyperparaemeter tuning) and evaluated through their provided cross-validation pipeline.

The coalescent simulation produces the simulated sequences for 14 X and 14 Y JSP chromosomes, where they are been fed through a wrapper that summarizes the data. There are three major ways we tried to process the coalescent simulation results before feeding them into Random Forest.

3.1 Summary Statistics per Window

Here, the simulation results are sliced into 460 ρ sized window, where we computed summary statistics for each window across the 14 X and 14 Y chromosomes. Obviously, adjacent windows tend to be correlated both in terms of genetic sequences and thus the computed summary statistics. The statistics we tried are: genetic diversity on the X and Y, total genetic diversity (π_{total}),

F_{ST} between the X and Y, D_{xy} between the X and Y, D_a between the X and Y, Tajima's D on the X, Y and all samples, relative density of SNPs on the X, Y and across all samples, and average correlation between SNPs on the X, Y, and across all samples (15 total). Input is flattened to a two dimension table to before random forest fitting.

3.2 Reduced Summary Statistics

Using the aforementioned summary statistics, we also tried fitting random forest using the average of these metrics across all windows, thus eliminating distance from the PAR as a factor influencing our input data but also eliminates a chance for the model to extract subtle differences across windows. We also thought to reduce the dimensions of our input data by using Principal Component Analysis (PCA) to instead extract the top 3 principal components that captures around 85% of the variation. We tried to apply PCA both as a filtering procedure (before any training occurs) and as a pipeline procedure (during training to decide the optimal number of PCs to keep).

3.3 Binned F_{ST} Peaks

An alternate method we tried to eliminate the complexity of the input data was to solely focus on F_{ST} and attempt to detect potential peaks of F_{ST} within each window. We accomplish this by comparing the distribution of F_{ST} in a window to a kernel function, where we compute a mean-squared-error (MSE) that denote the goodness of fit. The kernel function is scaled by the window's location from the PAR, so that we can have a reasonable expectation of the size of a peak, if any. Hence, our input data now consist of the maximum F_{ST} in a window, as well as the MSE for the highest F_{ST} in that window. Then, to extract away potential confounding effects of window location, we tried to bin these two metrics with various bin-widths in hope of improving the model performance.

4 Discussion

Ultimately, invariant of the approach, there's simply too much stochastic noise from our simulation results that overcrowded any SAS pattern, where our cross-validated accuracy would be around 0.5 (no better than guessing). This was somewhat surprising, given all training examples were generated with high strength of SAS. This is likely an indication of inadequate post-processing steps unrelated to RF, especially after we have accounted for the effect of window locations with respect to the PAR. Nevertheless, there remains a few options left to try; we could try to run RF using a weighted average of summary statistics across all windows based on their location, try more advanced dimensionality reduction methods (t-SNE), or attempt more sophisticated classification models (ie. deep learning).