SNPbinner

SNPbinner is a Python 2.7 package and command line utility for the generation of genotype binmaps based on SNP data across lines. The program consists of three parts: `crosspoints` , `bins` , and `visualize` .

Installation

The only non-standard dependency of SNPbinner is Pillow, a PIL fork.

To install the SNPbinner utility, download or clone the repository and run `$pip install repo_folder/` . **Alternitively, one can call the program with `$python2.7 repo_folder/snpbinner` instead of `$snpbinner` which does not require installation.**

Crosspoints

`crosspoints` uses Hidden Markov Models to identify the most likely cross-over locations on a chromosome given genotyped SNPs. One must also provide a minimum distance between crosspoints. The script also uses a greedy algorithm which “eats in” from the sides of groupings of too-short state regions in order to find reasonable crosspoints. That algorithm follows the steps below.

1. If a contiguous group of too-short regions is long enough to be its own acceptably-long region, it will be treated as such and assigned the most likely genotype using the 3-state HMM.
2. If the first or last too-short region is neighboring to an acceptably-long region of the same genotype, it can be considered part of that region, and removed from the group.
3. If the first or last too-short region is neighboring an acceptably-long heterogenous region, it will be assigned the heterogenous genotype and removed from the group as per step 2.
4. If neither the first or last too-short region is neighboring a heterogenous or same-genotype region, the shortest of those two regions will be assigned to the same genotype as the acceptably-long region neighboring it and then removed as per step 2.
5. Repeat from step 1 until each group is empty.

To Run:

```
$snpbinner crosspoints -i INPUT_FILE (-m MIN_LENGTH | -r MIN_RATIO) -o OUTPUT_FILE [OPTIONAL ARGUEMENTS]
```

`INPUT_FILE` : Path to a SNP TSV of the format described below.
`MIN_LENGTH` : Minimum distance between crosspoints in basepairs.
`MIN_RATIO` : Minimum distance between crosspoints in as a ratio. (0.01 would be 1% of the chromosome)
`OUTPUT_FILE` : Output filename.

Optional Arguments:
`-p PREDICTED_HOMOGENEITY` ideal homogenous percentage of homogenous regions, used to calculate emmision probability (default = 0.9)
`-c PREDICTED_CROSSOVERS` used to calculate transition probability (default = 4)
`-l CHROM_LENGTH` The length of the chromosome/scaffold which the SNPs are on. If no length is provided, the last SNP is considered to be the last site on the chromosome.

Bins

`bins` takes a CSV generated using `crosspoints` and a minimum bin size. It identifies a list of bins and their genotypes for each RIL. In order to do so, the crosspoints from all RILs are projected onto one representative chromosome. These crosspoints are then partitioned into groups where the distance between each consectutive crosspoint is less than the minimum bin size. Groups containing less than three crosspoints are combined into one representitive crosspoint located at the centroid of the group. If the group contains three or more crosspoints, the maximum number of breakpoints that could fit inside the region bounded by the first and last crosspoint in each list is first determined by dividing the region size by the minimum bin size (rounded up). Then, clustering is performed for all values of K from the maximum number of breakpoints to 1 using a modified 1D K-means algorithm which adjusts the centroid values such that they are greater than a minimum distance from eachother during the update step. Each of the produced clusterings are then scored using the average deviation for each cluster to the adjusted centroid. The adjusted centroids from the clustering with the lowest average deviation are then considered to be the representitive crosspoints for the group of crosspoints. The representative crosspoints determined for each group are then used as the bounds of the bins created. To genotype the bins for each RIL, the genotype which covers the most area inside of the bin fromthe original crosspoint data is used. The bin locations, bounds, and genotypes for each RIL are then output in CSV format.

To Run:

```
$snpbinner bins -i INPUT_FILE -l MIN_BIN_SIZE -o OUTPUT_FILE [OPTIONAL ARGUEMENTS]
```

`INPUT_FILE` : Path to a `crosspoints` output CSV file.
`MIN_BIN_SIZE` : Minimum size of a bin in the resulting binmap.
`OUTPUT_FILE` : Output filename.

Optional Arguments:
`-n BINMAP_ID` If provided, an extra header row will be added and each column labeled with the given ID string.

Visualize

`visualize` graphs the input and results of the `bins` and `crosspoints`. It can accept three filetypes (SNP input TSV, crosspoint script output CSV, and bin script output CSV). It then parses the files and groups the data by RIL, creating an image for each. In each colored row of the resulting images, regions are colored red, green, or blue, for genotype a, heterozygous, or genotype b, respecively. The binamp is represented in gray with adjacent bins alternating dark and light. The script can accept any combination or number of files for each of the different types.

To Run:

```
$snpbinner visualize -o OUT_FOLDER [OPTIONAL ARGUEMENTS]
```

`OUTPUT_FOLDER` : Folder where the resulting images will be saved. The folder must exist.

Optional Arguments:
`-s SNP_TSV` : Add SNPs from a SNP TSV to the images. This arguement can be repeated for multiple SNP files.
`-c CROSSPOINT_CSV` : Add genotype regions from a crosspoint CSV to the images. This arguement can be repeated for multiple crosspoint files. `-b BIN_CSV` : Add binned genotype regions and a binmap from a bin CSV to the images. This arguement can be repeated for multiple bin files.