1 Introduction

This documentation describes a set of functions for analyzing sequence variation within repetitive sequence using ConTExt output. Most of these tools revolve around two data structures: 1) a 3D numpy array containing SNP pileups for every sample in the analysis (generated by ConTExt) and a 2) the SequenceSummary class, which generates and stores various summaries of the SNP pileup.

2 SequenceSummary Object

2.1 Initializing

A SequenceSummary object is initialized by passing the path of a SNP pileup array as follows:

summary=SequenceSummary(filepath, ignore= ignore ,sample_file=", pop_rule=1 , FDR=.01, error_rate=1e-3, depth_cutoff=10)

Note that there are several options which may be specified in addition to the filepath. First, is sample_file which may point to a comma-separated file

sample_file: Points to a comma-separated list of strain names, sorted in the same order as the in the SNP-table. If not specified, the function assumes such a file exists in the parent directory of infile

ignore: A python list of samples or path to comma-separated file containing a list of samples that should be excluded from the analysis.

pop_rule: Assume that this many characters at the beginning of each sample name indicates the population to which it belongs. For example if Beijing samples are named B01, B04, while Netherland samples are named N03, N05, pop_rule should be set to 1. On the other hand if they are named Bei01, Bei04, Net03, Net05 pop_rule should be set to 3. In future, it will be possible to pass a dictionary or table describing how samples should be assigned to populations, but this is not currently implemented.

FDR: This sets the per sample false-discovery rate used when calling the presence or absence of variant alleles. See below for more details.

error_rate: This specifies the assumed rate of base-calling errors contaminating the SNP pileup. By default in the ConTExt pipeline, only positions in reads with PHRED quality scores ≥ 30 are including the the pileup, so this parameter is set to 30 by default.

depth_cutoff: Subsequent analyses require that each analyzed position be supported by a minimum number of reads in each sample. Positions that fail to satisfy this requirement are excluded.

2.2 Information stored by the SequenceSummary

The various summaries computed by the SequenceSummary class are stored as attributes.

summary.pileup: Stores the SNP pileup

summary.major_allele_proportion: A IxK table where each cell stores the major allele proportion in sample i at position k.

summary.passed_positions: A K-length array indicating whether position k passed the read depth cutoff.

summary.nan_filter: A boolean K-length array indicate all samples are free from NaNs at a given position k. May be used to filter NaNs from subsequence analyses.

summary.allele_frequency: A K-length array which indicates the proportion of samples that clearly have a variant allele in at least one repeat at a given position k.

summary.variant_table: An IxK table: An IxK array indicating whether a sample i has at least one variant allele at position k.

summary.FST: A differentiation statistic is computed for each position. This differs slightly from the standard F_{st} statistic in that it is computed on allele proportions, which range from 0 to 1, rather than on indicator variables, which take values of 0 or 1.

2.3 Mathematical details

2.3.1 Calling Variant Alleles

At a given position, variant alleles are called by rejecting the null hypothesis that all observed variants are sequencing errors assuming a constant base-calling error rate ϵ. To be conservative, ϵ is set to 10^{-3}, matching the minimum quality score for inclusion in the SNP pileup (PHRED 30). Note that this does not account for errors introduced by PCR, which may be important. For each position i in sample j the total read count X_ij and variant read count k_ij are determined. The putative variant allele is defined as the allele with the second greatest average allele proportion. The number of variant reads is modeled as

a arising from binomial distribution:

$$k_{ij} \sim Binom(X_{ij}, epsilon) \tag{1}$$

This is further conservative in that it assumes that all basecalling errors yield the same nucleotide. A p-value is computed for each sample and the null hypothesis is (or isn't) rejected using the Benjami-Hochsberg multiple testing correction for the number of samples at the given FDR (default .01).

2.4 Principle Component Analyses

PCA can be carried out on the major allele proportion of a sequence summary by calling the PerformPCA() method.

summary.PerformPCA(self, selection_method=None, colors=colors, nan_filter=True)

By default, this excludes positions that take a value of NaN in at least one sample. A selection_method may be specified to perform feature selection by way of regularized regression on the positions to determine which most strongly contribute to each component. Otherwise, the feature contributions are determined from the PCA itself.

This method adds the following attributes to the SequenceSummary:

self.components: The PC loadings of each sample

self.contributions: The contributions of each position to the PC loadings

self.PCA: The scikit-learn PCA object storing the fitted PCA

self.rescaled: The mean-centered data input into the PCA

Once a PCA has been added to the SequenceSummary, several plotting functions may be employed to visualize it.

To visualize a scree plot:

summary.PlotScree()
pyplot.show()

To generate a scatter plot between the first and second components:

summary.PlotPCA(1,2)
pyplot.show()

To generate a 3D scatter plot for the first, second and third components:

summary.PlotPCA3D(1,2,3)

```
pyplot.show()
```

To plot the contributions for the 1st component:

```
summary.PlotContributions(1)
pyplot.show()
```

Note that all of the plotting functions use the standard nomenclature for PCA components (1 and 2 for the 1st and 2nd component) rather than Python indices (0 and 1 for the 1st and 2nd components). Under the hood, these functions subtract one from each of the input parameters. However, the direct interactions with the components and contributions attributes still require standard python indices (0-based).