PEAS

Table of Contents

Requirements & Dependencies

Feature Extraction Requirements & Dependencies

Feature Extraction requires the following tools/commands to be installed/available:

Bash
Feature extraction utilizes a shell script which incorporates many other tools that only run in linux/unix-like environments.

Java (version 1.8.0_171 or more recent):
Download Location: https://java.com/en/download/
Command: java -jar

SAMTools (Tested using version 1.2):
Download Location: https://github.com/samtools/samtools/releases
Command: samtools

MACS2 (Tested using version 2.1.1.20160309):
Download Location: https://github.com/taoliu/MACS
Command: macs2

HOMER
Download Location: http://homer.ucsd.edu/homer/
Command: findMotifsGenome.pl
Command: annotatePeaks.pl

In addition, install the human hg19 package (or a different package depending on your reference genome) in HOMER.

Disk Space Requirements
PEAS requires 3X the size of the input BAM alone for processing insert related features in addition to space for other related functions. For example, if a BAM file is 6gb, PEAS will require approximately 18gb of disk space.

Predicting and Training Requirements & Dependencies

Python:
Download Location: https://www.python.org/downloads/
Command: python (This can be configured in the GUI)

Python Libraries:
1. numpy
2. pandas
3. sklearn
4. matplotlib

These packages can be installed using *pip* or *conda*:

```
pip install numpy pandas scikit-learn matplotlib
```

```
conda install --upgrade numpy pandas scikit-learn matplotlib
```

PEASTools (Only if building PEASTools.jar from source)

htsjdk.samtools
Download Location: https://github.com/samtools/htsjdk

apache math commons
Download Location: http://commons.apache.org/proper/commons-math/

Running PEAS

PEAS can be run via either it's GUI or through executing it's shell, python, or PEASTools jar file individually. There are six scripts in total that are to be run in a terminal:

1. PEASFeatureExtraction.sh
2. PEASTools.jar
3. PEASPredictor.py
4. PEASTrainer.py
5. PEASPredictionAnnotator.py
6. PEASTssPromoter.py

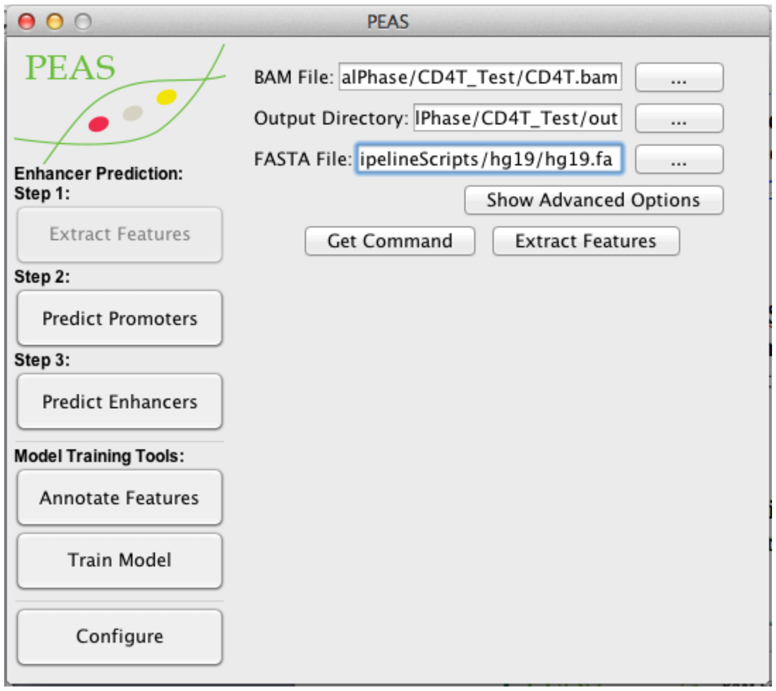
Additionally, PEAS.jar is included as a graphical user interface (GUI), which can be executed by double clicking on the file in most operating systems. PEAS.jar is the preferred method for using PEAS to predict enhancers as it takes care of most of the work in organizing input arguments.

Specific information on running the different command line scripts is provided in the file description section.

Feature Extraction

Step 1: Specify input files

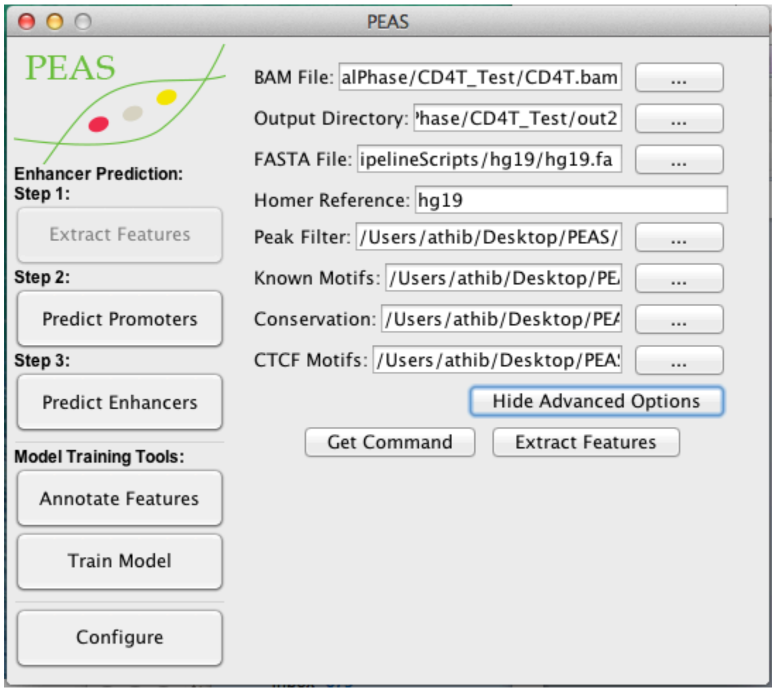
To run feature extraction, provide the bam file of paired-end ATAC-seq alignments, the output directory (ideally a newly created directory) and the FASTA file of the reference genome, for example the hg19 FASTA file from UCSC which can be obtained from http://hgdownload.cse.ucsc.edu/goldenPath/hg19/bigZips/ using 2bit extraction.


BAM File: alPhase/CD4T_Test/CD4T.bam

Output Directory: lPhase/CD4T_Test/out

FASTA File: ipelineScripts/hg19/hg19.fa

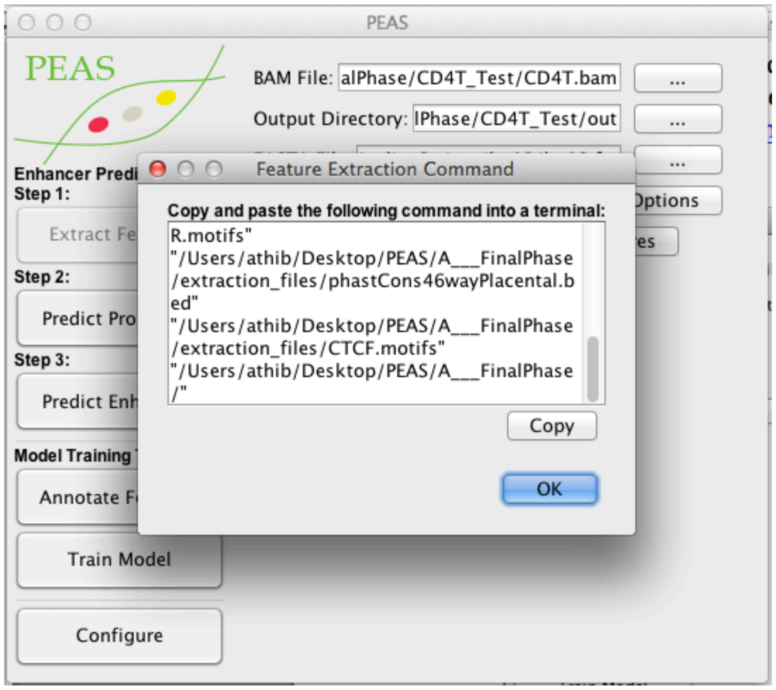
Advanced options allow for specify additional information which are required, but also included with PEAS and have already been specified based on the path of where PEAS is located:



BAM File: alPhase/CD4T_Test/CD4T.bam
Output Directory: hase/CD4T_Test/out2
FASTA File: ipelineScripts/hg19/hg19.fa
Homer Reference: hg19
Peak Filter: /Users/athib/Desktop/PEAS/
Known Motifs: /Users/athib/Desktop/PE
Conservation: /Users/athib/Desktop/PE
CTCF Motifs: /Users/athib/Desktop/PEA

Step 2a: Copying the command to a terminal

If you wish to run PEASFeatureExtraction in a terminal and/or make minor edits to the script, the “Get Command” button will display the command you would use to run the feature extraction.

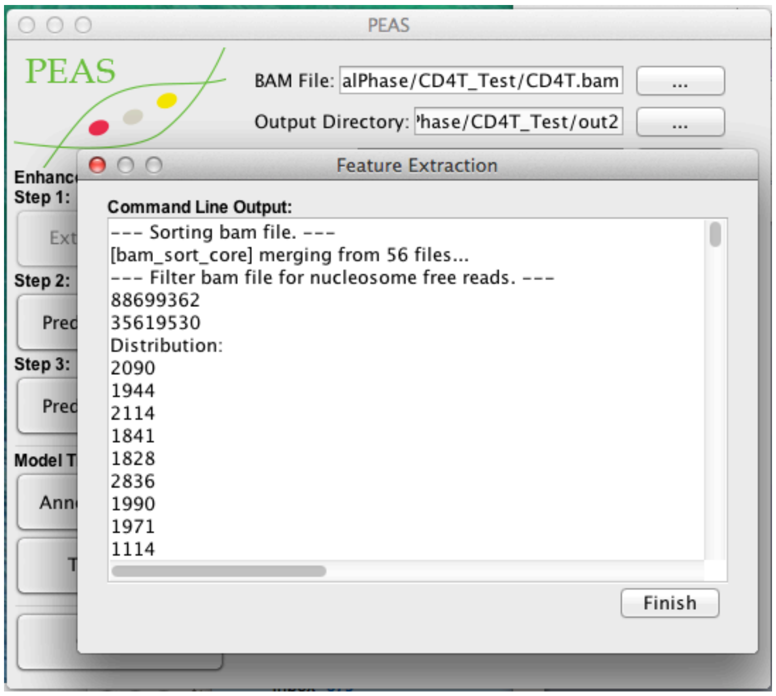


Copy and paste the following command into a terminal:

```
R.motifs"
"/Users/athib/Desktop/PEAS/A___FinalPhase
/extraction_files/phastCons46wayPlacental.b
ed"
"/Users/athib/Desktop/PEAS/A___FinalPhase
/extraction_files/CTCF.motifs"
"/Users/athib/Desktop/PEAS/A___FinalPhase
/"
```

Step 2b: Running feature extraction within the GUI:

"Extract Features" begins the feature extraction and displays the standard output of PEASFeatureExtraction.sh



BAM File: alPhase/CD4T_Test/CD4T.bam

Output Directory: 'hase/CD4T_Test/out2

Command Line Output:

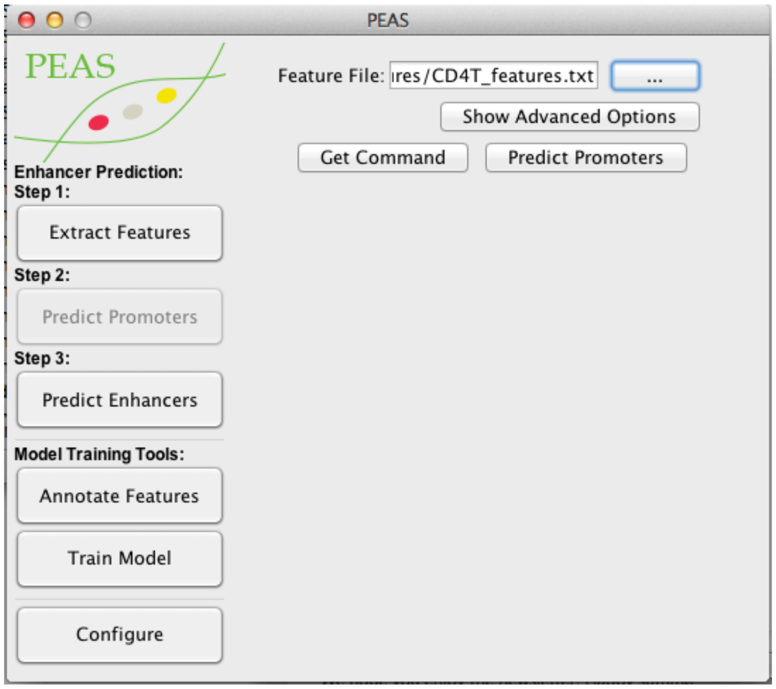
```
--- Sorting bam file. ---
[bam_sort_core] merging from 56 files...
--- Filter bam file for nucleosome free reads. ---
88699362
35619530
Distribution:
2090
1944
2114
1841
1828
2836
1990
1971
1114
```

If everything runs correctly you should find a *_features.txt file within the peak_features folder of the output directory specified.

Promoter Prediction

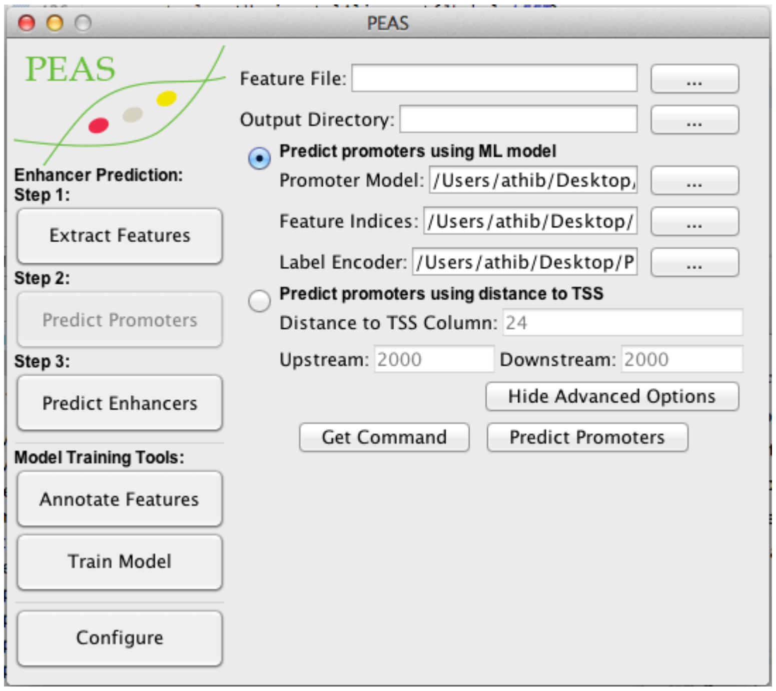
Step 1: Specify the feature file

Select the feature file from the extract features method. This will be labeled with *_features.txt within the output directories peak_features folder. If you have extracted features using the GUI in the current execution of PEAS, the file path should be entered for you already.



Step 2: Specify the type of promoter prediction (Optional)

PEAS provides two methods (within the advanced options) for predicting/annotating promoters. The first method uses a model previously trained (../models/promotermodel.pkl).

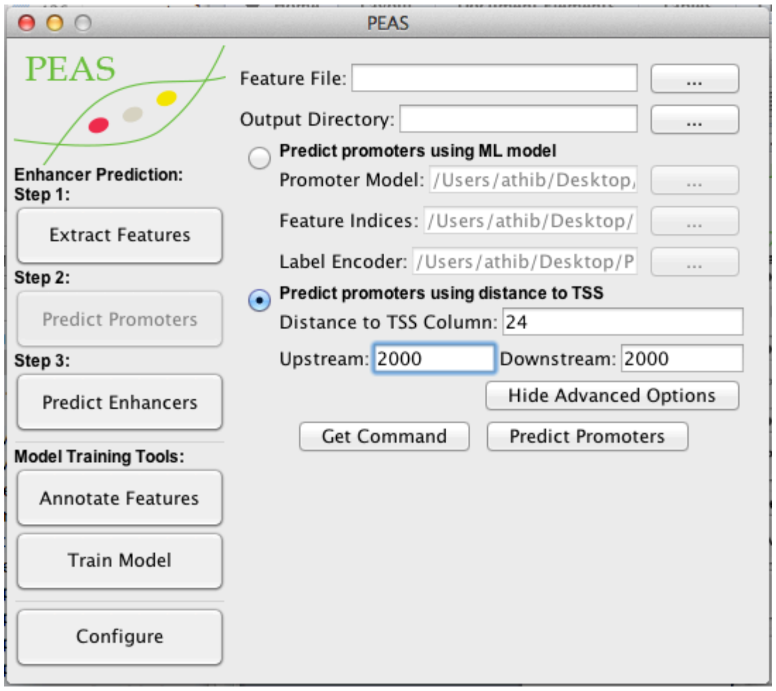

PEAS

Enhancer Prediction:
Step 1: Extract Features
Step 2: Predict Promoters
Step 3: Predict Enhancers
Model Training Tools: Annotate Features, Train Model
Configure

Feature File:
Output Directory:
[✓ Predict promoters using ML model]
Promoter Model: /Users/athib/Desktop/
Feature Indices: /Users/athib/Desktop/
Label Encoder: /Users/athib/Desktop/P
Predict promoters using distance to TSS
Distance to TSS Column: 24
Upstream: 2000 Downstream: 2000
Hide Advanced Options
Get Command
Predict Promoters

The fields for feature indices specify which columns contain the feature data, while the label encoder will transform non-numeric features into numeric features using the columns specified. These options are already filled for you but are available if there are any changes made to the features in the future. Finally, the output directory specifies where files will be saved. If no directory is specified, the directory from the feature file will be used.

The alternative method for predicting promoters uses the distance to TSS feature to specify promoters. Predicting promoters using TSS has three parameters to specify which column contains the distance to TSS (0 indexed, i.e., the first column has an index of 0) and the threshold for upstream and downstream distances.



Feature File:

Output Directory:

( ) Predict promoters using ML model

Promoter Model: /Users/athib/Desktop/

Feature Indices: /Users/athib/Desktop/

Label Encoder: /Users/athib/Desktop/P

(●) Predict promoters using distance to TSS

Distance to TSS Column: 24

Upstream: 2000 Downstream: 2000

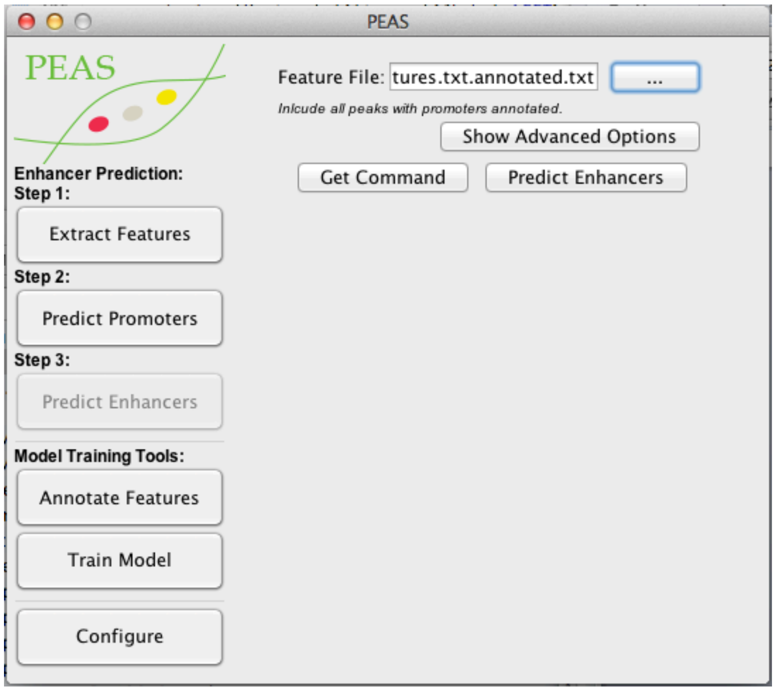
Step 3: Predict Promoters

As mentioned before in feature extraction, the get command option provides the command used to predict promoters in the terminal. However, using the GUI and using “Predict Promoters” includes a step that annotates the promoters (1 if promoter, 0 if other) for you in addition to setting the enhancer prediction file.

Enhancer Prediction

Step 1: Specify the feature file

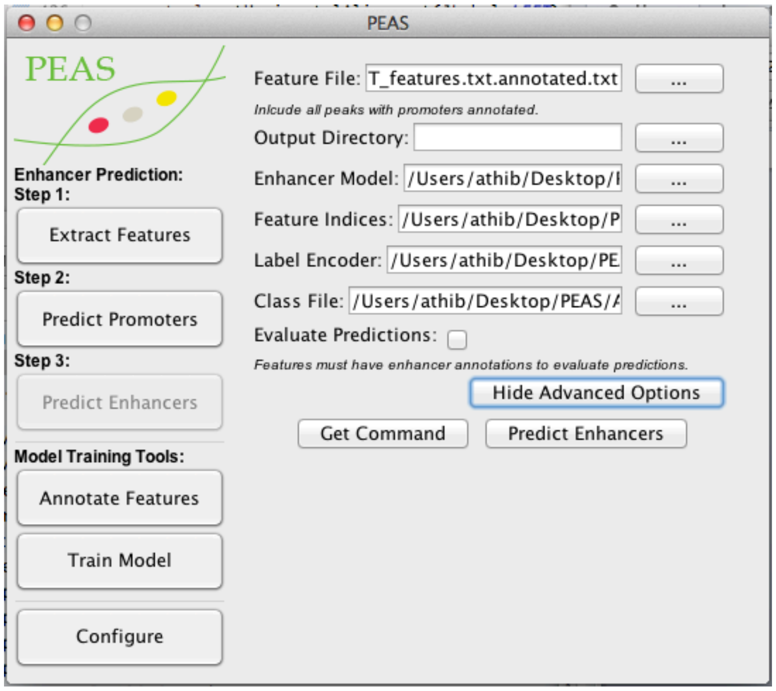
Select the feature file from the extract features method with either promoters annotated or removed. If you have extracted features using the GUI in the current execution of PEAS, the file path should be entered for you already.



Feature File: tures.txt.annotated.txt

Inlcude all peaks with promoters annotated.

Step 2: Specify the enhancer prediction configuration (Optional)



Feature File: T_features.txt.annotated.txt

Inlcude all peaks with promoters annotated.

Output Directory:

Enhancer Model: /Users/athib/Desktop/I

Feature Indices: /Users/athib/Desktop/P

Label Encoder: /Users/athib/Desktop/PE

Class File: /Users/athib/Desktop/PEAS/A

Evaluate Predictions: ☐

Features must have enhancer annotations to evaluate predictions.

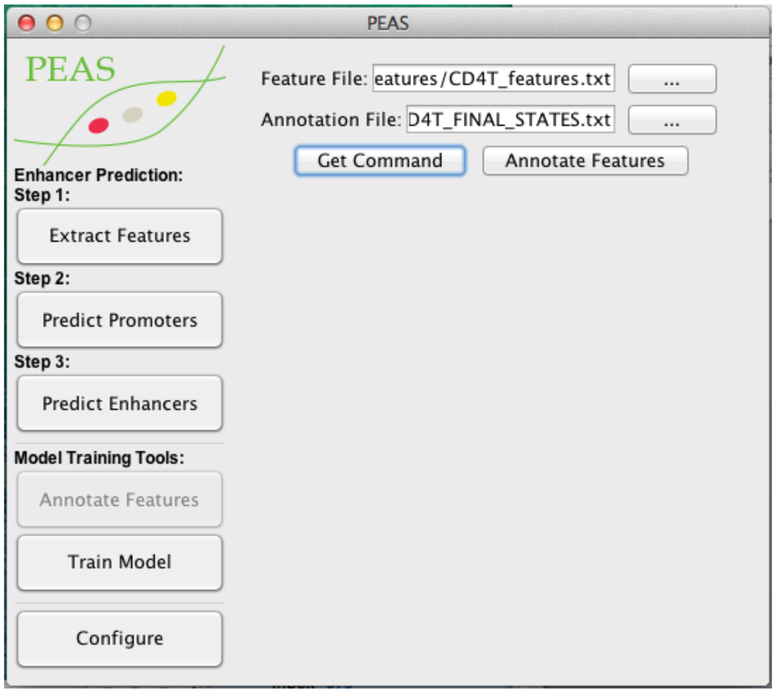
Enhancer prediction allows you to specify the output directory, the enhancer model, feature indices, label encoder, class file and whether or not to evaluate predictions. The output directory determines the location of output files. If no output directory is specified, the directory of the feature file is used. The enhancer model is used for setting the weights of the neural network. By default this file is provided for you. Feature indices and label encoders specify which columns in the feature file are features and how to encode non-numeric columns. The class file provides the way of which annotations are encoded as integers where 0 is a non-enhancer and 1 is an enhancer. Finally, if the feature file is annotated with enhancers already, the evaluate predictions will produce receiver operating characteristic and precision recall curves as well as confusion matrices of all features provided.

Step 3: Predict Enhancers

As mentioned before in the previous sections, the get command option provides the command used to predict enhancers in the terminal. To predict enhancers through the GUI use “Predict Enhancers”.

Annotate Features

PEAS provides a tool for annotating features directly with the “Annotate Features” option. Simply provide the un-annotated feature file and a 4 column file which specifies the chromosome, start, end, and annotation to annotate the feature file.

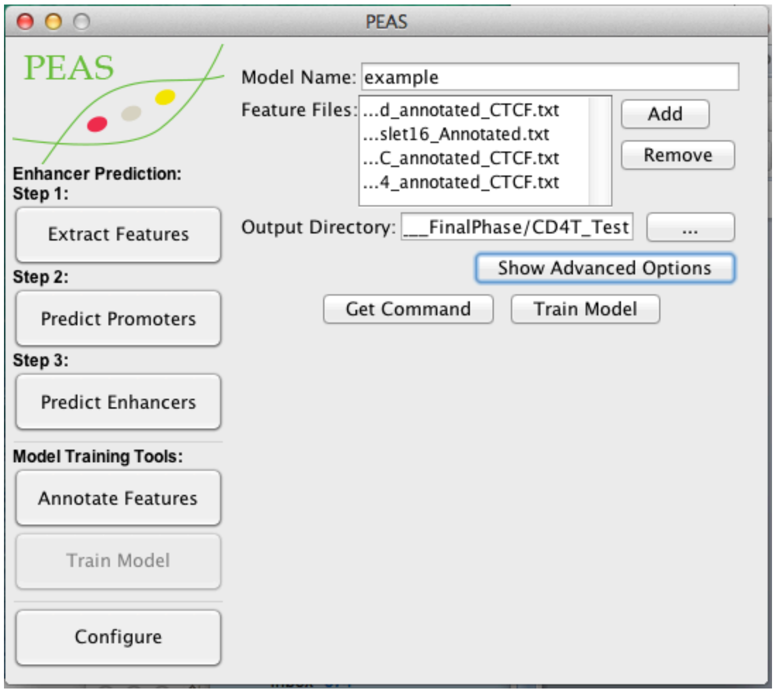


Feature File: eatures/CD4T_features.txt

Annotation File: D4T_FINAL_STATES.txt

Train Models

To train your own model, PEAS provides an easy interface which takes as input multiple annotated feature files and produces a *.pkl file for prediction.

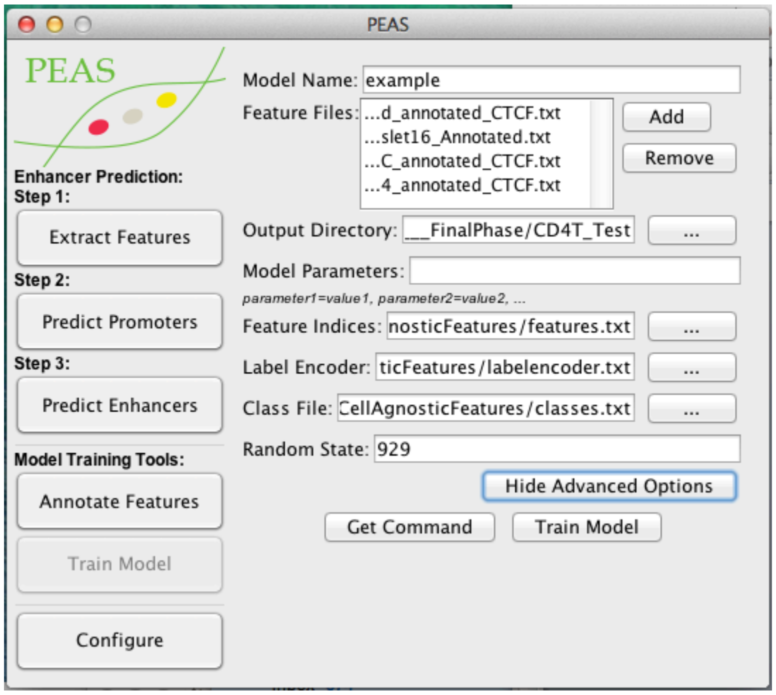


Model Name: example

Feature Files:
...d_annotated_CTCF.txt
...slet16_Annotated.txt
...C_annotated_CTCF.txt
...4_annotated_CTCF.txt

Output Directory: ___FinalPhase/CD4T_Test

Everything needed for configuring the model is provided in the advanced options menu. Just be sure the class file matches the annotations provided in the input feature files.



Model Name: example
Feature Files: ...d_annotated_CTCF.txt
...slet16_Annotated.txt
...C_annotated_CTCF.txt
...4_annotated_CTCF.txt
Output Directory: ___FinalPhase/CD4T_Test
Model Parameters:
parameter1=value1, parameter2=value2, ...
Feature Indices: nosticFeatures/features.txt
Label Encoder: ticFeatures/labelencoder.txt
Class File: CellAgnosticFeatures/classes.txt
Random State: 929

Model parameters are specified by the MLP classifier documentation from scikit-learn: http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

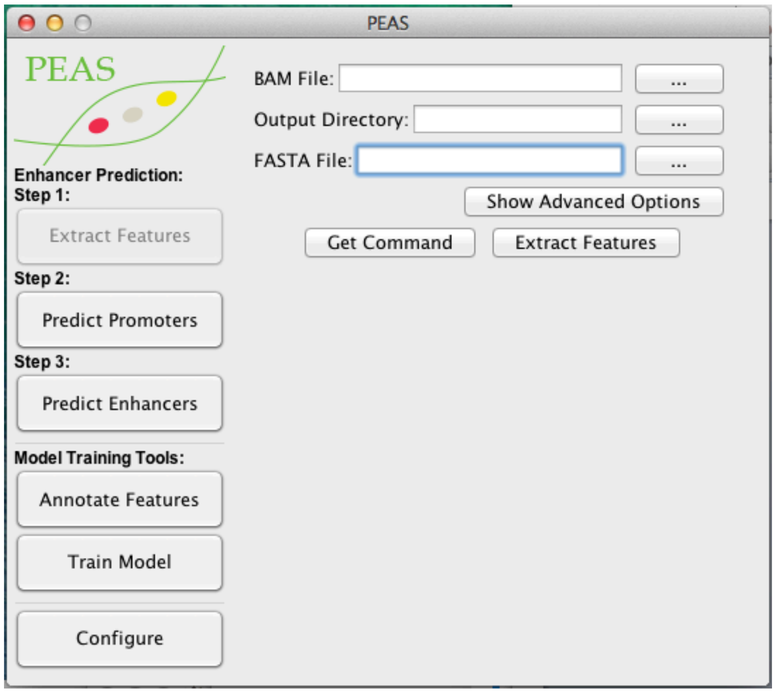
To train a model with parameters other than default, simply included a parameter string similar to the following example:

solver='adam', beta_1=0.999, beta_2=0.9999, epsilon=0.0000000001, activation='relu', hidden_layer_sizes=(25,)

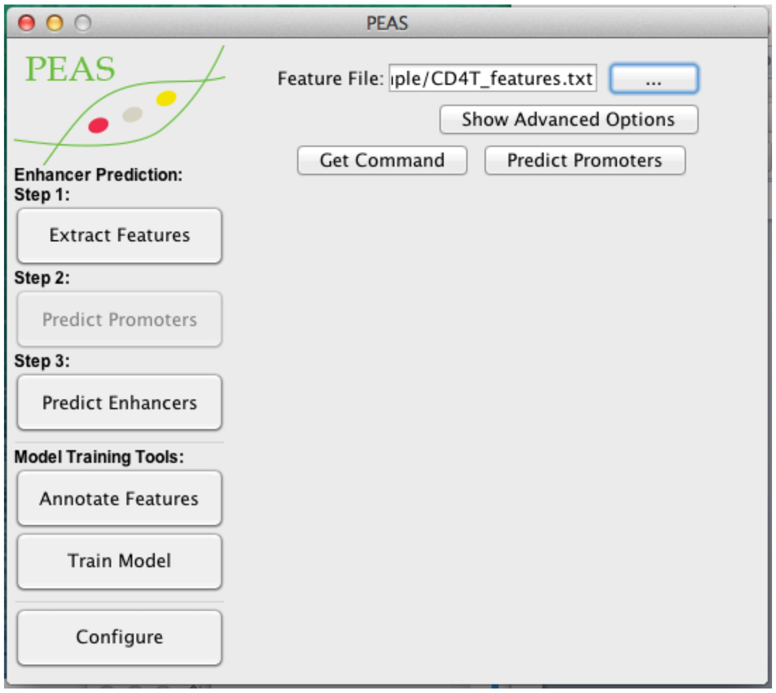
Note: Be sure to type single quotes ' for string values to avoid errors due to slight differences in characters.

Enhancer Prediction Example

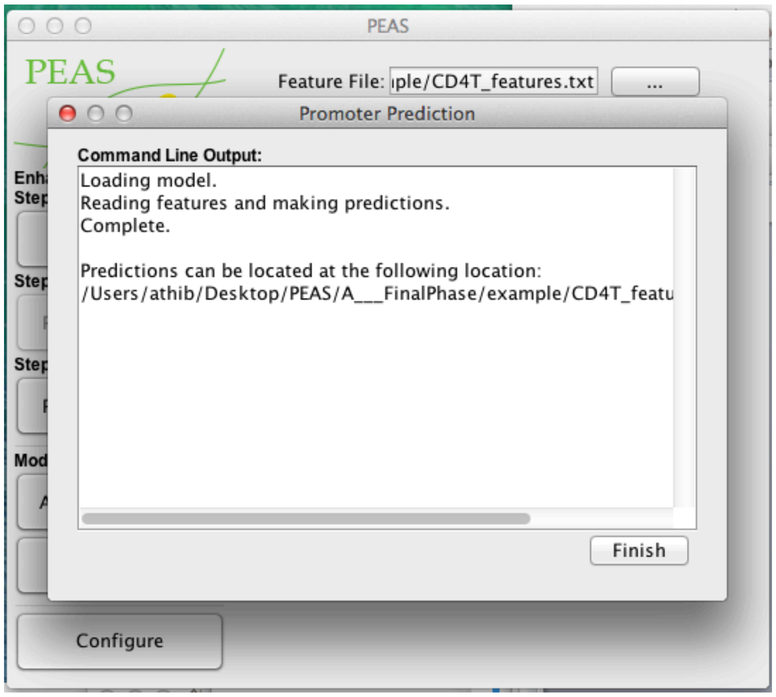
Here is a quick example of predicting enhancers after features have been extracted.

1. Open the PEAS.jar GUI by double clicking the jar file or running the jar file in a terminal:
   ```
   java –jar ./PEAS.jar
   ```
   

2. Since we are start with a feature file, navigate to "Predict Promoters" and set the feature file to "CD4T_features.txt" files that can be found in the example directory and select predict promoters.



You should see the following output message in the standard output viewer:



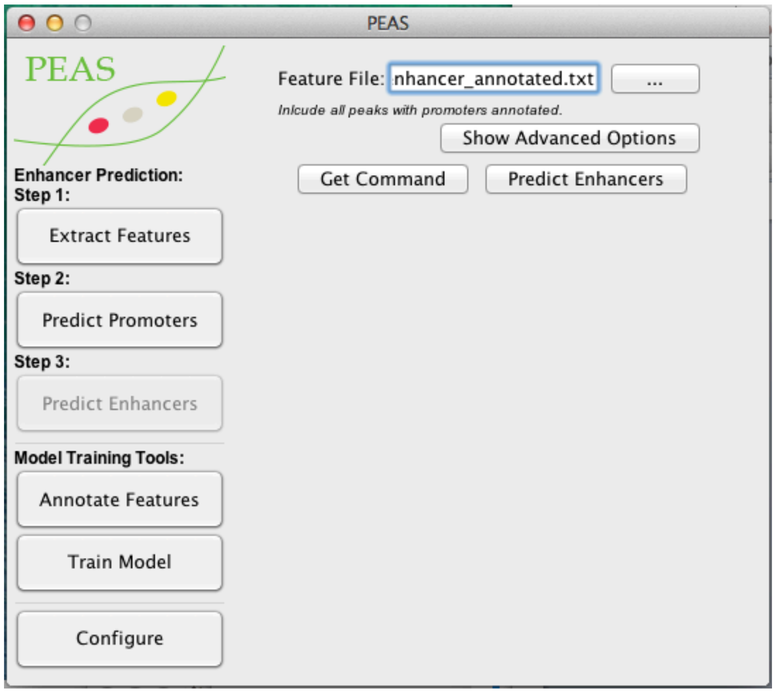
```
Loading model.
Reading features and making predictions.
Complete.

Predictions can be located at the following location:
/Users/athib/Desktop/PEAS/A___FinalPhase/example/CD4T_featu
```

3. Notice the file path provided by the standard out. This path is the location of the feature file with promoters annotated and would be used in the next step,

however for the purposes of this example, a CD4T file annotated with chromHMM enhancers has been provided.

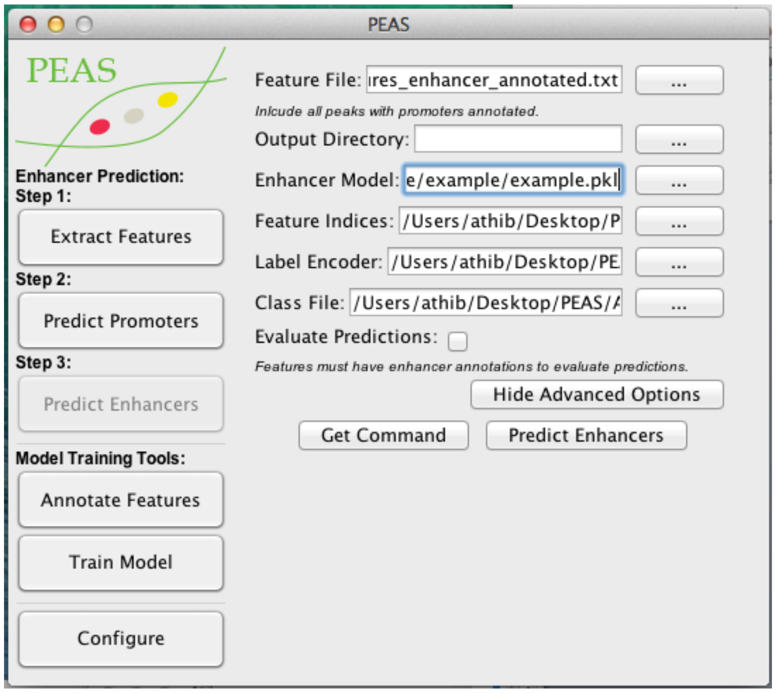
4. Navigate to “Predict Enhancers” and set the feature file to “CD4T_features_enhancer_annotated.txt” that can also be found in the example directory



Feature File: nhancer_annotated.txt

Inlcude all peaks with promoters annotated.

5. We could predict enhancers immediately using the default enhancer model, however, this would be biased since the default enhancer model has been trained using these features before. Instead, we will change the model to one that has not been trained using this dataset. To do this, select “Show Advanced Options” and change the enhancer model to “ example.pkl” file in the example folder.



Feature File: ıres_enhancer_annotated.txt

Inlcude all peaks with promoters annotated.

Output Directory:

Enhancer Model: e/example/example.pkl

Feature Indices: /Users/athib/Desktop/P

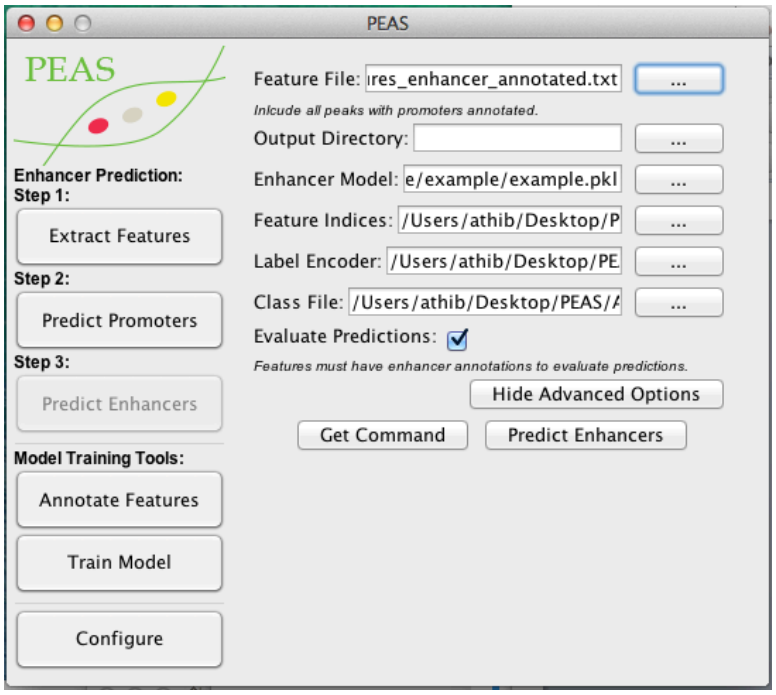
Label Encoder: /Users/athib/Desktop/PE

Class File: /Users/athib/Desktop/PEAS/

Evaluate Predictions: ☐

Features must have enhancer annotations to evaluate predictions.

6. Since we are working with annotated features, we can evaluate the predictions, producing a confusion matrix and receiver operating characteristic and precision recall curves. Check this box to generate these files.



Feature File: ıres_enhancer_annotated.txt

Inlcude all peaks with promoters annotated.

Output Directory:

Enhancer Model: e/example/example.pkl

Feature Indices: /Users/athib/Desktop/P

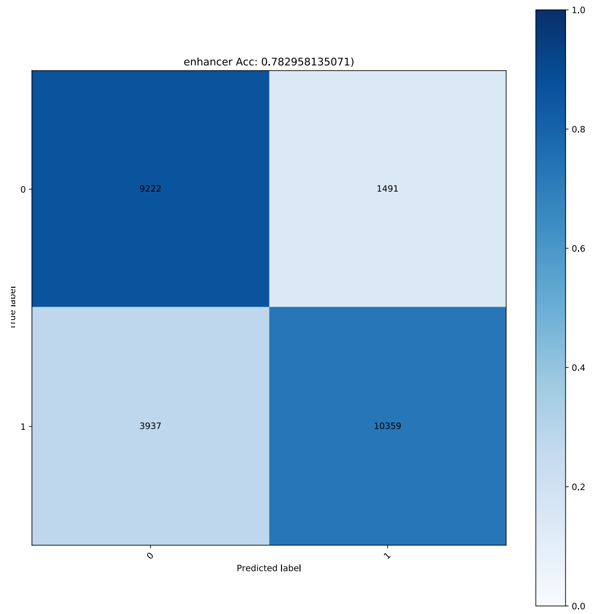
Label Encoder: /Users/athib/Desktop/PE

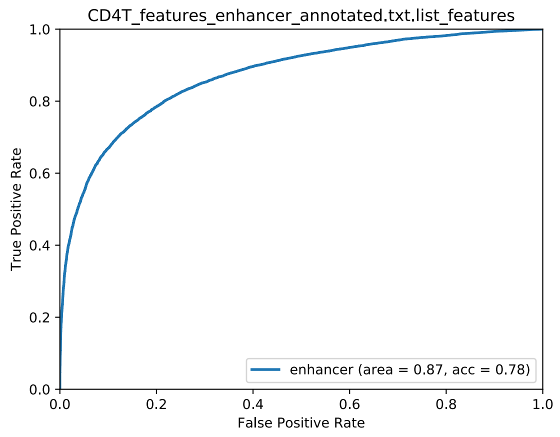
Class File: /Users/athib/Desktop/PEAS/

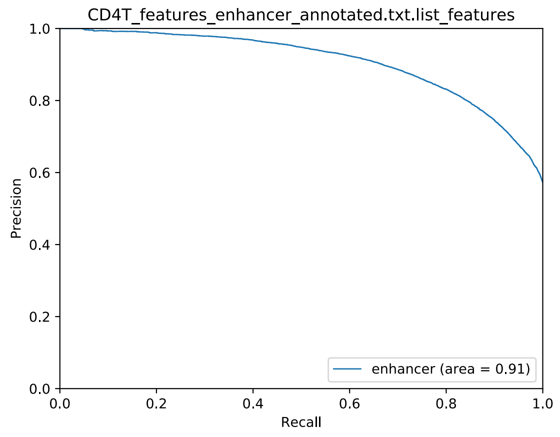
Evaluate Predictions: ✓

Features must have enhancer annotations to evaluate predictions.

7. Finally, press the “Predict Enhancers” button. You should find 3 result *.pdf files in the example folder that match the results below as well as a file with for enhancer predictions *_enhancer_predictions.txt:







File Descriptions

Jar Files

PEAS.jar:
The graphical user interface(GUI) for PEAS. Double clicking or running this file in the command line will bring up the GUI that interfaces with the other jar, python, and shell files to more easily predict enhancers from paired-end ATAC-seq data.

PEASTools.jar
A jar file that is runnable through command line/terminal. Most of its functionalities are demonstrated in PEASFeatureExtraction.sh with annotating features demonstrated in PEAS.jar. PEASTools.jar has the following options:

1. annotate
Description: Appends annotations and annotation statistics to a file when given a 4 column tab delimited file representing: chr, start, end, annotation.

Usage:
```
java –jar ./PEASTools.jar annotate <path of file to
annotate> <annotation file> <output file>
```

Columns Appended:
Best Annotation: The annotation given to the selected region.

Best Annotation Frequency: The percentage of the best annotation within the region.

Annotation Breakdown: The percent breakdown of all annotations overlapping the region.

Unique Annotation Count: The number of unique annotations found within the region.

2.bamfilter
Description: Creates a new bam file, filtering for reads of a specific threshold. This is used to call peaks on nucleosome free reads (reads with inserts <= 150bp)

Usage:
```
java –jar ./PEASTools.jar bamfilter <sorted bam file>
<size>
```

3. conservation
Description: Gets conservation scores from a bed file when given a bed file containing conservation information.

Usage:
```
java –jar ./PEASTools.jar conservation <bed file>
<conservation score bed file> <output file>
```

4. filter
Description: Removes locations from a bed file that may contain error prone regions.

Usage:
```
java –jar ./PEASTools.jar filter <bed file> <bed file
containing regions to remove> <output file>
```

5. insertsizethresh
Description: Obtains the insert size threshold for remove erroneous inserts. The output file will be located in the output directory as thresh.txt. Additionally, this method will split a bam file by chromosome.

Usage:
```
java –jar ./PEASTools.jar insertsizethresh <sorted bam
file> <output directory>
```

6. insertmetrics
Description: Obtains insert features for a specified chromosome within a given peak set.

Usage:
```
java –jar ./PEASTools.jar insertmetrics <chr> <sorted
bam file of the chromosome> <peak file including id
(4th column)> <output file> <insert size threshold>
```

Columns:
1. chr
2. start
3. end
4. peakid
5. insert count
6. cut count
7. insert mean
8. insert median
9. long/short insert ratio
10. (0,50]

11. (50, 150]
12. (150,300]
13. (300, 500]
14. (500,)
15. # of overrepresented cuts

7. merge
Description: merges different features into one feature file. Including MERGE at the end merges motifs into percentages and is the standard procedure for PEAS. Not including MERGE at the end will keep all motif counts separate. Additionally, each bed file will need to include a 4th column representing the peak ID.

Usage:
```
java –jar ./PEASTools.jar merge <peak file> <peak xls
file from MACS2> <annotated peak file from homer>
<peaks with insert metrics> <peaks with conservation
scores> <peaks with denovo motif annotations> <peaks
with CTCF motif annotations> <output file> MERGED
```

Python files

For all python files other than PEASUtil.py, simply run the python file without arguments for a description of input arguments and available options. A brief description of these files is provided below.

PEASPredictionAnnotator.py
Description: Annotates a feature file using a prediction file output. This is used to streamline annotating promoters after predicting them in order to annotate enhancers.

Usage:
```
python ./PEASPredictionAnnotator.py <feature file>
<prediction file>
```

PEASPredictor.py
Description: Makes predictions of features when given a model.

Usage:
```
python ./PEASPredictor.py [options] <model file *.pkl>
<feature file>
```

PEASTrainer.py
Description: Trains a model from feature files and creates a model (*.pkl file)

Usage:
```
python ./PEASTrainer.py [options] <Annotated Feature File>
```

PEASTSSPromoter.py
Description: Predicts promoters from a feature file using the distance to transcription start site (TSS) information.

Usage:
```
python ./PEASTssPromoter.py [options] <Feature File>
```

PEASUtil.py
Description: A library of functions used in PEAS python files. This utility file is mean to be imported for its various functions for reading data and plotting figures. Descriptions of each method can be read within the script.

PEASFeatureExtraction.sh

To extract features using command line arguments, locate the PEASFeatureExtraction.sh shell file and execute the following command using the following 9 arguments:

```
    <path to shell file>/PEASFeatureExtraction.sh <arg1>
    <arg2> <arg3> <arg4> <arg5> <arg6> <arg7> <arg8>
```

arg1: The path to the bam file's directory without the trailing "/".
Example: If the path is /user/documents/cd4t.bam, provide: **/user/documents**

arg2: The filename prefix (before .bam).
Example: If the filename is cd4t.bam, provide **cd4t**

arg3: The path to the fasta file location.
Example: **/user/documents/hg19.fa**

arg4: The reference genome to use for HOMER. This reference genome will need to be installed/configured in HOMER as well.
Example: **hg19**

arg5: The path to the bed file containing error prone regions of the genome to remove. This file is provided in the PEAS Github: https://github.com/UcarLab/PEAS.
Example: **/user/documents/hg19.filter.bed**

arg6: The path to the motifs file. This file is provided in the PEAS Github: https://github.com/UcarLab/PEAS.
Example: **/user/documents/humantop_Nov2016_HOMER.motifs**

arg7: The path to the conservation bed file. This file is provided in the PEAS Github: https://github.com/UcarLab/PEAS.
Example: **/user/documents/phastCons46wayPlacental.bed**

arg8: The path to the CTCF motifs file. This file is provided in the PEAS Github: https://github.com/UcarLab/PEAS. Example: **/user/documents/CTCF.motifs**

arg9: The path to the PEASTools.jar file without the trailing "/".
Example: If the PEASTools.jar file is located in /user/documents/peas/, provide **/user/documents/peas**

Example Feature Extraction Command:

```
    /user/documents/peas/PEASFeatureExtraction.sh
    /user/documents cd4t /user/documents/hg19.fa
    /user/documents/hg19.filter.bed hg19
    /user/documents/humantop_Nov2016_HOMER.motifs
    /user/documents/phastCons46wayPlacental.bed
    /user/documents/CTCF.motifs
    /user/documents/peas
```

Prediction and Training Files

classes.txt
Description: A tab-delimited file specifying the class label column and integer representation. One line is required for each class label, otherwise the class label is ignored.

Column 1: Class label column index
Column 2: Label in feature file
Column 3: Integer label to convert to. (Must start from 0 and increase incrementally)

For binary classification, positive classes should be converted to 1 and negative classes should be converted to 0. This can be used to combine different classes into the same class.

Note: Classes are relabeled in place and therefore if one class relabeling changes values all to 0, if the next class relabeling converts 0 to 1, for example, all of the previously converted values will be converted to 1.

enhancermodel.pkl & promotermodel.pkl
Description: Saved models for loading into PEASPredictor.py to make predictions.

features.txt
Description: A tab delimited file specifying which columns in the feature files are used. Multiple lines can be provided to specify different intervals in the following format:

Column 1*:* Start index (inclusive)
Column 2*:* End index (exclusive)

For example if the start index is 3 and the end index is 27, all columns from 3 up to (but excluding) column 27 will be included. Columns are 0 indexed (i.e. the first column index is 0).

labelencoder.txt
Description: Used for specifying non-numeric features that need to be converted to integers. One line per column, 0 indexed.

Column 1: Index column of the non-numeric feature
Column 2+: List of all non-numeric values, one column for each value.

***_predictions.txt**
Description: The output of PEASPredictor.py

Column 1: Chromosome
Column 2: Start
Column 3: End
Column 4: Integer *prediction* class label
Column 5 (only if in predictions are made in evaluation mode): Integer *true* class label
Column 5/6+: Probability for each class label starting from 0

Multiple Feature Files
To train and predict across multiple feature files at once, simply provide two-column tab delimited file in the following format where each line represents on feature file:

Column 1: Feature file label
Column 2: Feature File Path

Example:
CD4T ./user/documents/CD4T_features.txt