PREDICT User Manual

This manual provides detailed guidance for using each module in the PREDICT pipeline. It is intended for researchers familiar with genomic data analysis who wish to identify and characterize cis-regulatory elements (CREs) and their relationship with gene expression.

Table of Contents

Installation

PREDICT requires a Conda environment and includes scripts for both local and HPC (high-performance computing) installations.

1. Clone the Repository

```
git clone https://github.com/ChiaYiCheng-NTU/PREDICT.git
cd PREDICT/Install
```

2. Install Conda Environment

Option A: Local Installation

```
bash Install.sh
cd ../
```

Option B: HPC Installation

```
qsub -v CONDA_BASE=$(conda info --base) Install_PBS.sh
cd ../
```

> ⚠️ Ensure you modify `Install_PBS.sh` to match your HPC's job queue settings (e.g., `#PBS -q your_queue_name`).

3. Run the Example

Download and unpack the toy dataset:

```
wget https://github.com/ChiaYiCheng-NTU/PREDICT/releases/download/v1.1.0/PRED
ICT_ToyData.tar.gz
tar -xzvf PREDICT_ToyData.tar.gz
```

Run the pipeline using:

```
bash ExampleRUN.sh
```

You are now ready to begin running modules manually or adapting the pipeline to your own data.

---

1. Overview

PREDICT is a modular pipeline composed of five main components:

- ImpactKmers: Extract enriched k-mers from gene regulatory regions.
- Kmer2Motif: Map k-mers to known transcription factor binding motifs.
- RanKmers: Score k-mers based on their predictive value using machine learning.
- TeamKmers: Analyze co-occurrence patterns between motifs.
- ViewKmers: Visualize motif distribution in genomic context.

These modules can be executed independently or in sequence depending on the analysis goals.

---

2. Input Requirements

All modules require:

- **Reference Genome** (FASTA format)
- **Gene Annotation** (GFF3 format)
- **Lists of genes**: typically two sets:
  - **TP (True Positives)** – DEGs
  - **TN (True Negatives)** – non-DEGs

Additional inputs may include:

- **Known motifs** (FASTA format): A text file where each motif entry starts with a header line beginning with > followed by the motif name, and the next line is the motif consensus sequence. For example:

  ```
  >AT5G18090
  NNNGATGAANNNNND
  >AT5G25475
  CAAGCA
  >AT5G60130
  AAGNAAAWNNNAAGNAAAW
  >AT5G60130
  DTTTTGCTTAWDTTTTGCTTA
  ```

  A ready-to-use example motif file for Arabidopsis (`MotifList.fasta`) is included in the `PREDICT_ToyData.tar.gz` archive.

- **Predefined k-mer list**: A plain text file with one k-mer sequence per line. No headers or extra columns are required. For example:

  ```
  ATCGG
  TACGTAC
  ```

---

3. Module Descriptions

3.1 ImpactKmers

Goal: Identify enriched k-mers that distinguish TP from TN genes.

Arguments:

- `--gff` [required]: path to annotation file (.gff or .gff3)
- `--genome` [required]: path to genome FASTA file
- `--tp`, `--tn` [required]: gene lists for true positives and negatives
- `--features` [default: gene]: feature type in GFF file
- `--pThreshold` [default: 0.01]: P-value threshold for filtering enriched kmers.
- `--up_stream` [default: 1000]: upstream length from TSS (bp)
- `--down_stream` [default: 500]: downstream length from TSS (bp)
- `--alg` [default: `RandomForest`]: ML algorithm, options: `RandomForest`, `GradientBoosting`, `LogisticRegression`, `SVM`

Execution:

```
python path/to/PREDICT/PREDICT.py ImpactKmers \
  --gff input.gff3 \
```

```
  --genome genome.fa \
  --tp tp.txt \
  --tn tn.txt \
  --features gene \
  -- pThreshold 0.01 \
  --up_stream 1000 \
  --down_stream 500 \
  --alg RandomForest
```

Key Outputs:

- `AllCopies_FeatureImportance.txt`: Ranked k-mers and scores
- `KmerXKmer_table.tsv`: Co-occurrence matrix
- `TPGeneXKmer_Table.tsv`, `TNGeneXKmer_Table.tsv`: Count matrices per gene

3.2 Kmer2Motif

Goal: Match k-mers to known TF motifs

Arguments:

- `--kmer` [required]: path to k-mer list file
- `--motif` [required]: motif file in FASTA format
- `--TopKmers` [default: 0.1]: top fraction of k-mers to retain (range: 0–1)
- `--KeepTopMotifs` [default: 0.2]: top fraction of motifs to retain (range: 0–1)
- `--ScoreCutOff` [default: 0.9]: minimum similarity score for matching (range: 0–1)

Execution:

```
python path/to/PREDICT/PREDICT.py Kmer2Motif \
  --kmer Kmer_ShortTable.tsv \
  --motif known_motifs.fa \
  --TopKmers 0.1 \
  --KeepTopMotifs 0.2 \
  --ScoreCutOff 0.9
```

Key Output:

- `Kmer2Motif.tsv`: k-mer–motif mapping with similarity scores

3.3 RanKmers

Goal: Evaluate and rank user-defined k-mers using ML

Arguments:

- `--gff` [required]: path to annotation file (.gff or .gff3)

- `--genome` [required]: path to genome FASTA file
- `--tp`, `--tn` [required]: gene lists for true positives and negatives
- `--kmer` [required]: user-defined list of k-mers
- `--features` [default: gene]: feature type in GFF file
- `--up_stream` [default: 1000], `--down_stream` [default: 500]: region window from TSS
- `--alg` [default: `RandomForest`]: ML algorithm, same options as in ImpactKmers

Execution:

```
python path/to/PREDICT/PREDICT.py RanKmers \
  --gff input.gff3 \
  --genome genome.fa \
  --tp tp.txt \
  --tn tn.txt \
  --kmer user_kmers.tsv \
  --features gene \
  --up_stream 1000 \
  --down_stream 500 \
  --alg RandomForest
```

Key Outputs: Same as ImpactKmers

3.4 TeamKmers

Goal: Analyze co-occurring k-mers and/or motifs

Arguments:

- `--tp`, `--tn` [required]: gene × k-mer count tables from ImpactKmers/RanKmers
- `--motif` [optional]: motif file in FASTA format (if co-motif scoring is enabled)
- `--TPCSThreshold` [default: 0.75]: TP internal similarity threshold (range: 0–1)
- `--TPTNCSThreshold` [default: 0.3]: TP vs TN separation threshold (range: 0–1)

Execution:

```
python path/to/PREDICT/PREDICT.py TeamKmers \
  --tp TPGeneXKmer_Table.tsv \
  --tn TNGeneXKmer_Table.tsv \
  --motif known_motifs.fa \
  --TPCSThreshold 0.75 \
  --TPTNCSThreshold 0.3
```

Key Outputs:

- `CoocurredKmerPairs.tsv`, `MotifCoocurrenceScore.tsv`

3.5 ViewKmers

Goal: Visualize motif distribution interactively (via Streamlit)

Arguments:

- `--gff`, `--genome` [required]: genome and annotation files
- `--gene` [required]: list of genes to visualize
- `--kmer` [required]: list of k-mers to map
- `--motif` [optional]: motif file to overlay motif groups
- `--features` [default: gene]
- `--up_stream` [default: 1000], `--down_stream` [default: 500]

Execution:

```
streamlit run path/to/PREDICT/PREDICT.py -- ViewKmers \
  --gff input.gff3 \
  --genome genome.fa \
  --gene gene_list.txt \
  --kmer top_kmers.tsv \
  --motif known_motifs.fa
```

Key Output: Local web interface at `http://localhost:8501` Local web interface at `http://localhost:8501`

> Note: Best used with ≤ 5 k-mers and ≤ 20 genes for performance.

---

4. Best Practices

- Always validate your gene IDs match annotation.
- Use the same version of genome and annotation files across modules.
- Adjust `up_stream`/`down_stream` to match species-specific promoter size.
- Use moderate-sized k-mer lists for better performance in ViewKmers.
- Interpret co-occurrence results carefully—co-regulation may not imply direct interaction.

---

5. Output File Descriptions

This section provides descriptions for key output files generated by each module.

1. ImpactKmers / RankKmers

`AllCopies_FeatureImportance.txt`

Column	Description
Final_Rank	K-mers' average rank across 5-fold cross-validation

Column	Description
Weighted_percentile	Average percentile score across folds (lower is better)
Weighted_Percentile_sd	Standard deviation of percentile scores
Kmer	K-mer sequence with strand info prefix (e.g., nt_TCTTC)
Kmer_ID	Unique identifier for each k-mer (e.g., nt_K05_23223)
Orientation	Strand: Template (t) or NonTemplate (nt)
Average_Score	Mean feature importance score
Score_sd	Standard deviation of feature importance
Counts	Number of times selected across 5CV

Score Interpretation:

- Lower `Final_Rank` and `Weighted_percentile` values indicate more informative k-mers.
- High `Average_Score` suggests stronger predictive signal.
- High `Counts` (up to 5) indicate consistency across folds.

`ALLCopies_mean_score.txt`

Column	Description
Model	Machine learning algorithm used (e.g., RandomForest)
Copy Number	Feature aggregation method (e.g., ALL_COPIES_MEAN)
AUC	Area under ROC curve (0–1)
F1	F1-score (0–1)
MCC	Matthews correlation coefficient (–1 to 1)

Score Interpretation:

- `AUC` close to 1 suggests strong classification performance.
- `F1` near 1 means balanced precision and recall.
- `MCC` near 1 indicates strong overall predictive quality.

`Kmer_ShortTable.tsv`

Column	Description
Kmer	K-mer sequence (e.g., nt_TCTTC)
Kmer_ID	Formatted ID: strand + k-mer length + numeric code

`KmerXKmer_table.tsv`

- Symmetric matrix showing co-occurrence of each k-mer pair in the dataset.
- Diagonal: individual k-mer counts.
- High values indicate frequent co-occurrence.

`TPGeneXKmer_Table.tsv`, `TNGeneXKmer_Table.tsv`

- Rows: genes from TP or TN lists
- Columns: individual k-mers
- Values: frequency/count of each k-mer in the gene

---

2. Kmer2Motif

`Kmer2Motif.tsv`

Column	Description
KmerSeq	K-mer with strand info
KmerRank	Rank inherited from ImpactKmers result
MotifSeq	Matched motif consensus sequence
MotifInfo	Annotation or motif family name
Score	Similarity score (0–1, higher is better)

Score Interpretation:

- High `Score` indicates strong similarity between k-mer and motif.
- Useful for inferring TF binding potential.

---

3. TeamKmers

`CoocurredKmerPairs.tsv`

Column	Description
Kmer1, Kmer2	Pair of co-occurring k-mers
TPTN Cosine Similarity	Similarity score across TP and TN sets
TP Cosine Similarity	Similarity score within TP genes only
Kmer1_Count, Kmer2_Count	Occurrence counts across dataset
2Kmer_CO-Count	Count of genes where both k-mers co-occur

Score Interpretation:

- High cosine similarity suggests coordinated appearance in TP genes.
- `2Kmer_CO-Count` reflects potential co-regulatory modules.

`KmerXKmerTPCosineSimilarityTable.tsv`, `KmerXKmerTPTNCosineSimilarityTable.tsv`

- Symmetric matrices of cosine similarity values between k-mers.
- One matrix is based on TP genes only, the other on TP+TN combined.

`MotifCoocurrenceScore.tsv`

Column	Description
Motif1, Motif2	Co-occurring motif names
MCo Score	Co-occurrence score based on associated k-mers
Motif1_Count, Motif2_Count	Total number of mapped k-mers
2Motifs_CO-Count	Number of genes where both motifs co-occur

Score Interpretation:

- High `MCo Score` suggests frequent co-binding of TFs.
- Large `2Motifs_CO-Count` implies strong co-regulatory motif modules.

Score Calculation Formulas

Below are formulas used across various output files to compute statistical and machine learning evaluation metrics:

- **F1 Score:** F1 = (2 × Precision × Recall) / (Precision + Recall)
- **Matthews Correlation Coefficient (MCC):** MCC = (TP × TN - FP × FN) / sqrt((TP + FP)(TP + FN)(TN + FP)(TN + FN))
- **AUC:** Computed as the area under the ROC curve based on model prediction probabilities.
- **Weighted Percentile:** Aggregated percentile rank of feature importance across cross-validation folds.
- **Cosine Similarity:** CosSim(A, B) = dot(A, B) / (||A|| × ||B||), where A and B are occurrence vectors.
- **Motif Co-occurrence Score (MCo):** Sum of all k-mer pairwise co-occurrence counts between two motifs, normalized by total gene count.

6. Troubleshooting

Problem: No output files generated

- Check paths to input files (especially gene lists)
- Ensure annotation format matches expected gene entries

Problem: Low AUC or poor model performance

- Consider using another algorithm (e.g., GradientBoosting)
- Check whether gene classes (TP/TN) are imbalanced

Problem: ViewKmers is slow or unresponsive

- Reduce the number of genes or k-mers

- Run locally, not on a shared server

For further support, consult the GitHub Issues section or refer to example data and scripts in the `PREDICT_ToyData` package.