Methylation-Selection Analysis Toolkit

A comprehensive computational framework for inferring differential selection pressures across methylation states and mutation origins in cancer genomics.

Table of Contents

Overview

This toolkit implements a novel methodology to study how DNA methylation states influence evolutionary selection pressures on both germline and somatic mutations. The framework integrates multi-omics data to reveal methylation-dependent genetic constraints and their implications for cancer evolution.

Key Features

- **Comprehensive methylation classification** with multiple algorithms
- **Multi-format mutation annotation** (VCF, TSV) with functional consequences

- **Advanced selection inference** using dN/dS ratios, site frequency spectra, and Bayesian methods
- **Statistical comparison framework** with permutation testing and effect size estimation
- **Automated pipeline orchestration** with quality control and validation
- **Publication-ready visualizations** and comprehensive reports

Scientific Innovation

- **Joint modeling** of methylation state and mutation origin
- **Context-specific selection coefficients** revealing epigenetic-genetic interactions

- **Tumor evolution pathway preferences** identification
- **Novel therapeutic target** discovery through synthetic vulnerability mapping

Scientific Background

Hypothesis

DNA methylation state modulates selection pressures through multiple mechanisms:

Hypomethylated Regions: - Open chromatin structure - Active transcription and repair machinery - Stronger purifying selection - Higher functional constraint

Hypermethylated Regions:
- Closed chromatin structure - Transcriptional silencing - Relaxed selection pressure - Reduced repair efficiency

Differential Origins: - **Germline mutations**: Population-level evolutionary constraints - **Somatic mutations**: Tumor-specific selection advantages

Statistical Framework

The toolkit employs multiple complementary approaches:

1. **dN/dS Analysis**: Classic molecular evolution metric with confidence intervals
2. **Site Frequency Spectrum**: Population genetics approach for selection inference

3. **McDonald-Kreitman Tests**: Comparative method using polymorphism vs divergence
4. **Bayesian Hierarchical Models**: Account for gene-level and patient-level effects
5. **Permutation Testing**: Robust significance assessment with multiple testing correction

Installation

Prerequisites

```
pip install pandas numpy scipy matplotlib seaborn
pip install scikit-learn pybedtools pysam PyVCF3
pip install argparse pathlib datetime pyyaml
```

Python Requirements (>=3.8)

```
install.packages(c("tidyverse", "data.table", "lme4", "lmerTest",
                   "ggplot2", "cowplot", "viridis", "corrplot",
                   "broom", "car", "parallel", "optparse"))
```

R Requirements (>=4.0)

```
# BEDTools for genomic interval operations
sudo apt-get install bedtools

# Optional: VEP for variant annotation
# Follow installation instructions at: https://ensembl.org/info/docs/tools/vep/
```

External Tools

Installation Steps

1. **Clone the repository:**

```
git clone https://github.com/your-repo/methylation-selection-toolkit.git
cd methylation-selection-toolkit
```

2. **Set up Python environment:**

```
python -m venv methylation_env
source methylation_env/bin/activate
pip install -r requirements.txt
```

3. **Verify R packages:**

```
Rscript -e "library(tidyverse); library(lme4); cat('R packages loaded successfully\n')"
```

4. **Test installation:**

```
python methylation_selection_pipeline.py --create-config
python data_validation_utils.py --help
```

Quick Start

1. Create Configuration File

```
python methylation_selection_pipeline.py --create-config
```

This creates `example_config.yaml` with all parameters. Edit the file paths:

```
input_files:
  methylation_data: "/path/to/your/methylation_data.bedgraph"
  germline_mutations: "/path/to/your/germline_variants.vcf"
  somatic_mutations: "/path/to/your/somatic_variants.vcf"
  # ... additional optional files
```

2. Validate Your Data

```
python data_validation_utils.py \
  --methylation-data /path/to/methylation_data.bedgraph \
  --germline-mutations /path/to/germline_variants.vcf \
  --somatic-mutations /path/to/somatic_variants.vcf \
  --create-plots \
  --output validation_results/
```

3. Run Complete Pipeline

```
python methylation_selection_pipeline.py \
  --config your_config.yaml \
  --output results/
```

4. View Results

```
# Main results directory
ls results/

# Key files:
# - comprehensive_analysis_report.html (if generated)
# - comprehensive_analysis.png (main plots)
# - selection_analysis_selection_metrics.tsv (raw metrics)
# - ANALYSIS_SUMMARY.txt (executive summary)
```

Complete Workflow

The analysis consists of five main steps:

Step 1: Methylation Classification

```
python methylation_classify.py \
  --input methylation_data.bedgraph \
  --output methylation_classified.bed \
  --method threshold \
  --hypo-threshold 0.3 \
  --hyper-threshold 0.7 \
  --cpg-islands cpg_islands.bed \
  --plot-output methylation_distribution.png
```

Input formats supported: - BedGraph: `chr start end methylation_level` - Bismark: Standard Bismark cytosine report - Custom: Tab-separated with required columns

Classification methods: - `threshold`: Simple cutoff-based (default) - `adaptive`: Data-driven percentile thresholds
- `kmeans`: Unsupervised clustering approach

Step 2: Mutation Annotation

```
python annotate_mutations.py \
  --germline germline_variants.vcf \
  --somatic somatic_variants.vcf \
  --methylation-regions methylation_classified.bed \
  --output mutations_annotated.tsv \
  --vep-annotations vep_output.txt \
  --cadd-scores cadd_scores.tsv \
  --calculate-burdens \
  --calculate-selection
```

Features: - Multi-format mutation support (VCF, TSV) - Functional annotation integration (VEP, ANNOVAR, CADD) - Methylation state assignment with distance metrics - Preliminary selection metric calculation

Step 3: Selection Calculation

```
Rscript calculate_selection.R \
  --mutations mutations_annotated.tsv \
  --output selection_analysis \
  --gene-lengths gene_lengths.tsv \
  --min-mutations 5 \
  --bootstrap-n 1000 \
  --cores 4
```

Statistical methods: - **dN/dS ratios** with confidence intervals and neutrality tests - **Site frequency spectrum** metrics (Tajima's D, Fu & Li's D) - **McDonald-Kreitman** test statistics
- **Bayesian hierarchical** selection estimation - **Mixed-effects models** with gene-level random effects

Step 4: Comprehensive Comparison

```
Rscript compare_selection.R \
  --selection-scores selection_analysis_selection_metrics.tsv \
  --mutations mutations_annotated.tsv \
  --output comprehensive_analysis \
  --permutations 10000 \
  --generate-report
```

Advanced analyses: - **Permutation testing** (10,000 replicates) for robust significance - **Effect size estimation** (Cohen's d) with confidence intervals - **Multiple testing correction** (Benjamini-Hochberg) - **Bayesian multilevel modeling** with complex interactions - **Publication-ready visualizations** and HTML reports

Step 5: Quality Control

Automated quality control includes: - Input/output file validation - Pipeline execution monitoring
- Performance metrics tracking - Data integrity checks - Comprehensive reporting

Tools Description

Core Analysis Tools

1. `methylation_classify.py` **Purpose:** Classify genomic regions by methylation state

Key features: - Multiple classification algorithms (threshold, adaptive, k-means) - Genomic context integration (CpG islands, promoters, enhancers) - Regional analysis with sliding windows - Comprehensive validation and visualization

Output: BED file with methylation classifications, summary statistics, distribution plots

2. `annotate_mutations.py` **Purpose:** Annotate mutations with methylation states and functional consequences

Key features: - Multi-format input support (VCF, TSV) - Methylation state assignment with distance calculation - Functional annotation integration (VEP, ANNOVAR, CADD, conservation scores) - Mutation burden calculation - Preliminary selection metrics

Output: Comprehensive mutation annotation file, burden statistics, selection metrics

3. `calculate_selection.R` **Purpose:** Calculate comprehensive selection metrics

Key features: - dN/dS calculation with proper statistical framework - Site frequency spectrum analysis - McDonald-Kreitman tests - Bayesian hierarchical modeling - Bootstrap confidence intervals - Mixed-effects modeling with gene-level random effects

Output: Selection metrics table, statistical test results, model summaries

4. `compare_selection.R` **Purpose:** Advanced statistical comparison and visualization

Key features: - Comprehensive permutation testing - Effect size estimation with confidence intervals - Multiple testing correction - Bayesian multilevel modeling - Publication-ready visualizations - HTML report generation

Output: Comparison results, effect sizes, comprehensive plots, HTML report

Utility Tools

5. `methylation_selection_pipeline.py` **Purpose:** Complete pipeline orchestration

Key features: - Configuration-based execution - Automatic dependency tracking - Error handling and recovery - Quality control integration - Performance monitoring - Comprehensive logging

6. `data_validation_utils.py` **Purpose:** Comprehensive data validation and quality control

Key features: - Multi-format file validation - Data integrity checks - Quality metrics calculation - Validation plots generation - Detailed reporting with recommendations

Configuration

Configuration File Structure

```yaml
input_files:
  # Required files
  methylation_data: "/path/to/methylation.bedgraph"

  # At least one mutation file required
  germline_mutations: "/path/to/germline.vcf"
  somatic_mutations: "/path/to/somatic.vcf"

  # Optional annotation files
  cpg_islands: "/path/to/cpg_islands.bed"
  promoters: "/path/to/promoters.bed"
  enhancers: "/path/to/enhancers.bed"
```

```yaml
  # Optional functional annotations
  vep_annotations: "/path/to/vep_output.txt"
  annovar_annotations: "/path/to/annovar_output.txt"
  cadd_scores: "/path/to/cadd_scores.tsv"
  conservation_scores: "/path/to/conservation.tsv"
  gene_lengths: "/path/to/gene_lengths.tsv"

parameters:
  # Methylation classification
  hypo_threshold: 0.3               # Hypomethylation cutoff
  hyper_threshold: 0.7              # Hypermethylation cutoff
  classification_method: "threshold"  # threshold, adaptive, or kmeans
  methylation_format: "bedgraph"    # bedgraph, bismark, or custom

  # Mutation processing
  germline_format: "vcf"            # vcf or table
  somatic_format: "vcf"             # vcf or table

  # Analysis parameters
  regional_analysis: true           # Enable regional methylation analysis
  window_size: 1000                 # Regional analysis window size
  min_mutations_per_gene: 5         # Minimum mutations for gene analysis
  bootstrap_iterations: 1000        # Bootstrap replicates
  permutation_tests: 10000          # Permutation test replicates

  # Statistical parameters
  fdr_method: "BH"                  # Multiple testing correction method
  effect_size_threshold: 0.1        # Minimum meaningful effect size
  significance_threshold: 0.05      # Alpha level
  confidence_level: 0.95            # Confidence interval level

  # Computational parameters
  cores: 4                          # Number of CPU cores
  continue_on_failure: false        # Continue pipeline on step failure

output_settings:
  cleanup_intermediates: false      # Remove intermediate files
  generate_html_report: true        # Generate HTML report
  create_plots: true                # Generate visualization plots
```

Parameter Tuning Guidelines

Methylation Thresholds: - **Conservative**: 0.2 (hypo) / 0.8 (hyper) - Strict classification - **Standard**: 0.3 (hypo) / 0.7 (hyper) - Balanced approach
- **Liberal**: 0.4 (hypo) / 0.6 (hyper) - More intermediate states

Statistical Parameters: - **Permutation tests**: 1,000 (quick) / 10,000 (standard) / 100,000 (publication) - **Bootstrap iterations**: 100 (quick) / 1,000 (standard) / 10,000 (precise) - **Min mutations per gene**: 3 (liberal) / 5 (standard) / 10 (conservative)

Effect Size Interpretation: - Small effect: $|d| = 0.2$ - Medium effect: $|d| = 0.5$
- Large effect: $|d| = 0.8$

Output Interpretation

Key Output Files

1. Selection Metrics Table (`selection_metrics.tsv`)

gene_symbol	methylation_class	mutation_origin	dnds	ci_lower	ci_upper	p_value
TP53	hypomethylated	germline	0.324	0.156	0.492	0.001
TP53	hypermethylated	germline	0.756	0.423	1.089	0.234
KRAS	hypomethylated	somatic	1.234	0.987	1.481	0.012

Key columns: - `dnds`: Selection coefficient (< 1 = purifying, > 1 = positive, $= 1$ = neutral) - `ci_lower/ci_upper`: 95% confidence intervals - `p_value`: Test for deviation from neutrality (dN/dS = 1)

2. Effect Sizes Table (`effect_sizes.csv`)

comparison	cohens_d	ci_lower	ci_upper	magnitude
hypomethylated_vs_hypermethylated_germline	-0.524	-0.789	-0.259	Medium
germline_vs_somatic_hypomethylated	0.312	0.078	0.546	Small

Interpretation: - **Negative Cohen's d**: First group has lower dN/dS (stronger selection) - **Positive Cohen's d**: First group has higher dN/dS (weaker selection) - **Confidence intervals**: Significant if CI doesn't include 0

3. Permutation Results (`permutation_results.csv`)

test_statistic	observed_value	p_value	significant
hypo_vs_hyper_germline	-0.234	0.0023	TRUE
germ_vs_som_hypomethylated	0.156	0.0456	TRUE

Biological Interpretation

Typical Patterns Pattern 1: Methylation-Dependent Selection - Hypomethylated regions: dN/dS < 1 (strong purifying selection) - Hypermethylated regions: dN/dS $= 1$ or > 1 (relaxed/positive selection) - **Interpretation**: Open chromatin enables efficient selection against deleterious mutations

Pattern 2: Origin-Specific Effects
- Germline mutations: Generally stronger selection (lower dN/dS) - Somatic mutations: Context-dependent selection patterns - **Interpretation**: Population-level vs. tumor-specific constraints

Pattern 3: Gene-Specific Modulation - Essential genes: Strong selection regardless of methylation - Non-essential genes: Methylation-dependent selection patterns - **Interpretation**: Functional importance overrides epigenetic context

Statistical Significance

P-value interpretation: - $p < 0.001$: Highly significant departure from neutrality - $p < 0.01$: Significant selection - $p < 0.05$: Weak evidence for selection
- $p >= 0.05$: No evidence for selection (potentially neutral)

Effect size guidelines: - |Cohen's d| < 0.2: Trivial effect - $0.2 <=$ |Cohen's d| < 0.5: Small effect - $0.5 <=$ |Cohen's d| < 0.8: Medium effect
- |Cohen's d| $>= 0.8$: Large effect

Clinical Implications

Therapeutic Target Identification: 1. **Hypermethylated oncogenes** with relaxed selection → Demethylating agents 2. **Hypomethylated tumor suppressors** with strong selection → Synthetic lethality approaches 3. **Context-specific vulnerabilities** → Precision therapy strategies

Biomarker Development: 1. **Selection signature scores** for prognosis 2. **Methylation-mutation interaction** patterns for treatment response 3. **Evolutionary trajectory** prediction for resistance mechanisms

Advanced Usage

Custom Analysis Workflows

```r
# After running the pipeline, perform GSEA on selection results
library(fgsea)

# Load results
selection_data <- read.table("results/selection_analysis_selection_metrics.tsv",
                             header=TRUE, sep="\t")

# Prepare gene rankings
gene_rankings <- selection_data %>%
  filter(mutation_origin == "somatic", methylation_class == "hypomethylated") %>%
  select(gene_symbol, dnds) %>%
  deframe()

# Run GSEA with hallmark gene sets
hallmark_sets <- gmtPathways("h.all.v7.5.symbols.gmt")
gsea_results <- fgsea(pathways = hallmark_sets,
                      stats = gene_rankings,
                      nperm = 10000)
```

1. Gene Set Enrichment Analysis

```python
# Analyze selection changes over tumor evolution stages
import pandas as pd
import numpy as np

def temporal_selection_analysis(mutations_df, time_points):
    """
    Analyze selection patterns across tumor evolution stages
    """
    results = []

    for time_point in time_points:
        subset = mutations_df[mutations_df['time_point'] == time_point]
```

```
        # Calculate selection metrics for this time point
        selection_metrics = calculate_selection_for_timepoint(subset)
        selection_metrics['time_point'] = time_point

        results.append(selection_metrics)

    return pd.concat(results)

# Usage
temporal_results = temporal_selection_analysis(mutations_df,
                                               ['primary', 'metastatic', 'recurrent'])
```

2. Temporal Analysis

```
# Compare selection patterns across cancer types
compare_cancer_types <- function(selection_data) {
  # Group by cancer type and methylation class
  cancer_comparison <- selection_data %>%
    group_by(cancer_type, methylation_class, mutation_origin) %>%
    summarise(
      median_dnds = median(dnds, na.rm = TRUE),
      iqr_dnds = IQR(dnds, na.rm = TRUE),
      n_genes = n(),
      .groups = "drop"
    )

  # Statistical testing
  pairwise_results <- cancer_comparison %>%
    group_by(methylation_class, mutation_origin) %>%
    do(model = aov(median_dnds ~ cancer_type, data = .)) %>%
    mutate(
      p_value = map_dbl(model, ~ summary(.)[[1]][["Pr(>F)"]][1]),
      significant = p_value < 0.05
    )

  return(list(comparison = cancer_comparison,
              statistics = pairwise_results))
}
```

3. Multi-Cancer Comparison

Integration with Existing Pipelines

```
# Download and process TCGA data
def process_tcga_data(cancer_type):
    # Download methylation data
    meth_data = download_tcga_methylation(cancer_type)
```

```
    # Download mutation data
    mut_data = download_tcga_mutations(cancer_type)

    # Process for pipeline
    meth_processed = process_methylation_for_pipeline(meth_data)
    mut_processed = process_mutations_for_pipeline(mut_data)

    return meth_processed, mut_processed

# Usage
meth_data, mut_data = process_tcga_data("BRCA")
```

1. TCGA Data Integration

```
# Integrate with single-cell methylation data
process_sc_methylation <- function(sc_meth_data, cell_annotations) {
  # Aggregate by cell type
  celltype_meth <- sc_meth_data %>%
    left_join(cell_annotations, by = "cell_id") %>%
    group_by(cell_type, genomic_region) %>%
    summarise(
      mean_methylation = mean(methylation_level),
      cell_count = n(),
      .groups = "drop"
    )

  # Create cell-type-specific methylation profiles
  celltype_profiles <- celltype_meth %>%
    pivot_wider(names_from = cell_type,
                values_from = mean_methylation)

  return(celltype_profiles)
}
```

2. Single-Cell Integration

Performance Optimization

```
# Process large datasets in chunks
def process_large_methylation_file(filepath, chunk_size=100000):
    """
    Process methylation files too large for memory
    """
    chunk_results = []

    for chunk in pd.read_csv(filepath, sep='\t', chunksize=chunk_size):
        # Process chunk
```

```
        chunk_classified = classify_methylation_chunk(chunk)
        chunk_results.append(chunk_classified)

    # Combine results
    return pd.concat(chunk_results)
```

1. Large Dataset Handling

```
# Parallelize selection calculations
library(parallel)
library(foreach)
library(doParallel)

# Setup parallel backend
cl <- makeCluster(detectCores() - 1)
registerDoParallel(cl)

# Parallel selection calculation
selection_results <- foreach(gene = unique_genes, .combine = rbind) %dopar% {
  gene_data <- mutations_df[mutations_df$gene_symbol == gene, ]
  calculate_gene_selection(gene_data)
}

stopCluster(cl)
```

2. Parallel Processing

Troubleshooting

Common Issues and Solutions

1. Memory Issues **Problem:** Pipeline fails with memory errors on large datasets

Solutions:

```
# Increase system memory limits
ulimit -v 16777216  # 16GB virtual memory

# Use chunked processing
python methylation_classify.py --chunk-size 50000

# Enable memory-efficient mode
python annotate_mutations.py --memory-efficient
```

2. R Package Dependencies **Problem:** Missing R packages cause script failures

Solutions:

```r
# Install missing packages
install.packages(c("lme4", "lmerTest", "emmeans"))

# Install Bioconductor packages
BiocManager::install(c("GenomicRanges", "rtracklayer"))

# Check package versions
packageVersion("lme4")  # Should be >= 1.1-26
```

3. File Format Issues **Problem:** Input files not recognized or parsed correctly

Solutions:

```bash
# Validate file formats
python data_validation_utils.py --methylation-data input.bedgraph

# Convert between formats
# BedGraph to custom format
awk 'OFS="\t" {print $1, $2, $3, $4, ".", "+"}' input.bedgraph > output.bed

# Fix chromosome naming
sed 's/^chr//' input.bed > output_nochr.bed
```

4. Statistical Convergence Issues **Problem:** Mixed-effects models fail to converge

Solutions:

```r
# Simplify random effects structure
model <- lmer(dnds ~ methylation_class + (1|gene_symbol), data = data)

# Use different optimizers
model <- lmer(dnds ~ methylation_class * mutation_origin + (1|gene_symbol),
              data = data,
              control = lmerControl(optimizer = "bobyqa"))

# Scale predictors
data$scaled_cadd <- scale(data$cadd_score)
```

5. Low Statistical Power **Problem:** Few significant results due to insufficient data

Solutions:

```yaml
# Adjust parameters in config
parameters:
  min_mutations_per_gene: 3        # Lower threshold
  permutation_tests: 100000        # More permutations
  bootstrap_iterations: 10000      # More bootstrap samples
  effect_size_threshold: 0.05      # Lower effect size threshold
```

Performance Benchmarks

Typical Runtime Expectations

Dataset Size	Step 1	Step 2	Step 3	Step 4	Total
Small (1M sites, 10K mutations)	5 min	10 min	15 min	20 min	50 min
Medium (10M sites, 100K mutations)	30 min	45 min	1.5 hr	2 hr	4.5 hr
Large (100M sites, 1M mutations)	3 hr	4 hr	8 hr	12 hr	27 hr

Memory Requirements

Dataset Size	RAM Required	Disk Space	Cores Recommended
Small	4 GB	10 GB	2-4
Medium	16 GB	50 GB	4-8
Large	64 GB	200 GB	8-16

Getting Help

```
# Check pipeline logs
tail -f results/pipeline.log

# Search for errors
grep -i "error\|failed" results/pipeline.log

# Check validation results
cat results/quality_control_report.txt
```

1. Log File Analysis

```
# System information
python --version
R --version
bedtools --version

# Package versions
python -c "import pandas; print(pandas.__version__)"
Rscript -e "packageVersion('lme4')"
```

2. Diagnostic Information

```
# Generate synthetic test data
python generate_test_data.py --output test_data/

# Run pipeline on test data
python methylation_selection_pipeline.py --config test_config.yaml --output test_results/
```

3. Test Data

Citation

If you use this toolkit in your research, please cite:

```
[Your Citation Here]
Methylation-Selection Analysis Toolkit: A comprehensive framework for inferring
differential selection pressures across methylation states in cancer genomics.
```

Contributing

We welcome contributions! Please see CONTRIBUTING.md for guidelines.

License

This project is licensed under the MIT License - see LICENSE file for details.

Acknowledgments

- Research supported by [Your Funding Sources]
- Built with open-source tools: pandas, R/tidyverse, BEDTools
- Inspired by evolutionary genomics and cancer biology communities