Ballpushing screen PCA analysis

The PCA we developped is designed to find robust hits in the TNT silencing screen we used to find brain regions involved in ball pushing. Briefly, it involves the following steps :

1. Generate descriptive quantitative and binary metrics describing flies activity levels, behavior and strategies
2. Run a correlation based analysis using two strategies to filter out highly correlated metrics or metrics containing too many missing values
3. Run a PCA parameters optimisation script showing the 4 best combinations of PCA, Sparse PCA and two lists of metrics
4. Run a PCA with each of these parameters, along with non optimised but reasonable parameters. Generate a consistency score based on how many times each line was found significant
5. Run a metric by metric comparison on the hits wits over 80% consistency

Below is a more detailed description of each step

Full list of metrics used for the PCA

chamber_exit_time : First time the fly exits the initial chamber

distance_moved : Total ball distance moved

distance_ratio : Ratio of distance moved over the length of the corridor

first_major_event : First big push (index)

first_major_event_time : First big push (time)

flailing : Proportion of time spent flailing

fraction_not_facing_ball : Fraction of time where the fly is trying to turn back

has_finished : Binary metric indicating whether the fly pushed the ball to the end

has_major : Binary metric indicating whether the fly had at least one big push

has_significant : Binary metric indicating whether the fly had at least one push above noise threshold

has_long_pauses : Binary metric indicating whether the fly had at least one long pause

head_pushing_ratio : Ratio of interaction time spent with the head closer to the ball than the legs

interaction_persistence : Average duration of interaction events

max_distance : Maximum ball distance from starting position

max_event : Event at which the ball was brought to its max distance

max_event_time : Time at which the ball was brought to its max distance

median_stop_duration : Median duration of stops

nb_events : Number of interactions

nb_long_pauses : Number of long pauses

nb_pauses : Number of pauses

nb_significant_events : Number of significant events

nb_stops : Number of stops

normalized_velocity : Fly velocity normalized by available space to accelerate (i.e., ball position)

overall_interaction_rate : Rate of interaction over time

persistence_at_end : Fraction of time spent at the end of the corridor once the ball has been pushed there

pulled : Number of pulling events

pulling_ratio : Ratio of pulling over pushing events

significant_ratio : Ratio of interactions leading to ball movements over number of interactions

time_chamber_beginning : Time spent in the chamber during the first 25% of the experiment

total_long_pause_duration : Cumulative duration spent in long pauses

total_pause_duration : Cumulative duration spent in pauses

total_stop_duration : Cumulative duration spent in stops

velocity_during_interactions : Fly speed when interacting with the ball

velocity_trend : Variation of velocity across the experiment

For more details on thresholds, metrics computation and descriptions, see the ballpushing metrics readme.

Correlation based analyses

We developed two complementary approaches for metric selection to reduce redundancy and improve PCA interpretability. Both methods filter metrics with excessive missing values (>5% threshold) and identify highly correlated metrics ($|\rho| \geq 0.8$) for redundancy removal.

Method 1: Objective Scoring Approach (`correlation_analysis.py`)

This method uses an **objective scoring system** to select representative metrics from correlation clusters:

Pipeline:

1. **Missing Value Filtering**: Remove metrics with >5% missing values
2. **Robust Scaling**: Apply RobustScaler to handle outliers consistently
3. **Correlation Degree Analysis**: Calculate how many high correlations (≥0.8) each metric has

4. **Bootstrap Stability Analysis**: Run 100 bootstrap iterations to assess selection stability
5. **Hierarchical Clustering**: Group highly correlated metrics using average linkage (distance threshold = 0.2, equivalent to $|\rho| \geq 0.8$)
6. **Objective Representative Selection**: For each cluster, select the metric with the lowest objective score

Objective Scoring Function (lower score = better for selection):

- **Correlation degree penalty**: Number of high correlations (more connections = higher penalty)
- **Missingness penalty**: Percentage of missing values
- **Stability bonus**: Negative selection frequency from bootstrap analysis

Outputs:

- `final_metrics_for_pca.txt`: Selected representative metrics
- `full_metrics_pca.txt`: All metrics passing missing value filter
- `metrics_correlation_matrix_selected.csv`: Correlation matrix of selected metrics
- `objective_metrics_analysis.csv`: Detailed analysis with degrees and scores

Method 2: Family-Based Clustering (`correlation_analysis_alt.py`)

This method uses **biological domain knowledge** to group metrics into functional families before correlation analysis:

Metric Families:

- **Timing**: `chamber_exit_time`, `first_major_event_time`, pause/stop durations
- **Intensity**: `nb_events`, `distance_moved`, `max_distance`, velocity metrics
- **Rates/Ratios**: `significant_ratio`, `pulling_ratio`, `head_pushing_ratio`
- **State/Success**: `has_finished`, `has_major`, `has_significant` binary flags
- **Persistence/Dynamics**: `interaction_persistence`, learning trends, `auc`
- **Orientation/Kinematics**: `fraction_not_facing_ball`, `flailing`, head positioning

Pipeline:

1. **Missing Value & Bias Analysis**: Assess how missing value filtering affects different genotype groups
2. **Family Assignment**: Automatically categorize metrics using keyword matching
3. **Per-Family Clustering**: Within each family, cluster highly correlated metrics
4. **Domain-Priority Selection**: Select representatives using biological importance and correlation properties
5. **Bootstrap Stability Validation**: 100 bootstrap iterations to validate selection consistency
6. **Cross-Family Deduplication**: Optional final step to remove remaining high correlations between families

Outputs:

- `final_metrics_for_pca_alt.txt`: Selected representative metrics
- `stable_metrics_for_pca_alt.txt`: Metrics selected in >50% of bootstrap iterations
- `final_metrics_by_family.txt`: Per-family representatives

- `metrics_selection_stability.csv`: Bootstrap stability analysis results

Comparison and Usage

Both methods typically reduce ~30-35 initial metrics to ~15-20 representatives while maintaining biological interpretability and statistical orthogonality. The **objective method** is more statistically rigorous and automated, while the **family-based method** preserves biological domain structure and allows for prior knowledge integration. We also keep the initial list for further tests.

PCA optimisation script

The PCA optimization script (`PCA_optimisation.py`) systematically evaluates different parameter combinations to identify the most effective PCA configurations for downstream consistency analysis. It tests both standard PCA and SparsePCA methods on the two metric lists generated from correlation analysis.

Parameter Grid Search

Standard PCA Parameters:

- **n_components**: [7, 10, 12, 15, 20] - Range of principal components to retain

SparsePCA Parameters:

- **n_components**: [7, 10, 12, 15] - Fewer components tested due to computational cost
- **alpha**: [0.01, 0.05, 0.1, 0.5, 1.0, 2.0, 5.0, 10.0] - L1 regularization strength (sparsity control)
- **ridge_alpha**: [0.01, 0.1, 1.0] - L2 regularization for numerical stability
- **method**: ["lars", "cd"] - Optimization algorithms (LARS vs Coordinate Descent)
- **max_iter**: [2000, 3000] - Maximum iterations for convergence
- **tol**: [1e-4, 1e-5] - Convergence tolerance

Multi-Criteria Evaluation System

Each parameter combination is evaluated using multiple performance metrics:

Core Metrics:

- **Explained Variance Ratio**: Total variance captured by principal components
- **Reconstruction Error**: Mean squared error between original and reconstructed data
- **Sparsity Metrics**: For SparsePCA, measures interpretability through loading sparsity
- **Convergence Success**: Whether the algorithm converged within iteration limits

BIC Penalty Integration:

The optimization includes Bayesian Information Criterion (BIC) to prevent overfitting:

```
BIC = n × log(RSS/n) + k × log(n)
```

where RSS is residual sum of squares, k is model complexity (n_components), and n is sample size.

Composite Scoring Function

For Standard PCA:

- Explained Variance (40%) + Reconstruction Quality (20%) + Component Interpretability (20%) + Convergence (10%) + Sparsity Baseline (10%)

For SparsePCA:

- Explained Variance (30%) + Sparsity Optimization (25%) + Component Interpretability (20%) + Reconstruction Quality (15%) + Convergence (10%)

BIC Penalty Application:

- Base score minus weighted BIC penalty (default weight: 0.15)
- Stronger penalty for SparsePCA to account for additional complexity

Output Generation

Top Configuration Selection:

- Retains top 5 parameter combinations per condition (metric list × method)
- Typically generates 4 conditions: List1_PCA, List1_SparsePCA, List2_PCA, List2_SparsePCA
- Total: ~20 optimized configurations for consistency analysis

Files Generated:

- `top_configurations.json`: Complete parameter specifications for consistency analysis
- `all_conditions_results_with_bic.csv`: Detailed results for all tested combinations
- `optimization_summary_with_bic.txt`: Human-readable summary with complexity analysis
- `multi_condition_summary_with_bic.png`: Visualization of performance vs complexity tradeoffs

Complexity-Performance Balance

The BIC penalty ensures selected configurations balance:

- **High explained variance** (statistical power)
- **Reasonable model complexity** (generalizability)
- **Method-appropriate sparsity** (interpretability)
- **Computational feasibility** (convergence success)

This optimization typically reduces the parameter space from ~1000+ combinations to 20 high-quality configurations, providing a robust foundation for consistency scoring across multiple PCA approaches.

PCA statistical analysis

The statistical analysis script (`PCA_Static.py`) implements a comprehensive robustness testing framework that evaluates PCA configurations across both optimized parameters and deliberately challenging edge

cases. This dual approach validates whether significant hits are genuine biological signals or optimization artifacts.

Enhanced Configuration Framework

Optimized Configurations (4 total):

- Top 5 parameter sets from each condition: List1_PCA, List1_SparsePCA, List2_PCA, List2_SparsePCA
- Derived from systematic optimization with BIC penalty balancing performance and complexity

Balanced Edge Case Design (9 total):

- **Best Parameters + Full Metrics (4)**: Optimal parameters tested on complete unfiltered metric set
- **Default Parameters + Full Metrics (2)**: Conservative parameters (10 components) with all metrics
- **Default Parameters + Optimized Metrics (3)**: Conservative parameters with correlation-filtered metrics

This 4:9 ratio provides balanced statistical power while testing robustness across parameter sensitivity.

Multivariate Statistical Testing Framework

The analysis employs a **multivariate-only testing approach** (`--multivariate-only` flag) that focuses on genuine multivariate relationships rather than univariate effects:

Core Statistical Tests:

1. **Permutation Test**: Tests multivariate mean differences using Euclidean distance
    - Null hypothesis: No difference in multivariate centroids between experimental and control groups
    - Uses 1000 permutations for robust p-value estimation
    - Captures overall multivariate effect size
2. **Mahalanobis Distance Test**: Tests group separation accounting for covariance structure
    - Incorporates correlation structure between principal components
    - More sensitive to subtle multivariate patterns than Euclidean distance
    - Uses permutation-based significance testing

Statistical Workflow:

```
For each genotype vs control comparison:
1. Extract PCA scores for experimental and control groups
2. Apply permutation test (observed vs null distribution)
3. Apply Mahalanobis distance test with covariance correction
4. Apply FDR correction (Benjamini-Hochberg) across all genotypes
5. Require significance in BOTH multivariate tests (FDR-corrected p < 0.05)
```

Dual Consistency Scoring System

The analysis generates **two complementary consistency metrics** for different analytical purposes:

1. Optimized-Only Consistency:

- Fraction of optimized configurations (0-4) where genotype is significant
- Represents performance under ideal analytical conditions
- Best for evaluating optimization effectiveness

2. Combined Consistency:

- Fraction of ALL configurations (0-13) where genotype is significant
- Includes both optimized and edge case performance
- Provides comprehensive robustness assessment

Robustness Classification

Genotypes are classified into biological significance categories:

Truly Robust Hits:

- Significant in both optimized AND edge case configurations
- Represents genuine, optimization-independent biological signals
- High confidence for publication and follow-up studies

Optimization-Dependent Hits:

- Significant only in optimized configurations
- May represent weaker effects requiring optimal analytical conditions
- Requires additional validation before biological interpretation

Edge Case Specific:

- Significant only in edge case configurations (rare)
- Often indicates analytical artifacts or overfitting

Output Generation and Ranking Systems

Primary Output Files:

1. `enhanced_consistency_scores.csv`: Complete results for all tested genotypes with dual consistency metrics
2. `optimized_only_consistency_ranking.csv`: Ranked by optimization performance (best for method validation)
3. `combined_consistency_ranking.csv`: Ranked by overall robustness (best for biological conclusions)
4. `robustness_analysis.csv`: Detailed breakdown of robust vs optimization-dependent hits

Consistency scores description

The consistency scoring system provides a robust measure of hit reliability across different analytical conditions. Rather than relying on single parameter configurations, consistency scores quantify how frequently each genotype emerges as significant across the full range of tested PCA approaches.

Consistency Score Calculation

Individual Configuration Testing: Each genotype undergoes statistical testing (multivariate permutation + Mahalanobis tests) across all 13 configurations (4 optimized + 9 edge cases). A genotype receives a "hit" for each configuration where both multivariate tests achieve FDR-corrected significance ($p < 0.05$).

Dual Consistency Metrics:

1. **Optimized-Only Consistency**: Hits in optimized configurations / 4 total optimized
2. **Combined Consistency**: Total hits across all configurations / 13 total configurations

By default, we focus on the **combined consistency**.

Percentage Conversion:

Consistency scores are expressed as percentages (0-100%) for intuitive interpretation, where:

- 100% = significant in ALL relevant configurations
- 80% = significant in 80% of configurations (high consistency threshold)
- 50% = significant in half of configurations
- 0% = never significant across any configuration

Metric by metric statistics

The final analysis component (`plot_detailed_metric_statistics.py`) performs post-hoc investigation of high-consistency hits (≥80% threshold) using individual behavioral metrics. Since robust hits have already been identified through the multivariate PCA consistency pipeline, this analysis focuses on **metric-by-metric decomposition** to understand which specific behavioral changes drive the multivariate signatures.

Post-Hoc Analysis Framework

Pre-Selected High-Consistency Hits: Only genotypes achieving ≥80% combined consistency from the PCA pipeline are analyzed, ensuring focus on the most robust biological signals.

Individual Metric Statistical Testing: Each high-consistency genotype undergoes **Mann-Whitney U tests** for each behavioral metric versus control, followed by FDR correction across metrics. This univariate approach directly identifies which specific behaviors are altered in each genotype.

No Redundant Multivariate Testing: Since multivariate significance has already been established through the consistency scoring, the post-hoc analysis focuses purely on interpretability through individual metric effects.

Two-Way Hierarchical Clustering Framework

The centerpiece visualization is a **two-way dendrogram** that simultaneously clusters genotypes and behavioral metrics using complementary distance metrics:

Row Clustering (Genotypes):

- **Distance Metric**: Euclidean distance on signed p-value matrix
- **Linkage**: Ward linkage for compact, interpretable clusters
- **Matrix Values**: Signed significance weights where:
    - Sign indicates effect direction (higher/lower than control)
    - Magnitude reflects Mann-Whitney U significance with gradient scaling:
        - $p < 0.05$: Full intensity (100%) - clearly significant
        - $0.05 \leq p < 0.1$: Medium intensity (60%) - trending effects
        - $0.1 \leq p < 0.2$: Low intensity (30%) - weak evidence
        - $p \geq 0.2$: Minimal intensity (10%) - background

Column Clustering (Metrics):

- **Distance Metric**: Correlation-based distance (1 - |correlation|)
- **Linkage**: Ward linkage on behavioral correlation structure
- **Purpose**: Groups functionally related metrics (timing, intensity, persistence, etc.)