**🧬 Parameter-Weighted Score (PWS): A Bacterial Microbiome Correlation Tool**

A comprehensive bioinformatics tool for analyzing correlations between bacterial species abundance and external parameters (productivity, environmental factors, etc.) with advanced statistical testing and visualization capabilities.

**🎯 Overview**

This tool implements a **Parameter-Weighted Score (PWS)** methodology to identify bacterial species that significantly correlate with external parameters across multiple experimental conditions. It features:

* **Flexible parameter correlation analysis** (productivity, pH, temperature, etc.)
* **Advanced statistical testing** with automatic test selection
* **Dual analysis approach**: individual optimized tests vs. conservative standardized tests
* **Professional visualization** with significance pattern classification
* **Automated pipeline** for complete analysis workflow

**🔬 Key Features**

**Statistical Rigor**

* **Normality testing** (Shapiro-Wilk) for appropriate test selection
* **Variance homogeneity** testing (Levene) for parametric test validation
* **Automatic test selection**:
  + Student's t-test (normal data, equal variances)
  + Welch's t-test (normal data, unequal variances)
  + Mann-Whitney U (non-parametric alternative)
* **Dual filtering approach** for maximum transparency

**Analysis Workflow**

1. **Parameter-weighted correlation** analysis with abundance data
2. **Multi-level statistical testing** (global + group-specific)
3. **Significance-based filtering** with abundance thresholds
4. **Comparative visualization** between analysis approaches
5. **Comprehensive reporting** with statistical details

**Output Classification**

* 🔴 **Red**: Globally significant (p < 0.05)
* 🟠 **Orange**: Multiple groups significant
* 🟡 **Yellow**: Single group significant
* ⚪ **Gray**: Not significant

**📋 Requirements**

**Dependencies**

pandas >= 1.3.0

numpy >= 1.21.0

scipy >= 1.7.0

matplotlib >= 3.4.0

**System Requirements**

* Python 3.8 or higher
* 4GB+ RAM (for large datasets)
* Unix/Linux/macOS/Windows

**🚀 Installation**

**Option 1: Clone Repository**

git clone https://github.com/yourusername/bacterial-pws-analysis.git

cd bacterial-pws-analysis

pip install -r requirements.txt

**Option 2: Direct Download**

wget https://github.com/yourusername/bacterial-pws-analysis/archive/main.zip

unzip main.zip

cd bacterial-pws-analysis-main

pip install pandas numpy scipy matplotlib

**📁 Input Data Structure**

**1. Abundance Data (abundance\_data.csv)**

Bacterial abundance matrix with taxonomic classification:

Phylum,Class,Order,Family,Genus,Species,F3A1,F3A2,F3A3,F4A1,F4A2,F4A3,F3B1,F3B2,F3B3,F4B1,F4B2,F4B3

Bacillota,Clostridia,Lachnospirales,Lachnospiraceae,Lachnoclostridium,Clostridium fimetarium,0,0.0039,0,0.0102,0,0,0,0.0009,0,0,0,0

Bacteroidota,Flavobacteriia,Flavobacteriales,Flavobacteriaceae,Flavobacterium,Flavobacterium buctense,0.0104,0.0027,0,0,0,0.0043,0.0398,0.0017,0.0254,0.0479,0.0027,0.0134

**Format specifications:**

* Taxonomic columns: Phylum, Class, Order, Family, Genus, Species
* Sample columns: Follow pattern {Group}{Condition}{Replicate} (e.g., F3A1, F4B2)
* Values: Relative abundance (0-1) or percentage (0-100)

**2. Metadata (metadata.tsv)**

Sample information with experimental design:

sample-id group condition parameter

F3A1 F3 A 3500

F3A2 F3 A 3500

F3A3 F3 A 3500

F3B1 F3 B 3500

F3B2 F3 B 3500

F3B3 F3 B 3500

F4A1 F4 A 9500

F4A2 F4 A 9500

**Column descriptions:**

* sample-id: Exact match with abundance data columns
* group: Experimental group identifier (e.g., Farm, Site, Treatment)
* condition: Comparison conditions (A = before, B = after)
* parameter: Numerical parameter for correlation (productivity, pH, etc.)

**🔧 Usage**

**Quick Start (Automated Pipeline)**

bash

python main\_pws.py

This runs the complete analysis pipeline automatically and organizes outputs in out\_pwd/.

**Manual Step-by-Step**

bash

*# Step 1: Parameter-weighted analysis*

python pws.py

*# Step 2: Statistical filtering*

python filtering5.py

*# Step 3: Generate visualizations*

python plotting\_script.py

**Interactive Parameter Input**

When prompted:

My parameter for correlation is: Productivity

Enter a single word describing your parameter (e.g., Productivity, pH, Temperature).

**📊 Output Files**

**Core Analysis Results**

* **parameter\_weighted\_analysis.csv**: Complete correlation analysis with PWS scores
* **top30.txt**: Summary of top 30 most impactful species
* **sup\_statistics.csv**: Detailed statistical analysis for all species

**Filtered Results**

* **filtrado.csv**: Species passing filters (individual optimized tests)
* **filtrado\_conservativetest.csv**: Species passing filters (conservative standardized test)
* **filtering\_report.txt**: Comparative analysis report

**Visualizations**

* **{Parameter}\_weighted\_barplot.png**: Individual tests visualization
* **{Parameter}\_weighted\_barplot2.png**: Conservative test visualization

**Analysis Log**

* **pipeline\_log.txt**: Complete execution log with timestamps

**📈 Interpretation Guide**

**Parameter-Weighted Score (PWS)**

The PWS represents the impact of each bacterial species on the parameter, weighted by experimental group characteristics:

PWS = Σ(Relative\_Change\_i × Parameter\_Factor\_i) / N\_groups

Where:

* Relative\_Change\_i: (After - Before) / Before for group i
* Parameter\_Factor\_i: Group\_Parameter / Mean\_Parameter
* N\_groups: Number of groups where species is present

**Statistical Significance**

* **Global test**: Compares all condition A vs. all condition B samples
* **Group tests**: Compares A vs. B within each experimental group
* **Test selection**: Automatic based on normality and variance testing

**Filtering Criteria**

Species included if they meet **both** conditions:

1. Mean abundance ≥ 0.5% across all samples
2. At least one significant test (p < 0.05) **OR** listed in top 30 PWS ranking

**🧪 Example Analysis**

**Sample Dataset**

* **Groups**: 4 farms (F3, F4, F5, F6)
* **Conditions**: Before treatment (A) vs. After treatment (B)
* **Parameter**: Crop productivity (kg/ha)
* **Replicates**: 3 biological replicates per condition

**Expected Results**

Top Contributing Species:

1. Flavobacterium buctense (PWS: +5.15, Global significance)

2. Chryseolinea serpens (PWS: +2.94, Multiple farms)

3. Clostridium fimetarium (PWS: -1.23, Single farm)

**🔍 Advanced Features**

**Dual Analysis Approach**

1. **Individual Tests**: Optimal statistical test for each species
   * Maximizes statistical power
   * Species-specific test selection
   * Higher sensitivity
2. **Conservative Test**: Standardized test for all species
   * Ensures comparability
   * More stringent significance
   * Publication-ready consistency

**Quality Control**

* **Normality assessment** for 100 randomly selected species
* **Variance homogeneity** testing when applicable
* **Missing data handling** with appropriate substitution
* **Minimum sample size** validation for statistical tests

**🤝 Contributing**

We welcome contributions! Please see our [Contributing Guidelines](CONTRIBUTING.md) for details.

**Development Setup**

bash

git clone https://github.com/yourusername/bacterial-pws-analysis.git

cd bacterial-pws-analysis

python -m venv venv

source venv/bin/activate *# On Windows: venv\Scripts\activate*

pip install -r requirements.txt

**📚 Citation**

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

bibtex

@software{bacterial\_pws\_analysis,

author = {Your Name},

title = {Bacterial Microbiome Parameter Correlation Analysis Tool},

url = {https://github.com/yourusername/bacterial-pws-analysis},

year = {2025}

}

**📝 License**

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

**🆘 Support**

**Common Issues**

* **Memory errors**: Reduce dataset size or increase system RAM
* **Statistical warnings**: Check for low abundance species (< 0.1%)
* **Plotting errors**: Ensure parameter name contains no special characters

**Getting Help**

* 📧 **Email**: [your.email@institution.edu](mailto:your.email@institution.edu)
* 🐛 **Issues**: [GitHub Issues](https://github.com/yourusername/bacterial-pws-analysis/issues)
* 💬 **Discussions**: [GitHub Discussions](https://github.com/yourusername/bacterial-pws-analysis/discussions)

**🏆 Acknowledgments**

* Statistical methodology inspired by ecological correlation analysis
* Visualization design following microbiome analysis best practices
* Testing framework adapted from clinical biostatistics protocols

<div align="center">

**🧬 Advancing Microbiome Research Through Rigorous Statistical Analysis 🧬**

[⭐ Star this repo](https://github.com/yourusername/bacterial-pws-analysis) if it helped your research!

</div>

**🧪 Alternative Analysis: Welch's t-test**

For users requiring a **standardized statistical approach** with robust variance handling, the tool provides an alternative workflow using **Welch's t-test** exclusively.

**📋 When to Use Welch's t-test**

**Recommended for:**

* **Publication requirements** demanding consistent statistical methodology
* **Datasets with suspected unequal variances** between groups
* **Conservative analysis** with well-established statistical precedent
* **Regulatory submissions** requiring standardized approaches

**Advantages:**

* ✅ **Robust to unequal variances** (no homoscedasticity assumption)
* ✅ **Consistent methodology** across all comparisons
* ✅ **Widely accepted** in scientific literature
* ✅ **Conservative approach** reduces Type I errors

**🔧 Welch's t-test Workflow**

**Step 1: Run Main Pipeline**

First, execute the standard analysis pipeline:

python main\_pws.py

This generates the required input files (parameter\_weighted\_analysis.csv, top30.txt).

**Step 2: Welch's t-test Filtering**

Execute the Welch-specific filtering:

python filtering\_welch.py

**Output:** filtrado1.csv - Species filtered using Welch's t-test exclusively

**Step 3: Generate Welch-specific Visualization**

Create plots based on Welch's test results:

python plot\_weighted\_scores.py

**Output:** parameter\_weighted\_barplot\_welch.png - Visualization labeled with Welch's methodology

**📊 Welch's t-test Outputs**

**Core Files**

* **filtrado1.csv**: Filtered species using Welch's t-test
* **welch\_filtering\_report.txt**: Detailed statistical report
* **parameter\_weighted\_barplot\_welch.png**: Welch-specific visualization

**File Structure (filtrado1.csv)**

Cor,Phylum,Class,Order,Family,Genus,Species,Parameter\_Weighted\_Score,F3A1,F3A2,...,Global-t-test,F3t-test,...,MEAN\_TOTAL,Mean\_before,Mean\_after,SD\_bef,SD\_after

**Note:** All statistical tests in this file use Welch's t-test (equal\_var=False)

**🔍 Comparison: Hybrid vs. Welch's Approach**

| **Aspect** | **Main Pipeline (Hybrid)** | **Welch's Alternative** |
| --- | --- | --- |
| **Test Selection** | Automatic (optimal per species) | Welch's t-test (all species) |
| **Variance Assumption** | Adaptive | Unequal variances |
| **Statistical Power** | Maximum (species-specific) | Conservative (standardized) |
| **Consistency** | Variable tests | Uniform methodology |
| **Publication Ready** | Research/exploration | Regulatory/conservative |
| **Output Files** | filtrado.csv + filtrado\_conservativetest.csv | filtrado1.csv |

**📈 Interpretation Guidelines**

**Color Classification (Welch's t-test)**

* 🔴 **Red (Vermelho)**: Global Welch's test significant (p < 0.05)
* 🟠 **Orange (Laranja)**: Multiple groups significant
* 🟡 **Yellow (Amarelo)**: Single group significant
* ⚪ **Gray (Cinza)**: No significant differences

**Statistical Significance**

All p-values in filtrado1.csv represent **Welch's t-test results**:

* **Robust** for unequal variances between conditions
* **Conservative** compared to Student's t-test
* **Appropriate** for small sample sizes (triplicates)

**🎯 Use Case Examples**

**Example 1: Regulatory Submission**

# Standard PWS analysis

python main\_pws.py

# Conservative Welch's filtering for submission

python filtering\_welch.py

python plot\_weighted\_scores.py

# Submit: filtrado1.csv + welch\_filtering\_report.txt

**Example 2: Comparative Analysis**

# Run both approaches for comparison

python main\_pws.py # Generates hybrid results

python filtering\_welch.py # Generates Welch's results

# Compare outputs:

# - filtrado.csv (individual tests)

# - filtrado\_conservativetest.csv (conservative)

# - filtrado1.csv (Welch's only)

**Example 3: Publication Workflow**

1. **Exploration**: Use main pipeline (filtrado.csv) for discovery
2. **Validation**: Confirm findings with Welch's approach (filtrado1.csv)
3. **Reporting**: Present Welch's results for methodological consistency

**⚠️ Important Notes**

* **Run main pipeline first**: Welch's scripts require parameter\_weighted\_analysis.csv and top30.txt
* **Same filtering criteria**: 0.5% abundance threshold + significance/TOP30 rule
* **Different statistical approach**: Only the test methodology differs
* **Complementary analysis**: Use both approaches for comprehensive insights

**🔬 Technical Details**

**Welch's t-test specification:**

scipy.stats.ttest\_ind(data1, data2, equal\_var=False)

**Key assumptions:**

* ✅ Independent samples
* ✅ Approximately normal distributions (robust to violations)
* ❌ **No equal variance assumption** (major advantage)
* ✅ Suitable for small sample sizes

**💡 Recommendation:** For most research applications, run both the main pipeline and Welch's alternative to benefit from comprehensive statistical coverage and methodological transparency.