# Running the Comprehensive Timbre Analyzer

## Prerequisites

First, ensure you have the required libraries installed:

```bash
pip install librosa numpy pandas matplotlib seaborn
```

## Directory Structure

Organize your audio files like this:

```
project_folder/
 ├── comprehensive_timbre_analyzer.py  # The main script
 ├── audio_samples/                    # Directory containing .wav files
│   ├── violin_sample.wav
│   ├── flute_sample.wav
│   ├── piano_sample.wav
│   └── guitar_sample.wav
```

## Basic Usage Examples

### 1. Analyze a Single Audio File

```python
from comprehensive_timbre_analyzer import ComprehensiveTimbreAnalyzer

# Initialize the analyzer
analyzer = ComprehensiveTimbreAnalyzer()

# Analyze a single file
result = analyzer.analyze_file("audio_samples/violin_sample.wav")

if result:
    # Print detailed analysis
    analyzer.print_detailed_analysis(result)

    # Create visualizations
    analyzer.plot_comprehensive_analysis([result])
```

## 2. Analyze Multiple Files in a Directory

```python
from comprehensive_timbre_analyzer import ComprehensiveTimbreAnalyzer

# Initialize the analyzer
analyzer = ComprehensiveTimbreAnalyzer()

# Analyze all .wav files in directory
results = analyzer.analyze_directory("audio_samples/")

if results:
    # Print individual analyses
    for result in results:
        analyzer.print_detailed_analysis(result)

    # Compare tonal qualities across files
    analyzer.compare_tonal_qualities(results)

    # Create comprehensive visualization dashboard
    analyzer.plot_comprehensive_analysis(results)

    # Create DataFrame for further analysis
    df = analyzer.create_comprehensive_dataframe(results)
    print(df.head())
```

## 3. Complete Analysis Script

Save this as `run_analysis.py`:

```python
```

```
#!/usr/bin/env python3

from comprehensive_timbre_analyzer import ComprehensiveTimbreAnalyzer
import sys
from pathlib import Path

def main():
    # Initialize analyzer with custom parameters if needed
    analyzer = ComprehensiveTimbreAnalyzer(
        n_mfcc=13,      # Number of MFCC coefficients
        sr=22050,       # Sample rate
        hop_length=512, # Frame hop length
        n_fft=2048      # FFT window size
    )

    # Check if directory argument provided
    if len(sys.argv) > 1:
        audio_dir = sys.argv[1]
    else:
        audio_dir = "audio_samples"  # Default directory

    # Verify directory exists
    if not Path(audio_dir).exists():
        print(f"Error: Directory '{audio_dir}' not found!")
        print("Please create the directory and add .wav files")
        return

    print(f"Analyzing audio files in: {audio_dir}")
    print("=" * 60)

    # Analyze all files in directory
    results = analyzer.analyze_directory(audio_dir)

    if not results:
        print("No .wav files found or analysis failed!")
        return

    print(f"\nSuccessfully analyzed {len(results)} files:")
    for result in results:
        print(f"  - {result['filename']} ({result['duration']:.2f}s)")

    # Generate comprehensive analysis
    print("\n" + "=" * 60)
    print("GENERATING COMPREHENSIVE ANALYSIS")
    print("=" * 60)

    # 1. Individual detailed analyses
    for result in results:
        analyzer.print_detailed_analysis(result)

    # 2. Comparative analysis
    if len(results) > 1:
        analyzer.compare_tonal_qualities(results)

    # 3. Create DataFrame and save to CSV
    df = analyzer.create_comprehensive_dataframe(results)
    csv_filename = "timbre_analysis_results.csv"
    df.to_csv(csv_filename, index=False)
    print(f"\nResults saved to: {csv_filename}")

    # 4. Generate comprehensive visualization dashboard
    print("\nGenerating visualization dashboard...")
    analyzer.plot_comprehensive_analysis(results)
    print("Dashboard complete! Close the plot window to continue.")

    print("\n" + "=" * 60)
    print("ANALYSIS COMPLETE!")
    print("=" * 60)

if __name__ == "__main__":
    main()
```

## Running the Script

### Method 1: Command Line with Default Directory

```bash
python run_analysis.py
```

### Method 2: Command Line with Custom Directory

```bash
python run_analysis.py /path/to/your/audio/files
```

### Method 3: Interactive Python Session

```python
# Start Python interpreter
python3

# Then run interactively:
from comprehensive_timbre_analyzer import ComprehensiveTimbreAnalyzer

analyzer = ComprehensiveTimbreAnalyzer()
results = analyzer.analyze_directory("your_audio_directory")
analyzer.plot_comprehensive_analysis(results)
```

## Expected Output

When you run the script, you'll get:

1. **Console Output:**
    - File-by-file analysis progress
    - Detailed MFCC coefficient interpretations
    - Tonal quality comparisons
    - Overall timbre profiles with emojis
2. **Visual Dashboard:**
    - 10 comprehensive plots showing all aspects of timbre analysis
    - Interactive matplotlib window with all visualizations
3. **CSV Export:**
    - `timbre_analysis_results.csv` with all numerical data
    - Ready for further analysis in Excel, R, or other tools

## Troubleshooting

### Common Issues:

#### "No module named 'librosa'"

```bash
pip install librosa
```

**"No .wav files found"**

- Ensure your audio files are in `.wav` format
- Check that the directory path is correct
- The script only processes `.wav` files by default

**"Error loading audio file"**

- Verify the .wav file isn't corrupted
- Check that the sample rate is supported
- Try converting with: `ffmpeg -i input.mp3 -ar 22050 output.wav`

## Customization Options:

You can modify the analyzer parameters:

```python
# For higher quality analysis
analyzer = ComprehensiveTimbreAnalyzer(
    n_mfcc=13,      # Keep all 13 coefficients
    sr=44100,       # Higher sample rate
    hop_length=256, # Smaller hop = more detail
    n_fft=4096      # Larger FFT = better frequency resolution
)
```

The script is now ready to provide comprehensive timbral analysis of your audio files!