Testing Plan for baselineARPLss package (DATA501)

https://posit.cloud/content/8810333

install.packages('remotes') library(remotes) remotes::install_github("econdatatech/baselineARPLss") library(baselineARPLss) data("Abelsonite")

res<-baseline_estimation(Abelsonite$measurement)

1. Package Structure and Functionality Objective: Ensure all functions in the package perform as intended. Tests: Check that all exported functions are accessible. Validate the internal functions (if any) for their specific roles. Confirm that help files and documentation for each function are complete.
2. Baseline Estimation Methods Objective: Verify the correct operation of the core baseline estimation algorithms. Tests: Test each baseline estimation method (e.g., polynomial fitting, smoothing) against known inputs and expected outputs. Ensure the accuracy of the results by comparing against standard or reference Raman spectra with known baselines. Simulate spectra with different noise levels, peak structures, and baselines to evaluate how well the methods perform under different scenarios.
3. Input Validation Objective: Confirm that functions handle different types of input data appropriately. Tests: Test handling of valid inputs such as numeric vectors representing Raman spectra. Provide invalid inputs (e.g., missing values, non-numeric data, inappropriate dimensions) and check for informative error messages. Test behavior with edge cases like empty spectra or spectra with only a few data points.
4. Performance and Efficiency Objective: Ensure the package performs efficiently for large datasets. Tests: Time the execution of baseline estimation for both small and large datasets. Test memory usage and confirm the package does not excessively increase memory consumption for large Raman spectra. Evaluate multi-threading or parallel processing capabilities if implemented.
5. Integration with Other Packages Objective: Check compatibility with other R packages and formats. Tests: Confirm compatibility with input formats from popular Raman spectroscopy data processing packages (e.g., hyperSpec, ggplot2). Ensure compatibility with popular export formats (e.g., .csv, .txt) for saving results.
6. User Interface and Usability Objective: Validate that the package provides a user-friendly interface. Tests: Test that all functions have clear, user-friendly error and warning messages. Check that the documentation covers all major use cases and contains examples. Ensure that the package can be installed and loaded without errors.
7. Unit Tests Objective: Implement automated testing for each function. Tests: Use the testthat package to write unit tests for each function, ensuring correctness for a variety of test cases. Validate that all tests pass after any code modifications. Test edge cases, typical cases, and stress the package with difficult spectra (e.g., highly noisy data).
8. Statistical and Numerical Validity Objective: Confirm that the underlying mathematical and statistical methods are correctly implemented. Tests: Compare the baseline estimation results with known theoretical models or other widely used software. Check numerical stability by testing spectra with large or very small values and ensure the methods produce consistent results.
9. Visualization (if applicable) Objective: Verify the visualization tools for spectra and baseline correction (if part of the package). Tests: Ensure that plots of raw spectra and corrected spectra (after baseline subtraction) display correctly. Test interaction between the package's plot functions and standard R visualization packages like ggplot2.
10. Continuous Integration Objective: Set up automated testing to ensure long-term code reliability. Tests: Implement a continuous integration (CI) service such as GitHub Actions or Travis CI to automatically run tests with each commit. Include tests for multiple R versions to ensure compatibility across environments.