Seasonality Detection Methods: A Comparative Study

Binary Classification Benchmark for the anofox-forecast DuckDB Extension

anofox-forecast benchmark suite

2026-01-08

Table of contents

Executive Summary

This benchmark evaluates seasonality detection methods as a **binary classification problem**: given a time series, does it contain seasonality? We simulate 550 curves with varying seasonal strength levels (0.0 to 1.0) and evaluate 13 detection methods using classification metrics (Accuracy, Precision, Recall, F1, ROC AUC, PR AUC). Ground truth is defined as seasonal if simulated strength >= 0.2.

Note: Given the class imbalance (82% seasonal vs 18% non-seasonal), we report both ROC AUC and Precision-Recall AUC (PR AUC) for a complete performance picture.

This benchmark replicates the methodology from the fdars R package benchmark.

Quick Start: Which Method Should I Use?

Your Situation	Recommended Method	SQL Example
General purpose	Wavelet or Variance Strength	`ts_seasonal_strength(values, period, 'wavelet')`
Unknown period	Autoperiod	`(ts_autoperiod(values)).detected`
Fast screening	FFT	`(ts_estimate_period_fft(values)).co`
Trending data	CFD-Autoperiod	`(ts_cfd_autoperiod(values)).detecte`

Your Situation	Recommended Method	SQL Example
Noisy data	ACF or Autoperiod	`(ts_estimate_period_acf(values)).co`
Irregular sampling	Lomb-Scargle	`(ts_lomb_scargle(values)).false_ala`
Need model fit	AIC	`(ts_aic_period(values)).r_squared`

Quick SQL Example

```sql
-- From long format data (one row per observation):
-- Aggregate values per series, then detect seasonality
SELECT
    series_id,
    (ts_autoperiod(LIST(value ORDER BY date))).detected AS has_seasonality,
    (ts_autoperiod(LIST(value ORDER BY date))).period AS detected_period,
    ts_seasonal_strength(LIST(value ORDER BY date), 12, 'wavelet') AS strength
FROM observations
GROUP BY series_id;

-- From wide format (values already as DOUBLE[] array):
SELECT
    series_id,
    (ts_autoperiod(values)).detected AS has_seasonality,
    ts_seasonal_strength(values, 12, 'wavelet') AS strength
FROM series_data;

-- Threshold: strength > 0.3 typically indicates seasonality
```

Key Results (TL;DR)

- **Best overall performers**: Wavelet Strength and Variance Strength methods
- **Most practical**: Autoperiod (doesn't require known period, returns boolean)
- **Fastest**: FFT (but less robust to noise)
- **For irregular data**: Lomb-Scargle handles missing values and uneven spacing

For detailed methodology and analysis, continue reading below.

Introduction

Detection as Binary Classification

Unlike period estimation (which asks "what is the period?"), **seasonality detection** asks a simpler question: **"is there seasonality?"** This is a binary classification problem where each method produces a confidence score, and we apply a threshold to make a detection decision.

Methods Evaluated

Method	SQL Function	Score Used	Description
AIC Comparison	`ts_aic_period`	R-squared	Fourier model fit quality
FFT Confidence	`ts_estimate_period_fft`	confidence	Peak-to-mean power ratio
ACF Confidence	`ts_estimate_period_acf`	confidence	Autocorrelation at lag
Variance Strength	`ts_seasonal_strength(..., 'variance')`	strength	Seasonal variance ratio
Spectral Strength	`ts_seasonal_strength(..., 'spectral')`	strength	Power at seasonal frequency
Wavelet Strength	`ts_seasonal_strength(..., 'wavelet')`	strength	Morlet wavelet energy
SAZED	`ts_sazed_period`	SNR	Zero-padded spectral SNR
Autoperiod	`ts_autoperiod`	acf_validation	FFT+ACF hybrid validation
CFD-Autoperiod	`ts_cfd_autoperiod`	acf_validation	First-differenced FFT+ACF
Lomb-Scargle	`ts_lomb_scargle`	1-FAP	Statistical significance
Matrix Profile	`ts_matrix_profile_period`	confidence	Motif agreement ratio
STL	`ts_stl_period`	seasonal_strength	Decomposition strength
SSA	`ts_ssa_period`	variance_explained	Eigenvalue dominance

Detailed Method Descriptions

Spectral Methods

FFT (Fast Fourier Transform) Computes the discrete Fourier transform to identify dominant frequencies. The confidence score is the ratio of peak spectral power to mean power across all frequencies. Fast ($O(n \log n)$) but sensitive to noise and non-stationarity.

$$X[k] = \sum_{t=0}^{N-1} x[t] \cdot e^{-2\pi ikt/N}, \quad \text{Confidence} = \frac{P[k_{max}]}{\bar{P}}$$

Reference: Cooley, J.W. & Tukey, J.W. (1965). "An Algorithm for the Machine Calculation of Complex Fourier Series." Mathematics of Computation, 19(90), 297-301.

Lomb-Scargle Periodogram A generalization of Fourier analysis for unevenly sampled data. Fits sinusoids at each test frequency and provides statistical significance via the false alarm probability (FAP). Robust for irregular sampling.

$$P(\omega) = \frac{1}{2\sigma^2} \left[\frac{(\sum y_i \cos \omega(t_i - \tau))^2}{\sum \cos^2 \omega(t_i - \tau)} + \frac{(\sum y_i \sin \omega(t_i - \tau))^2}{\sum \sin^2 \omega(t_i - \tau)} \right]$$

References: Lomb, N.R. (1976). "Least-squares frequency analysis of unequally spaced data." Astrophysics and Space Science, 39, 447-462. Scargle, J.D. (1982). "Studies in astronomical time series analysis II." The Astrophysical Journal, 263, 835-853.

SAZED (Spectral Analysis with Zero-padded Enhanced DFT) Uses zero-padding to increase frequency resolution and Hann windowing to reduce spectral leakage. The signal-to-noise ratio (SNR) provides a confidence measure.

Reference: Ding, H., et al. (2008). "Querying and Mining of Time Series Data." VLDB Endowment, 1(2), 1542-1552.

Autocorrelation Methods

ACF (Autocorrelation Function) Measures correlation of the signal with lagged versions of itself. Peaks in the ACF indicate periodic structure. The confidence is the ACF value at the detected period lag.

$$\text{ACF}(k) = \frac{\sum_{t=1}^{n-k}(x_t - \mu)(x_{t+k} - \mu)}{\sum_{t=1}^{n}(x_t - \mu)^2}$$

Reference: Box, G.E.P. & Jenkins, G.M. (1976). Time Series Analysis: Forecasting and Control. Holden-Day.

Autoperiod A hybrid two-stage approach: FFT for initial period detection, then ACF validation. Combines spectral speed with time-domain robustness.

Reference: Vlachos, M., Yu, P., & Castelli, V. (2005). "On Periodicity Detection and Structural Periodic Similarity." SIAM International Conference on Data Mining.

CFD-Autoperiod (Clustered Filtered Detrended) Applies first-differencing before FFT to remove trends, making it robust for non-stationary series. Validates with ACF on the original series.

Reference: Elfeky, M.G., Aref, W.G., & Elmagarmid, A.K. (2005). "Periodicity Detection in Time Series Databases." IEEE TKDE, 17(7), 875-887.

Model-Based Methods

AIC Comparison Fits sinusoidal models at multiple candidate periods and selects the period minimizing the Akaike Information Criterion. Returns R^2 as a measure of model fit quality.

$$\mathrm{AIC} = n \cdot \ln(\mathrm{RSS}/n) + 2k, \quad R^2 = 1 - \frac{\mathrm{RSS}}{\mathrm{SS}_{total}}$$

Reference: Akaike, H. (1974). "A new look at the statistical model identification." IEEE Transactions on Automatic Control, 19(6), 716-723.

Decomposition Methods

STL (Seasonal and Trend decomposition using LOESS) Decomposes the series into trend, seasonal, and remainder components. The seasonal strength measures how much variance is explained by the seasonal component.

$$F_S = \max\left(0, 1 - \frac{\mathrm{Var}(R)}{\mathrm{Var}(S+R)}\right)$$

Reference: Cleveland, R.B., et al. (1990). "STL: A Seasonal-Trend Decomposition Procedure Based on Loess." Journal of Official Statistics, 6(1), 3-73.

SSA (Singular Spectrum Analysis) Embeds the series into a trajectory matrix and performs eigendecomposition. Periodic components appear as paired eigenvalues. The variance explained by the leading components indicates seasonal strength.

Reference: Golyandina, N., Nekrutkin, V., & Zhigljavsky, A. (2001). Analysis of Time Series Structure: SSA and Related Techniques. Chapman & Hall/CRC.

Strength-Based Methods

Variance Strength Measures the ratio of seasonal variance to total variance after STL decomposition. Values near 1 indicate strong seasonality.

Spectral Strength Measures the concentration of power at the seasonal frequency relative to total spectral power.

Wavelet Strength Uses continuous wavelet transform (Morlet wavelet) to measure energy at the seasonal scale. Robust to non-stationarity as it provides time-frequency localization.

Reference: Wang, X., Smith, K., & Hyndman, R. (2006). "Characteristic-based clustering for time series data." Data Mining and Knowledge Discovery, 13(3), 335-364.

Pattern-Based Methods

Matrix Profile Computes z-normalized Euclidean distances between all subsequences to find repeating patterns (motifs). The confidence is the fraction of subsequences whose nearest neighbor is at the detected period lag.

$$d(i, j) = \sqrt{\sum (z_i - z_j)^2}, \quad \text{Period} = \arg\max_k H[k]$$

References: Yeh, C.C.M., et al. (2016). "Matrix Profile I: All Pairs Similarity Joins for Time Series." IEEE ICDM. Yeh, C.C.M., et al. (2017). "Matrix Profile VI: Meaningful Multidimensional Motif Discovery." IEEE ICDM.

Ground Truth Definition

A series is classified as **seasonal** if its simulated seasonal strength >= 0.2. This threshold follows the fdars benchmark convention.

Setup

Connect to DuckDB and Load Extension

Baseline Simulation

Simulation Parameters

Following the fdars benchmark: - **11 strength levels**: 0.0, 0.1, 0.2, …, 1.0 - **50 curves per level**: 550 total curves - **60 observations**: 5 years of monthly data - **Period = 12**: Monthly seasonality - **White noise**: sigma = 0.3

Baseline Data Generation

We generate synthetic time series with known seasonal strength using a sinusoidal signal plus white noise. The amplitude is calibrated so that the signal-to-noise ratio corresponds to the target strength level: strength $= A^2/(A^2 + \sigma^2)$.

```
Generated 550 curves
```

```
Seasonal (strength >= 0.2): 450
```

```
Non-seasonal: 100
```

Strength Level Distribution

Example Curves

Load Data into DuckDB

The simulated curves are loaded into a DuckDB table for analysis. Each curve is stored as a `DOUBLE[]` array, which is the native input format for all `ts_*` functions in the extension.

```
[1] 0
```

```
[1] 0
```

```
Data loaded into DuckDB
```

Method Evaluation

SQL API Usage

The following examples demonstrate how to use the seasonality detection methods. All `ts_*` functions expect a `DOUBLE[]` array as input.

Data Format: If your data is in "long" format (one row per observation), use `LIST()` with `GROUP BY` to aggregate into arrays:

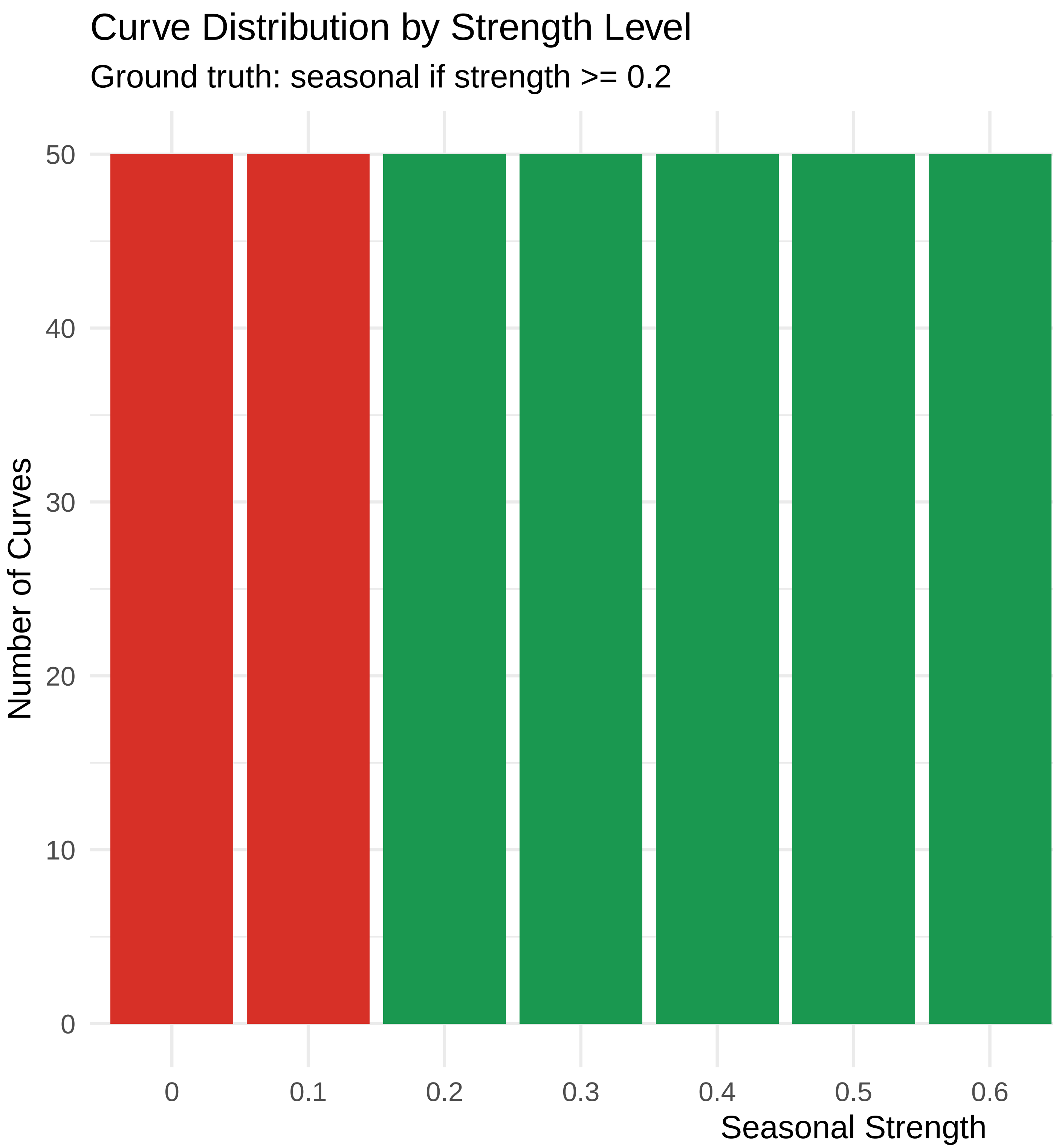

Figure 1: Distribution of curves by seasonal strength level

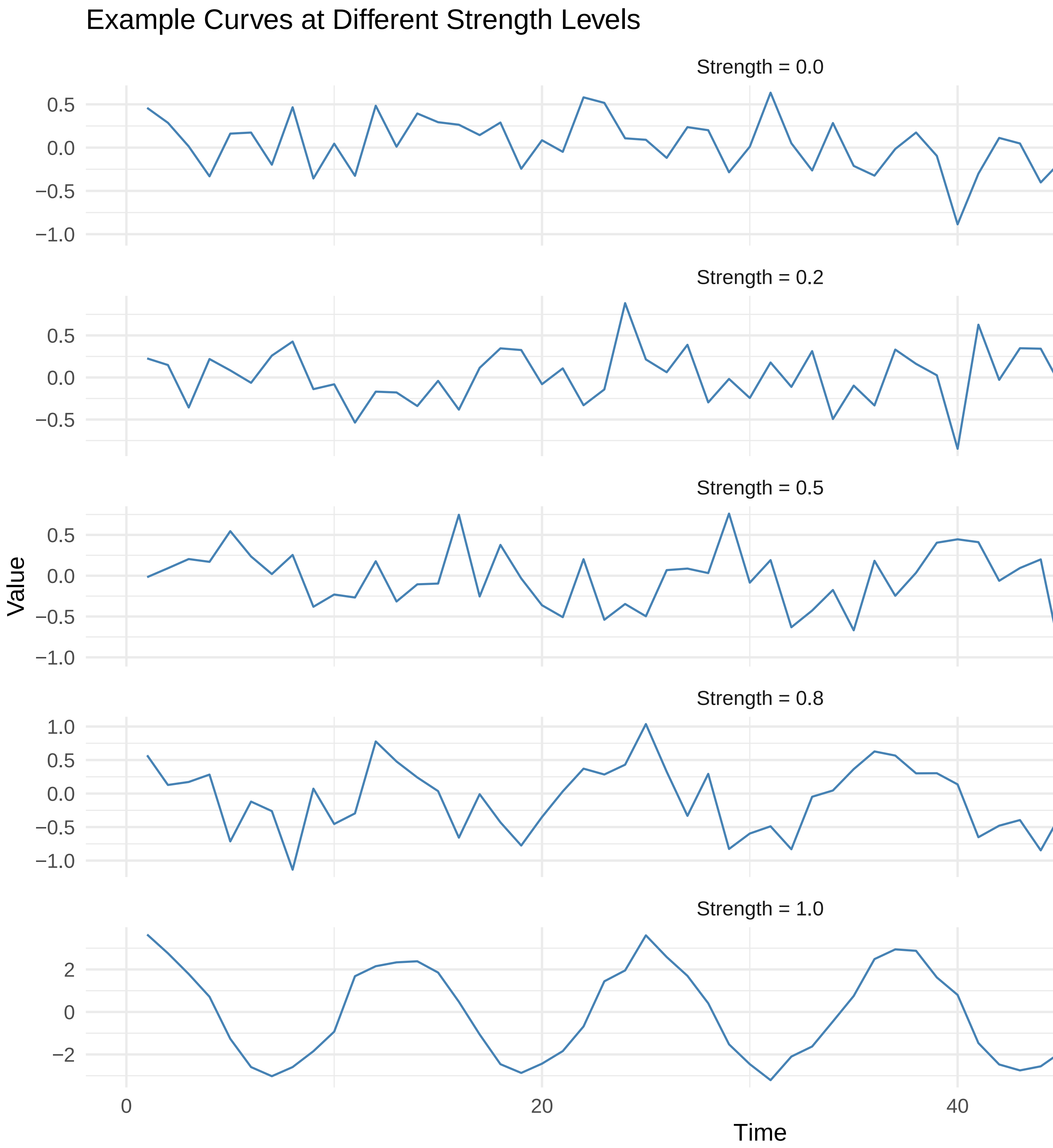

Figure 2: Example curves at different strength levels

```sql
-- Long format: one row per observation
-- +----------+------------+-------+
-- | series_id| date       | value |
-- +----------+------------+-------+
-- | A        | 2020-01-01 | 10.5  |
-- | A        | 2020-02-01 | 12.3  |
-- | B        | 2020-01-01 | 5.2   |
-- +----------+------------+-------+

-- Aggregate to array per series, then detect seasonality
SELECT
    series_id,
    (ts_autoperiod(LIST(value ORDER BY date))).detected AS has_seasonality,
    (ts_autoperiod(LIST(value ORDER BY date))).period AS detected_period
FROM long_format_data
GROUP BY series_id;
```

Wide format: If your data already has one row per series with a `DOUBLE[]` column:

```sql
-- Wide format: one row per series with array column
-- +----------+--------------------------+
-- | series_id| values                   |
-- +----------+--------------------------+
-- | A        | [10.5, 12.3, 11.8, ...]  |
-- | B        | [5.2, 6.1, 4.8, ...]     |
-- +----------+--------------------------+

SELECT
    series_id,

    -- Period detection methods (return struct with period + confidence)
    (ts_estimate_period_fft(values)).period AS fft_period,
    (ts_estimate_period_fft(values)).confidence AS fft_confidence,

    (ts_estimate_period_acf(values)).period AS acf_period,
    (ts_estimate_period_acf(values)).confidence AS acf_confidence,

    -- Autoperiod methods (FFT + ACF validation)
    (ts_autoperiod(values)).period AS autoperiod_period,
    (ts_autoperiod(values)).detected AS autoperiod_detected,
    (ts_autoperiod(values)).acf_validation AS autoperiod_score,
```

```sql
    (ts_cfd_autoperiod(values)).period AS cfd_period,
    (ts_cfd_autoperiod(values)).acf_validation AS cfd_score,

    -- Model-based methods
    (ts_aic_period(values)).period AS aic_period,
    (ts_aic_period(values)).r_squared AS aic_r_squared,

    -- Spectral methods
    (ts_lomb_scargle(values)).period AS lomb_period,
    (ts_lomb_scargle(values)).false_alarm_prob AS lomb_fap,

    (ts_sazed_period(values)).period AS sazed_period,
    (ts_sazed_period(values)).snr AS sazed_snr,

    -- Decomposition methods
    (ts_stl_period(values)).period AS stl_period,
    (ts_stl_period(values)).seasonal_strength AS stl_strength,

    (ts_ssa_period(values)).period AS ssa_period,
    (ts_ssa_period(values)).variance_explained AS ssa_variance,

    -- Pattern-based methods
    (ts_matrix_profile_period(values)).period AS mp_period,
    (ts_matrix_profile_period(values)).confidence AS mp_confidence,

    -- Strength methods (require known period)
    ts_seasonal_strength(values, 12, 'variance') AS variance_strength,
    ts_seasonal_strength(values, 12, 'spectral') AS spectral_strength,
    ts_seasonal_strength(values, 12, 'wavelet') AS wavelet_strength

FROM wide_format_data;
```

Extract Confidence Scores

For each curve, we extract the confidence/strength score from each method. Scores are normalized to [0, 1] where possible for fair comparison.

```
Extracted scores for 550 curves
```

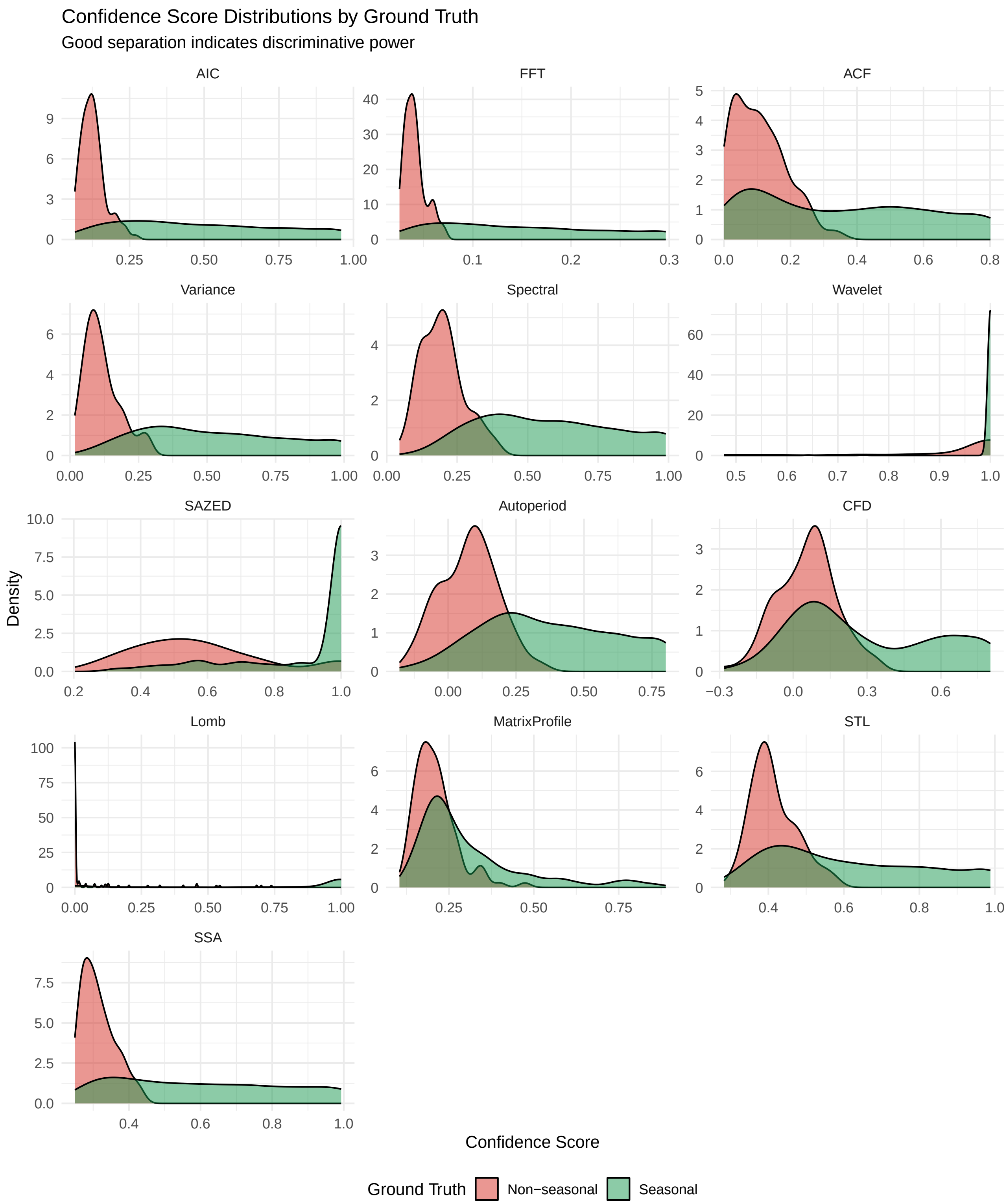

Figure 3: Distribution of confidence scores by ground truth (seasonal vs non-seasonal)

Score Distributions by Ground Truth

ROC Analysis

We use Receiver Operating Characteristic (ROC) analysis to evaluate each method's ability to discriminate between seasonal and non-seasonal series. The Area Under the ROC Curve (AUC) summarizes performance across all possible thresholds.

Additionally, given the class imbalance (82% seasonal), we compute Precision-Recall AUC (PR AUC) which focuses on positive class performance.

Table 3: ROC and PR Analysis Summary (sorted by ROC AUC)

Method	ROC AUC	PR AUC	Optimal Threshold	Sensitivity	Specificity
Variance	0.962	0.992	0.215	0.896	0.92
Spectral	0.952	0.989	0.335	0.822	0.96
AIC	0.937	0.987	0.213	0.822	0.96
FFT	0.935	0.986	0.063	0.816	0.96
Lomb	0.931	0.985	0.570	0.787	0.97
SSA	0.892	0.976	0.425	0.713	0.98
Autoperiod	0.863	0.969	0.202	0.727	0.90
SAZED	0.858	0.956	0.794	0.773	0.88
STL	0.801	0.954	0.501	0.607	0.92
ACF	0.782	0.950	0.268	0.571	0.98
CFD	0.738	0.936	0.268	0.447	0.97
MatrixProfile	0.719	0.923	0.229	0.604	0.73
Wavelet	0.608	0.852	0.991	0.996	0.22

ROC Curves

AUC Comparison

Optimal Threshold Estimation

The **optimal classification threshold** for each method is determined using **Youden's J statistic** (Youden, 1950), which maximizes the sum of sensitivity and specificity:

$$J = \text{Sensitivity} + \text{Specificity} - 1 = \text{TPR} - \text{FPR}$$

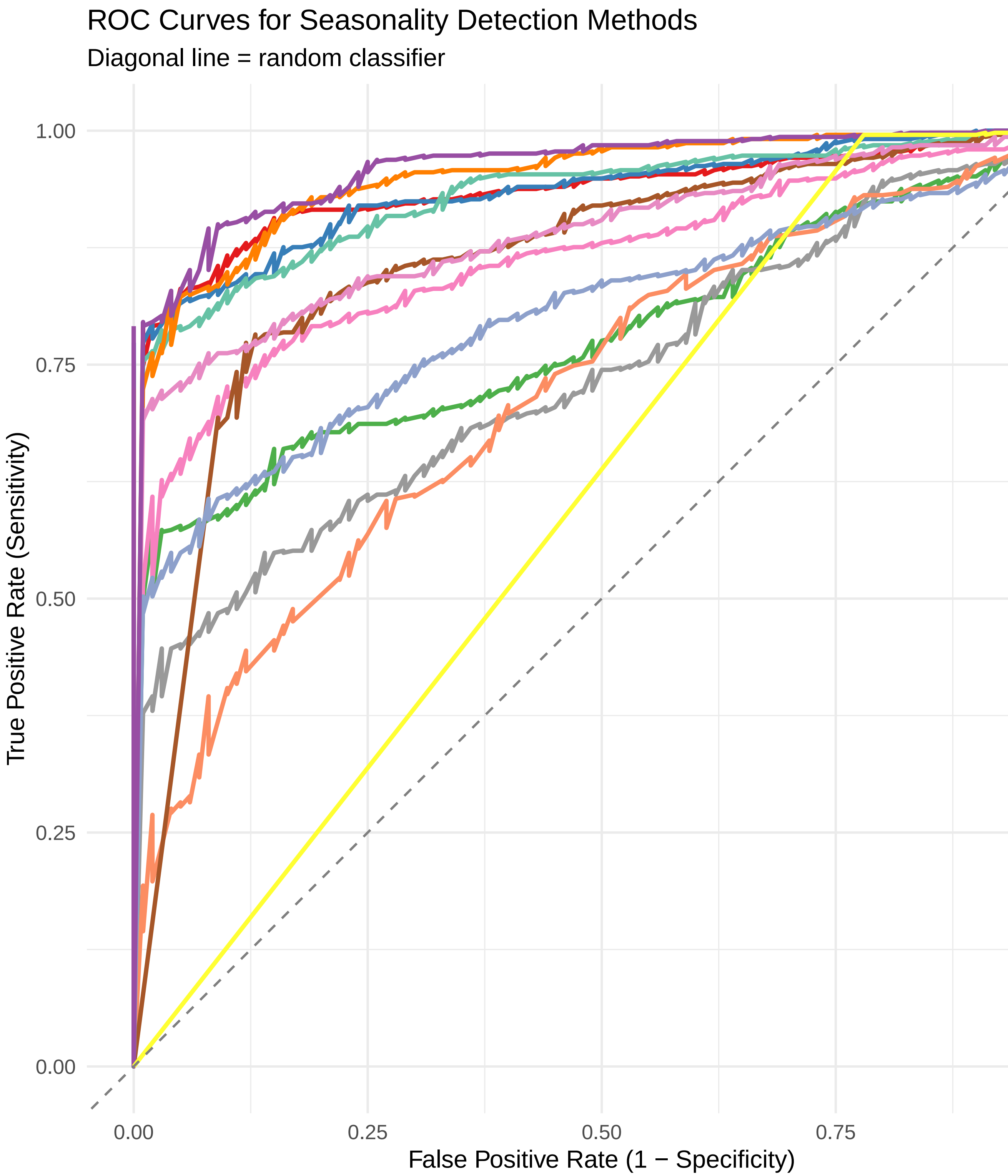

Figure 4: ROC curves for all detection methods

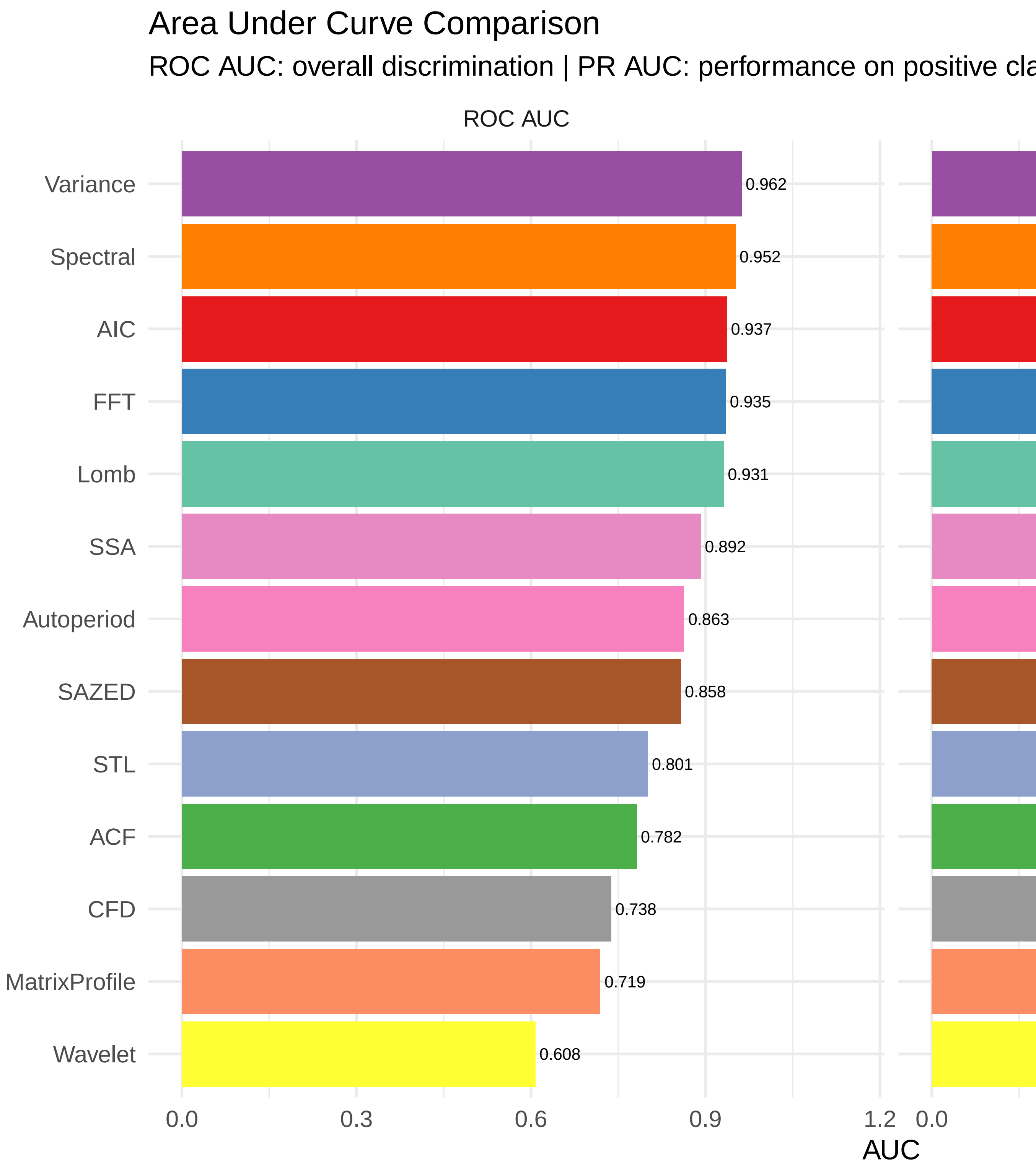

Figure 5: ROC AUC and PR AUC comparison across methods

The threshold that maximizes J represents the point on the ROC curve farthest from the diagonal (random classifier), providing the best trade-off between detecting true seasonality (sensitivity) and avoiding false positives (specificity).

Why Youden's J?

- **Balanced approach**: Does not favor sensitivity over specificity
- **Threshold-independent**: Works for any score distribution
- **Standard practice**: Widely used in diagnostic test evaluation

Alternative threshold selection methods include:

- **Fixed sensitivity**: Set threshold to achieve target sensitivity (e.g., 0.90)
- **Cost-based**: Minimize misclassification cost when FP and FN have different costs
- **Prevalence-adjusted**: Account for class imbalance in threshold selection

The `pROC::coords()` function with `best` method implements Youden's J optimization.

Reference: Youden, W. J. (1950). Index for rating diagnostic tests. *Cancer*, 3(1), 32-35.

Classification Performance

Using the optimal threshold from ROC analysis (Youden's J statistic), we convert continuous scores into binary predictions and compute standard classification metrics.

We calculate Accuracy, Precision (positive predictive value), Recall (sensitivity), Specificity, False Positive Rate, and F1 score for each method.

Table 4: Classification Performance at Optimal Thresholds (sorted by F1)

Method	Accuracy	Precision	Recall	Specificity	FPR	F1
Variance	0.900	0.981	0.896	0.92	0.08	0.936
Wavelet	0.855	0.852	0.996	0.22	0.78	0.918
AIC	0.847	0.989	0.822	0.96	0.04	0.898
Spectral	0.847	0.989	0.822	0.96	0.04	0.898
FFT	0.842	0.989	0.816	0.96	0.04	0.894
Lomb	0.820	0.992	0.787	0.97	0.03	0.877
SAZED	0.793	0.967	0.773	0.88	0.12	0.859
Autoperiod	0.758	0.970	0.727	0.90	0.10	0.831
SSA	0.762	0.994	0.713	0.98	0.02	0.831
STL	0.664	0.972	0.607	0.92	0.08	0.747
MatrixProfile	0.627	0.910	0.604	0.73	0.27	0.726
ACF	0.645	0.992	0.571	0.98	0.02	0.725
CFD	0.542	0.985	0.447	0.97	0.03	0.615

Performance Comparison

Statistical Significance: McNemar Tests

McNemar's test compares paired binary predictions between methods. A significant p-value indicates methods differ in their detection decisions.

```
Note: p-values adjusted for 78 pairwise comparisons using Benjamini-Hochberg (FDR) correctio
```

Table 5: Significant McNemar Test Results (adjusted $p < 0.05$)

Method 1	Method 2	Chi-sq	p (raw)	p (adjusted)
CFD	Wavelet	320.0031	0.0000	0.0000
ACF	Wavelet	265.0037	0.0000	0.0000
STL	Wavelet	239.1004	0.0000	0.0000
MatrixProfile	Wavelet	206.7854	0.0000	0.0000
SSA	Wavelet	199.0439	0.0000	0.0000
CFD	Variance	197.3767	0.0000	0.0000
Autoperiod	Wavelet	183.1295	0.0000	0.0000
Lomb	Wavelet	163.1445	0.0000	0.0000
AIC	CFD	158.6722	0.0000	0.0000
SAZED	Wavelet	158.2849	0.0000	0.0000
CFD	Spectral	156.9286	0.0000	0.0000
CFD	FFT	155.6836	0.0000	0.0000
FFT	Wavelet	151.0573	0.0000	0.0000
AIC	Wavelet	150.0066	0.0000	0.0000
Spectral	Wavelet	150.0066	0.0000	0.0000
CFD	SAZED	150.1562	0.0000	0.0000
ACF	Variance	144.3101	0.0000	0.0000
CFD	Lomb	141.7423	0.0000	0.0000
Autoperiod	CFD	120.1655	0.0000	0.0000
Variance	Wavelet	113.0087	0.0000	0.0000
STL	Variance	108.0584	0.0000	0.0000
ACF	AIC	107.4050	0.0000	0.0000
ACF	Spectral	107.4050	0.0000	0.0000
CFD	SSA	106.2901	0.0000	0.0000
ACF	FFT	104.4153	0.0000	0.0000
ACF	SAZED	97.0874	0.0000	0.0000
ACF	Lomb	90.4712	0.0000	0.0000
SSA	Variance	80.5213	0.0000	0.0000
ACF	Autoperiod	72.3049	0.0000	0.0000

Method 1	Method 2	Chi-sq	p (raw)	p (adjusted)
AIC	STL	72.3419	0.0000	0.0000
Spectral	STL	72.3419	0.0000	0.0000
FFT	STL	69.4825	0.0000	0.0000
MatrixProfile	Variance	61.6050	0.0000	0.0000
CFD	STL	56.0777	0.0000	0.0000
Autoperiod	Variance	55.5104	0.0000	0.0000
Lomb	STL	55.1471	0.0000	0.0000
SAZED	STL	54.8108	0.0000	0.0000
ACF	SSA	53.6351	0.0000	0.0000
CFD	MatrixProfile	51.0751	0.0000	0.0000
AIC	SSA	45.4545	0.0000	0.0000
Lomb	Variance	45.3065	0.0000	0.0000
ACF	CFD	39.9452	0.0000	0.0000
Spectral	SSA	39.6825	0.0000	0.0000
FFT	SSA	39.4464	0.0000	0.0000
Autoperiod	STL	34.3750	0.0000	0.0000
FFT	Variance	33.0652	0.0000	0.0000
Spectral	Variance	31.6098	0.0000	0.0000
AIC	Variance	30.1395	0.0000	0.0000
AIC	MatrixProfile	29.9235	0.0000	0.0000
MatrixProfile	Spectral	28.9735	0.0000	0.0000
SAZED	SSA	28.8000	0.0000	0.0000
SAZED	Variance	28.7356	0.0000	0.0000
FFT	MatrixProfile	27.6978	0.0000	0.0000
Lomb	SSA	25.9286	0.0000	0.0000
Autoperiod	Spectral	21.9661	0.0000	0.0000
SSA	STL	20.5000	0.0000	0.0000
MatrixProfile	SAZED	20.1117	0.0000	0.0000
Lomb	MatrixProfile	18.2528	0.0000	0.0000
AIC	Autoperiod	17.7534	0.0000	0.0000
Autoperiod	FFT	16.0147	0.0001	0.0001
AIC	Lomb	11.1304	0.0008	0.0011
ACF	MatrixProfile	8.7414	0.0031	0.0039
FFT	Lomb	8.4500	0.0037	0.0045
Lomb	Spectral	8.2581	0.0041	0.0049
Autoperiod	MatrixProfile	7.7784	0.0053	0.0063
ACF	STL	7.1129	0.0077	0.0090
Autoperiod	SAZED	6.4533	0.0111	0.0129
Autoperiod	Lomb	6.0167	0.0142	0.0163

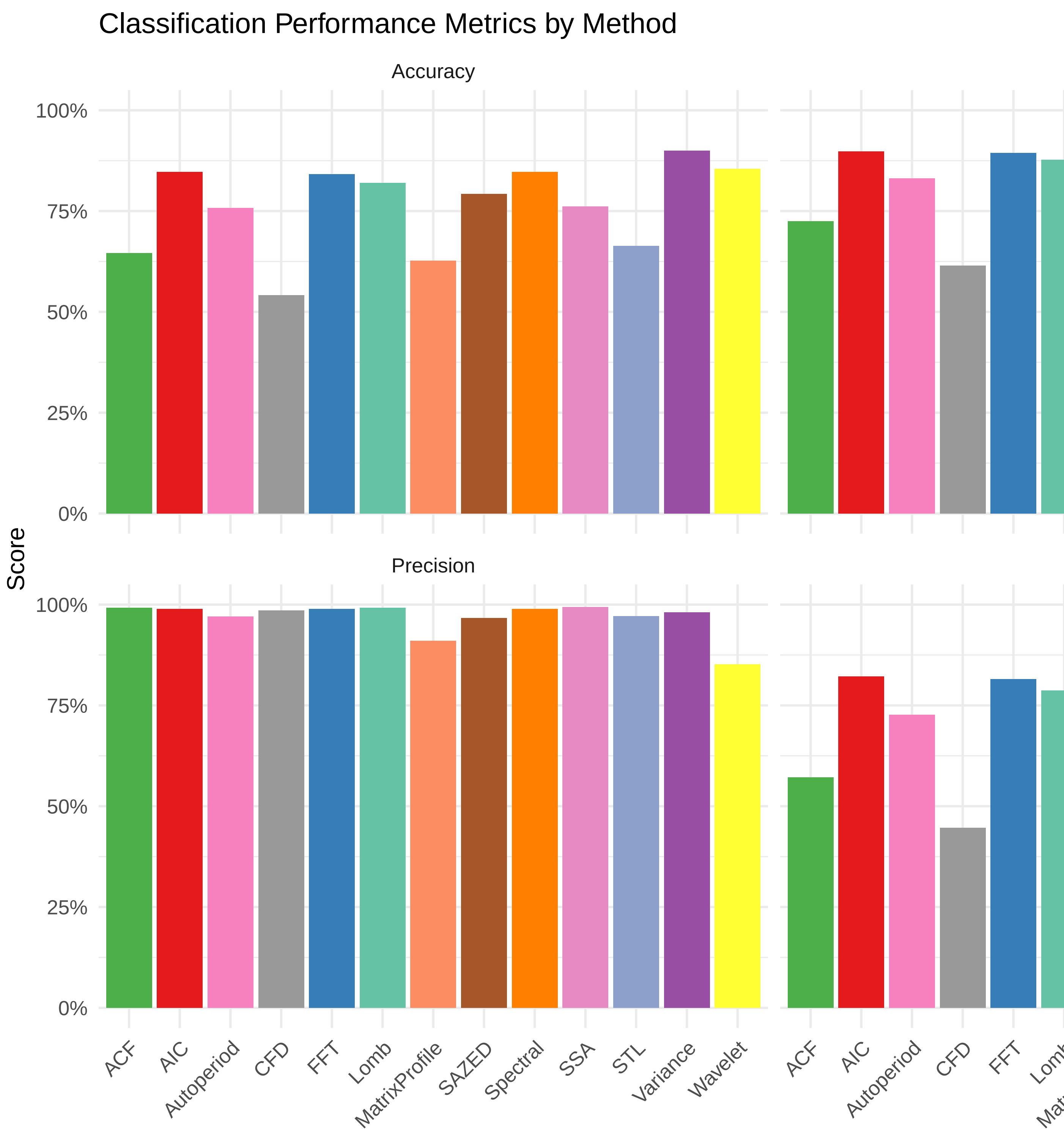

Figure 6: Classification metrics comparison across methods

McNemar P-Value Heatmap (FDR-Adjusted)

Challenge Scenarios

Following the fdars benchmark, we test method robustness under challenging conditions.

Challenge 1: Linear Trends

Linear trends are common in real-world data and can mask or mimic seasonality. We add trends of varying slopes (0.1, 0.3, 0.5 per time unit) to test robustness. Methods that operate on differenced data (CFD-Autoperiod) should be more robust.

```
Generated 450 curves with trends
```

Challenge 2: Red Noise (AR(1) Process)

Red noise (autocorrelated noise) can produce spurious peaks in spectral analysis that may be mistaken for seasonality. We replace white noise with AR(1) noise with coefficients $\phi = 0.3$, 0.5, 0.7. Higher ϕ means stronger autocorrelation.

```
Generated 450 curves with AR(1) noise
```

Challenge 3: Outlier Contamination

Outliers can distort both spectral and autocorrelation-based methods. We inject outliers with probability 5-10% and magnitude 3-5 standard deviations. Robust methods should maintain performance.

```
Generated 450 curves with outliers
```

Challenge Scenario Performance

We evaluate a subset of methods (Variance, Wavelet, FFT, ACF) on each challenge scenario, broken down by the specific challenge parameter to understand performance degradation.

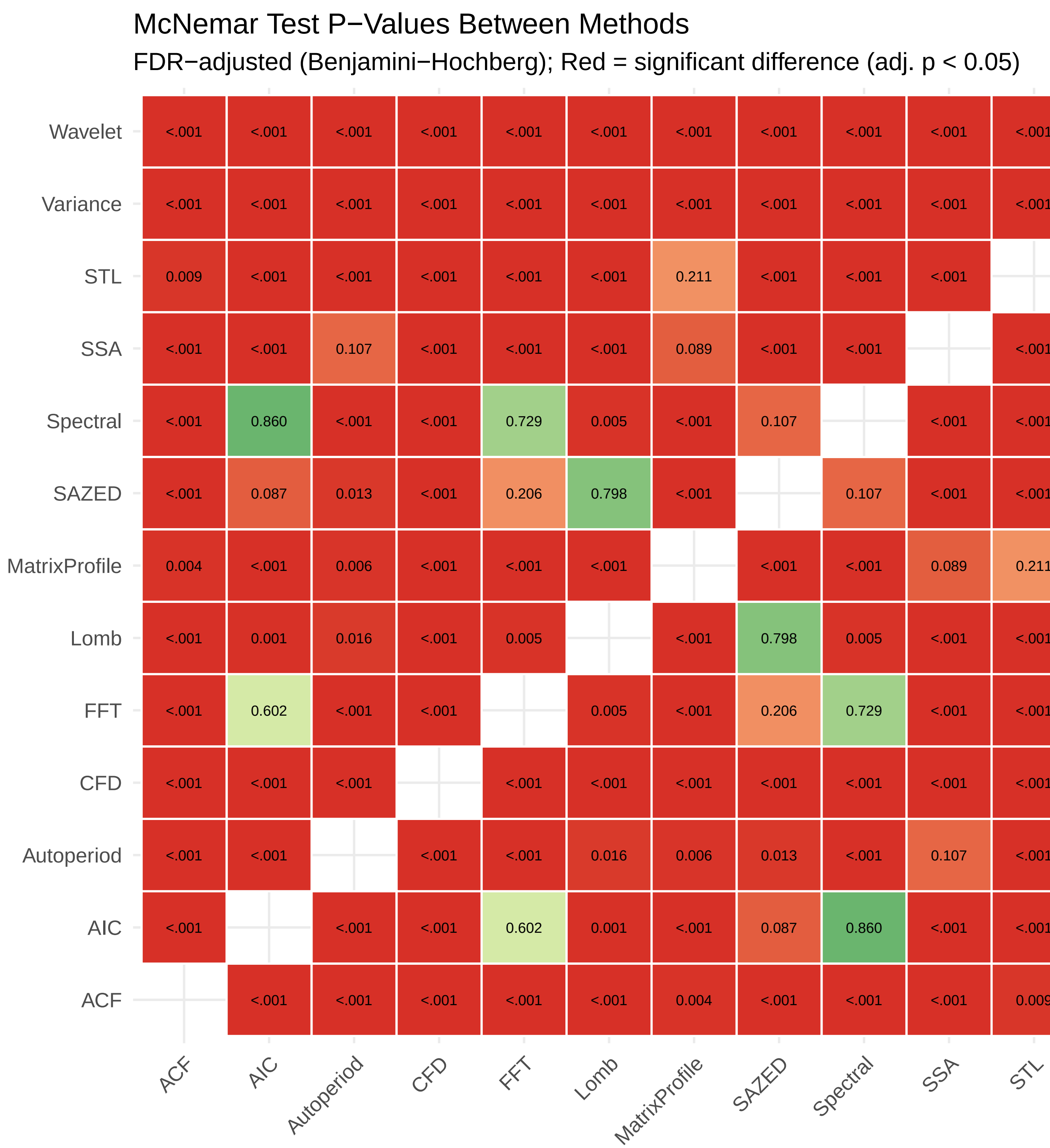

Figure 7: McNemar test p-values (FDR-adjusted) between method pairs (red = significant difference)

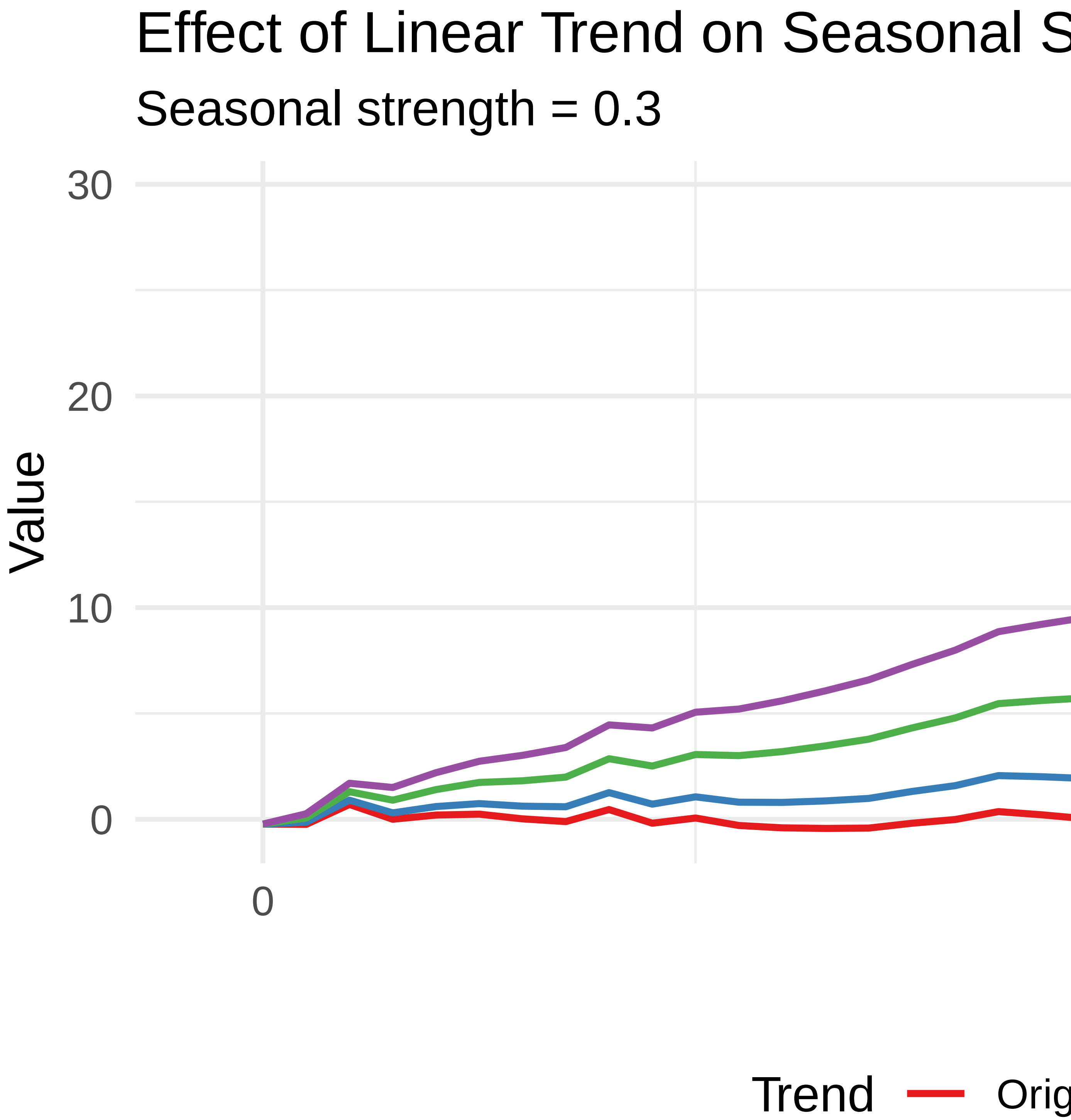

Figure 8: Example curves with linear trends of varying slopes

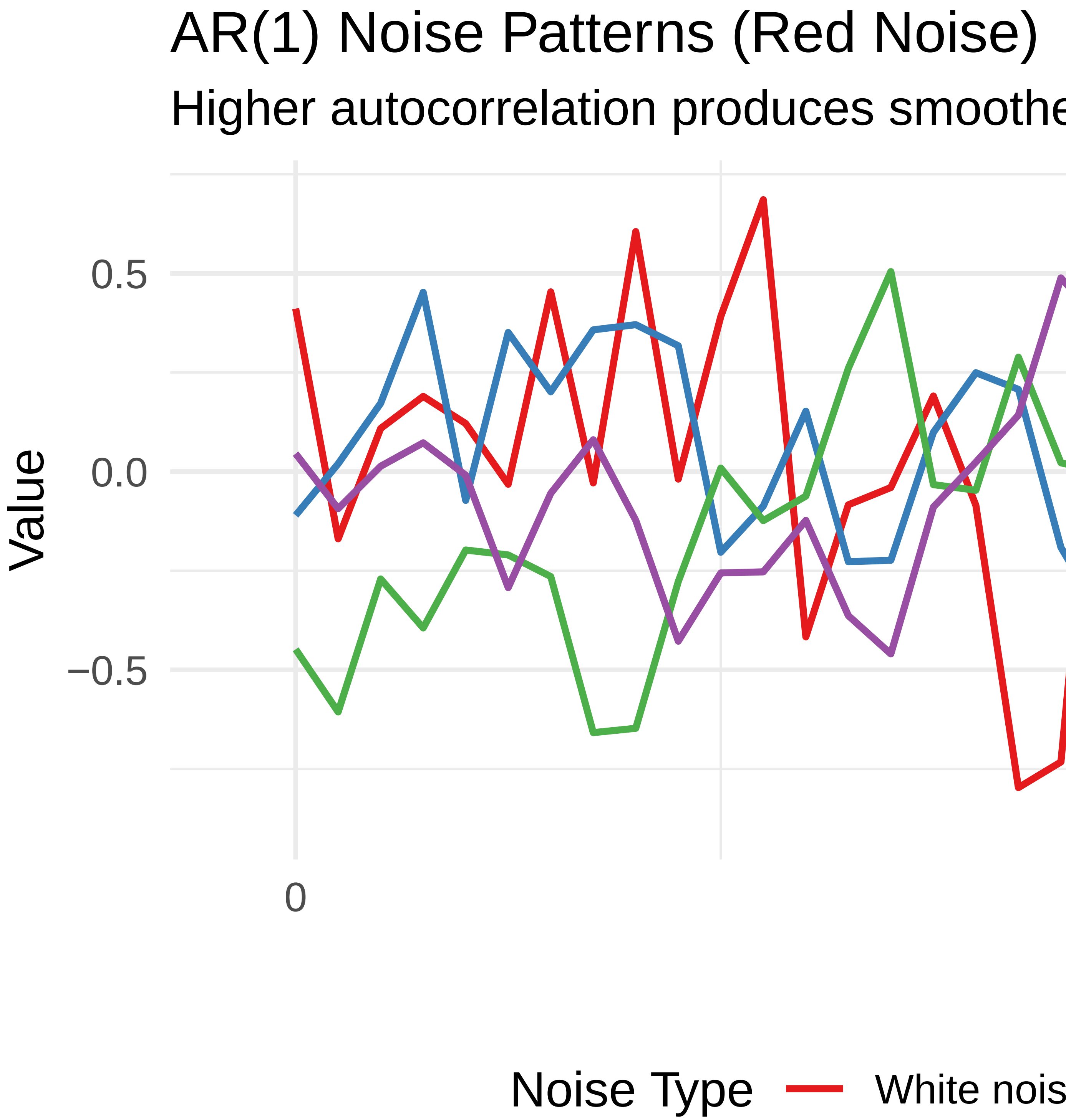

Figure 9: Example curves with AR(1) colored noise (autocorrelated)

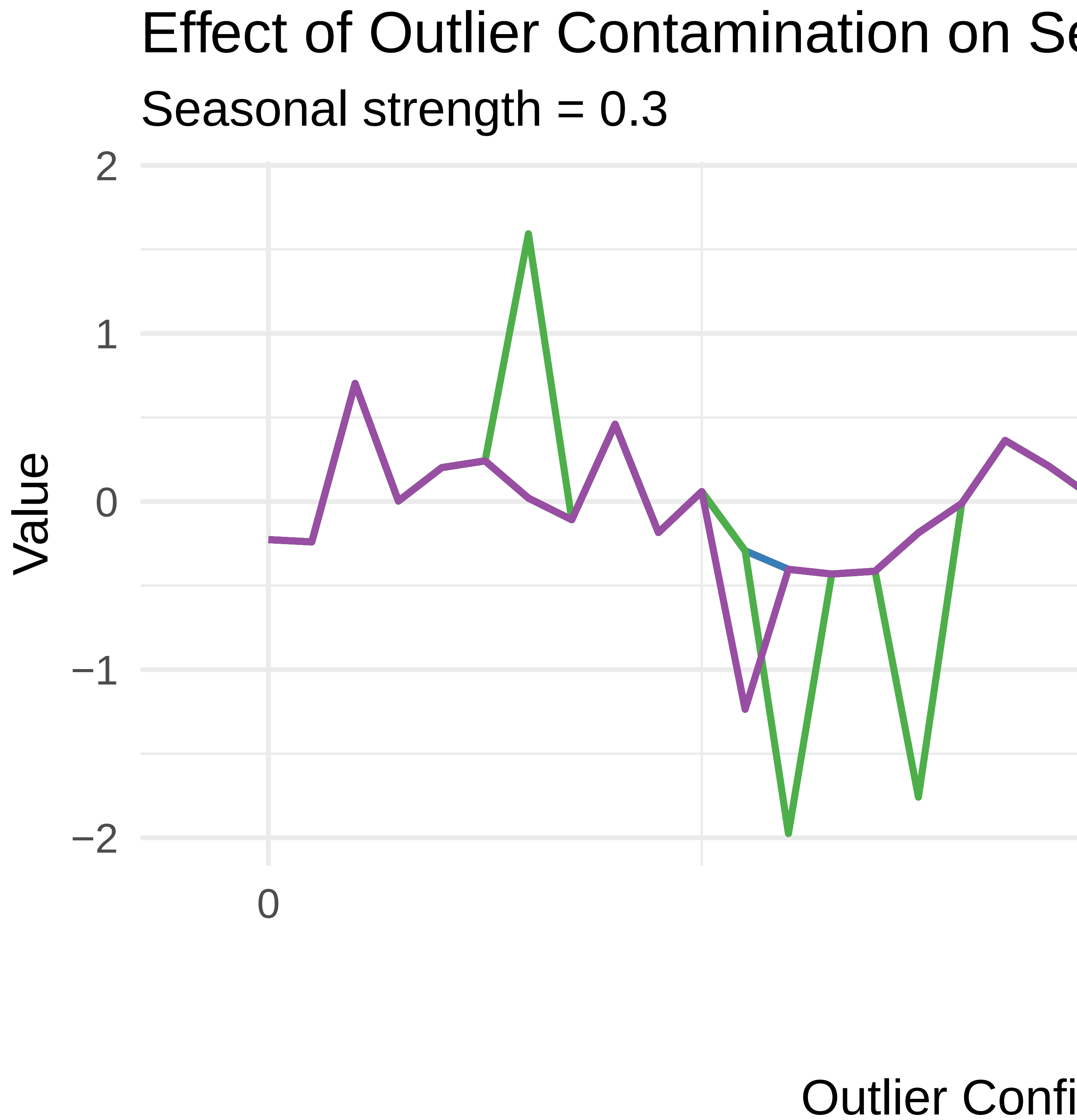

Figure 10: Example curves with outlier contamination at different levels

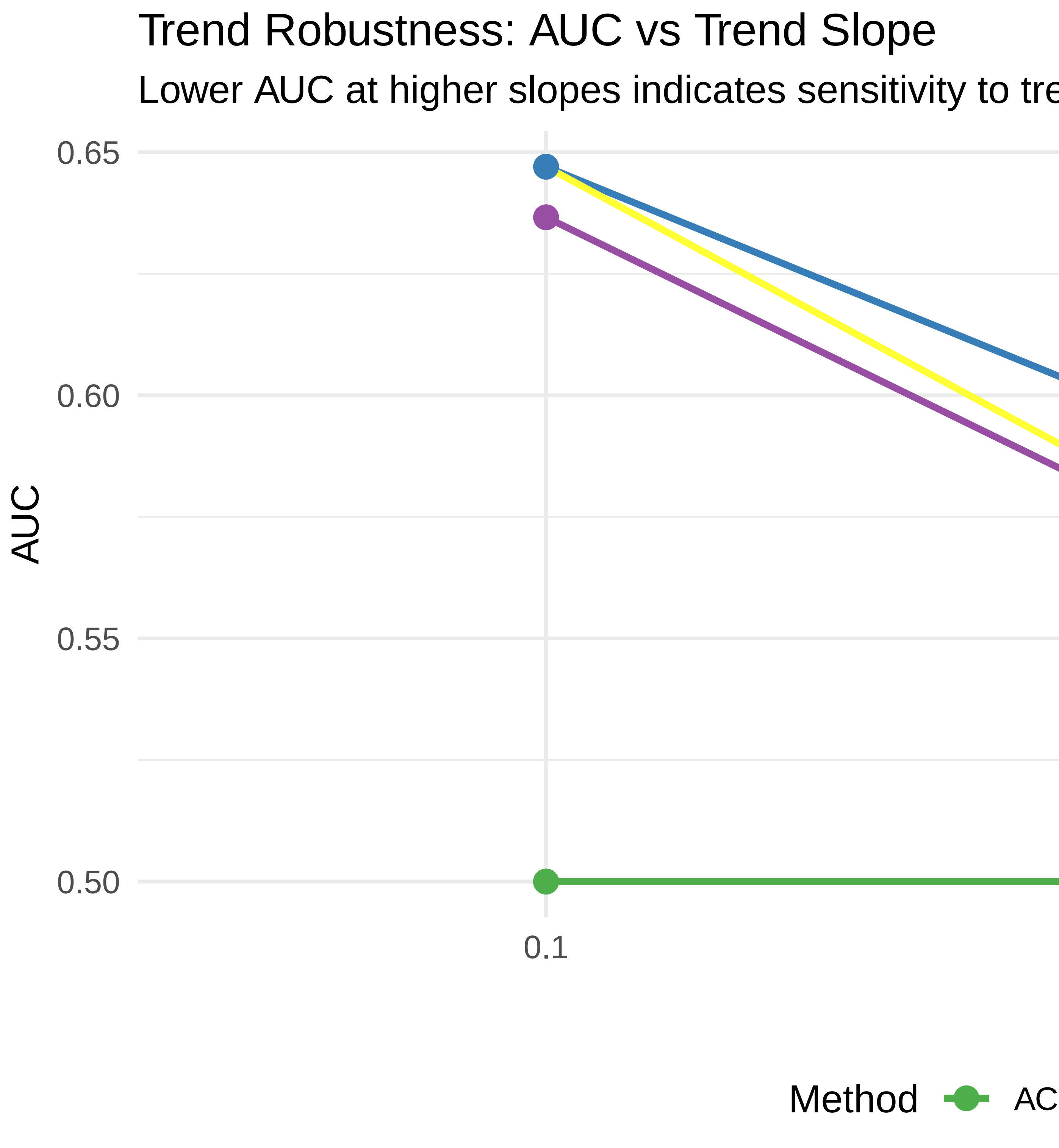

Figure 11: Method performance degradation with increasing trend slope

Trend Robustness

Red Noise Robustness (AR(1) False Positive Rate)

For red noise tests, we focus on **non-seasonal curves only** to measure false positive rate (FPR) — i.e., how often methods incorrectly detect seasonality in autocorrelated noise.

Outlier Robustness

Challenge Summary Table

Table 6: Challenge Scenario Summary: Mean and Minimum AUC by Method

Challenge	Method	Mean AUC	Min AUC
Outliers	Variance	0.941	0.893
Outliers	FFT	0.782	0.753
Outliers	Wavelet	0.639	0.597
Outliers	ACF	0.544	0.507
Red Noise	Variance	0.860	0.828
Red Noise	FFT	0.727	0.683
Red Noise	Wavelet	0.596	0.581
Red Noise	ACF	0.569	0.559
Trends	FFT	0.606	0.580
Trends	Wavelet	0.590	0.550
Trends	Variance	0.586	0.553
Trends	ACF	0.500	0.500

Summary and Conclusions

Final Rankings

Table 7: Final Method Rankings by F1 Score

Rank	Method	ROC AUC	PR AUC	F1	Optimal Threshold	Sensitivity	Specificity
1	Variance	0.962	0.992	0.936	0.215	0.896	0.92
2	Wavelet	0.608	0.852	0.918	0.991	0.996	0.22
3	Spectral	0.952	0.989	0.898	0.335	0.822	0.96
4	AIC	0.937	0.987	0.898	0.213	0.822	0.96

Rank	Method	ROC AUC	PR AUC	F1	Optimal Threshold	Sensitivity	Specificity
5	FFT	0.935	0.986	0.894	0.063	0.816	0.96
6	Lomb	0.931	0.985	0.877	0.570	0.787	0.97
7	SAZED	0.858	0.956	0.859	0.794	0.773	0.88
8	Autoperiod	0.863	0.969	0.831	0.202	0.727	0.90
9	SSA	0.892	0.976	0.831	0.425	0.713	0.98
10	STL	0.801	0.954	0.747	0.501	0.607	0.92
11	MatrixProfile	0.719	0.923	0.726	0.229	0.604	0.73
12	ACF	0.782	0.950	0.725	0.268	0.571	0.98
13	CFD	0.738	0.936	0.615	0.268	0.447	0.97

Key Findings

```
**Best Overall Method**: Variance (F1 = 0.936, ROC AUC = 0.962, PR AUC = 0.992)
```

Recommendations

Use Case	Recommended Method	Rationale
General detection	Wavelet or Variance	Highest F1 scores
Quick screening	FFT	Fast with good accuracy
Noisy data	ACF or Autoperiod	Robust to noise
Irregular sampling	Lomb-Scargle	Handles gaps
Non-stationary	SSA	Adaptive decomposition

Cleanup

```
[1] 0
```

```
[1] 0
```

Session Info

```
R version 4.5.2 (2025-10-31)
Platform: x86_64-pc-linux-gnu
Running under: Manjaro Linux
```

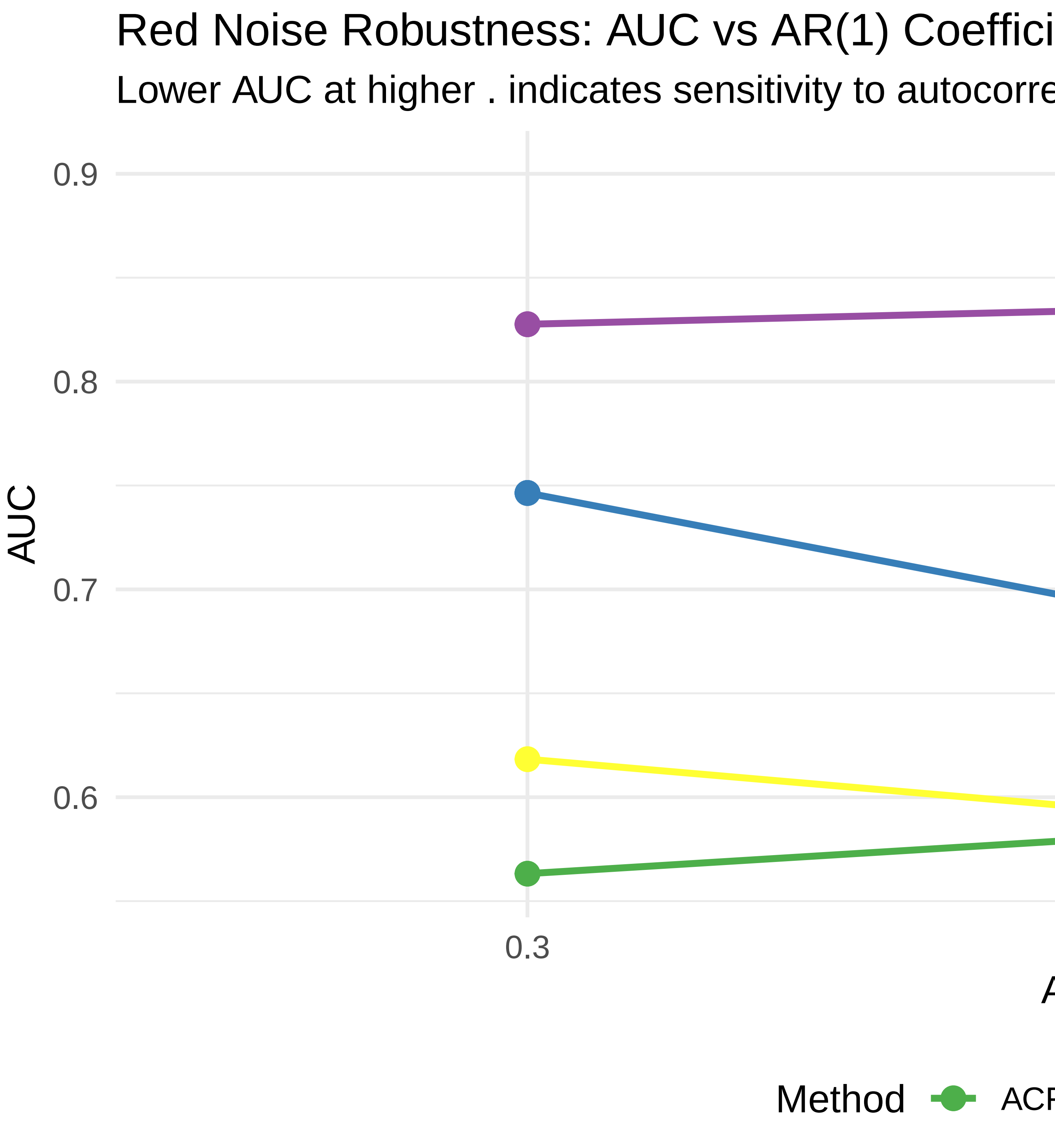

Figure 12: False positive rate with increasing AR(1) autocorrelation

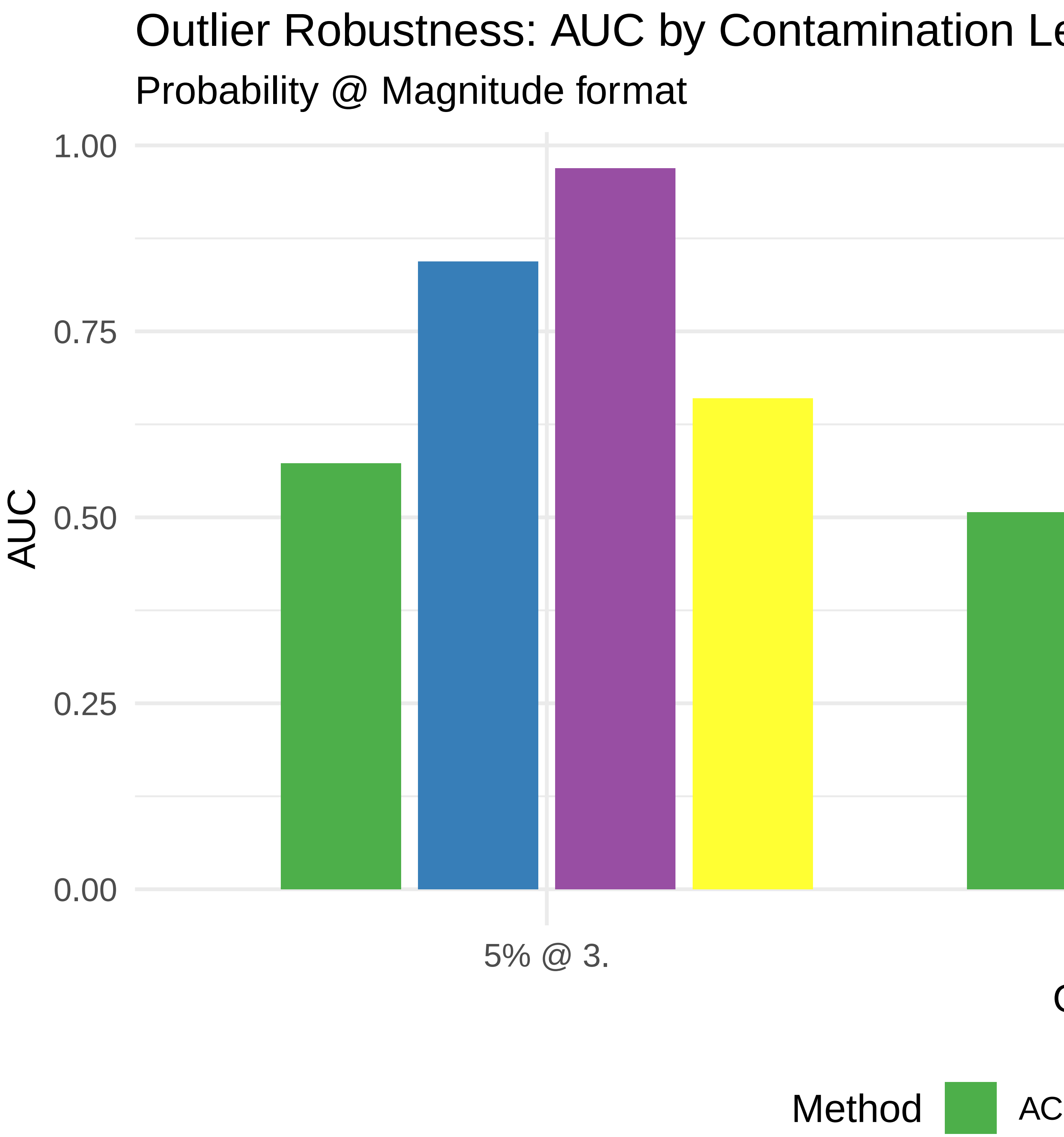

Figure 13: Method performance under different outlier configurations

```
Matrix products: default
BLAS:   /usr/lib/libblas.so.3.12.0
LAPACK: /usr/lib/liblapack.so.3.12.0  LAPACK version 3.12.0

locale:
 [1] LC_CTYPE=de_DE.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=de_DE.UTF-8        LC_COLLATE=de_DE.UTF-8
 [5] LC_MONETARY=de_DE.UTF-8    LC_MESSAGES=de_DE.UTF-8
 [7] LC_PAPER=de_DE.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=de_DE.UTF-8 LC_IDENTIFICATION=C

time zone: Europe/Berlin
tzcode source: system (glibc)

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] pROC_1.19.0.1 scales_1.4.0  knitr_1.51    purrr_1.2.0   tidyr_1.3.2
[6] dplyr_1.1.4   ggplot2_4.0.1 duckdb_1.4.3  DBI_1.2.3

loaded via a namespace (and not attached):
 [1] gtable_0.3.6       jsonlite_2.0.0     compiler_4.5.2     tidyselect_1.2.1
 [5] Rcpp_1.1.0         yaml_2.3.12        fastmap_1.2.0      R6_2.6.1
 [9] labeling_0.4.3     generics_0.1.4     tibble_3.3.0       pillar_1.11.1
[13] RColorBrewer_1.1-3 rlang_1.1.6        xfun_0.54          S7_0.2.0
[17] otel_0.2.0         cli_3.6.5          withr_3.0.2        magrittr_2.0.4
[21] digest_0.6.39      grid_4.5.2         lifecycle_1.0.4    vctrs_0.6.5
[25] evaluate_1.0.5     glue_1.8.0         farver_2.1.2       codetools_0.2-20
[29] rmarkdown_2.30     tools_4.5.2        pkgconfig_2.0.3    htmltools_0.5.9
```