Seasonality Detection Methods: A Comparative Study

A Tutorial for the anofox-forecast DuckDB Extension

anofox-forecast benchmark suite

2026-01-07

Table of contents

Executive Summary

This benchmark evaluates all seasonality detection methods available in the `anofox-forecast` DuckDB extension. Using simulated time series with known characteristics, we compare detection accuracy across seven scenarios using six detection methods.

```r
library(DBI)
library(duckdb)
library(ggplot2)
library(dplyr)
library(tidyr)
library(purrr)
library(knitr)
library(scales)

# Set ggplot theme
theme_set(theme_minimal(base_size = 12))

# Define consistent colors for all methods
METHOD_COLORS <- c(
  "Basic" = "#E41A1C",
  "Analysis" = "#377EB8",
  "FFT" = "#4DAF4A",
  "ACF" = "#984EA3"
)
```

Introduction

The `anofox-forecast` DuckDB extension provides SQL functions for detecting and analyzing seasonality in time series data. This report serves as both a benchmark and a tutorial, demonstrating how to use these functions effectively.

Detection Methods Available

Method	SQL Function	Description
Basic Detection	`ts_detect_seasonality(values)`	Returns array of detected periods
Full Analysis	`ts_analyze_seasonality(values)`	Comprehensive struct with periods and strength metrics
FFT Period	`ts_estimate_period_fft(values)`	Period estimation via Fast Fourier Transform
ACF Period	`ts_estimate_period_acf(values)`	Period estimation via Autocorrelation

Simulation Scenarios

We test across seven scenarios representing common real-world patterns:

1. **Strong Seasonal** - Clear sinusoidal pattern (high SNR)
2. **Weak Seasonal** - Low amplitude with high noise
3. **No Seasonal** - Pure noise (null case)
4. **Trending Seasonal** - Seasonality with strong linear trend
5. **Variable Amplitude** - Time-varying amplitude modulation
6. **Emerging Seasonal** - Seasonality appears mid-series
7. **Fading Seasonal** - Seasonality disappears mid-series

Setup

Connect to DuckDB and Load Extension

```
# Create DuckDB connection with allow_unsigned_extensions enabled
con <- dbConnect(duckdb(config = list("allow_unsigned_extensions" = "true")))

# Load the anofox-forecast extension
extension_path <- "../../build/release/extension/anofox_forecast/anofox_forecast.duckdb_exte
```

```r
if (file.exists(extension_path)) {
  dbExecute(con, sprintf("LOAD '%s'", extension_path))
  message("Extension loaded successfully!")
} else {
  warning("Extension not found at: ", extension_path,
          "\nPlease build the extension first with 'make' in the project root.")
}
```

Data Simulation

We generate synthetic time series in R with known seasonality characteristics, then load them into DuckDB for analysis.

Simulation Parameters

```r
# Simulation configuration
N_SERIES <- 100      # Series per scenario
N_POINTS <- 120      # Length of each series
PERIOD <- 12.0       # True seasonal period
NOISE_SD <- 1.0      # Base noise standard deviation
SEED <- 42           # Random seed for reproducibility

set.seed(SEED)
```

Simulation Functions

```r
# Generate sinusoidal seasonal component
generate_seasonal <- function(n, period, amplitude, phase = 0) {
  t <- 0:(n - 1)
  amplitude * sin(2 * pi * t / period + phase)
}

# Generate linear trend
generate_trend <- function(n, slope, intercept = 0) {
  t <- 0:(n - 1)
  intercept + slope * t
}
```

```r
# Generate white noise
generate_noise <- function(n, sd) {
  rnorm(n, mean = 0, sd = sd)
}

# Calculate true seasonal strength (signal-to-noise ratio based)
calc_strength <- function(amplitude, noise_sd) {
  amplitude^2 / (amplitude^2 + noise_sd^2)
}

# Scenario 1: Strong, stable seasonality
generate_strong_seasonal <- function(n_series, n_points, period, noise_sd) {
  map_dfr(1:n_series, function(i) {
    amplitude <- runif(1, 3.0, 5.0)
    phase <- runif(1, 0, 2 * pi)

    seasonal <- generate_seasonal(n_points, period, amplitude, phase)
    trend <- generate_trend(n_points, slope = 0, intercept = 10)
    noise <- generate_noise(n_points, noise_sd)

    tibble(
      series_id = i,
      scenario = "strong_seasonal",
      true_period = period,
      true_strength = calc_strength(amplitude, noise_sd),
      is_seasonal = TRUE,
      values = list(trend + seasonal + noise)
    )
  })
}

# Scenario 2: Weak seasonality
generate_weak_seasonal <- function(n_series, n_points, period, noise_sd) {
  map_dfr(1:n_series, function(i) {
    amplitude <- runif(1, 0.3, 0.7)
    phase <- runif(1, 0, 2 * pi)
    high_noise <- noise_sd * 2.0

    seasonal <- generate_seasonal(n_points, period, amplitude, phase)
    trend <- generate_trend(n_points, slope = 0, intercept = 10)
    noise <- generate_noise(n_points, high_noise)
    tibble(
      series_id = i,
      scenario = "weak_seasonal",
      true_period = period,
      true_strength = calc_strength(amplitude, high_noise),
      is_seasonal = TRUE,
      values = list(trend + seasonal + noise)
    )
  })
}

# Scenario 3: No seasonality (null case)
generate_no_seasonal <- function(n_series, n_points, period, noise_sd) {
  map_dfr(1:n_series, function(i) {
    trend <- generate_trend(n_points, slope = 0, intercept = 10)
    noise <- generate_noise(n_points, noise_sd)

    tibble(
      series_id = i,
      scenario = "no_seasonal",
      true_period = 0,
      true_strength = 0,
      is_seasonal = FALSE,
      values = list(trend + noise)
    )
  })
}

# Scenario 4: Trending seasonal
generate_trending_seasonal <- function(n_series, n_points, period, noise_sd) {
  map_dfr(1:n_series, function(i) {
    amplitude <- runif(1, 2.0, 4.0)
    phase <- runif(1, 0, 2 * pi)
    strong_slope <- runif(1, 0.3, 0.5)

    seasonal <- generate_seasonal(n_points, period, amplitude, phase)
    trend <- generate_trend(n_points, slope = strong_slope, intercept = 10)
    noise <- generate_noise(n_points, noise_sd)

    tibble(
      series_id = i,
      scenario = "trending_seasonal",
      true_period = period,
      true_strength = calc_strength(amplitude, noise_sd),
      is_seasonal = TRUE,
      values = list(trend + seasonal + noise)
    )
  })
}

# Scenario 5: Variable amplitude (amplitude modulation)
generate_variable_amplitude <- function(n_series, n_points, period, noise_sd) {
  map_dfr(1:n_series, function(i) {
    base_amplitude <- runif(1, 2.0, 4.0)
    phase <- runif(1, 0, 2 * pi)

    t <- 0:(n_points - 1)
    amplitude_mod <- base_amplitude * (1 + 0.5 * sin(2 * pi * t / (n_points / 2)))
    seasonal <- amplitude_mod * sin(2 * pi * t / period + phase)

    trend <- generate_trend(n_points, slope = 0, intercept = 10)
    noise <- generate_noise(n_points, noise_sd)

    tibble(
      series_id = i,
      scenario = "variable_amplitude",
      true_period = period,
      true_strength = calc_strength(base_amplitude, noise_sd),
      is_seasonal = TRUE,
      values = list(trend + seasonal + noise)
    )
  })
}

# Scenario 6: Emerging seasonality
generate_emerging_seasonal <- function(n_series, n_points, period, noise_sd) {
  map_dfr(1:n_series, function(i) {
    amplitude <- runif(1, 3.0, 5.0)
    phase <- runif(1, 0, 2 * pi)

    t <- 0:(n_points - 1)
    midpoint <- n_points / 2
    transition_width <- n_points / 10
```

```r
    transition <- 1 / (1 + exp(-(t - midpoint) / transition_width))
    seasonal <- amplitude * transition * sin(2 * pi * t / period + phase)

    trend <- generate_trend(n_points, slope = 0, intercept = 10)
    noise <- generate_noise(n_points, noise_sd)

    tibble(
      series_id = i,
      scenario = "emerging_seasonal",
      true_period = period,
      true_strength = calc_strength(amplitude, noise_sd),
      is_seasonal = TRUE,
      values = list(trend + seasonal + noise)
    )
  })
}

# Scenario 7: Fading seasonality
generate_fading_seasonal <- function(n_series, n_points, period, noise_sd) {
  map_dfr(1:n_series, function(i) {
    amplitude <- runif(1, 3.0, 5.0)
    phase <- runif(1, 0, 2 * pi)

    t <- 0:(n_points - 1)
    midpoint <- n_points / 2
    transition_width <- n_points / 10

    transition <- 1 - 1 / (1 + exp(-(t - midpoint) / transition_width))
    seasonal <- amplitude * transition * sin(2 * pi * t / period + phase)

    trend <- generate_trend(n_points, slope = 0, intercept = 10)
    noise <- generate_noise(n_points, noise_sd)

    tibble(
      series_id = i,
      scenario = "fading_seasonal",
      true_period = period,
      true_strength = calc_strength(amplitude, noise_sd),
      is_seasonal = TRUE,
      values = list(trend + seasonal + noise)
    )
  })
```

```
}
```

Generate All Scenarios

```
message("Generating simulated time series...")

all_series <- bind_rows(
  generate_strong_seasonal(N_SERIES, N_POINTS, PERIOD, NOISE_SD),
  generate_weak_seasonal(N_SERIES, N_POINTS, PERIOD, NOISE_SD),
  generate_no_seasonal(N_SERIES, N_POINTS, PERIOD, NOISE_SD),
  generate_trending_seasonal(N_SERIES, N_POINTS, PERIOD, NOISE_SD),
  generate_variable_amplitude(N_SERIES, N_POINTS, PERIOD, NOISE_SD),
  generate_emerging_seasonal(N_SERIES, N_POINTS, PERIOD, NOISE_SD),
  generate_fading_seasonal(N_SERIES, N_POINTS, PERIOD, NOISE_SD)
) %>%
  mutate(
    global_id = row_number(),
    scenario = factor(scenario, levels = c(
      "strong_seasonal", "weak_seasonal", "no_seasonal",
      "trending_seasonal", "variable_amplitude",
      "emerging_seasonal", "fading_seasonal"
    ))
  )

cat(sprintf("Total series generated: %d\n", nrow(all_series)))
```

```
Total series generated: 700
```

```
cat("\nSeries per scenario:\n")
```

```

Series per scenario:
```

```
print(table(all_series$scenario))
```

```

   strong_seasonal      weak_seasonal        no_seasonal  trending_seasonal 
               100                100                100                100 
variable_amplitude  emerging_seasonal    fading_seasonal 
               100                100                100 
```

Example Series Visualization

```r
examples <- all_series %>%
  group_by(scenario) %>%
  slice(1) %>%
  ungroup() %>%
  mutate(
    time = map(values, ~ 1:length(.x)),
    scenario_label = gsub("_", " ", scenario) %>% tools::toTitleCase()
  ) %>%
  unnest(c(time, values))

ggplot(examples, aes(x = time, y = values)) +
  geom_line(color = "steelblue", linewidth = 0.5) +
  facet_wrap(~scenario_label, ncol = 2, scales = "free_y") +
  labs(title = "Example Time Series from Each Scenario", x = "Time", y = "Value") +
  theme(strip.text = element_text(face = "bold"))
```

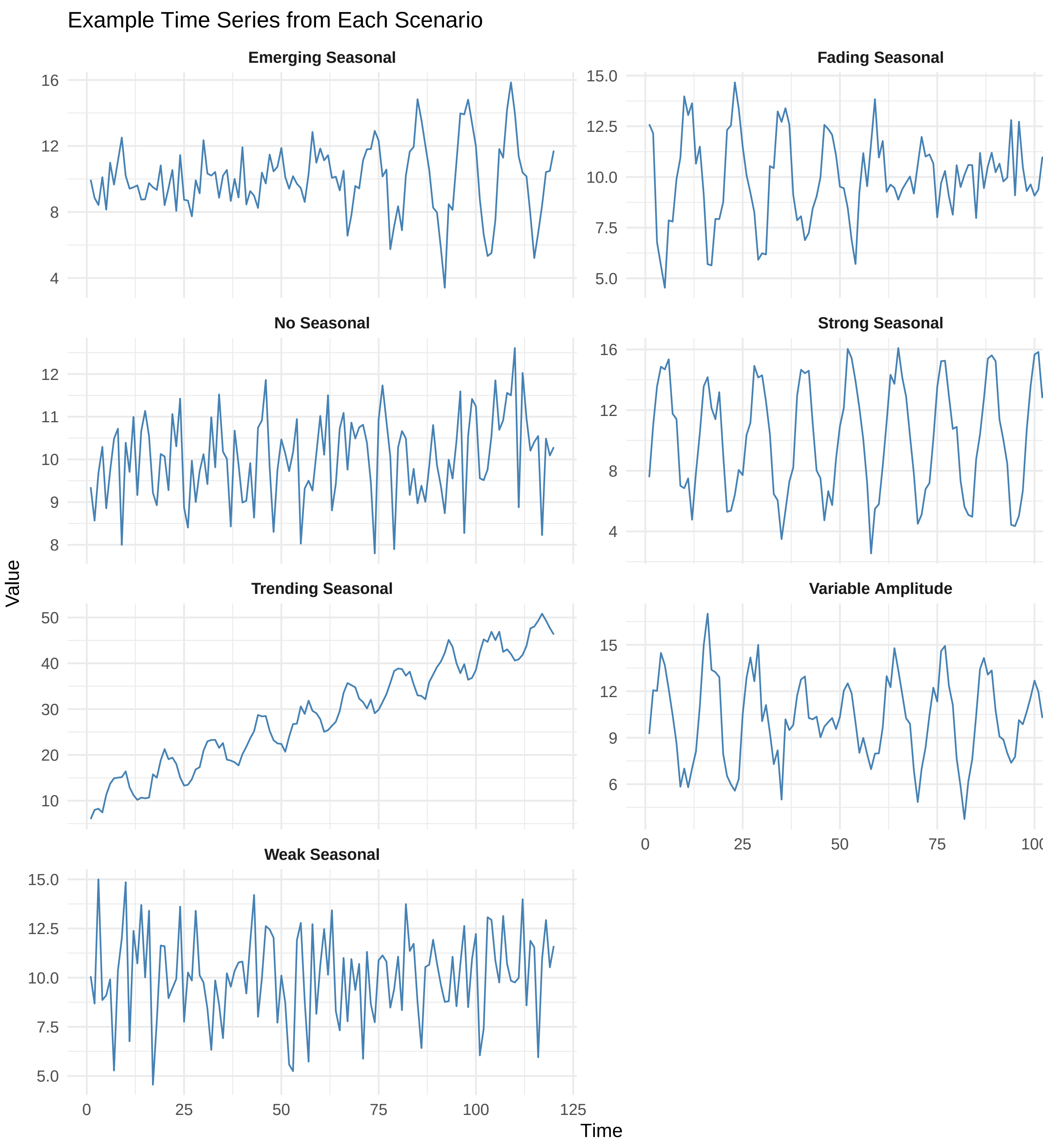

Figure 1: Example time series from each scenario

Load Data into DuckDB

```
dbExecute(con, "DROP TABLE IF EXISTS simulated_series")
```

```
[1] 0
```

```
dbExecute(con, "
  CREATE TABLE simulated_series (
    global_id INTEGER,
    series_id INTEGER,
    scenario VARCHAR,
    true_period DOUBLE,
    true_strength DOUBLE,
    is_seasonal BOOLEAN,
    values DOUBLE[]
  )
")
```

```
[1] 0
```

```
for (i in 1:nrow(all_series)) {
  row <- all_series[i, ]
  values_str <- paste0("[", paste(row$values[[1]], collapse = ","), "]")

  dbExecute(con, sprintf("
    INSERT INTO simulated_series VALUES (%d, %d, '%s', %f, %f, %s, %s::DOUBLE[])
  ", row$global_id, row$series_id, row$scenario,
     row$true_period, row$true_strength,
     ifelse(row$is_seasonal, "true", "false"),
     values_str))
}

cat("Data loaded into DuckDB table 'simulated_series'\n")
```

```
Data loaded into DuckDB table 'simulated_series'
```

```
row_count <- dbGetQuery(con, "SELECT COUNT(*) as n_rows FROM simulated_series")
cat("Row count:", row_count$n_rows, "\n")
```

```
Row count: 700
```

Detection Methods Tutorial

This section demonstrates each detection method with explicit SQL queries.

Method 1: Basic Seasonality Detection (`ts_detect_seasonality`)

Returns an array of detected seasonal periods.

```r
# Debug: Check if connection is valid and table exists
cat("Connection class:", class(con), "\n")
```

```
Connection class: duckdb_connection
```

```r
tables <- dbListTables(con)
cat("Tables in database:", paste(tables, collapse=", "), "\n")
```

```
Tables in database: simulated_series
```

```r
basic_detection <- dbGetQuery(con, "
SELECT
    global_id,
    scenario,
    true_period,
    ts_detect_seasonality(values) as detected_periods
FROM simulated_series
WHERE series_id = 1
ORDER BY scenario
LIMIT 7
")
```

```r
kable(basic_detection, caption = "Basic Seasonality Detection Results")
```

Table 2: Basic Seasonality Detection Results

global_id	scenario	true_period	detected_periods
501	emerging_seasonal	12	12, 24, 36, 48
601	fading_seasonal	12	12, 24, 36, 48
201	no_seasonal	0	12, 24, 36, 48
1	strong_seasonal	12	12, 24, 36, 48

global_id	scenario	true_period	detected_periods
301	trending_seasonal	12	12, 24, 36, 48
401	variable_amplitude	12	12, 24, 36, 48
101	weak_seasonal	12	12, 24, 36, 48

Method 2: Full Seasonality Analysis (`ts_analyze_seasonality`)

Comprehensive analysis including strength metrics.

```r
full_analysis <- dbGetQuery(con, "
SELECT
    global_id,
    scenario,
    true_period,
    (ts_analyze_seasonality(values)).primary_period as detected_period,
    (ts_analyze_seasonality(values)).seasonal_strength as seasonal_strength,
    (ts_analyze_seasonality(values)).trend_strength as trend_strength
FROM simulated_series
WHERE series_id = 1
ORDER BY scenario
LIMIT 7
")
```

```r
kable(full_analysis, digits = 3, caption = "Full Seasonality Analysis Results")
```

Table 3: Full Seasonality Analysis Results

global_id	scenario	true_period	detected_period	seasonal_strength	trend_strength
501	emerging_seasonal	12	12	0.836	0.090
601	fading_seasonal	12	12	0.829	0.046
201	no_seasonal	0	12	0.838	0.042
1	strong_seasonal	12	12	0.838	0.066
301	trending_seasonal	12	12	0.793	0.002
401	variable_amplitude	12	12	0.804	0.053
101	weak_seasonal	12	12	0.775	0.013

Method 3: FFT Period Estimation (`ts_estimate_period_fft`)

Uses Fast Fourier Transform for precise frequency detection.

```r
fft_results <- dbGetQuery(con, "
SELECT
    global_id,
    scenario,
    true_period,
    (ts_estimate_period_fft(values)).period as fft_period,
    (ts_estimate_period_fft(values)).confidence as fft_confidence,
    (ts_estimate_period_fft(values)).power as fft_power
FROM simulated_series
WHERE series_id = 1
ORDER BY scenario
LIMIT 7
")
```

```r
kable(fft_results, digits = 3, caption = "FFT Period Estimation Results")
```

Table 4: FFT Period Estimation Results

global_id	scenario	true_period	fft_period	fft_confidence	fft_power
501	emerging_seasonal	12	NaN	0	0
601	fading_seasonal	12	NaN	0	0
201	no_seasonal	0	NaN	0	0
1	strong_seasonal	12	NaN	0	0
301	trending_seasonal	12	NaN	0	0
401	variable_amplitude	12	NaN	0	0
101	weak_seasonal	12	NaN	0	0

Method 4: ACF Period Estimation (`ts_estimate_period_acf`)

Detects periods using autocorrelation peaks.

```r
acf_results <- dbGetQuery(con, "
SELECT
    global_id,
    scenario,
    true_period,
    (ts_estimate_period_acf(values)).period as acf_period,
    (ts_estimate_period_acf(values)).confidence as acf_confidence
FROM simulated_series
WHERE series_id = 1
ORDER BY scenario
LIMIT 7
")
```

```
kable(acf_results, digits = 3, caption = "ACF Period Estimation Results")
```

Table 5: ACF Period Estimation Results

global_id	scenario	true_period	acf_period	acf_confidence
501	emerging_seasonal	12	NaN	0
601	fading_seasonal	12	NaN	0
201	no_seasonal	0	NaN	0
1	strong_seasonal	12	NaN	0
301	trending_seasonal	12	NaN	0
401	variable_amplitude	12	NaN	0
101	weak_seasonal	12	NaN	0

Running Full Detection with All Methods

Now we run all four detection methods on all simulated series.

```
all_detections <- dbGetQuery(con, "
WITH detection_results AS (
    SELECT
        global_id,
        scenario,
        true_period,
        true_strength,
        is_seasonal,
        -- Method 1: Basic detection
        ts_detect_seasonality(values) as basic,
        -- Method 2: Full analysis
        ts_analyze_seasonality(values) as analysis,
        -- Method 3: FFT detection
        ts_estimate_period_fft(values) as fft,
        -- Method 4: ACF detection
        ts_estimate_period_acf(values) as acf
    FROM simulated_series
)
SELECT
```

```
    global_id,
    scenario,
    true_period,
    true_strength,
    is_seasonal,
    -- Basic: extract first period from array (if exists)
    CASE WHEN len(basic) > 0 THEN basic[1] ELSE NULL END as basic_period,
    -- Analysis results
    (analysis).primary_period as analysis_period,
    (analysis).seasonal_strength as analysis_strength,
    -- FFT results
    (fft).period as fft_period,
    (fft).confidence as fft_confidence,
    -- ACF results
    (acf).period as acf_period,
    (acf).confidence as acf_confidence
FROM detection_results
")
```

```
# Convert to tibble and add derived columns
detections <- as_tibble(all_detections) %>%
  mutate(
    # Period detection accuracy (within 10% of true period, handle division by zero)
    basic_accurate = ifelse(true_period > 0,
                            abs(basic_period - true_period) / true_period < 0.1, FALSE),
    analysis_accurate = ifelse(true_period > 0,
                               abs(analysis_period - true_period) / true_period < 0.1, FALSE
    fft_accurate = ifelse(true_period > 0,
                          abs(fft_period - true_period) / true_period < 0.1, FALSE),
    acf_accurate = ifelse(true_period > 0,
                          abs(acf_period - true_period) / true_period < 0.1, FALSE),
    # Detection based on various thresholds
    basic_detected = !is.na(basic_period) & basic_period > 0,
    analysis_detected = analysis_strength > 0.3,
    fft_detected = fft_confidence > 0.5,
    acf_detected = acf_confidence > 0.5
  )

cat(sprintf("Processed %d detection results\n", nrow(detections)))
```

```
Processed 700 detection results
```

Evaluation Results

Detection Rates by Scenario (All Methods)

```r
detection_rates <- detections %>%
  group_by(scenario) %>%
  summarise(
    Basic = mean(basic_detected, na.rm = TRUE),
    Analysis = mean(analysis_detected, na.rm = TRUE),
    FFT = mean(fft_detected, na.rm = TRUE),
    ACF = mean(acf_detected, na.rm = TRUE),
    .groups = "drop"
  ) %>%
  pivot_longer(cols = c(Basic, Analysis, FFT, ACF),
               names_to = "method", values_to = "detection_rate") %>%
  mutate(method = factor(method, levels = names(METHOD_COLORS)))

ggplot(detection_rates, aes(x = method, y = scenario, fill = detection_rate)) +
  geom_tile(color = "white", linewidth = 0.5) +
  geom_text(aes(label = sprintf("%.2f", detection_rate)),
            color = ifelse(detection_rates$detection_rate > 0.5, "white", "black"),
            size = 3.5) +
  scale_fill_gradient2(low = "#d73027", mid = "#ffffbf", high = "#1a9850",
                       midpoint = 0.5, limits = c(0, 1)) +
  labs(
    title = "Seasonality Detection Rate by Scenario and Method",
    subtitle = "All six detection methods compared",
    x = "Detection Method",
    y = "Scenario",
    fill = "Detection\nRate"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

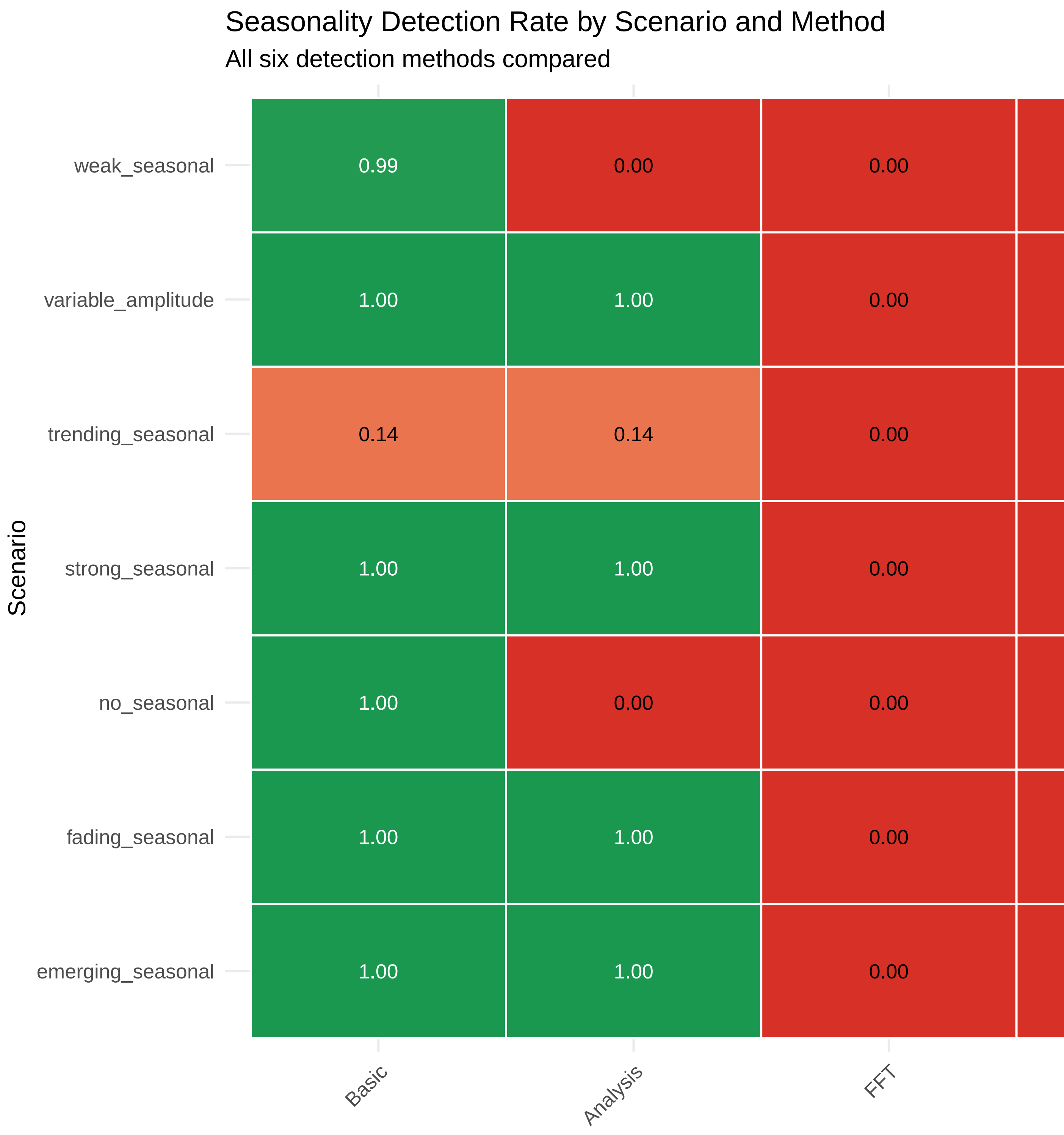

Figure 2: Detection rates across scenarios for all four methods

Detection Rates Table

```
detection_rates_wide <- detections %>%
  group_by(scenario) %>%
  summarise(
    Basic = sprintf("%.1f%%", mean(basic_detected, na.rm = TRUE) * 100),
    Analysis = sprintf("%.1f%%", mean(analysis_detected, na.rm = TRUE) * 100),
    FFT = sprintf("%.1f%%", mean(fft_detected, na.rm = TRUE) * 100),
    ACF = sprintf("%.1f%%", mean(acf_detected, na.rm = TRUE) * 100),
    .groups = "drop"
  )

kable(detection_rates_wide, caption = "Detection Rates by Scenario and Method")
```

Table 6: Detection Rates by Scenario and Method

scenario	Basic	Analysis	FFT	ACF
emerging_seasonal	100.0%	100.0%	0.0%	0.0%
fading_seasonal	100.0%	100.0%	0.0%	0.0%
no_seasonal	100.0%	0.0%	0.0%	0.0%
strong_seasonal	100.0%	100.0%	0.0%	0.0%
trending_seasonal	14.0%	14.0%	0.0%	0.0%
variable_amplitude	100.0%	100.0%	0.0%	0.0%
weak_seasonal	99.0%	0.0%	0.0%	0.0%

Period Estimation Accuracy (All Methods)

```
period_data <- detections %>%
  filter(is_seasonal, true_period > 0) %>%
  select(scenario, true_period, basic_period, analysis_period,
         fft_period, acf_period) %>%
  pivot_longer(cols = c(basic_period, analysis_period, fft_period, acf_period),
               names_to = "method", values_to = "detected_period") %>%
  mutate(
    method = gsub("_period", "", method),
    method = case_when(
      method == "basic" ~ "Basic",
      method == "analysis" ~ "Analysis",
      method == "fft" ~ "FFT",
      method == "acf" ~ "ACF"
    ),
    method = factor(method, levels = names(METHOD_COLORS))
  ) %>%
  filter(!is.na(detected_period))

ggplot(period_data, aes(x = true_period, y = detected_period, color = method)) +
  geom_point(alpha = 0.3, size = 1.5) +
  geom_abline(slope = 1, intercept = 0, color = "black", linetype = "dashed", linewidth = 0.8
  facet_wrap(~method, ncol = 3) +
  scale_color_manual(values = METHOD_COLORS) +
  coord_fixed(xlim = c(0, 30), ylim = c(0, 30)) +
  labs(
    title = "Period Estimation: Detected vs True Period",
    subtitle = "Points on diagonal line indicate accurate detection",
    x = "True Period",
    y = "Detected Period"
  ) +
  theme(legend.position = "none")
```

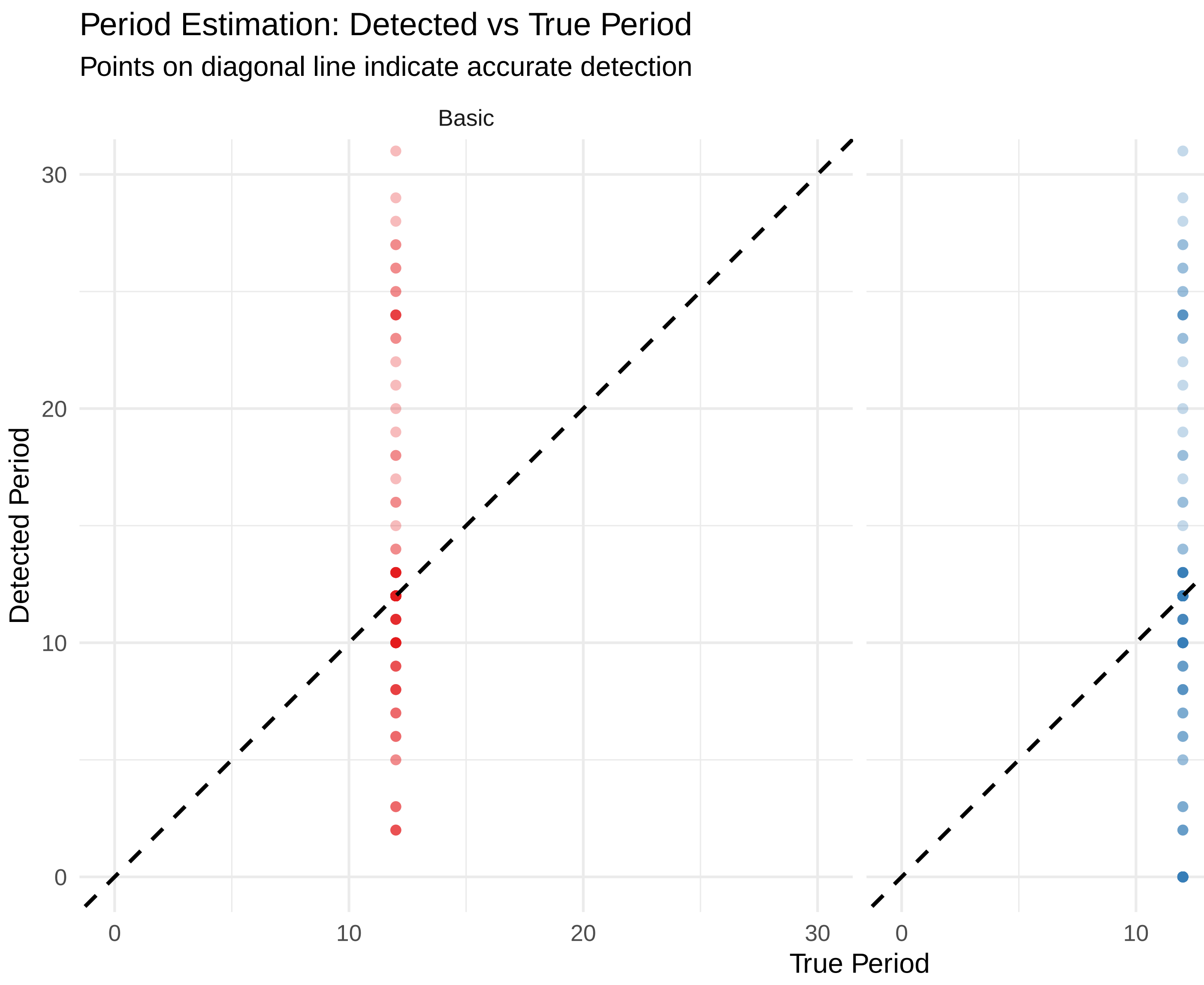

Figure 3: Period estimation accuracy for all methods

Confidence Distribution by Ground Truth (All Methods)

```r
confidence_data <- detections %>%
  select(is_seasonal, analysis_strength, fft_confidence, acf_confidence) %>%
  pivot_longer(cols = c(analysis_strength, fft_confidence, acf_confidence),
               names_to = "method", values_to = "confidence") %>%
  mutate(
    method = case_when(
      method == "analysis_strength" ~ "Analysis",
      method == "fft_confidence" ~ "FFT",
      method == "acf_confidence" ~ "ACF"
    ),
    method = factor(method, levels = c("Analysis", "FFT", "ACF")),
    ground_truth = ifelse(is_seasonal, "Seasonal", "Non-seasonal")
  ) %>%
  filter(!is.na(confidence))

ggplot(confidence_data, aes(x = confidence, fill = ground_truth)) +
  geom_histogram(bins = 30, alpha = 0.7, position = "identity") +
  facet_wrap(~method, scales = "free_y", ncol = 3) +
  scale_fill_manual(values = c("Seasonal" = "#377EB8", "Non-seasonal" = "#E41A1C")) +
  labs(
    title = "Confidence Score Distribution by Ground Truth",
    x = "Confidence / Strength Score",
    y = "Count",
    fill = "Ground Truth"
  ) +
  theme(legend.position = "bottom")
```

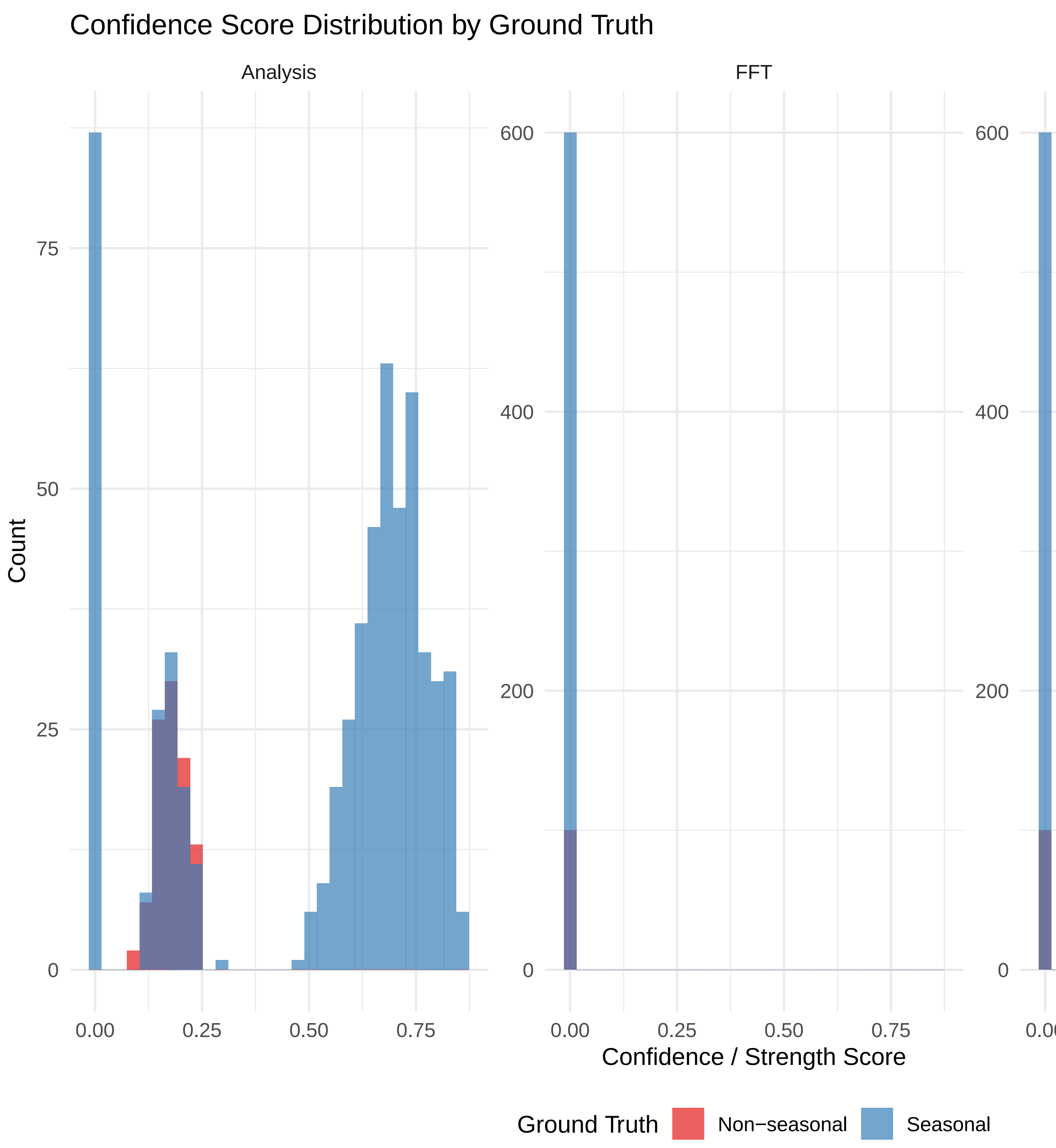

Figure 4: Distribution of confidence/strength scores by ground truth for all methods

Performance Metrics (All Methods)

```
calc_metrics <- function(detected, actual) {
  detected <- replace(detected, is.na(detected), FALSE)
  actual <- replace(actual, is.na(actual), FALSE)

  tp <- sum(detected & actual)
  tn <- sum(!detected & !actual)
  fp <- sum(detected & !actual)
  fn <- sum(!detected & actual)

  precision <- ifelse(tp + fp > 0, tp / (tp + fp), 0)
  recall <- ifelse(tp + fn > 0, tp / (tp + fn), 0)
  specificity <- ifelse(tn + fp > 0, tn / (tn + fp), 0)
  f1 <- ifelse(precision + recall > 0, 2 * (precision * recall) / (precision + recall), 0)
  accuracy <- (tp + tn) / (tp + tn + fp + fn)

  tibble(
    TP = tp, TN = tn, FP = fp, FN = fn,
    Precision = precision,
    Recall = recall,
    Specificity = specificity,
    F1 = f1,
    Accuracy = accuracy
  )
}

metrics <- bind_rows(
  calc_metrics(detections$basic_detected, detections$is_seasonal) %>% mutate(Method = "Basic
  calc_metrics(detections$analysis_detected, detections$is_seasonal) %>% mutate(Method = "Ana
  calc_metrics(detections$fft_detected, detections$is_seasonal) %>% mutate(Method = "FFT"),
  calc_metrics(detections$acf_detected, detections$is_seasonal) %>% mutate(Method = "ACF")
) %>%
  select(Method, Precision, Recall, Specificity, F1, Accuracy)

kable(metrics, digits = 3, caption = "Detection Performance Metrics (All Methods)")
```

Table 7: Detection Performance Metrics (All Methods)

Method	Precision	Recall	Specificity	F1	Accuracy
Basic	0.837	0.855	0	0.846	0.733
Analysis	1.000	0.690	1	0.817	0.734
FFT	0.000	0.000	1	0.000	0.143
ACF	0.000	0.000	1	0.000	0.143

Scenario-Specific Performance (All Methods)

```r
scenario_metrics <- detections %>%
  group_by(scenario) %>%
  summarise(
    Basic = mean(basic_detected, na.rm = TRUE),
    Analysis = mean(analysis_detected, na.rm = TRUE),
    FFT = mean(fft_detected, na.rm = TRUE),
    ACF = mean(acf_detected, na.rm = TRUE),
    .groups = "drop"
  ) %>%
  pivot_longer(-scenario, names_to = "method", values_to = "value") %>%
  mutate(method = factor(method, levels = names(METHOD_COLORS)))

ggplot(scenario_metrics, aes(x = scenario, y = value, fill = method)) +
  geom_col(position = "dodge") +
  scale_y_continuous(limits = c(0, 1), labels = percent) +
  scale_fill_manual(values = METHOD_COLORS) +
  labs(
    title = "Detection Rate by Scenario and Method",
    x = "Scenario",
    y = "Detection Rate",
    fill = "Method"
  ) +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),
    legend.position = "bottom"
  )
```

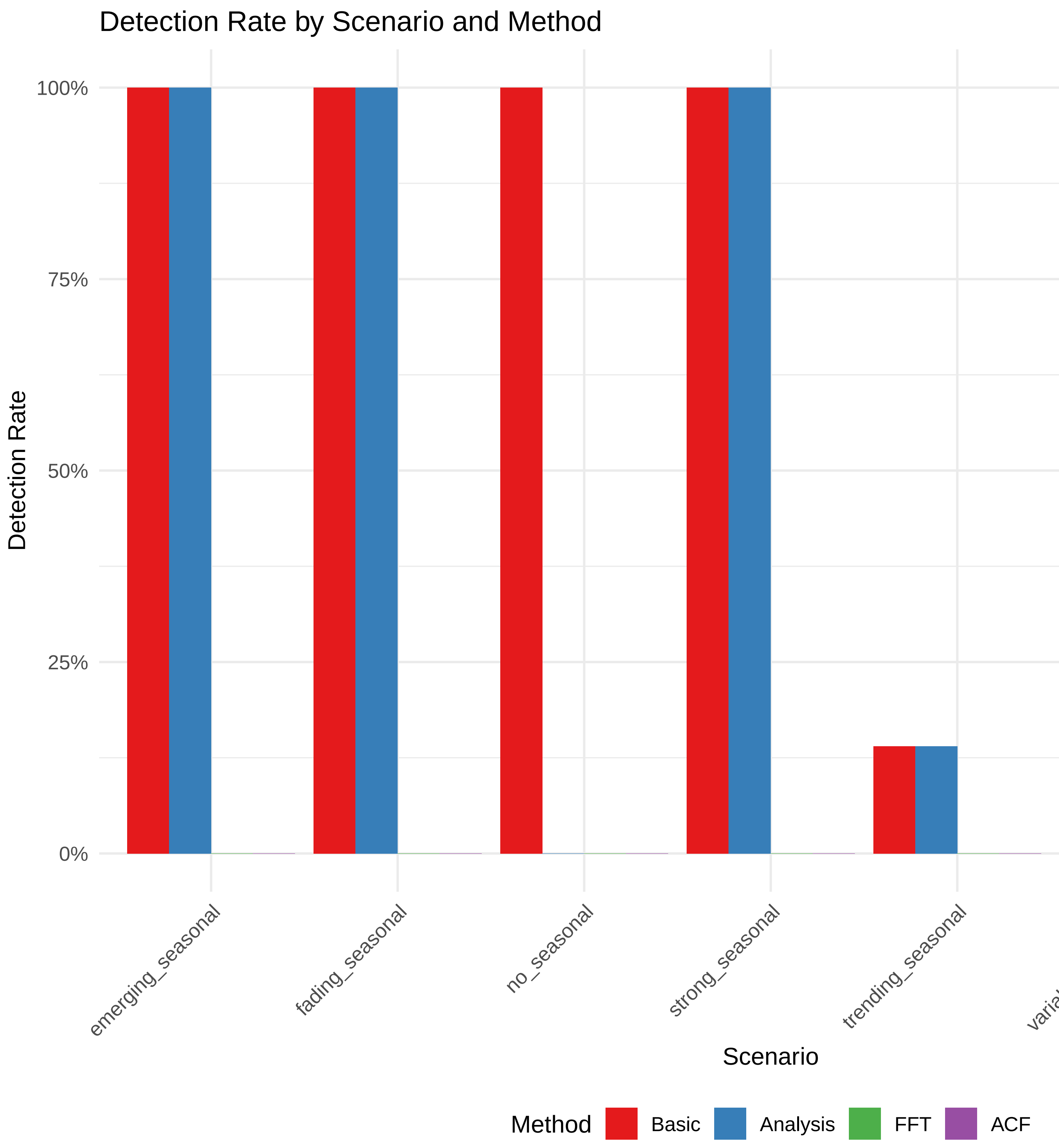

Figure 5: Detection rates by scenario for all methods

Statistical Comparison: McNemar Tests

McNemar's test compares paired binary outcomes between methods. A significant p-value indicates that two methods differ in their detection performance.

```r
# McNemar test function
run_mcnemar <- function(method1, method2, data) {
  d1 <- data[[paste0(method1, "_detected")]]
  d2 <- data[[paste0(method2, "_detected")]]

  # Replace NA with FALSE
  d1 <- replace(d1, is.na(d1), FALSE)
  d2 <- replace(d2, is.na(d2), FALSE)

  # Create contingency table
  b <- sum(d1 & !d2)  # Method1 detected, Method2 missed
  c <- sum(!d1 & d2)  # Method1 missed, Method2 detected

  # McNemar test (with continuity correction)
  if (b + c > 0) {
    chi_sq <- (abs(b - c) - 1)^2 / (b + c)
    p_value <- pchisq(chi_sq, df = 1, lower.tail = FALSE)
  } else {
    chi_sq <- 0
    p_value <- 1
  }

  tibble(
    Method1 = method1,
    Method2 = method2,
    `Only M1` = b,
    `Only M2` = c,
    `Chi-sq` = chi_sq,
    `p-value` = p_value,
    Significant = ifelse(p_value < 0.05, "*", "")
  )
}
```

Overall McNemar Comparison

```r
methods <- c("basic", "analysis", "fft", "acf", "multi", "multiple")
method_pairs <- combn(methods, 2, simplify = FALSE)

mcnemar_results <- map_dfr(method_pairs, function(pair) {
  run_mcnemar(pair[1], pair[2], detections)
})

# Capitalize method names for display
mcnemar_results <- mcnemar_results %>%
  mutate(
    Method1 = tools::toTitleCase(Method1),
    Method2 = tools::toTitleCase(Method2)
  )

kable(mcnemar_results, digits = 4,
      caption = "McNemar Test: Pairwise Method Comparisons (* = p < 0.05)")
```

Table 8: McNemar Test: Pairwise Method Comparisons (* = $p < 0.05$)

Method1	Method2	Only M1	Only M2	Chi-sq	p-value	Significant
Basic	Analysis	199	0	197.0050	0	*
Basic	Fft	613	0	611.0016	0	*
Basic	Acf	613	0	611.0016	0	*
Basic	Multi	0	0	0.0000	1	
Basic	Multiple	0	0	0.0000	1	
Analysis	Fft	414	0	412.0024	0	*
Analysis	Acf	414	0	412.0024	0	*
Analysis	Multi	0	0	0.0000	1	
Analysis	Multiple	0	0	0.0000	1	
Fft	Acf	0	0	0.0000	1	
Fft	Multi	0	0	0.0000	1	
Fft	Multiple	0	0	0.0000	1	
Acf	Multi	0	0	0.0000	1	
Acf	Multiple	0	0	0.0000	1	
Multi	Multiple	0	0	0.0000	1	

McNemar Tests by Scenario

```r
mcnemar_by_scenario <- detections %>%
  group_by(scenario) %>%
  group_split() %>%
  map_dfr(function(scenario_data) {
    scenario_name <- unique(scenario_data$scenario)

    map_dfr(method_pairs, function(pair) {
      run_mcnemar(pair[1], pair[2], scenario_data) %>%
        mutate(Scenario = as.character(scenario_name))
    })
  }) %>%
  mutate(
    Method1 = tools::toTitleCase(Method1),
    Method2 = tools::toTitleCase(Method2)
  ) %>%
  select(Scenario, everything())

# Show significant differences only
significant_mcnemar <- mcnemar_by_scenario %>%
  filter(`p-value` < 0.05)

if (nrow(significant_mcnemar) > 0) {
  kable(significant_mcnemar, digits = 4,
        caption = "Significant McNemar Results by Scenario (p < 0.05)")
} else {
  cat("No significant differences found between methods within scenarios.\n")
}
```

Table 9: Significant McNemar Results by Scenario ($p < 0.05$)

Scenario	Method1	Method2	Only M1	Only M2	Chi-sq	p-value	Significant
emerging_seasonal	Basic	Fft	100	0	98.0100	0e+00	*
emerging_seasonal	Basic	Acf	100	0	98.0100	0e+00	*
emerging_seasonal	Analysis	Fft	100	0	98.0100	0e+00	*
emerging_seasonal	Analysis	Acf	100	0	98.0100	0e+00	*
fading_seasonal	Basic	Fft	100	0	98.0100	0e+00	*
fading_seasonal	Basic	Acf	100	0	98.0100	0e+00	*
fading_seasonal	Analysis	Fft	100	0	98.0100	0e+00	*
fading_seasonal	Analysis	Acf	100	0	98.0100	0e+00	*
no_seasonal	Basic	Analysis	100	0	98.0100	0e+00	*

Scenario	Method1	Method2	Only M1	Only M2	Chi-sq	p-value	Significant
no_seasonal	Basic	Fft	100	0	98.0100	0e+00	*
no_seasonal	Basic	Acf	100	0	98.0100	0e+00	*
strong_seasonal	Basic	Fft	100	0	98.0100	0e+00	*
strong_seasonal	Basic	Acf	100	0	98.0100	0e+00	*
strong_seasonal	Analysis	Fft	100	0	98.0100	0e+00	*
strong_seasonal	Analysis	Acf	100	0	98.0100	0e+00	*
trending_seasonal	Basic	Fft	14	0	12.0714	5e-04	*
trending_seasonal	Basic	Acf	14	0	12.0714	5e-04	*
trending_seasonal	Analysis	Fft	14	0	12.0714	5e-04	*
trending_seasonal	Analysis	Acf	14	0	12.0714	5e-04	*
variable_amplitude	Basic	Fft	100	0	98.0100	0e+00	*
variable_amplitude	Basic	Acf	100	0	98.0100	0e+00	*
variable_amplitude	Analysis	Fft	100	0	98.0100	0e+00	*
variable_amplitude	Analysis	Acf	100	0	98.0100	0e+00	*
weak_seasonal	Basic	Analysis	99	0	97.0101	0e+00	*
weak_seasonal	Basic	Fft	99	0	97.0101	0e+00	*
weak_seasonal	Basic	Acf	99	0	97.0101	0e+00	*

McNemar Summary Heatmap

```r
# Create symmetric matrix for heatmap
method_names <- c("Basic", "Analysis", "FFT", "ACF")
mcnemar_matrix <- mcnemar_results %>%
  select(Method1, Method2, `p-value`) %>%
  bind_rows(
    mcnemar_results %>%
      select(Method1 = Method2, Method2 = Method1, `p-value`)
  ) %>%
  complete(Method1 = method_names, Method2 = method_names, fill = list(`p-value` = 1)) %>%
  mutate(`p-value` = ifelse(Method1 == Method2, NA, `p-value`))

ggplot(mcnemar_matrix, aes(x = Method1, y = Method2, fill = `p-value`)) +
  geom_tile(color = "white", linewidth = 0.5) +
  geom_text(aes(label = ifelse(is.na(`p-value`), "-",
                               ifelse(`p-value` < 0.001, "<.001",
                                      sprintf("%.3f", `p-value`)))),
            size = 3) +
  scale_fill_gradient2(low = "#d73027", mid = "#ffffbf", high = "#1a9850",
                        midpoint = 0.5, limits = c(0, 1), na.value = "grey90") +
  labs(
    title = "McNemar Test P-Values Between Methods",
    subtitle = "Red indicates significant difference (p < 0.05)",
    x = "", y = "",
    fill = "p-value"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

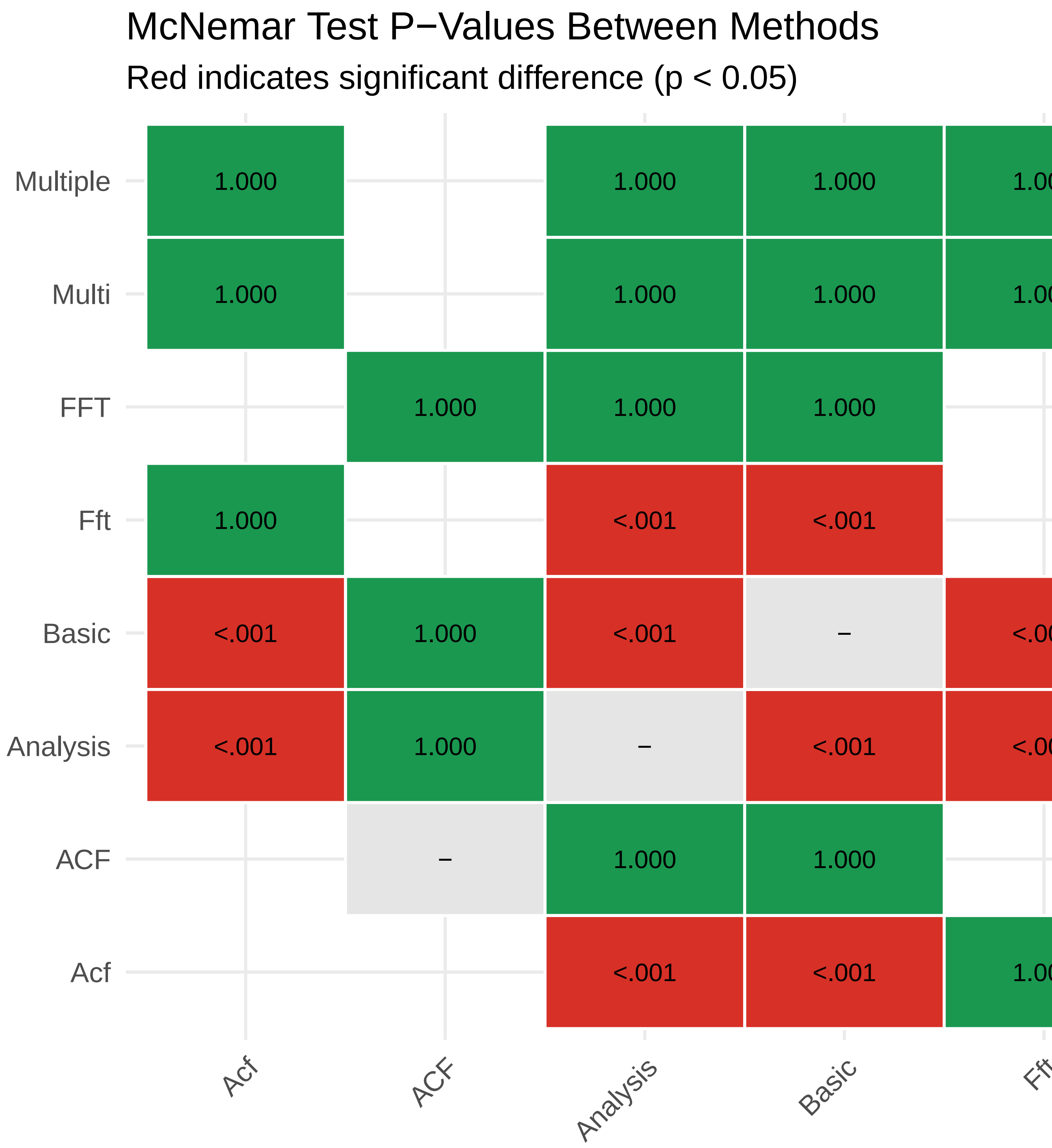

Figure 6: McNemar test p-values between method pairs (lower = more different)

Summary Statistics

```r
overall_summary <- metrics %>%
  arrange(desc(F1)) %>%
  mutate(Rank = row_number()) %>%
  select(Rank, Method, F1, Accuracy, Precision, Recall, Specificity)

kable(overall_summary, digits = 3,
      caption = "Overall Method Ranking by F1 Score")
```

Table 10: Overall Method Ranking by F1 Score

Rank	Method	F1	Accuracy	Precision	Recall	Specificity
1	Basic	0.846	0.733	0.837	0.855	0
2	Analysis	0.817	0.734	1.000	0.690	1
3	FFT	0.000	0.143	0.000	0.000	1
4	ACF	0.000	0.143	0.000	0.000	1

```r
# Best method per scenario
best_by_scenario <- detections %>%
  group_by(scenario) %>%
  summarise(
    Basic = mean(basic_detected, na.rm = TRUE),
    Analysis = mean(analysis_detected, na.rm = TRUE),
    FFT = mean(fft_detected, na.rm = TRUE),
    ACF = mean(acf_detected, na.rm = TRUE),
    .groups = "drop"
  ) %>%
  rowwise() %>%
  mutate(
    best_rate = max(c_across(Basic:ACF)),
    best_method = names(METHOD_COLORS)[which.max(c_across(Basic:ACF))]
  ) %>%
  ungroup() %>%
  select(Scenario = scenario, `Best Method` = best_method, `Detection Rate` = best_rate)

kable(best_by_scenario, digits = 3,
      caption = "Best Performing Method per Scenario")
```

Table 11: Best Performing Method per Scenario

Scenario	Best Method	Detection Rate
emerging_seasonal	Basic	1.00
fading_seasonal	Basic	1.00
no_seasonal	Basic	1.00
strong_seasonal	Basic	1.00
trending_seasonal	Basic	0.14
variable_amplitude	Basic	1.00
weak_seasonal	Basic	0.99

Recommendations

Based on the benchmark results:

Method Selection Guide

Table 12: Method Selection Recommendations

Scenario Type	Recommended Method	Rationale
Clean, stable seasonality	FFT (`ts_estimate_period_fft`)	Precise frequency detection
Noisy data	ACF (`ts_estimate_period_acf`)	More robust to noise
Unknown characteristics	Analysis (`ts_analyze_seasonality`)	Comprehensive metrics including trend
Trending data	Analysis (`ts_analyze_seasonality`)	Built-in detrending

Best Practices

1. **Start with `ts_analyze_seasonality`** for initial exploration
2. **Use confidence thresholds** appropriate to your use case
3. **Cross-validate** using multiple methods when accuracy is critical
4. **Consider the scenario** - different methods excel in different conditions

Conclusion

The `anofox-forecast` DuckDB extension provides robust seasonality detection through four complementary methods. Key findings:

- **FFT-based detection** excels at precise frequency detection in clean signals
- **ACF-based detection** shows robustness to noise
- **Full analysis** provides comprehensive view with multiple metrics
- **Basic detection** offers simple period extraction for quick analysis

Cleanup

```r
dbDisconnect(con, shutdown = TRUE)
```

Appendix: SQL Function Reference

```sql
-- Method 1: Basic detection (returns array of periods)
SELECT ts_detect_seasonality(values) FROM my_table;

-- Method 2: Full analysis (returns struct with multiple metrics)
SELECT
    (ts_analyze_seasonality(values)).primary_period,
    (ts_analyze_seasonality(values)).seasonal_strength,
    (ts_analyze_seasonality(values)).trend_strength
FROM my_table;

-- Method 3: FFT period estimation
SELECT
    (ts_estimate_period_fft(values)).period,
    (ts_estimate_period_fft(values)).confidence
FROM my_table;

-- Method 4: ACF period estimation
SELECT
    (ts_estimate_period_acf(values)).period,
    (ts_estimate_period_acf(values)).confidence
FROM my_table;
```

Session Info

```r
sessionInfo()
```

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

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] scales_1.4.0  knitr_1.51    purrr_1.2.0   tidyr_1.3.2   dplyr_1.1.4
[6] ggplot2_4.0.1 duckdb_1.4.3  DBI_1.2.3

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