

Package: seasight (via r-universe)

September 14, 2026

Type Package

Title Tools for Seasonal Adjustment Diagnostics and Reporting

Version 0.1.0

Depends R (>= 4.1.0)

Description Provides tools for automated diagnostics, model selection and reporting for seasonal adjustment based on X-13ARIMA-SEATS. Wraps the 'seasonal' package to run and compare alternative specifications, extract quality indicators and residual diagnostics, and generate reproducible R Markdown reports for official statistics and other production workflows.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE, roclets = c("` namespace", "` rd", "` srr::srr_stats_roclet"))

RoxygenNote 7.3.3

Imports seasonal, tsbox, tibble, dplyr, purrr, stringr, htmltools, glue

Suggests callr, testthat (>= 3.0.0), knitr, rmarkdown, units

VignetteBuilder knitr

Config/testthat/edition 3

URL <https://github.com/p-wegmueller/seasight>,
<https://p-wegmueller.github.io/seasight/>

BugReports <https://github.com/p-wegmueller/seasight/issues>

Config/pak/sysreqs libicu-dev

Repository <https://ropensci-staging.r-universe.dev>

Date/Publication 2026-09-14 19:48:24 UTC

RemoteUrl <https://github.com/p-wegmueller/seasight>

RemoteRef HEAD

RemoteSha a89947bd5fd27f589134cecf750a7c2ec1140b63

Contents

auto_seasonal_analysis	2
build_user_xreg	5
extract_outliers	6
plot.auto_seasonal_analysis	7
probe_case	8
sa_align_regressor	9
sa_batch_run	10
sa_compare	11
sa_copyable_call	12
sa_engine_choice_card	13
sa_existence_call	13
sa_existence_card	14
sa_is_do_not_adjust	15
sa_report_html	15
sa_should_switch	18
sa_tests_model	19
sa_top_candidates_table	20
Index	22

auto_seasonal_analysis

Automatic seasonal analysis (candidate grid + ranking)

Description

Runs a grid of X-13ARIMA-SEATS specifications, computes diagnostics (QS tests, residual checks, revision metrics, distance to a baseline model, etc.), and ranks candidates using a composite score. Returns the best model together with a diagnostic table and seasonality call. The function may fit several X-13 models; use `max_specs` for fast exploratory runs and larger grids for review runs.

Usage

```
auto_seasonal_analysis(
  y,
  specs = NULL,
  use_fivebest = TRUE,
  max_specs = NULL,
  seasonal_only = TRUE,
  auto_outliers = TRUE,
  transform_fun = c("auto", "log", "none"),
  td_candidates = NULL,
  td_usertype = "td",
  include_easter = c("auto", "always", "off"),
  easter_len = 15L,
  include_history_top_n = 10,
```

```

early_period_end = NULL,
current_model = NULL,
current_sa = NULL,
current_seasonal = NULL,
w_qs = 1,
w_stability = 1,
w_aicc = 1,
w_rev = 0.5,
w_dist_sa = 1,
w_dist_seas = 0.5,
w_engine = 1,
engine = c("seats", "x11", "auto"),
outlier_types = c("AO", "LS", "TC"),
outlier_method = "AddOne",
outlier_critical = 4,
outlier_alpha = NULL,
comparison_mode = c("full_search", "incumbent_fixed")
)

```

Arguments

<code>y</code>	A time series (ts) or an object that can be converted to ts via <code>tsbox::ts_ts()</code> .
<code>specs</code>	Optional character vector of ARIMA specifications in the form "(p d q)(P D Q)". If NULL, default specs are chosen based on the series frequency.
<code>use_fivebest</code>	Logical. If TRUE, a small set of data-driven "five best" specs (as in <code>seasonal::seas(x, x11 = "")</code>) is added to the candidate grid.
<code>max_specs</code>	Optional positive integer limiting the number of candidate ARIMA specifications fitted after de-duplication and seasonal filtering.
<code>seasonal_only</code>	Logical. If TRUE (default), candidate ARIMA specifications must have a non-zero seasonal order ($P + D + Q > 0$).
<code>auto_outliers</code>	Logical. If TRUE, enables automatic outlier detection in the candidate models.
<code>transform_fun</code>	Transformation applied to the input series before seasonal adjustment. "auto" selects a safe log transform if the series is strictly positive, "log" forces a log transform, and "none" keeps the series in levels.
<code>td_candidates</code>	Optional named list of alternative trading-day regressors, e.g. <code>list(wd = wd.m, wd1 = wd1.m)</code> . Each element should be a time series aligned with <code>y</code> .
<code>td_usertype</code>	Character string passed as <code>regression.usertype</code> when <code>xreg</code> is used (default "td").
<code>include_easter</code>	Controls inclusion of Easter regressors: "auto" (default) asks X-13 to select among its supported Easter windows or reject Easter, "always" fixes <code>easter[easter_len]</code> , and "off" neither includes nor tests Easter. A logical value is also accepted and mapped to "auto"/"off".
<code>easter_len</code>	Integer, length (in days) of the Easter effect imposed by <code>include_easter = "always"</code> . It is ignored in "auto" and "off" modes.

include_history_top_n	Integer, number of top-ranked models for which revision metrics over the history are computed before the final ranking step.
early_period_end	Reserved for future use. Optional index or date marking the end of an early sample period used for stability checks.
current_model	Optional incumbent seasonal::seas object used as a baseline for distance measures and comparison.
current_sa	Optional baseline seasonally adjusted series. If not supplied, it is extracted from current_model when possible.
current_seasonal	Optional baseline seasonal component. If not supplied, it is extracted from current_model (preferring SEATS, falling back to X-11 if needed).
w_qs	Weight of the QS (seasonality) diagnostics in the composite ranking score.
w_stability	Weight of residual / stability diagnostics (e.g. Ljung-Box) in the composite score.
w_aicc	Weight of the information criterion (AICc) in the composite score.
w_rev	Weight of the revision metric (historical revision MAE) in the composite score.
w_dist_sa	Weight of the distance between seasonally adjusted series (candidate vs baseline) in the composite score.
w_dist_seas	Weight of the distance between seasonal components (candidate vs baseline) in the composite score.
w_engine	Weight / penalty term related to the decomposition engine (SEATS vs X-11) in the composite score.
engine	Engine preference for candidate models. One of "seats", "x11" or "auto" (tries both and lets the ranking decide, with a small penalty for deviations from the preferred engine).
outlier_types	Character vector of outlier types to detect, typically a subset of c("A0", "LS", "TC").
outlier_method	Character scalar giving the outlier detection method (passed to seasonal::seas(), e.g. "AddOne").
outlier_critical	Numeric critical value for outlier detection (e.g. t- or z-threshold).
outlier_alpha	Optional numeric significance level for outlier detection. If supplied, it is converted to a two-sided normal cutoff for outlier_critical.
comparison_mode	Comparison design. "full_search" (default) lets every candidate select its own regressors and outliers. "incumbent_fixed" requires current_model, retains its selected transformation and effective regression variables, and varies only the requested ARIMA specification and decomposition engine. Stored user regressors must be available in the incumbent model.

Details

Ranking combines residual seasonality on the adjusted series (QS_p), Ljung-Box residual diagnostics, AICc, revision metrics for the top candidates, distance from an incumbent model when supplied, and a small engine-preference penalty.

Value

An object of class "auto_seasonal_analysis" with components such as best (best seas model), table (diagnostic and ranking table, including actual Easter inclusion in with_easter and its selected window in easter_window, SEATS substitution metadata in SEATS_model_switch and SEATS_operative_model, and peak-to-trough seasonal amplitude in seasonal_amp_pct. For log models this is the seasonal-factor range in percentage points; for level models it is the additive range relative to the median absolute series level), specs_tried, frequency, transform, baseline, and seasonality. When current_model is supplied, baseline includes its diagnostics and an indicator of whether its AICc is comparable with the selected candidate. comparison_mode and regressor_comparison describe whether candidate AICc differences hold the incumbent regressors fixed.

Examples

```
if (requireNamespace("seasonal", quietly = TRUE)) {
  res <- auto_seasonal_analysis(
    AirPassengers,
    max_specs = 3,
    include_history_top_n = 2
  )
  res$seasonality$overall
}
```

 build_user_xreg

Build user-supplied calendar regressors (Genhol-style)

Description

Helper to create *generic* calendar regressors without relying on any country-specific datasets shipped with the package. This is deliberate: production holiday calendars should usually come from an authoritative institutional source, not from a generic package dataset.

Usage

```
build_user_xreg(
  y,
  td_candidates = NULL,
  holidays = NULL,
  frequency = NULL,
  td_usertype = "holiday",
  default_center = c("calendar", "mean", "none")
)
```

Arguments

<code>y</code>	A <code>ts</code> (or <code>ts-boxable</code>) target series used for alignment.
<code>td_candidates</code>	Optional named list of precomputed regressors (<code>ts-boxable</code>).
<code>holidays</code>	Optional list defining moving-holiday regressors. Each element may be: • a <code>Date</code> vector (holiday dates), or • a list with fields <code>dates</code> (<code>Date</code>), <code>start</code> (<code>int</code>), <code>end</code> (<code>int</code>), <code>center</code> (<code>chr</code>), <code>name</code> (<code>chr</code>) Windows are in days relative to the holiday date. Defaults: <code>start=-7</code>, <code>end=0</code>. Names on the holidays list are preserved when <code>name</code> is not supplied inside an element.
<code>frequency</code>	Optional frequency to use for holiday regressors (defaults to <code>frequency(y)</code>).
<code>td_usertype</code>	X-13 usertype label for <i>all</i> returned regressors (default "holiday").
<code>default_center</code>	Default centering for holiday regressors.

Details

Supports:

- Precomputed regressors (`td_candidates`): named list of `ts-boxable` series
- Moving-holiday regressors (`holidays`): built via `seasonal::genhol()`

The output is directly consumable by `auto_seasonal_analysis()` via its `td_candidates` argument.

Value

A named list of `ts` regressors aligned to `y`. Attribute `td_usertype` is attached for downstream use.

Examples

```
diwali_dates <- as.Date(c("2022-10-24", "2023-11-12", "2024-11-01"))
xreg <- build_user_xreg(
  y = AirPassengers,
  holidays = list(diwali = list(dates = diwali_dates, start = -1, end = 1)),
  td_usertype = "holiday"
)
names(xreg)
```

<code>extract_outliers</code>	<i>Extract X-13 outlier coefficients (AO/LS/TC) from a seas model</i>
-------------------------------	---

Description

Extract X-13 outlier coefficients (AO/LS/TC) from a seas model

Usage

```
extract_outliers(m)
```

Arguments

`m` A fitted `seasonal::seas` model.

Value

A tibble with columns `type`, `period`, `coef`. Empty when none found.

Examples

```
if (requireNamespace("seasonal", quietly = TRUE)) {
  m <- seasonal::seas(AirPassengers)
  extract_outliers(m)
}
```

`plot.auto_seasonal_analysis`

Plot an automatic seasonal-analysis result

Description

Provides a compact default plot for the main result class returned by `auto_seasonal_analysis()`. The plot is intended for quick inspection; the HTML report remains the richer review artifact.

Usage

```
## S3 method for class 'auto_seasonal_analysis'
plot(
  x,
  series = c("final", "seasonal", "original"),
  xlab = "time",
  ylab = NULL,
  ...
)
```

Arguments

`x` An object returned by `auto_seasonal_analysis()`.

`series` Which series to show. "final" plots the selected seasonally adjusted series when available and falls back to the original series for do-not-adjust cases. "seasonal" plots the selected seasonal component when available. "original" plots the original input series stored in the result.

`xlab, ylab` Axis labels.

`...` Additional arguments passed to `stats::plot.ts()`.

Value

Invisibly, x.

Examples

```
if (requireNamespace("seasonal", quietly = TRUE)) {
  res <- auto_seasonal_analysis(AirPassengers, max_specs = 3)
  plot(res)
}
```

probe_case

Probe one seasonal-adjustment case

Description

Convenience helper for reproducing a problematic single series. It writes an HTML report and returns the normalized output path invisibly.

Usage

```
probe_case(
  y,
  outfile = tempfile("seasight-probe-", fileext = ".html"),
  verbose = TRUE,
  report_fun = sa_report_html,
  ...
)
```

Arguments

y	Series to analyse.
outfile	Output HTML path. Defaults to a temporary file.
verbose	If TRUE, print the output path.
report_fun	Report function. Defaults to sa_report_html() .
...	Additional arguments passed to report_fun.

Value

Invisibly, the normalized output path.

Examples

```
fake_report <- function(y, outfile, ...) {
  writeLines("<html><body>probe</body></html>", outfile)
  list(report = outfile)
}
probe_case(
  y = AirPassengers,
  outfile = tempfile("probe-", fileext = ".html"),
  verbose = FALSE,
  report_fun = fake_report
)
```

sa_align_regressor	<i>Align a regressor to a target series</i>
--------------------	---

Description

Standardizes user-supplied regressors before they are passed to X-13. The helper checks frequency, windows the regressor to the span of y, and returns NULL if the result cannot be aligned exactly.

Usage

```
sa_align_regressor(y, x)
```

Arguments

y	Target series, coercible to ts.
x	Regressor series, coercible to ts.

Value

A ts object aligned to y, or NULL when alignment fails.

Examples

```
y <- AirPassengers
x <- stats::lag(y, -1)
aligned <- sa_align_regressor(y, x)
stats::frequency(aligned)
```

sa_batch_run

*Run seasonal-adjustment reports over a list of series***Description**

Sequential batch runner with per-series status rows and a structured CSV log. Existing outputs can be skipped via `resume = TRUE`.

Usage

```
sa_batch_run(
  series,
  out_dir = "sa_reports",
  ids = NULL,
  resume = TRUE,
  timeout = Inf,
  report_fun = sa_report_html,
  log_file = NULL,
  ...
)
```

Arguments

<code>series</code>	A <code>ts</code> object or a non-empty list of series.
<code>out_dir</code>	Directory where HTML reports and <code>_sa_run_log.csv</code> are written.
<code>ids</code>	Optional character vector of IDs, same length as <code>series</code> .
<code>resume</code>	If <code>TRUE</code> , existing output files are skipped and logged as skipped.
<code>timeout</code>	Per-series timeout in seconds. Use <code>Inf</code> to run in the current R process without a timeout. Finite values require the suggested <code>callr</code> package and run each case in a clean subprocess.
<code>report_fun</code>	Report function to call per item. Defaults to <code>sa_report_html()</code> .
<code>log_file</code>	Optional CSV log path. Defaults to <code>_sa_run_log.csv</code> in <code>out_dir</code> .
<code>...</code>	Additional arguments passed to <code>report_fun</code> .

Value

A tibble with one row per series: `id`, `ok`, `status`, `reason`, `message`, `outfile`, and `elapsed_seconds`.

Examples

```
fake_report <- function(y, outfile, ...) {
  writeLines("<html><body>ok</body></html>", outfile)
  list(report = outfile)
}
out <- sa_batch_run(
  series = list(series1 = AirPassengers),
```

```

    out_dir = tempdir(),
    report_fun = fake_report
  )
  out$status

```

sa_compare

*Compare an incumbent model with the selected best model***Description**

Summarises the incumbent `current_model` against the best model stored in an [auto_seasonal_analysis\(\)](#) result. The comparison is intentionally compact: it reports key diagnostics, the switching decision from [sa_should_switch\(\)](#), and aligned seasonally adjusted and seasonal-component series when they are available.

Usage

```
sa_compare(res, current_model)
```

Arguments

`res` Result from `auto_seasonal_analysis()`.

`current_model` A fitted seas object to compare against.

Value

A list of class "seasight_sa_compare" with elements: `decision`, `summary`, `series`, `diagnostics`, and `table`. The decision uses the incumbent-relative rule documented in [sa_should_switch\(\)](#).

Examples

```

if (requireNamespace("seasonal", quietly = TRUE)) {
  current_model <- seasonal::seas(AirPassengers)
  res <- auto_seasonal_analysis(
    y = AirPassengers,
    current_model = current_model,
    max_specs = 3
  )
  sa_compare(res, current_model)
}

```

sa_copyable_call	<i>Build a copy-pasteable seas() call from a fitted model</i>
------------------	---

Description

- SEATS (default engine) -> omit x11/seats args
- X-11 -> include x11 = ""

Usage

```
sa_copyable_call(
  m,
  x_expr,
  xreg_expr = NA,
  include_force = FALSE,
  engine = c("auto", "seats", "x11")
)
```

Arguments

m	Fitted seas model.
x_expr	Character code for x= (e.g. "y").
xreg_expr	NULL to drop, NA to keep as-is, or character code to set.
include_force	Logical; if TRUE adds force.type = "denton".
engine	"auto" (use model's engine) or force "seats"/"x11".

Value

A single string containing the seas(...) call.

Examples

```
if (requireNamespace("seasonal", quietly = TRUE)) {
  m <- seasonal::seas(AirPassengers)
  sa_copyable_call(m, x_expr = "AirPassengers")
}
```

sa_engine_choice_card *Engine choice rationale card (SEATS vs X-11)*

Description

Explains, in plain language, why the selected engine was chosen for the best model.

Usage

```
sa_engine_choice_card(res)
```

Arguments

res Object from `auto_seasonal_analysis()`.

Value

An htmltools tag (card).

Examples

```
if (requireNamespace("seasonal", quietly = TRUE)) {
  res <- auto_seasonal_analysis(AirPassengers, max_specs = 3)
  sa_engine_choice_card(res)
}
```

sa_existence_call *Aggregate existence-of-seasonality decision (ADJUST / BORDER-LINE / DO_NOT_ADJUST)*

Description

Uses IDS, M7 (strong/weak), QS on SA, and QS on ORIGINAL (QSori) to form a simple majority-style decision. Thresholds mirror those used elsewhere.

Usage

```
sa_existence_call(
  tbl,
  majority = 0.6,
  thr = list(qs = 0.1, m7_strong = 0.9, m7_weak = 1.05, borderline_min_share = 0.4,
    use_qsori = TRUE)
)
```

Arguments

tbl	Candidate table (e.g., res\$table).
majority	Required share for a positive overall call (default 0.6).
thr	List of thresholds: qs (0.10), m7_strong (0.90), m7_weak (1.05), borderline_min_share (0.4), use_qsori (TRUE).

Value

A tibble with call_overall and shares per test family.

Examples

```
tbl <- tibble::tibble(
  QS_p = c(0.04, 0.20, 0.08),
  M7 = c(0.82, 1.10, 0.95),
  IDS = c("yes", "no", "yes"),
  QSori_p = c(0.03, 0.15, 0.07)
)
sa_existence_call(tbl)
```

sa_existence_card	<i>Existence-of-seasonality narrative (HTML tag)</i>
-------------------	--

Description

Uses IDS, M7, QS on original, and presence/absence of a SEATS seasonal component to provide a short textual assessment of seasonality for the best model.

Usage

```
sa_existence_card(res)
```

Arguments

res	Result of <code>auto_seasonal_analysis()</code> .
-----	---

Value

An htmltools tag representing the card.

Examples

```
if (requireNamespace("seasonal", quietly = TRUE)) {
  res <- auto_seasonal_analysis(AirPassengers, max_specs = 3)
  sa_existence_card(res)
}
```

sa_is_do_not_adjust	<i>Row-level non-adjustment rule (wrapper)</i>
---------------------	--

Description

Delegates to the internal `.do_not_adjust()` to keep logic in one place.

Usage

```
sa_is_do_not_adjust(row)
```

Arguments

row	One-row tibble from <code>res\$table</code> .
-----	---

Details

The weak-amplitude threshold is one percentage point of peak-to-trough seasonal amplitude. For log models, the amplitude is the seasonal-factor range multiplied by 100. For level models, it is the additive seasonal range relative to the median absolute series level.

Value

TRUE if the row satisfies the "do not adjust" rule, FALSE otherwise.

Examples

```
row <- tibble::tibble(
  IDS = "no",
  M7 = 1.10,
  QSori_p_x11 = 0.12,
  QSori_p_seats = 0.15,
  SEATS_has_seasonal = FALSE,
  seasonal_amp_pct = 0.8,
  vola_reduction_pct = 2
)
sa_is_do_not_adjust(row)
```

sa_report_html	<i>Seasonal Adjustment: full HTML report</i>
----------------	--

Description

User-facing wrapper that runs the automatic seasonal analysis and writes a human-readable HTML report with plots, tables, diagnostics, and copy-pasteable `seas()` calls. The report fits the same candidate grid as [auto_seasonal_analysis\(\)](#) and can therefore take noticeable time on broad grids. Use `max_specs` for quick exploratory reports.

Usage

```
sa_report_html(
  y,
  current_model = NULL,
  td_usertype = "td",
  td_candidates = NULL,
  use_fivebest = TRUE,
  max_specs = NULL,
  seasonal_only = TRUE,
  title = "Seasonal Adjustment Report",
  outfile = "sa_report.html",
  file = NULL,
  png_width = 1400,
  png_height = 900,
  print_to_console = FALSE,
  print_which = c("new", "current", "both"),
  include_easter = c("auto", "always", "off"),
  easter_len = 15L,
  engine = c("seats", "x11", "auto"),
  w_engine = 1,
  outlier_types = c("AO", "LS", "TC"),
  outlier_method = "AddOne",
  outlier_critical = 4,
  outlier_alpha = NULL,
  comparison_mode = c("full_search", "incumbent_fixed")
)
```

Arguments

<code>y</code>	Time series to be analysed. Can be a <code>ts</code> object or anything that <code>tsbox::ts_ts()</code> can convert to <code>ts</code> .
<code>current_model</code>	Optional incumbent <code>seasonal::seas()</code> model to compare against the newly selected specification.
<code>td_usertype</code>	Character string passed as <code>regression.usertype</code> when trading-day regressors are used (default <code>"td"</code>).
<code>td_candidates</code>	Optional named list of trading-day candidate regressors (e.g. <code>list(wd = wd.m, wd1 = wd1.m)</code>), each aligned with <code>y</code> .
<code>use_fivebest</code>	Logical. If <code>TRUE</code> , include the "five best" automatic specs in the candidate grid.
<code>max_specs</code>	Optional positive integer limiting the number of candidate ARIMA specifications fitted.
<code>seasonal_only</code>	Logical. If <code>TRUE</code> (default), filter candidate ARIMA specifications to seasonal models before fitting.
<code>title</code>	Title of the report shown in the HTML page.
<code>outfile</code>	Path to the HTML file to be written.
<code>file</code>	Optional alias for <code>outfile</code> , for compatibility with callers that use file-oriented naming.

png_width, png_height	Width and height (in pixels) of PNG plots embedded in the report.
print_to_console	Logical. If TRUE, do not create an HTML file but print the copy-pasteable model code to the console and return it.
print_which	Which code blocks to print when print_to_console = TRUE. One of "new", "current" or "both".
include_easter	Controls inclusion of Easter regressors: "auto" (default) asks X-13 to select among its supported Easter windows or reject Easter, "always" fixes easter[easter_len], and "off" neither includes nor tests Easter. A logical value is also accepted and mapped to "auto"/"off".
easter_len	Integer, length (in days) of the Easter effect imposed by include_easter = "always". It is ignored in "auto" and "off" modes.
engine	Preferred decomposition engine for candidate models. One of "seats", "x11" or "auto".
w_engine	Numeric weight for the engine choice component in the composite ranking score.
outlier_types	Character vector of outlier types to detect, typically a subset of c("A0", "LS", "TC").
outlier_method	Character scalar giving the outlier detection method passed to seasonal::seas() (e.g. "AddOne").
outlier_critical	Numeric critical value for outlier detection (e.g. a t-/z-threshold).
outlier_alpha	Optional numeric significance level; if supplied, it is converted to a two-sided normal cutoff for outlier_critical.
comparison_mode	Comparison design passed to auto_seasonal_analysis() . "full_search" lets candidates select their own regressors and outliers; "incumbent_fixed" retains the incumbent's transformation and regression variables while varying ARIMA and engine.

Details

Notes:

- If the provided *current* specification equals the selected *best* model, the report omits the "Alternative model" comparison section.
- A reported SEATS model substitution is highlighted in both the summary and engine-choice cards, with requested and operative models when known.
- The "Top candidates" table starts with the best model and always includes the current model (flagged as current) when one is supplied.

Value

Invisibly, a list with elements `report` (path to the HTML file) and `res` (the corresponding [auto_seasonal_analysis\(\)](#) result).

See Also

[auto_seasonal_analysis\(\)](#), [sa_existence_card\(\)](#), [sa_top_candidates_table\(\)](#)

Examples

```
if (requireNamespace("seasonal", quietly = TRUE)) {
  sa_report_html(
    y = AirPassengers,
    outfile = tempfile(fileext = ".html"),
    max_specs = 3
  )
}
```

sa_should_switch	<i>Should we switch to the new best model?</i>
------------------	--

Description

Compares the first row of `res$table` with the incumbent model recorded by [auto_seasonal_analysis\(\)](#). The candidate must improve comparable AICc by a material amount, pass absolute residual diagnostics, avoid material deterioration relative to the incumbent, and satisfy the existing distance and seasonal-correlation safeguards. Identical adjusted series are kept.

Usage

```
sa_should_switch(
  res,
  thresholds = list(min_qs_p = 0.1, max_dist_sa_mult = 1.25, min_corr_seas = 0.9,
    min_lb_p = 0.05, min_delta_aicc = 2, max_qs_p_drop = 0.05, max_lb_p_drop = 0.05,
    identical_sa_tolerance = 1e-08),
  current_model = NULL,
  details = FALSE
)
```

Arguments

res	Result of auto_seasonal_analysis() .
thresholds	Named list with decision thresholds: • <code>min_qs_p</code>: minimum acceptable QS p-value on SA (overall) for the best model • <code>max_dist_sa_mult</code>: allow SA L1 distance up to this multiple of the cross-candidate median • <code>min_corr_seas</code>: minimum correlation of seasonal components (vs. incumbent) • <code>min_lb_p</code>: minimum acceptable Ljung-Box p-value on residuals

	• <code>min_delta_aicc</code>: minimum incumbent-minus-candidate AICc improvement • <code>max_qs_p_drop</code>: maximum allowed decline in QS p-value vs. incumbent • <code>max_lb_p_drop</code>: maximum allowed decline in Ljung-Box p-value vs. incumbent • <code>identical_sa_tolerance</code>: relative tolerance used to identify equal adjusted series
<code>current_model</code>	Optional fitted <code>seasonal::seas</code> incumbent. This is useful for older result objects that do not contain baseline diagnostics.
<code>details</code>	Logical. If FALSE (default), return only the decision string. If TRUE, return a list containing decision, reason, and metrics.

Value

One of "CHANGE_TO_NEW_MODEL", "KEEP_CURRENT_MODEL", "REVIEW_REQUIRED", or "NO_BASELINE"; with `details = TRUE`, a structured list containing the decision and its supporting evidence.

Examples

```
if (requireNamespace("seasonal", quietly = TRUE)) {
  current_model <- seasonal::seas(AirPassengers)
  res <- auto_seasonal_analysis(
    AirPassengers,
    current_model = current_model,
    max_specs = 3
  )
  sa_should_switch(res)
}
```

sa_tests_model	<i>Key seasonality diagnostics for a fitted model</i>
----------------	---

Description

Returns M7, IDS, QS on SA (X-11 & SEATS, plus overall min), QS on original (X-11 & SEATS, plus overall min), and Ljung-Box p-value on residuals.

Usage

```
sa_tests_model(m)
```

Arguments

`m` A fitted `seasonal::seas` object.

Details

Columns:

- M7, IDS
- QS_p_x11, QS_p_seats, QS_p (QS on SA)
- QSori_p_x11, QSori_p_seats, QSori_p (QS on original)
- LB_p

For backwards compatibility, the output also includes the legacy aliases QS_SA_x11, QS_SA_seats, QS_SA, QSori_x11, QSori_seats, QSori.

Value

A one-row tibble with the diagnostics above.

Examples

```
if (requireNamespace("seasonal", quietly = TRUE)) {
  m <- seasonal::seas(AirPassengers)
  sa_tests_model(m)
}
```

sa_top_candidates_table
<i>Top candidates table (HTML tag)</i>

Description

Build a colour-coded HTML table of the highest-ranked candidate models, always placing the **best** model at the top and **including the current model** (if supplied) even if it is not in the top n. The airline reference model ARIMA (0 1 1)(0 1 1) is also appended if absent.

Usage

```
sa_top_candidates_table(res, current_model = NULL, y = NULL, n = 5)
```

Arguments

res	Result of <code>auto_seasonal_analysis()</code> .
current_model	Optional fitted <code>seasonal::seas</code> model to mark as "current".
y	Optional original series (needed to compute a few current-model diagnostics).
n	Number of top rows to display (default 5). The airline/current rows may be appended even if not in the top n.

Details

Rows are lightly shaded: best (green), current (blue), airline (red). The table reports the requested ARIMA, the operative SEATS model when available, and a stable yes/no/n/a SEATS-switch indicator.

Value

An `htmltools` tag (`<table>`) you can insert into reports.

Examples

```
if (requireNamespace("seasonal", quietly = TRUE)) {  
  current_model <- seasonal::seas(AirPassengers)  
  res <- auto_seasonal_analysis(  
    y = AirPassengers,  
    current_model = current_model,  
    max_specs = 3  
  )  
  sa_top_candidates_table(res, current_model = current_model, y = AirPassengers, n = 5)  
}
```

Index

`auto_seasonal_analysis`, 2
`auto_seasonal_analysis()`, 6, 7, 11, 13–15, 17, 18, 20

`build_user_xreg`, 5

`extract_outliers`, 6

`plot.auto_seasonal_analysis`, 7
`probe_case`, 8

`sa_align_regressor`, 9
`sa_batch_run`, 10
`sa_compare`, 11
`sa_copyable_call`, 12
`sa_engine_choice_card`, 13
`sa_existence_call`, 13
`sa_existence_card`, 14
`sa_existence_card()`, 18
`sa_is_do_not_adjust`, 15
`sa_report_html`, 15
`sa_report_html()`, 8, 10
`sa_should_switch`, 18
`sa_should_switch()`, 11
`sa_tests_model`, 19
`sa_top_candidates_table`, 20
`sa_top_candidates_table()`, 18
`seasonal::genhol()`, 6
`seasonal::seas`, 7, 19, 20
`seasonal::seas()`, 16
`stats::plot.ts()`, 7