# Package ‘sautilities’

July 5, 2022

**Title** Seasonal Adjustment Utilities For Use With the Seasonal Package

**Version** 3.1

**Description** Several utilities to provide support for the seasonal package. This includees routines
that select the X-11 seasonal filter based on the magnitude of the estimate of the
seasonal moving average coefficient from the airline model, duplicates the functionality of the
TERROR software that performs quality control on time series based on one step ahead forecasts,
generate model summaries from seas objects, generate names and abbreviations for X-
13ARIMA-SEATS
tables, save spec files, seasonal objects, and metafiles into external files, process list objects
of numbers, indicate which elements of a list have try-errors, replace NA with a string, set outlier
critical values, add an outlier spec to a static seas element, get indexes and en-
tries from UDG output
generated from seasonal, save seasonal objects into R scripts and X-13ARIMA-SEATS spec files,
and functions to collect diagnostics summaries for various X-13ARIMA-SEATS diagnostics.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.2.0

**Depends** R (>= 2.10)

**Imports** plotrix, seasonal, stringr, xlsx

**NeedsCompilation** no

**Author** Brian Monsell [aut, cre]

**Maintainer** Brian Monsell <monsell.brian@bls.gov>

## R topics documented:

---

`absmax` *Maximum absolute value of a vector*

---

## Description

Generates the maximum of the absolute value of a numeric vector

## Usage

```
absmax(x)
```

## Arguments

| | |
|---|---|
| x | vector of numbers |

### Value

Maximum of the absolute value of a vector

### Examples

```
r50 <- rnorm(50)
r50.absmax <- absmax(r50)
```

---

## `acf_fail` *ACF Test failure message*

---

### Description

Tests whether the sample autocorrelation of the residuals from a time series model fails the Ljung-Box or Box-Pierce Q test

### Usage

```
acf_fail(udg_list = NULL, acf_lags_fail = c(1, 2, 3, 4, 12, 24), num_sig = 8)
```

### Arguments

| | |
|---|---|
| `udg_list` | • list object generated by `udg()` function of the `seasonal` package. |
| `acf_lags_fail` | • lags of the ACF to test |
| `num_sig` | • limit for number of lags with significant ACF values |

### Value

Logical object which is TRUE if series fails the ACF test, FALSE otherwise

### Examples

```
ukgas_seas <- seasonal::seas(UKgas, series.period = 4, arima.model = '(0 1 1)(0 1 1)',
                             x11='', transform.function = 'log', forecast.maxlead=20,
                             check.print = c( 'pacf', 'pacfplot' ))
ukgas_udg <- seasonal::udg(ukgas_seas)
ukgas_acf_fail <- acf_fail(ukgas_udg, acf_lags_fail = c(1, 2, 3, 4, 8), num_sig = 4)
```

## acf_fail_why — *ACF Test Explanation*

### Description

ACF Test Failure Message

### Usage

```
acf_fail_why(
  udg_list = NULL,
  acf_lags_fail = c(1, 2, 3, 4, 12, 24),
  num_sig = 8,
  return_both = FALSE
)
```

### Arguments

| | |
|---|---|
| `udg_list` | • list object generated by `udg()` function of the `seasonal` package. |
| `acf_lags_fail` | • lags of the ACF to test |
| `num_sig` | • limit for number of lags with significant ACF values |
| `return_both` | Logical scalar indicating whether the calling function will return both the test results and why the test failed or just produce a warning. Default is FALSE. |

### Details

Generates text on why the sample autocorrelation of the residuals from a time series model fails the Ljung-Box or Box-Pierce Q test

### Value

character object tells why series fails the ACF test, 'pass' otherwise.

### Examples

```
ukgas_seas <- seasonal::seas(UKgas, series.period = 4, arima.model = '(0 1 1)(0 1 1)',
                             x11='', transform.function = 'log', forecast.maxlead=20,
                             check.print = c( 'pacf', 'pacfplot' ))
ukgas_udg <- seasonal::udg(ukgas_seas)
ukgas_acf_fail_why <- acf_fail_why(ukgas_udg, acf_lags_fail = c(1, 2, 3, 4, 8),
                                   num_sig = 4, return_both = TRUE)
```

## acf_test *Global ACF test*

### Description

Tests whether the residuals from a time series model has acceptable autocorrelation in the residuals.

### Usage

```
acf_test(
  seas_obj = NULL,
  num_sig = 8,
  acf_lags_fail = c(1, 2, 3, 4, 12, 24),
  acf_lags_warn = c(12, 24),
  return_this = "test"
)
```

### Arguments

| | |
|---|---|
| `seas_obj` | • object generated by seas() of the seasonal package. |
| `num_sig` | • limit for number of lags with significant ACF values |
| `acf_lags_fail` | • lags of the ACF to test |
| `acf_lags_warn` | • lags of the ACF to test for warnings |
| `return_this` | character string; what the function returns - 'test' returns test results, 'why' returns why the test failed or received a warning, or 'both' |

### Value

A text string denoting if series passes, fails, or has a warning for residual autocorrelation. If model diagnostics not found, return 'none'.

### Examples

```
ukgas_seas <- seasonal::seas(UKgas, series.period = 4, arima.model = '(0 1 1)(0 1 1)',
                             x11='', transform.function = 'log', forecast.maxlead=20,
                             check.print = c( 'pacf', 'pacfplot' ))
ukgas_acf_test <- acf_test(ukgas_seas, num_sig = 4, acf_lags_fail = c(1, 2, 3, 4, 8),
                           acf_lags_warn = c(4, 8), return_this = 'both')
```

## acf_warn *ACF test warning message*

### Description

Tests whether the residuals from a time series model generates a warning for the AIC test

### Usage

```
acf_warn(udg_list = NULL, acf_lags_warn = c(12, 24))
```

### Arguments

| | |
|---|---|
| `udg_list` | • list object generated by `udg()` function of the `seasonal` package. |
| `acf_lags_warn` | • lags of the ACF to test for warnings |

### Value

Logical object which is TRUE if series generates a warning for the ACF test, FALSE otherwise

### Examples

```
ukgas_seas <- seasonal::seas(UKgas, series.period = 4, arima.model = '(0 1 1)(0 1 1)',
                             x11='', transform.function = 'log', forecast.maxlead=20,
                             check.print = c( 'pacf', 'pacfplot' ))
ukgas_udg <- seasonal::udg(ukgas_seas)
ukgas_acf_warn <- acf_warn(ukgas_udg, acf_lags_warn = c(4,8))
```

---

## `acf_warn_why` *ACF Test Warning Message*

---

### Description

Generates text on why the sample autocorrelation of the residuals from a time series model fails the Ljung-Box or Box-Pierce Q test

### Usage

```
acf_warn_why(udg_list, acf_lags_warn = c(12, 24), return_both = FALSE)
```

### Arguments

| | |
|---|---|
| `udg_list` | • list object generated by `udg()` function of the `seasonal` package. |
| `acf_lags_warn` | • lags of the ACF to test for warnings |
| `return_both` | Logical scalar indicating whether the calling function will return both the test results and why the test failed or produced a warning. Default is FALSE. |

### Value

character string which tells why the series generates a warning for the ACF test, 'pass' otherwise.

### Examples

```
ukgas_seas <- seasonal::seas(UKgas, series.period = 4, arima.model = '(0 1 1)(0 1 1)',
                             x11='', transform.function = 'log', forecast.maxlead = 20,
                             check.print = c( 'pacf', 'pacfplot' ))
ukgas_udg <- seasonal::udg(ukgas_seas)
ukgas_acf_warn_why <- acf_warn_why(ukgas_udg, acf_lags_warn = c(4, 8), return_both = TRUE)
```

## all_model_diag *Model diagnostic summary*

### Description

Generate a summary of model diagnostics for a single series

### Usage

```
all_model_diag(
  seas_obj = NULL,
  add_aicc = FALSE,
  add_norm = FALSE,
  add_auto_out = FALSE,
  return_list = FALSE
)
```

### Arguments

| | |
|---|---|
| `seas_obj` | `seas` object generated from a call of `seas` on a single time series |
| `add_aicc` | logical scalar; add AICC value to the summary |
| `add_norm` | logical scalar; add normality statistics to the summary |
| `add_auto_out` | logical scalar; add identified automatic outliers to the summary |
| `return_list` | logical scalar; return a list rather than a vector |

### Value

vector or list of model diagnostics for a given series

### Examples

```
air_seas <-
   seasonal::seas(AirPassengers, x11='', slidingspans = '',
                  regression.aictest = 'td', forecast.maxlead=36,
                  check.print = c( 'pacf', 'pacfplot' ))
air_diag <- all_model_diag(air_seas, add_aicc = TRUE, add_norm = TRUE,
                           add_auto_out = TRUE, return_list = TRUE)
```

## all_model_diag_list *Model diagnostic summary from a list*

### Description

Generate a summary of model diagnostics from a list of `seas` objects series

### Usage

```
all_model_diag_list(
  seas_obj_list,
  add_aicc = FALSE,
  add_norm = FALSE,
  add_auto_out = FALSE,
  add_spec = FALSE,
  save_summary = FALSE,
  save_file = "this_excel_file.xlsx",
  save_append = TRUE,
  save_sheetname = "diag"
)
```

### Arguments

| | |
|---|---|
| `seas_obj_list` | list of seas objects generated from a call of seas on a single time series |
| `add_aicc` | logical scalar; add AICC value to the summary |
| `add_norm` | logical scalar; add normality statistics to the summary |
| `add_auto_out` | logical scalar; add identified automatic outliers to the summary |
| `add_spec` | logical scalar; add test for spectral peaks to the summary |
| `save_summary` | logical scalar; save the summary matrix in a separate Excel file |
| `save_file` | character string; file name for saving summary matrix |
| `save_append` | logical scalar; if TRUE, append the sheet to the Excel file, otherwise overwrite the sheet. Default is TRUE |
| `save_sheetname` | character string; sheet name used for the Excel file |

### Value

vector of model diagnostics for a given series

### Examples

```
xt_lauto <- seasonal::seas(xt_data_list, slidingspans = "", transform.function = "log",
                           x11.seasonalma = "msr",
                           arima.model = "(0 1 1)(0 1 1)",
                           forecast.maxlead=36, check.print = c( "pacf", "pacfplot" ))
xt_diag <- all_model_diag_list(xt_lauto, add_aicc = TRUE,
                       add_norm = TRUE, add_auto_out = TRUE,
                       add_spec = TRUE)
```

## check_stats *Displays various X-13 diagnostics*

### Description

Displays various X-13 diagnostics for a single series.

## Usage

```
check_stats(
  seas_obj = NULL,
  print_summary = TRUE,
  test_full = TRUE,
  test_span = TRUE,
  acf_num_sig = 8,
  acf_lags_fail = c(1, 2, 3, 4, 12, 24),
  acf_lags_warn = c(12, 24),
  model_t_value = 3,
  model_p_value = 0.05,
  otl_auto_limit = 5,
  otl_all_limit = 5,
  d11f_p_level = 0.01,
  qs_p_limit_pass = 0.01,
  qs_p_limit_warn = 0.05,
  qs_p_limit_fail = 0.01,
  qs_robust_sa = TRUE,
  sf_limit = 25,
  change_limit = 40,
  mq_fail_limit = 1.2,
  mq_warn_limit = 0.8,
  return_list = FALSE
)
```

## Arguments

| | |
|---|---|
| `seas_obj` | object generated by `seas()` of the seasonal package. |
| `print_summary` | Logical object; print the result of summary(`seas_obj`); if FALSE, a model summary will be printed out |
| `test_full` | Logical scalar indicating whether to apply the QS test to the full series span |
| `test_span` | Logical scalar indicating whether to test the QS test to the final 8-year span used by the spectrum diagnostic |
| `acf_num_sig` | Numeric object; limit for number of lags with significant ACF values |
| `acf_lags_fail` | • Numeric vector; lags of the ACF to test |
| `acf_lags_warn` | • Numeric vector; lags of the ACF to test for warnings |
| `model_t_value` | • t-statistic limit for regressors |
| `model_p_value` | • p-value limit for regressors |
| `otl_auto_limit` | • limit for number of automatically identified outliers |
| `otl_all_limit` | • limit for number of automatically identified outliers |
| `d11f_p_level` | • p-level used to test the d11 f-test for residual seasonality |
| `qs_p_limit_pass` | Numeric scalar; P-value limit for QS statistic for passing |
| `qs_p_limit_warn` | Numeric scalar; P-value limit for QS statistic for warning |
| `qs_p_limit_fail` | Numeric scalar; P-value limit for model based seasonal F-statistic for passing |

| | |
|---|---|
| `qs_robust_sa` | Logical scalar indicating if original series adjusted for extremes is included in testing |
| `sf_limit` | Numeric object; limit for the percentage of seasonal spans flagged |
| `change_limit` | Numeric object; limit for the percentage of month-to-month changes flagged |
| `mq_fail_limit` | • numeric scalar; value above which the M or Q statistic fails; default is 1.2 |
| `mq_warn_limit` | • numeric scalar; value above which the M or Q statistic gives a warning message if it is less than `this_fail_Limit`; default is 0.8 |
| `return_list` | Logical scalar; indicates if the function will return a summary of diagnostics. Default is TRUE. |

## Value

Displays assorted seasonal adjusmtent and modeling diagnostics.

## Examples

```
ukgas_seas <-
   seasonal::seas(UKgas, series.period = 4, arima.model = '(0 1 1)(0 1 1)',
                  x11='', transform.function = 'log', forecast.maxlead=20,
                  check.print = c( 'pacf', 'pacfplot' ))
ukgas_qs_test <-
   check_stats(ukgas_seas, acf_num_sig = 5, acf_lags_fail = c(1, 2, 3, 4, 8),
               acf_lags_warn = c(4, 8), otl_auto_limit = 4, otl_all_limit = 6,
               return_list = TRUE)
```

---

`choose_optimal_seasonal_filter`
*Choose Optimal X-11 seasonal moving average*

---

## Description

Choose the optimal X-11 seasonal moving average based on the value of the seasonal moving average coefficient from an airline model.

## Usage

```
choose_optimal_seasonal_filter(
  this_seasonal_theta = NULL,
  dp_limits = TRUE,
  use_3x15 = TRUE
)
```

## Arguments

| | |
|---|---|
| `this_seasonal_theta` | numeric scalar; seasonal moving average coefficient from an airline model |
| `dp_limits` | logical scalar, if TRUE limits from Deputot and Planas will be used to choose the moving average, else limits from Bell Chow and Chu will be used. Default is TRUE. |
| `use_3x15` | logical scalar, if TRUE 3x15 seasonal filter will be returned if chosen, otherwise function will return a `3x9` value. Default is FALSE. |

## Value

The optimal X-11 seasonal filter, unless the airline model cannot be estimated.

## Examples

```
shoes_seas <- seasonal::seas(shoes2008, x11='', slidingspans = '',
                        transform.function = 'log', x11 = "",
                        arima.model = '(0 1 1)(0 1 1)',
                        regression.aictest = c('td', 'easter'),
                        forecast.maxlead=36, check.print = c( 'pacf', 'pacfplot' ))
shoes_seasonal_MA <- shoes_seas$est$coefficients[["MA-Seasonal-12"]]
this_seasonal  <- choose_optimal_seasonal_filter(shoes_seasonal_MA)
this_seasonal2 <- choose_optimal_seasonal_filter(shoes_seasonal_MA, dp_limits = FALSE)
```

---

`combined_spectrum_test`    *Combined spectrum test from Maravall (2012)*

---

## Description

generate a test for seasonality by combining the results from the AR(30) and Tukey nonparametric spectrums as laid out in Maravall (2012)

## Usage

```
combined_spectrum_test(
  this_seas = NULL,
  this_ar_spec_cv = NULL,
  this_series = "series.adjoriginal",
  take_log = TRUE,
  take_diff = TRUE
)
```

## Arguments

| | |
|---|---|
| `this_seas` | seas object for a single series |
| `this_ar_spec_cv` | List object with two elements - 99 and 95 percent critical values for the frequencies of the AR(30) spectrum as generated by the gen_ar_spec_cv function. |
| `this_series` | character string; the table used to generate the AR(30) spectrum. Default is "b1". |
| `take_log` | logical scalar; indicates if the AR spectrum is generated from the log of the data. Default is TRUE. |
| `take_diff` | logical scalar; indicates if the data is differenced before the AR spectrum is generated. Default is TRUE. |

## Value

TRUE if spectral evidence of seasonality is detected; FALSE if not.

### Examples

```
air_seas <- seasonal::seas(AirPassengers, series.save = "b1",
                   arima.model='(0 1 1)(0 1 1)',
                   forecast.maxlead = 36, slidingspans = '',
                   transform.function = 'log')
this_ar30_spec_cv <- gen_ar_spec_cv(1000, 97, 12)
this_spectrum_test <- combined_spectrum_test(air_seas, this_ar30_spec_cv)
```

---

## `compare_dates` *Date Match*

---

### Description

Compare two dates to see if they match

### Usage

```
compare_dates(this_date, comp_date = NULL)
```

### Arguments

| | |
|---|---|
| `this_date` | Integer array of length 2, a date where the first element is the year and the second element is the month or quarter |
| `comp_date` | Integer array of length 2, a date to comapare to `this_date` |

### Value

a logical scalar; TRUE if the dates match, FALSE if they don't

### Examples

```
match_start <- compare_dates(start(shoes2007), c(1990,1))
```

---

## `convert_date_string_to_date` *Convert date string from UDG output*

---

### Description

convert a date string from the X-13 UDG file to a `c(year,month)` date

### Usage

```
convert_date_string_to_date(this_date_string)
```

### Arguments

| | |
|---|---|
| `this_date_string` | date string usually extracted from the X-13 UDG output |

## Value

integer array of length 2 with the year and month/quarter of from the date string

## Examples

```
air_seas <- seasonal::seas(AirPassengers,
                    arima.model='(0 1 1)(0 1 1)',
                    forecast.maxlead = 36, slidingspans = '',
                    transform.function = 'log')
this_start_spec_string <- seasonal::udg(air_seas, "startspec")
this_start_spec        <- convert_date_string_to_date(this_start_spec_string)
```

---

`d11f_test` *D11 F-test for residual seasonality*

---

## Description

Generates X-11's f-test for residual seasonality in the seasonally adjusted data

## Usage

```
d11f_test(seas_obj = NULL, p_level = 0.01, return_this = "test")
```

## Arguments

| | |
|---|---|
| seas_obj | • object generated by `seas()` of the seasonal package. |
| p_level | • p-level used to test the d11 f-test for residual seasonality |
| return_this | character string; what the function returns - 'test' returns test results, 'why' returns why the test failed or received a warning, or 'both' |

## Value

A text string denoting if series passes or has a warning for residual seasonality. If d11f statistic not found, return 'none'.

## Examples

```
ukgas_seas <- seasonal::seas(UKgas, series.period = 4, arima.model = '(0 1 1)(0 1 1)',
                             x11='', transform.function = 'log', forecast.maxlead=20,
                             check.print = c( 'pacf', 'pacfplot' ))
ukgas_d11f_test <- d11f_test(ukgas_seas, p_level = 0.05, return_this = 'both')
```

## `d11f_test_why` *ACF Test Warning Message*

### Description

Why D11 f-test for residual seasonality fails

### Usage

```
d11f_test_why(udg_list = NULL, p_level = 0.01, return_both = FALSE)
```

### Arguments

| | |
|---|---|
| `udg_list` | • list object generated by `udg()` function of the `seasonal` package. |
| `p_level` | • p-level used to test the d11 f-test for residual seasonality |
| `return_both` | Logical scalar indicating whether the calling function will return both the test results and why the test failed or produced a warning. Default is FALSE. |

### Value

A text string denoting why a series fails or has a warning for residual seasonality. If d11f statistic not found, return 'none'.

### Examples

```
ukgas_seas <-
   seasonal::seas(UKgas, series.period = 4, arima.model = '(0 1 1)(0 1 1)',
                  x11='', transform.function = 'log', forecast.maxlead=20,
                  check.print = c( 'pacf', 'pacfplot' ))
ukgas_udg <- seasonal::udg(ukgas_seas)
ukgas_d11f_why <- d11f_test_why(ukgas_udg, p_level = 0.05, return_both = TRUE)
```

## `fix_diag_list` *Fix Diagnostic List*

### Description

Fix an incomplete diagnostic list by filling in missing elements with NAs

### Usage

```
fix_diag_list(this_test, this_names, return_this = "both")
```

### Arguments

| | |
|---|---|
| `this_test` | list object of a seasonal adjustment or modeling diagnostic |
| `this_names` | character vector; complete set of names to check against |
| `return_this` | character string; what the function returns - 'test' returns test results, 'why' returns why the test failed or received a warning, or 'both' |

### Value

diagnostic list object with missing names filled in

### Examples

```
m7_key  <- get_mq_key("M7")
```

## gen_ao_outlier_ts — *Generate level change regression variable as a ts object*

### Description

Generates a ts object for a AO (point) outlier regressor

### Usage

```
gen_ao_outlier_ts(
  ao_date,
  this_start,
  this_end,
  this_freq = 12,
  return_matrix = TRUE
)
```

### Arguments

| | |
|---|---|
| `ao_date` | Integer vector of length two - dates for AO outlier to be generated |
| `this_start` | Numeric vector; start date of AO outlier regressor generated. |
| `this_end` | Numeric vector; end date of AO outlier regressor generated. |
| `this_freq` | Numeric scalar; frequency of time series. Default: 12, for a monthly series |
| `return_matrix` | Logical scalar; If true, the object returned is a one column time series matrix object. Default: TRUE |

### Value

Generate ts object of a point outlier regressor

### Examples

```
UKgas_ao_date <- c(1970, 2)
UKgas_ao_1970_2 <-
   gen_ao_outlier_ts(UKgas_ao_date, this_start = c(1960, 1), this_end = c(1990, 4),
                     this_freq = 4)
```

## gen_ar_spec_cv *Generate critical values for AR(30) spectrum as in Maravall (2012)*

### Description

Generate critical values for AR(30) spectrum as in Maravall (2012)

### Usage

```
gen_ar_spec_cv(n_sim = 1e+05, series_length = 121, freq = 12)
```

### Arguments

| | |
|---|---|
| `n_sim` | integer scalar; number of simulations; default is 100000 |
| `series_length` | integer scalar; length of each series simulated |
| `freq` | integer scalar; frequency of the time series; default is 12 (monthly). |

### Value

List of critical values for each seasonal frequency for the 95th and 99th percentile.

### Examples

```
ar30_spec_cv <- gen_ar_spec_cv(1000, 97, 12)
```

## gen_hybrid_sa *Generate a hybrid seasonal adjustment*

### Description

Generates a "hybrid" seasonal adjustment by replacing a span of a multiplicative seasonal adjustment with an additive adjustment

### Usage

```
gen_hybrid_sa(this_mult_sa, this_add_sa, this_start_hybrid, this_end_hybrid)
```

### Arguments

| | |
|---|---|
| `this_mult_sa` | time series object of a multiplicative seasonal adjustment |
| `this_add_sa` | time series object of an additive seasonal adjustment |
| `this_start_hybrid` | integer vector of length 2, start of the span where additive adjustments replace multiplicative adjustment |
| `this_end_hybrid` | integer vector of length 2, end of the span where additive adjustments replace multiplicative adjustment |

### Value

time series object with hybrid seasonal adjustment

### Examples

```
air_mult_seas <- seasonal::seas(AirPassengers, transform.function = "log")
air_mult_sa   <- seasonal::final(air_mult_seas)
air_add_seas  <- seasonal::seas(AirPassengers, transform.function = "none")
air_add_sa    <- seasonal::final(air_add_seas)
air_hybrid_sa <- gen_hybrid_sa(air_mult_sa, air_add_sa, c(1956,1), c(1956,12))
```

---

## `gen_ls_outlier_ts` *Generate level change regression variable as a ts object*

---

### Description

Generates a ts object for a LS (level shift) outlier regressor

### Usage

```
gen_ls_outlier_ts(
  ls_date,
  this_start,
  this_end,
  this_freq = 12,
  x13type = TRUE,
  return_matrix = TRUE
)
```

### Arguments

| | |
|---|---|
| `ls_date` | Integer vector of length two - dates for LS outlier to be generated |
| `this_start` | Numeric vector; start date of LS outlier regressor generated. |
| `this_end` | Numeric vector; end date of LS outlier regressor generated. |
| `this_freq` | Numeric scalar; frequency of time series. Default: 12, for a monthly series |
| `x13type` | Logical scalar; Indicates if level change outlier is defined as in X-13ARIMA-SEATS. Default: TRUE |
| `return_matrix` | Logical scalar; If true, the object returned is a one column time series matrix object. Default: TRUE |

### Value

Generate ts object of a level change outlier regressor

### Examples

```
UKgas_ls_date <- c(1970, 2)
UKgas_ls_1970_2 <-
   gen_ls_outlier_ts(UKgas_ls_date, this_start = c(1960, 1), this_end = c(1990, 4),
                     this_freq = 4)
```

## `gen_tc_outlier_ts` *Generate temporary change outlier regression as a ts object*

### Description

Generates a ts object for a TC (temporary change) outlier regressor

### Usage

```
gen_tc_outlier_ts(
  tc_date,
  this_start,
  this_end,
  this_freq = 12,
  tc_alpha = NULL,
  return_matrix = TRUE
)
```

### Arguments

| | |
|---|---|
| `tc_date` | Integer vector of length two - dates for TC outlier to be generated |
| `this_start` | Numeric vector; start date of TC outlier regressor generated. |
| `this_end` | Numeric vector; end date of TC outlier regressor generated. |
| `this_freq` | Numeric scalar; frequency of time series. Default: 12, for a monthly series |
| `tc_alpha` | Numeric scalar; Rate of decay for the TC outlier. Default: will be computed as in `X-13ARIMA-SEATS` for a weekly series |
| `return_matrix` | Logical scalar; If true, the object returned is a one column time series matrix object. Default: TRUE |

### Value

ts object for a temporary change outlier regressor

### Examples

```
UKgas_tc_date <- c(1970, 2)
UKgas_tc_1970_2 <-
   gen_tc_outlier_ts(UKgas_tc_date, this_start = c(1960, 1), this_end = c(1990, 4),
                     this_freq = 4, tc_alpha = 0.9, return_matrix = TRUE)
```

## `gen_x13_table_list` *X-13 Tables Available*

### Description

generates a list of X-13 tables that can be extracted with the `seasonal` package

### Usage

```
gen_x13_table_list(this_table_type = "all")
```

### Arguments

| | |
|---|---|
| `this_table_type` | vector of character strings listing types of X-13 tables to output. Default is `'all'`, other choices are `'diagnostics'`, `'matrices'`, `'spectrum'`, `'timeseries'` |

### Value

A list of arrays with table names and abbreviations from X-13ARIMA-SEATS in several different elements specified by the user: diagnostics, matrices, spectrum, timeseries

### Examples

```
x13_tables_all <- gen_x13_table_list()
```

## `get_arima_estimates_matrix` *ARMA Coefficient Summary*

### Description

Generate a summary of ARMA coefficients for a single series

### Usage

```
get_arima_estimates_matrix(seas_obj = NULL, add_diff = FALSE)
```

### Arguments

| | |
|---|---|
| `seas_obj` | seas object generated from a call of `seas` on a single time series |
| `add_diff` | logical scalar; add differencing information, if included in model |

### Value

matrix of ARMA coefficients, standard errors, and t-statistics for a given series

## Examples

```
air_seas <-
   seasonal::seas(AirPassengers, x11='', slidingspans = '', transform.function = 'log',
                        arima.model = '(0 1 1)(0 1 1)', regression.aictest = 'td',
                        forecast.maxlead=36, check.print = c( 'pacf', 'pacfplot' ))
air_arima_matrix <- get_arima_estimates_matrix(air_seas, add_diff = TRUE)
```

---

`get_auto_outlier_string` *Get automatic outlier names*

---

## Description

Get the names of outliers identified in the seas object for a single series.

## Usage

```
get_auto_outlier_string(seas_obj = NULL)
```

## Arguments

| | |
|---|---|
| `seas_obj` | A seas object for a single series generated from the `seasonal` package |

## Value

Character string containing a summary of the outliers identified in the regARIMA model. If no regressors or automatic outliers in the model, the routine will return a blank character.

## Examples

```
m_air <- seasonal::seas(AirPassengers, arima.model = '(0 1 1)(0 1 1)', x11='')
this_auto_outlier <- get_auto_outlier_string(m_air)
```

---

`get_fcst_tval` *t-values of within sample forecasts*

---

## Description

returns t-values of within sample forecasts, up to 3

## Usage

```
get_fcst_tval(seas_obj = NULL, terror_lags)
```

## Arguments

| | |
|---|---|
| `seas_obj` | seas object for a single series |
| `terror_lags` | Integer scalar for number of forecast lags from the end of series we'll collect t-statistics. Must be either 1, 2, or 3. |

## Value

an array of t-values of within sample forecasts, up to length 3

## Examples

```
m_air_short <- seasonal::seas(AirPassengers, series.span = ',1960.9',
                    arima.model='(0 1 1)(0 1 1)', x11 = "",
                    forecast.maxlead = 36, slidingspans = '',
                    transform.function = 'log')
fcst_tstat <- get_fcst_tval(m_air_short, 3)
```

---

`get_model_ftest` *Get model based F-test*

---

## Description

Extract values associated with the model based F-test specified by the `this_ftest` argument

## Usage

```
get_model_ftest(seas_obj = NULL, this_ftest = "seasonal", return_this = "all")
```

## Arguments

| | |
|---|---|
| `seas_obj` | A seas object for a single series generated from the `seasonal` package |
| `this_ftest` | Character string; type of model based f-test to return. Default is "seasonal"; only other acceptable value is "td" |
| `return_this` | Character string, Code that controls what values are returned. Acceptable values are `"all"`, `"dof"` (degrees of freedom), `"ftest"` , (F-test value), or `"pval"` (F-test p-value). Default is "all". |

## Value

Numeric vector with seasonal or trading day F-statistic, degrees of freedom, p-value. If not found, return NULL

## Examples

```
m_td_seas_air <- seasonal::seas(AirPassengers, arima.model = '(0 1 1)',
                               regression.variables = c('seasonal', 'td'), x11='')
this_seas_ftest <- get_model_ftest(m_td_seas_air, this_ftest = 'seasonal',
                                  return_this = 'ftest')
this_td_ftest   <- get_model_ftest(m_td_seas_air, this_ftest = 'td',
                                  return_this = 'ftest')
```

---

`get_month_index` *Generate index of month abbreviation*

---

### Description

Process string of month abbrev to return a numeric index

### Usage

```
get_month_index(this_month_string)
```

### Arguments

| | |
|---|---|
| `this_month_string` | Character string; 3 character abbreviation of month |

### Value

Index of month - 1 for 'Jan', 2 for 'Feb', etc.

### Examples

```
thisOtl <- 'AO2015.Jan'
thisCode <- 'AO'
thisPerChar <- substr(thisOtl,nchar(thisCode)+6,nchar(thisOtl))
thisPerIndex <- get_month_index(thisPerChar)
```

---

`get_mq_key` *Make a UDG key for X-11-ARIMA M and Q statistics*

---

### Description

Generates the UDG key for X-11-ARIMA M and Q statistics based on a label

### Usage

```
get_mq_key(this_label)
```

### Arguments

| | |
|---|---|
| `this_label` | character string; name of an X-11-ARIMA M and Q statistics |

### Value

character string with the corresponding UDG label for `this_label`. If incorrect label is specified, returns NULL

### Examples

```
m7_key  <- get_mq_key('M7')
```

## `get_mq_label` *Make a label for X-11-ARIMA M and Q statistics*

### Description

Generates a label for X-11-ARIMA M and Q statistics

### Usage

```
get_mq_label(this_key = "f3.q")
```

### Arguments

| | |
|---|---|
| `this_key` | character string; name of an X-11-ARIMA M and Q statistics used in the UDG X-13 output |

### Value

character string with the corresponding label for `this_key`. If incorrect label is specified, returns NULL

### Examples

```
m7_label  <- get_mq_label('f3.m07')
```

## `get_nonseasonal_theta` *Nonseasonal Moving Average from Airline Model*

### Description

Get the value of a nonseasonal moving average coefficient estimated from an airline model.

### Usage

```
get_nonseasonal_theta(
  seas_obj = NULL,
  this_index = 1,
  return_string = TRUE,
  significant_digits = 3
)
```

### Arguments

| | |
|---|---|
| `seas_obj` | A seas object for a single series generated from the `seasonal` package |
| `this_index` | An integer scalar, an index of the vector values to be passed. Acceptable values are 1 (nonseasonal MA coefficient value), 2 (nonseasonal MA coefficent standard error), or 3 (t-value of the nonseasonal MA coefficient). Default is 1. |
| `return_string` | A Logical scalar; indicates whether value returned is a string or numeric. Default is TRUE. |
| `significant_digits` | an integer scalar; significant digits to be saved when a string is returned. Default is 3. |

### Value

Character string containing a value related to the seasonal MA coefficient from the regARIMA model fit in the seas object m. If return_string is FALSE, this is a numeric. The standard error or t-value of the seasonal MA coefficient can be returned depending on the value of this_index.

### Examples

```
m_air <- seasonal::seas(AirPassengers, transform.function = 'log',
                        arima.model = '(0 1 1)(0 1 1)', x11='')
this_nonseasonal_theta <- get_nonseasonal_theta(m_air, return_string = FALSE)
```

---

## `get_norm_stat` *Extract normality statistics from X-13*

---

### Description

Extract normality statistics from the `seas` object of a single series

### Usage

```
get_norm_stat(seas_obj = NULL, this_norm)
```

### Arguments

| | |
|---|---|
| `seas_obj` | `seas` object generated by the `seasonal` package for a single series. |
| `this_norm` | character string; type of normality statistic being extracted. Permissable values are 'a', 'kurtosis', 'skewness' |

### Value

Double precision number for normality statistic described in this_key. If incorrect this_key used, function returns a NULL value. If normality statistic not generated in this run, function returns a NULL value.

### Examples

```
air_seas <- seasonal::seas(AirPassengers, slidingspans = '', transform.function = 'log',
                 x11 = '', forecast.maxlead=36, check.print = c( 'pacf', 'pacfplot' ))
air_norm_stat <- get_norm_stat(air_seas, this_norm = 'kurtosis')
```

## get_regarima_estimates_matrix

*Generate summary of regARIMA model coefficients*

### Description

Generate a summary of coefficients from a regARIMA model for a single series

### Usage

```
get_regarima_estimates_matrix(
  seas_obj = NULL,
  add_diff = FALSE,
  this_xreg_names = NULL
)
```

### Arguments

| | |
|---|---|
| `seas_obj` | `seas` object generated from a call of `seas` on a single time series |
| `add_diff` | logical scalar; add differencing information, if included in model |
| `this_xreg_names` | Character array; name of user defined regressors. Default is NULL, no user defined regressors. Number of names in this vector should match number of user-defined regressors; if not, a warning message will be produced. |

### Value

matrix of regARIMA model coefficients, standard errors, and t-statistics for a given series

### Examples

```
air_seas <- seasonal::seas(AirPassengers, x11='', slidingspans = '',
                           transform.function = 'log', arima.model = '(0 1 1)(0 1 1)',
                           regression.aictest = 'td', forecast.maxlead=36,
                           check.print = c( 'pacf', 'pacfplot' ))
air_regarima_matrix <- get_regarima_estimates_matrix(air_seas)
```

## get_regression_estimates_matrix

*Generate regression coefficient summary*

### Description

Generate a summary of regression coefficients for a single series

### Usage

```
get_regression_estimates_matrix(seas_obj = NULL, this_xreg_names = NULL)
```

### Arguments

| | |
|---|---|
| `seas_obj` | `seas` object generated from a call of `seas` on a single time series |
| `this_xreg_names` | Character array; name of user defined regressors. Default is NULL, no user defined regressors. |

### Value

matrix of regression coefficients, standard errors, and t-statistics for a given series

### Examples

```
air_seas <- seasonal::seas(AirPassengers, x11='', slidingspans = '',
                           transform.function = 'log', arima.model = '(0 1 1)(0 1 1)',
                           regression.aictest = 'td', forecast.maxlead=36,
                           check.print = c( 'pacf', 'pacfplot' ))
air_reg_matrix <- get_regression_estimates_matrix(air_seas)
```

## `get_reg_string` *Get names of regressors*

### Description

Generate string of names for the regressors used in the model fit for a given series

### Usage

```
get_reg_string(seas_obj = NULL, xreg_names = NULL)
```

### Arguments

| | |
|---|---|
| `seas_obj` | `seas` object generated by the `seasonal` package for a single series. |
| `xreg_names` | Character vector with names of user defined regressors used in mopdel. Default is NULL, no user defined regressors. Number of names in this vector should match number of user-defined regressors; if not, a warning message will be produced. |

### Value

Character string containing a summary of the regressors in the regARIMA model. If no regressors in the model, the routine will return a blank character.

### Examples

```
air_seas <- seasonal::seas(AirPassengers, slidingspans = '',
                  transform.function = 'log',
                  x11 = '', forecast.maxlead=36,
                  check.print = c( 'pacf', 'pacfplot' ))
air_reg <- get_reg_string(air_seas)
```

## get_seasonal_ftest_all

*Generate model based F-test*

### Description

Generate model based F-test, changing the model to remove seasonal differences and adding seasonal regressors if necessary. This function is used in the overall seasonal test from Maravall (2012)

### Usage

```
get_seasonal_ftest_all(this_seas = NULL, this_series = "b1")
```

### Arguments

| | |
|---|---|
| `this_seas` | seas object for a single series |
| `this_series` | character string; the table used to generate the model based F-test. Default is "b1". |

### Value

a numeric vector with the degrees of freedom, F statistic, and probability generated for the model based seasonal f-test used in the seasonal testing procedure in Maravall(2012)

### Examples

```
m_air <-
     seasonal::seas(AirPassengers, arima.model='(0 1 1)(0 1 1)',
                    forecast.maxlead = 36, slidingspans = '',
                    transform.function = 'log')
air_ftest_all <- get_seasonal_ftest_all(m_air, this_series = "a1")
```

## get_seasonal_ftest_prob

*Probability of model based F-test*

### Description

Get probability for model based F-test, changing the model to remove seasonal differences and adding seasonal regressors if necessary. This function is used in the overall seasonal test from Maravall (2012)

### Usage

```
get_seasonal_ftest_prob(this_seas = NULL, this_series = "b1")
```

### Arguments

| | |
|---|---|
| `this_seas` | seas object for a single series |
| `this_series` | character string; the table used to generate the model based F-test. Default is "b1". |

## Value

test probability generated for the model based seasonal f-test used in the seasonal testing procedure in Maravall(2012)

## Examples

```
m_air <-
     seasonal::seas(AirPassengers, arima.model='(0 1 1)(0 1 1)',
                    forecast.maxlead = 36, slidingspans = '',
                    series.save = 'b1',
                    transform.function = 'log')
air_ftest_prob <- get_seasonal_ftest_prob(m_air)
```

---

`get_seasonal_theta` *Seasonal Moving Average from Airline Model*

---

## Description

Get the value of a seasonal moving average coefficient estimated from an airline model.

## Usage

```
get_seasonal_theta(
  seas_obj = NULL,
  freq = 12,
  this_index = 1,
  return_string = TRUE,
  significant_digits = 3
)
```

## Arguments

| | |
|---|---|
| `seas_obj` | A seas object for a single series generated from the `seasonal` package |
| `freq` | A numeric scalar, the frequency of the time series. Default is 12. |
| `this_index` | An integer scalar, an index of the vector values to be passed. Acceptable values are 1 (seasonal MA coefficient value), 2 (seasonal MA coefficent standard error), or 3 (t-value of the Seasonal MA coefficient). Default is 1. |
| `return_string` | A Logical scalar; indicates whether value returned is a string or numeric. Default is TRUE. |
| `significant_digits` | an integer scalar; significant digits to be saved when a string is returned. Default is 3. |

## Value

Character string containing a value related to the seasonal MA coefficient from the regARIMA model fit in the seas object m. The standard error or t-value of the seasonal MA coefficient can be returned depending on the value of this_index. If return_string is FALSE, this is a numeric.

### Examples

```
m_air <- seasonal::seas(AirPassengers, transform.function = 'log',
                        arima.model = '(0 1 1)(0 1 1)', x11='')
this_seasonal_theta <- get_seasonal_theta(m_air, return_string = FALSE)
```

---

## `get_transform` *Get transformation*

---

### Description

Get transformation from the seas object of a single time series

### Usage

```
get_transform(m_seas)
```

### Arguments

| | |
|---|---|
| `m_seas` | seas object generated from a call of `seas` on a single time series |

### Value

Character string with transformation used to model time series in seas run

### Examples

```
m_air <- seasonal::seas(AirPassengers, arima.model = '(0 1 1)(0 1 1)', x11='')
air_trans <- get_transform(m_air)
```

---

## `get_udg_entry` *returns a specific element of a list of udg entries*

---

### Description

returns a specific element of a list of udg entries

### Usage

```
get_udg_entry(this_seas, this_key, this_index = 0, convert = TRUE)
```

### Arguments

| | |
|---|---|
| `this_seas` | seas object for a single series |
| `this_key` | character scalar; keyword found in UDG output generated by X-13ARIMA-SEATS |
| `this_index` | integer scalar; index of entry in vector to extract. If set to 0 (the default), get the last entry. |
| `convert` | logical scalar; if TRUE, convert |

## Value

The `this_index` element of the array returned from the UDG entry for `this_key`

## Examples

```
m_air_short <- seasonal::seas(AirPassengers, series.span = ',1960.9',
                    arima.model='(0 1 1)(0 1 1)',
                    forecast.maxlead = 36, slidingspans = '',
                    transform.function = 'log')
fcst_tstat <- get_udg_entry(m_air_short, 'forctval01')
```

---

`get_udg_index` *Index for entry in UDG list*

---

## Description

Return index for entry in UDG list

## Usage

```
get_udg_index(udg_list, this_key)
```

## Arguments

| | |
|---|---|
| `udg_list` | List object generated by `udg()` function of the `seasonal` package. |
| `this_key` | Keyword found in udg files generated by X-13ARIMA-SEATS |

## Value

An integer denoting which element in the udg output matches the key provided by the user. If there is no match, the function returns the number 0.

## Examples

```
ukgas_seas <- seasonal::seas(UKgas, series.period = 4, arima.model = '(0 1 1)(0 1 1)',
                             x11='', transform.function = 'log', forecast.maxlead=20,
                             check.print = c( 'pacf', 'pacfplot' ))
ukgas_udg <- seasonal::udg(ukgas_seas)
ukgas_udg_index_a <- get_udg_index(ukgas_udg, this_key='a')
```

## `get_window` *Subspan time series*

### Description

Generate subspan of time series

### Usage

```
get_window(X, plot_start = NULL, plot_end = NULL)
```

### Arguments

| | |
|---|---|
| `X` | Time Series object |
| `plot_start` | Integer vector of length 2; Starting date for plot. Default is starting date for the time series. |
| `plot_end` | Integer vector of length 2; Starting date for plot. Default is ending date for the time series. |

### Value

generate subspan of time series `X` specified by `plot_start` and `plot_end`

### Examples

```
air50 <- get_window(AirPassengers, plot_start = c(1950,1), plot_end = c(1959,12))
```

## `input_saved_x13_file` *Import File Saved by X-13ARIMA-SEATS*

### Description

Import data from a file saved by the X-13ARIMA-SEATS program

### Usage

```
input_saved_x13_file(filename = NULL, pos = 2, ncol = 2)
```

### Arguments

| | |
|---|---|
| `filename` | Character string, filename of a file saved by the X-13ARIMA-SEATS program. This is a required entry. |
| `pos` | Integer scalar, column of data to be extracted from `filename`. Default is 2. |
| `ncol` | Integer scalar, number of columns of data that exist within `filename`. Default is 2. |

### Value

a time series array

### Examples

```
## Not run:
airline.sa <- input_saved_x13_file("airline.d11")

## End(Not run)
```

---

## make_diag_df *Generate diagnostic summary data frame*

---

### Description

Generate diagnostic summary data frame

### Usage

```
make_diag_df(
  this_data_names,
  this_acf_test = NULL,
  this_d11f_test = NULL,
  this_spec_peak_test = NULL,
  this_spec_peak_ori_test = NULL,
  this_qs_test = NULL,
  this_qs_rsd_test = NULL,
  this_qs_seasonal_test = NULL,
  this_model_test = NULL,
  this_sspan_test = NULL,
  this_m7_test = NULL,
  this_q2_test = NULL,
  return_this = "both"
)
```

### Arguments

| | |
|---|---|
| `this_data_names` | vector object with names of time series used in seasonal adjustment |
| `this_acf_test` | list object with results from test of regARIMA residual ACF |
| `this_d11f_test` | list object with results from test of D11F |
| `this_spec_peak_test` | list object with results from testing for spectral peaks in the seasonally adjusted series |
| `this_spec_peak_ori_test` | list object with results from testing for spectral peaks in the original series |
| `this_qs_test` | list object with results from QS test |
| `this_qs_rsd_test` | list object with results from residual QS test |
| `this_qs_seasonal_test` | list object with results from seasonal QS test |
| `this_model_test` | list object with results from model diagnostics test |

| | |
|---|---|
| `this_sspan_test` | list object with results from sliding spans test |
| `this_m7_test` | list object with results from M7 test |
| `this_q2_test` | list object with results from Q2 test |
| `return_this` | Character string; what the function returns - 'why' returns why the test failed or received a warning, 'test' returns test results, or 'both' |

### Value

A data frame with X-13 Diagnostics, with the elements not expressed as factors

### Examples

```
test_lauto <- seasonal::seas(xt_data_new,
                     x11 = '', slidingspans = '',
                     arima.model = "(0 1 1)(0 1 1)",
                     transform.function = 'log',
                     forecast.maxlead=60,
                     check.print = c( 'pacf', 'pacfplot' ))
test_lauto_update <-
     Filter(function(x) inherits(x, "seas"), test_lauto)
test_acf <- lapply(test_lauto_update, function(x)
  try(acf_test(x, return_this = 'both')))
test_d11f <- lapply(test_lauto_update, function(x)
  try(d11f_test(x, p_level = 0.05, return_this = 'both')))
test_spec_peak <- lapply(test_lauto_update, function(x)
  try(spec_peak_test(x, return_this = 'both')))
test_spec_peak_ori <- lapply(test_lauto_update, function(x)
  try(spec_peak_test(x, this_spec = "spcori", return_this = 'both')))
test_qs <- lapply(test_lauto_update, function(x)
  try(qs_test(x, test_full = FALSE, p_limit_fail = 0.01,
              p_limit_warn = 0.05, return_this = 'both')))
test_qs_rsd <- lapply(test_lauto_update, function(x)
  try(qs_rsd_test(x, test_full = FALSE, p_limit_fail = 0.01,
                  p_limit_warn = 0.05, return_this = 'both')))
test_qs_seasonal <- lapply(test_lauto_update, function(x)
  try(qs_seasonal_test(x, test_full = FALSE,
                       p_limit_pass = 0.01, p_limit_warn = 0.05,
                       robust_sa = FALSE, return_this = 'both')))
test_model <- lapply(test_lauto_update, function(x)
  try(model_test(x, return_this = 'both')))
test_sspan <- lapply(test_lauto_update, function(x)
  try(sspan_test(x, sf_limit = 15, change_limit = 35,
      return_this = 'both')))
test_m7 <- lapply(test_lauto_update, function(x)
  try(mq_test(x, return_this = 'both')))
test_q2 <- lapply(test_lauto_update, function(x)
  try(mq_test(x, this_label = 'Q2', return_this = 'both')))
test_names <- names(xt_data_new)
test_diag_df <-
    make_diag_df(test_names,
                 this_acf_test = test_acf,
                 this_d11f_test = test_d11f,
                 this_spec_peak_test = test_spec_peak,
                 this_spec_peak_ori_test = test_spec_peak_ori,
                 this_qs_test = test_qs,
                 this_qs_rsd_test = test_qs_rsd,
                 this_qs_seasonal_test = test_qs_seasonal,
                 this_model_test = test_model,
                 this_sspan_test = test_sspan,
                 this_m7_test = test_m7,
                 this_q2_test = test_q2)
```

## match_list *List element match*

### Description

Returns element of list that matches `this_string`

### Usage

```
match_list(this_list, this_string = "fail")
```

### Arguments

| | |
|---|---|
| `this_list` | List of character strings. |
| `this_string` | Character string to match against elements of the list, ie, `this_string` = `'pass'`. Default is 'fail' |

### Value

A vector of list element names that match this_string. If nothing matches, the function will output the string 'none'

### Examples

```
test_lauto <- seasonal::seas(xt_data_list, x11 = '', slidingspans = '',
                      arima.model = "(0 1 1)(0 1 1)",
                      transform.function = 'log',
                      forecast.maxlead=60,
                      check.print = c( 'pacf', 'pacfplot' ))
test_lauto_update <-
     Filter(function(x) inherits(x, "seas"), test_lauto)
test_acf_test <- lapply(test_lauto_update, function(x)
                        try(acf_test(x, return_this = 'test')))
test_acf_fail <- match_list(test_acf_test, 'fail')
test_acf_warn <- match_list(test_acf_test, 'warn')
test_acf_pass <- match_list(test_acf_test, 'pass')
```

## `match_list_number` *Number of list element matches*

### Description

Returns number of elements in list that matches `this_string`

### Usage

```
match_list_number(this_list, this_string = "fail")
```

### Arguments

| | |
|---|---|
| `this_list` | List of character strings. |
| `this_string` | Character string to match against elements of the list, ie, `this_string` = `'pass'` |

### Value

The number of list items that match this_string.

### Examples

```
test_lauto <- seasonal::seas(xt_data_list, x11 = '', slidingspans = '',
                      transform.function = 'log',
                      arima.model = "(0 1 1)(0 1 1)",
                      forecast.maxlead=60,
                      check.print = c( 'pacf', 'pacfplot' ))
test_lauto_update <-
     Filter(function(x) inherits(x, "seas"), test_lauto)
test_acf_test <- lapply(test_lauto_update, function(x)
                        try(acf_test(x, return_this = 'test')))
test_acf_number_fail <- match_list_number(test_acf_test, 'fail')
test_acf_number_warn <- match_list_number(test_acf_test, 'warn')
test_acf_number_pass <- match_list_number(test_acf_test, 'pass')
```

## `member_of_list` *Member of list*

### Description

Determines if a name is a member of a list

### Usage

```
member_of_list(this_list = NULL, this_name = NULL)
```

### Arguments

| | |
|---|---|
| `this_list` | A list of objects |
| `this_name` | character string; element name of `this_list` |

## Value

returns TRUE if `this_name` is an element of `this_list`, FALSE otherwise

## Examples

```
xt_lauto <- seasonal::seas(xt_data_list, slidingspans = '', x11 = "",
                      transform.function = 'log',
                      arima.model = "(0 1 1)(0 1 1)",
                      forecast.maxlead=36,
                      check.print = c( 'pacf', 'pacfplot' ))
if (member_of_list(xt_lauto, 'us24')) {
    ## Not run: save_spec_file(xt_lauto$us24, 'us24')
}
```

---

`model_test` *Tests Time Series Model.*

---

## Description

Tests whether the time series model has acceptable diagnostics.

## Usage

```
model_test(
  seas_obj = NULL,
  t_value = 3,
  p_value = 0.05,
  otl_auto_limit = 5,
  otl_all_limit = 5,
  return_this = "test"
)
```

## Arguments

| | |
|---|---|
| `seas_obj` | • object generated by `seas()` of the `seasonal` package. |
| `t_value` | • t-statistic limit for regressors |
| `p_value` | • p-value limit for regressors |
| `otl_auto_limit` | • limit for number of automatically identified outliers |
| `otl_all_limit` | • limit for number of automatically identified outliers |
| `return_this` | character string; what the function returns - 'test' returns test results, 'why' returns why the test failed or received a warning, or 'both' |

## Value

A text string denoting if the series passed or failed the tests of ARIMA diagnostics.

## Examples

```
ukgas_seas <-
   seasonal::seas(UKgas, series.period = 4, arima.model = '(0 1 1)(0 1 1)',
                  x11='', transform.function = 'log', forecast.maxlead=20,
                  check.print = c( 'pacf', 'pacfplot' ))
ukgas_model <- model_test(ukgas_seas, t_value=3.0, p_value=0.01, otl_auto_limit=4,
                          otl_all_limit=6)
```

---

`model_test_why` *Model Test Warning Message*

---

## Description

Generates text on why a time series model is inadequate

## Usage

```
model_test_why(
  udg_list = NULL,
  t_value = 3,
  p_value = 0.05,
  otl_auto_limit = 5,
  otl_all_limit = 5,
  return_both = FALSE
)
```

## Arguments

| | |
|---|---|
| `udg_list` | • list object generated by `udg()` function of the `seasonal` package. |
| `t_value` | • numeric scalar; t-statistic limit for regressors |
| `p_value` | • numeric scalar; p-value limit for regressors |
| `otl_auto_limit` | • integer scalar; limit for number of automatically identified outliers |
| `otl_all_limit` | • integer scalar; limit for number of automatically identified outliers |
| `return_both` | Logical scalar indicating whether the calling function will return both the test results and why the test failed or produced a warning. Default is FALSE. |

## Value

A text string denoting why the series passed or failed a series of tests of ARIMA diagnostics. ukgas_seas <- seasonal::seas(UKgas, series.period = 4, arima.model = '(0 1 1)(0 1 1)', x11='', transform.function = 'log', forecast.maxlead=20, check.print = c( 'pacf', 'pacfplot' )) ukgas_udg <- seasonal::udg(ukgas_seas) ukgas_model_why <- model_test_why(ukgas_udg, t_value=3.0, p_value=0.01, otl_auto_limit=4, otl_all_limit=6, return_both = TRUE) `mq_test` *Test X-11-ARIMA M and Q statistics*

## Description

Generates a test for X-11-ARIMA M and Q statistics

## Usage

```
mq_test(
  seas_obj = NULL,
  this_label = "m7",
  this_fail_limit = 1.2,
  this_warn_limit = 0.8,
  return_this = "test"
)
```

## Arguments

| | |
|---|---|
| `seas_obj` | • object generated by `seas()` of the seasonal package. |
| `this_label` | • character string; label for an M or Q statistic, such as ’M7’, ’Q’, or ’Q2’. |
| `this_fail_limit` | • numeric scalar; value above which the M or Q statistic fails; default is 1.2 |
| `this_warn_limit` | • numeric scalar; value above which the M or Q statistic gives a warning message if it is less than `this_fail_Limit`; default is 0.8 |
| `return_this` | character string; what the function returns - ’test’ returns test results, ’why’ returns why the test failed or received a warning, or ’both’ |

## Value

A text string denoting if series passes or has a warning for residual seasonality. If d11f statistic not found, return ’none’.

## Examples

```
ukgas_seas <- seasonal::seas(UKgas, series.period = 4, arima.model = '(0 1 1)(0 1 1)',
                             x11='', transform.function = 'log', forecast.maxlead=20,
                             check.print = c( 'pacf', 'pacfplot' ))
ukgas_q <- mq_test(ukgas_seas, this_label = 'q', return_this = 'both')
```

## norm_test *Normality Tests for Time Series Models.*

### Description

Tests different normality statistics available in X-13ARIMA-SEATS.

### Usage

```
norm_test(seas_obj = NULL, this_norm = NULL, return_this = "test")
```

### Arguments

| | |
|---|---|
| seas_obj | • object generated by `seas()` of the `seasonal` package. |
| this_norm | type of normality statistic being extracted; permissable values are 'a', 'kurtosis', 'skewness' |
| return_this | character string; what the function returns - 'test' returns test results, 'why' returns why the test failed or received a warning, or 'both' |

### Value

A text string denoting whether the series passed or failed the specific normality test. If improper value is specified for this_norm, return NULL. If no statistic is found, return NA.

### Examples

```
ukgas_seas <-
   seasonal::seas(UKgas, series.period = 4, arima.model = '(0 1 1)(0 1 1)',
                  x11 = '', transform.function = 'log', forecast.maxlead = 20,
                  check.print = c( 'pacf', 'pacfplot' ))
ukgas_norm_a <- norm_test(ukgas_seas, 'a')
ukgas_norm_kurtosis <- norm_test(ukgas_seas, 'kurtosis')
ukgas_norm_skewness <- norm_test(ukgas_seas, 'skewness')
```

## norm_test_why *Normality Test Warning Message*

### Description

generates message for why different normality statistics available in X-13ARIMA-SEATS fail.

### Usage

```
norm_test_why(udg_list = NULL, this_norm = NULL, return_both = FALSE)
```

### Arguments

| | |
|---|---|
| udg_list | • list object generated by `udg()` function of the `seasonal` package. |
| this_norm | type of normality statistic being extracted; permissable values are 'a', 'kurtosis', 'skewness' |
| return_both | Logical scalar indicating whether the calling function will return both the test results and why the test failed or produced a warning. Default is FALSE. |

### Value

A text string showing why a series failed the specific normality test

### Examples

```
ukgas_seas <- seasonal::seas(UKgas, series.period = 4, arima.model = '(0 1 1)(0 1 1)',
                             x11='', transform.function = 'log', forecast.maxlead=20,
                             check.print = c( 'pacf', 'pacfplot' ))
ukgas_udg <- seasonal::udg(ukgas_seas)
ukgas_norm_a_why <- norm_test_why(ukgas_udg, 'a', return_both = TRUE)
ukgas_norm_kurtosis_why <- norm_test_why(ukgas_udg, 'kurtosis', return_both = TRUE)
ukgas_norm_skewness_why <- norm_test_why(ukgas_udg, 'skewness', return_both = TRUE)
```

## `NP_test` *Non-Parametric test from Maravall (2012)*

### Description

Non-Parametric test for seasonality based on Kendall and Ord (1990), and originally due to Friedman from a paper by Maravall. This code is adapted from `kendalls` subroutine in `ansub11.f` from the `X-13ARIMA-SEATS` source code

### Usage

```
NP_test(x = NULL)
```

### Arguments

| | |
|---|---|
| `x` | `ts` time series object |

### Value

List object with three elements: ken (test statistic), df (degrees of freedom), cv (test probability)

### Examples

```
NP_test_air <- NP_test(AirPassengers)
```

## `optimal_seasonal_filter` *Optimal X-11 seasonal moving average selection*

### Description

Determine the optimal X-11 seasonal moving average based on the value of the seasonal moving average coefficient from an airline model.

## Usage

```
optimal_seasonal_filter(
  this_series,
  aictest = NULL,
  model = "(0 1 1)(0 1 1)",
  variables = NULL,
  outlier = TRUE,
  trans = NULL,
  missing_code = NULL,
  this_xreg = NULL,
  dp_limits = TRUE,
  use_msr = FALSE,
  use_3x15 = TRUE
)
```

## Arguments

| | |
|---|---|
| `this_series` | A time series object |
| `aictest` | a character string with the entries for the `regression.aictest` argument to the `seas` function from the `seasonal` package. Default is NULL, AIC testing not done. |
| `model` | a character string with the entry for the `arima.model` argument to the `seas` function from the `seasonal` package. Default is `'(0 1 1)(0 1 1)'`. Model should have a `(0 1 1)` seasonal term) |
| `variables` | a character string with the entries for the `regression.variables` argument to the `seas` function from the `seasonal` package. Default is NULL, no regressors added. |
| `outlier` | logical scalar, if TRUE outlier identification is done in the call to the `seas` function from the `seasonal` package. Default is TRUE. |
| `trans` | characater scalar, a character string with the entry for the `transform.function` argument to the `seas` function, Default is NULL, and the entry `auto` will be used. |
| `missing_code` | numeric scalar, a number with the entry for the `series.missing`code argument to the `seas` function, Default is NULL, no missing value code is used. |
| `this_xreg` | numeric matrix, a user defined regressor matrix to be used in the model estimation. Default is NULL, no user-defined regressors are used. |
| `dp_limits` | logical scalar, if TRUE limits from Deputot and Planas will be used to choose the moving average, else limits from Bell Chow and Chu will be used. Default is TRUE. |
| `use_msr` | logical scalar, if TRUE result of MSR selection will be used if model cannot be estimated, otherwise function will return a NULL value. Default is FALSE. |
| `use_3x15` | logical scalar, if TRUE 3x15 seasonal filter will be returned if chosen, otherwise function will return a `3x9` value. Default is FALSE. |

## Value

The optimal X-11 seasonal filter, unless the airline model cannot be estimated.

## Examples

```
this_seasonal  <-
   optimal_seasonal_filter(shoes2008, aictest = c('td', 'easter'), use_msr = TRUE)
this_seasonal2 <-
   optimal_seasonal_filter(shoes2008, aictest = c('td', 'easter'), dp_limits = FALSE,
                           use_msr = TRUE)
```

---

`overall_seasonal_test_1`

*First overall sasonality test from Maravall (2012)*

---

## Description

Conduct the first overall test for seasonality as laid out in Maravall (2012)

## Usage

```
overall_seasonal_test_1(this_seas, this_series = "a1", take_log = TRUE)
```

## Arguments

| | |
|---|---|
| `this_seas` | `seas` object for a single series |
| `this_series` | character string; the table used to generate the AR(30) spectrum. Default is "a1". |
| `take_log` | logical scalar; indicates if the AR spectrum is generated from the log of the data. Default is TRUE. |

## Value

A list with 3 elements: QStest (test probability for QS), NPtest (test probability for NP), and result (character string with test result - possible values of either "evidence of seasonality" and "no evidence of seasonality")

## Examples

```
air_seas <- seasonal::seas(AirPassengers,
                   arima.model='(0 1 1)(0 1 1)',
                   forecast.maxlead = 36, slidingspans = '',
                   transform.function = 'log')
first_test <- overall_seasonal_test_1(air_seas)
```

## overall_seasonal_test_2 *Second overall sasonality test from Maravall (2012)*

### Description

Conduct the second overall test for seasonality as laid out in Maravall (2012)

### Usage

```
overall_seasonal_test_2(
  this_seas,
  this_ar_spec_cv = NULL,
  this_series = "b1",
  take_log = TRUE,
  take_diff = TRUE
)
```

### Arguments

| | |
|---|---|
| `this_seas` | seas object for a single series |
| `this_ar_spec_cv` | List object with two elements - 99 and 95 percent critical values for the frequencies of the AR(30) spectrum as generated by the `gen_ar_spec_cv` function. |
| `this_series` | character string; the table used to generate the AR(30) spectrum. Default is "b1". |
| `take_log` | logical scalar; indicates if the AR spectrum is generated from the log of the data. Default is TRUE. |
| `take_diff` | logical scalar; indicates if the data is differenced before the AR spectrum is generated. Default is TRUE. |

### Value

A list with 5 elements: QStest (test probability for QS), NPtest (test probability for NP), Ftest (test probability for model based seasonal F-test), spectrum (character string with test result - possible values of either "evidence of seasonal peak", "no evidence of seasonal peak"), and result (character string with test result - possible values of either "strong seasonal", "weak seasonal", "no seasonal"

### Examples

```
air_seas <- seasonal::seas(AirPassengers,
                  arima.model='(0 1 1)(0 1 1)',
                  series.save = "b1",
                  forecast.maxlead = 36, slidingspans = '',
                  transform.function = 'log')
ar30_spec_cv <- gen_ar_spec_cv(1000, 97, 12)
second_test <- overall_seasonal_test_2(air_seas, ar30_spec_cv)
```

## process_list — *Process list object of numbers*

### Description

Process list object of numbers and return names of elements that are either greater than or less than a limit

### Usage

```
process_list(
  this_list = NULL,
  this_limit = NULL,
  abs_value = FALSE,
  greater_than = TRUE
)
```

### Arguments

| | |
|---|---|
| `this_list` | List of numeric values. The elements should be scalars, not arrays. |
| `this_limit` | Numeric scalar which serves as the limit of the numbers stored in this_list |
| `abs_value` | Logical scalar that indicates whether the absolute value is taken of the numbers before the comparison is made. (default is FALSE) |
| `greater_than` | logical object that specified whether the element names returned are greater than or less than the limit specified in this_limit (default is TRUE) |

### Value

A vector of list element names where the value in `this_list` is greater than or less than the limit specified in `this_limit`. If nothing matches, the function will output the string `'none'`

### Examples

```
xt_lauto <-
  seasonal::seas(xt_data_list, slidingspans = '', forecast.maxlead=36,
                       arima.model = "(0 1 1)(0 1 1)",
                       transform.function = 'log', x11 = "",
                       check.print = c( 'pacf', 'pacfplot' ))
xt_lauto_update <-
     Filter(function(x) inherits(x, "seas"), xt_lauto)
m7_key     <- get_mq_key('M7')
xt_m7_list <- lapply(xt_lauto_update, function(x) try(get_udg_entry(x, m7_key)))
xt_m7_pass <- process_list(xt_m7_list, this_limit = 1.0, abs_value = TRUE, greater_than = FALSE)
```

## proc_outlier *Extract dates from outlier text*

### Description

Process name of outlier regressor to extract the dates associated with the outlier

### Usage

```
proc_outlier(
  this_outlier = NULL,
  this_code = NULL,
  this_freq = 12,
  add_type = TRUE
)
```

### Arguments

| | |
|---|---|
| this_outlier | Character string; outlier regressor |
| this_code | Character string; code for outlier - possible values are 'ao', 'ls', 'tc', 'so', 'rp', 'tls' |
| this_freq | integer scalar; time series frequency. Default is 12. |
| add_type | logical scalar; determines if type of outlier is added to the output. Default is TRUE. |

### Value

list of either year and month/quarter of outlier, or year and month/quarter of start and end of outlier

### Examples

```
air_seas <-
   seasonal::seas(AirPassengers, x11='', slidingspans = '',
                  transform.function = 'log', arima.model = '(0 1 1)(0 1 1)',
                  regression.aictest = 'td', forecast.maxlead=36,
                  check.print = c( 'pacf', 'pacfplot' ))
this_auto_outlier <- get_auto_outlier_string(air_seas)
this_code         <- tolower(substr(this_auto_outlier, 1, 2))
this_outlier      <- proc_outlier(this_auto_outlier, this_code)
```

## qs_fail_why *QS diagnostic failure message*

### Description

generates text explaining why the QS diagnostic failed or generated a warning.

### Usage

```
qs_fail_why(
  udg_list = NULL,
  test_full = TRUE,
  test_span = TRUE,
  p_limit_fail = 0.01,
  robust_sa = TRUE,
  return_both = FALSE
)
```

### Arguments

| | |
|---|---|
| `udg_list` | • list object generated by `udg()` function of the `seasonal` package. |
| `test_full` | Logical scalar indicating whether to test the full series span |
| `test_span` | Logical scalar indicating whether to test the final 8-year span used by the spectrum diagnostic |
| `p_limit_fail` | Numeric scalar; P-value limit for QS statistic for failure |
| `robust_sa` | Logical scalar indicating if SA or irregular series adjusted for extremes is included in testing. Default is TRUE. |
| `return_both` | Logical scalar indicating whether the calling function will return both the test results and why the test failed or produced a warning. Default is FALSE. |

### Value

A text string denoting why the series failed the tests of QS diagnostics. ukgas_seas <- seasonal::seas(UKgas, series.period = 4, arima.model = '(0 1 1)(0 1 1)', x11='', transform.function = 'log', forecast.maxlead=20, check.print = c( 'pacf', 'pacfplot' )) ukgas_udg <- seasonal::udg(ukgas_seas) ukgas_qs_test <- qs_fail_why(ukgas_udg, test_full = FALSE, p_limit_fail = 0.01, return_both = TRUE)

---

## `qs_rsd_fail_why` *QS diagnostic for regarima residuals failure message*

---

### Description

generates text explaining why the QS diagnostic failed or generated a warning for regARIMA residuals.

### Usage

```
qs_rsd_fail_why(
  udg_list = NULL,
  test_full = TRUE,
  test_span = TRUE,
  p_limit_fail = 0.01,
  return_both = FALSE
)
```

### Arguments

| | |
|---|---|
| `udg_list` | • list object generated by `udg()` function of the `seasonal` package. |
| `test_full` | Logical scalar indicating whether to test the full series span |
| `test_span` | Logical scalar indicating whether to test the final 8-year span used by the spectrum diagnostic |
| `p_limit_fail` | Numeric scalar; P-value limit for QS statistic for warning |
| `return_both` | Logical scalar indicating whether the calling function will return both the test results and why the test failed or produced a warning. Default is FALSE. |

### Value

A text string denoting why the series failed the QS test of regARIMA residuals.

### Examples

```
ukgas_seas <- seasonal::seas(UKgas, series.period = 4, arima.model = '(0 1 1)(0 1 1)',
                            x11='', transform.function = 'log', forecast.maxlead=20,
                            check.print = c( 'pacf', 'pacfplot' ))
ukgas_udg <- seasonal::udg(ukgas_seas)
ukgas_qs_rsd_fail_why <-
  qs_rsd_fail_why(ukgas_seas, test_full = FALSE, p_limit_fail = 0.005, return_both = TRUE)
```

---

## `qs_rsd_test` *QS diagnostic test*

---

### Description

Tests using the QS diagnostic developed by Maravall

### Usage

```
qs_rsd_test(
  seas_obj = NULL,
  test_full = TRUE,
  test_span = TRUE,
  p_limit_fail = 0.01,
  p_limit_warn = 0.05,
  return_this = "test"
)
```

### Arguments

| | |
|---|---|
| `seas_obj` | `seas` object generated by the `seasonal` package. |
| `test_full` | Logical scalar indicating whether to test the full series span |
| `test_span` | Logical scalar indicating whether to test the final 8-year span used by the spectrum diagnostic |
| `p_limit_fail` | Numeric scalar; P-value limit for QS statistic for failure |
| `p_limit_warn` | Numeric scalar; P-value limit for QS statistic for warning |
| `return_this` | character string; what the function returns - 'test' returns test results, 'why' returns why the test failed or received a warning, or 'both' |

### Value

A text string denoting if the regarima residuals passed or failed tests for residual seasonality using the QS diagnostics. Can test the entire series or the last 8 years or both.

### Examples

```
ukgas_seas <- seasonal::seas(UKgas, series.period = 4, arima.model = '(0 1 1)(0 1 1)',
                             x11='', transform.function = 'log', forecast.maxlead=20,
                             check.print = c( 'pacf', 'pacfplot' ))
ukgas_qs_test_rsd <- qs_rsd_test(ukgas_seas, test_full = FALSE, p_limit_fail = 0.01,
                                 p_limit_warn = 0.05, return_this = 'both')
```

---

## `qs_rsd_warn_why` *Residual QS diagnostic warning message.*

---

### Description

generates text explaining why the QS diagnostic failed or generated a warning for regARIMA residuals.

### Usage

```
qs_rsd_warn_why(
  udg_list = NULL,
  test_full = TRUE,
  test_span = TRUE,
  p_limit_warn = 0.05,
  return_both = FALSE
)
```

### Arguments

| | |
|---|---|
| `udg_list` | • list object generated by `udg()` function of the `seasonal` package. |
| `test_full` | Logical scalar indicating whether to test the full series span |
| `test_span` | Logical scalar indicating whether to test the final 8-year span used by the spectrum diagnostic |
| `p_limit_warn` | Numeric scalar; P-value limit for QS statistic for warning |
| `return_both` | Logical scalar indicating whether the calling function will return both the test results and why the test failed or produced a warning. Default is FALSE. |

### Value

A text string denoting why the series generated a warning message for the QS of regARIMA residuals.

## Examples

```
ukgas_seas <- seasonal::seas(UKgas, series.period = 4, arima.model = '(0 1 1)(0 1 1)',
                             x11='', transform.function = 'log', forecast.maxlead=20,
                             check.print = c( 'pacf', 'pacfplot' ))
ukgas_udg <- seasonal::udg(ukgas_seas)
ukgas_qs_rsd_warn <-
   qs_rsd_warn_why(ukgas_udg, test_full = FALSE, p_limit_warn = 0.05, return_both = TRUE)
```

---

`qs_seasonal_fail_why` *QS Test for original series*

---

## Description

Tests using the QS diagnostic developed by Maravall to determine if the original series is seasonal

## Usage

```
qs_seasonal_fail_why(
  udg_list = NULL,
  test_full = TRUE,
  test_span = TRUE,
  p_limit_warn = 0.05,
  robust_sa = TRUE,
  return_both = FALSE
)
```

## Arguments

| | |
|---|---|
| `udg_list` | • list object generated by `udg()` function of the `seasonal` package. |
| `test_full` | Logical scalar indicating whether to test the full series span |
| `test_span` | Logical scalar indicating whether to test the final 8-year span used by the spectrum diagnostic |
| `p_limit_warn` | Numeric scalar; P-value limit for QS statistic for warning |
| `robust_sa` | Logical scalar indicating if original series adjusted for extremes is included in testing |
| `return_both` | Logical scalar indicating whether the calling function will return both the test results and why the test failed or produced a warning. Default is FALSE. |

## Value

A text string denoting if the series passed or failed the tests of ARIMA diagnostics.

## Examples

```
ukgas_seas <- seasonal::seas(UKgas, series.period = 4, arima.model = '(0 1 1)(0 1 1)',
                             x11='', transform.function = 'log', forecast.maxlead=20,
                             check.print = c( 'pacf', 'pacfplot' ))
ukgas_udg <- seasonal::udg(ukgas_seas)
ukgas_qs_fail_seasonal <-
  qs_seasonal_fail_why(ukgas_udg, test_full = FALSE,
                       p_limit_warn = 0.05, robust_sa=FALSE, return_both = FALSE)
```

## `qs_seasonal_test` *QS seasonal tests*

### Description

Tests using the QS diagnostic developed by Maravall to determine if the original series is seasonal

### Usage

```
qs_seasonal_test(
  seas_obj = NULL,
  test_full = TRUE,
  test_span = TRUE,
  p_limit_pass = 0.01,
  p_limit_warn = 0.05,
  robust_sa = TRUE,
  return_this = "test"
)
```

### Arguments

| | |
|---|---|
| `seas_obj` | `seas` object generated by the `seasonal` package. |
| `test_full` | Logical scalar indicating whether to test the full series span |
| `test_span` | Logical scalar indicating whether to test the final 8-year span used by the spectrum diagnostic |
| `p_limit_pass` | Numeric scalar; P-value limit for QS statistic for passing |
| `p_limit_warn` | Numeric scalar; P-value limit for QS statistic for warning |
| `robust_sa` | Logical scalar indicating if original series adjusted for extremes is included in testing |
| `return_this` | Character string; what the function returns - 'test' returns test results, 'why' returns why the test failed or received a warning, or 'both' |

### Value

A text string denoting if the series passed or failed tests for seasonality using the QS diagnostics. Can test the entire series or the last 8 years or both.

### Examples

```
ukgas_seas <-
   seasonal::seas(UKgas, series.period = 4, arima.model = '(0 1 1)(0 1 1)',
                  x11='', transform.function = 'log', forecast.maxlead=20,
                  check.print = c( 'pacf', 'pacfplot' ))
ukgas_qs_test_seasonal <-
   qs_seasonal_test(ukgas_seas, test_full = FALSE, p_limit_pass = 0.01,
                    p_limit_warn = 0.05, robust_sa=FALSE, return_this = 'both')
```

## `qs_seasonal_warn_why` *Warning or error messages for QS seasonal diagnostic*

### Description

Tests using the QS diagnostic developed by Maravall to determine if the original series is seasonal

### Usage

```
qs_seasonal_warn_why(
  udg_list = NULL,
  test_full = TRUE,
  test_span = TRUE,
  p_limit_pass = 0.05,
  robust_sa = TRUE,
  return_both = FALSE
)
```

### Arguments

| | |
|---|---|
| `udg_list` | • list object generated by `udg()` function of the `seasonal` package. |
| `test_full` | Logical scalar indicating whether to test the full series span |
| `test_span` | Logical scalar indicating whether to test the final 8-year span used by the spectrum diagnostic |
| `p_limit_pass` | Numeric scalar; P-value limit for QS statistic for passing |
| `robust_sa` | Logical scalar indicating if original series adjusted for extremes is included in testing |
| `return_both` | Logical scalar indicating whether the calling function will return both the test results and why the test failed or produced a warning. Default is FALSE. |

### Value

A text string denoting if the series had a worning message from the tests for seasonality using the QS diagnostics. Can test the entire series or the last 8 years or both.

### Examples

```
ukgas_seas <-
   seasonal::seas(UKgas, series.period = 4, arima.model = '(0 1 1)(0 1 1)',
                  x11='', transform.function = 'log', forecast.maxlead=20,
                  check.print = c( 'pacf', 'pacfplot' ))
ukgas_udg <- seasonal::udg(ukgas_seas)
ukgas_qs_warn_seasonal <-
   qs_seasonal_warn_why(ukgas_udg, test_full = FALSE, p_limit_pass = 0.025,
                        robust_sa=FALSE, return_both = TRUE)
```

## qs_series *QS diagnostic test on a number of series*

### Description

Apply QS Tests to a list of seas objevts

### Usage

```
qs_series(
  seas_obj_list = NULL,
  this_stat = "qsori",
  less_than = TRUE,
  p_limit = 0.01
)
```

### Arguments

| | |
|---|---|
| seas_obj_list | list object of seas object generated by the seasonal package. |
| this_stat | Character string that specifies which QS statistic is being tested. Allowable values are 'qsori', 'qsorievadj', 'qsrsd', 'qssadj', 'qssadjevadj', 'qsirr', 'qsirrevadj', 'qssori', 'qssorievadj', 'qssrsd', 'qsssadj', 'qsssadjevadj', 'qssirr', 'qssirrevadj' |
| less_than | Logical scalar which indicates if the test is going to be QS < p_limit (less_than = TRUE) or QS > p_limit (less_than = FALSE). |
| p_limit | Numeric scalar; P-value limit for QS statistic |

### Value

A vector of list element names that have the given QS statistic either less than or greater than the given P-value limit. If nothing matches, the function will output the string 'none'

### Examples

```
test_lauto <- seasonal::seas(xt_data_new,
                     x11 = '', slidingspans = '',
                     arima.model = "(0 1 1)(0 1 1)",
                     transform.function = 'log',
                     forecast.maxlead=60,
                     check.print = c( 'pacf', 'pacfplot' ))
test_lauto_update <-
     Filter(function(x) inherits(x, "seas"), test_lauto)
test_qs_names <- qs_series(test_lauto_update, 'qssori', less_than = FALSE, p_limit=0.025)
```

`qs_test` *QS Test for residual seasonality*

## Description

Tests using the QS diagnostic developed by Maravall on seasonally adjusted series and the irregular component n

## Usage

```
qs_test(
  seas_obj = NULL,
  test_full = TRUE,
  test_span = TRUE,
  p_limit_fail = 0.01,
  p_limit_warn = 0.05,
  robust_sa = TRUE,
  return_this = "test"
)
```

## Arguments

| | |
|---|---|
| `seas_obj` | `seas` object generated by the `seasonal` package. |
| `test_full` | Logical scalar indicating whether to test the full series span |
| `test_span` | Logical scalar indicating whether to test the final 8-year span used by the spectrum diagnostic |
| `p_limit_fail` | Numeric scalar; P-value limit for QS statistic for failure |
| `p_limit_warn` | Numeric scalar; P-value limit for QS statistic for warning |
| `robust_sa` | Logical scalar indicating if SA or irregular series adjusted for extremes is included in testing. Default is TRUE. |
| `return_this` | character string; what the function returns - 'test' returns test results, 'why' returns why the test failed or received a warning, or 'both' |

## Value

A text string denoting if the series passed or failed tests 1for residual seasonality using the QS diagnostics. Can test the entire series or the last 8 years or both.

## Examples

```
ukgas_seas <-
   seasonal::seas(UKgas, series.period = 4, arima.model = '(0 1 1)(0 1 1)',
                  x11='', transform.function = 'log', forecast.maxlead=20,
                  check.print = c( 'pacf', 'pacfplot' ))
ukgas_qs_test <- qs_test(ukgas_seas, test_full = FALSE, p_limit_fail = 0.01,
                         p_limit_warn = 0.05, return_this = 'both')
```

## qs_warn_why *warning message for QS Test for residual seasonality*

### Description

generates text explaining why the QS diagnostic generated a warning.

### Usage

```
qs_warn_why(
  udg_list = NULL,
  test_full = TRUE,
  test_span = TRUE,
  p_limit_warn = 0.05,
  robust_sa = TRUE,
  return_both = FALSE
)
```

### Arguments

| | |
|---|---|
| `udg_list` | • list object generated by `udg()` function of the `seasonal` package. |
| `test_full` | Logical scalar indicating whether to test the full series span |
| `test_span` | Logical scalar indicating whether to test the final 8-year span used by the spectrum diagnostic |
| `p_limit_warn` | Numeric scalar; P-value limit for QS statistic for warning |
| `robust_sa` | Logical scalar indicating if SA or irregular series adjusted for extremes is included in testing. Default is TRUE. |
| `return_both` | Logical scalar indicating whether the calling function will return both the test results and why the test failed or produced a warning. Default is FALSE. |

### Value

A text string denoting if the series passed or failed the tests of ARIMA diagnostics.

### Examples

```
ukgas_seas <-
   seasonal::seas(UKgas, series.period = 4, arima.model = '(0 1 1)(0 1 1)',
                  x11='', transform.function = 'log', forecast.maxlead=20,
                  check.print = c( 'pacf', 'pacfplot' ))
ukgas_udg <- seasonal::udg(ukgas_seas)
ukgas_qs_test <- qs_warn_why(ukgas_udg, test_full = FALSE, p_limit_warn = 0.025, return_both = TRUE)
```

## replace_na *Replace NA*

### Description

Replace NA with a string

### Usage

```
replace_na(this_vec, replace_string = "NA")
```

### Arguments

| | |
|---|---|
| `this_vec` | Vector object. |
| `replace_string` | Character scalar which replaces the NAs in the vector. Default is 'NA'. |

### Value

A vector with all NAs replaced by a character string

### Examples

```
sample_vec <- c(rnorm(25), NA, rnorm(24))
sample_vec_missing  <- replace_na(sample_vec, replace_string = 'Missing')
```

## r_terror *TERROR for R*

### Description

A function that duplicates the functionality of the TERROR software (Caporello and Maravall 2004) that performs quality control on time series based on one step ahead forecasts

### Usage

```
r_terror(
  this_series = NULL,
  max_lead = 36,
  log_transform = TRUE,
  aictest = NULL,
  terror_lags = 1
)
```

### Arguments

| | |
|---|---|
| `this_series` | Time series array |
| `max_lead` | Number of forecasts generated by the seas run. Default is 36. |
| `log_transform` | logical scalar, if TRUE `transform.function` will be set to `log` in the call to the `seas` function, otherwise `auto` will be used. Default is TRUE. |
| `aictest` | a character string with the entries for the `regression.aictest` argument to the `seas` function from the `seasonal` package. Default is NULL. |
| `terror_lags` | Integer scalar for number of forecast lags from the end of series we'll collect t-statistics. Must be either 1, 2, or 3. |

### Value

t-statistics generated by out of sample forecast error for the last 1 to 3 observation of each series in the list.

### Examples

```
air_terror <- r_terror(AirPassengers, log_transform = TRUE,
                       aictest = c('td', 'easter'), terror_lags = 3)
```

---

## `r_terror_list` *TERROR for R (applied to a list of series)*

---

### Description

A function that duplicates the functionality of the TERROR software (Caporello and Maravall 2004) that performs quality control on time series based on one step ahead forecasts

### Usage

```
r_terror_list(
  this_data_list = NULL,
  this_lead = 36,
  this_log = TRUE,
  this_aictest = NULL,
  this_terror_lags = 1
)
```

### Arguments

| | |
|---|---|
| `this_data_list` | List of time series (all series in list should be the same frequency and have the same ending date.) |
| `this_lead` | Number of forecasts generated by the seas run. Default is 36. |
| `this_log` | logical scalar, if TRUE `transform.function` will be set to `log` in the call to the `seas` function, otherwise `auto` will be used. Default is TRUE. |
| `this_aictest` | a character string with the entries for the `regression.aictest` argument to the `seas` function from the `seasonal` package. Default is NULL. |
| `this_terror_lags` | Integer scalar for number of forecast lags from the end of series we'll collect t-statistics. Must be either 1, 2, or 3. |

### Value

list of t-statistics generated by out of sample forecast error for the last 1 to 3 observation of each series in the list.

### Examples

```
xt_terror <- r_terror_list(xt_data_list, this_log = FALSE,
                           this_aictest = c('td', 'easter'), this_terror_lags = 3)
```

---

## save_metafile *Generate X-13ARIMA-SEATS metafile*

---

### Description

Generates external metafile for spec files generated from a list of `seas` objects

### Usage

```
save_metafile(
  this_seas_list = NULL,
  this_name_vec = NULL,
  metafile_name = NULL,
  this_directory = NULL,
  include_directory = FALSE
)
```

### Arguments

| | |
|---|---|
| `this_seas_list` | • list of seas objects the metafile will be generated from |
| `this_name_vec` | vector of character string; vector of series names from the list of seas objects that will be saved. Default is all elements of the seasonal object list `this_seas_list` are saved. |
| `metafile_name` | • character string; base name of metafile to be generated. If not specified, use name of list input as metafile name. Note - do not specify the `".mta"` file extension. |
| `this_directory` | • optional directory where the meta file is stored. If not specified, the metafile will be saved in the current working directory. |
| `include_directory` | • logical scalar; if TRUE, include directory specified in `this_directory` with file name output. Otherwise, output only names in `this_name_vec`. Default is FALSE. Note that the argument `this_directory` must also be specified. |

### Value

Generates metafile that can be used directly with the X-13ARIMA-SEATS program.

### Examples

```
xt_lauto <- seasonal::seas(xt_data_list, slidingspans = '', x11 = '',
                    arima.model = "(0 1 1)(0 1 1)",
                    transform.function = 'log',
                    forecast.maxlead=36,
                    check.print = c( 'pacf', 'pacfplot' ))
## Not run: save_metafile(xt_lauto, metafile_name = 'xt')
```

## `save_seas_object` *Save seas objects*

### Description

stores seas command to reproduce the seas object `this_seas_object` into the file `file_name.r`

### Usage

```
save_seas_object(
  this_seas_object = NULL,
  file_name = NULL,
  series_name = NULL,
  data_list = NULL,
  list_element = NULL,
  user_reg = NULL,
  this_window = FALSE,
  this_directory = NULL,
  this_sep = "_",
  print_out = FALSE
)
```

### Arguments

| | |
|---|---|
| `this_seas_object` | seasonal object |
| `file_name` | character string; file name where seas object is stored; default is the name of the seasonal object |
| `series_name` | character string; name of time series object used by the seas object; default is the name of the seasonal object |
| `data_list` | character string; name of the list object that holds data; there is no default |
| `list_element` | character string; name of the list element used as data; default is the name of the seasonal object |
| `user_reg` | character string; name of a time series matrix containing user defined regressors; there is no default. If not set, will set variables related to user defined regressors to NULL in the static version of the seas object. |
| `this_window` | logical indicator variable; determines if a span of the original series will be used in the analysis using the `window()` function. If FALSE, the entire series will be used in the saved file. |
| `this_directory` | character string; optional directory where the spec file is stored |
| `this_sep` | character string; separator between elements of the file name. Default is `"_"`. |
| `print_out` | logical indicator variable; determines if an `out()` function is printed at the end of the script. If FALSE, the `out()` function is commented out. |

### Value

stores the seas command to reproduce the `seas` object `this_seas_object` into the file `file_name.r` - if `file_name` is not specified, the name of the seasonal object will be used to form the output file name.

### Examples

```
ukgas_seas <-
   seasonal::seas(UKgas, series.period = 4, arima.model = '(0 1 1)(0 1 1)',
                  x11 = '', transform.function = 'log', forecast.maxlead = 20,
                  check.print = c( 'pacf', 'pacfplot' ))
## Not run: save_seas_object(ukgas_seas, file_name = "ukgas_seas", series_name = "ukgas",
                 print_out = TRUE)
## End(Not run)
```

---

## save_series *Save Series*

---

### Description

Save a user-defined regression array or matrix with time series attributes to an external ASCII file in X-13ARIMA-SEATS' datevalue format

### Usage

```
save_series(this_series, this_file)
```

### Arguments

| | |
|---|---|
| `this_series` | double precision time series array to be saved. |
| `this_file` | character string; name of file time series array to be saved to. |

### Value

file with user-defined regressors will be produced

### Examples

```
ukgas_seas <- seasonal::seas(UKgas, series.period = 4, arima.model = '(0 1 1)(0 1 1)',
                             x11='', transform.function = 'log', forecast.maxlead=20,
                             slidingspans = '', check.print = c( 'pacf', 'pacfplot' ))
ukgas_sa <- seasonal::final(ukgas_seas)
## Not run: save_series(ukgas_sa, 'ukgas_sa.txt')
```

## save_spec_file — *Save spec file representation of* seas *object*

### Description

stores the spec file representation of the seas object this_seas_object into the file file_name.spc

### Usage

```
save_spec_file(
  this_seas_object = NULL,
  file_name = NULL,
  this_directory = NULL,
  data_file_name = NULL,
  xreg_file_name = NULL,
  this_user_name = NULL,
  this_title = NULL
)
```

### Arguments

| | |
|---|---|
| this_seas_object | seasonal object |
| file_name | character string; file name where seas object is stored; default is the name of the seasonal object |
| this_directory | character string; optional directory where the spec file is stored |
| data_file_name | character string; optional external file name where data file is stored. Path should be included with file name if data file is not in working directory; quotes will be added by the routine. Default is no change in file entry in the spec file. |
| xreg_file_name | character string; optional external file name where user defined regressors are stored. Path should be included with file name if data file is not in working directory; quotes will be added by the routine. Default is no change in file entry in the spec file. |
| this_user_name | vector of character strings; optional names for the user-defined regressors. Should only appear if xreg_file_name is specified. |
| this_title | character string; optional custom title; quotes will be added by the routine. Default is no change in title entry in the spec file. |

### Value

stores the spec file representation of the seas object this_seas_object into the file file_name.spc

### Examples

```
xt_lauto <- seasonal::seas(xt_data_list, slidingspans = '', transform.function = 'log',
                       arima.model = "(0 1 1)(0 1 1)",
                       forecast.maxlead=36, check.print = c( 'pacf', 'pacfplot' ))
## Not run: save_spec_file(xt_lauto$us24, 'us24',
               data_file_name = "xtus24mu.dat",
              this_title = "Production run for Building Permits for US 2-4 Unit Houses")
## End(Not run)
```

## save_spec_file_vec — *stores the spec file representation of the* seas *object* this_seas_object *into the file* file_name.spc

### Description

stores the spec file representation of the seas object this_seas_object into the file file_name.spc

### Usage

```
save_spec_file_vec(
  this_seas_object_list = NULL,
  this_name_vec = NULL,
  this_directory = NULL,
  this_data_directory = NULL,
  this_ext = ".dat",
  this_title_list = NULL,
  this_title_base = NULL,
  this_xreg_list = NULL,
  this_user_list = NULL,
  make_metafile = FALSE,
  this_metafile_name = NULL,
  include_directory = FALSE
)
```

### Arguments

| | |
|---|---|
| this_seas_object_list | list of seasonal objects |
| this_name_vec | vector of character string; vector of series names from the list of seas objects that will be saved. Default is all elements of the seasonal object list this_seas_object_list are saved. |
| this_directory | character string; optional directory where the spec file is stored |
| this_data_directory | character string; optional directory where the data files are stored. Data files are assumed to have the same names as in this_name_vec with the file extension specified in this_ext. Default is no change in file entry in the spec file. |
| this_ext | character string; file extension for data files. Default is ".dat". |
| this_title_list | list of character strings with the titles for each series. Default is to set title to the series name. |
| this_title_base | character string; optional base for custom title; series name will be added at the end of the title; quotes will be added by the routine. Default is to set title to the series name. |
| this_xreg_list | list of character strings with the filenames of user defined regressors or NULL for each series. Default is to not set regression.file for the individual series. |
| this_user_list | list of vectors of character strings with the names of user defined regressors or NULL for each series. Default is to not set regression.file for the individual series. |

| | |
|---|---|
| `make_metafile` | logical scalar; if TRUE, generate a makefile for this set of files; do not otherwise. Default is FALSE. |
| `this_metafile_name` | • character string; base name of metafile to be generated. If not specified, use name of list input as metafile name. Note - do not specify the `".mta"` file extension. |
| `include_directory` | • logical scalar; if TRUE, include directory specified in `this_directory` with file name output. Otherwise, output only names in `this_name_vec`. Default is FALSE. |

### Value

stores the spec file representation of the `seas` object `this_seas_object` into the file `file_name.spc`

### Examples

```
xt_lauto <-
   seasonal::seas(xt_data_new, slidingspans = '',
                  arima.model = "(0 1 1)(0 1 1)",
                  transform.function = 'log',
                  forecast.maxlead=36,
                  check.print = c( 'pacf', 'pacfplot' ))
test_lauto_update <-
     Filter(function(x) inherits(x, "seas"), xt_lauto)
## Not run: save_spec_file_vec(test_lauto_update, c('mw1u', 'ne1u', 'so1u', 'we1u'),
                   this_data_directory = 'X:\\seasonalAdj\\testing',
                   this_title_base = 'Production Run for Building Permits : ',
                   make_metafile = TRUE, include_directory = TRUE)
## End(Not run)
```

---

## `seasonal_ftest` *Model-based F-Test for Time Series Models.*

---

### Description

Model based test for seasonality based on stable seasonal regressors

### Usage

```
seasonal_ftest(
  seas_obj = NULL,
  p_limit_fail = 0.01,
  p_limit_warn = 0.05,
  return_this = "test"
)
```

### Arguments

| | |
|---|---|
| `seas_obj` | object generated by `seas()` of the seasonal package. |
| `p_limit_fail` | Numeric scalar; P-value limit for model based seasonal F-statistic for passing |
| `p_limit_warn` | Numeric scalar; P-value limit for model based seasonal F-statistic for a warning |
| `return_this` | character string; what the function returns - 'test' returns test results, 'why' returns why the test failed or received a warning, or 'both' |

### Value

A text string denoting if the series passed or failed tests for seasonality using the model based F-test diagnostic.

### Examples

```
m_air <-
  seasonal::seas(AirPassengers, transform.function = 'log', arima.model = '(0 1 1)',
                 regression.variables = c('seasonal', 'td'), x11='')
this_seasonal_ftest <- seasonal_ftest(m_air, return_this = 'both')
```

---

## `set_critical_value` *Set outlier critical value*

---

### Description

Set outlier critical value using the Ljung algorithm as given in Ljung, G. M. (1993). On outlier detection in time series. Journal of Royal Statistical Society B 55, 559-567.

### Usage

```
set_critical_value(number_observations, cv_alpha = 0.01)
```

### Arguments

| | |
|---|---|
| `number_observations` | number of observations tested for outliers |
| `cv_alpha` | alpha for critical value |

### Value

outlier critical value generated by the algorithm given in Ljung (1993). The critical value in X-13 is different as it is adjusted to allow for smaller values to approximate the normal distribution.

### Examples

```
this_critical_value <- set_critical_value(12, 0.025)
```

---

## `set_legend_position` *generate position of plot legend*

---

### Description

Generate position code for the `legend` command based on the series being plotted.

## Usage

```
set_legend_position(
  data_matrix = NULL,
  this_plot_start = NULL,
  this_plot_freq = 12,
  time_disp = 3,
  value_disp = 1/6,
  default_code = "top"
)
```

## Arguments

| | |
|---|---|
| `data_matrix` | numeric matrix; matrix where all series being plotted are stored as columns. |
| `this_plot_start` | Integer scalar; start date of the plot. |
| `this_plot_freq` | Integer scalar; Frequency of time series plotted. Default is 12. |
| `time_disp` | Integer scalar; number of observations on the x-axis taken up by the legend. Default is 3. |
| `value_disp` | Numeric scalar; factor representing the percentage of the y axis taken up by the legend. Default is 1/6. |
| `default_code` | Character string; default position code if the corners are not available. Default is `"top"`. Possible values are `"bottomright"`, `"bottom"`, `"bottomleft"`, `"left"`, `"topleft"`, `"topright"`, `"top"`, `"right"` and `"center"`. |

## Value

Position codes for the legend command. Possible values are `"bottomright"`, `"bottom"`, `"bottomleft"`, `"topleft"`, `"topright"` and the value of `default_code`.

## Examples

```
shoes_seas <-
    seasonal::seas(shoes2007, slidingspans = "", transform.function = "log", x11 = "",
                   forecast.maxlead=36, check.print = c( "pacf", "pacfplot" ))
this_series <- shoes2007
this_sa     <- seasonal::final(shoes_seas)
this_legend_position <-
    set_legend_position(cbind(this_series, this_sa), start(this_series),
                        this_plot_freq = 4, time_disp = 8, value_disp = 1/8,
                        default_code = "top")
```

---

`shoes2007` *Retail sales of shoes, 2007*

---

## Description

A time series object

## Usage

```
shoes2007
```

### Format

Retail sales of shoes ending in December of 2007

---

## shoes2008 *Retail sales of shoes, 2008*

---

### Description

A time series object

### Usage

```
shoes2008
```

### Format

Retail sales of shoes ending in April of 2008

---

## spec_peak_fail_why *Failure text for spectral peaks*

---

### Description

generate text on why spectral peaks are flagged

### Usage

```
spec_peak_fail_why(
  udg_list = NULL,
  peak_level = 6,
  this_spec = "spcsa",
  return_both = FALSE
)
```

### Arguments

| | |
|---|---|
| `udg_list` | • list object generated by `udg()` function of the `seasonal` package. |
| `peak_level` | Integer scalar - limit to determine if a frequency has a spectral peak |
| `this_spec` | text string with the spectrum being tested allowable entries are 'spcori','spcsa','spcirr','spcrsd' |
| `return_both` | Logical scalar indicating whether the calling function will return both the test results and why the test failed or produced a warning. Default is FALSE. |

### Value

A text string denoting if the series passed the tests of spectrum diagnostics, or why the series did not pass. Note that for spcori, the series fails if none of the frequencies tested had peaks

## Examples

```
m_air <- seasonal::seas(AirPassengers, transform.function = 'log',
                        arima.model = '(0 1 1)(0 1 1)', x11 = '')
m_air_udg <- seasonal::udg(m_air)
this_spec_peak_fail_why <-
    spec_peak_fail_why(m_air_udg, this_spec = 'spcori', return_both = TRUE)
```

---

`spec_peak_test` *Test for spectral peaks*

---

## Description

Test if spectral peaks are flagged

## Usage

```
spec_peak_test(
  seas_obj = NULL,
  peak_level = 6,
  peak_warn = 3,
  this_spec = "spcsa",
  return_this = "test"
)
```

## Arguments

| | |
|---|---|
| `seas_obj` | object generated by `seas()` of the seasonal package. |
| `peak_level` | Integer scalar - limit to determine if a frequency has a spectral peak |
| `peak_warn` | Integer scalar - limit to produce a warning that a frequency may have a spectral peak |
| `this_spec` | text string with the spectrum being tested allowable entries are 'spcori','spcsa','spcirr','spcrsd' |
| `return_this` | character string; what the function returns - 'test' returns test results, 'why' returns why the test failed or received a warning, or 'both' |

## Value

A text string denoting if the series passed or failed the tests of spectrum diagnostics. Note that for spcori, the series fails if none of the frequencies tested had peaks

## Examples

```
m_air <- seasonal::seas(AirPassengers, transform.function = 'log',
                        arima.model = '(0 1 1)(0 1 1)', x11 = '')
this_spec_peak_test <- spec_peak_test(m_air, this_spec = 'spcori', return_this = 'both')
```

## spec_peak_warn_why *Warning message for spectral peaks*

### Description

generate warning message related to spectral peaks

### Usage

```
spec_peak_warn_why(
  udg_list = NULL,
  peak_warn_level = 3,
  this_spec = "spcsa",
  return_both = FALSE
)
```

### Arguments

| | |
|---|---|
| udg_list | • list object generated by udg() function of the seasonal package. |
| peak_warn_level | Integer scalar - limit to produce a warning that a frequency may have a spectral peak |
| this_spec | text string with the spectrum being tested allowable entries are 'spcori','spcsa','spcirr','spcrsd' |
| return_both | Logical scalar indicating whether the calling function will return both the test results and why the test failed or produced a warning. Default is FALSE. |

### Value

A text string denoting if the series passed the tests of spectrum diagnostics, or why the series did not pass. Note that for spcori, the series fails if none of the frequencies tested had peaks

### Examples

```
m_air <-
   seasonal::seas(AirPassengers, transform.function = 'log',
                  arima.model = '(0 1 1)(0 1 1)', x11 = '')
m_air_udg <- seasonal::udg(m_air)
this_spec_peak_warn_why <-
    spec_peak_warn_why(m_air_udg, this_spec = 'spcori', return_both = TRUE)
```

## sspan_test *Sliding Spans Diagnostic*

### Description

Tests using the sliding spans diagnostic

### Usage

```
sspan_test(
  seas_obj = NULL,
  sf_limit = 25,
  change_limit = 40,
  additivesa = FALSE,
  return_this = "test"
)
```

### Arguments

| | |
|---|---|
| `seas_obj` | object generated by `seas()` of the seasonal package. |
| `sf_limit` | Numeric object; limit for the percentage of seasonal spans flagged |
| `change_limit` | Numeric object; limit for the percentage of month-to-month changes flagged |
| `additivesa` | logical scalar; if true, the adjustment is assumed to be additive; default is FALSE |
| `return_this` | character string; what the function returns - 'test' returns test results, 'why' returns why the test failed or received a warning, or 'both' |

### Value

A text string denoting if the series passed or failed the tests of sliding spans diagnostics.

### Examples

```
ukgas_seas <- seasonal::seas(UKgas, series.period = 4, arima.model = '(0 1 1)(0 1 1)',
                             x11='', transform.function = 'log', forecast.maxlead=20,
                             slidingspans = '', check.print = c( 'pacf', 'pacfplot' ))
ukgas_sspan_test <-
   sspan_test(ukgas_seas, sf_limit = 15, change_limit = 35, return_this = 'both')
```

## `sspan_test_why` *Sliding Spans Diagnostic Warning Messages*

### Description

Generate text on why Tests using the sliding spans diagnostic fail

### Usage

```
sspan_test_why(
  udg_list = NULL,
  sf_limit = 25,
  change_limit = 40,
  additivesa = FALSE,
  return_both = FALSE
)
```

## Arguments

| | |
|---|---|
| `udg_list` | • list object generated by `udg()` function of the `seasonal` package. |
| `sf_limit` | Numeric object; limit for the percentage of seasonal spans flagged |
| `change_limit` | Numeric object; limit for the percentage of month-to-month changes flagged |
| `additivesa` | logical scalar; if true, the adjustment is assumed to be additive; default is FALSE |
| `return_both` | Logical scalar indicating whether the calling function will return both the test results and why the test failed or produced a warning. Default is FALSE. |

## Value

A text string denoting if the series passed the tests of sliding spans diagnostics diagnostics, or why the series failed.

## Examples

```
ukgas_seas <- seasonal::seas(UKgas, series.period = 4, arima.model = '(0 1 1)(0 1 1)',
                             x11='', transform.function = 'log', forecast.maxlead=20,
                             check.print = c( 'pacf', 'pacfplot' ))
ukgas_seas_udg <- seasonal::udg(ukgas_seas)
ukgas_sspan_test_why <-
   sspan_test_why(ukgas_seas_udg, sf_limit = 15, change_limit = 35, return_both = TRUE)
```

---

`static_with_outlier` *add outliers to* `seas` *object*

---

## Description

add arguments from the `outlier` spec to a `seas` object

## Usage

```
static_with_outlier(
  this_seas_object = NULL,
  new_data = NULL,
  outlier_span = ",",
  outlier_types = "ao,ls"
)
```

## Arguments

| | |
|---|---|
| `this_seas_object` | seasonal object |
| `new_data` | time series object; updated data set from the data used to generate `this_seas_object` |
| `outlier_span` | character string; sets the argument `outlier.span` |
| `outlier_types` | character string; sets the argument `outlier.types` |

## Value

an updated static seas object with outlier arguments included.

## Examples

```
shoes_seas <-
   seasonal::seas(shoes2007, slidingspans = '', transform.function = 'log', x11 = '',
                  forecast.maxlead=36)
shoes_seas_outlier <- static_with_outlier(shoes_seas, shoes2008, outlier_types = 'all')
```

---

## `static_with_outlier_list` *add outliers to list of* `seas` *object*

---

### Description

add outlier arguments to each element of a list of `seas` objects

### Usage

```
static_with_outlier_list(
  seas_obj_list = NULL,
  new_data_list = NULL,
  outlier_span = ",",
  outlier_types = "ao,ls"
)
```

### Arguments

| | |
|---|---|
| `seas_obj_list` | list of seasonal objects |
| `new_data_list` | list of time series objects; updated data sets from the data used to generate `seas_obj_list` |
| `outlier_span` | character string; sets the argument `outlier.span` |
| `outlier_types` | character string; sets the argument `outlier.types` |

### Value

a list of updated static seas object with outlier arguments included.

### Examples

```
xt_lauto_old <-
   seasonal::seas(xt_data_old, slidingspans = '',
                  transform.function = 'log',
                  x11 = '', forecast.maxlead=36)
xt_outlier_seas <-
   static_with_outlier_list(xt_lauto_old, xt_data_new)
```

## udg_series *Process a list of* seas *elements*

### Description

Process a list of seas elements to find the elements that are greater than or less than a particular limit for a diagnostic

### Usage

```
udg_series(
  seas_obj_list = NULL,
  this_key = "autoout",
  this_limit = 5,
  this_abs = FALSE,
  greater_than = TRUE
)
```

### Arguments

| | |
|---|---|
| `seas_obj_list` | • list of seas objects generated by the seasonal package. |
| `this_key` | • character string containing keyword of the udg function that returns a numeric value |
| `this_limit` | • numeric object which serves as the limit of the diagnostic referred to in `this_key` |
| `this_abs` | Logical scalar that indicates whether the absolute value is taken of the numbers before the comparison is made. (default is FALSE) |
| `greater_than` | • logical object that specified whether the element names returned are greater than or less than the limit specified in `this_limit` |

### Value

A vector of list element names where this_key is greater than or less than the limit specified in this_limit. If nothing matches, the function will output the string 'none'

### Examples

```
xt_lauto <- seasonal::seas(xt_data_list, slidingspans = '', transform.function = 'log', x11 = '',
                        arima.model = "(0 1 1)(0 1 1)",
                        forecast.maxlead=36, check.print = c( 'pacf', 'pacfplot' ))
xt_bad_m7 <- udg_series(xt_lauto, this_key = 'f3.m07', this_limit = 1.2)
xt_bad_q2 <- udg_series(xt_lauto, this_key = 'f3.qm2', this_limit = 1.2)
```

## update_diag_matrix *Update Diagnostic Matrix*

### Description

Update the matrix of diagnostics used to generate the diagnostic data frame in `make_diag_df`

### Usage

```
update_diag_matrix(this_diag_list, this_test_list, this_label)
```

### Arguments

| | |
|---|---|
| `this_diag_list` | list object with elements for seasonal adjustment or modeling diagnostic, titles, and the number of columns |
| `this_test_list` | list object of a specific seasonal adjustment or modeling diagnostic |
| `this_label` | character string; name of diagnostic in `this_test_list` |

### Value

list object with updated elements for seasonal adjustment or modeling diagnostic, titles, and the number of columns

### Examples

```
test_lauto <- seasonal::seas(xt_data_new,
           x11 = '', slidingspans = '',
           arima.model = "(0 1 1)(0 1 1)",
           transform.function = 'log',
           forecast.maxlead=60,
           check.print = c( 'pacf', 'pacfplot' ))
test_lauto_update <-
     Filter(function(x) inherits(x, "seas"), test_lauto)
test_acf <- lapply(test_lauto_update, function(x) try(acf_test(x, return_this = 'both')))
test_names <- names(xt_data_new)
num_names <- length(test_names)
all_diag_list <- list(n = 0, diag = 0, titles = 0)
if (!is.null(test_acf)) {
    if (length(test_acf) < num_names) {
        this_acf_test <- fix_diag_list(test_acf, test_names, return_this = 'both')
    }
    all_diag_list <-
        update_diag_matrix(all_diag_list, test_acf, "ACF")
}
```

## `update_vector` *Update vector.*

### Description

Fill unspecified elements of a vector with the first element of the input series

### Usage

```
update_vector(this_series, this_num)
```

### Arguments

| | |
|---|---|
| `this_series` | Original time series |
| `this_num` | Lenght of updated series. Must be more than the length of `this_series`. |

### Value

an updated vector of length `x_num` augmented with the first value of the input series.

### Examples

```
this_vector <- c(1,2)
updated_vector <- update_vector(this_vector, 4)
```

## `which_error` *Check list for try errors*

### Description

Checks list for try errors, returning element names with errors

### Usage

```
which_error(this_list = NULL)
```

### Arguments

| | |
|---|---|
| `this_list` | list object which potentially contains ’try-error’ class objects. |

### Value

vector of the names of list elements that are ’try-error’ class objects. If the list contains no ’try-error’ class objects, the function will return NULL

### Examples

```
xt_lauto <-
  seasonal::seas(xt_data_list, slidingspans = '', transform.function = 'log',
                             forecast.maxlead=36, arima.model = '(0 1 1)(0 1 1)',
                             check.print = c( 'pacf', 'pacfplot' ))
xt_lauto_errors <- which_error(xt_lauto)
```

## `xt_data_list` *US Building Permits*

### Description

A list object with 12 components of US Building Permits expressed as time series objects

### Usage

```
xt_data_list
```

### Format

A list object with 12 time series elements:

**mw1u** Midwest one family building permits
**mwto** Midwest total building permits
**ne1u** Northeast one family building permits
**neto** Northeast total building permits
**so1u** South one family building permits
**soto** South total building permits
**we1u** West one family building permits
**weto** West total building permits
**us1u** US one family building permits
**us24** US 2-4 family building permits
**us5p** US 5+ family building permits
**usto** US total family building permits

## `xt_data_new` *US Building Permits, One Family Buildings (new)*

### Description

A list object of US One family Building Permits for four regions expressed as time series objects that end in October, 2006

### Usage

```
xt_data_new
```

### Format

A list object with 4 time series elements:

**mw1u** Midwest one family building permits
**ne1u** Northeast one family building permits
**so1u** South one family building permits
**we1u** West one family building permits

`xt_data_old` *US Building Permits, One Family Buildings (old)*

## Description

A list object of US One family Building Permits for four regions expressed as time series objects that end in December, 2005

## Usage

```
xt_data_old
```

## Format

A list object with 4 time series elements:

**mw1u** Midwest one family building permits

**ne1u** Northeast one family building permits

**so1u** South one family building permits

**we1u** West one family building permits

# Index