Package ‘autovarCore’

June 4, 2018

Type Package

Title Automated Vector Autoregression Models and Networks

Version 1.0-4

Date 2018-06-04

BugReports https://github.com/roqua/autovarcore/issues

Maintainer Ando Emerencia <ando.emerencia@gmail.com>

Description Automatically find the best vector autoregression models and networks for a given time series data set. 'AutovarCore' evaluates eight kinds of models: models with and without log transforming the data, lag 1 and lag 2 models, and models with and without weekday dummy variables. For each of these 8 model configurations, 'AutovarCore' evaluates all possible combinations for including outlier dummies (at 2.5x the standard deviation of the residuals) and retains the best model. Model evaluation includes the Eigenvalue stability test and a configurable set of residual tests. These eight models are further reduced to four models because 'AutovarCore' determines whether adding weekday dummies improves the model fit.

License MIT + file LICENSE

Suggests testthat, roxygen2

Imports Rcpp (>= 0.11.4), Amelia, jsonlite, parallel, stats, urca, vars

LinkingTo Rcpp

RoxygenNote 6.0.1

NeedsCompilation yes

Author Ando Emerencia [aut, cre]

Repository CRAN

Date/Publication 2018-06-04 17:43:35 UTC
Description

Automatically find the best vector autoregression models and networks for a given time series data set. 'AutovarCore' evaluates eight kinds of models: models with and without log transforming the data, lag 1 and lag 2 models, and models with and without weekday dummy variables. For each of these 8 model configurations, 'AutovarCore' evaluates all possible combinations for including outlier dummies (at 2.5x the standard deviation of the residuals) and retains the best model. Model evaluation includes the Eigenvalue stability test and a configurable set of residual tests. These
eight models are further reduced to four models because 'AutovarCore' determines whether adding weekday dummies improves the model fit.

Details

The DESCRIPTION file:

| Package: | autovarCore |
| Type: | Package |
| Title: | Automated Vector Autoregression Models and Networks |
| Version: | 1.0-4 |
| Date: | 2018-06-04 |
| Authors@R: | c(person("Ando","Emerencia",role = c("aut","cre"), email = "ando.emerencia@gmail.com")) |
| BugReports: | https://github.com/roqua/autovarcore/issues |
| Maintainer: | Ando Emerencia <ando.emerencia@gmail.com> |
| Description: | Automatically find the best vector autoregression models and networks for a given time series data set. 'AutovarCore' determines whether adding weekday dummies improves the model fit. |
| License: | MIT + file LICENSE |
| Suggests: | testthat, roxygen2 |
| Imports: | Rcpp (>= 0.11.4), Amelia, jsonlite, parallel, stats, urca, vars |
| LinkingTo: | Rcpp |
| RoxygenNote: | 6.0.1 |
| Author: | Ando Emerencia [aut, cre] |

Index of help topics:

- **apply_ln_transformation**: Applies the natural logarithm to the data set
- **assess_joint_sktest**: Tests the skewness and kurtosis of a VAR model
- **assess_kurtosis**: Tests the kurtosis of a VAR model
- **assess_portmanteau**: Tests the white noise assumption for a VAR model using a portmanteau test on the residuals
- **assess_portmanteau_squared**: Tests the homoskedasticity assumption for a VAR model using a portmanteau test on the squared residuals
- **assess_skewness**: Tests the skewness of a VAR model
- **autovar**: Return the best VAR models found for a time series data set
- **autovarCore-package**: Automated Vector Autoregression Models and Networks
- **coefficients_of_kurtosis**: Kurtosis coefficients.
- **coefficients_of_skewness**: Skewness coefficients.
- **compete**: Returns the winning model
- **day_dummies**: Calculate weekday dummy variables
- **daypart_dummies**: Calculate day-part dummy variables
- **explode_dummies**: Explode dummy variables columns into separate dummy
variables
impute_datamatrix  Imputes the missing values in the input data
invalid_mask    Calculate a bit mask to identify invalid outlier dummies
model_is_stable  Eigenvalue stability condition checking
model_score      Return the model fit for the given varest model
needs_trend      Determines if a trend is required for the specified VAR model
outliers_column  Determine the outliers column for the given column data
portmanteau_test_statistics  An implementation of the portmanteau test.
print_correlation_matrix     Print the correlation matrix of the residuals of a model annotated with p-values
residual_outliers  Calculate dummy variables to mask residual outliers
run_tests           Execute a series of model validity assumptions
run_var             Calculate the VAR model and apply restrictions
select_valid_masks  Select and return valid dummy outlier masks
selected_columns   Convert an outlier_mask to a vector of column indices
significance_from_pearson_coef  Calculate the significance of a Pearson correlation coefficient
sktest_joint_p      SK test p-level
trend_columns       Construct linear and quadratic trend columns
validate_params     Validates the params given to the autovar function
validate_raw_dataframe  Validates the dataframe given to the autovar function

Please see the help of the autovar function for information on how to use this package.

Author(s)
NA
Maintainer: Ando Emerencia <ando.emerencia@gmail.com>

References

See Also
autovar
**apply_ln_transformation**

**Applies the natural logarithm to the data set**

**Description**

This applies the ln function columnwise to the given input matrix and returns the modified matrix. If necessary, columns undergo a linear translation to ensure that all resulting values are \( \geq 0 \).

**Usage**

```r
apply_ln_transformation(data_matrix)
```

**Arguments**

- `data_matrix`: The original data matrix.

**Value**

The log-transformed data matrix.
Examples

data_matrix <- matrix(1:10, dimnames = list(NULL, 'some_val'))
data_matrix
autovarCore::apply_ln_transformation(data_matrix)

assess_joint_sktest Tests the skewness and kurtosis of a VAR model

Description

This function tests the joint skewness and kurtosis for the residuals of the endogenous variables in the specified VAR model. This function uses an implementation equivalent to STATA's sktest. Of the p-levels resulting from assessing the significance of the joint sktest for the residuals of that variable, the minimum is returned.

Usage

assess_joint_sktest(varest)

Arguments

varest A varest model.

Value

This function returns a p-level.

Examples

data_matrix <- matrix(nrow = 40, ncol = 3)
data_matrix[, , ] <- runif(ncol(data_matrix) * nrow(data_matrix), 1, nrow(data_matrix))
colnames(data_matrix) <- c('rumination', 'happiness', 'activity')
varest <- autovarCore::run_var(data_matrix, NULL, 1)
avtovarCore::assess_joint_sktest(varest)

assess_kurtosis Tests the kurtosis of a VAR model

Description

This function tests the kurtosis for the residuals of the endogenous variables in the specified VAR model. This function uses an implementation equivalent to STATA's sktest. Of the p-levels resulting from assessing the significance of the kurtosis for the residuals of that variable, the minimum is returned.
assess_portmanteau

Usage

assess_portmanteau(varest)

Arguments

varest A varest model.

Value

This function returns a p-level.

Examples

data_matrix <- matrix(nrow = 40, ncol = 3)
data_matrix[, ] <- runif(ncol(data_matrix) * nrow(data_matrix), 1, nrow(data_matrix))
colnames(data_matrix) <- c('rumination', 'happiness', 'activity')
varest <- autovarCore::run_var(data_matrix, NULL, 1)
autovarCore::assess_portmanteau(varest)
assess_portmanteau_squared

*Tests the homeskedasticity assumption for a VAR model using a port-
manteau test on the squared residuals*

**Description**

This function tests the homeskedasticity assumption for the residuals of the endogenous variables in the specified VAR model. This function implements the portmanteau squared test known as the Ljung-Box test, and results are comparable with STATA’s `wntestq`. Of the p-levels resulting from assessing the homeskedasticity assumption for the squared residuals of that variable, the minimum is returned.

**Usage**

`assess_portmanteau_squared(varest)`

**Arguments**

- `varest`: A `varest` model.

**Value**

This function returns a p-level.

**Examples**

```r
data_matrix <- matrix(nrow = 40, ncol = 3)
data_matrix[, ] <- runif(ncol(data_matrix) * nrow(data_matrix), 1, nrow(data_matrix))
colnames(data_matrix) <- c('rumination', 'happiness', 'activity')
varest <- autovarCore::run_var(data_matrix, NULL, 1)
autovarCore::assess_portmanteau_squared(varest)
```

assess_skewness

*Tests the skewness of a VAR model*

**Description**

This function tests the skewness for the residuals of the endogenous variables in the specified VAR model. This function uses an implementation equivalent to STATA’s `sktest`. Of the p-levels resulting from assessing the significance of the skewness for the residuals of that variable, the minimum is returned.

**Usage**

`assess_skewness(varest)`
Arguments

varest A varest model.

Value

This function returns a p-level.

Examples

data_matrix <- matrix(nrow = 40, ncol = 3)
data_matrix[, ] <- runif(ncol(data_matrix) * nrow(data_matrix), 1, nrow(data_matrix))
colnames(data_matrix) <- c('rumination', 'happiness', 'activity')
varest <- autovarCore::run_var(data_matrix, NULL, 1)
autovarCore::assess_skewness(varest)

Description

This function evaluates possible VAR models for the given time series data set and returns a sorted list of the best models found. The first item in this list is the "best model" found.

Usage

autovar(raw.dataframe, selected.column.names, significance.levels = c(0.05, 0.01, 0.005), test.names = c("portmanteau", "portmanteau.squared", "skewness"), criterion = "BIC", imputation.iterations = 100, measurements_per_day = 1)

Arguments

raw.dataframe The raw, unimputed data frame. This can include columns other than the selected.column.names, as those may be helpful for the imputation. The measurements in the dataframe are expected to be sorted by time, and to be sequential. Missed measurements should be encoded as rows of NA values and will be imputed by EM imputation.

selected.column.names The endogenous variables in the models, specified as a vector of character strings. This argument is required. The selected column names should be a subset of the column names of raw.dataframe.

significance.levels The significance levels used for evaluating the significance of the residual tests. The variable significance.levels is a vector with descending p values that indicate cut-offs placing models in different buckets. If it is not specified, this parameter defaults to c(0.05, 0.01, 0.005). More, fewer, and/or different significance levels can be specified. In practice, specifying more significance levels gives more weight to the outcomes of the residual tests, while having
fewer significance levels gives more weight to the AIC/BIC scores and the number of dummy variables. If a test for a model has a lower p-level than the minimum specified significance level, it can still be returned by the program, but it will be assigned the special significance bucket $\emptyset$.

test_names

The residual tests that should be performed, specified as a vector of character strings. If not specified, this parameter defaults to c('portmanteau', 'portmanteau_squared', 'skewness', 'kurtosis', 'joint_sktest'), which are used to test the assumptions of independence, homoscedasticity, and normality, respectively. The possible tests are c('portmanteau', 'portmanteau_squared', 'skewness', 'kurtosis', 'joint_sktest'). In addition to the residual tests, please note that the Eigenvalue stability test is always performed.

criterion

The information criterion used to sort the models. Valid options are 'BIC' (the default) or 'AIC'.

imputation_iterations

The amount of times the Amelia imputation should be averaged over. The default value for this parameter is 100. For details on the Amelia imputation, please see http://r.iq.harvard.edu/docs/amelia/amelia.pdf.

measurements_per_day

The number of measurements per day in the time series data. The default value for this parameter is 1. All models considered include day-part dummy variables if there are multiple measurements per day. If this value is 0, then daypart- and weekday dummy variables are not included for any models.

Details

AutovarCore evaluates eight kinds of VAR models: models with and without log transforming the data, lag 1 and lag 2 models, and with and without weekday dummy variables. For the lag 2 models, all cross-lagged relations are constrained. For each of these 8 model configurations, we evaluate all possible combinations for including outlier dummies (at 2.5x the standard deviation of the residuals) and retain the best model (the procedure for selecting the best model is described in more detail below).

These eight models are further reduced to four models by determining whether adding weekday dummies improves the model fit. AutovarCore does so based first on the significance bucket (determined by the outcomes of the residual tests) and secondly on the AIC/BIC score. If the best model found with weekday dummies is a "better model" than the best model found without weekday dummies, then AutovarCore includes the model with weekday dummies and discards the one without weekday dummies. Otherwise, AutovarCore includes only the model without weekday dummies and discards the one with weekday dummies. Thus, the comparison between models with and without weekday dummies is based on two steps:

1. We first consider the significance bucket. If the two models being compared are in different significance buckets, AutovarCore chooses the one with the highest significance bucket, and otherwise proceeds to step 2. The significance buckets are formed between each of the (decreasingly sorted) specified significance_levels in the parameters to the autovar function call. For example, if the significance_levels are c(0.05, 0.01, 0.005), then the significance buckets are (0.05 <= x), (0.01 <= x < 0.05), (0.005 <= x < 0.01), and (x < 0.005). The metric used to place a model into a bucket is the maximum p-level at which all residual tests will still pass ("pass" meaning not invalidating the model assumptions of independence, homoscedasticity, and normality).
2. When the significance bucket for the two models being compared is the same, AutovarCore selects the model with the lowest AIC/BIC score. Whether the AIC or BIC is used here depends on the criterion option specified in the parameters to the autovar function call.

The result of this procedure is four models: models with and without log transforming the data, and lag 1 and lag 2 models. Next, AutovarCore will determine whether the models with lag 1 or the models with lag 2 are best, and sort the models accordingly. This comparison is again based firstly on the significance bucket. If the significance bucket is the same, it proceeds to the next step, which in this case is the number of outlier dummy variables; the model with the fewest outlier dummy variables is considered the best. If the number of outlier dummy variables is the same, it proceeds to the third step, in which AutovarCore prefers the model with the lowest AIC/BIC score. This procedure results in two sorted lists of models, one list with models without logtransformation, one list with models with logtransformation.

In the final step, AutovarCore merges the sorted lists of models with and without logtransformation. To this end, it first compares the best model of the set without logtransformation with the best logtransformed model. It will sort these models based on the significance bucket first and the AIC/BIC score secondly. After finding the best model, it is removed from its list, and the then-best models are compared. This process repeats itself until both lists are empty. The result of this procedure is a final sorted list of four models (with the best model listed first).

The reason for the different sorting algorithms is that in some cases we want to select the model with the fewest outlier dummy columns (i.e., the model that retains most of the original data), while in other cases we know that a certain operation (such as adding weekday dummies or logtransforming the data set) will affect the amount of dummies in the model and so a fair comparison would exclude this property. For example, we do not compare the number of outlier columns in the final step because this would have likely favored logtransformed models over models without logtransformation, as logtransformations typically have the effect of reducing the outliers of a sample.

Outliers are, for each variable, the measurements at >2.5 times the standard deviation away from the mean of the residuals or of the squared residuals. Outlier dummy variables are split up such that each time point that is considered an outlier has its own dummy outlier variable and adds one to the count of outlier columns. Checks are in place to ensure that a time point identified as an outlier by multiple variables only adds a single dummy outlier column to the equation. For the count of outlier columns, day-part dummies do not add to the count. This is because when they are included, they are included for each model and thus never have any discriminatory power.

We are able to compare the AIC/BIC scores of logtransformed and nonlogtransformed models fairly because we compensate the AIC/BIC scores to account for the effect of the logtransformation. We compensate for the logtransformation by adjusting the loglikelihood score of the logtransformed models in the calculation of their AIC/BIC scores (by subtracting the sum of the logtransformed data).

**Value**

A sorted list of the best models found. A "model" is a list with the properties logtransformed, lag, varest, model_score, bucket, and nr_dummy_variables. The number of models returned is at most four. In rare cases, where the Eigenvalue stability test fails for multiple models, a list with fewer than four models is returned. When the Eigenvalue test fails for all tested models (which is unlikely to happen in practice), an empty list() is returned.
Examples

```r
# Not run:
data_matrix <- matrix(ncol = 3)
data_matrix[, ] <- runif(ncol(data_matrix) * nrow(data_matrix), 1, nrow(data_matrix))
while (sum(is.na(data_matrix)) == 0)
data_matrix[as.logical(round(runif(ncol(data_matrix) * nrow(data_matrix), -0.3, 0.7)))] <- NA
colnames(data_matrix) <- c('rumination', 'happiness', 'activity')
dataframe <- as.data.frame(data_matrix)
autovar(dataframe, selected_column_names = c('rumination', 'happiness'),
significance_levels = c(0.05, 0.01, 0.005),
test_names = c('portmanteau',
               'portmanteau_squared',
               'skewness'),
criterion = 'AIC',
imputation_iterations = 100,
measurements_per_day = 1)
```

## End(Not run)

---

**coefficients_of_kurtosis**

Kurtosis coefficients.

### Description
Kurtosis coefficients.

### Usage

```r
coefficients_of_kurtosis(matrix)
```

### Arguments
- **matrix**: the matrix of residuals.

---

**coefficients_of_skewness**

Skewness coefficients.

### Description
Skewness coefficients.

### Usage

```r
coefficients_of_skewness(matrix)
```
compete

Arguments

matrix the matrix of residuals.

Returns the winning model

Description

This function returns the best model as explained in the documentation for the autovar function.

Usage

compete(best, challenger, compare_outliers)

Arguments

best A model given as a list with at least the properties model_score, nr_dummy_variables, and bucket.

challenger Another model, also given as a list with properties model_score, nr_dummy_variables, and bucket.

compare_outliers A boolean. When FALSE, the model comparison does not take the number of dummy variables into account.

Value

This function returns the best model of the two given models.

Examples

```r
model1 <- list(logtransformed = FALSE, lag = 1, nr_dummy_variables = 1,
model_score = 100, bucket = 0.05)
model2 <- list(logtransformed = FALSE, lag = 2, nr_dummy_variables = 2,
model_score = 200, bucket = 0.01)
autovarCore:::compete(model1, model2, TRUE)
```
daypart_dummies  
*Calculate day-part dummy variables*

**Description**
This function returns either NULL (if `measurements_per_day` is 0 or 1) or a matrix of dummy variables for the specified input configuration.

**Usage**
```
daypart_dummies(number_of_rows, measurements_per_day)
```

**Arguments**
- `number_of_rows`  the number of rows in the input data set.
- `measurements_per_day`  the number of measurements per day in the input data set.

**Value**
Either NULL or a matrix with `number_of_rows` rows and `measurements_per_day - 1` columns.

**Examples**
```
autovarCore::daypart_dummies(10, 3)
```

---

day_dummies  
*Calculate weekday dummy variables*

**Description**
This function returns either NULL (if `measurements_per_day` is 0) or a matrix of weekday dummy variables specified number of rows and measurements per day. In the latter case, we return a matrix of six columns.

**Usage**
```
day_dummies(number_of_rows, measurements_per_day)
```

**Arguments**
- `number_of_rows`  the number of rows in the input data set.
- `measurements_per_day`  the number of measurements per day in the input data set.
Value

Either NULL or a matrix with number_of_rows rows and 6 columns.

Examples

autovarCore::day_dummies(16, 2)

---

`explode_dummies`  
Explode dummies columns into separate dummy variables

Description

This function takes a matrix with dummy outlier columns, where there are possibly multiple ones. We first merge these columns to one and then explode them to obtain one dummy variable per column.

Usage

`explode_dummies(outlier_dummies)`

Arguments

- `outlier_dummies`  
  A matrix of outlier dummy variables in columns.

Value

A matrix with dummy variables in columns, each having one nonzero index. The columns are named `outlier_x`, with x being the 1-based row index of the position that this dummy variable is masking.

Examples

```r
outlier_dummies <- matrix(NA, 
nrow = 5, 
ncol = 3, 
dimnames = list(NULL, c('rumination', 'happiness', 'activity'))) 
outlier_dummies[, 1] <- c(0, 0, 1, 0, 1) 
outlier_dummies[, 2] <- c(0, 1, 1, 0, 0) 
outlier_dummies[, 3] <- c(1, 0, 0, 0, 1) 
outlier_dummies 
autovarCore::explode_dummies(outlier_dummies)
```
impute_datamatrix  Imputes the missing values in the input data

Description

This function uses Amelia::amelia to impute missing (NA) values in the input data set. This function averages over multiple Amelia imputations to obtain more consistent results. The Amelia imputation model uses all variables of the supplied data_matrix, the first lag of those variables, time, time squared, and day-part dummies.

Usage

impute_datamatrix(data_matrix, measurements_per_day, imputation_iterations)

Arguments

data_matrix    The raw, unimputed data matrix.
measurements_per_day
    The number of measurements per day. This variable is used for adding day part dummy variables to aid the imputation.
imputation_iterations
    The amount of times the Amelia imputation should be averaged over.

Value

This function returns the modified matrix.

Examples

# create a matrix with some missing values
data_matrix <- matrix(nrow = 40, ncol = 3)
data_matrix[, ] <- runif(ncol(data_matrix) * nrow(data_matrix), 1, nrow(data_matrix))
while (sum(is.na(data_matrix)) == 0)
data_matrix[as.logical(round(runif(ncol(data_matrix) * nrow(data_matrix), -0.3, 0.7)))] <- NA
colnames(data_matrix) <- c('rumination', 'happiness', 'activity')
data_matrix
autovarCore:::impute_datamatrix(data_matrix, 1, 100)
### invalid_mask

**Calculate a bit mask to identify invalid outlier dummies**

**Description**

Invalid outlier dummy variables are dummy variables that are all zeros (where the original variable had no outliers at the 2.5x standard deviation for either the residuals or the squared residuals. Interpreting the leftmost column as bit 0 and continuing with higher bits going from left to right in the matrix, this function returns a bit mask that has a 1 on all positions in the matrix where the dummy column is invalid. We use this in later functions to easily filter out the invalid outlier masks from the valid ones.

**Usage**

```r
invalid_mask(outlier_dummies)
```

**Arguments**

- `outlier_dummies` - A matrix of outlier dummy variables in columns.

**Value**

An integer mask indicating the invalid columns according to the procedure describe above.

**Examples**

```r
resid_matrix <- matrix(rnorm(39 * 3),
    nrow = 39,
    ncol = 3,
    dimnames = list(NULL, c('rumination', 'happiness', 'activity')))
outlier_dummies <- autovarCore:::residual_outliers(resid_matrix, 40)
autovarCore:::invalid_mask(outlier_dummies)
```

### model_is_stable

**Eigenvalue stability condition checking**

**Description**

This function returns whether the given model satisfies the Eigenvalue stability condition. The Eigenvalue stability condition is satisfied when all eigenvalues lie in the unit circle.

**Usage**

```r
model_is_stable(varest)
```
model_score returns the model fit for the given varest model as either an AIC or BIC score. We compensating for logtransformation so that the model scores of logtransformed and non-logtransformed models can be compared with each other directly. This compensation is implemented by subtracting the logtransformed data from the log-likelihood score and using the result as log-likelihood score for the AIC/BIC calculations.

Usage

model_score(varest, criterion, logtransformed)

Arguments

varest A varest model.
criterion A character string being either 'AIC' or 'BIC'.
logtransformed A boolean, either TRUE or FALSE, indicating whether the input data for the model has been logtransformed.

Value

This returns a floating point that is either the AIC or BIC criterion for the model. A lower number corresponds to a better model fit.
needs_trend

Examples

data_matrix <- matrix(nrow = 40, ncol = 3)
data_matrix[, 1] <- runif(ncol(data_matrix) * nrow(data_matrix), 1, nrow(data_matrix))
colnames(data_matrix) <- c('rumination', 'happiness', 'activity')
varest <- autovarCore::run_var(data_matrix, NULL, 1)
autovarCore::model_score(varest, 'AIC', FALSE)

needs_trend

Determines if a trend is required for the specified VAR model

Description

This function uses the Phillips-Perron Unit Root Test to determine whether a trend is required for a VAR model based on the given matrix of endogenous variables and the given lag. All variables are assessed individually. This function returns TRUE if any of the endogenous variables requires a trend.

Usage

needs_trend(endo_matrix, lag)

Arguments

endo_matrix The matrix of endogenous variables in the model.
lag An integer specifying the lag length of the model.

Value

A boolean indicating whether a trend is required for the specified VAR model.

Examples

data_matrix <- matrix(nrow = 40, ncol = 3)
data_matrix[, 1] <- runif(ncol(data_matrix) * nrow(data_matrix), 1, 10)
data_matrix[, 3] <- (1:40) + rnorm(40)
colnames(data_matrix) <- c('rumination', 'happiness', 'activity')
data_matrix
autovarCore::needs_trend(data_matrix, 1)
outliers_column

Determine the outliers column for the given column data

Description

Determine the outliers column for the given column data

Usage

outliers_column(column_data, number_of_rows, std_factor)

Arguments

column_data The column with data.
number_of_rows The number of rows that the returned outliers column should have.
std_factor The factor multiplied with the standard deviation that determines the threshold for the distance away from the mean at which data points switch over to outliers.

portmanteau_test_statistics

An implementation of the portmanteau test.

Description

See the paper of Ljung-Box test for the used definition of autocorrelation.

Usage

portmanteau_test_statistics(matrix)

Arguments

class matrix the matrix of residuals or squared residuals.
**print_correlation_matrix**

*Print the correlation matrix of the residuals of a model annotated with p-values*

**Description**

This function prints the correlation matrix of residuals of a model annotated with p-values. This is a lower triangular matrix, in the way that all elements in the upper triangular matrix are NA and the elements on the "diagonal" are 1 (note that there is not really a diagonal because the matrix is rectangular). The odd rows of the returned matrix contain the correlations while the even rows are the associated p-values. For each correlation in row $x$, column $y$, its p-value is located in row $x+1$, column $y$.

**Usage**

```r
print_correlation_matrix(varest)
```

**Arguments**

- `varest`: A varest model.

**Value**

This function returns the annotated correlation matrix.

**Examples**

```r
data_matrix <- matrix(nrow = 40, ncol = 3)
data_matrix[, ] <- runif(ncol(data_matrix) * nrow(data_matrix), 1, nrow(data_matrix))
colnames(data_matrix) <- c('rumination', 'happiness', 'activity')
varest <- autovarCore::run_var(data_matrix, NULL, 1)
autovarCore::print_correlation_matrix(varest)
```

**residual_outliers**

*Calculate dummy variables to mask residual outliers*

**Description**

This function returns a matrix with columns that have a 1 at indices where the residuals have an outlier, and a 0 everywhere else. Outliers are calculated per variable (column) separately. We consider residual outliers the rows in the column of residuals or in the column of squared residuals that are more than 2.5 times the standard deviation away from the mean (standard deviation and mean are calculated separately per column and for residuals/squared residuals). The dummy columns are prepended with zeros to match the size of the other input variables to the model.
Usage

residual_outliers(resid_matrix, number_of_rows)

Arguments

resid_matrix A matrix of residuals. Column names are copied over to the returned result.
number_of_rows The number of measurements that were input to the model. Since the length of
the residual matrix is shorter depending on the amount of lags in the model, we
use number_of_rows to specify the number of rows in the returned matrix.

Value

A matrix with dummy variables in columns following the procedure described above.

Examples

resid_matrix <- matrix(rnorm(39 * 3),
  nrow = 39,
  ncol = 3,
  dimnames = list(NULL, c('rumination', 'happiness', 'activity')))
resid_matrix[13, 2] <- 48
resid_matrix[23, 2] <- -62
resid_matrix[36, 2] <- 33
resid_matrix[27, 3] <- 75
resid_matrix

autovarCore::residual_outliers(resid_matrix, 40)

run_tests

Execute a series of model validity assumptions

Description

This function returns the given suite of tests on for the given VAR model. For each test, the result is
the minimum p-level of all the assumptions and p-levels checked within the test. In other words, the
result of a test is the p-level that should be used as a threshold below which outcomes are considered
statistically significant (e.g., a result of 0.06 is better than a result of 0.03). The run_tests function
returns a vector of results, one for each test, in the order corresponding to the test_names argument.

Usage

run_tests(varest, test_names)

Arguments

varest A varest model.
test_names A vector of names of tests given as character strings. Supported tests are speci-
fied in the autovarCore::supported_test_names() function.
**run_var**

**Value**

This function returns a vector of p-levels.

**Examples**

```r
data_matrix <- matrix(nrow = 40, ncol = 3)
data_matrix[, ] <- runif(nrow(data_matrix), 1, nrow(data_matrix))
colnames(data_matrix) <- c('rumination', 'happiness', 'activity')
varest <- autovarCore::run_var(data_matrix, NULL, 1)
autovarCore::run_tests(varest, 'portmanteau')
```

---

**Description**

This function calls the `vars::var` function to calculate the VAR model and applies restrictions if needed. We set the intercept to 1 for restricted equations because calculations go wrong otherwise (this is a bug in the vars library).

**Usage**

```r
run_var(endo_matrix, exo_matrix, lag)
```

**Arguments**

- **endo_matrix**: A numeric matrix of endogenous data.
- **exo_matrix**: Either NULL or a numeric matrix of exogenous data.
- **lag**: A nonnegative integer specifying the lag length of the model. Specifying 0 for the lag results in calculating a lag 1 model with all lag-1 terms restricted.

**Value**

A `varest` object with the VAR estimation result.

**Examples**

```r
endo_matrix <- matrix(rnorm(120), ncol = 2, nrow = 60,
                      dimnames = list(NULL, c("rumination", "activity")))
autovarCore::run_var(endo_matrix, NULL, 1)
```
select_valid_masks

`selected_columns(outlier_mask)`

### Description

This function returns an ordered vector of all the 1-toggled bits in the `outlier_mask` offset by 1.

### Usage

```
selected_columns(outlier_mask)
```

### Arguments

- `outlier_mask`: An integer representing the outlier mask.

### Value

A vector of column indices corresponding to the outlier mask.

### Examples

```
outlier_mask <- 7
autovarCore::selected_columns(outlier_mask)
```

select_valid_masks

`select_valid_masks(all_outlier_masks, invalid_mask)`

### Description

Valid dummy outlier masks are integers whose bitwise AND with the given `invalid_mask` is zero. This function returns the subset of integers of the given vector that does not share any bits with the given `invalid_mask`.

### Usage

```
select_valid_masks(all_outlier_masks, invalid_mask)
```

### Arguments

- `all_outlier_masks`: A vector of possible outlier masks (integers).
- `invalid_mask`: An integer encoding the invalid columns.

### Value

The valid outlier masks as a vector of integers.
Examples

```r
all_outlier_masks <- c(0, 1, 2, 3, 4, 5, 6, 7)
invalid_mask <- 1
autovarCore::select_valid_masks(all_outlier_masks, invalid_mask)
```

### significance_from_pearson_coef

*Calculate the significance of a Pearson correlation coefficient*

**Description**

Calculate the significance of a Pearson correlation coefficient

**Usage**

```r
significance_from_pearson_coef(p, n)
```

**Arguments**

- `p`: The Pearson coefficient.
- `n`: The degrees of freedom.

### sktest_joint_p

*SK test p-level*

**Description**

SK test p-level

**Usage**

```r
sktest_joint_p(Z1, Z2, n)
```

**Arguments**

- `Z1`: The Z score for skewness.
- `Z2`: The Z score for kurtosis.
- `n`: The number of rows in the residuals column.
trend_columns Construct linear and quadratic trend columns

Description
This function returns a matrix of linear and quadratic trends.

Usage
trend_columns(number_of_rows)

Arguments
number_of_rows the number of rows in the input data set.

Value
A matrix with number_of_rows rows and 2 columns, one for linear trends and one for quadratic trends.

Examples
autovarCore:::trend_columns(10)

validate_params Validates the params given to the autovar function

Description
This function uses a list of default params that may be overwritten the params argument. stop() errors are thrown when invalid params are supplied.

Usage
validate_params(data_matrix, params)

Arguments
data_matrix The raw, unimputed data matrix. This parameter is supplied so that we can verify the selected column names.
params A list with the following named entries:

• selected_column_names - The endogenous variables in the models, specified as a vector of character strings. This argument is required. The selected column names should be a subset of the column names of data_matrix.
validate_raw_dataframe

A list containing augmented params.

Examples

```r
data_matrix <- matrix(ncol = 3, nrow = 5)
data_matrix[, 1] <- 1
data_matrix[, 2] <- c(1, 3, 5, 6, 7)
data_matrix[, 3] <- c(1, 0, 1, NA, 1)
colnames(data_matrix) <- c('id', 'tijdstip', 'home')
autovarCore::validate_params(data_matrix,
  list(selected_column_names = c('tijdstip', 'home'),
       imputation_iterations = 20))
```

---

validate_raw_dataframe

Validates the dataframe given to the autovar function

Description

This function returns the given data frame as a numeric matrix, using as.numeric to convert any columns in the data frame that are not numeric. A stop() error is thrown if there is not enough data in the data frame.
Usage

validate_raw_dataframe(raw_dataframe)

Arguments

raw_dataframe The raw, unimputed data frame.

Value

A numeric matrix with converted values and names taken from the data frame.

Examples

raw_dataframe <- data.frame(id = rep(1, times = 5),
                          tijdstip = c(1, 3, 5, 6, 7),
                          home = c(1, 0, 0, NA, 1))
autovarCore::validate_raw_dataframe(raw_dataframe)
Index

*Topic multivariate
  autovarCore-package, 2

*Topic package
  autovarCore-package, 2

*Topic regression
  autovarCore-package, 2

*Topic ts
  autovarCore-package, 2

apply_ln_transformation, 5
assess_joint_sktest, 6
assess_kurtosis, 6
assess_portmanteau, 7
assess_portmanteau_squared, 8
assess_skewness, 8
autovar, 4, 9
autovarCore (autovarCore-package), 2
autovarCore-package, 2

coefficients_of_kurtosis, 12
coefficients_of_skewness, 12
compete, 13
day_dummies, 14
daypart_dummies, 14
explote_dummies, 15
impute_datamatrix, 16
invalid_mask, 17
model_is_stable, 17
model_score, 18
needs_trend, 19
outliers_column, 20
portmanteau_test_statistics, 20
print_correlation_matrix, 21
residual_outliers, 21
run_tests, 22
run_var, 23
select_valid_masks, 24
selected_columns, 24
significance_from_pearson_coef, 25
sktest_joint_p, 25
trend_columns, 26
validate_params, 26
validate_raw_dataframe, 27