Package 'mvMonitoring'

February 2, 2017

Type Package

Title Multi-State Adaptive Dynamic Principle Component Analysis for Multivariate Process Monitoring

Version 0.1.0

Author Gabriel Odom, Ben Barnard, Karen Kazor, and Amanda Hering

Maintainer Gabriel Odom <gabriel_odom@baylor.edu>

Description Use multi-state splitting to apply Adaptive-Dynamic PCA to data generated from a continuous-time multivariate industrial or natural process. Employ PCA-based dimension reduction to extract linear combinations of relevant features, reducing computational burdens.

License file LICENSE

Depends R (>= 2.10)

Imports BMS, lazyeval, zoo, xts, utils

Encoding UTF-8

LazyData true

RoxygenNote 5.0.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

R topics documented:

`fault1A_xts` *Process Data under a System Shift Fault*

Description

Three-feature, three-state simulated process data including observations under normal operating conditions and observations after a positive shift for each feature in the system.

Usage

```
fault1A_xts
```

Format

An xts data matrix with 10080 rows and four columns, corresponding to one week worth of data recorded at a 1-minute interval. The columns under normal conditions are defined in the help file for normal_switch_xts. The fault is a system shock to each of the three features by 2. The fault starts at row 8500, and the four columns under the fault state are defined here:

- state : the state indicator for the multivariate system, with three levels
- x : x(t) = t + 2 + error
- y : y(t) = t ^ 2 - 3t + 2 + error
- z : z(t) = - t ^ 3 + 3t ^ 2 + 2 + error

where t is a 10080-entry vector of autocorrelated and non-stationary hidden process realizations. The states alternate each hour and are defined as follows:

- State1 – As presented
- State2 – Rotated by (yaw = 0, pitch = 90, roll = 30) and scaled by (1 * x, 0.5 * y , 2 * z).
- State3 – Rotated by (yaw = 90, pitch = 0, roll = -30) and scaled by (0.25 * x, 0.1 * y , 0.75 * z).

See the vignette for more details.

Source

Simluated in R.

fault1B_xts *Process Data under a Feature Shift Fault*

Description

Three-feature, three-state simulated process data including observations under normal operating conditions and observations after a positive shift in values for one feature.

Usage

```
fault1B_xts
```

Format

An xts data matrix with 10080 rows and four columns, corresponding to one week worth of data recorded at a 1-minute interval. The columns under normal conditions are defined in the help file for normal_switch_xts. The fault is a system shock to the "x" feature only by 2. The fault starts at row 8500, and the four columns under the fault state are defined here:

- state : the state indicator for the multivariate system, with three levels
- x : x(t) = t + 2 + error
- y : y(t) = t ^ 2 - 3t + error
- z : z(t) = - t ^ 3 + 3t ^ 2 + error

where t is a 10080-entry vector of autocorrelated and non-stationary hidden process realizations. The states alternate each hour and are defined as follows:

- State1 – As presented
- State2 – Rotated by (yaw = 0, pitch = 90, roll = 30) and scaled by (1 * x, 0.5 * y , 2 * z).
- State3 – Rotated by (yaw = 90, pitch = 0, roll = -30) and scaled by (0.25 * x, 0.1 * y , 0.75 * z).

See the vignette for more details.

Source

Simluated in R.

fault2A_xts *Process Data under a System Drift Fault*

Description

Three-feature, three-state simulated process data including observations under normal operating conditions and observations after a positive drift in values for each feature in the system.

Usage

```
fault2A_xts
```

Format

An xts data matrix with 10080 rows and four columns, corresponding to one week worth of data recorded at a 1-minute interval. The columns under normal conditions are defined in the help file for normal_switch_xts. The fault is a drift on each feature by s / 10 ^ 3, where s is the observation index. The fault starts at row 8500, and the four columns under the fault state are defined here:

- state : the state indicator for the multivariate system, with three levels
- x : x(t) = t + drift + error
- y : y(t) = t ^ 2 - 3t + drift + error
- z : z(t) = - t ^ 3 + 3t ^ 2 + drift + error

where t is a 10080-entry vector of autocorrelated and non-stationary hidden process realizations. The states alternate each hour and are defined as follows:

- State1 – As presented
- State2 – Rotated by (yaw = 0, pitch = 90, roll = 30) and scaled by (1 * x, 0.5 * y , 2 * z).
- State3 – Rotated by (yaw = 90, pitch = 0, roll = -30) and scaled by (0.25 * x, 0.1 * y , 0.75 * z).

See the vignette for more details.

Source

Simluated in R.

---

fault2B_xts *Process Data under a Feature Drift Fault*

Description

Three-feature, three-state simulated process data including observations under normal operating conditions and observations after a positive drift in values for two features.

Usage

```
fault2B_xts
```

Format

An xts data matrix with 10080 rows and four columns, corresponding to one week worth of data recorded at a 1-minute interval. The columns under normal conditions are defined in the help file for normal_switch_xts. The fault is a drift on the "y" and "z" features by s / 10 ^ 3, where s is the observation index. The fault starts at row 8500, and the four columns under the fault state are defined here:

- state : the state indicator for the multivariate system, with three levels
- x : x(t) = t + error
- y : y(t) = t ^ 2 - 3t + drift + error
- z : z(t) = - t ^ 3 + 3t ^ 2 + drift + error

where t is a 10080-entry vector of autocorrelated and non-stationary hidden process realizations. The states alternate each hour and are defined as follows:

- State1 – As presented
- State2 – Rotated by (yaw = 0, pitch = 90, roll = 30) and scaled by (1 * x, 0.5 * y , 2 * z).
- State3 – Rotated by (yaw = 90, pitch = 0, roll = -30) and scaled by (0.25 * x, 0.1 * y , 0.75 * z).

See the vignette for more details.

Source

Simluated in R.

`fault3A_xts` *Process Data under a System Signal Amplification*

Description

Three-feature, three-state simulated process data including observations under normal operating conditions and observations after an amplification of the underlying process for each feature in the system.

Usage

```
fault3A_xts
```

Format

An xts data matrix with 10080 rows and four columns, corresponding to one week worth of data recorded at a 1-minute interval. The columns under normal conditions are defined in the help file for normal_switch_xts. The fault is a signal amplificaton in the underlying determining t vector. The fault starts at row 8500, and the four columns under the fault state are defined here:

- state : the state indicator for the multivariate system, with three levels
- x : x(t_*) = t_* + error
- y : y(t_*) = (t_*) ^ 2 - 3t + error
- z : z(t_*) = - (t_*) ^ 3 + 3(t_*) ^ 2 + error

where t_* = 3 * t * (10080 - s) / (2 * 10080), where s is the observation index, and t is a 10080-entry vector of autocorrelated and non-stationary hidden process realizations. The states alternate each hour and are defined as follows:

- State1 – As presented
- State2 – Rotated by (yaw = 0, pitch = 90, roll = 30) and scaled by (1 * x, 0.5 * y , 2 * z).
- State3 – Rotated by (yaw = 90, pitch = 0, roll = -30) and scaled by (0.25 * x, 0.1 * y , 0.75 * z).

See the vignette for more details.

Source

Simluated in R.

`fault3B_xts` *Process Data under a Feature Signal Dampening*

Description

Three-feature, three-state simulated process data including observations under normal operating conditions and observations after a dampening of one of the underlying features in the system.

Usage

```
fault3B_xts
```

Format

An xts data matrix with 10080 rows and four columns, corresponding to one week worth of data recorded at a 1-minute interval. The columns under normal conditions are defined in the help file for normal_switch_xts. The fault is a signal dampening in the underlying determining t vector for the "z" feature. The fault starts at row 8500, and the four columns under the fault state are defined here:

- state : the state indicator for the multivariate system, with three levels
- x : x(t) = t + error
- y : y(t) = t ^ 2 - 3t + error
- z : z(t_*) = - (t_*) ^ 3 + 3(t_*) ^ 2 + error

where t_* = log|t|, and t is a 10080-entry vector of autocorrelated and non- stationary hidden process realizations. The states alternate each hour and are defined as follows:

- State1 – As presented
- State2 – Rotated by (yaw = 0, pitch = 90, roll = 30) and scaled by (1 * x, 0.5 * y , 2 * z).
- State3 – Rotated by (yaw = 90, pitch = 0, roll = -30) and scaled by (0.25 * x, 0.1 * y , 0.75 * z).

See the vignette for more details.

Source

Simluated in R.

`faultDetect` *Process Fault Detection*

Description

Detect if a single multivariate observation is beyond normal operating conditions.

Usage

```
faultDetect(threshold_object, observation, ...)
```

Arguments

`threshold_object`	An object of classes "threshold" and "pca" returned by the internal threshold() function.
`observation`	A single row of an xts data matrix (a 1 x p matrix) to compare against the thresholds
`...`	Lazy dots for additional internal arguments

Details

This function takes in a threshold object returned by the threshold() function and a single observation as a matrix or xts row. Internally, the function multiplies the observation by the projection matrix returned within the threshold object, calculates the SPE and T2 process monitoring statistics for that observation, and compares these statistics against their corresponding threshold values to determine if the observation lies outside the expected boundaries. The function then returns a row vector of the SPE test statistic, a logical indicator marking if this statistic is beyond the threshold, the Hotelling's T2 statistic, and an indicator if this statistic is beyond the threshold. Observations with monitoring statistics beyond the calculated threshold are marked with a 1, while observations within normal operating conditions are marked with a 0. These threshold values are passed from the threshold() function through this function via a returned threshold object. This object will be used in higher function calls.

This internal function is called by faultFilter().

Value

A named 1 x 4 matrix with the following entries for the single row observation passed to this function:

- SPE – the SPE statistic value
- SPE_Flag – the SPE fault indicator, where 1 represents a flag and 0 marks that the observation is within the normal operating conditions
- T2 – the T2 statistic value
- T2_Flag – the T2 fault indicator, defined the same as SPE_Flag

Examples

```
data("normal_switch_xts")
scaledData <- scale(normal_switch_xts[,-1])
pca_obj <- pca(scaledData, var.amnt = 0.9)
thresh_obj <- threshold(pca_object = pca_obj, alpha = 0.05)

# Check a single observation. We see this observation is within the normal
# operating conditions at alpha = 0.05.
faultDetect(threshold_object = thresh_obj, observation = scaledData[1,])

# According to the Squared Prediction Error statistic, this observation is
# outside the range of "normal" operation at the 0.05 level.
faultDetect(threshold_object = thresh_obj, observation = scaledData[20,])

# We can also check an entire data matrix:
detect_ls <- lapply(1:nrow(scaledData), function(i){
     faultDetect(threshold_object = thresh_obj, scaledData[i,])
})
do.call(rbind, detect_ls)
```

`faultFilter` *Process Fault Filtering*

Description

Flag and filter out observations beyond normal operating conditions, then return the observations within normal operating conditions.

Usage

```
faultFilter(trainData, testData, updateFreq, faultsToTriggerAlarm = 3, ...)
```

Arguments

`trainData`	An xts data matrix of initial training observations
`testData`	The data not included in the training data set
`updateFreq`	The number of observations from the test data matrix that must be returned to update the training data matrix and move it forward.
`faultsToTriggerAlarm`	Specifies how many sequential faults will cause an alarm to trigger. Defaults to 3.
`...`	Lazy dots for additional internal arguments

Details

This function is essentially a wrapper function to call and organize the output from these other internal functions: faultDetect(), threshold(), and pca(). It is applied over a rolling window, with observation width equal to updateFreq, of the larger full data matrix via the processMonitor() function, wherein the testing and training data sets move forward in time across the entire data matrix.

This internal function is called by processMonitor().

Value

A list of class "fault_ls" with the following:

- faultObj – an xts flagging matrix with the same number of rows as "testData". This flag matrix has the following five columns:
  - SPE – the SPE statistic value for each observation in "testData"
  - SPE_Flag – a vector of SPE indicators recording 0 if the test statistic is less than or equal to the critical value passed through from the threshold object
  - T2 – the T2 statistic value for each observation in "testData"
  - T2_Flag – a vector of T2 fault indicators, defined like SPE_Flag
  - Alarm – a column indicating if there have been three flags in a row for either the SPE or T2 monitoring statistics or both. Alarm states are as follows: 0 = no alarm, 1 = Hotelling's T2 alarm, 2 = Squared Prediction Error alarm, and 3 = both alarms.
- nonAlarmedTestObs – an xts matrix of the first updateFreq number of rows of the training data which were not alarmed
- trainSpecs – the threshold object returned by the internal threshold() function. See the threshold() function's help file for more details.

Examples

```
data("normal_switch_xts")
# Select the data under state 1
data <- normal_switch_xts[normal_switch_xts[,1] == 1]

# Split the data into testing and training data sets
nTrainObs <- floor(0.2 * nrow(data))
nUpdate <- floor(0.5 * nTrainObs)
trainObs <- data[1:nTrainObs, -1]
testObs <- data[(nTrainObs + 1):nrow(data), -1]

faultFilter(trainData = trainObs,
            testData = testObs,
            updateFreq = nUpdate)
```

`mspMonitor` *Real-Time Process Monitoring Function*

Description

Monitor and flag (if necessary) incoming multivariate process observations.

Usage

```
mspMonitor(observations, labelVector, trainingSummary, ...)
```

Arguments

observations	an n x p xts matrix. For real-time monitoring via a script within a batch file, n = 1, so this must be a 1 x p matrix. If lags were included at the training step, then these observations will also have lagged features.
labelVector	an n x 1 integer vector of class memberships
trainingSummary	the TrainingSpecs list returned by the mspTrain() function. This list contains—for each class—the SPE and T2 thresholds, as well the projection matrix.
...	Lazy dots for additional internal arguments

Details

This function is designed to be run at specific time intervals (e.g.every 10 seconds, 30 seconds, 1 minute, 5 minutes, 10 minutes) through a scheduled operating script which calls this function and mspWarning(). We expect this script to be set up in Windows "Task Scheduler" or Macintosh OX "launchd" application suites. This function takes in the specific observations to monitor and their class memberships (if any) and returns an xts matrix of these observation columns concatenated with their monitoring statistic values, flag statuses, and an empty alarm column. Users should then append these rows onto a previously existing matrix of daily observations. The mspWarning() function will then take in the daily observation xts matrix with updated rows returned by this function and check the monitoring statistic flag indicators to see if an alarm status has been reached. For further details, see the mspWarning() function.

This function calls the faultDetect() function, and requires the training information returned by the mspTrain function. This function will return the xts matrix necessary for the mspWarning() function.

Value

An n x (p + 5) xts matrix, where the last five columns are:

- SPE – the SPE statistic value for each observation in "observations"
- SPE_Flag – a vector of SPE indicators recording 0 if the test statistic is less than or equal to the critical value passed through from the threshold object
- T2 – the T2 statistic value for each observation in "observations"
- T2_Flag – a vector of T2 fault indicators, defined like SPE_Flag
- Alarm – a column indicating if there have been three flags in a row for either the SPE or T2 monitoring statistics or both. Alarm states are as follows: 0 = no alarm, 1 = Hotelling's T2 alarm, 2 = Squared Prediction Error alarm, and 3 = both alarms.

Examples

```
data("normal_switch_xts")
# The state values are recorded in the first column.
n <- nrow(normal_switch_xts)
nTrainObs <- floor(0.4 * n)

# Calculate the training summary, but save five observations for monitoring.
trainResults_ls <- mspTrain(data = normal_switch_xts[1:(n - 5), -1],
                            labelVector = normal_switch_xts[1:(n - 5), 1],
                            trainObs = nTrainObs,
                            lagsIncluded = c(0,1))
```

```
# While training, we included 1 lag (the default), so we will also lag the
# observations we will test.
testObs <- normal_switch_xts[(n - 6):n, -1]
testObs <- stats::lag(testObs, 0:1)
testObs <- testObs[-1, ]
testObs <- cbind(normal_switch_xts[(n - 5):n, 1], testObs)

mspMonitor(observations = testObs[, -1],
           labelVector = testObs[, 1],
           trainingSummary = trainResults_ls$TrainingSpecs)
```

`mspTrain` *Multi-State Adaptive-Dynamic Process Training*

Description

This function performs Multi-State Adaptive-Dynamic PCA on a data set with time-stamped observations.

Usage

```
mspTrain(data, labelVector, trainObs, updateFreq = ceiling(0.5 * trainObs),
  Dynamic = TRUE, lagsIncluded = c(0, 1), faultsToTriggerAlarm = 3, ...)
```

Arguments

`data`	An xts data matrix
`labelVector`	Class labels as a logical (two states only) or finite numeric (two or more states) vector with length equal to the number of rows in "data". For data with only one state, this will be a vector of 1s.
`trainObs`	The number of observations upon which to train the algorithm
`updateFreq`	The algorithm update frequency. Defaults to half as many observations as the training frequency.
`Dynamic`	Specify if the PCA algorithm should include lagged variables. Defaults to TRUE.
`lagsIncluded`	A vector of lags to include. If Dynamic = TRUE, specify which lags to include. Defaults to c(0, 1), signifying that the Dynamic process observations will include current observations and observations from one time step previous. See "Details" for more information.
`faultsToTriggerAlarm`	The number of sequential faults needed to trigger an alarm. Defaults to 3
`...`	Lazy dots for additional internal arguments

Details

This function is designed to identify and sort out sequences of observations which fall outside normal operating conditions. We assume that the process data are time-dependent in both seasonal and non-stationary effects (which necessitate the Adaptive and Dynamic components, respectively). We

further assume that this data is drawn from a multivariate process under multiple mutually exclusive states, implying that the linear dimension reduction projection matrices may be different for each state. Therefore, in summary, this function lags the features to account for correlation between sequential observations, splits the data by classes, and re-estimates projection matrices on a rolling window to account for seasonality. Further, this function uses non-parametric density estimation to calculate the 1 - alpha quantiles of the SPE and Hotelling's T2 statistics from a set of training observations, then flags any observation in the testing data set with process monitoring statistics beyond these calculated critical values. Becuase of natural variablity inherent in all real data, we do not remove observations simply because they are have been flagged as outside normal operating conditions. This function records an alarm only for observations having three flags in a row, as set by the default argument value of "faultsToTriggerAlarm". These alarm-positive observations are then removed from the data set and held in a separate xts matrix for inspection.

Concering the lagsIncluded variable: the argument lagsIncluded = c(0,1) will column concatenate the current data with the same data from one discrete time step back. This will necesarily remove the first row of the data matrix, as we will have NA values under the lagged features. The argument lagsIncluded = 0:2 will column concatenate the current observations with the observations from one step previous and the observations from two steps previous, which will necessarily requirethe removal of the first two rows of the data matrix. To include only certain lags with the current data, specify lagsIncluded = c(0, lag_1, lag_2, ... , lag_K). This induce NA values in the first max(lag_k) rows, for k = 1, ... , K, and these rows will be removed from consideration.

Of note when considering performance: the example has 10080 rows on three features alternating between three states, and trains on 20 percent of the observations, while updating every 1008 (10 percent) observation. On a 2016 Macbook Pro with 16Gb of RAM, this example function call takes 15 second to run. Increasing the update frequency will decrease computation time, but may increase false alarm rates or decrease flagging accuracy. We recommend that you set the update frequency based on the natural and physical designs of your system. For example, if your system has a multi-state process which switches across one of four states every two hours, then test the update frequency at an eight or 12 hour level — enough observations to measure two to three full cycles of the switching process. For observations recorded every five minutes, try updateFreq = (60 / 5) * 8 = 96 or (60 / 5) * 12 = 144.

This user-facing function calls the processMonitor() function, and returns the training arguments necessary to call the mspMonitor() and mspWarning() functions.

For more details, see Kazor et al (2016):

http://link.springer.com/article/10.1007/s00477-016-1246-2

Value

A list with the following components:

- FaultChecks – an xts flagging matrix with the same number of rows as "data". This flag matrix has the following five columns:
  - SPE – the SPE statistic value for each observation in "data"
  - SPE_Flag – a vector of SPE indicators recording 0 if the test statistic is less than or equal to the critical value passed through from the threshold object
  - T2 – the T2 statistic value for each observation in "data"
  - T2_Flag – a vector of T2 fault indicators, defined like SPE_Flag
  - Alarm – a column indicating if there have been three flags in a row for either the SPE or T2 monitoring statistics or both. Alarm states are as follows: 0 = no alarm, 1 = Hotelling's T2 alarm, 2 = Squared Prediction Error alarm, and 3 = both alarms.
- Non_Alarmed_Obs – an xts data matrix of all the non-alarmed observations

- Alarms – an xts data matrix of the features and specific alarms for Alarmed observations with the alarm codes are listed above
- TrainingSpecs – a list of k lists, one for each class, with each list containing the specific threshold object returned by the internal threshold() function for that class. See the threshold() function's help file for more details.

Examples

```
data("normal_switch_xts")
nTrainObs <- floor(0.4 * nrow(normal_switch_xts))
# The state values are recorded in the first column.

mspTrain(data = normal_switch_xts[, -1],
         labelVector = normal_switch_xts[, 1],
         trainObs = nTrainObs)
```

`mspWarning` *Process Alarms*

Description

Trigger an alarm, if necessary, for incoming multivariate process observations.

Usage

```
mspWarning(mspMonitor_object, faultsToTriggerAlarm = 3)
```

Arguments

`mspMonitor_object`
: An xts matrix returned by the mspMonitor() function

`faultsToTriggerAlarm`
: Specifies how many sequential faults will cause an alarm to trigger. Defaults to 3.

Details

This function and the mspMonitor() function are designed to be ran via a scheduled task through Windows "Task Scheduler" or Macintosh OX "launchd" application suites. The file flow is as follows: at each time interval, run the mspMonitor() function on the matrix of daily observations to add a flag status to the most recent incoming observation in the matrix, and return this new xts matrix. Then, pass this updated daily observation matrix to the mspWarning() function, which will check if the process has recorded three or more sequential monitoring statistic flags in a row. Of note, because these functions are expected to be repeatedly called in real time, this function will only check for an alarm within the last row of the xts matrix. To check multiple rows for an alarm state, please use the mspTrain() function, which was designed to check multiple past observations.

This function requires an xts matrix returned by the mspMonitor() function.

Value

An xts matrix of the same dimensions as mspMonitor_object, with a recorded negative or positive and type-specific alarm status. Alarm codes are: 0 = no alarm, 1 = Hotelling's T2 alarm, 2 = Squared Prediction Error alarm, and 3 = both alarms.

Examples

```
data("normal_switch_xts")
# The state values are recorded in the first column.
n <- nrow(normal_switch_xts)
nTrainObs <- floor(0.4 * n)

# Calculate the training summary, but save five observations for monitoring.
trainResults_ls <- mspTrain(data = normal_switch_xts[1:(n - 5), -1],
                            labelVector = normal_switch_xts[1:(n - 5), 1],
                            trainObs = nTrainObs,
                            lagsIncluded = c(0,1))

# While training, we included 1 lag (the default), so we will also lag the
# observations we will test.
testObs <- normal_switch_xts[(n - 6):n, -1]
testObs <- stats::lag(testObs, 0:1)
testObs <- testObs[-1, ]
testObs <- cbind(normal_switch_xts[(n - 5):n, 1], testObs)

# Run the monitoring function.
dataAndFlags <- mspMonitor(observations = testObs[, -1],
                           labelVector = testObs[, 1],
                           trainingSummary = trainResults_ls$TrainingSpecs)

# Alarm check the last row of the matrix returned by the mspMonitor function
mspWarning(dataAndFlags)
```

---

normal_switch_xts *Process Data under Normal Conditions*

Description

Three-feature, three-state simulated process data under normal operating conditions as example data for different included functions.

Usage

```
normal_switch_xts
```

Format

An xts data matrix with 10080 rows and four columns, corresponding to one week worth of data recorded at a 1-minute interval, and four columns as defined here:

- state – the state indicator for the multivariate system, with three levels
- x : x(t) = t + error
- y : y(t) = t ^ 2 - 3t + error
- z : z(t) = - t ^ 3 + 3t ^ 2 + error

where t is a 10080-entry vector of autocorrelated and non-stationary hidden process realizations. The states alternate each hour and are defined as follows:

- State1 – As presented

- State2 – Rotated by (yaw = 0, pitch = 90, roll = 30) and scaled by (1 * x, 0.5 * y , 2 * z).
- State3 – Rotated by (yaw = 90, pitch = 0, roll = -30) and scaled by (0.25 * x, 0.1 * y , 0.75 * z).

See the vignette for more details.

Source

Simluated in R.

`pca` *PCA for Data Scatter Matrix*

Description

Calculate the principal component analysis for a data matrix, and also find the squared prediction error (SPE) and Hotelling's T2 test statistic values for each observation in this data matrix.

Usage

```
pca(data, var.amnt = 0.95, ...)
```

Arguments

`data`	A centred-and-scaled data matrix or xts matrix
`var.amnt`	The energy proportion to preserve in the projection, which dictates the number of principal components to keep. Defaults to 0.95.
`...`	Lazy dots for additional internal arguments

Details

This function takes in a training data matrix, without the label column, and the energy preservation proportion, which defaults to 95 percent per Kazor et al (2016). This proportion is the sum of the q largest eigenvalues divided by the sum of all p eigenvalues, where q is the number of columns of the p x q projection matrix P. This function then returns the projection matrix P, a diagonal matrix of the reciprocal eigenvalues (LambdaInv), a vector of the SPE test statistic values corresponding to the rows of the data matrix, and a T2 test statistic vector similar to the SPE vector.

This internal function is called by faultFilter().

Value

A list of class "pca" with the following:

- projectionMatrix – the q eigenvectors corresponding to the q largest eigenvalues as a p x q projection matrix
- LambdaInv – the diagonal matrix of inverse eigenvalues
- SPE – the vector of SPE test statistic values for each of the n observations contained in "data"
- T2 – the vector of Hotelling's T2 test statistic for each of the same n observations

Examples

```
data("normal_switch_xts")
scaledData <- scale(normal_switch_xts[,-1])
pca(scaledData, var.amnt = 0.9)
```

`processMonitor` *Adaptive Process Training*

Description

Apply Adaptive-Dynamic PCA to state-specific data matrices.

Usage

```
processMonitor(data, trainObs, updateFreq = ceiling(0.5 * trainObs),
  faultsToTriggerAlarm = 3, ...)
```

Arguments

`data`	An xts data matrix
`trainObs`	The number of training observations to be used
`updateFreq`	The number of non-flagged observations to collect before the function updates. Defaults to half as many observations as the number of training observations.
`faultsToTriggerAlarm`	The number of sequential faults needed to trigger an alarm. Defaults to 3.
`...`	Lazy dots for additional internal arguments

Details

This function is the class-specific implementation of the Adaptive- Dynamic PCA described in the details of the mspTrain() function. See the mspTrain() function's help file for further details.

This internal function is called by mspTrain(). This function calls the faultFilter() function.

Value

A list with the following components:

- FaultChecks – a class-specific xts flagging matrix with the same number of rows as "data". This flag matrix has the following five columns:
  - SPE – the SPE statistic value for each observation in "data"
  - SPE_Flag – a vector of SPE indicators recording 0 if the test statistic is less than or equal to the critical value passed through from the threshold object
  - T2 – the T2 statistic value for each observation in "data"
  - T2_Flag – a vector of T2 fault indicators, defined like SPE_Flag
  - Alarm – a column indicating if there have been three flags in a row for either the SPE or T2 monitoring statistics or both. Alarm states are as follows: 0 = no alarm, 1 = Hotelling's T2 alarm, 2 = Squared Prediction Error alarm, and 3 = both alarms.
- Non_Alarmed_Obs – a class-specific xts data matrix of all the non-alarmed observations (observations with alarm state equal to 0)

- Alarms – a class-specific xts data matrix of the features and specific alarms of Alarmed observations, where the alarm codes are listed above
- trainSpecs – a threshold object returned by the internal threshold() function. See the threshold() function's help file for more details.

Examples

```
data("normal_switch_xts")
# Select the data under state 1
data <- normal_switch_xts[normal_switch_xts[,1] == 1]
nTrainObs <- floor(0.4 * nrow(data))

# Remove the now unnecesary state column
featureCols <- data[,-1]

processMonitor(data = featureCols, trainObs = nTrainObs)
```

`rotate3D` *Three-Dimensional Rotation Matrix*

Description

Render a 3-Dimensional projection matrix given positive or negative degree changes in yaw, pitch, and / or roll.

Usage

```
rotate3D(yaw, pitch, roll)
```

Arguments

`yaw`	z-axis change in degrees; look left (+) or right (-). Consider this a rotation on the x-y plane.
`pitch`	y-axis change in degrees; look up (-) or down (+). Consider this a rotation on the x-z plane.
`roll`	x-axis change in degrees; this change appears as if you touch head to shoulders: right roll (+) and left roll (-).

Details

When plotting with the package scatterplot3d, the default perpective is such that the pitch action appears as a roll while the roll action appears as a pitch.

This function is used only in data generation of the package vignette. This function is called by rotateScale3D().

Value

A 3 x 3 projection matrix corresponding to the degree changes entered.

Examples

```
data("normal_switch_xts")
normal_switch_xts[,-1] %*% rotate3D(yaw = -10, pitch = 0, roll = 15)
```

rotateScale3D *Three-Dimensional Rotation and Scaling Matrix*

Description

Render a 3-Dimensional projection matrix given positive or negative degree changes in yaw, pitch, and / or roll and increment or decrement feature scales.

Usage

```
rotateScale3D(rot_angles = c(0, 0, 0), scale_factors = c(1, 1, 1))
```

Arguments

rot_angles	a list or vector containg the rotation angles in the order following: yaw, pitch, roll. Defaults to <0,0,0>.
scale_factors	a list or vector containing the values by which to multiply each dimension. Defaults to <1,1,1>.

Details

See the help file of function rotate_3D() for a brief explination of how these angles behave in scatterplot3d functionality (from package scatterplot3d).

This function is used only in data generation of the package vignette. This function calls rotate3D().

Value

A 3 x 3 projection matrix corresponding to the degree and scale changes entered.

Examples

```
data("normal_switch_xts")
angles <- list(yaw = -10, pitch = 0, roll = 15)
featureScales <- c(0.2, 1, 5)
normal_switch_xts[,-1] %*% rotateScale3D(rot_angles = angles,
                                         scale_factors = featureScales)
```

threshold *Non-parametric Threshold Estimation*

Description

Calculate the non-parametric critical value threshold estimates for the SPE and T2 monitoring test statistics.

Usage

```
threshold(pca_object, alpha = 0.001, ...)
```

Arguments

pca_object	A list with class "pca" from the internal pca() function
alpha	The upper 1 - alpha quantile of the SPE and T2 densities from the training data passed to this function. Defaults to 0.001.
...	Lazy dots for additional internal arguments

Details

This function takes in a pca object returned by the pca() function and a threshold level defaulting to alpha = 0.1 percent of the observations. This critical quantile is set this low to reduce false alarms, as described in Kazor et al (2016). The function then returns a calculated SPE threshold corresponding to the 1 - alpha critical value, a similar T2 threshold, and the projection and Lambda Inverse (1 / eigenvalues) matrices passed through from the pca() function call.

This internal function is called by faultFilter().

Value

A list with classes "threshold" and "pca" containing:

- SPE_threshold – the 1 - alpha quantile of the estimated SPE density
- T2_threshold – the 1 - alpha quantile of the estimated Hotelling's T2 density
- projectionMatrix – a projection matrix from the data feature space to the feature subspace which preserves some pre-specified proportion of the energy of the data scatter matrix. This pre-specified energy proportion is user supplied as the var.amnt argument in the pca() function. See the pca() function's help file for more details.
- LambdaInv – a diagonal matrix of the reciprocal eigenvalues of the data scatter matrix

Examples

```
data("normal_switch_xts")
scaledData <- scale(normal_switch_xts[,-1])
pca_obj <- pca(scaledData, var.amnt = 0.9)
threshold(pca_object = pca_obj, alpha = 0.05)
```

Index