Package 'monitor'

June 7, 2016

Title Vibration signal monitor for wind turbine generators

Version 0.1

URL https://github.com/hannea/monitor

Description A package for monitoring vibration signals from wind turbine generators. The data should include time stamps, and also have at least one observed load variable, such as generator speed or power. The package is developed as part of a master's thesis and should not be viewed as a stand-alone documentation.

Depends R (>= 3.1.2)

License GPL-3

Maintainer Hanne Lone Andersen <hanne.a@outlook.com>

Author Hanne Lone Andersen [aut, cre]

LazyData true

VignetteBuilder knitr

RoxygenNote 5.0.1

Imports dplyr, ggplot2

Suggests knitr, rmarkdown

R topics documented:

count_alarms_constant_var_faster
Count and plot alarms using a spline, assuming constant variance

Description

The function uses a spline to fit to `vib_training` against `load_training`. An alarm is generates when the test data fall outside a band of three sigma. The function generates a plot by default, showing the training data falling outside the 3 sigma band.

Usage

```
count_alarms_constant_var_faster(load_training, vib_training, knots, load_test,
  vib_test, show_figure = TRUE, load_min = 0)
```

Arguments

`load_training`	A vector containing the training data of the load variables such as power or generator speed.
`vib_training`	A vector containing the training data of the vibration signal.
`knots`	A vector containing the locations of the knots.
`load_test`	A vector with the test data of the load variable.
`vib_test`	A vector with the test data of the vibration signal.
`show_figure`	A logical vector indicating if a plot should be made.
`load_min`	The minimum load to be used.

Value

A count of the number of alarms, alarm rate, and residuals. Furthermore a plot is generated by default set by `show_figure`.

`count`	A count of the number of alarms in the test set.
`alarm_rate`	The alarm rate in the test set.
`residuals`	The residuals from the test set.

Examples

```
count_alarms_constant_var_faster(ltrain, vtrain, knots = c(20,25,27),
                                 ltest, vtest, load_min = 17)
```

`count_alarms_hetero_faster` *Count and plot alarms using a spline, assuming heterogeneous variance*

Description

Function that divides `load_training` into quantiles using `bins` as the number of quantiles. For each quantile the means and standard diviations are calculated. A spline is fitted to the means and two splines are fitted to the three sigma above and below the means, respectively. It generates alarms when the test data fall outside the band of three sigma. The function generates a plot by default, and can also show the summary of the quantiles if `show_summary = TRUE`.

Usage

```
count_alarms_hetero_faster(load_training, vib_training, knots, load_test,
  vib_test, show_figure = TRUE, show_summary = FALSE, bins = 160,
  load_min = 0)
```

Arguments

`load_training`	A vector containing the training data of the load variables such as power or generator speed.
`vib_training`	A vector containing the training data of the vibration signal.
`knots`	A vector containing the locations of the knots.
`load_test`	A vector with the test data.
`vib_test`	A vector with the test data.
`show_figure`	A logical vector indicating if a plot should be made.
`show_summary`	A logical vector stating whether the quantile summaries should be shown or not.
`bins`	The number of bins used.
`load_min`	The minimum load to be used.

Value

A count of the number of alarms, alarm rate, and residuals. Furthermore a plot is generated by default, set by `show_figure`.

`count`	A count of the number of alarms in the test set.
`alarm_rate`	The alarm rate in the test set.
`residuals`	The residuals in the test set.

Examples

```
count_alarms_hetero_var_faster(ltrain, vtrain, knots = c(20,25,27),
                               ltest, vtest, load_min = 17)
```

identify_WTs_func *Identifies the wind turbines if several unit IDs appear.*

Description

Identifies the wind turbines if there are several unit IDs, and returns the same data in a list containing data frames for each unit ID, and the time stamps are converted to POXIXct class.

Usage

```
identify_WTs_func(dataframe, id = names(dataframe[1]))
```

Arguments

dataframe	A data frame containing the data from condition monitoring. The data frame should contain a column with unit IDs.
id	The name as a character of the column with unit IDs.

Value

out	The data in dataframe sorted by id into a list containing data frames, one for each wind turbines, and the time stamps are converted to POSIXct class. A string is printed showing the number of wind turbines in dataframe.

Examples

```
identify_WTs_func(data, id = "UnitID")
```

interpolate_func_2mins *Interpolation function using 2 minute intervals.*

Description

The interpolation of a single wind turbine with an interval length of 2 minutes. If there are intervals larger than 10 minuttes missing data, no interpolation in done, and NAs are added.

Usage

```
interpolate_func_2mins(dataframe, var = "PowerActual")
```

Arguments

dataframe	A dataframe containing a single wind turbine case.
var	The variable to be interpolated in dataframe.

Value

A new dataframe similar to the existing where the var has been interpolated using 2 minute intervals.

out	A data frame with time stamps in the first column and the second column where the var has been interpolated using 2 minute intervals.

Examples

```
interpolate_func_2mins(data, var = "GeneratorSpeed")
```

---

interpolate_func_2mins_identified

Interpolation of identified data frame using 2 minute intervals.

Description

The interpolation of several wind turbines with an interval length of 2 minutes. If there are intervals larger than 10 minuttes missing data, no interpolation in done, and NAs are added.

Usage

```
interpolate_func_2mins_identified(dataframes, var = "PowerActual")
```

Arguments

dataframes	A list of data frames as a result of the identified function with several wind turbines.
var	The variable to be interpolated in dataframes.

Value

The output is a data frame:

out	A data frame with time stamps and where the var has been interpolated using 2 minute intervals.

Examples

```
interpolate_func_2mins(data, var = "GeneratorSpeed")
```

kalman_filter_arma *Kalman filter applied to a univariate stationary zero-mean ARMA process.*

Description

To get the Kalman filter to work the process should be written as a dynamic linear model. The ARMA process written as a dynamic linear model has the form, where $y_t = F\boldsymbol{x}_t$ is the observation equation and $\boldsymbol{x}_t = G\boldsymbol{x}_{t-1} + Hw_t$ is the state equation, and $Q = \text{Var}(Hw_t)$.

Usage

```
kalman_filter_arma(ts, F, G, Q, m0, C0)
```

Arguments

ts	A univariate time series with zero mean
F	The coefficient matrix in the observation equation, as shown above.
G	The matrix in the state equation as shown above.
Q	The variance matrix of the state equation.
m0	The initial value of $\boldsymbol{x}_t$.
C0	The initial value of the state variance.

Value

The return is a list of

innovations	The innovations.
sd	the standardised residuals.
y_predicted	the predicted values.

Examples

```
kalman_filter_arma(ts = data, F=F, G=G, Q=Q, m0=m0, C0=C0)
```

Kalman_filter_random_walk *Kalman filter applied to multivariate random walk.*

Description

Kalman filter applied to multivariate random walk.

Usage

```
Kalman_filter_random_walk(ts, F = c(1, 1, 1, 1), R = 0.1 * diag(4),
  Q = 0.1, m0 = 0, C0 = 1)
```

Arguments

ts	A matrix containing the data from a multivariate time series.
R	The covariance of the meassurement noise.
Q	The covariance of the state noise.
m0	Initial state.
C0	Initial covariance of the state process.

Value

Returns a list of

state_values	The state values.
state_covariances	The state covariances.

Examples

```
Kalman_filter_random_walk(data)
```

Index