Package ‘monitor’

May 16, 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.

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

Description

Count and plot alarms using a spline

Usage

```
count_alarms_constant_var_faster(load_training, vib_training, knots = NULL,
  load_test, vib_test, 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 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.
`load_min`	The minimum load used.

Value

A count of the number of alarms, alarm rate, and residuals. Further a plot is generated.

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 with heterogeneous variance.

Description

Count and plot alarms using a spline with heterogeneous variance.

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 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 if the quantile summaries should be shown.
`bins`	The number of bins used.
`load_min`	The minimum load used.

Value

A count of the number of alarms, alarm rate, and residuals. Further a plot is generated if `show_figure` = TRUE.

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

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 minutes 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 minutes intervals.

Examples

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

interpolate_func_2mins_identified

Interpolation of identified data frame using 2 minutes 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

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

Examples

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

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

Description

The dynamic linear model, is on 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 $Q = \mathrm{Var}(Hw_t)$.

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 innovations, the standardized residual process and 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	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

State values and state covariances.

Examples

```
Kalman_filter_random_walk(data)
```

Index