

# Package ‘signalextraction’

January 2, 2012

**Type** Package

**Title** Real-Time Signal Extraction (Direct Filter Approach)

**Version** 2.0.3

**Date** 2007-11-18

**Author** Marc Wildi & Marcel Dettling

**Maintainer** Marcel Dettling <dtli@zhaw.ch>

**Description** The Direct Filter Approach (DFA) provides efficient estimates of signals at the current boundary of time series in real-time. For that purpose, one-sided ARMA-filters are computed by minimizing customized error criteria. The DFA can be used for estimating either the level or turning-points of a series, knowing that both criteria are incongruent. In the context of real-time turning-point detection, various risk-profiles can be operationalized, which account for the speed and/or the reliability of the one-sided filter.

**Suggests** tcltk

**License** GPL (>= 2)

**Repository** CRAN

**Date/Publication** 2007-11-20 11:03:42

## R topics documented:

signalextraction-package . . . . .	2
coef.dfa . . . . .	2
dfa . . . . .	3
fit . . . . .	6
fitted.dfa . . . . .	7
outsamp . . . . .	8
plot.dfa . . . . .	9
x . . . . .	10

<b>Index</b>	<b>12</b>
--------------	-----------

---

signalextraction-package

*Direct Filter Approach*

---

### Description

The Direct Filter Approach (DFA) provides efficient estimates of signals at the current boundary of time series in real time. For that purpose, one-sided ARMA-filters are computed by minimizing customized error criteria. The DFA can be used for estimating either the level or turning-points of a series, owing to the fact that both criteria are incongruent and cannot be optimized simultaneously. In the context of real time turning-point detection, various risk profiles can be operationalized, which account for the speed and/or the reliability of the one-sided filter.

### Details

Package: signalextraction  
Type: Package  
Version: 2.0.0  
Date: 2007-03-08  
License: GPL 2.0 or newer

### Author(s)

Marc Wildi & Marcel Dettling, <[wia/dem]@zhwin.ch>

### References

Marc Wildi, *Real-Time Signal Extraction (Beyond Maximum Likelihood Principles)*, Springer. To appear in 2007.

---

coef.dfa

*ARMA-coefficients for the Direct Filter Approach's Output*

---

### Description

The one-sided filter that is obtained from `dfa` is translated into an ARMA(16,16)-model. These ARMA-coefficients are returned here.

### Usage

```
## S3 method for class 'dfa'  
coef(object, ...)
```

**Arguments**

object            An R object of `class` "dfa", typically the result of `dfa()`.  
...                Further arguments passed to or from other methods.

**Value**

A matrix with 2 columns of 16 entries each. The first column contains the AR-coefficients, whereas the second is filled with the MA-coefficients.

**Author(s)**

Marc Wildi & Marcel Dettling, <[wia/dem]@zhwin.ch>

**References**

Marc Wildi, *Real-Time Signal Extraction (Beyond Maximum Likelihood Principles)*, Springer. To appear in 2007.

**See Also**

[dfa](#)

**Examples**

```
data(fit)

## Instead of calling data(fit), one could run (time consuming)
## set.seed(21)
## data(x)
## fit <- dfa(x)

coef(fit)
plot(coef(fit))
```

---

dfa

*Direct Filter Approach*

---

**Description**

The Direct Filter Approach (DFA) provides efficient estimates of signals at the current boundary of time series in real time. For that purpose, one-sided ARMA-filters are computed by minimizing customized error criteria. The DFA can be used for estimating either the level or turning-points of a series, owing to the fact that that both criteria are incongruent and cannot be optimized simultaneously. In the context of real time turning-point detection, various risk profiles can be operationalized, which account for the speed and/or the reliability of the one-sided filter.

**Usage**

```
dfa(x, quart = FALSE, d = 0, pb = 1/14, sb = 1/7, tpfiler =
  FALSE, lambda = 3, expweight = 1.5, pbd = if((length(x)<100)) 1.08
  else 1.03, limamp = if(!tpfiler) 1.5 else 3, i2 = TRUE, n.loops =
  10, verbose = 1)
```

**Arguments**

x	Numerical vector, containing a time series with at least 61 monthly or 23 quarterly entries. Missing values are not allowed.
quart	Logical, describes whether argument x contains quarterly data. If so, set quart=TRUE, for monthly data the default value quart=FALSE is appropriate.
d	Numerical, defaults to 0. Corresponds to the integration order of the series. It may take the values {0, 1, 2,}. If the series is bounded, no differencing (d=0) is necessary. For trending series the optimization criterion is biased, see section 6.1 in Wildi (2007). Taking first order differences (d=1) in such a case enables to reestablish efficiency (unbiasedness) by a transformation of the pseudo-periodogram.
pb	Numerical, takes values between 0 and $\pi$ , defaults to $\pi/14$ and determines the pass band. Together with argument sb, it describes the shape of the symmetric transfer function.
sb	Numerical, takes values between 0 and $\pi$ , defaults to $\pi/7$ and determines the stop band. Together with argument pb, it describes the shape of the symmetric transfer function.
tpfiler	Logical, defaults to TRUE, which makes the dfa-routine a turning point filter. In this case, the two variables lambda and expweight account for speed and/or reliability of the filter, see section 5.3 in Wildi (2007). Else, if tpfiler=FALSE, the dfa-routine corresponds to a best level filter.
lambda	Numerical, defaults to 3, but only affects the output if tpfiler is set to TRUE. Argument lambda is a Lagrange parameter for phase restriction that corresponds to $\chi$ in formula 5.3 of Wildi (2007). Larger values induce smaller time delays, but noisier signal estimates. Typical values for lambda are in [1,20].
expweight	Numerical, defaults to 1.5, but only affects the output if tpfiler=TRUE. Reasonable values are in [0.5,2]. Argument expweight determines the shape of the frequency weighting function $W(\omega)$ in formula 5.3 of Wildi (2007). More precisely, $W(\omega) = abs(\omega)^{expweight}$ . Larger values emphasize the stop band of the filter. Therefore high-frequency components are damped more effectively. This results in increased reliability.
pbd	Numerical, takes values $> 1$ and defaults to 1.08 for series with less than 100 observations, and 1.03 else. This argument is a regularity constraint. Moduli of poles of the ARMA filter are constrained to exceed pbd. Larger values of pbd generally imply a smoother transfer function.
limamp	Numerical, defaults to 1.5 for turning point estimation, and to 3 for best level estimation. This argument is a regularity constraint that constrains the amplitude to be smaller than the product of limamp and A(0). It should be larger than 1,

and we have found that values between 1.5 and 2 are fine in the case of level approximation `tpfilter=FALSE`, whereas values between 3 and 5 are reasonable constraints in the case of turning point estimation.

<code>i2</code>	Logical, defaults to TRUE. If TRUE, it forces the time delay of the one-sided filter to vanish at frequency zero. In practice, it is often the case that the use of this constraint leads to better estimates, especially if turning points are of interest, i.e. if <code>tpfilter=TRUE</code> .
<code>n.loops</code>	Numerical, defaults to 10. It is the number of initial Stochastic Annealing proposals for the nonlinear optimization in computing the filter, of which the best is chosen and optimized further. A bigger number usually leads to a more precise solution, but the code takes longer to run.
<code>verbose</code>	Numerical, defaults to 1 and steers the amount of text output in the optimization routine. Allowed values are 0 (no output at all), 1 (minimal output for run-time control) and 2 (full output for tracking the optimization steps).

## Details

Please note that we here refer to Wildi (2007) for further details. Depending on how the arguments are set, the main function `dfa` minimizes one of the error criteria presented in the book. This is either the one given in formula 3.7 (for level estimation on stationary data: `d=0`, `tpfilter=FALSE`), 6.7 (for level estimation on non-stationary trending data: `d=1`, `tpfilter=TRUE`) or 5.4 (detection of turning points, `tpfilter=TRUE`). The level-criteria 3.7 and 6.7 mate the structure of the underlying estimation problem because one can show that they correspond to efficient estimates of the (unknown) mean-square real-time filter error. Criterion 5.4 is a generalization (for  $\chi=1$  and  $W(\omega)=1$  the preceding level criteria result) which enables to improve speed (smaller time delay in the pass-band) and reliability (better damping in the stop-band) of the filter. One can show that it improves level characteristics in the vicinity of turning points and that delays are weighted more heavily than anticipations, see sections 5.2 and 5.3 in Wildi (2007).

## Value

An object of class `dfa`, containing

<code>argli</code>	The function call in a list with values for all arguments
<code>xf</code>	The filtered real time series
<code>critval</code>	Corresponds to the minimized criterion value (either of the formulas 3.7, 5.4 or 6.7 in Wildi (2007), according to the arguments in the function call). In the case of level estimation, this value is also an estimate of the mean-square filter error (computed in the frequency domain).
<code>ar.coef</code>	AR-coefficients of the one-sided filter
<code>ma.coef</code>	MA-coefficients of the one-sided filter
<code>perall</code>	The periodogram (for <code>d=0</code> ) or pseudo-periodogram (for <code>d=1</code> ) of the input series
<code>amp</code>	Amplitude function of the one-sided real time filter
<code>pha</code>	Time delay (phase divided by frequency) of the one-sided real time filter

**Author(s)**

Marc Wildi & Marcel Dettling, <[wia/dem]@zhwin.ch>

**References**

Marc Wildi, *Real-Time Signal Extraction (Beyond Maximum Likelihood Principles)*, Springer. To appear in 2007.

**See Also**

[coef.dfa](#), [fitted.dfa](#), [plot.dfa](#), [outsamp](#)

**Examples**

```
data(fit)

## Instead of calling data(fit), one could run (time consuming)
## set.seed(21)
## data(x)
## fit <- dfa(x)

plot(fit)
```

---

fit

*Example of an object of class dfa*

---

**Description**

This is an example of an R object of [class "dfa"](#), i.e. the result of the assignment `fit <- dfa(x)` on the business survey dataset `x`. Its purpose is to enable illustrative and meaningful examples without running the CPU-time intensive optimization routine.

**Usage**

```
data(fit)
```

**Format**

An object of [class "dfa"](#). For an explanation of its contents, see the help file of [dfa](#).

**Source**

```
Obtained from set.seed(21); data(x); fit <- dfa(x)
```

**References**

Marc Wildi, *Real-Time Signal Extraction (Beyond Maximum Likelihood Principles)*, Springer. To appear in 2007.

**Examples**

```
data(fit)

## Instead of calling data(fit), one could run (time consuming)
## set.seed(21)
## data(x)
## fit <- dfa(x)

str(fit)
plot(fit)
```

---

fitted.dfa

*Fitted Values of the Direct Filter Approach's Output*

---

**Description**

This is the filtered real time series, i.e. the main output of the function [dfa](#).

**Usage**

```
## S3 method for class 'dfa'
fitted(object, ...)
```

**Arguments**

object	An R object of <code>class</code> "dfa", typically the result of <a href="#">dfa()</a> .
...	Further arguments passed to or from other methods

**Value**

Filter output

**Author(s)**

Marc Wildi & Marcel Dettling, <[\[wia/dem\]@zhwin.ch](mailto:[wia/dem]@zhwin.ch)>

**References**

Marc Wildi, *Real-Time Signal Extraction (Beyond Maximum Likelihood Principles)*, Springer. To appear in 2007.

**See Also**

[dfa](#)

**Examples**

```

data(fit)

## Instead of calling data(fit), one could run (time consuming)
## set.seed(21)
## data(x)
## fit <- dfa(x)

fitted(fit)
plot(fitted(fit), type="l")

```

---

outsamp

*Out of Sample Method for the Direct Filter Approach*


---

**Description**

Out of sample values based on the direct filter approach.

**Usage**

```
outsamp(object, newdata=NULL, sequel=TRUE, ...)
```

**Arguments**

object	An R object of <code>class</code> "dfa", typically the result of <code>dfa()</code> .
newdata	For out-of-sample values, this has to be a numerical vector. It can be of arbitrary length but has to contain further observations of the time series <code>x</code> . If <code>newdata</code> is omitted (and therefore equals <code>NULL</code> ), then the in-sample fitted values are returned.
sequel	Logical, defaults to <code>TRUE</code> and only has an effect if there are <code>newdata</code> . If set to <code>TRUE</code> , it is assumed that the new data are a direct sequel to the original series and thus, these original values are used for the asymmetric filter. If <code>sequel</code> is <code>FALSE</code> , then the observations prior to the start of the new series are estimated by a backcast.
...	Further arguments passed to or from other methods.

**Details**

This function yields a so-called nowcast for a time series, by applying the asymmetrical filter obtained from `dfa`. Initial values, at the beginning of a time series, are estimated by backcasting it. This function, `outsamp` can also be used to produce out-of-sample nowcasts, where the asymmetrical filter is applied to a time series that was not involved in its estimation.

**Value**

Output is a numerical vector of the same length as the input vector. It contains the filtered time series.

**Author(s)**

Marc Wildi & Marcel Dettling, <[wia/dem]@zhwin.ch>

**References**

Marc Wildi, *Real-Time Signal Extraction (Beyond Maximum Likelihood Principles)*, Springer. To appear in 2007.

**See Also**

[dfa](#)

**Examples**

```
data(fit)

## Instead of calling data(fit), one could run (time consuming)
## set.seed(21)
## data(x)
## fit <- dfa(x)

outsamp(fit)
plot(outsamp(fit))
plot(outsamp(fit, newdata=rnorm(100), sequel=FALSE))
```

---

plot.dfa

*Graphical Visualization of the Direct Filter Approach's Output*

---

**Description**

Yields a plot with 6 panels, i.e. an overlay of the original series with the filter output, and detailed plots of the filter output, periodogram of the input and output series, as well as the amplitude and time shift (delay) functions of the one-sided filter.

**Usage**

```
## S3 method for class 'dfa'
plot(x, ...)
```

**Arguments**

x                    An R object of class "dfa", typically the result of `dfa()`.  
...                   Further arguments passed to or from other methods.

**Details**

Please consult the reference given below for details about the interpretation of these plots.

**Author(s)**

Marc Wildi & Marcel Dettling, <[wia/dem]@zhwin.ch>

**References**

Marc Wildi, *Real-Time Signal Extraction (Beyond Maximum Likelihood Principles)*, Springer. To appear in 2007.

**See Also**

[dfa](#)

**Examples**

```
data(fit)

## Instead of calling data(fit), one could run (time consuming)
## set.seed(21)
## data(x)
## fit <- dfa(x)

str(fit)
plot(fit)
```

---

x

*Business Survey Data*

---

**Description**

In order to anticipate movements of the GDP growth rate the Swiss Institute of Business Cycle Research (KOF-ETH) conducts a survey based on a panel comprising around 7000 companies (manufacturing, construction, retail, hotel, catering, wholesale, banks). The questionnaire can be viewed at <http://survey.kof.ethz.ch>. The Business Survey Data that are provided here are a selection of the sample which is used for construction of the leading indicator (KOF-Konjunkturbarometer).

**Usage**

```
data(x)
```

**Format**

It is a named numerical vector, where the names describe the time of measurement.

**Source**

Swiss Institute for Business Cycle Research

**References**

Swiss Institute for Business Cycle Research, WEH D4, CH-8092 Zurich. Contact person: Richard Etter, <ind@kof.ethz.ch>. See also <http://www.kof.ethz.ch/>.

**Examples**

```
data(x)  
str(x)
```

# Index

- \*Topic **datasets**
  - fit, 6
  - x, 10
- \*Topic **hplot**
  - plot.dfa, 9
- \*Topic **package**
  - signalextraction-package, 2
- \*Topic **smooth**
  - dfa, 3
- \*Topic **ts**
  - coef.dfa, 2
  - dfa, 3
  - fitted.dfa, 7
  - outsamp, 8
  - plot.dfa, 9
  
- class, 3, 6–9
- coef.dfa, 2, 6
  
- dfa, 2, 3, 3, 6–10
  
- fit, 6
- fitted.dfa, 6, 7
  
- outsamp, 6, 8
  
- plot.dfa, 6, 9
  
- signalextraction
  - (signalextraction-package), 2
- signalextraction-package, 2
  
- x, 10